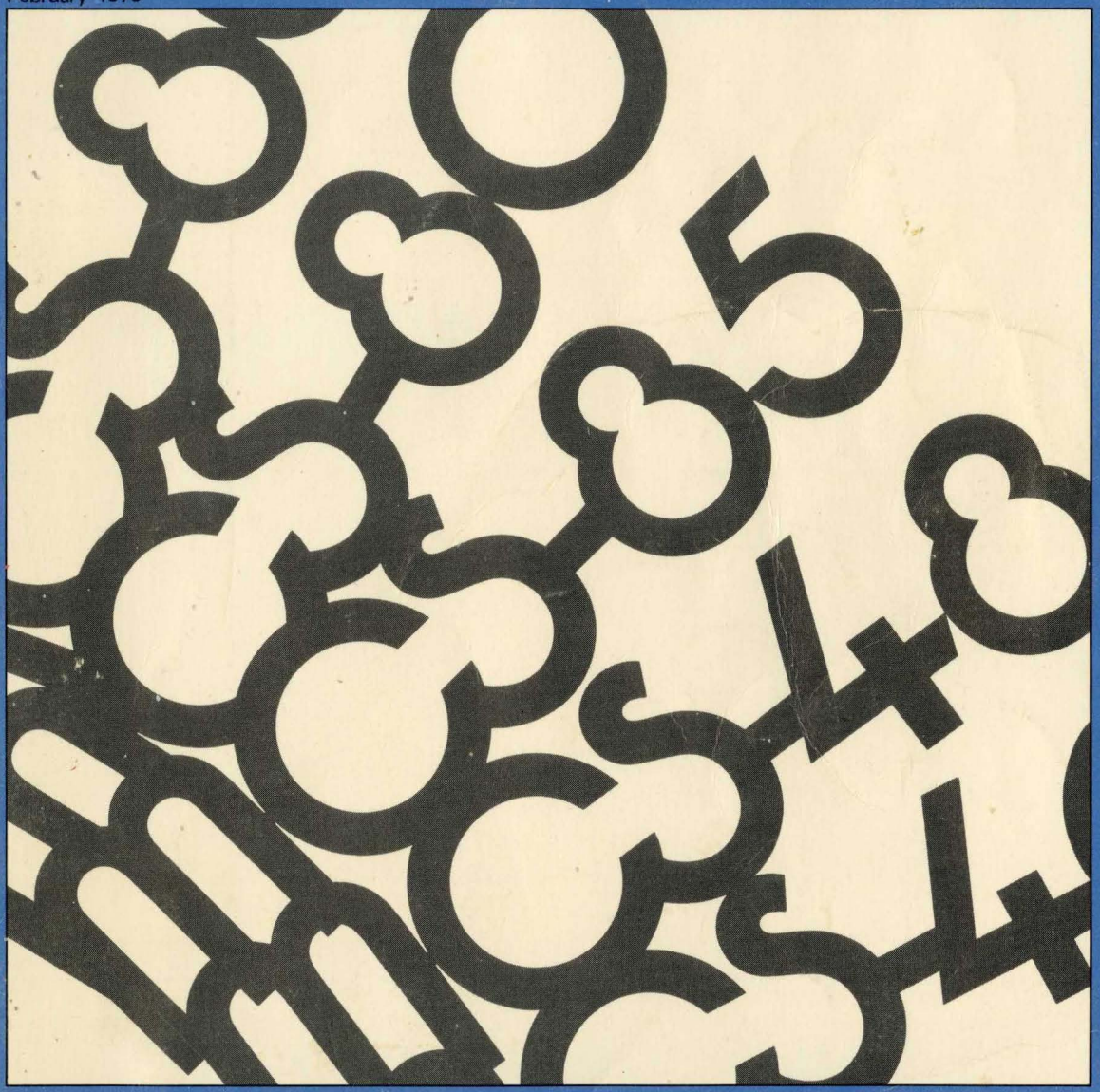
intel®

PERIPHERAL DESIGN HANDBOOK

February 1979



9800676B

intel®

PERIPHERAL DESIGN HANDBOOK

February 1979

SECTION 1
PERIPHERAL
DATA SHEETS

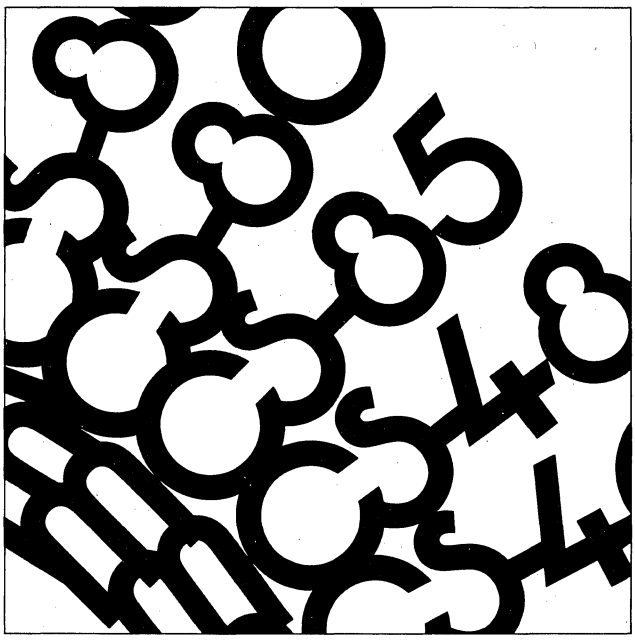


Table of Contents

SECTION 1 — PERIPHERAL DATA SHEETS

SECTION II — APPLICATION NOTES

APPENDIX 1 — ARTICLE REPRINTS

intel®

8041A/8741A
UNIVERSAL PERIPHERAL INTERFACE
8-BIT MICROCOMPUTER

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

- **8-Bit CPU plus ROM, RAM, I/O, Timer and Clock in a Single Package**
- **One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface**
- **DMA, Interrupt, or Polled Operation Supported**
- **1024 × 8 ROM/EPROM, 64 × 8 RAM, 8-Bit Timer/Counter, 18 Programmable I/O Pins**
- **Fully Compatible with MCS-48™, MCS-80™, MCS-85™, and MCS-86™ Microprocessor Families**
- **Expandable I/O**
- **ROM Power-Down Capability**
- **Single 5V Supply**

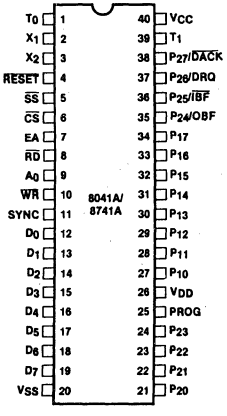
The Intel® 8041A/8741A is a general purpose, programmable interface device designed for use with a variety of 8-bit microprocessor systems. It contains a low cost microcomputer with program memory, data memory, 8-bit CPU, I/O ports, timer/counter, and clock in a single 40-pin package. Interface registers are included to enable the UPI device to function as a peripheral controller in MCS-48™, MCS-80™, MCS-85™, MCS-86™, and other 8-bit systems.

The UPI-41A™ has 1K words of program memory and 64 words of data memory on-chip. To allow full user flexibility the program memory is available as ROM in the 8041A version or as UV-erasable EPROM in the 8741A version. The 8741A and the 8041A are fully pin compatible for easy transition from prototype to production level designs.

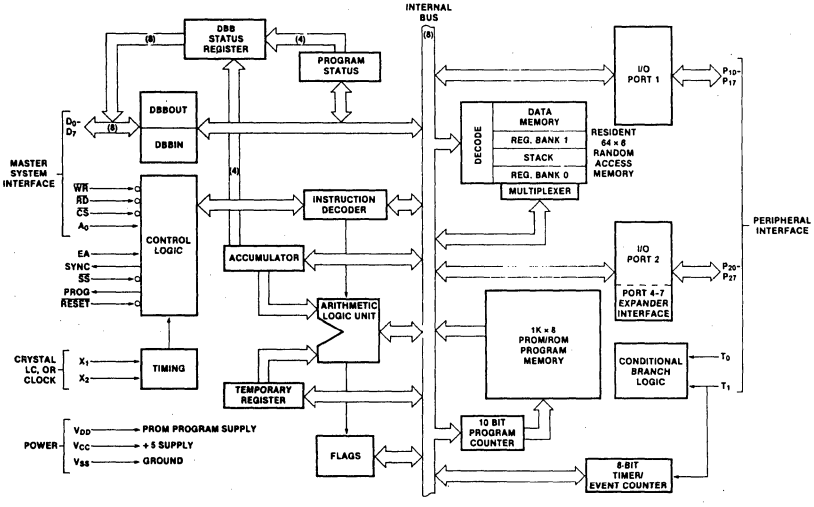
The device has two 8-bit, TTL compatible I/O ports and two test inputs. Individual port lines can function as either inputs or outputs under software control. I/O can be expanded with the 8243 device which is directly compatible and has 16 I/O lines. An 8-bit programmable timer/counter is included in the UPI device for generating timing sequences or counting external inputs. Additional UPI features include: single 5V supply, low power standby mode (in the 8041A), single-step mode for debug (in the 8741A), and dual working register banks.

