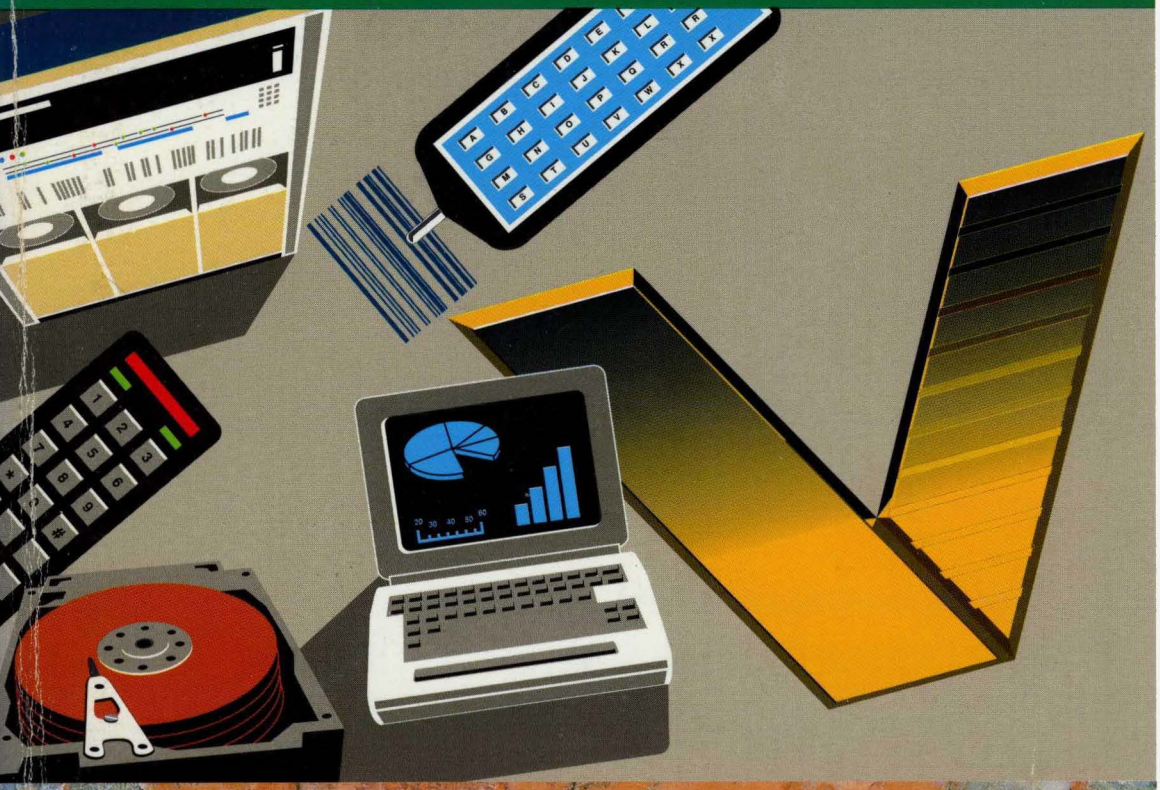


16-BIT V-SERIES MICROPROCESSOR DATA BOOK

NEC

NEC Electronics Inc.

NEC

1990
16-Bit V-Series Microprocessor Data Book

May 1990
Document No. 50054
©1990 NEC Electronics Inc./Printed in U.S.A.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics Inc. The information in this document is subject to change without notice. Devices sold by NEC Electronics Inc. are covered by the warranty and patent indemnification provisions appearing in NEC Electronics Inc. Terms and Conditions of Sale only. NEC Electronics Inc. makes no warranty, express, statutory, implied, or by description, regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. NEC Electronics Inc. makes no warranty of merchantability or fitness for any purpose. NEC Electronics Inc. assumes no responsibility for any errors that may appear in this document. NEC Electronics Inc. makes no commitment to update or to keep current the information contained in this document.

NEC

NEC

Contents

Section 1
Selection Guides

Section 2
Reliability and Quality Control

Section 2
Reliability and Quality Control (cont)

Section 3
16-Bit CPUs

Section 4
16-Bit Microcomputers

NEC

Section 4
16-Bit Microcomputers (cont)

Section 5
Peripherals for CPUs

Section 5
Peripherals for CPUs (cont)

Section 6
Development Tools

Section 7
Package Drawings

Numerical Index

NEC

Section 1
Selection Guides

Part Numbering System

μPD72001L Typical microdevice part number

μP	NEC monolithic silicon integrated circuit
D	Device type (D = digital MOS)
72001	Device identifier (alphanumeric)
L	Package type (L = PLCC)

A part number may include an alphanumeric suffix that identifies special device characteristics; for example, μPD72001L-11 has an 11-MHz data clock rating.

4-Bit, Single-Chip CMOS Microcomputers

Device, µPD	Features	Clock (MHz)	Supply Voltage (V)	ROM (X8)	RAM (X4)	I/O	# Package	Pins
7502	LCD controller/driver	0.41	2.5 to 6.0	2K	128	23	QFP	64
7503	LCD controller/driver	0.41	2.5 to 6.0	4K	224	23	QFP	64
7507	General-purpose	0.41	2.5 to 6.0	2K	128	32	DIP SDIP QFP	40 40 52
7507H	General-purpose	4.19	2.7 to 6.0	2K	128	32	DIP SDIP QFP	40 40 52
7508	General-purpose	0.41	2.5 to 6.0	4K	224	32	DIP SDIP QFP	40 40 52
7508H	General-purpose	4.19	2.7 to 6.0	4K	224	32	DIP SDIP QFP	40 40 52
75CG08	Piggyback EPROM	0.41	4.5 to 5.5	2K or 4K	224	32	Ceramic DIP	40
75CG08H	Piggyback EPROM	4.19	4.5 to 5.5	2K or 4K	224	32	Ceramic DIP	40
7527A	FIP controller/driver	0.61	2.7 to 6.0	2K	128	35	DIP SDIP	42 42
7528A	FIP controller/driver	0.61	2.7 to 6.0	4K	160	35	DIP SDIP	42 42
75CG28	Piggyback EPROM; FIP controller/driver	0.5	4.5 to 5.5	4K	160	35	Ceramic DIP	42
7533	A/D converter	0.5	2.7 to 6.0	4K	160	30	DIP SDIP QFP	42 42 44
75CG33	Piggyback EPROM; A/D converter	0.5	4.5 to 5.5	4K	160	30	Ceramic DIP	42
7537A	FIP controller/driver	0.61	2.7 to 6.0	2K	128	35	DIP SDIP	42 42
7538A	FIP controller/driver	0.61	2.7 to 6.0	4K	160	35	DIP SDIP	42 42
75CG38	Piggyback EPROM; FIP controller/driver	0.61	4.5 to 5.5	4K	160	35	Ceramic DIP	42
7554	Serial I/O; external clock or RC oscillator	0.71	2.5 to 6.0	1K	64	16	SDIP SOP	20 20
7554A	Serial I/O; external clock or RC oscillator	0.71	2.0 to 6.0	1K	64	16	SDIP SOP	20 20
75P54	Serial I/O; external clock or RC oscillator	0.71	4.5 to 6.0	1K OTPROM	64	16	SDIP SOP	20 20
7564/7564A	Serial I/O; ceramic oscillator	0.71	2.7 to 6.0	1K	64	15	SDIP SOP	20 20
75P64	Serial I/O; ceramic oscillator	0.71	4.5 to 6.0	1K OTPROM	64	15	SDIP SOP	20 20
7556	Comparator; external clock or RC oscillator	0.71	2.5 to 6.0	1K	64	20	SDIP SOP	24 24
7556A	Comparator; external clock or RC oscillator	0.71	2.0 to 6.0	1K	64	20	SDIP SOP	24 24
75P56	Comparator; external clock or RC oscillator	0.71	4.5 to 6.0	1K OTPROM	64	20	SDIP SOP	24 24
7566/7566A	Comparator; ceramic oscillator	0.71	2.7 to 6.0	1K	64	19	SDIP SOP	24 24
75P66	Comparator; ceramic oscillator	0.71	4.5 to 6.0	1K OTPROM	64	19	SDIP SOP	24 24
75004	General-purpose	4.19	2.7 to 6.0	4K	512	34	SDIP QFP	42 44

Plastic unless ceramic (or cerdip) is specified.
* Under development; consult Microcontroller Marketing for availability.

4-Bit, Single-Chip CMOS Microcomputers (cont)

Device, µPD	Features	Clock (MHz)	Supply Voltage (V)	ROM (X8)	RAM (X4)	I/O	# Package	Pins
75006	General-purpose	4.19	2.7 to 6.0	6K	512	34	SDIP QFP	42 44
75008	General-purpose	4.19	2.7 to 6.0	8K	512	34	SDIP QFP	42 44
75P008	General-purpose	4.19	4.5 to 5.5	8K OTPROM	512	34	SDIP QFP	42 44
75028 *	A/D converter	4.19	2.7 to 6.0	8K	512	48	SDIP QFP	64 64
75P036 *	A/D converter	4.19	2.7 to 6.0	16K	1024	48	SDIP QFP	64 64
75048 *	A/D converter; 1K x 4 EEPROM	4.19	2.7 to 6.0	8K	512	48	SDIP QFP	64 64
75P056 *	A/D converter; 1K x 4 EEPROM	4.19	2.7 to 6.0	16K	512	48	SDIP QFP	64 64
75104	High-end with 8-bit instruction	4.19	2.7 to 6.0	4K	320	58	SDIP QFP	64 64
75104A	High-end with 8-bit instruction	4.19	2.7 to 6.0	4K	320	58	QFP	64
75106	High-end with 8-bit instruction	4.19	2.7 to 6.0	6K	320	58	SDIP QFP	64 64
75108	High-end with 8-bit instruction	4.19	2.7 to 6.0	8K	512	58	SDIP QFP	64 64
75108A	High-end with 8-bit instruction	4.19	2.7 to 6.0	8K	512	58	QFP QFP	64 64
75P108	High-end with 8-bit instruction; on-chip OTPROM or UVEPROM	4.19	4.5 to 5.5	8K	512	58	SDIP QFP Shrink cerdip	64 64 64
75P108B	High-end with 8-bit instruction; on-chip OTPROM	4.19	2.7 to 6.0	8K	512	58	SDIP QFP	64 64
75112	High-end with 8-bit instruction	4.19	2.7 to 6.0	12K	512	58	SDIP QFP	64 64
75116	High-end with 8-bit instruction	4.19	2.7 to 6.0	16K	512	58	SDIP QFP	64 64
75P116	High-end with 8-bit instruction on-chip OTPROM	4.19	4.5 to 5.5	16K OTPROM	512	58	SDIP QFP	64 64
75206	FIP controller/driver	4.19	2.7 to 6.0	6K	369	33	SDIP QFP	64 64
75208	FIP controller/driver	4.19	2.7 to 6.0	8K	497	33	SDIP QFP	64 64
75CG208	FIP controller/driver; piggyback EPROM	4.19	4.5 to 5.5	8K	512	33	Ceramic SDIP Ceramic QFP	64 64
75212A	FIP controller/driver	4.19	2.7 to 6.0	12K	512	33	SDIP QFP	64 64
75216A	FIP controller/driver	4.19	2.7 to 6.0	16K	512	33	SDIP QFP	64 64
75CG216A	FIP controller/driver; piggyback EPROM	4.19	4.5 to 5.5	16K	512	33	Ceramic SDIP Ceramic QFP	64 64
75P216A	FIP controller/driver	4.19	4.5 to 5.5	16K OTPROM	512	33	SDIP	64
75268	FIP controller/driver	4.19	2.7 to 6.0	8K	512	32	SDIP QFP	64 64
75304	LCD controller/driver	4.19	2.7 to 6.0	4K	512	40	QFP	80
75306	LCD controller/driver	4.19	2.7 to 6.0	6K	512	40	QFP	80
75308	LCD controller/driver	4.19	2.7 to 6.0	8K	512	40	QFP	80
75P308	LCD controller/driver; on-chip OTPROM or UVEPROM	4.19	4.75 to 5.25	8K	512	40	QFP Ceramic LCC	80 80

4-Bit, Single-Chip CMOS Microcomputers (cont)

Device, µPD	Features	Clock (MHz)	Supply Voltage (V)	ROM (X8)	RAM (X4)	I/O	# Package	Pins
75312	LCD controller/driver	4.19	2.7 to 6.0	12K	512	40	QFP	80
75316	LCD controller/driver	4.19	2.7 to 6.0	16K	512	40	QFP	80
75P316	LCD controller/driver; on-chip OTPROM	4.19	4.75 to 5.25	16K OTPROM	512	40	QFP	80
75P316A *	LCD controller/driver; on-chip OTPROM or UVEPROM	4.19	2.7 to 6.0	16K OTPROM	512	40	QFP Ceramic LCC	80 80
75328	LCD controller/driver; A/D converter	4.19	2.7 to 6.0	8K	512	44	QFP	80
75P328	LCD controller/driver; A/D converter	4.19	4.5 to 5.5	8K OTPROM	512	44	QFP	80
75402A	Low-end	4.19	2.7 to 6.0	2K	64	22	DIP SDIP QFP	28 28 44
75P402	Low-end	4.19	4.5 to 5.5	2K OTPROM	64	22	DIP SDIP QFP	28 28 44
75512	High-end; A/D converter	4.19	2.7 to 6.0	12K	512	64	QFP	80
75516	High-end; A/D converter	4.19	2.7 to 6.0	16K	512	64	QFP	80
75P516	High-end; A/D converter	4.19	4.75 to 5.5	16K OTPROM	512	64	QFP Ceramic LCC	80 80

8-Bit, Single-Chip CMOS Microcomputers

Device, µPD	Features	Clock (MHz)	Supply Voltage (V)	ROM (X8)	RAM (X8)	I/O	# Package	Pins
78C10/78C10A	CMOS; A/D converter	15	4.5 to 5.5	External	256	32	QUIP SDIP QFP PLCC	64 64 64 68
78C11/78C11A	CMOS; A/D converter	15	4.5 to 5.5	4K	256	44	QUIP SDIP QFP PLCC	64 64 64 68
78C12A	CMOS; A/D converter	15	4.5 to 5.5	8K	256	44	QUIP SDIP QFP PLCC	64 64 64 68
78C14/78C14A	CMOS; A/D converter	15	4.5 to 5.5	16K	256	44	QUIP SDIP QFP PLCC	64 64 64 68
78CP14	CMOS; A/D converter	15	4.75 to 5.25	16K OTPROM	256	44	QUIP SDIP QFP PLCC	64 64 64 68
				16K UVEPROM	256	44	Ceramic QUIP Shrink cerdip	64 64
78CG14	CMOS; A/D converter; piggyback EPROM	15	4.5 to 5.5	4K, 8K or 16K	256	44	Ceramic QUIP	64
78213	CMOS; A/D converter; advanced peripherals	12	4.5 to 5.5	External	512	54	SDIP QUIP QFP PLCC	64 64 74 68
78214	CMOS; A/D converter; advanced peripherals	12	4.5 to 5.5	16K	512	54	SDIP QUIP QFP PLCC	64 64 74 68

8-Bit, Single-Chip NMOS/CMOS Microcomputers (cont)

Device, µPD	Features	Clock (MHz)	Supply Voltage (V)	ROM (X8)	RAM (X8)	I/O	# Package	Pins
78P214	CMOS; A/D converter; advanced peripherals	12	4.5 to 5.5	16K OTPROM	512	54	SDIP	64
							QUIP	64
							QFP	74
							PLCC	68
				16K UVEPROM	512	54	Shrink cerdip	64
78220	CMOS; analog comparator; large I/O	12	4.5 to 5.5	External	640	71	PLCC	84
							QFP	94
78224	CMOS; analog comparator; large I/O	12	4.5 to 5.5	16K	640	71	PLCC	84
							QFP	94
78P224	CMOS; analog comparator; large I/O	12	4.5 to 5.5	16K OTPROM	640	71	PLCC	84
							QFP	94
78233	CMOS; real-time outputs; A/D and D/A converters	12	4.5 to 5.5	External	640	64	QFP	80
							QFP	94
							PLCC	84
78234	CMOS; real-time outputs; A/D and D/A converters	12	4.5 to 5.5	16K	640	64	QFP	80
							QFP	94
							PLCC	84
78P238	CMOS; real-time outputs; A/D and D/A converters	12	4.5 to 5.5	32K OTPROM	640	64	QFP	80
							QFP	94
							PLCC	84
				32K UVEPROM	640	64	Ceramic LCC	94

8/16-Bit, Single-Chip CMOS Microcomputers

Device, µPD	Features	Clock (MHz)	Supply Voltage (V)	ROM (X8)	RAM (X8)	I/O	# Package	Pins
78310A	Real-time motor control	12	4.5 to 5.5	External	256	48	SDIP	64
							QUIP	64
							QFP	64
							PLCC	68
78312A	Real-time motor control	12	4.5 to 5.5	8K	256	48	SDIP	64
							QUIP	64
							QFP	64
							PLCC	68
78P312A	Real-time motor control	12	4.5 to 5.5	8K UVEPROM	256	48	Shrink cerdip	64
							Ceramic QUIP	64
				8K OTPROM	256	48	SDIP	64
							QUIP	64
							QFP	64
							PLCC	68
78320	High-end; advanced analog and digital peripherals	16	4.5 to 5.5	External	640	55	QFP	64
							PLCC	68
78322	High-end; advanced analog and digital peripherals	16	4.5 to 5.5	16K	640	55	QFP	64
							PLCC	68
78P322	High-end; advanced analog and digital peripherals	16	4.5 to 5.5	16K OTPROM	640	55	PLCC	68
							QFP	74
				16K UVEPROM	640	55	Ceramic LCC	68
							Ceramic LCC	74
71P301	Port and memory extender used with 7832X microcomputer family; UVEPROM or OTPROM	–	4.5 to 5.5	16K OTPROM	1K	16	PLCC	44
							QFP	64
							QUIP	64
				16K UVEPROM	1K	16	Ceramic LCC	44
							Ceramic LCC	64
							Ceramic QUIP	64

μPD75XX Series Development Tools Selection Guide

Part Number (Note 1)	Emulator*	Add-on Board*	System Evaluation Board	EPROM/OTP Device	PG-1500 Adapter (Note 2)	Absolute Assembler (Note 3)
μPD7502G-12	EVAKIT-7500B	EV7514	SE-7514A	—	—	ASM75
μPD7503G-12	EVAKIT-7500B	EV7514	SE-7514A	—	—	ASM75
μPD7507C	EVAKIT-7500B	—	—	μPD78CG08E	—	ASM75
μPD7507CU	EVAKIT-7500B	—	—	—	—	ASM75
μPD7507G-00	EVAKIT-7500B	—	—	—	—	ASM75
μPD7507HC	EVAKIT-7500B	EV7508H	—	μPD75CG08HE	—	ASM75
μPD7507HCU	EVAKIT-7500B	EV7508H	—	—	—	ASM75
μPD7507HG-22	EVAKIT-7500B	EV7508H	—	—	—	ASM75
μPD7508C	EVAKIT-7500B	—	—	μPD78CG08E	—	ASM75
μPD7508CU	EVAKIT-7500B	—	—	—	—	ASM75
μPD7508G-00	EVAKIT-7500B	—	—	—	—	ASM75
μPD75CG08E	EVAKIT-7500B	—	—	—	—	ASM75
μPD7508HC	EVAKIT-7500B	EV7508H	—	μPD78CG08HE	—	ASM75
μPD7508HCU	EVAKIT-7500B	EV7508H	—	—	—	ASM75
μPD7508HG-22	EVAKIT-7500B	EV7508H	—	—	—	ASM75
μPD75CG08HE	EVAKIT-7500B	EV7508H	—	—	—	ASM75
μPD7527AC	EVAKIT-7500B	EV7528	—	μPD78CG28E	—	ASM75
μPD7527ACU	EVAKIT-7500B	EV7528	—	—	—	ASM75
μPD7528AC	EVAKIT-7500B	EV7528	—	μPD78CG28E	—	ASM75
μPD7528ACU	EVAKIT-7500B	EV7528	—	—	—	ASM75
μPD75CG28E	EVAKIT-7500B	EV7528	—	—	—	ASM75
μPD7533C	EVAKIT-7500B	EV7533	—	μPD75CG33E	—	ASM75
μPD7533CU	EVAKIT-7500B	EV7533	—	—	—	ASM75
μPD7533G-22	EVAKIT-7500B	EV7533	—	—	—	ASM75
μPD75CG33E	EVAKIT-7500B	EV7533	—	—	—	ASM75
μPD7537AC	EVAKIT-7500B	EV7528	—	μPD75CG38E	—	ASM75
μPD7537ACU	EVAKIT-7500B	EV7528	—	—	—	ASM75
μPD7538AC	EVAKIT-7500B	EV7528	—	μPD75CG38E	—	ASM75
μPD7538ACU	EVAKIT-7500B	EV7528	—	—	—	ASM75
μPD75CG38E	EVAKIT-7500B	EV7528	—	—	—	ASM75
μPD7554CS	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P54CS	PA-75P54CS	ASM75
μPD7554G	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P54G	PA-75P54CS	ASM75
μPD7554ACS	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P54CS	PA-75P54CS	ASM75
μPD7554AG	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P54G	PA-75P54CS	ASM75
μPD75P54CS	EVAKIT-7500B	EV7554A	—	—	—	ASM75
μPD75P54G	EVAKIT-7500B	EV7554A	—	—	—	ASM75
μPD7556CS	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P56CS	PA-75P56CS	ASM75
μPD7556G	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P56G	PA-75P56CS	ASM75
μPD7556ACS	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P56CS	PA-75P56CS	ASM75
μPD7556AG	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P56G	PA-75P56CS	ASM75
μPD75P56CS	EVAKIT-7500B	EV7554A	—	—	—	ASM75
μPD75P56G	EVAKIT-7500B	EV7554A	—	—	—	ASM75
μPD7564CS	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P64CS	PA-75P54CS	ASM75
μPD7564G	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P64G	PA-75P54CS	ASM75
μPD7564ACS	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P64CS	PA-75P54CS	ASM75
μPD7564AG	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P64G	PA-75P54CS	ASM75
μPD75P64CS	EVAKIT-7500B	EV7554A	—	—	—	ASM75
μPD75P64G	EVAKIT-7500B	EV7554A	—	—	—	ASM75
μPD7566CS	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P66CS	PA-75P56CS	ASM75
μPD7566G	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P66G	PA-75P56CS	ASM75

* Required Tools

μPD75XX Series Development Tools Selection Guide (cont)

Part Number (Note 1)	Emulator*	Add-on Board*	System Evaluation Board	EPROM/OTP Device	PG-1500 Adapter (Note 2)	Absolute Assembler (Note 3)
μPD7566ACS	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P66CS	PA-75P56CS	ASM75
μPD7566AG	EVAKIT-7500B	EV7554A	SE-7554A	μPD75P66G	PA-75P56CS	ASM75
μPD75P66CS	EVAKIT-7500B	EV7554A	—	—	—	ASM75
μPD75P66G	EVAKIT-7500B	EV7554A	—	—	—	ASM75

* Required Tools

Notes:

(1) Packages:

Package	Description
C	40-pin plastic DIP (μPD7507/07H/08/08H)
	42-pin plastic DIP (μPD7527A/28A/33/37A/38A)
CS	20-pin plastic shrink DIP (μPD7554/54A/P54/64/64A/P64)
	24-pin plastic shrink DIP (μPD7556/56A/P56/66/66A/P66)
CU	40-pin plastic shrink DIP (μPD7507/07H/08/08H)
	42-pin plastic shrink DIP (μPD7527A/28A/33/37A/38A)
E	40-pin ceramic piggy-back DIP (μPD75CG08/08H)
	42-pin ceramic piggy-back DIP (μPD75CG28/33/38)
G	20-pin plastic SO (μPD7554/54A/P54/64/64A/P64)
	24-pin plastic SO (μPD7556/56A/P56/66/66A/P66)
G-00	52-pin plastic QFP
G-12	64-pin plastic QFP (μPD7502/03)
G-22	44-pin plastic QFP

(2) By using the specified adapter, the PG-1500 EPROM programmer can be used to program the OTP device.

(3) The ASM75 Absolute Assembler is provided to run under the MS-DOS® operating system. (ASM75-D52).

MS-DOS is a registered trademark of Microsoft Corporation.

μPD75XXX Series Development Tools Selection Guide

Part Number (Note 5)	Emulator*	Emulation Probe*	Optional Socket Adapter (Note 1)	EPROM/OTP Device (Note 2)	Relocatable Assembler (Note 3)	Structured Assembler (Note 4)
μPD75004CU	IE-75000-R	EP-75008CU-R	—	μPD75P008CU	RA75X	ST75X
μPD75006GB-3B4	IE-75000-R	EP-75008GB-R	EV-9200G-44	μPD75P008GB	RA75X	ST75X
μPD75006CU	IE-75000-R	EP-75008CU-R	—	μPD75P008CU	RA75X	ST75X
μPD75006GB-3B4	IE-75000-R	EP-75008GB-R	EV-9200G-44	μPD75P008GB	RA75X	ST75X
μPD75008CU	IE-75000-R	EP-75008CU-R	—	μPD75P008CU	RA75X	ST75X
μPD75008GB-3B4	IE-75000-R	EP-75008GB-R	EV-9200G-44	μPD75P008GB	RA75X	ST75X
μPD75P008CU	IE-75000-R	EP-75008CU-R	—	—	RA75X	ST75X
μPD75P008GB-3B4	IE-75000-R	EP-75008GB-R	EV-9200G-44	—	RA75X	ST75X
μPD75028CW	IE-75000-R	EP-75028CW-R	—	μPD75P036CW	RA75X	ST75X
μPD75028GC-AB8	IE-75000-R	EP-75028GC-R	EV-9200GC-64	μPD75P036GC	RA75X	ST75X
μPD75P036CW	IE-75000-R	EP-75028CW-R	—	—	RA75X	ST75X
μPD75P036GC-AB8	IE-75000-R	EP-75028GC-R	EV-9200GC-64	—	RA75X	ST75X
μPD75048CW	IE-75000-R	EP-75028CW-R	—	μPD75P056CW	RA75X	ST75X
μPD75048GC-AB8	IE-75000-R	EP-75028GC-R	EV-9200GC-64	μPD75P056GC	RA75X	ST75X
μPD75P056CW	IE-75000-R	EP-75028CW-R	—	—	RA75X	ST75X
μPD75P056GC-AB8	IE-75000-R	EP-75028GC-R	EV-9200GC-64	—	RA75X	ST75X
μPD75104CW	IE-75000-R	EP-75108CW-R	—	μPD75P108CW/DW μPD75P116CW	RA75X	ST75X
μPD75104G-1B	IE-75000-R	EP-75108GF-R	EV-9200G-64	μPD75P108G/GF μPD75P116GF	RA75X	ST75X
μPD75104GF-3BE	IE-75000-R	EP-75108GF-R	EV-9200G-64	μPD75P108G/GF μPD75P116GF	RA75X	ST75X
μPD75104AGC-AB8	IE-75000-R	EP-75108AGC-R	EV-9200GC-64	—	RA75X	ST75X
μPD75106CW	IE-75000-R	EP-75108CW-R	—	μPD75P108CW/DW μPD75P116CW	RA75X	ST75X
μPD75106G-1B	IE-75000-R	EP-75108GF-R	EV-9200G-64	μPD75P108G/GF μPD75P116GF	RA75X	ST75X
μPD75106GF-3BE	IE-75000-R	EP-75108GF-R	EV-9200G-64	μPD75P108G/GF μPD75P116GF	RA75X	ST75X
μPD75108AG-22	IE-75000-R	EP-75108AGC-R	EV-9200GC-64	—	RA75X	ST75X
μPD75108AGC-AB8	IE-75000-R	EP-75108AGC-R	EV-9200GC-64	—	RA75X	ST75X
μPD75108CW	IE-75000-R	EP-75108CW-R	—	μPD75P108CW/DW μPD75P116CW	RA75X	ST75X
μPD75108G-1B	IE-75000-R	EP-75108GF-R	EV-9200G-64	μPD75P108G/GF μPD75P116GF	RA75X	ST75X
μPD75108GF-3BE	IE-75000-R	EP-75108GF-R	EV-9200G-64	μPD75P108G/GF μPD75P116GF	RA75X	ST75X
μPD75P108BCW	IE-75000-R	EP-75108CW-R	—	—	RA75X	ST75X
μPD75P108BGF-3BE	IE-75000-R	EP-75108GF-R	EV-9200G-64	—	RA75X	ST75X
μPD75P108CW	IE-75000-R	EP-75108CW-R	—	—	RA75X	ST75X
μPD75P108DW	IE-75000-R	EP-75108CW-R	—	—	RA75X	ST75X
μPD75P108G-1B	IE-75000-R	EP-75108GF-R	EV-9200G-64	—	RA75X	ST75X
μPD75112CW	IE-75000-R	EP-75108CW-R	—	μPD75P116CW	RA75X	ST75X
μPD75112GF-3BE	IE-75000-R	EP-75108GF-R	EV-9200G-64	μPD75P116GF	RA75X	ST75X
μPD75116CW	IE-75000-R	EP-75108CW-R	—	μPD75P116CW	RA75X	ST75X
μPD75116GF-3BE	IE-75000-R	EP-75108GF-R	EV-9200G-64	μPD75P116GF	RA75X	ST75X
μPD75P116CW	IE-75000-R	EP-75108CW-R	—	—	RA75X	ST75X
μPD75P116GF-3BE	IE-75000-R	EP-75108GF-R	EV-9200G-64	—	RA75X	ST75X
μPD75206CW	IE-75000-R	EP-75216ACW-R	—	μPD75P216ACW	RA75X	ST75X
μPD75206G-1B	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75206BGF-3BE	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75208CW	IE-75000-R	EP-75216ACW-R	—	μPD75P216ACW	RA75X	ST75X

* Required Tools

μPD75XXX Series Development Tools Selection Guide (cont)

Part Number (Note 5)	Emulator*	Emulation Probe*	Optional Socket Adapter (Note 1)	EPROM/OTP Device (Note 2)	Relocatable Assembler (Note 3)	Structured Assembler (Note 4)
μPD75208G-1B	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75208GF-3BE	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75CG208E	IE-75000-R	EP-75216ACW-R	—	—	RA75X	ST75X
μPD75CG208EA	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75212ACW	IE-75000-R	EP-75216ACW-R	—	μPD75P216ACW	RA75X	ST75X
μPD75212AGF-3BE	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75216ACW	IE-75000-R	EP-75216ACW-R	—	μPD75P216ACW	RA75X	ST75X
μPD75216AGF	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75CG216AE	IE-75000-R	EP-75216ACW-R	—	—	RA75X	ST75X
μPD75CG216AEA	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75P216ACW	IE-75000-R	EP-75216ACW-R	—	μPD75P216ACW	RA75X	ST75X
μPD75268CW	IE-75000-R	EP-75216ACW-R	—	μPD75P216ACW	RA75X	ST75X
μPD75268GF-3BE	IE-75000-R	EP-75216AGF-R	EV-9200G-64	—	RA75X	ST75X
μPD75304GF-3B9	IE-75000-R	EP-75308GF-R	EV-9200G-80	μPD75P308GF/K	RA75X	ST75X
μPD75306GF-3B9	IE-75000-R	EP-75308GF-R	EV-9200G-80	μPD75P308GF/K	RA75X	ST75X
μPD75308GF-3B9	IE-75000-R	EP-75308GF-R	EV-9200G-80	μPD75P308GF/K	RA75X	ST75X
μPD75P308GF-3B9	IE-75000-R	EP-75308GF-R	EV-9200G-80	—	RA75X	ST75X
μPD75P308K	IE-75000-R	EP-75308GF-R	EV-9200G-80	—	RA75X	ST75X
μPD75312GF-3B9	IE-75000-R	EP-75308GF-R	EV-9200G-80	μPD75P316GF	RA75X	ST75X
μPD75316GF-3B9	IE-75000-R	EP-75308GF-R	EV-9200G-80	μPD75P316GF	RA75X	ST75X
μPD75P316GF-3B9	IE-75000-R	EP-75308GF-R	EV-9200G-80	—	RA75X	ST75X
μPD75P316AGF-3B9	IE-75000-R	EP-75308GF-R	EV-9200G-80	—	RA75X	ST75X
μPD75P316AK	IE-75000-R	EP-75308GF-R	EV-9200G-80	—	RA75X	ST75X
μPD75328GC-3B9	IE-75000-R	EP-75328GC-R	EV-9200GC-80	μPD75P328GC	RA75X	ST75X
μPD75P328GC-3B9	IE-75000-R	EP-75328GC-R	EV-9200GC-80	—	RA75X	ST75X
μPD75402AC	IE-75000-R	EP-75402C-R	—	μPD75P402C	RA75X	ST75X
μPD75402ACT	IE-75000-R	EP-75402C-R	—	μPD75P402CT	RA75X	ST75X
μPD75402AGB-3B4	IE-75000-R	EP-75402GB-R	EV-9200G-44	μPD75P402GB	RA75X	ST75X
μPD75P402C	IE-75000-R	EP-75402C-R	—	—	RA75X	ST75X
μPD75P402CT	IE-75000-R	EP-75402C-R	—	—	RA75X	ST75X
μPD75P402GB-3B4	IE-75000-R	EP-75402GB-R	EV-9200G-44	—	RA75X	ST75X
μPD75512GF-3B9	IE-75000-R	EP-75516GF-R	EV-9200G-80	μPD75P516GF/K	RA75X	ST75X
μPD75516GF-3B9	IE-75000-R	EP-75516GF-R	EV-9200G-80	μPD75P516GF/K	RA75X	ST75X
μPD75P516GF-3B9	IE-75000-R	EP-75516GF-R	EV-9200G-80	—	RA75X	ST75X
μPD75P516K	IE-75000-R	EP-75516GF-R	EV-9200G-80	—	—	—

Notes:

(1) The EV-9200G-XX is an LCC socket with the footprint of the flat package. One unit is supplied with the probe. Additional units are available as replacement parts in sets of five.

(2) All EPROM/OTP devices can be programmed using the NEC PG-1500. Refer to the PG-1500 Programming Socket Adapter Selection Guide for the appropriate socket adapter.

(3) The RA75X relocatable assembler package is provided for the following operating systems:
RA75X-D52 (MS-DOS®)
RA75X-VVT1 (VAX/VMS®)

(4) The ST75X structures assembler preprocessor is provided with RA75X.

(5) Packages:

Package	Description
C	28-pin plastic DIP
CT	28-pin plastic shrink DIP
CU	42-pin plastic shrink DIP
CW	64-pin plastic shrink DIP
DW	64-pin ceramic shrink DIP with window
E	64-pin ceramic piggy-back shrink DIP
EA	64-pin ceramic piggy-back QFP
G-1B	64-pin plastic QFP (2.05 mm thick)
G-22	64-pin plastic QFP (1.55 mm thick)
GB-3B4	44-pin plastic QFP
GC-AB8	64-pin plastic QFP (2.55 mm thick)
GC-3B9	80-pin plastic QFP
GF-3BE	64-pin plastic QFP (2.77 mm thick)
GF-3B9	80-pin plastic QFP
K	80-pin ceramic LCC

MS-DOS is a registered trademark of Microsoft Corporation.

VAX and VMS are registered trademarks of Digital Equipment Corporation.

μPD78XX Series Development Tools Selection Guide**

Part Number (Note 1)	Emulator*	Emulation Probe*	EPROM/OTP Device	PG-1500 Adapter (Note 2)	Relocatable Assembler (Note 9)	C Compiler (Note 9)
μPD78C10CW	IE-78C11-M	EV-9001-64 (Note 3)	—	—	RA87	CC87
μPD78C10G-36	IE-78C11-M	(Note 4)	—	—	RA87	CC87
μPD78C10G-1B	IE-78C11-M	(Note 5)	—	—	RA87	CC87
μPD78C10GF-3BE	IE-78C11-M	(Note 5)	—	—	RA87	CC87
μPD78C10L	IE-78C11-M	(Note 5)	—	—	RA87	CC87
μPD78C10ACW	IE-78C11-M (Note 8)	EV-9001-64 (Note 3)	—	—	RA87	CC87
μPD78C10AGQ-36	IE-78C11-M (Note 8)	(Note 4)	—	—	RA87	CC87
μPD78C10AGF-3BE	IE-78C11-M (Note 8)	(Note 5)	—	—	RA87	CC87
μPD78C10AL	IE-78C11-M (Note 8)	(Note 5)	—	—	RA87	CC87
μPD78C11CW	IE-78C11-M	EV-9001-64 (Note 3)	μPD78CP14CW/DW	PA-78CP14CW	RA87	CC87
μPD78C11G-36	IE-78C11-M	(Note 4)	μPD78CP14G-36/R μPD78CP14E	PA-78CP14GQ	RA87	CC87
μPD78C11G-1B	IE-78C11-M	(Note 5)	μPD78CP14GF-3BE	PA-78CP14GF	RA87	CC87
μPD78C11GF-3BE	IE-78C11-M	(Note 5)	μPD78CP14GF-3BE	PA-78CP14GF	RA87	CC87
μPD78C11L	IE-78C11-M	(Note 5)	μPD78CP14L	PA-78CP14L	RA87	CC87
μPD78C11ACW	IE-78C11-M (Note 7)	EV-9001-64 (Note 3)	μPD78CP14CW/DW (Note 6)	PA-78CP14CW	RA87	CC87
μPD78C11AGQ-36	IE-78C11-M (Note 7)	(Note 4)	μPD78CP14G-36/R (Note 6)	PA-78CP14GQ	RA87	CC87
μPD78C11AGF-3BE	IE-78C11-M (Note 7)	(Note 5)	μPD78CP14GF-3BE (Note 6)	PA-78CP14GF	RA87	CC87
μPD78C11AL	IE-78C11-M (Note 7)	(Note 5)	μPD78CP14L (Note 6)	PA-78CP14L	RA87	CC87
μPD78C12ACW	IE-78C11-M (Note 7)	EV-9001-64 (Note 3)	μPD78CP14CW/DW (Note 6)	PA-78CP14CW	RA87	CC87
μPD78C12AGQ-36	IE-78C11-M (Note 7)	(Note 4)	μPD78CP14G-36/R (Note 6)	PA-78CP14GQ	RA87	CC87
μPD78C12AGF-3BE	IE-78C11-M (Note 7)	(Note 5)	μPD78CP14GF-3BE (Note 6)	PA-78CP14GF	RA87	CC87
μPD78C12AL	IE-78C11-M (Note 7)	(Note 5)	μPD78CP14L (Note 6)	PA-78CP14L	RA87	CC87
μPD78C14CW	IE-78C11-M	EV-9001-64 (Note 3)	μPD78CP14CW/DW	PA-78CP14CW	RA87	CC87
μPD78C14G-36	IE-78C11-M	(Note 4)	μPD78CP14G-36/R μPD78CG14E	PA-78CP14GQ —	RA87	CC87
μPD78C14G-1B	IE-78C11-M	(Note 5)	μPD78CP14GF	PA-78CP14GF	RA87	CC87
μPD78C14GF-3BE	IE-78C11-M	(Note 5)	μPD78CP14GF	PA-78CP14GF	RA87	CC87
μPD78C14L	IE-78C11-M	(Note 5)	μPD78CP14L	PA-78CP14L	RA87	CC87
μPD78C14AG-AB8	IE-78C11-M (Note 7)	(Note 5)	—	—	RA87	CC87
μPD78CG14E (Note 8)	IE-78C11-M	(Note 4)	—	—	RA87	CC87
μPD78CP14CW	IE-78C11-M	EV-9001-64 (Note 3)	—	PA-78CP14CW	RA87	CC87
μPD78CP14DW	IE-78C11-M	EV-9001-64 (Note 3)	—	PA-78CP14CW	RA87	CC87

* Required Tools

**For all μPD78C1X devices, you may use the DDK-78C10 for evaluation purposes.

μPD78XX Series Development Tools Selection Guide** (cont)

Part Number (Note 1)	Emulator*	Emulation Probe*	EPROM/OTP Device	PG-1500 Adapter (Note 2)	Relocatable Assembler (Note 9)	C Compiler (Note 9)
μPD78CP14G-36	IE-78C11-M	(Note 4)	—	PA-78CP14GQ	RA87	CC87
μPD78CP14GF-3BE	IE-78C11-M	(Note 5)	—	PA-78CP14GF	RA87	CC87
μPD78CP14L	IE-78C11-M	(Note 5)	—	PA-78CP14L	RA87	CC87
μPD78CP14R	IE-78C11-M	(Note 4)	—	PA-78CP14GQ	RA87	CC87

* Required Tools

**For all μPD78C1X devices, you may use the DDK-78C10 for evaluation purposes.

Notes:

(1) Packages:

Package	Description
CW	64-pin plastic shrink DIP
DW	64-pin ceramic shrink DIP with window
E	64-pin ceramic piggy-back QUIP
G-1B	64-pin plastic QFP (Resin Thickness: 2.05 mm)
G-36	64-pin plastic QUIP
G-AB8	64-pin plastic QFP (Interpin Pitch: 0.8 mm)
GF-3BE	64-pin plastic QFP (Resin Thickness: 2.7 mm)
GQ-36	64-pin plastic QUIP
L	68-pin PLCC
R	64-pin ceramic QUIP with window

(2) By using the specified adapter, the PG-1500 EPROM programmer can be used to program the EPROM/OTP device.

(3) 64-pin shrink DIP adapter which plugs into the EP-7811HGQ emulation probe supplied with each IE.

(4) The emulation probe for the 64-pin QUIP package (EP-7811HGQ) is supplied with the IE.

(5) No emulation probe available.

(6) The μPD78CP14 EPROM/OTP devices do not have pull-up resistors on ports A, B, and C.

(7) The IE-78C11-M can be used by replacing the μPD78C10G-36 with a μPD78C10AGQ-36. However, it will not be able to emulate the optional pull-up resistors on ports A, B, and C.

(8) The μPD78CG14E is a piggy-back EPROM device in a ceramic QUIP package. It accepts 27C256 and 27C256A EPROMS.

(9) The following relocatable assemblers and C compilers are available:

RA87-D52 RA87-VVT1	(MS-DOS®) (VAX/VMS®)	Relocatable assemblers for 78XX series
CCMSD-I5DD-87 CCVMS-OT16-87 CCUNX-OT16-87	(MS-DOS) (VAX/VMS) (VAX/UNIX®) 4.2 BSD or Ultrix®)	C Compilers for 78XX Series

MS-DOS is a registered trademark of Microsoft Corporation.
VAX, VMS and Ultrix are registered trademarks of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories.

μPD782XX Series Development Tools Selection Guide (Note 1)

Part Number (Note 2)	Evaluation Kit (Note 3	Designer Kit (Note 4)	Emulator Kit (Note 5)	Low-End Emulator	Emulation System	Emulation Probe	Device (Note 6)
μPD78213CW	EK-78K2-21X	DK-78K2-21XCW	IK-78K2-21XCW	EB-78210-PC	IE-78210-R	EP-78210CW-R	—
μPD78213GJ-5BJ	EK-78K2-21X	DK-78K2-21XGJ	IK-78K2-21XGJ	EB-78210-PC	IE-78210-R	EP-78210GJ-R (7)	—
μPD78213GQ-36	EK-78K2-21X	DK-78K2-21XGQ	IK-78K2-21XGQ	EB-78210-PC	IE-78210-R	EP-78210GQ-R	—
μPD78213L	EK-78K2-21X	DK-78K2-21XL	IK-78K2-21XL	EB-78210-PC	IE-78210-R	EP-78210L-R	—
μPD78214CW	EK-78K2-21X	DK-78K2-21XCW	IK-78K2-21XCW	EB-78210-PC	IE-78210-R	EP-78210CW-R	μPD78P214CW/DW
μPD78214GJ-5BJ	EK-78K2-21X	DK-78K2-21XGJ	IK-78K2-21XGJ	EB-78210-PC	IE-78210-R	EP-78210GJ-R (7)	μPD78P214GJ
μPD78214GQ-36	EK-78K2-21X	DK-78K2-21XGQ	IK-78K2-21XGQ	EB-78210-PC	IE-78210-R	EP-78210GQ-R	μPD78P214GQ
μPD78214L	EK-78K2-21X	DK-78K2-21XL	IK-78K2-21XL	EB-78210-PC	IE-78210-R	EP-78210L-R	μPD78P214L
μPD78P214CW	EK-78K2-21X	DK-78K2-21XCW	IK-78K2-21XCW	EB-78210-PC	IE-78210-R	EP-78210CW-R	—
μPD78P214DW	EK-78K2-21X	DK-78K2-21XCW	IK-78K2-21XCW	EB-78210-PC	IE-78210-R	EP-78210CW-R	—
μPD78P214GJ-5BJ	EK-78K2-21X	DK-78K2-21XGJ	IK-78K2-21XGJ	EB-78210-PC	IE-78210-R	EP-78210GJ-R (7)	—
μPD78P214GQ-36	EK-78K2-21X	DK-78K2-21XGQ	IK-78K2-21XGQ	EB-78210-PC	IE-78210-R	EP-78210GQ-R	—
μPD78P214L	EK-78K2-21X	DK-78K2-21XL	IK-78K2-21XL	EB-78210-PC	IE-78210-R	EP-78210L-R	—
μPD78220GJ-5BG	EK-78K2-22X	DK-78K2-22XGJ	IK-78K2-22XGJ	EB-78220-PC	IE-78220-R	EP-78220GJ-R (8)	—
μPD78220L	EK-78K2-22X	DK-78K2-22XL	IK-78K2-22XL	EB-78220-PC	IE-78220-R	EP-78220L-R	—
μPD78224GJ-5BG	EK-78K2-22X	DK-78K2-22XGJ	IK-78K2-22XGJ	EB-78220-PC	IE-78220-R	EP-78220GJ-R (8)	μPD78P224GJ
μPD78224L	EK-78K2-22X	DK-78K2-22XL	IK-78K2-22XL	EB-78220-PC	IE-78220-R	EP-78220L-R	μPD78P224L
μPD78P224GJ-5BG	EK-78K2-22X	DK-78K2-22XGJ	IK-78K2-22XGJ	EB-78220-PC	IE-78220-R	EP-78220GJ-R (8)	—
μPD78P224L	EK-78K2-22X	DK-78K2-22XL	IK-78K2-22XL	EB-78220-PC	IE-78220-R	EP-78220L-R	—
μPD78233GC-3B9	EK-78K2-23X	DK-78K2-23XGC	IK-78K2-23XGC	EB-78230-PC	IE-78230-R	EP-78230GC-R	—
μPD78233GJ-5BG	EK-78K2-23X	DK-78K2-23XGJ	IK-78K2-23XGJ	EB-78230-PC	IE-78230-R	EP-78230GJ-R	—
μPD78233LQ	EK-78K2-23X	DK-78K2-23XL	IK-78K2-23XL	EB-78230-PC	IE-78230-R	EP-78230LQ-R	—
μPD78234GC-3B9	EK-78K2-23X	DK-78K2-23XGC	IK-78K2-23XGC	EB-78230-PC	IE-78230-R	EP-78230GC-R	μPD78P238GC
μPD78234GJ-5BG	EK-78K2-23X	DK-78K2-23XGJ	IK-78K2-23XGJ	EB-78230-PC	IE-78230-R	EP-78230GJ-R	μPD78P238GJ/KF
μPD78234LQ	EK-78K2-23X	DK-78K2-23XL	IK-78K2-23XL	EB-78230-PC	IE-78230-R	EP-78230LQ-R	μPD78P238LQ
μPD78P238GC-3B9	EK-78K2-23X	DK-78K2-23XGC	IK-78K2-23XGC	EB-78230-PC	IE-78230-R	EP-78230GC-R	—
μPD78P238GJ-5BG	EK-78K2-23X	DK-78K2-23XGJ	IK-78K2-23XGJ	EB-78230-PC	IE-78230-R	EP-78230GJ-R	—
μPD78P238KF	EK-78K2-23X	DK-78K2-23XGJ	IK-78K2-23XGJ	EB-78230-PC	IE-78230-R	EP-78230GJ-R	—
μPD78P238LQ	EK-78K2-23X	DK-78K2-23XL	IK-78K2-23XL	EB-78230-PC	IE-78230-R	EP-78230LQ-R	—

Notes:

1. The following software packages are available for the μPD782XX Series:

   RA78K2 relocatable assembler package
   RA78K2-D52 (MS-DOS®)
   RA78K2-VVT1 (VAX®/VMS®)
   ST78K2 Structured assembler preprocessor
   Provided with RA78K2
   CC782XX C Compiler package
   CCMSD-I5DD-782XX (MS-DOS®)

(2) Packages:

Package	Description
CW	64-pin plastic shrink DIP
DW	64-pin ceramic shrink DIP with window
GC-3B9	80-pin plastic QFP
GJ-5BG	94-pin plastic QFP
GJ-5BJ	74-pin plastic QFP
GQ-36	64-pin plastic QUIP
KF	94-pin ceramic LCC with window
L	68-pin PLCC (μPD78213/214/P214L)
	84-pin PLCC (μPD78220/224/P224L)
LQ	84-pin PLCC

MS-DOS is a registered trademark of Microsoft Corporation.

VAX and VMS are registered trademarks of Digital Equipment Corporation.

(3) The μPD782XX Evaluation Kit contains the appropriate DDB-78K2-2XX evaluation board for the part selected, the RA78K2 Relocatable Assembler Package, and the ST78K2 Structured Assembler Preprocessor.

(4) The μPD782XX Designer Kit contains the appropriate EB-782XX-PC low-end emulator and emulation probe for the part selected, the RA78K2 Relocatable Assembler Package, and the ST78K2 Structured Assembler Preprocessor.

(5) The μPD782XX Emulator Kit contains the appropriate IE-782XX system and emulation probe for the part selected, the RA78K2 Relocatable Assembler Package, and the ST78K2 Structured Assembler Preprocessor.

(6) All EPROM/OTP devices can be programmed using the NEC PG-1500. Refer to the PG-1500 Programming Socket Adapter Selection Guide for the appropriate programming adapter.

(7) The EP-78210GJ-R emulation probe is shipped with one EV-9200G-74, a 74-pin LCC socket with the footprint of the QFP package. Additional sockets are available as replacement parts in sets of five.

(8) The EP-78220GJ-R emulation probe is shipped with one EV-9200G-94, a 94-pin LCC socket with the footprint of the QFP package. Additional sockets are available as replacement parts in sets of five.

μPD783XX Series Development Tools Selection Guide (Note 10)

Part Number (Note 1)	Emulator*	Emulation Probe*	EPROM/OTP Device	PG-1500 Adapter (Note 2)	Relocatable Assembler (Note11)	Structured Assembler (Note 12)	C Compiler (Note 13)
μPD78310ACW	IE-78310A-R	(Note 3)	—	—	RA78K3	ST78K3	CC7831X (Note 5)
μPD78310AGF-3BE	IE-78310A-R	EP-78310GF (Note 6)	—	—	RA78K3	ST78K3	CC7831X (Note 5)
μPD78310AGQ-36	IE-78310A-R	(Note 4)	—	—	RA78K3	ST78K3	CC7831X (Note 5)
μPD78310AL	IE-78310A-R	EP-78310L	—	—	RA78K3	ST78K3	CC7831X (Note 5)
μPD78312ACW	IE-78310A-R	(Note 3)	μPD78P312ACW/DW	PA-78P312CW	RA78K3	ST78K3	CC7831X (Note 5)
μPD78312AGF-3BE	IE-78310A-R	EP-78310GF (Note 6)	μPD78P312AGF-3BE	PA-78P312GF	RA78K3	ST78K3	CC7831X (Note 5)
μPD78312AGQ-36	IE-78310A-R	(Note 4)	μPD78P312AGQ/RQ	PA-78P312GQ	RA78K3	ST78K3	CC7831X (Note 5)
μPD78312AL	IE-78310A-R	EP-78310L	μPD78P312AL	PA-78P312L	RA78K3	ST78K3	CC7831X (Note 5)
μPD78P312ACW	IE-78310A-R	(Note 3)	—	PA-78P312CW	RA78K3	ST78K3	CC7831X (Note 5)
μPD78P312ADW	IE-78310A-R	(Note 3)	—	PA-78P312CW	RA78K3	ST78K3	CC7831X (Note 5)
μPD78P312AGF-3BE	IE-78310A-R	EP-78310GF (Note 6)	—	PA-78P312GF	RA78K3	ST78K3	CC7831X (Note 5)
μPD78P312AGQ-36	IE-78310A-R	(Note 4)	—	PA-78P312GQ	RA78K3	ST78K3	CC7831X (Note 5)
μPD78P312AL	IE-78310A-R	EP-78310L	—	PA-78P312L	RA78K3	ST78K3	CC7831X (Note 5)
μPD78P312AR	IE-78310A-R	(Note 4)	—	PA-78P312GQ	RA78K3	ST78K3	CC7831X (Note 5)
μPD78320GJ-5BJ	IE-78320-R	EP-78320GJ-R (Note 7)	—	—	RA78K3	ST78K3	CC7832X
μPD78320L	IE-78320-R	EP-78320L-R	—	—	RA78K3	ST78K3	CC7832X
μPD78322GJ-5BJ	IE-78320-R	EP-78320GJ-R (Note 7)	μPD78P322GJ μPD78P322KD	PA-78P322GJ PA-78P322KD	RA78K3	ST78K3	CC7832X
μPD78322L	IE-78320-R	EP-78320L-R	μPD78P322L μPD78P322KC	PA-78P322L PA-78P322KC	RA78K3	ST78K3	CC7832X
μPD78P322GJ	IE-78320-R	EP-78320GJ-R (Note 7)	—	PA-78P322GJ	RA78K3	ST78K3	CC7832X
μPD78P322KC	IE-78320-R	EP-78320L-R	—	PA-78P322KC	RA78K3	ST78K3	CC7832X
μPD783P322KD	IE-78320-R	EP-78320GJ-R (Note 7)	—	PA-78P322KD	RA78K3	ST78K3	CC7832X
μPD78P322L	IE-78320-R	EP-78320L-R	—	PA-78P322L	RA78K3	ST78K3	CC7832X
μPD71P301GF-3BE	IE-78320-R	—	—	PA-71P301GF	—	—	—
μPD71P301GQ-36	IE-78320-R	—	—	PA-71P301GQ	—	—	—
μPD71P301KA (Note 8)	IE-78320-R	—	—	PA-71P301KA	—	—	—
μPD71P301KB (Note 9)	IE-78320-R	—	—	PA-71P301KB	—	—	—
μPD71P301L	IE-78320-R	—	—	PA-71P301L	—	—	—
μPD71P301RQ	IE-78320-R	—	—	PA-71P301GQ	—	—	—

* Required Tools

Notes:

(1) Packages:

Package	Description
CW	64-pin plastic shrink DIP
DW	64-pin ceramic shrink DIP with window
GF-3BE	64-pin plastic QFP (Resin Thickness: 2.7 mm)
GJ-5BJ	74-pin plastic QFP (20 mm × 20 mm)
GQ-36	64-pin plastic QUIP
KA	44-pin ceramic LCC with window
KB	64-pin ceramic LCC with window
KC	68-pin ceramic LCC with window
KD	74-pin ceramic LCC with window
L	44-pin PLCC (μPD71P301L)
	68-pin PLCC (μPD78310A/312A/P312AL, μPD78320/322L)
R	64-pin ceramic QUIP with window
RQ	64-pin ceramic QUIP with window

(2) By using the specified adapter, the PG-1500 EPROM programmer can be used to program the EPROM/OTP device.

(3) The emulation probe for the 64-pin shrink DIP package (EP-78310CW) is supplied with the IE.

(4) The emulation probe for the 64-pin QUIP package (EP-78310GQ) is supplied with the IE.

(5) There are two C Compilers for the μPD7831X devices: CC7831X from NEC Electronics and one from Lattice Corporation.

(6) The EP-78310GF emulation probe is shipped with one EV-9200G-64, a 64-pin LCC socket with the footprint of the QFP package. Additional sockets are available as replacement parts in sets of five.

(7) The EP-78320GJ-R emulation probe is shipped with one EV-9200G-74, a 74-pin LCC socket with the footprint of the QFP package. Additional sockets are available as replacement parts in sets of five.

(8) Sockets for the μPD71P301KA (44-pin LCC package) are available from Yamaichi, Inc. (IC61-0444-030).

(9) Sockets for the μPD71P301KB (64-pin LCC package) are available from NEC Electronics (EV-9200G-64) in sets of five.

(10) The following evaluation boards are available for the μPD783XX series:

Part Number	Design/Development Boards	Evaluation Boards
μPD7831XA	—	DDK-78310A
μPD7832X	EB-78320-PC	—

(11) The following relocatable packages are available:

RA-78K3-D52	(MS-DOS®)	Relocatable assembler
RA-78K3-VVT1	(VAX/VMS®)	for 783XX series

(12) The ST78K3 structured assembler preprocessor is provided with RA78K3.

(13) The following C Compiler packages are available:

CCMSD-I5DD-7831X	(MS-DOS®)	For μPD7831X series
CCMSD-I5DD-7832X	(MS-DOS®)	For μPD7832X series

MS-DOS is a registered trademark of Microsoft Corporation.

VAX and VMS are registered trademarks of Digital Equipment Corporation.

Socket Adapters and Adapter Modules

Target Chip	Socket Adapter (Note 1)	Adapter Module (Note 2)
Standard 27XXX EPROM Devices		
μPD27256 (12.5 V)	—	027A Board
μPD27256 (21 V)	—	027A Board
μPD27C256	—	027A Board
μPD27C256A	—	027A Board
μPD27C512	—	027A Board
μPD27C1000	—	027A Board
μPD27C1001	—	027A Board
μPD27C10124	—	027A Board
μPD75XX Series Devices		
μPD75P54CS	PA-75P54CS	04A Board
μPD75P54G	PA-75P54CS	04A Board
μPD75P56CS	PA-75P54CS	04A Board
μPD75P56G	PA-75P54CS	04A Board
μPD75P64CS	PA-75P54CS	04A Board
μPD75P64G	PA-75P54CS	04A Board
μPD75P66CS	PA-75P56CS	04A Board
μPD75P66G	PA-75P56CS	04A Board
μPD75XXX Series Devices		
μPD75P008CU	PA-75P008CU	04A Board
μPD75P008GB	PA-75P008CU	04A Board
μPD75P036CW	PA-75P036CW	04A Board
μPD75P036GC	PA-75P036GC	04A Board
μPD75P108BCW	PA-75P108CW	04A Board
μPD75P108CW	PA-75P108CW	04A Board
μPD75P108DW	PA-75P108CW	04A Board
μPD75P108BGF	PA-75P108G PA-75P116GF	04A Board 04A Board
μPD75P108G	PA-75P108G PA-75P116GF	04A Board 04A Board
μPD75P116CW	PA-75P108CW	04A Board
μPD75P116GF	PA-75P108G PA-75P116GF	04A Board 04A Board
μPD75P216GF	PA-75P216ACW	04A Board
μPD75P308GF	PA-75P308GF	04A Board
μPD75P308K	PA-75P308K	04A Board
μPD75P316GF	PA-75P308GF	04A Board
μPD75P316AGF	PA-75P308GF	04A Board
μPD75P316AK	PA-75P308K	04A Board
μPD75P328GC	PA-75P328GC	04A Board
μPD75P402C	(Note 3)	027A Board
μPD75P402CT	PA-75P402CT	027A Board
μPD75P402GB	PA-75P402GB	027A Board
μPD75P516GF	PA-75P516GF	04A Board
μPD75P516K	PA-75P516K	04A Board

Target Chip	Socket Adapter (Note 1)	Adapter Module (Note 2)
μPD78XX Series Devices		
μPD78CP14CW	PA-78CP14CW	027A Board
μPD78CP14DW	PA-78CP14CW	027A Board
μPD78CP14G	PA-78CP14GQ	027A Board
μPD78CP14GF	PA-78CP14GF	027A Board
μPD78CP14L	PA-78CP14L	027A Board
μPD78CP14R	PA-78CP14GQ	027A Board
μPD78XXX Series Devices		
μPD71P301GF	PA-71P301GF	027A Board
μPD71P301GQ	PA-71P301GQ	027A Board
μPD71P301KA	PA-71P301KA	027A Board
μPD71P301KB	PA-71P301KB	027A Board
μPD71P301L	PA-71P301L	027A Board
μPD78P214CW	PA-78P214CW	027A Board
μPD78P214GJ	PA-78P214GJ	027A Board
μPD78P214GQ	PA-78P214GQ	027A Board
μPD78P214L	PA-78P214L	027A Board
μPD78P224GJ	PA-78P224GJ	027A Board
μPD78P224L	PA-78P224L	027A Board
μPD78P238GC	PA-78P238GC	027A Board
μPD78P238GJ	PA-78P238GJ	027A Board
μPD78P238KF	PA-78P238KF	027A Board
μPD78P238LQ	PA-78P238LQ	027A Board
μPD78P312ACW	PA-78P312CW	027A Board
μPD78P312ADW	PA-78P312CW	027A Board
μPD78P312AGF	PA-78P312GF	027A Board
μPD78P312AGQ	PA-78P312GQ	027A Board
μPD78P312AL	PA-78P312L	027A Board
μPD78P312AR	PA-78P312GQ	027A Board
μPD78P322GJ	PA-78P322GJ	027A Board
μPD78P322KC	PA-78P322KC	027A Board
μPD78P322KD	PA-78P322KD	027A Board
μPD78P322L	PA-78P322L	027A Board
V-Series Devices		
μPD70P322K	PA-70P322L	027A Board
Digital Signal Processors		
μPD77P56CR	PA-77P56C	04A Board
μPD77P56G	PA-77P56C	04A Board
μPD77P25C	PA-77P25C	027A Board
μPD77P25D	PA-77P25C	027A Board
μPD77P230R	PA-77P230R	027A Board

Notes:

(1) All socket adapters must be purchased separately.

(2) The 27A and 04A Adapter Modules are shipped with the PG-1500.

(3) The μPD75P402C does not require a programming socket adapter. It can be plugged directly into the 027A board.

CMOS Microprocessors

Device, μPD	Features	Data Bits	Clock (MHz)	# Package	Pins
70008A	*Z80 microprocessor	8	8	DIP	40
				QFP	44
				PLCC	44
70108 (V20)	8088 compatible; enhanced	8/16	8 or 10	DIP	40
				Ceramic DIP	40
				QFP	52
				PLCC	44
70116 (V30)	8086 compatible; enhanced	16	8 or 10	DIP	40
				Ceramic DIP	40
				QFP	52
				PLCC	44
70208 (V40)	MS-DOS, V20 compatible CPU with peripherals	8/16	8 or 10	Ceramic PGA	68
				PLCC	68
				QFP	80
70216 (V50)	MS-DOS, V30 compatible CPU with peripherals	16/16	8 or 10	PGA	68
				PLCC	68
				QFP	80
70616 (V60)	32-bit; high-speed	16/32	16	PGA	68
70632 (V70)	32-bit; high-speed	32/32	20/25	PGA	132
70832 (V80)	32-bit; high-speed	32/32	25	Ceramic PGA	208
70136 (V33)	Hardwired, enhanced V30	16	12 or 16	PGA	68
				PLCC	68
70236 (V53)	V33 core-based; high-integration; DMA, serial I/O, interrupt controller, etc.	16	–	Ceramic PGA	132
				QFP	120
70320 (V25)	MS-DOS compatible; high-integration; DMA, serial I/O, interrupt controller, etc.	8/16	5 or 8	PLCC	84
				QFP	94
70330 (V35)	MS-DOS compatible; high-integration; DMA, serial I/O, interrupt controller, etc.	16	8	PLCC	84
				QFP	94
70325 (V25+)	MS-DOS compatible; high-integration; high-speed DMA	8/16	8 or 10	PLCC	84
				QFP	94
70335 (V35+)	MS-DOS compatible; high-integration; high-speed DMA	16	8 or 10	PLCC	84
				QFP	94
70327 (V25 Software Guard)	MS-DOS compatible; high-integration; software protection	8/16	8	PLCC	84
				QFP	94
70337 (V35 Software Guard)	MS-DOS compatible; high-integration; software protection	16	8	PLCC	84
				QFP	94
79011 (V25 RTOS)	MS-DOS compatible; high-integration; real-time operating system	8/16	8	PLCC	84
				QFP	94
79021 (V35 RTOS)	MS-DOS compatible; high-integration; real-time operating system	16	8	PLCC	84
				QFP	94
70322 (V25 ROM)	MS-DOS compatible; high-integration; 16K-byte ROM	8/16	8	PLCC	84
				QFP	94

Plastic unless ceramic (or cerdip) is specified.
* For additional information, refer to 1987 Microcomputer Data Book.

CMOS Microprocessors (cont)

Device, µPD	Features	Data Bits	Clock (MHz)	# Package	Pins
70P322	MS-DOS compatible; high-integration; 16K-byte UVEPROM; V25 or V35 mode	8/16	8	Ceramic LCC	84
70332 (V35 ROM)	MS-DOS compatible; high-integration; 16K-byte ROM	16	8	PLCC	84
				QFP	94

NMOS and HMOS Microprocessors

Device, µPD	Features	Data Bits	Clock (MHz)	# Package	Pins
8085A	*8-bit microprocessor; NMOS or HMOS	8	5	DIP	40
8086	*16-bit microprocessor; HMOS	16	8	Cerdip	40
8088	*8-bit microprocessor; HMOS	8	8	Ceramic DIP	40

CMOS System Support Products

Device, µPD	Name	Data Bits	Clock (MHz)	# Package	Pins
71011	Clock Pulse Generator/Driver	–	20	DIP	18
				SOP	20
71037	Programmable DMA Controller	8	10	DIP	40
				QFP	40
				PLCC	44
71051	Serial Control Unit	8	8/10	DIP	28
				QFP	44
				PLCC	28
71054	Programmable Timer/Controller	8	8/10	DIP	24
				QFP	44
				PLCC	28
71055	Parallel Interface Unit	8	8/10	DIP	40
				QFP	44
				PLCC	44
71059	Interrupt Control Unit	8	8/10	DIP	28
				QFP	44
				PLCC	28
71071	DMA Controller	8/16	8/10	DIP	48
				Ceramic DIP	48
				QFP	52
				PLCC	52
71082	Transparent Latch	8	8	DIP	20
				SOP	20
71083	Transparent Latch	8	8	DIP	20
				SOP	20
71084	Clock Pulse Generator/Driver	–	25	DIP	18
				SOP	20
71086	Bus Buffer/Driver	8	8	DIP	18
				SOP	20
71087	Bus Buffer/Driver	8	8	DIP	20
				SOP	20

CMOS System Support Products (cont)

Device, µPD	Name	Data Bits	Clock (MHz)	# Package	Pins
71088	System Bus Controller	–	8/10	DIP	20
				SOP	20
82C43	*Input/Output Expander	–	5	DIP	24
				Skinny DIP	24

NMOS System Support Products

Device, µPD	Name	Data Bits	Clock (MHz)	# Package	Pins
8155H	*256 x 8 RAM; I/O ports and timer	8	3 or 5	DIP	40
8156H	*256 x 8 RAM; I/O ports and timer	8	3 or 5	DIP	40
8237A	*Programmable DMA Controller	8	5	DIP	40
8243	*Input/Output Expander	–	5	DIP	24
8251A	*Programmable Communications Interface	8	3/5	DIP	28
8253	*Programmable Internal Timer	8	5	DIP	24
8255A	*Programmable Peripheral Interface	8	5	DIP	40
8257	*Programmable DMA Controller	8	5	DIP	40
8259A	*Programmable Interrupt Controller	8	5	DIP	28
8279	*Programmable Keyboard/Display Interface	–	5	DIP	40

Communications Controllers

Device, µPD	Name	Description	Data Rate	# Package	Pins
7201A	Multiprotocol Serial Communications Controller	Dual full-duplex serial channels; four DMA channels; programmable interrupt vectors; asychronous COP and BOP support; NMOS	1 Mb/s	DIP Ceramic DIP	40 40
72001	CMOS, Advanced Multiprotocol Serial Communications Controller	Functional superset of 8530; 8086/V30 interface; two full-duplex serial channels; two digital phase-locked loops; two baud-rate generators per channel; loopback test mode; short frame and mark idle detection	2.5 Mb/s	DIP QFP PLCC	40 52 52
72002	CMOS, Advanced Multiprotocol Serial Communications Controller	Low-cost, single-channel version of 72001; software compatible; direct interface to 8237 DMA. Not included in 1989-1990 IPD Data Book; refer to 72002 data sheet.	2.5 Mb/s	DIP QFP PLCC	40 44 44
72103	CMOS, HDLC Controller	Single full-duplex serial channel; on-chip DMA Controller. Not included in 1989-1990 IPD Data Book; refer to 72103 data sheet	4 Mb/s	DIP PLCC	64 68

Graphics Controllers

Device, µPD	Name	Description	Drawing Rate	# Package	Pins
7220A	High-Performance Graphics Display Controller	General-purpose, high-integration controller; hardwired support for lines, arc/circles, rectangles, and graphics characters; 1024x1024 pixel display with four planes	500 ns/dot	Ceramic DIP	40
72020	Graphics Display Controller	CMOS 7220A with 2M video memory; dual-port RAM control; write-masking on any bit; enhanced external synch	500 ns/dot	DIP QFP	40 52
72120	Advanced Graphics Display Controller	High-speed graphics operations including paint, area fill, slant, arbitrary angle rotate, up to 16x enlargement and reduction; dual-port RAM control; CMOS	500 ns/dot	PLCC QFP	84 94
72123	Advanced Graphics Display Controller II	Enhanced 72120; expanded command set; improved painting performance; laser printer interface controls; CMOS	400 ns/dot	PLCC QFP	84 94

Advanced Compression/Expansion Engine

Device, µPD	Name	Description	# Package	Pins
72185	Advanced Compression/ Expansion Engine	High-speed CCITT Group 3/4 bit-map image compression/expansion (A4 test chart, 400 PPI x 400 LPI in under 1 second); 32K-pixel line length; 32-megabyte image memory; on-chip DMA and refresh timing generator; CMOS	SDIP PLCC	64 68

Plastic unless ceramic (or cerdip) is specified.

Floppy-Disk Controllers

Device, µPD	Name	Description	Transfer Rate	# Package	Pins
765A/B	Floppy-Disk Controller	Industry-standard controller supporting IBM 3740 and IBM System 34 double-density format; enhanced 765B supports multitasking applications	500 kb/s	DIP	40
71065/66	Floppy-Disk Interface	Compatible with 765-family controllers and others; supports multiple data rates from 125 to 500 kb/s	500 kb/s	SOP SDIP	28 30
72064*	Floppy-Disk Controller	CMOS; All features of 72068 with complete AT register set. Pin compatible with WD 37C65/A/B but with higher performance DPLL and reliable multitasking operation	500 kb/s	PLCC QFP	44 52
72065/65B	CMOS Floppy-Disk Controller	100% 765A/B microcode compatible; compatible with 808x microprocessor families	500 kb/s	DIP PLCC QFP	40 44 52
72067	Floppy-Disk Controller	CMOS; 765A/B microcode compatible; clock generation/ switching circuitry; selectable write precompensation; digital phase-locked loop	500 kb/s	DIP QFP PLCC	48 52 52
72068	Floppy-Disk Controller	All features of the 72067 plus IBM-PC, PC/XT, PC/AT, or PS/2 style registers; 24-ma high-current drivers	500 kb/s	QFP PLCC	80 84
72069	Floppy-Disk Controller	All features of the 72067/68 with substitution of high-performance analog phase-locked loop for digital PLL	1 Mb/s	PLCC QFP	84 100

Hard-Disk Controllers

Device, µPD	Name	Description	Read/Write Clock	# Package	Pins
7261A/B	Hard-Disk Controller	Supports eight drives in SMD mode, four drives in ST506 mode; error correction and detection	23 MHz	Ceramic DIP	40
7262	Enhanced Small-Disk Interface (ESDI) Controller	Serial-mode ESDI compatible; controls up to seven drives; supports up to 80 heads; hard and soft-sector interfacing	18 MHz	Ceramic DIP	40
72061	CMOS Hard-Disk Controller	Supports SMD/SMD-E and ST506/412 type drives	24 MHz	DIP QFP PLCC	40 52 52
72111	Small Computer System Interface (SCSI) Controller	Selectable 8/16 data bus width; 16 high-level commands for reduced CPU load; single-command automatic execution; 5-Mb sync/async; CMOS	16 MHz	SDIP QFP PLCC	64 74 68

* Not included in 1989-90 IPD Data Book; refer to 72064 Data Sheet.

Digital Signal Processors

Device, µPD	Description	Instruction Cycle (ns)	Instruction ROM (Bits)	Data ROM (Bits)	Data RAM (Bits)	# Package	Pins
7720A	16-bit, fixed-point DSP; NMOS	244	512 x 23	510 x 13	128 x 16	DIP PLCC	28 44
77C20A	16-bit, fixed-point DSP; CMOS	244	512 x 23	510 x 13	128 x 16	DIP PLCC PLCC	28 28 44
77P20	16-bit, fixed-point DSP; CMOS	244	512 x 23 UVEPROM	510 x 13 UVEPROM	128 x 16	Cerdip	28
77C25	16-bit, fixed-point DSP; CMOS	122	2048 x 24	1024 x 16	256 x 16	DIP PLCC SOP	28 44 32
77P25	16-bit, fixed-point DSP; CMOS	122	2048 x 24 OTPROM	1024 x 16 OTPROM	256 x 16	DIP PLCC	28 44
			2048 x 24 UVEPROM	1024 x 16 UVEPROM	256 x 16	Cerdip	28
77220	24-bit, fixed-point DSP; CMOS	122	2048 x 32	1024 x 24	512 x 24	Ceramic PGA PLCC	68 68
77P220R	24-bit, fixed-point DSP; CMOS	122	2048 x 32 UVEPROM	1024 x 24 UVEPROM	512 x 24	Ceramic PGA	68
77230AR	32-bit, floating-point DSP; CMOS	150	2048 x 32	1024 x 32	1024 x 32	Ceramic PGA	68
77230AR-003	32-bit, floating-point DSP; CMOS; standard library software	150	n/a	n/a	n/a	Ceramic PGA	68
77P230R	32-bit, floating-point DSP; CMOS	150	2048 x 32 UVEPROM	1024 x 32 UVEPROM	1024 x 32	Ceramic PGA	68
77810	16-bit fixed-point modem DSP; CMOS	181	2048 x 24	1024 x 16	256 x 16	Ceramic PGA PLCC	68 68
7281	Image pipelined processor; NMOS	5-MHz clock	n/a	n/a	512 x 18	Ceramic DIP	40
72181	Image pipelined processor; CMOS	10-MHz clock	n/a	n/a	512 x 18	DIP QFP	40 68
9305	Support device for µPD7281 processors; CMOS	10-MHz clock	n/a	n/a	n/a	Ceramic PGA	132

Speech Processors

Device, µPD	Name	Technology	Clock (MHz)	Data ROM (Bits)	# Package	Pins
7730	ADPCM Speech Encoder/Decoder	NMOS	8	–	DIP	28
77C30	ADPCM Speech Encoder/Decoder	NMOS	8	–	DIP PLCC	28 44
7755	ADPCM Speech Synthesizer	CMOS	0.7	96K	DIP SOP	18 24
7756	ADPCM Speech Synthesizer	CMOS	0.7	256K	DIP SOP	18 24
77P56	ADPCM Speech Synthesizer	CMOS	0.7	256K OTPROM	DIP SOP	20 24
7757	ADPCM Speech Synthesizer	CMOS	0.7	512K	DIP SOP	18 24
7759	ADPCM Speech Synthesizer	CMOS	0.7	1024K external	DIP QFP	40 52

Plastic unless ceramic (or cerdip) is specified.

V-Series Development Tools

Part Number (Note 1)	Full Emulator	Full Emulator Probe	Mini-IE Emulator	Mini-IE Probe	Evaluation Boards	EPROM/OTP Device	Relocatable Assembler (Note 13)	C Compiler (Note 14)
μPD70136GJ-12	IE-70136-A016	EP-70136L-A (Note 2)	IE-70136-PC	EP-70136L-PC (Note 2)	DDK-70136	–	RA70136	CC70136
μPD70136GJ-16	IE-70136-A016	EP-70136L-A (Note 2)	IE-70136-PC	EP-70136L-PC (Note 2)	DDK-70136	–	RA70136	CC70136
μPD70136L-16	IE-70136-A016	EP-70136L-A	IE-70136-PC	EP-70136L-PC	DDK-70136	–	RA70136	CC70136
μPD70136L-12	IE-70136-A016	EP-70136L-A	IE-70136-PC	EP-70136L-PC	DDK-70136	–	RA70136	CC70136
μPD70136R-12	IE-70136-A016	EP-70136L-A (Note 3)	IE-70136-PC	EP-70136L-PC (Note 3)	DDK-70136	–	RA70136	CC70136
μPD70136R-16	IE-70136-A016	EP-70136L-A (Note 3)	IE-70136-PC	EP-70136L-PC (Note 3)	DDK-70136	–	RA70136	CC70136
μPD70208GF-8	IE-70208-A010	(Note 12)	EB-V40MINI-IE	–	EB-70208	–	RA70116	CC70116
μPD70208GF-10	IE-70208-A010	(Note 12)	EB-V40MINI-IE	–	EB-70208	–	RA70116	CC70116
μPD70208L-8	IE-70208-A010	IE-70000-2958	EB-V40MINI-IE	ADAPT68PGA 68PLCC (Note 4)	EB-70208	–	RA70116	CC70116
μPD70208L-10	IE-70208-A010	IE-70000-2958	EB-V40MINI-IE	ADAPT68PGA 68PLCC (Note 4)	EB-70208	–	RA70116	CC70116
μPD70208R-8	IE-70208-A010	IE-70000-2959	EB-V40MINI-IE	(Note 4)	EB-70208	–	RA70116	CC70116
μPD70208R-10	IE-70208-A010	IE-70000-2959	EB-V40MINI-IE	(Note 4)	EB-70208	–	RA70116	CC70116
μPD70216GF-8	IE-70216-A010	(Note 12)	EB-V50MINI-IE	–	EB70216	–	RA70116	CC70116
μPD70216GF-10	IE-70216-A010	(Note 12)	EB-V50MINI-IE	–	EB70216	–	RA70116	CC70116
μPD70216L-8	IE-70216-A010	IE-70000-2958	EB-V50MINI-IE	ADAPT68PGA 68PLCC (Note 4)	EB70216	–	RA70116	CC70116
μPD70216L-10	IE-70216-A010	IE-70000-2958	EB-V50MINI-IE	ADAPT68PGA 68PLCC (Note 4)	EB70216	–	RA70116	CC70116
μPD70216R-8	IE-70216-A010	IE-70000-2959	EB-V50MINI-IE	(Note 4)	EB70216	–	RA70116	CC70116
μPD70216R-10	IE-70216-A010	IE-70000-2959	EB-V50MINI-IE	(Note 4)	EB70216	–	RA70116	CC70116
μPD70320GJ	IE-70320-A008	EP-70320GJ (Note 5)	EB-V25MINI-IE-P	EP-70320GJ (Note 6)	DDK-70320	–	RA70320	CC70116
μPD70320GJ-8	IE-70320-A008	EP-70320GJ (Note 5)	EB-V25MINI-IE-P	EP-70320GJ (Note 6)	DDK-70320	–	RA70320	CC70116
μPD70320L	IE-70320-A008	EP-70320L	EB-V25MINI-IE-P	(Note 7)	DDK-70320	–	RA70320	CC70116
μPD70320L-8	IE-70320-A008	EP-70320L	EB-V25MINI-IE-P	(Note 7)	DDK-70320	–	RA70320	CC70116
μPD70322GJ	IE-70320-A008	EP-70320GJ (Note 5)	EB-V25MINI-IE-P	EP-70320GJ (Note 6)	DDK-70320	–	RA70320	CC70116
μPD70322GJ-8	IE-70320-A008	EP-70320GJ (Note 5)	EB-V25MINI-IE-P	EP-70320GJ (Note 6)	DDK-70320	–	RA70320	CC70116
μPD70322L	IE-70320-A008	EP-70320L	EB-V25MINI-IE-P	(Note 7)	DDK-70320	70P322K (Note 10)	RA70320	CC70116
μPD70322L-8	IE-70320-A008	EP-70320L	EB-V25MINI-IE-P	(Note 7)	DDK-70320	70P322K (Note 10)	RA70320	CC70116
μPD70325GJ-8	IE-70325-A008	EP-70320GJ (Note 5)	(Note 12)	(Note 12)	DDK-70325	–	RA70320	CC70116
μPD70325GJ-10	IE-70325-A008 (Note 8)	EP-70320GJ (Note 5)	(Note 12)	(Note 12)	DDK-70325	–	RA70320	CC70116
μPD70325L-8	IE-70325-A008	EP-70320L	(Note 12)	(Note 12)	DDK-70325	–	RA70320	CC70116
μPD70325L-10	IE-70325-A008 (Note 8)	EP-70320L	(Note 12)	(Note 12)	DDK-70325	–	RA70320	CC70116

V-Series Development Tools (cont)

Part Number (Note 1)	Full Emulator	Full Emulator Probe	Mini-IE Emulator	Mini-IE Probe	Evaluation Boards	EPROM/OTP Device	Relocatable Assembler (Note 10)	C Compiler (Note 11)
μPD70327GJ-8 (Note 9)	IE-70320-A008	EP-70320GJ (Note 5)	EB-V25MINI-IE-P	EP-70320GJ (Note 6)	–	–	RA70320	CC70116
μPD70327L-8 (Note 9)	IE-70230-A008	EP-70320L	EB-V25MINI-IE-P	(Note 7)	–	–	RA70320	CC70116
μPD70330GJ-8	IE-70330-A008	EP-70320GJ (Note 5)	EB-V35MINI-IE-P	EP-70320GJ (Note 6)	DDK-70330	–	RA70320	CC70116
μPD70330L-8	IE-70330-A008	EP-70320L	EB-V35MINI-IE-P	(Note 7)	DDK-70330	–	RA70320	CC70116
μPD70332GJ-8	IE-70330-A008	EP-70320GJ (Note 5)	EB-V35MINI-IE-P	EP-70320GJ (Note 6)	DDK-70330	–	RA70320	CC70116
μPD70332L-8	IE-70330-A008	EP-70320L	EB-V35MINI-IE-P	(Note 7)	DDK-70330	70P322K (Note 10)	RA70320	CC70116
μPD70335GJ-8	IE-70335-A008	EP-70320GJ (Note 5)	(Note 12)	(Note 12)	DDK-70330	–	RA70320	CC70116
μPD70335GJ-10	IE-70335-A008 (Note 8)	EP-70320GJ (Note 5)	(Note 12)	(Note 12)	DDK-70330	–	RA70320	CC70116
μPD70335L-8	IE-70335-A008	EP-70320L	(Note 12)	(Note 12)	DDK-70330	–	RA70320	CC70116
μPD70335L-10	IE-70335-A008 (Note 8)	EP-70320L	(Note 12)	(Note 12)	DDK-70330	–	RA70320	CC70116
μPD70337GJ-8 (Note 9)	IE-70330-A008	EP-70320GJ (Note 5)	EB-V35MINI-IE-P	EP-70320GJ (Note 6)	–	–	RA70320	CC70116
μPD70337L-8 (Note 9)	IE-70330-A008	EP-70320L	EB-V35MINI-IE-P	(Note 7)	–	–	RA70320	CC70116
μPD79011GJ-8 (Note 11)	IE-70320-A008	EP-70320GJ (Note 5)	(Note 12)	(Note 12)	–	–	RA70320	CC70116
μPD79011L-8 (Note 11)	+IE-70320-RTOS	EP-70320L	(Note 12)	(Note 12)	–	–	RA70320	CC70116
μPD79021L-8 (Note 11)	IE-70330-A008 +IE-70330-RTOS	EP-70320L	(Note 12)	(Note 12)	–	–	RA70320	CC70116

Notes:

(1) Packages:

Package	Description
GF	80-pin plastic QFP
GJ	74-pin or 94-pin plastic QFP
K	84-pin ceramic LCC with window
L	68-pin or 84-pin plastic LCC
R	68-pin PGA

(2) The EP-70136GL-A and EP-70136L-PC contain both a 68-pin PLCC probe and an adapter which converts the 68-pin PLCC probes to a 74-pin QFP footprint.

(3) 68-pin PGA parts are supported by using the EP-70136L-A PLCC probe or EP-70136L-PC PLCC probe, plus a PLCC socket with a PGA-pinout. A PLCC socket of this type is supplied with the EP-70136L-A.

(4) The EB-V40 MINI-IE and EB-V50 MINI-IE support PGA packages directly; the ADAPT68PGA68PLCC adapter converts the PGA-pinout on the MINI-IE to a PLCC footprint. This adapter is supplied with the MINI-IE.

(5) The EP-70320GJ is an adapter to the EP-70320L, which converts 84-pin PLCC probes to a 94-pin QFP footprint. For GJ parts, both the PLCC probe and the adapter are needed.

(6) The EP-70320GJ adapter can be used to convert the supplied 84-pin PLCC cable of the EB-V25 MINI-IE-P or EB-V35 MINI-IE-P to a 94-pin QFP.

(7) The EB-V25 MINI-IE-P and EB-V35 MINI-IE-P are supplied with an 84-pin PLCC cable.

(8) At the current time, the emulators for the μPD70325 and μPD70335 are specified to 8 MHz. Contact your local NEC Sales Office for the latest information on 10 MHz emulation.

(9) Development for the μPD70327 or μPD70337 can be done using the appropriate μPD70320 or μPD70330 tools; however, debugging the programs in the Software Guard mode is not supported at this time.

(10) The μPD70P322K EPROM device can be used for both μPD70322 and μPD70332 emulation. The μPD70P322K EPROM device can be programmed by using the PA-70P322L Programming Adapter and the PG-1500 EPROM Programmer.

V-Series Development Tools (cont)

Notes (continued):

(11) For emulation of μPD79011 or μPD79021, the base emulator (IE-70320 or IE-70330) plus Real-Time Operating System software IE-70320-RTOS or IE-70330-RTOS) is required.

(12) This emulation option is not currently supported, but may be available in the future. Contact your local NEC Sales Office for further information.

(13) The following relocatable assemblers are available:

RA70116-D52 RA70116-VVT1 RA70116-VXT1	For V20®/V30®/ V40™/V50™	(MS-DOS®) (VAX/VMS™) (VAX/UNIX™ 4.2 BSD or Ultrix™)
RA70136-D52 RA70136-VVT1 RA70136-VXT1	For V33™	(MS-DOS) (VAX/VMS) (VAX/UNIX 4.2 BSD or Ultrix)
RA70320-D52 RA70320-VVT1 RA70320-VXT1	For V25™ and V35™	(MS-DOS) (VAX/VMS) (VAX/UNIX 4.2 BSD or Ultrix)

(14) The following C compilers are available:

CC70116-D52 CC70116-VVT1 CC70116-VXT1	For V20/V30/ V40/V50 and V25/V35	(MS-DOS) (VAX/VMS) (VAX/UNIX 4.2 BSD or Ultrix)
CC70136-D52 CC70136-VVT1 CC70136-VXT1	For V33	(MS-DOS) (VAX/VMS) (VAX/UNIX 4.2 BSD or Ultrix)

1

V20 and V30 are registered trademarks of NEC Corporation.
V25, V33, V35, V40 and V50 are trademarks of NEC Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.
VAX, VMS and Ultrix are trademarks of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories.

DSP and Speech Development Tools

Part Number (Note 7)	Emulator	Evaluation Board	Assembler (Note 1)	Simulator (Note 2)	EPROM/OTP Device	PG-1500 Adapter (Note 3)
μPD7720AC	EVAKIT-7720B	–	ASM77	SIM77	μPD77P20D	(Note 5)
μPD7720AL	EVAKIT-7720B (Note 4)	–	ASM77	SIM77	–	–
μPD77P20D	EVAKIT-7720B	–	ASM77	SIM77	–	–
μPD77C20AC	EVAKIT-7720B	–	ASM77	SIM77	μPD77P20D	(Note 5)
μPD77C20AL	EVAKIT-7720B (Note 4)	–	ASM77	SIM77	–	–
μPD77C20ALK	EVAKIT-7720B (Note 4)	–	ASM77	SIM77	–	–
μPD77220L	EVAKIT-77220	–	RA77230	SM77230	–	–
μPD77220R	EVAKIT-77220	–	RA77230	SM77230	μPD77P220R	PA-77P230R
μPD77P220R	EVAKIT-77220	–	RA77230	SM77230	–	PA-77P230R
μPD77230AR	EVAKIT-77230	DDK-77230	RA77230	SM77230	μPD77P230R	PA-77P230R
μPD77P230R	EVAKIT-77230	DDK-77230	RA77230	SM77230	–	PA-77P230R
μPD77C25C	EVAKIT-77C25	–	RA77C25	–	μPD77P25C/D	PA-77P25C
μPD77C25L	EVAKIT-77C25 (Note 4)	–	RA77C25	–	μPD77P25L	–
μPD77P25C	EVAKIT-77C25	–	RA77C25	–	–	PA-77P25C
μPD77P25D	EVAKIT-77C25	–	RA77C25	–	–	PA-77P25C
μPD77P25L	EVAKIT-77C25 (Note 4)	–	RA77C25	–	–	–
μPD7755C	NV-300 System	EB-7759	–	–	μPD77P56CR	PA-77P56C
μPD7755G	NV-300 System	EB-7759 (Note 6)	–	–	μPD77P56G	PA-77P56C
μPD7756C	NV-300 System	EB-7759	–	–	μPD77P56CR	PA-77P56C
μPD7756G	NV-300 System	EB-7759 (Note 6)	–	–	μPD77P56G	PA-77P56C
μPD77P56CR	NV-300 System	EB-7759	–	–	–	PA-77P56C
μPD77P56G	NV-300 System	EB-7759 (Note 6)	–	–	–	PA-77P56C
μPD7757C	NV-300 System	EB-7759	–	–	–	–
μPD7757G	NV-300 System	EB-7750 (Note 6)	–	–	–	–
μPD7759C	NV-300 System	EB-7759	–	–	–	–
μPD7759GC	NV-300 System	EB-7759	–	–	–	–
μPD77810L	IE-77810	–	RA77810	–	–	–
μPD77810R	IE-77810	–	RA77810	–	–	–

Notes:

(1) The following assemblers are available:

Part Number	Description
ASM77-D52	Assembler for 7720 (MS-DOS®)
RA77C25-D52	Assembler for 77C25 (MS-DOS)
RA77C25-VVT1	Assembler for 77C25 (VAX/VMS™)
RA77230-D52	Assembler for 77230 (MS-DOS)
RA77230-VVT1	Assembler for 77230 (VAX/VMS)
RA77230-VXT1	Assembler for 77230 (VAX/UNIX™ 4.2 BSD or Ultrix™)

(2) The following simulators are available:

Part Number	Description
SIM77-D52	Simulator for 7720 (MS-DOS)
SM77230-VVT1	Simulator for 77230 (VAX/UNIX)
SM77230-VXT1	Simulator for 77230 (VAX/UNIX™ 4.2 BSD or Ultrix)

(3) By using the specified adapter, the NEC PG-1500 EPROM programmer can be used to program the EPROM/OTP device.

(4) Please check with your NEC Sales Representative on the availability of a PLCC emulation probe.

(5) The μPD77P20D can be programmed using the EVAKIT-7720B.

(6) The EB-7759 comes with an emulation probe for only the 18-pin DIP.

(7) Packages:

Package	Description
C	18, 28, or 40-pin plastic DIP
D	28-pin ceramic DIP
G	24-pin plastic SOP
GC	52-pin plastic QFP
L	44-or 68-pin PLCC
LK	28-pin PLCC
R	68-pin ceramic PGA

MS-DOS is a registered trademark of Microsoft Corporation.
VAX, VMS, and Ultrix are trademarks of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories.

NEC

Section 2
Reliability and Quality Control

Introduction

As large-scale integration reaches a higher level of density, the reliability of individual devices imposes a more profound impact on system reliability. Great emphasis has thus been placed on assuring device reliability.

Conventionally, performing reliability tests and attaining feedback from the field are the only methods by which reliability has been monitored and measured. At these higher levels of LSI density, however, it is increasingly difficult to activate all of the internal circuit elements in a device, moreover, to detect the degradation of those elements by measuring characteristics across external terminals. As a result, testing alone may not provide enough information to insure today's demanding reliability requirements. A different philosophy and methodology is needed for reliability assurance.

In order to guarantee and improve a high level of reliability for large-scale integrated circuits, it is essential to build quality and reliability into the product. Then, conventional testing can be performed to confirm that the product demonstrates acceptable reliability.

Built-In Quality and Reliability

NEC has introduced the concept of total quality control (TQC) across its entire semiconductor product line for implementing this philosophy. Rather than performing only a few simple quality inspections, quality control is distributed into each process step and then summed to form a consolidated system. TQC involves workers, engineers, quality control staffs, and all levels of management in company-wide activities. Please see Figure 1 for the quality control system flowchart. Through TQC, NEC builds quality into the product and thus can assure high reliability. Additionally, NEC has introduced a pre-screening method into the production line for eliminating potentially defective units. This combination of building quality in and screening projected early failures out has resulted in superior quality and excellent reliability.

Technology Description

Most large-scale integrated circuits utilize high density MOS technology. State-of-the-art high performance has been achieved by improving fine-line generation techniques. By reducing physical parameters, circuit density and performance increase while active circuit power dissipation decreases. The data presented here shows that this advanced technology, combined with the practice of TQC, yields products as reliable as those from previous technologies.

Approaches to Total Quality Control

TQC activities are geared towards total satisfaction of the customer. The success of these activities is dependent upon the total commitment of management to enhancing employee development, maintaining a customer-first attitude, and fulfilling community responsibilities.

First, the quality control function is embedded into each process. This method enables early detection of possible causes of failure and immediate feedback.

Second, the reliability and quality assurance policy reflects the beliefs and practices of the entire organization. This enables companywide quality control activities: at NEC, everyone is involved with the concept and methodology of total quality control.

Third, there is an ongoing research and development effort to set even higher standards of device quality and reliability.

Fourth, extensive failure analysis is performed periodically and appropriate corrective actions are taken as preventative measures. Process control is based on statistical data gathered from this analysis.

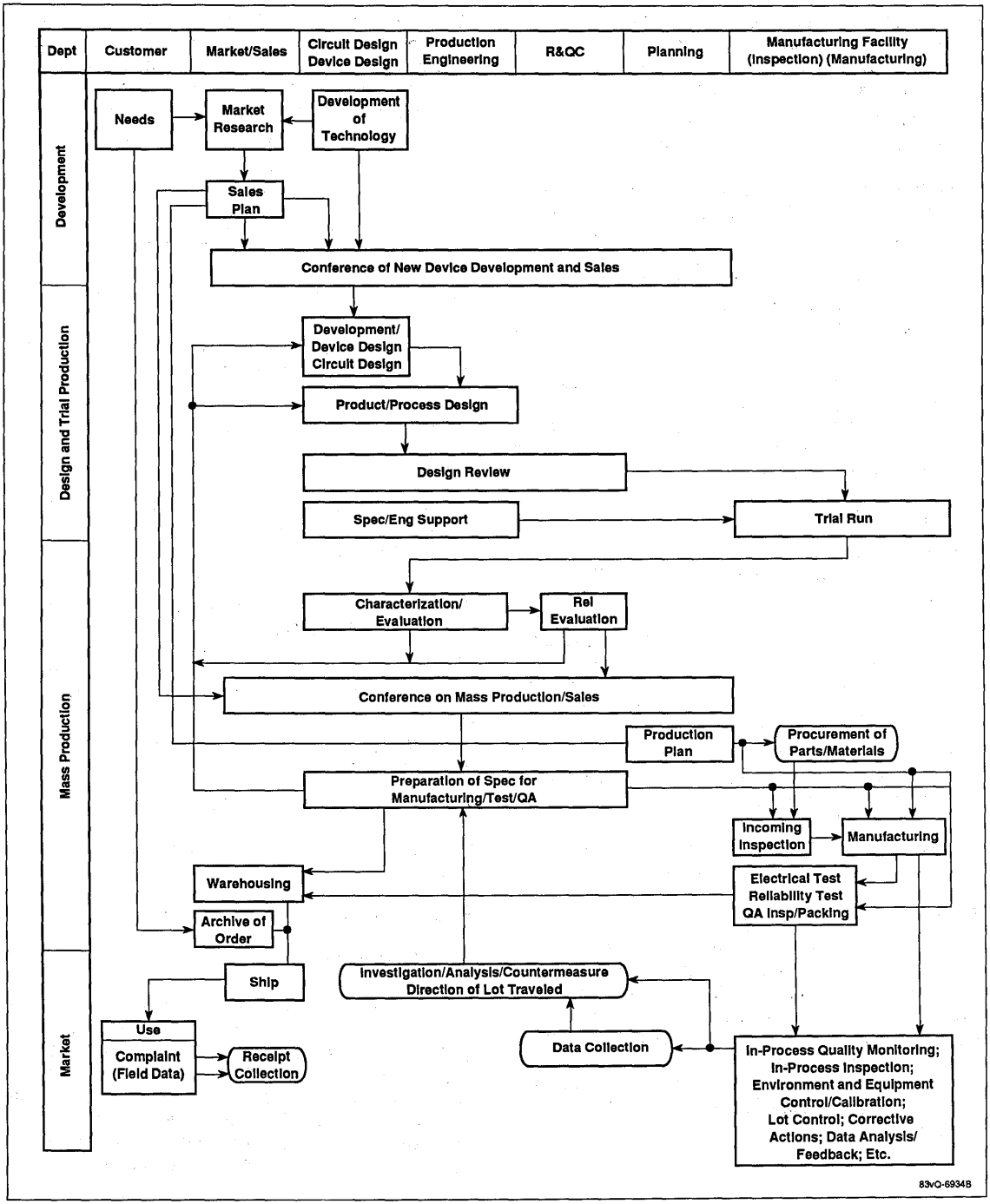
The new standard is continuously upgraded, and the iterative process continues. The goal is to maintain the superior product quality and reliability that has become synonymous with the NEC name.

Zero Defect Activities. One of the activities that involves every level of the NEC staff in quality control is the Zero Defects (ZD) Program. As the name implies, the purpose of the ZD program is to minimize, if not eradicate, defects due to controllable causes. Such activities must involve each and every worker and can be most effective when pursued by groups of workers. The groups of workers are organized by consideration of the following:

- A group must have a target to pursue
- Several groups can be organized to pursue the common target
- Each group must have a responsible person
- Each group is well supported

The item of the group target is to be selected among items relating to specifications, inspections, operation standards, and so forth. When data made in the past is available, it is used to make a Pareto diagram which is reviewed for selection of the item most conducive to quality improvement. Records are analyzed and compared with the target, in order to compute the numerical equivalents of the defects. Action is then taken to control these defects as required.

Figure 1. Quality Control System Flowchart



Statistical Approach. Another approach to quality control is the use of statistical analysis. NEC has been utilizing statistical analysis at each stage of LSI production development, trial runs, and mass production in order to build and maintain product quality. Some of the methods for implementing this statistical approach are:

- Design of experiments
- Control charts
- Data analysis: Variance, correlation, regression, multivariance, etc.
- Cp, Cpk study: Variables and attributes data (Normally, study is done on a monthly basis)

Process control sheets and other QC tools are used to monitor various important parameters such as Cp, Cpk, X, $\overline{X}$, $\overline{X}$-R, electrical parameters, pattern dimensions, bond strength, test percentage defects, etc.

The results of these studies are watched by the production staff, QC Engineers, and other responsible engineers. If any out-of-control or out-of-specification limit is observed, quick action is taken in accordance with corrective action procedures.

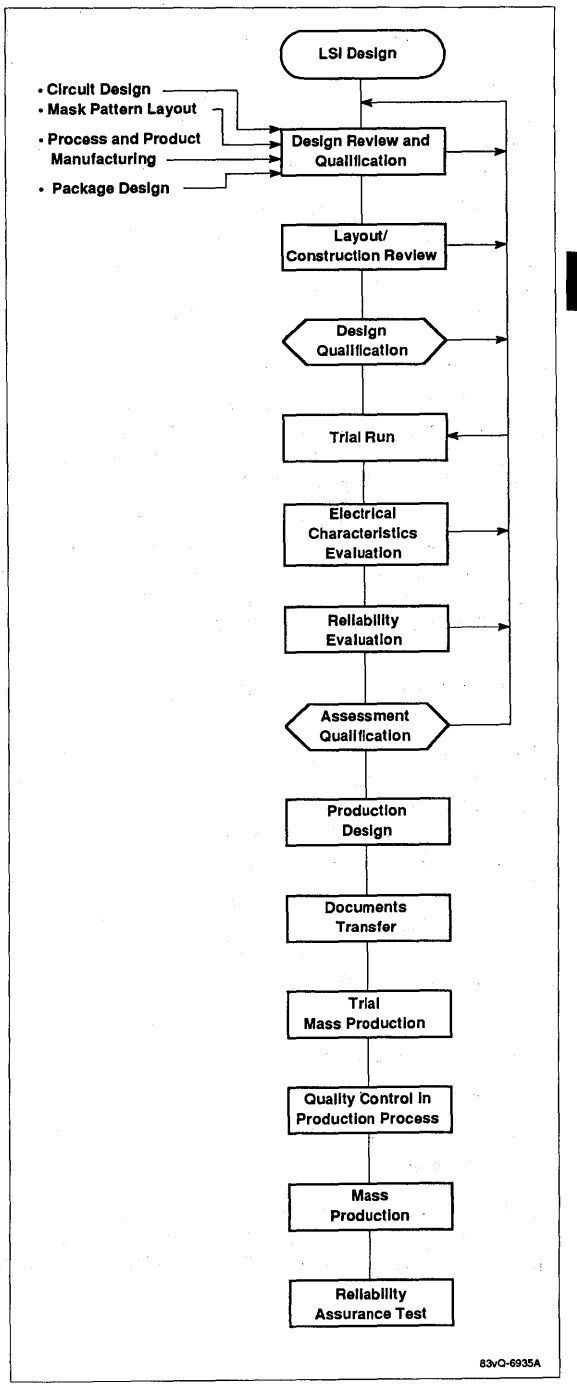
Implementation of Quality Control

Building quality into a product requires early detection of possible causes of failure at each process step, then immediate feedback to remove these causes. A fixed station quality inspection is often lacking in immediate feedback; it is therefore necessary to distribute quality control functions to each process step—including the conceptual stage. Following is a breakdown of the significant steps at which NEC has implemented these functions:

- Product development
- Incoming material inspection
- Wafer processing
- Chip mounting and packaging
- Electrical testing and thermal aging
- Outgoing material inspection
- Reliability testing
- Process/product changes

New Product Development Phase. The product development phase includes conception of a product, review of the device proposal, physical element design and organization, engineering evaluation, and finally, transfer of the product to manufacturing. Quality and reliability are considered at every step. The new product development flow is shown in Figure 2.

Figure 2. New Product Development Flow



2

Design. Design plays an extremely important role in determining the product quality and reliability. NEC believes that the foundation of device quality is determined at the design stage. The four steps involved in the design of LSI devices are circuit design, mask pattern layout, process and product manufacturing, and package design. Design standards and the standardization of design steps have been established to maximize quality and reliability.

Design Review. After completion of the design, a design review is performed. In this review, the design is compared with design standards and other factors which influence the reliability and quality. If necessary, modification or redesign is then performed. NEC believes that the design review is very essential for not only newly designed products but also for product modifications.

Trial Production/Evaluation/Mass Production. When the design passes the design review successfully, a trial run is carried out. The trial run is evaluated for the products' characteristics and quality/reliability.

Thorough evaluation is carried out by generating samples in which process conditions—ones that cause characteristic factors to change in mass production—are varied deliberately. In addition, reliability tests are conducted for durability, stress resistance, etc., to insure sufficient quality and reliability.

If no problems are found at this stage, the product is approved, after which mass production is possible.

Prior to the transfer, the production Design Department prepares a production schedule, including the reliability and quality control steps relating to the production. Even after the mass-production has started, the standards for those production and control steps are always reexamined for improvements.

Incoming Material Inspection. NEC has various programs to control incoming materials. Some are:

- Vendor/material qualification system
- Purchasing specifications for materials
- Incoming materials inspection
- Inspection data feedback
- Quality meetings with vendor
- Vendor audits

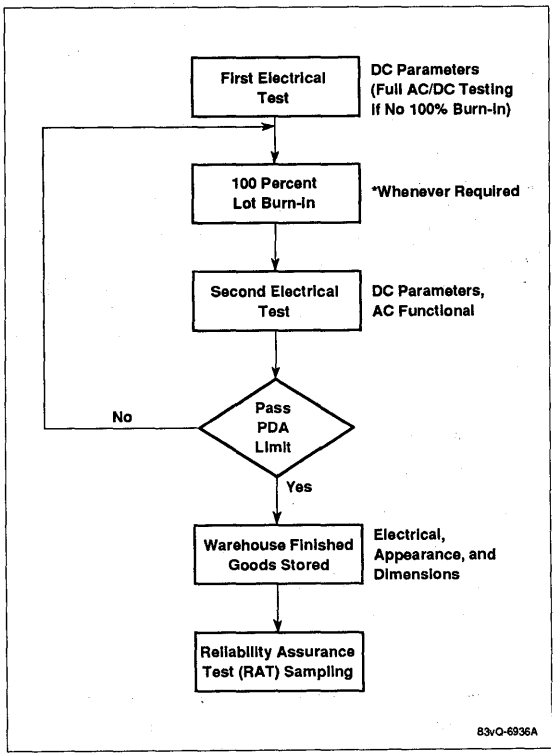
If any parts or materials are rejected at incoming inspection, they are returned to the vendor with a rejection notification form which specifies the failure items and modes. The results of these inspections are used to rate the vendors for future purchasing.

In-process Quality Inspections. Typical in-process quality inspections done at the wafer fabrication, chip mounting and packaging, and device testing stage are listed in Appendix 1.

Electrical Testing and Screening. A flowchart of the typical infant mortality screening (when required) and electrical testing is depicted in Figure 3.

At the first electrical test, DC parameters are tested according to the electrical specifications on 100% of each lot. This is a prescreening prior to any infant mortality test. At the second electrical test, AC functional tests as well as DC parameter tests are performed on 100% of each lot. If the percentage of defective units exceeds the limit, the lot is subjected to rescreen. During this time, the defective units undergo failure analysis, the results of which are fed back into the process through corrective actions.

Figure 3. Electrical Testing and Screening



Outgoing Inspection. Prior to warehouse storage, lots are subjected to an outgoing inspection according to the following sampling plan.

• Electrical test:	DC parameters LTPD	3%
	Functional test LTPD	3%
• Appearance:	Major LTPD	3%
	Minor LTPD	7%

Reliability Assurance Tests. Samples are continually taken prior to shipment and subjected to monitoring reliability tests. They are taken from similar process groups, so it may be assumed that the samples' reliability is representative of the reliability of the group.

Reliability Testing

Reliability is defined as the characteristics of an item expressed by the probability that it will perform a required function under stated conditions for a stated period of time. This involves the concepts of probability, the definition of required function(s), and the critical time used in defining the reliability.

Definition of a required function, by implication, treats the definition of a failure. Failure is defined as the termination of the ability of a device to perform its required function. A device is said to have failed if it shows the inability to perform within guaranteed parameters as given in an electrical specification.

Discussion of reliability and failure can be approached in two ways: with respect to systems or to individual devices. Important considerations are the constant failure period, the early failure (infant mortality) period, and overall reliability level.

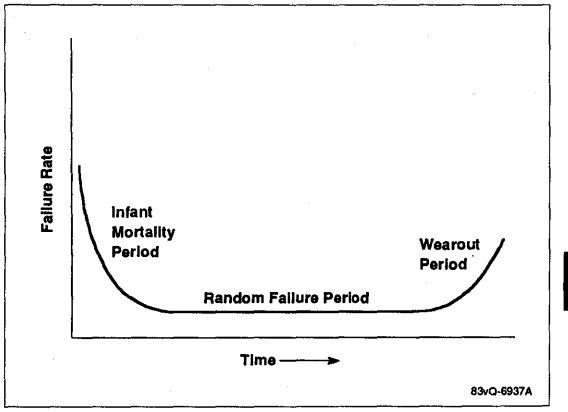
With regard to individual devices, areas of prime interest include specific failure mechanisms, failures in accelerated tests, and failures in screening tests.

The accumulation of normal device failure rates constitutes the expected failure rate of the system hardware: the probability that no device failures will occur in a system is the product of each device's probability that it will not fail. The failure rate of system hardware is then the sum of the failure rates of the components used to construct the system.

Life Distribution

The fundamental principles of reliability engineering predict that the failure rate of a group of devices will follow the well-known bathtub curve in Figure 4. The curve is divided into three regions: infant mortality, random failures, and wearout failures.

Figure 4. Reliability Life (Bathtub) Curve



Infant mortality, as the name implies, represents the early-life failures of devices. These failures are usually associated with one or more manufacturing defects.

After some period of time, the failure rate reaches a low value. This is the random failure portion of the curve, representing the useful portion of the life of a device. During this random failure period, there is a decline in the failure rate due to the depletion of potential random failures from the general population.

The wearout failures occur at the end of the device's useful life. They are characterized by a rapidly rising failure rate over time as devices wear out both physically and electrically.

Thus, for a device that has a very long life expectancy compared to the system which contains it, the areas of concern will be the infant mortality and the random failure portions of the bathtub curve.

Failure Distribution at NEC

In an effort to eliminate infant mortality failures, NEC subjects its products to production burn-in whenever necessary. This burn-in is performed at an elevated temperature for 100 percent of the lots involved and is designed to remove the potentially defective units.

To study the random failure population, integrated circuits returned to NEC from the field undergo extensive failure analysis at respective NEC Manufacturing Divisions. Failure mechanisms are identified and data fed back to cognizant Production and Engineering groups.

This data coupled with in-line data is then used to introduce corrective actions and quality improvement measures.

After elimination of early device failures, a system will be left to the random failure rate of its components. Thus, in order to make proper projections of the failure rate of the system in the operating environment, failure rates must be predicted for the system's components.

Infant Mortality Failure Screening

Establishing infant mortality screening requires knowledge of the likely failure mechanisms and their associated activation energies. The most likely problems associated with infant mortality failures are generally manufacturing defects and process anomalies. These defects and anomalies generally consist of contamination, cracked chips, wire bond shorts, or bad wire bonds. Since these describe a number of possible mechanisms, any one of which might predominate at a given time, the activation energy for infant mortality varies considerably.

Correspondingly, the effectiveness of a screening condition—preferably at some stress level in order to shorten the screening time—varies greatly with the failure mechanism. For example, failures due to ionic contamination have an activation energy of approximately 1.0 eV. Therefore, a 15-hour stress at 125°C junction temperature would be the equivalent of approximately 314 days of operation at a junction temperature of 55°C. On the other hand, failures due to oxide defects have an activation energy of approximately 0.3 eV, and a 15-hour stress at 125°C junction temperature would be the equivalent of approximately four day's operation at 55°C junction temperature. As indicated by this situation, the conditions and duration of infant mortality screening must be strongly dependent on the allowable component, hence system, failures in the field, as well as the economic factors involved.

Empirical data gathered at NEC indicates that early failures (if any) occur after less than 4 hours of stress at 125°C ambient temperature. This fact is supported by the bathtub curve created from the life test results of the same lots, where the failure rate shows a random distribution as opposed to a decreasing failure rate that runs into the random failure region.

Whenever necessary, NEC has adopted this initial infant mortality burn-in at 125°C as a standard production screening procedure. As a result, the field reliability of NEC devices is an order of magnitude higher than the goal set for NEC's integrated circuit products.

NEC believes it is imperative that failure modes associated with infant mortality screens be understood and fixed at the manufacturing level. If such failures can be minimized or eliminated, and countermeasures appropriately monitored, then such screens can be eliminated.

Long-Term Failure Rate

NEC's long-term failure rate goal, based on the mask and process design, is confirmed by life testing using the following conditions:

- A minimum of 1.2 million device hours (= sample size x test period) at 125°C should be accumulated to obtain the accuracy necessary for predicting a failure rate of 0.02% per 1000 hours at 55°C with a 60% confidence level.
- A minimum of 3 million device hours at 125°C should be accumulated to obtain the accuracy necessary for predicting a failure rate of 0.01% per 1000 hours at 55°C with a 60% confidence level.

Accelerated Reliability Testing

NEC performs extensive reliability testing both at pre-production and post-production levels to insure that its products meet the minimum expectations set by NEC. Accelerated reliability testing results are then used to quantitatively monitor the reliability.

As an example, assume that an electronic system contains 1000 integrated circuits and can tolerate 1 percent system failures per month. The failure rate per component is:

$$\frac{1\%\ \text{Failures}}{720\ \text{Hours} \times 1000\ \text{Pcs.}} = .0014\ \frac{\%\ \text{Failures}}{1000\ \text{Hrs}}$$

or 14 FITs

To demonstrate this failure rate, note that 14 FITs correspond to one failure in about 85 devices during an operating test of 10,000 hours. It is quickly apparent that a test condition is required to accelerate the time-to-failure in a predictable and understandable way. The implicit requirement for the accelerated stress test is that the relationship between the accelerated stress testing condition and the condition of actual use be known.

A most common time-to-failure relationship involves the effect of temperature, which accelerates many physio-chemical reactions which may lead to device failure. Other environmental conditions are voltage, current, humidity, vibration, or some combination of these. Appendix 2 lists typical reliability assurance tests performed at NEC for molded integrated circuits. Figure 5 shows the results of some of these tests for various process types.

High-Temperature Operating Life Test. This test is used to accelerate failure mechanisms by operating devices at an elevated temperature of 125°C. The data obtained is translated to a lower temperature by using the Arrhenius relationship.

Figure 5. Typical Reliability Test Results

	HTB	T/H	PCT	T/C
Micro:[1]				
NMOS	7/19113 (15 FIT)	15/9315	0/11752	—
CMOS	3/11892 (5.4 FIT)	2/7293	8/9476	—
Memory:	[HTOL]			
DRAM[2]	10/10052 (19 FIT)	0/9958	0/5880	1/2995
SRAM[3]	1/10421	2/8142	0/8768	—
1 MEG DRAM[4]	38/14300 (115 FIT)	0/3634	1/3060	1/1780
Asic:[5]				
CMOS	2/3506 (33 FIT)	1/1111	1/4764	4/2680
ECL	0/1080 (8.4 FIT)	—	—	0/141
BiCMOS	1/895 (18 FIT)	0/1073	0/935	0/1781

Information has been extracted from NEC Report Numbers:
[1] TRQ-89-05-0030 [2] TRQ-89-01-0021
[3] TRQ-88-09-0008 [4] TRQ-89-01-0020
[5] TRQ-89-04-0025

High-Temperature and High-Humidity Test. Semiconductor integrated circuits are highly sensitive to the effect of humidity causing electrolytic corrosion between biased lines. The high-temperature and high-humidity test is performed to detect failure mechanisms that are accelerated by these conditions, such as leakage-related problems and drifts in device parameters due to process instability.

High-Temperature Storage Test. Another common test is the high-temperature storage test, in which devices are subjected to elevated temperatures with no applied bias. This test is used to detect process instability and stress migration problems.

Environmental Tests. Other environmental tests are performed to detect problems related to the package, material, susceptibility to extremes in environment, and problems related to usage of the devices.

Failure Rate Calculation/Prediction

When predicting the failure rate at a certain temperature from accelerated life test data, the activation energies of the failure mechanisms involved should be considered. This is done whenever the exact cause of failures is known through failure analyses results.

In some cases, an average activation energy is assumed in order to accomplish a quick first order approximation. NEC assumes an average activation energy of 0.7 eV for such approximations. This average value has been assessed from extensive reliability test results and yields a conservative failure rate.

Since most semiconductor failures are temperature dependent, the Arrhenius relationship is used to normalize failure rate predictions at a system operation temperature of 55°C. It assumes that temperature dependence is an exponential function that defines the probability of occurrence, and that the degradation of a performance parameter is linear with time. The Arrhenius model includes the effects of temperature and activation energies of the failure mechanisms in the following Arrhenius equation:

$$A = \exp \frac{-E_A (T_{J1} - T_{J2})}{k(T_{J1})(T_{J2})}$$

Where:

A = Acceleration factor
E_A = Activation energy
T_{J1} = Junction temperature (in K) at $T_{A1} = 55°C$
T_{J2} = Junction temperature (in K) at $T_{A2} = 125°C$
k = Boltzmann's constant $= 8.62 \times 10^{-5}$ eV/K.

Because the thermal resistance and power dissipation of a particular device type cannot be ignored, junction temperatures (T_{J1} and T_{J2}) are used instead of ambient temperatures (T_{A1} and T_{A2}). We calculate junction temperatures using the following formula:

$T_J = T_A$ + (Thermal Resistance) (Power Diss. at T_A)

In order to estimate long term failure rate, the acceleration factor must be used to determine the simulated test time. From the high temperature operating life test results, failure rates can then be predicted at a 60% confidence level using the following equation:

$$L = \frac{X^2 10^5}{2T}$$

Where:

L = Failure rate in %/1000 hours
$*X^2$ = The tabular value of chi-squared distribution at a given confidence level and calculated degrees of freedom (2f + 2, where f = number of failures)
T = # of equivalent device hours
= (# of devices) x (# of test hours) x (acceleration factor)

*Since the failures of concern here are the random, not the infant mortality failures (that is, the end of the downward slope and the middle-constant-section of the bathtub curve in Figure 4), X^2 is determined assuming a one-sided, fixed time test.

Another method of expressing failures is in FITs (failures in time). One FIT is equal to one failure in 10^9 hours. Since L is already expressed as %/1000 hours (10^{-5} failure/hr), an easy conversion from %/1000 hours to FIT would be to multiply the value of L by 10^4.

EXAMPLE: A sample of 960 pieces was subjected to 1000 hours 125°C burn-in. One reject was observed. Given that the acceleration factor was calculated to be 34.6 using the Arrhenius equation, what is the failure rate normalized to 55°C using a confidence level of 60%? Express the failure rate in FIT:

Solution:

For $n = 2f + 2 = 2(1) + 2 = 4$, $X^2 = 4.046$.

$$\text{Then } L = \frac{X^2\, 10^5}{2T} \quad (\%/1000 \text{ hour})$$

$$= \frac{X^2\, 10^5 \quad (\%/1000 \text{ hr})}{2\ (\#\text{ of dev.})\ (\#\text{ of test hrs.})\ (\text{accl. factor})}$$

$$= \frac{(4.046) \quad 10^5}{2(960)\ (1000)\ (34.6)} = 0.0061\ (\%/1000 \text{ hr})$$

Therefore, FIT = $0.0061 \bullet (10^4) = 61$

Product/Process Changes

As mentioned previously, a design review is performed for product modifications or changes. Once the design is approved, and processes altered (if necessary) for maximum quality, the device goes through qualification testing to check the reliability. If the test results are acceptable, the product is released for mass production.

Testing is also performed when only a process modification or change is made.

The typical qualification/process change tests are listed in Appendix 3.

Failure Analysis

At NEC, failure analysis is performed not only on field failures, but also routinely on products which exhibit defects during the production process. This data is closely checked for correlation with the production process quality information, inspection results, and reliability test data. Information derived from these failure analyses is used to improve product quality.

As there are a lot of failure mechanisms of LSI devices, highly advanced analytical technologies are required to investigate such failures in detail. The standard failure analysis flowchart relating to the returned products from customers is shown in Appendix 4.

NEC's Goals on Failure Rates

The reject rate at customer's incoming inspection, the infant mortality rate, and the long term reliability, are all monitored and checked against NEC's quality and reliability targets (listed in Figure 6).

Figure 6. NEC Quality and Reliability Targets

	Reject Rate at Customer's Incoming Electrical Inspection (PPM)						Long Term Reliability (FIT)						Infant Mortality (RT)					
	Memory		µCOM	Gate Arrays			Memory		µCOM	Gate Arrays			Memory		µCOM	Gate Arrays		
Year	ECL RAM	MOS		BiCMOS	ECL	CMOS	ECL RAM	MOS		BiCMOS	ECL	CMOS	ECL RAM	MOS		BiCMOS	ECL	CMOS
1988	150	50	100	1000	300	300	100	50	100	1000	300	150	100	100	150	1000	300	400
1990	100	50	100	500	200	150	80	50	80	500	250	100	80	100	150	500	250	300

Summary and Conclusion

As has been discussed, building quality and reliability into products is the most efficient way to ensure product success. NEC's approach of distributing quality control functions to process steps, then forming a total quality control system, has produced superior quality and excellent reliability.

Prescreening, whenever necessary, has been a major factor in improving reliability. In addition, monthly reliability assurance tests have ensured high outgoing quality levels.

The combination of building quality into products, effective prescreening of potential failures, and monitoring of reliability through extensive testing, has established a singularly high standard of quality and reliability for NEC's large-scale integrated circuits.

Through a companywide quality control program, continuous research and development activities, extensive failure analysis, and process improvements, this higher standard of quality and reliability will continuously be set and maintained.

Appendix 1
Typical QC Flow for CMOS Fabrication

WAFER FABRICATION PROCESS QC FLOW (CMOS)

FLOW	PROCESS MATERIAL	IN-PROCESS INSPECTION/QUALITY MONITOR
	Silicon Wafer	
	Incoming Inspection	Resistivity (sampling by lot) Dimension (sampling by lot) Visual (sampling by lot)
	Wall Formation	
	Oxidation Photo Lithography	Oxide Thickness (sampling by lot) Alignment and Etching Accuracy (sampling by lot) Layer Resistance (sampling by day)
	Ion Implantation	
	Field Formation	
	Deposition Photo Lithography	Deposit Thickness (sampling by lot) Alignment and Etching Accuracy (sampling by lot) Oxide Thickness (sampling by lot)
	Oxidation	
	Channel Stopper Formation	
	Photo Lithography	Alignment and Etching Accuracy (sampling by lot) Layer Resistance (sampling by lot
	Ion Implantation Oxidation	Oxide Thickness (sampling by lot)
	Gate Formation	
	Deposition Doping Photo Lithography	Deposit Thickness (sampling by lot) Layer Resistance (sampling by lot) Alignment and Etching Accuracy (sampling by lot) Gate Electrode Width (sampling by lot)
	p/n SD Formation	
	Photo Lithography	Alignment and Etching Accuracy (sampling by lot) Layer Resistance (sampling by lot)
	Ion Implantation Anneal	
	Contact Hole	
	Deposition Photo Lithography	Deposit Thickness (sampling by lot) Alignment and Etching Accuracy (sampling by lot)
	Metallization	
	Metal Deposition Photo Lithography	Metal Thickness (sampling by run) Alignment and Etching Accuracy (sampling by lot) Parametric Test (sampling by lot)
	Alloy	
	Passivation	
	Deposition Photo Lithography	Deposit Thickness (sampling by lot) Alignment and Etching Accuracy (sampling by lot)
	Wafer Sort	

Contact Hole and Metallization Steps are Repeated Twice

83vQ-6939B

Appendix 1
Typical QC Flow for PLCC Assembly/Test

Process/Materials	The Check of Manufacturing Conditions				The Check of Manufacturing Qualities			
	Check Items	Frequency	Instrument	Checked By	Check Item	Frequency	Instrument	Checked By
1 Sorted Wafers								
2 Wafer Visual					Wafer Visual	100%	(Naked Eye)	Operator
3 Dicing	Table Speed DI Water Blade Height	Every Shift	Indicators Gauges	P.C.	Sawing Dimensions	Before Running	Microscope with Filter Eyepiece	Operator
4 Break and Expand	Wafer Break Conditions Wafer Expand Conditions	Every Shift	Indicators Gauges	P.C.	Wafer Visual	100%	(Naked Eye)	Operator
5 Die Visual Inspection					Die Visual	Every Lot Sampling (Or 100%)	Microscope	Operator
6 Lead Frames	Die Attached Conditions	Every Shift	Indicators Thermocouple, Potentiometer	P.C.	Die Visual Epoxy Coverage	Every Magazine	(Naked Eye)	Operator
7 Die Attached	Temperature					Every Shift	Microscope	
8 Epoxy Cure (Not Done for Gold Die Attached product)	Heat Temperature N_2 Flow	Every Shift	Indicators Gauges	P.C.	Shear Strength	Every Shift	Dynamometer	Operator
9 Fine Wire	Bonding Conditions	Every Shift	Indicators	P.C.	Visual	Every Magazine	Microscope	Operator
10 Wire Bonding	Temperature	Every Week	Thermocouple and Potentiometer	P.C.	Wire Pull Test	Every Shift	Tension Gauge	Operator
11 Pre-Seal Visual Inspection					Die Visual	Every Lot Sampling (or 100%)	Microscope	Inspector
12 Molding Compound	Temperature of Pellet, Expiration Date	Every Shift	Thermocouple	P.C.				
13 Molding	Temperature Profile of Die Set Preheat Temperatue Pressure Cure Time	Every Shift	Thermocouple, Potentiometer	P.C.	Visual	100%	(Naked Eye)	Operator
14 Mold Aging	Temperature	Every Shift	Indicator	P.C.				
15 Deflashing	Deflashing Conditions	Every Shift	Indicators	P.C.	Visual	Every Lot	(Naked Eye)	Operator
	Concentration	Every Week	Titration	Tech.				
	Density	Every Week	Density Meter	Tech.				
	Water Jet Pressure	Every Day	Gauge	Tech.				
16 Plating	Plating Conditions	Every Day	Indicators	P.C.				
	Concentration	Every Week	Titration	Tech.				

83VQ-69140B

2

Appendix 1
Typical QC Flow for PLCC Assembly/Test (Cont.)

Process/Materials	The Check of Manufacturing Conditions				The Check of Manufacturing Qualities			
	Check Items	Frequency	Instrument	Checked By	Check Item	Frequency	Instrument	Checked By
17 Plating Inspection					Visual	Every Lot	(Naked Eye)	Technician
					Plating Thickness	Every Lot	X-ray	Technician
					Composition	Every Lot	X-ray	Technician
					Solderability	Once/Day	(Naked Eye)	Technician
18 Marking Ink	Marking Conditions	Every Shift	Indicators	P.C.	Visual	Every Lot	(Naked Eye)	Operator
19 Marking								
20 Mark Cure	Temperature	Every Shift	Thermocouple	P.C.	Marking Permanency	Twice/Shift	Automatic Tester	Operator
21 Lead Forming	Dimensions	Every Shift (Before Running)	Test Jig. Caliper	Operator	Visual	Every Lot	(Naked Eye)	Operator
22 Final Assembly Inspection					Visual	Every Lot	Magnifying Lamp	Operator
23 1st Electrical Sorting	P.M. Check	Every Day	P.M. Jig.	Operator				
	Sample Check	Before Testing	Test Samples	Operator	Electrical Characteristics	100%	IC Tester	Operator
24 Burn-In (Whenever Necessary)	Burn-In Conditions	Every Batch	Indicator	P.C.				
25 1st Electrical Sorting		Every Day	P.M. Jig.	Operator				
		Before Testing	Test Samples	Operator	Electrical Characteristics	100%	IC Tester	Operator
26 Reliability Assurance Test		Every Month						
27 In-Warehouse Inspection		Every Day	P.M. Jig.		Electrical Characteristics	Every Lot	IC Tester	Inspector
		Before Testing	Test Samples		Visual (Major)	Every Lot	(Naked Eye) and Microscope	Inspector
					Visual (Minor)	Every Lot	(Naked Eye)	Inspector
28 Warehousing								

83vO-6941B

Appendix 2
Typical Reliability Assurance Tests

The life tests performed by NEC consist of high temperature bias life (HTB), high temperature storage life (HTSL), high temperature/high humidity (T/H), and high humidity storage life (HHSL) tests. Additionally, various environmental and mechanical tests are performed. The table below shows the conditions of the various life tests, environmental tests, and mechanical tests.

Test Item	Symbol	MIL-STD-883C Method	Condition	Remarks
High Temperature Bias Life	HTB	1005	T_A = 125°C, V_{DD} specified per device type.	(Note 1)
High Temperature Storage Life	HTSL	1008	T_A = 150°C.	(Note 1)
High Temperature/ High Humidity	T/H		T_A = 85°C, RH = 85%, V_{DD} = 5.5 V.	(Note 1)
High Humidity Storage Life	HHSL		T_A = 85°C, RH = 85%.	(Note 1)
Pressure Cooker	PCT		T_A = 125°C, P = 2.3 atm.	(Note 1)
Temperature Cycling	T/C	1010	– 65°C to 150°C, 1 hr/cycle.	(Note 1)
Lead Fatigue	C3	2004	90° bends. 3 bends without breaking.	(Note 2)
Solderability	C4	2003	230°C, 5 sec, Rosin Base Flux.	(Note 3)
Soldering Heat/ Temperature Cycle/ Thermal Shock	C6	(Note 4) 1010 1011	260°C, 10 sec, Rosin Base Flux/ 10-1 hr cycles, –65°C to 150°C/ 15-10 min cycles, 0°C to 100°C	(Note 1)

Notes:

(1) Electrical test per data sheet is performed. Devices that exceed the data sheet limits are considered to be rejects.

(2) Broken lead is considered to be a reject.

(3) Less than 95% coverage is considered to be a reject.

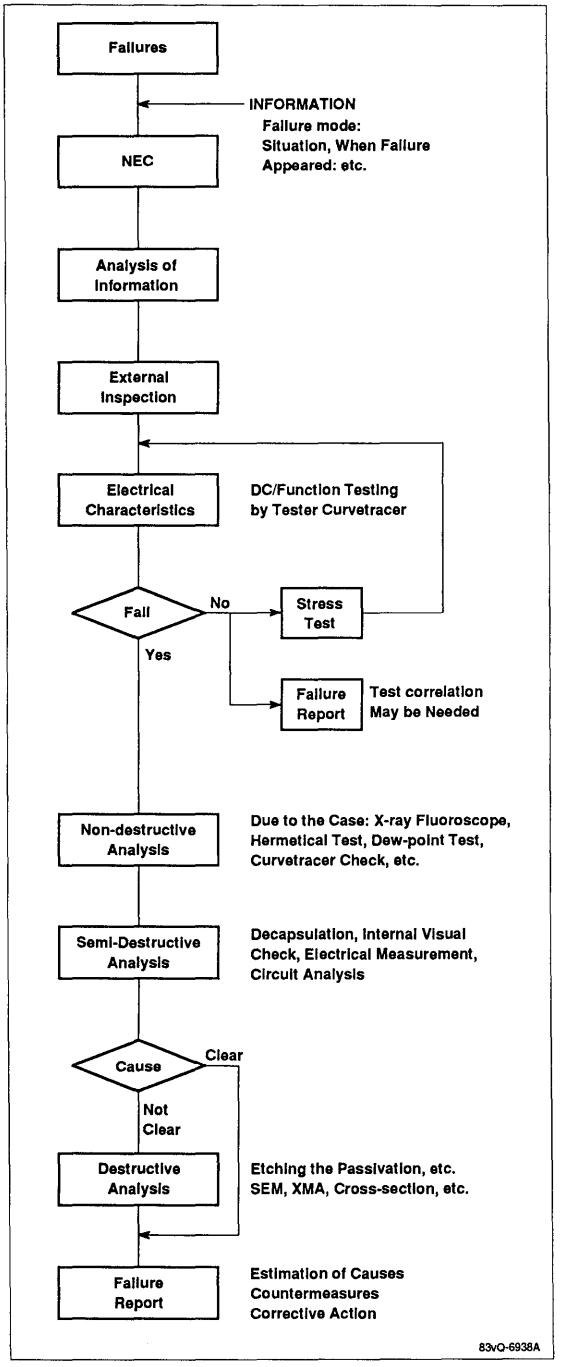
(4) MIL-STD-750A, method 2031.

Appendix 3
New Product / Process Change Tests

Test Item	Test Conditions	Sample Size	Newly Developed Product	Shrink Die	New Package	Wafer	Assembly
High Temp. Operating Life	See Appendix 2, 1000H	20 to 50 pcs X 1 to 3 lots	0	0	0	0	0
High Temp. Storage Life	T = 150°C (Plastic), 175°C (Ceramic), 1000H	10 to 20 pcs X 1 to 3 lots	0	0	0	0	0
High Temp. and Humidity Bias Life (Plastic Device)	See Appendix 2, 1000H	20 to 50 pcs X 1 to 3 lots	0	0	0	0	0
Pressure cooker (Plastic Device)	See Appendix 2, 288H	10 to 20 pcs X 1 to 3 lots	0	0	0	0	0
Thermal Environmental	See Appendix 2	10 to 20 pcs X 1 to 3 lots	0	X	0	X	0
Mechanical Environmental (Ceramic Device)	20G, 10 to 2000 Hz 1500G, 0.5 ms 20000G, 1 min	10 to 20 pcs X 1 to 3 lots	0	X	0	X	0
Lead Fatigue	See Appendix 2	5 pcs X 1 to 3 lots	X	—	X	—	X
Solderability	See Appendix 2	5 pcs X 1 to 3 lots	X	—	X	—	X
ESD	(1) C = 200 pF, R = 0Ω (2) C = 100 pF, R = 1.5 KΩ	20 pcs X 1 to 3 lots	0	0	X	0	X
Long Term T/C	See Appendix 2, 1000 cy	10 to 50 pcs X 1 to 3 lots	0	0	0	0	0

0 – Performed X – Perform if Necessary — – Not Performed

Appendix 4
Failure Analysis Flowchart



NEC

Section 3
16-Bit CPUs

NEC
NEC Electronics Inc.

μPD70108 (V20) 8/16-Bit Microprocessor: High-Performance, CMOS

Description

The μPD70108 (V20®) is a CMOS 16-bit microprocessor with internal 16-bit architecture and an 8-bit external data bus. The μPD70108 instruction set is a superset of the μPD8086/8088; however, mnemonics and execution times are different. The μPD70108 additionally has a powerful instruction set including bit processing, packed BCD operations, and high-speed multiplication/division operations. The μPD70108 can also execute the entire 8080 instruction set and comes with a standby mode that significantly reduces power consumption. It is software-compatible with the μPD70116 16-bit microprocessor.

Features

- ☐ Minimum instruction execution time: 250 ns (at 8 MHz), 200 ns (at 10 MHz)
- ☐ Maximum addressable memory: 1 Mbyte
- ☐ Abundant memory addressing modes
- ☐ 14 x 16-bit register set
- ☐ 101 instructions
- ☐ Instruction set is a superset of μPD8086/8088 instruction set
- ☐ Bit, byte, word, and block operations
- ☐ Bit field operation instructions
- ☐ Packed BCD instructions
- ☐ Multiplication/division instruction execution time: 2.4 μs to 7.1 μs (at 8 MHz), 1.9 μs to 5.7 μs at 10 MHz
- ☐ High-speed block transfer instructions: 1 Mbyte/s (at 8 MHz), 1.25 Mbyte/s (at 10 MHz)
- ☐ High-speed calculation of effective addresses: 2 clock cycles in any addressing mode
- ☐ Maskable (INT) and nonmaskable (NMI) interrupt inputs
- ☐ IEEE-796 bus compatible interface
- ☐ 8080 emulation mode
- ☐ CMOS technology
- ☐ Low power consumption
- ☐ Low-power standby mode
- ☐ Single power supply
- ☐ 8 MHz or 10 MHz clock

V20 is a registered trademark of NEC Corporation.

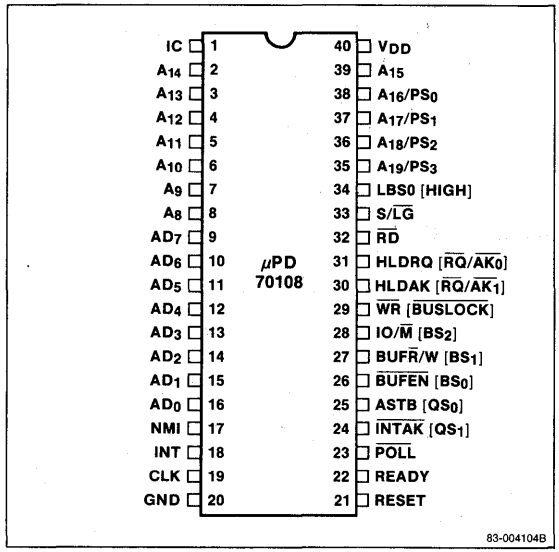
Ordering Information

Part Number	Max Frequency of Operation	Package Type
μPD70108C-8	8 MHz	40-pin plastic DIP
C-10	10 MHz	
L-8	8 MHz	44-pin PLCC
L-10	10 MHz	
GC-8	8 MHz	52-pin plastic QFP (P52GC-100-3B6)
GC-10	10 MHz	

3a

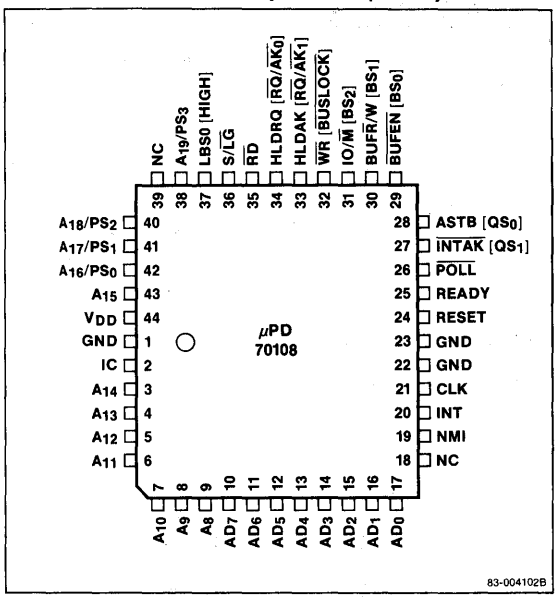
Pin Configurations

40-Pin Plastic DIP

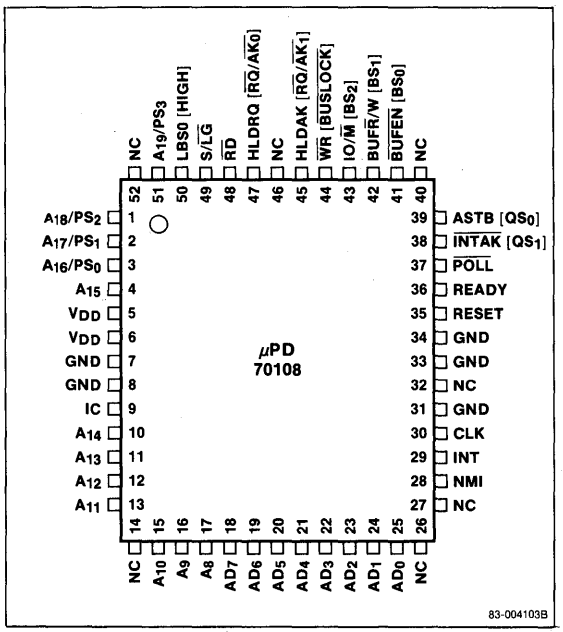


Pin Configurations (cont)

44-Pin Plastic Leaded Chip Carrier (PLCC)



52-Pin Plastic QFP



Pin Identification

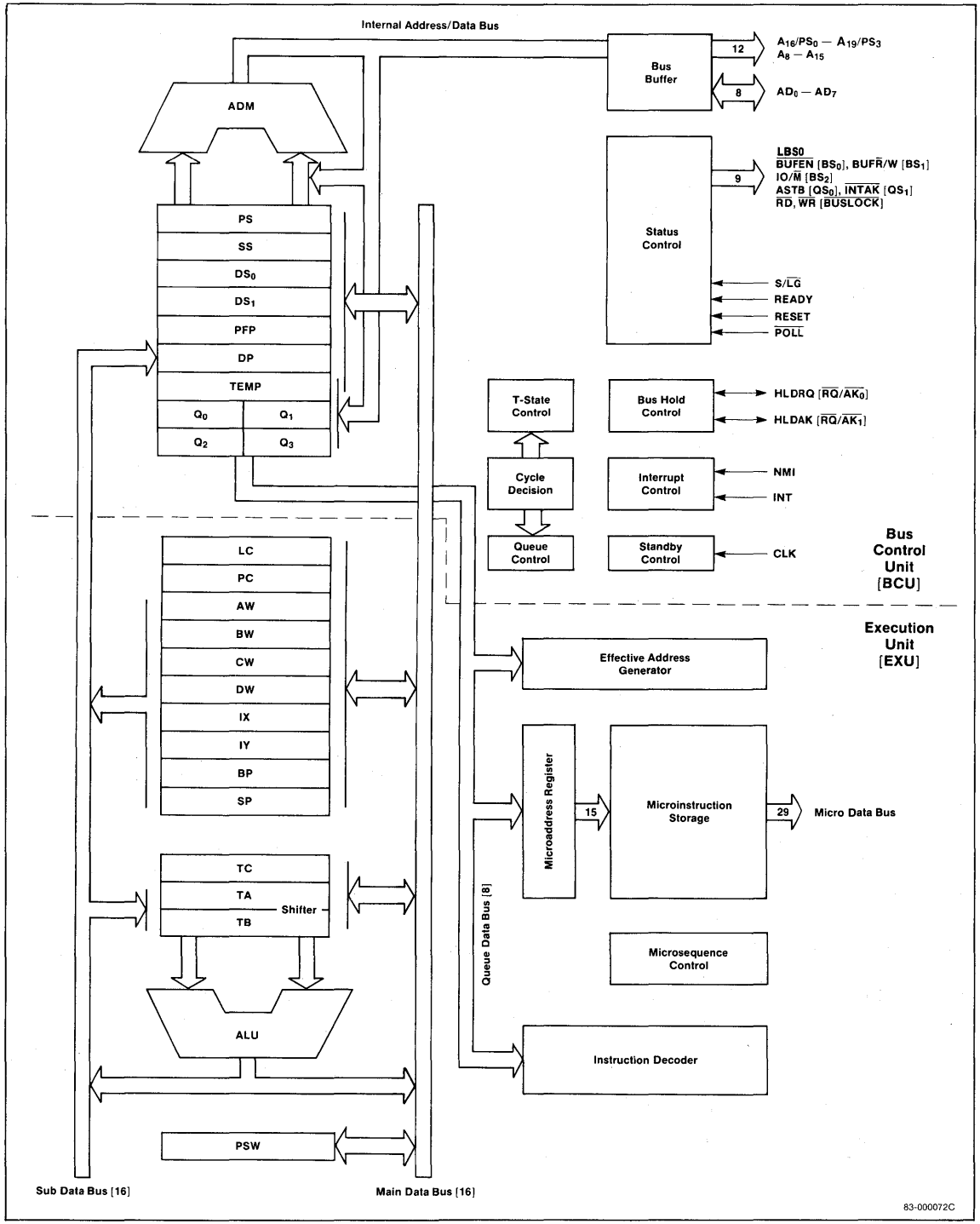
Symbol	Direction	Function
IC*		Internally connected
A_{14} - A_8	Out	Address bus, middle bits
AD_7 - AD_0	In/Out	Address/data bus
NMI	In	Nonmaskable interrupt input
INT	In	Maskable interrupt input
CLK	In	Clock input
GND		Ground potential
RESET	In	Reset input
READY	In	Ready input
$\overline{POLL}$	In	Poll input
$\overline{INTAK}$ (QS_1)	Out	Interrupt acknowledge output (queue status bit 1 output)
ASTB (QS_0)	Out	Address strobe output (queue status bit 0 output)
$\overline{BUFEN}$ (BS_0)	Out	Buffer enable output (bus status bit 0 output)
$BUF\overline{R}/W$ (BS_1)	Out	Buffer read/write output (bus status bit 1 output)
$IO/\overline{M}$ (BS_2)	Out	Access is I/O or memory (bus status bit 2 output)
$\overline{WR}$ ($\overline{BUSLOCK}$)	Out	Write strobe output (bus lock output)
HLDAK ($\overline{RQ}/\overline{AK}_1$)	Out (In/Out)	Hold acknowledge output, (bus hold request input/acknowledge output 1)
HLDRQ ($\overline{RQ}/\overline{AK}_0$)	In (In/Out)	Hold request input (bus hold request input/acknowledge output 0)
$\overline{RD}$	Out	Read strobe output
$S/\overline{LG}$	In	Small-scale/large-scale system input
LBS0 (HIGH)	Out	Latched bus status output 0 (always high in large-scale systems)
A_{19}/PS_3 - A_{16}/PS_0	Out	Address bus, high bits or processor status output
A_{15}	Out	Address bus, bit 15
V_{DD}		Power supply

Notes: * IC should be connected to ground.

Where pins have different functions in small- and large-scale systems, the large-scale system pin symbol and function are in parentheses.

Unused input pins should be tied to ground or V_{DD} to minimize power dissipation and prevent the flow of potentially harmful currents.

Block Diagram



3a

Pin Functions

Some pins of the μPD70108 have different functions according to whether the microprocessor is used in a small- or large-scale system. Other pins function the same way in either type of system.

A_{15} - A_8 [Address Bus]
For small- and large-scale systems.

The CPU uses these pins to output the middle 8 bits of the 20-bit address data. They are three-state outputs and become high impedance during hold acknowledge.

AD_7 - AD_0 [Address/Data Bus]
For small- and large-scale systems.

The CPU uses these pins as the time-multiplexed address and data bus. When high, an AD bit is a one; when low, an AD bit is a zero. This bus contains the lower 8 bits of the 20-bit address during T1 of the bus cycle and is used as an 8-bit data bus during T2, T3, and T4 of the bus cycle.

Sixteen-bit data I/O is performed in two steps. The low byte is sent first, followed by the high byte. The address/data bus is a three-state bus and can be at a high or low level during standby mode. The bus will be high impedance during hold and interrupt acknowledge.

NMI [Nonmaskable Interrupt]
For small- and large-scale systems.

This pin is used to input nonmaskable interrupt requests. NMI cannot be masked by software. This input is positive edge triggered and must be held high for five clocks to guarantee recognition. Actual interrupt processing begins, however, after completion of the instruction in progress.

The contents of interrupt vector 2 determine the starting address for the interrupt-servicing routine. Note that a hold request will be accepted even during NMI acknowledge.

This interrupt will cause the μPD70108 to exit the standby mode.

INT [Maskable Interrupt]
For small- and large-scale systems.

This pin is an interrupt request that can be masked by software.

INT is active high level and is sensed during the last clock of the instruction. The interrupt will be accepted if the interrupt enable flag IE is set. The CPU outputs the INTAK signal to inform external devices that the interrupt request has been granted. INT must be asserted until the interrupt acknowledge is returned.

If NMI and INT interrupts occur at the same time, NMI has higher priority than INT and INT cannot be accepted. A hold request will be accepted during INT acknowledge.

This interrupt causes the μPD70108 to exit the standby mode.

CLK [Clock]
For small- and large-scale systems.

This pin is used for external clock input.

RESET [Reset]
For small- and large-scale systems.

This pin is used for the CPU reset signal. It is an active high level. Input of this signal has priority over all other operations. After the reset signal input returns to a low level, the CPU begins execution of the program starting at address FFFF0H.

In addition to causing normal CPU start, RESET input will cause the μPD70108 to exit the standby mode.

READY [Ready]
For small- and large-scale systems.

When the memory or I/O device being accessed cannot complete data read or write within the CPU basic access time, it can generate a CPU wait state (Tw) by setting this signal to inactive (low level) and requesting a read/write cycle delay.

If the READY signal is active (high level) during either the T3 or Tw state, the CPU will not generate a wait state. READY is not synchronized internally. To guarantee correct operation external logic must ensure that setup and hold times relative to CLK are met.

$\overline{POLL}$ [Poll]
For small- and large-scale systems.

The CPU checks this input upon execution of the POLL instruction. If the input is low, then execution continues. If the input is high, the CPU will check the $\overline{POLL}$ input every five clock cycles until the input becomes low again.

The $\overline{POLL}$ and READY functions are used to synchronize CPU program execution with the operation of external devices.

$\overline{RD}$ [Read Strobe]
For small- and large-scale systems.

The CPU outputs this strobe signal during data read from an I/O device or memory. The IO/$\overline{M}$ signal is used to select between I/O and memory.

The three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

S/$\overline{LG}$ [Small/Large]
For small- and large-scale systems.

This signal determines the operation mode of the CPU. This signal is fixed at either a high or low level. When this signal is a high level, the CPU will operate in small-scale system mode, and when low, in the large-scale system mode. A small-scale system will have at most one bus master such as a DMA controller device on the bus. A large-scale system can have more than one bus master accessing the bus as well as the CPU.

$\overline{INTAK}$ [Interrupt Acknowledge]
For small-scale systems.

The CPU generates the $\overline{INTAK}$ signal low when it accepts an INT signal.

The interrupting device synchronizes with this signal and outputs the interrupt vector to the CPU via the data bus (AD_7 - AD_0). $\overline{INTAK}$ is a held at a high level in the standby mode.

ASTB [Address Strobe]
For small-scale systems.

The CPU outputs this strobe signal to latch address information at an external latch.

ASTB is held at a low level during standby mode and hold acknowledge.

$\overline{BUFEN}$ [Buffer Enable]
For small-scale systems.

This is used as the output enable signal for an external bidirectional buffer. The CPU generates this signal during data transfer operations with external memory or I/O devices or during input of an interrupt vector.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

BUF$\overline{R}$/W [Buffer Read/Write]
For small-scale systems.

The output of this signal determines the direction of data transfer with an external bidirectional buffer. A high output causes transmission from the CPU to the external device; a low signal causes data transfer from the external device to the CPU.

BUF$\overline{R}$/W is a three-state output and becomes high impedance during hold acknowledge.

IO/$\overline{M}$ [IO/Memory]
For small-scale systems.

The CPU generates this signal to specify either I/O access or memory access. A high-level output specifies I/O and a low-level signal specifies memory.

IO/M's output is three state and becomes high impedance during hold acknowledge.

$\overline{WR}$ [Write Strobe]
For small-scale systems.

The CPU generates this strobe signal during data write to an I/O device or memory. Selection of either I/O or memory is performed by the IO/M signal.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

HLDAK [Hold Acknowledge]
For small-scale systems.

The HLDAK signal is used to indicate that the CPU accepts the hold request signal (HLDRQ). When this signal is a high level, the address bus, address/data bus, and the control lines become high impedance.

HLDRQ [Hold Request]
For small-scale systems.

This input signal is used by external devices to request the CPU to release the address bus, address/data bus, and the control bus.

LBS0 [Latched Bus Status 0]
For small-scale systems.

The CPU uses this signal along with the IO/$\overline{M}$ and BUF$\overline{R}$/W signals to inform an external device what the current bus cycle is.

IO/$\overline{M}$	BUFR/$\overline{W}$	LBS0	Bus Cycle
0	0	0	Program fetch
0	0	1	Memory read
0	1	0	Memory write
0	1	1	Passive state
1	0	0	Interrupt acknowledge
1	0	1	I/O read
1	1	0	I/O write
1	1	1	Halt

3a

A_{19}/PS_3 - A_{16}/PS_0 [Address Bus/Processor Status]
For small- and large-scale systems.

These pins are time multiplexed to operate as an address bus and as processor status signals.

When used as the address bus, these pins are the high 4 bits of the 20-bit memory address. During I/O access, all 4 bits output data 0.

The processor status signals are provided for both memory and I/O use. PS_3 is always 0 in the native mode and 1 in 8080 emulation mode. The interrupt enable flag (IE) is on pin PS_2. Pins PS_1 and PS_0 indicate which memory segment is being accessed.

A_{17}/PS_1	A_{16}/PS_0	Segment
0	0	Data segment 1
0	1	Stack segment
1	0	Program segment
1	1	Data segment 0

The output of these pins is three state and becomes high impedance during hold acknowledge.

QS_1, QS_0 [Queue Status]
For large-scale systems.

The CPU uses these signals to allow external devices, such as the floating-point arithmetic processor chip, (μPD72091) to monitor the status of the internal CPU instruction queue.

QS_1	QS_0	Instruction Queue Status
0	0	NOP (queue does not change)
0	1	First byte of instruction
1	0	Flush queue
1	1	Subsequent bytes of instruction

The instruction queue status indicated by these signals is the status when the execution unit (EXU) accesses the instruction queue. The data output from these pins is therefore valid only for one clock cycle immediately following queue access. These status signals are provided so that the floating-point processor chip can monitor the CPU's program execution status and synchronize its operation with the CPU when control is passed to it by the FPO (Floating Point Operation) instructions. These outputs are held at a low level in the standby mode.

BS_2 - BS_0 [Bus Status]
For large-scale systems.

The CPU uses these status signals to allow an external bus controller to monitor what the current bus cycle is. The external bus controller decodes these signals and generates the control signals required to perform access of the memory or I/O device.

BS_2	BS_1	BS_0	Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Program fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state

The output of these signals is three state and becomes high impedance during hold acknowledge. These outputs are held at high level in the standby mode.

$\overline{BUSLOCK}$ [Bus Lock]
For large-scale systems.

The CPU uses this signal to secure the bus while executing the instruction immediately following the $\overline{BUSLOCK}$ prefix instruction, or during an interrupt acknowledge cycle. It is a status signal to the other bus masters in a multiprocessor system, inhibiting them from using the system bus during this time.

The output of this signal is three state and becomes high impedance during hold acknowledge. $\overline{BUSLOCK}$ is high during standby mode except if the HALT instruction has a $\overline{BUSLOCK}$ prefix, then it is held low.

$\overline{RQ}/\overline{AK}_1$, $\overline{RQ}/\overline{AK}_0$ [Hold Request/Acknowledge]
For large-scale systems.

These pins function as bus hold request inputs ($\overline{RQ}$) and as bus hold acknowledge outputs ($\overline{AK}$). $\overline{RQ}/\overline{AK}_0$ has a higher priority than $\overline{RQ}/\overline{AK}_1$.

These pins have three-state outputs with on-chip pull-up resistors which keep the pin at a high level when the output is high impedance.

V_{DD} [Power Supply]
For small- and large-scale systems.

This pin is used for the +5 V power supply.

GND [Ground]
For small- and large-scale systems.

This pin is used for ground.

IC [Internally Connected]
This pin is used for tests performed at the factory by NEC. The μPD70108 is used with this pin at ground potential.

Absolute Maximum Ratings

$T_A = +25°C$

Power supply voltage, V_{DD}	−0.5 V to +7.0 V
Power dissipation, PD_{MAX}	0.5 W
Input voltage, V_I	−0.5 V to V_{DD} + 0.3 V
CLK input voltage, V_K	−0.5 V to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 V to V_{DD} + 0.3 V
Operating temperature at 5 MHz, T_{OPT}	−40°C to +85°C
Storage temperature, T_{STG}	−65°C to +150°C

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Capacitance

$T_A = +25°C$, $V_{DD} = 0$ V

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input capacitance	C_I		15	pF	fc = 1 MHz Unmeasured pins returned to 0 V
I/O capacitance	C_{IO}		15	pF	

3a

DC Characteristics

$T_A = -10°C$ to $+70°C$, $V_{DD} = +5$ V ± 5%

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Input voltage high	V_{IH}	2.2		V_{DD} + 0.3	V	
Input voltage low	V_{IL}	−0.5		0.8	V	
CLK input voltage high	V_{KH}	3.9		V_{DD} + 1.0	V	
CLK input voltage low	V_{KL}	−0.5		0.6	V	
Output voltage high	V_{OH}	0.7 x V_{DD}			V	I_{OH} = −400 μA
Output voltage low	V_{OL}			0.4	V	I_{OL} = 2.5 mA
Input leakage current high	I_{LIH}			10	μA	$V_I = V_{DD}$
Input leakage current low	I_{LIL}			−10	μA	V_I = 0 V
Output leakage current high	I_{LOH}			10	μA	$V_O = V_{DD}$
Output leakage current low	I_{LOL}			−10	μA	V_O = 0 V
Supply current	I_{DD}	70108-8	45	80	mA	Normal operation
		8 MHz	6	12	mA	Standby mode
		70108-10	60	100	mA	Normal operation
		10 MHz	7	14	mA	Standby mode

AC Characteristics

$T_A = -10°C$ to $+70°C$, $V_{DD} = +5\ V \pm 5\%$

Parameter	Symbol	μPD70108-8 Min	μPD70108-8 Max	μPD70108-10 Min	μPD70108-10 Max	Unit	Conditions
Small/Large Scale							
Clock cycle	t_{CYK}	125	500	100	500	ns	
Clock pulse width high	t_{KKH}	44		41		ns	$V_{KH} = 3.0\ V$
Clock pulse width low	t_{KKL}	60		49		ns	$V_{KL} = 1.5\ V$
Clock rise time	t_{KR}		10		5	ns	1.5 V to 3.0 V
Clock fall time	t_{KF}		10		5	ns	3.0 V to 1.5 V
READY inactive setup to CLK↓	t_{SRYLK}	−8		−10		ns	
READY inactive hold after CLK↑	t_{HKRYH}	20		20		ns	
READY active setup to CLK↑	t_{SRYHK}	$t_{KKL} - 8$		$t_{KKL} - 10$		ns	
READY active hold after CLK↑	t_{HKRYL}	20		20		ns	
Data setup time to CLK ↓	t_{SDK}	20		10		ns	
Data hold time after CLK ↓	t_{HKD}	10		10		ns	
NMI, INT, $\overline{POLL}$ setup time to CLK ↑	t_{SIK}	15		15		ns	
Input rise time (except CLK)	t_{IR}		20		20	ns	0.8 V to 2.2 V
Input fall time (except CLK)	t_{IF}		12		12	ns	2.2 V to 0.8 V
Output rise time	t_{OR}		20		20	ns	0.8 V to 2.2 V
Output fall time	t_{OF}		12		12	ns	2.2 V to 0.8 V
Small Scale							
Address delay time from CLK ↓	t_{DKA}	10	60	10	48	ns	
Address hold time from CLK ↓	t_{HKA}	10		10		ns	
PS delay time from CLK ↓	t_{DKP}	10	60	10	50	ns	
PS float delay time from CLK ↑	t_{FKP}	10	60	10	50	ns	
Address setup time to ASTB ↓	t_{SAST}	$t_{KKL} - 30$		$t_{KKL} - 30$		ns	
Address float delay time from CLK ↓	t_{FKA}	t_{HKA}	60	t_{HKA}	50	ns	$C_L = 100\ pF$
ASTB ↑ delay time from CLK ↓	t_{DKSTH}		50		40	ns	
ASTB ↓ delay time from CLK ↑	t_{DKSTL}		55		45	ns	
ASTB width high	t_{STST}	$t_{KKL} - 10$		$t_{KKL} - 10$		ns	
Address hold time from ASTB ↓	t_{HSTA}	$t_{KKH} - 10$		$t_{KKH} - 10$		ns	

AC Characteristics (cont)

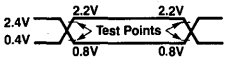
$T_A = -10°C$ to $+70°C$, $V_{DD} = +5 V \pm 5\%$

Parameter	Symbol	μPD70108-8 Min	μPD70108-8 Max	μPD70108-10 Min	μPD70108-10 Max	Unit	Conditions
Small Scale (cont)							
Control delay time from CLK	t_{DKCT}	10	65	10	55	ns	
Address float to RD↓	t_{AFRL}	0		0		ns	
$\overline{RD}$ ↓ delay time from CLK ↓	t_{DKRL}	10	80	10	70	ns	
$\overline{RD}$ ↑ delay time from CLK ↓	t_{DKRH}	10	80	10	60	ns	
Address delay time from RD ↑	t_{DRHA}	$t_{CYK} - 40$		$t_{CYK} - 35$		ns	
$\overline{RD}$ width low	t_{RR}	$2t_{CYK}-50$		$2t_{CYK}-40$		ns	$C_L = 100$ pF
Data output delay time from CLK ↓	t_{DKD}	10	60	10	50	ns	
Data float delay time from CLK ↓	t_{FKD}	10	60	10	50	ns	
$\overline{WR}$ width low	t_{WW}	$2t_{CYK}-40$		$2t_{CYK}-35$		ns	
HLDRQ setup time to CLK ↑	t_{SHQK}	20		20		ns	
HLDAK delay time from CLK ↓	t_{DKHA}	10	100	10	60	ns	
Large Scale							
Address delay time from CLK ↓	t_{DKA}	10	60	10	48	ns	
Address hold time from CLK ↓	t_{HKA}	10		10		ns	
PS delay time from CLK ↓	t_{DKP}	10	60	10	50	ns	
PS float delay time from CLK ↑	t_{FKP}	10	60	10	50	ns	
Address float delay time from CLK ↓	t_{FKA}	t_{HKA}	60	t_{HKA}	50	ns	
Address delay time from $\overline{RD}$ ↑	t_{DRHA}	$t_{CYK} - 40$		$t_{CYK} - 35$		ns	
ASTB delay time from BS ↓	t_{DBST}		15		15	ns	
BS ↓ delay time from CLK ↑	t_{DKBL}	10	60	10	50	ns	
BS ↑ delay time from CLK ↓	t_{DKBH}	10	65	10	50	ns	
$\overline{RD}$ ↓ delay time from address float	t_{DAFRL}	0		0		ns	$C_L = 100$ pF
$\overline{RD}$ ↓ delay time from CLK ↓	t_{DKRL}	10	80	10	70	ns	
$\overline{RD}$ ↑ delay time from CLK ↓	t_{DKRH}	10	80	10	60	ns	
$\overline{RD}$ width low	t_{RR}	$2t_{CYK}-50$		$2t_{CYK}-40$		ns	
Date output delay time from CLK ↓	t_{DKD}	10	60	10	50	ns	
Data float delay time from CLK ↓	t_{FKD}	10	60	10	50	ns	
$\overline{AK}$ delay time from CLK ↓	t_{DKAK}		50		40	ns	
$\overline{RQ}$ setup time to CLK ↑	t_{SRQK}	10		9		ns	
$\overline{RQ}$ hold time from CLK ↑	t_{HKRQ}	0		0		ns	
$\overline{RQ}$ hold time from CLK ↑	t_{HKRQ2}	30		20		ns	

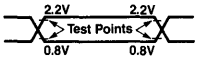
3a

Timing Waveforms

AC Test Input Waveform [Except CLK]

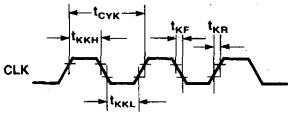


AC Output Test Points



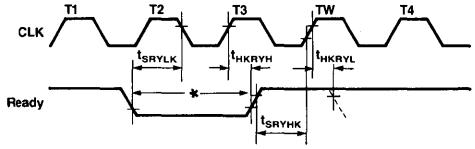
49-000238A

Clock Timing



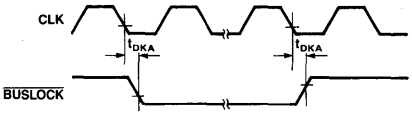
49-000239A

Wait [Ready] Timing

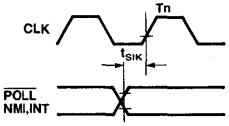


* READY input level must not be changed during this interval.

$\overline{\text{BUSLOCK}}$ Output Timing



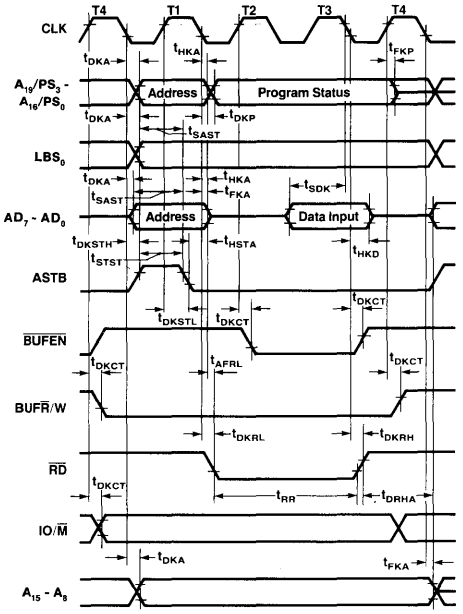
$\overline{\text{POLL}}$, NMI, INT Input Timing



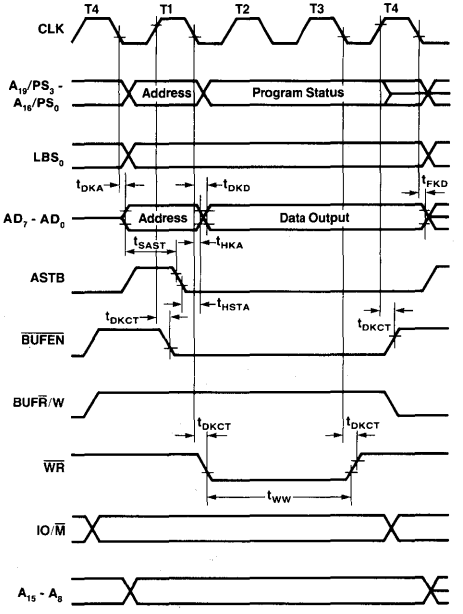
49-000240A

Timing Waveforms (cont)

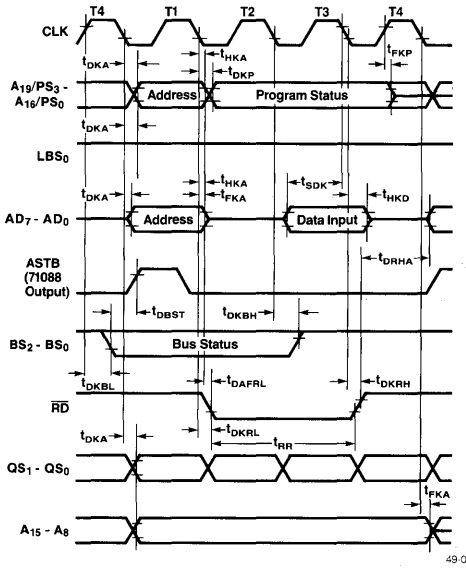
Read Timing [Small Scale]



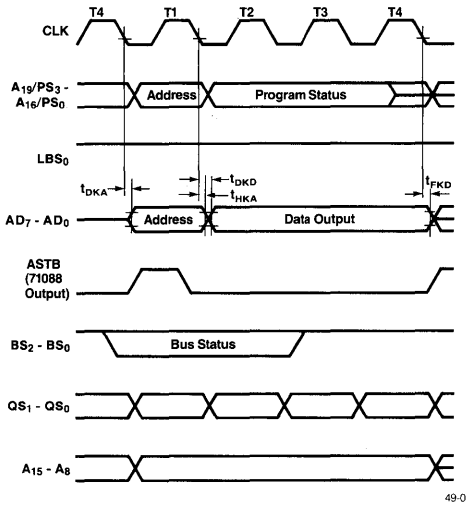
Write Timing [Small Scale]



Read Timing [Large Scale]



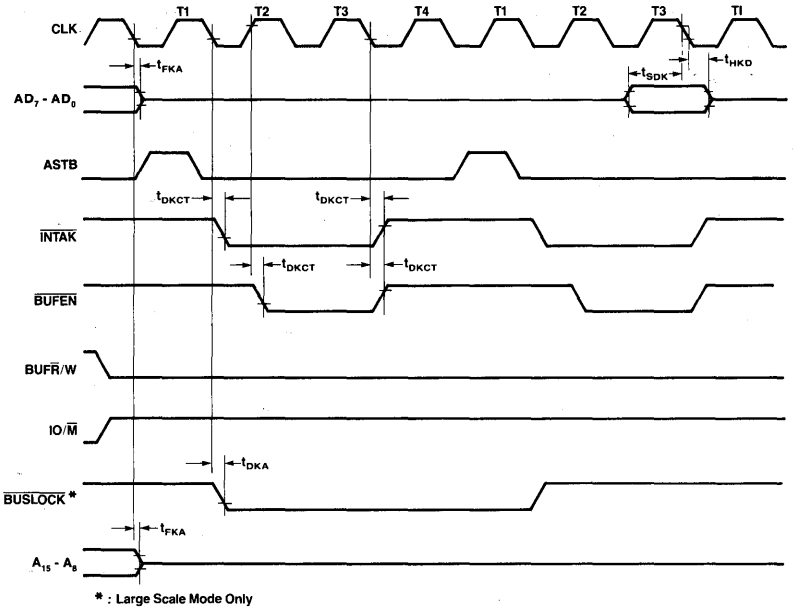
Write Timing [Large Scale]



3a

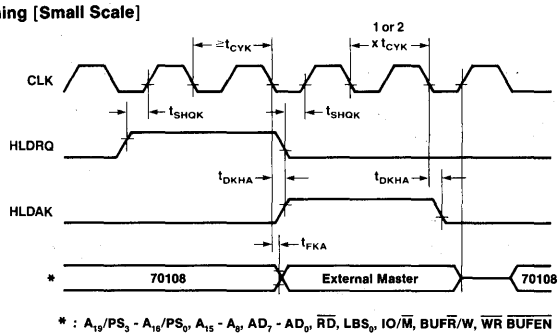
Timing Waveforms (cont)

Interrupt Acknowledge Timing



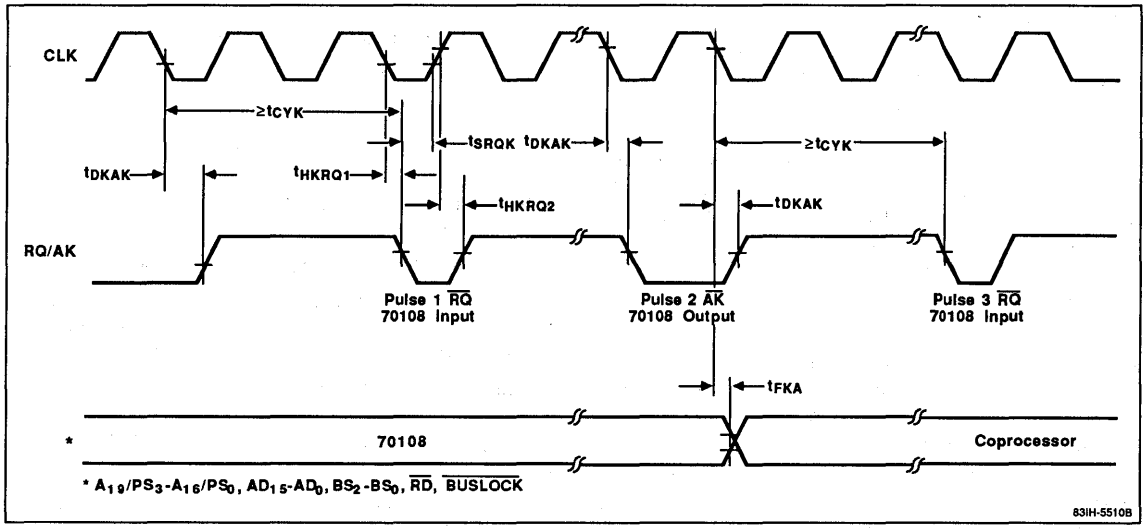
49-000245B

Hold Request/Acknowledge Timing [Small Scale]



49-000250B

Timing Waveforms (cont)



3a

Register Configuration

Program Counter [PC]

The program counter is a 16-bit binary counter that contains the segment offset address of the next instruction which the EXU is to execute.

The PC increments each time the microprogram fetches an instruction from the instruction queue. A new location value is loaded into the PC each time a branch, call, return, or break instruction is executed. At this time, the contents of the PC are the same as the Prefetch Pointer (PFP).

Prefetch Pointer [PFP]

The prefetch pointer (PFP) is a 16-bit binary counter which contains a segment offset which is used to calculate a program memory address that the bus control unit (BCU) uses to prefetch the next byte for the instruction queue. The contents of PFP are an offset from the PS (Program Segment) register.

The PFP is incremented each time the BCU prefetches an instruction from the program memory. A new location will be loaded into the PFP whenever a branch, call, return, or break instruction is executed. At that time the contents of the PFP will be the same as those of the PC (Program Counter).

Segment Registers [PS, SS, DS_0, and DS_1]

The memory addresses accessed by the μPD70108 are divided into 64K-byte logical segments. The starting (base) address of each segment is specified by a 16-bit segment register, and the offset from this starting address is specified by the contents of another register or by the effective address.

These are the four types of segment registers used.

Segment Register	Default Offset
PS (Program Segment)	PFP
SS (Stack Segment)	SP, effective address
DS_0 (Data Segment 0)	IX, effective address
DS_1 (Data Segment 1)	IY

General-Purpose Registers [AW, BW, CW, and DW]

There are four 16-bit general-purpose registers. Each one can be used as one 16-bit register or as two 8-bit registers by dividing them into their high and low bytes (AH, AL, BH, BL, CH, CL, DH, DL).

Each register is also used as a default register for processing specific instructions. The default assignments are:

AW: Word multiplication/division, word I/O, data conversion, translation, BCD rotation.

AL: Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation

AH: Byte multiplication/division

BW: Translation

CW: Loop control branch, repeat prefix

CL: Shift instructions, rototation instructions, BCD operations

DW: Word multiplication/division, indirect addressing I/O

Pointers [SP, BP] and Index Registers [IX, IY]

These registers serve as base pointers or index registers when accessing the memory using based addressing, indexed addressing, or based indexed addressing.

These registers can also be used for data transfer and arithmetic and logical operations in the same manner as the general-purpose registers. They cannot be used as 8-bit registers.

Also, each of these registers acts as a default register for specific operations. The default assignments are:

SP: Stack operations

IX: Block transfer (source), BCD string operations

IY: Block transfer (destination), BCD string operations

Program Status Word [PSW]

The program status word consists of the following six status and four control flags.

Status Flags

- V (Overflow)
- S (Sign)
- Z (Zero)
- AC (Auxiliary Carry)
- P (Parity)
- CY (Carry)

Control Flags

- MD (Mode)
- DIR (Direction)
- IE (Interrupt Enable)
- BRK (Break)

When the PSW is pushed on the stack, the word images of the various flags are as shown here.

PSW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MD	1	1	1	V	DIR	IE	BRK	S	Z	0	AC	0	P	1	CY

The status flags are set and reset depending upon the result of each type of instruction executed.

Instructions are provided to set, reset, and complement the CY flag directly.

Other instructions set and reset the control flags and control the operation of the CPU.

The MD flag can be set/reset only by the BRKEM, RETEM, CALLN, and RETI instructions.

High-Speed Execution of Instructions

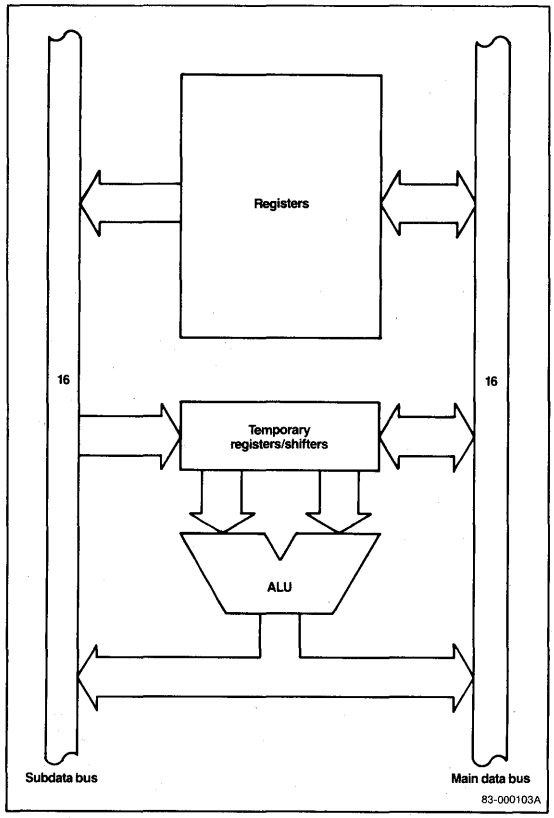
This section highlights the major architectural features that enhance the performance of the μPD70108.

- Dual data bus in EXU
- Effective address generator
- 16/32-bit temporary registers/shifters (TA, TB)
- 16-bit loop counter
- PC and PFP

Dual Data Bus Method

To reduce the number of processing steps for instruction execution, the dual data bus method has been adopted for the μPD70108 (figure 1). The two data buses (the main data bus and the subdata bus) are both 16 bits wide. For addition/subtraction and logical and comparison operations, processing time has been reduced by some 30% over single-bus systems.

Figure 1. Dual Data Buses



Example

ADD AW, BW ; AW ← AW + BW

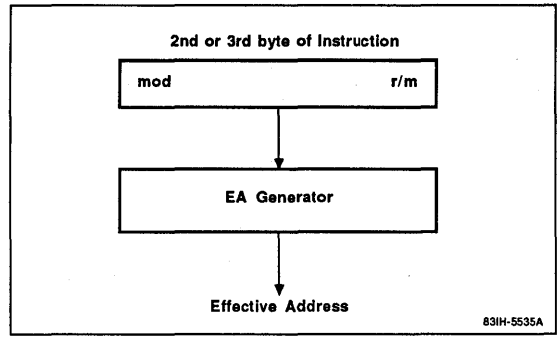
Single Bus	Dual Bus
Step 1 TA ← AW	TA ← AW, TB ← BW
Step 2 TB ← BW	AW ← TA + TB
Step 3 AW ← TA + TB	

Effective Address Generator

The Effective Address Generator (EAG) (figure 2) is a dedicated block of high-speed logic that computes effective addresses in two clock cycles. If an instruction uses memory, EAG decodes the second and/or third instruction bytes to determine the addressing mode, initiates any bus cycles needed to fetch data required to compute the effective address, and stores the computed effective address in the Data Pointer (DP) register.

Calculating an effective address by the microprogramming method normally requires 5 to 12 clock cycles. This circuit requires only two clock cycles for addresses to be generated for any addressing mode. Thus, processing is several times faster.

Figure 2. Effective Address Generator



16/32-Bit Temporary Registers/Shifters [TA, TB]

These 16-bit temporary registers/shifters (TA, TB) are provided for multiplication/division and shift/rotation instructions.

These circuits have decreased the execution time of multiplication/division instructions. In fact, these instructions can be executed about four times faster than with the microprogramming method.

TA + TB: 32-bit temporary register/shifter for multiplication and division instructions.

TB: 16-bit temporary register/shifter for shift/rotation instructions.

3a

Loop Counter [LC]

This counter is used to count the number of loops for a primitive block transfer instruction controlled by a repeat prefix instruction and the number of shifts that will be performed for a multiple bit shift/rotation instruction.

The processing performed for a multiple bit rotation of a register is shown below. The average speed is approximately doubled over the microprogram method.

Example

RORC AW, CL ; CL = 5

Microprogram method	**LC method**
8 + (4 x 5) = 28 clocks	7 + 5 = 12 clocks

Program Counter and Prefetch Pointer [PC and PFP]

The μPD70108 microprocessor has a program counter, (PC) which addresses the program memory location of the instruction to be executed next, and a prefetch pointer(PFP), which addresses the program memory location to be accessed next. Both functions are provided in hardware. A time saving of several clocks is realized for branch, call, return, and break instruction execution, compared with microprocessors that have only one instruction pointer.

Enhanced Instructions

In addition to the μPD8088/86 instructions, the μPD70108 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP R	Pops 8 general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Enhanced Stack Operation Instructions

PUSH imm

This instruction allows immediate data to be pushed onto the stack.

PUSH R/POP R

These instructions allow the contents of the eight general registers to be pushed onto or popped from the stack with a single instruction.

Enhanced Multiplication Instructions

MUL reg16, imm16/MUL mem16, imm16

These instructions allow the contents of a register or memory location to be 16-bit multiplied by immediate data.

Enhanced Shift and Rotate Instructions

SHL reg, imm8/SHR reg, imm8/SHRA reg, imm8

These instructions allow the contents of a register to be shifted by the number of bits defined by the immediate data.

ROL reg, imm8/ROR reg, imm8/ROLC reg, imm8/ RORC reg, imm8

These instructions allow the contents of a register to be rotated by the number of bits defined by the immediate data.

Check Array Boundary Instruction

CHKIND reg16, mem32

This instruction is used to verify that index values pointing to the elements of an array data structure are within the defined range. The lower limit of the array should be in memory location mem32, the upper limit in mem32 + 2. If the index value in reg16 is not between these limits when CHKIND is executed, a BRK 5 will occur. This causes a jump to the location in interrupt vector 5.

Block I/O Instructions

OUTM DW, src-block/INM dst-block, DW

These instructions are used to output or input a string to or from memory, when preceded by a repeat prefix.

Stack Frame Instructions

PREPARE imm16, imm8

This instruction is used to generate the stack frames required by block-structured languages, such as PASCAL and Ada. The stack frame consists of two areas. One area has a pointer that points to another frame which has variables that the current frame can access. The other is a local variable area for the current procedure.

DISPOSE

This instruction releases the last stack frame generated by the PREPARE instruction. It returns the stack and base pointers to the values they had before the PREPARE instruction was used to call a procedure.

Unique Instructions

In addition to the μPD8088/86 instructions and the enhanced instructions, the μPD70108 has the following unique instructions.

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	Adds packed decimal strings
SUB4S	Subtracts one packed decimal string from another
CMP4S	Compares two packed decimal strings
ROL4	Rotates one BCD digit left through AL lower 4 bits
ROR4	Rotates one BCD digit right through AL lower 4 bits
TEST1	Tests a specified bit and sets/resets Z flag
NOT1	Inverts a specified bit
CLR1	Clears a specified bit
SET1	Sets a specified bit
REPC	Repeats next instruction until CY flag is cleared
REPNC	Repeats next instruction until CY flag is set
FPO2	Additional floating point processor call

Variable Length Bit Field Operation Instructions

This category has two instructions: INS (Insert Bit Field) and EXT (Extract Bit Field). These instructions are highly effective for computer graphics and high-level languages. They can, for example, be used for data structures such as packed arrays and record type data used in PASCAL.

INS reg8, reg8/INS reg8, imm4

This instruction (figure 3) transfers low bits from the 16-bit AW register (the number of bits is specified by the second operand) to the memory location specified by the segment base (DS_1 register) plus the byte offset (IY register). The starting bit position within this byte is specified as an offset by the lower 4-bits of the first operand.

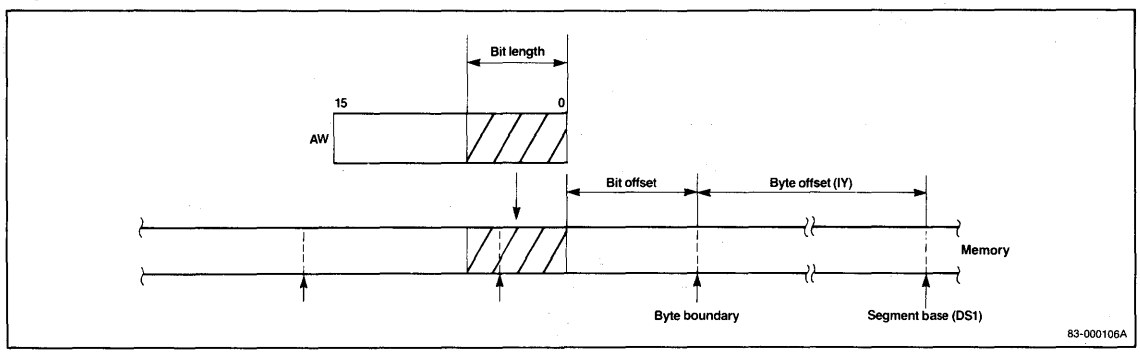
After each complete data transfer, the IY register and the register specified by the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may specify the number of bits transferred (second operand). Because the maximum transferable bit length is 16-bits, only the lower 4-bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

3a

Figure 3. Bit Field Insertion



EXT reg8, reg8/EXT reg8, imm4

This instruction (figure 4) loads to the AW register the bit field data whose bit length is specified by the second operand of the instruction from the memory location that is specified by the DS0 segment register (segment base), the IX index register (byte offset), and the lower 4-bits of the first operand (bit offset).

After the transfer is complete, the IX register and the lower 4-bits of the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may be specified for the second operand. Because the maximum transferrable bit length is 16 bits, however, only the lower 4-bits of the specified register (0H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Packed BCD Operation Instructions

The instructions described here process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte-format operands (ROR4, ROL4). Packed BCD strings may be from 1 to 254 digits in length.

When the number of digits is even, the zero and carry flags will be set according to the result of the operation. When the number of digits is odd, the zero and carry flags may not be set correctly in this case, (CL = odd), the zero flag will not be set unless the upper 4 bits of the highest byte are all zero. The carry flag will not be set unless there is a carry out of the upper 4 bits of the highest byte. When CL is odd, the contents of the upper 4 bits of the highest byte of the result are undefined.

ADD4S

This instruction adds the packed BCD string addressed by the IX index register to the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the overflow flag (V), the carry flag (CY), and zero flag (Z).

BCD string (IY, CL) ← BCD string (IY, CL) + BCD string (IX, CL)

SUB4S

This instruction subtracts the packed BCD string addressed by the IX index register from the packed BCD string addressed by the IY register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the overflow flag (V), the carry flag (CY), and zero flag (Z).

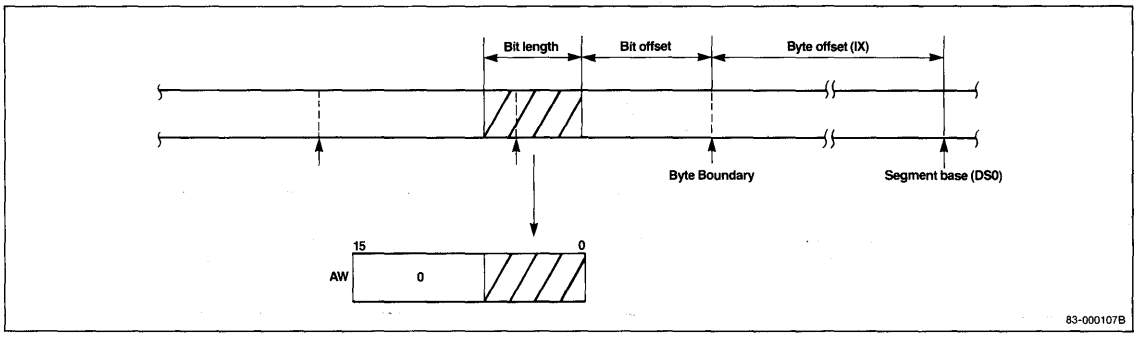
BCD string (IY, CL) ← BCD string (IY, CL) − BCD String (IX, CL)

CMP4S

This instruction performs the same operation as SUB4S except that the result is not stored and only the overflow (V), carry flags (CY) and zero flag (Z) are affected.

BCD string (IY, CL) − BCD string (IX, CL)

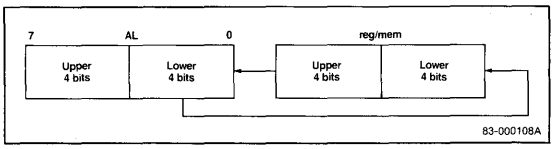
Figure 4. Bit Field Extraction



ROL4

This instruction (figure 5) treats the byte data of the register or memory directly specified by the instruction byte as BCD data and uses the lower 4-bits of the AL register (AL_L) to rotate that data one BCD digit to the left.

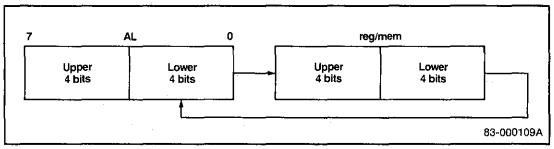
Figure 5. BCD Rotate Left (ROL4)



ROR4

This instruction (figure 6) treats the byte data of the register or memory directly specified by the instruction byte as BCD data and uses the lower 4-bits of the AL register (AL_L) to rotate that data one BCD digit to the right.

Figure 6. BCD Rotate Right (ROR4)



Bit Manipulation Instructions

TEST1

This instruction tests a specific bit in a register or memory location. If the bit is 1, the Z flag is reset to 0. If the bit is 0, the Z flag is set to 1.

NOT1

This instruction inverts a specific bit in a register or memory location.

CLR1

This instruction clears a specific bit in a register or memory location.

SET1

This instruction sets a specific bit in a register or memory location.

Repeat Prefix Instructions

REPC

This instruction causes the μPD70108 to repeat the following primitive block transfer instruction until the CY flag becomes cleared or the CW register becomes zero.

REPNC

This instruction causes the μPD70108 to repeat the following primitive block transfer instruction until the CY flag becomes set or the CW register is decremented to zero.

Floating Point Instruction

FPO2

This instruction is in addition to the μPD8088/86 floating point instruction, FPO1. These instructions are covered in a later section.

Mode Operation Instructions

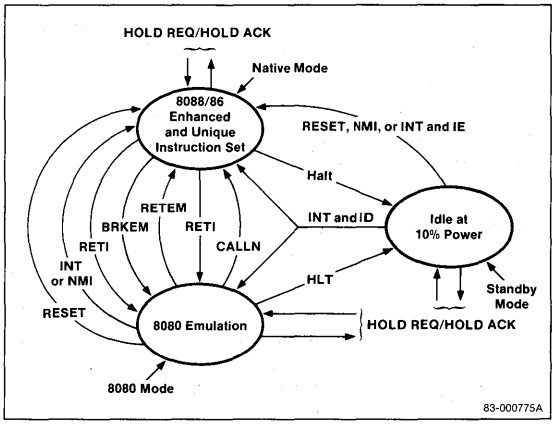
The μPD70108 has two operating modes (figure 7). One is the native mode which executes μPD8088/86, enhanced and unique instructions. The other is the 8080 emulation mode in which the instruction set of the μPD8080AF is emulated. A mode flag (MD) is provided to select between these two modes. Native mode is selected when MD is 1 and emulation mode when MD is 0. MD is set and reset, directly and indirectly, by executing the mode manipulation instructions.

Two instructions are provided to switch operation from the native mode to the emulation mode and back: BRKEM (Break for Emulation), and RETEM (Return from Emulation).

Two instructions are used to switch from the emulation mode to the native mode and back: CALLN (Call Native Routine), and RETI (Return from Interrupt).

The system will return from the 8080 emulation mode to the native mode when the RESET signal is present, or when an external interrupt (NMI or INT) is present.

Figure 7. V20 Modes



BRKEM imm8

This is the basic instruction used to start the 8080 emulation mode. This instruction operates exactly the same as the BRK instruction, except that BRKEM resets the mode flag (MD) to 0. PSW, PS, and PC are saved to the stack. MD is then reset and the interrupt vector specified by the operand imm8 of this command is loaded into PS and PC.

The instruction codes of the interrupt processing routine jumped to are then fetched. Then the CPU executes these codes as μPD8080AF instructions.

In 8080 emulation mode, registers and flags of the μPD8080AF are performed by the following registers and flags of the μPD70108.

	μPD8080AF	μPD70108
Registers:	A	AL
	B	CH
	C	CL
	D	DH
	E	DL
	H	BH
	L	BL
	SP	BP
	PC	PC
Flags:	C	CY
	Z	Z
	S	S
	P	P
	AC	AC

In the native mode, SP is used for the stack pointer. In the 8080 emulation mode this function is performed by BP.

This use of independent stack pointers allows independent stack areas to be secured for each mode and keeps the stack of one of the modes from being destroyed by an erroneous stack operation in the other mode.

The SP, IX, IY and AH registers and the four segment registers (PS, SS, DS_0, and DS_1) used in the native mode are not affected by operations in 8080 emulation mode.

In the 8080 emulation mode, the segment register for instructions is determined by the PS register (set automatically by the interrupt vector) and the segment register for data is the DS_0 register (set by the programmer immediately before the 8080 emulation mode is entered).

It is prohibited to nest BRKEM instructions.

RETEM [no operand]

When RETEM is executed in 8080 emulation mode (interpreted by the CPU as a μPD8080AF instruction), the CPU restores PS, PC, and PSW (as it would when returning from an interrupt processing routine), and returns to the native mode. At the same time, the contents of the mode flag (MD) which was saved to the stack by the BRKEM instruction, is restored to MD = 1. The CPU is set to the native mode.

CALLN imm8

This instruction makes it possible to call the native mode subroutines from the 8080 emulation mode. To return from subroutine to the emulation mode, the RETI instruction is used.

The processing performed when this instruction is executed in the 8080 emulation mode (it is interpreted by the CPU as μPD8080AF instruction), is similar to that performed when a BRK instruction is executed in the native mode. The imm8 operand specifies an interrupt vector type. The contents of PS, PC, and PSW are pushed on the stack and an MD flag value of 0 is saved. The mode flag is set to 1 and the interrupt vector specified by the operand is loaded into PS and PC.

RETI [no operand]

This is a general-purpose instruction used to return from interrupt routines entered by the BRK instruction or by an external interrupt in the native mode. When this instruction is executed at the end of a subroutine entered by the execution of the CALLN instruction, the operation that restores PS, PC, and PSW is exactly the same as the native mode execution. When PSW is restored, however, the 8080 emulation mode value of the mode flag (MD) is restored, the CPU is set in emulation mode, and all subsequent instructions are interpreted and executed as μPD8080AF instructions.

RETI is also used to return from an interrupt procedure initiated by an NMI or INT interrupt in the emulation mode.

Floating Point Operation Chip Instructions

FPO1 fp-op, mem/FPO2 fp-op, mem

These instructions are used for the external floating point processor. The floating point operation is passed to the floating point processor when the CPU fetches one of these instructions. From this point the CPU performs only the necessary auxiliary processing (effective address calculation, generation of physical addresses, and start-up of the memory read cycle).

The floating point processor always monitors the instructions fetched by the CPU. When it interprets one as an instruction to itself, it performs the appropriate processing. At this time, the floating point processor chip uses either the address alone or both the address and read data of the memory read cycle executed by the CPU. This difference in the data used depends on which of these instructions is executed.

Note: During the memory read cycle initiated by the CPU for FPO1 or FPO2 execution, the CPU does not accept any read data on the data bus from memory. Although the CPU generates the memory address, the data is used by the floating point processor.

Interrupt Operation

The interrupts used in the μPD70108 can be divided into two types: interrupts generated by external interrupt requests and interrupts generated by software processing. These are the classifications.

External Interrupts

(a) NMI input (nonmaskable)
(b) INT input (maskable)

Software Processing

As the result of instruction execution

— When a divide error occurs during execution of the DIV or DIVU instruction
— When a memory-boundary-over error is detected by the CHKIND instruction

Conditional break instruction

— When V = 1 during execution of the BRKV instruction

Unconditional break instructions

— 1-byte break instruction: BRK3
— 2-byte break instruction: BRK imm8

Flag processing (Single-step)

— When stack operations are used to set the BRK flag

8080 Emulation mode instructions

— BRKEM imm8
— CALLN imm8

Interrupt Vectors

Starting addresses for interrupt processing routines are either determined automatically by a single location of the interrupt vector table or selected each time interrupt processing is entered.

The interrupt vector table is shown in figure 8. The table uses 1K bytes of memory addresses 000H to 3FFH and can store starting address data for a maximum of 256 vectors (4 bytes per vector).

The corresponding interrupt sources for vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. These vectors consequently cannot be used for general applications.

The BRKEM instruction and CALLN instruction (in the emulation mode) and the INT input are available for general applications for vectors 32 to 255.

A single interrupt vector is made up of 4 bytes (figure 9). The 2 bytes in the low addresses of memory are loaded into PC as the offset, and the high 2 bytes are loaded into PS as the base address. The bytes are combined in reverse order. The lower-order bytes in the vector become the most significant bytes in the PC and PS, and the higher-order bytes become the least significant bytes.

3a

Figure 8. Interrupt Vector Table

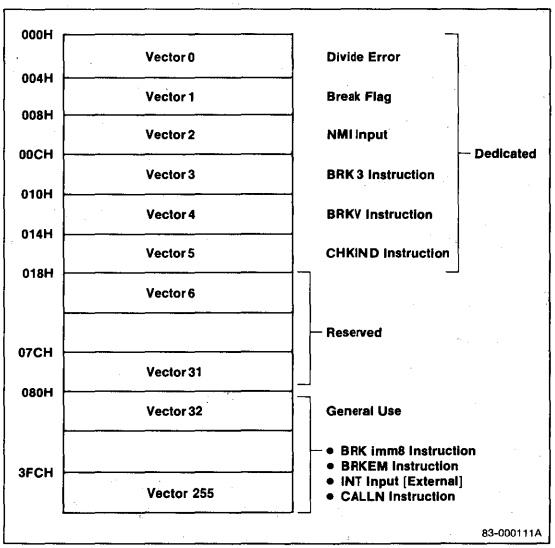
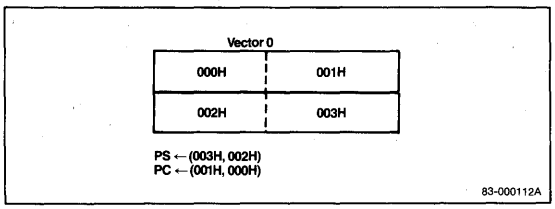


Figure 9. Interrupt Vector 0



Based on this format, the contents of each vector should be initialized at the beginning of the program.

The basic steps to jump to an interrupt processing routine are now shown.

(SP - 1, SP - 2) ← PSW
(SP - 3, SP - 4) ← PS
(SP - 5, SP - 6) ← PC
SP ← SP - 6
IE ← 0, BRK ← 0, MD ← 0
PS ← vector high bytes
PC ← vector low bytes

Standby Function

The μPD70116 has a standby mode to reduce power consumption during program wait states. This mode is set by the HALT instruction in both the native and the emulation mode.

In the standby mode, the internal clock is supplied only to those circuits related to functions required to release this mode and bus hold control functions. As a result, power consumption can be reduced to 1/10 the level of normal operation in either native or emulation mode.

The standby mode is released by inputting a RESET signal or an external interrupt (NMI, INT).

The bus hold function is effective during standby mode. The CPU returns to standby mode when the bus hold request is removed.

During standby mode, all control outputs are disabled and the addres/data bus will be at either high or low levels.

Instruction Set

Symbols

Preceding the instruction set, several tables explain symbols, abbreviations, and codes.

Clocks

In the Clocks column of the instruction set, the numbers cover these operations: instruction decoding, effective address calculation, operand fetch, and instruction execution.

Clock timings assume the instruction has been pre-fetched and is present in the four-byte instruction queue. Otherwise, add four clocks for each byte not present.

For instructions that reference memory operands, the number on the left side of the slash (/) is for byte operands and the number on the right side is for word operands.

For conditional control transfer or branch instructions, the number on the left side of the slash is applicable if the transfer or branch takes place. The number on the right side is applicable if it does not take place.

If a range of numbers is given, the execution time depends on the operands involved.

Symbols

Symbol	Meaning
acc	Accumulator (AW or AL)
disp	Displacement (8 or 16 bits)
dmem	Direct memory address
dst	Destination operand or address
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
far_label	Label within a different program segment
far_proc	Procedure within a different program segment
fp_op	Floating point instruction operation
imm	8- or 16-bit immediate operand
imm3/4	3- or 4-bit immediate bit offset
imm8	8-bit immediate operand
imm16	16-bit immediate operand
mem	Memory field (000 to 111); 8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
memptr16	Word containing the destination address within the current segment
memptr32	Double word containing a destination address in another segment
mod	Mode field (00 to 10)
near_label	Label within the current segment
near_proc	Procedure within the current segment
offset	Immediate offset data (16 bits)
pop_value	Number of bytes to discard from the stack
reg	Register field (000 to 111); 8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
regptr	16-bit register containing a destination address within the current segment
regptr16	Register containing a destination address within the current segment
seg	Immediate segment data (16 bits)
short_label	Label between −128 and +127 bytes from the end of the current instruction

Symbols (cont)

Symbol	Meaning
sr	Segment register
src	Source operand or address
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1 bit)
AC	Auxiliary carry flag
AH	Accumulator (high byte)
AL	Accumulator (low byte)
AND ∧	Logical product
AW	Accumulator (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
BP	Base pointer (16 bits)
BRK	Break flag
BW	BW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
CW	CW register (16 bits)
CY	Carry flag
DH	DW register (high byte)
DIR	Direction flag
DL	DW register (low byte)
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DW	DW register (16 bits)
IE	Interrupt enable flag
IX	Index register (source) (16 bits)

Symbol	Meaning
IY	Index register (destination) (16 bits)
MD	Mode flag
OR V	Logical sum
P	Parity flag
PC	Program counter (16 bits)
PS	Program segment register (16 bits)
PSW	Program status word (16 bits)
R	Register set
S	Sign extend operand field S = 0 No sign extension S = 1 Sign extend immediate byte operand
S	Sign flag
SP	Stack pointer (16 bits)
SS	Stack segment register (16 bits)
TA	Temporary register A (16 bits)
TB	Temporary register B (16 bits)
TC	Temporary register C (16 bits)
V	Overflow flag
W	Word/byte field (0 to 1)
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
XOR ⊻	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value
Z	Zero flag
( )	Values in parentheses are memory contents
←	Transfer direction
+	Addition
−	Subtraction
x	Multiplication
÷	Division
%	Modulo

3a

Flag Operations

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to result
u	Undefined
R	Restored to previous state

Memory Addressing Modes

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Register Selection (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Register Selection

sr	Segment Register
00	DS1
01	PS
10	SS
11	DS0

Instruction Set

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer Instructions																									
MOV	reg, reg	1	0	0	0	1	0	1	W	1	1		reg			reg		2	2						
	mem, reg	1	0	0	0	1	0	0	W	mod			reg			mem		9/13	2-4						
	reg, mem	1	0	0	0	1	0	1	W	mod			reg			mem		11/15	2-4						
	mem, imm	1	1	0	0	0	1	1	W	mod		0	0	0		mem		11/15	3-6						
	reg, imm	1	0	1	1	W		reg										4	2-3						
	acc, dmem	1	0	1	0	0	0	0	W									10/14	3						
	dmem, acc	1	0	1	0	0	0	1	W									9/13	3						
	sr, reg16	1	0	0	0	1	1	1	0	1	1	0	sr			reg		2	2						
	sr, mem16	1	0	0	0	1	1	1	0	mod		0	sr			mem		11/15	2-4						
	reg16, sr	1	0	0	0	1	1	0	0	1	1	0	sr			reg		2	2						
	mem16, sr	1	0	0	0	1	1	0	0	mod		0	sr			mem		10/14	2-4						
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod			reg			mem		18/26	2-4						
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod			reg			mem		18/26	2-4						
	AH, PSW	1	0	0	1	1	1	1	1									2	1						
	PSW, AH	1	0	0	1	1	1	1	0									3	1	x	x		x	x	x
LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod			reg			mem		4	2-4						
TRANS	src_table	1	1	0	1	0	1	1	1									9	1						
XCH	reg, reg	1	0	0	0	0	1	1	W	1	1		reg			reg		3	2						
	mem, reg	1	0	0	0	0	1	1	W	mod			reg			mem		16/24	2-4						
	AW, reg16	1	0	0	1	0		reg										2	1						
Repeat Prefixes																									
REPC		0	1	1	0	0	1	0	1									2	1						
REPNC		0	1	1	0	0	1	0	0									2	1						
REP REPE REPZ		1	1	1	1	0	0	1	1									2	1						
REPNE REPNZ		1	1	1	1	0	0	1	0									2	1						
Block Transfer Instructions																									
MOVBK	dst, src	1	0	1	0	0	1	0	W									11 + 8n	1						
CMPBK	dst, src	1	0	1	0	0	1	1	W									7 + 14n	1	x	x	x	x	x	x
CMPM	dst	1	0	1	0	1	1	1	W									7 + 10n	1	x	x	x	x	x	x
LDM	src	1	0	1	0	1	1	0	W									7 + 9n	1						
STM	dst	1	0	1	0	1	0	1	W									7 + 4n	1						
										n = number of transfers															
I/O Instructions																									
IN	acc, imm8	1	1	1	0	0	1	0	W									9/13	2						
	acc, DW	1	1	1	0	1	1	0	W									8/12	1						
OUT	imm8, acc	1	1	1	0	0	1	1	W									8/12	2						
	DW, acc	1	1	1	0	1	1	1	W									8/12	1						
INM	dst, DW	0	1	1	0	1	1	0	W									9 + 8n	1						
OUTM	DW, src	0	1	1	0	1	1	1	W									9 + 8n	1						
										n = number of transfers															

3a

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
BCD Instructions																									
ADJBA		0	0	1	1	0	1	1	1									3	1	x	x	u	u	u	u
ADJ4A		0	0	1	0	0	1	1	1									3	1	x	x	u	x	x	x
ADJBS		0	0	1	1	1	1	1	1									7	1	x	x	u	u	u	u
ADJ4S		0	0	1	0	1	1	1	1									7	1	x	x	u	x	x	x
ADD4S		0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	7 + 19n	2	u	x	u	u	u	x
SUB4S		0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	7 + 19n	2	u	x	u	u	u	x
CMP4S		0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	7 + 19n	2	u	x	u	u	u	x
ROL4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	25	3						
		1	1	0	0	0		reg																	
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	28	3-5						
		mod		0	0	0		mem																	
ROR4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	29	3						
		1	1	0	0	0		reg																	
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	33	3-5						
		mod		0	0	0		mem																	
								n = number of BCD digits divided by 2																	
Data Type Conversion Instructions																									
CVTBD		1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	15	2	u	u	u	x	x	x
CVTDB		1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	7	2	u	u	u	x	x	x
CVTBW		1	0	0	1	1	0	0	0									2	1						
CVTWL		1	0	0	1	1	0	0	1									4-5	1						
Arithmetic Instructions																									
ADD	reg, reg	0	0	0	0	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	0	0	0	0	W	mod			reg			mem		16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	0	0	0	1	W	mod			reg			mem		11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	0	0		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	0	0		mem		18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	0	0	1	0	W									4	2-3	x	x	x	x	x	x
ADDC	reg, reg	0	0	0	1	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	0	0	0	W	mod			reg			mem		16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	0	0	1	W	mod			reg			mem		11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	0		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	0		mem		18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	0	1	0	W									4	2-3	x	x	x	x	x	x
SUB	reg, reg	0	0	1	0	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	1	0	1	0	0	W	mod			reg			mem		16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	0	1	0	1	W	mod			reg			mem		11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	0	1		reg		4	3-4	x	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Arithmetic Instructions (cont)																									
SUB	mem, imm	1	0	0	0	0	0	S	W	mod			1	0	1	mem		18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	0	1	1	0	W									4	2-3	x	x	x	x	x	x
SUBC	reg, reg	0	0	0	1	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	1	0	0	W	mod			reg			mem		16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	1	0	1	W	mod			reg			mem		11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	1		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	1		mem		18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	1	1	0	W									4	2-3	x	x	x	x	x	x
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	0		mem		16/24	2-4	x		x	x	x	x
	reg16	0	1	0	0	0		reg										2	1	x		x	x	x	x
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	1		mem		16/24	2-4	x		x	x	x	x
	reg16	0	1	0	0	1		reg										2	1	x		x	x	x	x
MULU	reg	1	1	1	1	0	1	1	W	1	1	1	0	0		reg		21-30	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	0		mem		27-36	2-4	u	x	x	u	u	u
MUL	reg	1	1	1	1	0	1	1	W	1	1	1	0	1		reg		33-47	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	1		mem		39-57	2-4	u	x	x	u	u	u
	reg16,reg16,imm8	0	1	1	0	1	0	1	1	1	1		reg			reg		28-34	3	u	x	x	u	u	u
	reg16,mem16,imm8	0	1	1	0	1	0	1	1	mod			reg			mem		34-44	3-5	u	x	x	u	u	u
	reg16,reg16,imm16	0	1	1	0	1	0	0	1	1	1		reg			reg		36-42	4	u	x	x	u	u	u
	reg16,mem16,imm16	0	1	1	0	1	0	0	1	mod			reg			mem		46-52	4-6	u	x	x	u	u	u
DIVU	reg	1	1	1	1	0	1	1	W	1	1	1	1	0		reg		19-25	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	0		mem		25-35	2-4	u	u	u	u	u	u
DIV	reg	1	1	1	1	0	1	1	W	1	1	1	1	1		reg		29-43	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	1		mem		35-53	2-4	u	u	u	u	u	u
Comparison Instructions																									
CMP	reg, reg	0	0	1	1	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	1	1	1	0	0	W	mod			reg			mem		11/15	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	1	1	0	1	W	mod			reg			mem		11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	1	1		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	1	1		mem		13/17	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	1	1	1	0	W									4	2-3	x	x	x	x	x	x
Logical Instructions																									
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0		reg		2	2						
	mem	1	1	1	1	0	1	1	W	mod		0	1	0		mem		16/24	2-4						
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1		reg		2	2	x	x	x	x	x	x
	mem	1	1	1	1	0	1	1	W	mod		0	1	1		mem		16/24	2-4	x	x	x	x	x	x
TEST	reg, reg	1	0	0	0	0	1	0	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	1	0	0	0	0	1	0	W	mod			reg			mem		10/14	2-4	u	0	0	x	x	x
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0		reg		4	3-4	u	0	0	x	x	x

3a

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Logical Instructions (cont)																									
TEST	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			11/15	3-6	u	0	0	x	x	x
	acc, imm	1	0	1	0	1	0	0	W									4	2-3	u	0	0	x	x	x
AND	reg, reg	0	0	1	0	0	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	1	0	0	0	0	W	mod		reg			mem			16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	0	0	0	1	W	mod		reg			mem			11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem			18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	0	0	1	0	W									4	2-3	u	0	0	x	x	x
OR	reg, reg	0	0	0	0	1	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	0	0	1	0	0	W	mod		reg			mem			16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	0	0	1	0	1	W	mod		reg			mem			11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	0	0	1	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1	mem			18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									4	2-3	u	0	0	x	x	x
XOR	reg, reg	0	0	1	1	0	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	1	1	0	0	0	W	mod		reg			mem			16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	1	0	0	1	W	mod		reg			mem			11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	1	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0	mem			18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	0	0	1	0	W									4	2-3	u	0	0	x	x	x
Bit Manipulation Instructions																									
INS	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	35-133	3						
		1	1	reg			reg																		
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	35-133	4						
		1	1	0	0	0	reg																		
EXT	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	34-59	3						
		1	1	reg			reg																		
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	34-59	4						
		1	1	0	0	0	reg																		
TEST1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	3	3	u	0	0	u	u	x
		1	1	0	0	0	reg																		
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	12/16	3-5	u	0	0	u	u	x
		mod		0	0	0	mem																		
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	4	4	u	0	0	u	u	x
		1	1	0	0	0	reg																		
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	13/17	4-6	u	0	0	u	u	x
		mod		0	0	0	mem																		
SET1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	4	3						
		1	1	0	0	0	reg																		
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	13/21	3-5						
		mod		0	0	0	mem																		

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Bit Manipulation Instructions (cont)																									
SET1	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	5	4						
		1	1	0	0	0	reg																		
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	14/22	4-6						
		mod		0	0	0	mem																		
	CY	1	1	1	1	1	0	0	1									2	1		1				
	DIR	1	1	1	1	1	1	0	1									2	1						
CLR1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	5	3						
		1	1	0	0	0	reg																		
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	14/22	3-5						
		mod		0	0	0	mem																		
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	6	4						
		1	1	0	0	0	reg																		
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	15/27	4-6						
		mod		0	0	0	mem																		
	CY	1	1	1	1	1	0	0	0									2	1		0				
	DIR	1	1	1	1	1	1	0	0									2	1						
NOT1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	4	3						
		1	1	0	0	0	reg																		
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	18/26	3-5						
		mod		0	0	0	mem																		
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	5	4						
		1	1	0	0	0	reg																		
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	19/27	4-6						
		mod		0	0	0	mem																		
	CY	1	1	1	1	0	1	0	1									2	1		x				
Shift/Rotate Instructions																									
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0	mem			16/24	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0	reg			7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0	mem			19/27 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0	reg			7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0	mem			19/27 + n	3-5	u	x	u	x	x	x
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1	mem			16/24	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg			7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1	mem			19/27 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg			7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1	mem			19/27 + n	3-5	u	x	u	x	x	x

n = number of shifts

3a

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions (cont)																									
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1		reg		2	2	u	x	0	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1		mem		16/24	2-4	u	x	0	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1		reg		7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1		mem		19/27 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1		reg		7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1		mem		19/27 + n	3-5	u	x	u	x	x	x
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	0		mem		16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	0		mem		19/27 + n	2-4		x	u			
	reg, imm	1	1	0	0	0	0	0	W	1	1	0	0	0		reg		7 + n	3		x	u			
	mem, imm	1	1	0	0	0	0	0	W	mod		0	0	0		mem		19/27 + n	3-5		x	u			
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	1		mem		16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	1		mem		19/27 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	1		mem		19/27 + n	3-5		x	u			
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0		mem		16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0		mem		19/27 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0		mem		19/27 + n	3-5		x	u			
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1		mem		16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1		mem		19/27 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1		mem		19/27 + n	3-5		x	u			
										n = number of shifts															
Stack Manipulation Instructions																									
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0		mem		18/26	2-4						
	reg16	0	1	0	1	0		reg										8/12	1						
	sr	0	0	0		sr	1	1	0									8/12	1						
	PSW	1	0	0	1	1	1	0	0									8/12	1						
	R	0	1	1	0	0	0	0	0									35/67	1						
	imm	0	1	1	0	1	0	S	0									7/12	2-3						

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Stack Manipulation Instructions (cont)																									
POP	mem16	1	0	0	0	1	1	1	1	mod		0	0	0	mem			17/25	2-4						
	reg16	0	1	0	1	1	reg											8/12	1						
	sr	0	0	0	sr		1	1	1									8/12	1						
	PSW	1	0	0	1	1	1	0	1									8/12	1	R	R	R	R	R	R
	R	0	1	1	0	0	0	0	1									43/75	1						
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0	*imm8 = 0 : 16 imm8 > 1 : 23 + 16 (imm8 − 1)								*	4						
DISPOSE		1	1	0	0	1	0	0	1									6/10	1						
Control Transfer Instructions																									
CALL	near_proc	1	1	1	0	1	0	0	0									16/20	3						
	regptr	1	1	1	1	1	1	1	1	1	1	0	1	0	reg			14/18	2						
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0	mem			23/31	2-4						
	far_proc	1	0	0	1	1	0	1	0									21/29	5						
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1	mem			31/47	2-4						
RET		1	1	0	0	0	0	1	1									15/19	1						
	pop_value	1	1	0	0	0	0	1	0									20/24	3						
		1	1	0	0	1	0	1	1									21/29	1						
	pop_value	1	1	0	0	1	0	1	0									24/32	3						
BR	near_label	1	1	1	0	1	0	0	1									13	3						
	short_label	1	1	1	0	1	0	1	1									12	2						
	regptr	1	1	1	1	1	1	1	1	1	1	1	0	0	reg			11	2						
	memptr16	1	1	1	1	1	1	1	1	mod		1	0	0	mem			20/24	2-4						
	far_label	1	1	1	0	1	0	1	0									15	5						
	memptr32	1	1	1	1	1	1	1	1	mod		1	0	1	mem			27/35	2-4						
BV	short_label	0	1	1	1	0	0	0	0									14/4	2						
BNV	short_label	0	1	1	1	0	0	0	1									14/4	2						
BC, BL	short_label	0	1	1	1	0	0	1	0									14/4	2						
BNC, BNL	short_label	0	1	1	1	0	0	1	1									14/4	2						
BE, BZ	short_label	0	1	1	1	0	1	0	0									14/4	2						
BNE, BNZ	short_label	0	1	1	1	0	1	0	1									14/4	2						
BNH	short_label	0	1	1	1	0	1	1	0									14/4	2						
BH	short_label	0	1	1	1	0	1	1	1									14/4	2						
BN	short_label	0	1	1	1	1	0	0	0									14/4	2						
BP	short_label	0	1	1	1	1	0	0	1									14/4	2						
BPE	short_label	0	1	1	1	1	0	1	0									14/4	2						
BPO	short_label	0	1	1	1	1	0	1	1									14/4	2						
BLT	short_label	0	1	1	1	1	1	0	0									14/4	2						
BGE	short_label	0	1	1	1	1	1	0	1									14/4	2						

3a

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Control Transfer Instructions (cont)																									
BLE	short_label	0	1	1	1	1	1	1	0									14/4	2						
BGT	short_label	0	1	1	1	1	1	1	1									14/4	2						
DBNZNE	short_label	1	1	1	0	0	0	0	0									14/5	2						
DBNZE	short_label	1	1	1	0	0	0	0	1									14/5	2						
DBNZ	short_label	1	1	1	0	0	0	1	0									13/5	2						
BCWZ	short_label	1	1	1	0	0	0	1	1									13/5	2						
Interrupt Instructions																									
BRK	3	1	1	0	0	1	1	0	0									38/50	1						
	imm8	1	1	0	0	1	1	0	1									38/50	2						
BRKV		1	1	0	0	1	1	1	0									52/3	1						
RETI		1	1	0	0	1	1	1	1									27/39	1	R	R	R	R	R	R
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod		reg			mem			18/26 73-76	2-4						
BRKEM	imm8	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	38/50	3						
CPU Control Instructions																									
HALT		1	1	1	1	0	1	0	0									2	1						
BUSLOCK		1	1	1	1	0	0	0	0									2	1						
FPO1	fp_op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y		mem		11/15	2-4						
FPO2	fp_op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y		mem		11/15	2-4						
POLL		1	0	0	1	1	0	1	1									2 + 5n	1						
							n = number of times POLL pin is sampled.																		
NOP		1	0	0	1	0	0	0	0									3	1						
DI		1	1	1	1	1	0	1	0									2	1						
EI		1	1	1	1	1	0	1	1									2	1						
8080 Instruction Set Enhancements																									
RETEM		1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	27/39	2	R	R	R	R	R	R
CALLN	imm8	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	38/58	3						

μPD70116 (V30) 16-Bit Microprocessor: High-Performance, CMOS

Description

The μPD70116 (V30®) is a CMOS 16-bit microprocessor with an internal 16-bit architecture and a 16-bit external data bus. The μPD70116 instruction set is a superset of the μPD8086/8088; however, mnemonics and execution times are different. The μPD70116 additionally has a powerful instruction set including bit processing, packed BCD operations, and high-speed multiplication/ division operations. The μPD70116 can also execute the entire 8080 instruction set and comes with a standby mode that significantly reduces power consumption. It is software-compatible with the μPD70108 microprocessor.

Features

- ☐ Minimum instruction execution time: 250 ns (at 8 MHz), 200 ns to 10 MHz
- ☐ Maximum addressable memory: 1 Mbyte
- ☐ Abundant memory addressing modes
- ☐ 14 x 16-bit register set
- ☐ 101 instructions
- ☐ Instruction set is a superset of μPD8086/8088 instruction set
- ☐ Bit, byte, word, and block operations
- ☐ Bit field operation instructions
- ☐ Packed BCD instructions
- ☐ Multiplication/division instruction execution time: 2.4 μs to 7.1 μs at 8 MHz and 1.9 μs to 5.7 μs at 10 MHz
- ☐ High-speed block transfer instructions: 1 Mbyte/s at 8 MHz, 1.25 Mbyte/s at 10 MHz
- ☐ High-speed calculation of effective addresses: 2 clock cycles in any addressing mode
- ☐ Maskable (INT) and nonmaskable (NMI) interrupt inputs
- ☐ IEEE-796 bus compatible interface
- ☐ 8080 emulation mode
- ☐ CMOS technology
- ☐ Low power consumption
- ☐ Low-power standby mode
- ☐ Single power supply
- ☐ 8-MHz or 10-MHz clock

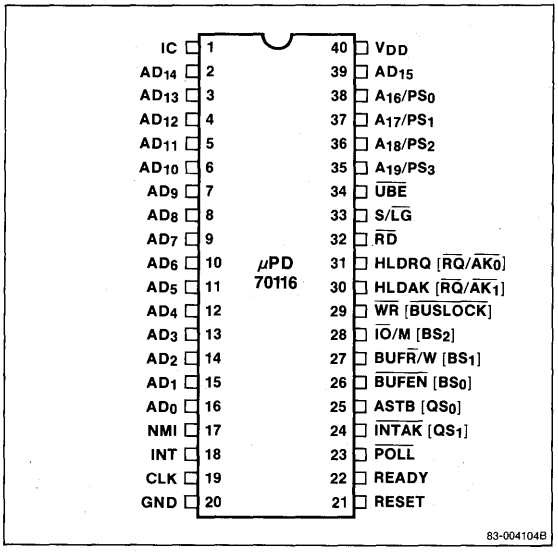
V30 is a registered trademark of NEC Corporation.

Ordering Information

Part Number	Max Frequency of Operation	Package Type
μPD70116C-8	8 MHz	40-pin plastic DIP
C-10	10 MHz	
L-8	8 MHz	44-pin PLCC
L-10	10 MHz	
GC-8	8 MHz	52-pin plastic QFP (P52GC-100-3B6)
GC-10	10 MHz	

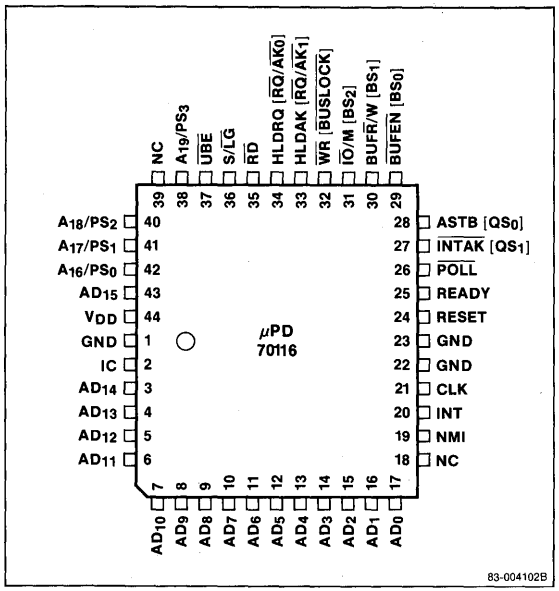
Pin Configurations

40-Pin Plastic DIP

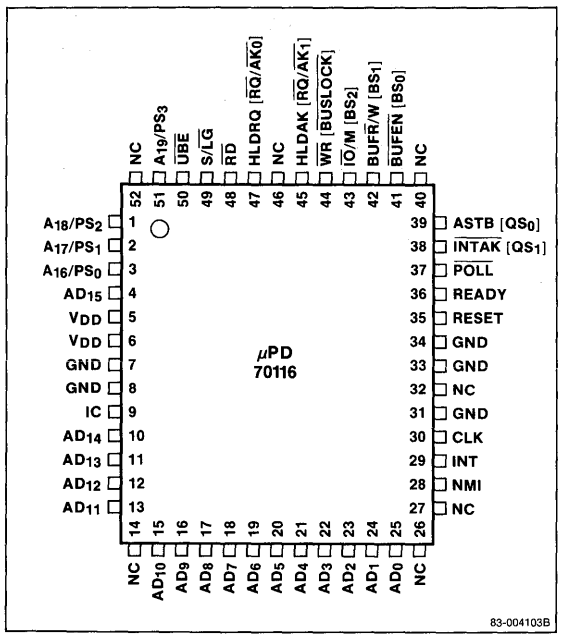


Pin Configurations (cont)

44-Pin Plastic Leaded Chip Carrier (PLCC)



52-Pin Plastic QFP



Pin Identification

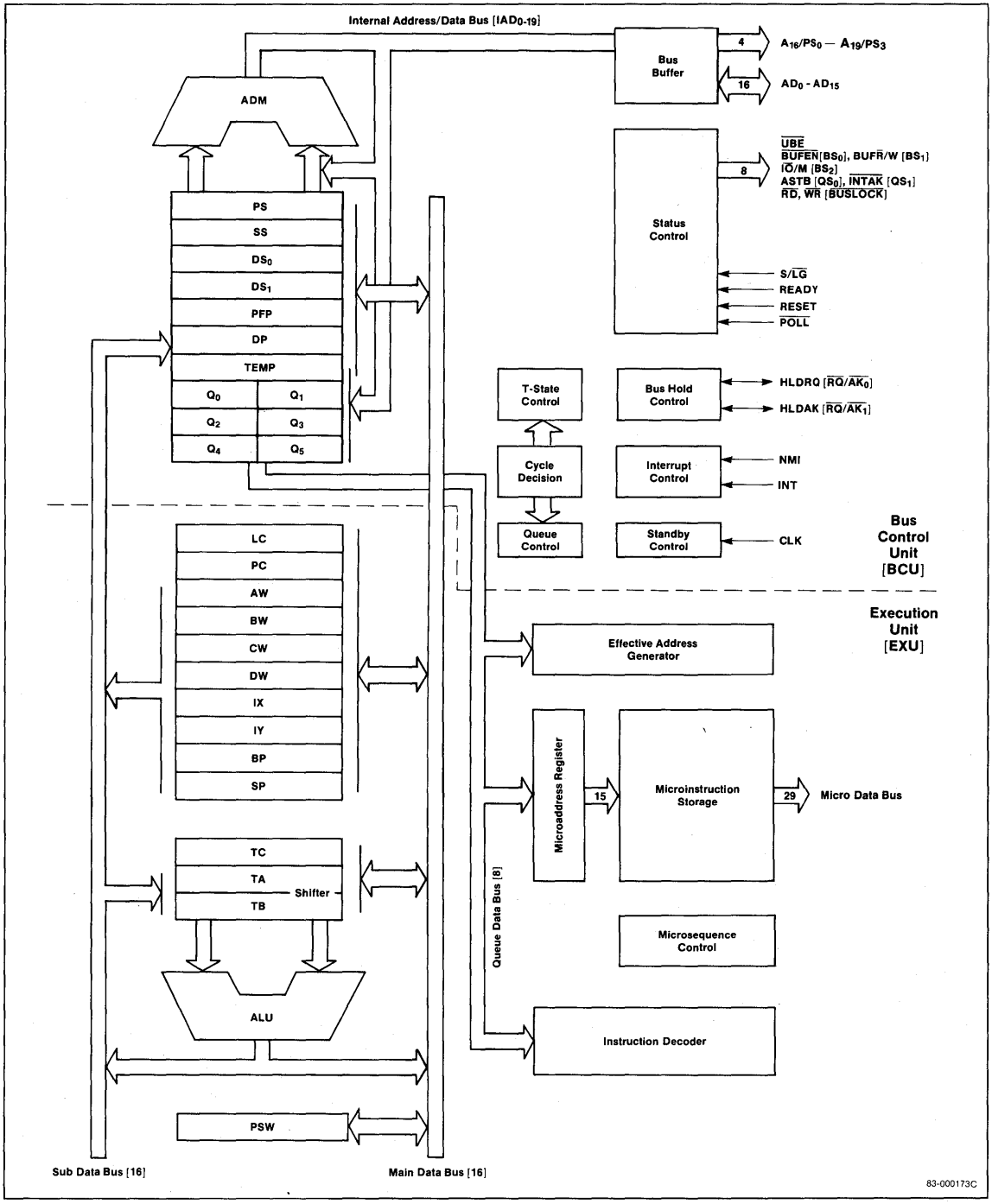
Symbol	Direction	Function
IC*		Internally connected
AD_{14} - AD_0	In/Out	Address/data bus
NMI	In	Nonmaskable interrupt input
INT	In	Maskable interrupt input
CLK	In	Clock input
GND		Ground potential
RESET	In	Reset input
READY	In	Ready input
$\overline{POLL}$	In	Poll input
$\overline{INTAK}$ (QS_1)	Out	Interrupt acknowledge output (queue status bit 1 output)
ASTB (QS_0)	Out	Address strobe output (queue status bit 0 output)
$\overline{BUFEN}$ (BS_0)	Out	Buffer enable output (bus status bit 0 output)
$BUF\overline{R}/W$ (BS_1)	Out	Buffer read/write output (bus status bit 1 output)
$\overline{IO}/M$ (BS_2)	Out	Access is I/O or memory (bus status bit 2 output)
$\overline{WR}$ ($\overline{BUSLOCK}$)	Out	Write strobe output (bus lock output)
HLDAK ($\overline{RQ}/\overline{AK_1}$)	Out (In/Out)	Hold acknowledge output, (bus hold request input/ acknowledge output 1)
HLDRQ ($\overline{RQ}/\overline{AK_0}$)	In (In/Out)	Hold request input (bus hold request input/ acknowledge output 0)
$\overline{RD}$	Out	Read strobe output
$S/\overline{LG}$	In	Small-scale/large-scale system input
$\overline{UBE}$	Out	Upper byte enable
A_{19}/PS_3 - A_{16}/PS_0	Out	Address bus, high bits or processor status output
AD_{15}	In/Out	Address/data bus, bit 15
V_{DD}		Power supply

Notes: * IC should be connected to ground.

Where pins have different functions in small- and large-scale systems, the large-scale system pin symbol and function are in parentheses.

Unused input pins should be tied to ground or V_{DD} to minimize power dissipation and prevent the flow of potentially harmful currents.

Block Diagram



3b

Pin Functions

Some pins of the μPD70116 have different functions according to whether the microprocessor is used in a small- or large-scale system. Other pins function the same way in either type of system.

AD_{15} - AD_0 [Address/Data Bus]

For small- and large-scale systems.

AD_{15} - AD_0 is a time-multiplexed address and data bus. When high, an AD bit is a one; when low, an AD bit is a zero. This bus contains the lower 16 bits of the 20-bit address during T1 of the bus cycle. It is used as a 16-bit data bus during T2, T3, and T4 of the bus cycle.

The address/data bus is a three-state bus and can be at a high or low level during standby mode. The bus will be high impedance during hold and interrupt acknowledge.

NMI [Nonmaskable Interrupt]

For small- and large-scale systems.

This pin is used to input nonmaskable interrupt requests. NMI cannot be masked by software. This input is positive edge triggered and must be held high for five clocks to guarantee recognition. Actual interrupt processing begins, however, after completion of the instruction in progress.

The contents of interrupt vector 2 determine the starting address for the interrupt-servicing routine. Note that a hold request will be accepted even during NMI acknowledge.

This interrupt will cause the μPD70116 to exit the standby mode.

INT [Maskable Interrupt]

For small- and large-scale systems.

This pin is an interrupt request that can be masked by software.

INT is active high level and is sensed during the last clock of the instruction. The interrupt will be accepted if the interrupt enable flag IE is set. The CPU outputs the $\overline{INTAK}$ signal to inform external devices that the interrupt request has been granted. INT must be asserted until the interrupt acknowledge signal is returned.

If NMI and INT interrupts occur at the same time, NMI has higher priority than INT and INT cannot be accepted. A hold request will be accepted during INT acknowledge.

This interrupt causes the μPD70116 to exit the standby mode.

CLK [Clock]

For small- and large-scale systems.

This pin is used for external clock input.

RESET [Reset]

For small- and large-scale systems.

This pin is used for the CPU reset signal. It is an active high level. Input of this signal has priority over all other operations. After the reset signal input returns to a low level, the CPU begins execution of the program starting at address FFFF0H. RESET input must be kept high for at least 4 clock cycles.

In addition to causing normal CPU start, RESET input will cause the μPD70116 to exit the standby mode.

READY [Ready]

For small- and large-scale systems.

When the memory or I/O device being accessed cannot complete data read or write within the CPU basic access time, it can generate a CPU wait state (Tw) by setting this signal to inactive (low level) and requesting a read/write cycle delay.

If the READY signal is active (high level) during either the T3 or Tw state, the CPU will not generate a wait state. READY is not synchronized internally. To guarantee correct operation, external logic must ensure that setup and hold times relative to CLK are met.

$\overline{POLL}$ [Poll]

For small- and large-scale systems.

The CPU checks this input upon execution of the POLL instruction. If the input is low, then execution continues. If the input is high, the CPU will check the $\overline{POLL}$ input every five clock cycles until the input becomes low again.

The $\overline{POLL}$ and READY functions are used to synchronize CPU program execution with the operation of external devices.

$\overline{RD}$ [Read Strobe]

For small- and large-scale systems.

The CPU outputs this strobe signal during data read from an I/O device or memory. The $\overline{IO}$/M signal is used to select between I/O and memory.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

S/$\overline{LG}$ [Small/Large]

For small- and large-scale systems.

This signal determines the operation mode of the CPU. This signal is fixed at either a high or low level. When this signal is a high level, the CPU will operate in small-scale system mode, and when low, in the large-scale system mode. A small-scale system will have at most one bus master such as a DMA controller device on the bus. A large-scale system can have more than one bus master accessing the bus as well as the CPU.

$\overline{INTAK}$ [Interrupt Acknowledge]
For small-scale systems.

The CPU generates the $\overline{INTAK}$ signal low when it accepts an INT signal.

The interrupting device synchronizes with this signal and outputs the interrupt vector to the CPU via the data bus (AD_7 - AD_0). $\overline{INTAK}$ is held at a high-level in the standby mode.

ASTB [Address Strobe]
For small-scale systems.

The CPU outputs this strobe signal to latch address information at an external latch.

ASTB is held at a low level during standby mode and hold acknowledge.

$\overline{BUFEN}$ [Buffer Enable]
For small-scale systems.

This is used as the output enable signal for an external bidirectional buffer. The CPU generates this signal during data transfer operations with external memory or I/O devices or during input of an interrupt vector.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

BUF$\overline{R}$/W [Buffer Read/Write]
For small-scale systems.

The output of this signal determines the direction of data transfer with an external bidirectional buffer. A high output causes transmission from the CPU to the external device; a low signal causes data transfer from the external device to the CPU.

BUF$\overline{R}$/W is a three-state output and enters the high-impedance state during hold acknowledge.

$\overline{IO}$/M [IO/Memory]
For small-scale systems.

The CPU generates this signal to specify either I/O access or memory access. A low-level output specifies I/O and a high-level signal specifies memory.

$\overline{IO}$/M's output is three state and becomes high impedance during hold acknowledge.

$\overline{WR}$ [Write Strobe]
For small-scale systems.

The CPU generates this strobe signal during data write to an I/O device or memory. Selection of either I/O or memory is performed by the $\overline{IO}$/M signal.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

HLDAK [Hold Acknowledge]
For small-scale systems.

The HLDAK signal is used to indicate that the CPU accepts the hold request signal (HLDRQ). When this signal is a high level, the address bus, address/data bus, the control lines become high impedance.

HLDRQ [Hold Request]
For small-scale systems.

This input signal is used by external devices to request the CPU to release the address bus, address/data bus, and the control bus.

$\overline{UBE}$ [Upper Byte Enable]
For small- and large-scale systems.

UBE indicates the use of the upper eight bits (AD_{15} -AD_8) of the address/data bus during a bus cycle. This signal is active low during T1 for read, write, and interrupt acknowledge cycles when AD_{15} - AD_8 are to be used. Bus cycles in which $\overline{UBE}$ is active are shown in the following table.

Type of Bus Operation	$\overline{UBE}$	AD_0	Number of Bus Cycles
Word at even address	0	0	1
Word at odd address	0 1	1* 0**	2
Byte at even address	1	0	1
Byte at odd address	0	1	1

Notes: * First bus cycle
** Second bus cycle

$\overline{UBE}$ is low continuously during the interrupt acknowledge state.

The three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

A_{19}/PS_3 - A_{16}/PS_0 [Address Bus/Processor Status]
For small- and large-scale systems.

These pins are time multiplexed to operate as an address bus and as processor status signals.

When used as the address bus, these pins are the high 4 bits of the 20-bit memory address. During I/O access, all 4 bits output data 0.

3b

The processor status signals are provided for both memory and I/O use. PS_3 is always 0 in the native mode and 1 in 8080 emulation mode. The interrupt enable flag (IE) is output on pin PS_2. Pins PS_1 and PS_0 indicate which memory segment is being accessed.

A_{17}/PS_1	A_{16}/PS_0	Segment
0	0	Data segment 1
0	1	Stack segment
1	0	Program segment
1	1	Data segment 0

The output of these pins is three state and becomes high impedance during hold acknowledge.

QS_1, QS_0 [Queue Status]

For large-scale systems.

The CPU uses these signals to allow external devices, such as the floating-point arithmetic processor chip (μPD72091), to monitor the status of the internal CPU instruction queue.

QS_1	QS_0	Instruction Queue Status
0	0	NOP (queue does not change)
0	1	First byte of instruction
1	0	Flush queue
1	1	Subsequent bytes of instruction

The instruction queue status indicated by these signals is the status when the execution unit (EXU) accesses the instruction queue. The data output from these pins is therefore valid only for one clock cycle immediately following queue access. These status signals are provided so that the floating-point processor chip can monitor the CPU's program execution status and synchronize its operation with the CPU when control is passed to it by the FPO (Floating Point Operation) instructions. These outputs are held low level in the standby mode.

BS_2 - BS_0 [Bus Status]

For large-scale systems.

The CPU uses these status signals to allow an external bus controller to monitor what the current bus cycle is.

The external bus controller decodes these signals and generates the control signals required to perform access of the memory or I/O device.

BS_2	BS_1	BS_0	Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Program fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state

The output of these signals is three state and becomes high impedance during hold acknowledge.

$\overline{BUSLOCK}$ [Bus Lock]

For large-scale systems.

The CPU uses this signal to secure the bus while executing the instruction immediately following the BUSLOCK prefix instruction and during interrupt acknowledge cycles. It is a status signal to the other bus masters in a multiprocessor system inhibiting them from using the system bus during this time.

The output of this signal is three state and becomes high impedance during hold acknowledge. $\overline{BUSLOCK}$ is high during standby mode except if the HALT instruction has a BUSLOCK prefix.

$\overline{RQ}/\overline{AK}_1$, $\overline{RQ}/\overline{AK}_0$ [Hold Request/Acknowledge]

For large-scale systems.

These pins function as bus hold request inputs ($\overline{RQ}$) and as bus hold acknowledge outputs ($\overline{AK}$). $\overline{RQ}/\overline{AK}_0$ has a higher priority than $\overline{RQ}/\overline{AK}_1$.

These pins have three-state outputs with on-chip pull-up resistors which keep the pin at a high level when the output is high impedance. $\overline{RQ}$ inputs must be properly synchronized to CLK.

V_{DD} [Power Supply]

For small- and large-scale systems.

This pin is used for the +5 V power supply.

GND [Ground]

For small- and large-scale systems.

This pin is used for ground.

IC [Internally Connected]

This pin is used for tests performed at the factory by NEC. The μPD70116 is used with this pin at ground potential.

Absolute Maximum Ratings

$T_A = +25°C$

Power supply voltage, V_{DD}	−0.5 V to +7.0 V
Power dissipation, PD_{MAX}	0.5 W
Input voltage, V_I	−0.5 V to V_{DD} + 0.3 V
CLK input voltage, V_K	−0.5 V to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 V to V_{DD} + 0.3 V
Operating temperature at 5 MHz, T_{OPT}	−40°C to +85°C
Storage temperature, T_{STG}	−65°C to +150°C

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Capacitance

$T_A = +25°C$, $V_{DD} = 0$ V

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input capacitance	C_I		15	pF	fc = 1 MHz Unmeasured pins returned to 0 V
I/O capacitance	C_{IO}		15	pF	

DC Characteristics

μPD70116-8, μPD70116-10, $T_A = -10°C$ to +70°C, $V_{DD} = +5$ V ± 5%

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Input voltage high	V_{IH}	2.2		V_{DD} + 0.3	V	
Input voltage low	V_{IL}	−0.5		0.8	V	
CLK input voltage high	V_{KH}	3.9		V_{DD} + 1.0	V	
CLK input voltage low	V_{KL}	−0.5		0.6	V	
Output voltage high	V_{OH}	0.7 x V_{DD}			V	$I_{OH} = -400$ μA
Output voltage low	V_{OL}			0.4	V	$I_{OL} = 2.5$ mA
Input leakage current high	I_{LIH}			10	μA	$V_I = V_{DD}$
Input leakage current low	I_{LIL}			−10	μA	$V_I = 0$ V
Output leakage current high	I_{LOH}			10	μA	$V_O = V_{DD}$
Output leakage current low	I_{LOL}			−10	μA	$V_O = 0$ V
Supply current	I_{DD}	70116-8	45	80	mA	Normal operation
		8 MHz	6	12	mA	Standby mode
		70116-10	60	100	mA	Normal operation
		10 MHz	7	14	mA	Standby mode

3b

AC Characteristics

$T_A = -10°C$ to $+70°C$, $V_{DD} = +5\ V \pm 5\%$

Parameter	Symbol	μPD70116-8 Min	μPD70116-8 Max	μPD70116-10 Min	μPD70116-10 Max	Unit	Conditions
Small/Large Scale							
Clock cycle	t_{CYK}	125	500	100	500	ns	
Clock pulse width high	t_{KKH}	44		41		ns	$V_{KH} = 3.0\ V$
Clock pulse width low	t_{KKL}	60		49		ns	$V_{KL} = 1.5\ V$
Clock rise time	t_{KR}		10		5	ns	1.5 V to 3.0 V
Clock fall time	t_{KF}		10		5	ns	3.0 V to 1.5 V
READY inactive setup to CLK↓	t_{SRYLK}	−8		−10		ns	
READY inactive hold after CLK↑	t_{HKRYH}	20		20		ns	
READY active setup to CLK↑	t_{SRYHK}	$t_{KKL} - 8$		$t_{KKL} - 10$		ns	
READY active hold after CLK↑	t_{HKRYL}	20		20		ns	
Data setup time to CLK ↓	t_{SDK}	20		10		ns	
Data hold time after CLK ↓	t_{HKD}	10		10		ns	
NMI, INT, $\overline{POLL}$ setup time to CLK ↑	t_{SIK}	15		15		ns	
Input rise time (except CLK)	t_{IR}		20		20	ns	0.8 V to 2.2 V
Input fall time (except CLK)	t_{IF}		12		12	ns	2.2 V to 0.8 V
Output rise time	t_{OR}		20		20	ns	0.8 V to 2.2 V
Output fall time	t_{OF}		12		12	ns	2.2 V to 0.8 V
Small Scale							
Address delay time from CLK ↓	t_{DKA}	10	60	10	48	ns	
Address hold time from CLK ↓	t_{HKA}	10		10		ns	
PS delay time from CLK ↓	t_{DKP}	10	60	10	50	ns	
PS float delay time from CLK ↑	t_{FKP}	10	60	10	50	ns	
Address setup time to ASTB ↓	t_{SAST}	$t_{KKL} - 30$		$t_{KKL} - 30$		ns	
Address float delay time from CLK ↓	t_{FKA}	t_{HKA}	60	t_{HKA}	50	ns	$C_L = 100\ pF$
ASTB ↑ delay time from CLK ↓	t_{DKSTH}		50		40	ns	
ASTB ↓ delay time from CLK ↑	t_{DKSTL}		55		45	ns	
ASTB width high	t_{STST}	$t_{KKL} - 10$		$t_{KKL} - 10$		ns	
Address hold time from ASTB ↓	t_{HSTA}	$t_{KKH} - 10$		$t_{KKH} - 10$		ns	

AC Characteristics (cont)

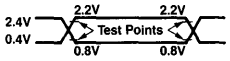
$T_A = -10°C$ to $+70°C$, $V_{DD} = +5\ V \pm 5\%$

Parameter	Symbol	μPD70116-8 Min	μPD70116-8 Max	μPD70116-10 Min	μPD70116-10 Max	Unit	Conditions
Small Scale (cont)							
Control delay time from CLK	t_{DKCT}	10	65	10	55	ns	
Address float to $\overline{RD}$↓	t_{AFRL}	0		0		ns	
$\overline{RD}$ ↓ delay time from CLK ↓	t_{DKRL}	10	80	10	70	ns	
$\overline{RD}$ ↑ delay time from CLK ↓	t_{DKRH}	10	80	10	60	ns	
Address delay time from $\overline{RD}$ ↑	t_{DRHA}	$t_{CYK} - 40$		$t_{CYK} - 35$		ns	
$\overline{RD}$ width low	t_{RR}	$2t_{CYK}-50$		$2t_{CYK}-40$		ns	$C_L = 100$ pF
Data output delay time from CLK ↓	t_{DKD}	10	60	10	50	ns	
Data float delay time from CLK ↓	t_{FKD}	10	60	10	50	ns	
$\overline{WR}$ width low	t_{WW}	$2t_{CYK}-40$		$2t_{CYK}-35$		ns	
HLDRQ setup time to CLK ↑	t_{SHQK}	20		20		ns	
HLDAK delay time from CLK ↓	t_{DKHA}	10	100	10	60	ns	
Large Scale							
Address delay time from CLK ↓	t_{DKA}	10	60	10	48	ns	
Address hold time from CLK ↓	t_{HKA}	10		10		ns	
PS delay time from CLK ↓	t_{DKP}	10	60	10	50	ns	
PS float delay time from CLK ↑	t_{FKP}	10	60	10	50	ns	
Address float delay time from CLK ↓	t_{FKA}	t_{HKA}	60	t_{HKA}	50	ns	
Address delay time from $\overline{RD}$ ↑	t_{DRHA}	$t_{CYK} - 40$		$t_{CYK} - 35$		ns	
ASTB delay time from BS ↓	t_{DBST}		15		15	ns	
BS ↓ delay time from CLK ↑	t_{DKBL}	10	60	10	50	ns	
BS ↑ delay time from CLK ↓	t_{DKBH}	10	65	10	50	ns	
$\overline{RD}$ ↓ delay time from address float	t_{DAFRL}	0		0		ns	$C_L = 100$ pF
$\overline{RD}$ ↓ delay time from CLK ↓	t_{DKRL}	10	80	10	70	ns	
$\overline{RD}$ ↑ delay time from CLK ↓	t_{DKRH}	10	80	10	60	ns	
$\overline{RD}$ width low	t_{RR}	$2t_{CYK}-50$		$2t_{CYK}-40$		ns	
Date output delay time from CLK ↓	t_{DKD}	10	60	10	50	ns	
Data float delay time from CLK ↓	t_{FKD}	10	60	10	50	ns	
$\overline{AK}$ delay time from CLK ↓	t_{DKAK}		50		40	ns	
$\overline{RQ}$ setup time to CLK ↑	t_{SRQK}	10		9		ns	
$\overline{RQ}$ hold time from CLK ↓	t_{HKRQ1}	0		0		ns	
$\overline{RQ}$ hold time from CLK ↑	t_{HKRQ2}	30		20		ns	

3b

Timing Waveforms

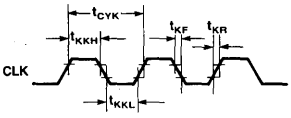
AC Test Input Waveform [Except CLK]



AC Output Test Points

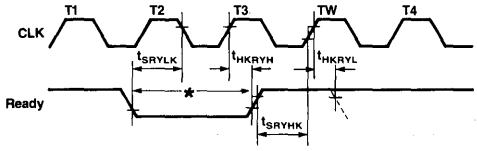
49-000247A

Clock Timing



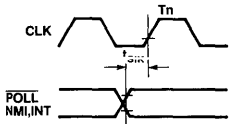
49-000248A

Wait [Ready] Timing

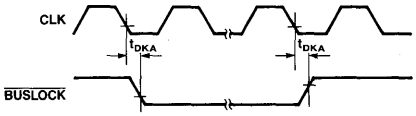


* READY input level must not be changed during this interval.

$\overline{\text{POLL}}$,NMI, INT Input Timing



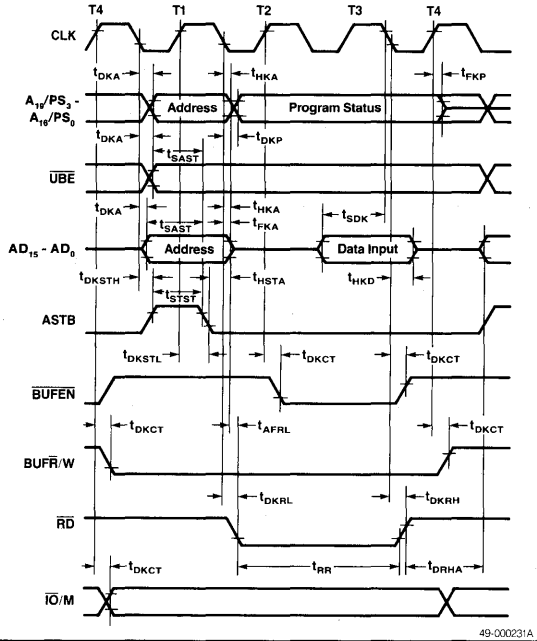
$\overline{\text{BUSLOCK}}$ Output Timing



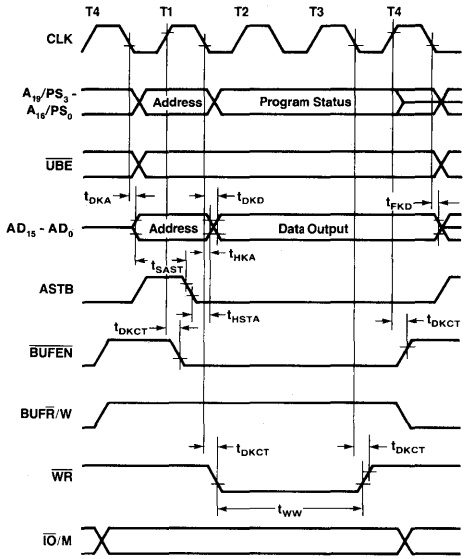
49-000249B

Timing Waveforms (cont)

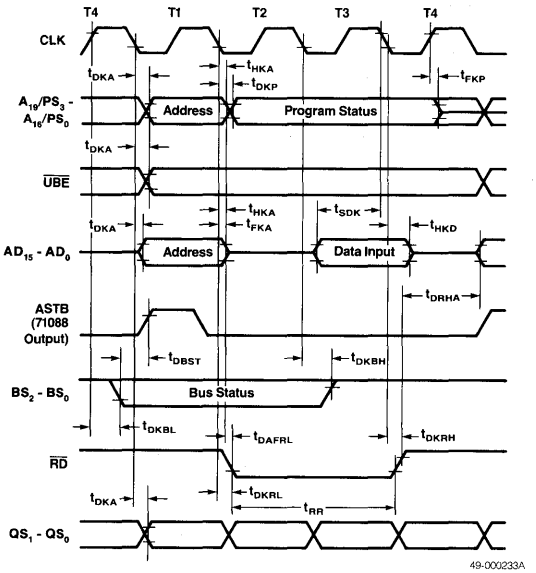
Read Timing [Small Scale]



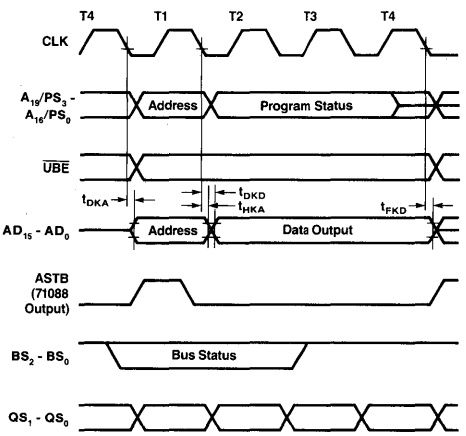
Write Timing [Small Scale]



Read Timing [Large Scale]



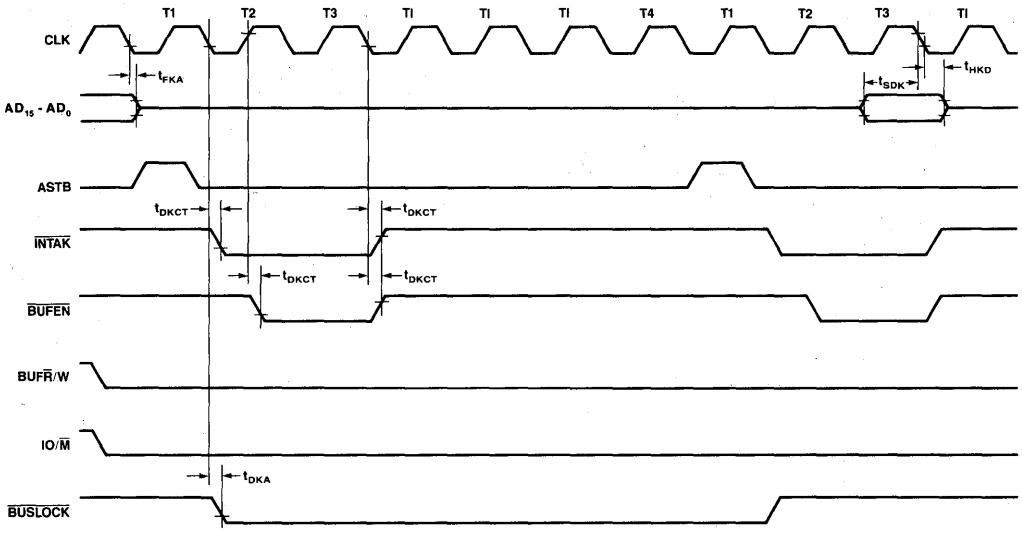
Write Timing [Large Scale]



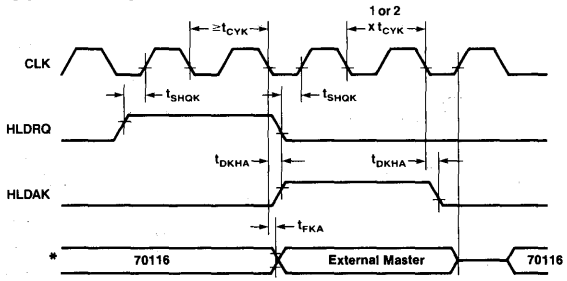
3b

Timing Waveforms (cont)

Interrupt Acknowledge Timing



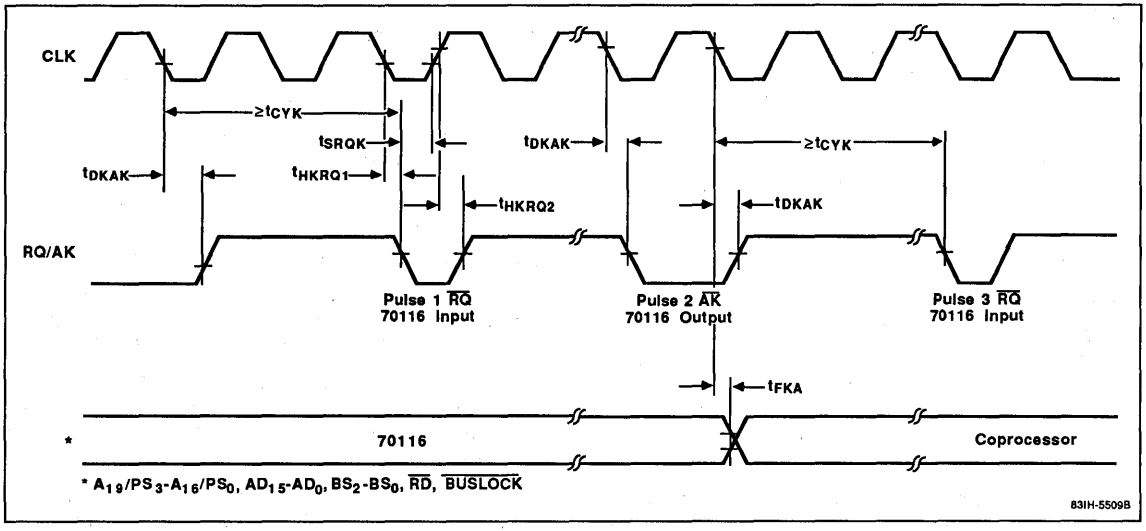
Hold Request/Acknowledge Timing [Small Scale]



*: A_{19}/PS_3 - A_{16}/PS_0, AD_{15} - AD_0, $\overline{RD}$, $\overline{UBE}$, $\overline{IO}$/M, BUF$\overline{R}$/W, $\overline{WR}$ $\overline{BUFEN}$

49-000236B

Timing Waveforms (cont)



3b

Register Configuration

Program Counter [PC]

The program counter is a 16-bit binary counter that contains the segment offset address of the next instruction which the EXU is to execute.

The PC increments each time the microprogram fetches an instruction from the instruction queue. A new location value is loaded into the PC each time a branch, call, return, or break instruction is executed. At this time, the contents of the PC are the same as the Prefetch Pointer (PFP).

Prefetch Pointer [PFP]

The prefetch pointer (PFP) is a 16-bit binary counter which contains a segment offset which is used to calculate a program memory address that the bus control unit (BCU) uses to prefetch the next word for the instruction queue. The contents of PFP are an offset from the PS (Program Segment) register.

The PFP is incremented each time the BCU prefetches an instruction from the program memory. A new location will be loaded into the PFP whenever a branch, call, return, or break instruction is executed. At that time the contents of the PFP will be the same as those of the PC (Program Counter).

Segment Registers [PS, SS, DS_0, and DS_1]

The memory addresses accessed by the μPD70116 are divided into 64K-byte logical segments. The starting (base) address of each segment is specified by a 16-bit segment register, and the offset from this starting address is specified by the contents of another register or by the effective address.

These are the four types of segment registers used.

Segment Register	Default Offset
PS (Program Segment)	PFP
SS (Stack Segment)	SP, effective address
DS_0 (Data Segment 0)	IX, effective address
DS_1 (Data Segment 1)	IY

General-Purpose Registers [AW, BW, CW, and DW]

There are four 16-bit general-purpose registers. Each one can be used as one 16-bit register or as two 8-bit registers by dividing them into their high and low bytes (AH, AL, BH, BL, CH, CL, DH, DL).

Each register is also used as a default register for processing specific instructions. The default assignments are:

AW: Word multiplication/division, word I/O, data conversion, translation, BCD rotation.

AL: Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation

AH: Byte multiplication/division

BW: Translation

CW: Loop control branch, repeat prefix

CL: Shift instructions, rototation instructions, BCD operations

DW: Word multiplication/division, indirect addressing I/O

Pointers [SP, BP] and Index Registers [IX, IY]

These registers serve as base pointers or index registers when accessing the memory using based addressing, indexed addressing, or based indexed addressing.

These registers can also be used for data transfer and arithmetic and logical operations in the same manner as the general-purpose registers. They cannot be used as 8-bit registers.

Also, each of these registers acts as a default register for specific operations. The default assignments are:

SP: Stack operations

IX: Block transfer (source), BCD string operations

IY: Block transfer (destination), BCD string operations

Program Status Word [PSW]

The program status word consists of the following six status and four control flags.

Status Flags

- V (Overflow)
- S (Sign)
- Z (Zero)
- AC (Auxiliary Carry)
- P (Parity)
- CY (Carry)

Control Flags

- MD (Mode)
- DIR (Direction)
- IE (Interrupt Enable)
- BRK (Break)

When the PSW is pushed on the stack, the word images of the various flags are as shown here.

PSW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MD	1	1	1	V	DIR	IE	BRK	S	Z	0	AC	0	P	1	CY

The status flags are set and reset depending upon the result of each type of instruction executed.

Instructions are provided to set, reset, and complement the CY flag directly.

Other instructions set and reset the control flags and control the operation of the CPU. The MD flag can be set/reset only by the BRKEM, RETEM, CALLN, and RETI instructions.

The MD flag can be set/reset only in between executions of BRKEM and RETEM. MD will not be restored, even as the RETI or POP PSW instruction is executed.

High-Speed Execution of Instructions

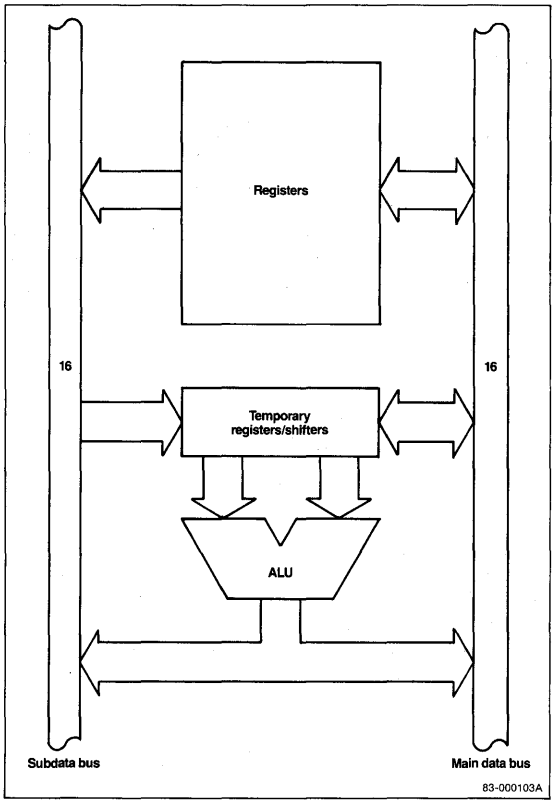
This section highlights the major architectural features that enhance the performance of the μPD70116.

- Dual data bus in EXU
- Effective address generator
- 16/32-bit temporary registers/shifters (TA, TB)
- 16-bit loop counter (LC)
- PC and PFP

Dual Data Bus Method

To reduce the number of processing steps for instruction execution, the dual data bus method has been adopted for the μPD70116 (figure 1). The two data buses (the main data bus and the subdata bus) are both 16 bits wide. For addition/subtraction and logical and comparison operations, processing time has been reduced by some 30% over single-bus systems.

Figure 1. Dual Data Buses



Example

ADD AW, BW ; AW ← AW + BW

Single Bus	Dual Bus
Step 1 TA ← AW	TA ← AW, TB ← BW
Step 2 TB ← BW	AW ← TA + TB
Step 3 AW ← TA + TB	

Effective Address Generator [EAG]

The Effective Address Generator (EAG) (figure 2) is a dedicated block of high-speed logic that computes effective addresses in two clock cycles. If an instruction uses memory, EAG decodes the second and/or third instruction bytes to determine the addressing mode, initiates any bus cycles needed to fetch data required to compute the effective address, and stores the computed effective address in the Data Pointer (DP) register.

Calculating an effective address by the microprogramming method normally requires 5 to 12 clock cycles. This circuit requires only two clock cycles for addresses to be generated for any addressing mode. Thus, processing is several times faster.

16/32-Bit Temporary Registers/Shifters [TA, TB]

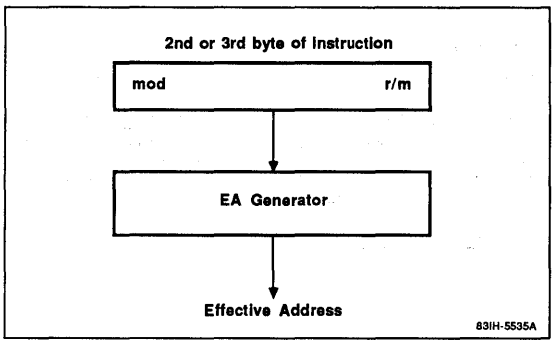
These 16-bit temporary registers/shifters (TA, TB) are provided for multiplication/division and shift/rotation instructions.

These circuits have decreased the execution time of multiplication/division instructions. In fact, these instructions can be executed about four times faster than with the microprogramming method.

TA + TB: 32-bit temporary register/shifter for multiplication and division instructions.

TB: 16-bit temporary register/shifter for shift/rotation instructions.

Figure 2. Effective Address Generator



3b

Loop Counter [LC]

This counter is used to count the number of loops for a primitive block transfer instruction controlled by a repeat prefix instruction and the number of shifts that will be performed for a multiple bit shift/rotation instruction.

The processing performed for a multiple bit rotation of a register is shown below. The average speed is approximately doubled over the microprogram method.

Example

RORC AW, CL ; CL = 5

Microprogram method	**LC method**
8 + (4 x 5) = 28 clocks	7 + 5 = 12 clocks

Program Counter and Prefetch Pointer [PC and PFP]

The μPD70116 microprocessor has a program counter, (PC) which addresses the program memory location of the instruction to be executed next, and a prefetch pointer(PFP), which addresses the program memory location to be accessed next. Both functions are provided in hardware. A time saving of several clocks is realized for branch, call, return, and break instruction execution, compared with microprocessors that have only one instruction pointer.

Enhanced Instructions

In addition to the μPD8088/86 instructions, the μPD70116 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP R	Pops 8 general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Enhanced Stack Operation Instructions

PUSH imm

This instruction allows immediate data to be pushed onto the stack.

PUSH R/POP R

These instructions allow the contents of the eight general registers to be pushed onto or popped from the stack with a single instruction.

Enhanced Multiplication Instructions

MUL reg16, imm16/MUL mem16, imm16

These instructions allow the contents of a register or memory location to be multiplied by immediate data.

Enhanced Shift and Rotate Instructions

SHL reg, imm8/SHR reg, imm8/SHRA reg, imm8

These instructions allow the contents of a register to be shifted by the number of bits defined by the immediate data.

ROL reg, imm8/ROR reg, imm8/ROLC reg, imm8/ RORC reg, imm8

These instructions allow the contents of a register to be rotated by the number of bits defined by the immediate data.

Check Array Boundary Instruction

CHKIND reg16, mem32

This instruction is used to verify that index values pointing to the elements of an array data structure are within the defined range. The lower limit of the array should be in memory location mem32, the upper limit in mem32 + 2. If the index value in reg16 is not between these limits when CHKIND is executed, a BRK 5 will occur. This causes a jump to the location in interrupt vector 5.

Block I/O Instructions

OUTM DW, src-block/INM dst-block, DW

These instructions are used to output or input a string to or from memory, when preceded by a repeat prefix.

Stack Frame Instructions

PREPARE imm16, imm8

This instruction is used to generate the stack frames required by block-structured languages, such as PASCAL and Ada. The stack frame consists of two areas. One area has a pointer that points to another frame which has variables that the current frame can access. The other is a local variable area for the current procedure.

DISPOSE

This instruction releases the last stack frame generated by the PREPARE instruction. It returns the stack and base pointers to the values they had before the PREPARE instruction was used to call a procedure.

Unique Instructions

In addition to the μPD8088/86 instructions and the enhanced instructions, the μPD70116 has the following unique instructions.

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	Adds packed decimal strings
SUB4S	Subtracts one packed decimal string from another
CMP4S	Compares two packed decimal strings
ROL4	Rotates one BCD digit left through AL lower 4 bits
ROR4	Rotates one BCD digit right through AL lower 4 bits
TEST1	Tests a specified bit and sets/resets Z flag
NOT1	Inverts a specified bit
CLR1	Clears a specified bit
SET1	Sets a specified bit
REPC	Repeats next instruction until CY flag is cleared
REPNC	Repeats next instruction until CY flag is set
FPO2	Additional floating point processor call

Variable Length Bit Field Operation Instructions

This category has two instructions: INS (Insert Bit Field) and EXT (Extract Bit Field). These instructions are highly effective for computer graphics and high-level languages. They can, for example, be used for data structures such as packed arrays and record type data used in PASCAL.

INS reg8, reg8/INS reg8, imm4

This instruction (figure 3) transfers low bits from the 16-bit AW register (the number of bits is specified by the second operand) to the memory location specified by the segment base (DS_1 register) plus the byte offset (IY register). The starting bit position within this byte is specified as an offset by the lower 4-bits of the first operand.

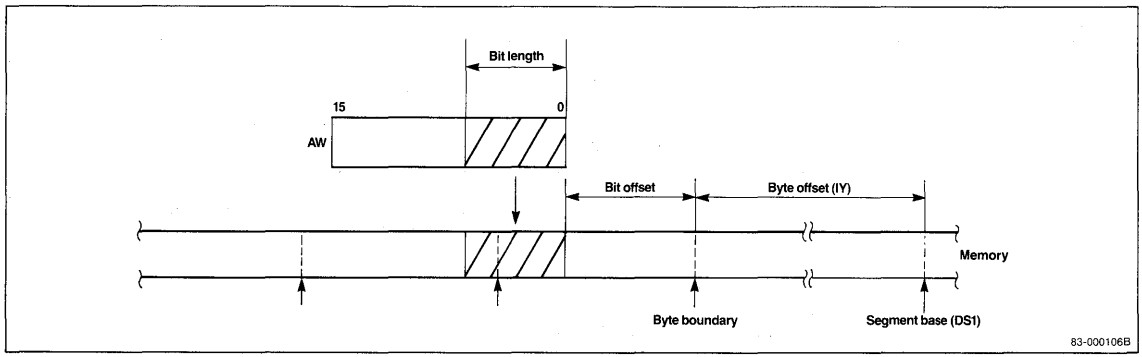
After each complete data transfer, the IY register and the register specified by the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may specify the number of bits transferred (second operand). Because the maximum transferable bit length is 16-bits, only the lower 4-bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

3b

Figure 3. Bit Field Insertion



EXT reg8, reg8/EXT reg8, imm4

This instruction (figure 4) loads to the AW register the bit field data whose bit length is specified by the second operand of the instruction from the memory location that is specified by the DS0 segment register (segment base), the IX index register (byte offset), and the lower 4-bits of the first operand (bit offset).

After the transfer is complete, the IX register and the lower 4-bits of the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may be specified for the second operand. Because the maximum transferrable bit length is 16 bits, however, only the lower 4-bits of the specified register (0H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Packed BCD Operation Instructions

The instructions described here process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte-format operands (ROR4, ROL4). Packed BCD strings may be from 1 to 254 digits in length.

When the number of digits is even, the zero and carry flags will be set according to the result of the operation. When the number of digits is odd, the zero and carry flags may not be set correctly in this case, (CL = odd), the zero flag will not be set unless the upper 4 bits of the highest byte are all zero. The carry flag will not be set unless there is a carry out of the upper 4 bits of the highest byte. When CL is odd, the contents of the upper 4 bits of the highest byte of the result are undefined.

ADD4S

This instruction adds the packed BCD string addressed by the IX index register to the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the overflow flag (V), the carry flag (CY), and zero flag (Z).

BCD string (IY, CL) ← BCD string (IY, CL) + BCD string (IX, CL)

SUB4S

This instruction subtracts the packed BCD string addressed by the IX index register from the packed BCD string addressed by the IY register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the overflow flag (V), the carry flag (CY), and zero flag (Z).

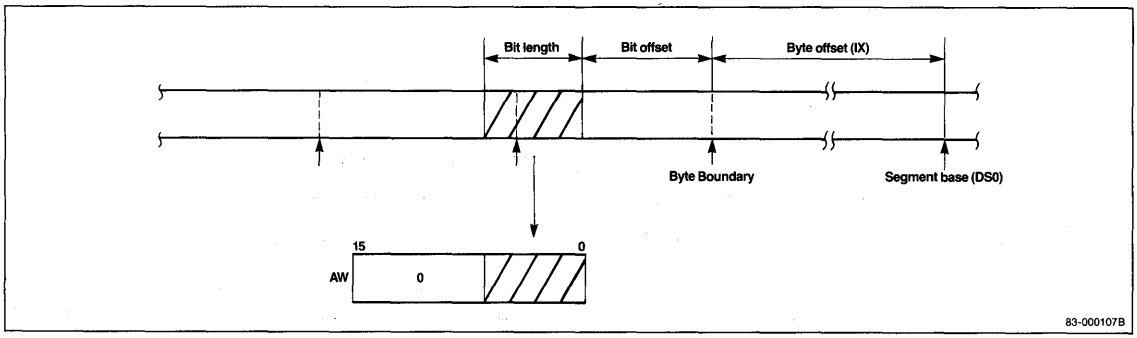
BCD string (IY, CL) ← BCD string (IY, CL) − BCD String (IX, CL)

CMP4S

This instruction performs the same operation as SUB4S except that the result is not stored and only the overflow (V), carry flags (CY) and zero flag (Z) are affected.

BCD string (IY, CL) − BCD string (IX, CL)

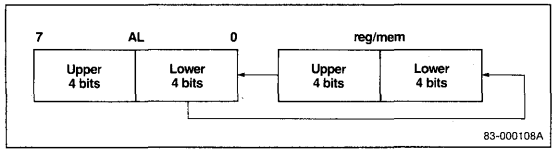
Figure 4. Bit Field Extraction



ROL4

This instruction (figure 5) treats the byte data of the register or memory operand specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the left.

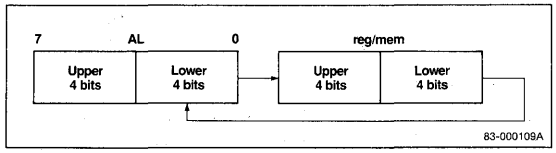
Figure 5. BCD Rotate Left (ROL4)



ROR4

This instruction (figure 6) treats the byte data of the register or memory specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the right.

Figure 6. BCD Rotate Right (ROR4)



Bit Manipulation Instructions

TEST1

This instruction tests a specific bit in a register or memory location. If the bit is 1, the Z flag is reset to 0. If the bit is 0, the Z flag is set to 1.

NOT1

This instruction inverts a specific bit in a register or memory location.

CLR1

This instruction clears a specific bit in a register or memory location.

SET1

This instruction sets a specific bit in a register or memory location.

Repeat Prefix Instructions

REPC

This instruction causes the μPD70116 to repeat the following primitive block transfer instruction until the CY flag becomes cleared or the CW register becomes zero.

REPNC

This instruction causes the μPD70116 to repeat the following primitive block transfer instruction until the CY flag becomes set or the CW register becomes zero.

Floating Point Instruction

FPO2

This instruction is in addition to the μPD8088/86 floating point instruction, FPO1. These instructions are covered in a later section.

Mode Operation Instruction

The μPD70116 has two operating modes (figure 7). One is the native mode which executes μPD8088/86, enhanced and unique instructions. The other is the 8080 emulation mode in which the instruction set of the μPD8080AF is emulated. A mode flag (MD) is provided to select between these two modes. Native mode is selected when MD is 1 and emulation mode when MD is 0. MD is set and reset, directly and indirectly, by executing the mode manipulation instructions.

Two instructions are provided to switch operation from the native mode to the emulation mode and back.

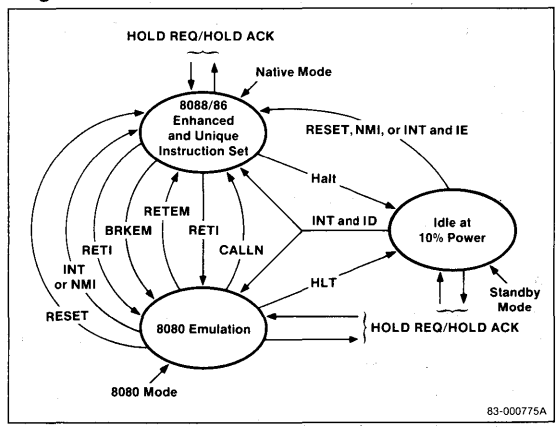
BRKEM (Break for Emulation)
RETEM (Return from Emulation)

Two instructions are used to switch from the emulation mode to the native mode and back.

CALLN (Call Native Routine)
RETI (Return from Interrupt)

The system will return from the 8080 emulation mode to the native mode when the RESET signal is present, or when an external interrupt (NMI or INT) is present.

Figure 7. V30 Modes



3b

BRKEM imm8

This is the basic instruction used to start the 8080 emulation mode. This instruction operates exactly the same as the BRK instruction, except that BRKEM resets the mode flag (MD) to 0. PSW, PS, and PC are saved to the stack. MD is then reset and the interrupt vector specified by the operand imm8 of this command is loaded into PS and PC.

The instruction codes of the interrupt processing routine jumped to are then fetched. Then the CPU executes these codes as μPD8080AF instructions.

In 8080 emulation mode, registers and flags of the μPD8080AF are performed by the following registers and flags of the μPD70116.

	μPD8080AF	μPD70116
Registers:	A	AL
	B	CH
	C	CL
	D	DH
	E	DL
	H	BH
	L	BL
	SP	BP
	PC	PC
Flags:	C	CY
	Z	Z
	S	S
	P	P
	AC	AC

In the native mode, SP is used for the stack pointer. In the 8080 emulation mode this function is performed by BP.

This use of independent stack pointers allows independent stack areas to be secured for each mode and keeps the stack of one of the modes from being destroyed by an erroneous stack operation in the other mode.

The SP, IX, IY and AH registers and the four segment registers (PS, SS, DS_0, and DS_1) used in the native mode are not affected by operations in 8080 emulation mode.

In the 8080 emulation mode, the segment register for instructions is determined by the PS register (set automatically by the interrupt vector) and the segment register for data is the DS_0 register (set by the programmer immediately before the 8080 emulation mode is entered).

It is prohibited to nest BRKEM instructions.

RETEM [no operand]

When RETEM is executed in 8080 emulation mode (interpreted by the CPU as a μPD8080AF instruction), the CPU restores PS, PC, and PSW (as it would when returning from an interrupt processing routine), and returns to the native mode. At the same time, the contents of the mode flag (MD) which was saved to the stack by the BRKEM instruction, is restored to MD = 1. The CPU is set to the native mode.

CALLN imm8

This instruction makes it possible to call the native mode subroutines from the 8080 emulation mode. To return from subroutine to the 8080 emulation mode, the RETI instruction is used.

The processing performed when this instruction is executed in the 8080 emulation mode (it is interpreted by the CPU as μPD8080AF instruction), is similar to that performed when a BRK instruction is executed in the native mode. The imm8 operand specifies an interrupt vector type. The contents of PS, PC, and PSW are pushed on the stack and an MD flag value of 0 is saved. The mode flag is set to 1 and the interrupt vector specified by the operand is loaded into PS and PC.

RETI [no operand]

This is a general-purpose instruction used to return from interrupt routines entered by the BRK instruction or by an external interrupt in the native mode. When this instruction is executed at the end of a subroutine entered by the execution of the CALLN instruction, the operation that restores PS, PC, and PSW is exactly the same as the native mode execution. When PSW is restored, however, the 8080 emulation mode value of the mode flag (MD) is restored, the CPU is set in emulation mode, and all subsequent instructions are interpreted and executed as μPD8080AF instructions.

RETI is also used to return from an interrupt procedure initiated by an NMI or INT interrupt in the emulation mode.

Floating Point Operation Chip Instructions

FPO1 fp-op, mem

FPO2 fp-op, mem

These instructions are used for the external floating point processor. The floating point operation is passed to the floating point processor when the CPU fetches one of these instructions. From this point the CPU performs only the necessary auxiliary processing (effective address calculation, generation of physical addresses, and start-up of the memory read cycle).

The floating point processor always monitors the instructions fetched by the CPU. When it interprets one as an instruction to itself, it performs the appropriate processing. At this time, the floating point processor chip uses either the address alone or both the address and read data of the memory read cycle executed by the CPU. This difference in the data used depends on which of these instructions is executed.

Note: During the memory read cycle initiated by the CPU for FPO1 or FPO2 execution, the CPU does not accept any read data on the data bus from memory. Although the CPU generates the memory address, the data is used by the floating point processor.

Interrupt Operation

The interrupts used in the μPD70116 can be divided into two types: interrupts generated by external interrupt requests and interrupts generated by software processing. These are the classifications.

External interrupts

(a) NMI input (nonmaskable)
(b) INT input (maskable)

Software processing

As the result of instruction execution

— When a divide error occurs during execution of the DIV or DIVU instruction
— When a memory-boundary-over error is detected by the CHKIND instruction

Conditional break instruction

— When V = 1 during execution of the BRKV instruction

Unconditional break instructions

— 1-byte break instruction: BRK3
— 2-byte break instruction: BRK imm8

Flag processing (Single-step)

— When stack operations are used to set the BRK flag

8080 Emulation mode instructions

— BRKEM imm8
— CALLN imm8

Starting addresses for interrupt processing routines are either determined automatically by a single location of the interrupt vector table or selected each time interrupt processing is entered.

The interrupt vector table is shown in figure 8. The table uses 1K bytes of memory addresses 000H to 3FFH and can store starting address data for a maximum of 256 vectors (4 bytes per vector).

The corresponding interrupt sources for vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. Consequently these vectors cannot be use for general applications.

The BRKEM instruction and CALLN instruction (in the emulation mode) and the INT input are available for general applications for vectors 32 to 255.

A single interrupt vector is made up of 4 bytes (figure 9). The 2 bytes in the low addresses of memory are loaded into PC as the offset, and the high 2 bytes are loaded into PS as the base address. The bytes are combined in reverse order. The lower-order bytes in the vector become the most significant bytes in the PC and PS, and the higher-order bytes become the least significant bytes.

3b

Figure 8. Interrupt Vector Table

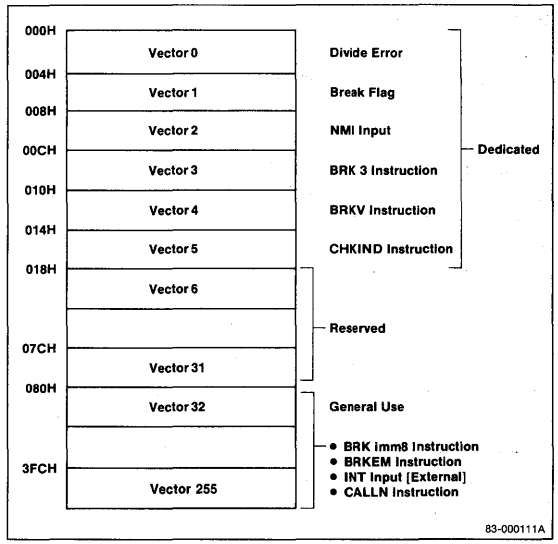
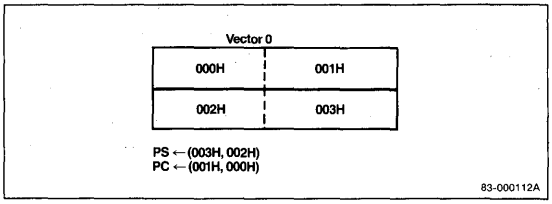


Figure 9. Interrupt Vector 0



Based on this format, the contents of each vector should be initialized at the beginning of the program.

The basic steps to jump to an interrupt processing routine are now shown.

(SP - 1, SP - 2) ← PSW
(SP - 3, SP - 4) ← PS
(SP - 5, SP - 6) ← PC
SP ← SP - 6
IE ← 0, BRK ← 0, MD ← 0
PS ← vector high bytes
PC ← vector low bytes

Standby Function

The μPD70116 has a standby mode to reduce power consumption during program wait states. This mode is set by the HALT instruction in both the native and the emulation mode.

In the standby mode, the internal clock is supplied only to those circuits related to functions required to release this mode and bus hold control functions. As a result, power consumption can be reduced to 1/10 the level of normal operation in either native or emulation mode.

The standby mode is released by inputting a RESET signal or an external interrupt (NMI, INT).

The bus hold function is effective during standby mode. The CPU returns to standby mode when the bus hold request is removed.

During standby mode, all control outputs are disabled and the addres/data bus will be at either high or low levels.

Instruction Set

Symbols

Preceding the instruction set, several tables explain symbols, abbreviations, and codes.

Clocks

In the Clocks column of the instruction set, the numbers cover these operations: instruction decoding, effective address calculation, operand fetch, and instruction execution.

Clock timings assume the instruction has been pre-fetched and is present in the four-byte instruction queue. Otherwise, add four clocks for each byte not present.

For instructions that reference memory operands, the number on the left side of the slash (/) is for byte operands and the number on the right side is for word operands.

For conditional control transfer or branch instructions, the number on the left side of the slash is applicable if the transfer or branch takes place. The number on the right side is applicable if it does not take place.

If a range of numbers is given, the execution time depends on the operands involved.

Symbols

Symbol	Meaning
acc	Accumulator (AW or AL)
disp	Displacement (8 or 16 bits)
dmem	Direct memory address
dst	Destination operand or address
dst-block	Name of block addressed by IY register
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
far_label	Label within a different program segment
far_proc	Procedure within a different program segment
fp_op	Floating point instruction operation
imm	8- or 16-bit immediate operand

Symbols

Symbol	Meaning
imm3/4	3- or 4-bit immediate bit offset
imm8	8-bit immediate operand
imm16	16-bit immediate operand
mem	Memory field (000 to 111); 8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
memptr16	Word containing the destination address within the current segment
memptr32	Double word containing a destination address in another segment
mod	Mode field (00 to 10)
near_label	Label within the current segment
near_proc	Procedure within the current segment
offset	Immediate offset data (16 bits)
pop_value	Number of bytes to discard from the stack
reg	Register field (000 to 111); 8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
regptr	16-bit register containing a destination address within the current segment
regptr16	Register containing a destination address within the current segment
seg	Immediate segment data (16 bits)
short_label	Label between −128 and +127 bytes from the end of the current instruction
sr	Segment register
src	Source operand or address
src-block	Name of block addressed by IX register
src-table	Name of 256-byte translation table
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1 bit)
AC	Auxiliary carry flag
AH	Accumulator (high byte)
AL	Accumulator (low byte)
AND ∧	Logical product
AW	Accumulator (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
BP	Base pointer (16 bits)
BRK	Break flag
BW	BW register (16 bits)
CH	CW register (high byte)

Symbol	Meaning
CL	CW register (low byte)
CW	CW register (16 bits)
CY	Carry flag
DH	DW register (high byte)
DIR	Direction flag
DL	DW register (low byte)
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DW	DW register (16 bits)
IE	Interrupt enable flag
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
MD	Mode flag
OR V	Logical sum
P	Parity flag
PC	Program counter (16 bits)
PS	Program segment register (16 bits)
PSW	Program status word (16 bits)
R	Register set
S	Sign extend operand field S = 0 No sign extension S = 1 Sign extend immediate byte operand
S	Sign flag
SP	Stack pointer (16 bits)
SS	Stack segment register (16 bits)
TA	Temporary register A (16 bits)
TB	Temporary register B (16 bits)
TC	Temporary register C (16 bits)
V	Overflow flag
W	Word/byte field (0 to 1)
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
XOR ⊻	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value
Z	Zero flag
( )	Values in parentheses are memory contents
←	Transfer direction
+	Addition
−	Subtraction
x	Multiplication
÷	Division
%	Modulo

3b

Flag Operations

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to result
u	Undefined
R	Restored to previous state

Memory Addressing Modes

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Register Selection (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Register Selection

sr	Segment Register
00	DS1
01	PS
10	SS
11	DS0

Instruction Set

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer Instructions																									
MOV	reg, reg	1	0	0	0	1	0	1	W	1	1		reg			reg		2	2						
	mem, reg	1	0	0	0	1	0	0	W	mod			reg			mem		9/13	2-4						
	reg, mem	1	0	0	0	1	0	1	W	mod			reg			mem		11/15	2-4						
	mem, imm	1	1	0	0	0	1	1	W	mod		0	0	0		mem		11/15	3-6						
	reg, imm	1	0	1	1	W		reg										4	2-3						
	acc, dmem	1	0	1	0	0	0	0	W									10/14	3						
	dmem, acc	1	0	1	0	0	0	1	W									9/13	3						
	sr, reg16	1	0	0	0	1	1	1	0	1	1	0	sr			reg		2	2						
	sr, mem16	1	0	0	0	1	1	1	0	mod		0	sr			mem		11/15	2-4						
	reg16, sr	1	0	0	0	1	1	0	0	1	1	0	sr			reg		2	2						
	mem16, sr	1	0	0	0	1	1	0	0	mod		0	sr			mem		10/14	2-4						
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod			reg			mem		18/26	2-4						
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod			reg			mem		18/26	2-4						
	AH, PSW	1	0	0	1	1	1	1	1									2	1						
	PSW, AH	1	0	0	1	1	1	1	0									3	1	x	x		x	x	x
LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod			reg			mem		4	2-4						
TRANS	src_table	1	1	0	1	0	1	1	1									9	1						
XCH	reg, reg	1	0	0	0	0	1	1	W	1	1		reg			reg		3	2						
	mem, reg	1	0	0	0	0	1	1	W	mod			reg			mem		16/24	2-4						
	AW, reg16	1	0	0	1	0		reg										3	1						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Repeat Prefixes																									
REPC		0	1	1	0	0	1	0	1									2	1						
REPNC		0	1	1	0	0	1	0	0									2	1						
REP REPE REPZ		1	1	1	1	0	0	1	1									2	1						
REPNE REPNZ		1	1	1	1	0	0	1	0									2	1						
Block Transfer Instructions																									
MOVBK	dst, src	1	0	1	0	0	1	0	W									11 + 8n	1						
CMPBK	dst, src	1	0	1	0	0	1	1	W									7 + 14n	1	x	x	x	x	x	x
CMPM	dst	1	0	1	0	1	1	1	W									7 + 10n	1	x	x	x	x	x	x
LDM	src	1	0	1	0	1	1	0	W									7 + 9n	1						
STM	dst	1	0	1	0	1	0	1	W									7 + 4n	1						
										n = number of transfers															
I/O Instructions																									
IN	acc, imm8	1	1	1	0	0	1	0	W									9/13	2						
	acc, DW	1	1	1	0	1	1	0	W									8/12	1						
OUT	imm8, acc	1	1	1	0	0	1	1	W									8/12	2						
	DW, acc	1	1	1	0	1	1	1	W									8/12	1						
INM	dst, DW	0	1	1	0	1	1	0	W									9 + 8n	1						
OUTM	DW, src	0	1	1	0	1	1	1	W									9 + 8n	1						
										n = number of transfers															
BCD Instructions																									
ADJBA		0	0	1	1	0	1	1	1									3	1	x	x	u	u	u	u
ADJ4A		0	0	1	0	0	1	1	1									3	1	x	x	u	x	x	x
ADJBS		0	0	1	1	1	1	1	1									7	1	x	x	u	u	u	u
ADJ4S		0	0	1	0	1	1	1	1									7	1	x	x	u	x	x	x
ADD4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	7 + 19n	2	u	x	u	u	u	x
SUB4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	7 + 19n	2	u	x	u	u	u	x
CMP4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	7 + 19n	2	u	x	u	u	u	x
ROL4	reg8	0 1	0 1	0 0	0 0	1 0	1 reg	1	1	0	0	1	0	1	0	0	0	25	3						
	mem8	0 mod	0	0 0	0 0	1 mem	1	1	1	0	0	1	0	1	0	0	0	28	3-5						
ROR4	reg8	0 1	0 1	0 0	0 0	1 0	1 reg	1	1	0	0	1	0	1	0	1	0	29	3						
	mem8	0 mod	0	0 0	0 0	1 mem	1	1	1	0	0	1	0	1	0	1	0	33	3-5						
										n = number of BCD digits divided by 2															

3b

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Type Conversion Instructions																									
CVTBD		1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	15	2	u	u	u	x	x	x
CVTDB		1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	7	2	u	u	u	x	x	x
CVTBW		1	0	0	1	1	0	0	0									2	1						
CVTWL		1	0	0	1	1	0	0	1									4-5	1						
Arithmetic Instructions																									
ADD	reg, reg	0	0	0	0	0	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	0	0	0	0	0	W	mod		reg			mem			16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	0	0	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	0	0	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	0	0	mem			18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	0	0	1	0	W									4	2-3	x	x	x	x	x	x
ADDC	reg, reg	0	0	0	1	0	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	0	0	0	W	mod		reg			mem			16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	0	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	0	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	0	mem			18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	0	1	0	W									4	2-3	x	x	x	x	x	x
SUB	reg, reg	0	0	1	0	1	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	1	0	1	0	0	W	mod		reg			mem			16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	0	1	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	0	1	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	0	1	mem			18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	0	1	1	0	W									4	2-3	x	x	x	x	x	x
SUBC	reg, reg	0	0	0	1	1	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	1	0	0	W	mod		reg			mem			16/24	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	1	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	1	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	1	mem			18/26	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	1	1	0	W									4	2-3	x	x	x	x	x	x
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0	reg			2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	0	mem			16/24	2-4	x		x	x	x	x
	reg16	0	1	0	0	0	reg											2	1	x		x	x	x	x
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1	reg			2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	1	mem			16/24	2-4	x		x	x	x	x
	reg16	0	1	0	0	1	reg											2	1	x		x	x	x	x
MULU	reg	1	1	1	1	0	1	1	W	1	1	1	0	0	reg			21-30	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	0	mem			27-36	2-4	u	x	x	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Arithmetic Instructions (cont)																									
MUL	reg	1	1	1	1	0	1	1	W	1	1	1	0	1	reg			33-47	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	1	mem			39-53	2-4	u	x	x	u	u	u
	reg16,reg16,imm8	0	1	1	0	1	0	1	1	1	1	reg			reg			28-34	3	u	x	x	u	u	u
	reg16,mem16,imm8	0	1	1	0	1	0	1	1	mod		reg			mem			34-40	3-5	u	x	x	u	u	u
	reg16,reg16,imm16	0	1	1	0	1	0	0	1	1	1	reg			reg			36-42	4	u	x	x	u	u	u
	reg16,mem16,imm16	0	1	1	0	1	0	0	1	mod		reg			mem			46-48	4-6	u	x	x	u	u	u
DIVU	reg	1	1	1	1	0	1	1	W	1	1	1	1	0	reg			19-25	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	0	mem			25-31	2-4	u	u	u	u	u	u
DIV	reg	1	1	1	1	0	1	1	W	1	1	1	1	1	reg			29-43	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	1	mem			35-49	2-4	u	u	u	u	u	u
Comparison Instructions																									
CMP	reg, reg	0	0	1	1	1	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	1	1	1	0	0	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	1	1	0	1	W	mod		reg			mem			11/15	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	1	1	reg			4	3-4	x	x	x	x	x	x
	cem, imm	1	0	0	0	0	0	S	W	mod		1	1	1	mem			13/17	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	1	1	1	0	W									4	2-3	x	x	x	x	x	x
Logical Instructions																									
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg			2	2						
	mem	1	1	1	1	0	1	1	W	mod		0	1	0	mem			16/24	2-4						
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1	reg			2	2	x	x	x	x	x	x
	mem	1	1	1	1	0	1	1	W	mod		0	1	1	mem			16/24	2-4	x	x	x	x	x	x
TEST	reg, reg	1	0	0	0	0	1	0	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	1	0	0	0	0	1	0	W	mod		reg			mem			10/14	2-4	u	0	0	x	x	x
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			11/15	3-6	u	0	0	x	x	x
	acc, imm	1	0	1	0	1	0	0	W									4	2-3	u	0	0	x	x	x
AND	reg, reg	0	0	1	0	0	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	1	0	0	0	0	W	mod		reg			mem			16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	0	0	0	1	W	mod		reg			mem			11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem			18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	0	0	1	0	W									4	2-3	u	0	0	x	x	x
OR	reg, reg	0	0	0	0	1	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	0	0	1	0	0	W	mod		reg			mem			16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	0	0	1	0	1	W	mod		reg			mem			11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	0	0	1	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1	mem			18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									4	2-3	u	0	0	x	x	x

3b

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Logical Instructions (cont)																									
XOR	reg, reg	0	0	1	1	0	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	1	1	0	0	0	W	mod			reg			mem		16/24	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	1	0	0	1	W	mod			reg			mem		11/15	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	1	0		reg		4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0		mem		18/26	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	1	0	1	0	W									4	2-3	u	0	0	x	x	x
Bit Manipulation Instructions																									
INS	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	31-117	3						
		1	1		reg			reg																	
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	31-117	4						
		1	1	0	0	0		reg																	
EXT	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	26-55	3						
		1	1		reg			reg																	
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	26-55	4						
		1	1	0	0	0		reg																	
TEST1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	3	3	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	12	3-5	u	0	0	u	u	x
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	4	4	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	13	4-6	u	0	0	u	u	x
		mod		0	0	0		mem																	
SET1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	13	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	5	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	14	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	1									2	1		1				
	DIR	1	1	1	1	1	1	0	1									2	1						
CLR1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	5	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	14	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	6	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	15	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	0									2	1		0				
	DIR	1	1	1	1	1	1	0	0									2	1						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Bit Manipulation Instructions (cont)																									
NOT1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	18	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	5	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	19	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	0	1	0	1									2	1		x				
Shift/Rotate Instructions																									
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0	mem			16/24	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0	reg			7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0	mem			19 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0	reg			7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0	mem			19 + n	3-5	u	x	u	x	x	x
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1	mem			16/24	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg			7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1	mem			19 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg			7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1	mem			19 + n	3-5	u	x	u	x	x	x
										n = number of shifts															
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1	reg			2	2	u	x	0	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1	mem			16/24	2-4	u	x	0	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1	reg			7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1	mem			19 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1	reg			7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1	mem			19 + n	3-5	u	x	u	x	x	x
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	0	mem			16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	0	mem			19 + n	2-4		x	u			
	reg, imm	1	1	0	0	0	0	0	W	1	1	0	0	0	reg			7 + n	3		x	u			
	mem, imm	1	1	0	0	0	0	0	W	mod		0	0	0	mem			19 + n	3-5		x	u			
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	1	mem			16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	1	mem			19 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1	reg			7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	1	mem			19 + n	3-5		x	u			

3b

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions (cont)																									
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0	mem			16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0	mem			19 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0	reg			7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0	mem			19 + n	3-5		x	u			
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1	mem			16/24	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1	mem			19 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1	reg			7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1	mem			19 + n	3-5		x	u			
										n = number of shifts															
Stack Manipulation Instructions																									
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0	mem			18/26	2-4						
	reg16	0	1	0	1	0	reg											8/12	1						
	sr	0	0	0	sr		1	1	0									8/12	1						
	PSW	1	0	0	1	1	1	0	0									8/12	1						
	R	0	1	1	0	0	0	0	0									35/67	1						
	imm	0	1	1	0	1	0	S	0									7-8	2-3						
POP	mem16	1	0	0	0	1	1	1	1	mod		0	0	0	mem			17/25	2-4						
	reg16	0	1	0	1	1	reg											8/12	1						
	sr	0	0	0	sr		1	1	1									8/12	1						
	PSW	1	0	0	1	1	1	0	1									8/12	1	R	R	R	R	R	
	R	0	1	1	0	0	0	0	1									43/75	1						
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0									*	4						
										*imm8 = 0 : 12/16 imm8 ≥ 1 : 19 + 8 (imm8 − 1): even address 23 + 16 (imm8-1): odd address														
DISPOSE		1	1	0	0	1	0	0	1									6/10	1						
Control Transfer Instructions																									
CALL	near_proc	1	1	1	0	1	0	0	0									16/20	3						
	regptr	1	1	1	1	1	1	1	1	1	1	0	1	0	reg			14/18	2						
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0	mem			23/31	2-4						
	far_proc	1	0	0	1	1	0	1	0									21/29	5						
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1	mem			31/47	2-4						
RET		1	1	0	0	0	0	1	1									15/19	1						
	pop_value	1	1	0	0	0	0	1	0									20/24	3						
		1	1	0	0	1	0	1	1									21/29	1						
	pop_value	1	1	0	0	1	0	1	0									24/32	3						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Control Transfer Instructions (cont)																									
BR	near_label	1	1	1	0	1	0	0	1		13	3													
	short_label	1	1	1	0	1	0	1	1		12	2													
	regptr	1	1	1	1	1	1	1	1	1	1	1	0	0		reg		11	2						
	memptr16	1	1	1	1	1	1	1	1	mod		1	0	0		mem		20/24	2-4						
	far_label	1	1	1	0	1	0	1	0									15	5						
	memptr32	1	1	1	1	1	1	1	1	mod		1	0	1		mem		27/35	2-4						
BV	short_label	0	1	1	1	0	0	0	0									14/4	2						
BNV	short_label	0	1	1	1	0	0	0	1									14/4	2						
BC, BL	short_label	0	1	1	1	0	0	1	0									14/4	2						
BNC, BNL	short_label	0	1	1	1	0	0	1	1									14/4	2						
BE, BZ	short_label	0	1	1	1	0	1	0	0									14/4	2						
BNE, BNZ	short_label	0	1	1	1	0	1	0	1									14/4	2						
BNH	short_label	0	1	1	1	0	1	1	0									14/4	2						
BH	short_label	0	1	1	1	0	1	1	1									14/4	2						
BN	short_label	0	1	1	1	1	0	0	0									14/4	2						
BP	short_label	0	1	1	1	1	0	0	1									14/4	2						
BPE	short_label	0	1	1	1	1	0	1	0									14/4	2						
BPO	short_label	0	1	1	1	1	0	1	1									14/4	2						
BLT	short_label	0	1	1	1	1	1	0	0									14/4	2						
BGE	short_label	0	1	1	1	1	1	0	1									14/4	2						
BLE	short_label	0	1	1	1	1	1	1	0									14/4	2						
BGT	short_label	0	1	1	1	1	1	1	1									14/4	2						
DBNZNE	short_label	1	1	1	0	0	0	0	0									14/5	2						
DBNZE	short_label	1	1	1	0	0	0	0	1									14/5	2						
DBNZ	short_label	1	1	1	0	0	0	1	0									13/5	2						
BCWZ	short_label	1	1	1	0	0	0	1	1									13/5	2						
Interrupt Instructions																									
BRK	3	1	1	0	0	1	1	0	0									38/50	1						
	imm8	1	1	0	0	1	1	0	1									38/50	2						
BRKV	imm8	1	1	0	0	1	1	1	0									40/3	1						
RETI		1	1	0	0	1	1	1	1									27/39	1	R	R	R	R	R	R
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod			reg			mem		53-56/18	2-4						
BRKEM	imm8	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	38/50	3						

3b

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
CPU Control Instructions																									
HALT		1	1	1	1	0	1	0	0									2	1						
BUSLOCK		1	1	1	1	0	0	0	0									2	1						
FPO1	fp_op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y	mem			11/15	2-4						
FPO2	fp_op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y	mem			11/15	2-4						
POLL		1	0	0	1	1	0	1	1									2 + 5n	1						
						n = number of times POLL pin is sampled.																			
NOP		1	0	0	1	0	0	0	0									3	1						
DI		1	1	1	1	1	0	1	0									2	1						
EI		1	1	1	1	1	0	1	1									2	1						
8080 Instruction Set Enhancements																									
RETEM		1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	27/39	2	R	R	R	R	R	R
CALLN	imm8	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	38/58	3						

μPD70208 (V40) 8/16-Bit Microprocessor: High-Integration, CMOS

Description

The μPD70208 (V40™) is a high-performance, low-power 16-bit microprocessor integrating a number of commonly used peripherals to dramatically reduce the size of microprocessor systems. The CMOS construction makes the μPD70208 ideal for the design of portable computers, instrumentation, and process control equipment.

The μPD70208 contains a powerful instruction set that is compatible with the μPD70108/μPD70116 (V20®/V30®) and μPD8086/μPD8088 instruction sets. Instruction set support includes a wide range of arithmetic, logical, and control operations as well as bit manipulation, BCD arithmetic, and high-speed block transfer instructions. The μPD70208 can also execute the entire μPD8080AF instruction set using the 8080 emulation mode. Also available is the μPD70216 (V50™), identical to the μPD70208 but with a 16-bit external data bus.

Features

- ☐ Low-power CMOS technology
- ☐ V20/V30 instruction set compatible
- ☐ Minimum instruction execution time: 250 ns (at 8 MHz)
- ☐ Direct addressing of 1M bytes of memory
- ☐ Powerful set of addressing modes
- ☐ Fourteen 16-bit CPU registers
- ☐ On-chip peripherals including
  - — Clock generator
  - — Bus interface
  - — Bus arbitration
  - — Programmable wait state generator
  - — DRAM refresh controller
  - — Three 16-bit timer/counters
  - — Asynchronous serial I/O controller
  - — Eight-input interrupt controller
  - — Four-channel DMA controller
- ☐ Hardware effective address calculation logic
- ☐ Maskable and nonmaskable interrupts
- ☐ IEEE 796 compatible bus interface
- ☐ Low-power standby mode

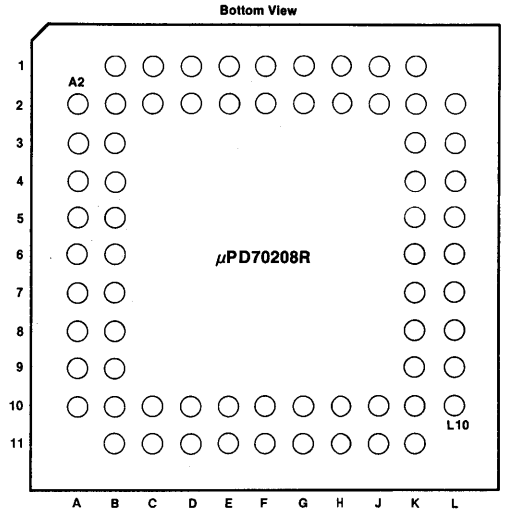
V20 and V30 are registered trademarks of NEC Corporation.
V40 and V50 are trademarks of NEC Corporation.

Ordering Information

Part Number	Max Frequency (MHz)	Package
μPD70208R-8	8	68-pin ceramic PGA
R-10	10	
L-8	8	68-pin PLCC
L-10	10	
GF-8	8	80-pin plastic QFP
GF-10	10	

Pin Configurations

68-Pin Ceramic PGA

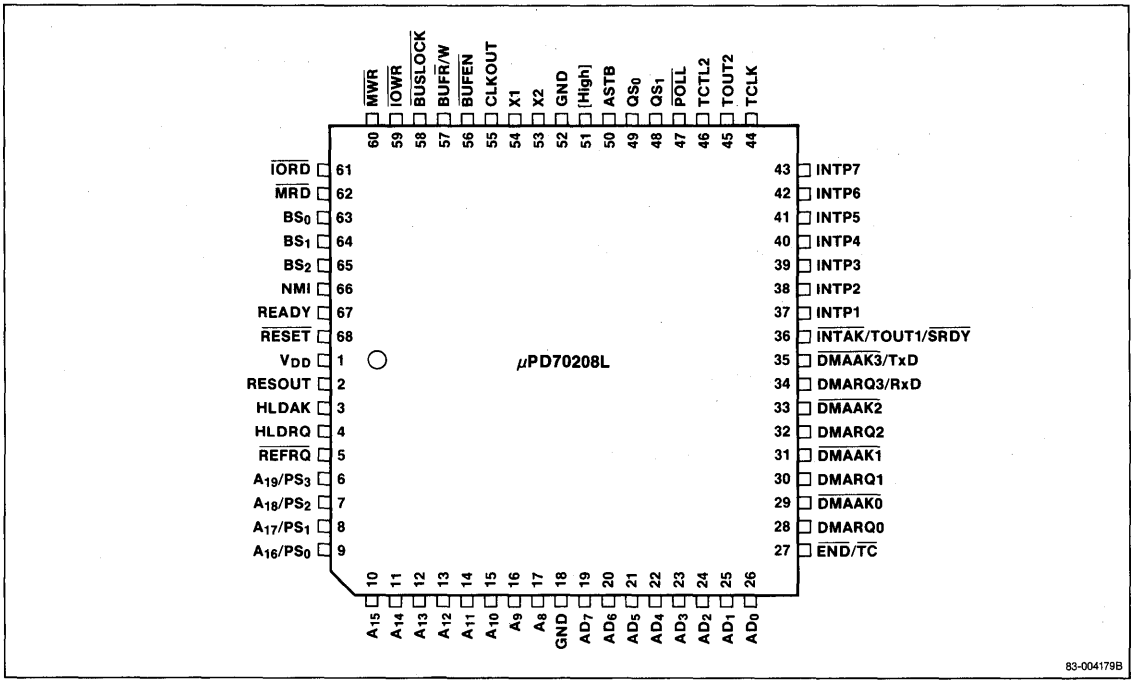


Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
A2	INTP7	B9	DMARQ1	F10	AD_7	K4	NMI
A3	INTP5	B10	DMARQ0	F11	GND	K5	$\overline{RESET}$
A4	INTP3	B11	AD_0	G1	X1	K6	RESOUT
A5	INTP1	C1	TCTL2	G2	CLKOUT	K7	HLDRQ
A6	$\overline{DMAAK3}$/TxD	C2	$\overline{POLL}$	G10	A_8	K8	A_{19}/PS_3
A7	$\overline{DMAAK2}$	C10	AD_1	G11	A_9	K9	A_{17}/PS_1
A8	$\overline{DMAAK1}$	C11	AD_2	H1	$\overline{BUFEN}$	K10	A_{14}
A9	$\overline{DMAAK0}$	D1	QS_1	H2	$BUF\overline{R}/W$	K11	A_{15}
A10	$\overline{END}/\overline{TC}$	D2	QS_0	H10	A_{10}	L2	$\overline{IORD}$
B1	TCLK	D10	AD_3	H11	A_{11}	L3	BS_0
B2	TOUT2	D11	AD_4	J1	$\overline{BUSLOCK}$	L4	BS_2
B3	INTP6	E1	ASTB	J2	$\overline{IOWR}$	L5	READY
B4	INTP4	E2	[High]	J10	A_{12}	L6	V_{DD}
B5	INTP2	E10	AD_5	J11	A_{13}	L7	HLDAK
B6	$\overline{INTAK}$/TOUT1/$\overline{SRDY}$	E11	AD_6	K1	$\overline{MWR}$	L8	$\overline{REFRQ}$
B7	DMARQ3/RxD	F1	GND	K2	$\overline{MRD}$	L9	A_{18}/PS_2
B8	DMARQ2	F2	X2	K3	BS_1	L10	A_{16}/PS_0

83-002716B

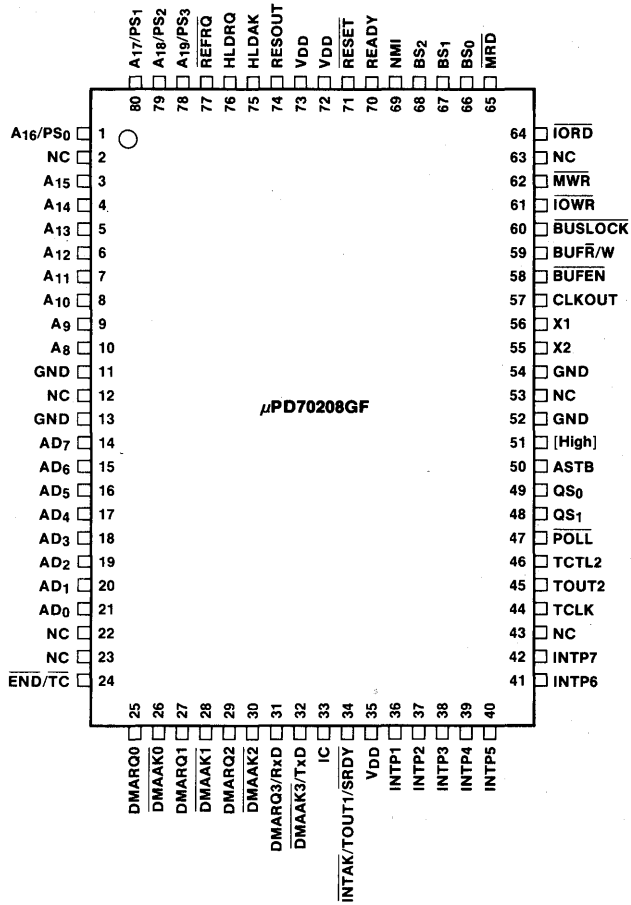
Pin Configurations (cont)

68-Pin Plastic Leaded Chip Carrier (PLCC)



Pin Configurations (cont)

80-Pin Plastic QFP



3c

Pin Identification

Symbol	Function
A_{19}-A_{16}/PS_3-PS_0	Multiplexed address/processor status outputs
A_{15}-A_8	Address bus outputs
AD_7-AD_0	Multiplexed address/data bus
ASTB	Address strobe output
BS_2-BS_0	Bus status outputs
$\overline{BUFEN}$	Data bus transceiver enable output
BUF$\overline{R}$/W	Data bus transceiver direction output
$\overline{BUSLOCK}$	Buslock output
CLKOUT	System clock output
$\overline{DMAAK0}$	DMA channel 0 acknowledge output
$\overline{DMAAK1}$	DMA channel 1 acknowledge output
$\overline{DMAAK2}$	DMA channel 2 acknowledge output
$\overline{DMAAK3}$/TxD	DMA channel 3 acknowledge output/Serial transmit data output
DMARQ0	DMA channel 0 request input
DMARQ1	DMA channel 1 request input
DMARQ2	DMA channel 2 request input
DMARQ3/RxD	DMA channel 3 request input/Serial receive data input
$\overline{END}$/$\overline{TC}$	End input/Terminal count output
GND	Ground
High	High-level output except during hold acknowledge when it is placed in the high-impedance state
HLDAK	Hold acknowledge output
HLDRQ	Hold request input
IC	Internal connection; leave unconnected
$\overline{INTAK}$/TOUT1/$\overline{SRDY}$	Interrupt acknowledge output/Timer/counter 1 output/Serial ready output

Symbol	Function
INTP1-INTP7	Interrupt request inputs
$\overline{IORD}$	I/O read strobe output
$\overline{IOWR}$	I/O write strobe output
$\overline{MRD}$	Memory read strobe output
$\overline{MWR}$	Memory write strobe output
NC	No connection
NMI	Nonmaskable interrupt input
$\overline{POLL}$	Poll input
QS_1-QS_0	CPU queue status outputs
READY	Ready input
$\overline{REFRQ}$	Refresh request output
$\overline{RESET}$	Reset input
RESOUT	Synchronized reset output
TCLK	Timer/counter external clock input
TCTL2	Timer/counter 2 control input
TOUT2	Timer/counter 2 output
V_{DD}	+5 V power supply input
X1, X2	Crystal/external clock inputs

Pin Functions

A_{19}-A_{16}/PS_3-PS_0 [Address/Status Bus]

These three-state output pins contain the upper 4 bits of the 20-bit address during T1 and processor status information during the T2, T3, TW, and T4 states of a bus cycle. During T1 of a memory read or write cycle, these pins contain the upper 4 bits of the 20-bit address. These pins are forced low during T1 of an I/O bus cycle.

Processor status is output during T2, T3, TW, and T4 of both memory and I/O bus cycles. PS_3 is zero during any CPU native mode bus cycle. During any DMA, refresh, or 8080 emulation mode bus cycle, PS_3 outputs a high level. PS_2 outputs the contents of the interrupt enable (IE) flag in the CPU PSW register. PS_1 and PS_0 indicate the segment register used to form the physical address of a CPU bus cycle as follows:

PS_1	PS_0	Segment
0	0	Data segment 1 (DS1)
0	1	Stack segment (SS)
1	0	Program segment (PS)
1	1	Data segment 0 (DS0)

These pins are in the high-impedance state during hold acknowledge.

A_{15}-A_8 [Address Bus]

These three-state pins form the middle byte of the active-high address bus. During any CPU, DMA, or refresh bus cycle, A_{15}-A_8 output the middle 8 bits of the 20-bit memory or I/O address. The A_{15}-A_8 pins enter the high-impedance state during hold acknowledge or an internal interrupt acknowledge bus cycle. During a slave interrupt acknowledge bus cycle, A_{10}-A_8 contain the address of the selected slave interrupt controller.

AD_7-AD_0 [Address/Data Bus]

These three-state pins form the active-high, time-multiplexed address/data bus. During T1 of a bus cycle, AD_7-AD_0 output the lower 8 bits of the 20-bit memory or I/O address. During the T2, T3, TW, and T4 states, AD_7-AD_0 form the 8-bit bidirectional data bus.

The AD_7-AD_0 pins enter the high-impedance state during hold acknowledge or internal interrupt acknowledge bus cycles or while $\overline{RESET}$ is asserted.

ASTB [Address Strobe]

This active-high output is used to latch the address from the multiplexed address bus in an external address latch during T1 of a bus cycle. ASTB is held at a low level during hold acknowledge.

3c

BS_2-BS_0 [Bus Status]

Outputs BS_2-BS_0 indicate the type of bus cycle being performed as follows. BS_2-BS_0 become active during the state preceding T1 and return to the passive state during the bus state preceding T4.

BS_2	BS_1	BS_0	Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt (Note 1)
1	0	0	Instruction fetch
1	0	1	Memory read (Note 2)
1	1	0	Memory write (Note 3)
1	1	1	Passive state

Note:

(1) BS_2-BS_0 in a halt bus cycle returns to the passive state one clock earlier than normal CPU bus cycles.

(2) Memory read bus cycles include CPU, DMA read, DMA verify, and refresh bus cycles.

(3) Memory write bus cycles include CPU and DMA write bus cycles.

BS_2-BS_0 are three-state outputs and are high impedance during hold acknowledge.

$\overline{\text{BUFEN}}$ [Buffer Enable]

$\overline{\text{BUFEN}}$ is an active-low output for enabling an external data bus transceiver during a bus cycle. $\overline{\text{BUFEN}}$ is asserted during T2 through T3 of a read cycle, T2 through T3 of a slave interrupt acknowledge cycle, and T1 through T4 of a write cycle. $\overline{\text{BUFEN}}$ is not asserted when the bus cycle corresponds to an internal peripheral, DMA, refresh, or internal interrupt acknowledge cycle. $\overline{\text{BUFEN}}$ enters the high-impedance state during hold acknowledge.

BUF$\overline{\text{R}}$/W [Buffer Read/Write]

BUF$\overline{\text{R}}$/W is a three-state, active-low output used to control the direction of an external data bus transceiver during CPU bus cycles. A high level indicates the μPD70208 will perform a write cycle and a low level indicates a read cycle. BUF$\overline{\text{R}}$/W enters the high-impedance state during hold acknowledge.

$\overline{\text{BUSLOCK}}$

This active-low output provides a means for the CPU to indicate to an external bus arbiter that the bus cycles of the next instruction are to be kept contiguous. $\overline{\text{BUSLOCK}}$ is asserted for the duration of the instruction following the $\overline{\text{BUSLOCK}}$ prefix. $\overline{\text{BUSLOCK}}$ is also asserted during interrupt acknowledge cycles and enters the high-impedance state during hold acknowledge. While $\overline{\text{BUSLOCK}}$ is asserted, DMAU, RCU, and external bus requests are ignored.

CLKOUT

CLKOUT is a buffered clock output used as a reference for all timing. CLKOUT has a 50-percent duty cycle at half the frequency of the input clock source.

$\overline{\text{DMAAK0}}$-$\overline{\text{DMAAK2}}$ [DMA Acknowledge]

This set of outputs contains the DMA acknowledge signals for channels 0-2 from the internal DMA controller and indicate that the peripheral can perform the requested transfer.

$\overline{\text{DMAAK3}}$/TxD [DMA Acknowledge 3]/[Serial Transmit Data]

Two output signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- $\overline{\text{DMAAK3}}$ is an active-low output and enables an external DMA peripheral to perform the requested DMA transfer for channel 3.
- TxD is the serial data output from the serial control unit.

DMARQ0-DMARQ2 [DMA Request]

These synchronized inputs are used by external peripherals to request DMA service for channels 0-2 from the internal DMA controller.

DMARQ3/RxD [DMA Request 3]/[Serial Receive Data]

Two input signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- DMARQ3 is used by an external peripheral to request a DMA transfer cycle for channel 3.
- RxD is the serial data input to the serial control unit.

$\overline{\text{END}}$/$\overline{\text{TC}}$ [End/Terminal Count]

This active-low bidirectional pin controls the termination of a DMA service. Assertion of $\overline{\text{END}}$ by external hardware during DMA service causes the service to terminate. When a DMA channel reaches its terminal count, the DMAU asserts $\overline{\text{TC}}$, indicating the programmed operation has completed.

$\overline{\text{END}}$/$\overline{\text{TC}}$ is an open-drain I/O pin, and requires an external 2.2-kΩ pull-up resistor.

HLDAK [Hold Acknowledge]

When an external bus requester has become the highest priority requester, the internal bus arbiter will assert the HLDAK output indicating the address, data, and control buses have entered a high-impedance state and are available for use by the external bus master.

Should the internal DMAU or RCU (demand mode) request the bus, the bus arbiter will drive HLDAK low. When this occurs, the external bus master should complete the current bus cycle and negate the HLDRQ signal. This allows the bus arbiter to reassign the bus to the higher priority requester.

If a higher priority internal bus master subsequently requests the bus, the high-level width of HLDAK is guaranteed to be a minimum of one CLKOUT period.

HLDRQ [Hold Request]

This active-high signal is asserted by an external bus master requesting to use the local address, data, and control buses. The HLDRQ input is used by the internal bus arbiter, which gives control of the buses to the highest priority bus requester in the following order.

Bus Master	Priority
RCU	Highest (demand mode)
DMAU	•
HLDRQ	•
CPU	•
RCU	Lowest (normal operation)

$\overline{\text{INTAK}}$/TOUT1/$\overline{\text{SRDY}}$ [Interrupt Acknowledge]/[Timer 1 Output]/[Serial Ready]

Three output signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- $\overline{\text{INTAK}}$ is an interrupt acknowledge signal used to cascade external slave μPD71059 Interrupt Controllers. $\overline{\text{INTAK}}$ is asserted during T2, T3, and TW states of an interrupt acknowledge cycle.
- TOUT1 is the output of timer/counter 1.
- $\overline{\text{SRDY}}$ is an active-low output and indicates that the serial control unit is ready to receive the next character.

INTP1-INTP7 [Peripheral Interrupts]

INTP1-INTP7 accept either rising-edge or high-level triggered asynchronous interrupt requests from external peripherals. These INTP1-INTP7 inputs are internally synchronized and prioritized by the interrupt control unit, which requests the CPU to perform an interrupt acknowledge bus cycle. External interrupt controllers such as the μPD71059 can be cascaded to increase the number of vectored interrupts.

These interrupt inputs cause the CPU to exit both the standby and 8080 emulation modes.

The INTP1-INTP7 inputs contain internal pull-up resistors and may be left unconnected.

3c

$\overline{\text{IORD}}$ [I/O Read]

This three-state pin outputs an active-low I/O read strobe during T2, T3, and TW of an I/O read bus cycle. Both CPU I/O read and DMA write bus cycles assert $\overline{\text{IORD}}$. $\overline{\text{IORD}}$ is not asserted when the bus cycle corresponds to an internal peripheral or register. It enters the high-impedance state during hold acknowledge.

$\overline{\text{IOWR}}$ [I/O Write]

This three-state pin outputs an active-low I/O write strobe during T2, T3, and TW of a CPU I/O write or an extended DMA read cycle and during T3 and TW of a DMA read bus cycle. $\overline{\text{IOWR}}$ is not asserted when the bus cycle corresponds to an internal peripheral or register. It enters the high-impedance state during hold acknowledge.

$\overline{MRD}$ [Memory Read Strobe]

This three-state pin outputs an active-low memory read strobe during T2, T3, and TW of a memory read bus cycle. CPU memory read, DMA read, and refresh bus cycles all assert $\overline{MRD}$. $\overline{MRD}$ enters the high-impedance state during hold acknowledge.

$\overline{MWR}$ [Memory Write Strobe]

This three-state pin outputs an active-low memory write strobe during T2, T3, and TW of a CPU memory write or DMA extended write bus cycle and during T3 and TW of a DMA normal write bus cycle. $\overline{MWR}$ enters the high-impedance state during hold acknowledge.

NMI [Nonmaskable Interrupt]

The NMI pin is a rising-edge-triggered interrupt input that cannot be masked by software. NMI is sampled by CPU logic each clock cycle and when found valid for one or more CLKOUT cycles, the NMI interrupt is accepted. The CPU will process the NMI interrupt immediately after the current instruction finishes execution by fetching the segment and offset of the NMI handler from interrupt vector 2. The NMI interrupt causes the CPU to exit both the standby and 8080 emulation modes. The NMI input takes precedence over the maskable interrupt inputs.

$\overline{POLL}$ [Poll]

The active-low $\overline{POLL}$ input is used to synchronize the operation of external devices with the CPU. During execution of the POLL instruction, the CPU checks the $\overline{POLL}$ input state every five clocks until $\overline{POLL}$ is once again asserted.

QS_1-QS_0 [Queue Status]

The QS_1 and QS_0 outputs maintain instruction synchronization between the μPD70208 CPU and external devices. These outputs are interpreted as follows.

QS_1	QS_0	Instruction Queue Status
0	0	No operation
0	1	First byte of instruction fetched
1	0	Flush queue contents
1	1	Subsequent byte of instruction fetched

Queue status is valid for one clock cycle after the CPU has accessed the instruction queue.

READY [Ready]

This active-high input synchronizes external memory and peripheral devices with the μPD70208. Slow memory and I/O devices can lengthen a bus cycle by negating the READY input and forcing the BIU to insert TW states. READY must be negated prior to the rising edge of CLKOUT during the T2 state or by the last internally generated TW state to guarantee recognition. When READY is once again asserted and recognized by the BIU, the BIU will proceed to the T4 state.

The READY input operates in parallel with the internal μPD70208 wait control unit and can be used to insert more than three wait states into a bus cycle.

$\overline{REFRQ}$ [Refresh Request]

$\overline{REFRQ}$ is an active-low output indicating the current bus cycle is a memory refresh operation. $\overline{REFRQ}$ is used to disable memory address decode logic and refresh dynamic memories. The 9-bit refresh row address is placed on A_8-A_0 during a refresh bus cycle.

$\overline{RESET}$ [Reset]

$\overline{RESET}$ is a Schmitt trigger input used to force the μPD70208 to a known state by resetting the CPU and on-chip peripherals. $\overline{RESET}$ must be asserted for a minimum of four clocks to guarantee recognition. After $\overline{RESET}$ has been released, the CPU will start program execution from address FFFF0H in the native mode.

$\overline{RESET}$ will release the CPU from the low-power standby mode and force it to the native mode.

RESOUT [Reset Output]

This active-high output is available to perform a system-wide reset function. $\overline{RESET}$ is internally synchronized with CLKOUT and output on the RESOUT pin.

TCLK

TCLK is an external clock source for the timer control unit. The three timer/counters can be programmed to operate with either the TCLK input or a prescaled CLKOUT input.

TCTL2

TCTL2 is the control input for timer/counter 2.

TOUT2

TOUT2 is the output of timer/counter 2.

X1, X2 [Clock Inputs]

These pins accept either a parallel resonant, fundamental mode crystal or an external oscillator input with a frequency twice the desired operating frequency.

In the case of an external clock generator, the X2 pin can be either left unconnected or be driven by the complement of the X1 pin clock source.

Pin States

Table 1 lists the output pin states during the Hold, Halt, Reset, and DMA Cascade conditions.

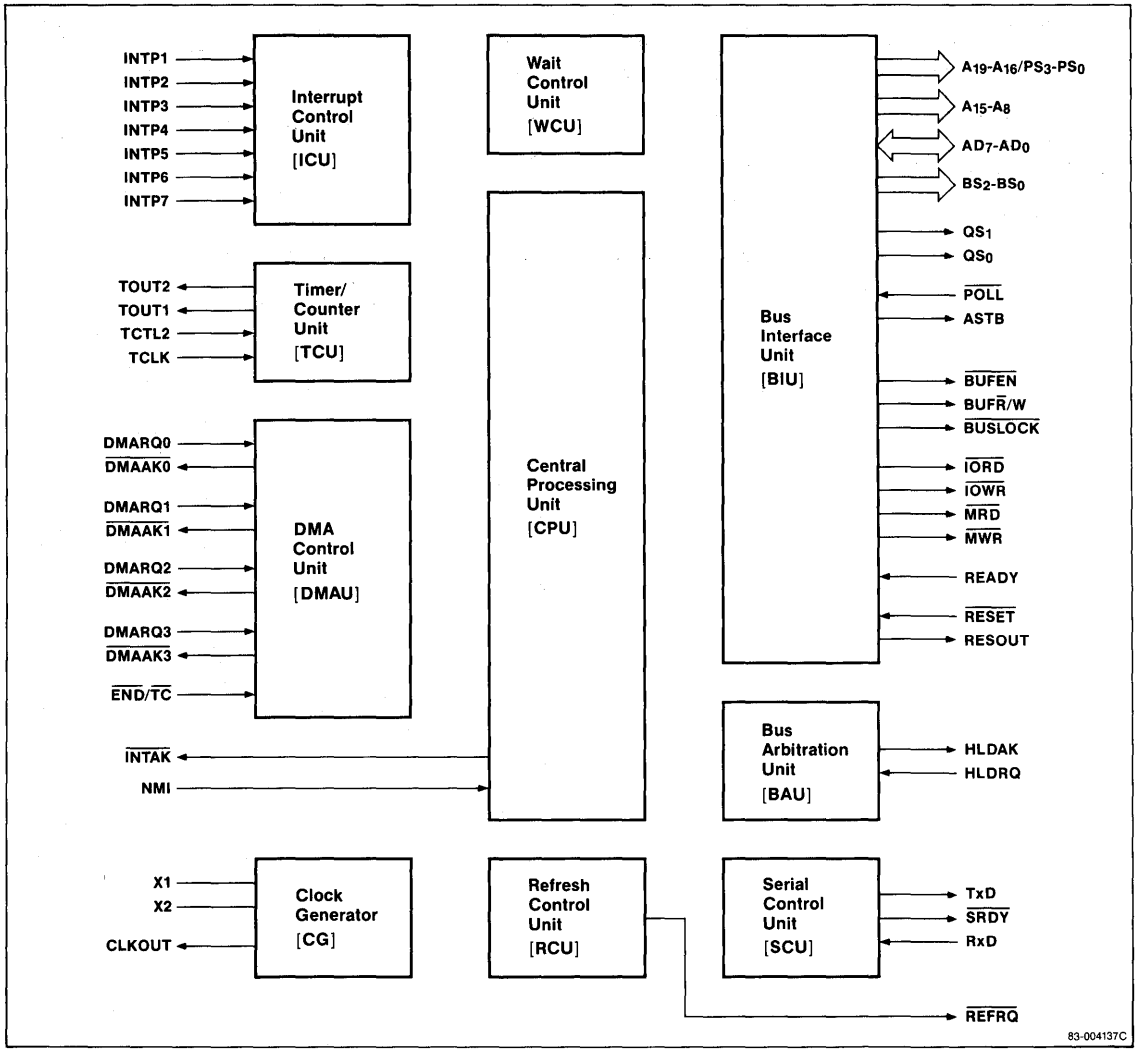
Table 1. Input/Output Pin States

Symbol	Pin Type	Hold	Halt	Reset	DMA Cascade
A_{19}-A_{16}/PS_3-PS_0, A_{15}-A_8	3-state Out	Hi-Z	H/L	H/L	Hi-Z
AD_7-AD_0	3-state I/O	Hi-Z	H/L	Hi-Z	Hi-Z
ASTB	Out	L	L	L	L
$\overline{BUFEN}$	3-state Out	Hi-Z	H	H	Hi-Z
BUF$\overline{R}$/W	3-state Out	Hi-Z	H/L	H	Hi-Z
$\overline{BUSLOCK}$	3-state Out	Hi-Z	H/L	H	Hi-Z
BS_2-BS_0	3-state Out	Hi-Z	H	H	H
CLKOUT	Out	H/L	H/L	H/L	H/L
$\overline{DMAAK0}$-$\overline{DMAAK2}$	Out	H	H/L	H	H/L
$\overline{DMAAK3}$	Out	H	H/L	H	H/L
TxD		H/L		H/L	H/L
$\overline{END}$/$\overline{TC}$	I/O	H	H/L	H	H
HLDAK	Out	H	H/L	L	L
$\overline{INTAK}$	Out	H	H	H	H
TOUT1		H/L	H/L		H/L
$\overline{SRDY}$		H/L	H/L		H/L
$\overline{IORD}$	3-state Out	Hi-Z	H	H	Hi-Z
$\overline{IOWR}$	3-state Out	Hi-Z	H	H	Hi-Z
$\overline{MRD}$	3-state Out	Hi-Z	H	H	Hi-Z
$\overline{MWR}$	3-state Out	Hi-Z	H	H	Hi-Z
QS_1-QS_0	Out	H/L	L	L	H/L
$\overline{REFRQ}$	Out	H	H/L	H	H
RESOUT	Out	L	L	H	L
TOUT2	Out	H/L	H/L	H/L	H/L

H: high level; L: low level; H/L: high or low level; Hi-Z: high impedance.

3c

Block Diagram



Absolute Maximum Ratings

$T_A = +25°C$

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.3 V
CLK input voltage, V_K	−0.5 to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 to V_{DD} + 0.3 V
Operating temperature, T_{OPT}	−10 to +70°C
Storage temperature, T_{STG}	−65 to +150°C

Comment: Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage. The device should be operated within the limits specified under DC and AC Characteristics.

Capacitance

$T_A = +25°C$, $V_{DD} = 0$ V

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input capacitance	C_I		15	pF	$f_C = 1$ MHz; unmeasured pins are returned to 0 V.
Output capacitance	C_O		15	pF	

DC Characteristics

$T_A = -10$ to $+70°C$, $V_{DD} = +5$ V ±10% (8 MHz), $V_{DD} = 5$ V ±5% (10 MHz)

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input voltage, high	V_{IH}	2.2	V_{DD} + 0.3	V	
Input voltage, low	V_{IL}	−0.5	0.8	V	
X1, X2 input voltage, high	V_{KH}	3.9	V_{DD} + 1.0	V	
X1, X2 input voltage, low	V_{KL}	−0.5	0.6	V	
Output voltage, high	V_{OH}	0.7 V_{DD}		V	$I_{OH} = -400$ μA
Output voltage, low	V_{OL}		0.4	V	$I_{OL} = 2.5$ mA
Input leakage current, high	I_{LIH}		10	μA	$V_I = V_{DD}$
Input leakage current, low	I_{LIPL}		−300	μA	$V_I = 0$ V, INTP input pins
	I_{LIL}		−10	μA	$V_I = 0$ V, other input pins
Output leakage current, high	I_{LOH}		10	μA	$V_O = V_{DD}$
Output leakage current, low	I_{LOL}		−10	μA	$V_O = 0$ V
Supply current 8 MHz	I_{DD}		90	mA	Normal mode
			20	mA	Standby mode
10 MHz	I_{DD}		120	mA	Normal mode
			25	mA	Standby mode

3c

AC Characteristics

$T_A = -10$ to $+70°C$; $V_{DD} = 5$ V ±10% (8 MHz), $V_{DD} = 5$ V ± 5% (10 MHz), $C_L = 100$ pF

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
External clock input cycle time	t_{CYX}	62	250	50	250	ns	
External clock pulse width, high	t_{XXH}	20		19		ns	$V_{KH} = 3.0$ V
External clock pulse width, low	t_{XXL}	20		19		ns	$V_{KL} = 1.5$ V
External clock rise time	t_{XR}		10		5	ns	1.5 → 3.0 V
External clock fall time	t_{XF}		10		5	ns	3.0 → 1.5 V
CLKOUT cycle time	t_{CYK}	124	500	100	500	ns	
CLKOUT pulse width, high	t_{KKH}	$0.5\ t_{CYK} - 7$		$0.5\ t_{CYK} - 5$		ns	$V_{KH} = 3.0$ V
CLKOUT pulse width, low	t_{KKL}	$0.5\ t_{CYK} - 7$		$0.5\ t_{CYK} - 5$		ns	$V_{KL} = 1.5$ V
CLKOUT rise time	t_{KR}		7		5	ns	1.5 → 3.0 V
CLKOUT fall time	t_{KF}		7		5	ns	3.0 → 1.5 V
CLKOUT delay time from external clock	t_{DXK}		55		40	ns	
Input rise time (except external clock)	t_{IR}		20		15	ns	0.8 → 2.2 V
Input fall time (except external clock)	t_{IF}		12		10	ns	2.2 → 0.8 V
Output rise time (except CLKOUT)	t_{OR}		20		15	ns	0.8 → 2.2 V
Output fall time (except CLKOUT)	t_{OF}		12		10	ns	2.2 → 0.8 V
$\overline{RESET}$ setup time to CLKOUT↓	t_{SRESK}	25		20		ns	
$\overline{RESET}$ hold time after CLKOUT↓	t_{HKRES}	35		25		ns	
RESOUT delay time from CLKOUT↓	t_{DKRES}	5	60	5	50	ns	
READY inactive setup time to CLKOUT↑	t_{SRYLK}	15		15		ns	
READY inactive hold time after CLKOUT↑	t_{HKRYL}	25		20		ns	
READY active setup time to CLKOUT↑	t_{SRYHK}	15		15		ns	
READY active hold time after CLKOUT↑	t_{HKRYH}	25		20		ns	
NMI, $\overline{POLL}$ setup time to CLKOUT↑	t_{SIK}	15		15		ns	
Data setup time to CLKOUT↓	t_{SDK}	15		15		ns	
Data hold time after CLKOUT↓	t_{HKD}	10		10		ns	
Address delay time from CLKOUT↓	t_{DKA}	10	55	10	50	ns	A_{19}-A_0 $\overline{UBE}$
Address hold time after CLKOUT↓	t_{HKA}	10		10		ns	
I/O recovery time	t_{AI}	$2t_{CYK} - 50$		$2t_{CYK} - 40$		ns	(Note 1)
PS delay time from CLKOUT↓	t_{DKP}	10	60	10	50	ns	
PS float delay time from CLKOUT↑	t_{FKP}	10	60	10	50	ns	
Address setup time to ASTB↓	t_{SAST}	$t_{KKL} - 20$		$t_{KKL} - 30$		ns	
Address float delay time from CLKOUT↓	t_{FKA}	t_{HKA}	60	t_{HKA}	50	ns	
ASTB↑ delay time from CLKOUT↓	t_{DKSTH}		45		40	ns	
ASTB↓ delay time from CLKOUT↑	t_{DKSTL}		50		45	ns	
ASTB pulse width, high	t_{STST}	$t_{KKL} - 10$		$t_{KKL} - 10$		ns	
Address hold time after ASTB↓	t_{HSTA}	$t_{KKH} - 20$		$t_{KKH} - 20$		ns	
Control delay time from CLKOUT	t_{DKCT1}	10	70	10	60	ns	(Note 2)
	t_{DKCT2}	10	60	10	55	ns	(Note 3)

AC Characteristics (cont)

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
$\overline{RD}$↓ delay time from address float	t_{DAFRL}	0		0		ns	(Note 4)
$\overline{RD}$↓ delay time from CLKOUT↓	t_{DKRL}	10	75	10	65	ns	
$\overline{RD}$↑ delay time from CLKOUT↓	t_{DKRH}	10	70	10	60	ns	
$\overline{REFRQ}$↑ delay from $\overline{MRD}$↑	t_{DRQHRH}	$t_{KKL}-30$		$t_{KKL}-30$		ns	(Note 5)
Address delay time from $\overline{RD}$↑	t_{DRHA}	$t_{CYK}-40$		$t_{CYK}-40$		ns	
$\overline{RD}$ pulse width, low	t_{RR}	$2t_{CYK}-50$		$2t_{CYK}-40$		ns	
BUF$\overline{R}$/W delay from $\overline{BUFEN}$↑	t_{DBECT}	$t_{KKL}-20$		$t_{KKL}-20$		ns	Read cycle
	t_{DWCT}	$t_{KKL}-20$		$t_{KKL}-20$		ns	Write cycle
Data output delay time from CLKOUT↓	t_{DKD}	10	60	10	55	ns	
Data float delay time from CLKOUT↓	t_{FKD}	10	60	10	55	ns	
$\overline{WR}$ pulse width, low	t_{WW}	$2t_{CYK}-40$		$2t_{CYK}-40$		ns	(Note 4)
BS↓ delay time from CLKOUT↑	t_{DKBL}	10	60	10	55	ns	
BS↑ delay time from CLKOUT↓	t_{DKBH}	10	60	10	55	ns	
HLDRQ setup time to CLKOUT↑	t_{SHQK}	20		15		ns	
HLDAK delay time from CLKOUT↓	t_{DKHA}	10	70	10	60	ns	
$\overline{DMAAK}$↓ delay time from CLKOUT↑	t_{DKHDA}	10	60	10	55	ns	
$\overline{DMAAK}$↓ delay time from CLKOUT↓	t_{DKLDA}	10	90	10	80	ns	Cascade mode
$\overline{WR}$ pulse width, low (DMA cycle)	t_{WW1}	$2t_{CYK}-40$		$2t_{CYK}-40$		ns	DMA extended write cycle
$\overline{WR}$ pulse width, low (DMA cycle)	t_{WW2}	$t_{CYK}-40$		$t_{CYK}-40$		ns	DMA normal write cycle
$\overline{RD}$↓, $\overline{WR}$↓ delay from DMAAK↓	t_{DDARW}	$t_{KKH}-30$		$t_{KKH}-30$		ns	
$\overline{DMAAK}$↑ delay from $\overline{RD}$↑	t_{DRHDAH}	$t_{KKL}-30$		$t_{KKL}-30$		ns	
$\overline{RD}$↑ delay from $\overline{WR}$↑	t_{DWHRH}	5		5		ns	
$\overline{TC}$ output delay time from CLKOUT↑	t_{DKTCL}		60		55	ns	
$\overline{TC}$ off delay time from CLKOUT↑	t_{DKTCF}		60		55	ns	
$\overline{TC}$ pulse width, low	t_{TCTCL}	$t_{CYK}-15$		$t_{CYK}-15$		ns	
$\overline{TC}$ pullup delay time from CLKOUT↑	t_{DKTCH}		$t_{KKH}+t_{CYK}-10$		$t_{KKH}+t_{CYK}-10$	ns	
$\overline{END}$ setup time to CLKOUT↑	t_{SEDK}	35		30		ns	
$\overline{END}$ pulse width, low	t_{EDEDL}	100		80		ns	
DMARQ setup time to CLKOUT↑	t_{SDQK}	35		30		ns	
INTPn pulse width, low	t_{IPIPL}	100		80		ns	
RxD setup time to SCU internal clock↓	t_{SRX}	1		0.5		μs	
RxD hold time after SCU internal clock↓	t_{HRX}	1		0.5		μs	
$\overline{SRDY}$ delay time from CLKOUT↓	t_{DKSR}		150		100	ns	

Notes:

(1) This is specified to guarantee a read/write recovery time for I/O devices.

(2) Delay from CLKOUT to DMA cycle $\overline{MWR}$/$\overline{IOWR}$ outputs.

(3) Delay from CLKOUT to BUF$\overline{R}$/W, $\overline{BUFEN}$, $\overline{INTAK}$, $\overline{REFRQ}$ outputs and CPU cycle $\overline{MWR}$/$\overline{IOWR}$ outputs.

(4) $\overline{RD}$ represents $\overline{IORD}$ and $\overline{MRD}$. $\overline{WR}$ represents $\overline{IOWR}$ and $\overline{MWR}$.

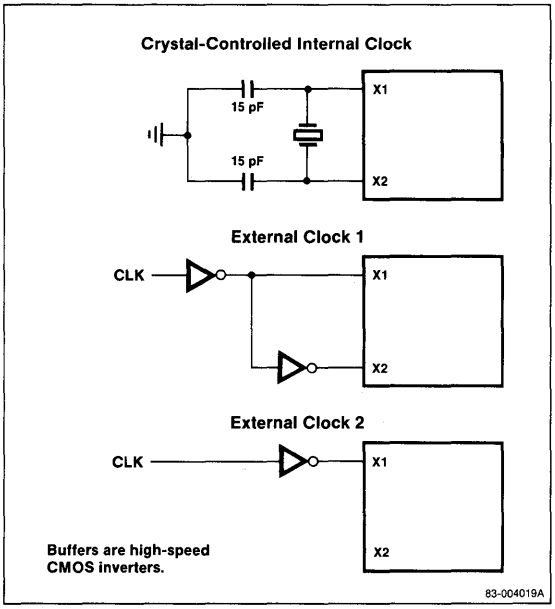
(5) This is specified to guarantee that $\overline{REFRQ}$ ↑ is delayed from $\overline{MRD}$ ↑ at all times.

3c

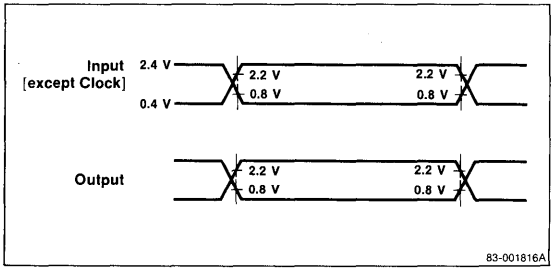
AC Characteristics (cont)

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
TxD delay time from TOUT1↓	t_{DTX}		500		200	ns	
TCTL2 setup time from CLKOUT↓	t_{SGK}	50		40		ns	
TCTL2 setup time to TCLK↑	t_{SGTK}	50		40		ns	
TCTL2 hold time after CLKOUT↓	t_{HKG}	100		80		ns	
TCTL2 hold time after TCLK↑	t_{HTKG}	50		40		ns	
TCTL2 pulse width, high	t_{GGH}	50		40		ns	
TCTL2 pulse width, low	t_{GGL}	50		40		ns	
TOUT output delay time from CLKOUT↓	t_{DKTO}		200		150	ns	
TOUT output delay time from TOUT↓	t_{DTKTO}		150		100	ns	
TOUT output delay time from TCTL2↓	t_{DGTO}		120		90	ns	
TCLK rise time	t_{TKR}		25		25	ns	
TCLK fall time	t_{TKF}		25		25	ns	
TCLK pulse width, high	t_{TKTKH}	50		45		ns	
TCLK pulse width, low	t_{TKTKL}	50		45		ns	
TCLK cycle time	t_{CYTK}	124	DC	100	DC	ns	
RESET pulse width low	t_{RESET1}	50		50		μs	After power on
	t_{RESET2}	4 t_{CYK}		4 t_{CYK}			During operation

Clock Input Configurations



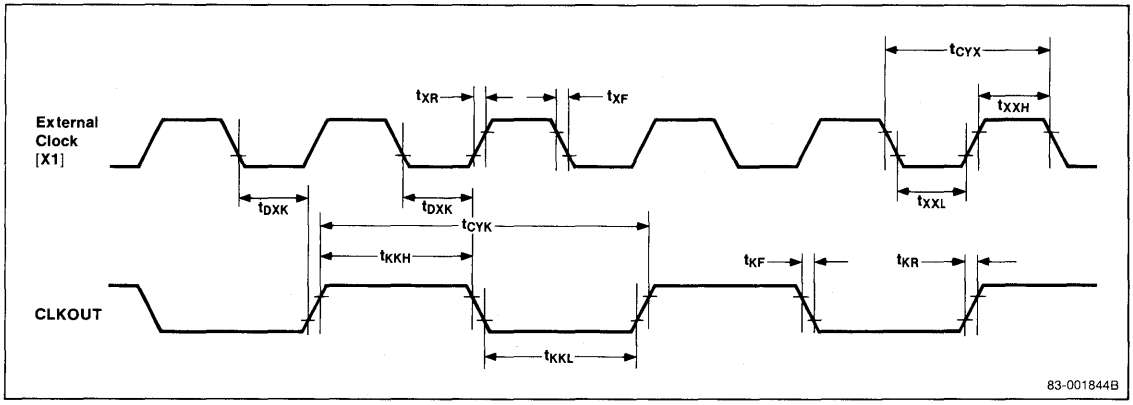
Timing Measurement Points



3c

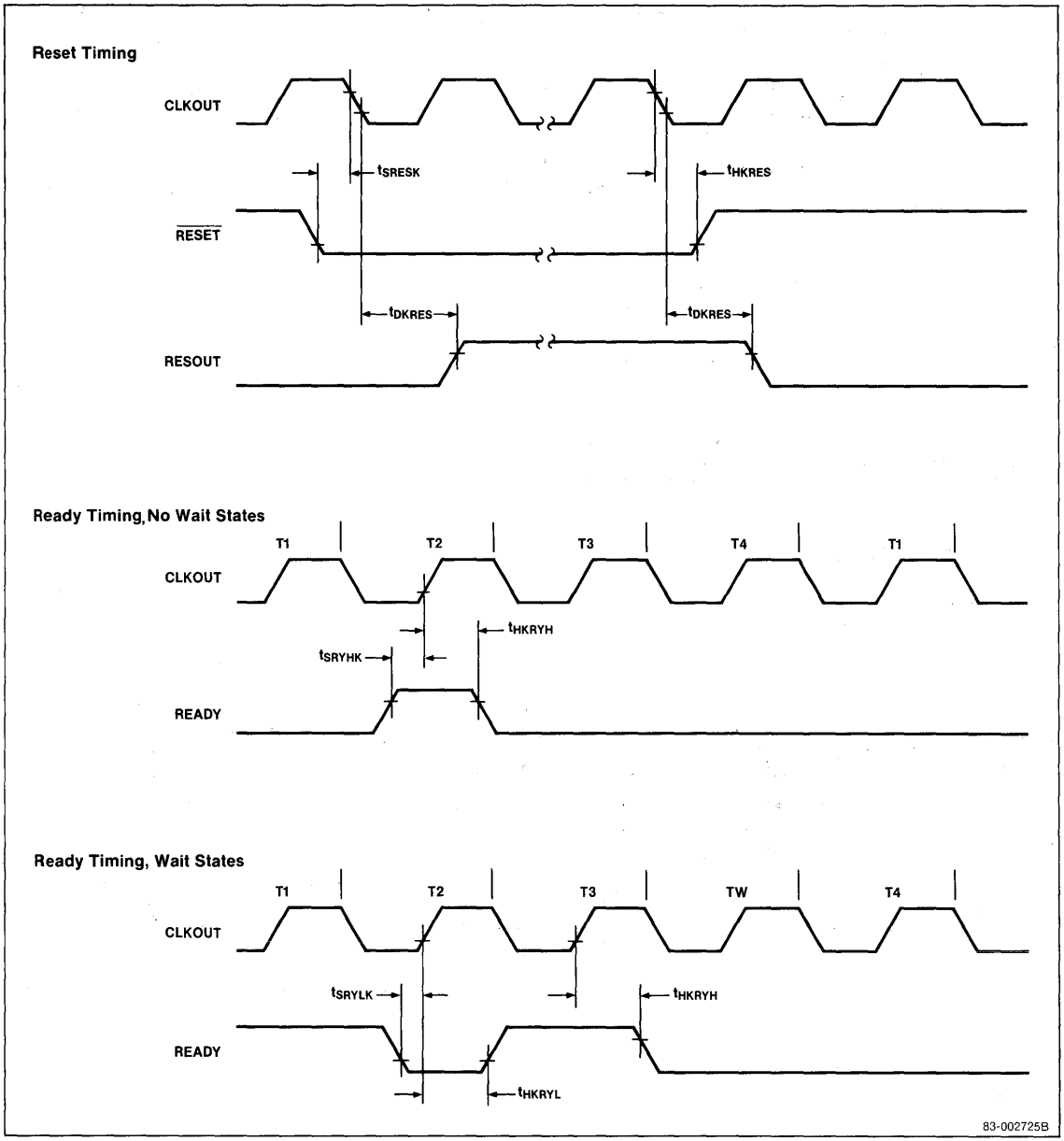
Timing Waveforms

Clock Timing



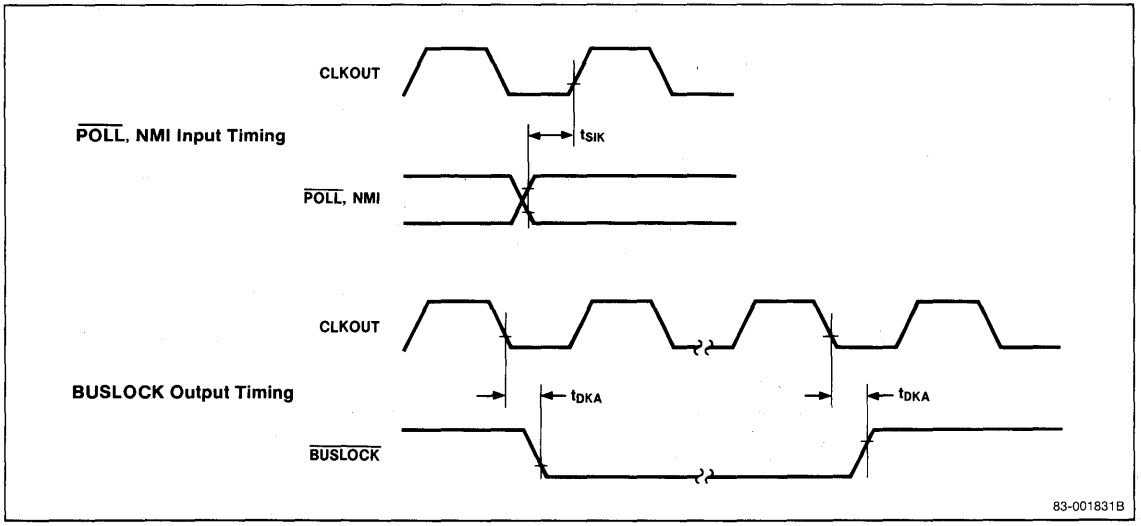
Timing Waveforms (cont)

Reset and Ready Timing

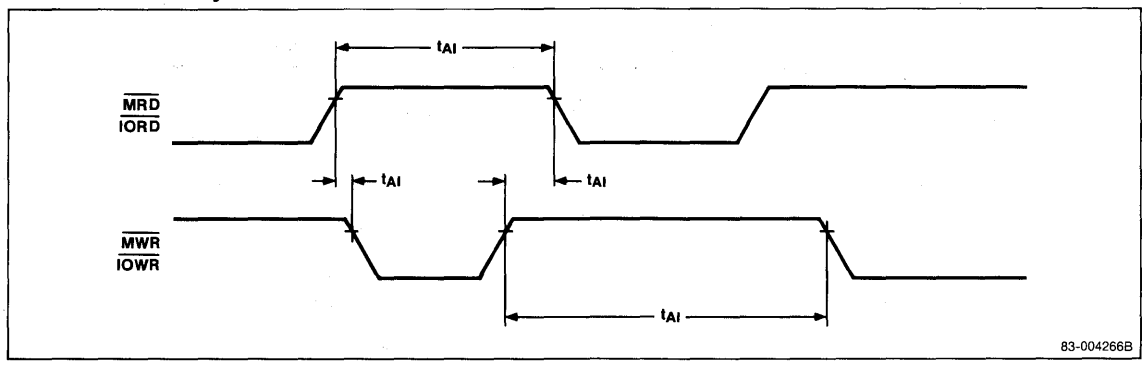


Timing Waveforms (cont)

Poll, NMI, and Buslock Timing



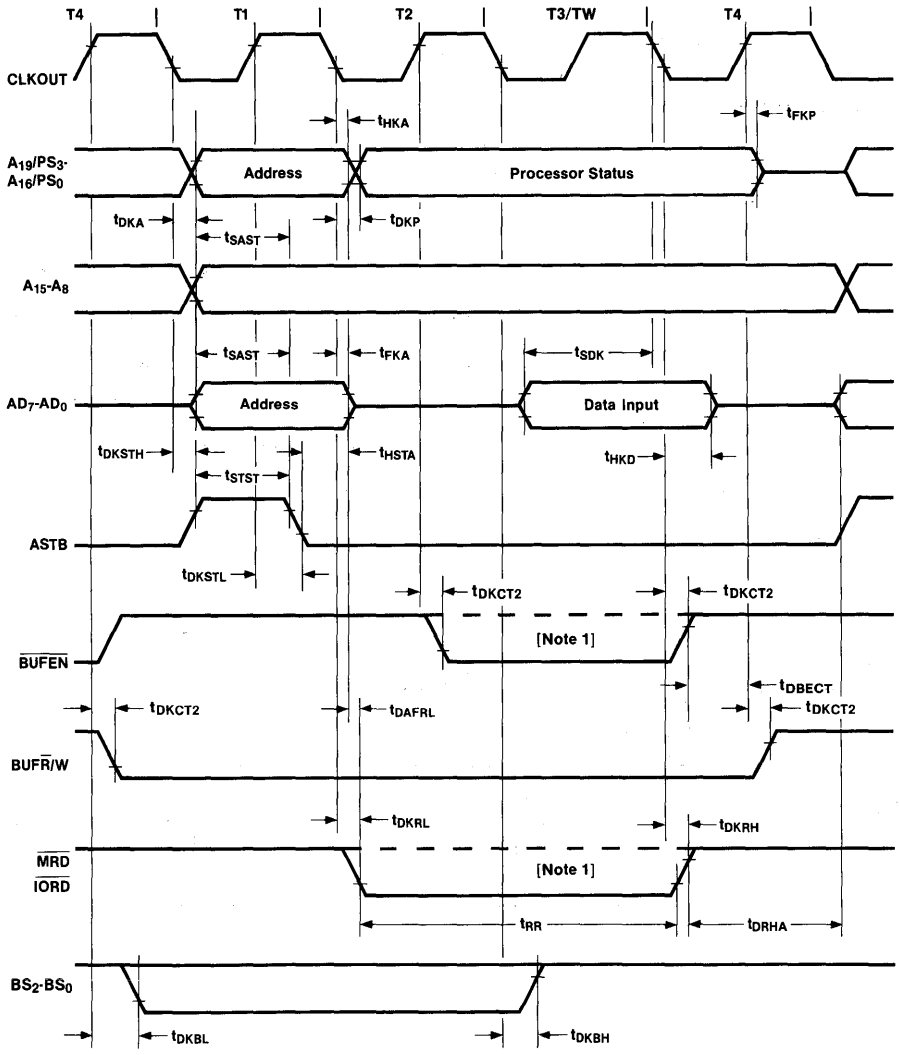
Read/Write Recovery Time



3c

Timing Waveforms (cont)

Read Timing

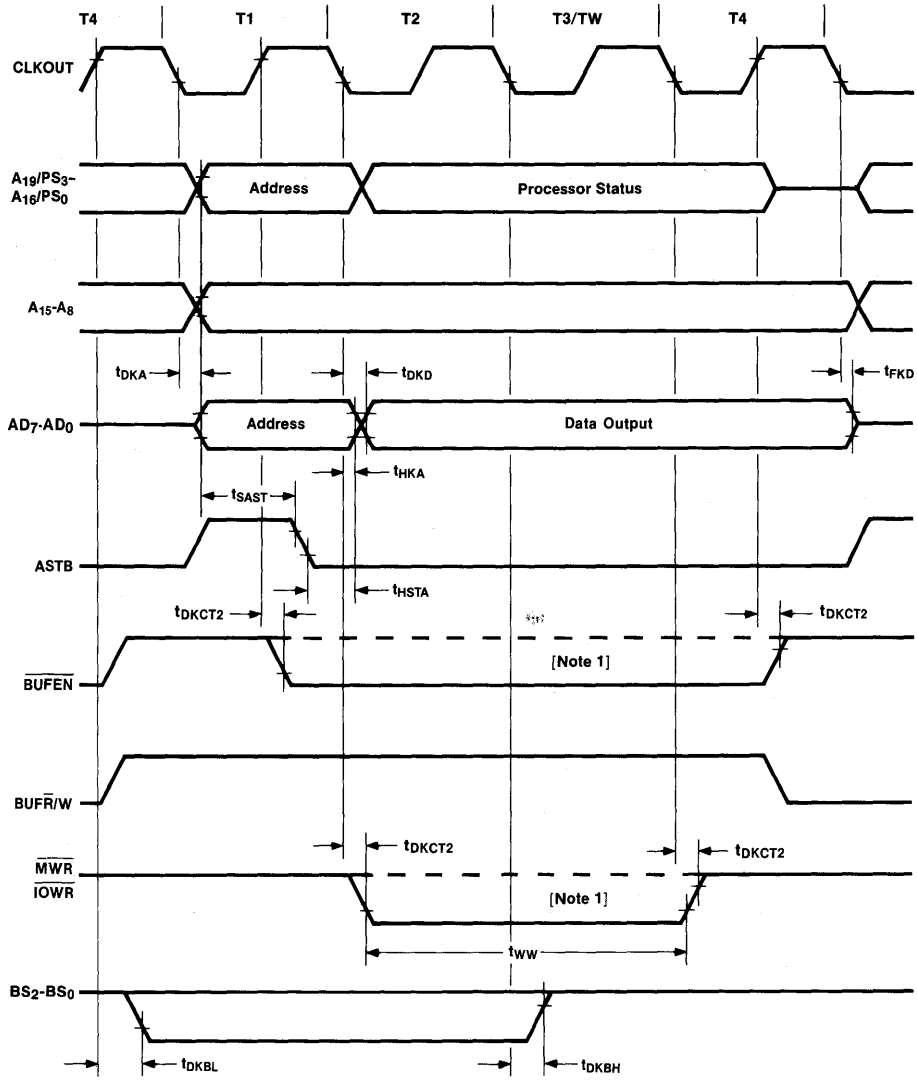


Note:
[1] Except internal I/O accesses.

83-001848 B

Timing Waveforms (cont)

Write Timing



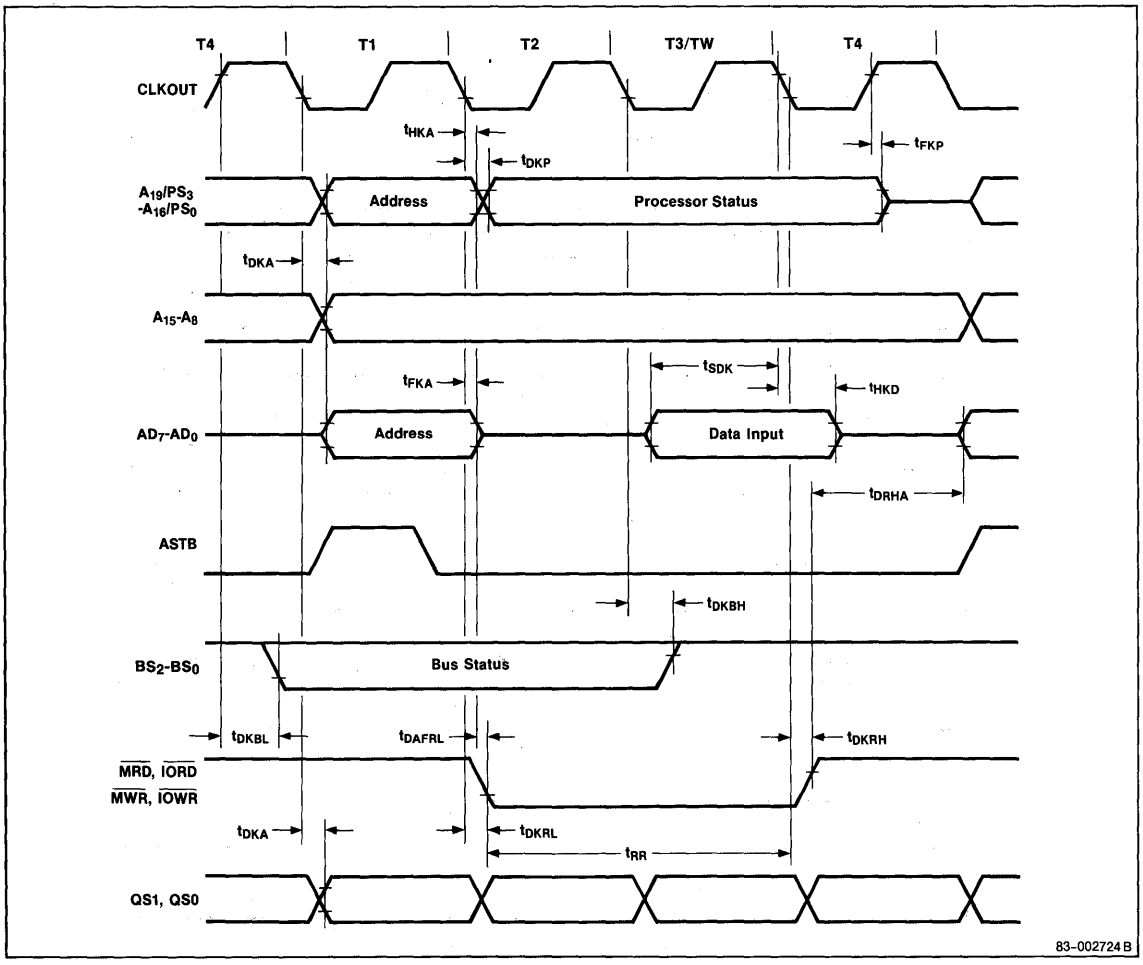
Note:
[1] Except internal I/O accesses.

83-001846 B

3c

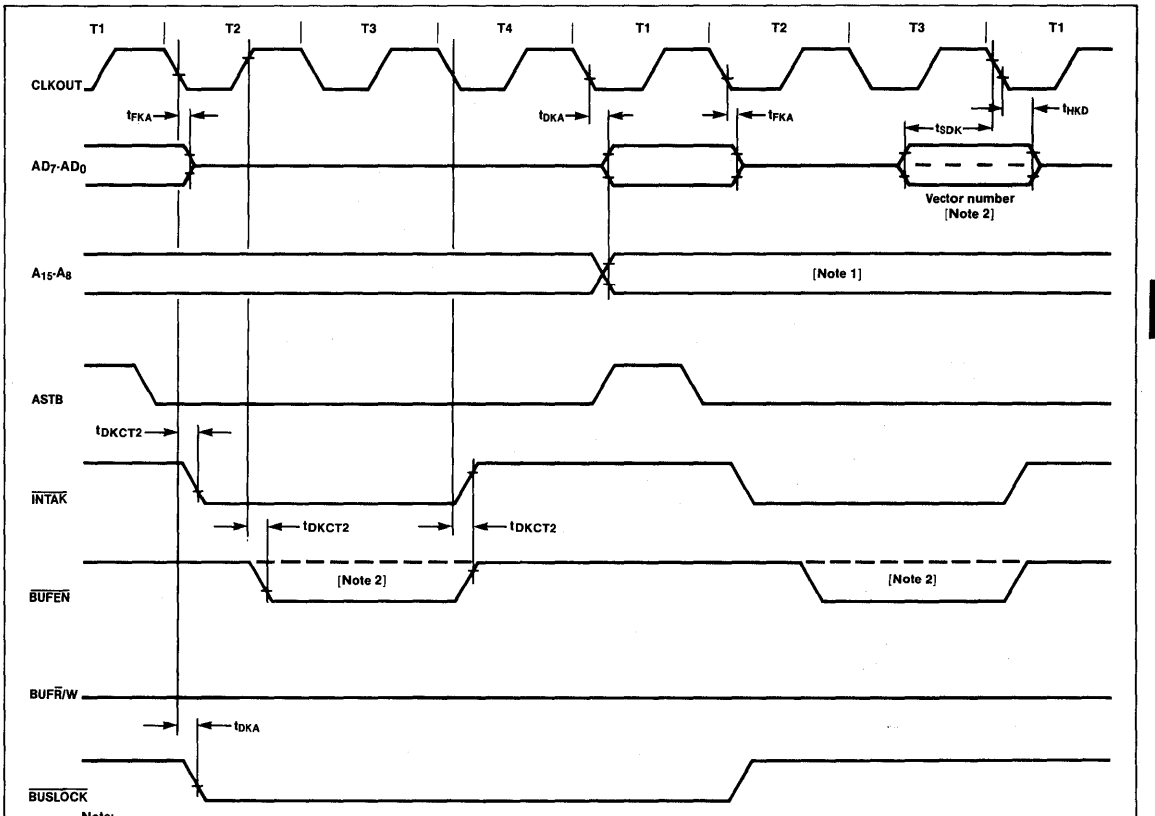
Timing Waveforms (cont)

Status Timing



Timing Waveforms (cont)

Interrupt Acknowledge Timing



Note:

[1] Slave address when the interrupt is from external μPD71059. Undefined when internal ICU interrupt.

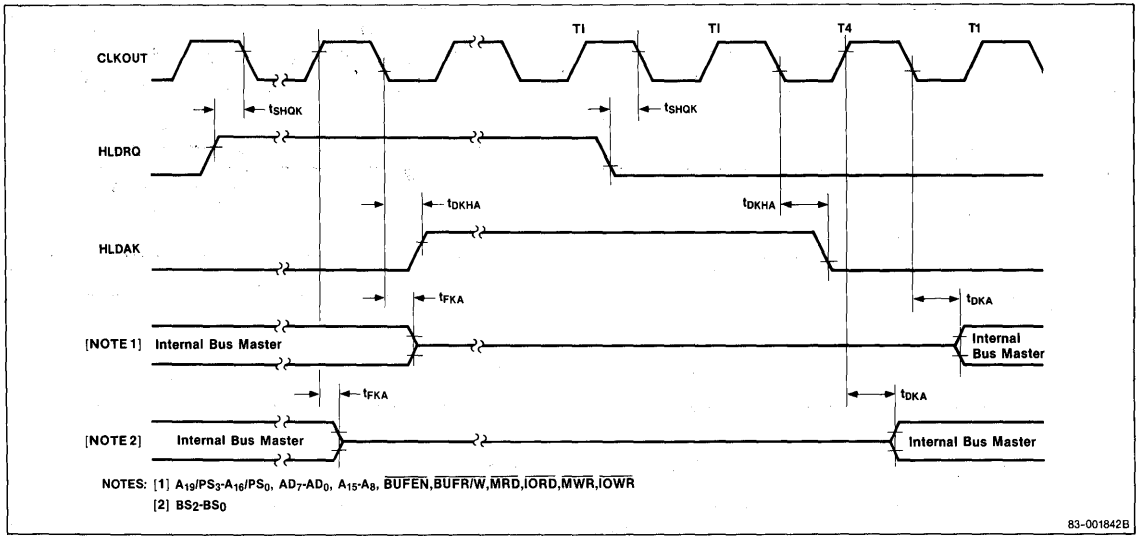
[2] Solid line when interrupt from external μPD71059. Dash line when internal ICU interrupt.

83-001840B

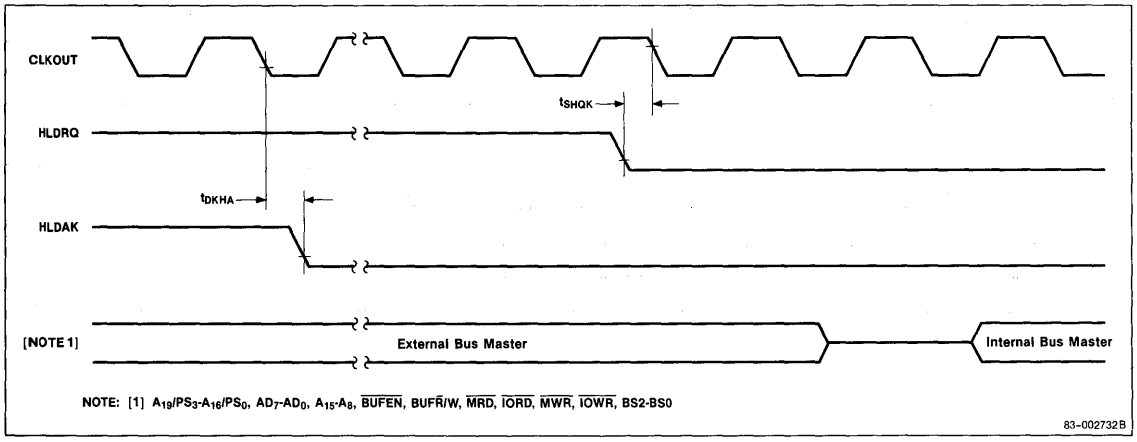
3c

Timing Waveforms (cont)

HLDRQ/HLDAK Timing, Normal Operation

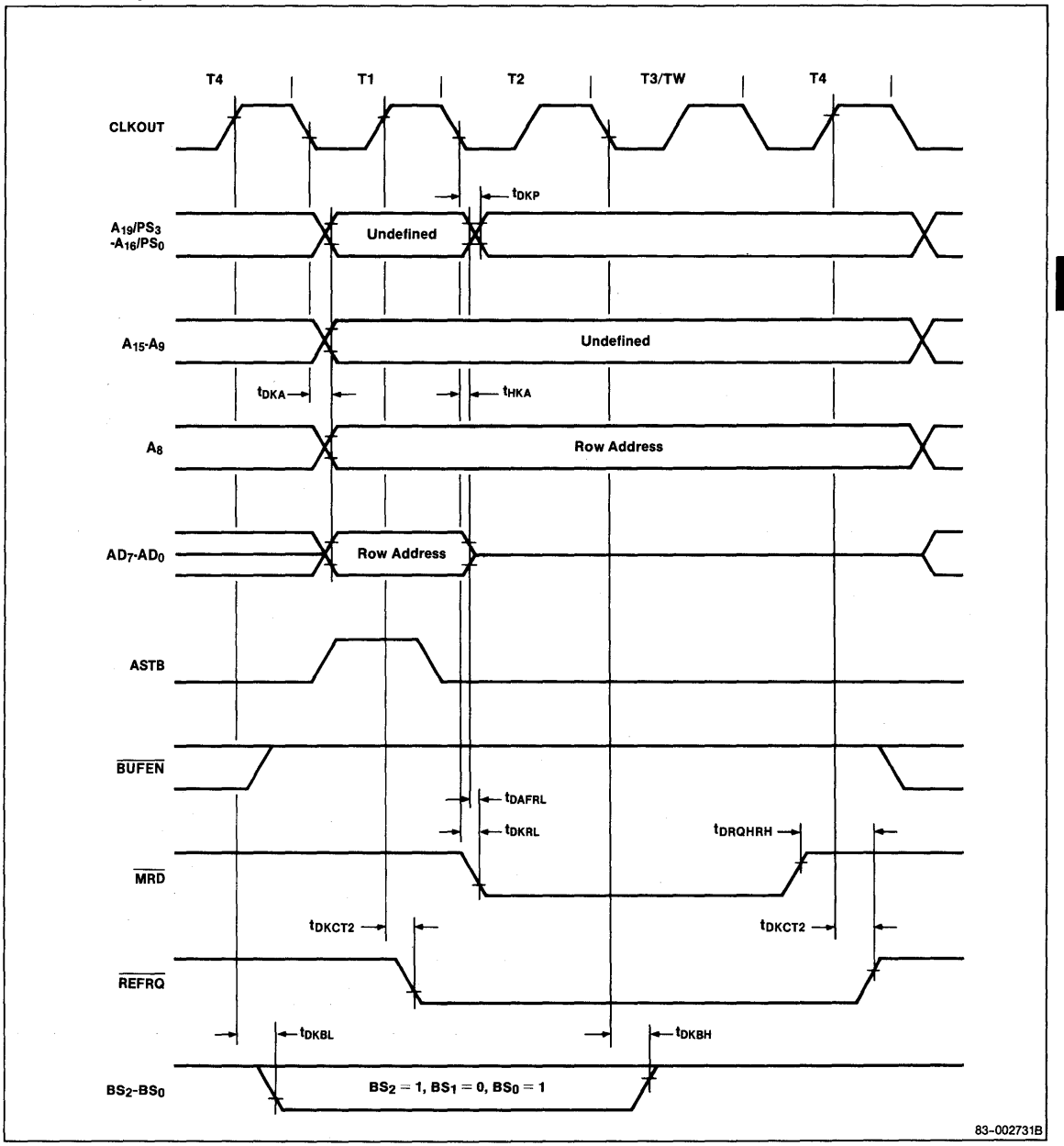


HLDRQ/HLDAK Timing, Bus Wait



Timing Waveforms (cont)

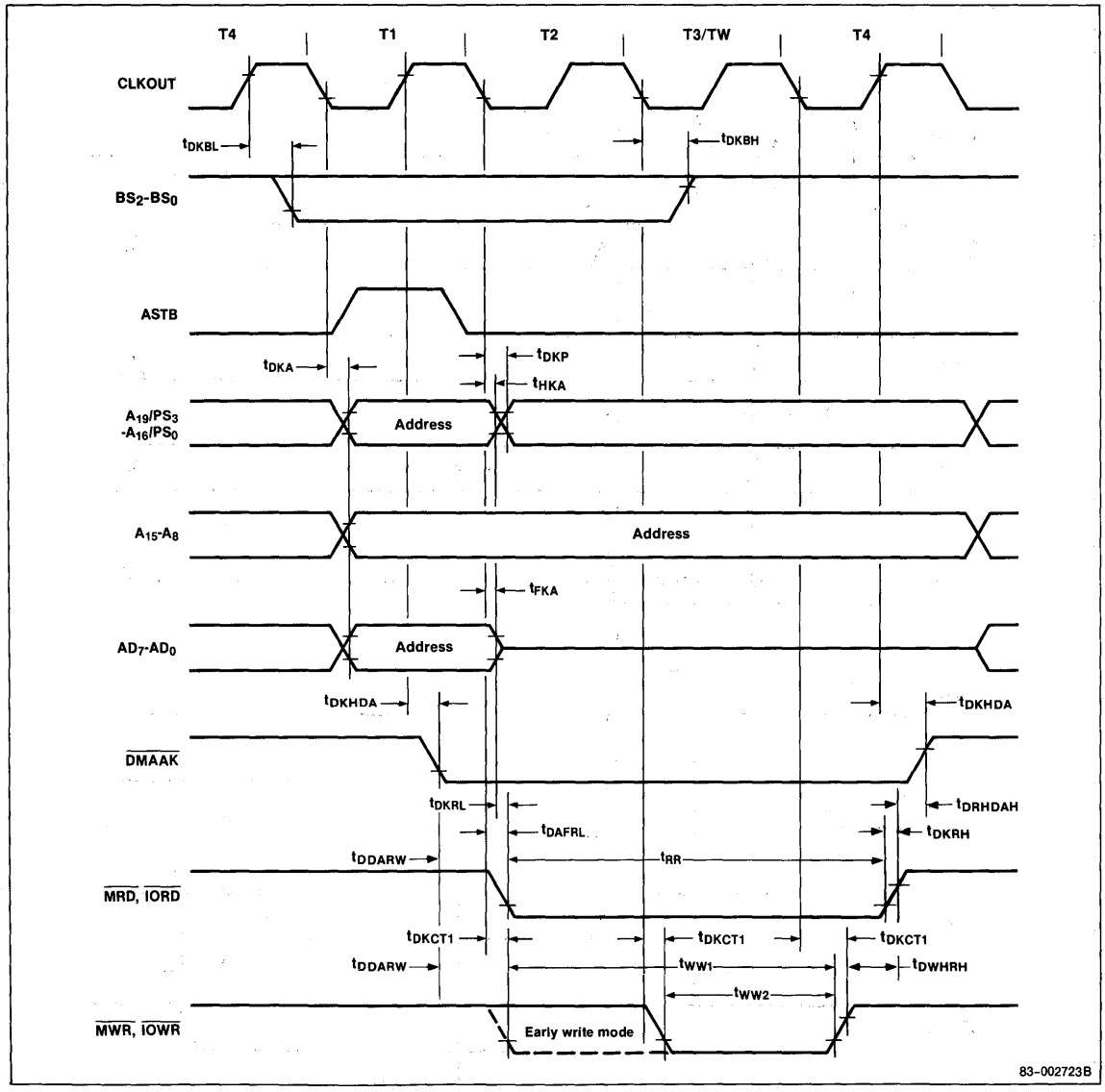
Refresh Timing



3c

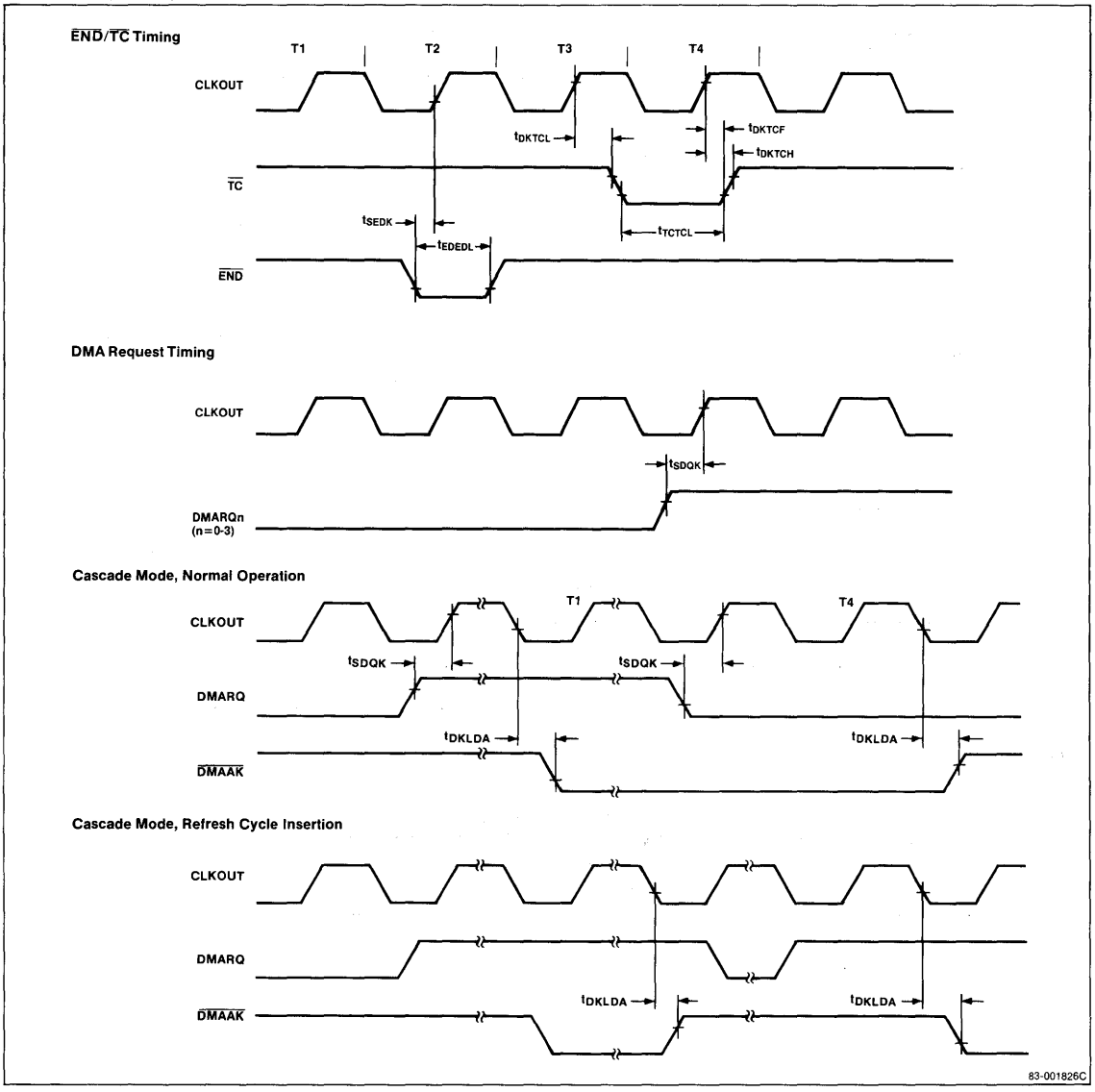
Timing Waveforms (cont)

DMAU, DMA Transfer Timing



Timing Waveforms (cont)

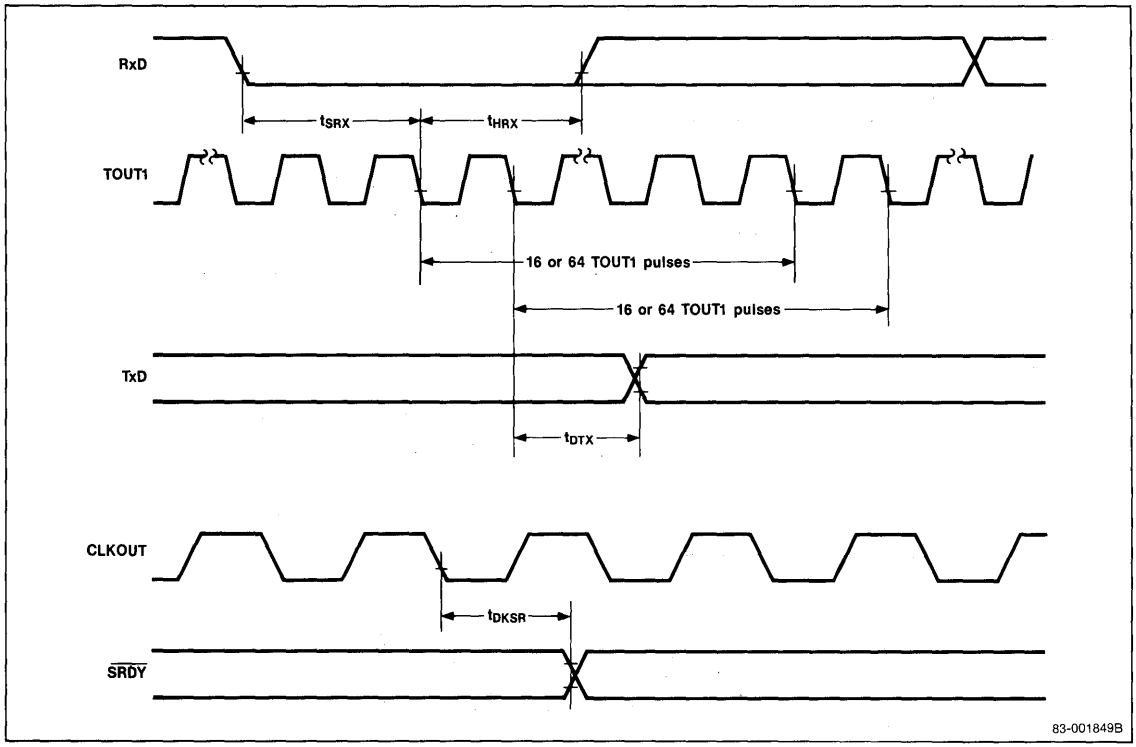
DMA Timing



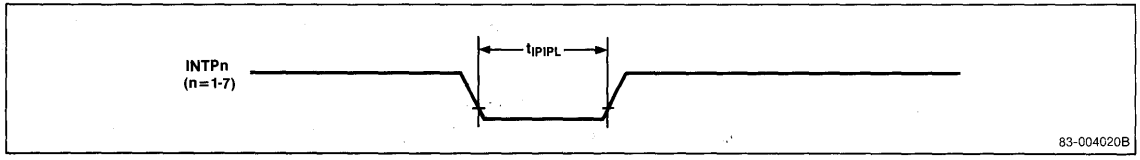
3c

Timing Waveforms (cont)

SCU Timing

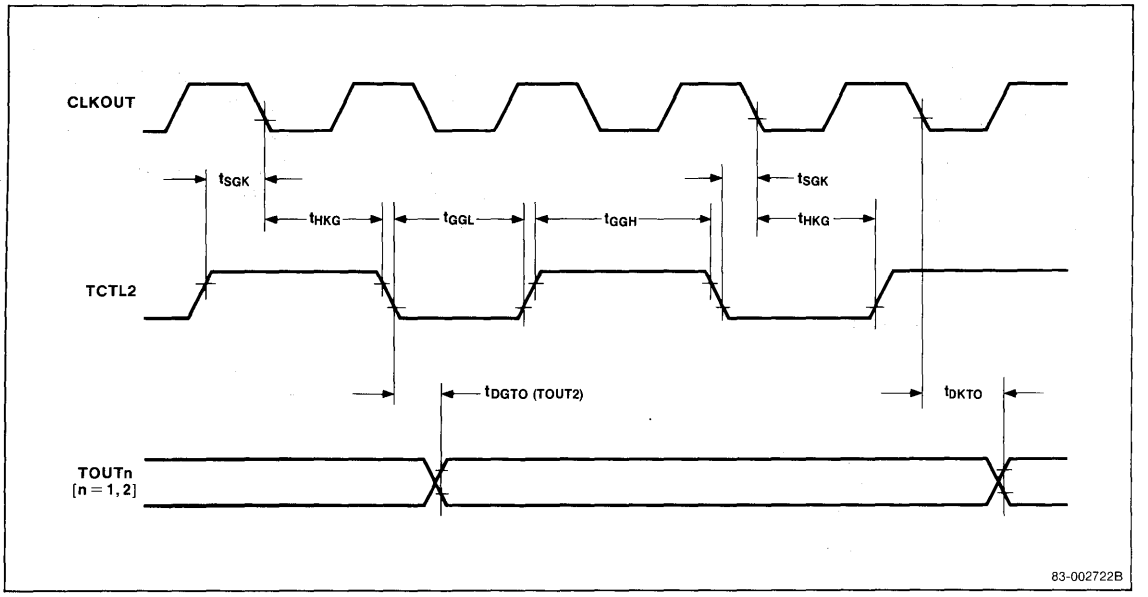


ICU Timing

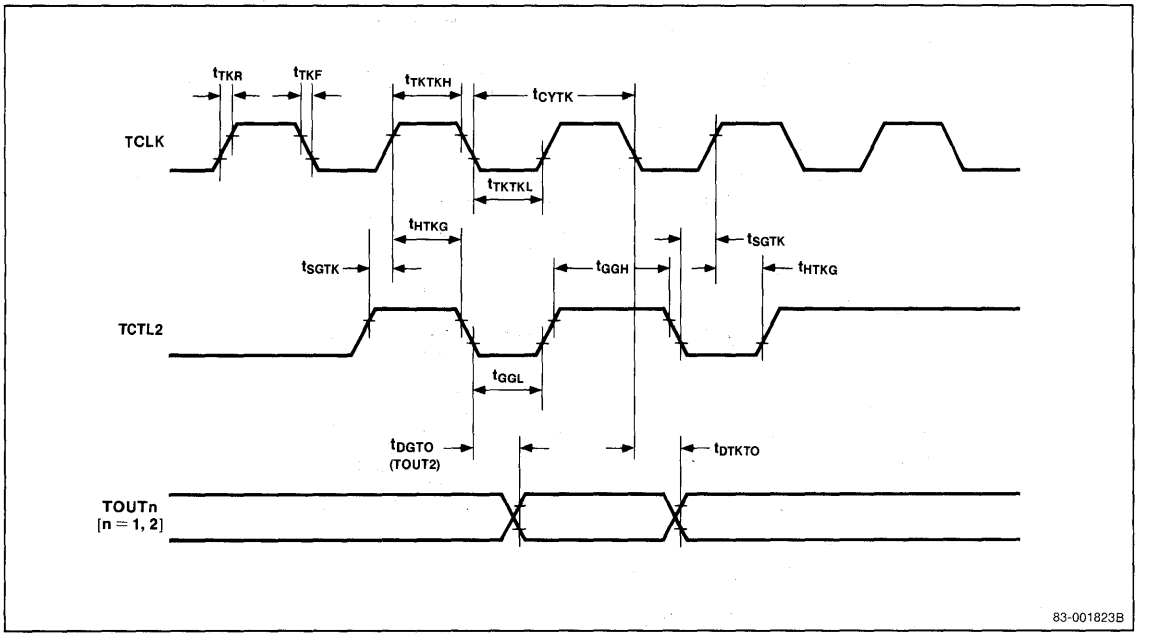


Timing Waveforms (cont)

TCU Timing, Internal Clock Source



TCU Timing, TCLK Source



3c

Functional Description

Refer to the μPD70208 block diagram for an overview of the ten major functional blocks listed below.

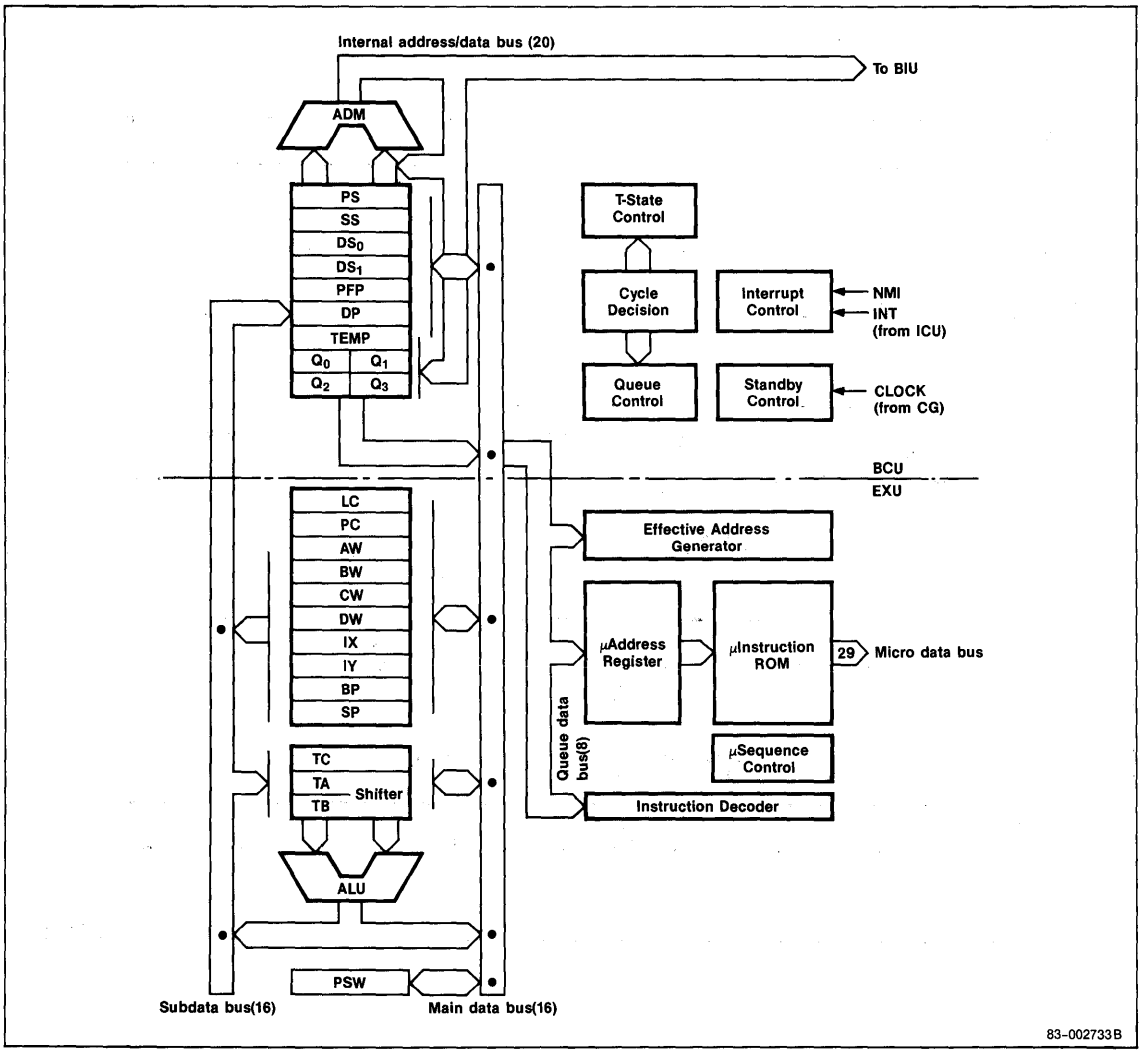
- Central processing unit (CPU)
- Clock generator (CG)
- Bus interface unit (BIU)
- Bus arbitration unit (BAU)
- Refresh control unit (RCU)
- Wait control unit (WCU)
- Timer/counter unit (TCU)
- Serial control unit (SCU)
- Interrupt control unit (ICU)
- DMA control unit (DMAU)

Central Processing Unit

The μPD70208 CPU functions similarly to the CPU of the μPD70108 CMOS microprocessor. However, because the μPD70208 has internal peripheral devices, its bus architecture has been modified to permit sharing the bus with internal peripherals. The μPD70208 CPU is object code compatible with both the μPD70108/μPD70116 and the μPD8086/μPD8088 microprocessors.

Figure 1 is the μPD70208 CPU block diagram. A listing of the μPD70208 instruction set is in the final sections of this data sheet.

Figure 1. μPD70208 CPU Block Diagram



Register Configuration

Program Counter [PC]. The program counter is a 16-bit binary counter that contains the program segment offset of the next instruction to be executed. The PC is incremented each time the microprogram fetches an instruction from the instruction queue. The contents of the PC are replaced whenever a branch, call, return, or break instruction is executed and during interrupt processing. At this time, the contents of the PC are the same as the prefetch pointer (PFP).

Prefetch Pointer [PFP]. The prefetch pointer is a 16-bit binary counter that contains the program segment offset of the next instruction to be fetched for the instruction queue. Because instruction queue prefetch is independent of instruction execution, the contents of the PFP and PC are not always identical. The PFP is updated each time the bus interface unit (BIU) fetches an instruction for the instruction queue. The contents of the PFP are replaced whenever a branch, call, return or break instruction is executed and during interrupt processing. At this time, the contents of the PFP and PC are the same.

Segment Registers [PS, SS, DS_0, DS_1]. The μPD70216 memory address space is divided into 64K-byte logical segments. A memory address is determined by the sum of a 20-bit base address (obtained from a segment register) and a 16-bit offset known as the effective address (EA). I/O address space is not segmented and no segment register is used. The four segment registers are program segment (PS), stack segment (SS), data segment 0 (DS_0), and data segment 1 (DS_1). The following table lists their offsets and overrides.

Default Segment Register	Offset	Override
PS	PFP register	None
SS	SP register	None
SS	Effective address (BP-based)	PS, DS_0, DS_1
DS_0	Effective address (non BP-based)	PS, SS, DS_1
DS_0	IX register (1)	PS, SS, DS_1
DS_1	IY register (2)	None

Note:

(1) Includes source block transfer, output, BCD string, and bit field extraction.

(2) Includes destination block transfer, input, BCD string, and bit field insertion.

General-Purpose Registers. The μPD70208 CPU contains four 16-bit, general-purpose registers (AW, BW, CW, DW), each of which can be used as a pair of 8-bit registers by dividing into upper and lower bytes (AH, AL, BH, BL, CH, CL, DH, DL). General-purpose registers may also be specified implicitly in an instruction. The implicit assignments are:

AW	Word multiplication/division, word I/O, data conversion
AL	Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
AH	Byte multiplication/division
BW	Translation
CW	Loop control, repeat prefix
CL	Shift/rotate bit counts, BCD operations
DW	Word multiplication/division, indirect I/O addressing

Pointer [SP, BP] and Index Registers [IX, IY]. These registers serve as base pointers or index registers when accessing memory using one of the base, indexed, or base indexed addressing modes. Pointer and index registers can also be used as operands for word data transfer, arithmetic, and logical instructions. These registers are implicitly selected by certain instructions as follows.

SP	Stack operations, interrupts
IX	Source block transfer, BCD string operations, bit field extraction
IY	Destination block transfer, BCD string operations, bit field insertion

Program Status Word [PSW]

The program status word consists of six status flags and four control flags.

Status Flags

- V (Overflow)
- S (Sign)
- Z (Zero)
- AC (Auxiliary Carry)
- P (Parity)
- CY (Carry)

Control Flags

- MD (Mode)
- DIR (Direction)
- IE (Interrupt Enable)
- BRK (Break)

3c

When pushed onto the stack, the word image of the PSW is as follows:

15							8
MD	1	1	1	V	DIR	IE	BRK

7							0
S	Z	0	AC	0	P	1	CY

The status flags are set and cleared automatically depending upon the result of the previous instruction execution. Instructions are provided to set, clear, and complement certain status and control flags. Other flags can be manipulated by using the POP PSW instruction.

Between execution of the BRKEM and RETEM instructions, the native mode RETI and POP PSW instructions can modify the MD bit. Care must be exercised by emulation mode programs to prevent inadvertent alteration of this bit.

CPU Architectural Features

The major architectural features of the μPD70208 CPU are:

- Dual data buses
- Effective address generator
- Loop counter
- PC and PFP

Dual Data Buses. To increase performance, dual data buses (figure 2) have been employed in the CPU to fetch operands in parallel and avoid the bottleneck of a single bus. For two-operand instructions and effective address calculations, the dual data bus approach is 30 percent faster than single-bus systems.

Effective Address Generator. Effective address (EA) calculation requires only two clocks regardless of the addressing mode complexity due to the hardware effective address generator (figure 3). When compared with microprogrammed methods, the hardware approach saves between 3 and 10 clock cycles during effective address calculation.

Loop Counter and Shifters. A dedicated loop counter is used to count the iterations of block transfer and multiple shift instructions. This logic offers a significant performance advantage over architectures that control block transfers and multiple shifts using microprogramming. Dedicated shift registers also speed up the execution of the multiply and divide instructions. Compared with microprogrammed methods, multiply and divide instructions execute approximately four times faster.

Figure 2. Dual Data Buses

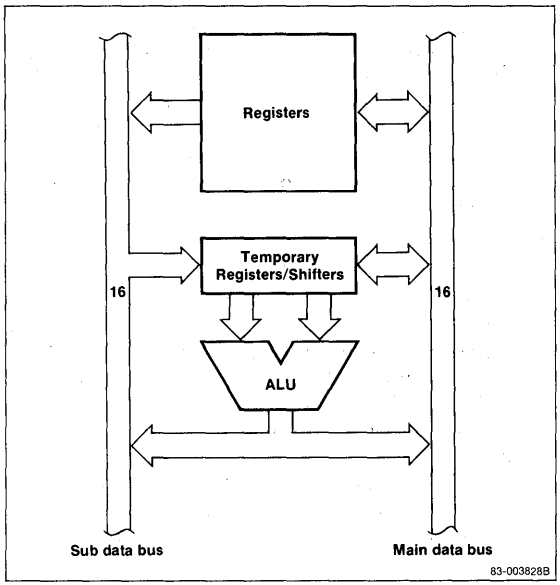
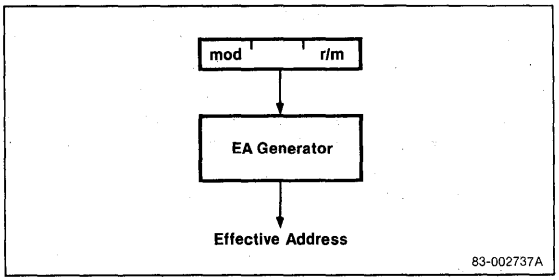


Figure 3. Effective Address Generator



Program Counter and Prefetch Pointer. The functions of instruction execution and queue prefetch are decoupled in the μPD70208. By avoiding a single instruction pointer and providing separate PC and PFP registers, the execution time of control transfers and the interrupt response latency can be minimized. Several clocks are saved by avoiding the need to readjust an instruction pointer to account for prefetching before computing the new destination address.

Enhanced Instruction Set

In addition to the μPD8086/88 instruction set, the μPD70208 has added the following enhanced instructions.

Instruction	Function
PUSH imm	Push immediate data onto stack
PUSH R	Push all general registers onto stack
POP R	Pop all general registers from stack
MUL imm	Multiply register/memory by immediate data
SHL imm8	Shift/rotate by immediate count
SHR imm8	
SHRA imm8	
ROL imm8	
ROR imm8	
ROLC imm8	
RORC imm8	
CHKIND	Check array index
INM	Input multiple
OUTM	Output multiple
PREPARE	Prepare new stack frame
DISPOSE	Dispose current stack frame

Unique Instruction Set

In addition to the μPD70208 enhanced instruction set, the following unique instructions are supported.

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	BCD string addition
SUB4S	BCD string subtraction
CMP4S	BCD string comparison
ROL4	Rotate BCD digit left
ROR4	Rotate BCD digit right
TEST1	Test bit
SET1	Set bit
CLR1	Clear bit
NOT1	Complement bit
REPC	Repeat while carry set
REPNC	Repeat while carry cleared
FPO2	Floating point operation 2

Bit Fields. Bit fields are data structures that range in length from 1 to 16 bits. Two separate operations on bit fields, insertion and extraction, with no restrictions on the position of the bit field in memory are supported. Separate segment, byte offset, and bit offset registers are used for bit field insertion and extraction. Because of their power and flexibility, these instructions are highly effective for graphics, high-level languages, and data packing/unpacking applications.

Insert bit field (INS) copies the bit field of specified length (0 = 1 bit, 15 = 16 bits) from the AW register to the bit field addressed by DS1:IY:reg8 (figure 4). The bit field length can be located in any byte register or supplied as an immediate value. The value in reg8 is a bit field offset. A content of 0 selects bit 0 and 15 selects bit 15 of the word that DS0:IX points to. Following execution, the IY and bit offset register are updated to point to the start of the next bit field.

Bit field extraction (EXT) copies the bit field of specified length (0 = 1 bit, 15 = 16 bits) from the bit field addressed by DS0:IX:reg8 to the AW register (figure 5). If the bit field is less than 16 bits, it is right justified with a zero fill. The bit field length can be located in any byte register or supplied as immediate data. The value in reg8 is a bit field offset. A content of 0 selects bit 0 and 15 selects bit 15 of the word that DS0:IX points to. Following execution, the IX and bit offset register are updated to point to the start of the next bit field.

Packed BCD Strings. These instructions are provided to efficiently manipulate packed BCD data as strings (length from 1 to 254 digits) or as a byte data type with a single instruction.

3c

BCD string arithmetic is supported by the ADD4S, SUB4S, and CMP4S instructions. These instructions allow the source string (addressed by DS0:IX) and the destination string (addressed by DS1:IY) to be manipulated with a single instruction. When the number of BCD digits is even, the Z and CY flags are set according to the result of the operation. If the number of digits is odd, the Z flag will not be correctly set unless the upper 4 bits of the result are zero. The CY flag will not be correctly set unless there is a carry out of the upper 4 bits of the result.

The two BCD rotate instructions (ROR4, ROL4) perform rotation of a single BCD digit in the lower half of the AL register through the register or memory operand.

Bit Manipulation. Four bit manipulation instructions have been added to the μPD70208 instruction set. The ability to test, set, clear, or complement a single bit in a register or memory operand increases code readability as well as performance over the logical operations traditionally used to manipulate bit data.

Repeat Prefixes. Two repeat prefixes (REPC, REPNC) allow conditional block transfer instructions to use the state of the CY flag as a terminating condition. The use of these prefixes allows inequalities to be used when working on ordered data, increasing the performance of searching and sorting algorithms.

Floating Point Operation Instructions. Two floating point operation (FPO) instruction types are recognized by the μPD70208 CPU. These instructions are detected by the CPU, which performs any auxiliary processing such as effective address calculation and the initial bus cycle if specified by the instruction. It is the responsibility of the external coprocessor to latch the address information and data (if a read cycle) from the bus and complete the execution of the instruction.