Because it's a complete microcomputer, the UPI provides more flexibility for the designer than conventional LSI interface devices. It is designed to be an efficient controller as well as an arithmetic processor. Applications include keyboard scanning, printer control, display multiplexing and similar functions which involve interfacing peripheral devices to microprocessor systems.

PIN CONFIGURATION



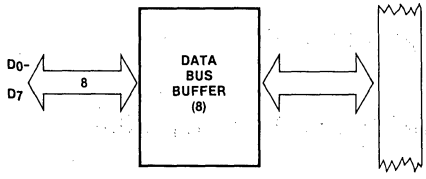
BLOCK DIAGRAM



FEATURES AND ENHANCEMENTS

UPI-41

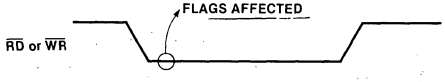
1. Single Data Bus Buffer



2. 4 Bits of Status

UNDEFINED				F_1	F_0	IBF	OBF
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

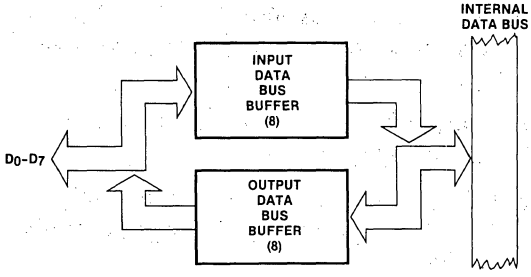
3. $\overline{RD}$ and $\overline{WR}$ are level triggered. IBF, OBF, F_1 and INT change internally when $\overline{RD}$ or $\overline{WR}$ are low.



4. P_{24} and P_{25} are port pins only.

UPI-41A

1. Two Data Bus Buffers, one for input and one for output. This allows a much cleaner Master/Slave protocol.



2. 8 Bits of Status

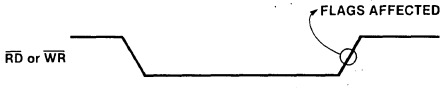
ST_7	ST_6	ST_5	ST_4	F_1	F_0	IBF	OBF
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

ST_4–ST_7 are user definable status bits. These bits are defined by the "MOV STS, A" single byte, single cycle instruction. Bits 4–7 of the accumulator are moved to bits 4–7 of the status register. Bits 0–3 of the status register are not affected.

MOV STS, A Op Code: 90H

1	0	0	1	0	0	0	0
D_7							D_0

3. $\overline{RD}$ and $\overline{WR}$ are edge triggered. IBF, OBF, F_1 and INT change internally after the trailing edge of $\overline{RD}$ or $\overline{WR}$.



4. P_{24} and P_{25} are port pins or Buffer Flag pins which can be used to interrupt a master processor. These pins default to port pins on Reset.

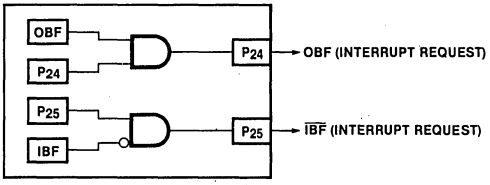
   If the "EN FLAGS" instruction has been executed, P_{24} becomes the OBF (Output Buffer Full) pin. A "1" written to P_{24} enables the OBF pin (the pin outputs the OBF Status Bit). A "0" written to P_{24} disables the OBF pin (the pin remains low). This pin can be used to indicate that valid data is available from the UPI-41A (in Output Data Bus Buffer).

UPI-41

5. P_{26} and P_{27} are port pins only.

UPI-41A

If "EN FLAGS" has been executed, P_{25} becomes the $\overline{IBF}$ (Input Buffer Full) pin. A "1" written to P_{25} enables the $\overline{IBF}$ pin (the pin outputs the inverse of the $\overline{IBF}$ Status Bit). A "0" written to P_{25} disables the $\overline{IBF}$ pin (the pin remains low). This pin can be used to indicate that the UPI-41 is ready for data.



DATA BUS BUFFER INTERRUPT CAPABILITY

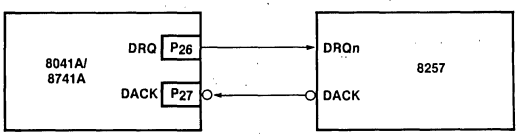
EN FLAGS Op Code: 0F5H

1	1	1	1	0	1	0	1
D_7							D_0

5. P_{26} and P_{27} are port pins or DMA handshake pins for use with a DMA controller. These pins default to port pins on Reset.

If the "EN DMA" instruction has been executed, P_{26} becomes the DRQ (DMA ReQuest) pin. A "1" written to P_{26} causes a DMA request (DRQ is activated). DRQ is deactivated by DACK·RD, DACK·WR, or execution of the "EN DMA" instruction.

If "EN DMA" has been executed, P_{27} becomes the $\overline{DACK}$ (DMA ACKnowledge) pin. This pin acts as a chip select input for the Data Bus Buffer registers during DMA transfers.



DMA HANDSHAKE CAPABILITY

EN DMA Op Code: 0E5H

1	1	1	0	0	1	0	1
D_1							D_0

PIN DESCRIPTION

Signal	Description
D_0-D_7	Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI-41A to an 8-bit master system data bus.
P_{10}-P_{17}	8-bit, PORT 1 quasi-bidirectional I/O lines.
P_{20}-P_{27}	8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits (P_{20}-P_{23}) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access. The upper 4 bits (P_{24}-P_{27}) can be programmed to provide Interrupt Request and DMA Handshake capability. Software control can configure P_{24} as OBF (Output Buffer Full), P_{25} as $\overline{IBF}$ (Input Buffer Full), P_{26} as DRQ (DMA Request), and P_{27} as $\overline{DACK}$ (DMA ACKnowledge).
$\overline{WR}$	I/O write input which enables the master CPU to write data and command words to the UPI-41A INPUT DATA BUS BUFFER.
$\overline{RD}$	I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
$\overline{CS}$	Chip select input used to select one UPI-41A out of several connected to a common data bus.
A_0	Address input used by the master processor to indicate whether byte transfer is data or command.
T_0, T_1	Input pins which can be directly tested using conditional branch instructions. T_1 also functions as the event timer input (under software control). T_0 is used during PROM programming and verification in the 8741A.
X_1, X_2	Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
SYNC	Output signal which occurs once per UPI-41A instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
EA	External access input which allows emulation, testing and PROM/ROM verification.
PROG	Multifunction pin used as the program pulse input during PROM programming. During I/O expander access the PROG pin acts as an address/data strobe to the 8243.
$\overline{RESET}$	Input used to reset status flip-flops and to set the program counter to zero. $\overline{RESET}$ is also used during PROM programming and verification.
$\overline{SS}$	Single step input used in the 8741A in conjunction with the SYNC output to step the program through each instruction.
V_{CC}	+5V power supply pin.
V_{DD}	+5V during normal operation. Programming supply pin during PROM programming. Low power standby pin in ROM version.
V_{SS}	Circuit ground potential.

UPI INSTRUCTION SET

Mnemonic	Description	Bytes	Cycles
ACCUMULATOR			
ADD A,Rr	Add register to A	1	1
ADD A,@Rr	Add data memory to A	1	1
ADD A,#data	Add immediate to A	2	2
ADDC A,Rr	Add immed. to A with carry	1	1
ADDC A,@Rr	Add immed. to A with carry	1	1
ADDC A,#data	Add immed. to A with carry	2	2
ANL A,Rr	AND register to A	1	1
ANL A,@Rr	AND data memory to A	1	1
ANL A,#data	AND immediate to A	2	2
ORL A,Rr	OR register to A	1	1
ORL A,@Rr	OR data memory to A	1	1
ORL A,#data	OR immediate to A	2	2
XRL A,Rr	Exclusive OR register to A	1	1
XRL A,@Rr	Exclusive OR data memory to A	1	1
XRL A,#data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap digits of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
INPUT/OUTPUT			
IN A,Pp	Input port to A	1	2
OUTL Pp,A	Output A to port	1	2
ANL Pp,#data	AND immediate to port	2	2
ORL Pp,#data	OR immediate to port	2	2
IN A,DBB	Input DBB to A, clear IBF	1	1
OUT DBB,A	Output A to DBB, set OBF	1	1
MOVD A,Pp	Input Expander port to A	1	2
MOVD Pp,A	Output A to Expander port	1	2
ANLD Pp,A	AND A to Expander port	1	2
ORLD Pp,A	OR A to Expander port	1	2
DATA MOVES			
MOV A,Rr	Move register to A	1	1
MOV A,@Rr	Move data memory to A	1	1
MOV A,#data	Move immediate to A	2	2
MOV Rr,A	Move A to register	1	1
MOV @Rr,A	Move A to data memory	1	1
MOV Rr,#data	Move immediate to register	2	2
MOV @Rr,#data	Move immediate to data memory	2	2
MOV A,PSW	Move PSW to A	1	1
MOV PSW,A	Move A to PSW	1	1
XCH A,Rr	Exchange A and register	1	1
XCH A,@Rr	Exchange A and data memory	1	1
XCHD A,@Rr	Exchange digit of A and register	1	1
MOVP A,@A	Move to A from current page	1	2
MOVP3, A,@A	Move to A from page 3	1	2
TIMER/COUNTER			
MOV A,T	Read Timer/Counter	1	1
MOV T,A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1

Mnemonic	Description	Bytes	Cycles
CONTROL			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
REGISTERS			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1
SUBROUTINE			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
FLAGS			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
MOV STS, A	A_4-A_7 to Bits 4-7 of Status	1	1
BRANCH			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R,addr	Decrement register and skip	2	2
JC addr	Jump on Carry = 1	2	2
JNC addr	Jump on Carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 Flag = 1	2	2
JF1 addr	Jump on F1 Flag = 1	2	2
JTF addr	Jump on Timer Flag = 1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag = 0	2	2
JOBF addr	Jump on OBF Flag = 1	2	2
JBb addr	Jump on Accumulator Bit	2	2

APPLICATIONS

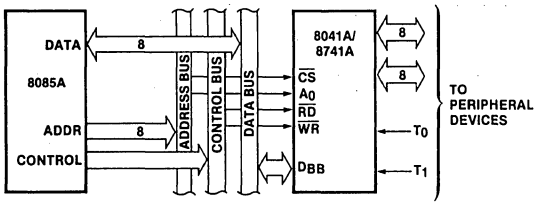


Figure 1. 8085A-8041A Interface

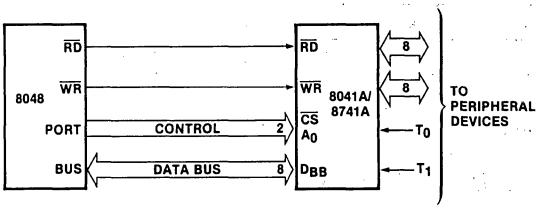


Figure 2. 8048-8041A Interface

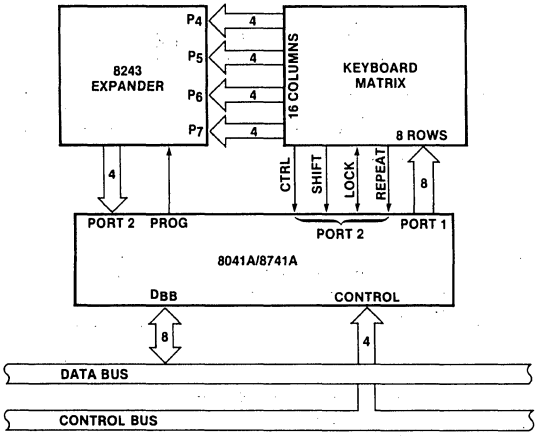


Figure 3. 8041A-8243 Keyboad Scanner

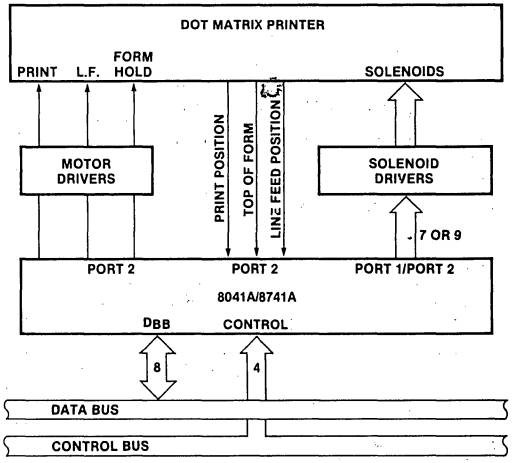


Figure 4. 8041A Matrix Printer Interface

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . – 65°C to + 150°C
Voltage on Any Pin With Respect
to Ground . . . . . 0.5V to + 7V
Power Dissipation . . . . . 1.5 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to 70°C, $V_{SS} = 0V$, 8041A: $V_{CC} = +5V \pm 10\%$, 8741A: $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage (All Except X_1, X_2)	-0.5	0.8	V	
V_{IH1}	Input High Voltage (All Except X_1, X_2, $\overline{RESET}$)	2.2	V_{CC}		
V_{IH2}	Input High Voltage (X_1, X_2, $\overline{RESET}$)	3.0	V_{CC}	V	
V_{OL1}	Output Low Voltage (D_2-D_7, Sync)		0.45	V	$I_{OL} = 2.0$ mA
V_{OL2}	Output Low Voltage (All Other Outputs Except Prog)		0.45	V	$I_{OL} = 1.6$ mA
V_{OL3}	Output Low Voltage (Prog)		0.45	V	$I_{OL} = 1.0$ mA
V_{OH1}	Output High Voltage (D_0-D_7)	2.4		V	$I_{OH} = -400\ \mu A$
V_{OH2}	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50\ \mu A$
I_{IL}	Input Leakage Current (T_0, T_1, $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0, EA)		± 10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OZ}	Output Leakage Current (D_0-D_7, High Z State)		± 10	μA	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$
I_{LI1}	Low Input Load Current ($P_{10}P_{17}$, $P_{20}P_{27}$		0.5	mA	$V_{IL} = 0.8V$
I_{LI2}	Low Input Load Current ($\overline{RESET}$, SS)		0.2	mA	$V_{IL} = 0.8V$
I_{DD}	V_{DD} Supply Current		1.5	mA	
$I_{CC} + I_{DD}$	Total Supply Current		125	mA	

A.C. CHARACTERISTICS

$T_A = 0°C$ to 70°C, $V_{SS} = 0V$, 8041A: $V_{CC} = V_{DD} = +5V \pm 10\%$, 8741A: $V_{CC} = V_{DD} = +5V \pm 5\%$

DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	$t_{CY} = 2.5\ \mu s$
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RD}	$\overline{RD}$↓ to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RDF}	$\overline{RD}$↑ to Data Float Delay		100	ns	
t_{RV}	Recovery Time Between Reads And/Or Write	300		ns	
t_{CY}	Cycle Time	2.5	15	μs	

DBB WRITE

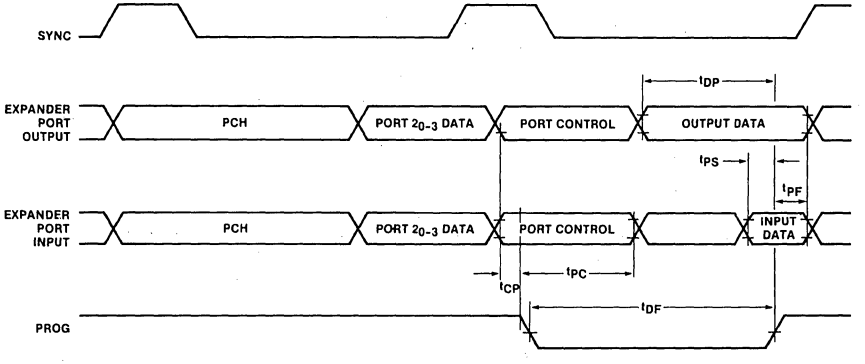
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	CS, A_0 Setup to WR↓	0		ns	
t_{WA}	CS, A_0 Hold After WR↑	0		ns	
t_{WW}	WR Pulse Width	250		ns	
t_{DW}	Data Setup to WR↑	150		ns	
t_{WD}	Data Hold Aftert WR↑	0		ns	

A.C. CHARACTERISTICS—PORT 2

$T_A = 0°C$ to $70°C$, 8041A: $V_{CC} = +5V \pm 10\%$, 8741A: $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CP}	Port Control Setup Before Falling Edge of $\overline{PROG}$	110		ns	
t_{PC}	Port Control Hold After Falling Edge of $\overline{PROG}$	140		ns	
t_{DP}	Output Data Setup Time	220		ns	
t_{PF}	Input Data Hold Time	110		ns	
t_{PP}	$\overline{PROG}$ Pulse Width	1400		ns	
t_{PS}	Input Data Setup Time	700		ns	