Figure 4. Bit Field Insertion

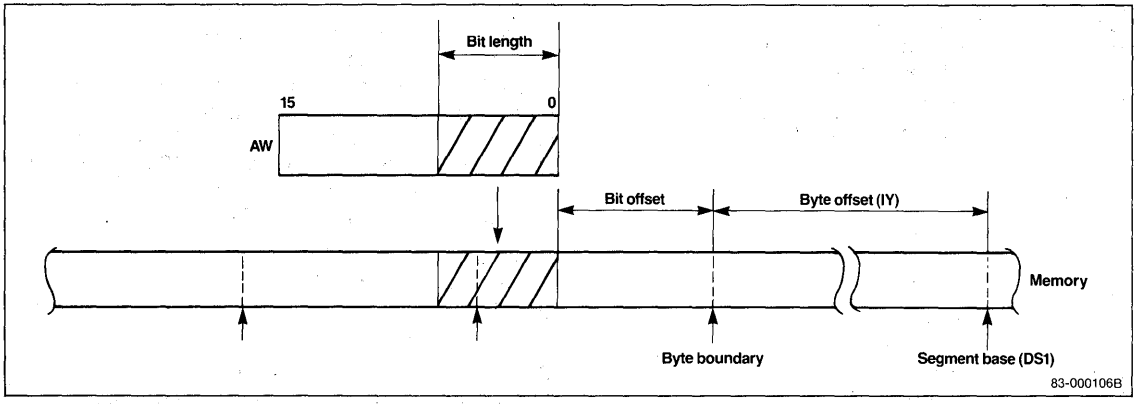
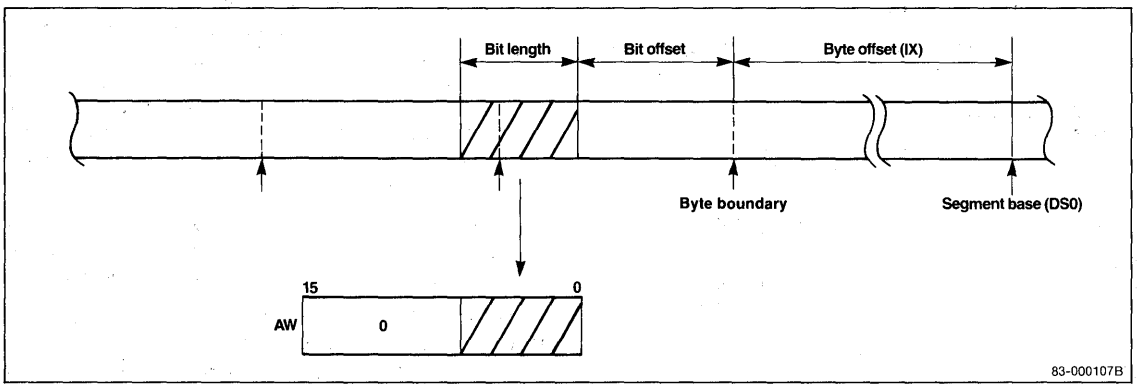


Figure 5. Bit Field Extraction



8080 Emulation Mode. The μPD70208 CPU can operate in either of two modes; see figure 6. Native mode allows the execution of the μPD8086/88, enhanced and unique instructions. The other operating mode is 8080 emulation mode, which allows the entire μPD8080AF instruction set to be executed. A mode (MD) flag is provided to distinguish between the two operating modes. Native mode is active when MD is 1 and 8080 emulation mode is active when MD is 0.

Two instructions are provided to switch from native to 8080 emulation mode and return back. Break for emulation (BRKEM) operates similarly to a BRK instruction, except that after the PSW has been pushed on the native mode stack, the MD flag becomes write-enabled and is cleared.

During 8080 emulation mode, the registers and flags of the 8080 are mapped onto the native mode registers and flags as shown below. Note that PS, SS, DS_0, DS_1, IX, IY, AH, and the upper half of the PSW registers are inaccessible to 8080 programs.

	μμPD8080AF	μPD70208
Registers	A/PSW	AL/PSW (lower)
	B	CH
	C	CL
	D	DH
	E	DL
	H	BH
	L	BL
	SP	BP
	PC	PC
Flags	C	CY
	Z	Z
	S	S
	P	P
	AC	AC

During 8080 emulation mode, the BP register functions as the 8080 stack pointer. The use of separate stack pointers prevents inadvertent damage to the native mode stack pointer by emulation mode programs.

The 8080 emulation mode PC is combined with the PS register to form the 20-bit physical address. All emulation mode data references use DS0 as the segment register. For compatibility with older 8080 software these registers must be equal. By using different segment register contents, separate 64K-byte code and data spaces are possible.

Either an NMI or maskable interrupt will cause the 8080 emulation mode to be suspended. The CPU pushes the PS, PC, and PSW registers on the native mode stack, sets the MD bit (indicating native mode), and enters the specified interrupt handler. When the return from interrupt (RETI) instruction is executed, the PS, PC, and PSW (containing MD=0) are popped from the native stack and execution in 8080 emulation mode continues. Reset will also force a return to native mode.

The 8080 emulation mode programs also have the capability to invoke native mode interrupt handlers by means of the call native (CALLN) instruction. This instruction operates like the BRK instruction except that the saved PSW indicates 8080 emulation mode.

To exit 8080 emulation mode, the return from emulation (RETEM) instruction pops the PS, PC, and PSW from the native mode stack, disables modification of the MD bit, and execution continues with the instruction following the BRKEM instruction. Nesting of 8080 emulation modes is prohibited.

Interrupt Operation

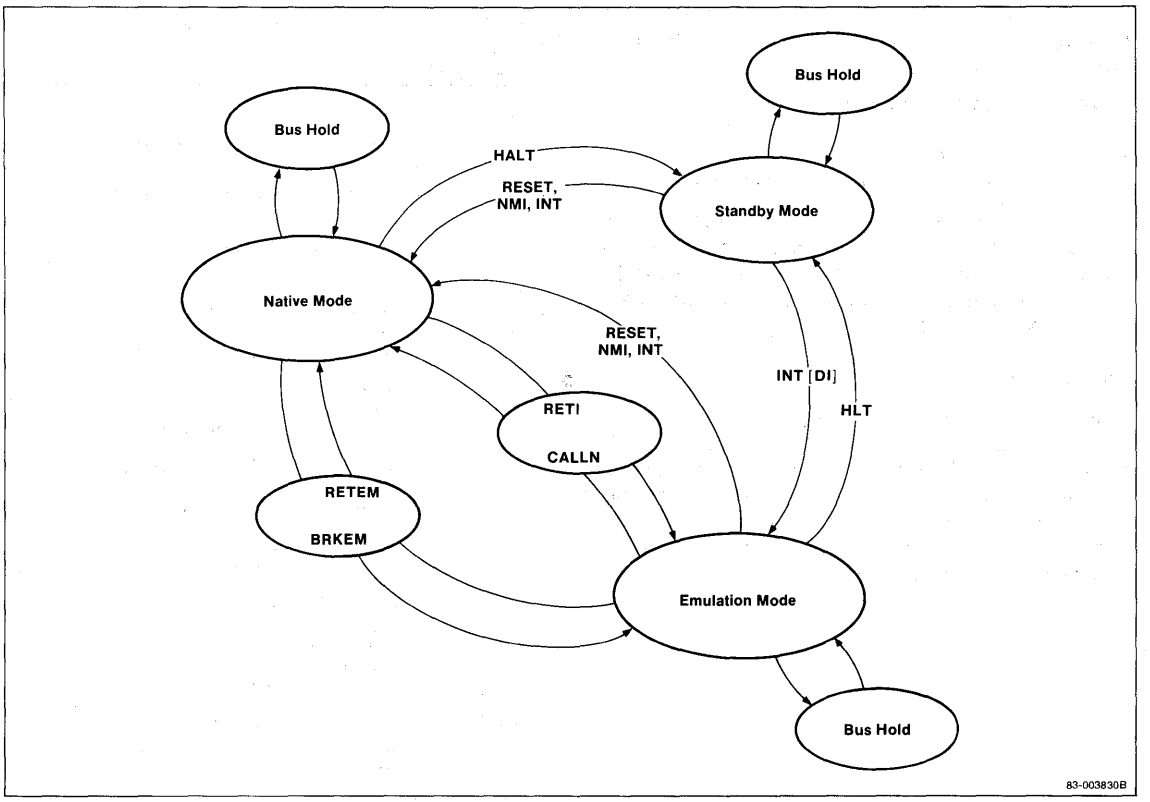
The μPD70208 supports a number of external interrupts and software exceptions. External interrupts are events asynchronous to program execution. On the other hand, exceptions always occur as a result of program execution.

The two types of external interrupts are:

- Nonmaskable interrupt (NMI)
- Maskable interrupt (INT)

3c

Figure 6. μPD70208 Modes



The six software exceptions are:

- Divide error (DIV, DIVU instructions)
- Array bound error (CHKIND instruction)
- Break on overflow (BRKV instruction)
- Break (BRK, BRK3 instructions)
- Single step (BRK bit in PSW set)
- Mode switch (BRKEM, CALLN instructions)

Interrupt vectors are determined automatically for exceptions and the NMI interrupt or supplied by hardware for maskable interrupts. The 256 interrupt vectors are stored in a table (figure 7) located at address 00000H. Vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. Interrupt vectors 32 to 255 are available for use by application software.

Each vector is made up of two words. The word located at the lower address contains the new PC for the interrupt handler. The word at the next-higher address is the new PS value for the interrupt handler. These must be initialized by software at the start of a program.

Nonmaskable interrupts and maskable interrupts (when enabled) are normally serviced following the execution of the current instruction. However, the following cases are exceptions to this rule and the occurrence of the interrupt will be delayed until after the execution of the next instruction.

- Moves to/from segment registers
- POLL instruction
- Instruction prefixes
- EI instruction (maskable interrupts only)

Another special case is the block transfer instructions. These instructions are interruptable and resumable, but because of the asynchronous operation of the BIU, the actual occurrence of the interrupt may be delayed up to three bus cycles.

Standby Mode

The μPD70208 CPU has a low-power standby mode, which can dramatically reduce power consumption during idle periods. Standby mode is entered by simply executing a native or 8080 emulation HALT instruction; no external hardware is required. All other peripherals such as the timer/counter unit, refresh control unit, and DMA control unit continue to operate as programmed.

During standby mode, the clock is distributed only to the circuits required to release the standby mode. When a RESET, NMI, or INT event is detected, the standby mode is released. Both NMI and unmaskable interrupts are processed before control returns to the instruction following the HALT. In the case of the INT input being masked, execution will begin with the instruction immediately following the HALT instruction without an intervening interrupt acknowledge bus cycle. When maskable interrupts are again enabled, the interrupt will be serviced.

Output signal states in the standby mode are listed below.

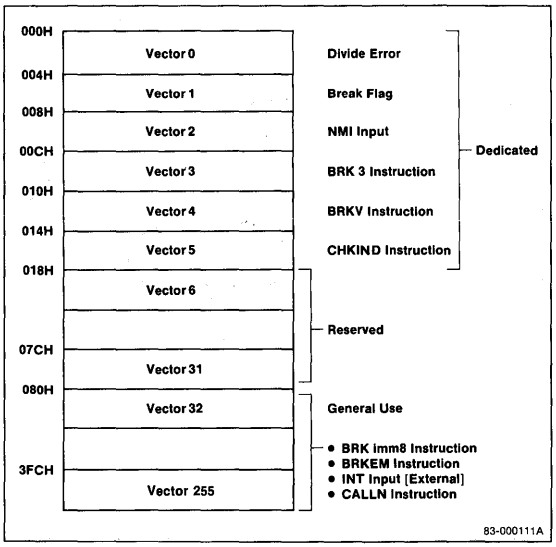
Output Signal	Status in Standby Mode
$\overline{INTAK}$, $\overline{BUFEN}$, $\overline{MRD}$, $\overline{MWR}$, $\overline{IOWR}$, $\overline{IORD}$	High level
BS_2-BS_0 (Note 2)	High level
QS_1-QS_0, ASTB	Low level
$\overline{BUSLOCK}$	High level (low level if the HALT instruction follows the BUSLOCK prefix)
$BUF\overline{R}/W$, A_{19}-A_{16}/PS_3-PS_0, A_{15}-A_8, AD_7-AD_0	High or low level

Note:

(1) Output pin states during refresh and DMA bus cycles will be as defined for those operations.

(2) Halt status is presented prior to entering the passive state.

Figure 7. Interrupt Vector Table



Clock Generator

The clock generator (CG) generates a clock signal half the frequency of a parallel-resonant, fundamental mode crystal connected to pins X1 and X2. Figure 8 shows the recommended circuit configuration. Capacitors C1 and C2, required for frequency stability, are selected to match the crystal load capacitance.

External clock sources are also accommodated as shown in figure 9. The CG distributes the clock to the CLKOUT pin and to each functional block of the μPD70208. The generated clock signal has a 50-percent duty cycle.

Bus Interface Unit

The bus interface unit (BIU) controls the external address, data, and control buses for the three internal bus masters: CPU, DMA control unit (DMAU), and refresh control unit (RCU). The BIU is also responsible for synchronization of the $\overline{\text{RESET}}$ and READY inputs with the clock. The synchronized reset signal is used internally by the μPD70208 and provided externally at the RESOUT pin as a system-wide reset. The synchronized READY signal is combined with the output of the wait control unit (WCU) and is distributed internally to the CPU, DMAU, and RCU. Figure 10 shows the synchronization of $\overline{\text{RESET}}$ and READY.

The BIU also has the capability of overlapping the execution of the next instruction with memory write bus cycles. There is no overlap of instruction execution with read or I/O write bus cycles.

Figure 9. External Oscillator Configuration

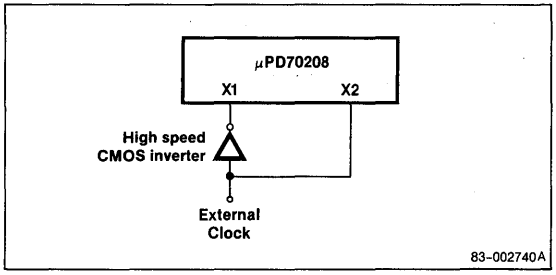
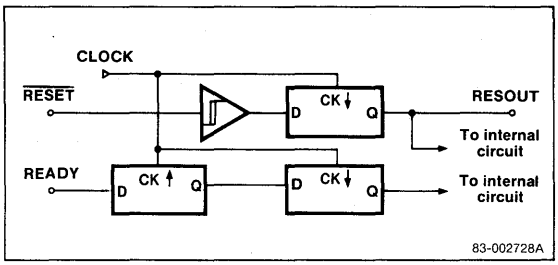
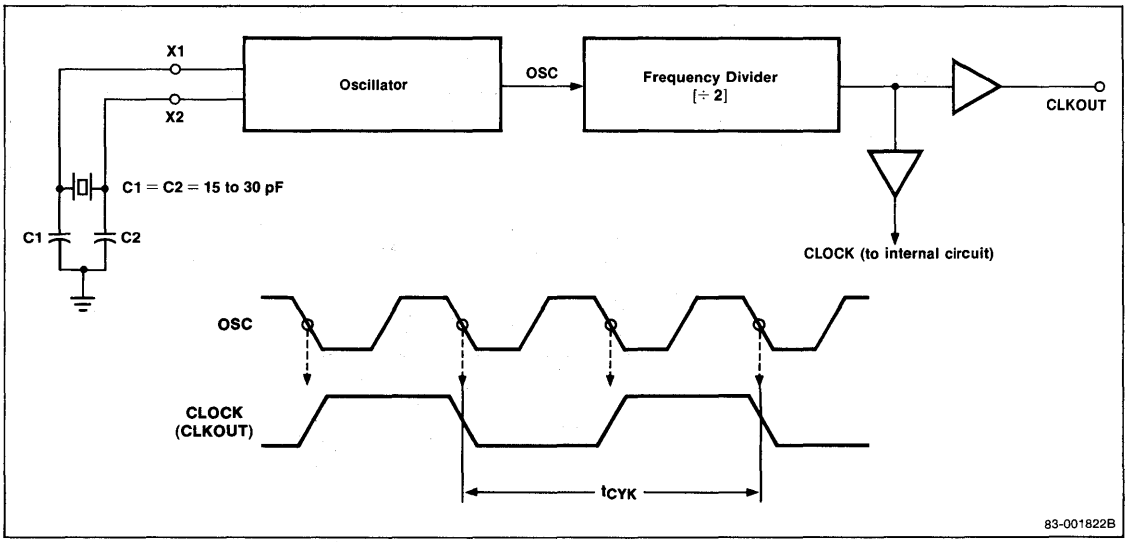


Figure 10. RESET/READY Synchronization



3c

Figure 8. Crystal Configuration



Bus Arbitration Unit

The bus arbitration unit (BAU) arbitrates the external address, data and control buses between the internal CPU, DMAU, and RCU bus requesters and an external bus master. The BAU bus priorities from the highest priority requester to the lowest are:

RCU (Demand mode)
DMAU
HLDRQ
CPU
RCU (Normal mode)

Note that RCU requests the bus at either the highest or lowest priority depending on the status of the refresh request queue. Bus masters other than the CPU are prohibited from using the bus when the CPU is executing an instruction containing a BUSLOCK prefix. Therefore, caution should be exercised when using the BUSLOCK prefix with instructions having a long execution time.

If a bus master with higher priority than the current bus master requests the bus, the BAU inactivates the current bus master's acknowledge signal. When the BAU sees the bus request from the current master go inactive, the BAU gives control of the bus to the higher priority bus master. Whenever possible, the BAU performs bus switching between internal bus masters without the introduction of idle bus cycles, enhancing system throughput.

System I/O Area

The I/O address space from addresses FF00H to FFFFH is reserved for use as the system I/O area. Located in this area are the 12 μPD70208 registers that determine the I/O addressing, enable/disable peripherals, and control pin multiplexing. Byte I/O instructions must be used to access the system I/O area.

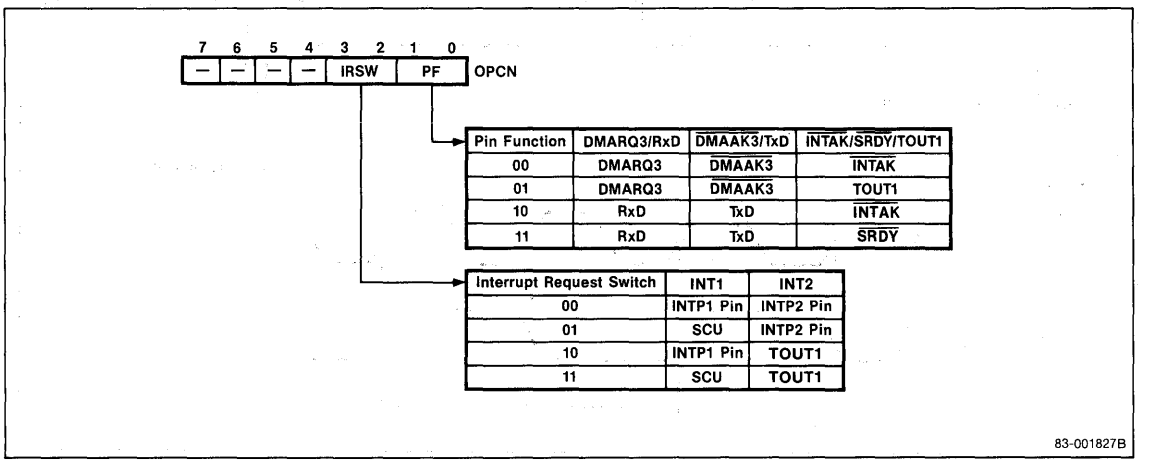
I/O Address	Register	Operation
FFFFH	Reserved	—
FFFEH	OPCN	Read/Write
FFFDH	OPSEL	Read/Write
FFFCH	OPHA	Read/Write
FFFBH	DULA	Read/Write
FFFAH	IULA	Read/Write
FFF9H	TULA	Read/Write
FFF8H	SULA	Read/Write
FFF7H	Reserved	—
FFF6H	WCY2	Read/Write
FFF5H	WCY1	Read/Write
FFF4H	WMB	Read/Write
FFF3H	Reserved	—
FFF2H	RFC	Read/Write
FFF1H	Reserved	—
FFF0H	TCKS	Read/Write

On-Chip Peripheral Connection Register

The on-chip peripheral connection (OPCN) register controls multiplexing of the μPD70208 multiplexed pins. Figure 11 shows the format of the OPCN register. The interrupt request switch (IRSW) field controls multiplexing of ICU interrupt inputs INT1 and INT2. The output of an internal peripheral or an external interrupt source can be selected as the INT1 and INT2 inputs to the ICU.

The pin function (PF) field in the OPCN selects one of four possible states for the $\overline{\text{DMARQ3}}$/RxD, $\overline{\text{DMAAK3}}$/TxD, and $\overline{\text{INTAK}}$/TOUT1/$\overline{\text{SRDY}}$ pins. Bit 0 of the

Figure 11. OPCN Register Format

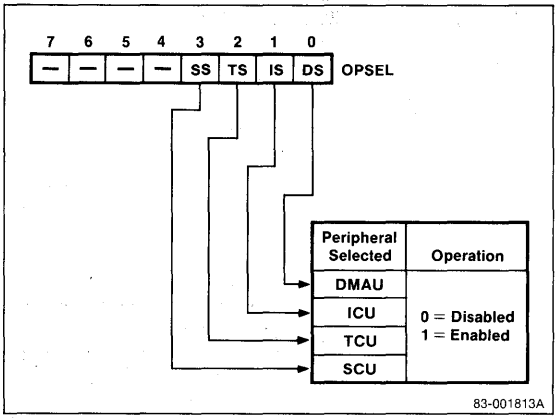


OPCN controls the function of the $\overline{\text{INTAK}}$/TOUT1/$\overline{\text{SRDY}}$ pin. If cleared, $\overline{\text{INTAK}}$ will appear on this output pin. If bit 0 is set, either TOUT1 or $\overline{\text{SRDY}}$ will appear at the output depending on the state of bit 1. If bit 1 is cleared, DMA channel 3 I/O signals will appear on the DMARQ3/RxD and $\overline{\text{DMAAK3}}$/TxD pins. If the SCU is to be used, bit 1 of the PF field must be set.

On-Chip Peripheral Selection Register

The on-chip peripheral selection (OPSEL) register is used to enable or disable the μPD70208 internal peripherals. Figure 12 shows the format of the OPSEL register. Any of the four (DMAU, TCU, ICU, SCU) peripherals can be independently enabled or disabled by setting or clearing the appropriate OPSEL bit.

Figure 12. OPSEL Register Format



Internal Peripheral Relocation Registers

The five internal peripheral relocation registers (figure 13) are used to fix the I/O addresses of the DMAU, ICU, TCU, and SCU. The on-chip peripheral high-address (OPHA) register is common to all four internal peripherals and fixes the high-order byte of the 16-bit I/O address. The individual DMAU low-address (DULA) register, ICU low-address (IULA) register, TCU low-address (TULA) register, and the SCU low-address (SULA) register select the low-order byte of the I/O addresses for the DMAU, ICU, TCU, and SCU peripherals.

The contents of the OPHA register are:

OPHA (7 … 0)

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

The formats for the individual internal peripheral registers appear below. Since address checking is not performed, do not overlap two peripheral I/O address spaces.

DULA (7 … 0)

A_7	A_6	A_5	A_4	—	—	—	—

IULA (7 … 0)

A_7	A_6	A_5	A_4	A_3	A_2	—	—

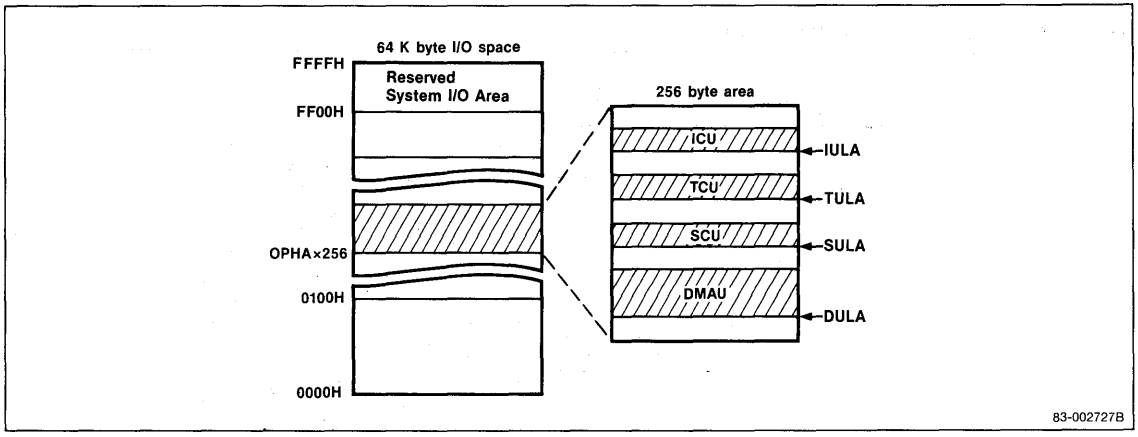
TULA (7 … 0)

A_7	A_6	A_5	A_4	A_3	A_2	—	—

SULA (7 … 0)

A_7	A_6	A_5	A_4	A_3	A_2	—	—

Figure 13. μPD70208 Peripheral Relocation



Timer Clock Selection Register

The timer clock selection (TCKS) register selects the clock source for each of the timer/counters as well as the divisor for the internal clock prescaler. Figure 14 shows the format of the TCKS register. The clock source for each timer/counter is independently selected from either the prescaled internal CPU clock or from an external clock source (TCLK). The internal clock is derived from the CLKOUT signal and can be divided by 2, 4, 8, or 16 before being presented to the clock select logic.

Refresh Control Unit

The refresh control unit (RCU) refreshes external dynamic RAM devices by outputting a 9-bit row address on address lines A_8-A_0 and performing a memory read bus cycle. External logic can distinguish a refresh bus cycle by monitoring the refresh request ($\overline{REFRQ}$) pin. Following each refresh bus cycle, the refresh row counter is incremented.

The refresh control (RFC) register in the system I/O area contains two fields. The refresh enable field enables or disables the refreshing function. The refresh timer (RTM) field selects a refresh interval to match the dynamic memory refresh requirements. Figure 15 shows the format for the RFC register.

To minimize the impact of refresh on the system bus bandwidth, the μPD70208 utilizes a refresh request queue to store refresh requests and perform refresh bus cycles in otherwise idle bus cycles.

The RCU normally requests the bus as the lowest-priority bus requester (normal mode). However, if seven refresh requests are allowed to accumulate in the RCU refresh request queue, the RCU will change to the highest-priority bus requester (demand mode). The RCU will then perform back-to-back refresh cycles until three requests remain in the queue. This guarantees the integrity of the DRAM system while maximizing performance.

The refresh count interval can be calculated as follows:

Refresh interval $= 8 \times N \times t_{CYK}$

where N is the timer factor selected by the RTM field.

When the μPD70208 is reset, the RE field in the RFC register is unaffected and the RTM field is set to 01000 (N = 9). No refresh bus cycles occur while $\overline{RESET}$ is asserted.

Figure 15. Refresh Control Register

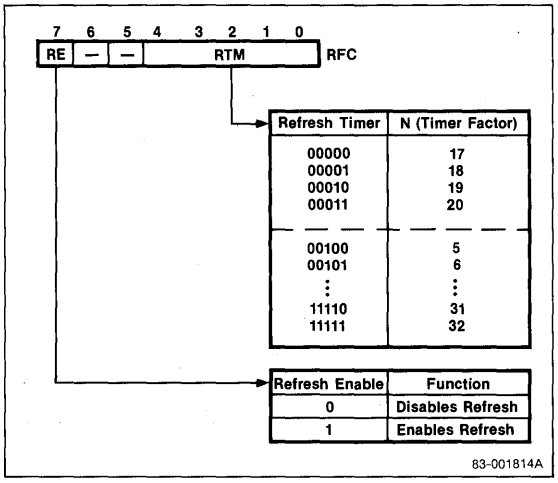
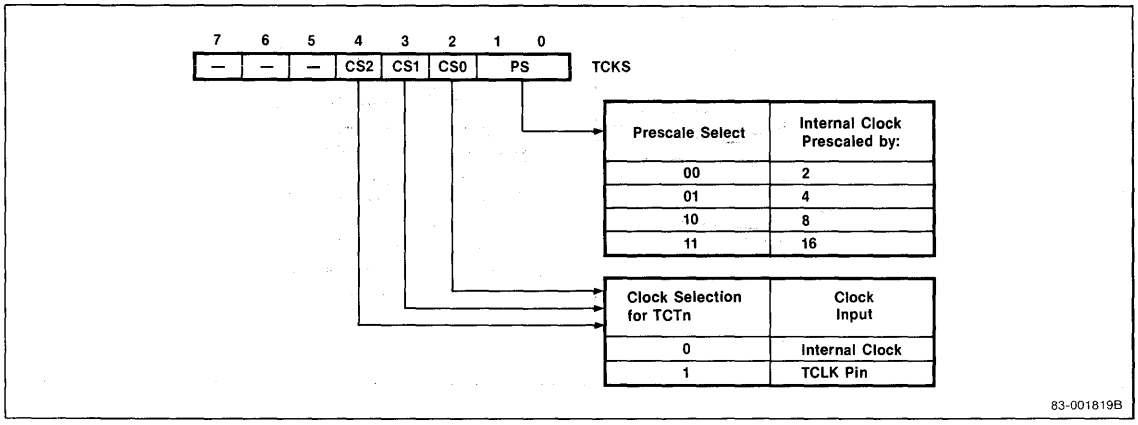


Figure 14. Timer Clock Selection Register



Wait Control Unit

The wait control unit (WCU) inserts from zero to three wait states into a bus cycle in order to compensate for the varying access times of memory and I/O devices. The number of wait states for CPU, DMAU, and RCU bus cycles is separately programmable. In addition, the memory address space is divided into three independent partitions to accommodate a wide range of system designs. $\overline{\text{RESET}}$ initializes the WCU to insert three wait states in all bus cycles. This allows operation with slow memory and peripheral devices before the initialization of the WCU registers.

The three system I/O area registers that control the WCU are wait cycle 1 (WCY1), wait cycle 2 (WCY2), and wait state memory boundary (WMB). The WCU always inserts wait states corresponding to the wait count programmed in WCY1 or WCY2 registers into a bus cycle, regardless of the state of the external READY input. After the programmed number of wait states occurs, the WCU will insert Tw states as long as the READY pin remains inactive. When READY is again asserted, the bus cycle continues with T4 as the next cycle. The μPD70208 internal peripherals never require wait states; four clock cycles will terminate an internal peripheral bus cycle.

CPU Wait States

The WMB register divides the 1M-byte memory address space into three independent partitions: lower, middle, and upper. Figure 16 shows the WMB register format.

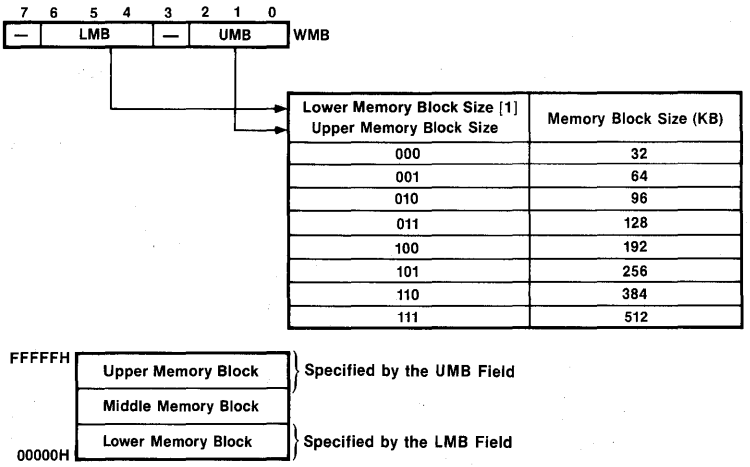
Initialization software can then set the number of wait states for each memory partition and the I/O partition via the WCY1 register (figure 17).

DMA and Refresh Wait States

The WCY2 register (figure 18) specifies the number of wait states to be automatically inserted in DMA and refresh bus cycles. DMA wait states must be set to the maximum of the DMA memory and I/O partitions. Refresh wait states should be set to the maximum value of all DRAM memory partitions.

3c

Figure 16. Wait State Memory Boundary Register



Lower Memory Block Size [1] Upper Memory Block Size	Memory Block Size (KB)
000	32
001	64
010	96
011	128
100	192
101	256
110	384
111	512

Note:
[1] By default, the address space remaining between the UBM and LBM is the middle memory block.

83-001820B

Figure 17. Wait Cycle 1 Register

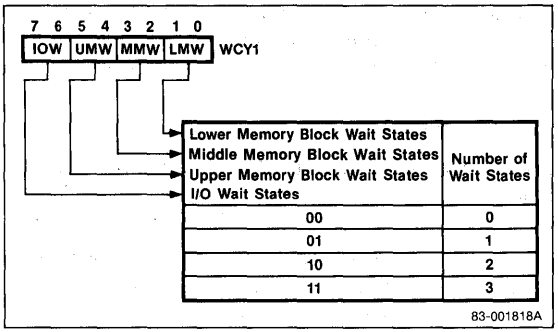
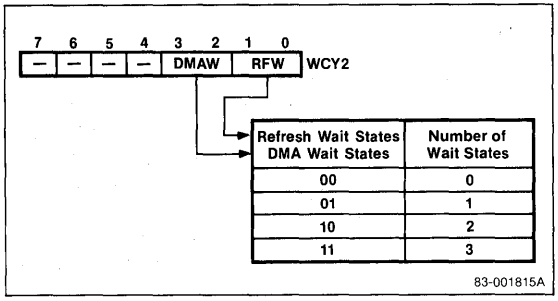


Figure 18. Wait Cycle 2 Register



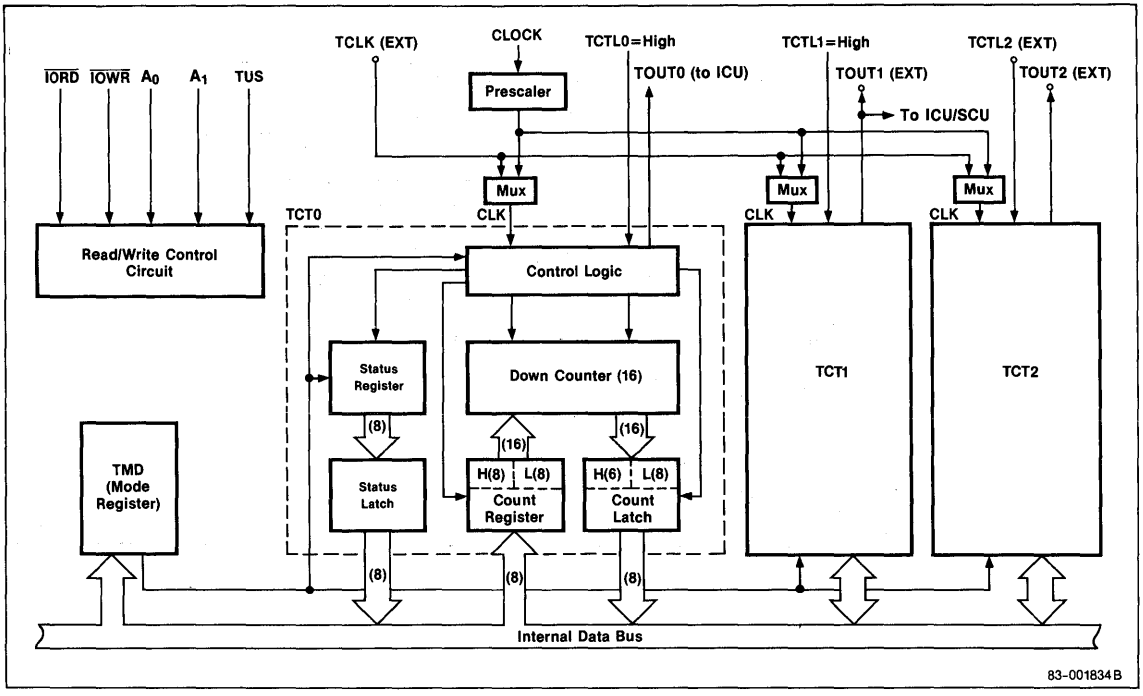
Timer/Counter Unit

The timer/counter unit (TCU) provides a set of three independent 16-bit timer/counters. The output signal of timer/counter 0 is hardwired internally as an interrupt source. The output of timer/counter 1 is available internally as an interrupt source, used as a baud rate generator, or used as an external output. The timer/counter 2 output is available as an external output. Due to mode restrictions, the TCU is a subset of the μPD71054 Programmable Timer/Counter. Figure 19 shows the internal block diagram of the TCU.

The TCU has the following features:

- Three 16-bit timer/counters
- Six programmable count modes
- Binary/BCD counting
- Multiple latch command
- Choice of two clock sources

Figure 19. TCU Block Diagram



Because $\overline{RESET}$ leaves the TCU in an uninitialized state, each timer/counter must be initialized by specifying an operating mode and a count. Once programmed, a timer/counter will continue to operate in that mode until another mode is selected. When the count has been written to the counter and transferred to the down counter, a new count operation starts. Both the current count and the counter status can be read while count operations are in progress.

TCU Commands

The TCU is programmed by issuing I/O instructions to the I/O port addresses programmed in the OPHA and TULA registers. The individual TCU registers are selected by address bits A_1 and A_0 as follows.

A_1	A_0	Register	Operation
0	0	TCT0 TST0	Read/Write Read
0	1	TCT1 TST1	Read/Write Read
1	0	TCT2 TST2	Read/Write Read
1	1	TMD	Write

The timer mode (TMD) register selects the operating mode for each timer/counter and issues the latch command for one or more timer/counters. Figure 20 shows the format for the TMD register.

Writes to the timer/counter 2-0 (TCT2-TCT0) registers stores the new count in the appropriate timer/counter. The count latch command is used before reading count data in order to latch the current count and prevent inaccuracies.

The timer status 2-0 (TST2-TST0) registers contain status information for the specified counter (figure 21). The latch command is used to latch the appropriate counter status before reading status information. If both status and counter data are latched for a counter, the first read operation returns the status data and subsequent read operations obtain the count data.

Count Modes

There are six programmable timer/counter modes. The timing waveforms for these modes are in figure 22.

Mode 0 [Interrupt on End of Count]. In this mode, TOUT changes from the low to high level when the specified count is reached. This mode is available on all timer/counters.

Mode 1 [Retriggerable One-Shot]. In mode 1, a low-level one-shot pulse, triggered by TCTL2 is output from the TOUT2 pin. This mode is available only on timer/counter 2.

Mode 2 [Rate Generator]. In mode 2, TOUT cyclically goes low for one clock period when the counter reaches the 0001H count. A counter in this mode operates as a frequency divider. All timer/counters can operate using mode 2.

Mode 3 [Square-Wave Generator]. Mode 3 is a frequency divider similar to mode 2, but the output has a symmetrical duty cycle. This mode is available on all three timer/counters.

Mode 4 [Software-Triggered Strobe]. In mode 4, when the specified count is reached, TOUT goes low for the duration of one clock pulse. Mode 4 is available on all timer/counters.

Mode 5 [Hardware-Triggered Strobe]. Mode 5 is similar to mode 4 except that operation is triggered by the TCTL2 input and can be retriggered. This mode is available only on timer/counter 2.

3c

Serial Control Unit

The serial control unit (SCU) is a single asynchronous serial channel that performs serial communication between the μPD70208 and an external serial device. The SCU is similar to the μPD71051 Serial Control Unit except for the lack of synchronous communication protocols. Figure 23 is the block diagram of the SCU.

The SCU has the following features.

- Full-duplex asynchronous serial controller
- Clock rate divisor (x16, x64)
- Baud rates to 250 kb/s supported
- 7-, 8-bit character lengths
- 1-, 2-bit stop bit lengths
- Break transmission and detection
- Full-duplex, double-buffered transmitter/receiver
- Even, odd, or no parity
- Parity, overrun, and framing error detection
- Receiver-full/transmitter-empty interrupt

The SCU contains four separately addressable registers for reading/writing data, reading status, and controlling operation of the SCU. The serial receive buffer (SRB) and the serial transmit buffer (STB) store the incoming and outgoing character data. The serial status (SST) register allows software to determine the current state of both the transmitter and receiver. The serial command (SCM) and serial mode (SMD) registers determine the operating mode of the SCU while the serial interrupt mask (SIMK) register allows software control of the SCU receive and transmit interrupts.

Figure 20. Timer Mode Register

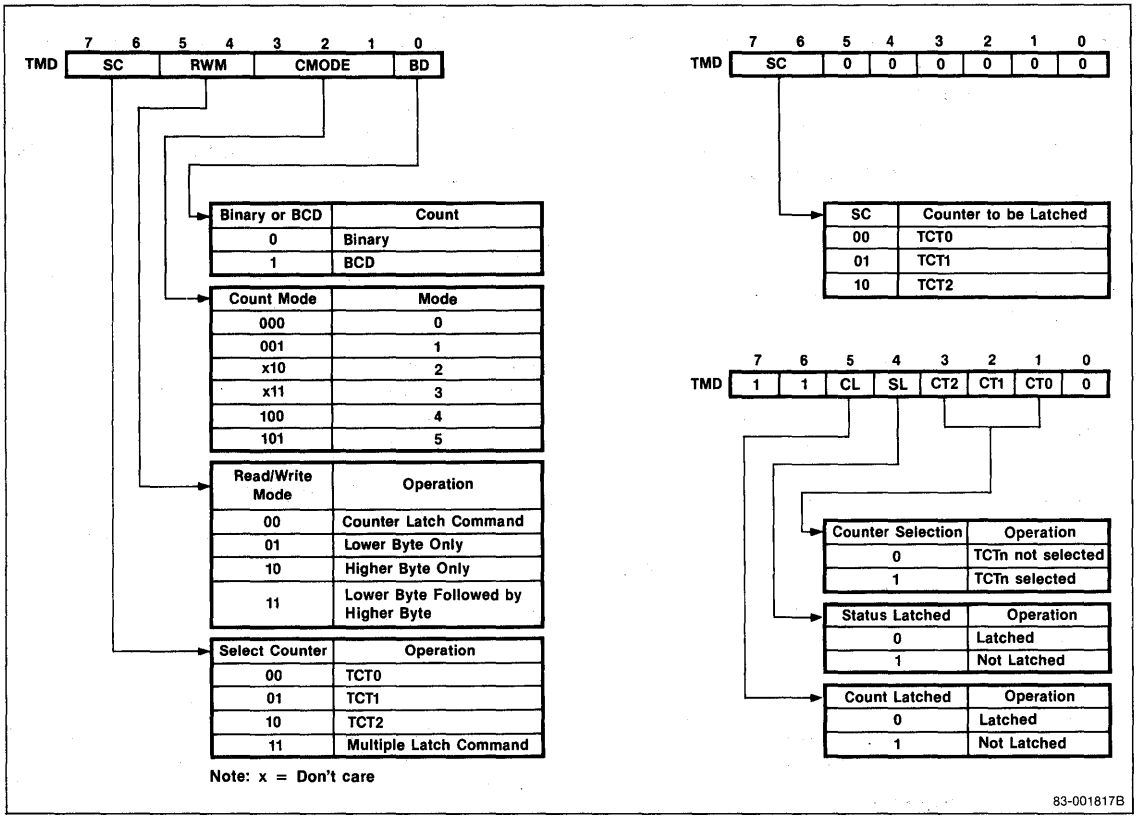


Figure 21. TCU Status Register

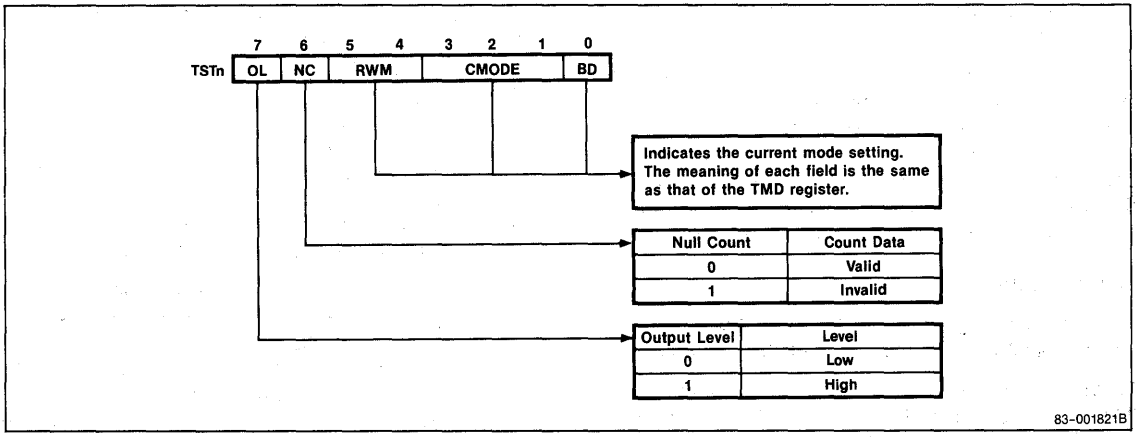
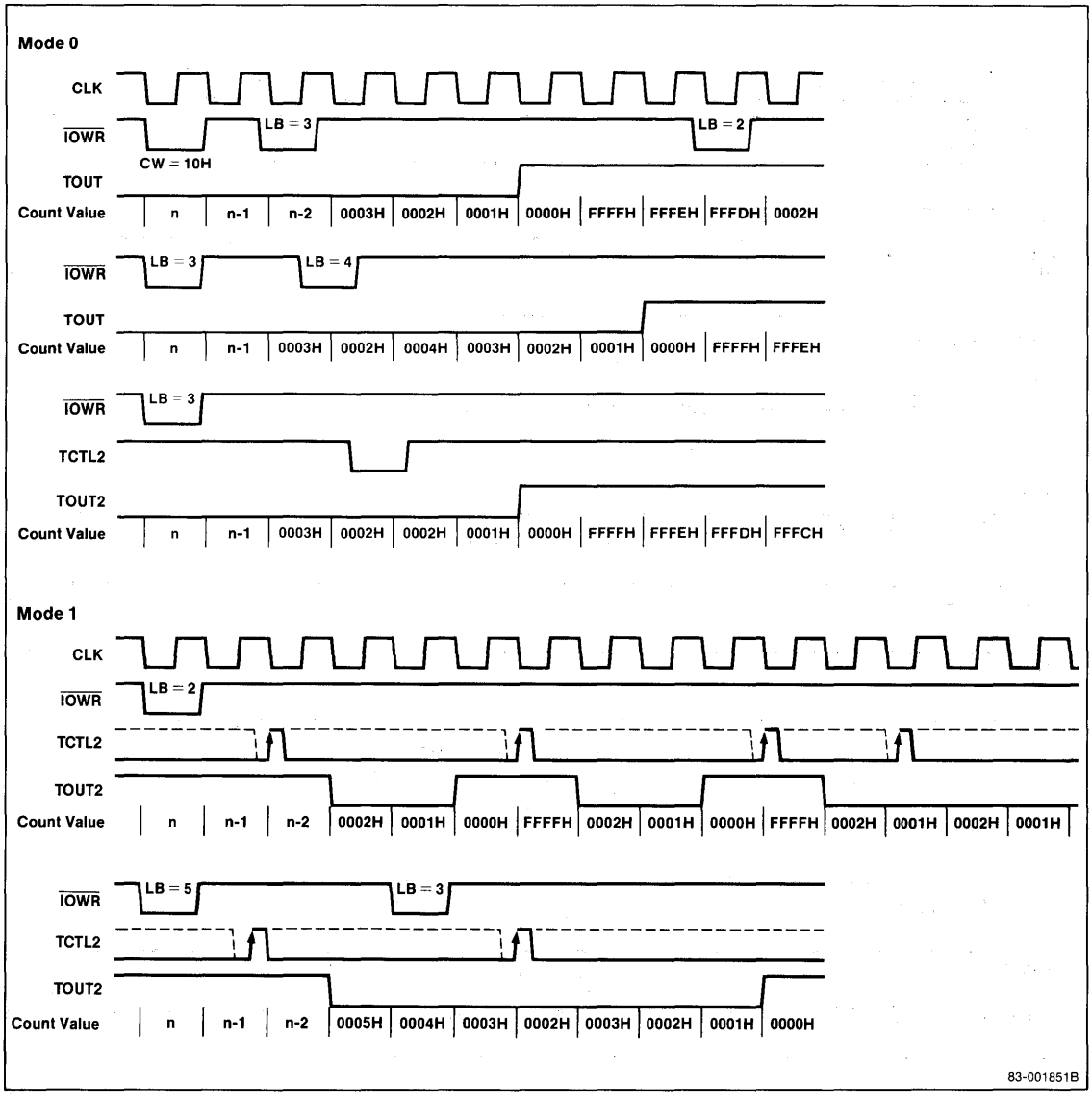


Figure 22. TCU Waveforms (Sheet 1 of 3)



3c

Figure 22. TCU Waveforms (Sheet 2 of 3)

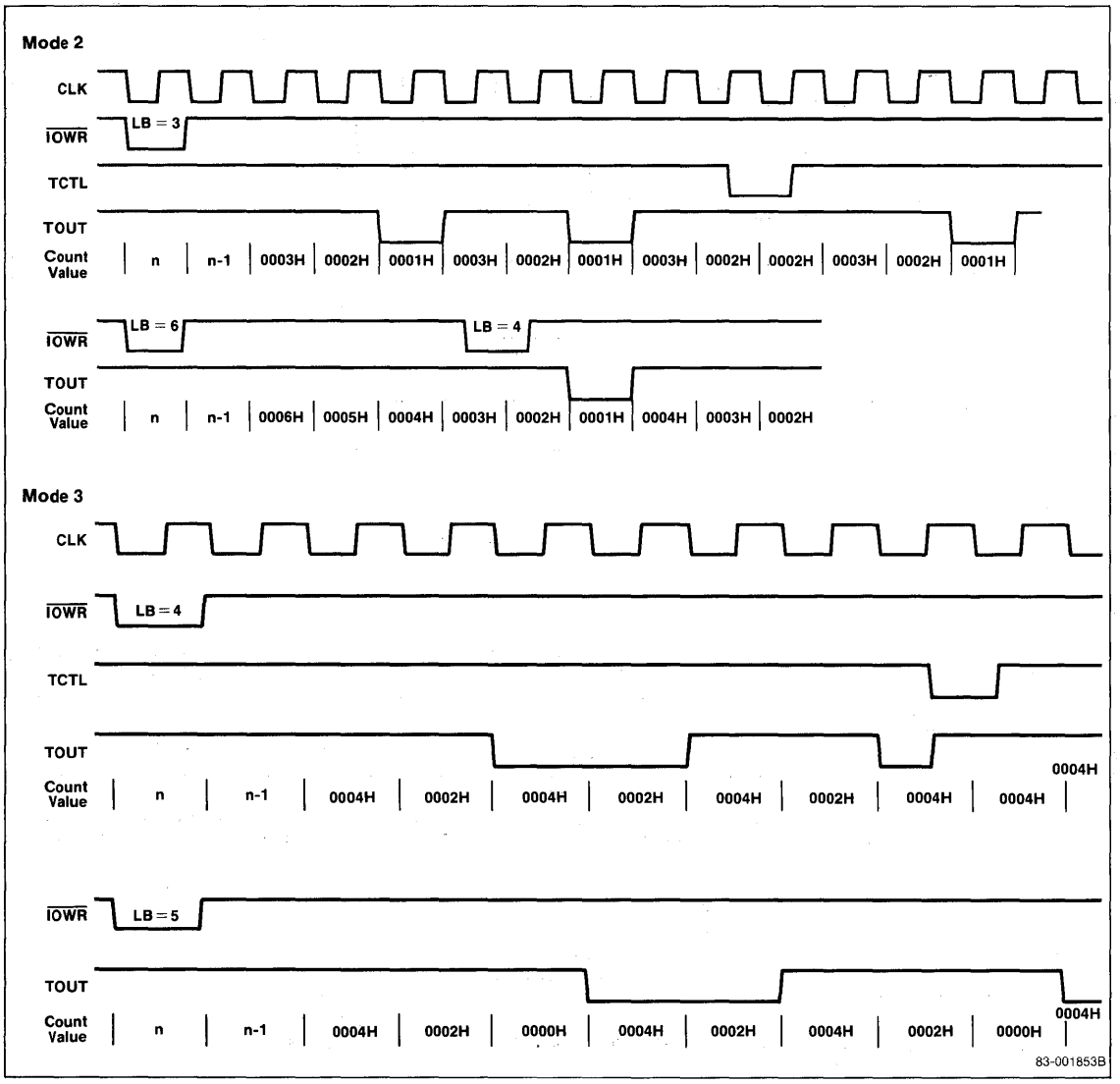
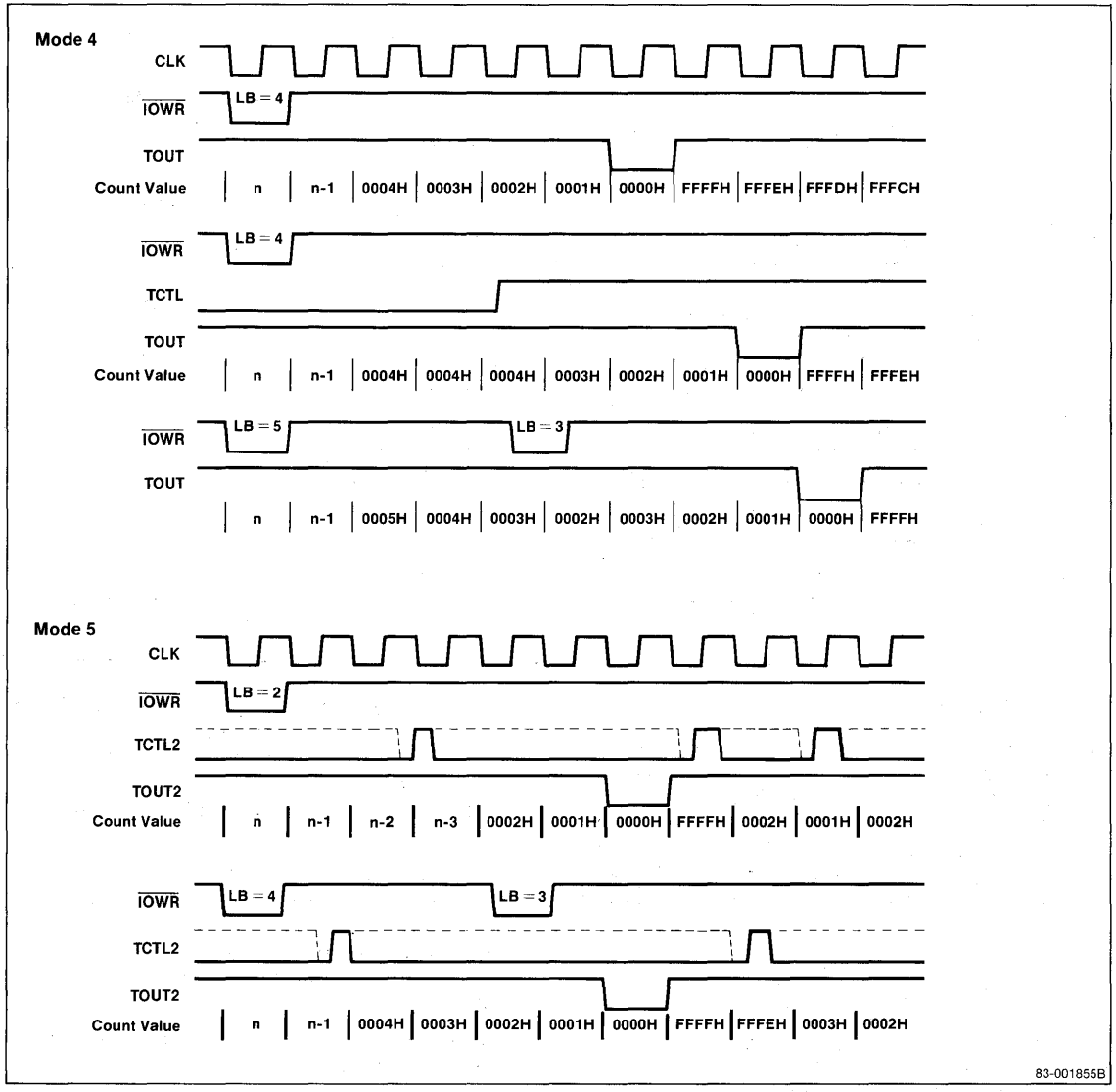
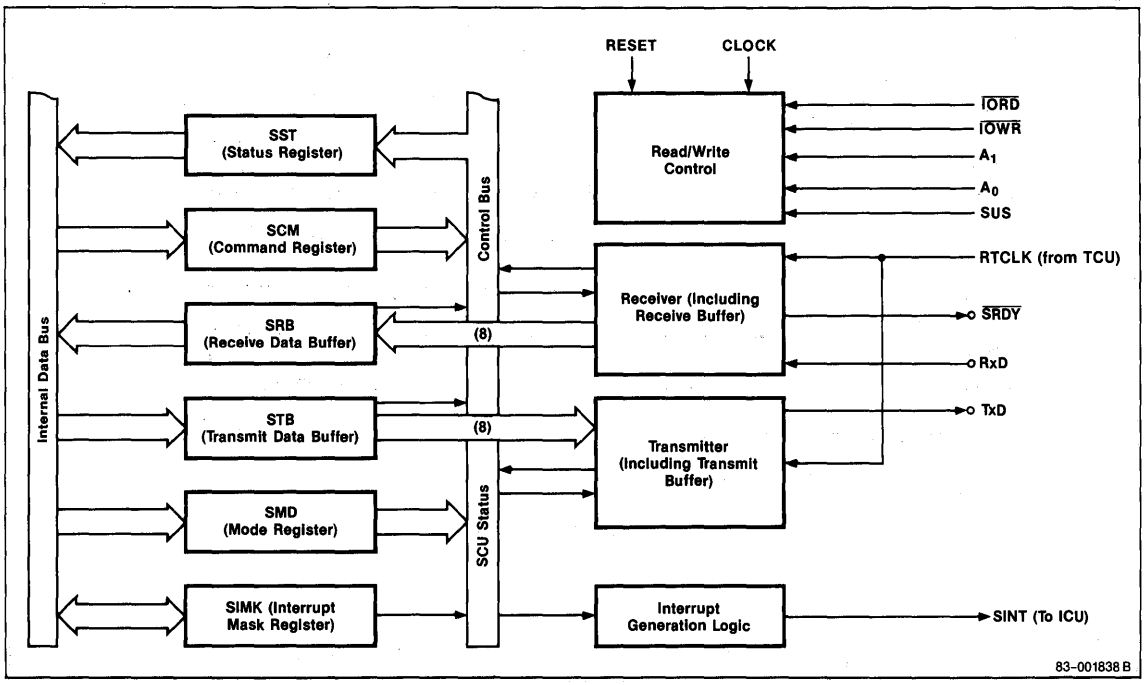


Figure 22. TCU Waveforms (Sheet 3 of 3)



3c

Figure 23. SCU Block Diagram



Receiver Operation

While the RxD pin is high, the receiver is in an idle state. A transition on RxD from high to low indicates the start of new serial data. When a complete character has been received, it is transferred to the SRB; the receive buffer ready (RBRDY) bit in the SST register is set and (if unmasked) an interrupt is generated. The SST also latches any parity, overrun, or framing errors at this time.

The receiver detects a break condition when a null character with zero parity is received. The BRK bit is set for as long as the subsequent receive data is low and resets when RxD returns to a high level. The MRDY bit (SCM) and RBRDY (SST) are gated to form the output $\overline{SRDY}$. $\overline{SRDY}$ prevents overruns from occurring when the program is unable to process the input data. Software can control MRDY to prevent data from being sent from the remote transmitter while RBRDY can prevent the immediate overrun of a received character.

Transmitter Operation

TxD is kept high while the STB register is empty. When the transmitter is enabled and a character is written to the STB register, the data is converted to serial format and output on the TxD pin. The start bit indicates the start of the transmission and is followed by the character stream (LSB to MSB) and an optional parity bit. One or two stop bits are then appended, depending on the programmed mode. When the character has been transferred from the STB, the TRBDY bit in the SST is set and if unmasked, a transmit buffer empty interrupt is generated.

Serial data can be transmitted and received by polling the SST register and checking the TBRDY or RBRDY flags. Data can also be transmitted and received by SCU-generated interrupts to the interrupt control unit. The SCU generates an interrupt in either of these conditions:

(1) The receiver is enabled, the SRB is full, and receive interrupts are unmasked.

(2) The transmitter is enabled, the STB is empty, and transmit interrupts are unmasked.

SCU Registers and Commands

I/O instructions to the I/O addresses selected by the OPHA and SULA registers are used to read/write the SCU registers. Address bits A_1 and A_0 and the read/write lines select one of the six internal registers as follows:

A_1	A_0	Register	Operation
0	0	SRB STB	Read Write
0	1	SST SCM	Read Write
1	0	SMD	Write
1	1	SIMK	Read/Write

The SRB and STB are 8-bit registers. When the character length is 7 bits, the lower 7 bits of the SRB register are valid and bit 7 is cleared to 0. If programmed for 7-bit characters, bit 7 of the STB is ignored.

The SST register (figure 24) contains the status of the transmit and receive data buffers and the error flags. Error flags are persistent. Once an error flag is set, it remains set until a clear error flags command is issued.

Figure 25 shows the SCM and SMD registers. The SCM register stores the command word that controls transmission, reception, error flag reset, break transmission, and the state of the $\overline{SRDY}$ pin. The SMD register stores the mode word that determines serial characteristics such as baud rate divisor, parity, character length, and stop bit length.

Initialization software should first program the SMD register followed by the SCM register. Unlike the μPD71051, the SMD register can be modified at any time without resetting the SCU.

The SIMK register (figure 26) controls the occurrence of RBRDY and TBRDY interrupts. When an interrupt is masked, it is prevented from propagating to the interrupt control unit.

3c

Baud Rate Generator

Timer/counter 1 is used as the baud rate generator when the SCU is enabled. The input baud rate clock is scaled by 16 or 64, as selected in the SMD register, to determine the receive/transmit data clock. There are no restrictions on the SCU input baud rate clock other than operating the TCU in mode 3 with a square-wave output.

Figure 24. SST Register

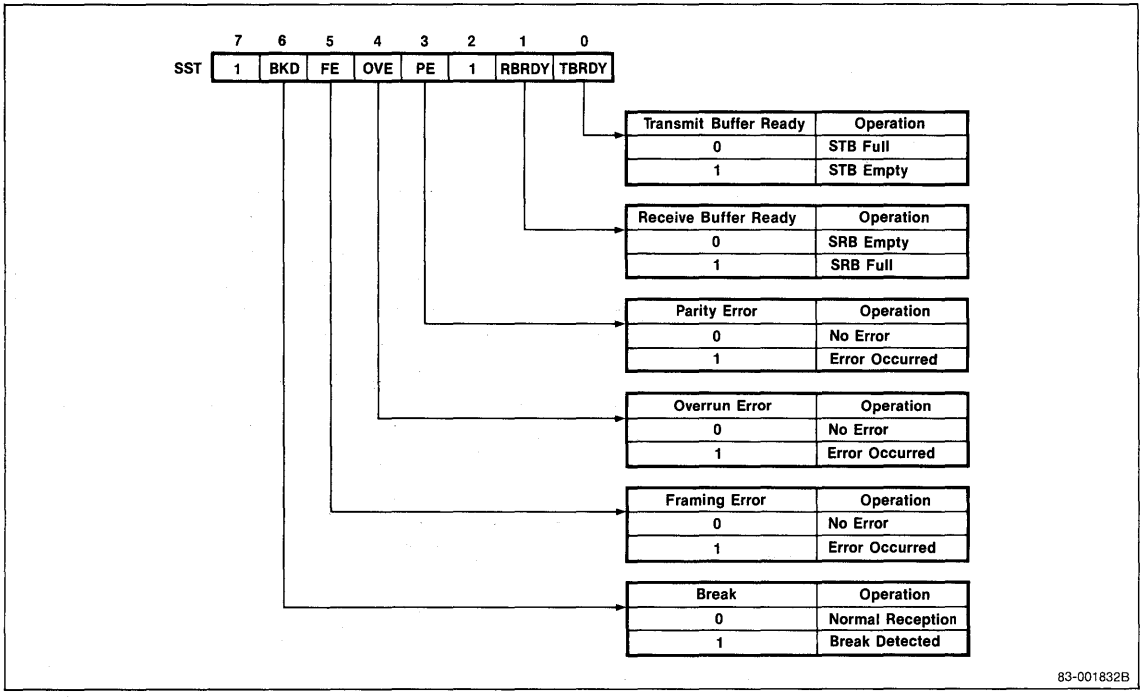


Figure 25. SCM and SMD Registers

SCM Register

	7	6	5	4	3	2	1	0
SCM	—	—	$\overline{MRDY}$	ECL	SBRK	RE	—	TE

Transmitter Enabled	Operation
0	Transmitter Disabled
1	Transmitter Enabled

Receiver Enabled	Operation
0	Receiver Disabled
1	Receiver Enabled

Send Break	Operation
0	Normal Operation
1	TxD = 0 (Break)

Error Clear	Operation
0	No Operation
1	Error Flag Clear

$\overline{MRDY}$	Mask Ready
0	$\overline{SRDY}$ = 1 (Mask)
1	Normal Operation of $\overline{SRDY}$ Output

SMD Register

	7	6	5	4	3	2	1	0
SMD	STL		PS		CL		BF	

Baud Rate Factor	Operation
0-	Illegal
10	RTCLK Frequency ÷ 16
11	RTCLK Frequency ÷ 64

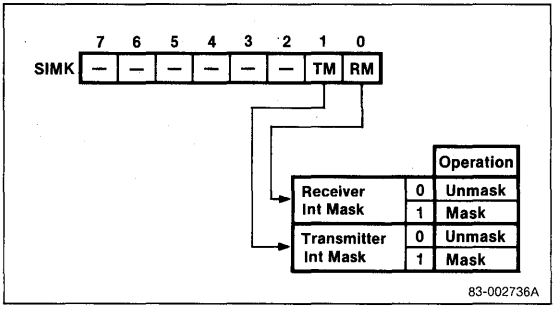
Character Length	Operation
0-	Illegal
10	7 Bit Characters
11	8 Bit Characters

Parity Select	Operation
-0	No Parity
01	Odd Parity
11	Even Parity

Stop Bit Length	Operation
-0	Illegal
01	1 Stop Bit
11	2 Stop Bits

83-001836B

Figure 26. SIMK Register



Interrupt Control Unit

The interrupt control unit (ICU) is a programmable interrupt controller equivalent to the μPD71059. The ICU arbitrates up to eight interrupt inputs, generates a CPU interrupt request, and outputs the interrupt vector number on the internal data bus during an interrupt acknowledge cycle. Cascading up to seven external slave μPD71059s permits the μPD70208 to support up to 56 interrupt sources. Figure 27 is the block diagram for the ICU.

The ICU has the following features.

- Eight interrupt request inputs
- Cascadable with μPD71059 Interrupt Controllers
- Programmable edge- or level-triggered interrupts (TCU, edge-triggered interrupts only)
- Individually maskable interrupt requests
- Programmable interrupt request priority
- Polling mode

ICU Registers

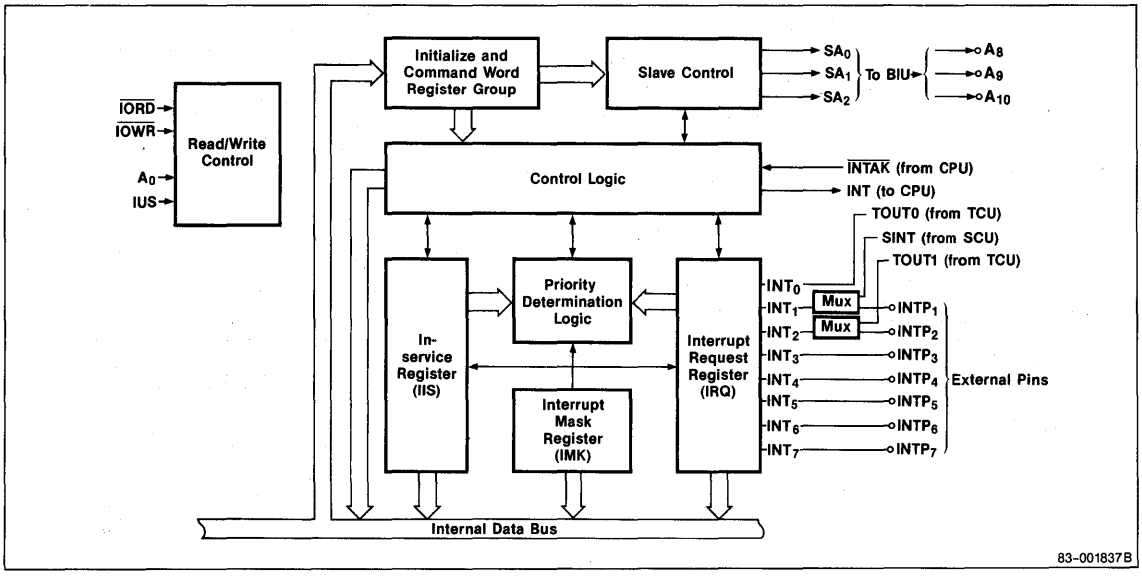
Use I/O instructions to the I/O addresses selected by the OPHA and IULA registers to read from and write to the ICU registers. Address bit A_0 and the command word selects an ICU internal register.

	A_0	Other Condition	Operation
Read	0	IMD selects IRQ	CPU ← IRQ data
	0	IMD selects IIS	CPU ← IIS data
	0	Polling phase	CPU ← Polling data
	1	—	CPU ← IMKW
Write	0	D4 = 1	CPU → IIW1
	0	D4 = 0 and D3 = 0	CPU → IPFW
	0	D4 = 0 and D3 = 1	CPU → IMDW
	1	During initialization	CPU → IIW2
	1		CPU → IIW3
	1		CPU → IIW4
	1	After initialization	CPU → IMKW

Note:

(1) In polling phase, polling data has priority over the contents of the IRQ or IIS register when read.

Figure 27. ICU Block Diagram



Initializing the ICU

The ICU is always used to service maskable interrupts in a μPD70208 system. Prior to accepting maskable interrupts, the ICU must first be initialized (figure 28). Following initialization, command words from the CPU can change the interrupt request priorities, mask/unmask interrupt requests, and select the polling mode. Figures 29 and 30 list the ICU initialization and command words.

Interrupt initialization words 1-4 (IIW1-IIW4) initialize the ICU, indicate whether external μPD71059s are connected as slaves, select the base interrupt vector, and select edge- or level-triggered inputs for INT1-INT7. Interrupt sources from the TCU are fixed as edge-triggering. INT0 is internally connected to TOUT0, and INT2 may be connected to TOUT1 by the IRSW field in the OPCN.

The interrupt mask word (IMKW) contains programmable mask bits for each of the eight interrupt inputs. The interrupt priority and finish word (IPFW) is used by the interrupt handler to terminate processing of an interrupt or change interrupt priorities. The interrupt mode word (IMDW) selects the polling register, interrupt request (IRQ) or interrupt in service (IIS) register, and the nesting mode.

The initialization words are written in consecutive order starting with IIW1. IIW2 sets the interrupt vector. IIW3 specifies which interrupts are connected to slaves. IIW3 is only required in extended systems. The ICU will only expect to receive IIW3 if SNGL = 0 (bit D_1 of IIW1). IIW4 is only written if II4 = 1 (bit D_0 of IIW1).

μPD71059 Cascade Connection

To increase the number of maskable interrupts, up to seven slave μPD71059 Interrupt Controllers can be cascaded. During cascade operation (figure 31), each slave μPD71059 INT output is routed to one of the μPD70208 INTP inputs. During the second interrupt acknowledge bus cycle, the ICU places the slave address on address lines A_{10}-A_8. Each slave compares this address with the slave address programmed using interrupt initialization word 3 (IIW3). If the same, the slave will place the interrupt vector on pins AD_7-AD_0 during the second interrupt acknowledge bus cycle.

Figure 28. Initialization Sequence

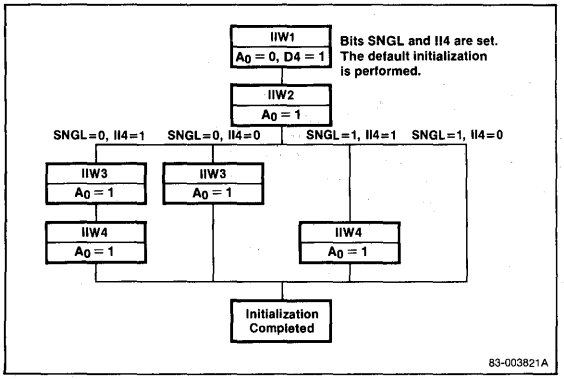


Figure 29. Interrupt Initialization Words 1-4

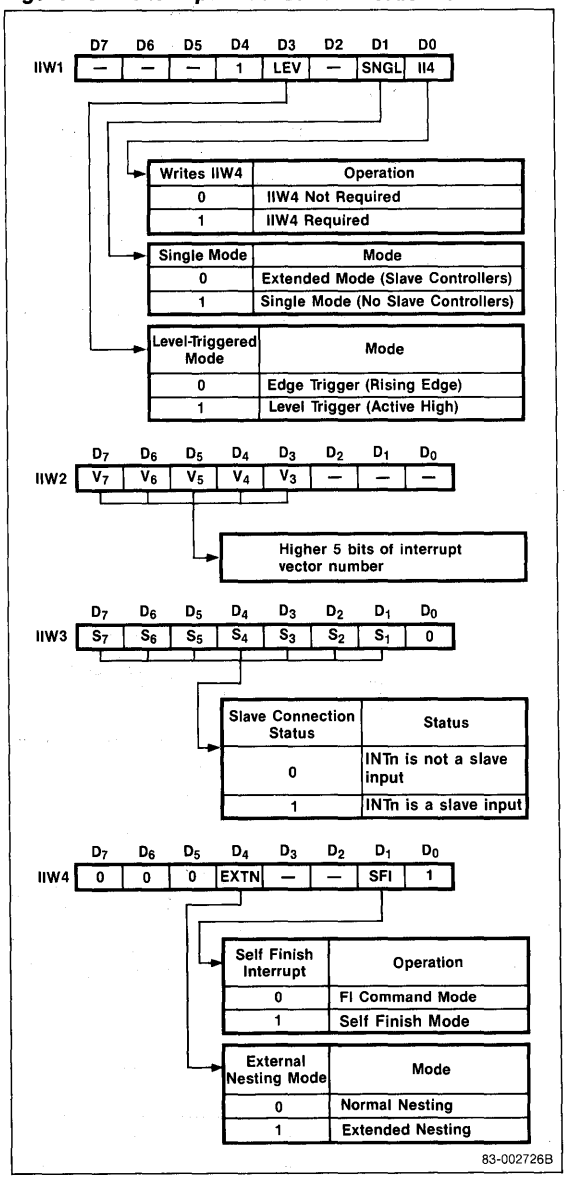


Figure 30. Command Words

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
IMKW	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

Interrupt Request Mask	Operation
0	INTn not Masked
1	INTn Masked

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
IPFW	RP	SIL	FI	0	0	IL2	IL1	IL0

Interrupt Level				
	0	0	0	INT0
	0	0	1	INT1
	0	1	0	INT2
	0	1	1	INT3
	1	0	0	INT4
	1	0	1	INT5
	1	1	0	INT6
	1	1	1	INT7

Rotate Priority Specify Interrupt Level, Finish Interrupt Commands	0	0	1	FI Command	No Level Specification	No Rotation	Normal FI Command
	1	0	1			Rotation	Normal Rotation FI Command
	0	1	1		Level Specification	No Rotation	FI Command for Specification
	1	1	1			Rotation	Specified Bit Rotation FI Command
	0	1	0	Non-FI Command		No Rotation	No Operation
	1	1	0			Rotation	Specified Bit Rotation Command
	0	0	0		No Level Specification	No Rotation	Self FI Mode Rotation Reset
	1	0	0			Rotation	Self FI Mode Rotation Set

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
IMDW	—	SNM	EXCN	0	1	POL	SR	IS/IR

Select Register to Read	In-Service/Request Register Select	Read Register Selection
0	—	No Operation
1	0	IRQ Selection
1	1	IIS Selection

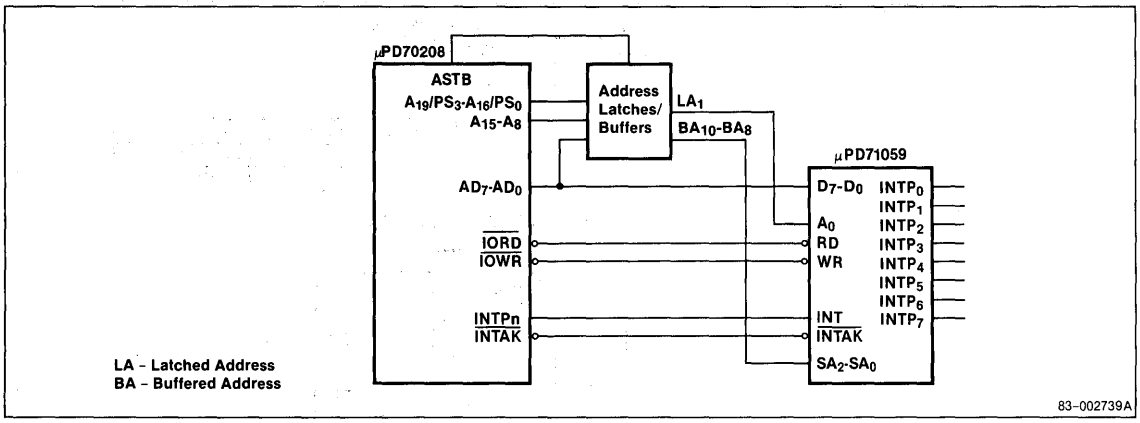
Polling Mode	Polling
0	No Operation
1	Polling Command

Set Nesting Mode	Exceptional Nesting Mode	Nesting Mode 2
0	—	No Operation
1	0	Exceptional Nesting Mode Release
1	1	Exceptional Nesting Mode Set

83-001835C

3c

Figure 31. μPD71059 Cascade Connection



DMA Control Unit

The DMA Control Unit (DMAU) is a high-speed DMA controller compatible with the μPD71071 DMA Controller. The DMAU has four independent DMA channels and performs high-speed data transfers between memory and external peripheral devices at speeds as high as 2 megabytes/second in an 8-MHz system. Figure 32 is the block diagram for the DMAU.

The DMAU has the following features.

- Four independent DMA channels
- Cascade mode for slave μPD71071 DMA controllers
- 20-bit address registers
- 16-bit transfer count registers
- Single, demand, and block transfer modes
- Bus release and bus hold modes
- Autoinitialization
- Address increment/decrement
- Fixed/rotating channel priorities
- TC output at transfer end
- Forced termination of service by $\overline{END}$ input

DMAU Basic Operation

The DMAU operates in either a slave or master mode. In the slave mode, the DMAU samples the four DMARQ input pins every clock. If one or more inputs are active, the corresponding DMA request bits are set and the DMAU sends a bus request to the BAU while continuing to sample the DMA request inputs. After the BAU returns the DMA bus acknowledge signal, the DMAU stops DMA request sampling, selects the DMA channel with the highest priority, and enters the bus master mode to perform the DMA transfer. While in the bus master mode, the DMAU controls the external bus and performs DMA transfers based on the preprogrammed channel information.

Terminal Count

The DMAU ends DMA service when the terminal count condition is generated or when the $\overline{END}$ input is asserted. A terminal count (TC) is produced when the contents of the current count register becomes zero. If autoinitialization is not enabled when DMA service terminates, the mask bit of the channel is set and the DMARQ input of that channel is masked. Otherwise, the current count and address registers are reloaded from the base registers and new DMA transfers are again enabled.

DMA Transfer Type

The type of transfer the DMAU performs depends on the following conditions.

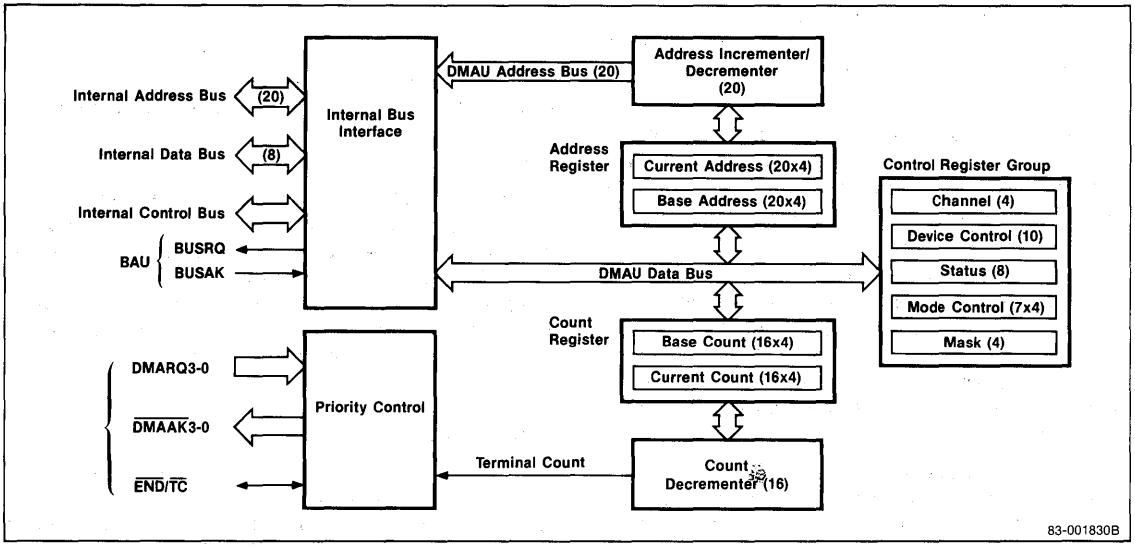
- Direction of the transfer (each channel)
- Transfer mode (each channel)
- Bus mode

Transfer Direction

All DMA transfers use memory as a reference point. Therefore, a DMA read operation transfers data from memory to an I/O port. A DMA write operation reads an I/O port and writes the data into memory. During memory-to-I/O transfer, the DMA mode (DMD) register is used to select the transfer directions for each channel and activate the appropriate control signals.

Operation	Transfer Direction	Activated Signals
DMA read	Memory → I/O	$\overline{IOWR}$, $\overline{MRD}$
DMA write	I/O → Memory	$\overline{IORD}$, $\overline{MWR}$
DMA verify		Addresses only; no transfer performed

Figure 32. DMAU Block Diagram

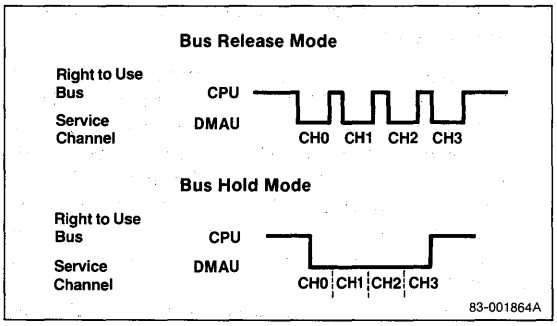


Bus Mode

The DMA device control (DDC) register selects operation in either the bus release or bus hold mode. The selected bus mode determines the DMAU conditions for return of the bus to the BAU. Figure 33 shows that in bus release mode, only a single channel is serviced after the DMAU obtains the bus. When DMA service ends (termination conditions depend on the transfer mode), the DMAU returns the bus to the BAU regardless of the state of other DMA requests, and the DMAU reenters the slave mode. When the DMAU regains use of the bus, a new DMA operation can begin.

In bus hold mode, several channels can receive contiguous service without releasing the bus. If there is another valid DMA request when a channel's DMA service is finished, the new DMA service can begin immediately after the previous service without returning the bus to the BAU.

Figure 33. Bus Modes



Transfer Modes

The DMD register also selects either single, demand, or block transfer mode for each channel. The conditions for the termination of each transfer characterize each transfer mode. The following table shows the various transfer modes and termination conditions.

Transfer Mode	Termination Conditions
Single	After each byte/word transfer
Demand	$\overline{\text{END}}$ input Terminal count Inactive DMARQ DMARQ of a higher priority channel becomes active (bus hold mode)
Block	$\overline{\text{END}}$ input Terminal count

3c

The operation of single, demand, and block mode transfers depends on whether the DMAU is in bus release or bus hold mode. Figure 34 shows the operation flow for the six possible transfer and bus mode operations in DMA transfer.

Single-Mode Transfer. In bus release mode, when a channel completes transfer of a single byte, the DMAU enters the slave mode regardless of the state of DMA request inputs. In this manner, other lower-priority bus masters will be able to access the bus.

In bus hold mode, when a channel completes transfer of a single byte, the DMAU terminates the channel's service even if the DMARQ request signal is asserted. The DMAU will then service any other requesting channel. If there are no requests from any other DMA channels, the DMAU releases the bus and enters the slave state.

Demand-Mode Transfer. In bus release mode, the currently active channel continues to transfer data as long as the DMA request of that channel is active, even though other DMA channels are issuing higher-priority requests. When the DMA request of the serviced channel becomes inactive, the DMAU releases the bus and enters the slave state.

In bus hold mode, when the active channel completes a single transfer, the DMAU checks the other DMA request lines without ending the current service. If there is a higher-priority DMA request, the DMAU stops the service of the current channel and starts servicing the highest-priority channel requesting service. If there is no higher request than the current one, the DMAU continues to service the currently active channel. Lower-priority DMA requests are honored without releasing the bus after the current channel service is complete.

Block-Mode Transfer. In bus release mode, the current channel continues DMA transfers until a terminal count or the external $\overline{END}$ input becomes active. During this time, the DMAU ignores all other DMA requests. After completion of the block transfer, the DMAU releases the bus and enters the slave state, even if DMA requests from other channels are active.

In bus hold mode, the current channel transfers data until an internal or external $\overline{END}$ signal becomes active. When the service is complete, the DMAU checks all DMA requests without releasing the bus. If there is an active request, the DMAU immediately begins servicing the request. The DMAU releases the bus after it honors all DMA requests or a higher-priority bus master requests the bus.

Byte Transfer

The DMD register can specify only byte DMA transfers for each channel. Depending on the mode selected, the address register can either increment or decrement, whereas the count register is always decremented.

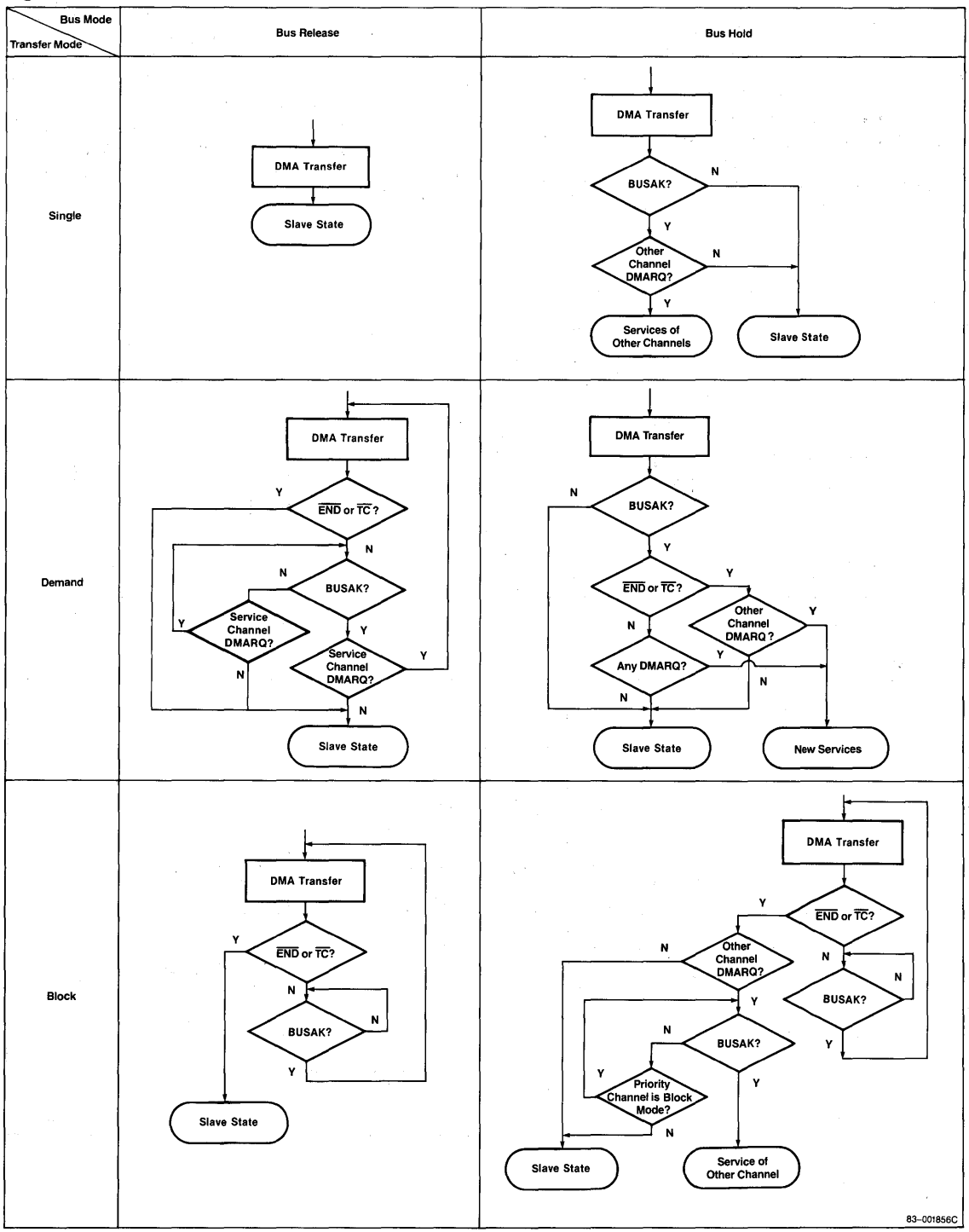
Autoinitialize

When the DMD register selects autoinitialize for a channel, the DMAU automatically reinitializes the address and count registers when $\overline{END}$ is asserted or the terminal count condition is reached. The contents of the base address and base count registers are transferred to the current address and current count registers, and the applicable bit of the mask register remains cleared.

Channel Priority

Each of the four DMAU channels is assigned a priority. When multiple DMA requests from several channels occur simultaneously, the channel with the highest priority will be serviced first. The DDC register selects one of two priority schemes: fixed or rotating (figure 35). In fixed priority, channel 0 is assigned the highest priority and channel 3, the lowest. In rotating priority, priority order is rotated after each service so that the channel last serviced receives the lowest priority. This method prevents the exclusive servicing of higher-priority channels and the lockout of lower-priority DMA channels.

Figure 34. Transfer Modes



3c

Cascade Connection

Slave μPD71071 DMA Controllers can be cascaded to easily expand the system DMA channel capacity to 16 DMA channels. Figure 36 shows an example of cascade connection. During cascade operation, the DMAU acts as a mediator between the BAU and the slave μPD71071s. During DMA cascade mode operation, it is the responsibility of external logic to isolate the cascade bus master from the μPD70208 control outputs. These outputs are listed in a table at the front of this data sheet.

The DMAU always operates in the bus hold mode while a cascade channel is in service, even when the bus release mode is programmed. Other DMA requests are held pending while a slave μPD71071 channel is in service. When the cascaded μPD71071 ends service and moves into the slave state, the DMAU also moves to the slave state and releases the bus. At this time, all bits of the DMAU request register are cleared. The DMAU continues to operate normally with the other noncascaded channels.

Figure 35. Priority Order

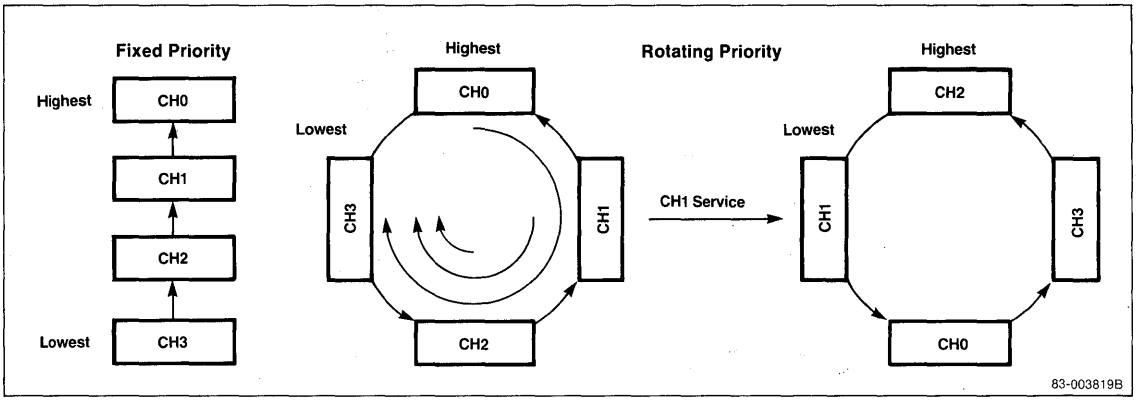
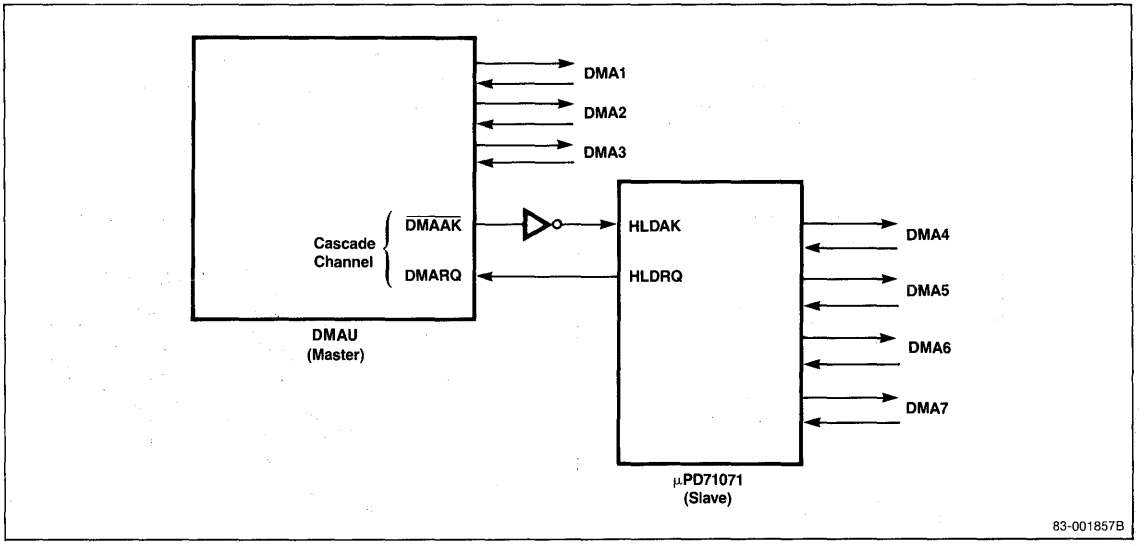


Figure 36. μPD71071 Cascade Example



Bus Waiting Operation

The DMAU will automatically perform a bus waiting operation (figure 37) whenever the RCU refresh request queue fills. When the DMA bus acknowledge goes inactive, the DMAU enters the bus waiting mode and inactivates the DMA bus request signal. Control of the bus is then transferred to the higher-priority RCU by the BAU.

Two clocks later, the DMAU reasserts its internal DMA bus request. The bus waiting mode is continued until the DMA bus acknowledge signal again becomes active and the interrupted DMA service is immediately restarted.

Programming the DMAU

To prepare a channel for DMA transfer, the following characteristics must be programmed.

- Starting address for the transfer
- Transfer count
- DMA operating mode
- Transfer size (byte/word)

The contents of the OPHA and DULA registers determine the base I/O port address of the DMAU. Addresses A_3-A_0 are used to select a particular register as follow:

A_3	A_2	A_1	A_0	Register	Operation
0	0	0	0	DICM	Write
0	0	0	1	DCH	Read/Write
0	0	1	0	DBC/DCC (low)	Read/Write
0	0	1	1	DBC/DCC (high)	Read/Write
0	1	0	0	DBA/DCA (low)	Read/Write
0	1	0	1	DBA/DCA (high)	Read/Write
0	1	1	0	DBA/DCA (upper)	Read/Write
0	1	1	1	Reserved	—
1	0	0	0	DDC (low)	Read/Write
1	0	0	1	DDC (high)	Read/Write
1	0	1	0	DMD	Read/Write
1	0	1	1	DST	Read
1	1	0	0	Reserved	—
1	1	0	1	Reserved	—
1	1	1	0	Reserved	—
1	1	1	1	DMK	Read/Write

Word I/O instructions can be used to read/write the register pairs listed below. All other registers are accessed via byte I/O instructions.

DBC/DCC
DBA/DCA (higher/lower only)
DDC

DMAU Registers

Initialize. The DMA initialize command (DICM) register (figure 38) is used to perform a software reset of the DMAU. The DICM is accessed using the byte OUT instruction.

Channel Register. Writes to the DMA channel (DCH) register (figure 39) select one of the four DMA channels for programming and also the base/current registers. Reads of the DCH register return the currently-selected channel and the register access mode.

Count Registers. When bit 2 of the DCH register is cleared, a write to the DMA count register updates both the DMA base count (DBC) and the DMA current count (DCC) registers with a new count. If bit 2 of the DCH register is set, a write to the DMA count register affects only the DBC register. The DBC register holds the initial count value until a new count is specified. If autoinitialization is enabled, this value is transferred to the DCC register when a terminal count or $\overline{END}$ condition occurs. For each DMA transfer, the current count register is decremented by one. The format of the DMA count register is shown below. The count value loaded into the DBC/DCC registers is one less than the desired transfer count.

2H, IN/OUT

7							0
C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

3H, IN/OUT

7							0
C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8

Address Register. Use either byte or word I/O instructions with the lower two bytes (4H and 5H) of the DMA address register. However, byte I/O instructions must be used to access the high-order byte (6H) of this register. When bit 2 of the channel register is cleared, a write to the DMA address register updates both the DMA base address (DBA) and the DMA current address (DCA) registers with the new address. If bit 2 of the DCH register is set, a write to the DMA address register affects only the DBA register.

4H, IN/OUT

7							0
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0

5H, IN/OUT

7							0
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

6H, IN/OUT (Byte only)

7							0
—	—	—	—	A_{19}	A_{18}	A_{17}	A_{16}

The DBA register holds the starting address value until a new address is specified. This value is transferred to the DCA register automatically if autoinitialization is selected. For each DMA transfer, the current address register is incremented/decremented by one.

Device Control Register. The DMA device control (DDC) register (figure 40) is used to to program the DMA transfer characteristics common to all DMA channels. It controls the bus mode, write timing, priority logic, and enable/disable of the DMAU.

Status Register. The DMA status (DST) register (figure 41) contains information about the current state of each DMA channel. Software can determine if a termination condition has been reached (TC_3-TC_0) or if a DMA service request is present (RQ_3-RQ_0). The byte IN instruction must be used to read this register.

Figure 37. Bus Waiting Operation

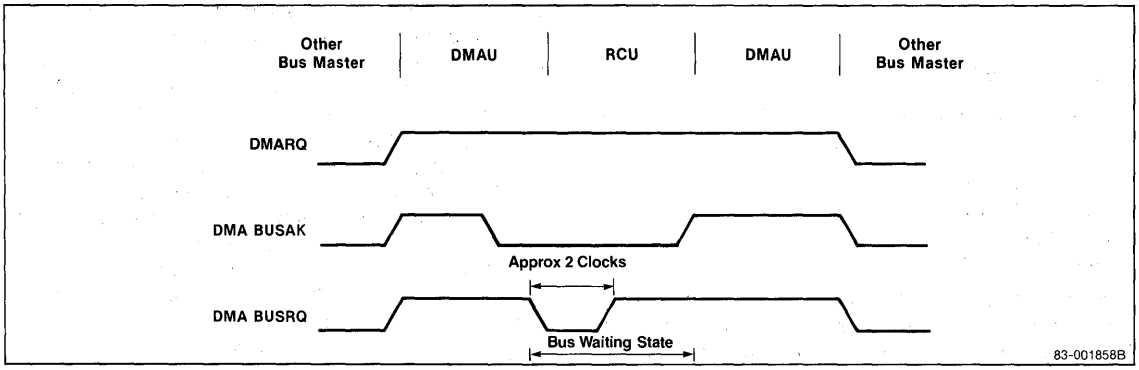
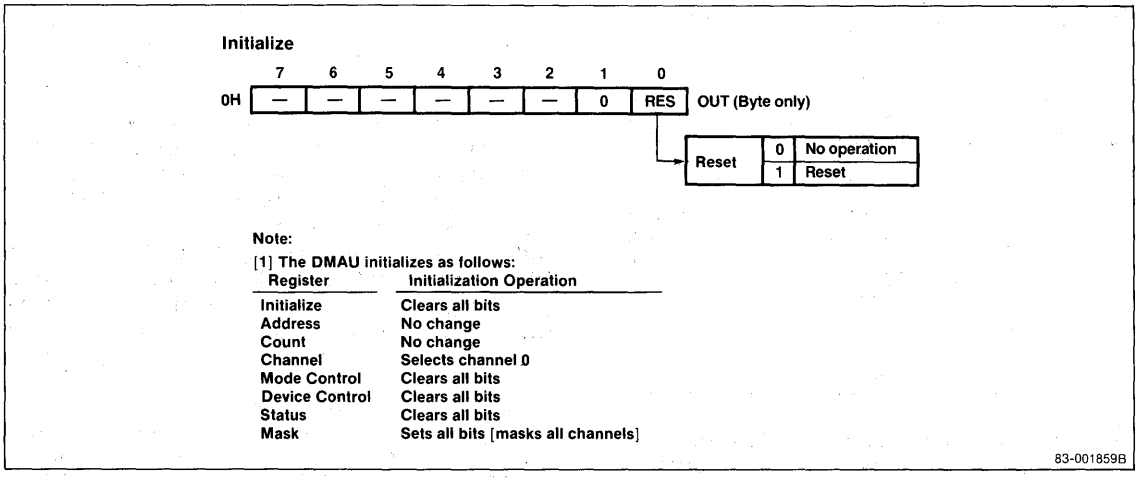


Figure 38. DMA Initialize Command Register



Note:

[1] The DMAU initializes as follows:

Register	Initialization Operation
Initialize	Clears all bits
Address	No change
Count	No change
Channel	Selects channel 0
Mode Control	Clears all bits
Device Control	Clears all bits
Status	Clears all bits
Mask	Sets all bits [masks all channels]

Figure 39. DMA Channel Register

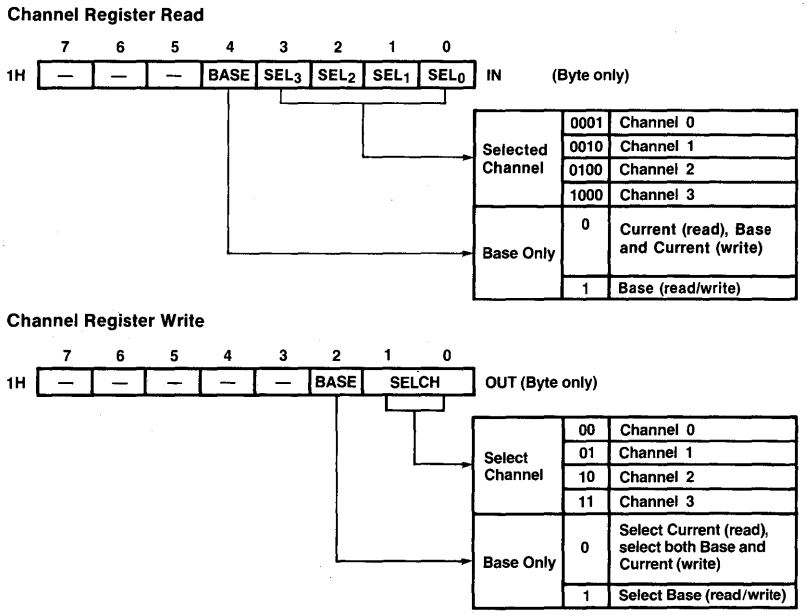
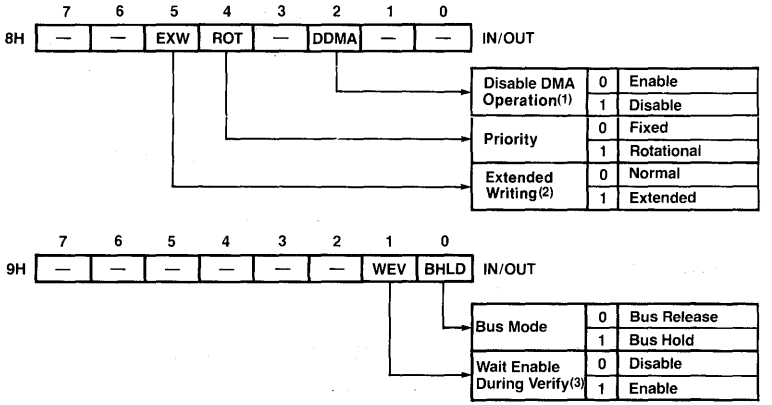


Figure 40. DMA Device Control Register



Note:

[1] Disables BUSRQ to the BAU to prevent incorrect DMA operation while the DMAU registers are being initialized or modified.

[2] When EXW is 0, the write signal becomes active [normal write] during T3 and TW [see timing waveforms]. When 1, the write signal becomes active during T2, T3, and TW [like the read signal].

[3] Wait states are generated by the READY signal during a verify transfer.

3c

Mode Control Register. The DMA mode (DMD) register (figure 42) selects the operating mode for each DMA channel. The DCH register selects which DMD register will be accessed. A byte IN/OUT instruction must be used to access this register.

Mask Register Read/Write. The DMA mask (DMK) register (figure 43) allows software to individually enable and disable DMA channels. The DMK register can only be accessed via byte I/O instructions.

Figure 41. DMA Status Register

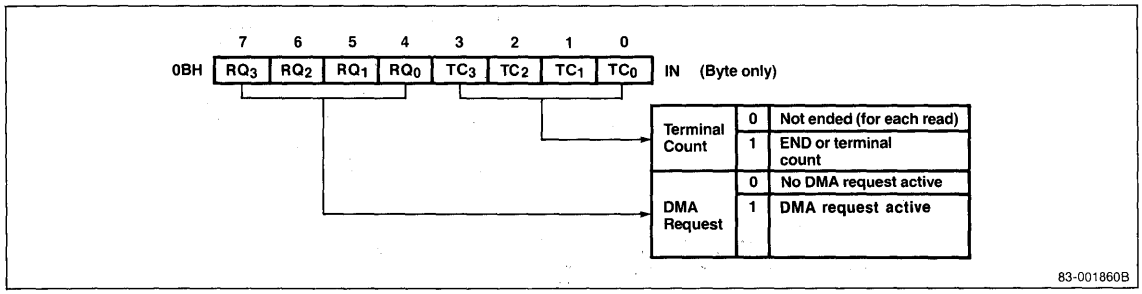


Figure 42. DMA Mode Register

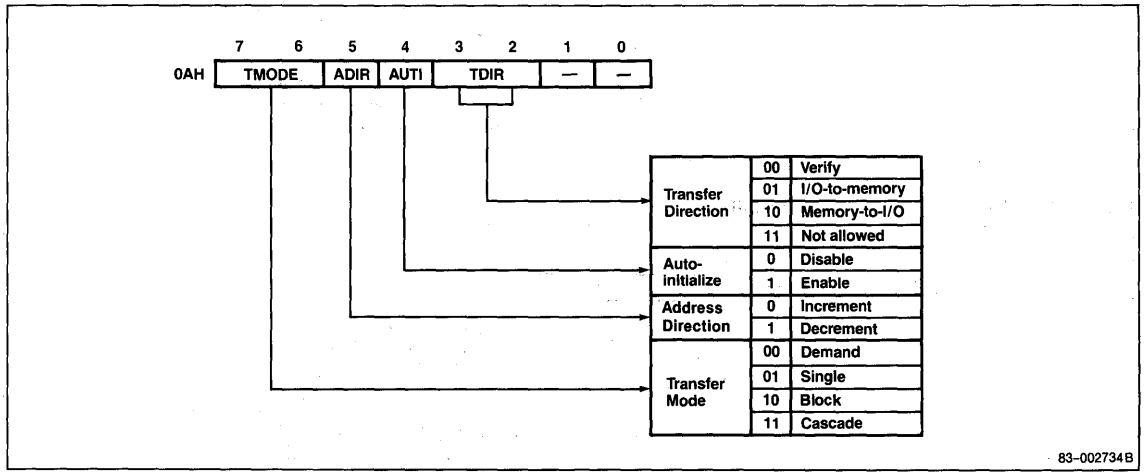
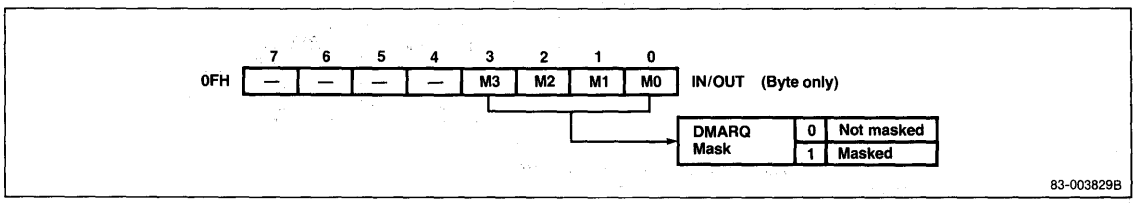


Figure 43. DMA Mask Register



Reset

The falling edge of the $\overline{RESET}$ signal resets the μPD70208. The signal must be held low for at least four clock cycles to be recognized as valid.

CPU Reset State Register	Reset Value
PFP	0000H
PC	0000H
PS	FFFFH
SS	0000H
DS0	0000H
DS1	0000H
PSW	F002H
AW, BW, CW, DW, IX, IY, BP, SP	Undefined
Instruction queue	Cleared

When RESET returns to the high level, the CPU will start fetching instructions from physical address FFFF0H.

Internal Peripheral Registers

Internal peripheral devices initialized on reset are listed in the following table. I/O devices not listed are not initialized on reset and must be initialized by software.

	Register	Reset Value
System I/O area	OPCN	----0000
	OPSEL	----0000
	WCY1	11111111
	WCY2	----1111
	WMB	-111-111
	TCKS	---00000
	RFC	x--01000
SCU	SMD	01001011
	SCM	--0000-0
	SIMK	------11
	SST	10000100
	DCH	---00001
	DMD	000000-0
DMAU	DDC (low)	--00-0--
	DDC (high)	------00
	DST	xxxx0000
	DMK	----1111

Symbols: x = unaffected; 0 = cleared; 1 = set; (-) = unused.

Output Pin Status

The following table lists output pin status during reset.

Signal	Status
$\overline{INTAK}$, $\overline{BUFEN}$, $BUF\overline{R}/W$, $\overline{MRD}$, $\overline{MWR}$, $\overline{END}/\overline{TC}$, $\overline{IOWR}$, $\overline{IORD}$, $\overline{REFRQ}$, BS_2-BS_0, $\overline{BUSLOCK}$, RESOUT, $\overline{DMAAK3}$-$\overline{DMAAK0}$	High level
QS_1-QS_0, ASTB, HLDAK	Low level
A_{19}-A_{16}/PS_3-PS_0, TOUT2	High or low level
A_{15}-A_8	High or low level
AD_7-AD_0	High impedance
CLKOUT	Continues to supply clock

3c

Instruction Set

Symbols

Preceding the instruction set, several tables explain symbols, abbreviations, and codes.

Clocks

In the Clocks column of the instruction set, the numbers cover these operations: instruction decoding, effective address calculation, operand fetch, and instruction execution.

Clock timings assume the instruction has been pre-fetched and is present in the four-byte instruction queue. Otherwise, add four clocks for each byte not present.

For instructions that reference memory operands, the number on the left side of the slash (/) is for byte operands and the number on the right side is for word operands.

For conditional control transfer or branch instructions, the number on the left side of the slash is applicable if the transfer or branch takes place. The number on the right side is applicable if it does not take place.

If a range of numbers is given, the execution time depends on the operands involved.

Symbols

Symbol	Meaning
acc	Accumulator (AW or AL)
disp	Displacement (8 or 16 bits)
dmem	Direct memory address
dst	Destination operand or address
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
far_label	Label within a different program segment
far_proc	Procedure within a different program segment
fp_op	Floating point instruction operation
imm	8- or 16-bit immediate operand
imm3/4	3- or 4-bit immediate bit offset
imm8	8-bit immediate operand
imm16	16-bit immediate operand
mem	Memory field (000 to 111); 8- or 16-bit memory location

Symbols

Symbol	Meaning
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
memptr16	Word containing the destination address within the current segment
memptr32	Double word containing a destination address in another segment
mod	Mode field (00 to 10)
near_label	Label within the current segment
near_proc	Procedure within the current segment
offset	Immediate offset data (16 bits)
pop_value	Number of bytes to discard from the stack
reg	Register field (000 to 111); 8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
regptr	16-bit register containing a destination address within the current segment
regptr16	Register containing a destination address within the current segment
seg	Immediate segment data (16 bits)
short_label	Label between −128 and +127 bytes from the end of the current instruction
sr	Segment register
src	Source operand or address
temp	Temporary register (8/16/32 bits)
AC	Auxiliary carry flag
AH	Accumulator (high byte)
AL	Accumulator (low byte)
AW	Accumulator (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
BP	BP register
BRK	Break flag
BW	BW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
CW	CW register (16 bits)
CY	Carry flag
DH	DW register (high byte)
DIR	Direction flag
DL	DW register (low byte)

Symbols (cont)

Symbol	Meaning
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DW	DW register (16 bits)
IE	Interrupt enable flag
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
MD	Mode flag
P	Parity flag
PC	Program counter (16 bits)
PS	Program segment register (16 bits)
PSW	Program status word (16 bits)
R	Register set
S	Sign extend operand field S = 0 No sign extension S = 1 Sign extend immediate byte operand
S	Sign flag
SP	Stack pointer (16 bits)
SS	Stack segment register (16 bits)
V	Overflow flag
W	Word/byte field (0 to 1)
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value
Z	Zero flag

Flag Operations

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to result
u	Undefined
R	Restored to previous state

Memory Addressing Modes

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Register Selection (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Register Selection

sr	Segment Register
00	DS1
01	PS
10	SS
11	DS0

Instruction Set

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer Instructions																									
MOV	reg, reg	1	0	0	0	1	0	1	W	1	1	reg			reg			2	2						
	mem, reg	1	0	0	0	1	0	0	W	mod		reg			mem			7/11	2-4						
	reg, mem	1	0	0	0	1	0	1	W	mod		reg			mem			10/14	2-4						
	mem, imm	1	1	0	0	0	1	1	W	mod		reg			mem			9/13	3-6						
	reg, imm	1	0	1	1	W	reg											4	2-3						
	acc, dmem	1	0	1	0	0	0	0	W									10/14	3						
	dmem, acc	1	0	1	0	0	0	1	W									9/13	3						
	sr, reg16	1	0	0	0	1	1	1	0	1	1	0	sr		reg			2	2						
	sr, mem16	1	0	0	0	1	1	1	0	mod		0	sr		mem			14	2-4						
	reg16, sr	1	0	0	0	1	1	0	0	1	1	0	sr		reg			2	2						
	mem16, sr	1	0	0	0	1	1	0	0	mod		0	sr		mem			12	2-4						
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod		reg			mem			25	2-4						
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod		reg			mem			25	2-4						
	AH, PSW	1	0	0	1	1	1	1	1									2	1						
	PSW, AH	1	0	0	1	1	1	1	0									3	1	x	x		x	x	x
LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod		reg			mem			4	2-4						
TRANS	src_table	1	1	0	1	0	1	1	1									9	1						
XCH	reg, reg	1	0	0	0	0	1	1	W	1	1	reg			reg			3	2						
	mem, reg	1	0	0	0	0	1	1	W	mod		reg			mem			13/21	2-4						
	AW, reg16	1	0	0	1	0	reg											3	1						
Repeat Prefixes																									
REPC		0	1	1	0	0	1	0	1									2	1						
REPNC		0	1	1	0	0	1	0	0									2	1						
REP REPE REPZ		1	1	1	1	0	0	1	1									2	1						
REPNE REPNZ		1	1	1	1	0	0	1	0									2	1						
Block Transfer Instructions																									
MOVBK	dst, src	1	0	1	0	0	1	0	W									1 9 (9) + 8n (W = 0) 9 (17) + 16n (W = 1)							
CMPBK	dst, src	1	0	1	0	0	1	1	W									1 7 (13) + 14n (W = 0) 7 (21) + 22n (W = 1)		x	x	x	x	x	x
CMPM	dst	1	0	1	0	1	1	1	W									1 7 (7) + 10n (W = 0) 7 (11) + 14n (W = 1)		x	x	x	x	x	x
LDM	src	1	0	1	0	1	1	0	W									1 7 (7) + 9n (W = 0) 7 (11) + 13n (W = 1)							
STM	dst	1	0	1	0	1	0	1	W									1 5 (5) + 4n (W = 0) 5 (9) + 8n (W = 1)							

n = number of transfers
String instruction execution clocks for a single instruction execution are in parentheses.

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
I/O Instructions																									
IN	acc, imm8	1	1	1	0	0	1	0	W									9/13	2						
	acc, DW	1	1	1	0	1	1	0	W									8/12	1						
OUT	imm8, acc	1	1	1	0	0	1	1	W									8/12	2						
	DW, acc	1	1	1	0	1	1	1	W									8/12	1						
INM	dst, DW	0	1	1	0	1	1	0	W									9 (10) + 8n (W = 0) 9 (18) + 16n (W = 1)	1						
OUTM	DW, src	0	1	1	0	1	1	1	W									9 (10) + 8n (W = 0) 9 (18) + 16n (W = 1)	1						
		n = number of transfers String instruction execution clocks for a single instruction execution are in parentheses.																							
BCD Instructions																									
ADJBA		0	0	1	1	0	1	1	1									7	1	x	x	u	u	u	u
ADJ4A		0	0	1	0	0	1	1	1									3	1	x	x	u	x	x	x
ADJBS		0	0	1	1	1	1	1	1									7	1	x	x	u	u	u	u
ADJ4S		0	0	1	0	1	1	1	1									3	1	x	x	u	x	x	x
ADD4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	7 + 19n	2	u	x	u	u	u	x
SUB4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	7 + 19n	2	u	x	u	u	u	x
CMP4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	7 + 19n	2	u	x	u	u	u	x
ROL4	reg8	0 1	0 1	0 0	0 0	1 0	1 reg	1	1	0	0	1	0	1	0	0	0	13	3						
	mem8	0 mod	0 0	0 0	0 0	1 0	1 mem	1	1	0	0	1	0	1	0	0	0	25	3-5						
ROR4	reg8	0 1	0 1	0 0	0 0	1 0	1 reg	1	1	0	0	1	0	1	0	1	0	17	3						
	mem8	0 mod	0 0	0 0	0 0	1 0	1 mem	1	1	0	0	1	0	1	0	1	0	29	3-5						
		n = number of BCD digits divided by 2																							
Data Type Conversion Instructions																									
CVTBD		1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	15	2	u	u	u	x	x	x
CVTDB		1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	7	2	u	u	u	x	x	x
CVTBW		1	0	0	1	1	0	0	0									2	1						
CVTWL		1	0	0	1	1	0	0	1									4/5	1						
Arithmetic Instructions																									
ADD	reg, reg	0	0	0	0	0	0	1	W	1	1	reg		reg				2	2	x	x	x	x	x	x
	mem, reg	0	0	0	0	0	0	0	W	mod		reg		mem				13/21	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	0	0	0	1	W	mod		reg		mem				10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	0	0	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	0	0	mem			15/23	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	0	0	1	0	W									4	2-3	x	x	x	x	x	x

3c

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Arithmetic Instructions (cont)																									
ADDC	reg, reg	0	0	0	1	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	0	0	0	W	mod			reg			mem		13/21	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	0	0	1	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	0		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	0		mem		15/23	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	0	1	0	W									4	2-3	x	x	x	x	x	x
SUB	reg, reg	0	0	1	0	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	1	0	1	0	0	W	mod			reg			mem		13/21	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	0	1	0	1	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	0	1		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	0	1		mem		15/23	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	0	1	1	0	W									4	2-3	x	x	x	x	x	x
SUBC	reg, reg	0	0	0	1	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	1	0	0	W	mod			reg			mem		13/21	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	1	0	1	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	1		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	1		mem		15/23	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	1	1	0	W									4	2-3	x	x	x	x	x	x
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	0		mem		13/21	2-4	x		x	x	x	x
	reg16	0	1	0	0	0		reg										2	1	x		x	x	x	x
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	1		mem		13/21	2-4	x		x	x	x	x
	reg16	0	1	0	0	1		reg										2	1	x		x	x	x	x
MULU	reg	1	1	1	1	0	1	1	W	1	1	1	0	0		reg		21-30	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	0		mem		26-39	2-4	u	x	x	u	u	u
MUL	reg	1	1	1	1	0	1	1	W	1	1	1	0	1		reg		33-47	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	1		mem		38-56	2-4	u	x	x	u	u	u
	reg16,reg16,imm8	0	1	1	0	1	0	1	1	1	1		reg			reg		28-34	3	u	x	x	u	u	u
	reg16,mem16,imm8	0	1	1	0	1	0	1	1	mod			reg			mem		37-43	3-5	u	x	x	u	u	u
	reg16,reg16,imm16	0	1	1	0	1	0	0	1	1	1		reg			reg		36-42	4	u	x	x	u	u	u
	reg16,mem16,imm16	0	1	1	0	1	0	0	1	mod			reg			mem		45-51	4-6	u	x	x	u	u	u
DIVU	reg	1	1	1	1	0	1	1	W	1	1	1	1	0		reg		19-25	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	0		mem		24-34	2-4	u	u	u	u	u	u
DIV	reg	1	1	1	1	0	1	1	W	1	1	1	1	1		reg		29-43	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	1		mem		34-52	2-4	u	u	u	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Comparison Instructions																									
CMP	reg, reg	0	0	1	1	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	1	1	1	0	0	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	1	1	0	1	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	1	1		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	1	1		mem		12/16	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	1	1	1	0	W									4	2-3	x	x	x	x	x	x
Logical Instructions																									
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0		reg		2	2						
	mem	1	1	1	1	0	1	1	W	mod		0	1	0		mem		13/21	2-4						
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1		reg		2	2	x	x	x	x	x	x
	mem	1	1	1	1	0	1	1	W	mod		0	1	1		mem		13/21	2-4	x	x	x	x	x	x
TEST	reg, reg	1	0	0	0	0	1	0	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	1	0	0	0	0	1	0	W	mod			reg			mem		9/13	2-4	u	0	0	x	x	x
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0		reg		4	3-4	u	0	0	x	x	x
	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0		mem		10/14	3-6	u	0	0	x	x	x
	acc, imm	1	0	1	0	1	0	0	W									4	2-3	u	0	0	x	x	x
AND	reg, reg	0	0	1	0	0	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	1	0	0	0	0	W	mod			reg			mem		13/21	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	0	0	0	1	W	mod			reg			mem		10/14	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	0	0		reg		4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0		mem		15/23	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	0	0	1	0	W									4	2-3	u	0	0	x	x	x
OR	reg, reg	0	0	0	0	1	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	0	0	1	0	0	W	mod			reg			mem		13/21	2-4	u	0	0	x	x	x
	reg, mem	0	0	0	0	1	0	1	W	mod			reg			mem		10/14	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	0	0	1		reg		4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1		mem		15/23	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									4	2-3	u	0	0	x	x	x
XOR	reg, reg	0	0	1	1	0	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	1	1	0	0	0	W	mod			reg			mem		13/21	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	1	0	0	1	W	mod			reg			mem		10/14	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	1	0		reg		4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0		mem		15/23	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	1	0	1	0	W									4	2-3	u	0	0	x	x	x

3c

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Bit Manipulation Instructions																									
INS	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	35-133	3						
		1	1		reg			reg																	
	reg8, imm8	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	35-133	4						
		1	1	0	0	0		reg																	
EXT	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	34-59	3						
		1	1		reg			reg																	
	reg8, imm8	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	34-59	4						
		1	1	0	0	0		reg																	
TEST1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	3	3	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	7/11	3-5	u	0	0	u	u	x
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	4	4	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	8/12	4-6	u	0	0	u	u	x
		mod		0	0	0		mem																	
SET1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	10/18	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	5	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	11/19	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	1									2	1		1				
	DIR	1	1	1	1	1	1	0	1									2	1						
CLR1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	5	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	11/19	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	6	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	12/20	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	0									2	1		0				
	DIR	1	1	1	1	1	1	0	0									2	1						
NOT1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	10/18	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	5	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	11/19	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	0	1	0	1									2	1		x				

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions																									
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0		reg		2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0		mem		13/21	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0		reg		7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0		mem		16/24 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0		reg		7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0		mem		16/24 + n	3-5	u	x	u	x	x	x
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1		reg		2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1		mem		13/21	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1		reg		7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1		mem		16/24 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1		reg		7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1		mem		16/24 + n	3-5	u	x	u	x	x	x
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1		reg		2	2	u	x	0	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1		mem		13/21	2-4	u	x	0	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1		reg		7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1		mem		16/24 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1		reg		7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1		mem		16/24 + n	3-5	u	x	u	x	x	x
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	0		mem		13/21	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	0		mem		16/24 + n	2-4		x	u			
	reg, imm	1	1	0	0	0	0	0	W	1	1	0	0	0		reg		7 + n	3		x	u			
	mem, imm	1	1	0	0	0	0	0	W	mod		0	0	0		mem		16/24 + n	3-5		x	u			
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1		reg		2	2		x	u			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	1		mem		13/21	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	1		mem		16/24 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	1		mem		16/24 + n	3-5		x	u			
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0		mem		13/21	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0		mem		16/24 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0		mem		16/24 + n	3-5		x	u			

n = number of shifts

3c

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift Rotate Instructions (cont)																									
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1	mem			13/21	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1	mem			16/24 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1	reg			7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1	mem			16/24 + n	3-5		x	u			
												n = number of shifts													
Stack Manipulation Instructions																									
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0	mem			23	2-4						
	reg16	0	1	0	1	0	reg											10	1						
	sr	0	0	0	sr		1	1	0									10	1						
	PSW	1	0	0	1	1	1	0	0									10	1						
	R	0	1	1	0	0	0	0	0									65	1						
	imm	0	1	1	0	1	0	S	0									9-10	2-3						
POP	mem16	1	0	0	0	1	1	1	1	mod		0	0	0	mem			25	2-4						
	reg16	0	1	0	1	1	reg											12	1						
	sr	0	0	0	sr		1	1	1									12	1						
	PSW	1	0	0	1	1	1	0	1									12	1	R	R	R	R	R	R
	R	0	1	1	0	0	0	0	1									75	1						
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0									*	4						
										*imm8 = 0 : 16 imm8 > 1 : 21 + 16 (imm8 − 1)														
DISPOSE		1	1	0	0	1	0	0	1									10	1						
Control Transfer Instructions																									
CALL	near_proc	1	1	1	0	1	0	0	0									20	3						
	regptr	1	1	1	1	1	1	1	1	1	1	0	1	0	reg			18	1						
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0	mem			31	2-4						
	far_proc	1	0	0	1	1	0	1	0									29	5						
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1	mem			47	2-4						
RET		1	1	0	0	0	0	1	1									19	1						
	pop_value	1	1	0	0	0	0	1	0									24	3						
		1	1	0	0	1	0	1	1									29	1						
	pop_value	1	1	0	0	1	0	1	0									32	3						
BR	near_label	1	1	1	0	1	0	0	1									13	3						
	short_label	1	1	1	0	1	0	1	1									12	2						
	reg	1	1	1	1	1	1	1	1	1	1	1	0	0	reg			11	2						
	memptr16	1	1	1	1	1	1	1	1	mod		1	0	0	mem			23	2-4						
	far_label	1	1	1	0	1	0	1	0									15	5						
	memptr32	1	1	1	1	1	1	1	1	mod		1	0	1	mem			34	2-4						
BV	near_label	0	1	1	1	0	0	0	0									14/4	2						
BNV	near_label	0	1	1	1	0	0	0	1									14/4	2						

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Control Transfer Instructions (cont)																									
BC, BL	near_label	0	1	1	1	0	0	1	0									14/4	2						
BNC, BNL	near_label	0	1	1	1	0	0	1	1									14/4	2						
BE, BZ	near_label	0	1	1	1	0	1	0	0									14/4	2						
BNE, BNZ	near_label	0	1	1	1	0	1	0	1									14/4	2						
BNH	near_label	0	1	1	1	0	1	1	0									14/4	2						
BH	near_label	0	1	1	1	0	1	1	1									14/4	2						
BN	near_label	0	1	1	1	1	0	0	0									14/4	2						
BP	near_label	0	1	1	1	1	0	0	1									14/4	2						
BPE	near_label	0	1	1	1	1	0	1	0									14/4	2						
BPO	near_label	0	1	1	1	1	0	1	1									14/4	2						
BLT	near_label	0	1	1	1	1	1	0	0									14/4	2						
BGE	near_label	0	1	1	1	1	1	0	1									14/4	2						
BLE	near_label	0	1	1	1	1	1	1	0									14/4	2						
BGT	near_label	0	1	1	1	1	1	1	1									14/4	2						
DBNZNE	near_label	1	1	1	0	0	0	0	0									14/5	2						
DBNZE	near_label	1	1	1	0	0	0	0	1									14/5	2						
DBNZ	near_label	1	1	1	0	0	0	1	0									13/5	2						
BCWZ	near_label	1	1	1	0	0	0	1	1									13/5	2						
Interrupt Instructions																									
BRK	3	1	1	0	0	1	1	0	0									50	1						
	imm8	1	1	0	0	1	1	0	1									50	2						
BRKV	imm8	1	1	0	0	1	1	1	0									52/3	1						
RETI		1	1	0	0	1	1	1	1									39	1	R	R	R	R	R	R
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod		reg			mem			25/72-75	2-4						
BRKEM	imm8	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	50	3						
CPU Control Instructions																									
HALT		1	1	1	1	0	1	0	0									2	1						
BUSLOCK		1	1	1	1	0	0	0	0									2	1						
FPO1	fp_op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y	mem			14	2-4						
FPO2	fp_op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y	mem			14	2-4						
POLL		1	0	0	1	1	0	1	1	n = number of times POLL pin is sampled.								2 + 5n	1						
NOP		1	0	0	1	0	0	0	0									3	1						
DI		1	1	1	1	1	0	1	0									2	1						
EI		1	1	1	1	1	0	1	1									2	1						
DS0:, DS1:, PS:, and SS: (segment override prefixes)		0	0	1	seg		1	1	0									2	1						
8080 Instruction Set Enhancements																									
RETEM		1	1	1	0		1	1	0	1	1	1	1	1	1	0	1	39	2	R	R	R	R	R	R
CALLN	imm8	1	1	1	0		1	1	0	1	1	1	1	0	1	1	0	1 58	3						

3c

μPD70216 (V50) 16-Bit Microprocessor: High-Integration, CMOS

Description

The μPD70216 (V50™) is a high-performance, low-power 16-bit microprocessor integrating a number of commonly-used peripherals to dramatically reduce the size of microprocessor systems. The CMOS construction makes the μPD70216 ideal for the design of portable computers, instrumentation, and process control equipment.

The μPD70216 contains a powerful instruction set that is compatible with the μPD70108/μPD70116 (V20®/V30®) and μPD8086/μPD8088 instruction sets. Instruction set support includes a wide range of arithmetic, logical, and control operations as well as bit manipulation, BCD arithmetic, and high-speed block transfer instructions. The μPD70216 can also execute the entire μPD8080AF instruction set using the 8080 emulation mode. Also available is the μPD70208 (V40™), identical to the μPD70216 but with an 8-bit external data bus.

Features

- ☐ Low-power CMOS technology
- ☐ V20/V30 instruction set compatible
- ☐ Minimum instruction execution time: 250 ns (at 8 MHz)
- ☐ Direct addressing of 1M bytes of memory
- ☐ Powerful set of addressing modes
- ☐ Fourteen 16-bit CPU registers
- ☐ On-chip peripherals including
  - — Clock generator
  - — Bus interface
  - — Bus arbitration
  - — Programmable wait state generator
  - — DRAM refresh controller
  - — Three 16-bit timer/counters
  - — Asynchronous serial I/O controller
  - — Eight-input interrupt controller
  - — Four-channel DMA controller
- ☐ Hardware effective address calculation logic
- ☐ Maskable and nonmaskable interrupt inputs
- ☐ IEEE 796 compatible bus interface
- ☐ Low-power standby mode

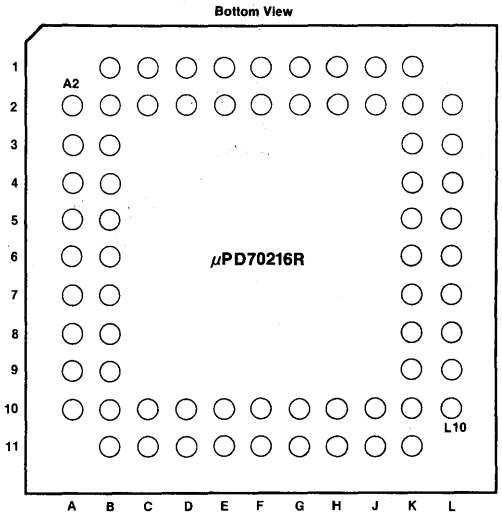
V20 and V30 are registered trademarks of NEC Corporation.
V40 and V50 are trademarks of NEC Corporation.

Ordering Information

Part Number	Max Frequency (MHz)	Package
μPD70216R-8	8	68-pin ceramic PGA
R-10	10	
L-8	8	68-pin PLCC
L-10	10	
GF-8	8	80-pin plastic QFP
GF-10	10	

Pin Configurations

68-Pin Ceramic PGA

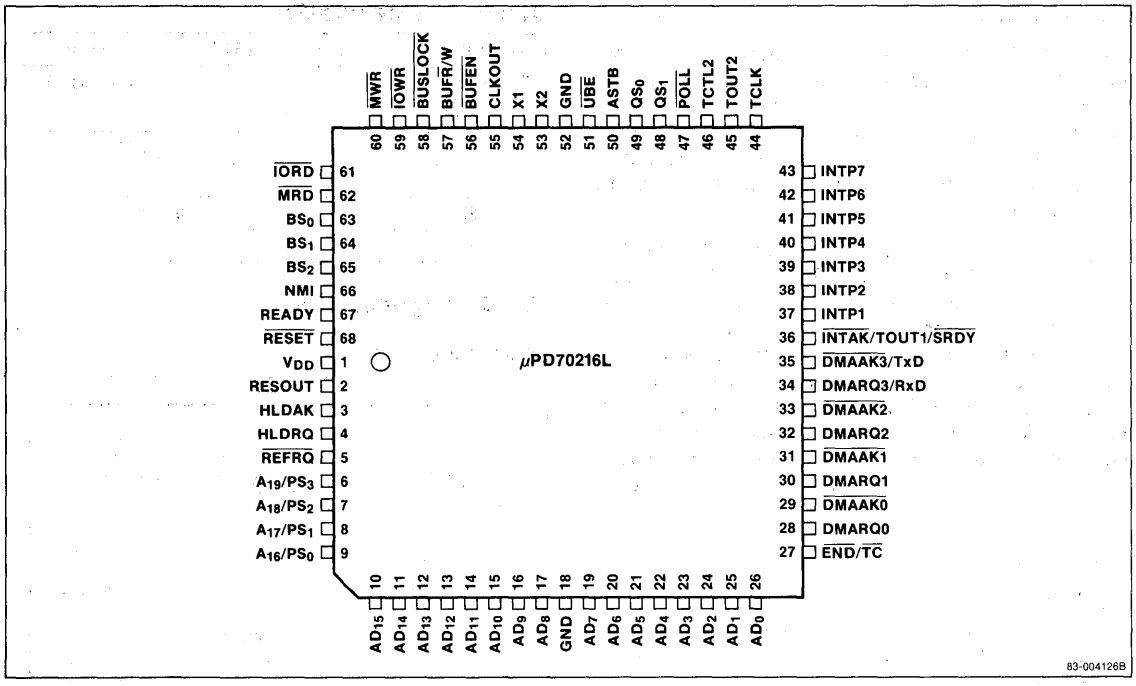


Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
A2	INTP7	B9	DMARQ1	F10	AD_7	K4	NMI
A3	INTP5	B10	DMARQ0	F11	GND	K5	$\overline{RESET}$
A4	INTP3	B11	AD_0	G1	X1	K6	RESOUT
A5	INTP1	C1	TCTL2	G2	CLKOUT	K7	HLDRQ
A6	$\overline{DMAAK3}$/TxD	C2	$\overline{POLL}$	G10	AD_8	K8	A_{19}/PS_3
A7	$\overline{DMAAK2}$	C10	AD_1	G11	AD_9	K9	A_{17}/PS_1
A8	$\overline{DMAAK1}$	C11	AD_2	H1	$\overline{BUFEN}$	K10	AD_{14}
A9	$\overline{DMAAK0}$	D1	QS_1	H2	$\overline{BUFR}$/W	K11	AD_{15}
A10	$\overline{END}/\overline{TC}$	D2	QS_0	H10	AD_{10}	L2	$\overline{IORD}$
B1	TCLK	D10	AD_3	H11	AD_{11}	L3	BS_0
B2	TOUT2	D11	AD_4	J1	$\overline{BUSLOCK}$	L4	BS_2
B3	INTP6	E1	ASTB	J2	$\overline{IOWR}$	L5	READY
B4	INTP4	E2	$\overline{UBE}$	J10	AD_{12}	L6	V_{DD}
B5	INTP2	E10	AD_5	J11	AD_{13}	L7	HLDAK
B6	$\overline{INTAK}$/TOUT1/$\overline{SRDY}$	E11	AD_6	K1	$\overline{MWR}$	L8	$\overline{REFRQ}$
B7	DMARQ3/RxD	F1	GND	K2	$\overline{MRD}$	L9	A_{18}/PS_2
B8	DMARQ2	F2	X2	K3	BS_1	L10	A_{16}/PS_0

83-002719B

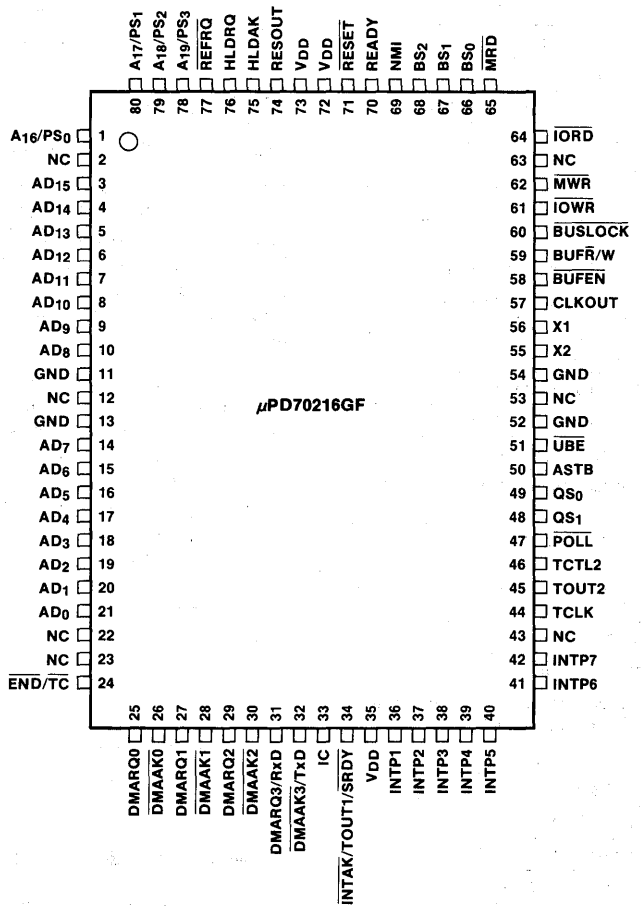
Pin Configurations (cont)

68-Pin Plastic Leaded Chip Carrier (PLCC)



Pin Configurations (cont)

80-Pin Plastic QFP



3d

Pin Identification

Symbol	Function
A_{19}-A_{16}/PS_3-PS_0	Multiplexed address/processor status outputs
AD_{15}-AD_0	Multiplexed address/data bus
ASTB	Address strobe output
BS_2-BS_0	Bus status outputs
$\overline{BUFEN}$	Data bus transceiver enable output
BUF$\overline{R}$/W	Data bus transceiver direction output
$\overline{BUSLOCK}$	Buslock output
CLKOUT	System clock output
$\overline{DMAAK0}$	DMA channel 0 acknowledge output
$\overline{DMAAK1}$	DMA channel 1 acknowledge output
$\overline{DMAAK2}$	DMA channel 2 acknowledge output
$\overline{DMAAK3}$/TxD	DMA channel 3 acknowledge output/Serial transmit data output
DMARQ0	DMA channel 0 request input
DMARQ1	DMA channel 1 request input
DMARQ2	DMA channel 2 request input
DMARQ3/RxD	DMA channel 3 request input/Serial receive data input
$\overline{END}$/$\overline{TC}$	End input/Terminal count output
GND	Ground
HLDAK	Hold acknowledge output
HLDRQ	Hold request input
IC	Internal connection; leave unconnected
$\overline{INTAK}$/TOUT1/$\overline{SRDY}$	Interrupt acknowledge output/Timer/counter 1 output/Serial ready output
INTP1-INTP7	Interrupt request inputs
$\overline{IORD}$	I/O read strobe output
$\overline{IOWR}$	I/O write strobe output
$\overline{MRD}$	Memory read strobe output
$\overline{MWR}$	Memory write strobe output
NC	No connection
NMI	Nonmaskable interrupt input
$\overline{POLL}$	Poll input
QS_1-QS_0	CPU queue status outputs
READY	Ready input
$\overline{REFRQ}$	Refresh request output
$\overline{RESET}$	Reset input
RESOUT	Synchronized reset output
TCLK	Timer/counter external clock input
TCTL2	Timer/counter 2 control input
TOUT2	Timer/counter 2 output
$\overline{UBE}$	Upper byte enable output
V_{DD}	+5 V power supply input
X1, X2	Crystal/external clock inputs

Pin Functions

A_{19}-A_{16}/PS_3-PS_0 [Address/Status Bus]

These three-state output pins contain the upper 4 bits of the 20-bit address during T1 and processor status information during the T2, T3, TW, and T4 states of a bus cycle. During T1 of a memory read or write cycle, these pins contain the upper 4 bits of the 20-bit address. These pins are forced low during T1 of an I/O bus cycle.

Processor status is output during T2, T3, TW, and T4 of both memory and I/O bus cycles. PS_3 is zero during any CPU native mode bus cycle. During any DMA, refresh, or 8080 emulation mode bus cycle, PS_3 outputs a high level. PS_2 outputs the contents of the interrupt enable (IE) flag in the CPU PSW register. PS_1 and PS_0 indicate the segment register used to form the physical address of a CPU bus cycle as follows:

PS_1	PS_0	Segment
0	0	Data segment 1 (DS1)
0	1	Stack segment (SS)
1	0	Program segment (PS)
1	1	Data segment 0 (DS0)

These pins are in the high-impedance state during hold acknowledge.

AD_{15}-AD_0 [Address/Data Bus]

These three-state pins form the middle byte of the active-high, time-multiplexed address/data bus. During T1 of a bus cycle, AD_{15}-AD_0 output the lower 16 bits of the 20-bit memory or I/O address. During the T2, T3, TW, and T4 states, AD_{15}-AD_0 form the 16-bit bi-directional data bus.

The memory and I/O address spaces are organized into a pair of byte-wide banks. The even bank is accessed whenever $AD_0 = 0$ during T1 of a bus cycle. Access to the odd bank is controlled by the $\overline{UBE}$ pin.

The AD_{15}-AD_0 pins enter the high-impedance state during hold acknowledge or internal interrupt acknowledge bus cycles or while $\overline{RESET}$ is asserted. Pins AD_{10}-AD_8 contain the slave address of an external interrupt controller during the second interrupt acknowledge bus cycle.

ASTB [Address Strobe]

This active-high output is used to latch the address from the multiplexed address bus in an external address latch during T1 of a bus cycle. ASTB is held at a low level during hold acknowledge.

BS_2-BS_0 [Bus Status]

Outputs BS_2-BS_0 indicate the type of bus cycle being performed as shown below. BS_2-BS_0 become active during the state preceding T1 and return to the passive state during the bus state preceding T4.

BS_2	BS_1	BS_0	Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Instruction fetch
1	0	1	Memory read (1)
1	1	0	Memory write (2)
1	1	1	Passive state

Note:

(1) Memory read bus cycles include CPU, DMA read, DMA verify, and refresh bus cycles.

(2) Memory write bus cycles include CPU and DMA write bus cycles.

BS_2-BS_0 are three-state outputs and are high impedance during hold acknowledge.

$\overline{BUFEN}$ [Buffer Enable]

$\overline{BUFEN}$ is an active-low output for enabling an external data bus transceiver during a bus cycle. $\overline{BUFEN}$ is asserted during T2 through T3 of a read cycle, T2 through T3 of a slave interrupt acknowledge cycle, and T1 through T4 of a write cycle. $\overline{BUFEN}$ is not asserted when the bus cycle corresponds to an internal peripheral, DMA, refresh, or internal interrupt acknowledge cycle. $\overline{BUFEN}$ enters the high-impedance state during hold acknowledge.

BUF$\overline{R}$/W [Buffer Read/Write]

BUF$\overline{R}$/W is a three-state, active-low output used to control the direction of an external data bus transceiver during CPU bus cycles. A high level indicates the μPD70216 will perform a write cycle and a low level indicates a read cycle. BUFR/W enters the high-impedance state during hold acknowledge.

$\overline{BUSLOCK}$

This active-low output provides a means for the CPU to indicate to an external bus arbiter that the bus cycles of the next instruction are to be kept contiguous. $\overline{BUSLOCK}$ is asserted for the duration of the instruction following the BUSLOCK prefix. $\overline{BUSLOCK}$ is also asserted during interrupt acknowledge cycles and enters the high-impedance state during hold acknowledge. While $\overline{BUSLOCK}$ is asserted, DMAU, RCU, and external bus requests are ignored.

CLKOUT

CLKOUT is a buffered clock output used as a reference for all timing. CLKOUT has a 50-percent duty cycle at half the frequency of the input clock source.

$\overline{DMAAK0}$-$\overline{DMAAK2}$ [DMA Acknowledge]

This set of outputs contains the DMA acknowledge signals for channels 0-2 from the internal DMA controller and indicate that the peripheral can perform the requested transfer.

$\overline{DMAAK3}$/TxD [DMA Acknowledge 3]/[Serial Transmit Data]

Two output signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- $\overline{DMAAK3}$ is an active-low output and enables an external DMA peripheral to perform the requested DMA transfer for channel 3.
- TxD is the serial data output from the serial control unit.

DMARQ0-DMARQ2 [DMA Request]

These synchronized inputs are used by external peripherals to request DMA service for channels 0-2 from the internal DMA controller.

DMARQ3/RxD [DMA Request 3]/[Serial Receive Data]

Two input signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- DMARQ3 is used by an external peripheral to request a DMA transfer cycle for channel 3.
- RxD is the serial data input to the serial control unit.

3d

$\overline{END}/\overline{TC}$ [End/Terminal Count]

This active-low bidirectional pin controls the termination of a DMA service. Assertion of $\overline{END}$ by external hardware during DMA service causes the service to terminate. When a DMA channel reaches its terminal count, the DMAU asserts $\overline{TC}$, indicating the programmed operation has completed.

$\overline{END}/\overline{TC}$ is an open-drain I/O pin, and requires an external 2.2-kΩ pull-up resistor.

HLDAK [Hold Acknowledge]

When an external bus requester has become the highest priority requester, the internal bus arbiter will assert the HLDAK output indicating the address, data, and control buses have entered a high-impedance state and are available for use by the external bus master.

Should the internal DMAU or RCU (demand mode) request the bus, the bus arbiter will drive HLDAK low. When this occurs, the external bus master should complete the current bus cycle and negate the HLDRQ signal. This allows the bus arbiter to reassign the bus to the higher priority requester.

If a higher priority internal bus master subsequently requests the bus, the high-level width of HLDAK is guaranteed to be a minimum of one CLKOUT period.

HLDRQ [Hold Request]

This active-high signal is asserted by an external bus master requesting to use the local address, data, and control buses. The HLDRQ input is used by the internal bus arbiter, which gives control of the buses to the highest priority bus requester in the following order.

Bus Master	Priority
RCU	Highest (demand mode)
DMAU	•
HLDRQ	•
CPU	•
RCU	Lowest (normal operation)

$\overline{INTAK}/TOUT1/\overline{SRDY}$ [Interrupt Acknowledge]/[Timer 1 Output]/[Serial Ready]

Three output signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- $\overline{INTAK}$ is an interrupt acknowledge signal used to cascade external slave μPD71059 Interrupt Controllers. $\overline{INTAK}$ is asserted during T2, T3, and TW states of an interrupt acknowledge cycle.
- TOUT1 is the output of timer/counter 1.
- $\overline{SRDY}$ is an active-low output and indicates that the serial control unit is ready to receive the next character.

INTP1-INTP7 [Peripheral Interrupts]

INTP1-INTP7 accept either rising-edge or high-level triggered asynchronous interrupt requests from external peripherals. These INTP1-INTP7 inputs are internally synchronized and prioritized by the interrupt control unit, which requests the CPU to perform an interrupt acknowledge bus cycle. External interrupt controllers such as the μPD71059 can be cascaded to increase the number of vectored interrupts.

These interrupt inputs cause the CPU to exit both the standby and 8080 emulation modes.

The INTP1-INTP7 inputs contain internal pull-up resistors and may be left unconnected.

$\overline{IORD}$ [I/O Read]

This three-state pin outputs an active-low I/O read strobe during T2, T3, and Tw of an I/O read bus cycle. Both CPU I/O read and DMA write bus cycles assert $\overline{IORD}$. $\overline{IORD}$ is not asserted when the bus cycle corresponds to an internal peripheral or register. It enters the high-impedance state during hold acknowledge.

$\overline{IOWR}$ [I/O Write]

This three-state pin outputs an active-low I/O write strobe during T2, T3, and TW of a CPU I/O write or an extended DMA read cycle and during T3 and TW of a DMA read bus cycle. $\overline{IOWR}$ is not asserted when the bus cycle corresponds to an internal peripheral or register. It enters the high-impedance state during hold acknowledge.

$\overline{MRD}$ [Memory Read Strobe]

This three-state pin outputs an active-low memory read strobe during T2, T3, and TW of a memory read bus cycle. CPU memory read, DMA read, and refresh bus cycles all assert $\overline{MRD}$. $\overline{MRD}$ enters the high-impedance state during hold acknowledge.

$\overline{MWR}$ [Memory Write Strobe]

This three-state pin outputs an active-low memory write strobe during T2, T3, and TW of a CPU memory write or DMA extended write bus cycle and during T3 and TW of a DMA normal write bus cycle. $\overline{MWR}$ enters the high-impedance state during hold acknowledge.

NMI [Nonmaskable Interrupt]

The NMI pin is a rising-edge-triggered interrupt input that cannot be masked by software. NMI is sampled by CPU logic each clock cycle and when found valid for one or more CLKOUT cycles, the NMI interrupt is accepted. The CPU will process the NMI interrupt immediately after the current instruction finishes execution by fetching the segment and offset of the NMI handler from interrupt vector 2. The NMI interrupt causes the CPU to exit both the standby and 8080 emulation modes. The NMI input takes precedence over the maskable interrupt inputs.

$\overline{\text{POLL}}$ [Poll]

The active-low $\overline{\text{POLL}}$ input is used to synchronize the operation of external devices with the CPU. During execution of the POLL instruction, the CPU checks the $\overline{\text{POLL}}$ input state every five clocks until $\overline{\text{POLL}}$ is once again asserted.

QS_1-QS_0 [Queue Status]

The QS_1 and QS_0 outputs maintain instruction synchronization between the μPD70216 CPU and external devices.These outputs are interpreted as follows.

QS_1	QS_0	Instruction Queue Status
0	0	No operation
0	1	First byte of instruction fetched
1	0	Flush queue contents
1	1	Subsequent byte of instruction fetched

Queue status is valid for one clock cycle after the CPU has accessed the instruction queue.

READY [Ready]

This active-high input synchronizes external memory and peripheral devices with the μPD70216. Slow memory and I/O devices can lengthen a bus cycle by negating the READY input and forcing the BIU to insert TW states. READY must be negated prior to the rising edge of CLKOUT during the T2 or by the last internally-generated TW state to guarantee recognition. When READY is once again asserted and recognized by the BIU, the BIU will proceed to the T4 state.

The READY input operates in parallel with the internal μPD70216 wait control unit and can be used to insert more than three wait states into a bus cycle.

$\overline{\text{REFRQ}}$ [Refresh Request]

$\overline{\text{REFRQ}}$ is an active-low output indicating the current bus cycle is a memory refresh operation. $\overline{\text{REFRQ}}$ is used to disable memory address decode logic and refresh dynamic memories. The 8-bit refresh row address is placed on A_8-A_1 during a refresh bus cycle.

$\overline{\text{RESET}}$ [Reset]

$\overline{\text{RESET}}$ is a Schmitt trigger input used to force the μPD70216 to a known state by resetting the CPU and on-chip peripherals. $\overline{\text{RESET}}$ must be asserted for a minimum of four clocks to guarantee recognition. After $\overline{\text{RESET}}$ has been released, the CPU will start program execution from address FFFF0H in the native mode.

$\overline{\text{RESET}}$ will release the CPU from the low-power standby mode and force it to the native mode.

3d

RESOUT [Reset Output]

This active-high output is available to perform a system-wide reset function. $\overline{\text{RESET}}$ is internally synchronized with CLKOUT and output on the RESOUT pin.

TCLK

TCLK is an external clock source for the timer control unit. The three timer/counters can be programmed to operate with either the TCLK input or a prescaled CLKOUT input.

TCTL2

TCTL2 is the control input for timer/counter 2.

TOUT2

TOUT2 is the output of timer/counter 2.

$\overline{\text{UBE}}$ [Upper Byte Enable]

$\overline{\text{UBE}}$ is an active-low output, asserted when the upper byte of the 16-bit data bus contains valid data. $\overline{\text{UBE}}$ is used along with A_0 by the memory decoding logic to select the even/odd banks as follows.

Operation	$\overline{\text{UBE}}$	A_0	Bus Cycles
Word, even address	0	0	1
Word, odd address	0 1	1 (1st bus cycle) 0 (2nd bus cycle)	2
Byte, even address	1	0	1
Byte, odd address	0	1	1

$\overline{\text{UBE}}$ is a three-state output and enters the high-impedance state during hold acknowledge.

X1, X2 [Clock Inputs]

These pins accept either a parallel resonant, fundamental mode crystal or an external oscillator input with a frequency twice the desired operating frequency.

In the case of an external clock generator, the X2 pin can either be left unconnected or be driven by the complement of the X1 pin clock source.

Pin States

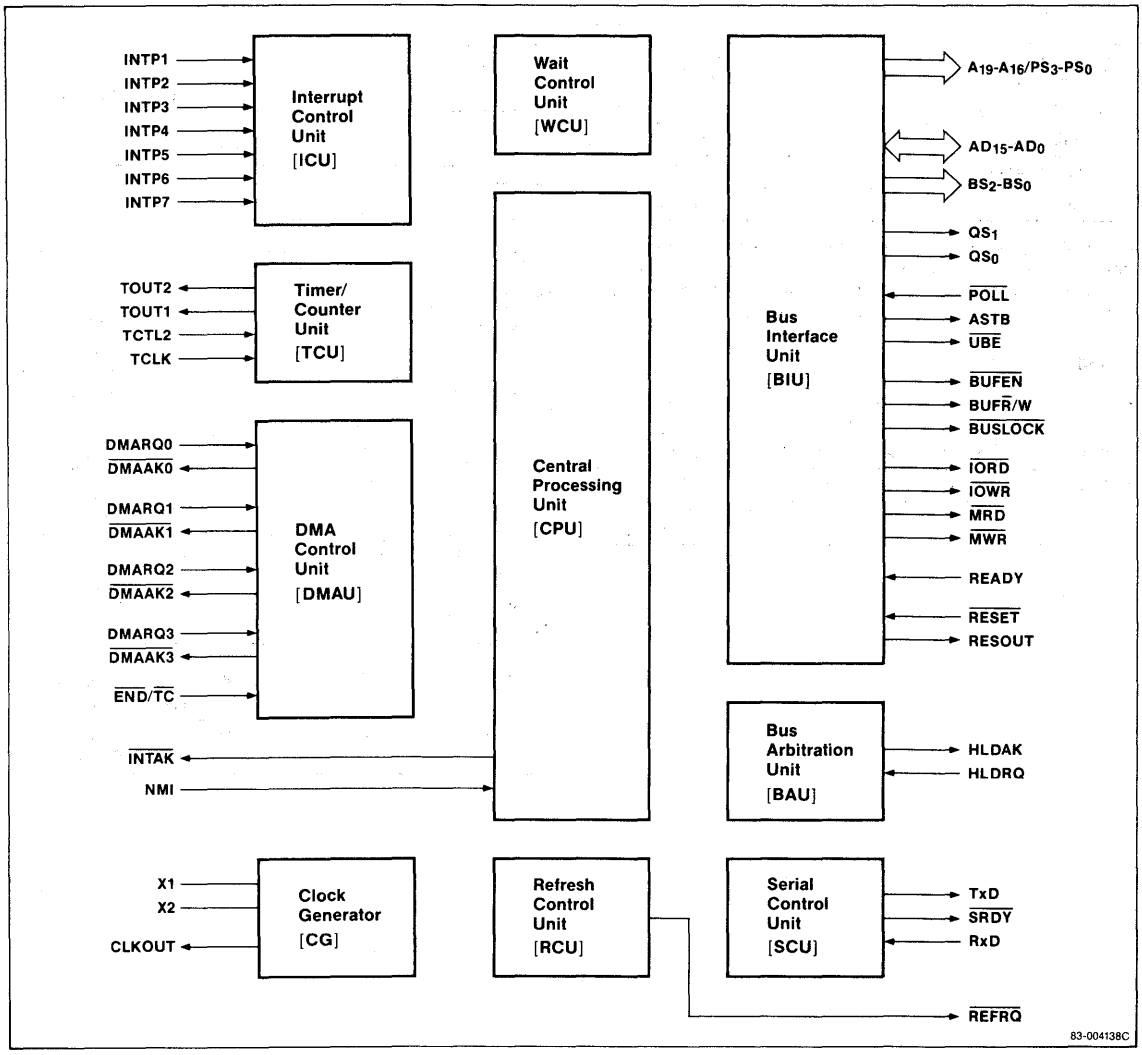
Table 1 lists the output pin states during the Hold, Halt, Reset and DMA Cascade conditions.

Table 1. Output Pin States

Symbol	Output	Hold	Halt	Reset	DMA Cascade
A_{19}-A_{16}/PS_3-PS_0	3-state Out	Hi-Z	H/L	H/L	Hi-Z
AD_{15}-AD_0	3-state I/O	Hi-Z	H/L	Hi-Z	Hi-Z
ASTB	Out	L	L	L	L
BS_2-BS_0	3-state Out	Hi-Z	H	H	H
$\overline{BUFEN}$	3-state Out	Hi-Z	H	H	Hi-Z
BUF$\overline{R}$/W	3-state Out	Hi-Z	H/L	H	Hi-Z
$\overline{BUSLOCK}$	3-state Out	Hi-Z	H/L	H	Hi-Z
CLKOUT	Out	H/L	H/L	H/L	H/L
$\overline{DMAAK0}$-$\overline{DMAAK2}$	Out	H	H/L	H	H/L
$\overline{DMAAK3}$/	Out	H	H/L	H	H/L
TxD		H/L		H/L	H/L
$\overline{END}$/$\overline{TC}$	I/O	H	H/L	H	H
HLDAK	Out	H	H/L	L	L
$\overline{INTAK}$	Out	H	H	H	H
TOUT1		H/L	H/L		H/L
$\overline{SRDY}$		H/L	H/L		H/L
$\overline{IORD}$	3-state Out	Hi-Z	H	H	Hi-Z
$\overline{IOWR}$	3-state Out	Hi-Z	H	H	Hi-Z
$\overline{MRD}$	3-state Out	Hi-Z	H	H	Hi-Z
$\overline{MWR}$	3-state Out	Hi-Z	H	H	Hi-Z
QS_1-QS_0	Out	H/L	L	L	H/L
$\overline{REFRQ}$	Out	H	H/L	H	H
RESOUT	Out	L	L	H	L
TOUT2	Out	H/L	H/L	H/L	H/L
$\overline{UBE}$	3-state Out	Hi-Z	H	H	Hi-Z

H: high level; L: low level; H/L: high or low level; Hi-Z: high impedance.

Block Diagram



3d

Absolute Maximum Ratings

$T_A = +25°C$

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.3 V
CLK input voltage, V_K	−0.5 to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 to V_{DD} + 0.3 V
Operating temperature, T_{OPT}	−10 to +70°C
Storage temperature, T_{STG}	−65 to +150°C

Comment: Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage. The device should be operated within the limits specified under DC and AC Characteristics.

Capacitance

$T_A = +25°C$, $V_{DD} = 0$ V

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input capacitance	C_I		15	pF	$f_C = 1$ MHz; unmeasured pins are returned to 0 V.
Output capacitance	C_O		15	pF	

DC Characteristics

$T_A = -10$ to $+70°C$, $V_{DD} = +5$ V ±10% (8 MHz), $V_{DD} = 5$ V ±5% (10 MHz)

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input voltage, high	V_{IH}	2.2	V_{DD} + 0.3	V	
Input voltage, low	V_{IL}	−0.5	0.8	V	
X1, X2 input voltage, high	V_{KH}	3.9	V_{DD} + 1.0	V	
X1, X2 input voltage, low	V_{KL}	−0.5	0.6	V	
Output voltage, high	V_{OH}	0.7 V_{DD}		V	$I_{OH} = -400$ μA
Output voltage, low	V_{OL}		0.4	V	$I_{OL} = 2.5$ mA
Input leakage current, high	I_{LIH}		10	μA	$V_I = V_{DD}$
Input leakage current, low	I_{LIPL}		−300	μA	$V_I = 0$ V, INTP input pins
	I_{LIL}		−10	μA	$V_I = 0$ V, other input pins
Output leakage current, high	I_{LOH}		10	μA	$V_O = V_{DD}$
Output leakage current, low	I_{LOL}		−10	μA	$V_O = 0$ V
Supply current 8 MHz	I_{DD}		90	mA	Normal mode
			20	mA	Standby mode
10 MHz	I_{DD}		120	mA	Normal mode
			25	mA	Standby mode

AC Characteristics

$T_A = -10$ to $+70°C$; $V_{DD} = 5$ V ±10% (8 MHz), $V_{DD} = 5$ V ±5% (10 MHz), $C_L = 100$ pF

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
External clock input cycle time	t_{CYX}	62	250	50	250	ns	
External clock pulse width, high	t_{XXH}	20		19		ns	$V_{KH} = 3.0$ V
External clock pulse width, low	t_{XXL}	20		19		ns	$V_{KL} = 1.5$ V
External clock rise time	t_{XR}		10		5	ns	1.5 → 3.0 V
External clock fall time	t_{XF}		10		5	ns	3.0 → 1.5 V
CLKOUT cycle time	t_{CYK}	124	500	100	500	ns	
CLKOUT pulse width, high	t_{KKH}	$0.5\ t_{CYK} - 7$		$0.5\ t_{CYK} - 5$		ns	$V_{KH} = 3.0$ V
CLKOUT pulse width, low	t_{KKL}	$0.5\ t_{CYK} - 7$		$0.5\ t_{CYK} - 5$		ns	$V_{KL} = 1.5$ V
CLKOUT rise time	t_{KR}		7		5	ns	1.5 → 3.0 V
CLKOUT fall time	t_{KF}		7		5	ns	3.0 → 1.5 V
CLKOUT delay time from external clock	t_{DXK}		55		40	ns	
Input rise time (except external clock)	t_{IR}		20		15	ns	0.8 → 2.2 V
Input fall time (except external clock)	t_{IF}		12		10	ns	2.2 → 0.8 V
Output rise time (except CLKOUT)	t_{OR}		20		15	ns	0.8 → 2.2 V
Output fall time (except CLKOUT)	t_{OF}		12		10	ns	2.2 → 0.8 V
$\overline{RESET}$ setup time to CLKOUT↓	t_{SRESK}	25		20		ns	
$\overline{RESET}$ hold time after CLKOUT↓	t_{HKRES}	35		25		ns	
RESOUT delay time from CLKOUT↓	t_{DKRES}	5	60	5	50	ns	
READY inactive setup time to CLKOUT↑	t_{SRYLK}	15		15		ns	
READY inactive hold time after CLKOUT↑	t_{HKRYL}	25		20		ns	
READY active setup time to CLKOUT↑	t_{SRYHK}	15		15		ns	
READY active hold time after CLKOUT↑	t_{HKRYH}	25		20		ns	
NMI, $\overline{POLL}$ setup time to CLKOUT↑	t_{SIK}	15		15		ns	
Data setup time to CLKOUT↓	t_{SDK}	15		15		ns	
Data hold time after CLKOUT↓	t_{HKD}	10		10		ns	
Address delay time from CLKOUT↓	t_{DKA}	10	55	10	50	ns	A_{19}-A_0 $\overline{UBE}$
Address hold time after CLKOUT↓	t_{HKA}	10		10		ns	
I/O recovery time	t_{AI}	$2t_{CYK} - 50$		$2t_{CYK} - 40$		ns	(Note 1)
PS delay time from CLKOUT↓	t_{DKP}	10	60	10	50	ns	
PS float delay time from CLKOUT↑	t_{FKP}	10	60	10	50	ns	
Address setup time to ASTB↓	t_{SAST}	$t_{KKL} - 20$		$t_{KKL} - 30$		ns	
Address float delay time from CLKOUT↓	t_{FKA}	t_{HKA}	60	t_{HKA}	50	ns	
ASTB↑ delay time from CLKOUT↓	t_{DKSTH}		45		40	ns	
ASTB↓ delay time from CLKOUT↑	t_{DKSTL}		50		45	ns	
ASTB pulse width, high	t_{STST}	$t_{KKL} - 10$		$t_{KKL} - 10$		ns	
Address hold time after ASTB↓	t_{HSTA}	$t_{KKH} - 20$		$t_{KKH} - 20$		ns	
Control delay time from CLKOUT	t_{DKCT1}	10	70	10	60	ns	(Note 2)
	t_{DKCT2}	10	60	10	55	ns	(Note 3)

3d

AC Characteristics (cont)

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
$\overline{RD}$↓ delay time from address float	t_{DAFRL}	0		0		ns	(Note 4)
$\overline{RD}$↓ delay time from CLKOUT↓	t_{DKRL}	10	75	10	65	ns	
$\overline{RD}$↑ delay time from CLKOUT↓	t_{DKRH}	10	70	10	60	ns	
$\overline{REFRQ}$↑ delay from $\overline{MRD}$↑	t_{DRQHRH}	$t_{KKL}-30$		$t_{KKL}-30$		ns	(Note 5)
Address delay time from $\overline{RD}$↑	t_{DRHA}	$t_{CYK}-40$		$t_{CYK}-40$		ns	
$\overline{RD}$ pulse width, low	t_{RR}	$2t_{CYK}-50$		$2t_{CYK}-40$		ns	
BUF$\overline{R}$/W delay from $\overline{BUFEN}$↑	t_{DBECT}	$t_{KKL}-20$		$t_{KKL}-20$		ns	Read cycle
	t_{DWCT}	$t_{KKL}-20$		$t_{KKL}-20$		ns	Write cycle
Data output delay time from CLKOUT↓	t_{DKD}	10	60	10	55	ns	
Data float delay time from CLKOUT↓	t_{FKD}	10	60	10	55	ns	
$\overline{WR}$ pulse width, low	t_{WW}	$2t_{CYK}-40$		$2t_{CYK}-40$		ns	(Note 4)
BS↓ delay time from CLKOUT↑	t_{DKBL}	10	60	10	55	ns	
BS↑ delay time from CLKOUT↓	t_{DKBH}	10	60	10	55	ns	
HLDRQ setup time to CLKOUT↑	t_{SHQK}	20		15		ns	
HLDAK delay time from CLKOUT↓	t_{DKHA}	10	70	10	60	ns	
$\overline{DMAAK}$↓ delay time from CLKOUT↑	t_{DKHDA}	10	60	10	55	ns	
$\overline{DMAAK}$↓ delay time from CLKOUT↓	t_{DKLDA}	10	90	10	80	ns	Cascade mode
$\overline{WR}$ pulse width, low (DMA cycle)	t_{WW1}	$2t_{CYK}-40$		$2t_{CYK}-40$		ns	DMA extended write cycle
$\overline{WR}$ pulse width, low (DMA cycle)	t_{WW2}	$t_{CYK}-40$		$t_{CYK}-40$		ns	DMA normal write cycle
$\overline{RD}$↓, $\overline{WR}$↓ delay from DMAAK↓	t_{DDARW}	$t_{KKH}-30$		$t_{KKH}-30$		ns	
$\overline{DMAAK}$↑ delay from $\overline{RD}$↑	t_{DRHDAH}	$t_{KKL}-30$		$t_{KKL}-30$		ns	
$\overline{RD}$↑ delay from $\overline{WR}$↑	t_{DWHRH}	5		5		ns	
$\overline{TC}$ output delay time from CLKOUT↑	t_{DKTCL}		60		55	ns	
$\overline{TC}$ off delay time from CLKOUT↑	t_{DKTCF}		60		55	ns	
$\overline{TC}$ pulse width, low	t_{TCTCL}	$t_{CYK}-15$		$t_{CYK}-15$		ns	
$\overline{TC}$ pullup delay time from CLKOUT↑	t_{DKTCH}		$t_{KKH}+t_{CYK}-10$		$t_{KKH}+t_{CYK}-10$	ns	
$\overline{END}$ setup time to CLKOUT↑	t_{SEDK}	35		30		ns	
$\overline{END}$ pulse width, low	t_{EDEDL}	100		80		ns	
DMARQ setup time to CLKOUT↑	t_{SDQK}	35		30		ns	
INTPn pulse width, low	t_{IPIPL}	100		80		ns	
RxD setup time to SCU internal clock↓	t_{SRX}	1		0.5		μs	
RxD hold time after SCU internal clock↓	t_{HRX}	1		0.5		μs	
$\overline{SRDY}$ delay time from CLKOUT↓	t_{DKSR}		150		100	ns	

Notes:

(1) This is specified to guarantee a read/write recovery time for I/O devices.

(2) Delay from CLKOUT to DMA cycle $\overline{MWR}$/$\overline{IOWR}$ outputs.

(3) Delay from CLKOUT to BUF$\overline{R}$/W, $\overline{BUFEN}$, $\overline{INTAK}$, $\overline{REFRQ}$ outputs and CPU cycle $\overline{MWR}$/$\overline{IOWR}$ outputs.

(4) $\overline{RD}$ represents $\overline{IORD}$ and $\overline{MRD}$. $\overline{WR}$ represents $\overline{IOWR}$ and $\overline{MWR}$.

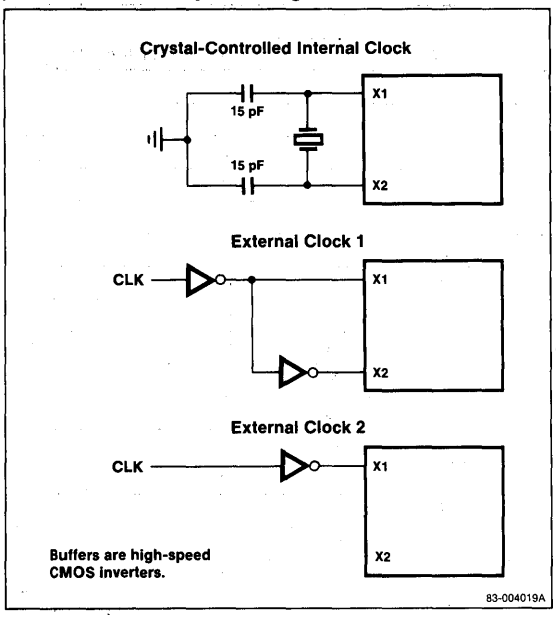
(5) This is specified to guarantee that $\overline{REFRQ}$ ↑ is delayed from $\overline{MRD}$ ↑ at all times.

AC Characteristics (cont)

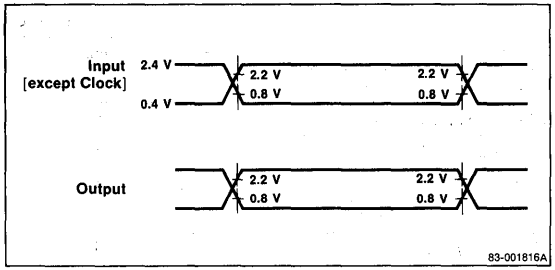
Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
TxD delay time from TOUT1↓	t_{DTX}		500		200	ns	
TCTL2 setup time from CLKOUT↓	t_{SGK}	50		40		ns	
TCTL2 setup time to TCLK↑	t_{SGTK}	50		40		ns	
TCTL2 hold time after CLKOUT↓	t_{HKG}	100		80		ns	
TCTL2 hold time after TCLK↑	t_{HTKG}	50		40		ns	
TCTL2 pulse width, high	t_{GGH}	50		40		ns	
TCTL2 pulse width, low	t_{GGL}	50		40		ns	
TOUT output delay time from CLKOUT↓	t_{DKTO}		200		150	ns	
TOUT output delay time from TOUT↓	t_{DTKTO}		150		100	ns	
TOUT output delay time from TCTL2↓	t_{DGTO}		120		90	ns	
TCLK rise time	t_{TKR}		25		25	ns	
TCLK fall time	t_{TKF}		25		25	ns	
TCLK pulse width, high	t_{TKTKH}	50		45		ns	
TCLK pulse width, low	t_{TKTKL}	50		45		ns	
TCLK cycle time	t_{CYTK}	124	DC	100	DC	ns	
RESET pulse width low	t_{RESET1}	50		50		μs	After power on
	t_{RESET2}	4 t_{CYK}		4 t_{CYK}			During operation

3d

μPD70216 Clock Input Configurations

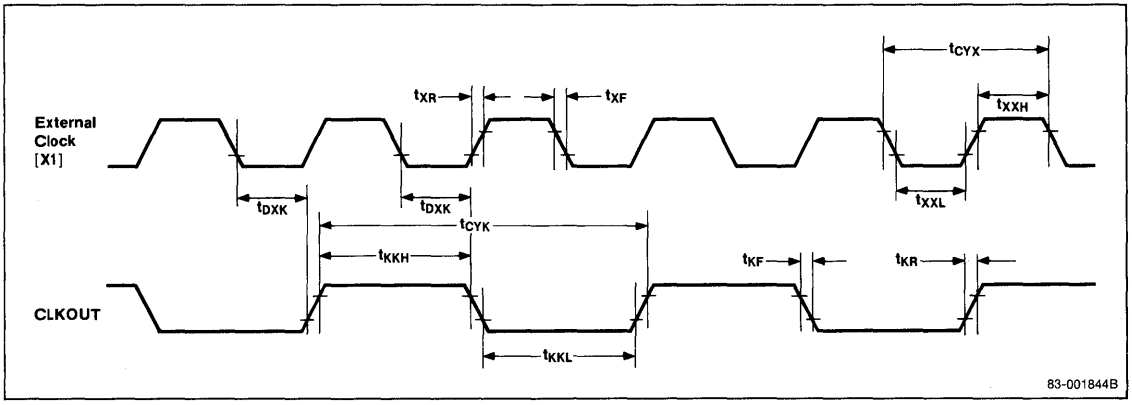


Timing Measurement Points



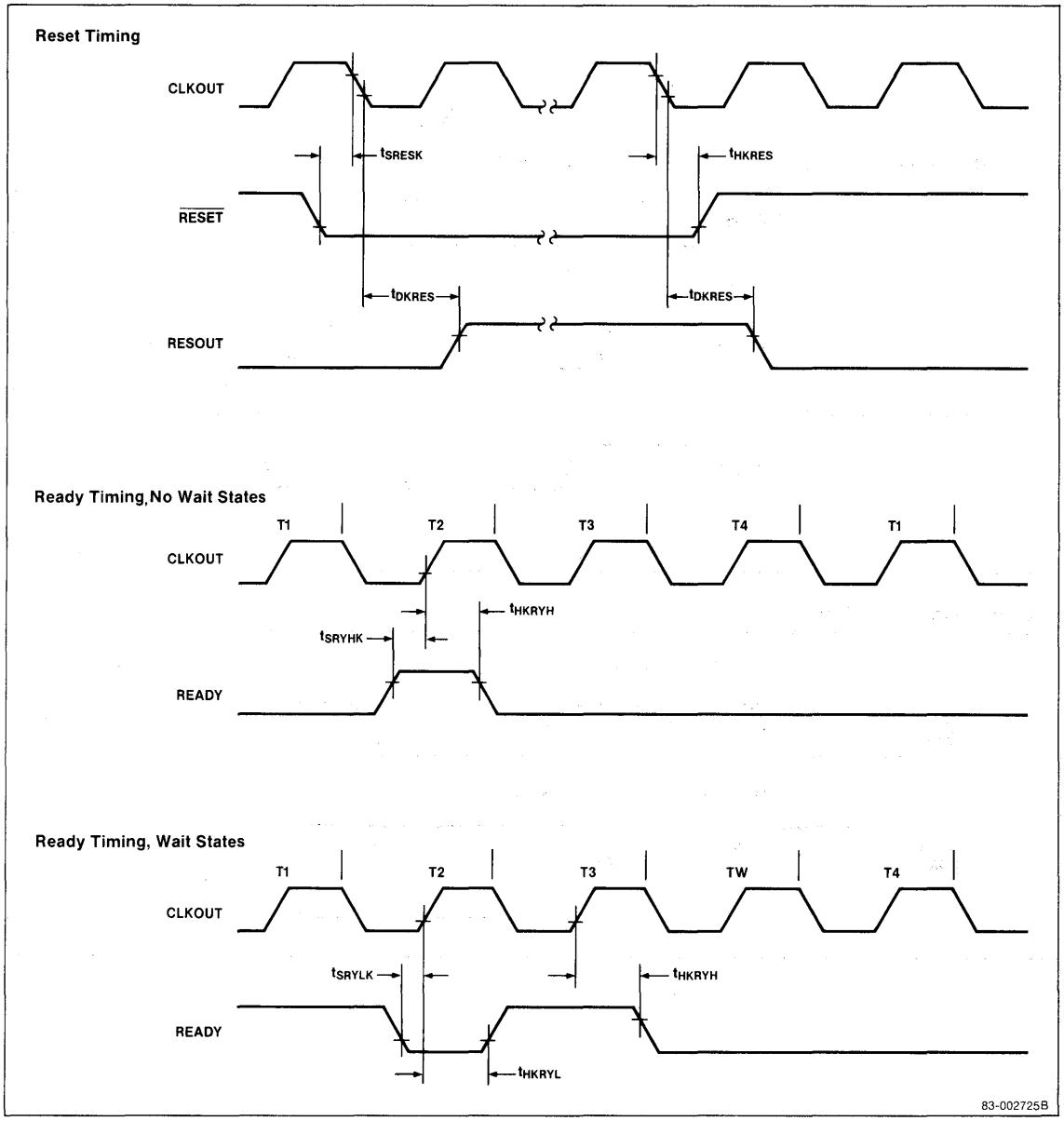
Timing Waveforms

Clock Timing



Timing Waveforms (cont)

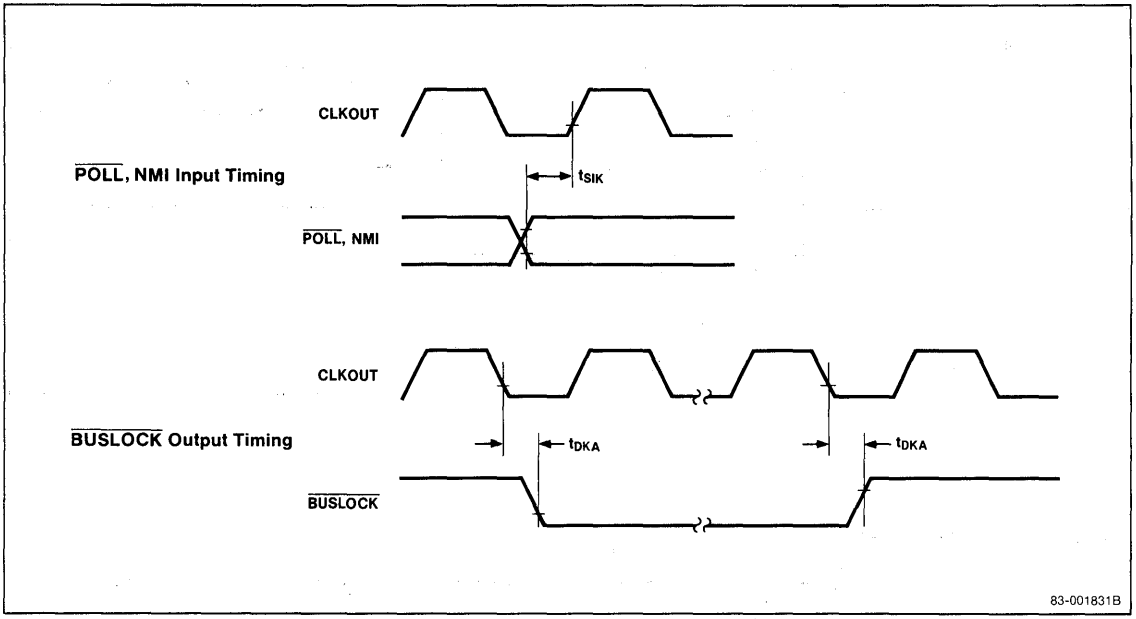
Reset and Ready Timing



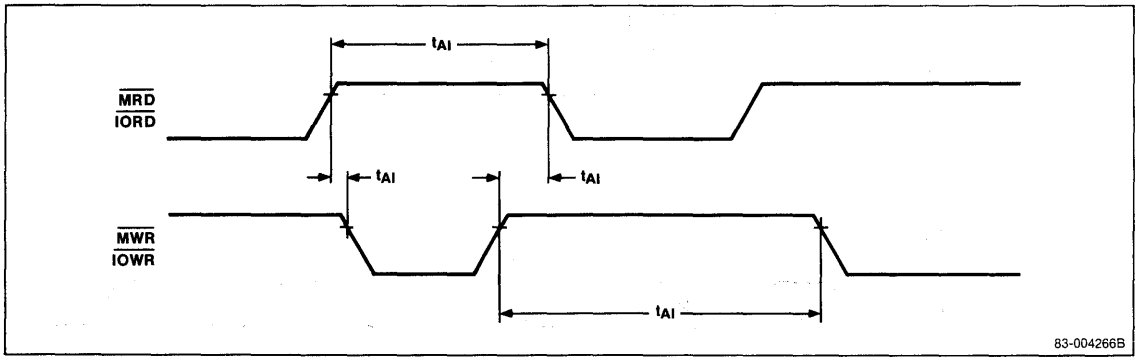
3d

Timing Waveforms (cont)

Poll, NMI, and Buslock Timing

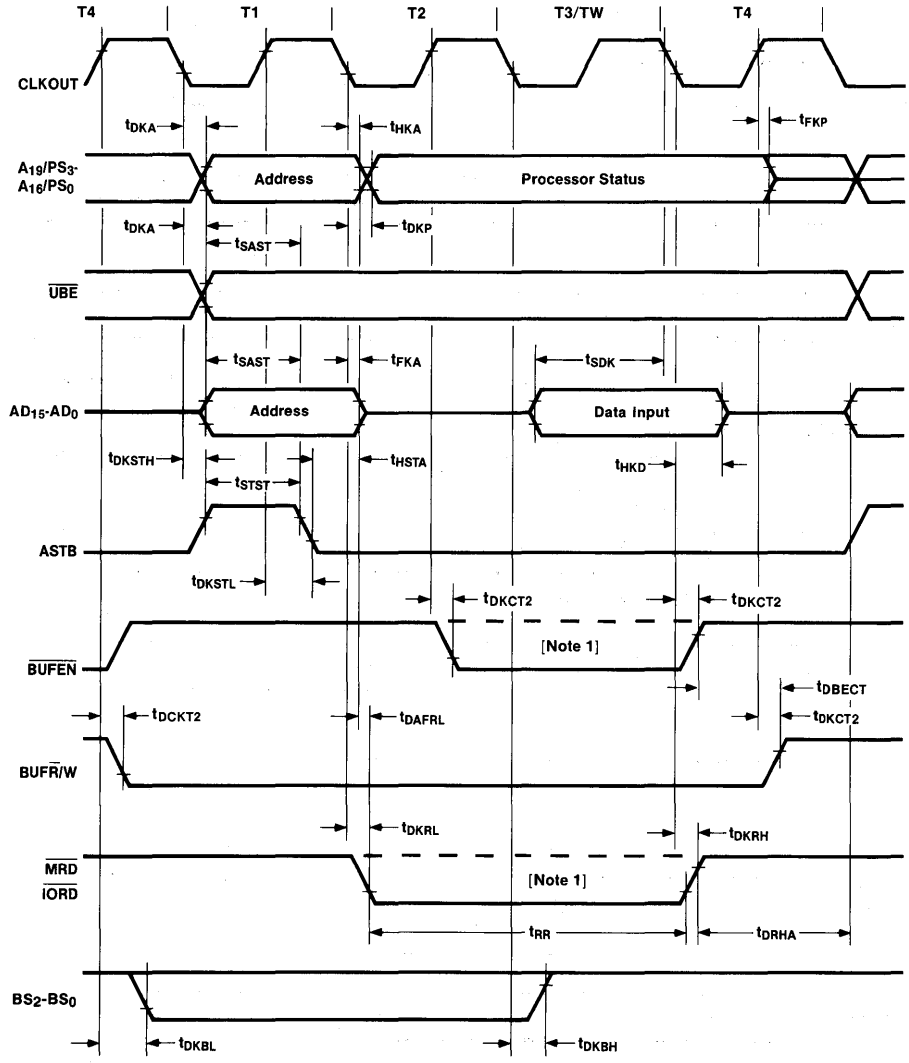


Read/Write Recovery Time



Timing Waveforms (cont)

Read Timing



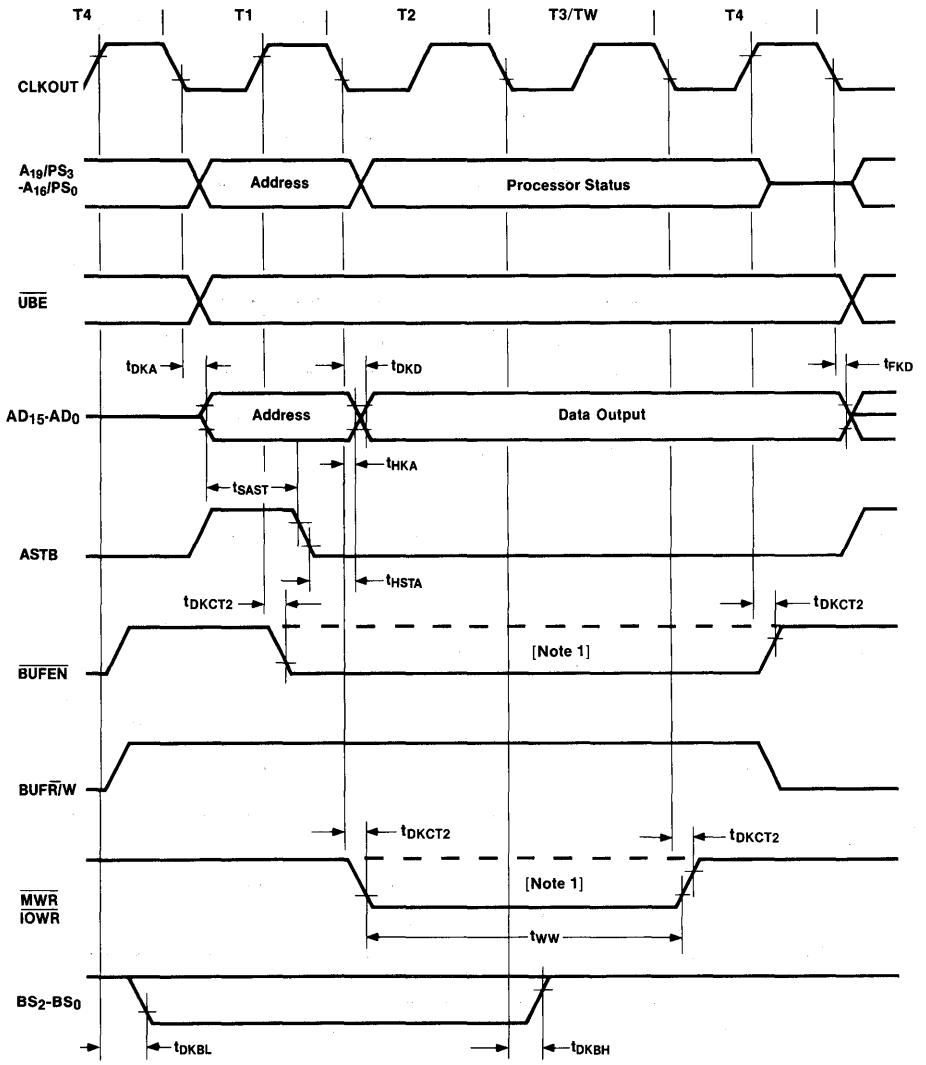
Note:
[1] Except internal I/O accesses.

83-001845B

3d

Timing Waveforms (cont)

Write Timing

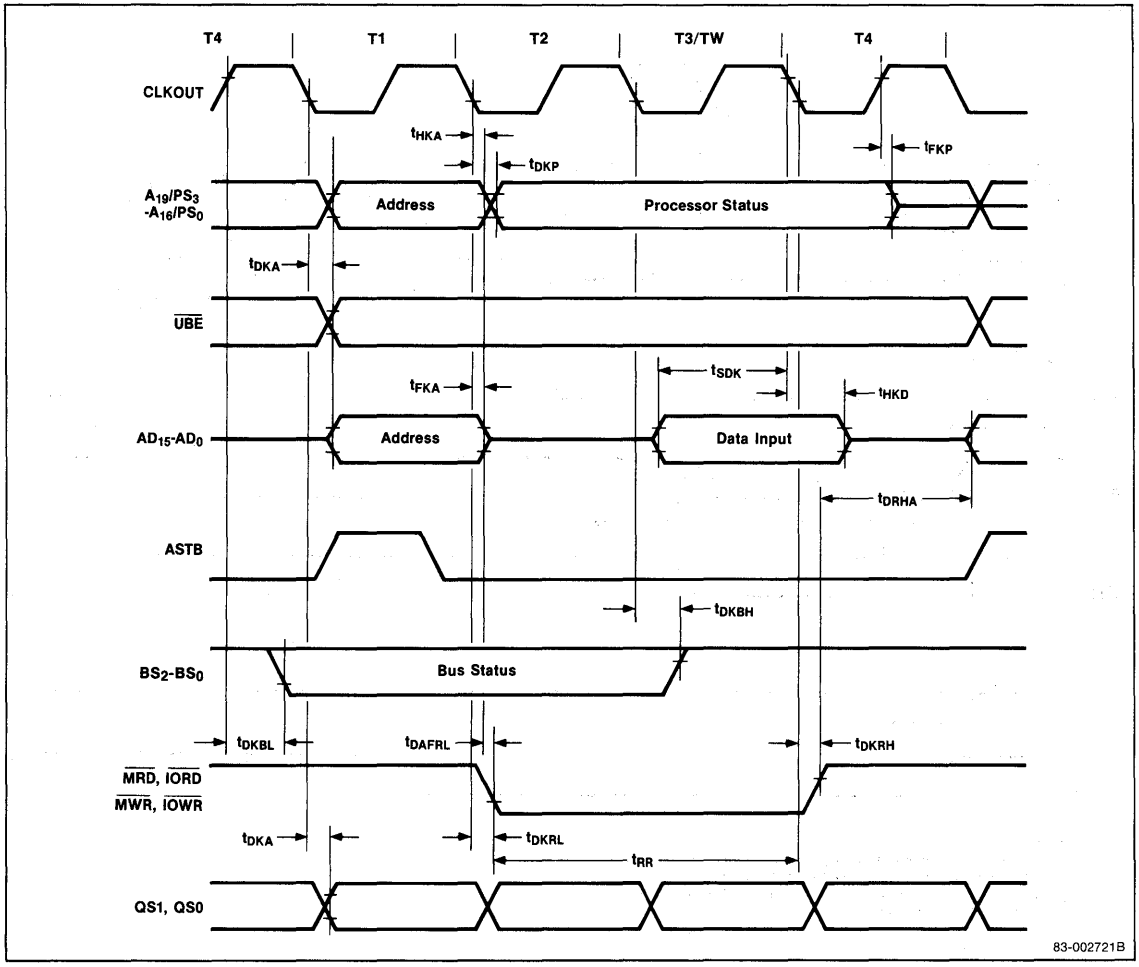


Note:
[1] Except internal I/O accesses.

83-001843B

Timing Waveforms (cont)

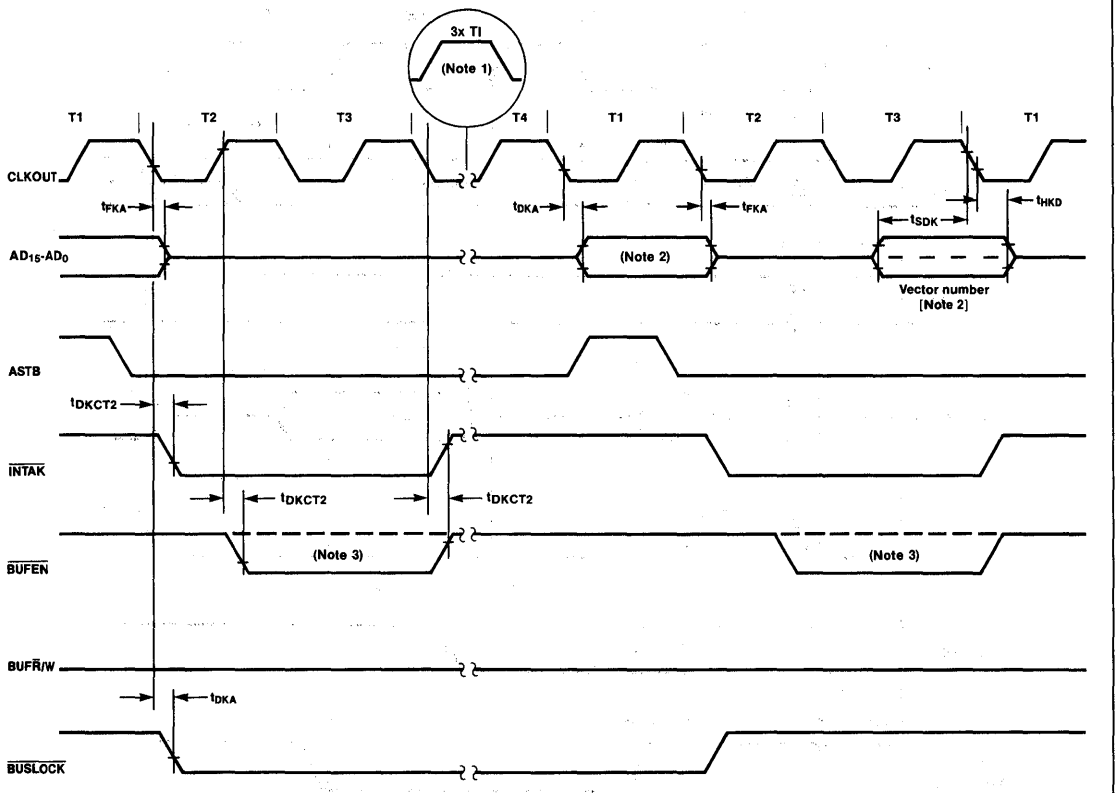
Status Timing



3d

Timing Waveforms (cont)

Interrupt Acknowledge Timing



Notes:

[1] If the interrupt is accepted following a control transfer, the three TI states can be replaced with an unused instruction fetch bus cycle.

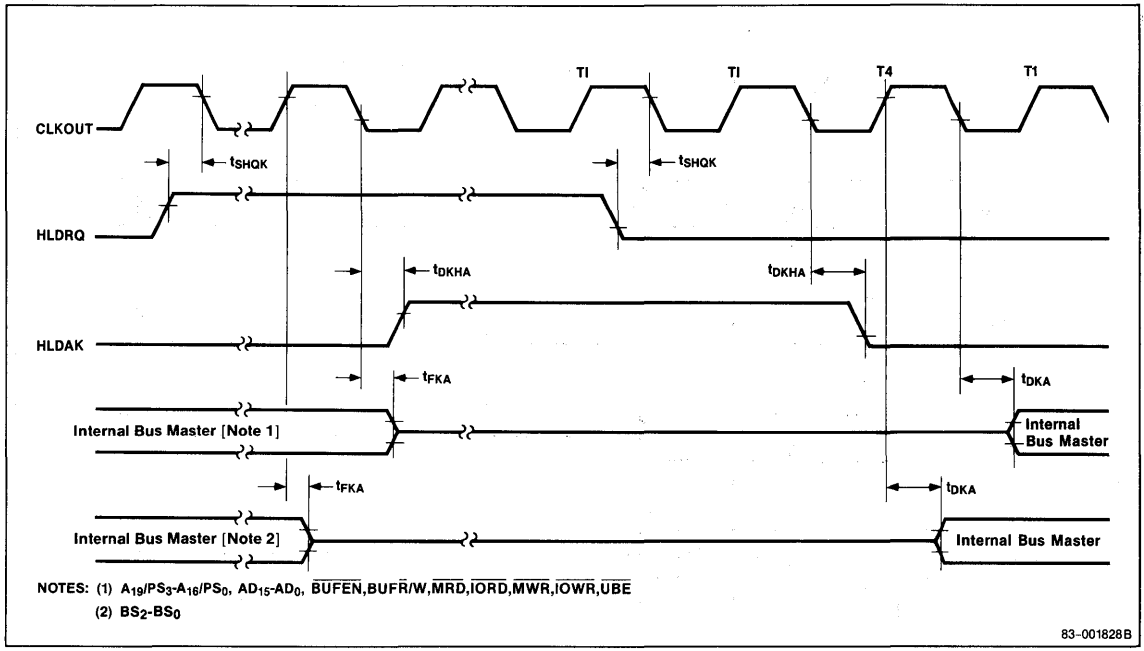
[2] Slave address when the interrupt is from external μPD70159. Undefined when internal ICU interrupt.

[3] Solid line when interrupt from external μPD70159. Dash line when internal ICU interrupt.

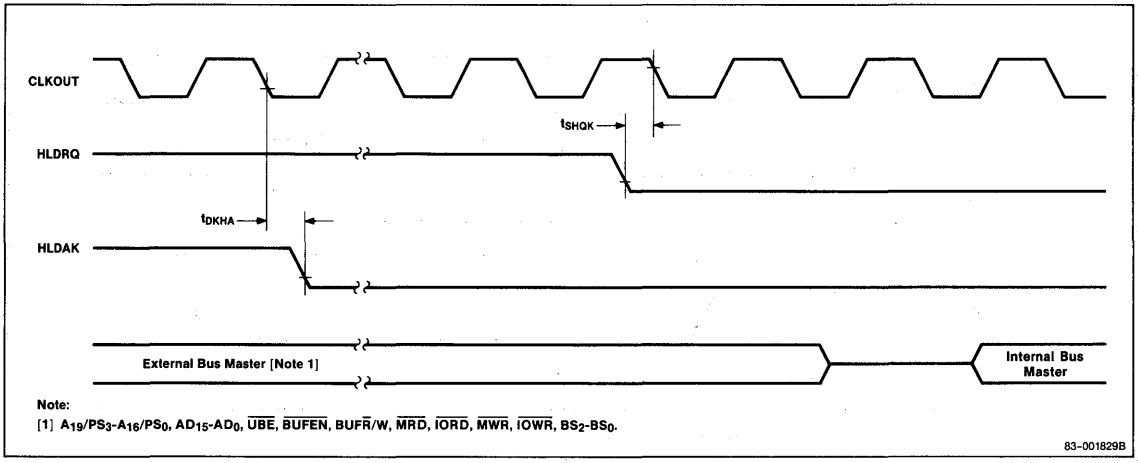
83-001841B

Timing Waveforms (cont)

HLDRQ/HLDAK Timing, Normal Operation



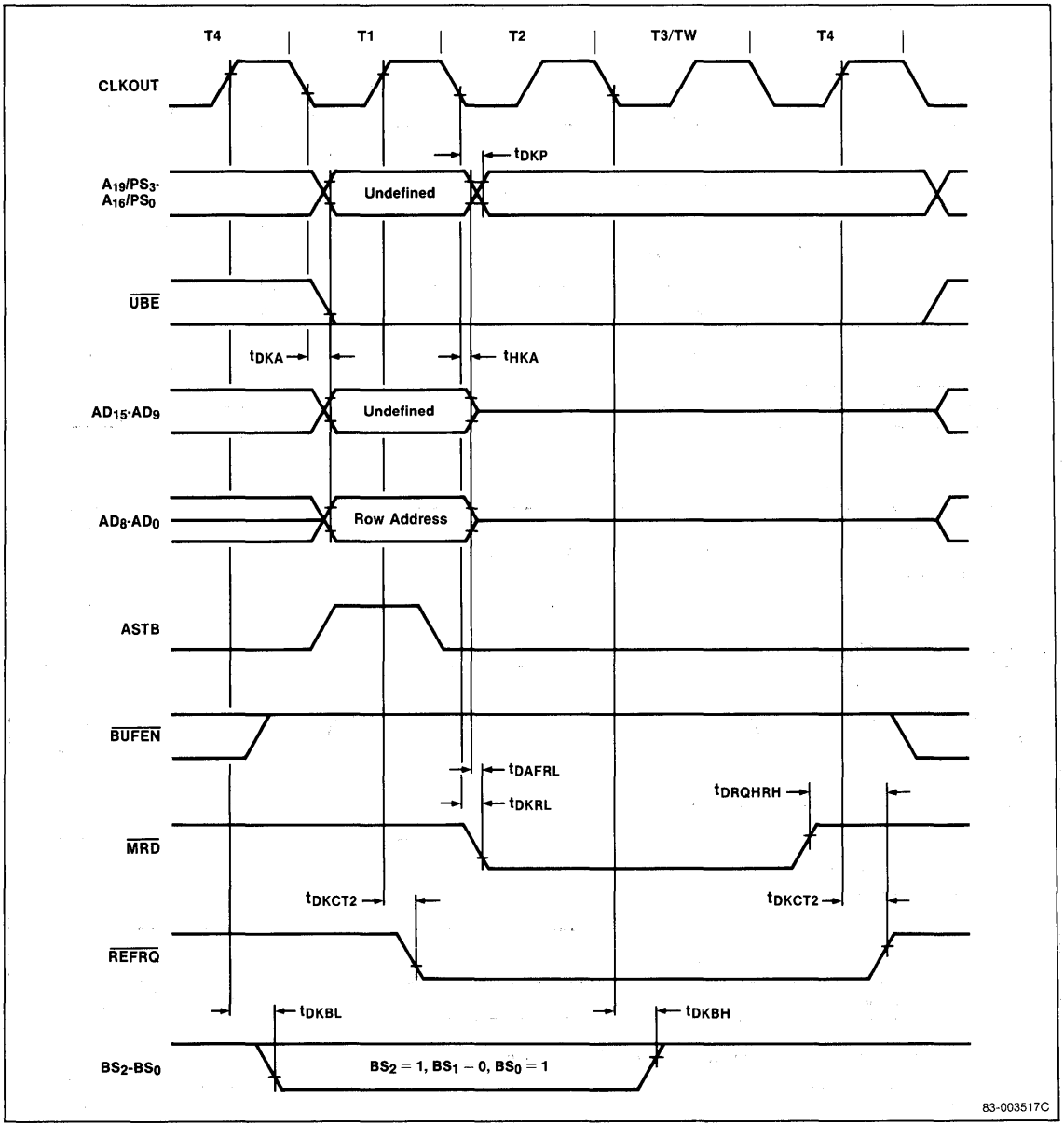
HLDRQ/HLDAK Timing, Bus Wait



3d

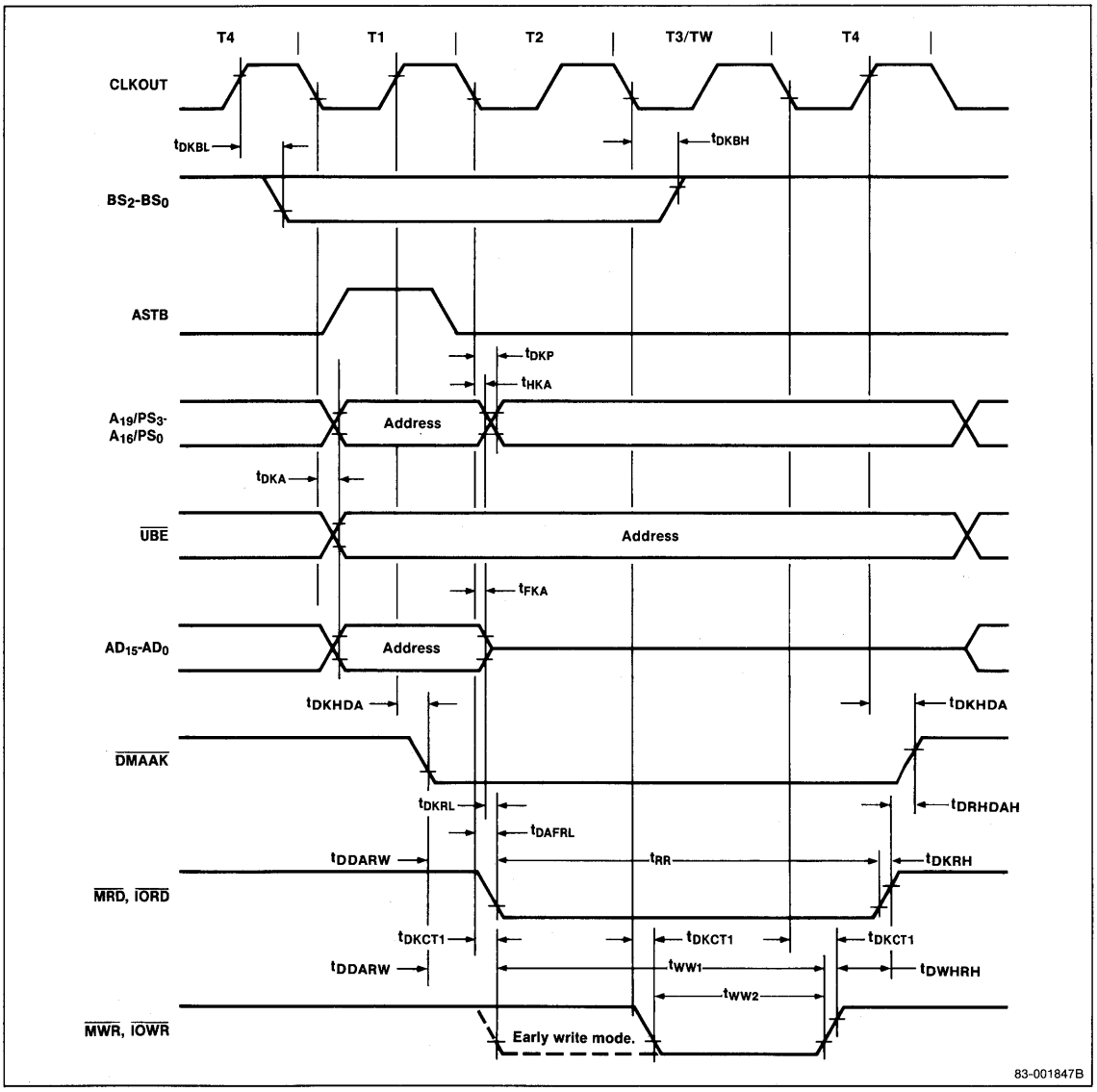
Timing Waveforms (cont)

Refresh Timing



Timing Waveforms (cont)

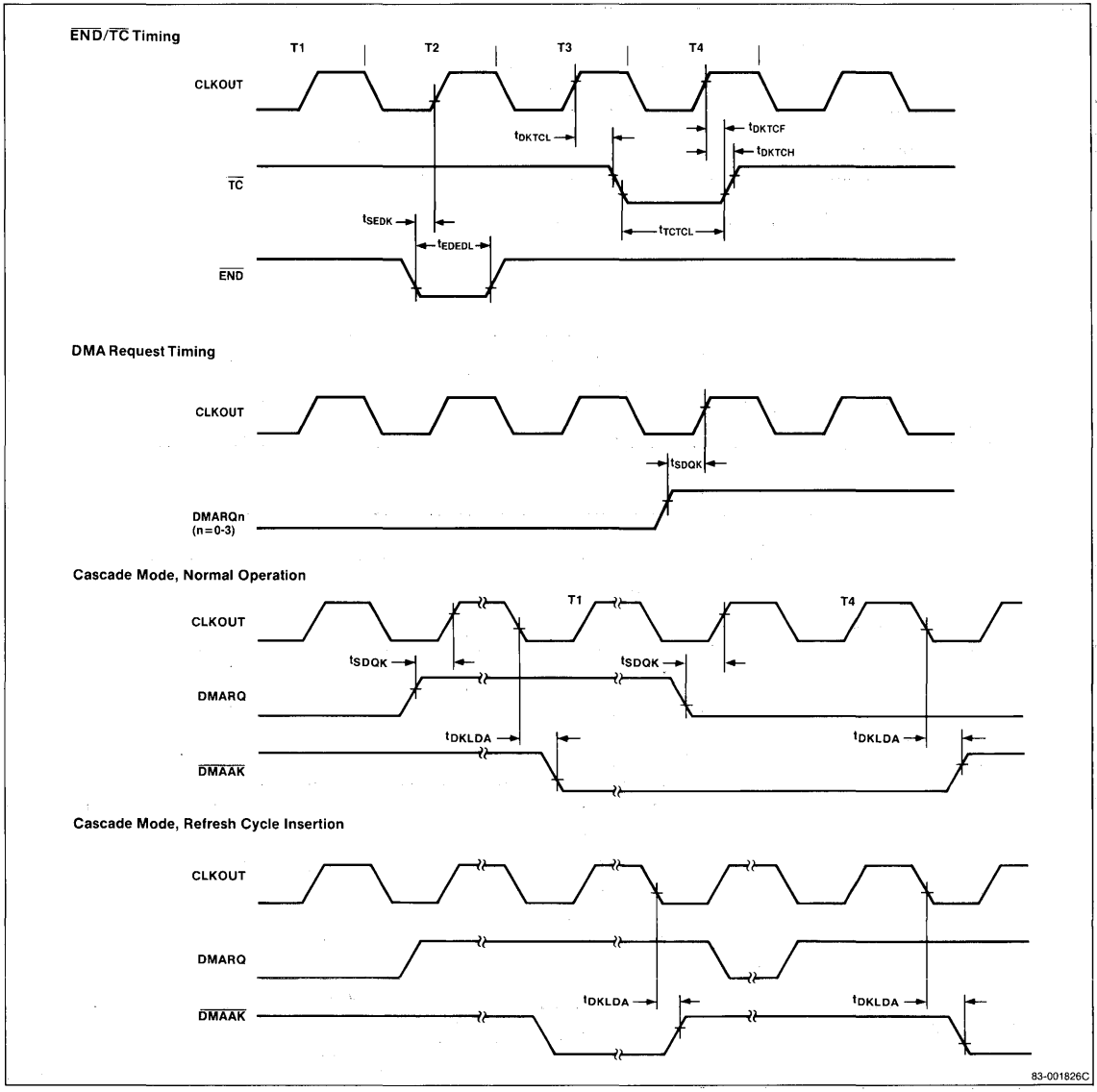
DMAU, DMA Transfer Timing



3d

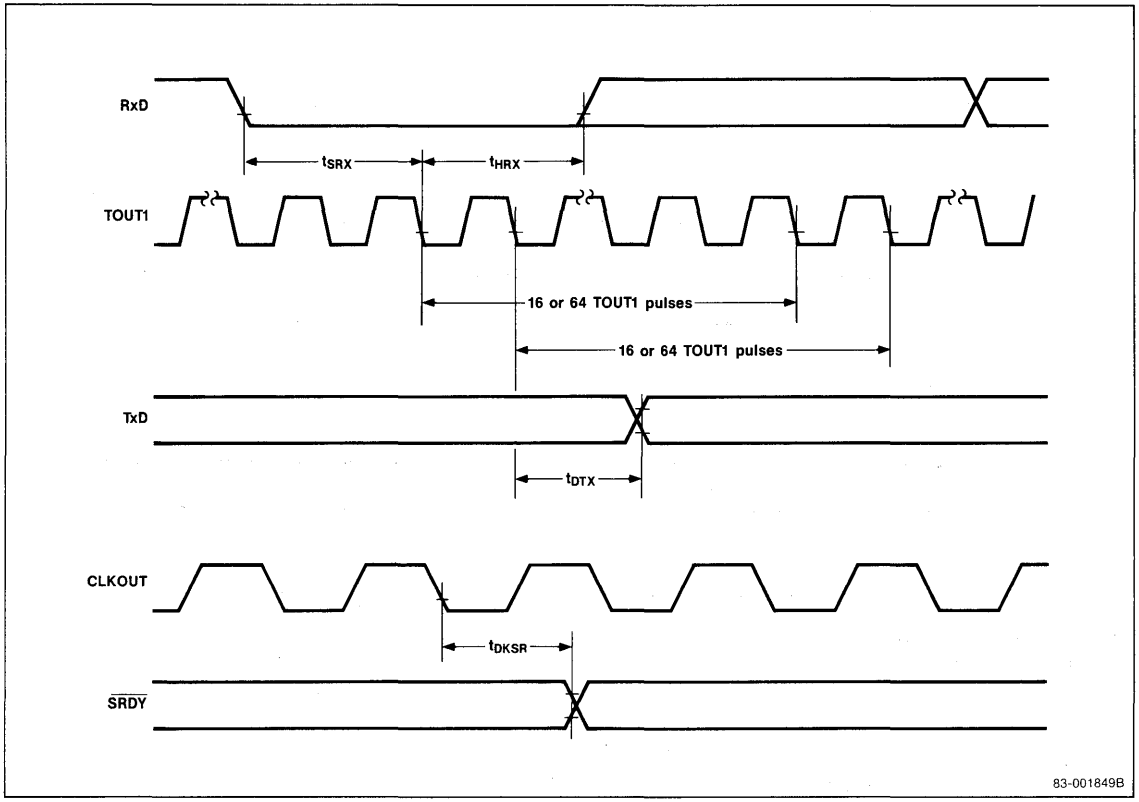
Timing Waveforms (cont)

DMA Timing

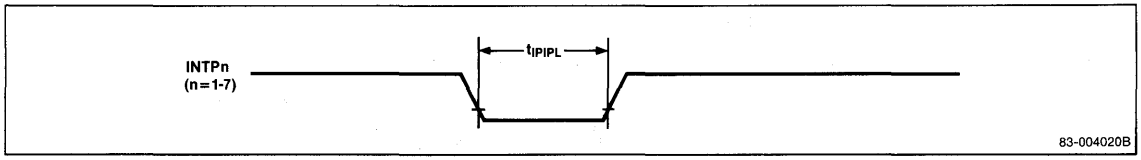


Timing Waveforms (cont)

SCU Timing



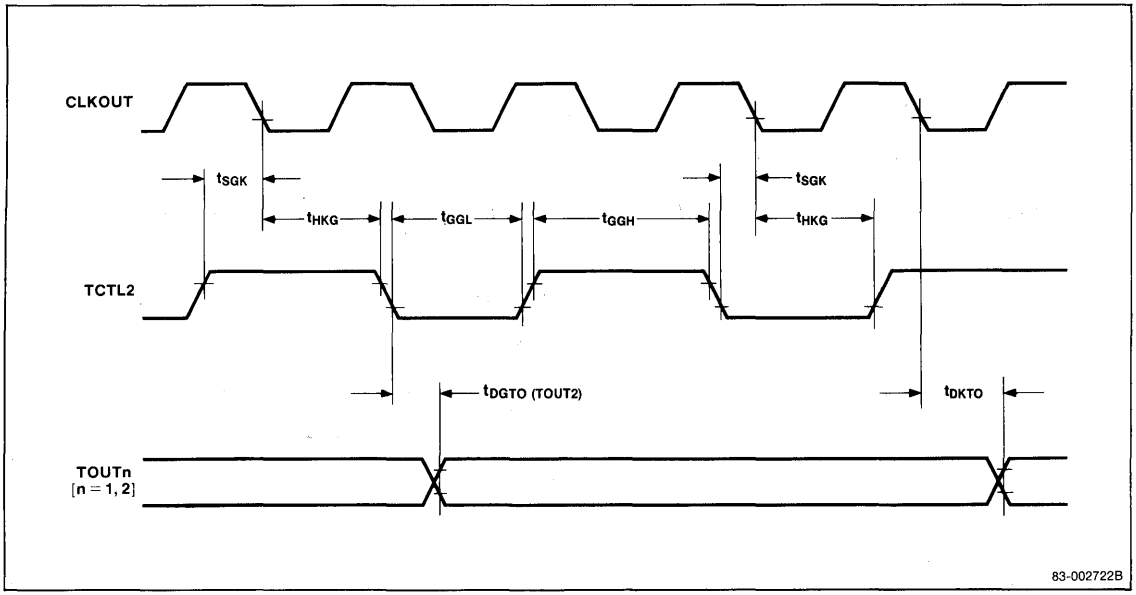
ICU Timing



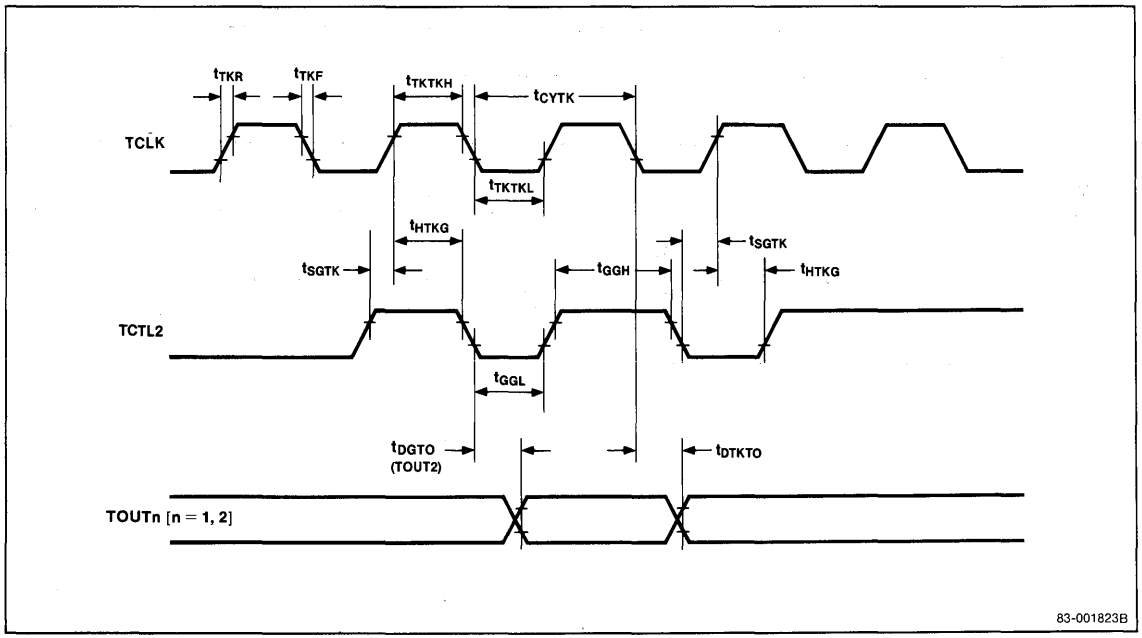
3d

Timing Waveforms (cont)

TCU Timing, Internal Clock Source



TCU Timing, TCLK Source



Functional Description

Refer to the μPD70216 block diagram for an overview of the ten major functional blocks listed below.

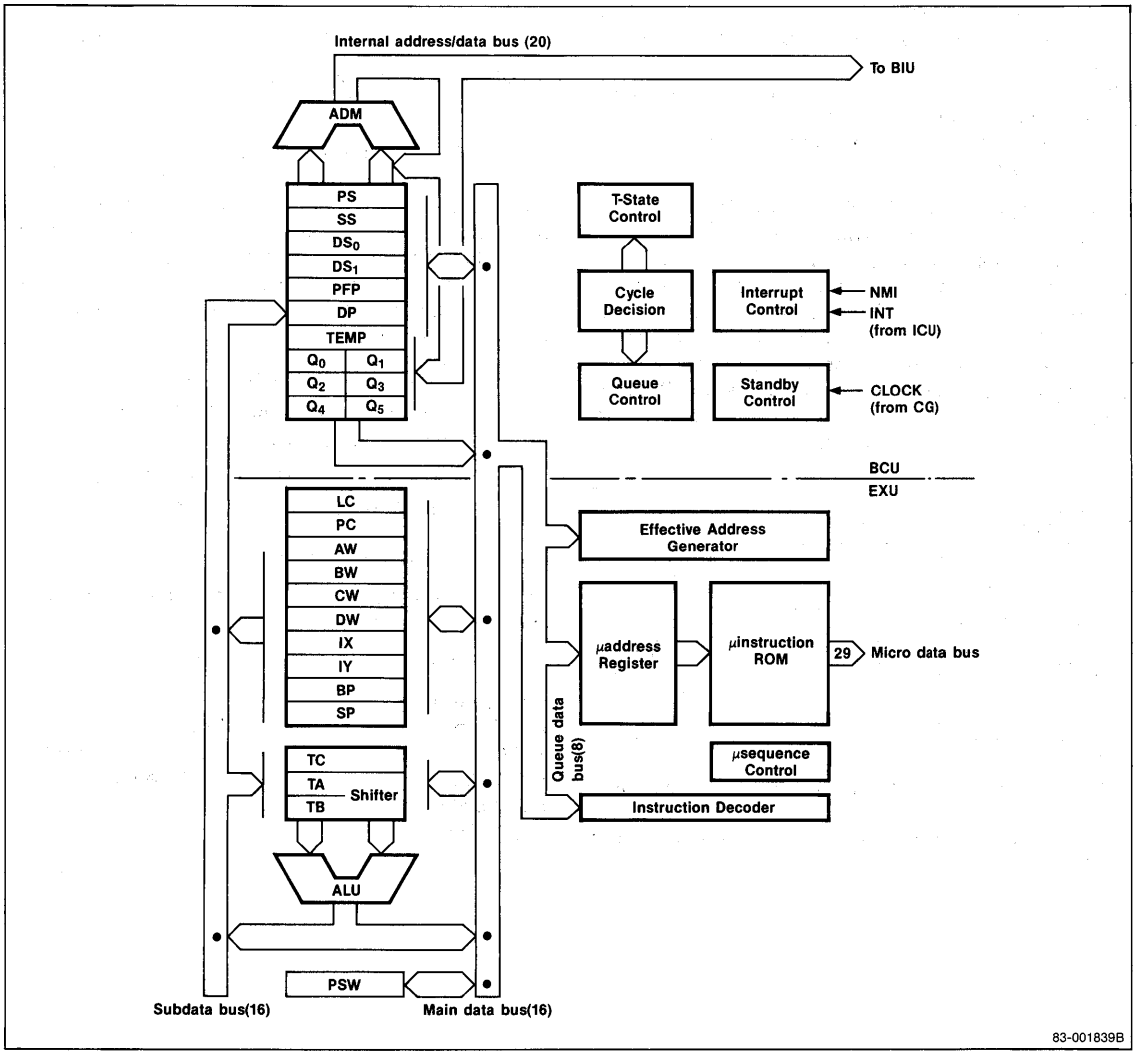
- Central processing unit (CPU)
- Clock generator (CG)
- Bus interface unit (BIU)
- Bus arbitration unit (BAU)
- Refresh control unit (RCU)
- Wait control unit (WCU)
- Timer/counter unit (TCU)
- Serial control unit (SCU)
- Interrupt control unit (ICU)
- DMA control unit (DMAU)

Central Processing Unit

The μPD70216 CPU functions similarly to the CPU of the μPD70116 CMOS microprocessor. However, because the μPD70216 has internal peripheral devices, its bus architecture has been modified to permit sharing the bus with internal peripherals. The μPD70216 CPU is object code compatible with both the μPD70108/μPD70116 and the μPD8086/μPD8088 microprocessors.

Figure 1 is the μPD70216 CPU block diagram. A listing of the μPD70216 instruction set is in the final sections of this data sheet.

Figure 1. μPD70216 CPU Block Diagram



3d

Register Configuration

Program Counter [PC]. The program counter is a 16-bit binary counter that contains the program segment offset of the next instruction to be executed. The PC is incremented each time the microprogram fetches an instruction from the instruction queue. The contents of the PC are replaced whenever a branch, call, return, or break instruction is executed and during interrupt processing. At this time, the contents of the PC are the same as the prefetch pointer (PFP).

Prefetch Pointer [PFP]. The prefetch pointer is a 16-bit binary counter that contains the program segment offset of the next instruction to be fetched for the instruction queue. Because instruction queue prefetch is independent of instruction execution, the contents of the PFP and PC are not always identical. The PFP is updated each time the bus interface unit (BIU) fetches an instruction for the instruction queue. The contents of the PFP are replaced whenever a branch, call, return or break instruction is executed and during interrupt processing. At this time, the contents of the PFP and PC are the same.

Segment Registers [PS, SS, DS_0, DS_1]. The μPD70216 memory address space is divided into 64K-byte logical segments. A memory address is determined by the sum of a 20-bit base address (obtained from a segment register) and a 16-bit offset known as the effective address (EA). I/O address space is not segmented and no segment register is used. The four segment registers are program segment (PS), stack segment (SS), data segment 0 (DS_0), and data segment 1 (DS_1). The following table lists their offsets and overrides.

Default Segment Register	Offset	Override
PS	PFP register	None
SS	SP register	None
SS	Effective address (BP-based)	PS, DS_0, DS_1
DS_0	Effective address (non BP-based)	PS, SS, DS_1
DS_0	IX register (1)	PS, SS, DS_1
DS_1	IY register (2)	None

Note:

(1) Includes source block transfer, output, BCD string, and bit field extraction.

(2) Includes destination block transfer, input, BCD string, and bit field insertion.

General-Purpose Registers. The μPD70216 CPU contains four 16-bit general-purpose registers (AW, BW, CW, DW), each of which can be used as a pair of 8-bit registers by dividing into upper and lower bytes (AH, AL, BH, BL, CH, CL, DH, DL). General purpose registers may also be specified implicitly in an instruction. The implicit assignments are:

AW	Word multiplication/division, word I/O, data conversion
AL	Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
AH	Byte multiplication/division
BW	Translation
CW	Loop control, repeat prefix
CL	Shift/rotate bit counts, BCD operations
DW	Word multiplication/division, indirect I/O addressing

Pointer [SP, BP] and Index Registers [IX, IY]. These registers serve as base pointers or index registers when accessing memory using one of the base, indexed, or base indexed addressing modes. Pointer and index registers can also be used as operands for word data transfer, arithmetic, and logical instructions. These registers are implicitly selected by certain instructions as follows.

SP	Stack operations, interrupts
IX	Source block transfer, BCD string operations, bit field extraction
IY	Destination block transfer, BCD string operations, bit field insertion

Program Status Word [PSW]

The program status word consists of six status flags and four control flags.

Status Flags

- V (Overflow)
- S (Sign)
- Z (Zero)
- AC (Auxiliary Carry)
- P (Parity)
- CY (Carry)

Control Flags

- MD (Mode)
- DIR (Direction)
- IE (Interrupt Enable)
- BRK (Break)

When pushed onto the stack, the word image of the PSW is as follows:

15							8
MD	1	1	1	V	DIR	IE	BRK

7							0
S	Z	0	AC	0	P	1	CY

The status flags are set and cleared automatically depending upon the result of the previous instruction execution. Instructions are provided to set, clear, and complement certain status and control flags. Other flags can be manipulated by using the POP PSW instruction.

Between execution of the BRKEM and RETEM instructions, the native mode RETI and POP PSW instructions can modify the MD bit. Care must be exercised by emulation mode programs to prevent inadvertent alteration of this bit.

CPU Architectural Features

The major architectural features of the μPD70216 CPU are:

- Dual data buses
- Effective address generator
- Loop counter
- PC and PFP

Dual Data Buses. To increase performance, dual data buses (figure 2) have been employed in the CPU to fetch operands in parallel and avoid the bottleneck of a single bus. For two-operand instructions and effective address calculations, the dual data bus approach is 30 percent faster than single-bus systems.

Effective Address Generator. Effective address (EA) calculation requires only two clocks regardless of the addressing mode complexity due to the hardware effective address generator (figure 3). When compared with microprogrammed methods, the hardware approach saves between 3 and 10 clock cycles during effective address calculation.

Loop Counter and Shifters. A dedicated loop counter is used to count the iterations of block transfer and multiple shift instructions. This logic offers a significant performance advantage over architectures that control block transfers and multiple shifts using microprogramming. Dedicated shift registers also speed up the execution of the multiply and divide instructions. Compared with microprogrammed methods, multiply and divide instructions execute approximately four times faster.

Figure 2. Dual Data Buses

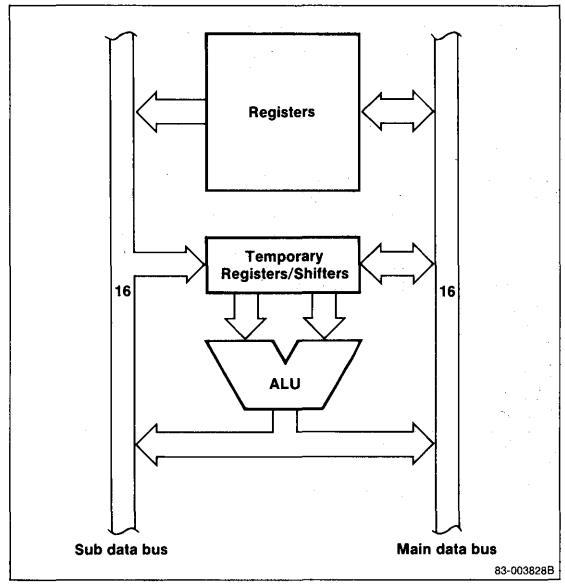
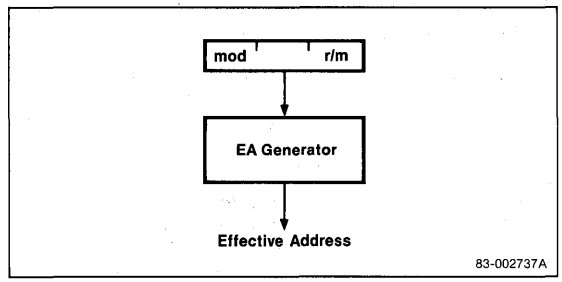


Figure 3. Effective Address Generator



Program Counter and Prefetch Pointer. The functions of instruction execution and queue prefetch are decoupled in the μPD70216. By avoiding a single instruction pointer and providing separate PC and PFP registers, the execution time of control transfers and the interrupt response latency can be minimized. Several clocks are saved by avoiding the need to readjust an instruction pointer to account for prefetching before computing the new destination address.

3d

Enhanced Instruction Set

In addition to the μPD8086/88 instruction set, the μPD70216 has added the following enhanced instructions.

Instruction	Function
PUSH imm	Push immediate data onto stack
PUSH R	Push all general registers onto stack
POP R	Pop all general registers from stack
MUL imm	Multiply register/memory by immediate data
SHL imm8	Shift/rotate by immediate count
SHR imm8	
SHRA imm8	
ROL imm8	
ROR imm8	
ROLC imm8	
RORC imm8	
CHKIND	Check array index
INM	Input multiple
OUTM	Output multiple
PREPARE	Prepare new stack frame
DISPOSE	Dispose current stack frame

Unique Instruction Set

In addition to the μPD70216 enhanced instruction set, the following unique instructions are supported.

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	BCD string addition
SUB4S	BCD string subtraction
CMP4S	BCD string comparison
ROL4	Rotate BCD digit left
ROR4	Rotate BCD digit right
TEST1	Test bit
SET1	Set bit
CLR1	Clear bit
NOT1	Complement bit
REPC	Repeat while carry set
REPNC	Repeat while carry cleared
FPO2	Floating point operation 2

Bit Fields. Bit fields are data structures that range in length from 1 to 16 bits. Two separate operations on bit fields, insertion and extraction, with no restrictions on the position of the bit field in memory are supported. Separate segment, byte offset, and bit offset registers are used for bit field insertion and extraction. Because of their power and flexibility, these instructions are highly effective for graphics, high-level languages, and data packing/unpacking applications.

Insert bit field (INS) copies the bit field of specified length (0 = 1 bit, 15 = 16 bits) from the AW register to the bit field addressed by DS1:IY:reg8 (figure 4). The bit field length can be located in any byte register or supplied as an immediate value. The value in reg8 is a bit field offset. A content of 0 selects bit 0 and 15 selects bit 15 of the word that DS0:IX points to. Following execution, the IY and bit offset register are updated to point to the start of the next bit field.

Bit field extraction (EXT) copies the bit field of specified length (0 = 1 bit, 15 = 16 bits) from the bit field addressed by DS0:IX:reg8 to the AW register (figure 5). If the bit field is less than 16 bits, it is right justified with a zero fill. The bit field length can be located in any byte register or supplied as immediate data. The value in reg8 is a bit field offset. A content of 0 selects bit 0 and 15 selects bit 15 of the word that DS0:IX points to. Following execution, the IX and bit offset register are updated to point to the start of the next bit field.

Packed BCD Strings. These instructions are provided to efficiently manipulate packed BCD data as strings (length from 1 to 254 digits) or as a byte data type with a single instruction.

BCD string arithmetic is supported by the ADD4S, SUB4S, and CMP4S instructions. These instructions allow the source string (addressed by DS0:IX) and the destination string (addressed by DS1:IY) to be manipulated with a single instruction. When the number of BCD digits is even, the Z and CY flags are set according to the result of the operation. If the number of digits is odd, the Z flag will not be correctly set unless the upper 4 bits of the result are zero. The CY flag will not be correctly set unless there is a carry out of the upper 4 bits of the result.

The two BCD rotate instructions (ROR4, ROL4) perform rotation of a single BCD digit in the lower half of the AL register through the register or memory operand.

Bit Manipulation. Four bit manipulation instructions have been added to the μPD70216 instruction set. The ability to test, set, clear, or complement a single bit in a register or memory operand increases code readability as well as performance over the logical operations traditionally used to manipulate bit data.

Repeat Prefixes. Two repeat prefixes (REPC, REPNC) allow conditional block transfer instructions to use the state of the CY flag as a terminating condition. The use of these prefixes allows inequalities to be used when working on ordered data, increasing the performance of searching and sorting algorithms.

Figure 4. Bit Field Insertion

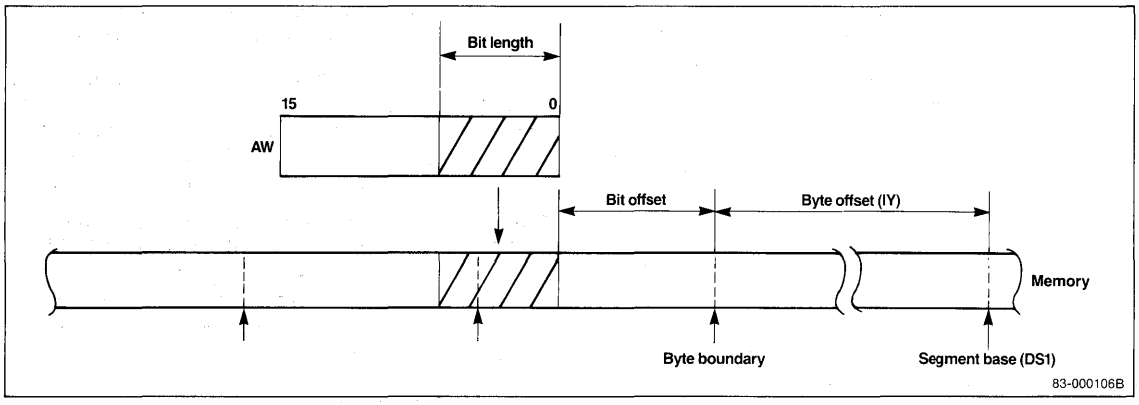
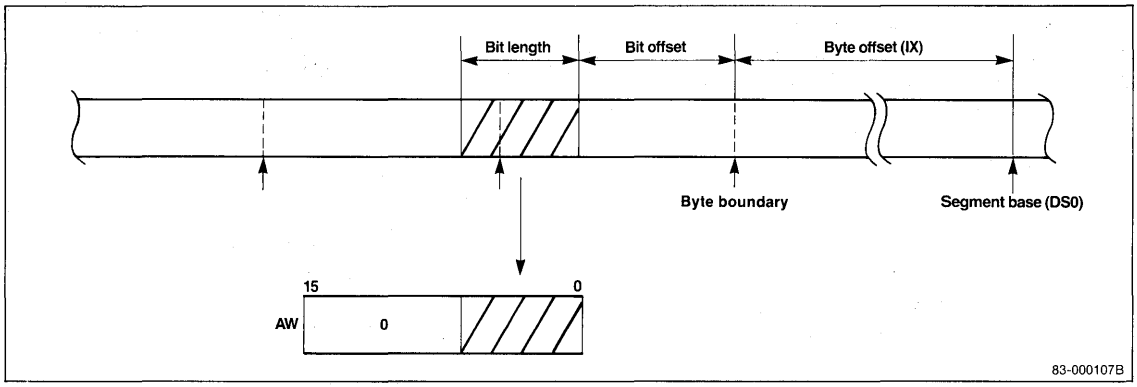


Figure 5. Bit Field Extraction.



Floating Point Operation Instructions. Two floating point operation (FPO) instruction types are recognized by the μPD70216 CPU. These instructions are detected by the CPU, which performs any auxiliary processing such as effective address calculation and the initial bus cycle if specified by the instruction. It is the responsibility of the external coprocessor to latch the address information and data (if a read cycle) from the bus and complete the execution of the instruction.

8080 Emulation Mode. The μPD70216 CPU can operate in either of two modes; see figure 6. Native mode allows the execution of the μPD8086/88, enhanced and unique instructions. The other operating mode is 8080 emulation mode, which allows the entire μPD8080AF instruction set to be executed. A mode (MD) flag is provided to distinguish between the two operating modes. Native mode is active when MD is 1 and 8080 emulation mode is active when MD is 0.

Two instructions are provided to switch from native to 8080 emulation mode and return back. Break for emulation (BRKEM) operates similarly to a BRK instruction, except that after the PSW has been pushed on the native mode stack, the MD flag becomes write-enabled and is cleared. During 8080 emulation mode, the registers and flags of the 8080 are mapped onto the native mode registers and flags as shown below. Note that PS, SS, DS_0, DS_1, IX, IY, AH and the upper half of the PSW registers are inaccessible to 8080 programs.

During 8080 emulation mode, the BP register functions as the 8080 stack pointer. The use of separate stack pointers prevents inadvertent damage to the native mode stack pointer by emulation mode programs.

The 8080 emulation mode PC is combined with the PS register to form the 20-bit physical address. All emulation mode data references use DS0 as the segment register. For compatibility with older 8080 software these registers must be equal. By using different segment register contents, separate 64K-byte code and data spaces are possible.

	μPD8080AF	μPD70216
Registers	A/PSW	AL/PSW (lower)
	B	CH
	C	CL
	D	DH
	E	DL
	H	BH
	L	BL
	SP	BP
	PC	PC
Flags	C	CY
	Z	Z
	S	S
	P	P
	AC	AC

Either an NMI or maskable interrupt will cause the 8080 emulation mode to be suspended. The CPU pushes the PS, PC, and PSW registers on the native mode stack, sets the MD bit (indicating native mode), and enters the specified interrupt handler. When the return from interrupt (RETI) instruction is executed, the PS, PC, and PSW (containing MD=0) are popped from the native stack and execution in 8080 emulation mode continues. Reset will also force a return to native mode.

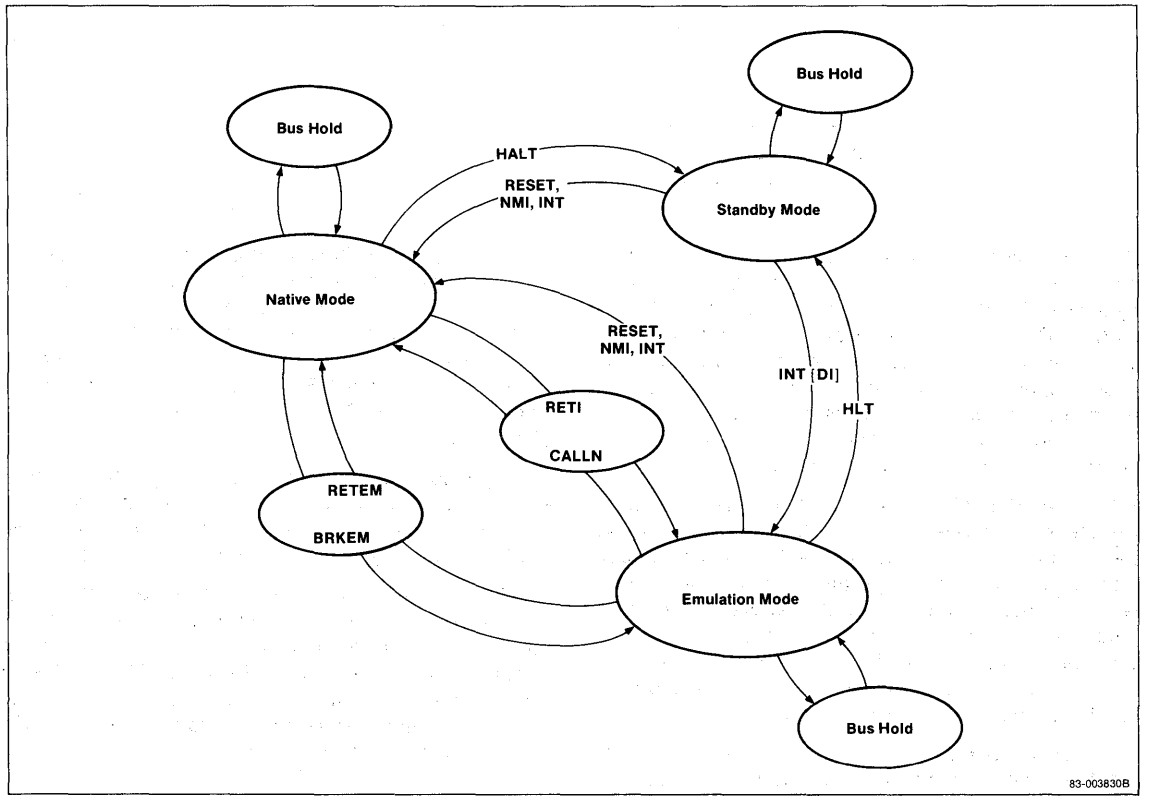
The 8080 emulation mode programs also have the capability to invoke native mode interrupt handlers by means of the call native (CALLN) instruction. This instruction operates like the BRK instruction except that the saved PSW indicates 8080 emulation mode.

To exit 8080 emulation mode, the return from emulation (RETEM) instruction pops the PS, PC, and PSW from the native mode stack, disables modification of the MD bit, and execution continues with the instruction following the BRKEM instruction. Nesting of 8080 emulation modes is prohibited.

Interrupt Operation

The μPD70216 supports a number of external interrupts and software exceptions. External interrupts are events asynchronous to program execution. On the other hand, exceptions always occur as a result of program execution.

Figure 6. μPD70216 Modes.



The two types of external interrupts are:

- Nonmaskable interrupt (NMI)
- Maskable interrupt (INT)

The six software exceptions are:

- Divide error (DIV, DIVU instructions)
- Array bound error (CHKIND instruction)
- Break on overflow (BRKV instruction)
- Break (BRK, BRK3 instructions)
- Single step (BRK bit in PSW set)
- Mode switch (BRKEM, CALLN instructions)

Interrupt vectors are determined automatically for exceptions and the NMI interrupt or supplied by hardware for maskable interrupts. The 256 interrupt vectors are stored in a table (figure 7) located at address 00000H. Vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. Interrupt vectors 32 to 255 are available for use by application software.

Each vector is made up of two words. The word located at the lower address contains the new PC for the interrupt handler. The word at the next-higher address is the new PS value for the interrupt handler. These must be initialized by software at the start of a program.

Nonmaskable interrupts and maskable interrupts (when enabled) are normally serviced following the execution of the current instruction. However, the following cases are exceptions to this rule and the occurrence of the interrupt will be delayed until after the execution of the next instruction.

- Moves to/from segment registers
- $\overline{\text{POLL}}$ instruction
- Instruction prefixes
- EI instruction (maskable interrupts only)

Another special case is the block transfer instructions. These instructions are interruptable and resumable, but because of the asynchronous operation of the BIU, the actual occurrence of the interrupt may be delayed up to three bus cycles later.

Standby Mode

The μPD70216 CPU has a low-power standby mode, which can dramatically reduce power consumption during idle periods. Standby mode is entered by simply executing a native or 8080 emulation HALT instruction; no external hardware is required. All other peripherals such as the timer/counter unit, refresh control unit, and DMA control unit continue to operate as programmed.

During standby mode, the clock is distributed only to the circuits required to release the standby mode. When a RESET, NMI, or INT event is detected, the standby mode is released. Both NMI and unmasked interrupts are processed before control returns to the instruction following the HALT. In the case of the INT input being masked, execution will begin with the instruction immediately following the HALT instruction without an intervening interrupt acknowledge bus cycle. When maskable interrupts are again enabled, the interrupt will be serviced.

Output signal states in the standby mode are listed below.

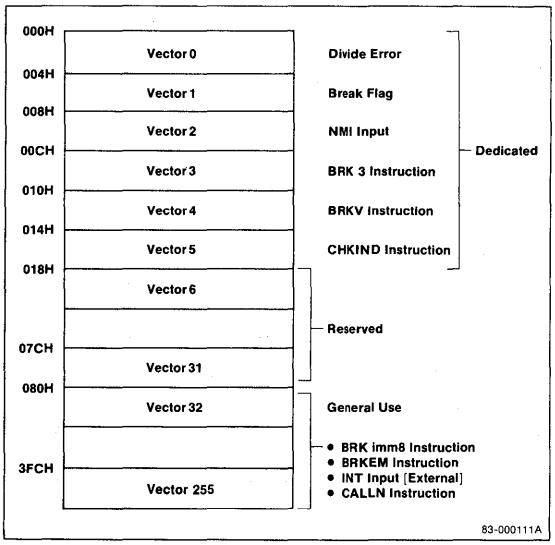
Output Signal	Status in Standby Mode
$\overline{\text{INTAK}}$, $\overline{\text{BUFEN}}$, $\overline{\text{MRD}}$, $\overline{\text{MWR}}$, $\overline{\text{IOWR}}$, $\overline{\text{IORD}}$ $\overline{\text{UBE}}$	High level
BS_2-BS_0 (Note 2)	Sends halt status (011), then remains high (111)
QS_1-QS_0, ASTB	Low level
$\overline{\text{BUSLOCK}}$	High level (low level if the $\overline{\text{HALT}}$ instruction follows the $\overline{\text{BUSLOCK}}$ prefix)
$\overline{\text{BUFR}}$/W, A_{19}-A_{16}/PS_3-PS_0, AD_{15}-AD_0	High or low level

Note:

(1) Output pin states during refresh and DMA bus cycles will be as defined for those operations.

(2) Halt status is presented prior to entering the passive state.

Figure 7. Interrupt Vector Table



Clock Generator

The clock generator (CG) generates a clock signal half the frequency of a parallel-resonant, fundamental mode crystal connected to pins X1 and X2. Figure 8 shows the recommended circuit configuration. Capacitors C1 and C2, required for frequency stability, are selected to match the crystal load capacitance.

External clock sources are also accommodated as shown in figure 9. The CG distributes the clock to the CLKOUT pin and to each functional block of the μPD70216. The generated clock signal has a 50-percent duty cycle.

Bus Interface Unit

The bus interface unit (BIU) controls the external address, data, and control buses for the three internal bus masters: CPU, DMA control unit (DMAU), and refresh control unit (RCU). The BIU is also responsible for synchronization of the $\overline{\text{RESET}}$ and READY inputs with the clock. The synchronized reset signal is used internally by the μPD70216 and provided externally at the RESOUT pin as a system-wide reset. The synchronized READY signal is combined with the output of the wait control unit (WCU) and is distributed internally to the CPU, DMAU, and RCU. Figure 10 shows the synchronization of $\overline{\text{RESET}}$ and READY.

The BIU also has the capability of overlapping the execution of the next instruction with memory write bus cycles. There is no overlap of instruction execution with read or I/O write bus cycles.

Figure 9. External Oscillator Configuration

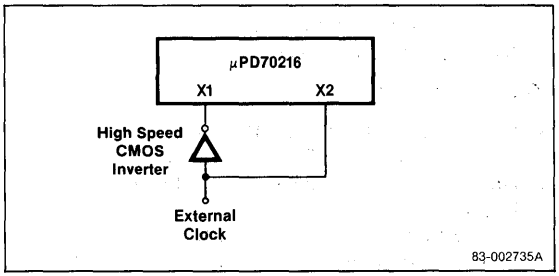


Figure 10. $\overline{\text{RESET}}$/READY Synchronization

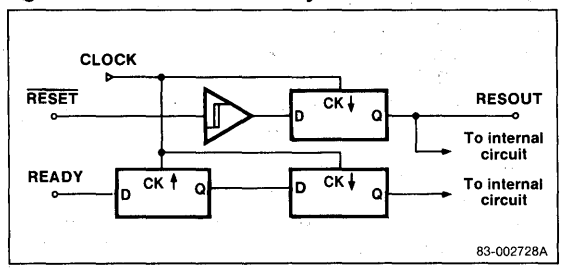
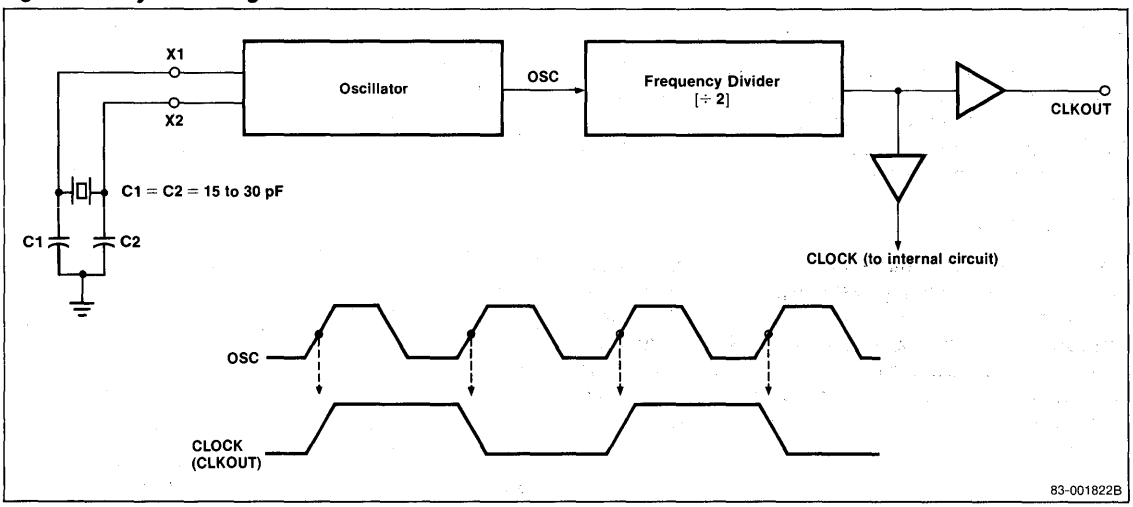


Figure 8. Crystal Configuration



Bus Arbitration Unit

The bus arbitration unit (BAU) arbitrates the external address, data, and control buses between the internal CPU, DMAU, and RCU bus requesters and an external bus master. The BAU bus priorities from the highest priority requester to the lowest are:

RCU (Demand mode)
DMAU
HLDRQ
CPU
RCU (Normal mode)

Note that RCU requests the bus at either the highest or lowest priority depending on the status of the refresh request queue. Bus masters other than the CPU are prohibited from using the bus when the CPU is executing an instruction containing a $\overline{\text{BUSLOCK}}$ prefix. Therefore, caution should be exercised when using the $\overline{\text{BUSLOCK}}$ prefix with instructions having a long execution time.

If a bus master with higher priority than the current bus master requests the bus, the BAU inactivates the current bus master's acknowledge signal. When the BAU sees the bus request from the current master go inactive, the BAU gives control of the bus to the higher priority bus master. Whenever possible, the BAU performs bus switching between internal bus masters without the introduction of idle bus cycles, enhancing system throughput.

System I/O Area

The I/O address space from addresses FF00H to FFFFH is reserved for use as the system I/O area. Located in this area are the 12 μPD70216 registers that determine the I/O addressing, enable/disable peripherals, and control pin multiplexing. Byte I/O instructions must be used to access the system I/O area.

I/O Address	Register	Operation
FFFFH	Reserved	—
FFFEH	OPCN	Read/Write
FFFDH	OPSEL	Read/Write
FFFCH	OPHA	Read/Write
FFFBH	DULA	Read/Write
FFFAH	IULA	Read/Write
FFF9H	TULA	Read/Write
FFF8H	SULA	Read/Write
FFF7H	Reserved	—
FFF6H	WCY2	Read/Write
FFF5H	WCY1	Read/Write
FFF4H	WMB	Read/Write
FFF3H	Reserved	—
FFF2H	RFC	Read/Write
FFF1H	Reserved	—
FFF0H	TCKS	Read/Write

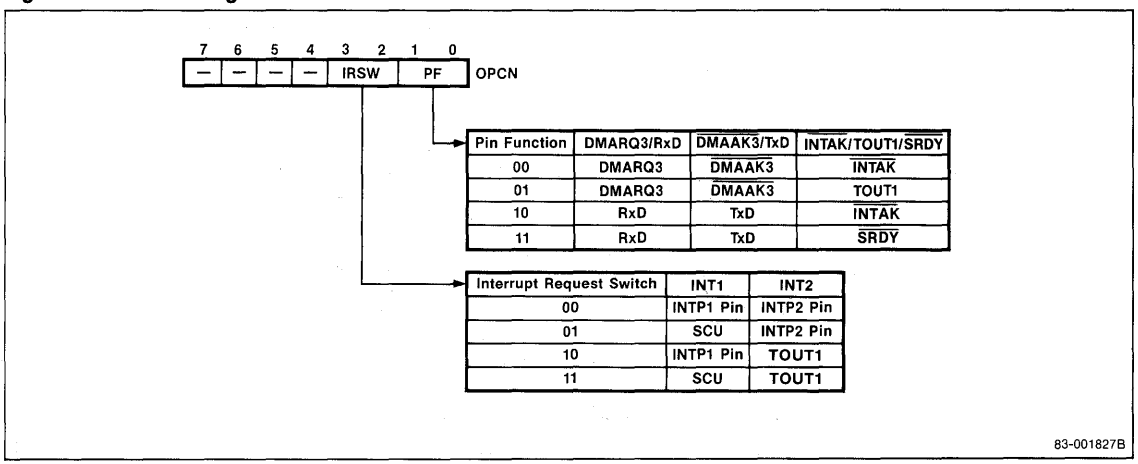
3d

On-Chip Peripheral Connection Register

The on-chip peripheral connection (OPCN) register controls multiplexing of the μPD70216 multiplexed pins. Figure 11 shows the format of the OPCN register. The interrupt request switch (IRSW) field controls multiplexing of ICU interrupt inputs INT1 and INT2. The output of an internal peripheral or an external interrupt source can be selected as the INT1 and INT2 inputs to the ICU.

The pin function (PF) field in the OPCN selects one of four possible states for the DMARQ3/RxD, $\overline{\text{DMAAK3}}$/TxD, and $\overline{\text{INTAK}}$/TOUT1/$\overline{\text{SRDY}}$ pins. Bit 0 of the

Figure 11. OPCN Register Format



Pin Function	DMARQ3/RxD	$\overline{\text{DMAAK3}}$/TxD	$\overline{\text{INTAK}}$/TOUT1/$\overline{\text{SRDY}}$
00	DMARQ3	$\overline{\text{DMAAK3}}$	$\overline{\text{INTAK}}$
01	DMARQ3	$\overline{\text{DMAAK3}}$	TOUT1
10	RxD	TxD	$\overline{\text{INTAK}}$
11	RxD	TxD	$\overline{\text{SRDY}}$

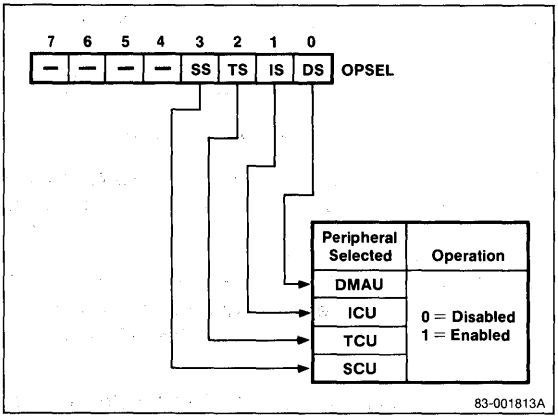
Interrupt Request Switch	INT1	INT2
00	INTP1 Pin	INTP2 Pin
01	SCU	INTP2 Pin
10	INTP1 Pin	TOUT1
11	SCU	TOUT1

OPCN controls the function of the $\overline{\text{INTAK}}/\overline{\text{SRDY}}$/TOUT1 pin. If cleared, $\overline{\text{INTAK}}$ will appear on this output pin. If bit 0 is set, either TOUT1 or $\overline{\text{SRDY}}$ will appear at the output depending on the state of bit 1. If bit 1 is cleared, DMA channel 3 I/O signals will appear on the DMARQ3/RxD and $\overline{\text{DMAAK3}}$/TxD pins. If the SCU is to be used, bit 1 of the PF field must be set.

On-Chip Peripheral Selection Register

The on-chip peripheral selection (OPSEL) register is used to enable or disable the μPD70216 internal peripherals. Figure 12 shows the format of the OPSEL register. Any of the four (DMAU, TCU, ICU, SCU) peripherals can be independently enabled or disabled by setting or clearing the appropriate OPSEL bit.

Figure 12. OPSEL Register Format



Internal Peripheral Relocation Registers

The five internal peripheral relocation registers (figure 13) are used to fix the I/O addresses of the DMAU, ICU, TCU, and SCU. The on-chip peripheral high-address (OPHA) register is common to all four internal peripherals and fixes the high-order byte of the 16-bit I/O address. The individual DMAU low-address (DULA) register, ICU low-address (IULA) register, TCU low-address (TULA) register, and the SCU low-address (SULA) register select the low-order byte of the I/O addresses for the DMAU, ICU, TCU, and SCU peripherals.

The contents of the OPHA register are:

OPHA

7							0
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

The formats for the individual internal peripheral registers appear below. Since address checking is not performed, do not overlap two peripheral I/O address spaces.

DULA

7							0
A_7	A_6	A_5	A_4	—	—	—	—

IULA

7							0
A_7	A_6	A_5	A_4	A_3	—	—	A_0

TULA

7							0
A_7	A_6	A_5	A_4	A_3	—	—	A_0

SULA

7							0
A_7	A_6	A_5	A_4	A_3	—	—	A_0

Figure 13. μPD70216 Peripheral Relocation

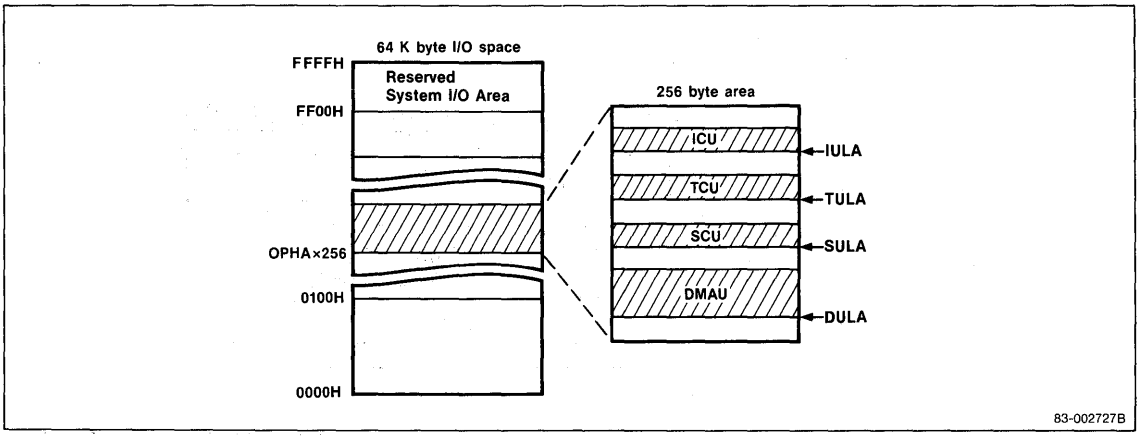
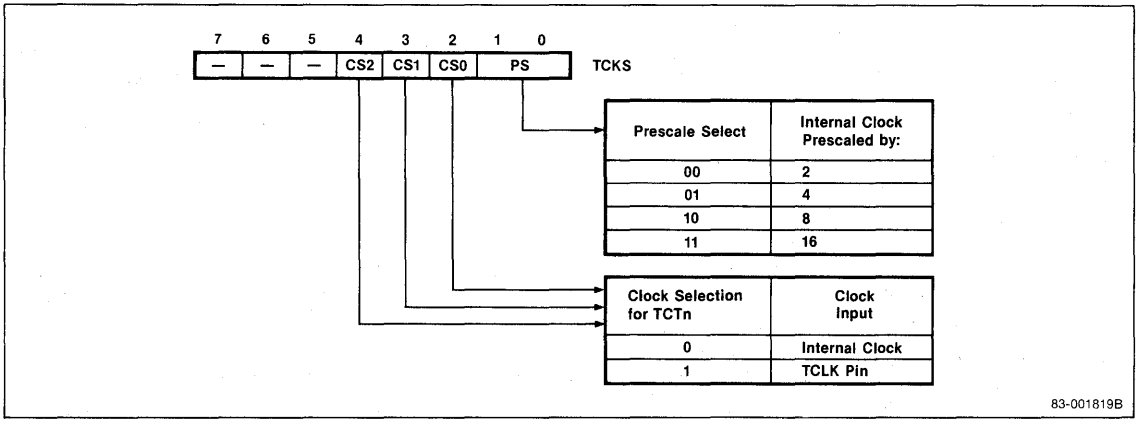


Figure 14. Timer Clock Selection Register



Timer Clock Selection Register

The timer clock selection (TCKS) register selects the clock source for each of the timer/counters as well as the divisor for the internal clock prescaler. Figure 14 shows the format of the TCKS register. The clock source for each timer/counter is independently selected from either the prescaled internal CPU clock or from an external clock source (TCLK). The internal clock is derived from the CLKOUT signal and can be divided by 2, 4, 8, or 16 before being presented to the clock select logic.

Refresh Control Unit

The refresh control unit (RCU) refreshes external dynamic RAM devices by outputting an 8-bit row address on address lines A_8-A_1 and performing a word-aligned memory read bus cycle. Both $\overline{UBE}$ and A_0 are asserted to allow the refresh of both the even and odd memory banks. External logic can distinguish a refresh bus cycle by monitoring the refresh request ($\overline{REFRQ}$) pin. Following each refresh bus cycle, the refresh row counter is incremented. The refresh control (RFC) register in the system I/O area contains two fields. The refresh enable field enables or disables the refreshing function. The refresh timer (RTM) field selects a refresh interval to match the dynamic memory refresh requirements. Figure 15 shows the format for the RFC register.

To minimize the impact of refresh on the system bus bandwidth, the μPD70216 utilizes a refresh request queue to store refresh requests and perform refresh bus cycles in otherwise idle bus cycles.

The RCU normally requests the bus as the lowest-priority bus requester (normal mode). However, if seven refresh requests are allowed to accumulate in the RCU refresh request queue, the RCU will change to the highest-priority bus requester (demand mode). The RCU will then perform back-to-back refresh cycles until three requests remain in the queue. This guarantees the integrity of the DRAM system while maximizing performance.

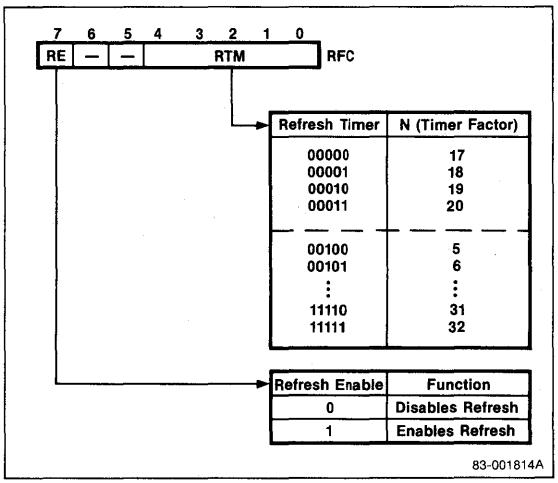
The refresh count interval can be calculated as follows:

Refresh interval = 8 x N x t_{CYK}

where N is the timer factor selected by the RTM field.

When the μPD70216 is reset, the RE field in the RFC register is unaffected and the RTM field is set to 01000 (N = 9). No refresh bus cycles occur while $\overline{RESET}$ is asserted.

Figure 15. Refresh Control Register



Wait Control Unit

The wait control unit (WCU) inserts from zero to three wait states into a bus cycle in order to compensate for the varying access times of memory and I/O devices. The number of wait states for CPU, DMAU, and RCU bus cycles is separately programmable. In addition, the memory address space is divided into three independent partitions to accommodate a wide range of system designs. $\overline{\text{RESET}}$ initializes the WCU to insert three wait states in all bus cycles. This allows operation with slow memory and peripheral devices before the initialization of the WCU registers.

The three system I/O area registers that control the WCU are wait cycle 1 (WCY1), wait cycle 2 (WCY2), and wait state memory boundary (WMB). The WCU always inserts wait states corresponding to the wait count programmed in WCY1 or WCY2 registers into a bus cycle, regardless of the state of the external READY input. After the programmed number of wait states occurs, the WCU will insert Tw states as long as the READY pin remains inactive. When READY is again asserted, the bus cycle continues with T4 as the next cycle. The μPD70216 internal peripherals never require wait states; four clock cycles will terminate an internal peripheral bus cycle.

CPU Wait States

The WMB register divides the 1M-byte memory address space into three independent partitions: lower, middle, and upper. Figure 16 shows the WMB register format.

Initialization software can then set the number of wait states for each memory partition and the I/O partition via the WCY1 register (figure 17).

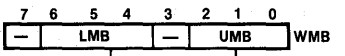
DMA and Refresh Wait States

The WCY2 register (figure 18) specifies the number of wait states to be automatically inserted in DMA and refresh bus cycles.

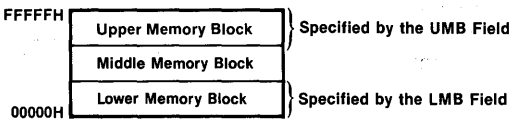
DMA and Refresh Wait States

The WCY2 register (figure 18) specifies the number of wait states to be automatically inserted in DMA and refresh bus cycles. DMA wait states must be set to the maximum of the DMA memory and I/O partitions. Refresh wait states should be set to the maximum value of all DRAM memory partitions.

Figure 16. Wait State Memory Boundary Register



Lower Memory Block Size [1] Upper Memory Block Size	Memory Block Size (KB)
000	32
001	64
010	96
011	128
100	192
101	256
110	384
111	512



Note:

[1] By default, the address space remaining between the UBM and LBM is the middle memory block.

83-001820B

Figure 17. Wait Cycle 1 Register

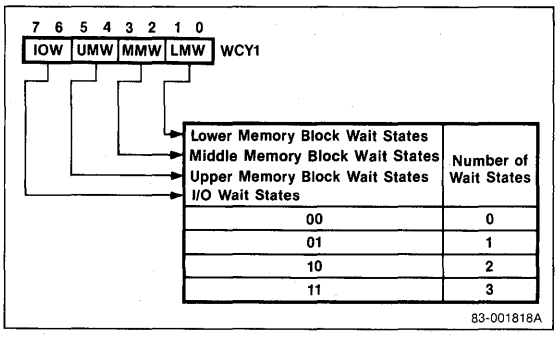
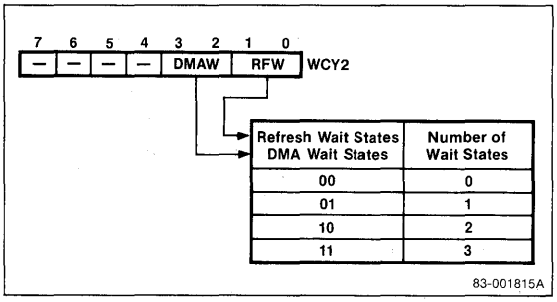


Figure 18. Wait Cycle 2 Register



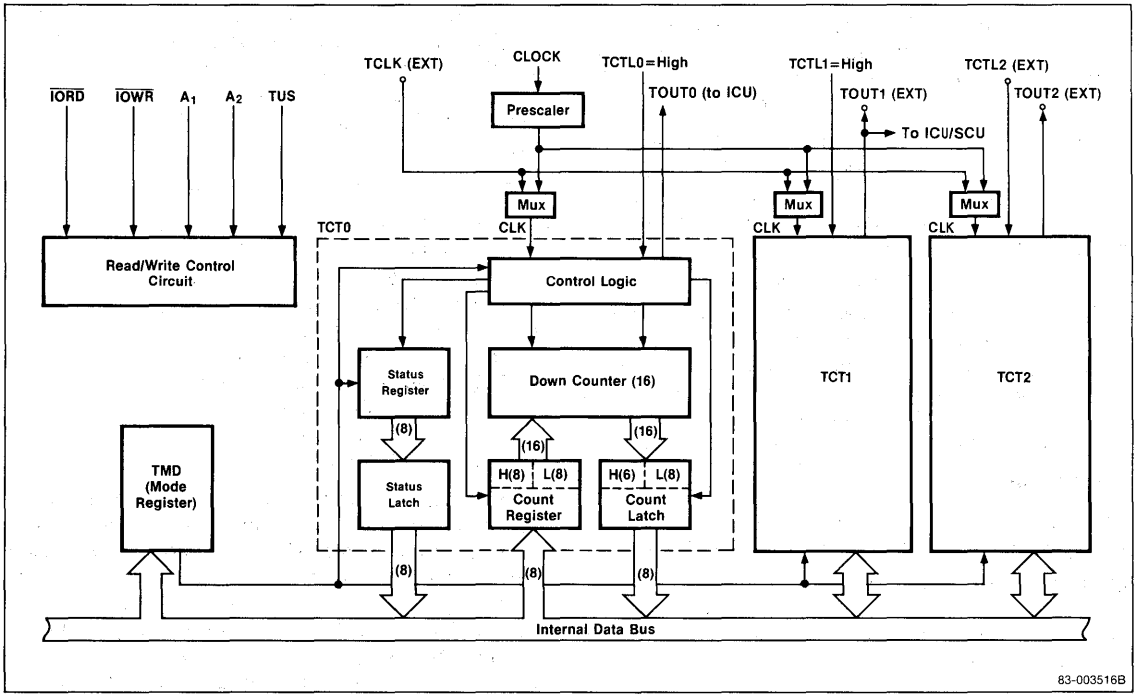
Timer/Counter Unit

The timer/counter unit (TCU) provides a set of three independent 16-bit timer/counters. The output signal of timer/counter 0 is hardwired internally as an interrupt source. The output of timer/counter 1 is available internally as an interrupt source, used as a baud rate generator, or used as an external output. The timer/counter 2 output is available as an external output. Due to mode restrictions, the TCU is a subset of the μPD71054 Programmable Timer/Counter. Figure 19 shows the internal block diagram of the TCU.

The TCU has the following features:

- Three 16-bit timer/counters
- Six programmable count modes
- Binary/BCD counting
- Multiple latch command
- Choice of two clock sources

Figure 19. TCU Block Diagram



Because $\overline{RESET}$ leaves the TCU in an uninitialized state, each timer/counter must be initialized by specifying an operating mode and a count. Once programmed, a timer/counter will continue to operate in that mode until another mode is selected. When the count has been written to the counter and transferred to the down counter, a new count operation starts. Both the current count and the counter status can be read while count operations are in progress.

TCU Commands

The TCU is programmed by issuing I/O instructions to the I/O port addresses programmed in the OPHA and TULA registers. The individual TCU registers are selected by address bits A_2 and A_1 as follows.

A_2	A_1	Register	Operation
0	0	TCT0 TST0	Read/Write Read
0	1	TCT1 TST1	Read/Write Read
1	0	TCT2 TST2	Read/Write Read
1	1	TMD	Write

The timer mode (TMD) register selects the operating mode for each timer/counter and issues the latch command for one or more timer/counters. Figure 20 shows the format for the TMD register.

Writes to the timer/counter 2-0 (TCT2-TCT0) registers stores the new count in the appropriate timer/counter. The count latch command is used before reading count data in order to latch the current count and prevent inaccuracies.

The timer status 2-0 (TST2-TST0) registers contain status information for the specified counter (figure 21). The latch command is used to latch the appropriate counter status before reading status information. If both status and counter data are latched for a counter, the first read operation returns the status data and subsequent read operations obtain the count data.

Count Modes

There are six programmable timer/counter modes. The timing waveforms for these modes are in figure 22.

Mode 0 [Interrupt on End of Count]. In this mode, TOUT changes from the low to high level when the specified count is reached. This mode is available on all timer/counters.

Mode 1 [Retriggerable One-shot]. In mode 1, a low-level one-shot pulse, triggered by TCTL2 is output from the TOUT2 pin. This mode is available only on timer/counter 2.

Mode 2 [Rate Generator]. In mode 2, TOUT cyclically goes low for one clock period when the counter reaches the 0001H count. A counter in this mode operates as a frequency divider. All timer/counters can operate using mode 2.

Mode 3 [Square Wave Generator]. Mode 3 is a frequency divider similar to mode 2, but the output has a symmetrical duty cycle. This mode is available on all three timer/counters.

Mode 4 [Software Triggered Strobe]. In mode 4, when the specified count is reached, TOUT goes low for the duration of one clock pulse. Mode 4 is available on all timer/counters.

Mode 5 [Hardware Triggered Strobe]. Mode 5 is similar to mode 4 except that operation is triggered by the TCTL2 input and can be retriggered. This mode is available only on timer/counter 2.

Serial Control Unit

The serial control unit (SCU) is a single asynchronous serial channel that performs serial communication between the μPD70216 and an external serial device. The SCU is similar to the μPD71051 Serial Control Unit except for the lack of synchronous communication protocols. Figure 23 is the block diagram of the SCU.

The SCU has the following features.

- Full-duplex asynchronous serial controller
- Clock rate divisor (x16, x64)
- Baud rates to 250 kb/s supported
- 7-, 8-bit character lengths
- 1-, 2-bit stop bit lengths
- Break transmission and detection
- Full-duplex, double-buffered transmitter/receiver
- Even, odd, or no parity
- Parity, overrun, and framing error detection
- Receiver-full/transmitter-empty interrupt

The SCU contains four separately addressable registers for reading/writing data, reading status, and controlling operation of the SCU. The serial receive buffer (SRB) and the serial transmit buffer (STB) store the incoming and outgoing character data. The serial status (SST) register allows software to determine the current state of both the transmitter and receiver. The serial command (SCM) and serial mode (SMD) registers determine the operating mode of the SCU while the serial interrupt mask (SIMK) register allows software control of the SCU receive and transmit interrupts.

Figure 20. Timer Mode Register

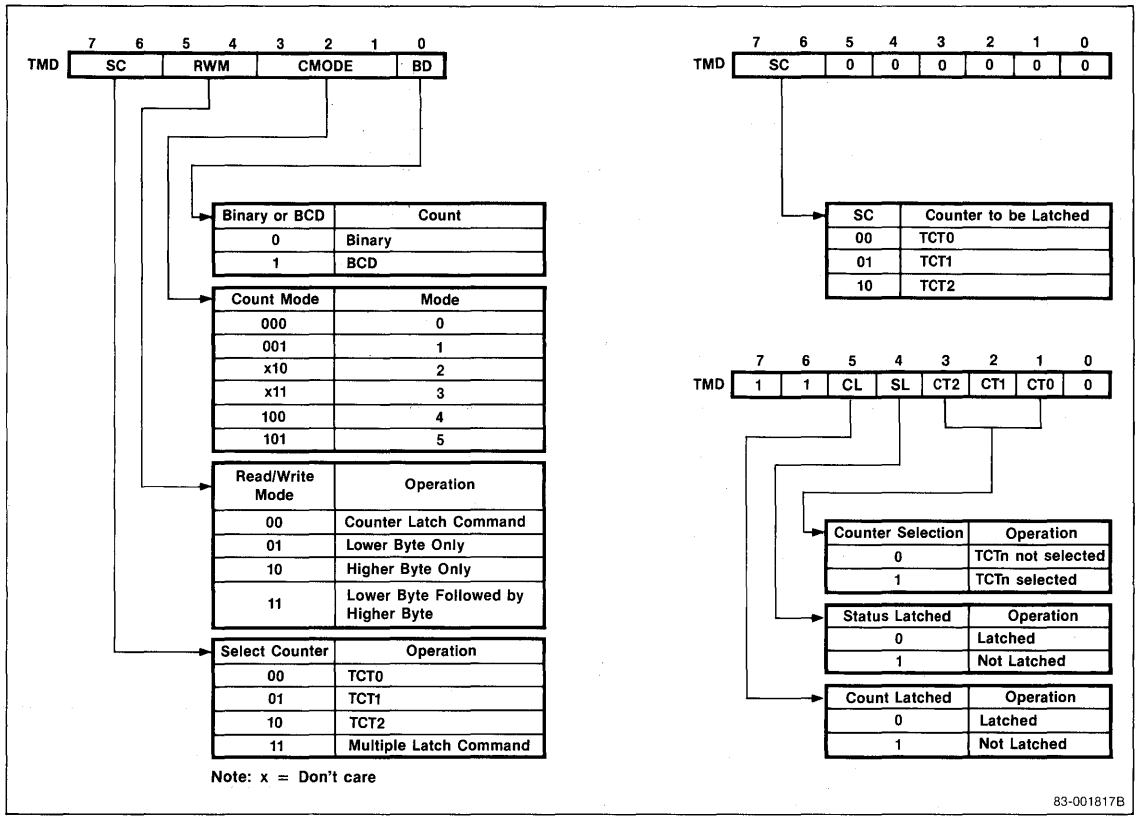
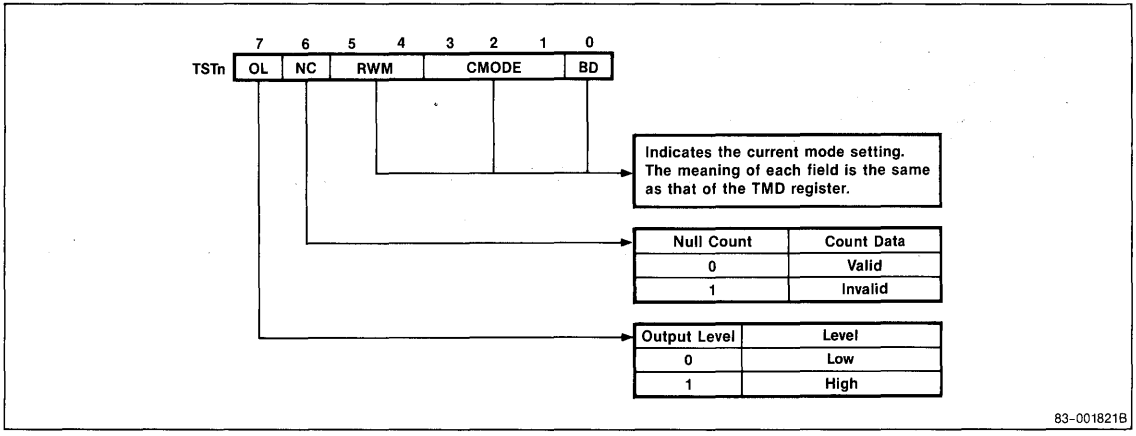


Figure 21. TCU Status Register



3d

Figure 22. TCU Waveforms (Sheet 1 of 3)

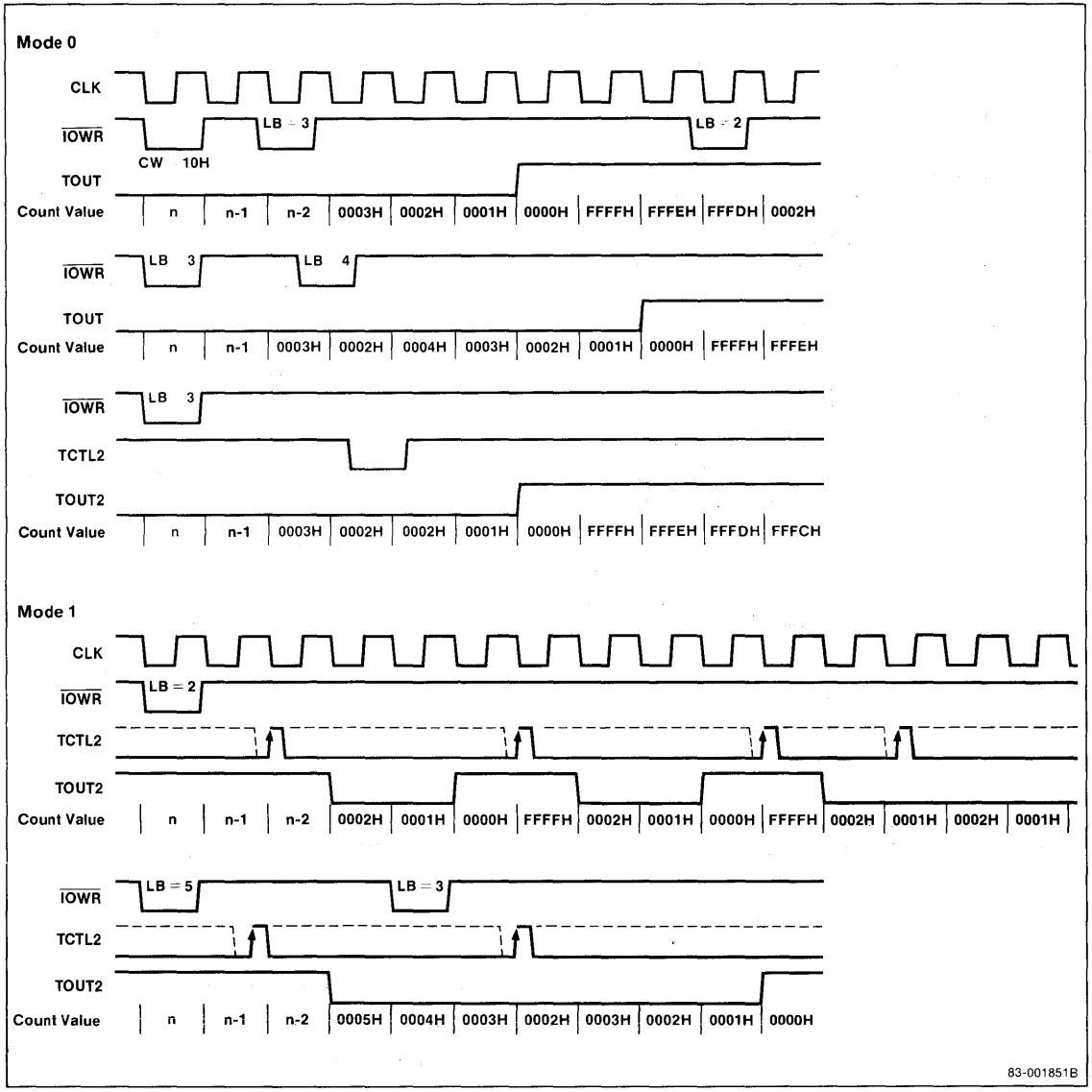
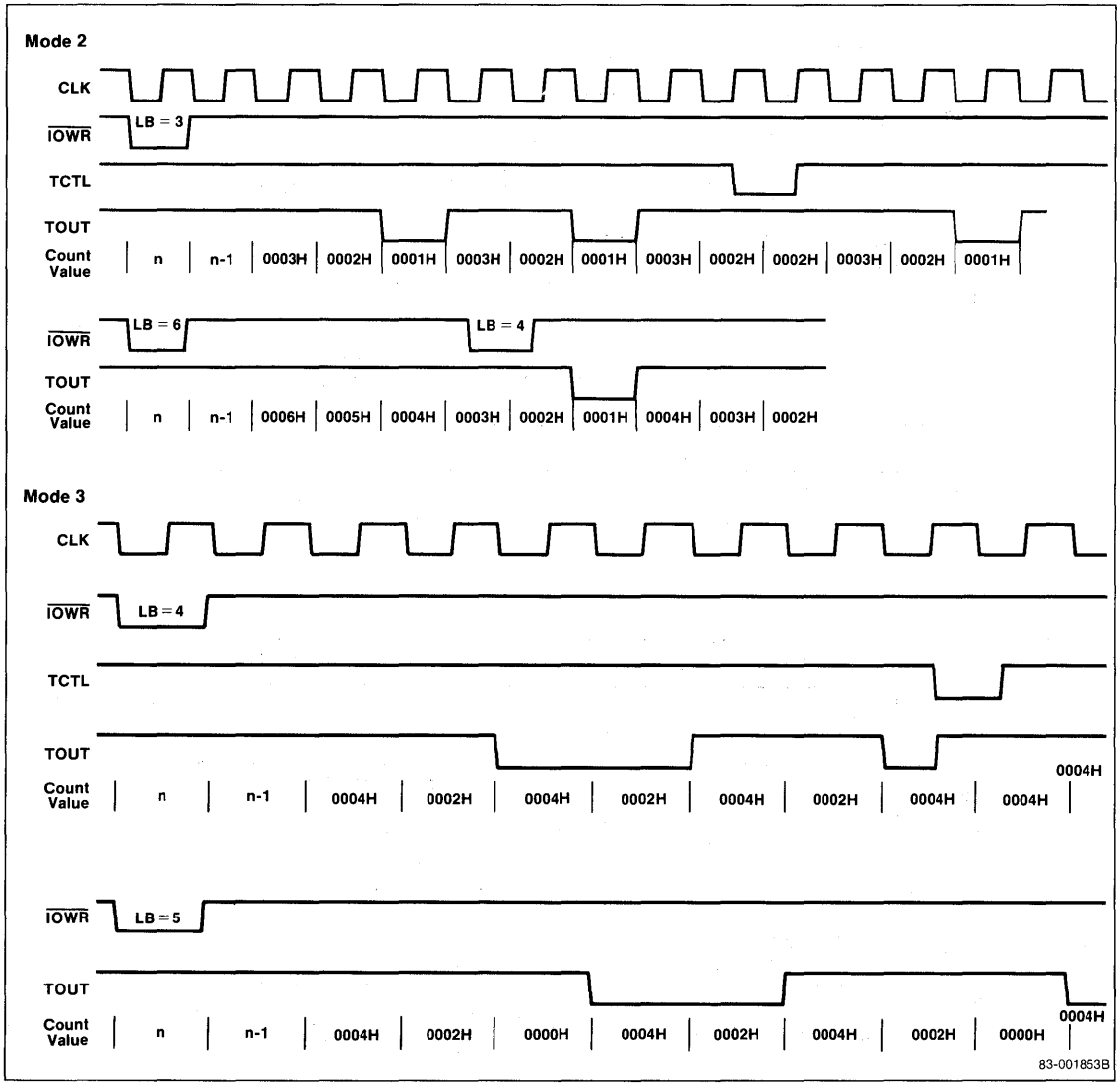


Figure 22. TCU Waveforms (Sheet 2 of 3)



3d

Figure 22. TCU Waveforms (Sheet 3 of 3)

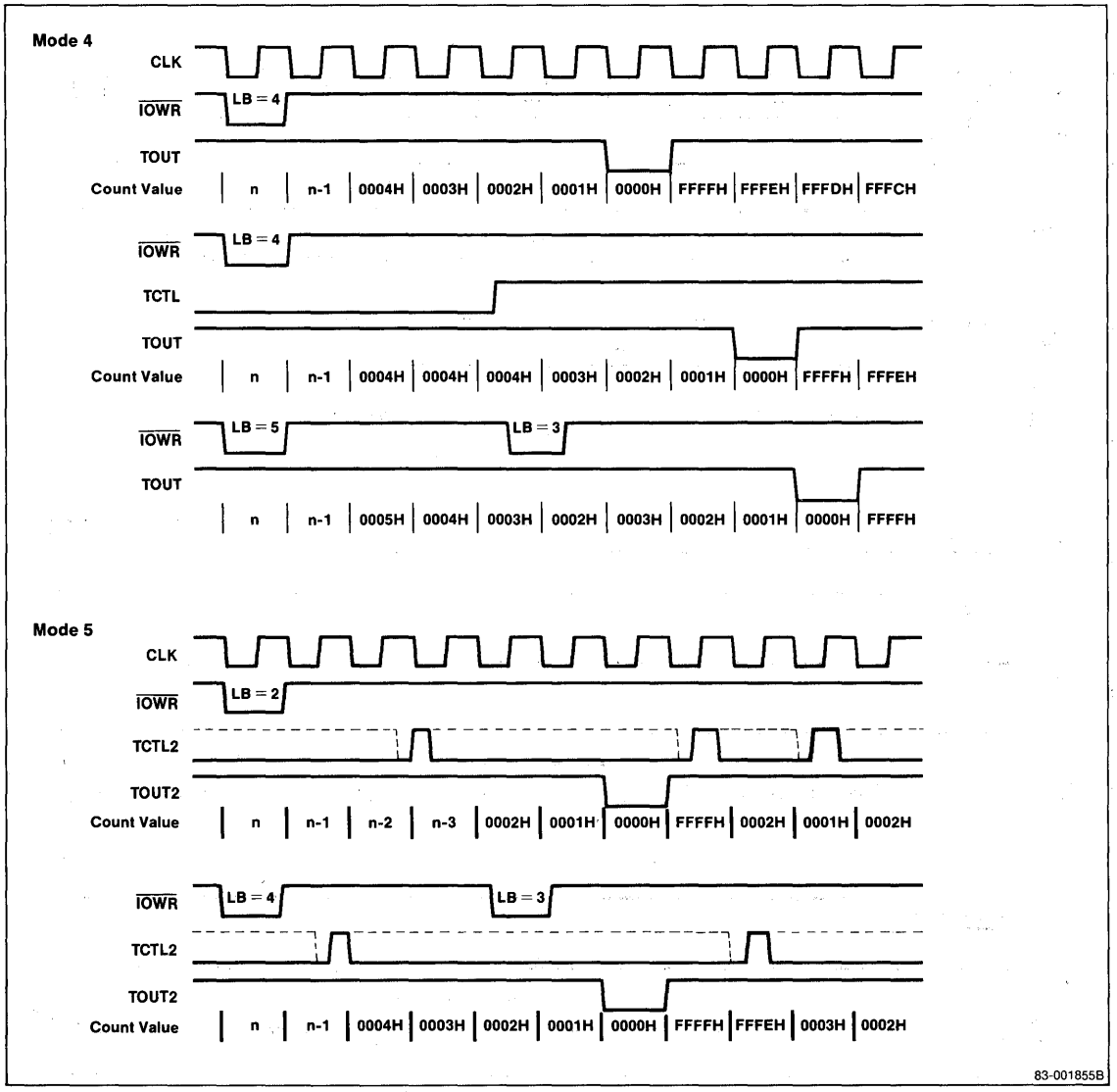
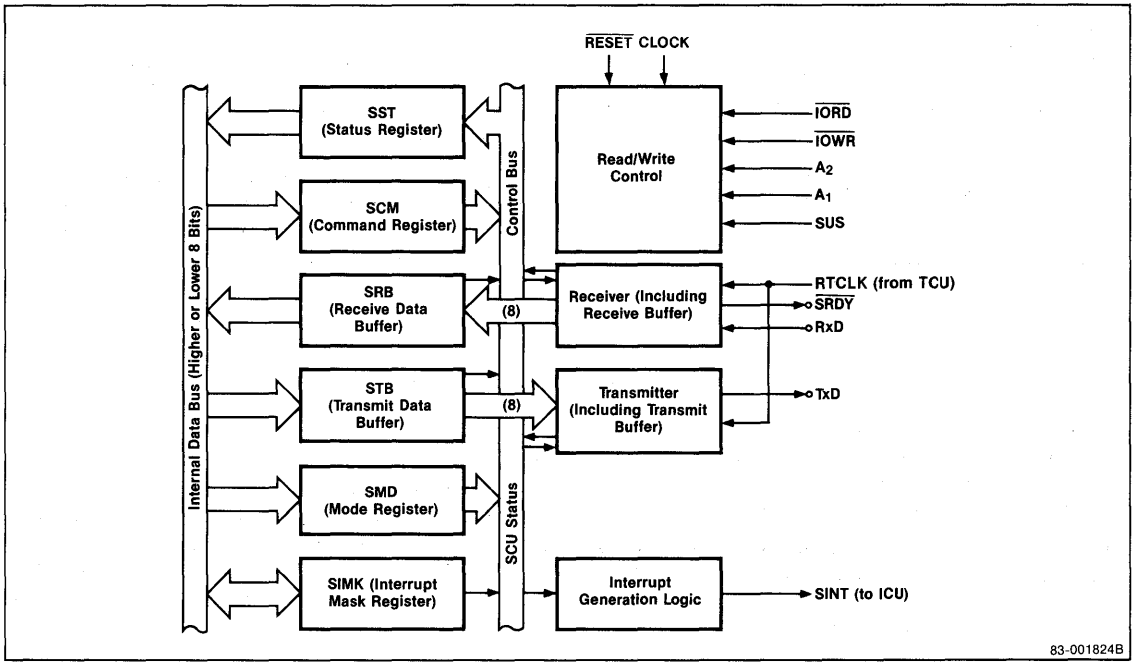


Figure 23. SCU Block Diagram



Receiver Operation

While the RxD pin is high, the receiver is in an idle state. A transition on RxD from high to low indicates the start of new serial data. When a complete character has been received, it is transferred to the SRB; the receive buffer ready (RBRDY) bit in the SST register is set and (if unmasked) an interrupt is generated. The SST also latches any parity, overrun, or framing errors at this time.

The receiver detects a break condition when a null character with zero parity is received. The BRK bit is set for as long as the subsequent receive data is low and resets when RxD returns to a high level. The MRDY bit (SCM) and RBRDY (SST) are gated to form the output $\overline{\text{SRDY}}$. $\overline{\text{SRDY}}$ prevents overruns from occurring when the program is unable to process the input data. Software can control MRDY to prevent data from being sent from the remote transmitter while RBRDY can prevent the immediate overrun of a received character.

Transmitter Operation

TxD is kept high while the STB register is empty. When the transmitter is enabled and a character is written to the STB register, the data is converted to serial format and output on the TxD pin. The start bit indicates the start of the transmission and is followed by the character stream (LSB to MSB) and an optional parity bit. One or two stop bits are then appended, depending on the programmed mode. When the character has been transferred from the STB, the TRBDY bit in the SST is set and if unmasked, a transmit buffer empty interrupt is generated.

Serial data can be transmitted and received by polling the SST register and checking the TBRDY or RBRDY flags. Data can also be transmitted and received by SCU-generated interrupts to the interrupt control unit. The SCU generates an interrupt in either of these conditions:

(1) The receiver is enabled, the SRB is full, and receive interrupts are unmasked.

(2) The transmitter is enabled, the STB is empty, and transmit interrupts are unmasked.

3d

SCU Registers and Commands

I/O instructions to the I/O addresses selected by the OPHA and SULA registers are used to read/write the SCU registers. Address bits A_1 and A_2 and the read/write lines select one of the six internal registers as follows:

A_2	A_1	Register	Operation
0	0	SRB STB	Read Write
0	1	SST SCM	Read Write
1	0	SMD	Write
1	1	SIMK	Read/write

The SRB and STB are 8-bit registers. When the character length is 7 bits, the lower 7 bits of the SRB register are valid and bit 7 is cleared to 0. If programmed for 7-bit characters, bit 7 of the STB is ignored.

The SST register (figure 24) contains the status of the transmit and receive data buffers and the error flags. Error flags are persistent. Once an error flag is set, it remains set until a clear error flags command is issued.

Figure 25 shows the SCM and SMD registers. The SCM register stores the command word that controls transmission, reception, error flag reset, break transmission, and the state of the $\overline{SRDY}$ pin. The SMD register stores the mode word that determines serial characteristics such as baud rate divisor, parity, character length, and stop bit length.

Initialization software should first program the SMD register followed by the SCM register. Unlike the μPD71051, the SMD register can be modified at any time without resetting the SCU.

The SIMK register (figure 26) controls the occurrence of RBRDY and TBRDY interrupts. When an interrupt is masked, it is prevented from propagating to the interrupt control unit.

Baud Rate Generator

Timer/counter 1 is used as the baud rate generator when the SCU is enabled. The input baud rate clock is scaled by 16 or 64, as selected in the SMD register, to determine the receive/transmit data clock. There are no restrictions on the SCU input baud rate clock other than operating the TCU in mode 3 with a square-wave output.

Figure 24. SST Register

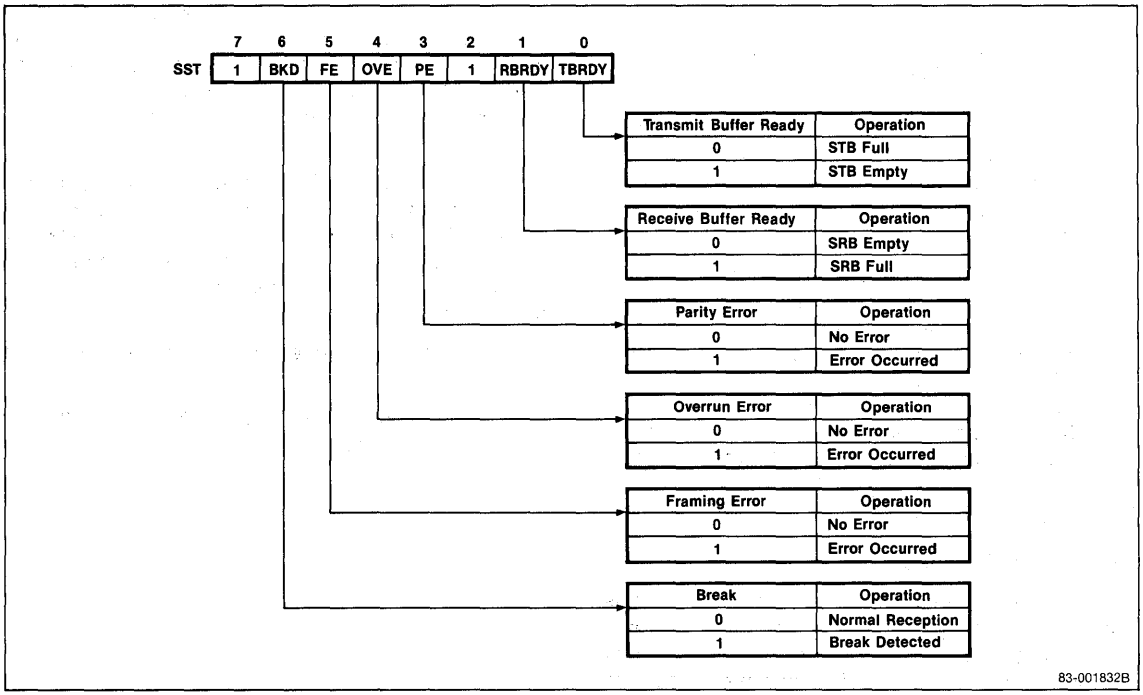


Figure 25. SCM and SMD Registers

SCM Register

	7	6	5	4	3	2	1	0
SCM	—	—	$\overline{\text{MRDY}}$	ECL	SBRK	RE	—	TE

Transmitter Enabled	Operation
0	Transmitter Disabled
1	Transmitter Enabled

Receiver Enabled	Operation
0	Receiver Disabled
1	Receiver Enabled

Send Break	Operation
0	Normal Operation
1	TxD = 0 (Break)

Error Clear	Operation
0	No Operation
1	Error Flag Clear

$\overline{\text{MRDY}}$	Mask Ready
0	$\overline{\text{SRDY}}$ = 1 (Mask)
1	Normal Operation of $\overline{\text{SRDY}}$ Output

SMD Register

	7	6	5	4	3	2	1	0
SMD	STL		PS		CL		BF	

Baud Rate Factor	Operation
0-	Illegal
10	RTCLK Frequency ÷ 16
11	RTCLK Frequency ÷ 64

Character Length	Operation
0-	Illegal
10	7 Bit Characters
11	8 Bit Characters

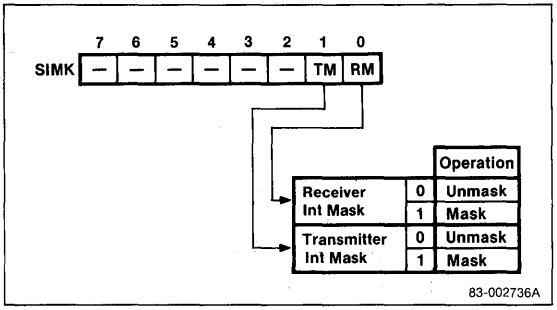
Parity Select	Operation
-0	No Parity
01	Odd Parity
11	Even Parity

Stop Bit Length	Operation
-0	Illegal
01	1 Stop Bit
11	2 Stop Bits

83-001836B

3d

Figure 26. SIMK Register



Interrupt Control Unit

The interrupt control unit (ICU) is a programmable interrupt controller equivalent to the μPD71059. The ICU arbitrates up to eight interrupt inputs, generates a CPU interrupt request, and outputs the interrupt vector number on the internal data bus during an interrupt acknowledge cycle. Cascading up to seven external slave μPD71059s permits the μPD70216 to support up to 56 interrupt sources. Figure 27 is the block diagram for the ICU.

The ICU has the following features.

- Eight interrupt request inputs
- Cascadable with μPD71059 Interrupt Controllers
- Programmable edge- or level-triggered interrupts (TCU, edge-triggered interrupts only)
- Individually maskable interrupt requests
- Programmable interrupt request priority
- Polling mode

ICU Registers

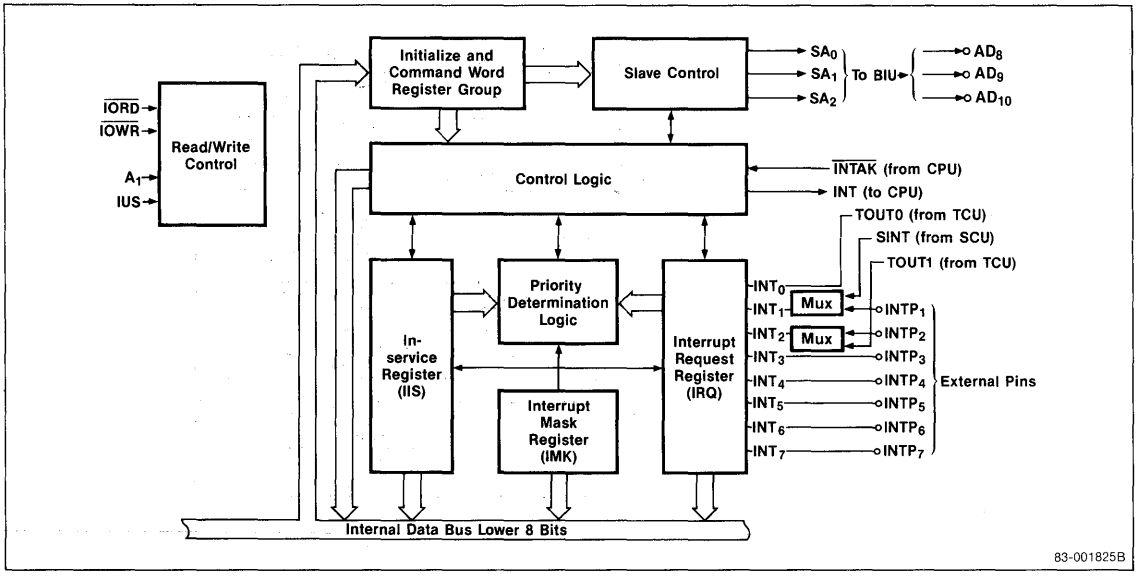
Use I/O instructions to the I/O addresses selected by the OPHA and IULA registers to read from and write to the ICU registers. Address bit A_1 and the command word selects an ICU internal register.

	A_1	Other Condition	Operation
Read	0	IMD selects IRQ	CPU ← IRQ data
	0	IMD selects IIS	CPU ← IIS data
	0	Polling phase	CPU ← Polling data
	1	—	CPU ← IMKW
Write	0	D4 = 1	CPU → IIW1
	0	D4 = 0 and D3 = 0	CPU → IPFW
	0	D4 = 0 and D3 = 1	CPU → IMDW
	1	During initialization	CPU → IIW2
	1		CPU → IIW3
	1		CPU → IIW4
	1	After initialization	CPU → IMKW

Note:

(1) In polling phase, polling data has priority over the contents of the IRQ or IIS register when read.

Figure 27. ICU Block Diagram



Initializing the ICU

The ICU is always used to service maskable interrupts in a μPD70216 system. Prior to accepting maskable interrupts, the ICU must first be initialized (figure 28). Following initialization, command words from the CPU can change the interrupt request priorities, mask/unmask interrupt requests, and select the polling mode. Figures 29 and 30 list the ICU initialization and command words.

Interrupt initialization words 1-4 (IIW1-IIW4) initialize the ICU, indicate whether external μPD71059s are connected as slaves, select the base interrupt vector, and select edge- or level-triggered inputs for INT1-INT7. Interrupt sources from the TCU are fixed as edge-triggering. INT0 is internally connected to TOUT0, and INT2 may be connected to TOUT1 by the IRSW field in the OPCN.

The interrupt mask word (IMKW) contains programmable mask bits for each of the eight interrupt inputs. The interrupt priority and finish word (IPFW) is used by the interrupt handler to terminate processing of an interrupt or change interrupt priorities. The interrupt mode word (IMDW) selects the polling register, interrupt request (IRQ) or interrupt in service (IIS) register, and the nesting mode.

The initialization words are written in consecutive order starting with IIW1. IIW2 sets the interrupt vector. IIW3 specifies which interrupts are connected to slaves. IIW3 is only required in extended systems. The ICU will only expect to receive IIW3 if SNGL = 0 (bit D_1 of IIW1). IIW4 is only written if II4 = 1 (bit D_0 of IIW1).

μPD71059 Cascade Connection

To increase the number of maskable interrupts, up to seven slave μPD71059 Interrupt Controllers can be cascaded. During cascade operation (figure 31), each slave μPD71059 INT output is routed to one of the μPD70216 INTP inputs. During the second interrupt acknowledge bus cycle, the ICU places the slave address on address lines AD_{10}-AD_8. Each slave compares this address with the slave address programmed using interrupt initialization word 3 (IIW3). If the same, the slave will place the interrupt vector on pins AD_7-AD_0 during the second interrupt acknowledge bus cycle.

Figure 28. Initialization Sequence

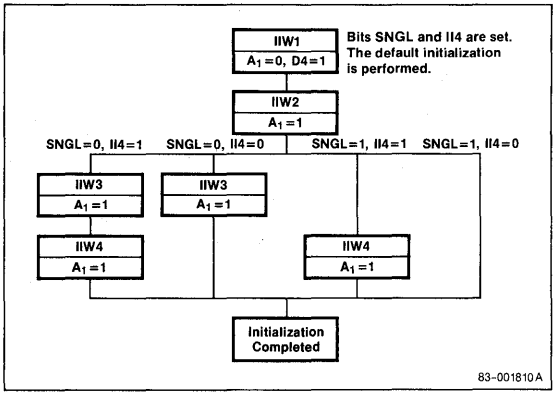
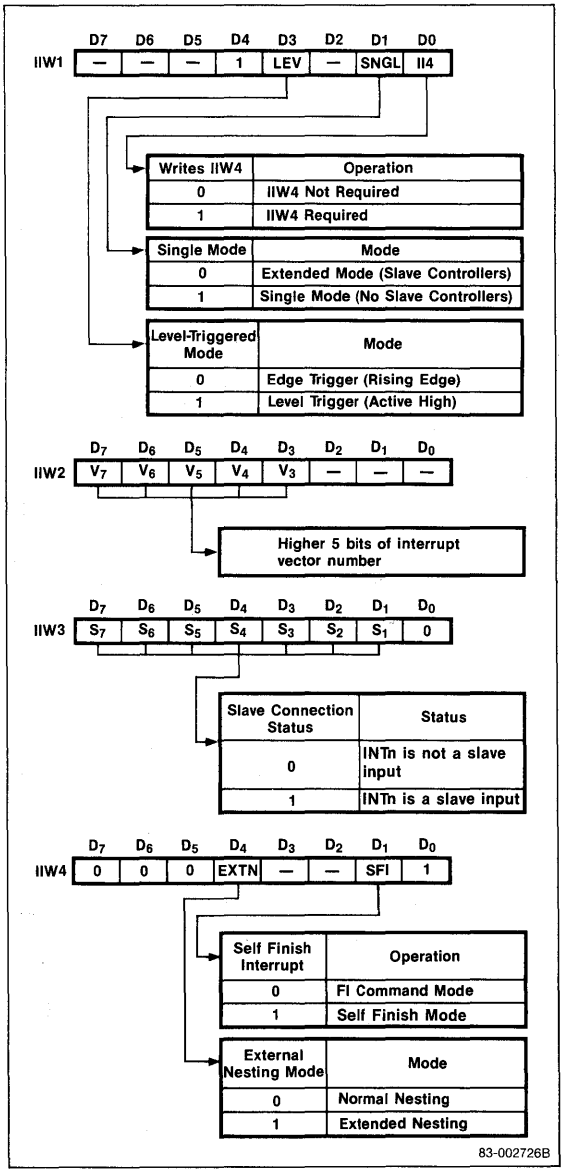


Figure 29. Interrupt Initialization Words 1-4



3d

Figure 30. Command Words

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
IMKW	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

Interrupt Request Mask	Operation
0	INTn not Masked
1	INTn Masked

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
IPFW	RP	SIL	FI	0	0	IL2	IL1	IL0

Interrupt Level	0	0	0	INT0
	0	0	1	INT1
	0	1	0	INT2
	0	1	1	INT3
	1	0	0	INT4
	1	0	1	INT5
	1	1	0	INT6
	1	1	1	INT7

Rotate Priority Specify Interrupt Level, Finish Interrupt Commands	0	0	1	FI Command	No Level Specification	No Rotation	Normal FI Command
	1	0	1			Rotation	Normal Rotation FI Command
	0	1	1		Level Specification	No Rotation	FI Command for Specification
	1	1	1			Rotation	Specified Bit Rotation FI Command
	0	1	0	Non-FI Command		No Rotation	No Operation
	1	1	0			Rotation	Specified Bit Rotation Command
	0	0	0		No Level Specification	No Rotation	Self FI Mode Rotation Reset
	1	0	0			Rotation	Self FI Mode Rotation Set

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
IMDW	—	SNM	EXCN	0	1	POL	SR	IS/IR

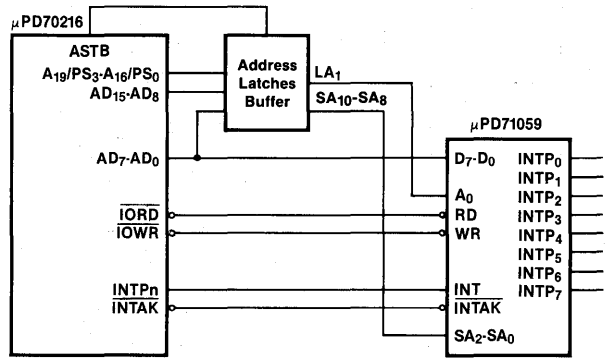
Select Register to Read	In-Service/Request Register Select	Read Register Selection
0	—	No Operation
1	0	IRQ Selection
1	1	IIS Selection

Polling Mode	Polling
0	No Operation
1	Polling Command

Set Nesting Mode	Exceptional Nesting Mode	Nesting Mode 2
0	—	No Operation
1	0	Exceptional Nesting Mode Release
1	1	Exceptional Nesting Mode Set

83-001835C

Figure 31. μPD71059 Cascade Connection



3d

DMA Control Unit

The DMA Control Unit (DMAU) is a high-speed DMA controller compatible with the μPD71071 DMA Controller. The DMAU has four independent DMA channels and performs high-speed data transfers between memory and external peripheral devices at speeds as high as 4 megabytes/second in an 8-MHz system. Figure 32 is the block diagram for the DMAU.

The DMAU has the following features.

- Four independent DMA channels
- Cascade mode for slave μPD71071 DMA controllers
- 20-bit address registers
- 16-bit transfer count registers
- Single, demand, and block transfer modes
- Bus release and bus hold modes
- Autoinitialization
- Address increment/decrement
- Fixed/rotating channel priorities
- TC output at transfer end
- Forced termination of service by $\overline{END}$ input

DMAU Basic Operation

The DMAU operates in either a slave or master mode. In the slave mode, the DMAU samples the four DMARQ input pins every clock. If one or more inputs are active, the corresponding DMA request bits are set and the DMAU sends a bus request to the BAU while continuing to sample the DMA request inputs. After the BAU returns the DMA bus acknowledge signal, the DMAU stops DMA request sampling, selects the DMA channel with the highest priority, and enters the bus master mode to perform the DMA transfer. While in the bus master mode, the DMAU controls the external bus and performs DMA transfers based on the preprogrammed channel information.

Terminal Count

The DMAU ends DMA service when the terminal count condition is generated or when the $\overline{END}$ input is asserted. A terminal count (TC) is produced when the contents of the current count register becomes zero. If autoinitialization is not enabled when DMA service terminates, the mask bit of the channel is set and the DMARQ input of that channel is masked. Otherwise, the current count and address registers are reloaded from the base registers and new DMA transfers are again enabled.

DMA Transfer Type

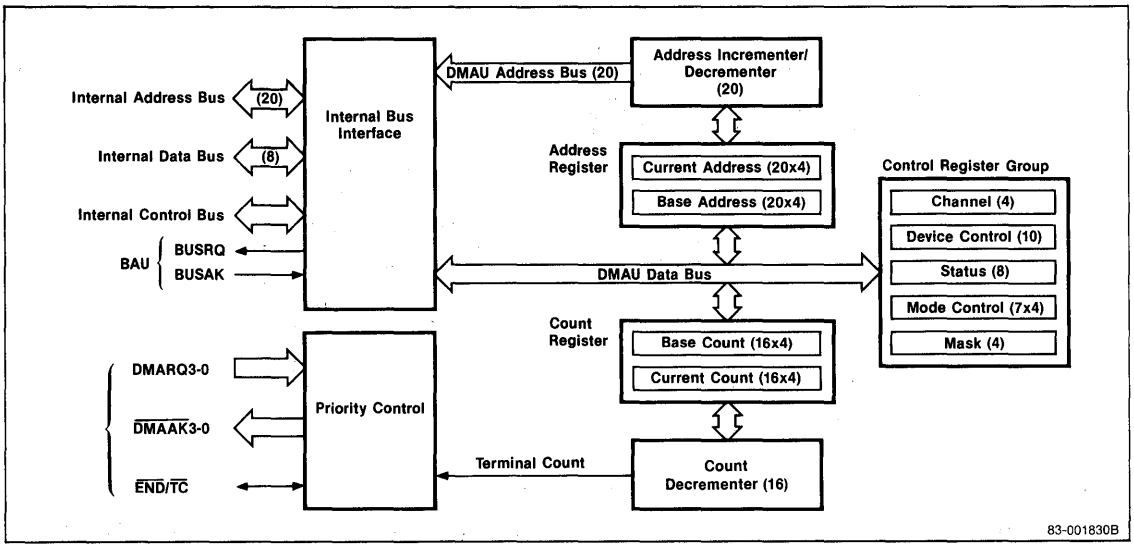
The type of transfer the DMAU performs depends on the following conditions.

- Direction of the transfer (each channel)
- Transfer mode (each channel)
- Bus mode

Transfer Direction

All DMA transfers use memory as a reference point. Therefore, a DMA read operation transfers data from memory to an I/O port. A DMA write operation reads an I/O port and writes the data into memory. During

Figure 32. DMAU Block Diagram



memory-to-I/O transfer, the DMA mode (DMD) register is used to select the transfer directions for each channel and activate the appropriate control signals.

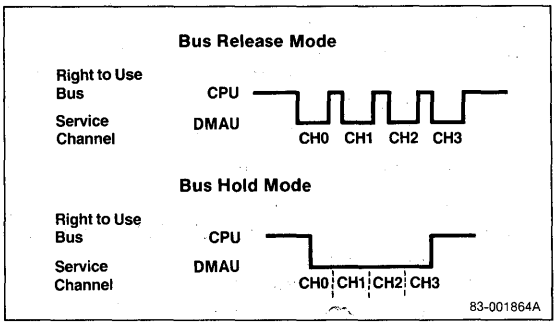
Operation	Transfer Direction	Activated Signals
DMA read	Memory → I/O	$\overline{IOWR}$, $\overline{MRD}$
DMA write	I/O → Memory	$\overline{IORD}$, $\overline{MWR}$
DMA verify		Addresses only; no transfer performed

Bus Mode

The DMA device control (DDC) register selects operation in either the bus release or bus hold mode. The selected bus mode determines the DMAU conditions for return of the bus to the BAU. Figure 33 shows that in bus release mode, only a single channel is serviced after the DMAU obtains the bus. When DMA service ends (termination conditions depend on the transfer mode), the DMAU returns the bus to the BAU regardless of the state of other DMA requests, and the DMAU reenters the slave mode. When the DMAU regains use of the bus, a new DMA operation can begin.

In bus hold mode, several channels can receive contiguous service without releasing the bus. If there is another valid DMA request when a channel's DMA service is finished, the new DMA service can begin immediately after the previous service without returning the bus to the BAU.

Figure 33. Bus Modes



Transfer Modes

The DMD register also selects either single, demand, or block transfer mode for each channel. The conditions for the termination of each transfer characterize each transfer mode. The following table shows the various transfer modes and termination conditions.

Transfer Mode	Termination Conditions
Single	After each byte/word transfer
Demand	$\overline{END}$ input Terminal count Inactive DMARQ DMARQ of a higher priority channel becomes active (bus hold mode)
Block	$\overline{END}$ input Terminal count

The operation of single, demand, and block mode transfers depends on whether the DMAU is in bus release or bus hold mode. Figure 34 shows the operation flow for the six possible transfer and bus mode operations in DMA transfer.

Single Mode Transfer. In bus release mode, when a channel completes transfer of a single byte or word, the DMAU enters the slave mode regardless of the state of DMA request inputs. In this manner, other lower-priority bus masters will be able to access the bus.

In bus hold mode, when a channel completes transfer of a single byte or word, the DMAU terminates the channel's service even if the DMARQ request signal is asserted. The DMAU will then service any other requesting channel. If there are no requests from any other DMA channels, the DMAU releases the bus and enters the slave state.

Demand Mode Transfer. In bus release mode, the currently active channel continues to transfer data as long as the DMA request of that channel is active, even though other DMA channels are issuing higher-priority requests. When the DMA request of the serviced channel becomes inactive, the DMAU releases the bus and enters the slave state.

In bus hold mode, when the active channel completes a single transfer, the DMAU checks the other DMA request lines without ending the current service. If there is a higher-priority DMA request, the DMAU stops the service of the current channel and starts servicing the highest-priority channel requesting service. If there is no higher request than the current one, the DMAU continues to service the currently active channel. Lower-priority DMA requests are honored without releasing the bus after the current channel service is complete.

Block Mode Transfer. In bus release mode, the current channel continues DMA transfers until a terminal count or the external $\overline{END}$ input becomes active. During this time, the DMAU ignores all other DMA requests. After completion of the block transfer, the DMAU releases the bus and enters the slave state, even if DMA requests from other channels are active.

In bus hold mode, the current channel transfers data until an internal or external $\overline{END}$ signal becomes active. When the service is complete, the DMAU checks all DMA requests without releasing the bus. If there is an active request, the DMAU immediately begins servicing the request. The DMAU releases the bus after it honors all DMA requests or a higher-priority bus master requests the bus.

Byte/Word Transfer

The DMD register can specify DMA transfers in byte or word units for each channel. Addresses and count registers are updated as follows during byte/word transfers.

	Byte Transfer	Word Transfer
Address register	±1	±2
Count register	−1	−1

During word transfers, two bytes starting at an even address are handled as a single word. If the starting address is odd, a DMA transfer is started after first decrementing the address by 1. For this reason, always select even addresses. The A_0 and $\overline{UBE}$ outputs control byte and word DMA transfers. The following shows the relationship between the data bus width, A_0 and $\overline{UBE}$ signals, and data bus status.

A_0	$\overline{UBE}$	Data Bus Status
0	1	D_7-D_0 valid
1	0	D_{15}-D_8 valid
0	0	D_{15}-D_0 valid

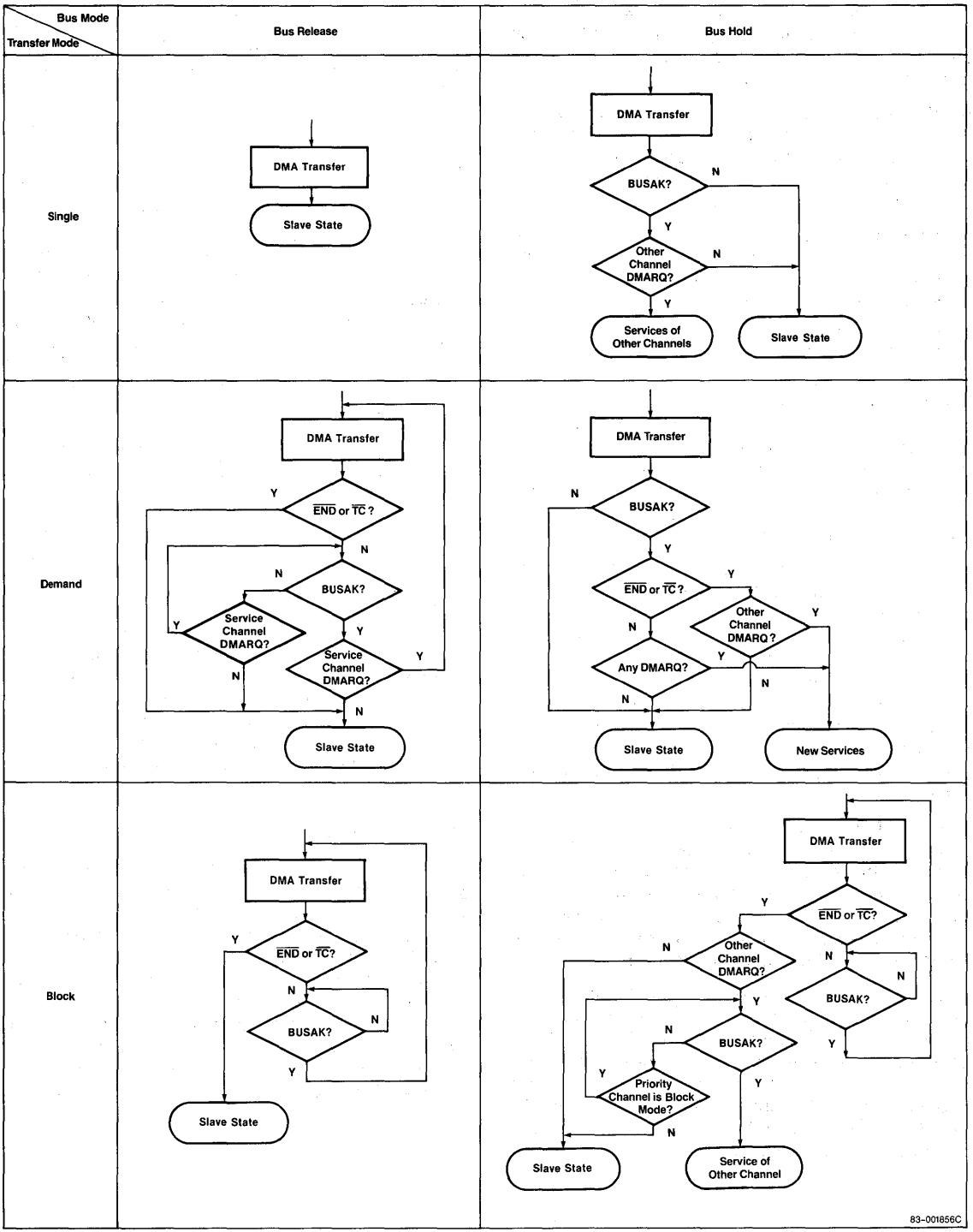
Autoinitialize

When the DMD register selects autoinitialize for a channel, the DMAU automatically reinitializes the address and count registers when $\overline{END}$ is asserted or the terminal count condition is reached. The contents of the base address and base count registers are transferred to the current address and current count registers, and the applicable bit of the mask register remains cleared.

Channel Priority

Each of the four DMAU channels is assigned a priority. When multiple DMA requests from several channels occur simultaneously, the channel with the highest priority will be serviced first. The DDC register selects one of two priority schemes: fixed or rotating (figure 35). In fixed priority, channel 0 is assigned the highest priority and channel 3, the lowest. In rotating priority, priority order is rotated after each service so that the channel last serviced receives the lowest priority. This method prevents the exclusive servicing of higher-priority channels and the lockout of lower-priority DMA channels.

Figure 34. Transfer Modes



Cascade Connection

Slave μPD71071 DMA Controllers can be cascaded to easily expand the system DMA channel capacity to 16 DMA channels. Figure 36 shows an example of cascade connection. During cascade operation, the DMAU acts as a mediator between the BAU and the slave μPD71071s. During DMA cascade mode operation, it is the responsibility of external logic to isolate the cascade bus master from the μPD70216 control outputs. These outputs are listed in a table at the front of this data sheet.

The DMAU always operates in the bus hold mode while a cascade channel is in service, even when the bus release mode is programmed. Other DMA requests are held pending while a slave μPD71071 channel is in service. When the cascaded μPD71071 ends service and moves into the slave state, the DMAU also moves to the slave state and releases the bus. At this time, all bits of the DMAU request register are cleared. The DMAU continues to operate normally with the other noncascaded channels.

Figure 35. Priority Order

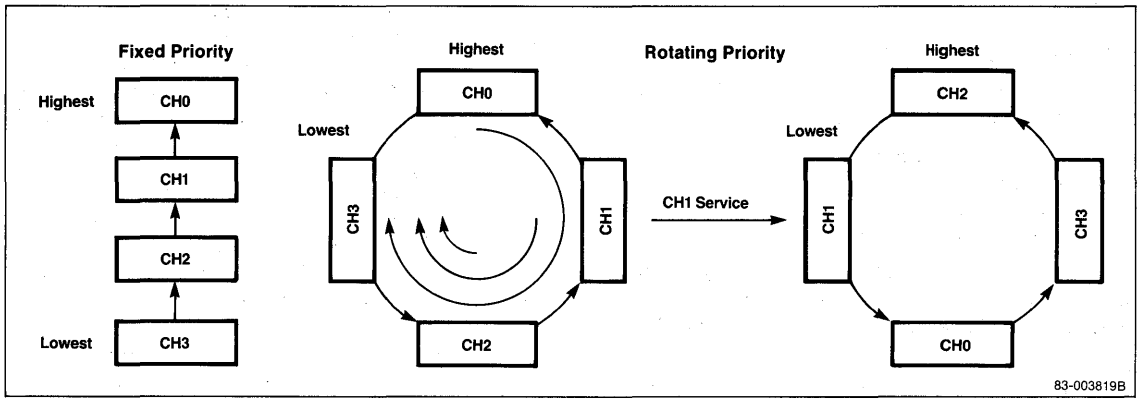
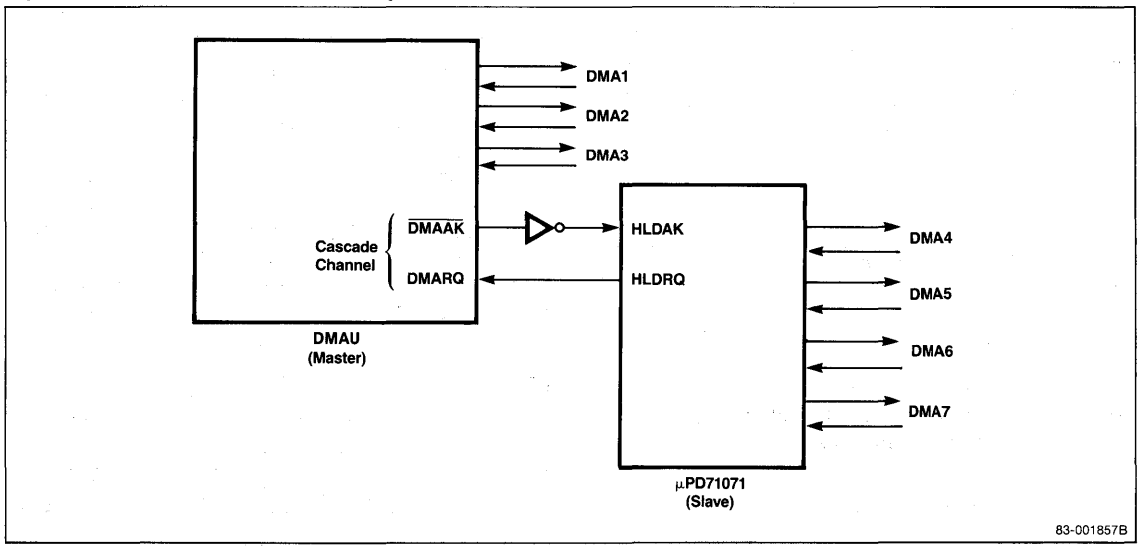


Figure 36. μPD71071 Cascade Example



3d

Bus Waiting Operation

The DMAU will automatically perform a bus waiting operation (figure 37) whenever the RCU refresh request queue fills. When the DMA bus acknowledge goes inactive, the DMAU enters the bus waiting mode and inactivates the DMA bus request signal. Control of the bus is then transferred to the higher-priority RCU by the BAU.

Two clocks later, the DMAU reasserts its internal DMA bus request. The bus waiting mode is continued until the DMA bus acknowledge signal again becomes active and the interrupted DMA service is immediately restarted.

Programming the DMAU

To prepare a channel for DMA transfer, the following characteristics must be programmed.

- Starting address for the transfer
- Transfer count
- DMA operating mode
- Transfer size (byte/word)

The contents of the OPHA and DULA registers determine the base I/O port address of the DMAU. Addresses A_3-A_0 are used to select a particular register as follow:

A_3	A_2	A_1	A_0	Register	Operation
0	0	0	0	DICM	Write
0	0	0	1	DCH	Read/Write
0	0	1	0	DBC/DCC (low)	Read/Write
0	0	1	1	DBC/DCC (high)	Read/Write
0	1	0	0	DBA/DCA (low)	Read/Write
0	1	0	1	DBA/DCA (high)	Read/Write
0	1	1	0	DBA/DCA (upper)	Read/Write
0	1	1	1	Reserved	—
1	0	0	0	DDC (low)	Read/Write
1	0	0	1	DDC (high)	Read/Write
1	0	1	0	DMD	Read/Write
1	0	1	1	DST	Read
1	1	0	0	Reserved	—
1	1	0	1	Reserved	—
1	1	1	0	Reserved	—
1	1	1	1	DMK	Read/Write

Word I/O instructions can be used to read/write the register pairs listed below. All other registers are accessed via byte I/O instructions.

DBC/DCC
DBA/DCA (higher/lower only)
DDC

DMAU Registers

Initialize. The DMA initialize command (DICM) register (figure 38) is used to perform a software reset of the DMAU. The DICM is accessed using the byte OUT instruction.

Channel Register. Writes to the DMA channel (DCH) register (figure 39) select one of the four DMA channels for programming and also the base/current registers. Reads of the DCH register return the currently-selected channel and the register access mode.

Count Registers. When bit 2 of the DCH register is cleared, a write to the DMA count register updates both the DMA base count (DBC) and the DMA current count (DCC) registers with a new count. If bit 2 of the DCH register is set, a write to the DMA count register affects only the DBC register. The DBC register holds the initial count value until a new count is specified. If autoinitialization is enabled, this value is transferred to the DCC register when a terminal count or END condition occurs. For each DMA transfer, the current count register is decremented by one. The format of the DMA count register is shown below. The count value loaded into the DBC/DCC registers is one less than the desired transfer count.

2H, IN/OUT

7							0
C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

3H, IN/OUT

7							0
C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8

Address Register. Use either byte or word I/O instructions with the lower two bytes (4H and 5H) of the DMA address register. However, byte I/O instructions must be used to access the high-order byte (6H) of this register. When bit 2 of the channel register is cleared, a write to the DMA address register updates both the DMA base address (DBA) and the DMA current address (DCA) registers with the new address. If bit 2 of the DCH register is set, a write to the DMA address register affects only the DBA register.

4H, IN/OUT

7							0
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0

5H, IN/OUT

7							0
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

6H, IN/OUT

7							0
—	—	—	—	A_{19}	A_{18}	A_{17}	A_{16}

The DBA register holds the starting address value until a new address is specified. This value is transferred to the DCA register automatically if autoinitialization is selected. For each DMA transfer, the current address register is updated by two during word transfers and by one during byte transfers.

Device Control Register. The DMA device control (DDC) register (figure 40) is used to to program the DMA transfer characteristics common to all DMA channels. It controls the bus mode, write timing, priority logic, and enable/disable of the DMAU.

Status Register. The DMA status (DST) register (figure 41) contains information about the current state of each DMA channel. Software can determine if a termination condition has been reached (TC_3-TC_0) or if a DMA service request is present (RQ_3-RQ_0). The byte IN instruction must be used to read this register.

Figure 37. Bus Waiting Operation

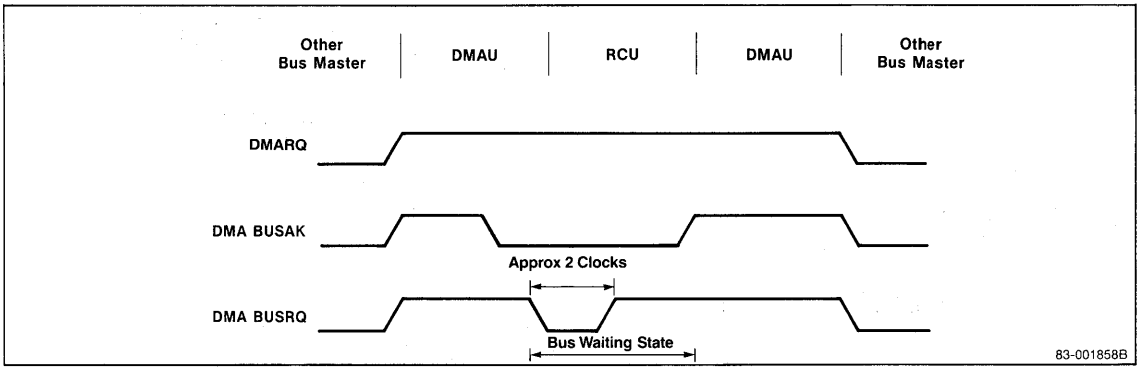
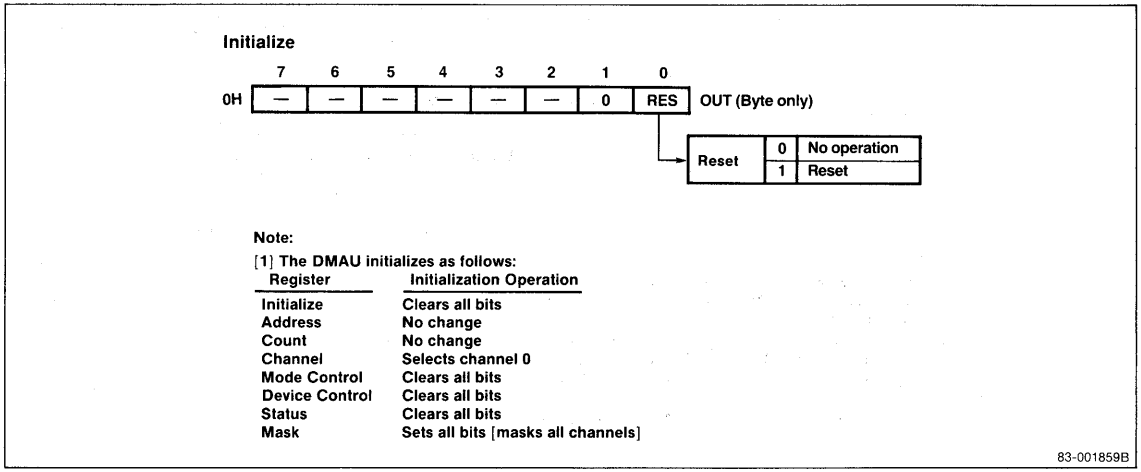


Figure 38. DMA Initialize Command Register

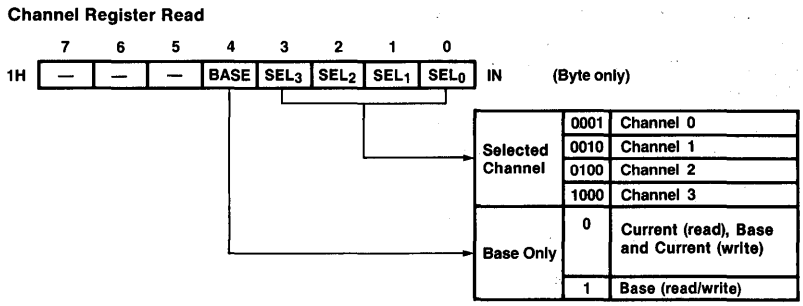


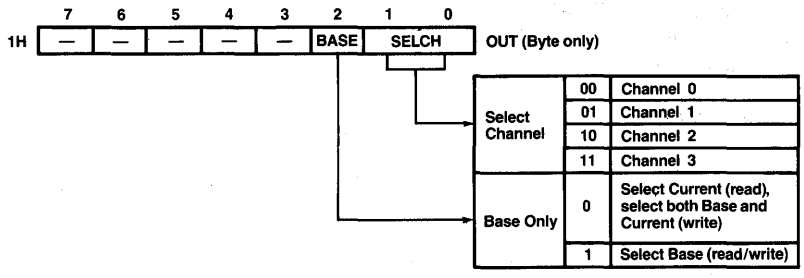
Note:

[1] The DMAU initializes as follows:

Register	Initialization Operation
Initialize	Clears all bits
Address	No change
Count	No change
Channel	Selects channel 0
Mode Control	Clears all bits
Device Control	Clears all bits
Status	Clears all bits
Mask	Sets all bits [masks all channels]

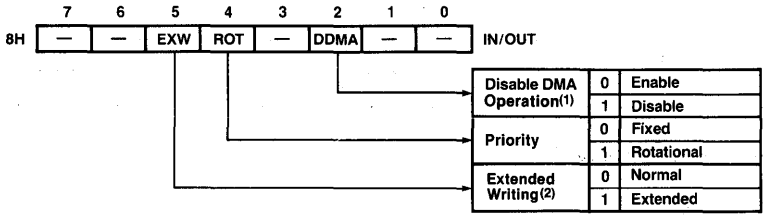
Figure 39. DMA Channel Register

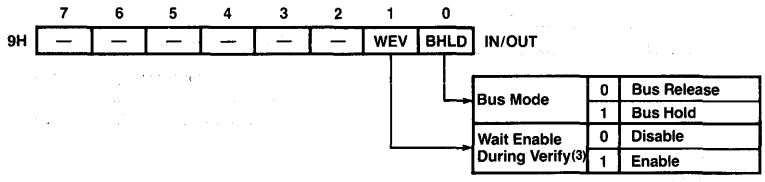




83-003820B

Figure 40. DMA Device Control Register





Note:

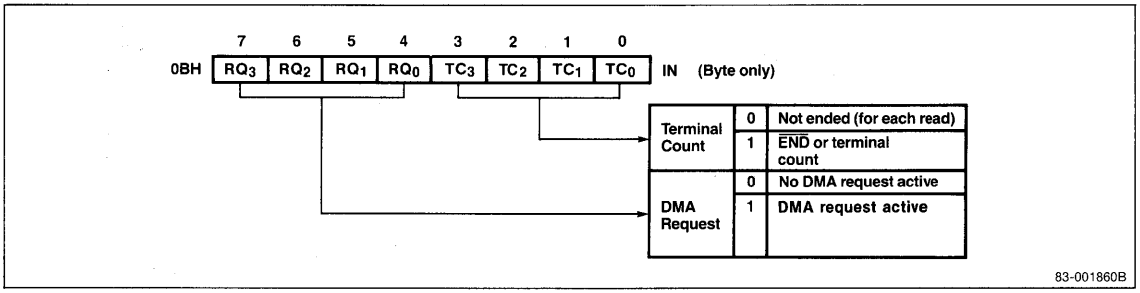
[1] Disables BUSRQ to the BAU to prevent incorrect DMA operation while the DMAU registers are being initialized or modified.

[2] When EXW is 0, the write signal becomes active [normal write] during T3 and TW [see timing waveforms]. When 1, the write signal becomes active during T2, T3, and TW [like the read signal].

[3] Wait states are generated by the READY signal during a verify transfer.

83-001863B

Figure 41. DMA Status Register



Mode Control Register. The DMA mode (DMD) register (figure 42) selects the operating mode for each DMA channel. The DCH register selects which DMD register will be accessed. A byte IN/OUT instruction must be used to access this register.

Mask Register Read/Write. The DMA mask (DMK) register (figure 43) allows software to individually enable and disable DMA channels. The DMK register can only be accessed via byte I/O instructions.

3d

Figure 42. DMA Mode Register

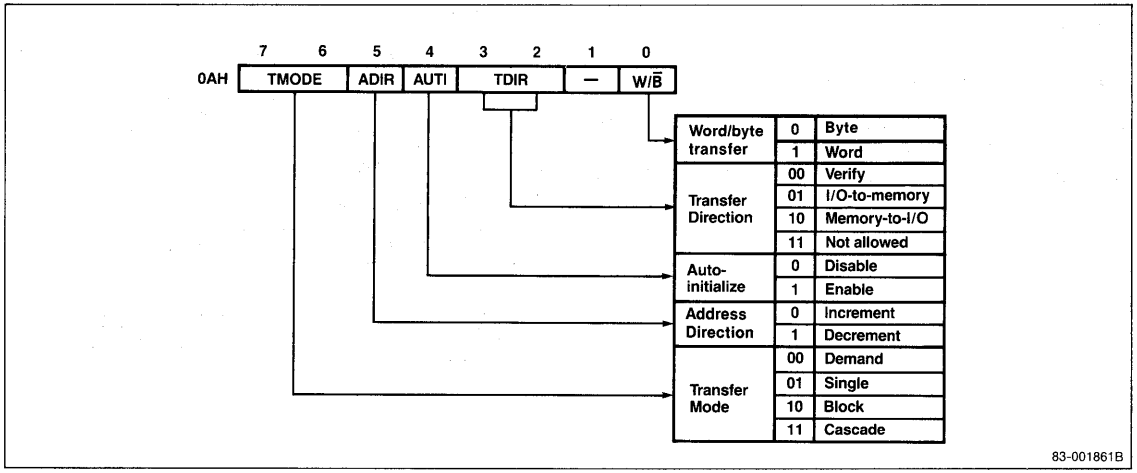
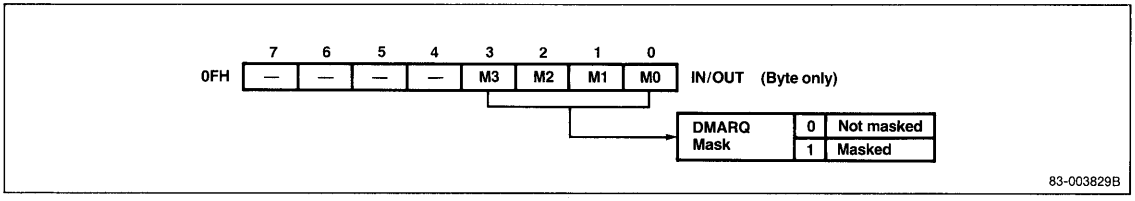


Figure 43. DMA Mask Register



Reset

The falling edge of the $\overline{RESET}$ signal resets the μPD70216. The signal must be held low for at least four clock cycles to be recognized as valid.

CPU Reset State Register	Reset Value
PFP	0000H
PC	0000H
PS	FFFFH
SS	0000H
DS0	0000H
DS1	0000H
PSW	F002H
AW, BW, CW, DW, IX, IY, BP, SP	Undefined
Instruction queue	Cleared

When $\overline{RESET}$ returns to the high level, the CPU will start fetching instructions from physical address FFFF0H.

Internal Peripheral Registers

Internal peripheral devices initialized on reset are listed in the following table. I/O devices not listed are not initialized on reset and must be initialized by software.

	Register	Reset Value
System I/O area	OPCN	----0000
	OPSEL	----0000
	WCY1	11111111
	WCY2	----1111
	WMB	-111-111
	TCKS	---00000
	RFC	x--01000
SCU	SMD	01001011
	SCM	--0000-0
	SIMK	------11
	SST	10000100
	DCH	---00001
	DMD	000000-0
DMAU	DDC (low)	--00-0--
	DDC (high)	------00
	DST	xxxx0000
	DMK	----1111

Symbols: x = unaffected; 0 = cleared; 1 = set; (-) = unused.

Output Pin Status

The following table lists output pin status during reset.

Signal	Status
$\overline{INTAK}$, $\overline{BUFEN}$, $BUF\overline{R}/W$, $\overline{MRD}$, $\overline{MWR}$, $\overline{END}/\overline{TC}$, $\overline{IOWR}$, $\overline{IORD}$, $\overline{REFRQ}$, $\overline{UBE}$, BS_2-BS_0, $\overline{BUSLOCK}$, RESOUT, $\overline{DMAAK3}$-$\overline{DMAAK0}$	High level
QS_1-QS_0, ASTB, HLDAK	Low level
A_{19}-A_{16}/PS_3-PS_0, TOUT2	High or low level
AD_{15}-AD_0	High impedance
CLKOUT	Continues to supply clock

Instruction Set

Symbols

Preceding the instruction set, several tables explain symbols, abbreviations, and codes.

Clocks

In the Clocks column of the instruction set, the numbers cover these operations: instruction decoding, effective address calculation, operand fetch, and instruction execution.

Clock timings assume the instruction has been prefetched and is present in the six-byte instruction queue. Otherwise, add four clocks for each pair of bytes not present.

Word operands require four additional clocks for each transfer to an unaligned (odd-addressed) memory operand. These times are shown on the right-hand side of the slash (/).

For conditional control transfer or branch instructions, the number on the left side of the slash is applicable if the transfer or branch takes place. The number on the right side is applicable if it does not take place.

If a range of numbers is given, the execution time depends on the operands involved.

Symbols

Symbol	Meaning
acc	Accumulator (AW or AL)
disp	Displacement (8 or 16 bits)
dmem	Direct memory address
dst	Destination operand or address
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
far_label	Label within a different program segment
far_proc	Procedure within a different program segment
fp_op	Floating point instruction operation
imm	8- or 16-bit immediate operand

Symbols

Symbol	Meaning
imm3/4	3- or 4-bit immediate bit offset
imm8	8-bit immediate operand
imm16	16-bit immediate operand
mem	Memory field (000 to 111); 8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
memptr16	Word containing the destination address within the current segment
memptr32	Double word containing a destination address in another segment
mod	Mode field (00 to 10)
near_label	Label within the current segment
near_proc	Procedure within the current segment
offset	Immediate offset data (16 bits)
pop_value	Number of bytes to discard from the stack
reg	Register field (000 to 111); 8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
regptr	16-bit register containing a destination address within the current segment
regptr16	Register containing a destination address within the current segment
seg	Immediate segment data (16 bits)
short_label	Label between −128 and +127 bytes from the end of the current instruction
sr	Segment register
src	Source operand or address
temp	Temporary register (8/16/32 bits)
AC	Auxiliary carry flag
AH	Accumulator (high byte)
AL	Accumulator (low byte)
AW	Accumulator (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
BP	BP register
BRK	Break flag
BW	BW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)

Symbols (cont)

Symbol	Meaning
CW	CW register (16 bits)
CY	Carry flag
DH	DW register (high byte)
DIR	Direction flag
DL	DW register (low byte)
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DW	DW register (16 bits)
IE	Interrupt enable flag
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
MD	Mode flag
P	Parity flag
PC	Program counter (16 bits)
PS	Program segment register (16 bits)
PSW	Program status word (16 bits)
R	Register set
S	Sign extend operand field S = 0 No sign extension S = 1 Sign extend immediate byte operand
S	Sign flag
SP	Stack pointer (16 bits)
SS	Stack segment register (16 bits)
V	Overflow flag
W	Word/byte field (0 to 1)
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value
Z	Zero flag

Flag Operations

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to result
u	Undefined
R	Restored to previous state

Memory Addressing Modes

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Register Selection (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Register Selection

sr	Segment Register
00	DS1
01	PS
10	SS
11	DS0

Instruction Set

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer Instructions																									
MOV	reg, reg	1	0	0	0	1	0	1	W	1	1	reg			reg			2	2						
	mem, reg	1	0	0	0	1	0	0	W	mod		reg			mem			7/11	2-4						
	reg, mem	1	0	0	0	1	0	1	W	mod		reg			mem			10/14	2-4						
	mem, imm	1	1	0	0	0	1	1	W	mod		reg			mem			9/13	3-6						
	reg, imm	1	0	1	1	W	reg											4	2-3						
	acc, dmem	1	0	1	0	0	0	0	W									10/14	3						
	dmem, acc	1	0	1	0	0	0	1	W									9/13	3						
	sr, reg16	1	0	0	0	1	1	1	0	1	1	0	sr		reg			2	2						
	sr, mem16	1	0	0	0	1	1	1	0	mod		0	sr		mem			10/14	2-4						
	reg16, sr	1	0	0	0	1	1	0	0	1	1	0	sr		reg			2	2						
	mem16, sr	1	0	0	0	1	1	0	0	mod		0	sr		mem			8/12	2-4						
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod		reg			mem			17/25	2-4						
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod		reg			mem			17/25	2-4						
	AH, PSW	1	0	0	1	1	1	1	1									2	1						
	PSW, AH	1	0	0	1	1	1	1	0									3	1	x	x		x	x	x
LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod		reg			mem			4	2-4						
TRANS	src_table	1	1	0	1	0	1	1	1									9	1						
XCH	reg, reg	1	0	0	0	0	1	1	W	1	1	reg			reg			3	2						
	mem, reg	1	0	0	0	0	1	1	W	mod		reg			mem			13/21	2-4						
	AW, reg16	1	0	0	1	0	reg											3	1						
Repeat Prefixes																									
REPC		0	1	1	0	0	1	0	1									2	1						
REPNC		0	1	1	0	0	1	0	0									2	1						
REP REPE REPZ		1	1	1	1	0	0	1	1									2	1						
REPNE REPNZ		1	1	1	1	0	0	1	0									2	1						
Block Transfer Instructions																									
MOVBK	dst, src	1	0	1	0	0	1	0	W									9 (9) + 8n (W = 0) 9 (9) + 8n (W = 1, even addresses) 9 (17) + 16n (W = 1, odd addresses) 9 (13) + 12n (W = 1, odd/even addresses)	1						
CMPBK	dst, src	1	0	1	0	0	1	1	W									7 (13) + 14n (W = 0) 7 (13) + 14n (W = 1, even addresses) 7 (21) + 22n (W = 1, odd addresses) 7 (17) + 18n (W = 1, odd/even addresses)	1	x	x	x	x	x	x
CMPM	dst	1	0	1	0	1	1	1	W									7 (7) + 10n (W = 0) 7 (7) + 10n (W = 1, even addresses) 7 (11) + 14n (W = 1, odd addresses)	1	x	x	x	x	x	x

3d

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Block Transfer Instructions (cont)																									
LDM	src	1	0	1	0	1	1	0	W									1 7 (7) + 9n (W = 0) 7 (7) + 9n (W = 1, even addresses) 7 (11) + 13n (W = 1, odd addresses)							
STM	dst	1	0	1	0	1	0	1	W									1 5 (5) + 4n (W = 0) 5 (5) + 4n (W = 1, even addresses) 5 (9) + 8n (W = 1, odd addresses)							
																		n = number of returns String instruction execution clocks for a single-instruction execution are in parentheses.							
I/O Instructions																									
IN	acc, imm8	1	1	1	0	0	1	0	W									9/13	2						
	acc, DW	1	1	1	0	1	1	0	W									8/12	1						
OUT	imm8, acc	1	1	1	0	0	1	1	W									8/12	2						
	DW, acc	1	1	1	0	1	1	1	W									8/12	1						
INM	dst, DW	0	1	1	0	1	1	0	W									1 9 (10) + 8n (W = 0) 9 (10) + 8n (W = 1, even addresses) 9 (18) + 16n (W = 1, odd addresses)							
OUTM	DW, src	0	1	1	0	1	1	1	W									1 9 (10) + 8n (W = 0) 9 (10) + 8n (W = 1, even addresses) 9 (18) + 16n (W = 1, odd addresses)							
		n = number of transfers String instruction execution clocks for a single instruction execution are in parentheses. Use the right side of the slash (/) for DMA I/O accesses.																							
BCD Instructions																									
ADJBA		0	0	1	1	0	1	1	1									7	1	x	x	u	u	u	u
ADJ4A		0	0	1	0	0	1	1	1									3	1	x	x	u	x	x	x
ADJBS		0	0	1	1	1	1	1	1									7	1	x	x	u	u	u	u
ADJ4S		0	0	1	0	1	1	1	1									3	1	x	x	u	x	x	x
ADD4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	7 + 19n	2	u	x	u	u	u	x
SUB4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	7 + 19n	2	u	x	u	u	u	x
CMP4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	7 + 19n	2	u	x	u	u	u	x
ROL4	reg8	0 1	0 1	0 0	0 0	1 0	1 reg	1	1	0	0	1	0	1	0	0	0	13	3						
	mem8	0 mod	0	0 0	0 0	1 0	1 mem	1	1	0	0	1	0	1	0	0	0	25	3-5						
ROR4	reg8	0 1	0 1	0 0	0 0	1 0	1 reg	1	1	0	0	1	0	1	0	1	0	17	3						
	mem8	0 mod	0	0 0	0 0	1 0	1 mem	1	1	0	0	1	0	1	0	1	0	29	3-5						
										n = number of BCD digits divided by 2															

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Type Conversion Instructions																									
CVTBD		1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	15	2	u	u	u	x	x	x
CVTDB		1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	7	2	u	u	u	x	x	x
CVTBW		1	0	0	1	1	0	0	0									2	1						
CVTWL		1	0	0	1	1	0	0	1									4/5	1						
Arithmetic Instructions																									
ADD	reg, reg	0	0	0	0	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	0	0	0	0	W	mod			reg			mem		13/21	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	0	0	0	1	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	0	0		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	0	0		mem		15/23	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	0	0	1	0	W									4	2-3	x	x	x	x	x	x
ADDC	reg, reg	0	0	0	1	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	0	0	0	W	mod			reg			mem		13/21	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	0	0	1	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	0		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	0		mem		15/23	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	0	1	0	W									4	2-3	x	x	x	x	x	x
SUB	reg, reg	0	0	1	0	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	1	0	1	0	0	W	mod			reg			mem		13/21	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	0	1	0	1	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	0	1		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	0	1		mem		15/23	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	0	1	1	0	W									4	2-3	x	x	x	x	x	x
SUBC	reg, reg	0	0	0	1	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	1	0	0	W	mod			reg			mem		13/21	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	1	0	1	W	mod			reg			mem		10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	1		reg		4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	1		mem		15/23	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	1	1	0	W									4	2-3	x	x	x	x	x	x
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	0		mem		13/21	2-4	x		x	x	x	x
	reg16	0	1	0	0	0		reg										2	1	x		x	x	x	x
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	1		mem		13/21	2-4	x		x	x	x	x
	reg16	0	1	0	0	1		reg										2	1	x		x	x	x	x
MULU	reg	1	1	1	1	0	1	1	W	1	1	1	0	0		reg		21-30	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	0		mem		26-35	2-4	u	x	x	u	u	u

3d

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Arithmetic Instructions (cont)																									
MUL	reg	1	1	1	1	0	1	1	W	1	1	1	0	1	reg			33-47	2	u	x	x	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	0	1	mem			38-52	2-4	u	x	x	u	u	u
	reg16,reg16,imm8	0	1	1	0	1	0	1	1	1	1	reg			reg			28-34	3	u	x	x	u	u	u
	reg16,mem16,imm8	0	1	1	0	1	0	1	1	mod		reg			mem			33-39	3-5	u	x	x	u	u	u
	reg16,reg16,imm16	0	1	1	0	1	0	0	1	1	1	reg			reg			36-42	4	u	x	x	u	u	u
	reg16,mem16,imm16	0	1	1	0	1	0	0	1	mod		reg			mem			41-47	4-6	u	x	x	u	u	u
DIVU	reg	1	1	1	1	0	1	1	W	1	1	1	1	0	reg			19-25	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	0	mem			24-30	2-4	u	u	u	u	u	u
DIV	reg	1	1	1	1	0	1	1	W	1	1	1	1	1	reg			29-43	2	u	u	u	u	u	u
	mem	1	1	1	1	0	1	1	W	mod		1	1	1	mem			34-48	2-4	u	u	u	u	u	u
Comparison Instructions																									
CMP	reg, reg	0	0	1	1	1	0	1	W	1	1	reg			reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	1	1	1	0	0	W	mod		reg			mem			10/14	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	1	1	0	1	W	mod		reg			mem			10/14	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	1	1	reg			4	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	1	1	mem			12/16	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	1	1	1	0	W									4	2-3	x	x	x	x	x	x
Logical Instructions																									
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg			2	2						
	mem	1	1	1	1	0	1	1	W	mod		0	1	0	mem			13/21	2-4						
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1	reg			2	2	x	x	x	x	x	x
	mem	1	1	1	1	0	1	1	W	mod		0	1	1	mem			13/21	2-4	x	x	x	x	x	x
TEST	reg, reg	1	0	0	0	0	1	0	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	1	0	0	0	0	1	0	W	mod		reg			mem			9/13	2-4	u	0	0	x	x	x
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			10/14	3-6	u	0	0	x	x	x
	acc, imm	1	0	1	0	1	0	0	W									4	2-3	u	0	0	x	x	x
AND	reg, reg	0	0	1	0	0	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	1	0	0	0	0	W	mod		reg			mem			13/21	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	0	0	0	1	W	mod		reg			mem			10/14	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem			15/23	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									4	2-3	u	0	0	x	x	x
OR	reg, reg	0	0	0	0	1	0	1	W	1	1	reg			reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	0	0	1	0	0	W	mod		reg			mem			13/21	2-4	u	0	0	x	x	x
	reg, mem	0	0	0	0	1	0	1	W	mod		reg			mem			10/14	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	0	0	1	reg			4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1	mem			15/23	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	1	0	1	0	W									4	2-3	u	0	0	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Logical Instructions (cont)																									
XOR	reg, reg	0	0	1	1	0	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	1	1	0	0	0	W	mod			reg			mem		13/21	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	1	0	0	1	W	mod			reg			mem		10/14	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	1	0		reg		4	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0		mem		15/23	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	0	0	1	0	W									4	2-3	u	0	0	x	x	x
Bit Manipulation Instructions																									
INS	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	31-117/	3						
		1	1		reg			reg										35-133							
	reg8, imm8	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	31-117/	4						
		1	1	0	0	0		reg										35-133							
EXT	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	26-55/	3						
		1	1		reg			reg										34-59							
	reg8, imm8	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	26-55/	4						
		1	1	0	0	0		reg										34-59							
TEST1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	3	3	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	7/11	3-5	u	0	0	u	u	x
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	4	4	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	8/12	4-6	u	0	0	u	u	x
		mod		0	0	0		mem																	
SET1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	10/18	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	5	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	11/19	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	1									2	1		1				
	DIR	1	1	1	1	1	1	0	1									2	1						
CLR1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	5	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	11/19	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	6	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	12/20	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	0									2	1		0				
	DIR	1	1	1	1	1	1	0	0									2	1						

3d

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Bit Manipulation Instructions (cont)																									
NOT1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	10/18	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	5	4						
		1	1	0	0	0		reg																	
	mem, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	11/19	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	0	1	0	1									2	1		x				
Shift/Rotate Instructions																									
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0		reg		2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0		mem		13/21	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0		reg		7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0		mem		16/24 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0		reg		7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0		mem		16/24 + n	3-5	u	x	u	x	x	x
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1		reg		2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1		mem		13/21	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1		reg		7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1		mem		16/24 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1		reg		7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1		mem		16/24 + n	3-5	u	x	u	x	x	x
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1		reg		2	2	u	x	0	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1		mem		13/21	2-4	u	x	0	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1		reg		7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1		mem		16/24 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1		reg		7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1		mem		16/24 + n	3-5	u	x	u	x	x	x
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	0		mem		13/21	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	0		mem		16/24 + n	2-4		x	u			
	reg, imm	1	1	0	0	0	0	0	W	1	1	0	0	0		reg		7 + n	3		x	u			
	mem, imm	1	1	0	0	0	0	0	W	mod		0	0	0		mem		16/24 + n	3-5		x	u			
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1		reg		2	2		x	u			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	1		mem		13/21	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	1		mem		16/24 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	1		mem		16/24 + n	3-5		x	u			

n = number of shifts

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions (cont)																									
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0		mem		13/21	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0		mem		16/24 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0		mem		16/24 + n	3-5		x	u			
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1		reg		2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1		mem		13/21	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1		reg		7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1		mem		16/24 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1		reg		7 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1		mem		16/24 + n	3-5		x	u			
										n = number of shifts															
Stack Manipulation Instructions																									
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0		mem		15/23	2-4						
	reg16	0	1	0	1	0		reg										6/10	1						
	sr	0	0	0	sr		1	1	0									6/10	1						
	PSW	1	0	0	1	1	1	0	0									6/10	1						
	R	0	1	1	0	0	0	0	0									33/65	1						
	imm	0	1	1	0	1	0	S	0									5-6/9-10	2-3						
POP	mem16	1	0	0	0	1	1	1	1	mod		0	0	0		mem		16/24	2-4						
	reg16	0	1	0	1	1		reg										8/12	1						
	sr	0	0	0	sr		1	1	1									8/12	1						
	PSW	1	0	0	1	1	1	0	1									8/12	1	R	R	R	R	R	R
	R	0	1	1	0	0	0	0	1									43/75	1						
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0	*imm8 = 0 : 12 imm8 > 1 : 17 + 8 (imm8 − 1)								*	4						
DISPOSE		1	1	0	0	1	0	0	1									6/10	1						
Control Transfer Instructions																									
CALL	near_proc	1	1	1	0	1	0	0	0									16/20	3						
	regptr	1	1	1	1	1	1	1	1	1	1	0	1	0		reg		14/18	1						
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0		mem		23/31	2-4						
	far_proc	1	0	0	1	1	0	1	0									21/29	5						
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1		mem		31/47	2-4						
RET		1	1	0	0	0	0	1	1									15/19	1						
	pop_value	1	1	0	0	0	0	1	0									20/24	3						
		1	1	0	0	1	0	1	1									21/29	1						
	pop_value	1	1	0	0	1	0	1	0									24/32	3						

3d

Instruction Set (cont)

Mnemonic	Operands	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Control Transfer Instructions (cont)																									
BR	near_label	1	1	1	0	1	0	0	1									13	3						
	short_label	1	1	1	0	1	0	1	1									12	2						
	reg	1	1	1	1	1	1	1	1	1	1	1	0	0	reg			11	2						
	memptr16	1	1	1	1	1	1	1	1	mod		1	0	0	mem			19/23	2-4						
	far_label	1	1	1	0	1	0	1	0									15	5						
	memptr32	1	1	1	1	1	1	1	1	mod		1	0	1	mem			26/34	2-4						
BV	near_label	0	1	1	1	0	0	0	0									14/4	2						
BNV	near_label	0	1	1	1	0	0	0	1									14/4	2						
BC, BL	near_label	0	1	1	1	0	0	1	0									14/4	2						
BNC, BNL	near_label	0	1	1	1	0	0	1	1									14/4	2						
BE, BZ	near_label	0	1	1	1	0	1	0	0									14/4	2						
BNE, BNZ	near_label	0	1	1	1	0	1	0	1									14/4	2						
BNH	near_label	0	1	1	1	0	1	1	0									14/4	2						
BH	near_label	0	1	1	1	0	1	1	1									14/4	2						
BN	near_label	0	1	1	1	1	0	0	0									14/4	2						
BP	near_label	0	1	1	1	1	0	0	1									14/4	2						
BPE	near_label	0	1	1	1	1	0	1	0									14/4	2						
BPO	near_label	0	1	1	1	1	0	1	1									14/4	2						
BLT	near_label	0	1	1	1	1	1	0	0									14/4	2						
BGE	near_label	0	1	1	1	1	1	0	1									14/4	2						
BLE	near_label	0	1	1	1	1	1	1	0									14/4	2						
BGT	near_label	0	1	1	1	1	1	1	1									14/4	2						
DBNZNE	near_label	1	1	1	0	0	0	0	0									14/5	2						
DBNZE	near_label	1	1	1	0	0	0	0	1									14/5	2						
DBNZ	near_label	1	1	1	0	0	0	1	0									13/5	2						
BCWZ	near_label	1	1	1	0	0	0	1	1									13/5	2						
Interrupt Instructions																									
BRK	3	1	1	0	0	1	1	0	0									38/50	1						
	imm8	1	1	0	0	1	1	0	1									38/50	2						
BRKV	imm8	1	1	0	0	1	1	1	0									40/3	1						
RETI		1	1	0	0	1	1	1	1									27/39	1	R	R	R	R	R	R
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod		reg			mem			17-25/ 52-55	2-4						
BRKEM	imm8	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	38/50	3						
CPU Control Instructions																									
HALT		1	1	1	1	0	1	0	0									2	1						
BUSLOCK		1	1	1	1	0	0	0	0									2	1						
FPO1	fp_op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y	mem			10/14	2-4						
FPO2	fp_op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	2	2						
	fp_op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y	mem			10/14	2-4						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
CPU Control Instructions (cont)																									
POLL		1	0	0	1	1	0	1	1									2 + 5n	1						
						n = number of times POLL pin is sampled.																			
NOP		1	0	0	1	0	0	0	0									3	1						
DI		1	1	1	1	1	0	1	0									2	1						
EI		1	1	1	1	1	0	1	1									2	1						
DS0:, DS1:, PS:, SS: (segment override prefixes)		0	0	1	seg		1	1	0									2	1						
8080 Instruction Set Enhancements																									
RETEM		1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	27/39	2	R	R	R	R	R	R
CALLN	imm8	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	38/58	3						

3d

μPD70136 (V33) 16-Bit Microprocessor: High-Speed, CMOS

Description

The μPD70136 (V33™) is a 16-bit, high-speed CMOS microprocessor that is object and source code compatible with the μPD70116 (V30®). Performance is four times that of the 10-MHz V30 due to a number of architectural features, such as hard-wired data path control and dedicated high-speed logic. The address space is expanded to 16M bytes using an internal address translation table.

The powerful instruction set includes bit processing, bit-field insertion and extraction, and BCD string arithmetic. Using a modified Booth's algorithm, the 16-MHz device can execute a 16-bit multiply in 750 ns.

The μPD70136 has separate 16-bit data and 24-bit address buses. Bus control is synchronous. The nominal bus cycle is two clock periods. Dynamic bus sizing is supported for devices that require an 8-bit data path. This allows the μPD70136 to be used in either 16- or 8-bit systems.

An undefined instruction trap allows instructions that are not part of the V-Series instruction set (such as commands for proprietary MMUs) to be emulated. The μPD72291, a high-speed CMOS floating-point coprocessor capable of 530K floating-point operations per second at 16 MHz, is offered.

Features

- 125-ns minimum instruction execution time at 16 MHz
- Expanded address space
  - 24-bit addressing to 16M bytes
  - LIM 4.0 compatible
- No microcode; better performance with hard-wired data path control
- Dynamic bus sizing for both memory and I/O
- Fully μPD70116 software compatible
- Undefined instruction trap
- High-speed multiplication: 16-bit multiply in 12 clocks (0.75 μs at 16 MHz)
- High-speed division: 16-bit divide in 19 clocks (1.19 μs at 16 MHz)
- μPD72291 floating-point coprocessor executes 530K floating-point operations per second
- BCD string arithmetic instructions
- CMOS with low-power standby mode
- 12.5-MHz or 16-MHz clock
- Single power supply

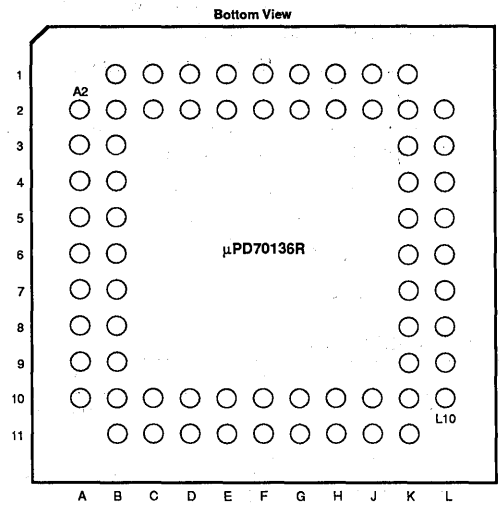
Ordering Information

Part Number	Clock (MHz)	Package
μPD70136R-12	12.5	68-pin ceramic PGA
R-16	16	
L-12	12.5	68-pin PLCC
L-16	16	
GJ-12	12.5	74-pin plastic QFP
GJ-16	16	

V30 is a registered trademark of NEC Corporation
V20, V33, V40, and V50 are trademarks of NEC Corporation

Pin Configurations

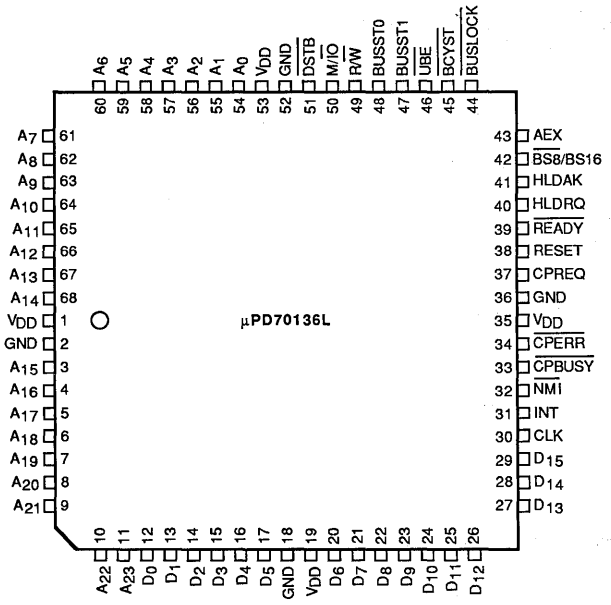
68-Pin Ceramic PGA



Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
A2	AEX	B9	CLK	F10	V_{DD}	K4	A_{12}
A3	HLDAK	B10	D_{14}	F11	GND	K5	A_{14}
A4	$\overline{READY}$	B11	D_{12}	G1	A_0	K6	GND
A5	CPREQ	C1	$\overline{UBE}$	G2	A_1	K7	A_{16}
A6	V_{DD}	C2	BUSST1	G10	D_5	K8	D_{18}
A7	$\overline{CPBUSY}$	C10	D_{11}	G11	D_4	K9	A_{20}
A8	INT	C11	D_{10}	H1	A_2	K10	A_{23}
A9	D_{15}	D1	BUSST0	H2	A_3	K11	A_{22}
A10	D_{13}	D2	$R/\overline{W}$	H10	D_3	L2	A_7
B1	$\overline{BUSLOCK}$	D10	D_9	H11	D_2	L3	A_9
B2	$\overline{BCYST}$	D11	D_8	J1	A_4	L4	A_{11}
B3	$\overline{BS8}$/BS16	E1	$M/\overline{IO}$	J2	A_5	L5	A_{13}
B4	HLDRQ	E2	$\overline{DSTB}$	J10	D_1	L6	V_{DD}
B5	RESET	E10	D_7	J11	D_0	L7	A_{15}
B6	GND	E11	D_6	K1	A_6	L8	A_{17}
B7	$\overline{CPERR}$	F1	GND	K2	A_8	L9	A_{19}
B8	$\overline{NMI}$	F2	V_{DD}	K3	A_{10}	L10	A_{21}

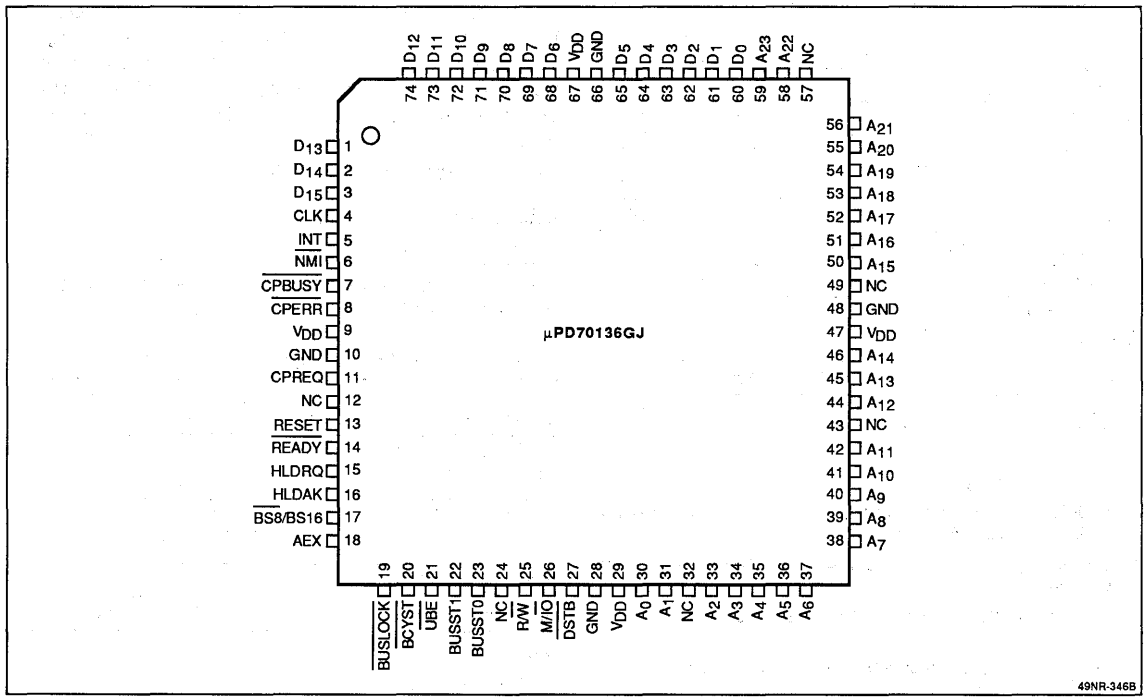
49NR-339B

68-Pin PLCC



49NR-340B

74-Pin Plastic QFP



Pin Identification

Symbol	I/O	Function
A_0-A_{23}	3-state	Address bus
D_0-D_{15}	3-state	Data bus
$\overline{UBE}$	3-state	Upper byte enable
R/$\overline{W}$	3-state	Read/write
M/$\overline{IO}$	3-state	Memory I/O
BUSST0, BUSST1	3-state	Bus status
$\overline{BCYST}$	3-state	Bus cycle start strobe
$\overline{DSTB}$	3-state	Data strobe
$\overline{BUSLOCK}$	Out	Bus lock
$\overline{READY}$	In	Ready
$\overline{BS8}$/BS16	In	Dynamic bus sizing control
AEX	Out	Address expansion flag
HLDRQ	In	Bus hold request

Symbol	I/O	Function
HLDAK	Out	Bus hold acknowledge
INT	In	Maskable interrupt
$\overline{NMI}$	In	Nonmaskable interrupt
$\overline{CPBUSY}$	In	Coprocessor busy
$\overline{CPERR}$	In	Coprocessor error
CPREQ	In	Coprocessor request
RESET	In	Reset
CLK	In	Clock
V_{DD}	—	+5-volt power supply
GND	—	Ground
IC	—	Internal connection; connect to ground
NC	—	No connection

3e

Table 1. Output Pin States

Symbol	States		
	Hold	Standby	Reset
A_0-A_{23} (Note 1)	Hi-z	L	Hi-z
D_{15}-D_0 (Note 1)	Hi-z	(Note 2)	Hi-z
$\overline{UBE}$ (Note 1)	Hi-z	H	Hi-z
R/$\overline{W}$ (Note 1)	Hi-z	L	Hi-z
M/$\overline{IO}$ (Note 1)	Hi-z	L	Hi-z
BUSST0, BUSST1 (Note 1)	Hi-z	H	Hi-z
$\overline{BCYST}$ (Note 1)	Hi-z	(Note 3)	Hi-z
$\overline{DSTB}$ (Note 1)	Hi-z	H	Hi-z
$\overline{BUSLOCK}$	(Note 4)	(Note 4)	H
AEX	(Note 5)	(Note 5)	L
HLDAK	H	L	L

Notes:

(1) Latched internally.

(2) Undefined during first two clock periods of the halt acknowledge cycle; three-state thereafter.

(3) Low during first clock period of the halt acknowledge cycle; high thereafter.

(4) Low if $\overline{BUSLOCK}$ prefix is used for halt instruction; high otherwise.

(5) Low if in extended addressing mode; high otherwise.

PIN FUNCTIONS

CLK (Clock)

CLK is the main clock. All timing is relative to this input. Each bus state is one CLK period wide. Instruction clock counts refer to this CLK input.

A_0-A_{23} (Address Bus)

A_0-A_{23} form the 24-bit physical address bus. It is used to access both the 16M-byte expanded and 1M-byte normal memory spaces and the 64K-byte I/O space. These three-state outputs become valid during T1 of all bus cycles and remain valid until after the bus cycle is completed. During HLDAK and when RESET is active, these outputs are not driven.

D_0-D_{15} (Data Bus)

D_0-D_{15} form the 16-bit data bus, which is used to transfer 16- and 8-bit data between the μPD70136 and the external system. To accommodate 8-bit devices, dynamic bus sizing can be selected so that only the lower 8 bits, D_0-D_7, are used. During CPU read cycles, the value on the data bus is latched by the CPU on the trailing edge of T2 or the last TW state. During CPU write cycles, D_0-D_{15} become valid after the rising edge of T1 and remain valid until after the rising edge of the clock cycle following T2 or the last TW state. During HLDAK and when RESET is asserted, D_0-D_{15} are not driven.

$\overline{UBE}$ (Upper Byte Enable)

$\overline{UBE}$ indicates that the upper 8 bits of the data bus will be used in the current CPU bus cycle. This signal is used in conjunction with A_0 as shown in table 2.

Table 2. Bus Operation vs $\overline{UBE}$ and A_0

Bus Operation	$\overline{UBE}$	A_0	Number of Bus Cycles
Word at even address, $\overline{BS8}$/BS16 = 0	0	0 (Note 1)	2
	1	1 (Note 3)	
Word at odd address	0	1 (Note 1)	2
	1	0 (Note 2)	
Byte at even address	1	0	1
Byte at odd address	0	1	1

Notes:

(1) First bus cycle

(2) Second bus cycle

(3) Second cycle for bus sizing

$\overline{UBE}$ has the same timing as A_0-A_{23} and is not driven during HLDAK or while RESET is asserted.

R/$\overline{W}$ (Read/Write)

R/$\overline{W}$ indicates whether the current bus cycle will be a read or a write. If R/$\overline{W}$ is high, then the cycle will be a read; if low, a write cycle. R/$\overline{W}$ has the same timing as A_0-A_{23} and is not driven during HLDAK or while RESET is asserted.

M/$\overline{IO}$ (Memory/$\overline{IO}$)

M/$\overline{IO}$ indicates whether the current bus cycle will be an access to the memory or I/O space. If M/$\overline{IO}$ is high, access will be to memory; if low, to the I/O space. M/$\overline{IO}$ is used with BUSST0 and BUSST1 to identify the cycle type. M/$\overline{IO}$ has the same timing as A_0-A_{23} and is not driven during HLDAK or while RESET is asserted.

BUSST0-BUSST1 (Bus Cycle Status)

BUSST0 and BUSST1, in conjunction with M/$\overline{IO}$ and R/$\overline{W}$, identify the current cycle type as shown in table 3.

Table 3. Bus Cycle Types

Status				
M/$\overline{IO}$	R/$\overline{W}$	BUSST1	BUSST0	Type of Bus Cycle
0	1	0	0	Interrupt acknowledge
0	1	0	1	I/O read
0	0	0	1	I/O write
0	1	1	0	Coprocessor read
0	0	1	0	Coprocessor write
0	0	1	1	HALT acknowledge
1	1	0	0	Instruction fetch
1	1	0	1	Memory read
1	0	0	1	Memory write
1	1	1	0	CP data read
1	0	1	0	CP data write

Note: All bus status signals change after the start of the T1 state.

The 11 cycle types are described in detail in the bus cycles section. The remaining five combinations of these inputs are reserved for future use.

BUSST0-BUSST1 have the same timing as the address bus, A_0-A_{23}, and are not driven during HLDAK or while RESET is asserted.

$\overline{BCYST}$ (Bus Cycle Start)

$\overline{BCYST}$ indicates the start of a bus cycle. It is asserted low during T1 of every bus cycle, and only for the first clock period of each bus cycle. $\overline{BCYST}$ is not driven during HLDAK or while RESET is asserted.

$\overline{DSTB}$ (Data Strobe)

$\overline{DSTB}$ indicates the status of the data on D_0-D_{15}. When asserted low during a write cycle, the μPD70136 drives the write data on D_0-D_{15}. When the CPU asserts this output during a read cycle, external logic should drive the read data onto D_0-D_{15}.

$\overline{DSTB}$ is asserted following the rising edge (middle) of T1, and stays asserted through T2 and any TW (wait) state that may be inserted. During write cycles, $\overline{DSTB}$ will be deasserted after the rising edge of either T2 or the last wait state. During read cycles, $\overline{DSTB}$ is deasserted after the trailing edge of T2 or the last wait state. $\overline{DSTB}$ is not driven during HLDAK, HALT acknowledge cycles, or while RESET is asserted.

$\overline{BUSLOCK}$ (Bus Lock)

$\overline{BUSLOCK}$ should be used by external logic to exclude any other bus master (e.g., a DMA controller) from using a shared resource that the μPD70136 currently is using. When $\overline{BUSLOCK}$ is asserted high, HLDRQ will be ignored.

$\overline{BUSLOCK}$ is asserted when the $\overline{BUSLOCK}$ prefix is executed or when the μPD70136 is performing a bus operation that must not be interfered with, such as an interrupt acknowledge cycle. $\overline{BUSLOCK}$ has the same timing as the address bus A_0-A_{23} and is driven high during HLDAK and RESET.

$\overline{READY}$ (System Ready)

$\overline{READY}$ is asserted low when the external system is ready for the current bus cycle to terminate. While $\overline{READY}$ is not asserted, the μPD70136 will add TW (wait) states to the current bus cycle. The bus state in which $\overline{READY}$ is sampled low will be the last state of the cycle.

$\overline{READY}$ is used during CPU read cycles to give slow devices time to drive the D_0-D_7 inputs, and during write cycles to give slow devices enough time to finish the write operation.

$\overline{READY}$ is sampled on the rising (middle) edge of T2 and all TW states. $\overline{READY}$ is ignored during the HLDAK state. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

$\overline{BS8}$/BS16 (8-Bit Bus Size/16-Bit Bus Size)

$\overline{BS8}$/BS16 is driven low by external logic when the μPD70136 addresses a device with an 8-bit data path. If the μPD70136 operand is 16 bits wide and $\overline{BS8}$/BS16 is low, then the μPD70136 will perform two 8-bit bus cycles. The current bus cycle will handle the low byte on D_0-D_7, and the next bus cycle will handle the upper byte also on D_0-D_7. This input is ignored during HLDAK, interrupt acknowledge, and coprocessor cycles.

$\overline{BS8}$/BS16 is sampled on the rising (middle) edge of T2 or the last TW state, coincident with $\overline{READY}$. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

AEX (Address Expansion)

AEX is asserted when the expanded addressing mode is enabled. When AEX is high, the memory address space is 16M bytes (24-bit address), and when low, 1M bytes (20-bit address).

HLDRQ (Hold Request)

HLDRQ is asserted high by external logic when an external bus master (e.g., a DMA controller) wants to take over the μPD70136 bus. When HLDRQ is detected high, the μPD70136 will release the bus after the current bus operation is completed. Note that this is not necessarily the current bus cycle. The μPD70136 releases its bus by floating the address, data, and control buses. See table 1.

HLDRQ is sampled on the rising edge of each clock. It will be ignored while $\overline{\text{BUSLOCK}}$ is asserted. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

HLDAK (Hold Acknowledge)

HLDAK is asserted when the μPD70136 enters the hold acknowledge state in response to HLDRQ. Data, address, and control buses are not driven. See table 1.

INT (Interrupt Request)

INT is asserted high by external logic to notify the CPU that an external event has occurred that requires the CPU's attention. After INT has been sampled high, and if the IE (enable interrupts) bit in the PSW is high, interrupt processing will begin after the current instruction is completed.

INT is sampled on the rising edge of each clock. After being asserted high, INT must be kept high until the first INTAK cycle begins. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

$\overline{\text{NMI}}$ (Nonmaskable Interrupt Request)

$\overline{\text{NMI}}$ is asserted by external logic to notify the CPU that an external event has occurred which requires the CPU's immediate attention. When $\overline{\text{NMI}}$ is sampled low, interrupt processing will begin immediately after the current instruction is completed. A trap will be taken through vector 2. The state of the IE bit in the PSW has no effect on $\overline{\text{NMI}}$ acceptance.

$\overline{\text{NMI}}$ is sampled on the rising edge of each clock. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

Once $\overline{\text{NMI}}$ is samped low, an internal flag is set, so that a one-clock pulse meeting setup and hold times will be recognized. The flag is cleared when the $\overline{\text{NMI}}$ is accepted and can be set again immediately.

$\overline{\text{CPBUSY}}$ (Coprocessor Busy)

$\overline{\text{CPBUSY}}$ is asserted low by a coprocessor (such as μPD72291) when it is busy with an internal operation. The μPD70136 uses this pin to check the status of the coprocessor.

$\overline{\text{CPBUSY}}$ is sampled on the falling edge of each clock. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

$\overline{\text{CPERR}}$ (Coprocessor Error)

$\overline{\text{CPERR}}$ is asserted low by a coprocessor to notify the μPD70136 of an error.

$\overline{\text{CPERR}}$ is sampled on the falling edge of each clock. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

CPREQ (Coprocessor Request)

CPREQ is asserted high by a coprocessor to request the μPD70136 to run a memory operation for the coprocessor.

CPREQ is sampled on the falling edge of each clock. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

RESET (Reset)

RESET is asserted high when external logic needs to initialize the μPD70136; for instance, after power-up. When RESET is asserted for at least 6 clock periods, the μPD70136 will abort any current bus cycles and initialize the registers as shown in table 4.

Table 4. Register Initialization by Reset

Register	Offset Value
PFP	0000H
PC	0000H
PS	FFFFH
SS	0000H
DS0	0000H
DS1	0000H

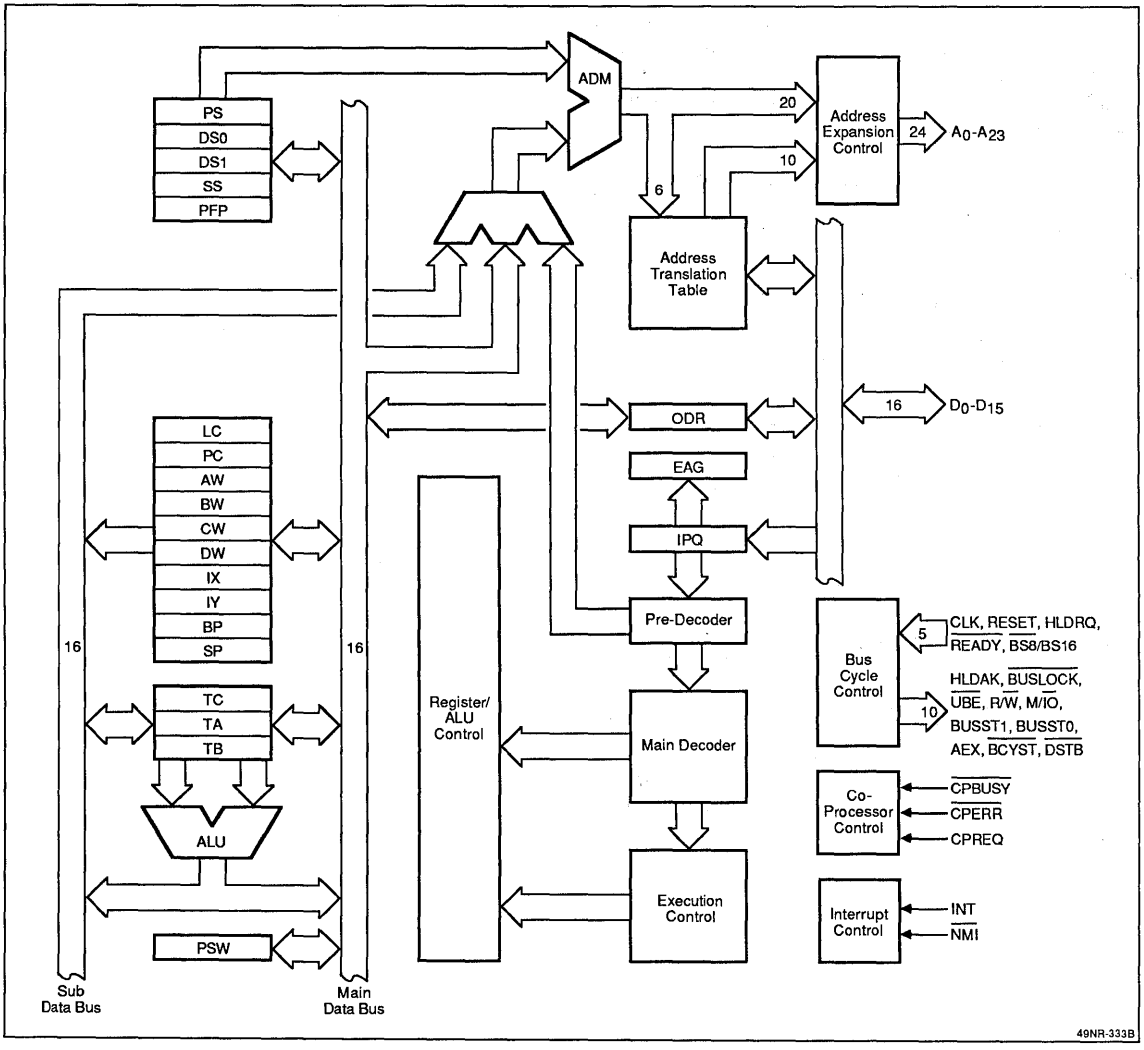
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSW	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0

Prefetch Queue	Cleared
Address Mode	Normal Address Mode
Other Registers	Undefined (if power has just been turned on) Unchanged (if power on, but RESET is asserted)

Refer to table 1 for the state of the μPD70136 outputs during reset. When RESET is deasserted low, the μPD70136 will begin fetching from address 0FFFF0H. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

3e

μPD70136 Block Diagram



ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = +25°C

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 V to V_{DD} + 0.3 V
CLK input voltage, V_K	−0.5 V to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 V to V_{DD} + 0.3 V
Operating temperature, T_{OPT}	−10 to +70°C
Storage temperature, T_{STG}	−65 to +150°C

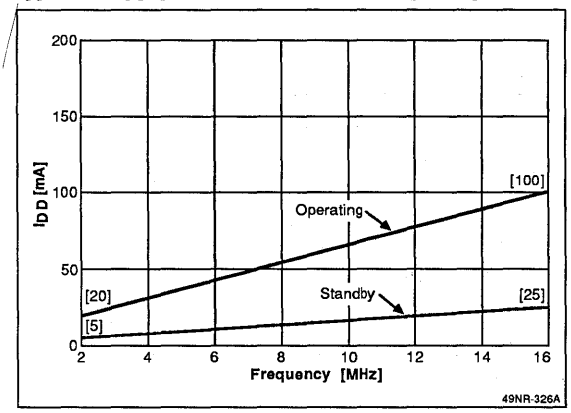
Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Capacitance

T_A = +25°C, V_{DD} = 0 V

Parameter	Symbol	Max	Unit	Conditions
Input capacitance	C_I	15	pF	fc = 1 MHz; unmeasured pins returned to 0 V.
I/O capacitance	C_{IO}	15	pF	

Typical Supply Current vs Clock Frequency



DC Characteristics

T_A = −10 to +70°C, V_{DD} = +5 V ±10%

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Input voltage high	V_{IH}	2.2		V_{DD} + 0.3	V	
Input voltage low	V_{IL}	−0.5		0.8	V	
CLK input voltage high	V_{KH}	0.8 V_{DD}		V_{DD} + 0.5	V	
CLK input voltage low	V_{KL}	−0.5		0.6	V	
Output voltage high	V_{OH}	0.7 V_{DD}			V	I_{OH} −400 mA
Output voltage low	V_{OL}			0.45	V	I_{OL} = 2.5 mA
Input leakage current high	I_{LIH}			−10	μA	V_I = V_{DD}
Input leakage current low	I_{LIL}			10	μA	V_I = 0 V
Output leakage current high	I_{LOH}			10	μA	V_O = V_{DD}
Output leakage current low	I_{LOL}			−10	μA	V_O = 0 V
Supply current	I_{DD} (see graph)	16 MHz	100	150	mA	Normal operation
			25	35	mA	Standby mode
		12.5 MHz	75	110	mA	Normal operation
			20	30	mA	Standby mode
*Stop mode current is not a function of CPU clock frequency		*		200	μA	Stop mode

3e

AC Characteristics

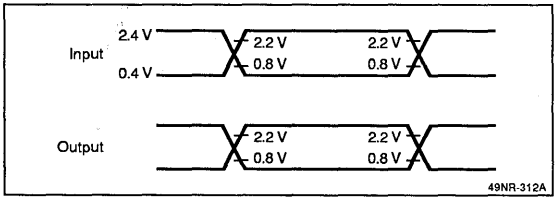
T_A = −10 to +70°C; V_{DD} = 5 V ±10%; C_L = 100 pF

Parameter	Symbol	12.5-MHz Limits Min	12.5-MHz Limits Max	16 MHz-Limits Min	16 MHz-Limits Max	Unit
Clock period	t_{CYK}	80	500	62.5	500	ns
Clock high-level width	t_{KKH}	35		25		ns
Clock low-level width	t_{KKL}	35		25		ns
Clock rise time (1.7 V → 3.0 V)	t_{KR}		5		5	ns
Clock fall time (3.0 V → 1.7 V)	t_{KF}		5		5	ns
Reset delay time (V_{DD} valid)	t_{DVRST}	1		1		μs
Reset setup time (CLK ↓)	t_{SRSTK}	10		10		ns
Reset hold time (CLK ↓)	t_{HKRST}	15		15		ns
Reset high time	t_{WRSTH}	6		6		t_{CYK}
CLK ↓ → $\overline{BCYST}$ delay	t_{DKBC}	5	40	5	40	ns
$\overline{BCYST}$ low-level width	t_{BCBCL}	t_{CYK} − 10		t_{CYK} − 10		ns
$\overline{BCYST}$ high-level width	t_{BCBCH}	t_{CYK} (n + 1) −10		t_{CYK} (n + 1) −10		ns
CLK ↓ → address delay	t_{DKA}	5	40	5	40	ns
CLK ↓ → status delay	t_{DKST}	5	40	5	40	ns
$\overline{READY}$ setup time (CLK ↑)	t_{SRYK}	7		7		ns
$\overline{READY}$ hold time (CLK ↑)	t_{HKRY}	15		15		ns
CLK ↑ → data output delay	t_{DKD}	5	40	5	40	ns
Floating delay	t_{FK}	0	50	0	50	ns
CLK ↑ → $\overline{DSTB}$ delay	t_{DKDS}	5	40	5	40	ns
Address/status output → $\overline{DSTB}$ ↓ delay time	t_{DADSL}	t_{KKL} + t_{KR} − 15		t_{KKL} + t_{KR} − 15		ns
$\overline{DSTB}$ ↑ address/status hold time	t_{HDSHA}	t_{KKL} + t_{KR} − 15		t_{KKL} + t_{KR} − 15		ns
$\overline{DSTB}$ low-level width	t_{DSDSL}	t_{CYK} (n+1) − 10		t_{CYK} (n+1) − 10		ns
$\overline{DSTB}$ high-level width	t_{DSDSH}	t_{KKL} + t_{KR} − 10		t_{KKL}+ t_{KR} − 10		ns
CLK ↓ → $\overline{DSTB}$ ↑ delay for read cycle	t_{DKDSRD}	5	40	5	40	ns
Address/status output → data delay time	t_{DAD}	t_{KKL} + t_{KR} − 15		t_{KKL} + t_{KR} − 15		ns
$\overline{DSTB}$ ↑ → data output delay time	t_{DDSHD}	t_{KKL} + t_{KR} − 15		t_{KKL} + t_{KR} − 15		ns
Data setup time (CLK ↓)	t_{SDK}	7		7		ns
Data hold time (CLK ↓)	t_{HKD}	10		10		ns
Data hold time ($\overline{DSTB}$ ↑)	t_{HDSD}	0		0		ns
Data hold time (R/W ↓)	t_{HRWD}	0		0		ns
$\overline{BS8}$/BS16 setup time	t_{SBSK}	7		7		ns
$\overline{BS8}$/BS16 hold time	t_{HKBS}	15		15		ns
HLDRQ setup time (CLK ↑)	t_{SHQK}	7		7		ns
HLDRQ hold time (CLK ↑)	t_{HKHQ}	15		15		ns
CLK ↑ → HLDAK delay time	t_{DKHA}	5	40	5	40	ns
Output float → HLDAK delay	t_{DFHA}	t_{KKL} + t_{KR} − 15		t_{KKL} + t_{KR} − 15		ns
$\overline{NMI}$, INT, $\overline{CPBUSY}$ setup time (CLK ↓)	t_{SIK}	10		10		ns
$\overline{NMI}$, INT, $\overline{CPBUSY}$ setup time (CLK ↓)	t_{HKI}	10		10		ns

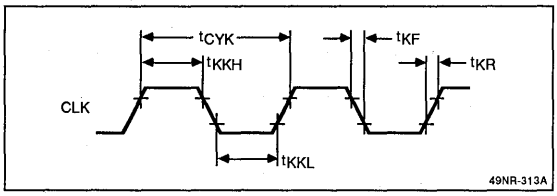
Note: 'n' means number of wait cycles to be inserted into bus cycle

Timing Waveforms

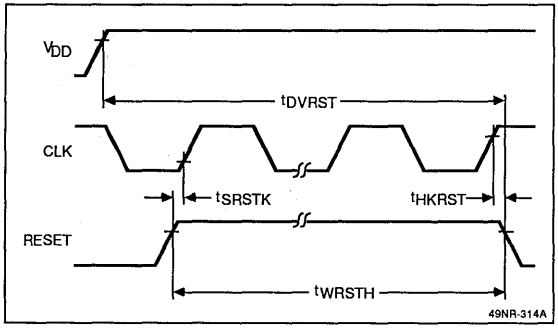
Input/Output Voltage Reference Levels



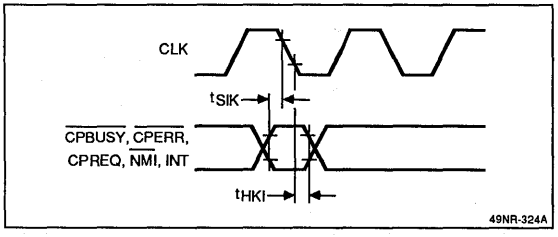
Clock Input



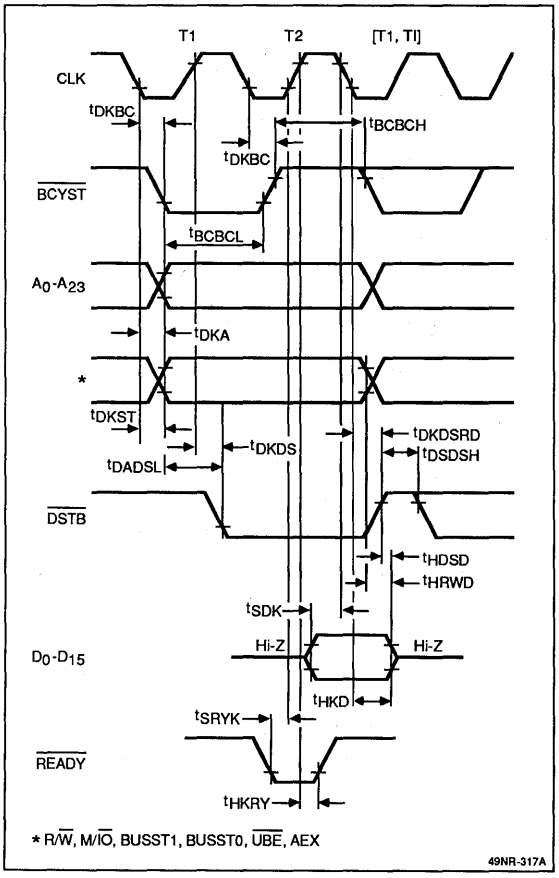
Reset



Input Setup/Hold Time

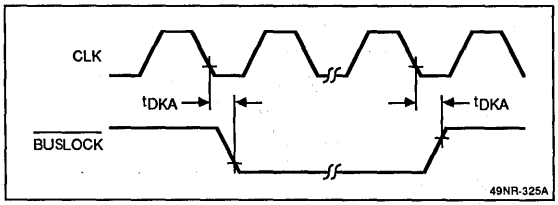


Basic Read Cycle (0 WAIT)

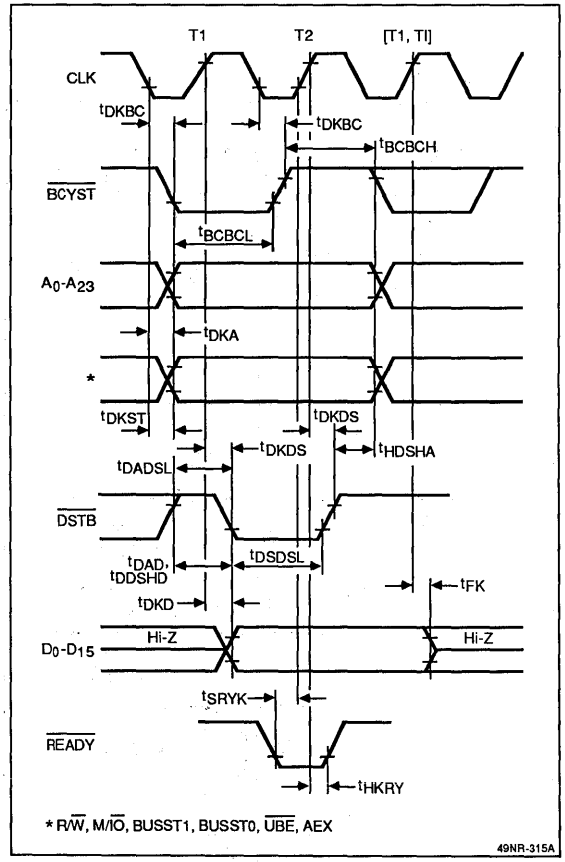


3e

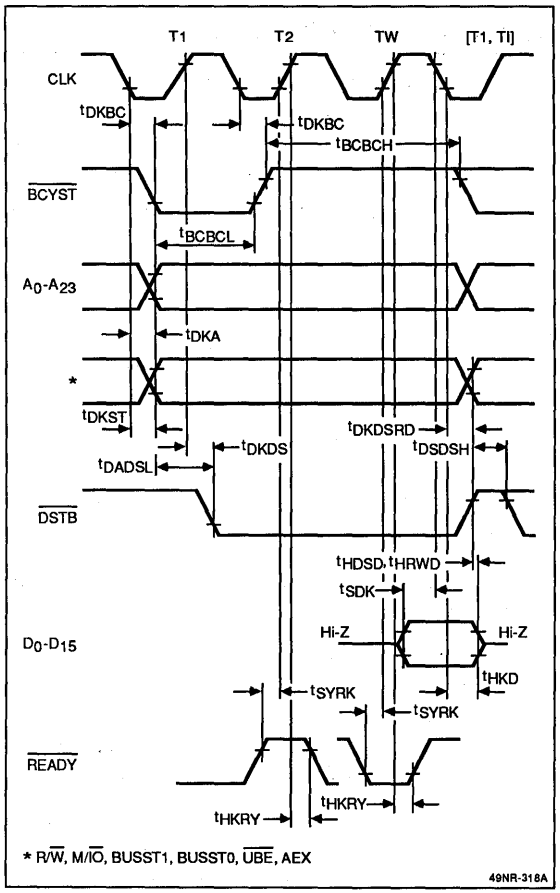
Bus Lock



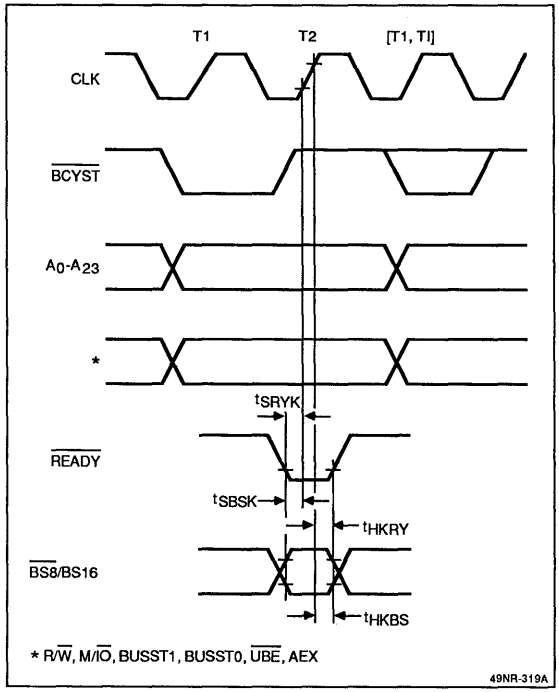
Basic Write Cycle (0 WAIT)



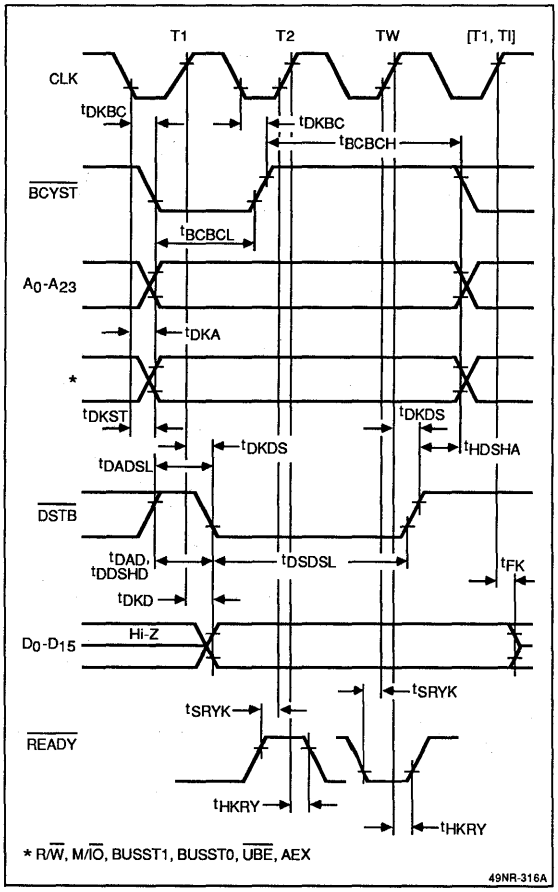
Basic Read Cycle (1 WAIT)



Basic Write Cycle (1 WAIT)

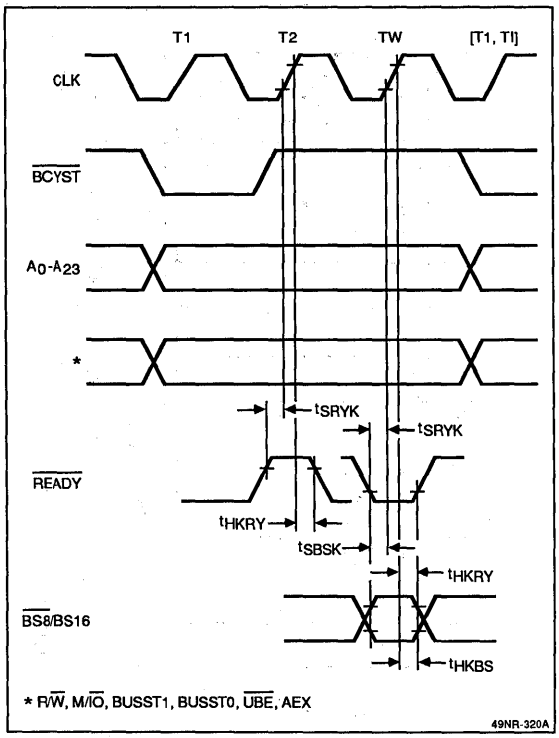


Bus Sizing Cycle (0 WAIT)

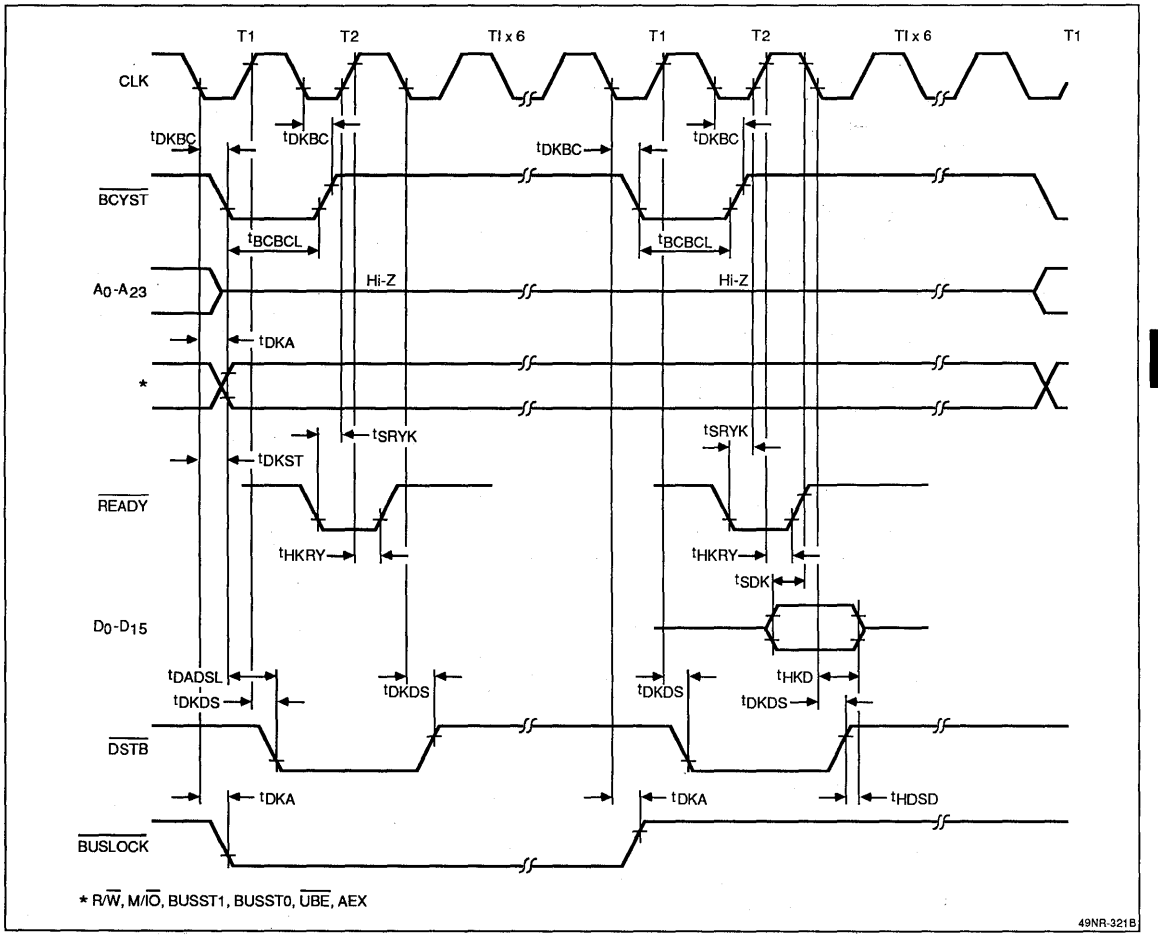


3e

Bus Sizing Cycle (1 WAIT)

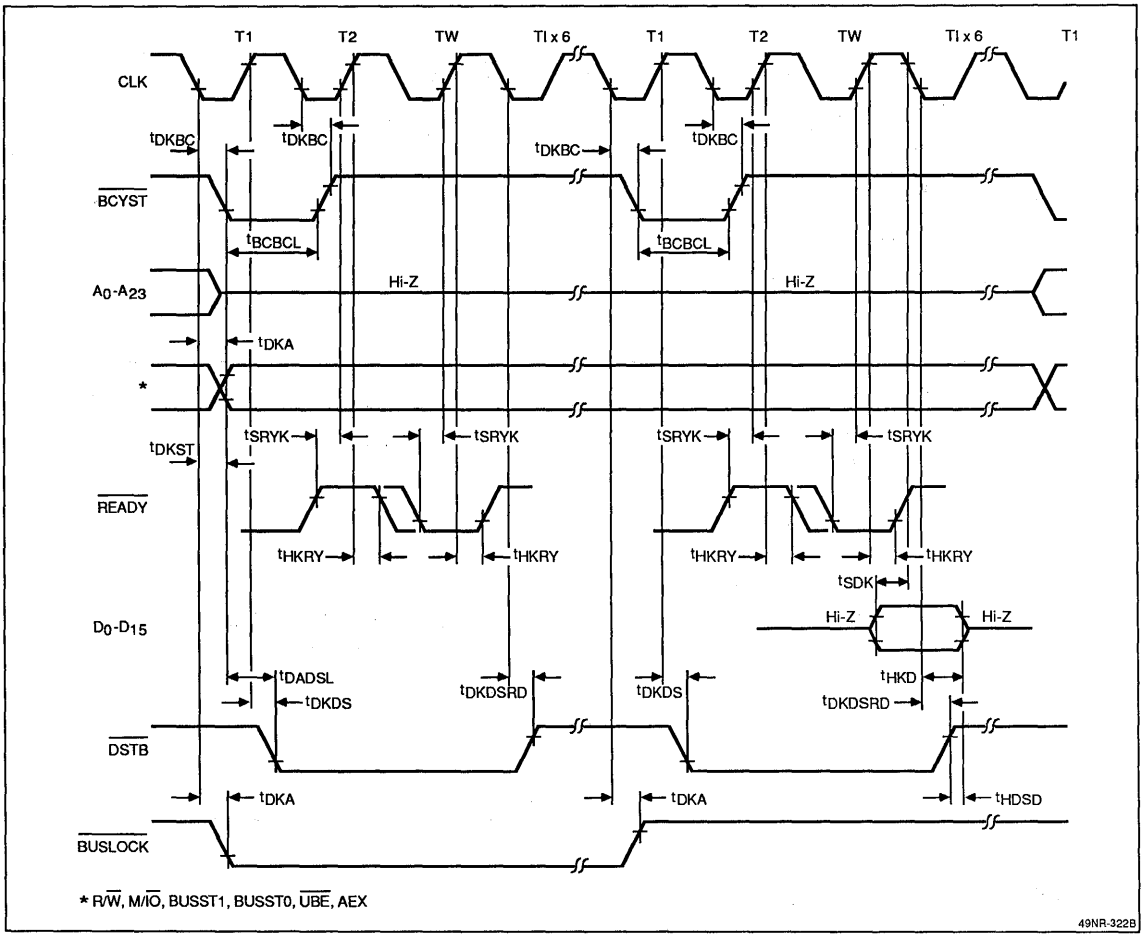


Interrupt Acknowledge (0 WAIT)

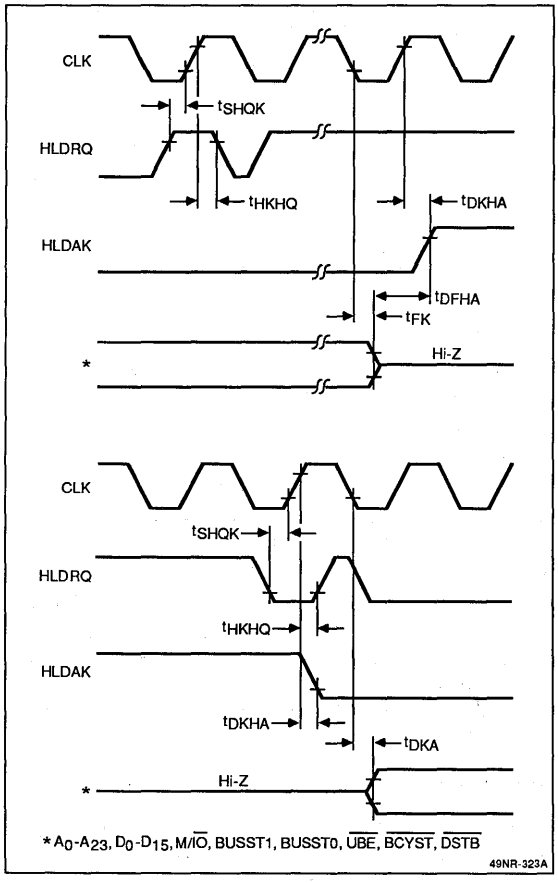


3e

Interrupt Acknowledge (1 WAIT)



Bus Hold


CLK
tSHQK
HLDRQ
tHKHQ
tDKHA
HLDAK
tDFHA
tFK
*
Hi-Z
CLK
tSHQK
HLDRQ
tHKHQ
HLDAK
tDKHA
tDKA
*
Hi-Z
* A0-A23, D0-D15, M/IO, BUSST1, BUSST0, UBE, BCYST, DSTB
49NR-323A

FUNCTIONAL DESCRIPTION

Architecture

A unique hardware architecture feature of the μPD70136 is that there is no microcode. Instruction decode and data path control are implemented using logic and small independent state machines. This greatly enhances the speed with which instructions can be executed, in the same way that programs written in assembly language can be faster and more efficient than high-level language code. The μPD70136 is four times faster than the μPD70116.