WAVEFORMS—PORT 2

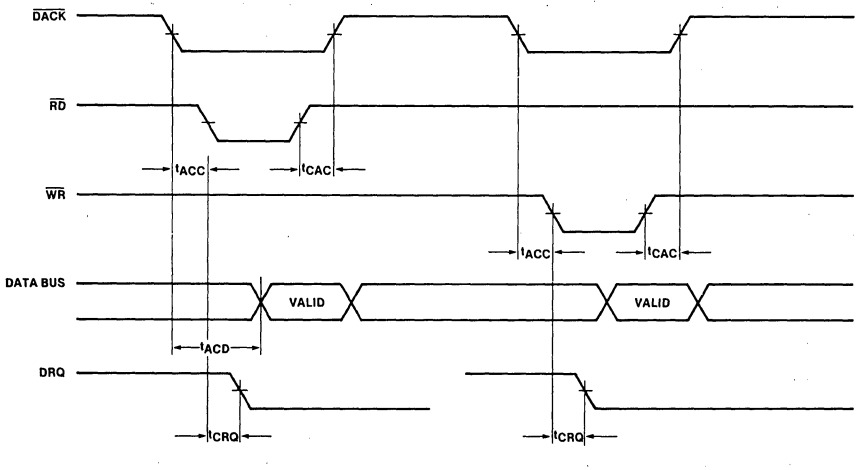


A.C. CHARACTERISTICS—DMA

$T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{ACC}	DAC to $\overline{WR}$ or $\overline{RD}$	0		ns	
t_{CAC}	$\overline{RD}$ or $\overline{WR}$ to $\overline{DACK}$	0		ns	
t_{ACD}	$\overline{DACK}$ to Data Valid		225	ns	$C_L = 150$ pF
t_{CRQ}	$\overline{RD}$ or $\overline{WR}$ to DRQ Cleared		200	ns	

WAVEFORMS—DMA

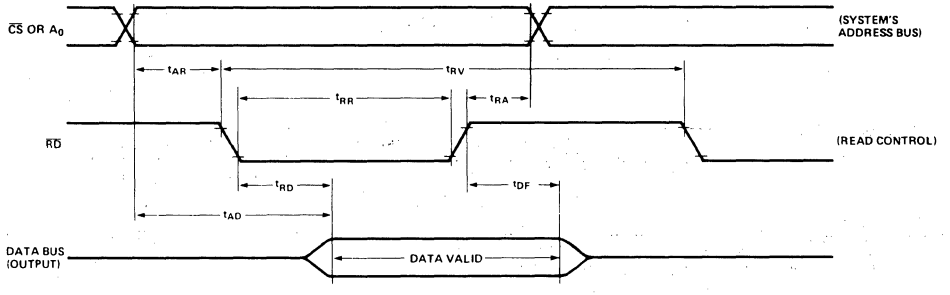


A.C. TEST CONDITIONS

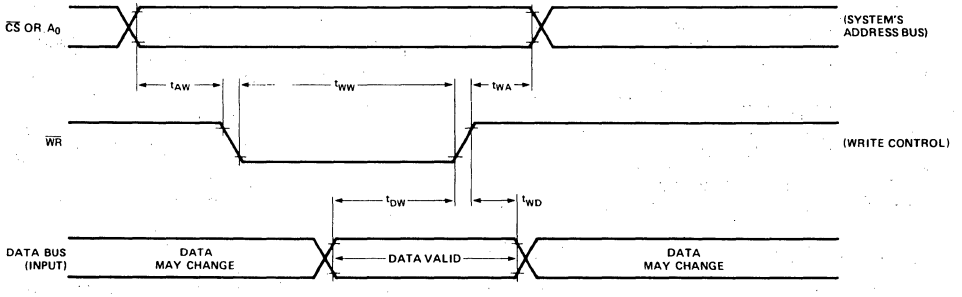
D_7-D_0 Outputs R_L = 2.2k to V_{SS}
4.3k to V_{CC}
C_L = 100 pF

WAVEFORMS

1. READ OPERATION—DATA BUS BUFFER REGISTER.



2. WRITE OPERATION—DATA BUS BUFFER REGISTER.



PROGRAMMING, VERIFYING, AND ERASING THE 8741A EPROM

Programming/Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	Clock input (1 to 6 MHz)
RESET	Initialization and address latching
TEST 0	Selection of program or verify mode
EA	Activation of program/verify modes
BUS	Address and data input data output during verify
P20-1	Address input
V_{DD}	Programming power supply
PROG	Program pulse input

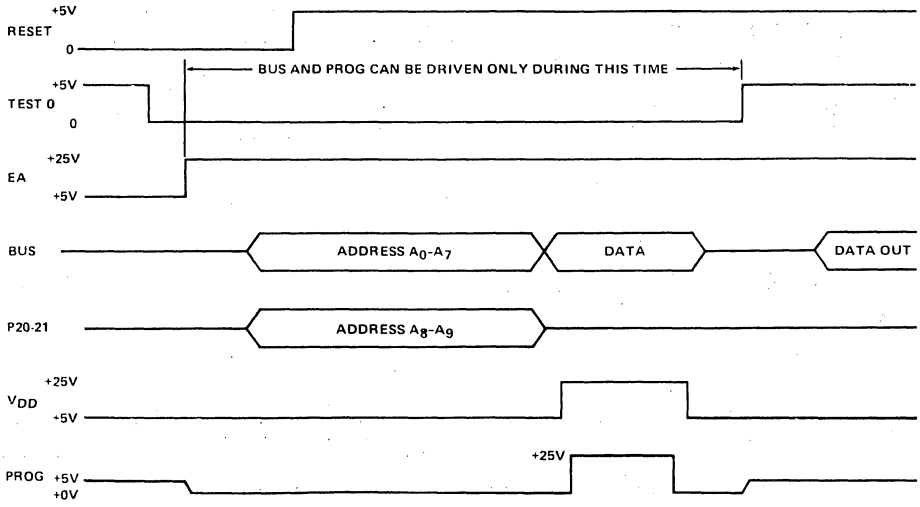
The program/verify sequence is:

1. V_{DD} = 5V, clock applied or internal oscillator operating, RESET = 0V, TEST 0 = 5V, EA = 5V, BUS and PROG floating.
2. Insert 8741A in programming socket.
3. TEST 0 = 0V (select program mode).
4. EA = 25V (activate program mode).
5. Address applied to BUS and P20-1.
6. RESET = 5V (latch address).
7. Data applied to BUS.
8. V_D = 25V (programming power).
9. PROG = 0V followed by one 50 ms pulse to 25V.
10. V_{DD} = 5V.
11. TEST 0 = 5V (verify mode).
12. Read and verify data on BUS.
13. TEST 0 = 0V.
14. RESET = 0V and repeat from step 5.
15. Programmer should be at conditions of step 1 when 8741A is removed from socket.

Programming Options

The 8741A EPROM can be programmed by either of two Intel products:

1. PROMPT-48 Microcomputer Design Aid.
2. Universal PROM Programmer (UPP-101 or UPP-102) Peripheral of the Intellec® Development System with a UPP-848 Personality Card.



WARNING: An attempt to program a missocketed 8741A will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC output. The lack of this clock may be used to disable the programmer.

Figure 5. Programming/Verification Sequence

8741A Erasure Characteristics

The erasure characteristics of the 8741A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8741A in approximately 3 years while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 8741A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8741A window to prevent unintentional erasure.

The recommended erasure procedure for the 8741 is exposure to shortwave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of 15 w-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm^2 power rating. The 8741A should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

A.C. TIMING SPECIFICATION FOR PROGRAMMING

$T_A = 25°C \pm 5°C$, $V_{CC} = 5V \pm 5\%$, $V_{DD} = 25V \pm 1V$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	Address Setup Time to $\overline{RESET}$ ↑	4tcy			
t_{WA}	Address Hold Time After $\overline{RESET}$ ↑	4tcy			
t_{DW}	Data in Setup Time to PROG ↑	4tcy			
t_{WD}	Data in Hold Time After PROG ↓	4tcy			
t_{PH}	$\overline{RESET}$ Hold Time to Verify	4tcy			
t_{VDDW}	V_{DD}	4tcy			
t_{VDDH}	V_{DD} Hold Time After PROG ↓	0			
t_{PW}	Program Pulse Width	50	60	MS	
t_{TW}	Test 0 Setup Time for Program Mode	4tcy			
t_{WT}	Test 0 Hold Time After Program Mode	4tcy			
t_{DO}	Test 0 to Data Out Delay		4tcy		
t_{WW}	$\overline{RESET}$ Pulse Width to Latch Address	4tcy			
t_r, t_f	V_{DD} and PROG Rise and Fall Times	0.5	2.0	μs	
t_{CY}	CPU Operation Cycle Time	5.0		μs	
t_{RE}	$\overline{RESET}$ Setup Time Before EA ↑	4tcy			

Note: If TEST 0 is high, t_{DO} can be triggered by $\overline{RESET}$ 1.

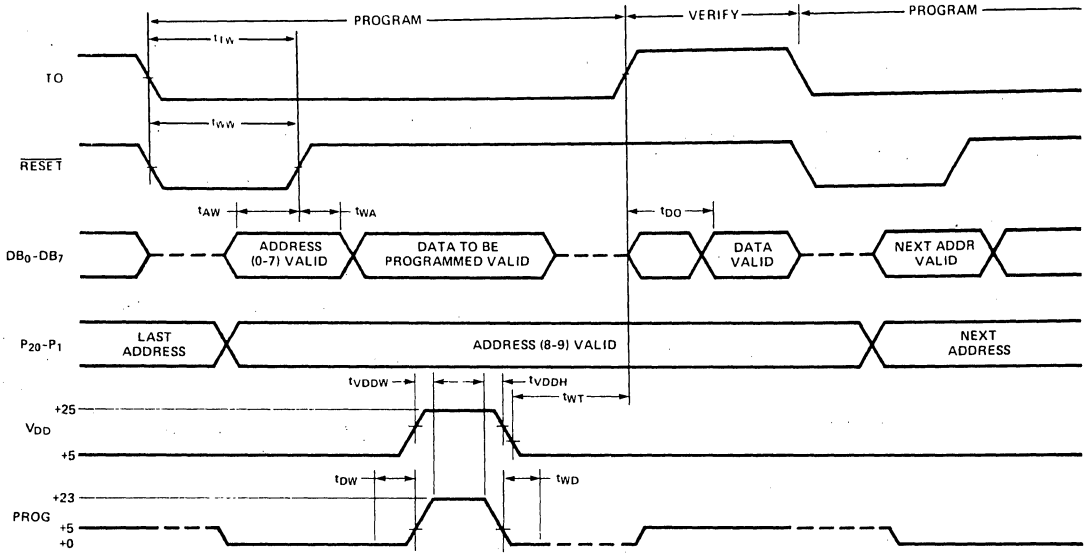
D.C. SPECIFICATION FOR PROGRAMMING

$T_A = 25°C \pm 5°C$, $V_{CC} = 5V \pm 5\%$, $V_{DD} = 25V \pm 1V$

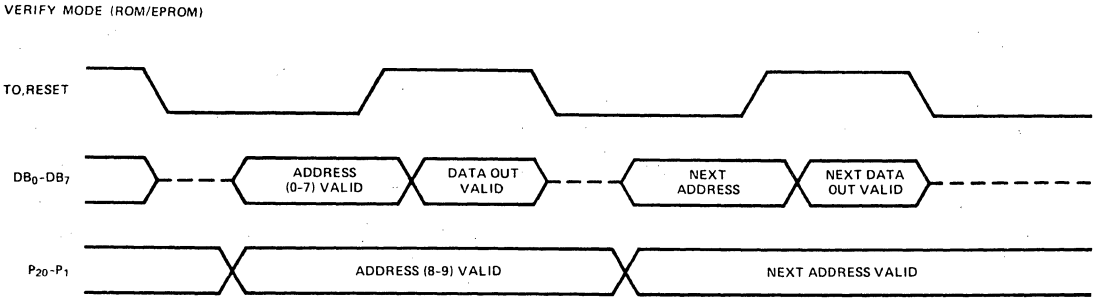
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{DOH}	V_{DD} Program Voltage High Level	24.0	26.0	V	
V_{DDL}	V_{DD} Voltage Low Level	4.75	5.25	V	
V_{PH}	PROG Program Voltage High Level	21.5	24.5	V	
V_{PL}	PROG Voltage Low Level		0.2	V	
V_{EAH}	EA Program or Verify Voltage High Level	21.5	24.5	V	
V_{EAL}	EA Voltage Low Level		5.25	V	
I_{DD}	V_{DD} High Voltage Supply Current		30.0	mA	
I_{PROG}	PROG High Voltage Supply Current		16.0	mA	
I_{EA}	EA High Voltage Supply Current		1.0	mA	

WAVEFORMS

Combination Program/Verify Mode (EPROMs Only)



Verify Mode (ROM/EPROM)



intel®

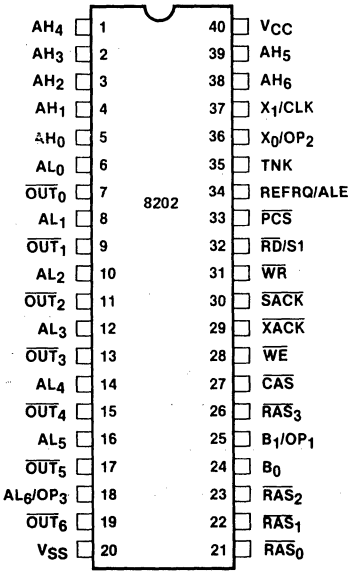
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8202
DYNAMIC RAM CONTROLLER

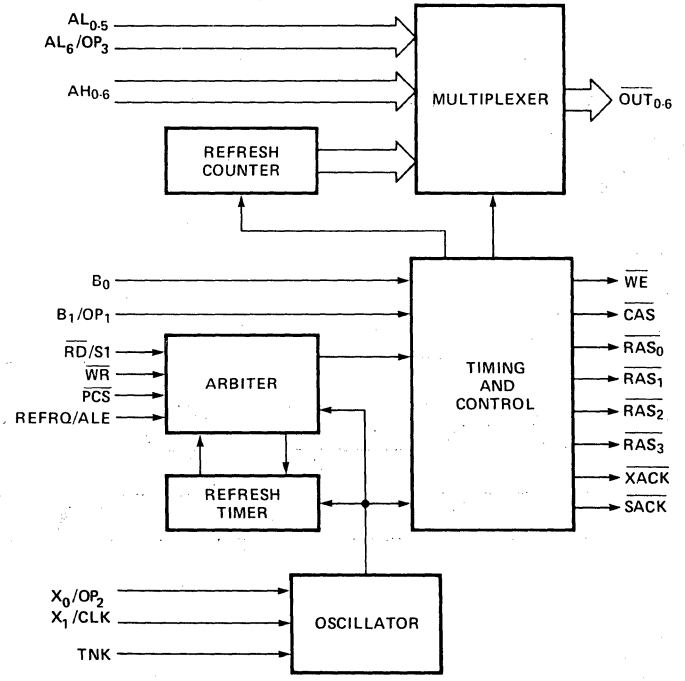
- **Provides All Signals Necessary to Control 2104A, 2117, or 2118 Dynamic Memories**
- **Directly Addresses and Drives Up to 128K Bytes Without External Drivers**
- **Provides Address Multiplexing and Strobes**
- **Provides a Refresh Timer and a Refresh Counter**
- **Refresh Cycles May be Internally or Externally Requested**
- **Provides Transparent Refresh Capability**
- **Fully Compatible with Intel® 8080A, 8085A and 8086 Microprocessors**
- **Decodes 8085A Status for Advanced Read Capability**
- **Provides System Acknowledge and Transfer Acknowledge Signals**
- **Internal or External Clock Capability**

The 8202 is a Dynamic RAM System Controller designed to provide all signals necessary to use 2104A, 2117, or 2118 Dynamic RAMs in microcomputer systems. The 8202 provides multiplexed addresses and address strobes, as well as refresh/access arbitration. Refresh cycles can be started internally or externally.