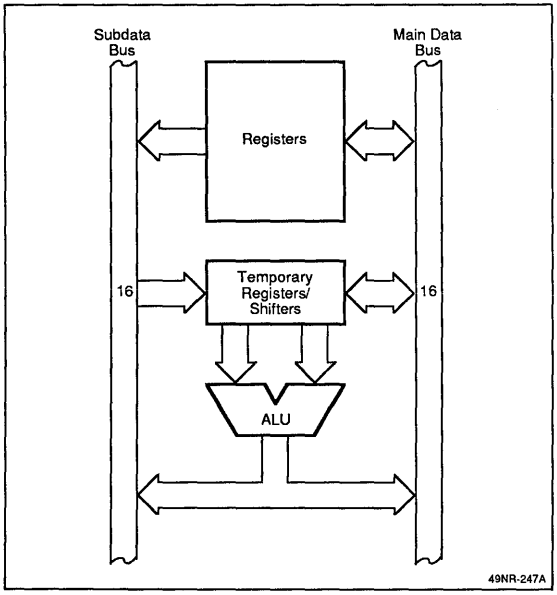
The μPD70136 hardware comprises the execution unit, a bus interface, and the address generator. See the μPD70136 Block Diagram.

Execution Unit

The μPD70136 execution unit consists of a register file, an ALU, instruction decode, and execution control logic.

Besides the hardware control logic, the most significant feature of the execution unit is a dual-bus internal data path. See figure 1. The ALU and many registers are dual-ported, with a data bus on each port. This allows two operands to be transferred in one clock cycle instead of two. Performance is improved by as much as 30 percent using the dual data bus concept.

Figure 1. Dual Data Buses



Register File. There are 12 registers in the internal RAM. Four are temporary registers used in the execution of certain instructions (LC, TA, TB, TC). The other eight are general-purpose registers (AW, BW, CW, DW, IX, IY, BP, SP) and either contain operand data or point to operand data in memory.

The temporary registers speed up instruction execution by serving as scratch pad registers during complex operations.

The loop counter (LC) is used during primitive block transfer operations. It contains the count value. It is also used as a shift counter for multiple-bit shift and rotate instructions.

Temporary registers TA, TB, and TC are the inputs to the ALU. They are used as temporary registers/shifters during multiply, divide, shift/rotate, and BCD rotate operations.

ALU. The ALU consists of a complete adder and logical operation unit. It executes arithmetic (ADD, SUB, MUL, DIV, INC, DEC, NEG, etc.) and logical (TEST, AND, OR, XOR, NOT, SET1, CLR1, etc.) instructions.

Data Path Control Logic. This logic comprises the main instruction decoder and the execution control blocks. Its purpose is to decide what operations must be done and to schedule them. It transfers operands as needed and controls the ALU. State machines are used to implement long, complex instructions.

Bus Interface

The bus interface comprises bus control logic, an operand data register (ODR), an 8-byte instruction prefetch queue (IPQ), and an effective address generator.

The bus control state machines implement the μPD70136 bus interface. To allow the bus machine to run independent of the execution unit, an operand data register is used. During a CPU write cycle, the write data is placed in the ODR and the execution of the next instruction proceeds without waiting. The bus interface finishes the write cycle when the bus is available. During a read cycle, if the operand requires two bus cycles (as in a read from an odd address), the full 16-bit value is assembled in the ODR, one byte at a time.

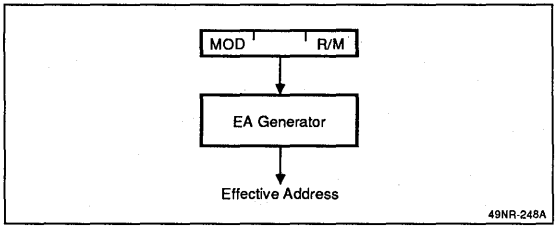
Instruction Prefetching. The μPD70136 is a pipelined machine. To keep the pipeline running efficiently, it should be kept full of instructions in various stages of execution. Instructions are fetched before they are needed and placed in the IPQ. Data in the IPQ is broken out by the pre-decoder logic to determine what addressing modes will be used and what CPU resources will be required to execute the prefetched instruction. To keep

the 8-byte IPQ full, the bus control logic will schedule an instruction prefetch cycle whenever there are at least 2 unused bytes in the IPQ.

The IPQ is cleared whenever a control transfer instruction (any branch, call, return, or break) is executed. This is done because a different instruction stream will be used following a control transfer, and the IPQ will then contain instruction data that will never be used. When this happens, the μPD70136's pipeline will empty out, hampering performance. To maximize performance, the number of control transfers should be minimized.

Effective Address Generator. The EAG logic computes a 16-bit effective address for each operand, which is an offset into one of the four segments. This effective address is passed on to the address modifier adder. The EAG decodes the first byte(s) of each instruction to determine the addressing mode and initiates any bus cycles required to fetch pointers/offsets from memory. Effective addresses are calculated in a maximum of 1 clock period, compared to 5 to 12 clocks for a microprogrammed machine. See figure 2.

Figure 2. Effective Address Generator



Address Generator

The address generator comprises the address register file, the address modifier (ADM), the address translation table, and the needed control logic.

The registers in the address register file are PS, SS, DS0, DS1, PC, and PFP. The ADM is a dedicated adder that adds one of the segment registers to the effective address to produce the 20-bit normal address. The ADM also increments the prefetch pointer. If expanded addressing is enabled, the address translation table is accessed to map the 20-bit address into a 24-bit expanded address.

For instruction stream data, addresses are generated differently. The prefetch pointer contains a 16-bit offset into the PS segment that points to the next instruction word to be prefetched. The program counter contains an offset into the PS segment that points to the instruction that is currently being executed. As part of all control transfers, the PFP is set to the same value as the PC.

ADDRESSING MECHANISM

The μPD70136 is completely compatible with the μPD70108/116 in its addressing modes, and in the way that addresses are computed. It offers a method of expanding the memory address space to 16M bytes.

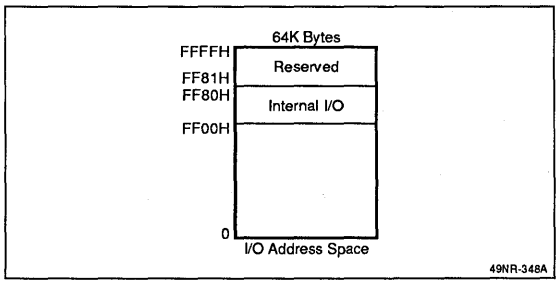
The I/O space is 64K bytes (16-bit address). The normal memory address space is 1M byte (20-bit address), and the expanded address space is 16M bytes (24-bit address). Expanded addressing is enabled or disabled using the BRKXA and RETXA instructions.

3e

The memory space is accessed when an instruction uses a memory addressing mode. Memory addresses are calculated as described below. The I/O space can only be accessed through the IN, OUT, INM, and OUTM instruction.

Certain areas of the μPD70136 address (physical for normal mode, and logical for expanded addressing mode) spaces are reserved. These areas are shown in figures 3 and 4. Memory addresses 0-3FCH are used for the Interrupt Vector table located in the Interrupt Operation section. Memory addresses FFFF0H-FFFFFH must contain a branch to boot code; PC, PFP, and PS are initialized at RESET to point to this area. I/O addresses FF00H-FF80H are reserved for the address translation registers.

Figure 3. I/O Address Space

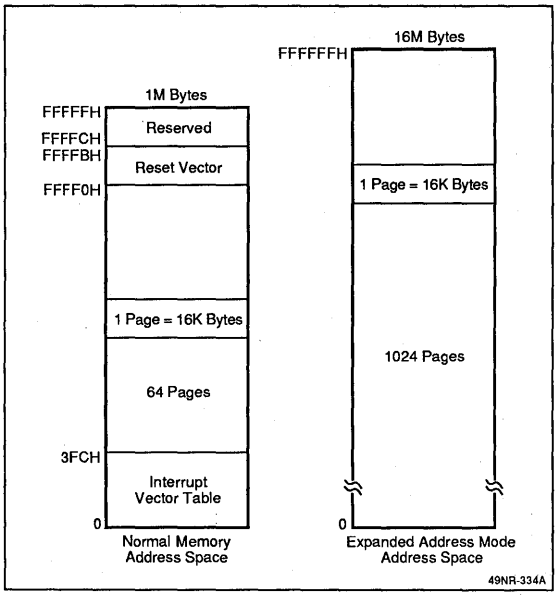


I/O Addresses

I/O addresses are always taken from 8-bit immediate data or the DW register. DW is always used as a direct pointer into the I/O address space. If I/O operations require the use of other more complex addressing modes, the I/O devices must be placed in the memory address space (using memory-mapped I/O techniques). For memory-mapped I/O devices, there are no restrictions on instruction or addressing mode usage. However,

the μPD70136 will not automatically insert 6 clock cycles after memory-mapped I/O operations; external logic must provide the needed I/O device recovery time.

Figure 4. Address Space

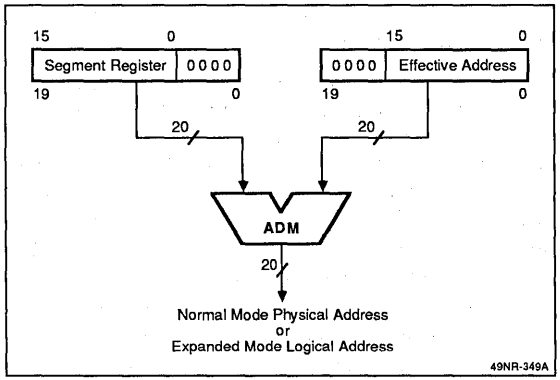


Normal Memory Addresses

The μPD70136 is a 16-bit device with 16-bit registers. To allow a memory address space larger than 64K bytes, memory segmentation is used. The 1M-byte memory address space is divided into 64K byte segments. Up to four segments can be in use at any given time. The base addresses of the four active segments (program segment, stack segment, data segment 0, and data segment 1) are contained in four 16-bit segment registers (PS, SS, DS0, and DS1, respectively). The 16-bit value in each register is the upper 16 bits of the 20-bit memory address. Thus, segments always start on 16-byte boundaries.

As described above, the μPD70136 hardware generates a 16-bit effective address for each memory operation. This effective address is an offset into one of the four active segments. The actual 20-bit memory address is computed by adding the EA to the segment register value expanded with zeros to 20 bits. Figure 5 shows this process.

Figure 5. 20-Bit Address



If normal addressing mode is enabled, then this 20-bit result is presented on the address bus during the bus cycle. If expanded addressing mode is enabled, this address is used as a logical address.

Expanded Addresses

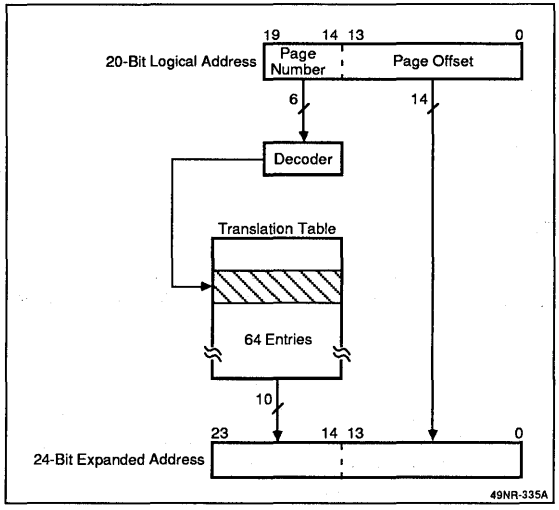
In the expanded addressing mode, the memory space is divided into 1024 pages. Each page is 16K bytes. Each page of the normal 20-bit address space is mapped to a page in the expanded address space using a 64-entry address translation table. The table is made up of 64 page registers that reside in the I/O space (figure 4).

The programming model of this mode is the same as for the normal mode. Address expansion is a layer added to the normal mode that is transparent to executing code. The program still sees a 20-bit contiguous logical memory address space, but the hardware sees 64 pages mapped into a set of 1024 physical pages.

The I/O space is not affected by the expanded addressing mode.

The address translation mechanism is shown in figure 6. The upper 6 bits of the logical 20-bit address select one of the entries in the address translation table, which supplies a 10-bit value. This value is substituted for the original 6 bits in the normal address to create a 24-bit expanded address.

Figure 6. Address Translation Mechanism



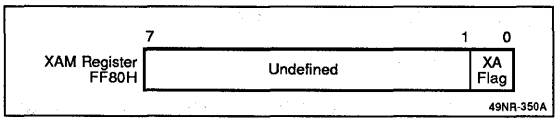
Address Expansion Registers

These are the page and XAM registers. Word IN and OUT instructions are used to access these registers. The table below shows page register usage and I/O addresses. The page registers contain the 10-bit physical page base address.

A_{19}-A_{14} Logical Address	PGR Selected	PGR I/O Address
0	PGR1	FF00
1	PGR2	FF02
2	PGR3	FF04
3	PGR4	FF06
:	:	:
63	PGR64	FF7E

The XAM register (figure 7) is a read-only status flag that indicates whether expanded addressing is enabled. Unused data bits in the XAM register are undefined. Expanded addressing must be disabled before accessing any of the page registers. I/O operations to these internal registers are not passed to the bus interface and will not be seen by external logic.

Figure 7. XAM Register

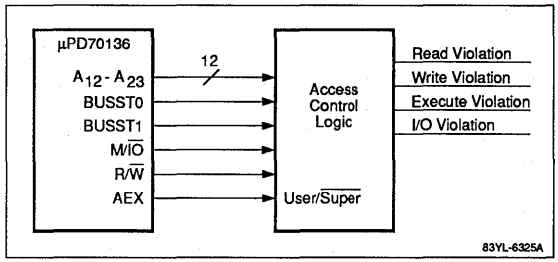


Memory Protection Mechanism

The μPD70136 expanded mode provides a harware memory protection mechanism (figure 8) that does not sacrifice software compatibility with existing μPD8088/8086 or V20/V30/V40/V50 programs. In expanded mode, the XAM and PGR registers cannot be accessed. This provides simple two-level protection.

A supervisory system task running in normal mode can set up restricted memory spaces for less privileged user tasks by programming the PGR registers and then starting up the user task in expanded mode. The user task will not be able to change its memory map to access privileged memory areas. External access control logic can monitor the AEX output to determine at which privilege level the CPU is currently running (AEX = 0 is supervisor mode, AEX = 1 is user mode) and permit or prevent each bus cycle, thereby providing additional memory and I/O protection. This scheme provides the basic hardware protection needed for most operating systems without forgoing full software compatibility.

Figure 8. Expanded Mode Protection Mechanism



OPERAND ADDRESSING MODES

For operand addressing, the μPD70136 offers 9 modes:

- Register
- Immediate
- Direct
- Register indirect
- Indexed
- Based
- Based index
- Bit
- Autoincrement/autodecrement

Register

The operand is in a μPD70136 register pointed to by the instruction.

Immediate

The operand is in the instruction stream following the opcode of the instruction. This data will have been prefetched. Immediate data uses the μPD70136 pipeline efficiently.

3e

Direct

Immediate data in the instruction stream points directly to the operand. This data can be a 16-bit effective address or a 4-bit bit field length.

Register Indirect

A 16-bit register (IX, IY, or BW) contains a 16-bit effective address.

Indexed

One or two bytes of immediate data are treated as a signed displacement that is added to the contents of a 16-bit index register (IX or IY) to obtain a 16-bit effective address.

Based

One or two bytes of immediate data are treated as a signed displacement that is added to the contents of a 16-bit base register (BP or BW) to form a 16-bit effective address.

Based Indexed

One or two bytes of immediate data are treated as a signed displacement that is added to two 16-bit registers (one of BP or BW with one of IX or IY) to form the effective address. This mode is useful for array addressing.

Bit

Used with NOT1, SET1, CLR1, or TEST1. A 4-bit immediate data value is used to select a bit in a 16-bit operand. For 8-bit operands, only 3 bits are used.

Autoincrement/Autodecrement

Some interactive operations (such as MOVBK or INS) will automatically increment or decrement index registers after each iteration. Specifically, IX is used in addressing a source pointer, and/or IY is used in addressing a destination pointer. After the operation, both will be incremented or decremented (according to the PSW DIR control flag) to point to the next operand in the array.

INSTRUCTION ADDRESSING MODES

These modes are basically the same as the operand addressing modes, but the PC is always used as the register. The seven modes are used in control transfer instructions:

- Direct
- Relative
- Register
- Register indirect
- Indexed
- Based
- Based indexed

Direct

Four bytes of immediate data are taken as an absolute address and loaded directly into the PS and PC (and PFP).

Relative

One or two bytes of immediate data are a signed displacement that is added to the contents of the PC and then placed in the PC (and PFP). This mode is used to create position-independent code.

Register

The register selected by the instruction (AW, BW, etc.) contains an effective address, which is loaded into the PC (and PFP).

Register Indirect

An index register (IX, IY, or BW) points to a memory location that contains an effective address (short pointer) or a segment register value and an effective address (far pointer). This effective address is read from memory and loaded into the PS and/or PC (and PFP).

Indexed

One or two bytes of immediate data are a signed displacement that is added to the contents of a 16-bit index register (IX or IY) to form an effective address. This address is used to fetch another effective address, which is loaded into the PC (and PFP).

Based

One or two bytes of immediate data are a signed displacement that is added to the contents of a 16-bit base register (BP or BW) to form an effective address. This address is used to fetch another effective address from memory, which is then loaded into the PC (and PFP).

Based Indexed

One or two bytes of immediate data are a signed displacement that is added to the contents of two 16-bit register (one of BP or BW with one of IX or IY) to form an

effective address. This address is used to fetch another effective address from memory, which is then loaded into the PC (and PFP).

REGISTER CONFIGURATION

Program Counter (PC)

The PC is a 16-bit register that contains the effective address of the instruction that is currently being executed. The PC is incremented each time the instruction decoder accepts a new instruction from the prefetch queue. The PC is loaded with a new value during execution of a branch, call, return, or break instruction and during interrupt processing.

Segment Registers (PS, SS, DS0, DS1)

There are four segment registers, each of which contains the upper 16 bits of the base address of a 64K logical segment. Since logical segments reside on 16-byte boundaries, the lower 4 bits of the base address are always 0. Normal 20-bit memory addresses are formed by adding the 16-bit effective address to the base address of one of the segments. When performing this operation, certain types of effective addresses will be paired with specific segment registers.

Segment Register	Default Offset
PS (program segment)	PFP
SS (stack segment)	SP, effective address
DS0 (data segment 0)	IX, effective address
DS1 (data segment 1)	IY

Program instructions will always be fetched from the program segment. Whenever the IY index register is used to address an operand, the DS1 segment register will be used. DS0 is usually used with IX. Stack operations using the SP will always use the stack segment. For other effective addresses, the preceding table shows the default segment used, but another segment may be selected by using a segment override prefix instruction.

General-Purpose Registers (AW, BW, CW, DW)

The four 16-bit general-purpose registers can be accessed as 16-bit or 8-bit quantities. When the AW, BW, CW, or DW designations are used, the register will be 16 bits. When AL, AH, BL, BH, CL, CH, DL, or DH is used, the register will be 8 bits. AL will be the low byte of AW, and AH the high byte, and so on.

Some operations require the use of specific registers:

AW	Word multiplication/division, word I/O, data conversion
AL	Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
AH	Byte multiplication/division
BW	Translation
CW	Shift instructions, rotation instructions, BCD operations
DW	Word multiplication/division, indirect addressing I/O

Pointer (SP, BP) and Index (IX, IY) Registers

These registers are used as base pointers and index registers when based, indexed, or based-indexed addressing modes are used. They may also be used as general-purpose registers in data transfer, arithmetic, and logical instructions. They can only be accessed as 16-bit registers.

Some operations use these registers in specific ways:

SP	Stack operations
IX	Source pointer for block transfer, bit field, and BCD string operations
IY	Destination pointer for block transfer, bit field, and BCD string operations

Program Status Word (PSW)

The program status word reflects the status of the CPU with six status flags, and affects the operation of the CPU through three control flags:

Status Flags		Control Flags	
V	Overflow	DIR	Direction
S	Sign	IE	Interrupt enable
Z	Zero	BRK	Break
AC	Auxiliary carry		
P	Parity		
CY	Carry		

The PSW cannot be accessed directly as a 16-bit register. Specific instructions are used to set/reset the control flags. When the PSW is pushed on the stack (as during interrupt processing), the following image is used.

1	1	1	1	V	DIR	IE	BRK
15							8

S	Z	0	AC	0	P	1	CY
7							0

BUS OPERATION OVERVIEW

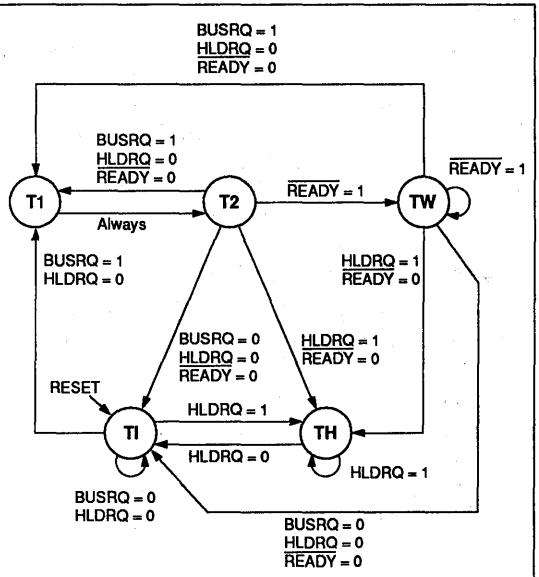
The μPD70136 uses a synchronous bus interface. The CLK input supplies the μPD70136 with a clock. All

3e

μPD70136 bus timings and instruction execution clock counts are specified relative to this clock. Bus cycles start on the falling edge of CLK. Each bus cycle is two clock periods long, and may be extended by adding wait states.

Figure 9 is the state diagram of the bus control state machine. The first state of every bus cycle is T1, followed immediately by T2. $\overline{\text{READY}}$ is sampled on the rising (middle) edge of T2. If $\overline{\text{READY}}$ is not asserted, then the next bus state will be a TW wait state. TWs will be inserted until $\overline{\text{READY}}$ is sampled low, after which the bus cycle will finish. The dynamic bus sizing input, $\overline{\text{BS8}}$/BS16, is sampled at the same time as $\overline{\text{READY}}$.

Figure 9. Bus State Diagram



Address and bus status are output after the leading edge of T1 and are maintained until after the cycle is completed. A strobe, $\overline{\text{BCYST}}$, is asserted during T1 to indicate the beginning of a bus cycle. $\overline{\text{BCYST}}$ is output following the leading edge of T1 and is deasserted after the leading edge of T2.

Write data is driven on D_0-D_{15} following the rising (middle) edge of T1 and is maintained until after the rising edge of the cycle following T2 or the last TW. The read data is sampled on the trailing edge of T2 or the last TW state. A strobe ($\overline{\text{DSTB}}$) gives the status of the μPD70136 data bus. $\overline{\text{DSTB}}$ is asserted after the rising edge (middle) of T1. $\overline{\text{DSTB}}$ is deasserted after the rising edge of T2 or the last TW for a write cycle, and after the trailing edge of T2 or the last TW for a read cycle.

I/O cycles are identical to memory cycles except for the encoding of the bus status lines. However, six idle states are inserted after every I/O bus cycle to provide a recovery time for slow I/O devices.

Dynamic Bus Sizing

The μPD70136 supports dynamic bus sizing. On a cycle by cycle basis, the width of the data bus can be changed from 16 to 8 bits. This simplifies the connection of 8-bit I/O devices that may have internal registers at consecutive byte addresses. Other 16-bit CPUs require two ROMs for startup code, but the μPD70136 dynamic bus sizing makes it possible to use a single 8-bit wide ROM.

External logic requests an 8-bit data path by driving $\overline{\text{BS8}}$/BS16 low in time for the μPD70136 to sample it on the rising edge of T2 (or TW). The μPD70136 will perform an additional bus cycle if needed to finish the operation in byte-wide pieces.

Referring to tables 5 and 6, if the bus operation is 8 bits wide, no further bus cycles will occur. For a read cycle, the data will be sampled on D_0-D_7. For a write cycle to an even address, data will be driven on D_0-D_7. On all byte writes to an odd address, the μPD70136 will put the byte data on both upper and lower data buses; the write data will be on D_0-D_7 as well as D_8-D_{15}.

If the bus operation is 16-bit, then two bus cycles will be required. The first one, in which $\overline{\text{BS8}}$/BS16 is sampled low, will handle the low byte. The second cycle will take the form of a byte read or write using D_0-D_7.

Bus Cycle Types

The 11 different types of μPD70136 bus cycles are classified as read, write, and acknowledge cycles.

Read Cycles

The read cycles are memory, I/O, coprocessor, data reads, and instruction fetch. All have the general timing described above. Coprocessor reads are used to access the internal registers of a coprocessor. Coprocessor data reads are used to transfer data from memory to an internal coprocessor register.

Table 5. Write Cycle Bus Sizing

					16-Bit Bus ($\overline{BS8}$/BS16 = 1)		8-Bit Bus ($\overline{BS8}$/BS16 = 0)	
Type	Address	A_0	$\overline{UBE}$	Cycle	D_{15}-D_8	D_7-D_0	D_{15}-D_8	D_7-D_0
Byte	Even	0	1	1st	Invalid	Byte	Invalid	Byte
	Odd	1	0	1st	Byte	Byte	Byte	Byte
Word	Even	0	1	1st	Upper	Lower	Upper	Lower
		1	0	2nd	Not needed for 16-bit bus		Upper	Upper
	Odd	1	0	1st	Lower	Lower	Lower	Lower
		0	1	2nd	Upper	Upper	Invalid	Upper

Table 6. Read Cycle Bus Sizing

					16-Bit Bus ($\overline{BS8}$/BS16 = 1)		8-Bit Bus ($\overline{BS8}$/BS16 = 0)	
Type	Address	A_0	$\overline{UBE}$	Cycle	D_{15}-D_8	D_7-D_0	D_{15}-D_8	D_7-D_0
Byte	Even	0	1	1st	Not used	Byte	Not used	Byte
	Odd	1	0	1st	Byte	Not used	Not used	Byte
Word	Even	0	1	1st	Upper	Lower	Not used	Lower
		1	0	2nd	Not needed for 16-bit bus		Not used	Upper
	Odd	1	0	1st	Lower	Lower	Not used	Lower
		0	1	2nd	Not used	Upper	Not used	Upper

3e

I/O and memory reads are used to transfer data to the μPD70136 from an I/O device or a memory location, respectively. Instruction fetches are used to fill the μPD70136's 8-byte instruction queue from the memory space.

Write Cycles

There are four types of write cycles. Memory writes transfer data from the μPD70136 to a memory location. I/O writes transfer data from the μPD70136 to an I/O device. Coprocessor data writes transfer data from the coprocessor to a memory location. Coprocessor writes transfer data from the μPD70136 directly to a coprocessor internal register.

Interrupt Acknowledge Cycle

The interrupt acknowledge operation takes two consecutive INTAK bus cycles. The first cycle is used to freeze the state of an external interrupt controller, such as the μPD71059. The second INTAK bus cycle reads an 8-bit vector number on D_0-D_7 supplied by the μPD71059. This vector number is used to index into the interrupt vector table to select an interrupt handler.

Halt Acknowledge Cycle

When a HALT instruction is executed, a halt acknowledge cycle is issued to notify external logic that the μPD70136 is entering standby mode. This cycle is always two clocks long; $\overline{READY}$ is ignored and $\overline{DSTB}$ is not asserted.

Hold Request and Hold Acknowledge

At times, an external bus master will need to use the μPD70136 bus. When the HLDRQ input is asserted by external logic, the μPD70136 recognizes this as a request for external bus mastership. The μPD70136 will finish the current bus operation, stop driving its address, data, and control buses, and assert HLDAK. The external device, such as the μPD71071 or μPD71037 DMA controller, may then drive the μPD70136 bus. Note that if the current bus operation involves more than one bus cycle, such as a 16-bit access to an odd address or due to dynamic bus sizing, the μPD70136 will finish both cycles before releasing the bus.

If the current instruction uses the $\overline{BUSLOCK}$ prefix, HLDRQ will be ignored. This will be indicated by the $\overline{BUSLOCK}$ output. Also, during interrupt acknowledge, $\overline{BUSLOCK}$ is asserted between the two INTAK cycles so that HLDRQ is ignored until after the second INTAK.

SYSTEM INTERFACING

System Memory Access Time

Table 7 shows the system memory access time required for 12.5-MHz and 16-MHz μPD70136 systems to run with

zero, one, two, and three wait states. This is the time from when the address bus is valid to when the external system must present the read data on the data bus. These numbers are based on the preliminary ac timing given in this document and are subject to change.

Table 7. Performance vs. Wait States

	12.5 MHz			16 MHz		
Number of Wait States	Memory Cycle Time (ns)	System Access Time (ns)	Relative Performance (%)	Memory Cycle Time (ns)	System Access Time (ns)	Relative Performance (%)
0	160	113	78	125	78	100
1	240	193	64	187.5	140.5	82
2	320	273	52	250	203	67
3	400	353	43	312.5	265.5	56

Note: Performance is relative to the 0 wait state, 16 MHz.

Wait States

Table 7 also shows the effect of wait states on performance. The μPD70136 overlaps bus interface operations in time with instruction execution. This greatly reduces the effect of wait states on performance. Each bus cycle is nominally two clocks long, while the minimum instruction time is two clocks, with many instructions taking longer. There is some idle bus time when the CPU is processing a long instruction and the prefetch queue is full. Wait states can often fill these idle states.

However, adding wait states to bus cycles reduces the bus bandwidth available for other bus masters, such as DMA controllers, since some of the idle time that would have been available to them is used for CPU cycles.

Note that in all cases, a 16-MHz μPD70136 with N+1 wait states is faster than a 12.5-MHz device with N wait states while using slower memories.

Please note also that these numbers were measured using a particular set of benchmarks and should be used for comparison purposes only. Different results will be obtained for other program mixes.

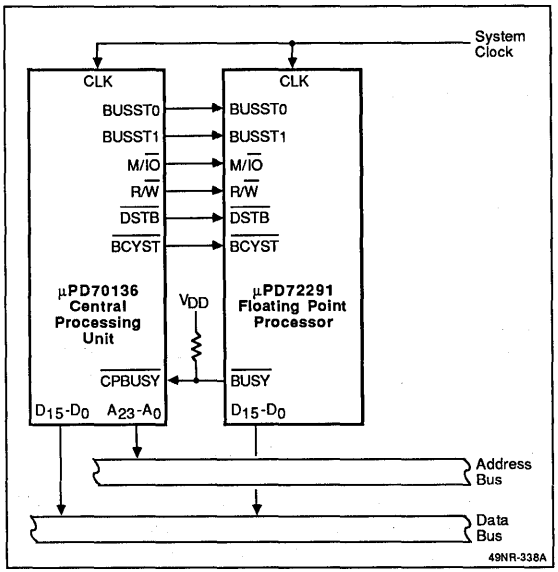
Interfacing to the μPD72291 Floating-Point Coprocessor

The μPD72291 (AFPP) is a very-high-performance floating-point coprocessor for the μPD70136 offering in excess of 530K floating-point operations per second at 16 MHz. The AFPP is programmed as an extension of the μPD70136 instruction set. The AFPP executes floating-point operations, computes transcendental functions, and performs vector multiplications.

AFPP instructions use the FP01 and FP02 formats. When one of these opcodes is encountered and an AFPP is connected, a coprocessor protocol routine is entered. The μPD70136 will compute any effective addresses required, read or write the operands for the AFPP, and instruct the AFPP as to what operation should be performed. The AFPP reponds by asserting its BUSY output when it starts the operation. The μPD70136 will not start another AFFP operation until BUSY is deasserted, but may execute CPU instructions. When BUSY is deasserted, the μPD70136 will transfer the AFPP status to the AW register.

Figure 10 shows how to connect a μPD70136 CPU to a μPD72291 AFPP. Figure 11 shows a typical system. The CPU reads and writes status and commands to the AFPP using coprocessor read and write cycles, which always take two clocks. AFPP operands are written/read using coprocessor memory write/read cycles, which always require one wait state. External $\overline{\text{READY}}$ logic must take care to include this wait state.

Figure 10. Connections Between μPD70136 and μPD72291

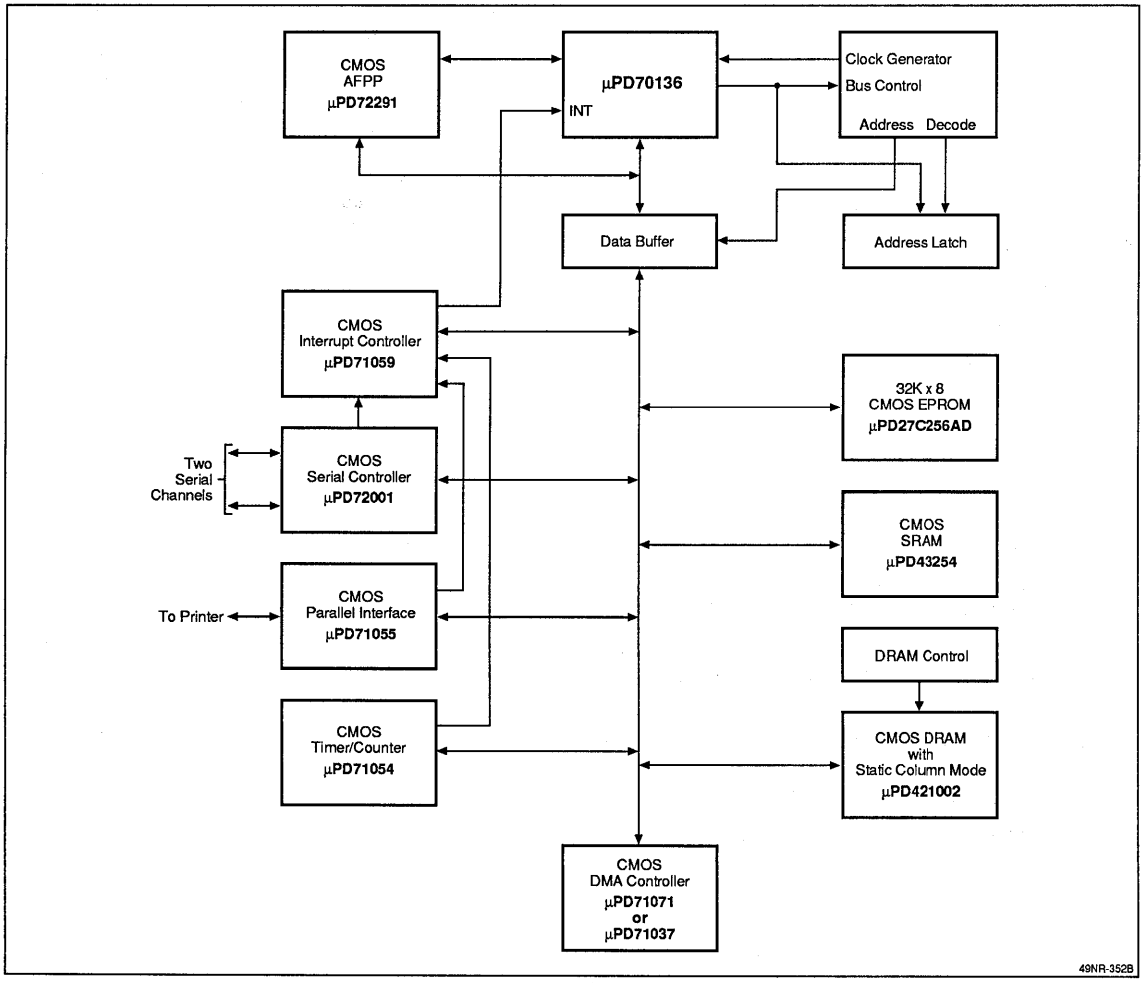


On RESET, $\overline{CPBUSY}$ is sampled. If it is low, the μPD70136 assumes that a coprocessor is connected. $\overline{CPERR}$ is also sampled to determine what kind of coprocessor is connected, as follows.

$\overline{CPBUSY}$	$\overline{CPERR}$	Coprocessor
1	X	None
0	GND	μPD72291
0	V_{DD}	Other

AFPP memory operands must always begin on even addresses and may not reside in 8-bit wide memory. Dynamic bus sizing may not be used for AFPP operands.

Figure 11. Typical μPD70136 System



3e

INTERRUPT OPERATION

The interrupts supported by the μPD70136 can be divided into two types: interrupts generated by external interrupt requests and traps generated by software processing. They are:

External Interrupts

- $\overline{\text{NMI}}$ input (nonmaskable)
- INT input (maskable)

Software Traps

- Divide error during DIV or DIVU instruction
- Array bound error during CHKIND
- Single-step (PSW BRK flag = 1)
- Undefined instruction
- Coprocessor error
- Coprocessor not connected
- Break instructions
  BRKV BRK imm8
  BRK3 BRKXA

Interrupt Priorities

Interrupts are prioritized as follows:

$\overline{\text{NMI}}$ > INT > BRK flag > others at same level

Interrupts are not accepted during certain times. $\overline{\text{NMI}}$, INT and BRK flags are not accepted in these cases:

(1) Between execution of MOV or POP that uses a segment register as an operand and the next instruction.

(2) Between a segment override prefix and the next instruction

(3) Between a repeat or $\overline{\text{BUSLOCK}}$ prefix and the next instruction

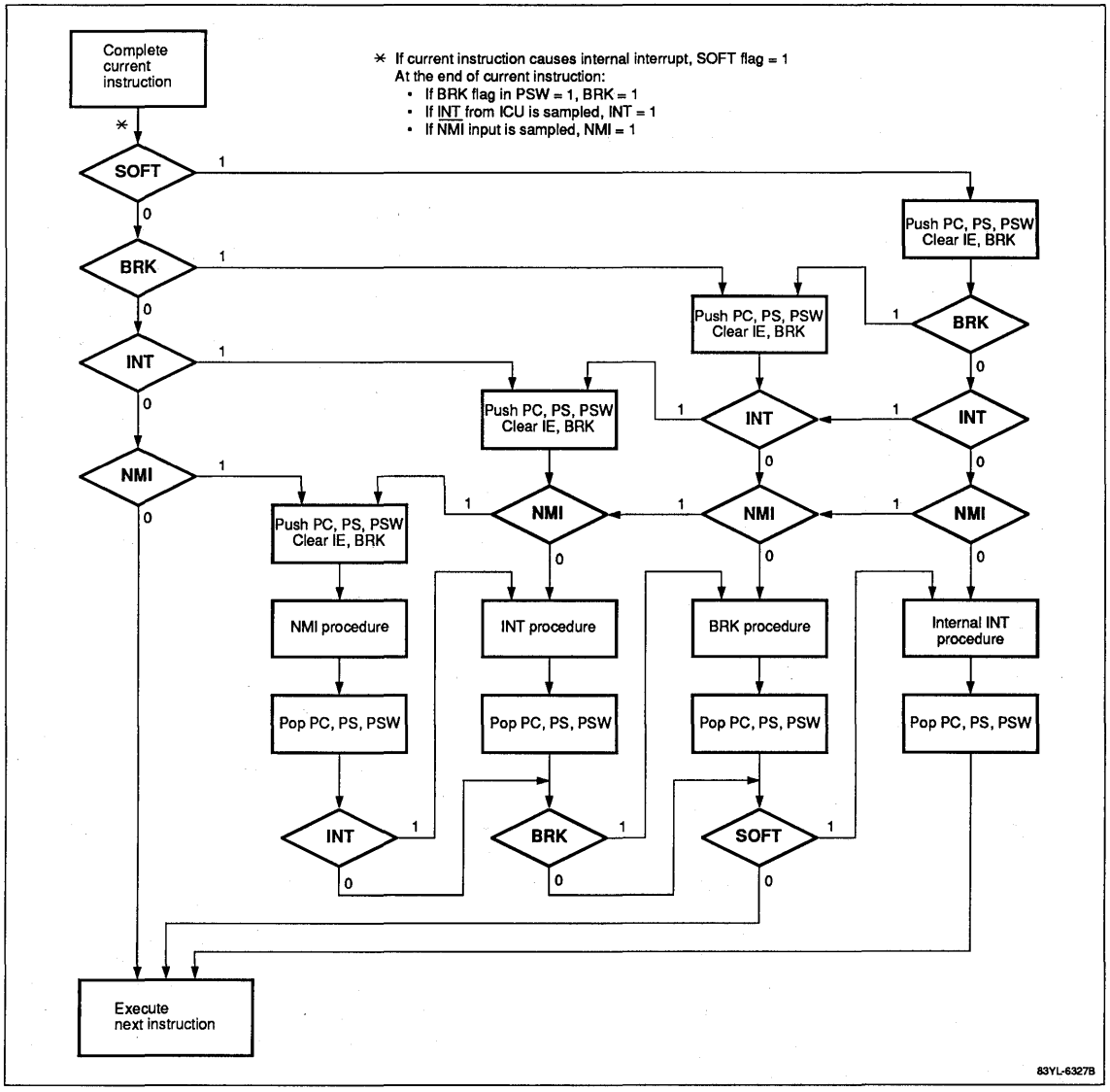
INT is not accepted when the PSW IE flag is 0, or between an RETI or POP PSW and the next instruction. Figure 12 is a flow diagram for processing interrupt requests.

Interrupt Vectors

Once an interrupt has been accepted, an interrupt service routine will be entered. The address of this routine is specified by an interrupt vector, which is stored in the interrupt vector table. For most interrupts, the vector used depends on what interrupt is being processed (e.g., $\overline{\text{NMI}}$ always uses vector 2). For INT and BRK imm8 interrupts, any vector may be used; the vector number is supplied by an external device in the case of INT (e.g., a μPD71059), or by immediate data in the case of BRK.

Figure 13 is the interrupt vector table. The table uses 1K bytes of memory—addresses 000H to 3FFH—and stores up to 256 vectors (4 bytes per vector).

Figure 12. Interrupt Prioritization Flow Diagram



Each interrupt vector consists of 4 bytes. The 2 bytes in the low addresses of memory are loaded into PC as the offset, and the 2 high bytes are loaded into PS as the base address. Interrupt vector 0 in figure 14 is an example. The bytes are combined in reverse order. The lower-order bytes in the vector become the most significant bytes in the PC and PS, and the higher-order bytes become the least significant.

Based on this format, the contents of each vector should be initialized at the beginning of the program. The basic mechanism for servicing an interrupt is:

(SP – 1, SP – 2) ← PSW
(SP – 3, SP – 4) ← PS
(SP – 5, SP – 6) ← PC
SP ← SP – 6
IE ← 0, BRK ← 0
PS ← vector high bytes
PC ← vector low bytes

Figure 13. Interrupt Vector Table

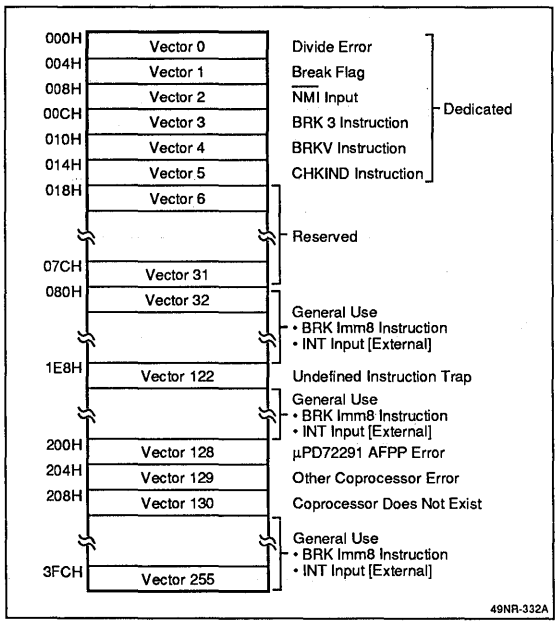
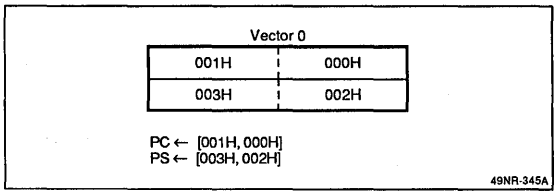


Figure 14. Interrupt Vector 0



During interrupt servicing, the third item pushed on the stack is the return PC value. For some types of traps (divide error, CHKIND, illegal opcode, AFPP error, coprocesor not present, or other CP error), this value points to the instruction that generated the trap. For the other interrupts (single-step, BRK3, BRKV, $\overline{\text{NMI}}$, or INT), this value points to the next instruction. Trap handlers for error traps can thus easily find the offending opcode, and other handlers can simply return after processing the interrupt.

STANDBY FUNCTION

The μPD70136 offers two standby modes to reduce power consumption: HALT and STOP. Both are entered after executing a HALT instruction.

HALT Standby Mode

In the HALT standby mode, the internal clock is supplied only to those circuits related to functions required to exit this mode and bus hold control functions. As a result, power consumption is reduced to one-fifth the level of normal operation.

The HALT standby mode is exited when RESET or an external interrupt ($\overline{\text{NMI}}$, INT) is received. If INT is used and interrupts were enabled before the HALT state was entered, an INTAK cycle will be performed to fetch a vector number. The interrupt service routine will be executed. After RETI, execution will resume with the instruction following the HALT. If interrupts were disabled, the interrupt service routine will not be entered, but execution will resume with the instruction following the HALT.

If $\overline{\text{NMI}}$ is used to exit the HALT standby mode, the $\overline{\text{NMI}}$ service routine will always be entered.

The bus hold (HLDRQ/HLDAK) function still operates during HALT standby mode. The CPU returns to HALT standby mode when the bus hold request is removed.

During HALT standby mode, when all control outputs go low, the address and data buses will be either high or low. Refer to table 1 for information about the states of other outputs in the standby mode.

STOP Standby Mode

In the STOP standby mode, the μPD70136 clock is stopped for maximum power reduction. To enter this mode, special steps must be taken to prepare the μPD70136 for having its clock stopped.

INT, $\overline{\text{NMI}}$ and HLDRQ must not be asserted while the μPD70136 is in STOP mode, or for at least 10 clock periods before STOP is entered, or for at least 10 clock periods after STOP mode is exited. External hardware must ensure that these intputs are not asserted during this time.

STOP mode is entered by disabling $\overline{\text{NMI}}$, INT, and HLDRQ, entering the HALT standby mode, and stopping the clock input 10 clock periods after the HALT acknowledge but cycle is issued. The CLK input must be stopped during the low phase of the clock. STOP mode is exited when external logic starts the clock, waits 10 clock periods, and enable $\overline{\text{NMI}}$, INT, and HLDRQ; the μPD70136 will return to the HALT standby mode.

All output pins in STOP mode are in the same state as in HALT standby mode. Refer to table 1.

INSTRUCTION SET HIGHLIGHTS

Enhanced Instructions

In addition to the μPD8088/86 instructions, the μPD70136 has enhanced instructions listed in table 8.

Table 8. Enhanced Instruction

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP R	Pops 8 general registers onto stack
MULL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Enhanced Stack Operation Instructions

PUSH imm. This instruction allows immediate data to be pushed onto the stack.

PUSH R; POP R. These instructions allow the contents of the eight general registers to be pushed onto or popped from the stack with a single instruction.

Enhanced Multiplication Instructions

MUL reg16, imm16; MUL mem16, imm16. These instructions allow the contents of a register or memory location to be multiplied by immediate data.

Enhanced Shift and Rotate Instructions

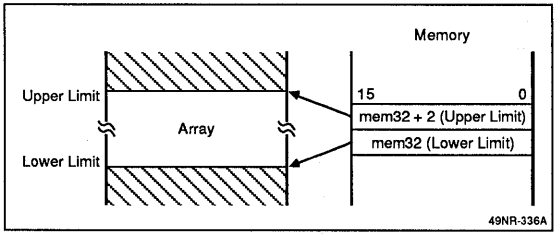
SHL reg, imm8; SHR reg, imm8; SHRA reg, imm8. These instructions allow the contents of a register to be shifted by the number of bits defined by the immediate data.

ROL reg, imm8; ROR reg imm8; ROLC reg, imm8; RORC reg, imm8. These instructions allow the contents of a register to be rotated by the number of bits defined by the immediate data.

Check Array Boundary Instruction

CHKIND reg16, mem32. This instruction is used to verify that index values pointing to the elements of an array data structure are within the defined range. See figure 15. The lower limit of the array should be in memory location mem32, the upper limit in mem32 + 2. If the index value in reg16 is not between these limits when CHKIND is executed, a BRK 5 will occur. This causes a jump to the location in interrupt vector 5.

Figure 15. Check Array Boundary



Block I/O Instruction

OUTM DW, src-block; INM dist-block, DW. These instructions are used to output or input a string to or from memory, when preceded by a repeat prefix.

3e

Stack Frame Instruction

PREPARE imm16,imm8. This instruction is used to generate the stack frames required by block-structured languages, such as PASCAL and Ada. The stack frame consists of two areas. One area has a pointer that points to another frame which has variables that the current frame can access. The other is a local variable area for the current procedure.

DISPOSE. This instruction releases that last stack frame generated by the PREPARE instruction. It returns the stack and base pointers to the values they had before the PREPARE instruction was used to call a procedure.

Unique Instructions

In addition to the μPD8088/86 instructions and the enhanced instructions, the μPD70136 has the unique instructions listed in table 9.

Table 9. Unique Instructions

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	Adds packed decimal strings
SUB4S	Subtracts one packed decimal string from another
CMP4S	Compares two packed decimal strings
ROL4	Rotates one BCD digit left through AL lower 4 bits
ROR4	Rotates one BCD digit right through AL lower 4 bits
BRKXA	Break and enable expanded addressing
RETXA	Return from break and disable expanded addressing
TEST1	Tests a specified bit and sets/resets Z flag
NOT1	Inverts a specified bit
CLR1	Clears a specified bit
SET1	Sets a specified bit
REPC	Repeats next instruction until CY flag is cleared
REPNC	Repeats next instruction until CY flag is set
FP02	Additional floating-point processor call

Variable Length Bit Field Operation Instructions

This category has two instructions: INS (Insert Bit Field) and EXT (Extract Bit Field). These instructions are highly effective for computer graphics and high-level languages. They can, for example, be used for data structures such as packed arrays and record type data used in PASCAL.

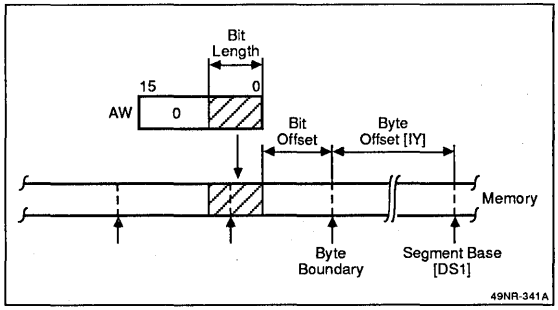
INS reg8, reg8; INS reg8, imm4. This instruction transfers low bits from the 16-bit AW register (the number of bits is specified by the second operand) to the memory location specified by the segment base (DS1 register) plus the byte offset (IY register). The starting bit position within this byte is specified as an offset by the lower 4 bits of the first operand. See figure 16.

After each complete data transfer, the IY register and the register specified by the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may specify the number of bits transferred (second operand). Because the maximum transferable bit length is 16 bits, only the lower 4 bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Figure 16. Bit Field Insertion



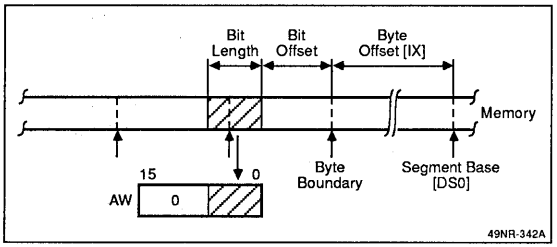
EXT reg8, reg8; EXT reg8, imm4. This instruction loads to the AW registers the bit field data whose bit length is specified by the second operand of the instruction from the memory location that is specified by the DS0 segment register (segment base), the IX index register (byte offset), and the lower 4 bits of the first operand (bit offset). See figure 17.

After the transfer is complete, the IX register and the lower 4 bits of the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may be specified for the second operand. Because the maximum transferable bit length is 16 bits, however, only the lower 4 bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Figure 17. Bit Field Extraction



Packed BCD Operation Instructions

The instructions described here process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte-format operands (ROR4, ROL4). Packed BCD strings may be from 1 to 254 digits in length.

When the number of digits is even, the zero (Z) and carry (CY) flags will be set according to the result of the operation. When the number of digits is odd, the Z and CY flags may not be set correctly. In this case (CL = odd), the Z flag will not be set unless the upper 4 bits of the highest byte are all 0s. The CY flag will not be set unless there is a carry out of the upper 4 bits of the highest byte. When CL is odd, the contents of the upper 4 bits of the highest byte of the result are undefined.

ADD4S. This instruction adds the packed BCD string addressed by the IX index register to the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the V (overflow), CY, and Z flags .

BCD string (IY, CL) ← BCD string (IY, CL) + BCD string (IX, CL)

SUB4S. This instruction subtracts the packed BCD string addressed by the IX index register from the packed BCD string addressed by the IY register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the V, CY, and Z flags.

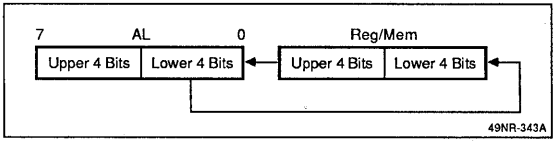
BCD string (IY, CL) ← BCD string (IY, CL) – BCD string (IX, CL)

CMP4S. This instruction performs the same operation as SUB4S except that the result is not stored and only the V, CY, and Z flags are affected.

BCD string (IY, CL) – BCD string (IX, CL)

ROL4. This instruction treats the byte data of the register or memory operand specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the left. See figure 18.

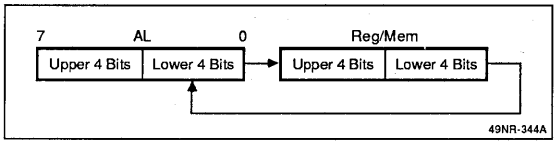
Figure 18. BCD Rotate Left



ROR4. This instruction treats the byte data of the register or memory specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the right. See figure 19.

3e

Figure 19. BCD Rotate Right



Bit Manipulation Instructions

TEST1. This instruction tests a specific bit in a register or memory location. If the bit is 1, the Z flag is reset to 0. If the bit is 0, the Z flag is set to 1.

NOT1. This instruction inverts a specific bit in a register or memory location.

CLR1. This instruction clears a specific bit in a register or memory location.

SET1. This instruction sets a specific bit in a register or memory location.

Repeat Prefix Instructions

REPC. This instruction causes the μPD70136 to repeat the following primitive block transfer instruction until the CY flag becomes cleared or the CW register becomes zero.

REPNC. This instruction causes the μPD70136 to repeat the following primitive block transfer instruction until the CY flag becomes set or the CW register becomes zero.

Address Expansion Control Instructions

BRKXA imm8. This instruction is used to turn on expanded addressing. The 8-bit immediate data specifies an interrupt vector. The PC field of this vector is loaded into the PC (and PFP). The XA flag in the XAM register is set to 1, thereby enabling the expanded addressing

mode. The μPD70136 will begin fetching from the new PFP through the address translation table. That is, the new PC is treated as a logical address and is translated to the new, larger physical address space.

This instruction does not save any return address information, such as PC, PS, or PSW to the stack.

RETXA imm8. This instruction is used to turn off expanded addressing. It is identical in operation to BRKXA, except that the expanded addressing mode is turned off before fetching from the new address. That is, the XA flag in the XAM register is set to 0, and the PC is loaded with the value of the PC field in the interrupt vector selected by the immediate data.

This instruction does not save any return address information such as PC, PS, or PSW to the stack.

Porting μPD70116/70108 Code to μPD70136

The μPD70136 is completely software compatible with the μPD70116/70108. However, the μPD70136 offers some improvements that may affect the porting of μPD70116 code to the μPD70136. These improvements are:

(1) The μPD70116 does not trap on undefined opcodes. The μPD70136 will trap, and also will trap when a register addressing mode is used for any of these instructions:

```
CHKIND        LDEA
MOV DS0/DS1   BR 1,id
CALL 1,id
```

(2) During signed division (DIV), if the quotient is 80H (byte operation) or 8000H (word), the μPD70116 will take a Divide By 0 trap. The μPD70136 will perform the calculation.

(3) When the μPD70116 executes the POLL instruction, it will wait for the POLL input signal to be asserted. The μPD70136 has no POLL input; instead, when this instruction is executed, if a coprocessor is not connected, then a Coprocessor Not Present trap will be taken. If a coprocessor is attached, then no operation takes place.

The μPD70116 accepts FP01 and FP02 as opcodes for the iAPX8087 coprocessor. The μPD70136 accepts these as opcodes for the μPD72291 coprocessor, which is not compatible with the iAPX8087.

(4) During the POP R instruction, the μPD70116 does not restore the SP register. The μPD70136 does restore the SP.

(5) When processing a divide error, the μPD70116 saves the address of the next instruction. The μPD70136 saves the address of the current instruction (the divide instruction).

(6) The μPD70116 allows up to 3 prefix instructions in any combination. The μPD70136 also allows 3 prefixes, but only one of each type can be used. The μPD70136 could operate incorrectly if there are two prefixes of the same type. For example, consider:

```
REP
REPC
CMPBK SS: src-block, dst-block
```

If the compare operation is interrupted, then when it resumes following the interrupt service, execution will begin at the REPC instruction, not the REP instruction, because two repeat prefixes were used.

(7) The μPD70116 accepts $\overline{\text{NMI}}$ requests even while processing an NMI. The μPD70136 does not allow nesting of NMIs; the NMI input will be ignored until the NMI interrupt handler is exited.

INSTRUCTION SET

Symbols

Preceding the instruction set, several tables explain symbols, abbreviations, and codes.

Clocks

In the Clocks column of the instruction set, the numbers cover these operations: instruction decoding, effective address calculation, operand fetch, and instruction execution.

Clock timings assume the instruction has been prefetched and is present in the 8-byte instruction queue. Otherwise, add two clocks for each pair of bytes not present.

Word operands require two additional clocks for each transfer to an unaligned (odd address) memory operand. These times are shown on the right side of the slash (/).

For conditional control transfer or branch instructions, the number on the left side of the slash is applicable if the transfer or branch takes place. The number on the right side is applicable if it does not take place.

If a range of numbers is given, the execution time depends on the operands involved.

Symbols

Symbol	Meaning
acc	Accumulator(AW or AL)
duso	Displacement (8 or 16 bits)
dmem	Direct memory address
dst	Destination operand or address
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
far_label	Label within a different program segment
far_proc	Procedure within a different program segment
fp_op	Floating-point instruction operation
imm	8- or 16-bit immediate operand
imm3/4	3- or 4-bit immediate bit offset
imm8	8-bit immediate operand
imm16	16-bit immediate operand
mem	Memory field (000 to 111); 8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
memptr16	Word containing the destination address within the current segment
memptr32	Double word containing a destination address in another segment
mod	Mode field (00 to 10)
near_label	Label within the current segment
near_proc	Procedure within the current segment
offset	Immediate offset data (16 bits)
pop_value	Number of bytes to discard from the stack
reg	Register field (000 to 111); 8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
regptr	16-bit register containing a destination address within the current segment
regptr16	Register containing a destination address within the current segment
seg	Immediate segment data (16 bits)
short_label	Label between −128 and +127 bytes from the end of the current instruction
sr	Segment register
src	Source operand or address
temp	Temporary register (8/16/32 bits)
AC	Auxiliary carry flag
AH	Accumulator (high byte)
AL	Accumulator (low byte)

Symbol	Meaning
AW	Accumulator (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
BP	BP register
BRK	Break flag
BW	BW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
CW	CW register (16 bits)
CY	Carry flag
DH	DW register (high byte)
DIR	Direction flag
DL	DW register (low byte)
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DW	DW register (16 bits)
IE	Interrupt enable flag
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
MD	Mode flag
P	Parity flag
PC	Program counter (16 bits)
PS	Program segment register (16 bits)
PSW	Program status word (16 bits)
R	Register set
S	Sign extend operand field S = No sign extension S = Sign extend immediate byte operand
S	Sign flag
SP	Stack pointer (16 bits)
SS	Stack segment register (16 bits)
V	Overflow flag
W	Word/byte field (0 to 1)
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating-point arithmetic chip
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value
Z	Zero flag

Flag Operations

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to result
u	Undefined
R	Restored to previous state

Memory Addressing Modes

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Register Selection (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Register Selection

sr	Segment Register
00	DS1
01	PS
10	SS
11	DS0

Instruction Set

		Opcode																		Flags					
Mnemonic	Operand	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	AC	CY	V	P	S	Z
Data Transfer Instructions																									
MOV	reg, reg	1	0	0	0	1	0	1	W	1	1		reg			reg		2	2						
	mem, reg	1	0	0	0	1	0	0	W	mod			reg			mem		3/5	2-4						
	reg, mem	1	0	0	0	1	0	1	W	mod			reg			mem		5/7	2-4						
	mem, imm	1	1	0	0	0	1	1	W	mod			000			mem		3/5	3-6						
	reg, imm	1	0	1	1	W		reg										2	2-3						
	acc, dmem	1	0	1	0	0	0	0	W									5/7	3						
	dmem, acc	1	0	1	0	0	0	1	W									3/5	3						
	sr, reg16	1	0	0	0	1	1	1	0	1	1	0	sr			reg		2	2						
	sr, mem16	1	0	0	0	1	1	1	0	mod		0	sr			mem		5/7	2-4						
	reg16, sr	1	0	0	0	1	1	0	0	1	1	0	sr			reg		2	2						
	mem16, sr	1	0	0	0	1	1	0	0	mod		0	sr			mem		3/5	2-4						
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod			reg			mem		10/14	2-4						
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod			reg			mem		10/14	2-4						
	AH, PSW	1	0	0	1	1	1	1	1									2	1						
	PSW, AH	1	0	0	1	1	1	1	0									2	1	x	x		x	x	x
LDEA	reg16, mem16	1	0	0	0	1	1	0	0	mod			reg			mem		2	2-4						
TRANS	src_table	1	1	0	1	0	1	1	1									5	1						
XCH	reg, reg	1	0	0	0	0	1	1	W	1	1		reg			reg		3	2						
	mem, reg	1	0	0	0	0	1	1	W	mod			reg			mem		8/12	2-4						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer Instructions (cont)																									
XCH (cont)	AW, reg16	1	0	0	1	0		reg										3	1						
Repeat Prefixes																									
REPC		0	1	1	0	0	1	0	1									2	1						
REPNC		0	1	1	0	0	1	0	0									2	1						
REP REPE REPZ		1	1	1	1	0	0	1	1									2	1						
REPNE REPNZ		1	1	1	1	0	0	1	0									2	1						
Block Transfer Instructions																									
MOVBK	dst, src	1	0	1	0	0	1	0	W										1						
																		3 + 4n (W = 0) 3 + 4n (W = 1, even addresses) 3 + 8n (W = 1, odd addresses) 3 + 6n (W = 1, odd/even addresses)						
CMPBK	dst, src	1	0	1	0	0	1	1	W										1	x	x	x	x	x	x
																		3 + 7n (W = 0) 3 + 7n (W = 1, even addresses) 3 + 11n (W = 1, odd addresses) 3 + 9n (W = 1, odd/even addresses)						
CMPM	dst	1	0	1	0	1	1	1	W										1	x	x	x	x	x	x
																		3 + 5n (W = 0) 3 + 5n (W = 1, even addresses) 3 + 7n (W = 1, odd addresses)						
LDM	src	1	0	1	0	1	1	0	W										1						
																		5 + 2n (W = 0) 5 + 2n (W = 1, even addresses) 5 + 4n (W = 1, odd addresses)						
STM	dst	1	0	1	0	1	0	1	W										1						
																		3 + 2n (W = 0) 3 + 2n (W = 1, even addresses) 3 + 4n (W = 1, odd addresses)						
	n = number of returns String instruction execution clocks for a single-instruction execution are in parentheses.																								
I/O Instructions																									
IN	acc, imm8	1	1	1	0	0	1	0	W									5/7	2						
	acc, DW	1	1	1	0	1	1	0	W									3/5	1						
OUT	imm8, acc	1	1	1	0	0	1	1	W									3/5	2						
	DW, acc	1	1	1	0	1	1	1	W									3/5	1						

3e

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
I/O Instructions (cont)																									
INM	dst, DW	0	1	1	0	1	1	0	W										1						
																		3 + 11n (W = 0) 3 + 8n (W = 1, even addresses) 3 + 22n (W = 1, odd addresses) 3 +20n (W = 1, odd/even addresses; odd for I/O) 3 + 13n (W = 1, odd/even addresses; odd for memory)							
OUTM	DW, src	0	1	1	0	1	1	1	W										1						
																		3 + 11n (W = 0) 3 + 8n (W = 1, even addresses) 3 + 22n (W = 1, odd addresses) 3 + 20n (W = 1, odd addresses; odd for I/O) 3 + 13n (W = 1, odd addresses; odd for memory)							
	n = number of transfers String instruction execution clocks for a single-instruction execution are in parentheses. Use the right side of the slash (/) for DMA I/O accesses.																								
BCD Instructions																									
ADJBA		0	0	1	1	0	1	1	1									4	1	x	x	u	u	u	u
ADJ4A		0	0	1	0	0	1	1	1									2	1	x	x	u	x	x	x
ADJBS		0	0	1	1	1	1	1	1									4	1	x	x	u	u	u	u
ADJ4S		0	0	1	0	1	1	1	1									2	1	x	x	u	x	x	x
ADD4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	2 + 18n	2	u	x	u	u	u	x
SUB4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	2 + 18n	2	u	x	u	u	u	x
CMP4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	7 + 14n	2	u	x	u	u	u	x
ROL4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	9	3						
		1	1	0	0	0		reg																	
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	15	3-5						
		mod		0	0	0		mem																	
ROR4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	13	3						
		1	1	0	0	0		reg																	
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	19	3-5						
		mod		0	0	0		mem																	
	n = number of BCD digits divided by 2																								
Data Type Conversion Instructions																									
CVTBD		1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	12	2	u	u	u	x	x	x
CVTDB		1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	8	2	u	u	u	x	x	x
CVTBW		1	0	0	1	1	0	0	0									2	1						
CVTWL		1	0	0	1	1	0	0	1									2	1						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Arithmetic Instructions																									
ADD	reg, reg	0	0	0	0	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	0	0	0	0	W	mod			reg			mem		7/11	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	0	0	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	0	0		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	0	0		mem		7/11	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	0	0	1	0	W									2	2-3	x	x	x	x	x	x
ADDC	reg, reg	0	0	0	1	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	0	0	0	W	mod			reg			mem		7/11	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	0	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	0		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	0		mem		7/11	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	0	1	0	W									2	2-3	x	x	x	x	x	x
SUB	reg, reg	0	0	1	0	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	1	0	1	0	0	W	mod			reg			mem		7/11	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	0	1	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	0	1		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	0	1		mem		7/11	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	0	1	1	0	W									2	2-3	x	x	x	x	x	x
SUBC	reg, reg	0	0	0	1	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	1	0	0	W	mod			reg			mem		7/11	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	1	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	1		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	1		mem		7/11	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	1	1	0	W									2	2-3	x	x	x	x	x	x
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	0		mem		7/11	2-4	x		x	x	x	x
	reg16	0	1	0	0	0		reg										2	1	x		x	x	x	x
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	1		mem		7/11	2-4	x		x	x	x	x
	reg16	0	1	0	0	1		reg										2	1	x		x	x	x	x
MULU	reg8	1	1	1	1	0	1	1	0	1	1	1	0	0		reg		8	2	u	x	x	u	u	u
	reg16	1	1	1	1	0	1	1	1	1	1	1	0	0		reg		12	2	u	x	x	u	u	u
	mem8	1	1	1	1	0	1	1	0	mod		1	0	0		mem		12	2-4	u	x	x	u	u	u
	mem16	1	1	1	1	0	1	1	1	mod		1	0	0		mem		16/18	2-4	u	x	x	u	u	u
MUL	reg8	1	1	1	1	0	1	1	0	1	1	1	0	1		reg		8	2	u	x	x	u	u	u
	reg16	1	1	1	1	0	1	1	1	1	1	1	0	1		reg		12	2	u	x	x	u	u	u
	mem8	1	1	1	1	0	1	1	0	mod		1	0	1		mem		12	2-4	u	x	x	u	u	u
	mem16	1	1	1	1	0	1	1	1	mod		1	0	1		mem		16/18	2-4	u	x	x	u	u	u
	reg16, reg16, imm8	0	1	1	0	1	0	1	1	1	1		reg			reg		12	3	u	x	x	u	u	u

3e

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Arithmetic Instructions (cont)																									
MUL (cont)	reg16, mem16, imm8	0	1	1	0	1	0	1	1	mod			reg			mem		16/18	3-5	u	x	x	u	u	u
	reg16, reg16, imm16	0	1	1	0	1	0	0	1	1	1		reg			reg		12	4	u	x	x	u	u	u
	reg16, mem16, imm16	0	1	1	0	1	0	0	1	mod			reg			mem		16/8	4-6	u	x	x	u	u	u
DIVU	reg8	1	1	1	1	0	1	1	0	1	1	1	1	0		reg		11	2	u	u	u	u	u	u
	reg16	1	1	1	1	0	1	1	1	1	1	1	1	0		reg		19	2	u	u	u	u	u	u
	mem8	1	1	1	1	0	1	1	0	mod		1	1	0		mem		15	2-4	u	u	u	u	u	u
	mem16	1	1	1	1	0	1	1	1	mod		1	1	0		mem		23/25	2-4	u	u	u	u	u	u
DIV	reg8	1	1	1	1	0	1	1	0	1	1	1	1	1		reg		16	2	u	u	u	u	u	u
	reg16	1	1	1	1	0	1	1	1	1	1	1	1	1		reg		24	2	u	u	u	u	u	u
	mem8	1	1	1	1	0	1	1	0	mod		1	1	1		mem		20	2-4	u	u	u	u	u	u
	mem16	1	1	1	1	0	1	1	1	mod		1	1	1		mem		28/30	2-4	u	u	u	u	u	u
Comparison Instructions																									
CMP	reg, reg	0	0	1	1	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	1	1	1	0	0	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	1	1	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	1	1		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	1	1		mem		6/8	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	1	1	1	0	W									2	2-3	x	x	x	x	x	x
Logical Instructions																									
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0		reg		2	2						
	mem	1	1	1	1	0	1	1	W	mod		0	1	0		mem		7/11	2-4						
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1		reg		2	2	x	x	x	x	x	x
	mem	1	1	1	1	0	1	1	W	mod		0	1	1		mem		7/11	2-4	x	x	x	x	x	x
TEST	reg, reg	1	0	0	0	0	1	0	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	1	0	0	0	0	1	0	W	mod			reg			mem		6/8	2-4	u	0	0	x	x	x
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0		reg		2	3-4	u	0	0	x	x	x
	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0		mem		6/8	3-6	u	0	0	x	x	x
	acc, imm	1	0	1	0	1	0	0	W									2	2-3	u	0	0	x	x	x
AND	reg, reg	0	0	1	0	0	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	1	0	0	0	0	W	mod			reg			mem		7/11	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	0	0	0	1	W	mod			reg			mem		6/8	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	0	0		reg		2	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0		mem		7/11	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									2	2-3	u	0	0	x	x	x
OR	reg, reg	0	0	0	0	1	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	0	0	1	0	0	W	mod			reg			mem		7/11	2-4	u	0	0	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Logical Instructions (cont)																									
OR (cont)	reg, mem	0	0	0	0	1	0	1	W	mod			reg			mem		6/8	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	0	0	1		reg		2	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1		mem		7/11	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									2	2-3	u	0	0	x	x	x
XOR	reg, reg	0	0	1	1	0	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	1	1	0	0	0	W	mod			reg			mem		7/11	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	1	0	0	1	W	mod			reg			mem		6/8	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	1	0		reg		2	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0		mem		7/11	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	1	0	1	0	W									2	2-3	u	0	0	x	x	x
Bit Manipulation Instructions																									
INS	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	37-61/	3						
		1	1		reg			reg										39-77							
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	37-69/	4						
		1	1	0	0	0		reg										39-77							
EXT	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	29-61/	3						
		1	1		reg			reg										33-63							
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	29-61/	4						
		1	1	0	0	0		reg										33-63							
TEST1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	4	3	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	8	3-5	u	0	0	u	u	x
		mod		0	0	0		mem																	
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1	8/10	3-5	u	0	0	u	u	x
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	4	4	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	0	13	4-6	u	0	0	u	u	x
		mod		0	0	0		mem																	
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1	8/10	4-6	u	0	0	u	u	x
		mod		0	0	0		mem																	
SET1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	9	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	4	4						
		1	1	0	0	0		reg																	
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	9	4-6						
		mod		0	0	0		mem																	

3e

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Bit Manipulation Instructions (cont)																									
SET1 (cont)	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	1	9/13	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	1									2	1		1				
	DIR	1	1	1	1	1	1	0	1									2	1						
CLR1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	4	3						
		1	1	0	0	0		reg																	
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	0	9	3-5						
		mod		0	0	0		mem																	
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	9/13	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	4	4						
		1	1	0	0	0		reg																	
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0	9	4-6						
		mod		0	0	0		mem																	
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1	9/13	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	0									2	1		0				
	DIR	1	1	1	1	1	1	0	0									2	1						
NOT1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	4	3						
		1	1	0	0	0		reg																	
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0	9	3-5						
		mod		0	0	0		mem																	
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	9/13	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	4	4						
		1	1	0	0	0		reg																	
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	9	4-6						
		mod		0	0	0		mem																	
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1	9/13	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	0	1	0	1									2	1		x				
Shift/Rotate Instructions																									
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0		reg		2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0		mem		7/11	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0		reg		2 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0		mem		6/10 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0		reg		2 + n	3	u	x	u	x	x	x
																		n = number of shifts							

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions (cont)																									
SHL (cont)	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0	mem			6/10 + n	3-5	u	x	u	x	x	x
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1	mem			7/11	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg			2 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1	mem			6/10 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg			2 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1	mem			6/10 + n	3-5	u	x	u	x	x	x
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1	reg			2	2	u	x	0	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1	mem			7/11	2-4	u	x	0	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1	reg			2 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1	mem			6/10 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1	reg			2 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1	mem			6/10 + n	3-5	u	x	u	x	x	x
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	0	mem			7/11	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0	reg			2 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	0	mem			6/10 + n	2-4		x	u			
	reg, imm	1	1	0	0	0	0	0	W	1	1	0	0	0	reg			2 + n	3		x	u			
	mem, imm	1	1	0	0	0	0	0	W	mod		0	0	0	mem			6/10 + n	3-5		x	u			
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1	reg			2 + n	2		x	u			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	1	mem			7/11	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	1	mem			6/10 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1	reg			2 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	1	mem			6/10 + n	3-5		x	u			
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0	mem			7/11	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0	reg			2 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0	mem			6/10 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0	reg			2 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0	mem			6/10 + n	3-5		x	u			
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1	mem			7/11	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1	reg			2 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1	mem			6/10 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1	reg			2 + n	3		x	u			

n = number of shifts

3e

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions (cont)																									
RORC (cont)	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1		mem		6/10 + n	3-5		x	u			
Stack Manipulation Instructions																									
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0		mem		5/9	2-4						
	reg16	0	1	0	1	0		reg										3/5	1						
	sr	0	0	0		sr	1	1	0									3/5	1						
	PSW	1	0	0	1	1	1	0	0									3/5	1						
	R	0	1	1	0	0	0	0	0									20/36	1						
	Imm	0	1	1	0	1	0	S	0									3/5	2-3						
POP	mem16	1	0	0	0	1	1	1	1	mod		0	0	0		mem		5/9	2-4						
	reg16	0	1	0	1	1		reg										5/7	1						
	sr	0	0	0		sr	1	1	1									5/7	1						
	PSW	1	0	0	1	1	1	0	1									5/7	1	R	R	R	R	R	R
	R	0	1	1	0	0	0	0	1									22/38	1						
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0									*	4						
										*imm8 = 0:15 imm8≥ 1: 17 + 12 (imm8 – 1) odd, 15 + 8 (imm8-1) even														
DISPOSE		1	1	0	0	1	0	0	1									6/10	1						
Control Transfer Instructions																									
CALL	near_proc	1	1	1	0	1	0	0	0									7/9	3						
	regptr16	1	1	1	1	1	1	1	1	1	1	0	1	0		reg		7/9	2						
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0		mem		11/15	2-4						
	far_proc	1	0	0	1	1	0	1	0									9/13	5						
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1		mem		15/23	2-4						
RET		1	1	0	0	0	0	1	1									10/12	1						
	pop_value	1	1	0	0	0	0	1	0									10/12	3						
		1	1	0	0	1	0	1	1									12/16	1						
	pop_value	1	1	0	0	1	0	1	0									12/16	3						
BR	near_label	1	1	1	0	1	0	0	1									7	3						
	short_label	1	1	1	0	1	0	1	1									7	2						
	regptr16	1	1	1	1	1	1	1	1	1	1	1	0	0		reg		7	2						
	memptr16	1	1	1	1	1	1	1	1	mod		1	0	0		mem		11/13	2-4						
	far_label	1	1	1	0	1	0	1	0									7	5						
	memptr32	1	1	1	1	1	1	1	1	mod		1	0	1		mem		13/17	2-4						
BV	short_label	0	1	1	1	0	0	0	0									3/6	2						
BNV	short_label	0	1	1	1	0	0	0	1									3/6	2						
BC, BL	short_label	0	1	1	1	0	0	1	0									3/6	2						
																		n = number of shifts							

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Control Transfer Instructions (cont)																									
BNC, BNL	short_label	0	1	1	1	0	0	1	1									3/6	2						
BE, BZ	short_label	0	1	1	1	0	1	0	0									3/6	2						
BNE, BNZ	short_label	0	1	1	1	0	1	0	1									3/6	2						
BNH	short_label	0	1	1	1	0	1	1	0									3/6	2						
BH	short_label	0	1	1	1	0	1	1	1									3/6	2						
BN	short_label	0	1	1	1	1	0	0	0									3/6	2						
BP	short_label	0	1	1	1	1	0	0	1									3/6	2						
BPE	short_label	0	1	1	1	1	0	1	0									3/6	2						
BPO	short_label	0	1	1	1	1	0	1	1									3/6	2						
Interrupt Instructions																									
BLT	short_label	0	1	1	1	1	1	0	0									3/6	2						
BGE	short_label	0	1	1	1	1	1	0	1									3/6	2						
BLE	short_label	0	1	1	1	1	1	1	0									3/6	2						
BGT	short_label	0	1	1	1	1	1	1	1									3/6	2						
DBNZNE	short_label	1	1	1	0	0	0	0	0									3/6	2						
DBNZE	short_label	1	1	1	0	0	0	0	1									3/6	2						
DBNZ	short_label	1	1	1	0	0	0	1	0									3/6	2						
BCWZ	short_label	1	1	1	0	0	0	1	1									3/6	2						
BRK	3	1	1	0	0	1	1	0	0									18/24	1						
	imm8	1	1	0	0	1	1	0	1									18/24	2						
BRKV	imm8	1	1	0	0	1	1	1	0									20/26	1						
RETI		1	1	0	0	1	1	1	1									13/19	1	R	R	R	R	R	R
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod		reg			mem			24-26/ 30-32	2-4						
CPU Control Instructions																									
HALT		1	1	1	1	0	1	0	0									2	1						
BUSLOCK		1	1	1	1	0	0	0	0									2	1						
FP01	fp_op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	*	2						
	fp_op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y		mem		*	2-4						
FP02	fp_op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	*	2						
	fp_op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y		mem		*	2-4						
POLL		1	0	0	1	1	0	1	1									2 + 5n	1						
												n = number of times POLL pin is sampled.													
NOP		1	0	0	1	0	0	0	0									3	1						
DI		1	1	1	1	1	0	1	0									2	1						
EI		1	1	1	1	1	0	1	1									2	1						
DS0:, DS1:, PS:, SS: (segment override prefixes)		0	0	1	seg		1	1	0									2	1						

3e

Instruction Set (cont)

		Opcode																		Flags					
Mnemonic	**Operand**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Clocks**	**Bytes**	**AC**	**CY**	**V**	**P**	**S**	**Z**
Address Expansion Control Instructions																									
BRKXA	imm8	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	12	3						
					imm8																				
RETXA	imm8	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	12	3						
					imm8																				

μPD70236 (V53) 16-Bit Microprocessor: High-Speed, High-Integration, CMOS

Description

The V53™ is a high-speed, high-integration 16-bit CMOS microprocessor with a CPU that is object and source code compatible with the V20®/V30®. Integrated on the same die is a 4-channel DMA controller, a UART, three timer/counters, an interrupt controller, a refresh controller, a clock generator, and a bus controller.

(1) The DMA unit has four channels of high-bandwidth DMA (up to 8M bytes/sec). It has two sets of control registers, one compatible with the μPD71087/8237 and another with the μPD71071.

(2) The UART offers asynchronous serial I/O and is functionally compatible with the μPD71051 (8251).

(3) The three 16-bit general-purpose timer/counters are compatible with the μPD71054 (8254).

(4) The interrupt controller is identical to the μPD71059 (8259) and offers eight interrupt channels. External μPD71059s may be cascaded.

(5) The refresh controller generates a 16-bit refresh cycle for use with dynamic or pseudostatic RAMs.

(6) The clock generator uses a crystal at two times the desired frequency to produce the internal clock for the CPU and peripherals. A peripheral clock is also output.

(7) The bus controller generates μPD71088-style control signals for easy interface to external devices. The full V33 bus is also provided. Bus cycles are nominally two clock cycles long and can be extended using the internal wait state generator. Dynamic bus sizing can be used to set the data-path width for every bus cycle. Both 8- and 16-bit cycles are supported, allowing the V53 to be used on both 8- and 16-bit systems.

The V53 CPU is identical to the μPD70136 (V33™). Hardwired data-path control and a high-bandwidth bus give a performance level of 16 MHz, which is increased to four times that of the 10-MHz V30. The 1M-byte addressing range of the V30 is to 16M bytes using an on-chip address translation table.

V20 and V30 are registered trademarks of NEC Corporation.
V33, V40, V50, and V53 are trademarks of NEC Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.

The V53 instruction set is upward compatible with the native modes of the V20, V30, V40™, and V50™. It includes bit processing, bit field insertion and extraction, and BCD string arithmetic. Using a modified Booth's algorithm, the 16-MHz V53 executes 16-bit multiplies in 750 ns. The CPU performance is the highest currently available in a high-integration microprocessor.

The V53 has an undefined instruction trap that allows instructions not part of the V-series instruction set (such as commands for proprietary MMUs) to be emulated. High-speed numerics support is provided by the μPD72291 CMOS floating-point unit (530K FLOPs at 16 MHz).

The V53's combination of high-speed CPU and DMA makes it ideal for high-bandwidth data control applications such as disk or LAN controllers. The high integration and software compatibility of the CPU and peripherals with the V33 and V30 makes the V53 ideal for very compact personal computer applications such as diskless work stations and lap top computers, or embedded MS-DOS® compatible PCs for POS terminals or control applications.

Features

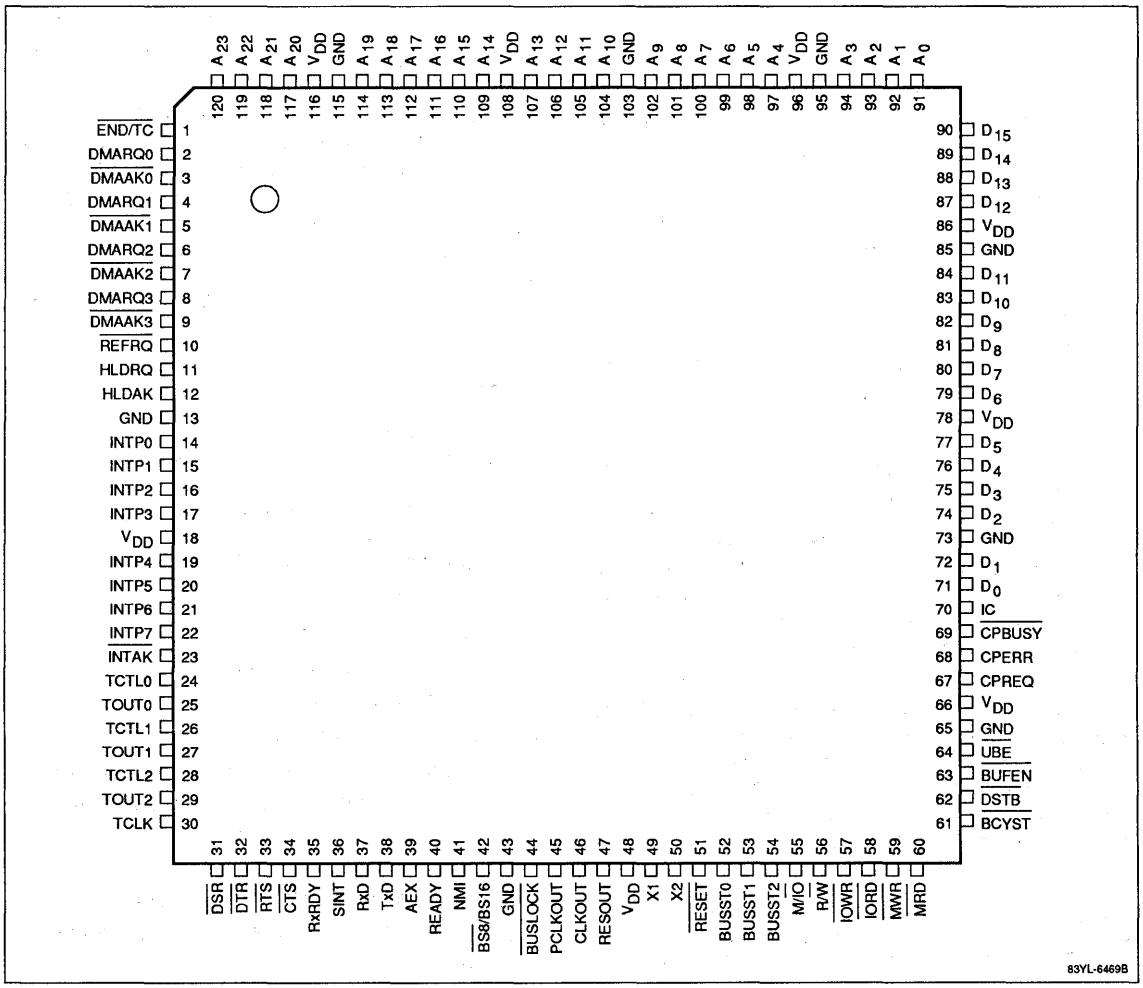
- High-speed, V30-compatible CPU
  - 125-ns minimum instruction execution time at 16 MHz
  - 750-ns 16-bit multiply at 16 MHz
  - 1.19 μs 16-bit divide (16 MHz)
  - Fastest high-integration MPU available
- Dual bus architecture
- 8-byte instruction queue
- Expanded LIM 4.0-compatible 24-bit addressing
- Four DMA channels (to 8M bytes/sec)
- On-chip serial I/O controller
- Three μPD71054-compatible 16-bit counter/timers
- Eight-channel μPD71059-compatible interrupt controller
- Refresh controller
- Bus controller with wait-state generator
- Clock generator with STOP mode control for low power
- 16-MHz (or 12.5-MHz) operation with 32-MHz (or 25-MHz) crystal

Ordering Information

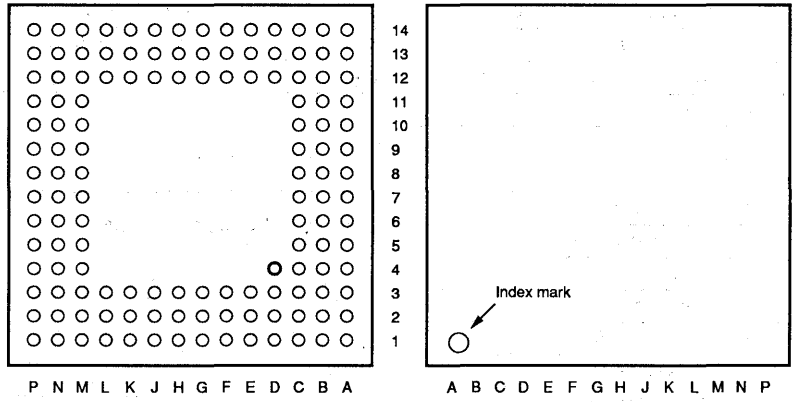
Part Number	Clock (MHz)	Package
μPD70236GD-10	10	120-pin plastic QFP
GD-12	12	
GD-16	16	
R-10	10	132-pin ceramic PGA
R-12	12	
R-16	16	

Pin Configurations

120-Pin Plastic QFP



132-Pin Ceramic PGA



Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
A1	A_{22}	B9	A_9	D3	DMARQ0	H1	INTP2	L13	GND	N7	$\overline{BUSLOCK}$
A2	A_{20}	B10	A_5	D12	D_{14}	H2	INTP3	L14	CPERR	N8	RESOUT
A3	GND	B11	GND	D13	IC	H3	V_{DD}	M1	TOUT0	N9	X2
A4	A_{19}	B12	A_2	D14	D_{11}	H12	GND	M2	TCTL2	N10	BUSST0
A5	A_{16}	B13	IC	E1	$\overline{HLDRQ}$	H13	D_2	M3	$\overline{TCLK}$	N11	R/$\overline{W}$
A6	A_{14}	B14	D_{12}	E2	$\overline{DMAAK3}$	H14	D_3	M4	$\overline{DTR}$	N12	$\overline{IORD}$
A7	A_{12}	C1	$\overline{DMAAK2}$	E3	DMARQ2	J1	INTP4	M5	RxRDY	N13	$\overline{BCYST}$
A8	A_{11}	C2	$\overline{DMAAK0}$	E12	V_{DD}	J2	INTP5	M6	AEX	N14	$\overline{UBE}$
A9	NC	C3	IC	E13	D_{10}	J3	INTP7	M7	GND	P1	$\overline{DSR}$
A10	A_8	C4	A_{23}	E14	D_8	J12	IC	M8	V_{DD}	P2	$\overline{CTS}$
A11	A_6	C5	IC	F1	NC	J13	D_1	M9	BUSST1	P3	SINT
A12	A_4	C6	A_{18}	F2	$\overline{HLDAK}$	J14	NC	M10	$\overline{IC}$	P4	TxD
A13	A_3	C7	V_{DD}	F3	$\overline{REFRQ}$	K1	INTP6	M11	$\overline{MRD}$	P5	READY
A14	A_0	C8	GND	F12	D_9	K2	$\overline{INTAK}$	M12	$\overline{IC}$	P6	BS8/BS16
B1	DMARQ1	C9	A_7	F13	D_7	K3	TCTL1	M13	$\overline{BUFEN}$	P7	PCLKOUT
B2	END/$\overline{TC}$	C10	V_{DD}	F14	D_6	K12	V_{DD}	M14	CPREQ	P8	CLKOUT
B3	A_{21}	C11	A_1	G1	INTP1	K13	CPBUSY	N1	TOUT1	P9	X1
B4	V_{DD}	C12	D_{15}	G2	INTP0	K14	D_0	N2	$\overline{IC}$	P10	$\overline{RESET}$
B5	A_{17}	C13	D_{13}	G3	GND	L1	TCTL0	N3	$\overline{RTS}$	P11	BUSST2
B6	A_{15}	C14	GND	G12	V_{DD}	L2	IC	N4	IC	P12	M/$\overline{IO}$
B7	A_{13}	D1	$\overline{DMARQ3}$	G13	D_5	L3	TOUT2	N5	RxD	P13	$\overline{IOWR}$
B8	A_{10}	D2	$\overline{DMAAK1}$	G14	D_4	L12	$\overline{DSTB}$	N6	NMI	P14	$\overline{MWR}$

83YL-6470B

3f

Pin Identification

Symbol	I/O	Function
A_0-A_{23}	Out	Address bus
AEX	Out	Address expansion mode flag
$\overline{BCYST}$	Out	Bus cycle start
$\overline{BS8}$/BS16	In	Data bus width specification
$\overline{BUFEN}$	Out	Buffer enable
$\overline{BUSLOCK}$	Out	Bus lock flag
BUSST0-BUSST2	Out	Bus status
CLKOUT	Out	System clock
$\overline{CPBUSY}$	In	Coprocessor busy
$\overline{CTS}$	Out	Clear to send
D_0-D_{15}	I/O	Data bus
$\overline{DMAAK0}$-$\overline{DMAAK3}$	Out	DMA acknowledge
DMARQ0-DMARQ3	In	DMA request
$\overline{DSR}$	In	Data set ready
$\overline{DSTB}$	Out	Data strobe
$\overline{DTR}$	Out	Data terminal ready
$\overline{END}$/$\overline{TC}$	I/O	DMA service forced-end input; DMA service complete output
HLDAK	Out	Bus hold acknowledge
HLDRQ	In	Bus hold request
$\overline{INTAK}$	Out	Interrupt acknowledge
INTP0-INTP7	In	Maskable interrupt request
$\overline{IORD}$	Out	I/O read
$\overline{IOWR}$	Out	I/O write
M/$\overline{IO}$	Out	Memory I/O select
$\overline{MRD}$	Out	Memory read
$\overline{MWR}$	Out	Memory write
NMI	In	Nonmaskable interrupt request
PCLKOUT	Out	External I/O clock
$\overline{READY}$	In	Bus cycle end
$\overline{REFRQ}$	Out	Refresh request
$\overline{RESET}$	In	Reset
RESOUT	Out	System reset
$\overline{RTS}$	Out	Request to send
R/$\overline{W}$	Out	Read/write
RxD	In	Serial receive data
RxRDY	Out	Serial receive ready
SINT	Out	Serial interrupt request
TCLK	In	Timer clock
TCTL0-TCTL2	In	Timer control
TOUT0-TOUT2	Out	Timer output
TxD	Out	Serial transmit data
$\overline{UBE}$	Out	Data bus higher byte enable
X1, X2	In	Crystal/external clock
V_{DD}	In	+5-volt power source
GND		Ground
IC		Internal connection
NC		No connection

Table 1. Output Pin States

Symbol	Hold	Halt	Reset	DMA Cascade
A_0-A_{23}	Hi-Z	L	Hi-Z	Hi-Z
AEX	Note 6	Note 6	H/L	Note 6
$\overline{BCYST}$	Hi-Z	Note 4	Hi-Z	Hi-Z
$\overline{BUFEN}$	Hi-Z	H	Hi-Z	Hi-Z
$\overline{BUSLOCK}$	Note 5	Note 5	H	H
BUSST0-BUSST2	Hi-Z	H	Hi-Z	H
CLKOUT	O	O	O	O
D_0-D_{15}	Hi-Z	Note 3	Hi-Z	Hi-Z
$\overline{DMAAK0}$-$\overline{DMAAK3}$	H	O	H	O
$\overline{DSTB}$	Hi-Z	H	Hi-Z	Hi-Z
$\overline{DTR}$	O	O	H	O
$\overline{END}$/$\overline{TC}$	Hi-Z	O	Hi-Z	O
HLDAK	H	H/L	L	L
$\overline{INTAK}$	H	H	H	H
$\overline{IORD}$	Hi-Z	H	Hi-Z	Hi-Z
$\overline{IOWR}$	Hi-Z	H	Hi-Z	Hi-Z
M/$\overline{IO}$	Hi-Z	L	Hi-Z	H
$\overline{MRD}$	Hi-Z	H	Hi-Z	Hi-Z
$\overline{MWR}$	Hi-Z	H	Hi-Z	Hi-Z
PCLKOUT	O	O	O	O
REFRQ	H	O	H	H
RESOUT	L	L	H	L
$\overline{RTS}$	O	O	H	O
R/$\overline{W}$	Hi-Z	L	Hi-Z	H
RxRDY	O	O	H	O
SINT	O	O	L	O

Table 1. Output Pin States (cont)

Symbol	Hold	Halt	Reset	DMA Cascade
TOUT0-TOUT2	O	O	O	O
TxD	O	O	H	O
$\overline{UBE}$	Hi-Z	H	Hi-Z	Hi-Z

Notes:

(1) The pin states are interpreted as follows: H is high level; L is low level; H/L is high or low level; Hi-Z is high impedance; O is indeterminate.

(2) Halt includes both the HALT and STOP modes.

(3) Undefined for the first two clocks of the halt acknowledge cycle and the Hi-Z.

(4) L for the first clock of the halt acknowledge cycle and then H.

(5) L under either of the following conditions: an instruction is executed during hold with a BUSLOCK prefix, or the HALT instruction is executed with a BUSLOCK prefix. Otherwise, the value is H.

(6) H in address expansion mode; L in nonexpansion mode.

PIN FUNCTIONS

A_0-A_{23} (Address Bus)

These pins constitute an address bus that outputs real addresses when memory or an I/O device is accessed. Up to 64K bytes of I/O space and up to 16M bytes of memory space (including reserved areas) can be accessed through the address bus.

The address bus enters the high-impedance state if one of the following occurs.

- $\overline{RESET}$ signal is applied
- Microprocessor is in HOLD mode
- DMA requests are cascade connected

The status of the address bus is undefined during an interrupt acknowledge cycle. When interrupt requests are cascade connected, the slave ICU address is output on pins A_0-A_2.

When I/O is accessed, pins A_{16}-A_{23} go low. The address can be expanded even when the interrupt vector table is accessed.

AEX (Address Extension)

AEX is asserted when the expanded addressing mode is enabled. When AEX is high, the memory address space is 16M bytes (24-bit address), and when low, 1M byte (20-bit address).

$\overline{BCYST}$ (Bus Cycle Start Strobe)

This signal indicates the start of a bus cycle by going low for one clock immediately after the bus cycle is started. When the bus is placed in the hold state, the $\overline{BCYST}$ pin enters the high-impedance state.

$\overline{BS8}$/BS16 (8-Bit Bus Size/16-Bit Bus Size)

$\overline{BS8}$/BS16 is driven low by external logic when the μPD70236 addresses a device with an 8-bit data path. If the μPD70236 operand is 16 bits wide and $\overline{BS8}$/BS16 is low, then the μPD70236 will perform two 8-bit bus cycles. The current bus cycle will handle the low byte on D_0-D_7, and the next bus cycle will handle the upper byte also on D_0-D_7. This input is ignored during HLDAK, interrupt acknowledge, and coprocessor cycles.

$\overline{BS8}$/BS16 is sampled on the rising (middle) edge of T2 or the last TW state, coincident with $\overline{READY}$. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

3f

$\overline{BUFEN}$ (Buffer Enable)

This signal is output to enable an external buffer, and becomes active during the read cycle, interrupt acknowledge cycle, and write cycle. It does not become active while the internal I/O is being accessed.

$\overline{BUSLOCK}$ (Bus Lock)

$\overline{BUSLOCK}$ should be used by external logic to exclude any other bus master (e.g., a DMA controller) from using a shared resource that the μPD70236 currently is using. When $\overline{BUSLOCK}$ is asserted high, HLDRQ will be ignored.

$\overline{BUSLOCK}$ is asserted when the $\overline{BUSLOCK}$ prefix is executed or when the μPD70236 is performing a bus operation that must not be interfered with, such as an interrupt acknowledge cycle. $\overline{BUSLOCK}$ has the same timing as the address bus A_0-A_{23} and is driven high during HLDAK and RESET.

BUSST0-BUSST2 (Bus Status)

These three pins encode and output information identifying the type of bus cycle currently being executed. They enter the high-impedance state in the bus hold mode. These pins are used with the M/$\overline{IO}$ and R/$\overline{W}$ signals, as shown in table 2.

Table 2. Bus Cycles

M/$\overline{IO}$	R/$\overline{W}$	BUSST2	BUSST1	BUSST0	Bus Cycle
0	1	0	0	0	Interrupt acknowledge cycle (from SLAVE)
0	1	1	0	0	Interrupt acknowledge cycle (from ICU)
0	1	0	0	1	External I/O read cycle
0	1	1	0	1	Internal I/O read cycle
0	0	0	0	1	External I/O write cycle
0	0	1	0	1	Internal I/O write cycle
0	1	0	1	0	Coprocessor read cycle
0	0	0	1	0	Coprocessor write cycle
0	0	0	1	1	Halt acknowledge cycle
1	1	0	0	0	Instruction fetch cycle
1	1	1	0	0	Refresh cycle
1	1	0	0	1	CPU memory read cycle
1	1	1	0	1	DMA read transfer cycle
1	0	0	0	1	CPU memory write cycle
1	0	1	0	1	DMA write transfer cycle
1	1	0	1	0	Coprocessor memory read cycle
1	0	0	1	0	Coprocessor memory write cycle
1	1	1	1	1	DMA cascade

Interrupt Acknowledge Cycle (from SLAVE). This cycle is the second interrupt acknowledge cycle during which an interrupt request from a slave interrupt control unit (ICU) is acknowledged. During this cycle, the data output by an external interrupt controller is processed as a vector. The bus sizing function cannot be effected in this cycle. The programmable wait function and $\overline{READY}$ signals are both valid, however.

Interrupt Acknowledge Cycle (from ICU). This cycle is output during the first interrupt acknowledge cycle, during which an interrupt request for a non-slave ICU is acknowledged. During this acknowledge cycle, the data output by the internal ICU is processed as a vector, and the bus sizing function cannot be effected. The programmable wait function and $\overline{READY}$ signal are both valid, however.

External I/O Read Cycle. This cycle is output when an external I/O area is read by executing the IN instruction. During this cycle, the bus sizing function can be effected. Also, the programmable wait function and $\overline{READY}$ signal are both valid.

Internal I/O Read Cycle. This cycle is output when the internal I/O area is read by executing the IN instruction. The bus sizing function cannot be effected. Both the programmable wait function and $\overline{READY}$ signal are invalid. However, two wait state clocks are automatically inserted into all internal I/O area cycles except those for the address expansion table and address expansion flag.

External I/O Write Cycle. This cycle is output when an external I/O area is written by executing the OUT instruction. The bus sizing function can be effected. Also, the programmable wait function and $\overline{READY}$ signal are both valid.

Internal I/O Write Cycle. This is output when the internal I/O area is written by executing the OUT instruction. The bus sizing function cannot be effected. Both the programmable wait function and $\overline{READY}$ signal are invalid. However, two wait state clocks are automatically inserted into all internal I/O area cycles except those for the address expansion table and address expansion flag.

Coprocessor Read Cycle. This cycle indicates that an external coprocessor is accessed for data read when a coprocessor instruction is executed. The bus timing and ac characteristics of this cycle are the same as those of the ordinary I/O read cycle.

Although the bus sizing function cannot be effected, coprocessor operations are not guaranteed if the bus sizing function is used. The programmable wait function is invalid, but the $\overline{READY}$ signal is valid.

Coprocessor Write Cycle. This cycle indicates that an external coprocessor instruction is executed. The bus timing and ac characteristics of this cycle are the same as those of the ordinary I/O write cycle.

Although the bus sizing function can be effected, coprocessor operations are not guaranteed if the bus sizing function is used. The programmable wait function is invalid, but the $\overline{READY}$ signal is valid.

Halt Acknowledge Cycle. This cycle is output when the HALT instruction is executed. During this bus cycle, the $\overline{DSTB}$ pin does not output a low level. The bus sizing function cannot be effected. Both the programmable wait function and $\overline{READY}$ signal are invalid.

Instruction Fetch Cycle. This cycle indicates that an instruction is being fetched. The bus sizing function can be effected. Also, the programmable wait function and $\overline{READY}$ signal are both valid.

Refresh Cycle. This cycle indicates that DRAM refreshing is in progress. The bus sizing function cannot be effected. (Note that $\overline{BS8}$/BS16 must be 16 bits.) The programmable wait function and $\overline{READY}$ signal are both valid.

CPU Memory Read Cycle. This cycle is output when the CPU reads data from memory. The bus sizing function

can be effected. Also, the programmable wait function and $\overline{READY}$ signal are both valid.

DMA Read Transfer Cycle. This cycle is output when DMA transfer (that is, data transfer from memory to I/O) takes place. The bus sizing function cannot be effected. The programmable wait function and $\overline{READY}$ signal are both valid.

CPU Memory Write Cycle. This cycle is output when the CPU writes data to memory. The bus sizing function can be effected. Also, the programmable wait function and $\overline{READY}$ signal are both valid.

DMA Write Transfer Cycle. This cycle is output when write DMA transfer (that is, data transfer from I/O to memory) takes place. The bus sizing function cannot be effected. The programmable wait function and $\overline{READY}$ signal are both valid.

Coprocessor Memory Read Cycle. This cycle is output when data read from memory is sent to the coprocessor. Although the bus sizing function cannot be effected, coprocessor operations are not guaranteed if bus sizing is used. The programmable wait function and $\overline{READY}$ signal are both valid.

Coprocessor Memory Write Cycle. This cycle is output when data for a coprocessor is written to memory. The CPU does not drive the data bus. Instead, the coprocessor drives the data bus to write data to memory.

Although the bus sizing function cannot be effected, coprocessor operations are not guaranteed if the bus sizing function is used. The programmable wait function and $\overline{READY}$ signal are both valid.

DMA Cascade. This cycle indicates that the DMA is cascade connected to an external slave DMA controller. During this cycle, the buses are relinquished.

CLKOUT (Clock Output)

This pin outputs a square-wave clock pulse. The frequency of the output clock pulse is obtained by dividing the frequency of the clock signal input to the X1 and X2 pins by a specific value. The duty factor of the output clock pulse is 50%. The output frequency is the same as the operating frequency of the CPU (programmable to one-half, one-fourth, one-eighth, or one-sixteenth of the oscillation frequency).

$\overline{CPBUSY}$ (Coprocessor Busy)

$\overline{CPBUSY}$ is asserted low by a coprocessor (such as μPD72291) when it is busy with an internal operation. The μPD70236 uses this pin to check the status of the coprocessor.

$\overline{CPBUSY}$ is sampled on the falling edge of each clock. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

If a coprocessor is not connected to the μPD70236, $\overline{CPBUSY}$ should be grounded.

$\overline{CTS}$ (Clear to Send)

This is a serial transmission control input pin. The SCU is ready for data transmission when bit 0 of the SCM register is set to 1 and this pin is at low level. When this pin is made high while data transmission is in progress, transmission is stopped after the current data has been completely transmitted, and the TxD pin goes high.

D_0-D_{15} (Data Bus)

These pins constitute a data bus that inputs or outputs write data and read data when the external main memory or I/O device is accessed. The data bus is in the input mode during any bus cycle other than a write cycle. During the write bus cycle, the bus outputs data starting from the rising edge of the T1 clock until the cycle following the write bus end cycle.

$\overline{DMAAK0}$-$\overline{DMAAK3}$ (DMA Acknowledge)

These pins output active-low DMA acknowledge signals from channels 0 to 3 of the internal DMAU.

DMARQ0-DMARQ3 (DMA Request)

These pins input active-high DMA request signals from channels 0 to 3 of the internal DMA control unit (DMAU).

$\overline{DSR}$ (Data Set Ready)

This is a general-purpose input pin. The status of this pin can be determined by reading bit 7 of the serial status (SST) register.

$\overline{DSTB}$ (Data Strobe)

This is a strobe signal for read and write operations. The signal does not go low during the halt acknowledge cycle that indicates that the HALT instruction has been executed. When the buses are placed in the hold state, the $\overline{DSTB}$ pin enters the high-impedance state. The signal output timing of this pin differs depending on whether a read or write operation is performed. The $\overline{DSTB}$ signal does not go low when the internal I/O area is accessed.

$\overline{DTR}$ (Data Terminal Ready)

This is a general-purpose output pin. The status of this pin can be set by bit 1 of the SCM register.

END/$\overline{TC}$ (End/Terminal Count)

This pin inputs the END signal to or outputs the TC signal from the internal DMAU.

$\overline{END}$ Input. When a low-level pulse is input to this pin during DMA transfer, the DMA service under execution is terminated after the current bus cycle is over.

$\overline{TC}$ Output. When the count register of the DMAU channel currently performing DMA transfer becomes 0, and when the DMA transfer has been performed the specified number of times, the TC pin outputs a low-level pulse.

HLDAK (Hold Acknowledge)

This is an acknowledge signal that indicates that the V53 has accepted the HLDRQ signal, placed the address, data, and control buses in the high-impedance state, and relinquished the buses to an external device. The external devices that can acquire the buses are assigned the following priority.

REFU (highest priority)
DMAU
HLDRQ
CPU
REFU

If a bus hold request takes place while the buses are idle (TI state), during the CPU bus cycle, or during lowest-priority refresh cycle, the HLDRQ signal is accepted immediately after the bus cycle is over and the buses are relinquished.

If a DMA request or top-priority refresh request is generated while the buses are in the hold state, the HLDAK signal is forcibly made inactive. In this case, the external device must return control of the bus to the V53 (making the HLDRQ signal inactive). Therefore, the high-level width of the HLDAK signal when it is made inactive forcibly is 1 clock minimum.

HLDRQ (Hold Request)

HLDRQ is asserted high by external logic when an external bus master (e.g., a DMA controller) wants to take over the μPD70236 bus. When HLDRQ is detected high, the μPD70236 will release the bus after the current bus operation is completed. Note that this is not necessarily the current bus cycle. The μPD70236 releases its bus by floating the address, data, and control buses.

HLDRQ is sampled on the rising edge of each clock. It will be ignored while $\overline{BUSLOCK}$ is asserted. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

INTAK (Interrupt Acknowledge)

This is an active-low acknowledge signal for a maskable interrupt.

INTP0-INTP7 (Interrupt from Peripherals)

These are asynchronous interrupt request input pins for the internal interrupt control unit (ICU). The input signals can be triggered either at the rising edge or at high level. The priority of these signals can be fixed or rotated. These interrupt request inputs are also used to release the HALT and STOP modes.

$\overline{IORD}$ (I/O Read)

This active-low read signal goes low during the I/O read cycle. This signal is also output when write DMA transfer is performed. However, it is not output during the CPU's internal I/O read cycle.

$\overline{IOWR}$ (I/O Write)

This is an active-low write signal that goes low during the I/O write cycle. This signal is also output when read DMA transfer is performed in two output timing modes: the expansion write mode and the ordinary write mode. It is not output during the CPU's internal I/O write cycle.

M/$\overline{IO}$ (Memory I/O)

This pin indicates whether a memory or other device (such as an I/O device or coprocessor) is currently accessed. The device to be accessed is determined by this pin and the BUSST0 and BUSST1 signals. The M/$\overline{IO}$ pin enters the high-impedance state in the bus hold mode. Its status changes at the falling edge of the T1 clock.

$\overline{MRD}$ (Memory Read)

This is an active-low read signal that goes low during a read cycle in which data is read from memory. This signal is output not only during the CPU's memory read, but also during the refresh cycle and when read DMA transfer is performed.

$\overline{MWR}$ (Memory Write)

This active-low write signal goes low when the memory write cycle is in progress. This signal is output not only during the CPU's memory write cycle, but also during the write DMA transfer and when write DMA transfer is performed in two output timing modes: the expansion write mode and the ordinary write mode.

$\overline{NMI}$ (Nonmaskable Interrupt Request)

$\overline{NMI}$ is asserted by external logic to notify the CPU that an external event requires the CPU's immediate attention. When $\overline{NMI}$ is sampled low, interrupt processing will begin immediately after the current instruction is completed. A trap will be taken through vector 2. The state of the IE bit in the PSW has no effect on $\overline{NMI}$ acceptance.

$\overline{NMI}$ is sampled on the falling edge of each CPU clock. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

Interrupt processing begins immediately after the end of the current instruction. Once NMI processing comences, no further NMI requests will be accepted until termination of the current NMI routine, which is indicated by the RETI instruction.

PCLKOUT (Peripheral Clock Output)

This pin outputs a square-wave clock pulse with a frequency one-fourth the frequency of the clock signal input to the X1 and X2 pins. The duty factor of the output clock pulse is 50%.

$\overline{READY}$ (System Ready)

The $\overline{READY}$ signal is asserted low when the external system is ready for the current bus cycle to terminate. While $\overline{READY}$ is not asserted, the μPD70236 will add TW (wait) states to the current bus cycle. The bus state in which $\overline{READY}$ is sampled low will be the last state of the cycle.

During CPU read cycles, $\overline{READY}$ gives slow devices time to drive the D_0-D_7 inputs, and during write cycles gives slow devices enough time to finish the write operation.

The $\overline{READY}$ input is sampled on the rising (middle) edge of T2 and all TW states. It is ignored during the HLDAK state. This input is not internally synchronized. To ensure proper device operation, minimum setup and hold times must be met.

$\overline{REFRQ}$ (Refresh Request)

This signal is asserted during refresh cycles.

$\overline{RESET}$ (Reset)

This signal initializes the processor. The processor is reset when this signal is held low for six clocks or longer and then returned to the high level.

RESOUT (Reset Output)

This pin outputs an active-high signal which is an asynchronous $\overline{RESET}$ signal synchronized with the internal clock. This signal can be used to reset the system.

$\overline{RTS}$ (Request to Send)

This is a general-purpose output pin. The status of this pin can be set by bit 5 of the serial command (SCM) register.

R/$\overline{W}$ (Read/Write)

This pin indicates whether the current bus cycle is a read cycle or a write cycle. This pin is valid only while a bus cycle is being executed, and goes high if the current bus cycle is a read cycle or during an interrupt acknowledge cycle; it goes low if the current bus cycle is a write cycle. The R/$\overline{W}$ pin enters the high-impedance state in the bus hold mode. The level of this pin changes at the falling edge of the T1 clock.

RxD (Receive Data)

When the serial control unit does not receive data, this pin is at high level (mark state). When the pin detects a start bit, the SCU starts receiving serial data from an external device.

RxRDY (Receive Ready)

When the serial control unit has received one character of data, and when that data is transferred to the receive data buffer (that is, when the receive data is ready to be read), this pin goes high.

SINT (Serial Interrupt)

This signal becomes active to output an interrupt request signal from the SCU when the transmit data buffer of the SCU is empty and when the interrupt of the transmitting side is not masked, or when it contains the SCU's receive buffer data to be read and the receive interrupt is not masked.

TCLK (Timer Clock)

This pin inputs a clock pulse from an external source to the internal timer/counter unit (TCU). When the system is initialized, either the external clock or the internal clock is selected to be supplied to the TCU.

TCTL0-TCTL2 (Timer Control)

These pins input control signals to the three TCU counters. The functions of the control signals input

through these pins differ depending on the mode (six modes are available) set by the TCU.

TOUT0-TOUT2 (Timer Output)

These are output pins for the internal timer/counter unit. The TCU outputs signals through these pins in six different modes.

TxD (Transmit Data)

When the serial control unit (SCU) has no data to be transmitted to an external device, this pin is at high level (mark state). When transmit data is set in the SCU, the TxD pin automatically outputs a start bit, serial data that has been set in the SCU, a parity bit, and 1 or 2 stop bits.

$\overline{UBE}$ (Upper Byte Enable)

When the microprocessor accesses external main memory or an I/O device that requires the upper 8 bits (D_8-D_{15}) of the data bus, this pin goes low at the falling edge of the T1 clock, enabling the upper byte on the bus. The lower 8 bits (D_0-D_7) of the data bus are controlled by the A_0 pin as shown in the following table.

$\overline{UBE}$	A_0	Operation
0	0	16 bits accessed
0	1	Upper 8 bits accessed
1	0	Lower 8 bits accessed
1	1	Second cycle (for use with bus sizing function)

When dynamic bus sizing is used to make a 16-bit access into an 8-bit, A_0 must be used as an address bit; $\overline{UBE}$ can be ignored.

X1, X2 (Crystal)

To use the internal clock generator, connect a crystal with a frequency twice the operating frequency across these pins. When using an external clock generator, input square waves with a frequency twice the operating frequency to the X1 pin. To the X2 pin, make the input signal 180° out of phase (an inverter output) with the signal input to the X1 pin.

UNIT OPERATION

Central Processing Unit (CPU)

The μPD70236 CPU is a high-performance engine whose performance surpasses most other 16-bit CPUs. To achieve this performance level, hardwired data path control was used (no microcode) so that instruction execution times are greatly reduced.

The μPD70236 CPU has functions equivalent to those of the μPD70136 (V33) and is therefore completely software compatible with the V33. The μPD70236 instruction set is upward compatible with the native modes of the V20, V30, V40, and V50.

Clock Generator (CG)

The clock generator divides the oscillation frequency of the crystal or external oscillator connected across pins X1 and X2 by 2, 4, 8, or 16 to generate a clock that is supplied to the CPU as an operation clock and to an external device through the CLKOUT pin. A clock having a frequency one-fourth the oscillation frequency is also output to the PCLKOUT pin.

Bus Interface Unit (BIU)

The bus interface unit controls the pins of the address bus, data bus, and control bus, which are used by the CPU, DMA unit (DMAU), and refresh control unit (REFU).

Bus Arbitration Unit (BAU)

The bus arbitration unit arbitrates the internal bus mastership. The priority of the bus mastership is:

CPU with $\overline{BUSLOCK}$ (highest priority)
REFU of top priority
DMAU
HLDRQ
Ordinary CPU
REFU of lowest priority

Wait Control Unit (WCU)

The function of the wait control unit is to insert wait states equivalent to 0 to 7 clocks automatically into the memory, I/O, DMA, and refresh cycles. The 16M-byte memory space can be divided into three blocks. In addition, any 1M-byte memory space can also be divided into three blocks.

Refresh Control Unit (REFU)

The REFU supports the DRAM refresh operation by generating 16-bit refresh addresses and a refresh signal ($\overline{REFRQ}$) indicating that the refresh cycle is currently taking place.

Timer/Counter Unit (TCU)

The timer/counter unit of the μPD70236 performs the same functions as the μPD71054. It provides a set of three independent 16-bit timer/counters.

Serial Control Unit (SCU)

The μPD70236 SCU has the same functions as the μPD71051 except the synchronous mode for supporting RS-232C protocol. This SCU is equipped with a dedicated baud rate generator.

The SCU provides serial communications functions of the start-stop synchronization type. Commands for the SCU in the V53 are similar to those of the μPD71051 except that the V53 uses two registers—SCM (serial command) register and SMD (serial mode) register—to implement the functions of the control word register of the μPD71051.

Interrupt Control Unit (ICU)

The ICU in the V53 has the same functions as those on the μPD71059 except the V53 does not have the CALL mode (8085 mode) or the slave mode of cascade connection. The μPD70236 ICU has eight external interrupt input pins and can arbitrate up to eight interrupt requests. The number of external interrupt inputs can be increased by cascade connecting the ICU to an external interrupt controller.

Unlike the μPD71059, μPD70208, and μPD70216, the INTP0 to INTP7 pins in the V53 do not have internal pullup resistors to reduce current dissipation.

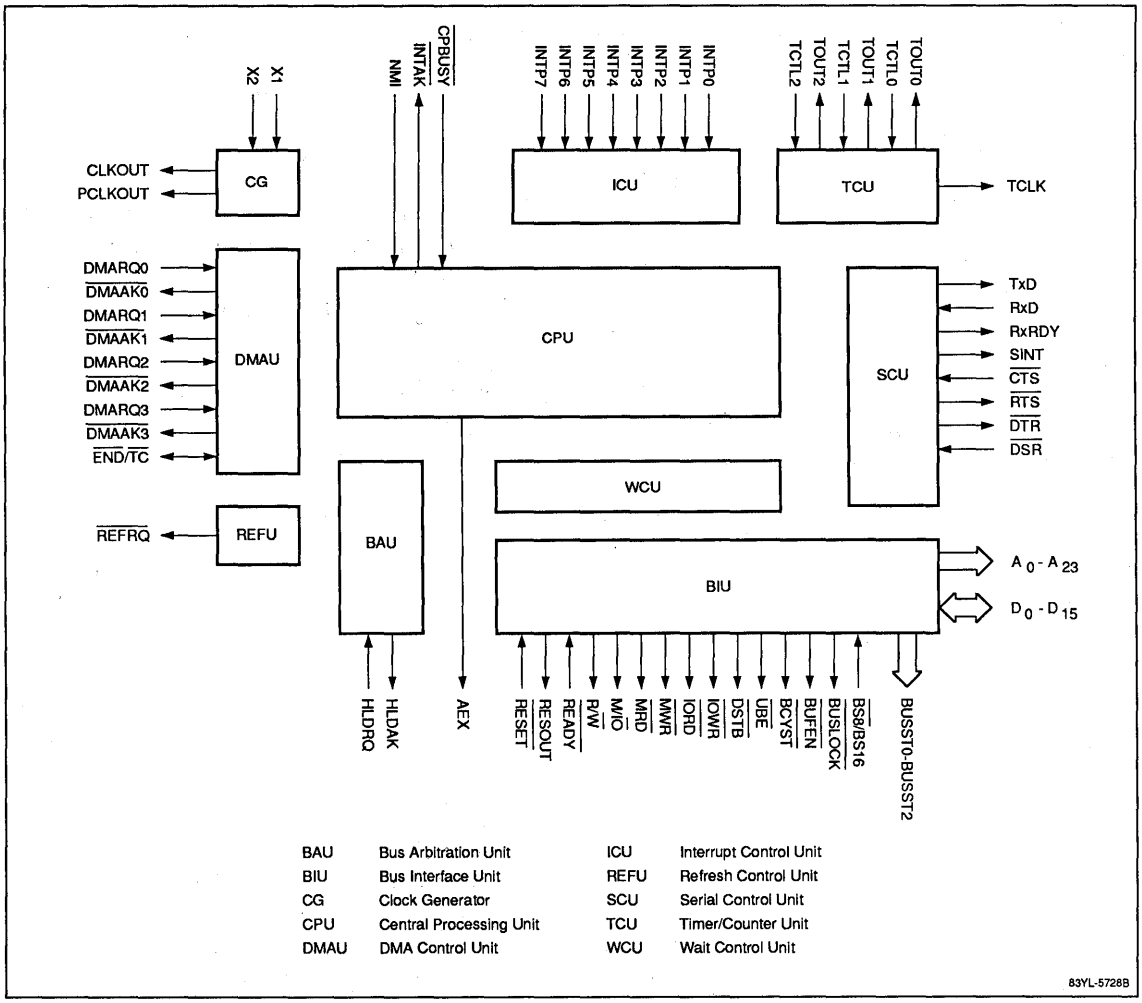
DMA Control Unit (DMAU)

The DMAU on the μPD70236 functions the same as the DMAUs on the μPD71071 and μPD71037 and, therefore, it can operate in two modes (μPD71071 mode and μPD71037 mode). You can set the operation modes using a register in the system I/O area.

In μPD71071 mode, source and destination addresses are 24 bits. In μPD71037 mode, source and destination addresses are 16 bits. To extend these addresses to 20 or 24 bits, four 8-bit bank registers are provided. These registers supply the upper address bits.

The DMA unit provides four channels of μPD71071-compatible or μPD71037-compatible DMA. External hardware requests DMA cycles via the DMA request inputs. DMA is always between an I/O device and memory (fly-by style DMA). External DMA controllers may be cascaded using the V53 DMAU.

μPD70236 Block Diagram



BAU	Bus Arbitration Unit	ICU	Interrupt Control Unit
BIU	Bus Interface Unit	REFU	Refresh Control Unit
CG	Clock Generator	SCU	Serial Control Unit
CPU	Central Processing Unit	TCU	Timer/Counter Unit
DMAU	DMA Control Unit	WCU	Wait Control Unit

83YL-5728B

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = +25°C

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.3 V
Clock input voltage, V_K	−0.5 to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 to V_{DD} + 0.3 V
Output short circuit current, I_O	50 mA
Operating temperature, T_{OPT}	−10 to +70°C
Storage temperature, T_{STG}	−65 to +150°C

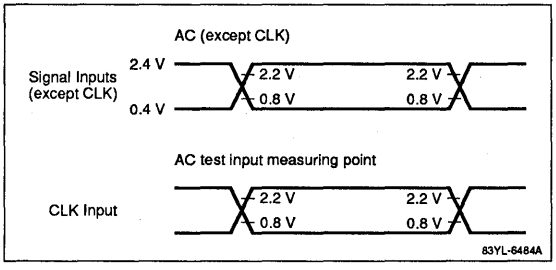
DC Characteristics

T_A = −10 to +70°C; V_{DD} = +5 V 10%

Parameter	Symbol	Min	Max	Unit	Conditions
Input voltage, high	V_{IH}	2.2	V_{DD} + 0.3	V	Except $\overline{RESET}$
		0.8 V_{DD}		V	$\overline{RESET}$
Input voltage, low	V_{IL}	−0.5	0.8	V	Except $\overline{RESET}$
			0.2 V_{DD}	V	$\overline{RESET}$
Clock input voltage, high	V_{KH}	0.8 V_{DD}	V_{DD} + 0.5	V	
Clock input voltage, low	V_{KL}	−0.5	0.6	V	
Output voltage,high	V_{OH}	0.7 V_{DD}		V	I_{OH} = −400 μA
Output voltage, low	V_{OL}		0.45	V	I_{OL} = 2.5 mA
Input leakage current, high	I_{LIH}		10	μA	V_I = V_{DD}
Input leakage current, low	I_{LIL}		−10	μA	V_I = 0 V
Output leakage current, high	I_{LOH}		10	μA	V_O = V_{DD}
Output leakage current, low	I_{LOL}		−10	μA	V_O = 0 V
Supply current	I_{DD}		10 f + 40	mA	Operating; f = 2 to 16 MHz
			40	mA	HALT mode
			200	μA	STOP mode

3f

Voltage Thresholds for Timing Measurements



AC Characteristics

T_A = −10 to +70°C; V_{DD} = 5 V ±10%; C_L of output terminals = 100 pF max

Parameter	Symbol	Min	Max	Unit
Clocks (figure 1)				
CLKOUT period	t_{CYK}	62.5	500	ns
CLKOUT high-level width	t_{KKH}	0.5 t_{CYK} − 7		ns
CLKOUT low-level width	t_{KKL}	0.5 t_{CYK} − 7		ns
CLKOUT rise time	t_{KR}		7	ns
CLKOUT fall time	t_{KF}		7	ns
X1 input period	t_{CYX}	31.25	250	ns
X1 input high-level width	t_{XKH}	11		ns
X1 input low-level width	t_{XKL}	11		ns
X1 input rise time	t_{XKR}		5	ns
X1 input fall time	t_{XKF}		5	ns
X1 to CLKOUT delay	t_{DXK}		20	ns
PCLKOUT period	t_{CYPK}	125	1000	ns
PCLKOUT high-level width	t_{PKH}	4 t_{CYK} − 7		ns
PCLKOUT low-level width	t_{PKL}	4 t_{CYK} − 7		ns
PCLKOUT rise time	t_{PKR}		7	ns
PCLKOUT fall time	t_{PKF}		7	ns
Reset (figure 2)				
$\overline{\text{RESET}}$ setup time vs CLKOUT ↓	t_{SRSTK}	30		ns
$\overline{\text{RESET}}$ hold time vs CLKOUT ↓	t_{HKRST}	15		ns
$\overline{\text{RESET}}$ low-level width	t_{WRSTL}	6		t_{CYC}
RESOUT delay from CLKOUT ↓	t_{DKRO}	0	40	ns
Write, Read (figures 3-12, 16-19, 23-24, 28, 31) Note 2				
$\overline{\text{BCYST}}$ delay from CLKOUT ↓	t_{DKBC}	5	40	ns
$\overline{\text{BCYST}}$ low-level width	t_{BCBCL}	t_{CYK} − 10		ns
$\overline{\text{BCYST}}$ high-level width	t_{BCBCH}	t_{CYK} (n+1) − 10		ns
Address delay from CLKOUT ↓	t_{DKA}	5	40	ns
Control 2 delay from CLKOUT	t_{DKCT2}	0	40	ns
Status delay from CLKOUT ↓	t_{DKST}	5	40	ns

Parameter	Symbol	Min	Max	Unit
Data float delay from CLKOUT	t_{FK}	0	50	ns
DSTB ↓ delay from CLKOUT ↓	t_{DKDS}	5	40	ns
$\overline{\text{DSTB}}$ low-level width	t_{DSDSL}	t_{CYC} (n+1) − 10		ns
$\overline{\text{DSTB}}$ high-level width	t_{DSDSH}	t_{KKL} + t_{KR} − 10		ns
CLKOUT to $\overline{\text{IOWR}}$ delay	t_{DKIW}	0	40	ns
CLKOUT to $\overline{\text{IORD}}$ delay	t_{DKIR}	0	40	ns
CLKOUT to $\overline{\text{MRD}}$ delay	t_{DKMR}	0	40	ns
CLKOUT to $\overline{\text{MWR}}$ delay	t_{DKMW}	0	40	ns
CLKOUT ↑ to DSTB ↑	t_{DKDSH}	5	40	ns
Address/status output delay to DSTB ↓	t_{DADSL}	t_{KKL} + t_{KR} − 15		ns
Address/status hold time from $\overline{\text{DSTB}}$ ↑	t_{HDSHA}	t_{KKL} + t_{KR} − 15		ns
Data output delay from DSTB ↑	t_{DDSHD}	t_{KKL} + t_{KR} − 15		ns
Data output delay from address/status output	t_{DAD}	t_{KKL} + t_{KR} − 15		ns
Data output delay from CLKOUT ↑	t_{DKD}	5	40	ns
Data setup time to CLKOUT ↓	t_{SDK}	7		ns
Data hold time from CLKOUT ↓	t_{HKD}	10		ns
Data hold time from $\overline{\text{DSTB}}$ ↑	t_{HDSD}	0		ns
Data hold time from change point of address or status	t_{HASD}	0		ns
Dat hold time from R/W ↑	t_{HRWD}	0		ns
$\overline{\text{READY}}$ setup time to CLKOUT ↑	t_{SRYK}	7		ns
$\overline{\text{READY}}$ hold time from CLKOUT ↑	t_{HKRY}	15		ns

Notes:

(1) t_{CYC} = CPU clock period
n = number of wait states

(2) The clock-to-signal delays in the −10 (10 MHz) and −12 (12.5 MHz) parts are 45 ns compared to 40 ns in the −16 (16 MHz) part. For full electrical characteristics of the −10 and −12 parts, contact NEC.

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit
Bus Sizing (figures 13, 14)				
$\overline{BS8}$/BS16 setup time to CLKOUT ↑	t_{SBSK}	7		ns
$\overline{BS8}$/BS16 hold time from CLKOUT ↑	t_{HKBS}	15		ns
Bus Hold (figure 17)				
HLDRQ setup time to CLKOUT ↑	t_{SHQK}	7		ns
HLDRQ hold time form CLKOUT ↑	t_{HKHQ}	15		ns
CLKOUT ↑ to HLDAK delay	t_{DKHA}	5	40	ns
Output floating to HLDAK delay	t_{DFHA}	$t_{KKL} + t_{KR} - 15$		ns
Input Setup and Hold (figure 15)				
$\overline{NMI}$, INTP0-INTP7, $\overline{CPBUSY}$ setup time to CLKOUT ↓	t_{SIK}	10		ns
$\overline{NMI}$, INTP0-INTP7, $\overline{CPBUSY}$ hold time from CLKOUT ↓	t_{HKT}	10		ns
Timer/Counter Unit TCU (figures 20-21)				
TCTL0-TCTL2 setup time to CLKOUT ↓	t_{SGK}	50		ns
TCTL0-TCTL2 hold time from CLKOUT ↓	t_{HKG}	100		
TCTL0-TCTL2 low-level width	t_{GGL}	50		ns
TCTL0-TCTL2 high-level width	t_{GGH}	50		ns
TOUT0-TOUT2 output delay from CLKOUT ↓	t_{DKTO}		100	ns
TCLK period	t_{CYTK}	100		ns
TCLK rise time	t_{TKR}		15	ns
TCLK fall time	t_{TKF}		15	ns
TCLK low-level width	t_{TKTKL}	45		ns
TCLK high-level width	t_{TKTKH}	30		ns
TCTL0-TCTL2 setup time to TCLK ↑	t_{SGTK}	50		ns
TCTL0-TCTL2 hold time from TCLK ↑	t_{HTKG}	100		ns
TOUT0-TOUT2 output delay from TCLK ↓	t_{DTKTO}		100	ns
TOUT0-TOUT2 output delay from TCTL ↓	t_{DGTO}		100	ns

Parameter	Symbol	Min	Max	Unit
Serial Control Unit, SCU (figure 22)				
RxD setup time vs SCU internal CLK ↓	t_{SRX}	1		ns
RxD hold time vs SCU internal CLK ↓	t_{HRX}	1		ns
TOUT1 ↑ to TxD delay	t_{DTX}		500	ns
Direct Memory Access, DMA (figures 24-26)				
CLKOUT ↓ to $\overline{MRD}$, IORD ↑ delay	t_{DKRH}	0	40	ns
CLKOUT ↓ to $\overline{MRD}$, IORD ↓ delay	t_{DKRL}	0	40	ns
CLKOUT ↑ to DMAAK0-DMAAK3 delay	t_{DKHDA}	0	40	ns
$\overline{IORD}$ ↓, $\overline{IOWR}$ ↓ delay from DMAAK0-DMAAK3 ↓	t_{DDARW}	$t_{KKH} - 30$		ns
DMAAK0-DMAAK3 ↑ delay from $\overline{IORD}$ ↑	t_{DRHDAH}	$t_{KKH} - 30$		ns
CLKOUT to control 1 delay	t_{DKCT1}	0	40	ns
$\overline{IORD}$ ↑ delay to $\overline{IOWR}$ ↑	t_{DWHRH}	5	40	
$\overline{TC}$ output delay from CLKOUT ↑	t_{DKTCL}	0	40	ns
$\overline{TC}$ off output delay from CLKOUT ↑	t_{DKTCF}	0	40	ns
$\overline{TC}$ pullup delay from CLKOUT ↑	t_{DKTCH}	0	40	ns
$\overline{TC}$ low-level width	t_{TCTCL}	$t_{CYC} - 15$		ns
END setup time to CLKOUT ↑	t_{SEDK}	35		ns
END low-level width	t_{EDEDL}	100		ns
$\overline{IORD}$, $\overline{MRD}$ low-level width	t_{RR}	$2t_{CYC} - 40$		
$\overline{IOWR}$, $\overline{MWR}$ low-level width (Expanded write)	t_{WW1}	$2t_{CYC} - 40$		
$\overline{IOWR}$, $\overline{MWR}$ low-level width (Normal write)	t_{WW2}	$t_{CYC} - 40$		
DMARQ0-DMARQ3 setup time to CLKOUT ↑	t_{SDQK}	15		ns
CLKOUT ↓ to $\overline{DMAAK0}$-$\overline{DMAAK3}$ delay	t_{DKLDA}	0	45	ns
Interrupt Control Unit, ICU (figure 27)				
INTP0-INTP7 low-level width	t_{IPIPL}	100		ns

3f

Figure 1. Clock Timing

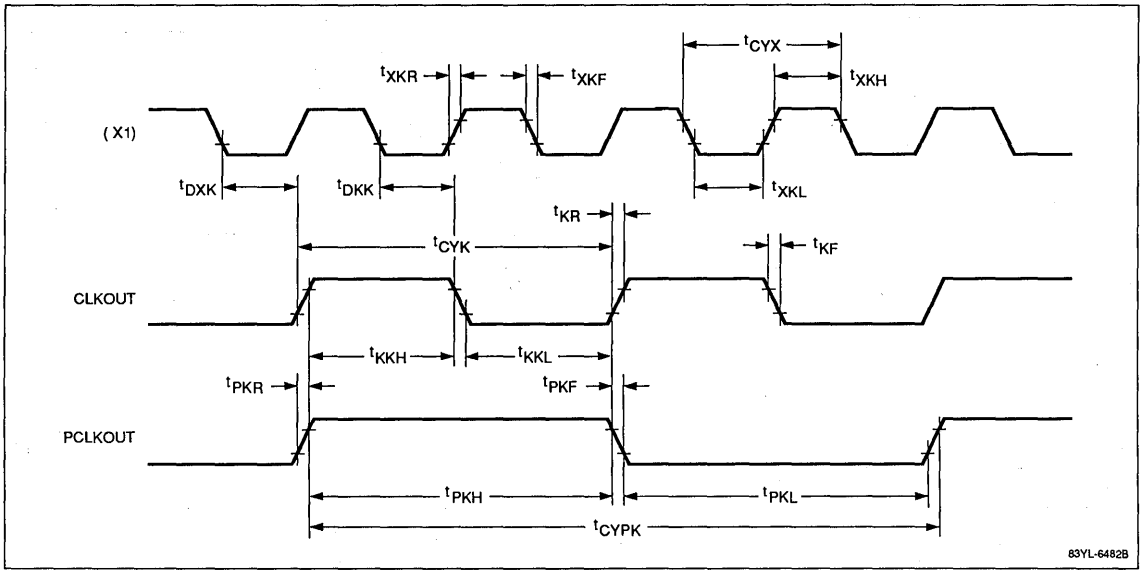


Figure 2. Reset Timing

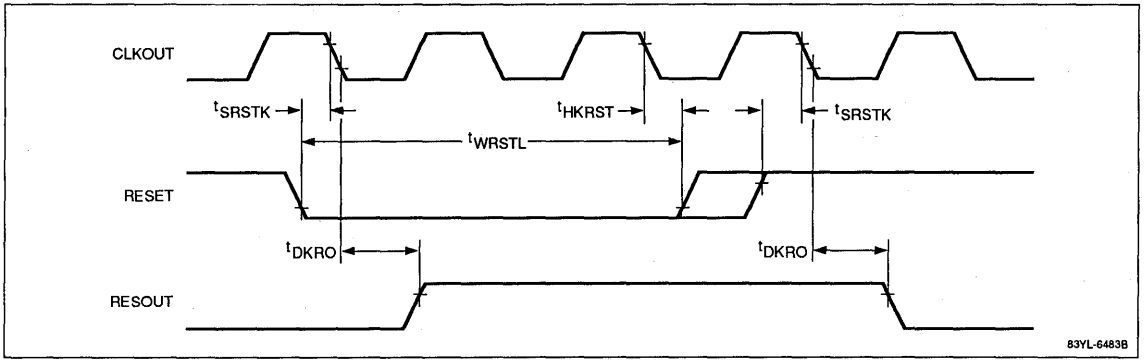


Figure 3. Basic Write (0 Wait)

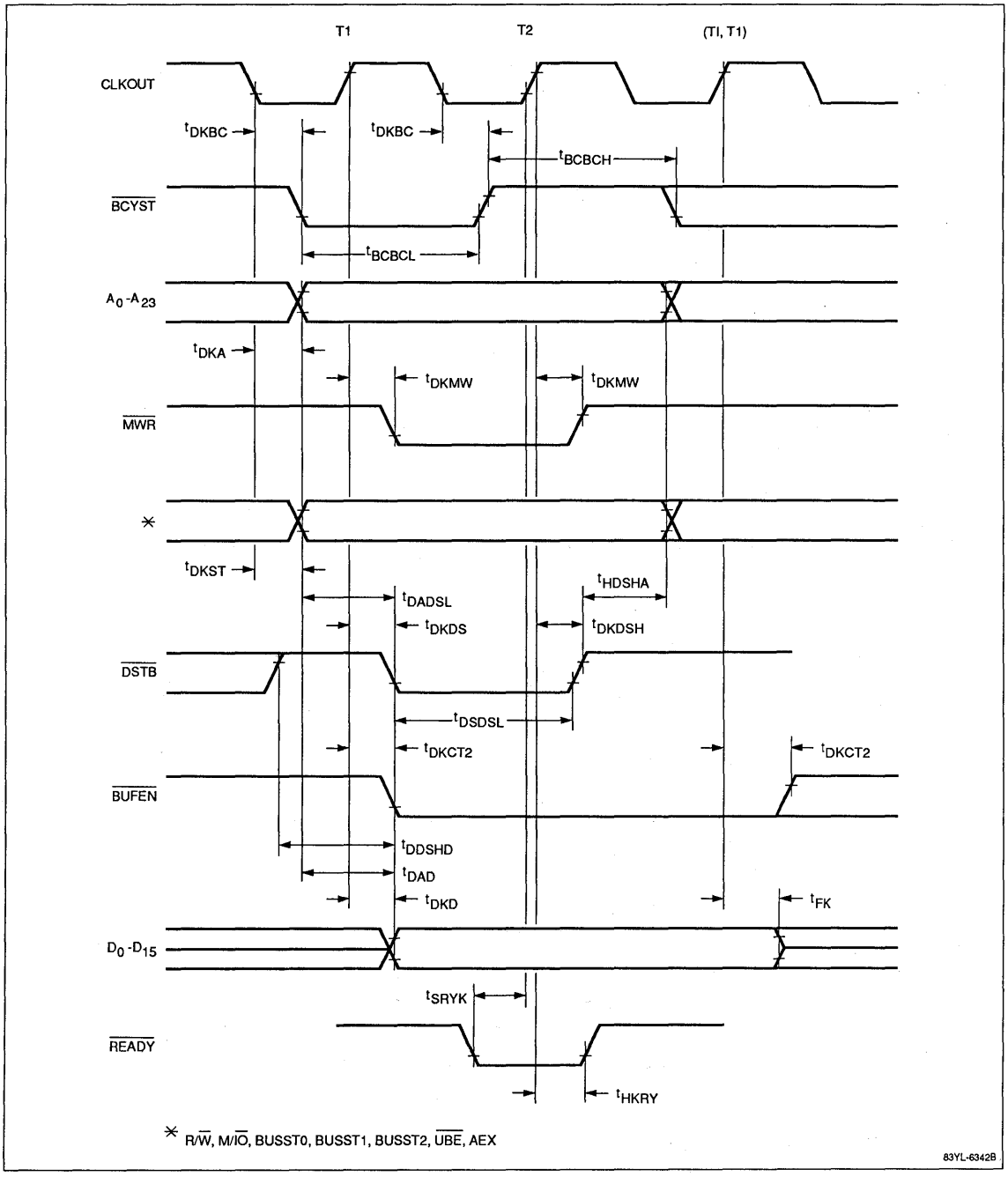


Figure 4. Basic Write (1 Wait)

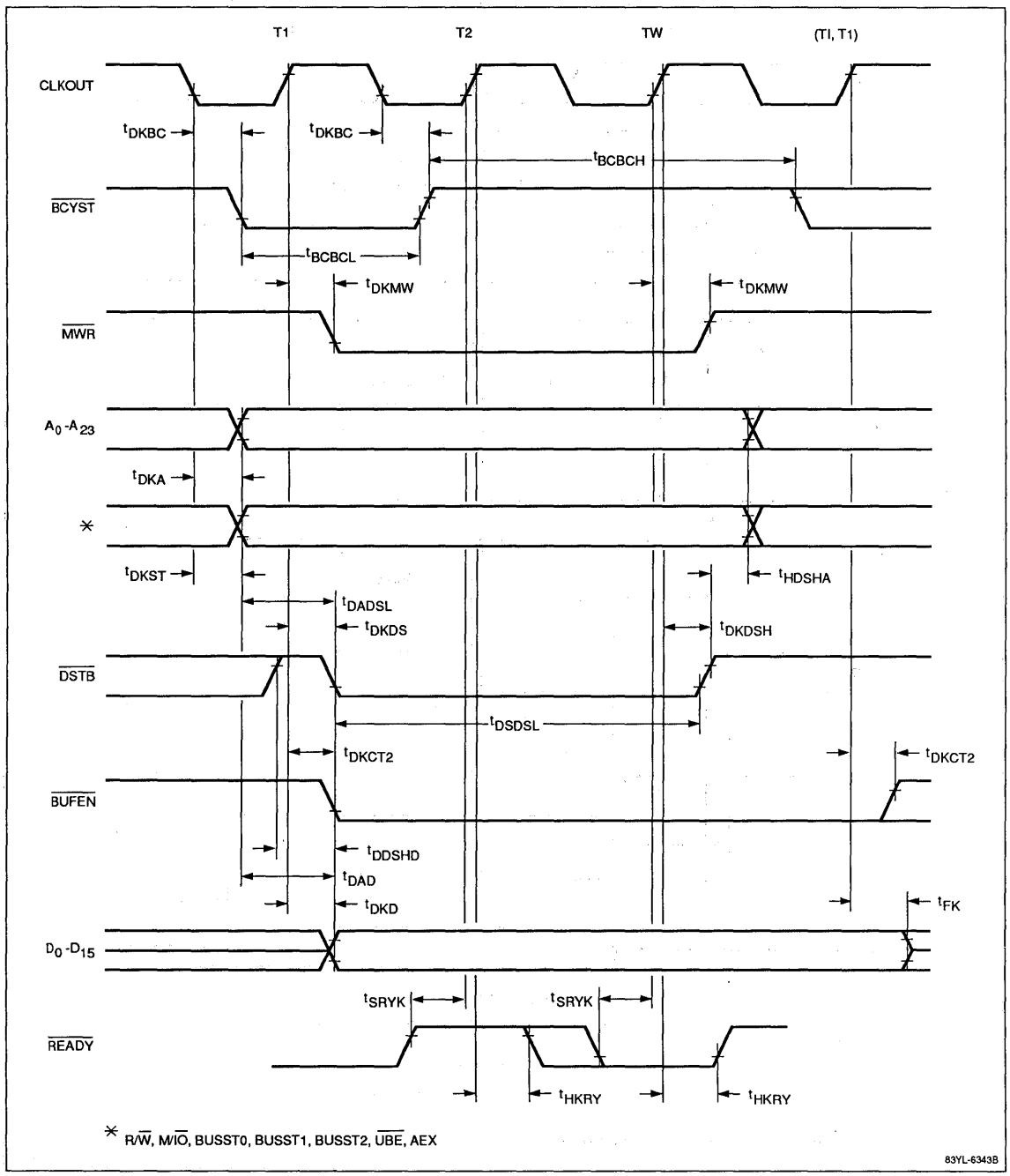
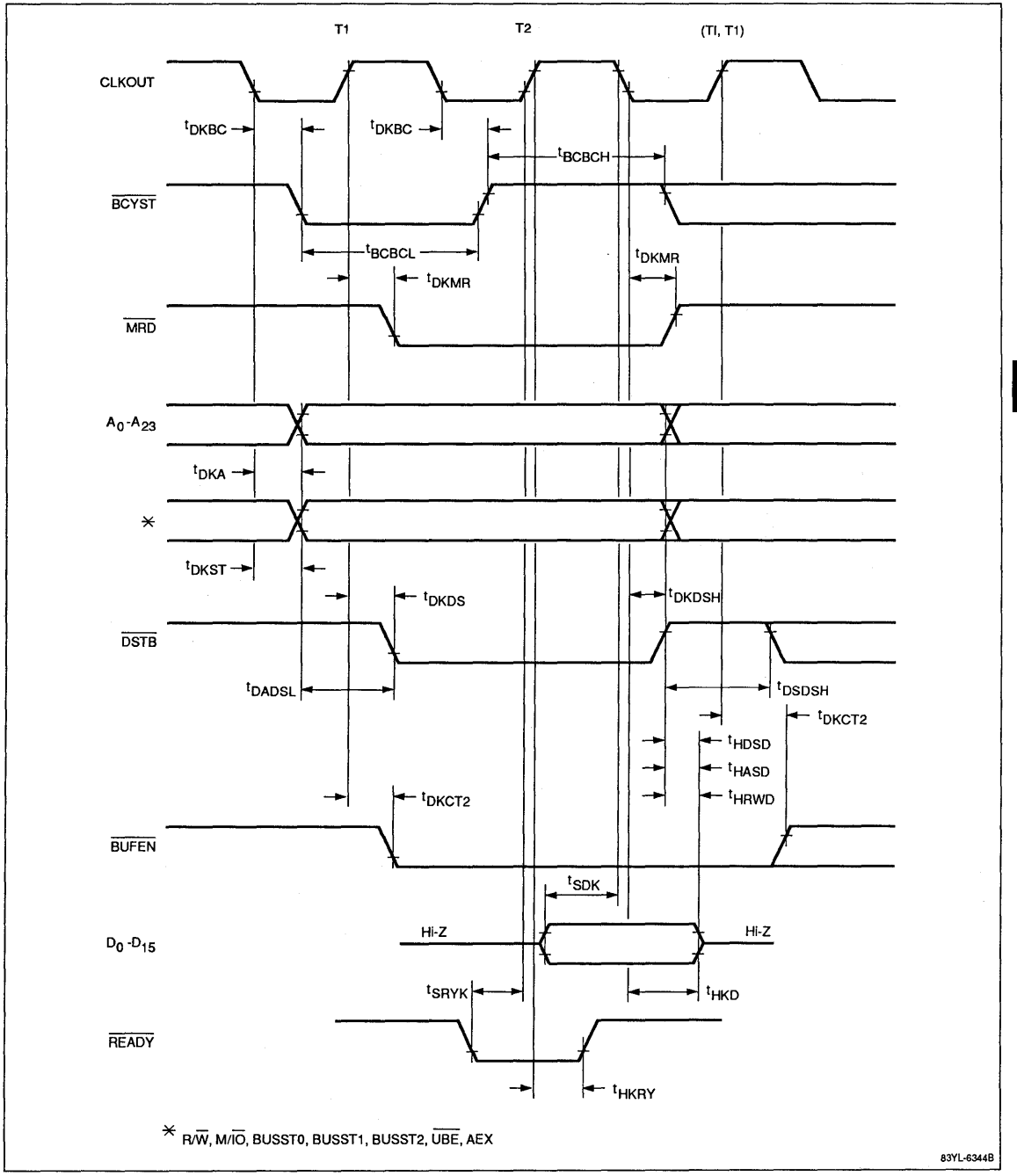


Figure 5. Basic Read (0 Wait)



3f

Figure 6. Basic Read (1 Wait)

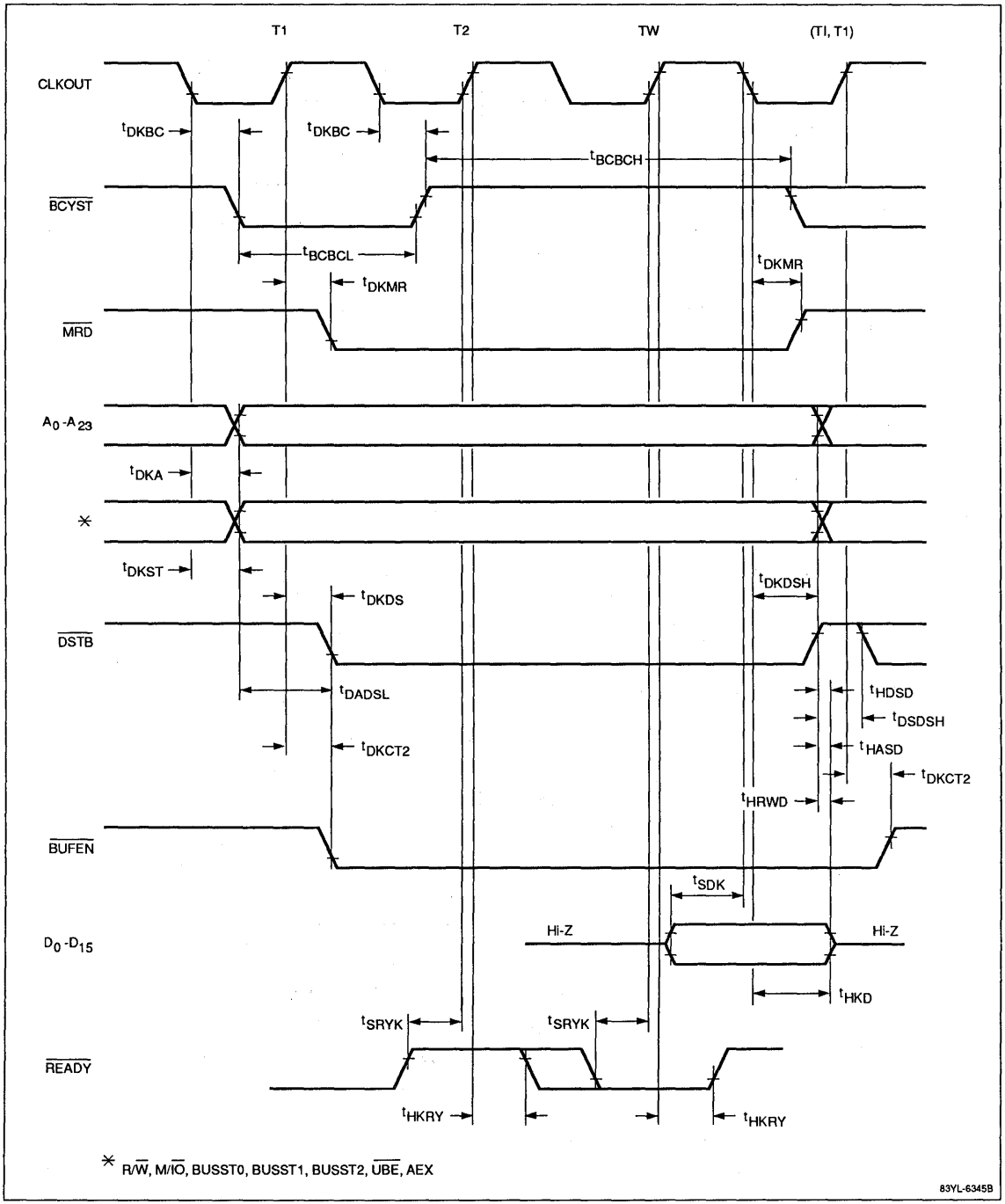
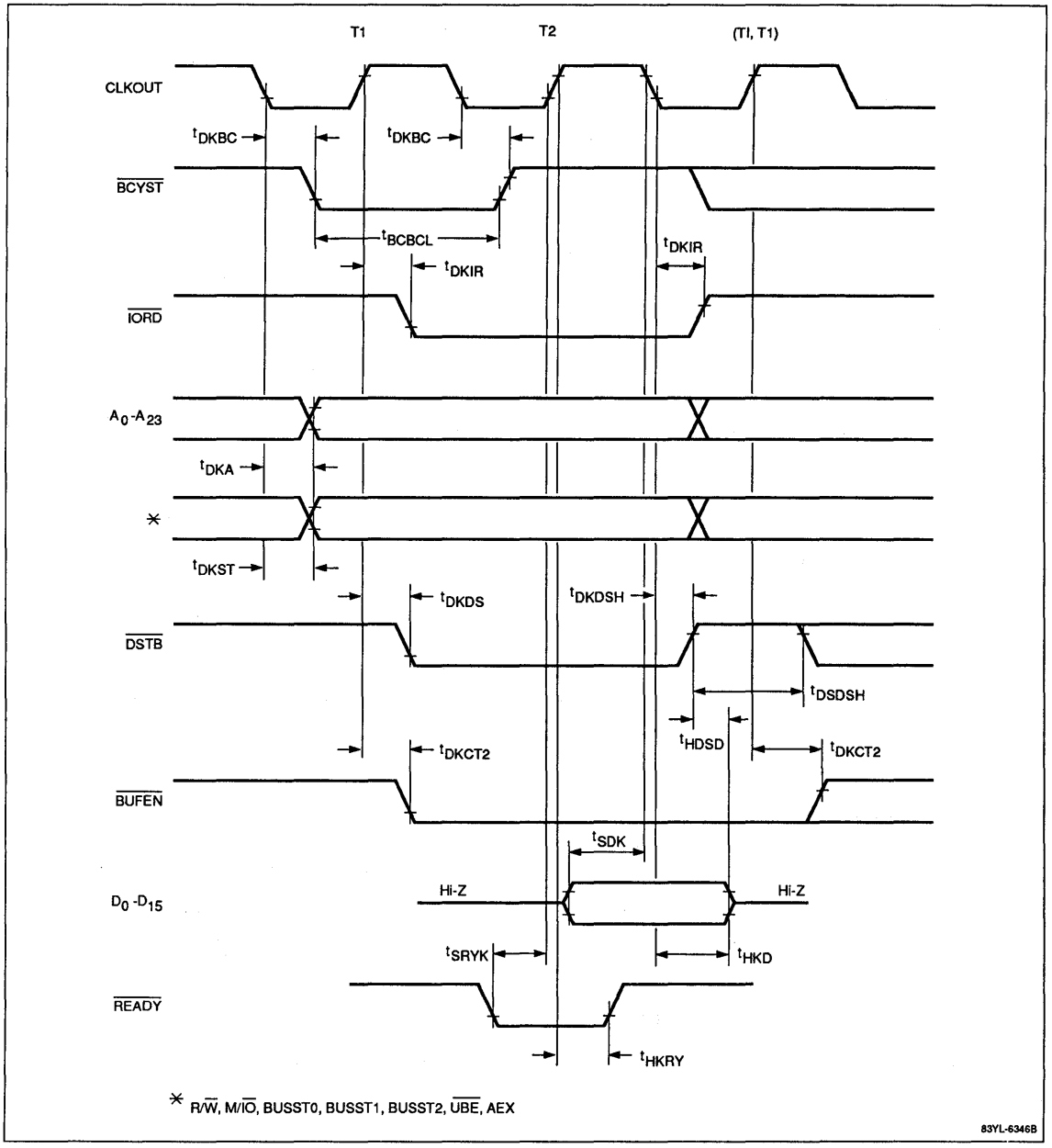


Figure 7. External I/O Read (0 Wait)



3f

Figure 8. External I/O Read (1 Wait)

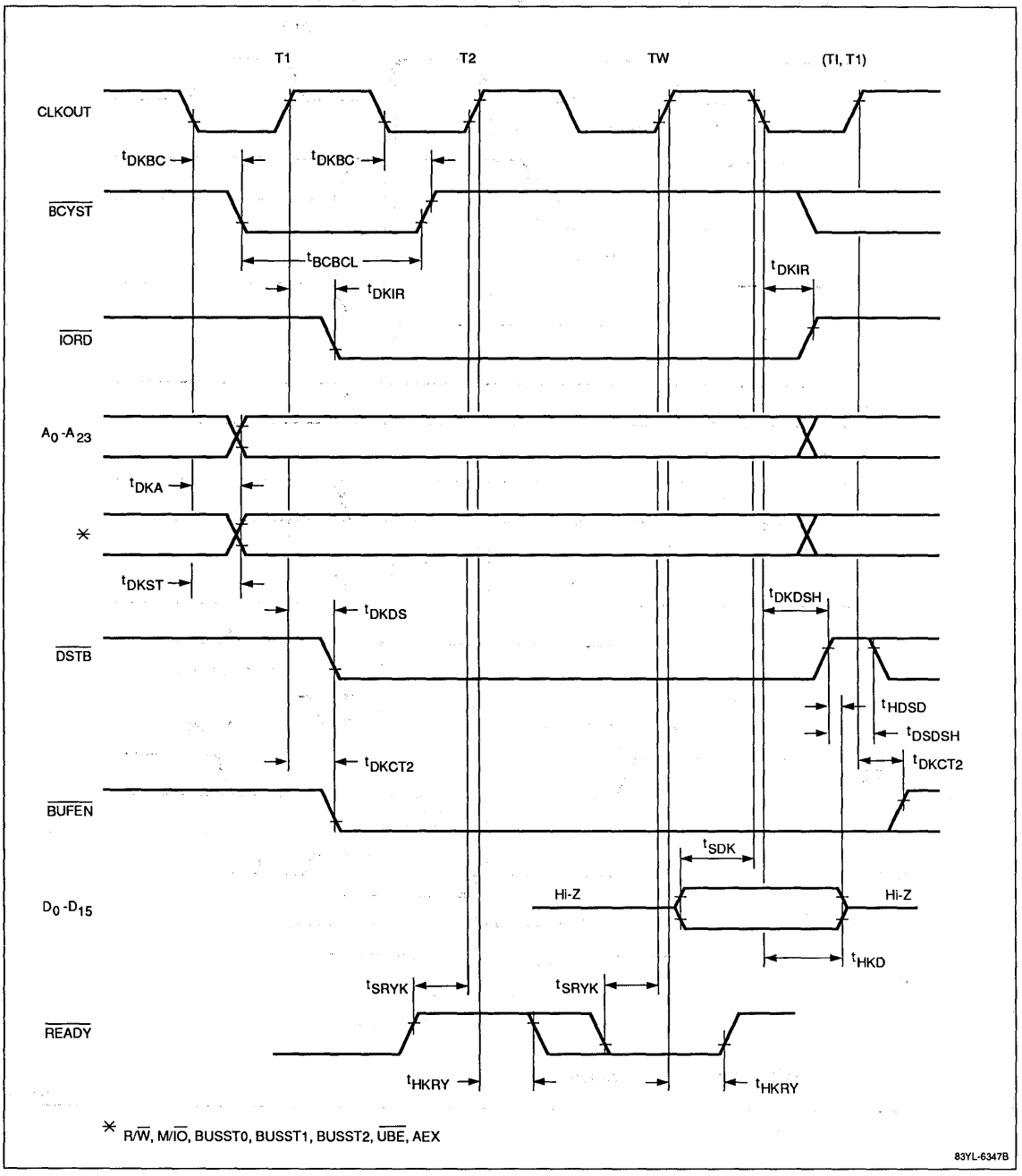
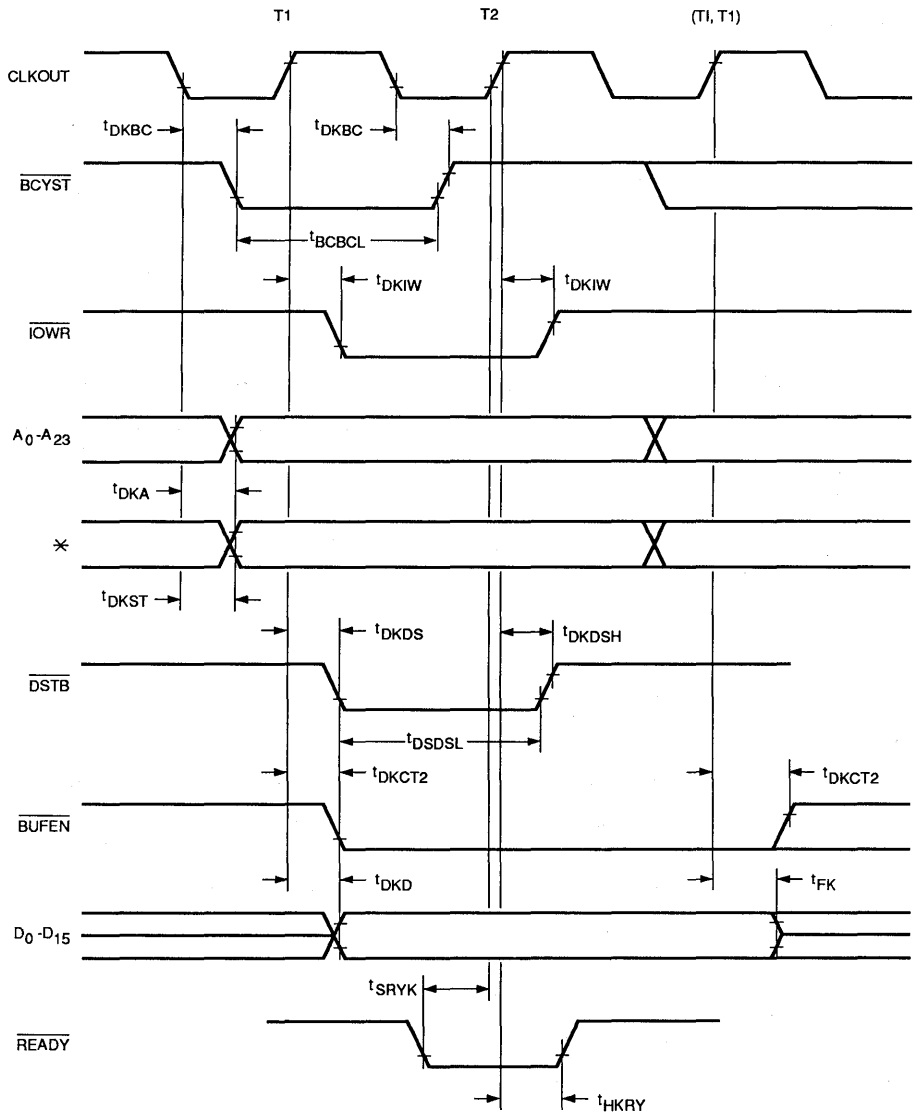


Figure 9. External I/O Write (0 Wait)



3f

Figure 10. External I/O Write (1 Wait)

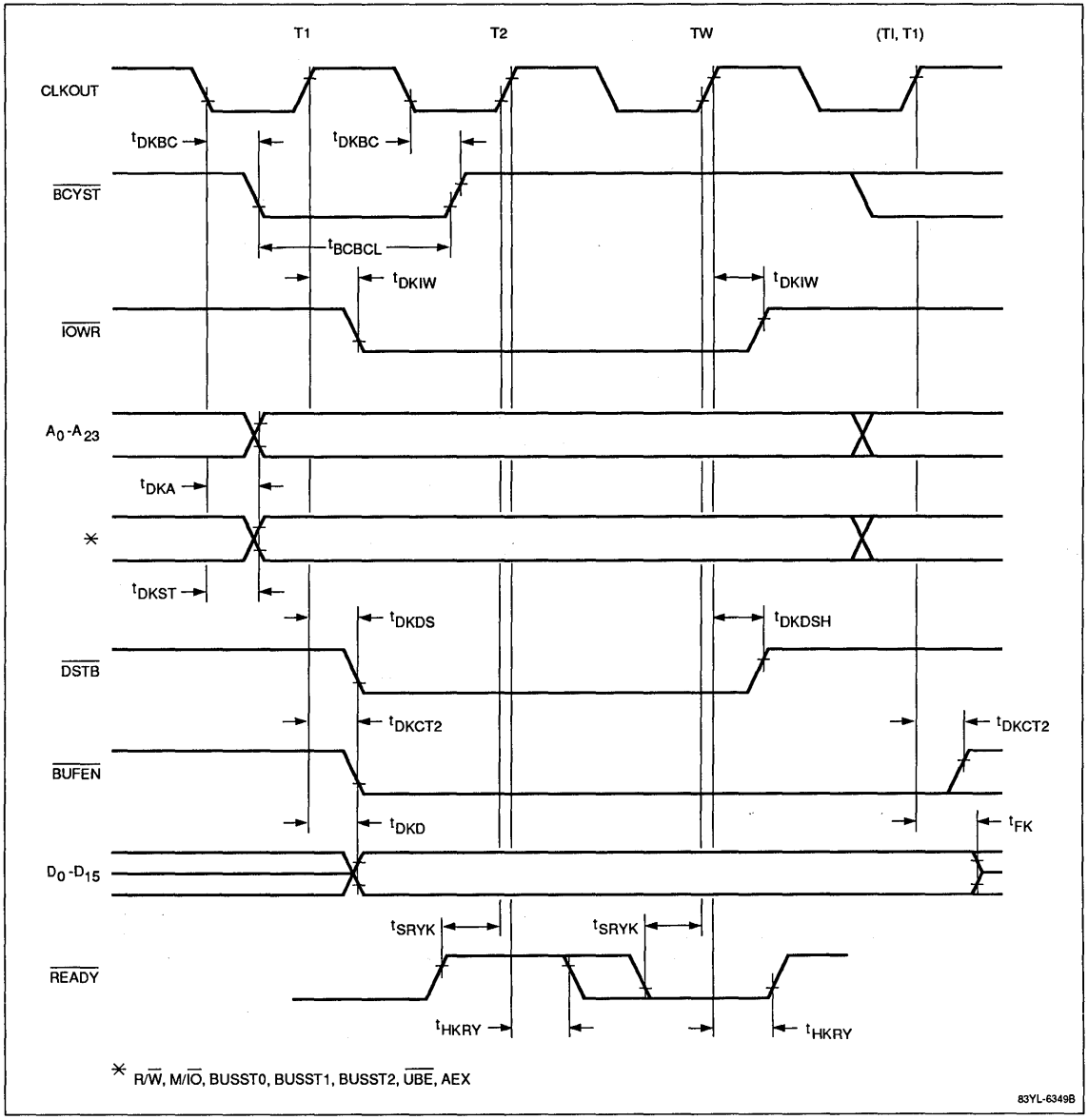


Figure 11. Internal I/O Read

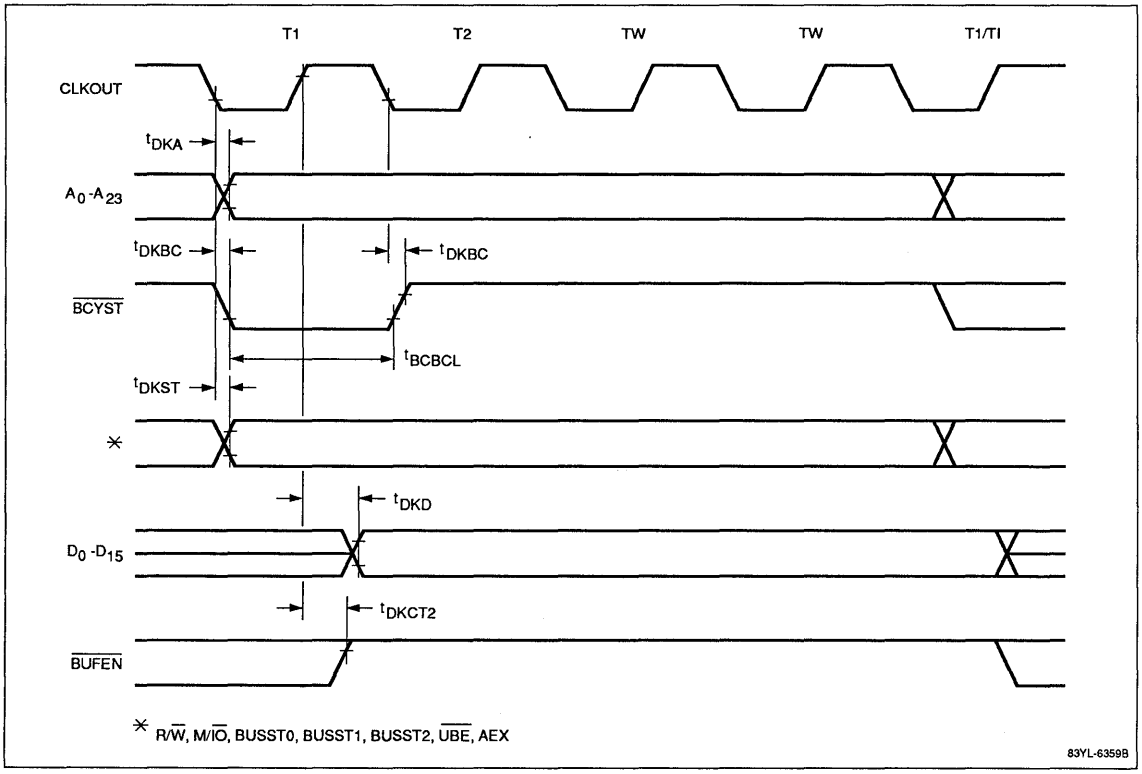


Figure 12. Internal I/O Write

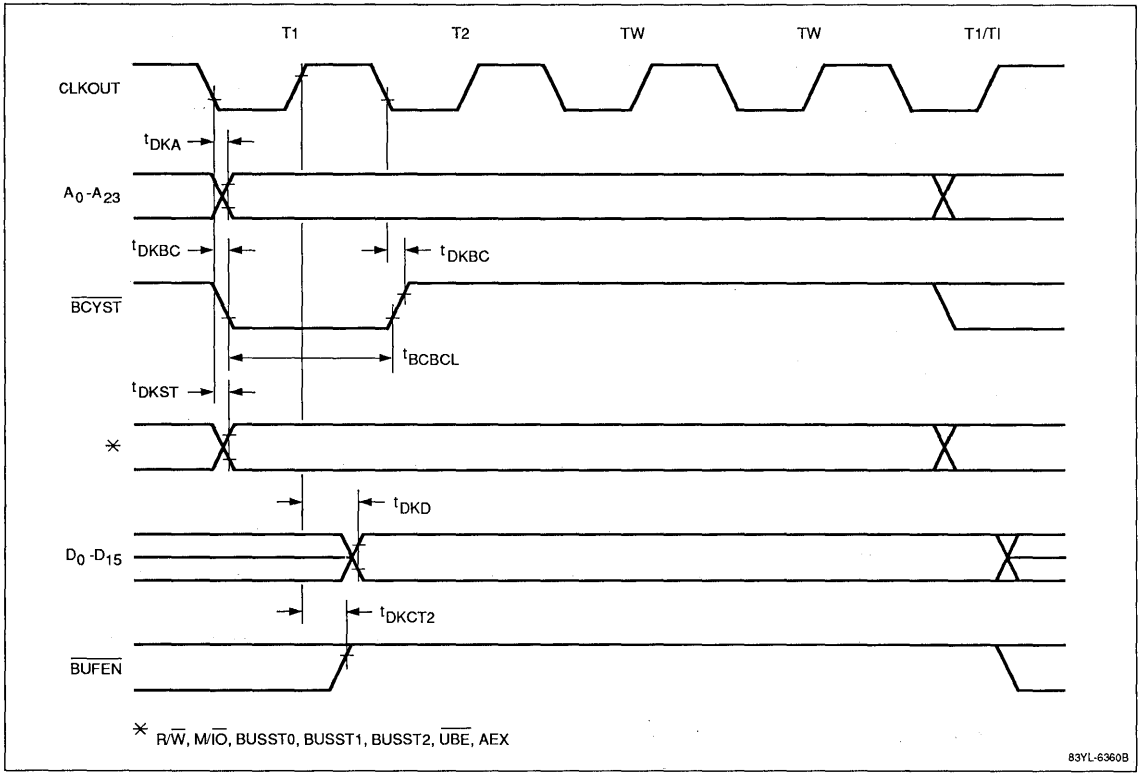


Figure 13. Bus Sizing (0 Wait)

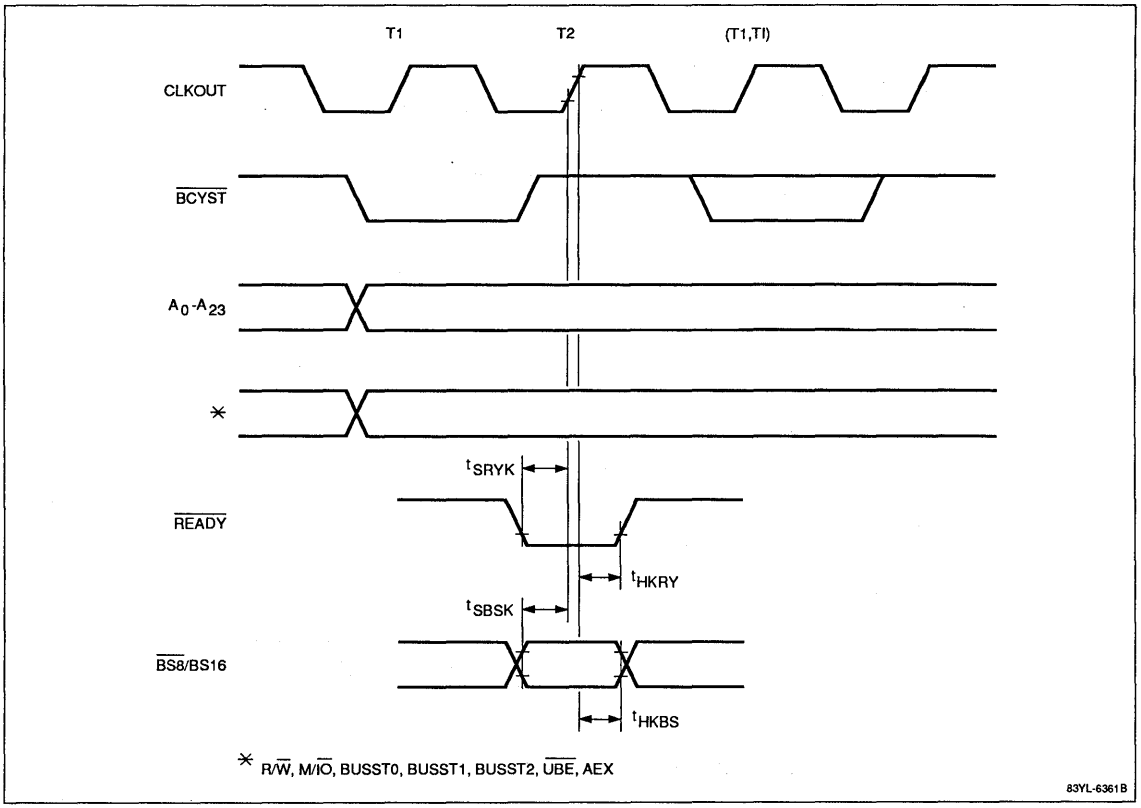


Figure 14. Bus Sizing (1 Wait)

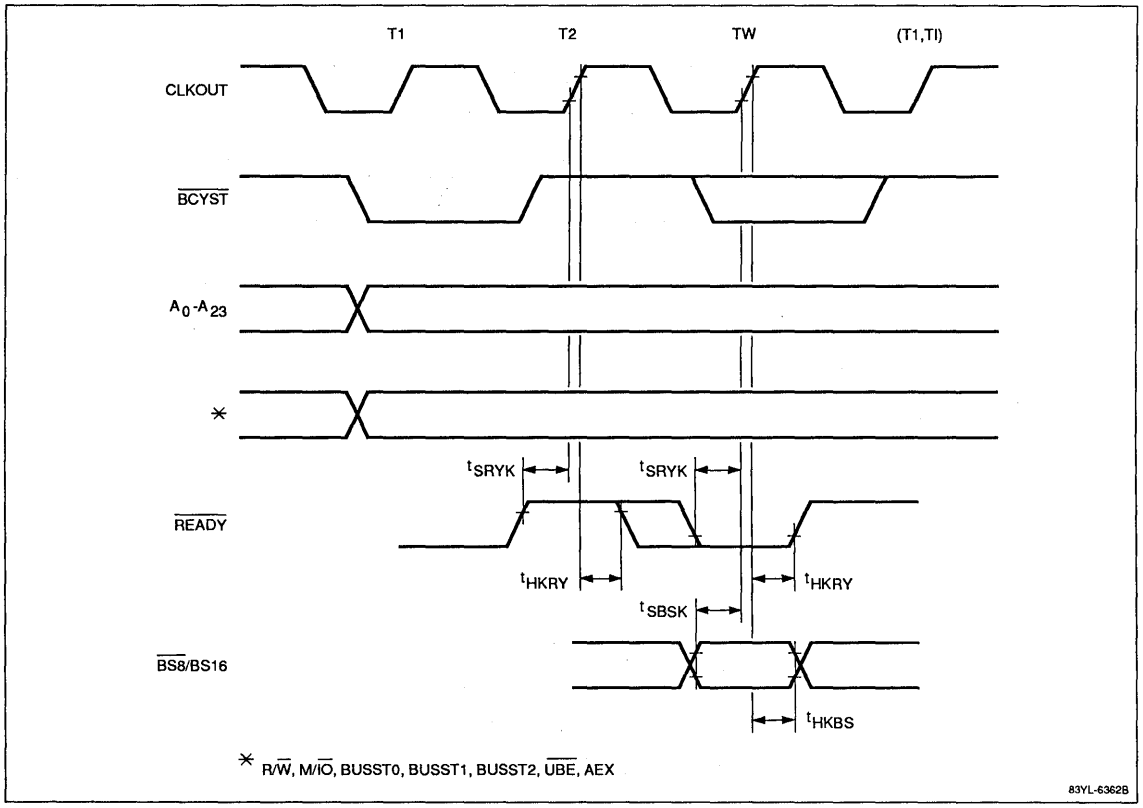


Figure 15. Input Setup/Hold

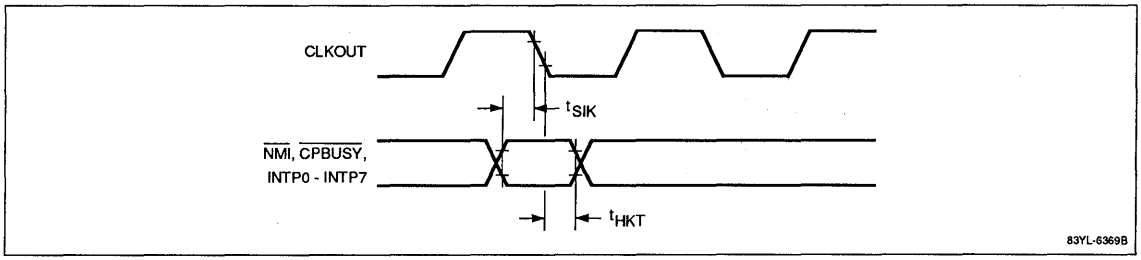
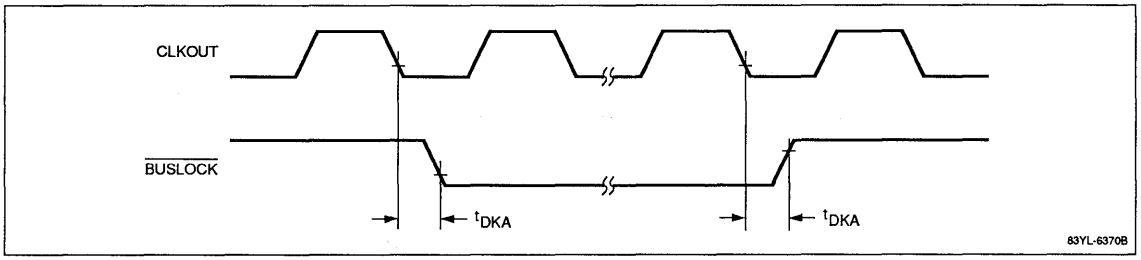
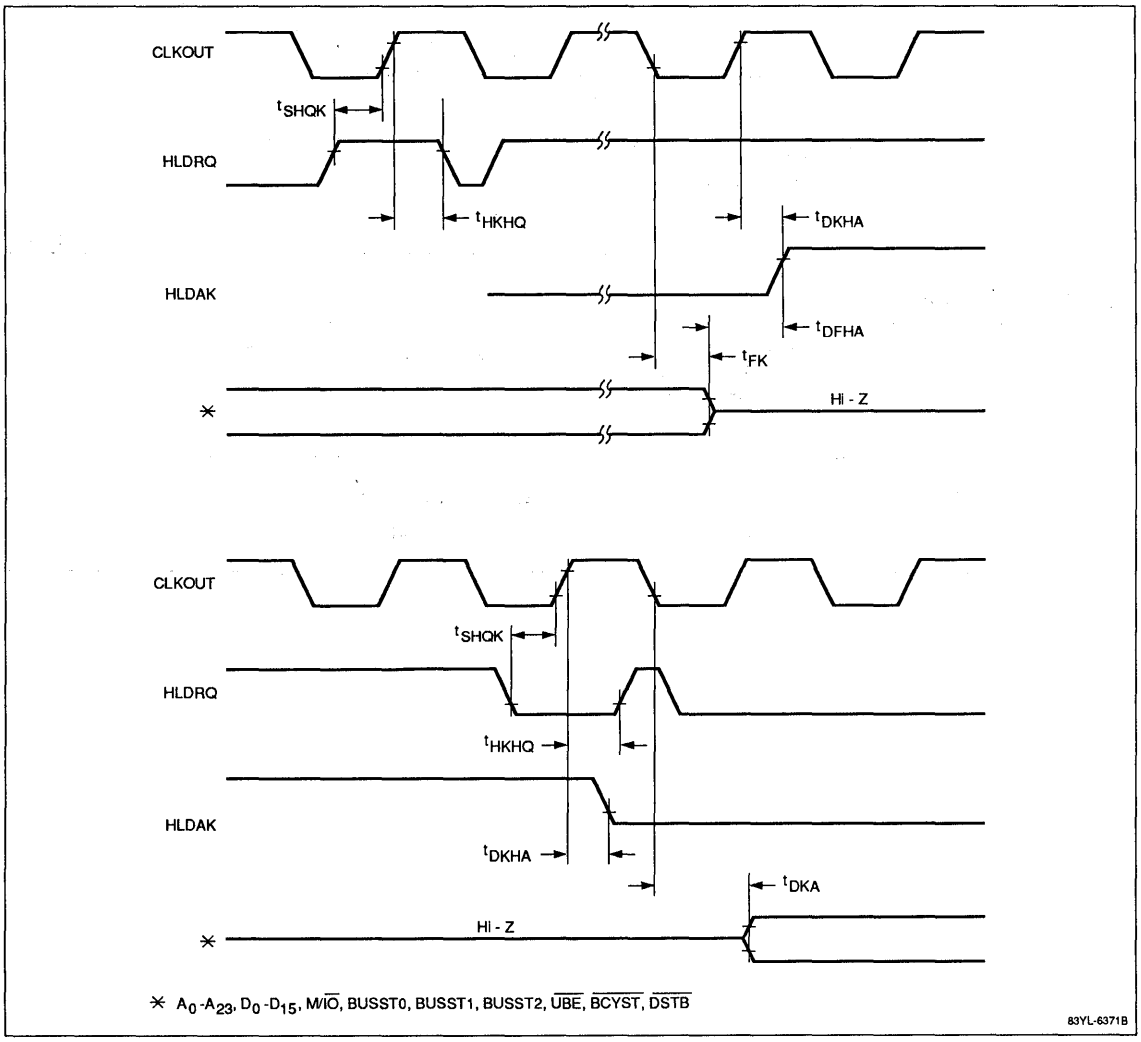


Figure 16. Bus Lock



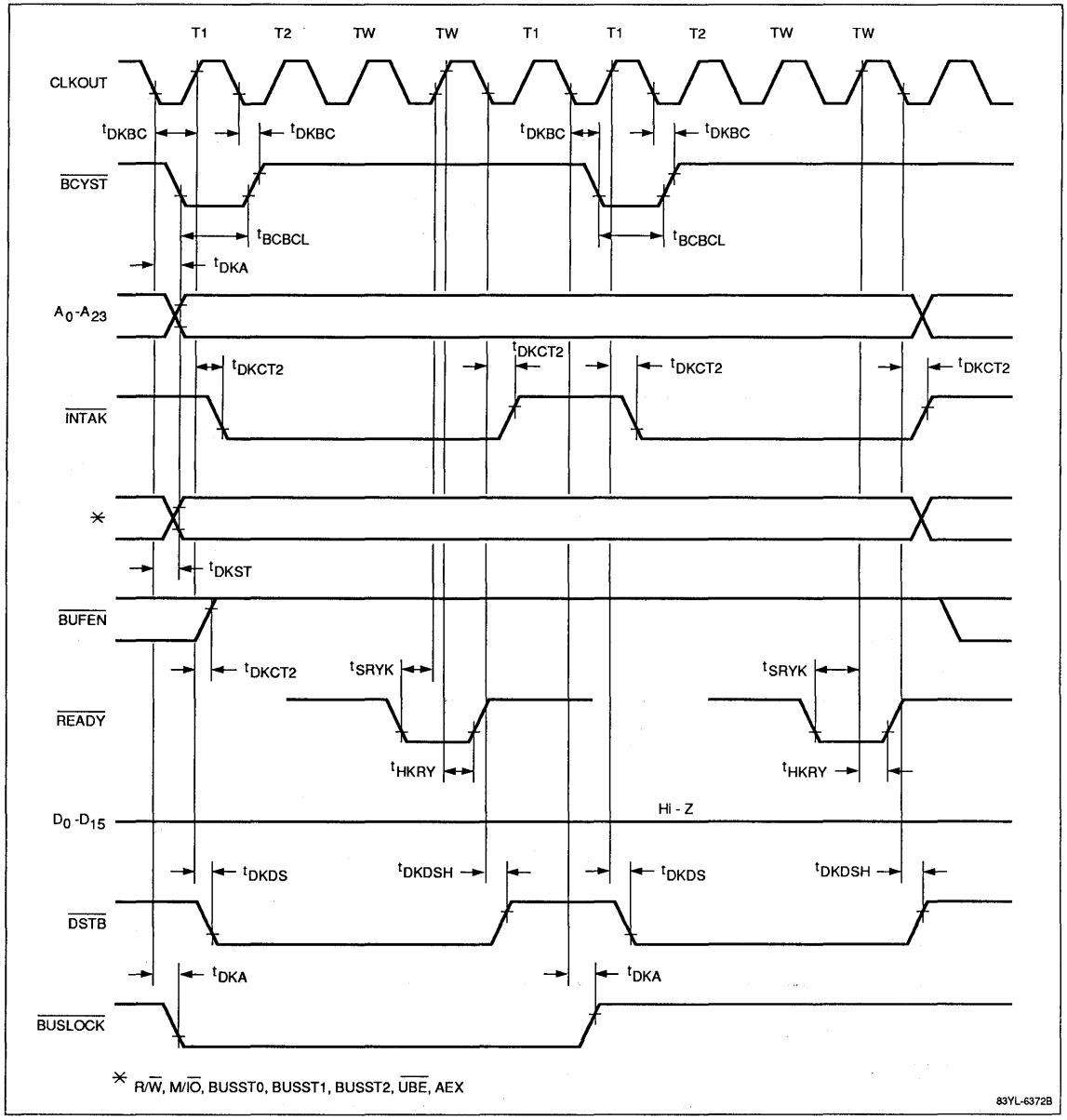
3f

Figure 17. Bus Hold



✳ A_0-A_{23}, D_0-D_{15}, M/$\overline{IO}$, BUSST0, BUSST1, BUSST2, $\overline{UBE}$, $\overline{BCYST}$, $\overline{DSTB}$

Figure 18. Interrupt Acknowledge (Single Mode)



3f

Figure 19. Interrupt Acknowledge (Cascade Mode)

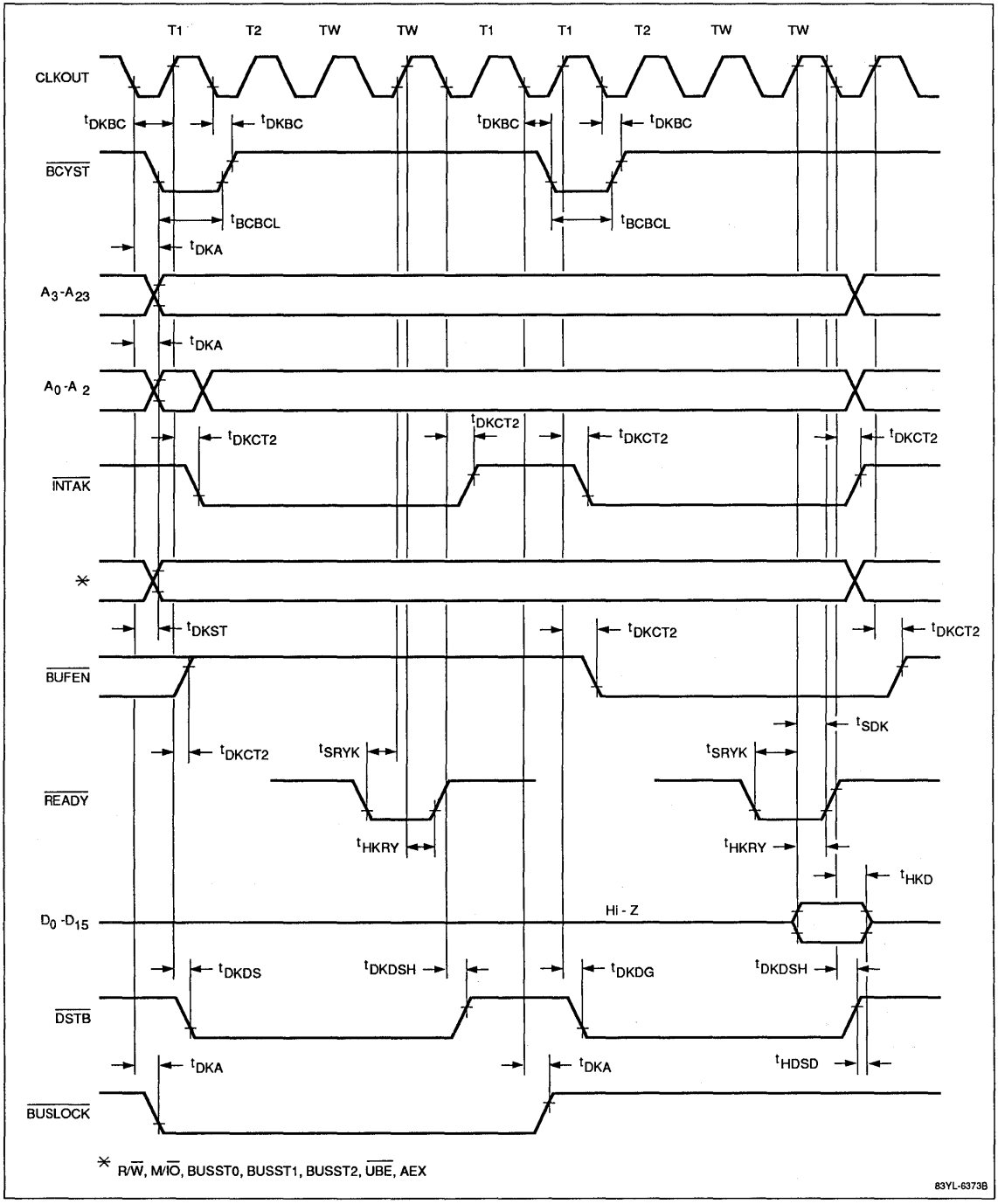
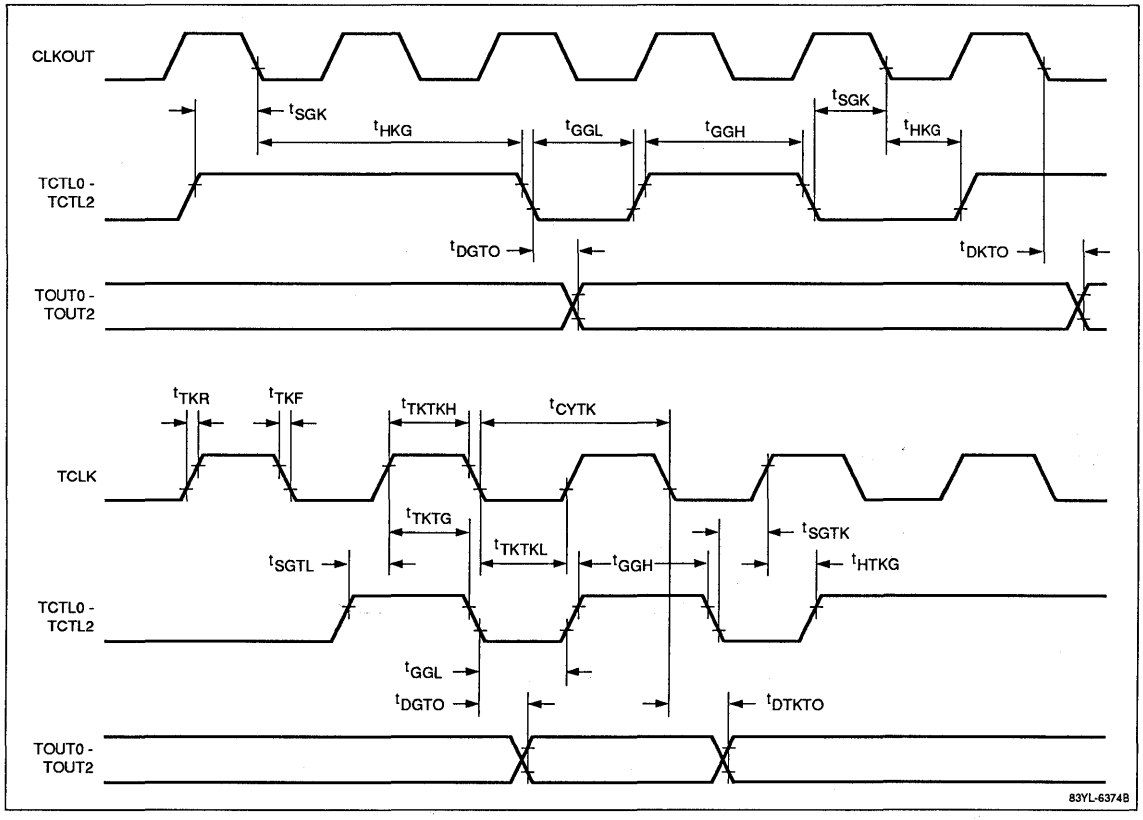


Figure 20. Timer Control Unit (TCU)



3f

Figure 21. Serial Control Unit (SCU)

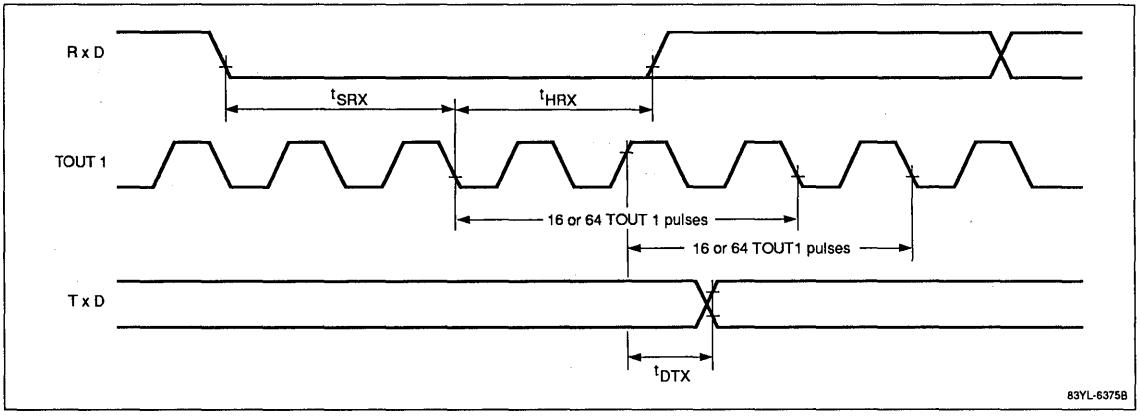


Figure 22. Refresh Timing

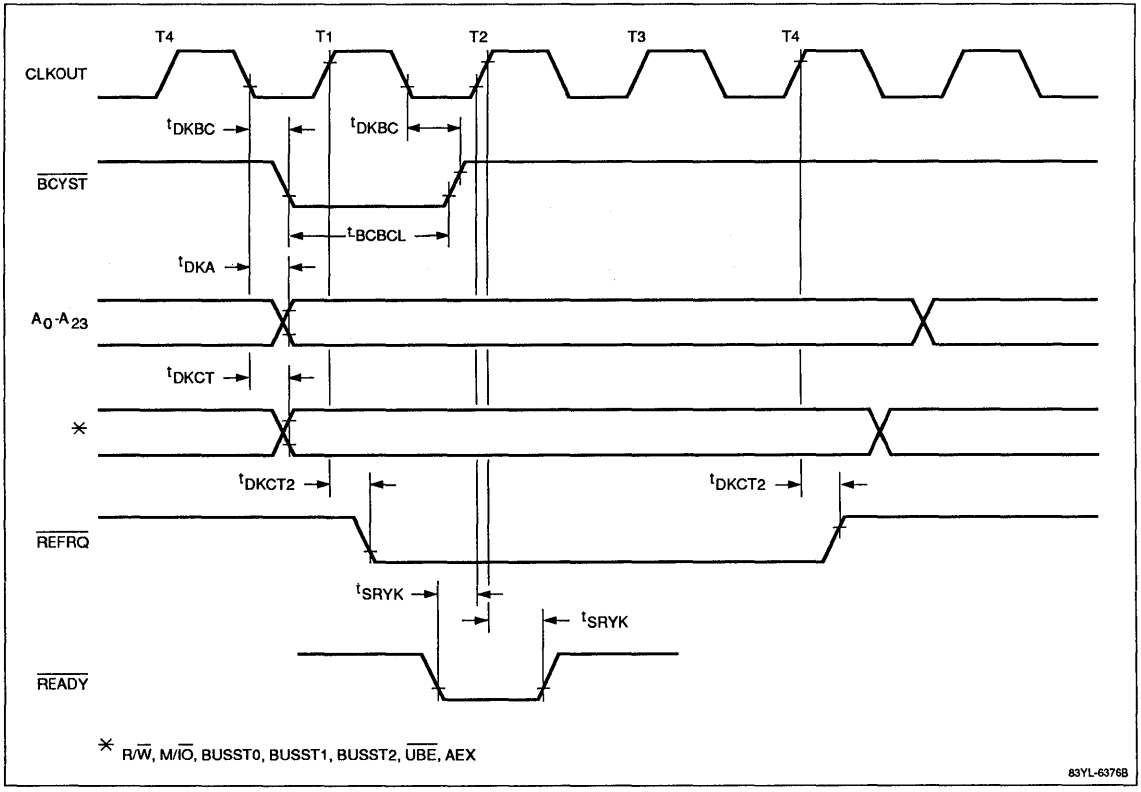
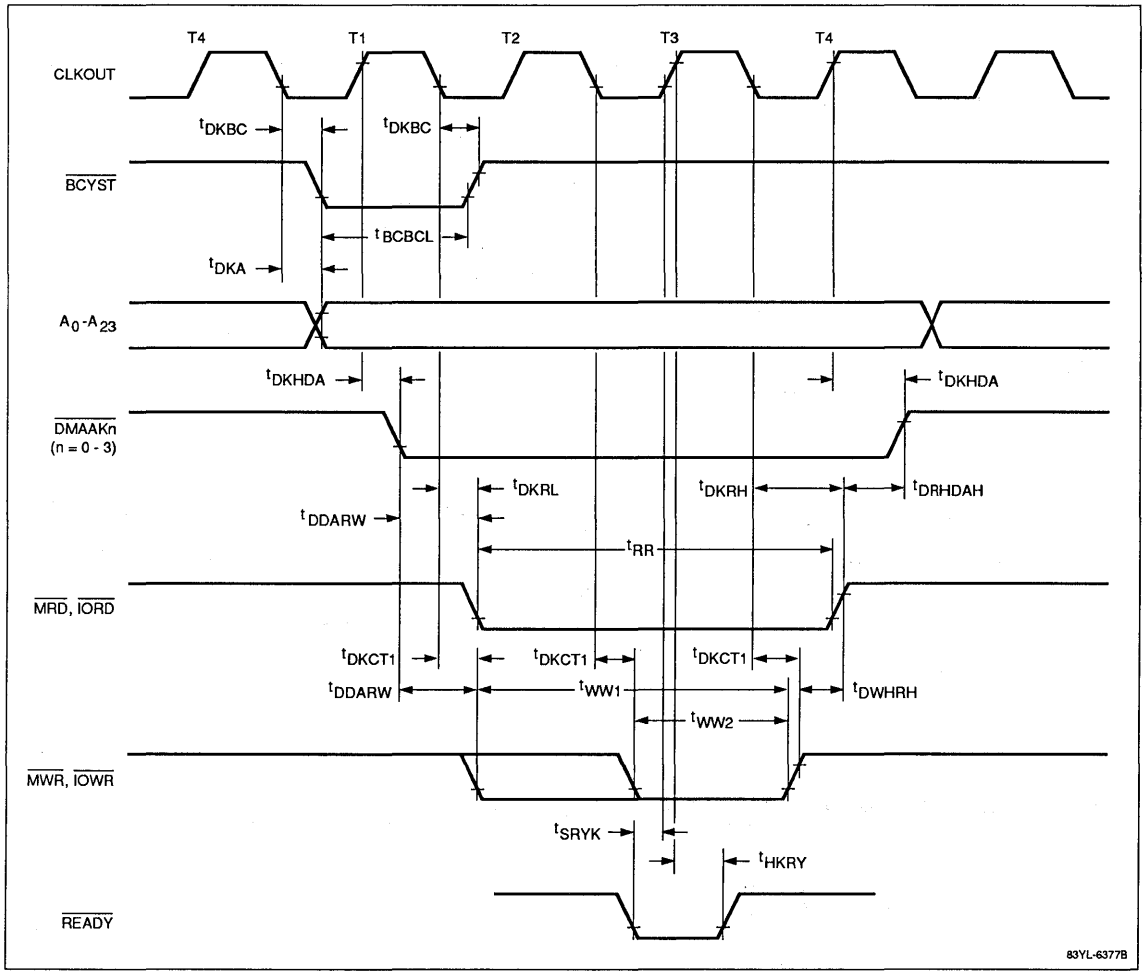


Figure 23. DMA Timing 1



3f

Figure 24. DMA Timing 2

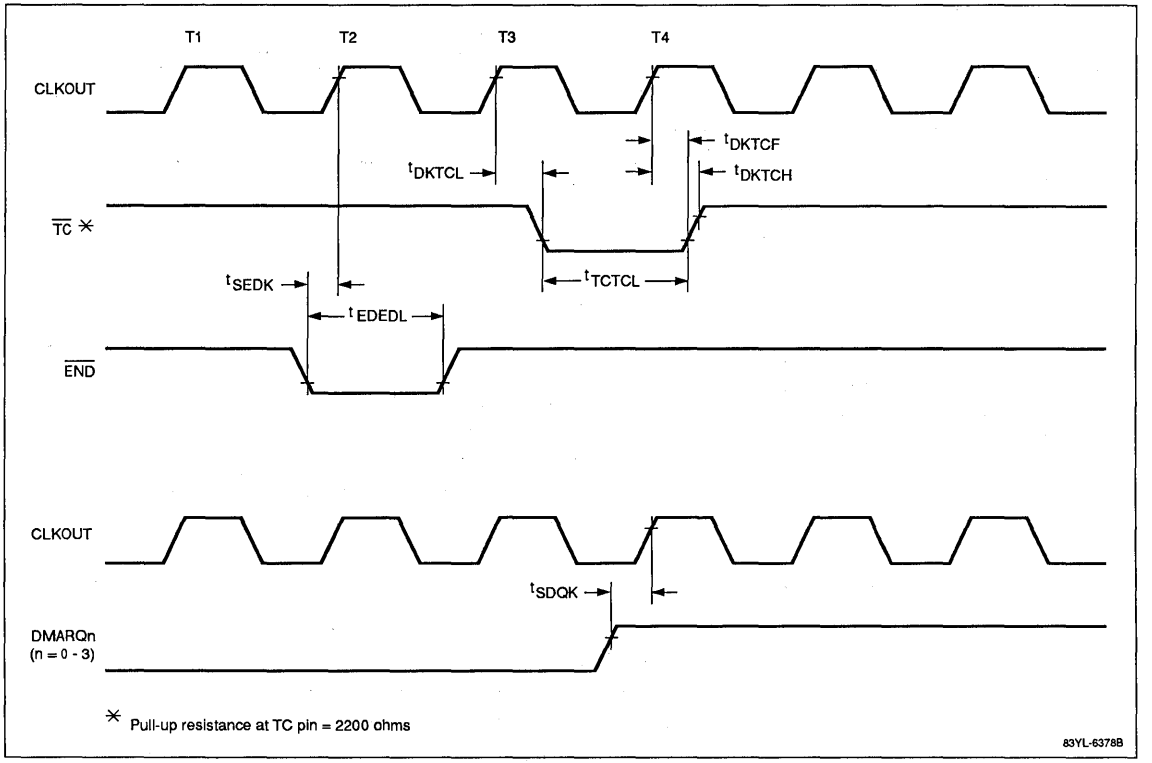


Figure 25. DMA Timing 3; Cascade Mode (Normal Operation)

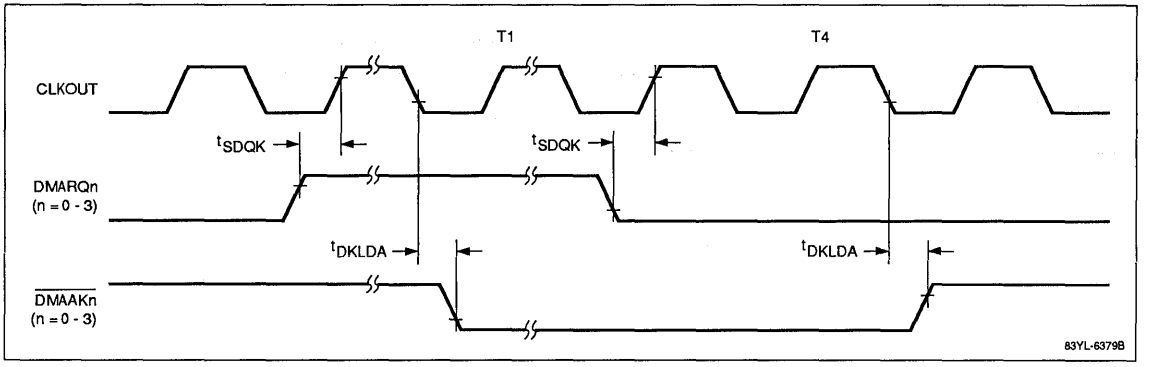


Figure 26. DMA Timing 4; Refresh Cycles To Be Inserted

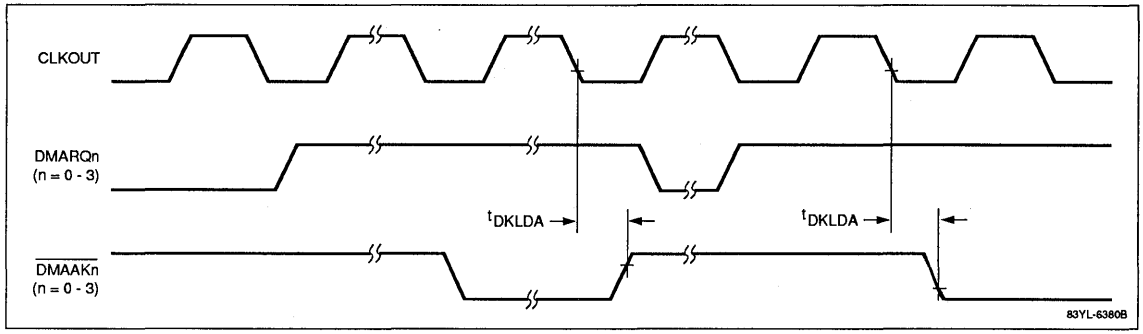
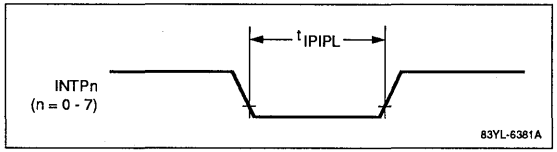


Figure 27. ICU Timing



3f

Figure 28. Memory Write for Coprocessor (0 Wait)

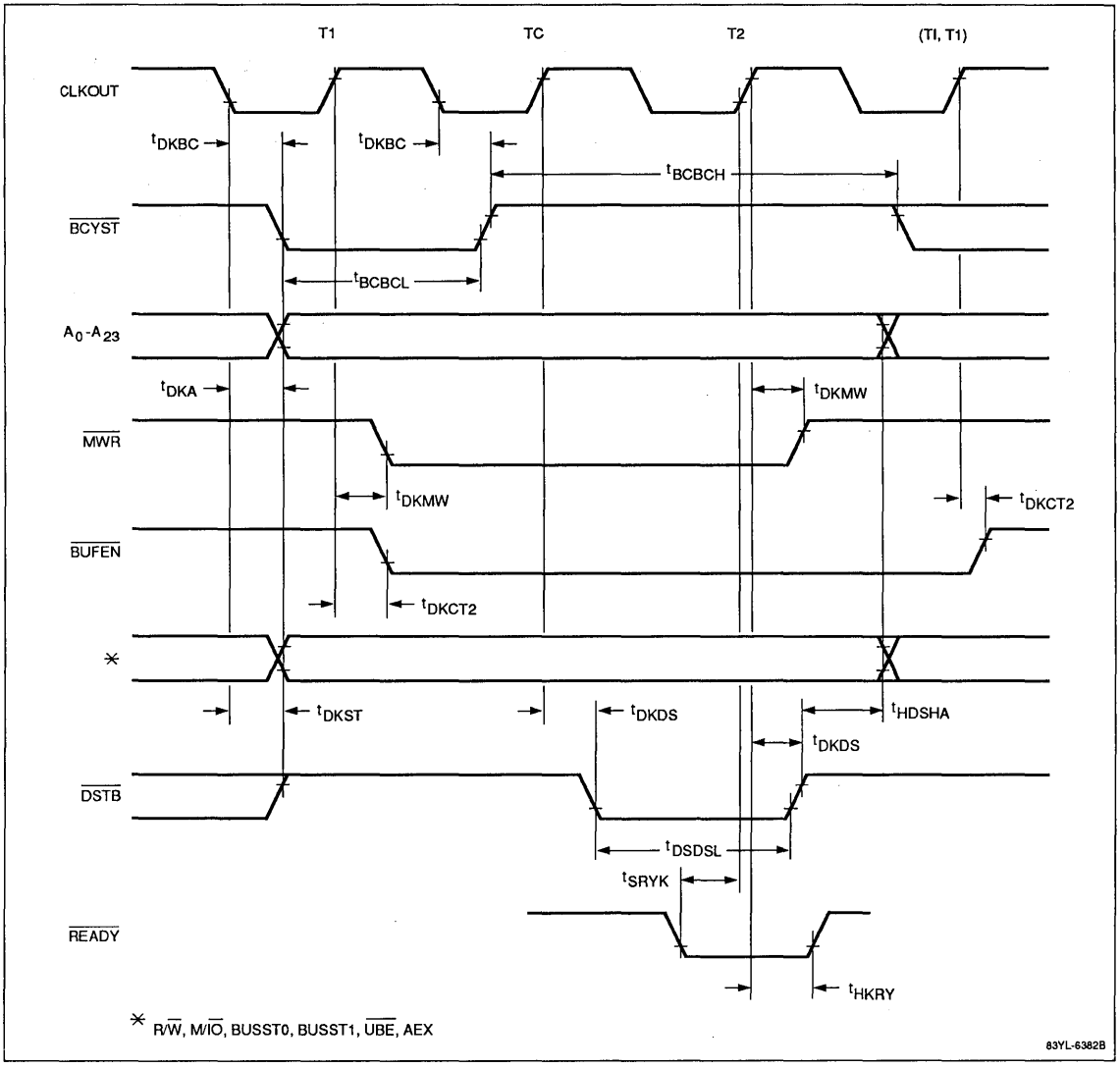
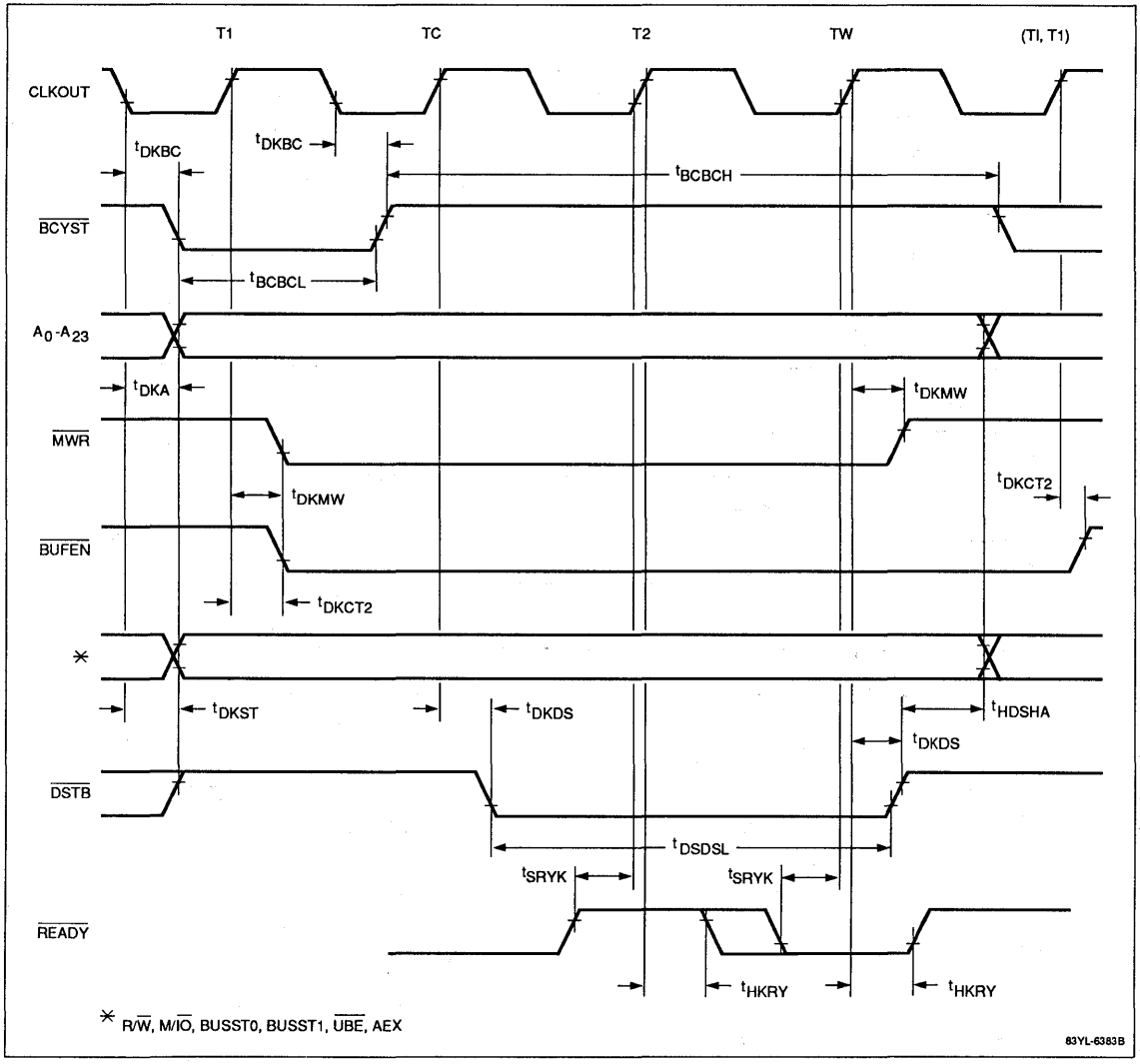


Figure 29. Memory Write for Coprocessor (1 Wait)



3f

Figure 30. Memory Read for Coprocessor (0 Wait)

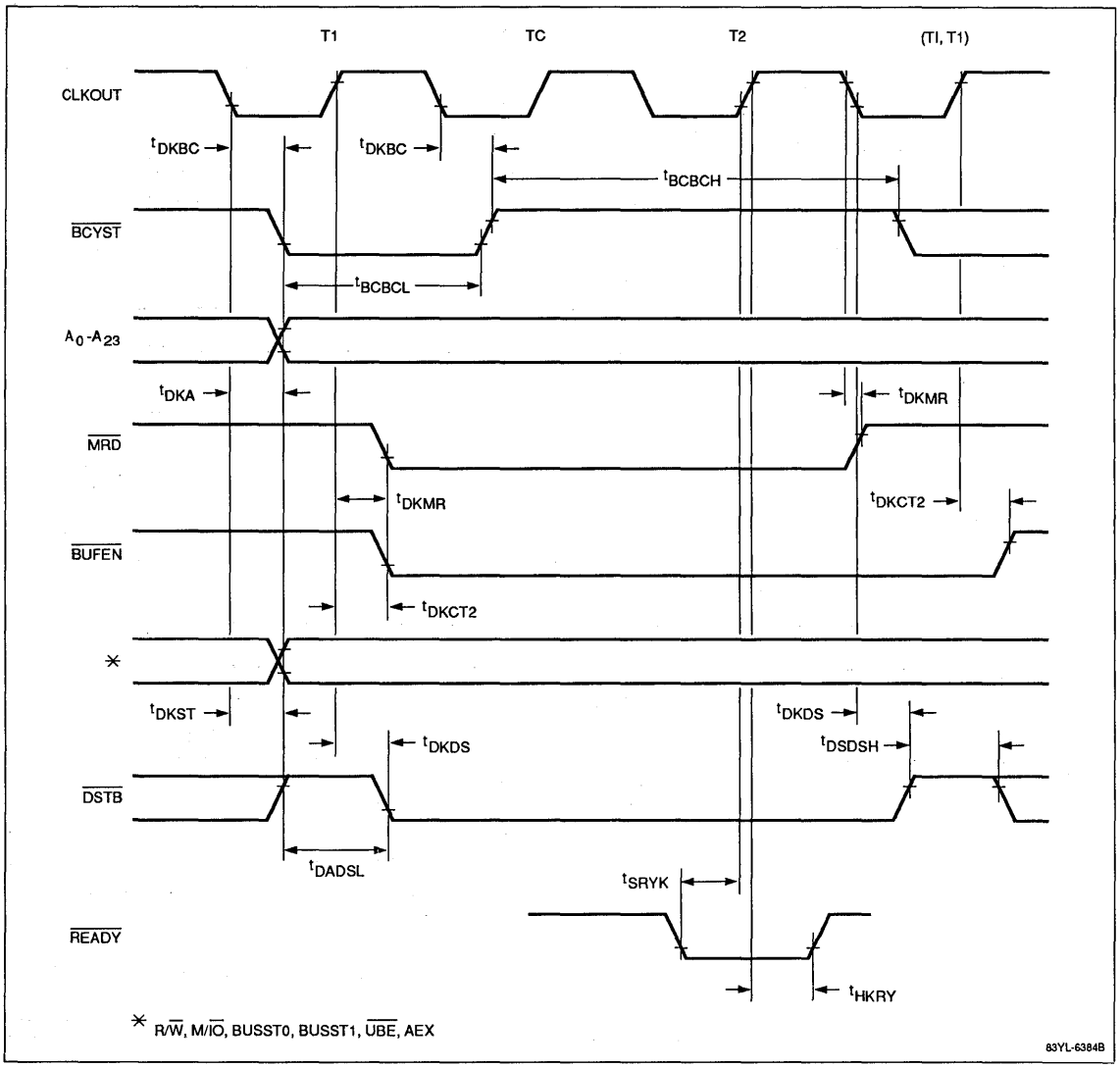
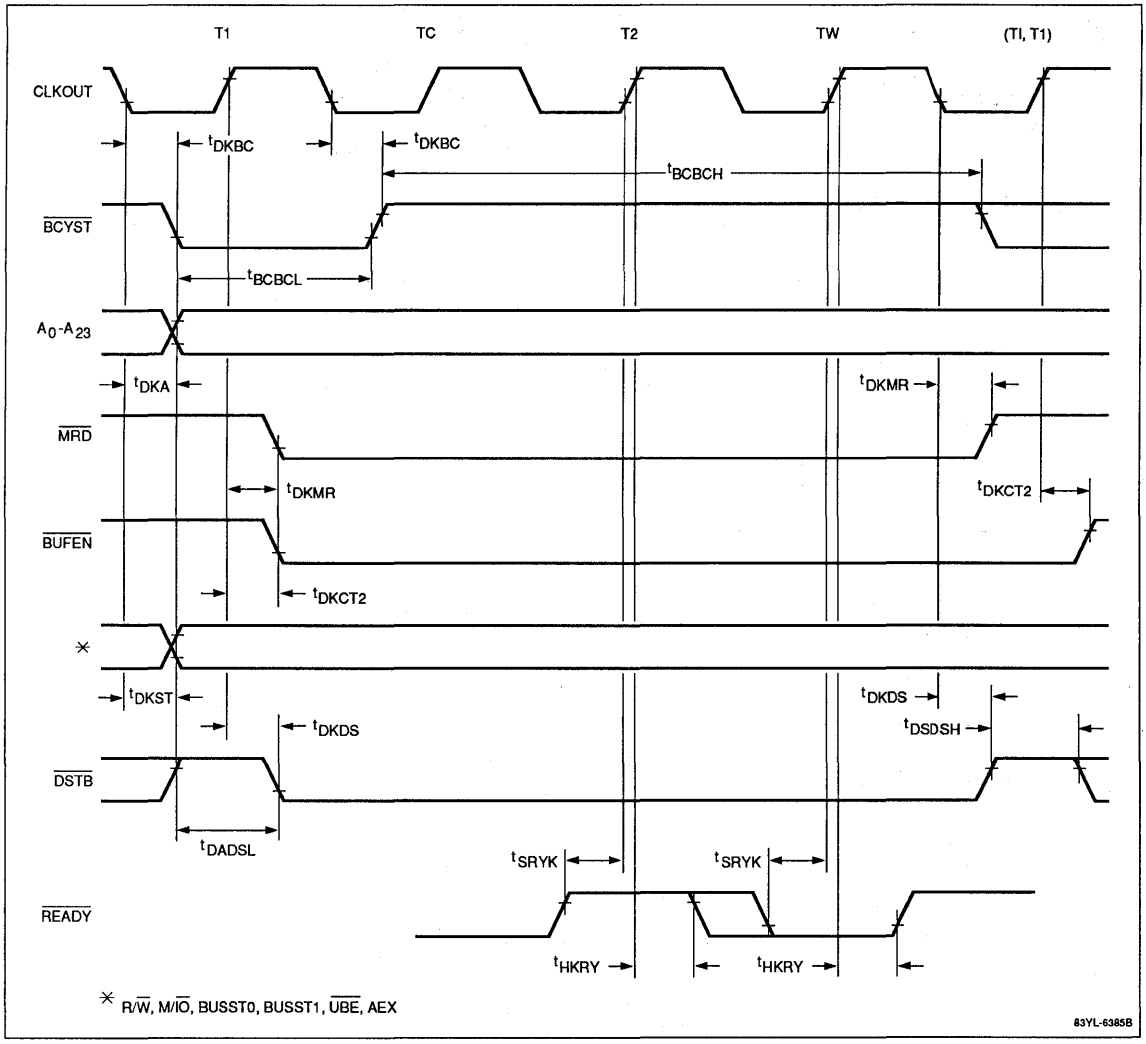


Figure 31. Memory Read for Coprocessor (1 Wait)



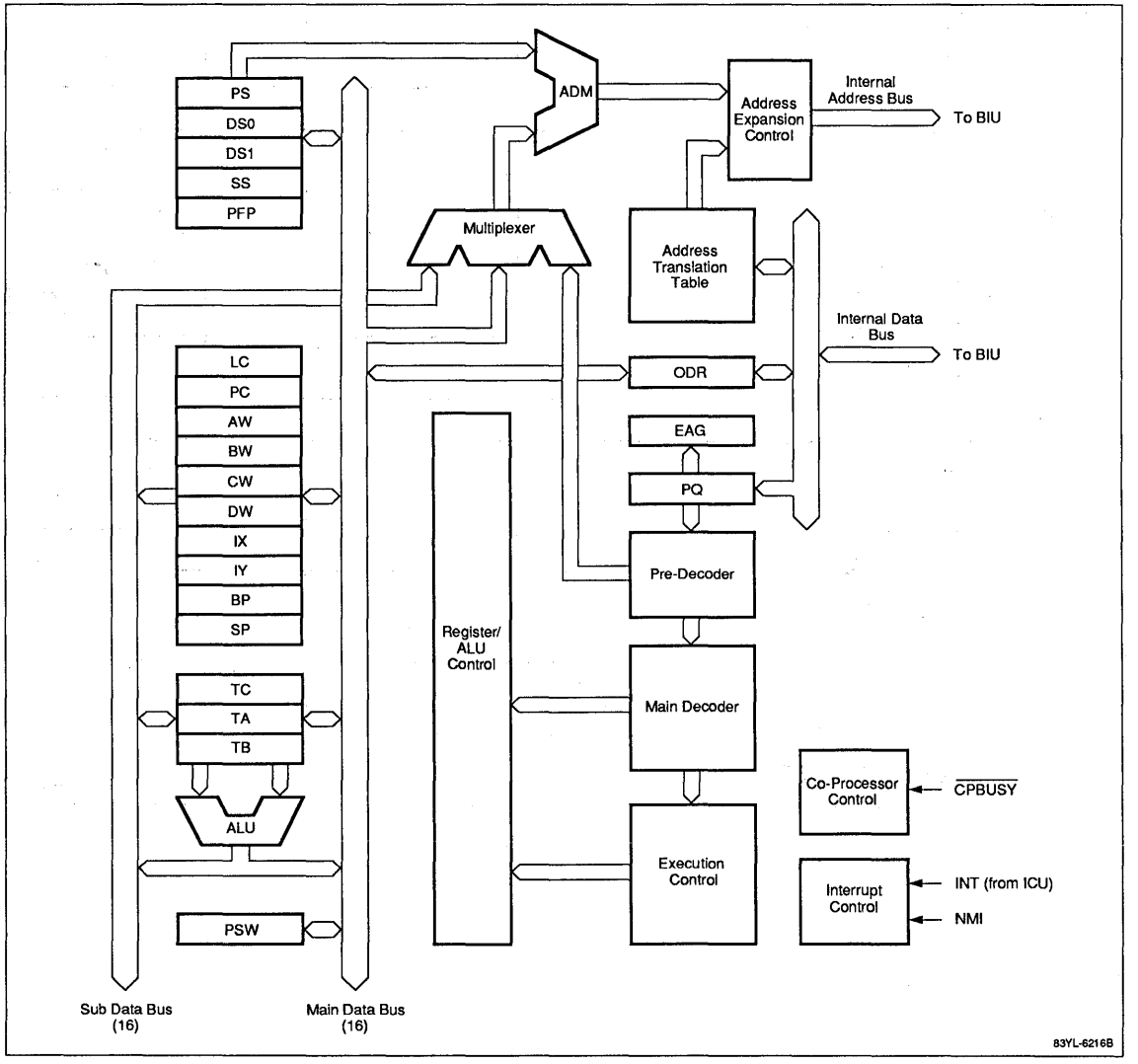
3f

FUNCTIONAL OPERATION

The μPD70236 is described under these major headings.

- Central Processing Unit
- Clock Generator
- Bus Operation
- System Control I/O
- Wait Control Unit
- Refresh Control Unit
- Timer/Counter Unit
- Serial Control Unit
- Interrupt Control Unit
- DMA Control Unit
- Power Conservation

Figure 32. CPU Block Diagram



CENTRAL PROCESSING UNIT (CPU)

Architecture

A unique hardware architecture feature of the CPU is that it contains no microcode. Instruction decode and data path control are implemented using logic and small independent state machines. This greatly enhances instruction execution speed. The V53 is four times faster than the V30.

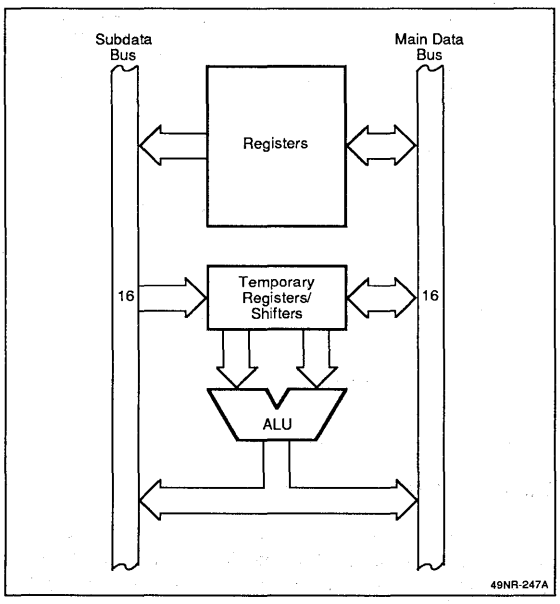
The CPU comprises the execution unit and the address generator. Figure 32 is the CPU block diagram.

CPU Execution Unit

The execution unit consists of a register file, an ALU, and instruction decode and execution control logic.

In addition to the hardware control logic, the most significant feature of the execution unit is a dual-bus internal data path (figure 33). The ALU and many registers are dual ported with a data bus on each port. This allows two operands to be transferred in one clock cycle instead of two. Performance is improved as much as 30% by the dual data bus concept.

Figure 33. Dual Data Buses



Register File. There are 12 registers in the internal RAM. Four are temporary registers used in the execution of certain instructions (LC, TA, TB, and TC). The other eight are general-purpose registers (AW, BW, CW, DW, IX, IY, BP, and SP). These contain either operand data or point-to-operand data in memory.

The temporary registers speed up instruction execution by serving as scratch pad registers during complex operations.

The loop counter (LC) is used during primitive block transfer operations. It contains the count value. It is also a shift counter for multiple-bit shift and rotate instructions.

Temporary registers TA, TB, and TC are inputs to the ALU. They are used as temporary registers/shifters during multiply, divide, shift/rotate, and BCD rotate operations.

ALU. The ALU consists of a complete adder and logical operation unit. It executes arithmetic (ADD, SUB, MUL, DIV, INC, DEC, NEG, etc.) and logical (TEST, AND, OR, XOR, NOT, SET1, CLR1, etc.) instructions.

Data Path Control Logic. This logic comprises the main instruction decoder and the execution control blocks. Its purpose is to determine which operations must be done and to schedule them. It transfers operands, as required, and controls the ALU. State machines implement long, complex instructions.

Instruction Prefetching. The V53 is a pipelined machine. To keep the pipeline running efficiently, it should be kept full of instructions in various stages of execution. Instructions are fetched before they are needed and placed in the instruction processing queue (IPQ).

Data in the IPQ is broken out by the decoder logic to determine what addressing modes will be used and what CPU resources are required to execute the prefetched instruction. To keep the 8-byte IPQ full, the bus control logic schedules an instruction prefetch cycle whenever there are at least 2 unused bytes in the IPQ.

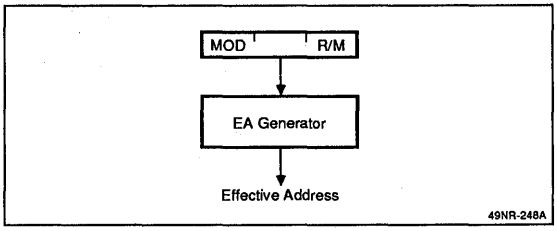
The IPQ is cleared whenever a control transfer instruction (any branch, call, return, or break is executed). This is done because a different instruction stream will be used following a control transfer, and the IPQ will then contain instruction data that will never be used. When this happens, the V53's pipeline is emptied and performance is reduced. To maximize performance, the number of control transfers should be minimized.

Effective Address Generator. The effective address generator (EAG) logic computes a 16-bit effective address for each operand. This address is an offset into one of the four segments. Refer to figure 34. This effective address is passed on to the address modifier adder. The EAG decodes the first byte(s) of each instruction to

3f

determine the addressing mode and initiates any bus cycles required to fetch pointers/offsets from memory. Effective addresses are calculated in a maximum of 1 clock period as compared with 5 to 12 clocks for a microprogrammed machine.

Figure 34. Effective Address Generator



Address Generator

The address generator comprises the address register file, the address modifier (ADM), the address translation table, and the needed control logic.

The registers in the address register file are PS, SS, DS0, DS1, PC, and PFP. The ADM is a dedicated adder that adds one of the segment registers to the effective address to produce the 20-bit normal address. The ADM also increments the prefetch pointer. If extended addressing is enabled, the address translation table is accessed to map the 20-bit address into a 24-bit extended address.

For instruction stream data, addresses are generated differently. The prefetch pointer contains a 16-bit offset into the PS segment that points to the next instruction word to be prefetched. The program counter contains an offset into the PS segment that points to the instruction that is currently being executed. As part of all control transfers, the PFP is set to the same value as the PC.

CPU Addressing Mechanism

The V53 is completely compatible with the μPD70108/116 in its addressing modes and in the way that addresses are computed. It offers a method of expanding the memory address space to 16M bytes.

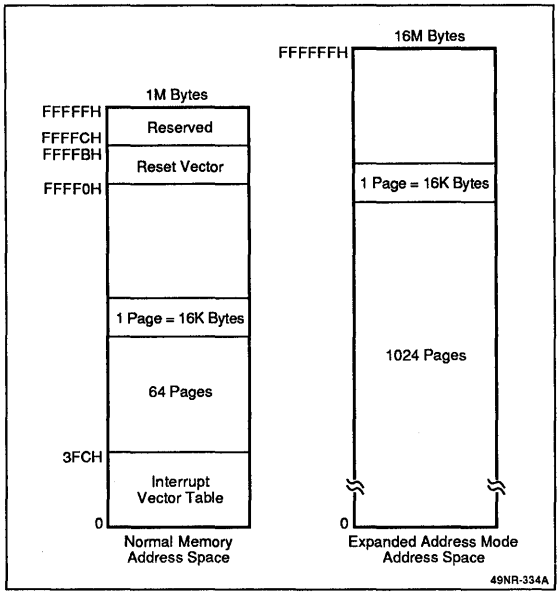
The I/O space is 64K bytes (16-bit address). The normal memory address space is 1M byte (20-bit address), and the expanded address space is 16M bytes (24-bit address). See figure 35. Expanded addressing is enabled or disabled using the BRKXA and RETXA instructions.

The memory space is accessed when an instruction uses a memory addressing mode. Memory addresses are calculated as described below. The I/O space can only be accessed through the IN, OUT, INM, and OUTM instructions.

Certain areas of the V53 address spaces (physical for normal mode and logical for expanded addressing mode) are reserved. . Memory addresses 0-3FCH are used for the interrupt vector table (figure 35) located in the interrupt operation section. Memory addresses FFFF0H-FFFFFH must contain a branch to boot code; PC, PFP, and PS are initialized at RESET to point to this area.

I/O addresses FF00H-FFFFH are reserved for the address translation registers and system control registers. The DMAU, TCU, ICU, and SCU sections each contain a block of registers with programmable base addresses. They may be located inside any 256-byte block in the I/O space. See figure 36.

Figure 35. Memory Address Space

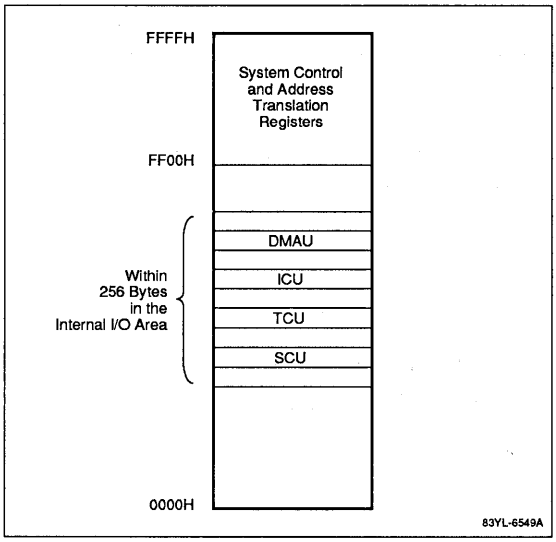


I/O Addresses

I/O devices can be referenced by 8-bit immediate addresses or by 16-bit addresses via the DW register. If I/O operations require other more complex addressing modes, the I/O devices must be placed in the memory address space (using memory-mapped I/O techniques). For memory-mapped I/O devices, there are no restrictions on instruction or addressing mode usage. However, the V53 will not automatically insert 6 clock cycles after

memory-mapped I/O operations; external logic must provide the necessary I/O device recovery time.

Figure 36. I/O Address Space

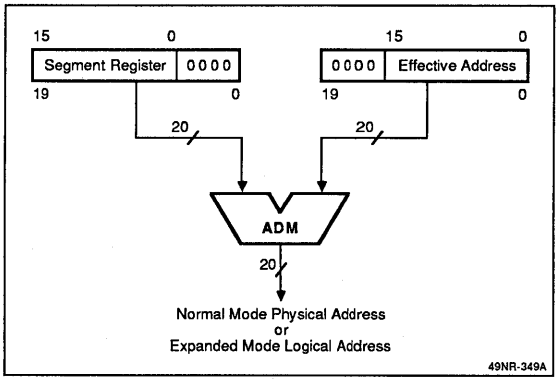


Normal Memory Addresses

The V53 is a 16-bit device with 16-bit registers. To allow a memory address space larger than 64K bytes, memory segmentation is used. The 1M-byte memory address space is divided into 64K-byte segments. Up to four segments can be in use at any given time. The base addresses of the four active segments (program segment, stack segment, data segment 0, and data segment 1) are contained in four 16-bit segment registers (PS, SS, DS0, and DS1, respectively). The 16-bit value in each register is the upper 16 bits of the 20-bit memory address. Thus, segments must start on 16-byte boundaries.

As described above, the V53 hardware generates a 16-bit effective address for each memory operation. This effective address is an offset into one of the four active segments. The actual 20-bit memory address is computed by adding the EA to the segment register value expanded with zeros to 20 bits. Figure 37 shows this process.

Figure 37. 20-Bit Address



If normal addressing mode is enabled, this 20-bit result is presented on the address bus during the bus cycle. If expanded addressing mode is enabled, this address is used as a logical address.

3f

Expanded Addresses

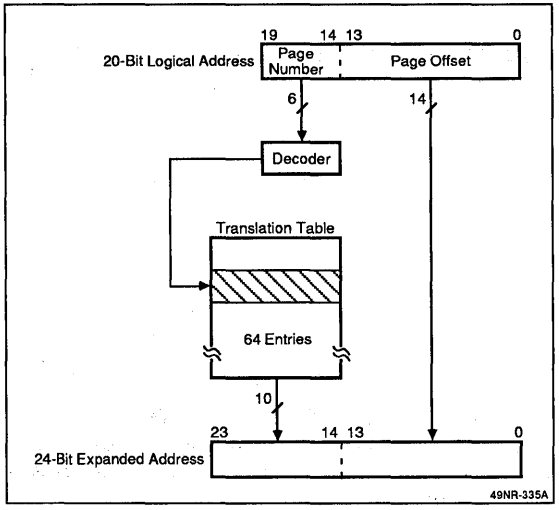
In the expanded addressing mode, the memory space is divided into 1024 pages (figure 35). Each page is 16K bytes. Each page of the normal 20-bit address space is mapped to a page in the expanded address space using a 64-entry address translation table. The table is made up of 64 page registers that reside in the I/O space.

The programming model of this mode is the same as for the normal mode. Address expansion is a layer added to the normal mode that is transparent to executing code. The program still sees a 20-bit contiguous logical memory address space, but the hardware sees 64 pages mapped into a set of 1024 physical pages.

The I/O space is not affected by the expanded addressing mode.

The address translation mechanism is shown in figure 38. The upper 6 bits of the logical 20-bit address select one of the entries in the address translation table, which supplies a 10-bit value. This value is substituted for the original 6 bits in the normal address to create a 24-bit expanded address.

Figure 38. Address Translation Mechanism



Address Expansion Registers

These are the page and XAM registers, accessed by the word IN and OUT instructions. Figure 39 shows page register usage and I/O addresses. The page registers contain the 10-bit physical page base address. The XAM register is a read-only status flag that indicates whether expanded addressing is enabled.

Unused data bits in the XAM register are read as 0. Expanded addressing must be disabled before accessing any of the page registers. That is, if expanded mode is enabled, the page registers cannot be accessed. This prevents an expanded mode task from accidentally modifying its memory map.

Figure 39. Address Expansion Registers

	Page Registers	
Logical Address A_{19}-A_{14}	**PGR Selected**	**PGR I/O Address**
0	PGR1	FF00
1	PGR2	FF02
2	PGR3	FF04
3	PGR4	FF06
:	:	:
63	PGR64	FF7E

XAM Register

	XA Flag
15 … 1	0

Operand Addressing Modes

For operand addressing, the V53 offers nine modes.

- Register
- Immediate
- Direct
- Register indirect
- Indexed
- Based
- Based indexed
- Bit
- Autoincrement/autodecrement

Register. The operand is in a V53 register pointed to by the instruction.

Immediate. The operand is in the instruction stream following the opcode of the instruction. This data will have been prefetched. Immediate data uses the V53 pipeline efficiently.

Direct. Immediate data in the instruction stream points directly to the operand. This data can be a 16-bit effective address or a bit field length of 4 bits.

Register Indirect. A 16-bit register (IX, IY, or BW) contains a 16-bit effective address.

Indexed. One or two bytes of immediate data are treated as a signed displacement that is added to the contents of a 16-bit index register (IX or IY) to obtain a 16-bit effective address.

Based. One or two bytes of immediate data are treated as a signed displacement that is added to the contents of a 16-bit base register (BP or BW) to form a 16-bit effective address.

Based Indexed. One or two bytes of immediate data are treated as a signed displacement that is added to two

16-bit registers (BP or BW and IX or IY) to form the effective address. This mode is useful for array addressing.

Bit. Used with NOT1, CLR1, or TEST1. A 4-bit immediate data value SET1 selects a bit in a 16-bit operand. For 8-bit operands, only 3 bits are used.

Autoincrement/Autodecrement. Some iterative operations (such as MOVBK or INS) will automatically increment or decrement index registers after each iteration. Specifically, IX is used in addressing a source pointer, and/or IY is used in addressing a destination pointer. After the operation, both will be incremented or decremented (according to the PSW DIR control flag) to point to the next operand in the array.

Instruction Addressing Modes

Instruction address modes are basically the same as the operand addressing modes, but the PC is always used in the register. These modes are used in control transfer instructions.

- Direct
- Relative
- Register
- Register indirect
- Indexed
- Based
- Based indexed

Direct. Four bytes of immediate data are taken as an absolute address and loaded directly into the PS and PC (and PFP).

Relative. One or two bytes of immediate data are a signed displacement that is added to the contents of the PC, and then placed in the PC (and PFP). This mode is useful to create position-independent code.

Register. The register selected by the instruction (AW, BW, etc.) contains an effective address, which is loaded into the PC (and PFP).

Register Indirect. An index register (IX, IY, or BW) points to a memory location that contains an effective address (short pointer) or a segment register value and the effective address (far pointer). This effective address is read from memory and loaded into the PS and/or PC (and PFP).

Indexed. One or two bytes of immediate data are a signed displacement added to the contents of a 16-bit index register (IX or IY) to form an effective address. This address is used to fetch another effective address from memory, which is then loaded into the PC (and PFP).

Based. One or two bytes of immediate date are a signed displacement added to the contents of a 16-bit base register (BP or BW) to form an effective address. This address is used to fetch another effective address from memory, which is then loaded into the PC (and PFP).

Based Indexed. One or two bytes of immediate data are a signed displacement added to the contents of two 16-bit registers (BP or BW and IX or IY) to form an effective address. This address is used to fetch another effective address from memory, which is then loaded into the PC (and PFP).

CPU Register Configuration

Program Counter (PC). The PC is a 16-bit register containing the effective address of the instruction currently being executed. The PC is incremented each time the instruction decoder accepts a new instruction from the prefetch queue. The PC is then loaded with a new value during execution of a branch, call, return, or break instruction, and during interrupt processing.

Segment Registers (PS, SS, DS0, DS1). There are four segment registers, each containing the upper 16 bits of the base address of a 64K logical segment. Since logical segments reside on 16-byte boundaries, the lower 4 bits of the base address are always zero. Normal 20-bit memory addresses are formed by adding the 16-bit effective address to the base address of one of the segments. During this operation, certain types of effective addresses will be paired with specific segment registers.

Segment Register	Default Offset
PS (program segment)	PFP
SS (stack segment)	SP, effective address
DS0 (data segment 0)	IX, effective address
DS1 (data segment 1)	IY

Program instructions will always be fetched from the program segment. Whenever the IY index register addresses an operand, the DS1 segment register will be used. DS0 is usually used with IX. Stack operations with the SP will always use the stack segment. For other effective addresses, the table above shows the default segment, but another segment may be selected by a segment override prefix instruction.

General-Purpose Registers (AW, BW, CW, DW). The four 16-bit general-purpose registers can be accessed as 16-bit or 8-bit quantities. When the AW, BW, CW, or DW destination is used, the register will be 16 bits. When AL, AH, BL, BH, CL, CH, DL, or DH is used, the register will be 8 bits. AL will be the low byte of AW and AH will be the high byte, etc.

Some operations require the use of specific registers.

Register	Operation
AW	Word multiplication/division, word I/O, data conversion
AL	Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
AH	Byte multiplication/division
BW	Translation
CW	Shift instructions, rotation instructions, BCD operations
DW	Word multiplication/division, indirect addressing I/O

Pointer (SP, BP) and Index Registers (IX, IY). These registers are used as base pointers and index registers when based, indexed, or based indexed addressing modes are used.

They may also be used as general-purpose registers for data transfer, arithmetic, and logical instructions. They can only be accessed as 16-bit registers.

Some operations use these registers in specific ways.

Register	Operation
SP	Stack operations
IX	Source pointer for block transfer, bit field, and BCD string operations
IY	Destination pointer for block transfer, bit field, and BCD string operations

Program Status Word (PSW). The program status word reflects the status of the CPU by six status flags and affects the operation of the CPU by three control flags.

PSW

15							8
1	1	1	1	V	DIR	IE	BRK

7							0
S	Z	0	AC	0	P	1	CY

Status Flags		Control Flags	
V	Overflow	DIR	Direction
S	Sign	IE	Interrupt enable
Z	Zero	BRK	Break
AC	Auxiliary carry		
P	Parity		
CY	Carry		

The DIR control flag determines whether address pointers are incremented (1) or decremented (0) for block (string) operations. IE enables interrupts (1) or disables interrupts (0). BRK enables (1) or disables (0) the single stepping trap (vector 1).

The PSW cannot be accessed directly as a 16-bit register. Specific instructions set/reset the control flags. When the PSW is pushed on the stack (as during interrupt processing), the PSW is set as shown in the diagram above.

Interrupt Operation

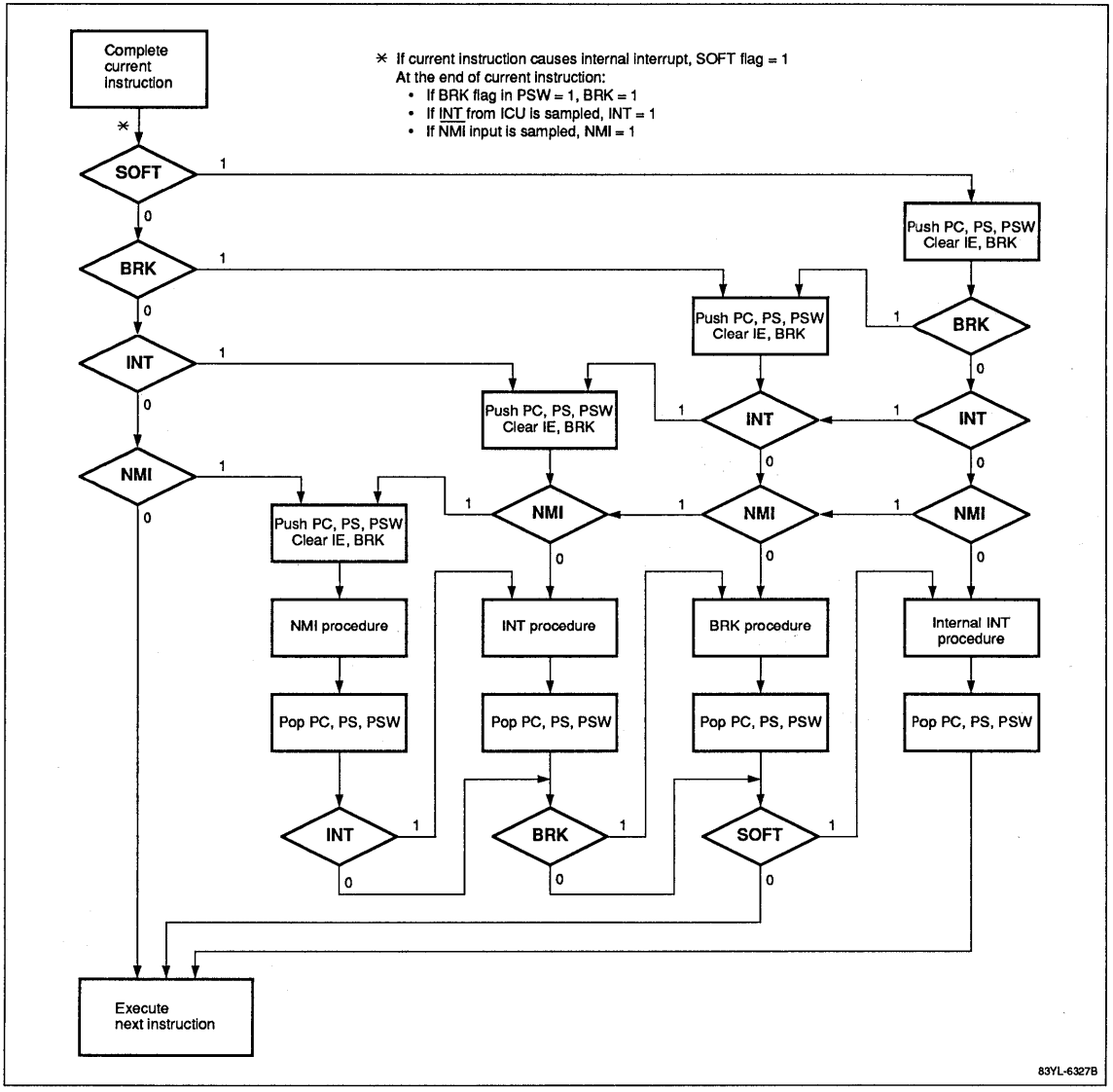
The interrupts supported by the V53 can be divided into two types: those generated by external interrupt requests and traps generated by software processing. Interrupts of each type are listed below.

- External Interrupts
  - NMI input (nonmaskable)
  - INTP0-INTP7 (maskable)
- Software Traps
  - Divide error during DIV or DIVU instruction
  - Array bound error during CHKIND
  - Single-step (PSW BRK flag = 1)
  - Undefined instruction
  - Coprocessor error
  - Coprocessor not connected
  - Break instructions (BRKV, BRK3, BRK imm8, BRKXA)

The eight INTP interrupts are handled by the interrupt control unit (ICU). The ICU prioritizes the INTPs and produces a single INT output, an internal signal that goes to the CPU interrupt logic. There the interrupt prioritization flow diagram (figure 40) is implemented. Interrupts are prioritized by the CPU as follows.

NMI (highest priority
INT
BRK flag
Other software interrupts and exceptions

Figure 40. Interrupt Prioritization Flow Diagram



3f

Interrupts are not accepted by the CPU at certain times. NMI, INT, and BRK flags are not accepted under the following conditions.

(1) Between execution of a MOV or POP that uses a segment register as an operand and the next instruction.

(2) Between a segment override prefix and the next instruction.

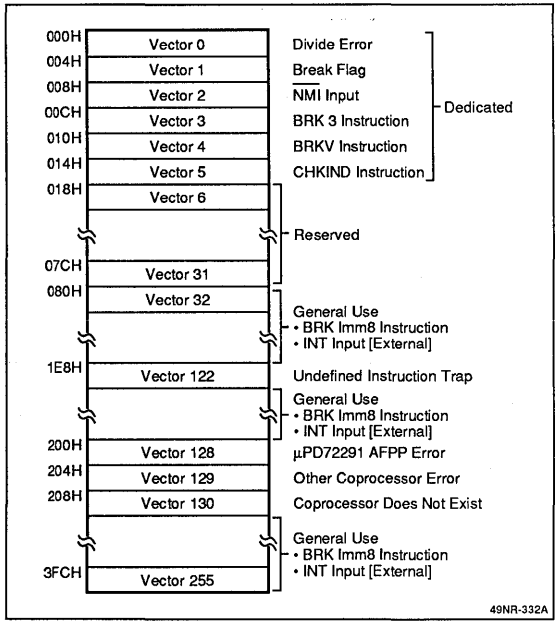
(3) Between a repeat or BUSLOCK prefix and the next instruction.

INT is not accepted when the PSW IE flag is 0, or between an RETI or POP PSW and the next instruction.

Once an interrupt has been accepted by the CPU, an interrupt service routine will be entered. The address of this routine is specified by an interrupt vector stored in the interrupt vector table (figure 41). For most interrupts, the vector used depends on what interrupt is being processed (e.g., NMI always uses vector 2). For INT and BRK imm8 interrupts, any vector may be used; the vector number is supplied by the ICU or an external device (such as a μPD71059) in the case of INT, or by immediate data in the case of BRK.

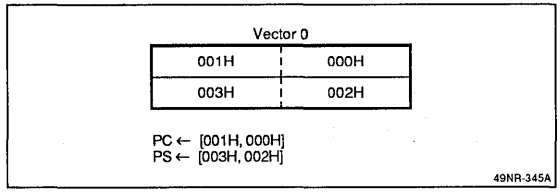
The interrupt vector table uses 1K bytes of memory at addresses 000H to 3FFH and stores up to 256 vectors.

Figure 41. Interrupt Vector Table



Each interrupt vector consists of four bytes. The two low bytes are loaded into the PC as the offset, and the two high bytes are loaded into the PS as the base address. See figure 42.

Figure 42. Interrupt Vector 0



Based on this format, the contents of each vector should be initialized at the beginning of the program. The basic mechanism for servicing an interrupt follows.

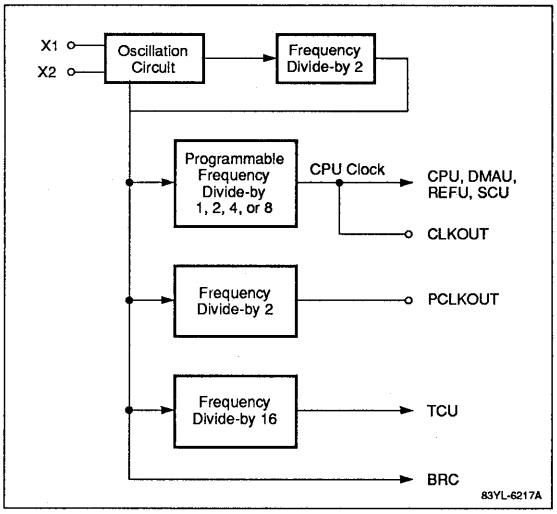
(SP – 1, SP – 2) ← PSW
(SP – 3, SP – 4) ← PS
(SP – 5, SP – 6) ← PC
SP ← SP – 6
IE ← 0, BRK ← 0
PS ← vector high bytes
PC ← vector low bytes

When an interrupt is accepted, two possible PC values could be saved. For some interrupts, the offset of the current instruction is saved. These interrupts are divide error, CHKIND, illegal opcode, μPD72291 FPP error, other coprocessor error, and CP not present. For the other interrupts (NMI, BRK flag, BRK instruction, or ICU interrupt), the offset of the next instruction is saved.

CLOCK GENERATOR (CG)

The clock generator (figure 43) is driven by a crystal connected to pins X1 and X2 or an oscillator connected to pin X1 with no connection at pin X2. The source frequency is divided to supply various clocks to internal units (CPU, DMAU, etc.) and to external devices at pins CLKOUT and PCLKOUT.

Figure 43. Clock Generator Diagram



BUS OPERATION

The V53 uses a synchronous bus interface. The X1 and X2 inputs provide a reference oscillator frequency for the internal clock generator, which supplies the main system clock to the other internal devices and to external devices via the CLKOUT pin. All V53 bus timings and instruction execution clock counts are specified relative to the CLKOUT signal. Bus cycles start on the falling edge of CLKOUT.

The V53's internal bus is a multimaster, shared bus. The CPU, DMAU, or REFU can all be bus masters. Each requests bus mastership from the bus arbitration unit (BAU). External devices can also request mastership of the bus using the HLDRQ input.

Bus Interface Unit (BIU)

The BIU contains the interface logic that allows the three internal bus masters (CPU, DMAU, and REFU) to control the external address, data, and control buses. The BIU also synchronizes the $\overline{BS8}$/BS16, $\overline{RESET}$, and $\overline{READY}$ inputs to the system clock. When a reset signal is accepted, the BIU asserts the RESOUT output.

Bus Arbitration Unit (BAU)

The BAU accepts and grants five different requests for bus mastership in the following priority order.

REFU demand (highest)
DMAU request
HLDRQ
CPU request
REFU request

The refresh unit is assigned both the highest and the lowest priorities. Normally, REFU requests are made, and if the bus is not granted, they are placed in a queue. Once the queue depth reaches seven requests, a refresh demand is made, and the BAU gives this the highest priority.

Bus Wait Function

When the bus is active and the BAU receives a higher priority request, the BAU will take away its grant to the current bus master. But the current master may not release the bus immediately. The BAU will wait until the current master takes away its request before granting the bus to the higher priority requester. This is called bus waiting.

For example, if an external device has been granted the bus via the HLDAK output, and the DMAU requests the bus (DMA is higher priority than HLDRQ), the V53 will deassert HLDAK but will not take the bus back until the external master deasserts HLDRQ. Note that the external master is not required to immediately release the bus back to the V53; the BAU will wait until HLDRQ is removed.

Usually a higher priority request will be granted quickly; for example, if a DMA request is accepted during T2 of a CPU bus cycle, the next bus cycle will usually be a DMA cycle. However, each internal bus master will hold onto the bus under certain circumstances.

The CPU will not let go of the bus as long as the $\overline{BUSLOCK}$ prefix is used, or until the current bus operation is completely finished (an unaligned or bus-sizing operation may take more than one bus cycle). Likewise, when it is in bus hold mode, the DMAU will not release the bus until all active DMA requests have been processed.

This mode should be used with care as it can result in DRAM refresh errors if the DMA takes a long time to complete. Note that bus hold mode is only available when DMAU is in μPD71071 compatibility mode; μPD71037 mode is always in bus release mode.

3f

External Bus Masters

At times, external bus masters will need to use the V53 bus. There are two methods provided for that purpose: hold request and DMA cascade. Up to five external bus masters can be connected to the V53.

Hold Request. The external bus master can request the bus using a hold request. Hold request is implemented using the HLDRQ and HLDAK signals. The V53 grants the bus by floating many of its outputs and asserting HLDAK to notify the external device that the bus is now free.

DMA Cascade. DMA cascade is very similar to hold request; the difference is that a DMARQ/$\overline{\text{DMAAK}}$ signal pair requests and grants the bus. While DMA cascade is meant to be used to connect additional DMA controllers, it can be used by any type of external bus master. Since there are four DMA channels, each of which can be in cascade mode, up to four external masters can be connected by DMA cascade.

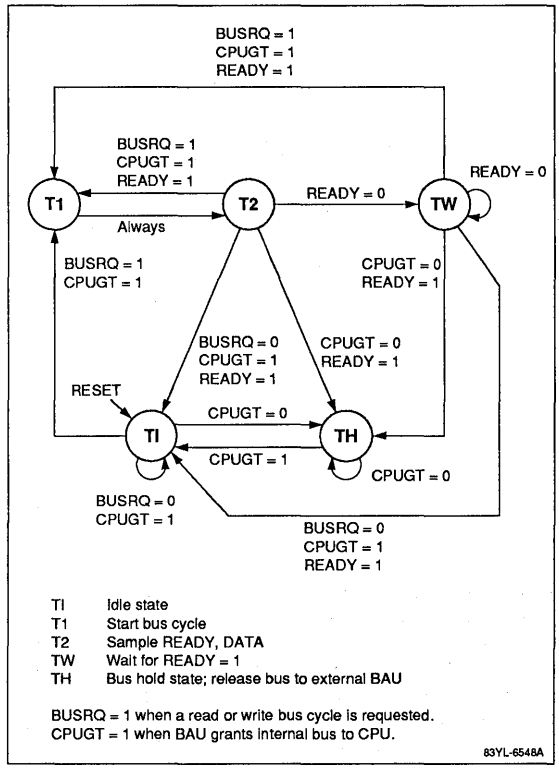
Bus Cycle Descriptions

Each of the internal bus masters uses the V53 bus interface in a different way: DMA bus cycles have a different structure than CPU bus cycles or REFU cycles. There are 18 different V53 bus cycles summarized previously in table 1.

CPU Bus Cycles

The bus state diagram for CPU cycles is shown in figure 44. CPU bus cycles are nominally two clock periods long, and may be extended by adding wait states using either the internal wait state generator or the external $\overline{\text{READY}}$ input.

Figure 44. CPU Bus State Diagram



The first state of every bus cycle is T1, and it is followed immediately by T2. $\overline{\text{READY}}$ is sampled on the rising (middle) edge of T2. If $\overline{\text{READY}}$ is not asserted, the next bus state will be the TW wait state. TWs will be inserted until $\overline{\text{READY}}$ is sampled low, after which the bus cycle will finish. TWs also will be inserted by the wait state generator, and the $\overline{\text{READY}}$ input is ignored until all TWs programmed in the wait state have been inserted. The dynamic bus sizing input, $\overline{\text{BS8}}$/BS16, is sampled at the same time as $\overline{\text{READY}}$.

Note that dynamic bus sizing is only implemented for CPU cycles; DMAU or REFU cycles do not use this input.

Address and bus status are output after the leading edge of T1, and maintained until after the cycle is completed. A strobe, BCYST, is asserted during T1 to indicate the beginning of a bus cycle. BCYST is output following the leading edge of T1 and deasserted after the leading edge of T2.

Write data is driven on D_0-D_{15} following the rising (middle) edge of T1, and maintained until after the rising edge of T2 or the last TW. Read data is sampled on the trailing

edge of T2 or the last TW state. A strobe, $\overline{DSTB}$, gives the status of the V53 data bus. $\overline{DSTB}$ is asserted after the rising (middle) edge of T1. $\overline{DSTB}$ is deasserted after the rising edge of T2 or the last TW for a write cycle, and after the trailing edge of T2 or the last TW for a read cycle.

I/O cycles are identical to memory cycles except for the encoding of the bus status lines. However, six idle states are inserted after every I/O bus cycle to provide a recovery time for the I/O devices.

Dynamic Bus Sizing for CPU Cycles

The V53 supports dynamic bus sizing for CPU cycles. On a cycle-by-cycle basis, the width of the data bus can be changed from 16 to 8 bits. This simplifies connection with 8-bit I/O devices that may have internal registers at consecutive byte addresses. Other 16-bit CPUs require two ROMs for startup code, but the V53 dynamic bus sizing makes it possible to use a single 8-bit wide ROM.

External logic requests an 8-bit data path by driving $\overline{BS8}$/BS16 low in time for the V53 to sample it on the rising edge of T2 (or TW). The V53 will perform an additional cycle if needed to finish the operation in byte-wide pieces.

If the bus operation is already 8 bits wide, no further bus cycles will occur (refer to tables 3 and 4). For a read cycle, the data will be sampled on D_7-D_0. For a write cycle to an even address, data will be driven on D_7-D_0. On all byte writes to an odd address, the V53 will put the byte data on both the upper and lower data buses so that the write data will be on D_7-D_0 as well as D_{15}-D_8.

If the bus operation is 16-bit, two bus cycles will be required. The first one, in which $\overline{BS8}$/BS16 is sampled low, will handle the low byte. The second cycle will take the form of a byte read or write using D_7-D_0.

3f

Table 3. Write Cycle Bus Sizing

					16-Bit Bus ($\overline{BS8}$/BS16 = 1)		8-Bit Bus ($\overline{BS8}$/BS16 = 0)	
Type	**Address**	**A_0**	**$\overline{UBE}$**	**Cycle**	**D_{15}-D_8**	**D_7-D_0**	**D_{15}-D_8**	**D_7-D_0**
Byte	Even	0	1	1st	Invalid	Lower	Invalid	Lower
	Odd	1	0	1st	Lower	Lower	Lower	Lower
Word	Even	0	1	1st	Upper	Lower	Upper	Lower
		1	0	2nd	Not needed for 16-bit bus		Upper	Upper
	Odd	1	0	1st	Lower	Lower	Lower	Lower
		0	1	2nd	Lower	Upper	Lower	Upper

Note: Lower = low-order byte; Upper = high-order byte

Table 4. Read Cycle Bus Sizing

					16-Bit Bus ($\overline{BS8}$/BS16 = 1)		8-Bit Bus ($\overline{BS8}$/BS16 = 0)	
Type	**Address**	**A_0**	**$\overline{UBE}$**	**Cycle**	**D_{15}-D_8**	**D_7-D_0**	**D_{15}-D_8**	**D_7-D_0**
Byte	Even	0	1	1st	Not used	Lower	Not used	Lower
	Odd	1	0	1st	Lower	Not used	Not used	Lower
Word	Even	0	1	1st	Upper	Lower	Not used	Lower
		1	0	2nd	Not needed for 16-bit bus		Not used	Upper
	Odd	1	0	1st	Lower	Not used	Not used	Lower
		0	1	2nd	Not used	Upper	Not used	Upper

Note: Lower = low-order byte; Upper = high-order byte

CPU Bus Cycle Types. There are many types of CPU bus cycles (shown previously in table 2). They comprise read, write, and acknowledge cycles.

CPU Read Cycles. There are six CPU read cycles: memory, external I/O, internal I/O coprocessor, coprocessor data reads, and instruction fetch. All have the general timing described previously. Coprocessor reads access the internal registers of an external coprocessor. Coprocessor data reads transfer data from memory to an internal coprocessor register. Instruction fetches fill the V53's 8-byte instruction queue from the memory space. I/O and memory reads transfer data to the V53 from an

internal or external I/O device or a memory location. During internal I/O reads, the $\overline{IORD}$ and $\overline{BUFEN}$ outputs are not asserted.

Dynamic bus sizing is ignored during internal I/O read cycles, and is not recommended for coprocessor data read cycles. The wait state generator does not affect internal I/O reads or coprocessor reads. $\overline{READY}$ is used for all CPU read cycles.

CPU Write Cycles. There are five types of CPU writes. Memory writes transfer data from the V53 to a memory location. External and internal I/O writes transfer data from the V53 to external or internal I/O devices. During internal I/O writes, the $\overline{IOWR}$ and $\overline{BUFEN}$ outputs are not asserted. Coprocessor data writes transfer data from an external coprocessor to a memory location. Coprocessor writes transfer data from the V53 directly to a coprocessor internal register.

Dynamic bus sizing is ignored during internal I/O read/writes, and is not recommended for coprocessor data write cycles. The wait state generator does not affect internal I/O writes or coprocessor writes. $\overline{READY}$ is used for all CPU write cycles.

Interrupt Acknowledge Cycles. The CPU interrupt acknowledge operation takes two consecutive bus cycles. The first cycle freezes the state of the internal interrupt control unit (ICU) and any external slave μPD71059 interrupt controllers. The second bus cycle reads an 8-bit vector number on D_7-D_0, supplied by either the ICU or an external slave. This vector number is then used by the CPU as an index into the interrupt vector table to select an interrupt handler. The $\overline{BUSLOCK}$ output is asserted for the first cycle, and remains asserted until after the second to guarantee that no other bus master will take control of the bus until the interrupt has been accepted.

There are two types of interrupt acknowledge cycles produced by the V53: a master and a slave. The $\overline{INTAK}$ output is asserted for both types, and should be connected to the interrupt acknowledge inputs of all slave devices. The master cycle is used for the first $\overline{INTAK}$ to both internal and external ICUs, and the second $\overline{INTAK}$ to the internal ICU. The slave cycle is used only for the second $\overline{INTAK}$ to an external slave device.

During the slave cycle, the address of the slave device to be used is presented on A_2-A_0. These address lines should be buffered and then connected to the slave address inputs of the external ICUs. Buffering is necessary because the slave address pins of external devices might be in an output state on power-up, producing a bus conflict on A_2-A_0 if they are connected directly.

Dynamic bus sizing is ignored during the interrupt acknowledge cycles. Wait states can be inserted by the internal wait state generator or by the $\overline{READY}$ input.

Halt Acknowledge Cycle. When the CPU executes a HALT instruction, a halt acknowledge bus cycle is issued to notify external logic that the V53 is entering a standby mode. This cycle is always two clocks long; $\overline{READY}$ is ignored and $\overline{DSTB}$ is not asserted. The V53 has several standby modes.

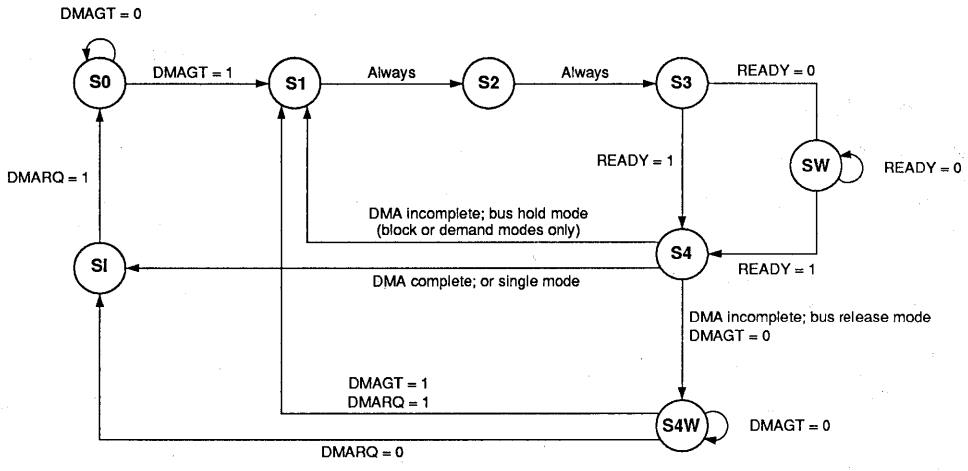
DMA Unit Bus Cycles

Figure 45 shows the bus state diagram for DMA bus cycles. There are eight different states. When the DMAU is idle, it is in state SI. In this state, it is continually sampling the four DMARQ inputs. When a request is detected, the DMAU requests use of the V53 bus from the BAU, and enters state S0. It remains in S0 until the BAU grants the bus to the DMAU, at which point the actual DMA bus cycle starts with state S1. Addresses and control status are output along with BCYST and $\overline{DMAAK}$.

DMA bus cycles are nominally four clocks long, but they can be stretched by the internal wait state generator or $\overline{READY}$. S1 always changes to S2 and then to S3. Memory and I/O strobes are asserted during S2, S3, and SW. $\overline{READY}$ is sampled during S2 for use during S3. If waits are inserted, the SW state is entered. Control stays in that state until no more waits are desired. If no waits are inserted, S3 moves to S4 and the current cycle is over.

Depending on the DMA mode, another DMA cycle might be ready to start immediately (e.g., in burst mode), or another DMA request input may now be asserted. During S4, a decision is made whether to begin another DMA cycle at S1, to return to SI, or to enter the bus wait state S4W. The latter transition will be made if another DMA cycle is ready to start but the BAU has taken the bus away from the DMAU. In S4W, the DMAU releases the bus, but is ready to begin as soon as the bus is granted again and the DMA request is still pending.

Figure 45. DMAU Bus State Diagram



SI	Idle state; sample DMA requests.
S0	Wait for BAU to grant bus to DMAU.
S1	Output DMA memory address.
S2	Assert strobes; sample READY, END.
S3	Check READY.
S4	End of DMA cycle; check DMA completion.
S4W	Bus wait state; release bus to BAU.
SW	Wait for READY = 1.

83YL-6330B

DMA Read Cycle. The DMAU performs "fly-by" DMA. During one DMA read bus cycle, data moves from the source address in memory to the destination I/O device. The V53 puts the memory address on A_{23}-A_0, and asserts $\overline{MRD}$. At the same time, $\overline{IOWR}$ is asserted. Memory will drive the DMA data onto the bus, and the $\overline{IOWR}$ signal will latch the data into the I/O device. $\overline{DMAAK}$ should be used to control chip select at the I/O device. Since the V53 does not use the data, $\overline{BUFEN}$ is not asserted during DMA bus cycles.

DMA Write Cycle. The DMAU performs "fly-by" DMA. During one DMA write bus cycle, data moves from the source I/O device to the destination address in memory. The V53 puts the memory address on A_{23}-A_0, and asserts $\overline{MWR}$. At the same time, $\overline{IORD}$ is asserted. The I/O device will drive the DMA data onto the bus, and the $\overline{MWR}$ signal will latch the data into memory. $\overline{DMAAK}$ should be used to control chip select at the I/O device. Since the V53 does not use the data, $\overline{BUFEN}$ is not asserted during DMA bus cycles

Note that when DMA writes are made to DRAM, it may be necessary to generate a delayed CAS strobe because the data is being supplied by an I/O device that may have long access time. The write data may not be valid when the normal CAS signal is asserted.

Dynamic bus sizing cannot be used for DMA operations. The internal wait state generator and $\overline{READY}$ can be used to stretch the cycle.

DMA Cascade. During DMA cascade, the DMA state machine releases the V53 bus to an external bus master such as a μPD71071 or μPD71037 DMA controller. $\overline{DMAAK}$ is connected to the HLDAK input of the external device. $\overline{DMAAK}$ will stay asserted until the external master deasserts DMARQ. If the V53 BAU needs to give the bus to a higher priority bus master, $\overline{DMAAK}$ will be deasserted. The external bus master is expected to then deassert the DMARQ input, at which point the bus will be given to the higher priority bus master.

Refresh Unit Bus Cycles

The refresh unit performs memory read cycles from consecutive memory addresses. These bus cycles are the same as CPU memory read cycles, except that the REFRQ output is asserted. External logic should use the REFRQ logic to enable RAS for all memory banks, regardless of the address decoding scheme, so that all banks are refreshed.

Dynamic bus sizing cannot be used during refresh operations. The internal wait state generator and $\overline{\text{READY}}$ can be used to stretch the cycle.

SYSTEM INTERFACE

System Memory Access Time

Table 5 shows the system memory access time required for 12.5-MHz and 16-MHz V53 systems to run with zero, one, two, and three wait states. This is the time from when the address bus is valid to when the external system must present the read data on the data bus. These numbers are based on the preliminary ac timing given in this document and are subject to change.

Table 5. Performance vs. Wait States

	12.5 MHz			16 MHz		
Number of Wait States	Memory Cycle Time (ns)	System Access Time (ns)	Relative Performance (%)	Memory Cycle Time (ns)	System Access Time (ns)	Relative Performance (%)
0	160	113	78	125	78	100
1	240	193	64	187.5	140.5	82
2	320	273	52	250	203	67
3	400	353	43	312.5	265.5	56

Note: Performance is relative to the 0 wait state, 16 MHz.

Wait States

Table 5 also illustrates the effect of wait states on performance. The V53 CPU overlaps bus interface operations in time with instruction execution. This greatly reduces the effect of wait states on performance. Each bus cycle is nominally two clocks long, while the minimum instruction is two clocks with many instructions taking longer.

There is some idle bus time when the CPU is processing a long instruction and the prefetch queue is full. Wait states can often fill these idle states. However, adding wait states to bus cycles reduces the bus bandwidth available for other bus masters, such as DMA controllers. This is because some of the idle time that would have been available to them is used for CPU cycles.

Note that in all cases, a 16-MHz V53 with N+1 wait states is faster than a 12.5-MHz device with N wait states but slower memory.

Note also that the numbers are for comparison only. Different results will be obtained for other program mixes.

Interfacing the μPD72291 AFPP

The AFPP is a very-high-performance floating-point coprocessor able to process more than 530K floating-point operations per second at 16 MHz.

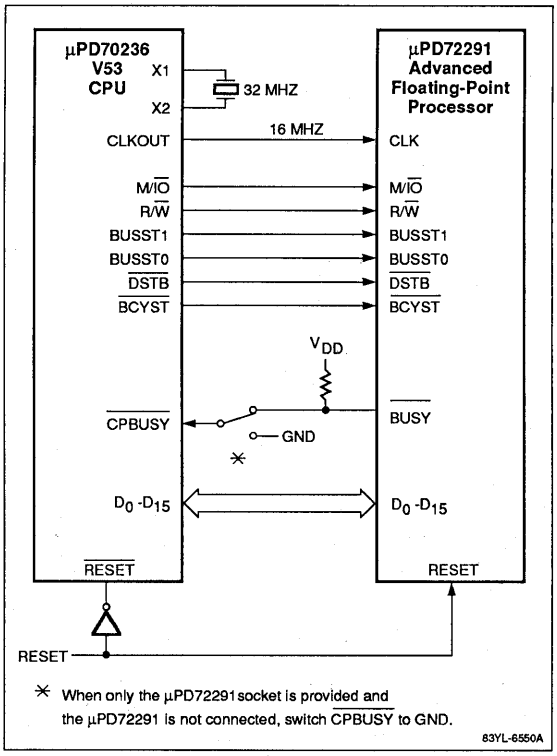
The AFPP is programmed as an extension of the V53 instruction set. The AFPP executes floating-point operations, computes transcendental functions, and performs vector multiplications.

AFPP instructions use the FP01 and FP02 formats. When one of these opcodes is encountered and an AFPP is connected, a coprocessor protocol routine is entered. The V53 computes any effective addresses required, reads or writes the operands for the AFPP, and tells the AFPP which operation should be performed.

The AFPP responds by asserting its BUSY output when it starts the operation. The V53 will not start another AFPP operation until BUSY is deasserted, but may execute CPU instructions. When BUSY is deasserted, the V53 will transfer the AFPP status to the AW register.

Figure 46 shows how to connect a V53 CPU to a μPD72291 AFPP. The CPU reads and writes status and commands to the AFPP using coprocessor read and write cycles, which always take two clocks. AFPP operands are written using coprocessor memory write/read cycles, which always require one wait state. The V53 automatically inserts one wait state into these cycles so no external wait generation logic is required.

Figure 46. Connections Between the V53 and μPD72291



On reset, $\overline{CPBUSY}$ is sampled. If it is low, the V53 assumes that a coprocessor is connected. CPERR is also sampled to determine what kind of coprocessor is connected as follows.

$\overline{CPBUSY}$	CPERR	Coprocessor Connected
1	x	None
0	0	μPD72291
0	1	Another kind

AFPP memory operands must always begin on an even address and may not reside in 8-bit wide memory. Dynamic bus sizing may not be used for AFPP operands.

SYSTEM CONTROL I/O

On-Chip Control Registers

The V53 provides many on-chip control registers. Some of these reside in the 256-byte system I/O area (I/O space addresses FF00 to FFFF). These are shown in table 6. Other registers reside in small blocks associated with an on-chip peripheral (addresses are programmable). There are register blocks for DMAU, TCU, ICU, and SCU. The base addresses for these register blocks are programmable using the OPHA, DULA, TULA, and SULA registers in the system I/O area. See figure 47.

Figure 47. Peripheral Relocation

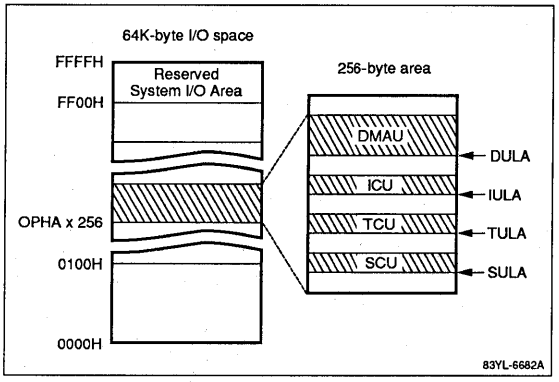


Table 6. System I/O Area

I/O Address	Register Name	Figure
FFFFH	Reserved	—
FFFEH	SCTL	48
FFFDH	OPSEL	49
FFFCH	OPHA	50
FFFBH	DULA	50
FFFAH	IULA	50
FFF9H	TULA	50
FFF8H	SULA	50
FFF7H	Reserved	—
FFF6H	WCY4	61
FFF5H	WCY3	60
FFF4H	WCY2	59
FFF3H	WMB1	55
FFF2H	RFC	62
FFF1H	SBCR	110
FFF0H	TCKS	51
FFEFH-FFEEF	Reserved	—
FFEDH	WAC	56
FFECH	WCY0	57
FFEBH	WCY1	58
FFEAH	WMB0	54
FFE9H	BRC	52
FFE8H	Reserved	—
FFE7H-FFE2H	Reserved	—

3f

Table 6. System I/O Area (cont)

I/O Address	Register Name	Figure
FFE1H	BADR	102
FFE0H	BSEL	103
FFDFH-FF81H	Reserved	—
FF80H	XAM (Read Only)	39
FF7FH-FF00H	PGR64-PGR1	39

Note: All registers are Read/Write except XAM.

System Control Register (SCTL)

The SCTL register (figure 48) selects the 8-bit or 16-bit boundary of an internal peripheral relocation address. It also sets the internal DMAU in the μPD71071 or μPD71037 modes. In μPD71037 mode, SCTL controls propagation of carry from A_{15} to A_{16} or from A_{19} to A_{20}. SCTL selects the baud rate generator or TOUT as the SCU clock.

Figure 48. System Control Register (SCTL)

—	—	—	SC	CE1	CE0	DMAM	IOAG

7 0

Address FFFEH

SC	**SCU Input Clock**
0	TOUT1
1	From baud rate generator
CE1	**Carry to A_{20} in μPD71037**
0	Does not propagate
1	Propagates
CE0	**Carry to A_{16} in μPD71037**
0	Does not propagate
1	Propagates
DMAM	**DMAU Mode**
0	μPD71071
1	μPD71037
IOAG	**Internal I/O Address**
0	Even or odd (16-bit boundary)
1	Contiguous (8-bit boundary)

On-Chip Peripheral Selection Register (OPSEL)

The OPSEL registers (figure 49) controls the V53 internal peripherals. Any of the four peripherals (DMAU, TCU, ICU, or SCU) can be independently enabled or disabled by setting the appropriate OPSEL bit.

Figure 49. On-Chip Peripheral Selection Register (OPSEL)

—	—	—	—	SS	TS	IS	DS

7 0

SS	**SCU Operation**
0	Disabled
1	Enabled
TS	**TCU Operation**
0	Disabled
1	Enabled
IS	**ICU Operation**
0	Disabled
1	Enabled
DS	**DMAU Operation**
0	Disabled
1	Enabled

Internal Peripheral Relocation Registers

The five internal peripheral registers fix the I/O addresses of the DMAU, ICU, TCU, and SCU. Register OPHA fixes the high-order byte of the 16-bit I/O addresses. Registers DULA, IULA, TULA, and SULA select the low-order byte of the I/O addresses for the DMAU, ICU, TCU, and SCU peripherals, respectively.

The formats of the individual internal peripheral registers are shown in figure 50. Since address checking is not performed, two peripheral I/O address spaces should not be overlapped.

Figure 50. Internal Peripheral Relocation Registers

OPHA Register

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

7 I/O Address FFFCH 0

DULA Register

IOAG = x*	A_7	A_6	A_5	A_4	–	–	–	–
IOAG = 0 #	A_7	A_6	A_5	A_4	–	–	–	–
IOAG = 1 #	A_7	A_6	A_5	–	–	–	–	A_0

7 I/O Address FFFBH 0

* μPD71071 mode
\# μPD71037 mode

IULA Register

IOAG = 0	A_7	A_6	A_5	A_4	A_3	A_2	–	–
IOAG = 1	A_7	A_6	A_5	A_4	A_3	–	–	A_0

7 I/O Address FFFAH 0

TULA Register

IOAG = 0	A_7	A_6	A_5	A_4	A_3	A_2	–	–
IOAG = 1	A_7	A_6	A_5	A_4	A_3	–	–	A_0

7 I/O Address FFF9H 0

SULA Register

IOAG = 0	A_7	A_6	A_5	A_4	A_3	A_2	–	–
IOAG = 1	A_7	A_6	A_5	A_4	A_3	–	–	A_0

7 I/O Address FFF8H 0

The IOAG bit of the SCTL register changes how the DULA, IULA, TULA, and SULA registers are used. When IOAG = 1, the DAMU, ICU, TCU, and SCU registers are on contiguous bytes. When IOAG = 0, each of these byte-wide registers is put on a word boundary. Bit A_0 selects the low or high byte of the word. This allows code written for a 16-bit system to be ported to a V53 design with no modifications. Because the DMAU registers in μPD71071 mode are 16-bit, the IOAG bit in figure 50 is noted as "x" (don't care).

Timer Clock Selection Register (TCKS)

The TCKS register (figure 51) selects the clock source for the timer/counters as well as the divisor for the internal clock prescaler. The clock source for each timer/counter is independently selected from an internal clock (figure 43) or an external clock source (TCLK).

The frequency of the internal clock selected by bits 2, 3, and 4 is programmable. The PS bits allow the clock to be set to the external oscillator frequency divided by 4, 8, 16, or 32.

3f

Figure 51. Timer Clock Selection Register (TCKS)

–	–	–	CS2	CS1	CS0	PS

7 I/O Address FFF0H 0

CS2	Clock Input to TCT2
0	Internal clock
1	TCLK pin
CS1	**Clock Input to TCT1**
0	Internal clock
1	TCLK pin
CS0	**Clock Input to TCT0**
0	Internal
1	TCLK pin
PS	**Prescale Divisor of External Oscillator**
0 0	4
0 1	8
1 0	16
1 1	32

Baud Rate Counter (BRC)

The BRC (figure 52) is an 8-bit, frequency-division counter for the dedicated baud rate generator. It sets the value by which an internal frequency is to be divided to provide the SCU with its baud rate clock.

Figure 52. Baud Rate Counter (BRC)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

7 I/O Address FFE9H 0

Table 7 illustrates the relationship between the baud rate and the value set in the BRC.

Table 7. Baud Rate Setting by BRC

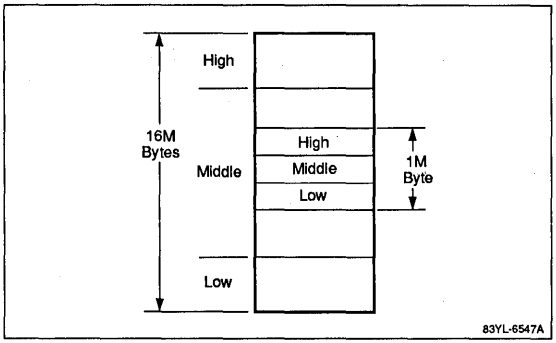
Oscillation frequency	24.576 MHz		29.4912 MHz	
Oscillation frequency ÷ 2	12.288 MHz		14.7456 MHz	
Baud rate factor (÷)	16	64	16	64
Internal frequency	0.768	0.192	0.9216	0.2304
Baud Rate	**Number of Counts Set In BRC**			
1200	—	160	—	192
2400	—	80	—	96
4800	160	40	192	48
9600	80	20	96	24
19,200	40	10	48	12
38,400	20	5	24	6

WAIT CONTROL UNIT

The wait control unit (WCU) inserts from 0 to 7 wait states (TW) into a bus cycle to compensate for the varying access times of different memory and I/O devices. Each wait state is equivalent to one CPU clock cycle. The number of wait states can be individually programmed for CPU, DMAU, REFU, INTAK, and external I/O cycles. The INTAK cycles can be programmed for 2-7 wait states.

For memory accesses, the address space is divided into a total of six sections (labeled High, Middle, and Low in figure 53). A different number of wait states can be programmed for each section, allowing much flexibility in the system design. The WCU works with the external $\overline{\text{READY}}$ input. After the proper number of TWs have been inserted into the bus cycle, $\overline{\text{READY}}$ will be sampled, and wait states will be inserted until it is asserted.

Figure 53. Memory Space Division



The WCU can insert waits into memory or external I/O cycles, but not into coprocessor, internal I/O, or halt acknowledge cycles.

Eight system I/O registers (figures 54-61) control the WCU. They are the wait state memory boundary registers (WMB0 and WMB1), the WCU address control register (WAC), and the wait state cycle count registers (WCY0-WCY4).

Memory Boundary Registers (WMB0, WMB1)

The WMB0 register divides the entire 16M-byte address space into three sections. The ELMB and EUMB fields specify the size of the upper and lower memory blocks. The middle block is the area left in between. The WCY0 and WCY1 registers specify the wait states of each expanded memory block.

In addition to dividing expanded memory, a specific 1M-byte memory area can also be partitioned into three blocks for wait state generation. The WAC register determines which 1M-byte area is referenced. The WMB1 register divides this area into three blocks in the same manner as described above for WMB0. Registers WCY2 and WCY3 specify the wait states for each block and also I/O.

Figure 54. Memory Boundary Register 0 (WMB0)

–	ELMB	–	EUMB

7 I/O Address FFEAH 0

ELMB/EUMB	Memory Block Size (Bytes)
000	1M
001	2M
010	3M
011	4M
100	5M
101	6M
110	7M
111	8M

Figure 55. Memory Boundary Register 1 (WMB1)

–	LMB	–	UMB

7 I/O Address FFF3H 0

LMB/UMB	Memory Block Size (Bytes)
000	32K
001	64K
010	96K
011	128K
100	192K
101	256K
110	384K
111	512K

Figure 56. WCU Address Control Register (WAC)

				UWA

7 I/O Address FFEDH 0

UWA = Upper 4 bits of expanded address specifying a 1M-byte memory space

Wait State Cycle Count Registers (WCY0-WCY4)

Each WCY register has one or two 3-bit fields that set the number of waits for a particular kind of cycle or the number of waits to be inserted into cycles during which certain memory blocks are accessed.

(1) WCY0 and WCY1 (figures 57 and 58) pertain to the 16M-byte memory space set by the WMB0 register.

(2) WCY2 and WCY3 (figures 59 and 60) pertain to the 1M-byte memory space set by the WMB1 register.

(3) Also, the IOW field of WCY3 sets the number of waits for external I/O cycles and interrupt acknowledge cycles.

(4) The waits set by WCY3 cannot be inserted into the internal I/O area read/write cycle.

(5) WCY4 (figure 61) sets the number of waits for DMA cycles and refresh cycles.

After RESET, the WCY registers are set to all 1s, thereby inserting seven waits into all cycles. This allows the use of slow ROMs. Initialization code must set the WCY registers to their values.

Figure 57. WCY0 Register

–	–	–	–	–	EUMW

7 I/O Address FFECH 0

EUMW	*Wait States
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

* Upper section of 16M-byte memory space

Figure 58. WCY1 Register

–	EMMW	–	ELMW

7 I/O Address FFEBH 0

EMMW/ELMW	*Wait States
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

* Middle and lower sections of 16M-byte memory space

Figure 59. WCY2 Register

–	MMW	–	LMW

7 I/O Address FFF4H 0

MMW/LMW	*Wait States
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

* Middle and lower sections of 1M-byte memory space

3f

Figure 60. WCY3 Register

—	IOW	—	UMW

7 I/O Address FFF5H 0

	Wait States	
IOW	**Ext I/O Cycles**	**Int Ack Cycles**
000	0	2
001	1	3
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7
UMW	***Wait States**	
000	0	
001	1	
010	2	
011	3	
100	4	
101	5	
110	6	
111	7	

* Upper section of 1M-byte memory space

Figure 61. WCY4 Register

—	DMAW	—	RFW

7 I/O Address FFF6H 0

DMAW/RFW	***Wait States**
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

* DMA cycle or refresh cycle.

REFRESH CONTROL UNIT

The refresh control unit (REFU) refreshes external dynamic devices by periodically performing a memory read cycle from consecutive, incrementing addresses. A 16-bit counter provides the refresh address. The upper bits (A_{23}-A_{16}) are low during refreshes. Each refresh bus cycle has two wait states inserted, so that it will be a minimum of 4 clocks long. Refresh cycles can be distinguished from other memory reads by the assertion of the REFRQ output or by the bus status code.

If the V53 is busy when it is time to perform a refresh, the refresh request is placed in a refresh queue until the bus is no longer busy. Normally, the REFU has the lowest bus priority. However, after seven refreshes are queued, the REFU is given the highest bus priority. The REFU gets control of the bus, performs a burst of four refreshes, and then falls back to the lowest priority. This refresh queue ensures that refresh cycles are not lost even when the V53 is busy for long periods of time.

Refresh Control Register (RFC)

The RFC (figure 62) controls the refresh control unit. The RE bit enables or disables the REFU. The refresh interval is set by the RTM field by choosing refresh interval factor N, which determines how many CPU clock cycles elapse between refreshes.

$$\text{Refresh interval} = 16 \times N \times t_{CYC}$$

With a 16-MHz CPU clock, this allows a range of intervals from 1 to 32 μs. After $\overline{\text{RESET}}$, N will be 9, which gives an interval of 9 μs.

Since the V53 may operate with either 8- or 16-bit memory devices, the refresh address can be incremented by 1 (for 8-bit memory) or by 2 (for 16-bit memory). The RDB8 bit in the RFC makes the selection. In the word mode, $\overline{\text{UBE}}$ is always low (active) for refresh cycles. In the byte mode, $\overline{\text{UBE}}$ is asserted only for refreshes to an odd address.

Figure 62. Refresh Control Register (RFC)

RE	RDB8	—	RTM

7 I/O Address FFF2H 0

RE	**Refresh**
0	Disable
1	Enable
RDB8	**TCU Clock for Channel 2**
0	Increment by 2 ($\overline{\text{UBE}}$ = low level)
1	Increment by 1 ($\overline{\text{UBE}}$ = high level for even addresses and low level for odd-addresses)
RTM	**Refresh Interval Factor N**
00000	1
00001	2
00010	3
00011	4
00100	5
⋮	⋮
11110	31
11111	32

TIMER/COUNTER UNIT

The timer/counter unit (TCU) provides a set of three independent 16-bit timer/counters. Each timer has an individual output and gate control input. The clock source for each channel is set individually to either the

prescaled CPU clock or the external TCLK. TOUT1 is also internally connected to supply the baud rate clock to the SCU. Figure 63 is the TCU block diagram.

The TCU has the following features.

- Three 16-bit timer/counters
- Six programmable count modes
- Binary/BCD counting
- Multiple latch command
- Count latch command
- Choice of two clock sources
- 16-MHz operation
- Functionally compatible with μPD71054 (8254)

Because $\overline{\text{RESET}}$ leaves the TCU in an uninitialized state, each timer/counter must be initialized by specifying an operating mode and a count. Once programmed, a timer/counter will continue to operate in that mode until another mode is selected. When the count has been written to the counter and transferred to the down counter, a new count operation starts. Both the current count and the counter status can be read while count operations are in progress. Figure 64 is a flow diagram for TCU operations.

Figure 63. TCU Block Diagram

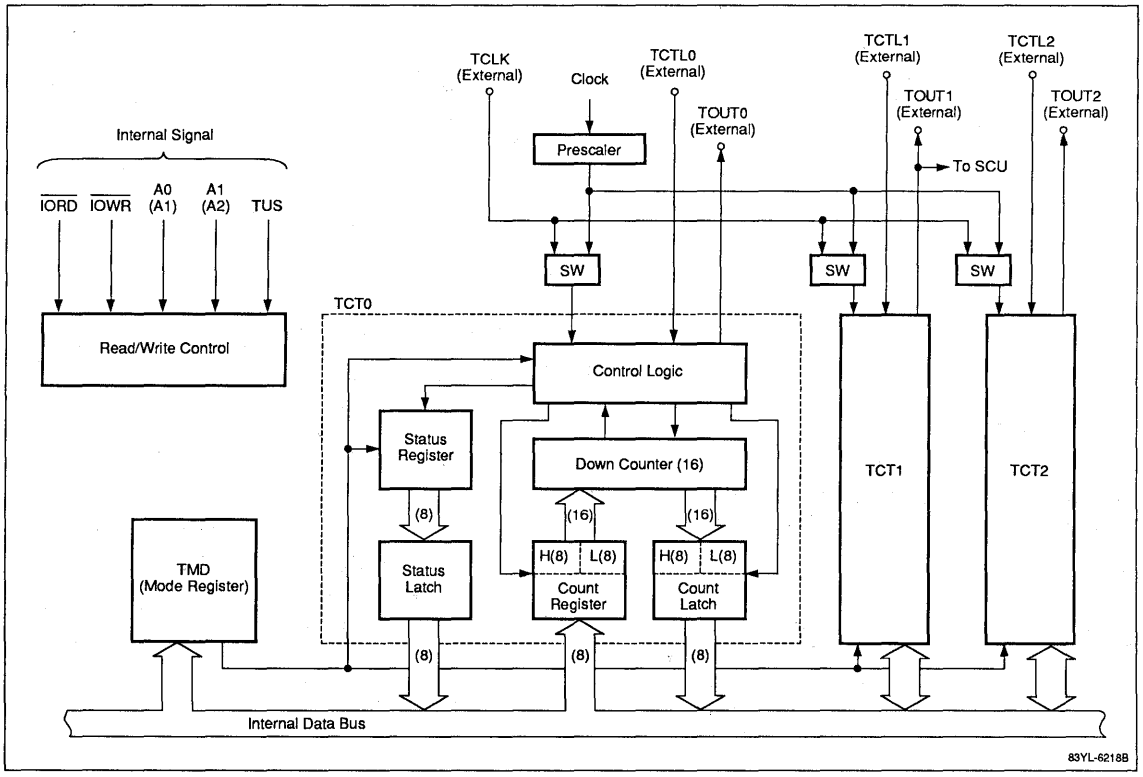
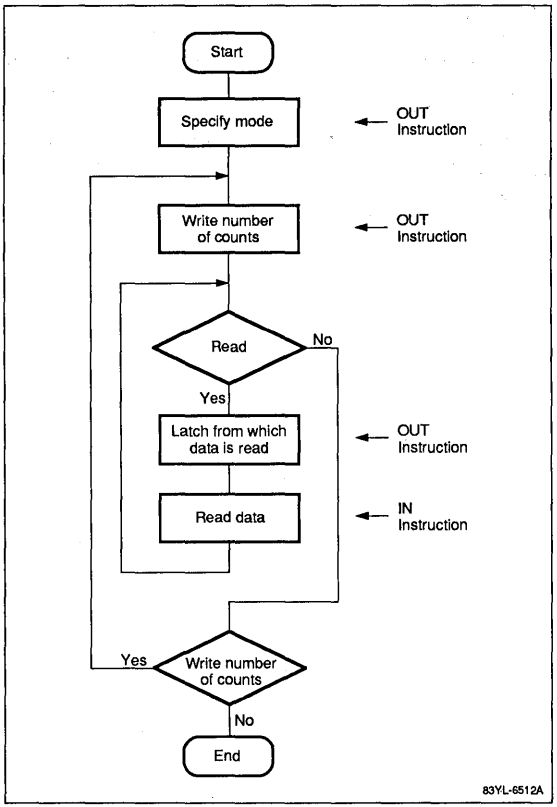


Figure 64. TCU Operating Procedure



TCU Commands

The TCU is programmed by issuing I/O instructions to the I/O port addresses programmed in the OPHA and TULA registers. The individual TCU registers are selected by address bits A_2 and A_1 or (A_1) and (A_0) as follows.

A_2 (A_1)	A_1 (A_0)	Register	Operation
0	0	TCT0	Read/write
		TST0	Read
0	1	TCT1	Read/write
		TST1	Read
1	0	TCT2	Read/write
		TST2	Read
1	1	TMD	Write

Timer Mode Register (TMD)

The TMD register selects the operating mode for each timer/counter and issues the latch command for one or more timer/counters. Figures 65, 66, and 67 show three configurations of the TMD register.

Figure 65. TMD Register; Mode Word

SC	RWM	CMODE	BD

7 0

SC	Counter
00	TCT0
01	TCT1
10	TCT2
11	Multiple latch command

RWM	Read/Write Mode
00	Counter latch command
01	Lower byte only
10	Upper byte only
11	Lower byte followed by upper byte

CMODE	Count Mode
000	Mode 0
001	Mode 1
x10	Mode 2
x11	Mode 3
100	Mode 4
101	Mode 5

BD	Count
0	Binary count
1	BCD count

x = Don't care.

Figure 66. TMD Register; Count Latch Command

SC	0	0	0	0	0	0

7 0

SC	Counter To Be Latched
00	TCT0
01	TCT1
10	TCT2

Figure 67. TMD Register; Multiple Latch Command

1	1	CL	SL	CT2	CT1	CT0	0

7 0

CL	Latches Count Data
0	Yes
1	No

SL	Latches Status
0	Yes
1	No

CTn	Selects Counter TCTn
0	No
1	Yes

Timer/Counter Registers (TCT)

Writes to the timer/counter registers (TCT0-TCT2) stores the new count in the appropriate timer/counter. The count latch command is used before reading count data to latch the current count and prevent inaccuracies.

Timer Status Registers (TST)

The timer status registers (TST0-TST2) contain status information for the specified counter. See figure 68. The latch command is used to latch the appropriate counter status before reading status information. If both status and counter data are latched for a counter, the first read operation returns the status data and subsequent read operations obtain the count data.

Figure 68. Timer Status Registers (TSTn)

OL	NC	RWM	CMODE	BD

7 0

OL	**TOUTn Level**
0	Low
1	High
NC	**Null Count**
0	Valid
1	Invalid
RWM	**Read/Write Mode**
	Same as TMD register.
CMODE	**Count Mode**
	Same as TMD register.
BD	**Count**
	Same as TMD register.

Count Modes

There are six programmable timer/counter modes. The timing waveforms for these modes are shown in figure 69.

Mode 0 (Interrupt on End of Count). In mode 0, TOUT changes from low to high level when the specified count is reached. This mode is available on all timer/counters.

Mode 1 (Retriggerable One-Shot). In mode 1, a low-level, one-shot pulse triggered by TCTL is output from the TOUT pin.

Mode 2 (Rate Generator). In mode 2, TOUT cyclically goes low for one clock period when the counter reaches the 0001H count. A counter in this mode operates as a frequency divider.

Mode 3 (Square-Wave Generator). Mode 3 is a frequency divider similar to mode 2, but the output has a symmetrical duty cycle.

Mode 4 (Software-Triggered Strobe). In mode 4, when the specified count is reached, TOUT goes low for the duration of one clock pulse.

Mode 5 (Hardware-Triggered Strobe). Mode 5 is similar to mode 4 except that operation is triggered by the TCTL input and can be retriggered.

Figure 69. Timer Counter Unit (TCU) Waveforms (Sheet 1 of 3)

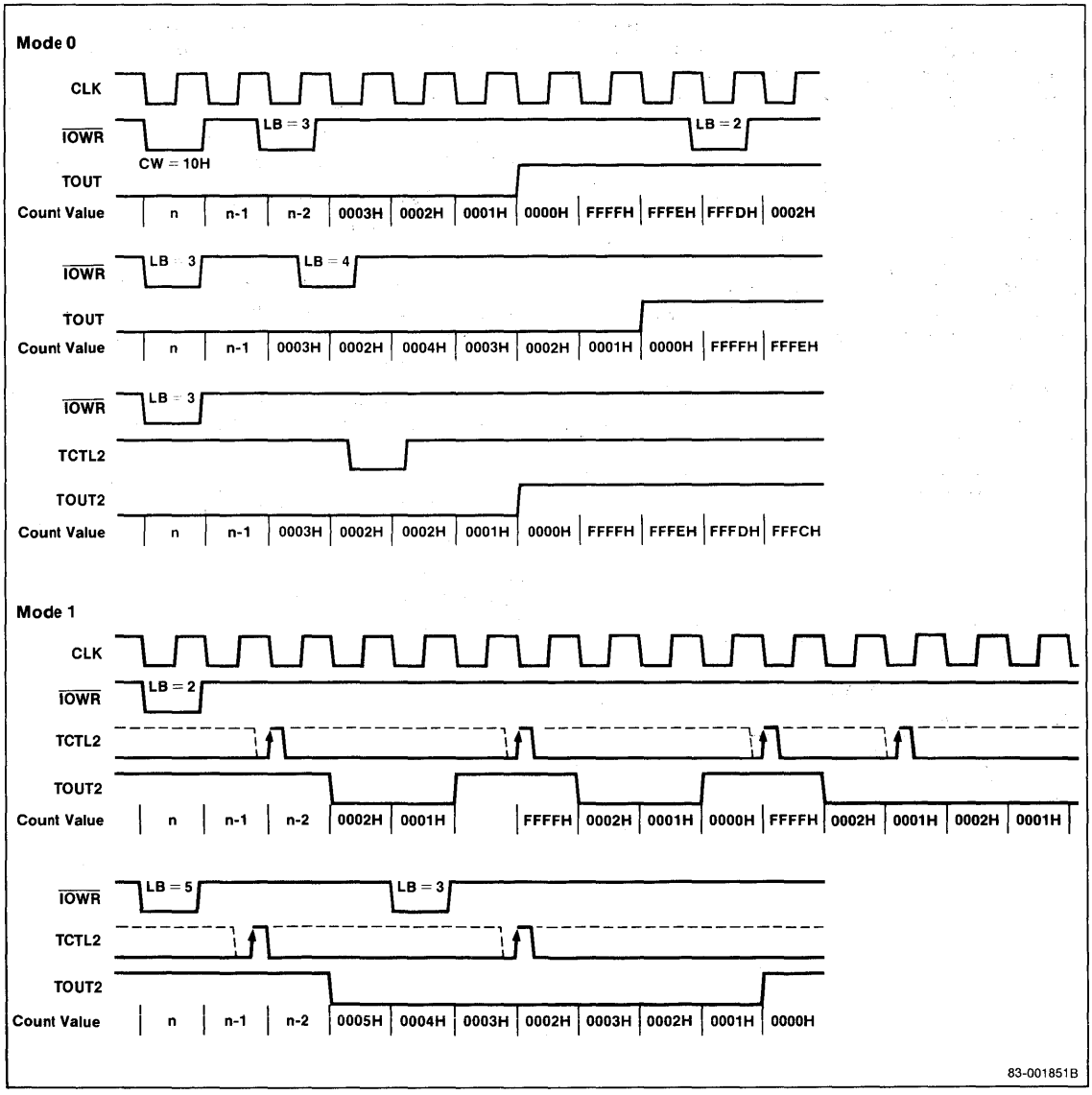
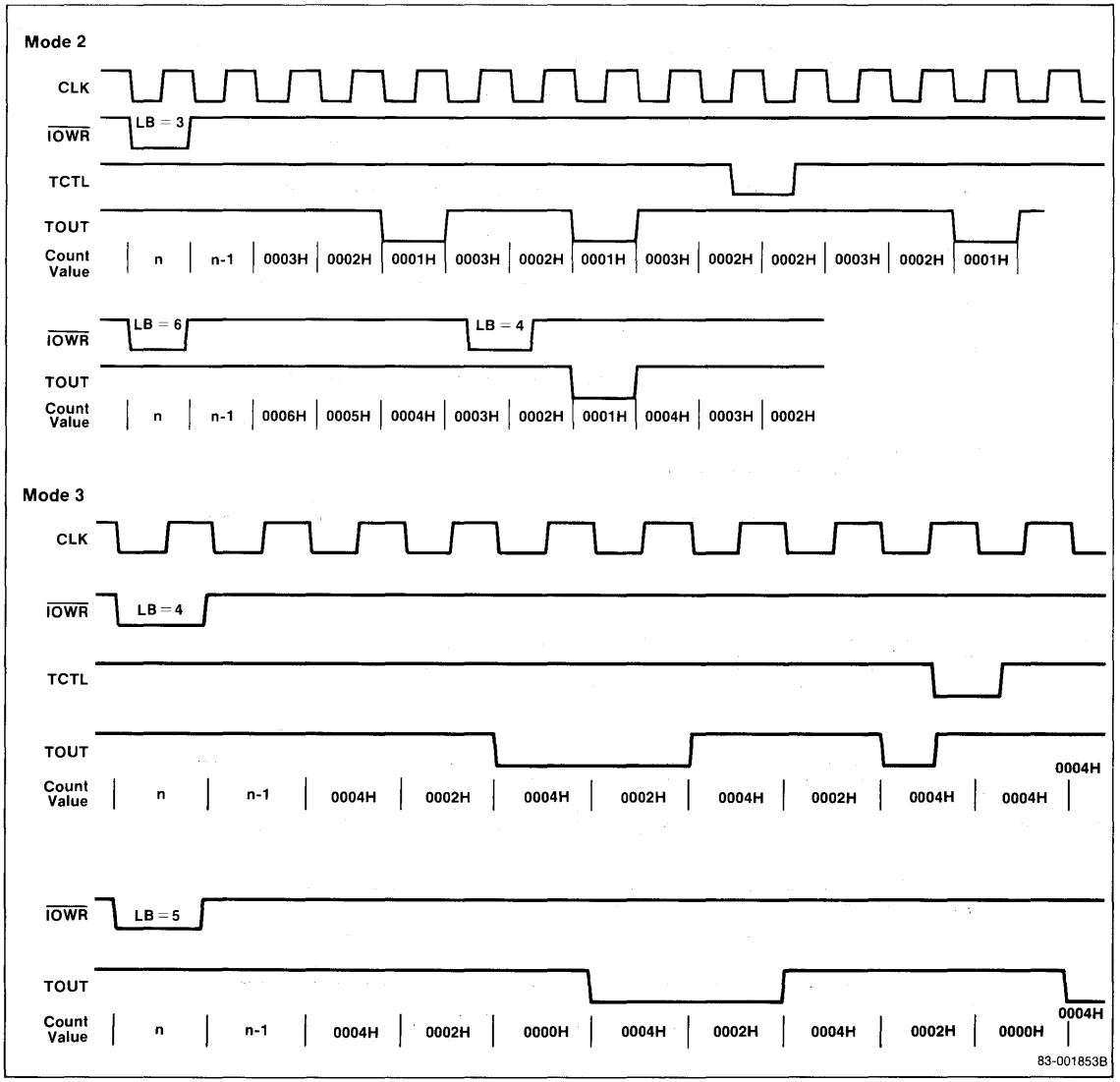
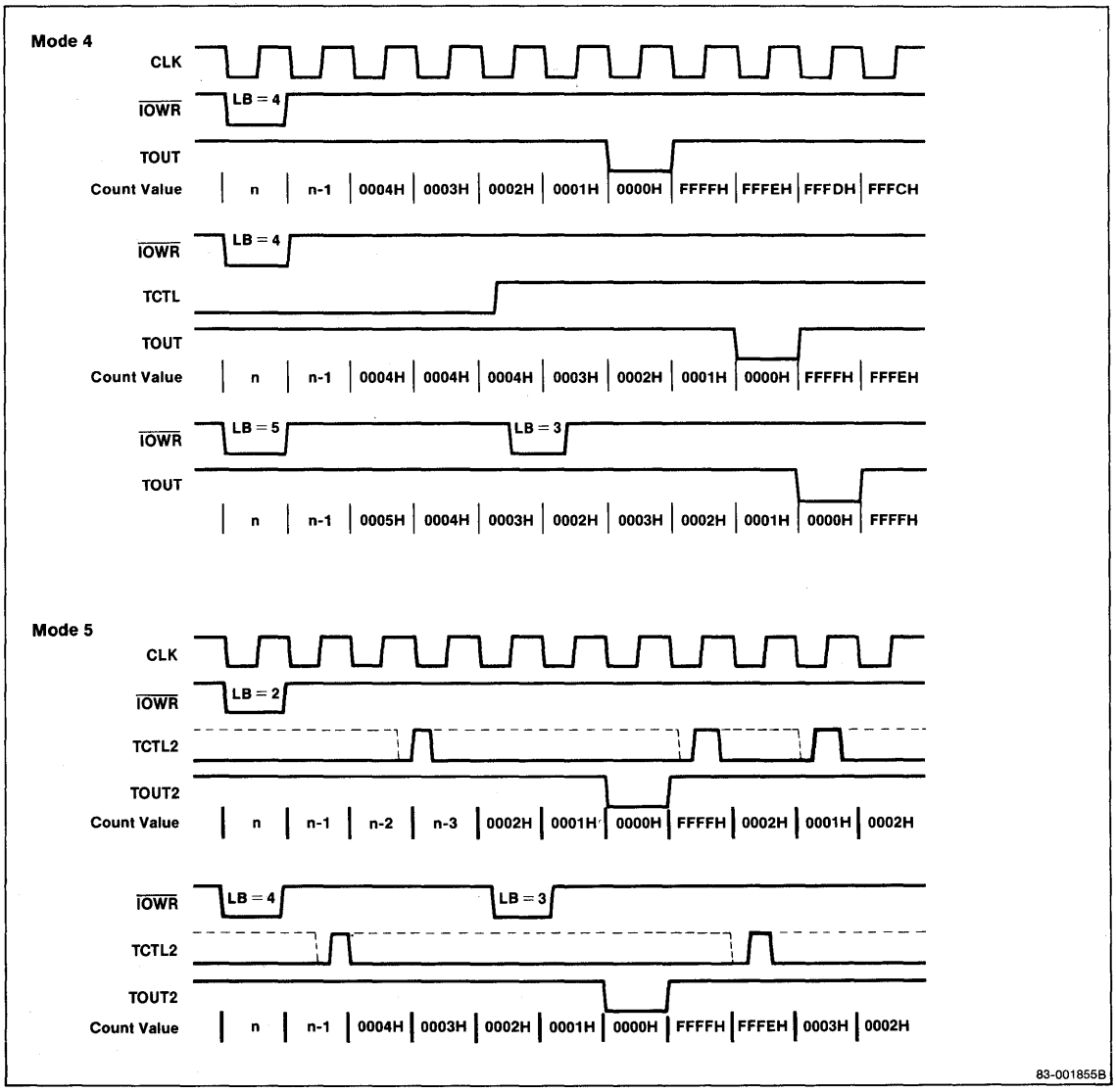


Figure 69. Timer Counter Unit (TCU) Waveforms (Sheet 2 of 3)



3f

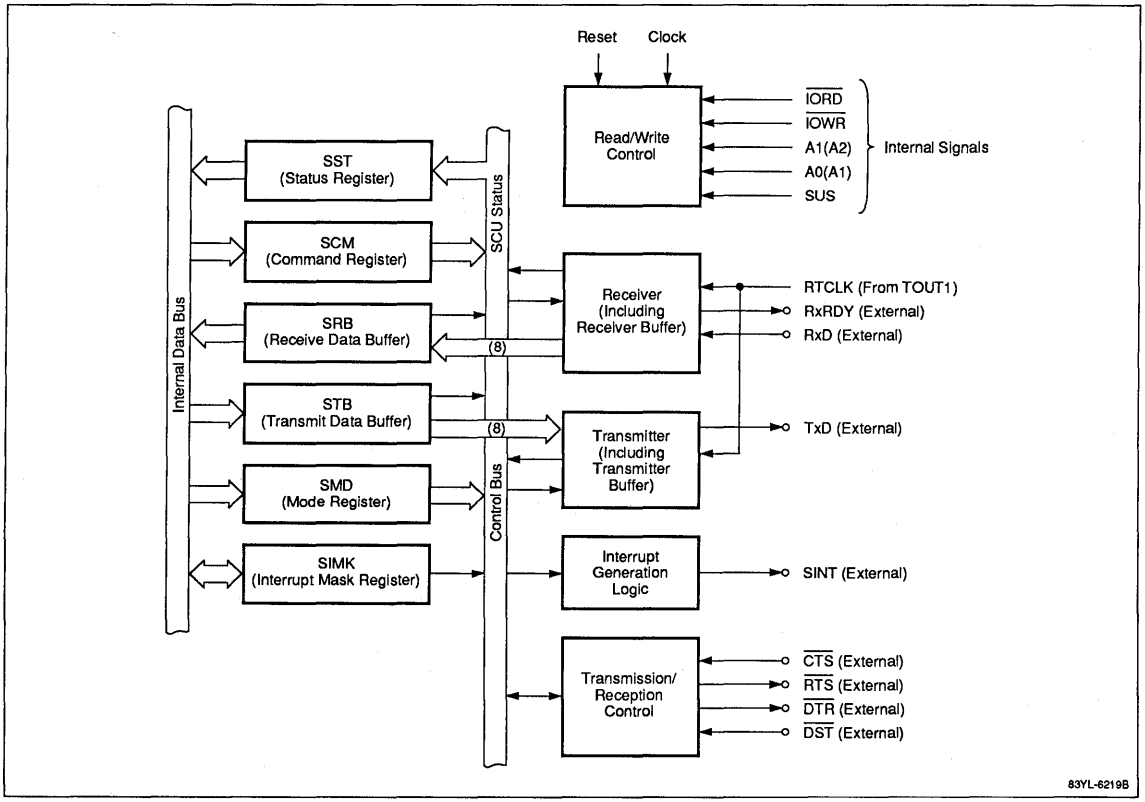
Figure 69. Timer Counter Unit (TCU) Waveforms (Sheet 3 of 3)



SERIAL CONTROL UNIT

The serial control unit (SCU) is a single asynchronous channel that performs serial communication between the V53 and an external device. The SCU is similar to the μPD71051 Serial Control Unit except for the lack of synchronous communication protocols. Figure 70 is a block diagram of the SCU.

Figure 70. SCU Block Diagram



The SCU has the following features.

- Full-duplex, asynchronous serial controller
- Clock rate divisor: 16 or 64
- Baud rates to 640 kb/s (external clock), 500 kb/s (internal clock)
- Dedicated baud-rate generator or can use timer 1
- Full modem signaling support (ATS, CTS, DSR, DTR)
- Character length: 7 or 8 bits
- Stop bit length: 1 or 2 bits
- Break transmission and detection
- Full-duplex, double-buffered transmitter/receiver
- Even, odd, or no parity
- Parity, overrun, and framing error detection
- Receiver-full/transmitter-empty interrupt

The SCU contains four separately addressable registers for reading/writing data, reading status, and controlling operation of the SCU. The serial receive buffer (SRB) and the serial transmit buffer (STB) store the incoming and outgoing character data. The serial status register (SST) allows software to determine the current state of both the transmitter and the receiver.

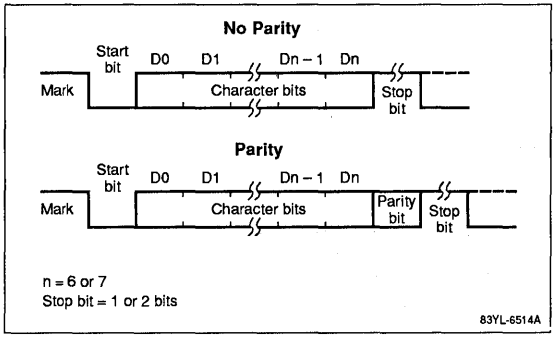
The serial command (SCM) and serial mode registers (SMD) determine the operating mode of the SCU while the serial interrupt mask register (SIMK) allows software control of the SCU receive and transmit interrupts.

Serial Data Format

Figure 71 shows the format of the serial data processed by the SCU. In this serial data, the character bits are

transferred between the CPU and SCU. The start bit, parity bit, and stop bit(s) sandwiching the character bits are control information necessary for serial data communications. They are automatically appended when data is transmitted or deleted when data is received by the SCU.

Figure 71. Serial Data Format



Receiver Operation

While the RxD pin is high, the receiver is in an idle state. A transition on RxD from high to low indicates the start of new serial data. When a complete character has been received, it is transferred to the SRB register. The receive buffer ready (RBRDY) bit in the SST register is set and (if unmasked) an interrupt is generated. The SST also latches any parity, overrun, or framing errors at this time.

The receiver detects a break condition when a null character with zero parity is received. The BRK bit is set for as long as the subsequent receive data is low and resets when RxD returns to a high level.

Transmitter Operation

TxD is kept high while the STB register is empty. When the transmitter is enabled and a character is written to the STB register, the data is converted to serial format and output on the TxD pin. The start bit indicates the start of the transmission and is followed by the character stream (LSB to MSB) and an optional parity bit. One or two stop bits are then appended, depending on the programmed mode. When the character has been transferred from the STB, the TBRDY bit in the SST is set and if unmasked, a transmit buffer empty interrupt is generated.

Serial data can be transmitted and received by polling the SST register and checking the TBRDY or RBRDY flags. Data can also be transmitted and received by SCU-generated interrupts. The SCU generates an interrupt in either of these conditions:

(1) The receiver is enabled, the SRB is full, and receive interrupts are unmasked.

(2) The transmitter is enabled, the STB is empty, and transmit interrupts are unmasked.

SCU Registers and Commands

I/O instructions to the I/O addresses selected by the OPHA and SULA registers are used to read/write the SCU registers. Address bits A_2 and A_1 (or A_1 and A_0) and the read/write lines select one of the six internal registers as shown below.

A_2 (A_1)	A_1 (A_0)	Register	Operation
0	0	SRB	Read
0	0	STB	Write
0	1	SST	Read
0	1	SCM	Write
1	0	SMD	Write
1	1	SIMK	Read/write

The baud rate counter (BRC) register is fixed at address FFE9H in the system I/O area.

The SRB and STB are 8-bit registers. When the character length is 7 bits, the lower 7 bits of the SRB register are valid and bit 7 is cleared to 0. If programmed for 7-bit characters, bit 7 of the STB is ignored.

The SST register (figure 72) contains the status of the transmit and receive data buffers and the error flags. Error flags are persistent. Once an error flag is set, it remains set until a clear error flags command is issued.

SCU Initialization

After a hardware reset, the SCU is set to the following condition.

Baud rate factor	x64
Character length	7 bits
Stop bit	1 bit
Transmit/receive	Disabled
Break detection	No
Errors	No
$\overline{RTS}$, $\overline{DTR}$ pins	High level

Figure 72. Serial Status Register (SST)

DSR	BKD	FE	OVE	PE	1	RBRDY	TBRDY
7							0

DSR	**$\overline{\text{DSR}}$ Input Pin**
0	High level
1	Low level
BKD	**Break Detection**
0	Normal reception
1	Break status detected
FE	**Framing Error**
0	No error
1	Error
OVE	**Overrun Error**
0	No error
1	Error
PE	**Parity Error**
0	No error
1	Error
RBRDY	**Receive Data Buffer**
0	SRB empty
1	SRB full
TBRDY	**Transmit Data Buffer**
0	STB full
1	STB empty

The SCM register (figure 73) stores the command word that controls transmission, reception, error flag reset and break transmission.

The SMD register (figure 74) stores the mode word that determines serial characteristics such as baud rate divisor, parity, character length, and stop bit length.

Initialization software should first program the SMD register followed by the SCM register. Unlike the μPD71051, the SMD register can be modified anytime without resetting the SCU.

Figure 73. Serial Command Register (SCM)

—	—	RTS	ECL	SBRK	RE	DTR	TE
7							0

RTS	**Controls $\overline{\text{RTS}}$ Output Pin**
0	High level
1	Low level
ECL	**Clears Error Flags**
0	No operation
1	Clears error flags
SBRK	**Break Transmission**
0	TxD pin operates normally
1	TxD pin outputs low level
RE	**Enables/Disables Reception**
0	Disables
1	Enables
DTR	**Controls $\overline{\text{DTR}}$ Pin**
0	High level
1	Low level
TE	**Enables/Disables Transmission**
0	Disables
1	Enables

Figure 74. Serial Mode Register (SMD)

STL	PS	CL	BF
7			0

STL	**Number of Stop Bits**
x0	Illegal
01	1 stop bit
11	2 stop bits
PS	**Parity Selection**
x0	Parity disabled
01	Odd parity
11	Even parity
CL	**Character Length**
x0	Illegal
10	7 bits
11	8 bits
BF	**Baud Rate**
0x	Illegal
10	RTCLK frequency/16
11	RTCLK frequency 64

The SIMK register (figure 75) controls the occurrence of RBRDY and TBRDY interrupts. When an interrupt is masked, it is prevented from propagating to the interrupt control unit.

3f

Figure 75. Serial Interrupt Mask Register (SIMK)

–	–	–	–	–	–	TM	RM

7 0

TM	TBRDY Interrupt Mask
0	Unmasked
1	Masked

RM	RBRDY Interrupt Mask
0	Unmasked
1	Masked

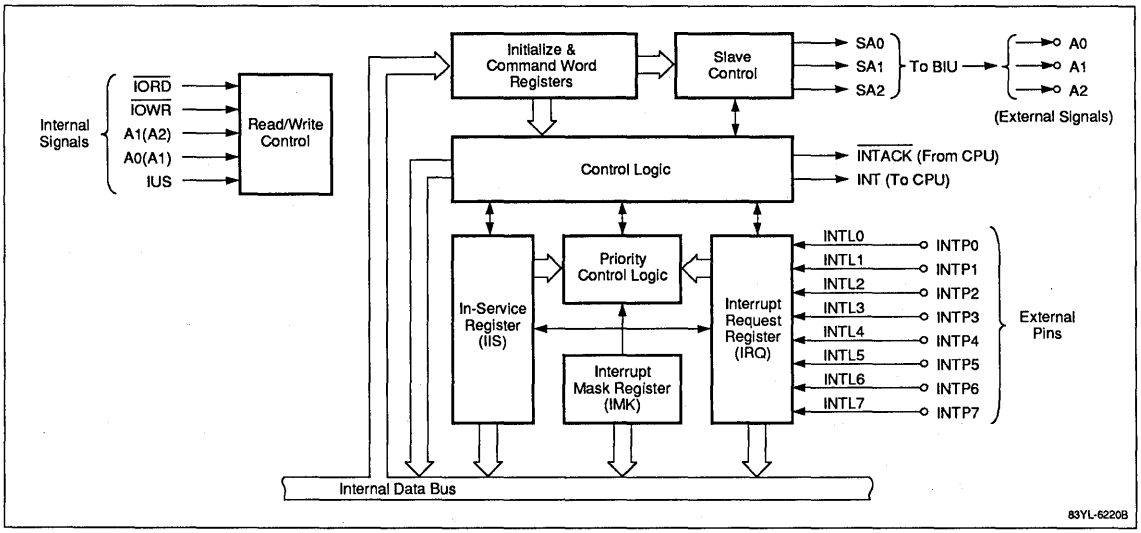
Baud Rate Clock

The baud rate clock may come from either of two sources: the internal baud rate generator or timer 1. The internal baud rate generator is discussed in the System I/O section, and timer 1 is described in the TCU section. The SCTL system I/O register controls the selection of the baud rate clock.

INTERRUPT CONTROL UNIT

The interrupt control unit (ICU) is a programmable interrupt controller equivalent to the μPD71059. The ICU arbitrates up to eight interrupt inputs, generates a CPU interrupt request, and outputs the interrupt vector number on the internal data bus during an interrupt acknowledge cycle. Cascading up to seven external slave μPD71059 interrupt controllers permits the V53 to support up to 56 interrupt sources. Figure 76 is the block diagram for the ICU.

Figure 76. ICU Block Diagram



To reduce current drain in the standby modes, the V53 does not have internal pullup resistors on the INTP0-INTP7 pins. This is different from the μPD71059 and V40/V50.

The ICU has the following features.

- Eight external interrupt request inputs
- Cascadable with μPD71059 interrupt controllers
- Programmable edge- or level-triggered interrupts (TCU, edge-triggered only)
- Individually maskable interrupt requests
- Programmable interrupt request priority
- Polling mode

ICU Registers

Use I/O instructions to the I/O addresses selected by the OPHA and IULA registers to read from and write to the ICU registers. Address bit A_1 and the command word select an ICU internal register. See table 7.

Table 7. ICU Register Selection

	A_1 (A_0)	Other Condition	Operation
Read	0	IMD selects IRQ	CPU ← IRQ data
	0	IMD selects IIS	CPU ← IIS data
	0	*Polling phase	CPU ← Polling data
	1	—	CPU ← IMKW
Write	0	D4 = 1	CPU → IIW1
	0	D4 = 0 and D3 = 0	CPU → IPFW
	0	D4 = 0 and D3 = 1	CPU → IMDW
	1	During initialization	CPU → IIW2
	1		CPU → IIW3
	1		CPU → IIW4
	1	After initialization	CPU → IMKW

* In the polling phase, polling data has priority over the contents of the IRQ or IIS register when read.

Initializing the ICU

The ICU is always used to service maskable interrupts in a V53 system. Prior to accepting maskable interrupts, the ICU must first be initialized. See figure 77. Note that RESET does not initialize the ICU.

Interrupt Initialization Words 1-4. Words IIW1-IIW4 (figures 78-81) indicate whether external μPD71059s are connected as slaves, select the base interrupt vector, and select edge- or level-triggered inputs for INT1-INT7. Interrupt sources from the TCU are fixed as edge-triggering. INT0 is internally connected to TOUT0, and INT2 may be connected to TOUT1 by the IRSW field in the OPCN.

The initialization words are written in consecutive order starting with IIW1. IIW2 sets the interrupt vector. IIW3 specifies which interrupts are connected to slaves. IIW3 is only required in extended systems. The ICU will only expect to receive IIW3 if SNGL = 0 (bit D_1 of IIW1). IIW4 is only written if II4 = 1 (bit D_0 of IIW1).

Figure 77. Initialization Sequence

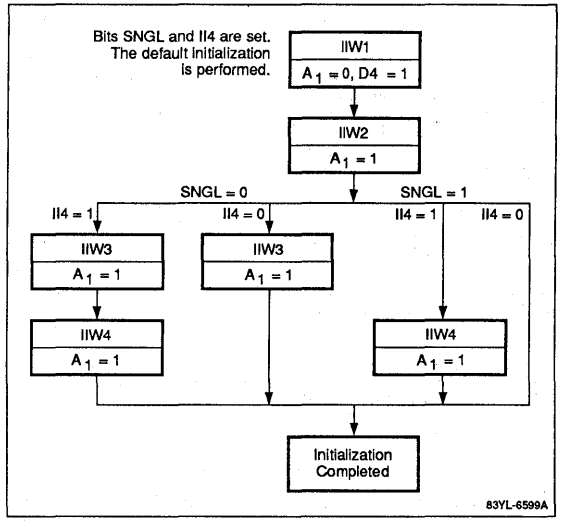


Figure 78. ICU Initialize, Word 1 (IIW1)

—	—	—	1	LEV	—	SNGL	II4

D7 D0

LEV	Input Trigger Mode
0	Rising-edge trigger
1	High-level trigger

SNGL	Mode
0	Expanded mode (slave controllers)
1	Single mode (no slave controllers)

II4	Write to W4
0	IIW4 not required
1	IIW4 required

Figure 79. ICU Initialize, Word 2 (IIW2)

V_7	V_6	V_5	V_4	V_3	—	—	—

D7 D0

V_7-V_3 = Higher 5 bits of interrupt vector number

3f

Figure 80. ICU Initialize, Word 3 (IIW3)

S_7	S_6	S_5	S_4	S_3	S_2	S_1	0
D7							D0

S_n	Slave Connection
0	INTn is not a slave input
1	INTn is a slave input

Figure 81. ICU Initialize, Word 4 (IIW4)

0	0	0	EXTN	—	—	SFI	1
D7							D0

EXTN	External Nesting Mode
0	Normal
1	Expanded

SFI	Self-Finish Interrupt
0	FI command mode
1	Self-finish mode

Command Words. The interrupt mask word (MKW) contains programmable mask bits for each of the eight interrupt inputs. The interrupt priority and finish word (IPFW) is used by the interrupt handler to terminate processing of an interrupt or change interrupt priorities. The interrupt mode word (IMDW) selects the polling register, interrupt request (IRQ) or interrupt in-service (IIS) register, and the nesting mode. See figures 82-84.

Figure 82. Command Word IMKW

M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0
D7							D0

M_n	Interrupt Request Mask
0	INTn not masked
1	INTn masked

Figure 83. Command Word IPFW

RP	SIL	FI	0	0	IL2	IL1	IL0
D7							D0

RP	Rotate Priority
0	No rotation
1	Rotation

SIL	Level
0	Not specified
1	Specified

FI	Finish Interrupt
0	Non-FI command
1	FI command

IL2-IL0	Interrupt Level
000	INT0
001	INT1
010	INT2
011	INT3
100	INT4
101	INT5
110	INT6
111	INT7

Figure 84. Command Word IMDW

—	SNM	EXCN	0	1	POL	SR	IS/IR
D7							D0

SNM	EXCN	Nesting Mode 2
0	—	No operation
1	0	Release exceptional nesting mode
1	1	Set exceptional nesting mode

POL		Polling Mode
0		No operation
1		Polling command

SR	IS/IR	Register to Be Read
0	—	No operation
1	0	Interrupt request register (IRQ)
1	1	Interrupt in-service register (IIS)

μPD71059 Cascade Connection

To increase the number of maskable interrupts, up to seven slave μPD71059 interrupt controllers can be cascaded. During cascade operation, each slave μPD71059 INT output is routed to one of the V53 INTP inputs.

During the second interrupt acknowledge bus cycle, the ICU places the slave address on the address lines AD_{10}-AD_8. Each slave compares this address with the slave address programmed using interrupt initialization word 3 (IIW3). If the same, the slave will place the interrupt vector on pins AD_7-AD_0 during the second interrupt acknowledge bus cycle.

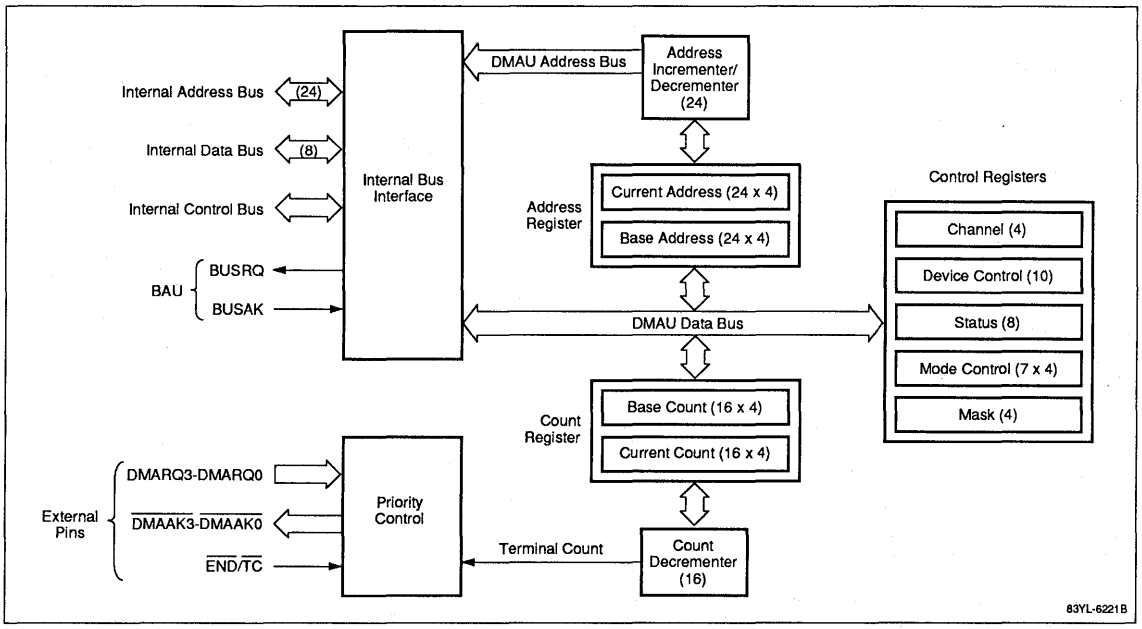
DMA CONTROL UNIT

The DMA control unit (DMAU) is a high-speed DMA controller compatible with the μPD71071 and μPD71037 DMA controllers. The DMAU has four independent DMA channels and performs high-speed data transfers between memory and external peripheral devices at speeds as high as 4M words/second in a 16-MHz system. Figure 85 is the block diagram for the DMAU.

The DMAU has the following features.

- Four independent DMA channels
- μPD71037 or μPD71071 compatibility modes
- Cascade mode for slave DMA controllers
- 24-bit address registers
- 16-bit transfer count registers
- Single, demand, and block transfer modes
- Autoinitialization
- Address increment/decrement
- Fixed/rotating channel priorities
- TC output at transfer end
- Forced termination of service by END input

Figure 85. DMAU Block Diagram



μPD71071 and μPD71037 Mode Comparison

The DMAU has two operating modes selected by the SCTL system control register. Respectively, the μPD71071 and μPD71037 modes offer hardware and software compatibility with existing systems based on the μPD71071 DMA controller (also the V40/V50 microprocessor) and the μPD8237 DMA controller.

In applications where DMA software compatibility is not an issue, programming flexibility is greater in the μPD71071 mode. However, the software DMA request capability of the μPD71037 mode is often useful.

The following compares the major functional differences between the two modes.

Function	μPD71037 Mode	μPD71071 Mode
DMA channel selection	Mode control register by write data (operand); other registers have a unique address	Referenced by channel register (DCH)
Base and current register access	Consecutive 8-bit quantities	16-bit quantities
Base registers	Write only	Read and write
DMA termination	Bus release mode	Bus release and bus hold modes
Software DMA requests	Yes	No
DMA transfers	Byte	Byte or word

The DMAU is intended for high-speed data transfers between memory and peripherals with minimum latency. Neither mode provides memory-to-memory DMA transfers because the powerful string moves of the CPU can accomplish block memory transfers as fast as dedicated DMA hardware could. The DMAU does not provide compressed timing as do the μPD71071 and μPD71037.

Master/Slave Mode

The DMAU operates in either master or slave mode. In slave mode, the DMAU samples the four DMARQ input pins every clock. If one or more inputs are active, the corresponding DMA request bits are set and the DMAU sends a bus request to the BAU while continuing to sample the DMA request inputs.

After the BAU returns the DMA bus acknowledge signal, the DMAU stops DMA request sampling, selects the DMA channel with the highest priority, and enters the bus master mode to perform the DMA transfer. While in the bus master mode, the DMAU controls the external bus and performs DMA transfers based on the preprogrammed channel information.

See figure 45 and the associated text for a detailed description of DMA bus cycles.

Terminal Count

The DMAU ends DMA service when the terminal count condition is generated or when the END input is asserted. A terminal count (TC) is produced when the contents of the current count register underflows from zero. If autoinitialization is not enabled when DMA service terminates, the mask bit of the channel is set and the DMARQ input of that channel is masked. Otherwise, the current count and address registers are reloaded from the base registers, and new DMA transfers are again enabled.

DMA Transfer Type

The type of transfer the DMAU performs depends on the following conditions.

- Transfer direction (each channel)
- Bus mode
- Transfer mode (each channel)

Transfer Direction

All DMA transfers use memory as a reference point. Therefore, a DMA read operation (figure 86) transfers data from memory to I/O port and writes the data into memory. During memory-to-I/O transfer, the DMA mode register (DMD) is used to select the transfer directions for each channel and activate the appropriate control signals.

Operation	Transfer	Signals Activated
DMA read	Memory to I/O	$\overline{IOWR}$, $\overline{MRD}$
DMA write	I/O to memory	$\overline{IORD}$, $\overline{MWR}$
DMA verify	No transfer	Addresses only

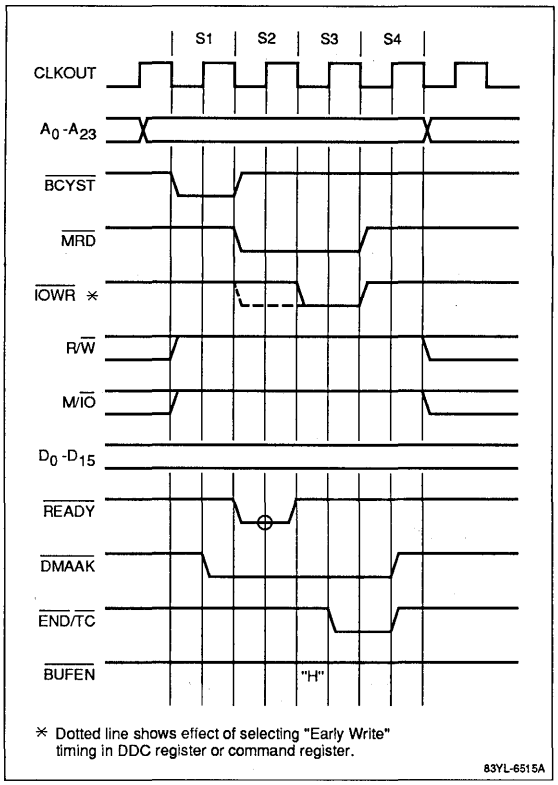
Bus Mode

The two available modes for determining how the DMAU releases the CPU bus are bus release and bus hold. In μPD71037 mode, the DMAU always functions in bus release mode. In μPD71071 mode, the DMAU is programmable for bus release or bus hold mode via the DMA device control (DDC) register.

In bus release mode, bus control is always relinquished each time the service has completed. Therefore, if multiple DMA requests are generated simultaneously, a bus cycle other than that for the DMAU is inserted between consecutive DMA services (see figure 87). Consequently, in certain applications DMA response may be delayed. However, bus release mode gives better assurance that the CPU will continue to execute programs in DMA intensive environments.

In bus hold mode, if another DMA request is generated before the end of one service, that request can be serviced without the DMAU relinquishing the bus. However, the same channel cannot be serviced consecutively. This mode provides better DMA response but may prevent CPU bus activity for extended periods of time.

Figure 86. Typical Memory-to-I/O DMA Cycle



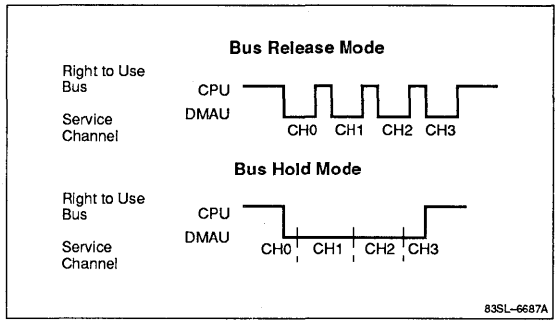
Transfer Modes

The DMAU has three transfer modes as listed below. In μPD71071 mode, bits 6 and 7 (TMODE) of the mode control register (DMD) select the transfer mode. In μPD71037 mode, bits 6 and 7 of the channel mode register specify the mode. Transfer mode operation is the same in both μPD71071 and μPD71037 modes.

Transfer Mode	Termination Conditions
Single	After each byte/word transfer $\overline{\text{END}}$ input Terminal count
Demand	$\overline{\text{END}}$ input Terminal count Service channel DMARQ dropped Generation of a higher priority DMARQ (bus hold mode)
Block	$\overline{\text{END}}$ input Terminal count

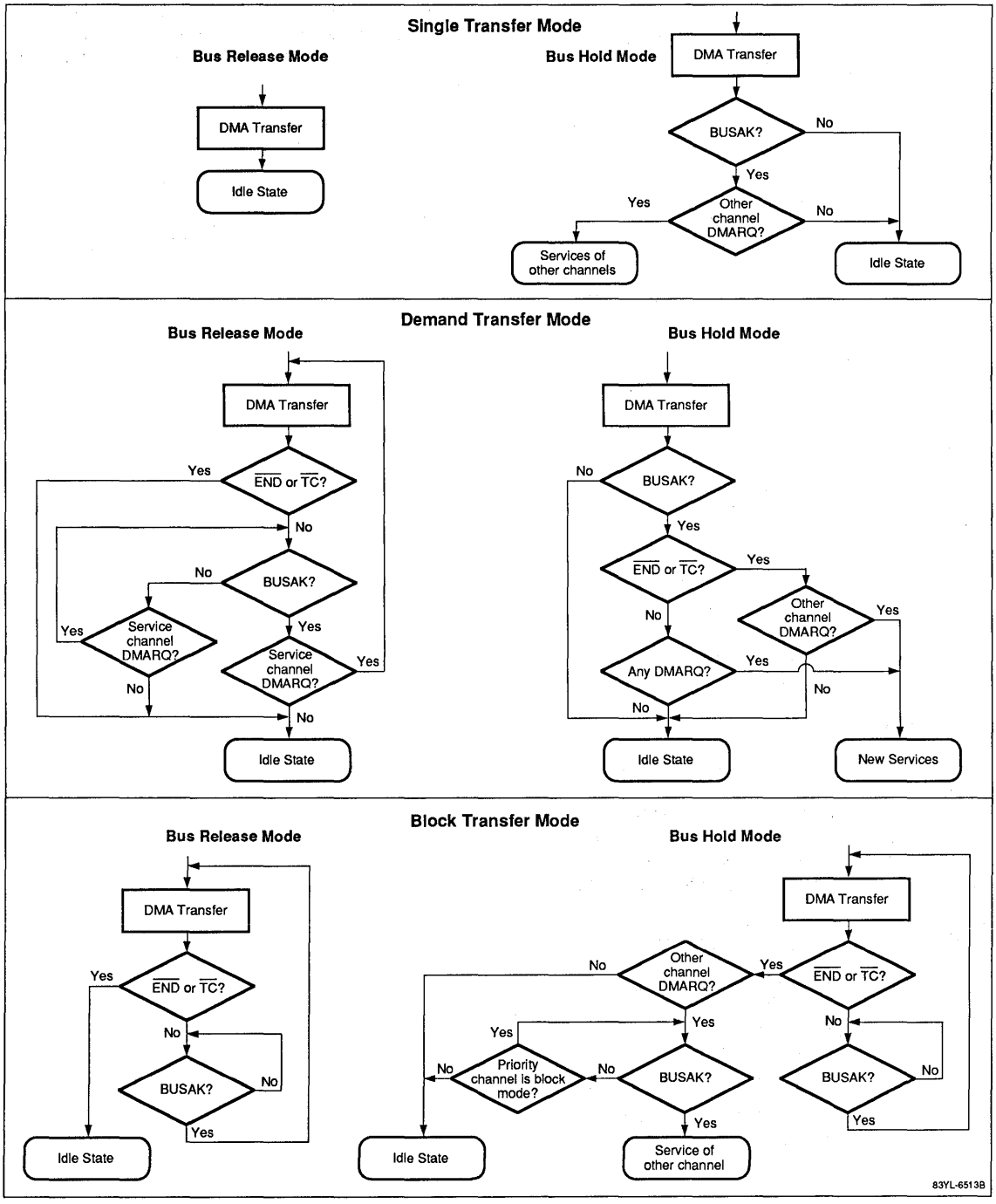
The operation of single, demand, and block transfers depends on whether the DMAU is in bus release or bus hold mode. Figure 88 shows the operations flow for the six possible transfer and bus mode operations in DMA transfer.

Figure 87. Bus Modes



3f

Figure 88. Transfer Modes



Single Transfer Mode. In bus release mode, when a channel completes transfer of a single byte or word, the DMAU enters the slave mode regardless of the state of DMA request inputs. In this manner, other lower priority bus masters can access the bus.

In bus hold mode (μPD71071 mode only), when a channel completes transfer of a single byte or word, the DMAU terminates the channel's service even if the DMARQ request signal is asserted. The DMAU will then service any other requesting channel. If there are no requests from any other DMA channels, the DMAU releases the bus and enters the idle state.

Demand Transfer Mode. In bus release mode, the currently active channel continues to transfer data as long as the DMA request of that channel is active, even though other DMA channels are issuing higher priority requests. When the DMA request of the serviced channel becomes inactive, the DMAU releases the bus and enters the idle state.

In bus hold mode (not available in μPD71037 mode), when the active channel completes a single transfer, the DMAU checks the other DMA request lines without ending the current service. If there is a higher priority DMA request, the DMAU stops the service of the current channel and starts servicing the highest priority channel requesting service. If there is no higher request than the current one, the DMAU continues to service the currently active channel. Lower priority DMA requests are honored without releasing the bus after the current channel service is complete.

Block Transfer Mode. In bus release mode, the current channel continues DMA transfers until a terminal count or the external $\overline{END}$ input becomes active. During this time, the DMAU ignores all other DMA requests. After completion of the block transfer, the DMAU releases the bus and enters the idle state, even if DMA requests from other channels are active.

In bus hold mode (μPD71071 mode only), the current channel transfers data until an internal or external $\overline{END}$ signal becomes active. When the service is complete, the DMAU checks all DMA requests without releasing the bus. If there is an active request, the DMAU immediately begins servicing the request. The DMAU releases the bus after it honors all DMA requests or a higher priority bus master requests the bus.

Autoinitialize

This function is enabled by programming the mode register (μPD71071 and μPD71037 modes).

When a mode register enables autoinitialize for a channel, the DMAU automatically reinitializes the address and count registers when $\overline{END}$ is asserted or the terminal count condition is reached. The contents of the base address and base count registers are transferred to the current address and current count registers, and the applicable bit of the mask register remains cleared.

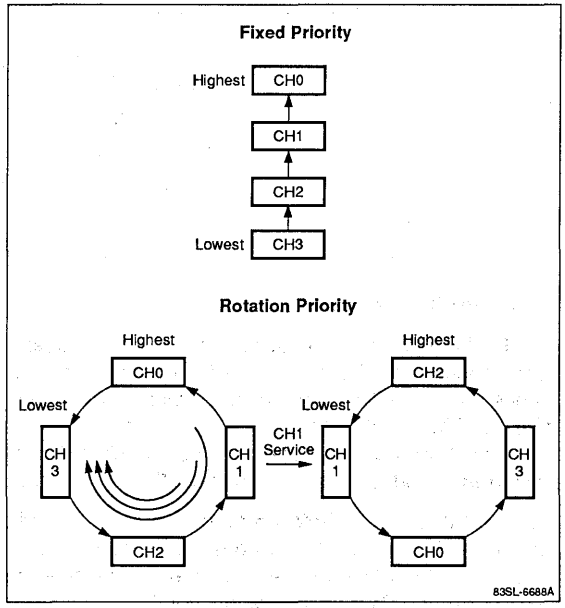
Channel Priority

Each of the four DMAU channels is assigned a priority. When multiple DMA requests from several channels occur simultaneously, the channel with the highest priority will be serviced first.

The device control register selects one of two priority schemes: fixed or rotating (figure 89). In fixed priority, channel 0 is assigned the highest priority, and channel 3, the lowest. In rotating priority, priority order is rotated after each service so that the channel last serviced receives the lowest priority. This method prevents the exclusive servicing of higher priority channels and the lockout of lower priority DMA channels.

The rotating priority feature is selected by programming the DMA device control (DDC) register in μPD71071 mode or by a write to the command register in μPD71037 mode.

Figure 89. Priority Order



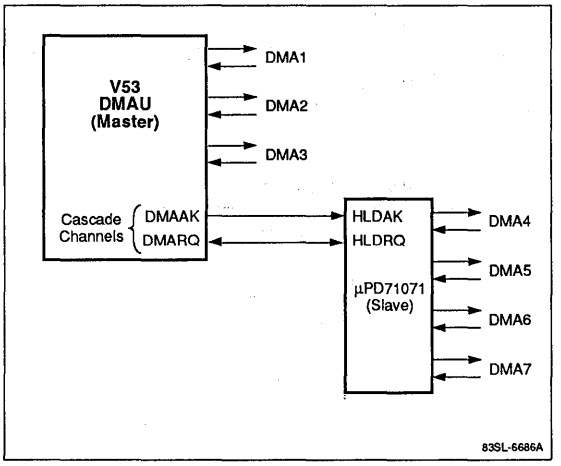
3f

Cascade Connection

Slave DMA controllers can be cascaded to easily expand the system DMA channel capacity to 16 DMA channels. Figure 90 shows an example of cascade connection. During cascade operation, the DMAU acts as a mediator between the BAU and the slave DMA controller. During DMA cascade mode operation, it is the responsibility of external logic to isolate the cascade bus master from the V53 control outputs. These outputs are listed near the beginning of this document.

The DMAU always operates in the bus release mode while a cascade channel is in service, even when the bus hold mode is programmed. Other DMA requests are held pending while a slave DMA controller channel is in service. When the cascaded device ends service and moves into the idle state, the DMAU also moves to the idle state and releases the bus. The DMAU continues to operate normally with the other noncascaded channels.

Figure 90. μPD71071 Cascade Example

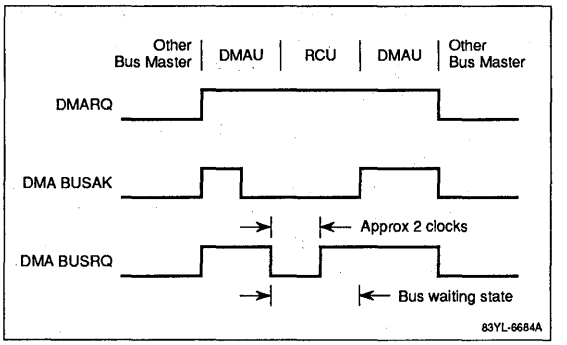


Bus Waiting Operation

The DMAU automatically performs a bus waiting operation (figure 91) whenever the REFU refresh request queue fills. When the DMA bus acknowledge goes inactive, the DMAU enters the bus waiting mode and inactivates the DMA bus request signal. Control of the bus is then transferred to the higher priority REFU by the BAU.

Two clocks later, the DMAU reasserts its internal DMA bus request. The bus waiting mode is continued until the DMA bus acknowledge signal again becomes active and the interrupted DMA service is immediately restarted.

Figure 91. Bus Waiting Operation



Address and Count Registers

Each DMA channel has a 24-bit base address register and a 24-bit current address register. In addition, each channel also has its own 16-bit current count register and base count register. The base registers hold a value determined by the CPU and transfer this value to the current registers during autoinitialization. These registers are available in both μPD71071 mode and μPD71037 mode, but the method of accessing these registers changes with compatibility mode.

The BNKR registers extend the μPD71037 mode addresses from 16 to 24 bits. In μPD71071 mode, the count register and lower word of the address registers can be accessed in 16-bit quantities. In μPD71037 mode, these registers must be accessed in 8-bit quantities.

Programming the DMAU

To prepare a channel for DMA transfer, the following characteristics must be programmed.

- Starting address for the transfer
- Transfer count
- DMA operating mode
- Transfer size (byte/word in μPD71071 mode)

The contents of the OPHA and DULA registers determine the base I/O port address of DMAU. Addresses A_3-A_0 are used to select a particular register. There are two register sets, one for μPD71071 mode and the other for μPD71037 mode.

μPD71071 Mode

The μPD71071 mode is selected by programming the DMAU bit of the SCTL register to zero. The register set for this mode (table 7) is mapped into A_3-A_0 regardless of the IOAG value in the SCTL register.

Table 7. Register Selection (μPD71071 Mode)

A_3-A_0	Address	Register	Operation	Notes
0000	0H	DICM	Write	1
0001	1H	DCH	Read/Write	1
0010	2H	DBC/DCC (low)	Read/Write	2
0011	3H	DBC/DCC (high)	Read/Write	2
0100	4H	DBA/DCA (low)	Read/Write	2
0101	5H	DBA/DCA (high)	Read/Write	2
0110	6H	DBA/DCA (upper)	Read/Write	1, 2
0111	7H	Reserved	—	
1000	8H	DDC (low)	Read/Write	
1001	9H	DDC (high)	Read/Write	
1010	AH	DMD	Read/Write	1, 2
1011	BH	DST	Read	1
1100	CH	Reserved	—	
1101	DH	Reserved	—	
1110	EH	Reserved	—	
1111	FH	DMK	Read/Write	1

Notes:

(1) Register can be accessed only with byte In/Out instructions. All others can be accessed with 16-bit In/Out instructions.

(2) There are four such registers, one for each DMA channel. The particular register accessed is determined by the DCH register.

DMAU Registers in μPD71071 Mode

Initialize. The DMA initialize command register (DICM) performs a software reset of the DMAU. The DICM is accessed using the byte OUT instruction. See figure 92.

The DMAU initializes the registers as follows.

Register	Name	Operation
DICM	Initialize	Clear
DCH	Channel	Select channel 0
DBC, DCC	Count	No change
DBA, DCA	Address	No change
DDC	Device control	Clear
DMD	Mode control	Clear
DST	Status	Clear
DMK	Mask	Set (mask all channels)

Figure 92. DMA Initialize Command Register (DICM); μPD71071 Mode

—	—	—	—	—	—	—	RES

7 — Address 0H, Byte OUT Instruction — 0

RES	Reset
0	No operation
1	Reset DMAU

Channel Register. Writes to the DMA channel register (DCH) select one of the four DMA channels for programming and also the base/current registers. Reads of the DCH register return the currently selected channel and the register access mode. See figure 93.

Figure 93. DMA Channel Register (DCH); μPD71071 Mode

Channel Register Read

—	—	—	BASE	SEL3	SEL2	SEL1	SEL0

7 — Address 1H, Byte IN Instruction — 0

BASE	Access Conditions
0	Read: current only Write: base and current
1	Read/write: base only

SEL3-SEL0	Selected Channel
0001	0
0010	1
0100	2
1000	3

Channel Register Write

—	—	—	—	—	BASE	SELCH

7 — Address 1H, Byte OUT Instruction — 0

BASE	Access Conditions
0	Read: current only Write: base and current
1	Read/write: base only

SELCH	Selected Channel
00	Channel 0
01	Channel 1
10	Channel 2
11	Channel 3

Count Registers. When bit 2 of the DCH register is cleared, a write to the DMA count register (figure 94) updates both the DMA base count (DBC) and the DMA current count (DCC) registers with a new count. If bit 2 of the DCH register is set, a write to the DMA count register affects only the DBC register.

3f

The DBC register holds the initial count value until a new count is specified. If autoinitialization is enabled, this value is transferred to the DCC register when a terminal count or END condition occurs. For each DMA transfer, the current count register is decremented by 1. The count value loaded into the DBC/DCC register is 1 less than the desired transfer count.

Figure 94. DMA Count Registers (DBC, DCC); μPD71071 Mode

C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

7 Address 2H 0
IN/OUT Instruction

C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8

7 Address 3H 0
IN/OUT Instruction

Address Register. Use either byte or word I/O instructions with the lower 2 bytes (4H and 5H) of the DMA address register (figure 95). However, byte I/O instructions must be used to access the high-order byte (6H) of this register. When bit 2 of the channel register is cleared, a write to the DMA address register updates both the DMA base address (DBA) and the DMA current address (DCA) registers with the new address. If bit 2 of the DCH register is set, a write to the DMA address register affects only the DBA register.

The DBA register holds the starting address value until a new address is specified. This value is transferred to the DCA register automatically if autoinitialization is selected. For each DMA transfer, the current address register is updated by 2 during word transfers and by 1 during byte transfers.

Figure 95. DMA Address Registers (DBA, DCA); μPD71071 Mode

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0

7 Address 4H 0
IN/OUT Instruction

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

7 Address 5H 0
IN/OUT Instruction

A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}

7 Address 6H 0
IN/OUT Instruction

Device Control Register. The DMA device control register (DDC) (figure 96) is used to program the DMA transfer characteristics common to all DMA channels. It controls the bus mode, write timing, priority logic, and enable/disable of the DMAU See figure 97.

Figure 96. DMA Device Control Register (DDC); μPD71071 Mode

–	–	EXW	ROT	–	DDMA	–	–

7 Address 8H 0
IN/OUT Instruction

EXW	**Writing** (Note 1)
0	Normal
1	Extended
ROT	**Priority**
0	Fixed
1	Rotational
DDMA	**DMA Operation** (Note 2)
0	Enable
1	Disable

–	–	–	–	–	–	WEV	BHLD

7 Address 9H 0
IN/OUT Instruction

WEV	**Wait During Verify** (Note 3)
0	Disable
1	Enable
BHLD	**Bus Mode**
0	Bus release
1	Bus hold

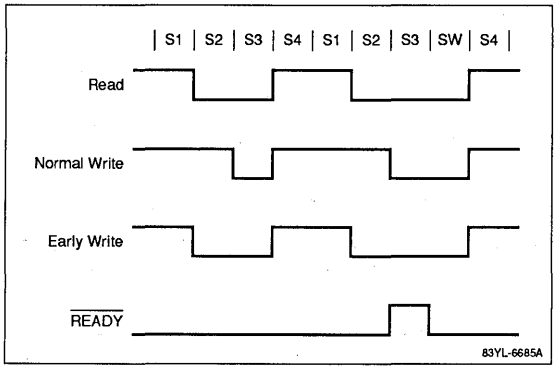
Notes:

(1) Disables BUSRQ to the BAU to prevent incorrect DMA operation while the DMAU registers are being initialized or modified.

(2) When EXW = 0, the write signal becomes active (normal write) during S3 and SW. When EXW = 1, the write signal becomes active during S2, S3, and SW (like the read signal).

(3) Wait states are generated by the $\overline{\text{READY}}$ signal during a verify transfer.

Figure 97. Early Write Cycle Timing



Mode Control Register. The DMA mode control register (DMD) selects the operating mode for each DMA channel. The DCH register selects which DMD register will be accessed. A byte IN/OUT instruction must be used to access this register. See figure 98.

Figure 98. DMA Mode Control Register (DMD); μPD71071 Mode

TMODE	ADIR	AUTI	TDIR	—	W/$\overline{B}$

7 Address 0AH 0

TMODE	Transfer Mode
00	Demand
01	Single
10	Block
11	Cascade
ADIR	**Address Direction**
0	Increment
1	Decrement
AUTI	**Autoinitialize**
0	Disable
1	Enable
TDIR	**Transfer Direction**
00	Verify
01	I/O-to-memory
10	Memory-to-I/O
11	Not allowed
W/$\overline{B}$	**Word/Byte Transfer**
0	Byte
1	Word

Addresses and count registers are updated as follows during byte/word transfers.

Register	Byte Transfer	Word Transfer
Address register	±1	±2
Count register	−1	−1

During word transfers, two bytes starting at an even address are handled as a single word. If the starting address is odd, a DMA transfer is started after first decrementing the address by 1. For this reason, always select even addresses. The A_0 and $\overline{UBE}$ outputs control byte and word DMA transfers. The following shows the relationship between the data bus width, A_0, and $\overline{UBE}$ signals, and data bus status.

A_0	$\overline{UBE}$	Data Bus Status
0	1	D_0-D_7 valid
1	0	D_8-D_{15} valid
0	0	D_0-D_{15} valid

Status Register. The DMA status register (DST) contains information about the current state of each DMA channel. Software can determine if a termination condition has been reached (TC0-TC3) or if a DMA service request is present (RQ0-RQ3). The byte IN instruction must be used to read this register. See figure 99.

3f

Figure 99. DMA Status Register (DST); μPD71071 Mode

RQ3	RQ2	RQ1	RQ0	TC3	TC2	TC1	TC0

7 Address 0BH 0
Byte IN Instruction

RQn	DMA Request, Channel n
0	No DMA request active
1	DMA request active
TCn	**Terminal Count, Channel n**
0	Not ended (for each read)
1	$\overline{END}$ or terminal count

Mask Register. The DMA mask register (DMK) allows software to individually enable and disable DMA channels. The DMK register can only be accessed via byte I/O instructions. See figure 100.

Figure 100. DMA Mask Register (DMK); μPD71071 Mode

—	—	—	—	M3	M2	M1	M0

7 Address 0FH 0
Byte IN/OUT Instruction

Mn	DMARQ Mask, Channel n
0	Not masked
1	Masked

μPD71037 Mode

The μPD71037 mode is selected by programming the DMAM bit of the SCTL register to 1. See figure 48. Note that on RESET, the DMAU is put into μPD71071 mode. The register set for the μPD71037 mode (table 8) is mapped into A_3-A_0 (IOAG = 0) or A_4-A_1 (IOAG = 1). For the case where IOAG = 1, the DULA system I/O register determines whether the DMAU responds to A_0 = 0 or 1.

Table 8. Register Set for μPD71037 Mode

Channel	Register	Read/Write	Address
0	DCA DCA, DCB	R W	0000
	DCC DCC, DBC	R W	0001
1	DCA DCA, DCB	R W	0010
	DCC DCC, DBC	R W	0011
2	DCA DCA, DCB	R W	0100
	DCC DCC, DBC	R W	0101
3	DCA DCA, DCB	R W	0110
	DCC DCC, DBC	R W	0111
	DST DDC	R W	1000
	DSRQ	W	1001
	DSCM	W	1010
	DMD	W	1011
	DMK	W	1111

The registers in table 8 can be accessed only by byte I/O operations. The IOAG bit of the SCTL register determines whether these registers reside in contiguous bytes, or whether they each occupy one-half word (i.e., whether the registers are byte or word aligned). If word aligned (IOAG = 1), the low bit of the DULA register determines whether the DMAU will use the upper or lower byte of the word. In μPD71071 mode, the setting of the IOAG bit makes no difference; the register addresses do not change.

μPD71037 Commands

In addition to the registers explained above, three I/O addresses cause commands to be executed when they are written to. The value of the data written is not important; it is the action of performing an I/O write to one of these addresses that initiates the desired action. The commands and their corresponding addresses (A_4-A_0) are shown here.

Command	IOAG = 0	IOAG = 1
Clear byte select flag	x1100	1100x
Initialize	x1101	1101x
Clear mask register	x1110	1110x

DMAU Registers in μPD71037 Mode

Most of the DMAU registers in this mode are the same as those in the μPD71071 mode, but with a different I/O address or method of access.

Count and Address Registers. The DCA, DBA, DCC, and DBC registers are 16 bits wide, but can only be accessed in byte-wide chunks. The byte select flag (BSF) determines which byte is accessed. When the BSF is low, the low byte is used; when the BSF is high, the high byte is used. The BSF cannot be read; to set it to a known state, a byte select flag clear command must be issued by performing an 8-bit I/O write to address x1100b. To read or write one of these registers, first clear the BSF, and then perform two consecutive 8-bit I/O operations. The low byte will be accessed first and the high byte second.

Bank Registers. The DMA memory addresses in the μPD71037 mode are 16 bits, compared with 24-bit addresses in the μPD71071 mode. To expand the 16-bit addresses into the full 24-bit address space of the V53, a set of bank registers is provided, BNKR0-BNKR3, one per DMA channel.

Each 8-bit register contains the upper address bits, A_{23}-A_{16}, to be used when a DMA channel is active. DMA addresses are modified after each transfer to point to the next address in the DMA buffer. The SCTL system I/O register, CE1-CE0 bits, control whether a carry is propagated into the upper address bits when the DMA address is incremented or decremented. CE0 controls the carry propagation to A_{16} and CE1 controls the carry to A_{20}.

The BNKR registers are read or written using byte I/O operations. See figure 101. As with other V53 internal registers, the I/O address to which the BNKR registers respond is programmable. The BADR system I/O register (address FFE1H) sets the base address of the BNKR registers in the 256-byte block of I/O space selected by the OPHA register. See figure 102.

Also, to allow maximum flexibility, the low two address bits of each BNKR register are programmable. The BSEL system I/O register (address FFE0H) sets the low two address bits for each BNKR register. See figure 103. As with other programmable addresses, the IOAG bit of the

SCTL register has the effect of shifting the settable address one bit position to the left.

The bank registers are only enabled in μPD71037 mode. In μPD71071 mode, they cannot be read or written.

Figure 101. DMA Bank Registers (BNKR); μPD71037 Mode

A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}

7 BNKR0 IN/OUT 0

A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}

7 BNKR1 IN/OUT 0

A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}

7 BNKR2 IN/OUT 0

A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}

7 BNKR3 IN/OUT 0

Figure 102. Bank Address Register (BADR); μPD71037 Mode

A_7	A_6	A_5	A_4	A_3	A_2	$*A_1$	$*A_0$

7 Address FFE1H IOAG = 0 0

A_7	A_6	A_5	A_4	A_3	$*A_2$	$*A_1$	A_0

7 Address FFE1H IOAG = 1 0

*Address bits are set by the BSEL register.

Figure 103. Bank Select Register (BSEL); μPD71037 Mode

BNK3	BNK2	BNK1	BNK0

7 Address FFE0H 0

BNKn	*Address Bits in BADR Register
00	00
01	01
10	10
11	11

* Address bits are A_1, A_0 if IOAG = 0 or A_2, A_1 if IOAG = 1. (IOAG is a bit in the SCTL register.)

Device Control Register. In μPD71037 mode, there are fewer device options. The wait during verify and bus hold control bits are not offered. The DMA device control register (DDC) has only one byte to control early write cycles, channel priority, and global DMA enable. See figure 104.

Figure 104. DMA Device Control Register (DDC); μPD71037 Mode

—	—	EXW	ROT	—	DDMA	—	—

7 Byte OUT Instruction 0

EXW	Write Timing (Note 1)
0	Normal
1	Early
ROT	**Channel Priority**
0	Fixed
1	Rotational
DDMA	**DMA Operation**
0	Enable
1	Disable

Notes:

(1) When EXW = 0, the write signal becomes active during S3 and SW. When EXW = 1, the write strobe is asserted earlier during S2, S3, and SW (same as read strobe).

Channel Mode Registers. Each channel has a mode register allocated to it. All four registers are accessed using the same I/O address. The low two bits of the data written to the DMD register select the channel. Note that byte transfers are supported but 16-bit transfers are not. Figure 105 shows the format of the channel mode register.

3f

Figure 105. DMA Channel Mode Registers (DMD); μPD71037 Mode

TMODE	ADIR	AUTI	TDIR	SELCH

7 Byte OUT Instruction 0

TMODE	Transfer Mode
00	Demand
01	Single
10	Block
11	Cascade
ADIR	**Address Direction**
0	Increment
1	Decrement
AUTI	**Autoinitialize**
0	Disable
1	Enable
TDIR	**Transfer Direction**
00	Verify
01	I/O-to-memory
10	Memory-to-I/O
11	Not allowed
SELCH	**Channel Selection for Mode Change**
00	Channel 0
01	Channel 1
10	Channel 2
11	Channel 3

Status Register. This DST register (figure 74) is identical to the μPD71071 mode DST register, but is at I/O address x1000b.

Figure 106. DMA Status Registers (DST); μPD71037 Mode

RQ3	RQ2	RQ1	RQ0	TC3	TC2	TC1	TC0

7 Address x1000b 0
Byte IN Instruction

RQn	DMA Request, Channel n
0	No DMA request active
1	DMA request active
TCn	**Terminal Count, Channel n**
0	Not ended (for each read)
1	$\overline{\text{END}}$ or terminal count

Mask Register and Single-Channel Mask Control Register. The format and I/O address of this DMK register (figure 107) is the same as in μPD71071 mode except that it cannot be read; it is a write-only register. The DMK register can be put into a known state by writing to it directly, by using the clear mask register command, or by using the single-channel mask control register (DSCM) at I/O address x1010b to set or clear the enable bit for an individual channel (figure 108).

Figure 107. DMA Mask Register (DMK); μPD71037 Mode

—	—	—	—	M3	M2	M1	M0

7 Address 0FH 0
Byte OUT Instruction

Mn	DMARQ Mask, Channel n
0	Not masked
1	Masked

Figure 108. DMA Single-Channel Mask Control Register (DSCM); μPD71037 Mode

—	—	—	—	—	—	SMQ	SELCH

7 Byte OUT Instruction 0

SMQ	Mask Setting
0	Clear mask bit
1	Set mask bit
SELCH	**DMARQ Mask Channel Selection**
00	Channel 0
01	Channel 1
10	Channel 2
11	Channel 3

Software DMA Request Register. The DSRQ register is used by software to trigger a DMA operation. One application is to simulate the assertion of a hardware DMA request for diagnostic purposes. This register is written with the number of the targeted channel and a bit that sets or clears an internal request flag associated with that channel. Figure 109 shows the format of this register.

Figure 109. Software DMA Request Register (DSRQ); μPD71037 Mode

—	—	—	—	—	—	SRQ	SELCH

7 Byte OUT Instruction 0

SRQ	Request
0	Clear request bit
1	Set request bit
SELCH	**Software DMARQ Channel Selection**
00	Channel 0
01	Channel 1
10	Channel 2
11	Channel 3

Initialization. In μPD71037 mode, there is no DICM initialize register. Instead, the DMAU is initialized by performing an I/O write to address x1100b.

POWER CONSERVATION

The V53 has three power conservation features.

- Scalable system clock
- Low-power HALT standby mode
- Very-low-power STOP mode

These features give three levels of power reduction, making the V53 ideal for use in portable or other low-power applications. The standby control register (SBCR) at address 0FFF1H in the system I/O area controls all three functions. See figure 110.

Scalable System Clock

The V53 is a CMOS device and power consumption is directly proportional to clock frequency. By reducing the frequency, power use can be significantly decreased. The system clock is used by the CPU and internal peripherals. The CLKC field in the SBCR selects a scale factor that divides the oscillation frequency by 2, 4, 8, or 16 to produce the system clock. This value can be changed dynamically to adjust the clock rate to the most efficient performance level for the task at hand.

Caution: The system clock must not be set to less than the minimum frequency specified in the AC Characteristics table.

Figure 110. Standby Control Register (SBCR)

–	–	–	CLKC	WT	STOP
7			Address FFF1H		0

CLKC	System Clock Frequency f_{CLK}
00	f_{CLK} = Osc freq ÷ 2
01	f_{CLK} = Osc freq ÷ 4
10	f_{CLK} = Osc freq ÷ 8
11	f_{CLK} = Osc freq ÷ 16

WT	* Oscillation Stabilization Time
00	$2^{19} \div f_{CLK}$
01	$2^{18} \div f_{CLK}$
10	$2^{17} \div f_{CLK}$
11	$2^{16} \div f_{CLK}$

STOP	When HALT Instruction Is Executed
0	Sets HALT mode
1	Sets STOP mode

* For example, if WT = 11 and f_{CLK} = 16 MHz, time = 4.096 ms

HALT Standby Mode

Power can be further reduced by putting the CPU in HALT standby mode. In this mode, the CPU is not operating, but all the internal peripherals are still enabled and may be drawing power. HALT mode is entered by setting the STOP bit in the SBCR to 0 and executing a HALT instruction.

The V53 will come out of HALT standby mode in response to RESET, NMI, or an interrupt from the internal interrupt control unit. If interrupts were enabled (IE = 1) before HALT mode was entered, an ICU interrupt wakeup will result in the interrupt handler being entered; if interrupts were not enabled (IE = 0), then execution will resume at the instruction following the HALT that put the CPU in the standby mode. If NMI wakes up the CPU, the NMI handler is always entered.

The bus hold (HLDRQ/HLDAK) function still operates during standby mode. External bus masters can take the bus from V53. Also, refresh and DMA cycles can still occur. The SCU and TCU can both be active, and can supply the wakeup interrupt if desired.

Refer to table 1 to find out what state the V53 outputs will be in HALT standby mode.

3f

STOP Mode

This mode provides the maximum power reduction. The clock generator is disabled; the oscillator circuit is turned off. Power usage is minimal. STOP mode is entered by setting the STOP bit in the SBCR to 1 and executing a HALT instruction. Since the system clock is not active, none of the on-chip peripherals can be used in this mode.

If the timer unit's TCLK input is used and driven by an external oscillator, the timer will continue to function and consume power.

The output pins in STOP mode are in the same state as in the HALT mode. Refer to table 1 for details. The V53 will wake up from STOP mode in response to a RESET or NMI.

Oscillator Stabilization Time

When the V53 is reset or when it wakes up from STOP mode, the oscillator circuit is started up. This circuit can take a relatively long time to come up to speed and to stabilize. The oscillator stabilization time field (WT) in the SBCR does not affect the physical startup time; it determines how long the V53 will wait for the clock generator oscillator circuit to stabilize. The user should determine the worst case stabilization time and select a longer value of WT.

RESET FUNCTION

The V53 is reset when a falling edge is input to the $\overline{\text{RESET}}$ pin and is subsequently held low for six clocks or longer than the oscillator stabilization time and then made high.

CPU Operations

When the V53 is reset, the CPU is initialized as shown in figure 111 and starts prefetching instructions from address FFFF0H.

Figure 111. CPU Reset Status

Prefetch Pointer	PFP	0000H
Program Counter	PC	0000H
Program Segment Register	PS	FFFFH
Stack Segment Register	SS	0000H
Data Segment 0 Register	DS0	0000H
Data Segment 1 Register	DS1	0000H
Queue		Cleared
Program Status Word	PSW	

1	1	1	1	V 0	DIR 0	IE 0	BRK 0

15 … 0

S 0	Z 0	0	AC 0	0	P 0	1	CY 0

7 … 0

Internal Register Operations

Some internal registers are also initialized by the $\overline{\text{RESET}}$ input signal. See figure 112. The rest of the registers retain the status they had immediately before the $\overline{\text{RESET}}$ signal was applied, but their contents are undefined at power up.

Figure 112. Register Reset Status

	Initial Value, Bits 7-0							
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
System I/O Area								
SCTL	—	—	—	0	0	0	0	0
OPSEL	—	—	—	—	0	0	0	0
WCY0	—	—	—	—	—	1	1	1
WCY1	—	1	1	1	—	1	1	1
WCY2	—	1	1	1	—	1	1	1
WCY3	—	1	1	1	—	1	1	1
WCY4	—	1	1	1	—	1	1	1
WMB0	—	1	1	1	—	1	1	1
WMB1	—	1	1	1	—	1	1	1
WAC	—	—	—	—	0	0	0	0
TCKS	—	—	—	0	0	0	0	0
RFC	—	0	—	0	1	0	0	0
SBCR	—	—	—	0	0	0	0	0

Figure 112. Register Reset Status (cont)

	Initial Value, Bits 7-0							
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Serial Control Unit								
SMD	0	1	0	0	1	0	1	1
SCM	—	—	0	0	0	0	0	0
SIMK	—	—	—	—	—	—	1	1
SST	—	0	0	0	0	1	0	0
DMA Control Unit								
DCH	—	—	—	0	0	0	0	1
DMD	0	0	0	0	0	0	—	0
DDC (8H)	—	—	0	0	—	0	—	—
DDC (9H)	—	—	—	—	—	—	0	0
DST	0	0	0	0	0	0	0	0
DMK	—	—	—	—	1	1	1	1

INSTRUCTION SET HIGHLIGHTS

Enhanced Instructions

In addition to the μPD8088/86 instructions, the μPD70236 has enhanced instructions listed in table 8.

Table 8. Enhanced Instruction

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP R	Pops 8 general registers onto stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Enhanced Stack Operation Instructions

PUSH imm. This instruction allows immediate data to be pushed onto the stack.

PUSH R; POP R. These instructions allow the contents of the eight general registers to be pushed onto or popped from the stack with a single instruction.

Enhanced Multiplication Instructions

MUL reg16, imm16; MUL mem16, imm16. These instructions allow the contents of a register or memory location to be multiplied by immediate data.

Enhanced Shift and Rotate Instructions

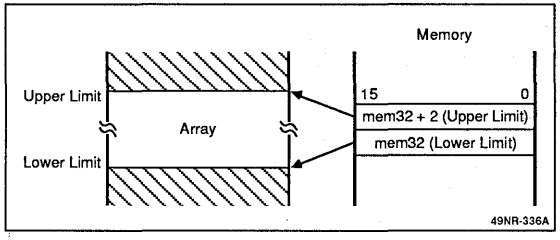
SHL reg, imm8; SHR reg, imm8; SHRA reg, imm8. These instructions allow the contents of a register to be shifted by the number of bits defined by the immediate data.

ROL reg, imm8; ROR reg imm8; ROLC reg, imm8; RORC reg, imm8. These instructions allow the contents of a register to be rotated by the number of bits defined by the immediate data.

Check Array Boundary Instruction

CHKIND reg16, mem32. This instruction is used to verify that index values pointing to the elements of an array data structure are within the defined range. See figure 113. The lower limit of the array should be in memory location mem32, the upper limit in mem32 + 2. If the index value in reg16 is not between these limits when CHKIND is executed, a BRK 5 will occur. This causes a jump to the location in interrupt vector 5.

Figure 113. Check Array Boundary



Block I/O Instruction

OUTM DW, src-block; INM dist-block, DW. These instructions are used to output or input a string to or from memory, when preceded by a repeat prefix.

Stack Frame Instruction

PREPARE imm16,Imm8. This instruction is used to generate the stack frames required by block-structured languages, such as PASCAL and Ada. The stack frame consists of two areas. One area has a pointer that points to another frame which has variables that the current frame can access. The other is a local variable area for the current procedure.

DISPOSE. This instruction releases that last stack frame generated by the PREPARE instruction. It returns the stack and base pointers to the values they had before the PREPARE instruction was used to call a procedure.

Unique Instructions

In addition to the μPD8088/86 instructions and the enhanced instructions, the μPD70236 has the unique instructions listed in table 9.

3f

Table 9. Unique Instructions

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	Adds packed decimal strings
SUB4S	Subtracts one packed decimal string from another
CMP4S	Compares two packed decimal strings
ROL4	Rotates one BCD digit left through AL lower 4 bits
ROR4	Rotates one BCD digit right through AL lower 4 bits
BRKXA	Break and enable expanded addressing
RETXA	Return from break and disable expanded addressing
TEST1	Tests a specified bit and sets/resets Z flag
NOT1	Inverts a specified bit
CLR1	Clears a specified bit
SET1	Sets a specified bit
REPC	Repeats next instruction until CY flag is cleared
REPNC	Repeats next instruction until CY flag is set
FP02	Additional floating-point processor call

Variable Length Bit Field Operation Instructions

This category has two instructions: INS (Insert Bit Field) and EXT (Extract Bit Field). These instructions are highly effective for computer graphics and high-level languages. They can, for example, be used for data structures such as packed arrays and record type data used in PASCAL.

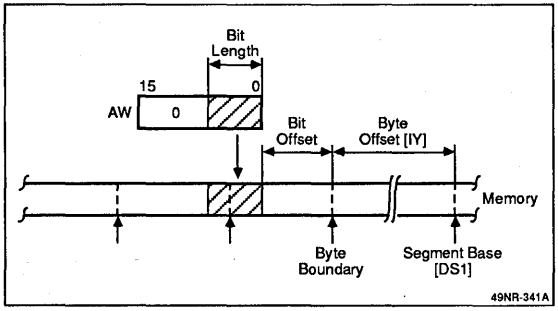
INS reg8, reg8; INS reg8, imm4. This instruction transfers low bits from the 16-bit AW register (the number of bits is specified by the second operand) to the memory location specified by the segment base (DS1 register) plus the byte offset (IY register). The starting bit position within this byte is specified as an offset by the lower 4 bits of the first operand. See figure 114.

After each complete data transfer, the IY register and the register specified by the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may specify the number of bits transferred (second operand). Because the maximum transferable bit length is 16 bits, only the lower 4 bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Figure 114. Bit Field Insertion



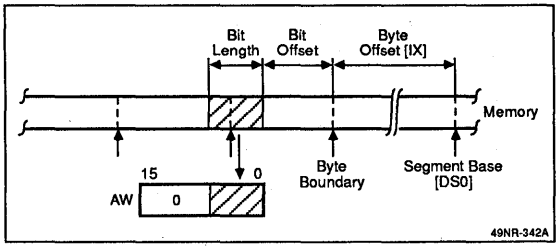
EXT reg8, reg8; EXT reg8, imm4. This instruction loads to the AW registers the bit field data whose bit length is specified by the second operand of the instruction from the memory location that is specified by the DS0 segment register (segment base), the IX index register (byte offset), and the lower 4 bits of the first operand (bit offset). See figure 115.

After the transfer is complete, the IX register and the lower 4 bits of the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may be specified for the second operand. Because the maximum transferable bit length is 16 bits, however, only the lower 4 bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Figure 115. Bit Field Extraction



Packed BCD Operation Instructions

The instructions described here process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte-format operands (ROR4, ROL4). Packed BCD strings may be from 1 to 254 digits in length.

When the number of digits is even, the zero (Z) and carry (CY) flags will be set according to the result of the operation. When the number of digits is odd, the Z and CY flags may not be set correctly. In this case (CL = odd), the Z flag will not be set unless the upper 4 bits of the highest byte are all 0s. The CY flag will not be set unless there is a carry out of the upper 4 bits of the highest byte. When CL is odd, the contents of the upper 4 bits of the highest byte of the result are undefined.

ADD4S. This instruction adds the packed BCD string addressed by the IX index register to the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the V (overflow), CY, and Z flags .

BCD string (IY, CL) ← BCD string (IY, CL) + BCD string (IX, CL)

SUB4S. This instruction subtracts the packed BCD string addressed by the IX index register from the packed BCD string addressed by the IY register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the V, CY, and Z flags.

BCD string (IY, CL) ← BCD string (IY, CL) – BCD string (IX, CL)

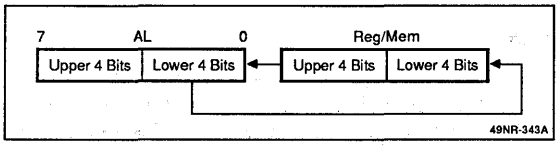
CMP4S. This instruction performs the same operation as SUB4S except that the result is not stored and only the V, CY, and Z flags are affected.

BCD string (IY, CL) – BCD string (IX, CL)

ROL4. This instruction treats the byte data of the register or memory operand specified by the instruction as

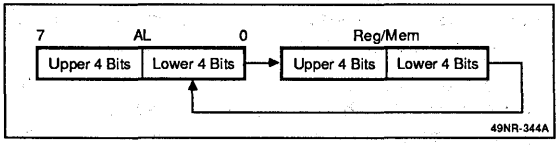
BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the left. See figure 116.

Figure 116. BCD Rotate Left



ROR4. This instruction treats the byte data of the register or memory specified by the instruction as BCD data and uses the lower 4 bits of the AL register (AL_L) to rotate that data one BCD digit to the right. See figure 117.

Figure 117. BCD Rotate Right



Bit Manipulation Instructions

TEST1. This instruction tests a specific bit in a register or memory location. If the bit is 1, the Z flag is reset to 0. If the bit is 0, the Z flag is set to 1.

NOT1. This instruction inverts a specific bit in a register or memory location.

CLR1. This instruction clears a specific bit in a register or memory location.

SET1. This instruction sets a specific bit in a register or memory location.

Repeat Prefix Instructions

REPC. This instruction causes the μPD70236 to repeat the following primitive block transfer instruction until the CY flag becomes cleared or the CW register becomes zero.

REPNC. This instruction causes the μPD70236 to repeat the following primitive block transfer instruction until the CY flag becomes set or the CW register becomes zero.

Address Expansion Control Instructions

BRKXA imm8. This instruction is used to turn on expanded addressing. The 8-bit immediate data specifies an interrupt vector. The PC field of this vector is loaded into the PC (and PFP). The XA flag in the XAM register is set to 1, thereby enabling the expanded addressing mode. The μPD70236 will begin fetching from the new PFP through the address translation table. That is, the new PC is treated as a logical address and is translated to the new, larger physical address space.

This instruction does not save any return address information, such as PC, PS, or PSW to the stack.

RETXA imm8. This instruction is used to turn off expanded addressing. It is identical in operation to BRKXA, except that the expanded addressing mode is turned off before fetching from the new address. That is, the XA flag in the XAM register is set to 0, and the PC is loaded with the value of the PC field in the interrupt vector selected by the immediate data.

This instruction does not save any return address information such as PC, PS, or PSW to the stack.