PIN CONFIGURATION



8202 BLOCK DIAGRAM



PIN DESCRIPTIONS

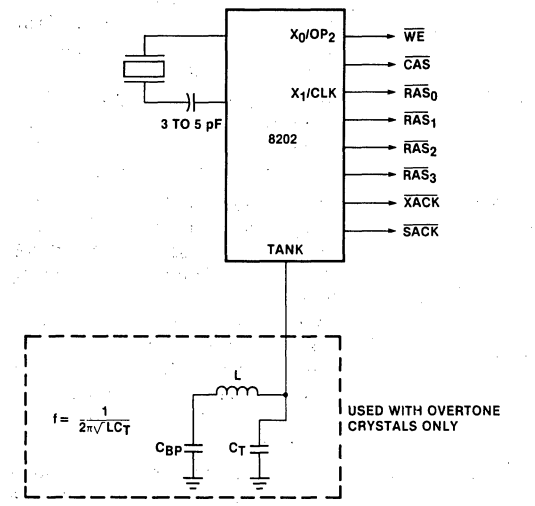
Pin Name	#	I/O	Pin Description
AL_0	6	I	Low-Order Address. These Address inputs are used to generate the Row Address for the Multiplexer. If the AL_6/OP_3 input is pulled to +12V through a 5KΩ resistor, the 8202 configures itself for 4K RAMs. If AL_6/OP_3 is driven with TTL levels, the 8202 configures itself for 16K RAMs.
AL_1	8	I	
AL_2	10	I	
AL_3	12	I	
AL_4	14	I	
AL_5	16	I	
AL_6/OP_3	18	I	
AH_0	5	I	High-Order Address. These Address inputs are used to generate the Column Address for the Multiplexer. If the 8202 is configured for 4K RAMs, AH_6 can be used as an active high Chip select for the memory controlled by 8202. For 16K RAM operation, AH_6 becomes the most significant column address bit.
AH_1	4	I	
AH_2	3	I	
AH_3	2	I	
AH_4	1	I	
AH_5	39	I	
AH_6	38	I	
$\overline{OUT_0}$	7	O	Output of the Multiplexer. These outputs are designed to drive the addresses of the Dynamic RAM array. For 4K RAM operation, $\overline{OUT_6}$ is designed to drive the 2104A $\overline{CS}$ input. (Note that the $\overline{OUT_{0-6}}$ pins do not require inverters or drivers for proper operation.
$\overline{OUT_1}$	9	O	
$\overline{OUT_2}$	11	O	
$\overline{OUT_3}$	13	O	
$\overline{OUT_4}$	15	O	
$\overline{OUT_5}$	17	O	
$\overline{OUT_6}$	19	O	
$\overline{WE}$	28	O	Write Enable. This output is designed to drive the Write Enable inputs of the Dynamic RAM array.
$\overline{CAS}$	27	O	Column Address Strobe. This output is used to latch the Column Address into the Dynamic RAM array.
$\overline{RAS_0}$	21	O	Row Address Strobe. These outputs are used to latch the Row Address into the bank of dynamic RAMs, selected by the 8202 Bank Address pins (B_0, B_1/OP_1)
$\overline{RAS_1}$	22	O	
$\overline{RAS_2}$	23	O	
$\overline{RAS_3}$	26	O	
B_0	24	I	Bank Address. These inputs are used to select one of four banks of dynamic RAM via the $\overline{RAS_{0-3}}$ outputs. If the B_1/OP_1 input is pulled to +12V through a 5KΩ resistor, the 8202 configures itself to the Advanced Read mode. This mode changes the function of the 8202 $\overline{RD}/S_1$ and $\overline{REFRQ}/ALE$ inputs and disables the $\overline{RAS_0}$ and $\overline{RAS_1}$ outputs.
B_1/OP_1	25	I	
$\overline{RD}/S_1$	32	I	Read/S_1 input. This input is used to request a read cycle. In normal operation, a low on this input informs the arbiter that a read cycle is requested. In the Advanced Read Mode, this input is designed to accept the S1 status signal from the 8085A (fully decoded for a read). The trailing edge of ALE informs the arbiter that a read cycle is requested by latching S_1.
$\overline{WR}$	31	I	Write Input. This input is used to request a write cycle. A low on this input informs the arbiter that a write cycle is desired.
$\overline{PCS}$	33	I	Protected Chip Select. A low on this input enables the $\overline{WR}$ and $\overline{RD}/S_1$ inputs. $\overline{PCS}$ is protected against terminating a cycle in progress.
REFRQ/ALE	34	I	Refresh Request/Address Latch Enable. During normal operation, a high on this input indicates to the arbiter that a refresh cycle is being requested. In the Advanced Read Mode, this input is used to latch the state of the 8085 S1 signal into the $\overline{RD}/S_1$ input. If S_1 is high at this time, a Read Cycle is requested. In this mode, transparent refresh is not possible.
$\overline{XACK}$	29	O	Transfer acknowledge. This output is a strobe indicating valid data during a read cycle or data written during a write cycle. $\overline{XACK}$ can be used to latch valid data from the RAM array.
$\overline{SACK}$	30	O	System Acknowledge. This output indicates the beginning of a memory access cycle. It can be used as an advanced transfer acknowledge to eliminate wait states. (Note: If a memory access request is made during a refresh cycle, $\overline{SACK}$ is delayed until $\overline{XACK}$ in the memory access cycle).
X_0/OP_2	36	I	Crystal Inputs. These inputs are designed for a quartz crystal to control the frequency of the oscillator. If X_0/OP_2 is pulled to +12V through a 1KΩ resistor, X_1/CLK becomes a TTL input for an external clock.
X_1/CLK	37	I	
TNK	35		Tank. This pin is used for a tank circuit connection.
V_{CC}	40		+5V ± 10%
V_{SS}	20		Ground.

BASIC FUNCTIONAL DESCRIPTION

The 8202 consists of six basic blocks; the oscillator, the arbiter, the refresh timer, the refresh counter, the multiplexer, and the timing and control block.

Oscillator

The oscillator provides the basic timing for all 8202 operations. The oscillator circuit is designed primarily for use with an external series resonant fundamental mode crystal. Overtone crystals may be used with the tank circuit shown in Figure 1. A small capacitor (3-5) pF should be placed in series with any crystal to block D.C. stress and assure oscillation at the proper frequency.



The tank input to the oscillator allows the use of overtone mode crystals. The tank circuit shunts the crystal's fundamental and high overtone frequencies and allows the third harmonic to oscillate. The external LC network is connected to the TANK input and is AC coupled to ground.

If the X_0/OP_2 pin is pulled to +12V, through a 1KΩ resistor, the 8202 can be driven by a TTL clock on the X_1/CLK input. No tank circuit is required in this mode.

Arbiter

The 8202 provides 3 different operational cycles:

1. Read Cycle
2. Write Cycle
3. Refresh Cycle

The read and write cycles are initiated by external requests ($\overline{RD}/S_1$ and $\overline{PCS}$ or $\overline{WR}$ and $\overline{PCS}$). A refresh cycle may be initiated by the internal refresh timer, or by an external request (REFRQ/ALE). The arbiter resolves conflicts between cycle requests and cycles in execution.

If the B_1/OP_1 input is pulled to +12V through a 5KΩ resistor (Advanced Read mode), $\overline{RD}/S_1$ becomes an input for the S_1 status signal of the 8085A (fully decoded for read). REFRQ/ALE becomes an input for the ALE signal of the 8085 (used to latch S_1. If S_1 is "high" at the falling edge of ALE, a read cycle will be requested. Transparent refresh is not possible in this mode.

Refresh Timer

The refresh timer is a simple timer that indicates to the arbiter that it is time for a refresh cycle. The refresh timer is reset when a refresh cycle is requested.

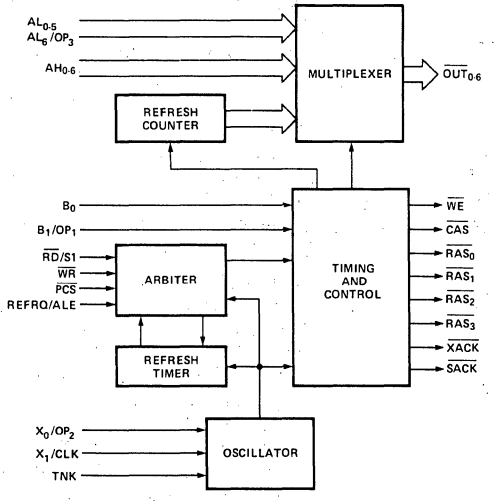
Refresh Counter

The refresh counter contains the address of the row to be refreshed. This counter is incremented after every refresh cycle.

Multiplexer

The multiplexer is designed to provide the dynamic RAM array with row addresses, column addresses and refresh addresses at the proper times. Its inputs consist of AL_{0-5}, AL_6/OP_3, AH_{0-6}, and the refresh counter.

If AL_6/OP_3 is pulled to +12V through a 5KΩ resistor, the 8202 configures itself for 4K RAMs. In this mode, AL_{0-5} provides the multiplexer with the six bit row address. AH_{0-5} provides the multiplexer with the six bit column address.



$\overline{OUT}_{0-5}$ provide the RAM array with twelve bits of multiplexed address. AH_6 can be used as an active high chip select for the RAM array if $\overline{OUT}_6$ drives $\overline{CS}$. Note that the $\overline{OUT}_{0-6}$ signals do not require inverters or drivers.

If the 8202 is configured for 16K RAMs, AL_{0-5} and AL_6/OP_3 provide the multiplexer with seven bits of row

address. AH_{0-6} provides it with seven bits of column address. $\overline{OUT}_{0-6}$ provides the RAM array with fourteen bits of multiplexed address.

Timing and Control Block

The timing and control block executes one of three operational cycles at the request of the arbiter (Read, Write, and Refresh cycles). It provides the RAM array with $\overline{WE}$, $\overline{CAS}$, and $\overline{RAS}$ signals. It provides the CPU with transfer and system acknowledge ($\overline{XACK}$ and $\overline{SACK}$) signals. It controls the multiplexer during all cycles. It resets the refresh timer and increments the refresh counter during refresh cycles.

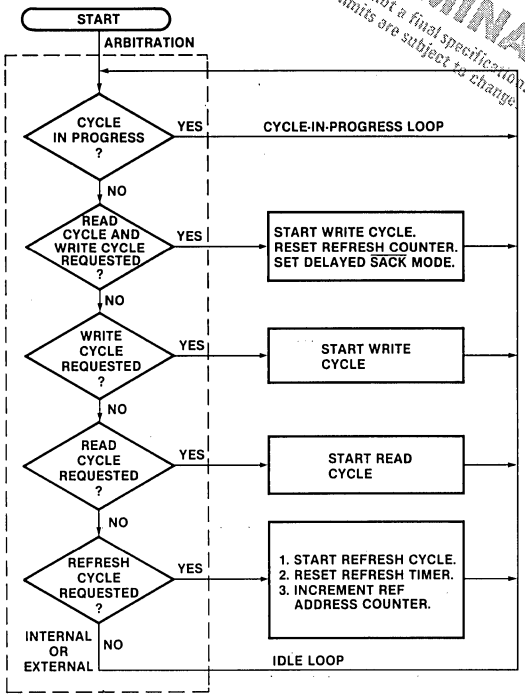
Inputs B_0 and B_1/OP_1 are used to select one of four banks of dynamic RAM via the $\overline{RAS}_{0-3}$ outputs.

If B_1/OP_1 is pulled to +12V through a 5KΩ resistor, the 8202 configures itself to the Advanced Read Mode. This mode changes the function of the $\overline{RD}/S_1$ and REFRQ/ALE inputs and disables the $\overline{RAS}_0$ and $\overline{RAS}_1$ outputs.

SYSTEM OPERATION

The 8202 is always in one of the following states:

1. Idle.
2. Performing a Test Cycle.
3. Performing a Write Cycle.
4. Performing a Read Cycle.
5. Performing a Refresh Cycle.



Idle

When the 8202 is idle, no cycle is in progress, the arbiter monitors internal and external cycle requests, and the refresh timer counts towards an internal refresh cycle request. (Fig. X.1)

While the 8202 is idle, the arbiter samples access cycle requests and refresh cycle requests, internal or external, on the rising edge of clock. If both Read and Write cycle requests are active when sampled, a test cycle is started. If a write-cycle request is active when sampled, a write cycle is started. If a read cycle request is active when sampled, a read cycle is started. If a refresh cycle request was previously pulsed or is active when sampled, a refresh cycle is started. Due to internal delays, if an access cycle request and a refresh cycle request occur simultaneously, the access cycle will be executed before the refresh cycle is executed.

Test Cycle

When a test cycle is started, (Read and Write Cycle Requests both active when sampled) the refresh counter is set to zero and the delayed $\overline{SACK}$ mode is reset, while the 8202 executes a write cycle. This cycle is used for testing only and is not recommended for normal system operation.

Write Cycle (Fig. X.2)

When a write cycle is started, (Write-Cycle Request active when sampled) the Multiplexer drives the $\overline{OUT}$ 0-6 pins with the low order address. Then, if the delayed $\overline{SACK}$ Mode is not set, $\overline{SACK}$ is activated. The row address is strobed into the selected bank of RAMs. The multiplexer then drives the $\overline{OUT}$ 0-6 pins with the high order address and the write enable ($\overline{WE}$) pin is activated. The column address is then strobed into the RAM array.

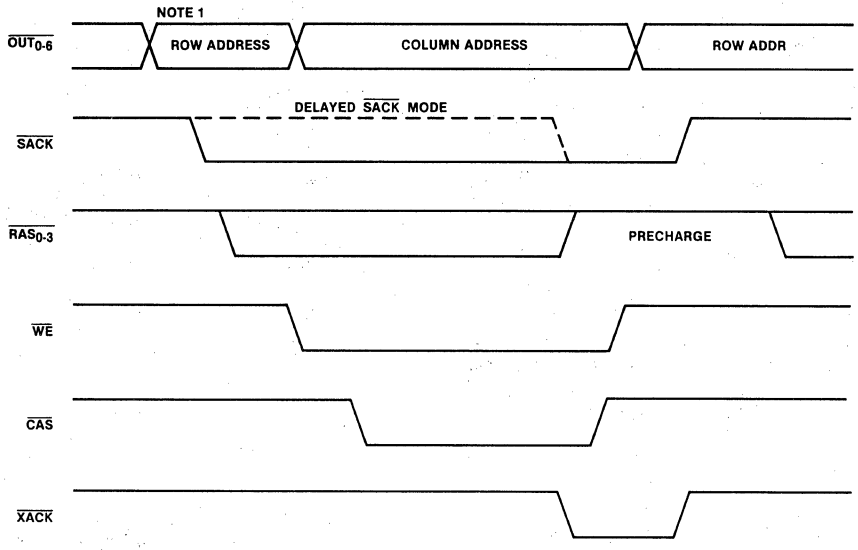
Near the end of the cycle, the $\overline{XACK}$ output is activated. If the Delayed $\overline{SACK}$ Mode is set, $\overline{SACK}$ had the same timing as $\overline{XACK}$. At the end of the cycle, all signals are deactivated, the Delayed $\overline{SACK}$ Mode is exited, and the precharge time begins. After the precharge time, the 8202 re-enters the idle state. The refresh timer continues to count during access cycles.

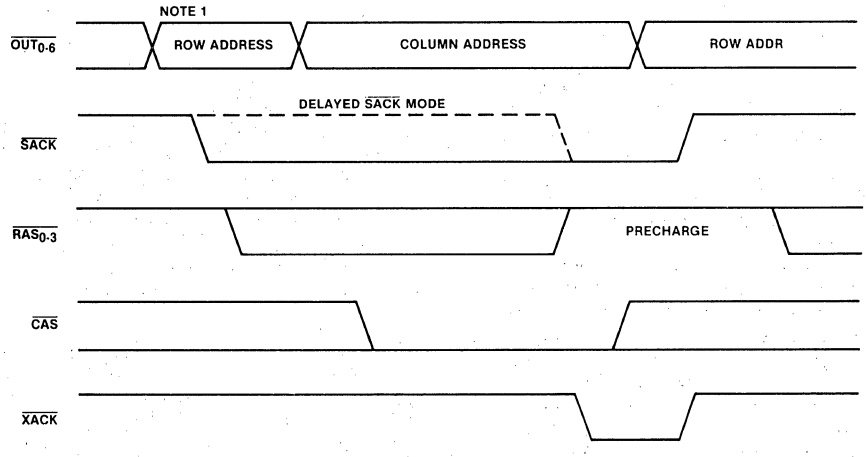
If the REFRQ pin is pulsed or held active while a write cycle is in progress, a refresh cycle will occur immediately following the write cycle, if the Advanced Read Mode is not selected.

Read Cycle (Fig. X.3)

Read cycle operation is the same as write cycle operation, except the write enable ($\overline{WE}$) signal is not activated.

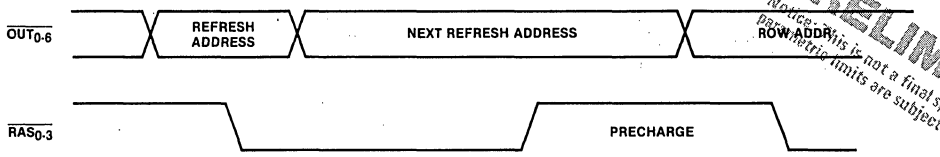
If the REFRQ pin is pulsed or held active while a read cycle is in progress, a refresh cycle will occur immediately following the read cycle, if the Advanced Read Mode is not selected.