Porting μPD70116/70108 Code to μPD70236

3f

The μPD70236 is completely software compatible with the μPD70116/70108. However, the μPD70236 offers some improvements that may affect the porting of μPD70116 code to the μPD70236. These improvements are:

(1) The μPD70116 does not trap on undefined opcodes. The μPD70236 will trap, and also will trap when a register addressing mode is used for any of these instructions:

CHKIND	LDEA
MOV DS0/DS1	BR 1,id
CALL 1,id	

(2) During signed division (DIV), if the quotient is 80H (byte operation) or 8000H (word), the μPD70116 will take a Divide By 0 trap. The μPD70236 will perform the calculation.

(3) When the μPD70116 executes the POLL instruction, it will wait for the POLL input signal to be asserted. The μPD70236 has no POLL input; instead, when this instruction is executed, if a coprocessor is not connected, then a Coprocessor Not Present trap will be taken. If a coprocessor is attached, then no operation takes place.

The μPD70116 accepts FP01 and FP02 as opcodes for the iAPX8087 coprocessor. The μPD70236 accepts these as opcodes for the μPD72291 coprocessor, which is not compatible with the iAPX8087.

(4) During the POP R instruction, the μPD70116 does not restore the SP register. The μPD70236 does restore the SP.

(5) When processing a divide error, the μPD70116 saves the address of the next instruction. The μPD70236 saves the address of the current instruction (the divide instruction).

(6) The μPD70116 allows up to three prefix instructions in any combination. The μPD70236 also allows three prefixes, but only one of each type can be used. The μPD70236 could operate incorrectly if there are two prefixes of the same type. For example, consider:

REP
REPC
CMPBK SS: src-block, dst-block

If the compare operation is interrupted, then when it resumes following the interrupt service, execution will begin at the REPC instruction, not the REP instruction, because two repeat prefixes were used.

(7) The μPD70116 accepts $\overline{\text{NMI}}$ requests even while processing an NMI. The μPD70236 does not allow nesting of NMIs; the $\overline{\text{NMI}}$ input will be ignored until the NMI interrupt handler is exited.

INSTRUCTION SET

Symbols

Preceding the instruction set, several tables explain symbols, abbreviations, and codes.

Clocks

In the Clocks column of the instruction set, the numbers cover these operations: instruction decoding, effective address calculation, operand fetch, and instruction execution.

Clock timings assume the instruction has been prefetched and is present in the 8-byte instruction queue. Otherwise, add two clocks for each pair of bytes not present.

Word operands require two additional clocks for each transfer to an unaligned (odd address) memory operand. These times are shown on the right side of the slash (/).

For conditional control transfer or branch instructions, the number on the left side of the slash is applicable if the transfer or branch takes place. The number on the right side is applicable if it does not take place.

If a range of numbers is given, the execution time depends on the operands involved.

Symbols

Symbol	Meaning
acc	Accumulator(AW or AL)
duso	Displacement (8 or 16 bits)
dmem	Direct memory address
dst	Destination operand or address
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
far_label	Label within a different program segment
far_proc	Procedure within a different program segment
fp_op	Floating-point instruction operation
imm	8- or 16-bit immediate operand
imm3/4	3- or 4-bit immediate bit offset
imm8	8-bit immediate operand
imm16	16-bit immediate operand
mem	Memory field (000 to 111); 8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
memptr16	Word containing the destination address within the current segment
memptr32	Double word containing a destination address in another segment
mod	Mode field (00 to 10)
near_label	Label within the current segment
near_proc	Procedure within the current segment
offset	Immediate offset data (16 bits)
pop_value	Number of bytes to discard from the stack
reg	Register field (000 to 111); 8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
regptr	16-bit register containing a destination address within the current segment
regptr16	Register containing a destination address within the current segment
seg	Immediate segment data (16 bits)
short_label	Label between −128 and +127 bytes from the end of the current instruction
sr	Segment register
src	Source operand or address
temp	Temporary register (8/16/32 bits)
AC	Auxiliary carry flag
AH	Accumulator (high byte)

Symbol	Meaning
AL	Accumulator (low byte)
AW	Accumulator (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
BP	BP register
BRK	Break flag
BW	BW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
CW	CW register (16 bits)
CY	Carry flag
DH	DW register (high byte)
DIR	Direction flag
DL	DW register (low byte)
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DW	DW register (16 bits)
IE	Interrupt enable flag
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
MD	Mode flag
P	Parity flag
PC	Program counter (16 bits)
PS	Program segment register (16 bits)
PSW	Program status word (16 bits)
R	Register set
S	Sign extend operand field S = No sign extension S = Sign extend immediate byte operand
S	Sign flag
SP	Stack pointer (16 bits)
SS	Stack segment register (16 bits)
V	Overflow flag
W	Word/byte field (0 to 1)
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating-point arithmetic chip
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value
Z	Zero flag

Flag Operations

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to result
u	Undefined
R	Restored to previous state

Memory Addressing Modes

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Register Selection (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Register Selection

sr	Segment Register
00	DS1
01	PS
10	SS
11	DS0

Instruction Set

		Opcode																		Flags					
Mnemonic	Operand	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	AC	CY	V	P	S	Z
Data Transfer Instructions																									
MOV	reg, reg	1	0	0	0	1	0	1	W	1	1		reg		reg			2	2						
	mem, reg	1	0	0	0	1	0	0	W	mod			reg		mem			3/5	2-4						
	reg, mem	1	0	0	0	1	0	1	W	mod			reg		mem			5/7	2-4						
	mem, imm	1	1	0	0	0	1	1	W	mod			000		mem			3/5	3-6						
	reg, imm	1	0	1	1	W		reg										2	2-3						
	acc, dmem	1	0	1	0	0	0	0	W									5/7	3						
	dmem, acc	1	0	1	0	0	0	1	W									3/5	3						
	sr, reg16	1	0	0	0	1	1	1	0	1	1	0	sr		reg			2	2						
	sr, mem16	1	0	0	0	1	1	1	0	mod		0	sr		mem			5/7	2-4						
	reg16, sr	1	0	0	0	1	1	0	0	1	1	0	sr		reg			2	2						
	mem16, sr	1	0	0	0	1	1	0	0	mod		0	sr		mem			3/5	2-4						
	DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod			reg		mem			10/14	2-4						
	DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod			reg		mem			10/14	2-4						
	AH, PSW	1	0	0	1	1	1	1	1									2	1						
	PSW, AH	1	0	0	1	1	1	1	0									2	1	x	x		x	x	x
LDEA	reg16, mem16	1	0	0	0	1	1	0	0	mod			reg		mem			2	2-4						
TRANS	src_table	1	1	0	1	0	1	1	1									5	1						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer Instructions (cont)																									
XCH	reg, reg	1	0	0	0	0	1	1	W	1	1		reg			reg		3	2						
	mem, reg	1	0	0	0	0	1	1	W	mod			reg			mem		8/12	2-4						
	AW, reg16	1	0	0	1	0		reg										3	1						
Repeat Prefixes																									
REPC		0	1	1	0	0	1	0	1									2	1						
REPNC		0	1	1	0	0	1	0	0									2	1						
REP REPE REPZ		1	1	1	1	0	0	1	1									2	1						
REPNE REPNZ		1	1	1	1	0	0	1	0									2	1						
Block Transfer Instructions																									
MOVBK	dst, src	1	0	1	0	0	1	0	W										1						
																		3 + 4n (W = 0) 3 + 4n (W = 1, even addresses) 3 + 8n (W = 1, odd addresses) 3 + 6n (W = 1, odd/even addresses)							
CMPBK	dst, src	1	0	1	0	0	1	1	W										1	x	x	x	x	x	x
																		3 + 7n (W = 0) 3 + 7n (W = 1, even addresses) 3 + 11n (W = 1, odd addresses) 3 + 9n (W = 1, odd/even addresses)							
CMPM	dst	1	0	1	0	1	1	1	W										1	x	x	x	x	x	x
																		3 + 5n (W = 0) 3 + 5n (W = 1, even addresses) 3 + 7n (W = 1, odd addresses)							
LDM	src	1	0	1	0	1	1	0	W										1						
																		5 + 2n (W = 0) 5 + 2n (W = 1, even addresses) 5 + 4n (W = 1, odd addresses)							
STM	dst	1	0	1	0	1	0	1	W										1						
																		3 + 2n (W = 0) 3 + 2n (W = 1, even addresses) 3 + 4n (W = 1, odd addresses)							

n = number of returns
String instruction execution clocks for a single-instruction execution are in parentheses.

3f

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
I/O Instructions (cont)																									
IN	acc, imm8	1	1	1	0	0	1	0	W									5/7	2						
	acc, DW	1	1	1	0	1	1	0	W									3/5	1						
OUT	imm8, acc	1	1	1	0	0	1	1	W									3/5	2						
	DW, acc	1	1	1	0	1	1	1	W									3/5	1						
INM	dst, DW	0	1	1	0	1	1	0	W										1						
																		3 + 11n (W = 0) 3 + 8n (W = 1, even addresses) 3 + 22n (W = 1, odd addresses) 3 +20n (W = 1, odd/even addresses; odd for I/O) 3 + 13n (W = 1, odd/even addresses; odd for memory)							
OUTM	DW, src	0	1	1	0	1	1	1	W										1						
																		3 + 11n (W = 0) 3 + 8n (W = 1, even addresses) 3 + 22n (W = 1, odd addresses) 3 + 20n (W = 1, odd addresses; odd for I/O) 3 + 13n (W = 1, odd addresses; odd for memory)							

n = number of transfers
String instruction execution clocks for a single-instruction execution are in parentheses.
Use the right side of the slash (/) for DMA I/O accesses.

Mnemonic	Operand	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	AC	CY	V	P	S	Z
BCD Instructions																									
ADJBA		0	0	1	1	0	1	1	1									4	1	x	x	u	u	u	u
ADJ4A		0	0	1	0	0	1	1	1									2	1	x	x	u	x	x	x
ADJBS		0	0	1	1	1	1	1	1									4	1	x	x	u	u	u	u
ADJ4S		0	0	1	0	1	1	1	1									2	1	x	x	u	x	x	x
ADD4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	2 + 18n	2	u	x	u	u	u	x
SUB4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	2 + 18n	2	u	x	u	u	u	x
CMP4S	dst, src	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	7 + 14n	2	u	x	u	u	u	x
ROL4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	9	3						
		1	1	0	0	0		reg																	
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	15	3-5						
		mod		0	0	0		mem																	
ROR4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	13	3						
		1	1	0	0	0		reg																	
	mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	19	3-5						
		mod		0	0	0		mem																	

n = number of BCD digits divided by 2

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Data Type Conversion Instructions																									
CVTBD		1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	12	2	u	u	u	x	x	x
CVTDB		1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	8	2	u	u	u	x	x	x
CVTBW		1	0	0	1	1	0	0	0									2	1						
CVTWL		1	0	0	1	1	0	0	1									2	1						
Arithmetic Instructions																									
ADD	reg, reg	0	0	0	0	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	0	0	0	0	W	mod			reg			mem		7/11	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	0	0	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	0	0		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	0	0		mem		7/11	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	0	0	1	0	W									2	2-3	x	x	x	x	x	x
ADDC	reg, reg	0	0	0	1	0	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	0	0	0	W	mod			reg			mem		7/11	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	0	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	0		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	0		mem		7/11	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	0	1	0	W									2	2-3	x	x	x	x	x	x
SUB	reg, reg	0	0	1	0	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	1	0	1	0	0	W	mod			reg			mem		7/11	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	0	1	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	0	1		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	0	1		mem		7/11	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	0	1	1	0	W									2	2-3	x	x	x	x	x	x
SUBC	reg, reg	0	0	0	1	1	0	1	W	1	1		reg			reg		2	2	x	x	x	x	x	x
	mem, reg	0	0	0	1	1	0	0	W	mod			reg			mem		7/11	2-4	x	x	x	x	x	x
	reg, mem	0	0	0	1	1	0	1	W	mod			reg			mem		6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	1		reg		2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		0	1	1		mem		7/11	3-6	x	x	x	x	x	x
	acc, imm	0	0	0	1	1	1	0	W									2	2-3	x	x	x	x	x	x
INC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	0		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	0		mem		7/11	2-4	x		x	x	x	x
	reg16	0	1	0	0	0		reg										2	1	x		x	x	x	x
DEC	reg8	1	1	1	1	1	1	1	0	1	1	0	0	1		reg		2	2	x		x	x	x	x
	mem	1	1	1	1	1	1	1	W	mod		0	0	1		mem		7/11	2-4	x		x	x	x	x
	reg16	0	1	0	0	1		reg										2	1	x		x	x	x	x
MULU	reg8	1	1	1	1	0	1	1	0	1	1	1	0	0		reg		8	2	u	x	x	u	u	u
	reg16	1	1	1	1	0	1	1	1	1	1	1	0	0		reg		12	2	u	x	x	u	u	u
	mem8	1	1	1	1	0	1	1	0	mod		1	0	0		mem		12	2-4	u	x	x	u	u	u
	mem16	1	1	1	1	0	1	1	1	mod		1	0	0		mem		16/18	2-4	u	x	x	u	u	u

3f

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Arithmetic Instructions (cont)																									
MUL	reg8	1	1	1	1	0	1	1	0	1	1	1	0	1	reg			8	2	u	x	x	u	u	u
	reg16	1	1	1	1	0	1	1	1	1	1	1	0	1	reg			12	2	u	x	x	u	u	u
	mem8	1	1	1	1	0	1	1	0	mod		1	0	1	mem			12	2-4	u	x	x	u	u	u
	mem16	1	1	1	1	0	1	1	1	mod		1	0	1	mem			16/18	2-4	u	x	x	u	u	u
	reg16, reg16, imm8	0	1	1	0	1	0	1	1	1	1		reg		reg			12	3	u	x	x	u	u	u
	reg16, mem16, imm8	0	1	1	0	1	0	1	1	mod			reg		mem			16/18	3-5	u	x	x	u	u	u
	reg16, reg16, imm16	0	1	1	0	1	0	0	1	1	1		reg		reg			12	4	u	x	x	u	u	u
	reg16, mem16, imm16	0	1	1	0	1	0	0	1	mod			reg		mem			16/8	4-6	u	x	x	u	u	u
DIVU	reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg			11	2	u	u	u	u	u	u
	reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg			19	2	u	u	u	u	u	u
	mem8	1	1	1	1	0	1	1	0	mod		1	1	0	mem			15	2-4	u	u	u	u	u	u
	mem16	1	1	1	1	0	1	1	1	mod		1	1	0	mem			23/25	2-4	u	u	u	u	u	u
DIV	reg8	1	1	1	1	0	1	1	0	1	1	1	1	1	reg			16	2	u	u	u	u	u	u
	reg16	1	1	1	1	0	1	1	1	1	1	1	1	1	reg			24	2	u	u	u	u	u	u
	mem8	1	1	1	1	0	1	1	0	mod		1	1	1	mem			20	2-4	u	u	u	u	u	u
	mem16	1	1	1	1	0	1	1	1	mod		1	1	1	mem			28/30	2-4	u	u	u	u	u	u
Comparison Instructions																									
CMP	reg, reg	0	0	1	1	1	0	1	W	1	1		reg		reg			2	2	x	x	x	x	x	x
	mem, reg	0	0	1	1	1	0	0	W	mod			reg		mem			6/8	2-4	x	x	x	x	x	x
	reg, mem	0	0	1	1	1	0	1	W	mod			reg		mem			6/8	2-4	x	x	x	x	x	x
	reg, imm	1	0	0	0	0	0	S	W	1	1	1	1	1	reg			2	3-4	x	x	x	x	x	x
	mem, imm	1	0	0	0	0	0	S	W	mod		1	1	1	mem			6/8	3-6	x	x	x	x	x	x
	acc, imm	0	0	1	1	1	1	0	W									2	2-3	x	x	x	x	x	x
Logical Instructions																									
NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg			2	2						
	mem	1	1	1	1	0	1	1	W	mod		0	1	0	mem			7/11	2-4						
NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1	reg			2	2	x	x	x	x	x	x
	mem	1	1	1	1	0	1	1	W	mod		0	1	1	mem			7/11	2-4	x	x	x	x	x	x
TEST	reg, reg	1	0	0	0	0	1	0	W	1	1		reg		reg			2	2	u	0	0	x	x	x
	mem, reg	1	0	0	0	0	1	0	W	mod			reg		mem			6/8	2-4	u	0	0	x	x	x
	reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg			2	3-4	u	0	0	x	x	x
	mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			6/8	3-6	u	0	0	x	x	x
	acc, imm	1	0	1	0	1	0	0	W									2	2-3	u	0	0	x	x	x
AND	reg, reg	0	0	1	0	0	0	1	W	1	1		reg		reg			2	2	u	0	0	x	x	x
	mem, reg	0	0	1	0	0	0	0	W	mod			reg		mem			7/11	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	0	0	0	1	W	mod			reg		mem			6/8	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg			2	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem			7/11	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									2	2-3	u	0	0	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Logical Instructions (cont)																									
OR	reg, reg	0	0	0	0	1	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	0	0	1	0	0	W	mod			reg			mem		7/11	2-4	u	0	0	x	x	x
	reg, mem	0	0	0	0	1	0	1	W	mod			reg			mem		6/8	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	0	0	1		reg		2	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		0	0	1		mem		7/11	3-6	u	0	0	x	x	x
	acc, imm	0	0	0	0	1	1	0	W									2	2-3	u	0	0	x	x	x
XOR	reg, reg	0	0	1	1	0	0	1	W	1	1		reg			reg		2	2	u	0	0	x	x	x
	mem, reg	0	0	1	1	0	0	0	W	mod			reg			mem		7/11	2-4	u	0	0	x	x	x
	reg, mem	0	0	1	1	0	0	1	W	mod			reg			mem		6/8	2-4	u	0	0	x	x	x
	reg, imm	1	0	0	0	0	0	0	W	1	1	1	1	0		reg		2	3-4	u	0	0	x	x	x
	mem, imm	1	0	0	0	0	0	0	W	mod		1	1	0		mem		7/11	3-6	u	0	0	x	x	x
	acc, imm	0	0	1	1	0	1	0	W									2	2-3	u	0	0	x	x	x
Bit Manipulation Instructions																									
INS	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	37-61/	3						
		1	1		reg			reg										39-77							
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	37-69/	4						
		1	1	0	0	0		reg										39-77							
EXT	reg8, reg8	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	29-61/	3						
		1	1		reg			reg										33-63							
	reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	29-61/	4						
		1	1	0	0	0		reg										33-63							
TEST1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	W	4	3	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	8	3-5	u	0	0	u	u	x
		mod		0	0	0		mem																	
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1	8/10	3-5	u	0	0	u	u	x
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	W	4	4	u	0	0	u	u	x
		1	1	0	0	0		reg																	
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	0	13	4-6	u	0	0	u	u	x
		mod		0	0	0		mem																	
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1	8/10	4-6	u	0	0	u	u	x
		mod		0	0	0		mem																	
SET1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	4	3						
		1	1	0	0	0		reg																	
	mem, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	W	9	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	W	4	4						
		1	1	0	0	0		reg																	

3f

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Bit Manipulation Instructions (cont)																									
SET1 (cont)	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	9	4-6						
		mod		0	0	0		mem																	
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	1	9/13	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	1									2	1		1				
	DIR	1	1	1	1	1	1	0	1									2	1						
CLR1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	W	4	3						
		1	1	0	0	0		reg																	
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	0	9	3-5						
		mod		0	0	0		mem																	
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1	9/13	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	W	4	4						
		1	1	0	0	0		reg																	
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0	9	4-6						
		mod		0	0	0		mem																	
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1	9/13	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	1	0	0	0									2	1		0				
	DIR	1	1	1	1	1	1	0	0									2	1						
NOT1	reg, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	W	4	3						
		1	1	0	0	0		reg																	
	mem8, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0	9	3-5						
		mod		0	0	0		mem																	
	mem16, CL	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	9/13	3-5						
		mod		0	0	0		mem																	
	reg, imm3/4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	4	4						
		1	1	0	0	0		reg																	
	mem8, imm3	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	W	9	4-6						
		mod		0	0	0		mem																	
	mem16, imm4	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1	9/13	4-6						
		mod		0	0	0		mem																	
	CY	1	1	1	1	0	1	0	1									2	1		x				

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions																									
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0	mem			7/11	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0	reg			2 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0	mem			6/10 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0	reg			2 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0	mem			6/10 + n	3-5	u	x	u	x	x	x
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg			2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1	mem			7/11	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg			2 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1	mem			6/10 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg			2 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1	mem			6/10 + n	3-5	u	x	u	x	x	x
SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1	reg			2	2	u	x	0	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1	mem			7/11	2-4	u	x	0	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1	reg			2 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1	mem			6/10 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1	reg			2 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1	mem			6/10 + n	3-5	u	x	u	x	x	x
ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	0	mem			7/11	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0	reg			2 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	0	mem			6/10 + n	2-4		x	u			
	reg, imm	1	1	0	0	0	0	0	W	1	1	0	0	0	reg			2 + n	3		x	u			
	mem, imm	1	1	0	0	0	0	0	W	mod		0	0	0	mem			6/10 + n	3-5		x	u			
ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1	reg			2 + n	2		x	u			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	0	1	mem			7/11	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1	reg			7 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	0	1	mem			6/10 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1	reg			2 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	0	1	mem			6/10 + n	3-5		x	u			
ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0	mem			7/11	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0	reg			2 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0	mem			6/10 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0	reg			2 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0	mem			6/10 + n	3-5		x	u			

n = number of shifts

3f

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Shift/Rotate Instructions (cont)																									
RORC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	1	reg			2	2		x	x			
	mem, 1	1	1	0	1	0	0	0	W	mod		0	1	1	mem			7/11	2-4		x	x			
	reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	1	reg			2 + n	2		x	u			
	mem, CL	1	1	0	1	0	0	1	W	mod		0	1	1	mem			6/10 + n	2-4		x	u			
	reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	1	reg			2 + n	3		x	u			
	mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	1	mem			6/10 + n	3-5		x	u			
																		n = number of shifts							
Stack Manipulation Instructions																									
PUSH	mem16	1	1	1	1	1	1	1	1	mod		1	1	0	mem			5/9	2-4						
	reg16	0	1	0	1	0		reg										3/5	1						
	sr	0	0	0	sr		1	1	0									3/5	1						
	PSW	1	0	0	1	1	1	0	0									3/5	1						
	R	0	1	1	0	0	0	0	0									20/36	1						
	imm	0	1	1	0	1	0	S	0									3/5	2-3						
POP	mem16	1	0	0	0	1	1	1	1	mod		0	0	0	mem			5/9	2-4						
	reg16	0	1	0	1	1		reg										5/7	1						
	sr	0	0	0	sr		1	1	1									5/7	1						
	PSW	1	0	0	1	1	1	0	1									5/7	1	R	R	R	R	R	R
	R	0	1	1	0	0	0	0	1									22/38	1						
PREPARE	imm16, imm8	1	1	0	0	1	0	0	0									*	4						
										*imm8 = 0:15 imm8 ≥ 1: 17 + 12 (imm8 – 1) odd, 15 + 8 (imm8-1) even														
DISPOSE		1	1	0	0	1	0	0	1									6/10	1						
Control Transfer Instructions																									
CALL	near_proc	1	1	1	0	1	0	0	0									7/9	3						
	regptr16	1	1	1	1	1	1	1	1	1	1	0	1	0	reg			7/9	2						
	memptr16	1	1	1	1	1	1	1	1	mod		0	1	0	mem			11/15	2-4						
	far_proc	1	0	0	1	1	0	1	0									9/13	5						
	memptr32	1	1	1	1	1	1	1	1	mod		0	1	1	mem			15/23	2-4						
RET		1	1	0	0	0	0	1	1									10/12	1						
	pop_value	1	1	0	0	0	0	1	0									10/12	3						
		1	1	0	0	1	0	1	1									12/16	1						
	pop_value	1	1	0	0	1	0	1	0									12/16	3						
BR	near_label	1	1	1	0	1	0	0	1									7	3						
	short_label	1	1	1	0	1	0	1	1									7	2						
	regptr16	1	1	1	1	1	1	1	1	1	1	1	0	0	reg			7	2						
	memptr16	1	1	1	1	1	1	1	1	mod		1	0	0	mem			11/13	2-4						
	far_label	1	1	1	0	1	0	1	0									7	5						
	memptr32	1	1	1	1	1	1	1	1	mod		1	0	1	mem			13/17	2-4						

Instruction Set (cont)

Mnemonic	Operand	Opcode 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Clocks	Bytes	Flags AC	CY	V	P	S	Z
Control Transfer Instructions (cont)																									
BV	short_label	0	1	1	1	0	0	0	0									3/6	2						
BNV	short_label	0	1	1	1	0	0	0	1									3/6	2						
BC, BL	short_label	0	1	1	1	0	0	1	0									3/6	2						
BNC, BNL	short_label	0	1	1	1	0	0	1	1									3/6	2						
BE, BZ	short_label	0	1	1	1	0	1	0	0									3/6	2						
BNE, BNZ	short_label	0	1	1	1	0	1	0	1									3/6	2						
BNH	short_label	0	1	1	1	0	1	1	0									3/6	2						
BH	short_label	0	1	1	1	0	1	1	1									3/6	2						
BN	short_label	0	1	1	1	1	0	0	0									3/6	2						
BP	short_label	0	1	1	1	1	0	0	1									3/6	2						
BPE	short_label	0	1	1	1	1	0	1	0									3/6	2						
BPO	short_label	0	1	1	1	1	0	1	1									3/6	2						
Interrupt Instructions																									
BLT	short_label	0	1	1	1	1	1	0	0									3/6	2						
BGE	short_label	0	1	1	1	1	1	0	1									3/6	2						
BLE	short_label	0	1	1	1	1	1	1	0									3/6	2						
BGT	short_label	0	1	1	1	1	1	1	1									3/6	2						
DBNZNE	short_label	1	1	1	0	0	0	0	0									3/6	2						
DBNZE	short_label	1	1	1	0	0	0	0	1									3/6	2						
DBNZ	short_label	1	1	1	0	0	0	1	0									3/6	2						
BCWZ	short_label	1	1	1	0	0	0	1	1									3/6	2						
BRK	3	1	1	0	0	1	1	0	0									18/24	1						
	imm8	1	1	0	0	1	1	0	1									18/24	2						
BRKV	imm8	1	1	0	0	1	1	1	0									20/26	1						
RETI		1	1	0	0	1	1	1	1									13/19	1	R	R	R	R	R	R
CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	mod			reg			mem		24-26/ 30-32	2-4						
CPU Control Instructions																									
HALT		1	1	1	1	0	1	0	0									2	1						
BUSLOCK		1	1	1	1	0	0	0	0									2	1						
FP01	fp_op	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	*	2						
	fp_op, mem	1	1	0	1	1	X	X	X	mod		Y	Y	Y		mem		*	2-4						
FP02	fp_op	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	*	2						
	fp_op, mem	0	1	1	0	0	1	1	X	mod		Y	Y	Y		mem		*	2-4						
POLL		1	0	0	1	1	0	1	1									2 + 5n	1						
													n = number of times POLL pin is sampled.												
NOP		1	0	0	1	0	0	0	0									3	1						
DI		1	1	1	1	1	0	1	0									2	1						
EI		1	1	1	1	1	0	1	1									2	1						

3f

Instruction Set (cont)

		Opcode																		Flags					
Mnemonic	**Operand**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Clocks**	**Bytes**	**AC**	**CY**	**V**	**P**	**S**	**Z**
CPU Control Instructions (cont)																									
DS0:, DS1:, PS:, SS: (segment override prefixes)		0	0	1	seg		1	1	0									2	1						
Address Expansion Control Instructions																									
BRKXA	imm8	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	12	3						
				imm8																					
RETXA	imm8	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	12	3						
				imm8																					

NEC

Section 4
16-Bit Microcomputers

μPD70320/70322 (V25) 16-Bit Microcomputers: Single-Chip, CMOS

Description

The μPD70320 and μPD70322 (V25™) are high-performance, 16-bit, single-chip microcomputers with an 8-bit external data bus. They combine the instruction set of the μPD70108 (V20™) with many of the on-chip peripherals in NEC's 78000 series.

The μPD70320/322 processor has software compatibility with the V20 (and subsequently the 8086/8088), faster memory accessing, superior interrupt processing ability, and enhanced control of internal peripherals.

A variety of on-chip components, including 16K bytes of mask programmable ROM (μPD70322 only), 256 bytes of RAM, serial and parallel I/O, comparator port lines, timers, and a DMA controller make the μPD70320/322 a sophisticated microsystem.

Eight banks of registers are mapped into internal RAM below an additional 256-byte special function register (SFR) area that is used to control on-chip peripherals. Internal RAM and the SFR area are together relocatable to anywhere in the 1M-byte address space. This maintains compatibility with existing system memory maps.

The μPD70322 is the mask ROM version, the μPD70320 is the ROM-less version, and the μPD70P322 is the EPROM version.

Features

- ☐ Complete single-chip microcomputer
  - — 16-bit ALU
  - — 16K bytes of ROM (μPD70322)
  - — 256 bytes of RAM
- ☐ 6-byte instruction prefetch queue
- ☐ 24 parallel I/O lines
- ☐ Eight analog comparator inputs with programmable threshold level
- ☐ Two independent DMA channels
- ☐ Two 16-bit timers
- ☐ Programmable time base counter
- ☐ Two full-duplex UARTs
- ☐ Programmable interrupt controller
  - — Eight priority levels
  - — Five external, 12 internal sources
  - — Register bank (eight) context switching
  - — Eight macro service function channels
- ☐ DRAM refresh pulse output
- ☐ Two standby modes
  - — HALT
  - — STOP
- ☐ Internal clock generator
  - — 5-MHz maximum CPU clock frequency (0.4-μs instruction cycle time)
  - — 8-MHz maximum CPU clock frequency (0.25-μs instruction cycle time)
- ☐ Programmable wait state generation
- ☐ Separate address/data bus interface
- ☐ CMOS technology

V20 and V25 are trademarks of NEC Corporation.

Ordering Information

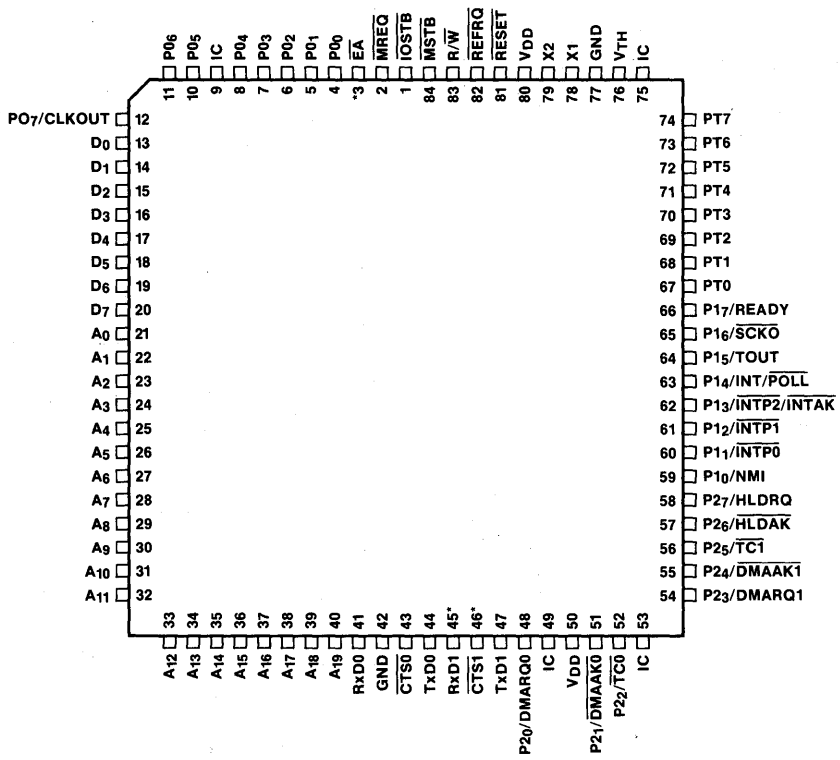
Part Number	Clock (MHz)	Package Type	ROM
μPD70320L	5	84-pin PLCC	ROM-less
L-8	8		
GJ	5	94-pin plastic QFP	
GJ-8	8		
μPD70322L-xxx	5	84-pin PLCC	Mask ROM
L-8-xxx	8		
GJ-xxx	5	94-pin plastic QFP	
GJ-8-xxx	8		
μPD70P322KE-8	8	84-pin LCC	EPROM (UV erasable)

PLCC = plastic leaded chip carrier

LCC = ceramic leadless chip carrier (with window)

Pin Configurations

84-Pin PLCC and LCC

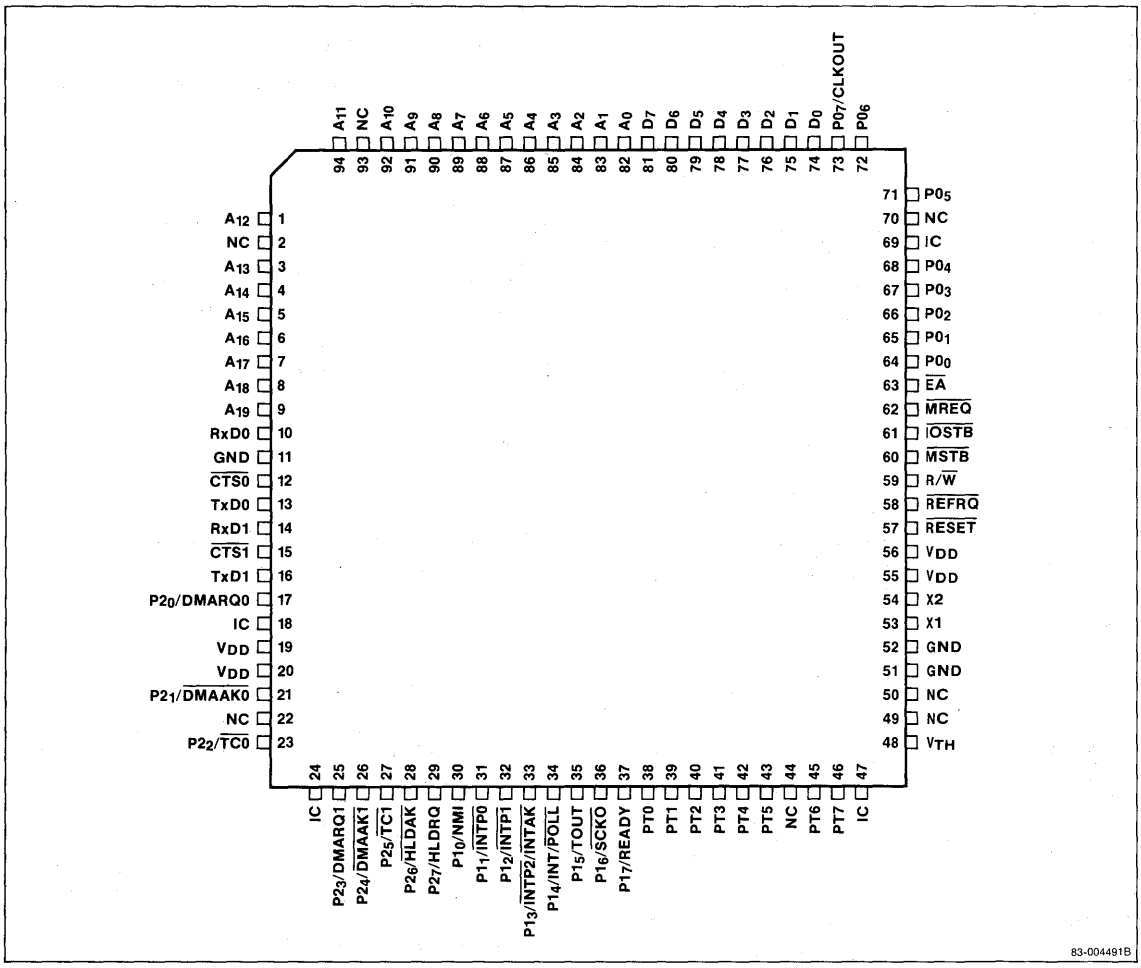


*Pin functions for normal operation of the μPD70P322 are changed as follows for programming.

Pin No.	Symbol	Function
3	V_{PP}	Write power supply input
45	$\overline{OE}$	Output enable signal input
46	$\overline{CE}$	Chip enable signal input

83-003960B

94-Pin Plastic QFP



4a

Pin Identification

Symbol	Function
A_0-A_{19}	Address bus outputs
CLKOUT	System clock output
$\overline{CTS0}$	Clear to send channel 0 input
$\overline{CTS1}$	Clear to send channel 1 input
D_0-D_7	Bidirectional data bus
$\overline{EA}$	External access
$\overline{IOSTB}$	I/O strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory strobe output
$P0_0$-$P0_7$	I/O port 0
$P1_0$/NMI	Port 1 input line/Nonmaskable interrupt input
$P1_1$-$P1_2$/ $\overline{INTP0}$-$\overline{INTP1}$	Port 1 input lines/External interrupt input lines
$P1_3$/$\overline{INTP2}$/$\overline{INTAK}$	Port 1 input line/External interrupt input line/Interrupt acknowledge output
$P1_4$/INT/$\overline{POLL}$	I/O port 1/Interrupt request input/ I/O poll input
$P1_5$/TOUT	I/O port 1/Timer out
$P1_6$/$\overline{SCK0}$	I/O port 1/Serial clock out
$P1_7$/READY	I/O port 1/Ready input
$P2_0$/DMARQ0	I/O port 2/DMA request 0
$P2_1$/$\overline{DMAAK0}$	I/O port 2/DMA acknowledge 0
$P2_2$/$\overline{TC0}$	I/O port 2/DMA terminal count 0
$P2_3$/DMARQ1	I/O port 2/DMA request 1
$P2_4$/$\overline{DMAAK1}$	I/O port 2/DMA acknowledge 1
$P2_5$/$\overline{TC1}$	I/O port 2/DMA terminal count 1
$P2_6$/$\overline{HLDAK}$	I/O port 2/Hold acknowledge output
$P2_7$/HLDRQ	I/O port 2/Hold request input
PT0-PT7	Comparator port input lines
$\overline{REFRQ}$	Refresh pulse output
$\overline{RESET}$	Reset input
RxD0	Serial receive data, channel 0 input
RxD1	Serial receive data, channel 1 input
R/$\overline{W}$	Read/Write output
TxD0	Serial transmit data, channel 0 output
TxD1	Serial transmit data, channel 1 output
X1, X2	Crystal connection terminals
V_{DD}	Positive power supply voltage
V_{TH}	Threshold voltage input
GND	Ground
IC	Internal connection

Pin Functions

A_0-A_{19} [Address Bus]

A_0-A_{19} is the 20-bit address bus used to access all external devices.

CLKOUT [System Clock]

This is the internal system clock. It can be used to synchronize external devices to the CPU.

$\overline{CTSn}$, RxDn, TxDn, $\overline{SCKO}$ [Clear to Send, Receive Data, Transmit Data, Serial Clock Out]

The two serial ports (channels 0 and 1) use these lines for transmitting and receiving data, handshaking, and serial clock output.

D_0-D_7 [Data Bus]

D_0-D_7 is the 8-bit external data bus.

DMARQn, $\overline{DMAAKn}$, $\overline{TCn}$ [DMA Request, DMA Acknowledge, Terminal Count]

These are the control signals to and from the on-chip DMA controller.

$\overline{EA}$ [External Access]

If this pin is low on reset, the μPD70322 will execute program code from external memory instead of from internal ROM.

$\overline{HLDAK}$ [Hold Acknowledge]

The $\overline{HLDAK}$ output (active low) informs external devices that the CPU has released the system bus.

HLDRQ [Hold Request]

The HLDRQ input (active high) is used by external devices to request the CPU to release the system bus to an external bus master. The following lines go into a high-impedance state with internal 4.7-kΩ pullup resistors: A_0-A_{19}, D_0-D_7, $\overline{MREQ}$, R/$\overline{W}$, $\overline{MSTB}$, $\overline{REFRQ}$, and $\overline{IOSTB}$.

INT [Interrupt Request]

INT is a maskable, active-high, vectored interrupt request input. After assertion, external hardware must provide the interrupt vector number.

$\overline{\text{INTAK}}$ [Interrupt Acknowledge]

After INT is asserted, the CPU will respond with $\overline{\text{INTAK}}$ (active low) to inform external devices that the interrupt request has been granted.

$\overline{\text{INTP0}}$-$\overline{\text{INTP2}}$ [External Interrupt]

$\overline{\text{INTP0}}$-$\overline{\text{INTP2}}$ allow external devices to generate interrupts. Each can be programmed to be rising or falling edge triggered.

$\overline{\text{IOSTB}}$ [I/O Strobe]

$\overline{\text{IOSTB}}$ is asserted during read and write operations to external I/O.

$\overline{\text{MREQ}}$ [Memory Request]

$\overline{\text{MREQ}}$ (active low) informs external memory that the current bus cycle is a memory access bus cycle.

$\overline{\text{MSTB}}$ [Memory Strobe]

$\overline{\text{MSTB}}$ (active low) is asserted during read and write operations to external memory.

NMI [Nonmaskable Interrupt]

NMI cannot be masked through software and is typically used for emergency processing. Upon execution, the interrupt starting address is obtained from interrupt vector number 2. NMI can release the standby modes and can be programmed to be either rising or falling edge triggered.

$P0_0$-$P0_7$ [Port 0]

$P0_0$-$P0_7$ are the lines of port 0, an 8-bit bidirectional parallel I/O port.

$P1_0$-$P1_7$ [Port 1]

The status of $P1_0$-$P1_3$ can be read but these lines are always control functions. $P1_4$-$P1_7$ are the remaining lines of parallel port 1, each line individually programmable as either an input, an output, or a control function.

$P2_0$-$P2_7$ [Port 2]

$P2_0$-$P2_7$ are the lines of port 2, an 8-bit bidirectional I/O port. The lines can also be used as control signals for the on-chip DMA controller.

$\overline{\text{POLL}}$ [Poll]

Upon execution of the POLL intruction, the CPU checks the status of this pin and, if low, program execution continues. If high, the CPU will check the level of the line every five clock cycles until it is low. $\overline{\text{POLL}}$ can be used to synchronize program execution to external conditions.

PT0-PT7 [Comparator Port]

PT0-PT7 are inputs to the analog comparator port.

READY [Ready]

After READY is de-asserted low, the CPU will synchronize and insert at least two wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than normal execution allows.

$\overline{\text{REFRQ}}$ [Refresh]

This active-low output pulse can refresh nonstatic RAM. It can be programmed to meet system specifications and is internally synchronized so that refresh cycles do not interfere with normal CPU operation.

$\overline{\text{RESET}}$ [Reset]

A low on $\overline{\text{RESET}}$ resets the CPU and all on-chip peripherals. $\overline{\text{RESET}}$ can also release the standby modes. After $\overline{\text{RESET}}$ returns high, program execution begins from address FFFF0H.

R/$\overline{\text{W}}$ [Read/Write]

An R/$\overline{\text{W}}$ output allows external hardware to determine if the current operation is a read or write cycle. It can also control the direction of bidirectional buffers.

TOUT [Timer Out]

TOUT is the square-wave output signal from the internal timer.

X1, X2 [Crystal Connections]

The internal clock generator requires an external crystal across these terminals as shown in figure 36. By programming the PRC register, the system clock frequency can be selected as the oscillator frequency (f_{OSC}) divided by 2, 4, or 8.

V_{DD} [Power Supply]

Two positive power supply pins (V_{DD}) reduce internal noise.

4a

V_{TH} [Threshold Voltage]

The comparator port uses this pin to determine the analog reference point. The actual threshold to each comparator line is programmable to V_{TH} x n/16, where n = 1 to 16.

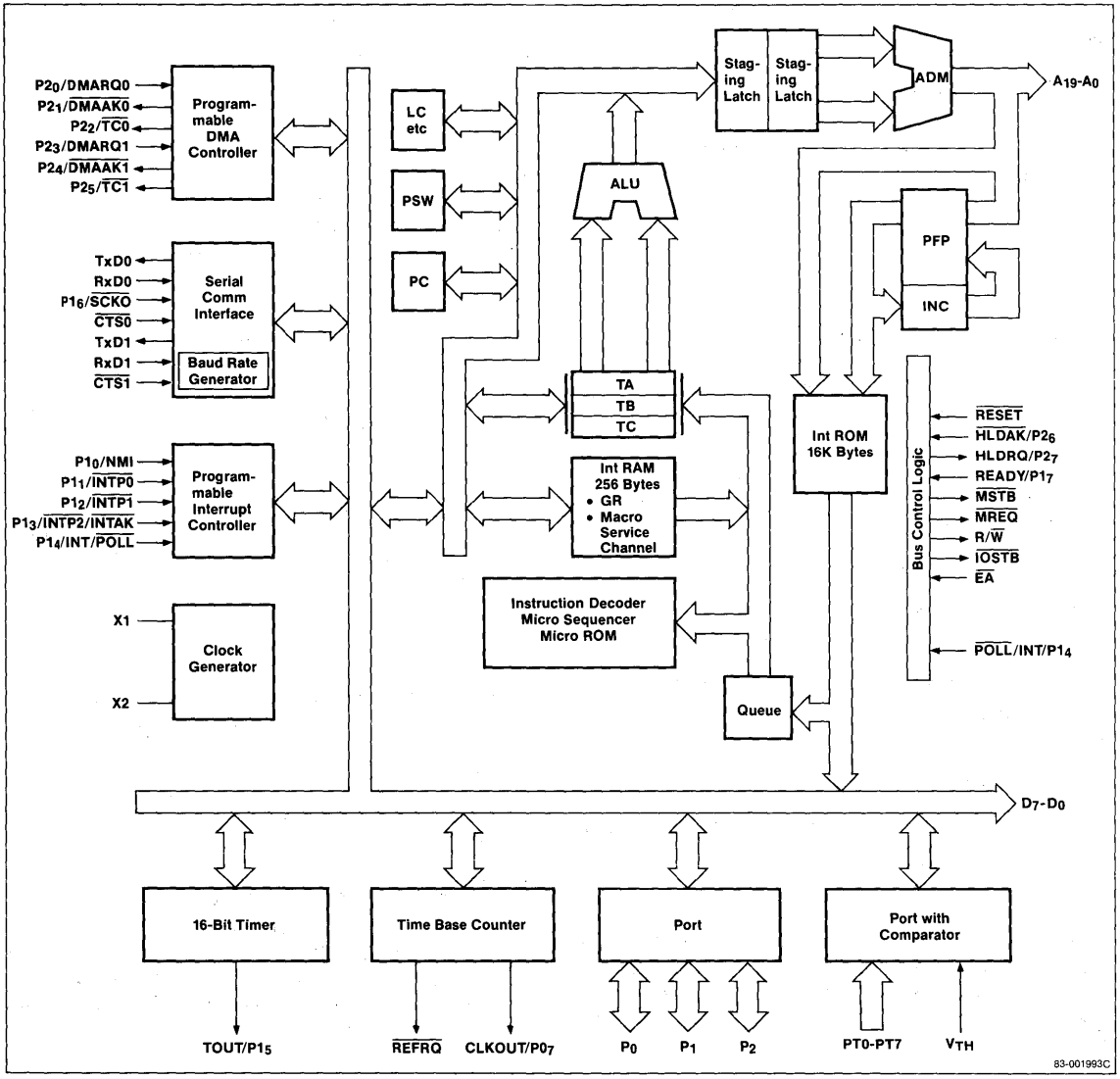
GND

Two ground connections reduce internal noise.

IC [Internal Connection]

All IC pins should be together and pulled up to V_{DD} with a 10K-20K resistor.

Block Diagram



Functional Description

Architectural Enhancements

The following features enable the μPD70320/322 to perform high-speed execution of instructions:

- Dual data bus
- 16-/32-bit temporary registers/shifters (TA, TB, TA + TB)
- 16-bit loop counter (LC)
- Program counter (PC) and prefetch pointer (PFP)
- Internal ROM pass bus (μPD70322 only)

Dual Data Bus. The μPD70320/322 has two internal 16-bit data buses: the main data bus and a subdata bus. This reduces the processing time required for addition/subtraction and logical comparison instructions by one-third over single-bus systems. The dual data bus method allows two operands to be fetched simultaneously from the general-purpose registers and transferred to the ALU.

16-/32-Bit Temporary Registers/Shifters. The 16-bit temporary registers/shifters (TA, TB) allow high-speed execution of multiplication/ division and shift/rotation instructions. By using the temporary registers/shifters, the μPD70320/322 can execute multiplication/division instructions about four times faster than with the microprogramming method.

Loop Counter [LC]. The dedicated hardware loop counter counts the number of loops for string operations and the number of shifts performed for multiple bit shift/rotation instructions. The loop counter works with internal dedicated shifters to speed the processing of multiplication/division instructions.

Program Counter and Prefetch Pointer [PC and PFP]. The hardware PC addresses the memory location of the instruction to be executed next. The hardware PFP addresses the program memory location to be accessed next. Several clocks are saved for branch, call, return, and break instructions compared with processors having only one instruction pointer.

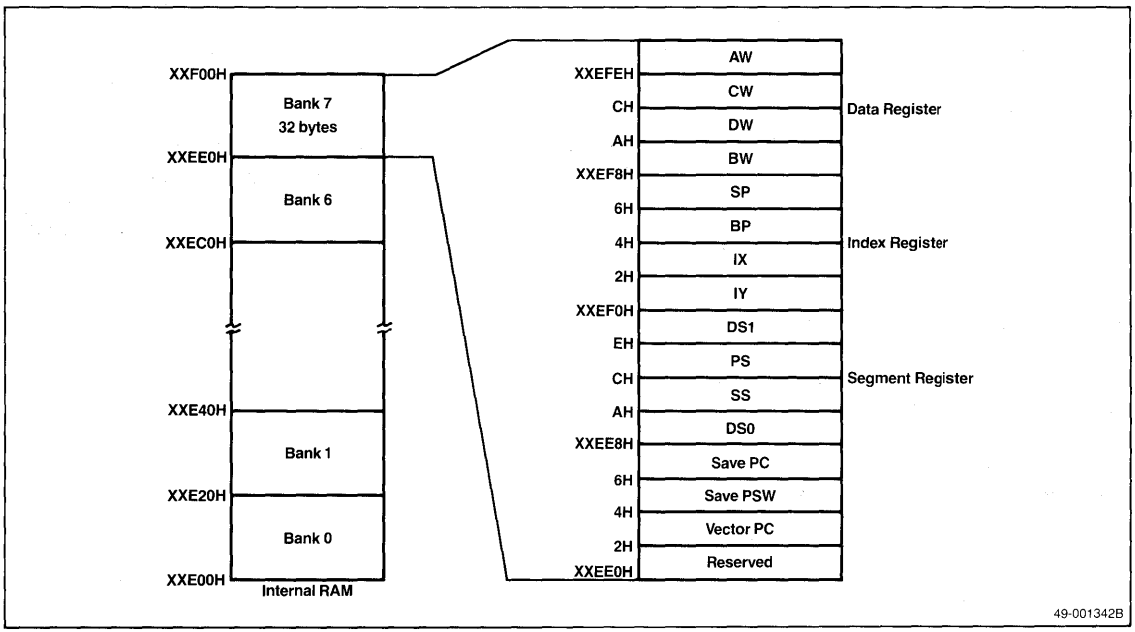
Internal ROM Pass Bus. The μPD70322 features a dedicated data bus between the internal ROM and the instruction pre-fetch queue. This allows internal ROM opcode fetches to be performed in a single clock cycle (200 ns at 5 MHz); it also makes it possible for opcode fetches to be performed while the external data bus is busy. This feature gives the V25 a 10-20% performance increase when executing from the internal ROM.

4a

Register Set

Figure 1 shows the μPD70320/322 has eight banks of registers functionally mapped into internal RAM. Each bank contains general-purpose registers, pointer and index registers, segment registers, and save areas.

Figure 1. Register Banks in Internal RAM



General-Purpose Registers [AW, BW, CW, DW]. There are four 16-bit general-purpose registers that can each serve as individual 16-bit registers or two independent 8-bit registers (AH, AL, BH, BL, CH, CL, DH, DL). The following instructions use the general-purpose registers for default:

- AW — Word multiplication/division, word I/O, data conversion
- AL — Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
- AH — Byte multiplication/division
- BW — Translation
- CW — Loop control branch, repeat prefix
- CL — Shift instructions, rotation instructions, BCD operations
- DW — Word multiplication/division, indirect addressing I/O

Pointers [SP, BP] and Index Registers [IX, IY]. These registers are used as 16-bit base pointers or index registers in based addressing, indexed addressing, and based indexed addressing. The registers are used as default registers under the following conditions:

- SP — Stack operations
- IX — Block transfer (source), BCD string operations
- IY — Block transfer (destination), BCD string operations

Segment Registers. The segment registers divide the 1M-byte address space into 64K-byte blocks. Each segment register functions as a base address to a block; the effective address is an offset from that base. Physical addresses are generated by shifting the associated segment register left four binary digits and then adding the effective address. The segment registers are:

Segment Register	Default Offset
PS (Program segment)	PC
SS (Stack segment)	SP, Effective address
DS0 (Data segment-0)	IX, Effective address
DS1 (Data segment-1)	IY, Effective address

Save Registers. Save PC and Save PSW are used as save areas during register bank context switching. The Vector PC save location contains the effective address of the interrupt service routine when register bank switching is used to service interrupts.

Program Counter [PC]. The PC is a 16-bit binary counter that contains the offset address from the program segment of the next instruction to be executed. It is incremented every time an instruction is received from the queue. It is loaded with a new location whenever a branch, call, return, break, or interrupt is executed.

Program Status Word [PSW]. The PSW contains the following status and control flags.

PSW

15							8
1	RB2	RB1	RB0	V	DIR	IE	BRK

7							0
S	Z	F1	AC	F0	P	BRKI	CY

Status Flags

- V — Overflow bit
- S — Sign
- Z — Zero
- AC — Auxiliary carry
- P — Parity
- CY — Carry

Control Flags

- DIR — Direction of string processing
- IE — Interrupt enable
- BRK — Break (after every instruction)
- RBn — Current register bank flags
- BRKI — I/O trap enable (see software interrupts)
- F0, F1 — General-purpose user flags (accessed through the Flag special function register)

The eight low-order bits of the PSW can be stored in the A4 register and restored by a MOV instruction execution. The only way to alter the RBn bits via software is to execute one of the bank switch instructions.

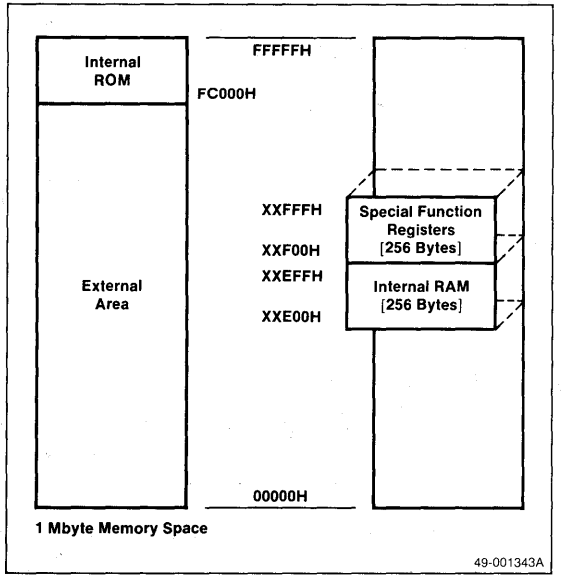
Memory Map

The μPD70320/322 has a 20-bit address bus that can directly access 1M bytes of memory. Figure 2 shows that the 16K bytes of internal ROM (μPD70322 only) are located at the top of the address space from FC000H to FFFFFH.

Figure 2 shows the internal data area (IDA) is a 256-byte internal RAM area followed consecutively by a 256-byte special function register (SFR) area. All the data and control registers for on-chip peripherals and I/O are mapped into the SFR area and accessed as RAM. For a description of these functions, see table 6. The IDA is dynamically relocatable in 4K-byte increments by changing the value in the internal data base (IDB) register. Whatever value is in this register will be assigned as the uppermost eight bits of the IDA address. The IDB register can be accessed from two different memory locations, FFFFFH and XXFFFH, where XX is the value in the IDB register.

On reset, the internal data base register is set to FFH which maps the IDA into the internal ROM space. However, since the μPD70322 has a separate bus to internal ROM, this does not present a problem. When these address spaces overlap, program code cannot be executed from the IDA and internal ROM locations cannot be accessed as data. You can select any of the eight possible register banks, which occupy the entire internal RAM space. Multiple register bank selection allows faster interrupt processing and facilitates multitasking.

Figure 2. Memory Map



In larger-scale systems where internal RAM is not required for data memory, the internal RAM can be removed completely from the address space and dedicated entirely to registers and control functions such as macro service and DMA channels. Clearing the RAMEN bit in the processor control register achieves this. When the RAMEN bit is cleared, internal RAM can only be accessed by register addressing or internal control processes. Many instructions are executed faster when the internal RAM is disabled.

Instruction Set

The μPD70320/322 instruction set is fully compatible with the V20 native mode instruction set. The V20 instruction set is a superset of the μPD8086/8088 instruction set with different execution times and mnemonics.

The μPD70320/322 does not support the V20 8080 emulation mode. All of the instructions pertaining to this have been deleted from the μPD70320/322 instruction set.

4a

Enhanced Instructions

In addition to the μPD8086/88 instructions, the μPD70320/322 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes eight general registers onto stack
POP R	Pops eight general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Unique Instructions

The μPD70320/322 has the following unique instructions.

Instruction	Function
INS	Inserts bit field
EXT	Extracts bit field
ADD4S	Performs packed BCD string addition
SUB4S	Performs packed BCD string subtraction
CMP4S	Performs packed BCD string comparison
ROL4	Rotates BCD digit left
ROR4	Rotates BCD digit right
TEST1	Tests bit
SET1	Sets bit
CLR1	Clears bit
NOT1	Complements bit
BTCLR	Tests bit; if true, clear and branch
REPC	Repeat while carry set
REPNC	Repeat while carry cleared

Variable Length Bit Field Operation Instructions

Bit fields are a variable length data structure that can range in length from 1 to 16 bits. The μPD70320/322 supports two separate operations on bit fields: insertion (INS) and extraction (EXT). There are no restrictions on the position of the bit field in memory. Separate segment, byte offset, and bit offset registers are used for insertion and extraction. Following the execution of these instructions, both the byte offset and bit offset are left pointing to the start of the next bit field, ready for the next operation. Bit field operation instructions are powerful and flexible and are therefore highly effective for graphics, high-level languages, and packing/unpacking applications.

Bit field insertion copies the bit field of specified length from the AW register to the bit field addressed by DS1:IY:reg8 (8-bit general-purpose register). The bit field length can be located in any byte register or supplied as immediate data. Following execution, both the IY and reg8 are updated to point to the start of the next bit field.

Bit field extraction copies the bit field of specified length from the bit field addressed by DS0:IX:reg8 to the AW register. If the length of the bit field is less than 16 bits, the bit field is right justified with a zero fill. The bit field length can be located in any byte register or supplied as immediate data. Following execution, both IX and reg8 are updated to point to the start of the next bit field.

Figures 3 and 4 show bit field insertion and bit field extraction.

Packed BCD Instructions

Packed BCD instructions process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte format operands (ROR4, ROL4). Packed BCD strings may be 1 to 254 digits in length. The two BCD rotation instructions perform rotation of a single BCD digit in the lower half of the AL register through the register or the memory operand.

Bit Manipulation Instructions

The μPD70320/322 has five unique bit manipulation instructions. The ability to test, set, clear, or complement a single bit in a register or memory operand increases code readability as well as performance over the logical operations traditionally used to manipulate bit data. This feature further enhances control over on-chip peripherals.

Figure 3. Bit Field Insertion

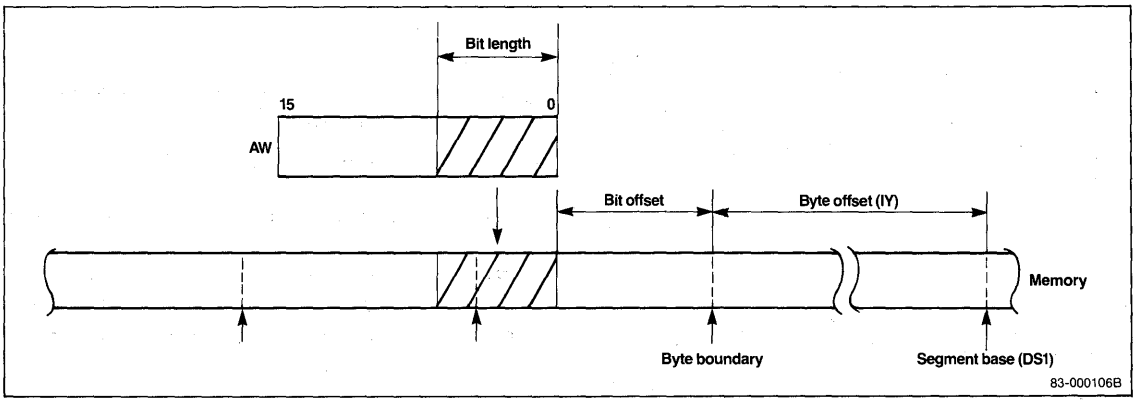
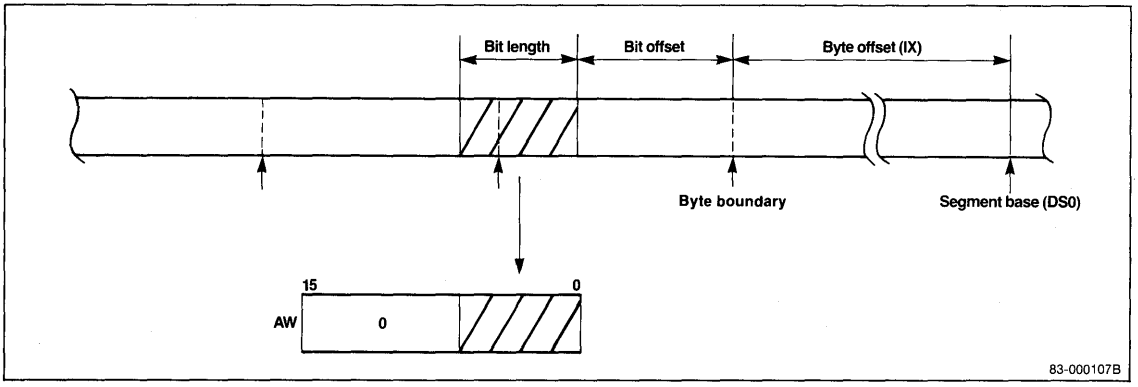


Figure 4. Bit Field Extraction



Additional Instructions

Besides the V20 instruction set, the μPD70320/322 has the four additional instructions described in table 1.

Table 1. Additional Instructions

Instruction	Function
BTCLR var,imm8, short label	Bit test and if true, clear and branch; otherwise, no operation
STOP (no operand)	Power down instruction, stops oscillator
RETRBI (no operand)	Return from register bank context switch interrupt
FINT (no operand)	Finished interrupt. After completion of a hardware interrupt request, this instruction must be used to reset the current priority bit in the in-service priority register (ISPR).*

*Do not use with NMI or INTR interrupt service routines.

Repeat Prefixes

Two new repeat prefixes (REPC, REPNC) allow conditional block transfer instructions to use the state of the CY flag as the termination condition. This allows inequalities to be used when working on ordered data, thus increasing performance when searching and sorting algorithms.

Bank Switch Instructions

The V25 has four new instructions that allow the effective use of the register banks for software interrupts and multitasking. These instructions are shown in table 2. Also, see figures 8 and 10.

Interrupt Structure

The μPD70320/322 can service interrupts generated both by hardware and by software. Software interrupts are serviced through vectored interrupt processing. See table 3 for the various types of software interrupts.

Table 2. Bank Switch Instructions

Instruction	Function
BRKCS reg 16	Performs a high-speed software interrupt with context switch to the register bank indicated by the lower 3-bits of reg 16. This operation is identical to the interrupt operation shown in figure 9.
TSKSW reg 16	Performs a high-speed task switch to the register bank indicated by the lower 3-bits of reg 16. The PC and PSW are saved in the old banks. PC and PSW save registers and the new PC and PSW values are retrieved from the new register bank's save areas. See figure 10.
MOVSPA	Transfers both the SS and SP of the old register bank to the new register bank after the bank has been switched by an interrupt or BRKCS instruction.
MOVSPB	Transfers the SS and the SP of the current register bank before the switch to the SS and SP of the new register bank indicated by the lower 3-bits of reg 16.

Table 3. Software Interrupts

Interrupt	Description
Divide error	The CPU will trap if a divide error occurs as the result of a DIV or DIVU instruction.
Single step	The interrupt is generated after every instruction if the BRK bit in the PSW is set.
Overflow	By using the BRKV instruction, an interrupt can be generated as the result of an overflow.
Interrupt instructions	The BRK 3 and BRK imm8 instructions can generate interrupts.
Array bounds	The CHKIND instruction will generate an interrupt if specified array bounds have been exceeded.
Escape trap	The CPU will trap on an FP01,2 instruction to allow software to emulate the floating point processor.
I/O trap	If the I/O trap bit in the PSW is cleared, a trap will be generated on every IN or OUT instruction. Software can then provide an updated peripheral address. This feature allows software interchangeability between different systems.

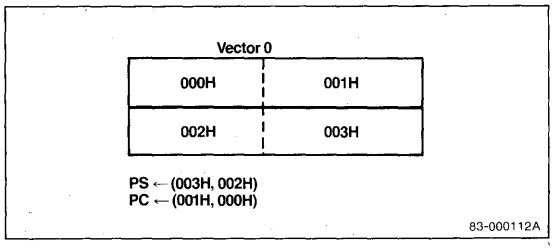
When executing software written for another system, it is better to implement I/O with on-chip peripherals to reduce external hardware requirements. However, since μPD70320/322 internal peripherals are memory mapped, software conversion could be difficult. The I/O trap feature allows easy conversion from external peripherals to on-chip peripherals.

Interrupt Vectors

The starting address of the interrupt processing routines may be obtained from table 3. The table begins at physical address 00H, which is outside the internal ROM space. Therefore, external memory is required to service these routines. By servicing interrupts via the macro service function or context switching, this requirement can be eliminated.

Each interrupt vector is four bytes wide. To service a vectored interrupt, the lower addressed word is transferred to the PC and the upper word to the PS. See figure 5.

Figure 5. Interrupt Vector 0



Execution of a vectored interrupt occurs as follows:

(SP-1, SP-2) ← PSW
(SP-3, SP-4) ← PS
(SP-5, SP-6) ← PC
SP ← SP-6
IE ← 0, BRK ← 0
PS ← vector high bytes
PC ← vector low bytes

Hardware Interrupt Configuration

The V25 features a high-performance on-chip controller capable of controlling multiple processing for interrupts from up to 17 different sources (5 external, 12 internal). The interrupt configuration includes system interrupts that are functionally compatible with those of the V20/V30 and unique high-performance microcontroller interrupts.

Table 4. Interrupt Vectors

Address	Vector No.	Assigned Use
00	0	Divide error
04	1	Break flag
08	2	NMI
0C	3	BRK3 instruction
10	4	BRKV instruction
14	5	CHKIND instruction
18	6	General purpose
1C	7	FPO instructions
20-2C	8-11	General purpose
30	12	INTSER0 (Interrupt serial error, channel 0)
34	13	INTSR0 (Interrupt serial receive, channel 0)
38	14	INTST0 (Interrupt serial transmit, channel 0)
3C	15	General purpose
40	16	INTSER1 (Interrupt serial error, channel 1)
44	17	INTSR1 (Interrupt serial receive, channel 1)
48	18	INTST1 (Interrupt serial transmit, channel 1)
4C	19	I/O trap
50	20	INTD0 (Interrupt from DMA, channel 0)
54	21	INTD1 (Interrupt from DMA, channel 1)
58	22	General purpose
5C	23	General purpose
60	24	INTP0 (Interrupt from peripheral 0)
64	25	INTP1 (Interrupt from peripheral 1)
68	26	INTP2 (Interrupt from peripheral 2)
6C	27	General purpose
70	28	INTTU0 (Interrupt from timer unit 0)
74	29	INTTU1 (Interrupt from timer unit 1)
78	30	INTTU2 (Interrupt from timer unit 2)
7C	31	INTTB (Interrupt from time base counter)
080-3FF	32-255	General purpose

Interrupt Sources

The 17 interrupt sources (table 5) are divided into groups for management by the interrupt controller. Using software, each of the groups can be assigned a priority from 0 (highest) to 7 (lowest). The priority of individual interrupts within a group is fixed in hardware. If interrupts from different groups occur simultaneously and the groups have the same assigned priority level, the priority followed will be as shown in the Default Priority column of table 5.

The ISPR is an 8-bit special function register; bits PR_0-PR_7 correspond to the eight possible interrupt request priorities. The ISPR keeps track of the priority of the interrupt currently being serviced by setting the appropriate bit. The address of the ISPR is XXFFCH. The ISPR format is shown below.

PR_7	PR_6	PR_5	PR_4	PR_3	PR_2	PR_1	PR_0

NMI and INTR are system-type external vectored interrupts. NMI is not maskable via software. INTR is maskable (IE bit in PSW) and requires that an external device provide the interrupt vector number. It allows expansion by the addition of an external interrupt controller (μPD71059).

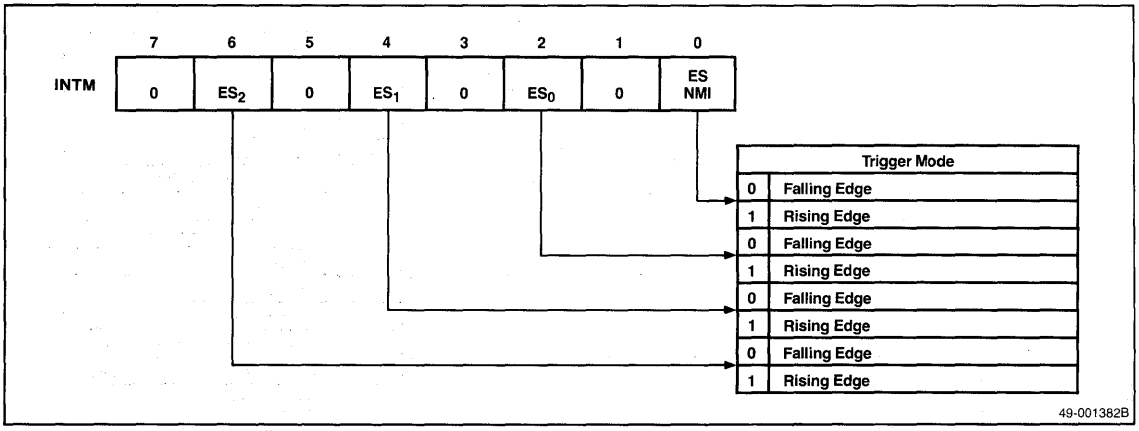
NMI, INTP0, and INTP1 are edge-sensitive interrupt inputs. By selecting the appropriate bits in the interrupt mode register, these inputs can be programmed to be either rising or falling edge triggered. ES_0-ES_2 correspond to INTP0-INTP2, respectively. See figure 6.

Table 5. Interrupt Sources

Group	Interrupt Source (Priority Within Group) 1	2	3	Default Priority
Non-maskable interrupt	NMI	—	—	0
Timer unit	INTTU0	INTTU1	INTTU2	1
DMA controller	INTD0	INTD1	—	2
External peripheral interrupt	INTP0	INTP1	INTP2	3
Serial channel 0	INTSER0	INTSR0	INTST0	4
Serial channel 1	INTSER1	INTSR1	INTST1	5
Time base counter	INTTB	—	—	6
Interrupt request	INTR	—	—	7

4a

Figure 6. Interrupt Mode Register (INTM)



Interrupt Processing Modes

Interrupts, with the exception of NMI, INT, and INTTB, have high-performance capability and can be processed in any of three modes: standard vector interrupt, register bank context switching, or macro service function. The processing mode for a given interrupt can be chosen by enabling the appropriate bits in the corresponding interrupt request control register. As shown in table 6, each individual interrupt, with the exception of INTR and NMI, has its own associated IRC register. The format for all IRC registers is shown in figure 7.

All interrupt processing routines other than those for NMI and INT must end with the execution of an FINT instruction. Otherwise, subsequently, only interrupts of a higher priority will be accepted.

In the vectored interrupt mode, the CPU traps to the vector location shown in table 4.

Register Bank Switching

Register bank context switching allows interrupts to be processed rapidly by switching register banks. After an interrupt, the new register bank selected is that which has the same register bank number (0-7) as the priority of the interrupt to be serviced. The PC and PSW are automatically stored in the save areas of the new register bank and the address of the interrupt routine is loaded from the vector PC storage location in the new register bank. As in the vectored mode, the IE and BRK bits in the PSW are cleared to zero. After interrupt processing, execution of the RETRBI (return from register bank interrupt) returns control to the former register bank and restores the former PC and PSW. Figures 8 and 9 show register bank context switching and register bank return.

Figure 7. Interrupt Request Control Registers (IRC)

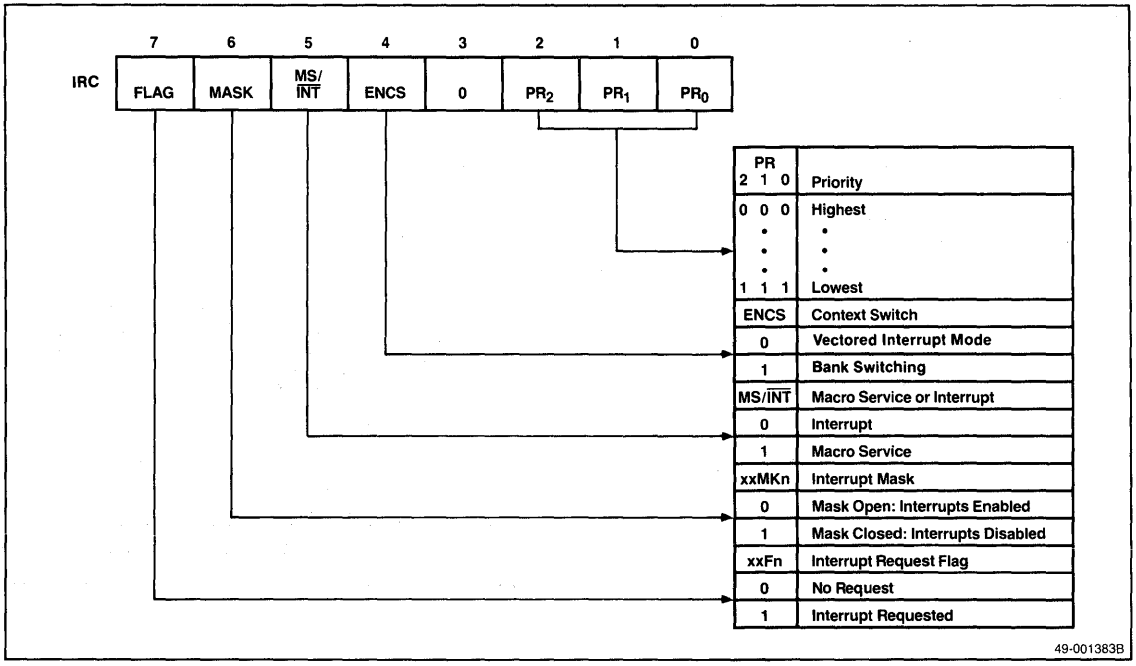


Figure 8. Register Bank Context Switching

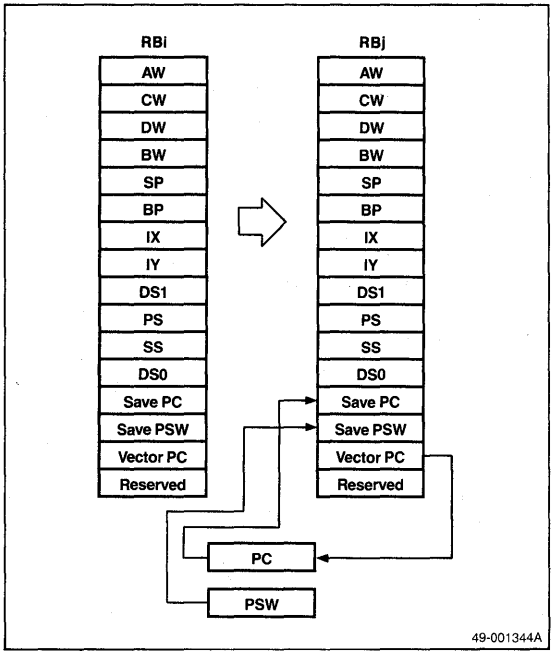
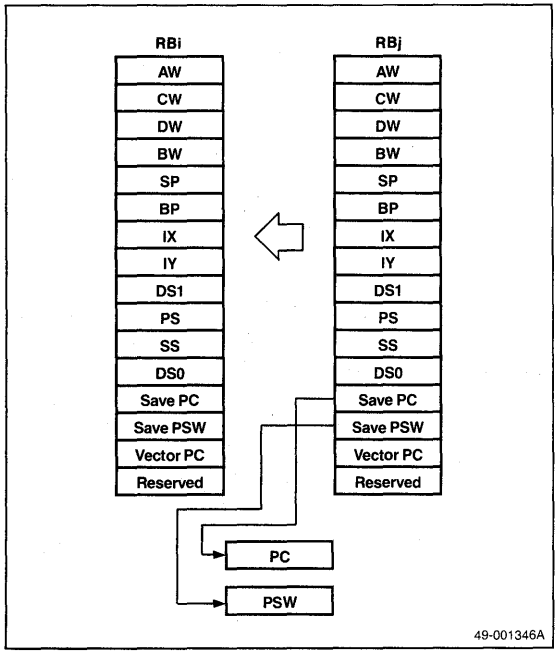


Figure 9. Register Bank Return



Macro Service Function

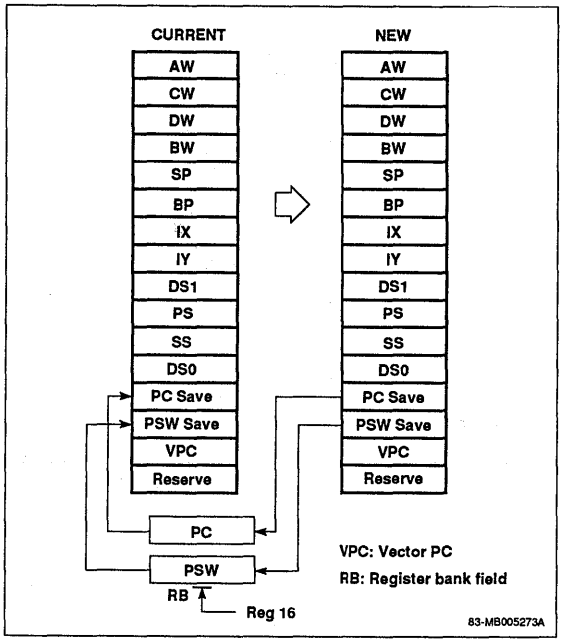
The macro service function (MSF) is a special microprogram that acts as an internal DMA controller between on-chip peripherals (special function registers, SFR) and memory. The MSF greatly reduces the software overhead and CPU time that other processors would require for register save processing, register returns, and other handling associated with interrupt processing.

If the MSF is selected for a particular interrupt, each time the request is received, a byte or word of data will be transferred between the SFR and memory without interrupting the CPU. Each time a request occurs, the macro service counter is decremented. When the counter reaches zero, an interrupt to the CPU is generated. The MSF also has a character search option. When selected, every byte transferred will be compared to an 8-bit search character and an interrupt will be generated if a match occurs or if the macro service counter counts out.

Like the NMI, INT and INTTB, the two DMA controller interrupts (INTD0, INTD1) do not have MSF capability.

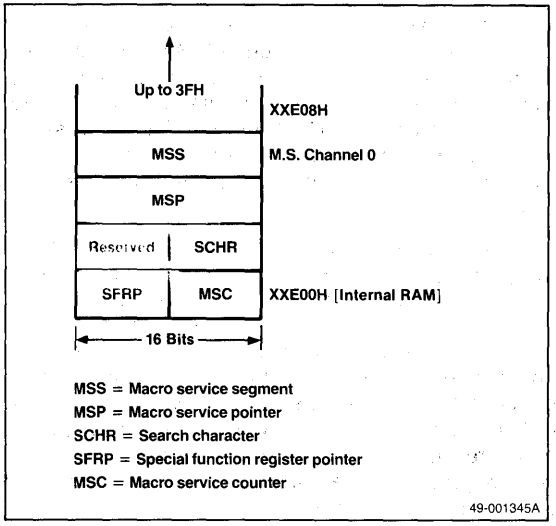
There are eight 8-byte macro service channels mapped into internal RAM from XXE00H to XXE3FH. Each macro service channel contains all of the necessary information to execute the macro service process. Figure 11 shows the components of each channel.

Figure 10. Task Switching



4a

Figure 11. Macro Service Channels



Setting the macro service mode for a given interrupt requires programming the corresponding macro service control register. Each individual interrupt, excluding INTR, NMI and TBC, has its own associated MSC register. See table 6. Format for all MSC registers is shown in figure 12.

On-Chip Peripherals

Timer Unit

The μPD70320/322 (figure 13) has two programmable 16-bit interval timers (TM0, TM1) on-chip, each with variable input clock frequencies. Each of the two 16-bit timer registers has an associated 16-bit modulus register (MD0, MD1). Timer 0 operates in the interval timer mode or one-shot mode; timer 1 has only the interval timer mode.

Interval Timer Mode. In this mode, TM0/TM1 are decremented by the selected input clock and, after counting out, the registers are automatically reloaded from the modulus registers and counting continues. Each time TM1 counts out, interrupts are generated through TF1 and TF2 (Timer Flags 1, 2). When TM0 counts out, an interrupt is generated through TF0. The timer-out signal can be used as a square-wave output whose half-cycle is equal to the count time. There are two selectable input clocks (SCLK: system clock = $f_{OSC}/2$; f_{OSC} = 10 MHz).

Clock	Timer Resolution	Full Count
SCLK/6	1.2 μs	78.643 ms
SCLK/128	25.6 μs	1.678 s

Figure 12. Macro Service Control Registers (MSC)

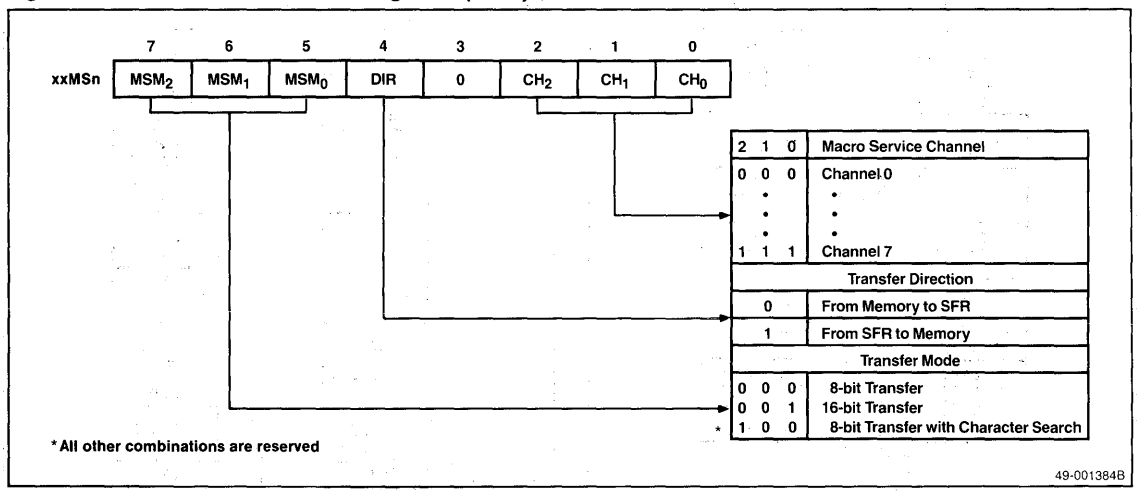
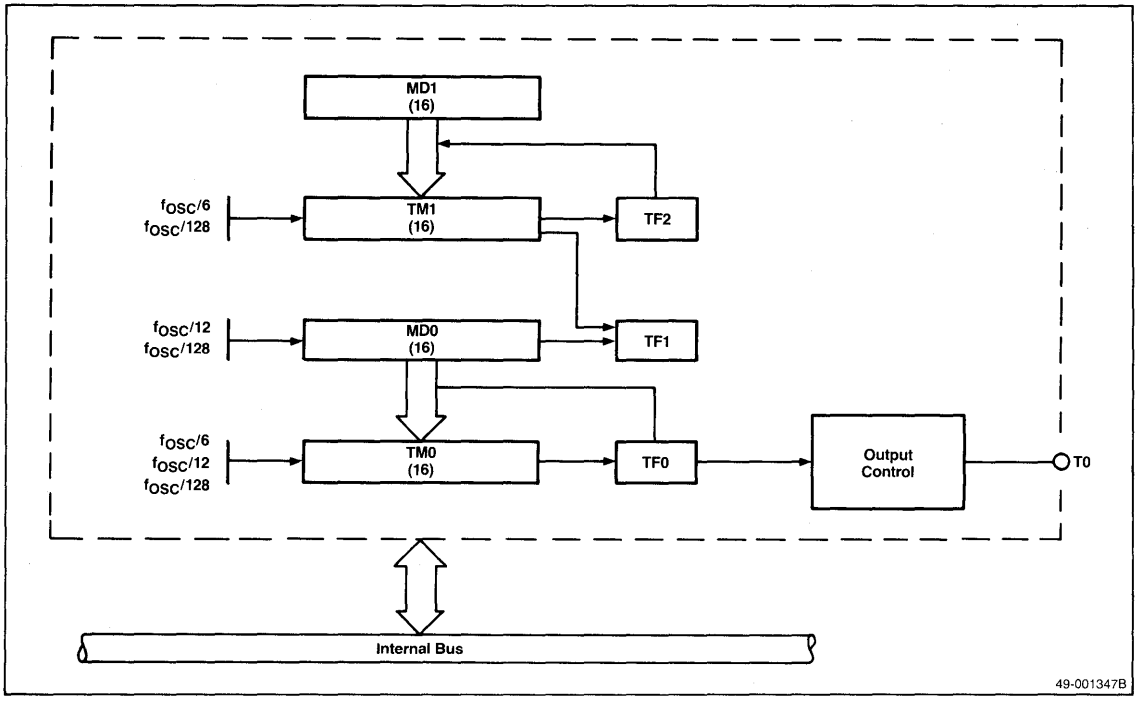


Figure 13. Timer Unit Block Diagram



4a

One-Shot Mode. In the one-shot mode, TM0 and MD0 operate as independent one-shot timers. Starting with a preset value, each is decremented to zero. At zero, counting ceases and an interrupt is generated by TF0 (from TM0) or TF1 (from MD0). One-shot mode allows two selectable input clocks (f_{OSC} = 10 MHz).

Clock	Timer Resolution	Full Count
SCLK/12	2.4 μs	157.283 ms
SCLK/128	25.6 μs	1.678 s

Setting the desired timer mode requires programming the timer control register. See figures 14 and 15 for format.

Figure 14. Timer Control Register 0

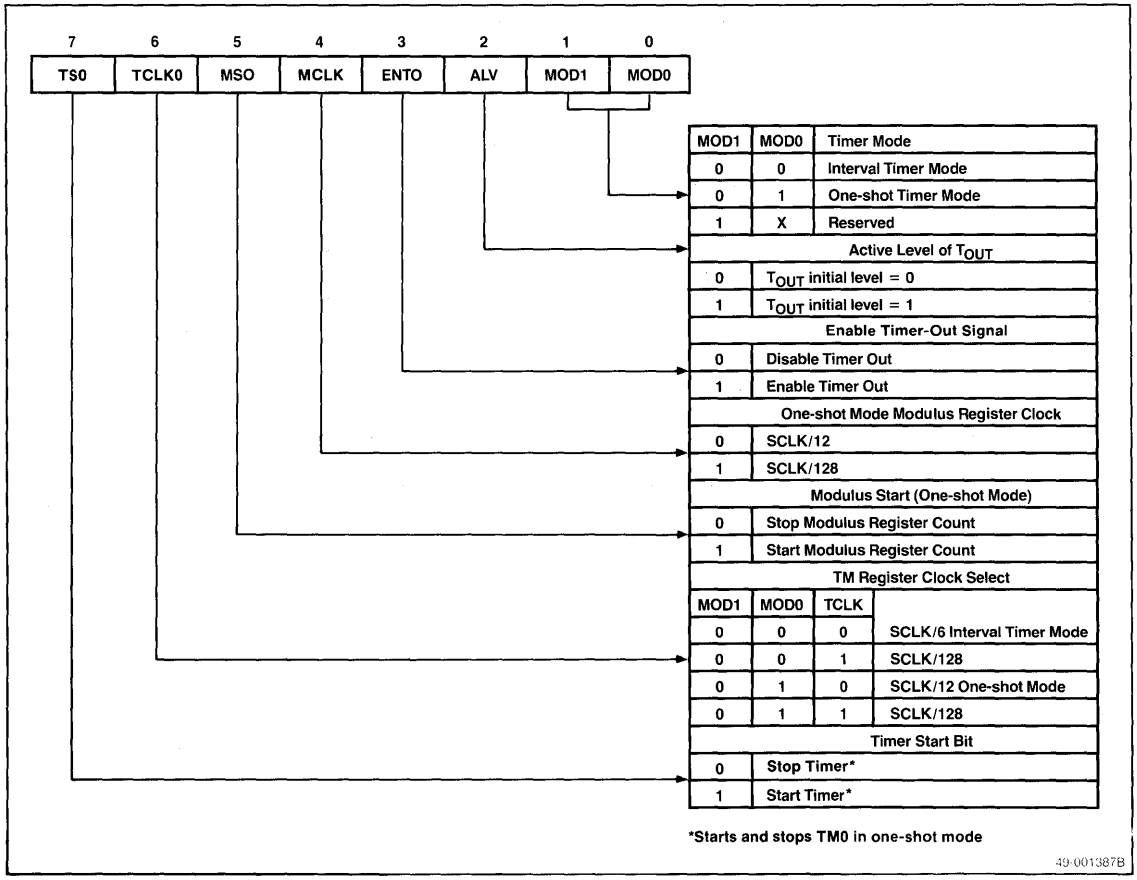
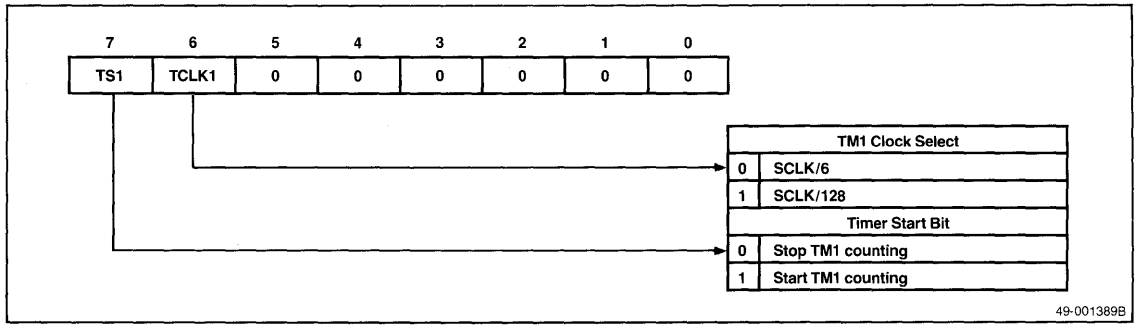


Figure 15. Timer Control Register 1



Time Base Counter/Processor Control Register

The 20-bit free-running time base counter controls internal timing sequences and is available to the user as the source of periodic interrupts at lengthy intervals. One of four interrupt periods can be selected by programming the TB0 and TB1 bits in the processor control register (PRC). The TBC interrupt is unlike the others in that it is fixed as a level 7 vectored interrupt. Macro service and register bank switching cannot be used to service this interrupt. See figures 16 and 17.

The RAMEN bit in the PRC register allows the internal RAM to be removed from the memory address space to implement faster instruction execution.

The TBC (figure 18) uses the system clock as the input frequency. The system clock can be changed by programming the PCK0 and PCK1 bits in the processor control register (PRC). Reset initializes the system clock to $f_{OSC}/8$ (f_{OSC} = external oscillator frequency).

Figure 18. Time Base Counter (TBC) Block Diagram

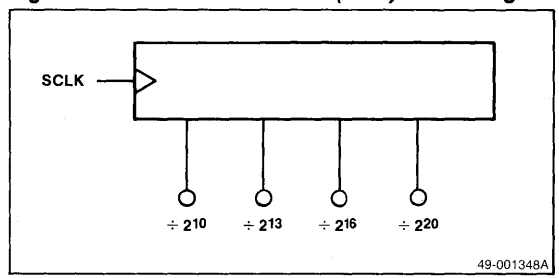


Figure 16. Time Base Interrupt Request Control Register

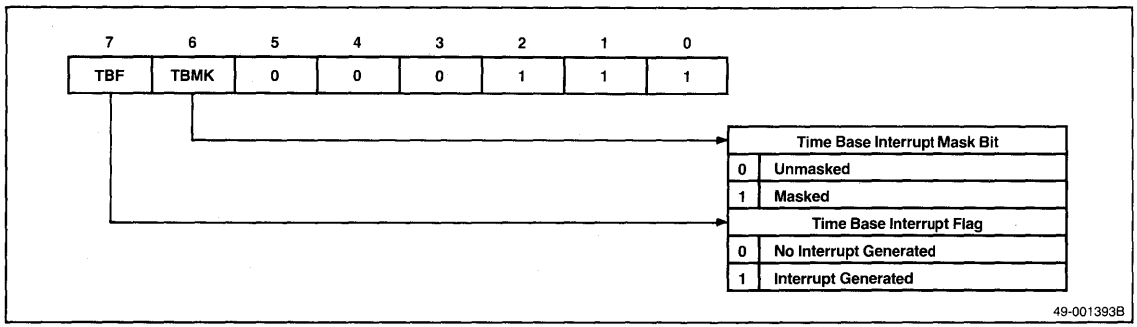
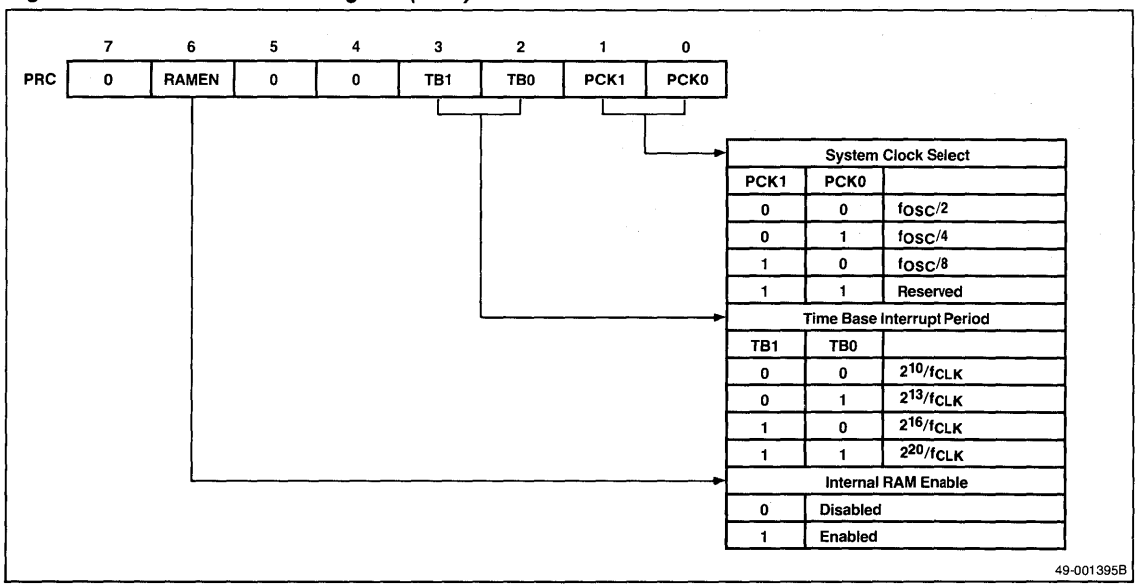


Figure 17. Processor Control Register (PRC)



Refresh Controller

The μPD70320/322 has an on-chip refresh controller for dynamic and pseudostatic RAM mass storage memories. The refresh controller generates refresh addresses and refresh pulses. It inserts refresh cycles between the normal CPU bus cycles according to refresh specifications.

The refresh controller outputs a 9-bit refresh address on address bits A_0-A_8 during the refresh bus cycle. Address bits A_9-A_{19} are all 1's. The 9-bit refresh address is automatically incremented at every refresh timing for 512 row addresses. The 8-bit refresh mode (RFM) register (figure 19) specifies the refresh operation and allows refresh during both CPU HALT and HOLD modes. Refresh cycles are automatically timed to $\overline{\text{REFRQ}}$ following read/write cycles to minimize the effect on system thoughput.

The following shows the $\overline{\text{REFRQ}}$ pin level in relation to bits 4 (RFEN) and 7 (RFLV) of the refresh mode register.

RFEN	RFLV	$\overline{\text{REFRQ}}$ Level
0	0	0
0	1	1
1	0	0
1	1	Refresh pulse output

Serial Interface

The μPD70320/322 has two full-duplex UARTs, channel 0 and channel 1. Each serial port channel has a transmit line (TxDn), a receive line (RxDn), and a clear to send (CTSn) input line for handshaking. Communication is synchronized by a start bit, and you can program the ports for even, odd, or no parity, character lengths of 7 or 8 bits, and 1 or 2 stop bits.

The μPD70320/322 has dedicated baud rate generators for each serial channel. This eliminates the need to obligate the on-chip timers. The baud rate generator allows a wide range of data transfer rates (up to 1.25 Mb/s). This includes all of the standard baud rates without being restricted by the value of the particular external crystal.

Each baud rate generator has an 8-bit baud rate generator (BRGn) data register, which functions as a prescaler to a programmable input clock selected by the serial communication control (SCCn) register. Together these must be set to generate a frequency equivalent to the desired baud rate.

The baud rate generator can be set to obtain the desired transmission rate according to the following formula:

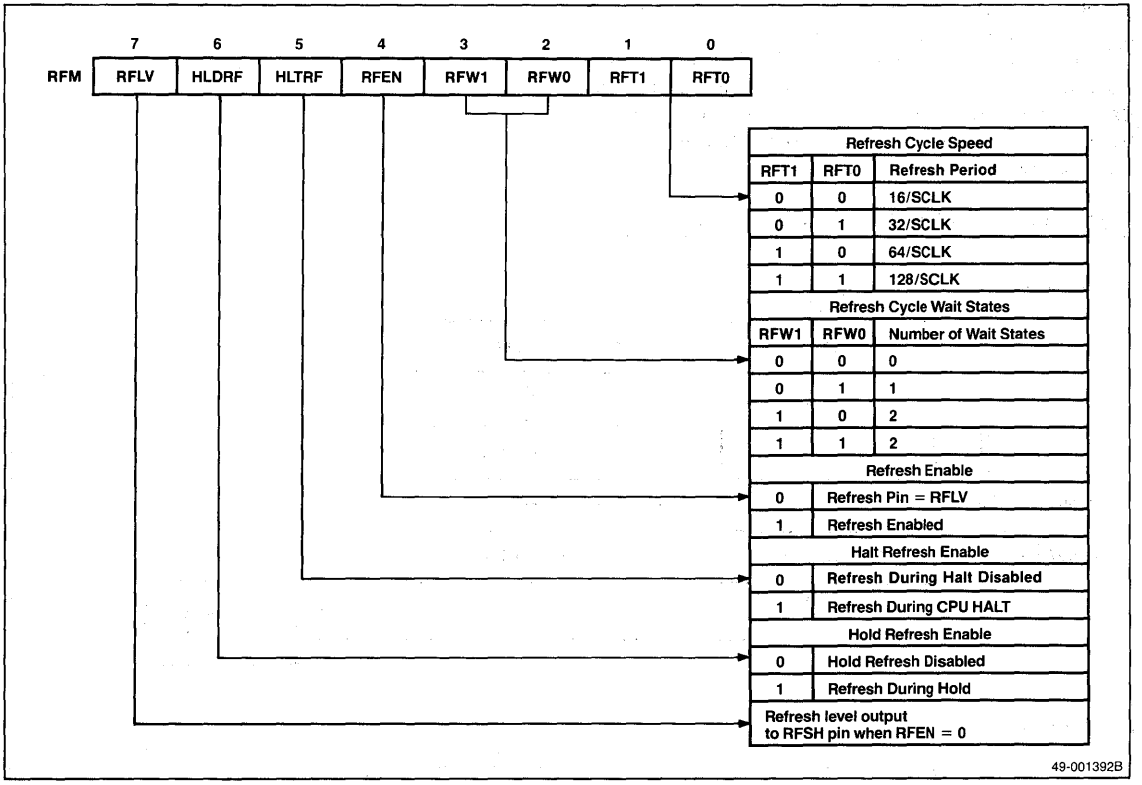
$$B \times G = \frac{SCLK \times 10^6}{2^{n+1}}$$

where B = baud rate
G = baud rate generator register (BRGn) value
n = input clock specifications (n between 0 and 8) This is the value that is loaded into the SCCn register (see figure 23).
SCLK = system clock frequency (MHz)

Based on the above expression, the following table shows the baud rate generator values used to obtain standard transmission rates when SCLK = 5 MHz.

Baud Rate	n	BRGn Value	Error (%)
110	7	178	0.25
150	7	130	0.16
300	6	130	0.16
600	5	130	0.16
1200	4	130	0.16
2400	3	130	0.16
4800	2	130	0.16
9600	1	130	0.16
19,200	0	130	0.16
38,400	0	65	0.16
1.25M	0	2	0

Figure 19. Refresh Mode Register (RFM)



4a

In addition to the asynchronous mode, channel 0 has a synchronous I/O interface mode. In this mode, each bit of data tranferred is synchronized to a serial clock ($\overline{SCKO}$). This is the same as the NEC μCOM75 and μCOM87 series, and allows easy interfacing to these devices. Figure 20 is the serial interface block diagram; figures 21, 22, and 23 show the three serial communication registers.

Figure 20. Serial Interface Block Diagram

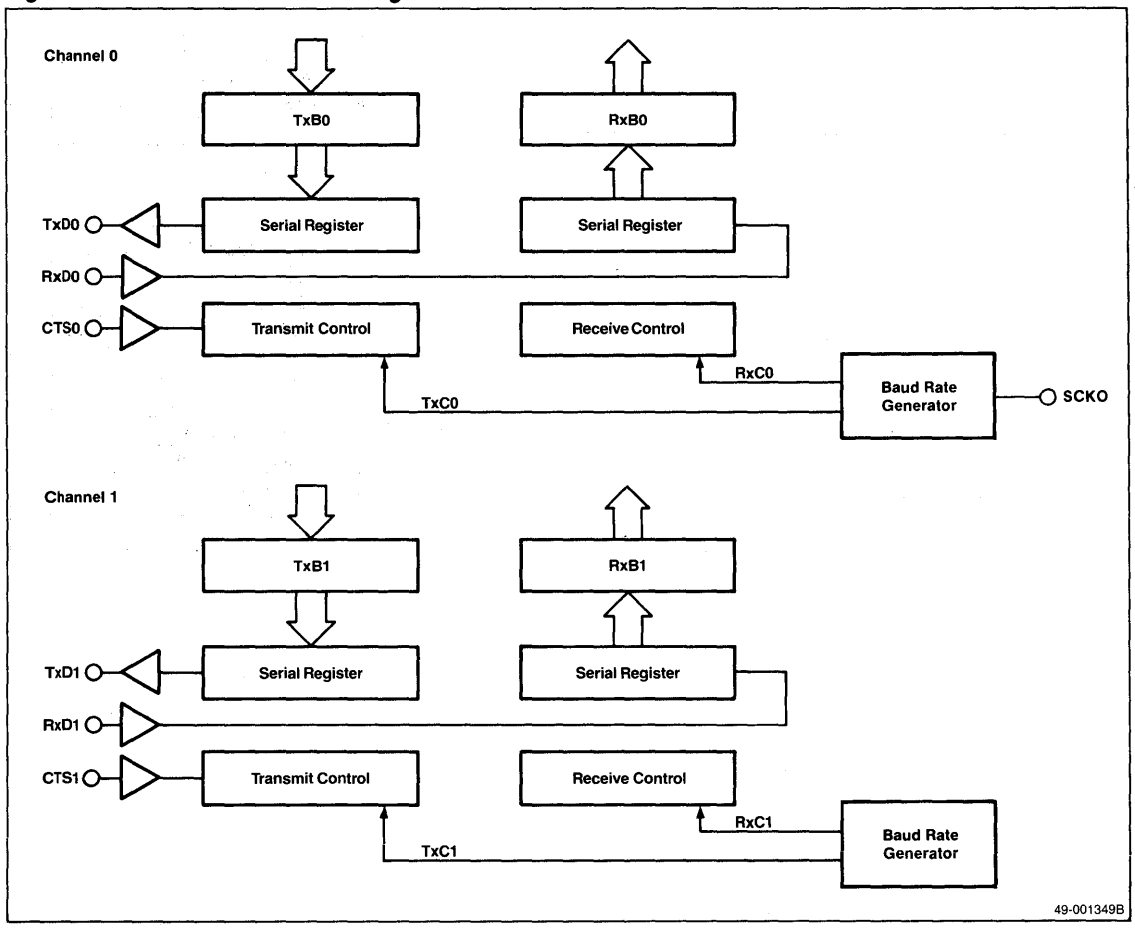


Figure 21. Serial Communication Mode Register (SCM)

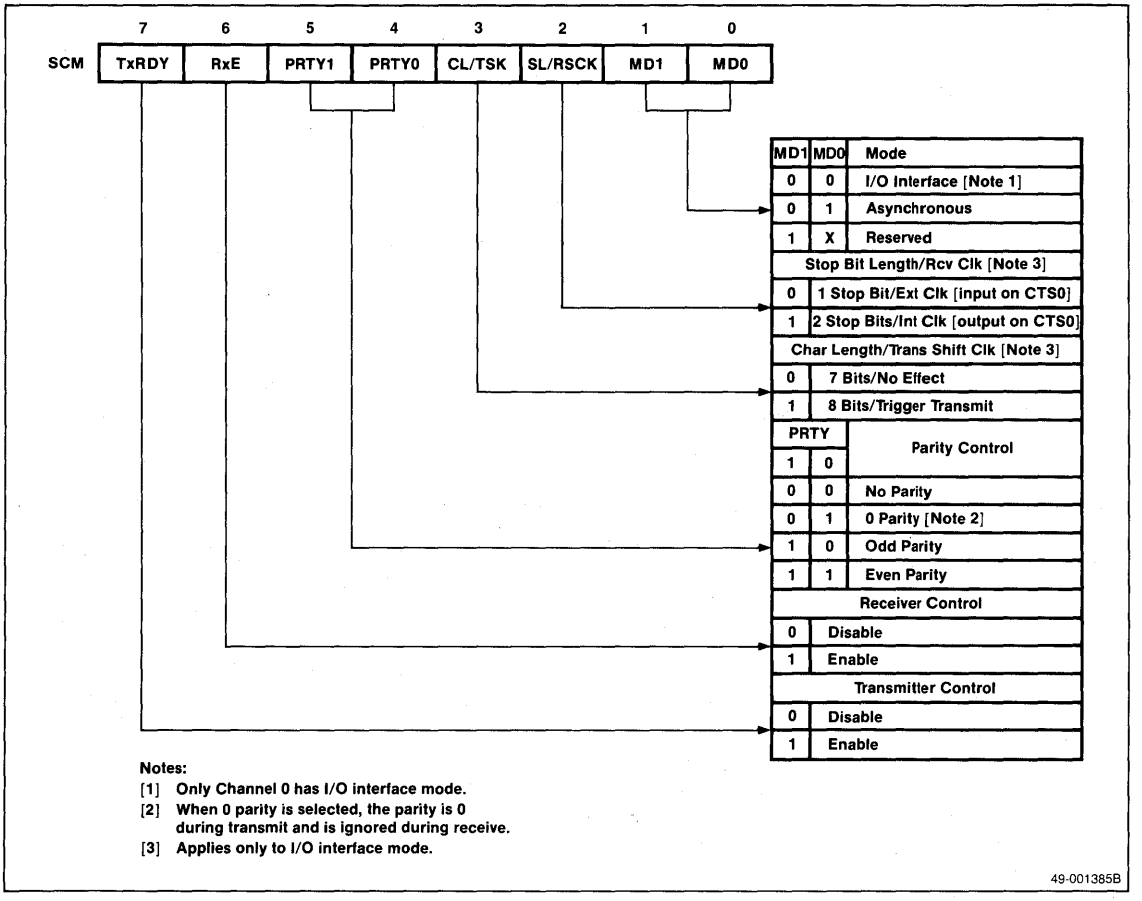


Figure 22. Serial Communication Error Registers (SCE)

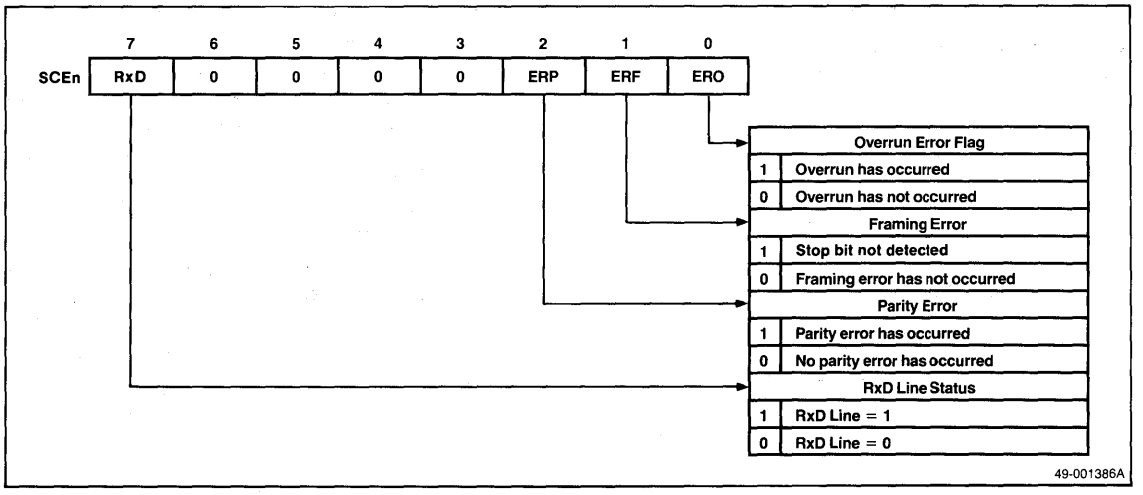


Figure 23. Serial Communication Control Register (SCC)

	7	6	5	4	3	2	1	0
SCC	0	0	0	0	PRS_3	PRS_2	PRS_1	PRS_0

PRS 3 2 1 0	Input clock for baud rate generator
0 0 0 0	SCLK/2
0 0 0 1	SCLK/4
0 0 1 0	SCLK/8
0 0 1 1	SCLK/16
0 1 0 0	SCLK/32
0 1 0 1	SCLK/64
0 1 1 0	SCLK/128
0 1 1 1	SCLK/256
1 0 0 0	SCLK/512*

*All other combinations after 1000 are illegal

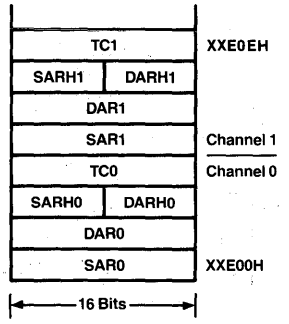
49-001388B

DMA Controller

The μPD70320/322 has a two-channel, on-chip DMA controller. This allows rapid data transfer between memory and auxiliary storage devices. The DMA controller supports four modes of operation, two for memory-to-memory transfers and two for transfers between I/O and memory. See figures 24, 25, and 26 for a graphic representation of the DMA registers.

Memory-to-Memory Transfers. In the single-step mode, when one DMA request is made, execution of one instruction and one DMA transfer are repeated alternately until the prescribed number of DMA transfers has occurred. Interrupts can be accepted while in this mode. In burst mode, one DMA request causes DMA transfer cycles to continue until the DMA terminal counter decrements to zero. Software can also initiate memory-to-memory transfers.

Figure 24. DMA Channels



49-001350A

Transfers Between I/O and Memory. In single-transfer mode, one DMA transfer occurs after each rising edge of DMARQ. After the transfer, the bus is returned to the CPU. In demand release mode, the rising edge of DMARQ enables DMA cycles, which continue as long as DMARQ is high.

In all modes, the $\overline{TC}$ (terminal count) output pin will pulse low and a DMA completion interrupt request will be generated after the predetermined number of DMA cycles has been completed.

The bottom of internal RAM contains all the necessary address information for the designated DMA channels. The DMA channel mnemonics are as follows:

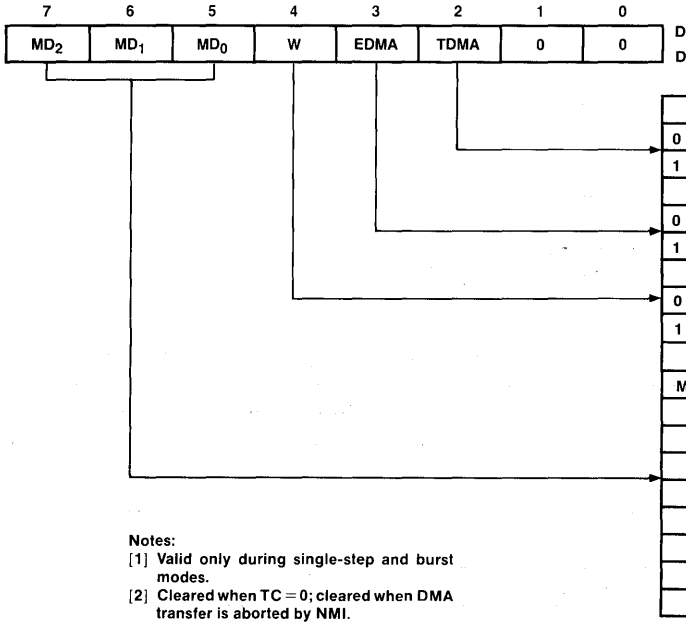
TC	Terminal counter
SAR	Source address register
SARH	Source address register high
DAR	Destination address register
DARH	Destination address register high

The DMA controller generates physical source addresses by offsetting SARH 12 bits to the left and then adding the SAR. The same procedure is also used to generate physical destination addresses. You can program the controller to increment or decrement source and/or destination addresses independently during DMA transfers.

When the EDMA bit is set, the internal DMARQ flag is cleared. Therefore, DMARQs are only recognized after the EDMA bit has been set.

See Execution Clock Counts for Operation and Bus Controller Latency tables for DMA latency and transfer rate information.

Figure 25. DMA Mode Registers (DMAM)



Trigger DMA [Note 1]	
0	No Effect
1	Trigger DMA

Enable DMA [Note 2]	
0	Disable DMA
1	Enable DMA

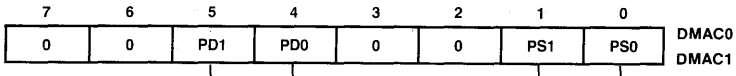
Word/byte	
0	Byte Transfers
1	Word Transfers

DMA Mode			
MD_2	MD_1	MD_0	
0	0	0	Single Step (Mem to Mem)
0	0	1	Demand Release (I/O to Mem)
0	1	0	Demand Release (Mem to I/O)
0	1	1	Reserved
1	0	0	Burst Mode (Mem to Mem)
1	0	1	Single Transfer (I/O to Mem)
1	1	0	Single Transfer (Mem to I/O)
1	1	1	Reserved

Notes:
[1] Valid only during single-step and burst modes.
[2] Cleared when TC = 0; cleared when DMA transfer is aborted by NMI.

49-001390B

Figure 26. DMA Address Control Registers (DMAC)



Source Address Increment/Decrement Control		
PS1	PS0	
0	0	Source Address not Incremented/Decremented
0	1	Increment Source Address
1	0	Decrement Source Address
1	1	Source Address not Incremented/Decremented

Destination Address Increment/Decrement Control		
PD1	PD0	
0	0	Destination Address not Incremented/Decremented
0	1	Increment Destination Address
1	0	Decrement Destination Address
1	1	Destination Address not Incremented/Decremented

49-001391B

4a

Parallel Ports

The μPD70320/322 has three 8-bit parallel I/O ports: P0, P1, and P2. Refer to figures 27 through 31. Special function register (SFR) locations can access these ports. The port lines are individually programmable as inputs or outputs. Many of the port lines have dual functions as port or control lines.

Use the associated port mode and port mode control registers to select the mode for a given I/O line.

Figure 27. Port Mode Registers 0 and 2 (PM0, PM2)

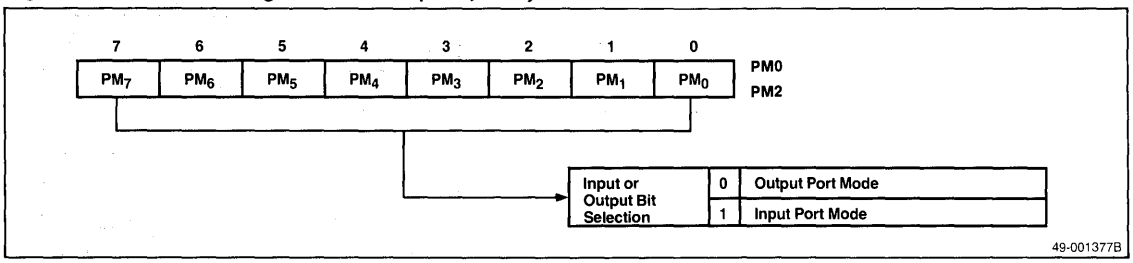


Figure 28. Port Mode Register 1 (PM1)

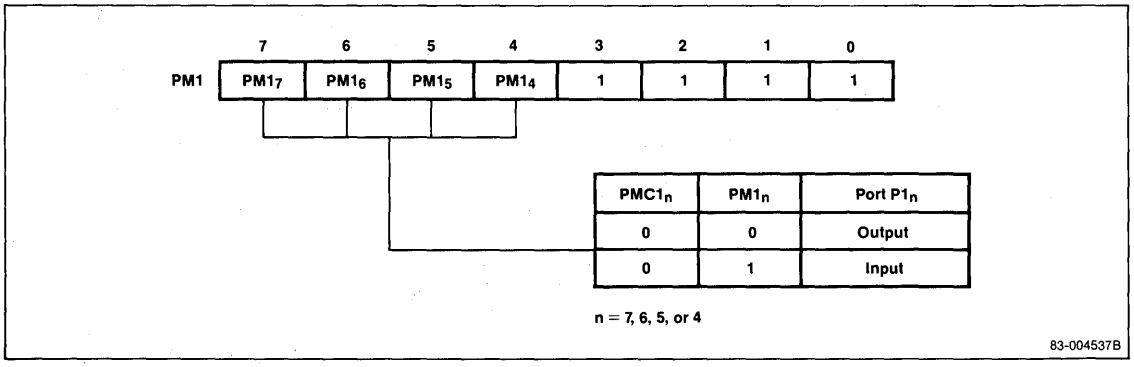


Figure 29. Port Mode Control Register 0 (PMC0)

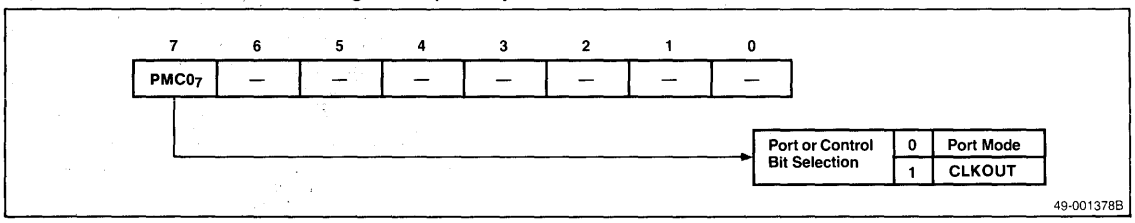


Figure 30. Port Mode Control Register 1 (PMC1)

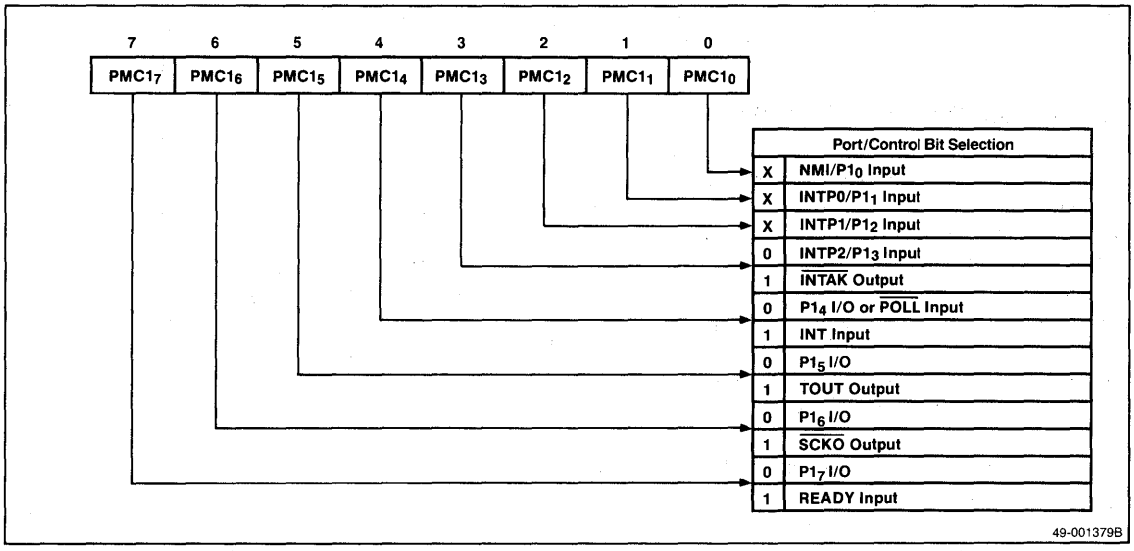
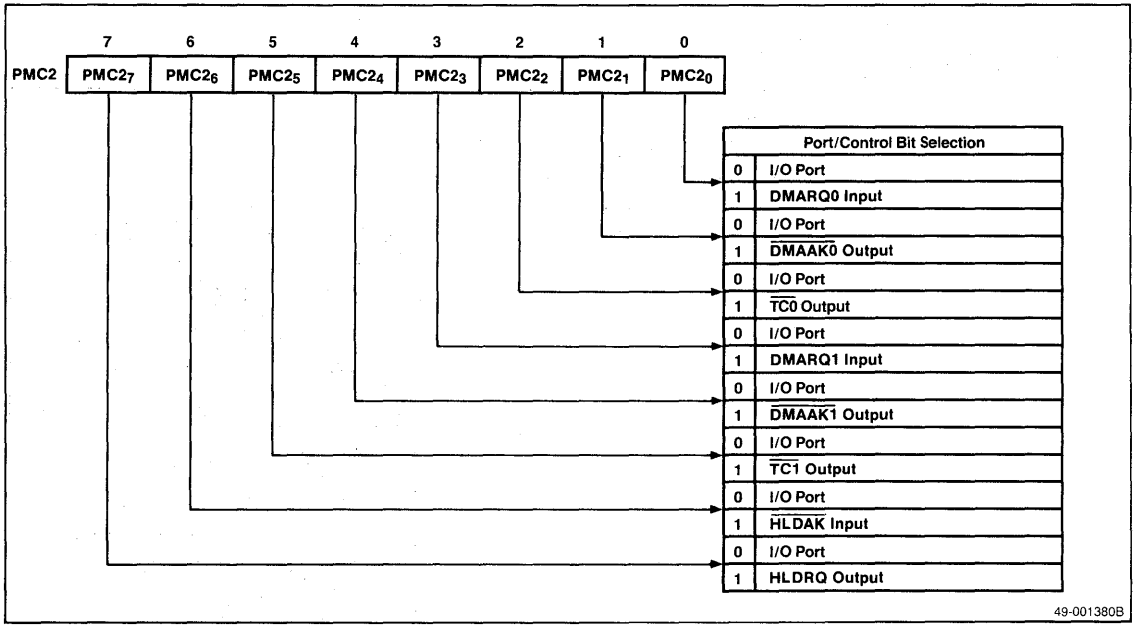


Figure 31. Port Mode Control Register 2 (PMC2)



4a

The analog comparator port (PT) compares each input line to a reference voltage. The reference voltage is programmable to be the V_{TH} input x n/16, where n = 1 to 16. See figure 32.

Programmable Wait State Generation

You can generate wait states internally to further reduce the necessity for external hardware. Insertion of these wait states allows direct interface to devices whose access times cannot meet the CPU read/write timing requirements.

When using this function, the entire 1M-byte memory address space is divided into 128K-blocks. Each block can be programmed for zero, one, or two wait states, or two plus those added by the extenal READY signal. The top two blocks are programmed together as one unit.

The appropriate bits in the wait control word (WTC) control wait state generation. Programming the upper two bits in the wait control word will set the wait state conditions for the entire I/O address space. Figure 33 shows the memory map for programmable wait state generation; see figure 34 for a graphic representation of the wait control word.

Figure 33. Programmable Wait State Generation

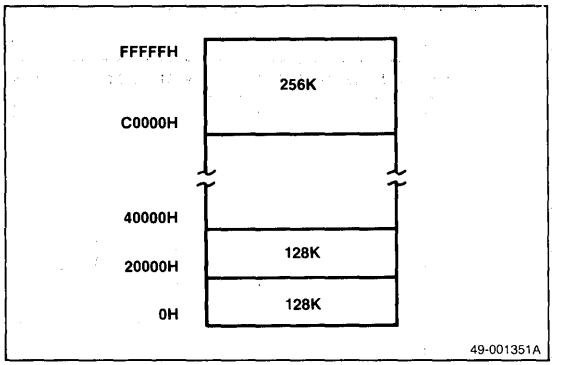


Figure 32. Port Mode Register T (PMT)

7	6	5	4	3	2	1	0	
0	0	0	0	PMT_3	PMT_2	PMT_1	PMT_0	PMT

Comparator Port Threshold Selection				
0	0	0	0	V_{TH} x 16/16
0	0	0	1	V_{TH} x 1/16
0	0	1	0	V_{TH} x 2/16
0	0	1	1	V_{TH} x 3/16
0	1	0	0	V_{TH} x 4/16
0	1	0	1	V_{TH} x 5/16
0	1	1	0	V_{TH} x 6/16
0	1	1	1	V_{TH} x 7/16
1	0	0	0	V_{TH} x 8/16
1	0	0	1	V_{TH} x 9/16
1	0	1	0	V_{TH} x 10/16
1	0	1	1	V_{TH} x 11/16
1	1	0	0	V_{TH} x 12/16
1	1	0	1	V_{TH} x 13/16
1	1	1	0	V_{TH} x 14/16
1	1	1	1	V_{TH} x 15/16

49-001381B

Standby Modes

The two low-power standby modes are HALT and STOP. Software causes the processor to enter either mode.

HALT Mode.

In the HALT mode, the processor is inactive and the chip consumes much less power than when operational. The external oscillator remains functional and all peripherals are active. Internal status and output port line conditions are maintained. Any unmasked interrupt can release this mode. In the EI state, interrupts subsequently will be processed in vector mode. In the DI state, program execution is restarted with the instruction following the HALT instruction.

STOP Mode.

The STOP mode allows the largest power reduction while maintaining RAM. The oscillator is stopped, halting all internal peripherals. Internal status is maintained. Only a reset or NMI can release this mode.

A standby flag in the SFR area is reset by rises in the supply voltage. Its status is maintained during normal operation and standby. The STBC register (figure 35) is not initialized by RESET. Use the standby flag to determine whether program execution is returning from standby or from a cold start by setting this flag before entering the STOP mode.

Figure 34. Wait Control Word

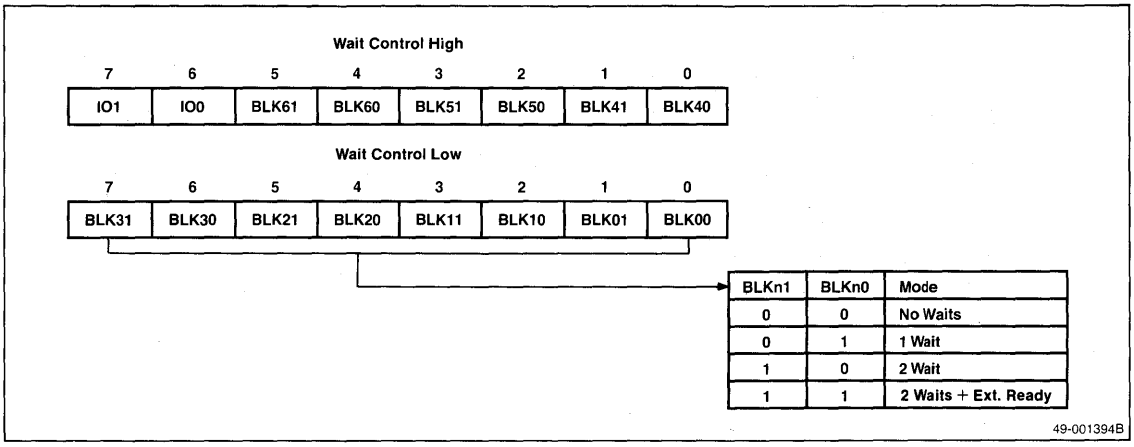
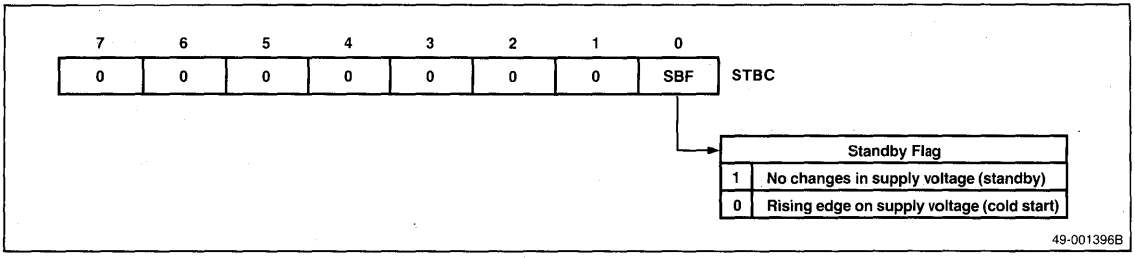


Figure 35. Standby Register



Special Function Registers

Table 6 shows the special function register mnemonic, type, address, reset value, and function. The 8 high-order bits of each address (xx) are specified by the IDB register.

SFR area addresses not listed in table 6 are reserved. If read, the contents of these addresses are undefined, and any write operation will be meaningless.

Table 6. Special Function Registers

Name	Byte/ Word	Address	Reset Value (Note 2)	R/W (Note 1)	Function
P0	B	xxF00H		R/W	Port 0
PM0	B	xxF01H	FFH	W	Port mode 0
PMC0	B	xxF02H	00H	W	Port mode control 0
P1	B	xxF08H		R/W	Port 1
PM1	B	xxF09H	FFH	W	Port mode 1
PMC1	B	xxF0AH	00H	W	Port mode control 1
P2	B	xxF10H		R/W	Port 2
PM2	B	xxF11H	FFH	W	Port mode 2
PMC2	B	xxF12H	00H	W	Port mode control 2
PT	B	xxF38H		R	Port T
PMT	B	xxF3BH	00H	R/W	Port mode T
INTM	B	xxF40H	00H	R/W	Interrupt mode
EMS0	B	xxF44H		R/W	External interrupt macro service 0
EMS1	B	xxF45H		R/W	External interrupt macro service 1
EMS2	B	xxF46H		R/W	External interrupt macro service 2
EXIC0	B	xxF4CH	47H	R/W	External interrupt control 0
EXIC1	B	xxF4DH	47H	R/W	External interrupt control 1
EXIC2	B	xxF4EH	47H	R/W	External interrupt control 2
RXB0	B	xxF60H		R	Receive buffer 0
TXB0	B	xxF62H		W	Transfer buffer 0
SRMS0	B	xxF65H		R/W	Serial receive macro service 0
STMS1	B	xxF66H		R/W	Serial transmit macro service 1
SCM0	B	xxF68H	00H	R/W	Serial communication mode 0
SCC0	B	xxF69H	00H	R/W	Serial communication control 0
BRG0	B	xxF6AH	00H	R/W	Baud rate generator 0
SCE0	B	xxF6BH	00H	R	Serial communication error 0
SEIC0	B	xxF6CH	47H	R/W	Serial error interrupt control 0
SRIC0	B	xxF6DH	47H	R/W	Serial receive interrupt control 0
STIC0	B	xxF6EH	47H	R/W	Serial transmit interrupt control 0
RXB1	B	xxF70H		R	Receive buffer 1
TXB1	B	xxF72H		W	Transmit buffer 1
SRMS1	B	xxF75H		R/W	Serial receive macro service 1
STMS1	B	xxF76H		R/W	Serial transmit macro service 1
SCM1	B	xxF78H	00H	R/W	Serial communication mode 1
SCC1	B	xxF79H	00H	R/W	Serial communication control 1
BRG1	B	xxF7AH	00H	R/W	Baud rate generator register 1
SCE1	B	xxF7BH	00H	R	Serial communication error 1
SEIC1	B	xxF7CH	47H	R/W	Serial error interrupt control 1
SRIC1	B	xxF7DH	47H	R/W	Serial receive interrupt control 1
STIC1	B	xxF7EH	47H	R/W	Serial transmit interrupt control 1
TM0	W	xxF80H		R/W	Timer register 0
TM0L	B	xxF80H		R/W	Timer register 0 low
TM0H	B	xxF81H		R/W	Timer register 0 high
MD0	W	xxF82H		R/W	Modulo register 0

Notes:

(1) Indicates if register is available for read/write operations.

(2) Reset values not specified are undefined.

Table 6. Special Function Registers (cont)

Name	Byte/ Word	Address	Reset Value (Note 2)	R/W (Note 1)	Function
MD0L	B	xxF82H		R/W	Modulo register 0 low
MD0H	B	xxF83H		R/W	Modulo register 0 high
TM1	W	xxF88H		R/W	Timer register 1
TM1L	B	xxF88H		R/W	Timer register 1 low
TM1H	B	xxF89H		R/W	Timer register 1 high
MD1	W	xxF8AH		R/W	Modulo register 1
MD1L	B	xxF8AH		R/W	Modulo register 1 low
MD1H	B	xxF8BH		R/W	Modulo register 1 high
TMC0	B	xxF90H	00H	R/W	Timer control 0
TMC1	B	xxF91H	00H	R/W	Timer control 1
TMMS0	B	xxF94H		R/W	Timer macro service 0
TMMS1	B	xxF95H		R/W	Timer macro service 1
TMMS2	B	xxF96H		R/W	Timer macro service 2
TMIC0	B	xxF9CH	47H	R/W	Timer interrupt control 0
TMIC1	B	xxF9DH	47H	R/W	Timer interrupt control 1
TMIC2	B	xxF9EH	47H	R/W	Timer interrupt control 2
DMAC0	B	xxFA0H		R/W	DMA control 0
DMAM0	B	xxFA1H	00H	R/W	DMA mode 0
DMAC1	B	xxFA2H		R/W	DMA control 1
DMAM1	B	xxFA3H	00H	R/W	DMA mode 1
DIC0	B	xxFACH	47H	R/W	DMA interrupt control 0
DIC1	B	xxFADH	47H	R/W	DMA interrupt control 1
SB	B	xxFE0H		R/W	Standby control
RFM	B	xxFE1H	10H	R/W	Refresh mode
WTC	W	xxFE8H	FFH	R/W	Wait control
WTCL	B	xxFE8H	FFH	R/W	Wait control low
WTCH	B	xxFE9H	FFH	R/W	Wait control high
FLAG	B	xxFEAH	00H	R/W	Flag register
PRC	B	xxFEBH	4EH	R/W	Processor control
TBIC	B	xxFECH	47H	R/W	Time base IRC register
ISPR	B	xxFFCH		R	In service priority register
IDB	B	xxFFFH FFFFFH		R/W	Internal data area base

Absolute Maximum Ratings

$T_A = 25°C$

Supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Output voltage, V_O	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Threshold voltage, V_{TH}	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Output current, low; I_{OL}	
Each output pin	4.0 mA
Total	50 mA
Output current, high; I_{OH}	
Each output pin	−2.0 mA
Total	−20 mA
Operating temperature range, T_{OPT}	−40 to +85°C
Storage temperature range, T_{STG}	−65 to +150°C

Comment: Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

DC Characteristics

$V_{DD} = +5$ V ±10%; $T_A = -10$ to +70°C (Note 1)

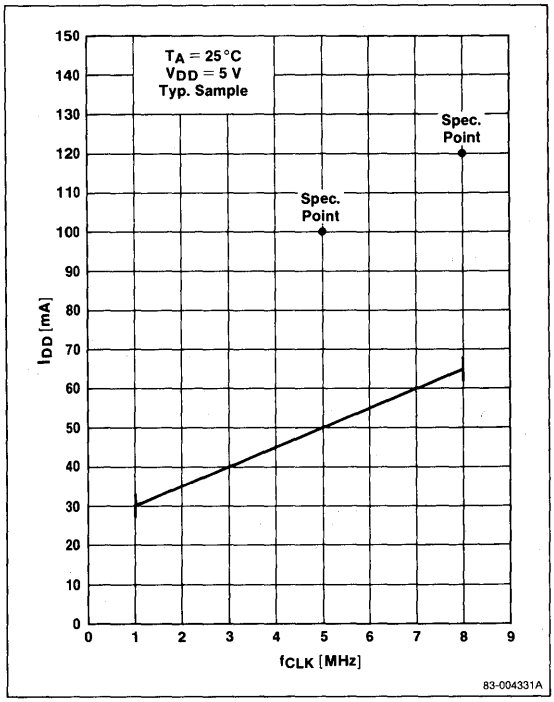
Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Supply current, operating	I_{DD1}		43	100	mA	$f_{CLK} = 5$ MHz
			58	120	mA	$f_{CLK} = 8$ MHz
Supply current, HALT mode	I_{DD2}		17	40	mA	$f_{CLK} = 5$ MHz
			21	50	mA	$f_{CLK} = 8$ MHz
Supply current, STOP mode	I_{DD3}		10	30	μA	
Threshold current	I_{TH}		0.5	1.0	mA	$V_{TH} = 0$ to V_{DD}
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2		V_{DD}	V	All inputs except $\overline{RESET}$, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 x V_{DD}		V_{DD}	V	$\overline{RESET}$, $P1_0$/NMI, X1, X2
Output voltage, low	V_{OL}			0.45	V	$I_{OL} = 1.6$ mA
Output voltage, high	V_{OH}	V_{DD} − 1.0			V	$I_{OH} = -0.4$ mA
Input current	I_{IN}			±20	μA	$\overline{EA}$, $P1_0$/NMI; $V_I = 0$ to V_{DD}
Input leakage current	I_{LI}			±10	μA	All except $\overline{EA}$, $P1_0$/NMI; $V_I = 0$ to V_{DD}
Output leakage current	I_{LO}			±10	μA	$V_O = 0$ to V_{DD}
Data retention voltage	V_{DDDR}	2.5		5.5	V	

Notes:

(1) The standard operating temperature range is −10 to +70°C. However, extended temperature range parts (−40 to +85°C) are available.

4a

Supply Current vs Clock Frequency



Comparator Characteristics

V_{DD} = +5 V ±10%; T_A = −10 to +70°C

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Accuracy	VA_{COMP}		±100	mV	
Threshold voltage	V_{TH}	0	V_{DD} + 0.1	V	
Comparison time	t_{COMP}	64	65	t_{CYK}	
PT input voltage	V_{IPT}	0	V_{DD}	V	

Capacitance Characteristics

V_{DD} = 0 V; T_A = 25°C

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input capacitance	C_I		10	pF	f_C = 1 MHz; Unmeasured pins returned to 0 V
Output capacitance	C_O		20	pF	
I/O capacitance	C_{IO}		20	pF	

AC Characteristics

V_{DD} = +5 V ±10%; T_A = −10 to +70°C; C_L = 100 pF (max)

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
V_{DD} rise, fall time	t_{RVD}, t_{FVD}	200		μs	STOP mode
Input rise, fall time	t_{IR}, t_{IF}		20	ns	Except X1, X2, $\overline{RESET}$, NMI
Input rise, fall time	t_{IRS}, t_{IFS}		30	ns	$\overline{RESET}$, NMI (Schmitt)
Output rise, fall time	t_{OR}, t_{OF}		20	ns	Except CLKOUT
X1 cycle time	t_{CYX}	98	250	ns	Note 3
		62	250	ns	Note 4
X1 width, low	t_{WXL}	35		ns	Note 3
		20		ns	Note 4
X1 width, high	t_{WXH}	35		ns	Note 3
		20		ns	Note 4
X1 rise, fall time	t_{XR}, t_{XF}		20	ns	
CLKOUT cycle time	t_{CYK}	200	2000	ns	Note 3
		125	2000	ns	Note 4
CLKOUT width, low	t_{WKL}	0.5T − 15		ns	Note 1
CLKOUT width, high	t_{WKH}	0.5T − 15		ns	
CLKOUT rise, fall time	t_{KR}, t_{KF}		15	ns	
Address delay time	t_{DKA}	15	90	ns	
Address hold time	t_{HMA}	0.5T − 30		ns	
Address valid to input data valid	t_{DADR}		T(n + 1.5) − 90	ns	Note 2
$\overline{MREQ}$ to data delay	t_{DMRD}		T(n + 1) − 75	ns	
$\overline{MSTB}$ to data delay	t_{DMSD}		T(n + 0.5) − 75	ns	
$\overline{MREQ}$ to MSTB delay	t_{DMRMS}	0.5T − 35	0.5T + 35	ns	
$\overline{MREQ}$ width, low	t_{WMRL}	T(n + 1) − 30		ns	
Input data hold time	t_{HMDR}	0		ns	
Next control setup time	t_{SCC}	T − 25		ns	

Notes:

(1) T = CPU clock period (t_{CYK}).

(2) n = number of wait states inserted.

(3) For 5 MHz parts (μPD70320/322).

(4) For 8 MHz parts (μPD70320/322-8).

AC Characteristics (cont)

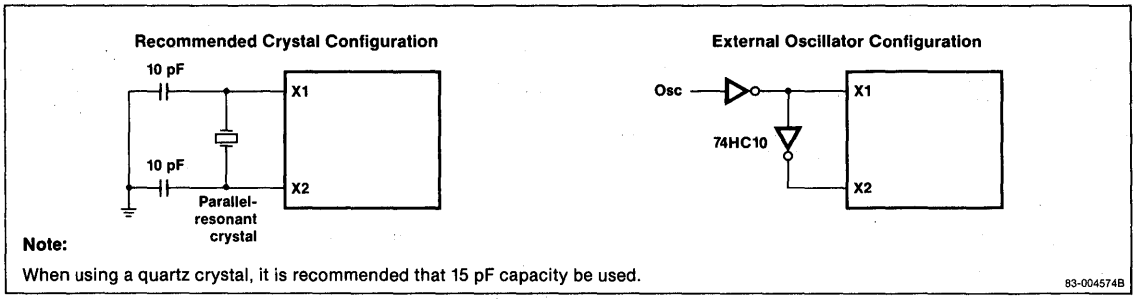
Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
$\overline{MREQ}$ to TC delay time	t_{DMRTC}		0.5T + 50	ns	
Address data output	t_{DADW}		0.5T + 50	ns	
$\overline{MREQ}$ delay time	t_{DAMR}	0.5T − 30	0.5T + 30	ns	
$\overline{MSTB}$ delay time	t_{DAMS}	T − 30	T + 30	ns	
$\overline{MSTB}$ width, low	t_{WMSL}	T(n + 0.5) − 30		ns	
Data output setup time	t_{SDM}	T(n + 1) − 50		ns	
Data output hold time	t_{HMDW}	0.5T − 30		ns	
$\overline{IOSTB}$ delay time	t_{DAIS}	0.5T − 30		ns	
$\overline{IOSTB}$ to data input	t_{DISD}		T(n + 1) − 90	ns	
$\overline{IOSTB}$ width, low	t_{WISL}	T(n + 1) − 30		ns	
Address hold time	t_{HISA}	0.5T − 30		ns	
Input data hold time	t_{HISDR}	0		ns	
Output data setup time	t_{SDIS}	T(n + 1) − 50		ns	
Output data hold time	t_{HISDW}	0.5T − 30		ns	
Next DMARQ setup time	t_{SDADQ}		T	ns	Demand mode
DMARQ hold time	t_{HDADQ}	0		ns	Demand mode
$\overline{DMAAK}$ read width, low	t_{WDMRL}	T(n + 1.5) − 30		ns	
$\overline{DMAAK}$ write width, low	t_{WDMWL}	T(n + 1) − 30		ns	
$\overline{DMAAK}$ to TC delay time	t_{DDATC}		0.5T + 50	ns	
$\overline{TC}$ width, low	t_{WTCL}	2T − 30		ns	
$\overline{REFRQ}$ delay time	t_{DARF}	0.5T − 30		ns	
$\overline{REFRQ}$ width, low	t_{WRFL}	T(n + 1) − 30		ns	
Address hold time	t_{HRFA}	0.5T − 30		ns	

AC Characteristics (cont)

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
$\overline{RESET}$ width low	t_{WRSL1}	30		ms	STOP/ POR (Power-on reset)
	t_{WRSL2}	5		μs	System reset
$\overline{MREQ}$, $\overline{IOSTB}$ to READY setup time	t_{SCRY}		T(n − 1) − 100	ns	n ≥ 2
$\overline{MREQ}$, $\overline{IOSTB}$ to READY hold time	t_{HCRY}	T(n − 1)		ns	n ≥ 2
HLDRQ setup time	t_{SHQK}	30		ns	
$\overline{HLDAK}$ output delay	t_{DKHA}		80	ns	
Bus control float to $\overline{HLDAK}$↓	t_{CFHA}	T − 50		ns	
$\overline{HLDAK}$↑ to control output time	t_{DHAC}	T − 50		ns	
HLDRQ to $\overline{HLDAK}$ delay	t_{DHQHA}		3T + 160	ns	
HLDRQ↓ to control float	t_{DHQC}	3T + 30		ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
$\overline{HLDAK}$ width, low	t_{WHAL}		T	ns	
$\overline{INTP}$, DMARQ setup	t_{SIQK}	30		ns	
$\overline{INTP}$, DMARQ width, high	t_{WIQH}	8T		ns	
$\overline{INTP}$, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{POLL}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{CTS}$ width, low	t_{WCTL}	2T		ns	
INTR setup time	t_{SIRK}	30		ns	
$\overline{INTAK}$ delay time	t_{DKIA}		80	ns	
INTR hold time	t_{HIAIQ}	0		ns	
$\overline{INTAK}$ width, low	t_{WIAL}	2T − 30		ns	
$\overline{INTAK}$ width, high	t_{WIAH}	T − 30		ns	
$\overline{INTAK}$ to data delay	t_{DIAD}		2T − 130	ns	
$\overline{INTAK}$ to data hold	t_{HIAD}	0	0.5T	ns	
$\overline{SCK0}$ (TSCK) cycle time	t_{CYTK}	1000		ns	
$\overline{SCK0}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{SCK0}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	
TxD hold time	t_{HTKD}	20		ns	
$\overline{CTS0}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{CTS0}$ (RSCK) width, high	t_{WSRH}	420		ns	
$\overline{CTS0}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

4a

Figure 36. External System Clock Control Source



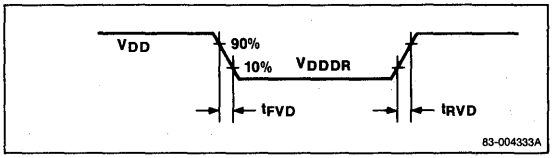
Note:
When using a quartz crystal, it is recommended that 15 pF capacity be used.

Recommended Ceramic Resonator and Capacitance Requirements

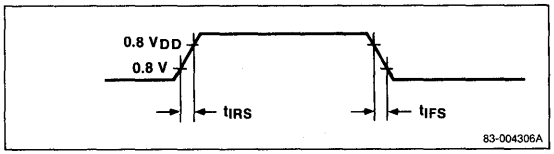
Manufacturer	Product Number	Recommended C1 (pF)	Constants C2 (pF)	Product Number	Recommended C1 (pF)	Constants C2 (pF)
Kyocera	KBR-10.0M	33	33			
Murata Manufacturing	CSA.10.0MT	47	47	CSA16.0MX040	30	30
TDK	FCR10.0M2S	30	30	FCR16.0M2S	15	6

Timing Waveforms

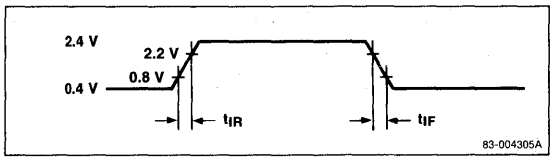
Stop Mode Data Retention Timing



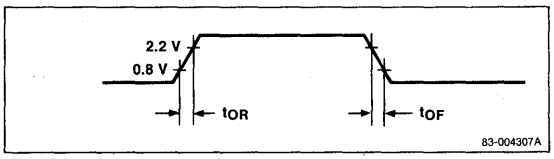
AC Input Waveform 2 (RESET, NMI)



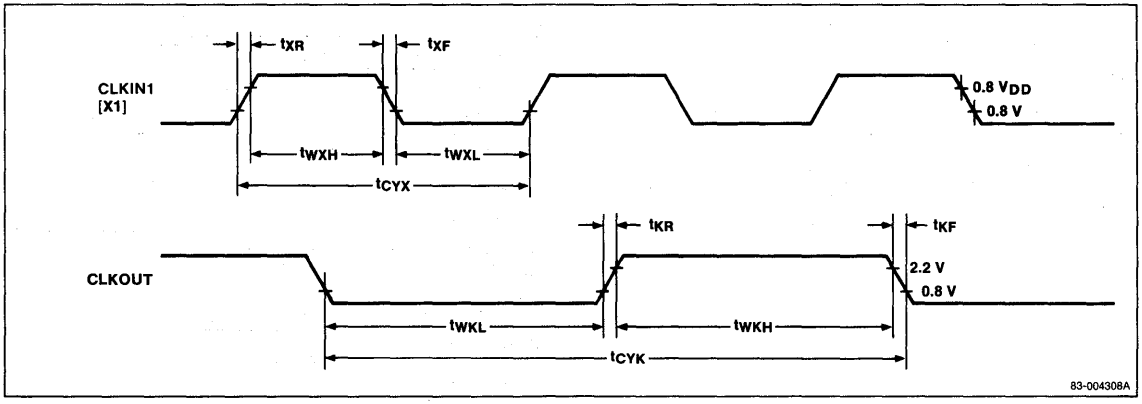
AC Input Waveform 1 (Except X1, X2, RESET, NMI)



AC Output Test Point (Except CLKOUT)

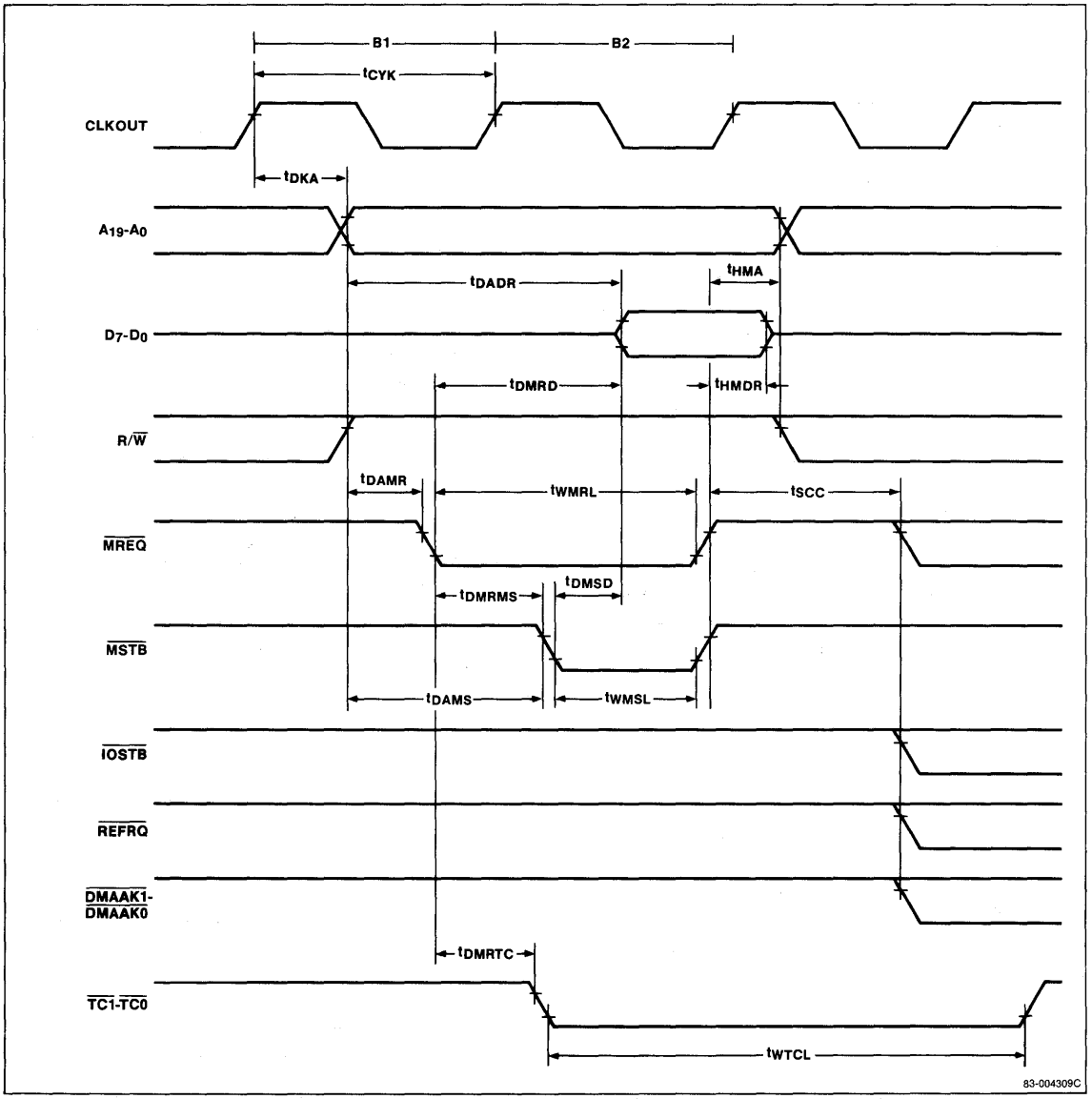


Clock In and Clock Out



Timing Waveforms (cont)

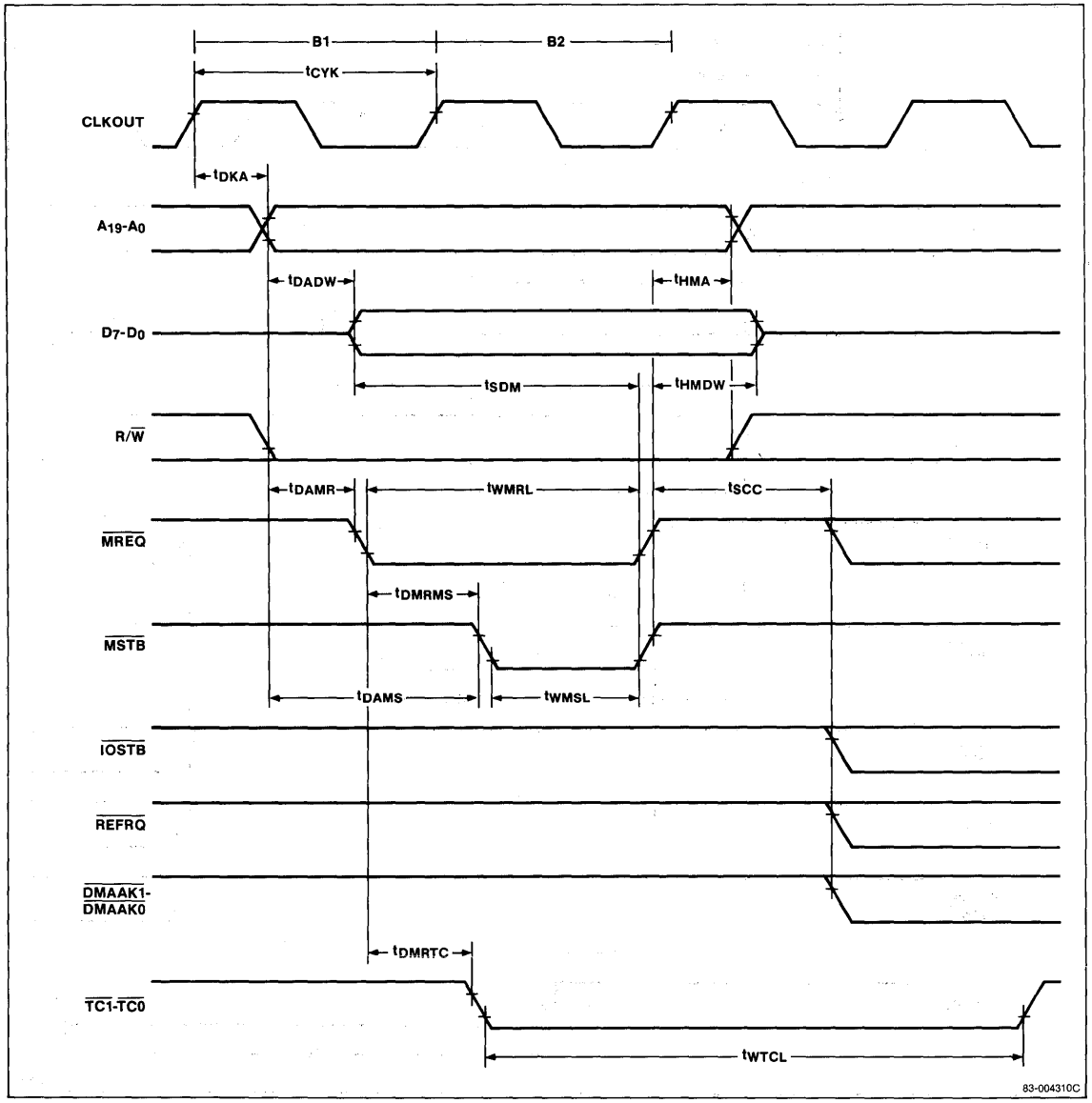
Memory Read



4a

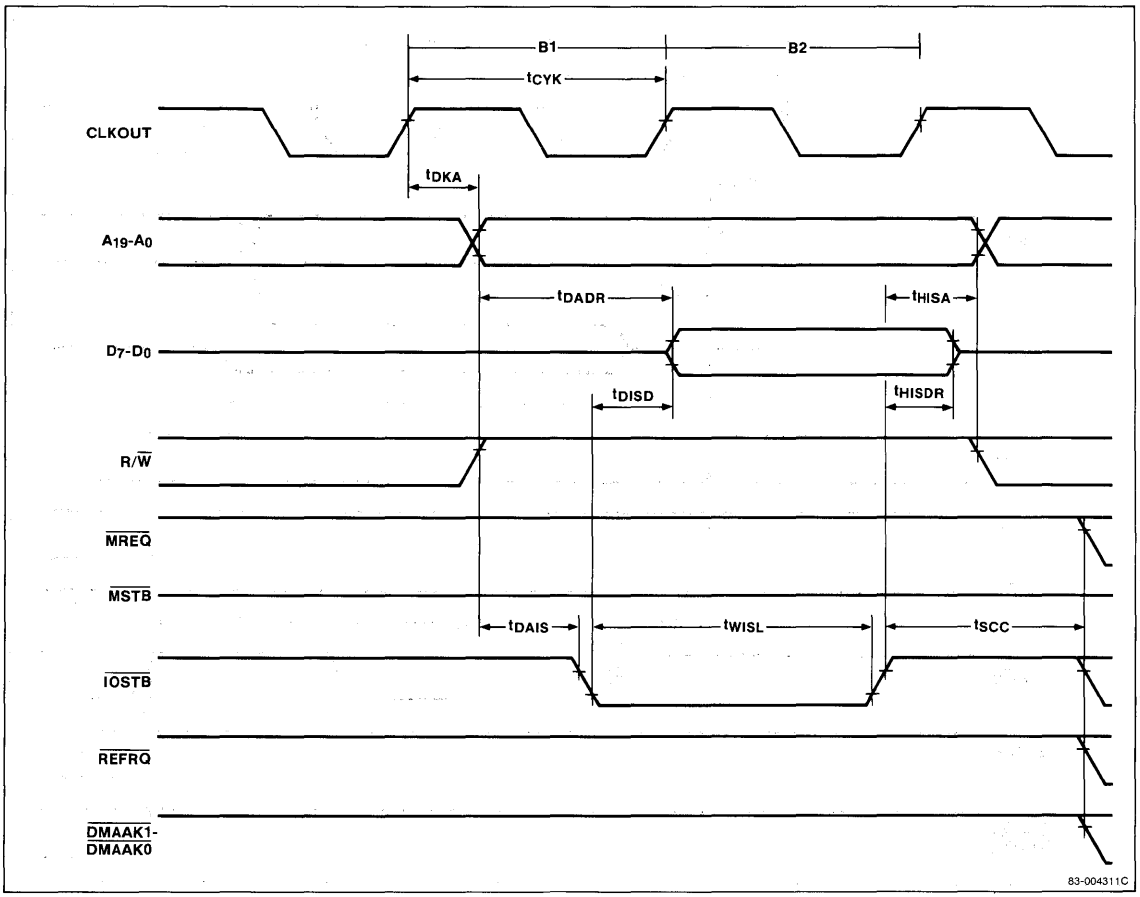
Timing Waveforms (cont)

Memory Write



Timing Waveforms (cont)

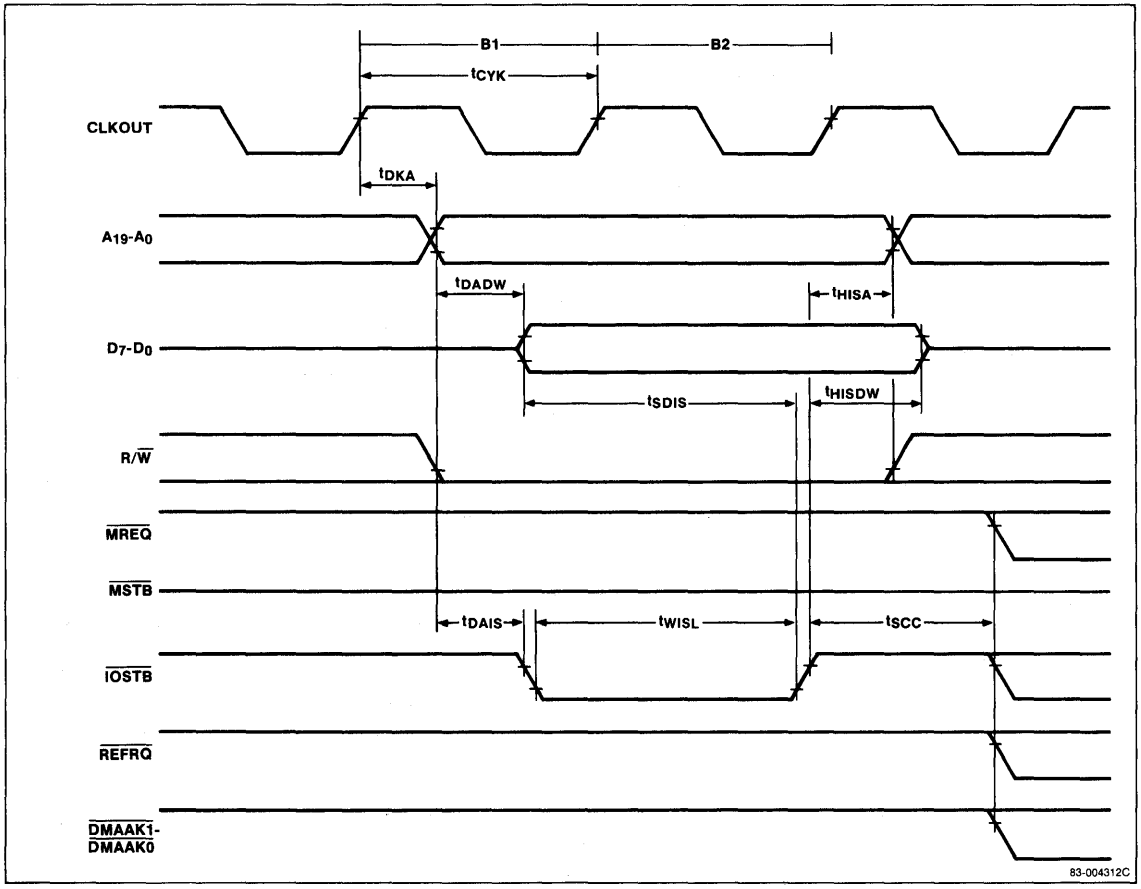
I/O Read



4a

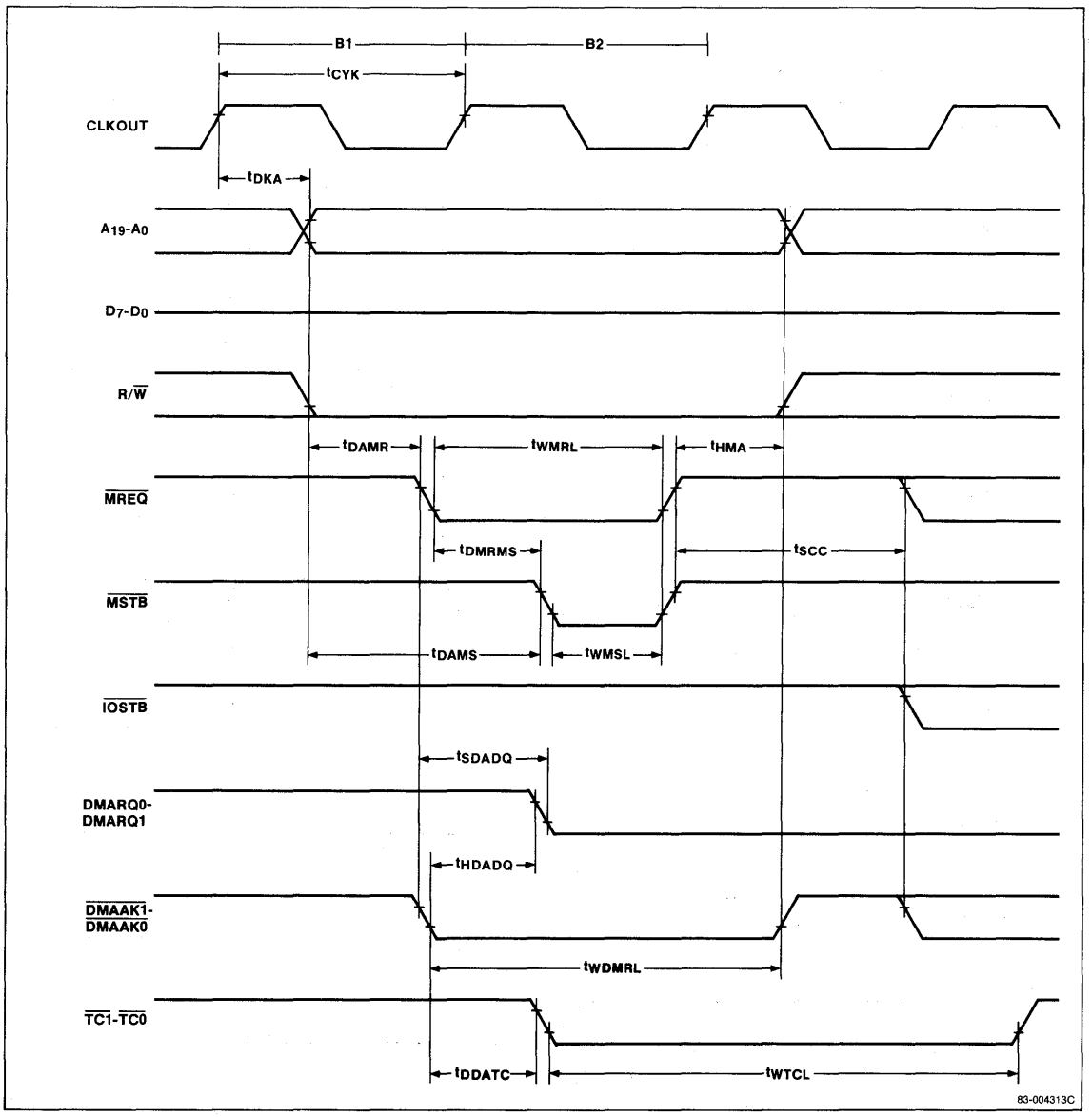
Timing Waveforms (cont)

I/O Write



Timing Waveforms (cont)

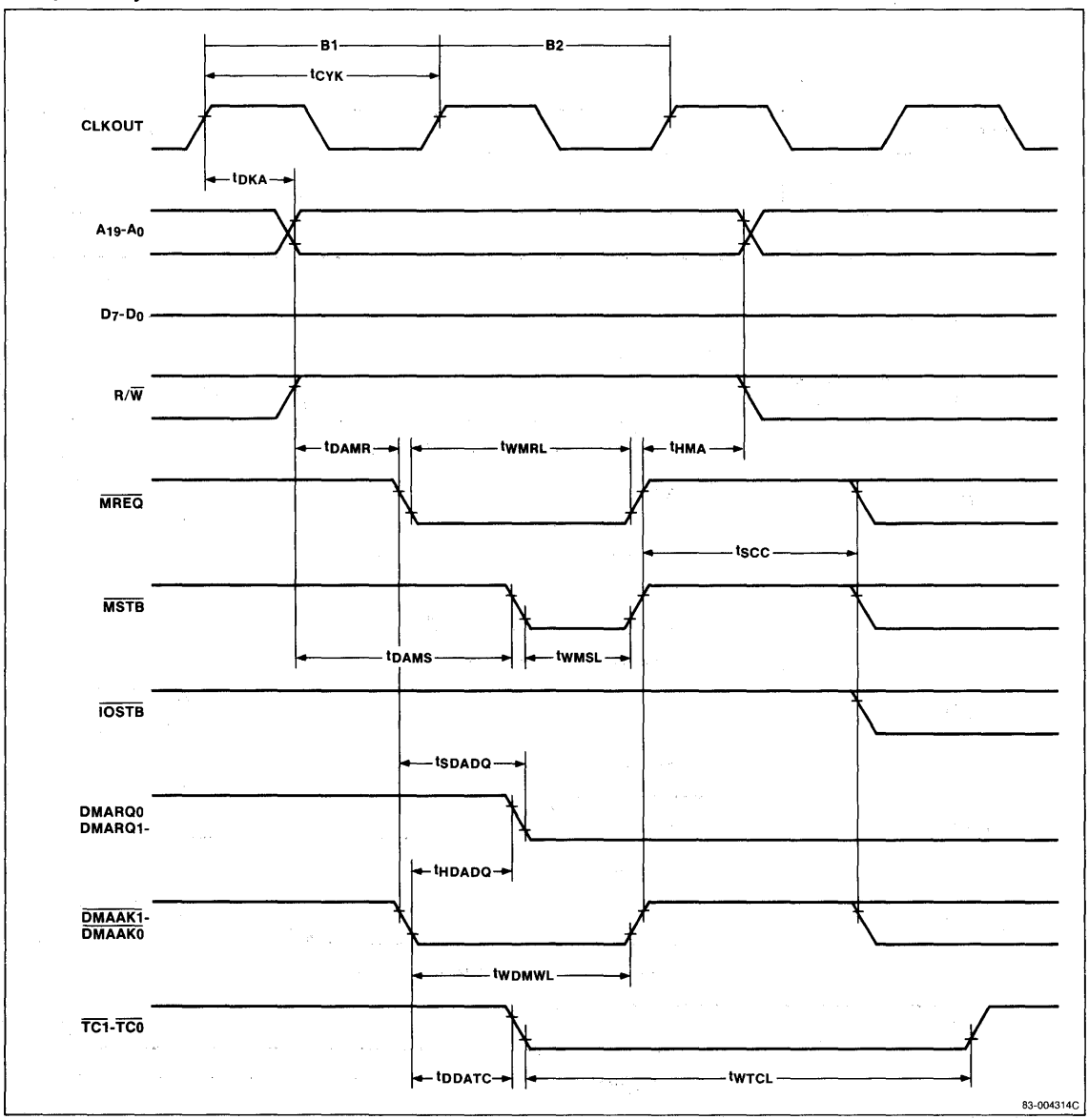
DMA, I/O to Memory



4a

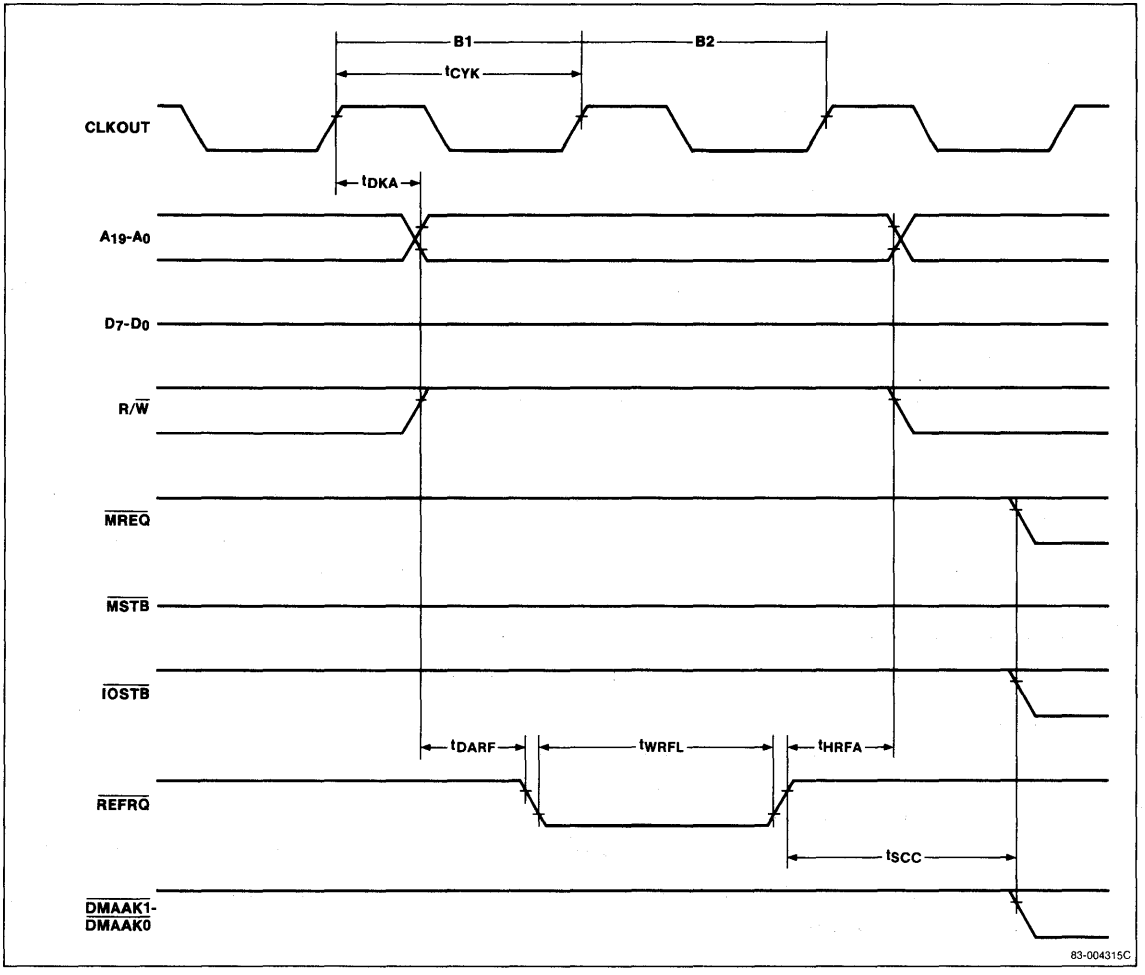
Timing Waveforms (cont)

DMA, Memory to I/O

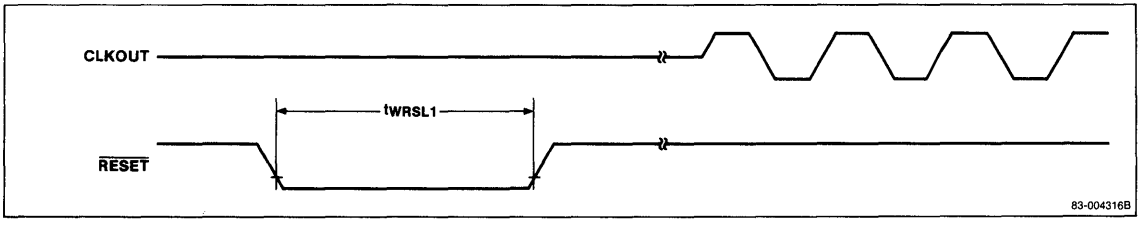


Timing Waveforms (cont)

Refresh



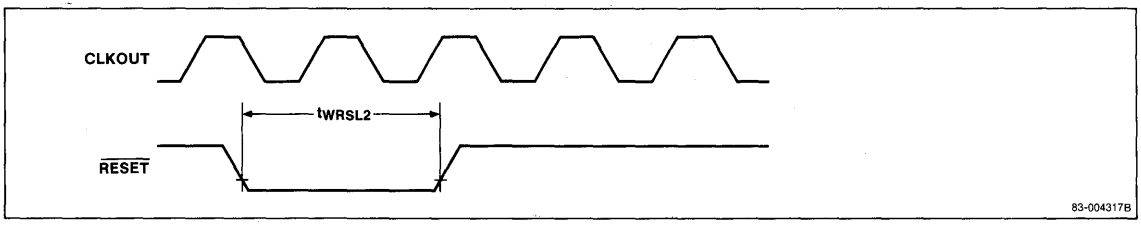
RESET 1



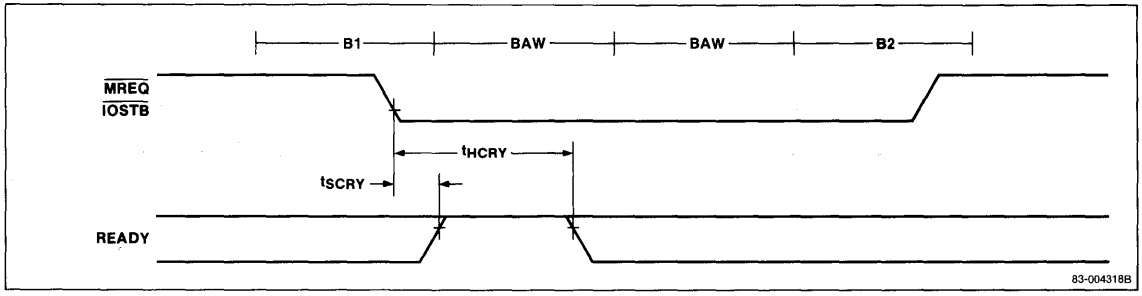
4a

Timing Waveforms (cont)

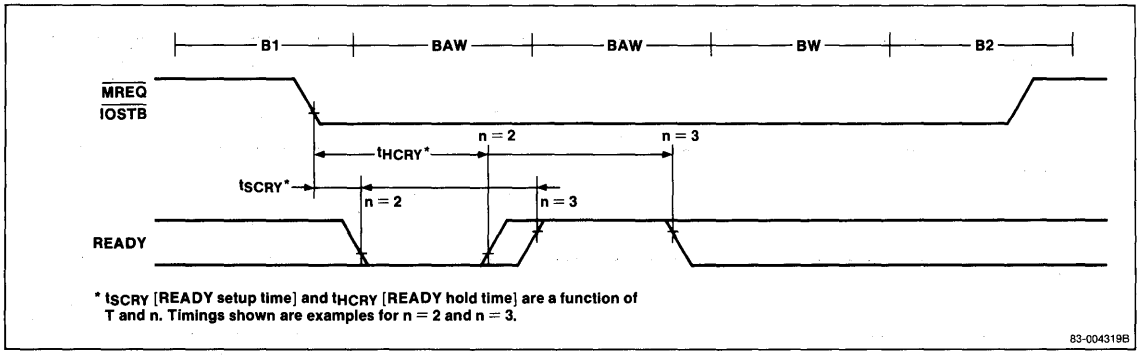
$\overline{RESET}$ 2



READY 1

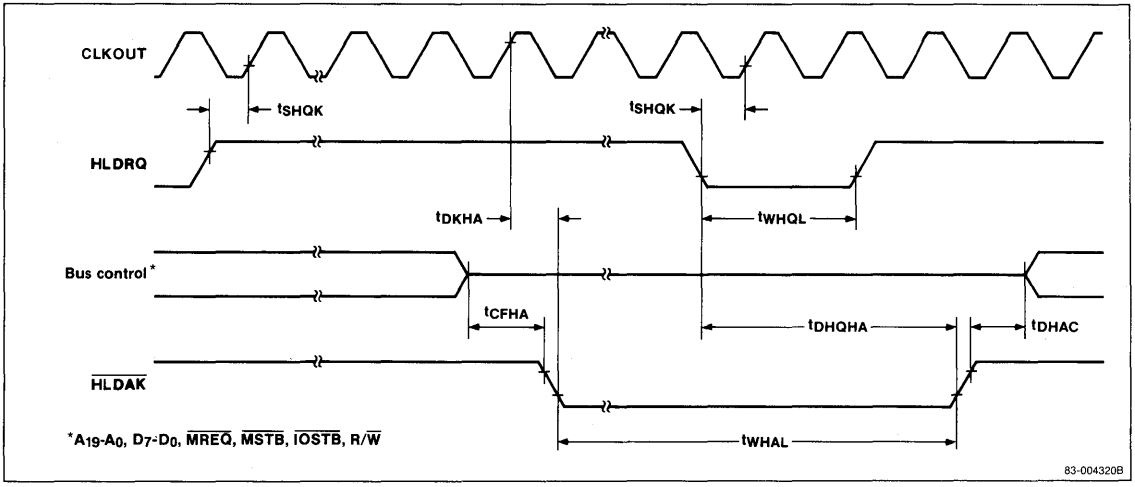


READY 2

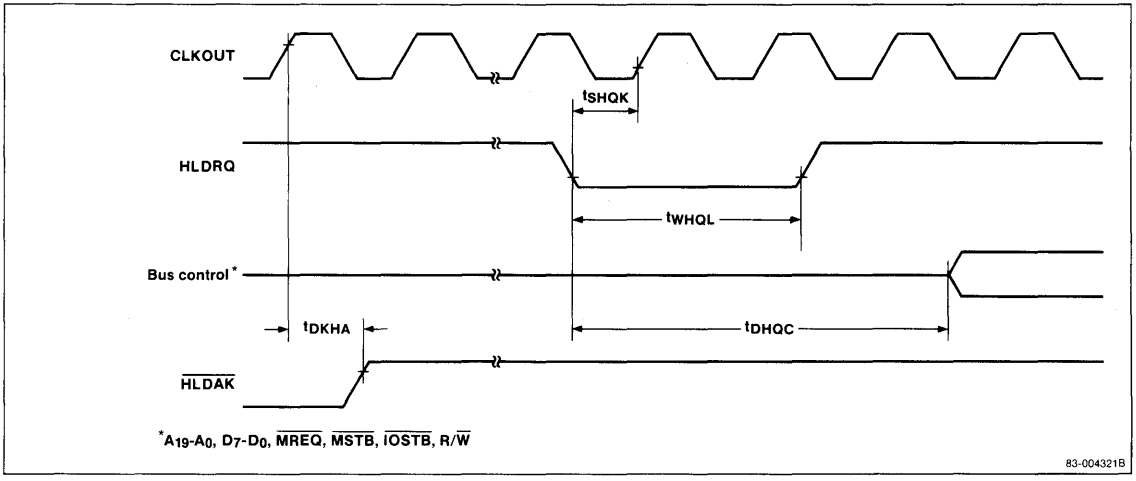


Timing Waveforms (cont)

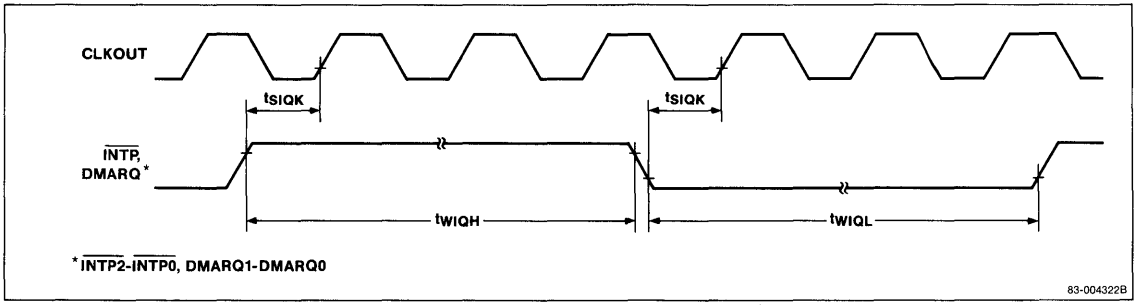
HLDRQ/$\overline{\text{HLDAK}}$ 1



HLDRQ/$\overline{\text{HLDAK}}$ 2



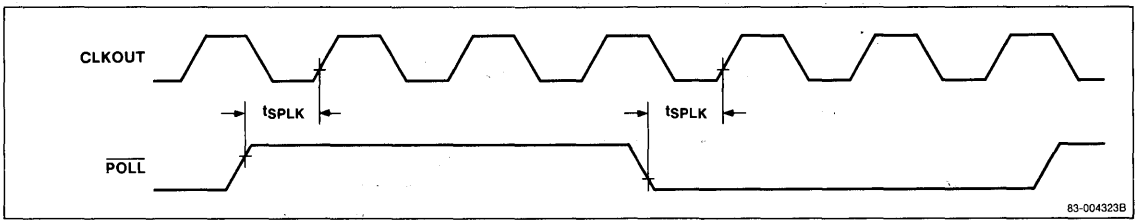
INTP, DMARQ Input



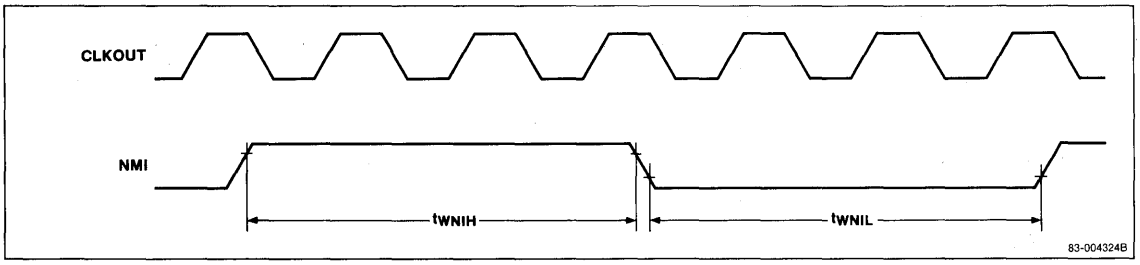
4a

Timing Waveforms (cont)

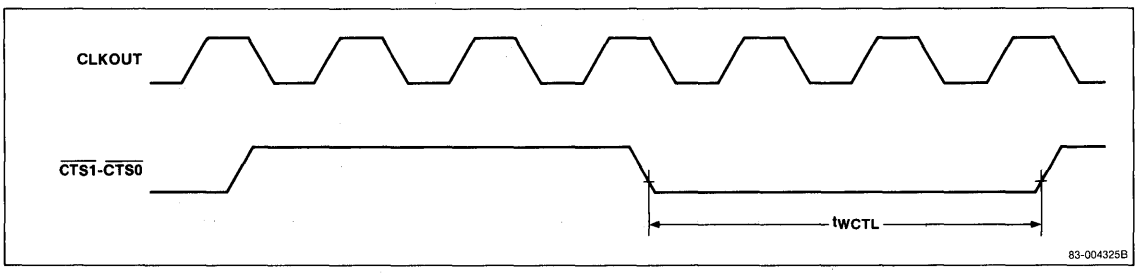
$\overline{POLL}$ Input



NMI Input

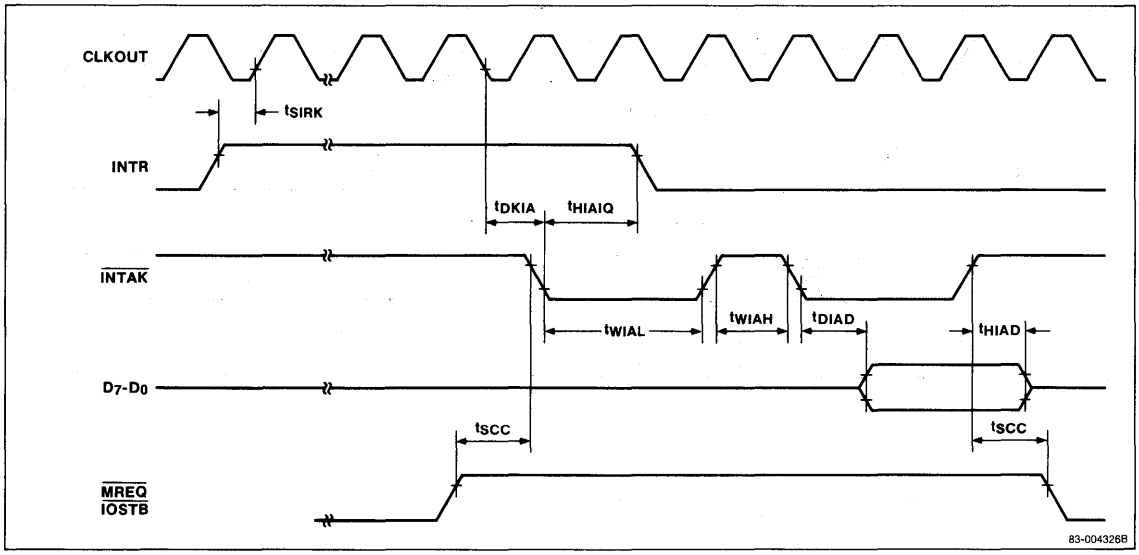


$\overline{CTS}$ Input

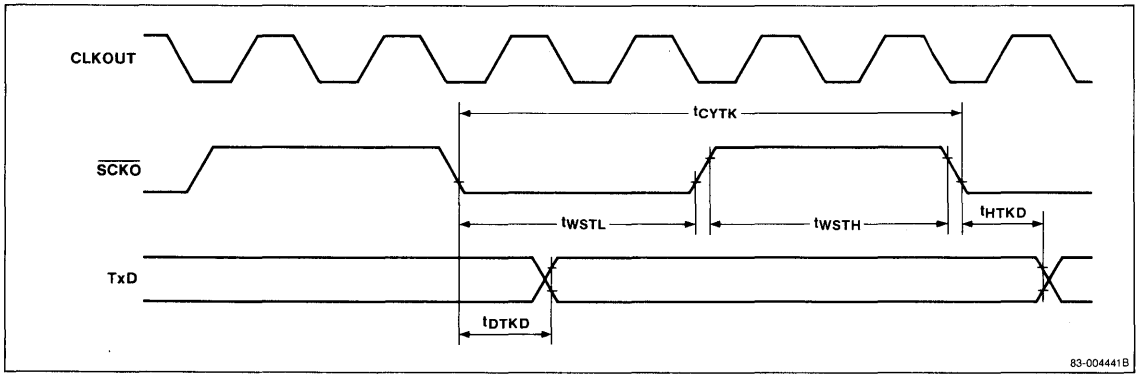


Timing Waveforms (cont)

INTR/$\overline{\text{INTAK}}$



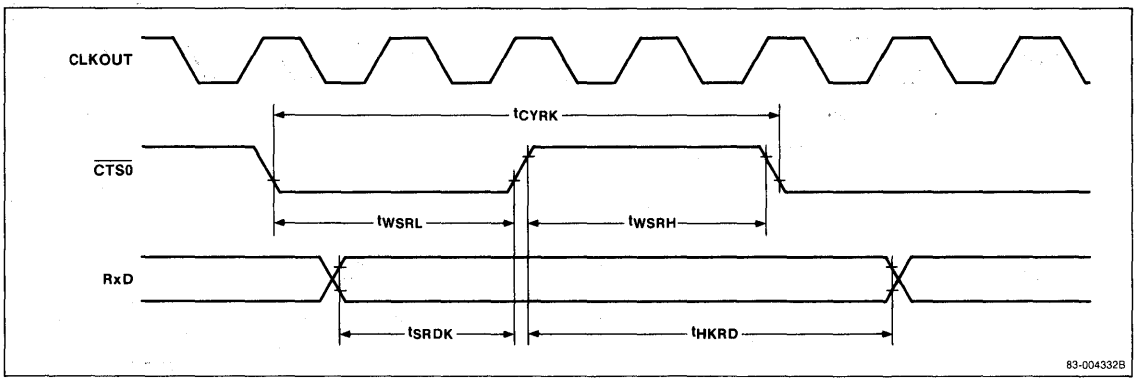
Serial Transmit



4a

Timing Waveforms (cont)

Serial Receive



Instruction Set

Instructions, grouped according to function, are described in a table near the end of this data sheet. Descriptions include source code, operation, opcode, number of bytes, and flag status. Supplementary information applicable to the instruction set is contained in the following tables.

- Symbols and Abbreviations
- Flag Symbols
- 8- and 16-Bit Registers. When mod = 11, the register is specified in the operation code by the byte/word operand (W = 0/1) and reg (000 to 111).
- Segment Registers. The segment register is specified in the operation code by sreg (00, 01, 10, or 11).
- Memory Addressing. The memory addressing mode is specified in the operation code by mod (00, 01, or 10) and mem (000 through 111).
- Instruction Clock Count. This table gives formulas for calculating the number of clock cycles occupied by each type of instruction. The formulas, which depend on byte/word operand and RAM enable/disable, have variables such as EA (effective address), W (wait states), and n (iterations or string instructions).

Symbols and Abbreviations

Identifier	Description
reg	8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
dmem	8- or 16-bit direct memory location
mem	8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
sfr	8-bit special function register location
imm	Constant (0 to FFFFH)
imm16	Constant (0 to FFFFH)
imm8	Constant (0 to FFH)
imm4	Constant (0 to FH)
imm3	Constant (0 to 7)
acc	AW or AL register
sreg	Segment register
src-table	Name of 256-byte translation table
src-block	Name of block addressed by the IX register
dst-block	Name of block addressed by the IY register
near-proc	Procedure within the current program segment
far-proc	Procedure located in another program segment
near-label	Label in the current program segment
short-label	Label between −128 and +127 bytes from the end of instruction
far-label	Label in another program segment
memptr16	Word containing the offset of the memory location within the current program segment to which control is to be transferred
memptr32	Double word containing the offset and segment base address of the memory location to which control is to be transferred
regptr16	16-bit register containing the offset of the memory location within the program segment to which control is to be transferred
pop-value	Number of bytes of the stack to be discarded (0 to 64K bytes, usually even addresses)
fp-op	Immediate data to identify the instruction code of the external floating point operation
R	Register set
W	Word/byte field (0 to 1)
reg	Register field (000 to 111)
mem	Memory field (000 to 111)
mod	Mode field (00 to 10)
S:W	When S:W = 01 or 11, data = 16 bits. At all other times, data = 8 bits.
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
AW	Accumulator (16 bits)
AH	Accumulator (high byte)
AL	Accumulator (low byte)
BP	Base pointer register (16 bits)
BW	BW register (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
CW	CW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
DW	DW register (16 bits)
DH	DW register (high byte)
DL	DW register (low byte)
SP	Stack pointer (16 bits)
PC	Program counter (16 bits)
PSW	Program status word (16 bits)

Symbols and Abbreviations (cont)

Identifier	Description
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
PS	Program segment register (16 bits)
SS	Stack segment register (16 bits)
DS_0	Data segment 0 register (16 bits)
DS_1	Data segment 1 register (16 bits)
AC	Auxiliary carry flag
CY	Carry flag
P	Parity flag
S	Sign flag
Z	Zero flag
DIR	Direction flag
IE	Interrupt enable flag
V	Overflow flag
BRK	Break flag
MD	Mode flag
(...)	Values in parentheses are memory contents
disp	Displacement (8 or 16 bits)
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1-bit)
seg	Immediate segment data (16 bits)
offset	Immediate offset data (16 bits)
←	Transfer direction
+	Addition
−	Subtraction
x	Multiplication
÷	Division
%	Modulo
AND	Logical product
OR	Logical sum
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value

Flag Symbols

Identifier	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
X	Set or cleared according to the result
U	Undefined
R	Value saved earlier is restored

8- and 16-Bit Registers (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Registers

sreg	Register
00	DS_1
01	PS
10	SS
11	DS_0

Memory Addressing

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Instruction Clock Count

Mnemonic	Operand	Clocks
ADD	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+8+2W [EA+6+W]
	mem16, reg16	EA+12+4W [EA+8+2W]
	reg8, imm8	5
	reg16, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+9+2W [EA+7+2W]
	mem16, imm8	EA+9+2W [EA+7+2W]
	mem16, imm16	EA+14+4W [EA+10+4W]
	AL, imm8	5
	AW, imm16	6
ADD4S		22+(27+3W)n [22+(25+3W)n]
ADDC		Same as ADD
ADJ4A		9
ADJ4S		9
ADJBA		17
ADJBS		17
AND	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+8+2W [EA+6+W]
	mem16, reg16	EA+12+4W [EA+8+2W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+9+2W [EA+7+2W]
	mem16, imm16	EA+14+4W [EA+10+4W]
Bcond (conditional branch)		8 or 15
BCWZ		8 or 15
BR	near-label	12
	short-label	12
	regptr16	13
	memptr16	EA+17+2W
	far-label	15
	memptr32	EA+25+4W

Mnemonic	Operand	Clocks
BRK	3	55+10W [43+10W]
	imm8	56+10W [44+10W]
BRKCS		15
BRKV		55+10W [43+10W]
BTCLR		29
BUSLOCK		2
CALL	near-proc	22+2W [18+2W]
	regptr16	22+2W [18+2W]
	memptr16	EA+26+4W [EA+24+4W]
	far-proc	36+4W [34+4W]
	memptr32	EA+36+8W [EA+24+8W]
CHKIND		EA+26+4W
CLR1	CY	2
	DIR	2
	reg8, CL	8
	reg16, CL	8
	mem8, CL	EA+14+2W [EA+12+W]
	mem16, CL	EA+18+4W [EA+14+2W]
	reg8, imm3	7
	reg16, imm4	7
	mem8, imm3	EA+11+2W [EA+9+W]
	mem16, imm4	EA+15+4W [EA+10+2W]
CMP	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+6+W
	mem16, reg16	EA+8+2W
	reg8, imm8	5
	reg16, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+7+W
	mem16, imm8	EA+10+2W
	mem16, imm16	EA+10+2W
	AL, imm8	5
	AW, imm16	6
CMP4S		22+(23+2W)n
CMPBK	mem8, mem8	23+2W [19+2W]
	mem16, mem16	27+4W [21+2W]

Notes:

(1) If the number of clocks is not the same for RAM enabled and RAM disabled conditions, the RAM enabled value is listed first, followed by the RAM disabled value in brackets; for example, EA+8+2W [EA+6+W].

(2) Symbols in the Clocks column are defined as follows.

EA = additional clock cycles required for calculation of the effective address

= 3 (mod 00 or 01) or 4 (mod 10)

W = number of wait states selected by the WTC register

n = number of iterations or string instructions

4a

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
CMPBKB		16+(21+2W)n
CMPBKW		16+(25+4W)n
CMPM	mem8 mem16	17+W 19+2W
CMPMB		16+(15+W)n
CMPMW		16+(17+2W)n
CVTBD		19
CVTBW		3
CVTDB		20
CVTWL		8
DBNZ		8 or 17
DBNZE		8 or 17
DBNZNE		8 or 17
DEC	reg8 reg16	5 2
	mem8 mem16	EA+11+2W [EA+9+2W] EA+15+4W [EA+11+4W]
DI		4
DISPOSE		12+2W
DIV	AW, reg8 AW, mem8	46-56 EA+48+W to EA+58+W
	DW:AW, reg16 DW:AW, mem16	54-64 EA+58+2W to EA+68+2W
DIVU	AW, reg8 AW, mem8	31 EA+33+W
	DW:AW, reg16 DW:AW, mem16	39 EA+43+2W
DS0:		2
DS1:		2
EI		12
EXT	reg8, reg8 reg8, imm4	41-121 42-122
FINT		2
FPO1		60+10W [48+10W]
FPO2		60+10W [48+10W]
HALT		0
IN	AL, imm8 AW, imm8	14+W 16+2W
	AL, DW AW, DW	13+W 15+2W
INC	reg8 reg16	5 2
	mem8 mem16	EA+11+2W [EA+9+2W] EA+15+4W [EA+11+4W]

Mnemonic	Operand	Clocks
INM	mem8, DW mem16, DW	19+2W [17+2W] 21+4W [17+4W]
	mem8, DW mem16, DW	18+(13+2W)n [18+(11+2W)n] 18+(15+4W)n [18+(11+4W)n]
INS	reg8, reg8 reg8, imm4	63-155 64-156
LDEA		EA+2
LDM	mem8	12+W
	mem16	16+(12+2W)n
LDMB	mem16	14+2W
LDMW	mem8	16+(10+W)n
MOV	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+6+W EA+8+2W
	mem8, reg8 mem16, reg16	EA+4+W [EA+2] EA+6+2W [EA+2]
	reg8, imm8 reg16, imm16	5 6
	mem8, imm8 mem16, imm16	EA+5+W EA+5+2W
	AL, dmem8 AW, dmem16	9+W 11+2W
	dmem8, AL dmem16, AW	7+W [5] 9+2W [5]
	sreg, reg16 sreg, mem16	4 EA+10+2W
	reg16, sreg mem16, sreg	3 EA+7+2W [EA+3]
	AH, PSW PSW, AH	2 3
	DS0, reg16, memptr32 DS1, reg16, memptr32	EA+19+4W EA+19+4W
MOVBK	mem8, mem8 mem16, mem16	20+2W [16+W] 16+(20+4W)n [16+(12+2W)n]
MOVBKB	mem8, mem8	16+(16+2W)n [16+(12+W)n]
MOVBKW	mem16, mem16	24+4W [20+2W]
MOVSPA		16
MOVSPB		11
MUL	AW, AL, reg8 AW, AL, mem8	31-40 EA+33+W to EA+42+W
	DW:AW, AW, reg16 DW:AW, AW, mem16	39-48 EA+43+2W to EA+52+2W
	reg16, reg16, imm8 reg16, mem16, imm8	39-49 EA+43+2W to EA+53+2W
	reg16, reg16, imm16 reg16, mem16, imm16	40-50 EA+44+2W to EA+54+2W

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
MULU	reg8	24
	mem8	EA+26+W
	reg16	32
	mem16	EA+34+2W
NEG	reg8	5
	reg16	5
	mem8	EA+11+2W [EA+9+W]
	mem16	EA+15+4W [EA+11+2W]
NOP		3
NOT	reg8	5
	reg16	5
	mem8	EA+11+2W [EA+9+W]
	mem16	EA+15+4W [EA+11+2W]
NOT1	CY	2
	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+11+W
	mem16, CL	EA+13+2W
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+8+W
	mem16, imm4	EA+10+2W
OR	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+8+2W [EA+6+2W]
	mem16, reg16	EA+12+4W [EA+8+4W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+9+W [EA+7+W]
	mem16, imm16	EA+14+4W [EA+10+4W]
	AL, imm8	5
	AW, imm16	6
OUT	imm8, AL	10+W
	imm8, AW	10+2W
	DW, AL	9+W
	DW, AW	9+2W
OUTM	DW, mem8	19+2W [17+2W]
	DW, mem16	21+4W [17+4W]
	DW, mem8	18+(13+2W)n [18+(11+2W)n]
	DW, mem16	18+(15+4W)n [18+11+4W)n]
POLL		0
POP	reg16	12+2W
	mem16	EA+16+4W [EA+12+2W]
	DS1	13+2W
	SS	13+2W
	DS0	13+2W
	PSW	14+2W
	R	82+16W [58]

Mnemonic	Operand	Clocks
PREPARE	imm16, imm8	imm8 = 0: 27+2W imm8 = 1: 39+4W imm8 = n > 1: 46+19 (n−1)+4W
PS:		2
PUSH	reg16	10+2W [6]
	mem16	EA+18+4W [EA+14+4W]
	DS1	11+2W [7]
	PS	11+2W [7]
	SS	11+2W [7]
	DS0	11+2W [7]
	PSW	10+2W [6]
	R	82+16W [50]
	imm8	13+2W [9]
	imm16	14+2W [10]
REP		2
REPE		2
REPZ		2
REPC		2
REPNC		2
REPNE		2
REPNZ		2
RET	null	20+2W
	pop-value	20+2W
	null	29+4W
	pop-value	30+4W
RETI		43+6W [35+2W]
RETRBI		12
ROL	reg8, 1	8
	reg16, 1	8
	mem8, 1	EA+14+2W [EA+12+W]
	mem16, 1	EA+18+4W [EA+14+2W]
	reg8, CL	11+2n
	reg16, CL	11+2n
	mem8, CL	EA+17+2W+2n [EA+15+W+2n]
	mem16, CL	EA+21+4W+2n [EA+17+2W+2n]
	reg8, imm8	9+2n
	reg16, imm8	9+2n
	mem8, imm8	EA+13+2W+2n [EA+11+W+2n]
	mem16, imm8	EA+17+4W+2n [EA+13+2W+2n]
ROL4	reg8	17
	mem8	EA+18+2W [EA+16+2W]
ROLC		Same as ROL
ROR		Same as ROL
ROR4	reg8	21
	mem8	EA+24+2W [EA+22+2W]
RORC		Same as ROL
SET1	CY	2
	DIR	2

4a

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
SET1 (cont)	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+13+2W [EA+11+W]
	mem16, CL	EA+17+4W [EA+13+2W]
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+10+2W [EA+8+W]
	mem16, imm4	EA+14+4W [EA+10+2W]
SHL		Same as ROL
SHR		Same as ROL
SHRA		Same as ROL
SS:		2
STM	mem8	12+2 [10]
	mem16	16+(10+2W)n [16+(6+2W)n]
STMB	mem8	16+(8+W)n [16+(6+W)n]
STMW	mem16	14+2W [10]
STOP		0
SUB		Same as ADD
SUB4S		22+(27+3W)n [22+(25+3W)n]
SUBC		Same as ADD
TEST	reg8, reg8	4
	reg16, reg16	4
	reg8, mem8	EA+8+W
	reg16, mem16	EA+10+2W
	mem8, reg8	EA+8+W
	mem16, reg16	EA+10+2W
	reg8, imm8	7
	reg16, imm16	8
	mem8, imm8	EA+11+W
	mem16, imm16	EA+11+2W
	AL, imm8	5
	AW, imm16	6
TEST1	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+11+W
	mem16, CL	EA+13+2W
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+8+W
	mem16, imm4	EA+10+2W
TRANS		10+W
TRANSB		10+W
TSKSW		11

Mnemonic	Operand	Clocks
XCH	reg8, reg8	3
	reg16, reg16	3
	reg8, mem8	EA+10+2W [EA+8+2W]
	reg16, mem16	EA+14+4W [EA+10+4W]
	mem8, reg8	EA+10+2W [EA+8+2W]
	mem16, reg16	EA+14+4W [EA+10+4W]
	AW, reg16	4
	reg16, AW	4
XOR		Same as AND

Execution Clock Counts for Operations

	Byte		Word	
	RAM Enable	RAM Disable	RAM Enable	RAM Disable
Context switch interrupt (Note 1)	—	—	28	28
DMA (Single-step mode) (Note 2)	20 + 2W	20 + 2W	24 + 4W	24 + 4W
DMA (Demand release mode)	15 + W	15 + W	17 + 2W	17 + 2W
DMA (Burst mode)	(12 + 2W)n	(12 + 2W)n	(12 + 4W)n	(12 + 4W)n
DMA (Single-transfer mode)	33 + W + N	33 + W + N	35 + 2W + N	35 + 2W + N
Interrupt (INT pin)	—	—	62 + 10W	50 + 10W
Macro service, sfr ← mem (Note 2)	23 + W	21 + W	27 + 2W	25 + 2W
Macro service, mem ← sfr	22 + W	20 + W	26 + 2W	24 + 2W
Macro service (Search char mode), sfr ← mem	37 + W	37 + W	—	—
Macro service (Search char mode), mem ← sfr	37 + W	37 + W	—	—
Priority vectored interrupt (Note 1)	—	—	58 + 10W	46 + 10W

N = number of clocks to complete the instruction currently executing.

Notes:

(1) Every interrupt has an additional associated latency time of 27 + N clocks. During the 27 clocks the interrupt controller performs some overhead tasks such as arbitrating priority. This time should be added to the above listed interrupt and macro service execution times.

(2) The DMA and macro service clock counts listed are the required number of CPU clocks for each transfer.

(3) When an external interrupt is asserted, a maximum of 6 clocks is required for internal synchronization before the interrupt request flag is set. For an internal interrupt, a maximum of 2 clocks is required.

Bus Controller Latency

	Mode	Clocks
HLDRQ latency		7 + 2W
DMA request latency (Note 1)	Burst	29 + N
	Single step	29 + N
	Demand release	29 + N
	Single transfer	31 + N

Notes:

(1) The listed DMA latency times are the maximum number of clocks when a DMA request is asserted until DMAAK or MREQ goes low in the corresponding DMA cycles. The test conditions are no wait states, no interrupts, no macro service requests and no hold requests.

4a

Instruction Set

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Data Transfer																									
MOV	reg, reg	reg ← reg	1	0	0	0	1	0	1	W	1	1		reg			reg		2						
	mem, reg	(mem) ← reg	1	0	0	0	1	0	0	W	mod			reg			mem		2-4						
	reg, mem	reg ← (mem)	1	0	0	0	1	0	1	W	mod			reg			mem		2-4						
	mem, imm	(mem) ← imm	1	1	0	0	0	1	1	W	mod		0	0	0		mem		3-6						
	reg, imm	reg ← imm	1	0	1	1	W		reg										2-3						
	acc, dmem	When W = 0 AL ← (dmem) When W = 1 AH ← (dmem + 1), AL ← (dmem)	1	0	1	0	0	0	0	W									3						
	dmem, acc	When W = 0 (dmem) ← AL When W = 1 (dmem + 1) ← AH, (dmem) ← AL	1	0	1	0	0	0	1	W									3						
	sreg, reg16	sreg ← reg16 sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	1	1	0	sreg			reg		2						
	sreg, mem16	sreg ← (mem16) sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	mod		0	sreg			mem		2-4						
	reg16, sreg	reg16 ← sreg	1	0	0	0	1	1	0	0	1	1	0	sreg			reg		2						
	mem16, sreg	(mem16) ← sreg	1	0	0	0	1	1	0	0	mod		0	sreg			mem		2-4						
	DS0, reg16, mem32	reg16 ← (mem32) DS0 ← (mem32 + 2)	1	1	0	0	0	1	0	1	mod			reg			mem		2-4						
	DS1, reg16, mem32	reg16 ← (mem32) DS1 ← (mem32 + 2)	1	1	0	0	0	1	0	0	mod			reg			mem		2-4						
	AH, PSW	AH ← S, Z, x, AC, x, P, x, CY	1	0	0	1	1	1	1	1									1						
	PSW, AH	S, Z, x, AC, x, P, x, CY ← AH	1	0	0	1	1	1	1	0									1	x	x		x	x	x
LDEA	reg16, mem16	reg16 ← mem16	1	0	0	0	1	1	0	1	mod			reg			mem		2-4						
TRANS	src-table	AL ← (BW + AL)	1	1	0	1	0	1	1	1									1						
XCH	reg, reg	reg ⟷ reg	1	0	0	0	0	1	1	W	1	1		reg			reg		2						
	mem, reg or reg, mem	(mem) ⟷ reg	1	0	0	0	0	1	1	W	mod			reg			mem		2-4						
	AW, reg16 or reg16, AW	AW ⟷ reg16	1	0	0	1	0		reg										1						
Repeat Prefixes																									
REPC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. When CY ≠ 1, exit the loop.	0	1	1	0	0	1	0	1									1						
REPNC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. When CY ≠ 0, exit the loop.	0	1	1	0	0	1	0	0									1						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Repeat Prefixes (cont)																									
REP REPE REPZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 1, exit the loop.	1	1	1	1	0	0	1	1									1						
REPNE REPNZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 0, exit the loop.	1	1	1	1	0	0	1	0									1						
Primitive Block Transfer																									
MOVBK	dst-block, src-block	When W = 0 (IY) ← (IX) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX − 1, IY ← IY − 1 When W = 1 (IY + 1, IY) ← (IX + 1, IX) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX − 2, IY ← IY − 2	1	0	1	0	0	1	0	W									1						
CMPBK	src-block, dst-block	When W = 0 (IX) − (IY) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX − 1, IY ← IY − 1 When W = 1 (IX + 1, IX) − (IY + 1, IY) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX − 2, IY ← IY − 2	1	0	1	0	0	1	1	W									1	x	x	x	x	x	x
CMPM	dst-block	When W = 0 AL − (IY) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY − 1 When W = 1 AW − (IY + 1, IY) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY − 2	1	0	1	0	1	1	1	W									1	x	x	x	x	x	x
LDM	src-block	When W = 0 AL ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX − 1 When W = 1 AW ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX − 2	1	0	1	0	1	1	0	W									1						
STM	dst-block	When W = 0 (IY) ← AL DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY − 1 When W = 1 (IY + 1, IY) ← AW DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY − 2	1	0	1	0	1	0	1	W									1						
Bit Field Transfer																									
INS	reg8, reg8	16-Bit field ← AW	0 1	0 1	0	0 reg	1	1	1 reg	1	0	0	1	1	0	0	0	1	3						
	reg8, imm4	16-Bit field ← AW	0 1	0 1	0	0 0	1 0	1	1 reg	1	0	0	1	1	1	0	0	1	4						

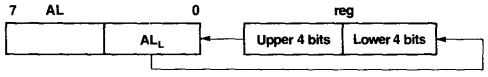
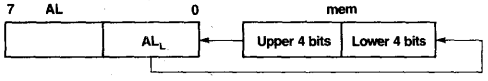
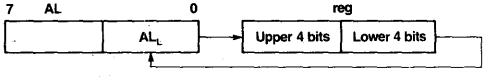
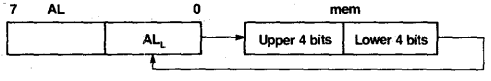
Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	Operation Code 7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	Flags CY	Flags V	Flags P	Flags S	Flags Z
Bit Field Transfer (cont)											
EXT	reg8, reg8	AW ← 16-Bit field	0 0 0 0 1 1 1 1 1 1 reg reg	0 0 1 1 0 0 1 1	3						
	reg8, imm4	AW ← 16-Bit field	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 1 1 0 1 1	4						
I/O											
IN	acc, imm8	When W = 0 AL ← (imm8) When W = 1 AH ← (imm8 + 1), AL ← (imm8)	1 1 1 0 0 1 0 W		2						
	acc, DW	When W = 0 AL ← (DW) When W = 1 AH ← (DW + 1), AL ← (DW)	1 1 1 0 1 1 0 W		1						
OUT	imm8, acc	When W = 0 (imm8) ← AL When W = 1 (imm8 + 1) ← AH, (imm8) ← AL	1 1 1 0 0 1 1 W		2						
	DW, acc	When W = 0 (DW) ← AL When W = 1 (DW + 1) ← AH, (DW) ← AL	1 1 1 0 1 1 1 W		1						
Primitive Block I/O Transfer											
INM	dst-block, DW	When W = 0 (IY) ← (DW) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY − 1 When W = 1 (IY + 1, IY) ← (DW + 1, DW) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY − 2	0 1 1 0 1 1 0 W		1						
OUTM	DW, src-block	When W = 0 (DW) ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX − 1 When W = 1 (DW + 1, DW) ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX − 2	0 1 1 0 1 1 1 W	n: number of transfers	1						
Addition/Subtraction											
ADD	reg, reg	reg ← reg + reg	0 0 0 0 0 0 1 W	1 1 reg reg	2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) + reg	0 0 0 0 0 0 0 W	mod reg mem	2-4	x	x	x	x	x	x
	reg, mem	reg ← reg + (mem)	0 0 0 0 0 0 1 W	mod reg mem	2-4	x	x	x	x	x	x
	reg, imm	reg ← reg + imm	1 0 0 0 0 0 S W	1 1 0 0 0 reg	3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) + imm	1 0 0 0 0 0 S W	mod 0 0 0 mem	3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL + imm When W = 1 AW ← AW + imm	0 0 0 0 0 1 0 W		2-3	x	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Addition/Subtraction (cont)																									
ADDC	reg, reg	reg ← reg + reg + CY	0	0	0	1	0	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) + reg + CY	0	0	0	1	0	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ← reg + (mem) + CY	0	0	0	1	0	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, imm	reg ← reg + imm + CY	1	0	0	0	0	0	S	W	1	1	0	1	0		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) + imm + CY	1	0	0	0	0	0	S	W	mod		0	1	0		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL + imm + CY When W = 1 AW ← AW + imm + CY	0	0	0	1	0	1	0	W									2-3	x	x	x	x	x	x
SUB	reg, reg	reg ← reg − reg	0	0	1	0	1	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) − reg	0	0	1	0	1	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ← reg − (mem)	0	0	1	0	1	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, imm	reg ← reg − imm	1	0	0	0	0	0	S	W	1	1	1	0	1		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) − imm	1	0	0	0	0	0	S	W	mod		1	0	1		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL − imm When W = 1 AW ← AW − imm	0	0	1	0	1	1	0	W									2-3	x	x	x	x	x	x
SUBC	reg, reg	reg ← reg − reg − CY	0	0	0	1	1	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) − reg − CY	0	0	0	1	1	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ← reg − (mem) − CY	0	0	0	1	1	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	
	reg, imm	reg ← reg − imm − CY	1	0	0	0	0	0	S	W	1	1	0	1	1		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) − imm − CY	1	0	0	0	0	0	S	W	mod		0	1	1		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL − imm − CY When W = 1 AW ← AW − imm − CY	0	0	0	1	1	1	0	W									2-3	x	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
BCD Operation											
ADD4S		dst BCD string ← dst BCD string + src BCD string	0 0 0 0 1 1 1 1	0 0 1 0 0 0 0 0	2	u	x	u	u	u	x
SUB4S		dst BCD string ← dst BCD string − src BCD string	0 0 0 0 1 1 1 1	0 0 1 0 0 0 1 0	2	u	x	u	u	u	x
CMP4S		dst BCD string − src BCD string	0 0 0 0 1 1 1 1	0 0 1 0 0 1 1 0	2	u	x	u	u	u	x
ROL4	reg8		0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 0 0	3						
	mem8		0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 0 0	3-5						
ROR4	reg8		0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 1 0	3						
	mem8		0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 1 0	3-5						
BCD Adjust											
ADJBA		When (AL AND 0FH) >9 or AC = 1, AL ← AL + 6, AH ← AH + 1, AC ← 1, CY ← AC, AL ← AL AND 0FH	0 0 1 1 0 1 1 1		1	x	x	u	u	u	u
ADJ4A		When (AL AND 0FH) >9 or AC = 1, AL ← AL + 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY — = AL ← AL + 60H, CY ← 1	0 0 1 0 0 1 1 1		1	x	x	u	x	x	x
ADJBS		When (AL AND 0FH) >9 or AC = 1, CY ← AC, AL ← AL AND 0FH	0 0 1 1 1 1 1 1		1	x	x	u	u	u	u
ADJ4S		When (AL AND 0FH) >9 or AC = 1, AL ← AL − 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY — = AL ← AL + 60H, CY ← 1	0 0 1 0 1 1 1 1		1	x	x	u	x	x	x

NEC

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Increment/Decrement																									
INC	reg8	reg8 ← reg8 + 1	1	1	1	1	1	1	1	0	1	1	0	0	0	reg			2	x		x	x	x	x
	mem	(mem) ← (mem) + 1	1	1	1	1	1	1	1	W	mod		0	0	0	mem			2-4	x		x	x	x	x
	reg16	reg16 ← reg16 + 1	0	1	0	0	0	reg											1	x		x	x	x	x
DEC	reg8	reg8 ← reg8 − 1	1	1	1	1	1	1	1	0	1	1	0	0	1	reg			2	x		x	x	x	x
	mem	(mem) ← (mem) − 1	1	1	1	1	1	1	1	W	mod		0	0	1	mem			2-4	x		x	x	x	x
	reg16	reg16 ← reg16 − 1	0	1	0	0	1	reg											1	x		x	x	x	x
Multiplication																									
MULU	reg8	AW ← AL x reg8 AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	0	reg			2	u	x	x	u	u	u
	mem8	AW ← AL x (mem8) AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod		1	0	0	mem			2-4	u	x	x	u	u	u
	reg16	DW, AW ← AW x reg16 DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	0	reg			2	u	x	x	u	u	u
	mem16	DW, AW ← AW x (mem16) DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod		1	0	0	mem			2-4	u	x	x	u	u	u
MUL	reg8	AW ← AL x reg8 AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	1	reg			2	u	x	x	u	u	u
	mem8	AW ← AL x (mem8) AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod		1	0	1	mem			2-4	u	x	x	u	u	u
	reg16	DW, AW ← AW x reg16 DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	1	reg			2	u	x	x	u	u	u
	mem16	DW, AW ← AW x (mem16) DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod		1	0	1	mem			2-4	u	x	x	u	u	u
	reg16, reg16, imm8	reg16 ← reg16 x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	1	1	reg			reg			3	u	x	x	u	u	u
	reg16, mem16, imm8	reg16 ← (mem16) x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	mod		reg			mem			3-5	u	x	x	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Multiplication (cont)																									
MUL (cont)	reg16, reg16, imm16	reg16 ← reg16 x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	1	1	reg			reg			4	u	x	x	u	u	u
	reg16, mem16, imm16	reg16 ← (mem16) x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	mod		reg			mem			4-6	u	x	x	u	u	u
Unsigned Division																									
DIVU	reg8	temp ← AW When temp ÷ reg8 > FFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg			2	u	u	u	u	u	u
	mem8	temp ← AW When temp ÷ (mem8) > FFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1	1	1	1	0	1	1	0	mod		1	1	0	mem			2-4	u	u	u	u	u	u
	reg16	temp ← AW When temp ÷ reg16 > FFFFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg16, AL ← temp ÷ reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg			2	u	u	u	u	u	u
	mem16	temp ← AW When temp ÷ (mem16) > FFFFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem16), AL ← temp ÷ (mem16)	1	1	1	1	0	1	1	1	mod		1	1	0	mem			2-4	u	u	u	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Signed Division											
DIV	reg8	temp ← AW When temp ÷ reg8 > 0 and temp ÷ reg8 > 7FH or temp ÷ reg8 < 0 and temp ÷ reg8 < 0 - 7FH – 1 (SP – 1, SP – 2) ← PSW, (SP – 3, SP – 4) ← PS (SP – 5, SP – 6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1 1 1 1 0 1 1 0	1 1 1 1 1 reg	2	u	u	u	u	u	u
	mem8	temp W ← When temp ÷ (mem8) > 0 and (mem8) > 7FH or temp ÷ (mem8) < 0 and temp ÷ (mem8) < 0 - 7FH — 1 (SP — 1, SP — 2) ← PSW, (SP — 3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1 1 1 1 0 1 1 0	mod 1 1 1 mem	2-4	u	u	u	u	u	u
	reg 16	temp ← AW When temp ÷ reg 16 > 0 and reg 16 > 7FFFH or temp ÷ reg 16 < 0 - 7FFFH — 1 (SP — 1, SP — 2) ← PSW, (SP — 3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg. 16, AL ← temp ÷ reg 16	1 1 1 1 0 1 1 1	1 1 1 1 1 reg	2	u	u	u	u	u	u
	mem 16	temp ← AW When temp ÷ (mem 16) > 0 and (mem 16) > 7FFFH or temp ÷ (mem 16) < 0 and temp ÷ [**mem 16**] < 0 - 7FFFH — 1 (SP — 1, SP — 2)) ← PSW, (SP —3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem 16), AL ← temp ÷ (mem 16)	1 1 1 1 0 1 1 1	mod 1 1 1 mem	2-4	u	u	u	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Data Conversion																									
CVTBD		AH ← AL ÷ 0AH, AL ← AL % 0AH	1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	2	u	u	u	x	x	x
CVTDB		AH ← 0, AL ← AH x 0AH + AL	1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	2	u	u	u	x	x	x
CVTBW		When AL < 80H, AH ← 0, all other times AH ← FFH	1	0	0	1	1	0	0	0									1						
CVTWL		When AL < 8000H, DW ← 0, all other times DW ← FFFFH	1	0	0	1	1	0	0	1									1						
Comparison																									
CMP	reg, reg	reg − reg	0	0	1	1	1	0	1	W	1	1	reg			reg			2	x	x	x	x	x	x
	mem, reg	(mem) − reg	0	0	1	1	1	0	0	W	mod		reg			mem			2-4	x	x	x	x	x	x
	reg, mem	reg − (mem)	0	0	1	1	1	0	1	W	mod		reg			mem			2-4	x	x	x	x	x	x
	reg, imm	reg − imm	1	0	0	0	0	0	S	W	1	1	1	1	1	reg			3-4	x	x	x	x	x	x
	mem, imm	(mem) − imm	1	0	0	0	0	0	S	W	mod		1	1	1	mem			3-6	x	x	x	x	x	x
	acc, imm	When W = 0, AL − imm When W = 1, AW − imm	0	0	1	1	1	1	0	W									2-3	x	x	x	x	x	x
Complement																									
NOT	reg	reg ← $\overline{reg}$	1	1	1	1	0	1	1	W	1	1	0	1	0	reg			2						
	mem	(mem) ← ($\overline{mem}$)	1	1	1	1	0	1	1	W	mod		0	1	0	mem			2-4						
NEG	reg	reg ← $\overline{reg}$ + 1	1	1	1	1	0	1	1	W	1	1	0	1	1	reg			2	x	x	x	x	x	x
	mem	(mem) ← ($\overline{mem}$) + 1	1	1	1	1	0	1	1	W	mod		0	1	1	mem			2-4	x	x	x	x	x	x
Logical Operation																									
TEST	reg, reg	reg AND reg	1	0	0	0	0	1	0	W	1	1	reg			reg			2	u	0	0	x	x	x
	mem, reg or reg, mem	(mem) AND reg	1	0	0	0	0	1	0	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, imm	reg AND imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg			3-4	u	0	0	x	x	x
	mem, imm	(mem) AND imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL AND imm8 When W = 1, AW AND imm8	1	0	1	0	1	0	0	W									2-3	u	0	0	x	x	x
AND	reg, reg	reg ← reg AND reg	0	0	1	0	0	0	1	W	1	1	reg			reg			2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) AND reg	0	0	1	0	0	0	0	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, mem	reg ← reg AND (mem)	0	0	1	0	0	0	1	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, imm	reg ← reg AND imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg			3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) AND imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem			3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL AND imm8 When W = 1, AW ← AW AND imm16	0	0	1	0	0	1	0	W									2-3	u	0	0	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Logical Operation (cont)																									
OR	reg, reg	reg ← reg OR reg	0	0	0	0	1	0	1	W	1	1		reg			reg		2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) OR reg	0	0	0	0	1	0	0	W	mod			reg			mem		2-4	u	0	0	x	x	x
	reg, mem	reg ← reg OR (mem)	0	0	0	0	1	0	1	W	mod			reg			mem		2-4	u	0	0	x	x	x
	reg, imm	reg ← reg OR imm	1	0	0	0	0	0	0	W	1	1	0	0	1		reg		3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) OR imm	1	0	0	0	0	0	0	W	mod		0	0	1		mem		3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL OR imm8 When W = 1, AW ← AW OR imm16	0	0	0	0	1	1	0	W									2-3	u	0	0	x	x	x
XOR	reg, reg	reg ← reg XOR reg	0	0	1	1	0	0	1	W	1	1		reg			reg		2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) XOR reg	0	0	1	1	0	0	0	W	mod			reg			mem		2-4	u	0	0	x	x	x
	reg, mem	reg ← reg XOR (mem)	0	0	1	1	0	0	1	W	mod			reg			mem		2-4	u	0	0	x	x	x
	reg, imm	reg ← reg XOR imm	1	0	0	0	0	0	0	W	1	1	1	1	0		reg		3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) XOR imm	1	0	0	0	0	0	0	W	mod		1	1	0		mem		3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL XOR imm8 When W = 1, AW ← AW XOR imm16	0	0	1	1	0	1	0	W									2-3	u	0	0	x	x	x
Bit Operation																									
						2nd byte*								3rd byte*											
TEST1	reg8, CL	reg8 bit no. CL = 0: Z ← 1 reg8 bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	0	1	1	0	0	0		reg		3	u	0	0	u	u	x
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	0	mod		0	0	0		mem		3-5	u	0	0	u	u	x
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	1	1	1	0	0	0		reg		3	u	0	0	u	u	x
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	1	mod		0	0	0		mem		3-5	u	0	0	u	u	x
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0	0	0	1	1	0	0	0	1	1	0	0	0		reg		4	u	0	0	u	u	x
	mem8, imm3	(mem8) bit no. imm3 = 0: Z ← 1 (mem8) bit no. imm3 = 1: Z ← 0	0	0	0	1	1	0	0	0	mod		0	0	0		mem		4-6	u	0	0	u	u	x
	reg16, imm4	reg16 bit no. imm4 = 0: Z ← 1 reg16 bit no. imm4 = 1: Z ← 0	0	0	0	1	1	0	0	1	1	1	0	0	0		reg		4	u	0	0	u	u	x
	mem16, imm4	(mem16) bit no. imm4 = 0: Z ← 1 (mem16) bit no. imm4 = 1: Z ← 0	0	0	0	1	1	0	0	1	mod		0	0	0		mem		4-6	u	0	0	u	u	x
						2nd byte*								3rd byte*											

*Note: First byte = 0FH

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)											
			2nd byte*	3rd byte*							
NOT1	reg8, CL	reg8 bit no. CL ← $\overline{\text{reg8 bit no. CL}}$	0 0 0 1 0 1 1 0	1 0 0 0 reg	3						
	mem8, CL	(mem8) bit no. CL ← $\overline{\text{(mem8) bit no. CL}}$	0 0 0 1 0 1 1 0	mod 0 0 0 mem	3-5						
	reg16, CL	reg16 bit no. CL ← $\overline{\text{reg16 bit no. CL}}$	0 0 0 1 0 1 1 1	1 1 0 0 0 reg	3						
	mem16, CL	(mem16) bit no. CL ← $\overline{\text{(mem16) bit no. CL}}$	0 0 0 1 0 1 1 1	mod 0 0 0 mem	3-5						
	reg8, imm3	reg8 bit no. imm3 ← $\overline{\text{reg8 bit no. imm3}}$	0 0 0 1 1 1 1 0	1 1 0 0 0 reg	4						
	mem8, imm3	(mem8) bit no. imm3 ← $\overline{\text{(mem8) bit no. imm3}}$	0 0 0 1 1 1 1 0	mod 0 0 0 mem	4-6						
	reg16, imm4	reg16 bit no. imm4 ← $\overline{\text{(reg16) bit no. imm4}}$	0 0 0 1 1 1 1 1	1 1 0 0 0 reg	4						
	mem16, imm4	(mem16) bit no. imm4 ← $\overline{\text{(mem16) bit no. imm4}}$	0 0 0 1 1 1 1 1	mod 0 0 0 mem	4-6						
			2nd byte* *Note: First byte = 0FH	3rd byte*							
	CY	CY ← $\overline{\text{CY}}$	1 1 1 1 0 1 0 1		1		x				
			2nd byte*	3rd byte*							
CLR1	reg8, CL	reg8 bit no. CL ← 0	0 0 0 1 0 0 1 0	1 1 0 0 0 reg	3						
	mem8, CL	(mem8) bit no. CL ← 0	0 0 0 1 0 0 1 0	mod 0 0 0 mem	3-5						
	reg16, CL	reg16 bit no. CL ← 0	0 0 0 1 0 0 1 1	1 1 0 0 0 reg	3						
	mem16, CL	(mem16) bit no. CL ← 0	0 0 0 1 0 0 1 1	mod 0 0 0 mem	3-5						
	reg8, imm3	reg8 bit no. imm3 ← 0	0 0 0 1 1 0 1 0	1 1 0 0 0 reg	4						
	mem8, imm3	(mem8) bit no. imm3 ← 0	0 0 0 1 1 0 1 0	mod 0 0 0 mem	4-6						
	reg16, imm4	reg16 bit no. imm4 ← 0	0 0 0 1 1 0 1 1	1 1 0 0 0 reg	4						
	mem16, imm4	(mem16) bit no. imm4 ← 0	0 0 0 1 1 0 1 1	mod 0 0 0 mem	4-6						
			2nd byte* *Note: First byte = 0FH	3rd byte*							
	CY	CY ← 0	1 1 1 1 1 0 0 0		1		0				
	DIR	DIR ← 0	1 1 1 1 1 1 0 0		1						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)											
SET1	reg8, CL	reg8 bit no. CL ← 1	0 0 0 1 0 1 0 0	1 1 0 0 0 reg	3						
	mem8, CL	(mem8) bit no. CL ← 1	0 0 0 1 0 1 0 0	mod 0 0 0 mem	3-5						
	reg16, CL	reg16 bit no. CL ← 1	0 0 0 1 0 1 0 1	1 1 0 0 0 reg	3						
	mem16, CL	(mem16) bit no. CL ← 1	0 0 0 1 0 1 0 1	mod 0 0 0 mem	3-5						
	reg8, imm3	reg8 bit no. imm3 ← 1	0 0 0 1 1 1 0 0	1 1 0 0 0 reg	4						
	mem8, imm3	(mem8) bit no. imm3 ← 1	0 0 0 1 1 1 0 0	mod 0 0 0 mem	4-6						
	reg16, imm4	reg16 bit no. imm4 ← 1	0 0 0 1 1 1 0 1	1 1 0 0 0 reg	4						
	mem16, imm4	(mem16) bit no. imm4 ← 1	0 0 0 1 1 1 0 1	mod 0 0 0 mem	4-6						
			2nd byte*	3rd byte*							
			*Note: First byte = 0FH								
	CY	CY ← 1	1 1 1 1 1 0 0 1		1		1				
	DIR	DIR ← 1	1 1 1 1 1 1 0 1		1						
Shift											
SHL	reg, 1	CY ← MSB of reg, reg ← reg x 2 When MSB of reg ≠ CY, V ← 1 When MSB of reg = CY, V ← 0	1 1 0 1 0 0 0 W	1 1 1 0 0 reg	2	u	x	x	x	x	x
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 When MSB of (mem) ≠ CY, V ← 1 When MSB of (mem) = CY, V ← 0	1 1 0 1 0 0 0 W	mod 1 0 0 mem	2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1 1 0 1 0 0 1 W	1 1 1 0 0 reg	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1 1 0 1 0 0 1 W	mod 1 0 0 mem	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1 1 0 0 0 0 0 W	1 1 1 0 0 reg	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1 1 0 0 0 0 0 W	mod 1 0 0 mem	3-5	u	x	u	x	x	x
			n: number of shifts								
SHR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 When MSB of reg ≠ bit following MSB of reg: V ← 1 When MSB of reg = bit following MSB of reg: V ← 0	1 1 0 1 0 0 0 W	1 1 1 0 1 reg	2	u	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Shift (cont)																									
SHR (cont)	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 When MSB of (mem) ≠ bit following MSB of (mem): V ← 1 When MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod		1	0	1	mem			2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	1	0	1	reg			2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1	1	0	1	0	0	1	W	mod		1	0	1	mem			2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	1	0	1	reg			3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 n: number of shifts	1	1	0	0	0	0	0	W	mod		1	0	1	mem			3-5	u	x	u	x	x	x
SHRA	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, V ← 0 MSB of operand does not change	1	1	0	1	0	0	0	W	1	1	1	1	1	reg			2	u	x	0	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, V ← 0, MSB of operand does not change	1	1	0	1	0	0	0	W	mod		1	1	1	mem			2-4	u	x	0	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1	1	0	1	0	0	1	W	1	1	1	1	1	reg			2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1	1	0	1	0	0	1	W	mod		1	1	1	mem			2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1	1	0	0	0	0	0	W	1	1	1	1	1	reg			3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change n: number of shifts	1	1	0	0	0	0	0	W	mod		1	1	1	mem			3-5	u	x	u	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Rotation																									
ROL	reg, 1	CY ← MSB of reg, reg ← reg x 2 + CY MSB of reg ≠ CY: V ← 1 MSB of reg = CY: V ← 0	1	1	0	1	0	0	0	W	1	1	0	0	0	reg			2		x	x			
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 + CY MSB of (mem) ≠ CY: V ← 1 MSB of (mem) = CY: V ← 0	1	1	0	1	0	0	0	W	mod		0	0	0	mem			2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	0	0	0	reg			2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp − 1	1	1	0	1	0	0	1	W	mod		0	0	0	reg			2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	0	0	0	reg			3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp − 1	1	1	0	0	0	0	0	W	mod		0	0	0	mem			3-5		x	u			
								n: number of shifts																	
ROR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← CY MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	0	0	1	reg			2		x	x			
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← CY MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod		0	0	1	mem			2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	0	0	1	reg			2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY temp ← temp − 1	1	1	0	1	0	0	1	W	mod		0	0	1	mem			2-4		x	u			
								n:number of shifts																	

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	AC	Flags CY	V	P	S	Z
Rotation (cont)											
ROR (cont)	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp − 1	1 1 0 0 0 0 0 W	1 1 0 0 1 reg	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 temp ← temp − 1	1 1 0 0 0 0 0 W	mod 0 0 1 mem n: number of shifts	3-5		x	u			
Rotate											
ROLC	reg, 1	tmpcy ← CY, CY ← MSB of reg reg ← reg x 2 + tmpcy MSB of reg = CY: V ← 0 MSB of reg ≠ CY: V ← 1	1 1 0 1 0 0 0 W	1 1 0 1 0 reg	2		x	x			
	mem, 1	tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy MSB of (mem) = CY: V ← 0 MSB of (mem) ≠ CY: V ← 1	1 1 0 1 0 0 0 W	mod 0 1 0 mem	2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp − 1	1 1 0 1 0 0 1 W	1 1 0 1 0 reg	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy temp ← temp − 1	1 1 0 1 0 0 1 W	mod 0 1 0 mem	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp − 1	1 1 0 0 0 0 0 W	1 1 0 1 0 reg	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy temp ← temp − 1	1 1 0 0 0 0 0 W	mod 0 1 0 mem n: number of shifts	3-5		x	u			

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Rotate (cont)																									
RORC	reg, 1	tmpcy ← CY, CY ← LSB of reg reg ← reg ÷ 2, MSB of reg ← tmpcy MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	0	1	1	reg			2		x	x			
	mem, 1	tmpcy ← CY, CY ← LSB of (mem) (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod		0	1	1	mem			2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	0	1	1	reg			2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	mod		0	1	1	mem			2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0 repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	0	1	1	reg			3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	mod		0	1	1	mem			3-5		x	u			
Subroutine Control Transfer																									
CALL	near-proc	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← PC + disp	1	1	1	0	1	0	0	0									3						
	regptr16	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← regptr16	1	1	1	1	1	1	1	1	1	1	0	1	0	reg			2						
	memptr16	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← (memptr16)	1	1	1	1	1	1	1	1	mod		0	1	0	mem			2-4						
	far-proc	(SP − 1, SP − 2) ← PS, (SP − 3, SP − 4) ← PC SP ← SP − 4, PS ← seg, PC ← offset	1	0	0	1	1	0	1	0									5						
	memptr32	(SP − 1, SP − 2) ← PS, (SP − 3, SP − 4) ← PC SP ← SP − 4, PS ← (memptr32 + 2), PC ← (memptr32)	1	1	1	1	1	1	1	1	mod		0	1	1	mem			2-4						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Subroutine Control Transfer (cont)										
RET		PC ← (SP + 1, SP), SP ← SP + 2	1 1 0 0 0 0 1 1	1						
	pop-value	PC ← (SP + 1, SP) SP ← SP + 2, SP ← SP + pop-value	1 1 0 0 0 0 1 0	3						
		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2) SP ← SP + 4	1 1 0 0 1 0 1 1	1						
	pop-value	PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2) SP ← SP + 4, SP ← SP + pop-value	1 1 0 0 1 0 1 0	3						
Stack Manipulation										
PUSH	mem16	(SP − 1, SP − 2) ← (mem16), SP ← SP − 2	1 1 1 1 1 1 1 1 mod 1 1 0 mem	2-4						
	reg16	(SP − 1, SP − 2) ← reg16, SP ← SP − 2	0 1 0 1 0 reg	1						
	sreg	(SP− 1, SP− 2) ← sreg, SP ← SP − 2	0 0 0 sreg 1 1 0	1						
	PSW	(SP − 1, SP − 2) ← PSW, SP ← SP − 2	1 0 0 1 1 1 0 0	1						
	R	Push registers on the stack	0 1 1 0 0 0 0 0	1						
	imm	(SP − 1, SP − 2) ← imm SP ← SP − 2, When S = 1, sign extension	0 1 1 0 1 0 S 0	2-3						
POP	mem16	(mem16) ← (SP + 1, SP), SP ← SP + 2	1 0 0 0 1 1 1 1 mod 0 0 0 mem	2-4						
	reg16	reg16 ← (SP + 1, SP), SP ← SP + 2	0 1 0 1 1 reg	1						
	sreg	sreg ← (SP + 1, SP) sreg : SS, DS0, DS1 SP ← SP + 2	0 0 0 sreg 1 1 1	1						
	PSW	PSW ← (SP + 1, SP), SP ← SP + 2	1 0 0 1 1 1 0 1	1	R	R	R	R	R	R
	R	Pop registers from the stack	0 1 1 0 0 0 0 1	1						
PREPARE	imm16, imm8	Prepare new stack frame	1 1 0 0 1 0 0 0 *: imm8 = 0: 16 imm8 > 1: 25 + 16 (imm8 − 1)	4						
DISPOSE		Dispose of stack frame	1 1 0 0 1 0 0 1	1						
Branch										
BR	near-label	PC ← PC + disp	1 1 1 0 1 0 0 1	3						
	short-label	PC ← PC + ext-disp8	1 1 1 0 1 0 1 1	2						
	regptr16	PC ← regptr16	1 1 1 1 1 1 1 1 1 1 1 0 0 reg	2						
	memptr16	PC ← (memptr16)	1 1 1 1 1 1 1 1 mod 1 0 0 mem	2-4						
	far-label	PS ← seg, PC ← offset	1 1 1 0 1 0 1 0	5						
	memptr32	PS ← (memptr32 + 2), PC ← (memptr32)	1 1 1 1 1 1 1 1 mod 1 0 1 mem	2-4						

NEC

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Conditional Branch																									
BV	short-label	if V = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	0	0									2						
BNV	short-label	if V = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	0	1									2						
BC, BL	short-label	if CY = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	1	0									2						
BNC, BNL	short-label	if CY = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	1	1									2						
BE, BZ	short-label	if Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	0	0									2						
BNE, BNZ	short-label	if Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	0	1									2						
BNH	short-label	if CY OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	1	0									2						
BH	short-label	if CY OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	1	1									2						
BN	short-label	if S = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	0	0									2						
BP	short-label	if S = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	0	1									2						
BPE	short-label	if P = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	1	0									2						
BPO	short-label	if P = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	1	1									2						
BLT	short-label	if S XOR V = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	0	0									2						
BGE	short-label	if S XOR V = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	0	1									2						
BLE	short-label	if (S XOR V) OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	1	0									2						
BGT	short-label	if (S XOR V) OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	1	1									2						
DBNZNE	short-label	CW ← CW − 1 if Z = 0 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	0									2						
DBNZE	short-label	CW ← CW − 1 if Z = 1 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	1									2						
DBNZ	short-label	CW ← CW − 1 if CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	1	0									2						
BCWZ	short-label	if CW = 0, PC ← PC + ext-disp8	1	1	1	0	0	0	1	1									2						
BTCLR	sfr. imm3, short-label	if bit no. imm3 of (sfr) = 1, PC ← PC + ext − disp8, bit no. imm3 of (sfr) ← 0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	5						
Interrupt																									
BRK	3	(SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PS ← (15, 14), PC ← (13, 12)	1	1	0	0	1	1	0	0									1						
	imm8 (≠ 3)	(SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PC ← (n x 4, + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1	1	0	0	1	1	0	1									2						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Interrupt (cont)										
BRKV		When V = 1 (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PS ← (19, 18), PC ← (17, 16)	1 1 0 0 1 1 1 0	1						
RETI		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1 1 0 0 1 1 1 1	1	R	R	R	R	R	R
RETRBI		PC ← Save PC, PSW ← Save PSW	0 0 0 0 1 1 1 1 1 0 0 1 0 0 0 1	2	R	R	R	R	R	R
FINT		Indicates that interrupt service routine to the interrupt controller built in the CPU has been completed	0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0	2						
CHKIND	reg16, mem32	When (mem32) > reg16 or (mem32 + 2) < reg16 (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (23, 22), PC ← (21, 20)	0 1 1 0 0 0 1 0 mod reg mem	2-4						
CPU Control										
HALT		CPU Halt	1 1 1 1 0 1 0 0		1					
STOP		CPU Halt	0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0		1					
BUSLOCK		Bus Lock Prefix	1 1 1 1 0 0 0 0		1					
FPO1 (Note 1)	fp-op	No Operation	1 1 0 1 1 X X X 1 1 Y Y Y Z Z Z	2						
	fp-op, mem	data bus ← (mem)	1 1 0 1 1 X X X mod Y Y Y mem	2-4						
FPO2 (Note 1)	fp-op	No Operation	0 1 1 0 0 1 1 X 1 1 Y Y Y Z Z Z	2						
	fp-op, mem	data bus ← (mem)	0 1 1 0 0 1 1 X mod Y Y Y mem	2-4						
POLL		Poll and wait	1 0 0 1 1 0 1 1	1						
NOP		No Operation	1 0 0 1 0 0 0 0	1						
DI		IE ← 0	1 1 1 1 1 0 1 0	1						
EI		IE ← 1	1 1 1 1 1 0 1 1	1						
DS0; DS1; PS; SS		Segment override prefix	0 0 1 sreg 1 1 0	1						

Notes:

(1) Does not execute on the V25, but does generate an interrupt.

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Register Bank Switching																									
MOVSPA			0	0	0	0	1	1	1	1	0	0	1	0	0	1	0	1	2						
BRKCS	reg16		0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	1	3						
MOVSPB	reg16		0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	1	3						
			1	1	1	1	1		reg																
TSKSW	reg16		0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	0	3	x	x	x	x	x	x
			1	1	1	1	1		reg																

NEC
NEC Electronics Inc.

μPD70330/70332 (V35) 16-Bit Microcomputers: Advanced, Single-Chip, CMOS

Description

The μPD70330/70332 (V35™) is a high-performance, 16-bit single-chip microcomputer with a 16-bit external data bus. The μPD70330/70332 is fully software compatible with μPD8086/8088 and μPD70108/70116 (V20®/30®) instruction set.

The μPD70330 is a ROMless part. The μPD70332 has 16K ROM, while the μPD70P322 has 16K EPROM and can be used as a μPD70330 (V35) or a μPD70320 (V25™).

Features

- ☐ Functionally compatible with μPD70320/322 (V25)
- ☐ Internal 16-bit architecture and external 16-bit data bus
- ☐ Software compatible with μPD8086/8088, μPD70108/70116 (V20/30) in the native mode
- ☐ New and enhanced instructions
- ☐ Six-byte prefetch queue
- ☐ Minimum instruction cycle: 500 ns at 8 MHz
- ☐ Internal memory
  - — ROM: 16K bytes (μPD70332 only)
  - — RAM: 256 bytes
- ☐ Memory space: 1M bytes
- ☐ Input port with comparator (port T): eight bits
- ☐ Bus interface optimized for use with dynamic RAMs
  - — Multiplexed address
  - — On-board refresh controller
- ☐ 24 parallel I/O lines
- ☐ Serial interface: two channels
  - — Dedicated baud rate generator
  - — Asynchronous mode, I/O interface mode
- ☐ Interrupt controller
  - — Programmable priority (eight levels)
  - — Three interrupt service functions
  - — Vectored interrupt, register bank switching, macro service
- ☐ DRAM, pseudo SRAM refresh function
- ☐ Two DMA channels
- ☐ Two 16-bit timers
- ☐ One 20-bit time base counter
- ☐ Clock generator
- ☐ Programmable wait function
- ☐ Low power modes
  - — HALT
  - — STOP
- ☐ 1.2-micron CMOS

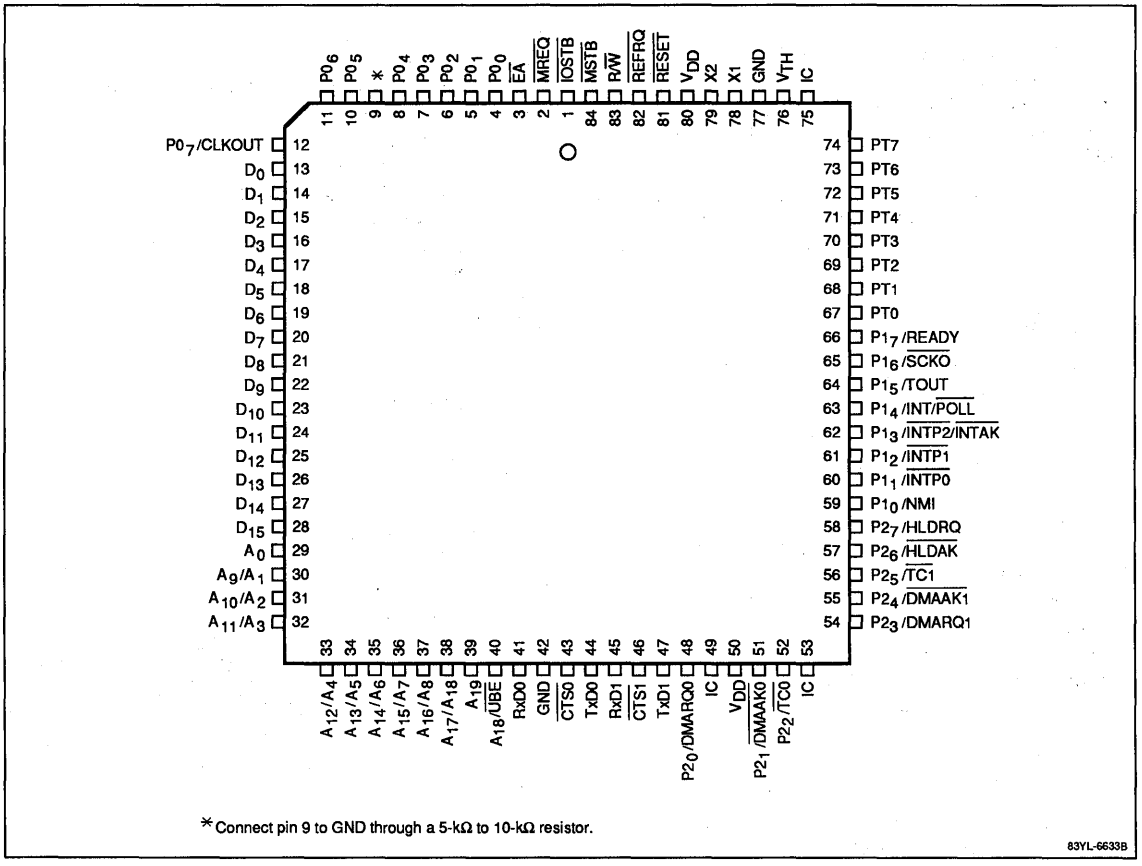
V20 and V30 are registered trademarks of NEC Corporation.
V25 and V35 are trademarks of NEC Corporation.

Ordering Information

Part Number	Clock (MHz)	Package	Internal ROM
μPD70330L-8	8	84-pin PLCC	ROMless
GJ-8	8	94-pin plastic QFP	
μPD70332L-8-xxx	8	84-pin PLCC	16K mask ROM
GJ-8-xxx	8	94-pin plastic QFP	
μPD70P322KE-8	8	84-pin LCC	16K EPROM (UV erasable)

Pin Configuration

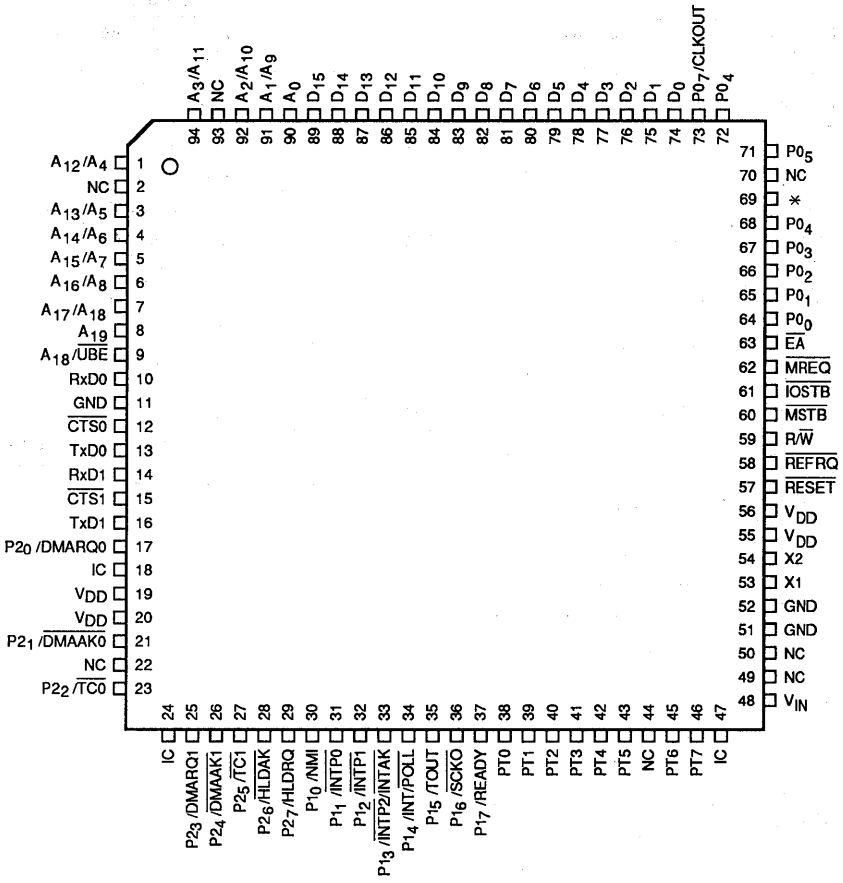
84-Pin PLCC and 84-Pin LCC



* Connect pin 9 to GND through a 5-kΩ to 10-kΩ resistor.

Pin Configuration (cont)

94-Pin Plastic QFP



*Connect pin 69 to GND through a 5-kΩ to 10-kΩ resistor.

83YL-6634B

4b

Pin Identification

Symbol	Function
A_{19}-A_0	Address bus outputs
CLKOUT	System clock output
$\overline{CTS0}$	Clear-to-send input, serial channel 0
$\overline{CTS1}$	Clear-to-send input, serial channel 1
D_{15}-D_0	Bidirectional data bus
$\overline{DMAAK0}$	DMA acknowledge output, DMA controller channel 0
$\overline{DMAAK1}$	DMA acknowledge output, DMA controller channel 1
DMARQ0	DMA request input, DMA controller channel 0
DMARQ1	DMA request input, DMA controller channel 1
$\overline{EA}$	External access; clamped low or high according to program access requirements
$\overline{HLDAK}$	Hold acknowledge output
HLDRQ	Hold request input
INT	Interrupt request input
$\overline{INTAK}$	Interrupt acknowledge output
$\overline{INTP0}$	Interrupt request 0 input
$\overline{INTP1}$	Interrupt request 1 input
$\overline{INTP2}$	Interrupt request 2 input
$\overline{IOSTB}$	I/O read or write strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory strobe output
NMI	Nonmaskable interrupt request
$\overline{POLL}$	Input on POLL synchronizes the CPU and external devices
$P0_7$-$P0_0$	I/O port 0
$P1_7$-$P1_0$	I/O port 1
$P2_7$-$P2_0$	I/O port 2
PT0-PT7	Comparator port input lines
READY	Ready signal input controls insertion of wait states
$\overline{REFRQ}$	DRAM refresh request output
$\overline{RESET}$	Reset signal input
R/$\overline{W}$	Read/write strobe output
RxD0	Receive data input, serial channel 0

Symbol	Function
RxD1	Receive data input, serial channel 1
$\overline{SCK0}$	Serial clock output
$\overline{TC0}$	Terminal count output; DMA completion, channel 0
$\overline{TC1}$	Terminal count output; DMA completion, channel 1
TOUT	Timer output
TxD0	Transmit data output, serial channel 0
TxD1	Transmit data output, serial channel 1
$\overline{UBE}$	Upper byte enable
X1, X2	Connections to external frequency control source (crystal, ceramic resonator, or clock)
V_{DD}	+5-volt power source input (two pins)
V_{TH}	Threshold voltage input to comparator circuits
GND	Ground reference (two pins)
IC	Internal connection; must be tied to V_{DD} externally through a pullup resistor

Pin Functions

A_{19}-A_0; Address Bus

To support dynamic RAMs, the 20-bit address is multiplexed on 11 lines. When $\overline{MREQ}$ is asserted, A_{17}-A_9 are valid. When $\overline{MSTB}$ or $\overline{IOSTB}$ are asserted, A_8-A_1 and A_{18} are valid. A_{18} is also multiplexed with $\overline{UBE}$ and is valid when $\overline{MREQ}$ is asserted. Therefore A_{18} is active throughout the bus cycle. A_{19} and A_0 are not multiplexed but have dedicated pins and are valid throughout the bus cycle.

CLKOUT; Clock Out

The system clock (CLK) is distributed from the internal clock generator to the CPU and output to peripheral hardware at the CLKOUT pin.

$\overline{CTS0}$; Clear-to-Send 0

This is the CTS pin of the channel 0 serial interface. In asynchronous mode, a low-level input on $\overline{CTS0}$ enables transmit operation. In I/O interface mode, $\overline{CTS0}$ is the receive clock pin.

$\overline{CTS1}$; Clear-to-Send 1

This is the CTS pin of the channel 1 serial interface. In asynchronous mode, a low-level input on $\overline{CTS1}$ enables transmit operation.

D_{15}-D_0; Data Bus

D_{15}-D_0 is the 16-bit data bus.

$\overline{DMAAK0}$ and $\overline{DMAAK1}$; DMA Acknowledge

These are the DMA acknowledge outputs of the DMA controller, channels 0 and 1. Signals are not output during DMA memory-to-memory transfer operations (burst mode, single-step mode).

DMARQ0 and DMARQ1; DMA Request

These are the DMA request inputs of the DMA controller, channels 0 and 1.

$\overline{EA}$; External Access

For the ROM-less μPD70330, connect this pin to ground. For the μPD70332, connect $\overline{EA}$ to ground if program code is in external memory; connect $\overline{EA}$ to +5 volts if program code is in the internal ROM.

$\overline{HLDAK}$; Hold Acknowledge

The $\overline{HLDAK}$ output signal indicates that the hold request (HLDRQ) has been accepted. When $\overline{HLDAK}$ is active (low), the following lines go to the high-impedance state with internal 4700-ohm pullup resistors: A_{19}-A_0, D_7-D_0, $\overline{IOSTB}$, $\overline{MREQ}$, $\overline{MSTB}$, $\overline{REFRQ}$, and R/$\overline{W}$.

HLDRQ; Hold Request

The HLDRQ input from an external device requests that the μPD70330/332 relinquish the address, data, and control buses to an external bus master.

INT; Interrupt

The INT input is a vectored interrupt request from an external device that can be masked by software. The active high level is detected in the last clock cycle of an instruction. The external device confirms that the INT interrupt request has been accepted by the $\overline{INTAK}$ signal output from the CPU.

The INT signal must be held high until the first INTAK signal is output. Together with $\overline{INTAK}$, INT is used for operation with an interrupt controller such as μPD71059.

$\overline{INTAK}$; Interrupt Acknowledge

The $\overline{INTAK}$ output is the acknowledge signal for the software-maskable interrupt request INT. The $\overline{INTAK}$ signal goes low when the CPU accepts INT. The external device inputs the interrupt vector to the CPU via data bus D_7-D_0 in synchronization with $\overline{INTAK}$.

$\overline{INTP0}$, $\overline{INTP1}$, $\overline{INTP2}$; Interrupt from Peripheral 0, 1, 2

The $\overline{INTPn}$ inputs (n = 0, 1, 2) are external interrupt requests that can be masked by software. The $\overline{INTPn}$ input is detected at the effective edge specified by external interrupt mode register INTM.

The $\overline{INTPn}$ input is also used to release the HALT mode.

$\overline{IOSTB}$; I/O Strobe

A low-level output on $\overline{IOSTB}$ indicates that the I/O bus cycle has been initiated and that the I/O address output on A_{15}-A_0 is valid.

$\overline{MREQ}$; Memory Request

A low-level output on $\overline{MREQ}$ indicates that the memory or I/O bus cycle has started and that address bits A_0, A_{17}-A_9, A_{19} and A_{18} are valid.

$\overline{MSTB}$; Memory Strobe

Together with $\overline{MREQ}$ and R/$\overline{W}$, $\overline{MSTB}$ controls memory accessing operations. $\overline{MSTB}$ should be used either to enable data buffers or as a data strobe. During memory write, a low-level output on $\overline{MSTB}$ indicates that data on the data bus is valid. A low-level output on $\overline{MSTB}$ indicates that multiplexed address bits A_8-A_1, A_{18}, and $\overline{UBE}$ are valid.

NMI; Nonmaskable Interrupt

The NMI input is an interrupt request that cannot be masked by software. The NMI is always accepted by the CPU; therefore, it has priority over any other interrupt.

The NMI input is detected at the effective edge specified by external interrupt mode register INTM. Sampled in each clock cycle, NMI is accepted when the active level lasts for some clock cycles. When the NMI is accepted, a number 2 vector interrupt is generated after completion of the instruction currently being executed.

The NMI input is also used to release the CPU standby mode.

$P0_7$-$P0_0$; Port 0

Port 0 is an 8-bit bidirectional I/O port.

$P1_7$-$P1_0$; Port 1

Lines $P1_7$-$P1_4$ are individually programmable as an input, output, or control function. The status of $P1_3$-$P1_0$ can be read but these lines are always control functions.

$P2_7$-$P2_0$; Port 2

$P2_7$-$P2_0$ are the lines of port 2, an 8-bit bidirectional I/O port. These lines can also be used as control signals for the on-chip DMA controllers. See table 2-3.

$\overline{POLL}$; Poll

The $\overline{POLL}$ input is checked by the POLL instruction. If the level is low, execution of the next instruction is initiated. If the level is high, the $\overline{POLL}$ input is checked every five clock cycles until the level becomes low.

The POLL functions are used to synchronize the CPU program and the operation of external devices.

Note: $\overline{POLL}$ is effective when $P1_4$ is specified for the input port mode; otherwise, $\overline{POLL}$ is assumed to be at low level when the POLL instruction is executed.

PT0-PT7; Port with Comparator

The PT input is compared with a threshold voltage that is programmable to one of 16 voltage steps individually for each of the eight lines.

READY

After READY is de-asserted low, the CPU will synchronize and insert at least two wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than normal execution allows.

$\overline{REFRQ}$; Refresh Request

This output pulse can refresh nonstatic RAM. It can be programmed to meet system specifications and is internally synchronized so that refresh cycles do not interfere with normal CPU operation.

$\overline{RESET}$

This input signal is asynchronous. A low on RESET for a certain duration resets the CPU and all on-chip peripherals regardless of clock operation. The reset operation has priority over all other operations.

The reset signal is used for normal initialization/startup and also for releasing the STOP or HALT mode. After the reset signal returns high, program execution begins from address FFFF0H.

R/$\overline{W}$; Read/Write Strobe

When the memory bus cycle is initiated, the R/$\overline{W}$ signal output to external hardware indicates a read (high level) or write (low level) cycle. It can also control the direction of bidirectional buffers.

RxD0, RxD1; Receive Data 0, 1

These pins input data from serial channels 0 and 1.

In the asynchronous mode, when receive operation is enabled, a low level on the RxD0 or RxD1 input pin is recognized as the start bit and receive operation is initiated.

In the I/O interface mode (channel 0 only), receive data is input to the serial register at the rising edge of the receive clock.

$\overline{SCKO}$; Serial Clock

The $\overline{SCKO}$ output is the transmit clock of serial channel 0.

$\overline{TC0}$, $\overline{TC1}$; Terminal Count 0, 1

The $\overline{TC0}$ and $\overline{TC1}$ outputs go low when the terminal count of DMA service channels 0 and 1, respectively, reach zero, indicating DMA completion.

TOUT; Timer Output

The TOUT signal is a square-wave output from the internal timer.

TxD0, TxD1; Transmit Data 0, 1

These pins output data from serial channels 0 and 1.

In the asynchronous mode, the transmit signal is in a frame format that consists of a start bit, 7 or 8 data bits (least significant bit first), parity bit, and stop bit. The TxD0 and TxD1 pins become mark state (high level) when transmit operation is disabled or when the serial register has no transmit data.

In the I/O interface mode (channel 0 only), the frame has 8 data bits and the most significant bit is transmitted first.

X1, X2; Clock Control

The frequency of the internal clock generator is controlled by an external crystal or ceramic resonator connected across pins X1 and X2. The crystal frequency is the same as the clock generator frequency f_X. By programming the PRC register, the system clock frequency f_{CLK} is selected as f_X divided by 2, 4, or 8.

As an alternative to the crystal or ceramic resonator, the positive and negative phases of an external clock (with frequency f_X) can be connected to pins X1 and X2.

V_{DD}

+5-volt power source (two pins).

V_{TH}

Comparator port PT0-PT7 uses threshold voltage V_{TH} to determine the analog reference points. The actual threshold to each comparator line is programmable to V_{TH} x n/16 where n = 1 to 16.

GND

Ground reference (two pins).

IC

Internal connection; must be tied to V_{DD} externally through a 10-kΩ to 20-kΩ resistor.

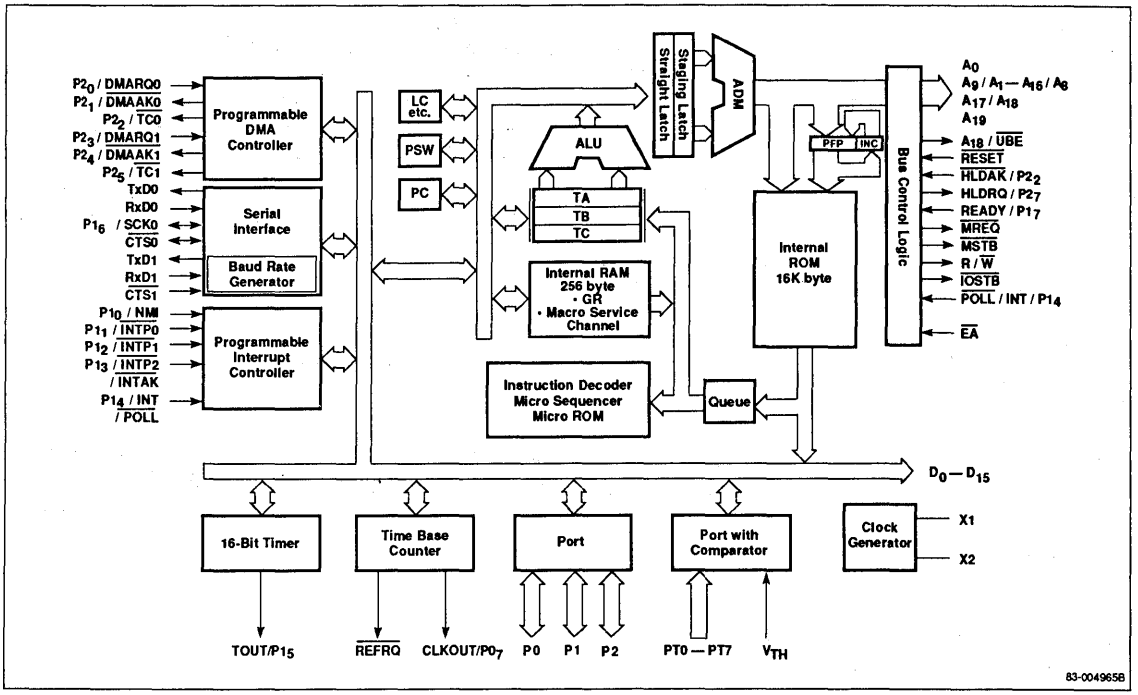
$\overline{UBE}$, Upper Byte Enable

$\overline{UBE}$ is a high-order memory bank selection signal output. $\overline{UBE}$ and A_0 are used to decide which bytes of the data bus will be used. $\overline{UBE}$ is used along with A_0 to select the even/odd banks as follows.

Operand	$\overline{UBE}$	A_0	Number of bus cycles
Even address word	0	0	1
Odd address word	0	1	2
	1	0	
Even address byte	1	0	1
Odd address byte	0	1	1

4b

Block Diagram



Functional Description

Architectural Enhancements

The following features enable the μPD70330/332 to perform high-speed execution of instructions:

- Dual data bus
- 16-/32-bit temporary registers/shifters (TA, TB, TA + TB)
- 16-bit loop counter (LC)
- Program counter (PC) and prefetch pointer (PFP)
- Internal ROM pass bus (μPD70332 only)

Dual Data Bus. The μPD70330/332 has two internal 16-bit data buses: the main data bus and a subdata bus. This reduces the processing time required for addition/subtraction and logical comparison instructions by one-third over single-bus systems. The dual data bus method allows two operands to be fetched simultaneously from the general-purpose registers and transferred to the ALU.

16-/32-Bit Temporary Registers/Shifters. The 16-bit temporary registers/shifters (TA, TB) allow high-speed execution of multiplication/division and shift/rotation instructions. By using the temporary registers/shifters, the μPD70330/332 can execute multiplication/division instructions about four times faster than with the microprogramming method.

Loop Counter [LC]. The dedicated hardware loop counter counts the number of loops for string operations and the number of shifts performed for multiple bit shift/rotation instructions. The loop counter works with internal dedicated shifters to speed the processing of multiplication/division instructions.

Program Counter and Prefetch Pointer [PC and PFP]. The hardware PC addresses the memory location of the instruction to be executed next. The hardware PFP addresses the program memory location to be accessed next. Several clocks are saved for branch, call, return, and break instructions compared with processors having only one instruction pointer.

Internal ROM Pass Bus. The μPD70332 features a dedicated data bus between the internal ROM and the instruction pre-fetch queue. This allows internal ROM opcode fetches to be performed in a single clock cycle (200 ns at 5 MHz); it also makes it possible for opcode fetches to be performed while the external data bus is busy. This feature gives the V35 a 10-20% performance increase when executing from the internal ROM.

Register Set

The μPD70330/70332 CPUs have general purpose register sets compatible with the μPD70108/70116 and the μPD70320/70322 microprocessors. Like the μPD70320/70322, they also have a set of special function registers for controlling the onboard peripherals. All registers reside in the CPU's memory space. They are grouped in a 4K byte block called the internal data area (IDA). The 256 byte internal RAM is also in the IDA. The addresses of the register are given as offsets into the IDA. The start address of the IDA is set by the Internal Data Area Base register (IDB), and may be programmed to any 4K boundary in the memory address space.

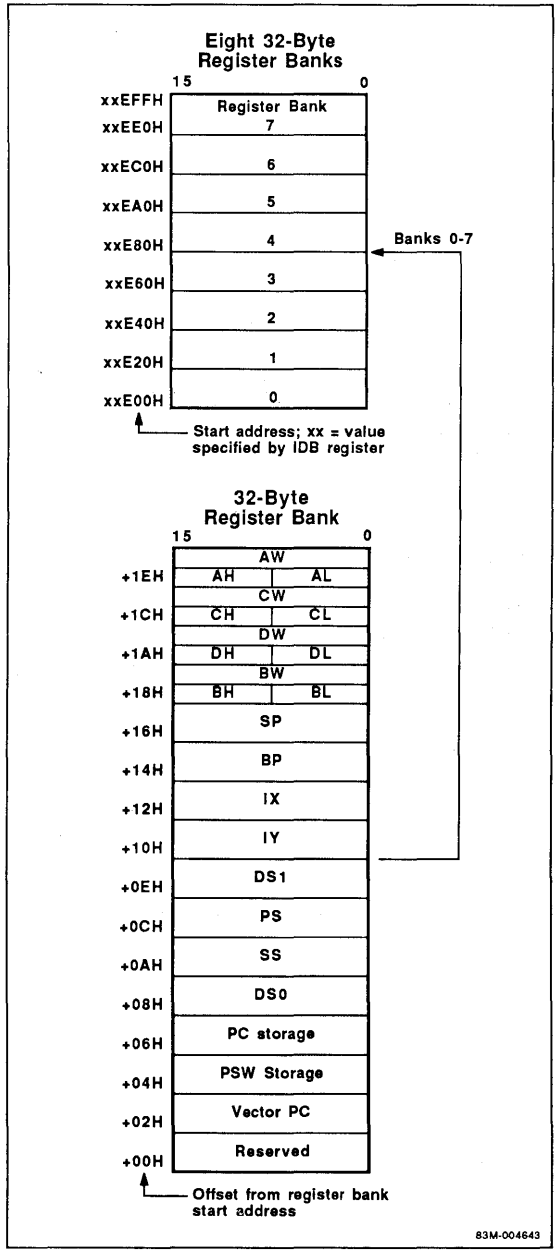
Register Banks. Because the general purpose register set is in internal RAM, it is possible to have multiple banks of registers. The μPD70330/70332 CPU supports up to 8 register banks. A bit field in the PSW selects which bank is currently being used. Each bank contains the entire CPU register set plus additional information needed for context switching. Register banks may be switched using special instructions (TSKSW, BRKCS, MOVSPA, MOVSPB), or may switch in response to an interrupt. This provides fast context switching and fast interrupt handling. During and after RESET, register bank 7 is selected.

Figure 1 shows the configuration of a register bank and how the banks are mapped to internal RAM. The Vector PC field contains the value that will be loaded into the PC when a register bank switch occurs. The PC Save and PSW Save fields contain the values of the PC and the PSW just before the banks are switched. The PSW is left unmodified after a bank switch; the PSW Save field is used to restore the PSW to its previous state is required.

General-Purpose Registers [AW, BW, CW, DW]. These four 16-bit general-purpose registers can also serve as independent 8-bit registers (AH, AL, BH, BL, CH, CL, DH, DL). The instructions below use general-purpose registers for default:

- AW — Word multiplication/division, word I/O, data conversion
- AL — Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
- AH — Byte multiplication/division
- BW — Translation
- CW — Loop control branch, repeat prefix
- CL — Shift instructions, rotation instructions, BCD operations
- DW — Word multiplication/division, indirect addressing I/O

Figure 1. Register Bank Configuration



4b

Pointers [SP, BP] and Index Registers [IX, IY]. These registers are used as 16-bit base pointers or index registers in based addressing, indexed addressing, and based indexed addressing. The registers are used as default registers under the following conditions:

SP	Stack operations
IX	Block transfer (source), BCD string operations
IY	Block transfer (destination), BCD string operations

Segment Registers. The segment registers divide the 1M-byte address space into 64K-byte blocks. Each segment register functions as a base address to a block; the effective address is an offset from that base. Physical addresses are generated by shifting the associated segment register left four binary digits and then adding the effective address. The segment registers are:

Segment Register	Default Offset
PS (Program segment)	PC
SS (Stack segment)	SP, Effective address
DS0 (Data segment-0)	IX, Effective address
DS1 (Data segment-1)	IY, Effective address

During RESET, PS is set to FFFFH; DS0, DS1 and SS are set to 0000H.

Program Counter [PC]. The PC is a 16-bit binary counter that contains the offset address from the program segment of the next instruction to be executed. It is incremented every time an instruction is received from the queue. It is loaded with a new location whenever a branch, call, return, break, or interrupt is executed. During RESET, PC is set to 0000H.

Program Status Word [PSW]. The PSW contains the following status and control flags.

PSW

15							8
1	RB2	RB1	RB0	V	DIR	IE	BRK

7							0
S	Z	F1	AC	F0	P	BRKI	CY

Status Flags

V	Overflow bit
S	Sign
Z	Zero
AC	Auxiliary carry
P	Parity
CY	Carry

Control Flags

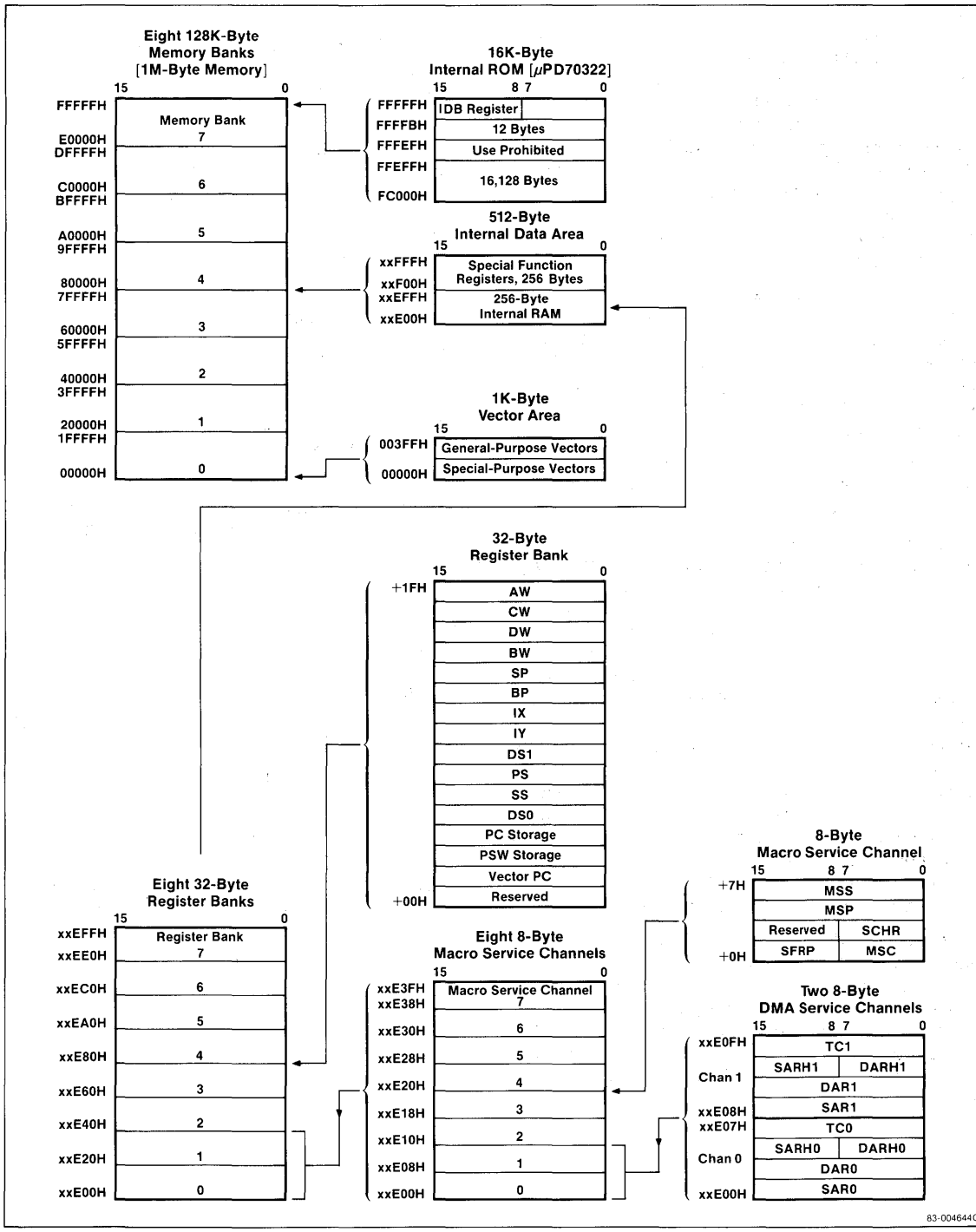
DIR	Direction of string processing
IE	Interrupt enable
BRK	Break (after every instruction)
RBn	Current register bank flags
BRKI	I/O trap enable (see software interrupts)
F0, F1	General-purpose user flags

The eight low-order bits of the PSW can be stored in the A4 register and restored by a MOV instruction execution. The only way to alter the RBn bits via software is to execute an RETRBI or RETI instruction. During RESET, PSW is set to F002H. The F0 and F1 flags may be accessed as bits in the FLAG special functioning register.

Memory Map

The μPD70330/332 has a 20-bit address bus that can directly access 1M bytes of memory. Figure 2 shows that the 16K bytes of internal ROM (μPD70332 only) are located at the top of the address space from FC000H to FFFFFH.

Figure 2. Memory Map



4b

Figure 2 shows the internal data area (IDA) is a 256-byte internal RAM area followed consecutively by a 256-byte special function register (SFR) area. All the data and control registers for on-chip peripherals and I/O are mapped into the SFR area and accessed as RAM. For a description of these functions, see table 6. The IDA is dynamically relocatable in 4K-byte increments by changing the value in the internal data base (IDB) register. Whatever value is in this register will be assigned as the uppermost eight bits of the IDA address. The IDB register can be accessed from two different memory locations, FFFFFH and XXFFFH, where XX is the value in the IDB register.

On reset, the internal data base register is set to FFH which maps the IDA into the internal ROM space. However, since the μPD70332 has a separate bus to internal ROM, this does not present a problem. When these address spaces overlap, program code cannot be executed from the IDA and internal ROM locations cannot be accessed as data.

Figure 2 shows that the internal data area is divided into 2 parts: the 256 byte internal RAM and the special function register area.

The internal RAM area serves various purposes. When the RAMEN bit in the Processor Control Register is set, this area may be accessed as RAM and code may be executed from it. Note that the processor may run slower when the RAMEN bit is set. See the Instruction Clock Count table. In addition, whether the RAMEN bit is on or off, each of the 8 macroservice channels has an 8 byte control block that is assigned to a fixed location in the low 64 bytes of the internal RAM. Similarly, the two 8 byte DMA control blocks are assigned to the low 16 bytes of the RAM. The 8 CPU register banks use 32 bytes each. Since the RAM can't be used for more than one purpose, there are restrictions on how V35 features can be combined. For example, if register bank 0 is used, then macroservice channels 0-3 and both DMA channels cannot be used. If DMA channel 1 is used, then macroservice channel 1 cannot be used.

The special function register area contains the registers used to control the onboard peripheral functions. Table 6 shows the SFRs. The address shown in the table is an offset from the IDB register. Most SFRs can be both read and written, but some are read-only; others are write-only. Some SFRs may be accessed one bit at a time; others only 8 bits at a time, and some SFRs are 16 bits wide.

Instructions

The μPD70330/332 instruction set is fully compatible with the V20 native mode instruction set. The V20 instruction set is a superset of the μPD8086/8088 instruction set with different execution times and mnemonics.

The μPD70330/332 does not support the V20 8080 emulation mode. All of the instructions pertaining to this have been deleted from the μPD70330/332 instruction set.

Enhanced Instructions

In addition to the μPD8086/88 instructions, the μPD70330/332 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes eight general registers onto stack
POP R	Pops eight general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Unique Instructions

The μPD70330/332 has the following unique instructions.

Instruction	Function
INS	Inserts bit field
EXT	Extracts bit field
ADD4S	Performs packed BCD string addition
SUB4S	Performs packed BCD string subtraction
CMP4S	Performs packed BCD string comparison
ROL4	Rotates BCD digit left
ROR4	Rotates BCD digit right
TEST1	Tests bit
SET1	Sets bit
CLR1	Clears bit
NOT1	Complements bit
BTCLR	Tests bit; if true, clear and branch
REPC	Repeat while carry set
REPNC	Repeat while carry cleared

Variable Length Bit Field Operation Instructions

Bit fields are a variable length data structure that can range in length from 1 to 16 bits. The μPD70330/332 supports two separate operations on bit fields: insertion (INS) and extraction (EXT). There are no restrictions on the position of the bit field in memory. Separate segment, byte offset, and bit offset registers are used for insertion and extraction. Following the execution of these instructions, both the byte offset and bit offset are left pointing to the start of the next bit field, ready for the next operation. Bit field operation instructions are powerful and flexible and are therefore highly effective for graphics, high-level languages, and packing/unpacking applications.

Bit field insertion copies the bit field of specified length from the AW register to the bit field addressed by DS1:IY:reg8 (8-bit general-purpose register). The bit field length can be located in any byte register or supplied as immediate data. Following execution, both the IY and reg8 are updated to point to the start of the next bit field.

Bit field extraction copies the bit field of specified length from the bit field addressed by DS0:IX:reg8 to the AW register. If the length of the bit field is less than 16 bits, the bit field is right justified with a zero fill. The bit field length can be located in any byte register or supplied as immediate data. Following execution, both IX and reg8 are updated to point to the start of the next bit field.

Figures 3 and 4 show bit field insertion and bit field extraction.

Packed BCD Instructions

Packed BCD instructions process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte format operands (ROR4, ROL4). Packed BCD strings may be 1 to 254 digits in length. The two BCD rotation instructions perform rotation of a single BCD digit in the lower half of the AL register through the register or the memory operand.

4b

Figure 3. Bit Field Insertion

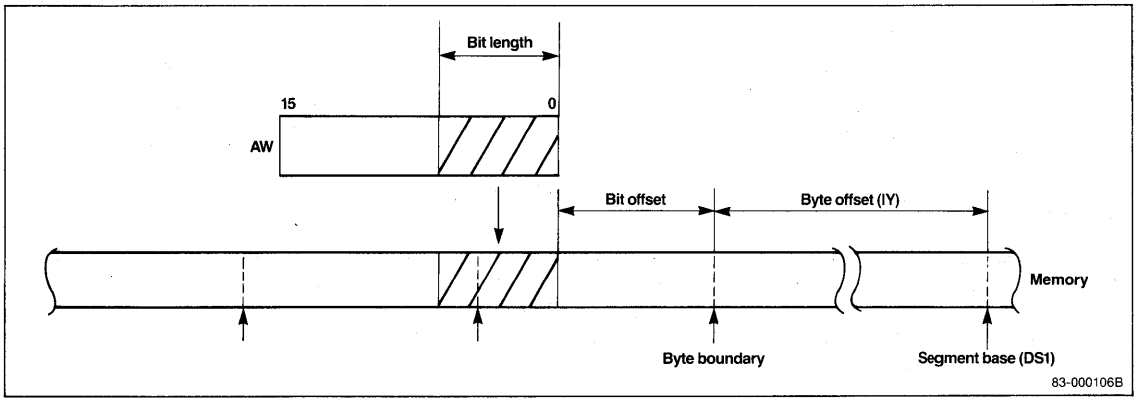
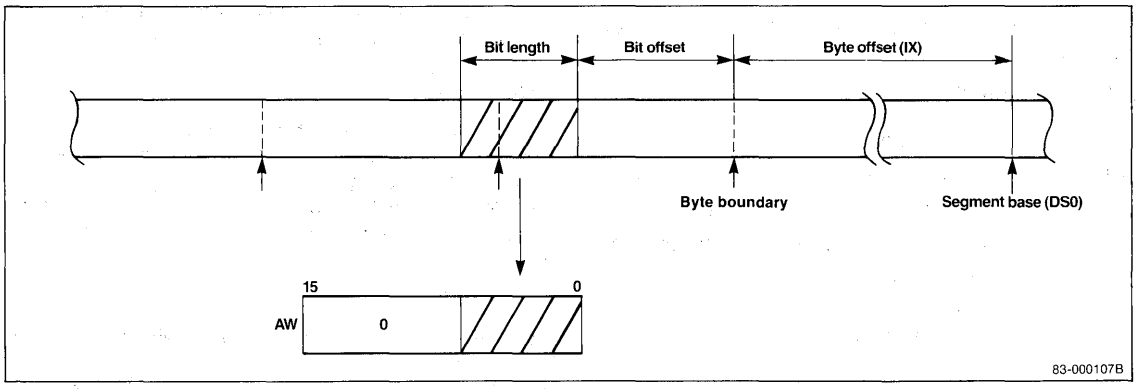


Figure 4. Bit Field Extraction



Bit Manipulation Instructions

The μPD70330/332 has five unique bit manipulation instructions. The ability to test, set, clear, or complement a single bit in a register or memory operand increases code readability as well as performance over the logical operations traditionally used to manipulate bit data. This feature further enhances control over on-chip peripherals.

Additional Instructions

Besides the V20 instruction set, the μPD70330/0332 has the eight additional instructions described in table 1.

Table 1. Additional Instructions

Instruction	Function
BTCLR var,imm8, short label	Bit test and if true, clear and branch; otherwise, no operation
STOP (no operand)	Power down instruction, stops oscillator
RETRBI (no operand)	Return from register bank context switch interrupt
FINT (no operand)	Finished interrupt. After completion of a hardware interrupt request, this instruction must be used to reset the current priority bit in the in-service priority register (ISPR).*

*Do not use with NMI or INTR interrupt service routines.

Repeat Prefixes

Two new repeat prefixes (REPC, REPNC) allow conditional block transfer instructions to use the state of the CY flag as the termination condition. This allows inequalities to be used when working on ordered data, thus increasing performance when searching and sorting algorithms.

Bank Switch Instructions

The V35 has four new instructions that allow the effective use of the register banks for software interrupts and multitasking. These instructions are shown in table 2. Also, see figures 8 and 10.

Table 2. Bank Switch Instructions

Instruction	Function
BRKCS reg 16	Performs a high-speed software interrupt with context switch to the register bank indicated by the lower 3-bits of reg 16. This operation is identical to the interrupt operation shown in figure 9.
TSKSW reg 16	Performs a high-speed task switch to the register bank indicated by the lower 3-bits of reg 16. The PC and PSW are saved in the old banks. PC and PSW save registers and the new PC and PSW values are retrieved from the new register bank's save areas. See figure 10.
MOVSPA	Transfers both the SS and SP of the old register bank to the new register bank after the bank has been switched by an interrupt or BRKCS instruction.
MOVSPB	Transfers the SS and the SP of the current register bank before the switch to the SS and SP of the new register bank indicated by the lower 3-bits of reg 16.

Interrupt Structure

The μPD70330/332 can service interrupts generated both by hardware and by software. Software interrupts are serviced through vectored interrupt processing. See table 3 for the various types of software interrupts.

Table 3. Software Interrupts

Interrupt	Description
Divide error	The CPU will trap if a divide error occurs as the result of a DIV or DIVU instruction.
Single step	The interrupt is generated after every instruction if the BRK bit in the PSW is set.
Overflow	By using the BRKV instruction, an interrupt can be generated as the result of an overflow.
Interrupt instructions	The BRK 3 and BRK imm8 instructions can generate interrupts.
Array bounds	The CHKIND instruction will generate an interrupt if specified array bounds have been exceeded.
Escape trap	The CPU will trap on an FP01,2 instruction to allow software to emulate the floating point processor.
I/O trap	If the I/O trap bit in the PSW is cleared, a trap will be generated on every IN or OUT instruction. Software can then provide an updated peripheral address. This feature allows software interchangeability between different systems.

When executing software written for another system, it is better to implement I/O with on-chip peripherals to reduce external hardware requirements. However, since μPD70330/332 internal peripherals are memory mapped, software conversion could be difficult. The I/O trap feature allows easy conversion from external peripherals to on-chip peripherals.

Interrupt Vectors

The starting address of the interrupt processing routines may be obtained from table 4. The table begins at physical address 00H, which is outside the internal ROM space. Therefore, external memory is required to service these routines. By servicing interrupts via the macro service function or context switching, this requirement can be eliminated.

Each interrupt vector is four bytes wide. To service a vectored interrupt, the lower addressed word is transferred to the PC and the upper word to the PS. See figure 5.

Figure 5. Interrupt Vector 0

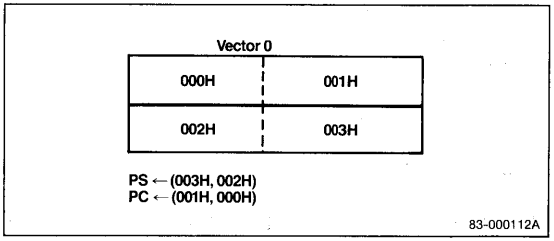


Table 4. Interrupt Vectors

Address	Vector No.	Assigned Use
00	0	Divide error
04	1	Break flag
08	2	NMI
0C	3	BRK3 instruction
10	4	BRKV instruction
14	5	CHKIND instruction
18	6	General purpose
1C	7	FPO instructions
20-2C	8-11	General purpose
30	12	INTSER0 (Interrupt serial error, channel 0)
34	13	INTSR0 (Interrupt serial receive, channel 0)
38	14	INTST0 (Interrupt serial transmit, channel 0)
3C	15	General purpose
40	16	INTSER1 (Interrupt serial error, channel 1)
44	17	INTSR1 (Interrupt serial receive, channel 1)
48	18	INTST1 (Interrupt serial transmit, channel 1)
4C	19	I/O trap
50	20	INTD0 (Interrupt from DMA, channel 0)
54	21	INTD1 (Interrupt from DMA, channel 1)
58	22	General purpose
5C	23	General purpose
60	24	INTP0 (Interrupt from peripheral 0)
64	25	INTP1 (Interrupt from peripheral 1)
68	26	INTP2 (Interrupt from peripheral 2)
6C	27	General purpose
70	28	INTTU0 (Interrupt from timer unit 0)
74	29	INTTU1 (Interrupt from timer unit 1)
78	30	INTTU2 (Interrupt from timer unit 2)
7C	31	INTTB (Interrupt from time base counter)
080-3FF	32-255	General purpose

Execution of a vectored interrupt occurs as follows:

(SP-1, SP-2) ← PSW
(SP-3, SP-4) ← PS
(SP-5, SP-6) ← PC
SP ← SP-6
IE ← 0, BRK ← 0
PS ← vector high bytes
PC ← vector low bytes

Hardware Interrupt Configuration

The V35 features a high-performance on-chip controller capable of controlling multiple processing for interrupts from up to 17 different sources (5 external, 12 internal). The interrupt configuration includes system interrupts that are functionally compatible with those of the V20/V30 and unique high-performance microcontroller interrupts.

Interrupt Sources

The 17 interrupt sources (table 5) are divided into groups for management by the interrupt controller. Using software, each of the groups can be assigned a priority from 0 (highest) to 7 (lowest). The priority of individual interrupts within a group is fixed in hardware.

If interrupts from different groups occur simultaneously and the groups have the same assigned priority level, the priority followed will be as shown in the Default Priority column of table 5.

The ISPR is an 8-bit SFR; bits PR_0-PR_7 correspond to the eight possible interrupt request priorities. The ISPR keeps track of the priority of the interrupt currently being serviced by setting the appropriate bit. The address of the ISPR is XXFFCH. The ISPR format is shown below.

PR_7	PR_6	PR_5	PR_4	PR_3	PR_2	PR_1	PR_0

NMI and INT are system-type external vectored interrupts. NMI is not maskable via software. INTR is maskable (IE bit in PSW) and requires that an external device provide the interrupt vector number. It allows expansion by the addition of an external interrupt controller (μPD71059).

NMI, INTP0, and INTP1 are edge-sensitive interrupt inputs. By selecting the appropriate bits in the interrupt mode register, these inputs can be programmed to be either rising or falling edge triggered. ES_0-ES_2 correspond to INTP0-INTP2, respectively. See figure 6.

Figure 6. Interrupt Mode Register (INTM)

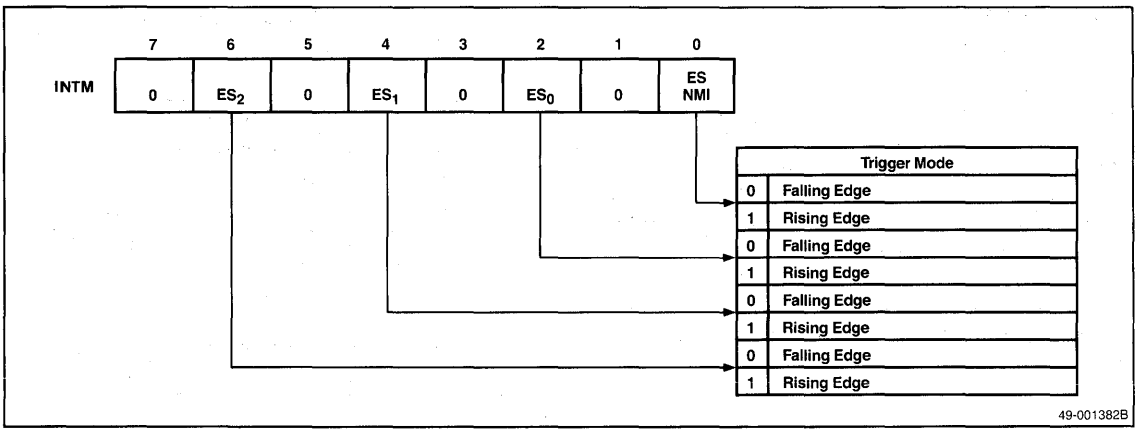


Table 5. Interrupt Sources

Interrupt Source	External/ Internal	Vector	Macro Service	Bank Switching	Priority Order: Setting Possible	Priority Order: Between Groups	Priority Order: Within Groups	Multiple Processing Control
NMI Nonmaskable interrupt	External	2	No	No	No	0	—	Not accepted
INTTU0 Interrupt from timer unit 0	Internal	28	Yes	Yes	Yes	1	1	Accepted
INTTU1 Interrupt from timer unit 1	Internal	29	Yes	Yes	Yes	1	2	
INTTU2 Interrupt from timer unit 2	Internal	30	Yes	Yes	Yes	1	3	
INTD0 Interrupt from DMA channel 0	Internal	20	No	Yes	Yes	2	1	Accepted
INTD1 Interrupt from DMA channel 1	Internal	21	No	Yes	Yes	2	2	
INTP0 Interrupt from peripheral 0	External	24	Yes	Yes	Yes	3	1	Accepted
INTP1 Interrupt from peripheral 1	External	25	Yes	Yes	Yes	3	2	
INTP2 Interrupt from peripheral 2	External	26	Yes	Yes	Yes	3	3	
INTSER0 Interrupt from serial error on channel 0	Internal	12	No	Yes	Yes	4	1	Accepted
INTSR0 Interrupt from serial receiver of channel 0	Internal	13	Yes	Yes	Yes	4	2	
INTST0 Interrupt from serial transmitter of channel 0	Internal	14	Yes	Yes	Yes	4	3	
INTSER1 Interrupt from serial error on channel 1	Internal	16	No	Yes	Yes	5	1	Accepted
INTSR1 Interrupt from serial receiver of channel 1	Internal	17	Yes	Yes	Yes	5	2	
INTST1 Interrupt from serial transmitter of channel 1	Internal	18	Yes	Yes	Yes	5	3	
INTTB Interrupt from time base counter	Internal	31	No	No	No (Preset to 7)	6	—	Accepted
INT Interrupt	External	Ext. input	No	No	No	7	—	Not accepted

4b

Interrupt Processing Modes

Interrupts, with the exception of NMI, INT, and INTTB, have high-performance capability and can be processed in any of three modes: standard vectored interrupt, register bank context switching, or macro service function. The processing mode for a given interrupt can be chosen by enabling the appropriate bits in the corresponding interrupt request control register. As shown in table 6, each individual interrupt, with the exception of INTR and NMI, has its own associated IRC register. The format for all IRC registers is shown in figure 7. There is an IRC for every interrupt source except NHI and INT.

All interrupt processing routines other than those for NMI and INT must end with the execution of an FINT instruction. Otherwise, subsequently, only interrupts of a higher priority will be accepted. FINT allows the internal interrupt controller to begin looking for new interrupts.

In the vectored interrupt mode, the CPU traps to the vector location in the interrupt vector table.

Register Bank Switching

Register bank context switching allows interrupts to be processed rapidly by switching register banks. After an interrupt, the new register bank selected is that which has the same register bank number (0-7) as the priority of the interrupt to be serviced. The PC and PSW are automatically stored in the save areas of the new register bank and the address of the interrupt routine is loaded from the vector PC storage location in the new register bank. As in the vectored mode, the IE and BRK bits in the PSW are cleared to zero. After interrupt processing, execution of the RETRBI (return from register bank interrupt) returns control to the former register bank and restores the former PC and PSW. Figures 8 and 9 show register bank context switching and register bank return.

Specific IRC registers include the following.

Symbol	IRC Register
DIC0, DIC1	DMA
EXIC0-EXIC2	External
SEIC0, SEIC1	Serial error
SRIC0, SRIC1	Serial receive
STIC0, STIC1	Serial transmit
TMIC0-TMIC2	Timer

Figure 7. Interrupt Request Control Registers (IRC)

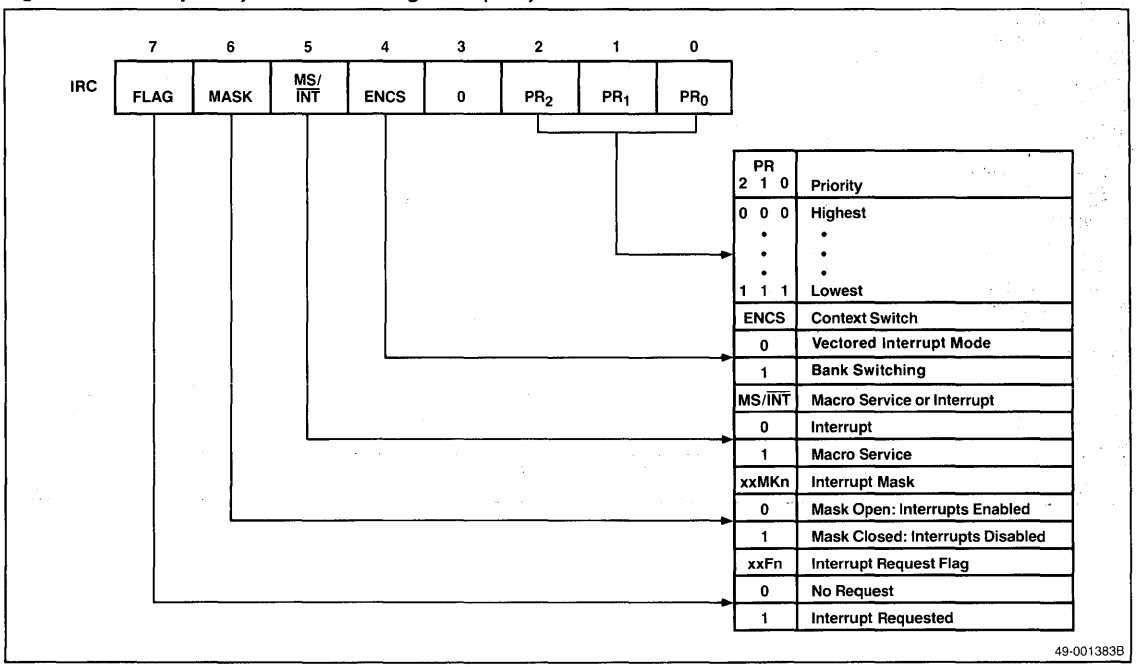


Figure 8. Register Bank Context Switching

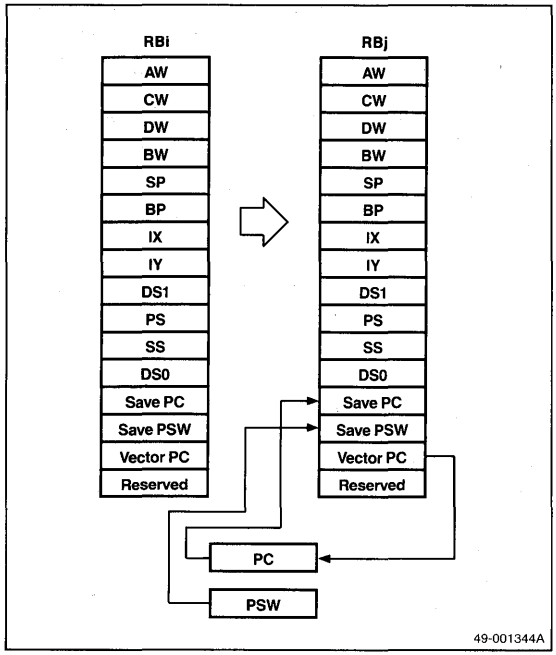
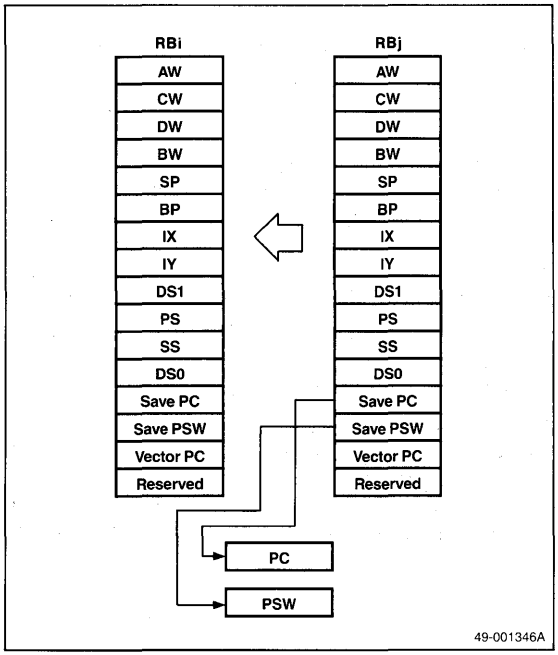


Figure 9. Register Bank Return



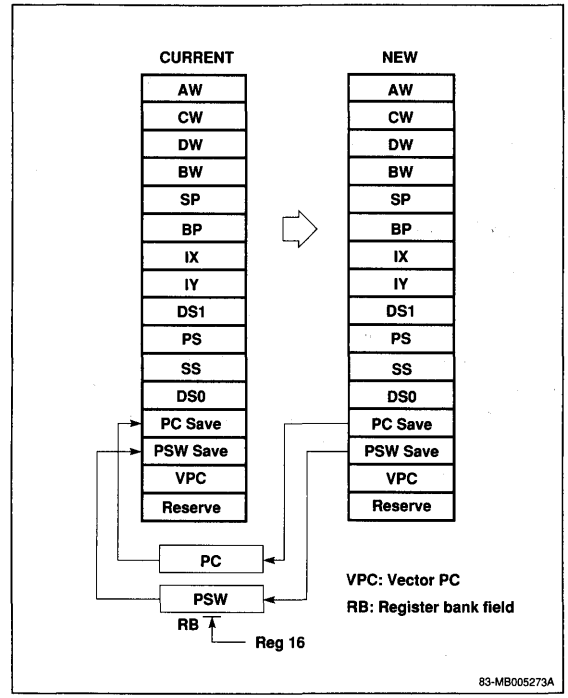
Macro Service Function

The macro service function (MSF) is a special microprogram that acts as an internal DMA controller between on-chip peripherals (special function registers, SFR) and memory. The MSF greatly reduces the software overhead and CPU time that other processors would require for register save processing, register returns, and other handling associated with interrupt processing.

If the MSF is selected for a particular interrupt, each time the request is received, a byte or word of data will be transferred between the SFR and memory without interrupting the CPU. Each time a request occurs, the macro service counter is decremented. When the counter reaches zero, an interrupt to the CPU is generated. The MSF also has a character search option. When selected, every byte transferred will be compared to an 8-bit search character and an interrupt will be generated if a match occurs or if the macro service counter counts out.

Like the NMI, INT and INTTB, the two DMA controller interrupts (INTD0, INTD1) do not have MSF capability.

Figure 10. Task Switching

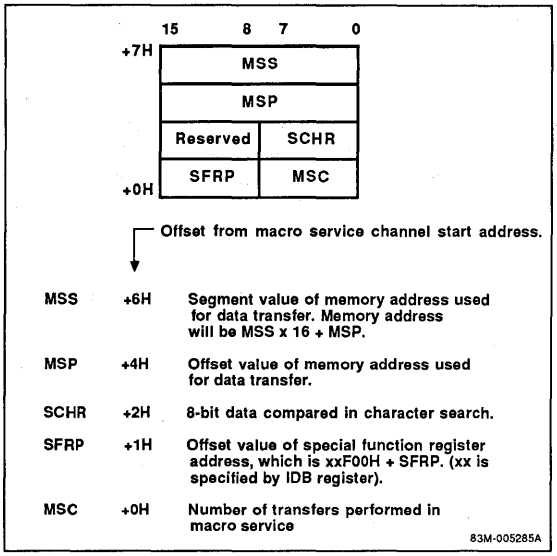


4b

There are eight 8-byte macro service channels mapped into internal RAM from XXE00H to XXE3FH. Figure 11 shows the components of each channel.

Setting the macro service mode for a given interrupt requires programming the corresponding macro service control register. Each individual interrupt, excluding INTR, NMI and TBC, has its own associated MSC register. See table 6. Format for all MSC registers is shown in figure 12.

Figure 11. Macro Service Channels



On-Chip Peripherals

Timer Unit

The μPD70330/332 (figure 13) has two programmable 16-bit interval timers (TM0, TM1) on-chip, each with variable input clock frequencies. Each of the two 16-bit timer registers has an associated 16-bit modulus register (MD0, MD1). Timer 0 operates in the interval timer mode or one-shot mode; timer 1 has only the interval timer mode.

Interval Timer Mode. In this mode, TM0/TM1 are decremented by the selected input clock and, after counting out, the registers are automatically reloaded from the modulus registers and counting continues. Each time TM1 counts out, interrupts are generated through TF1 and TF2 (Timer Flags 1, 2). When TM0 counts out, an interrupt is generated through TF0. The timer-out signal can be used as a square-wave output whose half-cycle is equal to the count time. There are two selectable input clocks (SCLK: system clock = $f_{OSC}/2$; f_{OSC} = 10 MHz).

Clock	Timer Resolution	Full Count
SCLK/6	1.2 μs	78.643 ms
SCLK/128	25.6 μs	1.678 s

One-Shot Mode. In the one-shot mode, TM0 and MD0 operate as independent one-shot timers. Starting with a preset value, each is decremented to zero. At zero, counting ceases and an interrupt is generated by TF0 (from TM0) or TF1 (from MD0). One-shot mode allows two selectable input clocks (f_{OSC} = 10 MHz).

Clock	Timer Resolution	Full Count
SCLK/12	2.4 μs	157.283 ms
SCLK/128	25.6 μs	1.678 s

Setting the desired timer mode requires programming the timer control register. See figures 14 and 15 for format.

Time Base Counter/Processor Control Register

The 20-bit free-running time base counter controls internal timing sequences and is available to the user as the source of periodic interrupts at lengthy intervals. One of four interrupt periods can be selected by programming the TB0 and TB1 bits in the processor control register (PRC). The TBC interrupt is unlike the others in that it is fixed as a level 7 vectored interrupt. Macro service and register bank switching cannot be used to service this interrupt. See figures 16 and 17.

The RAMEN bit in the PRC register allows the internal RAM to be removed from the memory address space to implement faster instruction execution.

The TBC (figure 18) uses the system clock as the input frequency. The system clock can be changed by programming the PCK0 and PCK1 bits in the processor control register (PRC). Reset initializes the system clock to $f_{OSC}/8$ (f_{OSC} = external oscillator frequency).

Figure 12. Macro Service Control Registers (MSC)

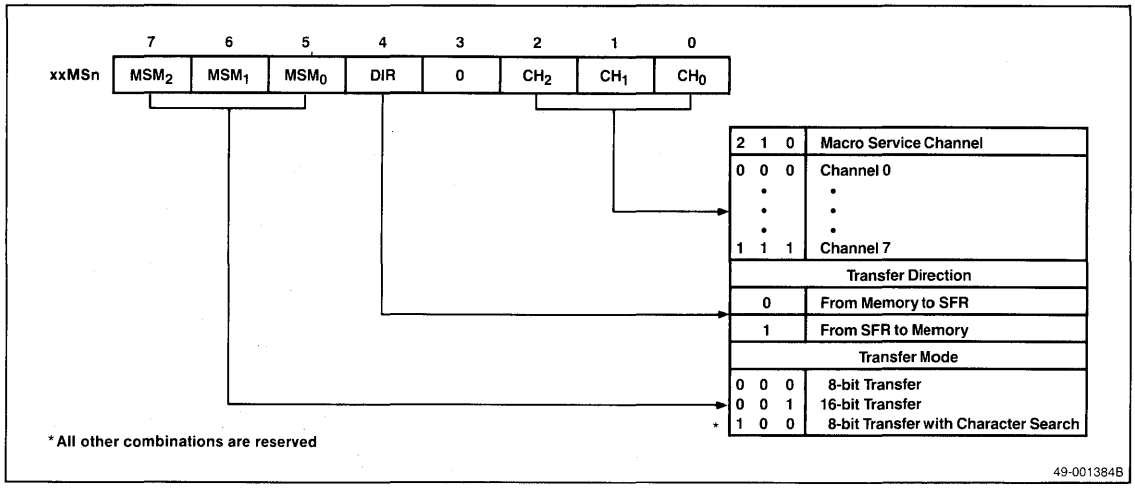
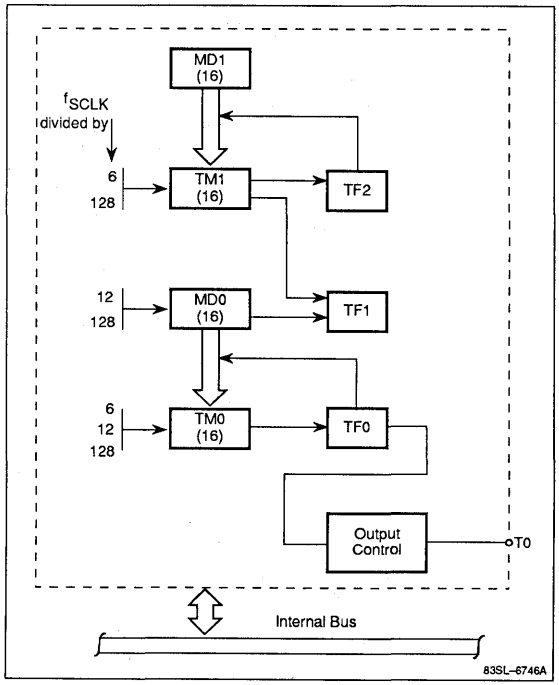


Figure 13. Timer Unit Block Diagram



4b

Figure 14. Timer Control Register 0

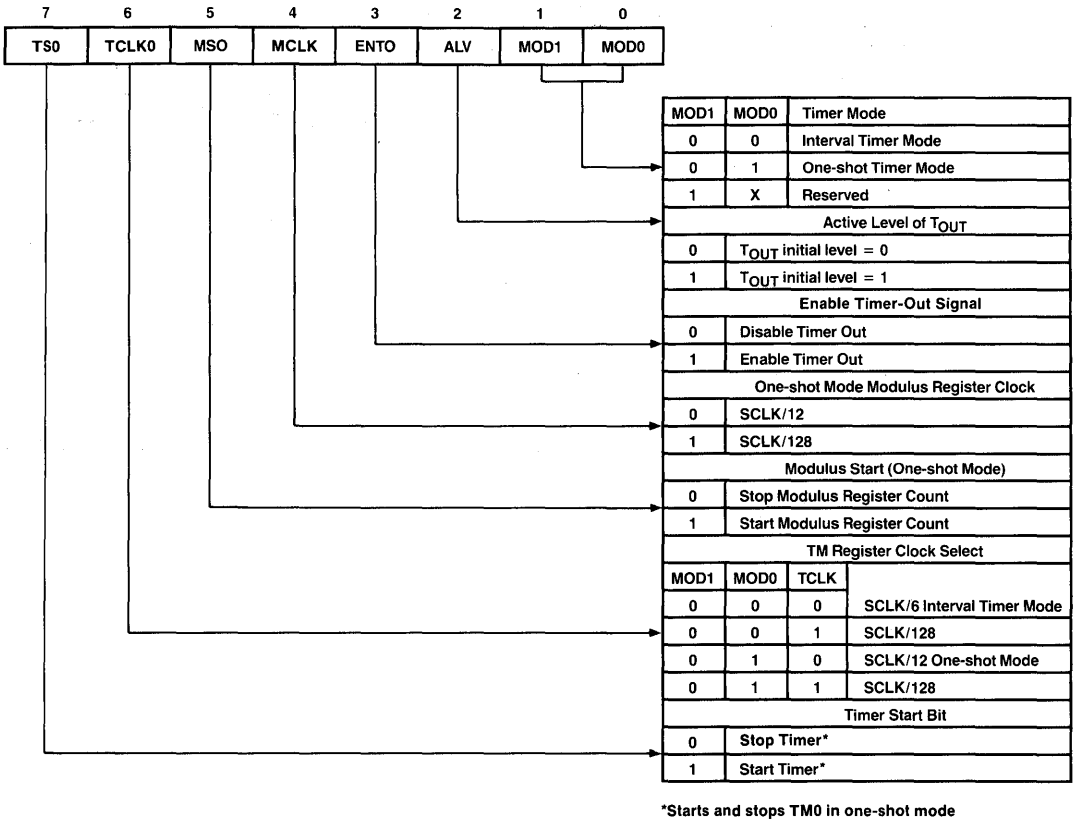


Figure 15. Timer Control Register 1

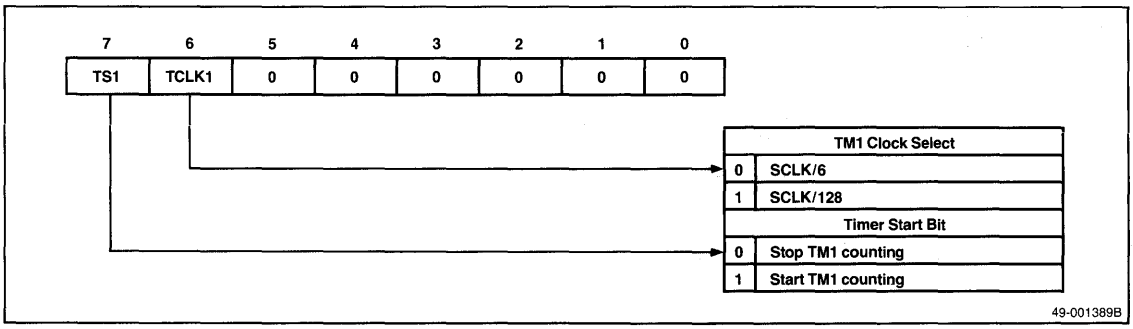


Figure 16. Time Base Interrupt Request Control Register

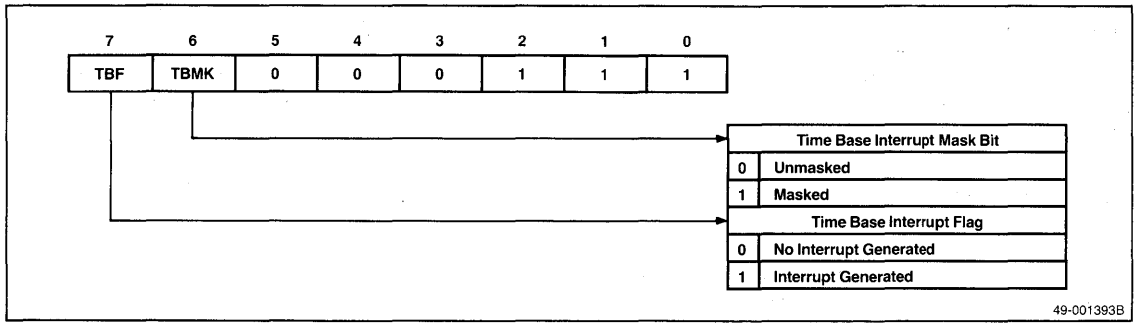


Figure 17. Processor Control Register (PRC)

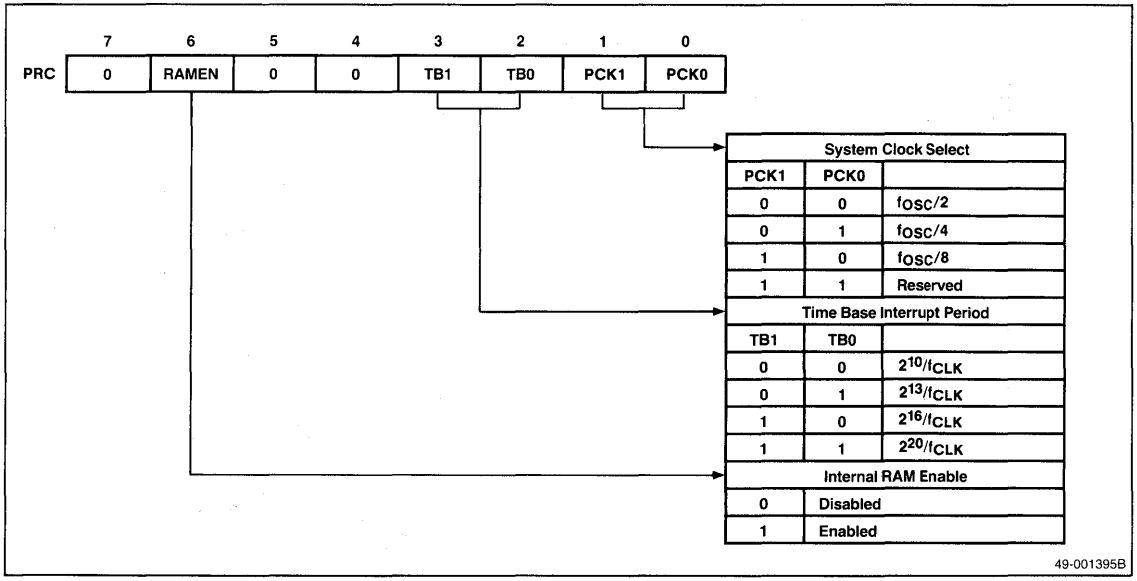
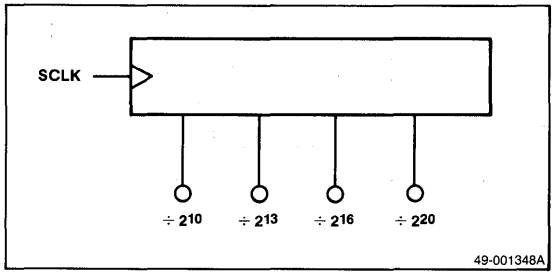


Figure 18. Time Base Counter (TBC) Block Diagram



4b

Refresh Controller

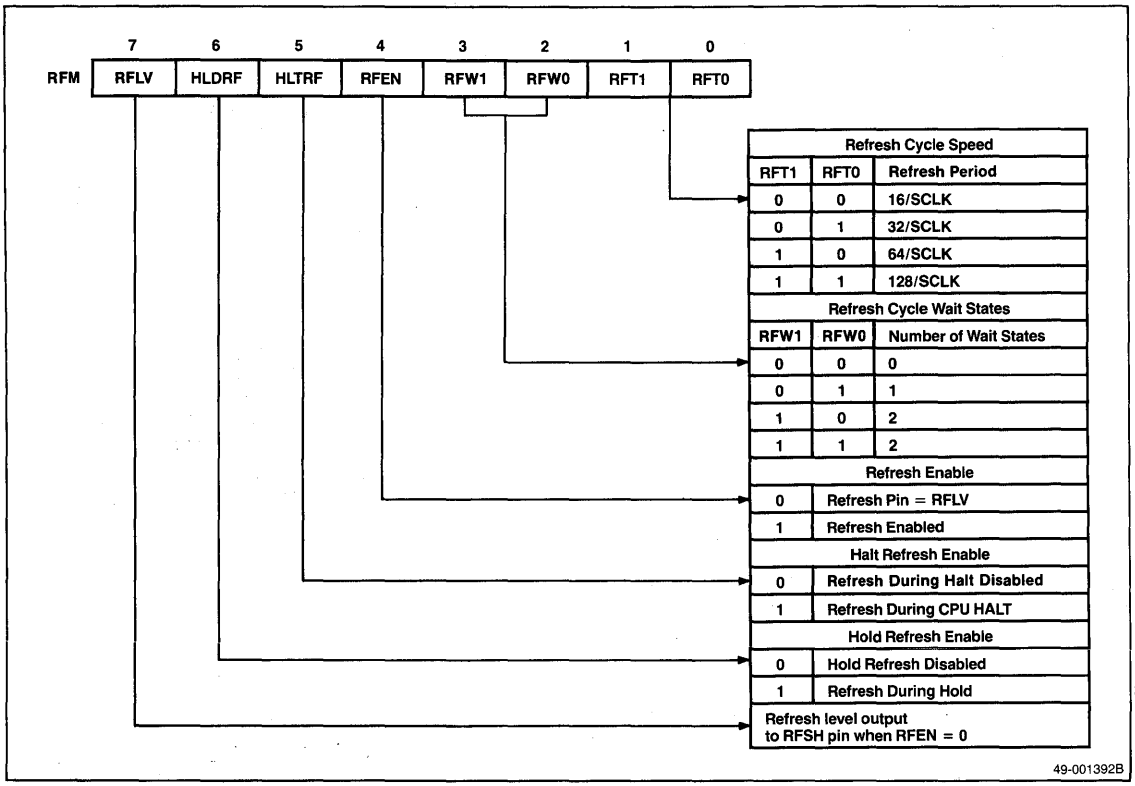
The μPD70330/332 has an on-chip refresh controller for dynamic and pseudostatic RAM mass storage memories. The refresh controller generates refresh addresses and refresh pulses. It inserts refresh cycles between the normal CPU bus cycles according to refresh specifications.

The refresh controller outputs a 9-bit refresh address on address bits A_0-A_8 during the refresh bus cycle. Address bits A_9-A_{19} are all 1's. The 9-bit refresh address is automatically incremented at every refresh timing for 512 row addresses. The 8-bit refresh mode (RFM) register (figure 19) specifies the refresh operation and allows refresh during both CPU HALT and HOLD modes. Refresh cycles are automatically timed to $\overline{\text{REFRQ}}$ following read/write cycles to minimize the effect on system thoughput.

The following shows the $\overline{\text{REFRQ}}$ pin level in relation to bits 4 (RFEN) and 7 (RFLV) of the refresh mode register.

RFEN	RFLV	$\overline{\text{REFRQ}}$ Level
0	0	0
0	1	1
1	0	0
1	1	Refresh pulse output

Figure 19. Refresh Mode Register (RFM)



Serial Interface

The μPD70330/332 has two full-duplex UARTs, channel 0 and channel 1. Each serial port channel has a transmit line (TxDn), a receive line (RxDn), and a clear to send (CTSn) input line for handshaking. Communication is synchronized by a start bit, and you can program the ports for even, odd, or no parity, character lengths of 7 or 8 bits, and 1 or 2 stop bits.

The μPD70330/332 has dedicated baud rate generators for each serial channel. This eliminates the need to obligate the on-chip timers. The baud rate generator allows a wide range of data transfer rates (up to 1.25 Mb/s). This includes all of the standard baud rates without being restricted by the value of the particular external crystal.

Each baud rate generator has an 8-bit baud rate generator (BRGn) data register, which functions as a prescaler to a programmable input clock selected by the serial communication control (SCCn) register. Together these must be set to generate a frequency equivalent to the desired baud rate.

The baud rate generator can be set to obtain the desired transmission rate according to the following formula:

$$B \times G = \frac{SCLK \times 10^6}{2^{n+1}}$$

where B = baud rate
G = baud rate generator register (BRGn) value
n = input clock specifications (n between 0 and 8). This is the value that is loaded into the SCCn register. See figure 23.
SCLK = system clock frequency (MHz)

Based on the above expression, the following table shows the baud rate generator values used to obtain standard transmission rates when SCLK = 5 MHz.

Baud Rate	n	BRGn Value	Error (%)
110	7	178	0.25
150	7	130	0.16
300	6	130	0.16
600	5	130	0.16
1200	4	130	0.16
2400	3	130	0.16
4800	2	130	0.16
9600	1	130	0.16
19,200	0	130	0.16
38,400	0	65	0.16
1.25M	0	2	0

In addition to the asynchronous mode, channel 0 has a synchronous I/O interface mode. In this mode, each bit of data tranferred is synchronized to a serial clock ($\overline{SCKO}$). This is the same as the NEC μCOM75 and μCOM87 series, and allows easy interfacing to these devices. Figure 20 is the serial interface block diagram; figures 21, 22, and 23 show the three serial communication registers.

4b

DMA Controller

The μPD70330/332 has a two-channel, on-chip DMA controller. This allows rapid data transfer between memory and auxiliary storage devices. The DMA controller supports four modes of operation, two for memory-to-memory transfers and two for transfers between I/O and memory. See figures 24, 25, and 26 for a graphic representation of the DMA registers.

Figure 20. Serial Interface Block Diagram

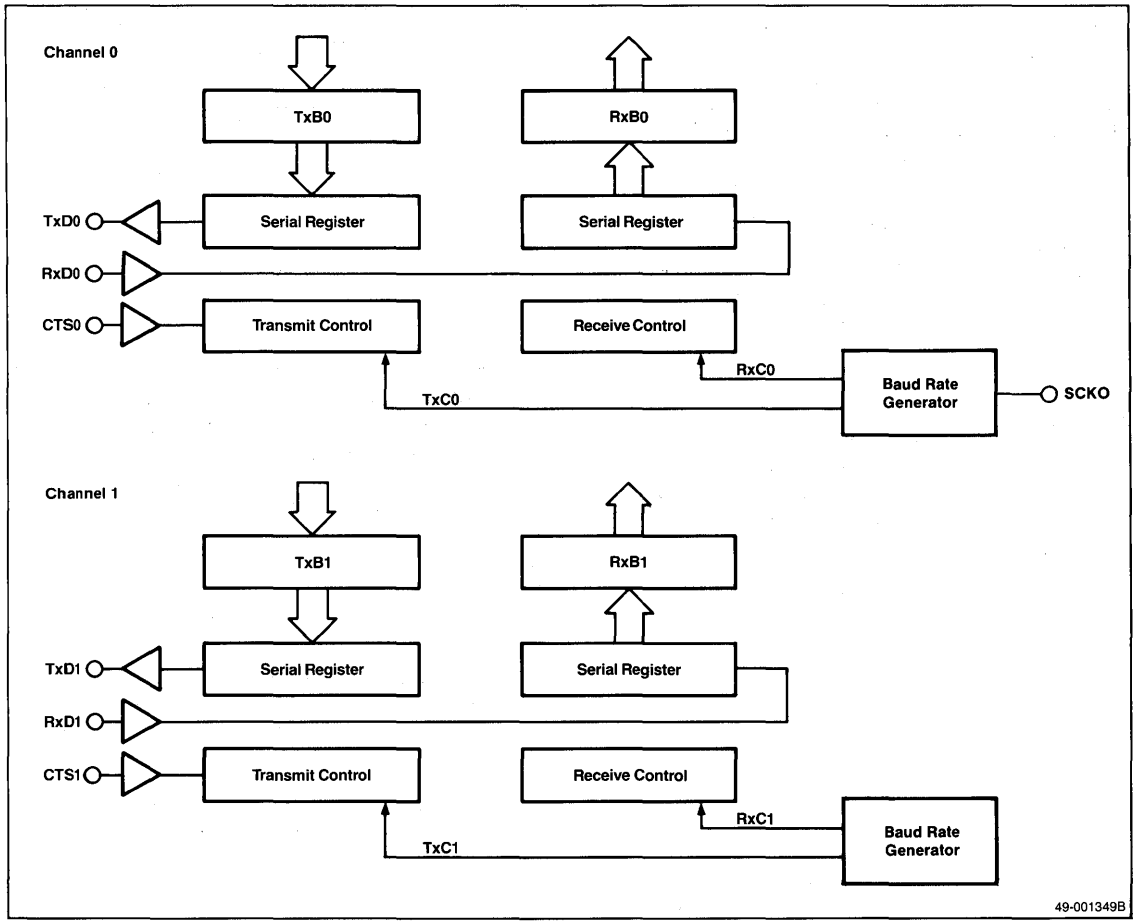
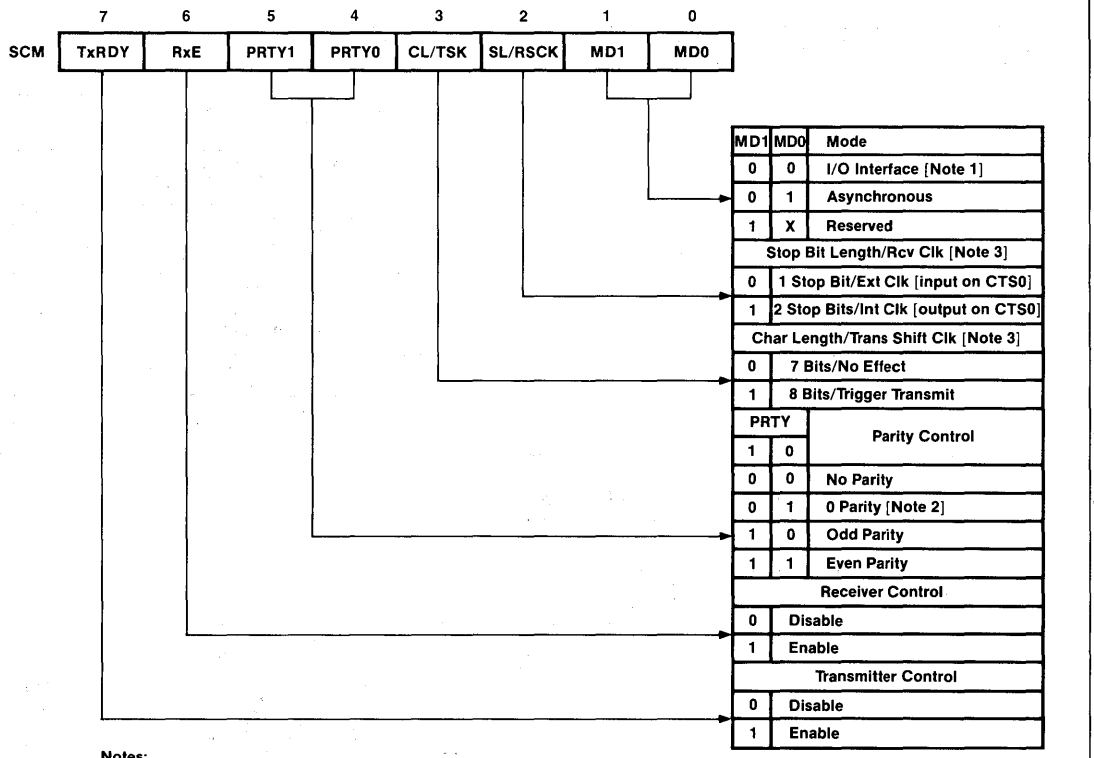


Figure 21. Serial Communication Mode Register (SCM)



Notes:
[1] Only Channel 0 has I/O interface mode.
[2] When 0 parity is selected, the parity is 0 during transmit and is ignored during receive.
[3] Applies only to I/O interface mode.

49-001385B

4b

Figure 22. Serial Communication Error Registers (SCE)

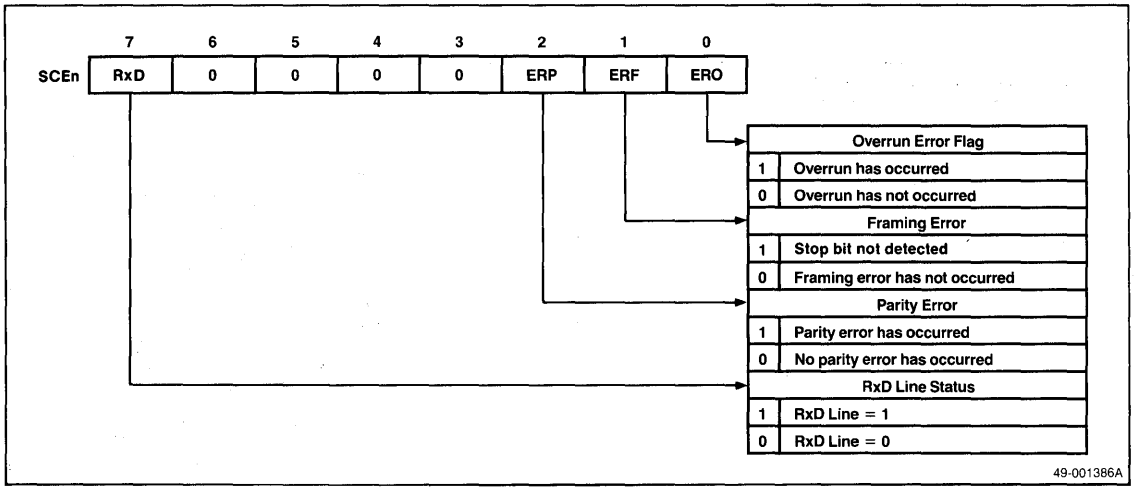


Figure 23. Serial Communication Control Register (SCC)

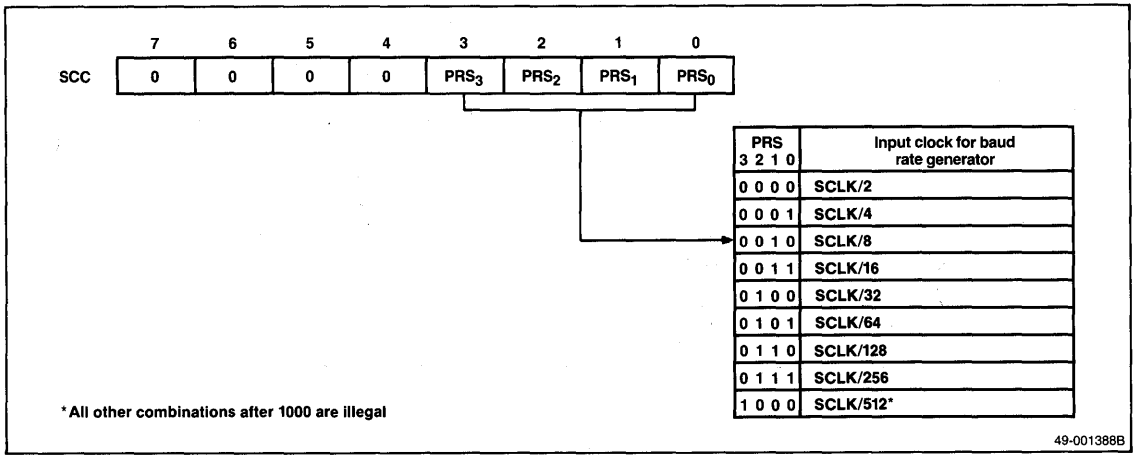


Figure 24. DMA Channels

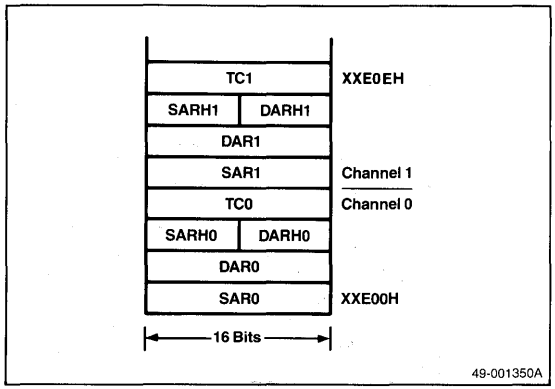
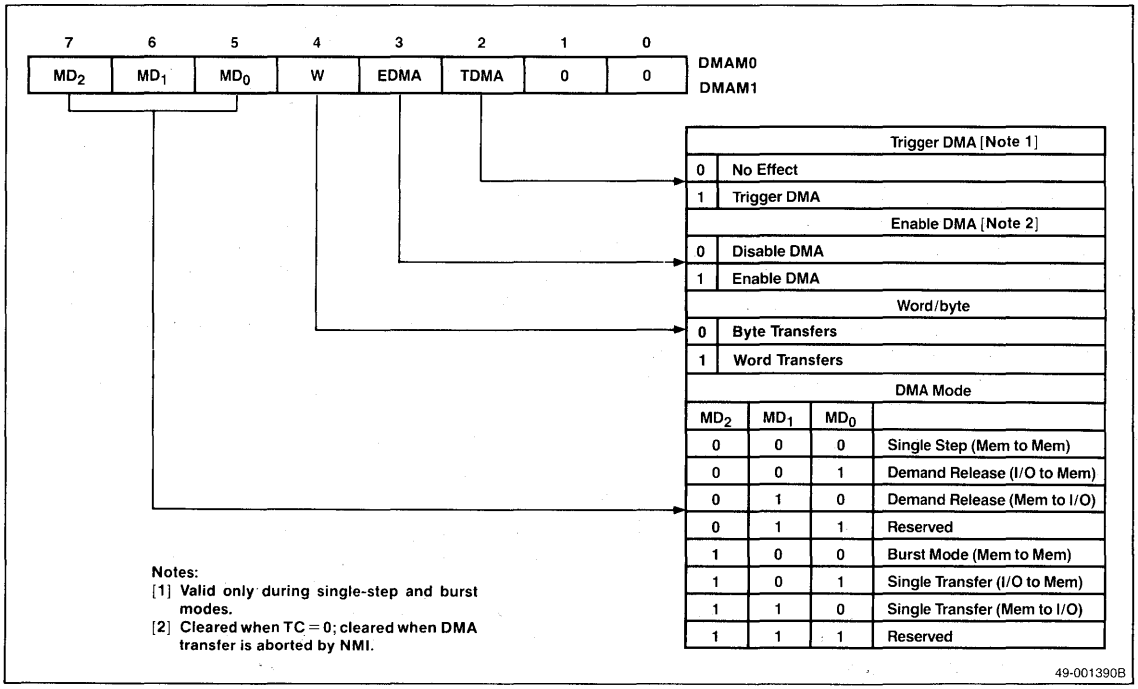
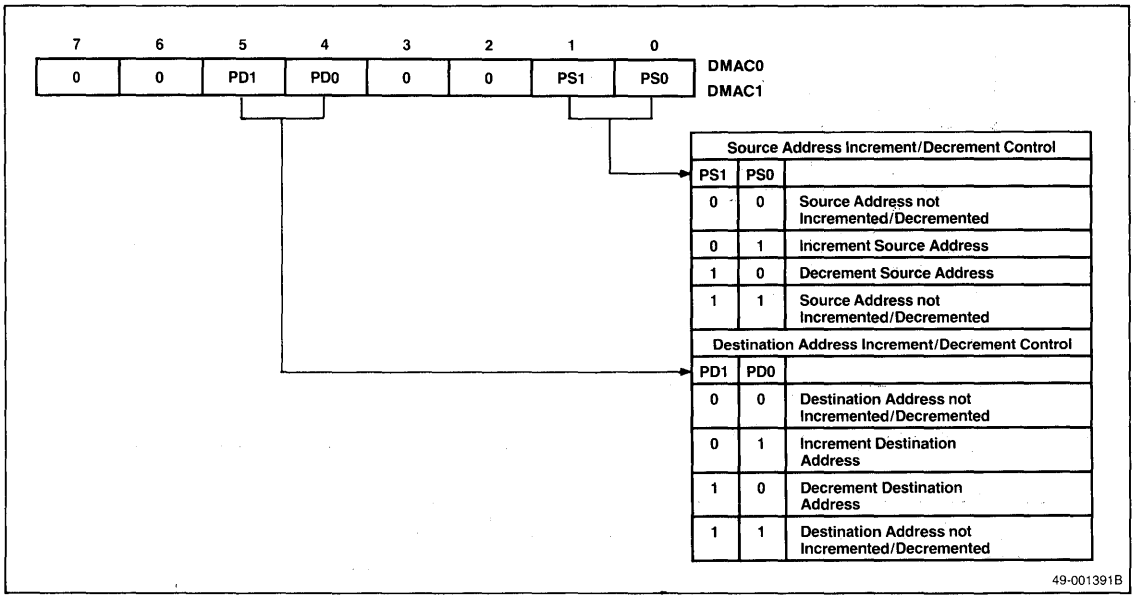


Figure 25. DMA Mode Registers (DMAM)



4b

Figure 26. DMA Control Registers (DMAC)



Memory-to-Memory Transfers. In the single-step mode, when one DMA request is made, execution of one instruction and one DMA transfer are repeated alternately until the prescribed number of DMA transfers has occurred. Interrupts can be accepted while in this mode. In burst mode, a DMA request causes DMA transfer cycles to continue until the DMA terminal counter decrements to zero. Software can also initiate memory-to-memory transfers.

Transfers Between I/O and Memory. In single-transfer mode, one DMA transfer occurs after each rising edge of DMARQ. After the transfer, the bus is returned to the CPU. In demand release mode, the rising edge of DMARQ enables DMA cycles, which continue as long as DMARQ is high.

In all modes, the $\overline{TC}$ (terminal count) output pin will pulse low and a DMA completion I/O request will be generated after the predetermined number of DMA cycles has been completed.

The bottom of internal RAM contains all the necessary address information for the designated DMA channels. The DMA channel mnemonics are as follows:

TC	Terminal counter
SAR	Source address register
SARH	Source address register high
DAR	Destination address register
DARH	Destination address register high

The DMA controller generates physical source addresses by offsetting SARH 12 bits to the left and then adding the SAR. The same procedure is also used to generate physical destination addresses. You can program the controller to increment or decrement source and/or destination addresses independently during DMA transfers.

When the EDMA bit is set, the internal DMARQ flag is cleared. Therefore, DMARQs are only recognized after the EDMA bit has been set.

Parallel Ports

The μPD70330/332 has three 8-bit parallel I/O ports: P0, P1, and P2. Refer to figures 27 through 31. Special function register (SFR) locations can access these ports. The port lines are individually programmable as inputs or outputs. Many of the port lines have dual functions as port or control lines.

Use the associated port mode and port mode control registers to select the mode for a given I/O line.

The analog comparator port (PT) compares each input line to a reference voltage. The reference voltage is programmable to be the V_{TH} input x n/16, where n = 1 to 16. See figure 32.

Figure 27. Port Mode Registers 0 and 2 (PM0, PM2)

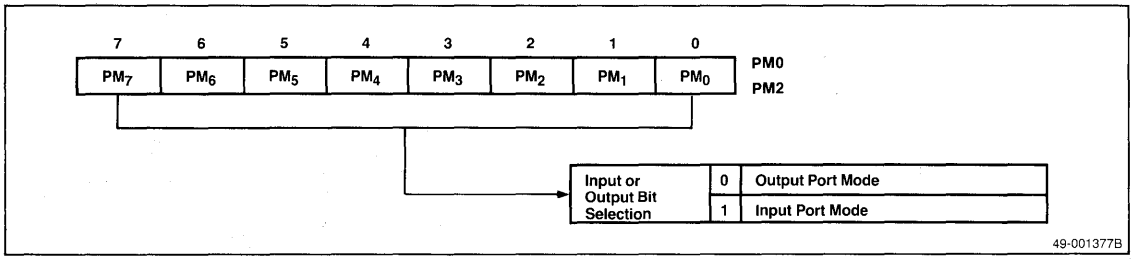


Figure 28. Port Mode Register 1 (PM1)

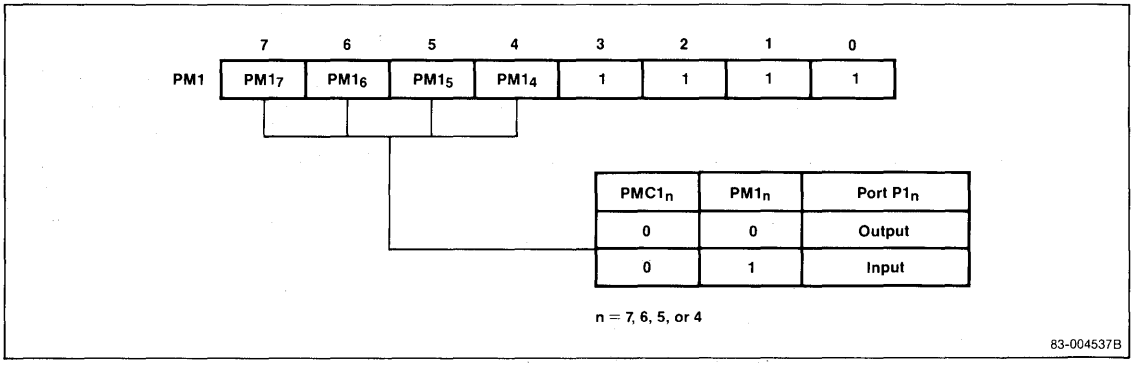
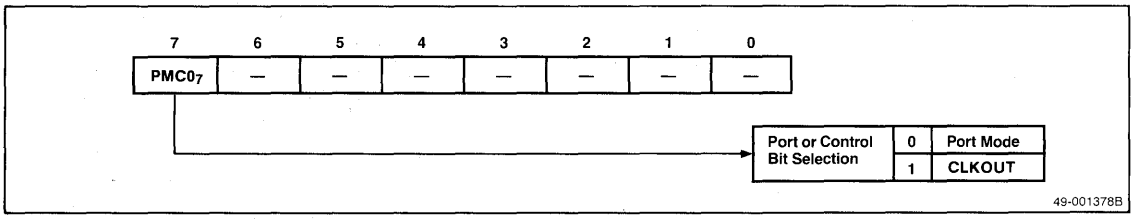


Figure 29. Port Mode Control Register 0 (PMC0)



4b

Figure 30. Port Mode Control Register 1 (PMC1)

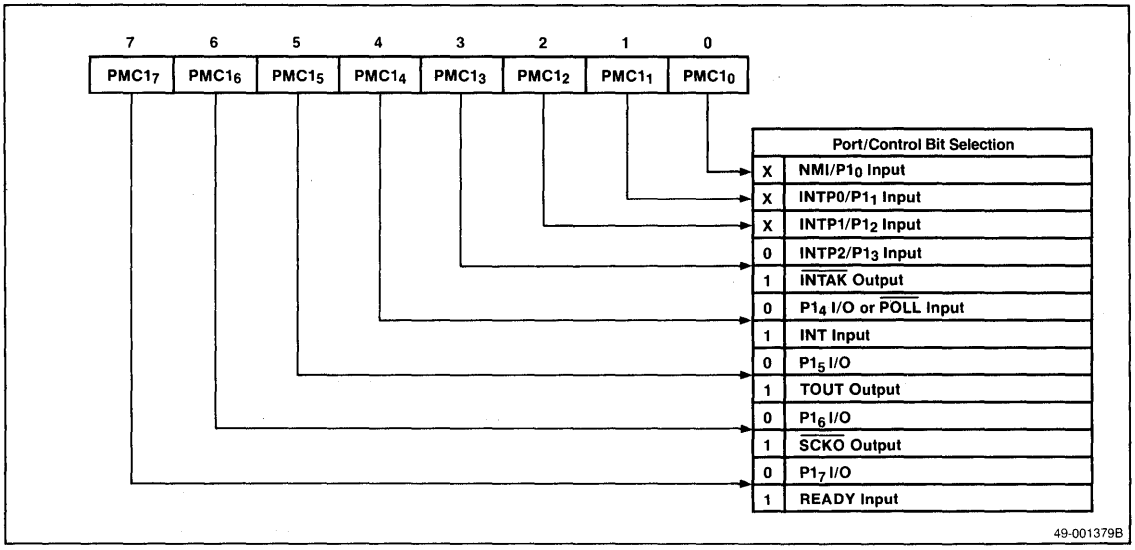


Figure 31. Port Mode Control Register 2 (PMC2)

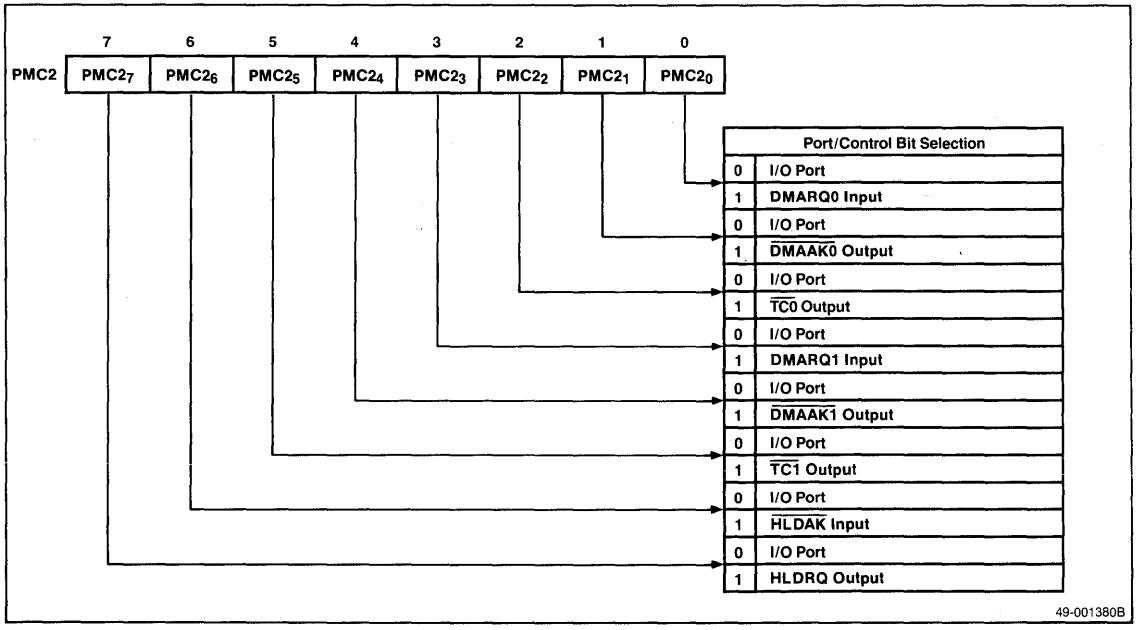


Figure 32. Port Mode Register T (PMT)

7	6	5	4	3	2	1	0	
0	0	0	0	PMT_3	PMT_2	PMT_1	PMT_0	PMT

Comparator Port Threshold Selection				
0	0	0	0	V_{TH} x 16/16
0	0	0	1	V_{TH} x 1/16
0	0	1	0	V_{TH} x 2/16
0	0	1	1	V_{TH} x 3/16
0	1	0	0	V_{TH} x 4/16
0	1	0	1	V_{TH} x 5/16
0	1	1	0	V_{TH} x 6/16
0	1	1	1	V_{TH} x 7/16
1	0	0	0	V_{TH} x 8/16
1	0	0	1	V_{TH} x 9/16
1	0	1	0	V_{TH} x 10/16
1	0	1	1	V_{TH} x 11/16
1	1	0	0	V_{TH} x 12/16
1	1	0	1	V_{TH} x 13/16
1	1	1	0	V_{TH} x 14/16
1	1	1	1	V_{TH} x 15/16

49-001381B

4b

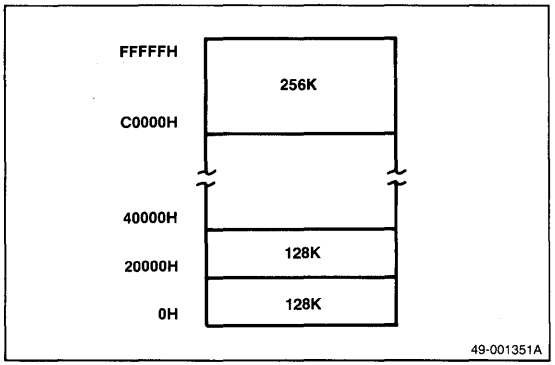
Programmable Wait State Generation

You can generate wait states internally to further reduce the necessity for external hardware. Insertion of these wait states allows direct interface to devices whose access times cannot meet the CPU read/write timing requirements.

When using this function, the entire 1M-byte memory address space is divided into 128K-blocks. Each block can be programmed for zero, one, or two wait states, or two plus those added by the extenal READY signal. The top two blocks are programmed together as one unit.

The appropriate bits in the wait control word (WTC) control wait state generation. Programming the upper two bits in the wait control word will set the wait state conditions for the entire I/O address space. Figure 33 shows the memory map for programmable wait state generation; see figure 34 for a graphic representation of the wait control word.

Figure 33. Programmable Wait State Generation



Standby Modes

The two low-power standby modes are HALT and STOP. Software causes the processor to enter either mode.

HALT Mode.

In the HALT mode, the processor is inactive and the chip consumes much less power than when operational. The external oscillator remains functional and all peripherals are active. Internal status and output port line conditions are maintained. Any unmasked interrupt can release this mode. In the EI state, interrupts subsequently will be processed in vector mode. In the DI state, program execution is restarted with the instruction following the HALT instruction.

STOP Mode.

The STOP mode allows the largest power reduction while maintaining RAM. The oscillator is stopped, halting all internal peripherals. Internal status is maintained. Only a reset or NMI can release this mode.

A standby flag in the SFR area is reset by rises in the supply voltage. Its status is maintained during normal operation and standby. The STBC register (figure 35) is not initialized by RESET. Use the standby flag to determine whether program execution is returning from standby or from a cold start by setting this flag before entering the STOP mode.

Special Function Registers

Table 6 shows the special function register mnemonic, type, address, reset value, and function. The 8 high-order bits of each address (xx) are specified by the IDB register.

SFR area addresses not listed in table 6 are reserved. If read, the contents of these addresses are undefined, and any write operation will be meaningless.

Figure 34. Wait Control Word

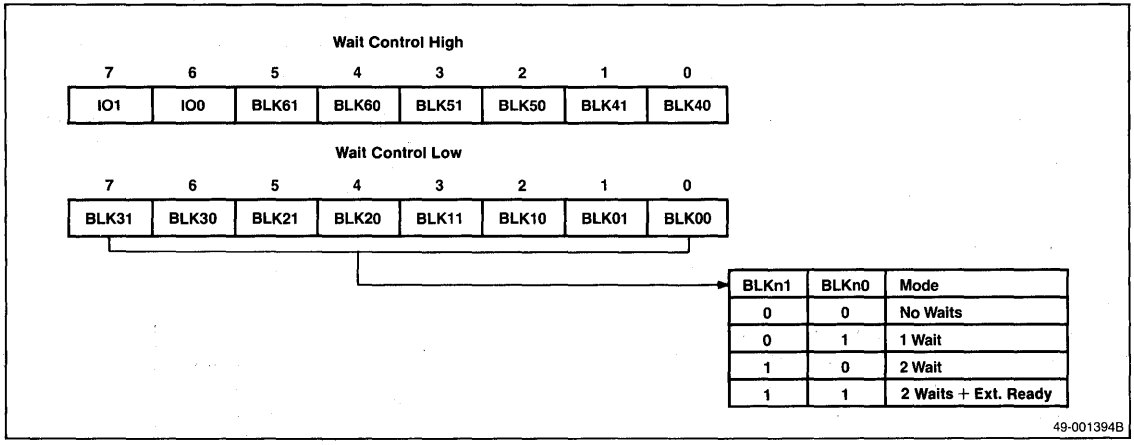


Figure 35. Standby Register

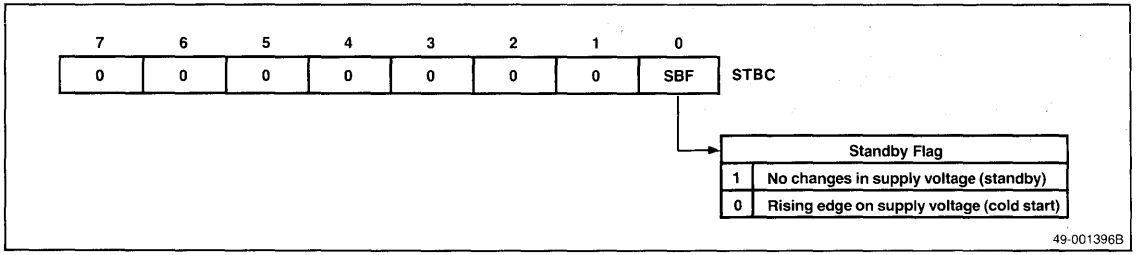


Table 6. Special-Function Registers

Address	Register Function	Symbol	R/W	Manipulation (Bit)	When RESET
xxF00H	Port 0	P0	R/W	8/1	Undefined
xxF01H	Port mode 0	PM0	W	8	FFH
xxF02H	Port mode control 0	PMC0	W	8	00H
xxF08H	Port 1	P1	R/W	8/1	Undefined
xxF09H	Port mode 1	PM1	W	8	FFH
xxF0AH	Port mode control 1	PMC1	W	8	00H
xxF10H	Port 2	P2	R/W	8/1	Undefined
xxF11H	Port mode 2	PM2	W	8	FFH
xxF12H	Port mode control 2	PMC2	W	8	00H
xxF38H	Port T	PT	R	8	Undefined
xxF3BH	Port mode T	PMT	R/W	8/1	00H
xxF40H	External interrupt mode	INTM	R/W	8/1	00H
xxF44H	External interrupt macro service control 0	EMS0	R/W	8/1	Undefined
xxF45H	External interrupt macro service control 1	EMS1	R/W	8/1	
xxF46	External interrupt macro service control 2	EMS2	R/W	8/1	
xxF4CH	External interrupt request control 0	EXIC0	R/W	8/1	47H
xxF4DH	External interrupt request control 1	EXIC1	R/W	8/1	
xxF4EH	External interrupt request control 2	EXIC2	R/W	8/1	
xxF60H	Receive buffer 0	RxB0	R	8	Undefined
xxF62H	Transmit buffer 0	TxB0	W	8	
xxF65H	Serial receive macro service control 0	SRMS0	R/W	8/1	
xxF66H	Serial transmit macro service control 0	STMS0	R/W	8/1	
xxF68H	Serial communication mode 0	SCM0	R/W	8/1	
xxF69H	Serial communication control 0	SCC0	R/W	8/1	00H
xxF6AH	Baud rate generator 0	BRG0	R/W	8/1	
xxF6BH	Serial communication error 0	SCE0	R	8	
xxF6CH	Serial error interrupt request control 0	SEIC0	R/W	8/1	47H
xxF6DH	Serial receive interrupt request control 0	SRIC0	R/W	8/1	
xxF6EH	Serial transmit interrupt request control 0	STIC0	R/W	8/1	
xxF70H	Receive buffer 1	RxB1	R	8	Undefined
xxF72H	Transmit buffer 1	TxB1	W	8	
xxF75H	Serial receive macro service control 1	SRMS1	R/W	8/1	
xxF76H	Serial transmit macro service control 1	STMS1	R/W	8/1	
xxF78H	Serial communication mode 1	SCM1	R/W	8/1	00H
xxF79H	Serial communication control 1	SCC1	R/W	8/1	
xxF7AH	Baud rate generator 1	BRG1	R/W	8/1	
xxF7BH	Serial communication error 1	SCE1	R	8	

Table 6. Special-Function Registers (cont)

Address	Register Function	Symbol	R/W	Manipulation (Bit)	When RESET
xxF7CH	Serial error interrupt request control 1	SEIC1	R/W	8/1	47H
xxF7DH	Serial receive interrupt request control 1	SRIC1	R/W	8/1	
xxF7EH	Serial transmit interrupt request control 1	STIC1	R/W	8/1	
xxF80H	Timer 0	TM0	R/W	16	Undefined
xxF82H	Modulo 0	MD0	R/W	16	
xxF88H	Timer 1	TM1	R/W	16	
xxF8AH	Modulo 1	MD1	R/W	16	
xxF90H	Timer control 0	TMC0	R/W	8/1	00H
xxF91H	Timer control 1	TMC1	R/W	8/1	
xxF94H	Timer macro service control 0	TMMS0	R/W	8/1	Undefined
xxF95H	Timer macro service control 1	TMMS1	R/W	8/1	
xxF96H	Timer macro service control 2	TMMS2	R/W	8/1	
xxF9CH	Timer interrupt request control 0	TMIC0	R/W	8/1	47H
xxF9DH	Timer interrupt request control 1	TMIC1	R/W	8/1	
xxF9EH	Timer interrupt request control 2	TMIC2	R/W	8/1	
xxFA0H	DMA control 0	DMAC0	R/W	8/1	Undefined
xxFA1H	DMA mode 0	DMAM0	R/W	8/1	00H
xxFA2H	DMA control 1	DMAC1	R/W	8/1	Undefined
xxFA3H	DMA mode 1	DMAM1	R/W	8/1	
xxFACH	DMA interrupt request control 0	DIC0	R/W	8/1	47H
xxFADH	DMA interrupt request control 1	DIC1	R/W	8/1	
xxFE0H	Standby control	STBC	R/W (Note 1)	8/1	Undefined (Note 2)
xxFE1H	Refresh mode	RFM	R/W	8/1	FCH
xxFE8H	Wait control	WTC	R/W	16/8	FFFFH
xxFEAH	User flag (Note 3)	FLAG	R/W	8/1	00H
xxFEBH	Processor control	PRC	R/W	8/1	4EH
xxFECH	Time base interrupt request control	TBIC	R/W	8/1	47H
xxFFCH	Inservice priority register	ISPR	R/W	8/1	Undefined
xxFFFH FFFFFH	Internal data area base	IDB	R/W	8/1	FFH

Notes:

(1) Each bit of the standby control register can be set to 1 by an instruction; however, once set, bits cannot be reset to 0 by an instruction (only 1 can be written to this register).

(2) Upon power-on reset = 00H; other = no change.

(3) For the user flag register (FLAG), manipulating bits other than bits 3 and 5 is meaningless. The contents of user flags 0 and 1 (F0 and F1) of the FLAG register are affected by manipulating F0 and F1 of the PSW.

Absolute Maximum Ratings

$T_A = 25°C$

Supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Output voltage, V_O	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Threshold voltage, V_{TH}	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Output current, low; I_{OL}	
Each output pin	4.0 mA
Total	50 mA
Output current, high; I_{OH}	
Each output pin	−2.0 mA
Total	−20 mA
Operating temperature range, T_{OPT}	−40 to +85°C
Storage temperature range, T_{STG}	−65 to +150°C

Comment: Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Comparator Characteristics

$V_{DD} = +5$ V ±10%; $T_A = -10$ to +70°C

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Accuracy	VA_{COMP}		±100	mV	
Threshold voltage	V_{TH}	0	V_{DD} + 0.1	V	
Comparison time	t_{COMP}	64	65	t_{CYK}	
PT input voltage	V_{IPT}	0	V_{DD}	V	

Capacitance Characteristics

$V_{DD} = 0$ V; $T_A = 25°C$

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input capacitance	C_I		10	pF	$f_c = 1$ MHz; Unmeasured pins returned to 0 V
Output capacitance	C_O		20	pF	
I/O capacitance	C_{IO}		20	pF	

DC Characteristics

$V_{DD} = +5$ V ±10%; $T_A = -10$ to +70°C (Note 1)

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Supply current, operating	I_{DD1}		50 65	100 120	mA mA	$f_{CLK} = 5$ MHz $f_{CLK} = 8$ MHz
Supply current, HALT mode	I_{DD2}		20 25	40 50	mA mA	$f_{CLK} = 5$ MHz $f_{CLK} = 8$ MHz
Supply current, STOP mode	I_{DD3}		10	30	μA	
Threshold current	I_{TH}		0.5	1.0	mA	$V_{TH} = 0$ to V_{DD}
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2		V_{DD}	V	All inputs except RESET, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 x V_{DD}		V_{DD}	V	RESET, $P1_0$/NMI, X1, X2
Output voltage, low	V_{OL}			0.45	V	$I_{OL} = 1.6$ mA
Output voltage, high	V_{OH}	V_{DD} − 1.0			V	$I_{OH} = -0.4$ mA
Input current	I_{IN}			±20	μA	$\overline{EA}$, $P1_0$/NMI; $V_I = 0$ to V_{DD}
Input leakage current	I_{LI}			±10	μA	All except $\overline{EA}$, $P1_0$/NMI; $V_I = 0$ to V_{DD}
Output leakage current	I_{LO}			±10	μA	$V_O = 0$ to V_{DD}
Data retention voltage	V_{DDDR}	2.5		5.5	V	

Notes:

(1) The standard operating temperature range is −10 to +70°C. However, extended temperature range parts (−40 to +85°C) are available.

4b

AC Characteristics

V_{DD} = +5 V ±10%; T_A = −10 to +70 °C; C_L = 100 pF (max)

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
V_{DD} rise, fall time	t_{RVD}, t_{FVD}	200		μs	STOP mode
Input rise, fall time	t_{IR}, t_{IF}		20	ns	Except X1, X2, $\overline{RESET}$, NMI
Input rise, fall time	t_{IRS}, t_{IFS}		30	ns	$\overline{RESET}$, NMI (Schmitt)
Output rise, fall time	t_{OR}, t_{OF}		20	ns	Except CLKOUT
X1 cycle time	t_{CYX}	98	250	ns	Note 3
		62	250	ns	Note 4
X1 width, low	t_{WXL}	35		ns	Note 3
		20		ns	Note 4
X1 width, high	t_{WXH}	35		ns	Note 3
		20		ns	Note 4
X1 rise, fall time	t_{XR}, t_{XF}		20	ns	
CLKOUT cycle time	t_{CYK}	200	2000	ns	Note 3
		125	2000	ns	Note 4
CLKOUT width, low	t_{WKL}	0.5T − 15		ns	Note 1
CLKOUT width, high	t_{WKH}	0.5T − 15		ns	
CLKOUT rise, fall time	t_{KR}, t_{KF}		15	ns	
Address delay time	t_{DKA}		90	ns	
Address valid to input data valid	t_{DADR}		T(n + 1.5) − 90	ns	Note 2
$\overline{MREQ}$ to address hold time	t_{HMRA}	0.5T − 30		ns	
$\overline{MREQ}$ to data delay	t_{DMRD}		T(n + 2) − 75	ns	
$\overline{MSTB}$ to data delay	t_{DMSD}		T(n + 1) − 75	ns	
$\overline{MREQ}$ to $\overline{MSTB}$ delay	t_{DMRMSR}	T − 35	T + 35	ns	
$\overline{MREQ}$ width, low	t_{WMRL}	T(n + 2) − 30		ns	
$\overline{MREQ}$, $\overline{MSTB}$ to address hold time	t_{HMA}	0.5T − 50		ns	
Input data hold time	t_{HMD}	0		ns	
Next control setup time	t_{SCC}	T − 25		ns	
$\overline{MREQ}$ to TC delay time	t_{DMRTC}		0.5T + 50	ns	

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
$\overline{MREQ}$ delay time	t_{DAMR}	0.5T − 30		ns	
$\overline{MSTB}$ delay time	t_{DAMSR}	T − 30		ns	
$\overline{MSTB}$ width, low	t_{WMSLR}	T(n + 1) − 30		ns	
Address data output	t_{DADW}		0.5T + 50	ns	
Data output setup time	t_{SDM}	T(n + 2) − 50		ns	
$\overline{MSTB}$ write delay time	t_{DAMSW}	T(n + 0.5) − 30		ns	
$\overline{MREQ}$ to $\overline{MSTB}$ write delay time	t_{DMRMSW}	T(n + 1) − 35		ns	
$\overline{MSTB}$ write width low	t_{WMSLW}	T − 30		ns	
Data output hold time	t_{HMDW}	0.5T − 50		ns	
$\overline{IOSTB}$ delay time	t_{DAIS}	0.5T − 30		ns	
$\overline{IOSTB}$ to data input	t_{DISD}		T(n + 1) − 90	ns	
$\overline{IOSTB}$ width, low	t_{WISL}	T(n + 1) − 30		ns	
Address hold time	t_{HISA}	0.5T − 30		ns	
Input data hold time	t_{HISDR}	0		ns	
Output data setup time	t_{SDIS}	T(n + 1) − 50		ns	
Output data hold time	t_{HISDW}	0.5T − 30		ns	
Next DMARQ setup time	t_{SDADQ}		T	ns	Demand mode
DMARQ hold time	t_{HDARQ}	0		ns	Demand mode
$\overline{DMAAK}$ read width, low	t_{WDMRL}	T(n + 2.5) − 30		ns	
$\overline{DMAAK}$ write width, low	t_{WDMWL}	T(n + 2) − 30		ns	
$\overline{DMAAK}$ to TC delay time	t_{DDATC}		0.5T + 50	ns	
$\overline{TC}$ width, low	t_{WTCL}	2T − 30		ns	
$\overline{REFRQ}$ delay time	t_{DARF}	0.5T − 30		ns	
$\overline{REFRQ}$ width, low	t_{WRFL}	T(n + 2) − 30		ns	
Address hold time	t_{HRFA}	0.5T − 30		ns	

AC Characteristics (cont)

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
$\overline{RESET}$ width low	t_{WRSL1}	30		ms	STOP/ POR (Power-on reset)
	t_{WRSL2}	5		μs	System reset
$\overline{MREQ}$, $\overline{IOSTB}$ to READY setup time	t_{SCRY}		T(n − 1) − 100	ns	n ≥ 2
$\overline{MREQ}$, $\overline{IOSTB}$ to READY hold time	t_{HCRY}	T(n)		ns	n ≥ 2
Ready setup time	t_{SRYK}	20		ns	
Ready hold time	t_{HKRY}	40		ns	
HLDRQ setup time	t_{SHQK}	30		ns	
$\overline{HLDAK}$ output delay	t_{DKHA}		80	ns	
Bus control float to $\overline{HLDAK}$↓	t_{CFHA}	T − 50		ns	
$\overline{HLDAK}$↑ to control output time	t_{DHAC}	T − 50		ns	
HLDRQ to $\overline{HLDAK}$ delay	t_{DHQHA}		3T + 160	ns	
HLDRQ↓ to control float	t_{DHQC}	3T + 30		ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
$\overline{HLDAK}$ width, low	t_{WHAL}		T	ns	
$\overline{INTP}$, DMARQ setup	t_{SIQK}	30		ns	
$\overline{INTP}$, DMARQ width, high	t_{WIQH}	8T		ns	
$\overline{INTP}$, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{POLL}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{CTS}$ width, low	t_{WCTL}	2T		ns	
INTR setup time	t_{SIRK}	30		ns	
$\overline{INTAK}$ delay time	t_{DKIA}		80	ns	
INTR hold time	t_{HIAIQ}	0		ns	
$\overline{INTAK}$ width, low	t_{WIAL}	2T − 30		ns	
$\overline{INTAK}$ width, high	t_{WIAH}	T − 30		ns	
$\overline{INTAK}$ to data delay	t_{DIAD}		2T − 130	ns	
$\overline{INTAK}$ to data hold	t_{HIAD}	0	0.5T	ns	
$\overline{SCKO}$ (TSCK) cycle time	t_{CYTK}	1000		ns	
$\overline{SCKO}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{SCKO}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	
$\overline{CTS0}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{CTS0}$ (RSCK) width, high	t_{WSRH}	420		ns	
$\overline{CTS0}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

Notes:

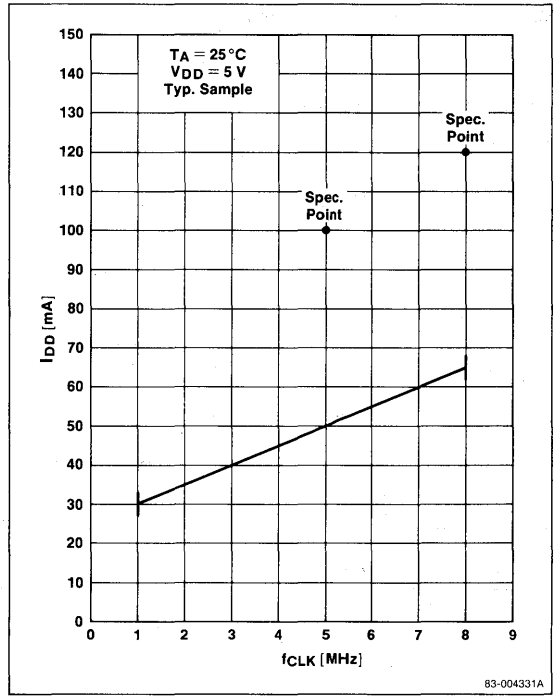
(1) T = CPU clock period (t_{CYK}).

(2) n = number of wait states inserted.

(3) For 5 MHz parts (μPD70320/322).

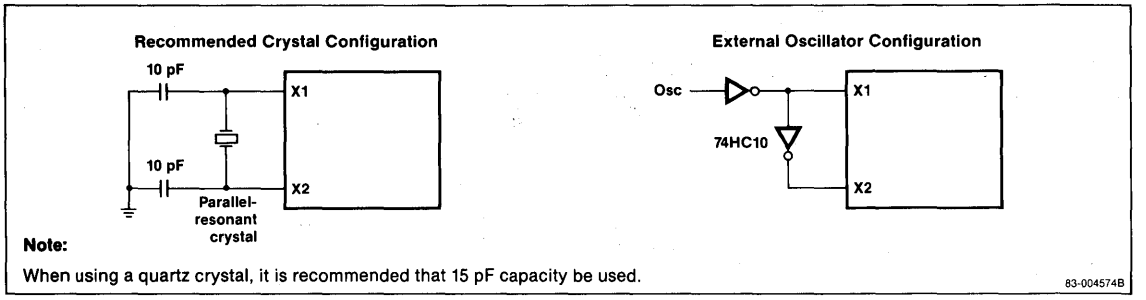
(4) For 10 MHz parts (μPD70320/322-8).

Supply Current vs Clock Frequency



4b

Figure 36. External System Clock Control Source



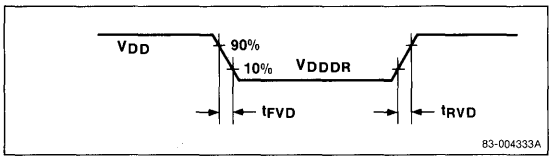
Note:
When using a quartz crystal, it is recommended that 15 pF capacity be used.

Resonator and Capacitance Requirements

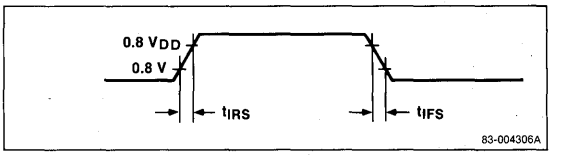
Manufacturer	Product Number	Recommended C1 (pF)	Constants C2 (pF)	Product Number	Recommended C1 (pF)	Constants C2 (pF)
Kyocera	KBR-10.0M	33	33			
Murata Manufacturing	CSA.10.0MT	47	47	CSA16.0MX040	30	30
TDK	FCR10.0M2S	30	30	FCR16.0M2S	15	6

Timing Waveforms

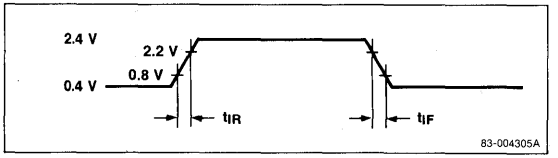
Stop Mode Data Retention Timing



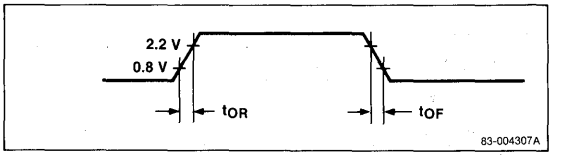
AC Input Waveform 2 ($\overline{RESET}$, NMI)



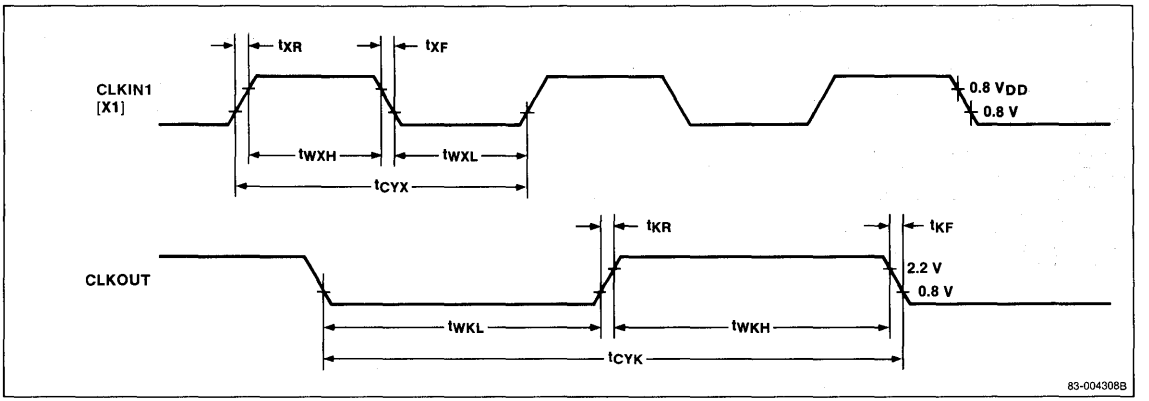
AC Input Waveform 1 (Except X1, X2, $\overline{RESET}$, NMI)



AC Output Test Point (Except CLKOUT)

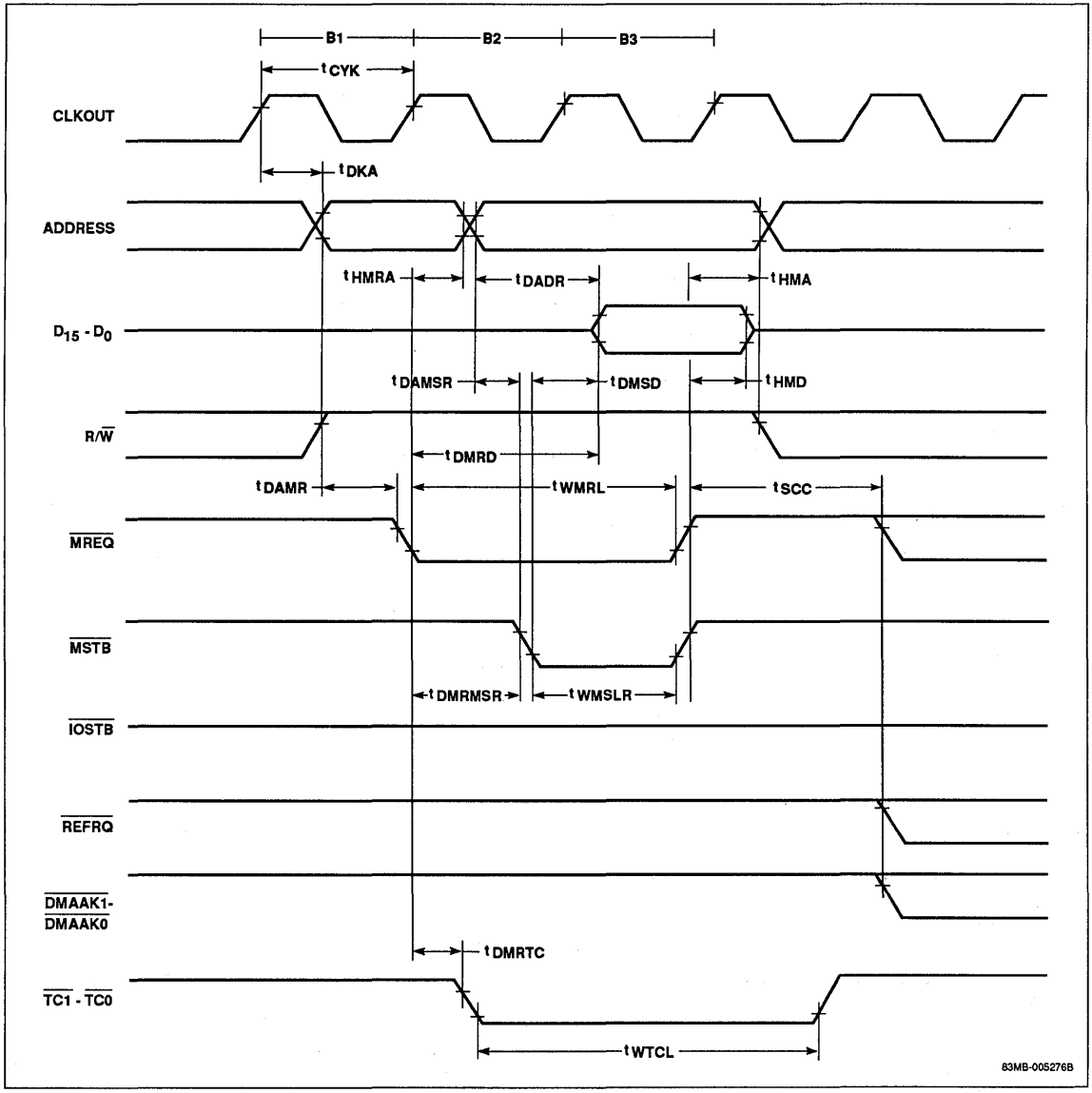


Clock In and Clock Out



Timing Waveforms (cont)

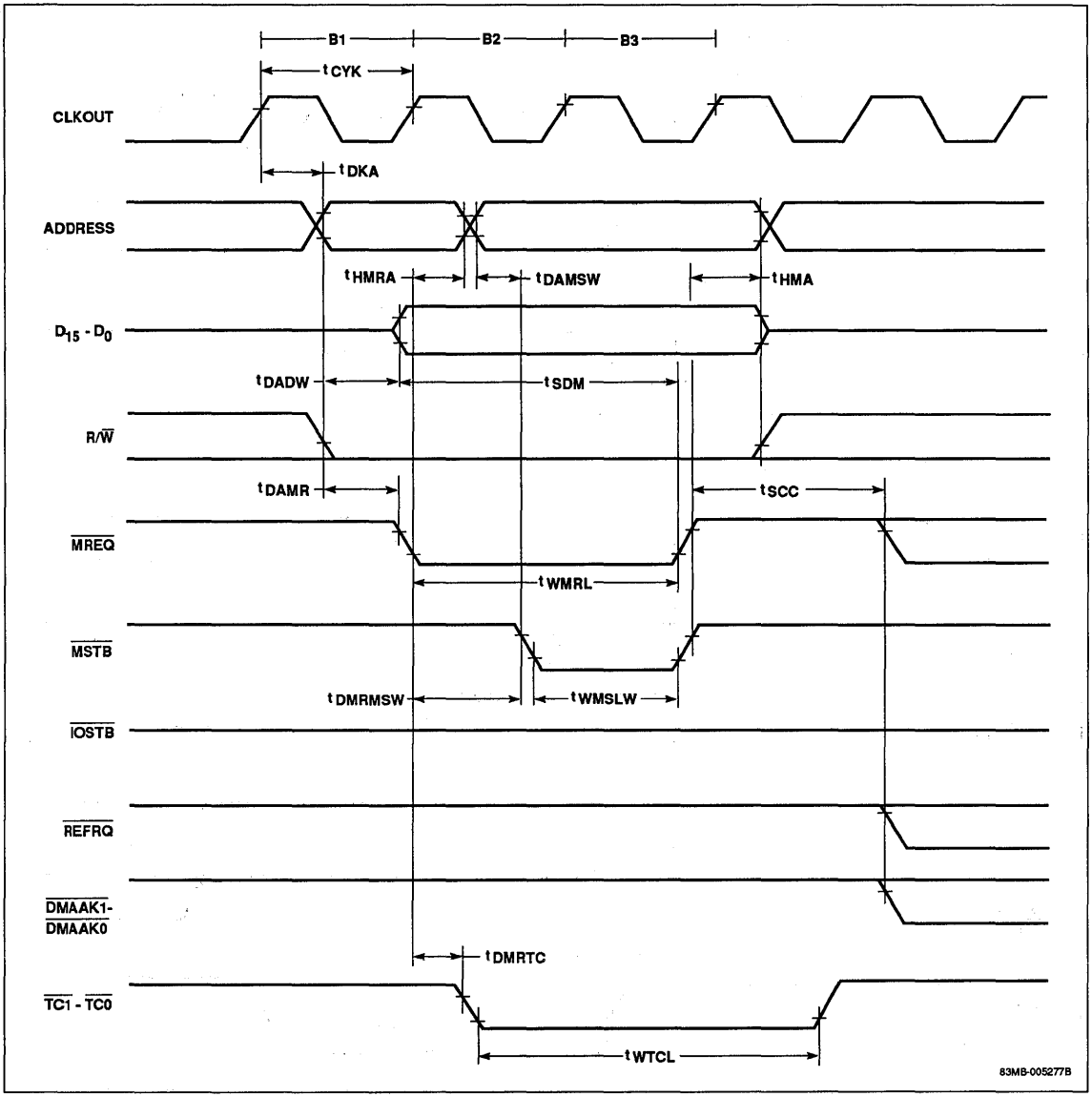
Memory Read



4b

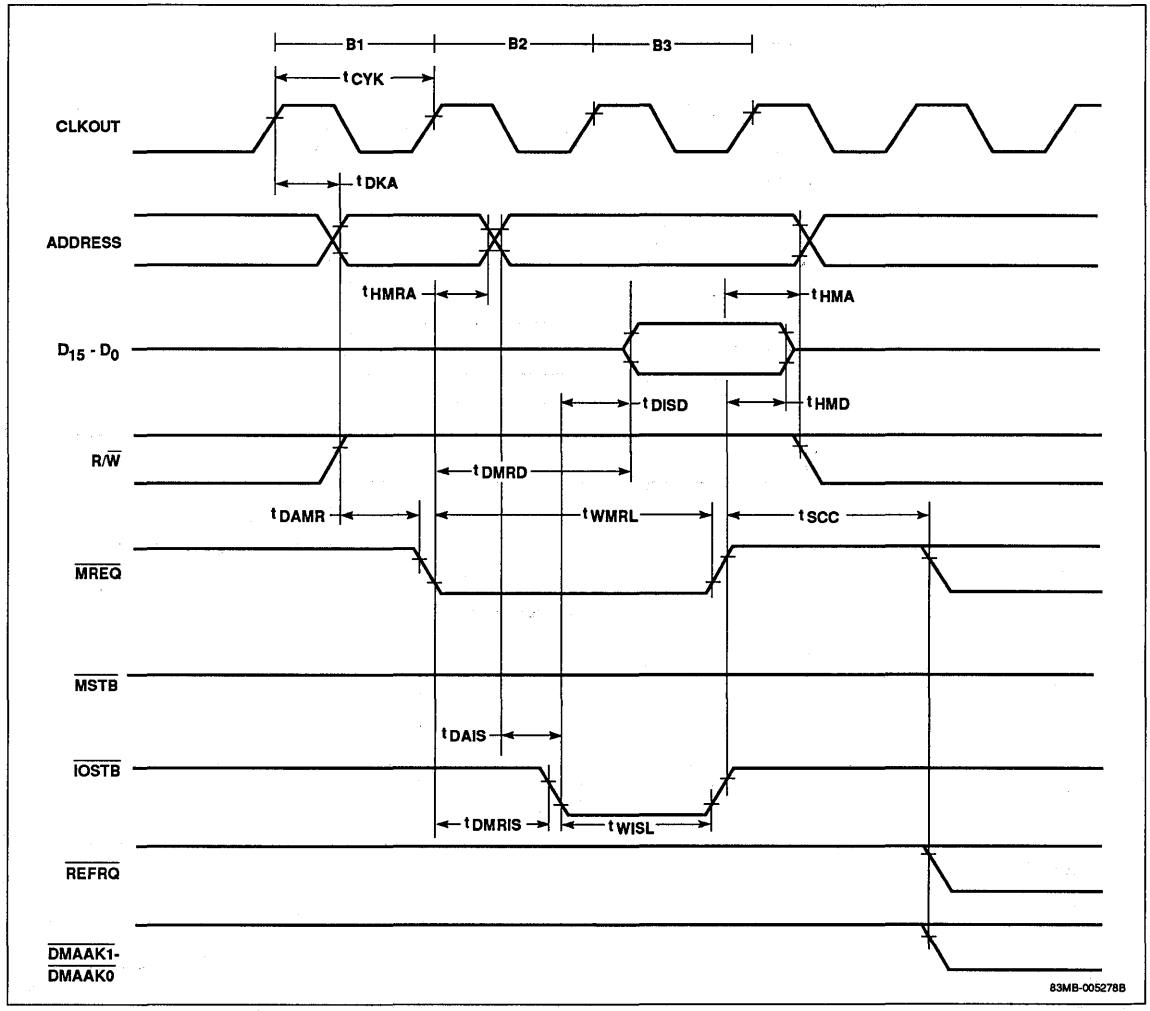
Timing Waveforms (cont)

Memory Write



Timing Waveforms (cont)

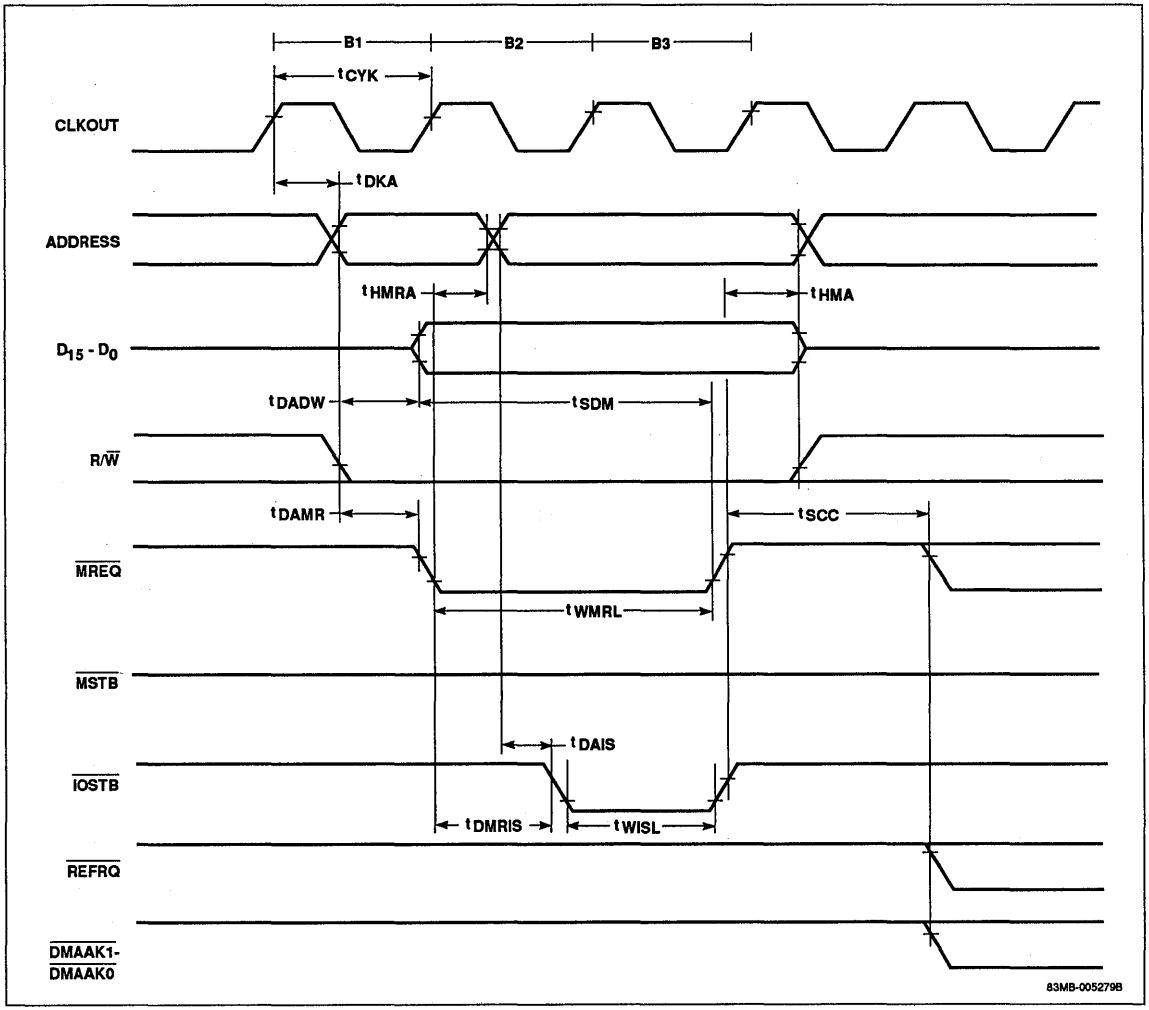
I/O Read



4b

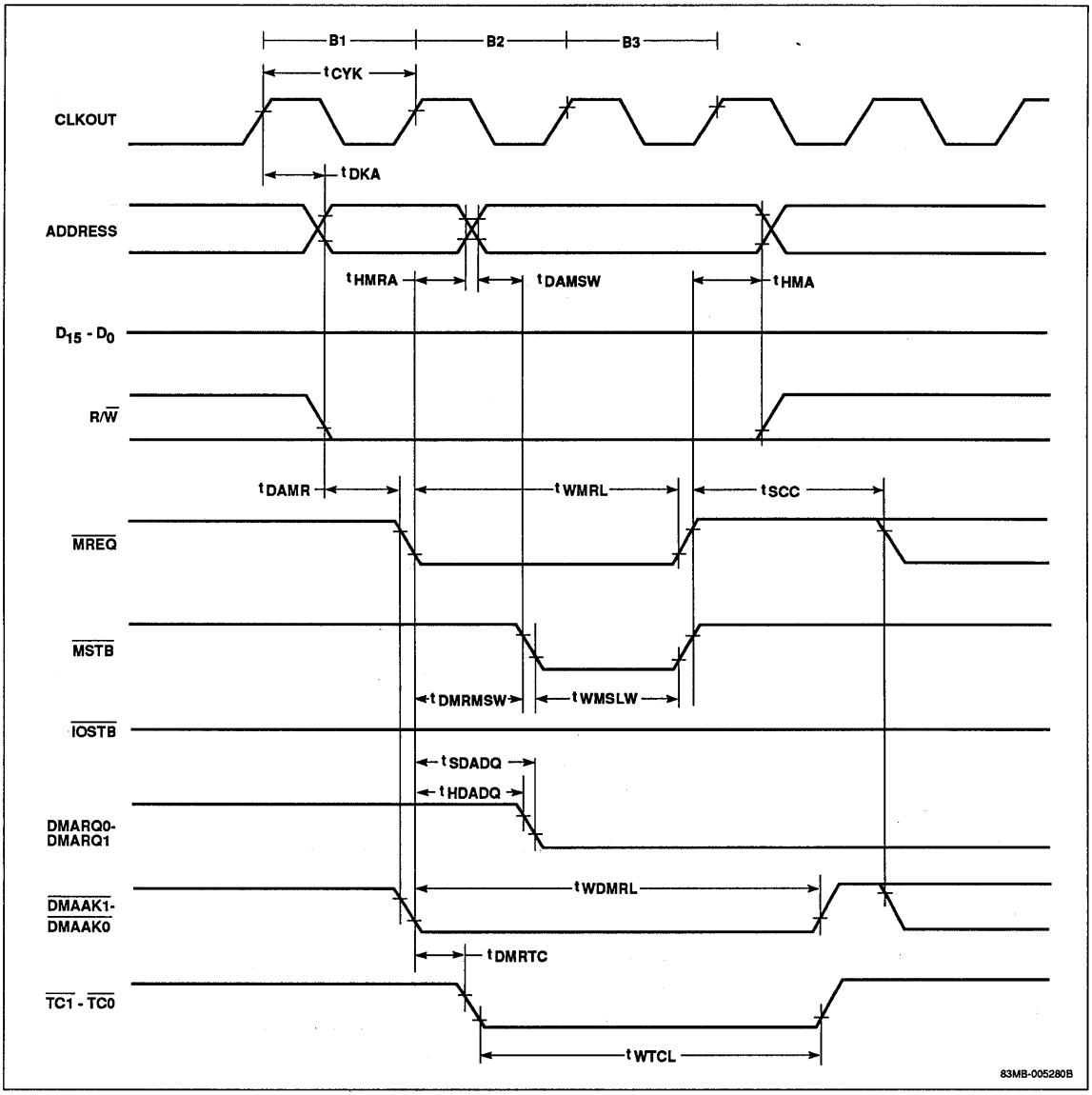
Timing Waveforms (cont)

I/O Write



Timing Waveforms (cont)

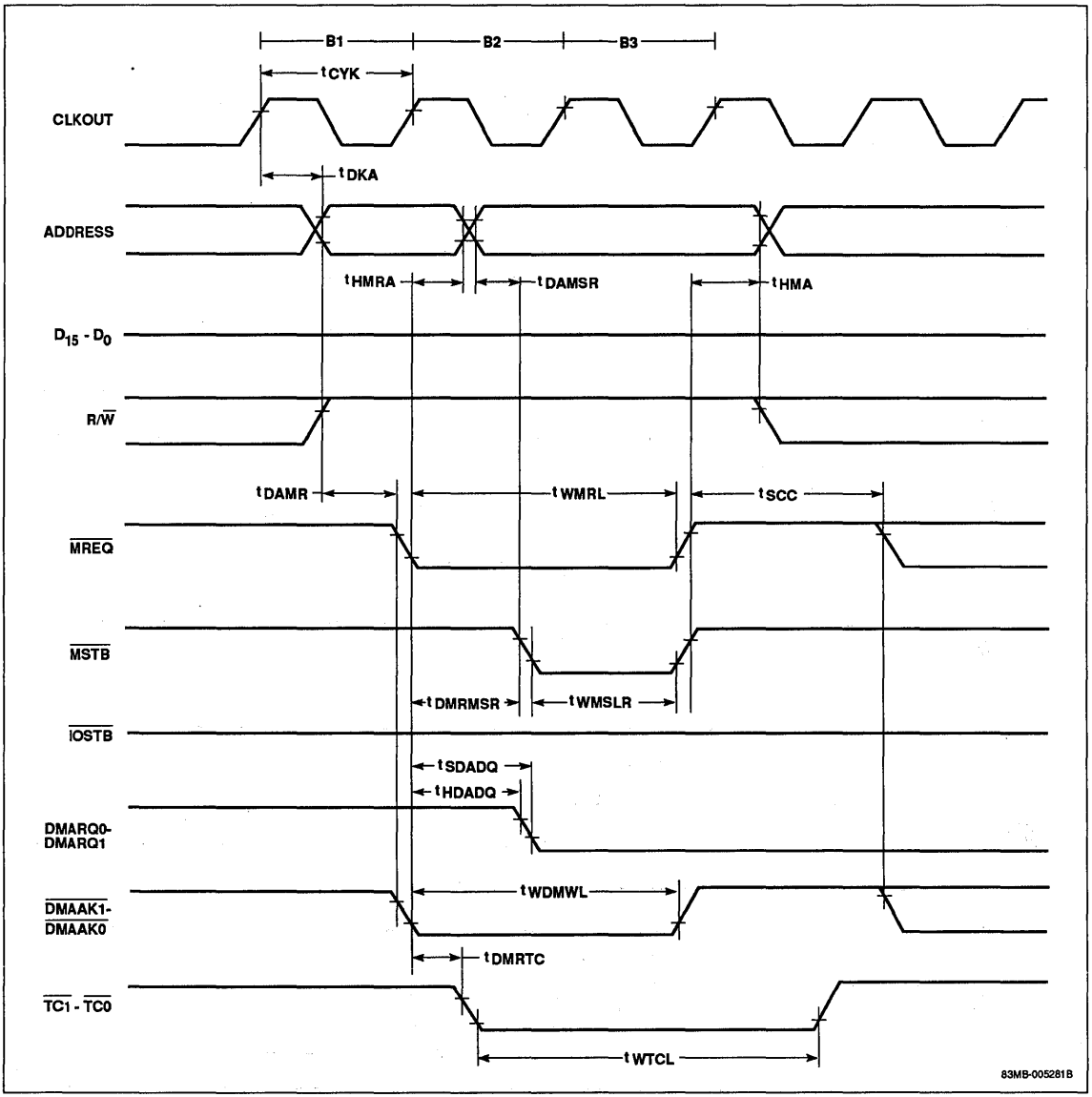
DMA, I/O to Memory



4b

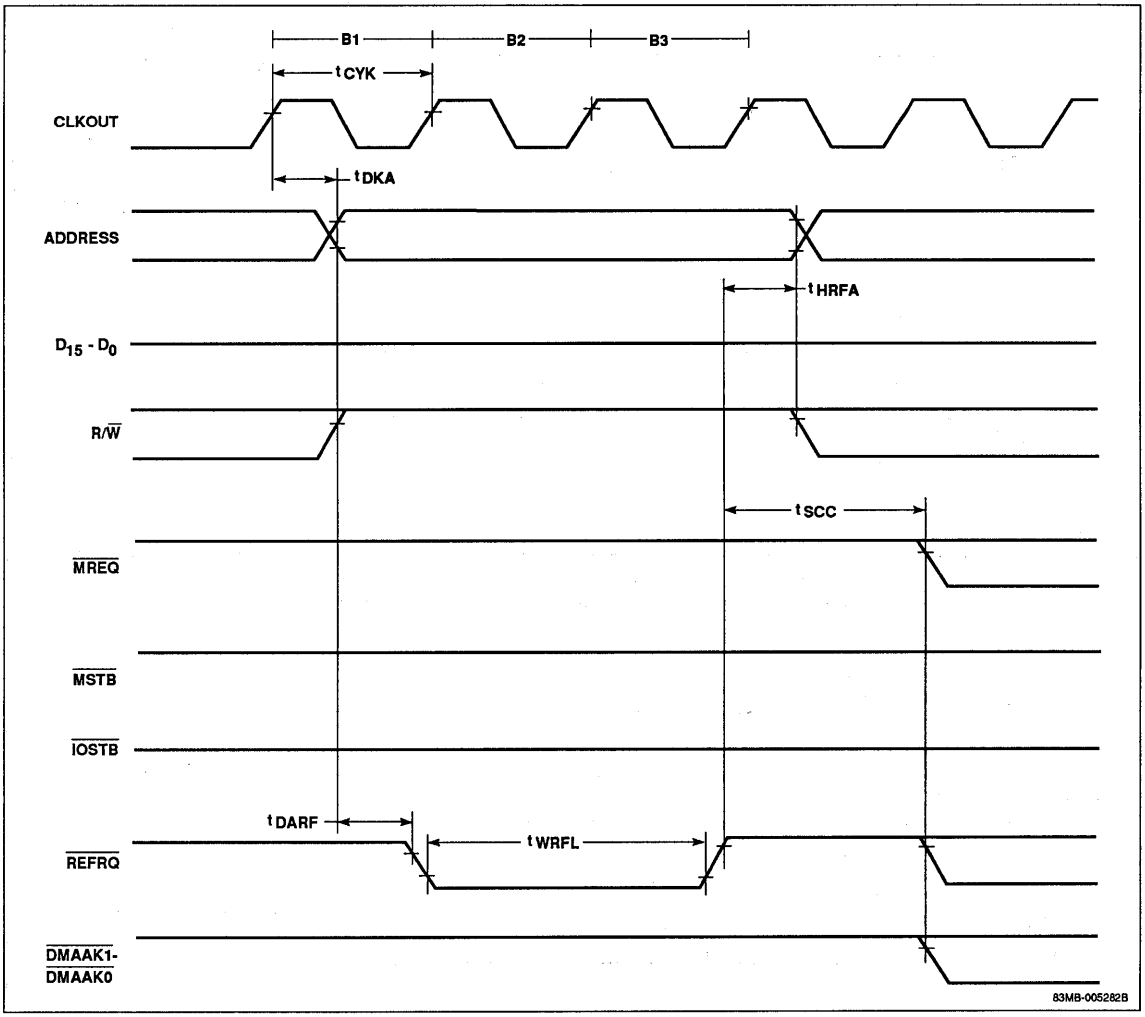
Timing Waveforms (cont)

DMA, Memory to I/O



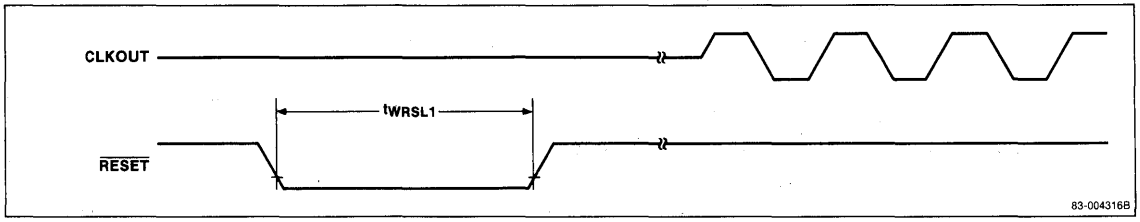
Timing Waveforms (cont)

Refresh

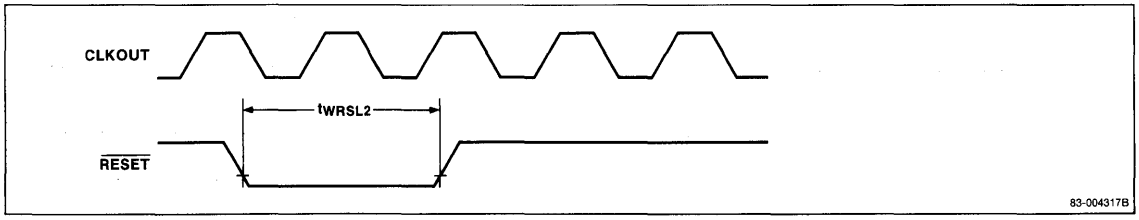


Timing Waveforms (cont)

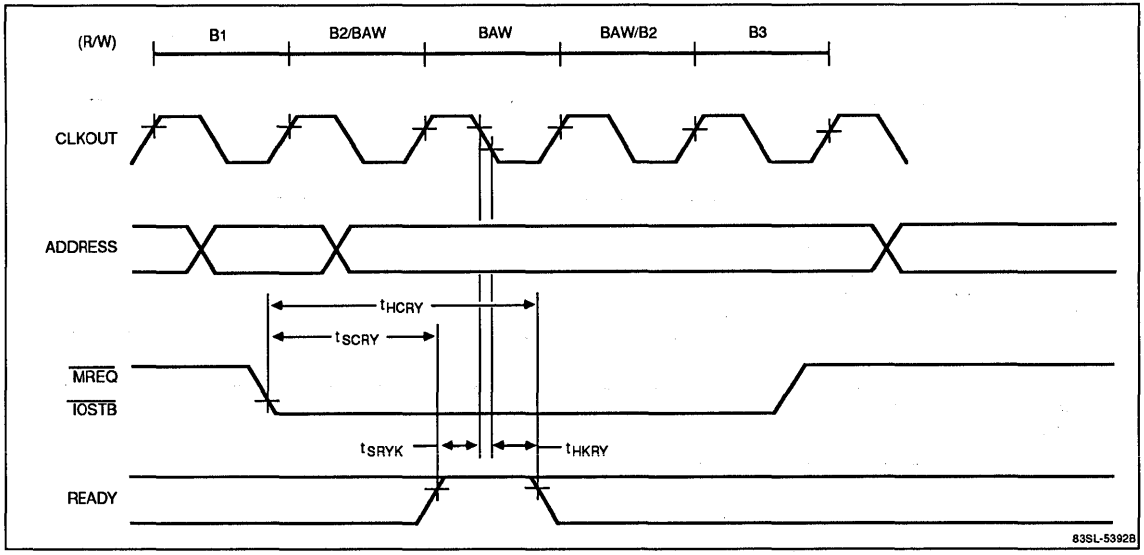
$\overline{RESET}$ 1



$\overline{RESET}$ 2

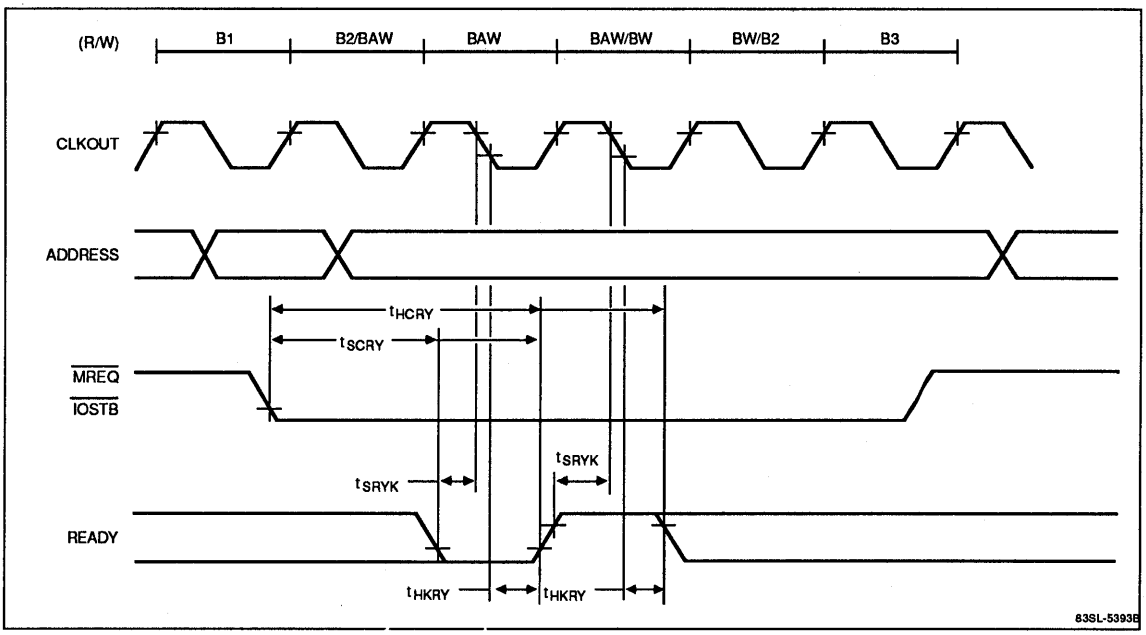


READY Timing 1

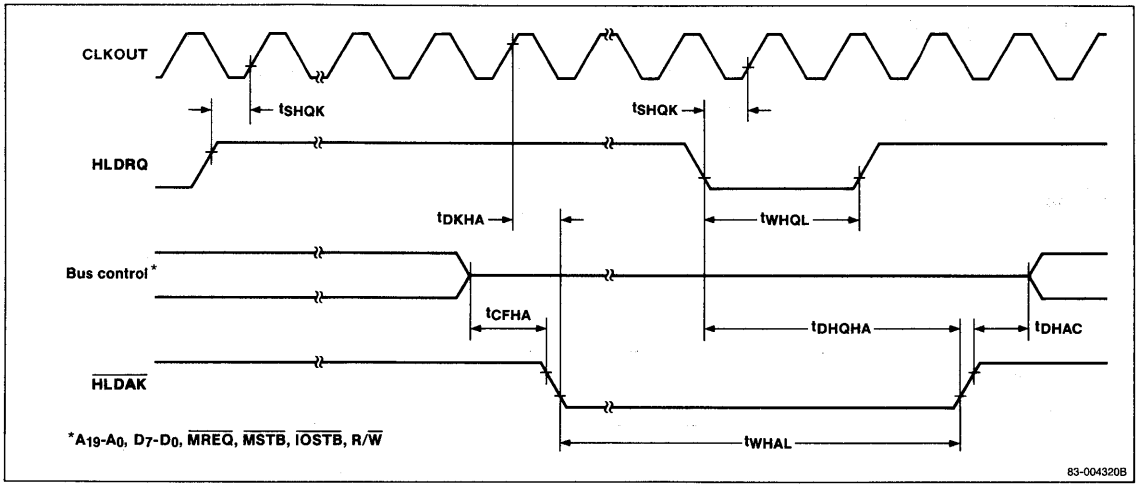


Timing Waveforms (cont)

READY Timing 2



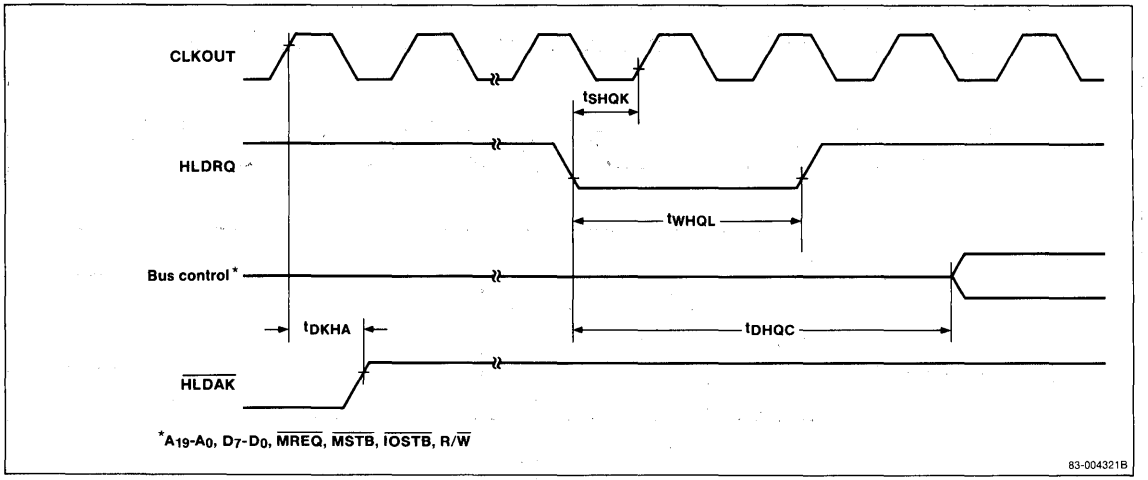
HLDRQ/HLDAK 1



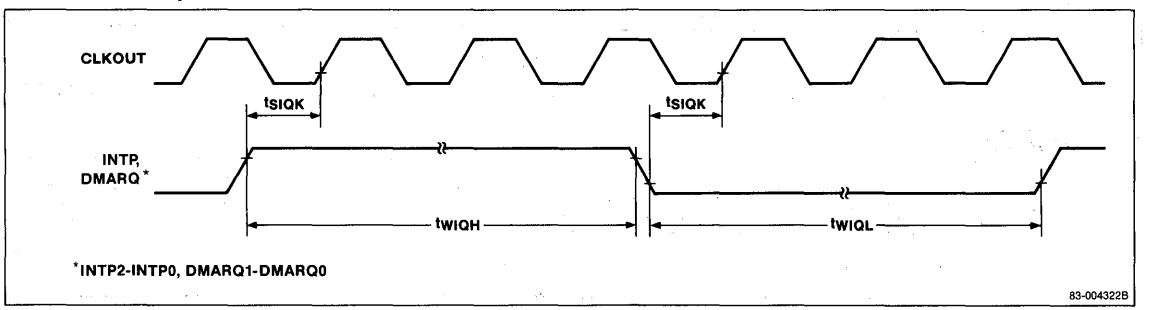
4b

Timing Waveforms (cont)

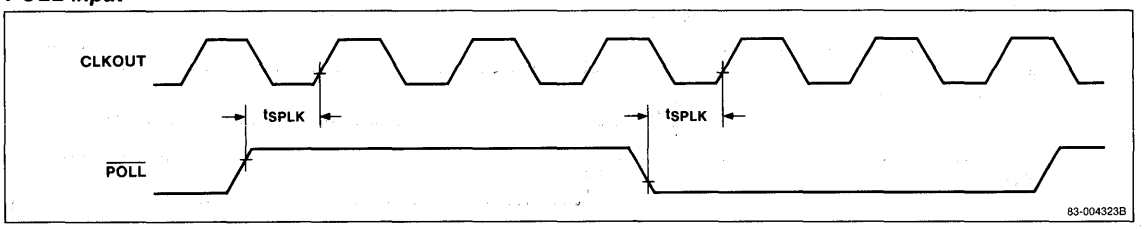
HLDRQ/$\overline{\text{HLDAK}}$ 2



INTP, DMARQ Input

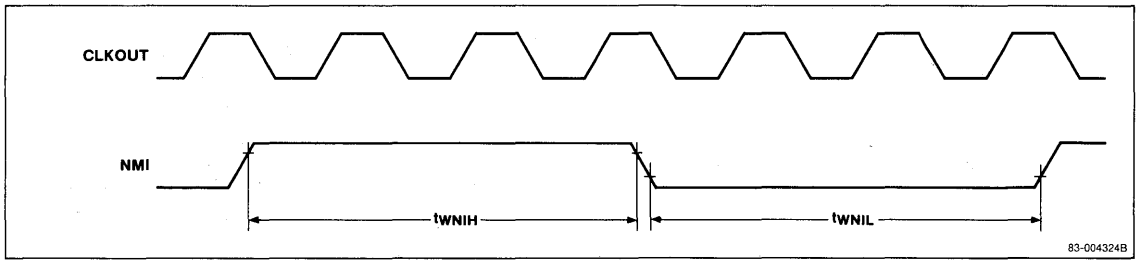


$\overline{\text{POLL}}$ Input

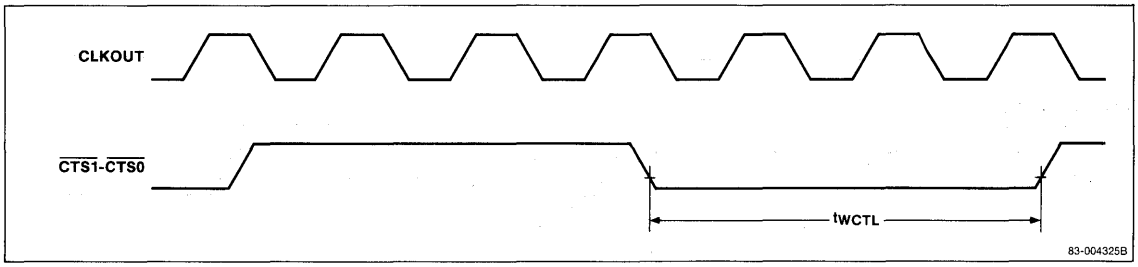


Timing Waveforms (cont)

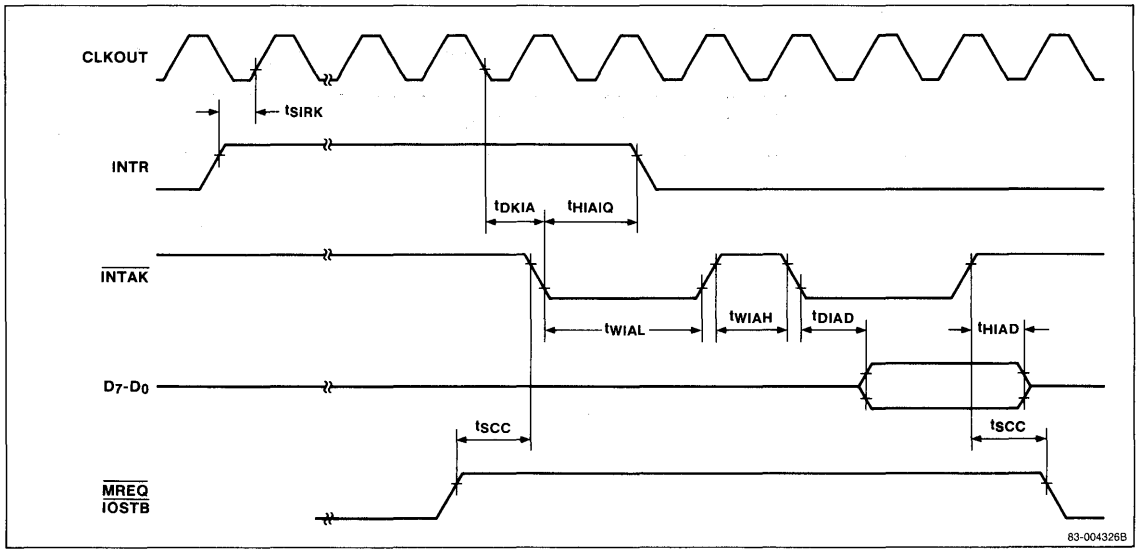
NMI Input



$\overline{CTS}$ Input



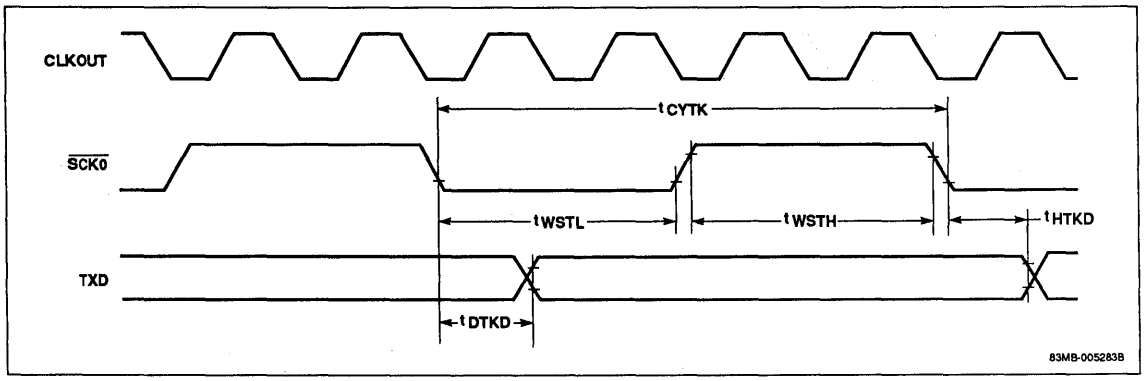
INTR/$\overline{INTAK}$



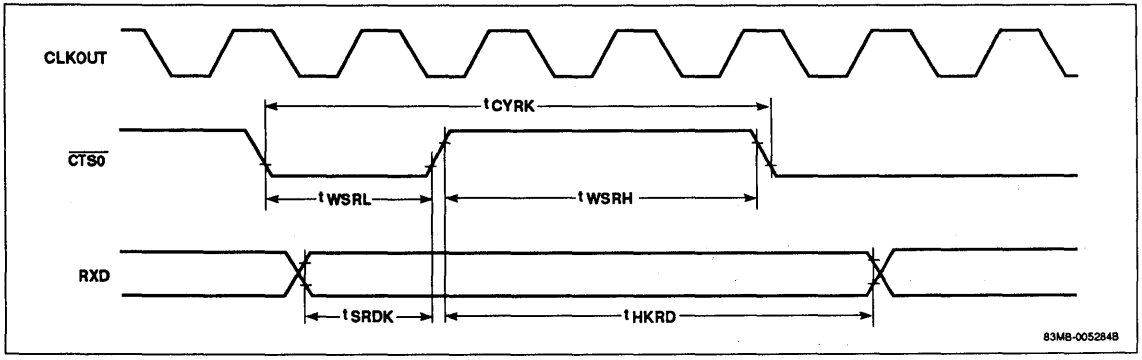
4b

Timing Waveforms (cont)

Serial Transmit



Serial Receive



Instruction Set

Instructions, grouped according to function, are described in a table near the end of this data sheet. Descriptions include source code, operation, opcode, number of bytes, and flag status. Supplementary information applicable to the instruction set is contained in the following tables.

- Symbols and Abbreviations
- Flag Symbols
- 8- and 16-Bit Registers. When mod = 11, the register is specified in the operation code by the byte/word operand (W = 0/1) and reg (000 to 111).
- Segment Registers. The segment register is specified in the operation code by sreg (00, 01, 10, or 11).
- Memory Addressing. The memory addressing mode is specified in the operation code by mod (00, 01, or 10) and mem (000 through 111).
- Instruction Clock Count. This table gives formulas for calculating the number of clock cycles occupied by each type of instruction. The formulas, which depend on byte/word operand and RAM enable/disable, have variables such as EA (effective address), W (wait states), and n (iterations or string instructions).

Symbols and Abbreviations

Identifier	Description
reg	8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
dmem	8- or 16-bit direct memory location
mem	8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
sfr	8-bit special function register location
imm	Constant (0 to FFFFH)
imm16	Constant (0 to FFFFH)
imm8	Constant (0 to FFH)
imm4	Constant (0 to FH)
imm3	Constant (0 to 7)
acc	AW or AL register
sreg	Segment register
src-table	Name of 256-byte translation table
src-block	Name of block addressed by the IX register
dst-block	Name of block addressed by the IY register
near-proc	Procedure within the current program segment
far-proc	Procedure located in another program segment
near-label	Label in the current program segment
short-label	Label between −128 and +127 bytes from the end of instruction
far-label	Label in another program segment
memptr16	Word containing the offset of the memory location within the current program segment to which control is to be transferred
memptr32	Double word containing the offset and segment base address of the memory location to which control is to be transferred
regptr16	16-bit register containing the offset of the memory location within the program segment to which control is to be transferred
pop-value	Number of bytes of the stack to be discarded (0 to 64K bytes, usually even addresses)
fp-op	Immediate data to identify the instruction code of the external floating point operation
R	Register set
W	Word/byte field (0 to 1)
reg	Register field (000 to 111)
mem	Memory field (000 to 111)
mod	Mode field (00 to 10)
S:W	When S:W = 01 or 11, data = 16 bits. At all other times, data = 8 bits.
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
AW	Accumulator (16 bits)
AH	Accumulator (high byte)
AL	Accumulator (low byte)
BP	Base pointer register (16 bits)
BW	BW register (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
CW	CW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
DW	DW register (16 bits)
DH	DW register (high byte)
DL	DW register (low byte)
SP	Stack pointer (16 bits)
PC	Program counter (16 bits)
PSW	Program status word (16 bits)

Symbols and Abbreviations (cont)

Identifier	Description
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
PS	Program segment register (16 bits)
SS	Stack segment register (16 bits)
DS_0	Data segment 0 register (16 bits)
DS_1	Data segment 1 register (16 bits)
AC	Auxiliary carry flag
CY	Carry flag
P	Parity flag
S	Sign flag
Z	Zero flag
DIR	Direction flag
IE	Interrupt enable flag
V	Overflow flag
BRK	Break flag
MD	Mode flag
(...)	Values in parentheses are memory contents
disp	Displacement (8 or 16 bits)
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1-bit)
seg	Immediate segment data (16 bits)
offset	Immediate offset data (16 bits)
←	Transfer direction
+	Addition
−	Subtraction
x	Multiplication
÷	Division
%	Modulo
AND	Logical product
OR	Logical sum
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value

Flag Symbols

Identifier	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
X	Set or cleared according to the result
U	Undefined
R	Value saved earlier is restored

8- and 16-Bit Registers (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Registers

sreg	Register
00	DS_1
01	PS
10	SS
11	DS_0

Memory Addressing

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Instruction Clock Count

Mnemonic	Operand	Clocks
ADD	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+10+2W [EA+7+W]
	mem16, reg16	EA+10+2W [EA+7+W]
	reg8, imm8	5
	reg16, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+11+2W [EA+9+2W]
	mem16, imm8	EA+9+2W [EA+7+2W]
	mem16, imm16	EA+12+2W [EA+8+2W]
	AL, imm8	5
	AW, imm16	6
ADD4S		22+(30+3W)n [22+(28+3W)n]
ADDC		Same as ADD
ADJ4A		9
ADJ4S		9
ADJBA		17
ADJBS		17
AND	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+10+2W [EA+7+W]
	mem16, reg16	EA+10+2W [EA+7+W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+11+2W [EA+9+2W]
	mem16, imm16	EA+12+2W [EA+8+2W]
Bcond (conditional branch)		8 or 15
BCWZ		8 or 15
BR	near-label	12
	short-label	12
	regptr16	13
	memptr16	EA+16+W
	far-label	15
	memptr32	EA+23+2W

Notes:

(1) If the number of clocks is not the same for RAM enabled and RAM disabled conditions, the RAM enabled value is listed first, followed by the RAM disabled value in brackets; for example, EA+8+2W [EA+6+W].

(2) Symbols in the Clocks column are defined as follows.

EA = additional clock cycles required for calculation of the effective address

= 3 (mod 00 or 01) or 4 (mod 10)

W = number of wait states selected by the WTC register

n = number of iterations or string instructions

Mnemonic	Operand	Clocks
BRK	3	50+5W [38+5W]
	imm8	51+5W [39+5W]
BRKCS		15
BRKV		50+5W [38+5W]
BTCLR		29
BUSLOCK		2
CALL	near-proc	21+W [17+W]
	regptr16	21+W [17+W]
	memptr16	EA+24+2W [EA+22+2W]
	far-proc	36+2W [32+2W]
	memptr32	EA+32+4W [EA+20+4W]
CHKIND		EA+24+2W
CLR1	CY	2
	DIR	2
	reg8, CL	8
	reg16, CL	8
	mem8, CL	EA+16+2W [EA+13+W]
	mem16, CL	EA+16+2W [EA+13+W]
	reg8, imm3	7
	reg16, imm4	7
	mem8, imm3	EA+13+2W [EA+10+W]
	mem16, imm4	EA+13+2W [EA+9+W]
CMP	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+7+W
	mem16, reg16	EA+7+W
	reg8, imm8	5
	reg16, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+8+W
	mem16, imm8	EA+9+W
	mem16, imm16	EA+9+W
	AL, imm8	5
	AW, imm16	6
CMP4S		22+(25+2W)n
CMPBK	mem8, mem8	25+2W [21+2W]
	mem16, mem16	25+2W [19+2W]

4b

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
CMPBKB		16+(23+2W)n
CMPBKW		16+(23+2W)n
CMPM	mem8 mem16	18+W 19+2W
CMPMB		16+(16+W)n
CMPMW		16+(16+2W)n
CVTBD		19
CVTBW		3
CVTDB		20
CVTWL		8
DBNZ		8 or 17
DBNZE		8 or 17
DBNZNE		8 or 17
DEC	reg8 reg16	5 2
	mem8 mem16	EA+13+2W [EA+11+2W] EA+13+2W [EA+9+2W]
DI		4
DISPOSE		11+W
DIV	AW, reg8 AW, mem8	46-56 EA+49+W to EA+59+W
	DW:AW, reg16 DW:AW, mem16	54-64 EA+57+W to EA+67+W
DIVU	AW, reg8 AW, mem8	31 EA+34+W
	DW:AW, reg16 DW:AW, mem16	39 EA+43+2W
DS0:		2
DS1:		2
EI		12
EXT	reg8, reg8 reg8, imm4	41-121 42-122
FINT		2
FP01		55+5W [43+5W]
FP02		55+5W [43+5W]
HALT		N/A
IN	AL, imm8 AW, imm8	15+W 15+W
	AL, DW AW, DW	14+W 14+W
INC	reg8 reg16	5 2
	mem8 mem16	EA+13+2W [EA+13+2W] EA+13+2W [EA+9+2W]

Mnemonic	Operand	Clocks
INM	mem8, DW mem16, DW	21+2W [19+2W] 19+2W [15+2W]
	mem8, DW mem16, DW	18+(15+2W)n [18+(13+2W)n] 18+(13+2W)n [18+(9+2W)n]
INS	reg8, reg8 reg8, imm4	63-155 64-156
LDEA		EA+2
LDM	mem8	13+W
	mem16	16+(11+W)n
LDMB	mem16	13+W
LDMW	mem8	16+(10+W)n
MOV	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+7+W EA+7+W
	mem8, reg8 mem16, reg16	EA+5+W [EA+2] EA+5+W [EA+2]
	reg8, imm8 reg16, imm16	5 6
	mem8, imm8 mem16, imm16	EA+6+W EA+6+W
	AL, dmem8 AW, dmem16	10+W 10+W
	dmem8, AL dmem16, AW	8+W [5] 8+W [5]
	sreg, reg16 sreg, mem16	4 EA+9+2W
	reg16, sreg mem16, sreg	3 EA+6+2W [EA+3]
	AH, PSW PSW, AH	2 3
	DS0, reg16, memptr32 DS1, reg16, memptr32	EA+17+2W EA+17+2W
MOVBK	mem8, mem8 mem16, mem16	22+2W [17+W] 22+2W [17+3W]
MOVBKB	mem8, mem8	16+(18+2W)n [16+(13+W)n]
MOVBKW	mem16, mem16	16+(18+2W)n [16+(10+W)n]
MOVSPA		16
MOVSPB		11
MUL	AW, AL, reg8 AW, AL, mem8	31-40 EA+34+W to EA+43+W
	DW:AW, AW, reg16 DW:AW, AW, mem16	39-48 EA+42+W to EA+51+W
	reg16, reg16, imm8 reg16, mem16, imm8	39-49 EA+42+W to EA+52+W
	reg16, reg16, imm16 reg16, mem16, imm16	40-50 EA+43+W to EA+53+W

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
MULU	reg8	24
	mem8	EA+27+W
	reg16	32
	mem16	EA+33+W
NEG	reg8	5
	reg16	5
	mem8	EA+13+2W [EA+10+W]
	mem16	EA+13+2W [EA+10+W]
NOP		4
NOT	reg8	5
	reg16	5
	mem8	EA+13+2W [EA+10+W]
	mem16	EA+13+2W [EA+10+W]
NOT1	CY	2
	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+15+2W [EA+12+W]
	mem16, CL	EA+15+2W [EA+12+W]
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+12+2W [EA+9+W]
	mem16, imm4	EA+12+2W [EA+9+W]
OR	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+10+2W [EA+7+W]
	mem16, reg16	EA+10+2W [EA+7+W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+11+2W [EA+9+2W]
	mem16, imm16	EA+12+2W [EA+8+2W]
	AL, imm8	5
	AW, imm16	6
OUT	imm8, AL	11+W
	imm8, AW	9+W
	DW, AL	10+W
	DW, AW	8+W
OUTM	DW, mem8	21+2W [19+2W]
	DW, mem16	21+4W [17+4W]
	DW, mem8	18+(15+2W)n [18+(13+2W)n]
	DW, mem16	18+(13+2W)n [18+9+2W)n]
POLL		N/A
POP	reg16	11+W
	mem16	EA+14+2W [EA+11+W]
	DS1	12+W
	SS	12+W
	DS0	12+W
	PSW	13+W
	R	74+8W [58]

Mnemonic	Operand	Clocks
PREPARE	imm16, imm8	imm8 = 0: 26+W imm8 = 1: 37+2W imm8 = n > 1: 44+19 (n−1)+2nW
PS:		2
PUSH	reg16	13+W [9+W]
	mem16	EA+16+2W [EA+12+2W]
	DS1	10+W [7]
	PS	10+W [7]
	SS	10+W [7]
	DS0	10+W [7]
	PSW	9+W [6]
	R	74+8W [50]
	imm8	12+W [9]
	imm16	13+W [10]
REP		2
REPE		2
REPZ		2
REPC		2
REPNC		2
REPNE		2
REPNZ		2
RET	null	19+W
	pop-value	19+W
	null	27+2W
	pop-value	28+W
RETI		40+3W [34+W]
RETRBI		12
ROL	reg8, 1	8
	reg16, 1	8
	mem8, 1	EA+16+2W [EA+13+W]
	mem16, 1	EA+16+2W [EA+13+W]
	reg8, CL	11+2n
	reg16, CL	11+2n
	mem8, CL	EA+19+2W+2n [EA+16+W+2n]
	mem16, CL	EA+19+2W+2n [EA+16+W+2n]
	reg8, imm8	9+2n
	reg16, imm8	9+2n
	mem8, imm8	EA+15+2W+2n [EA+12+W+2n]
	mem16, imm8	EA+15+2W+2n [EA+12+W+2n]
ROL4	reg8	17
	mem8	EA+20+2W [EA+18+2W]
ROLC		Same as ROL
ROR		Same as ROL
ROR4	reg8	21
	mem8	EA+26+2W [EA+24+2W]
RORC		Same as ROL
SET1	CY	2
	DIR	2

4b

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
SET1 (cont)	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+15+2W [EA+12+W]
	mem16, CL	EA+15+2W [EA+12+W]
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+12+2W [EA+9+W]
	mem16, imm4	EA+12+2W [EA+9+W]
SHL		Same as ROL
SHR		Same as ROL
SHRA		Same as ROL
SS:		2
STM	mem8	13+W [10]
	mem16	13+W [10]
STMB	mem8	16+(9+W)n [16+(7+W)n]
STMW	mem16	16+(9+W)n [16+(5+W)n]
STOP		N/A
SUB		Same as ADD
SUB4S		22+(30+3W)n [22+(28+3W)n]
SUBC		Same as ADD
TEST	reg8, reg8	4
	reg16, reg16	4
	reg8, mem8	EA+12+W
	reg16, mem16	EA+11+2W
	mem8, reg8	EA+12+W
	mem16, reg16	EA+11+2W
	reg8, imm8	7
	reg16, imm16	8
	mem8, imm8	EA+9+W
	mem16, imm16	EA+10+W
	AL, imm8	5
	AW, imm16	6
TEST1	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+12+W
	mem16, CL	EA+12+W
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+9+W
	mem16, imm4	EA+9+W
TRANS		11+W
TRANSB		11+W
TSKSW		20

Mnemonic	Operand	Clocks
XCH	reg8, reg8	3
	reg16, reg16	3
	reg8, mem8	EA+12+2W [EA+9+W]
	reg16, mem16	EA+12+2W [EA+9+W]
	mem8, reg8	EA+12+2W [EA+9+W]
	mem16, reg16	EA+12+2W [EA+9+W]
	AW, reg16	4
	reg16, AW	4
XOR		Same as AND

Instruction Clock Count for Operations

	Byte		Word	
	RAM Enable	RAM Disable	RAM Enable	RAM Disable
Context switch interrupt	—	—	33	33
DMA (Single-step mode)	20 + 2W	20 + 2W	20 + 2W	20 + 2W
DMA (Demand release mode)	10 + 15n	10 + 15n	10 + 15n	10 + 15n (min)
DMA (Burst mode)	13 + (12 + 2W)n	13 + (12 + 2W)n	13 + (12 + 2W)n	13 + (12 + 2W)n
DMA (Single-transfer mode)	17 + W	17 +W	17 + W	17 + W
Interrupt (INT pin)	—	—	57 + 3W	45 + 3W
Macro service, sfr ← mem	31 + W	26 + W	31 +W	26 + W
Macro service, mem ← sfr	28 + W	27 + W	28 + W	27 + W
Macro service (Search char mode), sfr ← mem	34 + W	34 + W	—	—
Macro service (Search char mode), mem ← sfr	44 + W	44 + W	—	—
Priority interrupt (Vectored mode)	—	—	55 + 5W	55 + 5W
NMI (Vectored mode)	—	—	53 + 5W	53 + 5W

W = number of wait states inserted into external bus cycle
n = number of iterations

Interrupt Latency

	Clocks	
Source	Typ	Max
NMI pin	12 + N	18 + N
INT pin	8 + N	8 + N
All others	27 + N	15 + N

N = number of clocks to complete the instruction currently executing

4b

Instruction Set

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Data Transfer										
MOV	reg, reg	reg ← reg	1 0 0 0 1 0 1 W 1 1 reg reg	2						
	mem, reg	(mem) ← reg	1 0 0 0 1 0 0 W mod reg mem	2-4						
	reg, mem	reg ← (mem)	1 0 0 0 1 0 1 W mod reg mem	2-4						
	mem, imm	(mem) ← imm	1 1 0 0 0 1 1 W mod 0 0 0 mem	3-6						
	reg, imm	reg ← imm	1 0 1 1 W reg	2-3						
	acc, dmem	When W = 0 AL ← (dmem) When W = 1 AH ← (dmem + 1), AL ← (dmem)	1 0 1 0 0 0 0 W	3						
	dmem, acc	When W = 0 (dmem) ← AL When W = 1 (dmem + 1) ← AH, (dmem) ← AL	1 0 1 0 0 0 1 W	3						
	sreg, reg16	sreg ← reg16 sreg : SS, DS0, DS1	1 0 0 0 1 1 1 0 1 1 0 sreg reg	2						
	sreg, mem16	sreg ← (mem16) sreg : SS, DS0, DS1	1 0 0 0 1 1 1 0 mod 0 sreg mem	2-4						
	reg16, sreg	reg16 ← sreg	1 0 0 0 1 1 0 0 1 1 0 sreg reg	2						
	mem16, sreg	(mem16) ← sreg	1 0 0 0 1 1 0 0 mod 0 sreg mem	2-4						
	DS0, reg16, mem32	reg16 ← (mem32) DS0 ← (mem32 + 2)	1 1 0 0 0 1 0 1 mod reg mem	2-4						
	DS1, reg16, mem32	reg16 ← (mem32) DS1 ← (mem32 + 2)	1 1 0 0 0 1 0 0 mod reg mem	2-4						
	AH, PSW	AH ← S, Z, x, AC, x, P, x, CY	1 0 0 1 1 1 1 1	1						
	PSW, AH	S, Z, x, AC, x, P, x, CY ← AH	1 0 0 1 1 1 1 0	1	x	x		x	x	x
LDEA	reg16, mem16	reg16 ← mem16	1 0 0 0 1 1 0 1 mod reg mem	2-4						
TRANS	src-table	AL ← (BW + AL)	1 1 0 1 0 1 1 1	1						
XCH	reg, reg	reg ⟷ reg	1 0 0 0 0 1 1 W 1 1 reg reg	2						
	mem, reg or reg, mem	(mem) ⟷ reg	1 0 0 0 0 1 1 W mod reg mem	2-4						
	AW, reg16 or reg16, AW	AW ⟷ reg16	1 0 0 1 0 reg	1						
Repeat Prefixes										
REPC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. When CY ≠ 1, exit the loop.	0 1 1 0 0 1 0 1	1						
REPNC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. When CY ≠ 0, exit the loop.	0 1 1 0 0 1 0 0	1						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Repeat Prefixes (cont)																									
REP REPE REPZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 1, exit the loop.	1	1	1	1	0	0	1	1									1						
REPNE REPNZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 0, exit the loop.	1	1	1	1	0	0	1	0									1						
Primitive Block Transfer																									
MOVBK	dst-block, src-block	When W = 0 (IY) ← (IX) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX − 1, IY ← IY − 1 When W = 1 (IY + 1, IY) ← (IX + 1, IX) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX − 2, IY ← IY − 2	1	0	1	0	0	1	0	W									1						
CMPBK	src-block, dst-block	When W = 0 (IX) − (IY) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX − 1, IY ← IY − 1 When W = 1 (IX + 1, IX) − (IY + 1, IY) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX − 2, IY ← IY − 2	1	0	1	0	0	1	1	W									1	x	x	x	x	x	x
CMPM	dst-block	When W = 0 AL − (IY) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY − 1 When W = 1 AW − (IY + 1, IY) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY − 2	1	0	1	0	1	1	1	W									1	x	x	x	x	x	x
LDM	src-block	When W = 0 AL ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX − 1 When W = 1 AW ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX − 2	1	0	1	0	1	1	0	W									1						
STM	dst-block	When W = 0 (IY) ← AL DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY − 1 When W = 1 (IY + 1, IY) ← AW DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY − 2	1	0	1	0	1	0	1	W									1						
Bit Field Transfer																									
INS	reg8, reg8	16-Bit field ← AW	0 1	0 1	0	0 reg	1	1	1 reg	1	0	0	1	1	0	0	0	1	3						
	reg8, imm4	16-Bit field ← AW	0 1	0 1	0	0 0	1	1	1 reg	1	0	0	1	1	1	0	0	1	4						

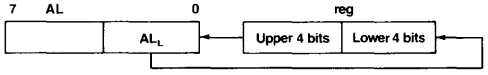
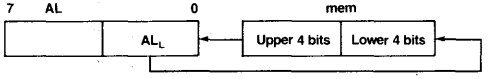
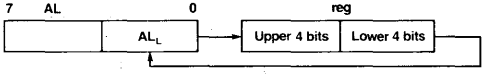
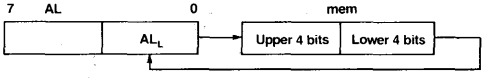
Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Bit Field Transfer (cont)																									
EXT	reg8, reg8	AW ← 16-Bit field	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	3						
			1	1	reg				reg																
	reg8, imm4	AW ← 16-Bit field	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	4						
			1	1	0	0	0		reg																
I/O																									
IN	acc, imm8	When W = 0 AL ← (imm8) When W = 1 AH ← (imm8 + 1), AL ← (imm8)	1	1	1	0	0	1	0	W									2						
	acc, DW	When W = 0 AL ← (DW) When W = 1 AH ← (DW + 1), AL ← (DW)	1	1	1	0	1	1	0	W									1						
OUT	imm8, acc	When W = 0 (imm8) ← AL When W = 1 (imm8 + 1) ← AH, (imm8) ← AL	1	1	1	0	0	1	1	W									2						
	DW, acc	When W = 0 (DW) ← AL When W = 1 (DW + 1) ← AH, (DW) ← AL	1	1	1	0	1	1	1	W									1						
Primitive Block I/O Transfer																									
INM	dst-block, DW	When W = 0 (IY) ← (DW) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY − 1 When W = 1 (IY + 1, IY) ← (DW + 1, DW) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY − 2	0	1	1	0	1	1	0	W									1						
OUTM	DW, src-block	When W = 0 (DW) ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX − 1 When W = 1 (DW + 1, DW) ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX − 2	0	1	1	0	1	1	1	W									1						
									n: number of transfers																
Addition/Subtraction																									
ADD	reg, reg	reg ← reg + reg	0	0	0	0	0	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) + reg	0	0	0	0	0	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ← reg + (mem)	0	0	0	0	0	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, imm	reg ← reg + imm	1	0	0	0	0	0	S	W	1	1	0	0	0		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) + imm	1	0	0	0	0	0	S	W	mod		0	0	0		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL + imm When W = 1 AW ← AW + imm	0	0	0	0	0	1	0	W									2-3	x	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Addition/Subtraction (cont)																									
ADDC	reg, reg	reg ← reg + reg + CY	0	0	0	1	0	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) + reg + CY	0	0	0	1	0	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ← reg + (mem) + CY	0	0	0	1	0	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, imm	reg ← reg + imm + CY	1	0	0	0	0	0	S	W	1	1	0	1	0		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) + imm + CY	1	0	0	0	0	0	S	W	mod		0	1	0		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL + imm + CY When W = 1 AW ← AW + imm + CY	0	0	0	1	0	1	0	W									2-3	x	x	x	x	x	x
SUB	reg, reg	reg ← reg − reg	0	0	1	0	1	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) − reg	0	0	1	0	1	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ← reg − (mem)	0	0	1	0	1	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, imm	reg ← reg − imm	1	0	0	0	0	0	S	W	1	1	1	0	1		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) − imm	1	0	0	0	0	0	S	W	mod		1	0	1		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL − imm When W = 1 AW ← AW − imm	0	0	1	0	1	1	0	W									2-3	x	x	x	x	x	x
SUBC	reg, reg	reg ← reg − reg − CY	0	0	0	1	1	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) − reg − CY	0	0	0	1	1	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ← reg − (mem) − CY	0	0	0	1	1	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	
	reg, imm	reg ← reg − imm − CY	1	0	0	0	0	0	S	W	1	1	0	1	1		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) − imm − CY	1	0	0	0	0	0	S	W	mod		0	1	1		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL − imm − CY When W = 1 AW ← AW − imm − CY	0	0	0	1	1	1	0	W									2-3	x	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
BCD Operation											
ADD4S		dst BCD string ← dst BCD string + src BCD string	0 0 0 0 1 1 1 1	0 0 1 0 0 0 0 0	2	u	x	u	u	u	x
SUB4S		dst BCD string ← dst BCD string − src BCD string	0 0 0 0 1 1 1 1	0 0 1 0 0 0 1 0	2	u	x	u	u	u	x
CMP4S		dst BCD string − src BCD string	0 0 0 0 1 1 1 1	0 0 1 0 0 1 1 0	2	u	x	u	u	u	x
ROL4	reg8		0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 0 0	3						
	mem8		0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 0 0	3-5						
ROR4	reg8		0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 1 0	3						
	mem8		0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 1 0	3-5						
BCD Adjust											
ADJBA		When (AL AND 0FH) >9 or AC = 1, AL ← AL + 6, AH ← AH + 1, AC ← 1, CY ← AC, AL ← AL AND 0FH	0 0 1 1 0 1 1 1		1	x	x	u	u	u	u
ADJ4A		When (AL AND 0FH) >9 or AC = 1, AL ← AL + 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY — = AL ← AL + 60H, CY ← 1	0 0 1 0 0 1 1 1		1	x	x	u	x	x	x
ADJBS		When (AL AND 0FH) >9 or AC = 1, CY ← AC, AL ← AL AND 0FH	0 0 1 1 1 1 1 1		1	x	x	u	u	u	u
ADJ4S		When (AL AND 0FH) >9 or AC = 1, AL ← AL − 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY — = AL ← AL + 60H, CY ← 1	0 0 1 0 1 1 1 1		1	x	x	u	x	x	x

NEC

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Increment/Decrement																									
INC	reg8	reg8 ← reg8 + 1	1	1	1	1	1	1	1	0	1	1	0	0	0	reg			2	x		x	x	x	x
	mem	(mem) ← (mem) + 1	1	1	1	1	1	1	1	W	mod		0	0	0	mem			2-4	x		x	x	x	x
	reg16	reg16 ← reg16 + 1	0	1	0	0	0	reg											1	x		x	x	x	x
DEC	reg8	reg8 ← reg8 − 1	1	1	1	1	1	1	1	0	1	1	0	0	1	reg			2	x		x	x	x	x
	mem	(mem) ← (mem) − 1	1	1	1	1	1	1	1	W	mod		0	0	1	mem			2-4	x		x	x	x	x
	reg16	reg16 ← reg16 − 1	0	1	0	0	1	reg											1	x		x	x	x	x
Multiplication																									
MULU	reg8	AW ← AL x reg8 AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	0	reg			2	u	x	x	u	u	u
	mem8	AW ← AL x (mem8) AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod		1	0	0	mem			2-4	u	x	x	u	u	u
	reg16	DW, AW ← AW x reg16 DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	0	reg			2	u	x	x	u	u	u
	mem16	DW, AW ← AW x (mem16) DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod		1	0	0	mem			2-4	u	x	x	u	u	u
MUL	reg8	AW ← AL x reg8 AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	1	reg			2	u	x	x	u	u	u
	mem8	AW ← AL x (mem8) AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod		1	0	1	mem			2-4	u	x	x	u	u	u
	reg16	DW, AW ← AW x reg16 DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	1	reg			2	u	x	x	u	u	u
	mem16	DW, AW ← AW x (mem16) DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod		1	0	1	mem			2-4	u	x	x	u	u	u
	reg16, reg16, imm8	reg16 ← reg16 x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	1	1	reg			reg			3	u	x	x	u	u	u
	reg16, mem16, imm8	reg16 ← (mem16) x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	mod		reg			mem			3-5	u	x	x	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Multiplication (cont)																									
MUL (cont)	reg16, reg16, imm16	reg16 ← reg16 x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	1	1	reg			reg			4	u	x	x	u	u	u
	reg16, mem16, imm16	reg16 ← (mem16) x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	mod		reg			mem			4-6	u	x	x	u	u	u
Unsigned Division																									
DIVU	reg8	temp ← AW When temp ÷ reg8 > FFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg			2	u	u	u	u	u	u
	mem8	temp ← AW When temp ÷ (mem8) > FFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1	1	1	1	0	1	1	0	mod		1	1	0	mem			2-4	u	u	u	u	u	u
	reg16	temp ← AW When temp ÷ reg16 > FFFFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg16, AL ← temp ÷ reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg			2	u	u	u	u	u	u
	mem16	temp ← AW When temp ÷ (mem16) > FFFFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem16), AL ← temp ÷ (mem16)	1	1	1	1	0	1	1	1	mod		1	1	0	mem			2-4	u	u	u	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Signed Division																									
DIV	reg8	temp ← AW When temp ÷ reg8 > 0 and temp ÷ reg8 > 7FH or temp ÷ reg8 < 0 and temp ÷ reg8 < 0 - 7FH – 1 (SP – 1, SP – 2) ← PSW, (SP – 3, SP – 4) ← PS (SP – 5, SP – 6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	1	1	1	1	1	reg			2	u	u	u	u	u	u
	mem8	temp W ← When temp ÷ (mem8) > 0 and (mem8) > 7FH or temp ÷ (mem8) < 0 and temp ÷ (mem8) < 0 - 7FH — 1 (SP — 1, SP — 2) ← PSW, (SP — 3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1	1	1	1	0	1	1	0	mod		1	1	1	mem			2-4	u	u	u	u	u	u
	reg 16	temp ← AW When temp ÷ reg 16 > 0 and reg 16 > 7FFFH or temp ÷ reg 16 < 0 - 7FFFH — 1 (SP — 1, SP — 2) ← PSW, (SP — 3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg. 16, AL ← temp ÷ reg 16	1	1	1	1	0	1	1	1	1	1	1	1	1	reg			2	u	u	u	u	u	u
	mem 16	temp ← AW When temp ÷ (mem 16) > 0 and (mem 16) > 7FFFH or temp ÷ (mem 16) < 0 and temp ÷ [**mem 16**] < 0 - 7FFFH — 1 (SP — 1, SP — 2)) ← PSW, (SP —3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem 16), AL ← temp ÷ (mem 16)	1	1	1	1	0	1	1	1	mod		1	1	1	mem			2-4	u	u	u	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Data Conversion											
CVTBD		AH ← AL ÷ 0AH, AL ← AL % 0AH	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0	2	u	u	u	x	x	x
CVTDB		AH ← 0, AL ← AH x 0AH + AL	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0	2	u	u	u	x	x	x
CVTBW		When AL < 80H, AH ← 0, all other times AH ← FFH	1 0 0 1 1 0 0 0		1						
CVTWL		When AL < 8000H, DW ← 0, all other times DW ← FFFFH	1 0 0 1 1 0 0 1		1						
Comparison											
CMP	reg, reg	reg − reg	0 0 1 1 1 0 1 W	1 1 reg reg	2	x	x	x	x	x	x
	mem, reg	(mem) − reg	0 0 1 1 1 0 0 W	mod reg mem	2-4	x	x	x	x	x	x
	reg, mem	reg − (mem)	0 0 1 1 1 0 1 W	mod reg mem	2-4	x	x	x	x	x	x
	reg, imm	reg − imm	1 0 0 0 0 0 S W	1 1 1 1 1 reg	3-4	x	x	x	x	x	x
	mem, imm	(mem) − imm	1 0 0 0 0 0 S W	mod 1 1 1 mem	3-6	x	x	x	x	x	x
	acc, imm	When W = 0, AL − imm When W = 1, AW − imm	0 0 1 1 1 1 0 W		2-3	x	x	x	x	x	x
Complement											
NOT	reg	reg ← reg	1 1 1 1 0 1 1 W	1 1 0 1 0 reg	2						
	mem	(mem) ← (mem)	1 1 1 1 0 1 1 W	mod 0 1 0 mem	2-4						
NEG	reg	reg ← reg + 1	1 1 1 1 0 1 1 W	1 1 0 1 1 reg	2	x	x	x	x	x	x
	mem	(mem) ← (mem) + 1	1 1 1 1 0 1 1 W	mod 0 1 1 mem	2-4	x	x	x	x	x	x
Logical Operation											
TEST	reg, reg	reg AND reg	1 0 0 0 0 1 0 W	1 1 reg reg	2	u	0	0	x	x	x
	mem, reg or reg, mem	(mem) AND reg	1 0 0 0 0 1 0 W	mod reg mem	2-4	u	0	0	x	x	x
	reg, imm	reg AND imm	1 1 1 1 0 1 1 W	1 1 0 0 0 reg	3-4	u	0	0	x	x	x
	mem, imm	(mem) AND imm	1 1 1 1 0 1 1 W	mod 0 0 0 mem	3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL AND imm8 When W = 1, AW AND imm8	1 0 1 0 1 0 0 W		2-3	u	0	0	x	x	x
AND	reg, reg	reg ← reg AND reg	0 0 1 0 0 0 1 W	1 1 reg reg	2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) AND reg	0 0 1 0 0 0 0 W	mod reg mem	2-4	u	0	0	x	x	x
	reg, mem	reg ← reg AND (mem)	0 0 1 0 0 0 1 W	mod reg mem	2-4	u	0	0	x	x	x
	reg, imm	reg ← reg AND imm	1 0 0 0 0 0 0 W	1 1 1 0 0 reg	3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) AND imm	1 0 0 0 0 0 0 W	mod 1 0 0 mem	3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL AND imm8 When W = 1, AW ← AW AND imm16	0 0 1 0 0 1 0 W		2-3	u	0	0	x	x	x

NEC

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Logical Operation (cont)											
OR	reg, reg	reg ← reg OR reg	0 0 0 0 1 0 1 W	1 1 reg reg	2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) OR reg	0 0 0 0 1 0 0 W	mod reg mem	2-4	u	0	0	x	x	x
	reg, mem	reg ← reg OR (mem)	0 0 0 0 1 0 1 W	mod reg mem	2-4	u	0	0	x	x	x
	reg, imm	reg ← reg OR imm	1 0 0 0 0 0 0 W	1 1 0 0 1 reg	3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) OR imm	1 0 0 0 0 0 0 W	mod 0 0 1 mem	3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL OR imm8 When W = 1, AW ← AW OR imm16	0 0 0 0 1 1 0 W		2-3	u	0	0	x	x	x
XOR	reg, reg	reg ← reg XOR reg	0 0 1 1 0 0 1 W	1 1 reg reg	2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) XOR reg	0 0 1 1 0 0 0 W	mod reg mem	2-4	u	0	0	x	x	x
	reg, mem	reg ← reg XOR (mem)	0 0 1 1 0 0 1 W	mod reg mem	2-4	u	0	0	x	x	x
	reg, imm	reg ← reg XOR imm	1 0 0 0 0 0 0 W	1 1 1 1 0 reg	3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) XOR imm	1 0 0 0 0 0 0 W	mod 1 1 0 mem	3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL XOR imm8 When W = 1, AW ← AW XOR imm16	0 0 1 1 0 1 0 W		2-3	u	0	0	x	x	x
Bit Operation											
			2nd byte*	3rd byte*							
TEST1	reg8, CL	reg8 bit no. CL = 0: Z ← 1 reg8 bit no. CL = 1: Z ← 0	0 0 0 1 0 0 0 0	1 1 0 0 0 reg	3	u	0	0	u	u	x
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0 0 0 1 0 0 0 0	mod 0 0 0 mem	3-5	u	0	0	u	u	x
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0 0 0 1 0 0 0 1	1 1 0 0 0 reg	3	u	0	0	u	u	x
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0 0 0 1 0 0 0 1	mod 0 0 0 mem	3-5	u	0	0	u	u	x
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0 0 0 1 1 0 0 0	1 1 0 0 0 reg	4	u	0	0	u	u	x
	mem8, imm3	(mem8) bit no. imm3 = 0: Z ← 1 (mem8) bit no. imm3 = 1: Z ← 0	0 0 0 1 1 0 0 0	mod 0 0 0 mem	4-6	u	0	0	u	u	x
	reg16, imm4	reg16 bit no. imm4 = 0: Z ← 1 reg16 bit no. imm4 = 1: Z ← 0	0 0 0 1 1 0 0 1	1 1 0 0 0 reg	4	u	0	0	u	u	x
	mem16, imm4	(mem16) bit no. imm4 = 0: Z ← 1 (mem16) bit no. imm4 = 1: Z ← 0	0 0 0 1 1 0 0 1	mod 0 0 0 mem	4-6	u	0	0	u	u	x
			2nd byte*	3rd byte*							

*Note: First byte = 0FH

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)											
			2nd byte*	3rd byte*							
NOT1	reg8, CL	reg8 bit no. CL ← reg8 bit no. CL	0 0 0 1 0 1 1 0	1 1 0 0 0 reg	3						
	mem8, CL	(mem8) bit no. CL ← (mem8) bit no. CL	0 0 0 1 0 1 1 0	mod 0 0 0 mem	3-5						
	reg16, CL	reg16 bit no. CL ← reg16 bit no. CL	0 0 0 1 0 1 1 1	1 1 0 0 0 reg	3						
	mem16, CL	(mem16) bit no. CL ← (mem16) bit no. CL	0 0 0 1 0 1 1 1	mod 0 0 0 mem	3-5						
	reg8, imm3	reg8 bit no. imm3 ← reg8 bit no. imm3	0 0 0 1 1 1 1 0	1 1 0 0 0 reg	4						
	mem8, imm3	(mem8) bit no. imm3 ← (mem8) bit no. imm3	0 0 0 1 1 1 1 0	mod 0 0 0 mem	4-6						
	reg16, imm4	reg16 bit no. imm4 ← (reg16) bit no. imm4	0 0 0 1 1 1 1 1	1 1 0 0 0 reg	4						
	mem16, imm4	(mem16) bit no. imm4 ← (mem16) bit no. imm4	0 0 0 1 1 1 1 1	mod 0 0 0 mem	4-6						
			2nd byte* *Note: First byte = 0FH	3rd byte*							
	CY	CY ← CY	1 1 1 1 0 1 0 1		1		x				
			2nd byte*	3rd byte*							
CLR1	reg8, CL	reg8 bit no. CL ← 0	0 0 0 1 0 0 1 0	1 1 0 0 0 reg	3						
	mem8, CL	(mem8) bit no. CL ← 0	0 0 0 1 0 0 1 0	mod 0 0 0 mem	3-5						
	reg16, CL	reg16 bit no. CL ← 0	0 0 0 1 0 0 1 1	1 1 0 0 0 reg	3						
	mem16, CL	(mem16) bit no. CL ← 0	0 0 0 1 0 0 1 1	mod 0 0 0 mem	3-5						
	reg8, imm3	reg8 bit no. imm3 ← 0	0 0 0 1 1 0 1 0	1 1 0 0 0 reg	4						
	mem8, imm3	(mem8) bit no. imm3 ← 0	0 0 0 1 1 0 1 0	mod 0 0 0 mem	4-6						
	reg16, imm4	reg16 bit no. imm4 ← 0	0 0 0 1 1 0 1 1	1 1 0 0 0 reg	4						
	mem16, imm4	(mem16) bit no. imm4 ← 0	0 0 0 1 1 0 1 1	mod 0 0 0 mem	4-6						
			2nd byte* *Note: First byte = 0FH	3rd byte*							
	CY	CY ← 0	1 1 1 1 1 0 0 0		1		0				
	DIR	DIR ← 0	1 1 1 1 1 1 0 0		1						

NEC

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)											
SET1	reg8, CL	reg8 bit no. CL ← 1	0 0 0 1 0 1 0 0	1 1 0 0 0 reg	3						
	mem8, CL	(mem8) bit no. CL ← 1	0 0 0 1 0 1 0 0	mod 0 0 0 mem	3-5						
	reg16, CL	reg16 bit no. CL ← 1	0 0 0 1 0 1 0 1	1 1 0 0 0 reg	3						
	mem16, CL	(mem16) bit no. CL ← 1	0 0 0 1 0 1 0 1	mod 0 0 0 mem	3-5						
	reg8, imm3	reg8 bit no. imm3 ← 1	0 0 0 1 1 1 0 0	1 1 0 0 0 reg	4						
	mem8, imm3	(mem8) bit no. imm3 ← 1	0 0 0 1 1 1 0 0	mod 0 0 0 mem	4-6						
	reg16, imm4	reg16 bit no. imm4 ← 1	0 0 0 1 1 1 0 1	1 1 0 0 0 reg	4						
	mem16, imm4	(mem16) bit no. imm4 ← 1	0 0 0 1 1 1 0 1	mod 0 0 0 mem	4-6						
			2nd byte*	3rd byte*							
			*Note: First byte = 0FH								
	CY	CY ← 1	1 1 1 1 1 0 0 1		1		1				
	DIR	DIR ← 1	1 1 1 1 1 1 0 1		1						
Shift											
SHL	reg, 1	CY ← MSB of reg, reg ← reg x 2 When MSB of reg ≠ CY, V ← 1 When MSB of reg = CY, V ← 0	1 1 0 1 0 0 0 W	1 1 1 0 0 reg	2	u	x	x	x	x	x
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 When MSB of (mem) ≠ CY, V ← 1 When MSB of (mem) = CY, V ← 0	1 1 0 1 0 0 0 W	mod 1 0 0 mem	2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1 1 0 1 0 0 1 W	1 1 1 0 0 reg	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1 1 0 1 0 0 1 W	mod 1 0 0 mem	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1 1 0 0 0 0 0 W	1 1 1 0 0 reg	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1 1 0 0 0 0 0 W	mod 1 0 0 mem	3-5	u	x	u	x	x	x
			n: number of shifts								
SHR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 When MSB of reg ≠ bit following MSB of reg: V ← 1 When MSB of reg = bit following MSB of reg: V ← 0	1 1 0 1 0 0 0 W	1 1 1 0 1 reg	2	u	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Shift (cont)																									
SHR (cont)	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 When MSB of (mem) ≠ bit following MSB of (mem): V ← 1 When MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod		1	0	1		mem		2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	1	0	1		reg		2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1	1	0	1	0	0	1	W	mod		1	0	1		mem		2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	1	0	1		reg		3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 n: number of shifts	1	1	0	0	0	0	0	W	mod		1	0	1		mem		3-5	u	x	u	x	x	x
SHRA	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, V ← 0 MSB of operand does not change	1	1	0	1	0	0	0	W	1	1	1	1	1		reg		2	u	x	0	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, V ← 0, MSB of operand does not change	1	1	0	1	0	0	0	W	mod		1	1	1		mem		2-4	u	x	0	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1	1	0	1	0	0	1	W	1	1	1	1	1		reg		2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1	1	0	1	0	0	1	W	mod		1	1	1		mem		2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1	1	0	0	0	0	0	W	1	1	1	1	1		reg		3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change n: number of shifts	1	1	0	0	0	0	0	W	mod		1	1	1		mem		3-5	u	x	u	x	x	x

NEC

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Rotation																									
ROL	reg, 1	CY ← MSB of reg, reg ← reg x 2 + CY MSB of reg ≠ CY: V ← 1 MSB of reg = CY: V ← 0	1	1	0	1	0	0	0	W	1	1	0	0	0	reg			2		x	x			
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 + CY MSB of (mem) ≠ CY: V ← 1 MSB of (mem) = CY: V ← 0	1	1	0	1	0	0	0	W	mod		0	0	0	mem			2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	0	0	0	reg			2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp − 1	1	1	0	1	0	0	1	W	mod		0	0	0	reg			2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	0	0	0	reg			3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp − 1	1	1	0	0	0	0	0	W	mod		0	0	0	mem			3-5		x	u			
								n: number of shifts																	
ROR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← CY MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	0	0	1	reg			2		x	x			
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← CY MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod		0	0	1	mem			2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	0	0	1	reg			2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY temp ← temp − 1	1	1	0	1	0	0	1	W	mod		0	0	1	mem			2-4		x	u			
								n:number of shifts																	

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	AC	Flags CY	V	P	S	Z
Rotation (cont)											
ROR (cont)	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp − 1	1 1 0 0 0 0 0 W	1 1 0 0 1 reg	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 temp ← temp − 1	1 1 0 0 0 0 0 W	mod 0 0 1 mem n: number of shifts	3-5		x	u			
Rotate											
ROLC	reg, 1	tmpcy ← CY, CY ← MSB of reg reg ← reg x 2 + tmpcy MSB of reg = CY: V ← 0 MSB of reg ≠ CY: V ← 1	1 1 0 1 0 0 0 W	1 1 0 1 0 reg	2		x	x			
	mem, 1	tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy MSB of (mem) = CY: V ← 0 MSB of (mem) ≠ CY: V ← 1	1 1 0 1 0 0 0 W	mod 0 1 0 mem	2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp − 1	1 1 0 1 0 0 1 W	1 1 0 1 0 reg	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy temp ← temp − 1	1 1 0 1 0 0 1 W	mod 0 1 0 mem	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp − 1	1 1 0 0 0 0 0 W	1 1 0 1 0 reg	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy temp ← temp − 1	1 1 0 0 0 0 0 W	mod 0 1 0 mem n: number of shifts	3-5		x	u			

NEC

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Rotate (cont)																									
RORC	reg, 1	tmpcy ← CY, CY ← LSB of reg reg ← reg ÷ 2, MSB of reg ← tmpcy MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	0	1	1	reg			2			x	x		
	mem, 1	tmpcy ← CY, CY ← LSB of (mem) (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod		0	1	1	mem			2-4			x	x		
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	0	1	1	reg			2			x	u		
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	mod		0	1	1	mem			2-4			x	u		
	reg, imm8	temp ← imm8, while temp ≠ 0 repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	0	1	1	reg			3			x	u		
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	mod		0	1	1	mem			3-5			x	u		
Subroutine Control Transfer																									
CALL	near-proc	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← PC + disp	1	1	1	0	1	0	0	0									3						
	regptr16	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← regptr16	1	1	1	1	1	1	1	1	1	1	0	1	0	reg			2						
	memptr16	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← (memptr16)	1	1	1	1	1	1	1	1	mod		0	1	0	mem			2-4						
	far-proc	(SP − 1, SP − 2) ← PS, (SP − 3, SP − 4) ← PC SP ← SP − 4, PS ← seg, PC ← offset	1	0	0	1	1	0	1	0									5						
	memptr32	(SP − 1, SP − 2) ← PS, (SP − 3, SP − 4) ← PC SP ← SP − 4, PS ← (memptr32 + 2), PC ← (memptr32)	1	1	1	1	1	1	1	1	mod		0	1	1	mem			2-4						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Subroutine Control Transfer (cont)										
RET		PC ← (SP + 1, SP), SP ← SP + 2	1 1 0 0 0 0 1 1	1						
	pop-value	PC ← (SP + 1, SP) SP ← SP + 2, SP ← SP + pop-value	1 1 0 0 0 0 1 0	3						
		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2) SP ← SP + 4	1 1 0 0 1 0 1 1	1						
	pop-value	PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2) SP ← SP + 4, SP ← SP + pop-value	1 1 0 0 1 0 1 0	3						
Stack Manipulation										
PUSH	mem16	(SP − 1, SP − 2) ← (mem16), SP ← SP − 2	1 1 1 1 1 1 1 1 mod 1 1 0 mem	2-4						
	reg16	(SP − 1, SP − 2) ← reg16, SP ← SP − 2	0 1 0 1 0 reg	1						
	sreg	(SP− 1, SP− 2) ← sreg, SP ← SP − 2	0 0 0 sreg 1 1 0	1						
	PSW	(SP − 1, SP − 2) ← PSW, SP ← SP − 2	1 0 0 1 1 1 0 0	1						
	R	Push registers on the stack	0 1 1 0 0 0 0 0	1						
	imm	(SP − 1, SP − 2) ← imm SP ← SP − 2, When S = 1, sign extension	0 1 1 0 1 0 S 0	2-3						
POP	mem16	(mem16) ← (SP + 1, SP), SP ← SP + 2	1 0 0 0 1 1 1 1 mod 0 0 0 mem	2-4						
	reg16	reg16 ← (SP + 1, SP), SP ← SP + 2	0 1 0 1 1 reg	1						
	sreg	sreg ← (SP + 1, SP) sreg : SS, DS0, DS1 SP ← SP + 2	0 0 0 sreg 1 1 1	1						
	PSW	PSW ← (SP + 1, SP), SP ← SP + 2	1 0 0 1 1 1 0 1	1	R	R	R	R	R	R
	R	Pop registers from the stack	0 1 1 0 0 0 0 1	1						
PREPARE	imm16, imm8	Prepare new stack frame	1 1 0 0 1 0 0 0 *: imm8 = 0: 16 imm8 > 1: 25 + 16 (imm8 − 1)	4						
DISPOSE		Dispose of stack frame	1 1 0 0 1 0 0 1	1						
Branch										
BR	near-label	PC ← PC + disp	1 1 1 0 1 0 0 1	3						
	short-label	PC ← PC + ext-disp8	1 1 1 0 1 0 1 1	2						
	regptr16	PC ← regptr16	1 1 1 1 1 1 1 1 1 1 1 0 0 reg	2						
	memptr16	PC ← (memptr16)	1 1 1 1 1 1 1 1 mod 1 0 0 mem	2-4						
	far-label	PS ← seg, PC ← offset	1 1 1 0 1 0 1 0	5						
	memptr32	PS ← (memptr32 + 2), PC ← (memptr32)	1 1 1 1 1 1 1 1 mod 1 0 1 mem	2-4						

NEC

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Conditional Branch																									
BV	short-label	if V = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	0	0									2						
BNV	short-label	if V = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	0	1									2						
BC, BL	short-label	if CY = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	1	0									2						
BNC, BNL	short-label	if CY = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	1	1									2						
BE, BZ	short-label	if Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	0	0									2						
BNE, BNZ	short-label	if Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	0	1									2						
BNH	short-label	if CY OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	1	0									2						
BH	short-label	if CY OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	1	1									2						
BN	short-label	if S = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	0	0									2						
BP	short-label	if S = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	0	1									2						
BPE	short-label	if P = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	1	0									2						
BPO	short-label	if P = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	1	1									2						
BLT	short-label	if S XOR V = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	0	0									2						
BGE	short-label	if S XOR V = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	0	1									2						
BLE	short-label	if (S XOR V) OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	1	0									2						
BGT	short-label	if (S XOR V) OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	1	1									2						
DBNZNE	short-label	CW ← CW − 1 if Z = 0 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	0									2						
DBNZE	short-label	CW ← CW − 1 if Z = 1 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	1									2						
DBNZ	short-label	CW ← CW − 1 if CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	1	0									2						
BCWZ	short-label	if CW = 0, PC ← PC + ext-disp8	1	1	1	0	0	0	1	1									2						
BTCLR	sfr. imm3, short-label	if bit no. imm3 of (sfr) = 1, PC ← PC + ext — disp8, bit no. imm3 of (sfr) ← 0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	0	0	5						
Interrupt																									
BRK	3	(SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PS ← (15, 14), PC ← (13, 12)	1	1	0	0	1	1	0	0									1						
	imm8 (≠ 3)	(SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PC ← (n x 4, + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1	1	0	0	1	1	0	1									2						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Interrupt (cont)																									
BRKV		When V = 1 (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PS ← (19, 18), PC ← (17, 16)	1	1	0	0	1	1	1	0									1						
RETI		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1	1	0	0	1	1	1	1									1	R	R	R	R	R	R
RETRBI		PC ← Save PC, PSW ← Save PSW	0	0	0	0	1	1	1	1	1	0	0	1	0	0	0	1	2	R	R	R	R	R	R
FINT		Indicates that interrupt service routine to the interrupt controller built in the CPU has been completed	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1	0	2						
CHKIND	reg16, mem32	When (mem32) > reg16 or (mem32 + 2) < reg16 (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (23, 22), PC ← (21, 20)	0	1	1	0	0	0	1	0	mod			reg			mem		2-4						
CPU Control																									
HALT		CPU Halt	1	1	1	1	0	1	0	0										1					
STOP		CPU Halt	0	0	0	0	1	1	1	1	1	0		1	1		1	1	1 0	1					
BUSLOCK		Bus Lock Prefix	1	1	1	1	0	0	0	0										1					
FPO1 (Note 1)	fp-op	No Operation	1	1	0	1	1	X	X	X	1	1	Y	Y	Y	Z	Z	Z	2						
	fp-op, mem	data bus ← (mem)	1	1	0	1	1	X	X	X	mod		Y	Y	Y		mem		2-4						
FPO2 (Note 1)	fp-op	No Operation	0	1	1	0	0	1	1	X	1	1	Y	Y	Y	Z	Z	Z	2						
	fp-op, mem	data bus ← (mem)	0	1	1	0	0	1	1	X	mod		Y	Y	Y		mem		2-4						
POLL		Poll and wait	1	0	0	1	1	0	1	1									1						
NOP		No Operation	1	0	0	1	0	0	0	0									1						
DI		IE ← 0	1	1	1	1	1	0	1	0									1						
EI		IE ← 1	1	1	1	1	1	0	1	1									1						
DS0; DS1; PS; SS		Segment override prefix	0	0	1	sreg		1	1	0									1						

Notes:

(1) Does not execute on the V25, but does generate an interrupt.

Register Banks

MOVSPA		0	0	0	0	1	1	1	1	0	0	1	0	0	1	0	1		2							
BRKCS	reg16	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	1		3							
MOVSPB	reg16	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	1		3							
		1	1	1	1	1		reg																		
TSKSW	reg16	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	0		3	x	x	x	x	x	x	
		1	1	1	1	1		reg																		

NEC
NEC Electronics Inc.

μPD70P322
16-Bit Microcomputer: Single-Chip, CMOS, With EPROM for V25/V35 Modes

Description

The μPD70P322 is a 16-bit, single-chip CMOS microcomputer operable as a μPD70322 (V25™) or a μPD70332 (V35™). The mask ROM of the V25/V35 is replaced in the μPD70P322 by an EPROM.

Ordering Information

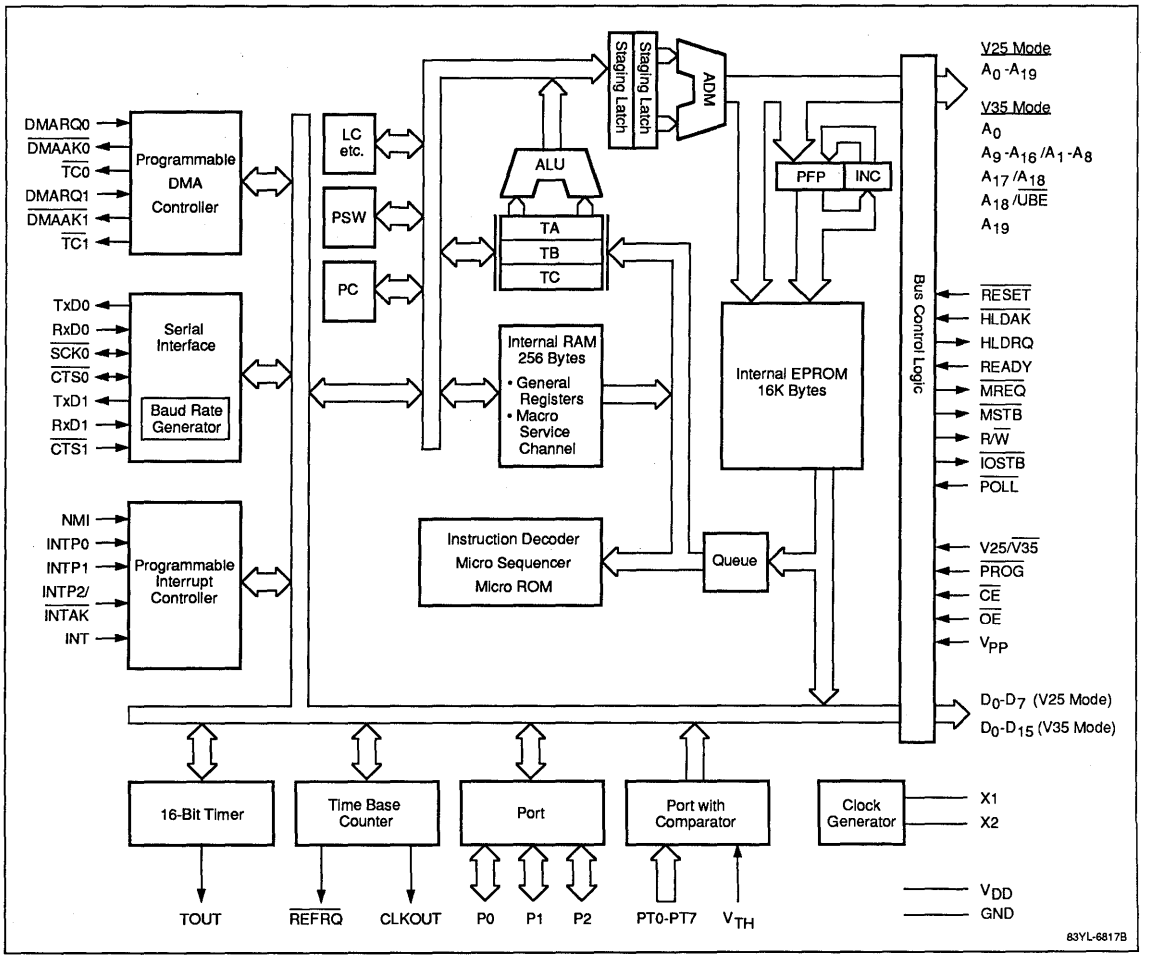
Part Number	Ext Input Frequency	Int System Clock	Package
μPD70P322KE-8	16 MHz	8 MHz	84-pin ceramic LCC with quartz window

V25 and V35 are trademarks of NEC Corporation

Features

- Reprogrammable EPROM appropriate for system evaluation of V25 or V35
- V25 mode (μPD70322 equivalent)
  - Internal 16-bit architecture
  - External 8-bit data bus
- V35 mode (μPD70332 equivalent)
  - Internal 16-bit architecture
  - External 16-bit data bus

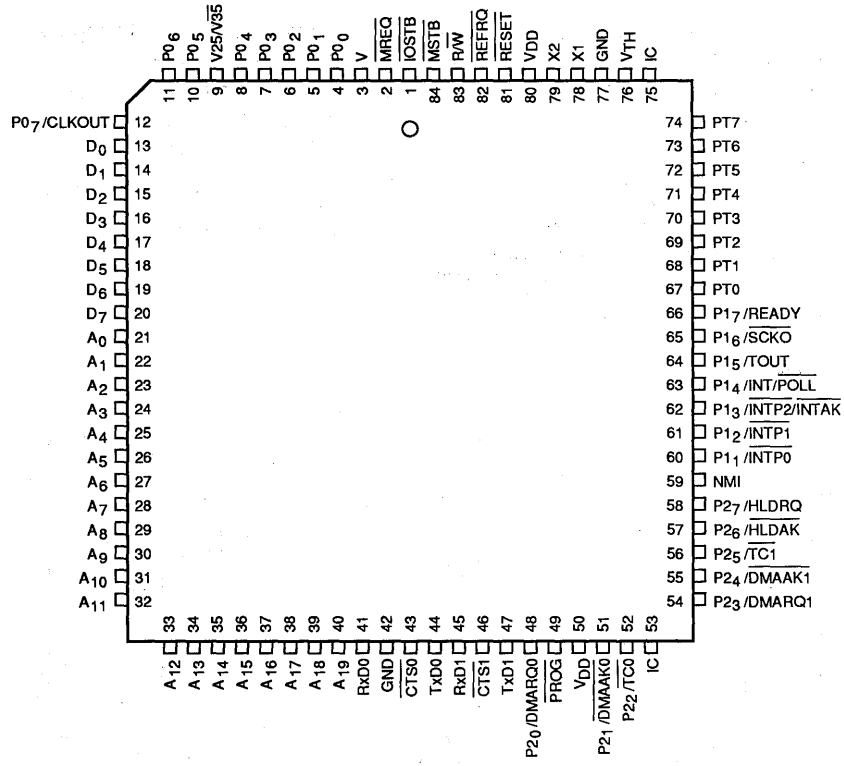
μPD70P322 Block Diagram



4c

Pin Configurations

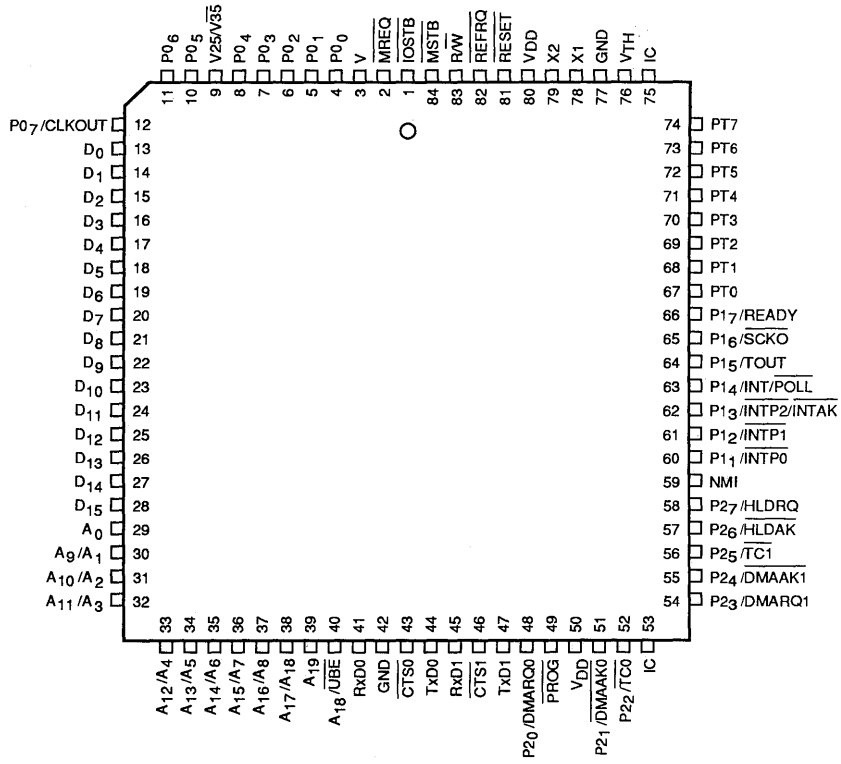
84-Pin LCC, V25 Mode



Pin	Symbol	Connect to
3	V	V_{DD}
9	V25/$\overline{V35}$	High level (H)
49	$\overline{PROG}$	High level (H)
53, 75	IC	High level through pullup resistor

83YL-6794B

84-Pin LCC, V35 Mode

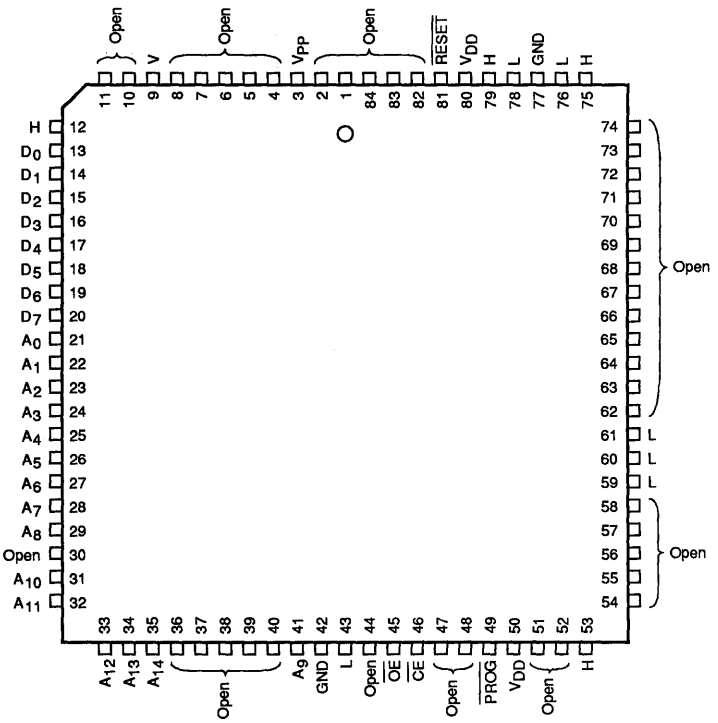


Pin	Symbol	Connect to
3	V	V_{DD}
9	V25/$\overline{V35}$	Low level (L)
49	$\overline{PROG}$	High level (H)
53, 75	IC	High level through pullup resistor

83YL-6796B

4c

84-Pin LCC, EPROM Programming Mode



Pin	Symbol	Connect to
9	V	V_{DD}
49	$\overline{PROG}$	Low level through pullup resistor
	L	Low level through pullup resistor
	H	High level through pullup resistor
	Open	Do not connect to these pins.

83YL-6795B

Pin Identification, V25/V35 Mode

Symbol	I/O	Function	Also Used For
Port Pins			
$P0_0$-$P0_6$	I/O	Input or output mode can be specified per bit	—
$P0_7$			CLKOUT
($P1_0$) NMI	In	Non-maskable interrupt; cannot be used as a general-purpose port pin.	—
$P1_1$	In	Port 1 input lines	$\overline{INTP0}$
$P1_2$			$\overline{INTP1}$
$P1_3$			$\overline{INTP2}$
$P1_4$	I/O	Input or output mode can be specified per bit.	$\overline{POLL}$/INT
$P1_5$			TOUT
$P1_6$			$\overline{SCK0}$
$P1_7$			READY
$P2_0$	I/O	Input or output mode can be specified per bit	DMARQ0
$P2_1$			$\overline{DMAAK0}$
$P2_2$			$\overline{TC0}$
$P2_3$			$\overline{DMARQ1}$
$P2_4$			$\overline{DMAAK1}$
$P2_5$			$\overline{TC1}$
$P2_6$			$\overline{HLDAK}$
$P2_7$			HLDRQ
PT0-PT7	In	Comparator input	—
Pins Other Than Port			
A_0	Out	V35 mode. Selects low-order memory bank	—
A_1-A_{19}	Out	V35 mode. Address bus	—
A_0-A_{19}	Out	V25 mode. Address bus	—
CLKOUT	Out	System clock	$P0_7$
$\overline{CTS0}$	I/O	Asynchronous mode: send instruction input I/O interface mode: receive clock input/output	—
$\overline{CTS1}$	In	Send instruction input	—
D_0-D_7	I/O	V25 mode. 8-bit data bus	—
D_0-D_{15}	I/O	V35 mode. 16-bit data bus	—
$\overline{DMAAK0}$	Out	DMA acknowledge	$P2_1$
$\overline{DMAAK1}$			$P2_4$
DMARQ0	In	DMA request	$P2_0$
DMARQ1			$P2_3$
$\overline{HLDAK}$	Out	Hold acknowledge	$P2_6$
HLDRQ	In	Hold request	$P2_7$
INT	In	External interrupt request	$P1_4$/$\overline{POLL}$

Symbol	I/O	Function	Also Used For
$\overline{INTAK}$	Out	Interrupt acknowledge	$P1_3$/$\overline{INTP2}$
$\overline{INTP0}$	In	External interrupt request	$P1_1$
$\overline{INTP1}$			$P1_2$
$\overline{INTP2}$			$P1_3$/$\overline{INTAK}$
$\overline{IOSTB}$	Out	I/O access strobe	
$\overline{MREQ}$	Out	Indicates memory bus cycle start	
$\overline{MSTB}$	Out	Memory access strobe	
$\overline{POLL}$	In	Wait insertion	$P1_4$/INT
READY	In	External ready	$P1_7$
$\overline{REFRQ}$	Out	DRAM refresh pulse	
$\overline{RESET}$	In	Chip reset	
R/$\overline{W}$	Out	Indicates read cycle/write cycle	
RxD0, RxD1	Out	Serial data	
$\overline{SCK0}$	Out	Serial clock	$P1_6$
$\overline{TC0}$	Out	Indicates DMA completion	$P2_2$
$\overline{TC1}$			$P2_5$
TOUT	Out	Timer output	$P1_5$
TxD0, TxD1	In	Serial data	
$\overline{UBE}$	Out	V35 mode. Selects high-order memory bank	
V25/$\overline{V35}$	In	V25 or V35 mode selection	
X1, X2	In	Internal oscillator: connect crystal or ceramic resonator to X1 and X2. External clock: connect opposite-phase clock inputs to X1 and X2.	
V_{DD}	In	+5-volt power supply pin (both V_{DD} pins are connected)	
V_{TH}	In	Comparator reference voltage	
GND		Ground pin (both GND pins are connected)	

4c

Pin Identification, EPROM Programming Mode

Symbol	I/O	Function
A_0-A_{14}	In	Address bus
$\overline{CE}$	In	Chip enable
D_0-D_7	I/O	Data bus
$\overline{OE}$	In	Output enable
$\overline{PROG}$	In	EPROM programming mode setting
$\overline{RESET}$	In	EPROM mode setting
V_{PP}	In	Write power supply pin
V_{DD}	In	+5-volt power supply pin (both V_{DD} pins are connected
GND	—	Ground pin (both GND pins are connected)

EPROM PROGRAMMING

The three basic modes of the μPD70P322 are controlled by the level at the $\overline{PROG}$, V25/$\overline{V35}$, and $\overline{RESET}$ pins. (H = high level; L = low level; x = don't care).

Mode	$\overline{PROG}$	V25/$\overline{V35}$	$\overline{RESET}$
V25	H	H	x
V35	H	L	x
Program	L	x	L

Table 1 lists the operations that take place in programming mode and the conditions at the power and control pins. Table 2 lists the recommended conditions at pins not used in programming mode.

Table 1. Conditions at Pins Used in Programming Mode

Operation Mode	$\overline{CE}$	$\overline{OE}$	V_{PP}	V_{DD}
Read	L	L	+5 V	+5 V
Output disable	L	H		
Standby	H	x		
Program	L	H	+12.5 V	+6 V
Program verify	x	L		
Program inhibit	H	H		

Notes:

(1) $\overline{PROG}$ and $\overline{RESET}$ = L

(2) Apply voltage to V_{DD} before V_{PP}. Remove voltage from V_{PP} before V_{DD}.

(3) Never apply more than 13.5 V to V_{PP} even with overshoot.

(4) x = L or H

(5) With A_{14} set to 1 (addresses 4000H-7FFFH), write ROM data. At verification, output data should be FFH. Thus, it is invalid to write program setting A_{14} to 1.

Table 2. Conditions at Pins Not Used in Programming Mode

Connection	Pins
Pullup resistor	12, 53, 75, 79
Pulldown resistor	43, 59-61, 76, 78
Pullup or pulldown	9
No connection	1-2, 4-8, 10-11, 30, 36-40, 47-48, 51-52, 54-58, 62-74, 82-84

EPROM Write Procedure

Figure 1 is the flowchart and figure 2 is the timing diagram for the following EPROM write procedure.

Note: The protection seal on the quartz window of the μPD70P322 must be in place.

(1) Apply +6 volts to the V_{DD} pins and +12.5 volts to the V_{PP} pin.

(2) Supply initial address A_{13}-A_0.

(3) Supply write data D_7-D_0.

(4) Apply 1-ms program pulse (active low) to $\overline{CE}$ pin.

(5) Change to verify operation mode. If data can be written normally, go to step (8).

(6) If data cannot be written normally, repeat steps (3) to (5).

(7) If data cannot be written after 25 repetitions, declare the device faulty. Stop the write operation.

(8) Supply write data.

(9) Increment address.

(10) Repeat steps (3) to (9) until the end address is reached.

EPROM Read Procedure

With the μPD70P322 set up for read operation (table 1), the EPROM contents are read into the external data bus according to the procedure below. Figure 3 is a simplified timing diagram.

(1) Apply +5 volts to the V_{DD} pins.

(2) Apply +5 volts to the V_{PP} pins.

(3) Input the address of the data to be read to pins A_{13}-A_0.

(4) Perform read mode operation.

(5) Output data to pins D_7-D_0.

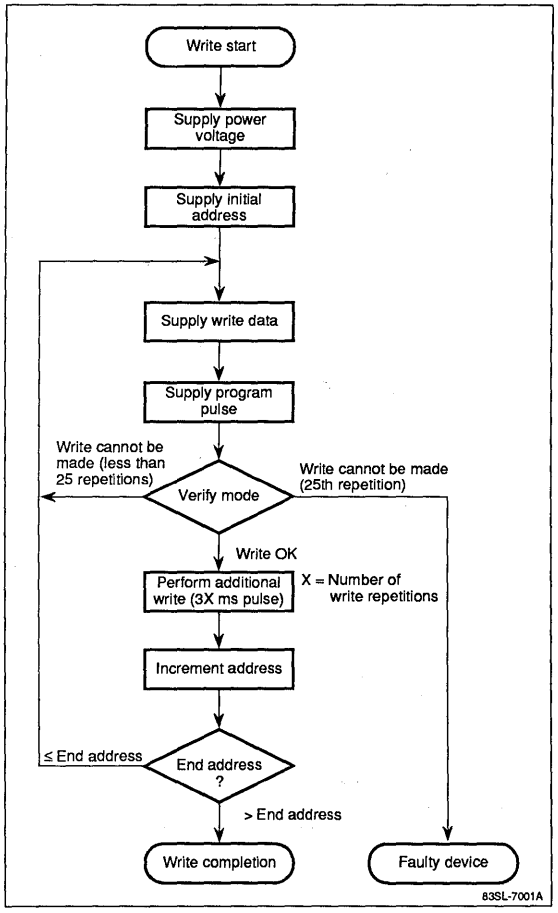
EPROM Erasure

Data in the EPROM is erased by exposing the quartz window in the ceramic package to light having a wavelength shorter than 400 nm, including ultraviolet rays, direct sunlight, and fluorescent light.

Note: To prevent unintentional erasure, the protection seal on the quartz window should not be removed except for EPROM erasure.

Typically, data is erased by 254-nm ultraviolet rays. A minimum lighting level of 15 W•s/cm^2 (ray intensity x exposure time) is required to completely erase the EPROM. Erasure by an ultraviolet lamp rated at 12 mW/cm^2 takes about 15 to 20 minutes. The time may be prolonged because of a degraded lamp, dirty window, etc. Remove any filter on the lamp and place the device within 2.5 cm of the lamp tubes.

Figure 1. EPROM Write Procedure Flowchart



4c

Figure 2. EPROM Write and Verify Timing

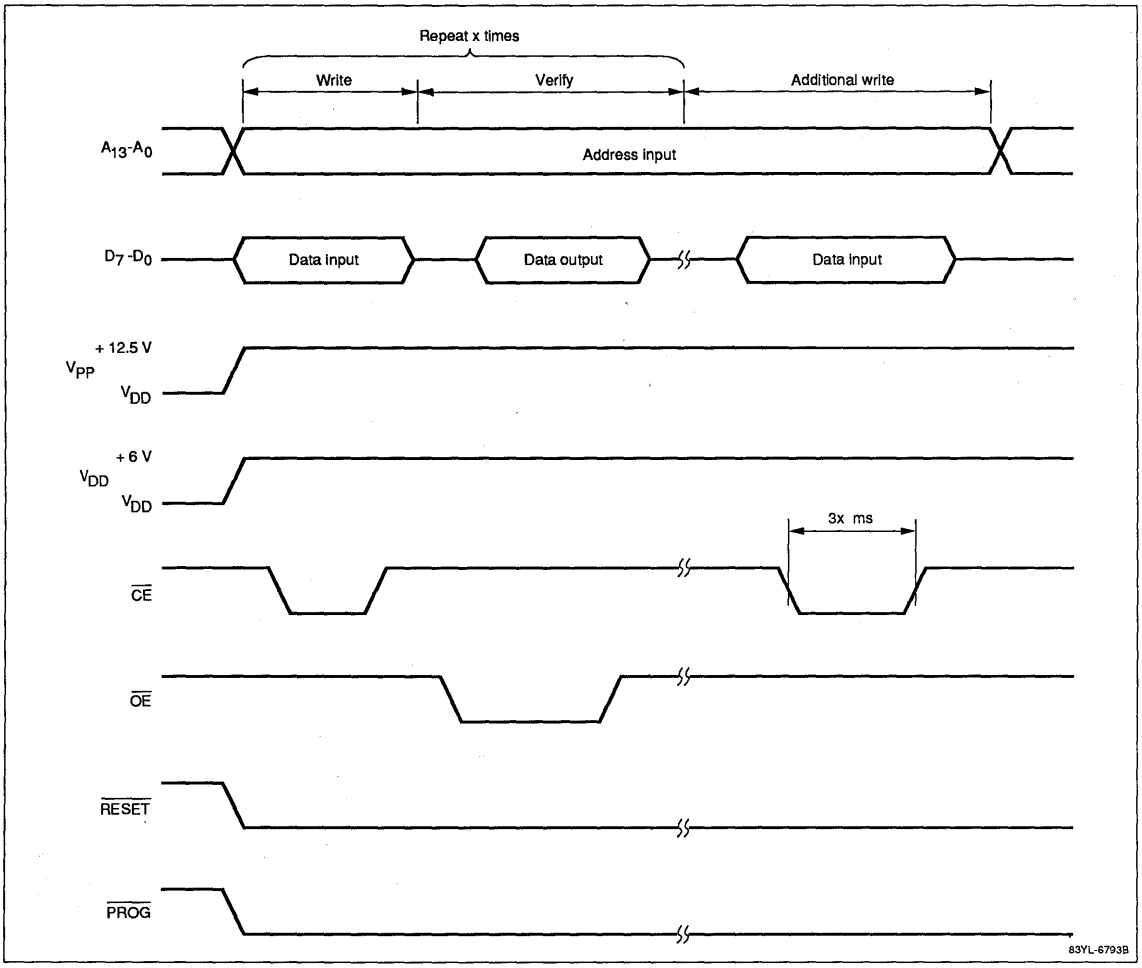
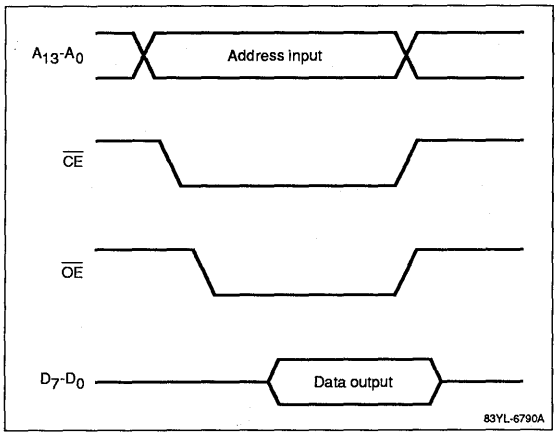


Figure 3. EPROM Read Timing



INSTALLATION

Direct soldering to pins of the μPD70P322 is not allowed. The device must be installed in a socket.

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = 25°C

Supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD}+0.5 ≤ +7.0 V
Output voltage, V_O	−0.5 to V_{DD}+0.5 ≤ +7.0 V
Threshold voltage, V_{TH}	−0.5 to V_{DD}+0.5 ≤ +7.0 V
Output current low, I_{OL}	Each output pin 4.0 mA (total 50 mA)
Output current high, I_{OH}	Each output pin −2.0 mA (total −20 mA)
Operating temperature range, T_{OPT}	−40 to +85°C
Storage temperature range, T_{STG}	−65 to +150°C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Capacitance

T_A = 25°C; V_{DD} = 0 V

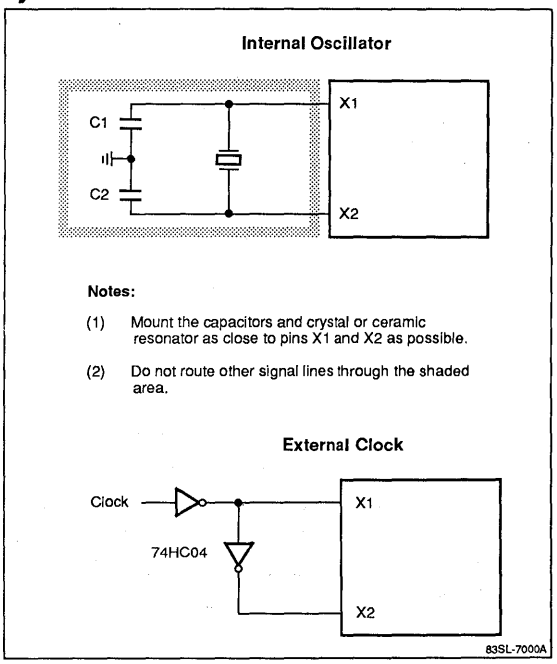
Parameter	Symbol	Max	Unit	Conditions
Input capacitance	C_I	10	pF	f_C = 1 MHz; unmeasured pins returned to ground
Output capacitance	C_O	20	pF	
I/O capacitance	C_{IO}	20	pF	

System Clock

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%; V_{SS} = 0 V; V_{TH} = 0 to V_{DD} + 1

	μPD70P322		μPD70P322-8		
Parameter	**Min**	**Max**	**Min**	**Max**	**Unit**
Internal Oscillator					
Frequency, f_{XX}	4	10	4	16	MHz
External Clock					
Frequency, f_X	4	10	4	16	MHz
Rise/fall time, t_R/t_F	0	10	0	10	ns
X1 input, high/low level width, t_{OH}/t_{OL}	35	250	20	250	ns

System Clock Control Circuit



4c

Recommended Oscillator Components

Ceramic Resonator (Note 1)		Capacitors	
Manufacturer	**Product No.**	**C1 (pF)**	**C2 (pF)**
Kyocera	KBR-10.0M	33	33
Murata Mfg.	CSA.10.0MT	47	47
	CSA16.0MX040	30	30
TDK	FCR10.M2S	30	30
	FCR16.0M2S	15	6

Crystal (Note 2)		Capacitors	
Manufacturer	**Product No.**	**C1 (pF)**	**C2 (pF)**
Kinseki	HC-49/U	15	15
	HC-43/U	15	15

Notes:

(1) Ceramic resonator product no. includes the frequency: 10.0 or 16.0 MHz.

(2) Crystal frequencies: 10, 16 MHz.

DC Characteristics 1; V25/V35 Mode

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10 %

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2		V_{DD}	V	All except $\overline{RESET}$, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 V_{DD}		V_{DD}	V	$\overline{RESET}$, $P1_0$/NMI, X1, X2
Output voltage, low	V_{OL}			0.45	V	I_{OL} = 1.6 mA
Output voltage, high	V_{OH}	V_{DD} − 1.0			V	I_{OH} = −0.4 mA
Input current	I_{IN}			±20	μA	$\overline{EA}$, $P1_0$/NMI; V_{IN} = 0 to V_{DD}
Input leakage current	I_{LI}			±10	μA	All except $\overline{EA}$, $P1_0$/NMI; V_{IN} = 0 to V_{DD}
Output leakage current	I_{LO}			±10	μA	V_O = 0 to V_{DD}
V_{TH} supply current	I_{TH}		0.5	1.0	mA	V_{TH} = 0 to V_{DD}
V_{DD} supply current, μPD70P322	I_{DD1}		50	100	mA	Operation mode
	I_{DD2}		20	40	mA	HALT mode
	I_{DD3}		10	30	μA	STOP mode
V_{DD} supply current, μPD70P322-8	I_{DD1}		65	120	mA	Operation mode
	I_{DD2}		25	50	mA	HALT mode
	I_{DD3}		10	30	μA	STOP mode

DC Characteristics 2; EPROM Program Operation

T_A = 25 ±5°C; V_{DD} = 6.0 ±0.25 V; V_{PP} = 12.5 ±0.3 V

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Input voltage, high	V_{IH}	2.2		V_{DD} + 0.3	V	
Input voltage, low	V_{IL}	−0.3		0.8	V	
Input leakage current	I_{LI}			10	μA	V_{IH} = 0 to V_{DD}
Output voltage, high	V_{OH}	V_{DD} − 1			V	I_{OH} = −400 μA
Output voltage, low	V_{OL}			0.45	V	I_{OL} = 1.6 mA
V_{DD} supply current	I_{DD}		40		mA	
V_{PP} supply current	I_{PP}		30		mA	$\overline{CE}$ = V_{IL}, $\overline{OE}$ = V_{IH}

DC Characteristics 3; EPROM Read Operation

T_A = 25 ±5°C

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Power supply voltage	V_{DD}	4.5	5.0	5.5	V	
Write power supply voltage	V_{PP}				V	$V_{PP} = V_{DD}$
Input voltage, high	V_{IH}	2.2		V_{DD} + 0.3	V	
Input voltage, low	V_{IL}	−0.3		0.8	V	
Input leakage current	I_{LI}			10	μA	V_{IN} = 0 to V_{DD}
Output voltage, high	V_{OH}	V_{DD} − 1			V	I_{OH} = −400 μA
Output voltage, low	V_{OL}			0.45	V	I_{OL} = 1.6 mA
Output leakage current	I_{LO}			10	μA	V_{OUT} = 0 to V_{DD}; $\overline{OE} = V_{IH}$
V_{DD} supply current	I_{DD}			40	mA	$\overline{CE} = V_{IL}$; $V_{IN} = V_{IH}$
V_{PP} supply current	I_{PP}		1	100	μA	$V_{PP} = V_{DD}$

AC Characteristics 1; V25/V35 Mode

T_A = −10 to +70°C; f_{CLK} = 0.5 to 5 MHz with V_{DD} = 5 V ±10%; f_{CLK} = 5 to 8 MHz with V_{DD} = 5 V ±5%

		70P322		70P322-8			
Parameter	Symbol	Min	Max	Min	Max	Unit	Conditions
Input rise, fall times	t_{IR}, t_{IF}		20		20	ns	Except X1, X2, $\overline{RESET}$, NMI
Input rise, fall times (Schmitt)	t_{IRS}, t_{IFS}		30		30	ns	$\overline{RESET}$, NMI
Output rise, fall times	t_{OR}, t_{OF}		20		20	ns	Except CLKOUT
X1 cycle time	t_{CYX}	98	250	62	250	ns	
X1 width, low	t_{WXL}	35		20		ns	
X1 width, high	t_{WXH}	35		20		ns	
X1 rise, fall times	t_{XR}, t_{XF}		20		20	ns	
CLKOUT cycle time	t_{CYK}	200	2000	125	2000	ns	CLKOUT = $f_X/2$
CLKOUT width, low	t_{WKL}	0.5T − 15		0.5T − 15		ns	T = t_{CYK}
CLKOUT width, high	t_{WKH}	0.5T − 15		0.5T − 15		ns	
CLKOUT rise, fall times	t_{KR}, t_{KF}		15		15	ns	

4c

AC Characteristics 2; V25 Mode

T_A = −10 to +70°C; C_L = 100 pF (max); T = t_{CYK}; n = number of wait states inserted
f_{CLK} = 0.5 to 5 MHz with V_{DD} = 5 V ±10%; f_{CLK} = 5 to 8 MHz with V_{DD} = 5 V ±5%

Parameter	Symbol	Min	Max	Unit	Conditions
Address delay time	t_{DKA}		90	ns	
Address valid to input data valid	t_{DADR}		(n+1.5)T − 90	ns	
$\overline{MREQ}$ to data delay time	t_{DMRD}		(n+1)T − 75	ns	
$\overline{MSTB}$ to data delay time	t_{DMSD}		(n+0.5)T − 75	ns	
$\overline{MREQ}$ to $\overline{TC}$ delay time	t_{DMRTC}		0.5T + 50	ns	
$\overline{MREQ}$ to $\overline{MSTB}$ delay time	t_{DMRMS}	0.5T − 35	0.5T + 35	ns	
$\overline{MREQ}$ width, low	t_{WMRL}	(n+1)T − 30		ns	
Address hold time	t_{HMA}	0.5T − 30		ns	
Input data hold time	t_{HMDR}	0		ns	
Next control setup time	t_{SCC}	T − 25		ns	
$\overline{TC}$ width, low	t_{WTCL}	2T − 30		ns	

AC Characteristics 2; V25 Mode (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
Address data output	t_{DADW}		0.5T + 50	ns	
$\overline{MREQ}$ delay time	t_{DAMR}	0.5T – 30		ns	
$\overline{MSTB}$ delay time	t_{DAMS}	T – 30		ns	
$\overline{MSTB}$ width, low	t_{WMSL}	(n+0.5)T – 30		ns	
Data output setup time	t_{SDM}	(n+1)T – 50		ns	
Data output hold time	t_{HMDW}	0.5T – 30		ns	
$\overline{IOSTB}$ delay time	t_{DAIS}	0.5T – 30		ns	
$\overline{IOSTB}$ to data input	t_{DISD}		(n+1)T – 90	ns	
$\overline{IOSTB}$ width, low	t_{WISL}	(n+1)T – 30		ns	
Address hold time	t_{HISA}	0.5T – 30		ns	
Data input hold time	t_{HISDR}	0		ns	
Output data setup time	t_{SDIS}	(n+1)T – 50		ns	
Output data hold time	t_{HISDW}	0.5T – 30		ns	
Next DMARQ setup time	t_{SDADQ}		T	ns	Demand mode
DMARQ hold time	t_{HDADQ}	0		ns	Demand mode
$\overline{DMAAK}$ read width, low	t_{WDMRL}	(n+1.5)T – 30		ns	
$\overline{DMAAK}$ to $\overline{TC}$ delay time	t_{DDATC}		0.5T + 50	ns	
$\overline{DMAAK}$ write width, low	t_{WDMWL}	(n+1)T – 30		ns	
$\overline{REFRQ}$ delay time	t_{DARF}	0.5T – 30		ns	
$\overline{REFRQ}$ width, low	t_{WRFL}	(n+1)T – 30		ns	
Address hold time	t_{HRFA}	0.5T – 30		ns	
$\overline{RESET}$ width, low	t_{WRSL1}	30		ms	STOP mode release/Power-on reset
$\overline{RESET}$ width, low	t_{WRSL2}	5		μs	System reset
$\overline{MREQ}$, $\overline{IOSTB}$ to READY setup time	t_{SCRY}		(n – 1)T – 100	ns	n ≥ 2
$\overline{MREQ}$, $\overline{IOSTB}$ to READY hold time	t_{HCRY}	(n – 1)T		ns	n ≥ 2
$\overline{HLDAK}$ output delay time	t_{DKHA}		80	ns	
BUS control float to $\overline{HLDAK}$ ↓	t_{CFHA}	T – 50		ns	
$\overline{HLDAK}$ ↑ to control output time	t_{DHAC}	T – 50		ns	
HLDRQ ↓ to control output time	t_{DHQC}	3T + 30		ns	
$\overline{HLDAK}$ width, low	t_{WHAL}	T		ns	
HLDRQ setup time	t_{SHQK}	30		ns	
HLDRQ to $\overline{HLDAK}$ delay time	t_{DHQHA}		3T + 160	ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
INTP, DMARQ setup time	t_{SIQK}	30		ns	
INTP, DMARQ width, high	t_{WIQH}	8T		ns	
INTP, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{POLL}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{CTS}$ width, low	t_{WCTL}	2T		ns	

AC Characteristics 2; V25 Mode (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
INT setup time	t_{SIRK}	30		ns	
INT hold time	t_{HIAIQ}	0		ns	
$\overline{INTAK}$ width, low	t_{WIAL}	2T – 30		ns	
$\overline{INTAK}$ delay time	t_{DKIA}		80	ns	
$\overline{INTAK}$ width, high	t_{WIAH}	T – 30		ns	
$\overline{INTAK}$ to data delay time	t_{DIAD}		2T – 130	ns	
$\overline{INTAK}$ to data hold time	t_{HIAD}	0	0.5T	ns	
$\overline{SCK0}$ cycle time	t_{CYTK}	1000		ns	
$\overline{SCK0}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{SCK0}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	
TxD hold time	t_{HTKD}	20		ns	
$\overline{CTS0}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{CTS0}$ (RSCK) width, high	t_{WSRH}	420		ns	
$\overline{CTS0}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

4c

AC Characteristics 3; V35 Mode

T_A = –10 to +70°C; C_L = 100 pF (max); T = t_{CYK}; n = number of wait states inserted
f_{CLK} = 0.5 to 5 MHz with V_{DD} = 5 V ±10%; f_{CLK} = 5 to 8 MHz with V_{DD} = 5 V ±5%

Parameter	Symbol	Min	Max	Unit	Conditions
Address delay time	t_{DKA}		90	ns	
Address valid to input data valid	t_{DADR}		(n+1.5)T – 90	ns	
$\overline{MREQ}$ to data delay time	t_{DMRD}		(n+2)T – 75	ns	
$\overline{MSTB}$ to data delay time	t_{DMSD}		(n+1)T – 75	ns	
$\overline{MREQ}$ to $\overline{TC}$ delay time	t_{DMRTC}		0.5T + 50	ns	
$\overline{MREQ}$ to $\overline{MSTB}$ delay time	t_{DMRMS1}	T – 35	T + 35	ns	Read operation
	t_{DMRMS2}	(n + 1)T – 35	(n + 1)T + 35	ns	Write operation
	t_{DMRMS}	0.5 T – 30		ns	
$\overline{MREQ}$ width, low	t_{WMRL}	(n+2)T – 30		ns	
Address hold time	t_{HMA}	0.5T – 30		ns	
Input data hold time	t_{HMDR}	0		ns	
Next control setup time	t_{SCC}	T – 25		ns	
$\overline{TC}$ width, low	t_{WTCL}	2T – 30		ns	
Address data output	t_{DADW}		0.5T + 50	ns	
$\overline{MREQ}$ delay time	t_{DAMR}	0.5T – 30		ns	
R/$\overline{W}$ to $\overline{MSTB}$ delay time	t_{DRMS}	0.5T – 30		ns	
	t_{DWMS}	(n + 0.5)T – 30		ns	
$\overline{MSTB}$ width, low	t_{WMSL1}	(n + 1)T – 30		ns	Read operation
	t_{WMSL2}	T – 30		ns	Write operation

AC Characteristics 3; V35 Mode (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
Data output setup time	t_{SDM}	(n+2)T – 50		ns	
Data output hold time	t_{HMDW}	0.5T – 50		ns	
$\overline{IOSTB}$ delay time	t_{DMRIS}	T – 35		ns	
$\overline{IOSTB}$ to data input	t_{DISD}		(n + 1) T – 90	ns	
$\overline{IOSTB}$ width, low	t_{WISL}	(n+1)T – 30		ns	
Address hold time	t_{HISA}	0.5T – 30		ns	
Data input hold time	t_{HISDR}	0		ns	
Output data setup time	t_{SDIS}	(n+2)T – 50		ns	
Output data hold time	t_{HISDR}	0		ns	
Next DMARQ setup time	t_{SDADQ}		T	ns	Demand mode
DMARQ hold time	t_{HDADQ}	0		ns	Demand mode
$\overline{DMAAK}$ read width, low	t_{WDMRL}	(n+1.5)T – 30		ns	
$\overline{DMAAK}$ to $\overline{TC}$ delay time	t_{DDATC}		0.5T + 50	ns	
$\overline{DMAAK}$ write width, low	t_{WDMWL}	(n+2)T – 30		ns	
$\overline{REFRQ}$ delay time	t_{DARF}	0.5T – 30		ns	
$\overline{REFRQ}$ width, low	t_{WRFL}	(n+1)T – 30		ns	
Address hold time	t_{HRFA}	0.5T – 30		ns	
$\overline{RESET}$ width, low	t_{WRSL1}	30		ms	STOP mode release/Power-on reset
$\overline{RESET}$ width, low	t_{WRSL2}	5		μs	System reset
$\overline{MREQ}$, $\overline{IOSTB}$ to READY setup time	t_{SCRY}		nT – 100	ns	n ≥ 2
$\overline{MREQ}$, $\overline{IOSTB}$ to READY hold time	t_{HCRY}	nT		ns	n ≥ 2
$\overline{HLDAK}$ output delay time	t_{DKHA}		80	ns	
BUS control float to $\overline{HLDAK}$ ↓	t_{CFHA}	T – 50		ns	
$\overline{HLDAK}$ ↑ to control output time	t_{DHAC}	T – 50		ns	
HLDRQ ↓ to control output time	t_{DHQC}	3T + 30		ns	
$\overline{HLDAK}$ width, low	t_{WHAL}	T		ns	
HLDRQ setup time	t_{SHQK}	30		ns	
HLDRQ to $\overline{HLDAK}$ delay time	t_{DHQHA}		3T + 160	ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
INTP, DMARQ setup time	t_{STQK}	30		ns	
INTP, DMARQ width, high	t_{WIQH}	8T		ns	
INTP, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{POLL}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{CTS}$ width, low	t_{WCTL}	2T		ns	
INT setup time	t_{SIRK}	30		ns	
INT hold time	t_{HAIQ}	0		ns	
$\overline{INTAK}$ width, low	t_{WIAL}	2T – 30		ns	
$\overline{INTAK}$ delay time	t_{DKIA}		80	ns	

AC Characteristics 3; V35 Mode (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{INTAK}$ width, high	t_{WIAH}	T – 30		ns	
$\overline{INTAK}$ to data delay time	t_{DIAD}		2T – 130	ns	
$\overline{INTAK}$ to data hold time	t_{HIAD}	0	0.5T	ns	
$\overline{SCK0}$ cycle time	t_{CYTX}	1000		ns	
$\overline{SCK0}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{SCK0}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	
TxD hold time	t_{HTKD}	20		ns	
$\overline{CTS0}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{CTS0}$ (RSCK) width, high	t_{WSRH}	420		ns	
$\overline{CTS0}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

AC Characteristics 4; EPROM Program Operation

T_A = 25 ±5°C, V_{DD} = 6.0 ±0.25 V; V_{PP} = 12.5 ±0.3 V

Parameter	Symbol	Min	Typ	Max	Unit	Condition
Address setup time to $\overline{CE}$ ↓	t_{AS}	2			μs	
$\overline{OE}$ setup time	t_{OES}	2			μs	
Data input setup time to $\overline{CE}$ ↓	t_{DS}	2			μs	
Address retention time	t_{AH}	2			μs	
Data input retention time	t_{DH}	2			μs	
$\overline{OE}$ to data output float delay	t_{DF}	0		1	μs	
V_{PP} setup time to $\overline{CE}$ ↓	t_{VPS}	2			μs	
V_{DD} setup time to $\overline{CE}$ ↓	t_{VDS}	2			μs	
Initial program pulse width	t_{PW}	0.95	1.0	1.05	ms	
Additional program pulse width	t_{OPW}	2.85		78.75	ms	
$\overline{OE}$ to data output delay time	t_{OE}			2	μs	

AC Characteristics 5; EPROM Read Operation

T_A = 25 ±5°C, V_{DD} = 5.0 ±0.5 V; V_{PP} = V_{DD}

Parameter	Symbol	Min	Typ	Max	Unit	Condition
Address to data output delay time	t_{ACC}			2	μs	$\overline{CE}$ = $\overline{OE}$ = V_{IL}
$\overline{CE}$ to data output delay time	t_{CE}			2	μs	$\overline{OE}$ = V_{IL}
$\overline{OE}$ to data output delay time	t_{OE}			1	μs	$\overline{CE}$ = V_{IL}
$\overline{OE}$ to data output float delay	t_{DF}	0		1	μs	$\overline{CE}$ = V_{IL}
Address to output retention	t_{OH}	0			μs	$\overline{CE}$ = $\overline{OE}$ = V_{IL}

Figure 4. EPROM Program Operation Timing

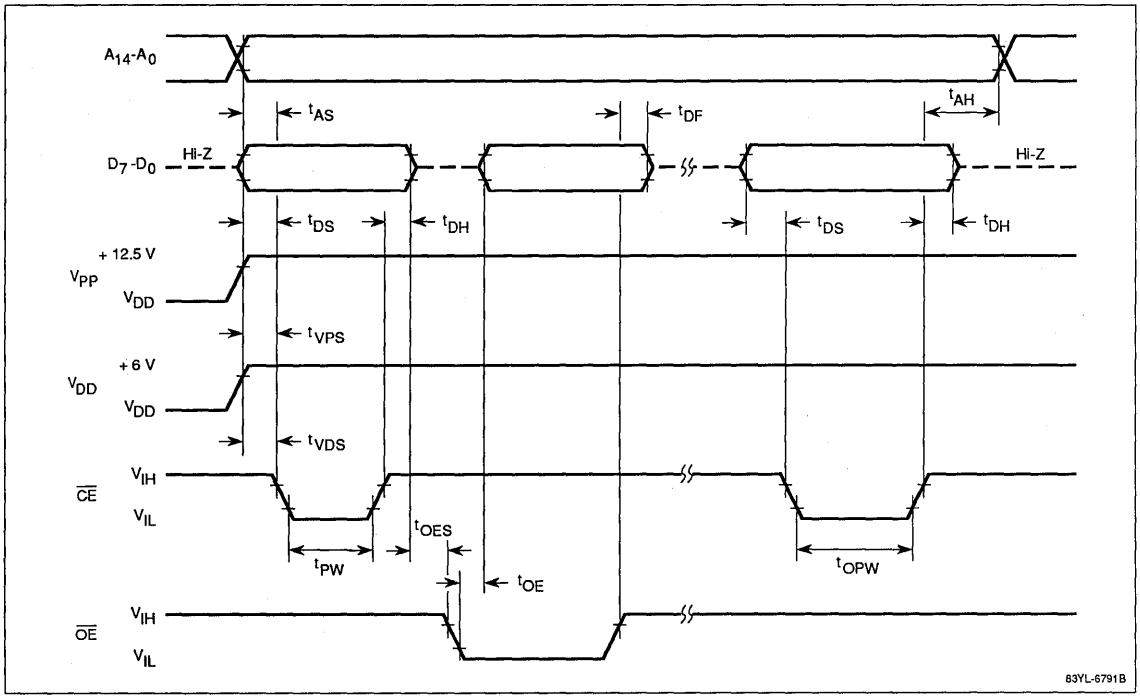
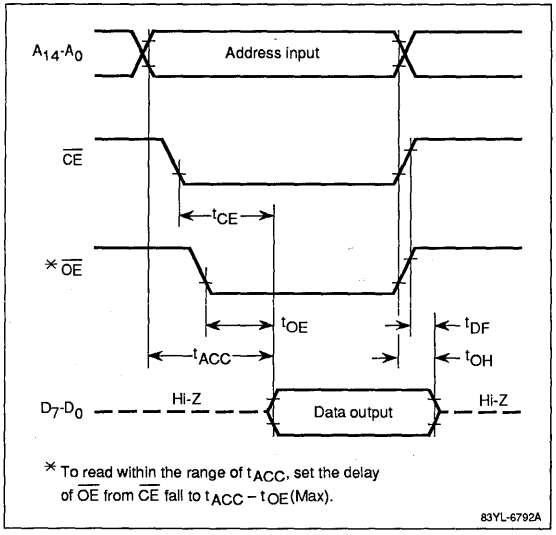


Figure 5. EPROM Read Operation Timing



Comparator Characteristics

T_A = −10 to 70°C; f_{CLK} = 0.5 to 5 MHz with V_{DD} = 5 V ±10%; f_{CLK} = 5 to 8 MHz with V_{DD} = 5 V ±5%

Parameter	Symbol	Min	Max	Unit
Accuracy	VA_{COMP}	–	±100	mV
Threshold voltage	V_{TH}	0	V_{DD}+0.1	V
Comparison time	t_{COMP}	64	65	t_{CYK}
PT input voltage	V_{IPT}	0	V_{DD}	V

Data Memory STOP Mode; Low Supply Voltage Data Retention

T_A = −10 to +70°C

Parameter	Symbol	Min	Max	Unit
Data retention supply voltage	V_{DDDR}	2.5	5.5	V
V_{DD} rise, fall time	t_{RVD}, t_{FVD}	200		μs

NEC

NEC Electronics Inc.

μPD70325 (V25 Plus) 16-Bit Microcomputer: High-Speed DMA, Single-Chip, CMOS

Description

The μPD70325 (V25 Plus) is a high-performance, 16-bit, single-chip microcomputer with an 8-bit external data bus. The μPD70325 is fully software compatible with the μPD70108/116 (V20®/30®) as well as the μPD70320/330 (V25™/35™). The V25 Plus microcomputer demonstrates numerous enhancements over the standard V25; however, it maintains strict pin compatibility with its predecessor, the V25.

The V25 Plus offers improved DMA transfer rates to 5 megabytes/second, additional serial channel status flags, improved memory access timing, and enhanced software control of register bank context switching.

The μPD70325 has the same complement of internal peripherals as the V25 and maintains compatibility with existing drivers; however, some modification of DMA device drivers may be necessary. The μPD70325 does not offer on-chip ROM or EPROM.

Features

- 16-bit CPU and internal data paths
- Functional and pin compatibility with V25
- Software compatible with μPD8086
- New and enhanced V-Series instructions
- 6-byte prefetch queue
- Two-channel high-speed DMA controller
- Minimum instruction cycle
  - — 250 ns at 8 MHz
  - — 200 ns at 10 MHz
- Internal 256-byte RAM memory
- 1-megabyte memory address space
- Eight internal memory-mapped register banks
- Four multifunction I/O ports
  - — 8-bit analog comparator port
  - — 20 bidirectional port lines
  - — Four input-only port lines
- Two independent full-duplex serial channels
- Priority interrupt controller
  - — Standard vectored service
  - — Register bank switching
  - — Macroservice
- Pseudo SRAM and DRAM refresh controller
- Two 16-bit timers
- On-chip time base counter
- Programmable wait state generator
- Two standby modes: STOP and HALT

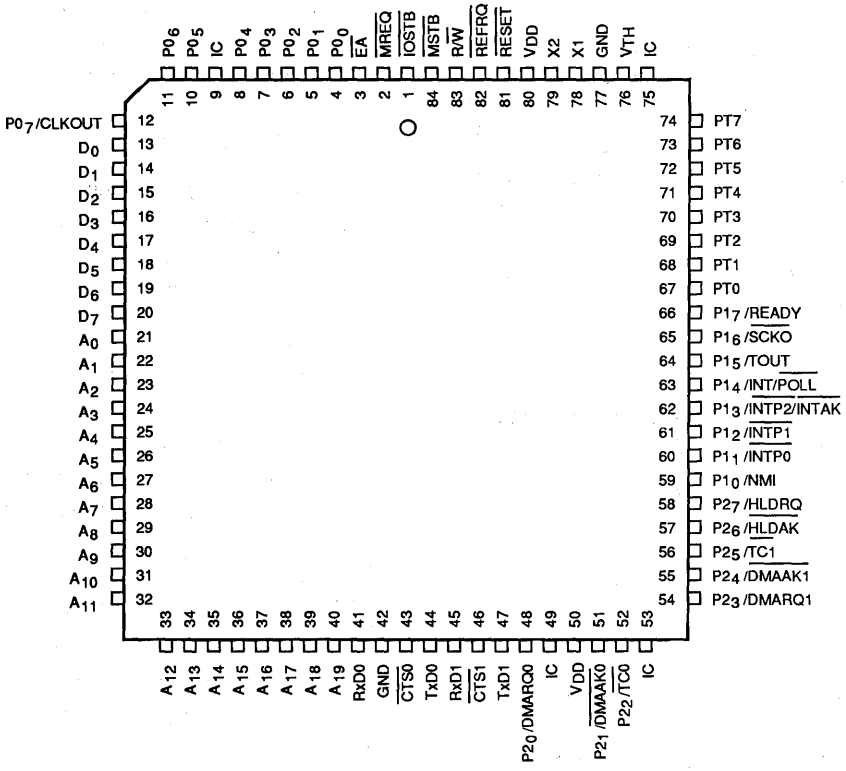
Ordering Information

Part Number	Clock (MHz)	Package
μPD70325L-8	8	84-pin PLCC
L-10	10	
GJ-8	8	94-pin plastic QFP
GJ-10	10	

V20 and V30 are registered trademarks of NEC Corporation.
V25 and V35 are trademarks of NEC Corporation.

Pin Configurations

84-Pin PLCC



Notes:

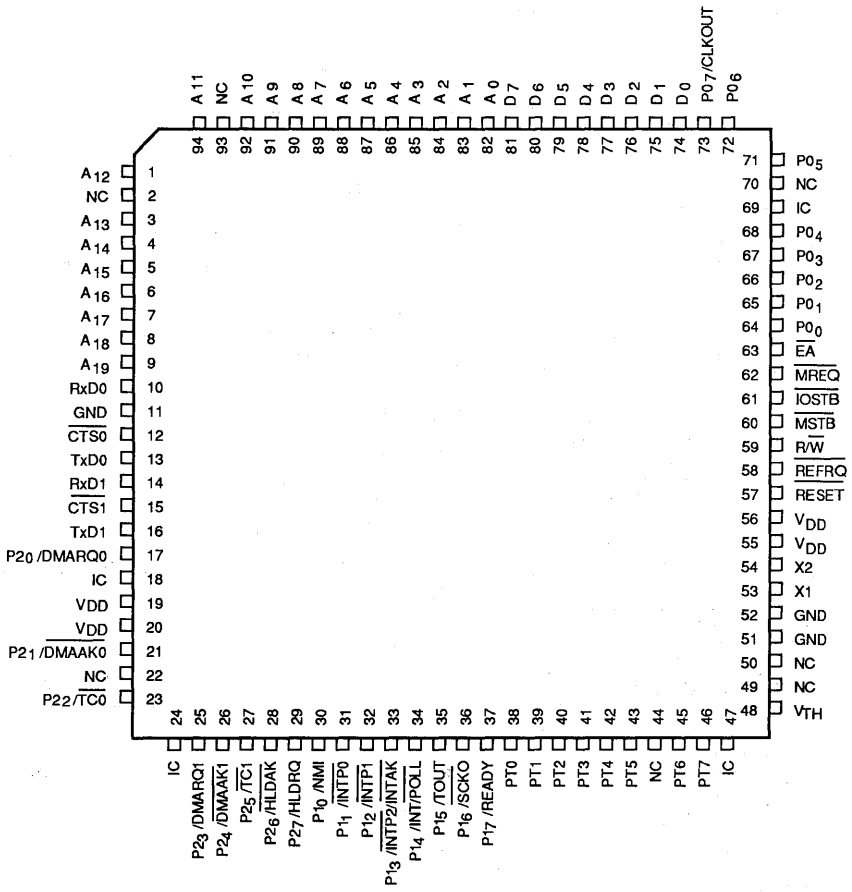
(1) Pin functions are identical to μPD70320.

(2) All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

(3) $\overline{EA}$ must be tied low because μPD70325 does not support internal ROM or EPROM.

83SL-6689B

94-Pin Plastic QFP



Notes:

(1) Pin functions are identical to μPD70320.

(2) All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

(3) $\overline{EA}$ must be tied low because μPD70325 does not support internal ROM or EPROM.

83SL-6690B

4d

Pin Identification

Symbol	Function
A_0-A_{19}	Address bus outputs
CLKOUT	System clock output
$\overline{CTS0}$	Clear to send channel 0 input
$\overline{CTS1}$	Clear to send channel 1 input
D_0-D_7	Bidirectional data bus
$\overline{EA}$	External access
$\overline{IOSTB}$	I/O strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory strobe output
$P0_0$-$P0_7$	I/O port 0
$P1_0$/NMI	Port 1 input line; nonmaskable interrupt
$P1_1$-$P1_2$/INTP0-INTP1	Port 1 input lines; external interrupt input lines
$P1_3$/INTP2/$\overline{INTAK}$	Port 1 input line; external interrupt input line; interrupt acknowledge output
$P1_4$/INT/$\overline{POLL}$	I/O port 1; interrupt request input; I/O poll input
$P1_5$/TOUT	I/O port 1; timer out
$P1_6$/$\overline{SCKO}$	I/O port 1; serial clock output
$P1_7$/READY	I/O port 1; ready input
$P2_0$/DMARQ0	I/O port 2; DMA request 0
$P2_1$/$\overline{DMAAK0}$	I/O port 2; DMA acknowledge 0
$P2_2$/$\overline{TC0}$	I/O port 2; DMA terminal count 0
$P2_3$/DMARQ1	I/O port 2; DMA request 1
$P2_4$/$\overline{DMAAK1}$	I/O port 2; DMA acknowledge 1
$P2_5$/$\overline{TC1}$	I/O port 2; DMA terminal count 1
$P2_6$/$\overline{HLDAK}$	I/O port 2; hold acknowledge output
$P2_7$/HLDRQ	I/O port 2; hold request input
PT0-PT7	Comparator port input lines
$\overline{REFRQ}$	Refresh pulse output
$\overline{RESET}$	Reset input
RxD0	Serial receive data channel 0 input
RxD1	Serial receive data channel 1 input
R/$\overline{W}$	Read/Write output
TxD0	Serial transmit data, channel 0 input
TxD1	Serial transmit data, channel 1 input
X1, X2	Crystal connection terminals
V_{DD}	Positive power supply voltage
V_{TH}	Threshold voltage input
GND	Ground reference
IC	Internal connection

PIN FUNCTIONS

A_0-A_{19} (Address Bus)

A_0-A_{19} is the nonmultiplexed 20-bit address bus used to access all external devices.

CLKOUT (System Clock)

This is the internal system clock. It can be used to synchronize external devices to the CPU.

$\overline{CTS}$n, RxDn, TxDn, $\overline{SCKO}$ (Clear to Send, Receive Data, Transmit Data, Serial Clock Out)

The two serial ports (channels 0 and 1) use these lines for transmitting and receiving data, handshaking, and serial clock output.

D_0-D_7 (Data Bus)

D_0-D_7 is the 8-bit external data bus.

DMARQn, $\overline{DMAAK}$n, TCn (DMA Request, DMA Acknowledge, Terminal Count)

These are the control signals to and from the on-chip DMA controller.

$\overline{EA}$(External Access)

If this pin is low on reset, the μPD70322 (V25) will execute program code from external memory instead of internal ROM.

Because the V25 Plus does not support internal ROM, the $\overline{EA}$ pin must be fixed low in hardware.

$\overline{HLDAK}$ (Hold Acknowledge)

The $\overline{HLDAK}$ output (active low) informs external devices that the CPU has released the system bus.

HLDRQ (Hold Request)

The HLDRQ input (active high) is used by external devices to request the CPU to release the system bus to an external bus master. The following lines go into a high-impedance status with internal 4.7-kΩ pullup resistors: A_0-A_{19}, D_0-D_7, $\overline{MREQ}$, R/$\overline{W}$, $\overline{MSTB}$, $\overline{REFRQ}$, and $\overline{IOSTB}$.

INT (Interrupt Request)

INT is a maskable, active-high, vectored interrupt request . After assertion, external hardware must provide the interrupt vector number.

The INT pin allows direct connection of slave μPD71059 interrupt controllers.

$\overline{\text{INTAK}}$ (Interrupt Acknowledge)

After INT is asserted, the CPU will respond with $\overline{\text{INTAK}}$ (active low) to inform external devices that the interrupt request has been granted.

INTP0-INTP2 (External Interrupt)

INTP0-INTP2 allow external devices to generate interrupts. Each can be programmed to be rising or falling edge triggered.

$\overline{\text{IOSTB}}$ (I/O Strobe)

$\overline{\text{IOSTB}}$ is asserted during read and write operations to external I/O.

$\overline{\text{MREQ}}$ (Memory Request)

$\overline{\text{MREQ}}$ (active low) informs external memory that the current bus cycle is a memory access bus cycle.

$\overline{\text{MSTB}}$ (Memory Strobe)

$\overline{\text{MSTB}}$ (active low) is asserted during read and write operations to external memory.

NMI (Nonmaskable Interrupt)

NMI cannot be masked through software and is typically used for emergency processing. Upon execution, the interrupt starting address is obtained from interrupt vector number 2. NMI can release the standby modes and can be programmed to be either rising or falling edge triggered.

$P0_0$-$P0_7$ (Port 0)

$P0_0$-$P0_7$ are the lines of port 0, an 8-bit bidirectional parallel I/O port.

$P1_0$-$P1_7$ (Port 1)

The status of $P1_0$-$P1_3$ can be read but these lines are always control functions. $P1_4$-$P1_7$ are the remaining lines of parallel port 1; each line is individually programmable as either an input, an output, or a control function.

$P2_0$-$P2_7$ (Port 2)

$P2_0$-$P2_7$ are the lines of port 2, an 8-bit bidirectional parallel I/O port. The lines can also be used as control signals for the on-chip DMA controller.

$\overline{\text{POLL}}$ (Poll)

Upon execution of the $\overline{\text{POLL}}$ instruction, the CPU checks the status of this pin and, if low, program execution continues. If high, the CPU checks the level of the line every five clock cycles until it is low. $\overline{\text{POLL}}$ can be used to synchronize program execution to external conditions.

PT0-PT7 (Comparator Port)

PT0-PT7 are inputs to the analog comparator port.

READY (Ready)

After READY is de-asserted low, the CPU synchronizes and inserts at least two wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than nominal μPD70325 bus cycles.

$\overline{\text{REFRQ}}$ (Refresh)

This active-low output pulse can refresh nonstatic RAM. It can be programmed to meet system specifications an is internally synchronized so that refresh cycles do not interfere with normal CPU operation.

$\overline{\text{RESET}}$ (Reset)

A low on $\overline{\text{RESET}}$ resets the CPU and all on-chip peripherals. $\overline{\text{RESET}}$ can also release the standby modes. After $\overline{\text{RESET}}$ returns high, program execution begins from address FFFF0H.

R/$\overline{\text{W}}$ (Read/Write)

R/$\overline{\text{W}}$ output allows external hardware to determine if the current operation is a read or a write cycle. It can also control the direction of bidirectional buffers.

TOUT (Timer Out)

TOUT is the square-wave output signal from the internal timer.

X1, X2 (Crystal Connections)

The internal clock generator requires an external crystal across these terminals. By programming the PRC register, the system clock frequency can be selected as the oscillator frequency (f_{OSC}) divided by 2, 4, or 8.

V_{DD} (Power Supply)

Two positive power supply pins (V_{DD}) reduce internal noise.

V_{TH} (Threshold Voltage)

The comparator port uses this pin to determine the analog reference point. The actual threshold to each comparator line is programmable to V_{TH} x n/16 where n = 1 to 16.

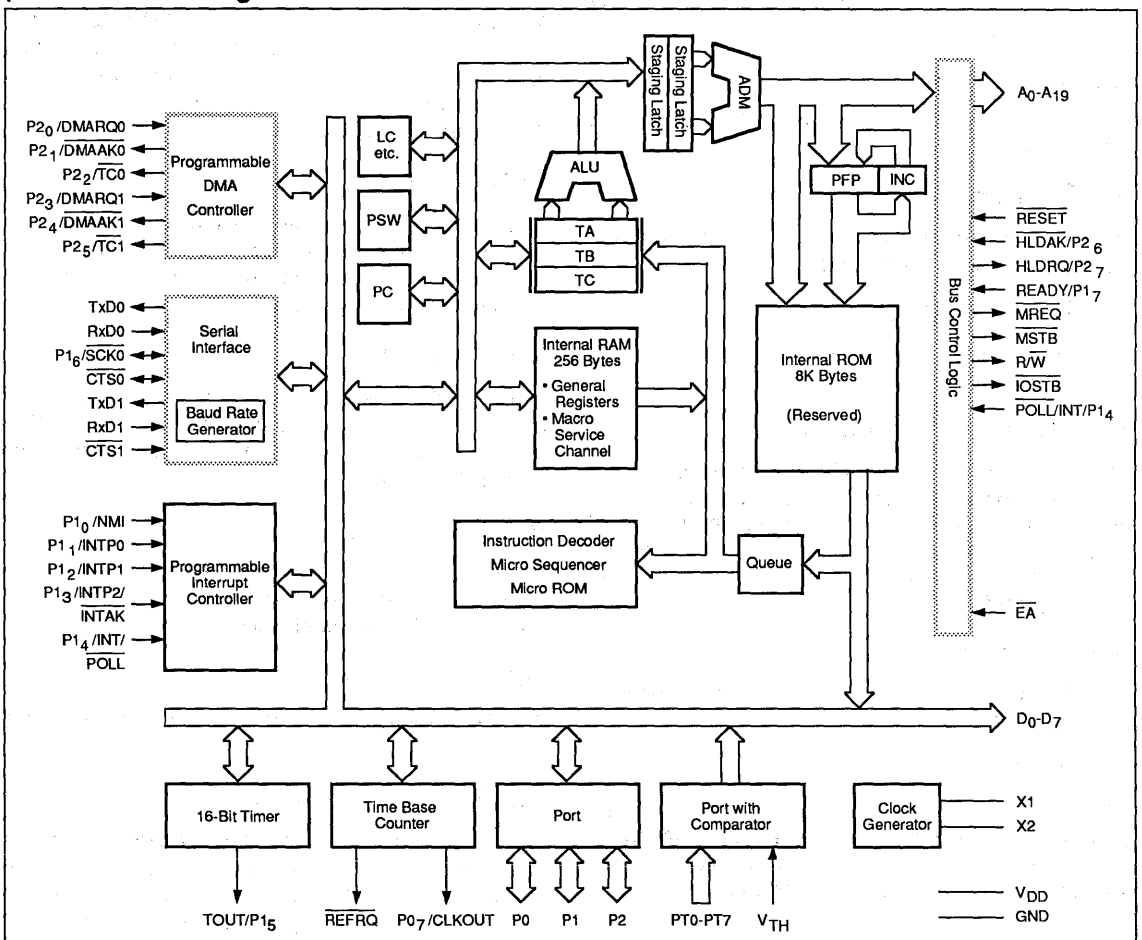
GND (Ground)

Two ground connections reduce internal noise.

IC (Internal Connection)

All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

μPD70325 Block Diagram



Notes:

(1) The μPD70325 (V25 Plus) is not a masked ROM product. Internal ROM is reserved and not accessible.

(2) Shaded blocks are functionally different on V25 Plus and V25.

FUNCTIONAL DESCRIPTION

The following features enable the μPD70325 to perform high-speed execution of instructions.

- Dual internal data bus
- 16- and 32-bit temporary registers/shifters
- 16-bit loop counter
- Program counter and prefetch pointer

Dual Data Bus

The μPD70325 has two internal 16-bit data buses: the main data bus and the secondary data bus. This reduces the processing time required for addition/subtraction and logical comparison instructions by one third over single-bus systems. The dual data bus method allows two operands to be fetched simultaneously from the general-purpose registers and transferred to the ALU.

16- and 32-Bit Temporary Registers/Shifters

The 16-bit temporary registers/shifters (TA and TB) allow high-speed execution of multiplication/division and shift/rotate instructions. Using the temporary registers, the μPD70325 can execute multiplication/division instructions about four times faster than with the microprogrammed method.

Loop Counter (LC)

The dedicated hardware loop counter (LC) counts the number of iterations for string operations and the number of shifts performed for multiple-bit shift/rotate instructions. The loop counter works with internal dedicated shifters to speed the processing of multiplication/division instructions.

Program Counter and Prefetch Pointer (PC and PFP)

The hardware PC addresses the memory location of the instruction to be executed next. The hardware PFP addresses the program memory location to be accessed by the instruction queued next. Several clock cycles are saved for branch, call, return, and break instructions.

Register Set

Figure 1 shows the eight banks of internal registers, which the μPD70325 has functionally mapped into internal RAM. Each bank contains general-purpose registers, pointer and index registers, segment registers, and save areas for context switching.

Although these memory locations may be accessed as normal RAM with the full set of memory addressing modes provided by the V25 family, the capability of context switching provides superior speed in register access. When used in the internal memory disabled state, many instructions execute considerably faster.

Eight macroservice channel control blocks are also mapped into register banks 0 and 1. The V25 Plus does not map the DMA channel control blocks into the internal RAM like the V25; instead, these control blocks are mapped into the special function register area.

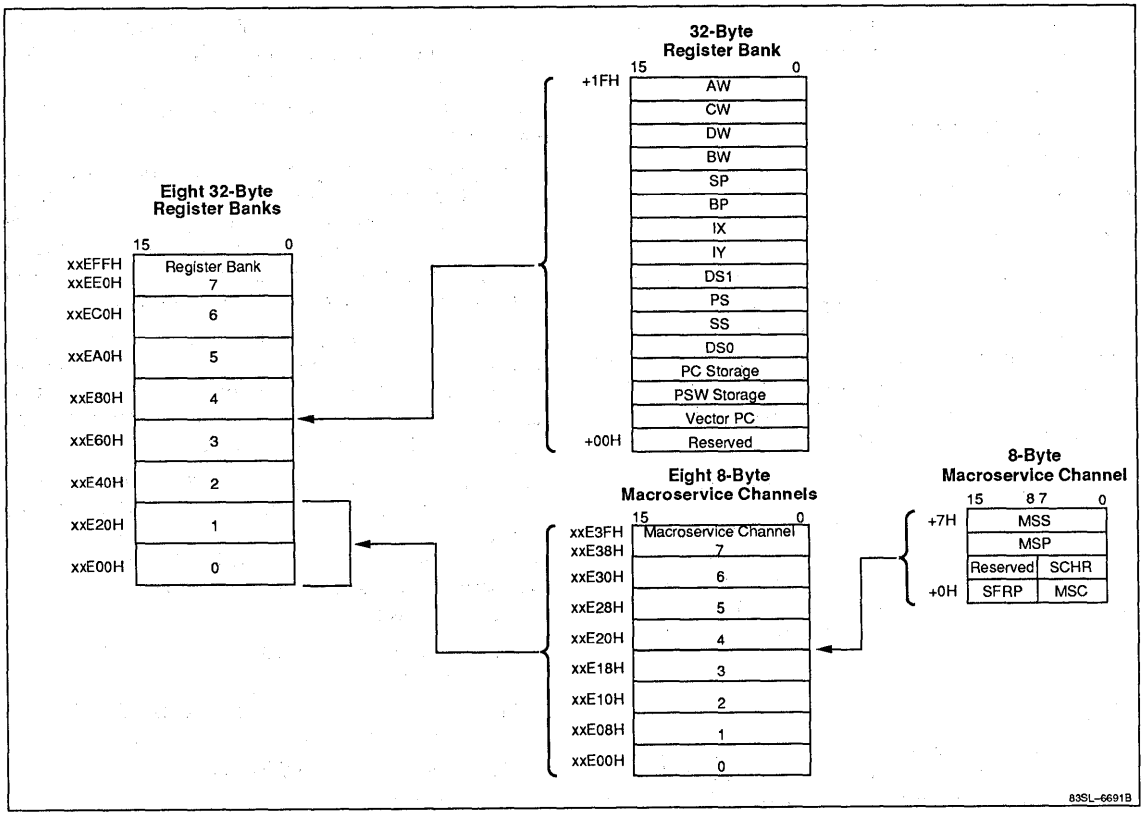
General-Purpose Registers (AW, BW, CW, DW). Four 16-bit general-purpose registers (AW, BW, CW, and DW) can serve as 16-bit registers or as four sets of dual 8-bit registers (AH, AL, BH, BL, CH, CL, DH, and DL). The instruction classes default to the following general-purpose registers.

- AW Word multiplication/division, word I/O, data conversion.
- AL Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation.
- AH Byte multiplication/division.
- BW Translation
- CW Loop control, branch, and repeat prefixes.
- CL Shift instructions, rotate instructions, BCD operations.
- DW Word multiplication/division, indirect I/O addressing.

Pointers (SP, BP) and Index Registers (IX, IY). These registers are 16-bit base pointers (SP, BP) or index registers (IX, IY) in based addressing, indexed addressing, and based indexed addressing. They are used as default registers under the following conditions.

- SP Stack operations
- IX Block transfer (source), BCD string operations
- IY Block transfer (destination), BCD string operations

Figure 1. Internal RAM Mapping



Segment Registers. The segment registers divide the 1M-byte address space into 64K-byte blocks. Each segment register functions as a base address to a block; the effective address is an offset from that base. Physical addresses are generated by shifting the associated segment register left by four binary digits and then adding the offset address. The segment registers and default offsets are listed below.

Segment Register	Default Offset
PS (Program Segment)	PC (Program Counter)
SS (Stack Segment)	SP and Effective Address
DS0 (Data Segment 0)	IX and Effective Address
DS1 (Data Segment 1)	IY and Effective Address

Save Registers (Save PC and Save PSW). Save PC and save PSW are used as the storage areas during register bank context-switching operations. The Vector PC save location contains the effective address of the interrupt service routine when register bank switching is used to service interrupts.

Program Counter (PC). The PC is a 16-bit binary counter that contains the offset address from the program segment of the next instruction to be executed. It is incremented every time an instruction is received from the queue. It is loaded with a new location whenever the branch, call, return, break, or interrupt is executed.

Program Status Word (PSW). The PSW contains status and control flags used by the CPU and two general-purpose user flags. The configuration of this 16-bit register is shown below.

15							8
1	RB2	RB1	RB0	V	DIR	IE	BRK

7							0
S	Z	F1	AC	F0	P	BRKI	CY

Status Flags		Control Flags	
V	Overflow bit	DIR	Direction of string processing
S	Sign	IE	Interrupt enable
Z	Zero	BRK	Break (after every instruction)
AC	Auxiliary carry	RBn	Current register bank flags
P	Parity	BRKI	I/O trap enable
CY	Carry	F0, F1	General-purpose user flags

The eight low-order bits of the PSW can be stored in register AH and restored using a MOV instruction. The only way to alter the RBn bits with software is to execute an RETRBI or RETI instruction.

Memory Map

The μPD70325 has a 20-bit address bus that can directly access 1 megabyte of memory. Figure 2 shows the memory map. The internal data area (IDA) is a 256-byte internal RAM area followed consecutively by a 256-byte special function register (SFR) area.

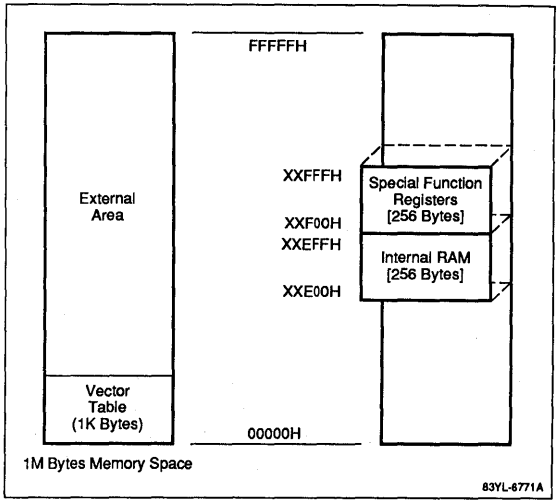
All the data and control registers for on-chip peripherals and I/O are mapped into the SFR area and accessed as RAM.

The IDA is dynamically relocatable in 4K-byte increments by changing the value in the internal data base (IDB) register. The value in this register is assigned as the uppermost eight bits of the IDA address. The IDB register is accessed from two memory locations, FFFFFH and XXFFFH, where XX is the value in the IDB register.

On reset, the internal data base register is set to FFH, which maps the IDA into the internal ROM space. However, since internal ROM is not present on the μPD70325, this does not present a problem. You can select any of the eight possible register banks, which occupy the entire internal RAM space. Multiple register bank selection allows faster interrupt processing and facilitates multitasking.

In large-scale systems where internal RAM is not required for data memory, internal RAM can be removed completely from the address space and dedicated entirely to register banks and control functions such as macroservice. You do this by clearing the RAMEN bit in the processor control register. When the RAMEN bit is cleared, internal RAM can only be accessed by register addressing or internal control processes. Many instructions execute faster when internal RAM is disabled.

Figure 2. Memory Map



INSTRUCTIONS

The μPD70325 instruction set is fully compatible with the V20 native mode instruction set. The V25 Plus is a superset of the μPD8086/8088 instruction set with different execution times and mnemonics.

The μPD70325 does not support the V20 8080 emulation mode.

4d

Enhanced Instructions

In addition to the μPD8086/8088 instructions, the μPD70325 provides the following enhanced instructions.

Instruction	Description
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP R	Pops 8 general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate data
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Unique Instructions

The μPD70325 provides the following unique instructions.

Instruction	Description
INS	Inserts bit field
EXT	Extracts bit field
ADD4S	Performs packed BCD string addition
SUB4S	Performs packed BCD string subtraction
CMP4S	Performs packed BCD string comparison
ROL4	Rotates BCD digit left
ROR4	Rotates BCD digit right
TEST1	Tests bit
SET1	Sets bit
CLR1	Clears bit
NOT1	Complements bit
BTCLR	Tests bit; if true, clear and branch
REPC	Repeat while carry set
REPNC	Repeat while carry cleared

Variable-Length Bit Field Operation Instructions

Bit fields are a variable-length data structure that can range from 1 to 16 bits. The μPD70325 supports two separate operations on bit fields: insertion (INS) and extraction (EXT). There are no restrictions on the position of the bit field in memory.

Separate segment, byte offset, and bit offset registers are used for insertion and extraction. Following the execution of these instructions, both the byte offset and bit offset are left pointing to the start of the next bit field, ready for the next operation. Bit field operation instructions are powerful and flexible and are therefore highly effective for graphics, high-level languages, and packing/unpacking applications.

Bit field insertion copies the bit field of specified length from the AW register to the bit field addressed by DS1:IY:reg8 (8-bit general-purpose register). The bit field length can be located in any byte register or supplied as immediate data. Following execution, both IY and reg8 are updated to point to the start of the next bit field.

Bit field extraction copies the bit field of specified length from the bit field addressed by DS0:IX:reg8 to the AW register. If the length of the bit field is less than 16 bits, the bit field is right justified with a zero fill. The bit field length can be located in any byte register or supplied as immediate data. Following execution, both IX and reg8 are updated to point to the start of the next bit field.

Figures 3 and 4 further illustrate bit field insertion and bit field extraction, respectively.

Packed BCD Instructions

Packed BCD instructions process packed BCD data either as strings (ADD4S, SUB4S, and CMP4S) or byte format operations (ROR4 and ROL4). Packed BCD strings may be 1 to 254 digits in length. The two BCD rotation instructions rotate a single BCD digit in the lower half of the AL register using the register or thememory operand.

Bit Manipulation Instructions

The μPD70325 provides five unique bit manipulation instructions that allow you to test, set, clear, or complement a single bit in a register or memory operand. This increases code readability as well as performance over the logical operations traditionally used to manipulate bit data. These instructions also give you additional control over on-chip peripherals.

Figure 3. Bit Field Insertion

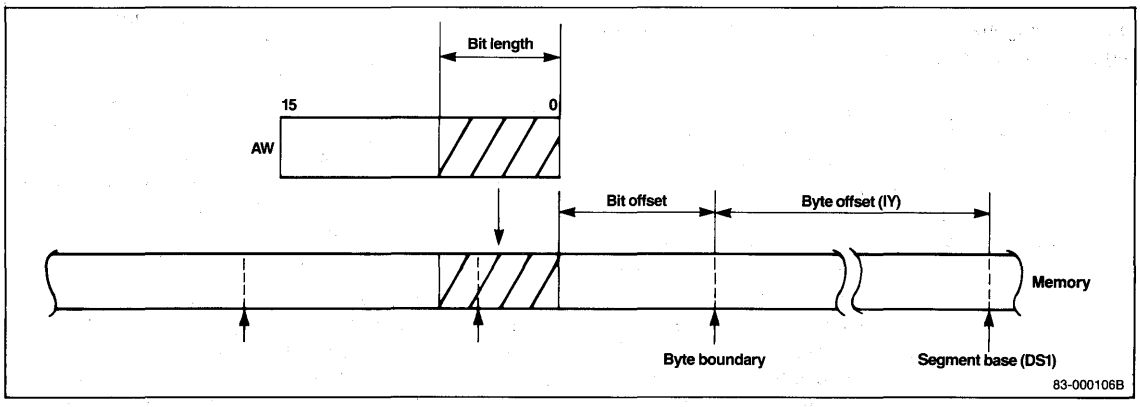
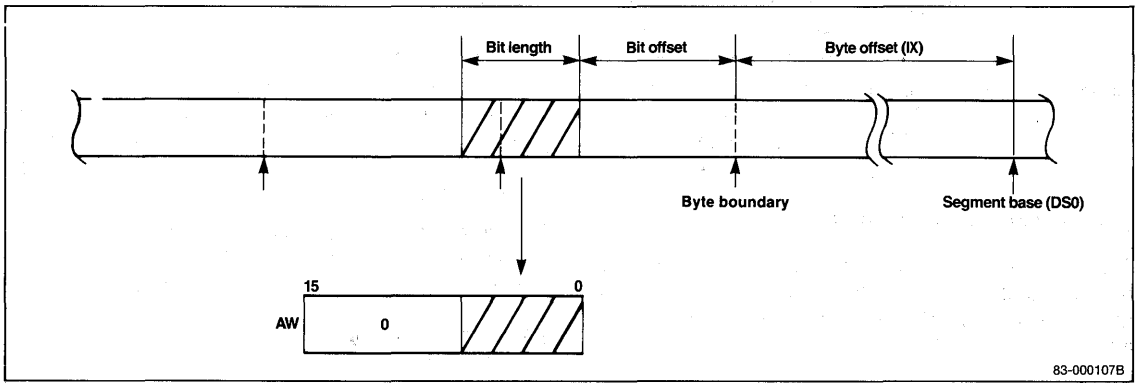


Figure 4. Bit Field Extraction



Additional Instructions

Besides the V20 instruction set, the μPD70325 provides the following additional instructions.

Instruction	Description
BTCLR Sfr.imm3, short-label	Bit test and if true, clear and branch; otherwise, no operation
STOP (no operand)	Power-down instruction; stops oscillator
RETRBI (no operand)	Return from register bank context switch interrupt
FINT (no operand)	Finished interrupt; after completion of a hardware interrupt request, this instruction must be used to reset the current priority bit in the in-service priority register, ISPR. Not for use with NMI or INT interrupt service routines.

Repeat Prefixes

Two repeat prefixes (REPC and REPNC) allow conditional block transfer instructions to use the state of the CY flag as the termination condition. This allows inequalities to be used when working on ordered data, thus increasing performance when searching and sorting algorithms.

4d

Bank Switch Instructions

The following instructions allow the effective use of the register banks for software interrupts and multitasking.

Instruction	Description
BRKCS reg 16	Performs a high-speed software interrupt with context switch to register bank indicated by lower 3 bits of register 16.
TSKSW reg 16	Performs a high-speed task switch to register bank indicated by lower 3 bits of register 16. The PC and PSW are saved in the old banks. PS and PSW save registers and the new PC and PSW values are retrieved from the new register bank's save areas.
MOVSPA	Transfers both SS and SP of old register bank to new register bank after bank has been switched by an interrupt or BRKCS instruction.
MOVSPB reg16	Transfers SS and SP of current register bank before switching to SS and SP of new register bank indicated by lower 3 bits of register 16.

INTERRUPT STRUCTURE

The μPD70325 can service interrupts generated by both hardware and software. Software interrupts are serviced through vectored interrupt processing. The following interrupts are provided.

Interrupt	Description
Divide error	The CPU traps if a divide error occurs as the result of a DIV or DIVU instruction.
Single step	The interrupt is generated after every instruction if the BRK bit in the PSW is set.
Overflow	Using the BRKV instruction, an interrupt can be generated as the result of an overflow.
Interrupt instructions	The BRK 3 and BRK imm8 instructions can generate interrupts.
Array bounds	The CHKIND instruction generates an interrupt if the specified array bounds are exceeded.
Escape trap	The CPU traps in an FP01,2 instruction to allow software to emulate the floating-point processor since the μPD70325 does not support an external hardware coprocessor.
I/O trap	If the I/O trap bit in the PSW is cleared, a trap is generated on every IN or OUT instruction. Software can then provide an updated peripheral address. This feature provides software portability between different systems.

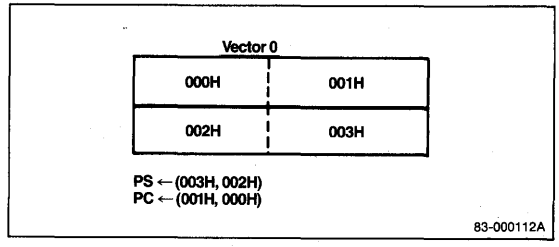
When executing software written for another system, it is better to implement I/O with on-chip peripherals to reduce external hardware requirements. However, since μPD70325 internal peripherals are memory mapped, software conversion may be difficult. The I/O trap feature allows for easy conversion from external peripherals to on-chip peripherals.

Interrupt Vectors

Table 1 lists the interrupt vectors beginning at physical address 00H. External memory is required to service these routines. By servicing interrupts via the macroservice function or context switching, this requirement can be eliminated.

Each interrupt vector is 4 bytes wide. To service a vectored interrupt, the lower addressed word is transferred to the PC and the upper word to the PS. See figure 5.

Figure 5. Interrupt Vector 0



Execution of a vectored interrupt occurs as follows:

(SP-1, SP-2) ← PSW
(SP-3, SP-4) ← PS
(SP-5, SP-6) ← PC
SP ← SP-6
IE ← 0, BRK ← 0
PS ← vector high bytes
PC ← vector low bytes

Table 1. Interrupt Vectors

Address	Vector	Assigned Use
00	0	Divide error
04	1	Break flag
08	2	NMI
0C	3	BRK3 instruction
10	4	BRKV instruction
14	5	CHKIND instruction
18	6	General purpose
1C	7	FPO instructions
20-2C	8-11	General purpose
30	12	INTSER0 (interrupt serial error, channel 0)
34	13	INTSR0 (interrupt serial receive, channel 0)
38	14	INTST0 (interrupt serial transmit, channel 0)
3C	15	General purpose
40	16	INTSER1 (interrupt serial error, channel 1)
44	17	INTSR1 (interrupt serial receive, channel 1)
48	18	INTST1 (interrupt serial transmit, channel 1)
4C	19	I/O trap
50	20	INTD0 (interrupt from DMA, channel 0)
54	21	INTD1 (interrupt from DMA, channel 1)
58	22	General purpose
5C	23	General purpose
60	24	INTP0 (interrupt from peripheral 0)
64	25	INTP1 (interrupt from peripheral 1)
68	26	INTP2 (interrupt from peripheral 2)
6C	27	General purpose
70	28	INTTU0 (interrupt from timer unit 0)
74	29	INTTU1 (interrupt from timer unit 1)
78	30	INTTU2 (interrupt from timer unit 2)
7C	31	INTTB (interrupt from time base counter)
080-3FF	32-255	General purpose

Hardware Interrupt Configuration

The V25 Plus features a high-performance on-chip controller capable of controlling multiple processing for interrupts from up to 17 different sources (5 external, 12 internal). The interrupt configuration includes system interrupts that are functionally compatible with those of the V20/V30 and unique high-performance microcontroller interrupts.

Interrupt Sources

The 17 interrupt sources are divided into groups for management by the interrupt controller. Using software, each of the groups can be assigned a priority from 0 (highest) to 7 (lowest). The priority of individual interrupts within a group is fixed in hardware.

Be careful when assigning the priority of a given interrupt group; the assignment is done by the three priority bits in only one interrupt control register in each group. If interrupts from different groups occur simultaneously and the groups have the same priority level, the priority is as shown in table 2.

Table 2. Interrupt Sources

	Interrupt Source (Priority Within Group)			
Group	1	2	3	Default Priority
Nonmaskable interrupt	NMI	–	–	0
Timer unit	INTTU0	INTTU1	INTTU2	1
DMA controller	INTD0	INTD1	–	2
External peripheral interrupt	INTP0	INTP1	INTP2	3
Serial channel 0	INTSER0	INTSR0	INTST0	4
Serial channel 1	INTSER1	INTSR1	INTST1	5
Time base counter	INTTB	–	–	6
Interrupt request	INT	–	–	7

The priority of the currently active interrupt is stored in the ISPR special function register. Bits PR_7-PR_0 correspond to the eight possible interrupt request priorities. The ISPR keeps track of the priority of active interrupts by setting the appropriate bit of this register. The address of this 8-bit register is xxFFCH, and the format is shown below.

PR_7	PR_6	PR_5	PR_4	PR_3	PR_2	PR_1	PR_0

NMI and INT are system type external vectored interrupts. NMI is not maskable via software, and is also recognized by the μPD70325 during DMA demand-release transfer. INT is maskable by the IE bit in the PSW and requires that an external device provide the interrupt vector number. It is designed to allow the interrupt controller to be expanded by the addition of an external interrupt controller such as the μPD71059.

NMI, INTP0-INTP1 are edge-sensitive inputs. By selecting the appropriate bits in the interrupt mode register, these inputs can be programmed to be either rising- or falling-edge triggered. Bits ES0-ES2 correspond to INTP0-INTP2, respectively, as shown in figure 6.

Figure 6. External Interrupt Mode Register (INTM)

0	ES2	0	ES1	0	ES0	0	ESNMI
7			Address xxF40H				0

ES2	INTP2 Input Effective Edge
0	Falling edge
1	Rising edge
ES1	**INTP1 Input Effective Edge**
0	Falling edge
1	Rising edge
ES0	**INTP0 Input Effective Edge**
0	Falling edge
1	Rising edge
ESNMI	**NMI Input Effective Edge**
0	Falling edge
1	Rising edge

The five external interrupts are:

NMI	Nonmaskable interrupt
INT	Cascaded PIC interrupt
INTP0	Interrupt from peripheral 0
INTP1	Interrupt from peripheral 1
INTP2	Interrupt from peripheral 2

The twelve internal interrupts are:

INTTU0	Timer unit 0 interrupt
INTTU1	Timer unit 1 interrupt
INTTU2	Timer unit 2 interrupt
INTD0	DMA channel 0 interrupt
INTD1	DMA channel 1 interrupt
INTSER0	Serial channel 0 error interrupt
INTSR0	Serial channel 0 receive interrupt
INTST0	Serial channel 0 transmit interrupt
INTSER1	Serial channel 1 error interrupt
INTSR1	Serial channel 1 receive interrupt
INTST1	Serial channel 1 transmit interrupt
INTTB	Time base counter interrupt

Table 3 shows the various interrupt request control registers, the options for service, their relative priorities, and the options for multiple control.

Table 3. Interrupt Processing

Interrupt Source	Interrupt Vector	Macro Service	Bank Switching	Priority Setting	Priority Between Groups	Priority Within Group	Multiple Process Control
NMI	2	No	No	Not available	0	—	Not accepted
INT	External	No	No	Not available	7	—	Not accepted
INTTU0	28	Yes	Yes	Available	1	1	Accepted
INTTU1	29					2	
INTTU2	30					3	
INTD0	20	No	Yes	Available	2	1	
INTD1	21					2	
INTP0	24	Yes	Yes	Available	3	1	
INTP1	25					2	
INTP2	26					3	
INTSER0	12	No	Yes	Available	4	1	
INTSR0	13	Yes				2	
INTST0	14	Yes				3	
INTSER1	16	No	Yes	Available	5	1	
INTSR1	17	Yes				2	
INTST1	18	Yes				3	
INTTB	31	No	No	Not available	6	—	

Interrupt Processing Modes

Interrupts, with the exception of NMI, INT, and INTTB have high-performance capability and can be processed in any of three modes: standard vector method (compatible with V20/V30), register bank context switching (supported in hardware), and macroservice (SFR transfers). The processing mode for a given interrupt can be chosen by enabling the appropriate bits in the corresponding interrupt request control register. Each interrupt, except INT and NMI, has its own associated IRC register. The general format for each of these registers is shown in figure 7.

All interrupt processing routines other than those for NMI and INT must end with the execution of the FINT instruction. This instruction informs the interrupt controller that the current interrupt service routine is complete; if FINT is not executed within the service routine, subsequently only interrupts of higher priority will be accepted.

In the vectored service mode, the CPU traps to a vector location.

Figure 7. Interrupt Request Control Registers (IRC)

IF	IMK	MS/INT	ENCS	0	PR_2	PR_1	PR_0
7							0

IF	**Interrupt Flag**
0	No interrupt request generated
1	Interrupt request generated
IMK	**Interrupt Mask**
0	Open (interrupts enabled)
1	Closed (interrupts disabled)
MS/INT	**Interrupt Response Method**
0	Vector interrupt or register bank switching
1	Macroservice function
ENCS	**Register Bank Switching Function**
0	Not used
1	Used
PR_2-PR_0	**Interrupt Group Priority (0-7)**
0 0 0	Highest (0)
	↓
1 1 1	Lowest (7)

Register Bank Switching.

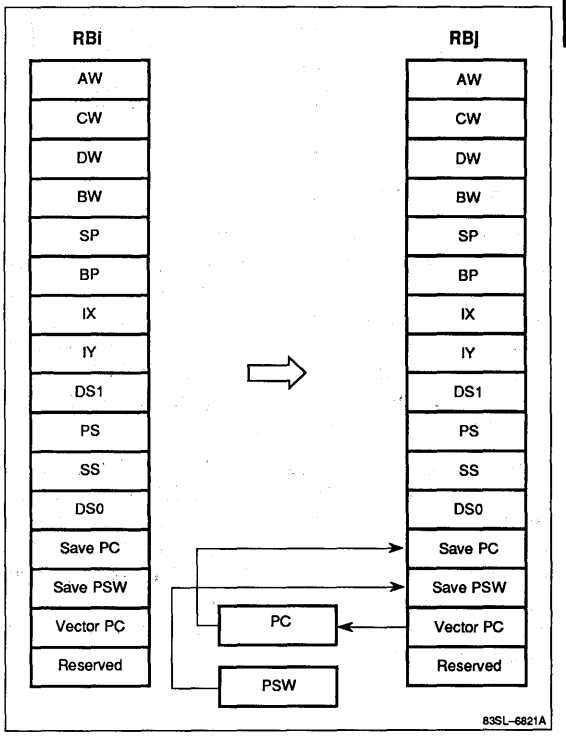
Register bank context switching allows interrupts to be processed rapidly by switching register banks. After an interrupt, the new register bank selected has the same bank number (0-7) as the priority programmed in the associated IRC register. The PC and PSW are automatically sorted in the save areas of the new register bank, and the address of the interrupt routine is loaded from the vector PC storage register in the new register bank.

As in the vectored mode, the IE and BRK bits in the PSW are cleared to zero. After processing, execution of the RETRBI instruction must be executed to return control to the original register bank and restore the former PC and PSW. Figures 8 and 9 show register bank context switching and register bank return.

This method of interrupt service offers a dramatic performance advantage over normal vectored service because there is no need to store and retrieve data/registers on the stack. This also allows hardware-based real-time task switching in high-speed environments.

In addition to context switching, the μPD70325 has a task switch opcode (TSKSW) that allows multiple independent processes to be internally resident. Figure 10 shows the task switching function.

Figure 8. Register Bank Context Switching



4d

Figure 9. Register Bank Return

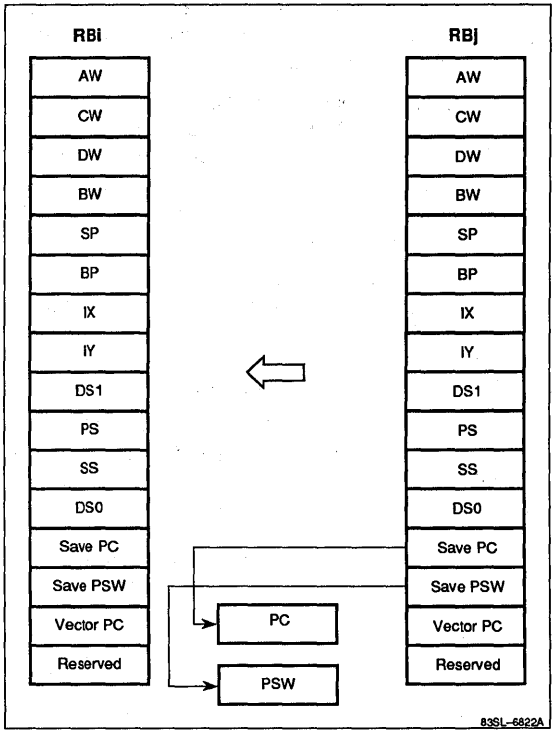
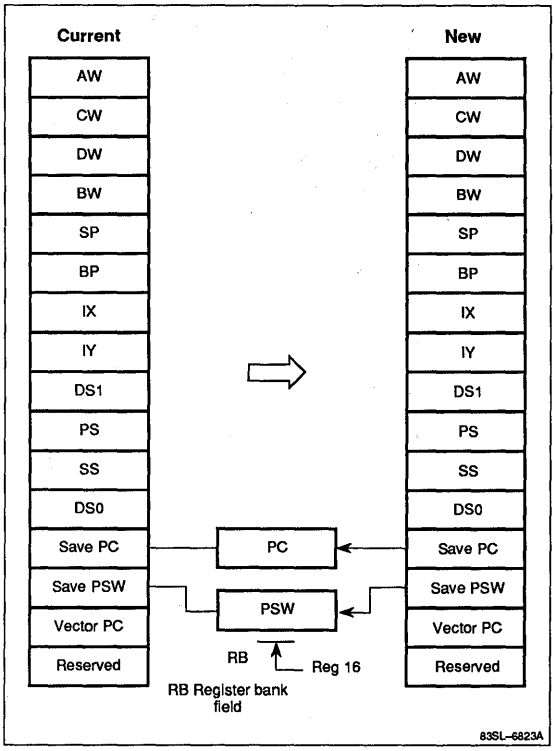


Figure 10. Task Switching



Interrupt Factor Register

The μPD70325 provides an additional register that stores the interrupt vector number of the last-serviced interrupt request. The register is located in special-function memory and is read only in 8-bit operations. This register facilitates the use of one register bank to service multiple interrupt sources, particularly those within the same group (interrupts within the same group will all context switch to the same register bank).

The interrupt vector is stored in the IRQS register as shown in figure 11, and is retained until the next interrupt request is accepted. The value of the IRQS register is not altered by NMI, INT, or macroservice transfers. It is generally recommended that the IRQS register be read before the EI bit is set within the interrupt service routine to assure that its contents will not be altered by multiple processing routines.

Figure 11. Interrupt Factor Register (IRQS)

0	0	0	Interrupt Vector
7		5	4 Address xxFEFH 0

Interrupt Factor	Interrupt Vector
INTTU0	1CH
INTTU1	1DH
INTTU2	1EH
INTD0	14H
INTD1	15H
INTP0	18H
INTP1	19H
INTP2	1AH
INTSER0	0CH
INTSR0	0DH
INTST0	0EH
INTSER1	10H
INTSR1	11H
INTST1	12H
INTTB	1FH

Macroservice Function

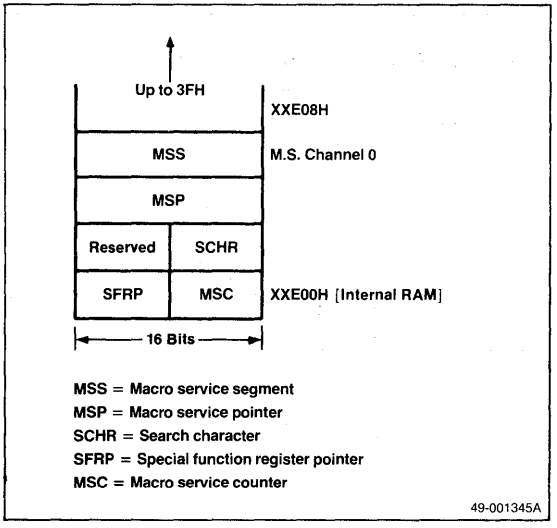
The macroservice function (MSF) is a special microprogram that acts as an internal DMA controller between on-chip peripheral special-function registers and memory. The MSF greatly reduces the software overhead and CPU time that other processors would require for register save processing, register returns, and other handling associated with interrupt processing.

If the MSF is selected for a particular interrupt, each time the request is received, a byte or word of data is transferred between an SFR and memory without interrupting the CPU. Each time a request occurs, the macroservice counter is decremented. When the counter reaches zero, an interrupt to the CPU is generated. The MSF also has a character search option. When selected, every byte transferred is compared to an 8-bit search character and an interrupt is generated if a match occurs or if the macroservice counter reaches zero.

Like the NMI, INT, and INTTB, the two DMA controller interrupts (INTD0 and INTD1) do not have MSF capability.

Eight 8-byte macroservice channels are mapped into internal RAM from XXE00H to XXE3FH. Each macroservice channel contains all necessary information to execute the macroservice process. Figure 12 shows the components of each channel.

Figure 12. Macroservice Channels



Setting the macroservice mode requires programming the corresponding macroservice control register. Each individual interrupt, excluding INT, NMI, serial error, DMA, and TBC, has its own associated register. Figure 13 shows the generic format for all MSC registers.

Figure 13. Macroservice Control Registers (MSC)

MSM_2	MSM_1	MSM_0	DIR	0	CH_2	CH_1	CH_0
7							0

MSM_2-MSM_0	Macroservice Mode
0 0 0	Normal (8-bit transfer)
0 0 1	Normal (16-bit transfer)
1 0 0	Character search (8-bit transfer
	Other combinations are not allowed.
DIR	**Data Transfer Direction**
0	Memory to SFR
1	SFR to memory
CH_2-CH_0	**Macroservice Channel**
0 0 0	Channel 0
	↓
1 1 1	Channel 7

4d

TIMER UNIT

The μPD70325 (figure 14) has two programmable 16-bit interval timers (TM0 and TM1) on chip, each with variable input clock frequencies. Each of the two 16-bit timer registers has an associated 16-bit modulus register (MD0 and MD1). Timer 0 operates in the interval timer mode or one-shot mode; timer 1 has only the interval timer mode.

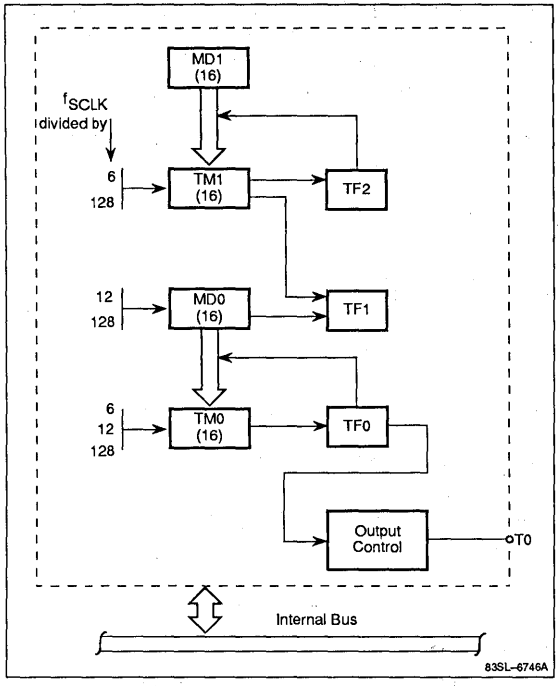
Interval Timer Mode

In this mode, TM0/TM1 are decremented by the selected input clock, and, after counting out, the registers are automatically reloaded from the modulus registers and counting continues. Each time TM1 counts out, interrupts are generated through TF1 and TF2 (timer flags 1 and 2). When TM0 counts out, an interrupt is generated through TF0. The timer-out signal can be used as a square-wave output whose half-cycle is equal to the count time.

Two input clocks derived from the system clock are SCLK/6 and SCLK/128. Typical timer values shown below are based on f_{OSC} = 10 MHz and f_{SCLK} = $f_{OSC}/2$.

Clock	Timer Resolution	Full Count
SCLK/6	1.2 μs	78.643 μs
SCLK/128	25.6 μs	1.678 s

Figure 14. Timer Unit Block Diagram



One-Shot Mode

In the one-shot mode, TM0 and MD0 operate as independent one-shot timers. Starting with a preset value, each is decremented to zero. At zero, counting ceases and an interrupt is generated by TF0 (from TM0) or TF1 (from MD0).

When TM0 is programmed to one-shot mode, TM1 may still operate in interval mode.

Two input clocks derived from the system clock are SCLK/12 and SCLK/128. Typical timer values shown below are based on f_{OSC} = 10 MHz and $f_{SCLK} = f_{OSC}/2$.

Clock	Timer Resolution	Full Count
SCLK/12	2.4 μs	157.283 ms
SCLK/128	25.6 μs	1.678 s

Timer Control Registers

Setting the desired timer mode requires programming the timer control register. See figures 15 and 16 for the TMC register format.

Figure 15. Timer Control Register 0 (TMC0)

TS0	TCLK0	MS0	MCLK0	ENT0	ALV	MOD_1	MOD_0
7			Address xxF90H				0

TS0	TM0 In Either Mode
0	Stop countdown
1	Start countdown

MOD_1	MOD_0	TCLK0	TM0 Register Clock Frequency
0	0	0	$f_{SCLK}/6$ (Interval)
0	0	1	$f_{SCLK}/128$ (Interval)
0	1	0	$f_{SCLK}/12$ (One-shot)
0	1	1	$f_{SCLK}/128$ (One-shot)

MS0	MD0 Register Countdown (One-Shot Mode)
0	Stop
1	Start

MCLK0	MD0 Register Clock Frequency
0	$f_{SCLK}/12$
1	$f_{SCLK}/128$

ENT0	TOUT Square-Wave Output
0	Disable
1	Enable

ALV	TOUT Initial Level When TOUT Disabled by ENT0 = 0
0	Low
1	High

MOD_1	MOD_0	Timer Unit Mode
0	0	Interval timer
0	1	One-shot
1	X	Reserved

Figure 16. Timer Control Register 1 (TMC1)

TS1	TCLK1	0	0	0	0	0	0
7			Address xxF91H				0

TS1	Timer 1 Countdown
0	Stop
1	Start

TCLK1	Timer 1 Clock Frequency
0	$f_{SCLK}/6$
1	$f_{SCLK}/128$

TIME BASE COUNTER

The 20-bit free-running time base counter (TBC) controls internal timing sequences and is available as the source of periodic interrupts at lengthy intervals. One of the four interrupt periods can be selected by programming the TB0 and TB1 bits in the processor control register (PRC). The TBC interrupt is unlike the others because it is fixed as a level 7 vectored interrupt. Macroservice and register bank switching cannot be used to service this interrupt. See figures 17 and 18.

Figure 17. Time Base Interrupt Request Control Register (TBIC)

TBF	TBMK	0	0	0	1	1	1
7			Address xxFECH				0

TBF	Time Base Interrupt Flag
0	No interrupt generated
1	Interrupt generated

TBMK	Time Base Interrupt Mask
0	Unmasked
1	Masked

Figure 18. Processor Control Register (PRC)

0	RAMEN	0	0	TB_1	TB_0	PCK_1	PCK_0
7			Address xxFEBH				0

RAMEN	Built-in RAM
0	Disable
1	Enable

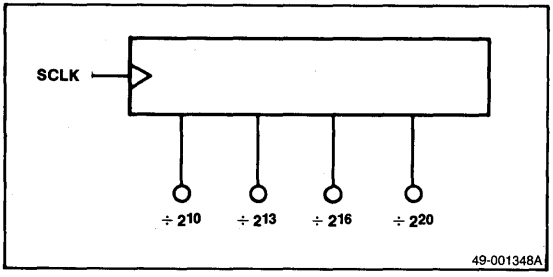
TB_1	TB_0	Time Base Interrupt Period
0	0	$2^{10}/f_{CLK}$
0	1	$2^{13}/f_{CLK}$
1	0	$2^{16}/f_{CLK}$
1	1	$2^{20}/f_{CLK}$

PCK_1	PCK_0	System Clock Frequency (f_{CLK})
0	0	$f_X/2$
0	1	$f_X/4$
1	0	$f_X/8$
1	1	Reserved

The RAMEN bit in the PRC register allows the internal RAM to be removed from the memory address space to implement faster instruction execution.

The TBC (figure 19) uses the system clock as the input frequency. The system clock can be changed by programming the PCK0 and PCK1 bits in the processor control register (PRC). Reset initializes the system clock to $f_{OSC}/8$ (f_{OSC} = external oscillator frequency).

Figure 19. Time Base Counter (TBC) Block Diagram



REFRESH CONTROLLER

The μPD70325 has an on-chip refresh controller for dynamic and pseudostatic RAM memory. The refresh controller generates refresh cycles between the normal CPU bus cycles according to the refresh specifications programmed.

The refresh controller outputs a 9-bit refresh address on address bits A_8-A_0 during the refresh bus cycle. Address bits A_{19}-A_9 are fixed to 0s during this cycle. The 9-bit refresh address is automatically incremented at every refresh cycle for 512 row addresses. The 8-bit refresh mode (RFM) register (figure 20) specifies the refresh operation and allows refresh during both CPU HALT and HOLD modes. Refresh cycles are automatically timed to $\overline{REFRQ}$ following read/write cycles to minimize the effect on system throughput.

As shown in figure 20, the $\overline{REFRQ}$ output level is determined by the by the RFLV and RFEN bits of the RFM register.

4d

Figure 20. Refresh Mode Register (RFM)

RFLV	HLDRF	HLTRF	RFEN	RFW_1	RFW_0	RFT_1	RFT_0
7			Address xxFE1H				0

RFLV	RFEN	$\overline{\text{REFRQ}}$ Output Signal Level
0	0	0
1	0	1
0	1	0
1	1	Refresh pulse

HLDRF	Automatic Refresh Cycle in HOLD Mode
0	Disabled
1	Enabled

HLTRF	Automatic Refresh Cycle in HALT Mode
0	Disabled
1	Enabled

RFEN	Automatic Refresh Cycle
0	Refresh pin = RFLV
1	Refresh enabled

RFW_1	RFW_0	No. of Wait States Inserted in Refresh Cycle
0	0	0
0	1	1
1	0	2
1	1	2

RFT_1	RFT_0	Refresh Period
0	0	16/SCLK
0	1	32/SCLK
1	0	64/SCLK
1	1	128/SCLK

SERIAL INTERFACE

The μPD70325 has two full-duplex UARTs, channels 0 and 1. Each channel has a transmit line (TxDn), a receive line (RxDn), and a clear-to-send (CTSn) handshaking line. Communication is synchronized by a start bit, and either even, odd, or no parity may be programmed. Character length may be programmed to either 7 or 8 bits, and either 1 or 2 stop bits may be selected.

Each serial channel of the μPD70325 has a dedicated baud rate generator, so there is no need to obligate any of the on-chip timers to handle this function. The baud rate generators allow individual transfer rates for each channel and support rates up to 1.25 Mb/s. All standard baud rates are available and are not restricted by the value of the particular external crystal.

Each baud rate generator has an 8-bit data register (BRGn) that functions as a prescaler to a programmable input clock selected by the serial communication control register (SCCn). Together these must be set to generate a frequency equivalent to the desired baud rate.

The baud rate generator can be programmed to obtain the desired transmission rate according to the following formula:

$$B \times G = \frac{SCLK \times 10^6}{2^{n+1}}$$

where:

B = baud rate
G = baud rate generator register (BRGn) value
n = input clock specification; the value loaded into the SCCn register ($0 < n < 8$)
SCLK = system clock frequency (MHz)

Based on the above formula, the following table shows the baud rate generator values used to obtain standard transmission rates when SCLK = 5 MHz.

Baud Rate	n	BRGn	Error (%)
110	7	178	0.25
150	7	130	0.16
300	6	130	0.16
600	5	130	0.16
1200	4	130	0.16
2400	3	130	0.16
4800	2	130	0.16
9600	1	130	0.16
19.2k	0	130	0.16
38.4k	0	65	0.16
1.25M	0	2	0.00

In addition to the asynchronous mode, channel 0 has a synchronous I/O interface mode. In this mode, each bit of data transferred is synchronized to a serial clock (SCLK0). The receive clock may be specified as either the internal baud rate generator output or an external signal provided on the $\overline{\text{CTS0}}$ input pin (the RSCK bit of the SCM0 register must be programmed to 0).

This mode is functionally equivalent to using the serial channel as a shift register because data is synchronous with the SCLK signal. Data bits from consecutive bytes may directly follow one another since no extra bits (parity, start, or stop) are added. This mode is compatible with the μCOM75 and μCOM87 series, and allows direct interfacing to these devices.

Figure 21 details the serial communication mode register, which controls the operational mode and data format of the serial channel. The serial communication control register shown in figure 22 specifies the baud rate generator input clock frequency.

Figure 21. Serial Communication Mode Registers (SCM)

TxRDY	RxB	PRTY1	PRTY0	CLTSK	SLRSCK	MD1	MD0
7							0

TxRDY	Transmitter Control
0	Disabled
1	Enabled
RxB	**Receiver Control**
0	Disabled
1	Enabled
PRTY1-PRTY0	**Parity Control**
0 0	No parity
0 1	0 parity (Note 1)
1 0	Odd parity
1 1	Even parity
CLTSK	**Character Length (Async Mode)** **Tx Shift Clock (I/O Interface Mode)**
0	7 bits (Async) No effect (I/O intfc)
1	8 bits (Async) Trigger transmit (I/O intfc)
SLRSCK	**Stop Bits (Async Mode)** **Receiver Clock (I/O Interface Mode)**
0	1 stop bit (Async) Ext clock input on CTS0 (I/O intfc)
1	2 stop bits (Async) Int clock output on CTS1 (I/O intfc)
MD1-MD0	**Mode**
0 0	I/O interface (Note 2)
0 1	Asynchronous
1 x	Reserved

Notes:

(1) Parity is 0 during transmit and ignored during receive.

(2) Channel only.

The serial communication error registers of the V25 are replaced in the μPD70325 with the serial status registers shown in figure 23. These registers provide error flags and buffer status information. The error bits are automatically cleared when the next data byte is received; otherwise, these flags are persistent.

Figure 22. Serial Communication Control Register (SCC)

0	0	0	0	PRS_3	PRS_2	PRS_1	PRS_0
7							0

PRS_3-PRS_0	Baud Rate Generator Input Clock Frequency
0000	$f_{SCLK}/2$ (n = 0)
0001	$f_{SCLK}/4$
0010	$f_{SCLK}/8$
0011	$f_{SCLK}/16$
0100	$f_{SCLK}/32$
0101	$f_{SCLK}/64$
0110	$f_{SCLK}/128$
0111	$f_{SCLK}/256$
1000	$f_{SCLK}/512$ (n = 8)

The TxBE and RxBF bits signal the status of the respective transmit and receive buffers. These bits are reset automatically when either the baud rate generator or serial control register contents are written. The AS (All Sent) bit is set when both the transmit buffer and the transmit shift register are empty.

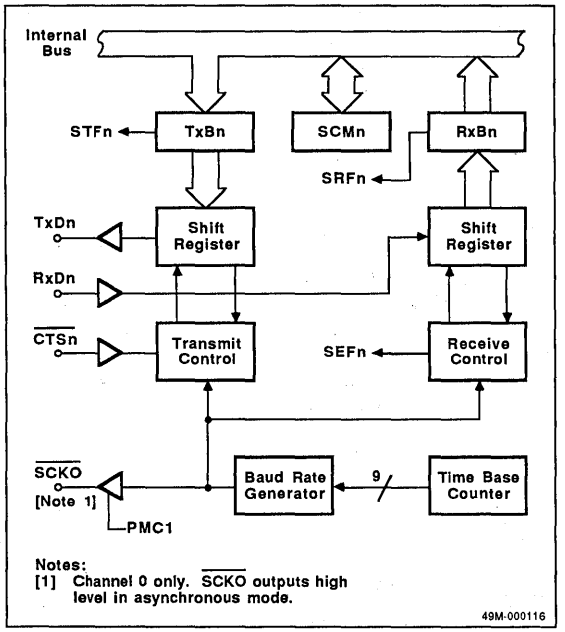
Figure 23. Serial Status Register (SST)

RxDn	AS	TxBE	RxBF	0	ERP	ERF	ERO
7							0

RxDn	Receive Terminal State
0, 1	Status of RxD pin
AS	**All Sent Flag**
0	Data has been written in transmit buffer
1	Data in transmit buffer and shift register has been sent
TxBE	**Transmit Buffer Empty Flag**
0	Data has been written in transmit buffer
1	Data in buffer has been sent to shift register
RxBF	**Receive Buffer Full Flag**
0	Data has been read from receive buffer
1	Data has been sent from shift register to buffer
ERP	**Parity Error Flag**
0	No error
1	Transmit and receive parity are different
ERF	**Framing Error Flag**
0	No error
1	Stop bit not detected
ERO	**Overrun Error Flag**
0	No error
1	Data is received before receive buffer outputs previous data

Figures 24 and 25 show the serial interface block diagram for asynchronous and I/O interface modes, respectively.

Figure 24. Serial Channels 0 and 1; Asynchronous Mode



DMA CONTROLLER

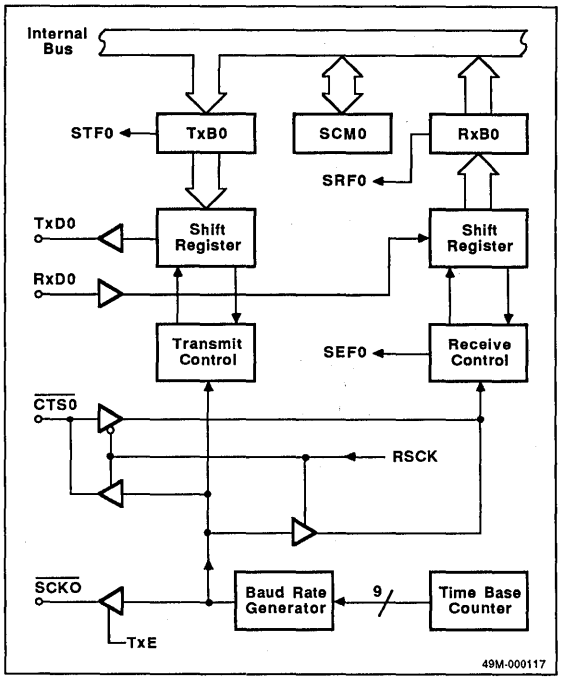
The μPD70325 has an on-chip, two-channel DMA controller capable of supporting data transfer at full bus bandwidth. Although the operating modes of the μPD70325 DMA unit are identical to those of the V25, the programming and data transfer rates are different.

Six I/O pins support the operation of the DMA unit.

- Two active-high request inputs
- Two active-low acknowledge outputs valid only in I/O-to-memory and memory-to-I/O transfer modes
- Two terminal count active-low outputs, driven low when the transfer count register is decremented from 0 (borrow occurs).

Two memory-to-memory transfer modes (single-step and burst) are supported as well as two I/O-to-memory modes (single transfer and demand release). Refer to table 4.

Figure 25. Serial Channel 0; I/O Interface Mode



Memory-to-Memory DMA Transfers

Single-Step Mode. The single-step mode allows direct memory access for memory-to-memory transfers. These transfers take two bus cycles (nominally two clock cycles each), one to read the data and the other to write it back to memory. The data is stored inside the μPD70325 in a dedicated temporary register between the two bus cycles.

When a DMA request occurs in this mode, DMA and CPU bus cycles alternate until the transfer count is reached. This mode is provided to allow the user program to continue executing at 50% of its normal speed when DMA is operational. The single request thus generates a full block of data transfer; that is, until terminal count is reached.

Burst Mode. This mode is also a memory-to-memory transfer running at full bus bandwidth. Once a request is recognized, the DMA unit takes control of the bus and continuously transfers data until the terminal count is reached for that channel. This mode forces all other lower priority bus masters (including the CPU) to hold their bus requests until the specified DMA block is transferred.

Table 4. DMA Unit Functional Interaction

	Single-Step Mode	Burst Mode	Single-Transfer Mode	Demand Release Mode
Transmission coverage	Memory - memory	Memory - memory	Memory - I/O	Memory - I/O
Function	Under one time of DMA request instruction, one bus cycle and one DMA transmission are alternately executed the specified number of times.	Under one time of DMA request, specified times of DMA transmission are executed.	One DMA transmission is executed every time DMA request occurs.	DMA transmission is executed while DMARQ terminal is kept high-level.
DMA start	Rise of DMARQ Setting TDMA bit of DMA control register	Rise of DMARQ Setting TDMA bit of DMA control register	Rise of DMARQ	High level of DMARQ
Halt method	Depends on software	None	Depends on software	Halted at low level during DMA transmission
Interrupt	All accepted	Not accepted during DMA transmission	All accepted	All accepted except during DMA transmission
During halt	Specified times of DMA transmission are executed consecutively	Specified times of DMA transmission are executed consecutively	Same as usual	Same as usual
DMA request during DMA transmission	DMA at channel 1 retained while DMA at channel 0 is executed	Other DMA is retained until DMA transmission is terminated.	DMA transmission under request is executed after one DMA transmission is over	DMA at channel 1 is retained while DMA at channel 0 is executed.

4d

I/O-to-Memory DMA Transfers

I/O-to-memory (source synchronized) and memory-to-I/O (destination synchronized) transfers are performed in one bus cycle (nominally two clock states). These transfers drive a single memory address and strobe the IORD or IOWR and $\overline{DMAAK}$ signals. Thus the appropriate I/O device and memory location are selected without the processor driving the data bus. These transfers are designed for high-speed, I/O data transfer reaching 5 megabytes/ second at 10 MHz.

Single Transfer. Single-transfer mode responds with one DMA transfer per rising edge of the DMA request line. After one DMA cycle is performed, control is transferred back to the CPU until another request is recognized. Transfers continue, one-per-request, until the terminal count is reached.

Care must be taken in the single-transfer mode to adhere to the t_{WIQH} and t_{WIQL} specifications. Although the DMA request is latched internally, if the above values are not met, the μPD70325 may not detect the following rising edge of the request input, and the subsequent DMA transfer will not take place.

Demand Release. Demand release mode continues to perform DMA transfers until either terminal count is reached or the DMA request is removed. When the DMARQ is removed during DMA operation, the DMA channel finishes the current transfer and releases the bus.

Note that the t_{WIQH} and t_{WIQL} parameters must also be taken into design consideration for the demand release mode. The t_{WIQH} parameter insures that the DMA request is held past the internal sampling point. When the $\overline{DMAAK}$ signal is asserted, the μPD70325 has obviously accepted the request. Thus, the t_{WIQH} parameter may not need to be fully satisfied; that is, the request may be removed after the $\overline{DMAAK}$ signal has been asserted. This should not present a problem because it is specified that the request must be active until the acknowledge is output.

Demand Release Termination. The demand release mode may be released in one of two ways: the programmed terminal count is reached; or the DMARQ signal is removed. These two conditions differ considerably in operational characteristics.

When terminating demand release DMA by the terminal count method, there are no restrictions placed on the transfer bandwidth, and the transfer rate will maintain 4 megabytes/second at 8 MHz or 5 megabytes/second at 10 MHz. This rate is achieved by programming the DMA to operate at zero wait states and by sustaining the DMARQ until the terminal count signal is asserted.

If demand release mode DMA is terminated by deassertion of the DMARQ line, then several specifications are at issue. The $\overline{DMAAK}$ signal is triggered from the falling clock edge in the middle of B1 with a propagation delay of 80 ns max. The DMA request line is sampled on the next rising edge of the CPU clock, which is ideally 62.5 ns later (minus the 10-ns DMARQ setup time). Therefore, on a worst-case device, the $\overline{DMAAK}$ signal is not asserted until after the μPD70325 has already sampled the DMARQ line. If $\overline{DMAAK}$ is used to negate DMARQ, the request may not be sampled correctly. This situation may be avoided by inserting wait states in the DMA transfer to delay the DMARQ sampling point.

A second issue is the internal propagation delay of the $\overline{DMAAK}$ signal. If the DMARQ input is gated with the $\overline{DMAAK}$ signal in a zero wait-state system, $\overline{DMAAK}$ is not held low long enough to allow internal timing signals to fully propagate through the device.

The result of the above situation must be broken into two cases: zero wait-state systems and two wait-state systems. The zero wait-state system using DMARQ to terminate the transfer may "overrun." This would produce one more DMA transfer than the desired number (unless terminal count was reached on the transfer during which DMARQ was removed).

The two wait-state system will function normally in the above situation since the wait states assure that the DMARQ line will be sampled and accepted, thereby terminating the transfer without any slippage. In this case, the t_{WIQL} specification may be taken at 3T (min), the width of the $\overline{DMAAK}$ pulse.

Addresses. All DMA addresses are mapped to external memory space on the μPD70325. When a location corresponding to an internal data area is addressed, external memory with the same address will be accessed.

Priority. Bus hold and refresh control maintain higher bus priority than the DMA unit and are active during DMA transmission. If an HLDRQ or $\overline{REFRQ}$ signal is presented during DMA operation, the DMA unit releases its $\overline{DMAAK}$ signal and relinquishes control of the bus to the requesting bus master. DMA transmission is also temporarily halted during an interrupt acknowledge cycle.

HALT State. The μPD70325 will acknowledge a DMA request while the CPU is in the HALT state, and the processor will return to HALT after the transfer is completed. When the DMA terminal count interrupt is presented in the HALT state, the processor releases the HALT state and returns to normal mode.

Latency. The μPD70325 DMA latency time is substantially faster than the standard V25. Since the DMA controller is hardwired, it responds to requests within instruction execution. The resulting worst-case latency is 14+2W clocks from request to acknowledge and this typical value will be 4 system clocks.

DMA Registers

The μPD70325 DMA registers differ from those on the standard V25. The control blocks for each channel are located in the special function register area and thus do not overlap the internal RAM register banks.

DMA mode registers are provided for each DMA channel. These registers, shown in figure 26, specify the transmission mode, byte/word data size, and the enable state of each DMA channel.

Figure 26. DMA Mode Registers (DMAM)

MD_2	MD_1	MD_0	W	EDMA	TDMA	0	0
7							0

MD_2-MD_0	Transfer Mode
0 0 0	Single-step (memory to memory)
0 0 1	Demand release (I/O to memory)
0 1 0	Demand release (memory to I/O)
0 1 1	Reserved
1 0 0	Burst (memory to memory)
1 0 1	Single-transfer (I/O to memory)
1 1 0	Single-transfer (memory to I/O)
1 1 1	Reserved

W	Transfer Method
0	Byte transfer
1	Word transfer

EDMA	TDMA	Transfer Condition
0	0	Disabled
1	0	Maintain condition
1	1	Start DMA transfer

The μPD70325 performs two consecutive 8-bit transfers to accomplish word data transfers with the lower addressed byte transferred first. Upon reaching terminal count, the DMA channel is automatically disabled by the EDMA bit of the mode register. A DMA request to a disabled channel is ignored. The DMA transfer bit (TDMA) has meaning only in the single-step and burst modes and allows software to initiate the DMA operation. This software request is valid only when the channel is enabled. The TDMA bit is write only and is always read as a 0.

The DMA address control registers (figure 27) specify the method of address pointer update. Two bits specify the source address and two specify the destination address update mode. The μPD70325 adjusts the address pointers by 1 for byte transfers and by 2 for word transfers.

Figure 27. DMA Address Control Registers (DMAC)

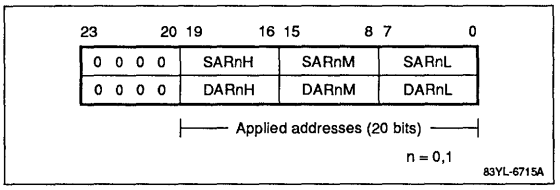
0	0	PD_1	PD_0	0	0	PS_1	PS_0
7							0

PD_1-PD_0	Destination Address Offset
0 0	No modification
0 1	Increment
1 0	Decrement
1 1	No modification
PS_1-PS_0	**Source Address Offset**
0 0	No modification
0 1	Increment
1 0	Decrement
1 1	No modification

Figure 28 shows the address pointer registers that specify the source and destination of the DMA transfer. Unlike the standard V25, the address registers are linear. This allows the 20-bit address to be completely specified in three byte-wide registers. The SARnL and DARnL registers contain the low 8 bits of the address; SARnM and DARnM contain the middle 8 bits; and SARnH and DARnH contain the high 4 bits of the address in the low nibble (the high nibble is set to 0).

All of these registers may be read or written with either 8- or 16-bit transfers and are updated as specified in the DMAC registers.

Figure 28. DMA Address Registers



The terminal count registers (TCnH and TCnL) are dual 8-bit registers that hold the current number of transfers remaining in the DMA block. These registers are read/write in 8- or 16-bit operations. They must be initially programmed to the desired number of transfers minus one. This is because the terminal count interrupt is generated by a borrow out of these registers; this borrow is generated by the decrement performed after each DMA transfer.

DMA Timing

DMA operation on the μPD70325 is considerably faster than on the standard V25. This speedup is realized by converting the DMA operation from a microcoded process to a hardwired one. As a result, the DMA latency times on the standard V25 do not occur on the μPD70325. However, bus controller latency is still present as is the nominal transfer time: 1 bus cycle for I/O-to-memory transfers and 2 bus cycles for memory-to-memory transfers.

Programmable wait state control is active for DMA operations with the programmed number of states added to both source and destination addresses even if these numbers are different. Memory-to-I/O transfers insert the number of wait states required by either the I/O device or the memory location (whichever is slower). I/O-to-memory transfers insert the memory wait states for the memory write cycle.

Figures 29 to 32 are examples of cycles for DMA operations. Figure 33 is a block diagram of the μPD70325 DMA controller and its internal registers.

Figure 29. DMA Single-Step Mode Timing

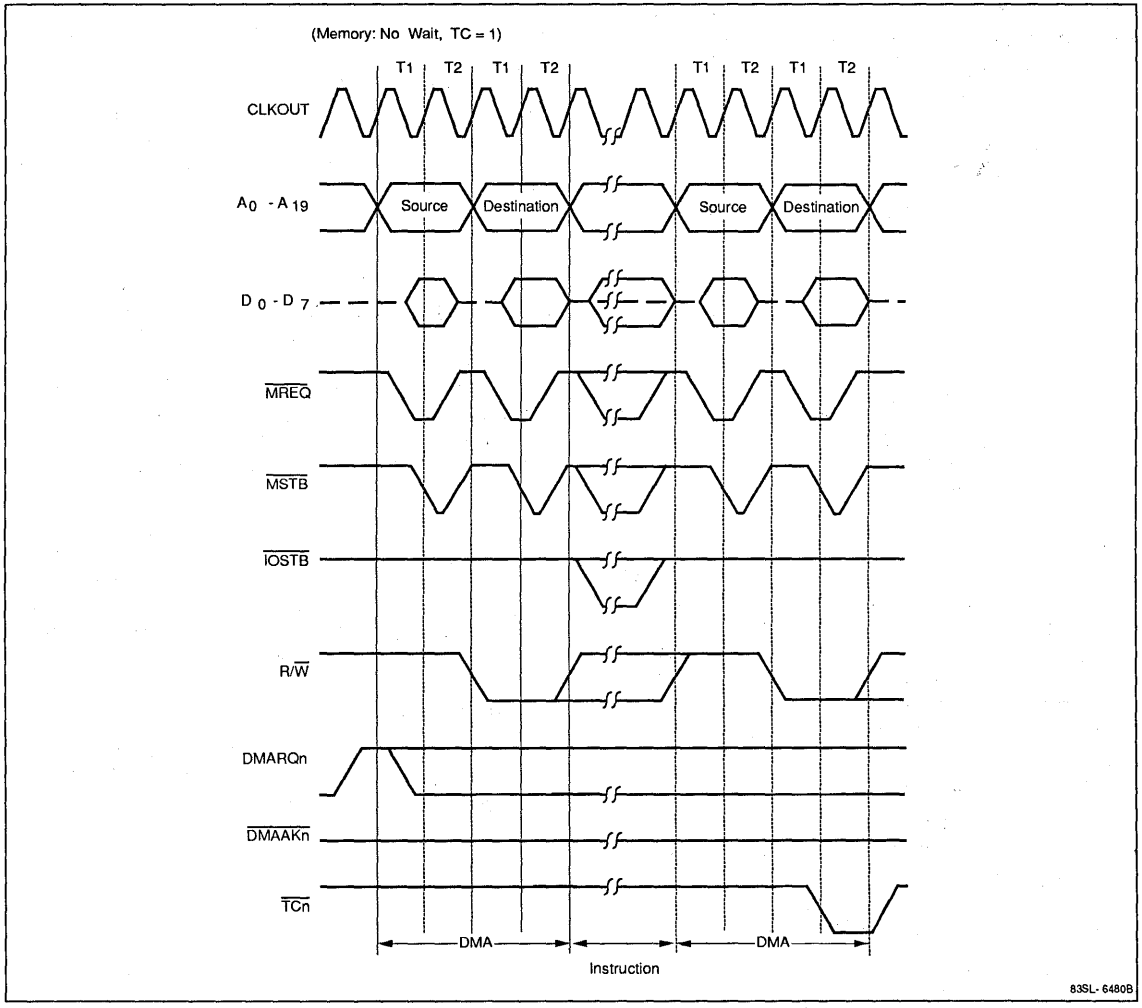
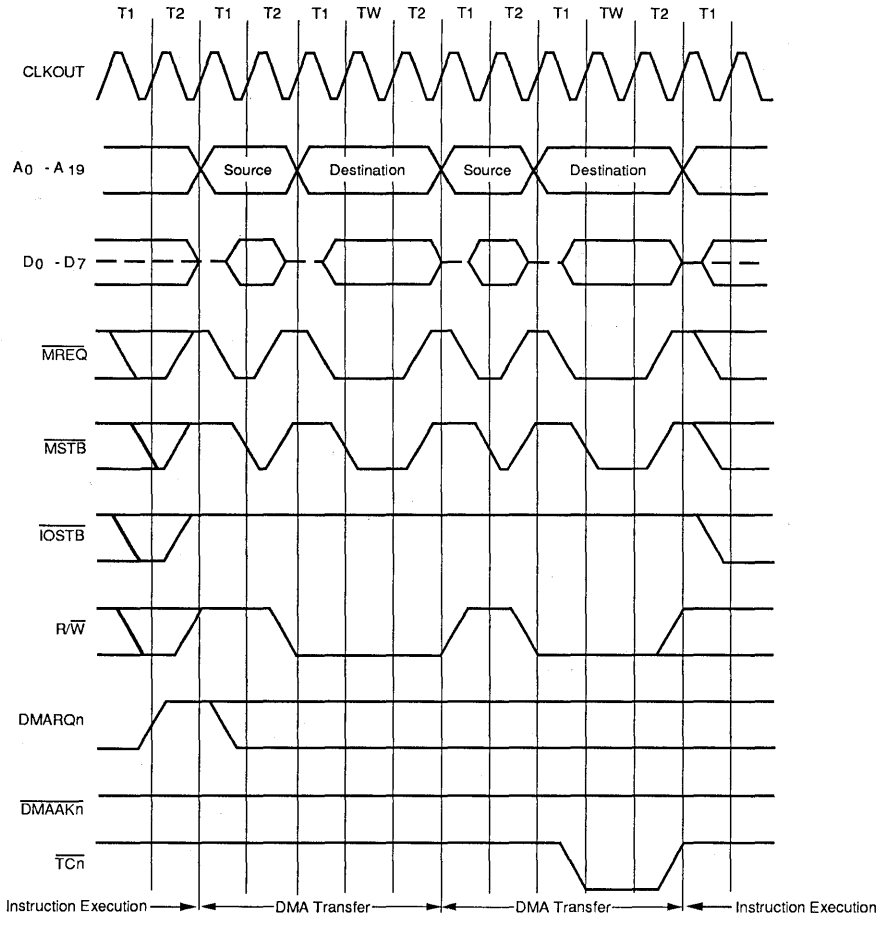


Figure 30. DMA Burst Mode Timing



Burst mode with no-wait for transfer source memory bank and insertion of one wait state for transfer destination when DMA is started by DMARQ signal when TC=1.

83SL-6478B

4d

Figure 31. DMA Single-Transfer Mode Timing

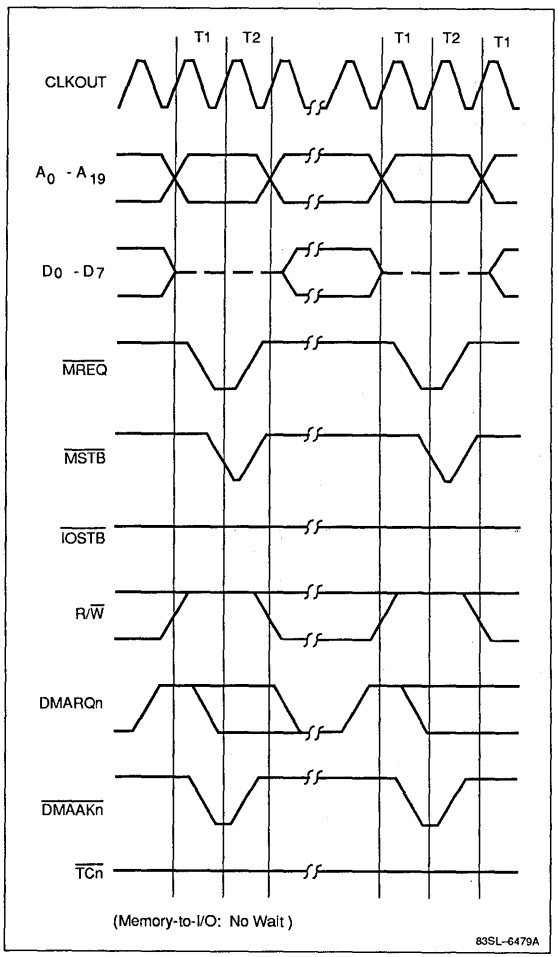


Figure 32. DMA Demand Release Timing

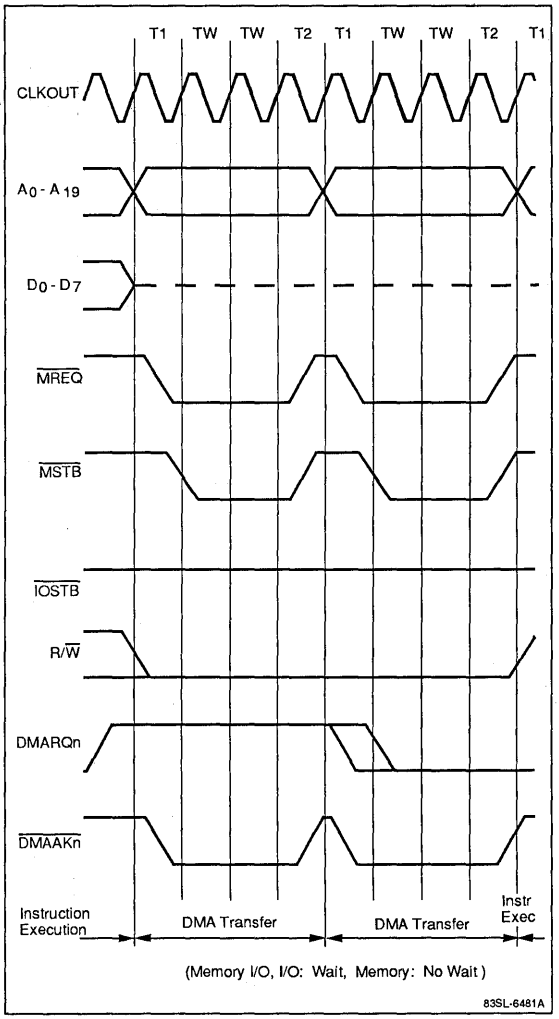
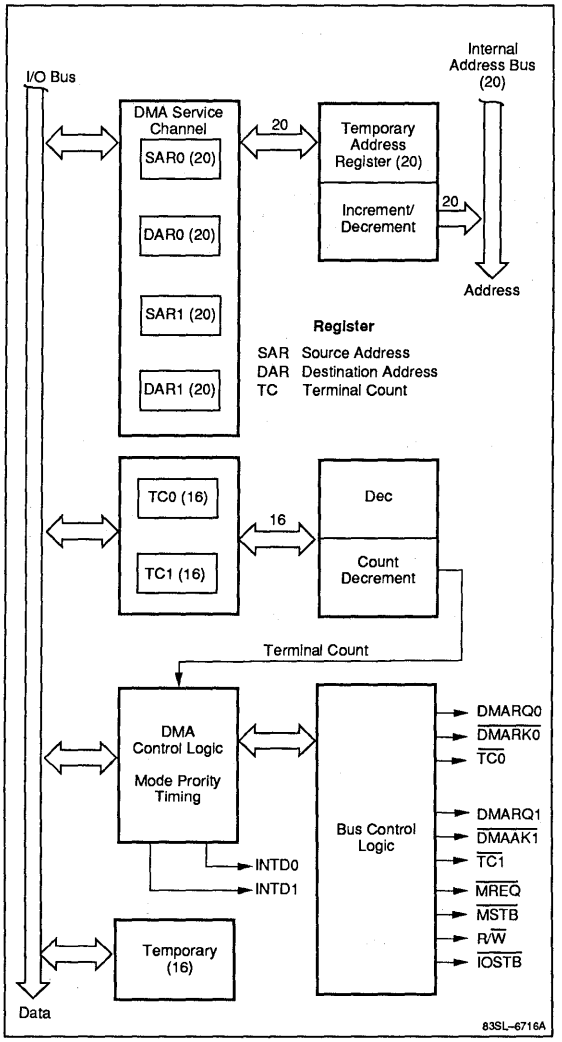


Figure 33. DMA Unit Block Diagram



PARALLEL I/O PORTS

The μPD70325 has three 8-bit parallel I/O ports: P0, P1, and P2. Associated registers are shown in figures 34, 35, and 36. Special-function register (SFR) locations can access these ports as memory. The port lines are individually programmable as inputs or outputs. Many of the port lines have dual functions as port or control lines.

Use the associated port mode control (PMC) and port mode (PM) registers to select the function for a given I/O line.

Figure 34. Port 0 Registers (PMC0, PM0)

$PMC0_7$	0	0	0	0	0	0	0
7			PMC0 Register				0

$PM0_7$	$PM0_6$	$PM0_5$	$PM0_4$	$PM0_3$	$PM0_2$	$PM0_1$	$PM0_0$
7			PM0 Register				0

		$PMC0_n = 0$	
Port Pin	**$PMC0_7 = 1$**	**$PM0_n = 1$**	**$PM0_n = 0$**
$P0_7$	CLKOUT	Input port	Output port
$P0_6$	—	Input port	Output port
$P0_5$	—	Input port	Output port
$P0_4$	—	Input port	Output port
$P0_3$	—	Input port	Output port
$P0_2$	—	Input port	Output port
$P0_1$	—	Input port	Output port
$P0_0$	—	Input port	Output port

Figure 35. Port 1 Registers (PMC1, PM1)

$PMC1_7$	$PMC1_6$	$PMC1_5$	$PMC1_4$	$PMC1_3$	0	0	0
7			PMC1 Register				0

$PM1_7$	$PM1_6$	$PM1_5$	$PM1_4$	1	1	1	1
7			PM1 Register				0

		$PMC1_n = 0$	
Port Pin	**$PMC1_7 = 1$**	**$PM1_n = 1$**	**$PM1_n = 0$**
$P1_7$	READY input	Input port	Output port
$P1_6$	$\overline{SCKO}$ output	Input port	Output port
$P1_5$	TOUT output	Input port	Output port
$P1_4$	INT input	$\overline{POLL}$ input	Output port
$P1_3$	$\overline{INTAK}$ output	INTP2 input	—
$P1_2$	—	INTP1 input	—
$P1_1$	—	INTP0 input	—
$P1_0$	—	NMI input	—

4d

Figure 36. Port 2 Registers (PMC2, PM2)

$PMC2_7$	$PMC2_6$	$PMC2_5$	$PMC2_4$	$PMC2_3$	$PMC2_2$	$PMC2_1$	$PMC2_0$
7			PMC2 Register				0

$PM2_7$	$PM2_6$	$PM2_5$	$PM2_4$	$PM2_3$	$PM2_3$	$PM2_2$	$PM2_1$	$PM2_0$
7			PM2 Register					0

		$PMC2_n = 0$	
Port Pin	$PMC2_n = 1$	$PM2_n = 1$	$PM2_n = 0$
$P2_7$	HLDRQ Input	Input port	Output port
$P2_6$	$\overline{HLDAK}$ output	Input port	Output port
$P2_5$	$\overline{TC1}$ output	Input port	Output port
$P2_4$	$\overline{DMAAK1}$ output	Input port	Output port
$P2_3$	DMARQ1 Input	Input port	Output port
$P2_2$	$\overline{TC0}$ output	Input port	Output port
$P2_1$	$\overline{DMAAK0}$ output	Input port	Output port
$P2_0$	DMARQ0 Input	Input port	Output port

The analog comparator port (PT) compares each input line to a reference voltage. The reference voltage can be programmed to the V_{TH} input x n/16, where n = 1 to 16. See figure 37.

Figure 37. Port T Mode Register (PMT)

0	0	0	0	PMT_3	PMT_2	PMT_1	PMT_0
7							0

Comparator Reference Voltage (V_{REF})	PMT_3	PMT_2	PMT_1	PMT_0
V_{TH} x 16/16	0	0	0	0
1/16	0	0	0	1
2/16	0	0	1	0
3/16	0	0	1	1
4/16	0	1	0	0
5/16	0	1	0	1
6/16	0	1	1	0
7/16	0	1	1	1
8/16	1	0	0	0
9/16	1	0	0	1
10/16	1	0	1	0
11/16	1	0	1	1
12/16	1	1	0	0
13/16	1	1	0	1
14/16	1	1	1	0
15/16	1	1	1	1

PROGRAMMABLE WAIT STATES

You can generate wait states internally to further reduce the necessity for external hardware. Insertion of these wait states allows direct interface to devices whose access times cannot meet the CPU read/write timing requirements.

When using this function, the entire 1 megabyte of memory address space is divided into 128K blocks. Each block can be programmed for zero, one, or two wait states, or two plus those added by the external READY signal. The top two blocks are programmed together as one unit.

The appropriate bits in the wait control word (WTC) control wait-state generation. Programming the upper two bits in the wait control word sets the wait-state conditions for the entire I/O address space. Figure 38 shows the memory map for programmable wait-state generation.

Figure 39 diagrams the wait control word. Note that READY pin control is enabled only when two internally generated wait states are selected by the "11" option.

Figure 38. Programmable Wait State Generation

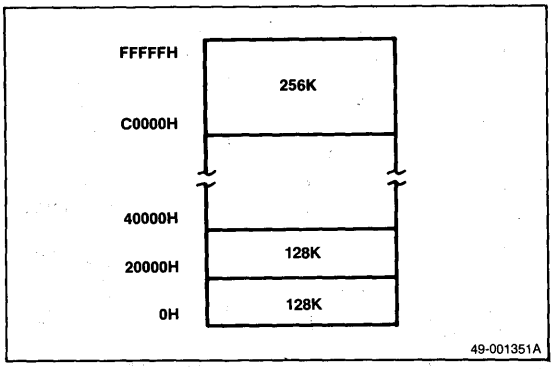


Figure 39. Wait Control Word (WTC)

IO1	IO0	Block 61	Block 60	Block 51	Block 50	Block 41	Block 40
7			Wait Control, High				0

Block 31	Block 30	Block 21	Block 20	Block 11	Block 10	Block 01	Block 00
7			Wait Control, Low				0

Wait States	Block n1	Block n0
0	0	0
1	0	1
2	1	0
2 or more (control from READYpin)	1	1

n = 0 thru 6

STANDBY MODES

The two low-power standby modes are HALT and STOP. Both modes are entered under software control.

HALT Mode

In HALT mode, the CPU is inactive and thus the chip consumes much less power than when fully operational. The external oscillator remains functional and all internal peripherals are active. Internal status and output port line conditions are maintained. Any unmasked interrupt can release this mode. In the EI state, interrupts are processed subsequently in vector mode. In the DI state, program execution is restarted with the instruction following the HALT instruction.

STOP Mode

The STOP mode allows the largest power reduction while maintaining internal RAM. The oscillator is stopped, halting the CPU as well as all internal peripherals. Internal status is maintained. Only a reset or NMI can release this mode.

A standby flag in the SFR area is reset by rises in the supply voltage. Its status is maintained during normal operation and standby. The STBC register (figure 40) is not initialized by $\overline{\text{RESET}}$. Use the standby flag to determine whether program execution is returning from standby or from a cold start by setting this flag before entering STOP mode.

Figure 40. Standby Register (STBC)

0	0	0	0	0	0	0	SBF
7			Address xxFE0H				0

SBF	Standby Flag
0	No changes in V_{DD} (standby)
1	Rising edge on V_{DD} (cold start)

SPECIAL-FUNCTION REGISTERS

Table 5 lists the special-function registers . The 8 high-order bits of each register address (denoted by. xx in table 5) is specified by the IDB register. This allows the special function register bank to be dynamically relocated in memory.

SFR addresses not listed in table 5 are reserved. If read, the contents of these addresses are undefined and any write operation is meaningless.

The Read/Write column in table 5 shows the legal data movement operations used to address these registers. Data size can be specified as: bit (1), byte (8), and/or word (16).

Table 5. Special-Function Registers

Address	Register Name	Label	Read/Write	Reset Condition
xxF00H	Port 0 data	P0	R/W 8/1	Undefined
xxF01H	Port 0 mode	PM0	W 8	0FFH
xxF02H	Port 0 control	PMC0	W 8	00H
xxF08H	Port 1 data	P1	R/W 8/1	Undefined
xxF09H	Port 1 mode	PM1	W 8	0FFH
xxF0AH	Port 1 control	PMC1	W 8	00H
xxF10H	Port 2 data	P2	R/W 8/1	Undefined
xxF11H	Port 2 mode	PM2	W 8	0FFH
xxF12H	Port 2 control	PMC2	W 8	00H
xxF38H	Threshold port data	PT	R 8	Undefined
xxF3BH	Threshold port mode	PMT	R/W 8/1	00H
xxF40H	External interrupt mode	INTM	R/W 8/1	00H
xxF44H	External interrupt macroservice control channel 0 (Note 4)	EMS0	R/W 8/1	Undefined
xxF45H	External interrupt macroservice control channel 1 (Note 4)	EMS1	R/W 8/1	Undefined
xxF46H	External interrupt macroservice control channel 2 (Note 4)	EMS2	R/W 8/1	Undefined
xxF4CH	External interrupt request control 0 (Note 4)	EXIC0	R/W 8/1	47H
xxF4DH	External interrupt request control 1 (Note 4)	EXIC1	R/W 8/1	47H
xxF4EH	External interrupt request control 2 (Note 4)	EXIC2	R/W 8/1	47H
xxF60H	Serial receive buffer channel 0	RxB0	R 8	Undefined
xxF62H	Serial transmit buffer channel 0	TxB0	W 8	Undefined
xxF65H	Serial receive macroservice register 0 (Note 4)	SRMS0	R/W 8/1	Undefined
xxF66H	Serial transmit macroservice register 0 (Note 4)	STMS0	R/W 8/1	Undefined
xxF68H	Serial mode register 0	SCM0	R/W 8/1	00H
xxF69H	Serial control register 0	SCC0	R/W 8/1	00H
xxF6AH	Baud rate generator 0	BRG0	R/W 8/1	00H
xxF6BH	Serial status register 0	SCS0	R 8	60H
xxF6CH	Serial error interrupt request register 0 (Note 4)	SEIC0	R/W 8/1	47H
xxF6DH	Serial receive interrupt request register 0 (Note 4)	SRIC0	R/W 8/1	47H
xF6EH	Serial transmit interrupt request register 0 (Note 4)	STIC0	R/W 8/1	47H
xxF70H	Serial receive buffer 1	RxB1	R 8	Undefined
xxF72H	Serial transmit buffer 1	TxB1	W 8	Undefined
xxF75H	Serial receive macroservice register 1 (Note 4)	SRMS1	R/W 8/1	Undefined
xxF76H	Serial transmit macroservice register 1 (Note 4)	STMS1	R/W 8/1	Undefined
xxF78H	Serial mode register 1	SCM1	R/W 8/1	00H
xxF79H	Serial control register 1	SCC1	R/W 8/1	00H
xxF7AH	Baud rate generator channel 1	BRG1	R/W 8/1	00H
xxF7BH	Serial status register 1	SCS1	R 8	60H
xxF7CH	Serial error interrupt request register 1 (Note 4)	SEIC1	R/W 8/1	47H

Table 5. Special-Function Registers (cont)

Address	Register Name	Label	Read/Write	Reset Condition
xxF7DH	Serial receive interrupt request register 1 (Note 4)	SRIC1	R/W 8/1	47H
xxF7EH	Serial transmit interrupt request register 1 (Note 4)	STIC1	R/W 8/1	47H
xxF80H	Timer register 0 (Note 5)	TM0	R/W 16	Undefined
xxF82H	Timer 0 modulo register (Note 5)	MD0	R/W 16	Undefined
xxF88H	Timer register 1 (Note 5)	TM1	R/W 16	Undefined
xxF8AH	Timer 1 modulo register (Note 5)	MD1	R/W 16	Undefined
xxF90H	Timer 0 control register (Note 5)	TMC0	R/W 8/1	00H
xxF91H	Timer 1 control register (Note 5)	TMC1	R/W 8/1	00H
xxF94H	Timer unit 0 macroservice register (Note 4)	TMMS0	R/W 8/1	Undefined
xxF95H	Timer unit 1 macroservice register (Note 4)	TMMS1	R/W 8/1	Undefined
xxF96H	Timer unit 2 macroservice register (Note 4)	TMMS2	R/W 8/1	Undefined
xxF9CH	Timer unit 0 interrupt request register (Note 4)	TMIC0	R/W 8/1	47H
xxF9DH	Timer unit 1 interrupt request register (Note 4)	TMIC1	R/W 8/1	47H
xxF9EH	Timer unit 2 interrupt request register (Note 4)	TMIC2	R/W 8/1	47H
xxFA0H	DMA address update control register 0	DMAC0	R/W 8/1	Undefined
xxFA1H	DMA mode register 0	DMAM0	R/W 8/1	47H
xxFA2H	DMA address update control register 1	DMAC1	R/W 8/1	Undefined
xxFA3H	DMA mode register 1	DMAM1	R/W 8/1	00H
xxFACH	DMA interrupt request control register 0 (Note 4)	DIC0	R/W 8/1	47H
xxFADH	DMA interrupt request control register 1 (Note 4)	DIC1	R/W 8/1	47H
xxFC0H	DMA channel 0 source address pointer low	SAR0L	R/W 16/8	Undefined
xxFC1H	DMA channel 0 source address pointer mid	SAR0M	R/W 16/8	Undefined
xxFC2H	DMA channel 0 source address pointer high	SAR0H	R/W 8	Undefined
xxFC4H	DMA channel 0 destination address pointer low	DAR0L	R/W 16/8	Undefined
xxFC5H	DMA channel 0 destination address pointer mid	DAR0M	R/W 16/8	Undefined
xxFC6H	DMA channel 0 destination address pointer high	DAR0H	R/W 8	Undefined
xxFC8H	DMA channel 0 count register	TC0	R/W 16/8	Undefined
xxFD0H	DMA channel 1 source address pointer low	SAR1L	R/W 16/8	Undefined
xxFD1H	DMA channel 1 source address pointer mid	SAR1M	R/W 16/8	Undefined
xxFD2H	DMA channel 1 source address pointer high	SAR1H	R/W 8	Undefined
xxFD4H	DMA channel 1 destination address pointer low	DAR1L	R/W 16/8	Undefined
xxFD5H	DMA channel 1 destination address pointer mid	DAR1M	R/W 16/8	Undefined
xxFD6H	DMA channel 1 destination address pointer high	DAR1H	R/W 8	Undefined
xxFD8H	DMA channel 1 terminal count register	TC1	R/W 16/8	Undefined
xxFE0H	Standby control register (Notes 1, 2)	STBC	R/W 8/1	Undefined
xxFE1H	Refresh mode register	RFM	R/W 8/1	0FCH
xxFE8H	Wait state control register	WTC	R/W 16/8	0FFFFH
xxFEAH	User flag register (Note 3)	FLAG	R/W 8/1	00H

Table 5. Special-Function Registers (cont)

Address	Register Name	Label	Read/Write	Reset Condition
xxFEBH	Processor control register	PRC	R/W 8/1	4EH
xxFECH	Time base interrupt request control register (Note 4)	TBIC	R/W 8/1	47H
xxFEFH	Interrupt factor register (Note 4)	IRQS	R 8	Undefined
xxFFCH	Interrupt priority control register (Note 4)	ISPR	R 8	00H
xxFFFH	Internal data area base address register (Note 4)	IDB	R/W 8/1	0FFH

Notes:

(1) Standby control register can be set by instruction but not cleared.

(2) At power-on reset: 00H. Otherwise not changed.

(3) Bit operations on FLAG register bits other than 3 and 5 have no meaning.

(4) One wait state is inserted in accesses to these registers.

(5) A maximum of six wait states are added into accesses to these registers.

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = 25°C

Supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD}+0.5 ≤ +7.0 V
Output voltage, V_O	−0.5 to V_{DD}+0.5 ≤ +7.0 V
Threshold voltage, V_{TH}	−0.5 to V_{DD}+0.5 ≤ +7.0 V
Output current low, I_{OL}	Each output pin 4.0 mA (Total 50 mA)
Output current high, I_{OH}	Each output pin −2.0 mA (Total −20 mA)
Operating temperature range, T_{OPT}	−10 to +70°C
Storage temperature range, T_{STG}	−65 to +150°C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Comparator Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ± 10%

Parameter	Symbol	Min	Max	Unit
Accuracy	VA_{COMP}	–	±100	mV
Threshold voltage	V_{TH}	0	V_{DD}+0.1	V
Comparison time	t_{COMP}	64	65	t_{CYK}
PT input voltage	V_{IPT}	0	V_{DD}	V

Capacitance

T_A = 25°C; V_{DD} = 0 V

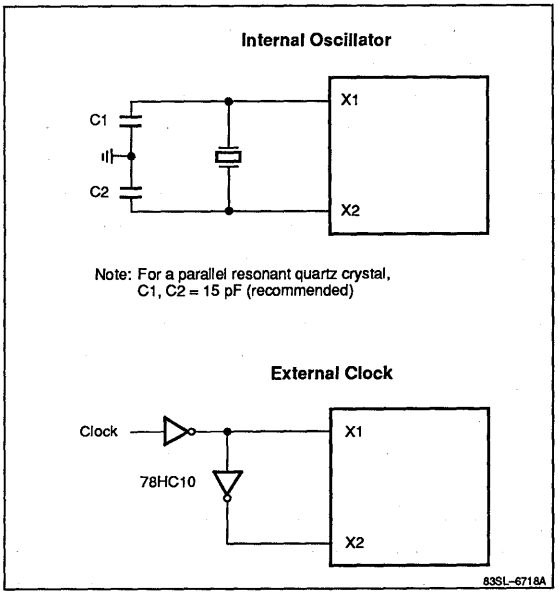
Parameter	Symbol	Max	Unit	Conditions
Input capacitance	C_I	10	pF	fc = 1 MHz; unmeasured pins returned to ground
Output capacitance	C_O	20	pF	
I/O capacitance	C_{IO}	20	pF	

DC Characteristics; μPD70325-8

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2		V_{DD}	V	All except $\overline{RESET}$, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 V_{DD}		V_{DD}	V	$\overline{RESET}$, $P1_0$/NMI, X1, X2
Output voltage, low	V_{OL}			0.45	V	I_{OL} = 1.6 mA
Output voltage, high	V_{OH}	V_{DD} − 1.0			V	I_{OH} = −0.4 mA
Input current	I_{IN}			±20	μA	$\overline{EA}$, $P1_0$/NMI; 0 ≤ V_{IN} ≤ V_{DD}
Input leakage current	I_{LI}			±10	μA	All except $\overline{EA}$, $P1_0$/NMI; 0 ≤ V_{IN} ≤ V_{DD}
Output leakage current	I_{LO}			±10	μA	0 ≤ V_O ≤ V_{DD}
V_{TH} supply current	I_{TH}		0.5	1.0	mA	0 ≤ V_{TH} ≤ V_{DD}
V_{DD} supply current	I_{DD1}		65	120	mA	Operation mode
	I_{DD2}		25	50	mA	HALT mode
	I_{DD3}		10	30	uA	STOP mode

External System Clock Control Source



Recommended Oscillator Components

Ceramic Resonator (Note 1)		Capacitors	
Manufacturer	**Product No.**	**C1 (pF)**	**C2 (pF)**
Kyocera	KBR-10.0M	33	33
Murata Mfg.	CSA.10.0MT	47	47
	CSA16.0MX040	30	30
TDK	FCR10.M2S	30	30
	FCR16.0M2S	15	6

Crystal (Note 2)		Capacitors	
Manufacturer	**Product No.**	**C1 (pF)**	C2 (pF)
Kinseki	HC-49/U	15	15
	HC-43/U	15	15

Notes:

(1) Ceramic resonator product no. includes the frequency: 10.0 or 16.0 MHz.

(2) Crystal frequencies: 10, 16, 20 MHz.

4d

AC Characteristics; μPD70325-8

T_A = −10 to +70°; V_{DD} = +5.0 V ±10%; C_L = 100 pF max; T = t_{CYK}; n = number of wait states inserted

Parameter	Symbol	Min	Max	Unit	Conditions
Input rise, fall times	t_{IR}, t_{IF}		20	ns	Except X1, X2, $\overline{RESET}$, NMI
Input rise, fall times (Schmitt)	t_{IRS}, t_{IFS}		30	ns	$\overline{RESET}$, NMI
Output rise, fall times	t_{OR}, t_{OF}		20	ns	Except CLKOUT
X_1 cycle time	t_{CYX}	62	250	ns	
X_1 width, low	t_{WXL}	20		ns	
X_1 width, high	t_{WXH}	20		ns	
X_1 rise, fall times	t_{XR}, t_{XF}		20	ns	
CLKOUT cycle time	t_{CYK}	125	2000	ns	CLKOUT = $f_X/2$
CLKOUT width, low	t_{WKL}	0.5T − 15		ns	
CLKOUT width, high	t_{WKH}	0.5T − 15		ns	
CLKOUT rise, fall times	t_{KR}, t_{KF}		15	ns	
Address delay time	t_{DKA}	15	90	ns	
Address valid to input data valid	t_{DADR}		(n+1.5)T − 70	ns	
$\overline{MREQ}$ to data delay time	t_{DMRD}		(n+1)T − 60	ns	
$\overline{MSTB}$ to data delay time	t_{DMSD}		(n+0.5)T − 60	ns	
$\overline{MREQ}$ to $\overline{TC}$ delay time	t_{DMRTC}		0.5T + 50	ns	
$\overline{MREQ}$ to $\overline{MSTB}$ delay time	t_{DMRMS}	0.5T − 35	0.5T + 35	ns	
$\overline{MREQ}$ width, low	t_{WMRL}	(n+1)T − 30		ns	
Address hold time	t_{HKA}	0.5T − 30		ns	
Input data hold time	t_{HKDR}	0		ns	
Next control setup time	t_{SCC}	T − 25		ns	
$\overline{TC}$ width, low	t_{WTCL}	(n+2)T − 30		ns	
Address data output	t_{DADW}		0.5T + 50	ns	
$\overline{MREQ}$ delay time	t_{DAMR}	0.5T − 30		ns	
$\overline{MSTB}$ delay time	t_{DAMS}	T − 30		ns	
$\overline{MSTB}$ width, low	t_{WMSL}	(n+0.5)T − 30		ns	
Data output setup time	t_{SDM}	(n+1)T − 50		ns	
Data output hold time	t_{HMDW}	0.5T − 30		ns	
$\overline{IOSTB}$ delay time	t_{DAIS}	0.5T − 30		ns	
$\overline{IOSTB}$ to data input	t_{DISD}		(n+1)T − 60	ns	
$\overline{IOSTB}$ width, low	t_{WISL}	(n+1)T − 30		ns	
Address hold time	t_{HISA}	0.5T − 30		ns	
Data input hold time	t_{HISDR}	0		ns	
Output data setup time	t_{SDIS}	(n+1)T − 50		ns	
Output data hold time	t_{HISDW}	0.5T − 30		ns	
Next DMARQ setup time	t_{SDADQ}		T − 50	ns	Demand mode
DMARQ hold time	t_{HDARQ}	0		ns	Demand mode
$\overline{DMAAK}$ read width, low	t_{WDMRL}	(n+1.5)T − 30		ns	
$\overline{DMAAK}$ to $\overline{TC}$ delay time	t_{DDATC}		0.5T + 50	ns	
$\overline{DMAAK}$ write width, low	t_{WDMWL}	(n+1)T − 30		ns	

AC Characteristics; μPD70325-8 (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{\text{REFRQ}}$ delay time	t_{DARF}	0.5T – 30		ns	
$\overline{\text{REFRQ}}$ width, low	t_{WRFL}	(n+1)T – 30		ns	
Address hold time	t_{HRFA}	0.5T – 30		ns	
$\overline{\text{RESET}}$ width, low	t_{WRSL1}	30		ms	Crystal osc; STOP/Power-On RESET
$\overline{\text{RESET}}$ width, low	t_{WRSL2}	5		μs	System reset
$\overline{\text{MREQ}}$, $\overline{\text{IOSTB}}$ to READY setup time	t_{SCRY}		(n – 1)T – 100	ns	n ≥ 2
$\overline{\text{MREQ}}$, $\overline{\text{IOSTB}}$ to READY hold time	t_{HCRY}	(n – 1)T		ns	n ≥ 2
$\overline{\text{HLDAK}}$ output delay time	t_{DKHA}	15	80	ns	
BUS control float to $\overline{\text{HLDAK}}$ ↓	t_{CFHA}	T – 50		ns	
$\overline{\text{HLDAK}}$ ↑ to control output time	t_{DHAC}	T – 50		ns	
HLDRQ ↓ to control output time	t_{DHQC}	3T + 30		ns	
$\overline{\text{HLDAK}}$ width, low	t_{WHAL}	T		ns	
HLDRQ setup time	t_{SHQK}	30		ns	
HLDRQ to $\overline{\text{HLDAK}}$ delay time	t_{DHQHA}		3T +160	ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
INTP, DMARQ setup time	t_{SIQK}	30		ns	
INTP, DMARQ width, high	t_{WIQH}	8T		ns	
INTP, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{\text{POLL}}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{\text{CTS}}$ width, low	t_{WCTL}	2T		ns	
INT setup time	t_{SIRK}	30		ns	
INT hold time	t_{HIAIQ}	0		ns	
$\overline{\text{INTAK}}$ width, low	t_{WIAL}	2T – 30		ns	
$\overline{\text{INTAK}}$ delay time	t_{DKIA}	15	80	ns	
$\overline{\text{INTAK}}$ width, high	t_{WIAH}	T – 30		ns	
$\overline{\text{INTAK}}$ to data delay time	t_{DIAD}		2T – 130	ns	
$\overline{\text{INTAK}}$ to data hold time	t_{HIAD}	0	0.5T	ns	
$\overline{\text{SCKO}}$ cycle time	t_{CYTK}	1000		ns	
$\overline{\text{SCKO}}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{\text{SCKO}}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	
TxD hold time	t_{HTKD}	20		ns	
$\overline{\text{CTS0}}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{\text{CTS0}}$ (RSCK) width, high	t_{WSRH}	420		ns	
$\overline{\text{CTS0}}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

4d

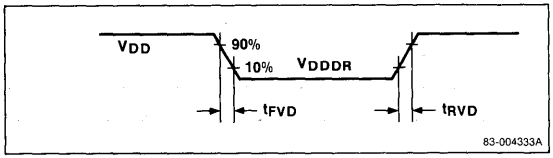
STOP Mode Data Retention Characteristics

T_A = −10 to +70°C

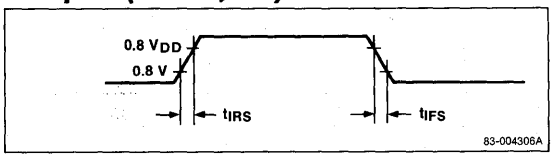
Parameter	Symbol	Min	Max	Unit
Data retention voltage	V_{DDDR}	2.4	5.5	V
V_{DD} rise time	t_{RVD}	200		μs
V_{DD} fall time	t_{FVD}	200		μs

Timing Waveforms

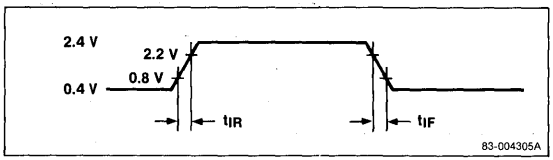
Stop Mode Data Retention Timing



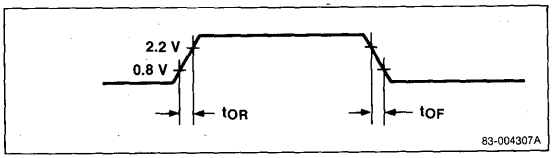
AC Input 2 ($\overline{RESET}$, NMI)



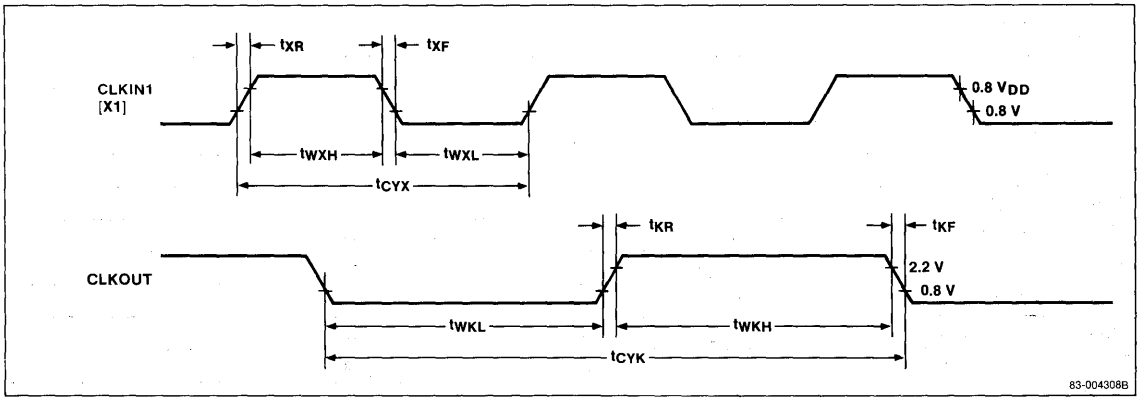
AC Input 1 (Except X1, X2, $\overline{RESET}$, NMI)



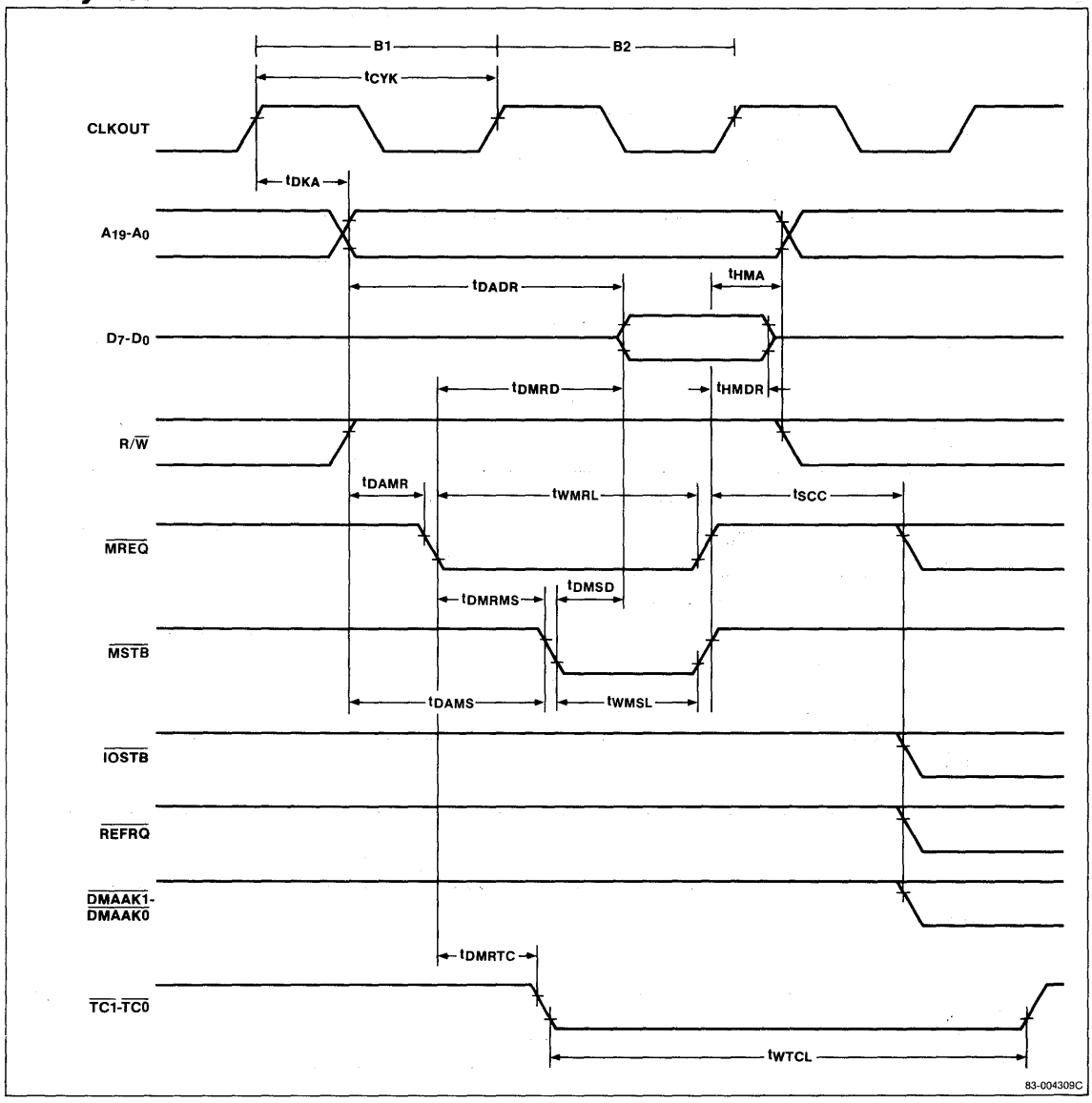
AC Output (Except CLKOUT)



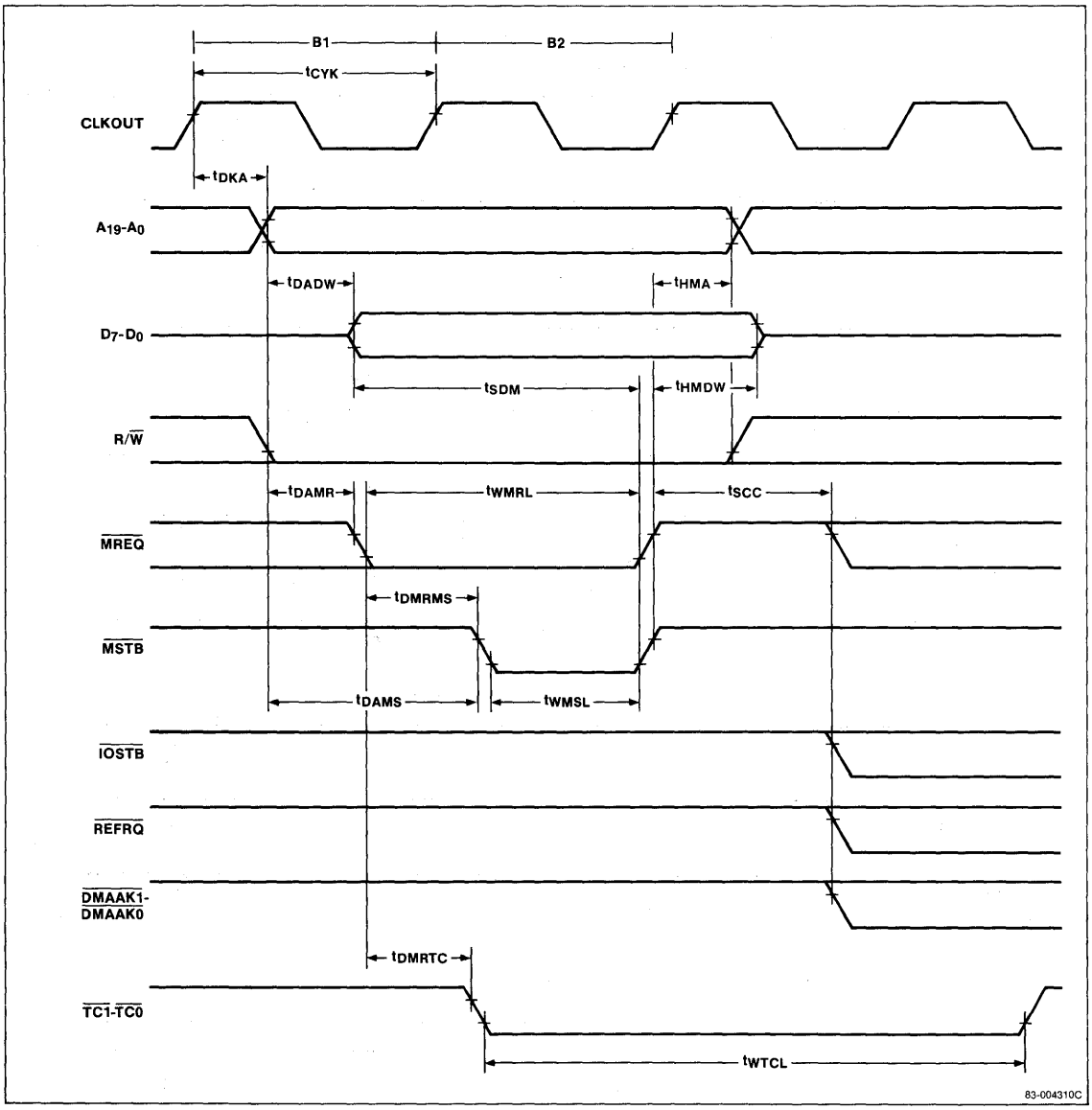
Clock In and Clock Out



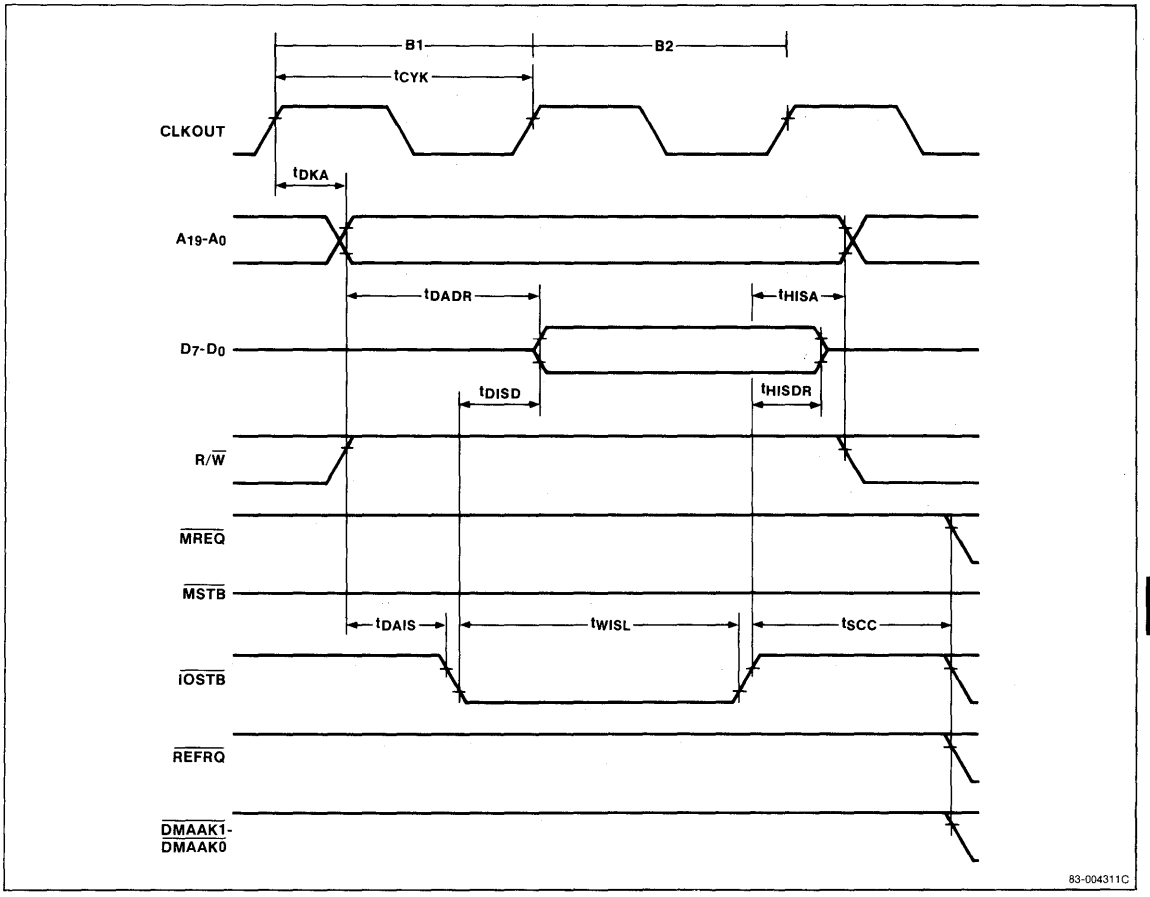
Memory Read


B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADR
tHMA
D7-D0
tDMRD
tHMDR
R/W
tDAMR
tWMRL
tSCC
MREQ
tDMRMS
tDMSD
MSTB
tDAMS
tWMSL
IOSTB
REFRQ
DMAAK1-
DMAAK0
tDMRTC
TC1-TC0
tWTCL
83-004309C

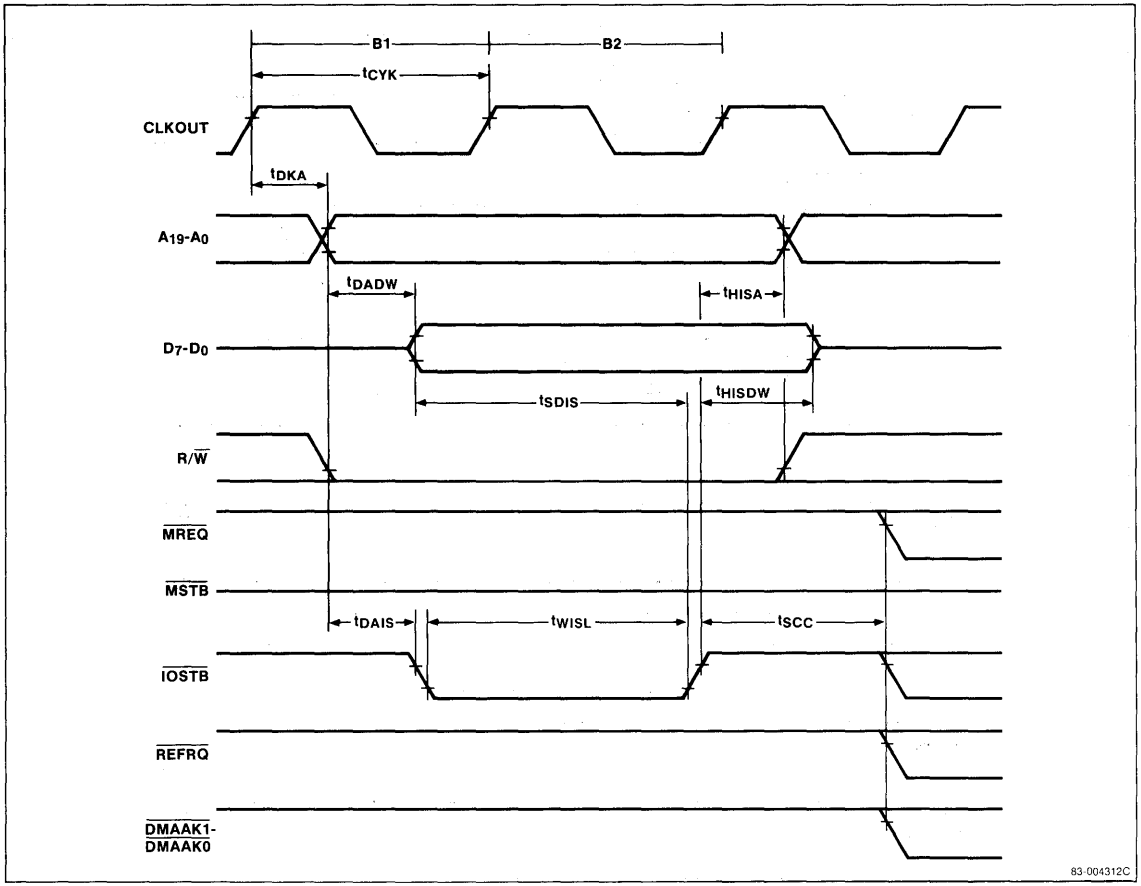
Memory Write



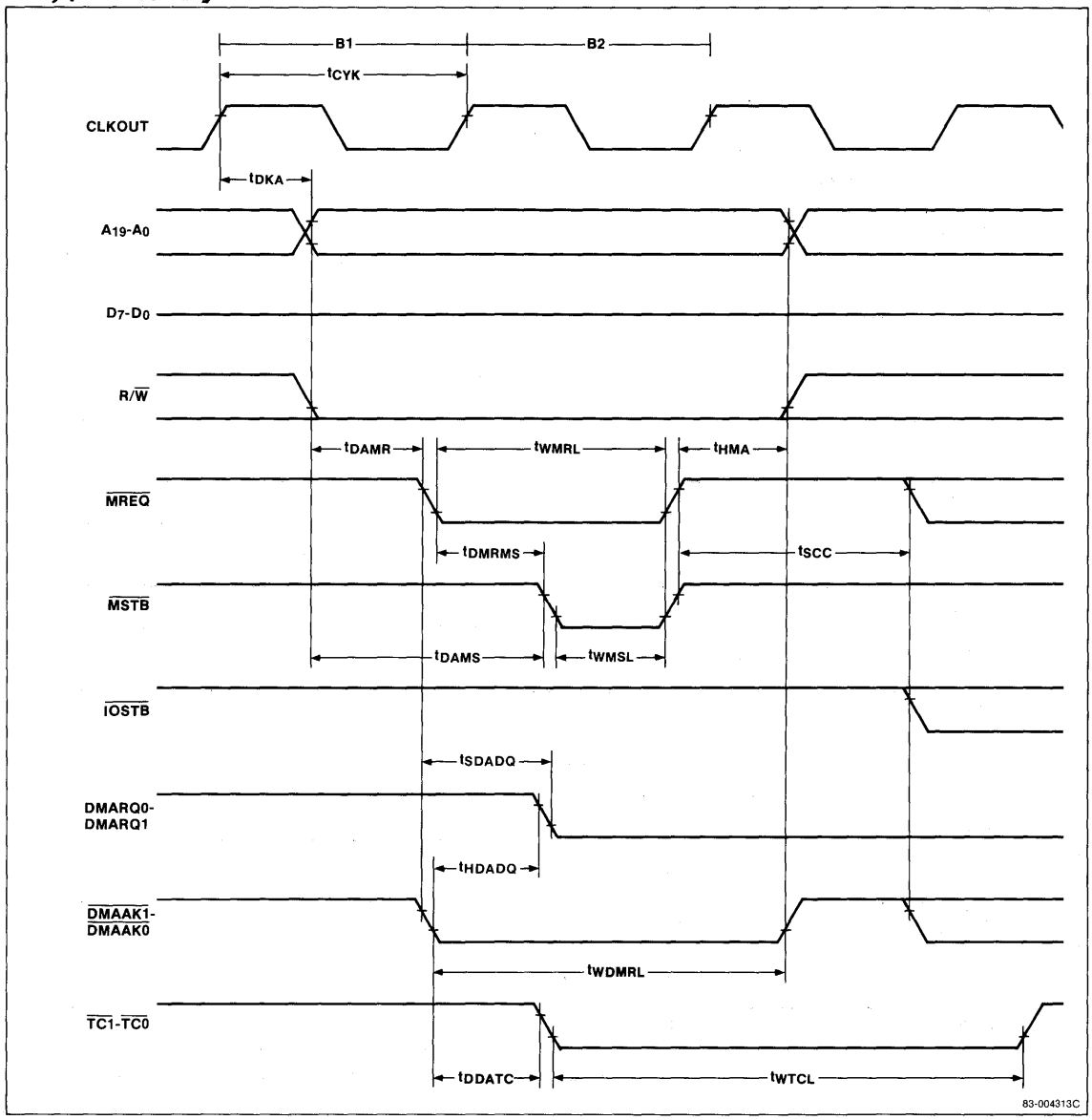
I/O Read


B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADR
tHISA
D7-D0
tDISD
tHISDR
R/W
MREQ
MSTB
tDAIS
tWISL
tSCC
IOSTB
REFRQ
DMAAK1-
DMAAK0
83-004311C

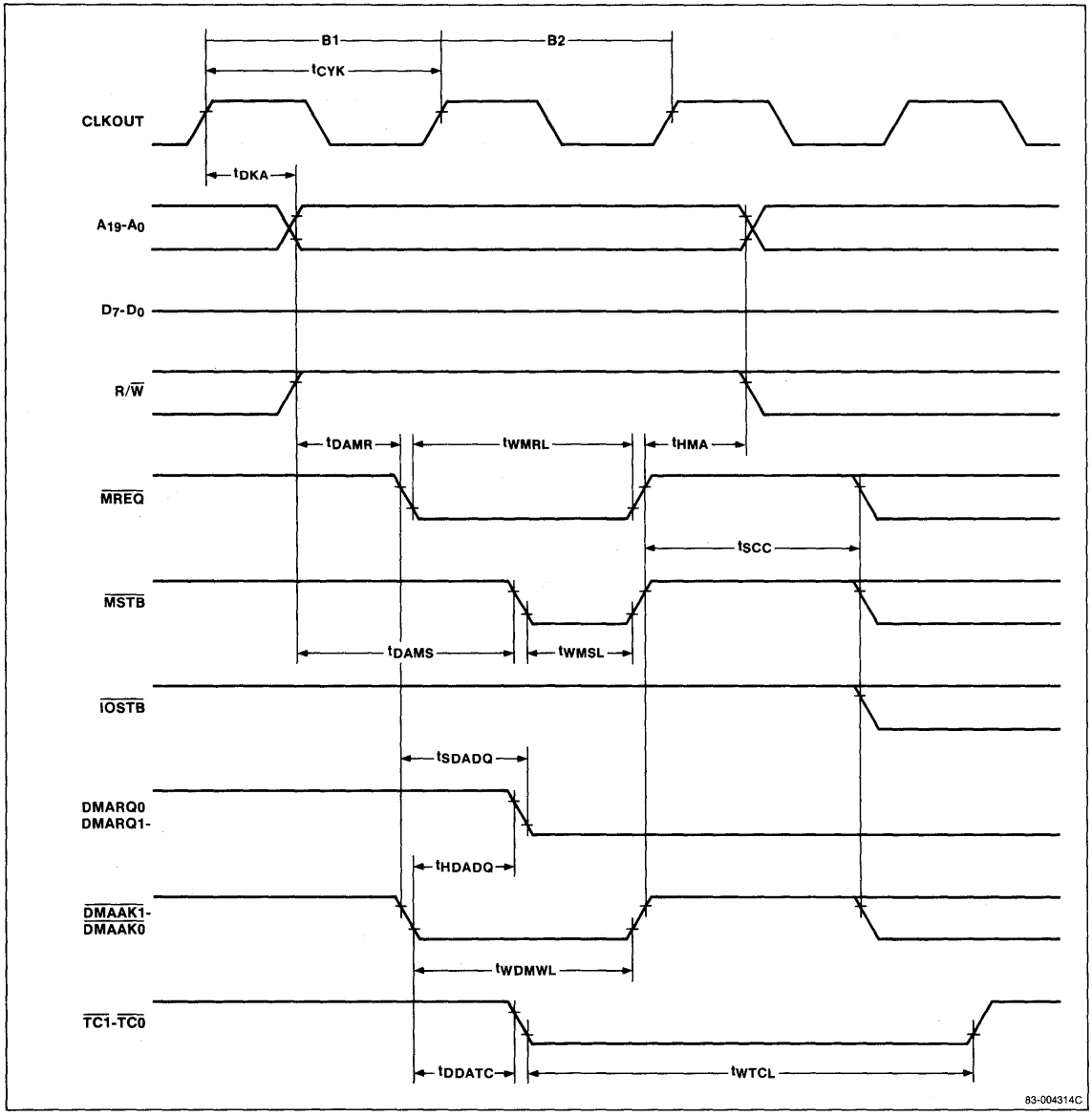
I/O Write


B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADW
tHISA
D7-D0
tSDIS
tHISDW
R/W
MREQ
MSTB
tDAIS
tWISL
tSCC
IOSTB
REFRQ
DMAAK1-
DMAAK0
83-004312C

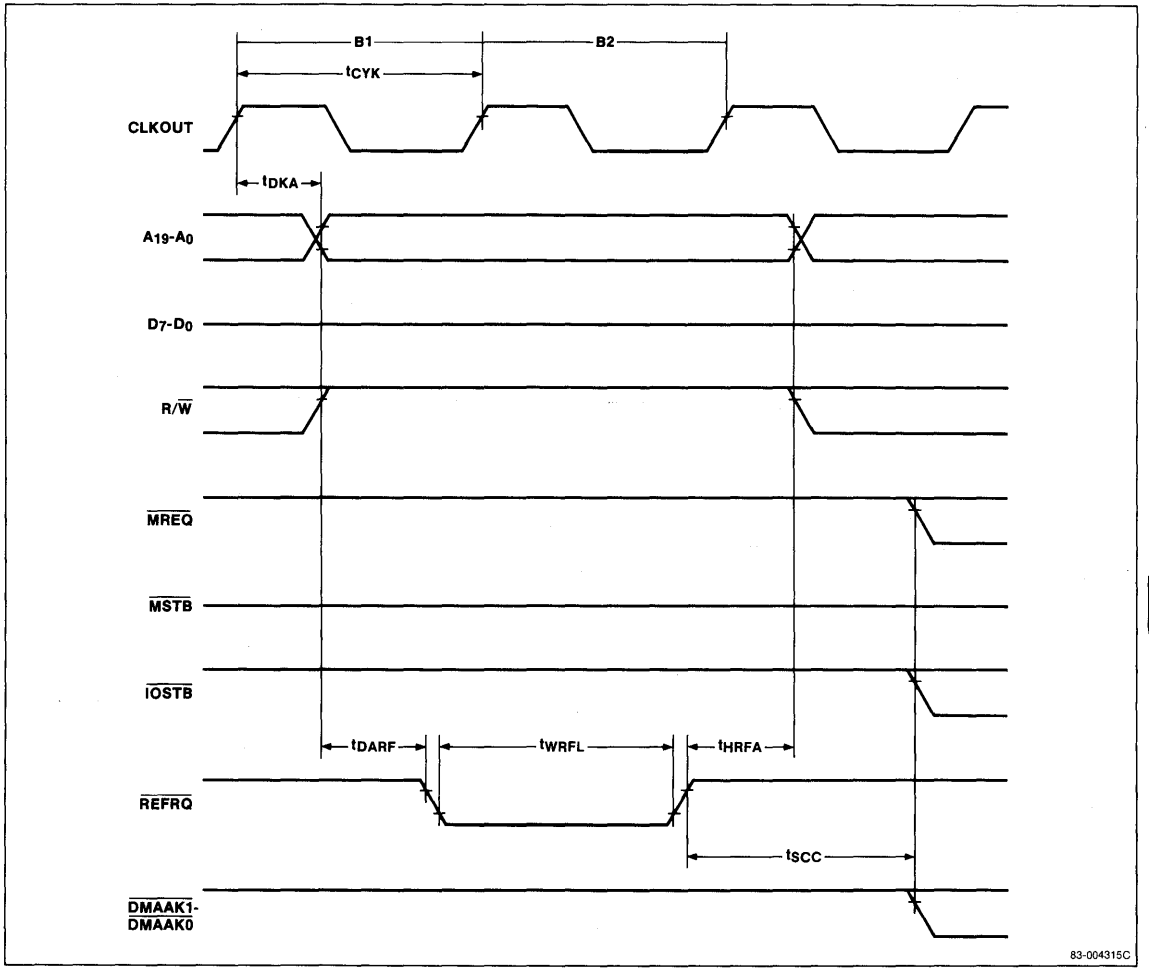
DMA, I/O to Memory



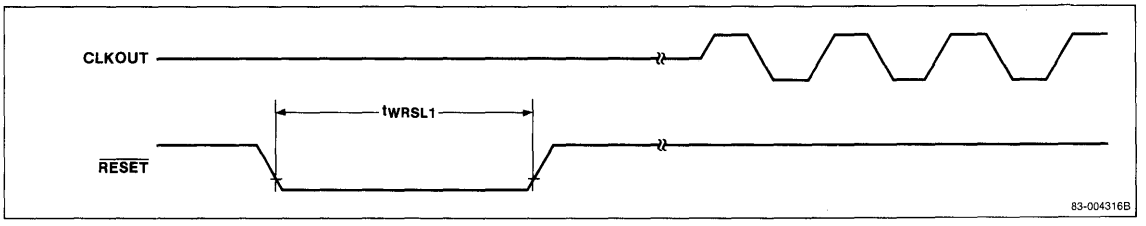
DMA, Memory to I/O



Refresh

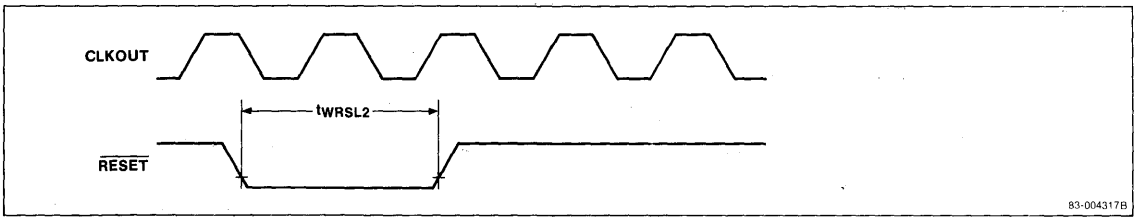


RESET 1

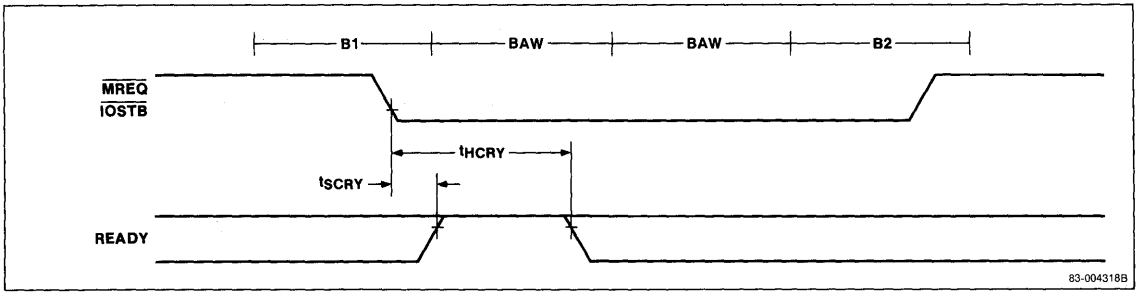


4d

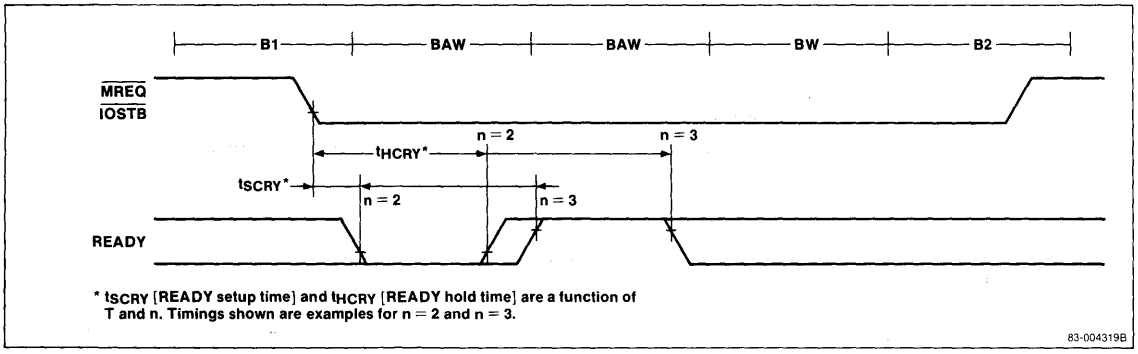
$\overline{RESET}$ 2



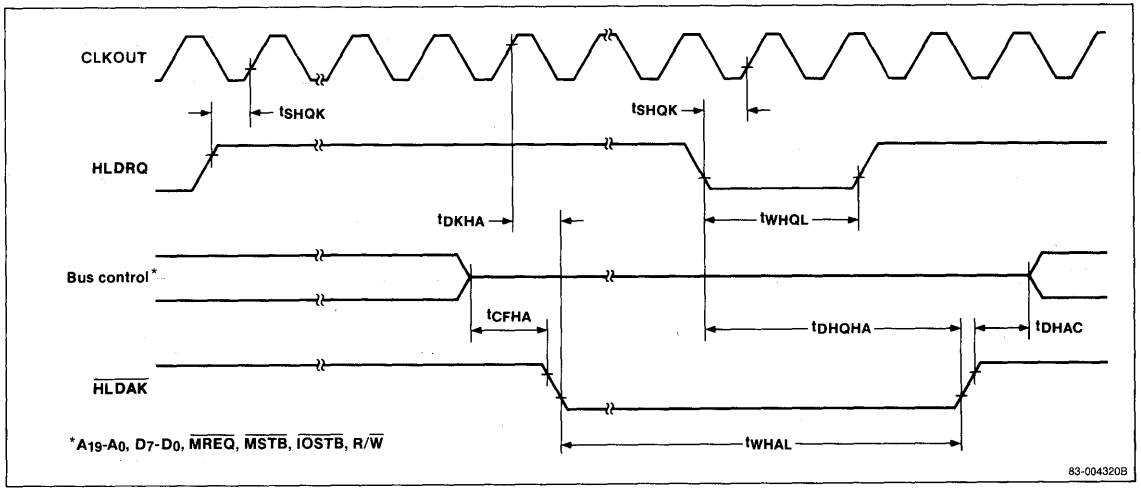
READY 1



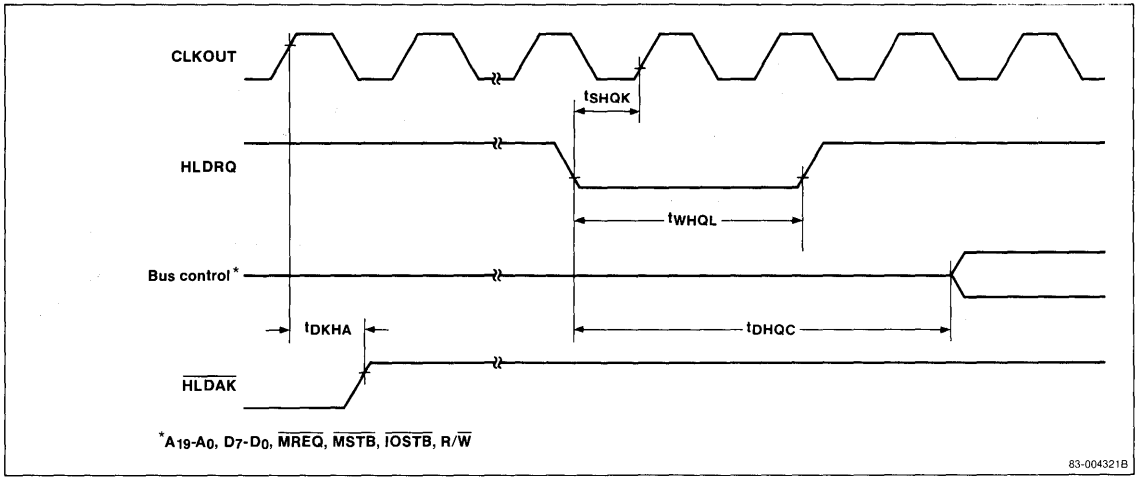
READY 2



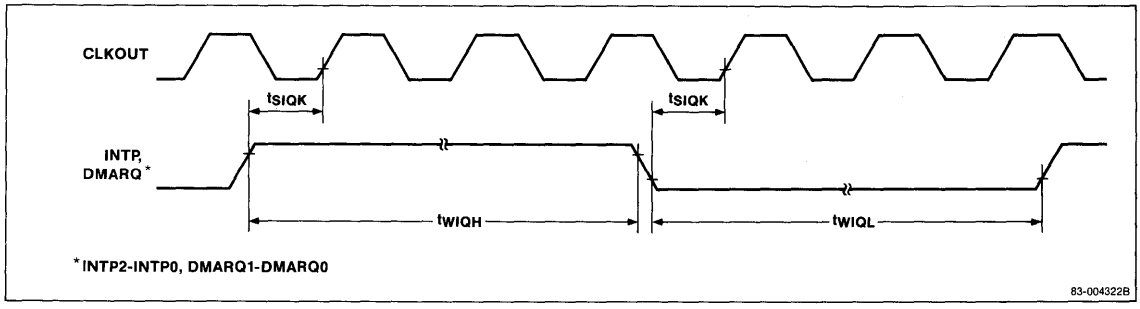
HLDRQ/$\overline{\text{HLDAK}}$ 1



HLDRQ/$\overline{\text{HLDAK}}$ 2

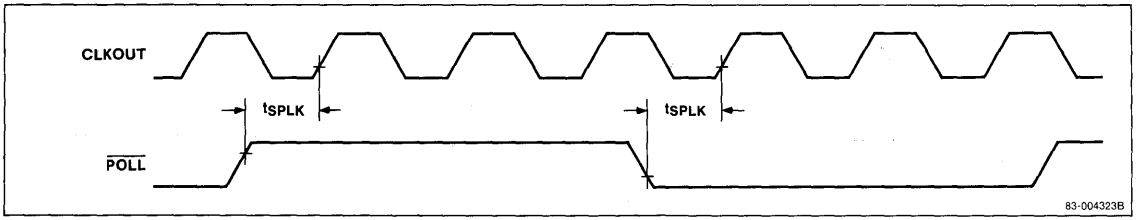


INTP, DMARQ Input

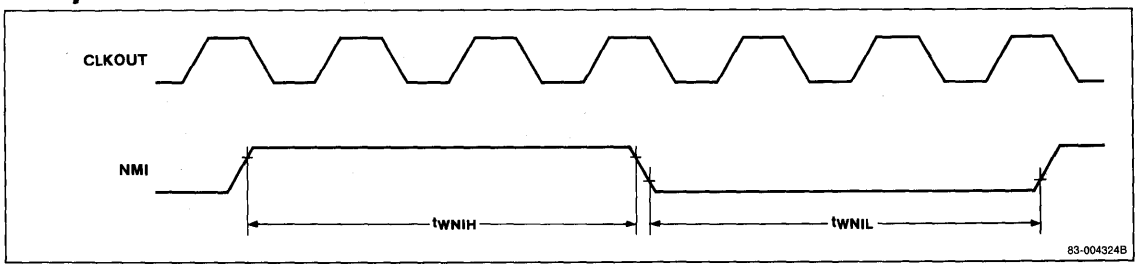


4d

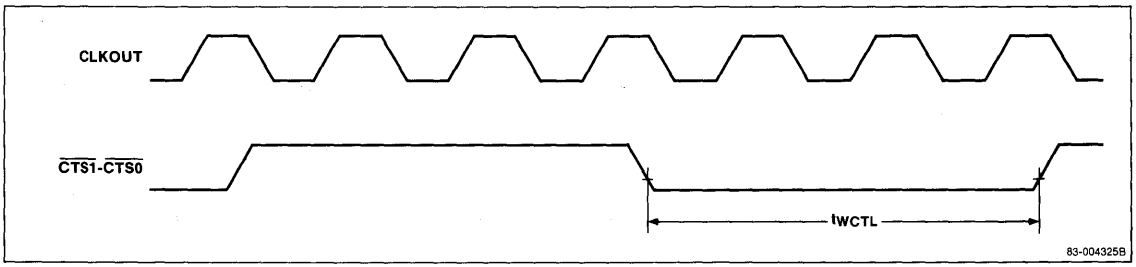
$\overline{POLL}$ Input



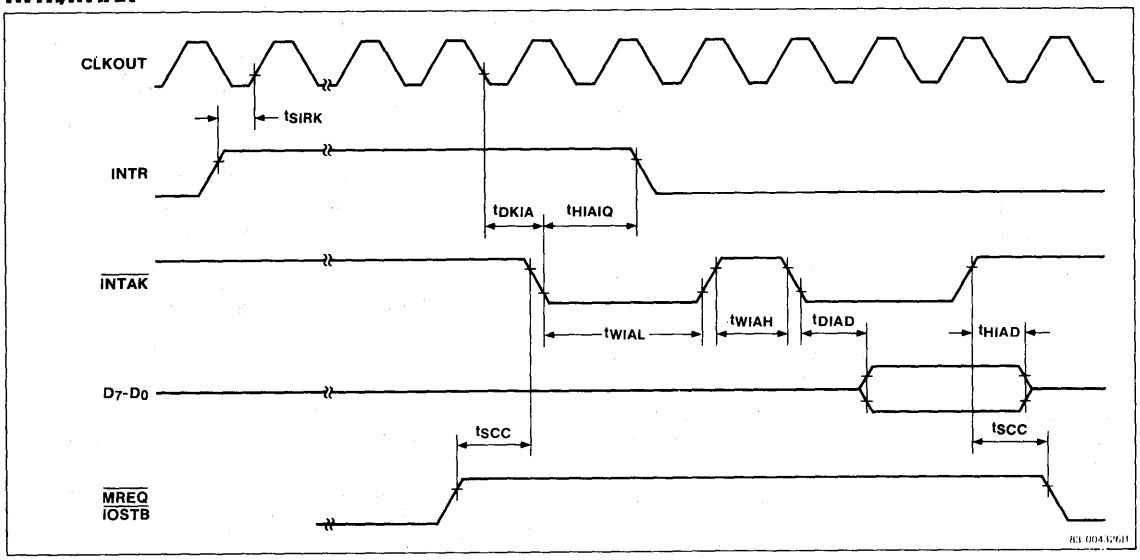
NMI Input



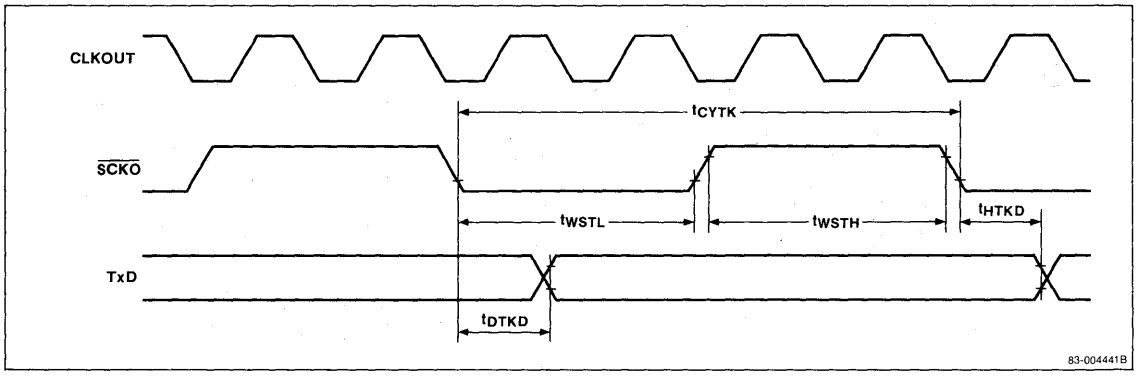
$\overline{CTS}$ Input



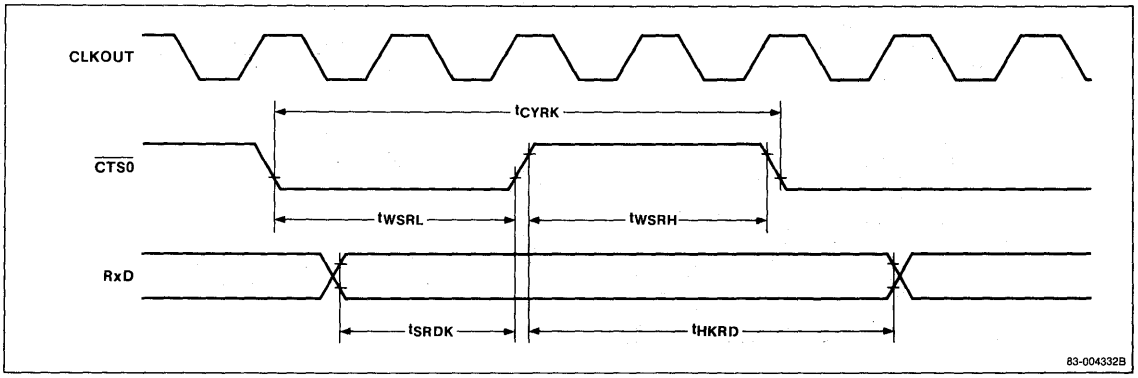
INTR/INTAK



Serial Transmit



Serial Receive



4d

INSTRUCTION SET

Instructions, grouped according to function, are described in a table near the end of this data sheet. Descriptions include source code, operation, opcode, number of bytes, and flag status. Supplementary information applicable to the instruction set is contained in the following tables.

- Symbols and Abbreviations
- Flag Symbols
- 8- and 16-Bit Registers. When mod = 11, the register is specified in the operation code by the byte/word operand (W = 0/1) and reg (000 to 111).
- Segment Registers. The segment register is specified in the operation code by sreg (00, 01, 10, or 11).
- Memory Addressing. The memory addressing mode is specified in the operation code by mod (00, 01, or 10) and mem (000 through 111).
- Instruction Clock Count. This table gives formulas for calculating the number of clock cycles occupied by each type of instruction. The formulas, which depend on byte/word operand and RAM enable/disable, have variables such as EA (effective address), W (wait states), and n (iterations or string instructions).

Symbols and Abbreviations

Identifier	Description
reg	8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
dmem	8- or 16-bit direct memory location
mem	8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
sfr	8-bit special function register location
imm	Constant (0 to FFFFH)
imm16	Constant (0 to FFFFH)
imm8	Constant (0 to FFH)
imm4	Constant (0 to FH)
imm3	Constant (0 to 7)
acc	AW or AL register
sreg	Segment register
src-table	Name of 256-byte translation table
src-block	Name of block addressed by the IX register
dst-block	Name of block addressed by the IY register
near-proc	Procedure within the current program segment
far-proc	Procedure located in another program segment
near-label	Label in the current program segment
short-label	Label between −128 and +127 bytes from the end of instruction
far-label	Label in another program segment
memptr16	Word containing the offset of the memory location within the current program segment to which control is to be transferred
memptr32	Double word containing the offset and segment base address of the memory location to which control is to be transferred
regptr16	16-bit register containing the offset of the memory location within the program segment to which control is to be transferred
pop-value	Number of bytes of the stack to be discarded (0 to 64K bytes, usually even addresses)
fp-op	Immediate data to identify the instruction code of the external floating-point operation
R	Register set
W	Word/byte field (0 to 1)
reg	Register field (000 to 111)
mem	Memory field (000 to 111)
mod	Mode field (00 to 10)
S:W	When S:W = 01 or 11, data = 16 bits. At all other times, data = 8 bits.
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating-point arithmetic chip
AW	Accumulator (16 bits)
AH	Accumulator (high byte)
AL	Accumulator (low byte)
BP	Base pointer register (16 bits)
BW	BW register (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
CW	CW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
DW	DW register (16 bits)
DH	DW register (high byte)
DL	DW register (low byte)
SP	Stack pointer (16 bits)
PC	Program counter (16 bits)
PSW	Program status word (16 bits)

Symbols and Abbreviations (cont)

Identifier	Description
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
PS	Program segment register (16 bits)
SS	Stack segment register (16 bits)
DS_0	Data segment 0 register (16 bits)
DS_1	Data segment 1 register (16 bits)
AC	Auxiliary carry flag
CY	Carry flag
P	Parity flag
S	Sign flag
Z	Zero flag
DIR	Direction flag
IE	Interrupt enable flag
V	Overflow flag
BRK	Break flag
MD	Mode flag
(...)	Values in parentheses are memory contents
disp	Displacement (8 or 16 bits)
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1-bit)
seg	Immediate segment data (16 bits)
offset	Immediate offset data (16 bits)
←	Transfer direction
+	Addition
–	Subtraction
x	Multiplication
÷	Division
%	Modulo
AND	Logical product
OR	Logical sum
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value

Flag Symbols

Identifier	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to the result
u	Undefined
r	Value saved earlier is restored

8- and 16-Bit Registers (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Registers

sreg	Register
00	DS_1
01	PS
10	SS
11	DS_0

Memory Addressing

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Instruction Clock Counts

Mnemonic	Operand	Clocks
ADD	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+8+2W [EA+6+W]
	reg16, mem16	EA+12+4W [EA+8+2W]
	reg8, imm8	5
	reg16, imm8	5
	mem16, imm16	6
	mem8,imm8	EA+9+2W [EA+7+2W]
	mem16, imm8	EA+9+2W [EA+7+2W]
	mem16, imm16	EA+14+4W [EA+10+4W]
	AL, imm8	5
	AW, imm16	6
ADD4S		22+(27+3W)n [22+(25+3W)n]
ADDC	Same as ADD	
ADJ4A		9
ADJ4S		9
ADJBA		17
ADJBS		17
AND	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+8+2W [EA+6+W]
	mem16, reg16	EA+12+4W [EA+8+2W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+9+2W [EA+7+2W]
	mem16, imm16	EA+14+4W [EA+10+4W]
Bcond (conditional branch)		8 or 15
BCWZ		8 or 15
BR	near-label	12
	short-label	12
	regptr16	13
	memptr16	EA+17+2W
	far-label	15
	memptr32	EA+25+4W
BRK	3	55+10W [43+10W]
	imm8	56+10W [44+10W]
BRKCS		15
BRKV		55+10W [43+10W]
BTCLR		29
BUSLOCK		2

Mnemonic	Operand	Clocks
CALL	near-proc	22+2W [18+2W]
	regptr16	22+2W [18+2W]
	memptr16	EA+26+4W [EA+24+4W]
	far-proc	36+4W [34+4W]
	memptr32	EA+36+8W [EA+24+8W]
CHKIND		EA+26+4W
CLR1	CY	2
	DIR	2
	reg8, CL	8
	reg16, CL	8
	mem8, CL	EA+14+2W [EA+12+W]
	mem16, CL	EA+18+4W [EA+14+2W]
	reg8, imm3	7
	reg16, imm4	7
	mem8, imm3	EA+11+2W [EA+9+W]
	mem16, imm4	EA+15+4W [EA+10+2W]
CMP	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+6+W
	mem16, reg16	EA+8+2W
	reg8, imm8	5
	reg16, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+7+W
	mem16, imm8	EA+10+2W
	mem16, imm16	EA+10+2W
	AL, imm8	5
	AW, imm16	6
CMP4S		22+(23+2W)n
CMPBK	mem8, mem8	23+2W [19+2W]
	mem16, mem16	27+4W [21+2W]
CMPBKB		16+(21+2W)n
CMPBKW		16+(25+4W)n
CMPM	mem8	17+W
	mem16	19+2W
CMPMB		16+(15+W)n
CMPMW		16+(17+2W)n
CVTBD		19
CVTBW		3
CVTDB		20
CVTWL		8

Instruction Clock Counts (cont)

Mnemonic	Operand	Clocks
DBNZ		8 or 17
DBNZE		8 or 17
DBNZNE		8 or 17
DEC	reg8 reg16	5 2
	mem8 mem16	EA+11+2W [EA+9+2W] EA+15+4W [EA+11+4W]
DI		4
DISPOSE		12+2W
DIV	AW, reg8 AW, mem8	46-56 EA+48+W to EA+58+W
	DW:AW, reg16 DW; AW, mem16	54-64 EA+58+2W to EA+68+2W
DIVU	AW, reg8 AW, mem8	31 EA+33+W
	DW:AW, reg16 DW: AW, mem16	39 EA+43+2W
DS0:		2
DS1:		2
EI		12
EXT	reg8, reg8 reg8, imm4	41-121 42-122
FINT		2
FP01		60+10W [48+10W]
FP02		60+10W [48+10W]
HALT		0
IN	AL, imm8 AW, imm8	14+W 16+2W
	AL, DW AW, DW	13+W 15+2W
INC	reg8 reg16	5 2
	mem8 mem16	EA+11+2W [EA+9+2W] EA+15+4W [EA+11+4W]
INM	mem8, DW mem16, DW	19+2W [17+2W] 21+4W [17+4W]
	mem8, DW mem16, DW	18+(13+2W)n [18+(11+2W)n] 18+(15+4W)n [18+(11+4W)n]
INS	reg8, reg8 reg8, imm4	63-155 64-156
LDEA		EA+2
LDM	mem8 mem16	12+W 16+(12+2W)n
LDMB	mem16	14+2W
LDMW	mem8	16+(10+W)n

Mnemonic	Operand	Clocks
MOV	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+6+W EA+8+2W
	mem8, reg8 mem16, reg16	EA+4+W [EA+2] EA+6+2W [EA+2]
	reg8, imm8 reg16, imm16	5 6
	mem8, imm8 mem16, imm16	EA+5+W EA+5+2W
	AL, dmem8 AW, dmem16	9+W 11+2W
	dmem8, AL dmem16, AW	7+W [5] 9+2W [5]
	sreg, reg16 sreg, mem16	4 EA+10+2W
	reg16, sreg mem16, sreg	3 EA+7+2W [EA+3]
	AH, PSW PSW, AH	2 3
	DS0, reg16, memptr32 DS1, reg16, memptr32	EA+19+4W EA+19+4W
MOVBK	mem8, mem8 mem16, mem16	20+2W [16+W] 16+(20+4W)n [16+(12+2W)n]
MOVBKB	mem8, mem8	16+(16+2W)n [16+(12+W)n]
MOVBKW	mem16, mem16	24+4W [20+2W]
MOVSPA		16
MOVSPB		11
MUL	AW, AL, reg8 AW, AL, mem8	31-40 EA+33+W to EA+42+W
	DW:AW, AW, reg16 DW:AW, AW, mem16	39-48 EA+43+2W to EA+52+2W
	reg16, reg16, imm8 reg16, mem16, imm8	39-49 EA+43+2W to EA+53+2W
	reg16, reg16, imm16 reg16, mem16, imm16	40-50 EA+44+2W to EA+54+2W
MULU	reg8 mem8	24 EA+26+W
	reg16 mem16	32 EA+34+2W

4d

Instruction Clock Counts (cont)

Mnemonic	Operand	Clocks
NEG	reg8 reg16	5 5
	mem8 mem16	EA+11+2W [EA+9+W] EA+15+4W [EA+11+2W]
NOP		3
NOT	reg8 reg16	5 5
	mem8 mem16	EA+11+2W [EA+9+W] EA+15+4W [EA+11+2W]
NOT1	CY	2
	reg8, CL reg16, CL	7 7
	mem8, CL mem16, CL	EA+13+2W [EA+11+W] EA+17+4W [EA+13+2W]
	reg8, imm3 reg16, imm4	6 6
	mem8, imm3 mem16, imm4	EA+10+2W [EA+8+W] EA+14+4W [EA+10+2W]
OR	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+6+W EA+8+2W
	mem8, reg8 mem16, reg16	EA+8+2W [EA+6+W] EA+12+4W [EA+8+2W]
	reg8, imm8 reg16, imm16	5 6
	mem8, imm8 mem16, imm16	EA+9+W [EA+7+W] EA+14+4W [EA+10+4W]
	AL, imm8 AW, imm16	5 6
OUT	imm8, AL imm8, AW	10+W 10+2W
	DW, AL DW, AW	9+W 9+2W
OUTM	DW, mem8 DW, mem16	19+2W [17+2W] 21+4W [17+4W]
	DW, mem8 DW, mem16	18+(13+2W)n [18+(11+2W)n] 18+(15+4W)n [18+(11+4W)n]
POLL		0
POP	reg16 mem16	12+2W EA+16+4W [EA+12+2W]
	DS1 SS	13+2W 13+2W
	DS0 PSW	13+2W 14+2W
	R	82+16W [58]

Mnemonic	Operand	Clocks
PREPARE	imm16, imm8	imm8 = 0:27+2W imm8 = 1:39+4W imm8 = n > 1:46+19 (n−1)+4W
PS:		2
PUSH	reg16 mem16	10+2W [6] EA+18+4W [EA+14+4W]
	DS1 PS	11+2W [7] 11+2W [7]
	SS DS0	11+2W [7] 11+2W [7]
	PSW R	10+2W [6] 82+16W [50]
	imm8 imm16	13+2W [9] 14+2W [10]
REP		2
REPE		2
REPZ		2
REPC		2
REPNC		2
REPNE		2
REPNZ		2
RET	null pop-value	20+2W 20+2W
	null pop-value	29+4W 30+4W
RETI		43+6W [35+2W]
RETRBI		12
ROL	reg8, 1 reg16, 1	8 8
	mem8, 1 mem16, 1	EA+14+2W [EA+12+W] EA+18+4W [EA+14+2W]
	reg8, CL reg16, CL	11+2n 11+2n
	mem8, CL mem16, CL	EA+17+2W+2n [EA+15+W+2n] EA+21+4W+2n [EA+17+2W+2n]
	reg8, imm8 reg16, imm8	9+2n 9+2n
	mem8, imm8 mem16, imm8	EA+13+2W+2n [EA+11+W+2n] EA+17+4W+2n [EA+13+2W+2n]
ROL4	reg8 mem8	17 EA+18+2W [EA+16+2W]

Instruction Clock Counts (cont)

Mnemonic	Operand	Clocks
ROLC	Same as ROL	
ROR	Same as ROL	
ROR4	reg8 mem8	21 EA+24+2W [EA+22+2W]
RORC	Same as ROL	
SET1	CY DIR	2 2
	reg8, CL reg16, CL	7 7
	mem8, CL mem16, CL	EA+13+2W [EA+11+W] EA+17+4W [EA+13+2W]
	reg8, imm3 reg16, imm4	6 6
	mem8, imm3 mem16, imm4	EA+10+2W [EA+8+W] EA+14+4W [EA+10+2W]
SHL	Same as ROL	
SHR	Same as ROL	
SHRA	Same as ROL	
SS:		2
STM	mem8 mem16	12+2 [10] 16+(10+2W)n [16+(6+2W)n]
STMB	mem8	16+(8+W)n [16+(6+W)n]
STMW	mem16	14+2W [10]
STOP		0
SUB	Same as ADD	
SUB4S		22+(27+3W)n [22+(25+3W)n]
SUBC	Same as ADD	
TEST	reg8, reg8 reg16, reg16	4 4
	reg8, mem8 reg16, mem16	EA+8+W EA+10+2W
	mem8, reg8 mem16, reg16	EA+8+W EA+10+W
	reg8, imm8 reg16, imm16	7 8
	mem8, imm8 mem16, imm16	EA+11+W EA+11+2W
	AL, imm8 AW, imm16	5 6
TEST1	reg8, CL reg16, CL	7 7
	mem8, CL mem16, CL	EA+11+W EA+13+2W
	reg8, imm3 reg16, imm4	6 6
	mem8, imm3 mem16, imm4	EA+8+W EA+10+2W
TRANS		10+W
TRANSB		10+W
TSKSW		11
XCH	reg8, reg8 reg16, reg16	3 3
	reg8, mem8 reg16, mem16	EA+10+2W [EA+8+2W] EA+14+4W [EA+10+4W]
	mem8, reg8 mem16, reg16	EA+10+2W [EA+8+2W] EA+14+4W [EA+10+4W]
	AW, reg16 reg16, AW	4 4
XOR	Same as AND	

Notes:

(1) If the number of clocks is not the same for RAM enabled and RAM disabled conditions, the RAM enabled value is listed first, followed by the RAM disabled value in brackets; for example, EA+8+2W [EA+6+W]

(2) Symbols in the Clocks column are defined as follows.
EA = additional clock cycles required for calculation of the effective address
= 3 (mod 00 or 01) or 4 (mod 10)
W = number of wait states selected by the WTC register
n = number of iterations or string instructions

4d

Execution Clock Counts for Operations

	Byte		Word	
	RAM Enable	RAM Disable	RAM Enable	RAM Disable
Context switch interrupt (Note 1)	—	—	28	28
DMA (Single-step mode) (Note 2)	4+2W	4+2W	8+4W	8+4W
DMA (Demand release mode)	2+W	2+W	4+W	4+2W
DMA (Burst mode)	(4+2W)n	(4+2W)n	(8+4W)n	(8+4W)n
DMA (Single-transfer mode)	2+W	2+W	4+2W	4+2W
Interrupt (INT pin) (Note 3)	—	—	62+10W	50+10W
Macroservice, sfr ← mem (Note 2)	23+W	21+W	27+2W	25+2W
Macroservice, mem ← sfr	22+W	20+W	26+2W	24+2W
Macroservice (Search char mode), sfr ← mem	37+W	37+W	—	—
Macroservice (Search char mode), mem ← sfr	37+W	37+W	—	—
Priority interrupt (Vectored) (Note 1)	—	—	58+10W	46+10W

N = number of clocks to complete the instruction currently executing.

Notes:

(1) Every interrupt has an additional associated latency time of 27 + N clocks. During the 27 clocks, the interrupt controller performs some overhead tasks such as arbitrating priority. This time should be added to the above listed interrupt and macroservice execution times.

(2) The DMA and macroservice clock counts listed are the required number of CPU clocks for each transfer.

(3) When an external interrupt is asserted, a maximum of 6 clocks is required for internal synchronization before the interrupt request flag is set. For an internal interrupt, a maximum of 2 clocks is required.

Bus Controller Latency

		Clocks	
Latency	Mode	Typ	Max
Hold request	Refresh active		9+3W
	Intack active		10+2W
	No refresh or intack		7+2W
DMA request (Notes 1, 2)	Burst	3	14+2W
	Single-step	3	14+2W
	Demand release	3	14+2W
	Single-transfer	4	14+2W

Notes:

(1) The listed DMA latency times are the maximum number of clocks when a DMA request is asserted until $\overline{\text{DMAAK}}$ or $\overline{\text{MREQ}}$ goes low in the corresponding DMA cycle.

(2) The test conditions are: no wait states, no interrupts, no macroservice requests, and no hold requests.

Instruction Set

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer																	
MOV	reg, reg	reg ← reg	1	0	0	0	1	0	1	W	2						
			1	1		reg			reg								
	mem, reg	(mem) ← reg	1	0	0	0	1	0	0	W	2-4						
			mod			reg			mem								
	reg, mem	reg ← (mem)	1	0	0	0	1	0	1	W	2-4						
			mod			reg			mem								
	mem, imm	(mem) ← imm	1	1	0	0	0	1	1	W	3-6						
			mod		0	0	0		mem								
	reg, imm	reg ← imm	1	0	1	1	W		reg		2-3						
	acc, dmem	When W = 0: AL ← (dmem) When W = 1: AH ← (dmem + 1), AL ← (dmem)	1	0	1	0	0	0	0	W	3						
	dmem, acc	When W = 0: (dmem) ← AL When W = 1: (dmem + 1) ← AH, (dmem) ← AL	1	0	1	0	0	0	1	W	3						
	sreg, reg16	sreg ← reg16 sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	2						
			1	1	0	sreg			reg								
	sreg, mem16	sreg ← (mem16) sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	2-4						
			mod		0	sreg			mem								
	reg16, sreg	reg16 ← sreg	1	0	0	0	1	1	0	0	2						
			1	1	0	sreg			reg								
	mem16, sreg	(mem16) ← sreg	1	0	0	0	1	1	0	0	2-4						
			mod		0	sreg			mem								
	DS0, reg16, mem32	reg16 ← (mem32), DS0 ← (mem32 + 2)	1	1	0	0	0	1	0	1	2-4						
			mod			reg			mem								
	DS1, reg16, mem32	reg16 ← (mem32), DS1 ← (mem32 + 2)	1	1	0	0	0	1	0	0	2-4						
			mod			reg			mem								
	AH, PSW	AH ← S, Z, x, AC, x, P, x, CY	1	0	0	1	1	1	1	1	1						
	PSW, AH	S, Z, x, AC, x, P, x, CY ← AH	1	0	0	1	1	1	1	0	1	x	x		x	x	x
LDEA	reg16, mem16	reg16 ← mem16	1	0	0	0	1	1	0	1	2-4						
			mod			reg			mem								
TRANS	src-table	AL ← (BW + AL)	1	1	0	1	0	1	1	1	1						
XCH	reg, reg	reg ↔ reg	1	0	0	0	0	1	1	W	2						
			1	1		reg			reg								
	mem, reg or reg, mem	(mem) ↔ reg	1	0	0	0	0	1	1	W	2-4						
			mod			reg			mem								
	AW, reg16 or reg16, AW	AW ↔ reg16	1	0	0	1	0		reg		1						

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Repeat Prefixes																	
REPC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (–1). If there is a waiting interrupt, it is processed. When CY ≠ 1, exit the loop.	0	1	1	0	0	1	0	1	1						
REPNC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (–1). If there is a waiting interrupt, it is processed. When CY ≠ 0, exit the loop.	0	1	1	0	0	1	0	0	1						
REP REPE REPZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (–1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 1, exit the loop.	1	1	1	1	0	0	1	1	1						
REPNE REPNZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (–1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 0, exit the loop.	1	1	1	1	0	0	1	0	1						
Primitive Block Transfer																	
MOVBK	dst-block, src-block	When W = 0: (IY) ← (IX) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX–1, IY ← IY–1 When W = 1: (IY + 1, IY) ← (IX + 1, IX) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX–2, IY ← IY–2	1	0	1	0	0	1	0	W	1						
CMPBK	src-block, dst-block	When W = 0: (IX) – (IY) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX–1, IY ← IY–1 When W = 1: (IX + 1, IX) – (IY + 1, IY) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX–2, IY ← IY–2	1	0	1	0	0	1	1	W	1	x	x	x	x	x	x
CMPM	dst-block	When W = 0: AL – (IY) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY–1 When W = 1: AW – (IY + 1, IY) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY–2	1	0	1	0	1	1	1	W	1	x	x	x	x	x	x
LDM	src-block	When W = 0: AL ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX–1 When W = 1: AW ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX–2	1	0	1	0	1	1	0	W	1						
STM	dst-block	When W = 0: (IY) ← AL DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY–1 When W = 1: (IY + 1, IY) ← AW DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY–2	1	0	1	0	1	0	1	W	1						

Instruction Set (cont)

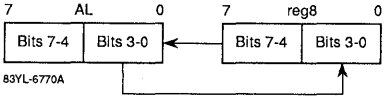
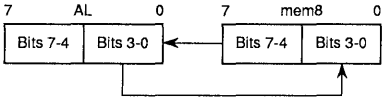
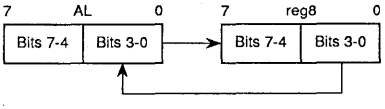
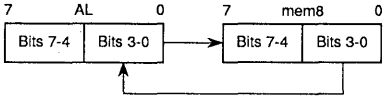
Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Field Transfer																	
INS	reg8, reg8	16-bit field ← AW	0	0	0	0	1	1	1	1	3						
			0	0	1	1	0	0	0	1							
			1	1		reg			reg								
	reg8, imm4	16-bit field ← AW	0	0	0	0	1	1	1	1	4						
			0	0	1	1	1	0	0	1							
			1	1	0	0	0		reg								
EXT	reg8, reg8	AW ← 16-bit field	0	0	0	0	1	1	1	1	3						
			0	0	1	1	0	0	1	1							
			1	1		reg			reg								
	reg8, imm4	AW ← 16-bit field	0	0	0	0	1	1	1	1	4						
			0	0	1	1	1	0	1	1							
			1	1	0	0	0		reg								
I/O																	
IN	acc, imm8	When W = 0: AL ← (imm8) When W = 1: AH ← (imm8 + 1), AL ← (imm8)	1	1	1	0	0	1	0	W	2						
	acc, DW	When W = 0: AL ← (DW) When W = 1: AH ← (DW + 1), AL ← (DW)	1	1	1	0	1	1	0	W	1						
OUT	imm8, acc	When W = 0: (imm8) ← AL When W = 1: (imm8 + 1) ← AH, (imm8) ← AL	1	1	1	0	0	1	1	W	2						
	DW, acc	When W = 0: (DW) ← AL When W = 1: (DW + 1) ← AH, (DW) ← AL	1	1	1	0	1	1	1	W	1						
Primitive Block I/O Transfer																	
INM	dst-block, DW	When W = 0: (IY) ← (DW) DIR = 0: IY ← IY + 1 DIR = 1: IY ← IY − 1 When W = 1: (IY + 1, IY) ← (DW + 1, DW) DIR = 0: IY ← IY + 2 DIR = 1: IY ← IY − 2	0	1	1	0	1	1	0	W	1						
OUTM	DW, src-block	When W = 0: (DW) ← (IX) DIR = 0: IX ← IX + 1 DIR = 1: IX ← IX − 1 When W = 1: (DW + 1, DW) ← (IX + 1, IX) DIR = 0: IX ← IX + 2 DIR = 1: IX ← IX − 2	0	1	1	0	1	1	1	W	1						

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Addition/Subtraction																	
ADD	reg, reg	reg ← reg + reg	0	0	0	0	0	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) + reg	0	0	0	0	0	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg + (mem)	0	0	0	0	0	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg + imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	0	0		reg								
	mem, imm	(mem) ← (mem) + imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	0	0		mem								
	acc, imm	When W = 0: AL ← AL + imm When W = 1: AW ← AW + imm	0	0	0	0	0	1	0	W	2-3	x	x	x	x	x	x
ADDC	reg, reg	reg ← reg + reg + CY	0	0	0	1	0	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) + reg + CY	0	0	0	1	0	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg + (mem) + CY	0	0	0	1	0	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg + imm + CY	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	1	0		reg								
	mem, imm	(mem) ← (mem) + imm + CY	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	1	0		mem								
	acc, imm	When W = 0: AL ← AL + imm + CY When W = 1: AW ← AW + imm + CY	0	0	0	1	0	1	0	W	2-3	x	x	x	x	x	x
SUB	reg, reg	reg ← reg − reg	0	0	1	0	1	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) − reg	0	0	1	0	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg − (mem)	0	0	1	0	1	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg − imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	1	0	1		reg								
	mem, imm	(mem) ← (mem) − imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		1	0	1		mem								
	acc, imm	When W = 0: AL ← AL − imm When W = 1: AW ← AW − imm	0	0	1	0	1	1	0	W	2-3	x	x	x	x	x	x
SUBC	reg, reg	reg ← reg − reg − CY	0	0	0	1	1	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) − reg − CY	0	0	0	1	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg − (mem) − CY	0	0	0	1	1	0	1	W	2-4	x	x	x	x	x	
			mod			reg			mem								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Addition/Subtraction (cont)																	
SUBC	reg, imm	reg ← reg − imm − CY	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	1	1		reg								
	mem, imm	(mem) ← (mem) − imm − CY	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	1	1		mem								
	acc, imm	When W = 0: AL ← AL − imm − CY When W = 1: AW ← AW − imm − CY	0	0	0	1	1	1	0	W	2-3	x	x	x	x	x	x
BCD Operation																	
ADD4S		dst BCD string ← dst BCD string + src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	0	0	0							
SUB4S		dst BCD string ← dst BCD string − src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	0	1	0							
CMP4S		dst BCD string − src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	1	1	0							
ROL4	reg 8		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	0	0	0							
			1	1	0	0	0		reg								
	mem 8		0	0	0	0	1	1	1	1	3-5						
			0	0	1	0	1	0	0	0							
			mod		0	0	0		mem								
ROR4	reg 8		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	0	1	0							
			1	1	0	0	0		reg								
	mem 8		0	0	0	0	1	1	1	1	3-5						
			0	0	1	0	1	0	1	0							
			mod		0	0	0		mem								
BCD Adjust																	
ADJBA		When (AL AND 0FH) > 9 or AC = 1: AL ← AL + 6, AH ← AH + 1, AC ← 1, CY ← AC, AL ← AL AND 0FH	0	0	1	1	0	1	1	1	1	x	x	u	u	u	u
ADJ4A		When (AL AND 0FH) > 9 or AC = 1: AL ← AL + 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY = 1: AL ← AL + 60H, CY ← 1	0	0	1	0	0	1	1	1	1	x	x	u	x	x	x
ADJBS		When (AL AND 0FH) > 9 or AC = 1: CY ← AC, AL ← AL AND 0FH	0	0	1	1	1	1	1	1	1	x	x	u	u	u	u
ADJ4S		When (AL AND 0FH) > 9 or AC = 1: AL ← AL − 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY = 1: AL ← AL + 60H, CY ← 1	0	0	1	0	1	1	1	1	1	x	x	u	x	x	x

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Increment/Decrement																	
INC	reg8	reg8 ← reg8 + 1	1	1	1	1	1	1	1	0	2	x		x	x	x	x
			1	1	0	0	0	reg									
	mem	(mem) ← (mem) + 1	1	1	1	1	1	1	1	W	2-4	x		x	x	x	x
			mod		0	0	0	mem									
	reg16	reg16 ← reg16 + 1	0	1	0	0	0	reg			1	x		x	x	x	x
DEC	reg8	reg8 ← reg8 − 1	1	1	1	1	1	1	1	0	2	x		x	x	x	x
			1	1	0	0	1	reg									
	mem	(mem) ← (mem) − 1	1	1	1	1	1	1	1	W	2-4	x		x	x	x	x
			mod		0	0	1	mem									
	reg16	reg16 ← reg16 − 1	0	1	0	0	1	reg			1	x		x	x	x	x
Multiplication																	
MULU	reg8	AW ← AL x reg8 AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	2	u	x	x	u	u	u
			1	1	1	0	0	reg									
	mem8	AW ← AL x (mem8) AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	2-4	u	x	x	u	u	u
			mod		1	0	0	mem									
	reg16	DW, AW ← AW x reg16 DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	2	u	x	x	u	u	u
			1	1	1	0	0	reg									
	mem16	DW, AW ← AW x (mem16) DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	2-4	u	x	x	u	u	u
			mod		1	0	0	mem									
MUL	reg8	AW ← AL x reg8 AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	2	u	x	x	u	u	u
			1	1	1	0	1	reg									
	mem8	AW ← AL x (mem8) AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	2-4	u	x	x	u	u	u
			mod		1	0	1	mem									
	reg16	DW, AW ← AW x reg16 DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	2	u	x	x	u	u	u
			1	1	1	0	1	reg									
	mem16	DW, AW ← AW x (mem16) DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	2-4	u	x	x	u	u	u
			mod		1	0	1	mem									
	reg16, reg16, imm8	reg16 ← reg16 x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	3	u	x	x	u	u	u
			1	1	reg			reg									
	reg16, mem16, imm8	reg16 ← (mem16) x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	3-5	u	x	x	u	u	u
			mod		reg			mem									
	reg16, reg16, imm16	reg16 ← reg16 x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	4	u	x	x	u	u	u
			1	1	reg			reg									
	reg16, mem16, imm16	reg16 ← (mem16) x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	4-6	u	x	x	u	u	u
			mod		reg			mem									

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Unsigned Division																	
DIVU	reg8	temp ← AW When temp ÷ reg8 > FFH: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % reg8, AL ← temp ÷ reg8	1 1	1 1	1 1	1 1	0 0	1	1 reg	0	2	u	u	u	u	u	u
	mem8	temp ← AW When temp ÷ (mem8) > FFH: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % (mem8), AL ← temp ÷ (mem8)	1 mod	1	1	1	0 0	1	1 mem	0	2-4	u	u	u	u	u	u
	reg16	temp ← AW When temp ÷ reg16 > FFFFH: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % reg16, AL ← temp ÷ reg16	1 1	1 1	1 1	1 1	0 0	1	1 reg	1	2	u	u	u	u	u	u
	mem16	temp ← AW When temp ÷ (mem16) > FFFFH: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp% (mem16), AL ← temp ÷ (mem16)	1 mod	1	1	1	0 0	1	1 mem	1	2-4	u	u	u	u	u	u
Signed Division																	
DIV	reg8	temp ← AW When temp ÷ reg8 > 0 and temp ÷ reg8 > 7FH or temp ÷ reg8 < 0 and temp ÷ reg8 < 0−7FH−1: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % reg8, AL ← temp ÷ reg8	1 1	1 1	1 1	1 1	0 1	1	1 reg	0	2	u	u	u	u	u	u
	mem8	temp ← AW When temp ÷ (mem8) > 0 and (mem8) > 7 FH or temp ÷ (mem8) < 0 and temp ÷ (mem8) < 0−7FH−1: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % (mem8), AL ← temp ÷ (mem8)	1 mod	1	1	1	0 1	1	1 mem	0	2-4	u	u	u	u	u	u

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Signed Division (cont)																	
DIV	reg16	temp ← DW, AW When temp ÷ reg16 > 0 and reg16 > 7FFFH or temp ÷ reg16 < 0 − 7FFFH − 1: (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % reg16, AL ← temp ÷ reg16	1	1	1	1	0	1	1	1	2	u	u	u	u	u	u
			1	1	1	1	1		reg								
	mem16	temp ← DW, AW When temp ÷ (mem16) > 0 and (mem16) > 7 FFFH or temp ÷ (mem16) < 0 and temp ÷ (mem16) < 0 − 7FFFH − 1: (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % (mem16), AL ← temp ÷ (mem16)	1	1	1	1	0	1	1	1	2-4	u	u	u	u	u	u
			mod		1	1	1		mem								
Data Conversion																	
CVTBD		AH ← AL ÷ 0AH, AL ← AL % 0AH	1	1	0	1	0	1	0	0	2	u	u	u	x	x	x
			0	0	0	0	1	0	1	0							
CVTDB		AH ← 0, AL ← AH x 0AH + AL	1	1	0	1	0	1	0	1	2	u	u	u	x	x	x
			0	0	0	0	1	0	1	0							
CVTBW		When AL < 80H: AH ← 0 All other times: AH ← FFH	1	0	0	1	1	0	0	0	1						
CVTWL		When AL < 8000H: DW ← 0 All other times: DW ← FFFFH	1	0	0	1	1	0	0	1	1						
Comparison																	
CMP	reg, reg	reg − reg	0	0	1	1	1	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) − reg	0	0	1	1	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg − (mem)	0	0	1	1	1	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg − imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	1	1	1		reg								
	mem, imm	(mem) − imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		1	1	1		mem								
	acc, imm	When W = 0: AL − imm When W = 1: AW − imm	0	0	1	1	1	1	0	W	2-3	x	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Complement																	
NOT	reg	reg ← $\overline{\text{reg}}$	1	1	1	1	0	1	1	W	2						
			1	1	0	1	0		reg								
	mem	(mem) ← ($\overline{\text{mem}}$)	1	1	1	1	0	1	1	W	2-4						
			mod		0	1	0		mem								
NEG	reg	reg ← $\overline{\text{reg}}$ + 1	1	1	1	1	0	1	1	W	2	x	x	x	x	x	x
			1	1	0	1	1		reg								
	mem	(mem) ← ($\overline{\text{mem}}$) + 1	1	1	1	1	0	1	1	W	2-4	x	x	x	x	x	x
			mod		0	1	1		mem								
Logical Operation																	
TEST	reg, reg	reg AND reg	1	0	0	0	0	1	0	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg or reg, mem	(mem) AND reg	1	0	0	0	0	1	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg AND imm	1	1	1	1	0	1	1	W	3-4	u	0	0	x	x	x
			1	1	0	0	0		reg								
	mem, imm	(mem) AND imm	1	1	1	1	0	1	1	W	3-6	u	0	0	x	x	x
			mod		0	0	0		mem								
	acc, imm	When W = 0: AL AND imm8 When W = 1: AW AND imm8	1	0	1	0	1	0	0	W	2-3	u	0	0	x	x	x
AND	reg, reg	reg ← reg AND reg	0	0	1	0	0	0	1	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) AND reg	0	0	1	0	0	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg AND (mem)	0	0	1	0	0	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg AND imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	1	0	0		reg								
	mem, imm	(mem) ← (mem) AND imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod		1	0	0		mem								
	acc, imm	When W = 0: AL ← AL AND imm8 When W = 1: AW ← AW AND imm16	0	0	1	0	0	1	0	W	2-3	u	0	0	x	x	x

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Logical Operation (cont)																	
OR	reg, reg	reg ← reg OR reg	0	0	0	0	1	0	1	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) OR reg	0	0	0	0	1	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg OR (mem)	0	0	0	0	1	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg OR imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	0	0	1		reg								
	mem, imm	(mem) ← (mem) OR imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod		0	0	1		mem								
	acc, imm	When W = 0: AL ← AL OR imm8 When W = 1: AW ← AW OR imm16	0	0	0	0	1	1	0	W	2-3	u	0	0	x	x	x
XOR	reg, reg	reg ← reg XOR reg	0	0	1	1	0	0	1	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) XOR reg	0	0	1	1	0	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg XOR (mem)	0	0	1	1	0	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg XOR imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	1	1	0		reg								
	mem, imm	(mem) ← (mem) XOR imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod		1	1	0		mem								
	acc, imm	When W = 0: AL ← AL XOR imm8 When W = 1: AW ← AW XOR imm16	0	0	1	1	0	1	0	W	2-3	u	0	0	x	x	x
Bit Operation																	
TEST1	reg8, CL	reg8 bit no. CL = 0: Z ← 1 reg8 bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3	u	0	0	u	u	x
			0	0	0	1	0	0	0	0							
			1	1	0	0	0		reg								
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3-5	u	0	0	u	u	x
			0	0	0	1	0	0	0	0							
			mod		0	0	0		mem								
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3	u	0	0	u	u	x
			0	0	0	1	0	0	0	1							
			1	1	0	0	0		reg								
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3-5	u	0	0	u	u	x
			0	0	0	1	0	0	0	1							
			mod		0	0	0		mem								
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0	0	0	0	1	1	1	1	4	u	0	0	u	u	x
			0	0	0	1	1	0	0	0							
			1	1	0	0	0		reg								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
TEST1	mem8, imm3	(mem8) bit no. imm3 = 0: Z ← 1 (mem8) bit no. imm3 = 1: Z ← 0	0	0	0	0	1	1	1	1	4-6	u	0	0	u	u	x
			0	0	0	1	1	0	0	0							
			mod		0	0	0	mem									
	reg16, imm4	reg16 bit no. imm4 = 0: Z ← 1 reg16 bit no. imm4 = 1: Z ← 0	0	0	0	0	1	1	1	1	4	u	0	0	u	u	x
			0	0	0	1	1	0	0	1							
			1	1	0	0	0	reg									
	mem16, imm4	(mem16) bit no. imm4 = 0: Z ← 1 (mem16) bit no. imm4 = 1: Z ← 0	0	0	0	0	1	1	1	1	4-6	u	0	0	u	u	x
			0	0	0	1	1	0	0	1							
			mod		0	0	0	mem									
NOT1	reg8, CL	reg8 bit no. CL ← $\overline{\text{reg8 bit no. CL}}$	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	1	0							
			1	1	0	0	0	reg									
	mem8, CL	(mem8) bit no. CL ← $\overline{\text{(mem8) bit no. CL}}$	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	1	0							
			mod		0	0	0	mem									
	reg16, CL	reg16 bit no. CL ← $\overline{\text{reg16 bit no. CL}}$	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	1	1							
			1	1	0	0	0	reg									
	mem16, CL	(mem16) bit no. CL ← $\overline{\text{(mem16) bit no. CL}}$	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	1	1							
			mod		0	0	0	mem									
	reg8, imm3	reg8 bit no. imm3 ← $\overline{\text{reg8 bit no. imm3}}$	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	1	0							
			1	1	0	0	0	reg									
	mem8, imm3	(mem8) bit no. imm3 ← $\overline{\text{(mem8) bit no. imm3}}$	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	1	0							
			mod		0	0	0	mem									
	reg16, imm4	reg16 bit no. imm4 ← $\overline{\text{reg16 bit no. imm4}}$	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	1	1							
			1	1	0	0	0	reg									
	mem16, imm4	(mem16) bit no. imm4 ← $\overline{\text{(mem16) bit no. imm4}}$	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	1	1							
			mod		0	0	0	mem									
	CY	CY ← CY	1	1	1	1	0	1	0	1	1		x				

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
CLR1	reg8, CL	reg8 bit no. CL ← 0	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	0	1	0							
			1	1	0	0	0	reg									
	mem8, CL	(mem8) bit no. CL ← 0	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	0	1	0							
			mod		0	0	0	mem									
	reg16, CL	reg16 bit no. CL ← 0	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	0	1	1							
			1	1	0	0	0	reg									
	mem16, CL	(mem16) bit no. CL ← 0	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	0	1	1							
			mod		0	0	0	mem									
	reg8, imm3	reg8 bit no. imm3 ← 0	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	0	1	0							
			1	1	0	0	0	reg									
	mem8, imm3	(mem8) bit no. imm3 ← 0	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	0	1	0							
			mod		0	0	0	mem									
	reg16, imm4	reg16 bit no. imm4 ← 0	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	0	1	1							
			1	1	0	0	0	reg									
	mem16, imm4	(mem16) bit no. imm4 ← 0	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	0	1	1							
			mod		0	0	0	mem									
	CY	CY ← 0	1	1	1	1	1	0	0	0	1		0				
	DIR	DIR ← 0	1	1	1	1	1	1	0	0	1						
SET1	reg8, CL	reg8 bit no. CL ← 1	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	0	0							
			1	1	0	0	0	reg									
	mem8, CL	(mem8) bit no. CL ← 1	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	0	0							
			mod		0	0	0	mem									
	reg16, CL	reg16 bit no. CL ← 1	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	0	1							
			1	1	0	0	0	reg									
	mem16, CL	(mem16) bit no. CL ← 1	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	0	1							
			mod		0	0	0	mem									

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
SET1	reg8, imm3	reg8 bit no. imm3 ← 1	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	0	0							
			1	1	0	0	0	reg									
	mem8, imm3	(mem8) bit no. imm3 ← 1	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	0	0							
			mod		0	0	0	mem									
	reg16, imm4	reg16 bit no. imm4 ← 1	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	0	1							
			1	1	0	0	0	reg									
	mem16, imm4	(mem16) bit no. imm4 ← 1	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	0	1							
			mod		0	0	0	mem									
	CY	CY ← 1	1	1	1	1	1	0	0	1	1		1				
	DIR	DIR ← 1	1	1	1	1	1	1	0	1	1						
Shift																	
SHL	reg, 1	CY ← MSB of reg, reg ← reg x 2 When MSB of reg ≠ CY, V ← 1 When MSB of reg = CY, V ← 0	1	1	0	1	0	0	0	W	2	u	x	x	x	x	x
			1	1	1	0	0	reg									
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 When MSB of (mem) ≠ CY, V ← 1 When MSB of (mem) = CY, V ← 0	1	1	0	1	0	0	0	W	2-4	u	x	x	x	x	x
			mod		1	0	0	mem									
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1	1	0	1	0	0	1	W	2	u	x	u	x	x	x
			1	1	1	0	0	reg									
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1	1	0	1	0	0	1	W	2-4	u	x	u	x	x	x
			mod		1	0	0	mem									
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1	1	0	0	0	0	0	W	3	u	x	u	x	x	x
			1	1	1	0	0	reg									
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1	1	0	0	0	0	0	W	3-5	u	x	u	x	x	x
			mod		1	0	0	mem									

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Shift (cont)																	
SHR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 When MSB of reg ≠ bit following MSB of reg: V ← 1 When MSB of reg = bit following MSB of reg: V ← 0	1 1	1 1	0 1	1 0	0 1	0	0 reg	W	2	u	x	x	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 When MSB of (mem) ≠ bit following MSB of (mem): V ← 1 When MSB of (mem) = bit following MSB of (mem): V ← 0	1 mod	1	0 1	1 0	0 1	0	0 mem	W	2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1 1	1 1	0 1	1 0	0 1	0	1 reg	W	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 mod	1	0 1	1 0	0 1	0	1 mem	W	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1 1	1 1	0 1	0 0	0 1	0	0 reg	W	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 mod	1	0 1	0 0	0 1	0	0 mem	W	3-5	u	x	u	x	x	x
SHRA	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, V ← 0 MSB of operand does not change	1 1	1 1	0 1	1 1	0 1	0	0 reg	W	2	u	x	0	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, V ← 0, MSB of operand does not change	1 mod	1	0 1	1 1	0 1	0	0 mem	W	2-4	u	x	0	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1	1 1	0 1	1 1	0 1	0	1 reg	W	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1 mod	1	0 1	1 1	0 1	0	1 mem	W	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1	1 1	0 1	0 1	0 1	0	0 reg	W	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1 mod	1	0 1	0 1	0 1	0	0 mem	W	3-5	u	x	u	x	x	x
Rotation																	
ROL	reg, 1	CY ← MSB of reg, reg ← reg x 2 + CY MSB of reg ≠ CY: V ← 1 MSB of reg = CY: V ← 0	1 1	1 1	0 0	1 0	0 0	0	0 reg	W	2		x	x			
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 + CY MSB of (mem) ≠ CY: V ← 1 MSB of (mem) = CY: V ← 0	1 mod	1	0 0	1 0	0 0	0	0 mem	W	2-4		x	x			

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Rotation (cont)																	
ROL	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY, temp ← temp − 1	1 1	1 1	0 0	1 0	0 0	0	1 reg	W	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY, temp ← temp − 1	1 mod	1	0 0	1 0	0 0	0	1 mem	W	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY, temp ← temp − 1	1 1	1 1	0 0	0 0	0 0	0	0 reg	W	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY, temp ← temp − 1	1 mod	1	0 0	0 0	0 0	0	0 mem	W	3-5		x	u			
ROR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1 1	1 1	0 0	1 0	0 1	0	0 reg	W	2		x	x			
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY, MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1 mod	1	0 0	1 0	0 1	0	0 mem	W	2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY, temp ← temp − 1	1 1	1 1	0 0	1 0	0 1	0	1 reg	W	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY, temp ← temp − 1	1 mod	1	0 0	1 0	0 1	0	1 mem	W	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY, temp ← temp − 1	1 1	1 1	0 0	0 0	0 1	0	0 reg	W	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 mod	1	0 0	0 0	0 1	0	0 mem	W	3-5		x	u			

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Rotate																	
ROLC	reg, 1	tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy MSB of reg = CY: V ← 0 MSB of reg ≠ CY: V ← 1	1	1	0	1	0	0	0	W	2		x	x			
			1	1	0	1	0	reg									
	mem, 1	tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy MSB of (mem) = CY: V ← 0 MSB of (mem) ≠ CY: V ← 1	1	1	0	1	0	0	0	W	2-4		x	x			
			mod		0	1	0	mem									
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	2		x	u			
			1	1	0	1	0	reg									
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	2-4		x	u			
			mod		0	1	0	mem									
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	3		x	u			
			1	1	0	1	0	reg									
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy temp ← temp − 1	1	1	0	0	0	0	0	W	3-5		x	u			
			mod		0	1	0	mem									
RORC	reg, 1	tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	2		x	x			
			1	1	0	1	1	reg									
	mem, 1	tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	2-4		x	x			
			mod		0	1	1	mem									
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	2		x	u			
			1	1	0	1	1	reg									
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	2-4		x	u			
			mod		0	1	1	mem									
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	3		x	u			
			1	1	0	1	1	reg									
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	3-5		x	u			
			mod		0	1	1	mem									

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Subroutine Control Transfer																	
CALL	near-proc	(SP−1, SP−2) ← PC, SP ← SP−2, PC ← PC + disp	1	1	1	0	1	0	0	0	3						
	regptr16	(SP−1, SP−2) ← PC, SP ← SP−2, PC ← regptr16	1	1	1	1	1	1	1	1	2						
			1	1	0	1	0	reg									
	memptr16	(SP−1, SP−2) ← PC, SP ← SP−2, PC ← (memptr16)	1	1	1	1	1	1	1	1	2-4						
			mod		0	1	0	mem									
	far-proc	(SP−1, SP−2) ← PS, (SP−3, SP−4) ← PC, SP ← SP−4, PS ← seg, PC ← offset	1	0	0	1	1	0	1	0	5						
	memptr32	(SP−1, SP−2) ← PS, (SP−3, SP−4) ← PC, SP ← SP−4, PS ← (memptr32 + 2), PC ← (memptr32)	1	1	1	1	1	1	1	1	2-4						
			mod		0	1	1	mem									
RET		PC ← (SP + 1, SP), SP ← SP + 2	1	1	0	0	0	0	1	1	1						
	pop-value	PC ← (SP + 1, SP), SP ← SP + 2, SP ← SP + pop-value	1	1	0	0	0	0	1	0	3						
		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), SP ← SP + 4	1	1	0	0	1	0	1	1	1						
	pop-value	PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), SP ← SP + 4, SP ← SP + pop-value	1	1	0	0	1	0	1	0	3						
Stack Manipulation																	
PUSH	mem16	(SP−1, SP−2) ← (mem16), SP ← SP−2	1	1	1	1	1	1	1	1	2-4						
			mod		1	1	0	mem									
	reg16	(SP−1, SP−2) ← reg16, SP ← SP−2	0	1	0	1	0	reg			1						
	sreg	(SP−1, SP−2) ← sreg, SP ← SP−2	0	0	0	sreg	1	1	0		1						
	PSW	(SP−1, SP−2) ← PSW, SP ← SP−2	1	0	0	1	1	1	0	0	1						
	R	Push registers on the stack	0	1	1	0	0	0	0	0	1						
	imm	(SP−1, SP−2) ← imm, SP ← SP−2, When S = 1, sign extension	0	1	1	0	1	0	S	0	2-3						
POP	mem16	(mem16) ← (SP + 1, SP), SP ← SP + 2	1	0	0	0	1	1	1	1	2-4						
			mod		0	0	0	mem									
	reg16	reg16 ← (SP + 1, SP), SP ← SP + 2	0	1	0	1	1	reg			1						
	sreg	sreg ← (SP + 1, SP), sreg : SS, DS0, DS1 SP ← SP + 2	0	0	0	sreg	1	1	1		1						
	PSW	PSW ← (SP + 1, SP), SP ← SP + 2	1	0	0	1	1	1	0	1	1	R	R	R	R	R	R
	R	Pop registers from the stack	0	1	1	0	0	0	0	1	1						
PREPARE	imm16, imm8	Prepare new stack frame	1	1	0	0	1	0	0	0	4						
DISPOSE		Dispose of stack frame	1	1	0	0	1	0	0	1	1						

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Branch																	
BR	near-label	PC ← PC + disp	1	1	1	0	1	0	0	1	3						
	short-label	PC ← PC + ext-disp8	1	1	1	0	1	0	1	1	2						
	regptr16	PC ← regptr16	1	1	1	1	1	1	1	1	2						
			1	1	1	0	0	reg									
	memptr16	PC ← (memptr16)	1	1	1	1	1	1	1	1	2-4						
			mod		1	0	0	mem									
	far-label	PS ← seg, PC ← offset	1	1	1	0	1	0	1	0	5						
	memptr32	PS ← (memptr32 + 2), PC ← (memptr32)	1	1	1	1	1	1	1	1	2-4						
			mod		1	0	1	mem									
Conditional Branch																	
BV	short-label	if V = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	0	0	2						
BNV	short-label	if V = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	0	1	2						
BC, BL	short-label	if CY = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	1	0	2						
BNC, BNL	short-label	if CY = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	1	1	2						
BE, BZ	short-label	if Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	0	0	2						
BNE, BNZ	short-label	if Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	0	1	2						
BNH	short-label	if CY OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	1	0	2						
BH	short-label	if CY OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	1	1	2						
BN	short-label	if S = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	0	0	2						
BP	short-label	if S = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	0	1	2						
BPE	short-label	if P = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	1	0	2						
BPO	short-label	if P = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	1	1	2						
BLT	short-label	if S XOR V = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	0	0	2						
BGE	short-label	if S XOR V = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	0	1	2						
BLE	short-label	if (S XOR V) OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	1	0	2						
BGT	short-label	if (S XOR V) OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	1	1	2						
DBNZNE	short-label	CW ← CW − 1 if Z = 0 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	0	2						
DBNZE	short-label	CW ← CW − 1 if Z = 1 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	1	2						
DBNZ	short-label	CW ← CW − 1 if CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	1	0	2						
BCWZ	short-label	if CW = 0, PC ← PC + ext-disp8	1	1	1	0	0	0	1	1	2						
BTCLR	sfr, imm3, short-label	if bit no. imm3 of (sfr) = 1, PC ← PC + ext−disp8, bit no. imm3 or (sfr) ← 0	0	0	0	0	1	1	1	1	5						
			1	0	0	1	1	1	0	0							

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Interrupt																	
BRK	3	(SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (15, 14), PC ← (13, 12)	1	1	0	0	1	1	0	0	1						
	imm8 (≠3)	(SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PC ← (n x 4 + 1, n x 4), PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1	1	0	0	1	1	0	1	2						
BRKV		When V = 1 (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (19, 18), PC ← (17, 16)	1	1	0	0	1	1	1	0	1						
RETI		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1	1	0	0	1	1	1	1	1	R	R	R	R	R	R
RETRBI		PC ← Save PC, PSW ← Save PSW	0	0	0	0	1	1	1	1	2	R	R	R	R	R	R
			1	0	0	1	0	0	0	1							
FINT		Indicates that interrupt service routine to the interrupt controller built in the CPU has been completed	0	0	0	0	1	1	1	1	2						
			1	0	0	1	0	0	1	0							
CHKIND	reg16, mem32	When (mem32) > reg16 or (mem32 + 2) < reg16 (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (23, 22), PC ← (21, 20)	0	1	1	0	0	0	1	0	2-4						
			mod			reg			mem								
CPU Control																	
HALT		CPU Halt	1	1	1	1	0	1	0	0		1					
STOP		CPU Halt	0	0	0	0	1	1	1	1		1					
			1	0	1	1	1	1	1	0							
BUSLOCK		Bus Lock Prefix	1	1	1	1	0	0	0	0		1					
FP01 (Note 1)	fp-op	No Operation	1	1	0	1	1	X	X	X	2						
			1	1	Y	Y	Y	Z	Z	Z							
	fp-op, mem	data bus ← (mem)	1	1	0	1	1	X	X	X	2-4						
			mod		Y	Y	Y		mem								
FP02 (Note 1)	fp-op	No Operation	0	1	1	0	0	1	1	X	2						
			1	1	Y	Y	Y	Z	Z	Z							
	fp-op, mem	data bus ← (mem)	0	1	1	0	0	1	1	X	2-4						
			mod		Y	Y	Y		mem								

Notes:
(1) Does not execute but does generate an interrupt.

4d

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
CPU Control (cont)																	
POLL		Poll and Wait	1	0	0	1	1	0	1	1	1						
NOP		No Operation	1	0	0	1	0	0	0	0	1						
DI		IE ← 0	1	1	1	1	1	0	1	0	1						
EI		IE ← 1	1	1	1	1	1	0	1	1	1						
DS0; DS1; PS; SS		Segment Override Prefix	0	0	1	sreg		1	1	0	1						
Register Bank Switching																	
MOVSPA			0	0	0	0	1	1	1	1	2						
			0	0	1	0	0	1	0	1							
BRKCS	reg16		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	1	0	1							
MOVSPB	reg16		0	0	0	0	1	1	1	1	3						
			1	0	0	1	0	1	0	1							
			1	1	1	1	1		reg								
TSKSW	reg16		0	0	0	0	1	1	1	1	3	x	x	x	x	x	x
			1	0	0	1	0	1	0	0							
			1	1	1	1	1		reg								

NEC
NEC Electronics Inc.

μPD70335 (V35 Plus) 16-Bit Microcomputer: Advanced, High-Speed DMA, Single-Chip, CMOS

Description

The μPD70335 (V35 Plus) is a high-performance, 16-bit single-chip microcomputer with a 16-bit external data bus. The μPD70335 is fully software compatible with the μPD70108/116 (V20®/V30®) as well as the μPD70320/330 (V25™/V35™). The V35 Plus demonstrates numerous enhancements over the standard V35; however, it maintains strict pin compatibility with its predecessor, the V35.

The V35 Plus offers improved DMA transfer rates (over 5M bytes per second), additional serial channel status flags, improved memory access timing, and enhanced software control of register bank context switching.

The μPD70335 has the same complement of internal peripherals as the V35, and maintains compatibility with existing drivers; however, some modification of the DMA drivers may be necessary. The μPD70335 does not offer on-chip ROM or EPROM.

Features

- 16-bit CPU and internal data paths
- 16-bit non-multiplexed external data path
- Direct RAS/CAS DRAM interface
- Functional and pin compatibility with the V35
- Software compatible with μPD8086
- New and enhanced V-Series instructions
- Minimum instruction cycle 200 ns (at 10 MHz)
- 6-byte prefetch queue
- Two-channel high-speed DMA controller
- Internal 256 bytes RAM memory
- One 1M-byte memory address space
- Eight internal memory-mapped register banks
- Four multifunction I/O ports
  - 8-bit analog comparator port
  - 20 bidirectional port lines
  - 4 input-only port lines
- Two independent full-duplex serial channels
- Priority interrupt controller
  - Standard vectored service
  - Register bank switching
  - Macroservice
- Pseudo-SRAM and DRAM refresh controller
- Two 16-bit timers
- On-chip time base counter
- Programmable wait state generator
- Two standby modes: STOP and HALT

Ordering Information

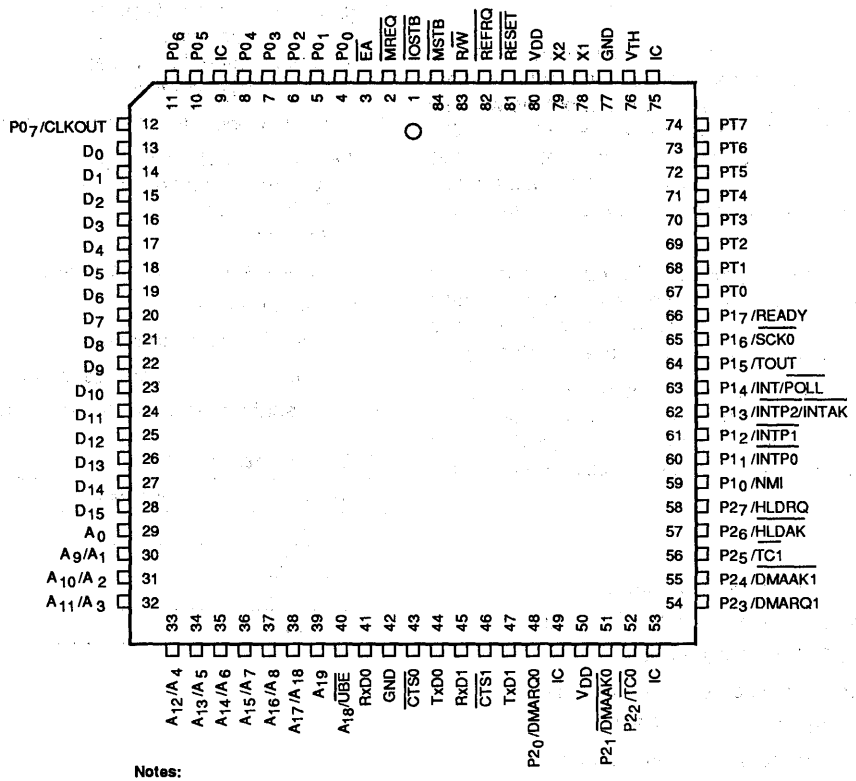
Part Number	Clock (MHz)	Package
μPD70335L-8	8	84-pin PLCC
L-10	10	
GJ-8	8	94-pin plastic QFP
GJ-10	10	

4e

V20 and V30 are registered trademarks of NEC Corporation.
V25 and V35 are trademarks of NEC Corporation.

Pin Configuration

84-Pin PLCC



Notes:

(1) Pin functions are identical to μPD70330.

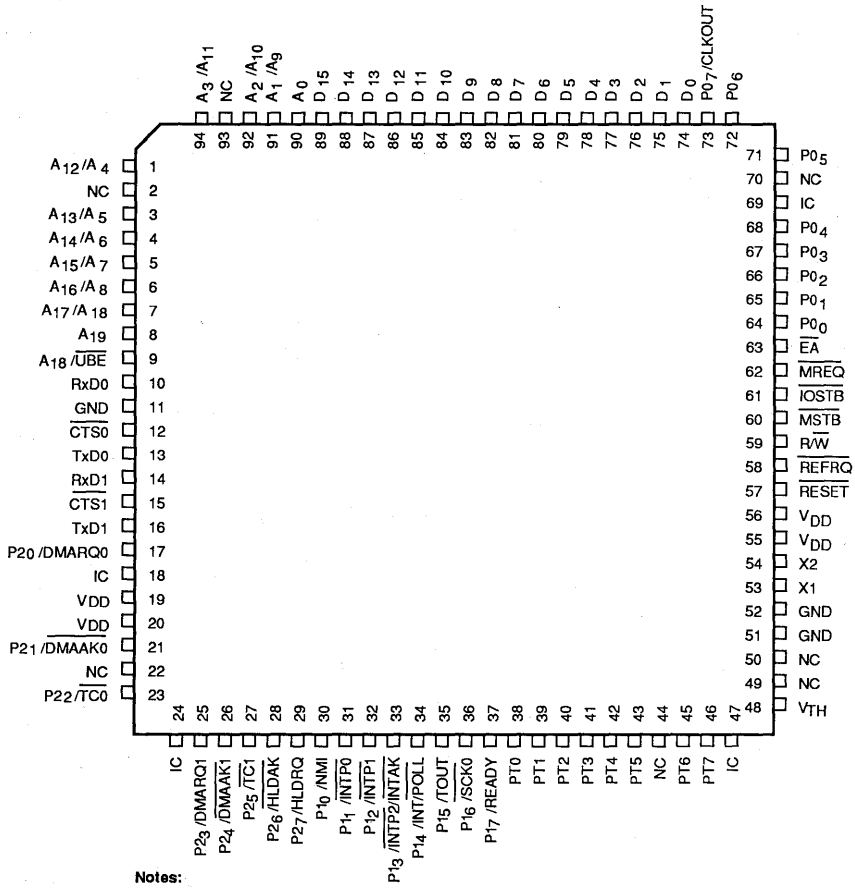
(2) IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

(3) $\overline{EA}$ must be tied low because μPD70335 does not support internal ROM or EPROM.

(4) Pin 9 should be tied to GND through a pull-down resistor.

83SL-6827B

94-Pin Plastic QFP



Notes:

(1) Pin functions are identical to μPD70330.

(2) IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

(3) $\overline{EA}$ must be tied low because μPD70335 does not support internal ROM or EPROM.

(4) Pin 69 should be tied to GND through a pull-down resistor.

83SL-6828B

4e

Pin Identification

Symbol	Function
A_0-A_{19}	Address bus outputs
CLKOUT	System clock output
$\overline{CTS0}$	Clear-to-send input, serial channel 0
$\overline{CTS1}$	Clear-to-send input, serial channel 1
D_0-D_{15}	Bidirectional data bus
$\overline{DMAAK0}$	DMA acknowledge output, DMA controller channel 0
$\overline{DMAAK1}$	DMA acknowledge output, DMA controller channel 1
DMARQ0	DMA request input, DMA controller channel 0
DMARQ1	DMA request input, DMA controller channel 1
$\overline{EA}$	External memory access; fixed low for V35 Plus
$\overline{HLDAK}$	Hold acknowledge output
HLDRQ	Hold request input
INT	Interrupt request input
$\overline{INTAK}$	Interrupt acknowledge output
$\overline{INTP0}$	Interrupt request 0 input
$\overline{INTP1}$	Interrupt request 1 input
$\overline{INTP2}$	Interrupt request 2 input
$\overline{IOSTB}$	I/O read or write strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory strobe output
NMI	Nonmaskable interrupt request
$P0_0$-$P0_7$	I/O port 0
$P1_0$-$P1_7$	I/O port 1
$P2_0$-$P2_7$	I/O port 2
$\overline{POLL}$	Input on $\overline{POLL}$ synchronizes the CPU and external devices
PT0-PT7	Comparator port input lines
READY	Ready signal input controls insertion of wait states
$\overline{REFRQ}$	DRAM refresh request output
$\overline{RESET}$	Reset signal input
$R/\overline{W}$	Read/write strobe output
RxD0	Receive data input, serial channel 0
RxD1	Receive data input, serial channel 1
$\overline{SCK0}$	Serial clock output
$\overline{TC0}$	Terminal count output; DMA completion, channel 0
$\overline{TC1}$	Terminal count output; DMA completion, channel 1
TOUT	Timer output
TxD0	Transmit data output, serial channel 0
TxD1	Transmit data output, serial channel 1
$\overline{UBE}$	Upper byte enable
X1, X2	Connections to external frequency control source (crystal, ceramic resonator, or clock)

Symbol	Function
V_{DD}	+5-volt power source input (two pins)
V_{TH}	Threshold voltage input to comparator circuits
GND	Ground reference (two pins)
IC	Internal connection; must be tied to V_{DD} externally through a pullup resistor

PIN FUNCTIONS

A_0-A_{19} (Address Bus)

To support dynamic RAMs, the 20-bit address is multiplexed on 11 lines. When $\overline{MREQ}$ is asserted, A_9-A_{17} are valid. When $\overline{MSTB}$ or $\overline{IOSTB}$ is asserted, A_1-A_8 and A_{18} are valid. A_{18} is also multiplexed with $\overline{UBE}$ and is valid when $\overline{MREQ}$ is asserted. Therefore A_{18} is active throughout the bus cycle. A_{19} and A_0 are not multiplexed but have dedicated pins and are valid throughout the bus cycle.

CLKOUT (Clock Out)

The system clock (CLK) is distributed from the internal clock generator to the CPU and output to peripheral hardware at the CLKOUT pin.

$\overline{CTS0}$ (Clear-to-Send 0)

This is the CTS pin of the channel 0 serial interface. In asynchronous mode, a low-level input on $\overline{CTS0}$ enables transmit operation. In I/O interface mode, $\overline{CTS0}$ is the receive clock pin.

$\overline{CTS1}$ (Clear-to-Send 1)

This is the CTS pin of the channel 1 serial interface. In asynchronous mode, a low-level input on $\overline{CTS1}$ enables transmit operation.

D_0-D_{15} (Data Bus)

D_0-D_{15} is the 16-bit data bus.

$\overline{DMAAK0}$ and $\overline{DMAAK1}$ (DMA Acknowledge)

These are the DMA acknowledge outputs of the DMA controller, channels 0 and 1. Signals are not output during DMA memory-to-memory transfer operations (burst mode, single-step mode).

$\overline{DMARQ0}$ and $\overline{DMARQ1}$ (DMA Request)

These are the DMA request inputs of the DMA controller, channels 0 and 1.

$\overline{EA}$ (External Access)

This pin must be externally fixed low. Since the μPD70335 has no internal ROM, this will force execution of program code from external memory instead of internal ROM.

$\overline{HLDAK}$ (Hold Acknowledge)

The $\overline{HLDAK}$ output signal indicates that the hold request (HLDRQ) has been accepted. When $\overline{HLDAK}$ is active (low), the following lines go to the high-impedance state with internal 4700-Ω pullup resistors: A_0-A_{19}, D_0-D_7, $\overline{IOSTB}$, $\overline{MREQ}$, $\overline{MSTB}$, $\overline{REFRQ}$, and R/$\overline{W}$.

HLDRQ (Hold Request)

The HLDRQ input from an external device requests that the μPD70335 relinquish the address, data, and control buses to an external bus master.

INT (Interrupt)

The INT input is a vectored interrupt request from an external device that can be masked by software. The active high level is detected in the last clock cycle of an instruction. The external device confirms that the INT interrupt request has been accepted by the $\overline{INTAK}$ signal output from the CPU.

The INT signal must be held high until the first $\overline{INTAK}$ signal is output. Together with $\overline{INTAK}$, INT is used for operation with an interrupt controller such as μPD71059.

$\overline{INTAK}$ (Interrupt Acknowledge)

The $\overline{INTAK}$ output is the acknowledge signal for the software-maskable interrupt request INT. The $\overline{INTAK}$ signal goes low when the CPU accepts INT. The external device inputs the interrupt vector to the CPU via data bus D_0-D_7 in synchronization with $\overline{INTAK}$.

$\overline{INTP0}$, $\overline{INTP1}$, $\overline{INTP2}$ (Interrupt from Peripheral 0, 1, 2)

The $\overline{INTPn}$ inputs (n = 0, 1, 2) are external interrupt requests that can be masked by software. The $\overline{INTPn}$ input is detected at the effective edge specified by external interrupt mode register INTM.

The $\overline{INTPn}$ inputs can be used to release the HALT mode.

$\overline{IOSTB}$ (I/O Strobe)

A low-level output on $\overline{IOSTB}$ indicates that the I/O bus cycle has been initiated and that the I/O address output on A_0-A_{15} is valid.

$\overline{MREQ}$ (Memory Request)

A low-level output on $\overline{MREQ}$ indicates that the memory or I/O bus cycle has started and that address bits A_0, A_9-A_{17}, A_{18} and A_{19} are valid.

$\overline{MSTB}$ (Memory Strobe)

Together with $\overline{MREQ}$ and R/$\overline{W}$, $\overline{MSTB}$ controls memory-accessing operations. $\overline{MSTB}$ should be used either to enable data buffers or as a data strobe. During memory write, a low-level output on $\overline{MSTB}$ indicates that data on the data bus is valid and that multiplexed address bits A_1-A_8, A_{18} and $\overline{UBE}$ are valid.

NMI (Nonmaskable Interrupt)

The NMI input is an interrupt request that cannot be masked by software. The NMI is always accepted by the CPU; therefore, it has priority over any other interrupt.

The NMI input is detected at the effective edge specified by external interrupt mode register INTM. Sampled in each clock cycle, NMI is accepted when the active level lasts for several clock cycles. When the NMI is accepted, a number 2 vector interrupt is generated after completion of the instruction currently being executed.

The NMI input is also used to release the CPU standby mode.

$P0_0$-$P0_7$ (Port 0)

Port 0 is an 8-bit bidirectional I/O port.

$P1_0$-$P1_7$ (Port 1)

Lines $P1_4$-$P1_7$ are individually programmable as an input, output, or control function. The status of $P1_0$- $P1_3$ can be read but these lines are always control functions.

$P2_0$-$P2_7$ (Port 2)

$P2_0$-$P2_7$ are the lines of port 2, an 8-bit bidirectional I/O port. These lines can also be used as control signals for the on-chip DMA controllers.

$\overline{POLL}$ (Poll)

The $\overline{POLL}$ input is checked by the POLL instruction. If the level is low, execution of the next instruction is initiated. If the level is high, the $\overline{POLL}$ input is checked every five clock cycles until the level becomes low. The $\overline{POLL}$ functions are used to synchronize the CPU program and the operation of external devices.

Note: $\overline{POLL}$ is effective when $P1_4$ is specified for the input port mode; otherwise, $\overline{POLL}$ is assumed to be at low level when the POLL instruction is executed.

PT0-PT7 (Port with Comparator)

The threshold port (PT) comprises 8 independent input bits, each of which is compared with a threshold voltage programmable to one of 16 voltage steps.

READY (Ready)

After READY is de-asserted low, the CPU will synchronize and insert wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than normal execution allows. Use of the READY pin is controlled by the WTC register.

$\overline{REFRQ}$ (Refresh Request)

This output pulse can refresh nonstatic RAM. It can be programmed to meet system specifications and is internally synchronized so that refresh cycles do not interfere with normal CPU operation. $\overline{REFRQ}$ also signals that A_0-A_8 contain a valid row address.

$\overline{RESET}$ (Reset)

This input signal is asynchronous. A low on $\overline{RESET}$ for the specified duration resets the CPU and all on-chip peripherals regardless of clock operation. The reset operation has priority over all other operations.

The reset signal is used for normal initialization/startup and also for releasing the STOP or HALT mode. After the reset signal returns high, program execution begins from address FFFF0H.

R/$\overline{W}$ (Read/Write Strobe)

When an external bus cycle is initiated, the R/$\overline{W}$ signal output to external hardware indicates a read (high-level) or write (low-level) cycle. It can also control the direction of bidirectional buffers.

RxD0, RxD1 (Receive Data 0, 1)

These pins input data to serial channels 0 and 1.

In the asynchronous mode, when receive operation is enabled, a low level on the RxD0 or RxD1 input pin is recognized as the start bit and receive operation is initiated.

In the I/O interface mode (channel 0 only), receive data is input to the serial register at the rising edge of the receive clock.

$\overline{SCK0}$ (Serial Clock)

The $\overline{SCK0}$ output is the transmit clock of serial channel 0.

$\overline{TC0}$, $\overline{TC1}$ (Terminal Count 0, 1)

The $\overline{TC0}$ and $\overline{TC1}$ outputs go low when the terminal count of DMA service channels 0 and 1, respectively, reach zero, indicating DMA completion.

TOUT (Timer Output)

The TOUT signal is a square-wave output from the internal timer unit zero.

TxD0, TxD1 (Transmit Data 0, 1)

These pins output data from serial channels 0 and 1.

In the asynchronous mode, the transmit signal is in a frame format that consists of a start bit, 7 or 8 data bits (least significant bit first), parity bit, and stop bit. The TxD0 and TxD1 pins become mark state (high level) when transmit operation is disabled or when the serial transmitter is idle.

In the I/O interface mode (channel 0 only), the frame has 8 data bits and the most significant bit is transmitted first.

$\overline{UBE}$ (Upper Byte Enable)

$\overline{UBE}$ is a high-order memory bank selection signal output. $\overline{UBE}$ and A_0 determine which bytes of the data bus will be used. UBE is used with A_0 to select the even/odd banks as follows.

Operand	$\overline{UBE}$	A_0	Number of Bus Cycles
Even address word	0	0	1
Odd address word	0	1	2
	1	0	
Even address byte	1	0	1
Odd address byte	0	1	1

X1, X2 (Clock Control)

The frequency of the internal clock generator is controlled by an external crystal or ceramic resonator connected across pins X1 and X2. The crystal frequency is the same as the clock generator frequency f_X. By programming the PRC register, the system clock frequency f_{CLK} is selected as f_X divided by 2, 4, or 8.

As an alternative to the crystal or ceramic resonator, the positive and negative phases of an external clock (with frequency f_X) can be connected to pins X1 and X2.

V_{DD} (Power Supply)

+5-volt power source (two pins).

V_{TH} (Threshold Voltage)

Comparator port PT0-PT7 uses threshold voltage V_{TH} to determine the analog reference points. The actual threshold each comparator input is tested against is programmable to V_{TH} x n/16 where n = 1 to 16.

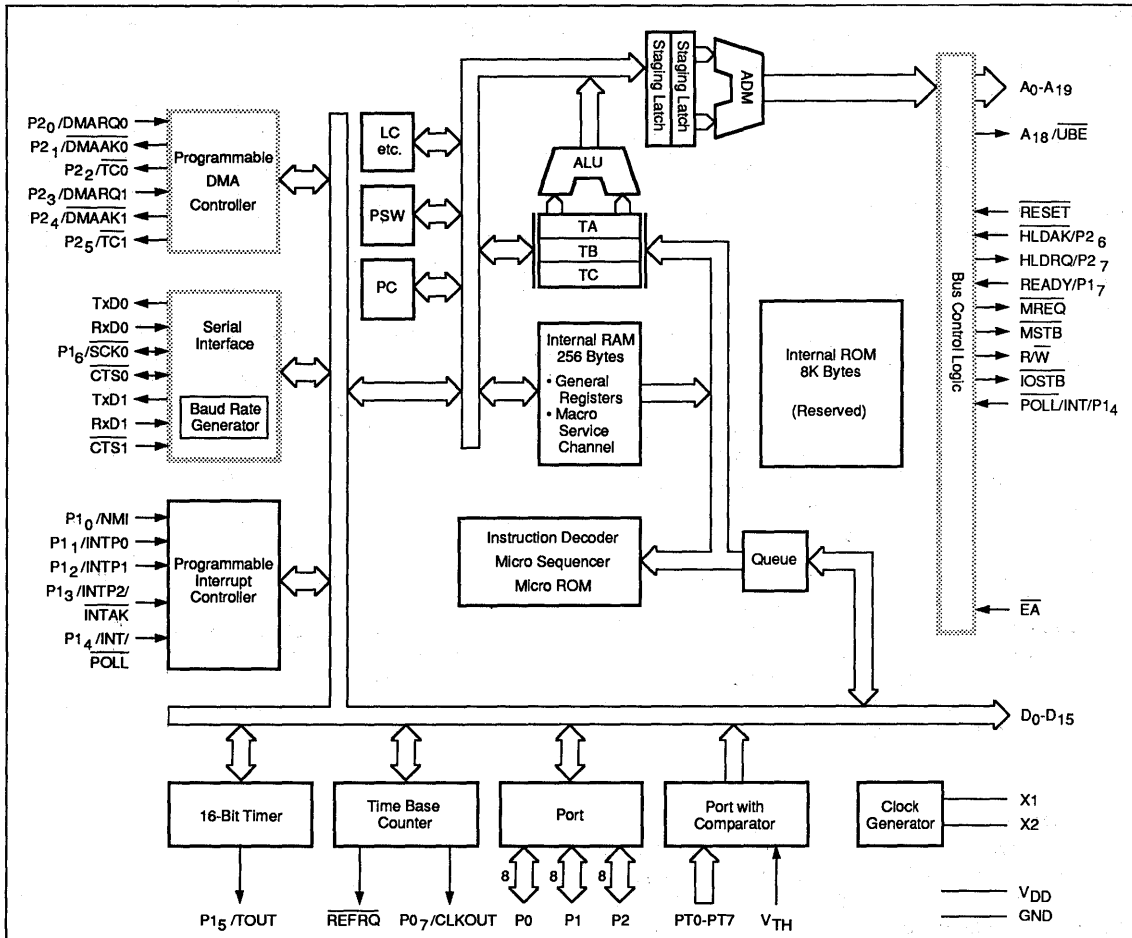
GND (Ground)

Ground reference (multiple pins).

IC (Internal Connection)

Internal connection; must be tied to V_{DD} externally through a 10-kΩ to 20-kΩ resistor.

μPD70335 Block Diagram



Notes:

(1) The μPD70335 (V35 Plus) is not a masked ROM product. Internal ROM is reserved and not accessible.

(2) Shaded blocks are modified from the standard V35.

83YL-6989B

4e

FUNCTIONAL DESCRIPTION

Architectural Enhancements

The following features enable the μPD70335 to perform high-speed execution of instructions.

- Dual data bus
- 16-/32-bit temporary registers/shifters (TA, TB, TA + TB)
- 16-bit loop counter (LC)
- Program counter (PC) and prefetch pointer (PFP)
- Internal ROM pass bus

Dual Data Bus. The μPD70335 has two internal 16-bit data buses: the main data bus and a subdata bus. This reduces the processing time required for addition/subtraction and logical comparison instructions by one-third over single-bus systems. The dual data bus method allows two operands to be fetched simultaneously from general-purpose registers and transferred to the ALU.

16-/32-Bit Temporary Registers/Shifters. The 16-bit temporary registers/shifters (TA, TB) allow high-speed execution of multiplication/division and shift/rotation instructions. By using the temporary registers/shifters,the μPD70335 can execute multiplication/division instructions about four times faster than with the microprogramming method.

Loop Counter (LC). The dedicated hardware loop counter counts the number of loops for string operations and the number of shifts performed for multiple bit shift/rotation instructions. The loop counter works with internal dedicated shifters to speed the processing of multiplication/division instructions.

Program Counter and Prefetch Pointer (PC and PFP). The hardware PC addresses the memory location of the instruction to be executed next. The hardware PFP addresses the program memory location to be accessed next. Several clocks are saved for branch, call, return, and break instructions compared with processors having only one instruction pointer.

Register Set

The μPD70335 CPU has a general-purpose register set compatible with the μPD70108/70116, the μPD70320/70322, and μPD70330/70332 microprocessors. Like the μPD70320/70322 and μPD70330/70332, it also has a set of special function registers for controlling the on-board peripherals. All registers reside in the CPU's memory space. They are grouped in a 512-byte block called the internal data area (IDA). The 256-byte internal RAM is also in the IDA. The addresses of the register are given as offsets into the IDA. The start address of the IDA is set by the Internal Data Area Base register (IDB), and may be programmed to any 4K boundary in the memory address space.

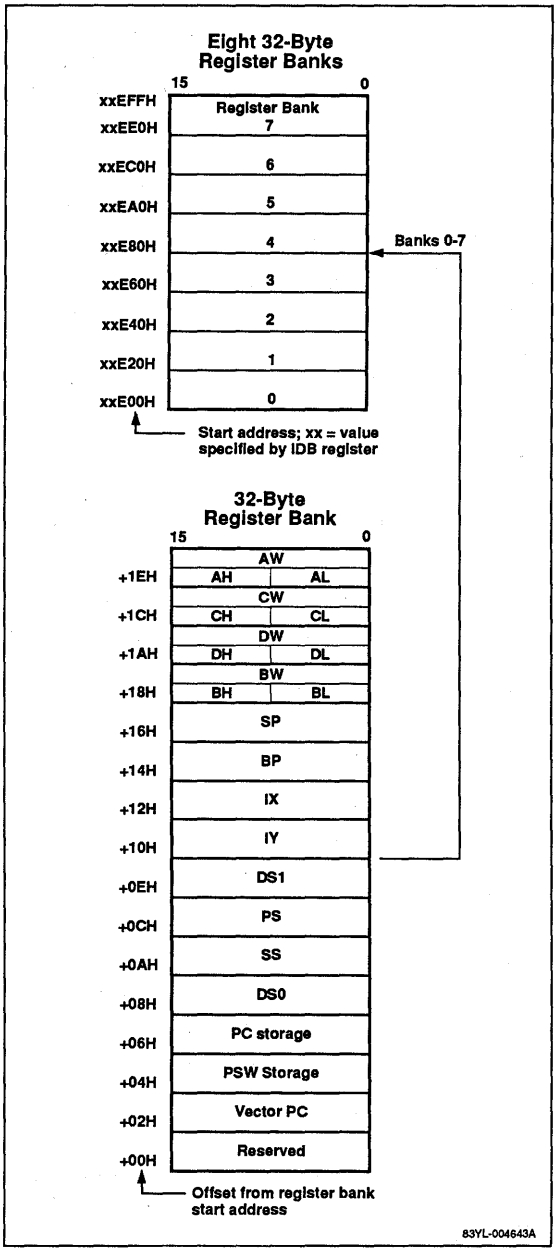
Register Banks. Because the general-purpose register set is in internal RAM, it is possible to have multiple banks of registers. The μPD70335 CPU supports up to 8 register banks. A bit field in the PSW selects which bank is currently being used. Each bank contains the entire CPU register set plus additional information needed for context switching. Register banks may be switched using special instructions (TSKSW, BRKCS, MOVSPA, MOVSPB), or may switch in response to an interrupt. This provides fast context switching and fast interrupt handling. During and after RESET, register bank 7 is selected.

Figure 1 shows the configuration of a register bank and how the banks are mapped to internal RAM. The Vector PC field contains the value that will be loaded into the PC when a register bank switch occurs. The PC Save and PSW Save fields contain the values of the PC and the PSW just before the banks are switched. The PSW is left unmodified after a bank switch; the PSW Save field is used to restore the PSW to its previous state upon termination of the context switch.

General-Purpose Registers (AW, BW, CW, DW). These four 16-bit general-purpose registers can also serve as independent 8-bit registers (AH, AL, BH, BL, CH, CL, DH, DL). The instructions below use general-purpose registers for default:

AW	Word multiplication/division, word I/O, data conversion
AL	Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
AH	Byte multiplication/division
BW	Translation
CW	Loop control branch, repeat prefix
CL	Shift instructions, rotation instructions, BCD operations
DW	Word multiplication/division, indirect addressing I/O

Figure 1. Register Bank Configuration



Pointers (SP, BP) and Index Registers (IX, IY). These registers are used as 16-bit base pointers or index registers in based addressing, indexed addressing, and based-indexed addressing. The registers are used as default registers under the following conditions:

SP	Stack operations
IX	Block transfer (source), BCD string operations
IY	Block transfer (destination), BCD string operations

Segment Registers. The segment registers divide the 1M-byte address space into 64K-byte blocks. Each segment register functions as a base address to a block; the effective address is an offset from that base. Physical addresses are generated by shifting the associated segment register left four binary digits and then adding the effective address. The segment registers are:

Segment Register	Default Offset
PS (Program segment)	PC
SS (Stack segment)	SP, Effective address
DS0 (Data segment-0)	IX, Effective address
DS1 (Data segment-1)	IY, Effective address

During RESET, PS is set to FFFFH; DS0, DS1 and SS are set to 0000H.

Program Counter (PC). The PC is a 16-bit binary counter that contains the offset address from the program segment of the next instruction to be executed. It is incremented every time an instruction is received from the queue. It is loaded with a new location whenever a branch, call, return, break, or interrupt is executed. During RESET, PC is set to 0000H.

Program Status Word (PSW). The PSW contains the following status and control flags.

PSW

15							8
1	RB2	RB1	RB0	V	DIR	IE	BRK

7							0
S	Z	F1	AC	F0	P	BRKI	CY

Status Flags	
V	Overflow bit
S	Sign
Z	Zero
AC	Auxiliary carry
P	Parity
CY	Carry

Control Flags	
DIR	Direction of string processing
IE	Interrupt enable
BRK	Break (after every instruction)
RBn	Current register bank flags
BRKI	I/O trap enable
F0, F1	General-purpose user flags

4e

The eight low-order bits of the PSW can be stored in the AH register and restored by a MOV instruction. The only way to alter the RBn bits via software is to execute an RETRBI or RETI instruction. During RESET, PSW is set to F002H. The F0 and F1 flags may be accessed as bits in the FLAG special functioning register.

Functional Comparison

The μPD70335 (V35 Plus) is built around the same core and contains the same peripherals as the μPD70325 (V25 Plus) as well as the μPD70330 (V35). The primary difference between the V35 and V25 is confined to the external bus interface and bus control logic. While V25 and V25 Plus are designed with an 8-bit external interface, V35 and V35 Plus provide the full 16-bit external data path.

The μPD70335 provides a direct DRAM style bus interface. This interface is obtained by multiplexing the 20 address lines in row/column fashion and also providing a non-multiplexed 16-bit external data bus. The resulting nominal bus cycle is three CLOCKOUT states. During the first bus state, the address lines output the high 9 bits of the physical address: A_9 to A_{17}.

During the second bus state, the address lines output the low address bits: A_1 to A_8. Address lines A_0 and A_{19} are not multiplexed and are valid during the entire bus cycle. The final address line (A18) is multiplexed with the Upper Byte Enable signal (UBE) and is valid as an address during bus state one. During 16-bit transfers to odd addresses (UBE = 0 and A_0 = 1), two bus cycles are performed; each cycle transfers eight bits.

Typically, the MREQ signal is used to generate the DRAM RAS control signal, and the MSTB signal is used to generate the CAS signal. Like the V35, the V35 Plus provides a refresh output from the internal refresh control unit, which is typically gated into the DRAM RAS signal.

As a result of this memory access scheme, the clock cycle counts for instruction execution on the V35 Plus are different from the V25 Plus.

Another V35 Plus difference is the operation of the READY input pin. This pin is sampled in the middle of the second bus cycle (BAW1) on the V25 Plus, whereas the V35 samples one clock period later in the middle of BAW2.

Other than these bus controller differences, the V35 Plus is identical to the V25 Plus in its operation. All internal peripherals are programmed and operate in the same manner as those of the V25 Plus. The instruction sets of the two processors are identical, and internally both processors operate on 16-bit data paths.

INSTRUCTIONS

The μPD70335 instruction set is fully upward compatible with the V20 native mode instruction set. The V20 instruction set is a superset of the μPD8086/8088 instruction set with different execution times and mnemonics.

The μPD70335 does not support the V20 8080 emulation mode. All of the instructions pertaining to this have been deleted from the μPD70335 instruction set.

Enhanced Instructions

In addition to the μPD8086/88 instructions, the μPD70335 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes eight general registers onto stack
POP R	Pops eight general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Unique Instructions

The μPD70335 has the following unique instructions.

Instruction	Function
INS	Inserts bit field
EXT	Extracts bit field
ADD4S	Performs packed BCD string addition
SUB4S	Performs packed BCD string subtraction
CMP4S	Performs packed BCD string comparison
ROL4	Rotates BCD digit left
ROR4	Rotates BCD digit right
TEST1	Tests bit
SET1	Sets bit
CLR1	Clears bit
NOT1	Complements bit
REPC	Repeat while carry set
REPNC	Repeat while carry cleared

Variable Length Bit Field Operation Instructions

Bit fields are a variable length data structure that can range in length from 1 to 16 bits. The μPD70335 supports two separate operations on bit fields: insertion (INS) and extraction (EXT). There are no restrictions on the position of the bit field in memory. Separate segment, byte offset, and bit offset registers are used for insertion and extraction. Following the execution of these instructions, both the byte offset and bit offset are left pointing to the start of the next bit field, ready for the next operation. Bit field operation instructions are powerful and flexible and are therefore highly effective for graphics, high-level languages, and packing/unpacking applications.

Bit field insertion copies the bit field of specified length from the AW register to the bit field addressed by DS1:IY:reg8 (8-bit general-purpose register). The bit field length can be located in any byte register or supplied as immediate data. Following execution, both the IY and reg8 are updated to point to the start of the next bit field.

Bit field extraction copies the bit field of specified length from the bit field addressed by DS0:IX:reg8 to the AW register. If the length of the bit field is less than 16 bits, the bit field is right justified with a zero fill. The bit field length can be located in any byte register or supplied as immediate data. Following execution, both IX and reg8 are updated to point to the start of the next bit field.

Figures 2 and 3 show bit field insertion and bit field extraction.

4e

Packed BCD Instructions

Packed BCD instructions process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte format operands (ROR4, ROL4). Packed BCD strings may be 1 to 254 digits in length. The two BCD rotation instructions perform rotation of a single BCD digit in the lower half of the AL register through the register or the memory operand.

Figure 2. Bit Field Insertion

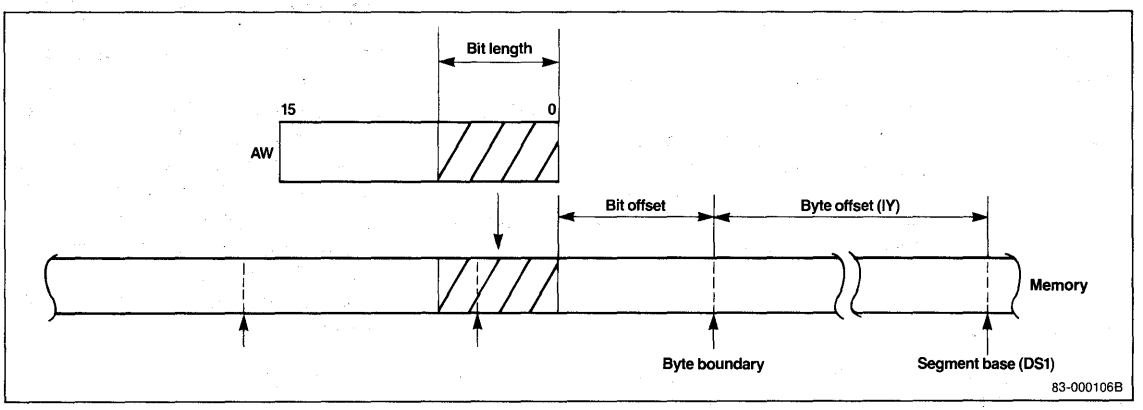
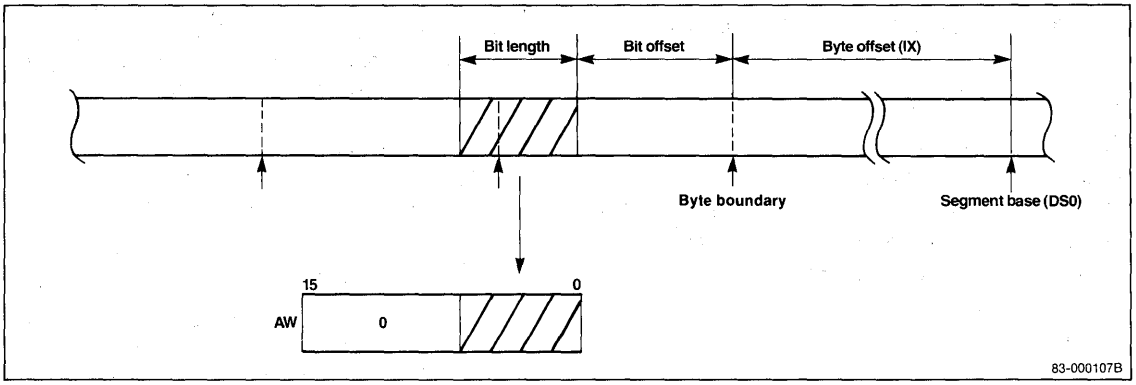


Figure 3. Bit Field Extraction



Bit Manipulation Instructions

The μPD70335 has five unique bit manipulation instructions. The ability to test, set, clear, or complement a single bit in a register or memory operand increases code readability as well as performance over the logical operations traditionally used to manipulate bit data. This feature further enhances control over on-chip peripherals.

Additional Instructions

Besides the V20 instruction set, the μPD70335 has the following four additional instructions.

Instruction	Function
BTCLR sfr, imm3 short label	Bit test and if true, clear and branch; otherwise, no operation
STOP (no operand)	Power down instruction, stops oscillator
RETRBI (no operand)	Return from register bank context switch interrupt
FINT (no operand)	Finished interrupt. After completion of a hardware interrupt request, this instruction must be used to reset the current priority bit in the in-service priority register (ISPR).*

*Do not use with NMI or INTR interrupt service routines.

Repeat Prefixes

Two new repeat prefixes (REPC, REPNC) allow con ditional block transfer instructions to use the state of the CY flag as the termination condition. This allows inequalities to be used when working on ordered data, thus increasing performance when searching and sorting algorithms.

Bank Switch Instructions

The V35 Plus has the following four instructions that allow the effective use of the register banks for software interrupts and multitasking. Also, see figures 7 and 9.

Instruction	Function
BRKCS reg 16	Performs a high-speed software interrupt with context switch to the register bank indicated by the lower 3-bits of reg 16. This operation is identical to the interrupt operation shown in figure 9.
TSKSW reg 16	Performs a high-speed task switch to the register bank indicated by the lower 3-bits of reg 16. The PC and PSW are saved in the old banks. PC and PSW save registers and the new PC and PSW values are retrieved from the new register bank's save areas. See figure 10.
MOVSPA	Transfers both the SS and SP of the old register bank to the new register bank after the bank has been switched by an interrupt or BRKCS instruction.
MOVSPB	Transfers the SS and the SP of the current register bank before the switch to the SS and SP of the new register bank indicated by the lower 3-bits of reg 16.

INTERRUPT STRUCTURE

The μPD70335 can service interrupts generated both by hardware and by software. Software interrupts are serviced through vectored interrupt processing. See table 1 for the various types of software interrupts.

Table 1. Software Interrupts

Interrupt	Description
Divide error	The CPU will trap if a divide error occurs as the result of a DIV or DIVU instruction.
Single step	The interrupt is generated after every instruction if the BRK bit in the PSW is set.
Overflow	By using the BRKV instruction, an interrupt can be generated as the result of an overflow.
Interrupt instructions	The BRK 3 and BRK imm8 instructions can generate interrupts.
Array bounds	The CHKIND instruction will generate an interrupt if specified array bounds have been exceeded.
Escape trap	The CPU will trap on an FP01, 2 instruction to allow software to emulate the floating point processor.
I/O trap	If the I/O trap bit in the PSW is cleared, a trap will be generated on every IN or OUT instruction. Software can then provide an updated peripheral address. This feature allows software interchangeability between different systems.

When executing software written for another system, it is better to implement I/O with on-chip peripherals to reduce external hardware requirements. However, since μPD70335 internal peripherals are memory mapped, software conversion could be difficult. The I/O trap feature allows easy conversion from external peripherals to on-chip peripherals.

Interrupt Vectors

The starting address of the interrupt processing routines may be obtained from table 2. The table begins at physical address 00H, which is outside the internal ROM space. Therefore, external memory is required to service these routines. By servicing interrupts via the macro service function or context switching, this requirement can be eliminated.

Each interrupt vector is four bytes wide. To service a vectored interrupt, the lower addressed word is transferred to the PC and the upper word to the PS. See figure 4.

Figure 4. Interrupt Vector

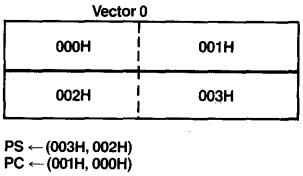


Table 2. Interrupt Vectors

Address	Vector No.	Assigned Use
00	0	Divide error
04	1	Break flag
08	2	NMI
0C	3	BRK3 instruction
10	4	BRKV instruction
14	5	CHKIND instruction
18	6	General purpose
1C	7	FPO instructions
20-2C	8-11	General purpose
30	12	INTSER0 (Interrupt serial error, channel 0)
34	13	INTSR0 (Interrupt serial receive, channel 0)
38	14	INTST0 (Interrupt serial transmit, channel 0)
3C	15	General purpose
40	16	INTSER1 (Interrupt serial error, channel 1)
44	17	INTSR1 (Interrupt serial receive, channel 1)
48	18	INTST1 (Interrupt serial transmit, channel 1)
4C	19	I/O trap
50	20	INTD0 (Interrupt from DMA, channel 0)
54	21	INTD1 (Interrupt from DMA, channel 1)
58	22	General purpose
5C	23	General purpose
60	24	INTP0 (Interrupt from peripheral 0)
64	25	INTP1 (Interrupt from peripheral 1)
68	26	INTP2 (Interrupt from peripheral 2)
6C	27	General purpose
70	28	INTTU0 (Interrupt from timer unit 0)
74	29	INTTU1 (Interrupt from timer unit 1)
78	30	INTTU2 (Interrupt from timer unit 2)
7C	31	INTTB (Interrupt from time base counter)
080-3FF	32-255	General purpose

Execution of a vectored interrupt occurs as follows:

(SP-1, SP-2) ← PSW
(SP-3, SP-4) ← PS
(SP-5, SP-6) ← PC
SP ← SP-6
IE ← 0, BRK ← 0
PS ← vector high bytes
PC ← vector low bytes

Hardware Interrupt Configuration

The V35 Plus features a high-performance on-chip controller capable of controlling multiple processing for interrupts from up to 17 different sources (5 external, 12 internal). The interrupt configuration includes system interrupts that are functionally compatible with those of the V20/V30 and unique high-performance microcontroller interrupts.

Interrupt Sources

The interrupt sources on the V35 Plus are similar to those on the V35. The 17 interrupt sources (table 3) are divided into groups for management by the interrupt controller. Using software, each of the groups can be assigned a priority from 0 (highest) to 7 (lowest). The priority of individual interrupts within a group is fixed in hardware.

The ISPR is an 8-bit SFR; bits PR_0-PR_7 correspond to the eight possible interrupt request priorities. The ISPR keeps track of the priority of the interrupt currently being serviced by setting the appropriate bit. The address of the ISPR is XXFFCH. The ISPR format is shown below.

PR_7	PR_2	PR_5	PR_4	PR_3	PR_2	PR_1	PR_0

NMI and INT are system-type external vectored interrupts. NMI is not maskable via software. INTR is maskable (IE bit in PSW) and requires that an external device provide the interrupt vector number. It allows expansion by the addition of an external interrupt controller (μPD71059).

NMI, INTP0, and INTP1 are edge-sensitive maskable interrupt inputs. By selecting the appropriate bits in the interrupt mode register, these inputs can be programmed to be either rising or falling edge triggered. ES0-ES2 correspond to INTP0-INTP2, respectively. See figure 5.

Figure 5. External Interrupt Mode Register (INTM)

0	ES2	0	ES1	0	ES0	0	ESNMI
7							0

ES2	**INTP2 Input Effective Edge**
0	Falling edge
1	Rising edge
ES1	**INTP1 Input Effective Edge**
0	Falling edge
1	Rising edge
ES0	**INTP0 Input Effective Edge**
0	Falling edge
1	Rising edge
ESNMI	**NMI Input Effective Edge**
0	Falling edge
1	Rising edge

Interrupt Factor Register

The primary enhancement of the V35 Plus interrupt control unit is the addition of a special function register that stores the vector type that last caused an interrupt. This IRQS register (figure 5A) stores the vector until the next interrupt request is accepted, but is not changed by response to NMI, INT, or macroservice interrupts.

The main purpose of the IRQS register is to allow several interrupts within a given priority level to be serviced with context switching. Once the interrupt service routine is executing, the cause of the interrupt can be determined only by reading this register, rather than by long and time-consuming software determination. It is recommended that the contents of the IRQS register be read before interrupts are re-enabled to avoid confusion within multiprocessing environments.

Figure 5A. Interrupt Factor Register (IRQS)

0	0	0	Interrupt Vector
7		5	4 0

Interrupt Factor	Interrupt Vector
INTTU0	1CH
INTTU1	IDH
INTTU2	IEH
INTD0	14H
INTD1	15H
INTP0	18H

Interrupt Factor	Interrupt Vector
INTP1	19H
INTP2	1AH
INTSER0	0CH
INTSR0	0DH
INTST0	0EH
INTSER1	10H
INTSR1	11H
INTST1	12H
INTTB	1FH

Table 3. Interrupt Sources

Interrupt Source	External/ Internal	Vector	Macro Service	Bank Switching	Priority Order: Setting Possible	Priority Order: Between Groups	Priority Order: Within Groups	Multiple Processing Control
NMI Nonmaskable interrupt	External	2	No	No	No	0	—	Not accepted
INTTU0 Interrupt from timer unit 0	Internal	28	Yes	Yes	Yes	1	1	Accepted
INTTU1 Interrupt from timer unit 1	Internal	29	Yes	Yes	Yes	1	2	
INTTU2 Interrupt from timer unit 2	Internal	30	Yes	Yes	Yes	1	3	
INTD0 Interrupt from DMA channel 0	Internal	20	No	Yes	Yes	2	1	Accepted
INTD1 Interrupt from DMA channel 1	Internal	21	No	Yes	Yes	2	2	
INTP0 Interrupt from peripheral 0	External	24	Yes	Yes	Yes	3	1	Accepted
INTP1 Interrupt from peripheral 1	External	25	Yes	Yes	Yes	3	2	
INTP2 Interrupt from peripheral 2	External	26	Yes	Yes	Yes	3	3	
INTSER0 Interrupt from serial error on channel 0	Internal	12	No	Yes	Yes	4	1	Accepted
INTSR0 Interrupt from serial receiver of channel 0	Internal	13	Yes	Yes	Yes	4	2	
INTST0 Interrupt from serial transmitter of channel 0	Internal	14	Yes	Yes	Yes	4	3	

Table 3. Interrupt Sources (cont)

Interrupt Source	External/ Internal	Vector	Macro Service	Bank Switching	Priority Order: Setting Possible	Priority Order: Between Groups	Priority Order: Within Groups	Multiple Processing Control
INTSER1 Interrupt from serial error on channel 1	Internal	16	No	Yes	Yes	5	1	Accepted
INTSR1 Interrupt from serial receiver of channel 1	Internal	17	Yes	Yes	Yes	5	2	
INTST1 Interrupt from serial transmitter of channel 1	Internal	18	Yes	Yes	Yes	5	3	
INTTB Interrupt from time base counter	Internal	31	No	No	No (preset to 7)	6	—	Accepted
INT Interrupt	External	Ext input	No	No	No	7	—	Not accepted

Interrupt Processing Modes

Interrupts, with the exception of NMI, INT, and INTTB, have high-performance capability and can be processed in any of three modes: standard vectored interrupt, register bank context switching, or macro service function. The processing mode for a given interrupt can be chosen by enabling the appropriate bits in the corresponding interrupt request control register. As shown in table 3, each individual interrupt, with the exception of INTR and NMI, has its own associated IRC register. The format for all IRC registers is shown in figure 6.

All interrupt processing routines other than those for NMI and INT must end with the execution of an FINT instruction. Otherwise, subsequently, only interrupts of a higher priority will be accepted. FINT allows the internal interrupt controller to reset the highest priority bit set in the ISPR register.

In the vectored interrupt mode, the CPU traps to the vector location in the interrupt vector table.

Register Bank Switching

Register bank context switching allows interrupts to be processed rapidly by switching register banks. After an interrupt, the new register bank selected is that which has the same register bank number (0-7) as the priority of the interrupt to be serviced. The PC and PSW are automatically stored in the save areas of the new register bank and the address of the interrupt routine is loaded from the vector PC storage location in the new register bank. As in the vectored mode, the IE and BRK bits in the PSW are cleared to zero.

After interrupt processing, execution of the RETRBI (return from register bank interrupt) returns control to the former register bank and restores the former PC and PSW. Figures 7 and 8 show register bank context switching and register bank return. Figure 9 shows software-initiated task switching.

Specific IRC registers include the following.

Symbol	IRC Register
DIC0, DIC1	DMA
EXIC0-EXIC2	External
SEIC0, SEIC1	Serial error
SRIC0, SRIC1	Serial receive
STIC0, STIC1	Serial transmit
TMIC0-TMIC2	Timer

Figure 6. Interrupt Request Control Registers (IRC)

IF	IMK	MS/INT	ENCS	0	PR_2	PR_1	PR_0
7							0

IF	**Interrupt Flag**
0	No interrupt request generated
1	Interrupt request generated
IMK	**Interrupt Mask**
0	Open (interrupts enabled)
1	Closed (interrupts disabled)
MS/INT	**Interrupt Response Method**
0	Vector interrupt or register bank switching
1	Macroservice function
ENCS	**Register Bank Switching Function**
0	Not used
1	Used
PR_2-PR_0	**Interrupt Group Priority (0-7)**
0 0 0	Highest (0)
	↓
1 1 1	Lowest (7)

Figure 7. Register Bank Context Switching

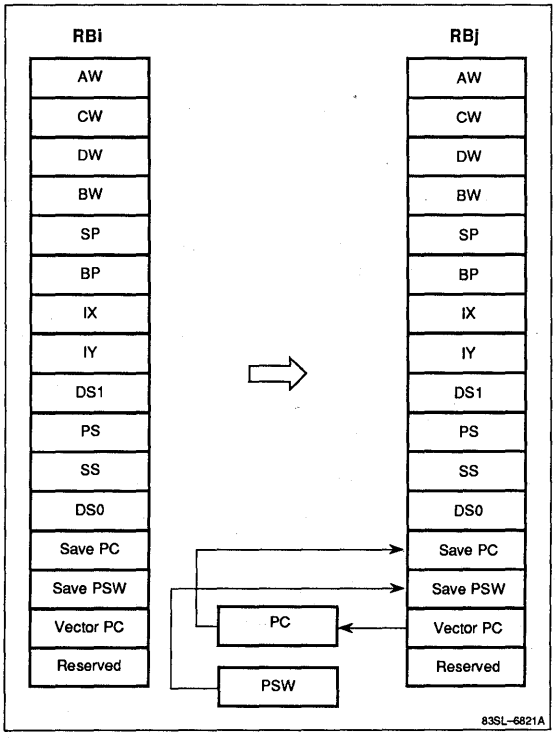
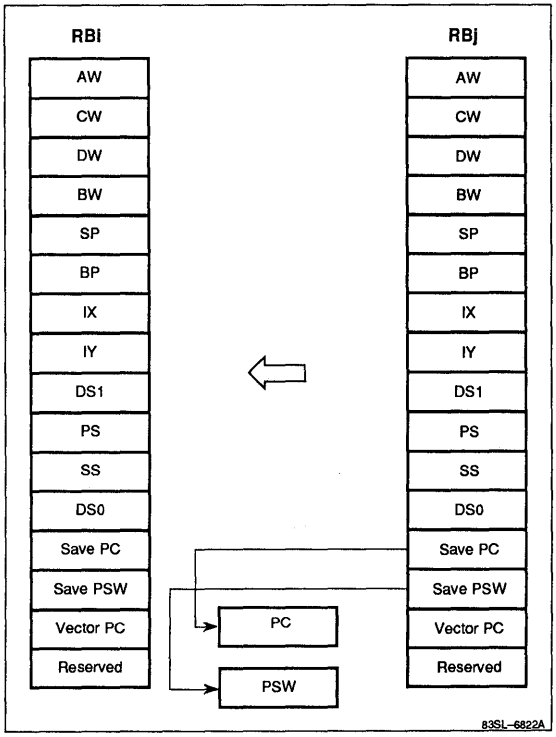
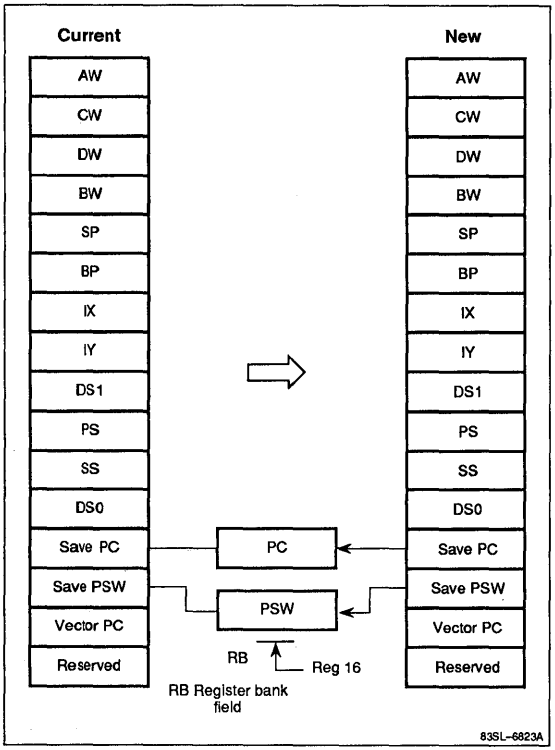


Figure 8. Register Bank Return



4e

Figure 9. Task Switching



MACROSERVICE FUNCTION

The macroservice function (MSF) is a special microprogram that acts as an internal DMA controller between on-chip peripherals (special-function registers, SFR) and memory. The MSF greatly reduces the software overhead and CPU time that other processors would require for register save processing, register returns, and other handling associated with interrupt processing.

If the MSF is selected for a particular interrupt, each time the request is received, a byte or word of data will be transferred between the SFR and memory without interrupting the CPU. Each time a request occurs, the macroservice counter is decremented. When the counter reaches zero, an interrupt to the CPU is generated. The MSF also has a character search option. When selected, every byte transferred will be compared to an 8-bit search character and an interrupt will be generated if a match occurs or if the macroservice counter counts out.

Like the NMI, INT, and INTTB, the two DMA controller interrupts (INTD0, INTD1) and the serial error interrupts (INTSER0, INTSER1) do not have MSF capability.

There are eight, 8-byte macroservice channels mapped into internal RAM from XXE00H to XXE3FH. Figure 10 shows the components of each channel.

Setting the macroservice mode for a given interrupt requires programming the corresponding macroservice control register. Each individual interrupt serviceable with the MSF has its own associated MSC special-function register. The general format for all MSC registers is shown in figure 11.

Figure 10. Macroservice Channels

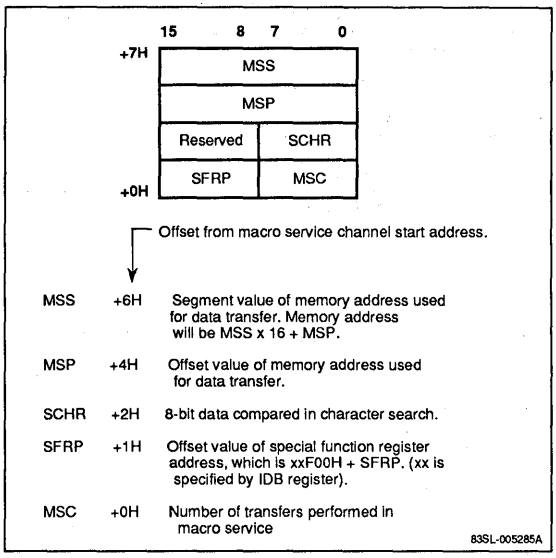


Figure 11. Macroservice Control Registers (MSC)

MSM_2	MSM_1	MSM_0	DIR	0	CH_2	CH_1	CH_0
7							0

MSM_2-MSM_0	Macroservice Mode
0 0 0	Normal (8-bit transfer)
0 0 1	Normal (16-bit transfer)
1 0 0	Character search (8-bit transfer
	Other combinations are not allowed.
DIR	**Data Transfer Direction**
0	Memory to SFR
1	SFR to memory
CH_2-CH_0	**Macroservice Channel**
0 0 0	Channel 0
	↓
1 1 1	Channel 7

TIMER UNIT

The μPD70335 (figure 12) has two programmable 16-bit interval timers (TM0, TM1) on-chip, each with variable input clock frequencies. Each of the two 16-bit timer registers has an associated 16-bit modulus register (MD0, MD1). Timer 0 operates in either the interval timer mode or one-shot mode; timer 1 has only the interval timer mode.

Interval Timer Mode

In this mode, TM0/TM1 are decremented by the selected input clock and, after counting out, the registers are automatically reloaded from the modulus registers and counting continues. Each time TM1 counts out, interrupts are generated through TF1 and TF2 (Timer Flags 1, 2). When TM0 counts out, an interrupt is generated through TF0. The timer-out signal can be used as a square-wave output whose half-cycle is equal to the count time. There are two selectable input clocks (SCLK: system clock = $f_{OSC}/2$; f_{OSC} = 10 MHz).

Clock	Timer Resolution	Full Count
SCLK/6	1.2 μs	78.643 ms
SCLK/128	25.6 μs	1.678 s

One-Shot Mode

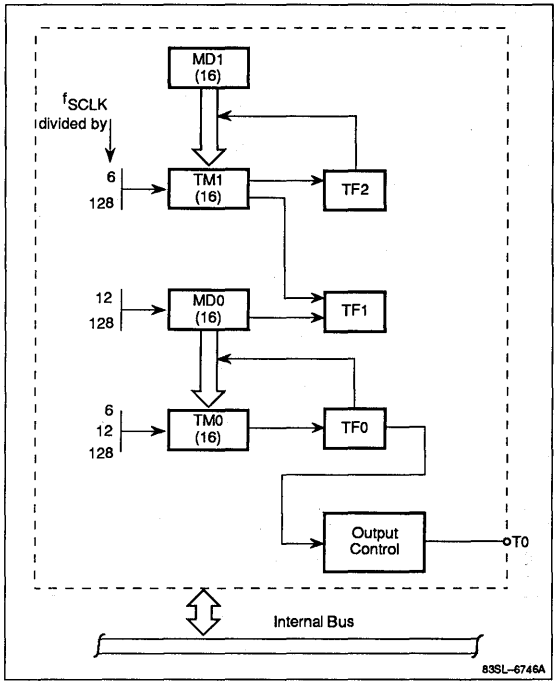
In the one-shot mode, TM0 and MD0 operate as independent one-shot timers. Starting with a preset value, each is decremented to zero. At zero, counting ceases and an interrupt is generated by TF0 (from TM0) or TF1 (from MD0). One-shot mode allows two selectable input clocks (f_{OSC} = 10 MHz).

Clock	Timer Resolution	Full Count
SCLK/12	2.4 μs	157.283 ms
SCLK/128	25.6 μs	1.678 s

Timer Control Registers

Setting the desired timer mode requires programming the timer control register. See figures 13 and 14 for format.

Figure 12. Timer Unit Block Diagram



4e

Figure 13. Timer Control Register 0 (TMC0)

TS0	TCLK0	MS0	MCLK0	ENT0	ALV	MOD_1	MOD_0
7							0

TS0	Timer 0 in Either Mode
0	Stop countdown
1	Start countdown

MOD_1	MOD_0	TCLK0	TM0 Register Clock Frequency
0	0	0	$f_{SCLK}/6$ (Interval)
0	0	1	$f_{SCLK}/128$ (Interval)
0	1	0	$f_{SCLK}/12$ (One-shot)
0	1	1	$f_{SCLK}/128$ (One-shot)

MS0	MD0 Register Countdown (One-Shot Mode)
0	Stop
1	Start

MCLK0	MD0 Register Clock Frequency
0	$f_{SCLK}/12$
1	$f_{SCLK}/128$

ENT0	TOUT Square-Wave Output
0	Disable
1	Enable

ALV	TOUT Initial Level (Counter Stopped)
0	Low
1	High

MOD_1	MOD_0	Timer Unit Mode
0	0	Interval timer
0	1	One-shot
1	X	Reserved

Figure 14. Timer Control Register 1 (TMC1)

TS1	TCLK1	0	0	0	0	0	0
7							0

TS1	Timer 1 Countdown
0	Stop
1	Start

TCLK1	Timer 1 Clock Frequency
0	$f_{SCLK}/6$
1	$f_{SCLK}/128$

TIME BASE COUNTER

The 20-bit free-running time base counter (TBC) controls internal timing sequences and is available as the source of periodic interrupts at lengthy intervals. One of four interrupt periods can be selected by programming the TB_0 and TB_1 bits in the processor control register (PRC). The TBC interrupt is unlike the others because it is fixed as a level 7 vectored interrupt. Macroservice and register bank switching cannot be used to service this interrupt. See figures 14A and 14B.

Figure 14A. Time Base Interrupt Request Control Register (TBIC)

TBF	TBMK	0	0	0	1	1	1
7			Address xxFECH				0

TBF	Time Base Interrupt Flag
0	No interrupt generated
1	Interrupt generated

TBMK	Time Base Interrupt Mask
0	Unmasked
1	Masked

Figure 14B. Processor Control Register (PRC)

0	RAMEN	0	0	TB_1	TB_0	PCK_1	PCK_0
7			Address xxFEBH				0

RAMEN	Built-In RAM
0	Disable
1	Enable

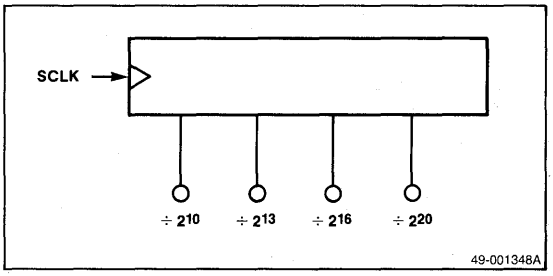
TB_1	TB_0	Time Base Interrupt Period
0	0	$2^{10}/f_{SCLK}$
0	1	$2^{13}/f_{SCLK}$
1	0	$2^{16}/f_{SCLK}$
1	1	$2^{20}/f_{SCLK}$

PCK_1	PCK_0	System Clock Frequency (f_{SCLK})
0	0	$f_{OSC}/2$
0	1	$f_{OSC}/4$
1	0	$f_{OSC}/8$
1	1	Reserved

The RAMEN bit in the PRC register allows the internal RAM to be removed from the memory address space to implement faster instruction execution.

The TBC (figure 14C) uses the system clock as the input frequency. The system clock can be changed by programming the PCK_0 and PCK_1 bits in the processor control register (PRC). Reset initializes the system clock to $f_{OSC}/8$ (f_{OSC} = external oscillator frequency).

Figure 14C. Time Base Counter (TBC) Block Diagram



REFRESH CONTROLLER

The μPD70335 has an on-chip refresh controller for dynamic and pseudostatic RAM mass storage memories. The refresh controller generates refresh addresses and refresh pulses. It inserts refresh cycles between the normal CPU bus cycles according to refresh specifications.

The refresh controller outputs a 9-bit refresh address on address bits A_0-A_8 during the refresh bus cycle. Address bits A_9-A_{19} are all zeros. The 9-bit refresh address is automatically incremented at every refresh timing for 512 row addresses. The 8-bit refresh mode (RFM) register (figure 15) specifies the refresh operation and allows refresh during both CPU HALT and HOLD modes. Refresh cycles are automatically timed to minimize the effect on system throughput.

The following shows the $\overline{\text{REFRQ}}$ pin level in relation to bits 4 (RFEN) and 7 (RFLV) of the refresh mode register.

RFEN	RFLV	$\overline{\text{REFRQ}}$ Level
0	0	0
0	1	1
1	0	0
1	1	Refresh pulse output

It should be noted that since the V35 Plus directly supports dynamic RAM memory, the refresh controller output should be gated into the RAS input of the memory chips. When combined with the chip select logic and the MREQ signals, a direct DRAM interface is supported.

SERIAL CONTROL UNIT

The serial unit of the μPD70335 is functionally identical to that of the standard V35, with the exception of several enhanced features.

All serial status information is moved to the Serial Status Register (SSTn) on the V35 Plus. Included in this register is an additional flag which signals that the transmit shift register is clear of data. This flag allows software to poll for the completion of a message (the last bit of the last byte is shifted out when the ALL SENT bit is set). All error flags are available in this register (refer to figure 16).

Please refer to the μPD70330 (V35) data sheet for additional information on the serial channels.

Figure 15. Refresh Mode Register (RFM)

RFLV	HLDRF	HLTRF	RFEN	RFW_1	RFW_0	RFT_1	RFT_0
7			Address xxFE1H				0

RFLV	RFEN	$\overline{\text{REFRQ}}$ Output Signal Level
0	0	0
1	0	1
0	1	0
1	1	Refresh pulse
HLDRF		**Automatic Refresh Cycle in HOLD Mode**
0		Disabled
1		Enabled
HLTRF		**Automatic Refresh Cycle in HALT Mode**
0		Disabled
1		Enabled
RFEN		**Automatic Refresh Cycle**
0		Refresh pin = RFLV
1		Refresh enabled
RFW_1	**RFW_0**	**No. of Wait States Inserted in Refresh Cycle**
0	0	0
0	1	1
1	0	2
1	1	2
RFT_1	**RFT_0**	**Refresh Period**
0	0	16/SCLK
0	1	32/SCLK
1	0	64/SCLK
1	1	128/SCLK

4e

Figure 16. Serial Status Register (SSTn)

RxDN	ASn	TxBEn	RxBFn	0	ERPn	ERFn	EROn
7							0

Receive Terminal Pin State	**RxDN**
Input state of RxD_n pin is checked by RxDN bit	
All Sent Flag	**ASn**
Reset when transmit data has been written to transmit shifter	0
Set when all the data in the transmit buffer and transmit shift register has been sent (Note 1)	1
Transmit Buffer Empty Flag	**TxBEn**
Reset when transmit data has been written to transmit buffer (Note 1)	0
Set when transmit data in transmit buffer has been sent to shift register	1
Receive Buffer Full Flag	**RxBFn**
Reset when receive data has been read from receive buffer (Note 2)	0
Set when receive data has been sent from shift register to receive buffer (Note 3)	1
Parity Error Flag	**ERPn**
Indicates that transmit parity was not consistent with receive parity (Note 5)	
Framing Error Flag	**ERFn**
Indicates that stop bit was not detected (Note 5)	
Overrun Error Flag	**EROn**
Indicates that succeeding receive has completed before the previous receive data is taken over from the receive buffer (Note 5)	

Notes:

(1) Transmitter flags are reset to 1 when the value of either the band rate generator or serial control register is written.

(2) Receive buffer full flag is also reset when either the band rate generator or serial control register is written.

(3) Receive buffer full flag is not related to the receive error state.

(4) Error flags are cleared when the next data byte is received.

(5) In the table, n = 0 or 1.

Table 4. DMA Controller Operation

	Single-Step Mode	Burst Mode	Single-Transfer Mode	Demand Release Mode
Transmission coverage	Memory - memory	Memory - memory	Memory - I/O	Memory - I/O
Function	Under one time of DMA request instruction, one bus cycle and one DMA transmission are alternately executed the specified number of times.	Under one DMA request, specified number of DMA transmissions are executed.	One DMA transfer is executed every time DMA request occurs.	DMA transmission is executed while DMARQ terminal is kept high-level.
DMA start	Rise of DMARQ Setting TDMA bit of DMA control register	Rise of DMARQ Setting TDMA bit of DMA control register	Rise of DMARQ	High level of DMARQ
Halt method	Depends on software Terminal count decremented from zero	None Terminal count decremented from zero	Depends on software Terminal count decremented from zero	Halted at low level of DMARQ during DMA transmission Terminal count decremented from zero
Interrupt	All accepted	Not accepted during DMA transmission	All accepted	All accepted except during DMA transmission
During halt	Specified times of DMA transmission are executed consecutively	Specified number of DMA transfers are executed consecutively	Active	Active
DMA request during DMA transmission	DMA at channel 1 is retained while DMA at channel 0 is executed	Other DMA is retained until DMA transmission is terminated.	DMA transmission under request is executed after one DMA transmission is over	DMA at channel 1 is retained while DMA at channel 0 is executed.

DMA CONTROLLER

Two memory-to-memory transfer modes (single-step and burst) are supported as well as two I/O-to-memory modes (single-transfer and demand release). Refer to table 4.

The most significant V35 Plus enhancement boosts the transfer rates of the dual internal DMA channels to full bus bandwidth. All operational modes remain the same as the V35, but since the V35 Plus DMA controller is implemented in hard-wired logic, the control delays of a microprogrammed method are not present. As a result, the demand release mode transfer rate boasts a theoretical transfer rate of over 6M bytes per second.

The μPD70335 DMA control registers are moved from the internal RAM to the SFR area; thus the V35 Plus may effectively have a larger internal RAM memory area than comparable designs on the standard V35.

Additionally, the μPD70335 DMA controller uses linear registers for both source and destination address pointers. Thus, three 8-bit registers completely specify the DMA address pointers as shown in figure 17. These pointers may be updated by byte (±1) or word (±2) quantities as programmed in the DMA channel mode register shown in figure 18. This register also specifies the operational mode of the channel. The EDMA bit is automatically cleared when terminal count is reached, and DMA requests are ignored when this bit is cleared.

Figure 17. DMA Address Registers

23 – 20	19 – 16	15 – 8	7 – 0
0 0 0 0	SARnH	SARnM	SARnL
0 0 0 0	DARnH	DARnM	DARnL

Applied addresses (20 bits)

n = 0,1

83YL-6715A

Figure 18. DMA Channel Mode Registers (DMAMn)

MD_2	MD_1	MD_0	W	EDMA	TDMA	0	0
7							0

MD_2-MD_0	Transfer Mode
0 0 0	Single-step (memory to memory)
0 0 1	Demand release (I/O to memory)
0 1 0	Demand release (memory to I/O)
0 1 1	Disabled
1 0 0	Burst (memory to memory)
1 0 1	Single-transfer (I/O to memory)
1 1 0	Single-transfer (memory to I/O)
1 1 1	Disabled

W	Transfer Method
0	Byte transfer
1	Word transfer

EDMA	TDMA	Transfer Condition
0	0	Disabled
1	0	DMA channel enabled
1	1	Software initiate DMA (memory to memory modes)

The TDMA bit is only valid for single-step and burst modes. This bit allows software initiation of the DMA transfer (provided the EDMA bit is set); the bit always reads as zero and has no meaning in the demand-release or single-transfer modes.

The DMA address pointers may be incremented or decremented per transfer as specified in the DMA address update register shown in figure 19. The address pointer can also be programmed to remain the same, allowing repeated transfers to or from a location.

Figure 19. DMA Address Control Registers (DMAC)

0	0	PD_1	PD_0	0	0	PS_1	PS_0
7							0

PD_1-PD_0	Destination Address Offset
0 0	No modification
0 1	Increment
1 0	Decrement
1 1	No modification

PS_1-PS_0	Source Address Offset
0 0	No modification
0 1	Increment
1 0	Decrement
1 1	No modification

The DMAAKn signals are not output for memory-to-memory transfer modes, but are driven low for each transfer I/O to/from memory. Nominal DMA bus cycles are three clock states; however, programmable wait

4e

states may be added. Wait states for memory-to-memory transfers are added to both source and destination addresses as programmed for each specific address.

During memory-to-I/O transfers, the number of wait states inserted is determined by the slower of the source and destination. I/O-to-memory transfers add the number of wait states required by the memory write address.

PARALLEL I/O PORTS

The μPD70335 has three 8-bit parallel I/O ports: P0, P1, and P2. Refer to figures 20 through 24. Special function register (SFR) locations can access these ports. The port lines are individually programmable as inputs or outputs. Many of the port lines have dual functions as port or control lines.

Use the associated port mode and port mode control registers to select the mode for a given I/O line.

The analog comparator port (PT) compares each input line to a reference voltage. The reference voltage is programmable to be the (V_{TH} input pin) x n/16, where n = 1 to 16. See figure 25.

Figure 20. Port Mode Registers 0 and 2 (PM0, PM2)

PM_7	PM_6	PM_5	PM_4	PM_3	PM_2	PM_1	PM_0
7							0

PM_n	Input or Output Bit Selection
0	Output port mode
1	Input port mode

n = 7 through 0

Figure 21. Port Mode Register 1 (PM1)

$PM1_7$	$PM1_6$	$PM1_5$	$PM1_4$	1	1	1	1
7							0

$PMC1_n$	$PM1_n$	Port Mode Input/Output (Port P1n)
0	0	Output port mode
0	1	Input port mode

n = 7, 6, 5, or 4.

Figure 22. Port Mode Control Register 0 (PMC0)

$PMC0_7$	—	—	—	—	—	—	—
7							0

$PMC0_7$	Port or Control Bit Selection
0	Port mode
1	CLKOUT

Figure 23. Port Mode Control Register 1 (PMC1)

$PMC1_7$	$PMC1_6$	$PMC1_5$	$PMC1_4$	$PMC1_3$	$PMC1_2$	$PMC1_1$	$PMC1_0$
7							0

$PMC1_7$	Port/Control Bit Selection
0	$P1_7$ I/O
1	READY input
$PMC1_6$	**Port/Control Bit Selection**
0	$P1_6$ I/O
1	$\overline{SCKO}$ output
$PMC1_5$	**Port/Control Bit Selection**
0	$P1_5$ I/O
1	TOUT output
$PMC1_4$	**Port/Control Bit Selection**
0	$P1_4$ I/O or $\overline{POLL}$ input
1	INT input
$PMC1_3$	**Port/Control Bit Selection**
0	INTP2/$P1_3$ input
1	$\overline{INTAK}$ output
$PMC1_2$	**Port/Control Bit Selection**
x	INTP1/$P1_2$ input
0	INTP2/$P1_3$ input
$PMC1_1$	**Port/Control Bit Selection**
x	INTP0/$P1_1$ input
$PMC1_0$	**Port/Control Bit Selection**
x	NMI/$P1_0$ input

Figure 24. Port Mode Control Register 2 (PMC2)

$PMC2_7$	$PMC2_6$	$PMC2_5$	$PMC2_4$	$PMC2_3$	$PMC2_2$	$PMC2_1$	$PMC2_0$

7 0

$PMC2_7$	Port/Control Bit Selection
0	I/O port
1	HLDRQ output
$PMC2_6$	**Port/Control Bit Selection**
0	I/O port
1	$\overline{HLDAK}$ input
$PMC2_5$	**Port/Control Bit Selection**
0	I/O port
1	$\overline{TC1}$ output
$PMC2_4$	**Port/Control Bit Selection**
0	I/O port
1	$\overline{DMAAKI}$ output
$PMC2_3$	**Port/Control Bit Selection**
0	I/O port
1	$\overline{DMARQ1}$ input
$PMC2_2$	**Port/Control Bit Selection**
x	I/O port
0	TC0 output
$PMC2_1$	**Port/Control Bit Selection**
x	I/O port
1	$\overline{DMAAK0}$ output
$PMC2_0$	**Port/Control Bit Selection**
0	I/O port
1	DMARQ0 input

Figure 25. Port T Mode Register (PMT)

0	0	0	0	PMT_3	PMT_2	PMT_1	PMT_0

7 0

PMT_3-PMT_0	V_{REF}
0000	V_{TH} x 16/16
0001	V_{TH} x 1/16
0010	V_{TH} x 2/16
0011	V_{TH} x 3/16
0100	V_{TH} x 4/16
0101	V_{TH} x 5/16
0110	V_{TH} x 6/16
0111	V_{TH} x 7/16
1000	V_{TH} x 8/16
1001	V_{TH} x 9/16
1010	V_{TH} x 10/16
1011	V_{TH} x 11/16
1100	V_{TH} x 12/16
1101	V_{TH} x 13/16
1110	V_{TH} x 14/16
1111	V_{TH} x 15/16

PROGRAMMABLE WAIT STATE GENERATION

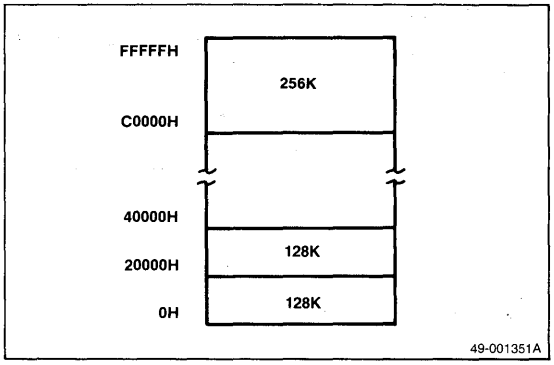
You can generate wait states internally to further reduce the necessity for external hardware. Insertion of these wait states allows direct interface to devices whose access times cannot meet the CPU read/write timing requirements.

When using this function, the entire 1M-byte memory address space is divided into 128K-blocks. Each block can be programmed for zero, one, or two wait states, or two plus those added by the external READY signal. The top two blocks are programmed together as one unit.

The appropriate bits in the wait control word (WTC) control wait state generation. Programming the upper two bits in the wait control word will set the wait state conditions for the entire I/O address space. Figure 26 shows the memory map for programmable wait state generation; see figure 27 for a graphic representation of the wait control word.

4e

Figure 26. Programmable Wait State Generation



STANDBY MODES

The two low-power standby modes are HALT and STOP. Software can cause the processor to enter either mode.

HALT Mode

In the HALT mode, the CPU is inactive and the chip consumes much less power than when operational. The external oscillator remains functional and all peripherals are active. Internal status and output port line conditions are maintained. Any unmasked interrupt can release this mode. In the EI state, interrupts subsequently will be serviced and the HALT state released. In the DI state, program execution is restarted with the instruction following the HALT instruction and the interrupt causing the release from HALT will be latched.

STOP Mode

The STOP mode allows the largest power reduction while maintaining RAM. The oscillator is stopped, halting the CPU and all internal peripherals. Internal status and port pin outputs are maintained. Only a RESET or NMI can release this mode.

A standby flag in the STBC register is reset by rises in the supply voltage. Its status is maintained during normal operation and standby. The STBC register (figure 28) is not initialized by RESET. Use the standby flag to determine whether program execution is returning from standby or from a cold start by setting this flag before entering the STOP mode.

SPECIAL-FUNCTION REGISTERS

Table 5 shows the special-function register mnemonic, type, address, reset value, and function. The eight high-order bits of each address (xx) are specified by the IDB register.

SFR area addresses not listed in table 5 are reserved. If read, the contents of these addresses are undefined, and any write operation will be meaningless.

Figure 27. Wait Control Word (WTC)

IO1	IO0	Block 61	Block 60	Block 51	Block 50	Block 41	Block 40
7			Wait Control, High				0

Block 31	Block 30	Block 21	Block 20	Block 11	Block 10	Block 01	Block 00
7			Wait Control, Low				0

Wait States	Block n1	Block n0
0	0	0
1	0	1
2	1	0
2 or more (control from READY pin)	1	1

n = 0 thru 6

Figure 28. Standby Register (STBC)

0	0	0	0	0	0	0	SBF
7							0

SBF	Standby Flag
0	No changes in V_{DD} (standby)
1	Rising edge on V_{DD} (cold start)

Table 5. Special-Function Registers

Address	Register Function	Symbol	R/W	Manipulation (Note 6)	When Reset
xxF00H	Port 0	P0	R/W	8/1	Undefined
xxF01H	Port mode 0	PM0	W	8	0FFH
xxF02H	Port mode control 0	PMC0	W	8	00H
xxF08H	Port 1	P1	R/W	8/1	Undefined
xxF09H	Port mode 1	PM1	W	8	0FFH
xxF0AH	Port mode control 1	PMC1	W	8	00H
xxF10H	Port 2	P2	R/W	8/1	Undefined
xxF11H	Port mode 2	PM2	W	8	0FFH
xxF12H	Port mode control 2	PMC2	W	8	00H
xxF38H	Threshold port	PT	R	8	Undefined
xxF3BH	Threshold port mode	PMT	R/W	8/1	00H
xxF40H	External interrupt mode	INTM	R/W	8/1	00H
xxF44H	External interrupt macro service control 0 (Note 1)	EMS0	R/W	8/1	Undefined
xxF45H	External interrupt macro service control 1 (Note 1)	EMS1	R/W	8/1	
xxF46H	External interrupt macro service control 2 (Note 1)	EMS2	R/W	8/1	
xxF4CH	External interrupt request control 0 (Note 1)	EXIC0	R/W	8/1	47H
xxF4DH	External interrupt request control 1 (Note 1)	EXIC1	R/W	8/1	
xxF4EH	External interrupt request control 2 (Note 1)	EXIC2	R/W	8/1	

Table 5. Special-Function Registers (cont)

Address	Register Function	Symbol	R/W	Manipulation (Note 6)	When Reset
xxF60H	Receive buffer 0	RxB0	R	8	Undefined
xxF62H	Transmit buffer 0	TxB0	W	8	
xxF65H	Serial receive macro service control 0 (Note 1)	SRMS0	R/W	8/1	
xxF66H	Serial transmit macro service control 0 (Note 1)	STMS0	R/W	8/1	
xxF68H	Serial mode register 0	SCM0	R/W	8/1	00H
xxF69H	Serial control register 0	SCC0	R/W	8/1	
xxF6AH	Baud rate generator 0	BRG0	R/W	8/1	
xxF6BH	Serial status register 0	SST0	R	8	60H
xxF6CH	Serial error interrupt request register 0 (Note 1)	SEIC0	R/W	8/1	47H
xxF6DH	Serial receive interrupt request register 0 (Note 1)	SRIC0	R/W	8/1	
xxF6EH	Serial transmit interrupt request register 0 (Note 1)	STIC0	R/W	8/1	
xxF70H	Serial receive buffer 1	RxB1	R	8	Undefined
xxF72H	Serial transmit buffer 1	TxB1	W	8	
xxF75H	Serial receive macro service register 1 (Note 1)	SRMS1	R/W	8/1	
xxF76H	Serial transmit macro service register 1 (Note 1)	STMS1	R/W	8/1	
xxF78H	Serial communication register 1	SCM1	R/W	8/1	00H
xxF79H	Serial control register 1	SCC1	R/W	8/1	
xxF7AH	Baud rate generator 1	BRG1	R/W	8/1	
xxF7BH	Serial status register 1	SCS1	R	8	60H
xxF7CH	Serial error interrupt request register I (Note 1)	SEIC1	R/W	8/1	47H
xxF7DH	Serial receive interrupt request register 1 (Note 1)	SRIC1	R/W	8/1	
xxF7EH	Serial transmit interrupt request register 1 (Note 1)	STIC1	R/W	8/1	
xxF80H	Timer register 0 (Note 2)	TM0	R/W	16	Undefined
xxF82H	Timer 0 modulo register (Note 2)	MD0	R/W	16	
xxF88H	Timer register 1 (Note 2)	TM1	R/W	16	
xxF8AH	Timer 1 modulo register (Note 2)	MD1	R/W	16	
xxF90H	Timer 0 control register (Note 2)	TMC0	R/W	8/1	00H
xxF91H	Timer 1 control register (Note 2)	TMC1	R/W	8/1	
xxF94H	Timer unit 0 macro service register (Note 1)	TMMS0	R/W	8/1	Undefined
xxF95H	Timer unit 1 macro service register (Note 1)	TMMS1	R/W	8/1	
xxF96H	Timer unit 2 macro service register (Note 1)	TMMS2	R/W	8/1	
xxF9CH	Timer unit 0 interrupt request register (Note 1)	TMIC0	R/W	8/1	47H
xxF9DH	Timer unit 1 interrupt request register (Note 1)	TMIC1	R/W	8/1	
xxF9EH	Timer unit 2 interrupt request register (Note 1)	TMIC2	R/W	8/1	
xxFA0H	DMA address update control register 0	DMAC0	R/W	8/1	Undefined
xxFA1H	DMA mode register 0	DMAM0	R/W	8/1	47H
xxFA2H	DMA address update control register 1	DMAC1	R/W	8/1	Undefined
xxFA3H	DMA mode register 1	DMAM1	R/W	8/1	00H
xxFACH	DMA interrupt request control register 0 (Note 1)	DIC0	R/W	8/1	47H
xxFADH	DMA interrupt request control register 1 (Note 1)	DIC1	R/W	8/1	

4e

Table 5. Special-Function Registers (cont)

Address	Register Function	Symbol	R/W	Manipulation (Note 6)	When Reset
xxFC0H	DMA channel 0 source address pointer low	SAE0L	R/W	16/8	Undefined
xxFC1H	DMA channel 0 source address pointer mid	SAE0M	R/W	16/8	
xxFC0H	DMA Channel 0 source address pointer low	SAE0L	R/W	16/8	
xxFC1H	DMA channel 0 source address pointer mid	SAE0M	R/W	16/8	
xxFC2H	DMA channel 0 source address pointer high	SAR0H	R/W	8	
xxFC4H	DMA channel 0 destination address pointer low	DAR0L	R/W	16/8	
xxFC5H	DMA channel 0 destination address pointer mid	DAR0M	R/W	16/8	
xxFC6H	DMA channel 0 destination address pointer high	DAR0H	R/W	8	
xxFC8H	DMA channel 0 count register	DMATC0	R/W	16/8	
xxFD0H	DMA channel 1 source address pointer low	SAR1L	R/W	16/8	
xxFD1H	DMA channel 1 source address pointer mid	SAR1M	R/W	16/8	
xxFD2H	DMA channel 1 source address pointer high	SAR1H	R/W	8	
xxFD4H	DMA channel 1 destination address pointer low	DAR1L	R/W	16/8	
xxFD5H	DMA channel 1 destination address pointer mid	DAR1M	R/W	16/8	
xxFD6H	DMA channel 1 destination address pointer high	DAR1H	R/W	8	
xxFD8H	DMA channel 1 terminal count register	DMATC1	R/W	16/8	
xxFE0H	Standby control register	STBC	R/W (Note 3)	8/1	Undefined (Note 4)
xxFE1H	Refresh mode register	RFM	R/W	8/1	0FCH
xxFE8H	Wait state control	WTC	R/W	16/8	0FFFFH
xxFEAH	User flag (Note 5)	FLAG	R/W	8/1	00H
xxFEBH	Processor control register	PRC	R/W	8/1	4EH
xxFECH	Time base interrupt request control register (Note 1)	TBIC	R/W	8/1	47H
xxFEFH	Interrupt factor register (Note 1)	IRQS	R	8	Undefined
xxFFCH	Interrupt priority control register (Note 1)	ISPR	R	8	00H
xxFFFH	Internal data area base	IDB	R/W	8/1	0FFH

Notes:

(1) One wait state is inserted into accesses to these registers.

(2) A maximum of 6 wait states are added into accesses to these registers.

(3) Each bit of the standby control register can be set to 1 by an instruction; however, once set, bits cannot be reset to 0 by an instruction (only 1 can be written to this register).

(4) Upon power-on reset = 00H; other = no change.

(5) For the user flag register (FLAG), manipulating bits other than bits 3 and 5 is meaningless. The contents of user flags 0 and 1 (F0 and F1) of the FLAG register are affected by manipulating F0 and F1 of the PSW.

(6) The manipulation column indicates which memory operations can read or modify the register according to the following key.

16	Work operations
8	Byte operations
1	Bit operations

ELECTRICAL SPECIFICATIONS

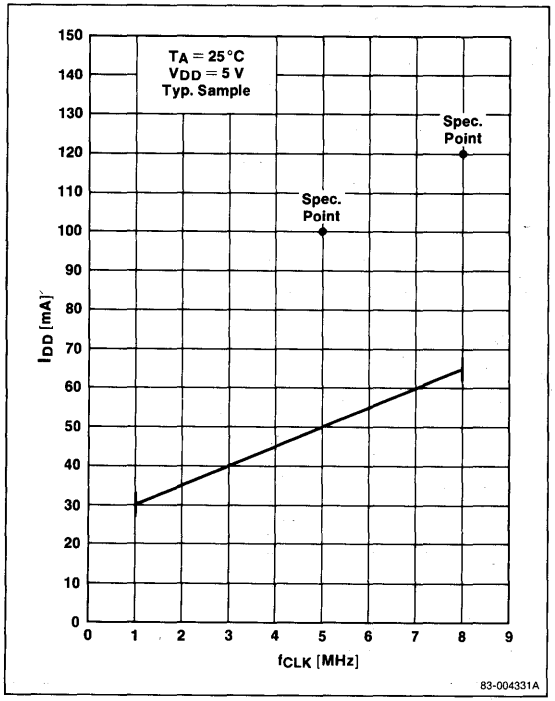
Absolute Maximum Ratings

T_A = 25 °C

Supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Output voltage, V_O	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Threshold voltage, V_{TH}	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Output current, low; I_{OL}	
Each output pin	4.0 mA
Total	50 mA
Output current, high; I_{OH}	
Each output pin	−2.0 mA
Total	−20 mA
Operating temperature range, T_{OPT}	−10 to +70 °C
Storage temperature range, T_{STG}	−65 to +150 °C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Supply Current vs Clock Frequency



Comparator Characteristics

V_{DD} = +5 V ±10%; T_A = −10 to +70°C

Parameter	Symbol	Min	Max	Unit	Conditions
Accuracy	VA_{COMP}		±100	mV	
Threshold voltage	V_{TH}	0	V_{DD} + 0.1	V	
Comparison time	t_{COMP}	64	65	t_{CYK}	
PT input voltage	V_{IPT}	0	V_{DD}	V	

Capacitance

V_{DD} = 0 V; T_A = 25 °C

Parameter	Symbol	Min	Max	Unit	Conditions
Input capacitance	C_I		10	pF	f_c = 1 MHz; unmeasured pins returned to 0 V
Output capacitance	C_O		20	pF	
I/O capacitance	C_{IO}		20	pF	

DC Characteristics

V_{DD} = +5 V ±10%; T_A = −10 to +70 °C (Note 1)

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Supply current, operating	I_{DD1}		65	120	mA	
Supply current, HALT mode	I_{DD2}		25	50	mA	
Supply current, STOP mode	I_{DD3}		10	30	μA	
V_{TH} supply current	I_{TH}		0.5	1.0	mA	V_{TH} = 0 to V_{DD}
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2		V_{DD}	V	All inputs except $\overline{RESET}$, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 x V_{DD}		V_{DD}	V	$\overline{RESET}$, $P1_0$/NMI, X1, X2
Output voltage, low	V_{OL}			0.45	V	I_{OL} = 1.6 mA
Output voltage, high	V_{OH}	V_{DD} − 1.0			V	I_{OH} = −0.4 mA
Input current	I_{IN}			±20	μA	$\overline{EA}$, $P1_0$/NMI; V_I = 0 to V_{DD}
Input leakage current	I_{LI}			±10	μA	All except $\overline{EA}$, $P1_0$/NMI; V_I = 0 to V_{DD}
Output leakage current	I_{LO}			±10	μA	V_O = 0 to V_{DD}

Notes:

(1) The standard operating temperature range is −10 to +70°C. However, extended temperature range parts (−40 to +85°C) are available.

4e

AC Characteristics

V_{DD} = +5 V ±10%; T_A −10 to +70°C; C_L = 100 pF (max)

Parameter	Symbol	Min	Max	Unit	Conditions
Input rise, fall time	t_{IR}, t_{IF}		20	ns	Except X1, X2, RESET, NMI
Input rise, fall time	t_{IRS}, t_{IFS}		30	ns	RESET, NMI (Schmitt)
Output rise, fall time	t_{OR}, t_{OF}		20	ns	Except CLKOUT
X1 cycle time	t_{CYX}	62	250	ns	
X1 width, low	t_{WXL}	20		ns	
X1 width, high	t_{WXH}	20		ns	
X1 rise, fall time	t_{XR}, t_{XF}		20	ns	
CLKOUT cycle time	t_{CYK}	125	2000	ns	
CLKOUT width, low	t_{WKL}	0.5T − 15		ns	Note 1
CLKOUT width, high	t_{WKH}	0.5T − 15		ns	
CLKOUT rise, fall time	t_{KR}, t_{KF}		15	ns	
Address delay time	t_{DKA}	15	90	ns	
Address valid to input data valid	t_{DADR}		T(n + 1.5) − 70	ns	Note 2
$\overline{MREQ}$ to address hold time	t_{HMRA}	0.5T − 30		ns	
$\overline{MREQ}$ to data delay	t_{DMRD}		T(n + 2) − 60	ns	
$\overline{MSTB}$ to data delay	t_{DMSD}		T(n + 1) − 60	ns	
$\overline{MREQ}$ to $\overline{MSTB}$ delay	t_{DMRMSR}	T − 35	T + 35	ns	
$\overline{MREQ}$ width, low	t_{WMRL}	T(n + 2) − 30		ns	
$\overline{MREQ}$, $\overline{MSTB}$ to address hold time	t_{HMA}	0.5T − 30		ns	
Input data hold time	t_{HMD}	0		ns	
Next control setup time	t_{SCC}	T − 25		ns	
$\overline{MREQ}$ to $\overline{TC}$ delay time	t_{DMRTC}		0.5T + 50	ns	
$\overline{MREQ}$ delay time	t_{DAMR}	0.5T − 30		ns	
$\overline{MSTB}$ read delay time	t_{DAMSR}	0.5 T − 30		ns	
MSTB width, low	t_{WMSLR}	T(n + 1) − 30		ns	
Address data output	t_{DADW}		0.5T + 50	ns	
Data output setup time	t_{SDM}	T(n + 2) − 50		ns	
$\overline{MSTB}$ write delay time	t_{DAMSW}	T(n + 0.5) − 30		ns	
$\overline{MREQ}$ to $\overline{MSTB}$ write delay time	t_{DMRMSW}	T(n + 1) − 35	T(n + 1) + 35	ns	
$\overline{MSTB}$ write width low	t_{WMSLW}	T − 30		ns	
Data output hold time	t_{HMDW}	0.5T − 30		ns	
$\overline{IOSTB}$ delay time	t_{DAIS}	0.5T − 30		ns	
$\overline{IOSTB}$ to data input	t_{DISD}		T(n + 1) − 60	ns	
$\overline{IOSTB}$ width, low	t_{WISL}	T(n + 1) − 30		ns	
$\overline{MREQ}$ to $\overline{IOSTB}$ delay time	t_{DMRIS}	T − 35		ns	
Next DMARQ setup time	t_{SDADQ}		T − 50	ns	Demand mode
DMARQ hold time	t_{HDARQ}	0		ns	
$\overline{DMAAK}$ read width, low	t_{WDMRL}	T(n + 2.5) −30		ns	
$\overline{DMAAK}$ write width, low	t_{WDMWL}	T(n + 2) − 30		ns	
$\overline{DMAAK}$ to $\overline{TC}$ delay time	t_{DDATC}		0.5T + 50	ns	

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{TC}$ width, low	t_{WTCL}	(n + 2)T – 30		ns	
$\overline{REFRQ}$ delay time	t_{DARF}	0.5T – 30		ns	
$\overline{REFRQ}$ width, low	t_{WRFL}	T(n + 2) – 30		ns	
Address hold time	t_{HRFA}	0.5T – 30		ns	
$\overline{RESET}$ width low	t_{WRSL1}	30		ms	Crystal oscillator; STOP/ Power on reset
	t_{WRSL2}	5		μs	System warm reset
$\overline{MREQ}$, $\overline{IOSTB}$ to $\overline{READY}$ setup time	t_{SCRY}		T(n) – 100	ns	n ≥ 2
$\overline{MREQ}$, $\overline{IOSTB}$ to $\overline{READY}$ hold time	t_{HCRY}	T(n)		ns	n ≥ 2
HLDRQ setup time	t_{SHQK}	30		ns	
$\overline{HLDAK}$ output delay time	t_{DKHA}	15	80	ns	
Bus control float to $\overline{HLDAK}$ ↓	t_{CFHA}	T – 50		ns	
$\overline{HLDAK}$ ↑ to control output time	t_{DHAC}	T – 50		ns	
HLDRQ to $\overline{HLDAK}$ delay	t_{DHQHA}		3T + 160	ns	
HLDRQ ↓ to control float time	t_{DHQC}	3T + 30		ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
$\overline{HLDAK}$ width, low	t_{WHAL}	T		ns	
$\overline{INTP}$, DMARQ setup	t_{SIQK}	30		ns	
$\overline{INTP}$, DMARQ width, high	t_{WIQH}	8T		ns	
$\overline{INTP}$, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{POLL}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{CTS}$ width, low	t_{WCTL}	2T		ns	
INT setup time	t_{SIRK}	30		ns	
$\overline{INTAK}$ delay time	t_{DKIA}	15	80	ns	
$\overline{INT}$ hold time	t_{HIAIQ}	0		ns	
$\overline{INTAK}$ width, low	t_{WIAL}	2T – 30		ns	
$\overline{INTAK}$ width, high	t_{WIAH}	T – 30		ns	
$\overline{INTAK}$ to data delay time	t_{DIAD}		2T – 130	ns	
$\overline{INTAK}$ to data hold time	t_{HIAD}	0	0.5T	ns	
$\overline{SCKO}$ (TSCK) cycle time	t_{CYTK}	1000		ns	
$\overline{SCKO}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{SCKO}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	
TxD hold time	t_{HTKD}	20		ns	
$\overline{CTS0}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{CTS0}$ (RSCK) width, high	t_{WSRH}	420		ns	

4e

AC Characteristics (cont)

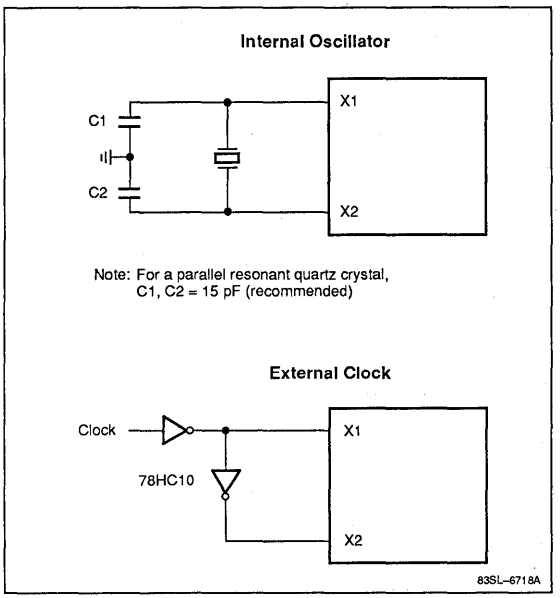
Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{CTS0}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

Notes:

(1) T = CPU clock period (t_{CYK}).

(2) n = number of wait states inserted.

External System Clock Control Source



Recommended Oscillator Components

Ceramic Resonator		Capacitors	
Manufacturer	Product No.	C1 (pF)	C2 (pF)
Kyocera	KBR-10.0M	33	33
Murata Mfg.	CSA.10.0MT	47	47
	CSA16.0MX040	30	30
TDK	FCR10.M2S	30	30
	FCR16.0M2S	15	6

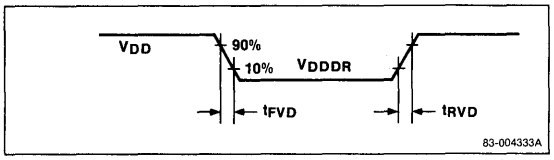
STOP Mode Data Retention Characteristics

T_A = −10 to +70°C

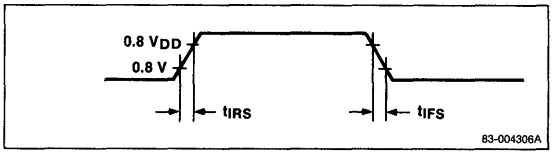
Parameter	Symbol	Min	Max	Unit
Data retention voltage	V_{DDDR}	2.4	5.5	V
V_{DD} rise time	t_{RVD}	200		μs
V_{DD} fall time	t_{FVD}	200		μs

Timing Waveforms

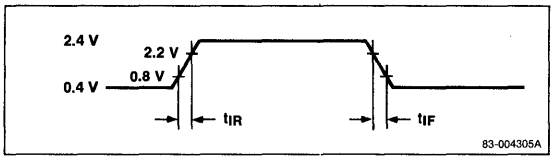
Stop Mode Data Retention Timing



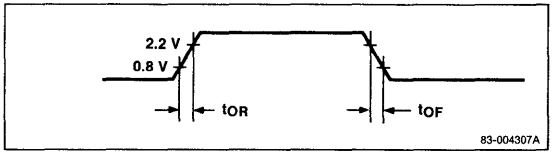
AC Input 2 (RESET, NMI)



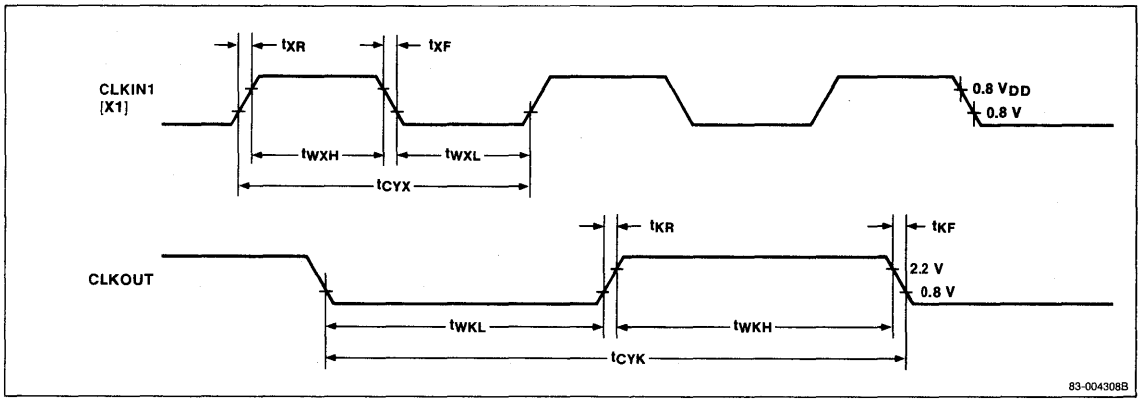
AC Input 1 (Except X1, X2, RESET, NMI)



AC Output (Except CLKOUT)

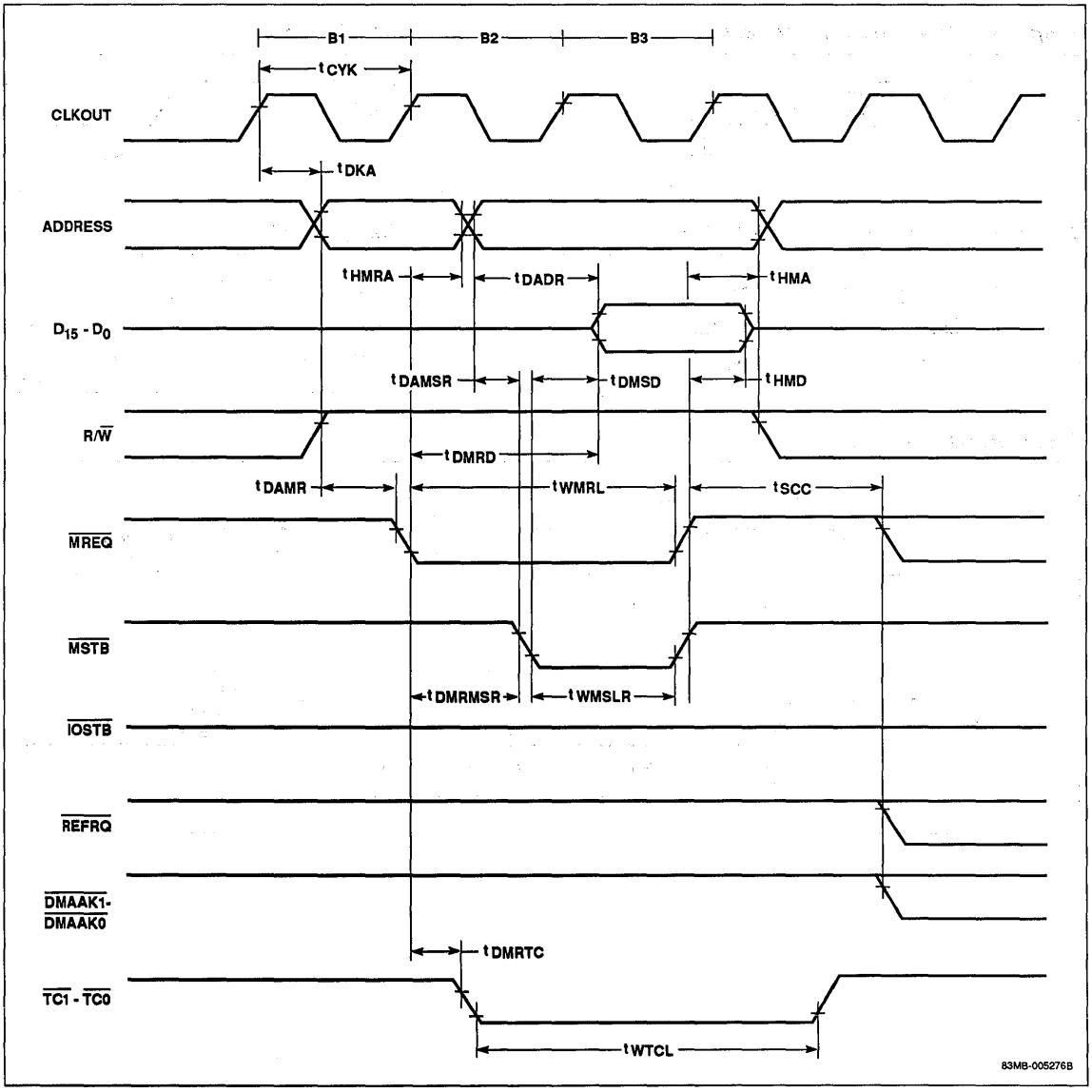


Clock In and Clock Out

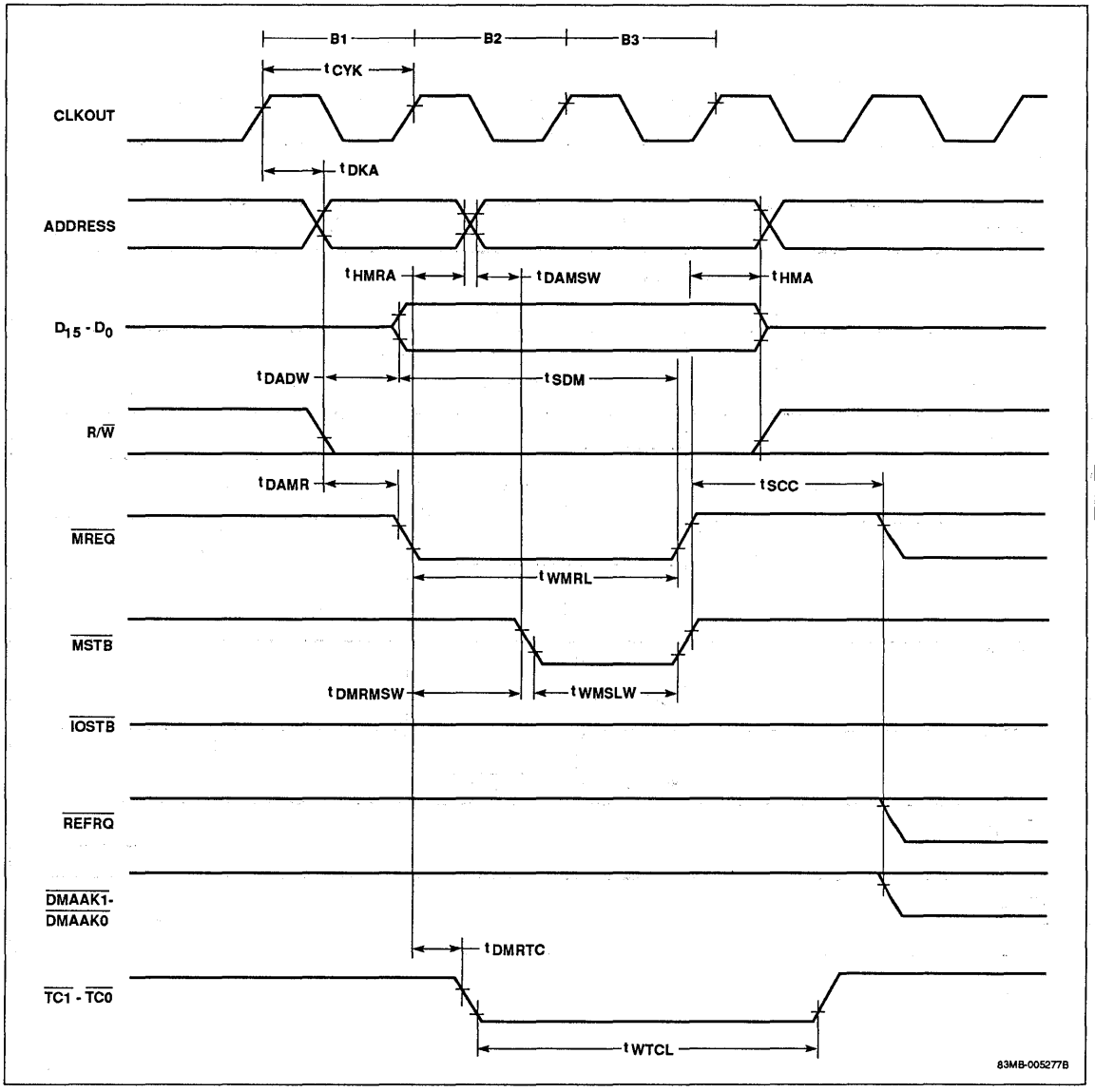


4e

Memory Read

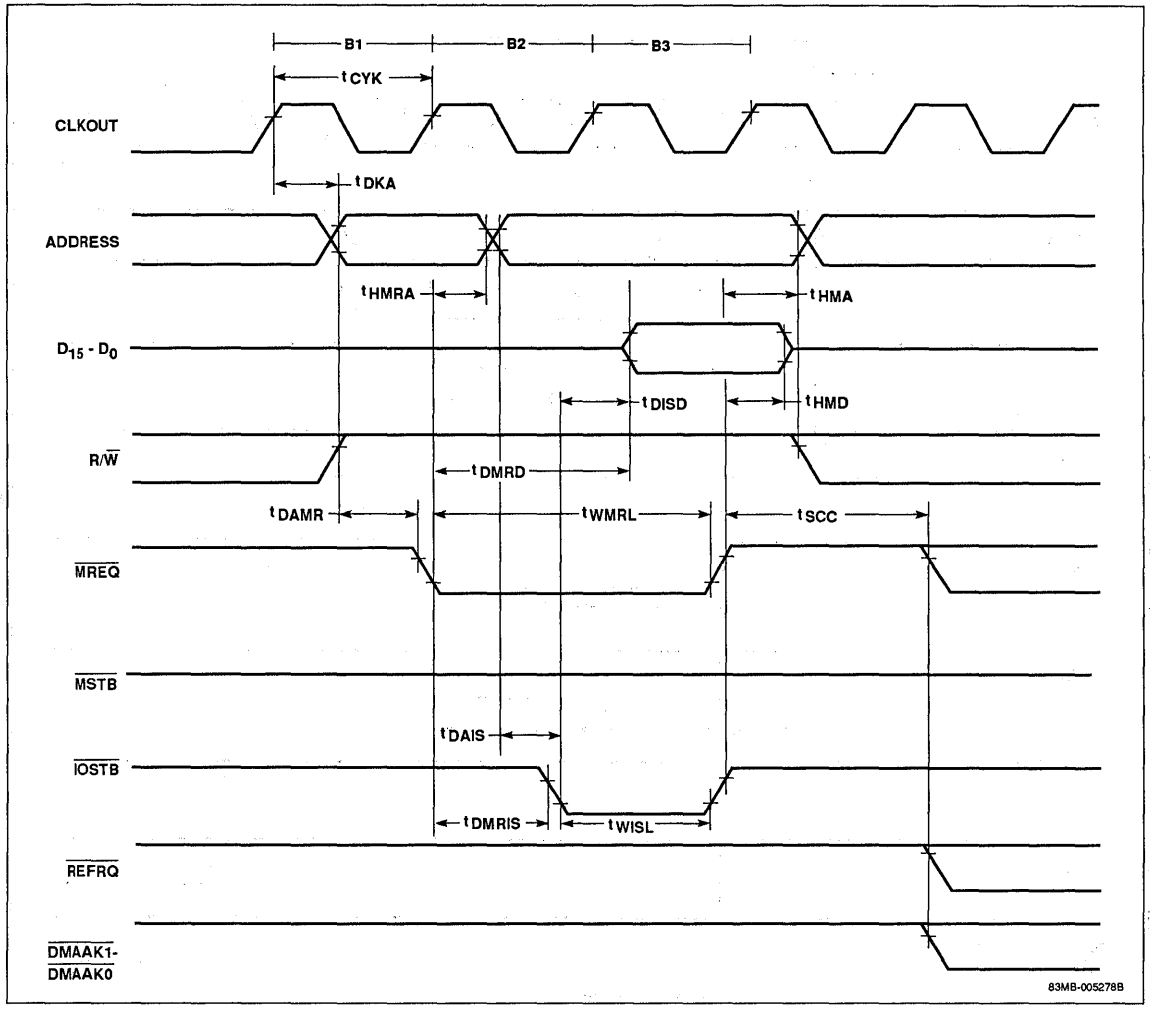

B1
B2
B3
tCYK
CLKOUT
tDKA
ADDRESS
tHMRA
tDADR
tHMA
D15 - D0
tDAMSR
tDMSD
tHMD
R/W
tDMRD
tDAMR
tWMRL
tSCC
MREQ
MSTB
tDMRMSR
tWMSLR
IOSTB
REFRQ
DMAAK1-
DMAAK0
tDMRTC
TC1 - TC0
tWTCL
83MB-005276B

Memory Write

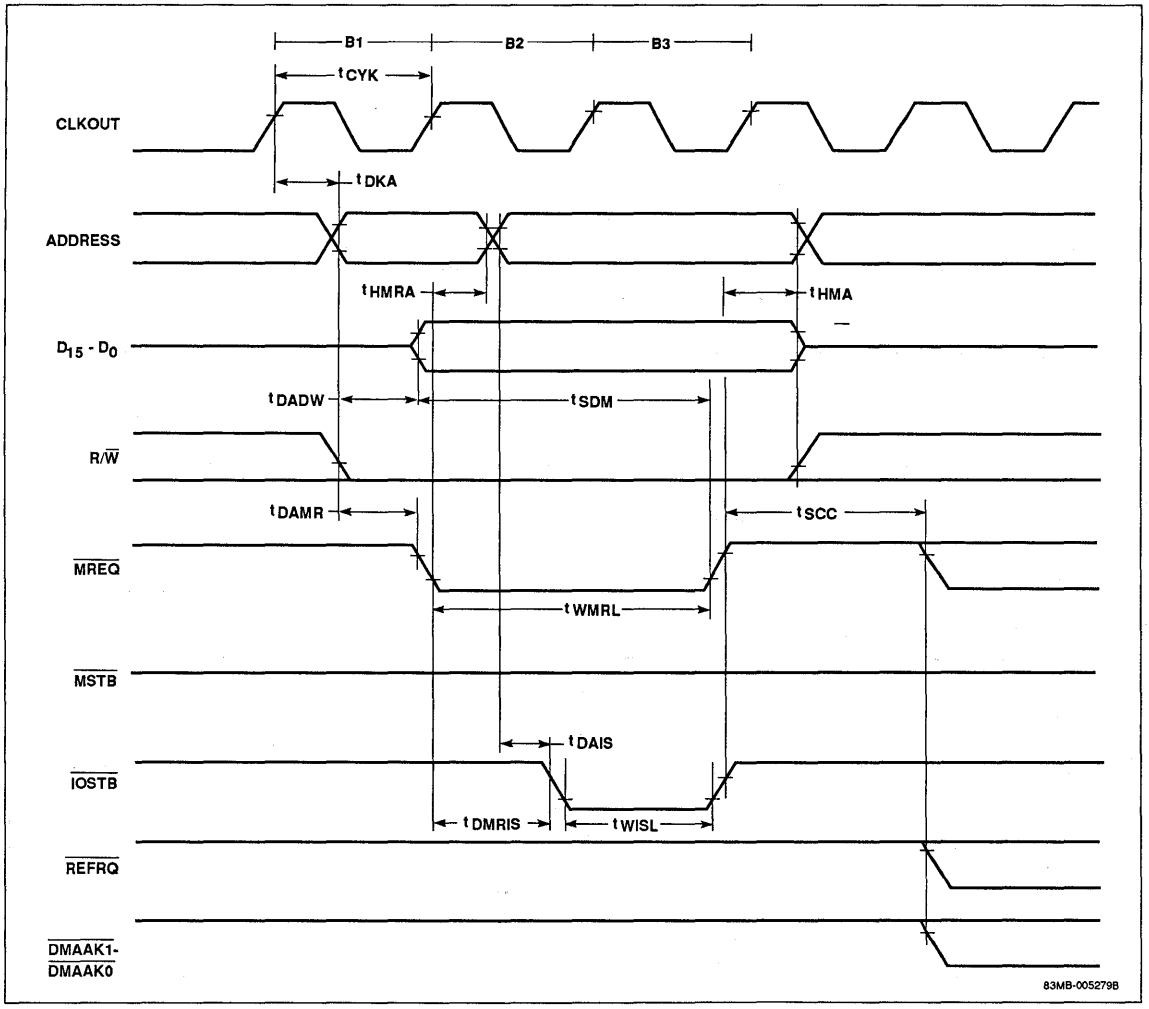


4e

I/O Read

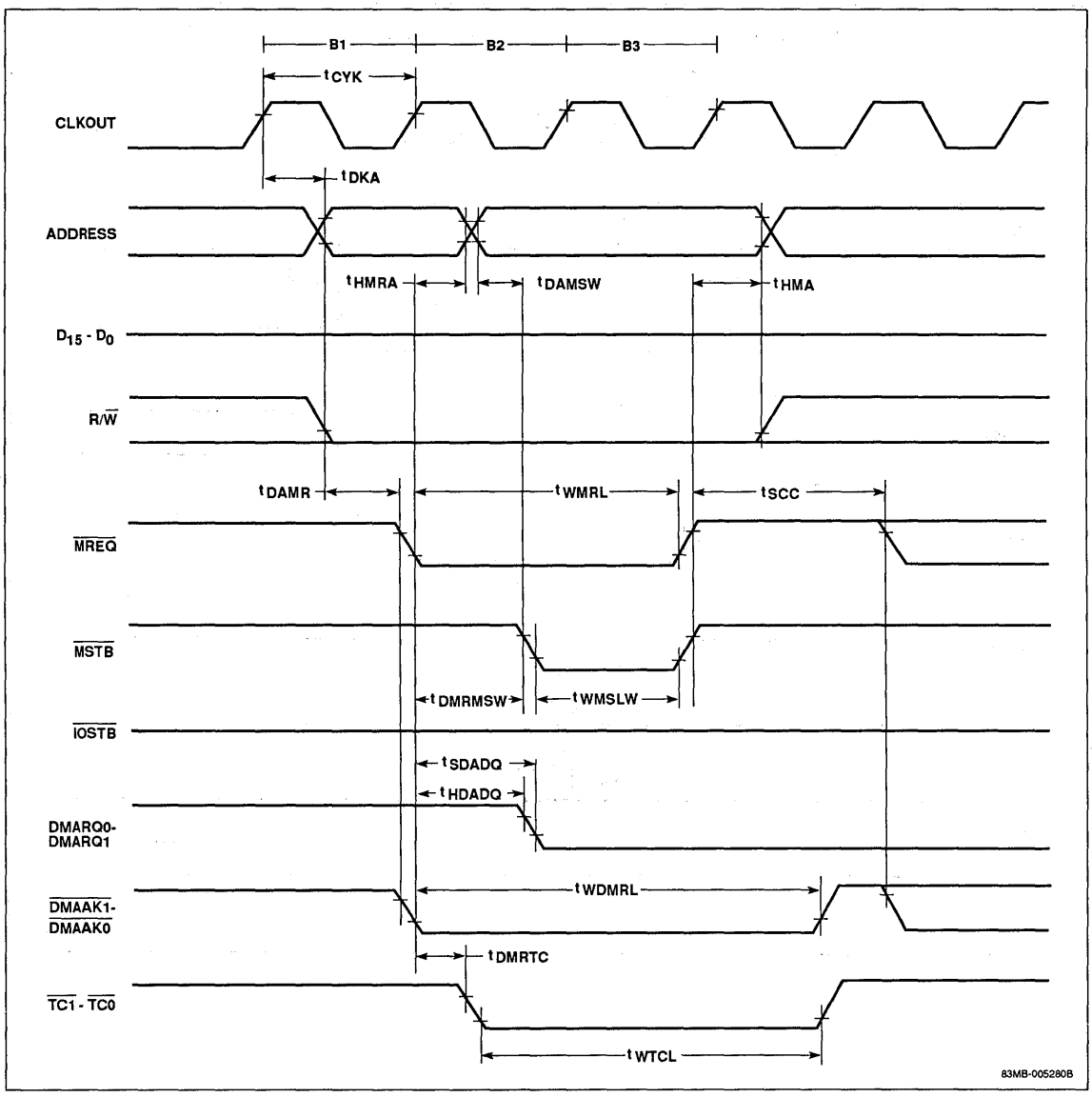

B1
B2
B3
tCYK
CLKOUT
tDKA
ADDRESS
tHMRA
tHMA
D15 - D0
tDISD
tHMD
R/W
tDMRD
tDAMR
tWMRL
tSCC
MREQ
MSTB
tDAIS
IOSTB
tDMRIS
tWISL
REFRQ
DMAAK1-
DMAAK0
83MB-005278B

I/O Write

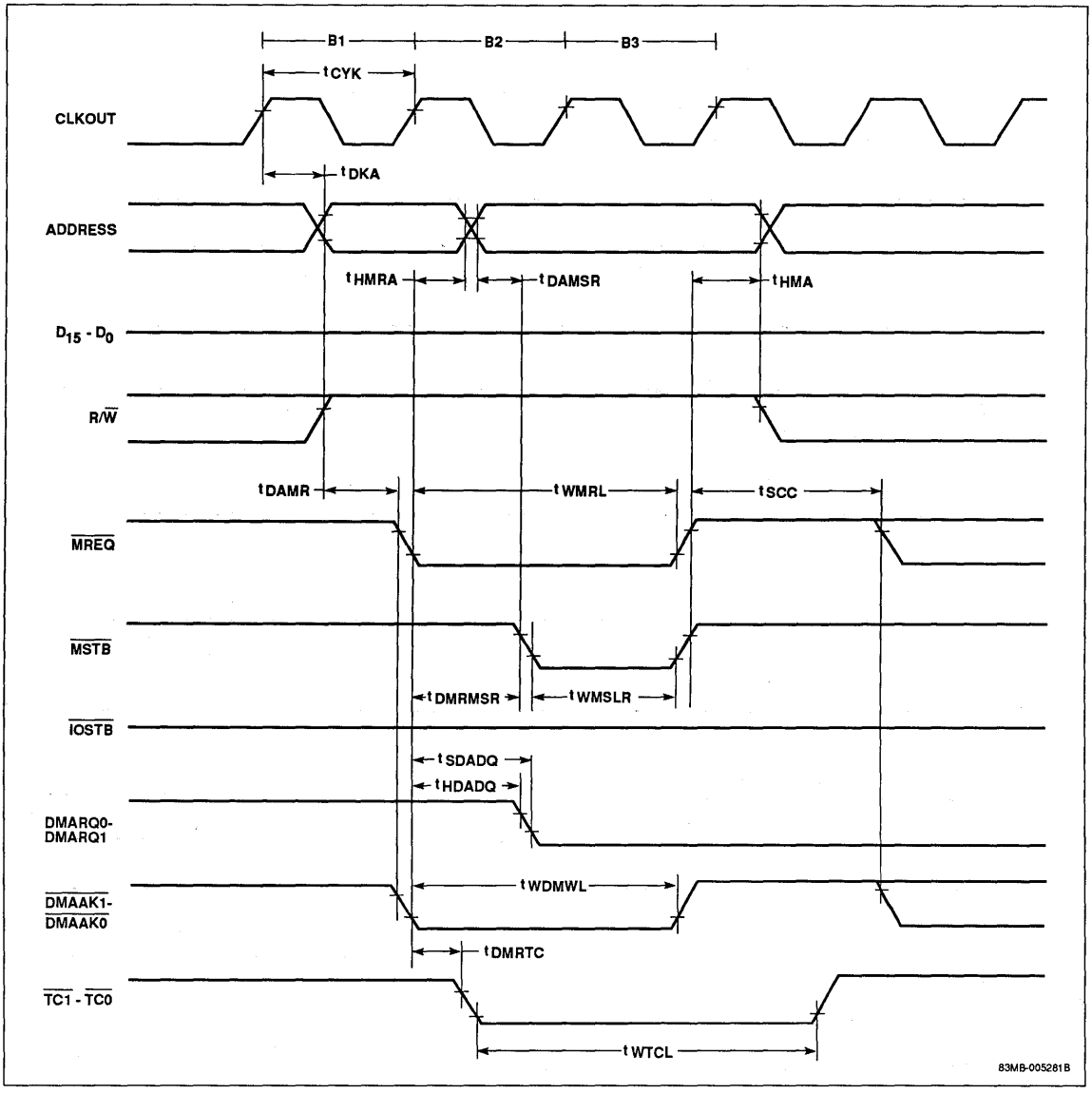

B1
B2
B3
tCYK
CLKOUT
tDKA
ADDRESS
tHMRA
tHMA
D15 - D0
tDADW
tSDM
R/W
tDAMR
tSCC
MREQ
tWMRL
MSTB
tDAIS
IOSTB
tDMRIS
tWISL
REFRQ
DMAAK1-
DMAAK0
83MB-005279B

4e

DMA, I/O to Memory

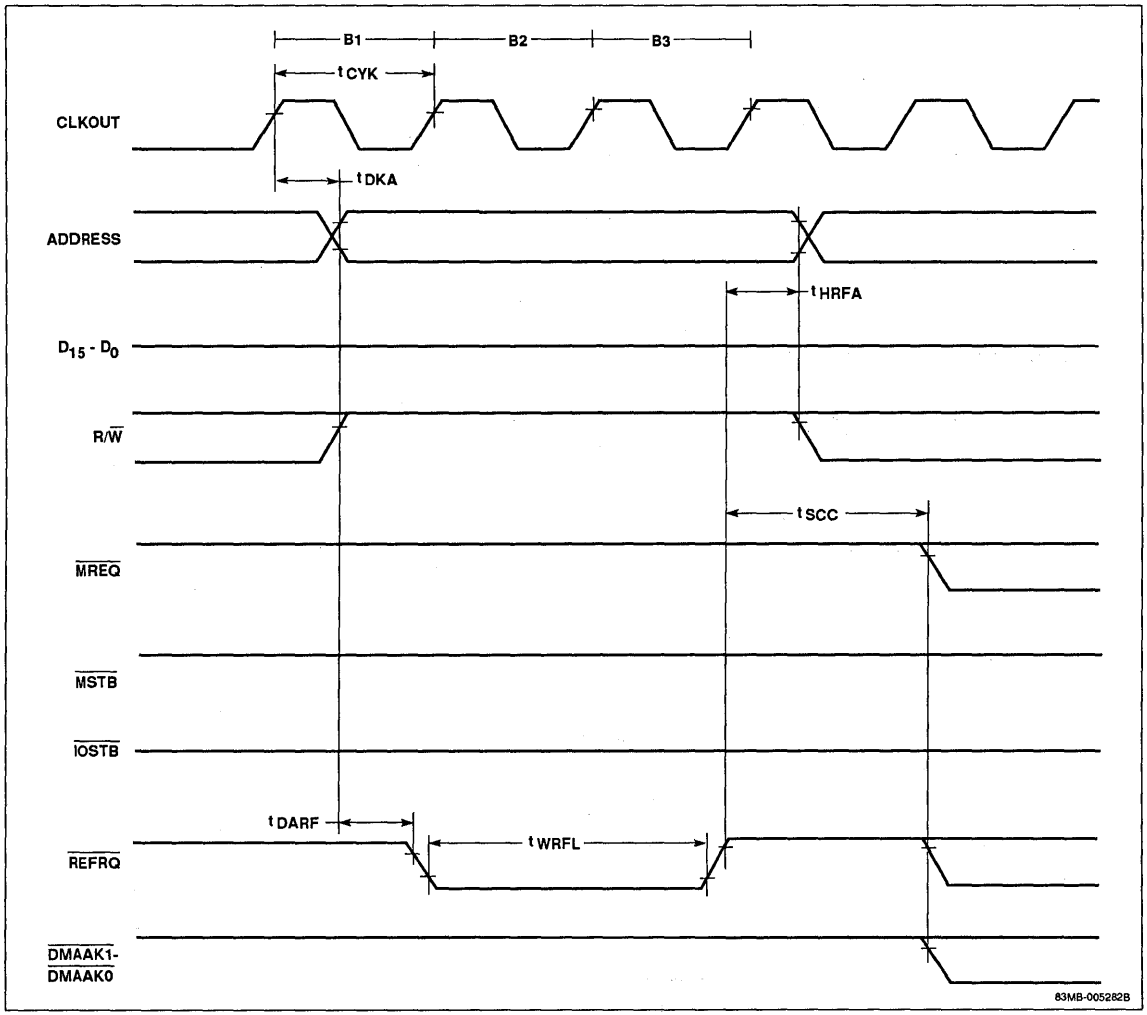


DMA, Memory to I/O

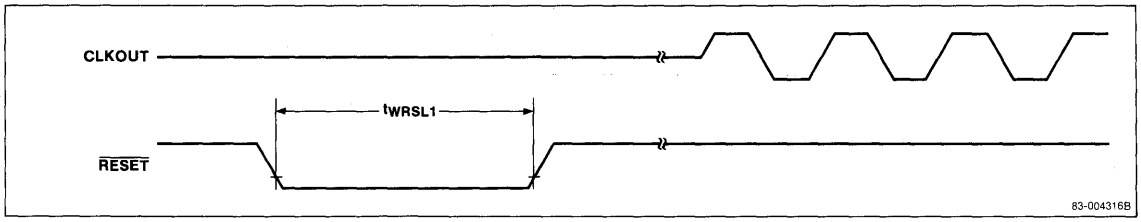


4e

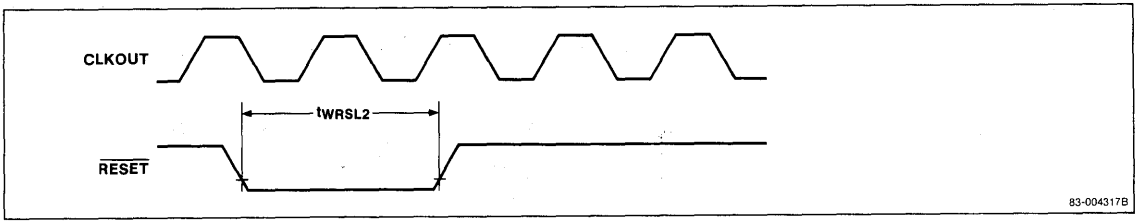
Refresh



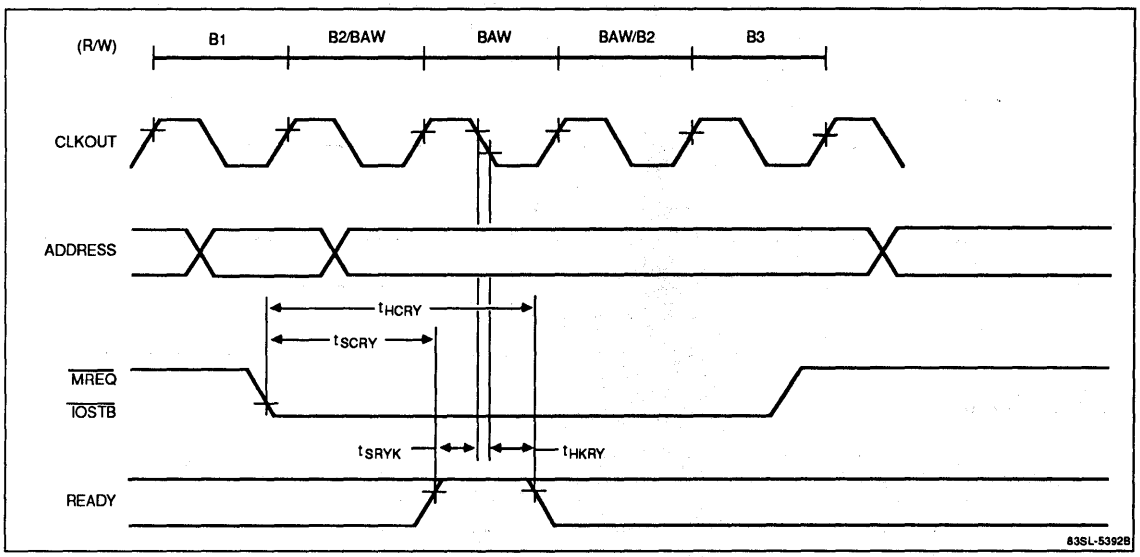
RESET 1



$\overline{\text{RESET}}$ 2

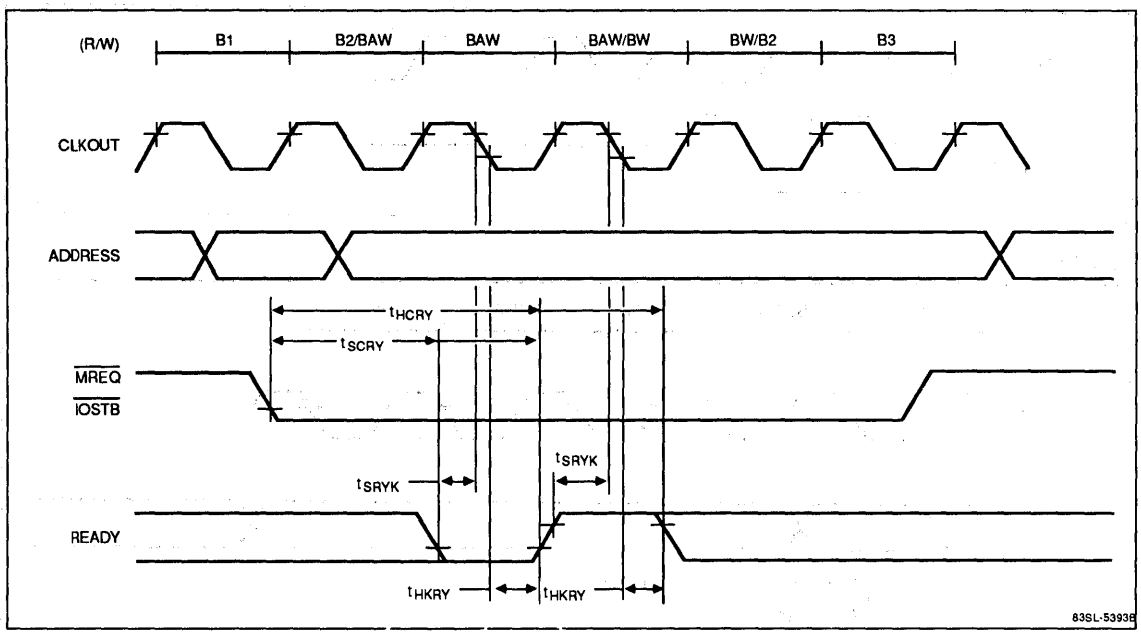


READY 1

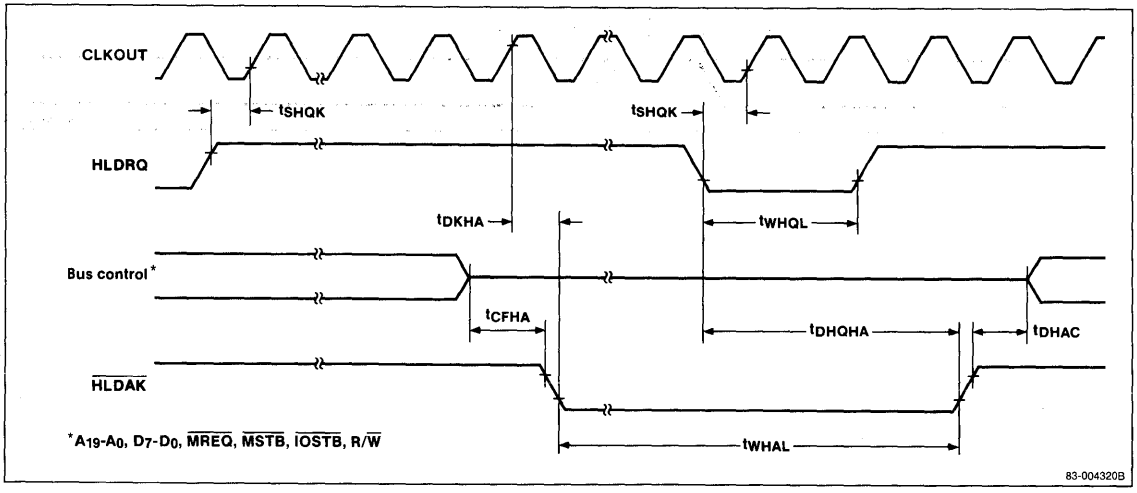


4e

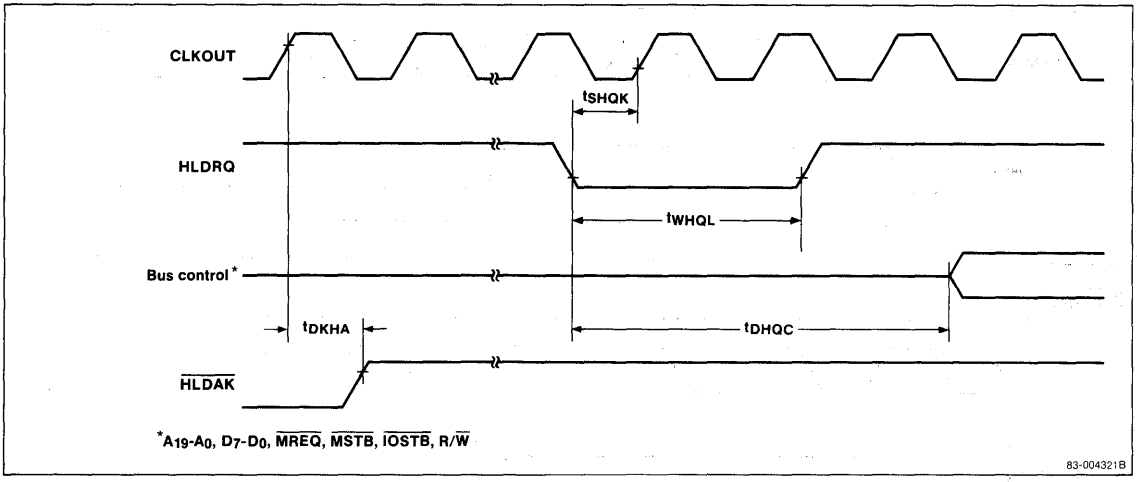
READY 2



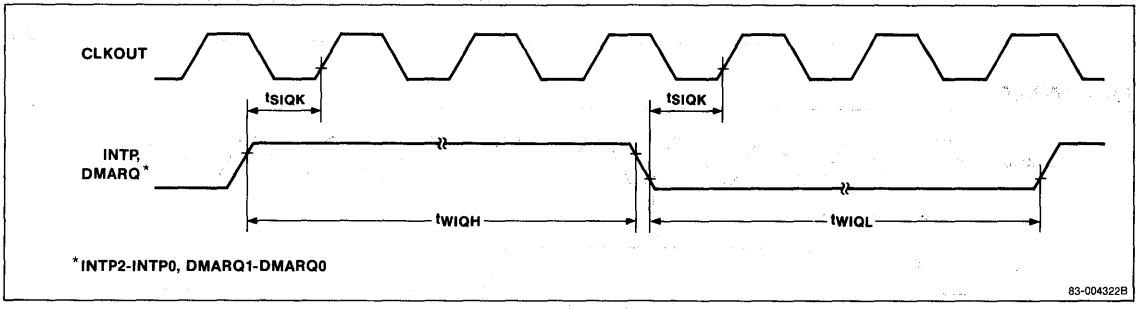
HLDRQ/ $\overline{HLDAK}$ 1



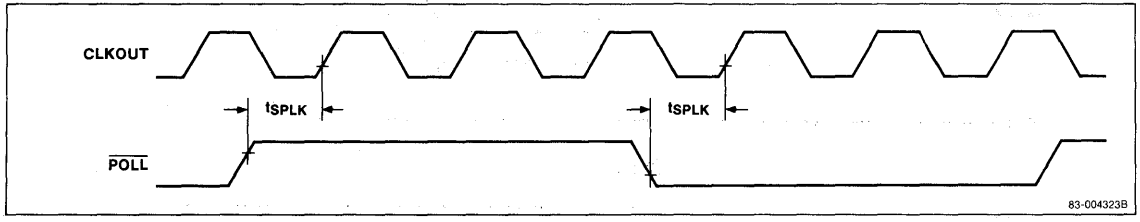
HLDRQ/$\overline{\text{HLDAK}}$ 2



INTP, DMARQ Input

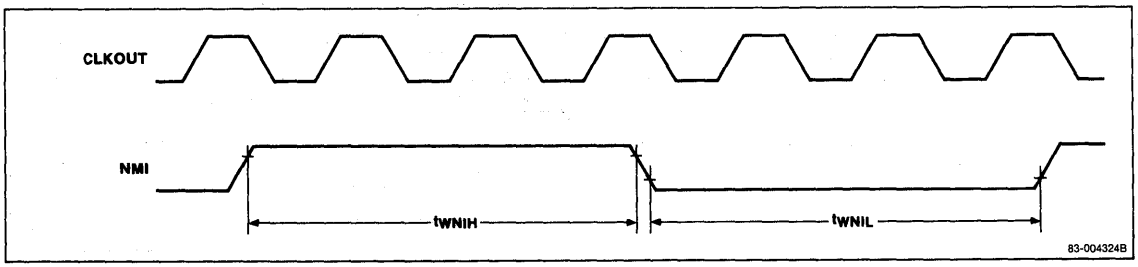


$\overline{\text{POLL}}$ Input

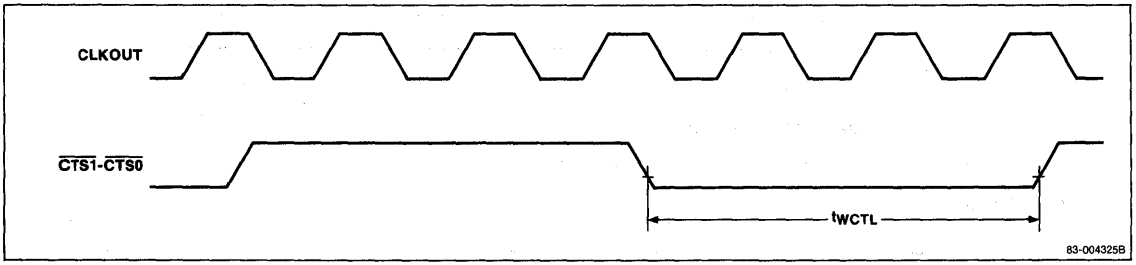


4e

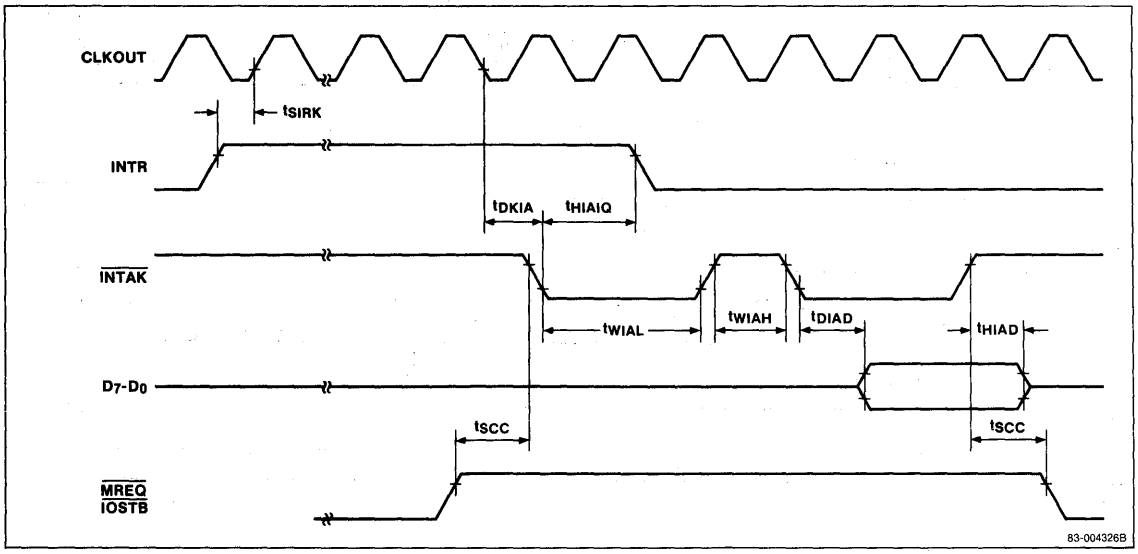
NMI Input



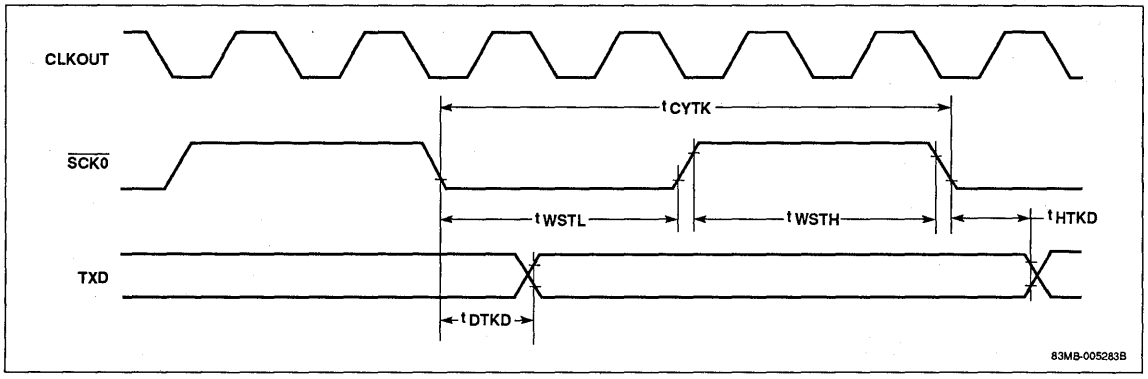
$\overline{CTS}$ Input



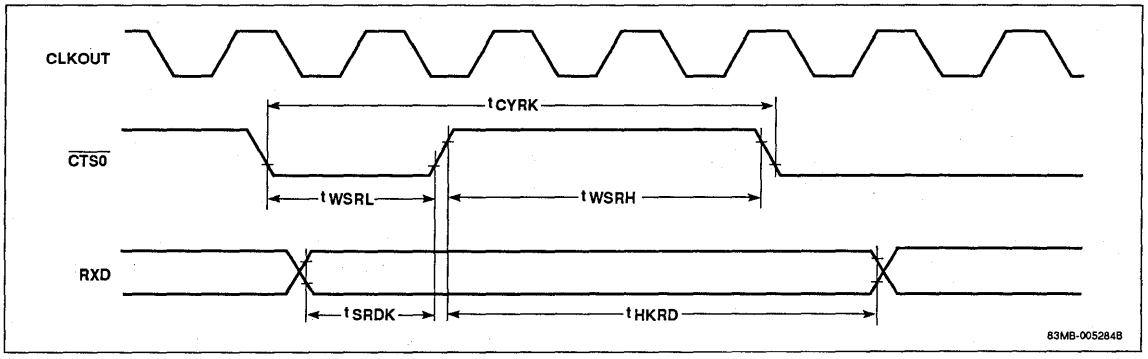
INTR/$\overline{INTAK}$



Serial Transmit



Serial Receive



4e

INSTRUCTION SET

Instructions, grouped according to function, are described in a table near the end of this data sheet. Descriptions include source code, operation, opcode, number of bytes, and flag status. Supplementary information applicable to the instruction set is contained in the following tables.

- Symbols and Abbreviations
- Flag Symbols
- 8- and 16-Bit Registers. When mod = 11, the register is specified in the operation code by the byte/word operand (W = 0/1) and reg (000 to 111).
- Segment Registers. The segment register is specified in the operation code by sreg (00, 01, 10, or 11).
- Memory Addressing. The memory addressing mode is specified in the operation code by mod (00, 01, or 10) and mem (000 through 111).
- Instruction Clock Count. This table gives formulas for calculating the number of clock cycles occupied by each type of instruction. The formulas, which depend on byte/word operand and RAM enable/disable, have variables such as EA (effective address), W (wait states), and n (iterations or string instructions).

Symbols and Abbreviations

Identifier	Description
reg	8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
dmem	8- or 16-bit direct memory location
mem	8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
sfr	8-bit special function register location
imm	Constant (0 to FFFFH)
imm16	Constant (0 to FFFFH)
imm8	Constant (0 to FFH)
imm4	Constant (0 to FH)
imm3	Constant (0 to 7)
acc	AW or AL register
sreg	Segment register
src-table	Name of 256-byte translation table
src-block	Name of block addressed by the IX register
dst-block	Name of block addressed by the IY register
near-proc	Procedure within the current program segment

Symbols and Abbreviations (cont)

Identifier	Description
far-proc	Procedure located in another program segment
near-label	Label in the current program segment
short-label	Label between −128 and +127 bytes from the end of instruction
far-label	Label in another program segment
memptr16	Word containing the offset of the memory location within the current program segment to which control is to be transferred
memptr32	Double word containing the offset and segment base address of the memory location to which control is to be transferred
regptr16	16-bit register containing the offset of the memory location within the program segment to which control is to be transferred
pop-value	Number of bytes of the stack to be discarded (0 to 64K bytes, usually even addresses)
fp-op	Immediate data to identify the instruction code of the external floating-point operation
R	Register set
W	Word/byte field (0 to 1)
reg	Register field (000 to 111)
mem	Memory field (000 to 111)
mod	Mode field (00 to 10)
S:W	When S:W = 01 or 11, data = 16 bits. At all other times, data = 8 bits.
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
AW	Accumulator (16 bits)
AH	Accumulator (high byte)
AL	Accumulator (low byte)
BP	Base pointer register (16 bits)
BW	BW register (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
CW	CW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
DW	DW register (16 bits)
DH	DW register (high byte)
DL	DW register (low byte)
SP	Stack pointer (16 bits)
PC	Program counter (16 bits)
PSW	Program status word (16 bits)
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)

Symbols and Abbreviations (cont)

Identifier	Description
PS	Program segment register (16 bits)
SS	Stack segment register (16 bits)
DS_0	Data segment 0 register (16 bits)
DS_1	Data segment 1 register (16 bits)
AC	Auxiliary carry flag
CY	Carry flag
P	Parity flag
S	Sign flag
Z	Zero flag
DIR	Direction flag
IE	Interrupt enable flag
V	Overflow flag
BRK	Break flag
MD	Mode flag
(...)	Values in parentheses are memory contents
disp	Displacement (8 or 16 bits)
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1-bit)
seg	Immediate segment data (16 bits)
offset	Immediate offset data (16 bits)
←	Transfer direction
+	Addition
−	Subtraction
x	Multiplication
÷	Division
%	Modulo
AND	Logical product
OR	Logical sum
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value

Flag Symbols

Identifier	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
X	Set or cleared according to the result
U	Undefined
R	Value saved earlier is restored

8- and 16-Bit Registers (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Registers

sreg	Register
00	DS_1
01	PS
10	SS
11	DS_0

Memory Addressing

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Instruction Clock Count

Mnemonic	Operand	Clocks
ADD	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+7+W EA+7+W
	mem8, reg8 reg16, mem16	EA+10+2W [EA+7+W] EA+10+2W [EA+7+W]
	reg8, imm8 mem16, imm16	5 6
	mem8,imm8 mem16, imm16	EA+11+2W [EA+9+2W] EA+12+2W [EA+8+2W]
	AL, imm8 AW, imm16	5 6
ADD4S		22+(30+3W)n [22+(28+3W)n]
ADDC		Same as ADD
ADJ4A		9
ADJ4S		9
ADJBA		17
ADJBS		17
AND	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+7+W EA+7+W
	mem8, reg8 mem16, reg16	EA+10+2W [EA+7+W] EA+10+2W [EA+7+W]
	reg8, imm8 reg16, imm16	5 6
	mem8, imm8 mem16, imm16	EA+11+2W [EA+9+2W] EA+12+2W [EA+8+2W]
	AL, imm8 AW, imm16	5 6
Bcond (conditional branch)		8 or 15
BCWZ		8 or 15
BR	near-label short-label	12 12
	regptr16 memptr16	13 EA+16+W
	far-label memptr32	15 EA+23+2W
BRK	3 imm8	50+5W [38+5W] 51+5W [39+5W]
BRKCS		15
BRKV		50+5W [38+5W]
BTCLR		29
BUSLOCK		2

Mnemonic	Operand	Clocks
CALL	near-proc regptr16	21+W [17+W] 21+W [17+W]
	memptr16 far-proc memptr32	EA+24+2W [EA+22+2W] 36+2W [32+2W] EA+32+4W [EA+20+4W]
CHKIND	reg16, mem32	EA+24+2W
CLR1	CY DIR	2 2
	reg8, CL reg16, CL	8 8
	mem8, CL mem16, CL	EA+16+2W [EA+13+W] EA+16+2W [EA+13+W]
	reg8, imm3 reg16, imm4	7 7
	mem8, imm3 mem16, imm4	EA+13+2W [EA+10+W] EA+13+2W [EA+9+W]
CMP	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+7+W EA+7+W
	mem8, reg8 mem16, reg16	EA+7+W EA+7+W
	reg8, imm8 reg16, imm8 reg16, imm16	5 5 6
	mem8, imm8 mem16, imm8 mem16, imm16	EA+8+W EA+9+W EA+9+W
	AL, imm8 AW, imm16	5 6
CMP4S		22+(25+2W)n
CMPBK	mem8, mem8 mem16, mem16	25+2W [21+2W] 25+2W [19+2W]
CMPBKB		16+(23+2W)n
CMPBKW		16+(23+2W)n
CMPM	mem8 mem16	18+W 19+2W
CMPMB	n > 1	16+(16+W)n
CMPMW	n > 1	16+(16+2W)n
CVTBD		19
CVTBW		3
CVTDB		20
CVTWL		8
DBNZ		8 or 17
DBNZE		8 or 17
DBNZNE		8 or 17

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
DEC	reg8 reg16	5 2
	mem8 mem16	EA+13+2W [EA+11+2W] EA+13+2W [EA+9+2W]
DI		4
DISPOSE		11+W
DIV	AW, reg8 AW, mem8	46-56 EA+49+W to EA+59+W
	DW:AW, reg16 DW: AW, mem16	54-64 EA+57+W to EA+67+W
DIVU	AW, reg8 AW, mem8	31 EA+34+W
	DW:AW, reg16 DW: AW, mem16	39 EA+43+2W
DSO:		2
DS1:		2
EI		12
EXT	reg8, reg8 reg8, imm4	41-121 42-122
FINT		2
FPO1		55+5W [43+5W]
FPO2		55+5W [43+5W]
HALT		N/A
IN	AL, imm8 AW, imm8	15+W 15+W
	AL, DW AW, DW	14+W 14+W
INC	reg8 reg16	5 2
	mem8 mem16	EA+13+2W [EA+11+2W] EA+13+2W [EA+9+2W]
INM	mem8, DW mem16, DW	21+2W [19+2W] 19+2W [15+2W]
	mem8, DW mem16, DW	18+(15+2W)n [18+(13+2W)n] 18+(13+2W)n [18+(9+2W)n]
INS	reg8, reg8 reg8, imm4	63-155 64-156
LDEA		EA+2
LDM	mem8 mem16	13+W 13+W
LDMB	n > 1	16+(11+W)n
LDMW	n > 1	16+(11+W)n

Mnemonic	Operand	Clocks
MOV	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+7+W EA+7+W
	mem8, reg8 mem16, reg16	EA+5+W [EA+2] EA+5+W [EA+2]
	reg8, imm8 reg16, imm16	5 6
	mem8, imm8 mem16, imm16	EA+6+W EA+6+W
	AL, dmem8 AW, dmem16	10+W 10+W
	dmem8, AL dmem16, AW	8+W [5] 8+W [5]
	sreg, reg16 sreg, mem16	4 EA+9+W
	reg16, sreg mem16, sreg	3 EA+6+W [EA+3]
	AH, PSW PSW, AH	2 3
	DS0, reg16, memptr32 DS1, reg16, memptr32	EA+17+2W EA+17+2W
MOVBK	mem8, mem8 mem16, mem16	22+2W [17+W] 22+2W [19+W]
MOVBKB	n > 1	16+(18+2W)n [16+(13+W)n]
MOVBKW	n > 1	16+(18+2W)n [16+(10+W)n]
MOVSPA		16
MOVSPB	reg16	11
MUL	AW, AL, reg8 AW, AL, mem8	31-40 EA+34+W to EA+43+W
	DW:AW, AW, reg16 DW:AW, AW, mem16	39-48 EA+42+W to EA+51+W
	reg16, reg16, imm8 reg16, mem16, imm8	39-49 EA+42+W to EA+52+W
	reg16, reg16, imm16 reg16, mem16, imm16	40-50 EA+43+W to EA+53+W
MULU	reg8 mem8	24 EA+27+W
	reg16 mem16	32 EA+33+W

4e

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
NEG	reg8 reg16	5 5
	mem8 mem16	EA+13+2W [EA+10+W] EA+13+2W [EA+10+W]
NOP		4
NOT	reg8 reg16	5 5
	mem8 mem16	EA+13+2W [EA+10+W] EA+13+2W [EA+10+W]
NOT1	CY	2
	reg8, CL reg16, CL	7 7
	mem8, CL mem16, CL	EA+15+W [EA+12+W] EA+15+2W [EA+12+W]
	reg8, imm3 reg16, imm4	6 6
	mem8, imm3 mem16, imm4	EA+12+2W [EA+9+W] EA+12+2W [EA+9+W]
OR	reg8, reg8 reg16, reg16	2 2
	reg8, mem8 reg16, mem16	EA+7+W EA+7+W
	mem8, reg8 mem16, reg16	EA+10+2W [EA+7+W] EA+10+2W [EA+7+W]
	reg8,imm8 reg16, imm16	5 6
	mem8, imm8 mem16, imm16	EA+11+2W [EA+9+2W] EA+12+2W [EA+8+2W]
	AL, imm8 AW, imm16	5 6
OUT	imm8, AL imm8, AW	11+W 9+W
	DW, AL DW, AW	10+W 8+W
OUTM	DW, mem8 DW, mem16	21+2W [19+2W] 19+2W [15+2W]
	DW, mem8 DW, mem16	18+(15+2W)n [18+(13+2W)n] 18+(13+2W)n [18+(9+2W)n]
POLL		N/A
POP	reg16 mem16	11+W EA+14+2W [EA+11+W]
	DS0,1 SS	12+W 12+W
	DS0 PSW	12+W 13+W
	R	74+8W [58]

Mnemonic	Operand	Clocks
PREPARE	imm16, imm8	imm8 = 0:26+W imm8 = 1:37+2W imm8 = n, n > 1:44+19 (n−1)+2Wn
PS:		2
PUSH	reg16 mem16	13+W [9+W] EA+16+2W [EA+12+2W]
	DS1 PS	10+W [7] 10+W [7]
	SS DS0	10+W [7] 10+W [7]
	PSW R	9+W [6] 74+8W [50]
	imm8 imm16	12+W [9] 13+W [10]
REP		2
REPE		2
REPZ		2
REPC		2
REPNC		2
REPNE		2
REPNZ		2
RET	null pop-value	19+W 19+W
	null pop-value	27+2W 28+2W
RETI		40+3W [34+W]
RETRBI		12
ROL	reg8 1 reg16, 1	8 8
	mem8, 1 mem16, 1	EA+16+2W [EA+13+W] EA+16+2W [EA+13+W]
	reg8, CL reg16, CL	11+2n 11+2n
	mem8, CL mem16, CL	EA+19+2W+2n [EA+16+W+2n] EA+19+2W+2n [EA+16+W+2n]
	reg8, imm8 reg16, imm8	9+2n 9+2n
	mem8, imm8 mem16, imm8	EA+15+2W+2n [EA+12+W+2n] EA+15+2W+2n [EA+12+W+2n]
ROL4	reg8 mem8	17 EA+20+2W [EA+18+2W]
ROLC		Same as ROL
ROR		Same as ROL

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
ROR4	reg8 mem8	21 EA+26+2W [EA+24+2W]
RORC		Same as ROL
SET1	CY DIR	2 2
	reg8, CL reg16, CL	7 7
	mem8, CL mem16, CL	EA+15+2W [EA+12+W] EA+15+2W [EA+12+W]
	reg8, imm3 reg16, imm4	6 6
	mem8, imm3 mem16, imm4	EA+12+2W [EA+9+W] EA+12+2W [EA+9+W]
SHL		Same as ROL
SHR		Same as ROL
SHRA		Same as ROL
SS:		2
STM	mem8 mem16	13+W [10] 13+W (10]
STMB	n > 1	16+(9+W)n [16+(7+W)n]
STMW	n > 1	16+(9+W)n [16+(5+W)n]
STOP		N/A
SUB		Same as ADD
SUB4S		22+(30+3W)n [22+(28+3W)n]
SUBC		Same as ADD
TEST	reg8, reg8 reg16, reg16	4 4
	reg8, mem8 reg16, mem16	EA+12+W EA+11+2W
	mem8, reg8 mem16, reg16	EA+12+W EA+11+2W
	reg8, imm8 reg16, imm16	7 8
	mem8, imm8 mem16, imm16	EA+9+W EA+10+W
	AL, imm8 AW, imm16	5 6

Mnemonic	Operand	Clocks
TEST1	reg8, CL reg16, CL	7 7
	mem8, CL mem16, CL	EA+12+W EA+12+W
	reg8, imm3 reg16, imm4	6 6
	mem8, imm3 mem16, imm4	EA+9+W EA+9+W
TRANS		11+W
TRANSB		11+W
TSKSW		20
XCH	reg8, reg8 reg16, reg16	3 3
	mem8, reg8/ reg8, mem8	EA+12+2W [EA+9+W]
	mem16, reg16/ reg16, mem16	EA+12+2W [EA+9+2W]
	AW, reg16 reg16, AW	4 4
XOR		Same as AND

Notes:

(1) If the number of clocks is not the same for RAM enabled and RAM disabled conditions, the RAM enabled value is listed first, followed by the RAM disabled value in brackets; for example, EA+8+2W [EA+6+W]

(2) Symbols in the Clocks column are defined as follows.
EA = additional clock cycles required for calculation of the effective address
= 3 (mod 00 or 01) or 4 (mod 10)
W = number of wait states selected by the WTC register
n = number of iterations or string instructions

4e

Instruction Clock Count for Operations

	Byte		Word	
	RAM Enable	RAM Disable	RAM Enable	RAM Disable
Context switch interrupt	—	—	33	33
DMA (Single-step mode)	6+2W	6+2W	6+2W	6+2W
DMA (Demand release mode)	3+W	3+W	3+W	3+W
DMA (Burst mode)	(6+2W)n	(6+2W)n	(6+2W)n	(6+2W)n
DMA (Single-transfer mode)	3+W	3+W	3+W	3+W
Interrupt (INT pin)	—	—	57+3W	57+3W
Macro service, sfr ← mem	31+W	26+W	31+W	26+W
Macro service, mem ← sfr	28+W	27+W	28+W	27+W
Macro service (Search char mode), sfr ← mem	34+W	34+W	—	—
Macro service (Search char mode), mem ← sfr	44+W	41+W	—	—
Priority interrupt (Vectored mode)	—	—	55+5W	55+5W
NMI (Vectored mode)	—	—	53+5W	53+5W

W = number of wait states inserted into external bus cycle
n = number of iterations
N = number of clocks to complete the instruction currently executing

Bus Controller Latency

		Clocks	
Latnecy	Mode	Typ	Max
Hold request	Refresh active		9+3W
	Intack active		10+2W
	No refresh or intack		7+2W
DMA request (Notes 1, 2)	Burst	3	14+2W
	Single-step	3	14+2W
	Demand releas	3	14+2W
	Single-transfer	4	14+2W

Notes:

(1) The listed DMA latency times are the maximum number of clocks when a DMA request is asserted until $\overline{\text{DMAAK}}$ or $\overline{\text{MREQ}}$ goes low in the corresponding DMA cycle.

(2) The test conditions are: no wait states, no interrupts, no macroservice requests, and no hold requests.

Interrupt Latency

	Clocks	
Source	Typ	Max
NMI pin	12+N	18+N
INT pin	8+N	8+N
All others	27+N	15+N

Instruction Set

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer																	
MOV	reg, reg	reg ← reg	1	0	0	0	1	0	1	W	2						
			1	1		reg			reg								
	mem, reg	(mem) ← reg	1	0	0	0	1	0	0	W	2-4						
			mod			reg			mem								
	reg, mem	reg ← (mem)	1	0	0	0	1	0	1	W	2-4						
			mod			reg			mem								
	mem, imm	(mem) ← imm	1	1	0	0	0	1	1	W	3-6						
			mod		0	0	0		mem								
	reg, imm	reg ← imm	1	0	1	1	W		reg		2-3						
	acc, dmem	When W = 0: AL ← (dmem) When W = 1: AH ← (dmem + 1), AL ← (dmem)	1	0	1	0	0	0	0	W	3						
	dmem, acc	When W = 0: (dmem) ← AL When W = 1: (dmem + 1) ← AH, (dmem) ← AL	1	0	1	0	0	0	1	W	3						
	sreg, reg16	sreg ← reg16 sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	2						
			1	1	0	sreg			reg								
	sreg, mem16	sreg ← (mem16) sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	2-4						
			mod		0	sreg			mem								
	reg16, sreg	reg16 ← sreg	1	0	0	0	1	1	0	0	2						
			1	1	0	sreg			reg								
	mem16, sreg	(mem16) ← sreg	1	0	0	0	1	1	0	0	2-4						
			mod		0	sreg			mem								
	DS0, reg16, mem32	reg16 ← (mem32), DS0 ← (mem32 + 2)	1	1	0	0	0	1	0	1	2-4						
			mod			reg			mem								
	DS1, reg16, mem32	reg16 ← (mem32), DS1 ← (mem32 + 2)	1	1	0	0	0	1	0	0	2-4						
			mod			reg			mem								
	AH, PSW	AH ← S, Z, x, AC, x, P, x, CY	1	0	0	1	1	1	1	1	1						
	PSW, AH	S, Z, x, AC, x, P, x, CY ← AH	1	0	0	1	1	1	1	0	1	x	x		x	x	x
LDEA	reg16, mem16	reg16 ← mem16	1	0	0	0	1	1	0	1	2-4						
			mod			reg			mem								
TRANS	src-table	AL ← (BW + AL)	1	1	0	1	0	1	1	1	1						
XCH	reg, reg	reg ↔ reg	1	0	0	0	0	1	1	W	2						
			1	1		reg			reg								
	mem, reg or reg, mem	(mem) ↔ reg	1	0	0	0	0	1	1	W	2-4						
			mod			reg			mem								
	AW, reg16 or reg16, AW	AW ↔ reg16	1	0	0	1	0		reg		1						

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Repeat Prefixes																	
REPC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (–1). If there is a waiting interrupt, it is processed. When CY ≠ 1, exit the loop.	0	1	1	0	0	1	0	1	1						
REPNC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (–1). If there is a waiting interrupt, it is processed. When CY ≠ 0, exit the loop.	0	1	1	0	0	1	0	0	1						
REP REPE REPZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (–1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 1, exit the loop.	1	1	1	1	0	0	1	1	1						
REPNE REPNZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (–1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 0, exit the loop.	1	1	1	1	0	0	1	0	1						
Primitive Block Transfer																	
MOVBK	dst-block, src-block	When W = 0: (IY) ← (IX) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX – 1, IY ← IY – 1 When W = 1: (IY + 1, IY) ← (IX + 1, IX) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX – 2, IY ← IY – 2	1	0	1	0	0	1	0	W	1						
CMPBK	src-block, dst-block	When W = 0: (IX) – (IY) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX – 1, IY ← IY – 1 When W = 1: (IX + 1, IX) – (IY + 1, IY) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX – 2, IY ← IY – 2	1	0	1	0	0	1	1	W	1	x	x	x	x	x	x
CMPM	dst-block	When W = 0: AL – (IY) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY – 1 When W = 1: AW – (IY + 1, IY) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY – 2	1	0	1	0	1	1	1	W	1	x	x	x	x	x	x
LDM	src-block	When W = 0: AL ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX – 1 When W = 1: AW ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX – 2	1	0	1	0	1	1	0	W	1						
STM	dst-block	When W = 0: (IY) ← AL DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY – 1 When W = 1: (IY + 1, IY) ← AW DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY – 2	1	0	1	0	1	0	1	W	1						

Instruction Set (cont)

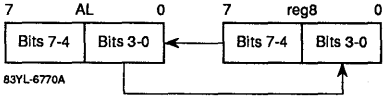
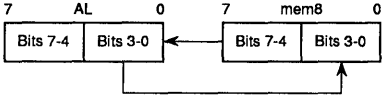
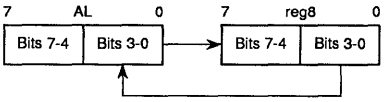
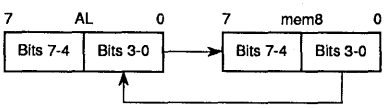
Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Field Transfer																	
INS	reg8, reg8	16-bit field ← AW	0	0	0	0	1	1	1	1	3						
			0	0	1	1	0	0	0	1							
			1	1		reg			reg								
	reg8, imm4	16-bit field ← AW	0	0	0	0	1	1	1	1	4						
			0	0	1	1	1	0	0	1							
			1	1	0	0	0		reg								
EXT	reg8, reg8	AW ← 16-bit field	0	0	0	0	1	1	1	1	3						
			0	0	1	1	0	0	1	1							
			1	1		reg			reg								
	reg8, imm4	AW ← 16-bit field	0	0	0	0	1	1	1	1	4						
			0	0	1	1	1	0	1	1							
			1	1	0	0	0		reg								
I/O																	
IN	acc, imm8	When W = 0: AL ← (imm8) When W = 1: AH ← (imm8 + 1), AL ← (imm8)	1	1	1	0	0	1	0	W	2						
	acc, DW	When W = 0: AL ← (DW) When W = 1: AH ← (DW + 1), AL ← (DW)	1	1	1	0	1	1	0	W	1						
OUT	imm8, acc	When W = 0: (imm8) ← AL When W = 1: (imm8 + 1) ← AH, (imm8) ← AL	1	1	1	0	0	1	1	W	2						
	DW, acc	When W = 0: (DW) ← AL When W = 1: (DW + 1) ← AH, (DW) ← AL	1	1	1	0	1	1	1	W	1						
Primitive Block I/O Transfer																	
INM	dst-block, DW	When W = 0: (IY) ← (DW) DIR = 0: IY ← IY + 1 DIR = 1: IY ← IY − 1 When W = 1: (IY + 1, IY) ← (DW + 1, DW) DIR = 0: IY ← IY + 2 DIR = 1: IY ← IY − 2	0	1	1	0	1	1	0	W	1						
OUTM	DW, src-block	When W = 0: (DW) ← (IX) DIR = 0: IX ← IX + 1 DIR = 1: IX ← IX − 1 When W = 1: (DW + 1, DW) ← (IX + 1, IX) DIR = 0: IX ← IX + 2 DIR = 1: IX ← IX − 2	0	1	1	0	1	1	1	W	1						

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Addition/Subtraction																	
ADD	reg, reg	reg ← reg + reg	0	0	0	0	0	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) + reg	0	0	0	0	0	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg + (mem)	0	0	0	0	0	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg + imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	0	0		reg								
	mem, imm	(mem) ← (mem) + imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	0	0		mem								
	acc, imm	When W = 0: AL ← AL + imm When W = 1: AW ← AW + imm	0	0	0	0	0	1	0	W	2-3	x	x	x	x	x	x
ADDC	reg, reg	reg ← reg + reg + CY	0	0	0	1	0	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) + reg + CY	0	0	0	1	0	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg + (mem) + CY	0	0	0	1	0	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg + imm + CY	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	1	0		reg								
	mem, imm	(mem) ← (mem) + imm + CY	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	1	0		mem								
	acc, imm	When W = 0: AL ← AL + imm + CY When W = 1: AW ← AW + imm + CY	0	0	0	1	0	1	0	W	2-3	x	x	x	x	x	x
SUB	reg, reg	reg ← reg − reg	0	0	1	0	1	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) − reg	0	0	1	0	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg − (mem)	0	0	1	0	1	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg − imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	1	0	1		reg								
	mem, imm	(mem) ← (mem) − imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		1	0	1		mem								
	acc, imm	When W = 0: AL ← AL − imm When W = 1: AW ← AW − imm	0	0	1	0	1	1	0	W	2-3	x	x	x	x	x	x
SUBC	reg, reg	reg ← reg−reg−CY	0	0	0	1	1	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem)−reg−CY	0	0	0	1	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg−(mem)−CY	0	0	0	1	1	0	1	W	2-4	x	x	x	x	x	
			mod			reg			mem								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Addition/Subtraction (cont)																	
SUBC	reg, imm	reg ← reg − imm − CY	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	1	1	reg									
	mem, imm	(mem) ← (mem) − imm − CY	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	1	1	mem									
	acc, imm	When W = 0: AL ← AL − imm − CY When W = 1: AW ← AW − imm − CY	0	0	0	1	1	1	0	W	2-3	x	x	x	x	x	x
BCD Operation																	
ADD4S		dst BCD string ← dst BCD string + src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	0	0	0							
SUB4S		dst BCD string ← dst BCD string − src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	0	1	0							
CMP4S		dst BCD string − src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	1	1	0							
ROL4	reg 8		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	0	0	0							
			1	1	0	0	0	reg									
	mem 8		0	0	0	0	1	1	1	1	3-5						
			0	0	1	0	1	0	0	0							
			mod		0	0	0	mem									
ROR4	reg 8		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	0	1	0							
			1	1	0	0	0	reg									
	mem 8		0	0	0	0	1	1	1	1	3-5						
			0	0	1	0	1	0	1	0							
			mod		0	0	0	mem									
BCD Adjust																	
ADJBA		When (AL AND 0FH) > 9 or AC = 1: AL ← AL + 6, AH ← AH + 1, AC ← 1, CY ← AC, AL ← AL AND 0FH	0	0	1	1	0	1	1	1	1	x	x	u	u	u	u
ADJ4A		When (AL AND 0FH) > 9 or AC = 1: AL ← AL + 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY = 1: AL ← AL + 60H, CY ← 1	0	0	1	0	0	1	1	1	1	x	x	u	x	x	x
ADJBS		When (AL AND 0FH) > 9 or AC = 1: CY ← AC, AL ← AL AND 0FH	0	0	1	1	1	1	1	1	1	x	x	u	u	u	u
ADJ4S		When (AL AND 0FH) > 9 or AC = 1: AL ← AL − 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY = 1: AL ← AL + 60H, CY ← 1	0	0	1	0	1	1	1	1	1	x	x	u	x	x	x

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Increment/Decrement																	
INC	reg8	reg8 ← reg8 + 1	1	1	1	1	1	1	1	0	2	x		x	x	x	x
			1	1	0	0	0	reg									
	mem	(mem) ← (mem) + 1	1	1	1	1	1	1	1	W	2-4	x		x	x	x	x
			mod		0	0	0	mem									
	reg16	reg16 ← reg16 + 1	0	1	0	0	0	reg			1	x		x	x	x	x
DEC	reg8	reg8 ← reg8 − 1	1	1	1	1	1	1	1	0	2	x		x	x	x	x
			1	1	0	0	1	reg									
	mem	(mem) ← (mem) − 1	1	1	1	1	1	1	1	W	2-4	x		x	x	x	x
			mod		0	0	1	mem									
	reg16	reg16 ← reg16 − 1	0	1	0	0	1	reg			1	x		x	x	x	x
Multiplication																	
MULU	reg8	AW ← AL x reg8 AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	2	u	x	x	u	u	u
			1	1	1	0	0	reg									
	mem8	AW ← AL x (mem8) AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	2-4	u	x	x	u	u	u
			mod		1	0	0	mem									
	reg16	DW, AW ← AW x reg16 DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	2	u	x	x	u	u	u
			1	1	1	0	0	reg									
	mem16	DW, AW ← AW x (mem16) DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	2-4	u	x	x	u	u	u
			mod		1	0	0	mem									
MUL	reg8	AW ← AL x reg8 AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	2	u	x	x	u	u	u
			1	1	1	0	1	reg									
	mem8	AW ← AL x (mem8) AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	2-4	u	x	x	u	u	u
			mod		1	0	1	mem									
	reg16	DW, AW ← AW x reg16 DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	2	u	x	x	u	u	u
			1	1	1	0	1	reg									
	mem16	DW, AW ← AW x (mem16) DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	2-4	u	x	x	u	u	u
			mod		1	0	1	mem									
	reg16, reg16, imm8	reg16 ← reg16 x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	3	u	x	x	u	u	u
			1	1	reg			reg									
	reg16, mem16, imm8	reg16 ← (mem16) x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	3-5	u	x	x	u	u	u
			mod		reg			mem									
	reg16, reg16, imm16	reg16 ← reg16 x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	4	u	x	x	u	u	u
			1	1	reg			reg									
	reg16, mem16, imm16	reg16 ← (mem16) x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	4-6	u	x	x	u	u	u
			mod		reg			mem									

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Unsigned Division																	
DIVU	reg8	temp ← AW When temp ÷ reg8 > FFH: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	2	u	u	u	u	u	u
			1	1	1	1	0	reg									
	mem8	temp ← AW When temp ÷ (mem8) > FFH: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % (mem8), AL ← temp ÷ (mem8)	1	1	1	1	0	1	1	0	2-4	u	u	u	u	u	u
			mod		1	1	0	mem									
	reg16	temp ← AW When temp ÷ reg16 > FFFFH: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % reg16, AL ← temp ÷ reg16	1	1	1	1	0	1	1	1	2	u	u	u	u	u	u
			1	1	1	1	0	reg									
	mem16	temp ← AW When temp ÷ (mem16) > FFFFH: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % (mem16), AL ← temp ÷ (mem16)	1	1	1	1	0	1	1	1	2-4	u	u	u	u	u	u
			mod		1	1	0	mem									
Signed Division																	
DIV	reg8	temp ← AW When temp ÷ reg8 > 0 and temp ÷ reg8 > 7FH or temp ÷ reg8 < 0 and temp ÷ reg8 < 0−7FH−1: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	2	u	u	u	u	u	u
			1	1	1	1	1	reg									
	mem8	temp ← AW When temp ÷ (mem8) > 0 and (mem8) > 7 FH or temp ÷ (mem8) < 0 and temp ÷ (mem8) < 0−7FH−1: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % (mem8), AL ← temp ÷ (mem8)	1	1	1	1	0	1	1	0	2-4	u	u	u	u	u	u
			mod		1	1	1	mem									

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Signed Division (cont)																	
DIV	reg16	temp ← DW, AW When temp ÷ reg16 > 0 and reg16 > 7FFFH or temp ÷ reg16 < 0−7FFFH−1: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp % reg16, AL ← temp ÷ reg16	1	1	1	1	0	1	1	1	2	u	u	u	u	u	u
			1	1	1	1	1		reg								
	mem16	temp ← DW, AW When temp ÷ (mem16) > 0 and (mem16) > 7 FFFH or temp ÷ (mem16) < 0 and temp ÷ (mem16) < 0−7FFFH−1: (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times: AH ← temp%(mem16), AL ← temp ÷ (mem16)	1	1	1	1	0	1	1	1	2-4	u	u	u	u	u	u
			mod		1	1	1		mem								
Data Conversion																	
CVTBD		AH ← AL ÷ 0AH, AL ← AL % 0AH	1	1	0	1	0	1	0	0	2	u	u	u	x	x	x
			0	0	0	0	1	0	1	0							
CVTDB		AH ← 0, AL ← AH x 0AH + AL	1	1	0	1	0	1	0	1	2	u	u	u	x	x	x
			0	0	0	0	1	0	1	0							
CVTBW		When AL < 80H: AH ← 0 All other times: AH ← FFH	1	0	0	1	1	0	0	0	1						
CVTWL		When AL < 8000H: DW ← 0 All other times: DW ← FFFFH	1	0	0	1	1	0	0	1	1						
Comparison																	
CMP	reg, reg	reg − reg	0	0	1	1	1	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) − reg	0	0	1	1	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg − (mem)	0	0	1	1	1	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg − imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	1	1	1		reg								
	mem, imm	(mem) − imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		1	1	1		mem								
	acc, imm	When W = 0: AL − imm When W = 1: AW − imm	0	0	1	1	1	1	0	W	2-3	x	x	x	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Complement																	
NOT	reg	reg ← $\overline{\text{reg}}$	1	1	1	1	0	1	1	W	2						
			1	1	0	1	0		reg								
	mem	(mem) ← ($\overline{\text{mem}}$)	1	1	1	1	0	1	1	W	2-4						
			mod		0	1	0		mem								
NEG	reg	reg ← $\overline{\text{reg}}$ + 1	1	1	1	1	0	1	1	W	2	x	x	x	x	x	x
			1	1	0	1	1		reg								
	mem	(mem) ← ($\overline{\text{mem}}$) + 1	1	1	1	1	0	1	1	W	2-4	x	x	x	x	x	x
			mod		0	1	1		mem								
Logical Operation																	
TEST	reg, reg	reg AND reg	1	0	0	0	0	1	0	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg or reg, mem	(mem) AND reg	1	0	0	0	0	1	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg AND imm	1	1	1	1	0	1	1	W	3-4	u	0	0	x	x	x
			1	1	0	0	0		reg								
	mem, imm	(mem) AND imm	1	1	1	1	0	1	1	W	3-6	u	0	0	x	x	x
			mod		0	0	0		mem								
	acc, imm	When W = 0: AL AND imm8 When W = 1: AW AND imm8	1	0	1	0	1	0	0	W	2-3	u	0	0	x	x	x
AND	reg, reg	reg ← reg AND reg	0	0	1	0	0	0	1	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) AND reg	0	0	1	0	0	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg AND (mem)	0	0	1	0	0	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg AND imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	1	0	0		reg								
	mem, imm	(mem) ← (mem) AND imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod		1	0	0		mem								
	acc, imm	When W = 0: AL ← AL AND imm8 When W = 1: AW ← AW AND imm16	0	0	1	0	0	1	0	W	2-3	u	0	0	x	x	x

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Logical Operation (cont)																	
OR	reg, reg	reg ← reg OR reg	0	0	0	0	1	0	1	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) OR reg	0	0	0	0	1	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg OR (mem)	0	0	0	0	1	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg OR imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	0	0	1		reg								
	mem, imm	(mem) ← (mem) OR imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod		0	0	1		mem								
	acc, imm	When W = 0: AL ← AL OR imm8 When W = 1: AW ← AW OR imm16	0	0	0	0	1	1	0	W	2-3	u	0	0	x	x	x
XOR	reg, reg	reg ← reg XOR reg	0	0	1	1	0	0	1	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) XOR reg	0	0	1	1	0	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg XOR (mem)	0	0	1	1	0	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg XOR imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	1	1	0		reg								
	mem, imm	(mem) ← (mem) XOR imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod		1	1	0		mem								
	acc, imm	When W = 0: AL ← AL XOR imm8 When W = 1: AW ← AW XOR imm16	0	0	1	1	0	1	0	W	2-3	u	0	0	x	x	x
Bit Operation																	
TEST1	reg8, CL	reg8 bit no. CL = 0: Z ← 1 reg8 bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3	u	0	0	u	u	x
			0	0	0	1	0	0	0	0							
			1	1	0	0	0		reg								
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3-5	u	0	0	u	u	x
			0	0	0	1	0	0	0	0							
			mod		0	0	0		mem								
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3	u	0	0	u	u	x
			0	0	0	1	0	0	0	1							
			1	1	0	0	0		reg								
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3-5	u	0	0	u	u	x
			0	0	0	1	0	0	0	1							
			mod		0	0	0		mem								
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0	0	0	0	1	1	1	1	4	u	0	0	u	u	x
			0	0	0	1	1	0	0	0							
			1	1	0	0	0		reg								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
TEST1	mem8, imm3	(mem8) bit no. imm3 = 0: Z ← 1 (mem8) bit no. imm3 = 1: Z ← 0	0	0	0	0	1	1	1	1	4-6	u	0	0	u	u	x
			0	0	0	1	1	0	0	0							
			mod		0	0	0		mem								
	reg16, imm4	reg16 bit no. imm4 = 0: Z ← 1 reg16 bit no. imm4 = 1: Z ← 0	0	0	0	0	1	1	1	1	4	u	0	0	u	u	x
			0	0	0	1	1	0	0	1							
			1	1	0	0	0		reg								
	mem16, imm4	(mem16) bit no. imm4 = 0: Z ← 1 (mem16) bit no. imm4 = 1: Z ← 0	0	0	0	0	1	1	1	1	4-6	u	0	0	u	u	x
			0	0	0	1	1	0	0	1							
			mod		0	0	0		mem								
NOT1	reg8, CL	reg8 bit no. CL ← $\overline{\text{reg8 bit no. CL}}$	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	1	0							
			1	1	0	0	0		reg								
	mem8, CL	(mem8) bit no. CL ← $\overline{\text{(mem8) bit no. CL}}$	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	1	0							
			mod		0	0	0		mem								
	reg16, CL	reg16 bit no. CL ← $\overline{\text{reg16 bit no. CL}}$	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	1	1							
			1	1	0	0	0		reg								
	mem16, CL	(mem16) bit no. CL ← $\overline{\text{(mem16) bit no. CL}}$	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	1	1							
			mod		0	0	0		mem								
	reg8, imm3	reg8 bit no. imm3 ← $\overline{\text{reg8 bit no. imm3}}$	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	1	0							
			1	1	0	0	0		reg								
	mem8, imm3	(mem8) bit no. imm3 ← $\overline{\text{(mem8) bit no. imm3}}$	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	1	0							
			mod		0	0	0		mem								
	reg16, imm4	reg16 bit no. imm4 ← $\overline{\text{reg16 bit no. imm4}}$	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	1	1							
			1	1	0	0	0		reg								
	mem16, imm4	(mem16) bit no. imm4 ← $\overline{\text{(mem16) bit no. imm4}}$	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	1	1							
			mod		0	0	0		mem								
	CY	CY ← CY	1	1	1	1	0	1	0	1	1		x				

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
CLR1	reg8, CL	reg8 bit no. CL ← 0	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	0	1	0							
			1	1	0	0	0	reg									
	mem8, CL	(mem8) bit no. CL ← 0	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	0	1	0							
			mod		0	0	0	mem									
	reg16, CL	reg16 bit no. CL ← 0	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	0	1	1							
			1	1	0	0	0	reg									
	mem16, CL	(mem16) bit no. CL ← 0	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	0	1	1							
			mod		0	0	0	mem									
	reg8, imm3	reg8 bit no. imm3 ← 0	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	0	1	0							
			1	1	0	0	0	reg									
	mem8, imm3	(mem8) bit no. imm3 ← 0	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	0	1	0							
			mod		0	0	0	mem									
	reg16, imm4	reg16 bit no. imm4 ← 0	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	0	1	1							
			1	1	0	0	0	reg									
	mem16, imm4	(mem16) bit no. imm4 ← 0	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	0	1	1							
			mod		0	0	0	mem									
	CY	CY ← 0	1	1	1	1	1	0	0	0	1		0				
	DIR	DIR ← 0	1	1	1	1	1	1	0	0	1						
SET1	reg8, CL	reg8 bit no. CL ← 1	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	0	0							
			1	1	0	0	0	reg									
	mem8, CL	(mem8) bit no. CL ← 1	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	0	0							
			mod		0	0	0	mem									
	reg16, CL	reg16 bit no. CL ← 1	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	0	1							
			1	1	0	0	0	reg									
	mem16, CL	(mem16) bit no. CL ← 1	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	0	1							
			mod		0	0	0	mem									

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
SET1	reg8, imm3	reg8 bit no. imm3 ← 1	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	0	0							
			1	1	0	0	0	reg									
	mem8, imm3	(mem8) bit no. imm3 ← 1	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	0	0							
			mod		0	0	0	mem									
	reg16, imm4	reg16 bit no. imm4 ← 1	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	0	1							
			1	1	0	0	0	reg									
	mem16, imm4	(mem16) bit no. imm4 ← 1	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	0	1							
			mod		0	0	0	mem									
	CY	CY ← 1	1	1	1	1	1	0	0	1	1		1				
	DIR	DIR ← 1	1	1	1	1	1	1	0	1	1						
Shift																	
SHL	reg, 1	CY ← MSB of reg, reg ← reg x 2 When MSB of reg ≠ CY, V ← 1 When MSB of reg = CY, V ← 0	1	1	0	1	0	0	0	W	2	u	x	x	x	x	x
			1	1	1	0	0	reg									
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 When MSB of (mem) ≠ CY, V ← 1 When MSB of (mem) = CY, V ← 0	1	1	0	1	0	0	0	W	2-4	u	x	x	x	x	x
			mod		1	0	0	mem									
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1	1	0	1	0	0	1	W	2	u	x	u	x	x	x
			1	1	1	0	0	reg									
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1	1	0	1	0	0	1	W	2-4	u	x	u	x	x	x
			mod		1	0	0	mem									
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1	1	0	0	0	0	0	W	3	u	x	u	x	x	x
			1	1	1	0	0	reg									
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1	1	0	0	0	0	0	W	3-5	u	x	u	x	x	x
			mod		1	0	0	mem									

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Shift (cont)																	
SHR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 When MSB of reg ≠ bit following MSB of reg: V ← 1 When MSB of reg = bit following MSB of reg: V ← 0	1 1	1 1	0 1	1 0	0 1	0	0 reg	W	2	u	x	x	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 When MSB of (mem) ≠ bit following MSB of (mem): V ← 1 When MSB of (mem) = bit following MSB of (mem): V ← 0	1 mod	1	0	1 1	0	0	0 mem	W	2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1 1	1 1	0 1	1 0	0 1	0	1 reg	W	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 mod	1	0	1 1	0	0	1 mem	W	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1 1	1 1	0 1	0 0	0 1	0	0 reg	W	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 mod	1	0	0 1	0 1	0	0 mem	W	3-5	u	x	u	x	x	x
SHRA	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, V ← 0 MSB of operand does not change	1 1	1 1	0 1	1 1	0 1	0	0 reg	W	2	u	x	0	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, V ← 0, MSB of operand does not change	1 mod	1	0	1 1	0 1	0	0 mem	W	2-4	u	x	0	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1	1 1	0 1	1 1	0 1	0	1 reg	W	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1 mod	1	0	1 1	0 1	0	1 mem	W	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1	1 1	0 1	0 1	0 1	0	0 reg	W	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1 mod	1	0	0 1	0 1	0	0 mem	W	3-5	u	x	u	x	x	x
Rotation																	
ROL	reg, 1	CY ← MSB of reg, reg ← reg x 2 + CY MSB of reg ≠ CY: V ← 1 MSB of reg = CY: V ← 0	1 1	1 1	0 0	1 0	0 0	0	0 reg	W	2		x	x			
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 + CY MSB of (mem) ≠ CY: V ← 1 MSB of (mem) = CY: V ← 0	1 mod	1	0	1 0	0 0	0	0 mem	W	2-4		x	x			

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Rotation (cont)																	
ROL	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY, temp ← temp − 1	1 1	1 1	0 0	1 0	0 0	0	1 reg	W	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY, temp ← temp − 1	1 mod	1	0 0	1 0	0 0	0	1 mem	W	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY, temp ← temp − 1	1 1	1 1	0 0	0 0	0 0	0	0 reg	W	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY, temp ← temp − 1	1 mod	1	0 0	0 0	0 0	0	0 mem	W	3-5		x	u			
ROR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1 1	1 1	0 0	1 0	0 1	0	0 reg	W	2		x	x			
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY, MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1 mod	1	0 0	1 0	0 1	0	0 mem	W	2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY, temp ← temp − 1	1 1	1 1	0 0	1 0	0 1	0	1 reg	W	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY, temp ← temp − 1	1 mod	1	0 0	1 0	0 1	0	1 mem	W	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY, temp ← temp − 1	1 1	1 1	0 0	0 0	0 1	0	0 reg	W	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 mod	1	0 0	0 0	0 1	0	0 mem	W	3-5		x	u			

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Rotate																	
ROLC	reg, 1	tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy MSB of reg = CY: V ← 0 MSB of reg ≠ CY: V ← 1	1 1	1 1	0 0	1 1	0 0	0	0 reg	W	2		x	x			
	mem, 1	tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy MSB of (mem) = CY: V ← 0 MSB of (mem) ≠ CY: V ← 1	1 mod	1	0 0	1 1	0 0	0	0 mem	W	2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy, temp ← temp − 1	1 1	1 1	0 0	1 1	0 0	0	1 reg	W	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy, temp ← temp − 1	1 mod	1	0 0	1 1	0 0	0	1 mem	W	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy, temp ← temp − 1	1 1	1 1	0 0	0 1	0 0	0	0 reg	W	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy temp ← temp − 1	1 mod	1	0 0	0 1	0 0	0	0 mem	W	3-5		x	u			
RORC	reg, 1	tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1 1	1 1	0 0	1 1	0 1	0	0 reg	W	2		x	x			
	mem, 1	tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1 mod	1	0 0	1 1	0 1	0	0 mem	W	2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1 1	1 1	0 0	1 1	0 1	0	1 reg	W	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, temp ← temp − 1	1 mod	1	0 0	1 1	0 1	0	1 mem	W	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1 1	1 1	0 0	0 1	0 1	0	0 reg	W	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, temp ← temp − 1	1 mod	1	0 0	0 1	0 1	0	0 mem	W	3-5		x	u			

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Subroutine Control Transfer																	
CALL	near-proc	(SP−1, SP−2) ← PC, SP ← SP−2, PC ← PC + disp	1	1	1	0	1	0	0	0	3						
	regptr16	(SP−1, SP−2) ← PC, SP ← SP−2, PC ← regptr16	1	1	1	1	1	1	1	1	2						
			1	1	0	1	0	reg									
	memptr16	(SP−1, SP−2) ← PC, SP ← SP−2, PC ← (memptr16)	1	1	1	1	1	1	1	1	2-4						
			mod		0	1	0	mem									
	far-proc	(SP−1, SP−2) ← PS, (SP−3, SP−4) ← PC, SP ← SP−4, PS ← seg, PC ← offset	1	0	0	1	1	0	1	0	5						
	memptr32	(SP−1, SP−2) ← PS, (SP−3, SP−4) ← PC, SP ← SP−4, PS ← (memptr32 + 2), PC ← (memptr32)	1	1	1	1	1	1	1	1	2-4						
			mod		0	1	1	mem									
RET		PC ← (SP + 1, SP), SP ← SP + 2	1	1	0	0	0	0	1	1	1						
	pop-value	PC ← (SP + 1, SP), SP ← SP + 2, SP ← SP + pop-value	1	1	0	0	0	0	1	0	3						
		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), SP ← SP + 4	1	1	0	0	1	0	1	1	1						
	pop-value	PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), SP ← SP + 4, SP ← SP + pop-value	1	1	0	0	1	0	1	0	3						
Stack Manipulation																	
PUSH	mem16	(SP−1, SP−2) ← (mem16), SP ← SP−2	1	1	1	1	1	1	1	1	2-4						
			mod		1	1	0	mem									
	reg16	(SP−1, SP−2) ← reg16, SP ← SP−2	0	1	0	1	0	reg			1						
	sreg	(SP−1, SP−2) ← sreg, SP ← SP−2	0	0	0	sreg		1	1	0	1						
	PSW	(SP−1, SP−2) ← PSW, SP ← SP−2	1	0	0	1	1	1	0	0	1						
	R	Push registers on the stack	0	1	1	0	0	0	0	0	1						
	imm	(SP−1, SP−2) ← imm, SP ← SP−2, When S = 1, sign extension	0	1	1	0	1	0	S	0	2-3						
POP	mem16	(mem16) ← (SP + 1, SP), SP ← SP + 2	1	0	0	0	1	1	1	1	2-4						
			mod		0	0	0	mem									
	reg16	reg16 ← (SP + 1, SP), SP ← SP + 2	0	1	0	1	1	reg			1						
	sreg	sreg ← (SP + 1, SP), sreg : SS, DS0, DS1 SP ← SP + 2	0	0	0	sreg		1	1	1	1						
	PSW	PSW ← (SP + 1, SP), SP ← SP + 2	1	0	0	1	1	1	0	1	1	R	R	R	R	R	R
	R	Pop registers from the stack	0	1	1	0	0	0	0	1	1						
PREPARE	imm16, imm8	Prepare new stack frame	1	1	0	0	1	0	0	0	4						
DISPOSE		Dispose of stack frame	1	1	0	0	1	0	0	1	1						

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Branch																	
BR	near-label	PC ← PC + disp	1	1	1	0	1	0	0	1	3						
	short-label	PC ← PC + ext-disp8	1	1	1	0	1	0	1	1	2						
	regptr16	PC ← regptr16	1	1	1	1	1	1	1	1	2						
			1	1	1	0	0	reg									
	memptr16	PC ← (memptr16)	1	1	1	1	1	1	1	1	2-4						
			mod		1	0	0	mem									
	far-label	PS ← seg, PC ← offset	1	1	1	0	1	0	1	0	5						
	memptr32	PS ← (memptr32 + 2), PC ← (memptr32)	1	1	1	1	1	1	1	1	2-4						
			mod		1	0	1	mem									
Conditional Branch																	
BV	short-label	if V = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	0	0	2						
BNV	short-label	if V = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	0	1	2						
BC, BL	short-label	if CY = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	1	0	2						
BNC, BNL	short-label	if CY = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	1	1	2						
BE, BZ	short-label	if Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	0	0	2						
BNE, BNZ	short-label	if Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	0	1	2						
BNH	short-label	if CY OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	1	0	2						
BH	short-label	if CY OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	1	1	2						
BN	short-label	if S = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	0	0	2						
BP	short-label	if S = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	0	1	2						
BPE	short-label	if P = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	1	0	2						
BPO	short-label	if P = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	1	1	2						
BLT	short-label	if S XOR V = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	0	0	2						
BGE	short-label	if S XOR V = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	0	1	2						
BLE	short-label	if (S XOR V) OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	1	0	2						
BGT	short-label	if (S XOR V) OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	1	1	2						
DBNZNE	short-label	CW ← CW − 1 if Z = 0 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	0	2						
DBNZE	short-label	CW ← CW − 1 if Z = 1 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	1	2						
DBNZ	short-label	CW ← CW − 1 if CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	1	0	2						
BCWZ	short-label	if CW = 0, PC ← PC + ext-disp8	1	1	1	0	0	0	1	1	2						
BTCLR	sfr, imm3, short-label	if bit no. imm3 of (sfr) = 1, PC ← PC + ext − disp8, bit no. imm3 or (sfr) ← 0	0	0	0	0	1	1	1	1	5						
			1	0	0	1	1	1	0	0							

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Interrupt																	
BRK	3	(SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (15, 14), PC ← (13, 12)	1	1	0	0	1	1	0	0	1						
	imm8 (≠3)	(SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PC ← (n x 4 + 1, n x 4), PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1	1	0	0	1	1	0	1	2						
BRKV		When V = 1 (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (19, 18), PC ← (17, 16)	1	1	0	0	1	1	1	0	1						
RETI		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1	1	0	0	1	1	1	1	1	R	R	R	R	R	R
RETRBI		PC ← Save PC, PSW ← Save PSW	0	0	0	0	1	1	1	1	2	R	R	R	R	R	R
			1	0	0	1	0	0	0	1							
FINT		Indicates that interrupt service routine to the interrupt controller built in the CPU has been completed	0	0	0	0	1	1	1	1	2						
			1	0	0	1	0	0	1	0							
CHKIND	reg16, mem32	When (mem32) > reg16 or (mem32 + 2) < reg16 (SP−1, SP−2) ← PSW, (SP−3, SP−4) ← PS, (SP−5, SP−6) ← PC, SP ← SP−6, IE ← 0, BRK ← 0, PS ← (23, 22), PC ← (21, 20)	0	1	1	0	0	0	1	0	2-4						
			mod			reg			mem								
CPU Control																	
HALT		CPU Halt	1	1	1	1	0	1	0	0		1					
STOP		CPU Halt	0	0	0	0	1	1	1	1		1					
			1	0	1	1	1	1	1	0							
BUSLOCK		Bus Lock Prefix	1	1	1	1	0	0	0	0		1					
FP01 (Note 1)	fp-op	No Operation	1	1	0	1	1	X	X	X	2						
			1	1	Y	Y	Y	Z	Z	Z							
	fp-op, mem	data bus ← (mem)	1	1	0	1	1	X	X	X	2-4						
			mod		Y	Y	Y		mem								
FP02 (Note 1)	fp-op	No Operation	0	1	1	0	0	1	1	X	2						
			1	1	Y	Y	Y	Z	Z	Z							
	fp-op, mem	data bus ← (mem)	0	1	1	0	0	1	1	X	2-4						
			mod		Y	Y	Y		mem								

Notes:
(1) Does not execute but does generate an interrupt.

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
CPU Control (cont)																	
POLL		Poll and Wait	1	0	0	1	1	0	1	1	1						
NOP		No Operation	1	0	0	1	0	0	0	0	1						
DI		IE ← 0	1	1	1	1	1	0	1	0	1						
EI		IE ← 1	1	1	1	1	1	0	1	1	1						
DS0; DS1; PS; SS		Segment Override Prefix	0	0	1	sreg		1	1	0	1						
Register Bank Switching																	
MOVSPA			0	0	0	0	1	1	1	1	2						
			0	0	1	0	0	1	0	1							
BRKCS	reg16		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	1	0	1							
MOVSPB	reg16		0	0	0	0	1	1	1	1	3						
			1	0	0	1	0	1	0	1							
			1	1	1	1	1		reg								
TSKSW	reg16		0	0	0	0	1	1	1	1	3	x	x	x	x	x	x
			1	0	0	1	0	1	0	0							
			1	1	1	1	1		reg								

NEC
NEC Electronics Inc.

μPD70327 (V25 Software Guard) 16-Bit Microcomputer: Software-Secure, Single-Chip, CMOS

Description

The μPD70327 (V25 Software Guard) is a high-performance, 16-bit, single-chip microcomputer with an 8-bit external data bus. The μPD70327 is fully software compatible with the μPD70108/116 (V20®/30®) as well as the μPD70320/330 (V25™/35™).

The μPD70327 allows external executable code to be encrypted by a user-defined translation table. The μPD70327 will automatically decode the encrypted op-codes internally before the instructions are moved into the instruction execution register. As a result, the μPD70327 offers identical performance to the standard V25 even during security mode operation. The security feature may be selected by hardware and/or software, and may be switched from one state to the other under software control.

The μPD70327 has the same complement of internal peripherals as the standard V25 and maintains compatibility with existing drivers. Other than the additional mode select pin, the μPD70327 also maintains pin compatibility with other members of the standard V25 family.

Note: The electrical specifications of the V25 Software Guard and the standard V25 are the same. The instruction sets are also the same except for (BRKS and BRKN added to control the Security and Normal operational modes. For electrical specifications and standard instructions, refer to the μPD70320/322 (V25) Data Sheet.

Features

- Security and normal operational modes
- System clock speeds to 8 MHz (16-MHz crystal)
- 16-bit CPU and internal data paths
- Functional compatibility with V25
- Software upward compatible with μPD8086
- New and enhanced V-Series instructions
- 6-byte prefetch queue
- Two-channel on-chip DMA controller
- Minimum instruction cycle: 250 ns at 8 MHz
- Internal 256-byte RAM memory
- 1-megabyte memory address space; 64K-byte I/O space
- Eight internal RAM-mapped register banks
- Four multifunction I/O ports
  - 8-bit analog comparator port
  - 20 bidirectional port lines
  - Four input-only port lines
- Two independent full-duplex serial channels
- Priority interrupt controller
  - Standard vectored service
  - Register bank switching
  - Macroservice
- Pseudo SRAM and DRAM refresh controller
- Two 16-bit timers
- On-chip time base counter
- Programmable wait state generator
- Two standby modes: STOP and HALT

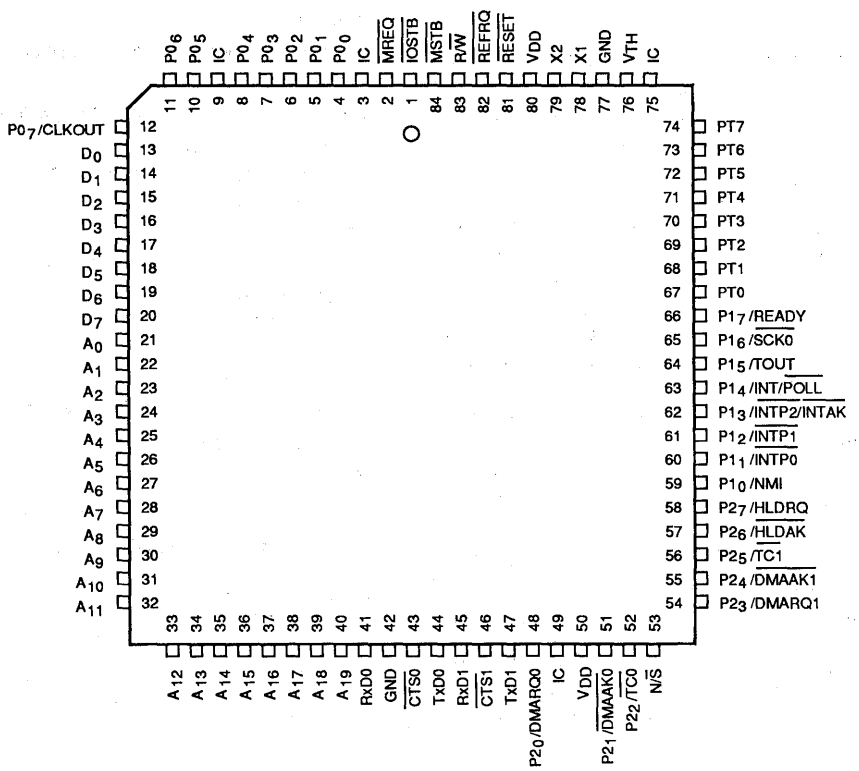
Ordering Information

Part Number	Clock (MHz)	Package
μPD70327L-8-xxx	8	84-pin PLCC
GJ-8-xxx	8	94-pin plastic QFP

V20 and V30 are registered trademarks of NEC Corporation.
V25 and V35 are trademarks of NEC Corporation.

Pin Configurations

84-Pin PLCC



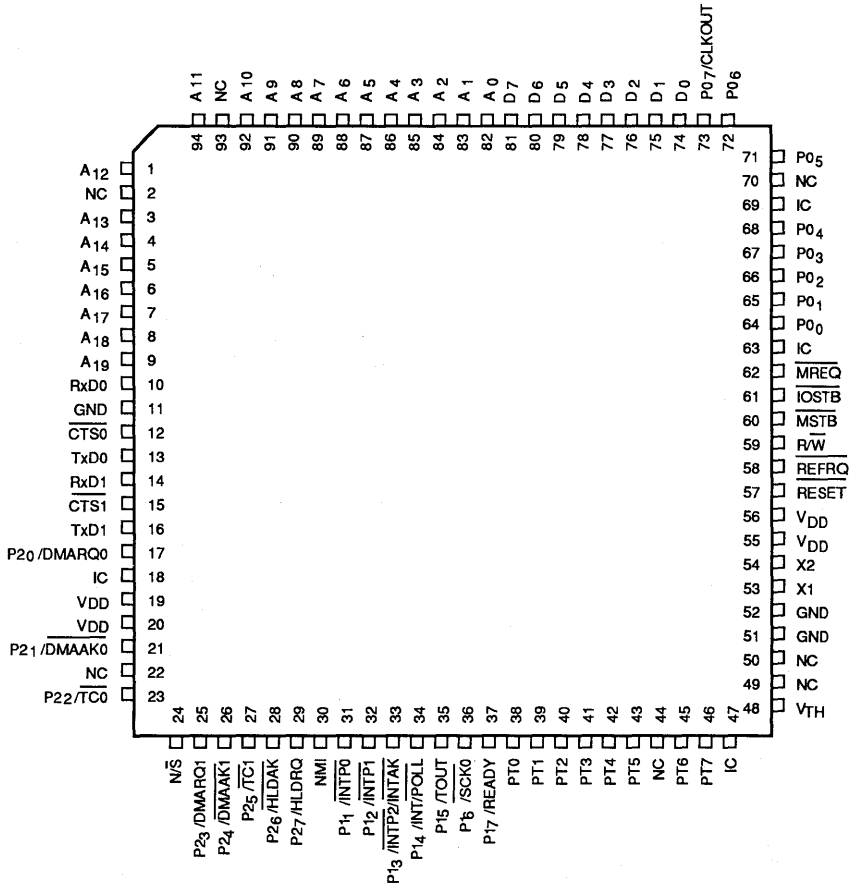
Notes:

(1) Pin functions are identical to μPD70320.

(2) All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

83SL-7039B

94-Pin Plastic QFP



Notes:

(1) Pin functions are identical to μPD70320.

(2) All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

83SL-7040B

4f

Pin Identification

Symbol	Function
A_0-A_{19}	Address bus output
$\overline{CTS0}$	Clear to send channel 0 input (Async mode); Receive clock input/output (I/O interface mode)
$\overline{CTS1}$	Clear to send channel 1 input
D_0-D_7	Bidirectional data bus
$\overline{IOSTB}$	I/O read or write strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory read/write strobe output
N/$\overline{S}$	Normal mode/security mode select input
$P0_0$-$P0_6$	I/O port 0
$P0_7$/ CLKOUT	I/O port 0; System clock output
$P1_0$/NMI	Port 1 input line; Nonmaskable interrupt input
$P1_1$-$P1_2$/$\overline{INTP0}$-$\overline{INTP1}$	Port 1 input lines; External interrupt input lines
$P1_3$/$\overline{INTP2}$/$\overline{INTAK}$	Port 1 input line; External interrupt input line; Interrupt acknowledge output
$P1_4$/INT/$\overline{POLL}$	I/O port 1; Interrupt request input; Poll input
$P1_5$/TOUT	I/O port 1; Timer out
$P1_6$/$\overline{SCK0}$	I/O port 1; Serial clock output
$P1_7$/READY	I/O port 1; Ready input
$P2_0$/DMARQ0	I/O port 2; DMA request 0 input
$P2_1$/$\overline{DMAAK0}$	I/O port 2; DMA acknowledge 0 output
$P2_2$/$\overline{TC0}$	I/O port 2; DMA terminal count 0 output
$P2_3$/DMARQ1	I/O port 2; DMA request 1 input
$P2_4$/$\overline{DMAAK1}$	I/O port 2; DMA acknowledge 1 output
$P2_5$/$\overline{TC1}$	I/O port 2; DMA terminal count 1 output
$P2_6$/$\overline{HLDAK}$	I/O port 2; Hold acknowledge output
$P2_7$/HLDRQ	I/O port 2; Hold request input
PT0-PT7	Comparator port input lines
$\overline{REFRQ}$	DRAM refresh pulse output
$\overline{RESET}$	Reset input
RxD0	Serial receive data channel 0 input
RxD1	Serial receive data channel 1 input
R/$\overline{W}$	Read cycle/write cycle ID output
TxD0	Serial transmit data channel 0 output
TxD1	Serial transmit data channel 1 output
X1, X2	Crystal connection terminals
V_{DD}	+5-volt power supply; connect both pins
V_{TH}	Threshold voltage input
GND	Ground reference; connect both pins
IC	Internal connection

PIN FUNCTIONS

A_0-A_{19} (Address Bus)

A_0-A_{19} is the 20-bit address bus used to access all external devices.

CLKOUT (System Clock)

This is the internal system clock. It can be used to synchronize external devices to the CPU.

$\overline{CTS}$n, RxDn, TxDn, $\overline{SCK0}$ (Clear to Send, Receive Data, Transmit Data, Serial Clock Out)

The two serial ports (channels 0 and 1) use these lines for transmitting and receiving data, handshaking, and serial clock output.

D_0-D_7 (Data Bus)

D_0-D_7 is the 8-bit external data bus.

DMARQn, $\overline{DMAAK}$n, TCn (DMA Request, DMA Acknowledge, Terminal Count)

These are the control signals to and from the on-chip DMA controller.

$\overline{HLDAK}$ (Hold Acknowledge)

The $\overline{HLDAK}$ output (active low) informs external devices that the CPU has released the system bus.

HLDRQ (Hold Request)

The HLDRQ input (active high) is used by external devices to request the CPU to release the system bus to an external bus master. The following lines go into a high-impedance status with internal 4.7-kΩ pullup resistors: A_0-A_{19}, D_0-D_7, $\overline{MREQ}$, R/$\overline{W}$, $\overline{MSTB}$, $\overline{REFRQ}$, and $\overline{IOSTB}$.

INT (Interrupt Request)

INT is a maskable, active-high, vectored interrupt request . After assertion, external hardware must provide the interrupt vector number.

The INT pin allows direct connection of slave μPD71059 interrupt controllers.

$\overline{INTAK}$ (Interrupt Acknowledge)

After INT is asserted, the CPU will respond with $\overline{INTAK}$ (active low) to inform external devices that the interrupt request has been granted.

INTP0-INTP2 (External Interrupt)

INTP0-INTP2 allow external devices to generate interrupts. Each can be programmed to be rising or falling edge triggered.

$\overline{\text{IOSTB}}$ (I/O Strobe)

$\overline{\text{IOSTB}}$ is asserted during read and write operations to external I/O.

$\overline{\text{MREQ}}$ (Memory Request)

$\overline{\text{MREQ}}$ (active low) informs external memory that the current bus cycle is a memory access bus cycle.

$\overline{\text{MSTB}}$ (Memory Strobe)

$\overline{\text{MSTB}}$ (active low) is asserted during read and write operations to external memory.

NMI (Nonmaskable Interrupt)

NMI cannot be masked through software and is typically used for emergency processing. Upon execution, the interrupt starting address is obtained from interrupt vector number 2. NMI can release the standby modes and can be programmed to be either rising or falling edge triggered.

N/$\overline{\text{S}}$ (N Mode/S Mode)

Normal or security mode is selected by a fixed high level (N) or low level (S) at this pin. This pin is sampled at system reset.

$P0_0$-$P0_7$ (Port 0)

$P0_0$-$P0_7$ are the lines of port 0, an 8-bit bidirectional parallel I/O port, specifiable by bit.

$P1_0$-$P1_7$ (Port 1)

The status of $P1_0$-$P1_3$ can be read but these lines are always control functions. $P1_4$-$P1_7$ are the remaining lines of parallel port 1; each line is individually programmable as either an input, an output, or a control function.

$P2_0$-$P2_7$ (Port 2)

$P2_0$-$P2_7$ are the lines of port 2, an 8-bit bidirectional parallel I/O port specifiable by bit. The lines can also be used as control signals for the on-chip DMA controller.

$\overline{\text{POLL}}$ (Poll)

Upon execution of the POLL instruction, the CPU checks the status of this pin and, if low, program execution continues. If high, the CPU checks the level of the line every five clock cycles until it is low. $\overline{\text{POLL}}$ can be used to synchronize program execution to external conditions.

PT0-PT7 (Comparator Port)

PT0-PT7 are inputs to the analog comparator port.

READY (Ready)

After READY is de-asserted low, the CPU synchronizes and inserts at least two wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than nominal μPD70325 bus cycles.

$\overline{\text{REFRQ}}$ (Refresh)

This active-low output pulse can refresh nonstatic RAM. It can be programmed to meet system specifications and is internally synchronized so that refresh cycles do not interfere with normal CPU operation.

$\overline{\text{RESET}}$ (Reset)

A low on $\overline{\text{RESET}}$ resets the CPU and all on-chip peripherals. $\overline{\text{RESET}}$ can also release the standby modes. After $\overline{\text{RESET}}$ returns high, program execution begins from address FFFF0H.

R/$\overline{\text{W}}$ (Read/Write)

R/$\overline{\text{W}}$ output allows external hardware to determine if the current operation is a read or a write cycle. It can also control the direction of bidirectional buffers.

TOUT (Timer Out)

TOUT is the square-wave output signal from the internal timer.

X1, X2 (Crystal Connections)

The internal clock generator requires an external crystal across these terminals. By programming the PRC register, the system clock frequency can be selected as the oscillator frequency (f_{OSC}) divided by 2, 4, or 8.

V_{DD} (Power Supply)

Two positive power supply pins (V_{DD}) reduce internal noise.

V_{TH} (Threshold Voltage)

The comparator port uses this pin to determine the analog reference point. The actual threshold to each comparator line is programmable to V_{TH} x n/16 where n = 1 to 16.

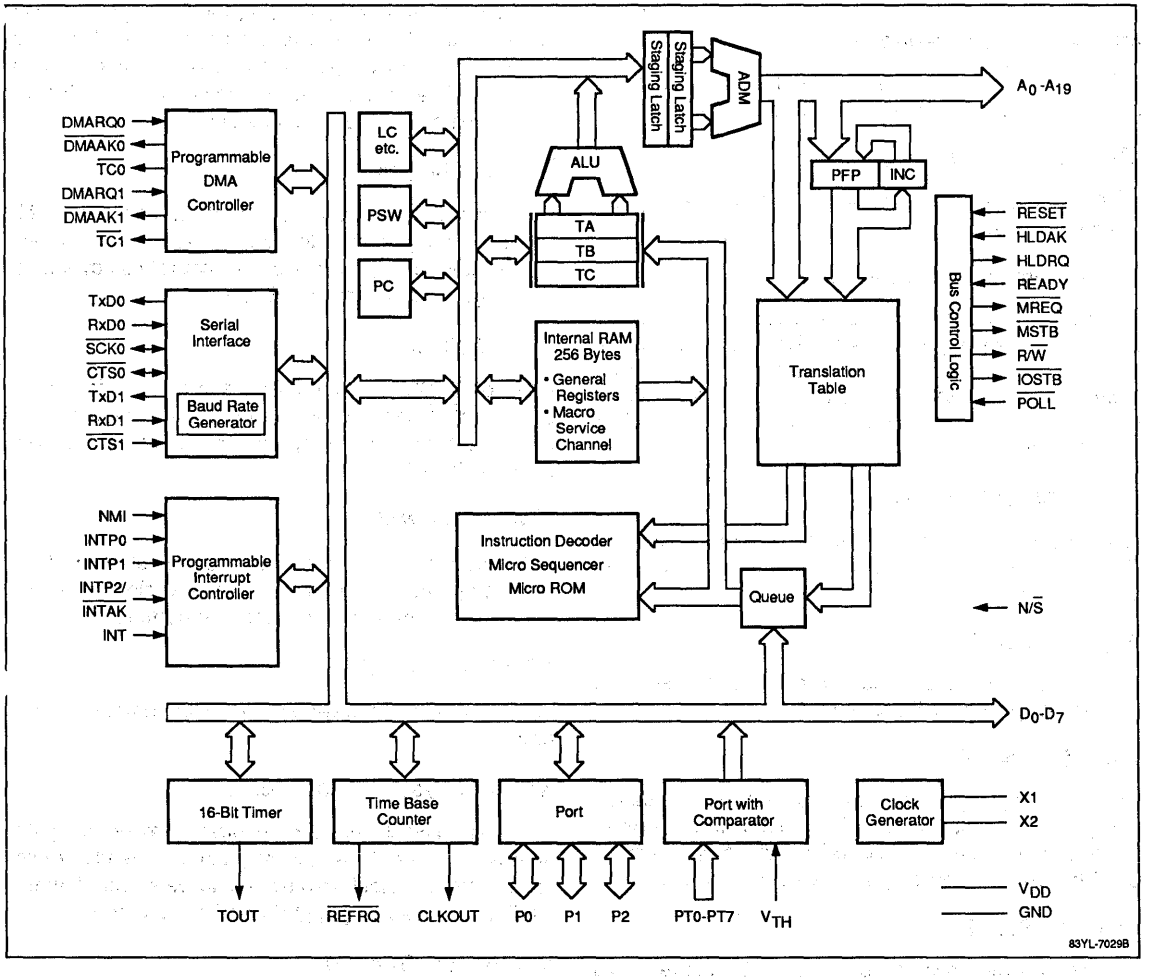
GND (Ground)

Two ground connections reduce internal noise.

IC (Internal Connection)

All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

μPD70325 Block Diagram



FUNCTIONAL DESCRIPTION

The following architectural enhancements enable the μPD70327 to perform high-speed execution of instructions.

- Dual internal data bus
- 16- and 32-bit temporary registers/shifters
- 16-bit loop counter
- Program counter and prefetch pointer

Dual Data Bus

The μPD70327 has two internal 16-bit data buses, main and secondary. This reduces the processing time required for addition/subtraction and logical comparison instructions by one third over single-bus systems. The dual data bus method allows two operands to be fetched simultaneously from the general-purpose registers and transferred to the ALU.

16- and 32-Bit Temporary Registers/Shifters

The 16-bit temporary registers/shifters (TA and TB) allow high-speed execution of multiplication/division and shift/rotate instructions. Using the temporary registers, the μPD70327 can execute multiplication/division instructions about four times faster than with the microprogrammed method.

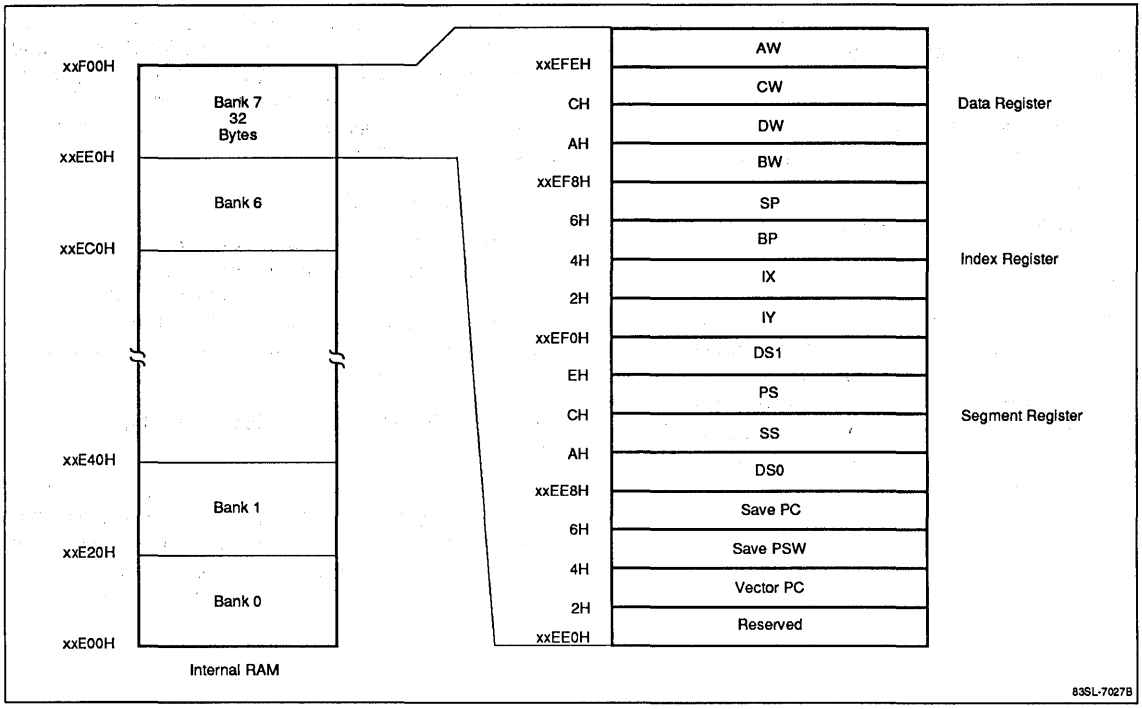
Loop Counter (LC)

The dedicated hardware loop counter (LC) counts the number of iterations for string operations and the number of shifts performed for multiple-bit shift/rotate instructions. The loop counter works with internal dedicated shifters to speed the processing of multiplication/division instructions.

Program Counter and Prefetch Pointer (PC and PFP)

The hardware PC addresses the memory location of the instruction to be executed next. The hardware PFP addresses the program memory location to be accessed by the instruction queued next. Several clock cycles are saved for branch, call, return, and break instructions.

Figure 1. μPD70327 Internal RAM Register Mapping



Register Set

Figure 1 shows the eight banks of internal registers, which the μPD70327 maps into internal RAM. Each bank contains general-purpose registers, pointer and index registers, segment registers, and save areas for context switching.

Although these memory locations may be accessed as normal RAM with the full set of memory addressing modes provided by the V25 family, the capability of context switching provides superior speed in register access. When used in the internal memory disabled state, many instructions execute considerably faster.

The eight macroservice channel control blocks are mapped into register banks 0 and 1. The μPD70327 also maps the DMA control registers over macroservice channels 0 and 1 within register bank 0.

As a result of this mapping, interrupt priorities, register banks, and macroservice channels should be allocated from the lowest priority to the highest (that is, starting from bank/priority 7 and proceeding to bank priority 0). Be careful not to map user RAM locations and/or DMA or macroservice control registers into the same internal RAM locations.

General-Purpose Registers. Four 16-bit general-purpose registers (AW, BW, CW, and DW) can serve as 16-bit registers or as four sets of dual 8-bit registers (AH, AL, BH, BL, CH, CL, DH, and DL). The instruction classes default to the following general-purpose registers.

- AW Word multiplication/division, word I/O, data conversion.
- AL Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation.
- AH Byte multiplication/division.
- BW Translation
- CW Loop control, branch, and repeat prefixes.
- CL Shift instructions, rotate instructions, BCD operations.
- DW Word multiplication/division, indirect I/O addressing.

Pointers and Index Registers. These registers are 16-bit base pointers (SP, BP) or index registers (IX, IY) in based addressing, indexed addressing, and based indexed addressing. They are used as default registers under the following conditions.

- SP Stack operations
- IX Block transfer (source), BCD string operations
- IY Block transfer (destination), BCD string operations

Segment Registers. The segment registers divide the 1M-byte address space into 64K-byte blocks. Each segment register functions as a base address to a block; the effective address is an offset from that base. Physical addresses are generated by shifting the associated segment register left by four binary digits and then adding the offset address. The segment registers and default offsets are listed below.

Segment Register	Default Offset
PS (Program Segment)	PC (Program Counter)
SS (Stack Segment)	SP and Effective Address
DS0 (Data Segment 0)	IX and Effective Address
DS1 (Data Segment 1)	IY and Effective Address

Save Registers. Save PC and save PSW are used as the storage areas during register bank context-switching operations. The Vector PC save location contains the effective address of the interrupt service routine when register bank switching is used to service interrupts.

Program Counter. The PC is a 16-bit binary counter that contains the offset address from the program segment of the next instruction to be executed. It is incremented every time an instruction is received from the queue. It is loaded with a new location whenever the branch, call, return, break, or interrupt is executed.

Program Status Word. The PSW contains status and control flags used by the CPU and two general-purpose user flags. The configuration of this 16-bit register is shown below.

15							8
MD	RB2	RB1	RB0	V	DIR	IE	BRK

7							0
S	Z	F1	AC	F0	P	BRKI	CY

Status Flags

V	Overflow bit
S	Sign
Z	Zero
AC	Auxiliary carry
P	Parity
CY	Carry

Control Flags

DIR	Direction of string processing
IE	Interrupt enable
BRK	Break (after every instruction)
RBn	Current register bank flags
BRKI	I/O trap enable
F0, F1	General-purpose user flags (accessed by flag SFR)
MD	Normal/Security mode select

The eight low-order bits of the PSW can be stored in register AH and restored using a MOV instruction. The only way to alter the RBn bits with software is to execute one of the special bank switch instructions.

Memory Map

The μPD70327 has a 20-bit address bus that can directly access 1M-byte memory. Unlike the standard V25, the V25 Software Guard does not support internal ROM.

The internal data area (IDA) is a 256-byte internal RAM area followed consecutively by a 256-byte special function register (SFR) area.

All the data and control registers for on-chip peripherals and I/O are mapped into the SFR area and accessed as RAM.

The IDA is dynamically relocatable in 4K-byte increments by changing the value in the internal data base (IDB) register. The value in this register is assigned as the uppermost eight bits of the IDA address. The IDB register is accessed from two memory locations, FFFFFH and XXFFFH, where XX is the value in the IDB register.

On reset, the internal data base register is set to FFH, which maps the IDA into the uppermost area of memory.

In large-scale systems where internal RAM is not required for data memory, internal RAM can be removed completely from the address space and dedicated entirely to registers and control functions such as macroservice and DMA channels. Clearing the RAMEN bit in the processor control register does this. When the RAMEN bit is cleared, internal RAM can only be accessed by register addressing or internal control processes. Many instructions execute faster when internal RAM is disabled.

INSTRUCTIONS

The V25 family instruction set is fully compatible with the V20 native mode. The V20 instruction set is a superset of the μPD8086/8088 instruction set with different execution times and mnemonics.

The μPD70327 does not support the V20 8080 emulation mode. All instructions pertaining to this mode have been deleted from the V25 family instruction set.

Enhanced Instructions

In addition to the basic V20 instruction set, the μPD70327 supports numerous enhanced instructions, which together form the standard V-Series instruction set. These enhanced instructions are described in full in the V25/V35 User's Manual and are listed below.

PUSH imm	CHKIND
PUSH R	INM
POP R	OUTM
MUL imm	PREPARE
SHL imm8	DISPOSE
ROL imm8	ROR imm8
ROLC imm8	RORC imm8

Unique Instructions

The μPD70327 supports a set of unique instructions, which are also discussed in the V25/V35 User's Manual. These instructions belong to one of the following groups: Variable Length Bit Field Operations, Packed BCD Instructions, Bit Manipulation Instructions, Repeat Prefixes, and Special V25 Family Register Bank Switching Instructions.

INS	EXT
ADD4S	CMP4S
ROL4	SET1
TEST1	ROR4
CLR1	NOT1
BTCLR	REPC
REPNC	BRKCS reg16
TSKSW reg16	MOVSPA
MOVSPB	RETRBI

INTERRUPT STRUCTURE

The μPD70327 can service interrupts generated by both hardware and software. Software interrupts are serviced through vectored interrupt processing. The following interrupts are provided.

Divide Error	Single Step
Overflow	BRK3
BRK imm8	Array Bounds
Escape Trap	I/O Trap

When executing software written for another system, it is preferable to implement I/O using the on-chip peripherals. However, since the μPD70327 peripherals are memory mapped, some software conversion may be required. This mapping strategy allows the internal peripherals to be accessed using all standard addressing modes, including the advanced bit and bit-field manipulation instructions of the μPD70327. Also, the I/O trap feature of the μPD70327 allows an easy hardware solution to remapping external devices to internal μPD70327 peripherals.

Interrupt Vectors

Table 1 lists the interrupt vectors beginning at physical address 00H. External memory is required to service these routines. By servicing interrupts via the macroservice function or context switching, this requirement can be eliminated.

Each interrupt vector is 4 bytes wide. To service a vectored interrupt, the lower addressed word is transferred to the PC and the upper word to the PS. The pseudocode for the vectored interrupt service overhead is shown below.

(SP-1, SP-2) ← PSW
(SP-3, SP-4) ← PS
(SP-5, SP-6) ← PC
SP ← SP-6
IE ← 0, BRK ← 0
PS ← vector high bytes
PC ← vector low bytes

Table 1. Interrupt Vectors

Address	Vector	Assigned Use
00	0	Divide error
04	1	Break flag
08	2	NMI
0C	3	BRK3 instruction
10	4	BRKV instruction
14	5	CHKIND instruction
18	6	General purpose
1C	7	FPO instructions
20-2C	8-11	General purpose
30	12	INTSER0 (Interrupt serial error, channel 0)
34	13	INTSR0 (Interrupt serial receive, channel 0)
38	14	INTST0 (Interrupt serial transmit, channel 0)

Table 1. Interrupt Vectors (cont)

Address	Vector	Assigned Use
3C	15	General purpose
40	16	INTSER1 (Interrupt serial error, channel 1)
44	17	INTSR1 (Interrupt serial receive, channel 1)
48	18	INTST1 (Interrupt serial transmit, channel 1)
4C	19	I/O trap
50	20	INTD0 (Interrupt from DMA, channel 0)
54	21	INTD1 (Interrupt from DMA, channel 1)
58	22	General purpose
5C	23	General purpose
60	24	INTP0 (Interrupt from peripheral 0)
64	25	INTP1 (Interrupt from peripheral 1)
68	26	INTP2 (Interrupt from peripheral 2)
6C	27	General purpose
70	28	INTTU0 (Interrupt from timer unit 0)
74	29	INTTU1 (Interrupt from timer unit 1)
78	30	INTTU2 (Interrupt from timer unit 2)
7C	31	INTTB (Interrupt from time base counter)
080-3FF	32-255	General purpose

Hardware Interrupt Configuration

The μPD70327 features a high-performance on-chip controller capable of controlling multiple processing for interrupts from up to 17 different sources (5 external, 12 internal). The interrupt configuration includes system interrupts that are functionally compatible with those of the V20/V30. In addition, two unique high-speed microcontroller interrupt servicing methods are available.

Interrupt Sources

The 17 interrupt sources are divided into groups for management by the interrupt controller. Using software, each of the groups can be assigned a priority from 0 (highest) to 7 (lowest). The priority of individual interrupts within a group is fixed in hardware.If interrupts from different groups occur simultaneously and the groups have the same priority level, the priority followed will be as shown in the Default Priority column of table 2.

Table 2. Interrupt Sources

Group	Interrupt Source (Priority Within Group) 1	2	3	Default Priority
Nonmaskable interrupt	NMI	–	–	0
Timer unit	INTTU0	INTTU1	INTTU2	1
DMA controller	INTD0	INTD1	–	2
External peripheral interrupt	INTP0	INTP1	INTP2	3
Serial channel 0	INTSER0	INTSR0	INTST0	4
Serial channel 1	INTSER1	INTSR1	INTST1	5
Time base counter	INTTB	–	–	6
Interrupt request	INT	–	–	7

The priority of the currently active interrupt is stored in the ISPR special function register. Bits PR_7-PR_0 correspond to the eight possible interrupt request priority groups. The ISPR keeps track of the priority of active interrupts by setting the appropriate bit of this register. The address of this 8-bit register is xxFFCH, and the format is shown below. Again it is recommended that priorities be assigned starting with the lowest (programmed to 6 or 7) and proceed up in priority to minimize the risk of overlapping macroservice and register bank switch interrupt service control blocks.

PR_7	PR_6	PR_5	PR_4	PR_3	PR_2	PR_1	PR_0

NMI and INT are system-type external vectored interrupts. NMI is not maskable by software. INT is maskable by the IE bit in the PSW and requires that an external device provide the interrupt vector number. It is designed to allow the interrupt controller to be expanded by the addition of an external interrupt controller such as the μPD71059.

NMI, INTP0, and INTP1 are edge-sensitive inputs. By selecting the appropriate bits in the interrupt mode register, these inputs can be programmed to be either rising- or falling-edge triggered. Bits ES0-ES2 correspond to INTP0-INTP2, respectively, as shown in figure 2.

Figure 2. External Interrupt Mode Register (INTM)

0	ES2	0	ES1	0	ES0	0	ESNMI
7			Address xxF40H				0

ES2	**INTP2 Input Effective Edge**
0	Falling edge
1	Rising edge
ES1	**INTP1 Input Effective Edge**
0	Falling edge
1	Rising edge
ES0	**INTP0 Input Effective Edge**
0	Falling edge
1	Rising edge
ESNMI	**NMI Input Effective Edge**
0	Falling edge
1	Rising edge

Interrupt Processing Modes

Interrupts, with the exception of NMI, INT, and INTTB have high-performance capability and can be processed in any of three modes: standard vector interrupt, register bank context switching and macroservice. The processing mode for a given interrupt can be chosen by enabling the appropriate bits in the corresponding interrupt request control register. Each interrupt, except INT and NMI, has its own associated IRC register. The general format for each of these registers is shown in figure 3.

All interrupt processing routines other than those for NMI and INT must end with the execution of the FINT instruction. Otherwise, subsequently, only interrupts of higher priority will be accepted.

In the vectored service mode, the CPU traps to a vector location.

Figure 3. Interrupt Request Control Registers (IRC)

IF	IMK	MS/INT	ENCS	0	PR_2	PR_1	PR_0
7							0

IF	**Interrupt Flag**
0	No interrupt request generated
1	Interrupt request generated
IMK	**Interrupt Mask**
0	Open (interrupts enabled)
1	Closed (interrupts disabled)
MS/INT	**Interrupt Response Method**
0	Vector interrupt or register bank switching
1	Macroservice function
ENCS	**Register Bank Switching Function**
0	Not used
1	Used
PR_2-PR_0	**Interrupt Group Priority (0-7)**
0 0 0	Highest (0)
	↓
1 1 1	Lowest (7)

Register Bank Switching.

Register bank context switching allows interrupts to be processed rapidly by switching register banks. After an interrupt, the new register bank selected has the same bank number (0-7) as the priority programmed in the associated IRC register. The PC and PSW are automatically saved in the save areas of the new register bank, and the address of the interrupt routine is loaded from the vector PC storage register in the new register bank.

As in the vectored mode, the IE and BRK bits in the PSW are cleared to zero. After interrupt processing, execution of the RETRBI instruction returns control to the original register bank and restores the former PC and PSW. Figures 4 and 5 show register bank context switching and register bank return.

This method of interrupt service offers a dramatic performance advantage over normal vectored service because there is no need to store and retrieve data/registers on the stack. This also allows hardware-based real-time task switching in high-speed environments.

In addition to context switching, the μPD70327 has a task switch opcode (TSKSW) that allows multiple independent processes to be internally resident. Figure 6 shows the task switching function.

Figure 4. Register Bank Context Switching

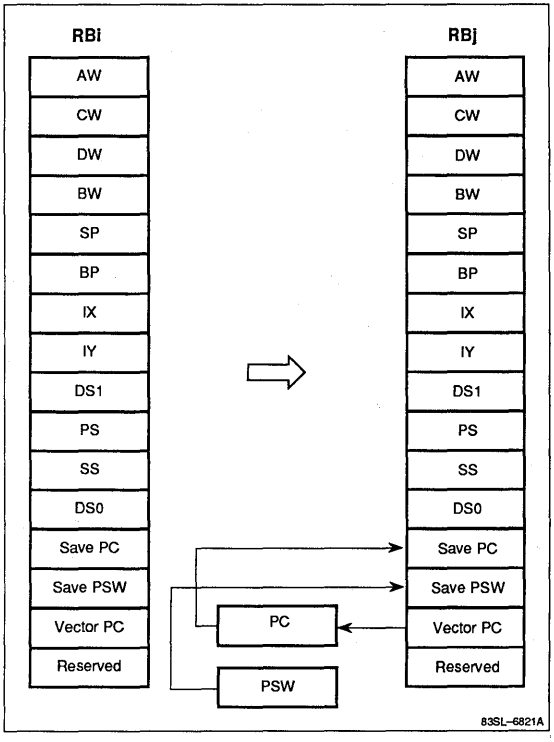
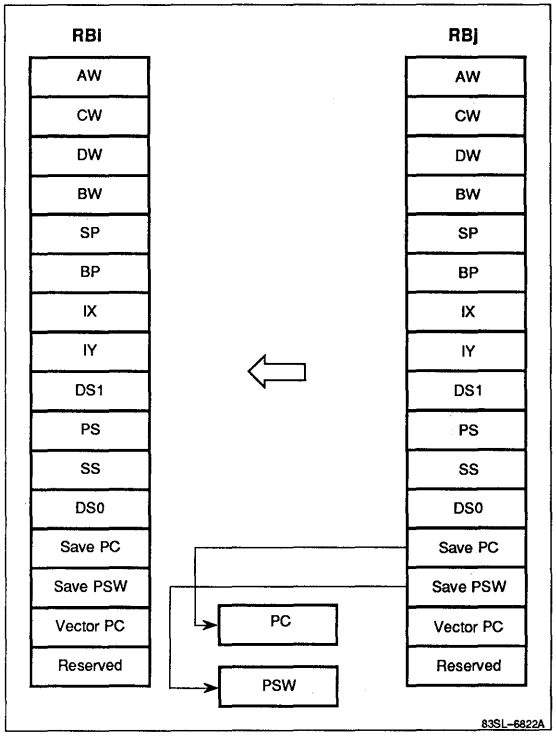
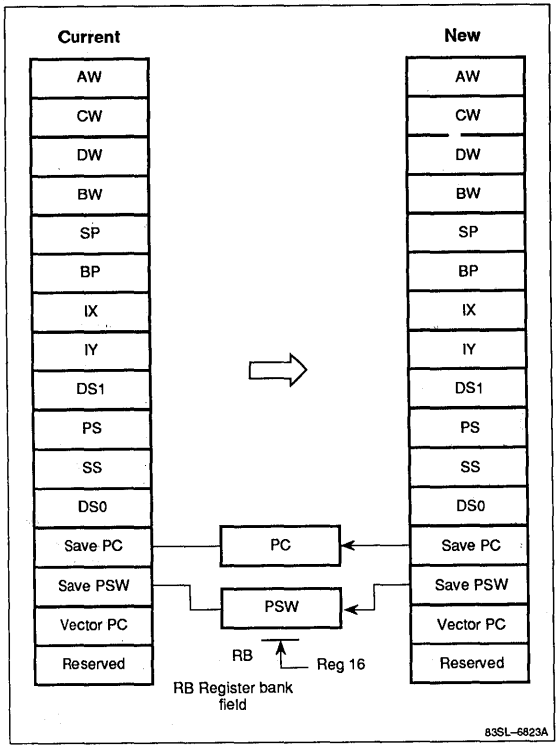


Figure 5. Register Bank Return



4f

Figure 6. Task Switching



Macroservice Function

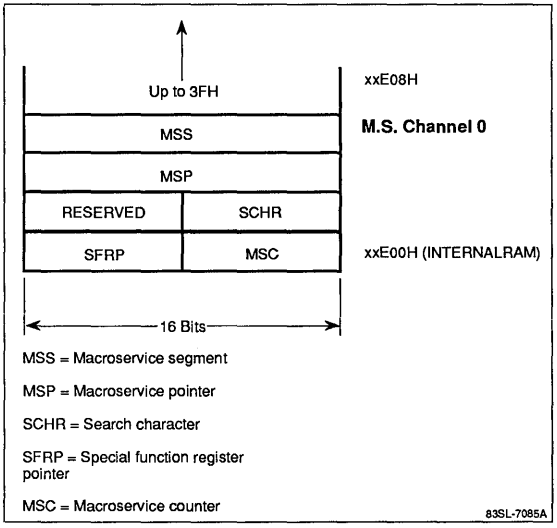
The macroservice function (MSF) is a special microprogram that acts as an internal DMA controller between on-chip peripherals (special-function registers, SFR and memory. The MSF greatly reduces software overhead and CPU time that other processors would require for register save processing, register returns, and other handling associated with interrupt processing.

If the MSF is selected for a particular interrupt, each time the request is received, a byte or word of data is transferred between an SFR and memory without interrupting the CPU. Each time a request occurs, the macroservice counter is decremented. When the counter reaches zero, an interrupt to the CPU is generated. The MSF also has a character search option. When selected, every byte transferred is compared to an 8-bit search character and an interrupt is generated if a match occurs or if the macroservice counter reaches zero.

Like the NMI, INT, and INTTB, the two DMA controller interrupts (INTD0 and INTD1) do not have MSF capability.

Eight 8-byte macroservice channels are mapped into internal RAM from XXE00H to XXE3FH. Each macroservice channel contains all necessary information to execute the macroservice process. Figure 7 shows the components of each channel.

Figure 7. Macroservice Channels



Setting the macroservice mode requires programming the corresponding macroservice control register. Each individual interrupt, excluding INT, NMI, serial error, DMA, and TBC, has its own associated MSC register. Figure 8 shows the generic format for all MSC registers.

Figure 8. Macroservice Control Registers (MSC)

MSM_2	MSM_1	MSM_0	DIR	0	CH_2	CH_1	CH_0
7							0

MSM_2-MSM_0	Macroservice Mode
0 0 0	Normal (8-bit transfer)
0 0 1	Normal (16-bit transfer)
1 0 0	Character search (8-bit transfer)
	Other combinations are not allowed.
DIR	**Data Transfer Direction**
0	Memory to SFR
1	SFR to memory
CH_2-CH_0	**Macroservice Channel**
0 0 0	Channel 0
	↓
1 1 1	Channel 7

TIMER UNIT

The μPD70327 (figure 9) has two programmable 16-bit interval timers (TM0 and TM1) on chip, each with variable input clock frequencies. Each of the two 16-bit timer registers has an associated 16-bit modulus register (MD0 and MD1). Timer 0 operates in the interval timer mode or one-shot mode; timer 1 has only the interval timer mode.

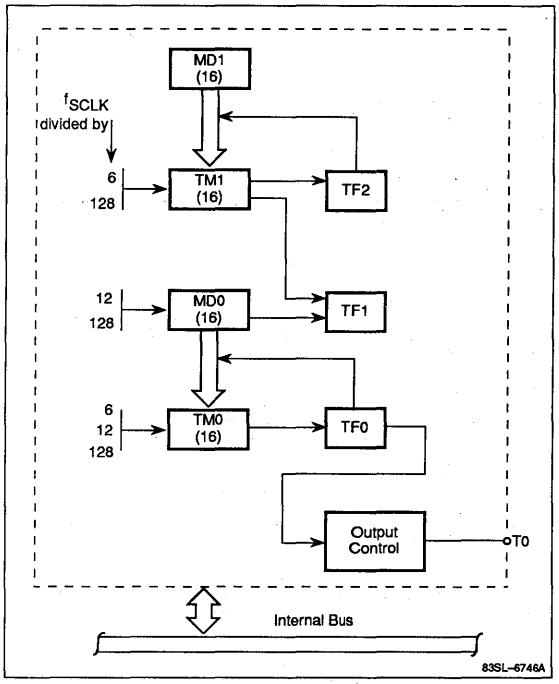
Interval Timer Mode

In this mode, TM0/TM1 are decremented by the selected input clock, and, after counting out, the registers are automatically reloaded from the modulus registers and counting continues. Each time TM1 counts out, interrupts are generated through TF1 and TF2 (timer flags 1 and 2). When TM0 counts out, an interrupt is generated through TF0. The timer-out signal can be used as a square-wave output whose half-cycle is equal to the count time.

Two input clocks derived from the system clock are SCLK/6 and SCLK/128. Typical timer values shown below are based on f_{OSC} = 10 MHz and f_{SCLK} = $f_{OSC}/2$.

Clock	Timer Resolution	Full Count
SCLK/6	1.2 μs	78.643 μs
SCLK/128	25.6 μs	1.678 s

Figure 9. Timer Unit Block Diagram



One-Shot Mode

In the one-shot mode, TM0 and MD0 operate as independent one-shot timers. Starting with a preset value, each is decremented to zero. At zero, counting ceases and an interrupt is generated by TF0 (from TM0) or TF1 (from MD0).

When TM0 is programmed to one-shot mode, TM1 may still operate in interval mode.

Two input clocks derived from the system clock are SCLK/12 and SCLK/128. Typical timer values shown below are based on f_{OSC} = 10 MHz and f_{SCLK} = $f_{OSC}/2$.

Clock	Timer Resolution	Full Count
SCLK/12	2.4 μs	157.283 ms
SCLK/128	25.6 μs	1.678 s

Timer Control Registers

Setting the desired timer mode requires programming the timer control register. See figures 10 and 11 for the TMC register format.

4f

Figure 10. Timer Control Register 0 (TMC0)

TS0	TCLK0	MS0	MCLK0	ENT0	ALV	MOD_1	MOD_0
7			Address xxF90H				0

TS0	TM0 in Either Mode
0	Stop countdown
1	Start countdown

MOD_1	MOD_0	TCLK0	TM0 Register Clock Frequency
0	0	0	$f_{SCLK}/6$ (Interval)
0	0	1	$f_{SCLK}/128$ (Interval)
0	1	0	$f_{SCLK}/12$ (One-shot)
0	1	1	$f_{SCLK}/128$ (One-shot)

MS0	MD0 Register Countdown (One-Shot Mode)
0	Stop
1	Start

MCLK0	MD0 Register Clock Frequency
0	$f_{SCLK}/12$
1	$f_{SCLK}/128$

ENT0	TOUT Square-Wave Output
0	Disable
1	Enable

ALV	TOUT Initial Level When TOUT Disabled by ENT0 = 0
0	Low
1	High

MOD_1	MOD_0	Timer Unit Mode
0	0	Interval timer
0	1	One-shot
1	X	Reserved

Figure 11. Timer Control Register 1 (TMC1)

TS1	TCLK1	0	0	0	0	0	0
7			Address xxF91H				0

TS1	Timer 1 Countdown
0	Stop
1	Start

TCLK1	Timer 1 Clock Frequency
0	$f_{SCLK}/6$
1	$f_{SCLK}/128$

TIME BASE COUNTER

The 20-bit free-running time base counter (TBC) controls internal timing sequences and is available to the user as the source of periodic interrupts at lengthy intervals. One of four interrupt periods can be selected by programming the TB0 and TB1 bits in the processor control register (PRC). The TBC interrupt is unlike the others because it is fixed as a level 7 vectored interrupt. Macroservice and register bank switching cannot be used to service this interrupt. See figures 12 and 13.

Figure 12. Time Base Interrupt Request Control Register (TBIC)

TBF	TBMK	0	0	0	1	1	1
7			Address xxFECH				0

TBF	Time Base Interrupt Flag
0	No interrupt generated
1	Interrupt generated

TBMK	Time Base Interrupt Mask
0	Unmasked
1	Masked

Figure 13. Processor Control Register (PRC)

0	RAMEN	0	0	TB_1	TB_0	PCK_1	PCK_0
7			Address xxFEBH				0

RAMEN	Built-In RAM
0	Disable
1	Enable

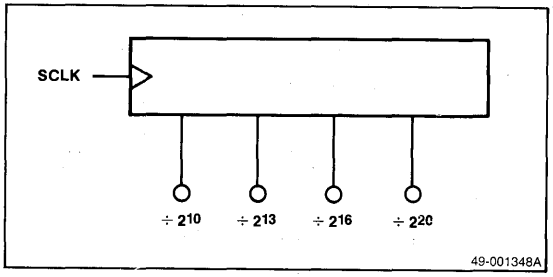
TB_1	TB_0	Time Base Interrupt Period
0	0	$2^{10}/f_{SCLK}$
0	1	$2^{13}/f_{SCLK}$
1	0	$2^{16}/f_{SCLK}$
1	1	$2^{20}/f_{SCLK}$

PCK_1	PCK_0	System Clock Frequency (f_{CLK})
0	0	$f_{OSC}/2$
0	1	$f_{OSC}/4$
1	0	$f_{OSC}/8$
1	1	Reserved

The RAMEN bit in the PRC register allows the internal RAM to be removed from the memory address space to implement faster instruction execution.

The TBC (figure 14) uses the system clock as the input frequency. The system clock can be changed by programming the PCK0 and PCK1 bits in the processor control register (PRC). Reset initializes the system clock to $f_{OSC}/8$ (f_{OSC} = external oscillator frequency).

Figure 14. Time Base Counter (TBC) Block Diagram



REFRESH CONTROLLER

The μPD70327 has an on-chip refresh controller for dynamic and pseudostatic RAM memory. The refresh controller generates refresh cycles between the normal CPU bus cycles according to the refresh specifications programmed.

The refresh controller outputs a 9-bit refresh address on address bits A_8-A_0 during the refresh bus cycle. Address bits A_{19}-A_9 are fixed to 0s during this cycle. The 9-bit refresh address is automatically incremented at every refresh cycle for 512 row addresses. The 8-bit refresh mode (RFM) register (figure 15) specifies the refresh operation and allows refresh during both CPU HALT and HOLD modes. Refresh cycles are automatically timed to $\overline{\text{REFRQ}}$ following read/write cycles to minimize the effect on system throughput.

The following shows the $\overline{\text{REFRQ}}$ pin level in relation to bits 4 (RFEN) and 7 (RFLV) of the refresh mode register.

RFEN	RFLV	$\overline{\text{REFRQ}}$ Level
0	0	0
0	1	1
1	0	0
1	1	Refresh pulse output

Figure 15. Refresh Mode Register (RFM)

RFLV	HLDRF	HLTRF	RFEN	RFW_1	RFW_0	RFT_1	RFT_0
7			Address xxFE1H				0

RFLV	RFEN	$\overline{\text{REFRQ}}$ Output Signal Level
0	0	0
1	0	1
0	1	0
1	1	Refresh pulse

HLDRF	Automatic Refresh Cycle in HOLD Mode
0	Disabled
1	Enabled

HLTRF	Automatic Refresh Cycle in HALT Mode
0	Disabled
1	Enabled

RFEN	Automatic Refresh Cycle
0	Refresh pin = RFLV
1	Refresh enabled

RFW_1	RFW_0	No. of Wait States Inserted in Refresh Cycle
0	0	0
0	1	1
1	0	2
1	1	2

RFT_1	RFT_0	Refresh Period
0	0	16/SCLK
0	1	32/SCLK
1	0	64/SCLK
1	1	128/SCLK

SERIAL INTERFACE

The μPD70327 has two full-duplex UARTs, channels 0 and 1. Each channel (figure 16) has a transmit line (TxDn), a receive line (RxDn), and a clear-to-send (CTSn) handshaking line. Communication is synchronized by a start bit, and either even, odd, or no parity may be programmed. Character length may be programmed to either 7 or 8 bits, and either 1 or 2 stop bits.

Each serial channel of the μPD70327 has a dedicated baud rate generator, so there is no need to obligate any of the on-chip timers to handle this function. The baud rate generators allow individual transfer rates for each channel and support rates up to 1.25 Mb/s. All standard baud rates are available and are not restricted by the value of the particular external crystal.

Each baud rate generator has an 8-bit data register (BRGn) that functions as a prescaler to a programmable input clock selected by the serial communication control register (SCCn). Together these must be set to generate a frequency equivalent to the desired baud rate.

The baud rate generator can be programmed to obtain the desired transmission rate according to the following formula:

$$B \times G = \frac{SCLK \times 10^6}{2^{n+1}}$$

where:

B = baud rate
G = baud rate generator register (BRGn) value
n = input clock specification; the value loaded into the SCCn register (n = 0 to 8)
SCLK = system clock frequency (MHz)

Based on the above formula, the following table shows the baud rate generator values used to obtain standard transmission rates when SCLK = 5 MHz.

Baud Rate	n	BRGn	Error (%)
110	7	178	0.25
150	7	130	0.16
300	6	130	0.16
600	5	130	0.16
1200	4	130	0.16
2400	3	130	0.16
4800	2	130	0.16
9600	1	130	0.16
19.2k	0	130	0.16
38.4k	0	65	0.16
1.25M	0	2	0.00

In addition to the asynchronous mode, channel 0 has a synchronous I/O interface mode. In this mode, each bit of data transferred is synchronized to a serial clock (SCLK0). This mode is compatible with the μCOM75 and μCOM87 series, and allows direct interfacing to these devices.

Figures 17, 18, and 19 show the three serial communication registers: SCM, SCC, and SCE.

Figure 16. Serial Interface Block Diagram

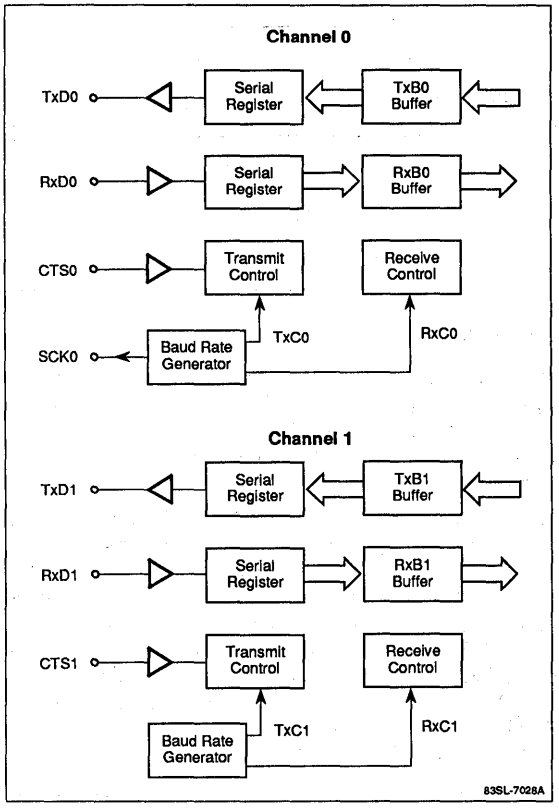


Figure 17. Serial Communication Mode Registers (SCM)

TxRDY	RxB	PRTY1	PRTY0	CLTSK	SLRSCK	MD1	MD0
7							0

TxRDY	Transmitter Control
0	Disabled
1	Enabled
RxB	**Receiver Control**
0	Disabled
1	Enabled
PRTY1-PRTY0	**Parity Control**
0 0	No parity
0 1	0 parity (Note 1)
1 0	Odd parity
1 1	Even parity
CLTSK	**Character Length (Async Mode)** **Tx Shift Clock (I/O Interface Mode)**
0	7 bits (Async) No effect (I/O intfc)
1	8 bits (Async) Trigger transmit (I/O intfc)
SLRSCK	**Stop Bits (Async Mode)** **Receiver Clock (I/O Interface Mode)**
0	1 stop bit (Async) Ext clock input on CTS0 (I/O intfc)
1	2 stop bits (Async) Int clock output on CTS1 (I/O intfc)
MD1-MD0	**Mode**
0 0	I/O interface (Note 2)
0 1	Asynchronous
1 x	Reserved

Notes:

(1) Parity is 0 during transmit and ignored during receive.

(2) I/O interface mode only.

(3) Channel 0 only.

Figure 18. Serial Communication Control Register (SCC)

0	0	0	0	PRS_3	PRS_2	PRS_1	PRS_0
7							0

PRS_3-PRS_0	Baud Rate Generator Input Clock Frequency
0 0 0 0	$f_{SCLK}/2$ (n = 0)
0 0 0 1	$f_{SCLK}/4$
0 0 1 0	$f_{SCLK}/8$
0 0 1 1	$f_{SCLK}/16$
0 1 0 0	$f_{SCLK}/32$
0 1 0 1	$f_{SCLK}/64$
0 1 1 0	$f_{SCLK}/128$
0 1 1 1	$f_{SCLK}/256$
1 0 0 0	$f_{SCLK}/512$ (n = 8)

Figure 19. Serial Communications Error Register (SCE)

RxDn	0	0	0	0	ERP	ERF	ERO
7							0

RxDn	Receive Terminal State
0, 1	Status of RxD pin
ERP	**Parity Error Flag**
0	No error
1	Transmit and receive parity are different
ERF	**Framing Error Flag**
0	No error
1	Stop bit not detected
ERO	**Overrun Error Flag**
0	No error
1	Data is received before receive buffer outputs previous data

DMA CONTROLLER

The μPD70327 has a two-channel, on-chip Direct Memory Access (DMA) controller. This allows rapid data transfer between memory and auxiliary devices. The DMA controller supports four modes of operation, two for memory-to-memory transfers and two for I/O to memory; in all cases, transfer direction is programmable.

Memory-to-Memory Transfers

In the single-step mode, a single DMA request will commence the alternation of one DMA cycle with one CPU cycle until the prescribed number of transfers (terminal count) is reached. Interrupts are accepted while in this mode.

Alternatively, in the burst mode, one DMA request causes DMA transfer cycles to continue consecutively until the DMA terminal count decrements to zero. Software can initiate memory-to-memory transfers.

I/O-to-Memory Transfers

The single transfer mode will yield exactly one DMA transfer per DMA request. After the transfer, the bus is returned to the CPU. Alternatively, in demand release mode, the rising edge of DMARQ enable DMA cycles which continue while the DMA request remains active.

DMA Registers

Figures 20 and 21 show the DMA mode registers (DMAM) and the DMA address control registers (DMAC).

In all modes, the TC (Terminal Count) output pin will pulse low and a DMA end-of-service interrupt request will be internally generated after the programmed number of transfers have been completed. The bottom of internal RAM contains all the necessary address information for the designated DMA channels. The DMA channel mnemonics are as follows.

TC	Terminal count
SAR	Source address register
SARH	Source address register high
DAR	Destination address register
DARH	Destination address register high

Figure 20. DMA Mode Registers (DMAM)

MD_2	MD_1	MD_0	W	EDMA	TDMA	0	0
7							0

MD_2-MD_0		Transfer Mode
0 0 0		Single-step (memory to memory)
0 0 1		Demand release (I/O to memory)
0 1 0		Demand release (memory to I/O)
0 1 1		Reserved
1 0 0		Burst (memory to memory)
1 0 1		Single-transfer (I/O to memory)
1 1 0		Single-transfer (memory to I/O)
1 1 1		Reserved
W		**Transfer Method**
0		Byte transfer
1		Word transfer
EDMA	**TDMA**	**Transfer Condition**
0	0	Disabled
1	0	Maintain condition
1	1	Start DMA transfer

Figure 21. DMA Address Control Registers (DMAC)

0	0	PD_1	PD_0	0	0	PS_1	PS_0
7							0

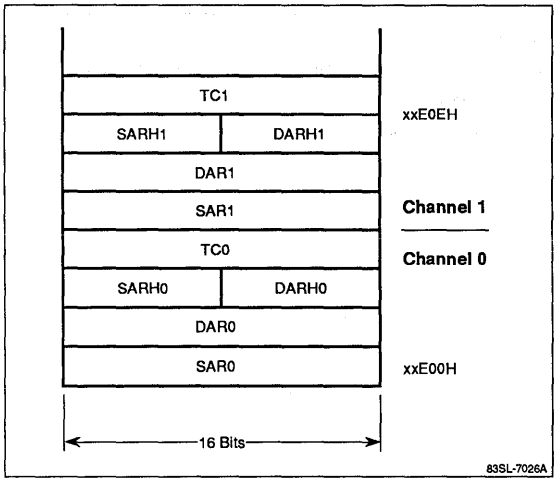
PD_1-PD_0	Destination Address Offset
0 0	No modification
0 1	Increment
1 0	Decrement
1 1	No modification
PS_1-PS_0	**Source Address Offset**
0 0	No modification
0 1	Increment
1 0	Decrement
1 1	No modification

These control registers (figure 22) are mapped into the same area of register bank 0 as the macroservice control block registers. These macroservice channels should not be used when the DMA controller is active.

The DMA controller generates the physical source and destination addresses by offsetting Address High register 12 bits to the left and then adding the Address register. The source and destination address registers can be programmed to increment or decrement independently for DMA operation.

When the EDMA bit is set, the internal DMARQ flag is cleared. Therefore, subsequent requests are recognized only after the EDMA bit has been set.

Figure 22. DMA Channels



PARALLEL I/O PORTS

Ports P0, P1, P2.

The μPD70327 has three 8-bit parallel I/O ports: P0, P1, and P2. Associated registers are shown in figures 23, 24, and 25. Special-function register (SFR) locations can access these ports as memory. The port lines are individually programmable as inputs or outputs. Many of the port lines have dual functions as port or control lines.

Use the associated port mode control (PMC) and port mode (PM) registers to select the function for a given I/O line.

Figure 23. Port 0 Registers (PMC0, PM0)

$PMC0_7$	0	0	0	0	0	0	0

7 **PMC0 Register** 0

$PM0_7$	$PM0_6$	$PM0_5$	$PM0_4$	$PM0_3$	$PM0_2$	$PM0_1$	$PM0_0$

7 **PM0 Register** 0

		$PMC0_7 = 0$	
Port Pin	$PMC0_7 = 1$	$PM0_n = 1$	$PM0_n = 0$
$P0_7$	CLKOUT	Input port	Output port
$P0_6$	—	Input port	Output port
$P0_5$	—	Input port	Output port
$P0_4$	—	Input port	Output port
$P0_3$	—	Input port	Output port
$P0_2$	—	Input port	Output port
$P0_1$	—	Input port	Output port
$P0_0$	—	Input port	Output port

Figure 24. Port 1 Registers (PMC1, PM1)

$PMC1_7$	$PMC1_6$	$PMC1_5$	$PMC1_4$	$PMC1_3$	0	0	0

7 **PMC1 Register** 0

$PM1_7$	$PM1_6$	$PM1_5$	$PM1_4$	1	1	1	1

7 **PM1 Register** 0

		$PMC1_n = 0$	
Port Pin	$PMC1_7 = 1$	$PM1_n = 1$	$PM1_n = 0$
$P1_7$	READY Input	Input port	Output port
$P1_6$	$\overline{SCKO}$ output	Input port	Output port
$P1_5$	TOUT output	Input port	Output port
$P1_4$	INT input	$\overline{POLL}$ input	Output port
$P1_3$	$\overline{INTAK}$ output	INTP2 input	—
$P1_2$	—	INTP1 input	—
$P1_1$	—	INTP0 input	—
$P1_0$	—	NMI input	—

Figure 25. Port 2 Registers (PMC2, PM2)

$PMC2_7$	$PMC2_6$	$PMC2_5$	$PMC2_4$	$PMC2_3$	$PMC2_2$	$PMC2_1$	$PMC2_0$

7 **PMC2 Register** 0

$PM2_7$	$PM2_6$	$PM2_5$	$PM2_4$	$PM2_3$	$PM2_3$	$PM2_2$	$PM2_1$	$PM2_0$

7 **PM2 Register** 0

		$PMC2_n = 0$	
Port Pin	$PMC2_n = 1$	$PM2_n = 1$	$PM2_n = 0$
$P2_7$	HLDRQ input	Input port	Output port
$P2_6$	$\overline{HLDAK}$ output	Input port	Output port
$P2_5$	$\overline{TC1}$ output	Input port	Output port
$P2_4$	$\overline{DMAAK1}$ output	Input port	Output port
$P2_3$	DMARQ1 input	Input port	Output port
$P2_2$	$\overline{TC0}$ output	Input port	Output port
$P2_1$	$\overline{DMAAK0}$ output	Input port	Output port
$P2_0$	DMARQ0 input	Input port	Output port

Port PT

The analog comparator port (PT) compares each input line to a reference voltage. The reference voltage can be programmed to the V_{TH} input x n/16, where n = 1 to 16. See figure 26.

4g

Figure 26. Port T Mode Register (PMT)

0	0	0	0	PMT_3	PMT_2	PMT_1	PMT_0
7							0

Comparator Reference Voltage (V_{REF})	PMT_3	PMT_2	PMT_1	PMT_0
V_{TH} x 16/16	0	0	0	0
1/16	0	0	0	1
2/16	0	0	1	0
3/16	0	0	1	1
4/16	0	1	0	0
5/16	0	1	0	1
6/16	0	1	1	0
7/16	0	1	1	1
8/16	1	0	0	0
9/16	1	0	0	1
10/16	1	0	1	0
11/16	1	0	1	1
12/16	1	1	0	0
13/16	1	1	0	1
14/16	1	1	1	0
15/16	1	1	1	1

PROGRAMMABLE WAIT STATES

You can generate wait states internally to further reduce the necessity for external hardware. Insertion of these wait states allows direct interface to devices whose access times cannot meet the CPU read/write timing requirements.

When using this function, the entire 1 megabyte of memory address space is divided into 128K-blocks. Each block can be programmed for zero, one, or two wait states, or two plus those added by the external READY signal. The top two blocks are programmed together as one unit.

The appropriate bits in the wait control word (WTC) control wait-state generation. Programming the upper two bits in the wait control word sets the wait-state conditions for the entire I/O address space. Figure 27 shows the memory map for programmable wait-state generation.

Figure 28 diagrams the wait control word. Note that READY pin control is enabled only when two internally generated wait states are selected by the "11" option.

Figure 27. Programmable Wait State Generation

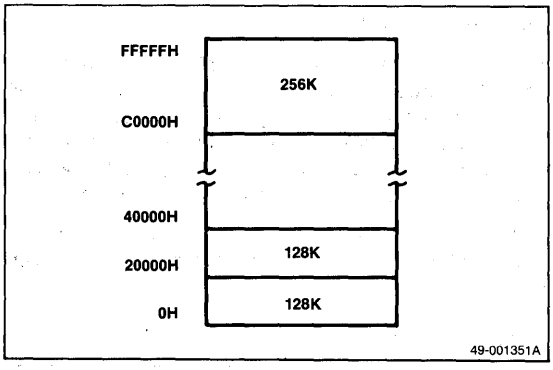


Figure 28. Wait Control Word (WTC)

IO1	IO0	Block 61	Block 60	Block 51	Block 50	Block 41	Block 40
7			Wait Control, High				0

Block 31	Block 30	Block 21	Block 20	Block 11	Block 10	Block 01	Block 00
7			Wait Control, Low				0

Wait States	Block n1	Block n0
0	0	0
1	0	1
2	1	0
2 or more (external control via READYpin)	1	1

n = 0 thru 6

STANDBY MODES

The two low-power standby modes are HALT and STOP. Both modes are entered under software control.

HALT Mode

In HALT mode, the CPU is inactive and thus the chip consumes much less power than when fully operational. The external oscillator remains functional and all internal peripherals are active. Internal status and output port line conditions are maintained. Any unmasked interrupt can release this mode. In the EI state, interrupts are processed subsequently in vector mode. In the DI state, program execution is restarted with the instruction following the HALT instruction.

STOP Mode

The STOP mode allows the largest power reduction while maintaining internal RAM. The oscillator is

stopped, halting the CPU as well as all internal peripherals. Internal status is maintained. Only a reset or NMI can release this mode.

A standby flag in the SFR area is reset by rises in the supply voltage. Its status is maintained during normal operation and standby. The STBC register (figure 29) is not initialized by $\overline{\text{RESET}}$. Use the standby flag to determine whether program execution is returning from standby or from a cold start by setting this flag before entering STOP mode.

Figure 29. Standby Register (STBC)

0	0	0	0	0	0	0	SBF
7			Address xxFE0H				0

SBF	Standby Flag
0	No changes in V_{DD} (standby)
1	Rising edge on V_{DD} (cold start)

SPECIAL-FUNCTION REGISTERS

Table 3 shows the special function register mnemonic, type, address, reset value, and function. The 8 high-order bits of each address (xx) are specified by the IDB register.

SFR area addresses not listed in table 3 are reserved. If read, the contents of these addresses are undefined, and any write operation will be meaningless.

Table 3. Special Function Registers

Name	Byte/Word	Address	Reset Value (Note 2)	R/W (Note 1)	Function
P0	B	xxF00H		R/W	Port 0
PM0	B	xxF01H	FFH	W	Port mode control 0
P1	B	xxF09H	00H		R/W Port 1
PM1	B	xxF09H	FFH	W	Port mode 1
PMC1	B	xxF0AH	00H	W	Port mode control 1
P2	B	xxF10H		R/W	Port 2
PM2	B	xxF11H	FFH	W	Port mode 2
PMC2	B	xxF12H	00H	W	Port mode control 2
PT	B	xxF38H		R	Port T
PMT	B	xxF3BH	00H	R/W	Port mode T
INTM	B	xxF40H	00H	R/W	Interrupt mode
EMS0	B	xxF44H		R/W	External interrupt macro service 0
EMS1	B	xxF45H		R/W	External interrupt macro service 1
EMS2	B	xxF46H		R/W	External interrupt macro service 2
EXIC0	B	xxF4CH	47H	R/W	External interrupt control 0
EXIC1	B	xxF4DH	47H	R/W	External interrupt contol 1
EXIC2	B	xxF4EH	47H	R/W	External interrupt control 2
RXB0	B	xxF60H		R	Receive buffer 0
TXB0	B	xxF62H		W	Transfer buffer 0
SRMS0	B	xxF65H		R/W	Serial receive macro service 0
STMS1	B	xxF66H		R/W	Serial transmit macro service 1
SCM0	B	xxF68H	00H	R/W	Serial communication mode 0
SCC0	B	xxF69H	00H	R/W	Serial communication control 0
BRG0	B	xxF6AH	00H	R/W	Baud rate generator 0
SCE0	B	xxF6BH	00H	R	Serial communication error 0
SEIC0	B	xxF6CH	47H	R/W	Serial error interrupt control 0

Table 3. Special Function Registers (cont)

Name	Byte/Word	Address	Reset Value (Note 2)	R/W (Note 1)	Function
SRIC0	B	xxF6DH	47H	R/W	Serial receive interrupt control 0
STIC0	B	xxF6EH	47H	R/W	Serial transmit interrupt control 0
RXB1	B	xxF70H		R	Receive buffer 1
TXB1	B	xxF72H		W	Transmit buffer 1
SRMS1	B	xxF75H		R/W	Serial receive macro service 1
STMS1	B	xxF76H		R/W	Serial transmit macro service 1
SCM1	B	xxF78H	00H	R/W	Serial communication mode 1
SCC1	B	xxF79H	00H	R/W	Serial communication control 1
BRG1	B	xxF7AH	00H	R/W	Baud rate generator register 1
SCE1	B	xxF7BH	00H	R	Serial communication error 1
SELIC1	B	xxF7CH	47H	R/W	Serial error interrupt control 1
SRIC1	B	xxF7DH	47H	R/W	Serial receive interrupt control 1
STIC1	B	xxF7EH	47H	R/W	Serial transmit interrupt control 1
TM0	W	xxF80H		R/W	Timer register 0
TM0L	B	XXF80H		R/W	Timer register 0 low
TM0H	B	xxF81H		R/W	Timer register 0 high
MD0	W	xxF82H		R/W	Modulo register 0
MD0L	B	xxF82H		R/W	Modulo register 0 low
MD0H	B	xxF83H		R/W	Modulo register 0 high
TM1	W	xxF88H		R/W	Timer register 1
TM1L	B	xxF88H		R/W	Timer register 1 low
TM1H	B	xxF89H		R/W	Timer register 1 high
MD1	W	xxF8AH		R/W	Modulo register 1
MD1L	B	xxF8AH		R/W	Modulo register 1 low
MD1H	B	xxF98BH		R/W	Modulo register 1 high
TMC0	B	xF90H	00H	R/W	Timer control 0
TMC1	B	xxF91H	00H	R/W	Timer control 1
TMMS0	B	xxF94H		R/W	Timer macro service 0
TMMS1	B	xxF95H		R/W	Timer macro service 1
TMMS2	B	xxF96H		R/W	Timer macro service 2
TMIC0	B	xxF9CH	47H	R/W	Timer interrupt control 0
TMIC1	B	xxF9DH	47H	R/W	Timer interrupt control 1
TMIC2	B	xxF9EH	47H	R/W	Timer interrupt control 2
DMAC0	B	xxFA0H		R/W	DMA control 0
DMAM0	B	xxFA1H		R/W	DMA mode 0
DMAC1	B	xxFA2H		R/W	DMA control 1
DMAM1	B	xxFA3H	00H	R/W	DMA mode 1
DIC0	B	xxFACH	47H	R/W	DMA interrupt control 0
DIC1	B	xxFADH	47H	R/W	DMA interrupt control 1
STBC	B	xxFE0H		R/W	Standby control
RFM	B	xxFE1H	FCH	R/W	Refresh mode

Table 3. Special Function Registers (cont)

Name	Byte/Word	Address	Reset Value (Note 2)	R/W (Note 1)	Function
WTC	W	xxFE8H	FFH	R/W	Wait control
WTCL	B	xxFE8H	FFH	R/W	Wait control low
WTCH	B	xxFE9H	FFH	R/W	Wait control high
FLAG	B	xxFEACH	00H	R/W	Flag register
PRC	B	xxFEBH	4EH	R/W	Processor control
TBIC	B	xxFECH	47H	R/W	Time base IRC register
ISPR	B	xxFFCH		R	In service priority register
IDB	B	xxFFFH, FFFFFH		R/W	Internal data area base

Notes:

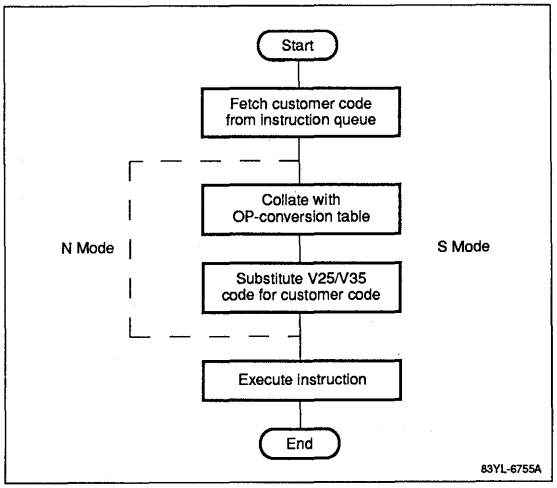
(1) R/W indicates whether register is available for read/write operations.

(2) Reset values not specified are undefined.

SECURITY MODE OPERATION

The security mode of the μPD70327 is designed to protect proprietary user software algorithms by encoding the user's programs resident in external EPROM or ROM memory. The process encodes only the first byte of each opcode via a linear translation table. The decoding process is performed in real time within the μPD70327 and thus does not impact system performance. The flow chart of the conversion process is shown in figure 30.

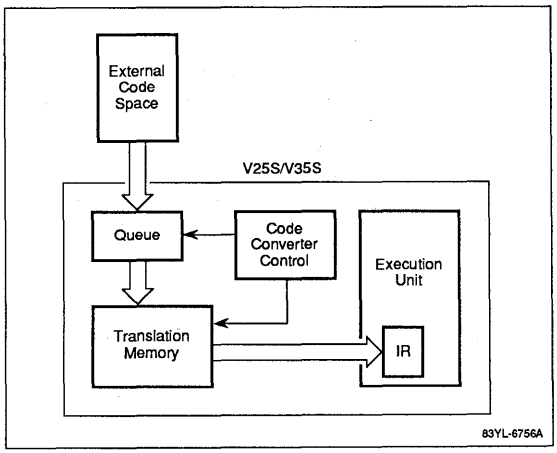
Figure 30. Opcode Translation Flowchart



The translation table is user-defined and is inserted into each μPD70327 mask at the factory. The μPD70327 can be dynamically switched from secure mode to normal mode, thus providing an additional measure of security as well as compatibility with existing ROM versions of V25 software. Note, however, that the V25 Software Guard does not support internal ROM.

The opcode translator is effectively a look-up table that is inserted between the instruction prefetch queue and the instruction register of the μPD70327. A conceptual diagram of this is in figure 31.

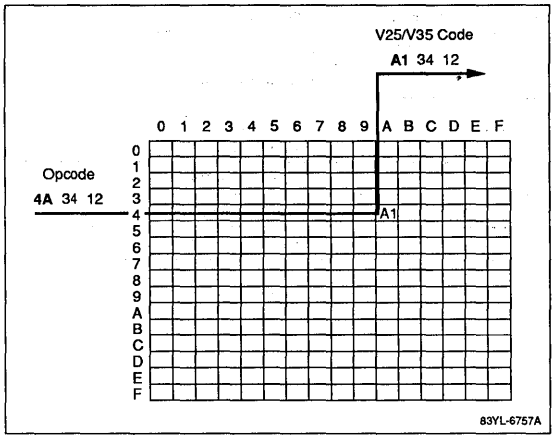
Figure 31. Code Converter Functional Diagram



The code converter uses the encrypted opcode from the prefetch queue as an address, and provides the correct V25 opcode as data to the instruction register. An example of this is shown in figure 32. Again, only the first byte of each opcode is decoded, and subsequent bytes are passed directly from the prefetch unit to the execution unit.

4f

Figure 32. Opcode Converter Translation Table

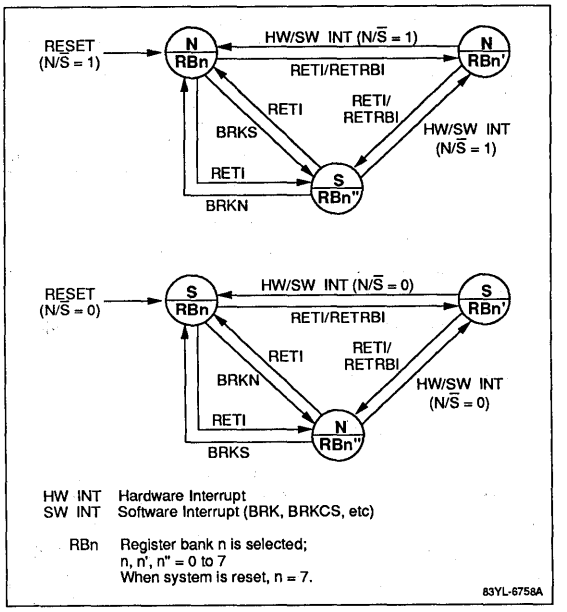


Mode Switching

The transition from normal V25 instruction execution to secure instruction decoding and execution can be performed in either hardware or software. The hardware trigger source is provided by the N/$\overline{S}$ pin of the μPD70327. This pin is listed as an internal connection pin on standard V25 systems, and as such, should be pulled up to V_{DD} through a resistor. Thus a μPD70327 used in a standard V25 design will execute in normal mode identically to the standard V25.

The state of the N/$\overline{S}$ pin is read by the processor at system reset and determines the operational mode of the device at that point. Regardless of the state of this pin, the μPD70327 will begin program execution using register bank 7 as the default register set. (See figure 33.) If the processor samples the N/$\overline{S}$ pin in the low state, the first opcode fetched from the reset address will be decoded using the on-chip translation table. The N/$\overline{S}$ pin has an internal pull-up resistor that will set the device to normal mode operation with no external connections. The N/$\overline{S}$ pin should be set in hardware to a fixed logic state.

Figure 33. Operational Mode State Transition Diagram



Software control of the operating mode is performed by the BRKS (Break for Secure Operation) and the BRKN (Break for Normal Operation) instructions. These opcodes are undefined codes on the standard V25, and should not be ported to standard V25 processor environments. These instructions are detailed in the instruction set section.

The operational state of the μPD70327 is specified by bit 15 (MD) of the Program Status Word (PSW). The remainder of the PSW is identical to that of the standard V25. Since portability of V25 and V25 Software Guard systems is sometimes desired, bit 15 of the PSW should always be written as a logical 1 in standard V25 systems. As with the V25/V35, the upper 4 bits of the PSW cannot be updated by POP; the upper 8 bits of the PSW cannot be updated by MOV. Refer to the PSW diagram shown previously in the "Register Set" section of this data sheet.

As can be seen in figure 33, the N/$\overline{S}$ pin is also sampled upon receipt of an interrupt (either software or hardware). The state of the N/$\overline{S}$ pin will determine the execution mode of the interrupt service routine. The mode of the interrupted routine will be restored by the RETI or RETRBI that terminates the interrupt handler. Software mode changes (via the BRKS and BRKN instructions

described later) will always change the state of the MD bit of the PSW. The MD bit for these software interrupts is restored by the RETI instruction at the end of the mode switch software interrupt.

Operation Timing

Operational execution of the standard V25 and that of the V25 Software Guard are identical regardless of the operational mode selected for the V25 Software Guard. However, since the μPD70327 is a ROMless device, all memory cycles are nominally two system clock periods long. (This is in contrast to the one clock cycle ROM code fetch of the μPD70322.) Due to its ROMless nature, the μPD70327 does not support the $\overline{EA}$ pin of the standard V25, and this pin (labeled IC) should be fixed to a logical high level in the hardware.

ELECTRICAL SPECIFICATIONS

The electrical specifications of the V25 Software Guard and the standard V25 are the same. Refer to the μPD70320/322 (V25) Data Sheet.

INSTRUCTION SET

The instruction set of the V25 Software Guard and the standard V25 are the same except for the addition of two mode change instructions for the V25 Software Guard (BRKS and BRKN) described below.

BRKS Instruction

The BRKS instruction switches operation to security (S) mode and generates a vectored interrupt. In S mode, the fetched operation code is executed after conversion in accordance with the built-in translation table.

The RETI instruction is used to return to the operating mode prior to execution of the BRKS instruction.

BRKN Instruction

The BRKN instruction switches operation to normal (N) mode and generates a vectored interrupt. In N mode, the fetched instruction is executed as a μPD70320/70322 (V25) operation code.

The RETI instruction is used to return to the operating mode prior to execution of the BRKN instruction.

Opcodes

Clock counts and opcodes applicable to the added mode change instructions are in tables 4 and 5.

Table 4. Instruction Clock Counts

Mnemonic	Operand	*Clocks
BRKS	imm8 (≠3)	56 + 10T [44 + 10T]
BRKN	imm8 (≠3)	56 + 10T [44 + 10T]

* Clock counts are specified for RAM enabled and [RAM disabled].

4f

Table 5. Mode Change Instructions

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	No. of Bytes	Flags
BRKS	imm8 (≠3)	(SP – 1, SP – 2) ← PSW, (SP – 3, SP – 4) ← PS, (SP – 5, SP – 6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, MD ← 0 PC ← (n x 4 + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1	1	1	1	0	0	0	1	2	Not applicable
BRKN	imm8 (≠3)	(SP – 1, SP – 2) ← PSW, (SP – 3, SP – 4) ← PS, (SP – 5, SP – 6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, MD ← 1 PC ← (n x 4 + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	0	1	1	0	0	0	1	1	2	Not applicable

NEC

NEC Electronics Inc.

μPD70337 (V35 Software Guard) 16-Bit Microcomputer: Software-Secure, Single-Chip, CMOS

Description

The μPD70337 (V35 Software Guard) is a high-performance, 16-bit, single-chip microcomputer with a 16-bit external data bus. The μPD70337 is fully software compatible with the μPD70108/116 (V20®/V30®) as well as the μPD70320/330 (V25™/V35™).

The μPD70337 allows external executable code to be encrypted by a user-defined translation table. The μPD70337 will automatically decode the encrypted op-codes internally before the instructions are moved into the instruction execution register. As a result, the μPD70337 offers identical performance to the standard V35 even during security mode operation. The security feature may be selected by hardware and/or software, and may be switched from one state to the other under software control.

The μPD70337 has the same complement of internal peripherals as the standard V35 and maintains compatibility with existing drivers. Other than the additional mode select pin, the μPD70337 also maintains pin compatibility with other members of the standard V35 family.

Note: The electrical specifications of the V35 Software Guard and the standard V35 are the same. The instruction sets are also the same except BRKS and BRKN are added to control the Security and Normal operational modes. For electrical specifications and standard instructions, refer to the μPD70330/332 (V35) Data Sheet.

Features

- Security and normal operational modes
- System clock speeds to 8 MHz (16-MHz crystal)
- 16-bit CPU and internal data paths
- Functional compatibility with V35
- Software upward compatible with μPD8086
- New and enhanced V-Series instructions
- 6-byte prefetch queue
- Two-channel on-chip DMA controller
- Minimum instruction cycle: 250 ns at 8 MHz
- Internal 256-byte RAM memory
- 1-megabyte memory address space; 64K-byte I/O space
- Eight internal RAM-mapped register banks
- Four multifunction I/O ports
  - 8-bit analog comparator port
  - 20 bidirectional port lines
  - Four input-only port lines
- Two independent full-duplex serial channels
- Priority interrupt controller
  - Standard vectored service
  - Register bank switching
  - Macroservice
- Pseudo SRAM and DRAM refresh controller
- Two 16-bit timers
- On-chip time base counter
- Programmable wait state generator
- Two standby modes: STOP and HALT

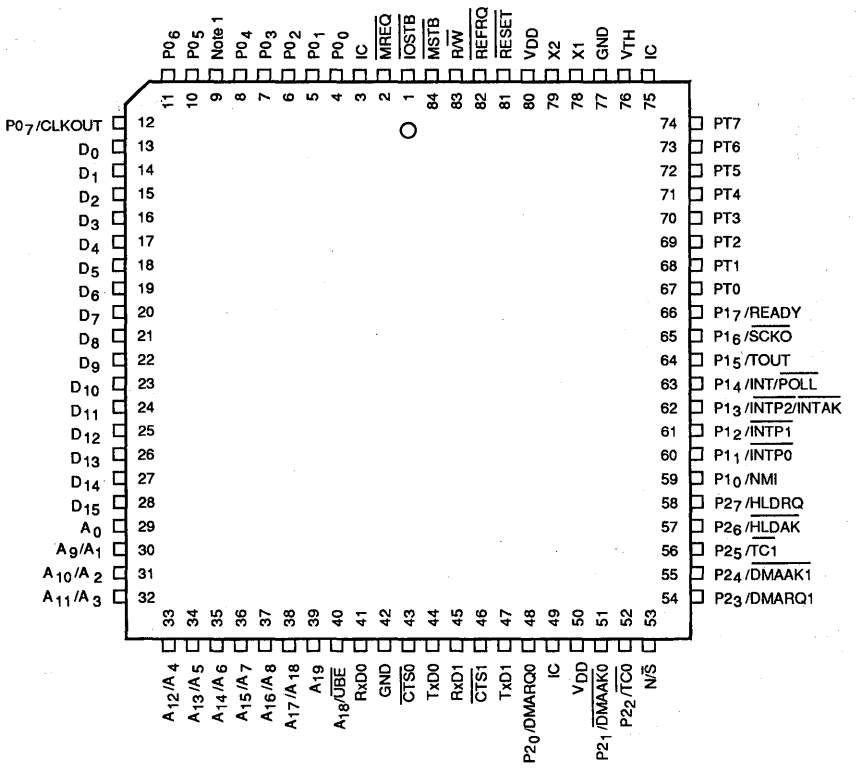
Ordering Information

Part Number	Clock (MHz)	Package
μPD70337L-8-xxx	8	84-pin PLCC
GJ-8-xxx	8	94-pin plastic QFP

V20 and V30 are registered trademarks of NEC Corporation.
V35 and V35 are trademarks of NEC Corporation.

Pin Configurations

84-Pin PLCC



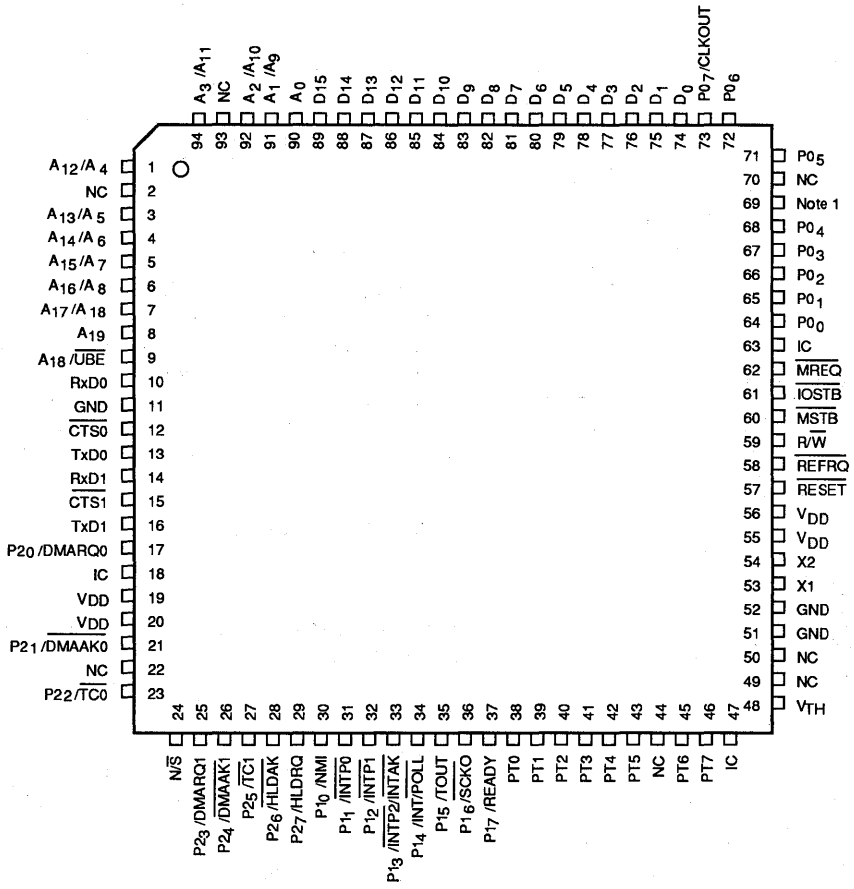
Notes:

(1) Connect pin 9 to GND through a 5- to 10-kΩ resistor.

(2) All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

83SL-7209B

94-Pin Plastic QFP



Notes:

(1) Connect pin 69 to GND through a 5- to 10kΩ resistor.

(2) All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

83SL-7210B

4g

Pin Identification

Symbol	Function
A_0-A_{19}	Address bus outputs
CLKOUT	System clock output
$\overline{CTS0}$	Clear-to-send input, serial channel 0
$\overline{CTS1}$	Clear-to-send input, serial channel 1
D_0-D_{15}	Bidirectional data bus
$\overline{DMAAK0}$	DMA acknowledge output, DMA controller channel 0
$\overline{DMAAK1}$	DMA acknowledge output, DMA controller channel 1
DMARQ0	DMA request input, DMA controller channel 0
DMARQ1	DMA request input, DMA controller channel 1
$\overline{HLDAK}$	Hold acknowledge output
HLDRQ	Hold request input
INT	Interrupt request input
$\overline{INTAK}$	Interrupt acknowledge output
$\overline{INTP0}$	Interrupt request 0 input
$\overline{INTP1}$	Interrupt request 1 input
$\overline{INTP2}$	Interrupt request 2 input
$\overline{IOSTB}$	I/O read or write strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory strobe output
NMI	Nonmaskable interrupt request
N/$\overline{S}$	Normal mode/security mode select input
$P0_0$-$P0_7$	I/O port 0
$P1_0$-$P1_7$	I/O port 1
$P2_0$-$P2_7$	I/O port 2
$\overline{POLL}$	Input on $\overline{POLL}$ synchronizes the CPU and external devices
PT0-PT7	Comparator port input lines
READY	Ready signal input controls insertion of wait states
$\overline{REFRQ}$	DRAM refresh request output
$\overline{RESET}$	Reset signal input
R/$\overline{W}$	Read/write strobe output
RxD0	Receive data input, serial channel 0
RxD1	Receive data input, serial channel 1
$\overline{SCK0}$	Serial clock output
$\overline{TC0}$	Terminal count output; DMA completion, channel 0
$\overline{TC1}$	Terminal count output; DMA completion, channel 1
TOUT	Timer output
TxD0	Transmit data output, serial channel 0
TxD1	Transmit data output, serial channel 1
$\overline{UBE}$	Upper byte enable
X1, X2	Connections to external frequency control source (crystal, ceramic resonator, or clock)
V_{DD}	+5-volt power source input (two pins)
V_{TH}	Threshold voltage input to comparator circuits
GND	Ground reference (two pins)
IC	Internal connection; must be tied to V_{DD} externally through a pullup resistor

PIN FUNCTIONS

A_0-A_{19} (Address Bus)

To support dynamic RAMs, the 20-bit address is multiplexed on 11 lines. When $\overline{MREQ}$ is asserted, A_9-A_{17} are valid. When $\overline{MSTB}$ or $\overline{IOSTB}$ is asserted, A_1-A_8 and A_{18} are valid. A_{18} is also multiplexed with $\overline{UBE}$ and is valid when $\overline{MREQ}$ is asserted. Therefore A_{18} is active throughout the bus cycle. A_{19} and A_0 are not multiplexed but have dedicated pins and are valid throughout the bus cycle.

CLKOUT (Clock Out)

The system clock (CLK) is distributed from the internal clock generator to the CPU and output to peripheral hardware at the CLKOUT pin. This pin is sampled at system reset.

$\overline{CTS0}$ (Clear-to-Send 0)

This is the CTS pin of the channel 0 serial interface. In asynchronous mode, a low-level input on $\overline{CTS0}$ enables transmit operation. In I/O interface mode, $\overline{CTS0}$ is the receive clock pin.

$\overline{CTS1}$ (Clear-to-Send 1)

This is the CTS pin of the channel 1 serial interface. In asynchronous mode, a low-level input on $\overline{CTS1}$ enables transmit operation.

D_0-D_{15} (Data Bus)

D_0-D_{15} is the 16-bit data bus.

$\overline{DMAAK0}$ and $\overline{DMAAK1}$ (DMA Acknowledge)

These are the DMA acknowledge outputs of the DMA controller, channels 0 and 1. Signals are not output during DMA memory-to-memory transfer operations (burst mode, single-step mode).

$\overline{DMARQ0}$ and $\overline{DMARQ1}$ (DMA Request)

These are the DMA request inputs of the DMA controller, channels 0 and 1.

$\overline{HLDAK}$ (Hold Acknowledge)

The $\overline{HLDAK}$ output signal indicates that the hold request (HLDRQ) has been accepted. When $\overline{HLDAK}$ is active (low), the following lines go to the high-impedance state with internal 4700-Ω pullup resistors: A_0-A_{19}, D_0-D_7, $\overline{IOSTB}$, $\overline{MREQ}$, $\overline{MSTB}$, $\overline{REFRQ}$, and R/$\overline{W}$.

HLDRQ (Hold Request)

The HLDRQ input from an external device requests that the μPD70335 relinquish the address, data, and control buses to an external bus master.

INT (Interrupt)

The INT input is a vectored interrupt request from an external device that can be masked by software. The active high level is detected in the last clock cycle of an instruction. The external device confirms that the INT interrupt request has been accepted by the $\overline{INTAK}$ signal output from the CPU.

The INT signal must be held high until the first $\overline{INTAK}$ signal is output. Together with $\overline{INTAK}$, INT is used for operation with an interrupt controller such as μPD71059.

$\overline{INTAK}$ (Interrupt Acknowledge)

The $\overline{INTAK}$ output is the acknowledge signal for the software-maskable interrupt request INT. The $\overline{INTAK}$ signal goes low when the CPU accepts INT. The external device inputs the interrupt vector to the CPU via data bus D_0-D_7 in synchronization with $\overline{INTAK}$.

$\overline{INTP0}$, $\overline{INTP1}$, $\overline{INTP2}$ (Interrupt from Peripheral 0, 1, 2)

The $\overline{INTPn}$ inputs (n = 0, 1, 2) are external interrupt requests that can be masked by software. The $\overline{INTPn}$ input is detected at the effective edge specified by external interrupt mode register INTM.

The $\overline{INTPn}$ input is also used to release the HALT mode.

$\overline{IOSTB}$ (I/O Strobe)

A low-level output on $\overline{IOSTB}$ indicates that the I/O bus cycle has been initiated and that the I/O address output on A_0-A_{15} is valid.

$\overline{MREQ}$ (Memory Request)

A low-level output on $\overline{MREQ}$ indicates that the memory or I/O bus cycle has started and that address bits A_0, A_9-A_{17}, A_{18} and A_{19} are valid.

$\overline{MSTB}$ (Memory Strobe)

Together with $\overline{MREQ}$ and R/$\overline{W}$, $\overline{MSTB}$ controls memory-accessing operations. $\overline{MSTB}$ should be used either to enable data buffers or as a data strobe. During memory write, a low-level output on $\overline{MSTB}$ indicates that data on the data bus is valid. A low-level output on $\overline{MSTB}$ indicates that multiplexed address bits A_1-A_8, A_{18}, and $\overline{UBE}$ are valid.

NMI (Nonmaskable Interrupt)

The NMI input is an interrupt request that cannot be masked by software. The NMI is always accepted by the CPU; therefore, it has priority over any other interrupt.

The NMI input is detected at the effective edge specified by external interrupt mode register INTM. Sampled in each clock cycle, NMI is accepted when the active level lasts for some clock cycles. When the NMI is accepted, a number 2 vector interrupt is generated after completion of the instruction currently being executed.

The NMI input is also used to release the CPU standby mode.

N/$\overline{S}$ (N Mode/S Mode)

Normal or security mode is selected by a fixed high level (N) or low level (S) at this pin. This pin is sampled at system reset and at the acceptance of interrupts.

$P0_0$-$P0_7$ (Port 0)

Port 0 is an 8-bit bidirectional I/O port.

$P1_0$-$P1_7$ (Port 1)

Lines $P1_4$-$P1_7$ are individually programmable as an input, output, or control function. The status of $P1_0$- $P1_3$ can be read but these lines are always control functions.

$P2_0$-$P2_7$ (Port 2)

$P2_0$-$P2_7$ are the lines of port 2, an 8-bit bidirectional I/O port. These lines can also be used as control signals for the on-chip DMA controllers.

$\overline{POLL}$ (Poll)

The $\overline{POLL}$ input is checked by the POLL instruction. If the level is low, execution of the next instruction is initiated. If the level is high, the $\overline{POLL}$ input is checked every five clock cycles until the level becomes low. The $\overline{POLL}$ functions are used to synchronize the CPU program and the operation of external devices.

Note: $\overline{POLL}$ is effective when $P1_4$ is specified for the input port mode; otherwise, $\overline{POLL}$ is assumed to be at low level when the POLL instruction is executed.

PT0-PT7 (Port with Comparator)

The PT input is compared with a threshold voltage that is programmable to one of 16 voltage steps individually for each of the eight lines.

READY (Ready)

After READY is de-asserted low, the CPU will synchronize and insert at least two wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than normal execution allows.

$\overline{REFRQ}$ (Refresh Request)

This output pulse can refresh nonstatic RAM. It can be programmed to meet system specifications and is internally synchronized so that refresh cycles do not interfere with normal CPU operation.

$\overline{RESET}$ (Reset)

This input signal is asynchronous. A low on $\overline{RESET}$ for a certain duration resets the CPU and all on-chip peripherals regardless of clock operation. The reset operation has priority over all other operations.

The reset signal is used for normal initialization/startup and also for releasing the STOP or HALT mode. After the reset signal returns high, program execution begins from address FFFF0H.

R/$\overline{W}$ (Read/Write Strobe)

When the memory bus cycle is initiated, the R/$\overline{W}$ signal output to external hardware indicates a read (high- level) or write (low-level) cycle. It can also control the direction of bidirectional buffers.

RxD0, RxD1 (Receive Data 0, 1)

These pins input data from serial channels 0 and 1.

In the asynchronous mode, when receive operation is enabled, a low level on the RxD0 or RxD1 input pin is recognized as the start bit and receive operation is initiated.

In the I/O interface mode (channel 0 only), receive data is input to the serial register at the rising edge of the receive clock.

$\overline{SCK0}$ (Serial Clock)

The $\overline{SCK0}$ output is the transmit clock of serial channel 0.

$\overline{TC0}$, $\overline{TC1}$ (Terminal Count 0, 1)

The $\overline{TC0}$ and $\overline{TC1}$ outputs go low when the terminal count of DMA service channels 0 and 1, respectively, reach zero, indicating DMA completion.

TOUT (Timer Output)

The TOUT signal is a square-wave output from the internal timer.

TxD0, TxD1 (Transmit Data 0, 1)

These pins output data from serial channels 0 and 1.

In the asynchronous mode, the transmit signal is in a frame format that consists of a start bit, 7 or 8 data bits (least significant bit first), parity bit, and stop bit. The TxD0 and TxD1 pins become mark state (high level) when transmit operation is disabled or when the serial register has no transmit data.

In the I/O interface mode (channel 0 only), the frame has 8 data bits and the most significant bit is transmitted first.

$\overline{UBE}$ (Upper Byte Enable)

$\overline{UBE}$ is a high-order memory bank selection signal output. $\overline{UBE}$ and A_0 determine which bytes of the data bus will be used. UBE is used with A_0 to select the even/odd banks as follows.

Operand	$\overline{UBE}$	A_0	Number of Bus Cycles
Even address word	0	0	1
Odd address word	0	1	2
	1	0	
Even address byte	1	0	1
Odd address byte	0	1	1

X1, X2 (Clock Control)

The frequency of the internal clock generator is controlled by an external crystal or ceramic resonator connected across pins X1 and X2. The crystal frequency is the same as the clock generator frequency f_X. By programming the PRC register, the system clock frequency f_{CLK} is selected as f_X divided by 2, 4, or 8.

As an alternative to the crystal or ceramic resonator, the positive and negative phases of an external clock (with frequency f_X) can be connected to pins X1 and X2.

V_{DD} (Power Supply)

+5-volt power source (two pins).

V_{TH} (Threshold Voltage)

Comparator port PT0-PT7 uses threshold voltage V_{TH} to determine the analog reference points. The actual threshold to each comparator line is programmable to V_{TH} x n/16 where n = 1 to 16.

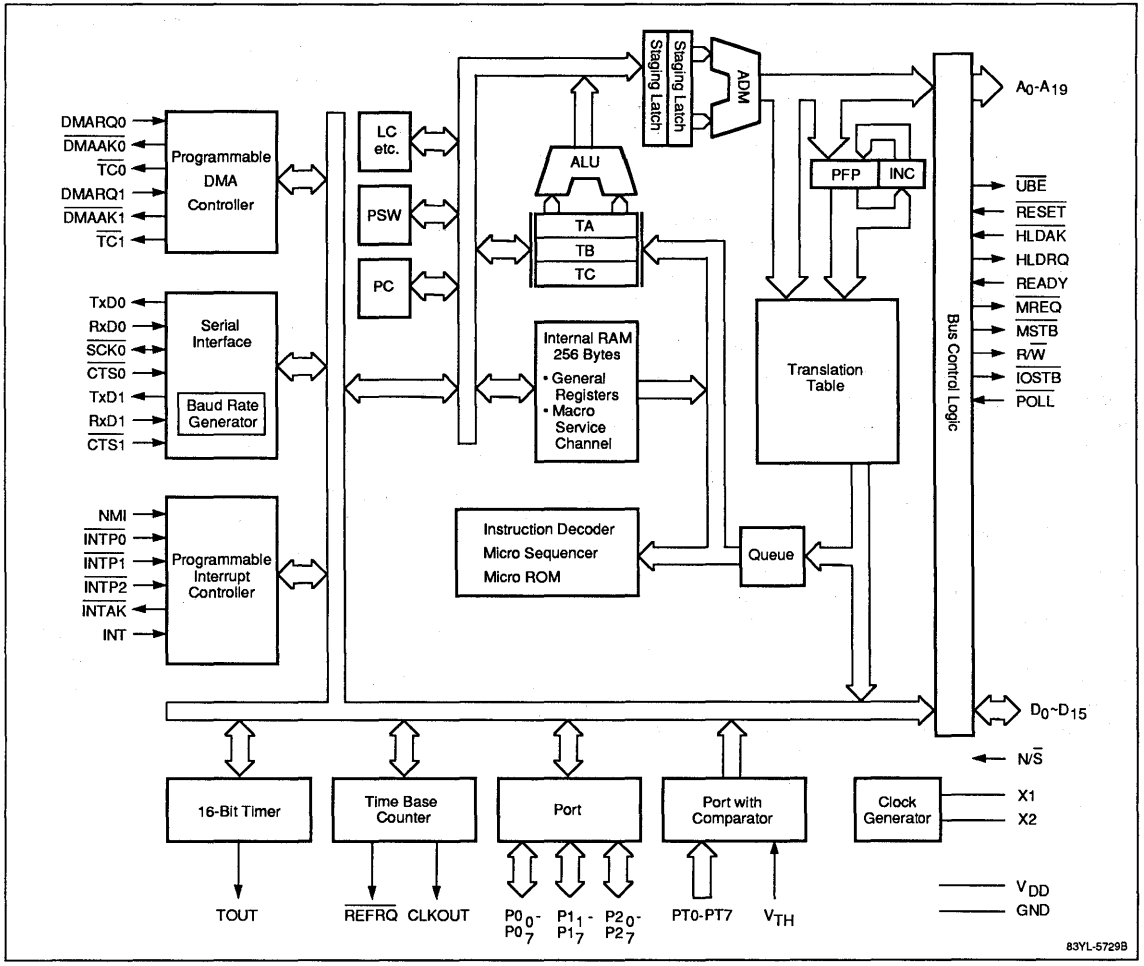
GND (Ground)

Ground reference (two pins).

IC (Internal Connection)

Internal connection; must be tied to V_{DD} externally through a 10-kΩ to 20-kΩ resistor.

μPD70337 Block Diagram



4g

V25/V35 FAMILY

This addition to the V25/V35 family of high-integration microcomputers—the V35 Software Guard (V35S)—offers a direct DRAM interface with an external 16-bit data path. It supports both native V25/V35 operational modes as well as the enhanced security mode of operation. The security mode allows external code memory to be encrypted, thus preventing the unauthorized inspection of proprietary algorithms.

V35S Comparison to V25S

The V35S is fully software compatible with the μPD8088/8086 and the μPD70118/70116 (V20/V30) instruction set. Because the V35S is a ROMless part, all code must be located in external memory. The external memory may contain both encrypted opcodes and/or normal V-series opcodes.

The V35S contains the same core and peripherals as the V25 Software Guard (V25S). The main difference between the two is confined to the external bus interface and bus control logic. While the V25S is designed with an 8-bit external interface, the V35S provides the full 16-bit external data path.

The V35S external data bus is non-multiplexed; however, the 20-bit address bus is multiplexed to provide a direct DRAM style RAS/CAS bus cycle. As a result, the nominal bus cycle is three CLKOUT states. During the first bus state, the address lines output the high 9 bits of the physical address, A_{17}-A_9. During the second bus state, the address lines output low address bits A_8-A_1. Address lines A_{19} and A_0 are not multiplexed and are valid during the entire bus cycle. Address line A_{18} is multiplexed with the Upper Byte Enable signal (UBE) and is valid as an address during bus state one. During 16-bit transfers to odd addresses (UBE = 0 and A_0 = 1), two 8-bit bus cycles are performed.

The memory control signals of the V35S and V25S are identical; however, certain timing specifications are different, particularly for static memory interfaces. Refer to the V35 Data Sheet for these timing parameters. Typically, the MREQ signal is used to generate the DRAM RAS control signal, and the MSTB signal is used to generate the CAS signal. Like the V25S, the V35S provides an output from the internal refresh control unit, which is also typically gated into the DRAM RAS signal.

Another function of the V35S that is different from the V25S is the operation of the READY input pin. This pin is sampled in the middle of the second bus cycle (BAW1) on the V25S; the V35S samples one clock period later in the middle of BAW2.

Other than these bus controller differences, the V35S is identical to the V25S in its operation. All internal peripherals are programmed and operate in the same manner as those of the V25S. The instruction sets of the two processors are identical, and internally both processors operate on 16-bit data paths. Additionally, the security mode of the V35S functions identically to that of the V25S, although it fetches 16-bits of opcode per fetch cycle.

V35S Comparison to Standard V35

The V35S contains the same peripherals and maintains full upward functional compatibility with the standard V35. All internal functional units operate and are programmed the same as those of the V35. The instruction set is also a direct superset of the standard V35, containing all instructions of the V35 and adding only two to select the secure/normal operational modes.

The pinouts of the V35S and the V35 are the same except for two pins.

(1) $\overline{EA}$ on the V35 is IC on the ROMless V35S.

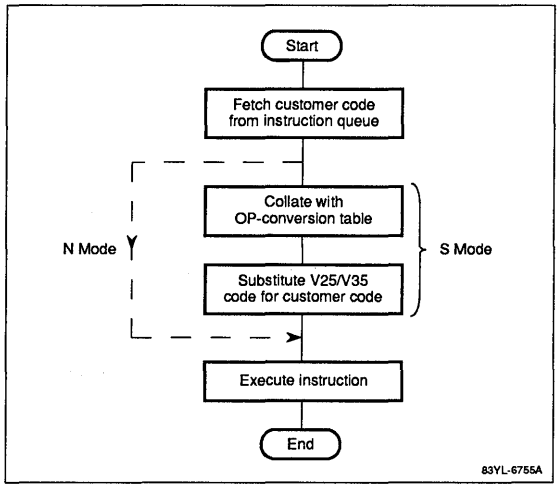
(2) N/$\overline{S}$ on the V35S is IC on the V35.

All other pins on the V35S perform identical functions to corresponding pins on the V35.

SECURITY MODE OPERATION

The security mode of the μPD70337 is designed to protect proprietary user software algorithms by encoding the user's programs resident in external system EPROM or ROM memory. The process encodes only the first byte of each opcode via a linear translation table. The decoding process is performed in real time within the μPD70337 and thus does not impact system performance. The flowchart of the conversion process is shown in figure 1.

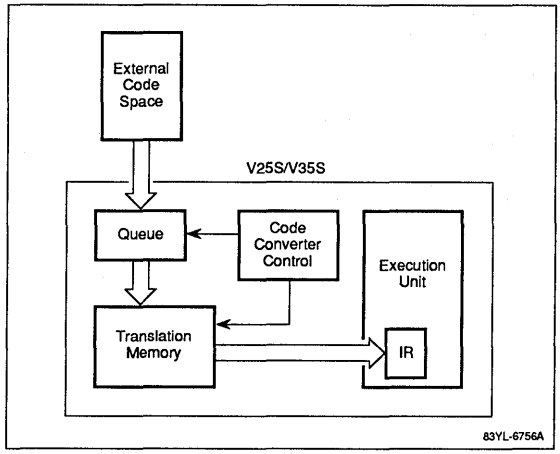
Figure 1. Opcode Translation Flowchart



The user-defined translation table is inserted into each μPD70337 mask at the factory. The μPD70337 can be dynamically switched from secure mode to normal mode, thus providing an additional measure of security as well as compatibility with existing ROM versions of V35 software. Note, however, that the V35 Software Guard does not support internal ROM.

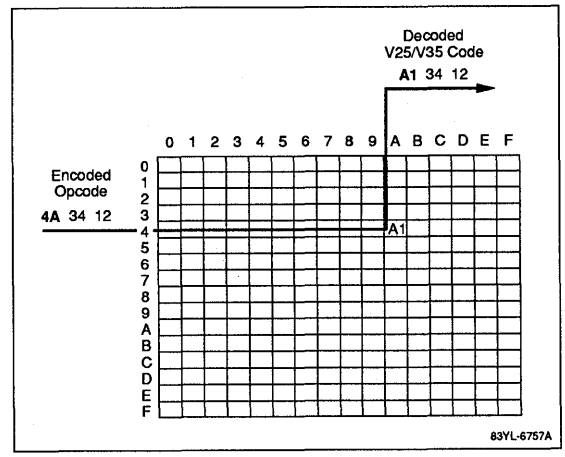
The opcode translator is effectively a look-up table that is inserted between the instruction prefetch queue and the instruction register of the μPD70337. A conceptual diagram of this is in figure 2.

Figure 2. Code Converter Functional Diagram



The code converter uses the encrypted opcode from the prefetch queue as an address, and provides the correct V35 opcode as data to the instruction register. An example of this is shown in figure 3. Again, only the first byte of each opcode is decoded, and subsequent bytes are passed directly from the prefetch unit to the execution unit.

Figure 3. Opcode Converter Translation Table



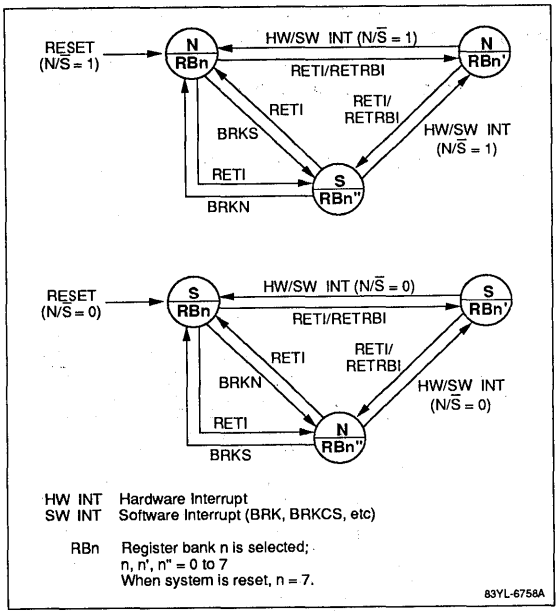
Mode Switching

The transition from normal V35 instruction execution to secure instruction decoding and execution can be performed in either hardware or software. The hardware trigger source is provided by the N/$\overline{S}$ pin of the μPD70337. This pin is listed as an internal connection pin on standard V35 systems, and as such, should be pulled up to V_{DD} through a resistor. Thus, a μPD70337 used in a standard V35 design will execute in normal mode identically to the standard V35.

The state of the N/$\overline{S}$ pin is read by the processor at system reset and determines the operational mode of the device at that point. Regardless of the state of this pin, the μPD70337 will begin program execution using register bank 7 as the default register set. (See figure 4.) If the processor samples the N/$\overline{S}$ pin in the low state, the first opcode fetched from the reset address will be decoded using the on-chip translation table. The N/$\overline{S}$ pin has an internal pull-up resistor that will set the device to normal mode operation with no external connections. The N/$\overline{S}$ pin should be set in hardware to a fixed logic state.

4g

Figure 4. Operational Mode State Transition Diagram



Software control of the operating mode is performed by the BRKS (Break for Secure Operation) and the BRKN (Break for Normal Operation) instructions. These opcodes are undefined codes on the standard V35 and should not be ported to standard V35 processor environments. These instructions are detailed in the instruction set section.

The operational state of the μPD70337 is specified by bit 15 (MD) of the Program Status Word (PSW). The remainder of the PSW (figure 5) is identical to that of the standard V35. Since portability of V35 and V35 Software Guard systems is sometimes desired, bit 15 of the PSW should always be written as a logical 1 in standard V35 systems. As with the V25/V35, the upper 4 bits of the PSW cannot be updated by POP; the upper 8 bits of the PSW cannot be updated by MOV.

Consult the μPD70327 (V25 Software Guard) data sheet for additional details of secure mode operation

Figure 5. Program Status Word (PSW)

MD	RB2	RB1	RB0	V	DIR	IE	BRK
15							8

S	Z	F1	AC	F0	P	BRKI	CY
7							0

Status Flag		Control Flags	
V	Overflow bit	DIR	Direction of string processing
S	Sign	IE	Interrupt enable
Z	Zero	BRK	Break (after every instruction)
AC	Auxiliary carry	RBn	Current register bank flags
P	Parity	BRKI	I/O trap enable
CY	Carry	F0, F1	General-purpose user flags (accessed by flag SFR)
		MD	Normal/security mode select

Operation Timing

Operational execution of the standard V35 and that of the V35 Software Guard are identical regardless of the operational mode selected for the V35 Software Guard. However, since the μPD70337 is a ROMless device, all memory cycles are nominally three system clock periods long. (This is in contrast to the one clock cycle ROM code fetch of the μPD70332.) Due to its ROMless nature, the μPD70337 does not support the $\overline{EA}$ pin of the standard V35, and this pin (labeled IC) should be fixed to a logical high level in the hardware.

ELECTRICAL SPECIFICATIONS

The electrical specifications of the V35 Software Guard and the standard V35 are the same. Refer to the μPD70330/332 (V35) Data Sheet.

INSTRUCTION SET

The instruction sets of the V35 Software Guard and the standard V35 are the same except for the addition of two mode change instructions for the V35 Software Guard (BRKS and BRKN) described below.

BRKS Instruction

The BRKS instruction switches operation to security (S) mode and generates a vectored interrupt. In S mode, the fetched operation code is executed after conversion in accordance with the built-in translation table.

The RETI instruction is used to return to the operating mode prior to execution of the BRKS instruction.

BRKN Instruction

The BRKN instruction switches operation to normal (N) mode and generates a vectored interrupt. In N mode, the fetched instruction is executed as a μPD70330/70332 (V35) operation code.

The RETI instruction is used to return to the operating mode prior to execution of the BRKN instruction.

Opcodes

Clock counts and opcodes applicable to the added mode change instructions are in tables 1 and 2.

Table 1. Instruction Clock Counts

Mnemonic	Operand	*Clocks
BRKS	imm8 (≠3)	56 + 10T [44 + 10T]
BRKN	imm8 (≠3)	56 + 10T [44 + 10T]

* Clock counts are specified for internal RAM enabled and [Internal RAM disabled].

Table 2. Mode Change Instructions

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	No. of Bytes	Flags
BRKS	imm8 (≠3)	(SP – 1, SP – 2) ← PSW, (SP – 3, SP – 4) ← PS, (SP – 5, SP – 6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, MD ← 0 PC ← (n x 4 + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1	1	1	1	0	0	0	1	2	Not applicable
BRKN	imm8 (≠3)	(SP – 1, SP – 2) ← PSW, (SP – 3, SP – 4) ← PS, (SP – 5, SP – 6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, MD ← 1 PC ← (n x 4 + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	0	1	1	0	0	0	1	1	2	Not applicable

NEC
NEC Electronics Inc.

μPD79011 16-Bit Microcomputer: Single-Chip, CMOS, With Built-In RTOS

Description

The μPD79011 is an upgraded μPD70322 (V25™) single-chip microcomputer with a built-in real-time operating system (RTOS).

The μPD79011 provides high-speed multitask processing particularly suited for real-time event processing and as a kernel of an embedded control system for process control and data processing applications.

The RTOS kernel provides extensive system calls for task synchronization, control, and communication as well as interrupt and time management.

The μPD79011 instruction set is the same as the V25 instruction set. The μPD79011 hardware is also identical to the standard V25, but uses 6K of the internal ROM for RTOS system code. Refer to the V25 Data Sheet.

Features

- Real-time multitask processing
- Supports five types of system calls
  - Task management
  - Communication management
  - Memory management
  - Time management
  - Interrupt management
- High-speed response to events
  - System call processing shortens time to 41 μs (minimum) when operated at 8 MHz
  - High-speed task switching using V25 register banks
- Flexibility to perform status changes by event driven task scheduling function
- System clock: 8 MHz maximum
- V25 hardware compatibility
- CMOS technology
- Development tools
  - V25 software can be used without modification
  - Relocatable assembler (RA70320)
  - C compiler (CC70116)
  - Concurrent CP/M®, MS-DOS®, VMS™, and UNIX™ base

Ordering Information

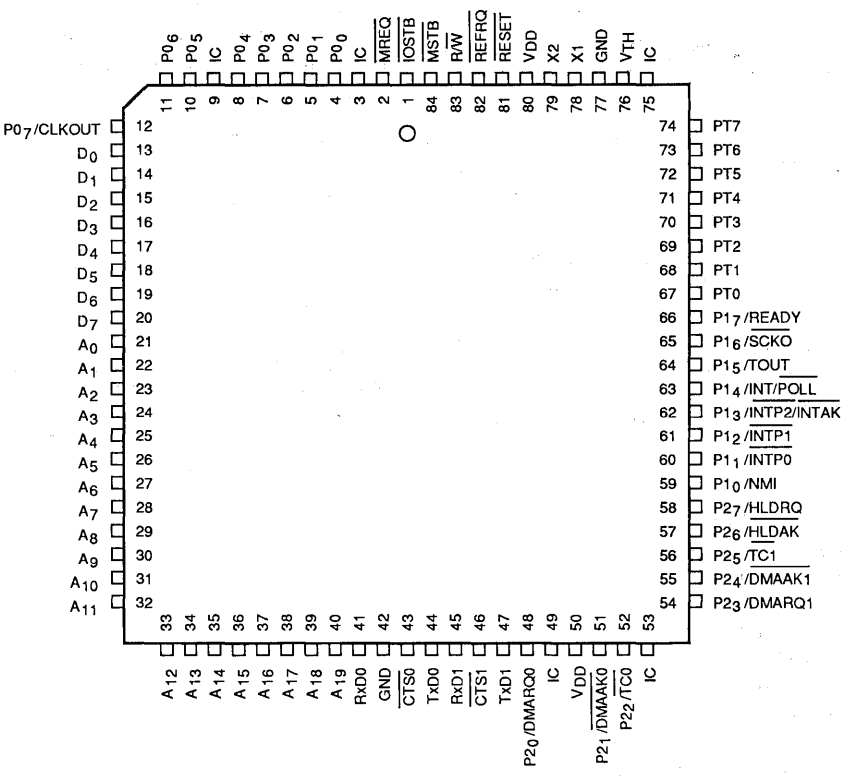
Part Number	Clock	Package
μPD79011L-8	8 MHz	84-pin PLCC
GJ-8	8 MHz	94-pin plastic QFP

4h

V25 is a trademark of NEC Corporation.
CP/M is a registered trademark of Digital Research, Inc.
MS-DOS is a registered trademark of Microsoft Corporation.
VMS is a trademark of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories.

Pin Configurations

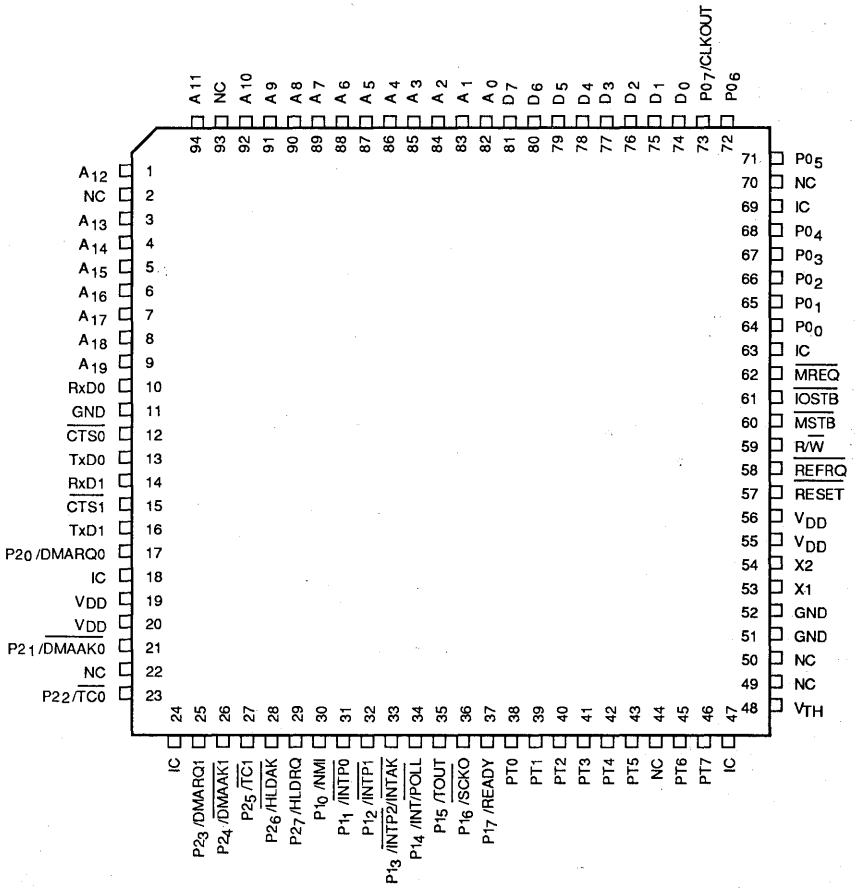
84-Pin PLCC



Note: All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

83SL-6719B

94-Pin Plastic QFP



Note: All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

83SL-6720B

4h

Pin Identification

Symbol	Function
A_0-A_{19}	Address bus outputs
CLKOUT	System clock output
$\overline{CTS0}$	Clear to send channel 0 input
$\overline{CTS1}$	Clear to send channel 1 input
D_0-D_7	Bidirectional data bus
$\overline{IOSTB}$	I/O strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory strobe output
$P0_0$-$P0_7$	I/O port 0
$P1_0$/NMI	Port 1 input line; nonmaskable interrupt
$P1_1$-$P1_2$/ $\overline{INTP0}$-$\overline{INTP1}$	Port 1 input lines; Interrupt requests from peripherals 0 and 1
$P1_3$/$\overline{INTP2}$/INTAK	Port 1 input line; Interrupt requests from peripheral 2; Interrupt acknowledge output
$P1_4$/INT/$\overline{POLL}$	I/O port 1; Interrupt request input; I/O poll input
$P1_5$/TOUT	I/O port 1; Timer out
$P1_6$/$\overline{SCKO}$	I/O port 1; Serial clock output
$P1_7$/READY	I/O port 1; Ready input
$P2_0$/DMARQ0	I/O port 2; DMA request 0
$P2_1$/$\overline{DMAAK0}$	I/O port 2; DMA acknowledge 0
$P2_2$/$\overline{TC0}$	I/O port 2; DMA terminal count 0
$P2_3$/DMARQ1	I/O port 2; DMA request 1
$P2_4$/$\overline{DMAAK1}$	I/O port 2; DMA acknowledge 1
$P2_5$/$\overline{TC1}$	I/O port 2; DMA terminal count 1
$P2_6$/$\overline{HLDAK}$	I/O port 2; Hold acknowledge output
$P2_7$/HLDRQ	I/O port 2; Hold request input
PT0-PT7	Comparator port input lines
$\overline{REFRQ}$	Refresh pulse output
$\overline{RESET}$	Reset input
RxD0	Serial receive data channel 0 input
RxD1	Serial receive data channel 1 input
R/$\overline{W}$	Read/write output
TxD0	Serial transmit data, channel 0 input
TxD1	Serial transmit data, channel 1 input
X1, X2	Crystal connection terminals
V_{DD}	Positive power supply voltage
V_{TH}	Threshold voltage input for comparator
GND	Ground reference
IC	Internal connection

PIN FUNCTIONS

A_0-A_{19} (Address Bus)

A_0-A_{19} is the 20-bit address bus used to access all external devices.

CLKOUT (System Clock)

This is the internal system clock. It can be used to synchronize external devices to the CPU.

$\overline{CTSn}$, RxDn, TxDn, $\overline{SCKO}$ (Clear to Send, Receive Data, Transmit Data, Serial Clock Out)

The two serial ports (channels 0 and 1) use these lines for transmitting and receiving data, handshaking, and serial clock output.

D_0-D_7 (Data Bus)

D_0-D_7 is the 8-bit external data bus.

DMARQn, $\overline{DMAAKn}$, $\overline{TCn}$ (DMA Request, DMA Acknowledge, Terminal Count)

These are the control signals to and from the on-chip DMA controller.

$\overline{HLDAK}$ (Hold Acknowledge)

The $\overline{HLDAK}$ output (active low) informs external devices that the CPU has released the system bus.

HLDRQ (Hold Request)

The HLDRQ input (active high) is used by external devices to request the CPU to release the system bus to an external bus master. The following lines go into a high-impedance state with internal 4.7-kΩ pullup resistors: A_0-A_{19}, D_0-D_7, $\overline{MREQ}$, R/$\overline{W}$, $\overline{MSTB}$, $\overline{REFRQ}$, and $\overline{IOSTB}$.

INT (Interrupt Request)

INT is a maskable, active-high, vectored request interrupt. After assertion, external hardware must provide the interrupt vector number.

$\overline{INTAK}$ (Interrupt Acknowledge)

After INT is asserted, the CPU will respond with $\overline{INTAK}$ (active low) to inform external devices that the interrupt request has been granted.

INTP0-INTP2 (External Interrupt)

$\overline{INTP0}$-$\overline{INTP2}$ allow external devices to generate interrupts. Each can be programmed to be rising or falling edge triggered.

$\overline{IOSTB}$ (I/O Strobe)

$\overline{IOSTB}$ is asserted during read and write operations to external I/O.

$\overline{MREQ}$ (Memory Request)

$\overline{MREQ}$ (active low) informs external memory that the current bus cycle is a memory access bus cycle.

$\overline{MSTB}$ (Memory Strobe)

$\overline{MSTB}$ (active low) is asserted during read and write operations to external memory.

NMI (Nonmaskable Interrupt)

NMI cannot be masked through software and is typically used for emergency processing. Upon execution, the interrupt starting address is obtained from interrupt vector number 2. NMI can release the standby modes and can be programmed to be either rising or falling edge triggered.

$P0_0$-$P0_7$ (Port 0)

$P0_0$-$P0_7$ are the lines of port 0, an 8-bit bidirectional parallel I/O port.

$P1_0$-$P1_7$ (Port 1)

The status of $P1_0$-$P1_3$ can be read but these lines are always control functions. $P1_4$-$P1_7$ are the remaining lines of parallel port 1; each line is individually programmable as either an input, an output, or a control function.

$P2_0$-$P2_7$ (Port 2)

$P2_0$-$P2_7$ are the lines of port 2, an 8-bit bidirectional parallel I/O port. The lines can also be used as control signals for the on-chip DMA controller.

$\overline{POLL}$ (Poll)

Upon execution of the POLL instruction, the CPU checks the status of this pin and, if low, program execution continues. If high, the CPU checks the level of the line every five clock cycles until it is low. $\overline{POLL}$ can be used to synchronize program execution to external conditions.

PT0-PT7 (Comparator Port)

PT0-PT7 are inputs to the analog comparator port.

READY (Ready)

After READY is de-asserted low, the CPU synchronizes and inserts at least two wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than normal execution.

$\overline{REFRQ}$ (Refresh)

This active-low output pulse can refresh nonstatic RAM. It can be programmed to meet system specifications and is internally synchronized so that refresh cycles do not interfere with normal CPU operation.

$\overline{RESET}$ (Reset)

A low on $\overline{RESET}$ resets the CPU and all on-chip peripherals. $\overline{RESET}$ can also release the standby modes. After $\overline{RESET}$ returns high, program execution begins from address FFFF0H.

R/$\overline{W}$ (Read/Write)

R/$\overline{W}$ output allows external hardware to determine if the current operation is a read or a write cycle. It can also control the direction of bidirectional buffers.

TOUT (Timer Out)

TOUT is the square-wave output signal from the internal timer.

X1, X2 (Crystal Connections)

The internal clock generator requires an external crystal across these terminals. By programming the PRC register, the system clock frequency can be selected as the oscillator frequency (f_{OSC}) divided by 2, 4, or 8.

V_{DD} (Power Supply)

Two positive power supply pins (V_{DD}) reduce internal noise.

V_{TH} (Threshold Voltage)

The comparator port uses this pin to determine the analog reference point. The actual threshold to each comparator line is programmable to V_{TH} x n/16 where n = 1 to 16.

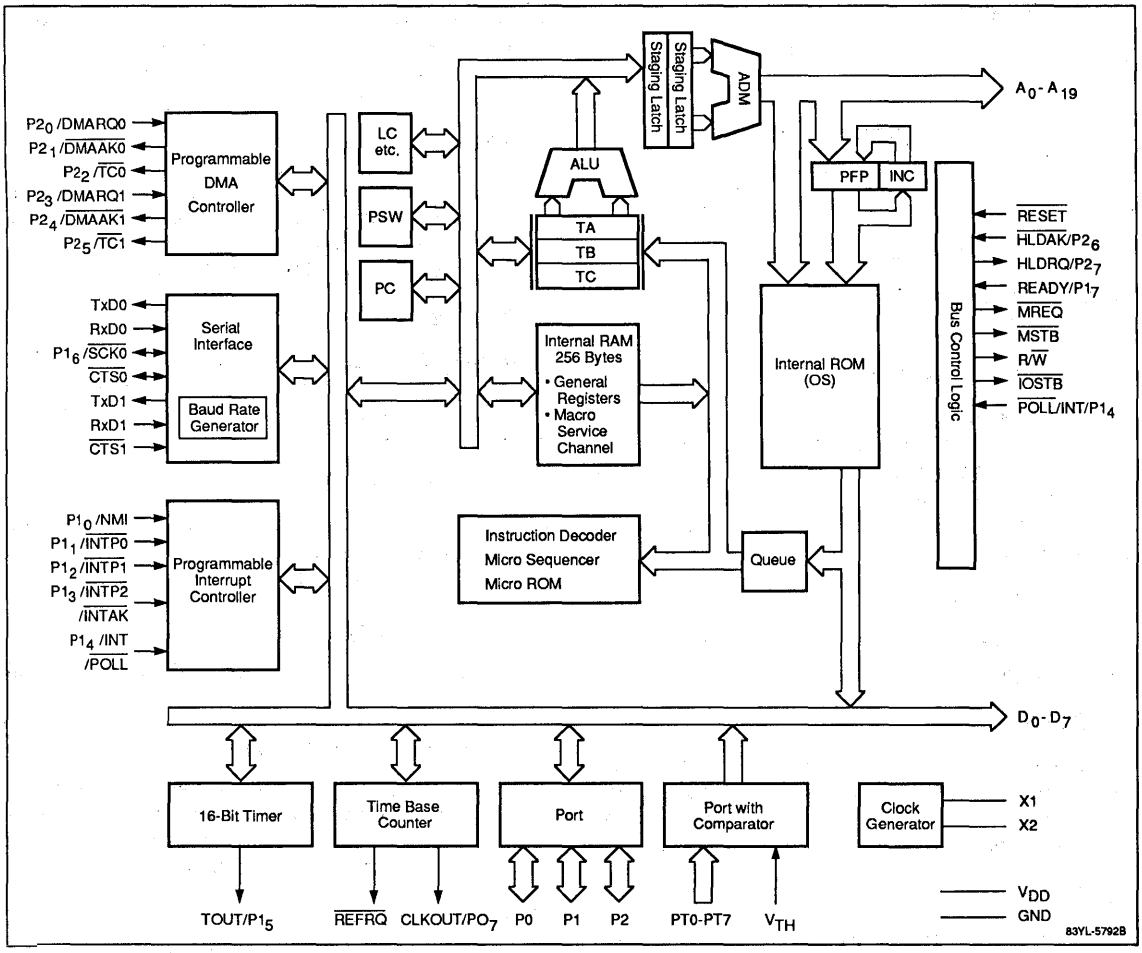
GND (Ground)

Two ground connections reduce internal noise.

IC (Internal Connection)

All IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

μPD79011 Block Diagram



ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = 25°C

Supply voltage, V_{DD}	−0.5 to 7.0 V
Input voltage, V_I	−0.5 to V_{DD}+0.5 (≤ +7.0 V)
Output voltage, V_O	−0.5 to V_{DD}+0.5 (≤ +7.0 V)
Threshold voltage, V_{TH}	−0.5 to V_{DD}+0.5 (≤ +7.0 V)
Output current low, I_{OL}	Each output pin 4.0 mA (Total 50 mA)
Output current high, I_{OH}	Each output pin −2.0 mA (Total −20 mA)
Operating temperature range, T_{OPT}	−40 to +85°C
Storage temperature range, T_{STG}	−65 to +150°C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Comparator Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%

Parameter	Symbol	Min	Max	Unit
Accuracy	VA_{COMP}		±100	mV
Threshold voltage	V_{TH}	0	V_{DD}+0.1	V
Comparison time	t_{COMP}	64	65	t_{CYK}
PT input voltage	V_{IPT}	0	V_{DD}	V

Capacitance

T_A = 25°C; V_{DD} = 0 V

Parameter	Symbol	Min	Max	Unit	Conditions
Input capacitance	C_I		10	pF	f = 1 MHz; unmeasured pins returned to ground
Output capacitance	C_O		20	pF	
I/O capacitance	C_{IO}		20	pF	

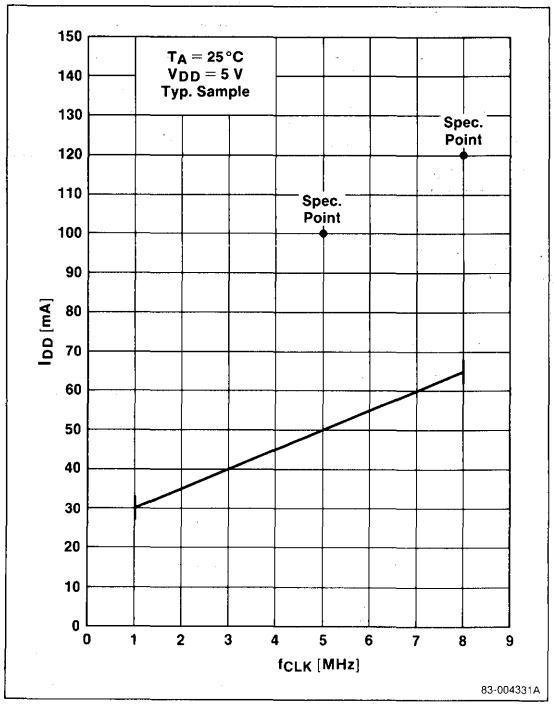
DC Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%

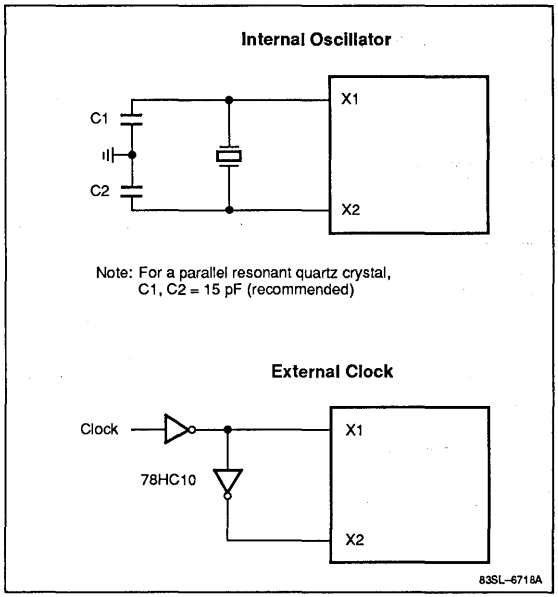
Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Supply current, operating mode	I_{DD1}		43	100	mA	f_{CLK} = 5 MHz
			58	120	mA	f_{CLK} = 8 MHz
Supply current, HALT mode	I_{DD2}		17	40	mA	f_{CLK} = 5 MHz
			21	50	mA	f_{CLK} = 8 MHz
Supply current, STOP mode	I_{DD3}		10	30	μA	
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2		V_{DD}	V	All except $\overline{RESET}$, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 x V_{DD}		V_{DD}	V	$\overline{RESET}$, $P1_0$/NMI, X1, X2
Output voltage, low	V_{OL}			0.45	V	I_{OL} = 1.6 mA
Output voltage, high	V_{OH}	V_{DD} − 1.0			V	I_{OH} = −0.4 mA
Input current	I_{IN}			±20	μA	$P1_0$/NMI; V_I = 0 to V_{DD}
Input leakage current	I_{LI}			±10	μA	All except $P1_0$/NMI; V_I = 0 to V_{DD}
Output leakage current	I_{LO}			±10	μA	V_O = 0 to V_{DD}
V_{TH} supply current	I_{TH}		0.5	1.0	mA	V_{TH} = 0 to V_{DD}
Data retention voltage	V_{DDR}	2.5		5.5	V	

4h

Supply Current vs Clock Frequency



External System Clock Control Source



Recommended Oscillator Components

Ceramic Resonator		Capacitors	
Manufacturer	**Product No.**	**C1 (pF)**	**C2 (pF)**
Kyocera	KBR-10.0M	33	33
Murata Mfg.	CSA.10.0MT	47	47
	CSA16.0MX040	30	30
TDK	FCR10.M2S	30	30
	FCR16.0M2S	15	6

AC Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%

Parameter	Symbol	Min	Max	Unit	Conditions
V_{DD} rise, fall time	t_{RVD}, t_{FVD}	200		μs	STOP mode
Input rise, fall time	t_{IR}, t_{IF}		20	ns	Except X1, X2, $\overline{RESET}$, NMI
Input rise, fall time (Schmitt)	t_{IRS}, t_{IFS}		30	ns	$\overline{RESET}$, NMI
Output rise, fall time	t_{OR}, t_{OF}		20	ns	Except CLKOUT
X1 cycle time	t_{CYX}	98	250	ns	5-MHz CPU clock
		62	250	ns	8-MHz CPU clock
X1 width, low	t_{WXL}	35		ns	5-MHz CPU clock
		20		ns	8-MHz CPU clock
X1 width, high	t_{WXH}	20		ns	5-MHz CPU clock
		20		ns	8-MHz CPU clock
X1 rise, fall time	t_{XR}, t_{XF}		20	ns	8-MHz CPU clock
CLKOUT cycle time	t_{CYK}	125	2000	ns	fx/2, T = t_{CYK}

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
CLKOUT width, low	t_{WKL}	0.5T – 15		ns	Note 1
CLKOUT width, high	t_{WKH}	0.5T – 15		ns	
CLKOUT rise, fall time	t_{KR}, t_{KF}		15	ns	
Address delay time	t_{DKA}	15	90	ns	
Address valid to input data valid	t_{DADR}		T(n+1.5) – 90	ns	Note 2
$\overline{MREQ}$ to data delay time	t_{DMRD}		T(n+1) – 75	ns	
$\overline{MSTB}$ to data delay time	t_{DMSD}		T(n+0.5) – 75	ns	
$\overline{MREQ}$ to $\overline{TC}$ delay time	t_{DMRTC}		0.5T+50	ns	
$\overline{MREQ}$ to $\overline{MSTB}$ delay time	t_{DMRMS}	0.5T – 35	0.5+35	ns	
$\overline{MREQ}$ width, low	t_{WMRL}	T(n+1) – 30		ns	
Address hold time	t_{HMA}	0.5T – 30		ns	
Input data hold time	t_{HMDR}	0		ns	
Next control setup time	t_{SCC}	T – 25		ns	
$\overline{TC}$ width, low	t_{WTCL}	2T – 30		ns	
Address data output	t_{DADW}	0.5T+50		ns	
$\overline{MREQ}$ delay time	t_{DAMR}	0.5T – 30		ns	
$\overline{MSTB}$ delay time	t_{DAMS}	T – 30		ns	
$\overline{MSTB}$ width, low	t_{WMSL}	T(n+0.5) – 30		ns	
Data output setup time	t_{SDM}	T(n+1) – 50		ns	
Data output hold time	t_{HMDW}	0.5T – 30		ns	
$\overline{IOSTB}$ delay time	t_{DAIS}	0.5T – 30		ns	
$\overline{IOSTB}$ to data input	t_{DISD}		T(n+1) – 90	ns	
$\overline{IOSTB}$ width, low	t_{WISL}	T(n+1) – 30		ns	
Address hold time	t_{HISA}	0.5T – 30		ns	
Data input hold time	t_{HISDR}	0		ns	
Output data setup time	t_{SDIS}	T(n+1) – 50		ns	
Output data hold time	t_{HISDW}	0.5T – 30		ns	
Next DMARQ setup time	t_{SDADQ}		T	ns	Demand mode
DMARQ hold time	t_{HDADQ}	0		ns	Demand mode
$\overline{DMAAK}$ read width, low	t_{WDMRL}	T(n+1.5) – 30		ns	
$\overline{DMAAK}$ to TC delay time	t_{DDATC}		0.5T+50	ns	
$\overline{DMAAK}$ write width, low	t_{WDMWL}	T(n+1) – 30		ns	
$\overline{REFRQ}$ delay time	t_{DARF}	0.5T – 30		ns	
$\overline{REFRQ}$ width, low	t_{WRFL}	(n+1)T – 30		ns	
Address hold time	t_{HRFA}	0.5T – 30		ns	
$\overline{RESET}$ width, low	t_{WRSL1}	30		ms	STOP mode release; power-on reset
$\overline{RESET}$ width, low	t_{WRSL2}	5		μs	System warm reset
$\overline{MREQ}$, $\overline{IOSTB}$ to READY setup time	t_{SCRY}		T(n – 1) – 100	ns	n ≥ 2
$\overline{MREQ}$, $\overline{IOSTB}$ to READY hold time	t_{HCRY}	T(n – 1)		ns	n ≥ 2

4h

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{HLDAK}$ output delay time	t_{DKHA}		80	ns	
Bus control float to $\overline{HLDAK}$ ↓	t_{CFHA}	T − 50		ns	
$\overline{HLDAK}$ ↑ to control output time	t_{DHAC}	T − 50		ns	
HLDRQ ↓ to control output time	t_{DHQC}	3T +30		ns	
$\overline{HLDAK}$ width, low	t_{WHAL}		T	ns	
HLDRQ setup time	t_{SHQK}	30		ns	
HLDRQ to $\overline{HLDAK}$ delay time	t_{DHQHA}		3T+160	ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
$\overline{INTP}$, DMARQ setup time	t_{SIQK}	30		ns	
$\overline{INTP}$, DMARQ width, high	t_{WIQH}	8T		ns	
$\overline{INTP}$, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{POLL}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{CTS}$ width, low	t_{WCTL}	2T		ns	
INTR setup time	t_{SIRK}	30		ns	
INTR hold time	t_{HIAIQ}	0		ns	
$\overline{INTAK}$ width, low	t_{WIAL}	2T − 30		ns	
$\overline{INTAK}$ delay time	t_{DKIA}		80	ns	
$\overline{INTAK}$ width, high	t_{WIAH}	T − 30		ns	
$\overline{INTAK}$ to data delay time	t_{DIAD}		2T − 130	ns	
$\overline{INTAK}$ to data hold time	t_{HIAD}	0	0.5T	ns	
$\overline{SCKO}$ cycle time	t_{CYTK}	1000		ns	
$\overline{SCKO}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{SCKO}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	
TxD hold time	t_{HTKD}	20		ns	
$\overline{CTS0}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{CTS0}$ (RSCK) width, high	t_{WSRH}	420		ns	
$\overline{CTS0}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

Notes: (1) T = CPU clock period (t_{CYK}) (2) n = number of wait states inserted

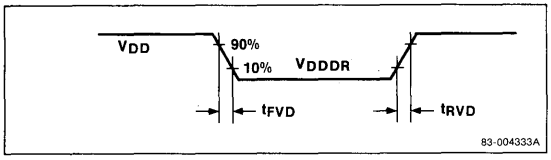
STOP Mode Data Retention Characteristics

T_A = −10 to +70°C

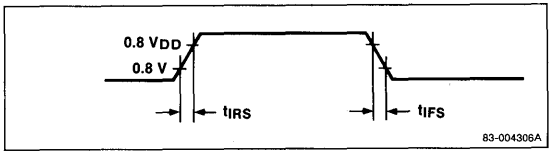
Parameter	Symbol	Min	Max	Unit
Data retention voltage	V_{DDDE}	2.5	5.5	V
V_{DD} rise time	t_{LFVD}	200		μs
V_{DD} fall time	$t_{FVD°C}$	200		μs

Timing Waveforms

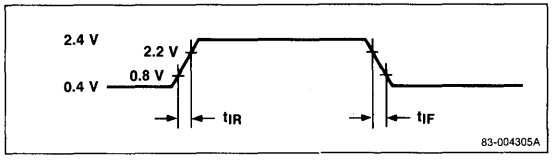
Stop Mode Data Retention Timing



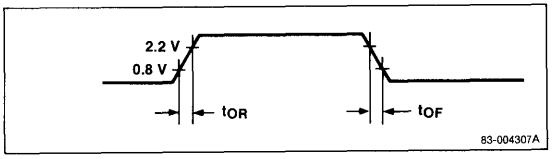
AC Input Waveform 2 (RESET, NMI)



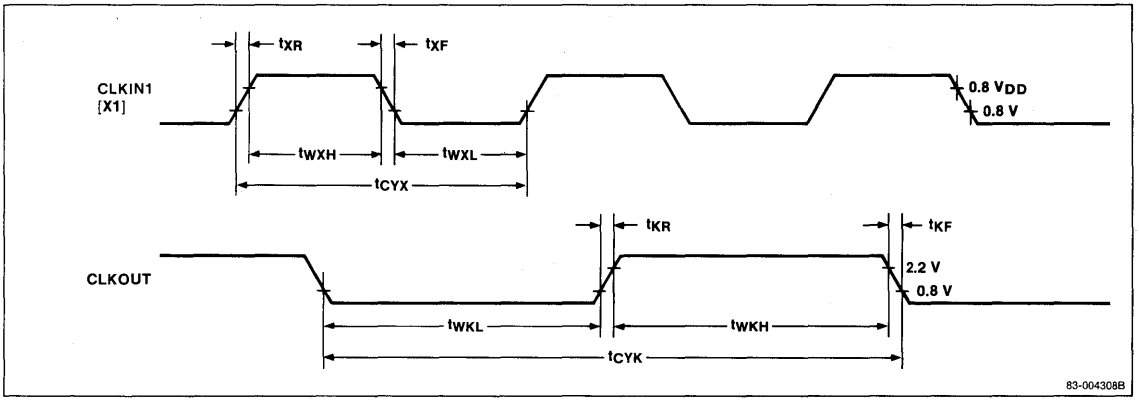
AC Input Waveform 1 (Except X1, X2, RESET, NMI)



AC Output Test Point (Except CLKOUT)

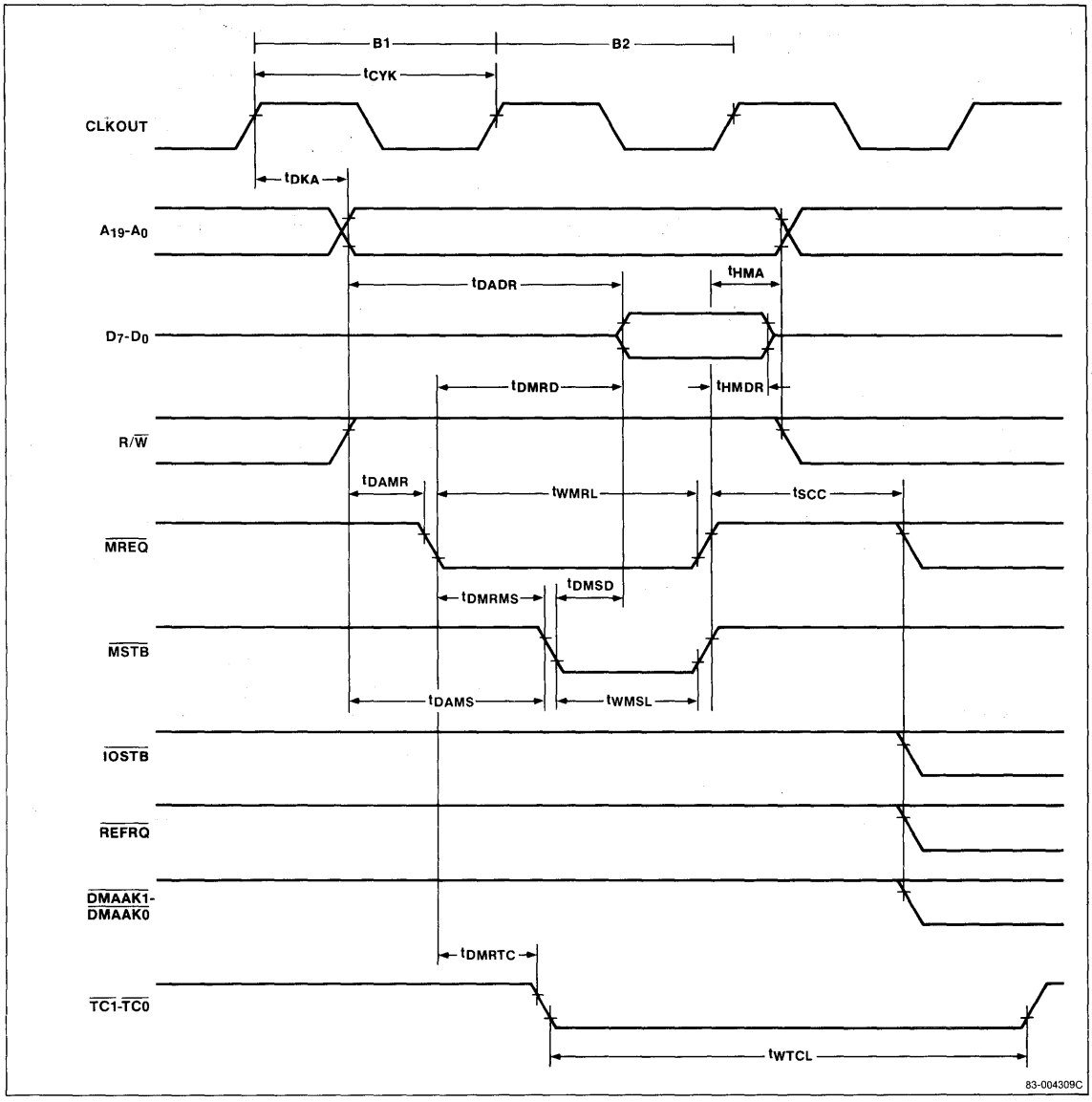


Clock In and Clock Out

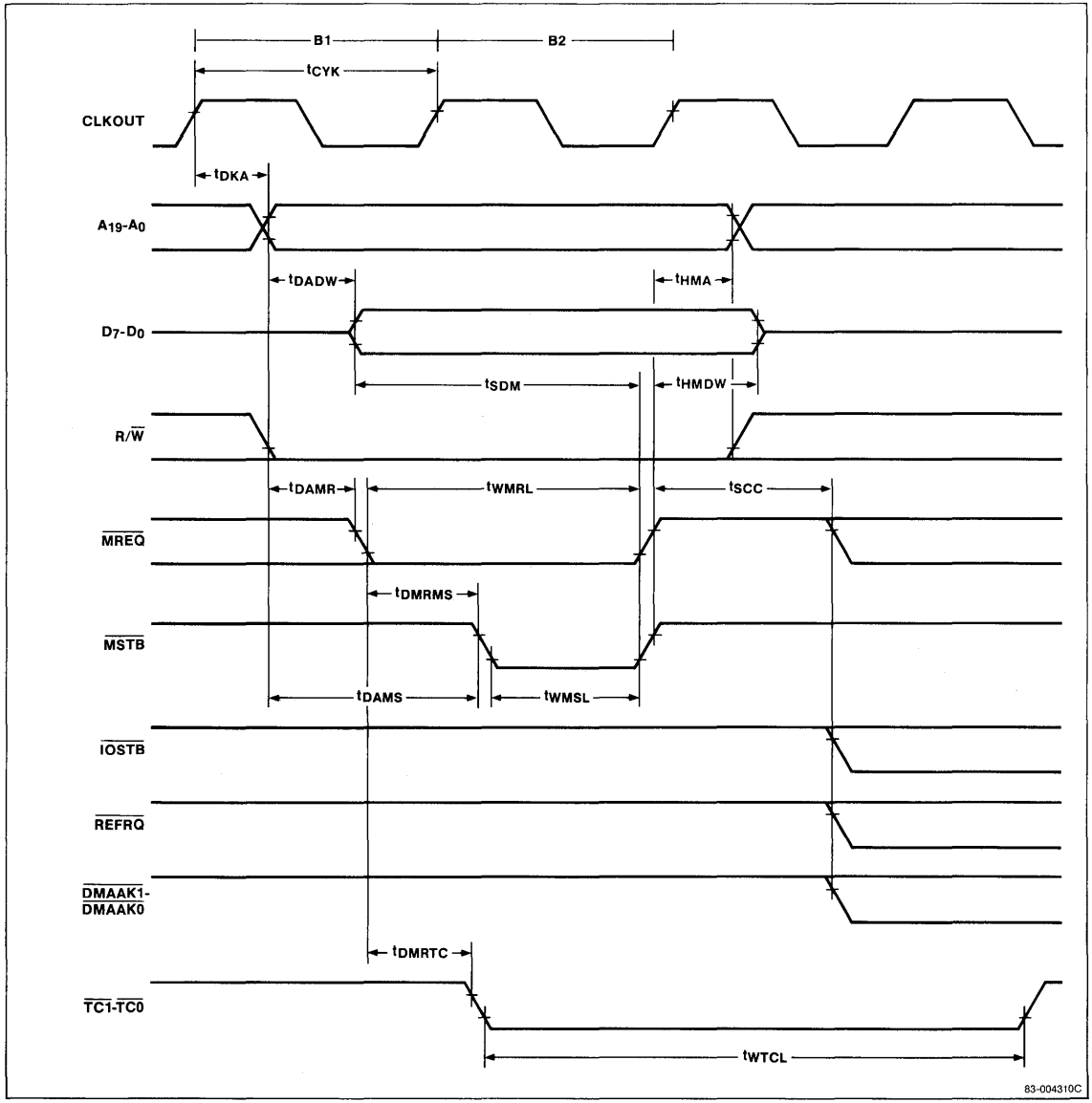


4h

Memory Read

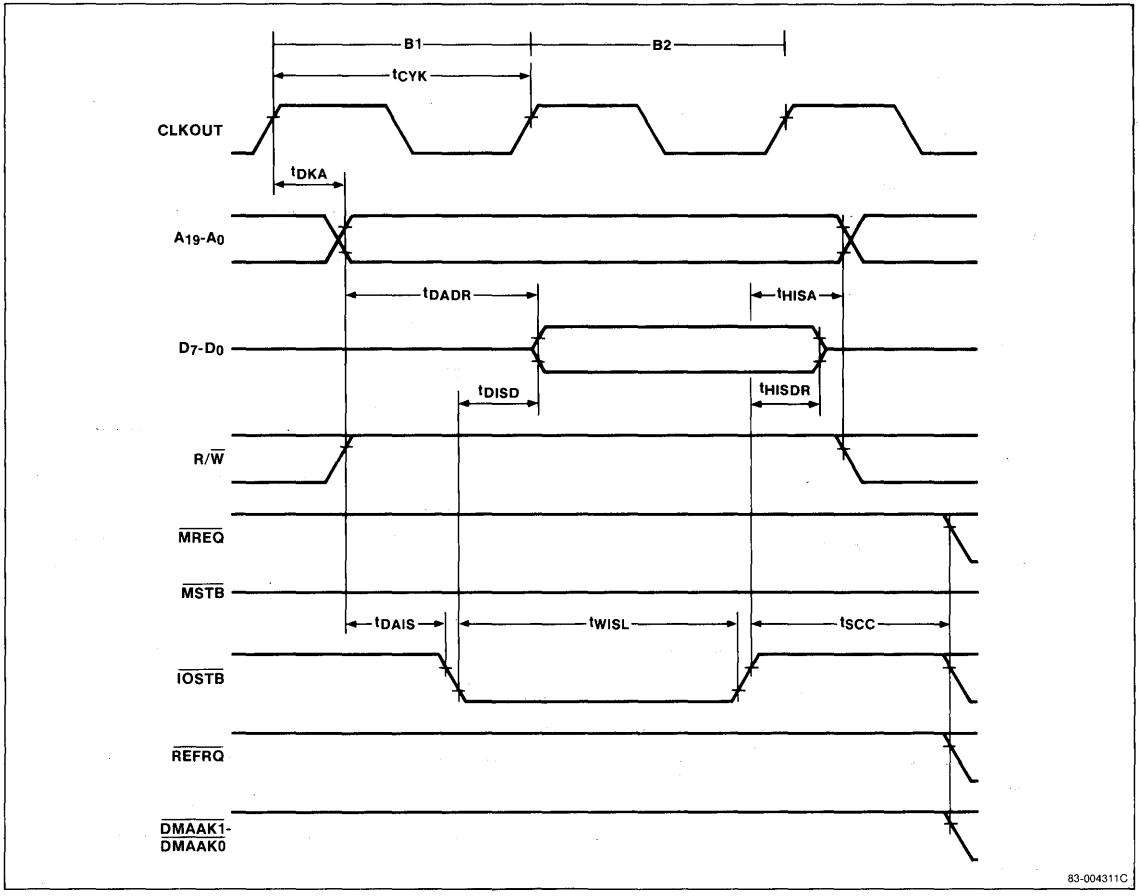

B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADR
tHMA
D7-D0
tDMRD
tHMDR
R/W
tDAMR
tWMRL
tSCC
MREQ
tDMRMS
tDMSD
MSTB
tDAMS
tWMSL
IOSTB
REFRQ
DMAAK1-
DMAAK0
tDMRTC
TC1-TC0
tWTCL
83-004309C

Memory Write

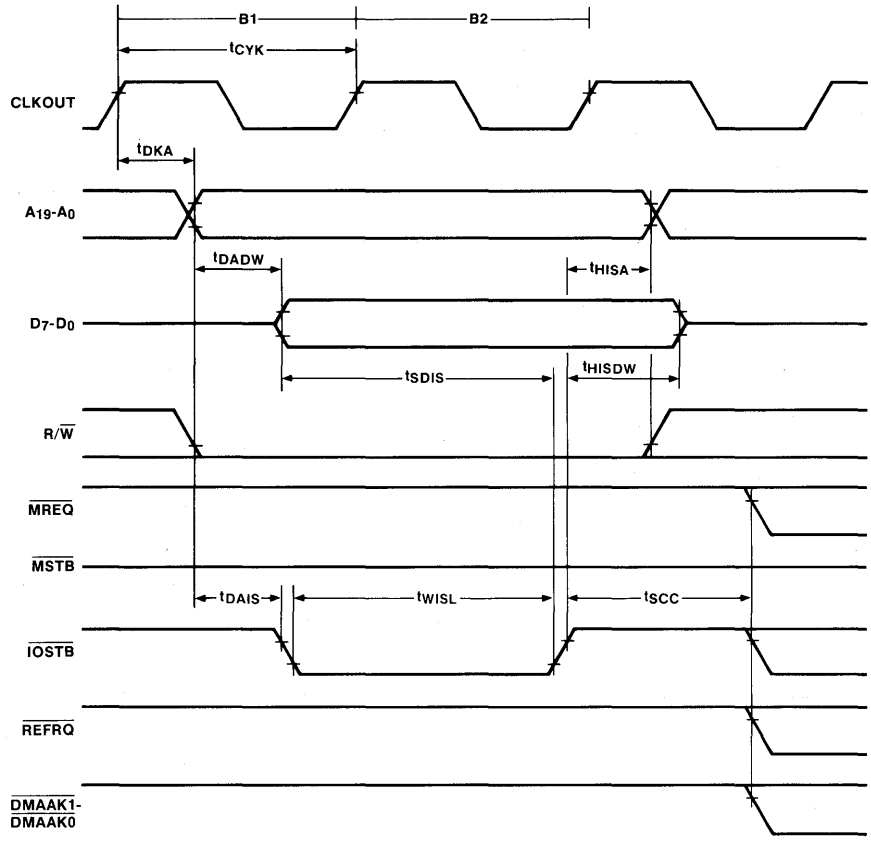


4h

I/O Read

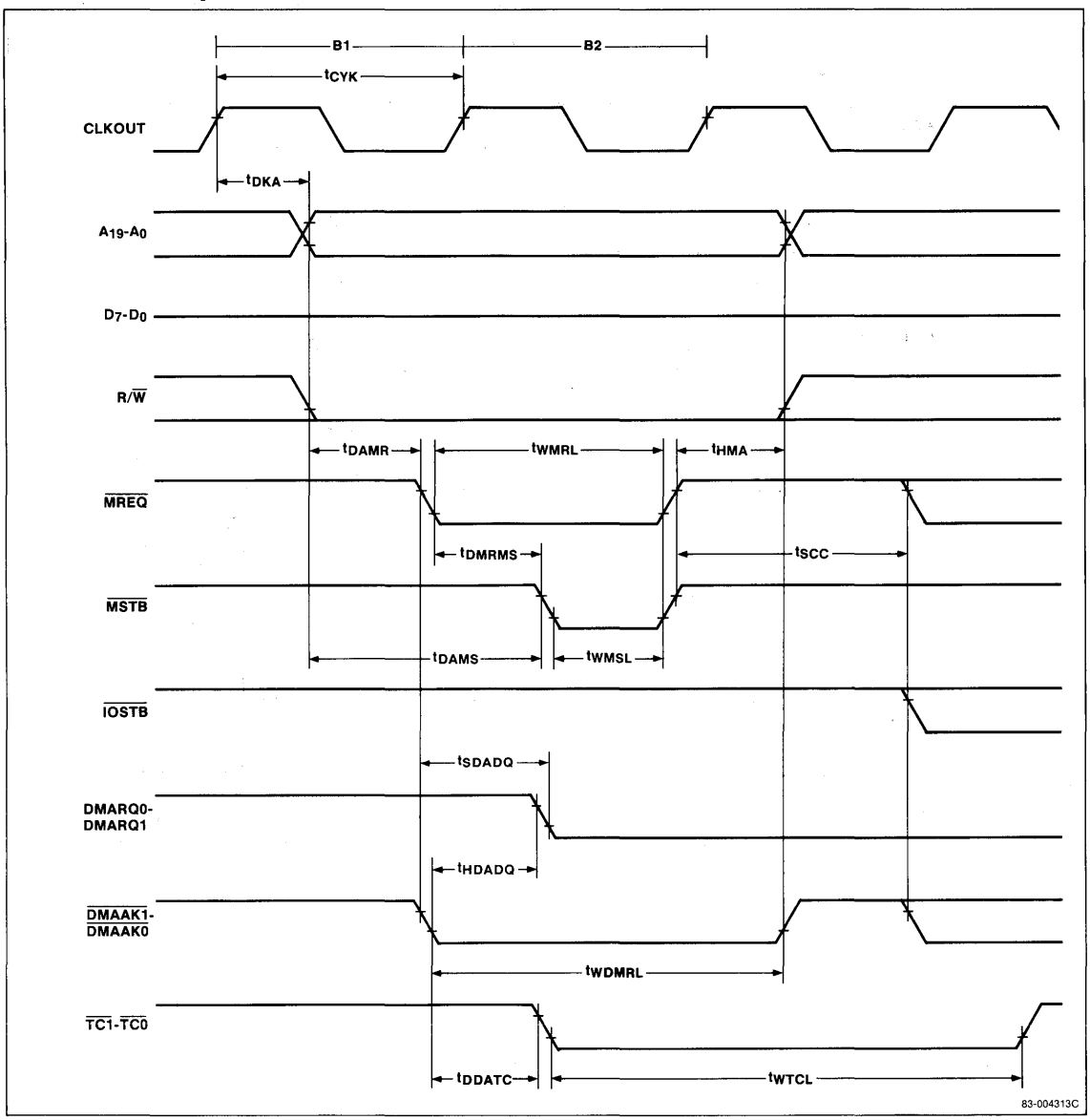

B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADR
tHISA
D7-D0
tDISD
tHISDR
R/W
MREQ
MSTB
tDAIS
tWISL
tSCC
IOSTB
REFRQ
DMAAK1-
DMAAK0
83-004311C

I/O Write

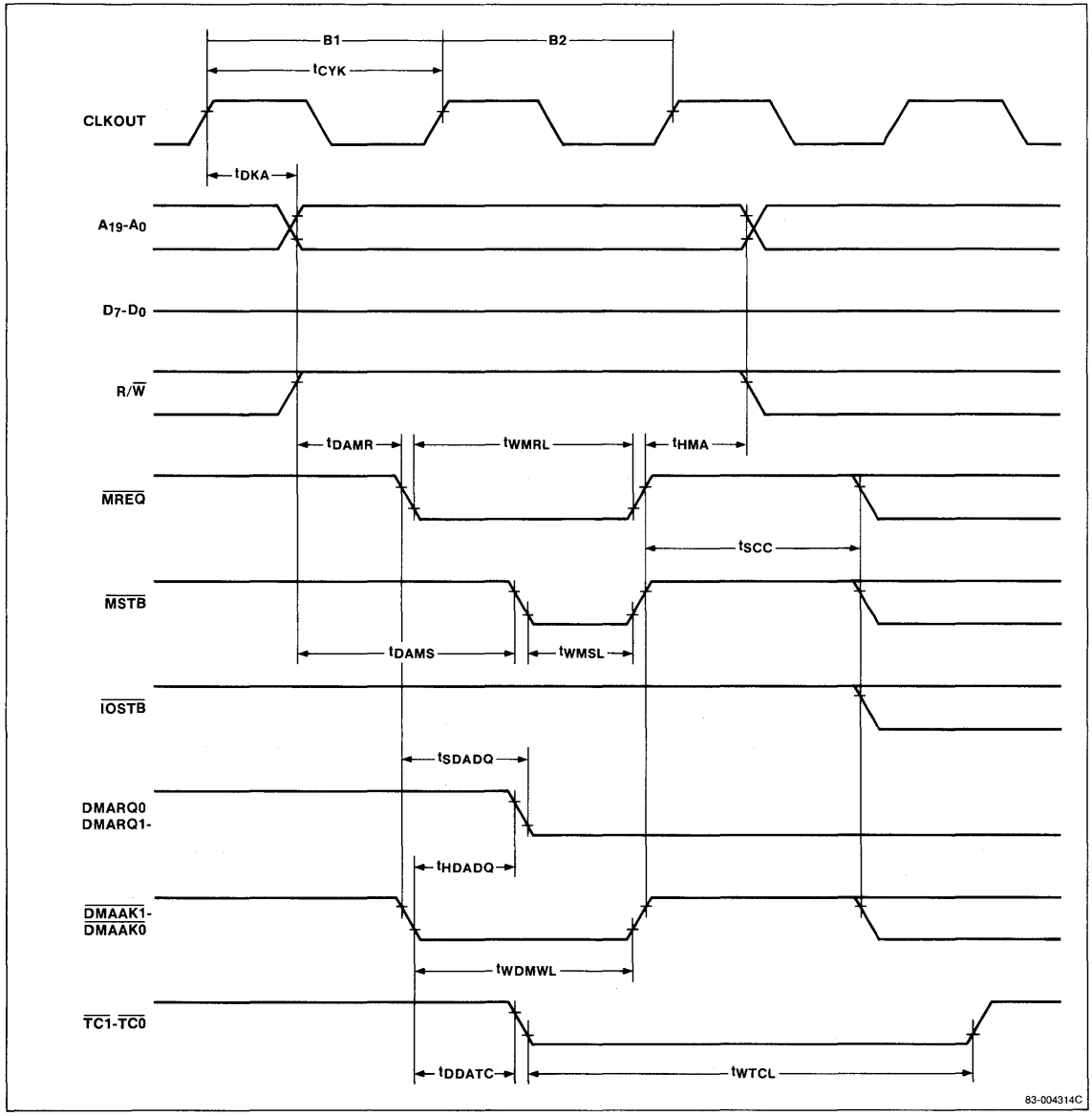

B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADW
tHISA
D7-D0
tSDIS
tHISDW
R/W
MREQ
MSTB
tDAIS
tWISL
tSCC
IOSTB
REFRQ
DMAAK1-
DMAAK0
83-004312C

4h

DMA, I/O to Memory

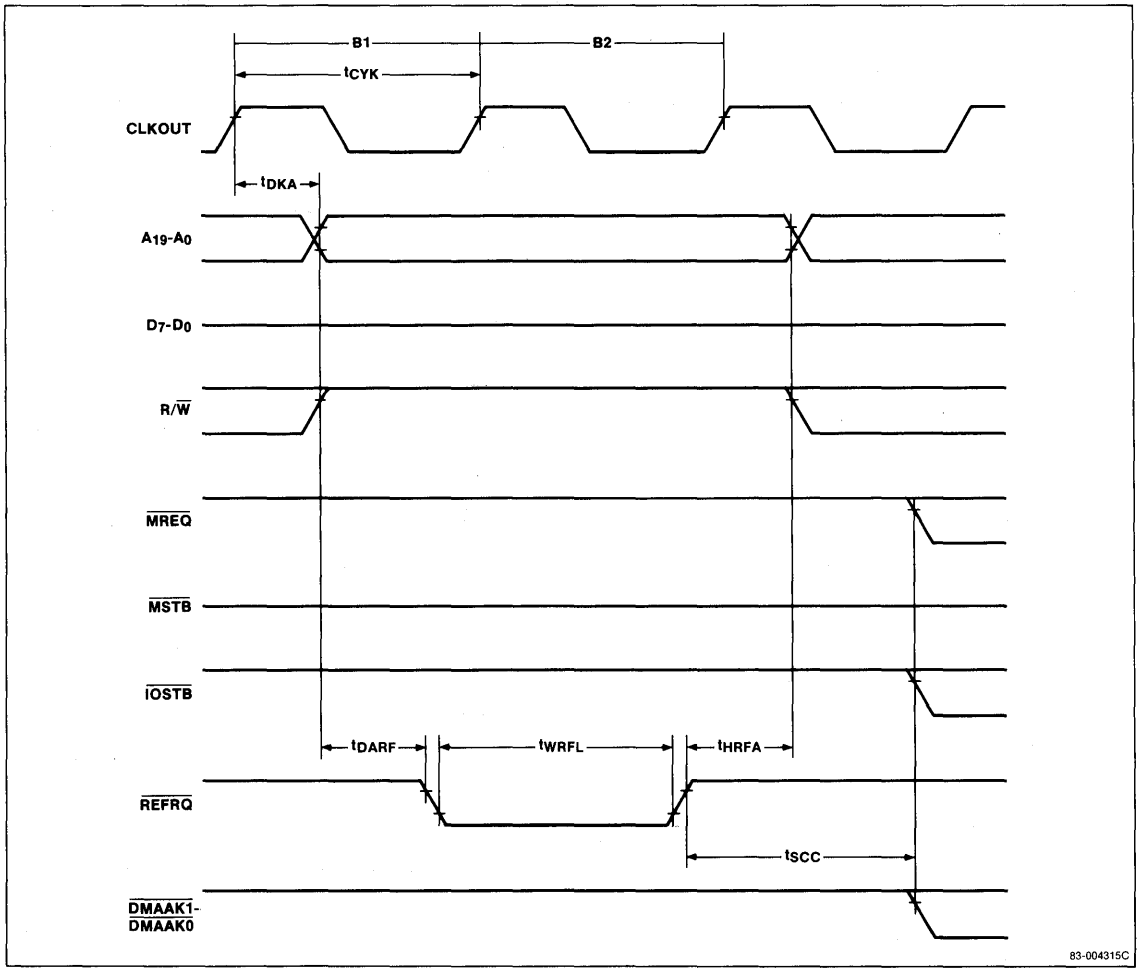


DMA, Memory to I/O

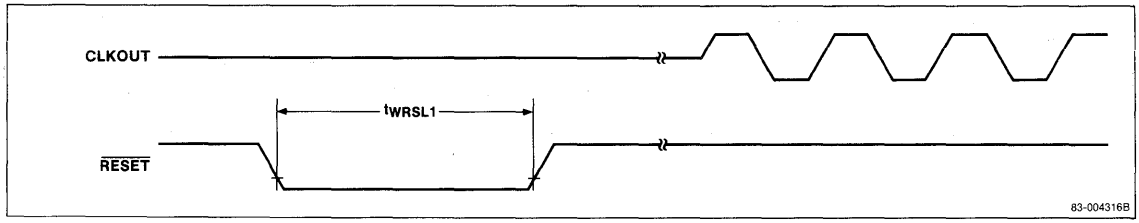


4h

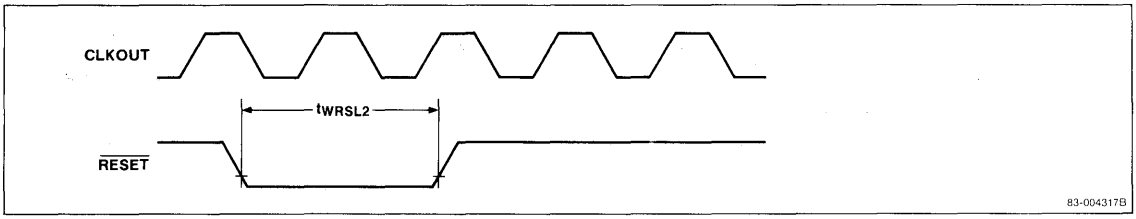
Refresh



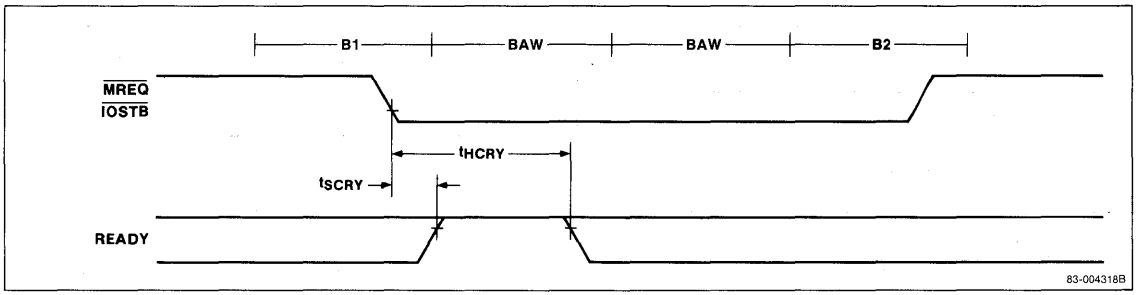
RESET 1



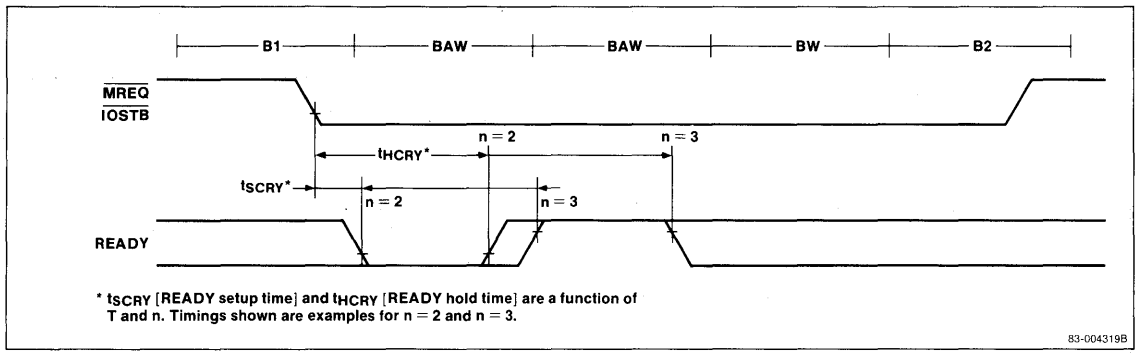
$\overline{RESET}$ 2



READY 1

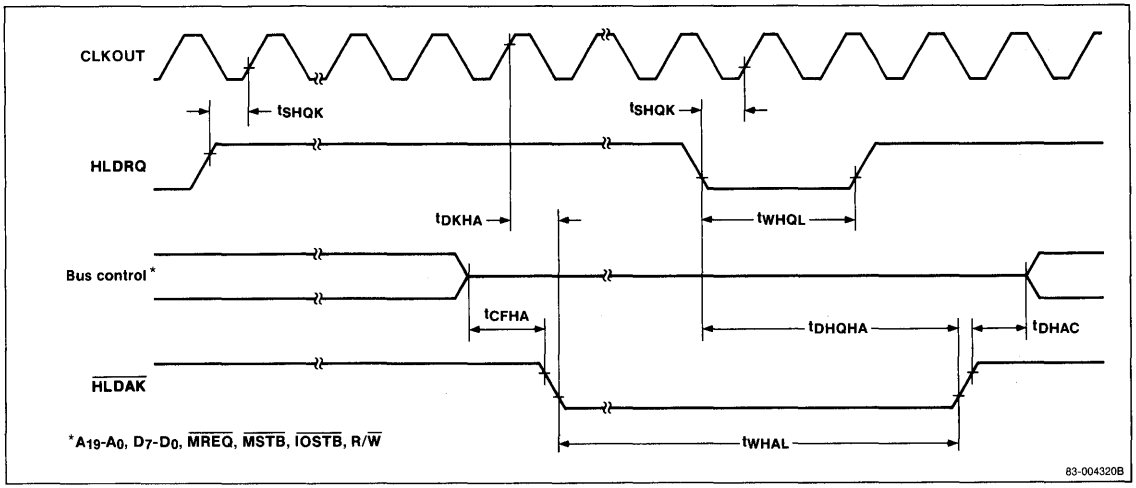


READY 2

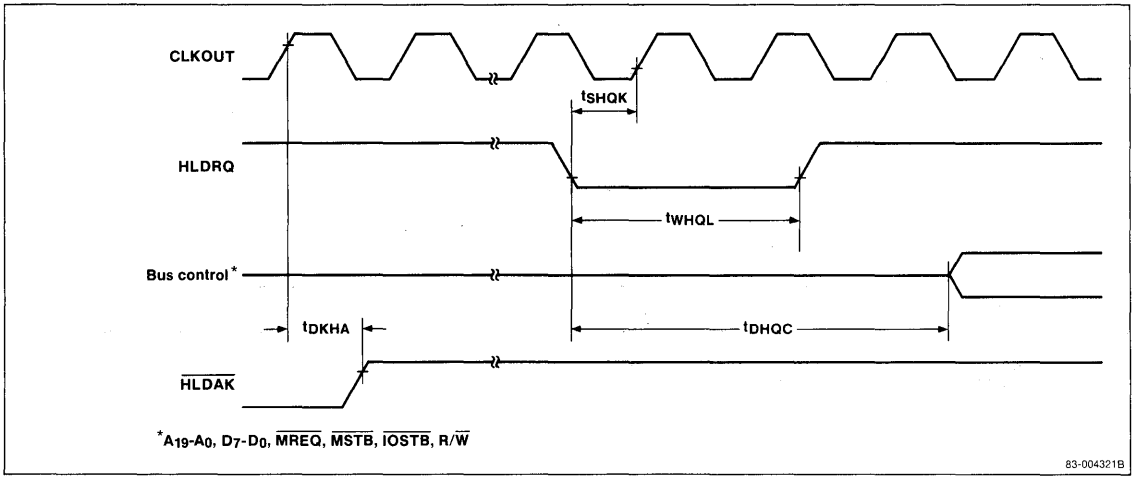


4h

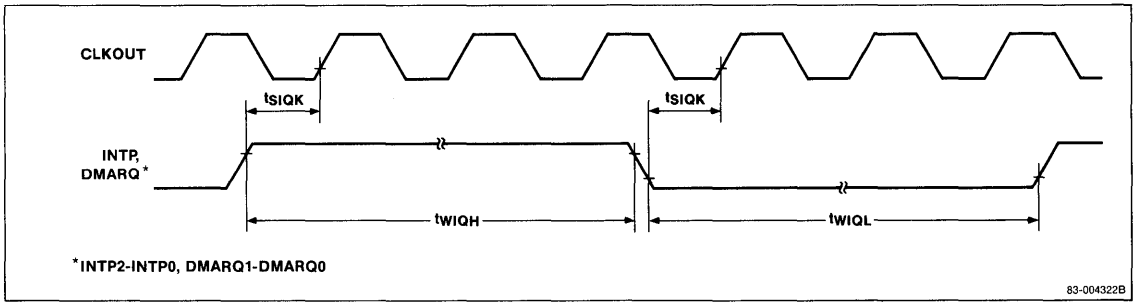
HLDRQ/$\overline{\text{HLDAK}}$ 1



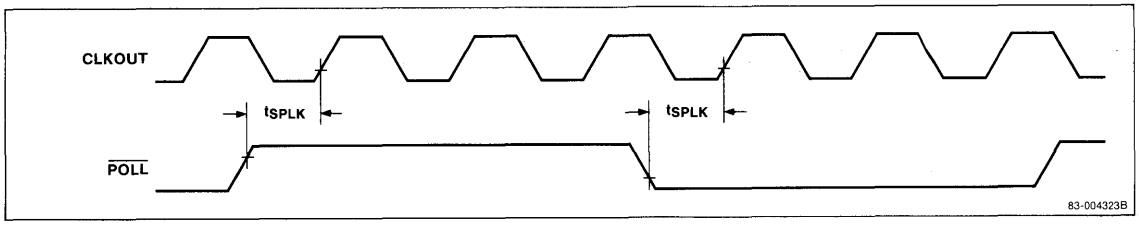
HLDRQ/$\overline{\text{HLDAK}}$ 2



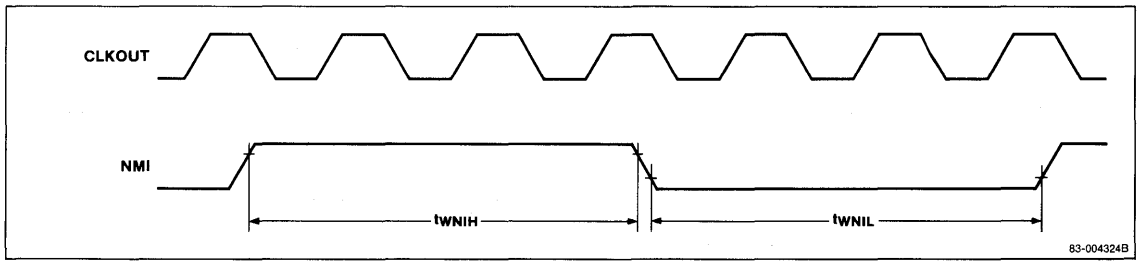
INTP, DMARQ Input



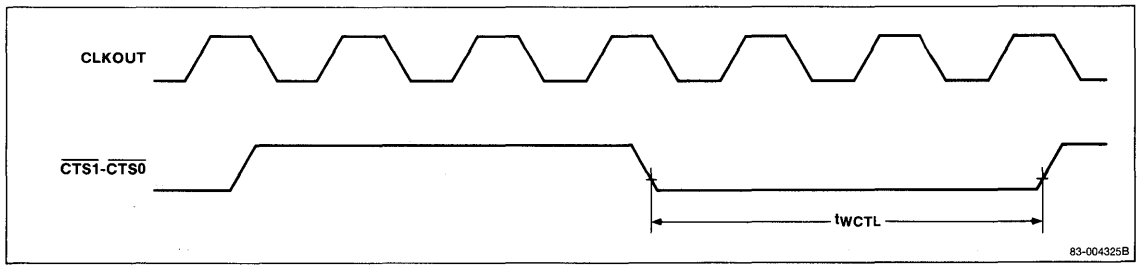
$\overline{POLL}$ Input



NMI Input

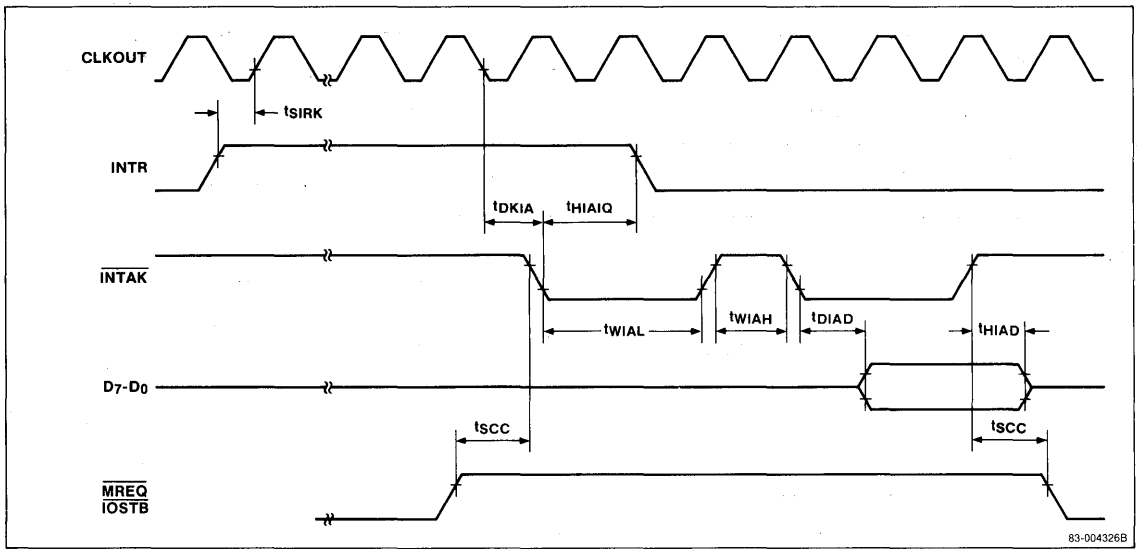


$\overline{CTS}$ Input

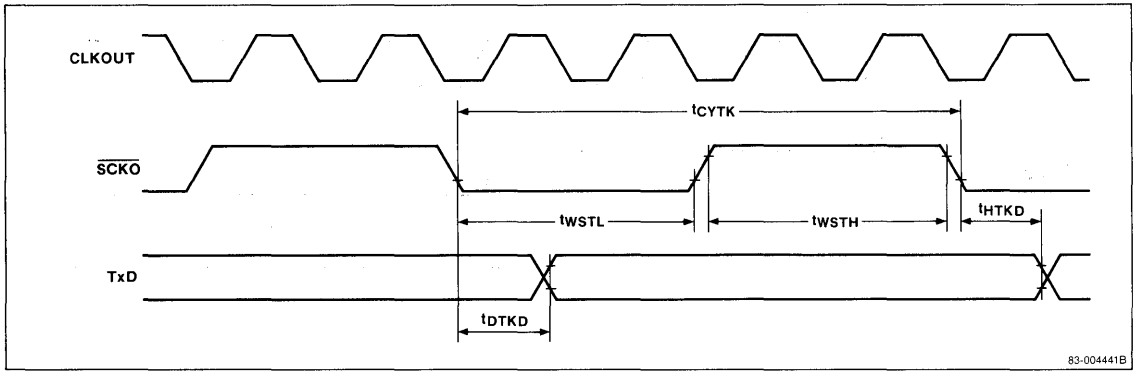


4h

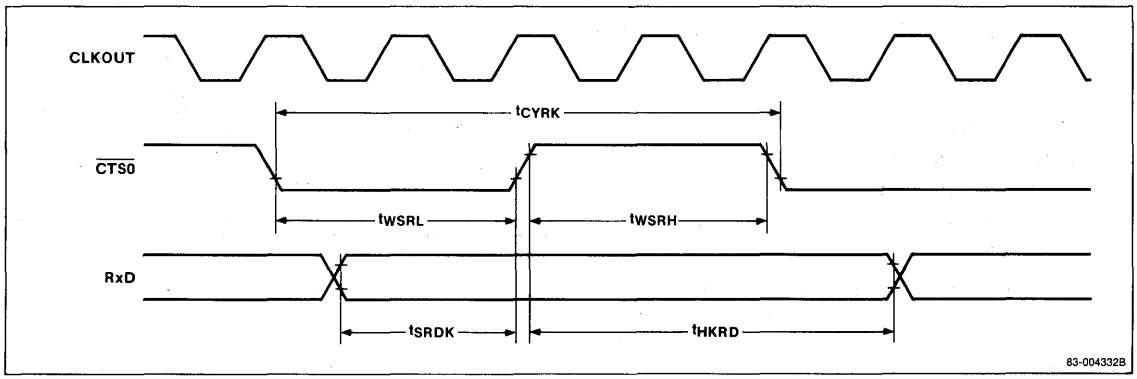
INTR/$\overline{\text{INTAK}}$



Serial Transmit



Serial Receive



ARCHITECTURAL DESCRIPTION

The μPD79011 is an upgraded version of μPD70322 (V25), NEC's original single-chip microcomputer. It has a real-time operating system built into internal ROM.

The μPD79011 is the same as the V25 in both hardware and software specifications except for the built-in ROM contents. For more information on the V25, refer to the μPD70320/70322 V25 Data Sheet

Memory Map

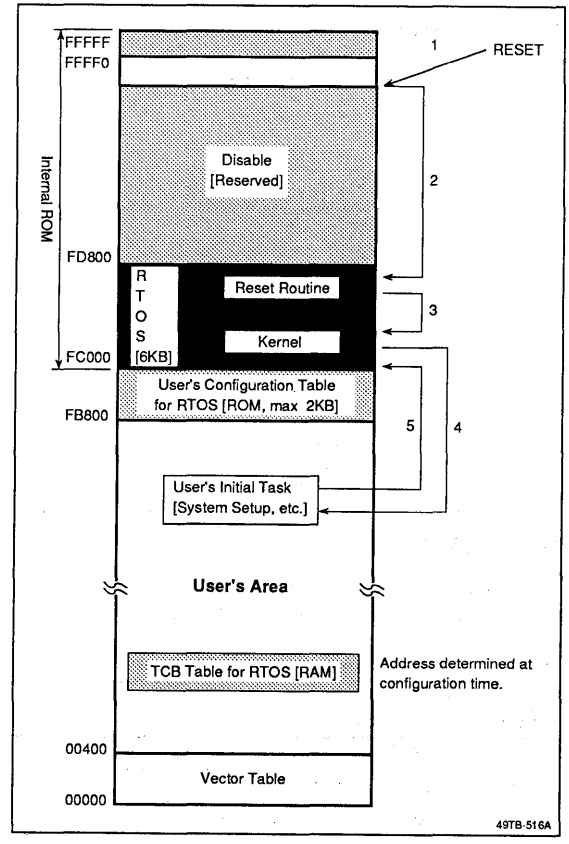
The μPD79011 can access a maximum of 1M bytes of memory via the 20-bit address bus. A 16K-byte segment of memory (FC000H to FFFFFH) is allocated to the on-chip ROM. The μPD79011 operating system is stored in this ROM area.

An external memory area of 2K bytes (FB800H to FBFFFH) contains a configuration table. When reset, the μPD79011 starts program execution at address FFFF0H, and performs the necessary initialization according to the information in this table. Then, program control is passed to each user-defined task.

A 1K-byte area (00000H to 003FFH) contains the vector tables. Thus, the total area for user tasks is from 00400H to FB7FFH.

Figure 1 is the μPD79011 memory map.

Figure 1. Memory Map



4h

Reset Operation

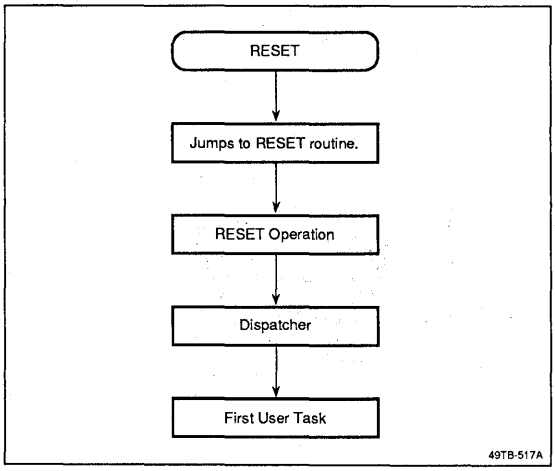
When reset, the μPD79011 begins program execution at address FFFF0H and jumps to the reset routine, which performs the following processing.

- Initializes special registers
- Initializes the interrupt vector table
- Generates the system table
- Specifies both semaphore and mailbox areas
- Generates and starts tasks

After completing the required reset processing, the μPD79011 jumps to the operating system dispatch routine, and then passes the program control to each user-defined task.

Figure 2 is a flowchart of system operation at reset time.

Figure 2. Reset Operation Flowchart



Interrupt Vectors

Up to 256 interrupt vectors (4 bytes/vector) can be stored in the vector table area. See table 1.

Table 1. Vector Table Area Assignments

Vector Number	Start Address	Use
0 to 31	00000H	Reserved for hardware as on μPD70322 (V25)
32 to 47	00080H	Available for use
48	000C0H	Operating system data table
49 to 55	000C4H	Available for use
56 to 63	000E0H	External μPD71059 (Master. Available for use)
64 to 71	00100H	External μPD71059 (Slave 0. Available for use)

Table 1. Vector Table Area Assignments (cont)

Vector Number	Start Address	Use
72 to 79	00120H	External μPD71059 (Slave 1. Available for use)
80 to 87	00140H	External μPD71059 (Slave 2. Available for use)
88 to 95	00160H	External μPD71059 (Slave 3. Available for use)
96 to 103	00180H	External μPD71059 (Slave 4. Available for use)
104 to 111	001A0H	External μPD71059 (Slave 5. Available for use)
112 to 119	001C0H	External μPD71059 (Slave 6. Available for use)
120 to 127	001E0H	External μPD71059 (Slave 7. Available for use)
128 to 255	00200H	Available for use

Note: Vectors 56 to 127 are assigned to the master and slave interrupt controllers when added to the μPD79011. Otherwise, the area is free to be used.

Configuration Table

The configuration table resides in memory from FB800H to FBFFFH. The reset routine obtains initialization information from the configuration table. Any items not initialized by the reset routine must be initialized by the user initial task.

Table 2 is an example of a configuration table. It shows the assembler sources (described by RA70116). The input values in the table are only examples.

Table 2. Configuration Table, Filing Example

CONF_TBL	Data Type	Example Value	Notes
PTR0	DW	INTERNAL_ RAM_ BASE	1
PTR1	DW	TASK_CNT	
PTR2	DW	SMA_CNT	
PTR3	DW	MBOX_CNT	
INTERNAL_ RAM_BASE	DB	FFH	2
PRC_INFO	DB	46H	
LOW_DS	DW	1000H	3
HIGH_DS	DW	2000H	
BLK_SIZE	DW	2FC0H	

Table 2. Configuration Table, Filing Example (cont)

CONF_TBL	Data Type	Example Value	Notes
PORT0	DW	1000H	4
PORT1	DW	2000H	
PORT2	DW	0FFFFH	
PORT3	DW	0FFFFH	
PORT4	DW	0FFFFH	
PORT5	DW	0FFFFH	
PORT6	DW	0FFFFH	
PORT7	DW	0FFFFH	
PORT8	DW	0FFFFH	
TASK_CNT	DB	2BH	5
MIN_TASK_NO	DB	0	
INIT_TASK	DB	0	
IDLE_SP	DW	1000H	6
IDLE_SS	DW	0F000H	
INIT_PC0	DW	0000H	7 User Task 0
INIT_PS0	DW	4000H	
INIT_ SP0	DW	2000H	
INIT_SS0	DW	0F000H	
INIT_DS0	DW	2000H	
INIT_PC1	DW	1000H	7 User Task 1
INIT_PS1	DW	4000H	
INIT_SP1	DW	3000H	
INIT_SS1	DW	0F000H	
INIT_DS1	DW	2000H	
SMA_CNT	DW	2	8
INIT_RSC0	DW	1	
	DW	10H	
MBOX_CNT	DW	10H	
RESERVE	DW	00H	9
CONF_TBL	ENDS		
	END		

Notes:

(1) Pointers
(2) System information
(3) RAM information
(4) Interrupt controller information
(5) User task information
(6) Idle task stack information
(7) User task register information
(8) Semaphore/mailbox information
(9) Reserved area

Pointers

A pointer is an offset value obtained using a segment value of 0FB08H. The following pointers are provided. The organization of the configuration table changes according to user system status.

Pointer	Size	Points to
PTR0	1 word	INTERNAL_RAM_BASE
PTR1	1 word	TASK_CNT
PTR2	1 word	SMA_CNT
PTR3	1 word	MBOX_CNT

System Information

INTERNAL_RAM_BASE: This byte is required to set the internal RAM base segment of the μPD79011. It is specified in the internal data area base register (IDB address 0FFFFFH).

If XXH is specified as the IDB value (where X is a hexadecimal number), the internal RAM base segment is assumed to be XX00H. Therefore, each register bank and the special function register (including IDB) are assigned to the 512-byte area starting at address XXE00H.

PRC_INFO. This byte sets the processor control register (PRC), which has the following functions.

- System clock divider of oscillator frequency
- Interval of time base interrupt
- Enable/disable of internal RAM

RAM Information

The configuration table provides the following RAM information.

LOW_DS/HIGH_DS: These two words specify the user free RAM area. Because it is a continuous memory area, both the upper and lower limit segment addresses (offset 0) must be used to specify this area.

The initialize routine sets the system table and each control block in this RAM area. Any remaining control blocks are queued in the system table as memory blocks (the section System Calls provides more information). The user free RAM area must be large enough to hold all control blocks.

BLK_SIZE: This word of information specifies the memory block size in units of 16 bytes. If BLK_SIZE of zero is specified, no memory blocks are generated.

Interrupt Controller

PORT0 through PORT8 (9 words) provide the information required when one or more external interrupt controllers (μPD71059) are connected to μPD79011.

PORT0 specifies the port address for the master interrupt controller. PORT1 through PORT8 specify the port addresses corresponding to the slave interrupt controllers (0 to 7).

If fewer than nine interrupt controllers are used, 0FFFFH indicates the addresses of the unused interrupt controllers.

User Task Information

TASK_CNT: This byte of information specifies the total number of user tasks (except for idle tasks). Up to 63 tasks can be specified.

MIN_TASK_NO: User task numbers are assigned sequentially starting from this number, the mimimum task number. Only tasks with numbers greater than the minimum task number are generated.

INIT_TASK: This byte of information indicates the number of the first task that the operating system must execute when the system is initialized. All other tasks are dormant when the system is initialized.

Idle Task Stack

IDLE_SP: This word of information specifies the idle task stack pointer (SP) value.

IDLE_SS: This word of information specifies the idle task stack segment (SS) value. When a stack is set, any value can be used for the address. The stack area must be a minimum of 32 bytes.

User Task Register Initialization

INIT_PC0: This word of information specifies the initial value of the program counter (PC) in relation to the minimum user task number specified for MIN_TASK_NO.

INIT_PS0: This word of information specifies the initial value of the program segment (PS) for the first user task.

INIT_SP0: This word of information specifies the initial value of the stack pointer (SP) for the first user task.

INIT_SS0: This word of information specifies the initial value of the stack segment (SS) for the first user task.

INIT_DS0: This word of information specifies the initial value of the data segment (DS) for the first user task.

The above set of register initial values is repeated for each user task.

Semaphore/Mailbox

SMA_CNT: This word of information specifies up to 256 semaphores to be used.

INIT_RSC0: This word of information supplies the initial number of resources for semaphore 0. After specification of semaphore 0, the initial number of resources of all other semaphores should be specified sequentially.

MBOX_CNT: This word of information specifies the number of mailboxes (up to 256) to be used.

Reserved Area

RESERVE is a one-word area. You must specify a value of 0 for RESERVE.

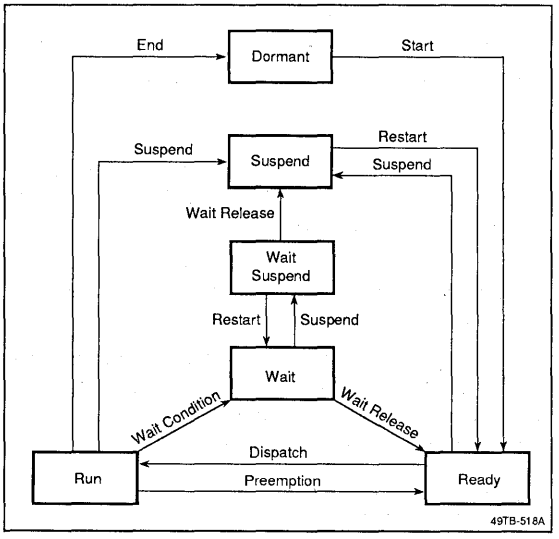
Task Status and Status Change

Table 3 shows the various task statuses. Figure 3 shows all task status changes.

Table 3. Task Status

Status	Meaning
RUN	One task, given priority to use the CPU, is currently being executed.
READY	A task is ready to execute. A READY task has a priority lower than the task currently under execution and is hence blocked by the priority handler.
WAIT	A task is waiting for an event to occur so it can go into the READY status. This status is caused by the following conditions: WAIT - a system call caused the status change and the task is either waiting for a resource with a semaphore, waiting for a message (through mail box or direct connection), or waiting for an interrupt.
SUSPEND	The system call SUS_TSK suspended execution forcibly when the task was in the RUN status. The task must wait for a system call to restart execution
WAIT SUSPEND	A task was forcibly moved into the WAIT status and has a double wait status. If the system call RSM_TSK is issued to a task in the WAIT SUSPEND status, the task is released from the SUSPEND status and goes into the WAIT status. If released from the WAIT status, the task goes into SUSPEND status.
DORMANT	When the system is initialized, only one task goes into the READY state. All other tasks go into the DORMANT status. If the system call EXT_TSK is issued to a task that is executing, this task becomes DORMANT.

Figure 3. Task Status Change



Idle Task

The μPD79011 operates an idle task when no user-set task needs to be executed. The user-specified maximum number plus 1 is used as the idle task number.

If the idle task begins execution, it executes the HALT instruction in the Interrupt Enable status, then waits for an interrupt to be issued.

FUNCTIONAL DESCRIPTION

The μPD79011 can handle up to 64 tasks numbered and assigned priorities from 0 to 63. Task numbers and priority levels correspond to each other. (For example, task 3 has a task priority of 3.) Level 0 is the highest priority; level 63 is the lowest priority.

Tasks are scheduled according to their priority levels. The μPD79011 selects and executes the READY task with the highest priority (RUN status).

Like the V25, the μPD79011 has 8 register banks (numbered 0 to 7). Task switching can be done at a high speed using these register banks. The operating system occupies bank 7. The remaining banks (0 to 6) are all assigned to tasks.

Of the 7 register banks, tasks numbered 0 to 5 are assigned to banks 0 to 5 and are resident in the banks. Because the bank-resident tasks do not require any processing to save/return the task status, task switching can be handled quickly.

The remaining tasks, numbered 6 to 63, are all assigned to bank 6. These tasks, unlike tasks resident in banks, require processing time to swap the task state to register bank 6.

Table 4 shows the register banks and tasks.

Table 4. Register Banks and Corresponding Tasks

Register Bank	Task	*Priority	Type
0	0	0	Resident
1	1	1	
2	2	2	
3	3	3	
4	4	4	
5	5	5	
6	6 to 63	6 to 63	Non-resident
7	—	—	Occupied by μPD79011 OS

No priority can be set for DMA or macroservice transfer.

Task Management

The task management function is used to terminate, start, suspend, restart tasks, and set the restart address.

If system call STA_TSK is issued to a task, the task exits the DORMANT status and goes into the READY status. If system call SUS_TSK is issued to a task, the specified task goes into the SUSPEND status. The task exits the SUSPEND status when system call RSM_TASK is issued, and its status becomes READY.

The restart address is set by issuing system call SET_ADR. The SET_ADR is always used with system call RES_INT to end the interrupt handler. (Refer to the section Interrupt Management for additional information.)

Synchronization/Communication Management

Tasks are synchronized by queuing or mutual exclusion. If tasks are queued, they are processed and executed one at a time.

Mutual exclusion is used in task processing to prohibit simultaneous access by more than one task to a shared resource (such as memory, an I/O device, etc.).

The μPD79011 uses semaphores for task synchronization and mailboxes for intertask communication.

Semaphores

The μPD79011 implements semaphores to manage resources and for queuing or mutually excluding tasks.

Both the P instruction (Obtain Resource) and the V instruction (Release Resource) manage only one resource at a time.

The P instruction can use the following system calls.

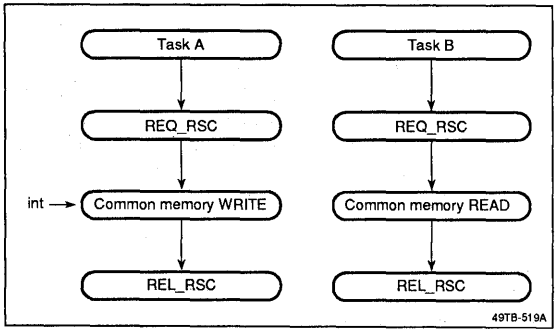
REQ_RSC: If the request to obtain resource is not accepted, the task goes into the WAIT status.

POL_RSC: If the request to obtain resource is not accepted, the system is notified that the request has been rejected.

The V instruction (system call REL_RSC) releases the occupied resource.

Figure 4 shows how to use system calls to avoid simultaneous read and write to shared memory. In figure 4, both tasks A and B share the same resource (memory). An interrupt is issued when task A is executed and control is passed between the two tasks. If the REQ_RSC request is not accepted because the resource is used by another task (task A), task B goes into the WAIT status.

Figure 4. Mutual Exclusion



Intertask Communication

Tasks communicate with each other in one of two ways, directly and nondirectly . Each task has a mailbox with a task queue for receiving messages and a message queue for sending messages. No mailbox is required for direct communication. Messages can be sent directly from one task to another.

If a task cannot receive a message for any reason (either directly or in a mailbox), one of the following system calls is issued.

RCV_MSG: Issued if a message was sent to a mailbox; the task goes into WAIT status.

RCV_DIR: Issued if a message was sent directly; the task goes into the WAIT status.

POL_MSG: Issued if a message was sent to a mailbox; notifies the system that no message can be received.

POL_DIR: Issued if a message was sent directly; notifies the system that no message can be received.

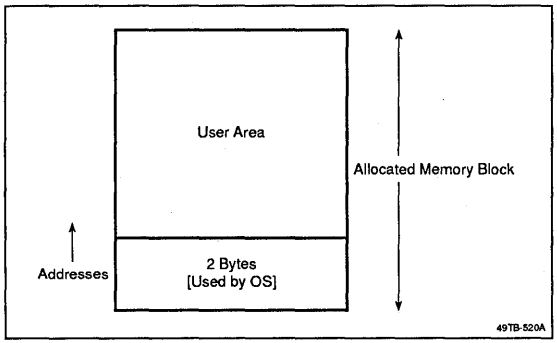
Memory Management

You can issue system calls to secure and return memory blocks dynamically on the μPD79011. The memory block size is specified at configuration time.

Task status remains the same and an the error code is returned when the GET_MEM system call is unable to secure a block of memory.

If a memory block is specified as the message area, the system uses the first two bytes of memory (figure 5). Consequently, available memory (specified in the configuration table) is reduced by 2 bytes.

Figure 5. Memory Block



Interrupt Management

For internally and externally generated interrupt requests, RTOS has the following functions to support the associated interrupt service routines.

- Interrupt handler assignment
- Interrupt handler return
- Interrupt enable/disable
- Interrupt wait status

When DEF_INT is issued, a correspondence is set between the request level (or vector type) of an external μPD71059 interrupt controller and the starting address of its service routine.

The ENA_INT and DIS_INT calls allow interrupts to be enabled or disabled.

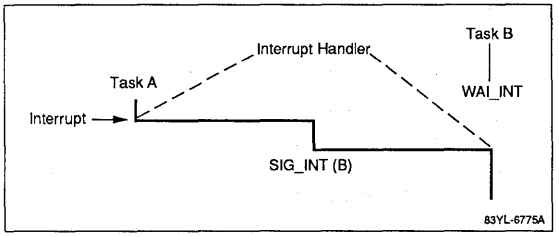
The SIG_INT and RES_INT system calls terminate the interrupt handler and pass control to the top-queued task (queued by the WAI_INT call).

Figure 6 shows how SIG_INT passes control to a task. The following events occur in the figure.

- Due to WAI_INT, task B waits for an interrupt.
- An interrupt is issued while task A is running.
- SIG_INT is issued to task B at the end of interrupt handling.

If the priority of task A is higher than that of task B, control is passed to task A when SIG_INT is executed (the interrupted task). If the priority of task B is higher, control is passed to task B.

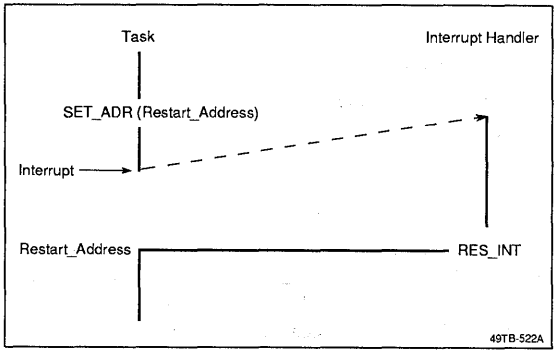
Figure 6. SIG_INT Examples



The RES_INT system call is always used with the SET_ADR system call to set the restart address. If SET_ADR has already been issued in an interrupted task handler that issues RES_INT, RES_INT passes control to the restart address specified by SET_ADR, not to the address where the interrupt was issued.

Figure 7 shows how to use the RES_INT system call to pass control to a task.

Figure 7. RES_INT Example



SYSTEM CALLS

The μPD79011 provides the following types of system calls.

- Task management
- Synchronization/communication management
- Memory management
- Time management
- Interrupt management

The system calls all have ID numbers assigned to them. Descriptions of system calls include their syntax and any error codes that may be returned to the task when the call is issued.

You can use the C language or assembly language to develop programs for the μPD79011. If using the C language, an error code is returned as a function value of the system call. If using assembly language, an error code is returned to the AW register of the μPD79011 as a return parameter.

C Language Interface

The μPD79011 supports the C language, a high-level language for developing large or small programs. To issue system calls in the C language, an assembler routine is required as an interface between the μPD79011 operating system and the C language. Refer to the Assembly Language Interface section for details on writing the interface.

Following is the syntax use for issuing calls in the C language.

err = <name> ([<parameter>]);

Argument	Description
err	Function value returned by RTOS
<name>	7-letter System Call Name
<parameter>	Input parameter

Assembly Language Interface

The μPD79011 has a C language-oriented architecture. Therefore, when issuing system calls using assembly language, the μPD79011 always sends and receives parameters via a stack. (If the system call requires no parameters, no stacking is needed.)

The syntax for issuing system calls using assembler and loading the stack for operation are shown below. If the parameter is a pointer, the offset value is stacked in the lower address area of the stack, and the segment value is stacked in the upper address area.

err = <name> (arg1, arg2, arg3);

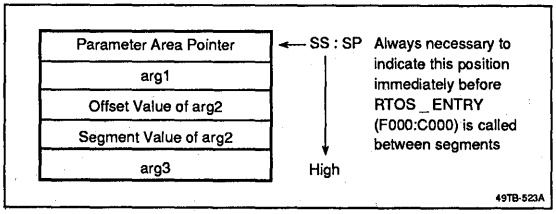
Argument	Description
<name>	7-letter System Call Name
arg1	unsigned int
arg2	int
arg3	unsigned int

The system call is issued in the following sequence.

- Parameter 3 (arg3) is stacked.
- Parameter 2 (arg2) is stacked.
- Parameter 1 (arg1) is stacked.
- A pointer to the parameter area is stacked.
- The system call number is set in the AW register.
- RTOS_ENTRY (FC000H) is called between segments.

An intersegment system call is needed even when the RTOS_ENTRY address is within the same segment.

Figure 8. Stacking Conditions



The procedures for issuing the SIG_INT and RES_INT system calls are different. They are explained later in this data sheet.

TASK MANAGEMENT SYSTEM CALLS

The following system calls are used for task management.

System Call	Description
STA_TSK	Starts task processing
EXT_TSK	Terminates task processing
SUS_TSK	Suspends task processing
RSM_TSK	Restarts task processing
SET_ADR	Sets restart address

Start Task (STA_TSK)

System Call 0. STA_TSK starts task processing during which the task goes into the READY status from the DORMANT status. It has the following syntax.

int STA_TSK (task_no)

(1) Parameter.

I/O	Name	Description
In	int task_no;	Task number (0 to 62)

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_DMT	1	Task is not DORMANT

(3) C format.

```
short task_no;
ercode = STA_TSK(task_no);
```

STA_TSK can only be issued to a task that is in the DORMANT status.

The started task processing is done in one of the following ways.

- Executed for the first time.
- After it is terminated once, it is restarted.

If a task is executed for the first time, the task processing starts from the initial address. Initial values from the configuration table are also used for the stack pointer, stack segment, and data segment values. Other register values are not defined.

If the task processing is ended once and then restarted, the task also resumes at the initial address. In this case, the stack pointer, stack segment, data segment values, and other register values assume the values they had just before the EXT_TSK system call was issued.

Exit Task (EXT_TSK)

System Call 1. EXT_TSK terminates task processing and moves the task into the DORMANT status from the RUN status. It has the following syntax.

int EXT_TSK ( )

(1) Return value.

Error Code	Number	Description
E_OK	0	Normal end

(2) C format.

ercode = EXT_TSK( );

If STA_TSK restarts a task in the DORMANT status (due to EXT_TSK), the start address returns to the initial value. Other register values retain the values they had when EXT_TSK was issued. Thus, the stack pointer, stack segment, data segment values may not match the values assumed at configuration time.

Suspend Task (SUS_TSK)

System Call 2. SUS_TSK suspends a task and puts it into the SUSPEND status. It has the following syntax.

int SUS_TSK (task_no)

(1) Parameter.

I/O	Name	Description
In	int task_no;	Task number (0 to 62)

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_DMT	1	Task is DORMANT
E_SUS	2	Task is in SUSPEND status

(3) C format.

```
short task_no;
ercode = SUS_TSK(task_no);
```

SUS_TSK cannot be issued to tasks that are in the DORMANT status or in the SUSPEND status.

If SUS_TSK is issued to a task in the WAIT status, the task goes into the WAIT SUSPEND status.

Resume Task (RSM_TSK)

System Call 3. RSM_TSK restarts a task that is in the SUSPEND status. It has the following syntax.

int RSM_TSK (task_no)

(1) Parameter.

I/O	Name	Description
In	int task_no;	Task number (0 to 62)

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_DMT	1	Task is DORMANT
E_SUS	2	Task is not in SUSPEND status

(3) C format.

```
short task_no;
ercode = RSM_TSK(task_no);
```

RSM_TSK cannot be issued to tasks that are in the DORMANT status or in the SUSPEND status.

If it is issued to a task in the WAIT SUSPEND status, the task is released from the SUSPEND status and goes into the WAIT status.

Set Restart Address (SET_ADR)

System Call 4. SET_ADR sets the restart address of a task. It has the following syntax.

int SET_ADR (restart_adr)

(1) Parameter.

I/O	Name	Description
In	int (restart_adr);	Task restart address

(2) Return value

Error Code	Number	Description
E_OK	0	Normal end

(3) C format.

```
ercode = STA_TSK(restart_adr);
pointer restart_adr;
```

SET_ADR is always used in conjunction with the RES_INT system call. If RES_INT is issued on return from the interrupt handler, control is passed to the restart address set previously by SET_ADR.

SET_ADR can be issued more than once, but the system only validates the last restart address that was issued. Setting the restart address to 0 clears current restart address.

4h

SYNCHRONIZATION/COMMUNICATION MANAGEMENT SYSTEM CALLS

The following system calls are used for synchronization/communication management:

System Call	Description
REQ_RSC	Requests resource from a semaphore
POL_RSC	Requests resource from a semaphore (no wait)
REL_RSC	Releases resource for a semaphore
RCV_MSG	Receives messages from a mailbox
POL_MSG	Receives messages from a mailbox (no wait)
SND_MSG	Sends messages to a mailbox
RCV_DIR	Receives messages sent to this task
POL_DIR	Receives messages sent to this task (no wait)
SND_DIR	Sends messages to the specified task

Request Resource (REQ_RSC)

System Call 5. REQ_RSC requests a resource from the specified semaphore. It has the following syntax.

int REQ_RSC (semaphore_no)

(1) Parameter.

I/O	Name	Description
In	int semaphore_no;	Semaphore number (0 to specified number)

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end

(3) C format.

```
ercode = REQ_RSC(semaphore_no);
short semaphore_no;
```

If REQ_RSC is issued when the resource count is 0, the task goes into the WAIT status. If the resource count is more than 1, the resource count is decremented by one.

Each semaphore has a task queue. But, if REQ_RSC causes a task to go into the WAIT status, the task is placed in the last position in the queue regardless of its priority.

Poll Resource (POL_RSC)

System Call 6. POL_RSC is used to request resources from the specified semaphore. It has the following syntax.

```
int POL_RSC (semaphore_no)
```

(1) Parameter.

I/O	Name	Description
In	int semaphore_no;	Semaphore number (0 to specified number)

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_RSC	6	Resource count is 0

(3) C format.

```
ercode = POL_RSC(semaphore_no);
short semaphore_no;
```

POL_RSC is used to determine whether any resources are left in the specified semaphore. Unlike the REQ_RSC, POL_RSC never causes a task to go into the WAIT status. Instead, it returns the E_RSC error code when the resource count is 0. If the resource count is more than 1, the count is decremented by 1.

Release Resource (REL_RSC)

System Call 7. REL_RSC releases resource for the specified semaphore. It has the following syntax.

```
int REL_RSC (semaphore_no)
```

(1) Parameter.

I/O	Name	Description
In	int semaphore_no;	Semaphore number (0 to specified number)

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end

(3) C format.

```
ercode = REL_RSC(semaphore_no);
short semaphore_no;
```

When REL_RSC is issued, the semaphore resource count is increased by 1. If WAIT tasks exist, the earliest-wait task is selected and released from the WAIT status.

The initial value of the semaphore resource count is set when the system is started. No error occurs even when the resource count exceeds the initial value as a result of issuing REL_RSC. If the resource count exceeds 65,535, the resource count is cleared to 0 automatically and no error is generated.

Receive Message (RCV_MSG)

System Call 8. RCV_MSG receives messages from mailboxes. It has the following syntax.

```
int RCV_MSG (mailbox_no)
```

(1) Parameter.

I/O	Name	Description
In	int mailbox_no;	Mailbox number (0 to specified number)

(2) C format.

```
seg = RCV_MSG(mailbox_no);
short mailbox_no;
```

If RCV_MSG is issued when messages are present in mailboxes, the earliest message is selected and the segment value of the message area is returned as the function value.

If there is no message, the task goes into the WAIT status and it is placed in the last position in the mailbox queue.

Poll Message (POL_MSG)

System Call 9. POL_MSG receives messages from mailboxes. It has the following syntax.

```
int POL_MSG (mailbox_no)
```

(1) Parameter.

I/O	Name	Description
In	int mailbox_no;	Mailbox number (0 to specified number)

(2) Return value. If there are any messages in the specified mailbox, the message area segment value is returned. If there is no message, the following error code is returned.

Error Code	Number	Description
E_MSG	7	No message found

(3) C format.

```
seg = POL_MSG(mailbox_no);
short mailbox_no;
```

If POL_MSG is issued when messages are present in mailboxes, the earliest message is selected and the segment value of the message area is returned as the function value.

If no message is found, unlike the RCV_MSG system call, the task never goes into the WAIT status. Instead, the E_MSG error code is returned.

Send Message (SND_MSG)

System Call 10. SND_MSG sends messages to mailboxes. It has the following syntax.

int SND_MSG (mailbox_no, msg_seg)

(1) Parameter.

I/O	Name	Description
In	int mailbox_no;	Mailbox number (0 to specified number)
In	int msg_seg;	Send message area segment

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end

(3) C format.

```
ercode = SND_MSG(mailbox_no, msg_seg);
short mailbox_no;
short msg_seg;
```

If SND_MSG is issued when a task is waiting to be processed, the task is released from the WAIT status, and the send message area segment value is returned.

If no tasks are in the WAIT status, the message is queued in the mailbox. Like tasks, messages are queued using the first-in, first-out (FIFO) method.

Receive Direct Message (RCV_DIR)

System Call 11. RCV_DIR receives messages sent directly to a task. It has the following syntax.

int RCV_DIR ( )

(1) C format.

ercode = RCV_DIR( );

If RCV_DIR is issued when there is no message, the task goes into the WAIT status. If a message is present, the message area segment value is returned.

Poll Direct Message (POL_DIR)

System Call 12. POL_DIR receives messages sent by a task to itself. It has the following syntax.

int POL_DIR ( )

(1) Return value. If POL_DIR is issued when a message is present, the message area segment value is returned. If no message is present, the following error code is returned and the task does not enter WAIT status.

Error Code	Number	Description
E_MSG	7	No message is present

(2) C format.

ercode = POL_DIR(n);

Send Direct Message (SND_DIR)

System Call 13. SND_DIR specifies a task and sends a message to the specified task. It has the following syntax.

int SND_DIR (task_no, msg_seg)

(1) Parameter.

I/O	Name	Description
In	int task_no;	Task number (0 to 62)
In	int msg_seg;	Send message area segment

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end

(3) C format.

```
ercode = SND_DIR(task_no, msg_seg);
short task_no;
short msg_seg;
```

If SND_DIR is issued when the specified task is waiting for a message directly, the task is released from the WAIT status. The message area segment value is returned to the task.

If the specified task is not waiting for any message directly, the message is placed in the task message queue using the FIFO method.

MEMORY MANAGEMENT SYSTEM CALLS

The following system calls are used for memory management.

System	Call Description
GET_MEM	Gets a memory block
REL_MEM	Releases the memory block

Get Memory (GET_MEM)

System Call 14. GET_MEM allocates a memory block. It has the following syntax.

int GET_MEM ( )

(1) Return value. If a memory block is available when GET_MEM is issued, the memory block segment value is returned. If no memory block is present, the following error code is returned.

Error Code	Number	Description
E_BLK	3	No memory block found

(2) C format.

ercode = GET_MEM( );

GET_MEM can use the memory block as a message area for intertask communications. The memory block size is specified when the system is started, and the value is fixed.

If the error code is returned, the task never goes into the WAIT status.

Release Memory (REL_MEM)

System Call 15. REL_MEM releases the specified memory block. It has the following syntax.

int REL_MEM (mem_blk)

(1) Parameter.

I/O	Name	Description
In	int mem_blk;	Segment value of the released memory block

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end

(3) C format.

ercode = REL_MEM(mem_blk);
short mem_blk;

REL_MEM cannot release memory blocks containing messages in a mailbox or task queue. The memory block can only be released after a message is received.

TIME MANAGEMENT SYSTEM CALLS

The following system calls are used for time management.

System Call	Description
GET_TIM	Reads the system time
SET_TIM	Sets the system time

Get Time (GET_TIM)

System Call 16. GET_TIM reads the system time. It has the following syntax.

int GET TIM (time_ptr)

(1) Parameter.

I/O	Name	Description
In	struct t_time * time_ptr;	Pointer to location of system time

(2) Time structure.

```
struct t_time{
 int l_time;
 int m_time;
 int h_time;};
```

(3) Return value.

Error Code	Number	Description
E_OK	0	Normal end

(4) C format.

ercode = GET_TIM(time_ptr);
pointer time_ptr;

The system time is 3-word data. The lower order word is stored in the lowest order address; the intermediate data is in the intermediate address; and the upper order data is in the highest address.

The minimum resolution of the system time is determined by the value set in the time base counter in the μPD79011. However, since interrupts to the μPD79011 are inhibited during system call processing, choose the minimum resolution of the system time with system call overhead time in mind.

Set Time (SET_TIM)

System Call 17. SET_TIM sets the system time. It has the following syntax.

int SET_TIM (time_ptr)

(1) Parameter.

I/O	Name	Description
In	struct t_time * time_ptr;	Time pointer

(2) Time structure.

```
struct t_time{
 int l_time;
 int m_time;
 int h_time;};
```

(3) Return value.

Error Code	Number	Description
E_OK	0	Normal end

(4) C format.

```
pointer time_ptr;
ercode = SET_TIM(time_ptr);
```

The μPD79011 uses the on-chip timer base counter output as the system real-time clock source.. The on-chip timer therefore starts its counting operation when the system is started.The interval from the SET_TIM call to the next real-time clock interrupt is an error term associated with the initial call to SET_TIM, and all subsequent calls produce additional pseudorandom error times. The real-time clock interval is set at configuration time.

INTERRUPT MANAGEMENT SYSTEM CALLS

The following system calls are used for interrupt management:

System Call	Description
DEF_INT	Sets the start address of the interrupt handler
SIG_INT	Starts a task waiting for an interrupt and terminates the interrupt handler operation
WAI_INT	Waits for an interrupt
CAN_INT	Releases a task waiting for an interrupt from WAIT status
DIS_INT	Disables interrupts by device number
ENA_INT	Enables interrupts by device number
RES_INT	Terminates interrupt handler operation and calls the restart address

Define Interrupt Handler (DEF_INT)

System Call 18. DEF_INT sets the start address of the interrupt handler. It has the following syntax.

int DEF_INT (device_no, start_adr)

(1) Parameter.

I/O	Name	Description
In	int device_no;	Device number (interrupt level or vector type)
In	int (start_adr) ( );	Pointer to interrupt handler start address

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_DVN	4	Device number error
E_SYS	5	System error

(3) C format.

```
ercode = DEF_INT(device_no, start_adr);
short device_no;
pointer start_adr;
```

If DEF_INT is issued, correspondence between interrupt request level of external μPD71059 interrupt controller (or interrupt request vector type) and start address of the interrupt handler is established. When an interrupt request vector type is specified, the interrupt request control register can also be set at the same time.

If 0 is specified for the interrupt handler start address, the existing start address is cleared. If the start address of the interrupt handler is cleared after an interrupt request level of the interrupt controller is specified, the mask bit (IMK) equivalent to the specified interrupt request level is set and the interrupt is masked. Then the existing start address is cleared.

If the start address of the interrupt handler is cleared after an interrupt request vector type is specified, the existing start address is cleared and the interrupt request control register is set. At this time, the interrupt mask can be set at the same time by explicitly setting bit 6 of the interrupt request control register.

If the start address of the interrupt handler is not 0, the address is set with no other changes. The IMK (mask bit) is never altered.

If the start address of the interrupt handler is set after the vector type of interrupt request is specified, the interrupt request control register is also set. Therefore, interrupt mask operation can be specified using bit 6 of the interrupt request control register.

External Interrupt Controller Definition

The interrupt request level of the external interrupt controller can be specified by setting 0 in bit 7. It is specified as follows.

Bit(s)	Description
0-2	Slave level interrupt request level
3	If 0, master/slave configuration; if 1, master only
4-6	Master interrupt request level
7	Fixed to 0
8-15	Upper-order byte is fixed to 0

The low-order byte is used to specify the interrupt level. Bits 0 to 2 specify the slave interrupt request level when in master-slave configuration; bit 3, whether to use any slave device; bits 4 to 6, the master interrupt request level.

The interrupt request level of the interrupt controller and each interrupt request vector type are in one-to-one corrrespondence The interrupt request is divided into 72 levels, and they correspond to interrupt request vector types 56 to 127.

For example, if, the device consists of only the master, 0 is specified for the master interrupt request level; this interrupt request vector type becomes 56. Slave interrupt request level 7 must be connected to master interrupt request level 7 when in master-slave configuration and becomes vector type 127.

The interrupt request vector type can be specified by setting 1 in bit 7. It is specified as follows.

Bit(s)	Description
0-6	Vector type
7	Fixed to 1
8-15	Interrupt request control register value

The μPD79011 operating system uses the on-chip time base counter as the system timer. As a result, other tasks cannot specify vector type 31 (equivalent to the time base counter) when the system timer function is used.

Signal Interrupt (SIG_INT)

System Call 19. SIG_INT activates a task waiting for an interrupt and terminates the currently executing interrupt handler. It has the following syntax.

```
void SIG_INT (task_no)
```

(1) Parameter.

I/O	Name	Description
In	int task_no;	Target task number (0 to 62)

(2) C format.

```
ercode = SIG_INT(task_no);
short task_no;
```

SIG_INT can be issued only from inside an interrupt handler.

If SIG_INT is issued, the interrupt handler operation ends and control is passed to the target task. Therefore, when SIG_INT is used, control is never passed to the address following SIG_INT.

If an error occurs, no error code is returned and the specified task is not started. In this case, control is returned immediately to the point where the interrupt was issued.

SIG_INT is not used to control multiprocessing of external or internal interrupt requests. Nesting management related to the interrupt handler and execution of the EOI (End Of Interrupt) and FINT (Finish Interrupt) instructions must be done in each interrupt handler.

The procedures used to issue SIG_INT (and system call RES_INT) differ from those to issue other system calls. When using assembly language, SIG_INT is issued as follows.

Procedure	Description
PUSH task_no	The target task number is set in stack
BR SIG_INT_ ENTRY	Far jump to absolute address 0FC00EH

Wait for Interrupt (WAI_INT)

System Call 20. WAI_INT moves a task into the WAIT status. It has the following syntax.

```
int WAI_INT ( )
```

(1) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_INT	8	Release from interrupt wait status

(2) C format.

```
ercode = WAI_INT;
```

When issuing this system call, the current task goes into the interrupt wait status. If the SIG_INT system call is issued to a waiting task (which was invoked by WAI_INT), the specified task is released from the WAIT status.

A task can release another task's WAIT (for interrupt) status by means of the CAN_INT system call. Otherwise an interrupt handler will release the WAIT status after an interrupt is presented.

If SIG_INT is used to release a task from the WAIT status, the error code E_OK is returned. If CAN_INT is used to release the WAIT status, the error code E_INT is returned.

Cancel Interrupt (CAN_INT)

System Call 21. CAN_INT releases the specified task from the WAIT status. It has the following syntax.

int CAN_INT (task_no)

(1) Parameter.

I/O	Name	Description
In	int task_no;	Task number (0 to 62)

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_INT	8	Task is not waiting for interrupt

(3) C format.

```
ercode = CAN_INT(task_no);
short task_no;
```

If CAN_INT is issued to a task that is waiting for an interrupt (due to system call WAI_INT), the specified task exits the WAIT status. If CAN_INT is issued when the specified task is not waiting for any interrupt, the E_INT error code is returned.

Disable Interrupt (DIS_INT)

System Call 22. DIS_INT disables interrupts in units of device number (interrupt request level or interrupt request vector type). It has the following syntax.

int DIS_INT (device_no)

(1) Parameter.

I/O	Name	Description
In	int device_no;	Device number

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_DVN	4	Device number error

(3) C format.

```
ercode = DIS_INT(device_no);
short device_no;
```

DIS_INT can be issued from either a task or an interrupt handler.

To specify the interrupt request level of the interrupt controller, set the corresponding IMR (mask bit) of the external 71059. To specify the interrupt request vector type, set bit 6 of the interrupt request control register.

Enable Interrupt (ENA_INT)

System Call 23. ENA_INT enables interrupts in units of device number (interrupt request level or interrupt request vector type). It has the following syntax.

int ENA_INT (device_no)

(1) Parameter.

I/O	Name	Description
In	int device_no;	Device number

(2) Return value.

Error Code	Number	Description
E_OK	0	Normal end
E_DVN	4	Device number error

(3) C format.

```
ercode = ENA_INT(device_no);
short device_no;
```

ENA_INT can be issued from either a task or an interrupt handler.

To specify the interrupt request level of the interrupt controller, reset the corresponding IMR (mask bit) of external 71059. To specify the interrupt request vector type, reset bit 6 of the interrupt request register.

Reset Interrupt (RES_INT)

System Call 24. RES_INT terminates interrupt handler operation and passes control to the restart address. It has the following syntax.

void RES_INT ( )

(1) C format.

```
ercode = RES_INT( );
```

RES_INT is always used in conjunction with system call SET_ADR.

If SET_ADR has been already issued in a task that was interrupted by a handler that issues RES_INT, control is passed to the specified restart address. If SET_ADR has not been issued to that task, control is returned to the point where the interrupt was issued.

RES_INT cannot be used to control multiple interrupt processing, neither for internal nor for external 71059

sources. Management of interrupt handler nesting and the execution of EOI (End Of Interrupt) and FINT (Finish Interrupt) instructions must be done in each interrupt handler.

The procedure for issuing RES_INT (and system call SIG_INT) differs from the procedure for issuing other system calls. Use the following syntax to issue RES_INT using assembly language.

```
BR RES_INT_ ENTRY;  Far jump to absolute
                    address FC020H
```

NEC
NEC Electronics Inc.

μPD79021 16-Bit Microcomputer: Single-Chip, CMOS, With Built-In RTOS

Description

The μPD79021 is an upgraded μPD70332 (V35™) single-chip microcomputer with a built-in real-time operating system (RTOS).

The μPD79021 provides high-speed multitask processing particularly suited for real-time event processing and as a kernel of an embedded control system for process control and data processing applications.

The RTOS kernel provides extensive system calls for task synchronization, control, and communication as well as interrupt and time management.

The μPD79021 instruction set is the same as the V35 instruction set. The μPD79021 hardware is also identical to the standard V35, but uses 6K of the internal ROM for RTOS system code. Refer to the V35 Data Sheet for hardware-related details and the μPD79011 Data Sheet for RTOS system call descriptions.

Features

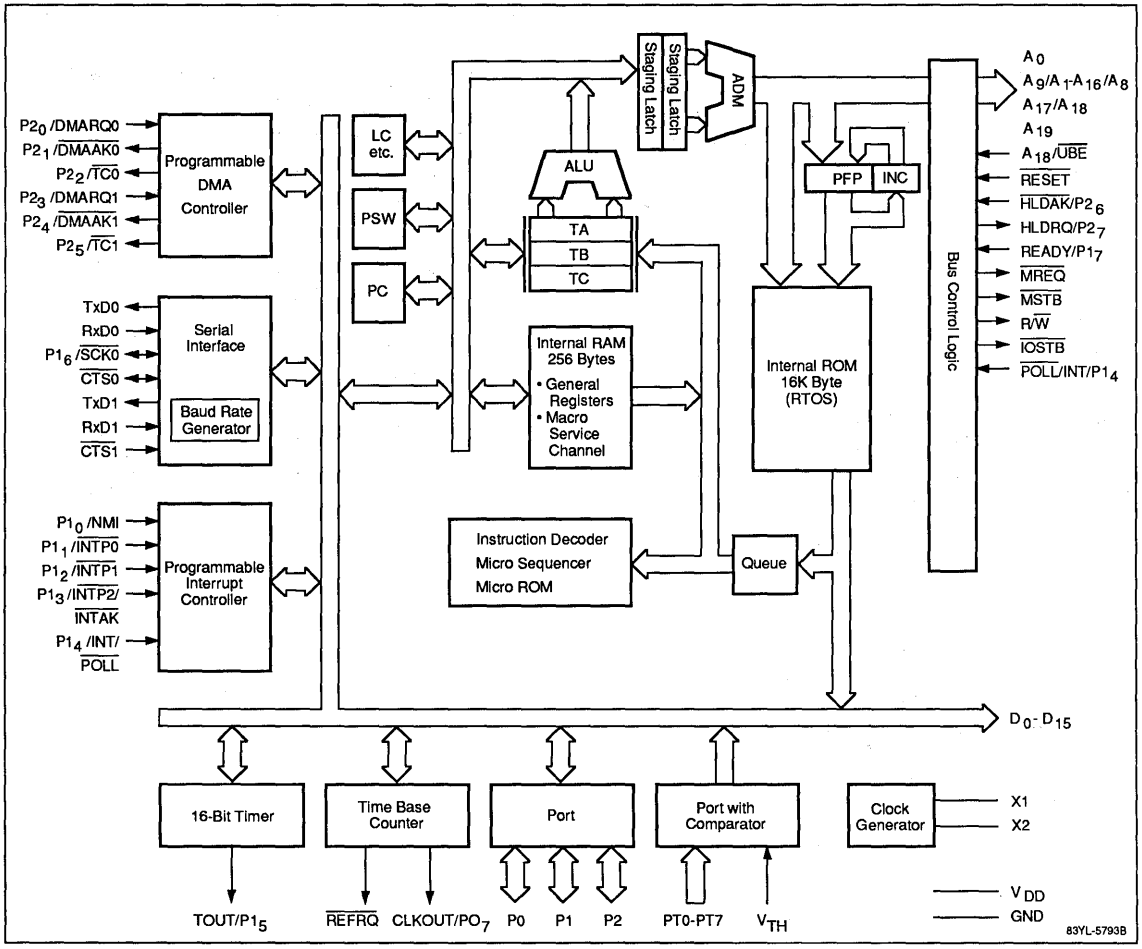
- Real-time multitask processing
- Supports five types of system calls
  - Task management
  - Communication management
  - Memory management
  - Time management
  - Interrupt management
- High-speed response to events
  - System call processing shortens time to 41 μs (minimum) when operated at 8 MHz
  - High-speed task switching using V35 register banks
- Flexibility to perform status changes by event driven task scheduling function
- System clock: 8 MHz maximum
- V35 hardware compatibility
- CMOS technology
- Development tools
  - V35 software can be used without modification
  - Relocatable assembler (RA70320)
  - C compiler (CC70116)
  - Concurrent CP/M®, MS-DOS®, VMS™, and UNIX™ base

Ordering Information

Part Number	Clock	Package
μPD79021L-8	8 MHz	84-pin PLCC
GJ-8	8 MHz	94-pin plastic QFP

V35 is a trademark of NEC Corporation.
CP/M is a registered trademark of Digital Research, Inc.
MS-DOS is a registered trademark of Microsoft Corporation.
VMS is a trademark of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories.

μPD79021 Block Diagram



NEC

Section 5
Peripherals for CPUs

Description

The μPD71011 is a clock pulse generator/driver for the V20®/V30® microprocessors and their peripherals using NEC's high-speed CMOS technology.

Features

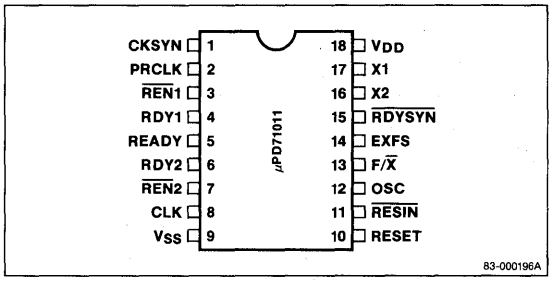
- CMOS technology
- Clock pulse generator/driver for μPD70108/70116 or other CMOS or NMOS CPUs and their peripherals
- 50% duty cycle
- Frequency source can be crystal or external clock input
- Reset signal with Schmitt-trigger circuit for CPU or peripherals
- Bus ready signal with two-bus system synchronization
- Clock synchronization with other μPD71011s
- Single +5-volt ±10% power supply
- Industrial temperature range: −40 to +85°C

Ordering Information

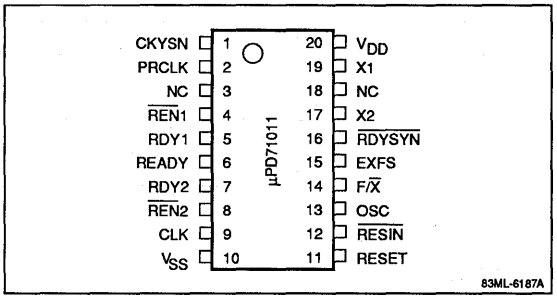
Part Number	Maximum Clockout Frequency	Package
μPD71011C-8	8 MHz	18-pin plastic DIP
C-10	10 MHz	
G-8	8 MHz	20-pin plastic SOP

Pin Configurations

18-Pin Plastic DIP



20-Pin Plastic SOP



5a

V20 and V30 are registered trademarks of NEC Corporation

Pin Identification

Symbol	Function
CKSYN	Clock synchronization input
PRCLK	Peripheral clock output
$\overline{REN1}$	Bus ready enable input 1
RDY1	Bus ready input 1
READY	Ready output
RDY2	Bus ready input 2
$\overline{REN2}$	Bus ready enable input 2
CLK	Processor clock output
V_{SS}	Ground potential
RESET	Reset output
$\overline{RESIN}$	Reset input
OSC	Oscillator output
F/$\overline{X}$	External frequency source/crystal select input
EXFS	External frequency source input
$\overline{RDYSYN}$	Ready synchronization select input
X2	Crystal input
X1	Crystal input
V_{DD}	+5-volt power supply
NC	No connection

PIN FUNCTIONS

X1, X2 (Crystal)

When F/$\overline{X}$ is low, a crystal connected to X1 and X2 will be the frequency source for a CPU and its peripherals. The crystal frequency should be two times the frequency of CLK.

EXFS (External Frequency Source)

EXFS input is the external frequency input in the external TTL-frequency source mode (F/$\overline{X}$ high). A square TTL-level clock signal two times the frequency of CLK's output should be used for the source.

F/$\overline{X}$ (Frequency/Crystal Select)

F/$\overline{X}$ input selects whether an external TTL-type input or an external crystal input is the frequency source of the CLK output. When F/$\overline{X}$ is low, CLK is generated from the crystal connected to X1 and X2. When F/$\overline{X}$ is high, CLK is generated from an external TTL-level frequency input on the EXFS pin. At the same time, the internal oscillator circuit will go into stop mode and the OSC output will be high.

CLK (Processor Clock)

The CLK output supplies the CPU and its local bus peripherals' clocks. CLK is a 50-percent duty cycle clock of one-half the frequency of the external frequency source. The CLK output is +0.4 V higher than the other outputs.

PRCLK (Peripheral Clock)

The PRCLK output supplies a 50-percent duty cycle clock at one-half the frequency of CLK to drive peripheral devices.

OSC (Oscillator)

OSC outputs a signal at the same frequency as the crystal input. When EXFS is selected, the OSC output is powered down, and its output will be a high.

CKSYN (Clock Synchronization)

CKSYN synchronizes one μPD71011 to other μPD71011s. A high level at CKSYN resets the internal counter, and a low level enables it to count.

$\overline{RESIN}$ (Reset)

This Schmitt-trigger input generates the RESET output. It is used as a power-on reset.

RESET (Reset)

This output is a reset signal for the CPU. Reset timing is provided by the $\overline{RESIN}$ input to a Schmitt-trigger input gate and a flip-flop which will synchronize the reset timing to the falling edge of CLK. Power-on reset can be provided by a simple RC circuit on the $\overline{RESIN}$ input.

RDY1, RDY2 (Bus Ready)

A peripheral device sends RDY1 or RDY2 to signal that the data on the system bus has been received or is ready to be sent. $\overline{REN1}$ and $\overline{REN2}$ enable the RDY1 or RDY2 signals.

$\overline{REN1}$, $\overline{REN2}$ (Bus Ready Enable)

$\overline{REN1}$ and $\overline{REN2}$ qualify their respective RDY inputs.

$\overline{\text{RDYSYN}}$ (Ready Sychronization Select)

$\overline{\text{RDYSYN}}$ selects the mode of READY signal synchronization. A low-level signal makes the synchronization a two-step process. This is used when RDY1 and RDY2 inputs are not synchronized to CLK. A high-level signal makes synchronization a one-step process. This is used when RDY1 and RDY2 are synchronized to CLK. See block diagram.

READY (Ready)

The READY signal to the processor is synchronized by the RDY inputs to the processor CLK. READY is cleared after the RDY signal goes low and the guaranteed hold time of the processor has been met.

Crystal

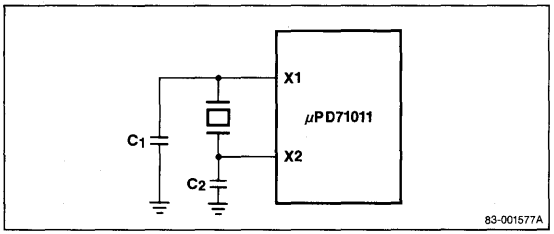
The oscillator circuit of the μPD71011 works with a parallel-resonant, fundamental mode, "AT cut" crystal connected to pins X1 and X2.

Figure 1 shows the recommended circuit configuration. Capacitors C1 and C2 are required for frequency stability. The values of C1 and C2 (C1 = C2) can be calculated from the load capacitance (C_L) specified by the crystal manufacturer.

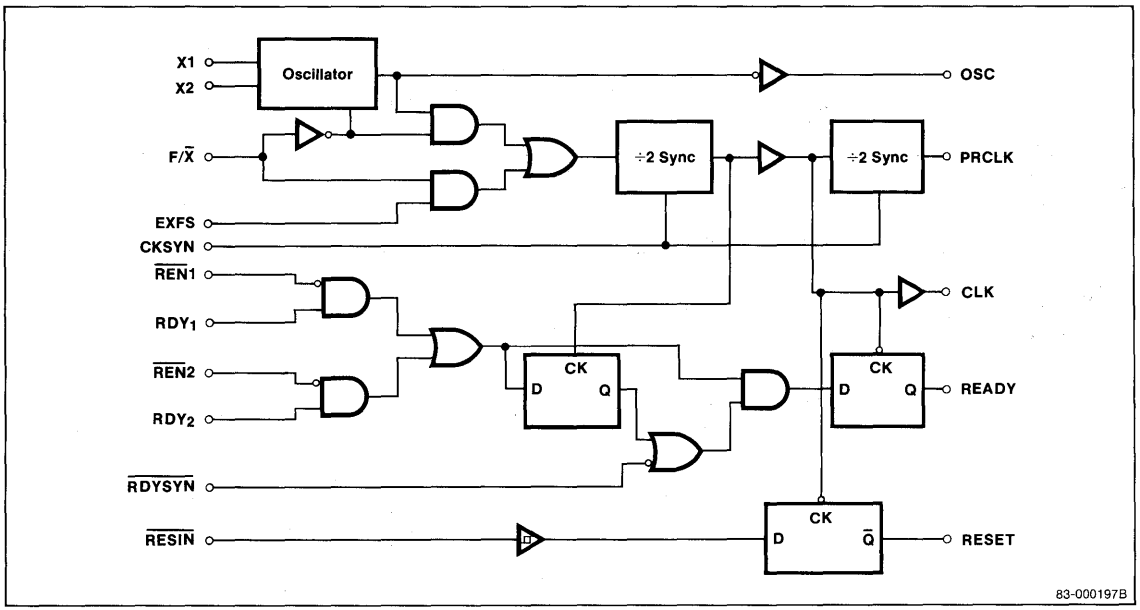
$$C_L = \frac{C1 \times C2}{C1 + C2} + C_S$$

Where CS is any stray capacitance in parallel with the crystal, such as the μPD71011 input capacitance C_{IN}.

Figure 1. Crystal Configuration Circuit



μPD71011 Block Diagram



5a

Absolute Maximum Ratings

T_A = 25°C; V_{SS} = 0 V

Power supply voltage, V_{DD}	− 0.5 to + 7.0 V
Input voltage, V_I	− 1.0 V to V_{DD} + 1.0 V
Output voltage, V_O	− 0.5 V to V_{DD} + 0.5 V
Operating temperature, T_{OPT}	− 40 to +85°C
Storage temperature, T_{STG}	− 65 to +150°C
Power dissipation, P_D (DIP)	500 mW
Power dissipation, P_D (SO package)	200 mW

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage. The device should be operated within the limits specified under DC and AC Characteristics.

Capacitance

T_A = 25°C; V_{DD} = +5 V

Parameter	Symbol	Min	Max	Unit	Conditions
Input capacitance	C_{IN}		12	pF	f_C = 1 MHz

DC Characteristics

T_A = −40 to +85°C; V_{DD} = 5 V ±10%

Parameter	Symbol	Min	Max	Unit	Conditions
Input voltage, high	V_{IH}	2.2		V	
		2.6		V	$\overline{\text{RESIN}}$ input
Input voltage, low	V_{IL}		0.8	V	
Output voltage, high	V_{OH}	V_{DD} −0.4		V	CLK output, I_{OH} = −4 mA
		V_{DD} −0.8		V	I_{OH} = −4 mA
Output voltage, low	V_{OL}		0.45	V	I_{OL} = 4 mA
Input leakage current	I_{IN}	−1.0	1.0	μA	
		−400	1.0	μA	$\overline{\text{RDYSYN}}$ input
$\overline{\text{RESIN}}$ input hysteresis		0.20		V	
Power supply current (static)	I_{DD}		200	μA	
Power supply current (dynamic)	I_{DDdyn}		30	mA	f_{in} = 20 MHz

AC Characteristics

f_{OSC} = 10 MHz; T_A −40 to +85°C; V_{DD} = +5 V ±10%
f_{OSC} = 16 MHz; T_A −10 to +70°C; V_{DD} = +5 V ±5%
f_{OSC} = 20 MHz; T_A −10 to +70°C; V_{DD} = +5 V ±5%

		μPD71011		μPD71011-10			
Parameter	Symbol	Min	Max	Min	Max	Unit	Conditions
Clock Timing							
EXFS input cycle time	t_{CYFS}	50		50		ns	
EXFS pulse width, high	t_{PWFSH}	20		20		ns	2.2 V measurement point
EXFS pulse width, low	t_{PWFSL}	20		20		ns	0.8 V measurement point
OSC cycle time	f_{OSC}	8	20	8	20	MHz	from EXFS
CKSYN pulse width	t_{PWCT}	$2t_{CYFS}$		$2t_{CYFS}$		ns	
CKSYN setup time	t_{HFSCTK}	20		20		ns	
CKSYN hold time	t_{SCTFS}	20		20		ns	
CLK cycle time	t_{CYCK}	125		100		ns	
CLK pulse width, high	t_{PWCKH}	80		41		ns	3.0 V, f_{OSC} = 10 MHz, f_{OSC} = 20 MHz
		50				ns	3.0 V, f_{OSC} = 16 MHz
CLK pulse width, low	t_{PWCKL}	90		49		ns	1.5 V, f_{OSC} = 10 MHz, f_{OSC} = 20 MHz
		60				ns	1.5 V, f_{OSC} = 16 MHz
CLK rise time	t_{LHCK}		10		5	ns	1.5 → 3.0 V, f_{OSC} = 10 MHz, f_{OSC} = 20 MHz
			8			ns	1.5 → 3.0 V, f_{OSC} = 16 MHz
CLK fall time	t_{HLCK}		10		5	ns	3.0 → 1.5 V, f_{OSC} = 10 MHz, f_{OSC} = 20 MHz
			7			ns	3.0 → 1.5 V, f_{OSC} = 16 MHz
OSC to CLK ↑ delay	t_{DCK}	2	30	2	30	ns	CLK ↑
OSC to CLK ↓ delay	t_{DCK}	−6	28	−6	28	ns	CLK ↓
PRCLK cycle time	t_{CYPRK}	250		200		ns	

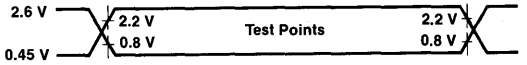
AC Characteristics (cont)

Parameter	Symbol	μPD71011 Min	μPD71011 Max	μPD71011-10 Min	μPD71011-10 Max	Unit	Conditions
Clock Timing (cont)							
PRCLK pulse width, high	t_{PWPRKH}	t_{CYCK} − 20		t_{CYCK} − 20		ns	
PRCLK pulse width, low	t_{PWPRKL}	t_{CYCK} −20		t_{CYCK} − 20		ns	
PRCLK ↑ delay from CLK ↓	t_{DPRKH}		22		22	ns	
PRCLK ↓ delay from CLK ↓	t_{DPRKL}		22		22	ns	
Reset Timing							
$\overline{RESIN}$ setup to CLK ↓	$t_{S\overline{RI}CK}$	65		65		ns	
$\overline{RESIN}$ hold from CLK ↓	$t_{HCK\overline{RI}}$	20		20		ns	
RESET delay from CLK ↓	t_{DCKRS}		40		20	ns	
Ready Timing ($\overline{RDYSYN}$ = 'H')							
$\overline{REN}$1, 2 setup to RDY1, 2	$t_{S\overline{RE}RY}$	15		15		ns	
$\overline{REN}$1, 2 hold from CLK ↓	$t_{HCK\overline{RE}}$	0		0		ns	
RDY1, 2 setup to CLK ↓	t_{SRYCK}	35		35		ns	$\overline{RDYSYN}$ high
RDY1, 2 hold from CLK ↓	t_{HCKRY}	0		0		ns	
$\overline{RDYSYN}$ setup to CLK ↓	$t_{S\overline{RYS}CK}$	50		50		ns	
$\overline{RDYSYN}$ hold from	$t_{HCK\overline{RYS}}$	0		0		ns	
READY output delay from CLK ↓	t_{DCKRDY}		8		8	ns	READY ↑
			8		8	ns	READY ↓
Ready Timing ($\overline{RDYSYN}$ = 'L')							
$\overline{REN}$1, 2 setup to RDY1, 2	$t_{S\overline{RE}RY}$	15		15		ns	
$\overline{REN}$1, 2 hold from CLK ↓	$t_{HCK\overline{RE}}$	0		0		ns	
RDY1, 2 setup to CLK	t_{SRYCK}	35		35		ns	$\overline{RDYSYN}$ low
RDY1, 2 hold from CLK ↓	t_{HCKRY}	0		0		ns	
RDYSYN setup to CLK ↓	$t_{S\overline{RYS}CK}$	50		50		ns	
RDYSYN hold from CLK ↓	$t_{HCK\overline{RYS}}$	0		0		ns	
READY output delay from CLK ↓	t_{DCKRDY}		8		8	ns	READY ↑
			8		8	ns	READY ↓
Output Pin Timing							
Rise time	t_{LH}		20		20	ns	0.8 → 2.0 V
Fall time	t_{HL}		12		12	ns	2.0 → 0.8 V

5a

Timing Waveforms

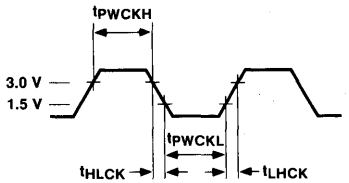
AC Test Input (except $\overline{RESIN}$)



RESIN: Input high level 3.00 V Measurement point 2.6 V
Input low level 0.45 V Measurement point 0.8 V

83-004046A

AC Test Output (CLK)

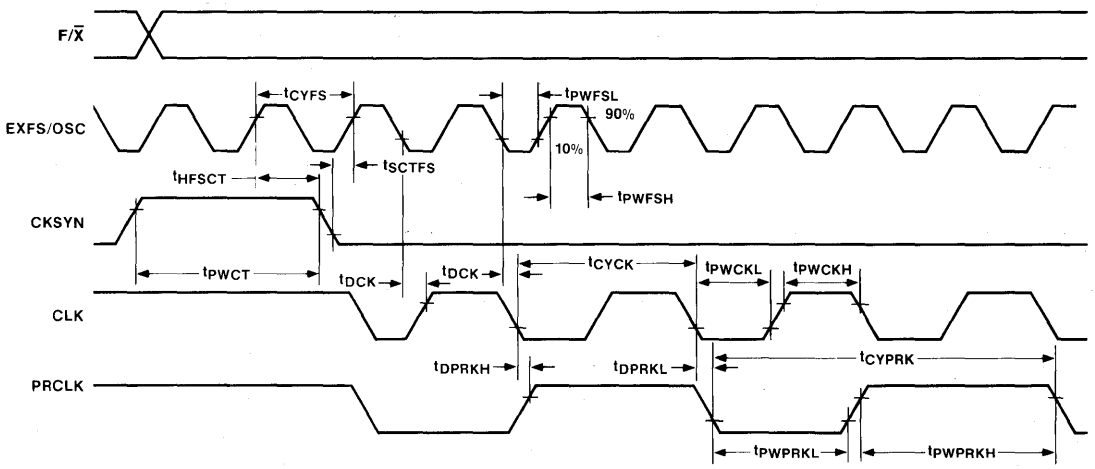


83-004048A

AC Test Output (except CLK)

83-004047A

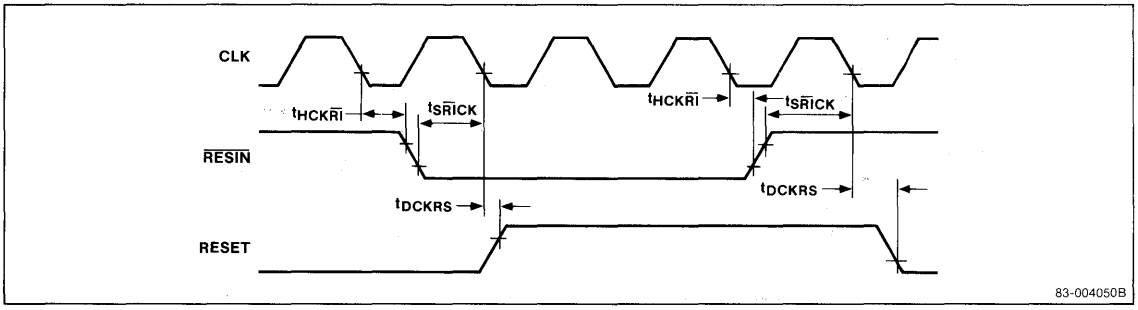
Clock Output



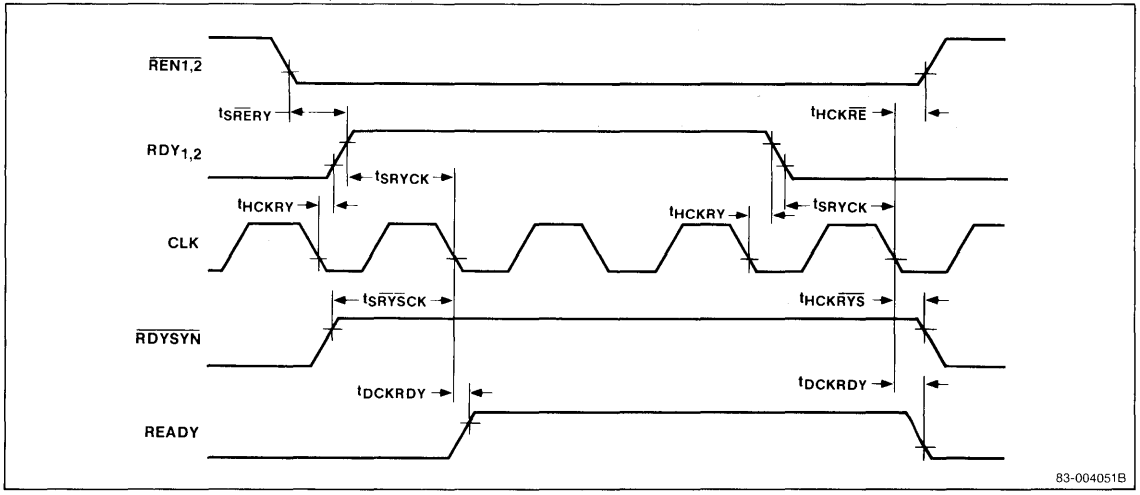
83-004049B

Timing Waveforms (cont)

RESET Pin



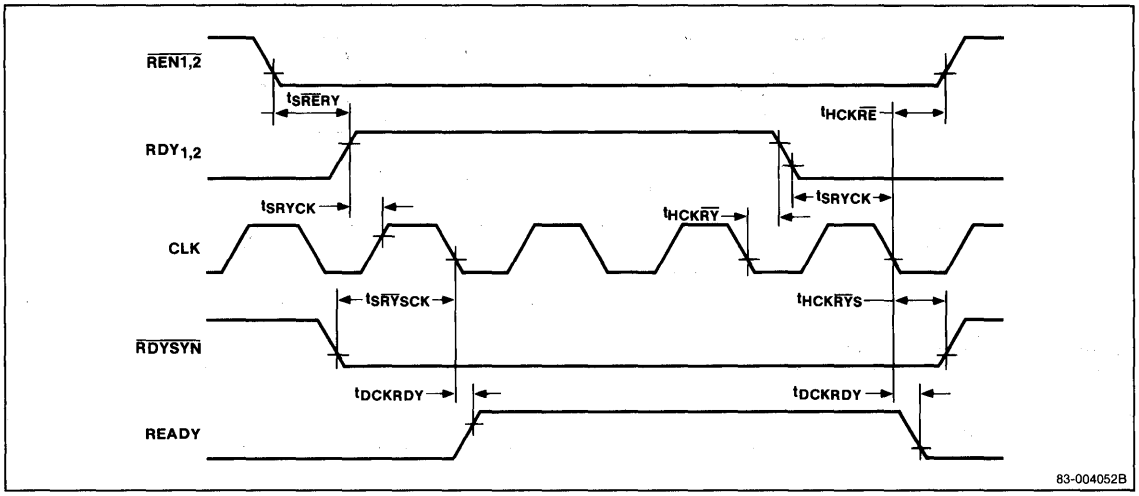
READY Pin ($\overline{RDYSYN}$ = 'H')



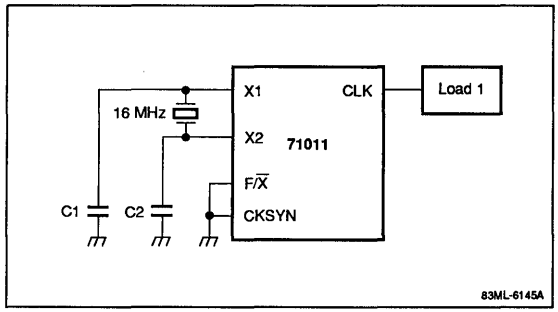
5a

Timing Waveforms (cont)

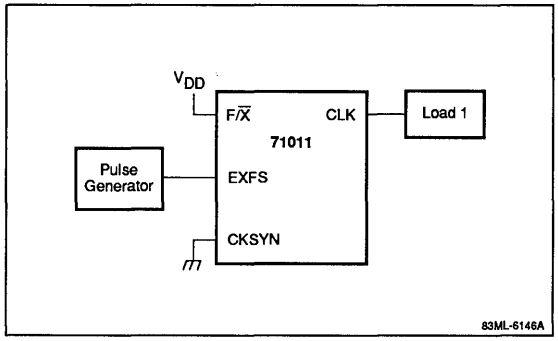
READY Pin ($\overline{RDYSYN}$ = 'L')



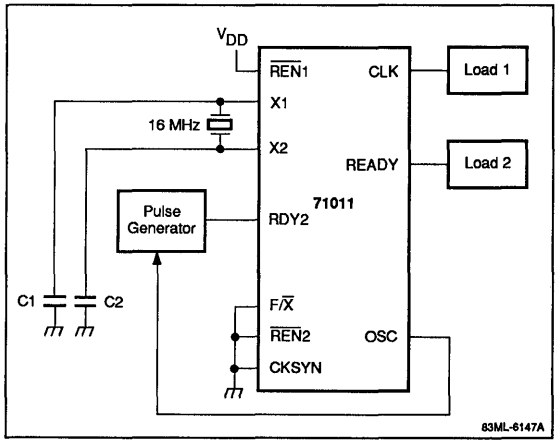
Test Circuit for CLK High or Low Time (in Crystal Oscillation Mode)



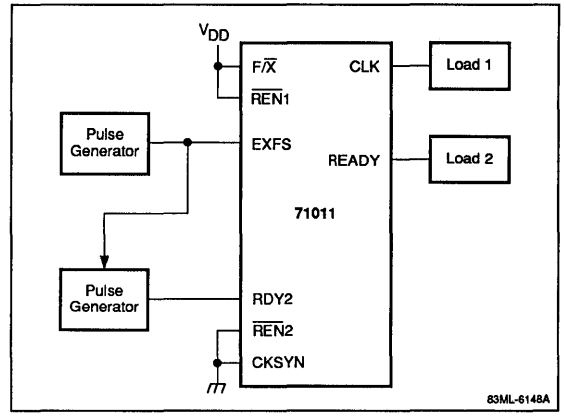
Test Circuit for CLK High or Low Time (in EXFS Oscillation Mode)



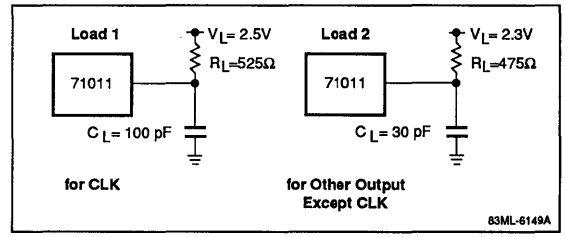
Test Circuit for CLK to READY (in Crystal Oscillation Mode)



Test Circuit for CLK to READY (in EXFS Oscillation Mode)



Loading Circuits



5a

μPD71037
Direct Memory Access (DMA) Controller

Description

The μPD71037 is a direct memory access (DMA) controller that provides high-speed data transfers between peripheral devices and memory for microprocessor systems. It is faster and draws less power than its predecessors. The unit has four DMA channels, each with a 64K-byte address area and a transfer byte count function. The channels enable I/O-to-memory and memory-to-memory data transfer.

The μPD71037 is a versatile DMA controller that can be used for the following applications.

- Office automation equipment (personal computers, small business computers, EWS, etc.)
- Communications
- Instrumentation
- Control

Features

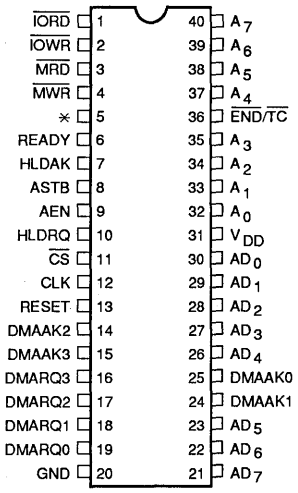
- Processing speed of 10 MHz (twice that of the μPD8237A-5)
- Four independent DMA channels
- Self-initialization for each channel
- Memory-to-memory data transfer
- Block-level memory initialization
- High-speed data transfer
  — 3.2 Mb/s, 10-MHz normal transfer
  — 5.0 Mb/s, 10-MHz compression transfer
- DMA channel count directly expandable in expansion mode
- END input for the end of transfer
- Software DMA request
- CMOS
- Low power consumption

Ordering Information

Part Number	Clock (MHz)	Package
μPD71037CZ-10	10	40-pin plastic DIP (600 mil)
GB-10	10	44-pin plastic QFP
LM-10	10	44-pin PLCC

Pin Configurations

40-Pin Plastic DIP

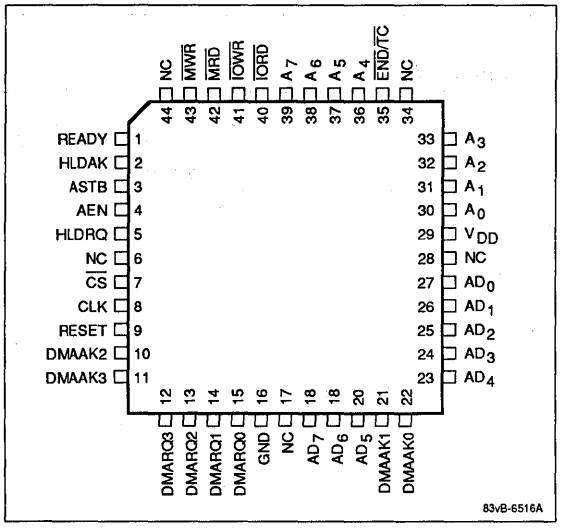


* Undefined; same non-function as μPD8237A

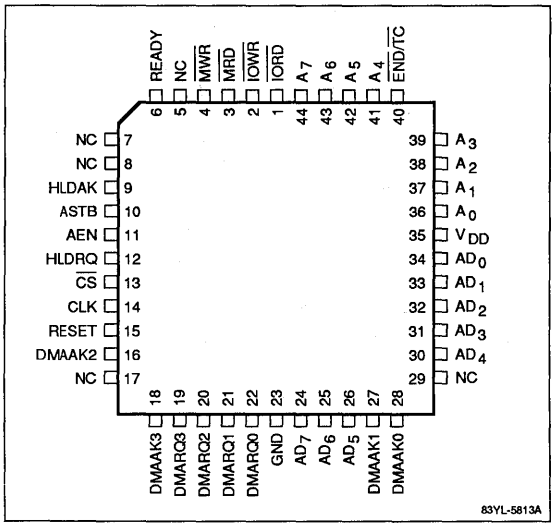
83YL-5812A

5b

44-Pin Plastic QFP



44-Pin PLCC



Pin Identification

Symbol	I/O	Function
A_0-A_3	3-state I/O	Four low-order bits of address bus
A_4-A_7	3-state out	Middle four bits of the output state address bus
AD_0-AD_7	3-state I/O	Eight high-order bits of the address and functions as an 8-bit data bus
AEN	Out	Permits output from an external latch connected to the μPD71037
ASTB	Out	Makes an external latch to the high-order address
CLK	In	Clock input for internal operations and data transfer speeds
$\overline{CS}$	In	Selects the μPD71037 as an I/O device and enables read/write operation
DMAAK0-DMAAK3	Out	Permits peripheral device to perform DMA transfer
DMARQ0-DMARQ3	In	Requests the μPD71037 to perform DMA transfer
$\overline{END/TC}$	I/O	Input that forces the μPD71037 to terminate DMA transfer; output that posts the end of DMA transfer
HLDAK	In	Permits the μPD71037 to hold the bus
HLDRQ	Out	Requests the host CPU to hold the bus
$\overline{IORD}$	3-state I/O	Input that enables the host CPU to read the μPD71037 status; output that enables the μPD71037 to read data from a peripheral device during DMA transfer
$\overline{IOWR}$	3-state I/O	Input that enables the host CPU to write data to the μPD71037; output that enables the μPD71037 to write data to a peripheral device during DMA transfer
$\overline{MRD}$	3-state out	Memory read during DMA transfer
$\overline{MWR}$	3-state out	Memory write during DMA transfer
READY	In	Requests extension of a read/write cycle during DMA transfer
RESET	In	Initializes the μPD71037
NC	—	No connection
V_{DD}	—	Positive power supply voltage
GND	—	Ground reference

PIN FUNCTIONS

A_0-A_3 (Address Bus)

A_0-A_3 input and output the four low-order bits of the address bus. During the inactive cycle, these pins are used as inputs to enable the host CPU to select an appropriate μPD71037 register. During the DMA cycle, A_0-A_3 output an address for memory access.

A_4-A_7 (Address Bus)

A_4-A_7 output the middle bits of the address bus. During the inactive cycle, these pins have high impedance. During the DMA cycle, A_4-A_7 output an address for memory access.

AD_0-AD_7 (Address/Data Bus)

AD_0-AD_7 input and output the high-order byte of an address bus and also function as a data bus. This is done by time multiplexing.

During the DMA cycle, the high-order byte of a memory address is output. During memory-to-memory transfer, memory addresses are output and these pins intermediate the memories for data transfer. During the inactive cycle, these pins act as a data bus when the host CPU reads or writes data from or to the μPD71037.

AEN (Address Enable)

AEN outputs an active-high AEN signal to enable an external latch which latches the high-order byte of an address during the DMA cycle. The AEN signal is fixed at a high level.

ASTB (Address Strobe)

ASTB outputs an active-high strobe signal to make an external latch retain the high-order byte of an address during the DMA cycle.

CLK (Clock)

CLK controls all internal μPD71037 operations and inputs a clock signal to control the DMA transfer speed. A maximum clock signal of 10 MHz can be input.

$\overline{CS}$ (Chip Select)

During the inactive cycle, $\overline{CS}$ inputs an active-low signal to enable the host CPU to handle the μPD71037 as an ordinary I/O device to read data to or write data from it. During the DMA cycle, this input is internally disabled to inhibit reading or writing by the host CPU.

DMAAK0-DMAAK3 (DMA Acknowledge)

DMAAK0-DMAAK3 indicate to peripheral devices that DMA service has been granted. DMAAK0-DMAAK3 respond respectively to DMA channels 0-3. Like the DMARQ pins, the active level of these pins is programmable; however, they can be set to an active-high level by a RESET input.

DMARQ0-DMARQ3 (DMA Request)

DMARQ0-DMARQ3 accept DMA service requests from peripheral devices. DMARQ0-DMARQ3 respond respectively to DMA channels 0-3 and the active level for these pins is programmable. However, they can be set to an active-high level by a RESET input. In fixed nest mode (see Channel Priority), the lowest priority is given to DMARQ3 and the highest to DMARQ0.

$\overline{END}/\overline{TC}$ (End/Terminal Count)

This is a bidirectional pin. The $\overline{END}$ input is used to terminate the current DMA transfer. $\overline{TC}$ indicates the designated cycles of the DMA count transfer have finished. If the low $\overline{END}$ signal is input during the DMA cycle, the μPD71037 forcibly terminates DMA services. If the specified number of data transfers is reached, the μPD71037 outputs a low $\overline{TC}$ signal.

When an $\overline{END}$ is input or $\overline{TC}$ is output, the μPD71037 clears the software DMA request (see Self-Initialization). The contents of the address set register are sent to the effective address register.

If self-initialization is not set, the $\overline{TC}$ bit of the status read register and the mask bit of the mask control register are set by the $\overline{END}/\overline{TC}$ signal. (The corresponding bit of the channel using DMA transfer is set.) In memory-to-memory transfer, $\overline{TC}$ is output when the specified number of DMA transfers is reached in channel 1.

If $\overline{END}$ is not input, pull up this pin to prevent a low signal from causing a malfunction.

HLDAK (Hold Acknowledge)

When active, HLDAK generates an active-high signal to indicate that the host CPU has granted the μPD71037 the use of the system bus. When this signal is input,the μPD71037 enters a DMA cycle.

HLDRQ (Hold Request)

HLDRQ outputs an active-high signal requesting the host CPU to hold the bus.

$\overline{IORD}$ (I/O Read)

$\overline{IORD}$ inputs or outputs an active-low signal to enable the host CPU to read μPD71037 data or to enable the μPD71037 to read from peripheral devices.

During the inactive cycle, the read signal is input to enable the host CPU to read a μPD71037 register. During the DMA cycle, the read signal is output to enable the μPD71037 to read data from a peripheral device.

$\overline{\text{IOWR}}$ (I/O Write)

$\overline{\text{IOWR}}$ inputs or outputs an active-low signal to enable the host CPU to write μPD71037 data or to enable the μPD71037 to write data to peripheral devices.

During the DMA cycle, the write signal is output to enable the μPD71037 to write data to a peripheral device.

During the inactive cycle, the write signal is input to enable the host CPU to write data to a μPD71037 register.

$\overline{\text{MRD}}$ (Memory Read)

During the DMA cycle, $\overline{\text{MRD}}$ outputs an active-low $\overline{\text{MRD}}$ signal to read data from memory. During the inactive cycle, this pin has high impedance.

$\overline{\text{MWR}}$ (Memory Write)

During the DMA cycle, $\overline{\text{MWR}}$ outputs an active-low $\overline{\text{MWR}}$ signal to write data into memory. During the inactive cycle, this pin has high impedance.

READY (Ready)

READY inputs an active-high signal to mark the end of each data transfer during a DMA operation. If a low-speed peripheral device fails to finish transferring data within a given read/write cycle, the width of the read/write signal output by the μPD71037 can be extended by inputting the low READY signal to this pin. During the extension, the μPD71037 enters a wait cycle (TW).

$\overline{\text{RESET}}$ (Reset)

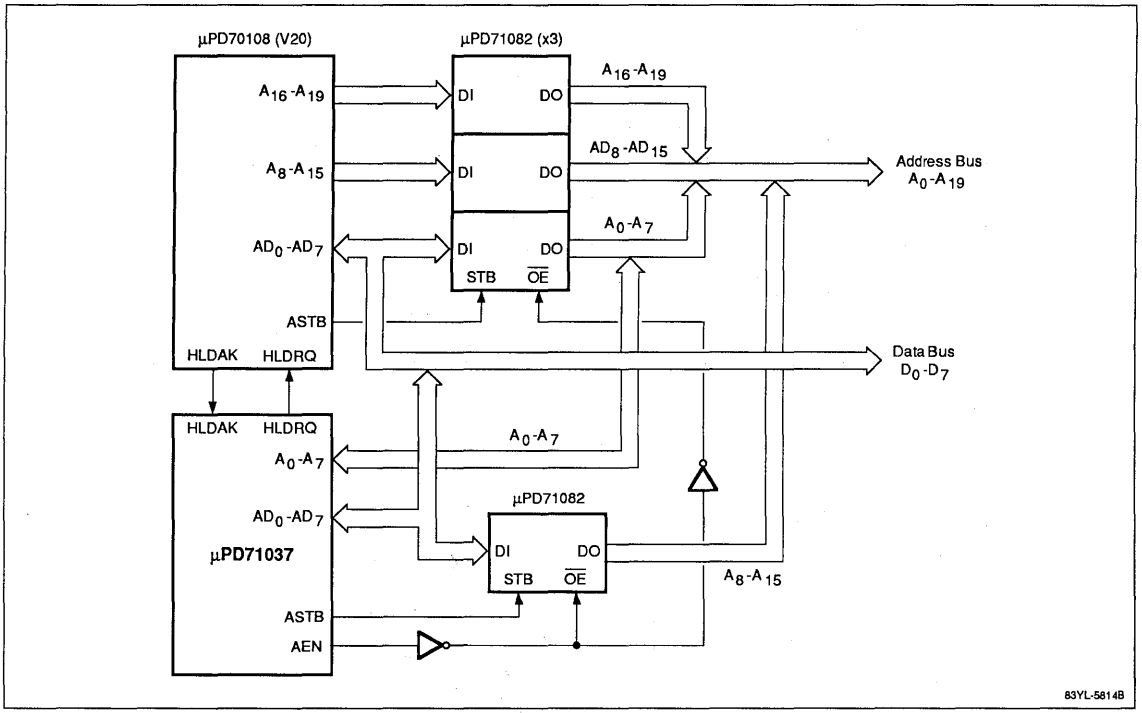
RESET inputs an active-high signal to initialize the internal μPD71037 statuses (register contents, etc.). Inputting this signal returns the μPD71037 to an inactive cycle.When RESET is input, the following control registers are cleared to 00H.

- Device control register
- Status read register
- Request control register
- Temporary data register

The DMA requests (DMARQ pin input) are masked for all four DMA channels. The control registers are described later in the Registers section.

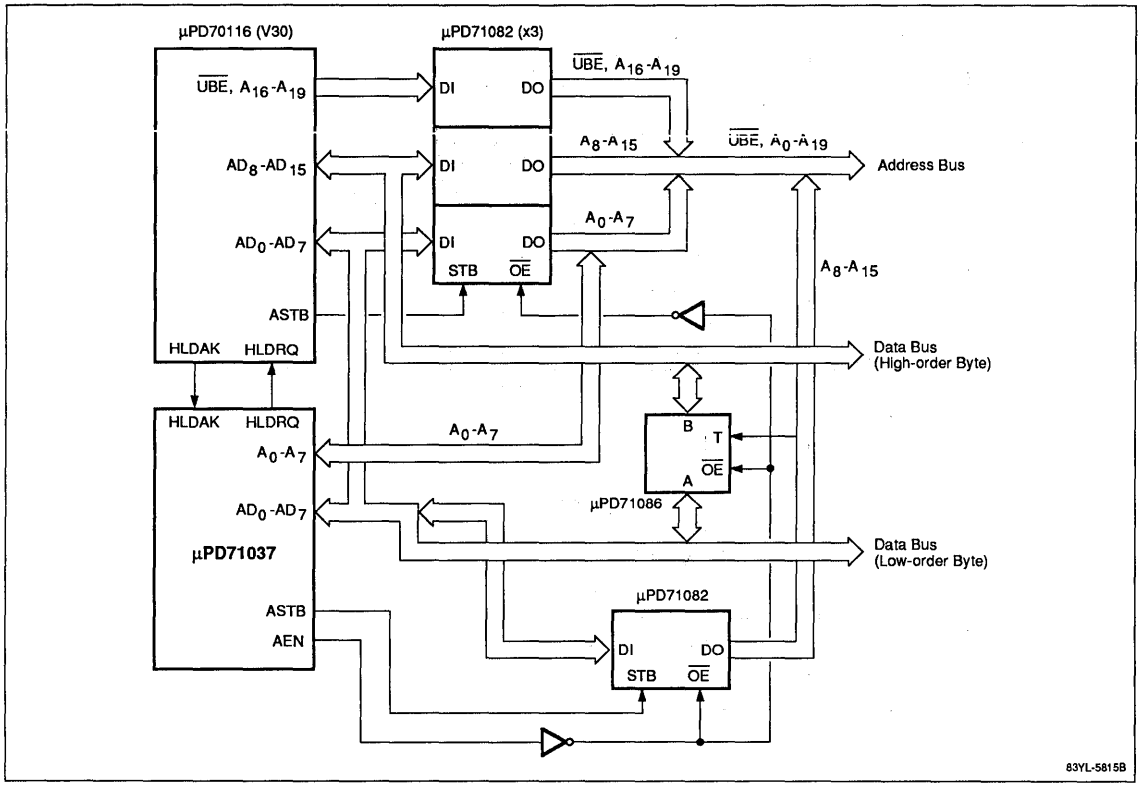
After RESET is input, an address-low byte command (see Commands section) is issued to the μPD71037 and the first byte is placed as the low-order byte in the address/count register.

System Configuration With μPD70108 (V20®) Microprocessor



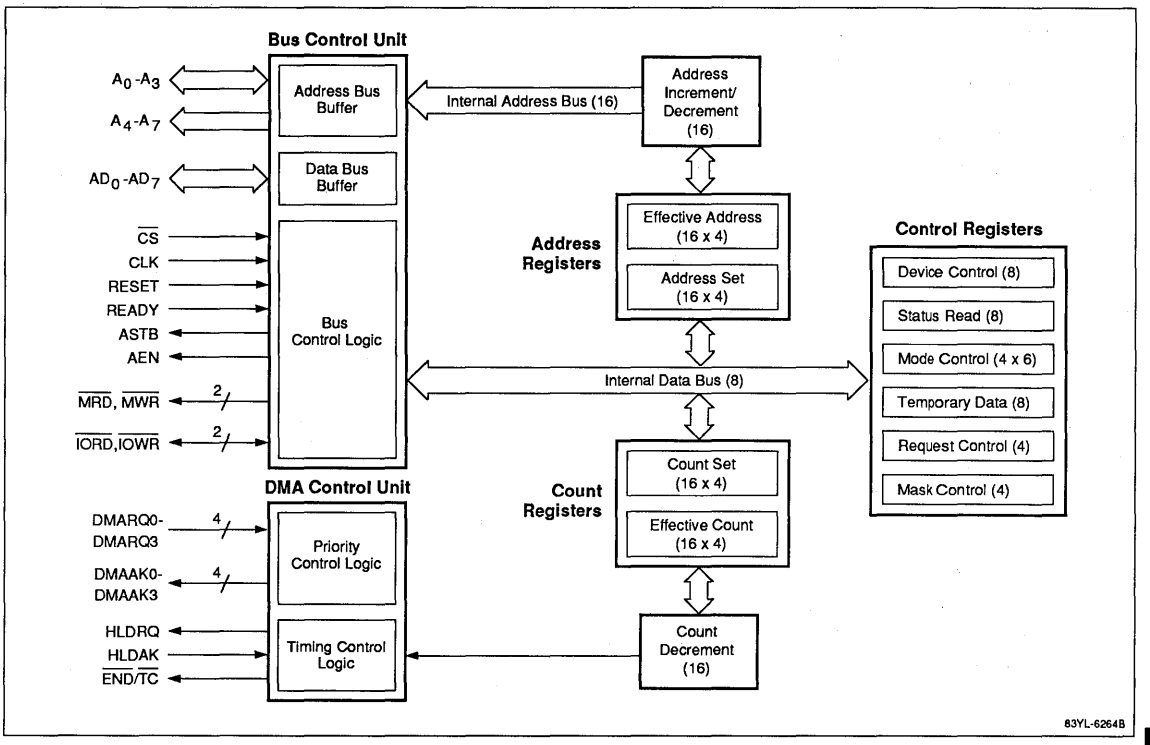
5b

System Configuration With μPD70116 (V30®) Microprocessor



V20 and V30 are registered trademarks of NEC Corporation

μPD71037 Block Diagram



INTERNAL BLOCK FUNCTIONS

The μPD71037 has the following functional units as shown in the block diagram.

- Bus control unit
- DMA control unit
- Address registers
- Address incrementer/decrementer
- Count registers
- Count decrementer
- Control registers

Bus Control Unit

The bus control unit consists of the address and data buffers and bus control logic. The bus control unit generates and receives signals that control addresses and data on the internal address and data buses.

DMA Control Unit

The DMA control unit contains the priority and timing control logic. The priority control logic determines the priority level of DMA requests and arbitrates the use of the bus in accordance with this priority level. The DMA control unit also provides internal timing and controls DMA operations.

Address Registers

Each of the four DMA channels has one address set register and one effective address register. Each register stores a DMA 16-bit address. The effective address register is updated for each single-byte DMA transfer and constantly holds the address to be transferred next. The contents of the address set register remain unchanged until the host CPU writes a new value to it. At self-initialization, the initial DMA address for the next DMA service is transferred from the address set register to the effective address register.

Address Incrementer/Decrementer

The address incrementer/decrementer updates the contents of the current address register whenever a DMA transfer completes.

5b

Count Registers

Each of the four DMA channels has one 16-bit count set register and one 16-bit effective count register that store a DMA transfer byte count. The count set register holds a value written by the CPU. At self-initialization the value is transferred to the effective address register where it is set as the number of DMA transfers in the next DMA service.

A channel's effective count register is decremented by 1 for each single-byte transfer and constantly holds the remaining number of DMA transfers. If a borrow occurs when this register is decremented, the terminal count is set to mark the end of the specified number of DMA transfers.

Count Decrementer

The count decrementer decrements the contents of the effective count register by 1 when each DMA transfer takes place.

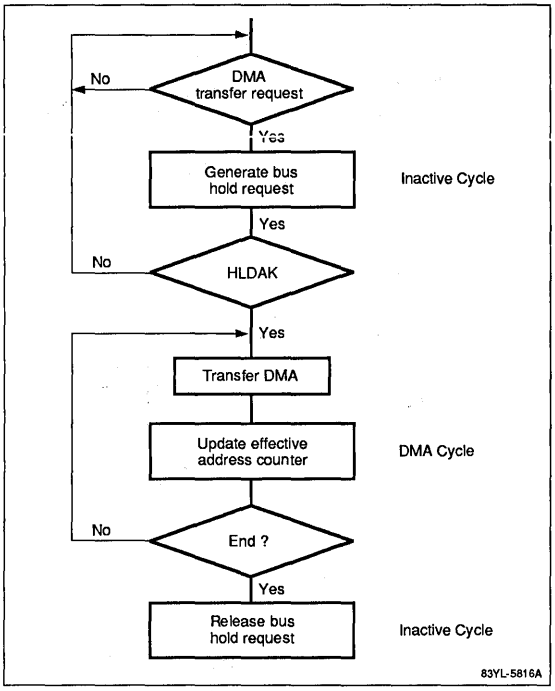
Control Registers

The μPD71037 contains six registers that control the bus mode, pin active levels (DMARQ and DMAAK), and the DMA transfer mode.

DMA OPERATION

The μPD71037 operates in an inactive cycle and a DMA cycle. Figure 1 illustrates basic DMA operation flow.

Figure 1. DMA Operation Flow



Inactive Cycle

During the inactive cycle, the host CPU has authority and the μPD71037 is in one of the following states.

- The μPD71037 has not yet received an effective DMA service request from a peripheral device.
- The μPD71037 has received an effective DMA service request, but has not yet received bus authority from the host CPU.
- The μPD71037 performs the following operations during the inactive cycle.
  — Detecting a DMA service request
  — Requesting bus authority
  — Selecting a DMA channel
  — Programming

In the inactive cycle, there are no active DMA cycles but there may be one or more active DMA requests. However, the CPU has not yet released the bus.

Detecting a DMA Service Request. The μPD71037 will sample the four DMARQ input pins for each clock signal.

Requesting Bus Authority. When an effective (unmasked) DMA request is received, a bus hold request signal (HLDRQ) is output to the host CPU. The μPD71037 continues to sample DMA requests until it obtains the bus by HLDAK input.

Selecting a DMA Channel. After the CPU returns an HLDAK signal and the μPD71037 obtains the bus, the μPD71037 stops DMA sampling and selects the DMA channel with the highest priority from the valid DMA request signals.

Programming. Before DMA transfer, the transfer addresses, the number of DMA transfers, the DMA transfer mode, and the active levels of the DMARQ and DMAAK pins must be determined.

While the host CPU holds bus authority, μPD71037 programming can be done by inputting a low signal to the $\overline{CS}$ pin. The four low-order address bits (A_0-A_3) specify a register for a read/write operation. Inputting an $\overline{IORD}$/$\overline{IOWR}$ signal performs the operation on the specified register.

DMA Cycle

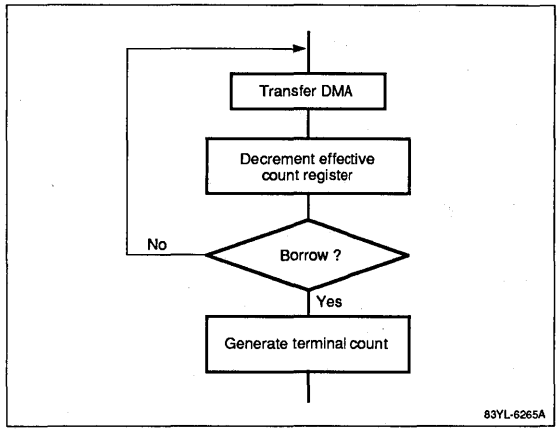
In a DMA cycle, the μPD71037 controls the bus and performs DMA transfer operations based on programmed information.

Terminal Count. External input of an $\overline{END}$ signal or internal generation of a terminal count terminates DMA transfer. The terminal count is generated when a borrow occurs as a result of decrementing the effective count register, which counts the number of DMA transfers in bytes. When this occurs, the μPD71037 outputs the low level pulse $\overline{TC}$.

Figure 2 shows that the effective count register is tested after each DMA operation. A borrow is detected after each DMA transfer is completed. As a result, the actual number of DMA transfers is one greater than the value set in the effective count register.

If self-initialization is not set when DMA service ends, the mask control register bits applicable to the channel where service is ended are set, and the DMARQ input of that channel is masked.

Figure 2. Generation of Terminal Count (TC)



DMA Transfer Type

The type of transfer the μPD71037 performs depends on the following conditions.

- Memory-to-memory transfer enable
- Direction of I/O-to-memory transfer for each channel
- Transfer mode of each channel

Memory-to-Memory Transfer Enable. The μPD71037 performs each DMA transfer (1 byte of data) between an I/O device and memory in a one-bus cycle and between memories in a two-bus cycle.

5b

Memory-to-memory transfer can occur only when bit 0 of the device control register is set to 1. The DMA channels used in memory-to-memory transfers are fixed, with channel 0 as the source channel and channel 1 as the destination channel. Channels 2 and 3 cannot be used in memory-to-memory transfers. The contents of the count registers of each channel should be the same when performing this type of DMA transfer.

The μPD71037 performs the following operations until a channel 0 terminal count or until $\overline{END}$ input is present. Only the block mode is valid for this type of transfer.

(1) The memory data pointed to by the effective address register of channel 0 is read into the temporary data register of the μPD71037 and the effective address register and effective count register of channel 0 are updated.

(2) The temporary register data is written to the memory location shown by the effective address register of channel 1; the effective address register and effective count register are updated.

During memory-to-memory transfers, the address of the transfer source can be fixed using the device control register. In this manner, a range of memory can be padded with the same value (0 or 1) since the contents of the source address never change. During memory-to-memory transfer, the DMAAK signal and channel 0's terminal count ($\overline{TC}$) are not output.

Note: If DMARQ1 (channel 1) becomes active, the μPD71037 will perform memory-to-I/O transfer even though memory-to-memory transfer is selected. Since this may cause erroneous memory-to-memory transfers, mask out channel 1 (DMARQ1) by setting bit 1 of the mask register to 1 before starting memory-to-memory transfers.

Direction of I/O-to-Memory Transfers. All DMA transfers use memory as a reference point. Therefore, a DMA read reads a memory location and writes to an I/O port. A DMA write reads an I/O port and writes the data to a memory location. In memory-to-I/O transfer, use the mode control register to set one of the transfer directions in table 1 for each channel and activate the appropriate control signals.

Table 1. Transfer Direction

Transfer Direction	Activated Signals
Memory to I/O (DMA read)	$\overline{IOWR}$, $\overline{MRD}$
I/O to memory (DMA write)	$\overline{IORD}$, $\overline{MWR}$
Verify (Outputs addresses only. Does not perform a transfer.)	—

Transfer Modes. In I/O-to-memory transfer, the mode control register selects the single, demand, or block mode of DMA transfer for each channel. Table 2 shows the various transfer modes and termination conditions.

Table 2. Transfer Termination

Transfer Mode	End of Transfer Condition
Single	After 1 byte of data is transferred
Demand	$\overline{END}$ input Generation of terminal count When DMARQ for the channel in DMA service becomes inactive
Block	$\overline{END}$ input Generation of terminal count

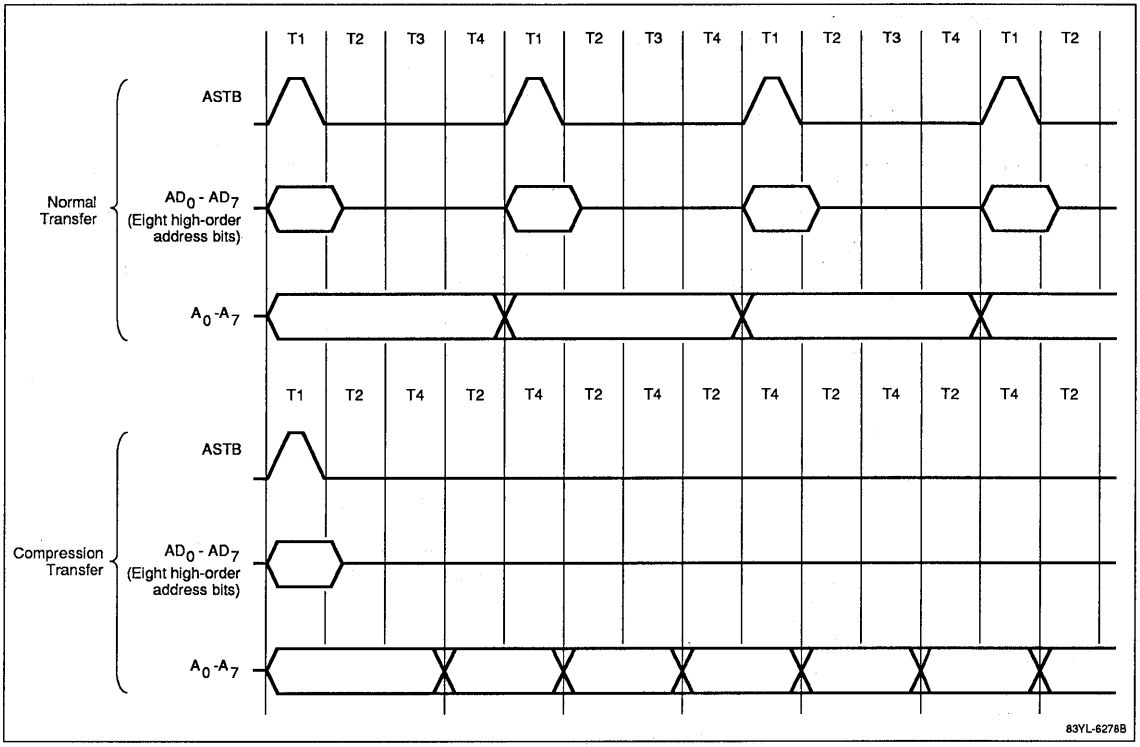
Compressed Timing

DMA transfer cycles are normally executed in four states for each bus cycle. However, when the device control register selects compressed timing, one DMA cycle can be executed in two states (T2 and T4) for each bus cycle. See figure 3.

Compressed timing may be used in block mode or demand mode. (Memory-to-memory transfer is excluded.) In this way, compressed timing permits twice the amount of data to be transferred when compared with normal DMA transfers.

In block mode or demand mode, addresses are output sequentially and the high-order address byte latched in external latches need not be updated except after a carry or borrow from the low-order byte. For this reason, the T1 state is omitted in the bus cycles except during the first bus cycle when the high-order address byte is changed.

Figure 3. Normal and Compression Transfer Timing



Software DMA Requests

The μPD71037 can accept software DMA requests in addition to DMA requests from the four DMARQ pins. Setting the appropriate bit in the request register generates a software request. The mask register does not mask software requests. Software DMA requests operate differently depending on which bus or transfer mode is used.

Single or Demand Mode. When single or demand mode is set, the applicable request bits for the corresponding channel are cleared, and software DMA service ends with the transfer of 1 byte of data (see Request Control Register).

Block Mode (Memory-to-Memory Transfer). When block mode or memory-to-memory mode is set, service continues until $\overline{END}$ is input or a terminal count is generated. In memory-to-memory transfer, only the request bit for channel 0 is cleared.

Self-Initialization

When the mode control register is set to self-initialization, the μPD71037 automatically initializes the address and count registers when $\overline{END}$ is input or a terminal count is generated. The contents of the address set register and the count set register are transferred to the effective address register and the effective count register, respectively. The applicable bit of the mask register is not affected. The applicable bit of the mask register is set for channels not programmed for self-initialization.

Channel Priority

Each of the μPD71037's four DMA channels is assigned a priority. When there are DMA requests from several channels simultaneously, the channel with the highest priority will be serviced. The device control register selects one of two channel priority methods: fixed nest mode or rotation nest mode.

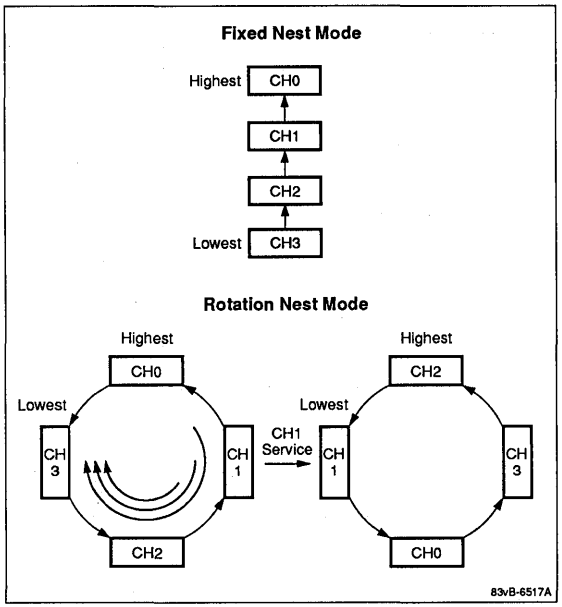
In fixed nest mode, the priority (starting with the highest) is channel 0, 1, 2, and 3, respectively. In rotation nest mode, priority is rotated so that the channel that has just

5b

been given service receives the lowest priority and the next highest channel number is given the highest priority. This method prevents exclusive servicing of some channels.

Figure 4 shows the two priority order methods.

Figure 4. DMA Channel Priorities



Cascade Connection

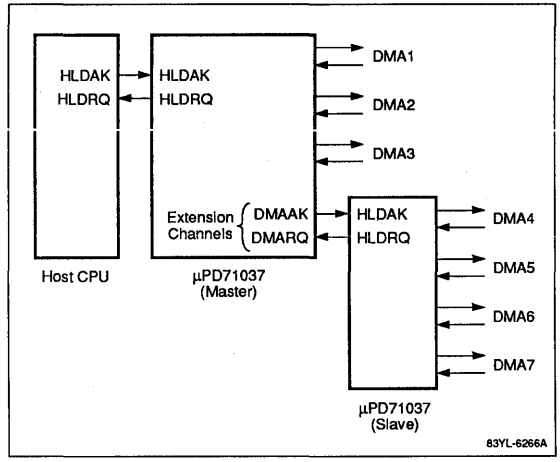
The μPD71037 can be cascaded to expand the system DMA channel capacity. To connect a μPD71037 for cascading (figure 5), perform the following operations.

(1) Connect pins HLDRQ and HLDAK of the second-stage (slave) μPD71037 to pins DMARQ and DMAAK of any channel of the first-stage (master) μPD71037.

(2) To select cascade mode of a particular channel of a master μPD71037, set bits 7 and 6 of that channel's mode control register to 11.

When a channel is set to cascade mode in a master μPD71037, DMARQ, DMAAK, HLDRQ, HLDAK, and RESET are the only valid signals in that master μPD71037. The other signals are disabled. The master cascade channel intermediates only hold request/hold acknowledge between slave and the host CPU.

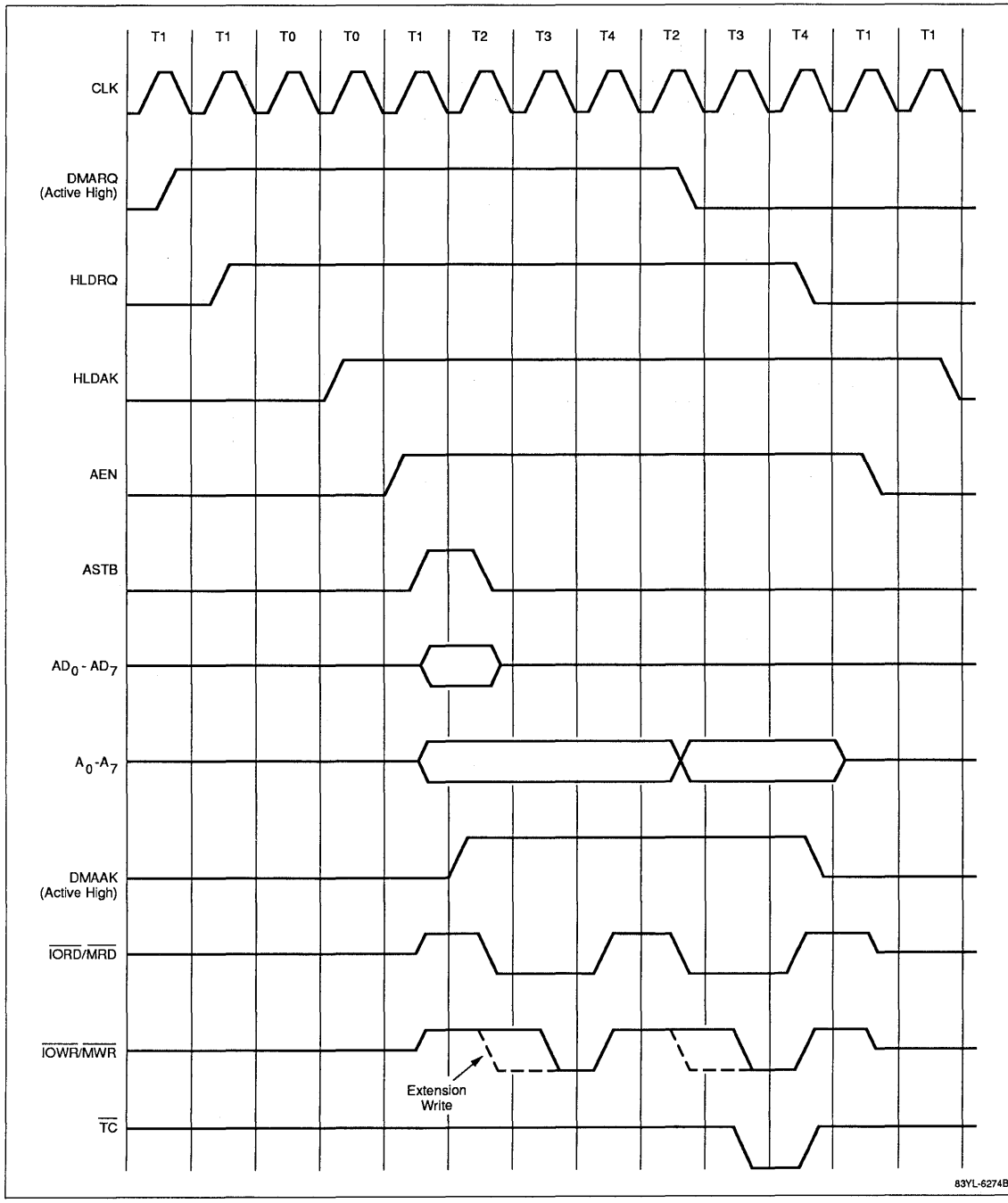
Figure 5. Cascade Connection Example



DMA Transfer Timing

Figures 6 through 9 are timing diagrams for the DMA transfer operations described previously.

Figure 6. I/O-to-Memory Transfer, Normal Mode Timing



5b

Figure 7. Memory-to-Memory Transfer Timing

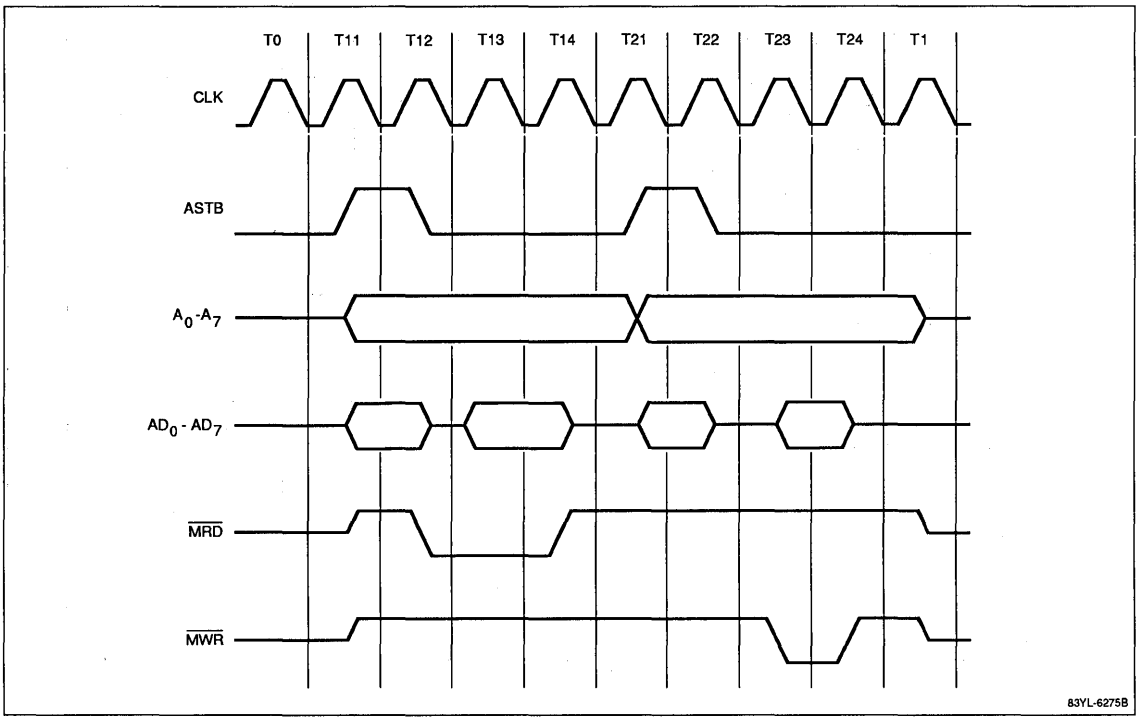


Figure 8. I/O-to-Memory Transfer, Compression Mode Timing

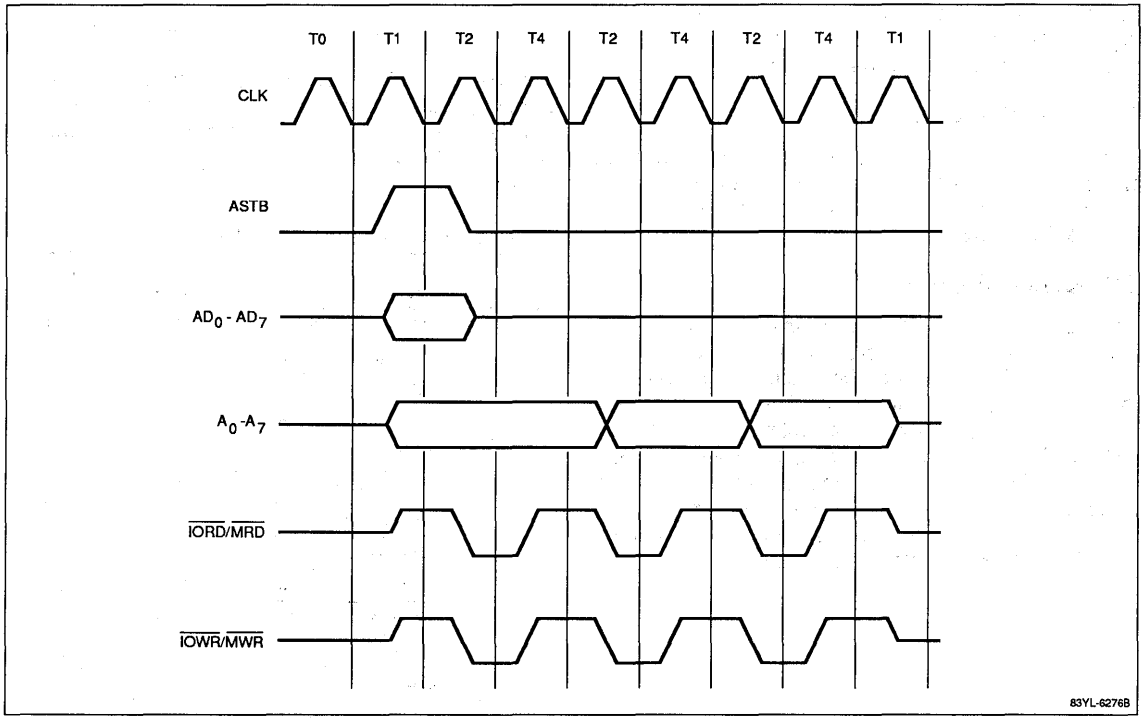
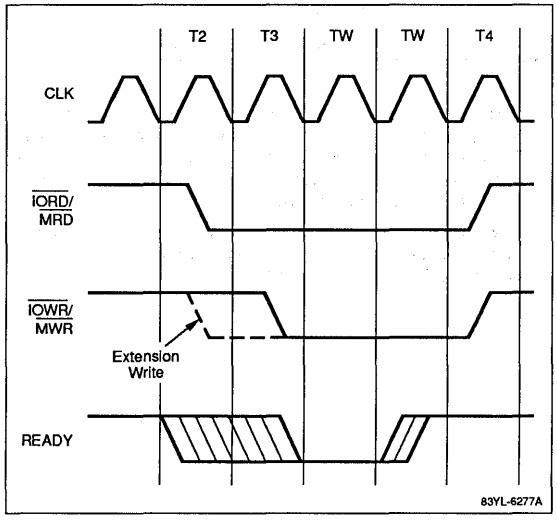


Figure 9. READY Timing



REGISTERS

The μPD71037 registers are considered to be address/count registers or control registers.

- Address/Count Registers
  - Effective address register
  - Set address register
  - Effective count register
  - Set count register
- Control Registers
  - Device control register
  - Mode control register
  - Request control register
  - Mask control register for each channel
  - Mask control register for all channels
  - Temporary data register
  - Status read register

Each address/count register consists of 16 bits. A read/write operation for this type of register is performed by accessing 2 bytes of an I/O address. The low-order address byte is accessed first, followed by the high-order address byte. To set a new value in an address/count register, write its low-order byte first by issuing an address low-byte command. The commands you can

use are presented later in Commands. Table 3 lists the address/count registers for each channel. Each control register consists of 8 bits. Table 4 lists the type of output and addresses for the control registers.

Effective Address Register

A 16-bit effective address register is assigned to each channel. It holds the address to be output during DMA transfer. This register is updated by ±1 for each single-byte DMA transfer.

Set Address Register

A 16-bit set address register is assigned to each channel. It contains the initial value of a DMA transfer address set by the host CPU. Unlike the effective address register, this register retains its contents until the host CPU writes a new byte count into it. At self-initialization, the contents of the set address register are transferred to the effective address register at the initial address register. Values are written into the set address register by writing its 2 bytes in succession. However, the host CPU cannot read the address data from this register.

Effective Count Register

The effective count register is a 16-bit register assigned to each channel. It contains the remaining byte count for DMA transfer. This register is decremented by 1 for each single-byte DMA transfer. As in the effective address register, a value is read from or written into the effective count register by reading or writing its 2 bytes in succession.

Table 3. Address/Count Registers

Channel	Register	R/W	Address (Bits A_3-A_0)
0	Set address register	W	0H
	Effective address register	R/W	
	Set count register	W	1H
	Effective count register	R/W	
1	Set address register	W	2H
	Effective address register	R/W	
	Set count register	W	3H
	Effective count register	R/W	
2	Set address register	W	4H
	Effective address register	R/W	
	Set count register	W	5H
	Effective count register	R/W	
3	Set address register	W	6H
	Effective address register	R/W	
	Set count register	W	7H
	Effective count register	R/W	

Notes:

(1) When a new value is written into a set address/count register, it is simultaneously written into an effective address/count register. Therefore, when setting an address and count, you need not consider the differences between a set address/count register and an effective address/count register.

(2) The set address/count registers are used only for writing. If an attempt is made to read these registers, the effective address/count registers are read instead.

Table 4. Control Registers

Register	R/W	Address (Bits A_3-A_0)
Device control register	W	8H
Status read register	R	
Request control register	W	9H
Mask control register (each channel)	W	AH
Mode control register	W	BH
Temporary data register	R	DH
Mask control register (all channels)	W	FH

Notes:

(1) An I/O address other than listed in this table is assigned to address/count registers or commands; otherwise, the address cannot be accessed.

(2) The I/O address DH (in the temporary data register) is assigned to a software reset command when it is written. Refer to the Commands section for details.

Set Count Register

A 16-bit set count register is assigned to each channel. It holds the initial value of the DMA transfer byte count written by the host CPU. Unlike the effective count register, the set count register retains its data until the host CPU writes a new byte count into it. A value is written into the set count register by writing to its 2 bytes in succession. However, this register is read-protected.

Device Control Register

The 8-bit device control register controls the DMA transfer modes, determines whether to permit or inhibit DMA operation, controls the active levels of DMARQ and DMAAK, and determines whether to permit or inhibit memory-to-memory transfer.

Figure 10 shows the format of the device control register and table 5 describes the bits .

Figure 10. Device Control Register

AKL	RQL	EXW	ROT	CMP	DDMA	AHLD	MTM
7	6	5	4	3	2	1	0

Address 8H

AKL	**DMAAK Active Level**
0	Active low
1	Active high
RQL	**DMARQ Active Level**
0	Active high
1	Active low
EXW	**Extension Writing**
0	Normal writing
1	Extension writing
ROT	**Rotation Nesting**
0	Fixed nest mode
1	Rotation nest mode
CMP	**Compression Transfer**
0	Normal transfer mode
1	Compression transfer mode
DDMA	**DMA Operation**
0	Permit
1	Inhibit
AHLD	**Channel 0 Address Hold**
0	Inhibit address fixing
1	Permit address fixing
MTM	**Memory-to-Memory Transfer**
0	Inhibit
1	Permit

5b

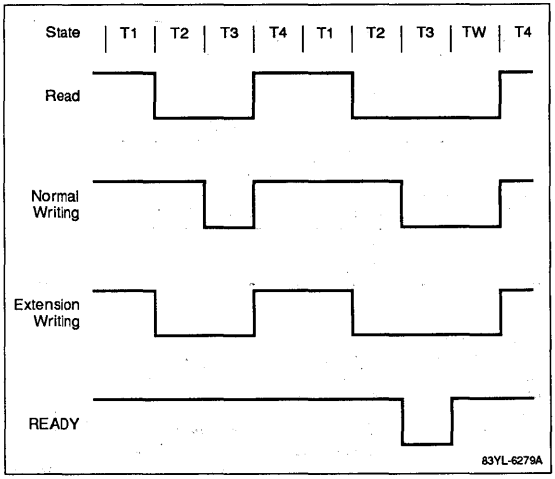
Table 5. Device Control Register Bits

Bit	Symbol	Description
7	AKL	DMA acknowledge active level. This bit specifies the active levels of the four DMAAK output signals. Setting this bit to 1 indicates the active-high DMAAK signals.
6	RQL	DMA request level. This bit specifies the active levels of the four DMARQ input signals. Setting this bit to 0 indicates the active-high DMARQ signals.
5	EXW	Extended write. When this bit is set, the μPD71037 outputs a write signal at the same timing as a read signal (extension writing, figure 11). You cannot specify extension writing during compressed transfer.
4	ROT	Rotate priority. Setting this bit determines DMA channel priorities in rotation nest mode.

Table 5. Device Control Register Bits (cont)

Bit	Symbol	Description
3	CMP	Compressed timing. When this bit is set, DMA transfer in the block or demand modes occurs at compression timing. Compression transfer mode must not be specified during memory-to-memory transfer.
2	DDMA	Disable DMA. While this bit is set, the μPD71037 does not output the HLDRQ signal to the host CPU even if it receives an effective DMA request. It also prevents invalid DMA transfer during μPD71037 programming.
1	AHLD	Channel 0 address hold. If memory-to-memory transfer is permitted, setting this bit fixes the address of the transfer source channel 0. If memory-to-memory transfer is inhibited, this bit is meaningless.
0	MTM	Memory to memory. Setting this bit permits memory-to-memory transfer.

Figure 11. Timing for Extension Writing



Mode Control Register

The mode control register specifies DMA transfer mode for each channel. Figure 12 shows the format and table 6 describes the bits.

Figure 12. Mode Control Register

TMODE		ADIR	SEFI	TDIR		SELCH	
7	6	5	4	3	2	1	0

Address 8H

TMODE	**DMA Transfer Mode**
0 0	Demand mode
0 1	Single mode
1 0	Block mode
1 1	Extension mode
ADIR	**Address Direction**
0	Increment an address
1	Decrement an address
SEFI	**Self-Initialization**
0	Inhibit
1	Permit
TDIR	**Transfer Direction**
0 0	Verify transfer
0 1	I/O-to-memory transfer
1 0	Memory-to-I/O transfer
1 1	Inhibit transfer
SELCH	**Channel Selection**
0 0	Channel 0
0 1	Channel 1
1 0	Channel 2
1 1	Channel 3

Table 6. Mode Control Register Bits

Bits	Symbol	Description
7, 6	TMODE	Transfer mode. These bits indicate the DMA transfer mode for I/O-to-memory transfer (meaningless during memory-to-memory transfer because block mode is automatically selected).
5	ADIR	Address direction. This bit indicates whether the effective address register is incremented or decremented. If set to 0, the register is incremented by 1 for each single-byte transfer. If set to 1, it is decremented by 1 for each single-byte transfer.
4	SEFI	Self-initialization. In memory-to-memory transfer, assign the same value to the SEFI bits of channel 0 (transfer source) and channel 1 (transfer destination).
3, 2	TDIR	Transfer direction. These bits specify the I/O-to-memory transfer direction (meaningless during memory-to-memory transfer).
1, 0	SELCH	Select channel. These bits specify the DMA channel to which DMA transfer modes are specified by bits 7-2 .

Status Read Register

The status read register indicates whether a DMA request or terminal count is generated and whether the $\overline{END}$ signal is externally input. This information is set for each channel. Figure 13 shows the register format and table 7 describes the bits.

Figure 13. Status Read Register

RQ3	RQ2	RQ1	RQ0	TC3	TC2	TC1	TC0
7	6	5	4	3	2	1	0

Address 8H

RQ3-RQ0	Hardware DMA Request (Channels 3-0)
0	Request not generated
1	Request generated
TC3-TC0	**Terminal Count (Channels 3-0)**
0	Transfer not yet terminated
1	$\overline{END}$ input or terminal count occurs

Table 7. Status Read Register Bits

Bits	Symbol	Description
7-4	RQ3-RQ0	DMA request. These bits specify whether a DMA hardware request is generated by inputting the DMARQ signal. If the corresponding channel is masked, these bits are reset as long as the DMARQ input is active for this channel. A hardware DMA request held by a mask can be detected by sampling these bits.
3-0	$\overline{TC3}$-$\overline{TC0}$	Terminal count. These bits specify whether the $\overline{END}$ signal is input or a terminal count is generated. This information is set for each channel. If a terminal count occurs within a channel or an $\overline{END}$ signal is externally input, the corresponding channel bit is set. These bits are reset each time the status read register is read.

Temporary Data Register

The temporary data register contains the data transferred last during memory-to-memory transfer. Figure 14 shows the register format.

Figure 14. Temporary Data Register

TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
7	6	5	4	3	2	1	0

Address DH

Request Control Register

The request control register controls software DMA requests. Figure 15 shows the register format and table 8 describes the bits.

Figure 15. Request Control Register

—	—	—	—	—	SRQ	SELCH	
7	6	5	4	3	2	1	0

Address 9H

SRQ	Software DMA Request
0	Reset DMA request bit
1	Set DMA request bit
SELCH	**Select Channel**
0 0	Channel 0
0 1	Channel 1
1 0	Channel 2
1 1	Channel 3

Table 8. Request Control Register Bits

Bits	Symbol	Description
2	SRQ	Software DMA request. This bit controls a software DMA request for the channel selected by bits 0 and 1. Setting this bit to 1 sets the request bit of the selected channel. Setting it to 0 resets the request bit.
1, 0	SELCH	Select channel. These bits specify the channel for which software DMA request control is executed.

Mask Control Registers

The mask control register controls the DMA request mask for each channel. Two types of mask control registers are provided. The first is a register that masks each of the channels separately (figure 16). The second is a register that masks all of the channels at once (figure 17). Table 9 describes the mask control register bits.

Figure 16. Mask Control Register (Each Channel)

—	—	—	—	—	MSET	SELCH	
7	6	5	4	3	2	1	0

Address AH

MSET	DMARQ Mask
0	Reset mask
1	Set mask
SELCH	**Select Channel**
0 0	Channel 0
0 1	Channel 1
1 0	Channel 2
1 1	Channel 3

5b

Figure 17. Mask Control Register (All Channels)

—	—	—	—	M3	M2	M1	M0
7	6	5	4	3	2	1	0

Address FH

M3-M0	DMARQ Mask (Channels 3-0)
0	Reset mask
1	Set mask

Table 9. Mask Control Register Bits

Bits	Symbol	Description
Each Channel		
2	MSET	Mask set. This bit masks the DMA request (DMARQ pin input) for the channel selected in bits 1 and 0.
1, 0	SELCH	Select channel. These bits select the channel to mask.
All Channels		
3-0	M3-M0	Mask. These bits specify whether a mask is set or reset for the four DMA channels. Setting any of the bits to 1 masks the respective channel and the DMA request (DMARQ pin input) for this channel is inhibited. Setting any of these bits to 0 resets the mask.

COMMANDS

The μPD71037 supports three commands (in addition to the registers) to control data transfer operations.

- Address low-byte command
- Software reset command
- Clear-all-masks command

The host CPU writes data into these commands (table 10) to enable them to control μPD71037 operation. Unlike the registers, any data can be written into the commands.

Table 10. List of Commands

Command	R/W	Address (Bits A_3-A_0)	Function
Address low-byte	W	CH	This command must be issued before a new value is set in an address/count register. The low-order byte of this new value is set first and then its high-order byte is set.
Software reset	W	DH	The reset operation performed by this command is the same as the normal hardware reset operation. The reset operation clears the device control register, status read register, request control register, and temporary data register to 00H. Masks are set for all channels.
Clear-all-masks	W	EH	The masks for all channels are released and the reception of a DMA transfer request is permitted.

Notes:

(1) After the reset operation by hardware or software, a value is written in the low-order byte of the address/count register.

(2) If an attempt is made to read I/O address DH, the temporary data register is read instead. See table 4.

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = 25°C

Supply voltage, V_{DD}	−0.5 to 7.0 V
Input voltage, V_I	−0.5 to V_{DD}+0.3 V
Output voltage, V_O	−0.5 to V_{DD}+0.3 V
Operating temperature range, T_{OPT}	−40 to +85°C
Storage temperature range, T_{STG}	−65 to +150°C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Capacitance

T_A = 25°C

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Input capacitance	C_I		8	15	pF	fc = 1MHz; unmeasured pins returned to 0 V
Output capacitance	C_O		4	8	pF	
I/O capacitance	C_{IO}		10	18	pF	

DC Characteristics

T_A = −40 to +85°C; V_{DD} = +5.0 V ± 10%

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Input voltage, low	V_{IL1}	−0.5		0.6	V	CLK pin
	V_{IL2}	−0.5		0.8	V	All except CLK
Input voltage, high	V_{IH1}	3.3		V_{DD} + 0.3	V	CLK pin
	V_{IH2}	2.2		V_{DD} + 0.3	V	All except CLK
Output voltage, low	V_{OL}			0.4	V	I_{OL} = 2.5 mA; I_{OL} = 4.5 mA (TC only)
Output voltage, high	V_{OH}	0.7 V_{DD}			V	I_{OH} = −400 μA
Input leakage current	I_{LI}			±10	μA	V_I = 0 V to V_{DD}
Output leakage current	I_{LO}			±10	μA	V_O = 0 V to V_{DD}
Power supply current	I_{DD}			20	mA	10-MHz operation

5b

AC Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ± 10%

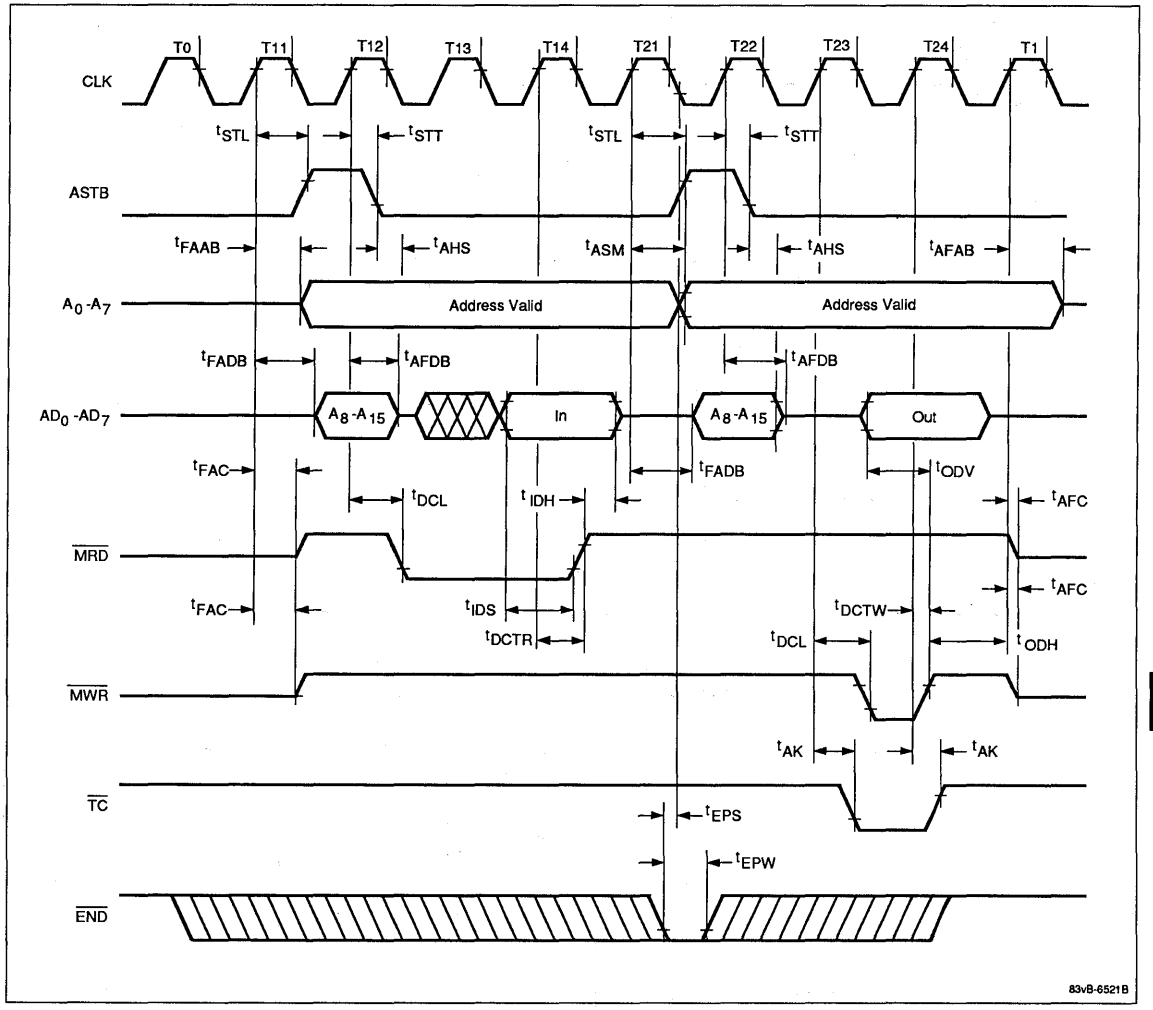
Parameter	Symbol	Min	Max	Unit	Conditions
DMA Master Mode					
AEN high delay time from CLK low	t_{AEL}		100	ns	S1
AEN low delay time from CLK high	t_{AET}		70	ns	S1
Address active to float delay from CLK high	t_{AFAB}		80	ns	
$\overline{IORD}/\overline{MRD}$ or $\overline{IOWR}/\overline{MWR}$ float from CLK high	t_{AFC}		80	ns	
DB active to float delay from CLK high	t_{AFDB}		120	ns	
Address hold time from $\overline{IORD}/\overline{MRD}$ high	t_{AHR}	t_{CY} − 100		ns	
DB hold time from ASTB low	t_{AHS}	20		ns	
Address hold time from $\overline{IOWR}/\overline{MWR}$ high	t_{AHW}	t_{CY} − 40		ns	
DMAAK valid delay time from CLK low	t_{AK}		100	ns	
$\overline{TC}$ high delay time from CLK high	t_{AK}		100	ns	
$\overline{TC}$ low delay time to CLK high	t_{AK}		90	ns	
Address stable from CLK high	t_{ASM}		80	ns	
Data bus setup time to ASTB low	t_{ASS}	40		ns	
Clock pulse width, high	t_{CH}	39		ns	Transitions ≤ 10 ns

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
DMA Master Mode (cont)					
Clock pulse width, low	t_{CL}	45		ns	Transitions ≤ 10 ns
CLK cycle	t_{CY}	100	1000	ns	
CLK high delay to $\overline{IORD}/\overline{MRD}$ or $\overline{IOWR}/\overline{MWR}$ low	t_{DCL}	10	80	ns	
$\overline{IORD}/\overline{MRD}$ high delay time from CLK high	t_{DCTR}	10	80	ns	S4
$\overline{IOWR}/\overline{MWR}$ high delay time from CLK high	t_{DCTW}	10	55	ns	S4
HLDRQ valid delay time from CLK high	t_{DQ1}/ t_{DQ2}		70	ns	
$\overline{END}$ low setup time from CLK low	t_{EPS}	25		ns	
$\overline{END}$ pulse width	t_{EPW}	100		ns	
Address float to active delay from CLK high	t_{FAAB}		80	ns	
$\overline{IORD}/\overline{MRD}$ or $\overline{IOWR}/\overline{MWR}$ active from CLK high	t_{FAC}		80	ns	
Data bus float to active delay from CLK high	t_{FADB}		70	ns	
HLDAK valid setup time to CLK high	t_{HS}	50		ns	
Input data hold time from $\overline{MRD}$ high	t_{IDH}	20		ns	
Input data setup time to $\overline{MRD}$ high	t_{IDS}	90		ns	
Output data hold time from $\overline{MWR}$ high	t_{ODH}	10		ns	
Output data valid to $\overline{MWR}$ high	t_{ODV}	65		ns	
DMARQ setup time to CLK low	t_{QS}	20		ns	S1, S4
CLK hold time to READY low	t_{RH}	20		ns	
READY setup time to CLK low	t_{RS}	25		ns	
ASTB high delay time from CLK high	t_{STL}		70	ns	
ASTB low delay time from CLK high	t_{STT}		70	ns	
Peripheral (Slave) Mode					
Address valid or $\overline{CS}$ low to $\overline{IORD}/\overline{MRD}$ low	t_{AR}	35		ns	
Address valid setup time to $\overline{IOWR}/\overline{MWR}$ high	t_{AW}	80		ns	
$\overline{CS}$ low setup time to $\overline{IOWR}/\overline{MWR}$ high	t_{CW}	90		ns	
Data valid setup time to $\overline{IOWR}/\overline{MWR}$ high	t_{DW}	80		ns	
Address or $\overline{CS}$ hold from $\overline{IORD}/\overline{MRD}$ high	t_{RA}	0		ns	
Data access from $\overline{IORD}/\overline{MRD}$ low	t_{RDE}		120	ns	
Data bus float delay from $\overline{IORD}/\overline{MRD}$ high	t_{RDF}	0	70	ns	
Power supply setup time high to RESET low	t_{RSTD}	500		ns	
RESET to first $\overline{IORD}$ or $\overline{IOWR}$	t_{RSTS}	t_{CY}		ns	
RESET pulse width	t_{RSTW}	200		ns	
$\overline{IORD}/\overline{MRD}$ width	t_{RW}	150		ns	
Address hold time from $\overline{IOWR}/\overline{MWR}$ high	t_{WA}	15		ns	
$\overline{CS}$ hold time high from $\overline{IOWR}/\overline{MWR}$ high	t_{WC}	15		ns	
Data hold time from $\overline{IOWR}/\overline{MWR}$ high	t_{WD}	20		ns	
$\overline{IOWR}/\overline{MWR}$ width	t_{WWS}	90		ns	
Access recovery time		125		ns	

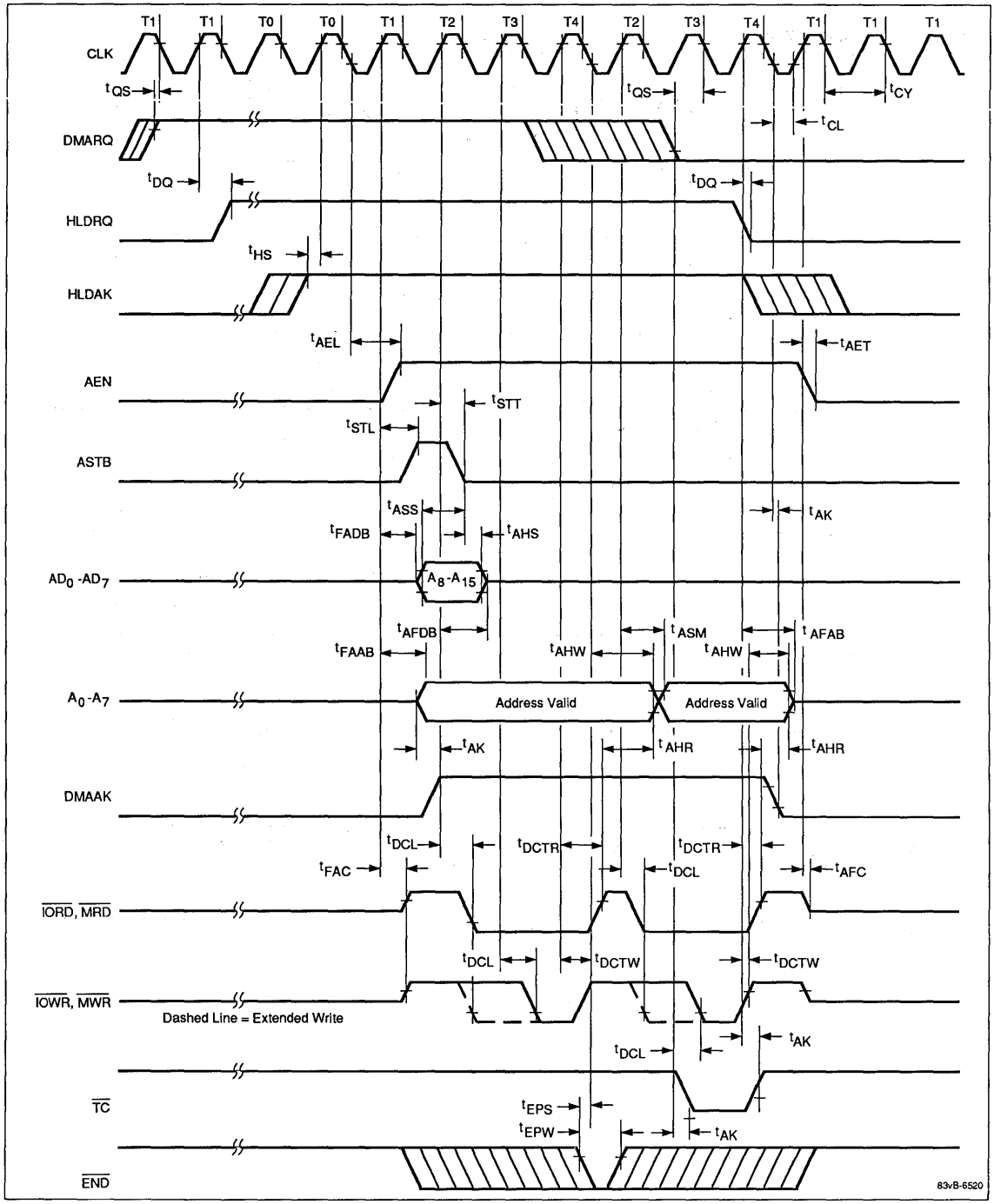
Timing Waveforms

Memory-to-Memory Transfer

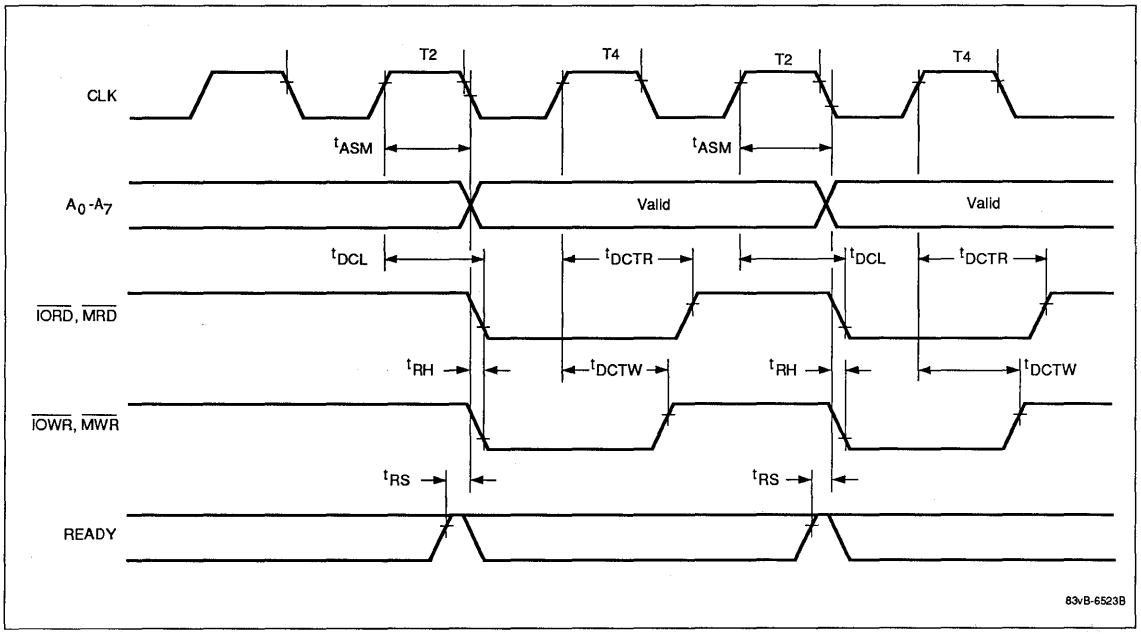


5b

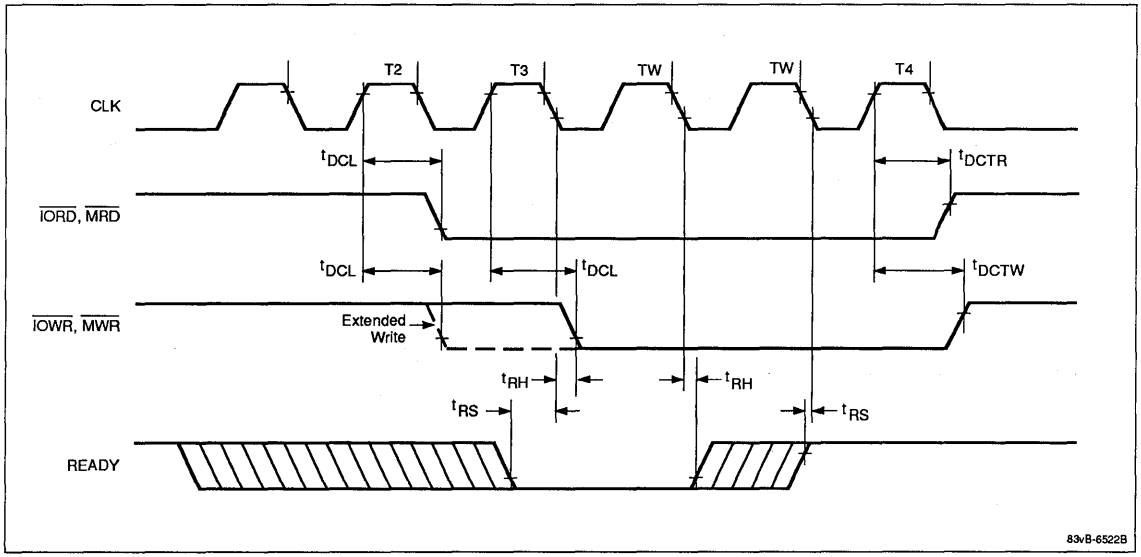
DMA Transfer



Compressed Transfer

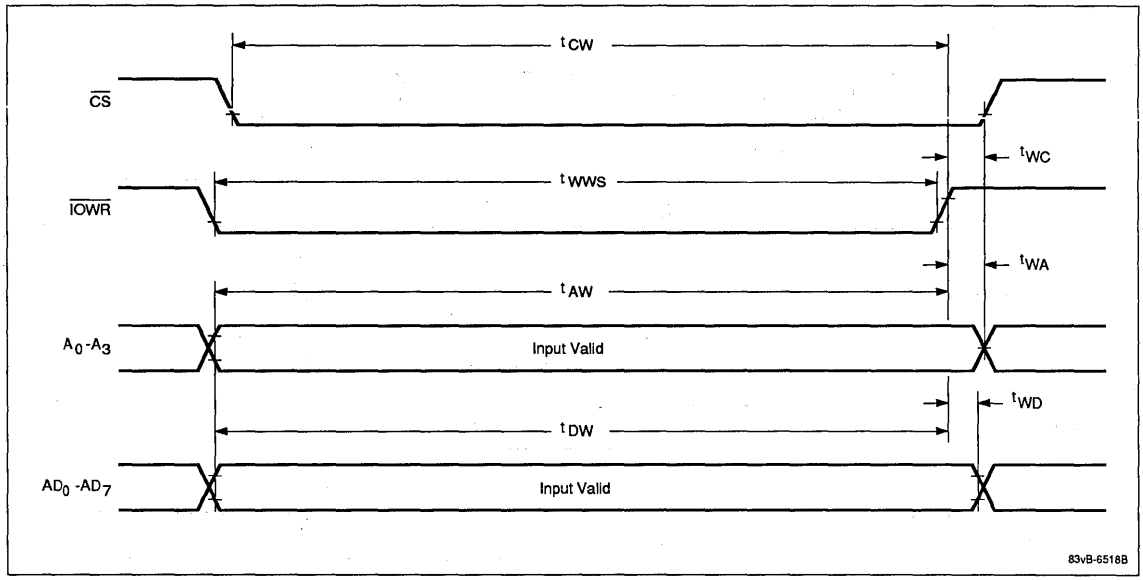


Ready

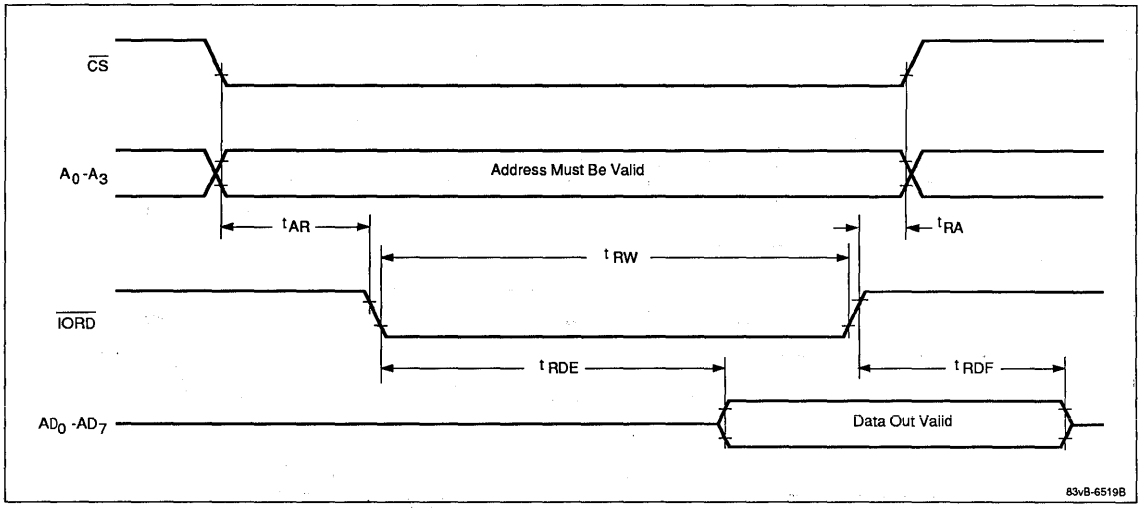


5b

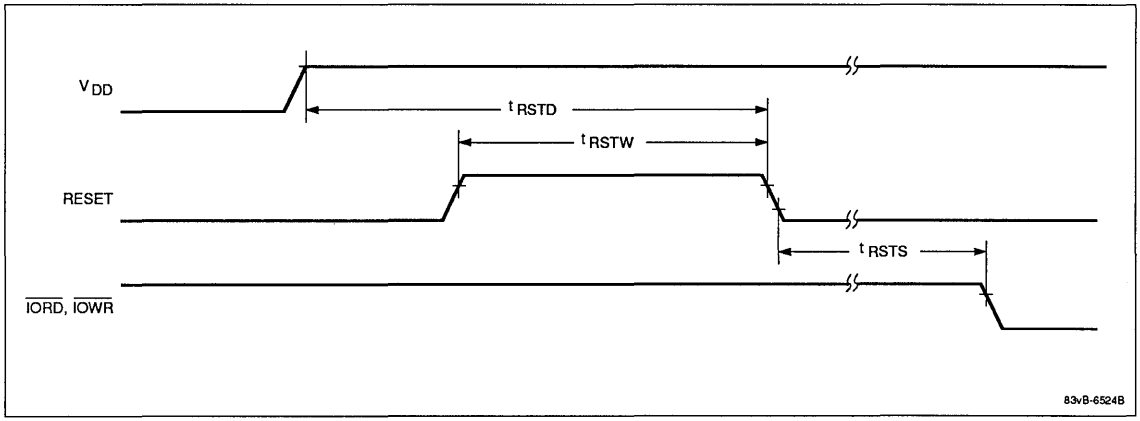
Slave Mode Write



Slave Mode Read



Reset



5b

Description

The μPD71051 serial control unit is a CMOS USART designed to provide serial data communications in microcomputer systems. The CPU uses it as a peripheral I/O device and programs it to communicate in synchronous or asynchronous serial data transmission protocols, including IBM bisync.

The USART receives serial data streams and converts them into parallel data characters for the CPU. While receiving serial data, the USART can also accept parallel data from the CPU, convert it to serial, and transmit the data. The USART signals the CPU when it has received or transmitted a character and requires service. The CPU may read complete USART status data at any time.

Features

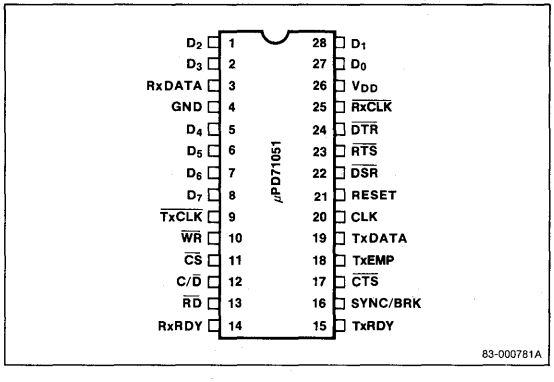
- ☐ Synchronous operation
  - One or two SYNC characters
  - Internal/external synchronization
  - Automatic SYNC character insertion
- ☐ Asynchronous operation
  - Clock rate: (baud rate)
  - x1, x16, or x64
  - Send stop bits: 1, 1.5, or 2 bits
  - Break transmission
  - Automatic break detection
  - Valid start bit detection
- ☐ Baud rate: DC - 240 kbit/s at x1 clock
- ☐ Full duplex, double-buffered transmitter/receiver
- ☐ Error detection: parity, overrun, and framing
- ☐ Five- to eight-bit characters
- ☐ Low-power standby mode
- ☐ Compatible with standard microcomputers
- ☐ Functionally equivalent to (except standby mode) and can replace the μPD8251AF
- ☐ CMOS technology
- ☐ Single +5 V ± 10% power supply
- ☐ Industrial temperature range −40 to +85 °C
- ☐ 28-pin plastic DIP or PLCC or 44-pin plastic QFP
- ☐ 8 MHz and 10 MHz

Ordering Information

Part Number	Clock (MHz)	Package
μPD71051C-8	8	28-pin plastic DIP
C-10	10	
GB-8	8	44-pin plastic QFP
GB-10	10	
L-8	8	28-pin PLCC
L-10	10	

Pin Configurations

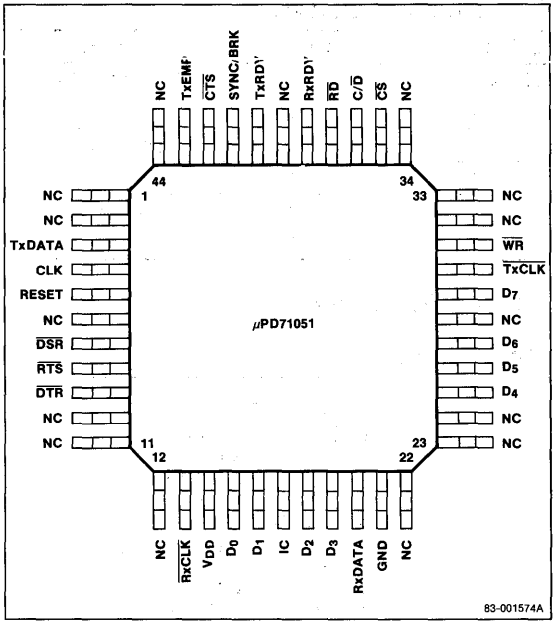
28-Pin Plastic DIP



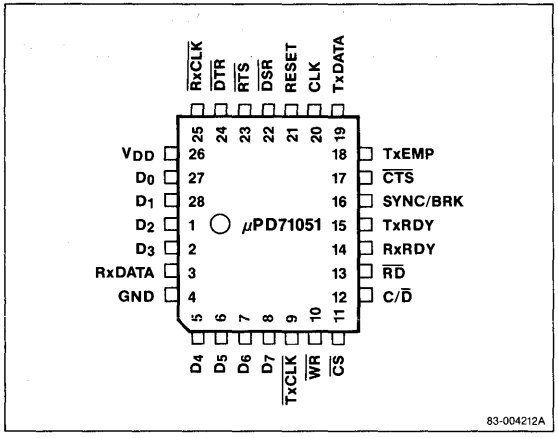
5c

Pin Configurations (cont)

44-Pin Plastic QFP



28-Pin Plastic Leaded Chip Carrier (PLCC)



Pin Identification

Symbol	Function
TxDATA	Transmit data output
CLK	Clock input
RESET	Reset input
$\overline{DSR}$	Data set ready input
$\overline{RTS}$	Request to send output
$\overline{DTR}$	Data terminal ready output
$\overline{RxCLK}$	Receiver clock input
V_{DD}	+5 V power supply
D_7-D_0	Data bus
IC	Internally connected (Do not connect any signal to an IC pin)
RxDATA	Receive data input
GND	Ground
$\overline{TxCLK}$	Transmitter clock input
$\overline{WR}$	Write strobe input
$\overline{CS}$	Chip select input
C/$\overline{D}$	Control or data input
$\overline{RD}$	Read strobe input
RxRDY	Receiver ready output
TxRDY	Transmitter ready output
SYNC/BRK	Synchronization/Break input/output
$\overline{CTS}$	Clear to send input
TxEMP	Transmitter empty output
NC	Not connected

Pin Functions

D_7-D_0 [Data Bus]

D_7-D_0 are an 8-bit, 3-state, bidirectional data bus. The bus transfers data by connecting to the CPU data bus.

RESET [Reset]

A high level to the RESET input resets the μPD71051 and puts it in an idle state. It performs no operations in the idle state. The μPD71051 enters standby mode when this signal falls from a high level to a low level. Standby mode is released when the CPU writes a mode byte to the μPD71051. The reset pulse width must be at least 6 t_{CYK} cycles and the clock must be enabled.

CLK [Clock]

This clock input produces internal timing for the μPD71051. The clock frequency should be at least 30 times the transmitter or receiver clock input frequency ($\overline{TxCLK}$, $\overline{RxCLK}$) in sync or async mode with the X1 clock. This assures stable operation. The clock frequency must be more than 4.5 times the $\overline{TxCLK}$ or $\overline{RxCLK}$ in async mode using x16 or x64 clock mode.

$\overline{CS}$ [Chip Select]

The $\overline{CS}$ input selects the μPD71051. The μPD71051 is selected by setting $\overline{CS} = 0$. When $\overline{CS} = 1$, the μPD71051 is not selected, the data bus (D_7-D_0) is in the high impedance state, and the $\overline{RD}$ and $\overline{WR}$ signals are ignored.

$\overline{RD}$ [Read Strobe]

The $\overline{RD}$ input is low when reading data or status information from the μPD71051.

$\overline{WR}$ [Write Strobe]

The $\overline{WR}$ input is low when writing data or a control byte to the μPD71051.

C/$\overline{D}$ [Control or Data]

The C/$\overline{D}$ input determines the data type when accessing the μPD71051. When C/$\overline{D}$ = 1, the data is a control byte (table 1) or status. When C/$\overline{D}$ = 0, the data is character data. This pin is normally connected to the least significant bit (A_0) of the CPU address bus.

Table 1. Control Signals and Operations

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	C/$\overline{D}$	μPD71051	CPU Operation
0	0	1	0	Receive data buffer ⇨ Data bus	Read receive data
0	0	1	1	Status register ⇨ Data bus	Read status
0	1	0	0	Data bus ⇨ Transmit data buffer	Write transmit data
0	1	0	1	Data bus ⇨ Control byte register	Write control byte
0	1	1	x	Data bus: High impedance	None
1	x	x	x	Data bus: High impedance	None

$\overline{DSR}$ [Data Set Ready]

$\overline{DSR}$ is a general-purpose input pin that can be used for modem control. The status of this pin can be determined by reading bit 7 of the status byte.

$\overline{DTR}$ [Data Terminal Ready]

$\overline{DTR}$ is a general-purpose output pin that can be used for modem control. The state of this pin can be controlled by writing bit 1 of the command byte. If bit 1 = 0, then $\overline{DTR} = 1$. If bit 1 = 1, then $\overline{DTR} = 0$.

$\overline{RTS}$ [Request to Send]

$\overline{RTS}$ is a general-purpose output pin that can be used for modem control. The status of this pin can be controlled by writing bit 5 of the command byte. If bit 5 = 1, then $\overline{RTS} = 0$. If bit 5 = 0, then $\overline{RTS} = 1$.

$\overline{CTS}$ [Clear to Send]

The $\overline{CTS}$ input controls data transmission. The μPD71051 is able to transmit serial data when $\overline{CTS} = 0$ and the command byte sets TxEN = 1. If $\overline{CTS}$ is set equal to 1 during transmission, the sending operation stops after sending all currently written data and the TxDATA pin goes high.

TxDATA [Transmit Data]

The μPD71051 sends serial data over the TxDATA output.

TxRDY [Transmitter Ready]

The TxRDY output tells the CPU that the transmit data buffer in the μPD71051 is empty; that is, that new transmit data can be written. This signal is masked by the TxEN bit of the command byte and by the $\overline{CTS}$ input. It can be used as an interrupt signal to request data from the CPU.

The status of TxRDY can be determined by reading bit 0 of the status byte. This allows the μPD71051 to be polled. Note that TxRDY of the status byte is not masked by $\overline{CTS}$ or TxEN.

TxRDY is cleared to 0 by the falling edge of $\overline{WR}$ when the CPU writes transmit data to the μPD71051. Data in the transmit data buffer that has not been sent is destroyed if transmit data is written while TxRDY = 0.

TxEMP [Transmitter Empty]

The μPD71051 reduces CPU overhead by using a double buffer; the transmit data buffer (second buffer) and the transmit buffer (first buffer) in the transmitter. When the CPU writes transmit data to the transmit data buffer (second buffer), the μPD71051 sends data by transferring the contents of the second buffer to the first buffer, after transmitting the contents of the first buffer.

This empties the second buffer and TxRDY is set to 1. The TxEMP output becomes 1 when the contents of the first buffer are sent and the second buffer is empty. Thus, TxEMP = 1 shows that both buffers are empty. In half-duplex operation, you can determine when to change from sending to receiving by testing TxEMP = 1.

When TxEMP = 1 occurs in async mode, the TxDATA pin goes high. When the CPU writes transmit data, TxEMP is set to 0 and data transmission resumes.

When TxEMP = 1 occurs in sync mode, the μPD71051 loads SYNC characters from the SYNC character register and sends them through the TxDATA pin. TxEMP is set to 0 and resumes sending data after sending (one or two) SYNC characters and the CPU writes new transmit data to the μPD71051.

$\overline{\text{TxCLK}}$ [Transmitter Clock]

The $\overline{\text{TxCLK}}$ input is the reference clock input that determines the transmission rate. Data is transmitted at the same rate as $\overline{\text{TxCLK}}$ in sync mode. In async mode, set $\overline{\text{TxCLK}}$ to 1, 16, or 64 times the transmission rate. Serial data from TxDATA is sent at the falling edge of $\overline{\text{TxCLK}}$.

For example, a rate of 19200 baud in sync mode means that $\overline{\text{TxCLK}}$ is 19.2 kHz. A rate of 2400 baud in async mode can represent a $\overline{\text{TxCLK}}$ of:

x1 clock = 2.4 kHz
x16 clock = 38.4 kHz
x64 clock = 153.6 kHz

RxDATA [Receive Data]

The μPD71051 receives serial data through the RxDATA input.

RxRDY [Receiver Ready]

The RxRDY output becomes 1 when the μPD71051 receives one character of data and transfers that data to the receive data buffer; that is, when the receive data can be read. This signal can be used as an interrupt signal for a data read request to the CPU. You can determine the status of RxRDY by reading bit 1 of the status byte and use the μPD71051 in a polling application. RxRDY becomes 0 when the CPU reads the receive data.

Unless the CPU reads the receive data (after RxRDY = 1 is set) before the next single character is received and transferred to the receive buffer, an overrun error occurs, and the OVE status bit is set. The unread data in the receive data buffer is overwritten by newly transferred data and lost.

RxRDY is set to 0 in the receive disable state. This state is set by changing the RxEN bit to 0 through the command byte. After RxEN is set to 1 (making receiving possible), RxRDY becomes 1 whenever new characters are received and transferred to the receive data buffer.

SYNC/BRK [Synchronization/Break]

The SYNC pin detects synchronization characters in sync mode. The SYNC mode byte selects internal or external SYNC detection. The SYNC pin becomes an output when internal synchronization is set, and an input when external synchronization is set.

The SYNC output goes high when the μPD71051 detects a SYNC character in internal synchronization. When two SYNC characters are used, SYNC goes high when the last bit of the two consecutive SYNC characters is detected. You can read the status of the SYNC signal in bit 6 of the status byte. Both the SYNC pin and status are set to 0 by a read status operation.

In external synchronization, in order for the external circuit to detect synchronization, a high level of at least one period of $\overline{\text{RxCLK}}$ must be input to the SYNC pin. When the μPD71051 detects the high level, it begins to receive data, starting at the rising edge of the next $\overline{\text{RxCLK}}$. The high level input may be removed when synchronization is released.

The BRK output is used only in async mode and shows the detection of a break state. BRK goes high when a low level signal is input to the RxDATA pin for two character bit lengths (including the start, stop, and parity bits). As with SYNC, you can read the status of BRK in bit 6 of the status byte. BRK is not cleared by the read operation.

The set BRK signal is cleared when the RxDATA pin returns to high level, or when the μPD71051 is reset by hardware or software. The SYNC/BRK pin goes low on reset, regardless of previous mode. Figure 1 shows the break state and BRK signal.

$\overline{\text{RxCLK}}$ [Receiver Clock]

$\overline{\text{RxCLK}}$ is a reference clock input that controls the receive data rate. In sync mode, the receiving rate is the same as $\overline{\text{RxCLK}}$. In async mode, $\overline{\text{RxCLK}}$ can be 1, 16, or 64 times the receive rate. Serial data from RxDATA is input by the rising edge of $\overline{\text{RxCLK}}$.

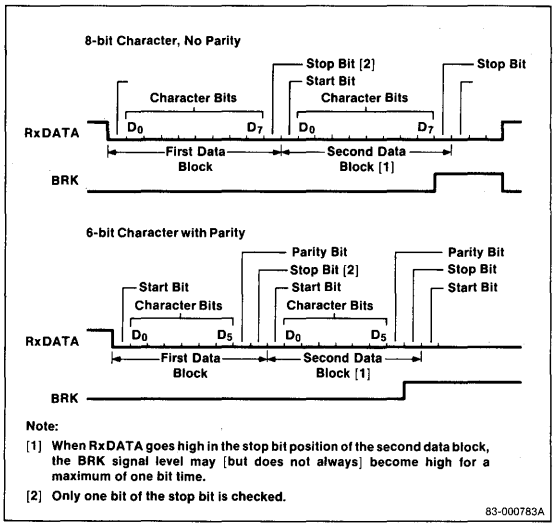
V_{DD} [Power]

+5 V power supply.

GND [Ground]

Ground.

Figure 1. Break Status and Break Signal



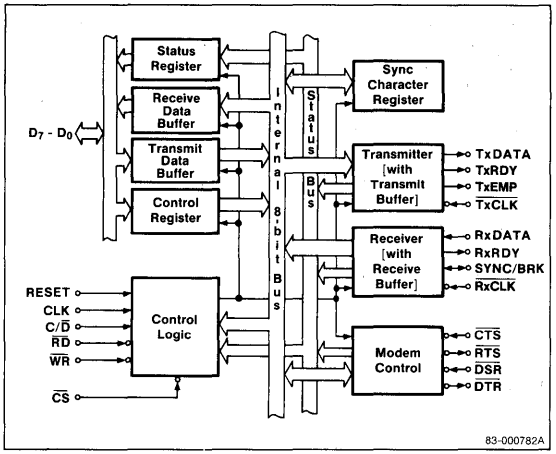
μPD71051 Functions

The μPD71051 is a CMOS serial control (USART) unit that provides serial communications in microcomputer systems. The CPU handles the μPD71051 as an ordinary I/O device.

The μPD71051 can operate in synchronous or asynchronous systems. In sync mode, the character bit length, number of sync characters, and sync detection mode must be designated. In async mode, the communication rate, character bit length, stop bit length, etc., must be designated. The parity bit may be designated in either mode.

The μPD71051 converts parallel data received from the CPU into serial transmitted data (from the TxDATA pin), and converts serial input data (from the RxDATA pin) into parallel data so that the CPU can read it (receiving operation).

Block Diagram



The CPU can read the current status of the μPD71051 and can process data after checking the status, after checking for transfer errors, and μPD71051 data buffer status.

The μPD71051 can be reset under hardware or software control to a standby mode that consumes less power and removes the device from system operation. In this mode, the μPD71051's previous operating mode is released and it waits for a mode byte to set the mode. The μPD71051 leaves standby mode and shifts to a designated operating mode when the CPU writes a mode byte to it.

Status Register

The status register allows the CPU to read the status of the μPD71051 except in standby mode. This register indicates status and allows the CPU to manage data reading, writing, and error handling during operations.

Receive Data Buffer

When the receiver has converted the serial data input from the RxDATA pin into parallel data, the converted data is stored in the receive data buffer. The CPU can then read it. Data for one character entering the receive buffer is transferred to the receive data buffer and RxRDY becomes 1, requesting that the CPU read the data.

Transmit Data Buffer

The transmit data buffer holds the parallel data from the CPU that the transmitter will convert to serial data and output from the TxDATA pin. When the CPU writes transmit data to the μPD71051, the μPD71051 stores data in the transmit data buffer. The transmit data buffer transfers the data to the transmitter, which sends the data from the TxDATA pin.

Control Register

This register stores the mode and the command bytes.

Control Logic

The control logic sends control signals to the internal blocks and controls the operation of the μPD71051 based on internal and external signals.

Synchronous Character Register

This register stores one or two SYNC characters used in sync mode. During transmission, the SYNC characters stored in this register are output from the TxDATA pin when the CPU does not send a new character and TxEMP status is set. During receiving, synchronization is established when the characters received and the SYNC characters stored in this register are the same.

Transmitter

The contents of the transmit data buffer are transferred to the transmitter, converted from parallel to serial, and output from the TxDATA pin. The transmitter adds start, stop, and parity bits.

Receiver

The receiver converts serial data input from the RxDATA pin into parallel data and transfers the parallel data to the receive data buffer, allowing the CPU to read it.

The receiver detects SYNC characters and checks parity bits in sync mode. It detects the start and stop bits, and checks parity in the async mode.

In async mode, receiving does not begin (the start bit is not detected) until one effective stop bit (high level) is input to the RxDATA pin and Receive Enable (RxEN = 1) is set after setting up the mode.

Modem Control

This block controls the $\overline{CTS}$, $\overline{RTS}$, $\overline{DSR}$, and $\overline{DTR}$ modem interface pins. The $\overline{RTS}$, $\overline{DSR}$, and $\overline{DTR}$ pins can also be used as general-purpose I/O pins.

Absolute Maximum Ratings

$T_A = +25°C$

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.3 V
Output voltage, V_O	−0.5 to V_{DD} + 0.3 V
Operating temperature, T_{OPT}	−40°C to +85°C
Storage temperature, T_{STG}	−65°C to +150°C
Power dissipation, PD_{MAX}	1.0 W

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Capacitance

$T_A = +25°C$, $V_{DD} = 0$ V

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
Input capacitance	C_I		10	pF	fc = 1MHz Unmeasured pins returned to 0 V
I/O capacitance	C_{IO}		20	pF	

DC Characteristics

$T_A = -40°C$ to $+85°C$, $V_{DD} = +5$ V ± 10%

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Input voltage high	V_{IH}	2.2		V_{DD}+0.3	V	
Input voltage low	V_{IL}	−0.5		0.8	V	
Output voltage high	V_{OH}	0.7 x V_{DD}			V	$I_{OH} = -400$ μA
Output voltage low	V_{OL}			0.4	V	$I_{OL} = 2.5$ mA
Input leakage current high	I_{LIH}			10	μA	$V_I = V_{DD}$
Input leakage current low	I_{LIL}			−10	μA	$V_I = 0$ V
Output leakage current high	I_{LOH}			10	μA	$V_O = V_{DD}$
Output leakage current low	I_{LOL}			−10	μA	$V_O = 0$ V
Supply current μPD71051	I_{DD1}			10	mA	Normal mode
	I_{DD2}		50	100	μA	Stand-by mode
μPD71051-10	I_{DD1}			10	mA	Normal mode
	I_{DD2}		2	50	μA	Stand-by mode

AC Characteristics

$T_A = -40°C$ to $+85°C$, $V_{DD} = 5$ V ±10%

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
Read Cycle							
Address setup to $\overline{RD}$ ↓	t_{SAR}	0		0		ns	$\overline{CS}$, C/$\overline{D}$
Address hold from $\overline{RD}$ ↑	t_{HRA}	0		0		ns	$\overline{CS}$, C/$\overline{D}$
$\overline{RD}$ low level width	t_{RRL}	150		95		ns	
Data delay from $\overline{RD}$ ↓	t_{DRD}		120		85	ns	$C_L = 150$ pF
Data float from $\overline{RD}$ ↑	t_{FRD}	10	80	10	65	ns	
Port ($\overline{DSR}$, $\overline{CTS}$) set-up to $\overline{RD}$ ↓	t_{SPR}	20		20		t_{CYK}	
Write Cycle							
Address setup to $\overline{WR}$ ↓	t_{SAW}	0		0		ns	$\overline{CS}$, C/$\overline{D}$
Address hold from $\overline{WR}$ ↑	t_{HWA}	0		0		ns	$\overline{CS}$, C/$\overline{D}$
$\overline{WR}$ low level width	t_{WWL}	150		95		ns	
Data setup to $\overline{WR}$ ↑	t_{SDW}	80		80		ns	
Data hold from $\overline{WR}$ ↑	t_{HWD}	0		0		t_{CYK}	
Port ($\overline{DTR}$, $\overline{RTS}$), delay from $\overline{WR}$ ↑	t_{DWP}		8		8	t_{CYK}	
Write recovery time	t_{RV}	6		6		t_{CYK}	Mode initialize
		8		8		t_{CYK}	Async mode
		16		16		t_{CYK}	Sync mode
Serial Transfer Timing							
CLK cycle time	t_{CYK}	125	DC	100	DC	ns	
CLK high level width	t_{KKH}	50		35		ns	
CLK low level width	t_{KKL}	35		25		ns	
CLK rise time	t_{KR}	5	20	5	20	ns	
CLK fall time	t_{KF}	5	20	5	20	ns	
TxDATA delay from $\overline{TxCLK}$	t_{DTKTD}		0.5		0.5	μs	
Transmitter input clock pulse width low level	t_{TKTKL}	12		12		t_{CYK}	1xBR (Note 1)
		1		1		t_{CYK}	16x, 64xBR
Transmitter input clock pulse width high level	t_{TKTKH}	15		15		t_{CYK}	1xBR
		3		3		t_{CYK}	16x, 64xBR
Transmitter input clock frequency	f_{TK} (Note 2)	DC	240	DC	300	kHZ	1xBR
		DC	1536	DC	1920	kHz	16xBR
		DC	1536	DC	1920	kHz	64xBR
Receiver input clock pulse width low level	t_{RKRKL}	12		12		t_{CYK}	1xBR
		1		1		t_{CYK}	16x, 64xBR
Receiver input clock pulse width high level	t_{RKRKH}	15		15		t_{CYK}	1xBR
		3		3		t_{CYK}	16x, 64xBR

5c

AC Characteristics (cont)

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
Serial Transfer Timing (cont)							
Receiver input clock frequency	f_{RK} (Note 2)	DC	240	DC	300	kHz	1xBR
		DC	1536	DC	1920	kHz	16xBR
		DC	1536	DC	1920	kHz	64xBR
RxDATA set-up to Sampling pulse	t_{SRDSP}	1		1		μs	
RxDATA hold from sampling pulse	t_{HSPRD}	1		1		μs	
TxEMP delay time (TxDATA)	t_{DTXEP}		20		20	t_{CYK}	
TxRDY delay time (TxRDY↑)	t_{DTXR}		8		8	t_{CYK}	
TxRDY delay time (TxRDY↓)	t_{DWTXR}		200		100	ns	
RxRDY delay time (RxRDY↑)	t_{DRXR}		26		26	t_{CYK}	
RxRDY delay time (RxRDY↓)	t_{DRRXR}		200		100	ns	
SYNC output delay time (for internal sync)	t_{DRKSY}		26		26	t_{CYK}	
SYNC input set-up time (for external sync)	t_{SSYRK}	18		18		t_{CYK}	
RESET pulse width		6		6		t_{CYK}	

Notes:

(1) BR = Baud rate

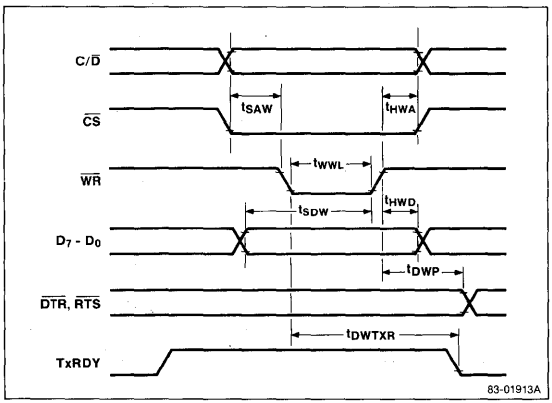
(2) 1xBR: f_{TK} or $f_{RK} \leq 1/30\, t_{CLK}$, 16x, 64xBR: f_{TK} or $f_{RK} \leq 1/4.5\, t_{CLK}$

(3) System CLK is needed during reset operation

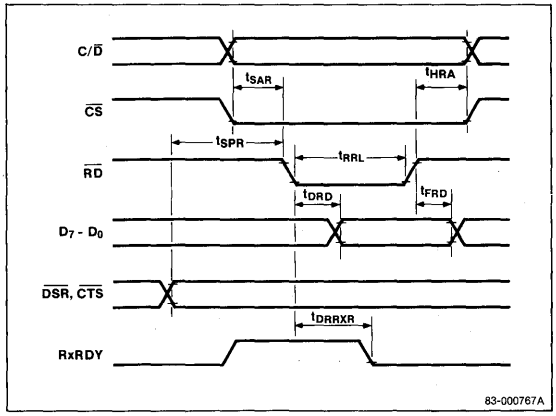
(4) Status update can have a maximum delay of 28 t_{CYK} from the event effecting the status.

Timing Waveforms

Write Data Cycle

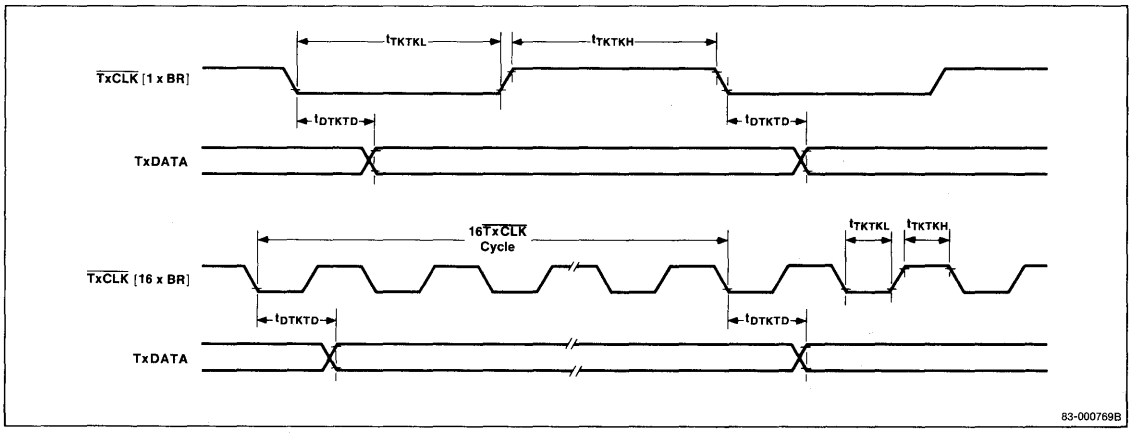


Read Data Cycle

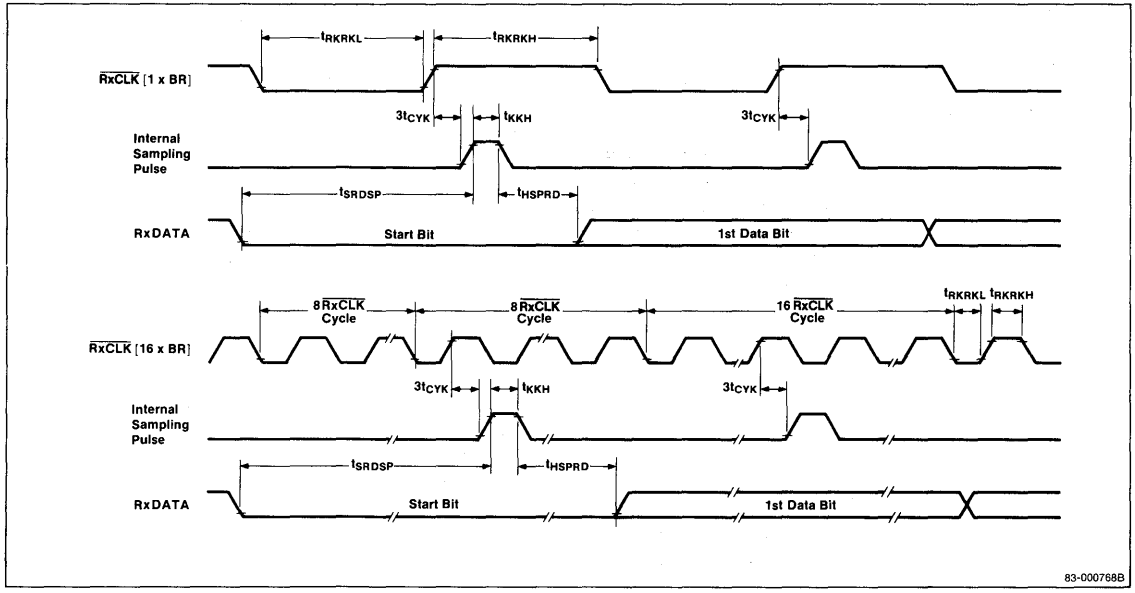


Timing Waveforms (cont)

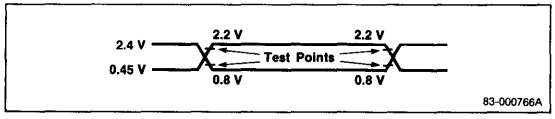
Transmitter Clock and TxDATA



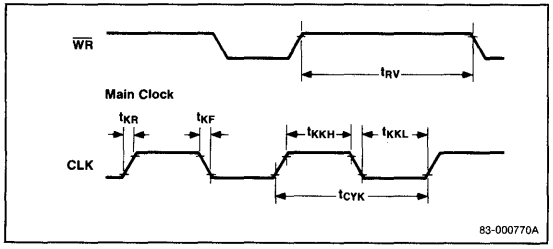
Receiver Clock and RxDATA Timing



AC Test Input



Write Recovery Time



5c

Connecting the μPD71051 to the System

The CPU uses the μPD71051 as an I/O device by allocating two I/O addresses, set by the value of C/$\overline{D}$. One I/O address is allocated when the level of C/$\overline{D}$ is low and becomes a port to the transmit and receive data register. The other I/O address is allocated when C/$\overline{D}$ is high and becomes a port to the mode, command, and status registers. Generally, the least significant bit (A_0) of the CPU address bus is connected to C/$\overline{D}$ to get a continuous I/O address. This is shown in figure 2.

Pins TxRDY and RxRDY are connected to the CPU or, when interrupts are used, to the interrupt pin of the interrupt controller.

Operating the μPD71051

Start with a hardware reset (set the RESET pin high) after powering on the μPD71051. This puts the μPD71051 into standby mode and it waits for a mode byte. In async mode, the μPD71051 is ready for a command byte after the mode byte; the mode byte sets the communication protocol to the async mode. In sync mode, the μPD71051 waits for one or two SYNC characters to be sent after the mode byte; set C/$\overline{D}$ = 1. A command byte may be sent after the SYNC characters are written. Figure 3 shows this operation sequence.

In both modes, it is possible to write transmit data, read receive data, read status, and write more command bytes after the first command byte is written. The μPD71051 performs a reset, enters standby mode, and returns to a state where it waits for a mode byte when the command byte performs a software reset.

Figure 2. System Connection

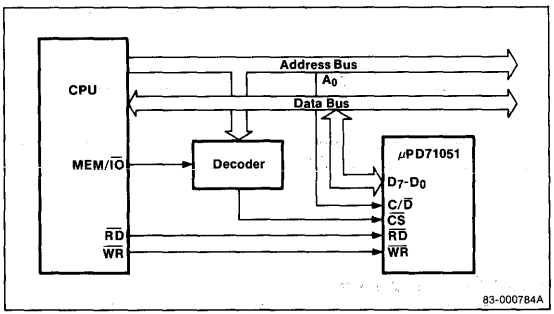
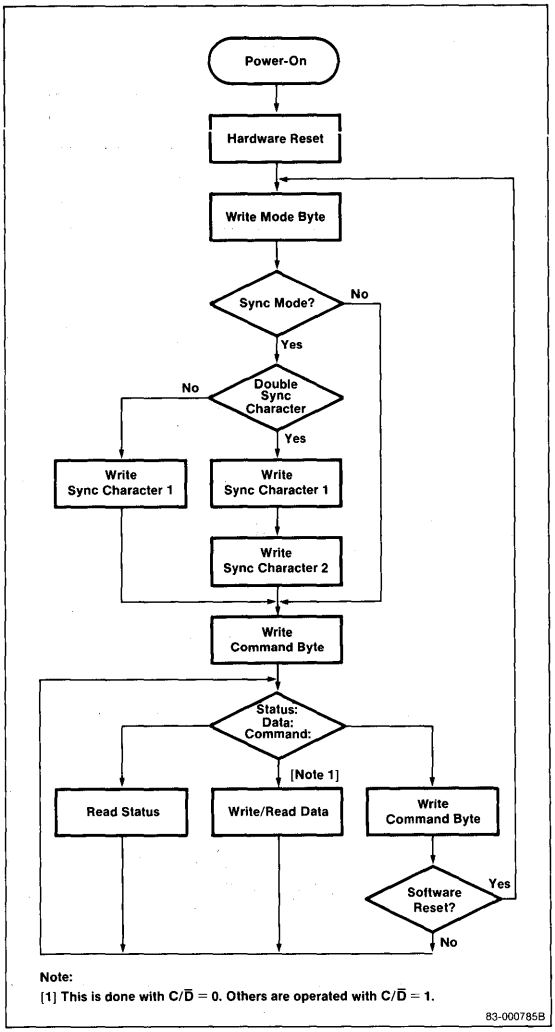


Figure 3. μPD71051 Operating Procedure



Mode Register

When the μPD71051 is in standby mode, writing a mode byte to it will release standby mode. Figure 4 shows the mode byte format for designating async mode. Figure 5 shows the mode byte format for designating sync mode. Bits 0 and 1 must be 00 to designate sync mode. Async mode is designated by all other combinations of bits 0 and 1.

The P1, P0 and L1, L0 bits are common to both modes. Bits P1 and P0 (parity) control the generation and checking (sending and receiving) functions. These parity bit functions do not operate when P0 = 0. When P1, P0 = 01, the μPD71051 generates and checks odd parity. When P1, P0 = 11, it generates and checks even parity.

Bits L1 and L0 set the number of bits per character (n). Additional bits such as parity bits are not included in this number. Given n bits, the μPD71051 receives the lower n bits of the 8-bit data written by the CPU. The upper bits (8 -n) of data that the CPU reads from the μPD71051 are set to zero.

The ST1, ST0 and B1, B0 bits are used in async mode. The ST1 and ST0 bits determine the number of stop bits added by the μPD71051 during transmission.

The B1 and B0 bits determine the relationship between the baud rates for sending and receiving, and the clocks $\overline{TxCLK}$ and $\overline{RxCLK}$. B1 and B0 select a multiplication rate of 1, 16, or 64 for the frequency of the sending and receiving clock relative to the baud rate. Multiplication by 1 is not normally used in async mode. Note that the data and clock must be synchronized on the sending and receiving sides when multiplication by 1 is used.

The SSC and EXSYNC bits are used in sync mode. The SSC bit determines the number of SYNC characters. SSC = 1 designates one SYNC character. SSC = 0 designates two SYNC characters. The number of SYNC characters determined by the SSC bit are written to the μPD71051 immediately after writing the mode byte.

The EXSYNC bit determines whether sync detection during receiving operations is internal or external. EXSYNC = 1 selects external sync detection and EXSYNC = 0 selects internal sync detection.

Figure 4. Mode Byte for Setting Asynchronous Mode

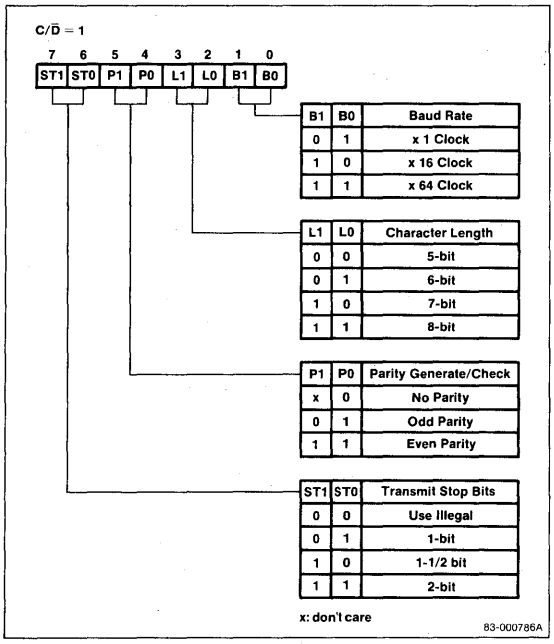
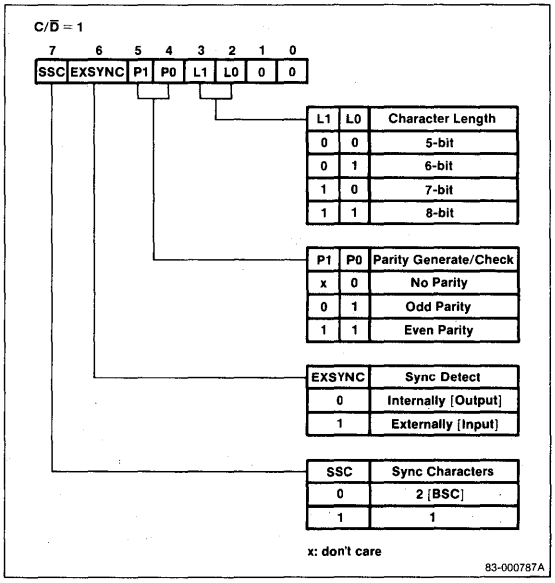


Figure 5. Mode byte for Setting Synchronous Mode



5C

Command Register

Commands are issued to the μPD71051 by the CPU by command bytes that control the sending and receiving operations of the μPD71051. A command byte is sent after the mode byte (in sync mode, a command byte may only be sent after writing SYNC characters) and the CPU must set C/$\overline{D}$ = 1. Figure 6 shows the command byte format.

Bit EH is set to 1 when entering hunt phase to synchronize in sync mode. Bit RxEN should also be set to 1 at that time. Data reception begins when SYNC characters are detected and synchronization is achieved, thus releasing hunt phase.

When bit SRES is set to 1, a software reset is executed, and the μPD71051 goes into standby mode and waits for a mode byte.

Bit RTS controls the $\overline{RTS}$ output pin. $\overline{RTS}$ is low when the RTS bit = 1 , and goes high when RTS = 0.

Setting bit ECL to 1 clears the error flags (PE, OVE, and FE) in the status register. Set ECL to 1 when entering the hunt phase or enabling the receiver.

Bit SBRK sends a break. When SBRK = 1, the data currently being sent is destroyed and the TxDATA pin goes low. Set SBRK = 0 to release a break. Break also works when TxEN = 0 (send disable).

Bit RxEN enables and disables the receiver. RxEN = 1 enables the receiver and RxEN = 0 disables the receiver. Synchronization is lost if RxEN = 0 during sync mode.

Bit DTR controls the $\overline{DTR}$ output pin. $\overline{DTR}$ goes low when the DTR bit = 1 and goes high when the DTR bit = 0.

The TxEN bit enables and disables the transmitter. TxEN = 1 enables the transmitter and TxEN = 0 disables the transmitter. When TxEN = 0, sending stops and the TxDATA pin goes high (mark status) after all the currently written data is sent.

Status Register

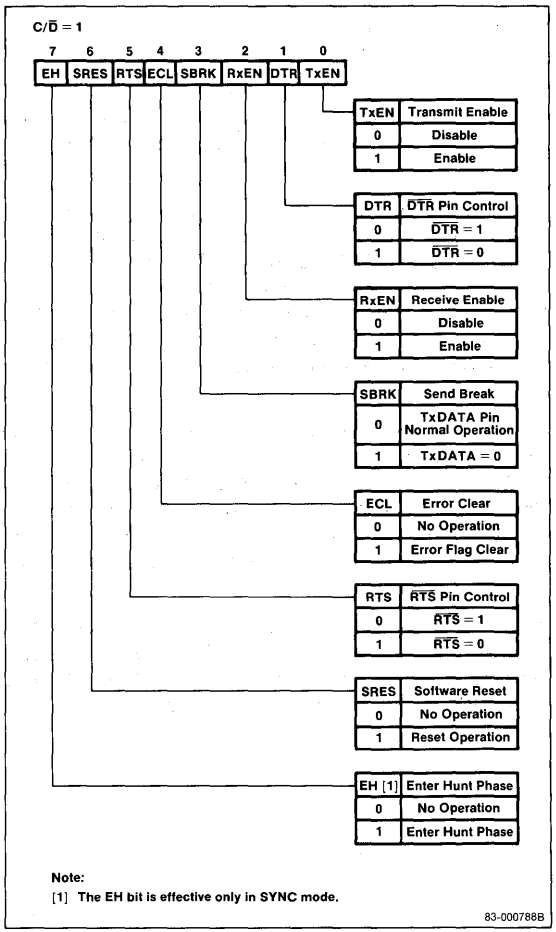
The CPU can read the status of the μPD71051 at any time except when the μPD71051 is in standby mode. Status can be read after setting C/$\overline{D}$ = 1 and $\overline{RD}$ = 0. Status is not updated while being read. Status updating is delayed at least 28 clock periods after an event that affects the status. Figure 7 shows the format of the status register.

The TxEMP and RxRDY bits have the same meaning as the pins of the same name. The SYNC/BRK bit generally has the same meaning as the SYNC/BRK pin. In external synchronization mode, the status of this bit does not always coincide with the pin. In this case, the SYNC pin becomes an input and the status bit goes to 1 when a rising edge is detected at the input. The status bit remains at 1 until it is read, even when the input level at the SYNC pin goes low. The status bit becomes 1 when a SYNC character is input with the RxDATA input, even when the pin is at a low level.

The DSR bit shows the status of the $\overline{DSR}$ input pin. The status bit is 1 when the $\overline{DSR}$ pin is low.

The FE bit (framing error) becomes 1 when less than one stop bit is detected at the end of each data block during asynchronous receiving. Figure 8 shows how a framing error can happen.

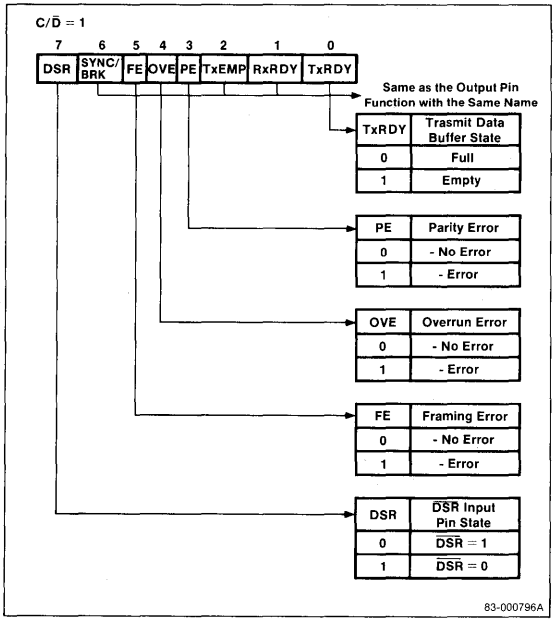
Figure 6. Command Byte Format



The OVE bit (overrun error) becomes 1 when the CPU delays reading the received data and two new data bytes have been received. In this case, the first data byte received is overwritten and lost in the receive data buffer. Figure 9 shows how an overrun can happen.

The PE bit (parity error) becomes 1 when a parity error occurs in a receive state.

Figure 7. Status Register Format



Framing, overrun, and parity errors do not disable the μPD71051's operations. All three error flags are cleared to 0 by a command byte that sets the ECL bit to 1.

The TxRDY bit becomes 1 when the transmit data buffer is empty. The TxRDY output pin becomes 1 when the transmit data buffer is empty, the $\overline{CTS}$ pin is low, and TxEN = 1. That is, bit TxRDY = Transmit Data Buffer Empty, pin TxRDY = (Transmit Data Buffer Empty)•($\overline{CTS}$ = 0)•(TxEN = 1).

Figure 8. Framing Error

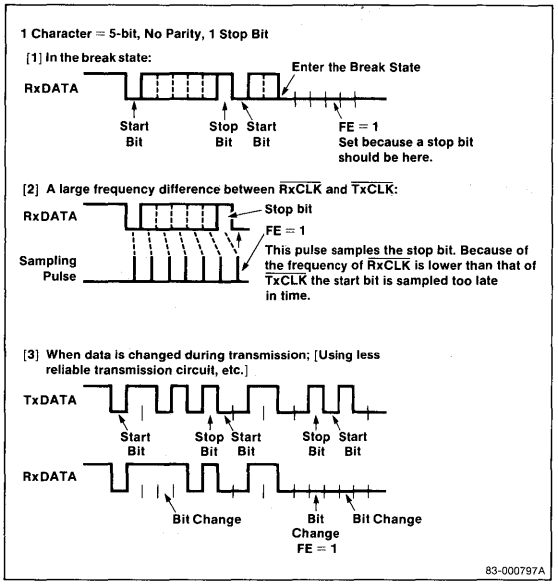
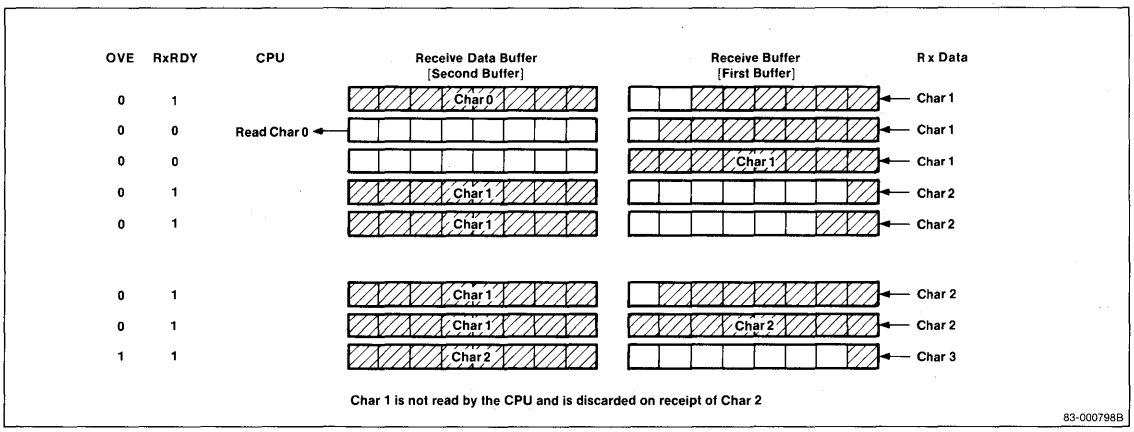


Figure 9. Overrun Error



5c

Sending in Asynchronous Mode

The TxDATA pin is typically in the high state (marking) when data is not being sent. When the CPU writes transmit data to the μPD71051, the μPD71051 transfers the transmit data from the transmit data buffer to the send buffer and sends the data from the TxDATA pin after adding one start bit (low level) and a programmed stop bit. If parity is used, a parity bit is inserted between the character and the stop bit. Figure 10 shows the data format for async mode characters. Serial data is sent by the falling edge of the signal that divided $\overline{TxCLK}$ (1/1, 1/16, or 1/64).

When bit SBRK is set to 1, the TxDATA pin goes low (break status), regardless of whether data is being sent. Figure 11 is a fragment of a typical program to send data in the async mode. Figure 12 shows the output from pin TxDATA.

Figure 10. Asynchronous Mode Data Format

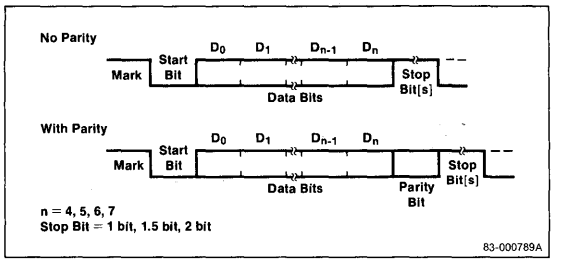
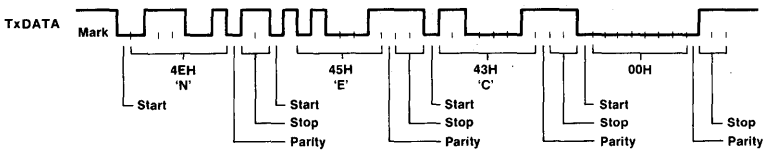


Figure 11. Asynchronous Transmitter Example

```
ASYNTX :    CALL    ASYNMOD              ;Set async mode
            MOV     AL, 00010001B        ;Command: clear error flag, transmit enable
            OUT     PCTRL,AL
            MOV     BW, OFFSET TXDADR    ;Transmit data area
TXSTART :   IN      AL, PCTRL
            TEST1   AL, 0                ;Read status
            BNE     TXSTART              ;Wait until TxRDY = 1
            MOV     AL, [BW]             ;Write transmit data
            OUT     PDATA, AL
            INC     BW                   ;Set next data address
            CMP     AL, 00H
            BNE     TXSTART              ;End if data = 0
            RET
TXDADR      DB      'NEC'                ;Transmit data 4EH, 45H, 43H, 00
            DB      0
ASYNMOD :   MOV     AL, 0                ;Writes control bytes three times
            OUT     PCTRL, AL            ;with 00H to unconditionally
            OUT     PCTRL, AL            ;accept the new command byte
            OUT     PCTRL, AL
            MOV     AL, 01000000B        ;Software reset
            OUT     PCTRL, AL
            MOV     AL, 11111010B        ;Write mode byte
            OUT     PCTRL, AL            ;Stop bit = 2 bits, even parity
            RET                          ;7 bits/character, x16 clock
```

Figure 12. TxDATA Pin Output



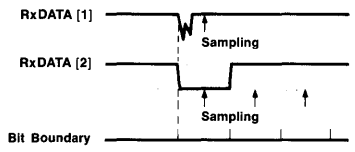
Receiving in Asynchronous Mode

The RxDATA pin is normally in the high state when data is not being received, as shown in figure 13. The μPD71051 detects the falling edge of a low level signal when a low level signal enters it.

The μPD71051 samples the level of the RxDATA input (only when x16 or x64 clock is selected) in a position 1/2 bit time after the falling edge of the RxDATA input to check whether this low level is a valid start bit. It is considered a valid start bit if a low level is detected at that time. If a low level is not detected, it is not regarded as a start bit and the μPD71051 continues testing for a valid start bit.

When a start bit is detected, the sampling points of the data bits, parity bit (when used), and stop bit are decided by a bit counter. The sampling is performed by the rising edge of the $\overline{\text{RxCLK}}$ when an X1 clock is used. When a x16 or x64 clock is used, it is sampled at the nominal middle of $\overline{\text{RxCLK}}$.

Figure 13. Start Bit Detection



Note:
[1] Start bit is not recognized because R x Data is high at the sampling time.
[2] Start is recognized because R x Data is low at the sampling time.

83-000791A

Data for one character entering the receive buffer is transferred to the receive data buffer and causes RxRDY = 1, requesting that the CPU read the data. When the CPU reads the data, RxRDY becomes 0.

When a valid stop bit is detected, the μPD71051 waits for the start bit of the next data. If a low level is detected in the stop bit, a framing error flag is set; however, the receiving operation continues as if the correct high level had been detected. A parity error flag is set if a parity error is detected. An overrun error flag is set when the CPU does not read the data in time, and the next receiving data is transferred to the receive data buffer, overwriting the unread data. The μPD71051's sending and receiving operations are not affected by these errors.

If a low level is input to the RxDATA pin for more than two data blocks during a receive operation, the μPD71051 considers it a break state and the SYNC/BRK pin status becomes 1.

In async mode, the start bit is not detected until a high level of more than one bit is input to the RxDATA pin and the receiver is enabled. Figure 14 is a fragment of a typical program to receive the data sent in the previous async transmit example.

5c

Figure 14. Asynchronous Receiver Example

```
ASYNRX :    CALL    ASYNMOD             ;Set ASYNC mode
            MOV     AL, 00010100B       ;Command: clear error flag, receive
                                        ;enable
            OUT     PCTRL,AL
            MOV     BW, OFFSET RXDADR   ;Data store area
RXSTART :   IN      AL, PCTRL
            TEST1   AL, 1               ;Read status
            BNE     RXSTART             ;Wait until RxRDY = 1
            IN      AL, PDATA           ;Read and store the receive data
            MOV     [BW], AL
            INC     BW                  ;Set next store address
            CMP     AL, 00H             ;End if data = 0
            BNE     RXSTART
            RET
RXDADR      DB      256 DUP             ;Reserve receive data area
```

Sending in Synchronous Mode

Following the establishment of sync mode and the enabling of the transmitter, the TxDATA pin stays high until the CPU writes the first character (normally, SYNC characters). When data is written, the TxDATA pin sends one bit for each falling edge of TxCLK if the $\overline{CTS}$ pin is low. Unlike async mode, start and stop bits are not used. However, a parity bit may be set. Figure 15 shows these data formats.

Once sending begins, the CPU must write data to the μPD71051 at the same rate as that of TxCLK. If TxEMP goes to 1 because of a delay in writing by the CPU, the μPD71051 sends SYNC characters until the CPU writes data. TxEMP goes to 0 when data is written, and the data is sent as soon as transmission of SYNC characters stops.

Figure 15. Synchronous Mode Data Format

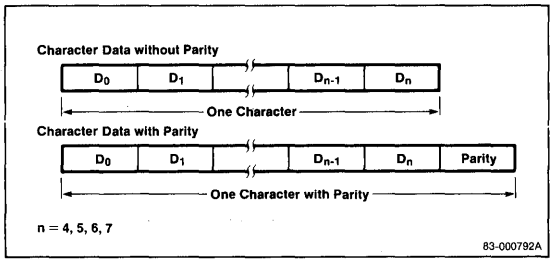


Figure 16. Synchronous Mode Transmit Timing

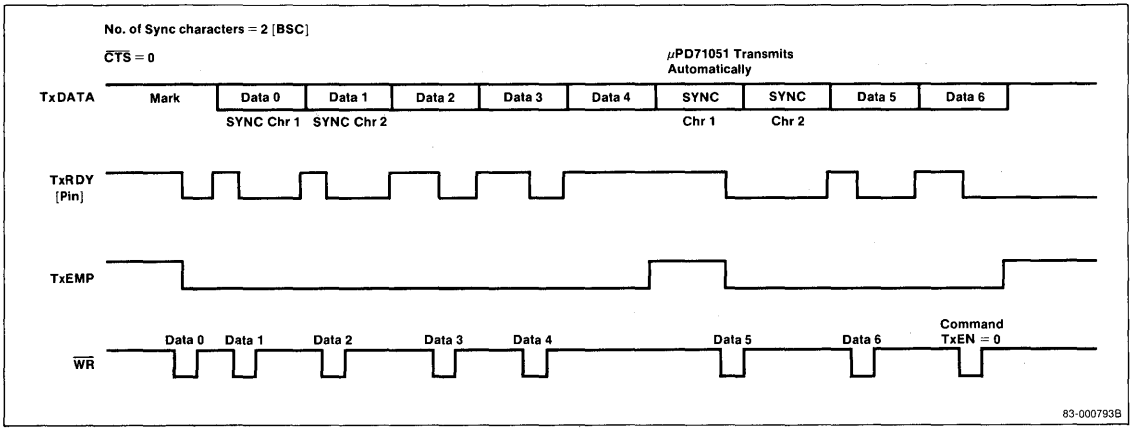
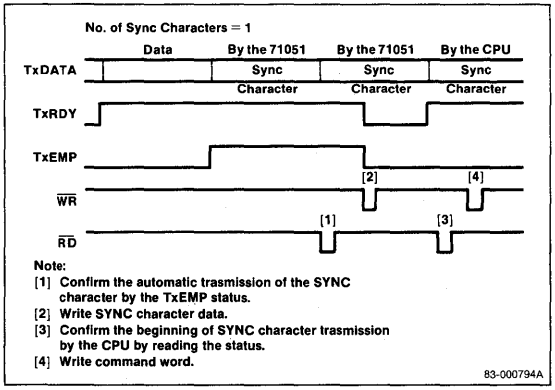


Figure 17. Issuing a Command During SYNC Character Transmission



Automatic transmission of SYNC characters begins after the CPU sends new data. SYNC characters are not automatically sent by enabling the transmitter. Figure 16 shows these timing sequences.

If a command is sent to the μPD71051 while SYNC characters are automatically being sent and TxEMP = 1, the μPD71051 may interpret the command as a data byte and transmit it as data. If a command must be sent under these conditions, the CPU should send a SYNC character to the μPD71051 and send the command while the SYNC character is being transmitted. This is shown in figure 17.

Figure 18 is a fragment of a typical program for sending in sync mode.

Figure 18. Synchronous Transmitter Example

```
SYNTX :   CALL    SYNMOD              ;Set sync mode
          MOV     AL, 00010001B       ;Command; clear error
          OUT     PCTRL, AL           ;flags, transmit enable
          MOV     BW, OFFSET TXDADR   ;Start location of data area TxDADR
          MOV     CL, LDLEN           ;Set number of bytes (LDLEN) to be transmitted
          MOV     CH, 00H
TXLEN :   IN      AL, PCTRL           ;Transmit the length byte
          TEST1   AL, 0
          BZ      TXLEN
          MOV     AL, LDLEN
          OUT     PDATA, AL
TXDATA :  IN      AL, PCTRL
          TEST1   AL, 0
          BZ      TXDATA              ;Transmit the number of
          MOV     AL, (BW)            ;bytes specified by LDLEN
          OUT     PDATA, AL
          INC     BW
          DBNZ    TXDATA
          MOV     AL, 00010000B       ;Command; clear error
          OUT     PCTRL, AL           ;flags, transmit disable
          RET
SYNC1     DB      ?                   ;SYNC character 1
SYNC2     DB      ?                   ;SYNC character 2
LDLEN     DB      ?                   ;transmit data count
TXDADR    DB      255 DUP (?)         ;transmit data
SYNMOD :  MOV     AL, 00H
          OUT     PCTRL, AL           ;Write control bytes
          OUT     PCTRL, AL           ;three times with 00H to
          OUT     PCTRL, AL           ;unconditionally accept the new
                                      ;command byte
          MOV     AL, 01000000B       ;Software reset
          OUT     PCTRL, AL
          MOV     AL, 00111100B       ;Write mode byte: 2 SYNC
          OUT     PCTRL, AL           ;characters, internal sync detect,
                                      ;even parity, 8 bits/character
          MOV     AL, SYNC1
          OUT     PCTRL, AL           ;Write SYNC characters
          MOV     AL, SYNC2
          OUT     PCTRL, AL
          RET
```

5c

Receiving in Synchronous Mode

In order to receive in sync mode, synchronization must be established with the sending side. The first command after setting sync mode and writing the SYNC character must be EH = 1, ECL = 1, and RxEN = 1. When hunt phase is entered all the bits in the receive buffer are set to 1. In internal synchronization, data on the RxDATA pin is input to the receive buffer for each rising edge of $\overline{RxCLK}$ and is compared with the SYNC character at the same time. Figure 19 shows this internal sync detection.

When the receive buffer and the SYNC character coincide, and parity is not used, the μPD71051 ends hunt phase and SYNC is set to 1 in the center of the last SYNC bit. When parity is used, SYNC becomes 1 in the center of the parity bit. Receiving starts with the bit which follows the bit when SYNC is set to 1.

In external sync detection, synchronization is achieved by setting the SYNC pin high from an external circuit for at least one period of $\overline{RxCLK}$. Hunt phase ends, and data reception can start. At this time, the SYNC status bit becomes 1, and goes to 0 when the status is read. The SYNC status bit is set to 1 when the SYNC input has a rising edge followed by a high level of more than one period of $\overline{RxCLK}$, even after synchronization is achieved.

The μPD71051 can regain lost synchronization anytime by issuing an enter hunt phase command.

After synchronization, the SYNC character is compared with each character regardless of whether internal or external synchronization is used. When the characters coincide, SYNC becomes 1, indicating that a SYNC character has been received. SYNC (SYNC status bit only in external detection) becomes 0 when the status is read.

Overrun and parity errors are checked the same way as in async mode, affecting only the status flag. Parity checking is not performed in the hunt phase. Figure 20 is a fragment of a typical program that receives the data sent by the previous sync transmit program example. Note that the frequencies of $\overline{TxCLK}$ on the transmitter and $\overline{RxCLK}$ on the receiver must be the same.

Figure 19. Internal Sync Detection Example

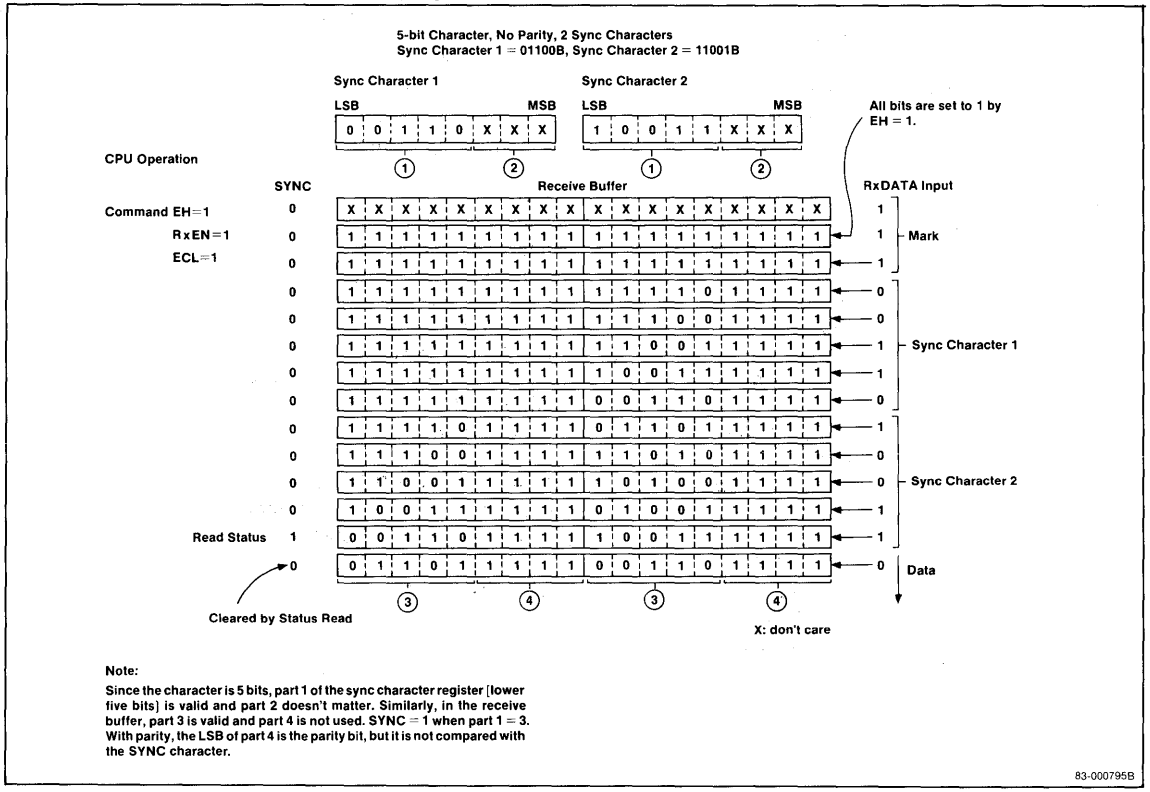


Figure 20. Synchronous Receiver Example

```
SYNRX :   CALL   SYNMOD              ;Set sync mode
          MOV    AL, 10010100B       ;Command: enter hunt
          OUT    PCTRL, AL           ;phase, clear error flags, receive enable
          MOV    BW, OFFSET RXDADR   ;Set receive data store address
RXLEN :   IN     AL, PCTRL
          TEST1  AL, 1
          BZ     RXLEN               ;Receive the number of
          IN     AL, DATA            ;receive data
          MOV    STLEN, AL           ;Set the number of
          MOV    CL, AL              ;receive data to both variable and
                                     ;counter
          MOV    CH, 00H
RXDATA :  IN     AL, PCTRL
          TEST1  AL, 1
          BZ     RXDATA              ;Receive and store the
          IN     AL, PDATA           ;number of data bytes
          MOV    [BW],AL             ;stated by the counter
          INC    BW
          DBNZ   RXDATA
          MOV    AL, 00000000B       ;Command: receive disable
          OUT    PCTRL, AL
          RET
STLEN     DB     ?                   ;Set number of receiver data
RXDADR    DB     256 DUP (0)         ;Reserve receive data area
```

5c

Standby Mode

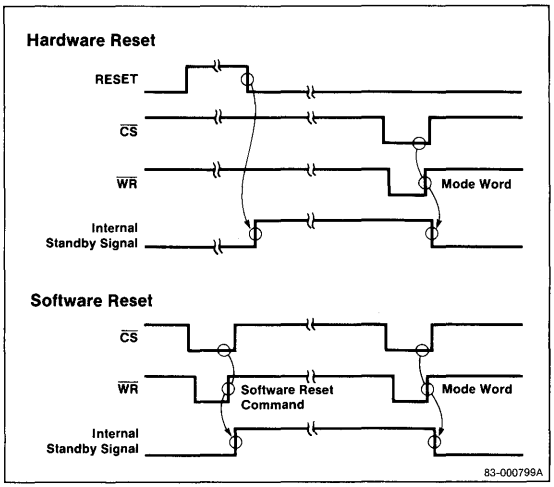
The μPD71051 is a low-power CMOS device. In standby mode, it disables the external input clocks to the inside circuitry (CLK, $\overline{TxCLK}$, and $\overline{RxCLK}$), thereby consuming less power.

A hardware reset is one way to enter standby mode. The input of a high level to the RESET pin causes the μPD71051 to enter standby mode at the falling edge of the high level. A software reset command is the other way to enter standby mode. The only way to take the μPD71051 out of standby mode is to write a mode byte.

In standby mode, the TxRDY, TxEMP, RxRDY, and SYNC/BRK pins are at low level and the TxDATA, $\overline{DTS}$, and $\overline{RTS}$ pins are at high level.

Figure 21 shows the timing for standby mode. While the internal standby signal is high, the external clocks to the μPD71051 are ignored. If data ($C/\overline{D} = 0$) is written to the μPD71051 in standby mode, the operations are undefined and unpredictable operation may result.

Figure 21. Standby Mode Timing



Description

The μPD71054 is a high-performance, programmable counter for microcomputer system timing control. Three 16-bit counters, each with its own clock input, gate input, and OUT pin, can be clocked from DC to 8 MHz. Under software control, the μPD71054 can generate accurate time delays. Initialize the counter, and the μPD71054 counts the delay, and interrupts the CPU when the task is complete. This eliminates the need for software timing loops.

The μPD71054 contains three counters capable of binary or BCD operation. There are six programmable count modes. The counters operate independently and each can be set to a different mode. Use address lines A_1, A_0 to select a counter and perform a read/write operation.

Features

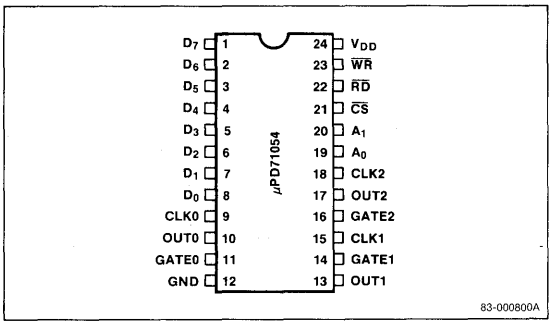
- ☐ Three independent 16-bit counters
- ☐ Six programmable counter modes
- ☐ Binary or BCD count
- ☐ Multiple latch command
- ☐ Clock rate DC (standby mode) to 8 MHz
- ☐ Low-power standby mode
- ☐ CMOS technology
- ☐ Single power supply, 5 V ±10%
- ☐ Industrial temperature range −40 to +85 °C
- ☐ 8 MHz and 10 MHz

Ordering Information

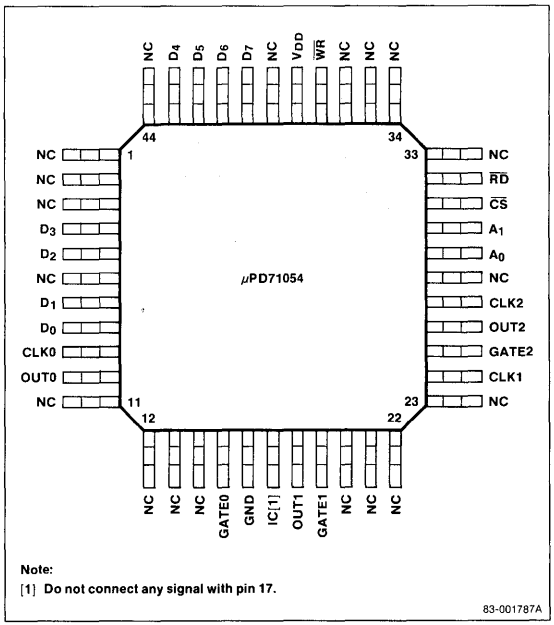
Part Number	Package Type
μPD71054C-8	24-pin plastic DIP
C-10	
G-8	44-pin plastic QFP (P44G-80-22)
GB-8	44-pin plastic QFP (P44GB-80-3B4)
GB-10	
L-8	28-pin PLCC
L-10	

Pin Configurations

24-Pin Plastic DIP



44-Pin Plastic QFP

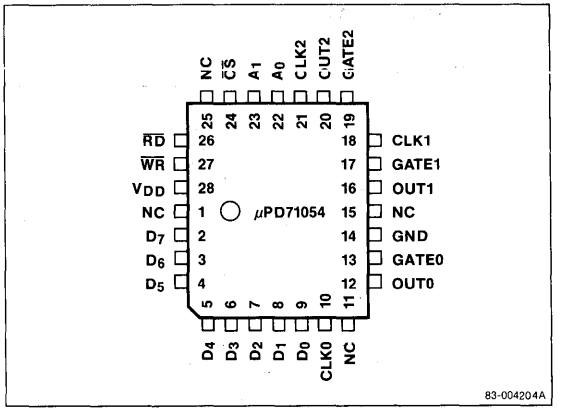


Note:
[1] Do not connect any signal with pin 17.

5d

Pin Configurations (cont)

28-Pin PLCC (Plastic Leaded Chip Carrier)



Pin Identification

Symbol	Function
D_7-D_0	Three-state, bidirectional data bus
CLKn	Counter n clock output (n = 0-2)
OUTn	Counter n output (n = 0-2)
GATEn	Output to inhibit or trigger counter n (n = 0-2)
GND	Ground
IC	Internally connected
A_0-A_1	Select counter input 0, 1, or 2
$\overline{CS}$	Chip select
$\overline{RD}$	Read strobe
$\overline{WR}$	Write strobe
V_{DD}	+5 V
NC	Not connected

Pin Functions

D_7-D_0 [Data Bus]

These pins are an 8-bit three-state bidirectional data bus. This bus is used to program counter modes and to read status and count values. The data bus is active when $\overline{CS}$ = 0, and is high impedance when $\overline{CS}$ = 1.

CLKn [Counter Clock, n = 0-2]

These pins are the clock input that determine the count rate for counter n. The clock rate may be DC (standby mode) to 8 MHz.

OUTn [Counter Output, n = 0-2]

These are the output pins for counter n. A variety of outputs is available depending on the count mode. When the μPD71054 is used as an interrupt source, these pins can output an interrupt request signal.

GATEn [Counter Gate, n = 0-2]

These output pins inhibit or trigger counter n according to the mode selected.

A_1, A_0 [Address]

These input pins select the counter. A_1, A_0 equal to 00, 01, or 10 selects counter 0, 1, or 2, respectively. The control register is selected when A_1, A_0 equals 11. These pins are normally connected to the address bus.

$\overline{CS}$ [Chip Select]

When the $\overline{CS}$ input = 1, all the bits of the data bus become high impedance. $\overline{CS}$ must be low to access the μPD71054.

$\overline{RD}$ [Read Strobe]

The $\overline{RD}$ input must be low to read data from the μPD71054.

$\overline{WR}$ [Write Strobe]

The $\overline{WR}$ input must be low to write data to the μPD71054. The contents of the data bus are written to the μPD71054 at the rising edge of $\overline{WR}$.

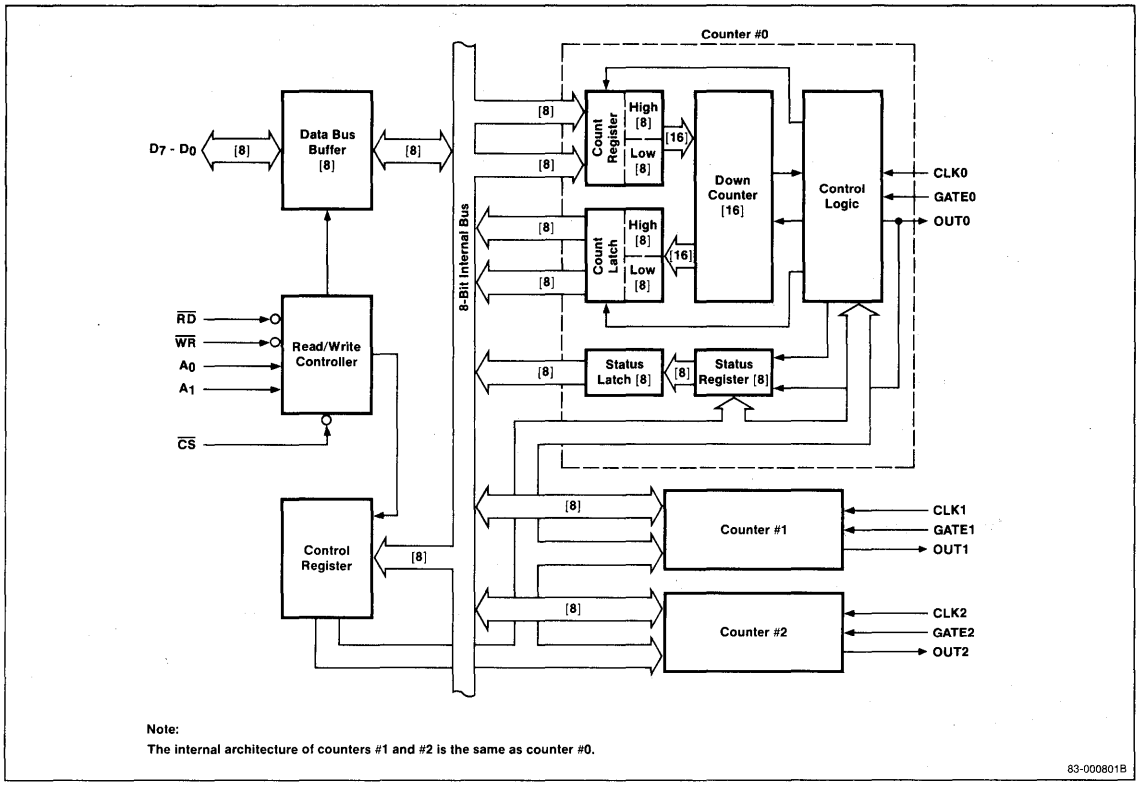
V_{DD} [Power]

+5 V.

GND [Ground]

Ground.

Block Diagram



Note:
The internal architecture of counters #1 and #2 is the same as counter #0.

5d

Block Functions

Data Bus Buffer

This is an 8-bit three-state bidirectional buffer that acts as an interface between the μPD71054 and the system data bus. The data bus buffer handles control commands, the count to be written to the count register, count data read from the count latch, and status data read from the status latch.

Read/Write Control

This circuit decodes signals from the system bus and sends control signals to other blocks of the μPD71054. A_1 and A_0 select one of the counters or the control register. A low signal on $\overline{RD}$ or $\overline{WR}$ selects a read or write operation. $\overline{CS}$ must be low to enable these operations.

Control Register

This is an 8-bit register into which is written the control command that determines the operating mode of the counter. Data is written to this register when the CPU executes an OUT command when A_1, A_0 = 11. The contents of this register cannot be read if the CPU executes an IN command when A_1, A_0 = 11. However, the multiple latch command allows you to read the mode and status of each counter.

Counter n [n = 0-2]

A 16-bit synchronous down counter performs the actual count operation within the counter. You can preset this counter and select binary or BCD operation.

The count register is a 16-bit register that stores the count when it is first written to the counter. The count is transferred to the down counter and a count operation for a specified number of counts begins.

The 8-bit width of the internal data bus permits the transfer of only eight bits at a time when the count is written to the count register. However, when data is written from the count register to the down counter, all 16 bits can be written at once. When the count is written to the count register while the counter is in read/write one byte mode, a 00H is written to the remaining byte of the register.

The count latch normally holds the current value of the down counter. If the contents of the down counter change, the contents of the count latch also change so that the two values are the same. When the μPD71054 receives a count latch command, the count latch latches the value of the down counter and holds it until the CPU can read it. When the data is read, the count latch returns to tracking the value of the down counter.

When the mode specified is written to the counter, the lower six bits of the control register are copied to the lower six bits of the 8-bit status register. The remaining two bits show the status of the OUT pin and the null count flag. When the multiple latch command is sent to the counter, the current value of the status register is latched into the status latch. This data is held in the latch until the CPU can read it.

The control logic controls each internal block according to the mode and the state of the CLK and GATE pins. The result is output to and sets the state of the OUT pin.

Absolute Maximum Ratings

$T_A = +25\,°C$

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.3 V
Output voltage, V_O	−0.5 to V_{DD} + 0.3 V
Operating temperature, T_{OPT}	−40 °C to 85 °C
Storage temperature, T_{STG}	−65 °C to +150 °C
Power dissipation, PD_{MAX}	1.0 W

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Capacitance

$T_A = +25\,°C$, $V_{DD} = 0$ V

Parameter	Symbol	Limits Min	Max	Unit	Test Conditions
Input capacitance	C_{IN}		10	pF	fc = 1 MHz Unmeasured pins returned to 0 V
I/O capacitance	$C_{I/O}$		20	pF	

DC Characteristics

$T_A = -40\,°C$ to $+85\,°C$, $V_{DD} = +5$ V ±10%

Parameter	Symbol	Limits Min	Typ	Max	Unit	Test Conditions
Input voltage high	V_{IH}	2.2		V_{DD} + 0.3	V	
Input voltage low	V_{IL}	−0.5		0.8	V	
Output voltage high	V_{OH}	0.7 x V_{DD}			V	$I_{OH} = -400\,\mu A$
Output voltage low	V_{OL}			0.4	V	$I_{OL} = 2.5$ mA
Input leakage current high	I_{LIH}			10	μA	$V_I = V_{DD}$
Input leakage current low	I_{LIL}			−10	μA	$V_I = 0$ V
Output leakage current high	I_{LOH}			10	μA	$V_O = V_{DD}$
Output leakage current low	I_{LOL}			−10	μA	$V_O = 0$ V
Supply current μPD71054	I_{DD1}			30	mA	Normal
	I_{DD2}		2	50	μA	Stand-by mode
μPD71054-10	I_{DD1}		10	20	mA	Normal
	I_{DD2}		2	50	μA	Stand-by mode

AC Characteristics

$T_A = -40°C$ to $+85°C$, $V_{DD} = 5$ V ±10%

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
Read Cycle							
Address set-up to $\overline{RD}$ ↓	t_{SAR}	30		20		ns	
Address hold from $\overline{RD}$ ↑	t_{HRA}	10		0		ns	
$\overline{CS}$ set-up to $\overline{RD}$ ↓	t_{SCR}	0		0		ns	
$\overline{RD}$ low level width	t_{RRL}	150		95		ns	
Data delay from $\overline{RD}$ ↓	t_{DRD}		120		85	ns	$C_L = 150$ pF
Data float from $\overline{RD}$ ↑	t_{FRD}	10	85	10	65	ns	$C_L = 20$ pF; $R_L = 2$ kΩ
Data delay from address	t_{DAD}		220		185	ns	$C_L = 150$ pF
Read recovery time	t_{RV}	200		165		ns	
Write Cycle							
Address set-up to $\overline{WR}$ ↓	t_{SAW}	0		0		ns	
Address hold from $\overline{WR}$ ↑	t_{HWA}	0		0		ns	
$\overline{CS}$ set-up to $\overline{WR}$ ↓	t_{SCW}	0		0		ns	
$\overline{WR}$ low level width	t_{WWL}	160		95		ns	
Data set-up to $\overline{WR}$ ↑	t_{SDW}	120		95		ns	
Data hold from $\overline{WR}$ ↑	t_{HWD}	0		0		ns	
Write recovery time	t_{RV}	200		165		ns	
CLK and Gate Timing							
CLK cycle time	t_{CYK}	125	DC	100	DC	ns	
CLK high level width	t_{KKH}	60		30		ns	
CLK low level width	t_{KKL}	60		45		ns	
CLK rise time	t_{KR}		25		25	ns	
CLK fall time	t_{KF}		25		25	ns	
GATE high level width	t_{GGH}	50		50		ns	
GATE low level width	t_{GGL}	50		50		ns	
GATE set-up to CLK ↑	t_{SGK}	50		40		ns	
GATE hold from CLK ↑	t_{HKG}	50		50		ns	
Clock delay from $\overline{WR}$ ↑ (count transfer)	t_{DWK}	100		40		ns	$t_{KKH} \geq 125$ ns
		$225 - t_{KKH}$		40		ns	$t_{KKH} \leq 125$ ns
Clock set-up to $\overline{WR}$ ↑ (latch)	t_{SKW}	85		60		ns	
GATE delay from $\overline{WR}$ ↑	t_{DWG}	0		0		ns	
OUT delay from GATE ↓	t_{DGO}		120		100	ns	$C_L = 150$ pF
OUT delay from CLK ↓	t_{DKO}		150		100	ns	$C_L = 150$ pF
OUT delay from $\overline{WR}$ ↑ (initial out)	t_{DWO}		295		240	ns	$C_L = 150$ pF

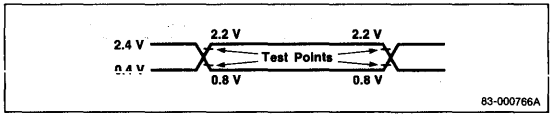
Notes:

(1) AC timing test points for output $V_{OH} = 2.2$ V, $V_{OL} = 0.8$ V

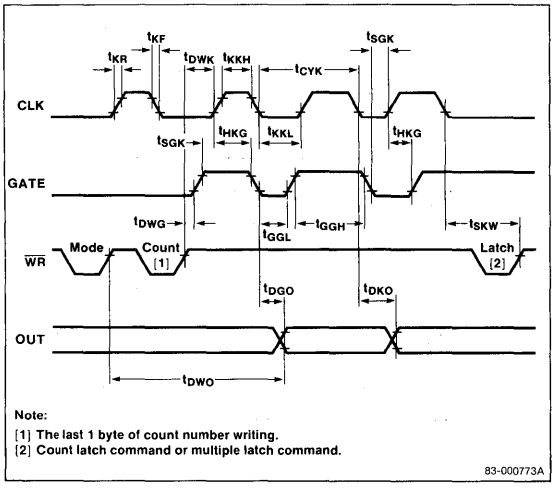
5d

Timing Waveforms

AC Test Input

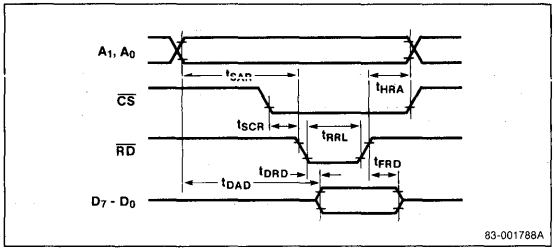


CLK and GATE Timing

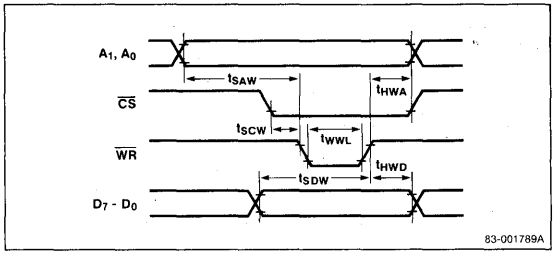


Note:
[1] The last 1 byte of count number writing.
[2] Count latch command or multiple latch command.

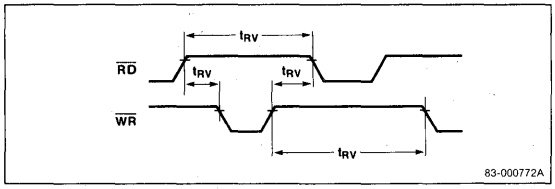
Read Cycle



Write Cycle



Read/Write Recovery



Functional Description

μPD71054 System Configuration Example

The CPU views the three counters and the control register as four I/O ports. A_1 and A_0 are connected to the A_1 and A_0 pins of the system address bus. $\overline{CS}$ is generated by decoding the address and $\overline{IO}$/MEM signals so that $\overline{CS}$ goes low when the address bus is set to the target I/O address and I/O is selected. These connections are shown in figure 1.

You can use the μPD71054 in memory-mapped I/O configurations. However, the decoding should be such that $\overline{CS}$ goes low when memory is selected.

Programming and Reading the Counter

The counter must be programmed and the operating mode specified before you can use the μPD71054. Once a mode has been selected for a counter, it operates in that mode until another mode is set. The count is written to the count register and when that data is transferred to the down counter, a new count operation begins. The current count and status can be read while the counter is in operation. Figure 2 outlines the steps of operation.

Programming the Counter

The μPD71054 is controlled by a microcomputer program. The program must write a control command to set the counter mode and write the count data that determines the length of the count operation. Table 1 shows the values for A_1 and A_0 that determine the target counter for write operations.

Table 1. Write Operations ($\overline{CS}$ = 0, $\overline{RD}$ = 1, $\overline{WR}$ = 0)

A_1	A_0	Write Target
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control word register

Control and Mode Setting

The control command must be written to set the counter mode before operating the counter. If a write operation is performed when A_1, A_0 = 11, a control command is written to the control register. Figure 3 shows the format of the 8-bit control command.

Bits SC1 and SC0 specify a counter or the multiple latch command. When a counter is chosen, the specifications described below apply to the counter.

Bits RMW1 and RMW2 specify the read/write operation to the counter or select the count latch command.

Bits CM2, CM1, and CM0 set the counter mode (0 to 5).

Bit BCD selects binary or BCD operation. The count may be 0 to FFFFH in binary mode or 0 to 9999 in BCD.

If a control command written to the counter specifies a mode, the lower six bits of the control command are copied to the lower six bits of the status register of the counter selected by SC1 and SC0. The mode selected remains in effect until a new mode is set. This is not true if the control command specifies the count latch or multiple latch command.

Writing the Count

The count is written to the counter after the mode is set. Set A_1, A_0 to specify the target counter as shown in table 1. A new count can be written to a counter at any time, but the read/write mode selected (when the mode was written) must be used when writing the count.

In high 1-byte and low 1-byte modes only, the higher or lower byte of the count register is written by the first write. The write operation ends and 00H is automatically written to the remaining byte by the μPD71054. In the 2-byte modes, the lower byte is written by the first write and the higher byte by the second.

For example, if the 2-byte count 8801H is written to a counter set in lower 1-byte mode, the lower byte (01H) is written first, followed by the higher byte (88H). Therefore, the data written to the count register is 0001H for the first write and 0088H for the second. This is shown in Table 2.

Table 2. Read/Write Mode and Count Write

Read/Write Mode	No. of Writes	Count Register Higher Byte	Count Register Lower Byte
Low 1-byte	1	00H	nnH
High 1-byte	1	nnH	00H
Low/High 2-byte	2	nnH (2nd write)	nnH (1st write)

nnH = Two-digit hexadecimal value

Reading the Counter

The following three methods allow you to read the contents of the down counter during operation. In particular, the multiple latch command reads the current count data and the counter mode or the state of the OUT pin. Table 3 shows the values of A_1, A_0 used to select the counter to be read.

5d

Figure 1. Typical System Configuration

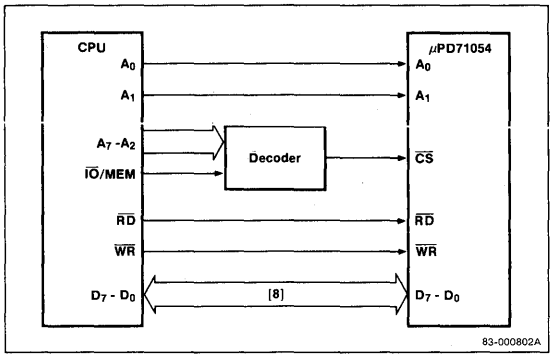


Figure 2. Basic Operating Procedure

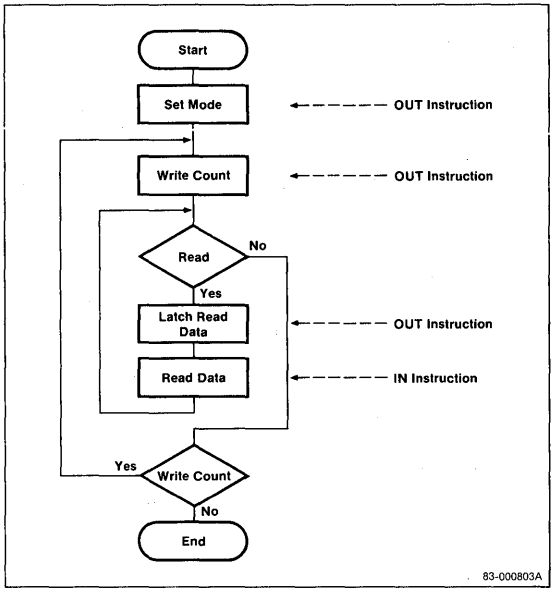
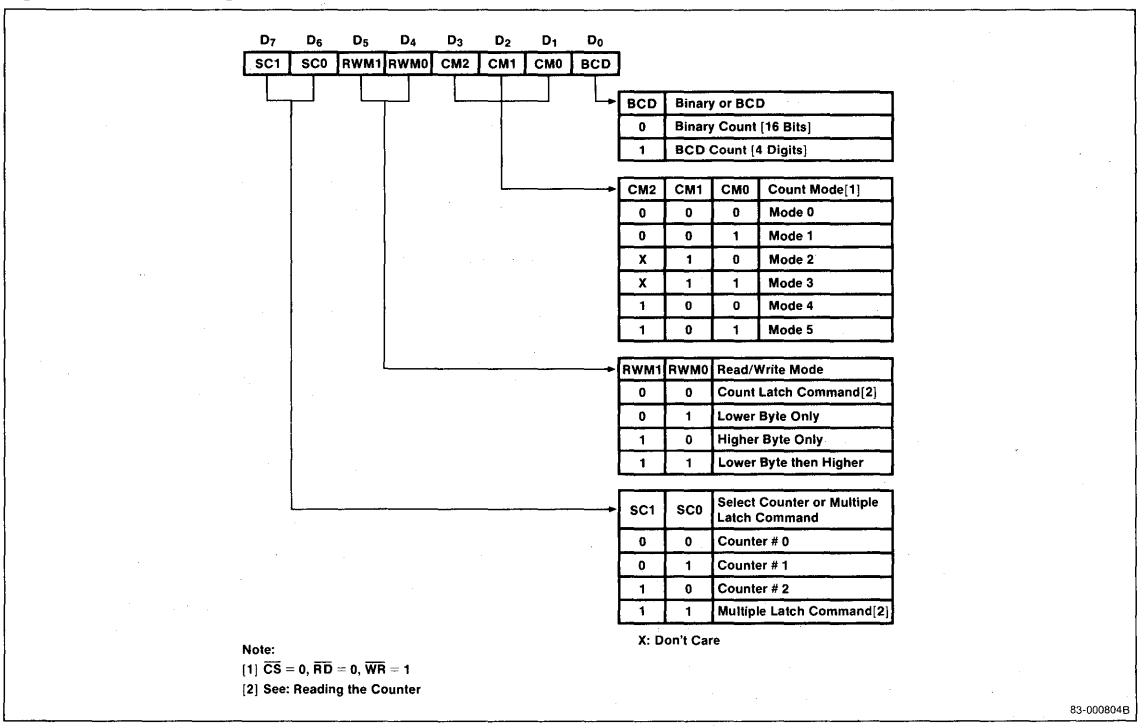


Figure 3. Control Register Format



BCD	Binary or BCD
0	Binary Count [16 Bits]
1	BCD Count [4 Digits]

CM2	CM1	CM0	Count Mode[1]
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

RWM1	RWM0	Read/Write Mode
0	0	Count Latch Command[2]
0	1	Lower Byte Only
1	0	Higher Byte Only
1	1	Lower Byte then Higher

SC1	SC0	Select Counter or Multiple Latch Command
0	0	Counter # 0
0	1	Counter # 1
1	0	Counter # 2
1	1	Multiple Latch Command[2]

X: Don't Care

Note:
[1] $\overline{CS} = 0$, $\overline{RD} = 0$, $\overline{WR} = 1$
[2] See: Reading the Counter

Table 3. Read Operations ($\overline{CS}$ = 0, $\overline{RD}$ = 0, $\overline{WR}$ = 1)

A_1	A_0	Read Target
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2

Directly Reading the Counter

You can read the current value of the counter by reading the counter selected by A_1, A_0 as shown in table 3. This involves reading the count latch; since the value of the down counter may change while the the count latch is read, this method may not provide an accurate reading. You must control the CLK or GATE input to stop the counter and read it for a correct reading.

Using the Count Latch Command

When the count latch command is executed, the current counter value is latched into the counter latch. This value is held by the latch until it is read or until a new mode is set. This provides an accurate reading of the counter value when the command is executed without affecting counter operation. Figure 4 shows the format for the count latch command.

If the counter value that was latched into the count latch is not read before a second count latch command is executed, the second command is ignored. This is because the counter value latched by the first command is held until it is read or until a new mode is set. When the data in the count latch is read, the latch is released and continues tracking the value of the down counter.

Using the Multiple Latch Command

When the multiple latch command is received, the counter value and status register for any counter may be selectively latched into the count latch and status latch. Bits D_1-D_5 of the multiple latch command specify the counter latching. The CPU can then read the status and counter value for the selected counter. Figure 5 shows the format for this command.

Bits CNT2, CNT1, and CNT0 correspond to counters 2, 1, and 0. The command is executed for all counters whose corresponding bit is 1. This allows the data for more than one counter to be latched by a single count latch command.

When the count bit is 0, the counter value of the selected counters is latched into the count latches.

When the status bit is 0, the status of the selected counters is latched into the status latches. Bits D_5-D_0 of the status register show the mode status of the counter. The output bit (D_7) shows the state of the OUT pin of that counter. These bits are shown in figure 6. The null count bit (D_6) indicates whether the count data is valid. When the count is transferred from the count register to the down counter, this bit changes to 0 to show that the data is valid. Table 4 shows how the null count flag operates.

Table 4. Null Count Flag Operation

Operation	Null Count Flag
Write control word for mode set	1
Write count to count register(1)	1
Transfer count from count register to down counter	0

Note:

(1) When 2-byte mode is selected, the flag becomes 1 when the second byte is written.

Figure 4. Control Register Format for Count Latch Command

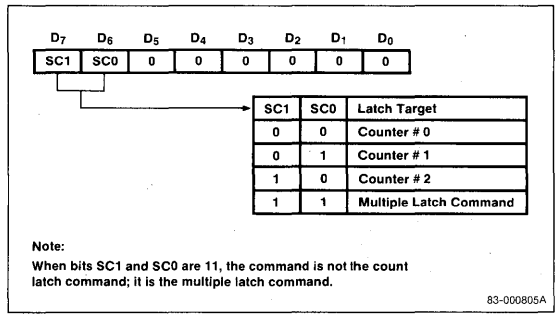
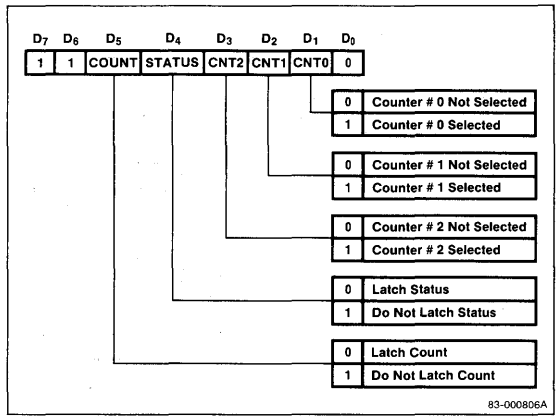


Figure 5. Control Register Format for Multiple Latch Command



5d

If the data that was latched is not read before a second multiple latch command is executed, the second command is ignored for those latches whose contents have not been read. This is because the data latched by the first command is held until it is read or until a new mode is set. When the data in the latch is read, the latch is released. See figure 7.

It is possible to latch both the count and status using two multiple latch commands. However, regardless of which data is latched first, the status is always read first. The count data is read by the next read operation (1- or 2-step read as determined by read/write mode). If additional read commands are received, the count data that has not been latched (the contents of the down counter as reflected by the current counter value) is read.

Read operations must be performed in accordance with read/write mode. In 2-byte mode, two bytes of data must always be read. This does not imply that the second byte must be read immediately after the first; other counter operations may be performed between the two reads. For example, you could read the lower byte, write a new lower byte, read the higher byte, and write a new higher byte.

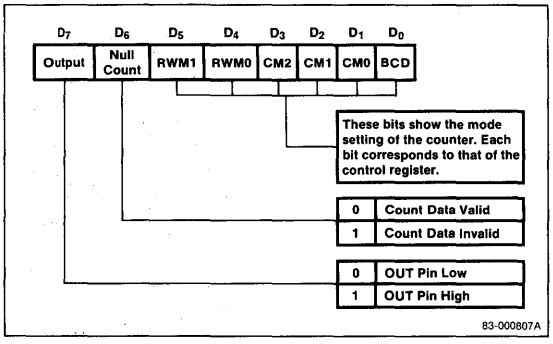
Definitions

CLK pulse refers to the time from the rising to the falling edge of the CLKn input.

Trigger refers to the rising edge of the GATEn input.

The GATEn input is sampled at each rising edge of the CLKn input. The GATE input can be level or rising edge sensitive. In the latter case, counter n's internal flip-flop is set at the rising edge of the GATE signal, sensed at the rising edge of the next CLK pulse, and reset immediately. This allows edge-triggering to be sensed whenever it occurs.

Figure 6. Status Data



Initial OUT refers to the state of the OUT pin immediately after the mode is set.

Count transfer refers to the transfer from the count register to the down counter. The down counter is decremented at the falling edge of the CLK pulse.

Count zero is the state of the down counter when the counter is decremented to zero.

PCNT0, PCNT1, and PCNT2 are the I/O ports for counters 0, 1, and 2, respectively. PCTRL is the I/O port for the control command.

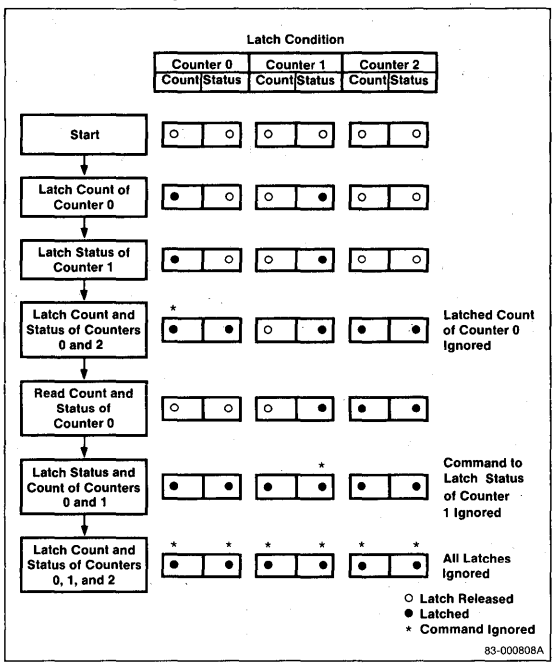
CW is the control command.

HB is the higher byte of the count.

LB is the lower byte of the count.

In the timing charts for each counter mode, counter 0 is in the read/write 1-byte and binary count mode. When no GATE signal appears in the charts, assume a high level signal. The value shown below the OUT signal is the counter value. The maximum value that can be set for the count in each mode is 0. When this value is set, a maximum value of 10000H (hexadecimal count) or 10000 (BCD count) is obtained.

Figure 7. Multiple Latch Command Execution Example



Counter Modes

Mode 0: Interrupt on End of Count. In this mode, the OUT output changes from low to high level when the end of the specified count is reached. See table 5 and figure 8.

Table 5. Mode 0 Operation

Function	Result
Initial OUT	Low level
GATE High	Count enable
GATE Low	Count disable
Count Write	The OUT pin goes low independent of the CLK pulse. In 2-byte mode, the count is disabled when the first byte is written. The OUT pin goes low. OUT goes low when a new mode or new count is written.
Count Transfer and Operation	When the count is written with GATE high: Transfer is performed at the first CLK pulse after the count value is written. The down counter is decremented beginning at the first CLK pulse after data transfer. If a count of n is set, the OUT pin goes high after n + 1 CLK pulses. When the count is written with GATE low: Transfer is performed at the first CLK pulse after the count is written. The down counter is decremented beginning at the first CLK pulse after the GATE signal goes high. If a count of n is set, OUT is low for a period of n CLK pulses after GATE goes high.
Count Zero	The signal at the OUT pin goes high. The count operation does not stop and counts down to FFFFH (hexadecimal) or 9999 (BCD) and continues to count down.
Minimum Count	1

Mode 0 Program Example. This subroutine causes a delay of 10004 (decimal, or 2710H) CLK pulses. In this program, counter 2 is set to 2-byte mode and binary count. See figure 9.

```
SUBR0:  MOV   AL,10110000B   ;set mode: counter 2,
                             ;2-byte mode,
        OUT   PCTRL,AL       ;count mode 0, binary
        MOV   AL,10H
        OUT   PCNT2,AL
        MOV   AL,27H         ;write count 10000 (decimal)
        OUT   PCNT2,AL
        RET
```

Mode 1: GATE Retriggerable One-Shot. In mode 1, the μPD71054 functions as a retriggerable one-shot. A low-level pulse triggered by the GATE input is output from the OUT pin. See table 6 and figure 10.

Table 6. Mode 1 Operation

Function	Result
Initial OUT	High level
GATE Trigger(1)	Count data is transferred at the CLK pulse after the trigger.
Count Write	The count is written without affecting the current operation.
Count Transfer and Operation	Transfer is performed at the first CLK pulse after the trigger. At the same time, the signal at the OUT pin goes low to start the one-shot pulse operation. The count is decremented beginning at the next CLK pulse. If a count of n is set, the one-shot output from the OUT pin continues for n CLK pulses.
Count Zero	The signal at the OUT pin becomes high. Count operation does not stop and wraps to FFFFH (hexadecimal) or 9999 (BCD) and continues to count.
Minimum Count	1

Note:

(1) The trigger is ignored when the count has not been written after the mode is set, or when only one byte of the count has been written in 2-byte count mode.

5d

Figure 8. Mode 0 Timing Chart

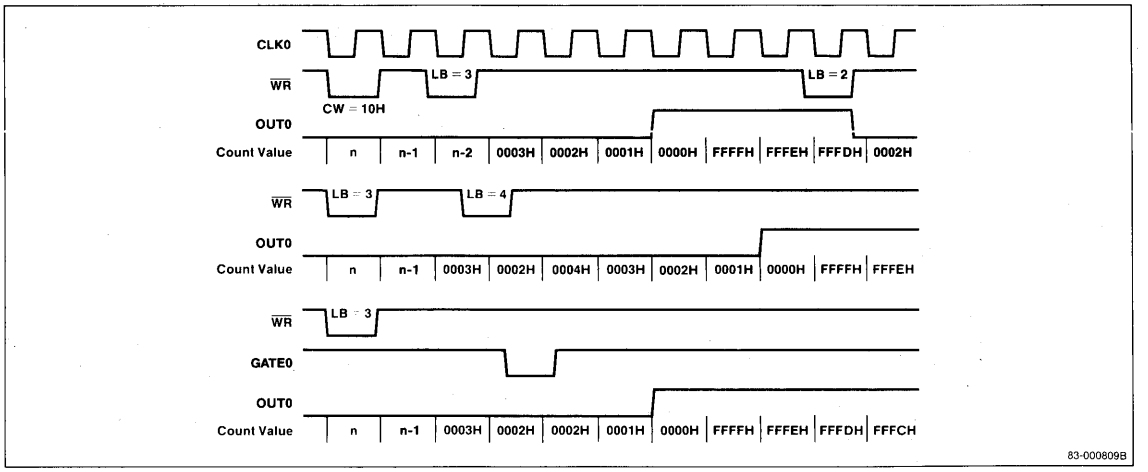


Figure 9. Mode 0 Program Example Timing Chart

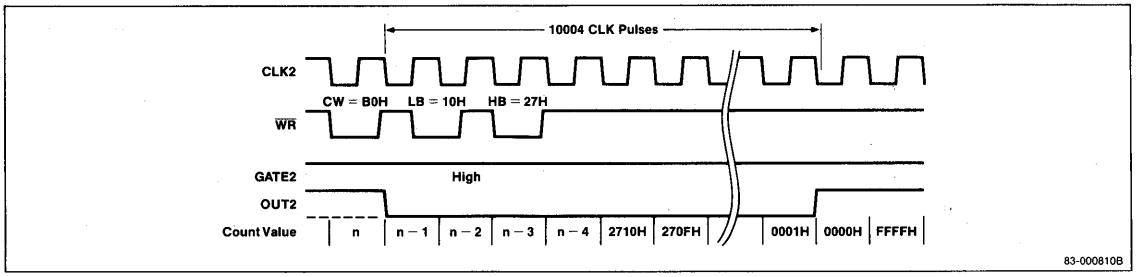
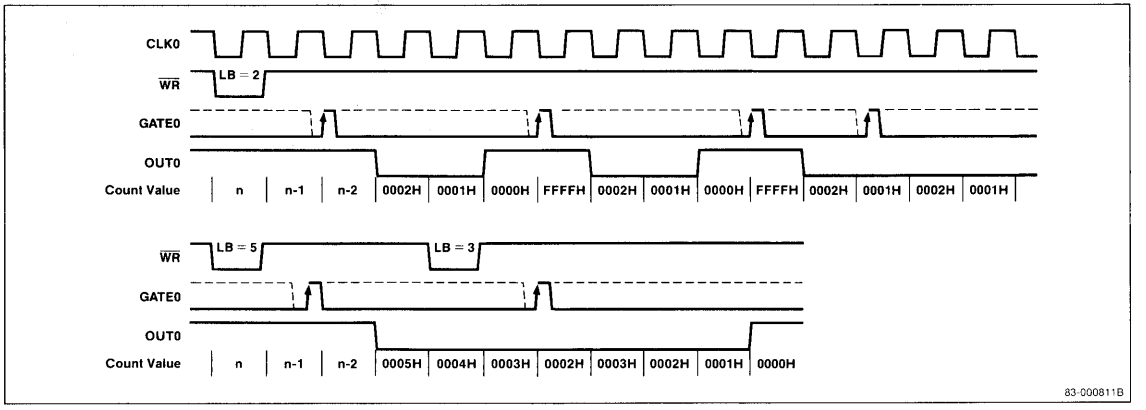


Figure 10. Mode 1 Timing Chart



Mode 1 Program Example. This subroutine waits until no trigger is generated for an interval of 200 or more CLK pulses after the first gate trigger and returns to the main program. Counter 1 is set to low-byte read/write mode and binary count. See figure 11.

```
SUBR1:  MOV   AL,01010010B  ;set mode: counter 1, low-byte
        OUT   PCTRL,AL      ;read/write mode, count mode 1,
        MOV   AL,200        ;binary
        OUT   PCNT1,AL      ;write low byte of count
        ;
FSTTRG: MOV   AL,11100100B  ;multiple latch command:
                            ;counter 1,
        OUT   PCNT1,AL      ;status
        IN    AL,PCNT1
        TEST1 AL,7          ;wait for first trigger
        BNZ   FSTTRG
        ;
WAIT:   MOV   AL,11100100B  ;multiple latch command:
                            ;counter 1,
        OUT   PCTRL,AL      ;status
        IN    AL,PCNT1
        TEST1 AL,7          ;wait until output goes high
        BZ    WAIT
        RET
```

Mode 2: Rate Generator. In mode 2, the signal from the OUT pin cyclically goes low for one clock period when the counter reaches 0001H. The counter operates as a frequency divider. See table 7 and figure 12.

Table 7. Mode 2 Operation

Function	Result
Initial OUT	High level
GATE High	Count enable
GATE Low	Count disabled. If GATE goes low when OUT is low, OUT will go high (independent of the CLK pulse).
GATE Trigger(1)	Transfer is performed at the first CLK pulse after the trigger.
Count Write	Count is written without affecting the current operation.
Count Transfer and Operation	Transfer is performed at the CLK pulse after the count is written following the mode setting. The counter is then decremented. Transfer is again performed at the first CLK pulse after the count becomes 1. When the trigger is used, transfer is performed at the next CLK pulse. When the contents of the down counter becomes 1, OUT goes low for one CLK pulse and returns to high. If a count of n is set, OUT repeats this sequence every n CLK pulses.
Count Zero	Never occurs in this mode.
Minimum Count	2

Note:

(1) The trigger is ignored when the count has not been written or when only one byte of the count has been written in 2-byte mode.

Figure 11. Mode 1 Program Example Timing Chart

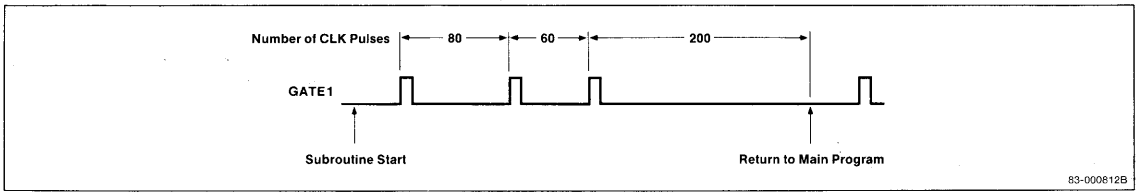
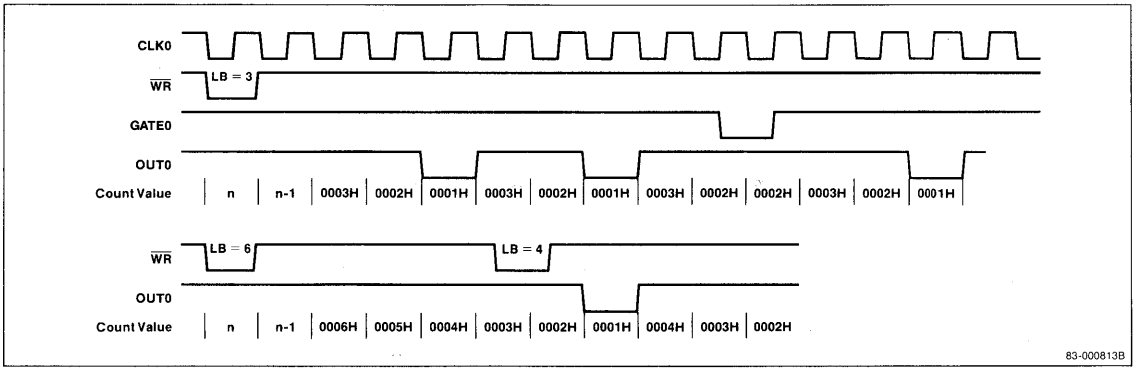


Figure 12. Mode 2 Operation Timing Chart



5d

Mode 2 Program Example. This subroutine generates an interrupt to the CPU each time 10000 (decimal) clock pulses elapse. Counter 0 is in 2-byte mode and binary counting. See figure 13.

```
SUBR3:  MOV  AL,00110100B   ;mode setting: counter 0, 2-byte
        OUT  PCTRL,AL       ;mode, count mode 2, binary
        MOV  AL,10H
        OUT  PCNT0,AL
        MOV  AL,27H         ;write count 10000 (decimal)
        OUT  PCNT0,AL
        RET
```

Mode 3: Square Wave Generator. Mode 3 is a frequency divider similar to mode 2, but with a different duty cycle. See table 8 and figure 14.

Figure 13. Mode 2 Configuration

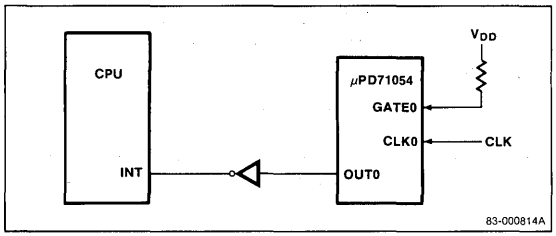


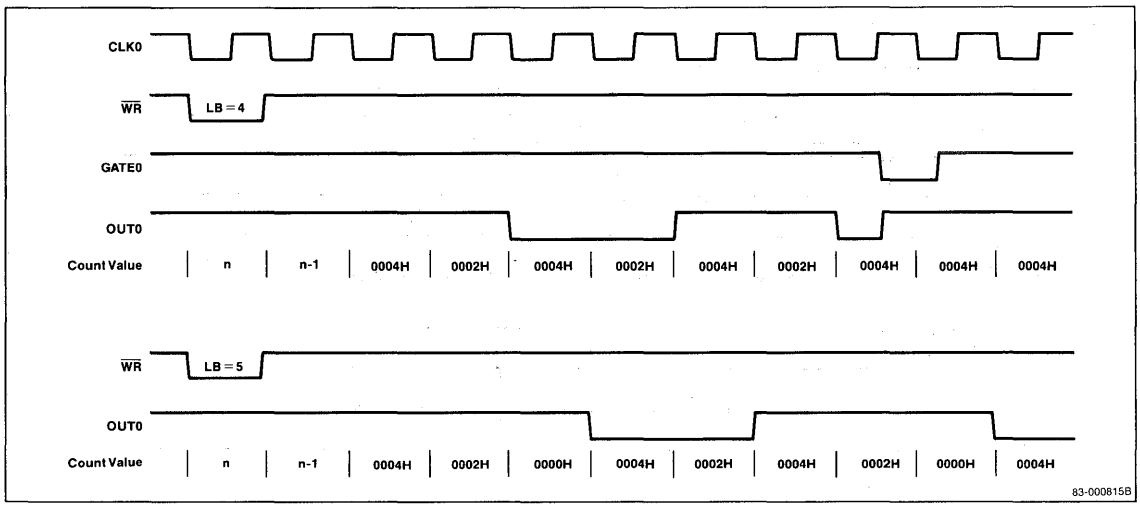
Table 8. Mode 3 Operation

Function	Result
Initial OUT	High level
GATE High	Count enable
GATE Low	Count disable. If GATE goes low when OUT is low, OUT will go high (independent of the CLK pulse).
GATE Trigger(1)	Transfer is performed at the first CLK pulse after the trigger.
Count Write	Current operation is not affected. The count is transferred at the end of the half-period of the current square wave and the OUT pin goes high.
Count Transfer and Operation	Count data is transferred at the first CLK pulse after the count write following the mode setting. Transfer is performed at the end of the current half-cycle and the OUT pin is inverted. Transfer is also performed at the CLK pulse after the trigger. The operation performed depends on whether count n is even or odd. When n is even, the count is decremented by two on each following clock pulse. At the end of the count of two, the count is again transferred and the OUT pin is inverted. This is taken as a half-cycle and repeated. When n is odd, n − 1 is transferred and the count is decremented by two on each following clock pulse. The half-cycle when the OUT pin is high continues until the end of count 0 and n − 1 is transferred again at the next CLK pulse. The half-cycle while OUT is low continues until the end of count 2. Thus, the half-cycle while OUT is high is one CLK longer than the half-cycle while OUT is low.
Count Zero	Occurs only when the count is odd.
Minimum Count	3

Note:

(1) The trigger is ignored when the count has not been written after the mode is set or when only one byte of count has been written in 2-byte mode.

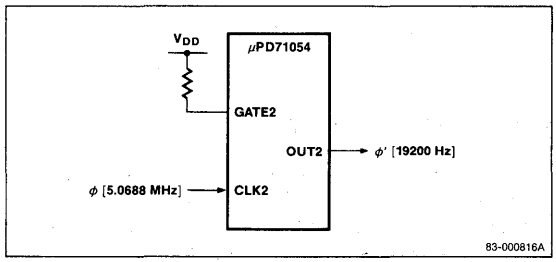
Figure 14. Mode 3 Timing Chart



Mode 3 Program Example. This subroutine divides the input CLK frequency (5.0688 MHz) by 264 to get a 19,200 Hz clock. Counter 2 is in 2-byte binary mode. See figure 15.

```
SUBR4:  MOV   AL,10110110B   ;mode setting: counter 2, 2-byte
        OUT   PCTRL,AL       ;mode, count mode 3, binary
        MOV   AL,08H
        OUT   PCNT2,AL
        MOV   A,01H          ;264 frequency division
        OUT   PCNT2,AL
        RET
```

Figure 15. Frequency Division

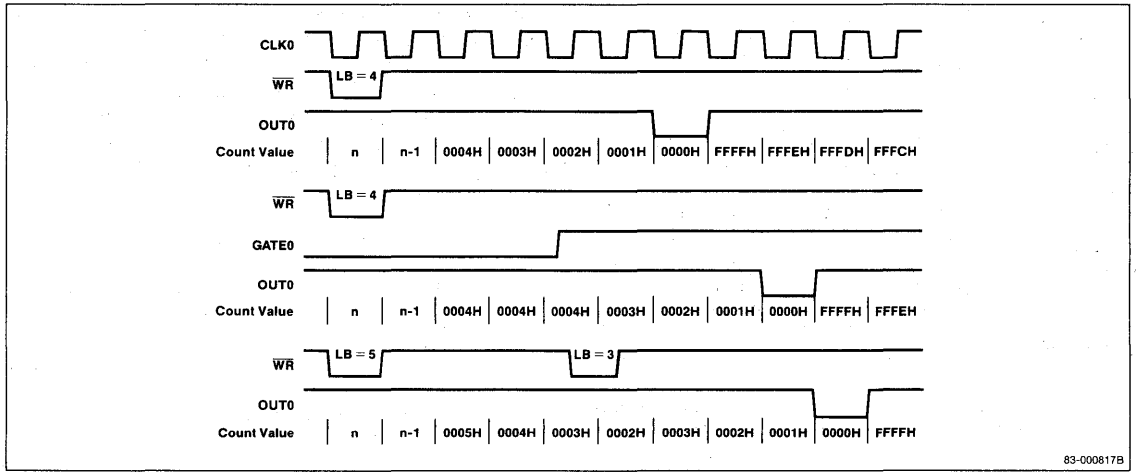


Mode 4: Software-Triggered Strobe. In mode 4, when the specified count is reached, OUT goes low for one CLK pulse. See table 9 and figure 16.

Table 9. Mode 4 Operation

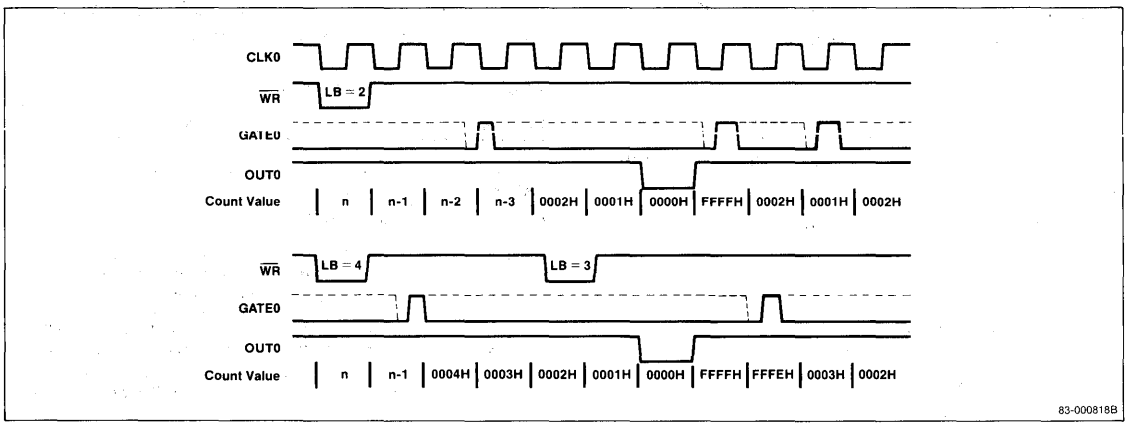
Function	Result
Initial OUT	High level
GATE High	Count enable
GATE Low	Count disable
Count Write	Count is transferred at the next CLK pulse when the count is written. In 2-byte mode, data is transferred after the second byte is written.
Count Transfer and Operation	Count is transferred at the first CLK following the count write. If GATE is high, the down counter begins to decrement from the next CLK. If GATE is low, decrement begins at the first CLK after GATE goes high.
Count Zero	OUT is low for one CLK pulse and returns to high. The down counter wraps to FFFFH (hexadecimal) or 9999 (BCD) without stopping counter operation.
Minimum Count	1

Figure 16. Mode 4 Timing Chart



5d

Figure 17. Mode 5 Timing Chart



Mode 5: Hardware-Triggered Strobe [Retriggerable]. Mode 5 is similar to mode 4 except that operation is triggered by the GATE input and can be retriggered. See table 10 and figure 17.

Table 10. Mode 5 Operation

Function	Result
Initial OUT	High level
GATE Trigger(1)	The count is transferred at the CLK pulse after the trigger. The GATE has no effect on the OUT signal.
Count Write	The count is written without affecting the current operation.
Count Transfer and Operation	Count is transferred at the first CLK pulse after a trigger, providing that the mode and count have been written. Decrement begins from the first CLK pulse after a data transfer. If a count of n is set, OUT goes low for n + 1 CLK pulses after the trigger.
Count Zero	OUT is low for one CLK and goes high again. The down counter counts to FFFFH (hexadecimal) or 9999 (BCD) without stopping the counter operation.
Minimum Count	1

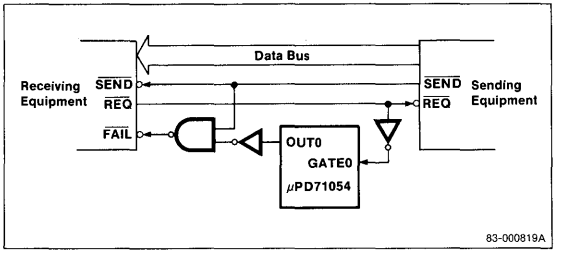
Note:

(1) The trigger is ignored when the count has not been written after the mode is set or when only one byte has been written in 2-byte mode.

Mode 5 Program Example. Use mode 5 to add a fail-safe function to an interface. For example, the receiving equipment requests data by issuing a $\overline{REQ}$ signal to the sending equipment. The sending equipment responds by outputting data to the data bus and returning a $\overline{SEND}$ signal to the receiving equipment. In this type of system, if a malfunction exists in the sending equipment and no $\overline{SEND}$ signal is sent, the receiving equipment waits indefinitely for the $\overline{SEND}$ signal and system operation stops. The following subroutine remedies this situation. If no $\overline{SEND}$ signal is output within a given period (50 CLK cycles in this example) after the $\overline{REQ}$ signal is output, the system assumes the sending equipment is malfunctioning and a $\overline{FAIL}$ signal is sent to the receiving equipment.

```
SUBR5:  MOV  AL,00011010B   ;mode setting: counter 0, low
                            ;1-byte
        OUT  PCTRL,AL       ;mode, count mode 5, binary
        MOV  AL,50          ;set interval: 50 CLK pulses
        OUT  PCNT0,AL
        RET
```

Figure 18. Interface Fail-safe Example



Description

The μPD71055 is a low-power CMOS programmable parallel interface unit for use in microcomputer systems. Typically, the unit's three I/O ports interface peripheral devices to the system bus.

Features

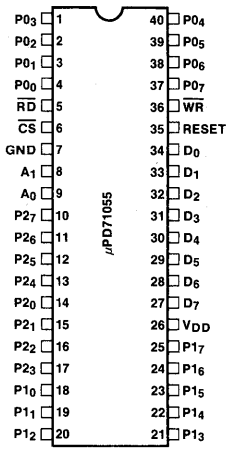
- ☐ Three 8-bit I/O ports
- ☐ Three programmable operation modes
- ☐ Bit manipulation command
- ☐ Microcomputer compatible
- ☐ CMOS technology
- ☐ Single +5 V ±10% power supply
- ☐ Industrial temperature range: −40 to +85 °C
- ☐ 8 MHz and 10 MHz

Ordering Information

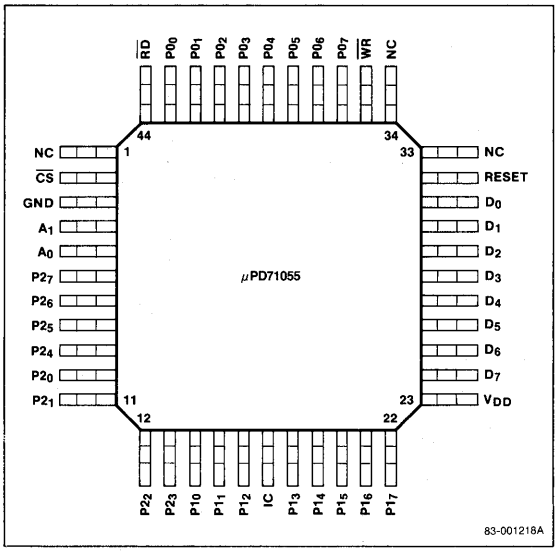
Part Number	Clock (MHz)	Package
μPD71055C-8	8	40-pin plastic DIP
C-10	10	
G-8	8	44-pin plastic QFP (P44G-80-22)
GB-8	8	44-pin plastic QFP (P44GB-80-3B4)
GB-10	10	
L-8	8	44-pin PLCC
L-10	10	

Pin Configurations

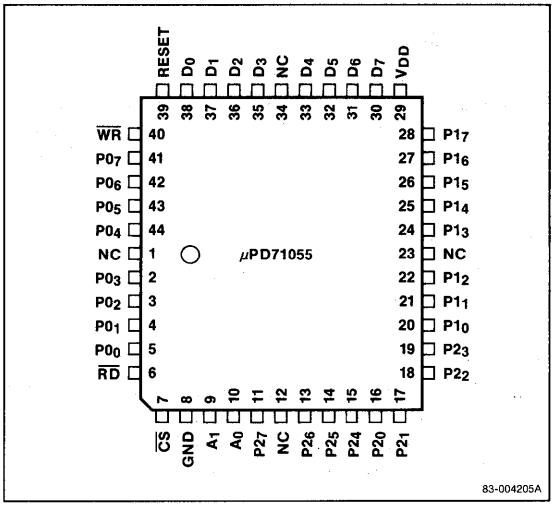
40-Pin Plastic DIP



44-Pin Plastic QFP



44-Pin Plastic Leaded Chip Carrier (PLCC)



5e

Pin Identification

Symbol	Function
$\overline{CS}$	Chip select input
GND	Ground
A_1, A_0	Address inputs 1 and 0
$P0_7$-$P0_0$	I/O port 0, bits 7-0
$P1_7$-$P1_0$	I/O port 1, bits 7-0
$P2_7$-$P2_0$	I/O port 2, bits 7-0
IC	Internally connected
V_{DD}	+5 V
D_7-D_0	I/O data bus
RESET	Reset input
$\overline{WR}$	Write strobe input
$\overline{RD}$	Read strobe input
NC	No connection

Pin Functions

D_7-D_0 [Data Bus]

D_7-D_0 make up an 8-bit, three-state, bidirectional data bus. The bus is connected to the system data bus. It is used to send commands to the μPD71055 and to send data to and from the μPD71055.

$\overline{CS}$ [Chip Select]

The $\overline{CS}$ input is used to select the μPD71055. When $\overline{CS} = 0$, the μPD71055 is selected and the states of the D_7-D_0 pins are determined by the $\overline{RD}$ and $\overline{WR}$ inputs. When $\overline{CS} = 1$, the μPD71055 is not selected and its data bus is high-impedance.

$\overline{RD}$ [Read Strobe]

The $\overline{RD}$ input is set low when data is being read from the μPD71055 data bus.

$\overline{WR}$ [Write Strobe]

The $\overline{WR}$ input should be set low when data is to be written to the μPD71055 data bus. The contents of the data bus are written to the μPD71055 at the rising edge (low to high) of the $\overline{WR}$ signal.

A_1, A_0 [Address]

The A_1 and A_0 inputs are used in combination with the $\overline{RD}$ and $\overline{WR}$ signals to select one of the three ports or the command register. A_1 and A_0 are usually connected to the lower two bits of the system address bus (table 1).

$\overline{WR}$ [Write Strobe]

The $\overline{WR}$ input should be set low when data is to be written to the μPD71055 data bus. The contents of the data bus are written to the μPD71055 at the rising edge (low to high) of the $\overline{WR}$ signal.

A_1, A_0 [Address]

The A_1 and A_0 inputs are used in combination with the $\overline{RD}$ and $\overline{WR}$ signals to select one of the three ports or the command register. A_1 and A_0 are usually connected to the lower two bits of the system address bus (table 1).

Table 1. Control Signals and Operation

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	Operation	μPD71055 Operation
0	0	1	0	0	Port 0 to data bus	Input
0	0	1	0	1	Port 1 to data bus	Input
0	0	1	1	0	Port 2 to data bus	Input
0	0	1	1	1	Use prohibited	
0	0	0	x	x		
0	1	0	0	0	Data bus to port 0	Output
0	1	0	0	1	Data bus to port 1	Output
0	1	0	1	0	Data bus to port 2	Output
0	1	0	1	1	Data bus to command register	Output
0	1	1	x	x	Data bus high impedance	
1	x	x	x	x		

RESET [Reset]

When the RESET input is high, the μPD71055 is reset. The group 0 and the group 1 ports are set to mode 0 (basic I/O port mode). All port bits are cleared to zero and all ports are set for input.

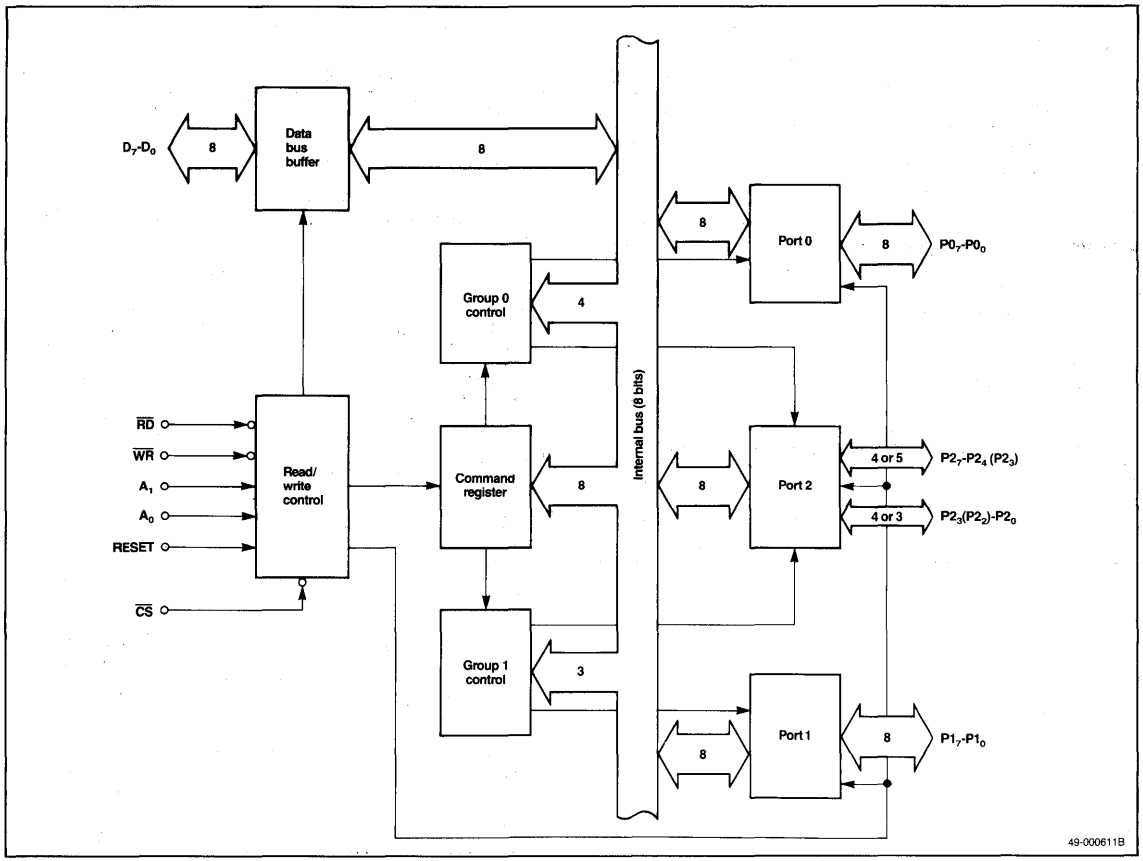
$P0_7$-$P0_0$, $P1_7$-$P1_0$, $P2_7$-$P2_0$ [Ports 0, 1, 2]

Pins $P0_7$-$P0_0$, $P1_7$-$P1_0$, and $P2_7$-$P2_0$ are the port 0, 1, and 2 I/O pins, bits 7-0, respectively.

IC [Internally Connected]

Pins marked IC are used internally and must be left unconnected.

Block Diagram



Functional Description

Ports 0, 1, 2

The μPD71055 has three 8-bit I/O ports, referred to as port 0, port 1, and port 2. These ports are divided into two groups, group 0 and group 1. The groups can be in one of three modes, mode 0, mode 1, and mode 2. Modes can be set independently for each group.

When port 0 is in mode 0, port 0 and the four upper bits of port 2 belong to group 0, and port 1 and the four lower bits of port 2 belong to group 1. When port 0 is in mode 1 or 2, port 0 and the 5 upper bits of port 2 belong to group 0 and port 1 and the three lower bits of port 2 belong to group 1.

Command Register

The host writes command words to the μPD71055 in this register. These commands control group 0 and group 1. Note that the contents of this register cannot be read.

Group 0 Control and Group 1 Control

These blocks control the operation of group 0 and group 1.

Read/Write Control

The read/write control controls the read/write operations for the ports and the data bus in response to the $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, and address signals. It also handles RESET signals and the A_0, A_1 address inputs.

Data Bus Buffer

The data bus buffer latches information going to or from the system data bus.

5e

Absolute Maximum Ratings

(T_A = 25 °C)

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.3 V
Output voltage, V_O	−0.5 to V_{DD} + 0.3 V
Power dissipation, PD_{MAX}	500 mW
Operating temperature, T_{opt}	−40 to +85 °C
Storage temperature, T_{stg}	−65 to +150 °C

Comment: These devices are not meant to be operated outside the limits specified above. Exposure to stresses beyond those listed in Absolute Maximum Ratings could cause damage. Exposure to an absolute maximum rating for extended periods may affect reliability.

Capacitance

(T_A = 25 °C, V_{DD} = GND = 0 V)

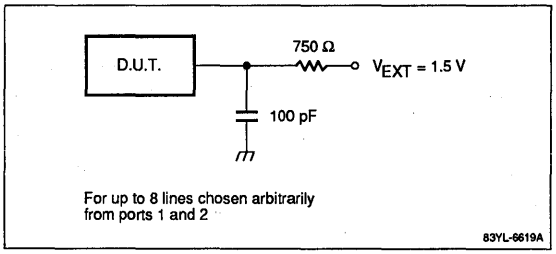
Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Units	Test Conditions
Input capacitance	C_I			10	pF	fc = 1 MHz Unmeasured pins returned to 0 V
I/O capacitance	C_{IO}			20	pF	

DC Characteristics

(T_A = −40 to +85 °C, V_{DD} = 5 V ±10%)

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Units	Test Conditions
Input voltage high	V_{IH}	2.2		V_{DD} + 0.3	V	
Input voltage low	V_{IL}	−0.5		0.8	V	
Output voltage high	V_{OH}	0.7 V_{DD}			V	I_{OH} = −400 μA
Output voltage low	V_{OL}			0.4	V	I_{OL} = 2.5 mA
Darlington drive current	I_{DAR}	−1.0		−4.0	mA	See test setup diagram
Input leakage current high	I_{LIH}			10	μA	$V_I = V_{DD}$
Input leakage current low	I_{LIL}			−10	μA	V_I = 0 V
Output leakage current high	I_{LOH}			10	μA	$V_O = V_{DD}$
Output leakage current low	I_{LOL}			−10	μA	V_O = 0 V
Supply current (dynamic) μPD71055	I_{DD1}			10	mA	Normal operation
μPD71055-10	I_{DD1}		5	10	mA	Normal operation
Supply current (standby)	I_{DD2}		2	50	μA	Inputs: RESET = 0.1 V, others = V_{DD} − 0.1 V Outputs: Open

Test Setup for I_{DAR} Measurement



AC Characteristics

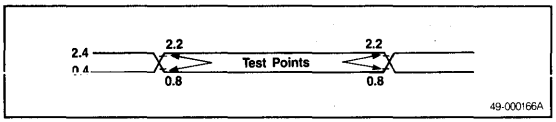
(T_A = −40 to +85°C, V_{DD} = 5 V ±10%)

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
Read Timing							
A_1, A_0, $\overline{CS}$ set-up to $\overline{RD}$ ↓	t_{SAR}	0		0		ns	
A_1, A_0, $\overline{CS}$ hold from $\overline{RD}$ ↑	t_{HRA}	0		0		ns	
$\overline{RD}$ pulse width	t_{RRL}	160		150		ns	
Data delay from $\overline{RD}$ ↓	t_{DRD}		120		100	ns	C_L = 150 pF
Data float from $\overline{RD}$ ↑	t_{FRD}	10	85	10	60	ns	C_L = 20 pF; R_L = 2 kΩ
Read recovery time	t_{RV}	200		150		ns	
Write Timing							
A_1, A_0, $\overline{CS}$ set-up to $\overline{WR}$ ↓	t_{SAW}	0		0		ns	
A_1, A_0, $\overline{CS}$ hold from $\overline{WR}$ ↑	t_{HWA}	0		0		ns	
$\overline{WR}$ pulse width	t_{WWL}	120		100		ns	
Data set-up to $\overline{WR}$ ↑	t_{SDW}	100		100		ns	
Data hold from $\overline{WR}$ ↑	t_{HWD}	0		0		ns	
Write recovery time	t_{RV}	200		150		ns	
Other Timing							
Port set-up time to $\overline{RD}$ ↓	t_{SPR}	0		0		ns	
Port hold time from $\overline{RD}$ ↑	t_{HRP}	0		0		ns	
Port set-up time to $\overline{STB}$ ↓	t_{SPS}	0		0		ns	
Port hold time from $\overline{STB}$ ↑	t_{HSP}	150		150		ns	
Port delay time from $\overline{WR}$ ↑	t_{DWP}		350		200	ns	C_L = 150 pF
$\overline{STB}$ pulse width	t_{SSL}	350		100		ns	
$\overline{DAK}$ pulse width	t_{DADAL}	300		100		ns	
Port delay time from $\overline{DAK}$ ↓(mode 2)	t_{DDAP}		300		150	ns	CL = 150 pF
Port float time from $\overline{DAK}$ ↑(mode 2)	t_{FDAP}	20	250	20	250	ns	C_L = 20 pF; R_L = 2 kΩ
$\overline{OBF}$ set delay from $\overline{WR}$ ↑	t_{DWOB}		300		150	ns	C_L = 150 pF
$\overline{OBF}$ clear delay from $\overline{DAK}$ ↓	t_{DDAOB}		350		150	ns	
IBF set delay from $\overline{STB}$ ↓	t_{DSIB}		300		150	ns	
IBF clear delay from $\overline{RD}$ ↑	t_{DRIB}		300		150	ns	
INT set delay from $\overline{DAK}$ ↑	t_{DDAI}		350		150	ns	
INT clear delay from $\overline{WR}$ ↓	t_{DWI}		450		200	ns	
INT set delay from $\overline{STB}$ ↑	t_{DSI}		300		150	ns	
INT clear delay from $\overline{RD}$ ↓	t_{DRI}		400		200	ns	
RESET pulse width	t_{RESET1}	50		50		μs	During right after power-on
	t_{RESET2}	500		500		ns	During operation

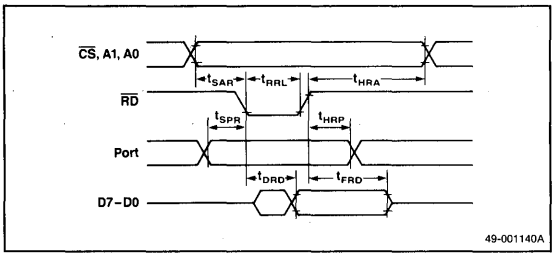
5e

Timing Waveforms

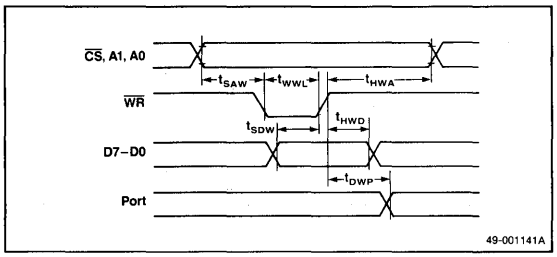
AC Test Waveform



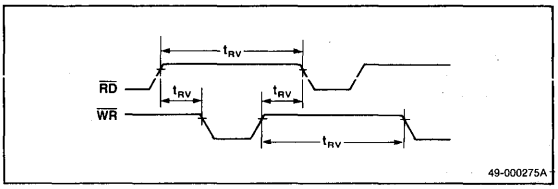
Timing Mode 0: Input



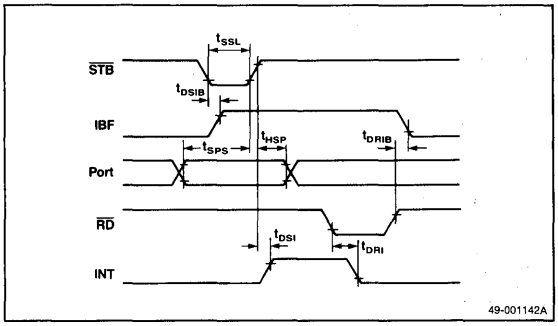
Mode 0: Output



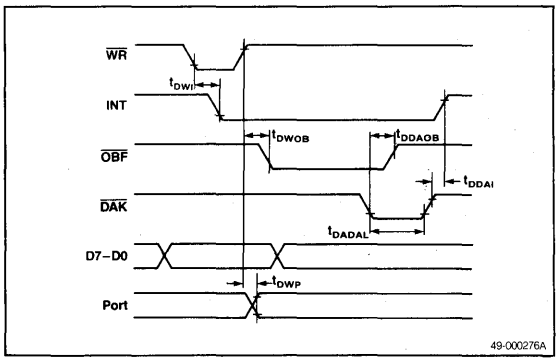
Recovery Time



Mode 1: Input

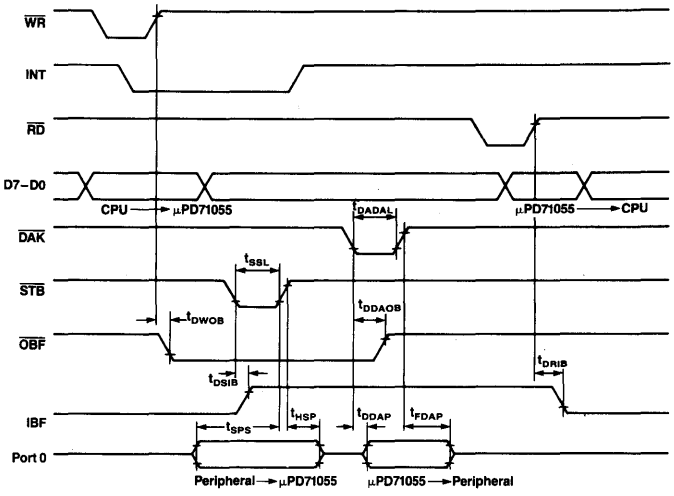


Mode 1: Output



Timing Waveforms (cont)

Mode 2



5e

μPD71055 Commands

Two commands control μPD71055 operation. The mode select command determines the operation of group 0 and group 1 ports. The bit manipulation command sets or resets the bits of port 2. These commands are executed by writing an 8-bit command word to the command register ($A_1A_0 = 11$).

Mode Select

The μPD71055 port groups have three modes. Modes 0 and 1 can be specified for groups 0 and 1, but mode 2 can only be specified for group 0. The bits of all ports are cleared when a mode is selected or when the μPD71055 is reset.

Mode 0. Basic input/output port operation.

Mode 1. Strobed input/output operation controlled by three or four bits of port 2 used as control/status signals.

Mode 2. (Only available for group 0). Port 0 is the bidirectional I/O port and the higher 5 bits of port 2 are used for status and control signals.

To specify the mode, set the command word as shown in figure 1 and write it to the command register.

Bit Manipulation Command

This command (figure 2) affects only port 2. It is mainly used in mode 1 and mode 2 to control the port 2 bits which are used as control/status signals. It is also used to enable and disable μPD71055-generated interrupts and to set and reset port 2 general input/output pins.

For example, to set bit 2 of port 2 to 1 ($P2_2 = 1$), set the command word as shown in figure 3 (05H) in the command register.

Operation in Each Mode

The operation mode for each group in the μPD71055 can be set according to the application. Group 0 can be in modes 0, 1, or 2, while group 1 is in mode 0 or 1. Group 1 cannot be used in mode 2.

The $\overline{RD}$ and $\overline{WR}$ signals that appear in the descriptions of each mode refer to the port in question as addressed by A_1 and A_0. These signals only affect the port addressed by A_1 and A_0.

Where the port addressed may not be clear, 0 or 1 is appended to the signal name to indicate the port.

Mode 0

In this mode the ports of the μPD71055 are used to perform basic I/O operations. Each port operates with a buffered input and a buffered latched output. See figure 4.

Depending on the control word sent to the μPD71055 from the system bus, ports 0, 1, and 2 can be independently specified for input or output.

Input Port Operation

While the $\overline{RD}$ signal is low, data from the port selected by the A_1A_0 signals is put on the data bus. See figure 5.

Output Port Operation

When the μPD71055 is written to ($\overline{WR} = 0$), the data on the data bus will be latched in the port selected by the A_1A_0 signals at the rising edge of $\overline{WR}$ and output to the port pins (figure 6). Following the programming of mode 0, all outputs are at a low level.

By reading a port which is set for output, the output value of the port can be obtained.

Note: When group 0 is in mode 1 or mode 2, only bits $P2_2$-$P2_0$ of port 2 can be used by group 1. Bit $P2_3$ belongs to group 0.

Mode 0 Example

This is an example of a CPU connected to an A/D converter via a μPD71055 (figure 7). Here both group 0 and group 1 are set to mode 0 and port 2 is used to start conversion and detect the end of the conversion process.

Figure 8 is a subroutine that reads the converted data from an A/D converter.

Figure 1. Mode Select Command Word

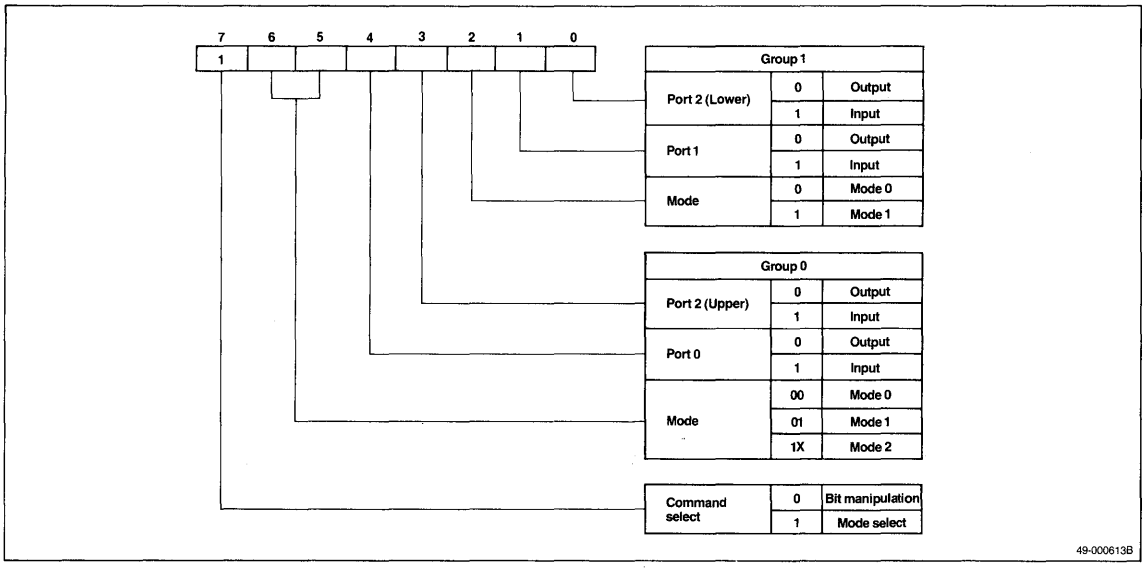


Figure 2. Bit Manipulation Command Word

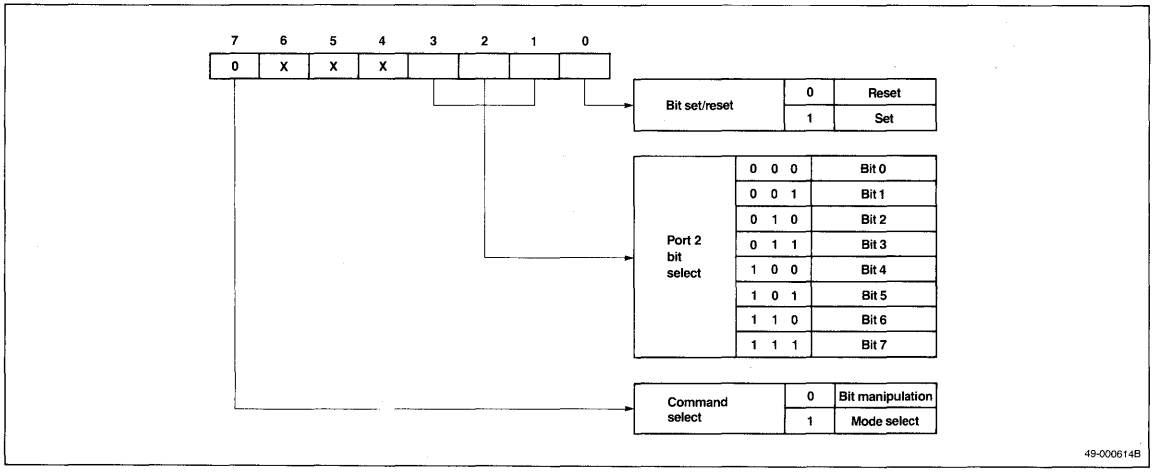


Figure 3. Bit Manipulation Command Example

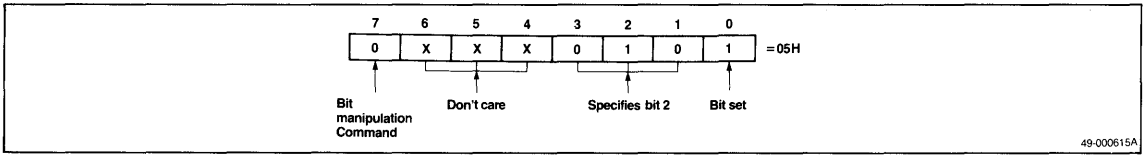


Figure 4. Mode 0

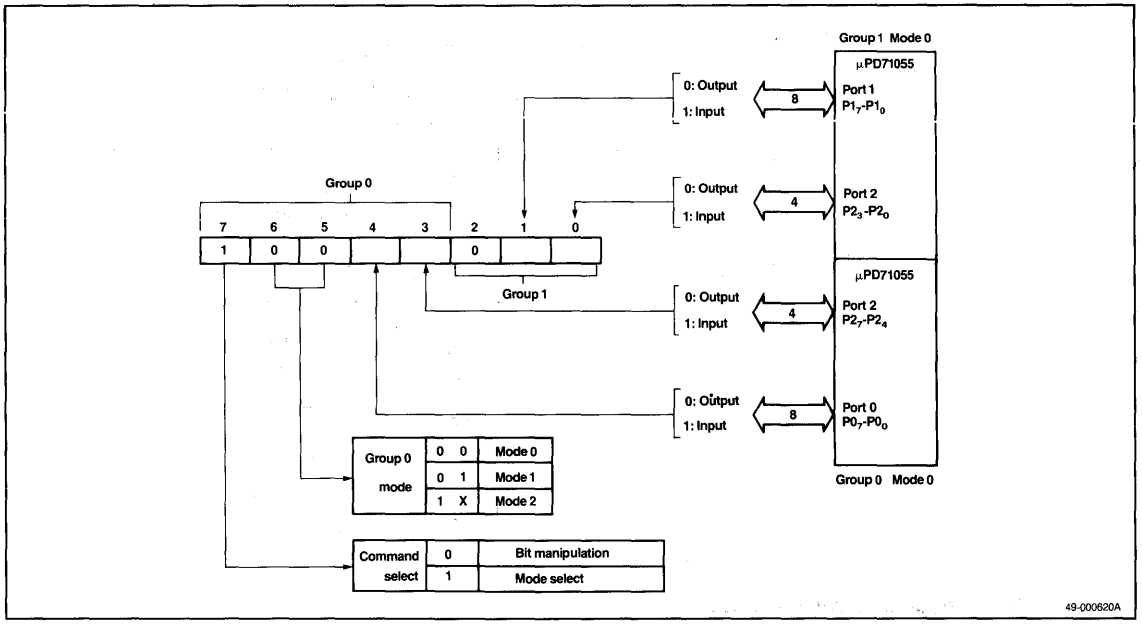


Figure 5. Mode 0 Input Timing

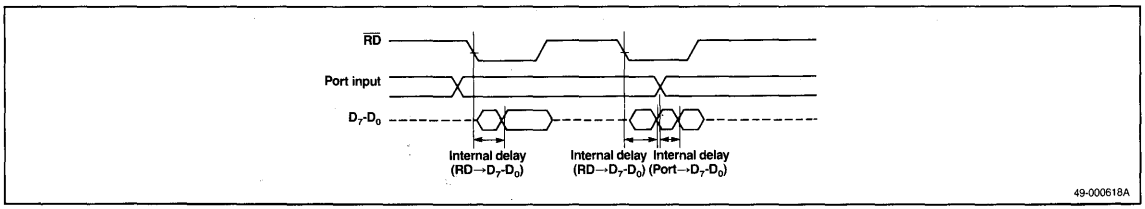


Figure 6. Mode 0 Output Timing

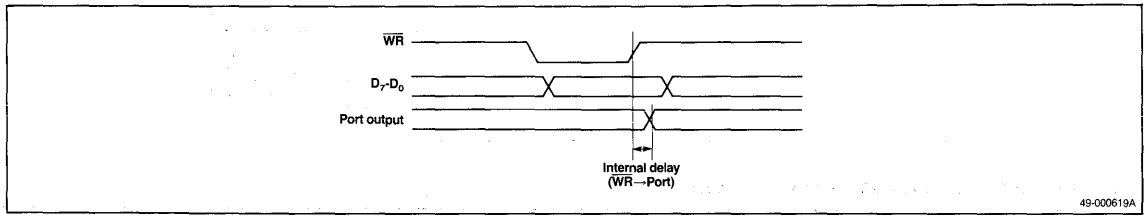


Figure 7. A/D Converter Connection Example

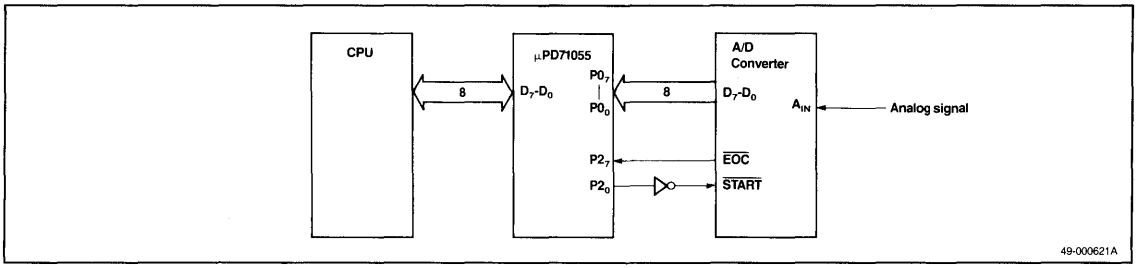


Figure 8. A/D Converter Example

```
READ_A/D:  MOV    AL,10011000B      ;μPD71055 Mode Setting:
           OUT    CTRLPORT,AL       ;Group 0, group 1 in mode 0
                                    ;Port 0 & port 2 (upper) are inputs
                                    ;Port 1 & port 2 (lower) are outputs
           MOV    AL,00000001B
           OUT    CTRLPORT,AL       ;Conversion starts by setting P20 high
WAIT_EOC:  IN     AL,PORT2          ;End of conversion wait loop
           TEST1  AL,7              ;Conversion ends when P27 = 0
           BNZ    WAIT_EOC
           IN     AL,PORT0          ;Read A/D converted values
           RET
```

5e

Mode 1

In this mode, the control and status signals control the I/O data. In group 0, port 0 functions as the data port and the upper five bits of port 2 function as control/status. In group 1, port 1 functions as the data port and the lower three bits of port 2 function as control/status.

In mode 1, the bit manipulation command is used to write the bits of port 2.

Group 0 Mode 1

When group 0 is used in mode 1, the upper five bits of port 2 become part of group 0. Of these five bits, three are used for control/status and the remaining two can be used for I/O (using the bit manipulation command). See figure 9.

Group 1 Mode 1

When group 1 is used in mode 1, the lower three or four bits of port 2 become part of group 1. Of these four bits, three are used for control/status. The remaining bit, $P2_3$, can be used for I/O only if group 0 is in mode 0. Otherwise, $P2_3$ belongs to group 0 as a control/status bit. See figure 9 and table 4.

Mode 1 Input Operation

In mode 1, port 0 is the data port for group 0, and port 1 for group 1. The control/status bits (port 2) are used as listed below. Figure 10 shows the signal timing.

$\overline{STB}$ [Strobe]. The data input at port 0 is latched in port 0 when the $\overline{STB0}$ input is brought low. The data input at port 1 is latched in port 1 by $\overline{STB1}$.

IBF [Input Buffer Full F/F]. The IBF output goes high to indicate that the input buffer has become full. IBF goes high when the $\overline{STB}$ signal goes low. IBF goes low at the rising edge of the $\overline{RD}$ signal when $\overline{STB} = 1$.

The IBF F/F is cleared when mode 1 is programmed.

INT [Interrupt Request]. INT goes high when the data is latched in the input port, when RIE is 1 and $\overline{STB}$, IBF and $\overline{RD}$ are all high. INT goes low at the falling edge of the $\overline{RD}$ signal. It can function as a data read request interrupt signal to a CPU.

INT is cleared when mode 1 is programmed.

Figure 9. Mode 1 Input

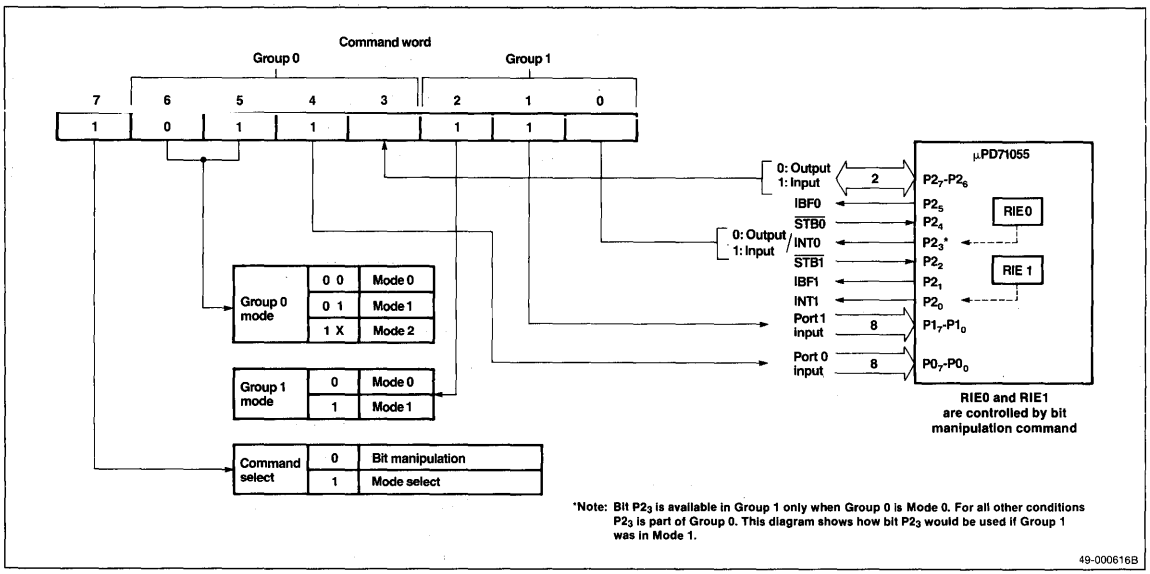
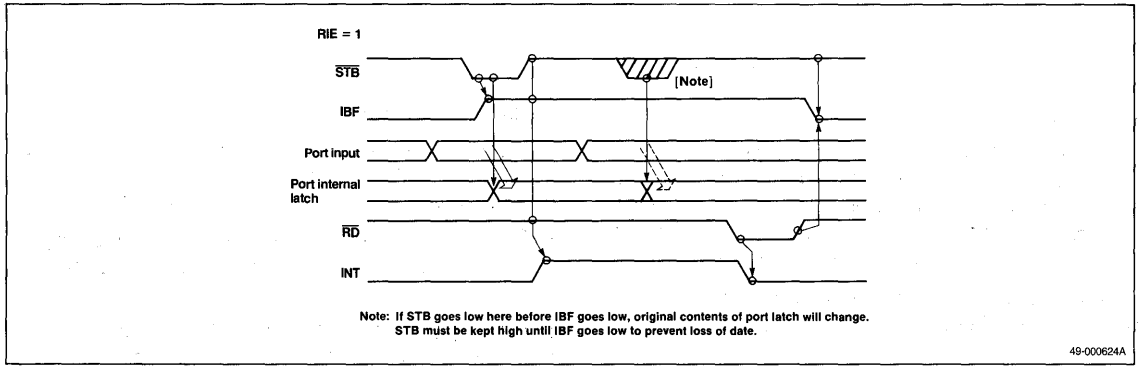


Figure 10. Mode 1 Input Timing



RIE [Read Interrupt Enable Flag]. RIE controls the interrupt output. Interrupts can be enabled by using the bit manipulation command to set this bit to 1, and disabled by resetting it to 0. This signal is internal to the μPD71055 and is not an output. The state of RIE does not affect the function of $\overline{STB0}$ or $\overline{STB1}$, which are inputs to the same bits ($P2_4$ and $P2_2$) of port 2.

When input is specified in mode 1, the status of IBF, INT and RIE can be read by reading the contents of port 2.

Mode 1 Output Operation

In mode 1 output operation (figure 11), the status/control bits (port 2) are used as listed below. Figure 12 shows the signal timing.

$\overline{OBF}$ [Output Buffer Full F/F]. $\overline{OBF}$ goes low when data is received by the μPD71055 and is latched in output ports 1 or 0. $\overline{OBF}$ functions as a data receive flag. $\overline{OBF}$ goes low at the rising edge of $\overline{WR}$ when $\overline{DAK}$ = 1 (write complete). It goes high when the $\overline{DAK}$ signal goes low.

Figure 11. Mode 1 Output

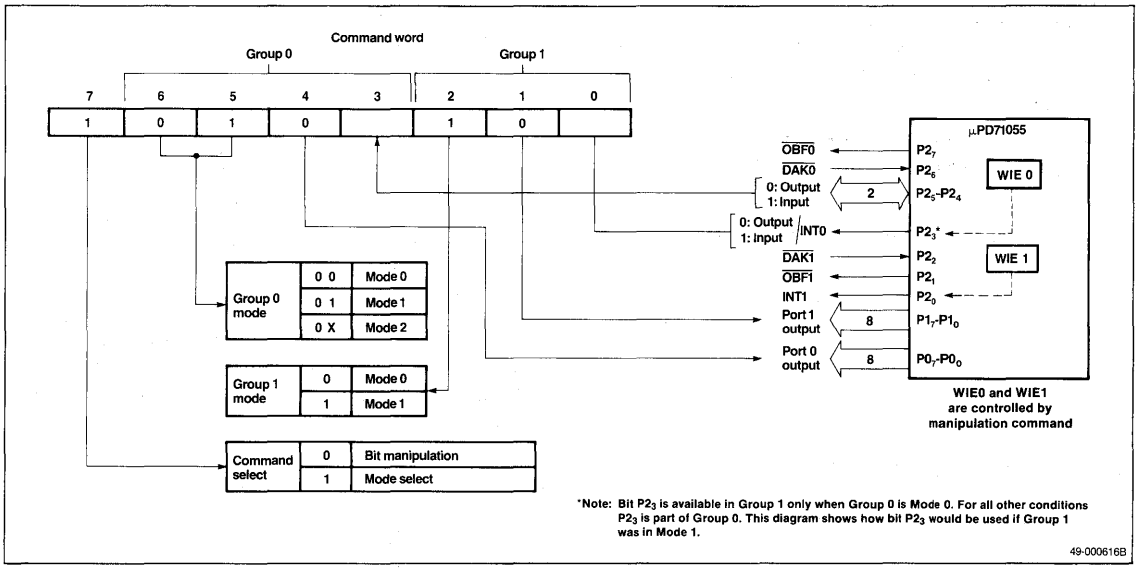
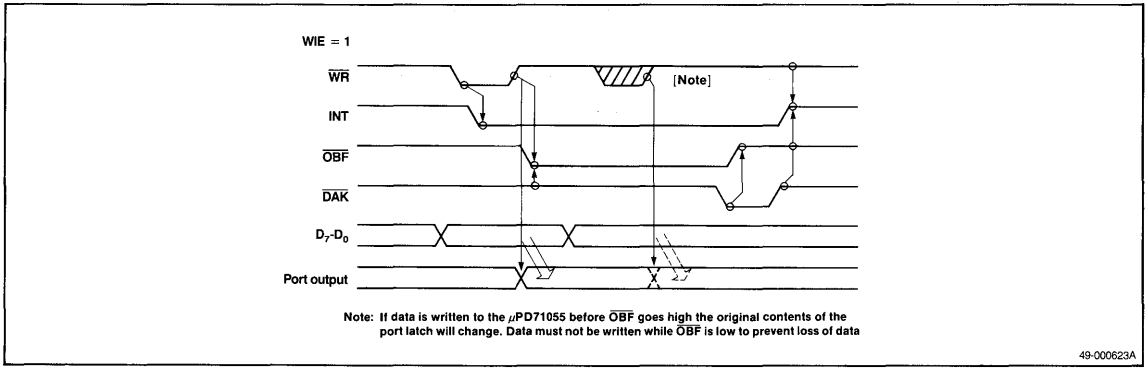


Figure 12. Mode 1 Output Timing



5e

$\overline{DAK}$ [Data Acknowledge]. When this input is low, it signals the μPD71055 that output port data has been taken from the 71055.

INT [Interrupt Request]. INT goes high when the output data is taken when WIE is set to 1 and $\overline{WR}$, $\overline{OBF}$ and $\overline{DAK}$ are all high. It goes low at the falling edge of the $\overline{WR}$ signal. INT therefore functions as a write request signal, indicating that new data should be sent to the μPD71055.

WIE [Write Interrupt Enable Flag]. WIE controls the interrupt output. Interrupts can be enabled by using the bit manipulation command to set this bit to 1 and disabled by resetting it to 0. This signal is internal to the μPD71055 and is not an output. The state of WIE does not affect the function of $\overline{DAK}$ addressed to the same bits of port 2.

When output is specified in mode 1, the status of $\overline{OBF}$, INT and WIE can be obtained by reading the contents of port 2.

Table 2 shows a summary of these signals.

Table 2. Functions of Port 2 Bits in Mode 1

Group	Bit	Data Input	Data Output
1	$P2_0$	INT1 (Interrupt request)	INT1 (Interrupt request)
	$P2_1$	IBF1 (Input buffer full f/f)	$\overline{OBF1}$ (Output buffer full f/f)
	$P2_2$	$\overline{STB1}$ (Strobe input) RIE1 (Read interrupt enable flag)	$\overline{DAK1}$ (Data acknowledge input) WIE1 (Write interrupt enable flag)
	$P2_3$	I/O (Note)	I/O (Note)
0	$P2_3$	INT0 (Interrupt request)	INT0 (Interrupt request)
	$P2_4$	$\overline{STB0}$ (Strobe input) RIE0 (Read interrupt enable flag)	I/O
	$P2_5$	IBF0 (Input buffer full f/f)	I/O
	$P2_6$	I/O	$\overline{DAK0}$ (Data acknowledge input) WIE0 (Write interrupt enable flag)
	$P2_7$	I/O	$\overline{OBF0}$ (Output buffer full f/f)

Note: Can be used with group 1 only when group 0 is set to mode 0. In other modes, $P2_3$ belongs to group 0.

Mode 1 Example

This example (figure 13) demonstrates connecting a printer to the μPD71055. Group 0 is used in mode 1 output. Group 1 can operate in mode 0 or 1; in this example it is set to mode 0.

Figure 13. Connection to Printer

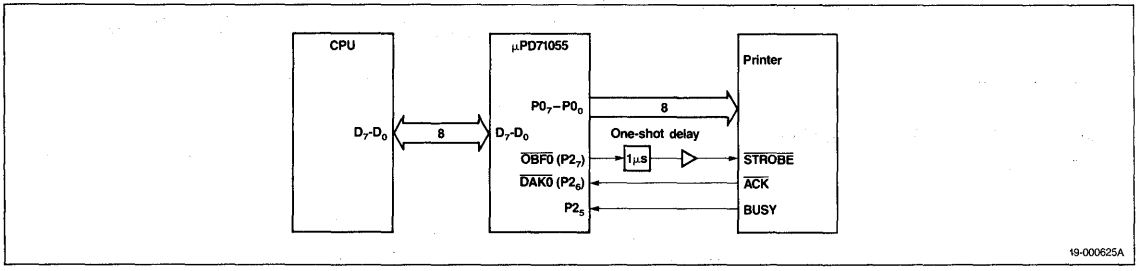


Figure 14. Printer Example Subroutine

```
;This subroutine sends character strings to the printer
INIT:       MOV     AL,10101000B        ;μPD71055 Mode Setting:
                                        ;Group 0: mode 1 output
                                        ;Group 1: mode 0
            OUT     CTRLPORT,AL
            RET
SENDPRN:    MOV     BW,DATA             ;Output data address
PRNLOOP:    MOV     AL,[BW]
            CMP     AL,0FFH             ;End if data = 0FFH
            BNZ     WAIT
            RET
WAIT:       IN      AL,PORT2
            TEST1   AL,7                ;Wait until output buffer is empty
            BZ      WAIT
            TEST1   AL,5                ;Wait until printer can accept data
            BNZ     WAIT
            MOV     AL,[BW]             ;Send data to printer
            OUT     PORT0,AL
            INC     BW
            BR      PRNLOOP
```

5e

Mode 2

Mode 2 can only be used by group 0. In this mode, port 0 functions as a bidirectional 8-bit data port operating under the control of the upper five bits of port 2 as control/status signals. In this mode, port 0 combines the input and output operations of mode 1. See figures 15 and 16.

In mode 2, the status of the following signals can be determined by reading port 2: $\overline{OBF0}$, IBF0, INT0, WIE0, and RIE0.

The $\overline{DAK0}$ and $\overline{STB0}$ signals are used to select input or output for port 0. By using these signals, bidirectional operation between the μPD71055 and peripheral can be realized.

In mode 2, the bit manipulation command is used to write to port 2.

Control/Status Port Operation

The following control/status signals are used for output:

$\overline{OBF0}$ [Output Buffer Full]. $\overline{OBF0}$ goes low when data is received from the D_0-D_7 data bus and is latched in the port 0 output buffer. It therefore functions as a receive request signal to the peripheral. $\overline{OBF0}$ goes low at the rising edge of the $\overline{WR0}$ signal (end of data write). It goes high when $\overline{DAK0}$ is low (output data from port 0 received).

$\overline{DAK0}$ [Data Acknowledge]. $\overline{DAK0}$ is sent to the μPD71055 in response to the $\overline{OBF0}$ signal. It should be set low when data is received from port 0 of the μPD71055.

WIE0 [Write Interrupt Enable Flag]. WIE0 controls the write interrupt request output. Interrupts are enabled by using the bit manipulation command to set this bit to 1 and disabled by setting it to 0. The state of WIE does not affect the $\overline{DAK}$ function of this pin.

The following control/status signals are used for input:

$\overline{STB0}$ [Strobe Input]. When $\overline{STB0}$ goes low, the data being sent to the μPD71055 is latched in port 0.

IBF0 [Input Buffer Full F/F]. When IBF0 goes high, it indicates that the input buffer is full. It functions as a signal which can be used to prohibit further data transfer. IBF0 goes high when $\overline{STB0}$ goes low. It goes low at the rising edge of $\overline{RD0}$ when $\overline{STB0}$ = 1 (read complete).

Figure 15. Mode 2

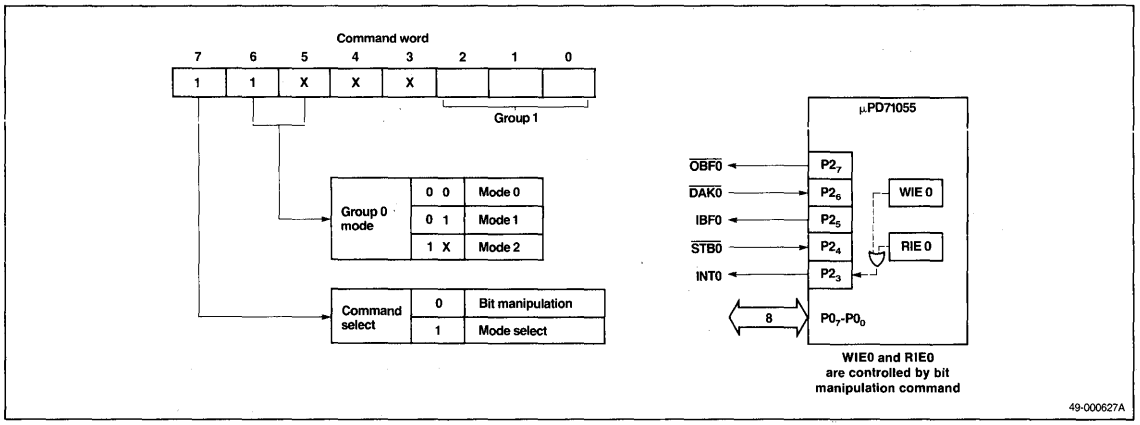
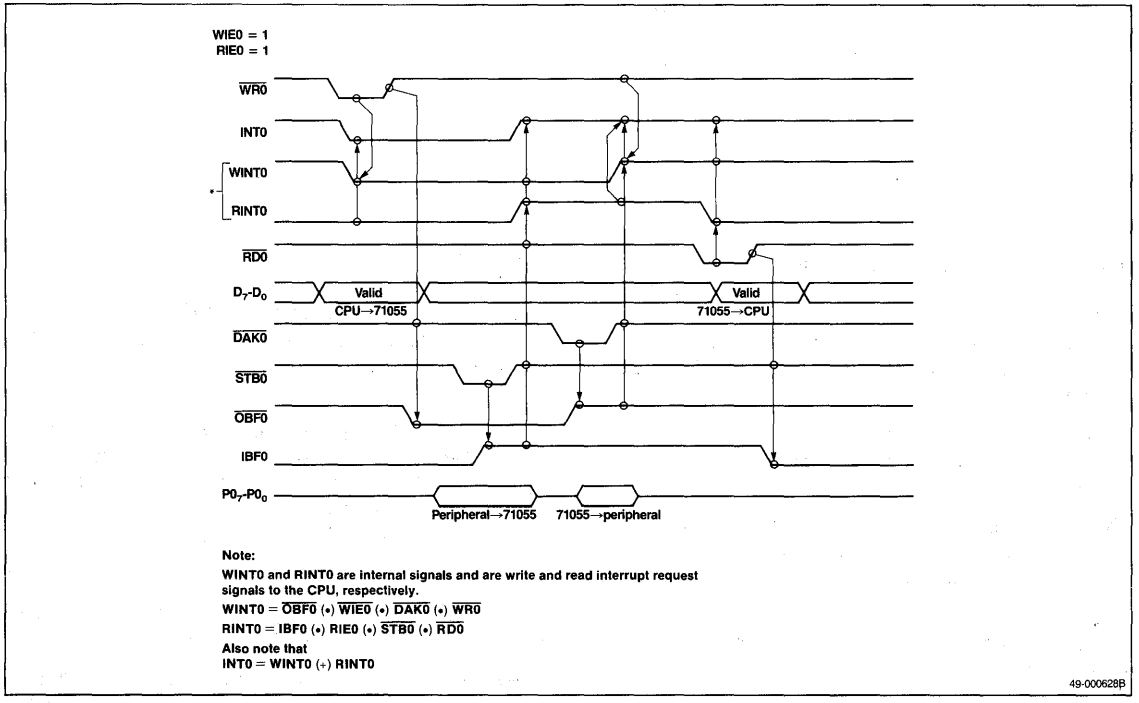


Figure 16. Mode 2 Timing



RIE0 [Read Interrupt Enable Flag]. RIE0 controls the read interrupt request output. Interrupts are enabled by using the bit manipulation command to set this bit to 1 and disabled by setting it to 0. The state of RIE0 does not affect the $\overline{STB0}$ function of this pin.

This control/status signal is used for both input and output:

INT0 [Interrupt Request]. During input operations, INT0 functions as a read request interrupt signal. During output, it functions as a write request interrupt signal. This signal is the logical OR of the INT signal for data read (RINT0) and the INT signal for write (WINT0) in mode 1 (RINT0 OR WINT0).

In mode 2, the status of $\overline{OBF0}$, IBF0, INT0, WIE0, and RIE0 can be determined by reading port 2.

Table 3 is a summary of these signals.

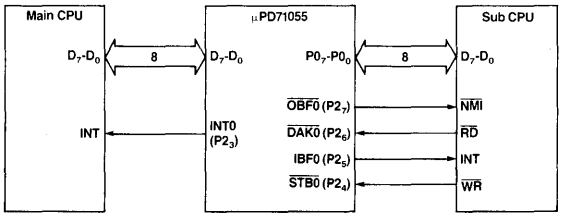
Table 3. Functions of Port 2 in Mode 2

Bit	Function
$P2_3$	INT0 (Interrupt request)
$P2_4$	$\overline{STB0}$ (Strobe input) RIE0 (Read interrupt enable flag)
$P2_5$	IBF0 (Input buffer full f/f)
$P2_6$	$\overline{DAK0}$ (Data acknowledge input) WIE0 (Write interrupt enable flag)
$P2_7$	$\overline{OBF0}$ (Output buffer full f/f)

Mode 2 Example

Figures 17, 18, and 19 show data transfer between two CPUs.

Figure 17. Connecting Two CPUs



5e

Figure 18. Main CPU Flowchart

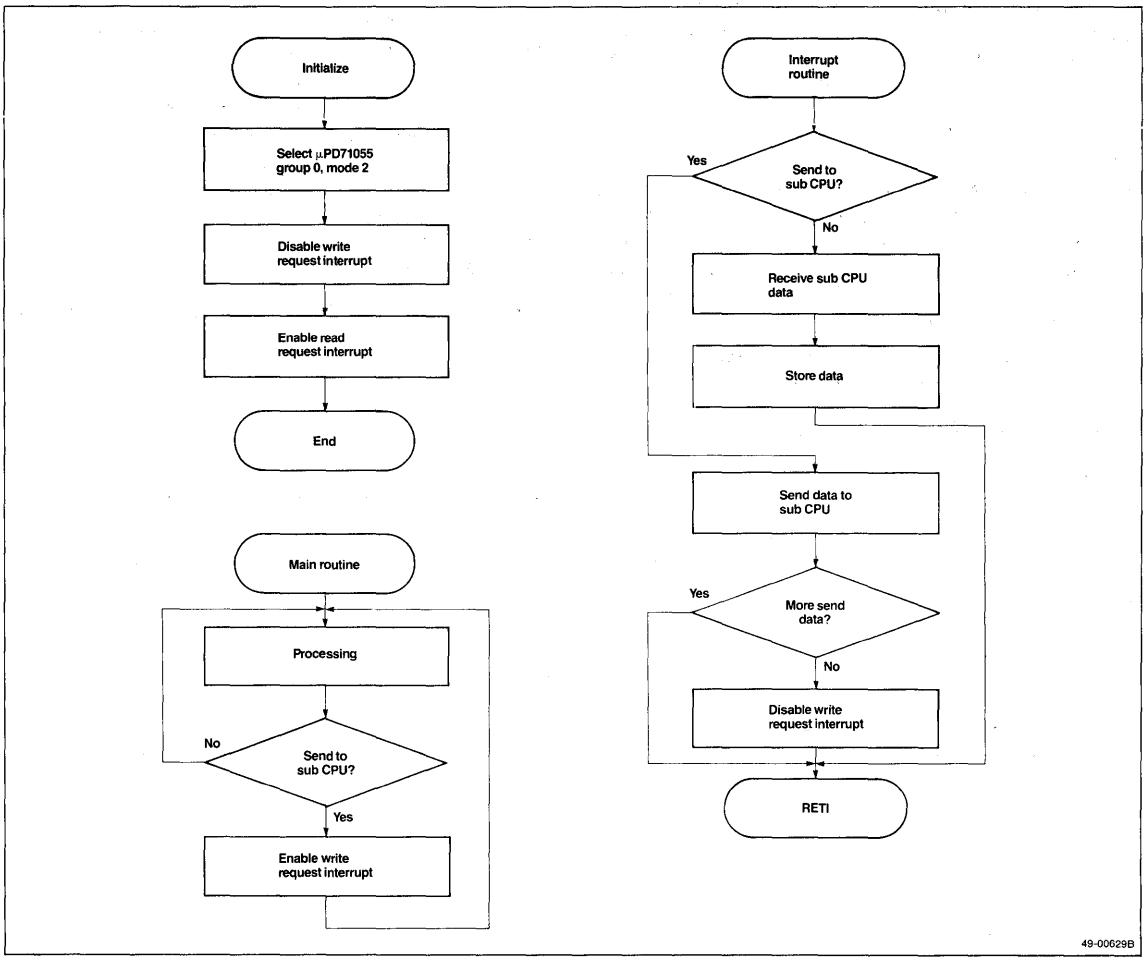
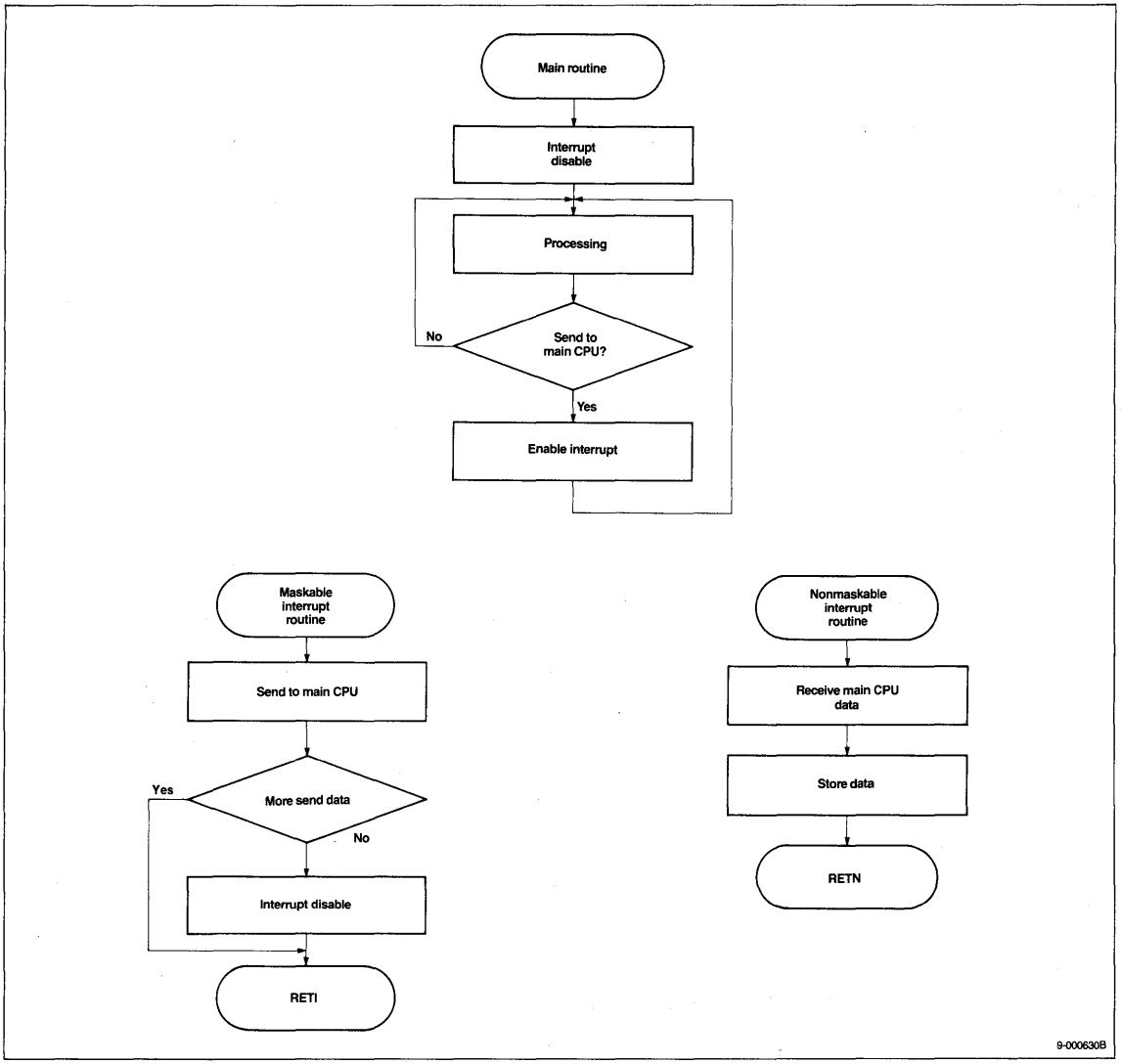


Figure 19. Sub CPU Flowchart



Mode Combinations

Table 4 is a complete list of all the combinations of modes and groups, and the function of the port 2 bits in each mode.

Table 4. Mode Combinations and Port 2 Bit Functions

Group 0							Group 1					
Mode	$P0_7$-$P0_0$	$P2_7$	$P2_6$	$P2_5$	$P2_4$	$P2_3$	Mode	$P1_7$-$P1_0$	$P2_3$	$P2_2$	$P2_1$	$P2_0$
0	In	D	D	D	D	NA	0	In	D	D	D	D
0	In	D	D	D	D	NA	0	Out	D	D	D	D
0	In	D	D	D	D	NA	1	In	B	$\overline{STB1}$ (RIE1)	IBF1	INT1
0	In	D	D	D	D	NA	1	Out	B	$\overline{DAK1}$ (WIE1)	$\overline{OBF1}$	INT1
0	Out	D	D	D	D	NA	0	In	D	D	D	D
0	Out	D	D	D	D	NA	0	Out	D	D	D	D
0	Out	D	D	D	D	NA	1	In	B	$\overline{STB1}$ (RIE1)	IBF1	INT1
0	Out	D	D	D	D	NA	1	Out	B	$\overline{DAK1}$ (WIE1)	$\overline{OBF1}$	INT1
1	In	B	B	IBF0	$\overline{STB0}$ (RIE0)	INT0	0	In	NA	D	D	D
1	In	B	B	IBF0	$\overline{STB0}$ (RIE0)	INT0	0	Out	NA	D	D	D
1	In	B	B	IBF0	$\overline{STB0}$ (RIE0)	INT0	1	In	NA	$\overline{STB1}$ (RIE1)	IBF1	INT1
1	In	B	B	IBF0	$\overline{STB0}$ (RIE0)	INT0	1	Out	NA	$\overline{DAK1}$ (WIE1)	$\overline{OBF1}$	INT1
1	Out	$\overline{OBF0}$	$\overline{DAK0}$ (WIE0)	B	B	INT0	0	In	NA	D	D	D
1	Out	$\overline{OBF0}$	$\overline{DAK0}$ (WIE0)	B	B	INT0	0	Out	NA	D	D	D
1	Out	$\overline{OBF0}$	$\overline{DAK0}$ (WIE0)	B	B	INT0	1	In	NA	$\overline{STB1}$ (RIE1)	IBF1	INT1
1	Out	$\overline{OBF0}$	$\overline{DAK0}$ (WIE0)	B	B	INT0	1	Out	NA	$\overline{DAK1}$ (WIE1)	$\overline{OBF1}$	INT1
2	I/O	$\overline{OBF0}$	$\overline{DAK0}$ (WIE0)	IBF0	$\overline{STB0}$ (RIE0)	INT0	0	In	NA	D	D	D
2	I/O	$\overline{OBF0}$	$\overline{DAK0}$ (WIE0)	IBF0	$\overline{STB0}$ (RIE0)	INT0	0	Out	NA	D	D	D
2	I/O	$\overline{OBF0}$	$\overline{DAK0}$ (WIE0)	IBF0	$\overline{STB0}$ (RIE0)	INT0	1	In	NA	$\overline{STB1}$ (RIE1)	IBF1	INT1
2	I/O	$\overline{OBF0}$	$\overline{DAK0}$ (WIE0)	IBF0	$\overline{STB0}$ (RIE0)	INT0	1	Out	NA	$\overline{DAK1}$ (WIE1)	$\overline{OBF1}$	INT1

Note:

(1) In this chart, "NA" indicates that the bit cannot be used by this group.

(2) The symbol "B" indicates bits that can only be rewritten by the bit manipulation command.

(3) In this chart, "D" indicates that is used by the user.

(4) Symbols in parentheses are internal flags. They are not output to port 2 pins and they cannot be read by the host.

(5) In indicates Input, Out indicates Output, and I/O indicates Input/Output.

Description

The μPD71059 is a low-power CMOS programmable interrupt control unit for microcomputer systems. It can process eight interrupt request inputs, allocating a priority level to each one. It transfers the interrupt with the highest priority to the CPU, along with interrupt address information. By cascading up to eight slave μPD71059s to a master μPD71059, a system can process up to 64 interrupt requests. System scale, interrupt routine address, interrupt request priority, and masking are all under complete program control.

Features

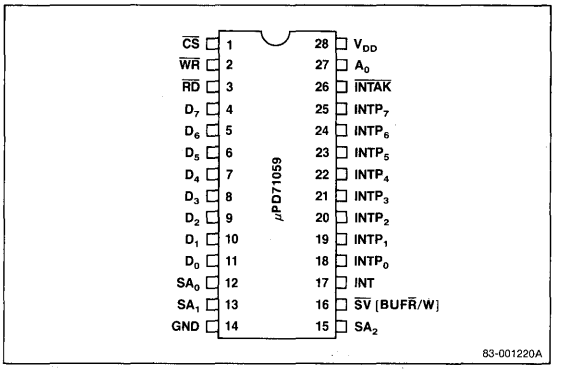
- ☐ μPD8085A compatible (CALL mode)
- ☐ μPD70108/70116 compatible (vector mode)
- ☐ Eight interrupt request inputs per chip
- ☐ Up to 64 interrupt request inputs per system (extended mode)
- ☐ Edge- or level-triggered interrupt request inputs
- ☐ Each interrupt maskable
- ☐ Programmable priority level
- ☐ Polling operation
- ☐ Single +5 V ±10% power supply
- ☐ Industrial temperature range: −40 to +85 °C
- ☐ CMOS technology
- ☐ 8 MHz and 10 MHz

Ordering Information

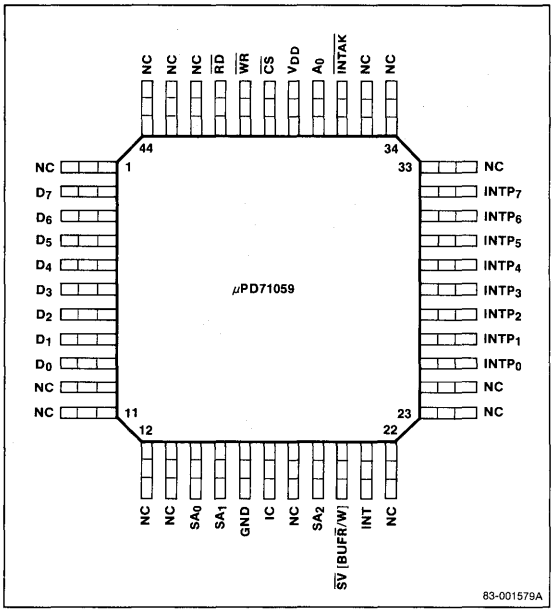
Part Number	Package
μPD71059C-8	28-pin plastic DIP (600 mil)
C-10	
G-8	44-pin plastic QFP (P44G-80-22)
GB-8	44-pin plastic QFP (P44GB-80-3B4)
GB-10	
L-8	28-pin PLCC
L-10	

Pin Configurations

28-Pin Plastic DIP



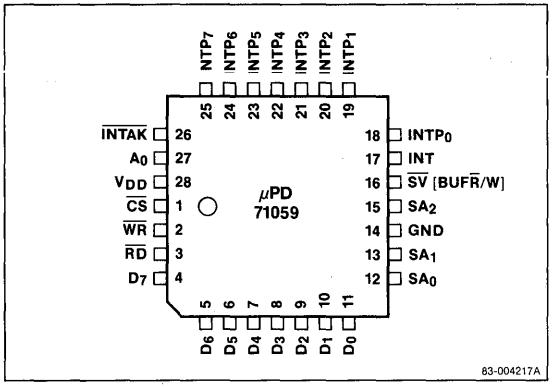
44-Pin Plastic QFP



5f

Pin Configurations (cont)

28-Pin Plastic Leaded Chip Carrier (PLCC)



Pin Identification

Symbol	Function
D_7-D_0	Data bus I/O
SA_2-SA_0	Slave address I/O, bits 2, 1, 0
GND	Ground potential
IC	Internally connected
$\overline{SV}$ (BUF$\overline{R}$/W)	Slave (Buffer read write) I/O
INT	Interrupt output
$INTP_0$-$INTP_7$	Interrupt inputs
$\overline{INTAK}$	Interrupt acknowledge input
A_0	Address input
V_{DD}	Power supply
$\overline{CS}$	Chip select input
$\overline{WR}$	Write strobe input
$\overline{RD}$	Read strobe input
NC	Not connected

Pin Functions

D_7-D_0 [Data Bus]

The 8-bit 3-state bidirectional bus transfers data to and from the CPU through the system bus. The data bus becomes active when data is sent to the CPU in the $\overline{INTAK}$ sequence. Otherwise, the data bus is high impedance.

$\overline{CS}$ [Chip Select]

The CPU uses the μPD71059's $\overline{CS}$ input to select a μPD71059 to read from (IN instructions) or write to (OUT instructions). The $\overline{RD}$ and $\overline{WR}$ signals to the μPD71059 are enabled when $\overline{CS}$ is low. $\overline{CS}$ is not used for the INTAK sequence.

$\overline{RD}$ [Read Strobe]

The CPU sets the $\overline{RD}$ input to 0 when reading the internal registers IMR, IRR and ISR, and during polling operations to read polling data.

$\overline{WR}$ [Write Strobe]

The CPU sets the $\overline{WR}$ input to 0 when writing initializing words IW1-IW4 and command words IMW, PFCW and MCW.

A_0 [Address]

The A_0 input is used with $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ to read or write to the μPD71059. Normally, A_0 is connected to A_0 of the address bus. Table 1 shows the relationship between read/write operations and the control signals ($\overline{CS}$, $\overline{WR}$, $\overline{RD}$, and A_0).

$INTP_7$-$INTP_0$ [Interrupt Request from Peripheral]

$INTP_7$-$INTP_0$ are eight asynchronous interrupt request inputs. They can be set to be either edge-or level-triggered. These pins are pulled up by an internal resistance. Their power consumption is lower at high-level input than at low-level input.

INT [Interrupt]

INT is the interrupt request output from a μPD71059 to the CPU or master μPD71059. When an interrupt from a peripheral is input to an INTP pin and acknowledged, the μPD71059 asserts INT high to generate an interrupt request at the CPU or master μPD71059.

$\overline{INTAK}$ [Interrupt Acknowledge]

The $\overline{INTAK}$ input from the CPU acknowledges an interrupt from the μPD71059. After acknowledging the interrupt request, the CPU returns three low-level pulses (μPD8085) or two low-level pulses (μPD70108/70116). Synchronizing to these pulses, the μPD71059 sends a CALL instruction in three bytes, or an interrupt vector number in one byte through the data bus.

$\overline{SV}$ [BUF$\overline{R}$/W] [Slave, Buffer Read/Write]

This pin has two functions. When no external buffer is used in the data bus, it is the $\overline{SV}$ input. When $\overline{SV}$ is low, the μPD71059 acts as a slave. It operates as a master when $\overline{SV}$ is high. $\overline{SV}$ has no master/slave meaning when the μPD71059 is set to single mode.

As the BUF$\overline{R}$/W output, this pin can allow a bus transceiver to be controlled by the μPD71059, if one is required. When the μPD71059 changes its data bus to output, it sets BUF$\overline{R}$/W low. It sets BUF$\overline{R}$/W high when the data bus changes to input.

SA_2-SA_0 [Slave Address]

These pins are only used in systems with cascaded μPD71059s. The master μPD71059 uses these pins to address up to eight slave μPD71059s. These pins are output pins for masters, and input pins for slaves.

Note: In the single mode, SA_2-SA_0 are output pins, but the output data has no meaning.

V_{DD} [Power Supply]

This is the positive power supply.

GND [Ground]

This is the ground potential.

IC [Internally Connected]

This pin must be left unconnected.

Table 1. Read/Write Operations

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_0	Other Conditions	μPD71059 Operation	CPU Operation
0	0	1	0	IRR set by MCW	IRR to Data bus	IRR read
				ISR set by MCW	ISR to Data bus	ISR read
				Polling phase (Note 1)	Polling data to Data bus	Polling
0	0	1	1		IMR to Data bus	IMR read
0	1	0	0	$D_4 = 1$	Data bus to IW1 register	IW1 write
				$D_4, D_3 = 0$	Data bus to PFCW register	PFCW write
				$D_4 = 0, D_3 = 1$	Data bus to MCW register	MCW write
0	1	0	1	(Note 2)	Data bus to IW2 register	IW2 write
					Data bus to IW3 register	IW3 write
					Data bus to IW4 register	IW4 write
				After initializing	Data bus to IMR	IMW write
0	1	1	x		Data bus high impedance	
1	x	x	x			
0	0	0	x		Illegal	

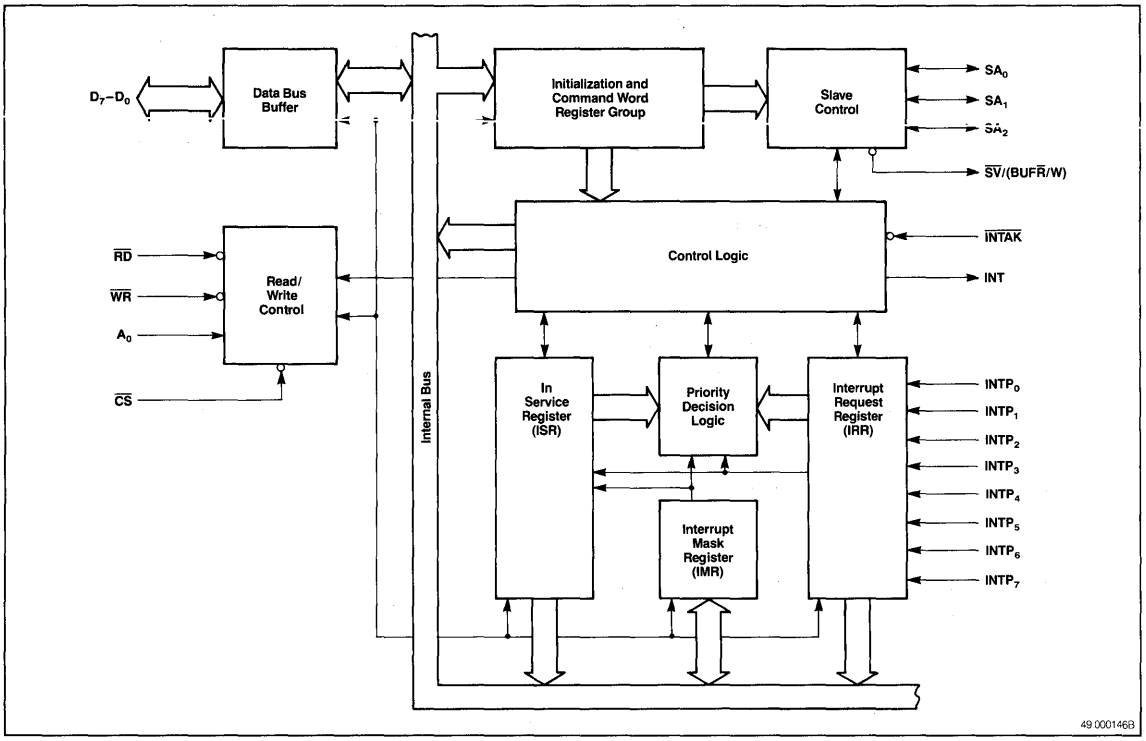
Note:

(1) In the polling phase, polling data is read instead of IRR and ISR.

(2) Refer to Control Words section for IW2-IW4 writing procedure.

5f

Block Diagram



Block Diagram Functions

Data Bus Buffer

The data bus buffer is a buffer between D_7-D_0 and the μPD71059's internal bus.

Read/Write Control

The read/write control controls the CPU's reading and writing to and from the μPD71059 registers.

Initialization and Command Word Registers

These registers store initializing words IW1-IW4 and command words PFCW (priority and finish control word) and MCW (mode control word). The CPU cannot read these registers.

Interrupt Mask Register [IMR]

The interrupt mask register stores the interrupt mask word (IMW) command word. Each bit masks an interrupt. If bit n of this register is 1, the interrupt request $INTP_n$ is masked and cannot be accepted by the μPD71059. The CPU can read this register by performing an IN instruction with $A_0 = 1$.

Interrupt Request Register [IRR]

The interrupt request register shows which interrupt levels are currently being requested. If bit n of the IRR is 1, $INTP_n$ is requesting an interrupt. The CPU can read this register.

In-Service Register [ISR]

The in-service register shows all interrupt levels currently in service. If bit n of this register is 1, the interrupt routine corresponding to $INTP_n$ is currently being executed. The CPU can read this register.

Slave Control

Slave control is used in systems with cascaded μPD71059s. A master μPD71059 uses it to control slave μPD71059s, and a slave uses it to interface with the master μPD71059.

Control Logic

The control logic receives and generates the signals that control the sequence of events in an interrupt.

Priority Decision Logic

The priority decision logic determines which interrupt request from the IRR will be serviced next. The decision is made based upon the current interrupt mask, interrupt service status, mode status, and current priority.

Absolute Maximum Ratings

$T_A = 25°C$

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.3 V
Output voltage, V_O	−0.5 to V_{DD} + 0.3 V
Power dissipation, PD_{MAX}	500 mW
Operating temperature, T_{opt}	−40 to +85°C
Storage temperature, T_{stg}	−65 to +150°C

Comment: Exposing the device to stresses above those listed in the absolute maximum ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational section of this specification. Exposure to absolute maximum ratings for extended periods may affect device reliability.

Capacitance

$T_A = 25°C$; $V_{DD} = GND = 0$ V

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Input capacitance	C_I			10	pF	$f_c = 1$ MHz
I/O capacitance	C_{IO}			20	pF	Unmeasured pins returned to 0 V

DC Characteristics

$T_A = -40$ to $+85°C$; $V_{DD} = 5$ V ±10%

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Input voltage, high	V_{IH}	2.2		V_{DD} + 0.3	V	
Input voltage, low	V_{IL}	−0.5		0.8	V	
Output voltage, high	V_{OH}	0.7 x V_{DD}			V	$I_{OH} = -400$ μA
Output voltage, low	V_{OL}			0.4	V	$I_{OL} = 2.5$ mA
Input leakage current, high	I_{LIH}			10	μA	$V_I = V_{DD}$
Input leakage current, low	I_{LIL}			−10	μA	$V_I = 0$ V
Output leakage current, high	I_{LOH}			10	μA	$V_O = V_{DD}$
Output leakage current, low	I_{LOL}			−10	μA	$V_O = 0$ V
INTP input leakage current, high	I_{LIPH}			10	μA	$V_I = V_{DD}$
INTP input leakage current, low	I_{LIPL}			−300	μA	$V_I = 0$ V
Supply current (dynamic)						
μPD71059	I_{DD1}		3.5	9	mA	
μPD71059-10	I_{DD1}		4	9	mA	
Supply current (power down mode)	I_{DD2}		2	50	μA	Input pins: $V_{IH} = V_{DD} - 0.1$ V, $V_{IL} = 0.1$ V, Output pins: open (Note 1)

Notes:

(1) In power down mode, $INTP_0$-$INTP_7$, $\overline{INTAK}$, and $\overline{CS}$ must be at high level ($V_{IH} = V_{DD} - 0.1$ V).

5f

AC Characteristics

$T_A = -40$ to $+85°C$; $V_{DD} \pm 5$ V + 10%

Parameter	Symbol	8 MHz Limits Min	8 MHz Limits Max	10 MHz Limits Min	10 MHz Limits Max	Unit	Test Conditions
Read Timing							
A_0, $\overline{CS}$ setup to $\overline{RD}$ ↓	t_{SAR}	0		0		ns	
A_0, $\overline{CS}$ hold from $\overline{RD}$ ↑	t_{HRA}	0		0		ns	
$\overline{RD}$ pulse width low	t_{RRL}	160		120		ns	
$\overline{RD}$ pulse width high	t_{RRH}	120		90		ns	
Data delay from $\overline{RD}$ ↓	t_{DRD}		120		95	ns	$C_L = 150$ pF
Data float from $\overline{RD}$ ↑	t_{FRD}	10	85	10	60	ns	$C_L = 100$ pF
Data delay from A_0, $\overline{CS}$	t_{DAD}		200		120	ns	$C_L = 150$ pF
$BUF\overline{R}/W$ delay from $\overline{RD}$ ↓	t_{DRBL}		100		80	ns	
$BUF\overline{R}/W$ delay from $\overline{RD}$ ↑	t_{DRBH}		150		100	ns	
Write Timing							
A_0, $\overline{CS}$ setup to $\overline{WR}$ ↓	t_{SAW}	0		0		ns	
A_0, $\overline{CS}$ hold from $\overline{WR}$ ↑	t_{HWA}	0		0		ns	
$\overline{WR}$ pulse width low	t_{WWL}	120		100		ns	
$\overline{WR}$ pulse width high	t_{WWH}	120		90		ns	
Data setup from $\overline{WR}$ ↑	t_{SDW}	120		100		ns	
Data hold from $\overline{WR}$ ↑	t_{HWD}	0		0		ns	
Interrupt Timing							
INTP pulse width	t_{IPIPL}	100		80		ns	(Note 1)
SA setup to second, third $\overline{INTAK}$ ↓	t_{SSIA}	40		40		ns	Slave
$\overline{INTAK}$ pulse width low	t_{IAIAL}	160		120		ns	
$\overline{INTAK}$ pulse width high	t_{IAIAH}	120		90		ns	$\overline{INTAK}$ Sequence
INT delay from INTP ↑	t_{DIPI}		300		200	ns	$C_L = 150$ pF
SA delay from first $\overline{INTAK}$ ↓	t_{DIAS}		360		250	ns	Master, $C_L = 150$ pF
Data delay from $\overline{INTAK}$ ↓	t_{DIAD}		120		95	ns	$C_L = 150$ pF
Data float from $\overline{INTAK}$ ↑	t_{FIAD}	10	85	10	60	ns	
Data delay from SA	t_{DSD}		200		150	ns	Slave, $C_L = 150$ pF
$BUF\overline{R}/W$ delay from $\overline{INTAK}$ ↓	t_{DIABL}		100		80	ns	$C_L = 150$ pF
$BUF\overline{R}/W$ delay from $\overline{INTAK}$ ↑	t_{DIABH}		150		100	ns	
Other Timing							
Command recovery time	t_{RV1}	120		90		ns	(Note 2)
$\overline{INTAK}$ recovery time	t_{RV2}	250		90		ns	(Note 3)
$\overline{INTAK}$/command recovery time	t_{RV3}	250		90		ns	(Note 4)

Notes:

(1) The time to clear the input latch in edge-trigger mode.

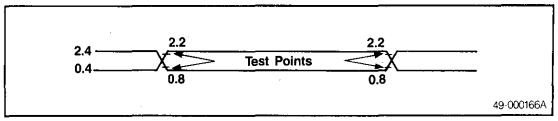
(2) The time to move from read to write operation.

(3) The time to move to the next $\overline{INTAK}$ operation.

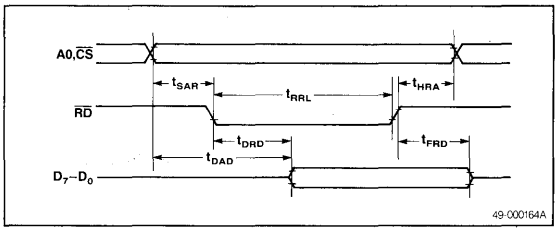
(4) The time to move $\overline{INTAK}$ to/from command (read/write).

Timing Waveforms

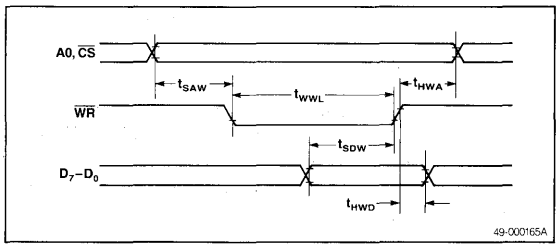
AC Test Input/Output Waveform



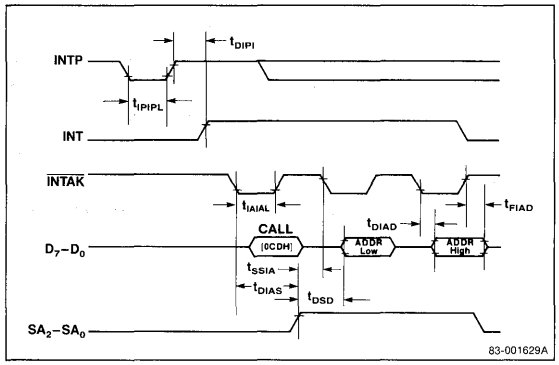
Read Cycle



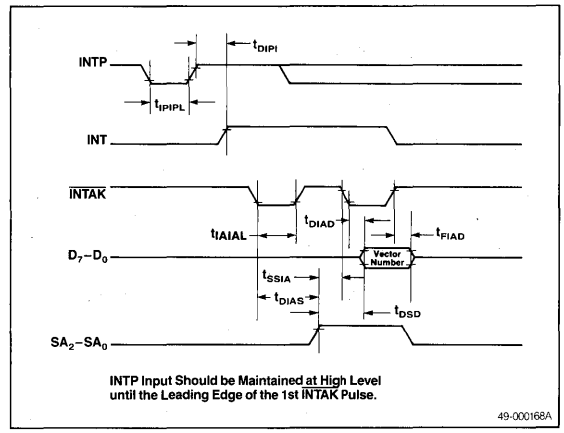
Write Cycle



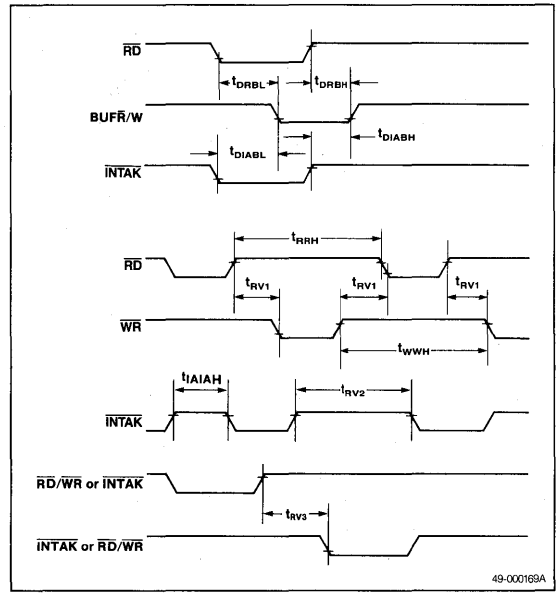
INTAK Sequence (CALL Mode)



INTAK Sequence (Vector Mode)



Other Timing



5f

Interrupt Operation

Almost all microcomputer systems use interrupts to reduce software overhead when controlling peripherals. However, the number of interrupt pins on a CPU is limited. When the number of interrupt lines increases beyond that limit, external circuits like the μPD71059 become necessary.

The μPD71059 can process eight interrupt request according to an allocated priority order and transmit the signal with the highest priority to the CPU. It also supplies the CPU with information to ascertain the interrupt routine start address. Cascading μPD71059s by connecting up to eight "slave" μPD71059s to a single "master" μPD71059 permits expansion to up to a maximum of 64 interrupt request signals.

Interrupt system scale (master/slave), interrupt routine addresses, interrupt request priority, and interrupt request masking are all programmable, and can be set by the CPU.

Normal interrupt operation for a single μPD71059 is as follows. First, the initialization registers are set with a sequence of initialization words. When the μPD71059 detects an interrupt request from a peripheral to an INTP pin it sets the corresponding bit of the interrupt request register (IRR). The interrupt is checked against the interrupt mask register (IMR) and the interrupt service register (ISR). If the interrupt is not masked and there is no other interrupt with a higher priority in service or requesting service, it generates an INT signal to the CPU.

The CPU acknowledges the interrupt by bringing the $\overline{\text{INTAK}}$ line low. The μPD71059 then outputs interrupt CALL or vector data onto the data bus in response to $\overline{\text{INTAK}}$ pulses. During the last $\overline{\text{INTAK}}$ pulse, the μPD71059 sets the corresponding bit in its ISR to indicate that this interrupt is in service and to disable interrupts with lower priority. It resets the bit in the IRR at this point. When the CPU has finished processing the interrupt, it will inform the μPD71059 by sending a finish interrupt (FI) command. This resets the bit in the ISR and allows the μPD71059 to accept interrupts with lower priorities. If the μPD71059 is in the self-FI mode, the ISR bit is reset automatically and this step is not necessary.

Software Features

The μPD71059 has the following software features:

- Interrupt types: CALL/vector
- Interrupt masking: Normal/extended nesting
- End of interrupt: Self-FI/normal FI/ specific FI
- Priority rotation: Normal nested/extended nested/exceptional nested Automatic priority rotation Rotate to specific priority
- Polled mode
- CPU-readable registers

Hardware Configurations

The μPD71059 has the following hardware configurations:

- Interrupt input: Edge/level sensitive
- Cascading μPD71059s: Single/extended (master/slave)
- Output driver control: Buffered/non-buffered

Mode Control

These features and configurations are selected and controlled by the four initialization words (IW1-IW4) and the three command words (IMW, PFCW, and MCW). The format of these words are shown in figures 2 and 3, respectively.

Control Words

There are two types of μPD71059 control words: initialization words and command words.

There are four initialization words: IW1-IW4. These words must be written to the μPD71059 at least once to initialize it. They must be written in sequence.

There are three types of command words: interrupt mask word (IMW), priority and finish control word (PFCW), and the mode control word (MCW). These words can be written freely after initialization.

Initialization Words

Initialization sequence. When data is written to a μPD71059 after setting $A_0 = 0$ and $D_4 = 1$, data is always accepted as IW1. This results in a default initialization as shown below. See figure 1.

(1) The edge-trigger circuit of the INTP input is reset. IRR is cleared in the edge-trigger mode.

(2) ISR and IMR are cleared.

(3) $INTP_7$ receives the lowest priority; $INTP_0$ receives the highest.

(4) The exceptional nesting mode is released. IRR is set as the register to be read.

(5) Register IW4 is cleared. The normal nesting mode, non-buffer mode, FI command mode, and CALL mode are set.

Initialization Words. The initialization words are written consecutively, and in order. The first two, IW1 and IW2, set the interrupt address or vector. IW3 specifies which interrupts are slaves for master systems, and defines the slave number of a slave system. Therefore, IW3 is only required in extended systems. The μPD71059 will only expect it if bit D_1 of IW1, SNGL = 0. IW4 is only written if bit D_0 of IW1, I4 = 1. See figure 2 for the format of the initialization words.

Command Words

The command words give various commands to a μPD71059 during its operation to change interrupt masks and priorities, to end interrupt processing, etc. See figure 3.

IMW [Interrupt Mask Word]. This word masks the IRR and disables the corresponding INTP interrupt requests. It also masks the ISR in the exceptional nesting mode. Bits M_7-M_0 correspond to the interrupt levels of $INTP_7$-$INTP_0$, respectively.

In the exceptional nesting mode, interrupts corresponding to the bits of IRR and ISR are masked if the M_n bit is set to 1.

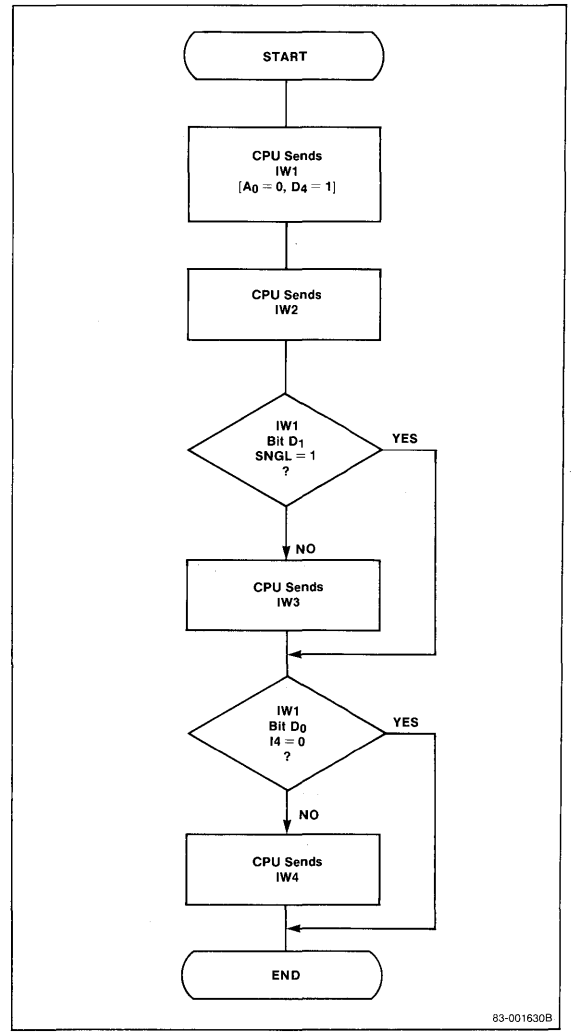
PFCW [Priority and Finish Control Word]. This word sets the FI (finish interrupt) command that defines the way that interrupts are ended, and the commands that change interrupt request priorities.

When RP (rotate priority) is set to 1, the priorities of the interrupt requests change (rotate). The priority order of the 8 INTP pins is as shown in figure 4. Setting a level as the lowest priority sets all the other levels correspondingly. For example, if $INTP_3$ is the lowest priority, $INTP_4$ will be the highest. ($INTP_7$ has the lowest priority after initialization).

SIL (specify interrupt level) is set to 1 to change the priority order or designate an interrupt level. It is used with the RP and FI bits (bits D_7 and D_5). When SIL = 1 and RP or FI = 1, the level identified by IL_2-IL_0 is designated as the lowest priority level. The other priorities will be set correspondingly. When used with FI = 1, it resets the ISR bit corresponding to the interrupt level IL_2-IL_0.

MCW [Mode Control Word]. This word is used to set the exceptional nesting mode, to poll the μPD71059, and to read the ISR and IRR registers.

Figure 1. Initialization Sequence



5f

Bits SR and IS/$\overline{IR}$ are used to read the contents of the IRR and ISR registers. When SR = 0, no operation is performed. To read IRR or ISR, set $A_0 = 0$ and select the IRR or ISR register by writing to MCW. To select the IRR register, write MCW with SR = 1 and IS/$\overline{IR}$ = 0. To select the ISR, write MCW with SR = 1 and IS/$\overline{IR}$ = 1. The selection is retained, and MCW does not have to be rewritten to read the same register again. IRR and ISR are not masked by the IMR.

Figure 2. Initialization Word Formats (Sheet 1 of 2)

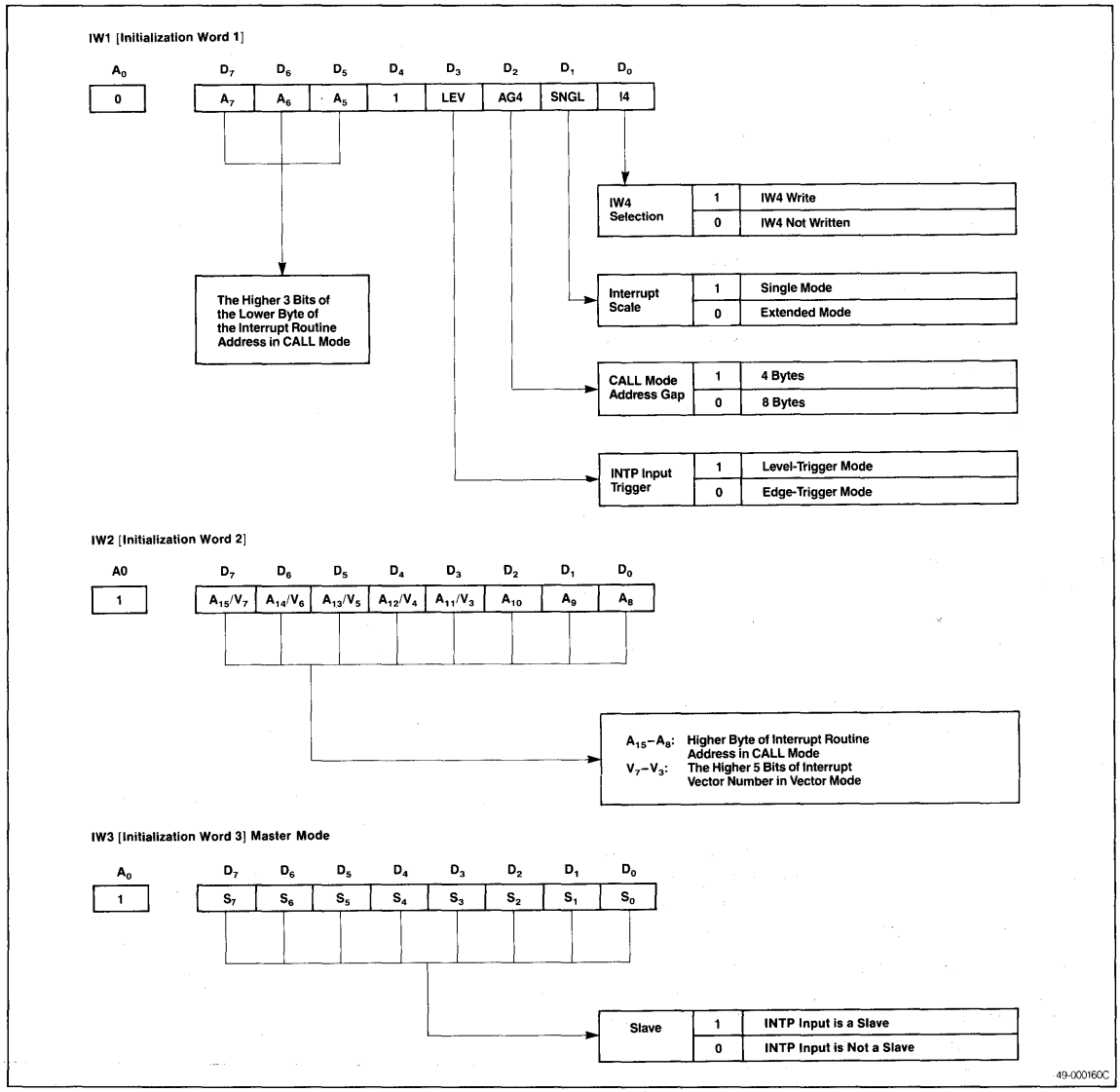


Figure 2. Initialization Word Formats (Sheet 2 of 2)

IW3 [Initialization Word 3] Slave Mode

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	SN_2	SN_1	SN_0

Slave Number	0	0	0	0
	0	0	1	1
	0	1	0	2
	0	1	1	3
	1	0	0	4
	1	0	1	5
	1	1	0	6
	1	1	1	7

IW4 [Initialization Word 4]

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	EXTN	BUF	$\overline{BSV}$	SFI	$V/\overline{C}$

Interrupt Mode Select	1	Vector Mode
	0	CALL Mode

FI Mode	1	Self-FI Mode
	0	FI Command Mode

	BUF	$\overline{BSV}$	
Buffer Mode Select	0	X	Non-Buffer Mode
	1	1	Buffer Mode Master
	1	0	Buffer Mode Slave

Extended Nest	1	Extended Nest Mode
	0	Normal Nest Mode

49-000161C

5f

Figure 3. Command Word Format

IMW [Interrupt Mask Word]

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

Interrupt Mask	1	Set Interrupt Mask
	0	Reset Interrupt Mask

PFCW [Priority Finish and Control Word]

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	RP	SIL	FI	0	0	IL_2	IL_1	IL_0

Internal Level	0	0	0	0
	0	0	1	1
	0	1	0	2
	0	1	1	3
	1	0	0	4
	1	0	1	5
	1	1	0	6
	1	1	1	7

Priority Rotation and FI Command	0	0	1	FI Command	No Level Designation	Without Rotation	Normal FI Command
	1	0	1			With Rotation	Normal Rotation FI Command
	0	1	1		With Level Designation	Without Rotation	Specific FI Command
	1	1	1			With Rotation	Specific Rotation FI Command
	0	1	0	Non-FI Command		Without Rotation	No Operation
	1	1	0			With Rotation	Specific Rotation Command
	0	0	0		No Level Designation	Without Rotation	Self-FI Mode Rotation Reset
	1	0	0			With Rotation	Self-FI Mode Rotation Set

MCW [Mode Control Word]

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	X	SNM	EXCN	0	1	POL	SR	IS/$\overline{IR}$

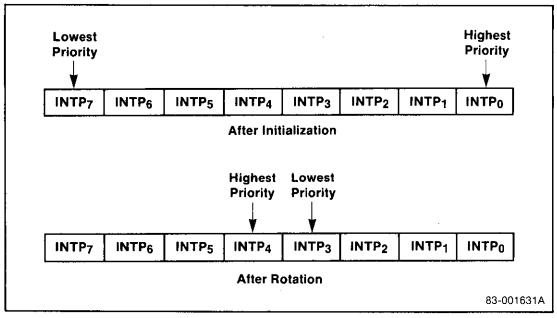
Polling	1	Polling Command
	0	No Operation

Read Register Select	0	X	No Operation
	1	0	ISR Select
	1	1	IRR Select

Exceptional Nesting Mode	0	X	No Operation
	1	1	Exceptional Nesting Mode Set
	1	0	Exceptional Nesting Mode Release

49-000162C

Figure 4. INTP Priority Order



CALL or Vector Modes

The μPD71059 passes interrupt routine address data to the CPU in two modes, depending on the CPU type. This mode is set by bit V/$\overline{C}$ in initialization word IW4. V/$\overline{C}$ is set to one to to select the vector mode for μPD70108/70116 CPUs, and reset to zero to select the CALL mode for μPD8085A CPUs.

CALL Mode [μPD8085A CPUs]

In this mode, when an interrupt is acknowledged by the CPU, the μPD71059 outputs three bytes of interrupt data to the data bus in its $\overline{INTAK}$ sequence. During the first $\overline{INTAK}$ pulse from the CPU, the μPD71059 outputs the CALL opcode 0CDH. During the next $\overline{INTAK}$ pulse, it outputs the lower byte of a two-byte interrupt routine address. During the third $\overline{INTAK}$ pulse, it outputs the upper byte of the address. The CPU interprets these three bytes as a CALL instruction and executes the CALL interrupt routine. See figure 5 and the $\overline{INTAK}$ sequence (CALL mode) μPD8085 diagram in the AC Timing Waveforms.

Interrupt routine addresses are set using words IW1 and IW2 during initialization. However, only the higher ten or eleven bits of the interrupt addresses are set, A_{15}-A_6 or A_{15}-A_5. The μPD71059 sets the remaining low bits (D_5-D_0 or D_4-D_0) to get the address of $INTP_n$'s interrupt routine. The addresses for $INTP_1$-$INTP_7$ are set in order of interrupt level. The space between interrupt addresses is determined by setting the AG4 bit (address gap 4 bytes) of IW1. When AG4 = 1, the interrupt routine starting addresses are 4 bytes apart. Therefore, the starting address for $INTP_n$ is the starting address for $INTP_0$ plus four times n. When AG4 = 0, starting addresses are eight bytes apart, so the starting address for $INTP_n$ is the starting address for $INTP_0$ plus eight times n. See figure 6.

Vector Mode [μPD70108/70116 CPUs]

In the vector mode, the μPD71059 outputs a one-byte interrupt vector number to the data bus in the $\overline{INTAK}$ sequence. The CPU uses that vector number to generate an interrupt routine address. See figure 7.

The higher five bits of the vector number, V_7-V_3, are set by IW2 during initialization. The μPD71059 sets the remaining three bits to the number of the interrupt input (0 for $INTP_0$, 1 for $INTP_1$, etc). See figure 8.

The CPU generates an interrupt vector by multiplying the vector number by four, and using the result as the address of a location in an interrupt vector table located at addresses 000H-3FFH. See figure 9.

System Scale Modes

The μPD71059 can operate in either single mode, with up to eight interrupt lines or extended mode, with more than one μPD71059 and more than eight interrupt lines. In extended mode a μPD71059 is in either master or slave mode.

Bit D_1, SNGL (single mode), of the first initialization word IW1 designates the scale of the interrupt system. SNGL = 1 designates that only one μPD71059 is being used (single mode system). SNGL = 0 designates an extended mode system with a master and slave μPD71059s. In the single mode (SNGL = 1), the SV input and IW4 buffer mode bits D_3 and D_2 do not indicate a master/slave relation for the μPD71059.

5f

Single Mode

This mode is the normal mode of μPD71059 operation. It has been described in the Interrupt Operation description. See figure 10 for a system example.

Extended Mode

In this mode, up to 64 interrupt requests can be processed using a master (μPD71059 in master mode) connected to a maximum of eight slaves (μPD71059s in slave mode). See figure 11 for a system example.

Figure 5. CALL Mode Interrupt Sequence

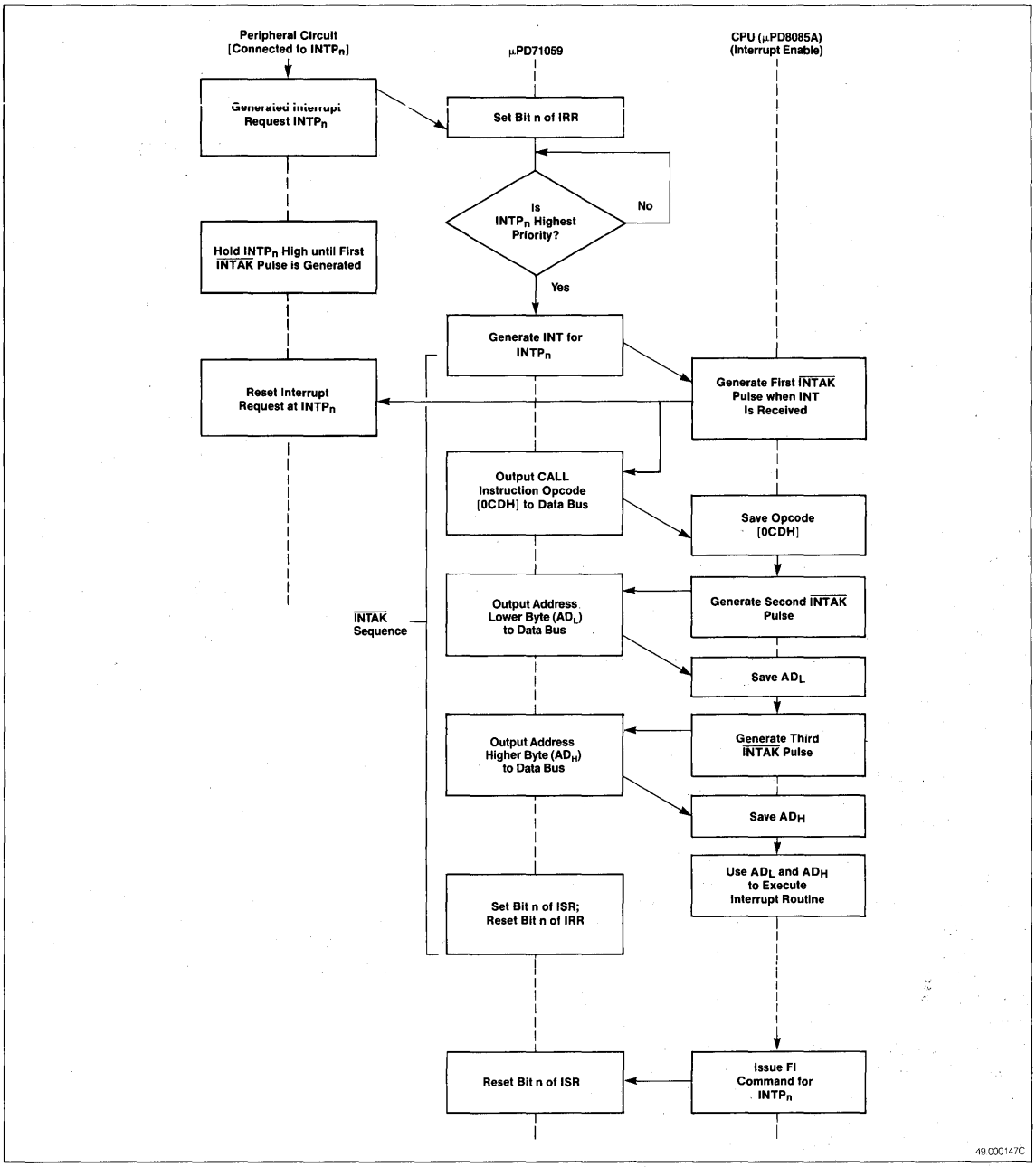


Figure 6. CALL Mode Interrupt Address Sequence

• Address Lower Byte [AD_L] During Second $\overline{INTAK}$

AG4 = 1 (4-Byte Spacing Address)

Interrupt Level	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$INTP_0$	A_7	A_6	A_5	0	0	0	0	0
$INTP_1$	A_7	A_6	A_5	0	0	1	0	0
$INTP_2$	A_7	A_6	A_5	0	1	0	0	0
$INTP_3$	A_7	A_6	A_5	0	1	1	0	0
$INTP_4$	A_7	A_6	A_5	1	0	0	0	0
$INTP_5$	A_7	A_6	A_5	1	0	1	0	0
$INTP_6$	A_7	A_6	A_5	1	1	0	0	0
$INTP_7$	A_7	A_6	A_5	1	1	1	0	0

AG4 = 0 (8-Byte Spacing Address)

Interrupt Level	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$INTP_0$	A_7	A_6	0	0	0	0	0	0
$INTP_1$	A_7	A_6	0	0	1	0	0	0
$INTP_2$	A_7	A_6	0	1	0	0	0	0
$INTP_3$	A_7	A_6	0	1	1	0	0	0
$INTP_4$	A_7	A_6	1	0	0	0	0	0
$INTP_5$	A_7	A_6	1	0	1	0	0	0
$INTP_6$	A_7	A_6	1	1	0	0	0	0
$INTP_7$	A_7	A_6	1	1	1	0	0	0

Note: When AG4 = 0, bit A_5 is ignored.

• Address Higher Byte [AD_H] During Third $\overline{INTAK}$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

83-001632A

5f

Figure 7. Vector Mode Interrupt Sequence

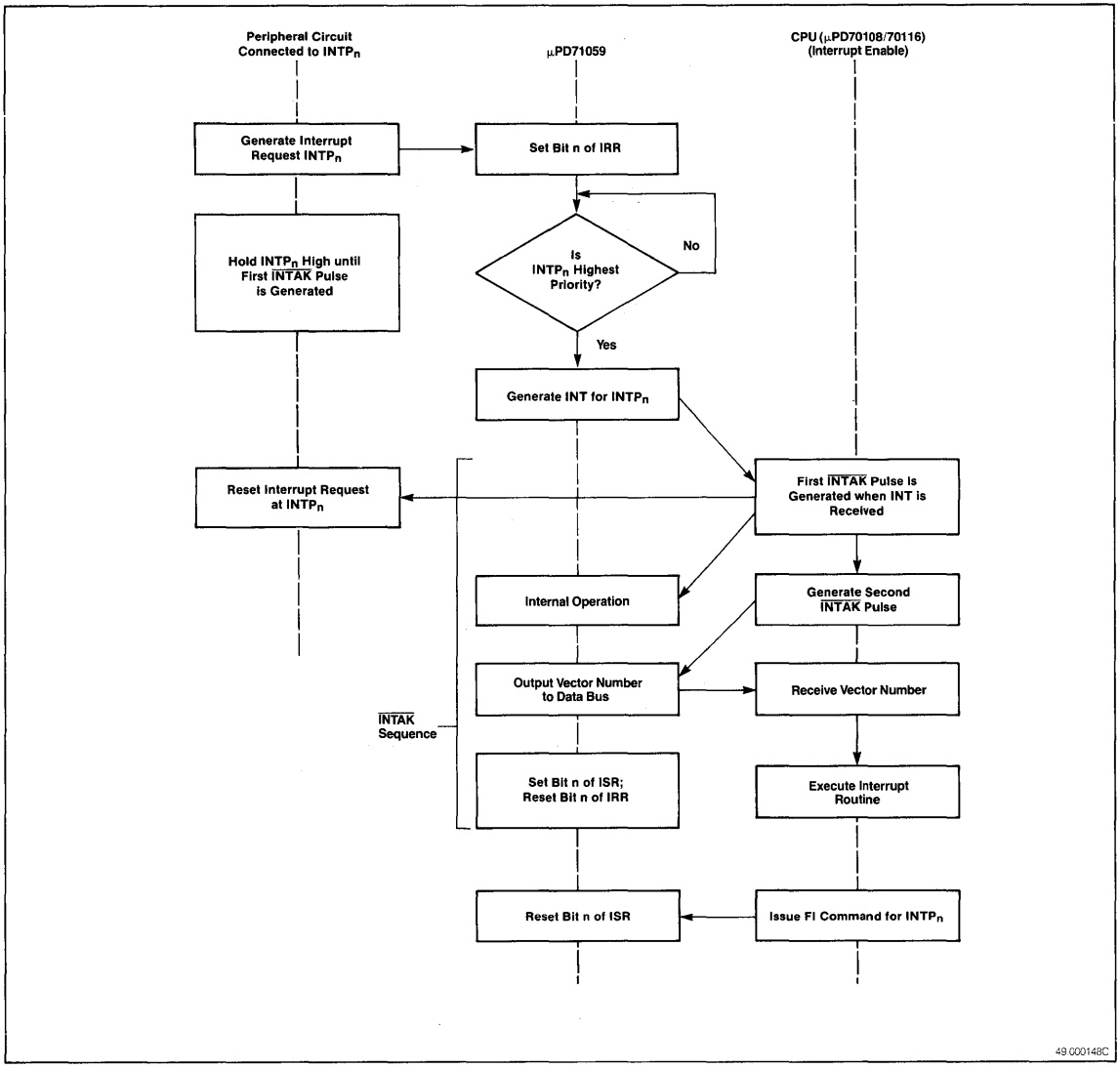


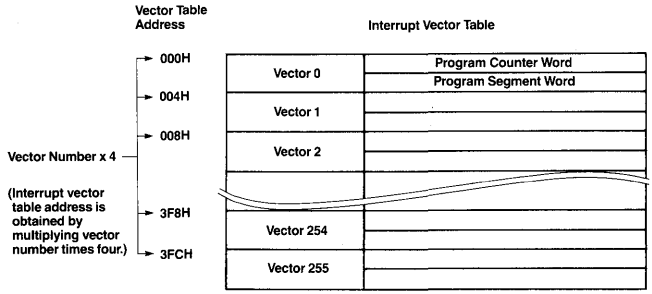
Figure 8. Vector Numbers Output in Vector Mode

Output During the Second $\overline{INTAK}$

Interrupt Levels	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$INTP_0$	V_7	V_6	V_5	V_4	V_3	0	0	0
$INTP_1$	V_7	V_6	V_5	V_4	V_3	0	0	1
$INTP_2$	V_7	V_6	V_5	V_4	V_3	0	1	0
$INTP_3$	V_7	V_6	V_5	V_4	V_3	0	1	1
$INTP_4$	V_7	V_6	V_5	V_4	V_3	1	0	0
$INTP_5$	V_7	V_6	V_5	V_4	V_3	1	0	1
$INTP_6$	V_7	V_6	V_5	V_4	V_3	1	1	0
$INTP_7$	V_7	V_6	V_5	V_4	V_3	1	1	1

83-001633B

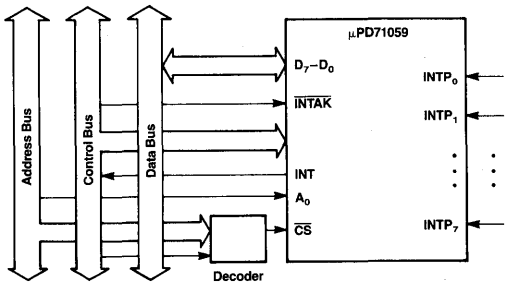
Figure 9. Interrupt Vectors for the μPD70108/70116



49-000152A

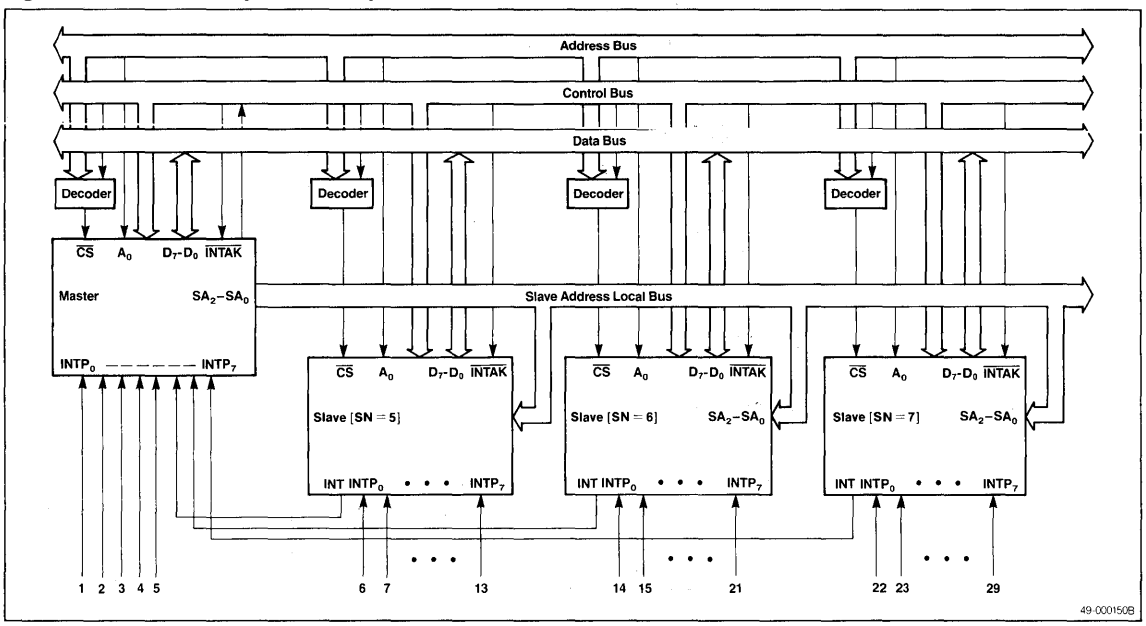
5f

Figure 10. Single Mode System



49.000149A

Figure 11. Extended System Example with Three Slaves



Master Mode

When a μPD71059 is a master in an extended mode system, S_7-S_0 of IW3 (master mode) define which of $INTP_7$-$INTP_0$ are inputs from slave μPD71059s or peripheral interrupts.

Consider an interrupt request from $INTP_n$. If $S_n = 0$, the interrupt is from a peripheral (for example, $INTP_0$ of the master μPD71059 in Figure 11), and the μPD71059 treats it the same way it would if it were in the single mode. SA_2-SA_0 outputs are low level and the master provides the interrupt address or vector number.

If $S_n = 1$, the interrupt is from a slave (for example, $INTP_7$ of the master). The master sends an interrupt to the CPU if the slave requesting the interrupt has priority. The master then outputs slave address n to pins SA_2-SA_0 on the first $\overline{INTAK}$ pulse by the CPU. It lets slave n perform the rest of the $\overline{INTAK}$ sequence.

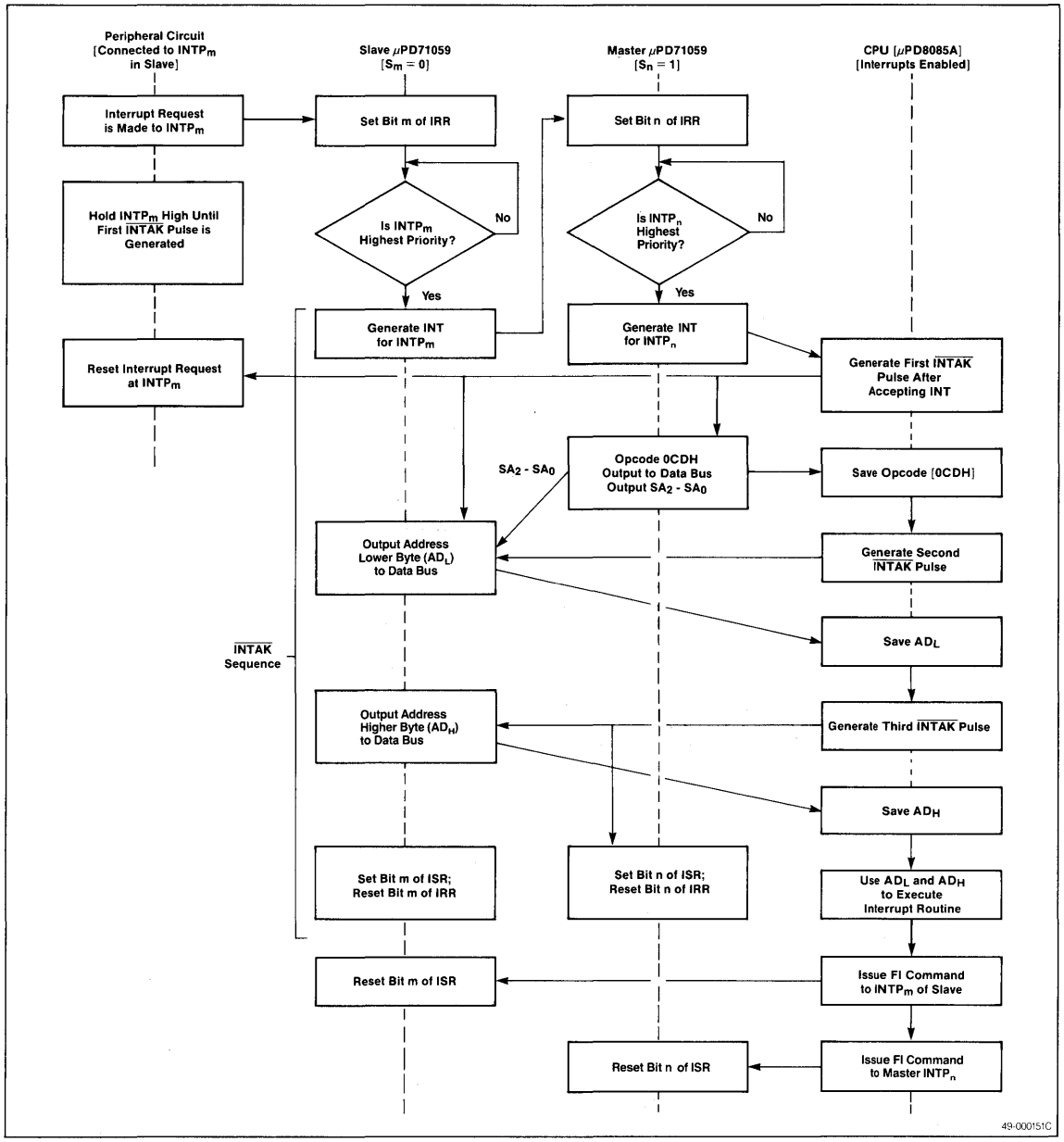
Slave Mode

When a slave receives an interrupt request from a peripheral, and the slave has no interrupts with higher priority in service, it sends an interrupt request to the master through its INT output. When the interrupt is accepted by the CPU through the master, the master outputs the slave's address on pins SA_2-SA_0. Each slave compares the address on SA_2-SA_0 to its own address. The slave that sent the interrupt will find a match. It completes the $\overline{INTAK}$ sequence the same way as a single μPD71059 would.

The master outputs slave address 0 when it is processing a non-slave interrupt. Therefore, do not use 0 as a slave address if there are less than eight slaves connected to the master.

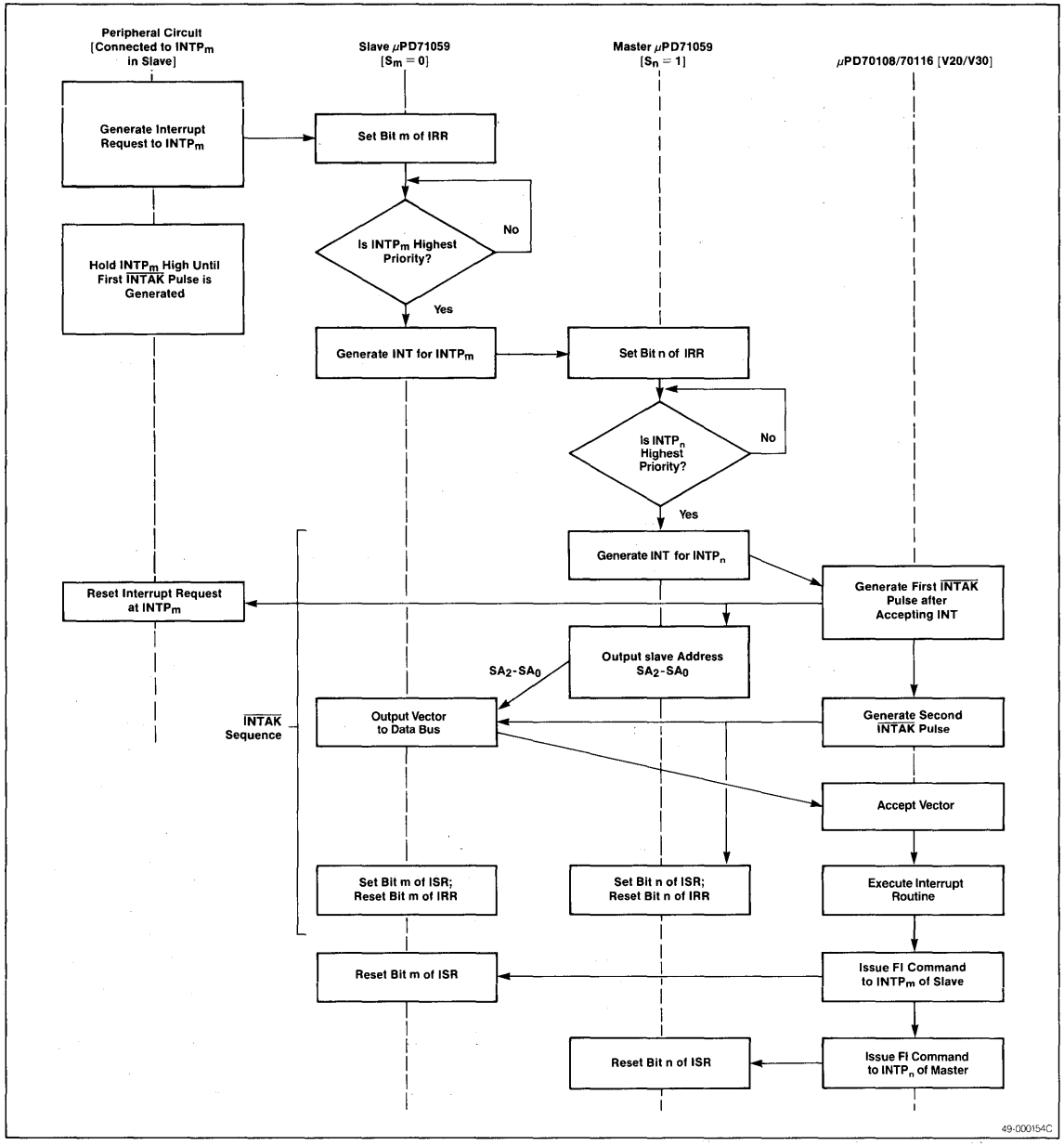
Figures 12 and 13 show the interrupt operating sequences for slaves in the extended mode.

Figure 12. Interrupt from Slave (CALL Mode)



5f

Figure 13. Interrupt from Slave (Vector Mode)



Buffer and Non-Buffer Modes

In a large system, a buffer may be needed by the μPD71059 to drive the data bus. A buffer mode is supplied, with a signal to specify the buffer direction. In the buffer mode, $\overline{SV}$ (BUF$\overline{R}$/W) is used to select the buffer direction and $\overline{SV}$ cannot be used to specify the master/slave mode. The master/slave selection must be set by IW4. IW4 bit D_3, BUF (buffer) and D_2, B$\overline{SV}$ (buffered slave) are used together to set the buffer mode and master/slave relation. When BUF = 0, the non-buffer mode is set and B$\overline{SV}$ has no meaning. When BUF = 1, the buffer mode is set. In buffer mode, the μPD71059 is a master when B$\overline{SV}$ = 1, a slave when BSV = 0. See figure 14.

Nesting Modes

The way a μPD71059 handles interrupts when there is already an interrupt in service depends on the nesting mode.

Normal Nesting Mode

This mode is set when IW4 is not written or when IW4 has EXTN = 0. It is the most common nesting mode. See figure 15.

When an interrupt is being executed in this mode (corresponding bit of ISR = 1), only interrupt requests with higher priority can be accepted.

Extended Nesting Mode

This mode is only applicable to a master in the extended mode. A slave's eight interrupt priority levels become only one priority level when viewed by the master. Therefore, a request made by a slave with a higher priority than a previous request from the same slave will not be accepted. This cannot be called complete nesting since priority ranking within slaves loses its significance.

The extended nesting mode is set by setting bit D4 of IW4 in both the master and the slave. Interrupt requests of a higher level than the one currently being serviced can be accepted in the master from the same slave in the extended nesting mode.

Care should be exercised when issuing an FI (finish interrupt) command in the extended nesting mode. In an interrupt by a slave, the CPU first issues an FI command to the slave. Then, the CPU reads the slave's in-service register (ISR) to see if that slave still has interrupts in service. If there are no interrupts in service, (ISR = 00H) an FI command is issued to the master, as in the single mode when an interrupt is made by a peripheral.

Figure 14. Buffer Mode

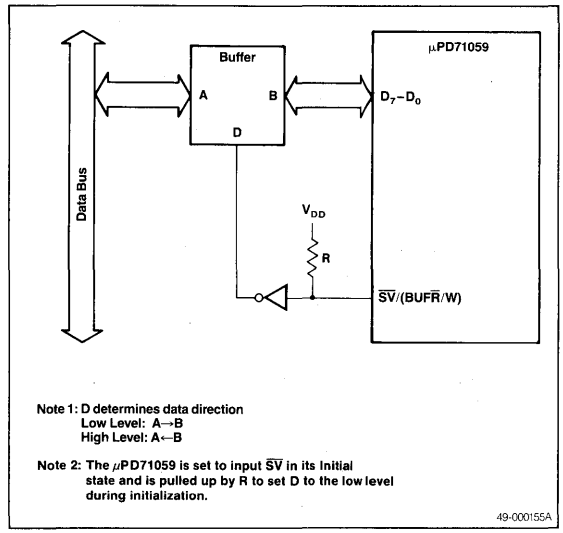
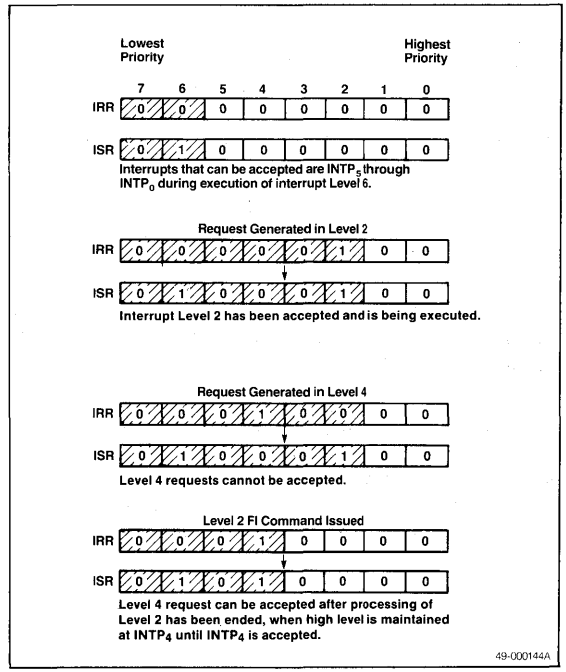


Figure 15. Normal Nesting Mode



5f

Exceptional Nesting Mode

A μPD71059 in the normal or extended nesting mode cannot accept interrupts of a lower priority than the interrupts in service. Sometimes, however, it is desirable that requests with lower priority be accepted while higher-priority interrupts are being serviced. Setting the exceptional nesting mode allows this. After releasing the exceptional mode, the previous mode is resumed.

The exceptional nesting mode is controlled by the SNM (set nesting mode) and EXCN (exceptional nesting mode) bits (D_6 and D_5) of MCW. They set and release the exceptional nesting mode. The mode doesn't change when SNM = 0. Exceptional nesting is set if SNM and EXCN = 1 and released when SNM =1 and EXCN = 0.

Setting a bit in the IMW in the exceptional nesting mode, inhibits interrupts of that level and allows unmasked interrupts to all other levels, higher or lower priority.

The procedure for setting the exceptional nesting (EN) mode is as follows:

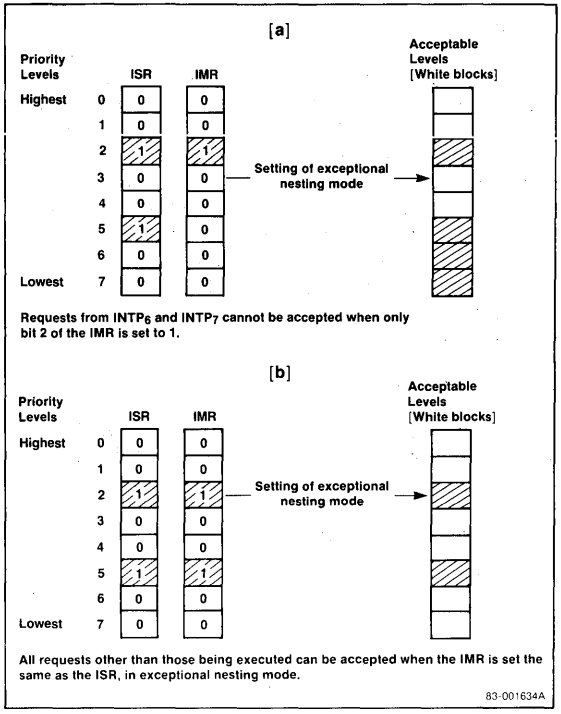
(1) Read the ISR.
(2) Write the ISR data to the IMR.
(3) Set the exceptional nesting mode.

In this way, all interrupt requests not currently in service will be enabled.

Figure 16 (a) shows what happens if IMR is not set to ISR. When the exceptional nesting is set, bit 2 of ISR will be ignored, and bit 5 will be serviced. Servicing bit 5 will mask the lower priority interrupts 6 and 7. When the ISR is set equal to the IMR as in (b), all interrupts except 2 and 5 can be serviced when the exceptional nesting mode is set.

Issuing an FI command to a level masked by the exceptional nesting mode requires caution. Since the ISR bit is masked, the normal FI command will not work. For this reason, a specific FI command specifying the ISR bit must be issued. After the exceptional mode is released, the normal FI command may be used.

Figure 16. Exceptional Nesting Mode



Finishing Interrupts (FI) and Changing the Priority Levels

The priority and finish control word (PFCW) issues FI commands and changes interrupt priorities.

Normal FI Command

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PFCW =	0	0	1	0	0	X	X	X

When a normal FI command is issued, the μPD71059 resets the ISR bit corresponding to the highest priority level selected from the interrupts in service. This operation assumes that the interrupt accepted last has ended.

When an interrupt routine changes the priority level or the exceptional nesting mode is set, this command will not operate correctly because the highest priority interrupt is not necessarily the last interrupt in service.

Specific FI Command

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PFCW =	0	1	1	0	0	IL_2	IL_1	IL_0

When the specific FI command is issued, the μPD71059 resets the ISR bit designated by bits IL_2-IL_0 of the PFCW. This command is used when the normal nesting mode isn't being used.

Self-FI Mode

When SFI of IW4 = 1, the μPD71059 is set to the self-FI mode. In this mode, the ISR bit corresponding to the interrupt is set and reset during the third $\overline{INTAK}$ pulse. Therefore, the CPU does not have to issue an FI command when the interrupt routine ends. In this mode, however, the ISR does not store the routine in service. Unless interrupts are disabled by the interrupt routine, newly generated interrupt requests are generated without priority limitation by the ISR. This can cause a stack overflow when frequent interrupt requests occur, or when the interrupt is level triggered.

Self-FI Rotation

Rotation of interrupt priorities can be added to the self-FI mode. In this case, the corresponding interrupt is set to the lowest priority level when a bit is reset in the ISR at the end of the $\overline{INTAK}$ sequence.

Self-FI Rotation Set:

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PFCW =	1	0	0	0	0	X	X	X

Self-FI Rotation Reset:

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PFCW =	0	0	0	0	0	X	X	X

Normal Rotation FI Command

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PFCW =	1	0	1	0	0	X	X	X

When the normal rotation FI command is issued, the μPD71059 resets the ISR bit corresponding to the highest priority level selected from the interrupts in service, then rotates the priority levels so that the interrupt just completed has the lowest priority.

Specific Rotation FI Command

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PFCW =	1	1	1	0	0	IL_2	IL_1	IL_0

When the specific rotation FI command is issued, the μPD71059 resets the ISR bit designated by bits IL_2-IL_0 of the PFCW and rotates the interrupt priorities so that the interrupt just reset becomes the lowest priority. This change in priority levels is different from the normal nesting mode, therefore, it is the user's responsibility to manage nesting.

Specific Rotation Command

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PFCW =	1	1	0	0	0	IL_2	IL_1	IL_0

When the specific rotation command is issued, the μPD71059 sets the interrupt priority specified by IL_2-IL_0 to the lowest priority. In this case also, the user must manage nesting.

5f

Triggering Mode

Bit D_3 of the first initialization word, IW1, is LEV (level-trigger mode bit). LEV sets the trigger mode of the INTP inputs. The level-trigger mode is set when LEV = 1. The rising-edge-triggered mode is set when LEV = 0.

Edge-Trigger Mode

In the edge-trigger mode, an interrupt is detected by the rising edge of the signal on an INTP input. Although an IRR bit goes high when INTP is high, the IRR bit is not latched until the CPU returns an $\overline{INTAK}$ pulse. Therefore, the INTP input should be maintained high until $\overline{INTAK}$ is received. This filters out noise spikes on the INT lines. To send the next interrupt request, temporarily lower the INTP input, then raise it.

Level-Trigger Mode

In the level-trigger mode, an IRR bit is set by the INTP input being at a high level. As in the edge-trigger mode, the INTP must be maintained high until the $\overline{INTAK}$ is received. Interrupts are requested as long as the INTP input remains high. Care should be taken so as not to cause a stack overflow in the CPU. See figure 17.

Note: The μPD71059 operates as if the $INTP_7$ interrupt had occurred if the $\overline{INTAK}$ pulse is sent to the μPD71059 by the CPU when the μPD71059 INT output level is low. Bit 7 of ISR is not set. Accordingly, if it is expected that this will occur, the $INTP_7$ interrupt should be reserved for servicing incomplete interrupts. The FI should not be issued for incomplete interrupts. See figure 18.

Figure 17. INTP Input

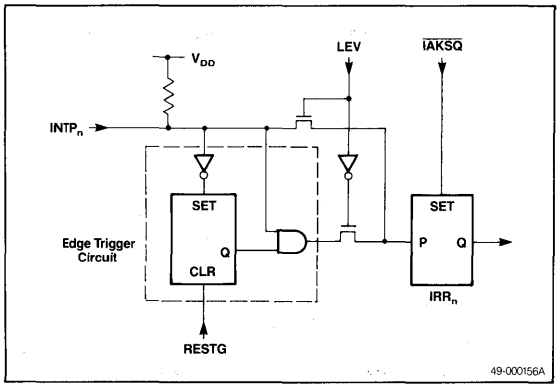
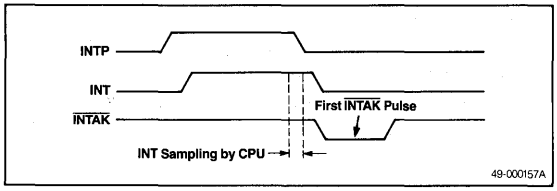


Figure 18. Incomplete Interrupt Request



Polling Operation

When polling, the CPU should disable its INT input. Next, it issues a polling command to the μPD71059 using MCW with POL = 1. This command sets the μPD71059 in polling mode until the CPU reads one of the μPD71059's registers.

When the CPU performs a read operation with $A_0 = 0$ in the polling mode, polling data as shown in figure 19 is read instead of ISR or IRR. The μPD71059 then ends the polling mode.

Figure 19. Polling Data

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MCW =	INT	0	0	0	0	PL_2	PL_1	PL_0

83-001635A

The INT bit has the same meaning as the INT pin. When it is set to 1, it means that the μPD71059 has accepted an INTP input.

The PL_2-PL_0 (permitted level) bits show which INTP input requested an interrupt when INT = 1.

If INT in the polling data is 1, the μPD71059 sets the ISR bit corresponding to the interrupt level shown by bits PL_2-PL_0 of the polling data and considers that interrupt as being executed. The CPU then processes the interrupt accordingly, based on the polling data read. An FI command should be issued when this processing ends.

Note: When a read is performed with A0 = 1 after the polling command is sent to the μPD71059, the IMR will be read instead of polling data. However, when the polling command is sent, the μPD71059 operates in the same manner when $A_0 = 0$ as it does when $A_0 = 1$. This means that although A_0 was set to 1, the μPD71059 will send the contents of the IMR, but it will also set an ISR bit just as it would if A0 had been set to zero. This may disturb the nesting. Therefore, performing a read operation with $A_0 = 1$ immediately after sending the polling command should be avoided.

Description

The μPD71071 is a high-speed, high-performance direct memory access (DMA) controller that provides high-speed data transfers between peripheral devices and memory. A programmable bus width allows bidirectional data transfer in both 8- and 16-bit systems. In addition, the μPD71071 uses CMOS technology to reduce power consumption.

The μPD71071 can perform a variety of transfer functions including byte/word, memory-to-memory, and transfers between memory and I/O. The μPD71071 also utilizes single, demand, and block mode transfers; release and bus hold modes; and normal and compressed timing.

Features

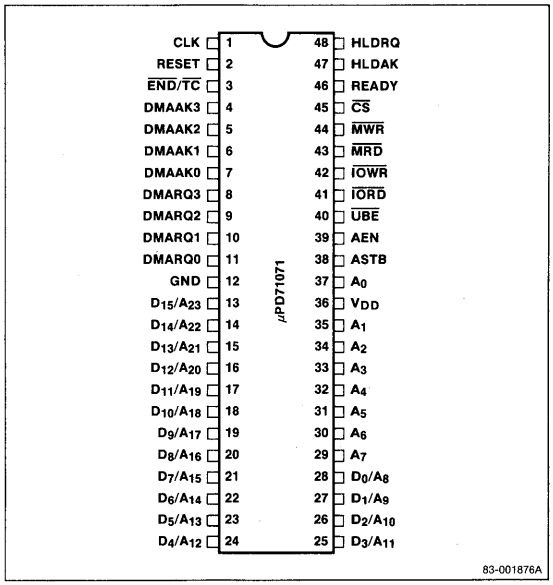
- ☐ Four independent DMA channels
- ☐ 16M-byte addressing
- ☐ 64K-byte/word transfer count
- ☐ 8- or 16-bit programmable data bus width
- ☐ Enable/disable of individual DMA requests
- ☐ Software DMA requests
- ☐ Enable/disable of autoinitialize
- ☐ Address increment/decrement
- ☐ Fixed/rotational DMA channel priority
- ☐ Terminal count output signal
- ☐ Forced transfer termination input
- ☐ Cascade capability
- ☐ Programmable DMA request and acknowledge signal polarities
- ☐ High performance: transfers to 5.33 Mbytes/s
- ☐ μPD70108/70116-compatible
- ☐ CMOS technology
- ☐ Low-power standby mode
- ☐ Single power supply, 5 V ±10%
- ☐ Industrial temperature range, −40 to +85 °C
- ☐ 10 MHz operation

Ordering Information

Part Number	Package
μPD71071C-10	48-pin plastic DIP
L-10	52-pin PLCC

Pin Configurations

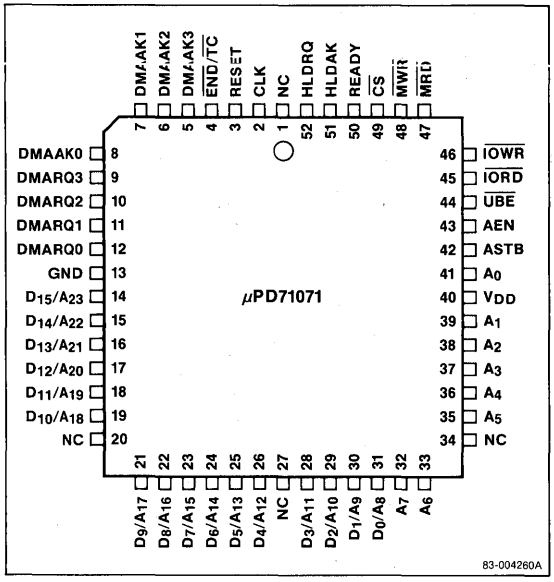
48-Pin Plastic DIP



5g

Pin Configurations (cont)

52-Pin Plastic Leaded Chip Carrier (PLCC)



Pin Functions

CLK [Clock]

CLK controls the internal operation and data transfer speed of the μPD71071.

RESET [Reset]

RESET initializes the controller's internal registers and leaves the controller in the idle cycle (CPU controls the bus). Active high.

$\overline{END}/\overline{TC}$ [End/Terminal Count]

This is a bidirectional pin. The $\overline{END}$ input is used to terminate the current DMA transfer. $\overline{TC}$ indicates the designated cycles of the DMA count transfer have finished. $\overline{END}/\overline{TC}$ is open drain and requires an external pull-up resistor. Active low.

DMAAK3-DMAAK0 [DMA Acknowledge]

DMAAK3-DMAAK0 indicates to peripheral devices that DMA service has been granted. DMAAK3-DMAAK0 respond respectively to DMA channels 3-0 and the polarities are user programmable.

Pin Identification

Symbol	Function
A_{23}-A_8/ D_{15}-D_0	Bidirectional address/data bus
IC	Internally connected; leave open
A_7-A_4	Address bus output
NC	Not connected
A_3-A_0	Bidirectional address bus
V_{DD}	Power supply
ASTB	Address strobe output
AEN	Address enable output
$\overline{UBE}$	Upper byte enable input/output
$\overline{IORD}$	I/O read input/output
$\overline{IOWR}$	I/O write input/output
$\overline{MRD}$	Memory read output
$\overline{MWR}$	Memory write output
$\overline{CS}$	Chip select input
READY	Ready input
HLDAK	Hold acknowledge input
HLDRQ	Hold request output
CLK	Clock input
RESET	Reset input
$\overline{END}/\overline{TC}$	End DMA transfer input/terminal count output
DMAAK3-DMAAK0	DMA acknowledge output
DMARQ3-DMARQ0	DMA request input
GND	Ground

DMARQ3-DMARQ0 [DMA Request]

DMARQ3-DMARQ0 accept DMA service requests from peripheral devices. DMARQ3-DMARQ0 respond respectively to DMA channels 3-0 and the polarities are user programmable. DMARQ must remain asserted until DMAAK is asserted.

GND [Ground]

GND connects to the power supply ground terminal.

A_{23}-A_8/D_{15}-D_0 [Address/Data Bus]

A_{23}-A_8/D_{15}-D_0 function as a 16-bit, multiplexed address/data bus when the μPD71071 is in the 16-bit data mode. In the 8-bit data mode, A_{23}-A_{16} (pins 13-20) become address bits only and A_{15}-A_8/D_7-D_0 (pins 21-28) remain an 8-bit multiplexed address/data bus. A_{23}-A_8/D_{15}-D_0 are three-state.

A_7-A_4, A_3-A_0 [Address Bus]

A_7-A_4, A_3-A_0 function as the lower eight bits of the address bus. A_7-A_4 output memory addresses during the DMA cycle and become high impedance in the idle cycle. A_3-A_0 function as the lower four bits of the address bus. In the idle cycle, A_3-A_0 become address inputs to select internal registers for the CPU to read or write. In the DMA cycle, A_3-A_0 output memory addresses.

V_{DD} [Power Supply]

V_{DD} connects to the +5-V power supply.

ASTB [Address Strobe]

ASTB latches address A_{23}-A_8 (16-bit mode)/A_{15}-A_8 (8-bit mode) from the address/data bus into an external address latch at the falling edge of ASTB during a DMA cycle. Active high.

AEN [Address Enable]

AEN enables the output of an external latch that holds DMA addresses. AEN becomes high during the DMA cycle.

$\overline{UBE}$ [Upper Byte Enable]

$\overline{UBE}$ indicates the upper byte of the data bus is valid during 16-bit mode. In the idle cycle during data transfer, the μPD71071 acknowledges data on D_{15}-D_8 when $\overline{UBE}$ is asserted. During a DMA cycle, UBE goes low to signify the presence of valid data on D_{15}-D_8. $\overline{UBE}$ has no meaning in 8-bit mode and becomes high impedance in the idle cycle and high level in the DMA cycle. Three-state, active low.

$\overline{IORD}$ [I/O Read]

In the idle cycle, $\overline{IORD}$ inputs a read signal from the CPU. In the DMA cycle, $\overline{IORD}$ outputs a read signal to an I/O device. Three-state, active low.

$\overline{IOWR}$ [I/O Write]

In the idle cycle, $\overline{IOWR}$ inputs a write signal from the CPU. In the DMA cycle, $\overline{IOWR}$ outputs a write signal to an I/O device. Three-state, active low.

$\overline{MRD}$ [Memory Read]

During the DMA cycle, $\overline{MRD}$ outputs a read signal to memory. $\overline{MRD}$ is high impedance during the idle cycle. Three-state, active low.

$\overline{MWR}$ [Memory Write]

During the DMA cycle, $\overline{MWR}$ outputs a write signal to memory. $\overline{MWR}$ is high impedance during the idle cycle. Three-state, active low.

$\overline{CS}$ [Chip Select]

During the idle cycle, $\overline{CS}$ selects the μPD71071 as an I/O device. Active low.

READY [Ready]

During a DMA operation, READY indicates that a data transfer for one cycle has been completed and may be terminated. To meet the requirements of low-speed I/O devices or memory, READY may be negated to insert wait states to extend the bus cycle until READY is again asserted.

HLDAK [Hold Acknowledge]

When active, HLDAK indicates that the CPU has granted the μPD71071 the use of the system bus. Active high.

HLDRQ [Hold Request]

HLDRQ outputs a bus hold request to the CPU. Active high.

Block Diagram Description

The μPD71071 has the following functional units.

- Bus control unit
- DMA control unit
- Address registers
- Address incrementer/decrementer
- Count registers
- Count decrementer
- Control registers

5g

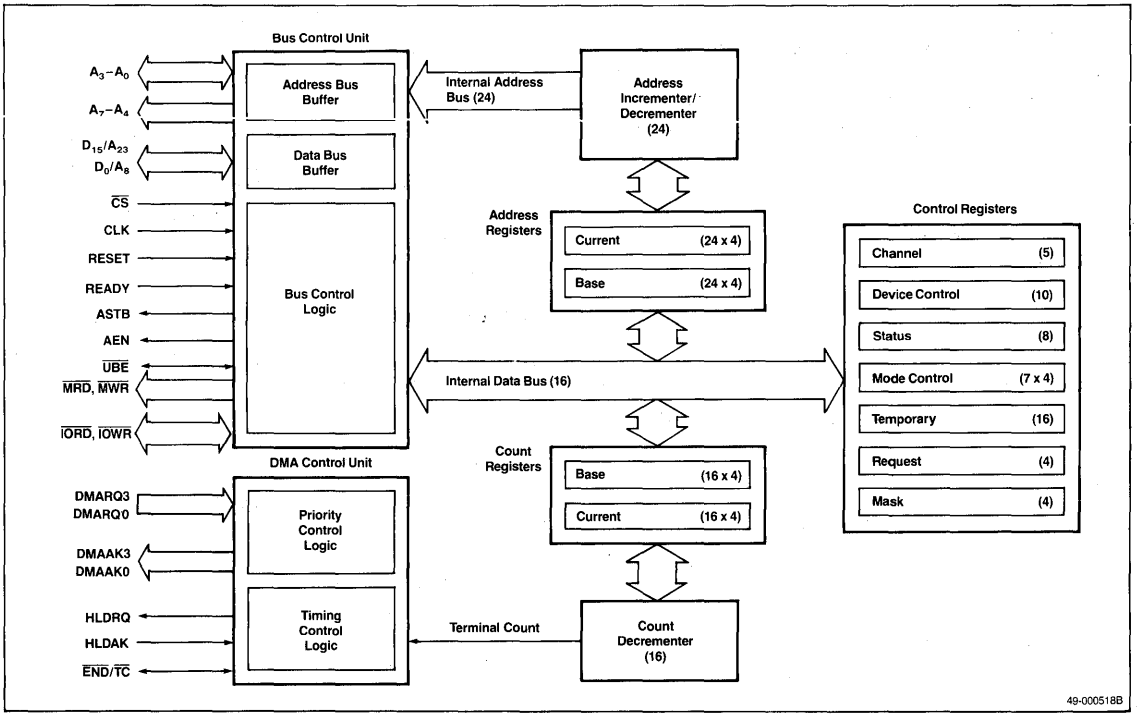
Bus Control Unit

The bus control unit consists of the address and data buffers, and bus control logic. The bus control unit generates and receives signals that control addresses and data on the internal address and data buses.

DMA Control Unit

The DMA control unit contains the priority and timing control logic. The priority control logic determines the priority level of DMA requests and arbitrates the use of the bus in accordance with this priority level. The DMA control unit also provides internal timing and controls DMA operations.

Block Diagram



Address Registers

Each of the four DMA channels has one 24-bit base address register and one 24-bit current address register. The base address register holds a value determined by the CPU and transfers this value to the current address register during autoinitialization (address and count are automatically initialized). The channel's current address register is incremented/decremented for each transfer and always contains the address of the data to be transferred next.

Address Incrementer/Decrementer

The address incrementer/decrementer updates the contents of the current address register whenever a DMA transfer completes.

Count Registers

Each of the four DMA channels has one 16-bit base count register and one 16-bit current count register. The base count register holds a value written by the CPU and transfers the value to the current count register during autoinitialization. A channel's current count register is decremented for each transfer and generates a terminal count when the count register is decremented to FFFFH.

Note: The number of DMA transfer cycles is actually the value of the current count register + 1. Therefore, when programming the count register, specify the number of DMA transfers minus one.

Count Decrementer

The count decrementer decrements the contents of the current count register by one when each DMA transfer cycle ends.

Control Registers

The μPD71071 contains the following control registers.

- Channel
- Device
- Status
- Mode
- Temporary
- Request
- Mask

These registers control bus mode, pin active levels, DMA operation mode, mask bits, and other μPD71071 operating functions.

Absolute Maximum Ratings

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.3 V
Output voltage, V_O	−0.5 to V_{DD} + 0.3 V
Operating temperature, T_{OPT}	−40 to +85 °C
Storage temperature, T_{STG}	−65 to +150 °C

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Capacitance

T_A = 25 °C

Parameter	Symbol	Limits Typ	Limits Max	Unit	Test Conditions
Output capacitance	C_O	4	8	pF	f_c = 1.0 MHz unmeasured pins returned to 0 V
Input capacitance	C_I	8	15	pF	
I/O capacitance	C_{IO}	10	18	pF	

DC Characteristics

T_A = −40 to +85 °C, V_{DD} = 5 V ±10%

Parameter	Symbol	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
Input high voltage	V_{IH}	3.3		V_{DD} + 0.3	V	CLK input pin
		2.2		V_{DD} + 0.3	V	Other inputs
Input low voltage	V_{IL}	−0.5		0.8	V	
Output high voltage	V_{OH}	0.7 V_{DD}			V	I_{OH} = −400 μA
Output low voltage	V_{OL}			0.4	V	I_{OL} = 2.5 mA; 4.5 mA (TC)
Input leakage current	I_{LI}			±10	μA	0 V ≤ V_I ≤ V_{DD}
Output leakage current	I_{LO}			±10	μA	0 V ≤ V_O ≤ V_{DD}
Supply current (dynamic)	I_{DD1}		15	30	mA	
Supply current (stable)	I_{DD2}		10		μA	Inputs stable outputs open
Supply current (static)	I_{DD2}		10		μA	

5g

AC Characteristics

$T_A = -40$ to $+85°C$, $V_{DD} = 5$ V ±10%

Parameter	Symbol	Min	Max	Unit	Test Conditions
DMA Mode					
Clock cycle	t_{CYK}	100		ns	
Clock pulse width high	t_{KKH}	39		ns	
Clock pulse width low	t_{KKL}	49		ns	
Clock rise time	t_{KR}		10	ns	1.5 V → 3.0 V
Clock fall time	t_{KF}		10	ns	3.0 V → 1.5 V
Input rise time	t_{IR}		20	ns	
Input fall time	t_{IF}		12	ns	
Output rise time	t_{OR}		20	ns	
Output fall time	t_{OF}		12	ns	
DMARQ setup time to CLK high	t_{SDQ}	20		ns	S1, S0, S3, SW, S4w
HLDRQ high delay from CLK low	t_{DHQH}	5	70	ns	S1, S4w
HLDRQ low delay from CLK low	t_{DHQL}	5	70	ns	S1, S0, S4w
HLDRQ low level period	t_{HQHQL}	$2t_{CYK} - 50$		ns	S4w
HLDAK high setup time to CLK low	t_{SHA}	20		ns	S0, S4, S4w
AEN high delay from CLK low	t_{DAEH}	5	70	ns	S1, S2
AEN low delay time from CLK low	t_{DAEL}	5	70	ns	S1, S4w
ASTB high delay time from CLK low	t_{DSTH}	5	70	ns	S1
ASTB low delay time from CLK high	t_{DSTL}	5	70	ns	S1
ASTB high level period	t_{STSTH}	$t_{KKL} - 15$		ns	
ADR/$\overline{UBE}$/$\overline{RD}$/$\overline{WR}$ active delay from CLK low (Note 1)	t_{DA}	5	80	ns	S1, S2
ADR/$\overline{UBE}$/$\overline{RD}$/$\overline{WR}$ float time from CLK low	t_{FA}	0	70	ns	S1, S4w
ADR setup time to ASTB low	t_{SAST}	$t_{KKL} - 40$		ns	
ADR hold time to ASTB low	t_{HSTA}	$t_{KKH} - 20$		ns	

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Test Conditions
DMA Mode (cont)					
ADR/$\overline{UBE}$ off delay time from CLK low	t_{DAF}	0	70	ns	S1, S2
$\overline{RD}$ low delay time from ADR float	t_{DAR}	−10		ns	
Input data delay time from $\overline{MRD}$ low	t_{DMRID}		$2t_{CYK} - 80$	ns	S12
Input data hold time from MRD high	t_{HMRID}	0		ns	S14
Output data delay time from CLK low	t_{DOD}	10	80	ns	S22
Output data hold time from CLK high	t_{HOD}	10		ns	S24
Output data hold time from $\overline{MWR}$ high	t_{HMWOD}	$t_{KKL} - 35$		ns	
$\overline{RD}$ low delay time from CLK high	t_{DKHR}			ns	S2 compressed timing
$\overline{RD}$ low level period	t_{RRL1}	$2t_{CYK} - 30$		ns	Normal timing
	t_{RRL2}	$t_{CYK} + t_{KKH} - 30$		ns	Compressed timing
$\overline{RD}$ high delay time from CLK low	t_{DRH}	10	70	ns	S4
ADR delay time from $\overline{RD}$ high	t_{DRA}	$t_{CYK} - 30$		ns	
$\overline{WR}$ low delay time from CLK low	t_{DWL1}	5	50	ns	S3 normal write
$\overline{WR}$ low delay time from CLK low	t_{DWL2}	5	50	ns	S2 extended write, normal timing
$\overline{WR}$ low delay time from CLK high	t_{DWL3}	5	50	ns	S2 extended write, compressed timing
$\overline{WR}$ low level period	t_{WWL1}	$t_{CYK} - 30$		ns	Normal write
	t_{WWL2}	$2t_{CYK} - 30$		ns	Extended write, normal timing
	t_{WWL3}	$t_{CYK} + t_{KKH} - 30$		ns	Extended write, compressed timing
$\overline{WR}$ high delay from CLK low	t_{DWH}	5	50	ns	S4
$\overline{RD}$ low delay time from CLK low	t_{DKLR}	5	50	ns	S2 normal timing
$\overline{RD}$, $\overline{WR}$ low delay from DMAAK active	t_{DDARW}	0		ns	S1, S2
$\overline{RD}$ high delay time from $\overline{WR}$ high	t_{DWHRH}	5		ns	
DMAAK delay time from CLK high	t_{DKHDA}	5	70	ns	S1 I/O memory timing
DMAAK delay time from CLK low	t_{DKLDA}	10	90	ns	S1 cascade mode
DMAAK inactive delay time from CLK high	t_{DDAI1}	5		ns	S4

5g

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Test Conditions
DMA Mode (cont)					
DMAAK inactive delay time from	t_{DDAI2}	5	$t_{KKL}+70$	ns	S4 cascade mode, HLDAK low HLDAK low in S4
	t_{DDAI3}		$4t_{KKL}+70$	ns	S4 cascade mode, HLDAK low except in S4
DMAAK active level period	t_{DADA}			ns	Cascade mode
$\overline{TC}$ low delay time from CLK high	t_{DTCL}	5	70	ns	S3
$\overline{TC}$ off delay time from CLK high	t_{DTCF}		30	ns	S4
$\overline{TC}$ high delay time from CLK high	t_{DTCH}		$t_{KKH}+t_{CYK}-10$	ns	0 to 2.2 V (Note 2)
$\overline{TC}$ low level period	t_{TCTCL}	$t_{CYK}-15$		ns	
$\overline{END}$ low setup time to CLK high	t_{SED}	20		ns	S2
$\overline{END}$ low level period	t_{EDEDL}	50		ns	
READY setup time to CLK high	t_{SRY}	20		ns	S3, SW
READY hold time from CLK high	t_{HRY}	10		ns	S3, SW
Programming Mode and RESET					
$\overline{IOWR}$ low level period	t_{IWIWL}	80		ns	
$\overline{CS}$ low setup time to $\overline{IOWR}$ high	t_{SCSIW}	80		ns	
$\overline{CS}$ hold time from IOWR high	t_{HIWCS}	0		ns	
ADR/$\overline{UBE}$ setup time to $\overline{IOWR}$ high	t_{SAIW}	80		ns	
ADR/$\overline{UBE}$ hold time from $\overline{IOWR}$ high	t_{HIWA}	0		ns	
Input data setup time to $\overline{IOWR}$ high	t_{SIDIW}	80		ns	
Input data hold time from $\overline{IOWR}$ high	t_{HIWID}	0		ns	
$\overline{IORD}$ low level period	t_{IRIRL}	120		ns	
ADR/$\overline{CS}$ setup time to $\overline{IORD}$ low	t_{SAIR}	20		ns	
ADR/$\overline{CS}$ hold time from $\overline{IORD}$ high	t_{HIRA}	0		ns	
Output data delay time from $\overline{IORD}$ low	t_{DIROD}	10	100	ns	
Output data float time from $\overline{IORD}$ high	t_{FIROD}		80	ns	
RESET high level period	t_{RESET}	$2t_{CYK}$		ns	
V_{DD} setup time to RESET low	t_{SVDD}	500		ns	
$\overline{IOWR}$/$\overline{IORD}$ wait time from RESET low	t_{SYIWR}	$2t_{CYK}$		ns	RESET low to first read/write
$\overline{IOWR}$/$\overline{IORD}$ recovery time	t_{RVIWR}	160		ns	

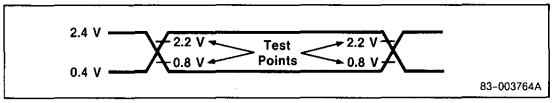
Notes:

(1) $\overline{RD}$/$\overline{WR}$ refers to $\overline{IORD}$ or $\overline{MRD}$ and $\overline{IOWR}$ or $\overline{MWR}$, respectively.

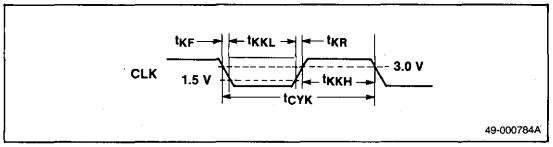
(2) For $\overline{END}$/$\overline{TC}$, output load capacitance = 75 pF maximum. To meet the t_{DTCH} parameter use a 2.2-kΩ pull-up resistor with a load capacitance of 75 pF. For other than $\overline{END}$/$\overline{TC}$, output load capacitance = 100 pF maximum.

Timing Waveforms

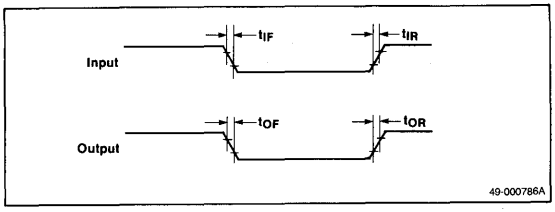
Timing Measurement Points



Clock Timing



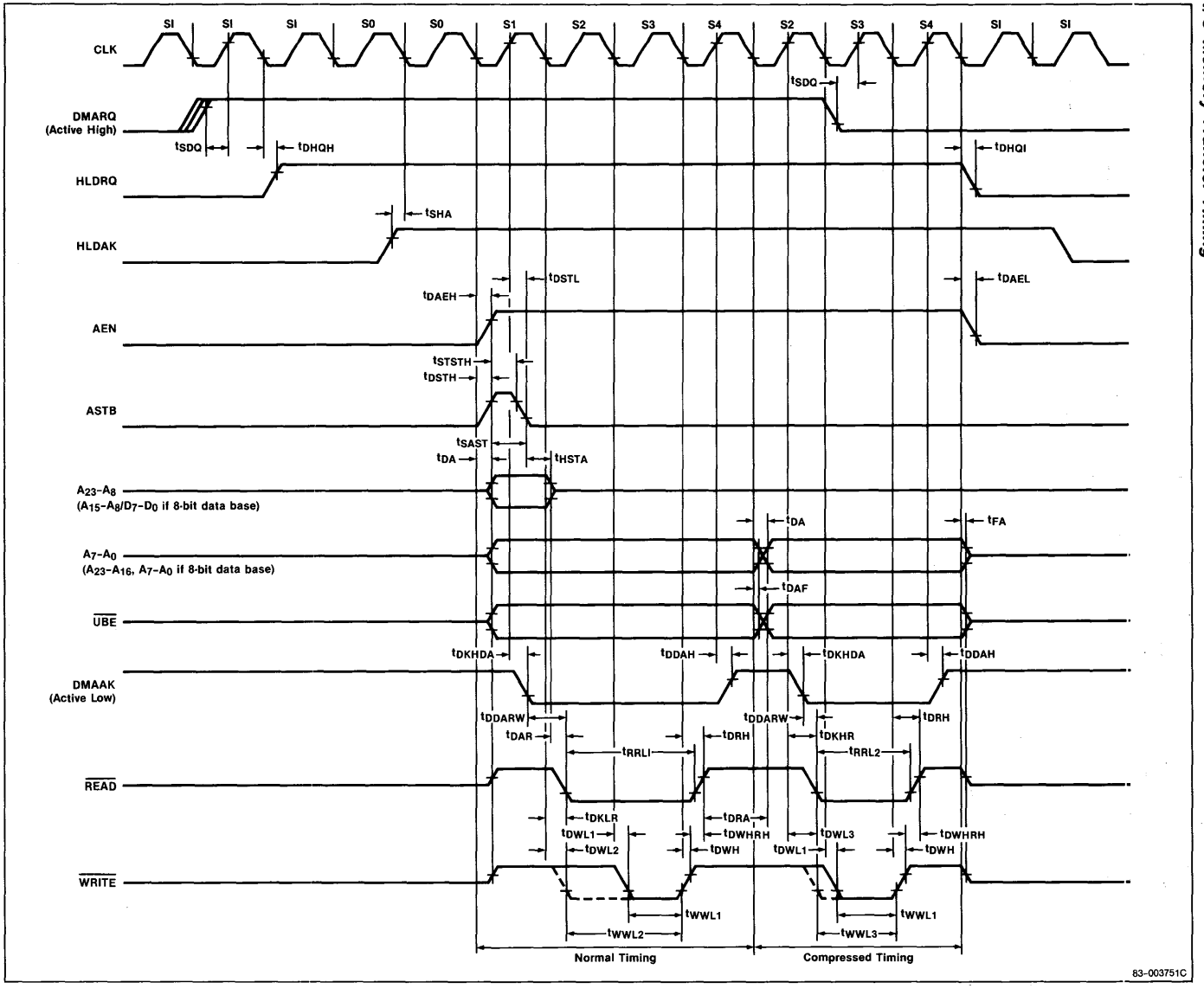
Input/Output Edge Timing



5g

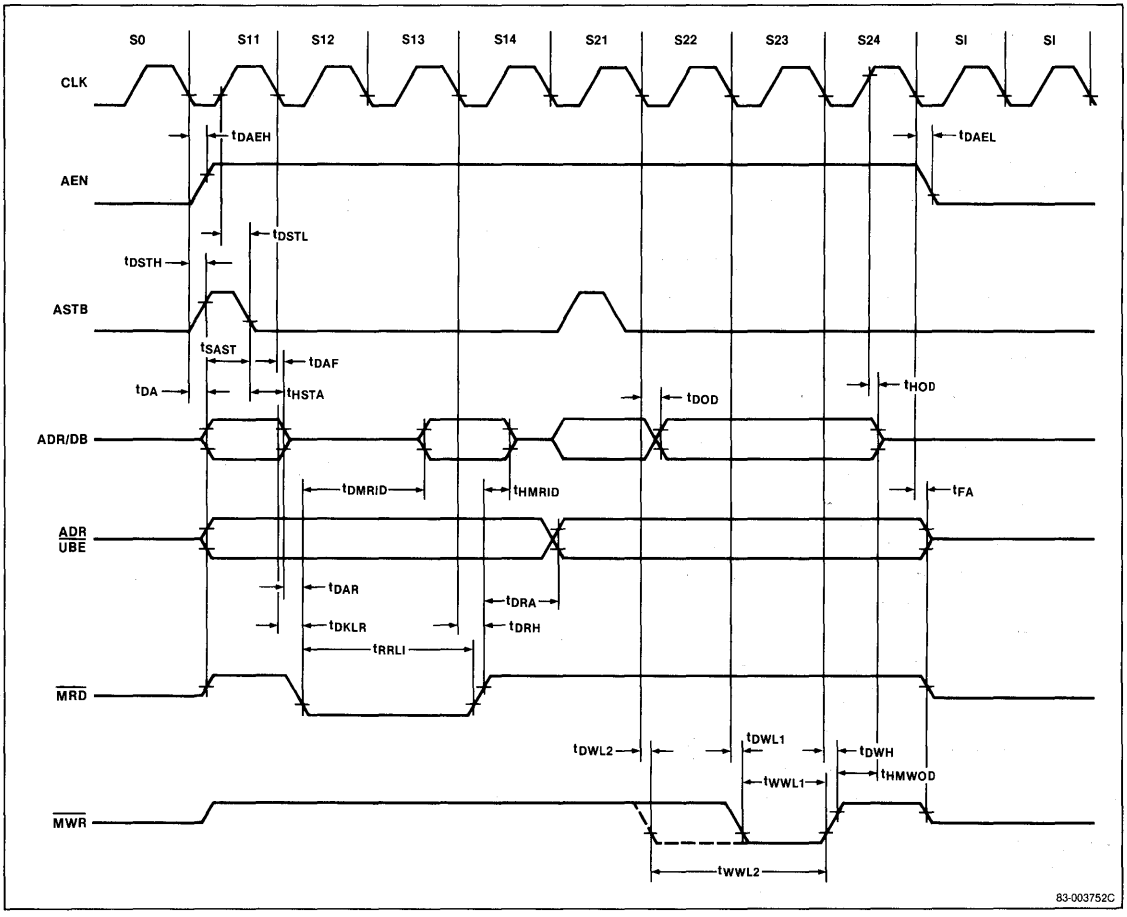
Timing Waveforms (cont)

I/O-Memory Transfer Timing

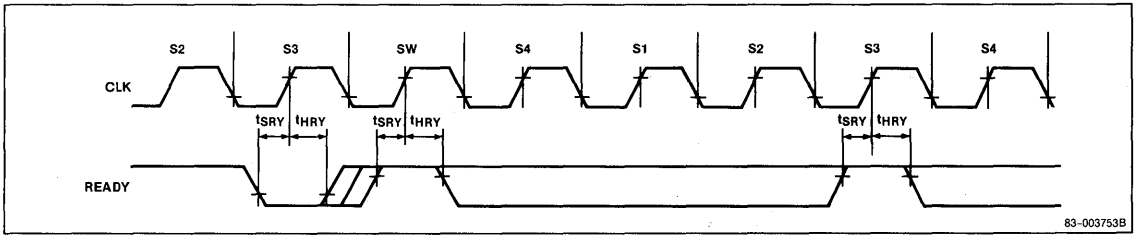


Timing Waveforms (cont)

Memory-to-Memory Transfer Timing



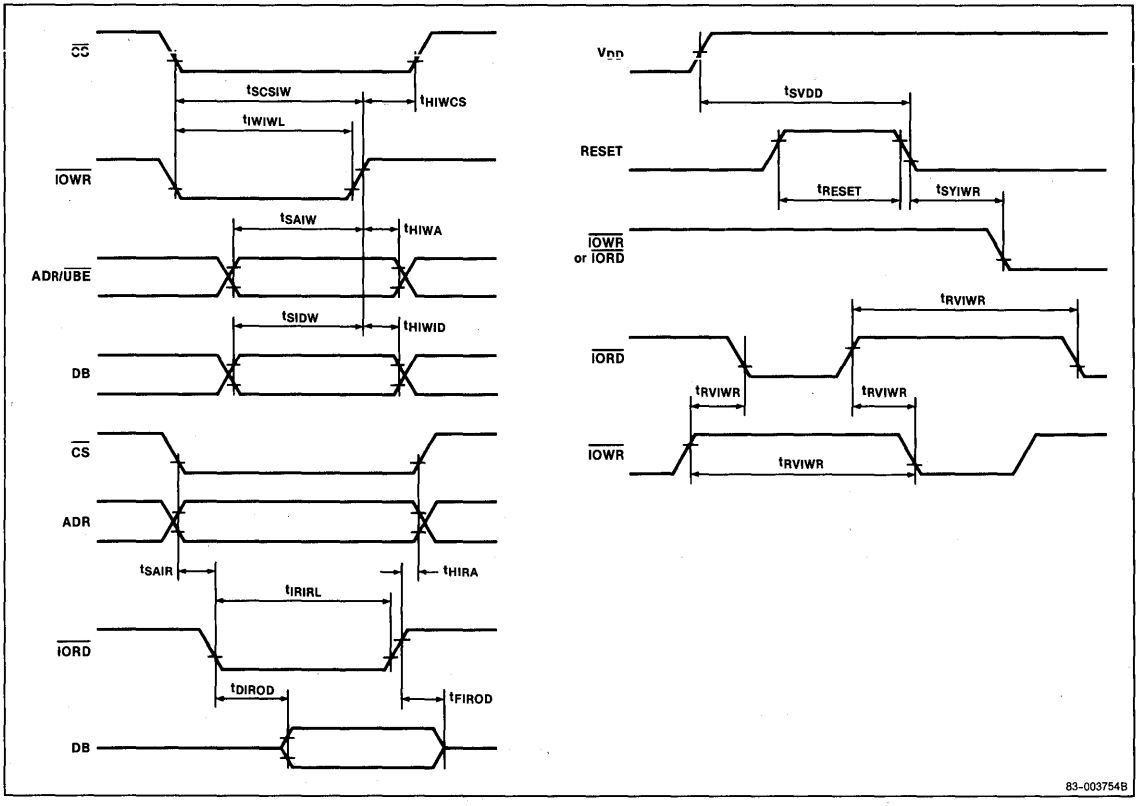
Ready Timing



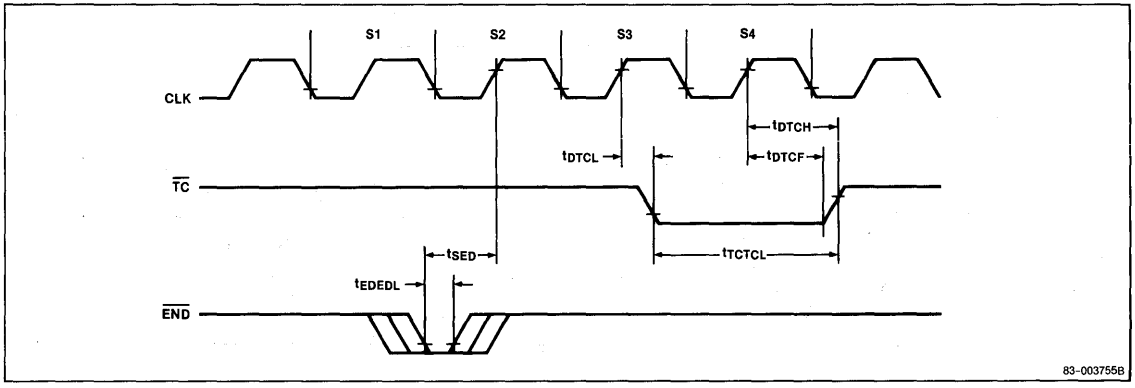
5g

Timing Waveforms (cont)

Programming Mode and RESET Timing

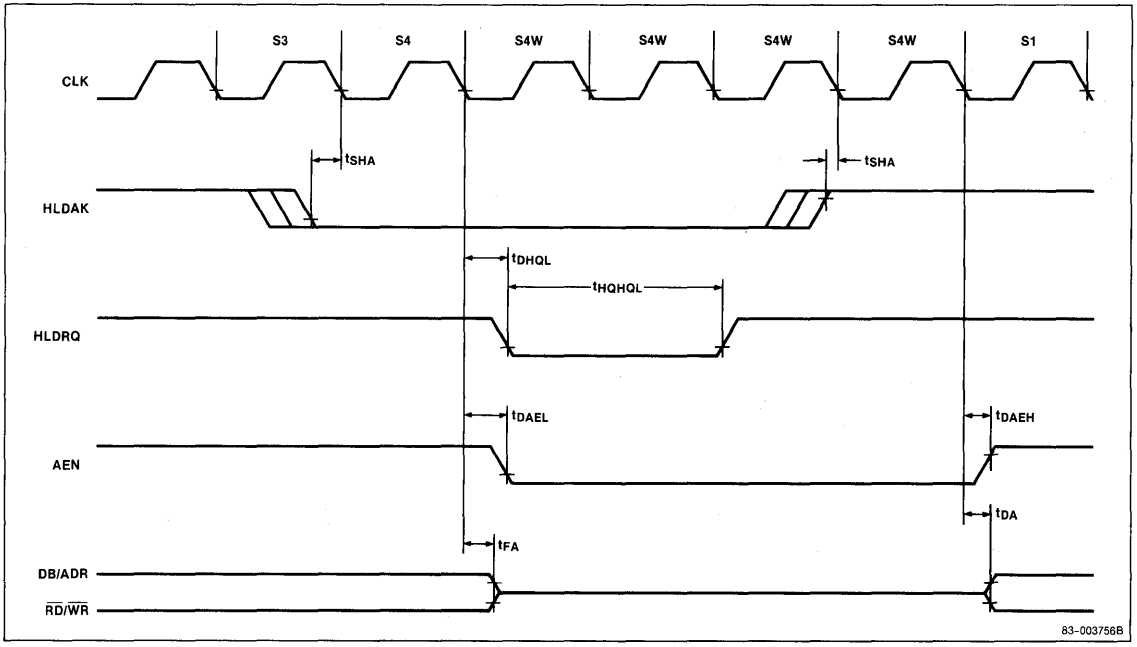


END/TC Timing

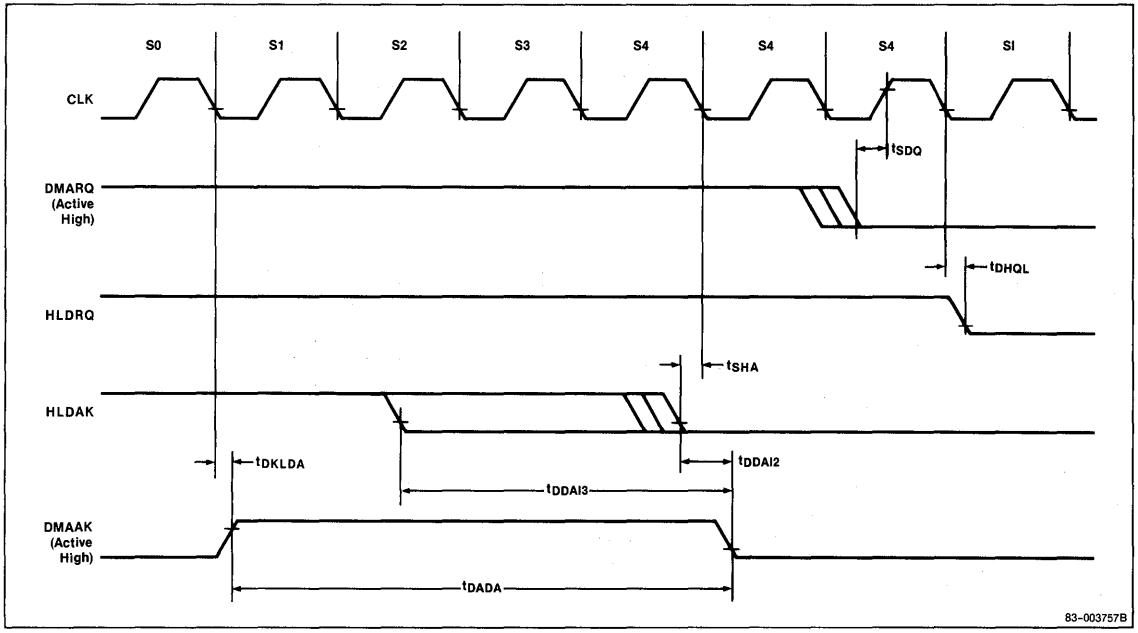


Timing Waveforms (cont)

Bus Wait Timing



Cascade Timing



5g

Functional Description

DMA Operation

The μPD71071 functions in three cycles: idle, DMA, and standby. In an idle or standby cycle, the CPU uses the bus, while in a DMA cycle, the μPD71071 uses it.

Idle Cycle. In an idle cycle, there are no DMA cycles active, but there may be one or more active DMA requests; however, the CPU has not released the bus. The μPD71071 will sample the four DMARQ input pins at every clock. If one or more inputs are active, the corresponding DMA request bits (RQ) are set in the status register and the μPD71071 sends a bus hold request to the CPU. The μPD71071 continues to sample DMA requests until it obtains the bus.

After the CPU returns a HLDAK signal and the μPD71071 obtains the bus, the μPD71071 stops DMA sampling and selects the DMA channel with the highest priority from the valid DMA request signals. Programming of the μPD71071 is done when the μPD71071 is in the idle cycle or the standby mode.

DMA Cycle. In a DMA cycle, the μPD71071 controls the bus and performs DMA transfer operations based on programmed information. Figure 1 outlines the sequentialflow of a DMA operation.

Standby Mode. The μPD71071 can also be used in standby mode. It is in standby mode and consumes the static supply current (I_{DD2}) when the clock is turned off and no I/O read or write operations are being performed. All internal registers will retain their contents.

The μPD71071 can be programmed (using $\overline{IOWR}$) and read (using $\overline{IORD}$) with the clock off. The μPD71071 only uses the clock for the DMA data transfer cycles. The clock may be turned off without altering the internal registers when the μPD71071 is in the idle cycle. If the clock is turned off during a DMA transfer, the μPD71071 will not operate correctly. When the clock is off, the DMARQ inputs will not be recognized. The DMARQ inputs could be externally logically ORed and cause an interrupt to the CPU. The CPU could then turn on the clock, thus activating the μPD71071. If the previously programmed mode of operation is still valid, the μPD71071 does not have to be reprogrammed.

Data Bus Width

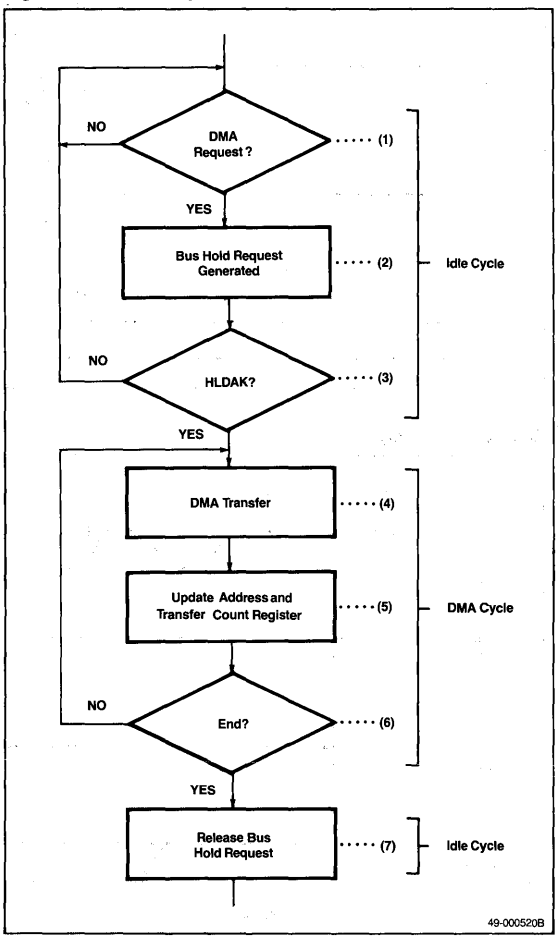
In order to allow an easy interface with an 8- or 16-bit CPU, the data bus width of the μPD71071 is user programmable for 8 or 16 bits. A 16-bit data bus allows 16-bit memory-to-memory DMA transfers and also provides a one-I/O bus cycle access to the 16-bit internal registers.

Table 1 shows the relationship of the data bus width, A_0, $\overline{UBE}$, and the internal registers.

Table 1. Data Bus Width

Bus Width	A_0	$\overline{UBE}$	Internal Read/Write Registers
8 bits	X	X	D_7-D_0 ⟷ 8-bit internal register
16 bits	0	1	D_7-D_0 ⟷ 8-bit internal register
	1	0	D_{15}-D_8 ⟷ 8-bit internal register
	0	0	D_{15}-D_0 ⟷ 16-bit internal register

Figure 1. DMA Operation Flow



Terminal Count

The μPD71071 ends DMA service when it generates a terminal count ($\overline{TC}$) or when the $\overline{END}$ input becomes active. A terminal count is produced when a borrow is generated by the current count register and a low-level pulse is output to the $\overline{TC}$ pin. Figure 2 shows that the current count register is tested after each DMA operation.

If autoinitialize is not set when DMA service ends, the mask register bit applicable to the channel where service ended is set, and the DMARQ input of that channel is masked.

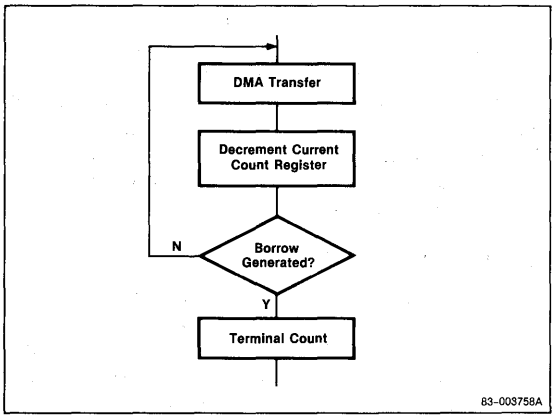
DMA Transfer Type

The type of transfer the μPD71071 performs depends on the following conditions.

- Memory-to-memory transfer enable
- Direction of memory-to-I/O transfer (each channel)
- Transfer mode (each channel)
- Bus mode

Memory-to-Memory Transfer Enable. The μPD71071 can perform memory-to-I/O transfers (one transfer cycle in one bus cycle) and memory-to-memory transfers (one transfer in two bus cycles). To select memory-to-memory transfer, set bit 0 of the device control register to 1. The DMA channels used in memory-to-memory transfers are fixed, with channel 0 as the source channel and channel 1 as the destination channel. Channels 2 and 3 cannot be used in memory-to-memory transfers. The contents of the count registers and word/byte transfer modes of channels 0 and 1 should be the same when performing memory-to-memory transfer.

Figure 2. Generation of Terminal Count ($\overline{TC}$)



For memory-to-memory byte transfer in 16-bit data bus mode, a read data from upper data bus is to be written to upper data bus, while a read data from lower data bus is to be written to lower data bus. Therefore, start addresses for source and destination must be the even-even or odd-odd. For word transfer, only even-even addresses are to be set for source and destination. (See Byte/Word Transfer paragraphs below.) When DMARQ0 (channel 0) becomes active, the transfer is initiated.

During memory-to-memory bus cycles in the 16-bit mode, data read from the DMAC's upper (lower) data bus is written to the upper (lower) data bus of the destination device. Thus, for word transfers, only even source and destination addresses should be used.

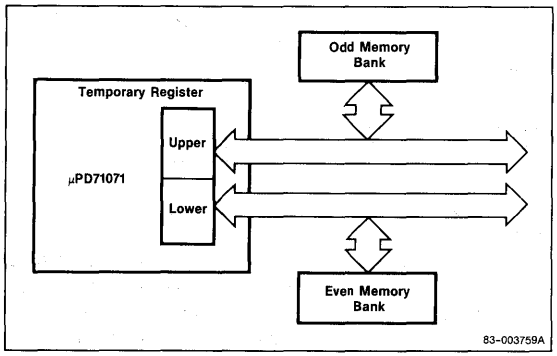
The DMA request input pin or a software DMA request to channel 0 may initiate memory-to-memory transfers. The μPD71071 performs the following operations until a channel 1 terminal count or $\overline{END}$ input is present:

- During the first bus cycle, the memory data pointed to by the current address register of channel 0 is read into the temporary register of the μPD71071 and the address and count of channel 0 are updated.
- During the second bus cycle, the temporary register data is written to the memory location shown by the current address register of channel 1, and the address and count of channel 1 are updated.

Note: If DMARQ1 (channel 1) becomes active, the μPD71071 will perform memory-to-I/O transfer even though memory-to-memory transfer is selected. Since this may cause erroneous memory-to-memory transfers, mask out channel 1 (DMARQ1) by setting bit 1 of the mask register to 1 before starting memory-to-memory transfers.

During memory-to-memory transfers, the addresses on the source side (channel 0) can be fixed by setting bit 1 of the device control register to 1. In this manner, a

Figure 3. Memory-to-Memory Transfer in 16-Bit Data Bus Mode



5g

range of memory can be initialized with the same value since the contents of the source address never change. During memory-to-memory transfer, the DMAAK signal and channel 0's terminal count ($\overline{TC}$) pulse are not output. (See figure 3.)

Direction of Memory-to-I/O Transfers. All DMA transfers use memory as a reference point. Therefore, a DMA read reads a memory location and writes to an I/O port. A DMA write reads an I/O port and writes the data to a memory location. In memory-to-I/O transfer, use the mode control register to set one of the transfer directions in table 2 for each channel and activate the appropriate control signals.

Table 2. Transfer Direction

Transfer Direction	Activated Signals
Memory → I/O (DMA read)	$\overline{IOWR}$, $\overline{MRD}$
I/O → memory (DMA write)	$\overline{IORD}$, $\overline{MWR}$
Verify (Outputs addresses only. Does not perform a transfer.)	—

Transfer Modes. In memory-to-I/O transfer, the mode control register selects the single, demand, or block mode of DMA transfer for each channel. The conditions for the termination of each transfer characterize each transfer mode. Memory-to-memory transfers have no relationship to single, demand, or block mode. Memory-to-memory transfers are a separate and distinct type of transfer mode. Table 3 shows the various transfer modes and termination conditions.

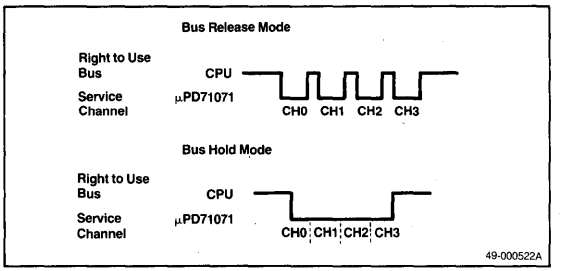
Table 3. Transfer Termination

Transfer Mode	End of Transfer Conditions
Single	After each byte/word
Demand	$\overline{END}$ input Generation of terminal count When DMA request of the channel in service becomes inactive When DMA request of a channel in higher priority becomes active (bus hold mode)
Block	$\overline{END}$ input Generation of terminal count
Memory-to-memory	$\overline{END}$ input Generation of terminal count

Bus Modes. The device control register selects either the bus release or bus hold mode. The bus mode determines when the μPD71071 returns the system bus to the CPU. The μPD71071 can be in either the release or hold modes for the single, demand, or block mode transfers. Therefore, there are six possible mode combinations.

Figure 4 shows that in bus release mode, only one channel can receive service after obtaining the bus. When DMA service ends (end of transfer conditions depend on the transfer mode), the channel returns the bus to the CPU (regardless of the state of other DMA requests) and the μPD71071 enters the idle cycle. When the μPD71071 regains use of the bus, a new DMA operation begins.

Figure 4. Bus Modes



In bus hold mode, several channels can receive service without releasing the bus after obtaining it. If there is another valid DMA request when a channel's DMA service is finished, the new DMA service can begin after the previous service without returning the bus to the CPU. End of transfer conditions depend on the transfer mode. A channel cannot terminate (end count) a transfer mode and immediately start on its next set of transfers. There must be another DMA channel service interleaved or the μPD71071 will put in an idle cycle. The following shows an example of the possible sequences for Channel 2.

CHAN2 → CHANn (n = 0,1,3) → CHAN2
or,
CHAN2 → idle → CHAN2

The operation of single, demand, and block mode transfers depends on whether the μPD71071 is in bus release or bus hold mode. In bus release mode, only one type of bus mode (single, demand, or block) is used each time the μPD71071 has the bus. In bus hold mode, multiple types of transfers are possible. Channel 0 might operate in the demand mode, and channel 1, which could get the bus immediately after channel 0, could operate in block mode.

Single Mode Transfer

In bus release mode, when a channel completes the transfer of a single byte or word, the μPD71071 enters the idle cycle regardless of the state of the DMA request inputs. In this manner, other devices will be able to access the bus on alternate bus cycles.

In bus hold mode, when a channel completes the transfer of a single byte or word, the μPD71071 terminates the channel's service even if it is still asserting a DMA request signal. The μPD71071 will then service the highest priority channel requesting the bus. If there are no requests from any other channel, the μPD71071 releases the bus and enters the idle cycle.

Demand Mode Transfer

In bus release mode, the currently active channel continues its data transfer as long as the DMA request of that channel is active, even though other DMA channels are issuing higher priority requests. When the DMA request of the serviced channel becomes inactive, the μPD71071 releases the bus and enters the idle state, even if the DMA request lines of other channels are active.

In bus hold mode, when the active channel completes a single transfer, the μPD71071 checks DMA request lines (other request lines when $\overline{END}$ or TC, all request lines including the last serviced channel when there is no $\overline{END}$ or TC). If there are active requests, the μPD71071 starts servicing the highest priority channel requesting service. If there is no request, the μPD71071 releases the bus and enters the idle state.

Block Mode Transfer

In bus release mode, the current channel continues data transfer until a terminal count or the external $\overline{END}$ signal becomes active. During this time, the μPD71071 ignores all other DMA requests. After completion of the block transfer, the μPD71071 releases the bus and enters the idle cycle even if DMA requests from other channels are active.

In bus hold mode, the current channel transfers data until a terminal count or the external $\overline{END}$ signal becomes active. When the service is complete, the μPD71071 checks all DMA requests without releasing the bus. If there is an active request, the μPD71071 immediately begins servicing the request. The μPD71071 releases the bus after it honors all DMA requests or a higher priority bus master requests the bus.

Figure 5 shows the operation flow for the six possible transfer and bus mode operations in DMA transfer.

Byte/Word Transfer

If the initialize command selects a 16-bit data bus width, the mode control register can specify DMA transfer in byte or word units for each channel. Table 4 shows the update of the address and count registers during byte/word transfer.

Table 4. Address and Count Registers

Register	Byte Transfer	Word Transfer
Address	±1	±2
Count	−1	−1

During word transfers, two bytes starting at an even address are handled as one word. If word transfer is selected and the initial value of the set address is odd, the μPD71071 will always decrement that address by 1, thus making the address even for the data transfer. For this reason, it is best to select even addresses when transferring words, to avoid destroying data. A_0 and $\overline{UBE}$ control byte and word transfers.

Table 5 shows the relationship between the data bus width, A_0 and $\overline{UBE}$ signals, and data bus status.

Table 5. Data Bus Status

Data Bus Width	A_0	$\overline{UBE}$	Data Bus Status
8 bits	X	1 (1)	D_7-D_0 valid byte
16 bits	0	1	D_7-D_0 valid byte
	1	0	D_{15}-D_8 valid byte
	0	0	D_{15}-D_0 valid word

Note:

(1) Always 1 for an 8-bit bus.

5g

Compressed Timing

In transfers between I/O and memory, a DMA transfer cycle is normally executed in four clocks. However, when the device control register selects compressed timing, one DMA cycle can be executed in a three-clock bus cycle. Compressed timing may be used in the release or hold modes when doing block transfers between I/O and memory. In the demand mode, only use compressed timing in the bus release mode. Compressed timing mode increases data transfer rates by 33%.

The μPD71071 is able to omit one clock period during compressed timing by not updating the upper 16 bits of the latched address. In block mode and demand bus release mode, addresses are output sequentially and the upper 16 bits of addresses latched in external latches need not be updated except after a carry or borrow from A_7 to A_8. For this reason, during compressed timing, the S1 state (output of upper 16 bits of an address for external latching) is omitted in the bus cycles except during the first bus cycle when the upper 16 bits of an address are changed. Figure 6 shows one word waveforms for normal and compressed timing.

Figure 5. Transfer and Bus Modes Operations

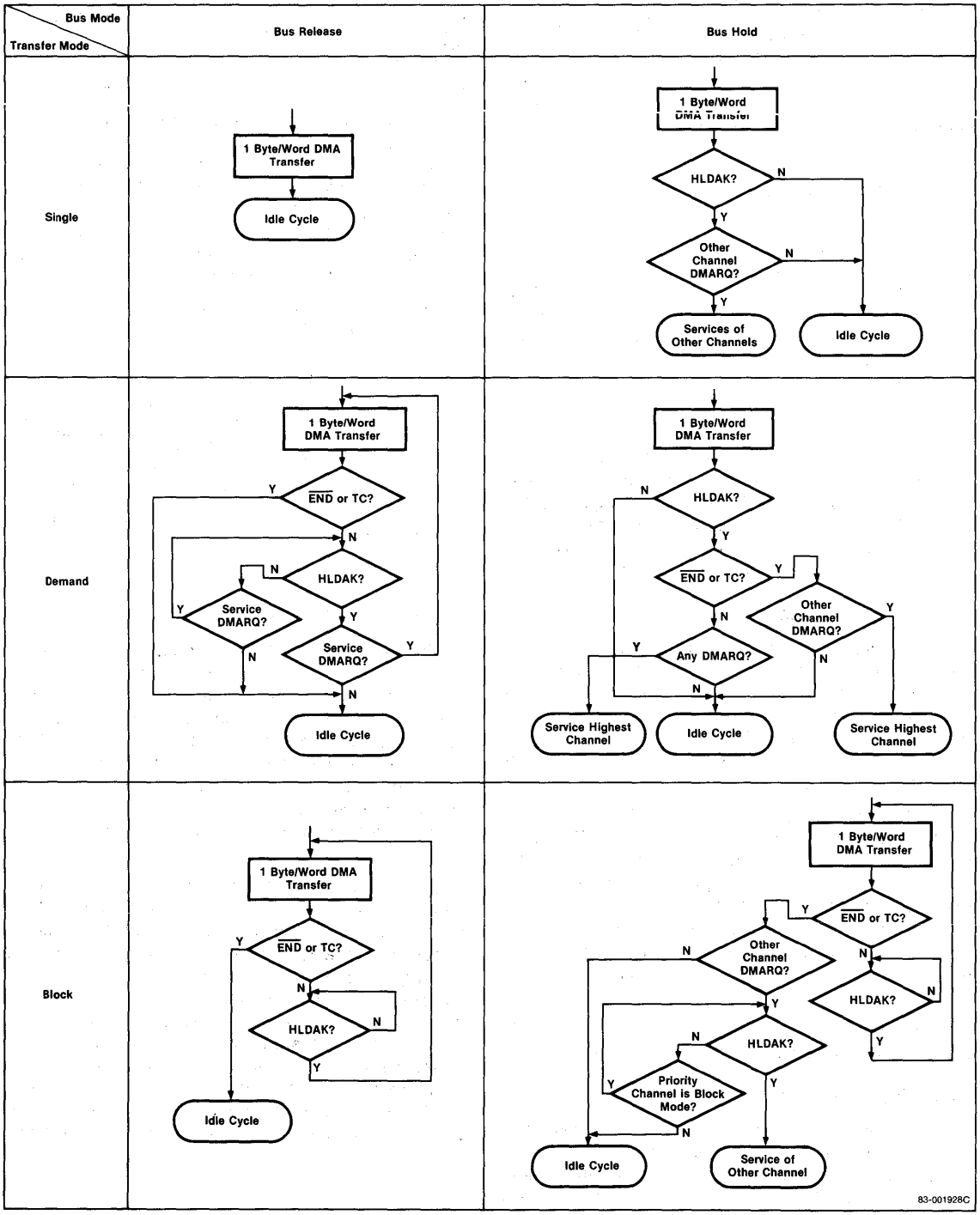
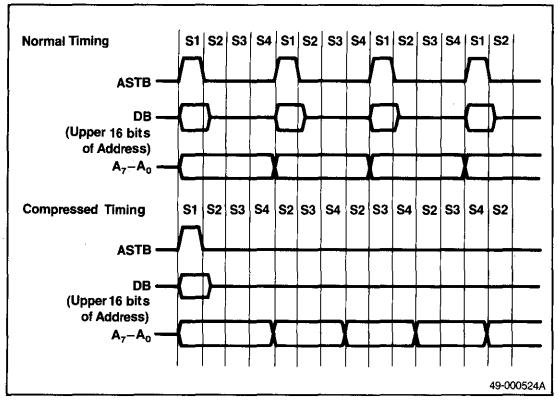


Figure 6. Normal and Compressed Timing Waveforms



Software DMA Requests

The μPD71071 can accept software DMA requests in addition to DMA requests from the four DMARQ pins. Setting the appropriate bit in the request register generates a software DMA request. The mask register does not mask software DMA requests. Software DMA requests operate differently depending on which bus or transfer mode is used.

Bus Mode. When bus release mode is set, the highest priority channel among software DMA requests and DMARQ pins is serviced, and all bits of the request register are cleared when the service is over. Therefore, there is a chance that other software DMA requests will be cancelled.

When bus hold mode is set, only the corresponding bit of the request register is cleared after a DMA service is over. Therefore, all software DMA requests will be serviced in the sequence of their priority level.

Software DMA requests for cascade channels (see Cascade Connection) must be performed in bus hold mode. When a cascade channel is serviced, the master μPD71071 operational mode is changed to bus release mode temporarily and all bits of the request register are cleared when the cascade channel service is over. To avoid this, it is necessary to mask any cascade channels before issuing a software DMA request. After confirming that all DMA software services are complete and all bits of the request register are cleared, the cascade channel masks can be cleared.

Transfer Mode. When single or demand mode is set, the applicable request bits are cleared and software DMA service ends with the transfer of one byte/word. When block mode or memory-to-memory modes are set, service continues until $\overline{END}$ is input or a terminal count is generated. Applicable request bits are cleared when service ends.

Autoinitialize

When the mode control register is set to autoinitialize a channel, the μPD71071 automatically initializes the address and count registers when $\overline{END}$ is input or a terminal count is generated. The contents of the base address and base count registers are transferred to the current address and current count registers, respectively. The applicable bit of the mask register is unaffected. The applicable bit of the mask register is set for channels not programmed for autoinitialize.

The autoinitialize function is useful for the following types of transfers.

Repetitive Input/Output of Memory Area. Figure 7 shows an example of DMA transfer between a CRT controller and memory. After setting the value in the base and current registers, autoinitialize allows repetitive DMA transfer between the CRT controller and the video memory area without CPU involvement.

Continuous Transfer of Several Memory Areas. The CPU can indirectly write to the address or count registers by writing to the base registers. New values can be written to the base registers. In the autoinitialize mode, the value in the base register will be transferred to the address/count registers when termination is reached in the address/count registers. Because of this, the autoinitialize function can perform continuous transfer of several contiguous or noncontiguous memory areas during single or demand bus release modes in the following manner.

During the transfer of data in area 1 (the first area being transferred), the CPU can write address and count information about area 2 (the second area to be transferred). Generation of a terminal count for area 1 results in the transfer of information of area 2 to the address and count registers. This will cause area 2 to be transferred. Figure 8 illustrates this procedure.

Channel Priority

Each of the μPD71071's four channels has its own priority. When there are DMA requests from several channels simultaneously, the channel with the highest priority will be serviced. The device control register selects one of two channel priority methods: fixed and rotational priority. In fixed priority, the priority (starting with the highest) is channel 0, 1, 2, and 3, respectively. In rotational priority, priority order is rotated so that the channel that has just been given service receives the lowest priority and the next highest channel number is given the highest priority. This method prevents exclusive servicing of some channel(s). Figure 9 shows the two priority order methods.

Figure 7. Autoinitialize Application 1

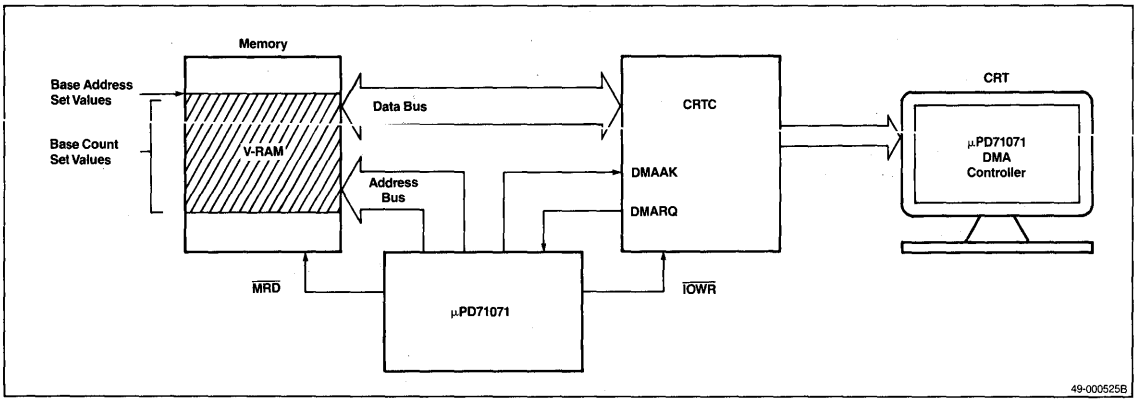


Figure 8. Autoinitialize Application 2

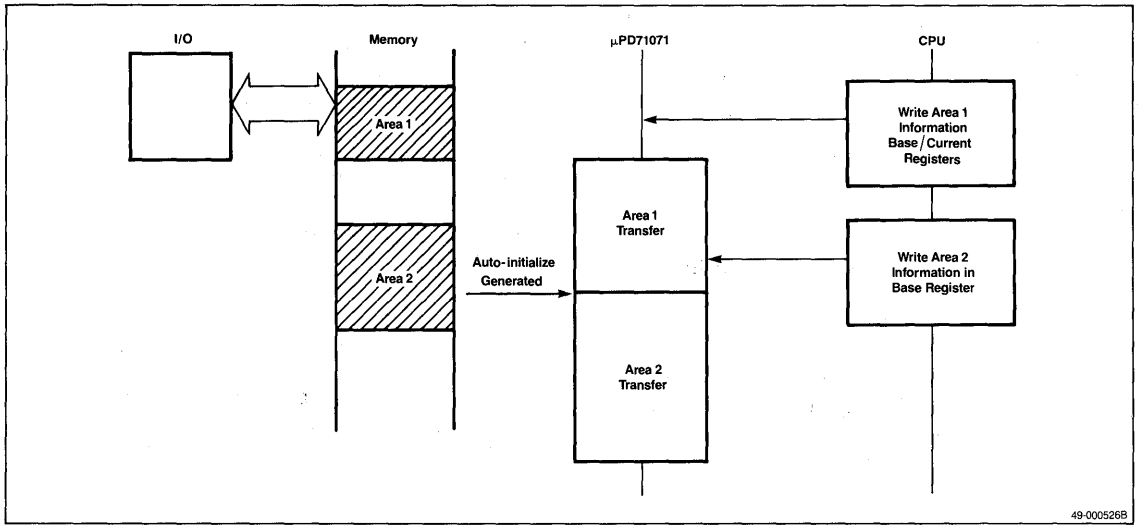
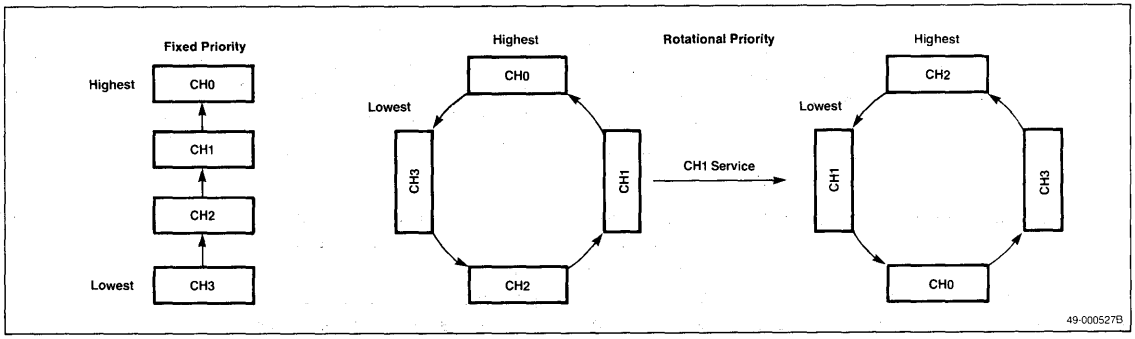


Figure 9. Priority Order



Cascade Connection

The μPD71071 can be cascaded to expand the system DMA channel capacity. To connect a μPD71071 for cascading (figure 10), perform the following operations.

(1) Connect pins HLDRQ and HLDAK of the second-stage (slave) μPD71071 to pins DMARQ and DMAAK of any channel of the first-stage (master) μPD71071.

(2) To select the cascade mode of a particular channel of a master μPD71071, set bits 7 and 6 of that channel's mode control register to 11.

When a channel is set to the cascade mode in a master μPD71071, DMARQ, DMAAK, HLDRQ, HLDAK, and RESET are the only valid signals in the master μPD71071. The other signals are disabled. The master cascade channel only intermediates hold request/hold acknowledge between the slave and CPU.

The master μPD71071 always operates in the bus release mode when a cascade channel is in service (even when the bus hold mode is set). Other DMA requests are ignored while a cascade channel is in service. When the slave μPD71071 ends DMA service and moves into an idle cycle, the master also moves to an idle cycle and releases the bus. At this time, all bits of the master's request register are cleared. The master operates its non-cascaded channels normally.

Bus Wait Operation

In systems using a μPD70208/70216 (V40/V50) as the CPU, the refresh control unit in the CPU changes the HLDAK signal to inactive (even during a DMA cycle) and uses the bus. Here, the μPD71071 automatically performs a bus wait operation. This system has a bus master (V40/V50) whose priority level is higher than that of the μPD71071.

The μPD71071 executes the bus wait operation when the HLDAK signal becomes inactive in an operating mode where transfer is executed continuously in block mode, during demand bus release mode, or during memory-to-memory transfer.

When HLDAK becomes inactive during service in other operating modes, the operation returns to the idle cycle and transfers control of the bus to the higher bus master.

Figure 11 shows that when the HLDAK signal becomes inactive during a continuous transfer, the μPD71071 is set up in an S4w state (bus wait). Operation moves to the idle cycle if DMARQ is inactive in the demand mode. The HLDRQ signal is made inactive for a period of about two clocks and the bus is released. The S4w state is repeated until the HLDAK signal again becomes active and the interrupted service is immediately restarted.

5g

Figure 10. Cascade Connection Example

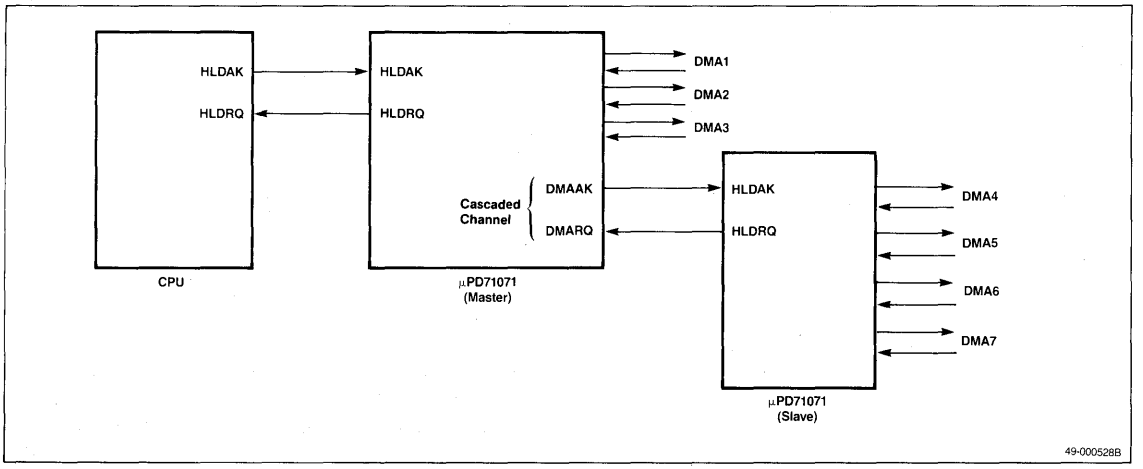
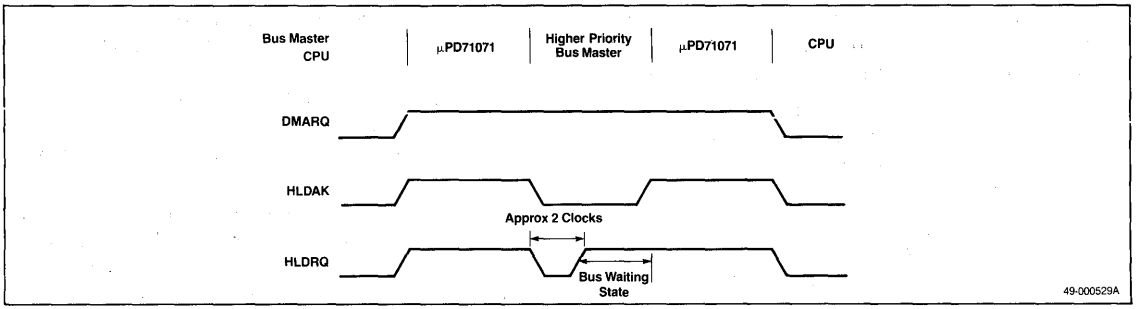


Figure 11. Bus Wait Operation



Programming the μPD71071

To prepare a channel for DMA transfer, you must select the following characteristics.

- Starting address for the transfer
- Number of byte/word transfers
- DMA operating modes
- Data bus widths
- Active levels of the DMARQ and DMAAK signals

When reading from or writing to a μPD71071 internal register, address lines A_3-A_0 select the register, $\overline{IORD}$ or $\overline{IOWR}$ select the data transfer direction, and $\overline{CS}$ enables the transfer. Table 6 shows the register and command configurations.

Table 6. Register Configuration

Register	Bit size
Channel	5
Base address	24 (4)
Current address	24 (4)
Base count	16 (4)
Current count	16 (4)
Mode control	7 (4)
Device control	10
Status	8
Request	4
Mask	4
Temporary	16

Note:

When using a 16-bit CPU and selecting a 16-bit data bus, the word IN/OUT instruction can be used to read/write information two bytes at a time. However, commands in table 7 suffixed with B must be issued with the byte IN/OUT instruction.

Initialize

Use the initialize command as a software initialize to the μPD71071 or to set the width of the data bus. When using a 16-bit CPU, set the data bus width to 16 bits first. Figure 12 shows the initialize command format.

Bit 0. When the RES bit is set, the internal state of the μPD71071 is initialized and will be the same as when a hardware reset is used (except for data bus width selection). A software reset leaves bit 16B intact whereas a hardware reset selects the 8-bit data bus. After initialization, the registers are as in table 8 and the RES bit is cleared automatically.

Table 8. Register Initialization

Register	Initialization Operation
Initialize	Clears bit 0 only
Address	No change
Count	No change
Channel	Selects channel 0, current and base
Mode control	Clears all bits
Device control	Clears all bits
Status	Clears bits 3-0 only
Request	Clears all bits
Mask	Sets all bits (masks all channels)
Temporary	Clears all bits

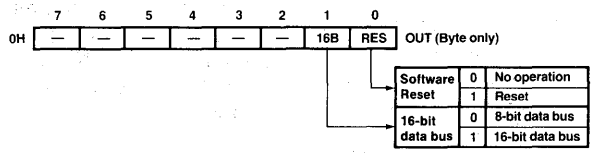
Bit 1. The 16B bit determines the data bus width. When using the μPD71071 in a 16-bit system, set this bit immediately after a hardware reset since a hardware reset always initializes it to the 8-bit data bus mode.

Table 7. Command Configuration

Address	R/W	Command Name	MSB			Format				LSB
0H	W(B)	Initialize	—	—	—	—	—	—	16B	RES
1H	R(B)	Channel Register Read	—	—	—	BASE	SEL3	SEL2	SEL1	SEL0
	W(B)	Channel Register Write	—	—	—	—	—	BASE	SELCH	
2H	R/W	Count Register Read/Write	C7	C6	C5	C4	C3	C2	C1	C0
3H	R/W		C15	C14	C13	C12	C11	C10	C9	C8
4H	R/W	Address Register Read/Write	A7	A6	A5	A4	A3	A2	A1	A0
5H	R/W		A15	A14	A13	A12	A11	A10	A9	A8
6H	R/W(B)		A23	A22	A21	A20	A19	A18	A17	A16
8H	R/W	Device Control Reg. Read/Write	AKL	RQL	EXW	ROT	CMP	DDMA	AHLD	MTM
9H	R/W		—	—	—	—	—	—	WEV	BHLD
0AH	R/W(B)	Mode Control Reg. Read/Write	TMODE		ADIR	AUTI	TDIR		—	W/$\overline{B}$
0BH	R(B)	Status Register Read	RQ3	RQ2	RQ1	RQ0	TC3	TC2	TC1	TC0
0CH	R	Temporary Reg. (lower) Read	T7	T6	T5	T4	T3	T2	T1	T0
0DH	R	Temporary Reg. (higher) Read	T15	T14	T13	T12	T11	T10	T9	T8
0EH	R/W(B)	Request Reg. Read/Write	—	—	—	—	SRQ3	SRQ2	SRQ1	SRQ0
0FH	R/W(B)	Mask Reg. Read/Write	—	—	—	—	M3	M2	M1	M0

49-000603B

Figure 12. Initialize Command Format



49-000604B

Channel Register

This command reads and writes the channel register that selects one of four DMA channels for programming the address, count, and mode control registers. Figure 13 shows the channel register read/write format.

Channel Register Read

SEL3-SEL0. These mutually exclusive bits show which of the four channels is currently selected for programming.

BASE. Base = 0. The current register may be read. During a write, the base and current registers will be written to simultaneously.

Base = 1. Only the base registers may be read or written to.

Channel Register Write

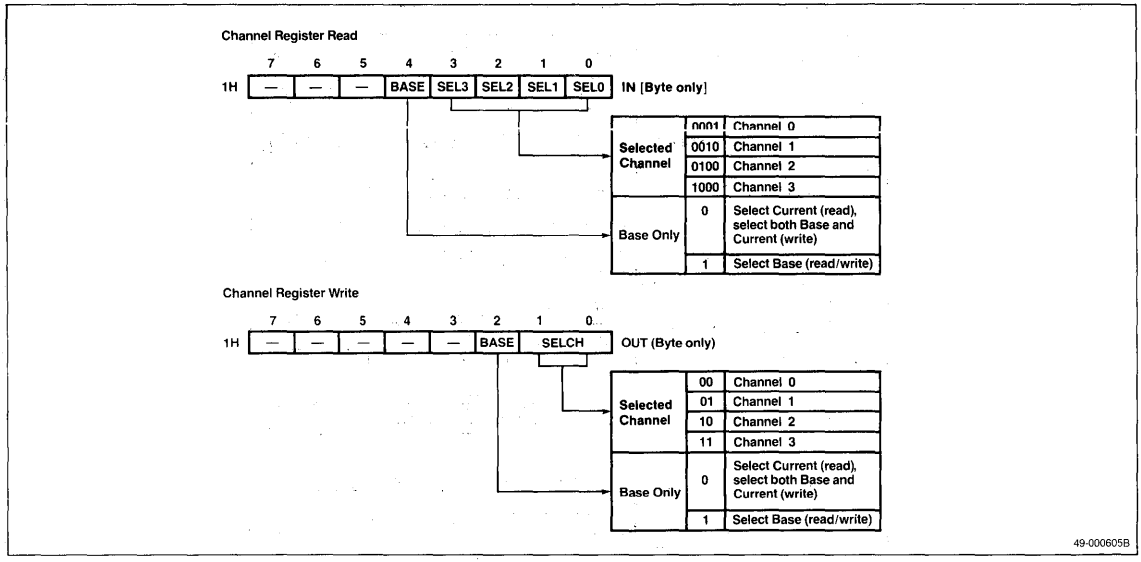
SELCH. This bit selects the channel to be programmed.

BASE. Base = 0. The current register may be read. During a write, the base and current registers will be written to simultaneously.

Base = 1. Only the base registers may be read or written to.

5g

Figure 13. Channel Register Format



Count Register Read/Write

When the 16-bit bus mode is selected, the IN/OUT instruction can directly transfer 16-bit data. The channel register selects one of the count registers. When bit 2 of the channel register write is cleared, a write to the count register updates both the base and current count registers with the new data. If bit 2 of the channel register write is set, a write to the count register only affects the base count register.

The base count registers hold the initial count value until a new count is specified. If autoinitialize is enabled, this value is transferred to the current count register when an $\overline{END}$ or $\overline{TC}$ is generated. For each DMA transfer, the current count register is decremented by one. Figure 14 shows the count register read/write format.

Address Register Read/Write

When a 16-bit data bus width is selected, the IN/OUT instruction can directly transfer the lower two bytes (4H and 5H) of the register. You must use the byte IN/OUT instruction with the upper byte (6H) of the register. The channel register selects one of the address registers. When bit 2 of the channel register is cleared, a write to the address register updates both the base and current address registers with the new data. If bit 2 of the channel register is set, a write to the address register only affects the base address register.

The base register holds the starting address value until a new setting is made and this value is transferred to the current address register during autoinitialization. For each DMA transfer, the current address register is updated ±2 during word transfer and ±1 during byte transfer. Figure 15 shows the address register read/write format.

Device Control Register Read/Write

The device control command reads from and writes to the device control register. When using a 16-bit data bus, use the word IN/OUT instruction to read and write 16-bit data. Figure 16 shows the device control register read/write format.

Figure 14. Count Register Read/Write Format

	7	6	5	4	3	2	1	0	
2H	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	In/Out
3H	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	In/Out

83-001951A

Figure 15. Address Register Read/Write Format

	7	6	5	4	3	2	1	0	
4H	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	In/Out
5H	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	In/Out
6H	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}	In/Out [Byte only]

83-001952A

Figure 16. Device Control Register Read/Write Format

	7	6	5	4	3	2	1	0	
8H	AKL	RQL	EXW	ROT	CMP	DDMA	AHLD	MTM	IN/OUT

Memory-to-Memory	0	Disable
	1	Enable
Fixed Address [1]	0	Disable for CH0
	1	Enable for CH0
Disable DMA Operation [2]	0	Enable
	1	Disable
Compressed Timing [3]	0	Normal
	1	Compressed
Rotational Priority	0	Fixed
	1	Rotational
Extended Writing [4]	0	Normal
	1	Extended
DMARQ Active Level	0	Active High
	1	Active Low
DMAAK Active Level	0	Active Low
	1	Active High

	7	6	5	4	3	2	1	0	
9H	—	—	—	—	—	—	WEV	BHLD	IN/OUT

Bus Mode	0	Bus Release
	1	Bus Hold
Wait Enable During Verify [5]	0	Disable
	1	Enable

Note:

[1] This bit is only used when MTM = 1, [memory-to-memory transfers].

[2] Disables HLDRQ to the CPU to prevent incorrect DMA operation while the μPD71071's registers are being initialized or modified.

[3] When 1, causes the μPD71071 to use compressed timing in the demand bus release mode or in the block mode.

[4] When EXW is 0, the write signal becomes active [normal write] during S3 and SW. When 1, the write signal becomes active during S2, S3, and SW. See figures 27-29.

[5] This bit enables or disables the wait state generated by the READY signal during a verify transfer.

49-000606C

5g

Mode Control Register Read/Write

This command reads from and writes to the mode control register to specify the operating mode for each channel. The channel register selects the mode control register to be programmed. This command must be issued by the byte IN/OUT instruction. Figure 17 shows the mode control register read/write format.

Status Register Read

This command reads the status register for the individual DMA channels. The register has DMA request states and terminal count or $\overline{END}$ information. This command must be issued by the byte IN instruction. Figure 18 shows the status register read format.

Temporary Register Read

When a 16-bit data bus is selected, the IN instruction will read 16-bit data with this command. The last data transferred in memory-to-memory transfer is stored in the temporary register. Figure 19 shows the temporary register read format.

Request Register Read/Write

This command reads from and writes to the request register to generate DMA requests by software for the four corresponding DMA channels. This command may be issued by the byte IN/OUT instruction. Figure 20 shows the reguest register read/write format.

Mask Register Read/Write

This command reads from and writes to the mask register to mask or unmask external DMA requests for the corresponding four DMA channels (DMARQ3-DMARQ0). This command may be issued by the byte IN/OUT instruction. Figure 21 shows the mask register read/write format.

DMA Transfer Modes

Figures 22-27 show state transition diagrams for the different modes of DMA transfer.

Figure 23 shows the state of a master μPD71071 when an input from a slave μPD71071 (cascaded μPD71071) is using the system bus.

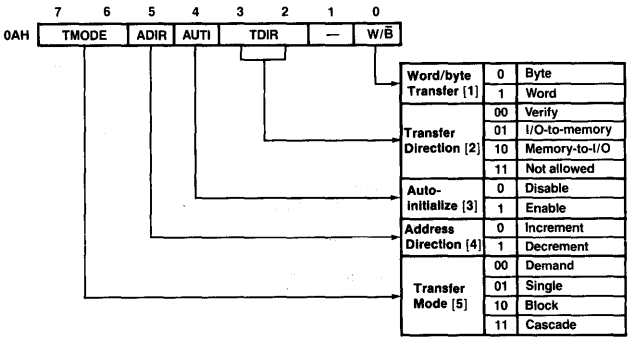
Transfer Timing

Figures 28-30 show μPD71071 timing waveforms.

Examples of System Configuration

Figures 31-32 show system configuration examples using the 8-bit μPD70108 CPU and the 16-bit μPD70116 CPU. The μPD71082 externally latches addresses and data.

Figure 17. Mode Control Register Read/Write Format



Note:

[1] This bit selects byte or word transfer for DMA transfers. This bit is used only in 16-bit data bus mode.

[2] These bits select the DMA transfer direction between memory and I/O. These bits are meaningless during memory-to-memory transfer.

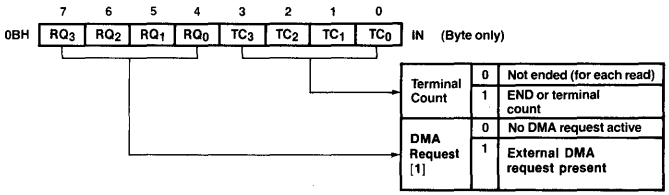
[3] Channel 0 and 1 must have the same AUTI bit value when performing memory-to-memory transfer.

[4] This bit decides the update direction of the Current Address Register. When ADIR is 0, the register increments by 1 for a byte transfer and by 2 for a word transfer. When ADIR is 1, the register decrements by 1 for a byte transfer and by 2 for a word transfer.

[5] These bits select the transfer mode during DMA transfer between memory and I/O and are meaningless during memory-to-memory transfer.

49-000608B

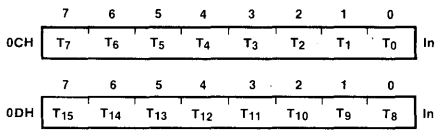
Figure 18. Status Register Read Format



Note:
[1] Bits RQ_3-RQ_0 will be set if an external hardware DMA request is pending even if its request bit is masked. Software-generated DMA requests, hardware reset, and software reset will not affect these bits.

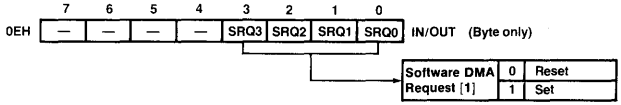
49-000607B

Figure 19. Temporary Register Read Format



83-001953A

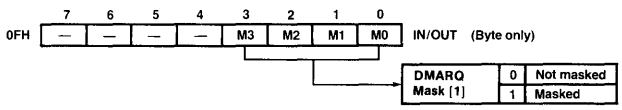
Figure 20. Request Register Read/Write Format



Note:
[1] In memory-to-memory applications, only bit SRQ0 will be cleared at terminal count or when an $\overline{END}$ input is present.

83-001887B

Figure 21. Mask Register Read/Write Format



Note:
[1] In memory-to-memory applications, only bits M0 and M1 will be set at terminal count or when an $\overline{END}$ input is present.

49-000609B

5g

Figure 22. Idle Cycle

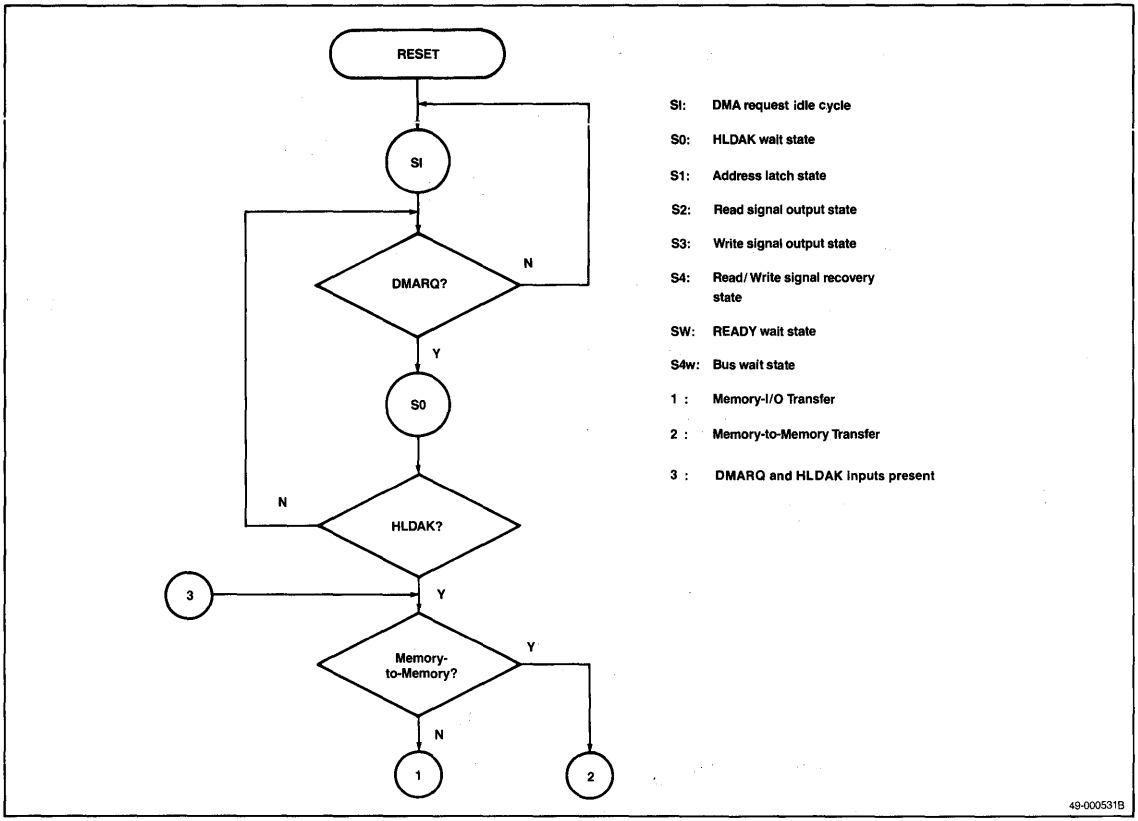


Figure 23. DMA Cycle, Cascade Mode

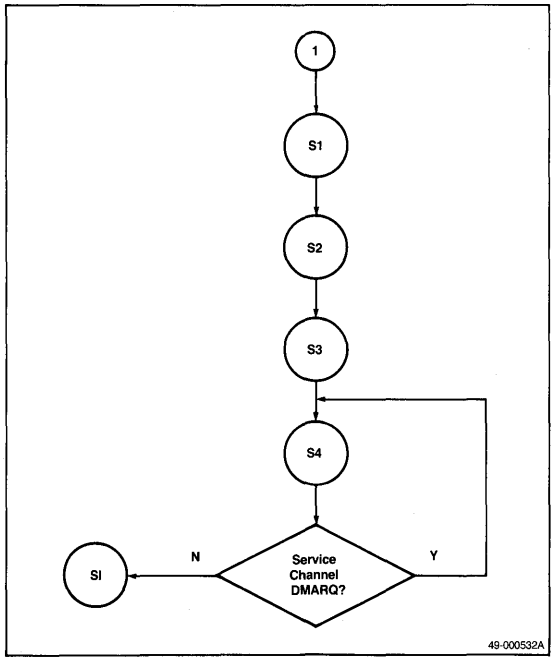
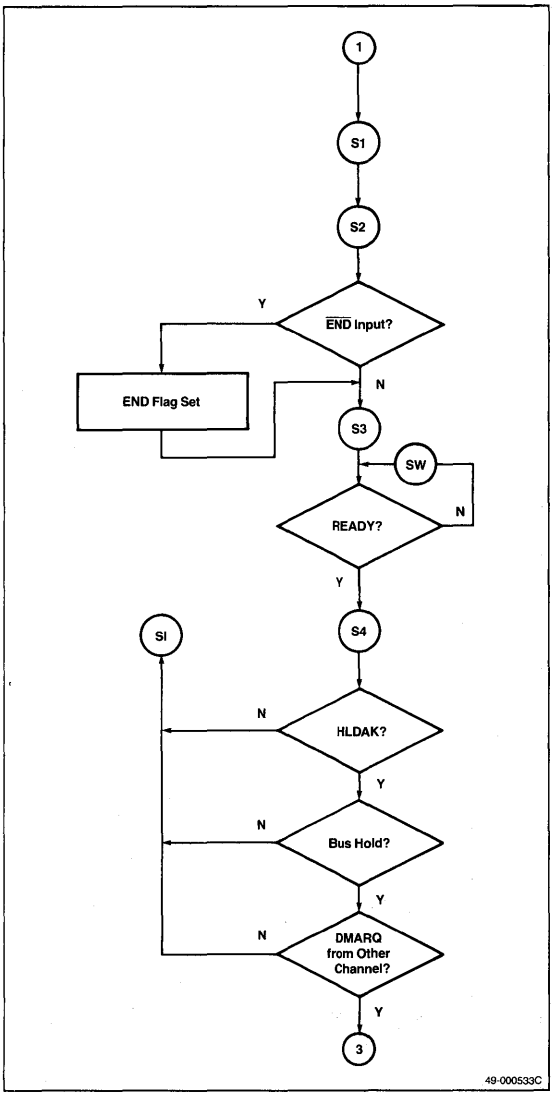
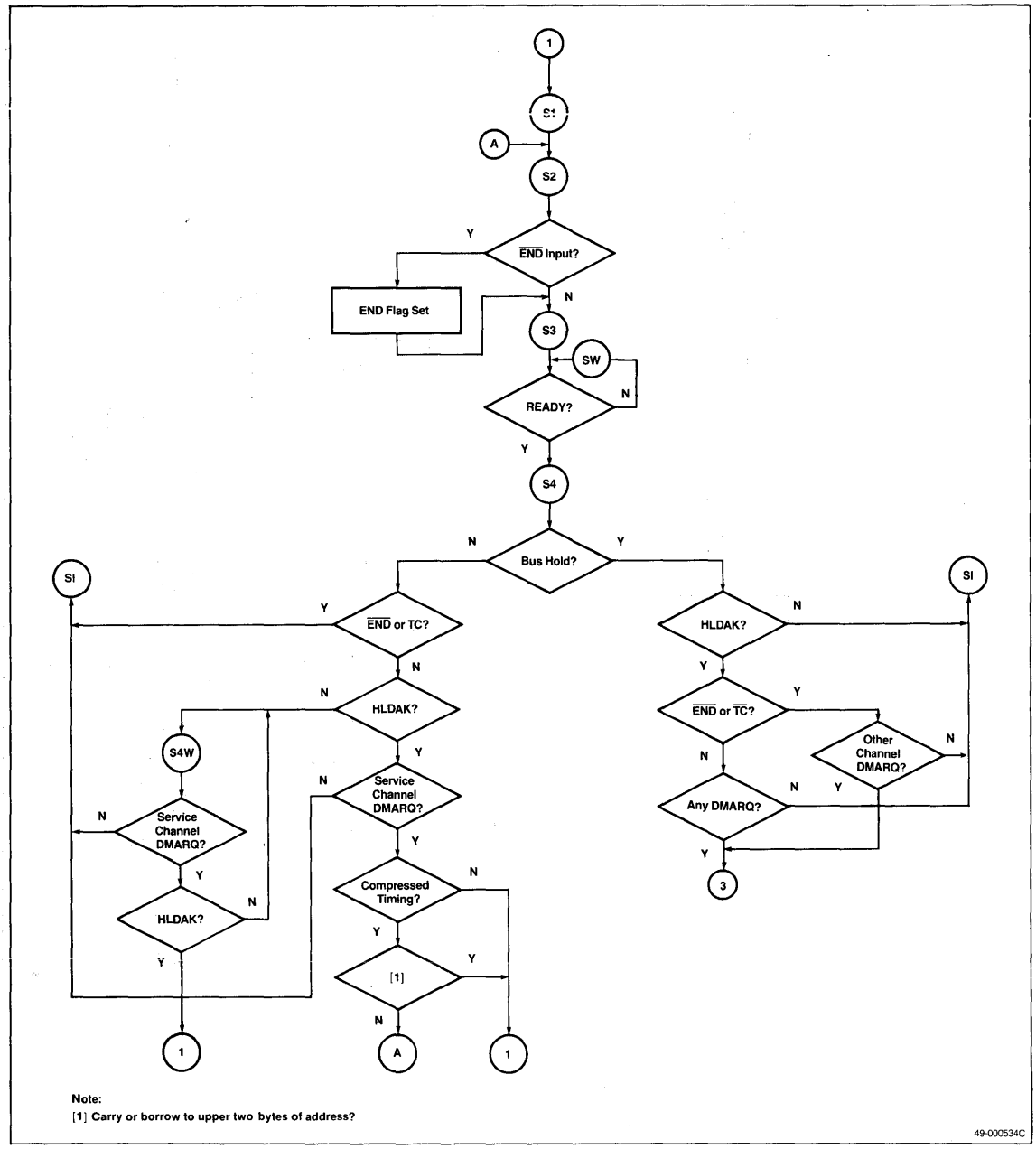


Figure 24. DMA Cycle, Single Mode



5g

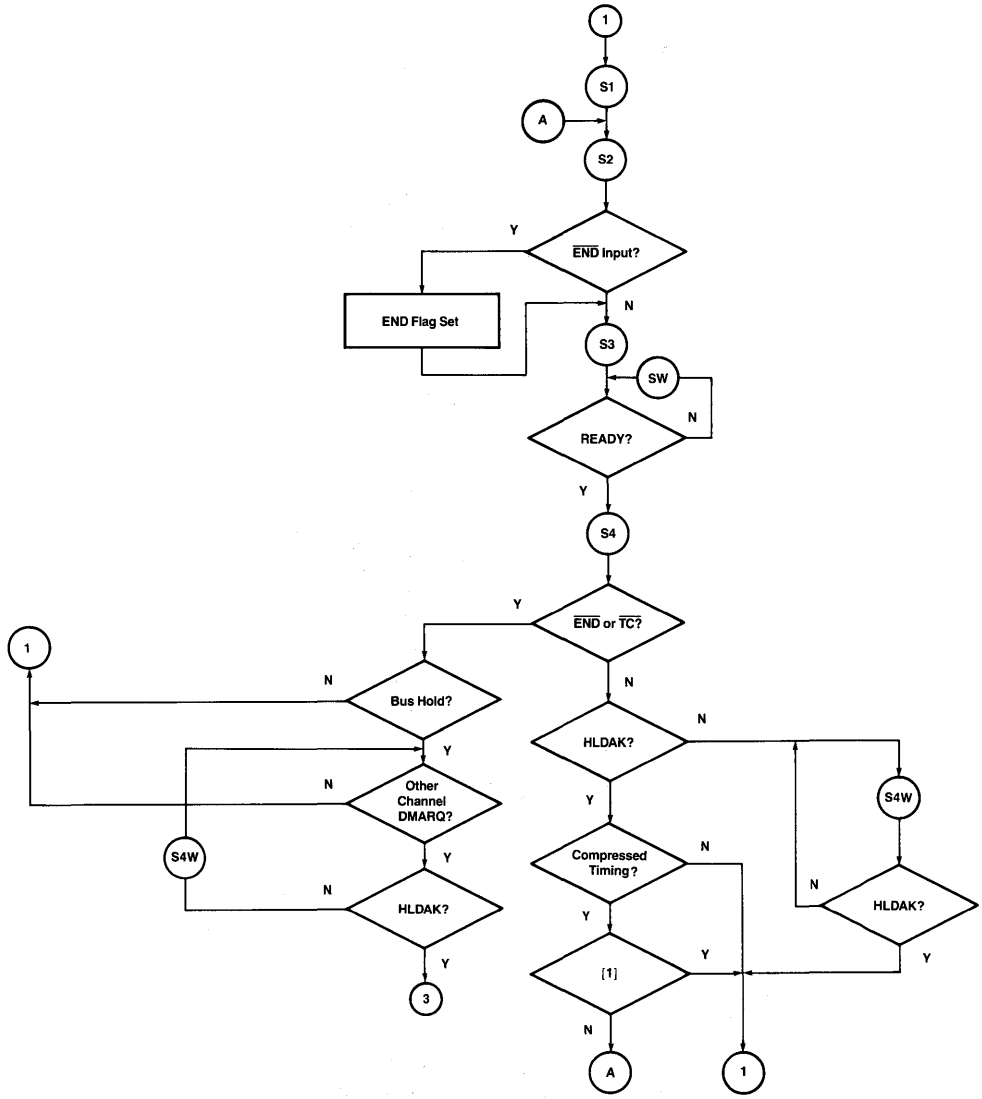
Figure 25. DMA Cycle, Demand Mode



Note:
[1] Carry or borrow to upper two bytes of address?

49-000534C

Figure 26. DMA Cycle, Block Mode

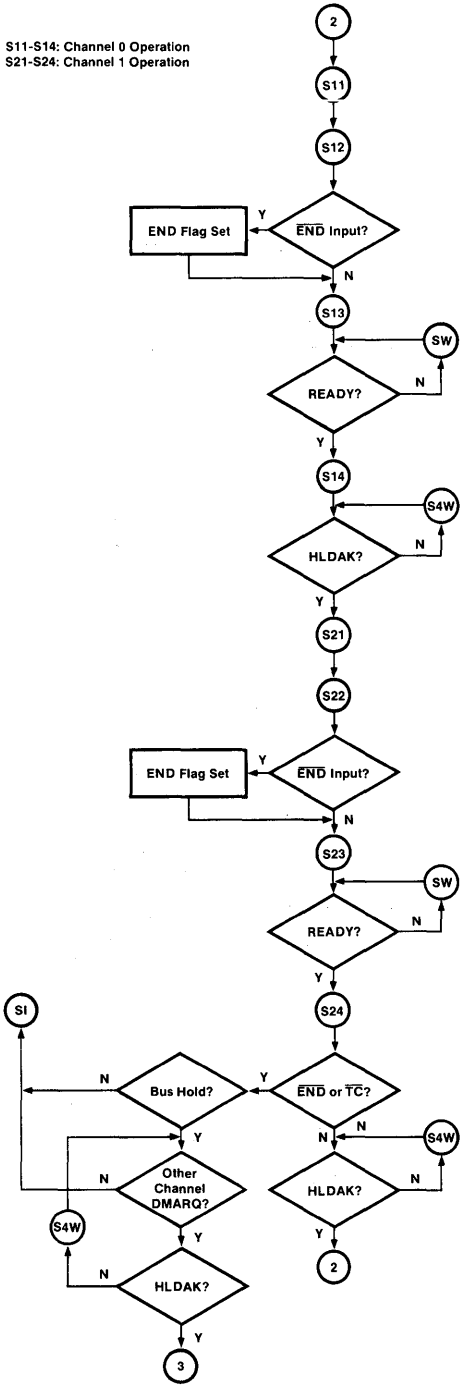


Note:
[1] Carry or borrow to upper two bytes of address?

49-000535C

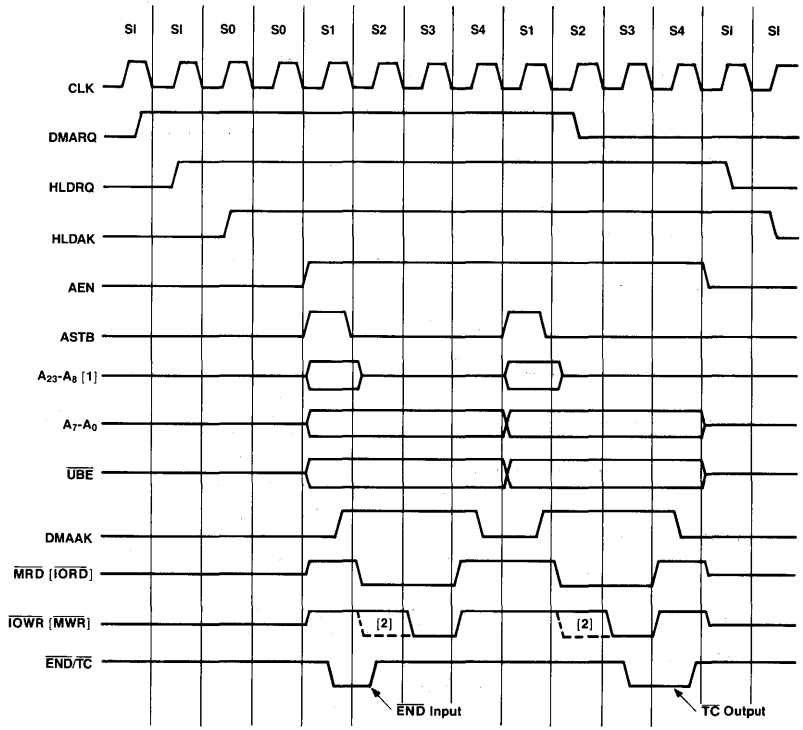
5g

Figure 27. DMA Cycle, Memory-to-Memory Transfer



83-001674C

Figure 28. Memory-I/O Transfer, Normal Timing



Note:

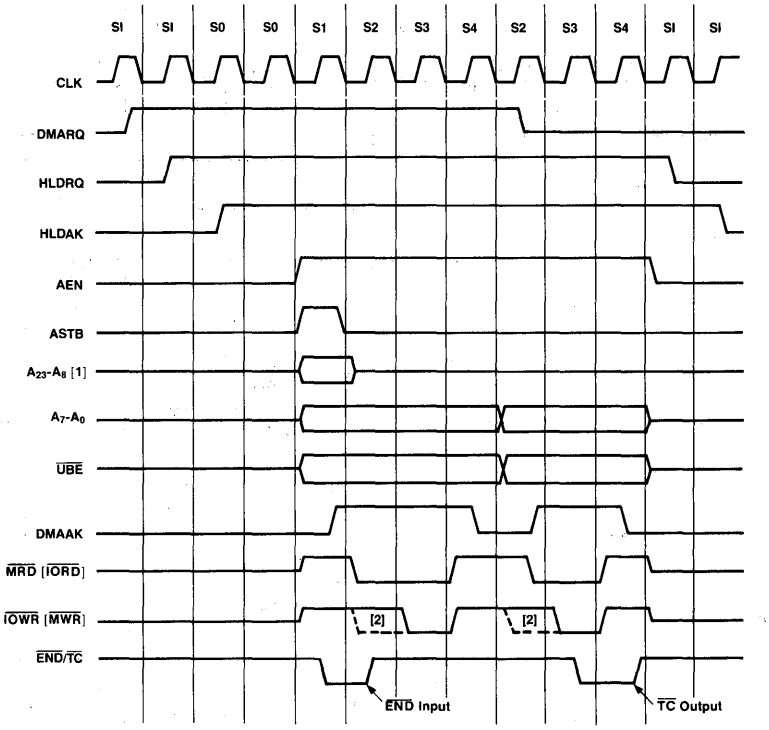
[1] When an 8-bit data bus is selected, D_{15}-D_8 are not used. Therefore, A_{23}-A_{16} are not multiplexed address/data signals and will have the same timing as A_7-A_0.

[2] The broken lines of the write signal are for extended write timing.

49-000537C

5g

Figure 29. Memory-I/O Transfer, Compressed Timing



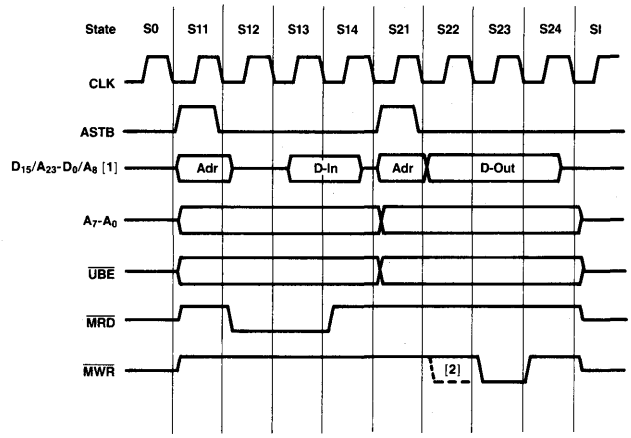
Note:

[1] When an 8-bit data bus is selected, D_{15}-D_8 are not used. Therefore, A_{23}-A_{16} are not multiplexed address/data signals and will have the same timing as A_7-A_0.

[2] The broken lines of the write signal are for extended write timing.

49-000538C

Figure 30. Memory-to-Memory Transfer



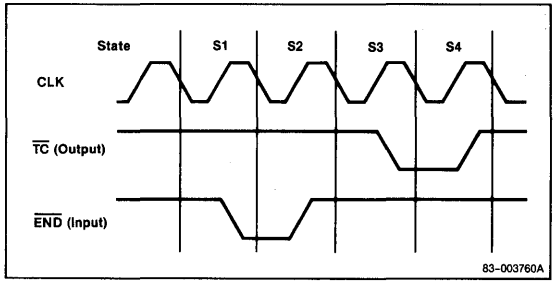
Note:

[1] When an 8-bit data bus is selected, D_{15}-D_8 are not used. Therefore, A_{23}-A_{16} are not multiplexed address/data signals and will have the same timing as A_7-A_0.

[2] The broken lines of the $\overline{MWR}$ signal are for extended write timing.

49-000539B

Figure 31. END/TC Input/Output



83-003760A

5g

Figure 32. System Configuration with μPD70108

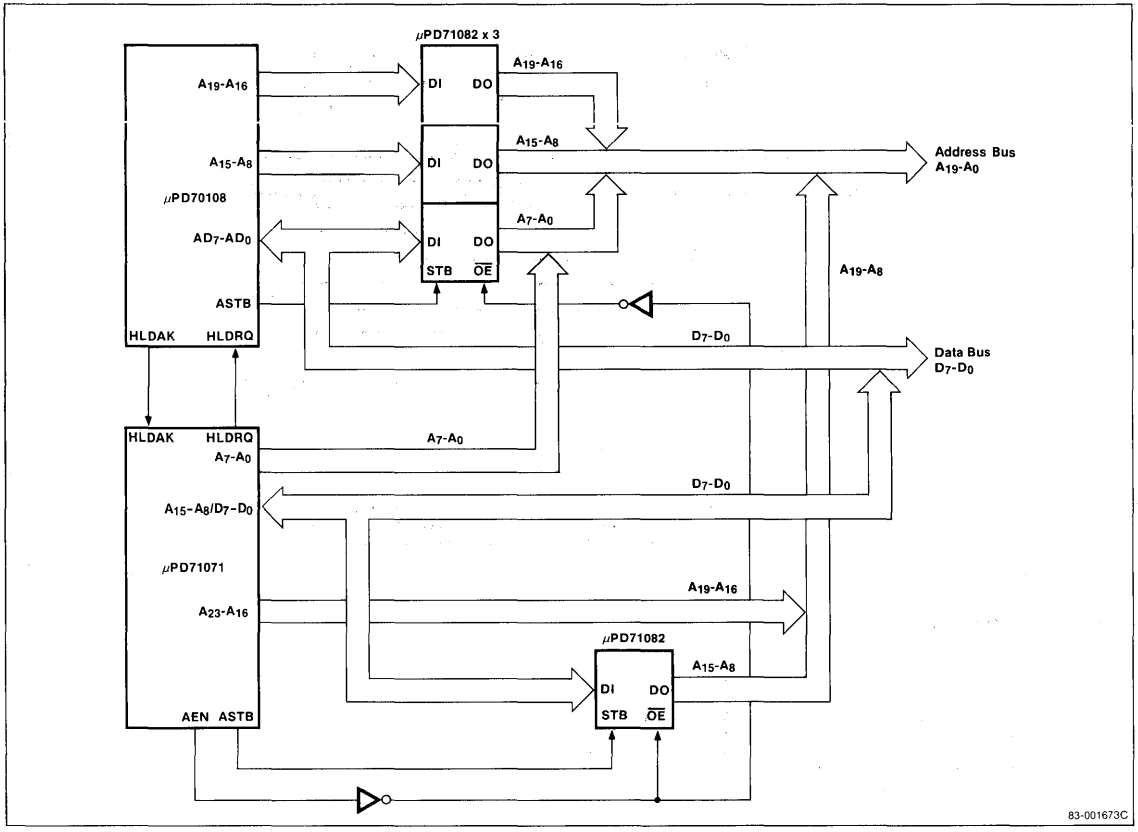
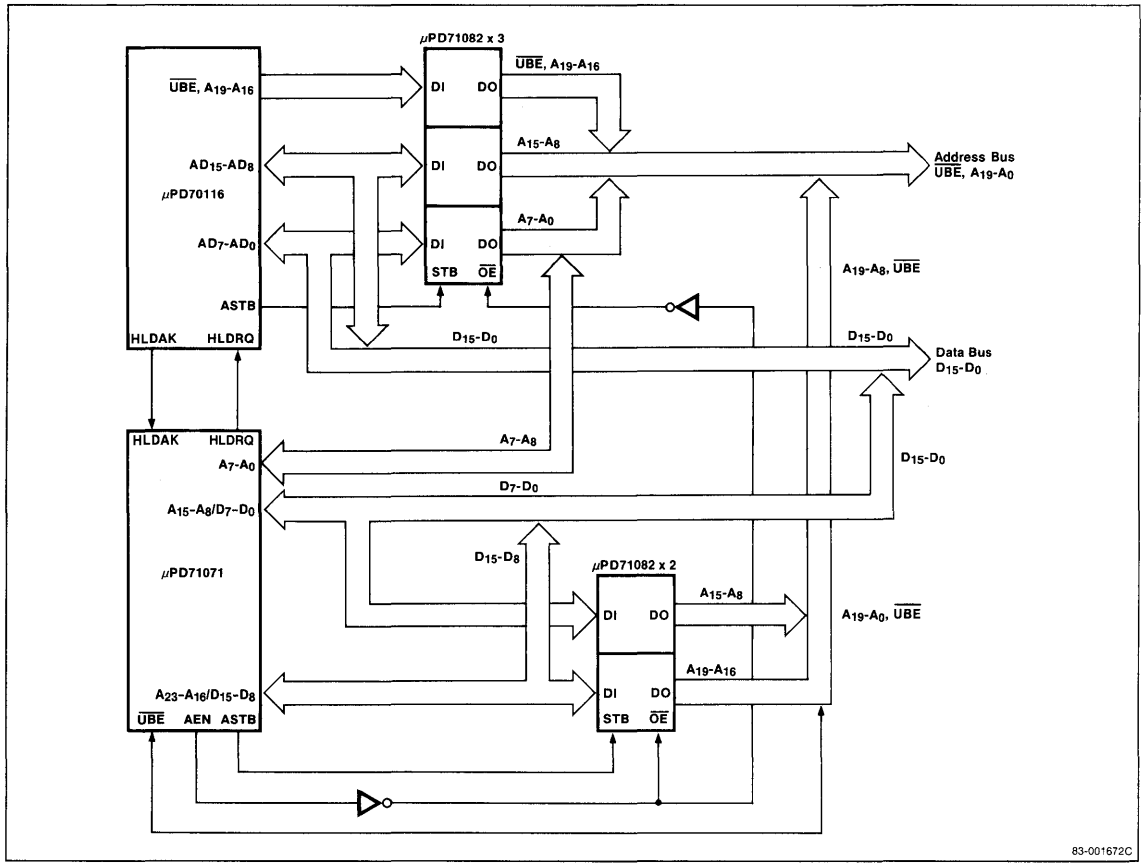


Figure 33. System Configuration with μPD70116



5g

Description

μPD71082 and μPD71083 are CMOS 8-bit transparent latches with three-state output buffers. They are used as bus buffers or bus multiplexers in microprocessor systems. Their high-drive capability makes them suitable for data latch, buffer, or I/O port applications.

Features

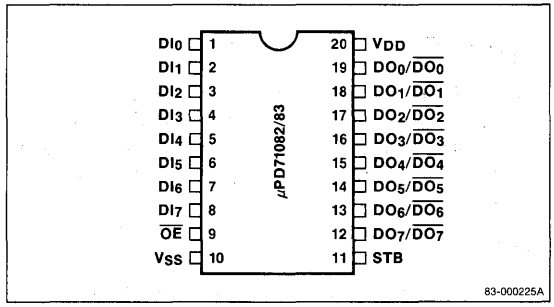
- CMOS technology
- 8-bit parallel data register
- Three-state output buffer
- High drive capability output buffer (I_{OL} = 12 mA)
- μPD8085A, 8048, 8086, 8088, μPD70108/116, and μPD70208/216 system compatible
- μPD71082 — non-inverted output; μPD71083 — inverted output
- Single +5 V ±10% power supply
- Transparent operation
- Industrial temperature range: −40 to +85°C

Ordering Information

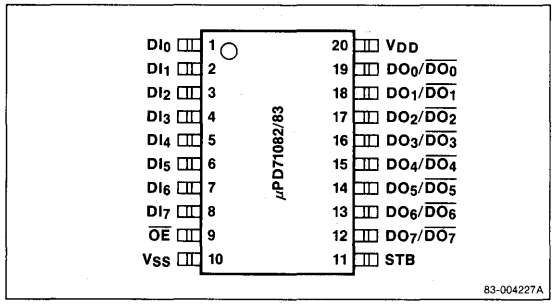
Part Number	Package	Output
μPD71082C	20-pin plastic DIP	Non-inverted
μPD71082G	20-pin plastic SOP	
μPD71083C	20-pin plastic DIP	Inverted
μPD71083G	20-pin plastic SOP	

Pin Configurations

20-Pin Plastic DIP



20-Pin Plastic SOP



Pin Identification

Symbol	Function
DI_0-DI_7	Data input, bits 0-7
DO_0-DO_7/ $\overline{DO_0}$-$\overline{DO_7}$	Data output, bits 0-7; non-inverted (μPD71082) or inverted (μPD71083)
STB	Strobe input
$\overline{OE}$	Output enable input
V_{DD}	+5 V power supply
V_{SS}	Ground

PIN FUNCTIONS

DI_0-DI_7 (Data Input)

DI_0-DI_7 are data input lines to the 8-bit data latch. Data on DI lines passes through the latch while STB is high. The data is latched to DO/$\overline{DO}$ with the falling edge of STB.

DO_0-DO_7/$\overline{DO}_0$-$\overline{DO}_7$ (Data Output)

DO_0-DO_7/$\overline{DO}_0$-$\overline{DO}_7$ are the three-state data output lines from the 8-bit data latch. When $\overline{OE}$ is high, these lines go into the high-impedance state. When $\overline{OE}$ is low, data from the latch is output, either non-inverted (μPD71082) or inverted (μPD71083).

STB (Strobe)

STB is the input strobe signal for the 8-bit latch. When STB is high, data on the DI lines passes through the 8-bit latch. Data is latched on the falling edge of STB. When STB is low, the DO_0-DO_7/$\overline{DO}_0$-$\overline{DO}_7$ outputs do not change.

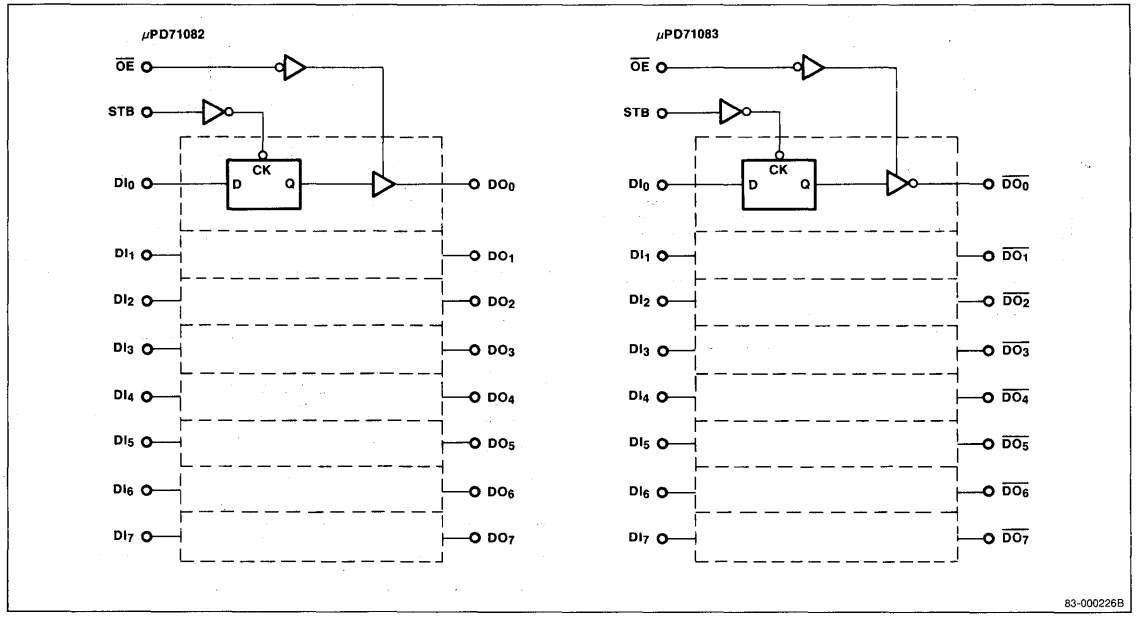
$\overline{OE}$ (Output Enable)

$\overline{OE}$ input is the output enable signal for the three-state DO/$\overline{DO}$ lines. When $\overline{OE}$ is high, DO/$\overline{DO}$ lines are high impedance. When $\overline{OE}$ is low, data from the 8-bit latch is output to DO_0-DO_7/$\overline{DO}_0$-$\overline{DO}_7$. See table 1.

Table 1. Latch Operation

STB	$\overline{OE}$	DO_0-DO_7/$\overline{DO}_0$-$\overline{DO}_7$	8-Bit Data Latch
Low	Low	Latched data from 8-bit data latch is enabled	DI line data has been latched with falling edge of STB (high to low)
	High	High impedance	
High	Low	Data on DI_0-DI_7	DI passed through to DO/$\overline{DO}$
	High	High impedance	

Block Diagram



FUNCTIONAL DESCRIPTION

The μPD71082 and μPD71083 are 8-bit data latches strobed by the STB signal. They have high-drive capability output buffers controlled by the $\overline{OE}$ signal. Data on the DI lines is latched by the trailing edge of STB (high to low). When STB is high, data passes through the latch. When $\overline{OE}$ is high, DO lines are high impedance. When $\overline{OE}$ is low, the contents of the latches are output on DO_0-DO_7. The DO lines are isolated from $\overline{OE}$ switching noise.

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = 25°C; V_{SS} = 0 V

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−1.0 to V_{DD} + 1 V
Output voltage, V_O	−0.5 to V_{DD} + 0.5 V
Power dissipation, P_{DMAX}, DIP	500 mW
Power dissipation, P_{DMAX}, SO	200 mW
Operating temperature, T_{opt}	−40 to +85°C
Storage temperature, T_{stg}	−65 to +150°C

Exposing the device to stresses above those listed in the absolute maximum ratings could cause permanent damage. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC Characteristics

T_A = −40 to +85°C; V_{DD} = 5 V ±10%

Parameter	Symbol	Min	Max	Units	Conditions
Input voltage, high	V_{IH}	2.2		V	V_{OL} = 0.45 V V_{OH} = V_{DD} − 0.8 V
Input voltage, low	V_{IL}		0.8	V	V_{OL} = 0.45 V V_{OH} = V_{DD} − 0.8 V
Output voltage, high	V_{OH}	V_{DD} − 0.8		V	I_{OH} = −4 mA
Output voltage, low	V_{OL}		0.45	V	I_{OL} = 12 mA
Input current	I_I	−1.0	1.0	μA	V_I = V_{DD}, V_{SS}
Leakage current, high impedance	I_{OFF}	−10	10	μA	$\overline{OE}$ = V_{DD}
Power supply current (static)	I_{DD}		80	μA	V_I = V_{DD}, V_{SS}
Power supply current (dynamic)	I_{DDdyn}		20	mA	f_{in} = 10 MHz C = 200 pF

Capacitance

T_A = 25°C; V_{DD} = +5 V

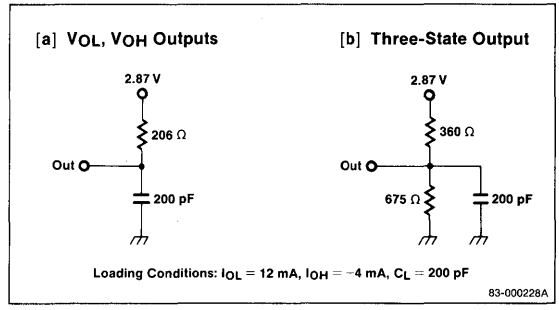
Parameter	Symbol	Min	Max	Units	Conditions
Input capacitance	C_{in}		12	pF	f = 1 MHz

AC Characteristics

T_A = −40 to +85°C; V_{DD} = 5 V ±10%

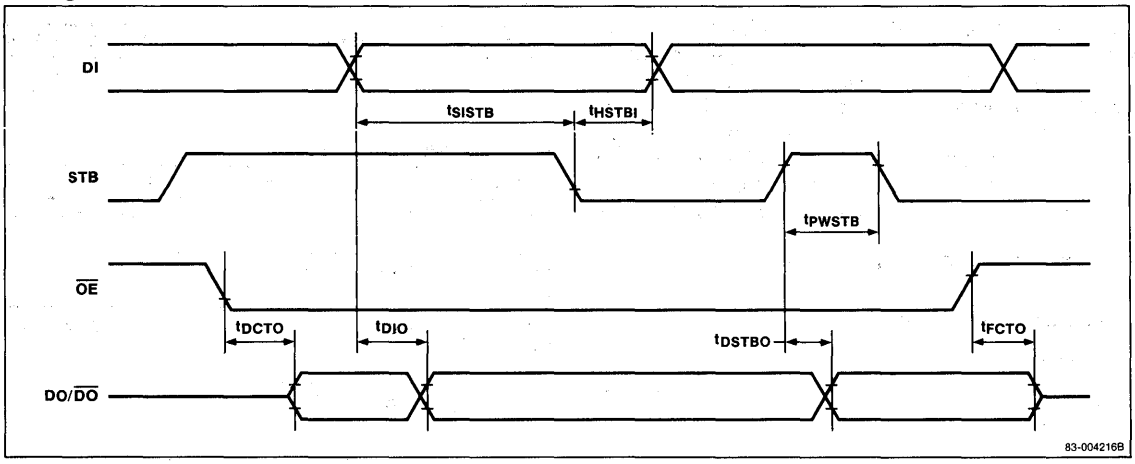
Parameter	Symbol	Min	Max	Units	Conditions
Input to output delay	t_{DIO}	5	40	ns	Loading circuit (a)
STB to output delay	t_{DSTBO}	10	60	ns	
Data float time from $\overline{OE}$ high	t_{FCTO}	5	30	ns	Loading circuit (b)
Data output delay from $\overline{OE}$ low	t_{DCTO}	10	40	ns	
Input to STB setup time	t_{SISTB}	0		ns	Loading circuit (a)
Input to STB hold time	t_{HSTBI}	25		ns	
STB high pulse width	t_{PWSTB}	20		ns	
Signal rise time	t_{LH}		20	ns	0.8 to 2.0 V
Signal fall time	t_{HL}		12	ns	2.0 to 0.8 V

Loading Circuits for AC Testing

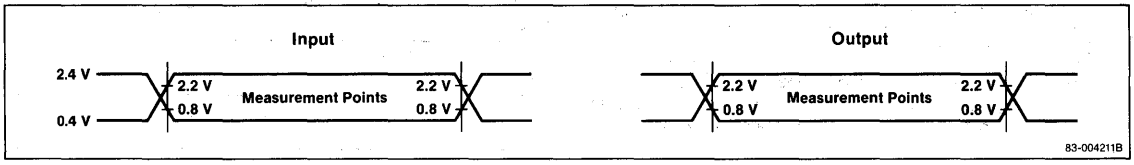


5h

Timing Waveforms



Timing Measurement Points



μPD71084 Clock Pulse Generator/Driver

Description

The μPD71084 is a clock pulse generator/driver for microprocessors including the V20® and V30® and their peripherals using NEC's high-speed CMOS technology.

Features

- CMOS technology
- Clock pulse generator/driver for μPD70108/70116 or other CMOS or NMOS CPUs and their peripherals
- Frequency source can be crystal or external clock input
- Reset signal with Schmitt-trigger circuit for CPU or peripherals
- Bus ready signal with two-bus system synchronization
- Clock synchronization with other μPD71084s
- Single +5 V ±10% power supply
- Industrial temperature range: −40 to +85°C

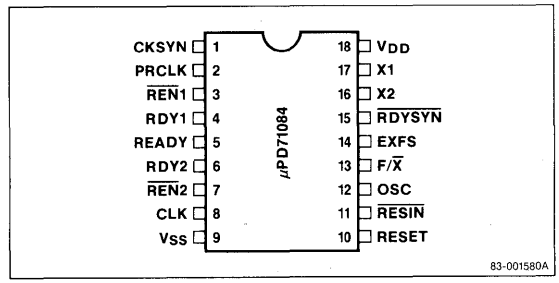
Ordering Information

Part Number	CLK Out, Max	Package
μPD71084C-8	8 MHZ	18-pin plastic DIP
C-10	10 MHz	
G-8	8 MHz	20-pin plastic SOP

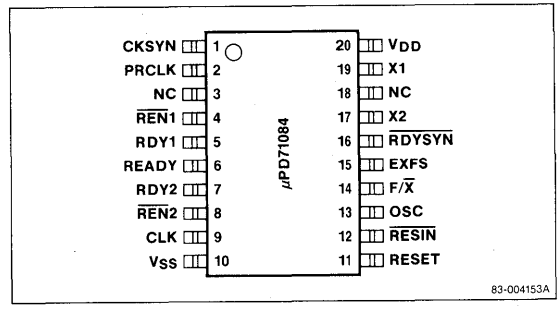
V20 and V30 are registered trademarks of NEC Corporation.

Pin Configurations

18-Pin Plastic DIP



20-Pin Plastic SOP



5i

Pin Identification

Symbol	Function
CKSYN	Clock synchronization input
PRCLK	Peripheral clock output
$\overline{REN1}$	Bus ready enable input 1
RDY1	Bus ready input 1
READY	Ready output
RDY2	Bus ready input 2
$\overline{REN2}$	Bus ready enable input 2
CLK	Processor clock output
V_{SS}	Ground potential
RESET	Reset output
$\overline{RESIN}$	Reset input
OSC	Oscillator output
F/$\overline{X}$	External frequency source/crystal select
EXFS	External frequency source input
$\overline{RDYSYN}$	Ready synchronization select input
X2	Crystal input
X1	Crystal input
V_{DD}	+5 V power supply
NC	No connection

PIN FUNCTIONS

X1, X2 (Crystal)

When the F/$\overline{X}$ input is low, a crystal connected to X1 and X2 will be the frequency source to generate clocks for a CPU and its peripherals. The crystal frequency should be three times the frequency of CLK.

EXFS (External Frequency)

EXFS is the external frequency input in the external TTL frequency source mode (F/$\overline{X}$ high). A TTL-level clock signal three times the frequency of the CLK output should be used for the source.

F/$\overline{X}$ (Frequency/Crystal Select)

F/$\overline{X}$ input selects whether an external TTL-level input or an external crystal input is the frequency source of the CLK output. When F/$\overline{X}$ is low, CLK is generated from the crystal connected to X1 and X2. When F/$\overline{X}$ is high, CLK is generated from an external TTL-level frequency input on the EXFS pin. At the same time, the internal oscillator circuit will stop and the OSC output will be high.

CLK (Processor Clock)

CLK output supplies the CPU and its local bus peripherals. CLK is a 33% duty cycle clock, one-third the frequency of the frequency source. The CLK output is +0.4 V higher than the other outputs.

PRCLK (Peripheral Clock)

PRCLK output supplies a 50% duty cycle clock at one-half the CLK frequency to drive peripheral devices.

OSC (Oscillator)

OSC outputs a signal at the same frequency as the crystal input. When EXFS is selected, the OSC output is powered down, and its output will be high.

CKSYN (Clock Synchronization)

CKSYN input synchronizes one μPD71084 to other μPD71084s. A high level at CKSYN resets the internal counter, and a low level enables it to count.

$\overline{RESIN}$ (Reset)

This Schmitt-trigger input generates the RESET output. It is used as a power-on reset.

RESET (Reset)

This output is a reset signal for the CPU. Reset timing is provided by the $\overline{RESIN}$ input to a Schmitt-trigger input gate and a flip-flop which will synchronize the reset timing to the falling edge of CLK. Power-on reset can be provided by a simple RC circuit on the $\overline{RESIN}$ input.

RDY1, RDY2 (Bus Ready)

A peripheral device drives the RDY1 or RDY2 inputs to signal that the data on the system bus has been received or is ready to be sent. $\overline{REN1}$ and $\overline{REN2}$ enable the RDY1 and RDY2 signals.

$\overline{REN1}$, $\overline{REN2}$ (Address Enable)

$\overline{REN1}$ and $\overline{REN2}$ inputs qualify their respective RDY inputs.

$\overline{RDYSYN}$ (Ready Synchronization Select)

$\overline{RDYSYN}$ input selects the mode of READY signal synchronization. A low-level signal makes the synchronization a two-step process. Two-step synchronization is used when RDY1 or RDY2 are not synchronized to the microprocessor clock and therefore cannot be guaranteed to meet the READY setup time. A high-level signal makes synchronization a one-step process. One-step

synchronization is used when RDY1 and RDY2 are synchronized to the processor clock. See Block Diagram.

READY (Ready)

The READY output signal to the processor is synchronized by the RDY inputs to the processor CLK. READY is cleared after RDY goes low and the guaranteed hold time of the processor has been met.

CRYSTAL

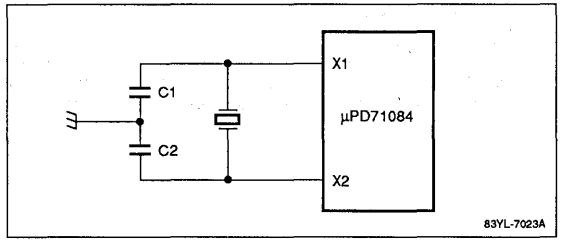
The oscillator circuit of the μPD71084 works with a parallel-resonant, fundamental mode, "AT-cut" crystal connected to pins X1 and X2.

Figure 1 shows the recommended circuit configuration. Capacitors C1 and C2 are required for frequency stability. The values of C1 and C2 (C1 = C2) can be calculated from the load capacitance (C_L) specified by the crystal manufacturer.

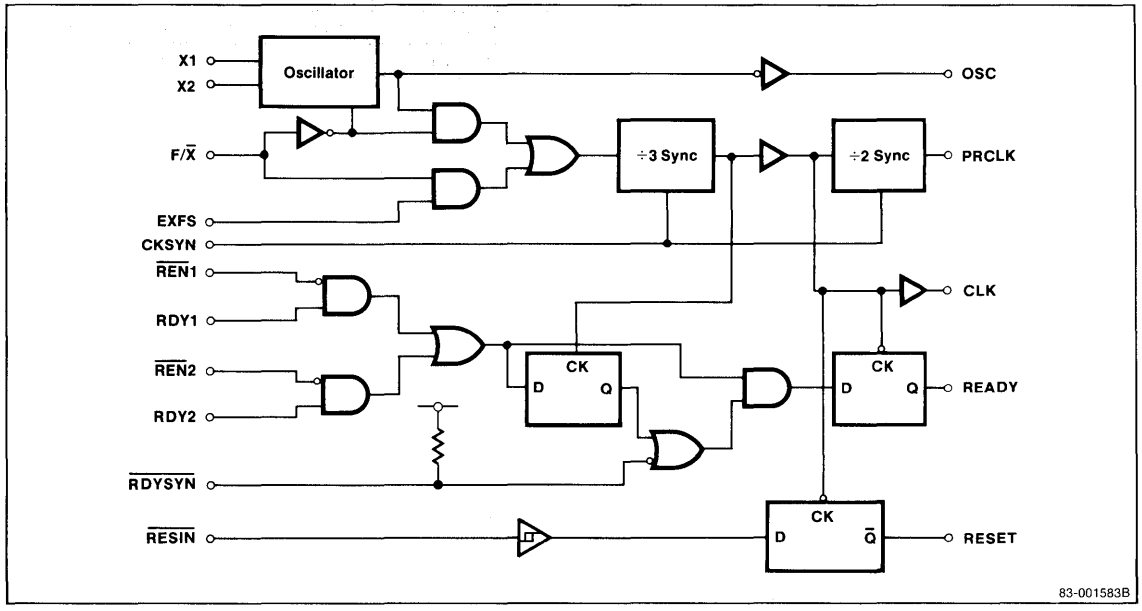
$$C_L = \frac{C1 \times C2}{C1 + C2} + C_S$$

C_S is any stray capacitance in parallel with the crystal, such as the μPD71084 input capacitance C_I

Figure 1. Crystal Configuration Circuit



μPD71084 Block Diagram



5i

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = 25°C; V_{SS} = 0 V

Power supply voltage, V_{DD}	−0.5 to + 7.0 V
Input voltage, V_I	−1.0 V to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 V to V_{DD} + 0.5 V
Operating temperature, T_{OPT}	−40 to +85°C
Storage temperature, T_{STG}	−65 to +150°C
Power dissipation, P_D (DIP)	500 mW
Power dissipation, P_D (SOP)	200 mW

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

DC Characteristics

T_A = −40 to +85°C; V_{DD} = 5 V ±10%

Parameter	Symbol	Min	Max	Unit	Conditions
Input voltage, high	V_{IH}	2.2		V	
		2.6		V	$\overline{RESIN}$ input
Input voltage, low	V_{IL}		0.8	V	
Output voltage, high	V_{OH}	V_{DD} − 0.4		V	CLK output, I_{OH} = −4 mA
		V_{DD} − 0.8		V	I_{OH} = −4 mA
Output voltage, low	V_{OL}		0.45	V	I_{OL} = 4 mA
Input leakage current	I_{IN}	−1.0	1.0	μA	
		−400	1.0	μA	$\overline{RDYSYN}$ input
$\overline{RESIN}$ hysteresis		0.25		V	
Power supply current (static)	I_{DD}		200	μA	
Power supply current (dynamic)	I_{DDdyn}		30	mA	f_{in} = 24 MHz

Capacitance

T_A = 25°C; V_{DD} = +5 V

Parameter	Symbol	Min	Max	Unit	Conditions
Input capacitance	C_{in}		12	pF	f = 1 MHz

AC Characteristics

T_A = −40 to +85°C; V_{DD} 5 V ±10%

Parameter	Symbol	Min	Max	Unit	Conditions
EXFS high	t_{EHEL}	16		ns	At 2.2 V
EXFS low	t_{ELEH}	16		ns	At 0.8 V
EXFS period	t_{ELEL}	40		ns	
XTAL frequency		12	25	MHz	
RDY1, 2 setup to CLK↓	t_{R1VCL}, t_{R1VCH}	35		ns	
RDY1, 2 hold to CLK↓	t_{CLR1X}	0		ns	
$\overline{RDYSYN}$ setup to CLK↓	t_{RSYVCL}	50		ns	
$\overline{RDYSYN}$ hold to CLK	t_{CLRSYX}	0		ns	
$\overline{REN}$1, 2 setup to RDY1, 2	t_{A1R1V}	15		ns	
$\overline{REN}$1, 2 hold to CLK↓	t_{CLA1X}	0		ns	
CKSYN setup to EXFS	t_{YHEH}	20		ns	
CKSYN hold to EXFS	t_{EHYL}	20		ns	
CKSYN width	t_{YHYL}	$2t_{ELEL}$		ns	
$\overline{RESIN}$ setup to CLK	t_{I1HCL}	65		ns	
$\overline{RESIN}$ hold to CLK	t_{CLI1H}	20		ns	
CLK cycle period	t_{CLCL}	125		ns	
CLK high	t_{CHCL}	41		ns	3 V, f_{OSC} = 24 MHz (Note 1)
		1/3 (t_{CLCL}) +2		ns	1.5 V, f_{OSC} ≤ 24 MHz (Note 2)
CLK low	t_{CLCH}	68		ns	1.5 V, f_{OSC} = 24 MHz (Note 1)
		2/3 (t_{CLCL}) −15		ns	1.5 V, f_{OSC} ≤ 24 MHz (Note 2)
CLK rise and fall time	t_{CLH}, t_{CHL}		10	ns	1.5 to 3.5 V, 3.5 to 1.5 V
PRCLK high	t_{PHPL}	t_{CLCL} − 20		ns	(Note 3)
PRCLK low	t_{PLPH}	t_{CLCL} − 20		ns	(Note 3)
READY inactive to CLK↓	t_{RYLCL}		8	ns	
READY active to CLK↑	t_{RYHCH}		8	ns	

Parameter	Symbol	Min	Max	Unit	Conditions
CLK to RESET delay	t_{CLIL}		40	ns	
CLK to PRCLK ↑ delay	t_{CLPH}		22	ns	
CLK to PRCLK ↓ delay	t_{CLPL}		22	ns	
OSC CLK↑ delay	t_{OLCH}	−5	22	ns	
OSC CLK↓ delay	t_{OLCL}	2	35	ns	
Signal rise time (except CLK)	t_{LH}		20	ns	0.8 to 2.0 V
Signal fall time (except CLK)	t_{HL}		12	ns	2.0 to 0.8 V

Notes:

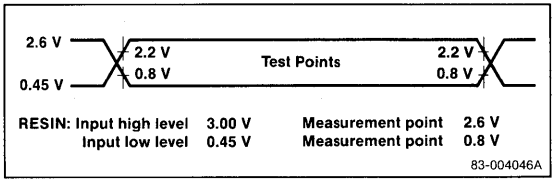
(1) Test points are specified in accordance with V-Series CMOS peripherals.

(2) Test points are specified in accordance with the μPD8284.

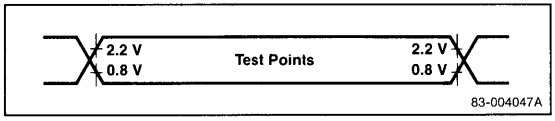
(3) t_{PHPL} + t_{PLPH} total must meet a minimum of 250 ns.

Timing Waveforms

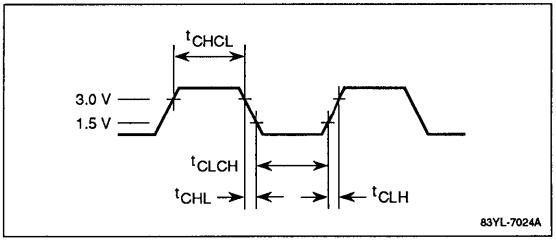
AC Test Input (Except $\overline{RESIN}$)



AC Test Output (Except CLK)

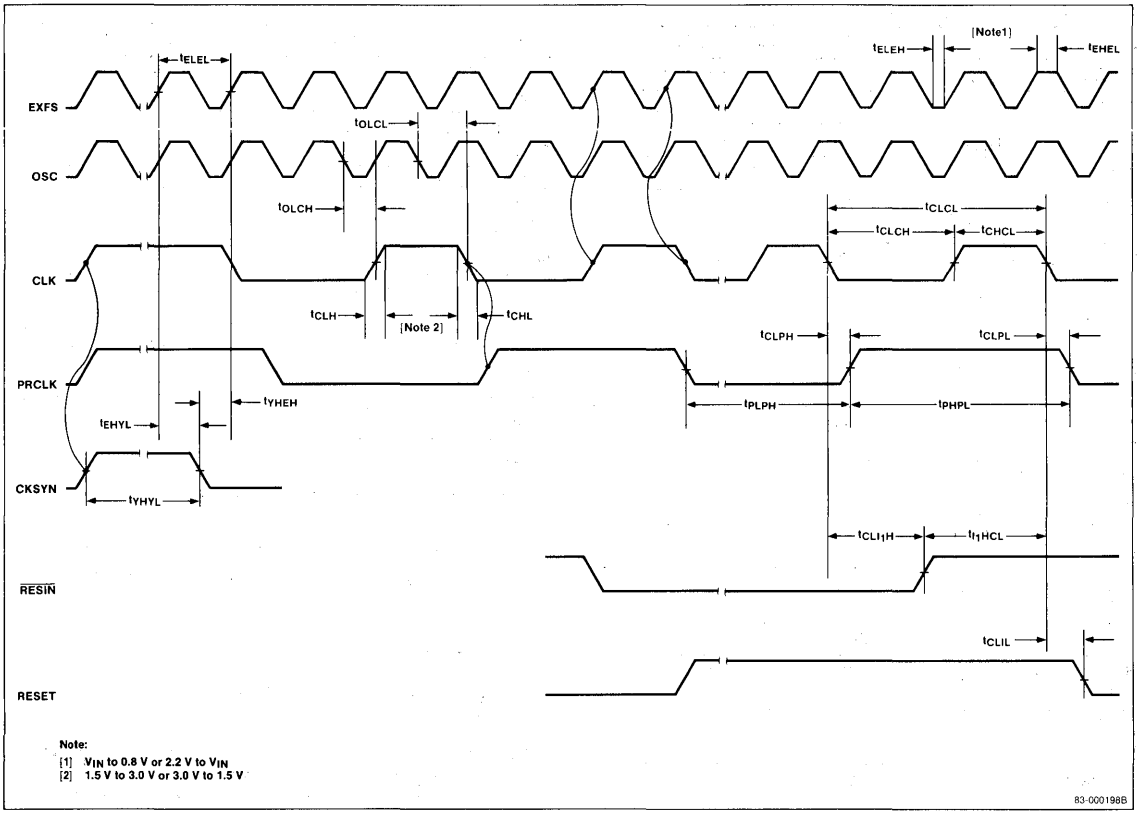


CLK Output



5i

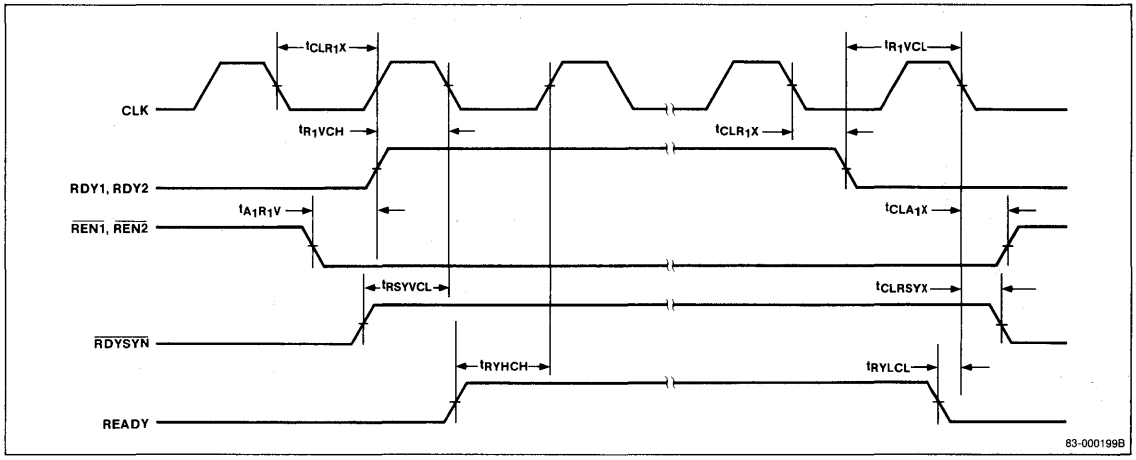
CLK, RESET Signals



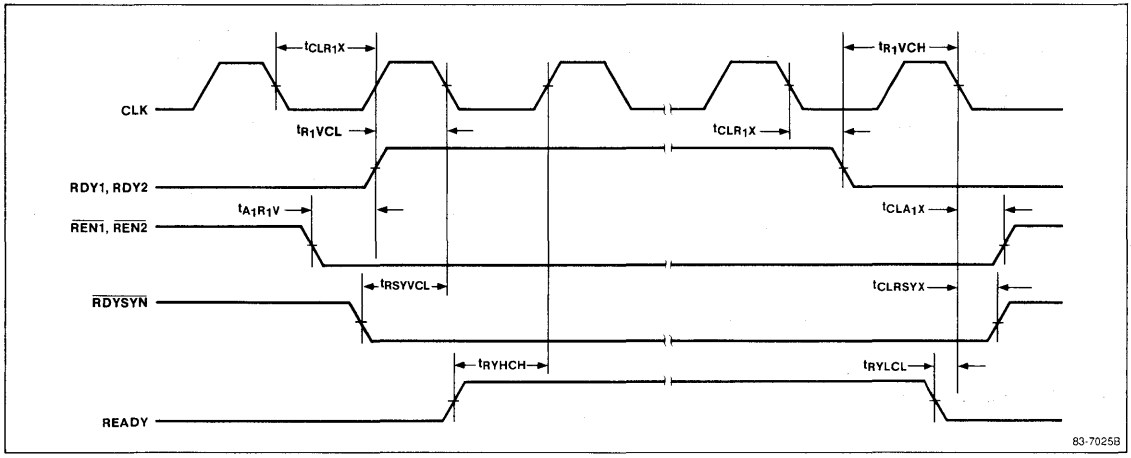
Note:
[1] V_{IN} to 0.8 V or 2.2 V to V_{IN}
[2] 1.5 V to 3.0 V or 3.0 V to 1.5 V

83-000198B

READY Pin ($\overline{RDSYN}$ = High)

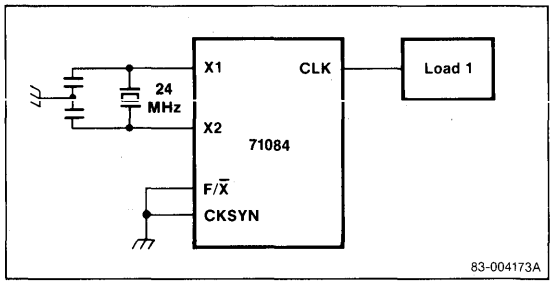


READY Pin ($\overline{RDSYN}$ = Low)

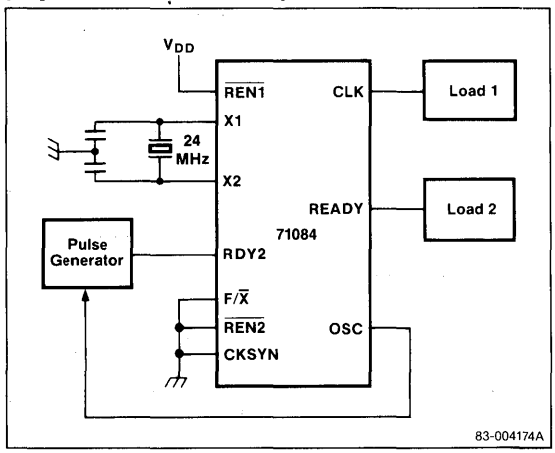


5i

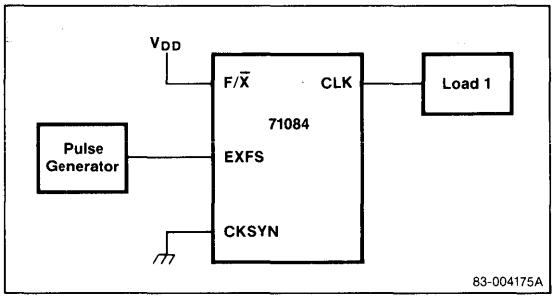
Test Circuit for CLK High or Low Time (Crystal Oscillation Mode)



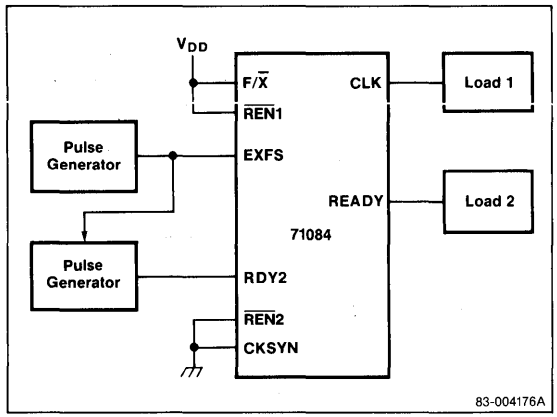
Test Circuit for CLK to READY (Crystal Oscillation Mode)



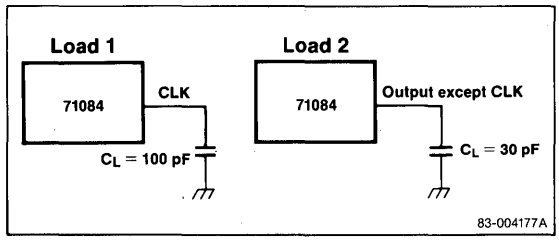
Test Circuit for CLK High or Low Time (EXFS Oscillation Mode)



Test Circuit for CLK to READY (EXFS Oscillation Mode)



Loading Circuits



Description

μPD71086 and μPD71087 are 8-bit, bidirectional bus buffer/drivers with three-state outputs. The system bus outputs are noninverted (μPD71086) or inverted (μPD71087). These devices are used to expand CPU bus drive capability. The input/output lines are isolated from $\overline{OE}$ and BUF$\overline{R}$/W switching noise.

Features

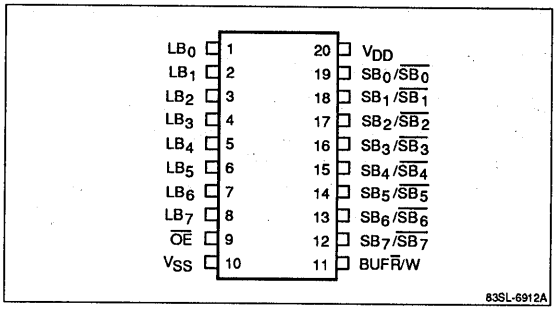
- CMOS technology
- Bidirectional 8-bit parallel bus buffer
- Three-state output
- High system bus-drive capability (I_{OL} = 12 mA)
- Compatible with μPD70108/116, μPD70208/216, and other CMOS or NMOS designs
- μPD71086: noninverted system bus output
  μPD71087: inverted system bus output
- Single +5 V ±10% power supply
- Industrial temperature range: −40 to +85°C

Ordering Information

Part Number	Package	Output
μPD71086C	20-pin plastic DIP (300 mil)	Noninverted
G	20-pin plastic SOP	
μPD71087C	20-pin plastic DIP (300 mil)	Inverted
G	20-pin plastic SOP	

Pin Configurations

20-Pin Plastic DIP and SOP



Pin Identification

Symbol	Function
LB_7-LB_0	CPU local I/O data bus, bits 7-0
SB_7-SB_0/$\overline{SB}_7$-$\overline{SB}_0$	System I/O data bus, bits 7-0; noninverted (μPD71086) or inverted (μPD71087)
$\overline{OE}$	Output enable input
BUF$\overline{R}$/W	Buffer read/write input
V_{DD}	+5 V power supply
V_{SS}	Ground

PIN FUNCTIONS

LB_7-LB_0 (Local Data Bus)

LB_7-LB_0 are three-state inputs/outputs that connect to the CPU local data bus. They move data between the CPU and memory, I/O, or other peripherals. Data read/write mode is controlled by the BUF$\overline{R}$/W signal input.

SB_7-SB_0/$\overline{SB}_7$-$\overline{SB}_0$ (System Data Bus)

SB_7-SB_0/$\overline{SB}_7$-$\overline{SB}_0$ are three-state inputs/outputs that connect to the system bus, along with the memory, I/O, or other peripherals. The μPD71086 causes no signal inversion, the μPD71087 inverts the signal. Input/output condition is determined by BUF$\overline{R}$/W status. See table 1.

$\overline{OE}$ (Output Enable)

$\overline{OE}$ input controls the output buffers. When $\overline{OE}$ is high, all output buffers go to the high-impedance state. When $\overline{OE}$ is low, data is output from the buffers specified by the BUF$\overline{R}$/W signal.

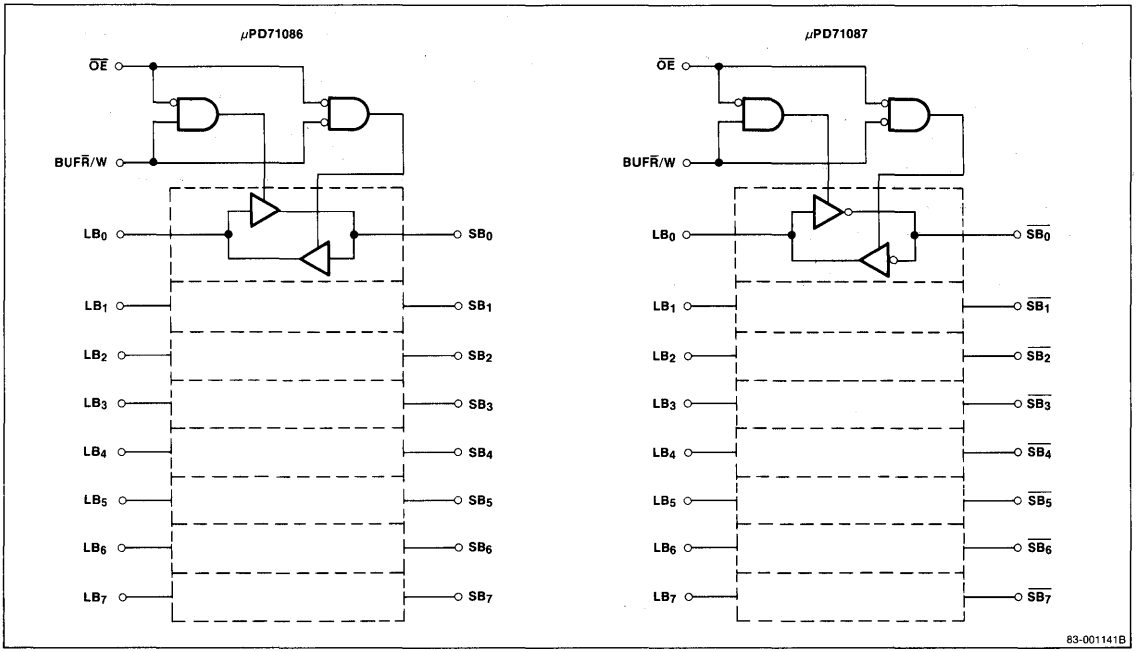
BUF$\overline{R}$/W (Buffer Read/Write)

The data read/write mode is controlled by the BUF$\overline{R}$/W signal input. When BUF$\overline{R}$/W is high, LB lines are inputs and SB lines are outputs. When BUF$\overline{R}$/W is low, SB lines are inputs and LB lines are outputs. See table 1.

Table 1. Data Read/Write Mode

$\overline{OE}$	BUF$\overline{R}$/W	LB Pins	SB/$\overline{SB}$ Pins	Mode
Low	Low	Output	Input	System bus to local bus
Low	High	Input	Output	Local bus to system bus
High	Don't care	High-Z	High-Z	

μPD71086, 71087 Block Diagram



ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A= 25°C; V_{SS} = 0 V

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−1.0 to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 to V_{DD} + 0.5 V
Power dissipation, P_D	
DIP	500 mW
SOP	200 mW
Operating temperature, T_{OPT}	−40 to +85°C
Storage temperature, T_{STG}	−65 to +150°C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Capacitance

T_A = 25°C; V_{DD} = +5 V

Parameter	Symbol	Min	Max	Units	Conditions
Input capacitance	C_{IN}		24	pF	f_c = 1 MHz

DC Characteristics

T_A = −45 to +85°C; V_{DD} = 5 V ±10%

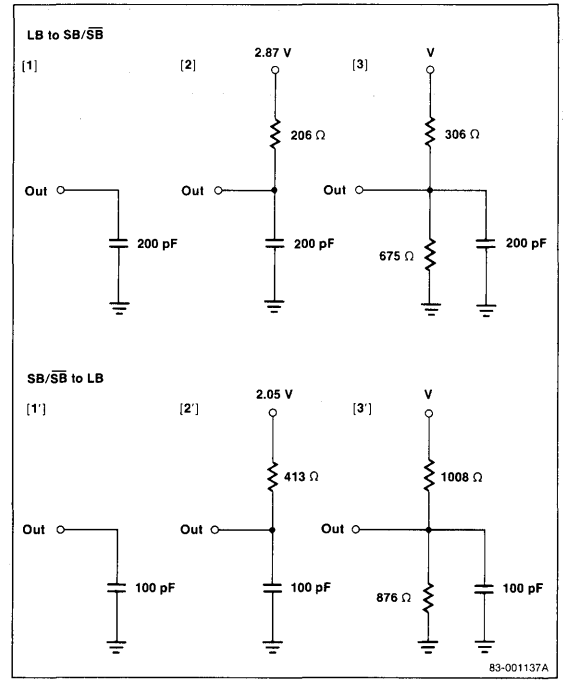
Parameter	Symbol	Min	Max	Units	Conditions
Input voltage high	V_{IH}	2.2		V	
Input voltage low	V_{IL}		0.8	V	
Output voltage high	V_{OH}	V_{DD} −0.8		V	I_{OH} = −4 mA
Output voltage low	V_{OL}		0.45	V	LB, I_{OL} = 4 mA
Output voltage low	V_{OL}		0.45	V	SB, I_{OL} = 12 mA
Input leakage current	I_{IL}	−1.0	1.0	μA	V_I = V_{DD}, V_{SS}
Leakage current, high impedance	I_{OFF}	−10	10	μA	$\overline{OE}$ = V_{DD}
Power supply current (static)	I_{DD}		80	μA	V_I = V_{DD}, V_{SS}
Power supply current (dynamic)	I_{DDdyn}		40	mA	f_{in} = 2 MHz

AC Characteristics

T_A = −40 to 85°C; V_{DD} = 5 V ±10%

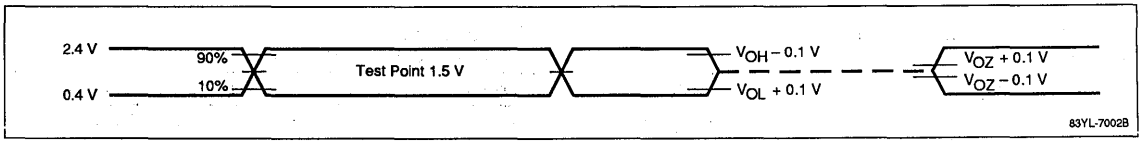
Parameter	Symbol	Min	Max	Units	Conditions
Input to output delay	t_{DIO}	5	40	ns	Load (1), (1') and (2), (2')
BUF$\overline{R}$/W hold time from $\overline{OE}$	t_{HCTRW}	5		ns	
BUF$\overline{R}$/W setup time to $\overline{OE}$	t_{SRWCT}	10		ns	
Data float time from $\overline{OE}$	t_{FCTO}	5	30	ns	Load (3) and (3')
Data output delay from $\overline{OE}$	t_{DCTO}	10	40	ns	
Signal rise time	t_R		20	ns	0.8 to 2.0 V
Signal fall time	t_F		12	ns	2.0 to 0.8 V

Loading Circuit for AC Test

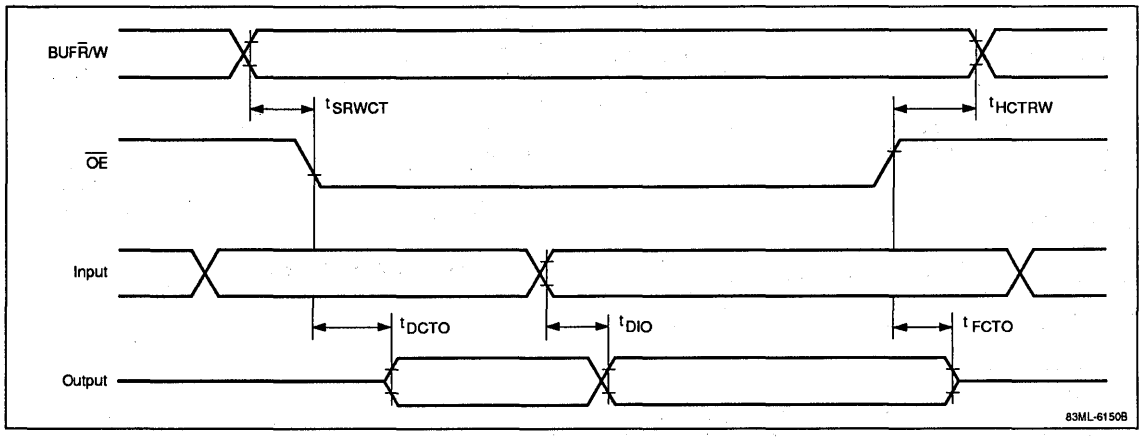


5j

AC Test Voltages



Timing Waveforms



Description

The μPD71088 is a CMOS system bus controller for a μPD70108 (V20®) or μPD70116 (V30®) microprocessor system. It controls the memory or I/O system bus.

Features

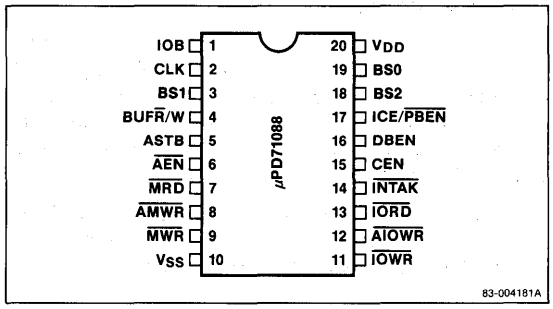
- CMOS technology
- Bus controller for microcomputer system expansion
- Command outputs for system bus control
- Control outputs for I/O peripheral bus control
- High drive capability for command and control outputs (I_{OL} = 12 mA)
- Three-state outputs for command outputs
- Advanced I/O and memory write command outputs
- μPD70108, μPD70116 compatible
- +5-volt ±10% single power supply
- 20-pin plastic DIP (300 mil) or SOP package
- Industrial temperature range: −40 to +85°C

Ordering Information

Part Number	Clock (MHz)	Package Type
μPD71088C-8	8	20-pin plastic DIP
C-10	10	
G-8	8	20-pin plastic SOP

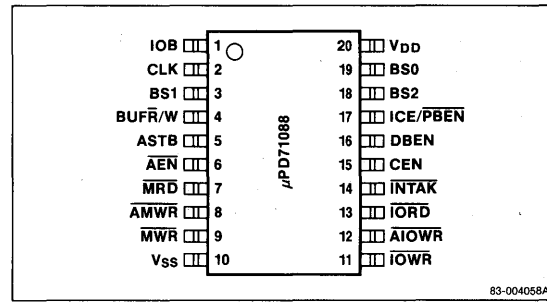
Pin Configurations

20-Pin Plastic DIP



V20 and V30 are registered trademarks of NEC Corporation.

20-Pin Plastic SOP



Pin Identification

Symbol	Function
IOB	Input/output bus mode input
CLK	Clock input
BS1	Bus status input 1
BUFR/W	Buffer read/write output
ASTB	Address strobe output
AEN	Address enable input
MRD	Memory read output
AMWR	Advanced memory write output
MWR	Memory write command output
V_{SS}	Ground
IOWR	I/O write command output
AIOWR	Advanced I/O write command output
IORD	I/O read command output
INTAK	Interrupt acknowledge output
CEN	Command enable input
DBEN	Data buffer enable output
ICE/PBEN	Interrupt cascade enable/Peripheral data bus enable output
BS2	Bus status input 2
BS0	Bus status input 0
V_{DD}	Power supply

PIN FUNCTIONS

BS0-BS2 (Bus Status Inputs 0, 1, 2)

The BS0-BS2 inputs are connected to the encoded CPU status outputs. The μPD71088 decodes these status outputs into command and control outputs for timing control. See table 1 for an explanation of these inputs.

CLK (Clock)

The CLK input is connected to the same clock output that drives the CPU clock, usually the CLK output of a μPD71084 or a μPD71011. It is the internal system clock of the μPD71088.

$\overline{\text{AEN}}$ (Address Enable)

The $\overline{\text{AEN}}$ input controls the command output buffers. When IOB is low, a low-level $\overline{\text{AEN}}$ causes the command buffers to output command output signals. A high-level $\overline{\text{AEN}}$ makes all command lines go to high impedance. When IOB is high, the μPD71088 is in I/O bus mode, and the command lines are not affected by $\overline{\text{AEN}}$.

CEN (Command Enable)

The CEN input controls DBEN, $\overline{\text{PBEN}}$ and all command outputs. When CEN is high, all these outputs are active. When CEN is low, they are inactive.

IOB (I/O Bus Mode)

When the IOB input is high, the bus control mode is I/O bus mode. When IOB is low, the bus control mode is system bus mode.

$\overline{\text{MRD}}$ (Memory Read Command)

The $\overline{\text{MRD}}$ output is the signal to read data from a memory device. $\overline{\text{MRD}}$ is three-state, active low.

$\overline{\text{MWR}}$ (Memory Write Command)

The $\overline{\text{MWR}}$ output is the signal to write data to a memory device. $\overline{\text{MWR}}$ is three-state, active low.

$\overline{\text{AMWR}}$ (Advanced Memory Write Command)

This command output is the same as $\overline{\text{MWR}}$, except that itis generated one state (clock cycle) earlier than $\overline{\text{MWR}}$.

$\overline{\text{IORD}}$ (I/O Read Command)

The $\overline{\text{IORD}}$ output is the signal to read data from an I/O device. $\overline{\text{IORD}}$ is three-state, active low.

$\overline{\text{IOWR}}$ (I/O Write Command)

The $\overline{\text{IOWR}}$ output is the signal to write data to an I/O device. $\overline{\text{IOWR}}$ is three-state, active low.

$\overline{\text{AIOWR}}$ (Advanced I/O Write Command)

This command output is the same as $\overline{\text{IOWR}}$, except that it is generated one state (clock cycle) earlier than $\overline{\text{IOWR}}$.

$\overline{\text{INTAK}}$ (Interrupt Acknowledge)

The $\overline{\text{INTAK}}$ output acknowledges interrupt requests. Requesting devices output an interrupt vector address in response to $\overline{\text{INTAK}}$. $\overline{\text{INTAK}}$ is three-state, active low.

ASTB (Address Strobe)

The ASTB output control signal latches the address outputs from the CPU into an external address latch, such as a μPD71082 or μPD71083. Address data should be strobed with the trailing edge (high to low) of ASTB.

DBEN (Data Buffer Enable)

The DBEN output activates a data bus buffer/driver such as a μPD71086 or μPD71087 to input or output data between the CPU local bus and the memory or I/O system bus.

BUF$\overline{\text{R}}$/W (Buffer Read/Write)

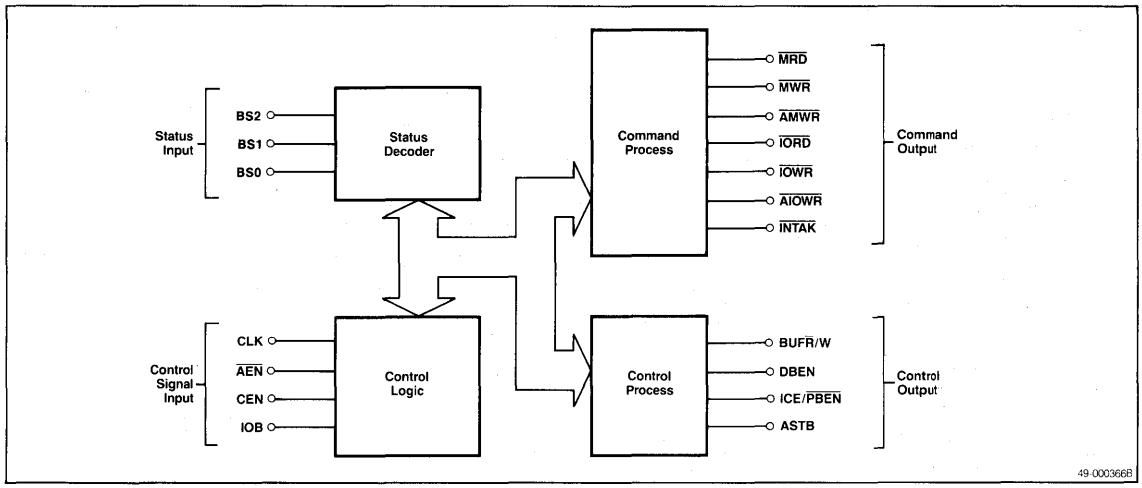
The BUF$\overline{\text{R}}$/W output controls the direction in which data moves through a transceiver between the CPU and the memory or I/O peripherals. When BUF$\overline{\text{R}}$/W is high, data is transferred from the CPU local bus to the memory or I/O system bus. When BUF$\overline{\text{R}}$/W is low, data is transferred from the memory or I/O system bus to the CPU local bus.

ICE/$\overline{\text{PBEN}}$ (Interrupt Cascade Enable/Peripheral Data Bus Enable)

The meaning of this output signal depends on IOB. If IOB is low (system bus mode), it is the ICE output. ICE controls the cascade address transfer from a master priority interrupt controller to slave priority interrupt controllers. The slave reads the address from the master when ICE goes high.

When IOB is high, it becomes $\overline{\text{PBEN}}$. $\overline{\text{PBEN}}$ controls the I/O bus the same way that DBEN controls the system bus. In this case, however, the output is active low.

Block Diagram



Absolute Maximum Ratings

T_A = 25°C; V_{SS} = 0 V

Power supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−1.0 to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 to V_{DD} + 0.5 V
Operating temperature, T_{OPT}	−40 to +85 °C
Storage temperature, T_{STG}	−65 to +150 °C
Power dissipation, P_D (DIP)	500 mW
Power dissipation, P_D (SO)	200 mW

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Capacitance

T_A = 25°C; V_{DD} = +5 V

Parameter	Symbol	Min	Max	Units	Conditions
Input capacitance	C_{IN}		12	pF	f = 1 MHz

5k

DC Characteristics

T_A −40°C to +85 °C; V_{DD} = 5 V ±10%

Parameter	Symbol	Min	Max	Unit	Conditions
Input voltage, high	V_{IH}	2.2		V	
Input voltage, low	V_{IL}		0.8	V	
Output voltage, high	V_{OH}	V_{DD} −0.8		V	Controls: I_{OH} = −8 mA @ 10 MHz, −4 mA @ 8 MHz
Output voltage, low	V_{OL}		0.45	V	Commands: I_{OL} = 24 mA @ 10 MHz, 12 mA @ 8 MHz Controls: I_{OL} = 8 mA @ 10 MHz, 4 mA @ 8 MHz
Input current leakage	I_{IL}	−1.0	1.0	μA	V_I = V_{DD}, V_{SS}
Leakage current at high impedance	I_{OFF}	−10	10	μA	
Power supply current (static)	I_{DD}		80	μA	V_I = V_{DD}, V_{SS}
Power supply current (dynamic)	I_{DDdyn}		20	mA	f_{in} = 10 MHz

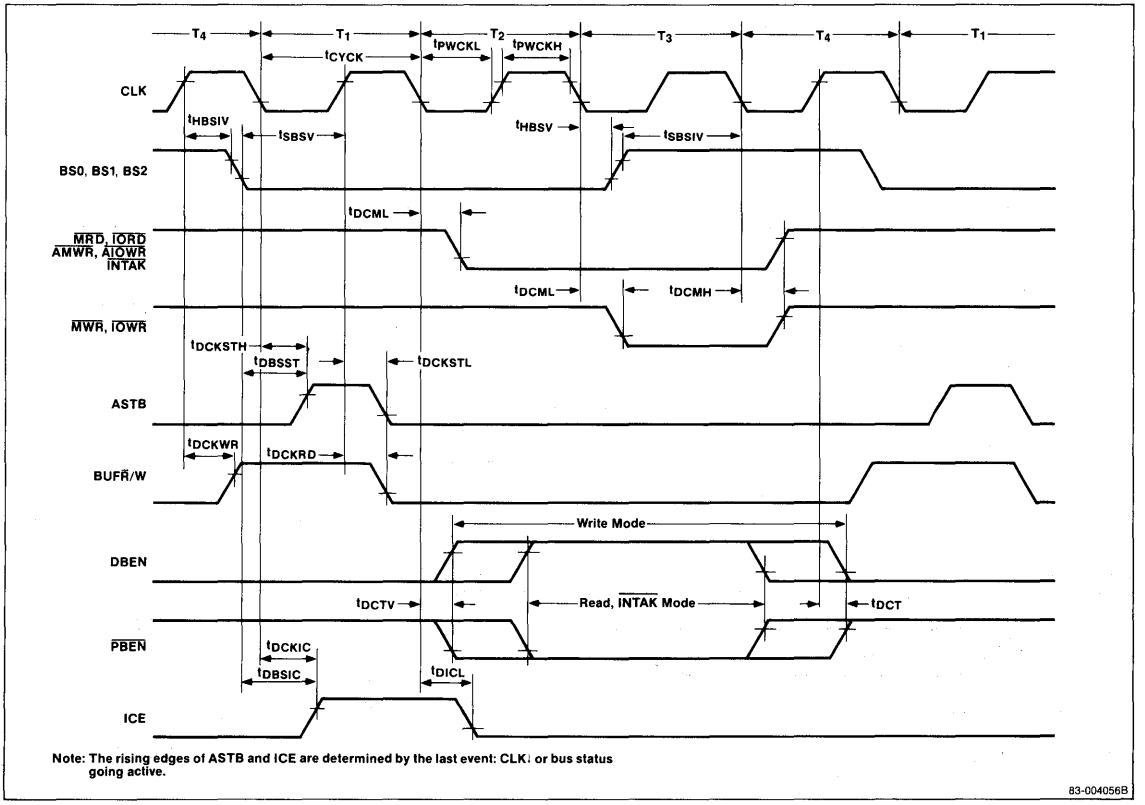
AC Characteristics

T_A = −40 to +85°C; V_{DD} = 5 V ±10%

Parameter	Symbol	μPD71088 Min	μPD71088 Max	μPD71088C-10 Min	μPD71088C-10 Max	Units	Conditions
CLK cycle period	t_{CYCK}	125		100		ns	
CLK pulse width, high	t_{PWCKH}	40		41		ns	
CLK pulse width, low	t_{PWCKL}	60		49		ns	
Setup time for bus status active to CLK ↑	t_{SBSV}	40		35		ns	
Hold time for bus status inactive from CLK ↓	t_{HBSV}	10		10		ns	
Setup time for bus status inactive to CLK ↓	t_{SBSIV}	35		35		ns	
Hold time for bus status inactive from CLK ↑	t_{HBSIV}	10		10		ns	
Command active delay from CLK ↓	t_{DCML}	10	40	10	35	ns	
Command inactive delay from CLK ↓	t_{DCMH}	10	40	10	35	ns	
Command output on delay from $\overline{AEN}$ ↓	t_{DAECM}		40		40	ns	
Command active output delay from $\overline{AEN}$ ↓	t_{DAECML}	100	295	115	200	ns	
Command disable delay from $\overline{AEN}$ ↑	t_{FAECM}		50		20	ns	
Command active delay from CEN ↑	t_{DCECM}		t_{DCML}		t_{DCML}	ns	I_{OL} = 4 mA I_{OH} = −4 mA C_L = 100 pF
ASTB active delay from CLK ↓	t_{DCKSTH}		30		20	ns	
ASTB active delay from BS2, 1, 0	t_{DBSST}		25		20	ns	
ASTB inactive delay from CLK ↑	t_{DCKSTL}	7	25	7	25	ns	
DBEN, $\overline{PBEN}$ active delay from CLK ↓	t_{DCTV}	10	50	10	35	ns	
DBEN, $\overline{PBEN}$ inactive delay from CLK ↑	t_{DCT}	10	50	10	35	ns	
DBEN, $\overline{PBEN}$ active delay from $\overline{AEN}$ ↓	t_{DAECT}		30		30	ns	
DBEN, $\overline{PBEN}$ active delay	t_{DCECT}		30		30	ns	
BUF$\overline{R}$/W ↑ delay from CLK ↑	t_{DCKWR}		40		40	ns	
BUF$\overline{R}$/W ↓ delay from CLK ↑	t_{DCKRD}		60		40	ns	
ICE active delay from CLK ↓	t_{DCKIC}		30		30	ns	
ICE active delay from BS2, 1, 0	t_{DBSIC}		25		20	ns	
ICE inactive delay from CLK ↓	t_{DICL}	10	50	10	40	ns	
Input rise time	t_{RI}		20		20	ns	0.8 V to 2.0 V
Output rise time	t_{RO}		20		20	ns	
Input fall time	t_{FI}		12		12	ns	2.0 V to 0.8 V
Output fall time	t_{FO}		12		12	ns	

Timing Waveforms

General

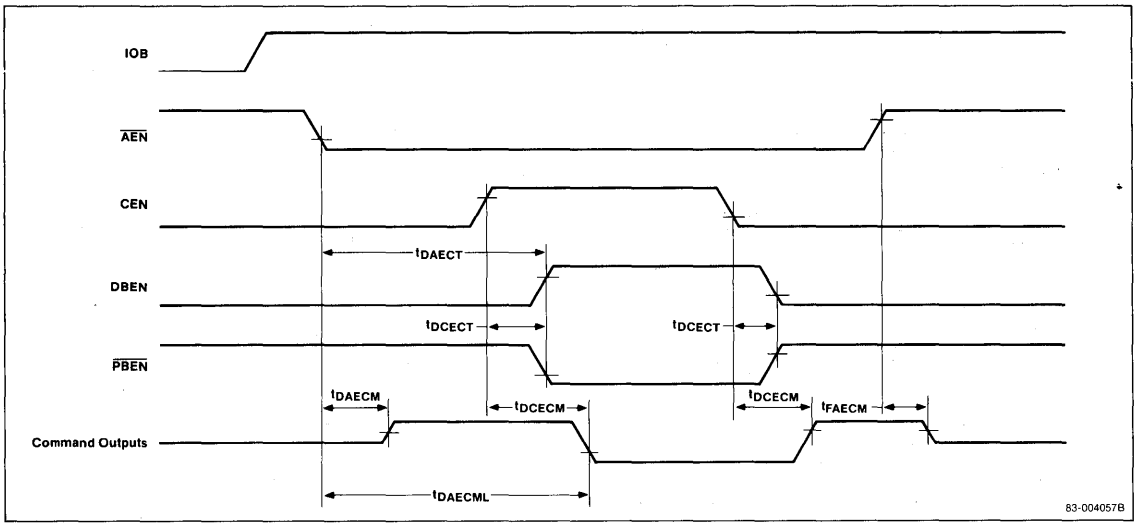


Note: The rising edges of ASTB and ICE are determined by the last event: CLK↓ or bus status going active.

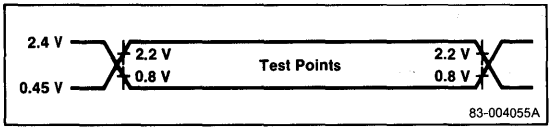
5k

Timing Waveforms (cont)

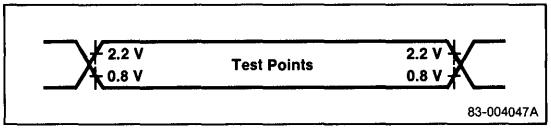
DBEN, $\overline{PBEN}$, and Command Output



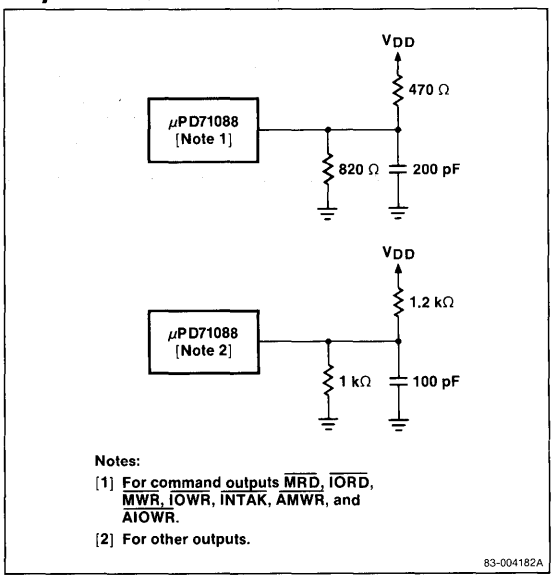
AC Test Input



AC Test Output



Output Test Loads



FUNCTIONAL DESCRIPTION

Command Logic

The PD71088 decodes the CPU bus status outputs into command outputs. The bus status outputs (BS0-BS2) and their decoded commands are shown in table 1.

Bus Control Mode

The CEN, IOB, and $\overline{AEN}$ signals control the bus controller mode as shown in table 2.

Table 1. Command Logic

BS2	BS1	BS0	CPU Status	μPD71088 Command Output
Low	Low	Low	Interrupt acknowledge	$\overline{INTAK}$
Low	Low	High	I/O read mode	$\overline{IORD}$
Low	High	Low	I/O write mode	$\overline{IOWR}$, $\overline{AIOWR}$
Low	High	High	Halt mode	None
High	Low	Low	Instruction fetch mode	$\overline{MRD}$
High	Low	High	Memory read mode	$\overline{MRD}$
High	High	Low	Memory write mode	$\overline{MWR}$, $\overline{AMWR}$
High	High	High	No bus cycle mode	None

Table 2. Bus Control Mode

Control Input			Command Output		Control Output	
			Memory	I/O		
CEN	IOB	$\overline{AEN}$	$\overline{MRD}$, $\overline{MWR}$, $\overline{AMWR}$	$\overline{IOWR}$, $\overline{AIOWR}$, $\overline{IORD}$, $\overline{INTAK}$	ICE/$\overline{PBEN}$	ASTB, BUF$\overline{R}$/W, DBEN
H	H (I/O bus mode)	H	High impedance	Outputs enabled (NC)	$\overline{PBEN}$ (NC)	Outputs enabled (NC)
		L	Outputs enabled			
H	L (System bus mode)	H	High impedance	High impedance	ICE (NC)	Outputs enabled (NC)
		L	Outpus enabled	Outputs enabled		
L (Command disable mode)	x	x	H	H	$\overline{PBEN}$ = H	Outputs enabled (DBEN = L:ASTB, BUF$\overline{R}$/W are NC)

Note:
x = Don't care, NC = No change, H = High, L = Low

5k

μPD71641
Cache Memory Controller

Description

The μPD71641 is an LSI cache controller chip offering advanced features, unequaled flexibility, and built-in reliability to system designers. The μPD71641 makes it practical and economical to use sophisticated caches in microprocessor-based systems.

The implementation of μPD71641 is transparent to the application program. The μPD71641 is configurable from direct-mapped to 4-way set-associative mapping. The μPD71641 allows up to 128K bytes of cache memory. Cache updating is made efficient with sub-block partition and burst mode features.

The μPD71641 can be easily used with many general-purpose, high-performance 32-bit or 16-bit microprocessors. Its architecture is suitable for multiprocessors and multimaster environments. Cache data consistency is ensured by bus monitoring and dual comparator techniques. The μPD71641 uses a write-through strategy to update main memory, which guarantees the best cache consistency in a multiprocessor and multimaster system. External data storage is flexible in size and organization. The μPD71641 will work with any word width.

The μPD71641 is unique in offering features to implement a highly reliable cache memory subsystem. The μPD71641 provides built-in reliability checks, such as address tag parity check, multiple hit detection, and self-diagnosis for directory faults. Upon detection of an erroneous condition, the μPD71641 can either be disabled, or continue to operate in a functionally degraded mode.

Features

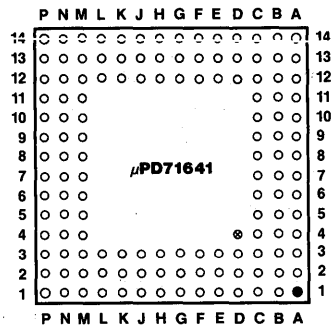
- General-purpose interface supports high-performance microprocessors
- Transparent to application programs
- Flexible placement algorithm: direct 2-, 4-way set-associative
- Large tag memory configuration:
  - 1024 sets x 1 way x 2 sub-blocks
  - 512 sets x 2 ways x 2 sub-blocks
  - 256 sets x 4 ways x 2 sub-blocks
- Programmable sub-block size up to 64 bytes
- Bus replacement cycle variable from 1 to 16 words
- Supports large cache memory up to 128K bytes
- Supports up to 4G bytes of main memory
- LRU replacement algorithm
- Write-through strategy
- Data consistency check by bus monitoring
- External PURGE input to flush tag store
- Increased reliability through internal error detection
  - Parity check on tag store
  - Incorrect match check
  - Multiple hit check
  - LRU output check
- Unique level degradation feature to maximize cache system up time
- 16- and 20-MHz operation
- 132-pin PGA package

Ordering Information

Part Number	Max Clockout Frequency	Package
μPD71641R	20 MHz	132-pin Ceramic PGA

Pin Configuration

132-Pin Ceramic PGA



Pin Identification

Pin	Symbol
A1	$\overline{RREQ}$
A2	D_1
A3	$\overline{RACK}$
A4	$\overline{MAS}$
A5	RC2
A6	RA0
A7	RA2
A8	RA3
A9	IC
A10	$\overline{BERR}$
A11	RT/$\overline{BY}$
A12	$\overline{SRDY}$
A13	$\overline{CR20}$
A14	A_3
B1	MA_5
B2	GND
B3	D_0
B4	D_3
B5	RC1
B6	RC3
B7	RA1
B8	IC

Pin Identification

Pin	Symbol
B9	IC
B10	$\overline{RBCY}$
B11	$\overline{CW20}$
B12	$\overline{CR21}$
B13	V_{DD}
B14	A_5
C1	MA_8
C2	MA_4
C3	V_{DD}
C4	GND
C5	D_2
C6	RC0
C7	V_{DD}
C8	GND
C9	BRC
C10	$\overline{RAE}$
C11	$\overline{CW21}$
C12	GND
C13	A_4
C14	A_8

Pin	Symbol
D1	MA_9
D2	MA_6
D3	MA_3
D12	GND
D13	A_6
D14	A_9
E1	MA_{12}
E2	MA_{10}
E3	MA_7
E12	A_7
E13	A_{10}
E14	A_{12}
F1	MA_{14}
F2	MA_{13}
F3	MA_{11}
F12	A_{11}
F13	A_{13}
F14	GND
G1	MA_{16}
G2	MA_{15}
G3	GND
G12	A_{14}
G13	A_{15}
G14	A_{16}
H1	MA_{17}
H2	MA_{18}
H3	MA_{19}
H12	A_{18}
H13	V_{DD}
H14	A_{17}
J1	MA_{20}
J2	MA_{21}
J3	MA_{23}
J12	A_{22}
J13	A_{20}
J14	A_{19}
K1	MA_{22}
K2	MA_{24}
K3	MA_{27}
K12	A_{26}
K13	A_{23}

Pin	Symbol
K14	A_{21}
L1	MA_{25}
L2	MA_{28}
L3	MA_{31}
L12	A_{30}
L13	A_{27}
L14	A_{24}
M1	MA_{26}
M2	MA_{30}
M3	GND
M4	$\overline{BCY}$
M5	DPAR
M6	GND
M7	V_{DD}
M8	GND
M9	$\overline{MISS}$
M10	IC
M11	V_{DD}
M12	GND
M13	A_{29}
M14	A_{25}
N1	MA_{29}
N2	V_{DD}
N3	$\overline{CS}$
N4	$\overline{BYPI}$
N5	PURGE
N6	FAULT
N7	CLK
N8	$\overline{CR3}$
N9	$\overline{CW1}$
N10	$\overline{CW3}$
N11	IC
N12	AHIT
N13	GND
N14	A_{28}
P1	R/$\overline{W}$
P2	$\overline{CMD}$
P3	RESET
P4	FATAL
P5	CLAMP
P6	$\overline{CR0}$

Pin Identification (cont)

Pin	Symbol	Pin	Symbol
P7	$\overline{CR1}$	P11	$\overline{BYPO}$
P8	$\overline{CR2}$	P12	IC
P9	$\overline{CW0}$	P13	$\overline{CRDY}$
P10	$\overline{CW2}$	P14	A_{31}

PIN FUNCTIONS

CPU Interface

A_3-A_{31} (Address Bus). CPU address outputs are connected to these inputs. Depending on cache organization, from 8 to 10 of the low address bits are used to select a set of tags. These inputs are latched internally once a μPD71641 cycle begins.

$\overline{BCY}$ (Bus Cycle Start). This active-low input is sampled on the rising edge of CLK. It indicates that a CPU cycle is ready to be submitted to the μPD71641. When $\overline{BCY}$ is detected, the μPD71641 will begin its cycle.

$\overline{CS}$ (Chip Select). This active-low input is sampled on the rising edge of CLK. If this pin is not asserted, then the cache will not operate. This input is used to separate cacheable (e.g., memory access) and non-cacheable (e.g., I/O) CPU bus cycles. If this input is not asserted, $\overline{BCY}$ is ignored, but $\overline{CMD}$ operations can still take place and the bus monitor function continues to operate.

R/$\overline{W}$ (Read/Write). This input is used to separate CPU read cycles from CPU write cycles, since the μPD71641 handles each type of cycle differently. If this input is low when $\overline{BCY}$ is sampled, a cache write operation will begin. If R/$\overline{W}$ is high, read operation will begin. This input is also used during $\overline{CMD}$ accesses.

$\overline{CMD}$ (Command Mode). This active-low input is sampled on the rising edge of CLK. It should be asserted when the CPU needs to access one of the μPD71641's internal registers. Inputs RA0-RA3 select which internal register will be used. $\overline{CMD}$ overrides the $\overline{BCY}$ and $\overline{CS}$ inputs.

CLAMP (Clamp Address Input). This active-high input prevents any problems due to a floating level on any μPD71641 inputs. In the event it becomes necessary to float the inputs to the μPD71641 (e.g., during DMA), this input should be asserted. If this input is used, external pullup resistors will not be required.

$\overline{BYPI}$ (Bypass In). This active-low input is sampled on the rising edge of CLK. If this input is asserted when $\overline{BCY}$ is asserted, the μPD71641 will treat the cycle as non-cacheable, and assert $\overline{BYPO}$. The cache memory is not accessed.

DPAR (Data Parity). This active-high input signals that an error has occurred in the cache data store. When these errors are received, the μPD71641 will disable the current level and enter a functionally degraded mode. That is, if a parity error occurs in bank 3 of a 4-way set-associative cache memory, level 3 will be disabled.

AHIT (Asynchronous Cache Hit). This active-high asynchonous output is asserted when the current address inputs (A_3-A_{31}) produce a tag match, indicating that the data being accessed is mapped into the cache data store.

$\overline{CRDY}$ (CPU Ready). This active-low output can be used to signal the CPU that the bus cycle can be ended. $\overline{CRDY}$ is asserted when the cycle is a cache hit, when the requested data is available at the end of a cache update cycle, when a $\overline{CMD}$ access is completed, or when a bypass cycle has completed (actually controlled by external logic via the $\overline{SRDY}$ input).

$\overline{MISS}$ (Cache Miss). This active-low output is asserted when the μPD71641 has detected a cache miss on a CPU read cycle. It remains asserted throughout the cache update operation. $\overline{MISS}$ is distinct from AHIT because it is synchronous to the CLK, while AHIT is not. $\overline{MISS}$ is also asserted when an internal error forces a cache update, or if a cache bypass cycle is aborted by $\overline{BERR}$.

$\overline{BYPO}$ (Bypass Out). This active-low output indicates to external hardware that the current CPU bus cycle will bypass the cache memory. Either the address is not cacheable, or the cycle is a memory write. In the case of a memory write, the data will be written in parallel to the cache memory. External logic must complete the bus cycle. This output is asserted whenever the $\overline{BYPI}$ input is asserted at the start of a cycle, during all cache write-through cycles, and whenever the μPD71641 is unable to complete a cycle due to an internal error condition.

FAULT (Fault). This active-high output indicates that the μPD71641 is operating in a functionally degraded mode, that is, some portion of the tag store has been disabled due to an internal or external error.

FATAL (Fatal Fault). This active-high output in conjuction with FAULT indicates the status of the μPD71641. If an unrecoverable internal error occurs, and the μPD71641 removes itself from the system, this output will remain asserted.

RA0-RA3 (Replacement Address). These inputs select the internal register for a $\overline{CMD}$ access. They also supply the starting offset for a fetch bypass operation. During

normal replacement operations, the replace cycle begins at an offset of 0. If fetch bypass is enabled, the first bus cycle of the replacement will be to the address determined by these inputs, so that the data requested by the CPU will be available as soon as possible.

D_0-D_3 (Data Bus). These bidirectional pins form the 4-bit data bus used to access the μPD71641 internal registers. This bus is also used to output tag store contents during a cache directory dump operation.

$\overline{CR0}$-$\overline{CR3}$ (4-Way Cache Read Strobes). When a 4-way set-associative cache organization is selected, these four active-low outputs select which of the four banks of cache data storage will supply the data on a CPU read cycle. They are also asserted during a cache memory dump operation.

$\overline{CR20}$-$\overline{CR21}$ (2-Way Cache Read Strobes). When a 2-way set-associative cache organization is selected, these two active-low outputs select which of the two banks of cache data storage will supply the data on a CPU read cycle.

They are also asserted during a cache memory dump operation. These outputs are also used for direct-mapped caches. $\overline{CR20}$ and $\overline{CR21}$ must be logically ORed with external logic to produce a single read strobe for the one bank of RAM.

$\overline{CW0}$-$\overline{CW3}$ (4-Way Cache Write Strobes). When a 4-way set-associative cache organization is selected, these four active-low outputs select which bank of the cache data store will be written during a CPU write cycle.

$\overline{CW20}$-$\overline{CW21}$ (2-Way Cache Write Strobes). When a 2-way set-associative cache organization is selected, these two active-low outputs select which bank of the cache data store will be written during a CPU write cycle. These outputs are also used for direct-mapped caches. $\overline{CW20}$ and $\overline{CW21}$ must be logically ORed with external logic to produce a single write strobe for the one bank of RAM.

Bus Monitor Interface

$\overline{MAS}$ (Bus Monitor Address Strobe). This active-low input is sampled on the rising edge of CLK. When it is sampled low, the μPD71641 will begin a cache invalidate cycle on the address presented on inputs MA_3-MA_{31}. Each check-and-invalidate operation takes two clocks.

MA_3-MA_{31} (Bus Monitor Address). These inputs are sampled when $\overline{MAS}$ is sampled low. The system bus address lines should be logically connected to these inputs.

A write cycle to global memory could change a memory location that has been cached. If this happens, the data in the local cache is no longer consistent with global memory. To ensure cache data consistency, the μPD71641 provides bus monitoring. When an address is presented on these MA inputs, the μPD71641 will perform a tag search on that address. If a high is detected, the indicated sub-block will be invalidated. This tag search operation is completely independent from the CPU address tag search, since the μPD71641 tag store is fully dual-ported with two sets of comparators.

MA_3-MA_{31} become outputs during cache dump operations, supplying the upper address bits for the dump.

System Bus Interface

$\overline{RREQ}$ (Replacement Request). This active-low output is asserted when the μPD71641 wants to use the global bus for a cache data replacement cycle or for a cache directory or data dump operation.

$\overline{RACK}$ (Replacement Acknowledge). This active-low input is asserted by the system bus interface when access to the shared memory for a replacement or a dump cycle has been granted. Once begun, replacement or dump cycles can be suspended by desserting this input. The μPD71641 will interrupt the miss cycle, thereby releasing the global bus to a higher priority bus master, and will resume the interrupted operation when $\overline{RACK}$ is asserted again. Bus monitoring is disabled while the replacement or dump cycle is suspended.

$\overline{RAE}$ (Replacement Address Enable). This active-low output indicates that the μPD71641 has begun a cache data replacement cycle to service a cache read miss. This output should be used to control the multiplexing between the RC outputs and CPU address. Also, any other logic that changes during a replacement cycle should use the $\overline{RAE}$ signal.

RC0-RC3 (Replacement Count). Sub-blocks can be up to 16 words in length. During a replacement cycle, the RC outputs provide the offset address in the sub-block. Depending on the size of the sub-block, not all these pins will be used.

$\overline{RBCY}$ (Replacement Bus Cycle Start). This active-low output signals the start of a replacement bus cycle or a dump cycle.

$\overline{SRDY}$ (System Ready). This active-low input is asserted by the system bus interface or shared memory control logic when the bus operation that the μPD71641 requested has been completed. The μPD71641 will pass this signal through to the CPU via its $\overline{CRDY}$ output.

BRC (Burst Replacement Cycle). This active-high input determines on a cycle-by-cycle basis if the replacement cycle is to use a burst data transfer.

$\overline{\text{BERR}}$ (Bus Error). This active-low input signals a system bus error. If the current cycle is a replacement operation, RT/$\overline{\text{BY}}$ can be used to either abort the cycle or try it again. If the current cycle is a cache bypass (such as a CPU data write cycle), the μPD71641 will assert its $\overline{\text{MISS}}$ output to let the CPU know that the operation did not complete.

RT/$\overline{\text{BY}}$ (Retry/Bypass). When $\overline{\text{BERR}}$ has been detected, the μPD71641 uses the state of this input pin to decide whether to abort the cycle (RT/$\overline{\text{BY}}$ = 0) or to retry the cycle (RT/$\overline{\text{BY}}$ = 1).

This input is also used to implement external write buffering, or a "posted write" system write performance. Normally, during a CPU data write cycle, both the CPU and the μPD71641 are suspended waiting for the write cycle to end. However, if the RT/$\overline{\text{BY}}$ input is high, the μPD71641 will not wait for $\overline{\text{SRDY}}$. It will immediately end its internal cycle and go into the Ti state, allowing the μPD71641 to begin processing the next CPU bus cycle if one is available. External logic can finish the write cycle.

Other Signals

RESET (Reset). This active-high input will reset the μPD71641. All tag stores will be invalidated, all preset commands will be cleared, and all status bits will be cleared. The cache will be disabled until enabled by the software command. RESET must be asserted for at least 10 clock pulses.

PURGE (Purge). This active-high input will purge all the tag stores when it is asserted. An identical software-generated PURGE operation is also available.

PURGE may be used to invalidate the cache tag store in the event the bus monitoring function is unable to keep up with external bus activity.

The PURGE input must be asserted until the purge operation is complete. This takes a maximum of 5 clocks, so PURGE must be asserted for at least 5 clocks.

CLK (Clock). This is the μPD71641 clock and should be sychronized to the CPU clock.

IC. Internally connected; leave disconnected.

V_{DD}. Supply pins for +5 V power supply.

GND. Supply pins for power ground.

Block Diagram

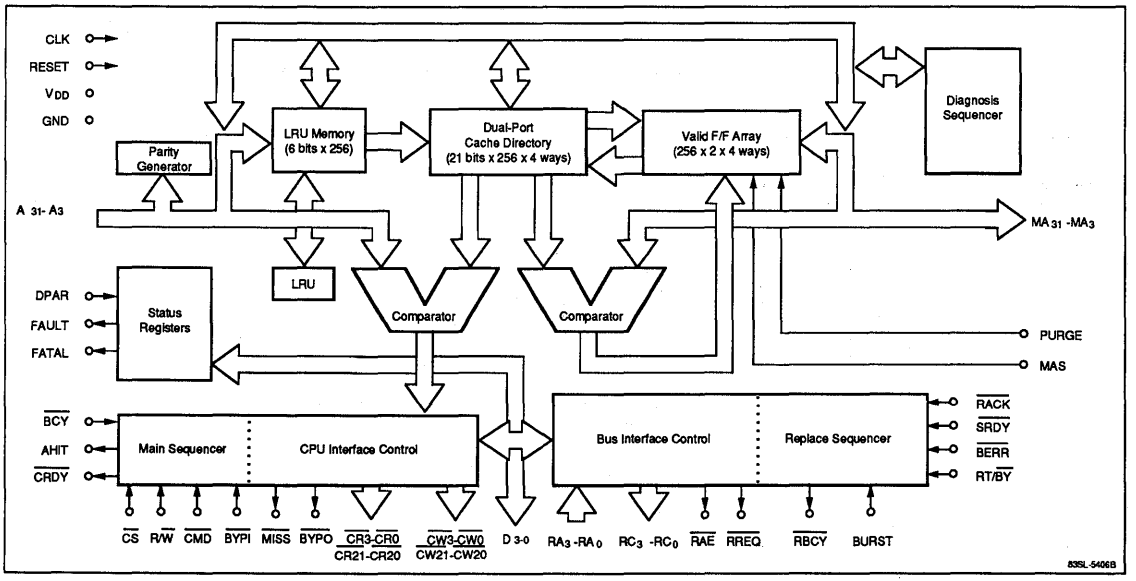


Figure 1. Memory Size Expansion

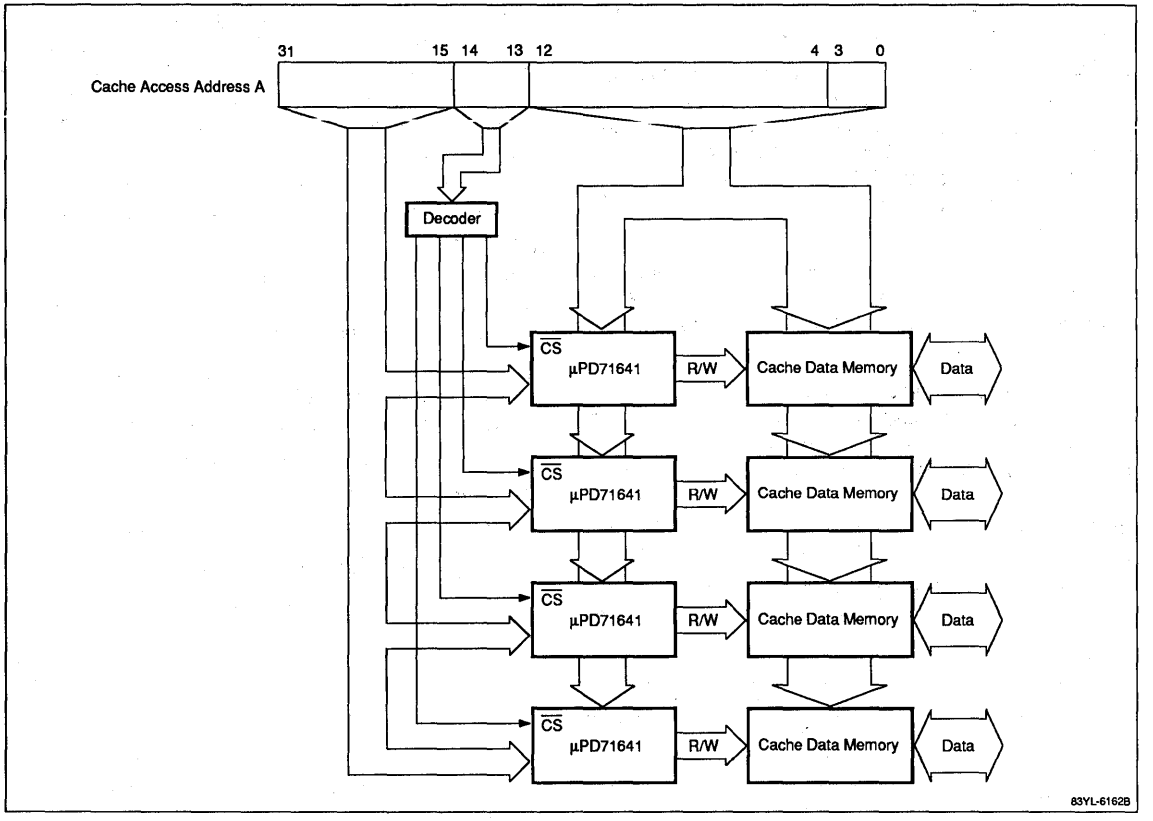
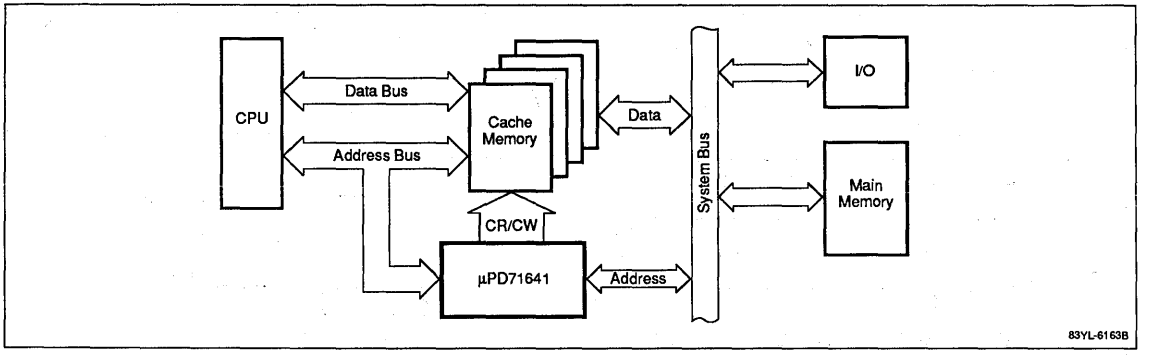


Figure 2. Typical System Configuration



INTERNAL BLOCK FUNCTIONS

Dual-Port Address Tag Memory

Dual-port address tag memory is dual-port memory that decodes the index field input independently from the CPU side and monitor side and outputs the contents of the memory (block entry) as address tag. When a mis-hit occurs during CPU access, a new block address is registered.

Valid Bit Memory

Valid bit memory is a group of flags that indicate whether each block entry is valid or invalid. These flags are set each time the replacement of a sub-block is completed, and reset upon a monitor hit.

Comparators

Two comparators are provided. One is for the CPU side and the other is for the monitor side. The comparator on the CPU side is used to check if the address tag registered in the directory coincides with the address used to access the system bus.

Parity Generator

The parity generator generates the parity for the address tag associated with CPU accessing, and compares it with the address parity registered in the directory. This compare operation is performed simultaneously with the CPU address comparison. When a mis-hit occurs, the generated parity is registered together with the new address.

Status Register

The status register indicates reduction data related to parity errors or multi-hit errors. In addition, it also indicates whether or not the μPD71641 is in the fatal state.

LRU and LRU Memory

LRU and LRU memory manage the CPU block accessing order and perform LRU calculations based on valid bit, error status, number of associative units, etc.

Main Sequencer

The main sequencer decodes the bus cycle signals ($\overline{BCY}$, $\overline{CS}$, R/$\overline{W}$, $\overline{BYPI}$, $\overline{CMD}$, etc.) generated by the CPU, and generates the timing for the basic cache memory operation cycle. The main sequencer determines whether the accessing is a cache memory access, a cache memory bypass access, or a cache command access. The main sequencer enters the standby state when the replace sequencer is in operation.

CPU Interface Controller

The CPU interface controller generates a normal/not-normal response (data acknowledge, bus error, fatal error, etc.).

Replace Sequencer

The replace sequencer regenerates the replace timing and replace count value when a mis-hit occurs.

Bus Interface Control Block

The bus interface control block generates the read/write strobe signal to the external cache data memory.

Table 1 describes each of the bus states. Figure 3 is a state diagram for the bus interface.

Table 1. Bus Interface States

Bus State	Description
TI	Idle state. Waiting for $\overline{BCY}$ and $\overline{CS}$ to be asserted.
T2	First state of any cacheable CPU cycle or IO cycle to μPD71641 registers. T2 is the dispatch state that decides what the μPD71641 will do. If there is a cache hit or a 2-clock cycle request, T2 is also the last active state. If there is a miss, Tm is entered. If this is a cache write cycle, Tb is entered. Tc is entered if it is a μPD71641 command cycle.
Tc	Command Mode state. If this is a μPD71641 command cycle, the Tc is entered following T2, then repeated before returning to the idle state.
Tm	Miss state. Entered if there is cache read miss. The system bus is requested, and the RTI state is entered. If RACK is already asserted, the RT state sequence is entered.
RTI	Replacement Transfer Idle state. Waits for the system bus to be granted via RACK, then transitions to RT1.
RT1	First Replacement Transfer state. While RACK is on, proceeds directly to RT2.
RT2	Second replacement transfer state. Waits for $\overline{SRDY}$ to be asserted, indicating the system bus memory is ready to begin the transfer, the goes to RT3.
RT3	Third RT state. The read data is written to the cache data store, and if the sub-block transfer is complete or the transfer is aborted, the RTw state is entered. If not complete, and burst transfers are not requested, a return is made to RT1 to begin another single-word transfer. If in burst mode, RT4 is enered to get the rest of the sub-block. If a retry is requested, RTr is entered.
RT4	Fourth RT state. Used only for burst transfers, which take two clocks each. Until replacement is complete, the state machine ping-pongs between RT4 and RT5. The data is written at RT5.
RT5	Last RT state. Accepts data from burst transfers. If the transfer is over, RTw is entered. On a retry request, RTr is entered.

51

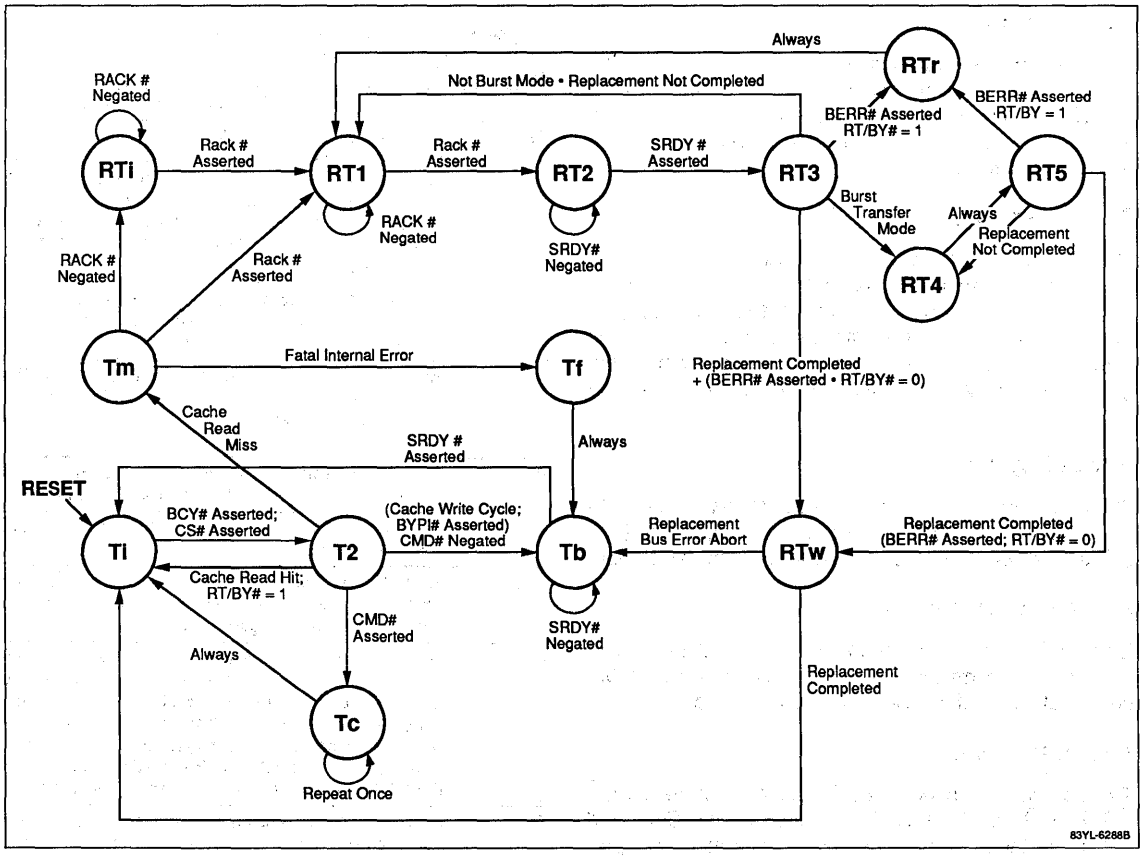
Table 1. Bus Interface States (cont)

Bus State	Description
RTr	RT retry state. Transitions directly to RT1, which begins the cycle again.
RTw	RT recovery state. During this state, if the replacement was successful, the μPD71641 tag store is updated, and the data that the CPU requested is made available. If the cycle was aborted, the bypass state is entered.
Tb	Bypass state. The bypass cycle was started by T2 so Tb waits for the system memory to respond with SRDY, then returns to TI.
Tf	Fatal error state. If during Tm an unrecoverable error occurs, Tf is entered to record the error, then a bypass cycle is entered by going to Tb.

Diagnosis Sequencer

The diagnosis sequencer consists of the directory trace sequencer and the internal diagnosis sequencer. The directory trace sequencer performs cache self-diagnostics and dump processing. The internal diagnosis sequencer sequentially generates the memory diagnostics data for the issuance of internal diagnostics commands, and the collects the data.

Figure 3. Bus State Diagram



ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

T_A = 25°C

Supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage	
Other than CLK, V_{I1}	−0.5 to V_{DD} +0.3 V
CLK V_{I2}	−0.5 to V_{DD} +1.0 V
Output voltage, V_O	−0.5 to V_{DD} +0.3 V
Operating temperature range, T_{OPT}	−10 to +70°C
Storage temperature range, T_{STG}	−60 to +150°C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Capacitance

T_A = +25°C; V_{DD} = GND = 0 V

Parameter	Symbol	Min	Max	Unit	Condition
Input capacitance	C_I		15	pF	f = 1 MHz; unmeasured pins returned to 0 V
Output capacitance	C_O		15	pF	
Input/output capacitance	C_{IO}		15	pF	

DC Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%

Parameter	Symbol	Min	Max	Unit	Condition
Input voltage, high	V_{IH1}	2.2	V_{DD} + 0.3	V	Other than CLK
	V_{IH2}	4.0	V_{DD} + 0.3	V	CLK
Input voltage, low	V_{IL1}	−0.5	0.8	V	Other than CLK
	V_{IL2}	−0.5	0.6	V	CLK
Output voltage, high	V_{OH}	2.4		V	I_{OH} = −400 μA
Output voltage, low	V_{OL}		0.45	V	I_{OL} = 3.2 mA
Input leakage current	I_{LI}		±10	μA	
Output leakage current	I_{LO}		±10	μA	
Operating current	I_{DD}		250	mA	f_{IN} = 20 MHz

AC Characteristics

T_A = −10 to +70°C; V_{DD} = +5 V ±5%

Parameter	Symbol	Min	Max	Unit	Condition
Clock					
Clock frequency	t_{CYK}	50	125	ns	
CLK high-level width	t_{KKH}	20		ns	V_{IH2} = 3.0 V

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Condition
CLK low-level width	t_{KKL}	20		ns	V_{IL2} = 1.7 V
CLK rise time	t_{RK}		3	ns	1.7 to 3.0 V
CLK fall time	t_{FK}		3	ns	3.0 to 1.7 V
Reset, Purge					
Number of reset clock	t_{CYKRS}	10		t_{CYK}	C_L = 100 pF
RESET setup time (vs. CLK ↓)	t_{SRSK}	7		ns	
RESET retention time (vs. CLK ↓)	t_{HKRS}	8		ns	
Number of purge clock	t_{CYKPG}	3		t_{CYK}	
PURGE setup time (vs. CLK ↓)	t_{SPGK}	7		ns	
PURGE retention time (vs. CLK ↓)	t_{HKPG}	8		ns	
CPU Input					
Address setup time (vs. CLK ↑)	t_{SAK}	5		ns	C_L = 100 pF
Address retention time (vs. CLK ↑)	t_{HKA}	15		ns	
$\overline{BCY}$ setup time (vs. CLK ↑)	t_{SBCK}	7		ns	
$\overline{BCY}$ retention time (vs. CLK ↑)	t_{HKBC}	8		ns	
$\overline{CS}$ setup time (vs. CLK ↑)	t_{SCSK}	7		ns	
$\overline{CS}$ retention time (vs. CLK ↑)	t_{HKCS}	8		ns	
$\overline{CMD}$ setup time (vs. CLK ↑)	t_{SCMK}	7		ns	
$\overline{CMD}$ retention time (vs. CLK ↑)	t_{HKCM}	8		ns	
$\overline{BYPI}$ setup time (vs. CLK ↑)	t_{SBIK}	7		ns	
$\overline{BYPI}$ retention time (vs. CLK ↑)	t_{HKBI}	8		ns	
R/$\overline{W}$ setup time (vs. CLK ↑)	t_{SRWK}	7		ns	
R/$\overline{W}$ retention time (vs. CLK ↑)	t_{HKRW}	8		ns	
CLAMP setup time (vs. CLK ↑)	t_{SCPK}	25		ns	
CLAMP retention time (vs. CLK ↑)	t_{HKCP}	0		ns	
RT/$\overline{BY}$ setup time (vs. CLK ↓)	t_{SRTK}	7		ns	

5I

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Condition
RT/$\overline{BY}$ retention time (vs. CLK ⇓)	t_{HKRT}	8		ns	C_L = 100 pF
DPAR setup time (vs. CLK ⇓)	t_{SDPK}	7		ns	
DPAR retention time (vs. CLK ⇓)	t_{HKDP}	8		ns	

CPU-Side Output Timing Specifications

Parameter	Symbol	Max	Unit	Condition
Add valid → AHIT delay	t_{DAAH}	50	ns	C_L = 25 pF
CLK ↓ → $\overline{CRDY}$ valid delay	t_{DKCRY}	25	ns	C_L = 100 pF
CLK ↓ → $\overline{MISS}$ valid delay	t_{DKMS}	25	ns	
CLK ↑ → $\overline{BYPO}$ valid delay	t_{DKBO}	25	ns	
CLK ↑ → $\overline{CR0}$ - $\overline{CR3}$ ↓ delay	t_{DKCRL}	25	ns	
CLK ↓ → $\overline{CR0}$ - $\overline{CR3}$ ↑ delay	t_{DKCRH}	25	ns	
CLK ↓ → $\overline{CW0}$ - $\overline{CW3}$ valid delay	t_{DKCW}	25	ns	
CLK ↑ → $\overline{CR20}$ - $\overline{CR21}$ ↓ delay	t_{DKCR2L}	25	ns	
CLK ↓ → $\overline{CR20}$ - $\overline{CR21}$ ↑ delay	t_{DKCR2H}	25	ns	
CLK ↓ → $\overline{CW20}$- $\overline{CW21}$ delay	t_{DKCW2}	25	ns	
CLK ↑ → FATAL valid delay	t_{DKFT}	25	ns	
CLK ↑ → FAULT valid delay	t_{DKFL}	25	ns	

Command Access/Bus Monitor Timing Specifications

Parameter	Symbol	Min	Max	Unit	Conditions
RA0-RA3 set-up time (CLK ⇑)	t_{SRAK}	10		ns	C_L = 100 pF
RA0-RA3 hold time (CLK ⇑)	t_{HKRA}	10		ns	
D_0-D_3 setup time (CLK ⇑)	t_{SDK}	15		ns	
D_0-D_3 hold time (CLK ⇑)	t_{HKD}	8		ns	
D_0-D_3 valid delay	t_{DKD}		25	ns	
D_0-D_3 float delay	t_{FKD}			ns	
MA hold time (CLK ⇓)	t_{HKMA}	8		ns	
MA setup time (CLK ⇓)	t_{SMAK}	15		ns	
$\overline{MAS}$ hold time (CLK ⇓)	t_{HKMS}	10		ns	
$\overline{MAS}$ setup time (CLK ⇓)	t_{SMSK}	10		ns	

Replace/Bypass Timing Specifications

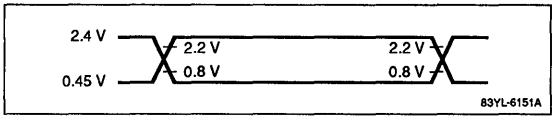
Parameter	Symbol	Min	Max	Unit	Conditions
BRC setup time (CLK ⇓)	t_{SBRK}	7		ns	C_L = 100 pF
BRC hold time (CLK ⇓)	t_{HKBR}	8		ns	
CLK ↑ → $\overline{RREQ}$ valid delay	t_{DKRRQ}		25	ns	
$\overline{RACK}$ setup time (CLK ⇓)	t_{SRKK}	7		ns	
$\overline{RACK}$ hold time (CLK ⇓)	t_{HKRK}	8		ns	
CLK ↓ → $\overline{RAE}$ valid delay	t_{DKRAE}		25	ns	
CLK ↑ → $\overline{RBCY}$ valid delay	t_{DKRB}		25	ns	
CLK ↑ → RC valid delay	t_{DKRC}		25	ns	
CLK ↑ → RC float delay	t_{FKRC}			ns	
$\overline{SRDY}$ setup time (CLK ⇓)	t_{SSRYK}	7		ns	
$\overline{SRDY}$ hold time (CLK ⇓)	t_{HKSRY}	8		ns	
$\overline{BERR}$ setup time (CLK ⇓)	t_{SBEK}	7		ns	
$\overline{BERR}$ hold time (CLK ⇓)	t_{HKBE}	8		ns	

DUMP Timing Specifications

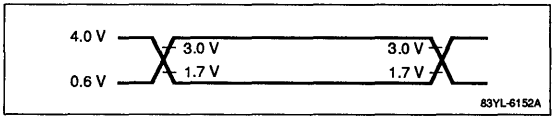
Parameter	Symbol	Min	Max	Unit	Conditions
CLK ↓ → MA_3-MA_{31} delay	t_{DKMA}		25	ns	C_L = 100 pF
CLK ↓ → MA_3-MA_{31} floating	t_{FKMA}			ns	

Timing Waveforms

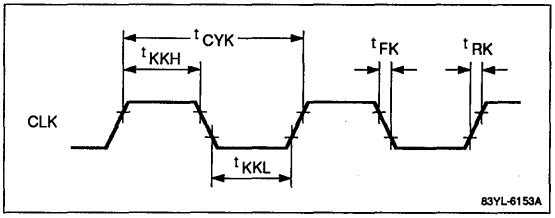
AC Test In/Output (except CLK)



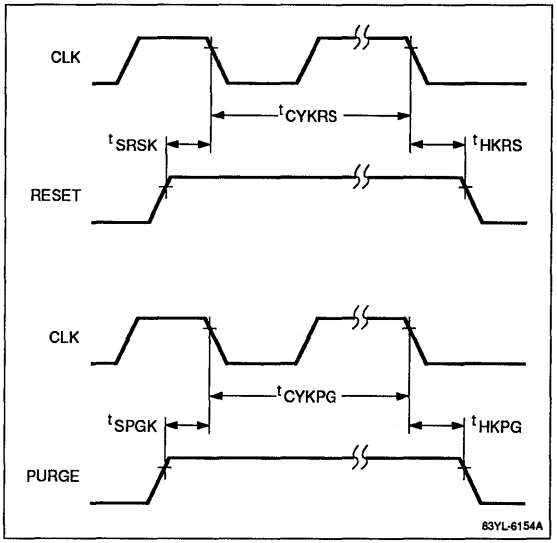
AC Test In/Output (CLK)



Clock Timing

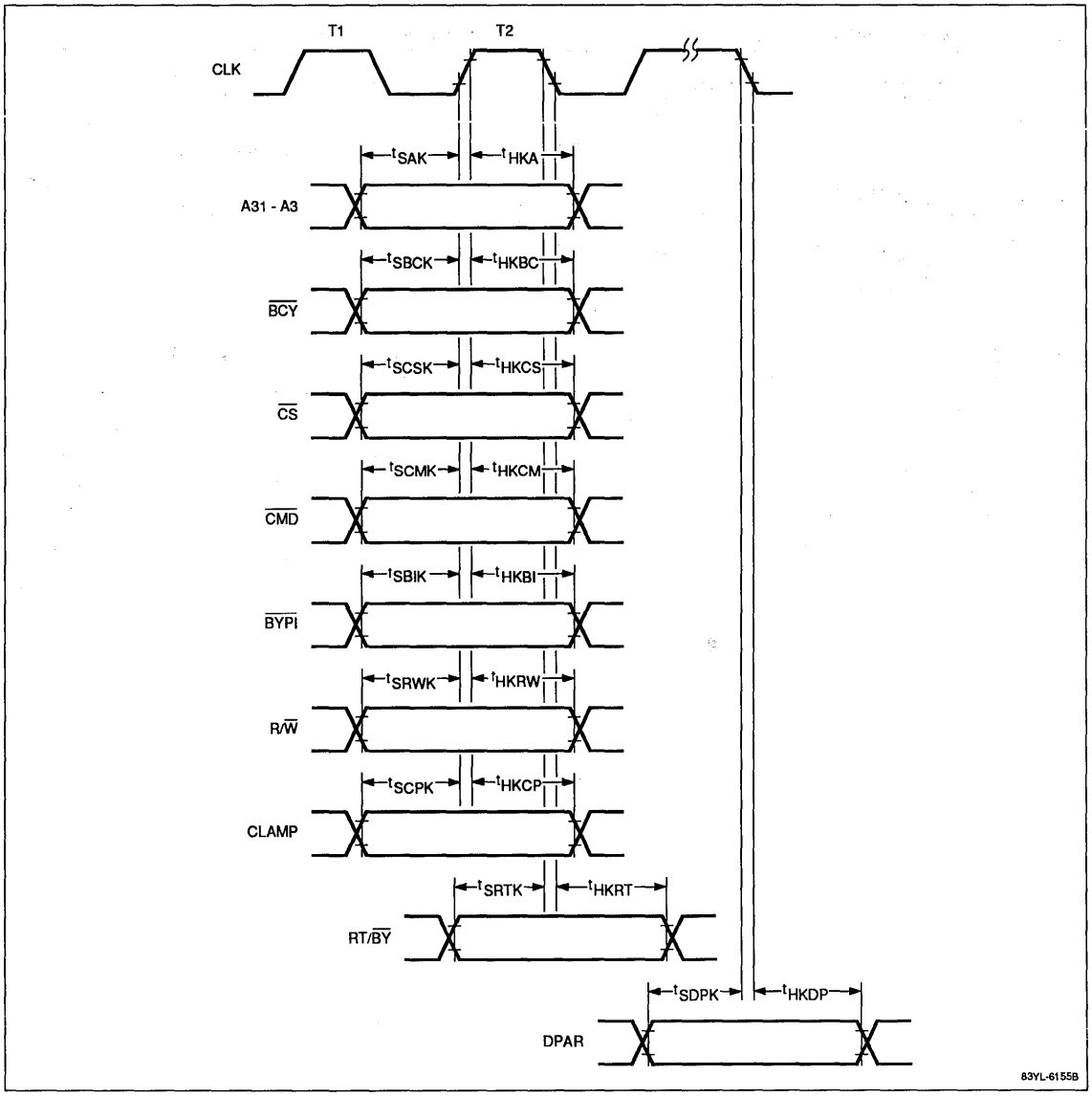


Reset/Purge Timing

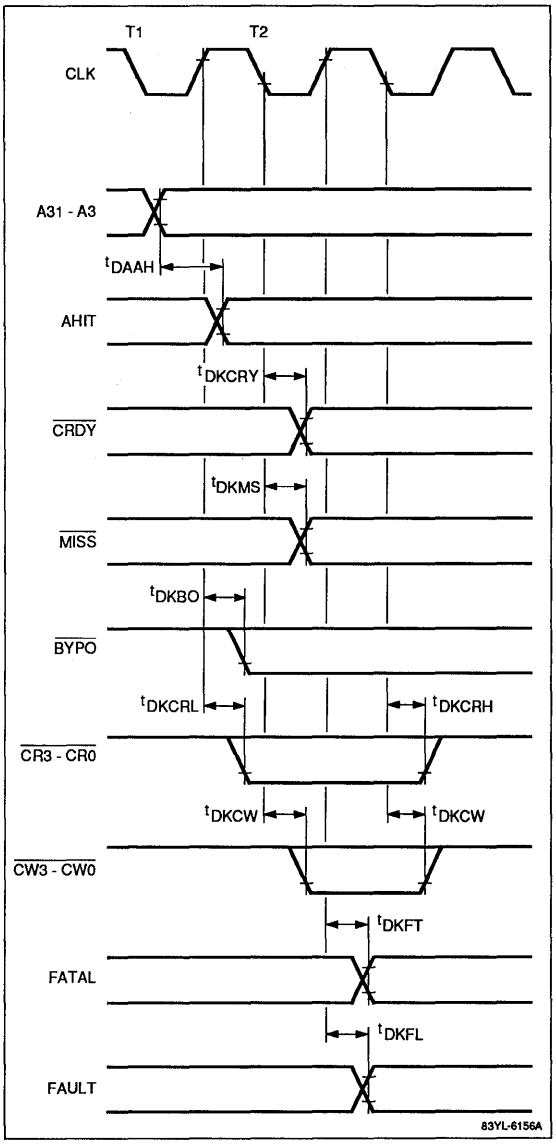


51

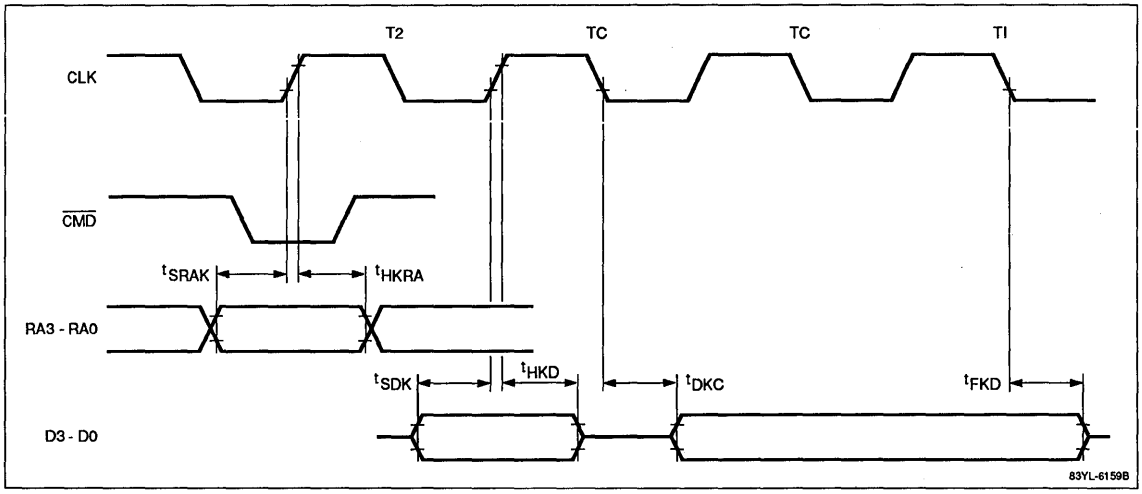
CPU Input


T1
T2
CLK
tSAK
tHKA
A31 - A3
tSBCK
tHKBC
BCY
tSCSK
tHKCS
CS
tSCMK
tHKCM
CMD
tSBIK
tHKBI
BYPI
tSRWK
tHKRW
R/W
tSCPK
tHKCP
CLAMP
tSRTK
tHKRT
RT/BY
tSDPK
tHKDP
DPAR
83YL-6155B

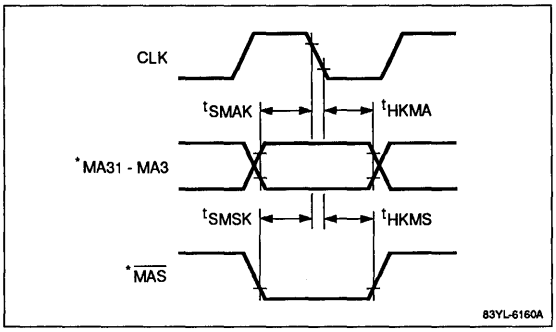
CPU Output


T1
T2
CLK
A31 - A3
tDAAH
AHIT
tDKCRY
CRDY
tDKMS
MISS
tDKBO
BYPO
tDKCRL
tDKCRH
CR3 - CR0
tDKCW
tDKCW
CW3 - CW0
tDKFT
FATAL
tDKFL
FAULT
83YL-6156A

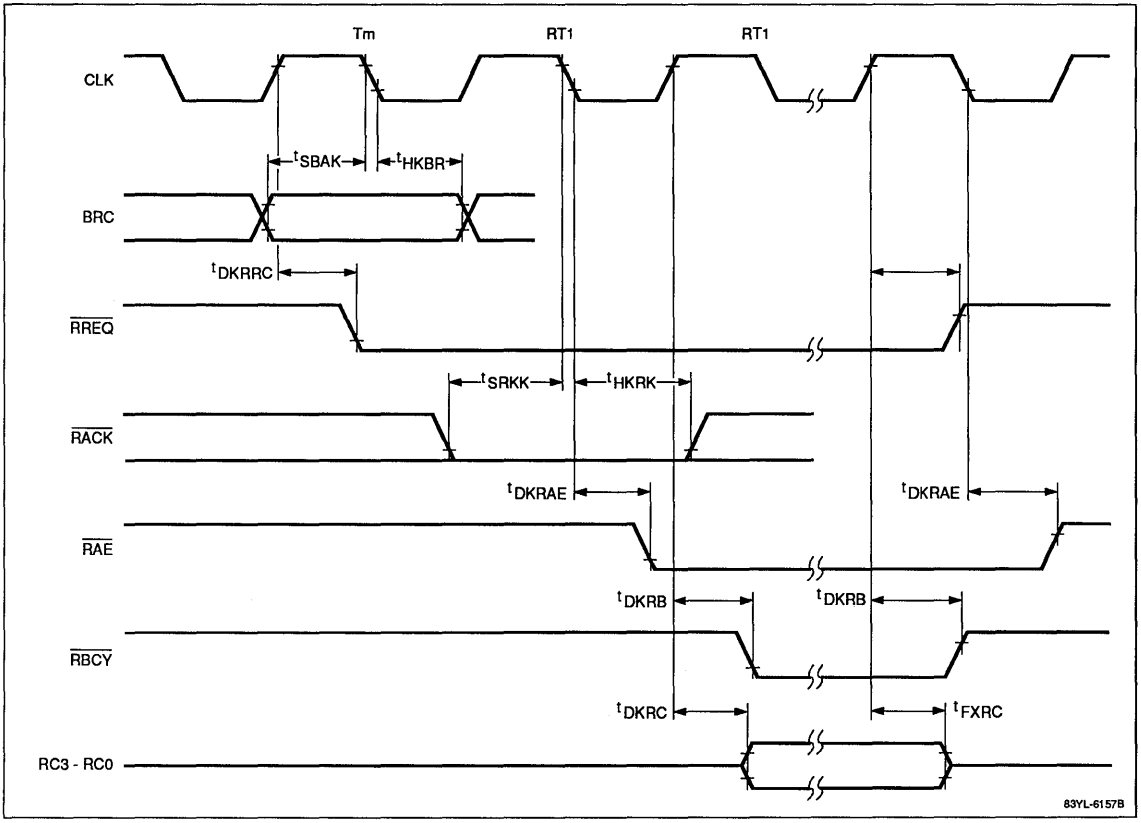
Command Access Timing



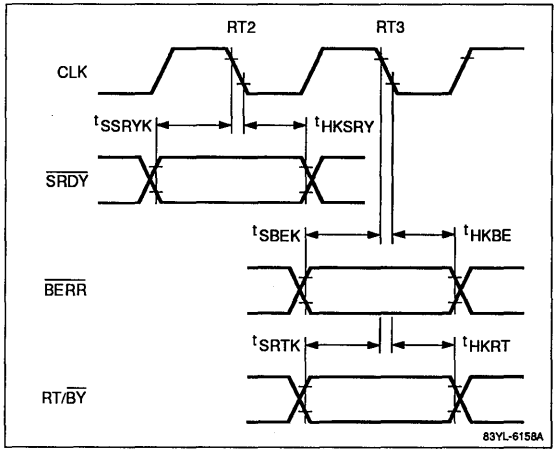
Bus Monitor Timing



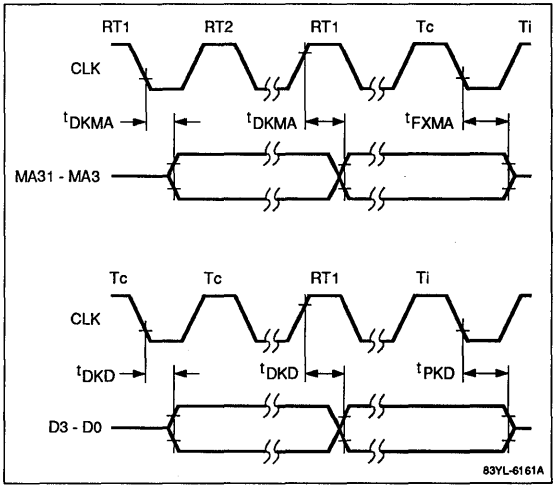
Replace Timing



Bypass Timing



Dump Timing



5I

NEC

Section 6
Development Tools

Description

The CC70116 C Compiler package converts standard C source code into relocatable object modules for the NEC V20® (μPD70108), V30® (μPD70116), V40™ (μPD70208), and V50™ (μPD70216) microprocessors and the V25™ (μPD70320) and V35™ (μPD70330) single-chip microcomputers. These modules are compatible with those produced by the RA70116 V20-V50 Relocatable Assembler package and the RA70320 V25/V35 Relocatable Assembler package and may be linked with other modules using the appropriate linker provided with the assembler package.

Features

- Standard Kernighan and Ritchie C
  - Defined in UNIX System III
- Supports small and large memory models
- NEC enhancements
  - Enumeration data type support
  - Assignment of all members by a structure name
  - Ability to use identical names in identifiers of different types in different structures
  - Addition of a void type to declare functions with no return value
  - Addition of char as a data type for which unsigned can be specified
  - Ability to initialize structures with bit fields
- CC70116 library conforms to UNIX System III specifications
  - MS-DOS® and CP/M-86® dependent functions included
- C macros for sorting and converting ASCII code characters
- User-selectable object code optimizer
- Runs under a variety of operating systems
  - MS-DOS
  - VAX/VMS®
  - VAX/UNIX™ 4.2 BSD or Ultrix®

V20 and V30 are registered trademarks of NEC Corporation.
V25, V35, V40 and V50 are trademarks of NEC Corporation.
UNIX is a trademark of AT&T.
CP/M-86 is a registered trademark of Digital Research Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.
VAX, VMS, and Ultrix are registered trademarks of Digital Equipment Corporation.

Ordering Information

Part Number	Description
CC70116-D52	MS-DOS, 5″ double-density floppy diskette
CC70116-VVT1	VAX/VMS, 9-track 1600BPI magnetic tape
CC70116-VXT1	VAX/UNIX 4.2 BSD or Ultrix, 9-track 1600BPI magnetic tape

Extensions to Standard C

Enumeration data type: Similar to the enumerated type in Pascal, this allows the definition of a limited set of [mnemonic] values for a variable, thus preventing assignment of invalid values to the variable.

Assignment of all members by a structure name: Allows assignment of all the members of a structure from another structure of the same type simply by using the names of the structures.

Ability to use identical names for identifiers of different types in different structures: Allows the same member name in two different structures to have a different name in each.

Addition of a void type: Prevents the use of functions which return no value in expressions where a return value is required.

Addition of char as a data type for which unsigned can be specified: Allows the full 8 bits of the char data type to be used as a value, rather than only the low-order 7 bits.

Ability to initialize structures with bit fields: Allows structure with bit-field members to be initialized via the C construct.

Compiler Options

The CC70116 C Compiler supports the following options during compilation.

A	Generates assembly language listing
B	Specifies drive for compiler phases
C	Leaves comments in C source image file (used with P)
D	Defines a name

E	Outputs C source image to a standard output device after only compiler control line processing
H	Generates an object module for the interrupt handling section of RX116 (realtime operating system)
I	Specifies drive for include files
NS	Suppresses symbol table information in object module
O	Selects optimization
P	Outputs C source image to a file after only compiler control line processing
T	Limits compiler phases to be input from drive selected by B option
U	Nullifies the initial definition of the name defined by D option
X	Generates object code for the large model

Memory Models

The CC70116 supports the small and large memory models. With the small model, a total of 128K bytes of memory can be accessed: 64K bytes for data and 64K bytes for code. With the large model, up to 1M byte of memory can be accessed for code and data. In addition, data constants can be permanently encoded into the ROM space. Programs using the small memory model are more efficient in terms of space utilization and execution speed and are recommended when your program can fit into one data and one code segment.

CC70116 Library Functions

The CC70116 C Compiler library conforms to the UNIX System III library specifications. Some library routines are operating system dependent and may only be used to create programs that run under MS-DOS or CP/M-86; other functions do not call the operating system, and may be used in any program.

In the list below, the asterisk (*) means user system, MS-DOS, or CP/M-86 dependent.

abort*	fread	MSDOS1*	srand
abs	freopen	MSDOS2*	sscanf
access*	fscanf	MSDOS3*	strcat
atoi	fseek	open*	strchr
atol	ftell	perror	strcmp
close*	fwrite	printf	strcpy
creat*	getchar	putchar	strlen
exit*	gets	puts	strncat
fclose	getw	putw	strncmp
fdopen	lseek*	qsort	strncpy
fflush	mktemp	rand	swab
fgetc	mpm1*	read*	ungetc
fgets	mpm2*	rewind	unlink*
fopen	mpm3*	sbrk*	write*
fprintf	mpm4*	scanf	
fputc	mpm5*	setbuf	
fputs	mpm6*	sprintf	

C Macros

Included with the C compiler are C macros for sorting and converting ASCII code characters. These macros are listed below.

_tolower	getchar	isgraph	isxdigit
_toupper	salnum	islower	putc
feof	isalpha	isprint	putchar
ferror	isascii	ispunct	toascii
fileno	iscntrl	isspace	
getc	isdigit	isupper	

Start-Up Modules

Six small and large start-up modules are included in the CC70116 C Compiler package to initialize programs generated by the compiler: modules for embedded systems; modules for executing programs under the MS-DOS operating system; modules for executing programs under the CP/M-86 operating system.

Operating Environment

The CC70116 package can be supplied to run under a variety of operating systems. One version is available for an MS-DOS system with one or more disk drives and at least 128K bytes of system memory. Other versions are available to run on a Digital Equipment Corporation VAX computer with UNIX 4.2 BSD or Ultrix or VMS (Version 4.1 or later) operating systems.

To produce executable programs, a linker (LK70116 or LK70320) and a hexadecimal object code converter (OC70116 or OC70320) program are required and the librarian (LB70116 or LB70320) program is useful. These programs are available separately from NEC Electronics Inc. as part of the RA70116 V20-V50 Relocatable Assembler package or the RA70320 V25/V35 Relocatable Assembler package.

The linker produces an absolute load module containing both symbol and source code line number information and the absolute object code. This module may be loaded directly into the appropriate NEC in-circuit emulator for execution with full symbolic debug capabilities.

The hexadecimal object code converter produces an extended hexadecimal format object code file that may be loaded into a PROM programmer.

Documentation

For further information on source program format, compiler operation, and actual program examples, the following documentation is furnished with the CC70116 compiler package. Additional copies may be obtained from NEC Electronics Inc.

- CC70116, V-Series C Compiler Operation Manual (MS-DOS)
- CC70116, V-Series C Compiler Operation Manual (VMS)
- CC70116, V-Series C Compiler Operation Manual (UNIX)

License Agreement

CC70116 is sold under terms of a license agreement that must be completed and returned to NEC Electronics before the C compiler is shipped. A copy of this license agreement may be obtained from any NEC Electronics Sales Office. Software updates are provided to registered users.

NEC
NEC Electronics Inc.

DDK-70320 Evaluation Board for V25 Microcomputer

Description

The DDK-70320 is an evaluation board for NEC's μPD70320 (V25™) 16-bit, single-chip microcomputer. The DDK-70320 gives you maximum flexibility when evaluating and designing with the μPD70320. It features a μPD70320 with 32K bytes of EPROM, 32K bytes of RAM, two RS-232C communication ports, and a powerful monitor program. The DDK-70320 is on an IBM PC compatible card and includes a prototyping area for building your application-specific hardware.

A copy of RA70320, the V25/V35™ Family Relocatable Assembler package for use on an IBM PC, PC/XT®, PC AT®, or compatible host computer, is shipped with each DDK-70320 to allow you to develop code for evaluation purposes. Also included with the DDK-70320 is an emulator controller program for the IBM PC, a variety of V-Series software utilities, a small demonstration program, and a complete set of documentation. This total package provides you with a fast, efficient means for evaluating the capabilities of the μPD70320 for your application.

Features

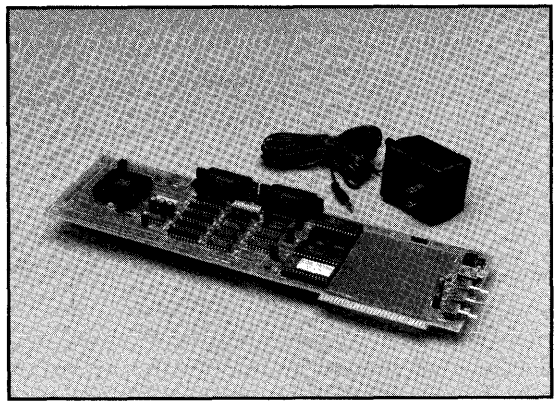
- μPD70320 (V25) Evaluation Board with power supply
- On-board memory
  - 32K-byte EPROM (user expandable to 64K bytes)
  - 32K-byte RAM (user expandable to 64K bytes)
- Powerful on-board debug monitor
  - Real-time and single-step operation
  - Display/change memory and internal registers
  - Display/change special-function registers
  - Line assembler and disassembler
  - Multiple software breakpoints
  - Input/output from I/O ports
  - User program download capability
- Two RS-232C serial interfaces
  - One for a terminal or host computer
  - One for a user application
- Prototyping area for user circuitry
- IBM PC card form factor
- RA70320 V25/V35 Relocatable Assembler package
- Host control software for IBM PC, PC/XT, PC AT, or compatibles
- V-Series software utilities and demonstration programs

Ordering Information

Part Number	Description
DDK-70320	μPD70320 Evaluation Board

V25 and V35 are trademarks of NEC Corporation.
PC/XT and PC AT are registered trademarks of International Business Machines Corporation.

DDK-70320 Evaluation Board



Hardware

The DDK-70320 (figure 1) features 64K bytes of on-board memory: 32K bytes of EPROM and 32K bytes of RAM. The EPROM area, which is dedicated to a powerful monitor program, can be expanded up 64K bytes. The RAM area can also be expanded to 64K bytes and contains the interrupt vector space, the monitor work area, and a user area for program downloading.

A μPD71051 serial communications controller is connected through an RS-232C driver/receiver to a DB25 pin connector. The monitor uses this serial interface to communicate with an external terminal or host computer. A second RS-232C channel is connected to serial port 0 of the V25 for user applications.

A RESET switch returns the DDK-70320 to the power-up state with or without losing the contents of the external RAM. An NMI switch returns control to the monitor from a user program while saving the user's state.

An ac/dc converter supplies power to the DDK-70320 in the stand-alone mode. The DDK-70320 can also receive its power directly from the IBM PC bus.

Software

The DDK-70320 comes with a powerful interactive monitor to facilitate software design using the μPD70320. A user program can be downloaded into user RAM and executed in real-time with or without breakpoints, or it can be executed one instruction at a time. During single-stepping, the registers, program counter, and the next instruction to be executed are displayed.

The DDK-70320 has 10 address breakpoints, which are set in the GO command line. The monitor sets a breakpoint by substituting a BRK 3 instruction (opcode CCH) for an instruction in the user's program.

Additional commands are available to display, fill, change, or move memory; display or change the general-purpose and special-function registers; input from or output to an I/O port; disassemble memory; assemble a line of source code; and display the command list. Table 1 lists all of the DDK-70320 monitor commands.

Figure 1. DDK-70320 Block Diagram

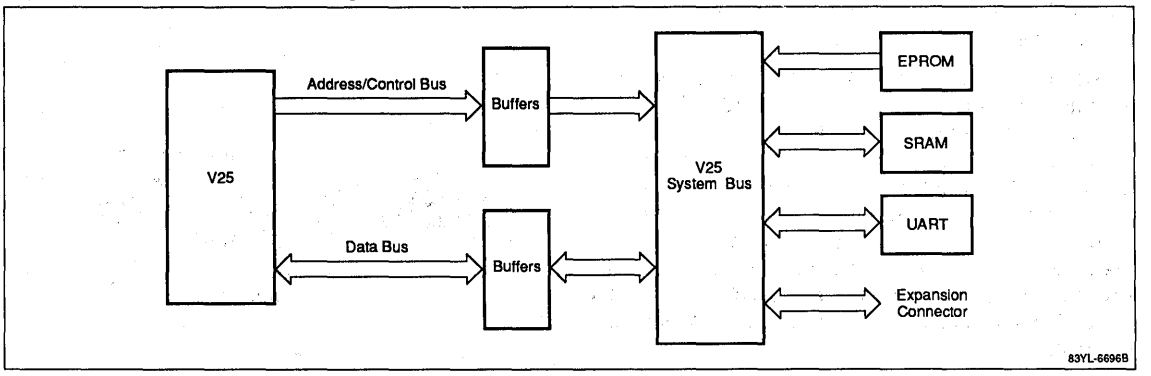


Table 1. DDK-70320 Command List

Command	Description
A	A(ssemble) a line of source code
D	D(ump) memory
E	E(nter) memory
F	F(ill) memory
G	G(o) with optional breakpoints
H	H(elp) menu
I	I(nput) a byte from an I/O port
L	L(oad) a HEX file onto the DDK-70320
M	M(ove) a block of memory
O	O(utput) a byte to an I/O port
R	R(egister) display/alter R[reg]
S	S(pecial) function register display/alter
T	T(race) execution
U	U(nassemble) a block of memory
?	H(elp) menu

RA70320 Relocatable Assembler Package

The RA70320 Relocatable Assembler package converts symbolic source code for NEC's V25/V35 family of microcomputers into executable absolute address object code. A copy of RA70320 is included with the DDK-70320 for use on an IBM PC, PC/XT, PC AT, or compatibles. With this software, you can easily write evaluation programs for the μPD70320.

Emulator Controller Program

Absolute address object files produced by the RA70320 Relocatable Assembler package can be downloaded to the DDK-70320 using the NEC Emulator Controller program supplied with the DDK-70320. This controller program allows you to download files from your IBM PC or compatible to the DDK-70320 board. The program provides the following additional capabilities.

- Complete DDK-70320 control from host console
- On-line help facilities
- Host system directory and file display
- Storage of debug session on disk

Documentation

For further information, refer to the DDK-70320 User's Manual supplied with the evaluation board. Additional copies may be obtained from NEC Electronics Inc.

License Agreement

RA70320 is provided under the terms of a license agreement included with the DDK-70320 board. The accompanying card must be completed and returned to NEC Electronics Inc. to register the license. Software updates are provided to registered users.

DDK-70330 Evaluation Board for V35 Microcomputer

Description

The DDK-70330 is an evaluation board for NEC's μPD70330 (V35™) 16-bit, single-chip microcomputer. The DDK-70330 gives you the maximum flexibility when evaluating and designing with the μPD70330. It features a μPD70330 with 128K bytes of EPROM, 128K bytes of RAM, two RS-232C communication ports, and a powerful monitor program. The DDK-70330 board includes a prototyping area for building your application specific hardware.

A copy of RA70320, the V25™/V35™ Family Relocatable Assembler package for use on an IBM PC, PC/XT®, PC AT®, or compatible host computer, is shipped with each DDK-70330 to allow you to develop code for evaluation purposes. Also included with the DDK-70330 is an emulator controller program for the IBM PC, a variety of V-Series software utilities, a small demonstration program, and a complete set of documentation. This total package provides you with a fast, efficient means for evaluating the capabilities of the μPD70330 for your application.

Features

- μPD70330 (V35) Evaluation Board with power supply
- On-board memory
  - 128K-byte EPROM
  - 128K-byte RAM (user expandable to 512K bytes)
- Powerful on-board debug monitor
  - Real-time and single-step operation
  - Display/change memory and internal registers
  - Display/change special-function registers
  - Line assembler and disassembler
  - Multiple software breakpoints
  - Input/output from I/O ports
  - User program download capability
- Two RS-232C serial interfaces
  - One for a terminal or host computer
  - One for a user application
- Prototyping area for user circuitry
- RA70320 V25/V35 Relocatable Assembler package
- Host control software for IBM PC, PC/XT, PC AT, or compatibles
- V-Series software utilities and demonstration programs

Ordering Information

Part Number	Description
DDK-70330	μPD70330 Evaluation Board

Hardware

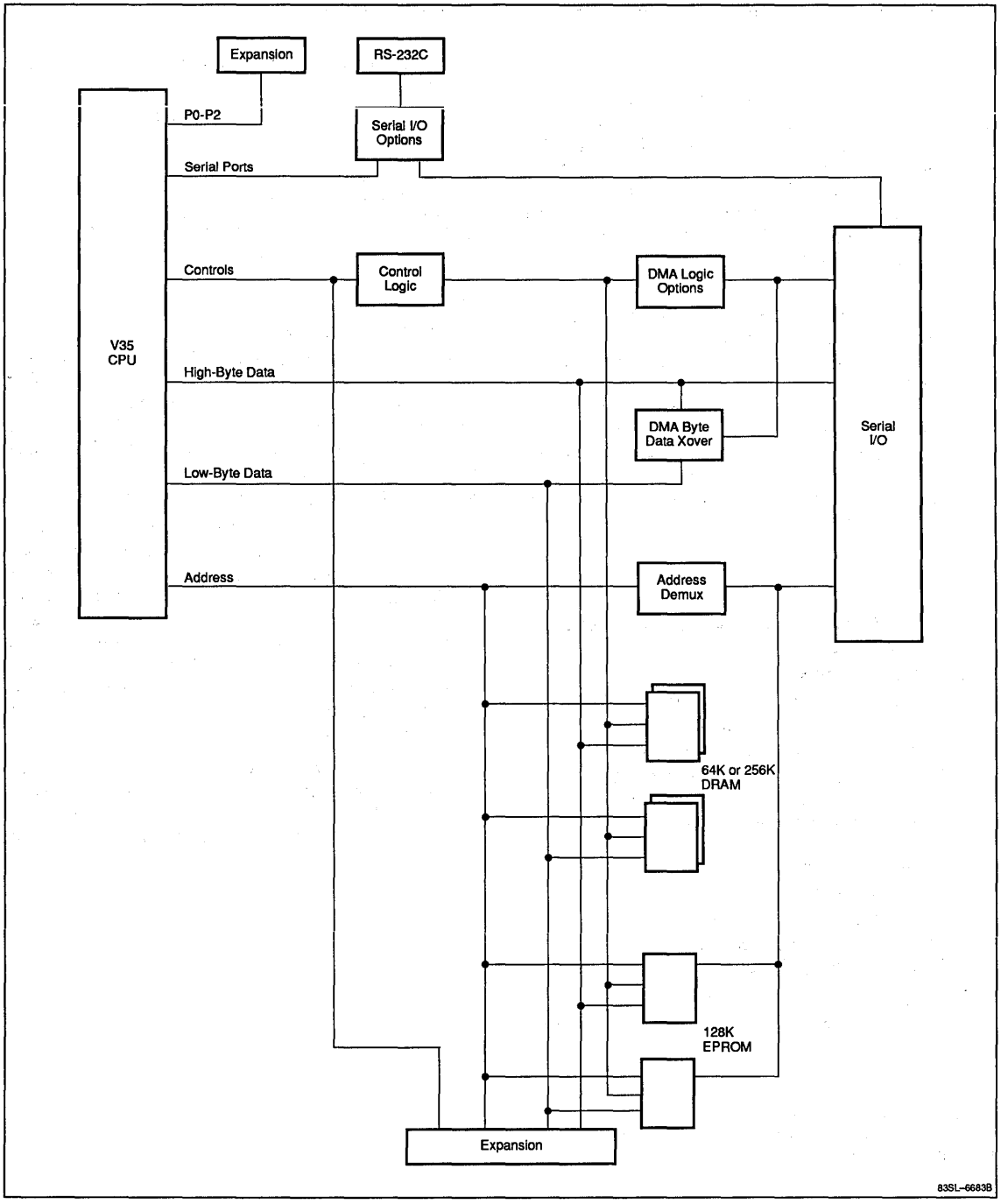
The DDK-70330 (figure 1) features 256K bytes of on-board memory: 128K bytes of EPROM and 128K bytes of RAM. The EPROM area includes a powerful monitor program and a user area. The RAM area can be expanded to 512K bytes by replacing the 64K x 4 dynamic RAMS with 256K x 4 dynamic RAMS. This RAM space contains the interrupt vectors, the monitor work area, and a user area for program downloading.

A μPD72001 multiprotocal serial communications controller is connected through an RS-232C driver/receiver to a DB25 pin connector. The monitor uses this serial interface to communicate with an external terminal or host computer. A second RS-232C channel is available for connection to the second channel of the μPD72001 or one of the serial ports of the μPD70330 for user applications. DMA interface circuitry between the μPD72001 and the V35 is provided to allow the you to evaluate the DMA capabilities of both devices.

A RESET switch returns the DDK-70330 to the power-up state with or without losing the contents of the external RAM. An NMI switch returns control to the monitor from a user program while saving the user's state. An ac/dc converter supplies power to the DDK-70330.

V25 and V35 are trademarks of NEC Corporation.
PC/XT and PC AT are registered trademarks of International Business Machines Corporation.

Figure 1. DDK-70330 Block Diagram



Software

The DDK-70330 comes with a powerful interactive monitor to facilitate software design using the μPD70330. A user program can be downloaded into user RAM and executed in real-time with or without breakpoints, or it can be executed one instruction at a time. During single-stepping, the registers, program counter, and the next instruction to be executed are displayed.

The DDK-70330 has 10 address breakpoints, which are set in the GO command line. The monitor sets a breakpoint by substituting a BRK 3 instruction (opcode CCH) for an instruction in the user's program.

Additional commands are available to display, fill, change, or move memory; display or change the general-purpose and special-function registers; input from or output to an I/O port; disassemble memory; assemble a line of source code; and display the command list. Table 1 lists all of the DDK-70330 monitor commands.

Table 1. DDK-70330 Command List

Command	Description
A	A(ssemble) a line of source code
D	D(ump) memory
E	E(nter) memory
F	F(ill) memory
G	G(o) with optional breakpoints
H	H(elp) menu
I	I(nput) a byte from an I/O port
L	L(oad) a HEX file onto the DDK-70330
M	M(ove) a block of memory
O	O(utput) a byte to an I/O port
R	R(egister) display/alter R[reg]
S	S(pecial) function register display/alter
T	T(race) execution
U	U(nassemble) a block of memory
?	H(elp) menu

RA70320 Relocatable Assembler Package

The RA70320 Relocatable Assembler package converts symbolic source code for NEC's V25/V35 family of microcomputers into executable absolute address object code. A copy of RA70320 is included with the DDK-70330 for use on an IBM PC, PC/XT, PC AT, or compatibles. With this software, you can easily write evaluation programs for the μPD70330.

Emulator Controller Program

Absolute address object files produced by the RA70320 Relocatable Assembler package can be downloaded to the DDK-70330 using the NEC Emulator Controller program supplied with the DDK-70330. This controller program allows you to download files from your IBM PC or compatible to the DDK-70330 board. The program provides the following additional capabilities.

- Complete DDK-70330 control from host console
- On-line help facilities
- Host system directory and file display
- Storage of debug session on disk

Documentation

For further information, refer to the DDK-70330 User's Manual supplied with the evaluation board. Additional copies may be obtained from NEC Electronics Inc.

License Agreement

RA70320 is provided under the terms of a license agreement included with the DDK-70330 board. The accompanying card must be completed and returned to NEC Electronics Inc. to register the license. Software updates are provided to registered users.

NEC
NEC Electronics Inc.

IE-70136 In-Circuit Emulator for μPD70136 (V33) Microprocessor

Description

The IE-70136 is a portable, stand-alone, in-circuit emulator that provides hardware emulation and software debug capabilities for the μPD70136 (V33™) 16-bit microprocessor.

Real-time and single-step emulation—coupled with software performance analysis, sophisticated memory mapping, symbolic debugging, macrofile command facilities, user-programmable breakpoints and trace qualifiers—create a powerful development environment.

Command entry is simplified by eight dynamically reprogrammed function keys, called softkeys, that visually prompt a user with the next level of commands. User programs can be uploaded/downloaded from a variety of host systems by a serial link or they can be loaded directly from an MS-DOS® disk.

Features

- Portable, stand-alone, in-circuit emulator
  - 9.5-inch amber CRT display
  - Two 5-inch, 640K-byte floppy-disk drives
  - ASCII keyboard with eight function keys
  - EPROM programmer: 2732, 2764, 27128, 27256, 27512
  - Supports NEC's V20®, V30®, V25™, V35™, V40™, V50™, V53™, and V60™ microprocessors
- Precise real-time and single-step emulation
  - Selectable internal clock: 8 MHz or 16 MHz
  - Up to 16-MHz external TTL clock
- Emulation memory bus size selection
- 256K bytes of memory for prototype memory emulation; mappable into 16M-byte address space in 4K-byte units
- Automatic break with disconnected target I/O access
- Six user-programmable hardware breakpoints
  - Real-time break on address, data, CPU status, or external input
  - Selectable as execution or nonexecution
- 256 user-programmable software breakpoints
- Trace buffer: disassemble, frame, and code trace display
  - 8192 frames by 64 bits
  - Programmable enable/disable points
- Performance modes available for software performance evaluation: time interval, module activity, and count modes
- Full symbolic debug capabilities
- Symbolic line assembler and disassembler
- Macrofile command capability
- Dual window display in emulation mode
- Softkey and menu-driven user input
- System commands are available while in emulation mode

V20 and V30 are registered trademarks of NEC Corporation.
V25, V33, V35, V40, V50, V53, and V60 are trademarks of NEC Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.

Ordering Information

Part Number	Description
IE-70136-A016	In-circuit emulator for μPD70136 (V33)
EP-70136L-A	68-pin PLCC/PGA emulation probe/socket
EP-70136GJ-A	74-pin plastic QFP emulation probe

HARDWARE DESCRIPTION

The IE-70136 (figure 1) consists of a system chassis with a detachable ASCII keyboard and an emulation pod unit. The chassis houses a 9.5-inch amber CRT, two 5-1/4 inch, 640K-byte floppy-disk drives, an EPROM programmer, a card cage, a power supply, and three control boards. The boards are main CPU, expansion system memory, and emulation pod interface.

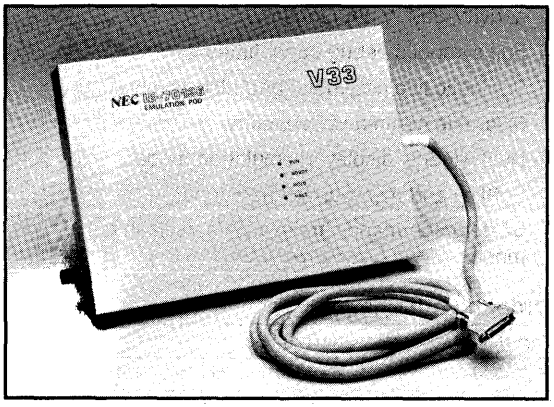
The main CPU board contains a supervising CPU, 512K bytes of system memory, and the peripheral interfaces. The expansion system memory board provides an additional 512K bytes of system memory.

The emulation pod interface contains the clock selection logic, emulation bus sizing switch, and enabling of the RDY and BS8/BS16 signals from the target to emulator memory. The emulation pod contains the V33 EVACHIP, external event input, trigger out, and target interface. This unit connects to the target system by the EP-70136L-A emulation probe for the 68-pin PLCC/PGA. For the 74-pin plastic QFP, an EP-70136GJ-A probe adapter is also needed.

The IE-70136 supports the following external interfaces: two RS-232C serial ports, one Centronics parallel printer port, and one RGB video output.

The emulator can be converted to support NEC's V20, V25, V30, V35, V40, V50, V53, and V60 CMOS microprocessors by exchanging the appropriate control boards and the emulation pod unit.

Figure 1. IE-70136 System Configuration



Memory and I/O Mapping Capabilities

The IE-70136 contains 256K bytes of zero wait-state emulation memory; 252K bytes are available for emulating target RAM (read/write) or ROM (read only) and the remaining 4K bytes are reserved for use by the system monitor.

The complete 16M-byte memory space of the μPD70136 must be mapped into one of the following categories.

Target	Memory resident in target system (read/write)
ROM	External ROM emulation memory (read only)
RAM	External RAM emulation memory (read/write)
Locked	Access inhibited memory (remaining unmapped addresses)

All memory mapping is executed in 4K-byte blocks using the Configure and Mapping softkey commands. If an address mapped as "locked" is accessed or a write is attempted to ROM, a break in the emulation will occur.

The I/O space of the μPD70136 is always mapped to the target. If the probe is not connected to a target system, a break will occur in emulation when any target I/O is accessed.

The target NMI and INT signals may be enabled or disabled by the Configure command.

Emulation

The IE-70136 executes μPD70136 user programs in real time in three different modes: break, trace, and performance. In break emulation mode, the program is run in real time or in single-step mode until a breakpoint is encountered. In trace emulation mode, the program is executed in real time while filling the trace buffer with bus information.

In performance emulation mode, the emulator can:

(1) Measure the execution time between enable and disable points

(2) Measure the accumulated execution time in three areas of memory and then display the ratio of time spent within each area in both absolute and relative terms

(3) Count the number of times a particular trigger point is satisfied between enable and disable points.

Once emulation is stopped in either break or trace mode, the trace automatically displays one screen of data, ending on the last instruction executed. In performance mode, time is shown in a table and in a bar graph. At this point, it is possible to display the contents of memory, the general-purpose and special registers, the symbol tables, directories, and other information. All can be displayed individually or in split screen with the trace display. The windows may be scrolled independently.

Break Capabilities

The IE-70136 has six hardware breakpoints. Five can be set to occur in response to a real-time event with the BCond command. The remaining one is reserved for setting a real-time address (execution only) breakpoint with the GO command.

You can use the BCond command to set breakpoints to occur on an address, a data value, or a CPU state. Hardware breakpoints can be either an execution or nonexecution type. An execution break occurs when an instruction is clocked into the execution unit of the V33. A nonexecution break occurs when the bus condition is satisfied regardless of the status of the instruction queue.

An enable and disable point can also be specified to indicate when detection of hardware break and trace is to start and stop. If not specified, trace and break detection is enabled when the GO command is exe-

cuted. The external channel can also be used to create a breakpoint. Either the rising or falling edge can be selected.

Up to 256 software breakpoints can be set plus an additional one in the GO command. To set a software breakpoint, the emulator replaces an instruction in the user's program with a BRK 0 instruction. A break will occur when this instruction is executed, and the user's program will be restored. This capability is not available for program code executing out of ROM. A Check command is provided to verify that the breakpoint can actually be set.

Trace Capabilities

The trace buffer is 8192 frames by 64 bits wide and sampling is done on every machine cycle. The buffer is filled in round-robin fashion. The emulator traces the external address and data buses, the CPU and queue status, and control signals.

The IE-70136 has five trace trigger points. When the trigger condition is true, a positive logic pulse is output on the TRIGGER OUT pin of the emulator pod. Separate ENABLE and DISABLE points can be set to restrict tracing or trigger detection to a particular section of program code. If not used, tracing and trigger detection will be enabled when the GO command is issued. An address point can be set with the GO command to record the trace after, about, or before the condition is true. The external input can also be used as a trigger point. Either a positive or negative edge can be specified.

The trace data may be displayed in one of three modes: disassembly, frame, or code. In disassembly mode, fetch and execution cycles are displayed separately, and bus cycles are rearranged so that a fetch cycle and its associated execution cycle are displayed next to each other. In frame mode, the CPU bus cycle conditions are displayed as traced. In code mode, only the executed fetch cycles are displayed. In disassembly and code modes, the fetch cycle is displayed bright, the execution cycle is dim, and a trigger point is indicated by reverse video.

SOFTWARE DESCRIPTION

Software Performance Analysis

Using the IE-70136, software performance can be evaluated in real-time in any one of three different modes: run-time interval mode, module activity mode, and count mode. Improvements in software performance can be statistically measured.

Run-Time Interval Mode. Execution time from the enable point to the disable point is repeatedly measured, and the mean value, the maximum value, the minimum value, and the distributed values are stored. The measured distribution values, from a maximum of six time intervals, can be displayed graphically.

Module Activity. The accumulated execution time for three areas of code (from enable to disable points) can be measured. The ratio of time spent in each one compared to the total measured time (ABS) and the ratio of time spent in each one compared to the time spent in all three areas (REL) can be displayed. If the overall processing speed does not satisfy the desired speed, the portion of the code taking more time can be determined and improved. This mode can be very effective for increasing total throughput.

Count Mode. The number of times a trigger point is satisfied within an area of code (from enable to disable points) can be counted. Up to four trigger points can be set and the ratio of the number of times each trigger point is satisfied to the total number of all triggers is displayed.

Address Specification

An address can be specified as either a 24-bit absolute address (beginning with a ;) or as a segment:offset address. Segment:offset addresses can be either physical, addressing only the base 1M byte, or logical. Logical addresses are converted to 24-bit physical addresses by the emulator according to the address extension table (EXPAND TABLE) specified by the SEText command.

There are two types of expand tables. One uses the MMU (Memory Management Unit) in the CPU. However, conversion is done only during a break if the XA flag is set. Otherwise, no conversion is performed. A user-defined expand table is independent of the MMU. This table is defined by loading the extended linker locator output file (EL70136), by definition in the program stage, or by copying the contents of the MMU. With the user-defined table, conversion is performed regardless of the status of the XA flag.

If the user-defined expand table is selected, conversion back to logical address is performed according to this table when displaying trace results in disassembly mode.

System Software

The IE-70136 is controlled by the MIOS/U proprietary operating system. Command input is simplified by eight function keys, providing a choice of up to 24 softkeys within any menu level. The dynamically reprogrammed softkeys visually prompt with the next valid set of com-

mands. The softkeys are at the bottom of the display screen and correspond to the eight function keys on the keyboard. To select a command, press a softkey. The softkeys are automatically relabeled with the next set of commands.

All system functions can be used in emulator mode. The IE-70136 system software uses an overlay method so that a necessary program is loaded when a command is input and then executed. File handling, communications, and PROM writing can be performed during emulation, reducing development time.

Table 1 shows some of the utility programs provided with the emulator.

Table 1. IE-70136 Utility Programs

Utility	Function
EMUV35	IE-70136 emulator software
KERMIT	Communication program for file transfer
FILESERV	File management for system disks
EDITOR	Full screen editor
FORMAT	Floppy-disk formatter
PROM	Built-in EPROM programmer control program
TERMINAL	Terminal utility program for file transfer between emulator and another intelligent device
SYMBOL	Symbol table converter; converts non-SROC symbol formats to SROC format
OBJCONV	Object file converter: converts object files to and from the Motorola SROC format
TIMESET	Internal battery backed-up clock and calendar setting
DEFINE	Soft key definition
MDEVICE	Disk format specification

Connecting to Host Systems

Host systems may be connected to the IE-70136 using the RS-232C connectors at the rear of the machine. Parameters such as baud rates, character length, parity, and number of stop bits are software programmable to suit the system being attached. The KERMIT communications program is supplied with the emulator and can be used for uploading and downloading files. NEC currently provides KERMIT for the VAX® under VMS® and UNIX™ 4.2BSD or Ultrix™, the IBM PC, IBM PC/XT®, IBM PC/AT®, or compatibles under PC-DOS® or MS-DOS.

Files may also be transferred to the emulator via the RS-232C ports using the TERMINAL utility. The emulator acts as a terminal for data transfer.

Another means of loading files into the IE-70136 is available with the Multiple File Handler utility, a program that runs in the emulator itself and is supplied as part of the IE-70136 package. The Multiple File Handler allows the emulator to read MS-DOS disks, among others.

Symbolic Debug and Line Assembly/ Disassembly

The IE-70136 supports complete symbolic debugging of programs produced by NEC's RA70136 Relocatable Assembler Package and various third-party software packages, including those from Intel and Microsoft. The symbols can be used as address and data constants in break, trace, and emulation control commands and are displayed during disassembly. A symbolic line assembler is also available to make modifications to existing programs or to enter code from the keyboard.

SPECIFICATIONS

Table 2 gives the electrical, environmental, and physical specifications of the equipment.

Table 2. IE-70136 Specifications

AC power	90 to 132 V, 50/60 Hz, 400 W maximum
Temperature	Operating: +5 to +40°C
	Storage: −20 to +50°C
Relative humidity (noncondensing)	Operating: 20 to 80%
	Storage: 10 to 90%
Weight	Main chassis: 40 pounds
	Pod and cables: 4-3/4 pounds
Dimensions (L x W x H)	19.7 x 16.7 x 8.7 inches

DOCUMENTATION

The following manuals are supplied with the in-circuit emulator. Additional copies may be obtained from NEC Electronics Inc.

- IE-70136 In-Circuit Emulator User's Manual
- IE-70XXX-A Hardware User's Manual
- IE-70XXX-A Software Utilities User's Manual

VAX and VMS are registered trademarks of Digital Equipment Corporation.
Ultrix is a trademark of Digital Equipment Corporation.
UNIX is a trademark of AT&T.
PC/XT, PC AT, and PC-DOS are registered trademarks of International Business Machines Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.

NEC
NEC Electronics Inc.

IE-70136-PC In-Circuit Emulator for μPD70136 (V33) Microprocessor

Description

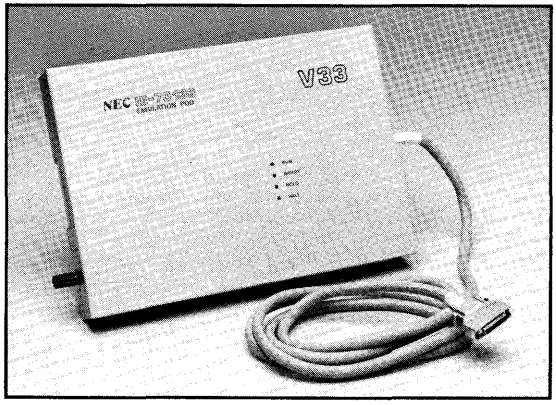
The IE-70136-PC is a low-cost in-circuit emulator that provides hardware emulation and software debug capabilities for the NEC μPD70136 (V33™) 16-bit microprocessor. It is designed to be used with an IBM PC, PC/XT®, PC AT®, or compatible machine under PC-DOS® or MS-DOS®.

Features

- Interfaces to host PC via a parallel interface
  - PC interface card and cable included
- Real-time and single-step emulation
  - 16-MHz internal clock
  - 2- to 16-MHz external clock
  - 65,535 instructions can be stepped in single command
- One 64K-byte block of emulation memory
- Four user-programmable breakpoints
  - Real-time break on address during instruction fetch, data memory read/write, or I/O read/write
- User-programmable software breakpoints
  - 100 logical/physical address breakpoints maximum
  - Forced break possible
- Memory display/fill/move capability in byte or word format
  - Access to normal or extended memory
  - Disassembler
- Macro/batch processing capabilities
- Direct access to PC-DOS and MS-DOS system commands
- LED displays for CPU status: RUN, NOREADY, HOLD, HALT
- Load/Save programs using Extended COFF, NEC LNK, Intel Extended HEX, and Motorola SROC formats

V33 is a trademark of NEC Corporation.
IBM PC, PC/XT, PC AT, and PC-DOS are registered trademarks of International Business Machines Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.

Figure 1. IE-70136-PC System Configuration



Ordering Information

Part Number	Description
IE-70136-PC	In-circuit emulator for μPD70136/70332
EP-70136L-PC	68-pin PLCC/PGA emulation probe/socket

6e

NEC
NEC Electronics Inc.

IE-70208, IE-70216 In-Circuit Emulators for μPD70208 (V40) and μPD70216 (V50) Microprocessors

Description

The IE-70208 and IE-70216 are portable, stand-alone, in-circuit emulators providing both hardware emulation and software debug capabilities for the NEC μPD70208 (V40™) and μPD70216 (V50™) 16-bit microprocessors.

Real-time and single-step emulation in both native and 8080 emulation modes—coupled with sophisticated memory mapping, user-programmable breakpoints and trace qualifiers, symbolic debug, and macrofile command facilities—create a powerful development environment.

Command entry is simplified by eight dynamically reprogrammed function keys, called softkeys, that visually prompt the user with the next level of commands. User programs can be uploaded/downloaded from a variety of host systems by a serial link or loaded directly from an MS-DOS® disk.

Features

- Portable, stand-alone, in-circuit emulator
  - 9.5-inch amber CRT display
  - Two 5-inch, 640K-byte floppy-disk drives
  - ASCII keyboard with eight function keys
  - EPROM programmer: 2732, 2764, 27128, 27256, 27512
  - Supports NEC's V20®, V25™, V30®, V33™, V35™, V53™, and V60™ microprocessors
- Precise real-time and single-step emulation
  - Programmable internal clock: 2 to 10 MHz in 1-kHz steps
  - External clock: 2 to 10 MHz
- Memory and I/O space mappable in 1K-byte blocks
- 256K bytes of memory for prototype memory emulation; expandable to 768K bytes
- Eight user-programmable hardware breakpoints
  - Real-time break on address, data, CPU status, or external probes
  - Break on pass count and register, memory, or I/O values
  - Selectable as execution or code fetch break
- 16 user-programmable software breakpoints
- Trace buffer: machine cycle, mnemonic, and jump trace display
  - 1024 frames by 64 bits
  - Programmable trigger point and trace qualifiers
- Eight optional probes for tracing target system signals
- Full symbolic debug capabilities
- Symbolic line assembler and disassembler
- Macrofile command capability
- Dual window display in emulation mode
- Softkey and menu driven user input

Ordering Information

Part Number	Package
IE-70208-A010	In-circuit emulator for μPD70208 (with V40 pod)
IE-70216-A010	In-circuit emulator for μPD70216 (with V50 pod)
IE-70000-2958	68-pin PLCC package emulation probe
IE-70000-2959	68-pin PGA package emulation probe
IE-70208-2010	Optional pod unit for μPD70208 emulation
IE-70216-2010	Optional pod unit for μPD70216 emulation
IE-70000-2954	Optional external logic probes
IE-70000-2957	Optional 512K-byte expansion emulation memory

HARDWARE DESCRIPTION

The IE-70208/216 (figure 1) consists of a system chassis with a detachable ASCII keyboard and an emulation pod unit. The chassis houses a 9.5-inch amber CRT, two 5-1/4 inch 640K-byte floppy disk drives, an EPROM programmer, card cage, power supply, and three control boards. The boards are main CPU, emulation control, and trace-emulation memory.

The main CPU board contains the supervising CPU, 512K bytes of system memory, and the peripheral interfaces. The emulation control board controls the memory mapping, event detection, and the break and emulation CPU status circuitry. The trace-emulation board contains a trace buffer and 256K bytes of external emulation memory. The optional IE-70000-2957, a 512K-byte expansion emulation memory board, may be installed to increase the external emulation memory to 768K bytes.

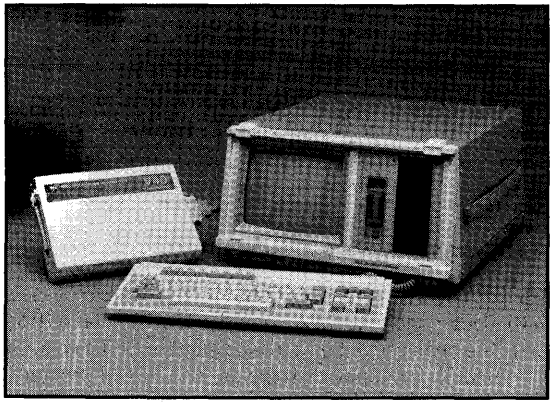
V20 and V30 are registered trademarks of NEC Corporation
V25, V33, V40, V50, V53, and V60 are trademarks of NEC Corporation

The external emulation pod unit houses the emulation CPU, high-speed buffers, and clock selection logic. The emulation pod unit can be connected to the target system with one of two emulation probes: IE-70000-2958 supports the 68-pin PLCC package and the IE-70000-2959 supports the 68-pin PGA package. These probes are ordered separately.

The IE-70208/216 supports the following external interfaces: two RS-232C serial ports, one Centronics parallel printer port, an RGB video output, and eight optional external logic probes. These probes (IE-70000-2954) can be used for tracing and/or breaking on signals from the target system.

Ease of migration within the V-Series family of emulators is provided in the system design. To alternate between V40 and V50 emulation simply requires changing the emulation pod unit. The emulator can also be converted to support the V20, V25, V30, V33, V35, V53, and V60 CMOS microprocessors by exchanging the appropriate control boards and the emulation pod unit.

Figure 1. IE-70208/216 System Configuration



Memory and I/O Mapping Capabilities

The IE-70208/216 contains 256K bytes (expandable to 768K bytes) of emulation RAM (0 wait states) for emulating external RAM or ROM. The complete 1M-byte memory space of the μPD70208/70216 must be mapped into one of the following categories of memory:

Target	Memory resident in target system (read/write)
ROM	External ROM emulation memory (read only)
RAM	External RAM emulation memory (read/write)
Locked	Access inhibited memory (remaining unmapped addresses)

All memory mapping is done in 1K-byte blocks using the Configure and Memory softkey commands. If an address mapped as "locked" is accessed, a break in the emulation will occur.

The complete 64K-byte I/O space of the μPD70208/70216 must be mapped in 1K-byte blocks to the emulation memory, the target system, or as "locked" memory.

Emulation

The IE-70208/216 executes μPD70208 and μPD70216 user programs in real-time in four different modes: break, trace, count, and time.

(1) In break emulation mode, the user's program is executed in real-time or in single-step until a breakpoint is encountered.

(2) In trace emulation mode, the user's program is executed until the trace buffer is filled.

(3) During the count emulation mode, the emulator counts the number of times a particular trigger point is reached within a given set of conditions.

(4) In time emulation mode, the emulator measures execution time between the specified enable and disable points. The measurable time range is from 0 to 72 minutes (in microseconds).

Once emulation is stopped in either break or trace mode, the trace automatically displays one screen of data, ending on the last instruction executed. In count or time mode, the current count or elapsed time is displayed. At this point, it is possible to display the contents of memory, the general-purpose and special registers, the symbol tables, directories, and other information. All can be displayed individually or by split screen with the trace display. The windows may be scrolled independently.

Prior to the start of emulation, the user can specify whether an external or internal clock will be used for emulation, whether the internal or external ready signal is used, and whether the NMI signal from the target system should be enabled or disabled. If the internal clock is used, it can be set from 2 to 10 MHz in 1-kHz steps.

Break Capabilities

The IE-70208/216 has eight hardware breakpoints. Seven can be set to occur on a real-time event or a non-real-time condition. The remaining one is reserved for setting a real-time address breakpoint in the GO command.

A real-time breakpoint can be set to occur on an address, a data value, a CPU state, or the external probe status. A non-real-time breakpoint can be set to occur after satisfying an address/condition setting a certain number of times (maximum 4096).

Conditions pertaining to the general-purpose registers, memory locations, I/O locations or the external probes can be defined. For non-real-time breakpoints, the user program is executed in real time until it reaches the address, then emulation pauses while the conditions are checked. If the conditions are not satisfied, emulation will continue in this manner until they are met.

To distinguish between the condition occurring at an op-code fetch or at the execution of an instruction, each breakpoint can be tagged with either a nonexecution or execution flag.

Up to 16 software breakpoints can be set plus an additional one in the GO command. To set a software breakpoint, the emulator replaces an instruction in the user's program with a BRK 0 instruction. A break will occur when this instruction is executed, and the user's program will be restored. This capability is not available for program code executing out of ROM.

Trace Capabilities

The trace buffer is 1024 frames by 64 bits wide and sampling is done on every machine cycle. The buffer is filled in a round-robin fashion. The emulator traces the external address and data buses, the CPU and queue status, and the eight external logic probes.

The IE-70208/216 has eight trace specification points. One of these is reserved for setting a trigger point in the GO command. The other seven can be specified as trace trigger, enable, disable, qualify, or check points. Check points are used to display register, memory, or I/O contents each time a certain event or address occurs. The trace buffer can be split into a maximum of 64 partitions to allow tracing of particular segments of the user program (i.e., subroutines).

The trace data may be displayed in one of three modes: machine, disassembly, or jump. In machine display mode, all bus activity is displayed in machine code. In disassembly mode, all instructions are disassembled. In jump mode, only instructions that alter program flow are displayed.

SOFTWARE DESCRIPTION

System Software

The IE-70208/216 is controlled by the MIOS/U proprietary operating system. Command input is simplified by eight function keys (providing a choice of up to 24 softkeys within any menu level). The dynamically reprogrammed softkeys visually prompt with the next valid set of commands. The softkeys are at the bottom of the display screen and correspond to one of the eight function keys on the keyboard. To select a command, press a softkey. The softkeys are automatically relabeled with the next set of commands.

Table 1 lists the utility programs provided with the emulator.

Table 1. IE-70208/216 Utility Programs

Utilities	Function
EMUV4050	IE-70208/216 emulator software
KERMIT	Communication program for file transfer
FILESERV	File management for system disks
EDITOR	Full screen editor
FORMAT	Floppy-disk formatter
PROM	Built-In EPROM programmer control program
TERMINAL	Terminal utility program for file transfer between emulator and another intelligent device
SYMBOL	Symbol table converter; converts non-SROC symbol formats to SROC format.
OBJCONV	Object file converter; converts object files to and from the Motorola SROC format.
TIMESET	Internal battery backed-up clock and calendar setting
DEFINE	Softkey definition
MDEVICE	Disk format specification

Connecting to Host Systems

Host systems may be connected to the IE-70208/216 via the RS-232C connectors at the rear of the machine. Parameters such as baud rates, character length, parity, and number of stop bits are software programmable to suit the system being attached. The KERMIT communications program supplied with the emulator can be used for uploading and downloading files. Currently, NEC provides KERMIT for the VAX® under VMS® and UNIX™ 4.2 BSD or Ultrix™, and the IBM PC, PC/XT®, PC AT®, or compatibles under PC-DOS™ or MS-DOS®.

Files may also be transferred to the emulator via the RS-232C ports by using the TERMINAL utility. The emulator acts as a terminal for data transfer.

Another means of loading files into the IE-70208/216 is available with the Multiple File Handler utility, a program that runs in the emulator itself and is supplied as part of the IE-70208/216 package. The Multiple File Handler allows the emulator to read MS-DOS disks, among others.

Symbolic Debug and Line Assembly/ Disassembly

The IE-70208/216 supports complete symbolic debugging of programs produced by NEC's RA70116 Relocatable Assembler Package and various other third-party software packages, including those from Intel and Microsoft. The symbols can be used as address and data constants in break, trace, and emulation control commands and are displayed during disassembly. A symbolic line assembler is also available to make modifications to existing programs or to enter code from the keyboard.

SPECIFICATIONS

Table 2 gives the electrical, environmental, and physical specifications of the equipment.

Table 2. IE-70208/216 Specifications

AC power	90 to 132 V, 50/60 Hz, 400 W maximum
Temperature	Operating: +5 to +40°C
	Storage: −20 to +50°C
Relative humidity (noncondensing)	Operating: 20 to 80%
	Storage: 10 to 90%
Weight	Main chassis: 40 pounds
	Pod and cables: 4-3/4 pounds
Dimensions (L x W x H)	19.7 x 16.7 x 8.7 inches

DOCUMENTATION

The following manuals are supplied with the in-circuit emulator. Additional copies may be obtained from NEC Electronics Inc.

- IE-70208/216 In-Circuit Emulator User's Manual
- IE-70XXX-A Hardware User's Manual
- IE-70XXX-A Software Utilities User's Manual

VAX and VMS are registered trademarks of Digital Equipment Corporation.
Ultrix is a trademark of Digital Equipment Corporation.
UNIX is a trademark of AT&T.
PC/XT, PC AT, and PC-DOS are registered trademarks of International Business Machines Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.

NEC

NEC Electronics Inc.

IE-70320 In-Circuit Emulator for μPD70320/70322 (V25) Microcomputers

Description

The IE-70320 is a portable, stand-alone, in-circuit emulator that provides hardware emulation and software debug capabilities for the NEC μPD70320/70322 (V25™) 16-bit, single-chip microcomputers.

Real-time and single-step emulation, coupled with sophisticated memory mapping, symbolic debugging, macrofile command facilities, and user-programmable breakpoints and trace qualifiers create a powerful development environment.

Command entry is simplified by eight dynamically reprogrammed function keys, called softkeys, that visually prompt a user with the next level of commands. User programs can be uploaded/downloaded from a variety of host systems by a serial link or they can be loaded directly from an MS-DOS® disk.

Features

- Portable stand-alone in-circuit emulator:
  - 9.5-inch amber CRT display
  - Two 5-inch, 640K-byte floppy-disk drives
  - ASCII keyboard with eight function keys
  - EPROM programmer: 2732, 2764, 27128, 27256, 27512
  - Can be converted to support NEC's V20®, V30®, V33™, V35™, V40™, V50™, V53™, and V60™ microprocessors
- Precise real-time and single-step emulation
  - Programmable internal clock: 1 to 16 MHz in 1- kHz steps
  - Up to 16 MHz external TTL clock
- Memory and I/O space mappable in 4K-byte blocks
- 32K bytes of memory for internal ROM emulation
- 124K bytes of memory for prototype memory emulation; expandable to 636K bytes
- Eight user-programmable hardware breakpoints
  - Real-time break on address, data, CPU status, or external probes
  - Break on pass count and register, memory, or I/O values
  - Selectable as execution or nonexecution
- 16 user-programmable software breakpoints
- Trace buffer: machine cycle, mnemonic, and jump trace display
  - 2047 frames by 108 bits
  - Programmable trigger point and trace qualifiers
- Eight optional probes for tracing of target systems signals
- Full symbolic debug capabilities
- Symbolic line assembler and disassembler
- Macrofile command capability
- Dual window display in emulation mode
- Softkey and menu-driven user input

V20 and V30 are registered trademarks of NEC Corporation.
V25, V33, V35, V40, V50, V53, and V60 are trademarks of NEC Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.

Ordering Information

Part Number	Package
IE-70320-A008	In-circuit emulator for μPD70320/70322
IE-70320-RTOS	μPD79011 RTOS System Software for IE-70320
EP-70320L	μPD70320/70322 84-pin PLCC emulation probe
IE-70000-2957	Optional 512K-byte expansion emulation memory
IE-70000-2954	Optional external logic probes
EP-70320GJ	Optional 94-pin plastic QFP package probe adapter for use with EP-70320L

Hardware

The IE-70320 (figure 1) consists of a system chassis with a detachable ASCII keyboard and an emulation pod unit. The chassis houses a 9.5-inch amber CRT, two 5-1/4 inch 640K-byte floppy-disk drives, an EPROM programmer, card cage, power supply, and five control boards. The boards are main CPU, expansion system memory, emulation control I and II, and trace emulation memory.

The main CPU board contains a supervising CPU, 512K bytes of system memory, and the peripheral interfaces. The expansion system memory board provides an additional 512K bytes of system memory.

The two emulation control boards control memory mapping, event detection, and the break and emulation CPU status circuitry. The trace emulation board contains a trace buffer and 124K bytes of external emulation memory. The optional IE-70000-2957, a 512K-byte expansion emulation memory board, may be installed to increase the external emulation memory to 636K bytes.

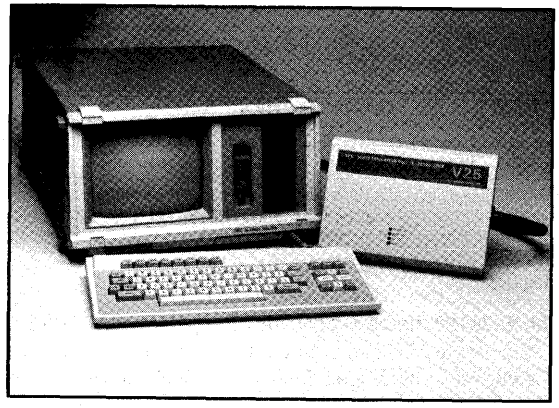
The emulation pod unit houses the μPD70329 EVACHIP used to emulate the μPD70320 or μPD70322, the internal ROM emulation memory, the high-speed buffers, and the clock selection logic. This unit can be connected to the target system by the EP-70320L, an emulation probe for the 84-pin PLCC. For the 94-pin plastic QFP package, an EP-70320GJ probe adapter is also needed.

The IE-70320 supports the following external interfaces: two RS-232C serial ports, one Centronics parallel printer port, and one RGB video output.

An optional external logic probe unit (IE-70000-2954) is also available . The eight probes contained in the unit allow signals on the target system to be used in the break and trace functions.

The emulator can be converted to support NEC's V20, V30, V33, V35, V40, V50, V53, or V60 CMOS microprocessors by exchanging the appropriate control boards and the emulation pod unit. .

Figure 1. IE-70320 System Configuration



Memory and I/O Mapping Capabilities

The IE-70320 contains two kinds of emulation memory: 32K bytes of high-speed RAM that can be accessed in one clock cycle per byte for emulating the internal ROM of the μPD70322, and 124K bytes (expandable to 636K bytes) of two-cycle RAM (0 wait states) for emulating external RAM or ROM.

The complete 1M-byte memory space of the μPD70320/μPD70322 must be mapped into one of the following categories:

INTROM	Internal μPD70322 ROM emulation memory (0, 8, 16, or 32K bytes selectable)
ROM	External ROM emulation memory (read only)
RAM	External RAM emulation memory (read/write)
Target	Memory resident in target system (read/write)
Locked	Access inhibited memory (remaining unmapped addresses)

All memory mapping except INtrom is executed in 4K-byte blocks using the CONFIGure and MEMory softkey commands. If an address that has been mapped as "locked" is accessed, a break in emulation will occur.

The complete 64K-byte input/output space of the μPD70320 or μPD70322 must be mapped in 4K-byte blocks either to the RAM emulation memory, to the target system, or as "locked" memory.

Emulation

The IE-70320 executes μPD70320 and μPD70322 user programs in real time in four different modes: break, trace, count, and time.

(1) In break emulation mode, the program is run in real time or in single step until a breakpoint is encountered.

(2) In trace emulation mode, the program is executed until the trace buffer is filled.

(3) In count emulation mode, the emulator counts the number of times a particular trigger point is reached within a given set of conditions.

(4) In time emulation mode, the emulator times execution between the specified enable and disable points. The measurable time range is from 0 to 72 minutes (in microseconds).

Once emulation is stopped in either break or trace mode, the trace automatically displays one screen of data, ending on the last instruction executed. In count or time mode, the current count or elapsed time is displayed. At this point, it is possible to display the contents of memory, the general-purpose and special registers, the symbol tables, directories, and other information. All can be displayed individually or by split screen with the trace display. The windows may be scrolled independently.

Prior to the start of emulation, the user can specify the internal ROM size (if any), whether an external or internal clock will be used for emulation, and whether the NMI, READY, and HOLD signals from the target system should be enabled or disabled. If the internal clock is used, it can be set from 1 to 16 MHz in 1-kHz steps.

Break Capabilities

The IE-70320 has eight hardware breakpoints. Seven can be set to occur on a real-time event or a non-real-time condition. The remaining one is reserved for setting a real-time address breakpoint in the GO command.

A real-time breakpoint can be set to occur on an address, a data value, a CPU state and an external probe status. A non-real-time breakpoint can be set to occur after an address/condition setting has been satisfied for a certain number of times (maximum 4096).

Conditions pertaining to the general-purpose registers, memory locations, input/output locations, or external probes can be defined. For non-real-time breakpoints, the user program is executed in real time until it reaches the break address. Emulation stops while the conditions are checked. If the conditions are not satisfied, emulation will continue in this manner until they are met.

To distinguish between an address condition occurring at any memory read/write access or the execution of an instruction, each breakpoint can be tagged with either a nonexecution or execution flag.

Up to 16 software breakpoints can be set plus an additional one in the GO command. To set a software breakpoint, the emulator replaces an instruction in the user's program with a BRK 0 instruction. A break will occur when this instruction is executed, and the user's program will be restored. This capability is not available for program code executing out of ROM.

Trace Capabilities

The trace buffer is 2047 frames by 108 bits wide and sampling is done on every machine cycle. The buffer is filled in a round-robin fashion. The emulator traces the external address and data buses, the internal ROM address and data buses, the CPU and queue status, the DMAAK0/DMAAK1 pins, and the eight external logic probes.

The IE-70320 has eight trace specification points. One of these is reserved for setting a trigger point in the GO command. The other seven can be specified as trace trigger, enable, disable, qualify, or check points. Check points are used to display the register, memory, or input/output contents each time a certain event or address occurs. The trace buffer can be split into a maximum of 128 partitions to allow tracing of particular segments of the user program (i.e., subroutines).

The trace data may be displayed in one of three modes: machine, disassembly, or jump. In machine display mode, all bus activity is displayed in machine code. In disassembly mode, all instructions are disassembled. In jump mode, only instructions that alter program flow are displayed.

IE-70320-RTOS System Software

The optional IE-70320-RTOS system software allows the IE-70320 to be used for hardware emulation and software debugging for the μPD79011, a V25 16-bit, single-chip microcomputer with an on-board real-time operating system (RTOS). When using the IE-70320-RTOS system software, the RTOS object code is loaded into the 16K bytes of internal ROM emulation memory whenever the IE-70320 is powered up or the CAncel command is executed.

In addition, the IE-70320-RTOS system software adds the following commands to the IE-70320.

Mem/reg SYstime	Sets system time of the RTOS.
Display TStat	Displays system time, task status, number of unused memory blocks, segment value of all messages queued in the TCB, start address of the initialization routine, and interrupt return address in the TCB for a specified task.
Display MAilbox	Displays status of specified mailbox.
Display SEmaph 4	Displays number of tasks waiting for specified semaphore and remaining number of free resources.
Display TMap	Displays a list of all tasks currently being managed by RTOS and their state.

6g

System Software

The IE-70320 is controlled by the MIOS/U proprietary operating system. Command input is simplified by eight function keys (providing a choice of up to 24 softkeys within any menu level). The dynamically reprogrammed softkeys visually prompt the user with the next valid set of commands. The softkeys are at the bottom of the display and correspond to the eight function keys. To select a command, the desired softkey is entered, and the softkeys are automatically relabeled with the next set of commands.

Table 1 shows some of the utility programs provided with the emulator.

Table 1. IE-70320 Utility Programs

Utility	Function
EMUV25	IE-70320 emulator software
KERMIT	Communication program for file transfer
FILESERV	File management for system disks
EDITOR	Full screen editor
FORMAT	Floppy-disk formatter
PROM	Built-In EPROM programmer control program
TERMINAL	Terminal utility program for file transfer between emulator and another intelligent device
SYMBOL	Symbol Table Converter: converts non-SROC symbol formats to SROC format.
OBJCONV	Object File Converter: converts object files to and from the Motorola SROC format.
TIMESET	Internal battery backed-up clock and calendar setting
DEFINE	Softkey definition
MDEVICE	Disk format specification

Connecting to Host Systems

Host systems may be connected to the IE-70320 by the RS-232C connectors at the rear of the machine. Parameters such as baud rates, character length, parity, and number of stop bits are software programmable to suit the system being attached. The KERMIT communications program is supplied with the emulator and can be used for uploading and downloading files. NEC currently provides KERMIT for the VAX® under VMS® and UNIX™ 4.2 BSD or Ultrix®, the IBM PC, PC/XT®, IBM PC AT®, or compatibles under PC-DOS® or MS-DOS.

VAX, VMS, and Ultrix are registered trademarks of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories
PC/XT, PC AT, and PC-DOS are registered trademarks of International Business Machines Corporation.

Files may also be transferred to the emulator via the RS-232C ports by using the TERMINAL utility. The emulator acts as a terminal for data transfer.

Another means of loading files into the IE-70320 is available with the Multiple File Handler utility, a program that runs in the emulator itself and which is also supplied as part of the IE-70320 package. The Multiple File Handler allows the emulator to read MS-DOS disks, among others.

Symbolic Debug and Line Assembly/ Disassembly

The IE-70320 supports complete symbolic debugging of programs produced by NEC's RA70320 Relocatable Assembler package and various other third-party software packages, including those of Intel and Microsoft. The symbols can be used as address and data constants in break, trace, and emulation control commands and are displayed during disassembly. A symbolic line assembler is also available to make modifications to existing programs or to enter code from the keyboard.

Specifications

Table 2 gives the electrical, environmental, and physical specifications of the equipment.

Table 2. IE-70320 Specifications

Ac power	90 to 132 V, 50/60 Hz, 400 W maximum
Temperature	Operating: +5 to +40°C
	Storage: –20 to +50°C
Relative humidity (noncondensing)	Operating: 20 to 80%
	Storage: 10 to 90%
Weight	Main chassis: 40 pounds
	Pod and cables: 4-3/4 pounds
Dimensions (L x W x H)	19.7 x 16.7 x 8.7 inches

Documentation

The following manuals are supplied with the in-circuit emulator. Additional copies may be obtained from NEC Electronics Inc.

- IE-70320 In-Circuit Emulator User's Manual
- IE-70XXX-A Hardware User's Manual
- IE-70XXX-A Software Utilities User's Manual
- IE-70320-RTOS μPD79011 RTOS System Software User's Manual

NEC

NEC Electronics Inc.

IE-70330 In-Circuit Emulator for μPD70330/70332 (V35) Microcomputers

Description

The IE-70330 is a portable, stand-alone, in-circuit emulator that provides hardware emulation and software debug capabilities for the μPD70330/70332 (V35™) 16-bit, single-chip microcomputers.

Real-time and single-step emulation, coupled with sophisticated memory mapping, symbolic debugging, macrofile command facilities, and user-programmable breakpoints and trace qualifiers, create a powerful development environment.

Command entry is simplified by eight dynamically reprogrammed function keys, called softkeys, that visually prompt a user with the next level of commands. User programs can be uploaded/downloaded from a variety of host systems by a serial link, or they can be loaded directly from an MS-DOS® disk.

Features

- Portable, stand-alone, in-circuit emulator
  - 9.5-inch amber CRT display
  - Two 5-inch, 640K-byte floppy-disk drives
  - ASCII keyboard with eight function keys
  - EPROM programmer: 2732, 2764, 27128, 27256, 27512
  - Supports NEC's V20®, V30®, V33™, V25™, V40™, V50™, V53™, and V60™ microprocessors
- Precise real-time and single-step emulation
  - Programmable internal clock: 1 to 16 MHz in 1-kHz steps
  - Up to 16-MHz external TTL clock
- Memory and I/O space mappable in 4K-byte blocks
- 32K bytes of memory for internal ROM emulation
- 124K bytes of memory for prototype memory emulation; expandable to 636K bytes
- Eight user-programmable hardware breakpoints
  - Real-time break on address, data, CPU status, or external probes
  - Break on pass count and register, memory, or I/O values
  - Selectable as execution or nonexecution
- 16 user-programmable software breakpoints
- Trace buffer: machine cycle, mnemonic, and jump trace display
  - 2047 frames by 108 bits
  - Programmable trigger point and trace qualifiers
- Eight optional probes for tracing target system signals
- Full symbolic debug capabilities
- Symbolic line assembler and disassembler
- Macrofile command capability
- Dual window display in emulation mode
- Softkey and menu-driven user input

Ordering Information

Part Number	Description
IE-70330-A008	In-circuit emulator for μPD70330/70332 (V35)
IE-70330-RTOS	μPD79021 RTOS system software for IE-70330-A008
EP-70320L	μPD70320/70322 84-pin PLCC emulation probe
IE-70000-2957	Optional 512K-byte expansion emulation memory
IE-70000-2954	Optional external logic probes
EP-70320GJ	Optional 94-pin plastic QFP package probe adapter for use with EP-70320L

Hardware

The IE-70330 (figure 1) consists of a system chassis with a detachable ASCII keyboard and an emulation pod unit. The chassis houses a 9.5-inch amber CRT, two 5-1/4 inch 640K-byte floppy-disk drives, an EPROM programmer, card cage, power supply, and five control boards. The boards are main CPU, expansion system memory, emulation control I and II, and trace emulation memory.

The main CPU board contains a supervising CPU, 512K bytes of system memory, and the peripheral interfaces. The expansion system memory board provides an additional 512K bytes of system memory.

V20 and V30 are registered trademarks of NEC Corporation.
V25, V33, V35, V40, V50, V53, and V60 are trademarks of NEC Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.

The two emulation control boards control memory mapping, event detection, and the break and emulation CPU status circuitry. The trace emulation board contains a trace buffer and 124K bytes of external emulation memory. The optional IE-70000-2957, a 512K-byte expansion emulator memory board, may be installed to increase the external emulator memory to 636K bytes.

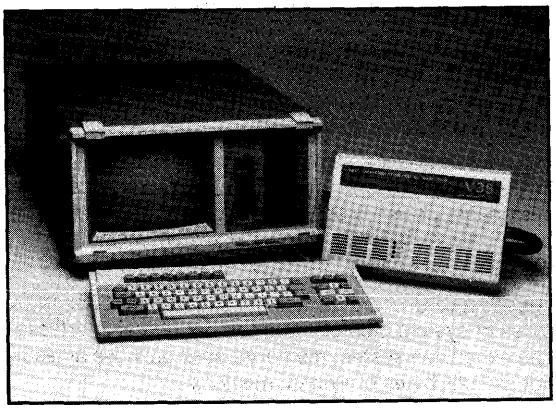
The emulation pod unit houses the μPD70339 EVACHIP used to emulate the μPD70330 or μPD70332, the internal ROM emulation memory, the high-speed buffers, and the clock selection logic. This unit can be connected to the target system by the EP-70320L, an emulation probe for the 84-pin PLCC. For the 94-pin plastic QFP, an EP-70320GJ probe adapter is also needed.

The IE-70330 supports the following external interfaces: two RS-232C serial ports, one Centronics parallel printer port, and one RGB video output.

An optional external logic probe unit (IE-70000-2954) is also available. The eight probes contained in the unit allow signals on the target system to be used in the break and trace functions.

The emulator can be converted to support NEC's V20, V25, V30, V33, V40, V50, V53, and V60 CMOS microprocessors by exchanging the appropriate control boards and the emulation pod unit.

Figure 1. IE-70330 System Configuration



Memory and I/O Mapping Capabilities

The IE-70330 contains two kinds of emulation memory: 32K bytes of high-speed RAM that can be accessed in one clock cycle per byte for emulating the internal ROM of the μPD70332, and 124K bytes (expandable to 636K bytes) of two-cycle RAM (0 wait states) for emulating external RAM or ROM.

The complete 1M-byte memory space of the μPD70330/70332 must be mapped into one of the following categories:

INTROM	Internal μPD70332 ROM emulation memory (0, 8, 16, 32K bytes selectable)
ROM	External ROM emulation memory (read only)
RAM	External RAM emulation memory (read/write)
Target	Memory resident in target system (read/write)
Locked	Access inhibited memory (remaining unmapped addresses)

All memory mapping except INTROM is executed in 4K-byte blocks using the Configure and Memory softkey commands. If an address that has been mapped as "locked" is accessed, a break in emulation will occur.

The complete 64K-byte input/output space of the μPD70330 or μPD70332 must be mapped in 4K-byte blocks either to the RAM emulation memory, to the target system, or as "locked" memory.

Emulation

The IE-70330 executes μPD70330 and μPD70332 user programs in real time in four different modes: break, trace, count, and time.

(1) In break emulation mode, the program is run in real time or in single step until a breakpoint is encountered.

(2) In trace emulation mode, the program is executed until the trace buffer is filled.

(3) In count emulation mode, the emulator counts the number of times a particular trigger point is reached within a given set of conditions.

(4) In time emulation mode, the emulator times execution between the specified enable and disable points. The measurable time range is from 0 to 72 minutes (in microseconds).

Once emulation is stopped in either break or trace mode, the trace automatically displays one screen of data, ending on the last instruction executed. In count or time mode, the current count or elapsed time is displayed. At this point, it is possible to display the contents of memory, the general-purpose and special registers, the symbol tables, directories, and other information. All can be displayed individually or by split screen with the trace display. The windows may be scrolled independently.

Prior to the start of emulation, the user can specify the internal ROM size (if any), whether an external or internal clock will be used for emulation, and whether the NMI, READY, and HOLD signals from the target system should be enabled or disabled. If the internal clock is used, it can be set from 1 to 16 MHz in 1-KHz steps.

Break Capabilities

The IE-70330 has eight hardware breakpoints. Seven can be set to occur on a real-time event or a non-real-time condition. The remaining one is reserved for setting a real-time address breakpoint in the GO command.

A real-time breakpoint can be set to occur on an address, a data value, a CPU state, or an external probe status. A non-real-time breakpoint can be set to occur after an address/condition setting has been satisfied for a certain number of times (maximum 4096).

Conditions pertaining to the general-purpose registers, memory locations, input/output locations, or external probes can be defined. For non-real-time breakpoints, the user program is executed in real time until it reaches the break address. Emulation stops while the conditions are checked. If the conditions are not satisfied, emulation will continue in this manner until they are met.

To distinguish between an address condition occurring at any memory read/write access or the execution of an instruction, each breakpoint can be tagged with either a nonexecution or execution flag.

Up to 16 software breakpoints can be set plus an additional one in the GO command. To set a software breakpoint, the emulator replaces an instruction in the user's program with a BRK 0 instruction. A break will occur when this instruction is executed, and the user's program will be restored. This capability is not available for program code executing out of ROM.

Trace Capabilities

The trace buffer is 2047 frames by 108 bits wide and sampling is done on every machine cycle. The buffer is filled in a round-robin fashion. The emulator traces the external address and data buses, the internal ROM address and data buses, the CPU and queue status, the DMAAK0/1 pins, and the eight external logic probes.

The IE-70330 has eight trace specification points. One of these is reserved for setting a trigger point in the GO command. The other seven can be specified as trace trigger, enable, disable, qualify, or check points. Check points are used to display the register, memory, or input/output contents each time a certain event or address occurs. The trace buffer can be split into a maximum of 128 partitions to allow tracing of particular segments of the user program (i.e., subroutines).

The trace data may be displayed in one of three modes: machine, disassembly, or jump. In machine display mode, all bus activity is displayed in machine code. In disassembly mode, all instructions are disassembled. In jump mode, only instructions that alter program flow are displayed.

System Software

The IE-70330 is controlled by the MIOS/U proprietary operating system. Command input is simplified by eight function keys (providing a choice of up to 24 softkeys within any menu level). The dynamically reprogrammed softkeys visually prompt the user with the next valid set of commands. The softkeys are at the bottom of the display screen and correspond to the eight function keys on the keyboard. To select a command, the desired softkey is entered, and the softkeys are automatically relabeled with the next set of commands.

Table 1 lists some of the utility programs provided with the emulator.

IE-70330-RTOS System Software

The optional IE-70330-RTOS system software allows the IE-70330 to be used for hardware emulation and software debugging for the μPD79021, a V35 16-bit single-chip microcomputer with an on-board real-time operating system (RTOS). The RTOS object code is loaded into the 16K bytes of internal ROM emulation memory whenever the IE-70330 is powered up or the CAncel command is executed.

In addition, the IE-70330-RTOS system software adds the commands in table 2 to the IE-70330.

Table 1. IE-70330 Utility Programs

Utility	Function
EMUV35	IE-70330 emulator software
KERMIT	Communication program for file transfer
FILESERV	File management for system disks
EDITOR	Full screen editor
FORMAT	Floppy-disk formatter
PROM	Built-in EPROM programmer control program
TERMINAL	Terminal utility program for file transfer between emulator and another intelligent device
SYMBOL	Symbol table converter; converts non-SROC symbol formats to SROC format
OBJCONV	Object file converter; converts object files to and from the Motorola SROC format
TIMESET	Internal battery backed-up clock and calendar setting
DEFINE	Softkey definition
MDEVICE	Disk format specification

Table 2. IE-70330-RTOS Commands

Command	Description
Mem/reg SYstime	Sets RTOS system time.
Display TStat	Displays system time, task status, number of unused memory blocks, segment value of all messages queued in the TCB, start address of initialization routine, and interrupt return address in the TCB for a specified task.
Display MAilbox	Displays status of specified mailbox.
Display SEmaph	Displays number of tasks waiting for specified semaphore and remaining number of free resources.
Display TMap	Displays a list of all tasks currently being managed by RTOS and their current state.

Connecting to Host Systems

Host systems may be connected to the IE-70330 through RS-232C connectors at the rear of the machine. Parameters such as baud rates, character length, parity, and number of stop bits are software programmable to suit the system being attached. The KERMIT communications program is supplied with the emulator and can be used for uploading and downloading files. NEC currently provides KERMIT for the VAX® under VMS® and UNIX™ 4.2BSD or Ultrix®, the IBM PC, PC/XT®, IBM PC AT®, or compatibles under PC-DOS® or MS-DOS.

VAX, VMS, and Ultrix are registered trademarks of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories.
PC/XT, PC AT, and PC-DOS are registered trademarks of International Business Machines Corporation.

Files may also be transferred to the emulator via the RS-232C ports by using the TERMINAL utility. The emulator acts as a terminal for data transfer.

Another means of loading files into the IE-70330 is available with the Multiple File Handler utility, a program that runs in the emulator itself and is supplied as part of the IE-70330 package. The Multiple File Handler allows the emulator to read MS-DOS disks, among others.

Symbolic Debug and Line Assembly/ Disassembly

The IE-70330 supports complete symbolic debugging of programs produced by NEC's RA70320 Relocatable Assembler package and various other third-party software packages, including those of Intel and Microsoft. The symbols can be used as address and data constants in break, trace, and emulation control commands and are displayed during disassembly. A symbolic line assembler is also available to make modifications to existing programs or to enter code from the keyboard.

Specifications

Table 3 gives the electrical, environmental, and physical specifications of the equipment.

Table 3. IE-70330 Specifications

Ac power	90 to 132 V, 50/60 Hz, 400 W maximum
Temperature	Operating: +5 to +40°C
	Storage: −20 to +50°C
Relative humidity (noncondensing)	Operating: 20 to 80%
	Storage: 10 to 90%
Weight	Main chassis: 40 pounds
	Pod and cables: 4-3/4 pounds
Dimensions (L x W x H)	19.7 x 16.7 x 8.7 inches

Documentation

The following manuals are supplied with the in-circuit emulator. Additional copies may be obtained from NEC Electronics Inc.

- IE-70330 In-Circuit Emulator User's Manual
- IE-70XXX-A Hardware User's Manual
- IE-70XXX-A Software Utilities User's Manual
- IE-70330-RTOS, μPD79021 RTOS System Software User's Manual

NEC
NEC Electronics Inc.

RA70116
Relocatable Assembler Package for V20-V50 Microprocessors

Description

The RA70116 Relocatable Assembler package converts symbolic source code for the μPD70108 (V20®), μPD70116 (V30®), μPD70208 (V40™), and μPD70216 (V50™) microprocessors into executable absolute address object code. The package consists of four separate programs: an assembler (RA70116), a linker (LK70116), a hexadecimal format object code converter (OC70116), and a librarian (LB70116).

RA70116 translates a symbolic source module into a relocatable object module. This symbolic source module can contain both V20-V50 microprocessor instructions and Intel 8087 Floating-Point Arithmetic Coprocessor instructions. The assembler verifies that each instruction assembled is valid and produces a listing file and a relocatable object module.

LK70116 combines relocatable object modules and absolute load modules and converts them into an absolute load module. OC70116 converts an absolute object module or an absolute load module to an expanded hexadecimal (7-bit ASCII) object file.

LB70116 allows commonly used relocatable object modules to be stored in one file and linked into multiple programs, greatly increasing programming efficiency. When the input of the linker contains a library file, the linker first extracts only those modules required to resolve external references from the file and relocates and links them.

Features

- Absolute address object code output
- Macro and code macro capability
- User-selectable and directable output files
- Extensive error reporting
- Powerful Librarian
- Runs under the following operating systems:
  - MS-DOS®
  - VAX/VMS® and VAX/UNIX® 4.2BSD or Ultrix®

V20 and V30 are registered trademarks of NEC Corporation.
V40 and V50 are trademarks of NEC Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.
VAX, VMS, and Ultrix are registered trademarks of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories.

Ordering Information

Part Number	Package
RA70116-D52	MS-DOS, 5-1/4" double-density floppy-diskette
RA70116-VVT1	VAX/VMS, 9-track 1600-BPI magnetic tape
RA70116-VXT1	VAX/UNIX 4.2BSD or Ultrix, 9-track 1600-BPI magnetic tape

SOFTWARE DESCRIPTION

Program Syntax

An RA70116 source module consists of a series of code, data, or stack segments. Each segment consists of statements composed of up to four fields: symbol, mnemonic, operand, and comment.

The symbol field may contain a label whose value is the instruction or data address, or a name that represents an instruction address, data address, or constant. The mnemonic field may contain an instruction or assembler directive. The operand field contains the data or expression for the specified instruction or directive. Explanations of statements may be inserted in the comment field.

Character constants are translated into 7-bit ASCII codes. Numeric constants may be specified as binary, octal, decimal, or hexadecimal. Arithmetic expressions may include the operators +, –, *, /, MOD, OR, AND, NOT, XOR, EQ, NE, LT, LE, GT, GE, SHR, SHL, LOW, HIGH, PTR, SHORT, THIS, SEG, OFFSET, SMSIZE, GRSIZE, SMOFFSET, GROFFSET, TYPE, LENGTH, SIZE, MASK, WIDTH, ( ), [ ], period (.), colon (:), < >.

Macro and Code Macro Capability

RA70116 allows the definition of macrocode sequences with parameters, LOCAL symbols, and special repeated code sequences. The macrocode sequence is different from a subroutine call. That is, the invocation of a macro in the source code results in the direct replacement of a macro call with the defined code sequence.

RA70116 also allows the definition of code macros to give the user the capability of defining a new instruction (mnemonic). Although an instruction definition could also be defined using the ordinary macro facility, code macros specify the allowable operand types for the new instructions whereas ordinary macros cannot.

Assembler Directives

Assembler directives give instructions to the assembler but are not translated into machine code during assembly. Basic assembler directives include those for storage definition and allocation (DB, DW, DD, DBS, DWS, DDS, STRUC/ENDS, RECORD); symbol control (EQU, LABEL, PURGE); and location counter control (ORG, EVEN, ALIGN).

Program control directives include those for segment definition and control (SEGMENT/ENDS, PROC/ENDP, ASSUME, GROUP, END); linkage (NAME, PUBLIC, EXTRN); and PARITY.

The relocation types for SEGMENT/ENDS directives are specified in the operand column and include BYTE, WORD, PARA, PAGE, and INPAGE. The combination types of PUBLIC, COMMON, AT, STACK, and MEMORY, which are also specified in the operand column, define the means of linking segments and groups of the same name.

Assembler Controls

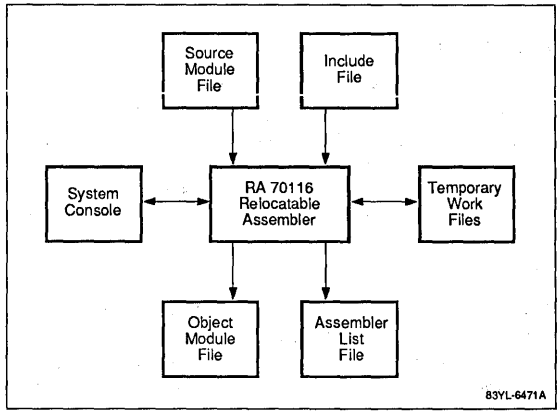
Two types of assembler controls are available for the RA70116.

- Basic controls (specified in the assembler command line)
  - File specification
  - Output file selection
  - Output file destination
  - Listing format controls
  - Debug information output selection
  - Symbol case selection
  - Macro processing selection
- General controls (specified in the source program)
  - Inclusion of other source files
  - Page eject
  - Generation/suppression of listing
  - Generation/suppression of macro listings
  - Listing titles

A list file may contain the complete assembly listing, or it may contain only lines with errors and a symbol or cross-reference table. The symbol table lists all defined symbols in alphabetical order and also shows their types, attributes, and the values initially assigned to them. The cross-reference table contains all defined symbols as well as the numbers of all statements referring to them.

The object file contains the relocatable object module. The format of this module is an NEC proprietary relocatable object module format. Figure 1 is a functional diagram of the assembler.

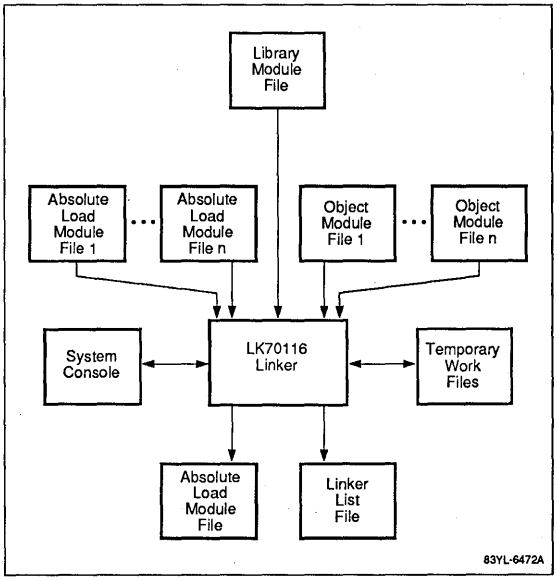
Figure 1. Relocatable Assembler Functional Diagram



Linker

The LK70116 linker combines relocatable object modules and absolute load modules and produces one absolute load module. See figure 2. The controls for LK70116 may be specified in either the command line or in a parameter file. In addition to being able to specify the module name and the starting address and order for code/data/stack segments, you can also protect areas of memory from being assigned. Furthermore, you can instruct the program to create a list file containing a link map, a local symbol table, or a public symbol table. The absolute load module contains symbol information for the symbolic debugger and absolute object code.

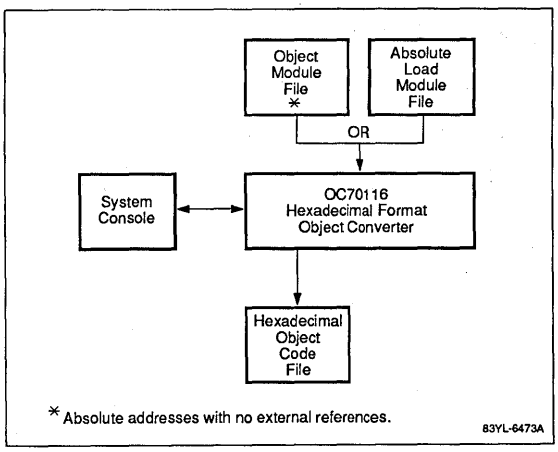
Figure 2. Linker Functional Diagram



Hexadecimal Object Code Converter

The OC70116 object code converter translates an absolute load module file into an expanded hexadecimal format (7-bit ASCII) file that may be downloaded to a PROM programmer. Addresses may be specified as being output in the order in which they were input or in ascending order. Figure 3 is a functional diagram of the hexadecimal object code converter.

Figure 3. Hexadecimal Object Code Converter Functional Diagram



Librarian

The LB70116 librarian creates and maintains files containing relocatable object modules. The program reduces the number of files that need to be linked together by allowing several modules to be kept in a single file. It also provides an easy way to link frequently used modules into programs. Modules may be added to, deleted from, or replaced within a library file.

Operating Environment

The RA70116 package can be supplied to run under several different operating systems. One version is for an MS-DOS system with one or more disk drives and at least 512K bytes of system memory. Other versions run on a Digital Equipment Corporation VAX computer with UNIX 4.2 BSD or Ultrix, or VMS (Version 4.1 or later) operating systems.

Downloading Files Into the Emulator

Absolute load modules produced by the RA70116 package for the V40 and V50 can be debugged using the NEC IE-70208 (V40) or IE-70216 (V50) stand-alone in-circuit emulator. Communication between these emulators and the host system is through an RS-232C serial line using the KERMIT communication protocol developed at Columbia University. With the appropriate version of the KERMIT Communication Program running on both the emulator and host system, absolute load modules or hexadecimal object code files may be transferred between machines.

A version of the KERMIT Communication Program is supplied with the IE-70208 and IE-70216. Versions of KERMIT run on the IBM PC, PC/XT®, PC AT®, or compatibles under MS-DOS, and the DEC VAX under VMS, UNIX 4.2BSD or Ultrix. An appropriate version is provided with each relocatable assembler package at no extra charge. Versions of KERMIT for other host systems are available directly from Columbia University.

A second means of loading files into the emulator is also available in the Multiple File Handler, a utility program that runs in the emulator and is supplied with the IE-70208 and IE-70216. The Multiple File Handler allows the emulator to read MS-DOS formatted disks, among others.

PC/XT, PC AT, and PC-DOS are registered trademarks of International Business Machines Corporation.

DOCUMENTATION

For further information on source program formats, assembler operation, and actual program examples, refer to the following manuals supplied with the RA70116. Additional copies may be obtained from NEC Electronics inc.

- RA70116 V20-V50 Relocatable Assembler Package Language Manual
- RA70116 V20-V50 Relocatable Assembler Package Operation Manual (MS-DOS)
- RA70116 V20-V50 Relocatable Assembler Package Operation Manual (UNIX)
- RA70116 V20-V50 Relocatable Assembler Package Operation Manual (VMS)

LICENSE AGREEMENT

RA70116 is sold under terms of a license agreement included with purchased copies of the assembler. The accompanying card must be completed and returned to NEC Electronics Inc. to register the license. Software updates are provided to registered users.

RA70136
Relocatable Assembler Package for V33 Microprocessor

Description

The RA70136 Relocatable Assembler package converts symbolic source code for the μPD70136 (V33™) microprocessor into executable absolute address object code. The package consists of five separate programs: an assembler (RA70136), a linker (LK70136), an extended mode locater (EL70136), a hexadecimal format object code converter (OC70136), and a librarian (LB70136).

RA70136 translates a symbolic source module into a relocatable object module. This symbolic source module can contain both V33 microprocessor instructions and NEC μPD71291 Advanced Floating-Point Processor (AFPP) instructions. The assembler verifies that each instruction assembled is valid and produces a listing file and a relocatable object module.

LK70136 combines relocatable object modules and absolute load modules and converts them into an absolute load module. If V33 normal addressing mode is being used, OC70136 is used to convert an absolute object module or an absolute load module to an expanded hexadecimal (7-bit ASCII) object file. If V33 extended addressing mode is being used, the EL70136 converts load modules produced by LK70136 to an extended load module file in extended COFF format.

LB70136 allows commonly used relocatable object modules to be stored in one file and linked into multiple programs, greatly increasing programming efficiency. When the input of the linker contains a library file, the linker first extracts only those modules required to resolve external references from the file and relocates and links them.

Features

- Absolute address object code output
  - In extended hexadecimal format for normal addressing mode
  - In extended COFF format for extended addressing mode
- Macro and code macro capability
- User-selectable and directable output files
- Extensive error reporting
- Powerful Librarian
- Runs under the following operating systems
  - MS-DOS®
  - VAX/VMS® and VAX/UNIX™ 4.2BSD or Ultrix™

V33 is a trademark of NEC Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.
VAX, VMS, and Ultrix are registered trademarks of Digital Equipment Corporation.
UNIX is a trademark of AT&T.

Ordering Information

Part Number	Description
RA70136-D52	MS-DOS, 5-1/4" double-density floppy diskette
RA70136-VVT1	VAX/VMS, 9-track 1600-BPI magnetic tape
RA70136-VXT1	VAX/UNIX 4.2 BSD or Ultrix, 9-track 1600-BPI magnetic tape

SOFTWARE DESCRIPTION

Program Syntax

An RA70136 source module consists of a series of code, data, or stack segments. Each segment consists of statements composed of up to four fields: symbol, mnemonic, operand, and comment.

The symbol field may contain a label, whose value is the instruction or data address, or a name that represents an instruction address, data address, or constant. The mnemonic field may contain an instruction or assembler directive. The operand field contains the data or expression for the specified instruction or directive. Explanations for statements may be inserted in the comment field.

Character constants are translated into 7-bit ASCII codes. Numeric constants may be specified as binary, octal, decimal, or hexadecimal. Arithmetic expressions may include the operators +, –, *, /, MOD, OR, AND, NOT, XOR, EQ, NE, LT, LE, GT, GE, SHR, SHL, LOW, HIGH, PTR, SHORT, THIS, SEG, OFFSET, SMSIZE, GRSIZE, SMOFFSET, GROFFSET, TYPE, LENGTH, SIZE, MASK, WIDTH, ( ), [ ], period (.), colon (:), < >.

Macro and Code Macro Capability

RA70136 allows the definition of macrocode sequences with parameters, LOCAL symbols, and special repeated code sequences. The macrocode sequence is different from a subroutine call. That is, the invocation of a macro in the source code results in the direct replacement of a macro call with the defined code sequence.

RA70136 also allows the definition of code macros to give the user the capability of defining a new instruction (mnemonic). Although an instruction definition could

also be defined using the ordinary macro facility, code macros specify the allowable operand types for the new instructions whereas ordinary macros cannot.

Assembler Directives

Assembler directives give instructions to the assembler but are not translated into machine code during assembly. Basic assembler directives include those for storage definition and allocation (DB, DW, DD, DQ, DS, DL, DBS, DWS, DDS, DQS, DSS, DLS, STRUC/ENDS, RECORD); symbol control (EQU, LABEL, PURGE); and location counter control (ORG, EVEN, ALIGN).

Program control directives include those for segment definition and control (SEGMENT/ENDS, PROC/ENDP, ASSUME, GROUP, END); linkage (NAME, PUBLIC, EXTRN); and PARITY.

The relocation types for SEGMENT/ENDS directives are specified in the operand column and include BYTE, WORD, PARA, PAGE, and INPAGE. The combination types of PUBLIC, COMMON, AT, STACK, and MEMORY, which are also specified in the operand column, define the means of linking segments and groups of the same name.

Assembler Controls

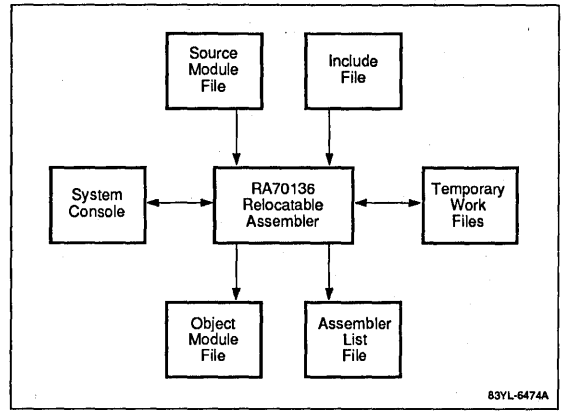
Two types of assembler controls are available for the RA70136.

- Basic controls (specified in the assembler command line)
  - File specification
  - Output file selection
  - Output file destination
  - Listing format controls
  - Debug information output selection
  - Symbol case selection
  - Macroprocessing selection
- General controls (specified in the source program)
  - Inclusion of other source files
  - Page eject
  - Generation/suppression of listing
  - Generation/suppression of macro listings
  - Listing titles

A list file may contain the complete assembly listing, or it may contain only lines with errors and a symbol or cross-reference table. The symbol table lists all defined symbols in alphabetical order and also shows their types, attributes, and the values initially assigned to them. The cross-reference table contains all defined symbols as well as the numbers of all statements referring to them.

The object file contains the relocatable object module. The format of this module is an NEC proprietary relocatable object module format. Figure 1 is a functional diagram of the assembler.

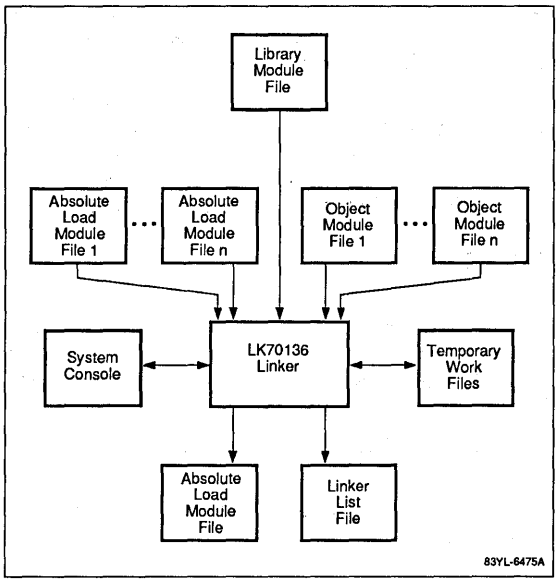
Figure 1. Relocatable Assembler Functional Diagram



Linker

The LK70136 linker combines relocatable object modules and absolute load modules and produces one absolute load module. See figure 2. The controls for LK70136 may be specified in either the command line or in a parameter file. In addition to being able to specify the module name and the starting address and order for code/data/stack segments, you can also protect areas of memory from being assigned. Furthermore, you can instruct the program to create a list file containing a link map, a local symbol table, or a public symbol table. The absolute load module contains symbol information for the symbolic debugger and absolute object code.

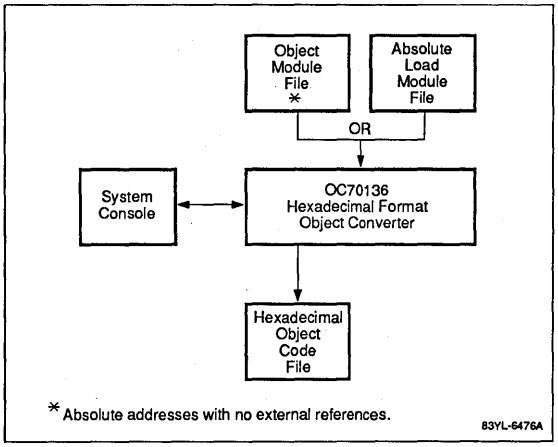
Figure 2. Linker Functional Diagram



Hexadecimal Object Code Converter

The OC70136 object code converter translates an absolute load module file into an expanded hexadecimal format (7-bit ASCII) file that may be downloaded to a PROM programmer. This program is used with the V33 in normal addressing mode (1M-byte address space). Addresses may be specified as being output in the order in which they were input or in ascending order. Figure 3 is a functional diagram of the hexadecimal object code converter.

Figure 3. Hexadecimal Object Code Converter Functional Diagram



Extended Mode Locater

The EL70136 extended mode locater converts multiple load modules produced by LK70136 into one extended load module file in extended COFF format (figure 4). This program is used with the V33 in extended address mode (16M-byte address space). Starting addresses for each load module are specified in the Locate Information file. The name of this file along with the name of the extended load module file and any locater options are included in the command line when EL70136 is invoked. EL70136 can be instructed to create a locate map file and to include debugging information in the extended load module file. To support debugging with the IE-70136, EL70136 also sets initial values for the IE-70136 PGR tables in the extended load module file.

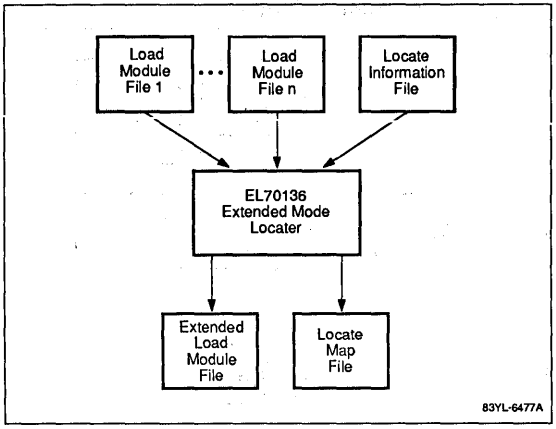
To simplify the task of using the extended addressing mode of the V33, NEC Electronics provides three subroutines with the RA70136 package.

(1) V33_MAP. Maps the μPD70136 Page Registers (PGRs)

(2) V33_BRK. Branches from the normal address mode to the interrupt routine starting address in the extended address mode.

(3) V33_RET. Branches from the extended address mode to the interrupt routine starting address in the normal address mode.

Figure 4. Extended Locater Functional Diagram



Librarian

The LB70136 librarian creates and maintains files containing relocatable object modules. The program reduces the number of files that need to be linked together by allowing several modules to be kept in a single file. It also provides an easy way to link frequently used modules into programs. Modules may be added to, deleted from, or replaced within a library file.

Operating Environment

The RA70136 package can be supplied to run under many different operating systems. One version is for an MS-DOS system with one or more disk drives and at least 512K bytes of system memory. Other versions run on a DEC VAX computer with UNIX 4.2BSD or Ultrix, or VMS (Version 4.1 or later) operating systems.

Downloading Files Into the Emulator

Absolute load modules and extended load modules produced by the RA70136 package for the V33 can be debugged by using the NEC IE-70136 stand-alone in-circuit emulator. Communication between the IE-70136 and the host system is through an RS-232C serial line using the KERMIT communication protocol developed at Columbia University. With the appropriate version of the KERMIT Communication Program running on both the emulator and host system, absolute load modules, extended load modules, or hexadecimal object code files may be transferred between machines.

A version of the KERMIT Communication Program is supplied with the IE-70136.. Versions of KERMIT run on the IBM PC, PC/XT®, PC AT®, or compatibles under MS-DOS, and the DEC VAX under VMS, UNIX 4.2BSD or Ultrix. An appropriate version is provided with each relocatable assembler package at no extra charge. Versions of KERMIT for other host systems are available directly from Columbia University.

A second means of loading files into the emulator is also available in the Multiple File Handler, a utility program that runs in the emulator and is supplied with the IE-70136. The Multiple File Handler allows the emulator to read MS-DOS formatted disks, among others.

DOCUMENTATION

For further information on source program formats, assembler operation, and actual program examples, refer to the following manuals supplied with the RA70136. Additional copies may be obtained from NEC Electronics Inc.

RA70136 V33 Relocatable Assembler Package Language Manual

RA70136 V33 Relocatable Assembler Package Operation Manual.

LICENSE AGREEMENT

RA70136 is sold under terms of a license agreement included with purchased copies of the assembler. The accompanying card must be completed and returned to NEC Electronics Inc. to register the license. Software updates are provided to registered users.

PC/XT and PC AT are registered trademarks of International Business Machines Corporation.

NEC
NEC Electronics Inc.

RA70320
Relocatable Assembler Package for V25/V35 Microcomputers

Description

The RA70320 Relocatable Assembler package converts symbolic source code for the V25™/V35™ family of microprocessors into executable absolute address object code. The package consists of four programs: RA70320 assembler, LK70320 linker, OC70320 hexadecimal object code converter, and LB70320 librarian.

The RA70320 assembler translates a symbolic source module into a relocatable object module. The LK70320 linker combines relocatable object modules and absolute load modules and converts them into one absolute load module. The OC70320 converts an absolute object module or absolute load module to an expanded hexadecimal (7-bit ASCII) object file.

The LB70320 librarian allows commonly used relocatable object modules to be stored in one file and linked into multiple programs, greatly increasing programming efficiency. When the input of the linker contains a library file, the linker first extracts only those modules required to resolve external references from the file and then relocates and links these modules.

Features

- Absolute address object code output
- Macro and code macro capability
- User-selectable and directable output files
- Extensive error reporting
- Powerful librarian
- Multisystem compatibility
  - MS-DOS®
  - VAX®/VMS®
  - VAX/UNIX™ 4.2BSD or Ultrix®

V25 and V35 are trademarks of NEC Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.
VAX, VMS, and Ultrix are registered trademarks of Digital Equipment Corp.
UNIX is a trademark of AT&T.

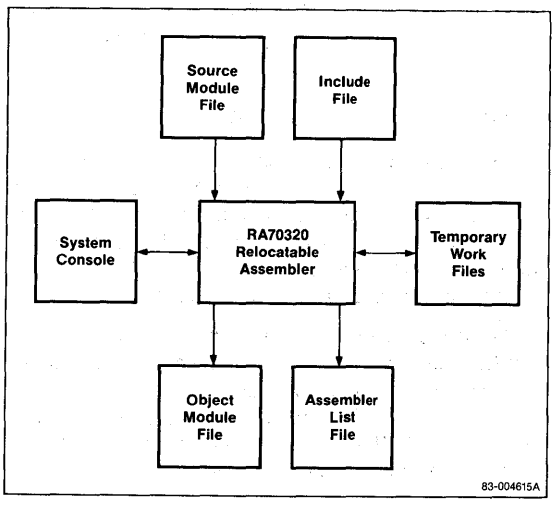
Ordering Information

Part Number	Description
RA70320-D52	MS-DOS; 5-1/4" double-density floppy diskette
VVT1	VAX/VMS; 9-track 1600 BPI magnetic tape
VXT1	VAX/UNIX 4.2BSD or Ultrix; 9-track 1600 BPI magnetic tape

Assembler

The RA70320 assembler program translates a symbolic source module into a relocatable object module by first verifying that each instruction assembled is valid for the target microprocessor and then producing a list file and a relocatable object module (figure 1).

Figure 1. Relocatable Assembler Functional Diagram



6k

Program Syntax

An RA70320 source module consists of a series of code, data, or stack segments. Each segment contains lines composed of up to four fields: symbol, mnemonic, operand, and comment.

The symbol field may contain either a label—whose value is an instruction or data address—or a name that represents an instruction address, data address, or constant. The mnemonic field may contain an instruction or assembler directive. The operand field contains the data or expression for the specified instruction or directive. Explanations for the statements may be inserted into the comment field.

Character constants are translated into 7-bit ASCII codes. Numeric constants may be specified as binary, octal, decimal, or hexadecimal. Arithmetic expressions may include the operators +, –, *, /, MOD, OR, AND, NOT, XOR, EQ, NE, LT, LE, GT, GE, SHR, SHL, LOW, HIGH, PTR, SHORT, THIS, SEG, OFFSET, SMSIZE, GRSIZE, SMOFFSET, GROFFSET, TYPE, LENGTH, SIZE, MASK, WIDTH, ( ), [ ], period (.), colon (:), and < >.

Macro and Code Macro Capability

RA70320 allows the definition of macro code sequences with parameters, LOCAL symbols, and special repeated code sequences. The macro code sequence is different from a subroutine call in that the invocation of a macro in the source code results in the direct replacement of a macro call with the defined code sequence.

RA70320 also allows the definition of code macros to give the user the capability of defining a new instruction (mnemonic). Although an instruction definition could also be defined using the ordinary macro facility, code macros specify the allowable operand types for the new instructions whereas ordinary macros cannot.

Directives

Assembler directives give instructions to the program but are not translated into machine code during assembly. Basic directives include those for storage definition and allocation (DB, DW, DD, DQ, DT, DBS, DWS, DDS, DQS, DTS, STRUC/ENDS, RECORD); symbol control (EQU, LABEL, PURGE); and program counter control (ORG, EVEN, ALIGN). Program control directives include those for segment definition and control (SEGMENT/ENDS, PROC/ENDP, ASSUME, GROUP, END); special function registers and internal RAM (SETIDB, ASGNSFR); linkage (NAME, PUBLIC, EXTRN); and PARITY.

The relocation types for SEGMENT/ENDS directives are specified in the operand column and include BYTE, WORD, PARA, PAGE, and INPAGE. The combination types of PUBLIC, COMMON, AT, STACK, and MEMORY, which are also specified in the operand column, define the means of linking segments and groups of the same name.

Controls

There are two types of assembler controls for the RA70320:

- Basic (specified in the assembler command line)
  - File specification
  - Output file selection
  - Output file destination
  - Listing format controls
  - Debug information output selection
  - Symbol case selection
  - Macro processing selection
- General (specified in the source program)
  - Inclusion of other source files
  - Page eject
  - Generation/suppression of listing
  - Listing titles

A list file may contain the complete assembly listing or it may contain only lines with errors and a symbol or cross-reference table. The symbol table lists all defined symbols in alphabetical order and also shows their types, attributes, and the values initially assigned to them. The cross-reference table contains all defined symbols, as well as the numbers of all statements referring to them.

The object file contains the relocatable object module. The format of this module conforms to NEC's proprietary relocatable object module format.

Linker

The LK70320 linker combines relocatable object modules and absolute load modules and produces one absolute load module (figure 2). The controls for the linker may be specified in either the command line or in a parameter file. In addition to being able to specify the date, the module name, the starting address and the order for code/data/stack segments, it is also possible to protect areas of memory from being assigned. Furthermore, it is possible to instruct the program to create a list file containing a link map, a local symbol table, or a public symbol table. The absolute load module contains symbol information for the symbolic debugger and the absolute object code.

Hexadecimal Object Code Converter

The OC70320 object code converter (figure 3) translates an absolute load module file into an expanded hexadecimal (7-bit ASCII) file that may be downloaded to a PROM programmer. Addresses may be specified as being output in the order in which they were input or in ascending order.

Librarian

The LB70320 librarian creates and maintains files containing relocatable object modules. The program reduces the number of files that need to be linked together by allowing several modules to be kept in a single file, and also provides an easy way to link frequently used modules into programs. Modules may be added, deleted, or replaced within a library file.

Operating Environment

The RA70320 package has been designed to run under a variety of operating systems. One version is available to run on an MS-DOS system with one or more disk drives and at least 512K of system memory. Other versions are available to run on a Digital Equipment Corporation VAX computer under the UNIX 4.2BSD, Ultrix, and the VMS (Version 4.1 or later) operating systems.

Downloading Files into the Emulator

Absolute load modules produced by the RA70320 Relocatable Assembler package can be debugged by using the NEC IE-70320 (V25) or IE-70330 (V35) stand-alone in-circuit emulator. Communication between these emulators and the host system is handled through an RS-232C serial line that uses the KERMIT communications protocol developed at Columbia University. With the appropriate version of KERMIT running on both the emulator and host system, absolute load modules or hexadecimal object code files may be transferred between machines.

A version of the KERMIT Communication Program is supplied with each NEC emulator. NEC supplies versions of KERMIT to run on the IBM PC, PC/XT™, PC AT™, or compatibles under MS-DOS operating systems, and the Digital Equipment VAX under VMS, UNIX 4.2BSD, or Ultrix. An appropriate version is provided with each relocatable assembler package at no extra charge. Versions of KERMIT for other host systems are available directly from Columbia University

A second means of loading files into the emulator is also available in the Multiple File Handler, a utility program that runs in the emulator and is supplied as part of the IE-70320 and IE-70330 packages. The Multiple File Handler allows the emulator to read MS-DOS formatted disks, among others.

Figure 2. Linker Functional Diagram

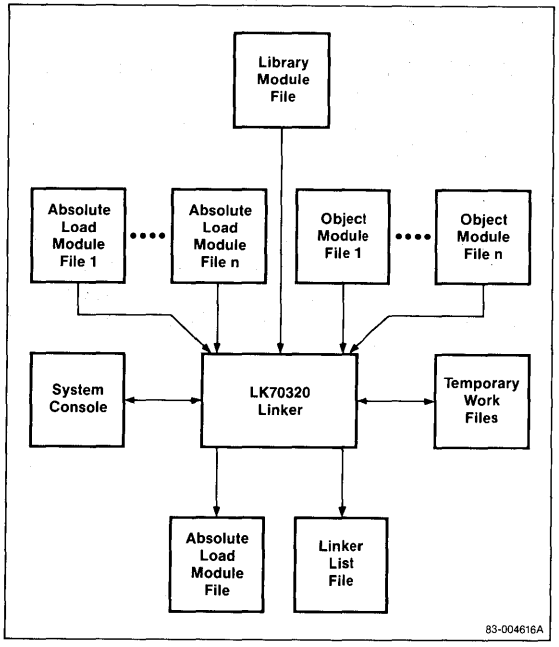
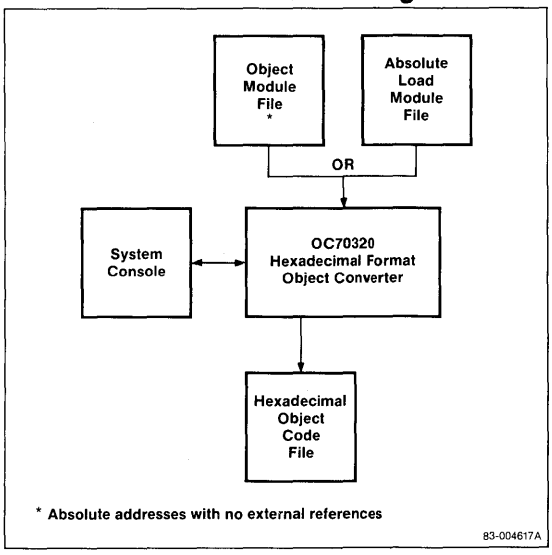


Figure 3. Hexadecimal Object Code Converter Functional Diagram



PC/XT and PC AT are trademarks of International Business Machines Corp.

6k

Documentation

For further information on source program formats, assembler operation, and actual program examples, refer to the following manuals supplied with the RA70320. Additional copies may be obtained from NEC Electronics Inc.

- RA70320 V25/V35 Relocatable Assembler Package Language Manual
- RA70320 V25/V35 Relocatable Assembler Package Operation Manual.

License Agreement

RA70320 is sold under terms of a license agreement included with purchased copies of the assembler. The accompanying card must be completed and returned to NEC Electronics Inc. to register the license. Software updates are provided to registered users.

Description

The V25 MINI-IE Plus and V35 MINI-IE Plus are low-cost In-Circuit Emulators for the μPD70320 (V25™) and μPD70330 (V35™) microcomputers from NEC Electronics. Low cost is achieved by using an IBM PC/XT®, PC AT®, IBM PS/2™, or compatible machine. The control software for the MINI-IE Plus is AdVICE (Advance V-Series In-Circuit Emulator), which acts as both a monitor and debugger. Debugging with breakpoint and non-real-time tracing of executing programs are accomplished in software using a V25/V35 microcomputer located on the MINI-IE Plus board. An optional real-time trace (RTT) board is available for those who need this additional tool.

Features

- Emulates μPD70320/70330 at up to 8 MHz.
- Jumper selectable internal or external (target) clock.
- Parallel interface with host PC; interface card and cable included
- Connects to target system via flexible PLCC socket adapter
  - Emulation memory may be mapped to MINI-IE Plus or target system
  - Supports two 64K-byte mappable user emulation RAM areas
- Software break and trace capabilities
  - Up to eight conditional breakpoints plus one in command line
  - Additional breakpoints can be given in the command line
  - Various actions can be programmed to take place on a break
  - Error checking of break entry conditions
- Optional real-time trace (RTT) board
  - 8K frames by 48-bit trace buffer
  - Two hardware breakpoints with don't care features
  - Eight external data inputs
  - Hardware trigger output
  - Qualifier controlled recording
  - 32-bit timer with 250-ns resolution

IBM PC/XT, IBM PC AT, and IBM PS/2 are registered trademarks of International Business Machines Corporation.
V25 and V35 are trademarks of NEC Corporation.

- Executes NEC .LNK absolute files and Microsoft. COM and .EXE files
  - Files can be downloaded to and uploaded from the MINI-IE Plus
  - Supports real-time and single-step emulation
  - User programmable public symbol buffer size
- Controlled by powerful AdVICE monitor and debugging program
  - Symbolic full screen debugger
  - Displays six window areas with second-level break setup window
  - Updates display information as program single-steps
  - On-line assembler
  - Programmable trace and symbol buffer sizes
  - On-line help menus
  - Keyboard macros speed up repetitive operations
  - User definable commands
  - Resident operation with hot key activation
  - Message exchange capabilities between PC and emulator
- Sample batch file contains demonstration program

Ordering Information

Part Number	Description
EB-V25MINI-IE-P	μPD70320 MINI-IE Plus package
EB-V35MINI-IE-P	μPD70330 MINI-IE Plus package
EB-V25/35-RTT	V25/V35 MINI-IE Plus real-time trace option

HARDWARE

The V25/V35 MINI-IE Plus package consists of five components.

- V25/V35 MINI-IE Plus box with target adapter
- Modified printer adapter card
- Interface cable
- Dc power plug
- AdVICE software and user's manual

The MINI-IE Plus contains two 64K-byte blocks of static RAM that is allocated by software to any 64K-byte boundary within the 1M-byte address range of the V25/35. ROM simulation is performed by write protecting this memory. An additional 64K bytes of RAM and a 128-byte I/O block are used by the internal monitor and can be relocated by command to avoid conflicts with external addressing needs.

Typical memory mapping allocates one block at the beginning of the address space (00000H) and the other at address 0F0000H. This may represent the final hardware configuration. The upper block would contain program code and be write protected. Any writes to this area would stop the program execution and allow the user to analyze the program. This feature allows debug when the program tries to write to ROM. Execution of the reset procedure is done by activating the target's $\overline{\text{RESET}}$ pin. Emulator hardware/software will not be reset by this action.

An optional real-time trace (RTT) board can be plugged on top of the emulator card to provide a 48-bit by 8K deep trace buffer. Eight external logic pins can be monitored along with the V25/V35 bus signals.

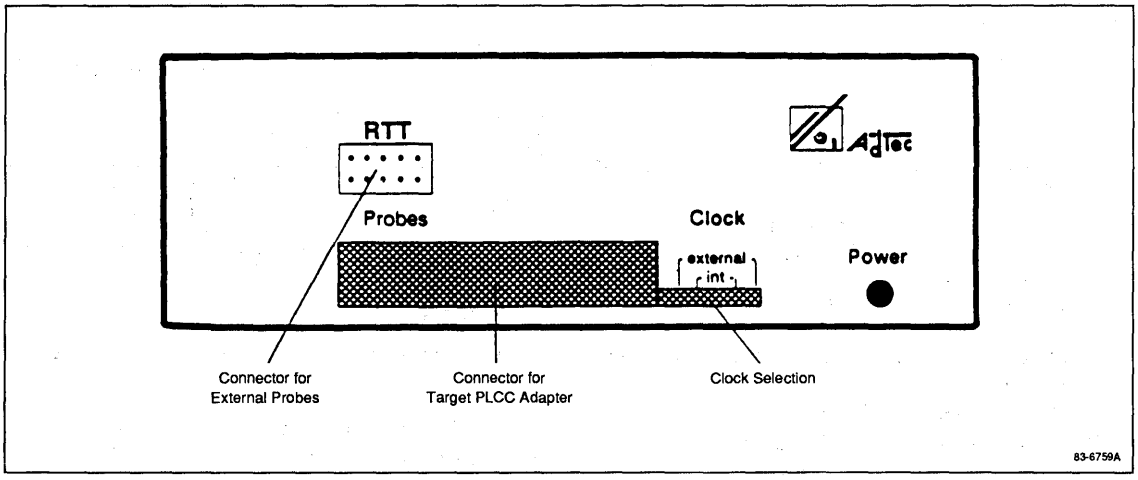
Command control of the $\overline{\text{EA}}$ pin of the V25/V35 allows use of ROM-based devices. After initialization, $\overline{\text{EA}}$ is forced low to access external memory, but can be forced high by command or may be controlled by the target hardware.

The NMI signal is normally used by the MINI-IE Plus to stop program execution using interrupt vector 02. Use of the target NMI signal in an application is possible by assigning an unused interrupt vector in place of the normal vector. Any high-to-low transitions of the target NMI signal will cause the emulator to execute this new vector.

The interface card is a modified printer adapter card that allows fast bidirectional communications between the host PC and the MINI-IE Plus. This card is not needed with an IBM PS/2. The interface cable connects the MINI-IE Plus to the interface card on PC/XT/AT or compatible or to the printer port of an IBM PS/2. Power for the MINI-IE Plus is supplied via the interface card. With PS/2, the dc power plug and a user-supplied external +5-volt power supply power the MINI-IE Plus.

The flexible target adapter allows direct connection to a PLCC socket of the target hardware. The cable is approximately 16 cm long and protrudes from the front of the MINI-IE Plus box. See figure 1.

Figure 1. MINI-IE Plus Front Panel



SOFTWARE

The control software (AV35N.EXE or AVRTT35N.EXE) uses the PC screen to display program and memory data. All information is updated after every command, and it keeps the user informed of the current state of the emulation. The screen is divided into seven areas. These areas display the current contents of the registers and bottom of stack, the command line, two memory dump areas with an additional ASCII dump display, the disassembler, and function key assignments. See figure 2.

Help and other setup screens, such as breakpoint menus, overlay some of the windows described above. Executing a new command will restore the display to the original screen. Figure 3 shows the AdVICE screen with a help window and the command line prompts.

AdVICE controls all the monitor and debug functions of the V25/V35 MINI-IE Plus including upload and download of programs, breaking, tracing, program execution, disassembly, line assembly, and register/memory display and manipulation. The cursor can be moved anywhere within the window displays for immediate change of the memory areas, registers, flags, and breakpoints.

Emulation

User programs loaded into emulator RAM can be executed in real-time or in single-step mode. Single-step mode executes only one instruction, and procedure step executes an entire subroutine or software interrupt routine. Real-time execution is command activated and terminates when a breakpoint is encountered or the user terminates execution from the keyboard. Program execution is also stopped when an exception interrupt or an interrupt with an unitialized vector occurs.

Message exchange between the PC and an executing application program is provided. See figure 4. An interrupt function similar to the DOS INT21 allows communication to the application program without a keyboard or display attached to the target system.

Figure 2. AdVICE Main Screen

```
AW F700   IX 0000   PS F800   PC 000F      Stack +0 FFFF    PSW F002  IntM:1  V25
BW 0000   IY 0000   DS F700                      +2 423E   RB 7
CW 3BA8   BP 0000   ES 0000   HS 0040            +4 23C3   V DIR IE S  Z AC  P CY
DW 0000   SP 00FE   SS 0040   FS 0040 Resident   +6 A352   0  0  0  0  0  0  0  0
 CMD >                                          1          0  1  2  3  4  5  6  7
                                                DS:0000   FF FF FF FF FF FF FF FF
  RESET:                                        DS:0008   FF FF FF FF FF FF FF FF
0000 B8FFFF          MOV     AW,FFFF            DS:0010   FF FF FF FF FF FF FF FF
0003 8ED8            MOV     DS,AW              DS:0018   FF FF FF FF FF FF FF FF
0005 C6060F00F7      MOV     [000F],F7          DS:0020   02 00 00 00 00 00 00 00
000A B800F7          MOV     AW,F700            DS:0028   00 00 00 00 00 81 00 00
000D 8ED8            MOV     DS,AW              DS:0030   00 00 00 00 00 00 00 00
000F C606020FFF      MOV     PMC0,FF            DS:0038   00 00 00 00 00 00 00 00
0014 C606E10FB2      MOV     RFM,B2             DS:0040   FF FF FF FF FF FF FF FF
0019 C706E80F5555    MOV     WTC,5555           DS:0048   FF FF FF FF FF FF FF FF

2          0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
FS:0000   CC 0F 42 6C D4 2F EE A2   1F 00 76 FE 98 84 7F F0   ├.Bl└/εó   ..v■ÿ≡
FS:0010   47 4D FE FB 5E 2A AC 53   0C 99 FD E7 6F 6B 81 FB   GM■√^*¼S   .Ö²τoküü√
FS:0020   FF FF FF FF FF FF FF FD   FF FF 3F FF FD FF FF FF                ²    ? ²
FS:0030   FF FF FF FF FF FF FF FF   FF FF FF FF FF 7F FF FF
FS:0040   22 44 CE AE A2 C7 67 FD   B2 09 EF F7 11 05 7F 73   "D╬«ó├g²   ▒.∩≈.s

1  Step   2ProcStep 3Retrieve 4Help ON  5BRK Menu 6 DOS  7 up  8 dn  9 le 10 ri
```

83-6760A

Figure 3. AdVICE Screen With Help Menu and Command Line Prompts

```
AW F700   IX 0000   PS F800   PC 000F      Stack +0 FFFF    PSW F002  IntM:1  V25
BW 0000   IY 0000   DS F700                     +2 423E   RB 7
CW 3BA8   BP 0000   ES 0000   HS 0040            +4 23C3   V DIR IE S  Z AC  P CY
DW 0000   SP 00FE   SS 0040   FS 0040 Resident  +6 A352   0  0  0  0  0  0  0  0
 ------D  or  DEF  or  DIR-------------------------------------------------------
 CMD >D                                          1         0  1  2  3  4  5  6  7
                                          -FFB8- DS:0000   FF FF FF FF FF FF FF FF
00E7 8BDF            MOV     BW,IY               DS:0008   FF FF FF FF FF FF FF FF
00E9 2EFF15          CALL    PS:[IY]             DS:0010   FF FF FF FF FF FF FF FF
00EC EBC6            BR      V40_CLI             DS:0018   FF FF FF FF FF FF FF FF
  VECTOR_TABLE:                                  DS:0020   02 00 00 00 00 00 00 00
00EE 56              PUSH    IX                  DS:0028   00 00 00 00 00 81 00 00
00EF 029802F0        ADD     BL,[F002+BW+IX]     DS:0030   00 00 00 00 00 00 00 00
00F3 0450            ADD     AL,50               DS:0038   00 00 00 00 00 00 00 00
00F5 03B40328        ADD     IX,[2803+IX]        DS:0040   FF FF FF FF FF FF FF FF
00F9 024805          ADD     CL,[BW+IX+05]       DS:0048   FF FF FF FF FF FF FF FF

 D [/M ON | OFF] addr
     Display - code at the specified address. With 'D *' or 'Ctrl-Enter' the
     address of the current instruction will be used. If a memory location is
     accessed by the actual instruction its value is shown on the screen. Use
     to optional /M parameter to control the display of this memory data.
     Standard segment PS:
                                   ---------With PgUp/PgDn text can be paged---------
1  Step   2ProcStep 3Retrieve 4Help OFF 5BRK Menu 6 DOS  7 up  8 dn  9 le 10 ri
```

83-6761A

Figure 4. Executing Program with Message Exchange

```
AW F700   IX 0000   PS 0065   PC 0000      Stack +0 FFFF    PSW F002  IntM:1  V25
BW 0000   IY 0000   DS F700                     +2 340C   RB 7
CW 00B2   BP 0000   ES 0000   HS 0060            +4 0012   V DIR IE S  Z AC  P CY
DW 0000   SP 01FE   SS 0040   FS 0040 Resident  +6 5400   0  0  0  0  0  0  0  0
 ------G  or  GC------------------------------------------------------------------
 CMD >G                                          1         0  1  2  3  4  5  6  7
                                                 DS:0000   FF FF FF FF FF FF FF FF
2   > *** E X E C U T I N G ***        Analyze   DS:0008   FF FF FF FF FF FF FF FF

 Emulator Messages :     < from emulator       > to emulator
<The V35/25 timer was started !
<Wait for the break condition to become true
```

83-6762A

Break Capabilities

The breakpoint entry menu is a second-level menu and can be entered by F5 key of the PC keyboard. The menu contains six fields: breakpoint number, breakpoint conditions, a count, number of occurrences, and action to be taken.

Up to eight address breakpoints can be defined in the menu. They can be tagged with conditions and a count value (maximum of 65,535) for the number of times the breakpoint is satisfied. Up to eight conditions may be entered for each breakpoint, including satisfying other breakpoints and comparisons of register and memory contents (direct or indirect addressing modes) with those of other registers, memory locations, and immediate values. See figure 5.

The action field defines an operation or operations to be done when a breakpoint occurs. Such actions are trace on/off, analyze on/off, restart the breakpoint, browse through a file, turn on/off the snap feature, and jump to a different section of code.

To set up the analyze mode, an address field in the breakpoint menu is left empty but conditions are placed in the condition field. Then, on enabling the analyze mode, the program is run in single-step mode while the conditions are checked. In the snap mode, the instruction and the register values are put in the trace buffer when conditions for a breakpoint are met.

Actions to be performed are taken when the occurrence counter is equal to a specified count. This occurrence counter is incremented only when all specified conditions are true. When emulation is to resume after a breakpoint, an option can be used in the GO command that will not reset the occur field and the trace mode.

Trace Capabilities

The MINI-IE Plus traces program execution in single-step mode. This software method allows the AdVICE monitor program to record the current register contents and the top four words of the stack. In order to start filling the trace buffer, the trace on action must be given at an address breakpoint. The buffer, which can be displayed from the main screen or from the menu screen, is filled in a round-robin fashion. See figure 6

The trace buffer is located in the MINI-IE monitor RAM. The default size of 37K bytes allows for 1000 records to be stored. A trace record consists of the V25/V35 instruction and associated register and stack contents. The buffer can be varied in size from 1k to 38k by means of a startup invocation switch.

The real-time trace option is available when requirements do not allow the single step of the user program for software tracing. The RTT will record the data on the bus in real time and the RTT software allows the user to view the data in disassembled format when the execution is stopped. See figure 7.

The RTT buffer is 8191 frames by 48 bits. A qualifier can be specified to restrict recording to specific conditions. Two hardware breakpoints—made from address, data, bus cycle type, and external logic inputs—allow multiple actions to be executed (such as RTrace on/off, break, timer on/off).

CONNECTION TO IBM PC

The V25/V35 MINI-IE Plus can be used with IBM PC/XT/AT or full compatibles, and on IBM PS/2. A monochrome, CGA, Hercules® color graphics, or EGA adapter and corresponding monitor is required. With IBM PC/XT/AT, at least one expansion slot must be available for use with the parallel interface adapter included with the MINI-IE Plus.

At least one printer port address should also be available. IBM PS/2 models will use the system printer port and will require a user-provided +5-volt dc power supply to power the MINI-IE Plus. A minimum of 88K bytes of free memory is needed to run AdVICE. A hard disk is not required but is recommended.

The parallel board of the MINI-IE Plus is hardwired for address 0278H of the PC I/O space. A provision on the parallel board allows you to change the I/O address to a different value, and a command line switch tells AdVICE where the parallel board is located.

With the MINI-IE Plus board connected to the parallel board via the 25-line cable, running the AdVICE program will initiate communications with the MINI-IE Plus and/or the target system.

Figure 5. Example of Breakpoint Entry Menu With Help

```
Can be used to define a condition that must be true when the breakpoint
is hit. Up to 8 conditions can be specified for each breakpoint. If all
conditions are true the 'Occur'-counter is incremented until it is equal
to 'Count' and the 'Action' is performed.
Example: BR3  AW>=123  DS:[IX]= /W ES:32B4  CY=1
For more details refer to the user's manual.

BR# Break_ADR  Condition                                Count Occur  Action
  1 MAIN20     ..................................       1         1  trace ON
  2 MAIN60     ..................................       1         1  TR OFF A ON
  3            SWTIM >= 0100 ....................       1         1  br
  4 -          ..................................       0         0
  5 -          ..................................       0         0
  6 -          ..................................       0         0
  7 -          ..................................       0         0
  8 -          ..................................       0         0
    Address    AddrM   Type Dir   Dat DaM   ExD ExM     P2 P2M     Action
EV1 TIMER      FFFFF   M    R     0   00    0   0       0  00      RT ON TI ON
EV2 0          FFFFF   M    R     0   00    0   0       0  00
QUA 0          000     X    X               0   0       RTT Mode: C

1View Trace 3Read Setup 4 Help OFF  5 CMD line  7Save Setup 9Clear BR10Clear OCC
```

83-6763A

Figure 6. Soft Trace Buffer Display

```
                T R A C E   B U F F E R   D I S P L A Y
 Buffer Offset :    0
*** Begin of TRACE buffer **
  MAIN20:                                AW=FF00  IX=0004  PS=0065  Stack+0 E803
0027 CALL    INIT_TIMER                  BW=0000  IY=0074  DS=0060       +2 340C
>Started by BR1   V DIR IE S  Z AC  P CY CW=00B2  BP=0000  ES=FF00       +4 0012
                  0  0  0  0  1  0  1  0 DW=1388  SP=01FE  SS=0040       +6 5400
  INIT_TIMER:                            AW=FF00  IX=0004  PS=0065  Stack+0 002A
003E MOV     ES:TMC0,C0                  BW=0000  IY=0074  DS=0060       +2 E803
                  V DIR IE S  Z AC  P CY CW=00B2  BP=0000  ES=FF00       +4 340C
                  0  0  0  0  1  0  1  0 DW=1388  SP=01FC  SS=0040       +6 5400
0044 MOV     ES:MD0,0140                 AW=FF00  IX=0004  PS=0065  Stack+0 002A
                                         BW=0000  IY=0074  DS=0060       +2 E803
                  V DIR IE S  Z AC  P CY CW=00B2  BP=0000  ES=FF00       +4 340C
                  0  0  0  0  1  0  1  0 DW=1388  SP=01FC  SS=0040       +6 5400
004B MOV     ES:TMIC0,07                 AW=FF00  IX=0004  PS=0065  Stack+0 002A
                                         BW=0000  IY=0074  DS=0060       +2 E803
                  V DIR IE S  Z AC  P CY CW=00B2  BP=0000  ES=FF00       +4 340C
                  0  0  0  0  1  0  1  0 DW=1388  SP=01FC  SS=0040       +6 5400
0051 RET                                 AW=FF00  IX=0004  PS=0065  Stack+0 002A
                                         BW=0000  IY=0074  DS=0060       +2 E803
                  V DIR IE S  Z AC  P CY CW=00B2  BP=0000  ES=FF00       +4 340C
                  0  0  0  0  1  0  1  0 DW=1388  SP=01FC  SS=0040       +6 5400

   Use cursor keys to scroll data up and down      F1 or '↵' to return
```

83-6764A

Figure 7. Real-Time Trace Buffer Display

```
                    R e a l - T i m e   T r a c e   D a t a
  Addr  Ctl  Data  External     P2                          Record :    1 of 2271
 00B02  M R   FA   FF=11111111 FF  _astart:
                                   FA              DI
 00B03  M R   B8   FF=11111111 FF  B82103          MOV     AW,0321
 00B04  M R   21   FF=11111111 FF  vdgroup@:
 00B05  M R   03   FF=11111111 FF
 00B06  M R   BB   FF=11111111 FF  BB2103          MOV     BW,0321
 00B07  M R   21   FF=11111111 FF
 00B08  M R   03   FF=11111111 FF
 00B09  M R   B9   FF=11111111 FF  B9B003          MOV     CW,03B0
 00B0A  M R   B0   FF=11111111 FF
 00B0B  M R   03   FF=11111111 FF
 00B0C  M R   B9   FF=11111111 FF  B91202          MOV     CW,0212
 00B0D  M R   12   FF=11111111 FF
 00B0E  M R   02   FF=11111111 FF
 00B0F  M R   FC   FF=11111111 FF  FC              CLR1    DIR
 00B10  M R   8E   FF=11111111 FF  8EDB            MOV     DS,BW
 00B11  M R   DB   FF=11111111 FF
 00B12  M R   BA   FF=11111111 FF  BAFFFF          MOV     DW,FFFF
 00B13  M R   FF   FF=11111111 FF
 00B14  M R   FF   FF=11111111 FF
 00B15  M R   8E   FF=11111111 FF  8EC2            MOV     ES,DW
 00B16  M R   C2   FF=11111111 FF
1Show MK1 2Show MK2 3Set MK1 4Set MK2  Ctl-PgDn/Up:+/- 100  Ctl-End/Home:+/-1000
```

83-6765A

V40/V50 MINI-IE In-Circuit Emulator

Description

The V40 MINI-IE and V50 MINI-IE are low-cost in-circuit emulators for the μPD70208 (V40™) and μPD70216 (V50™) microcomputers. The MINI-IE is designed to be used with an IBM PC/XT®, PC AT®, IBM PS/2®, or compatible machine. The control software for the MINI-IE is AFD-Sym (Advanced Full-Screen Debug with Symbols), which acts as both a monitor and debugger. Debugging with breakpoints and non-real-time tracing of executing programs are accomplished in software using a V40/V50 microprocessor located on the MINI-IE board.

Features

- Emulates μPD70208/70216 at 8 MHz
- Parallel interface with host PC; interface card and cable included
- Connects to target system via PGA or PLCC probe
- MINI-IE hardware
  - 64K bytes of static RAM starting at address 0000:0
  - MINI-IE reserved PROM at address F800:0 through FFFF:F
  - MINI-IE memory may be disabled to use target memories
  - 32 bytes of MINI-IE I/O starting at address 8000:0
  - Suppresses MREQ, MSTB, and IOSTB when accessing MINI-IE components
- Software break and trace capabilities
  - Up to eight conditional breakpoints in break definition mode
  - Additional breakpoints can be given in the command line
  - Various actions can be programmed to take place on a break
  - Error checking of break entry conditions
- Executes NEC .LNK absolute files and Microsoft .COM and .EXE files
  - Files can be downloaded to and uploaded from the MINI-IE
  - Supports real-time and single-step emulation
  - User-programmable public symbol buffer size
- Controlled by powerful AFD-Sym monitor and debugging program
  - Symbolic full-screen debugger
  - Uses function keys as well as direct command entry
  - Displays six window areas with second-level break setup window
  - Updates display information as program single steps
  - On-line assembler
  - Evaluates and displays arithmetic expressions
  - Displays color on color systems
  - Programmable trace and symbol buffer sizes
  - On-line help menus
  - Macro command capability
- Sample batch file contains demonstration program

IBM PC/XT, IBM PC AT, and IBM PS/2 are registered trademarks of International Business Machines Corporation.
Hercules is a registered trademark of Hercules Computer Technology Inc.
V40 and V50 are trademarks of NEC Corporation.

Ordering Information

Part Number	Description
EB-V40MINI-IE	μPD70208 (V40) PC-based MINI-IE with adapter
EB-V50MINI-IE	μPD70216 (V50) PC-based MINI-IE with adapter
ADAPT68PGA68PLCC	Adapter for 68-pin PLCC socket (shipped with MINI-IE)

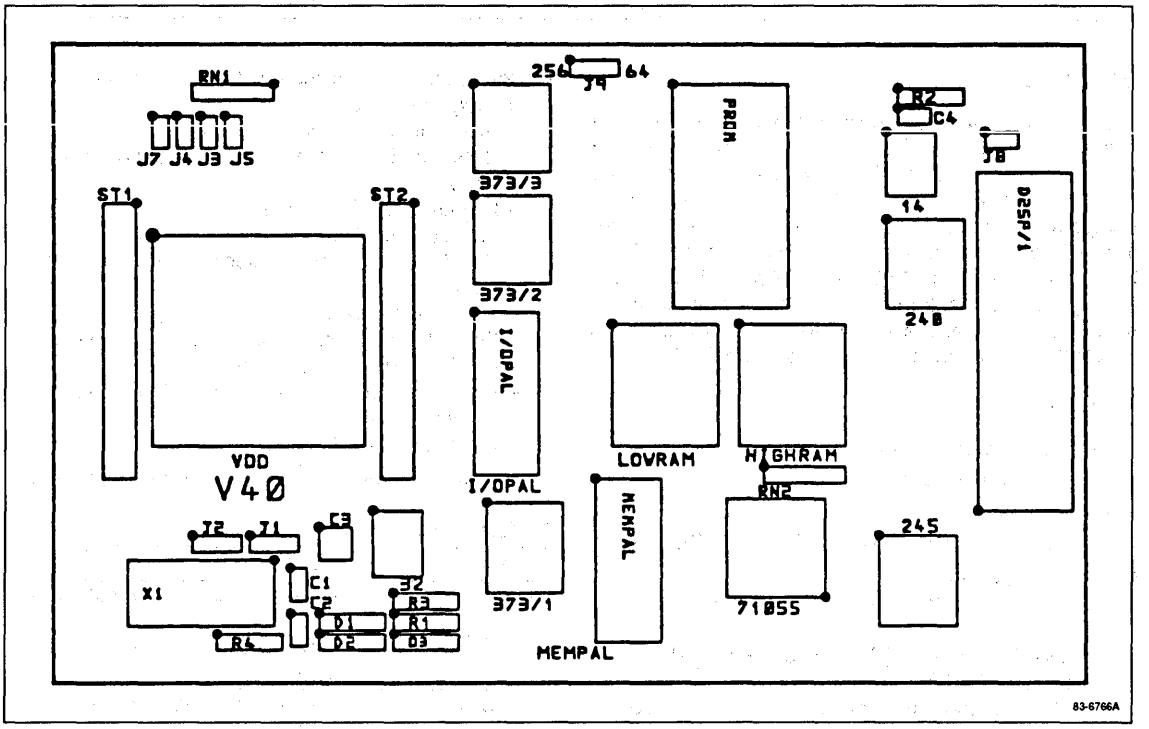
HARDWARE

The MINI-IE (figure 1) consists of a parallel interface board providing bidirectional data communication and a μPD70208 or μPD70216 emulation board (or MINI-IE board). The parallel board fits into a free slot in the PC and is hardwired for address 0278H within the PC I/O space.

The MINI-IE board consists of a μPD70208 or μPD70216, 32K bytes of EPROM, 64K bytes of static RAM, decoding logic, and a parallel interface. It can receive power from the PC, the target system, or an external supply. The MINI-IE board connects to the parallel board in the PC by a 25-line cable.

The MINI-IE can be connected directly into a 68-pin PGA socket on the target system or into a 68-pin PLCC socket on the target system through adapter ADAPT68PGA68-PLCC. In this configuration, the MINI-IE RAM and EPROM can be disabled via jumpers so that target memory is accessed, or a combination of MINI-IE and target memories can be used.

Figure 1. V40/V50 MINI-IE Board (Component Side)



83-6766A

Figure 2. AFD-Sym Main Screen

```
AW 0207   IX 0000   PS 0221   PC 0000     Stack +0 FFFF    PSW F002  IntM:1  V40
BW 0000   IY 0184   DS 0220                     +2 340C
CW 0089   BP 0000   ES 0000   HS 0220           +4 0012   V DIR IE S  Z AC  P CY
DW FFFD   SP 01FE   SS 0200   FS 01BB           +6 0000   0  0  0  0  0  0  0  0
      G  or  GC
 CMD >G                                                  1          0  1  2  3  4  5  6  7
                                                         DS:0000   0C 34 12 00 00 00 00 00
2   > *** E X E C U T I N G ***        Analyze           DS:0008   00 00 00 00 00 00 00 00
  MAIN:                                                  DS:0010   B8 20 02 8E D8 90 FA 33
0000 B82002          MOV     AW,0220                     DS:0018   C0 8E C0 90 FC BF 80 01
0003 8ED3            MOV     DS,AW                       DS:0020   B8 65 00 AB 8C C8 AB E8
0005 90              NOP                                 DS:0028   08 00 33 C0 FB 40 90 EB
0006 FA              DI                                  DS:0030   FC CC BA FC FF B0 00 EE
0007 33C0            XOR     AW,AW                       DS:0038   BA FA FF B0 60 EE BA F9
0009 8EC0            MOV     ES,AW                       DS:0040   FF B0 50 EE BA F0 FF B0
000B 90              NOP                                 DS:0048   01 EE BA FD FF B8 02 00

2          3  4  5  6  7  8  9  A    B  C  D  E  F  0  1  2
HS:0003   00 00 00 00 00 00 00 00   00 00 00 00 00 B8 20 02   ........  .....¬ .
HS:0013   8E D8 90 FA 33 C0 8E C0   90 FC BF 80 01 B8 65 00   Ä╪É·3└└  Éη¬Ç.╕e.
HS:0023   AB 8C C8 AB E8 08 00 33   C0 FB 40 90 EB FC CC BA   ½î╦½Φ..3  └√@Éδη╠│
HS:0033   FC FF B0 00 EE BA FA FF   B0 60 EE BA F9 FF B0 50   η ▒.ε│·   ▒'ε│·  P
HS:0043   EE BA F0 FF B0 01 EE BA   FD FF B8 02 00 EE B0 13   ε│≡ ▒.ε│  ² ¬..ε▒.

1  Step   2ProcStep 3Retrieve 4Help ON  5BRK Menu 6      7 up  8 dn  9 le 10 ri
```

83-6767A

SOFTWARE

The control software, AFD-Sym, provides six windows on the main screen display (figure 2).

- μPD70208/70216 registers and flags and the top four words of the stack
- Command entry line
- Offset addresses, opcodes, and disassembly of eight lines of the program load area
- 80-byte memory dump area
- Second 80-byte memory dump area
- ASCII equivalent of the second 80M-byte memory area

AFD-Sym controls all the monitor and debug functions of the MINI-IE including uploading and downloading programs, program execution with or without breakpoints, recording of trace history, disassembly, in-line code assembly, and register/memory display and manipulation. The cursor may be moved anywhere within the window displays for immediate change of the memory areas, registers, flags, and breakpoints.

AFD-Sym software running on a PC requires 75K to 147K bytes of system memory depending on the options required. It may be copied to a hard disk or run directly from the floppy diskette.

Emulation RAM area starting at address 0000:0 is partly used by the AFD-Sym monitor and the interrupt vectors. The lowest available user memory is indicated by the segment registers after reset and depends on the trace buffer size allocated. The AFD-Sym code in EPROM is located at F800:0 and the following 32K bytes are reserved for this purpose.

Emulation

User programs loaded into emulator RAM or EPROM can be executed in real-time or in single-step mode. Single-step mode executes only one instruction; procedure step executes an entire subroutine or software interrupt routine. It is possible to single-step a hardware interrupt handler when the associated interrupt mode is selected. Real-time execution is command activated and terminates when a breakpoint is encountered or the user terminates execution from the keyboard. Program execution is also stopped when an exception interrupt or an interrupt with an uninitialized vector occurs.

Programs generated using NEC's RA70320 Relocatable Assembler package allow symbols to be loaded directly with the program code. Programs produced with development tools that generate .EXE or .COM files can be loaded into the MINI-IE, but symbols must be loaded separately. Several symbol files can be loaded into the internal symbol table. The buffer used for symbol storage is allocated in the PC and can be varied in size from 1K to 64K bytes. Symbols can be added, deleted, and renamed interactively.

Break Capabilities

Two types of breakpoints are available. Immediate breakpoints are entered directly with the execution command and stop execution when the instruction at the specified location is to be executed. The second type of breakpoint is specified in a separate breakpoint menu.

The breakpoint entry menu (figure 3) is a second-level menu and can be entered by F5 key of the PC keyboard. The menu contains six fields: breakpoint number, breakpoint conditions, a count, number of occurrences, and action to be taken.

Up to eight address breakpoints can be defined in the menu. They can be tagged with conditions and a count value (up to 65,535) for the number of times the breakpoint is satisfied. Up to eight conditions may be entered for each breakpoint, including satisfying other breakpoints and comparisons of register and memory contents (direct or indirect addressing modes) with those of other registers, memory locations, and immediate values.

The action field defines an operation or operations to be done when a breakpoint occurs. Such actions include trace on/off, analyze on/off, restart the breakpoint, browse through a file, turn on/off the snap feature, and jump to a different section of code.

To set up the analyze mode, an address field in the breakpoint menu is left empty but conditions are placed in the condition field. Then, on enabling the analyze mode, the program is run in single-step mode while the conditions are checked. In the snap mode, the instruction and the register values are put in the trace buffer when conditions for a breakpoint are met.

Actions are performed when the occurrence counter is equal to a specified count. This occurrence counter is incremented only when all specified conditions are true. When emulation is to resume after a breakpoint, an option can be used in the GO command that will not reset the occur field and the trace mode.

Trace Capabilities

The MINI-IE trace is handled in software by the AFD-Sym. When trace is active, the current register contents and the top four words of the stack are saved to the trace buffer, along with the instruction that is to be executed in single-step mode. Program execution is not in real time when trace or analyze modes are enabled. Interrupt routines are, however, executed in real time and are not recorded in the trace buffer (unless the user selects interrupt step mode of the MINI-IE).

The trace buffer resides in the emulation memory and its size can be set from 1K to 64K bytes by the user. The default size of 4K bytes allows 100 records to be stored. Trace data, which can be displayed from the main screen or from the menu screen, can be shown with register contents or instructions only. See figure 4. Records can be printed or saved to a file. If symbols are loaded, they will be shown in the trace records.

CONNECTION TO IBM PC

The V40 or V50 MINI-IE can be used with IBM PC/XT/AT or full compatibles, and on IBM PS/2. A monochrome, CGA, Hercules® color graphics or EGA adapter and corresponding monitor is required. With IBM PC/XT/AT, at least one expansion slot must be available for use with the parallel interface adapter included with the MINI-IE.

At least one printer port address should also be available. IBM PS/2 models will use the system printer port and will require a user-provided +5-volt dc power supply to power the MINI-IE. A minimum of 75K bytes of free memory is needed to run AFD-Sym. A hard disk is not required but is recommended.

The parallel board of the MINI-IE is hardwired for address 0278H of the PC I/O space. A provision on the parallel board allows you to change the I/O address to a different value, and a command line switch tells AFD-Sym where the parallel board is located.

With the MINI-IE board connected to the parallel board via the 25-line cable, running the AFD-Sym program initiates communications with the MINI-IE and/or the target system.

Figure 3. Breakpoint Entry Menu

```
AW 02A0   IX 0000   PS 0221   PC 001E     Stack +0 FFFF    PSW F216  IntM:1  V40
BW 0000   IY 0184   DS 0220                    +2 340C
CW 0089   BP 0000   ES 0000   HS 0220          +4 0012   V DIR IE S  Z AC  P CY
DW FFFD   SP 01FE   SS 0200   FS 01BB          +6 0001   0  0  1  0  0  1  1  0

    Defines the address of a breakpoint. With '-' as the first character
    the breakpoint is inactive. A breakpoint with an empty address field
    will be checked on every break if count>0 and an action is specified
    for this breakpoint. With 'Alt'+n (n=1..8) the address of the actual
    instruction will be stored to the corresponding breakpoint.
    Standard segment PS:

BR# Break_ADR  Condition                          Count Occur  Action
  1 MAIN20     ..................................  1       1    trace ON
  2 MAIN60     ..................................  1       1    TR OFF A ON
  3            SWTIM >= 0100 ....................  1       1    br
  4 -          ..................................  0       0
  5 -          ..................................  0       0
  6 -          ..................................  0       0
  7 -          ..................................  0       0
  8 -          ..................................  0       0

1View Trace 3Read Setup 4 Help OFF  5 CMD line  7Save Setup 9Clear BR10Clear OCC
```

83-6768A

Figure 4. Trace Buffer Display

```
                T R A C E   B U F F E R   D I S P L A Y
 Buffer Offset :    0
*** Begin of TRACE buffer **
  MAIN20:                                  AW=0221  IX=0000  PS=0221  Stack+0 FFFF
0017 CALL    INITV40                       BW=0000  IY=0184  DS=0220       +2 340C
>Started by BR1   V DIR IE S  Z AC  P CY CW=0089  BP=0000  ES=0000       +4 0012
                  0  0  0  0  1  0  1  0 DW=FFFD  SP=01FE  SS=0200       +6 0000
  INITV40:                                 AW=0221  IX=0000  PS=0221  Stack+0 001A
0022 MOV     DW,FFFC                       BW=0000  IY=0184  DS=0220       +2 FFFF
                  V DIR IE S  Z AC  P CY CW=0089  BP=0000  ES=0000       +4 340C
                  0  0  0  0  1  0  1  0 DW=FFFD  SP=01FC  SS=0200       +6 0000
0025 MOV     AL,00                         AW=0221  IX=0000  PS=0221  Stack+0 001A
                                           BW=0000  IY=0184  DS=0220       +2 FFFF
                  V DIR IE S  Z AC  P CY CW=0089  BP=0000  ES=0000       +4 340C
                  0  0  0  0  1  0  1  0 DW=FFFC  SP=01FC  SS=0200       +6 0000
0027 OUT     DW,AL                         AW=0200  IX=0000  PS=0221  Stack+0 001A
                                           BW=0000  IY=0184  DS=0220       +2 FFFF
                  V DIR IE S  Z AC  P CY CW=0089  BP=0000  ES=0000       +4 340C
                  0  0  0  0  1  0  1  0 DW=FFFC  SP=01FC  SS=0200       +6 0000
0028 MOV     DW,FFFA                       AW=0200  IX=0000  PS=0221  Stack+0 001A
                                           BW=0000  IY=0184  DS=0220       +2 FFFF
                  V DIR IE S  Z AC  P CY CW=0089  BP=0000  ES=0000       +4 340C
                  0  0  0  0  1  0  1  0 DW=FFFC  SP=01FC  SS=0200       +6 0000

   Use cursor keys to scroll data up and down      F1 or '↵' to return
```

836/69A

6m

NEC

Section 7
Package Drawings

Package/Device Cross-Reference

Package	Device, μPD
18-Pin Plastic DIP	71011C-8 71011C-10 71084C-8 71084C-10
20-Pin Plastic DIP (300 mil)	71082C 71083C 71086C 71087C 71088C-8 71088C-10
20-Pin Plastic SOP (300 mil)	71011G-8 71082G 71083G 71084G-8 71086G 71087G 71088G-8
24-Pin Plastic DIP (600 mil)	71054C-8 71054C-10
28-Pin Plastic DIP (600 mil)	71051C-8 71051C-10 71059C-8 71059C-10
28-Pin PLCC	71051L-8 71051L-10 71054L-8 71054L-10 71059L-8 71059L-10
40-Pin Plastic DIP (600 mil)	70108C-8 70108C-10 70116C-8 70116C-10 71037CZ-10 71055C-8 71055C-10
44-Pin Plastic QFP (P44G-80-22); 1.45 mm thick	71054G-8 71055G-8 71059G-8

Package	Device, μPD
44-Pin Plastic QFP (P44GB-80-3B4); 2.70 mm thick	71037GB-10 71051GB-8 71051GB-10 71054GB-8 71054GB-10 71055GB-8 71055GB-10 71059GB-8 71059GB-10
44-Pin PLCC	70108L-8 70108L-10 70116L-8 70116L-10 71037LM-10 71055L-8 71055L-10
48-Pin Plastic DIP	71071C-10
52-Pin Plastic QFP	70108GC-8 70108GC-10 70116GC-8 70116GC-10
52-Pin PLCC	71071L-10
68-Pin PLCC	70136L-12 70136L-16 70208L-8 70208L-10 70216L-8 70216L-10
68-Pin Ceramic PGA	70136R-12 70136R-16 70208R-8 70208R-10 70216R-8 70216R-10
74-Pin Plastic QFP	70136GJ-12 70136GJ-16
80-Pin Plastic QFP	70208GF-8 70208GF-10 70216GF-8 70216GF-10

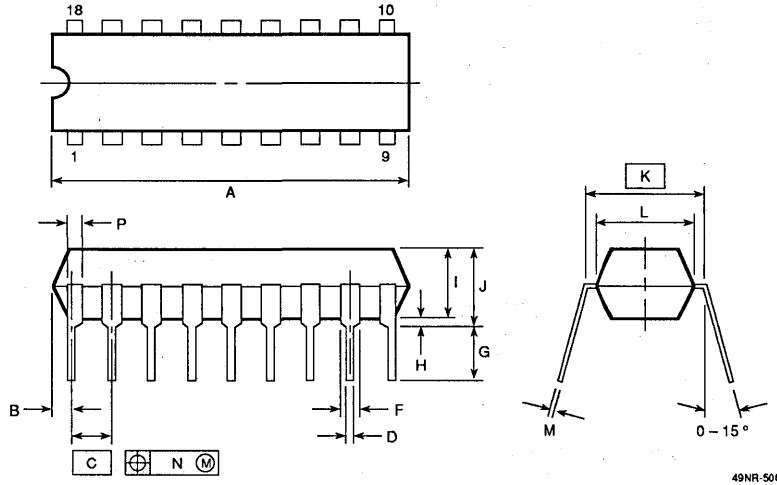
Package/Device Cross-Reference (cont)

Package	Device, μPD
84-Pin PLCC	70320L 70320L-8
	70322L-xxx 70322L-8-xxx
	70325L-8 70325L-10
	70327L-8-xxx 70330L-8 70332L-8-xxx
	70335L-8 70335L-10
	70337L-8-xxx 79011L-8 79021L-8
84-Pin Ceramic LCC	70P322KE-8
94-Pin Plastic QFP	70320GJ 70320GJ-8
	70322GJ-xxx 7022GJ-8-xxx
	70325GJ-8 70325GJ-10
	70327GJ-8-xxx 70330GJ-8 70332GJ-8-xxx
	70335GJ-8 70335GJ-10
	70337GJ-8-xxx 79011GJ-8 79021GJ-8
120-Pin Plastic QFP	70236GD-10 70236GD-12 70236GD-16
132-Pin Ceramic PGA	70236R-10 70236R-12 70236R-16
	71641R

18-Pin Plastic DIP

Item	Millimeters	Inches
A	22.86 max	.900 max
B	1.27 max	.050 max
C	2.54 (TP)	.100 (TP)
D	0.50 ± 0.10	.020 $^{+.004}_{-.005}$
F	1.2 min	.047 min
G	3.5 ± 0.3	.138 ± .012
H	0.51 min	.020 min
I	4.31 max	.170 max
J	5.08 max	.200 max
K *	7.62 (TP)	.300 (TP)
L	6.4	.252
M	0.25 $^{+0.10}_{-0.05}$	.010 $^{+.004}_{-.003}$
N	0.25	.010
P	1.0 min	.039 min

* Item K to center of leads when formed parallel.



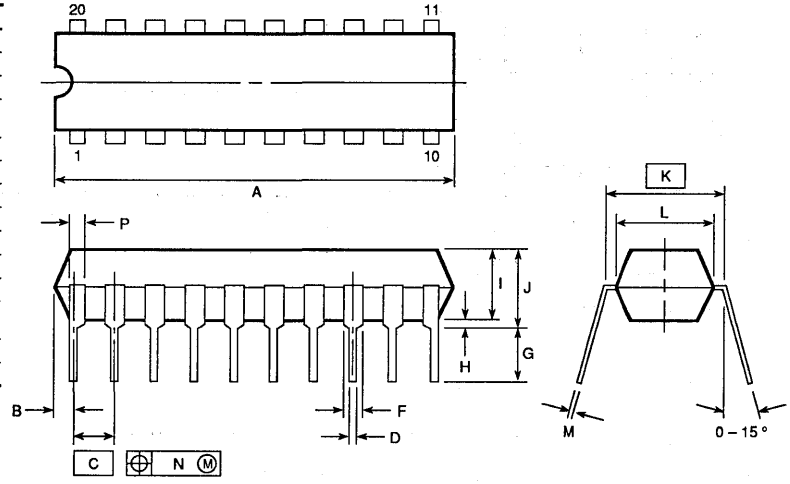
P18C-100-300A, C

49NR-506B (4/89)

20-Pin Plastic DIP (300 mil)

Item	Millimeters	Inches
A	25.40 max	1.000 max
B	1.27 max	.050 max
C	2.54 (TP)	.100 (TP)
D	0.50 ± 0.10	.020 $^{+.004}_{-.005}$
F	1.1 min	.043 min
G	3.5 ± 0.3	.138 ± .012
H	0.51 min	.020 min
I	4.31 max	.170 max
J	5.08 max	.200 max
K *	7.62 (TP)	.300 (TP)
L	6.4	.252
M	0.25 $^{+0.10}_{-0.05}$	.010 $^{+.004}_{-.003}$
N	0.25	.010
P	0.9 min	.035 min

* Item K to center of leads when formed parallel.



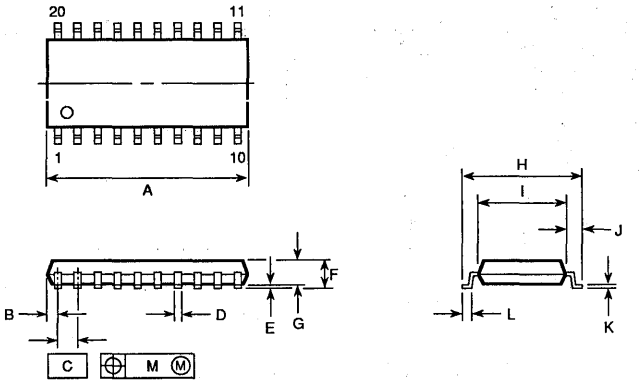
P20C-100-300A, C

49NR-624B (11/89)

7

20-Pin Plastic SOP (300 mil)

Item	Millimeters	Inches
A	13.00 max	.512 max
B	0.78 max	.031 max
C	1.27 (TP)	.050 (TP)
D	$0.40^{+0.10}_{-0.05}$	$.016^{+.004}_{-.003}$
E	0.1 ±0.1	.004 ±.004
F	1.8 max	.071 max
G	1.55	.061
H	7.7 ±0.3	.303 ±.012
I	5.6	.220
J	1.1	.043
K	$0.20^{+0.10}_{-0.05}$	$.008^{+.004}_{-.002}$
L	0.6 ±0.2	$.024^{+.008}_{-.009}$
M	0.12	.005



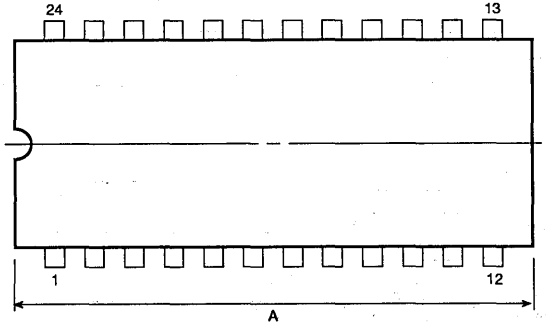
P20GM-50-300B, C

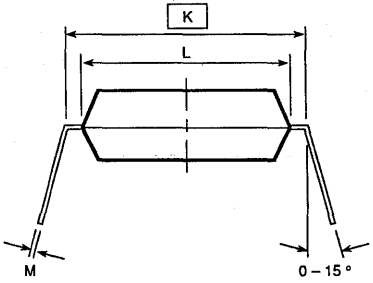
49NR-594B (9/89)

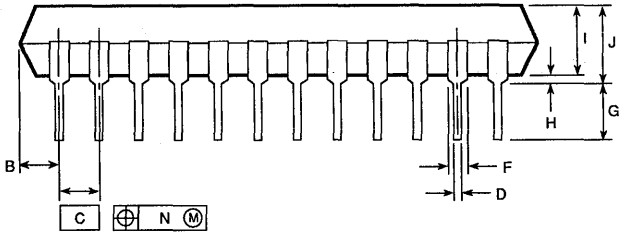
24-Pin Plastic DIP (600 mil)

Item	Millimeters	Inches
A	33.02 max	1.300 max
B	2.54 max	.100 max
C	2.54 (TP)	.100 (TP)
D	0.50 ±0.10	$.020^{+.004}_{-.005}$
F	1.2 min	.047 min
G	3.5 ±0.3	.138 ±.012
H	0.51 min	.020 min
I	4.31 max	.170 max
J	5.72 max	.226 max
K *	15.24 (TP)	.600 (TP)
L	13.2	.520
M	$0.25^{+0.10}_{-0.05}$	$.010^{+.004}_{-.003}$
N	0.25	.010

* Item K to center of leads when formed parallel.







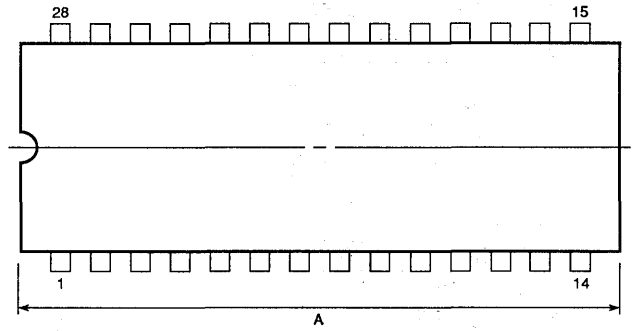
P24C-100-600

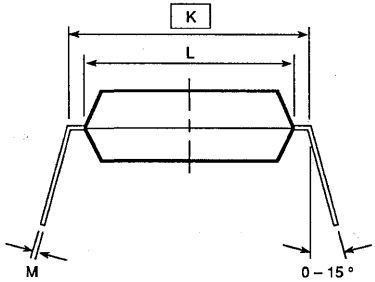
49NR-592B (9/89)

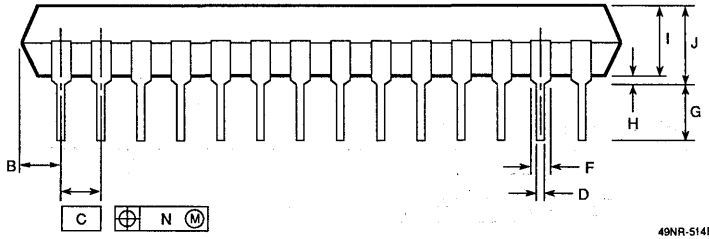
28-Pin Plastic DIP (600 mil)

Item	Millimeters	Inches
A	38.10 max	1.500 max
B	2.54 max	.100 max
C	2.54 (TP)	.100 (TP)
D	0.50 ±0.10	.020 $^{+.004}_{-.005}$
F	1.2 min	.047 min
G	3.6 ±0.3	.142 ±.012
H	0.51 min	.020 min
I	4.31 max	.170 max
J	5.72 max	.226 max
K *	15.24 (TP)	.600 (TP)
L	13.2	.520
M	0.25 $^{+0.10}_{-0.05}$	.010 $^{+.004}_{-.003}$
N	0.25	.010

* Item K to center of leads when formed parallel.







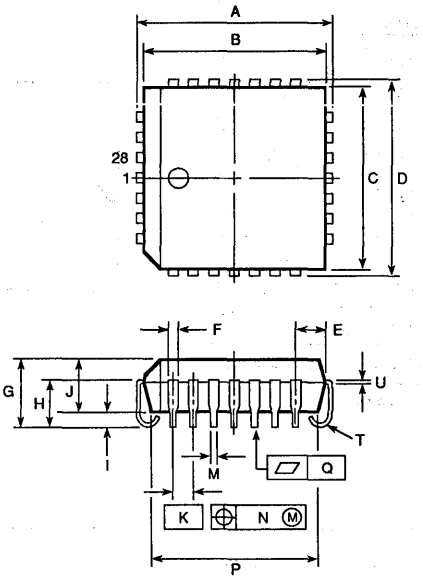
P28C-100-600A1

49NR-514B (5/89)

7

28-Pin PLCC

Item	Millimeters	Inches
A	12.45 ±0.2	.490 ±.008
B	11.50	.453
C	11.50	.453
D	12.45 ±0.2	.490 ±.008
E	1.94 ±0.15	.076 $^{+.007}_{-.008}$
F	0.6	.024
G	4.4 ±0.2	.173 $^{+.009}_{-.008}$
H	2.8 ±0.2	.110 $^{+.009}_{-.008}$
I	0.9 min	.035 min
J	3.4	.134
K	1.27 (TP)	.050 (TP)
M	0.40 ±0.10	.016 $^{+.004}_{-.005}$
N	0.12	.005
P	10.42 ±0.20	.410 $^{+.009}_{-.008}$
Q	0.15	.006
T	0.8 rad	.031 rad
U	0.20 $^{+0.10}_{-0.05}$	.008 $^{+.004}_{-.002}$



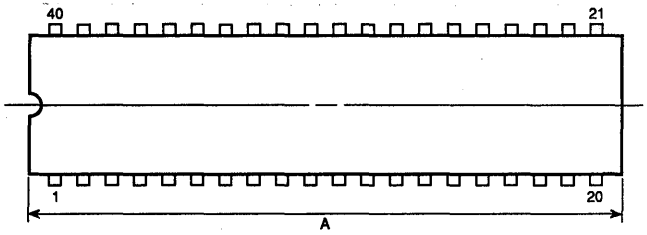
P28L-50A1

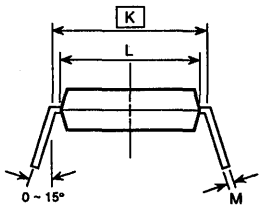
49NR-525B (5/89)

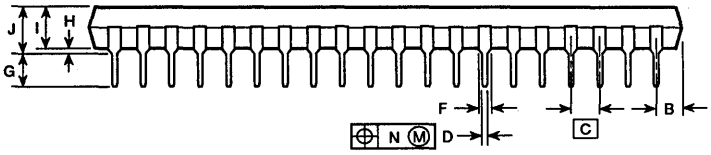
40-Pin Plastic DIP (600 mil)

Item	Millimeters	Inches
A	53.34 max	2.100 max
B	2.54 max	.100 max
C	2.54 (TP)	.100 (TP)
D	0.50 ±0.10	.020 $^{+.004}_{-.005}$
F	1.2 min	.047 min
G	3.6 ±0.3	.142 ±.012
H	0.51 min	.020 min
I	4.31 max	.170 max
J	5.72 max	.226 max
K*	15.24 (TP)	.600 (TP)
L	13.2	.520
M	0.25 $^{+0.10}_{-0.05}$	.010 $^{+.004}_{-.003}$
N	0.25	.010

* Item K to center of leads when formed parallel





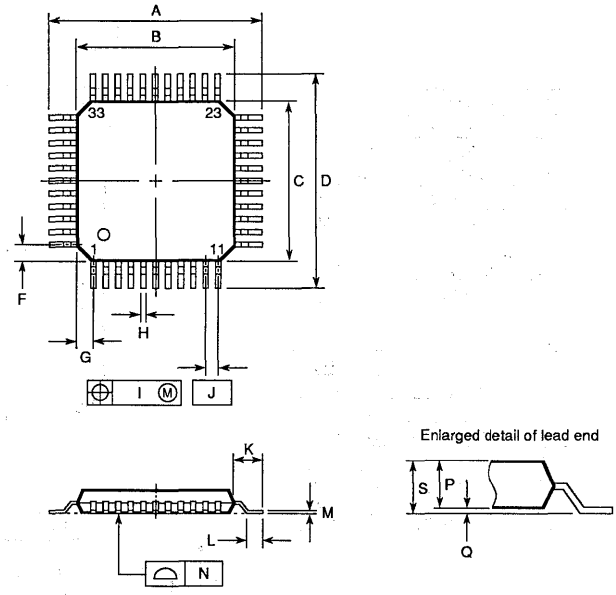


P40C-100-600A

83vQ-6140B (1/90)

44-Pin Plastic QFP (P44G-80-22); 1.45 mm thick

Item	Millimeters	Inches
A	13.6 ±0.4	.535 $^{+.017}_{-.016}$
B	10.0 ±0.2	.394 $^{+.008}_{-.009}$
C	10.0 ±0.2	.394 $^{+.008}_{-.009}$
D	13.6 ±0.4	.535 $^{+.017}_{-.016}$
F	1.0	.039
G	1.0	.039
H	0.35 $^{+0.20}_{-0.10}$	.014 $^{+.008}_{-.005}$
I	0.15	.006
J	0.8 (TP)	.031 (TP)
K	1.8 ±0.2	.071 $^{+.008}_{-.009}$
L	1.0 ±0.2	.039 $^{+.009}_{-.008}$
M	0.15 $^{+0.10}_{-0.05}$	.006 $^{+.004}_{-.003}$
N	0.15	.006
P	1.45 ±0.1	.057 $^{+.005}_{-.004}$
Q	0.0 ±0.1	.000 ±.004
S	1.65 max	.065 max

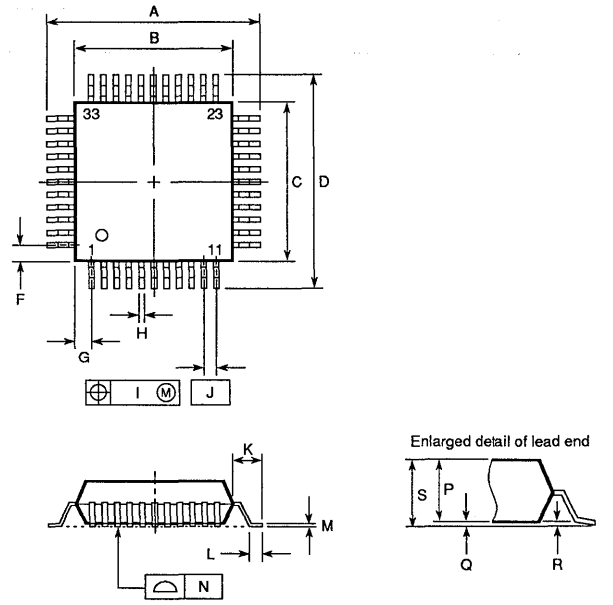


P44G-80-22

49NR-636B (11/89)

44-Pin Plastic QFP (P44GB-80-3B4); 2.70 mm thick

Item	Millimeters	Inches
A	13.6 ±0.4	.535 $^{+.017}_{-.016}$
B	10.0 ±0.2	.394 $^{+.008}_{-.009}$
C	10.0 ±0.2	.394 $^{+.008}_{-.009}$
D	13.6 ±0.4	.535 $^{+.017}_{-.016}$
F	1.0	.039
G	1.0	.039
H	0.35 ±0.10	.014 $^{+.004}_{-.005}$
I	0.15	.006
J	0.8 (TP)	.031 (TP)
K	1.8 ±0.2	.071 $^{+.008}_{-.009}$
L	0.8 ±0.2	.031 $^{+.009}_{-.008}$
M	0.15 $^{+0.10}_{-0.05}$	.006 $^{+.004}_{-.003}$
N	0.15	.006
P	2.7	.106
Q	0.1 ±0.1	.004 ±.004
R	0.1 ±0.1	.004 ±.004
S	3.0 max	.119 max



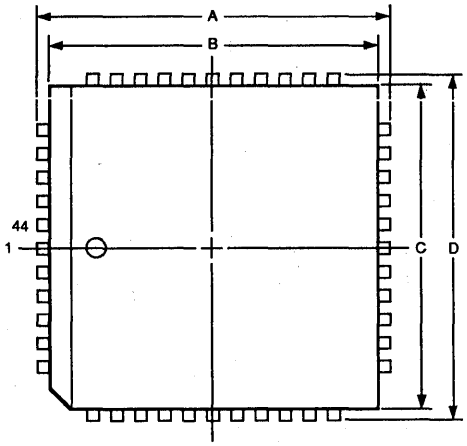
P44GB-80-3B4-1

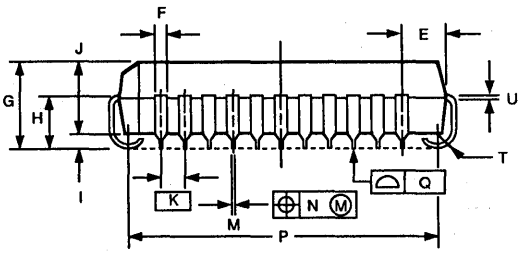
49NR-556B (1/90)

7

44-Pin PLCC

item	millimeters	inches
A	17.5 ±0.2	.689 ±.008
B	16.58	.653
C	16.58	.653
D	17.5 ±0.2	.689 ±.008
E	1.94 ±0.15	.076 ±.006
F	0.6	.024
G	4.4 ±0.2	.173 ±.008
H	2.8 ±0.2	.110 ±.008
I	0.9 min	.035 min
J	3.4	.134
K	1.27 (TP)	.050 (TP)
M	0.40 ±0.10	.016 ±.004
N	0.12	.005
P	15.50 ±0.20	.610 ±.008
Q	0.15	.006
T	0.8 radius	.031 radius
U	$0.20^{+0.10}_{-0.05}$	$.008^{+.004}_{-.002}$



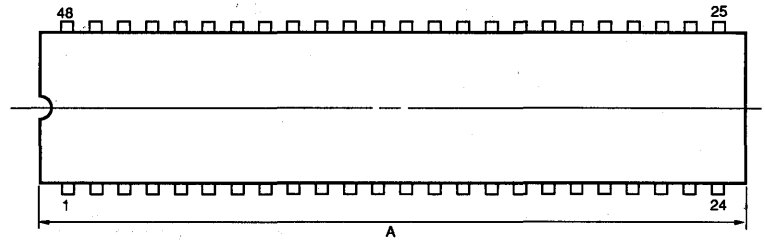


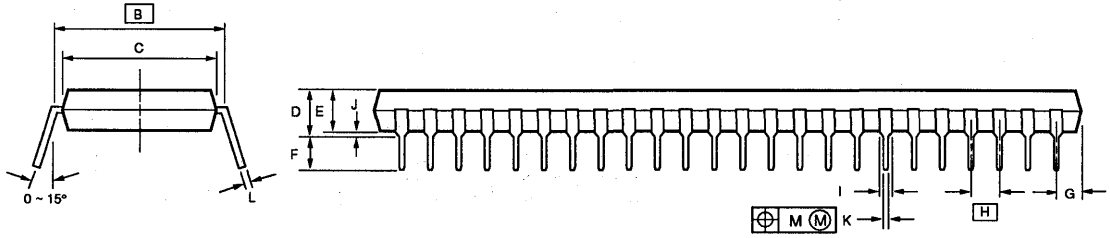
P44L-50A1-1

3/90 83YL-5804B

48-Pin Plastic DIP

Item	Millimeters	Inches
A	63.50 max	2.500 max
B	15.24 [TP]	.600 [TP]
C	13.8	.543
D	5.72 max	.225 max
E	4.31 max	.170 max
F	3.6 ±0.3	.142 ±.012
G	2.54 max	.100 max
H	2.54 [TP]	.100 [TP]
I	1.1 min	.043 min
J	0.51 min	.020 min
K	0.50 ±0.10	.020 ±.004
L	0.25 $^{+0.10}_{-0.05}$	.010 $^{+.004}_{-.002}$
M	0.25	.010





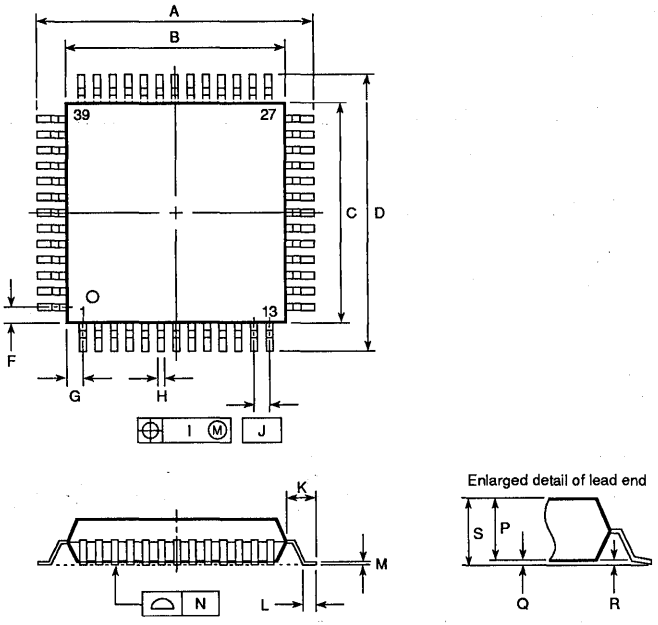
P48C-100-600A

83vQ-6138B (6/89)

NEC

52-Pin Plastic QFP

Item	Millimeters	Inches
A	17.6 ±0.4	.693 ±.016
B	14.0 ±0.2	.551 $^{+.009}_{-.008}$
C	14.0 ±0.2	.551 $^{+.009}_{-.008}$
D	17.6 ±0.4	.693 ±.016
F	1.0	.039
G	1.0	.039
H	0.40 ±0.10	.016 $^{+.004}_{-.005}$
I	0.20	.008
J	1.0 (TP)	.039 (TP)
K	1.8 ±0.2	.071 $^{+.008}_{-.009}$
L	0.8 ±0.2	.031 $^{+.009}_{-.008}$
M	0.15 $^{+0.10}_{-0.05}$	.006 $^{+.004}_{-.003}$
N	0.15	.006
P	2.7	.106
Q	0.1 ±0.1	.004 ±.004
R	0.1 ±0.1	.004 ±.004
S	3.0 max	.119 max

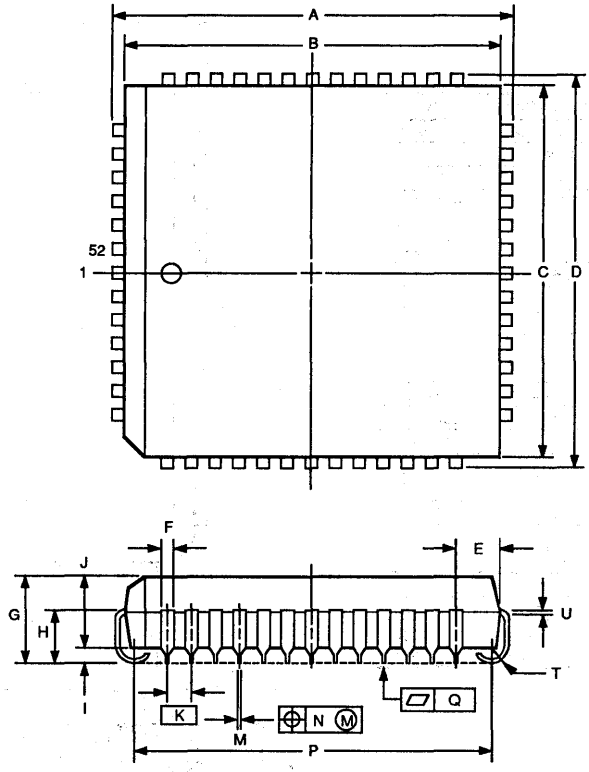


P52GC-100-3B6

49NR-493B (5/89)

52-Pin PLCC

Item	Millimeters	Inches
A	20.1 ±0.2	.791 ±.008
B	19.12	.753
C	19.12	.753
D	20.1 ±0.2	.791 ±.008
E	1.94 ±0.15	.076 ±.006
F	0.6	.024
G	4.4 ±0.2	.173 ±.008
H	2.8 ±0.2	.110 ±.008
I	0.9 min	.035 min
J	3.4	.134
K	1.27 (TP)	.050 (TP)
M	0.40 ±0.10	.016 ±.004
N	0.12	.005
P	18.04 ±0.20	.710 ±.008
Q	0.15	.006
T	0.8 radius	.031 radius
U	$0.20^{+0.10}_{-0.05}$	$.008^{+.004}_{-.002}$

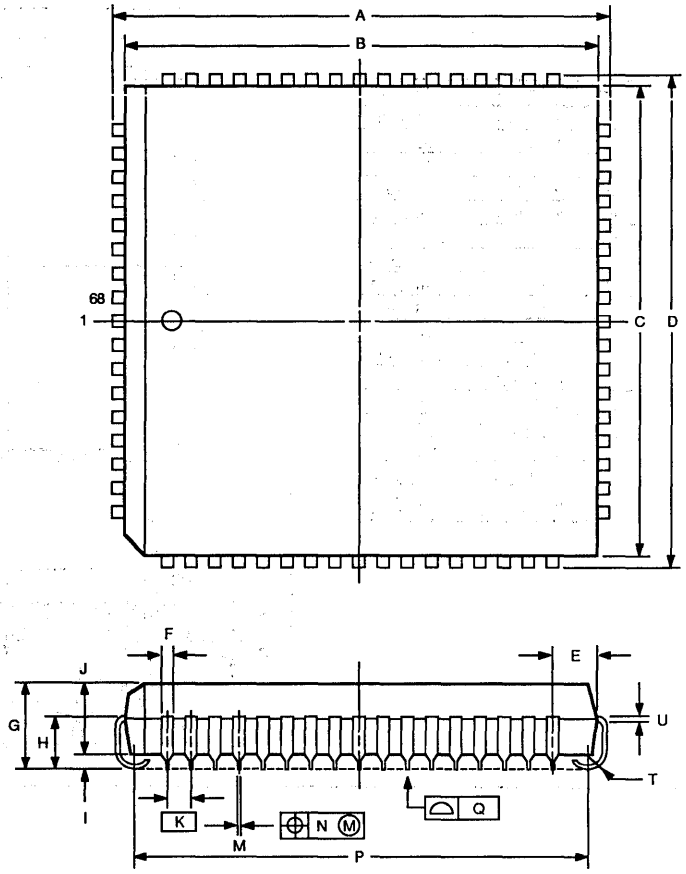


P52L-50A1

83YL-5805B

68-Pin PLCC

Item	Millimeters	Inches
A	25.2 ±0.2	.992 ±.008
B	24.20	.953
C	24.20	.953
D	25.2 ±0.2	.992 ±.008
E	1.94 ±0.15	$.076^{+.007}_{-.006}$
F	0.6	.024
G	4.4 ±0.2	$.173^{+.009}_{-.008}$
H	2.8 ±0.2	$.110^{+.009}_{-.008}$
I	0.9 min	.035 min
J	3.4	.134
K	1.27 (TP)	.050 (TP)
M	0.40 ±0.10	$.016^{+.004}_{-.005}$
N	0.12	.005
P	23.12 ±0.20	$.910^{+.009}_{-.008}$
Q	0.15	.006
T	0.8 radius	.031 radius
U	$0.20^{+0.10}_{-0.05}$	$.008^{+.004}_{-.002}$

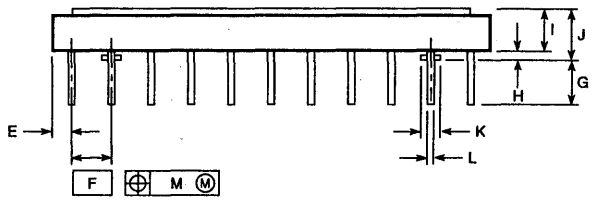


P68L-50A1-1

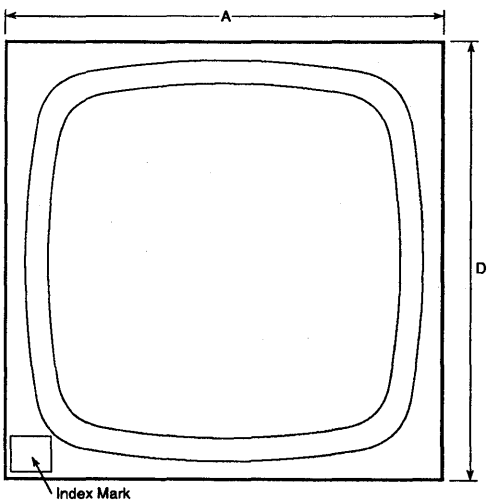
(2/90)
83YL-5561B

68-Pin Ceramic PGA

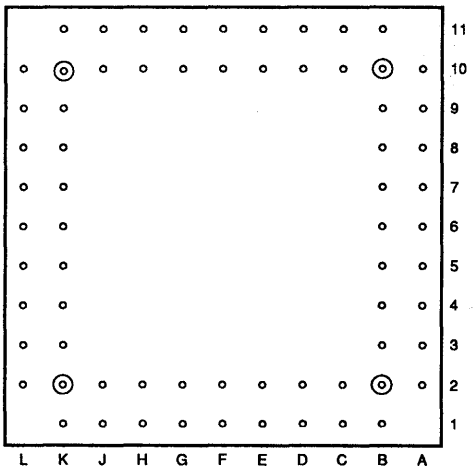
Item	Millimeters	Inches
A	27.94 ± 0.4	1.100 $^{+.016}_{-.015}$
D	27.94 ± 0.4	1.100 $^{+.016}_{-.015}$
E	1.27	.050
F	2.54 (TP)	.100 (TP)
G	2.8 ± 0.3	.110 $^{+.012}_{-.011}$
H	0.5 min	.019 min
I	2.70	.106
J	4.57 max	.180 max
K	1.2 ± 0.2 dia	.047 $^{+.008}_{-.007}$
L	0.46 ± 0.05 dia	.018 $^{+.002}_{-.001}$
M	0.5	.020



Top View



Bottom View



X68R-100A-1

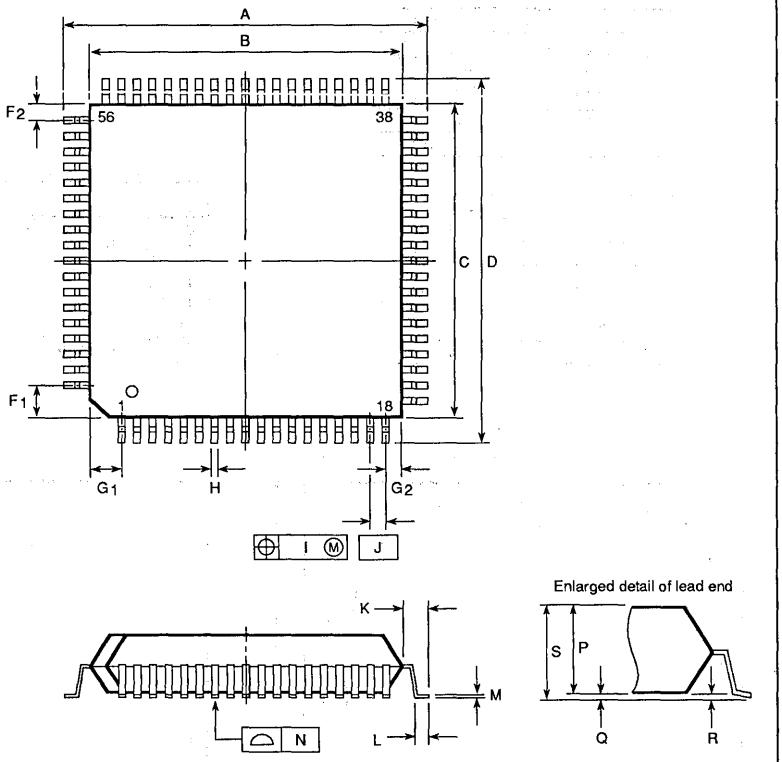
49NR-528B (5/89)

7

NEC

74-Pin Plastic QFP

Item	Millimeters	Inches
A	23.2 ±0.4	.913 $^{+.017}_{-.016}$
B	20.0 ±0.2	.787 $^{+.009}_{-.008}$
C	20.0 ±0.2	.787 $^{+.009}_{-.008}$
D	23.2 ±0.4	.913 $^{+.017}_{-.016}$
F_1	2.0	.079
F_2	1.0	.039
G_1	2.0	.079
G_2	1.0	.039
H	0.40 ±0.10	.016 $^{+.004}_{-.005}$
I	0.20	.008
J	1.0 (TP)	.039 (TP)
K	1.6 ±0.2	.063 ±.002
L	0.8 ±0.2	.031 $^{+.009}_{-.008}$
M	0.15 $^{+0.10}_{-0.05}$	.006 $^{+.004}_{-.005}$
N	0.15	.006
P	3.7	.146
Q	0.1 ±0.1	.004 ±.004
R	0.1 ±0.1	.004 ±.004
S	4.0 max	.158 max

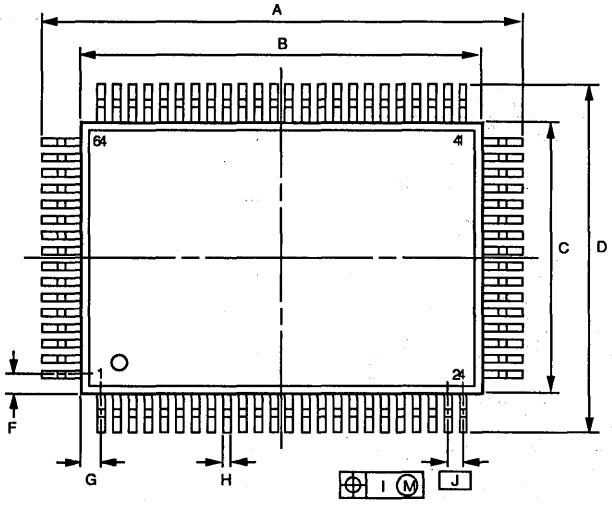


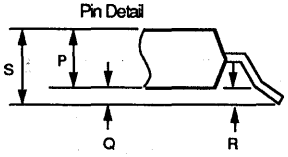
S74GJ-100-5BJ-1

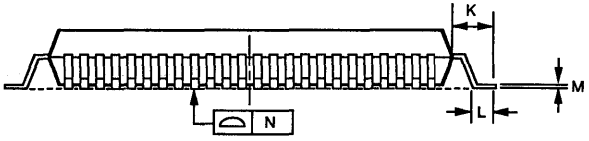
49NR-347B (2/90)

80-Pin Plastic QFP

Item	Millimeters	Inches
A	23.6±0.4	.929±.016
B	20.0±0.2	.787 +.009 −.008
C	14.0±0.2	.551 +.009 −.008
D	17.6±0.4	.693±.016
F	1.0	.039
G	0.8	.031
H	0.35±0.10	.014 +.004 −.005
I	0.15	.006
J	0.8 (TP)	.031 (TP)
K	1.8±0.2	.071 +.009 −.008
L	0.8±0.2	.031 +.009 −.008
M	0.15 +0.10 −0.05	.006 +.004 −.002
N	0.15	.006
P	2.7	.106
Q	0.1±0.1	.004±.004
R	0.1±0.1	.004±.004
S	3.0 max	.118 max





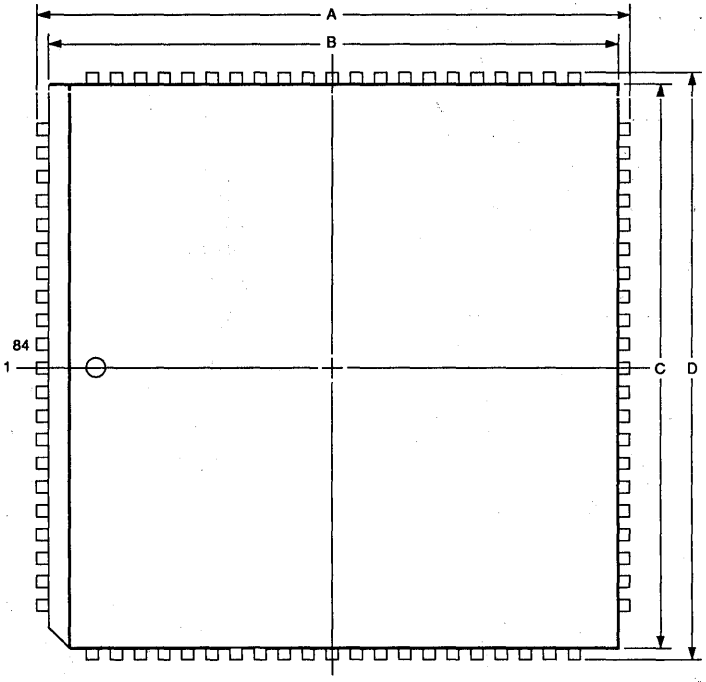


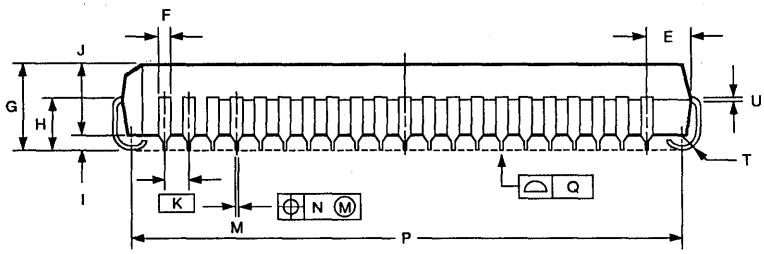
P80GF-80-3B9-1

83H-5543B (2/90)

84-Pin PLCC

Item	Millimeters	Inches
A	30.2 ±0.2	1.189 ±.008
B	29.28	1.153
C	29.28	1.153
D	30.2 ±0.2	1.189 ±.008
E	1.94 ±0.15	.076 ±.006
F	0.6	.024
G	4.4 ±0.2	.173 ±.008
H	2.8 ±0.2	.110 ±.008
I	0.9 min	.035 min
J	3.4	.134
K	1.27 (TP)	.050 (TP)
M	0.40 ±0.10	.016 ±.004
N	0.12	.005
P	28.20 ±0.20	1.110 ±.008
Q	0.15	.006
T	0.8 radius	.031 radius
U	0.20 $^{+0.10}_{-0.05}$	.008 $^{+.004}_{-.002}$



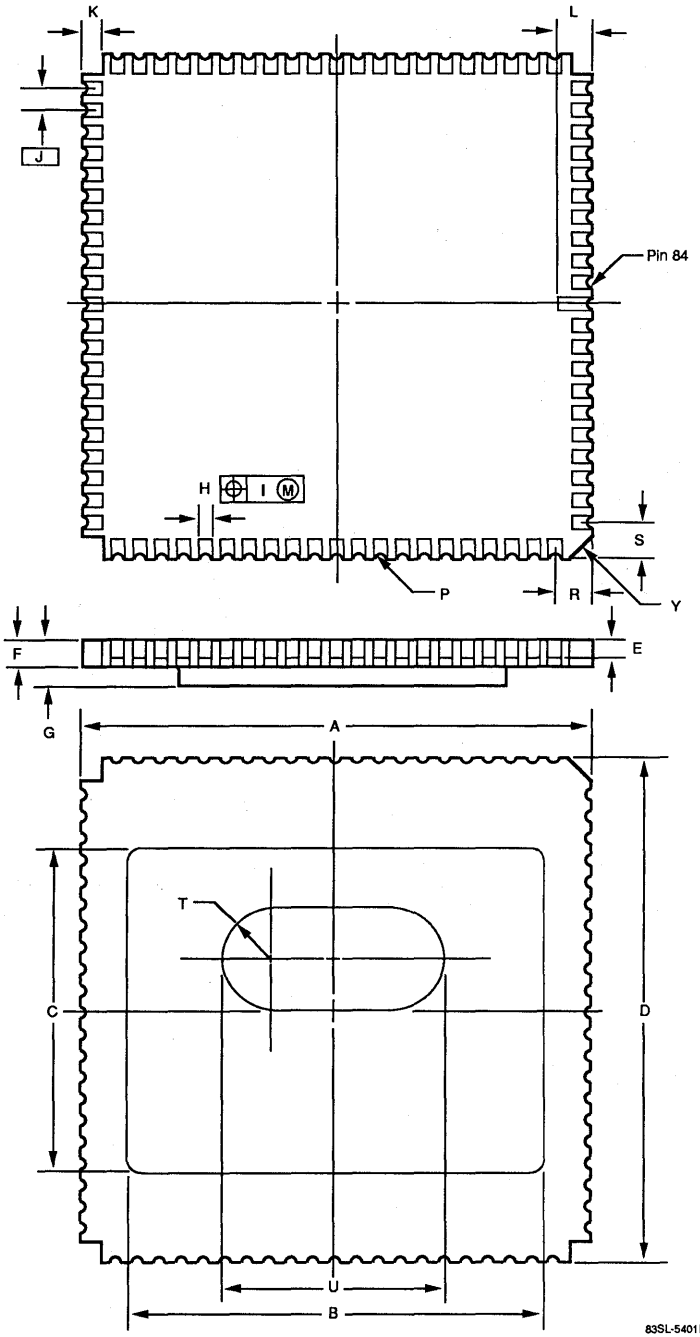


P84L-50A3-1

(2/90)
83YL-5806B

84-Pin Ceramic LCC

Item	Millimeters	Inches
A	29.2±0.4	1.150 $^{+.016}_{-.017}$
B	24.0	.945
C	19.75	.778
D	29.2 ± 0.4	1.150 $^{+.016}_{-.017}$
E	1.28	.050
F	1.66	.065
G	3.556	.140
H	0.91 ± 0.10	.036 $^{+.004}_{-.005}$
I	0.12	.005
J	1.27 (TP)	.050 (TP)
K	1.27 ± 02	.050 ± .008
L	2.54 ± 02	.100 ± .008
P	0.25 radius	.010 radius
R	1.9	.075
S	1.9	.075
T	3.0 radius	.118 radius
U	12.0	.472
Y	0.76 corner	.030 corner



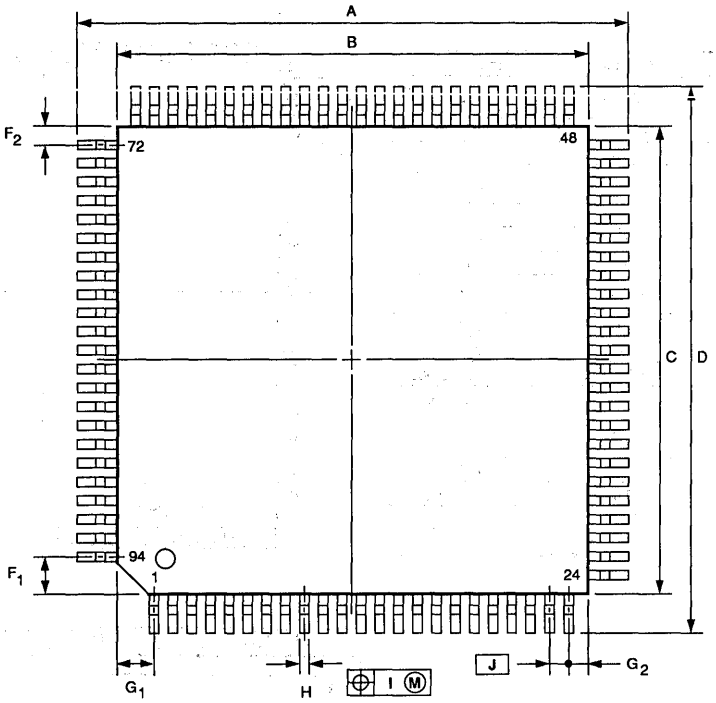
X64KW-50A

83SL-5401B (3/90)

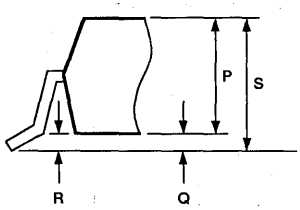
NEC

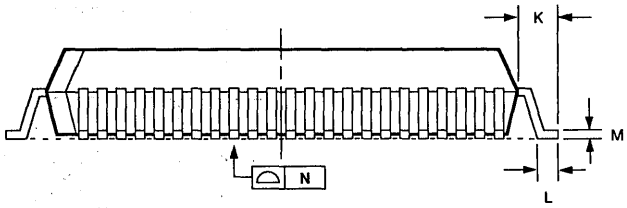
94-Pin Plastic QFP

Item	Millimeters	Inches
A	23.2 ±0.4	.913 +.017 −.016
B	20.0 ±0.2	.787 +.009 −.008
C	20.0 ±0.2	.787 +.009 −.008
D	23.2 ±0.4	.913 +.017 −.016
F_1	1.6	.063
F_2	0.8	.031
G_1	1.6	.063
G_2	0.8	.031
H	0.35 ±0.10	.014 +.004 −.005
I	0.15	.006
J	0.8 (TP)	.031 (TP)
K	1.6 ±0.2	.063 ±.008
L	0.8 ±0.2	.031 +.009 −.008
M	0.15 +0.10 −0.05	.006 +.004 −.003
N	0.15	.006
P	3.7	.146
Q	0.1 ±0.1	.004 ±.004
R	0.1 ±0.1	.004 ±.004
S	4.0 max	.158 max



Detail of lead end



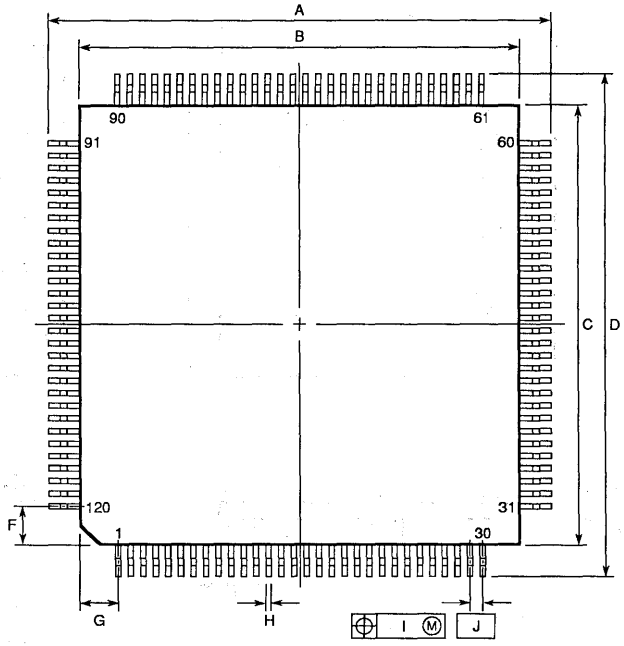


S94GJ-50-5BG-1

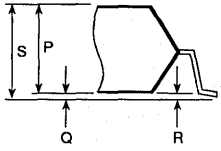
(2/90)
83YL-5810B

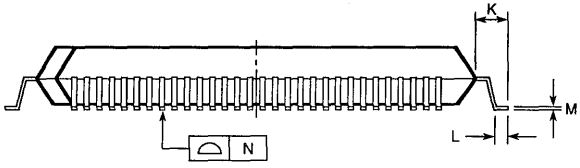
120-Pin Plastic QFP

Item	Millimeters	Inches
A	32.0 ±0.4	1.260 ±.016
B	28.0 ±0.2	1.102 $^{+.009}_{-.008}$
C	28.0 ±0.2	1.102 $^{+.009}_{-.008}$
D	32.0 ±0.4	1.260 ±.016
F	2.4	.094
G	2.4	.094
H	0.35 ±0.10	.014 $^{+.004}_{-.005}$
I	0.15	.006
J	0.8 (TP)	.031 (TP)
K	2.0 ±0.2	.079 $^{+.009}_{-.008}$
L	0.8 ±0.2	.031 $^{+.009}_{-.008}$
M	0.15 $^{+0.10}_{-0.05}$	.006 $^{+.004}_{-.003}$
N	0.15	.006
P	3.7	.146
Q	0.1 ±0.1	.004 ±.004
R	0.1 ±0.1	.004 ±.004
S	4.0 max	.157 max



Enlarged detail of lead end



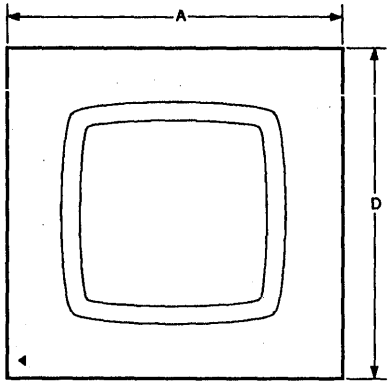


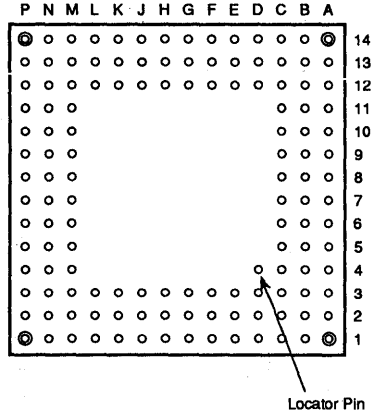
P120GD-80-5BB

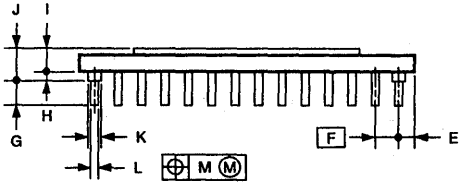
49NR-653B (12/89)

NEC

132-Pin Ceramic PGA







Item	Millimeters	Inches
A	35.56 ±0.4	$1.400^{+.016}_{-.015}$
D	35.56 ±0.4	$1.400^{+.016}_{-.015}$
E	1.27	.050
F	2.54 (TP)	.100 (TP)
G	2.8 ±0.3	$.110^{+.012}_{-.011}$
H	0.5 min	.019 min
I	2.9	.114
J	4.57 max	.180 max
K	ø1.2 ±0.2	$ø.047^{+.008}_{-.007}$
L	ø0.46 ±0.05	$ø.018^{+.002}_{-.001}$
M	0.5	.020

X132R-100A

6/89
83YL-5817B

NEC

Notes:

Notes:

NEC

Notes:

Notes:

Notes:

Notes:

Notes:

NEC

FIELD SALES OFFICES

EASTERN REGION

901 Lake Destiny Drive
Suite 321
Maitland, FL 32751
TEL: 407-875-1145
FAX: 407-875-0962

The Centre at Stirling
and Palm
9900 Stirling Road
Suite 206
Cooper City, FL 33024
TEL: 305-436-8114
FAX: 305-436-8116

6625 The Corners Parkway
Suite 210
Norcross, GA 30092
TEL: 404-447-4409
FAX: 404-447-8228

One Natick Executive Park
Natick, MA 01760
TEL: 508-650-4100
FAX: 508-655-1605

EASTERN REGION [cont]

2525 Meridian Parkway
Suite 320
Durham, NC 27713
TEL: 919-544-4132
FAX: 919-544-4109

200 Perinton Hills
Office Park
Fairport, NY 14450
TEL: 716-425-4590
FAX: 716-425-4594

300 Westage Business Center
Suite 280
Fishkill, NY 12524
TEL: 914-897-2101
FAX: 914-897-2215

Two Jericho Plaza
Jericho, NY 11753
TEL: 516-932-5700
FAX: 516-932-5710

EASTERN REGION [cont]

One Windsor Plaza
One Windsor Drive
Suite B101
Allentown, PA 18195
TEL: 215-391-9094
FAX: 215-391-9107

CENTRAL REGION

1500 West Shure Drive
Suite 250
Arlington Heights, IL 60004
TEL: 708-577-9090
FAX: 708-577-2147

201 E. Big Beaver Road
Suite 350
Troy, MI 48084
TEL: 313-680-0506
FAX: 313-680-1015

1550 East 79th Street
Suite 805
Bloomington, MN 55425
TEL: 612-854-4443
FAX: 612-854-1346

CENTRAL REGION [cont]

1105 Schrock Road
Suite 515
Columbus, OH 43229
TEL: 614-436-1778
FAX: 614-436-1769

30050 Chagrin Blvd.
Suite 320
Pepper Pike, OH 44124
TEL: 216-831-0067
FAX: 216-831-0758

16475 Dallas Parkway
Suite 380
Dallas, TX 75248
TEL: 214-931-0641
FAX: 214-931-1182

12777 Jones Road
Suite 196
Houston, TX 77070
TEL: 713-955-2191
FAX: 713-955-2198

WESTERN REGION

Two Lincoln Center
10220 S.W. Greenburg Road
Suite 125
Portland, OR 97223
TEL: 503-245-1600
FAX: 503-245-3716

WESTERN REGION [cont]

Encino Office Park Two
6345 Balboa Blvd.
Suite 240
Encino, CA 91316
TEL: 818-342-3112
FAX: 818-342-0842

200 E. Sandpointe
Bldg. 8
Suite 150
Santa Ana, CA 92707
TEL: 714-546-0501
FAX: 714-432-8793

14001 East Iliff Avenue
Suite 411
Aurora, CO 80014
TEL: 303-755-6353
FAX: 303-755-6728

NORTHERN CALIFORNIA REGION

401 Ellis Street
P.O. Box 7241
Mountain View, CA 94039
TEL: 415-965-6200
FAX: 415-965-6683

NEC
NEC Electronics Inc.
CORPORATE HEADQUARTERS
401 Ellis Street
P.O. Box 7241
Mountain View, CA 94039
TEL 415-960-6000
TLX 3715792

For literature, call toll-free 8 a.m. to 4 p.m. Pacific time:
1-800-632-3531

50054
©1990 NEC Electronics Inc./Printed in U.S.A.