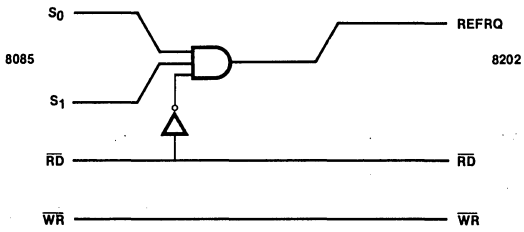
Refresh Cycle (Fig. X.4)

When a refresh cycle is started, (refresh-cycle request previously pulsed or active when sampled) the 8202 resets the Refresh Timer. The Multiplexer drives the OUT 0-6 pins with the refresh address contained in the Refresh Counter. The 8202 then activates the Row Address Strobe (RAS 0-3) signals. At the end of the refresh cycle, all signals are deactivated, the refresh counter is incremented, and the precharge time begins. After the precharge time, the 8202 re-enters the Idle State.



Hidden Refresh Cycle

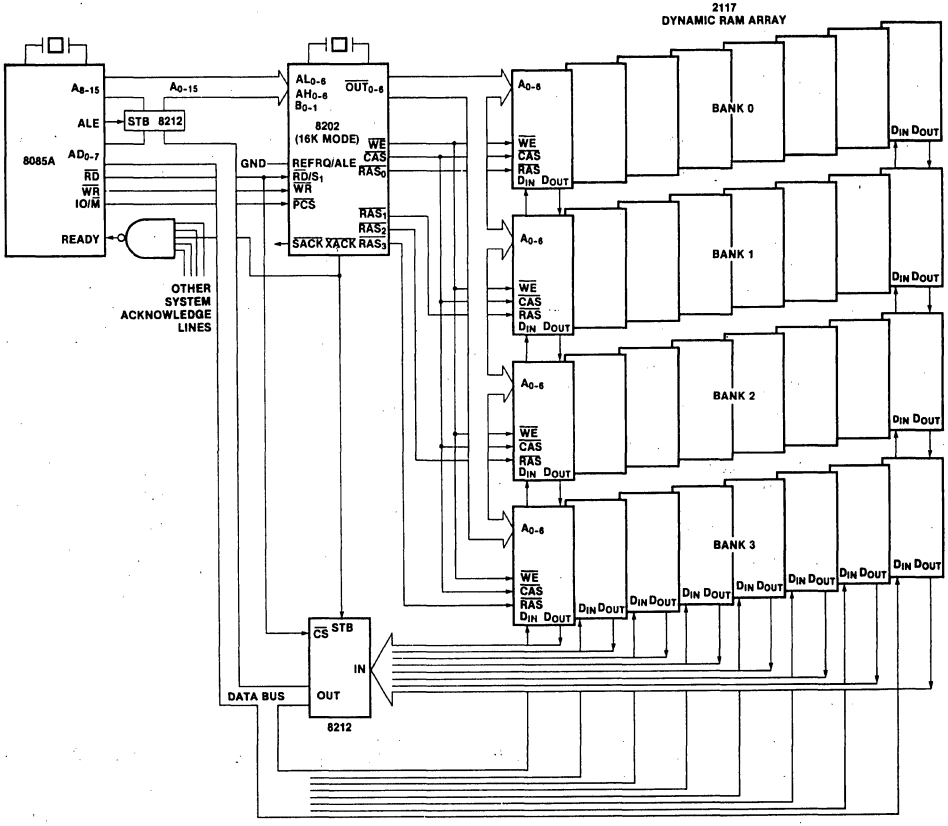
Distributed hidden refresh operation is most efficient if REFRQ is strobed during a command cycle such as fetch, where it is intended for the refresh cycle to follow. This is illustrated for 8085 in the following diagram.



System Configurations

Currently, there exists a wide range of processor bus structures, processor speeds, and memory speeds. As a result, the 8202 offers many possible system configurations with equally many cost-performance tradeoffs.

The following system block diagram illustrates just one of the possible system configurations supported by the 8202:



Other system configurations are described in the Intel, Application Note AP45, "Using the 8202 Dynamic RAM Controller." Other related documents are:

- "Intel Memory Design Handbook" (Dynamic Ram sections).
- AR-1, "Simplify Your Dynamic RAM/Microprocessor Interface."
- AP-38, "Application Techniques for the Intel 8085A Bus."

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias.....0°C to 70°C
Storage Temperature.....-65°C to +150°C
Voltage On Any Pin
With Respect to Ground.....-0.5V to +7V
Power Dissipation.....1.4 Watts

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$; $V_{CC} = 5.0V \pm 10\%$; GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
V_C	Input Clamp Voltage		−1.0	V	$I_C = -5$ mA
I_{CC}	Power Supply Current		250	mA	
I_F	Forward Input Current X_1/CLK All Other Inputs		 −2.0 −320	 mA μA	 $V_F = 0.45V$ $V_F = 0.45V$
I_R	Reverse Input Current		40	μA	$V_R = V_{CC}$
V_{OL}	Output Low Voltage $\overline{SACK}$, $\overline{XACK}$ All Other Outputs		 0.45 0.45	 V V	 $I_{OL} = 5$ mA $I_{OL} = 3$ mA
V_{OH}	Output High Voltage $\overline{SACK}$, $\overline{XACK}$ All Other Outputs	 2.4 2.6		 V V	 $I_{OH} = -1$ mA $I_{OH} = -1$ mA
V_{IL}	Input Low Voltage		0.8	V	$V_{CC} = 5.0V$
V_{IH}	Input High Voltage	2.0		V	$V_{CC} = 5.0V$

CAPACITANCE

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		30	pF	F = 1 MHz $V_{BIAS} = 2.5V$, $V_{CC} = 5V$ $T_A = 25°C$

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 10\%$

Loading:
32 devices

$\overline{SACK}$, $\overline{XACK}$	CL = 30 pF
$\overline{OUT_0} - \overline{OUT_6}$	CL = 160 pF
$\overline{RAS_1} - \overline{RAS_4}$	CL = 115 pF
$\overline{WE}$	CL = 224 pF
$\overline{CAS}$	CL = 320 pF

Measurements made with respect to $\overline{RAS_1} - \overline{RAS_4}$, CAS, WE, $OUT_0 - OUT_6$ are at 2.4V and 0.8V. All other pins are measured at 1.5V.

Symbol	Parameter	Min	Max	Units
t_P	Clock (Internal/External) Period (See Note 1)	40	54	ns
t_{RC}	Memory Cycle Time	$10\ t_P - 30$	$12\ t_P$	ns
t_{RAH}	Row Address Hold Time	$t_P - 10$		ns
t_{ASR}	Row Address Setup Time	t_{PH}		ns
t_{CAH}	Column Address Hold Time	$5\ t_P$		ns
t_{ASC}	Column Address Setup Time	$t_P - 35$		ns
t_{RCD}	$\overline{RAS}$ to $\overline{CAS}$ Delay Time	$2\ t_P - 10$	$2\ t_P + 30$	ns
t_{WCS}	$\overline{WE}$ Setup to $\overline{CAS}$	$t_P - 40$		ns
t_{RSH}	$\overline{RAS}$ Hold Time	$5\ t_P - 30$		ns
t_{CAS}	$\overline{CAS}$ Pulse Width	$5\ t_P$		ns
t_{RP}	$\overline{RAS}$ Precharge Time (See Note 2)	$4\ t_P - 30$		ns
t_{WCH}	$\overline{WE}$ Hold Time to $\overline{CAS}$	$5\ t_P - 20$		ns
t_{REF}	Internally Generated Refresh to Refresh Time 64 Cycle 128 Cycle	 $548\ t_P$ $264\ t_P$	 $576\ t_P$ $288\ t_P$	 ns ns
t_{CR}	$\overline{RD}$, $\overline{WR}$ to $\overline{RAS}$ Delay	$t_{PH} + 30$	$t_{PH} + t_P + 75$	ns
t_{CC}	$\overline{RD}$, $\overline{WR}$ to $\overline{CAS}$ Delay	$t_{PH} + 2\ t_P + 25$	$t_{PH} + 3\ t_P + 85$	ns
t_{RFR}	REFRQ to $\overline{RAS}$ Delay	$1.5\ t_P + 30$	$2.5\ t_P + 100$	ns
t_{AS}	$A_0 - A_{15}$ to $\overline{RD}$, $\overline{WR}$ Setup Time (See Note 4)	0		ns
t_{CA}	$\overline{RD}$, $\overline{WR}$ to $\overline{SACK}$ Leading Edge		$t_P + 40$	ns
t_{CK}	$\overline{RD}$, $\overline{WR}$ to $\overline{XACK}$, $\overline{SACK}$ Trailing Edge Delay		30	ns
t_{KCH}	$\overline{RD}$, $\overline{WR}$ Inactive Hold to $\overline{SACK}$ Trailing Edge	10		ns
t_{SC}	$\overline{RD}$, $\overline{WR}$, $\overline{PCS}$ to X/CLK Setup Time (See Note 3)	15		ns
t_{CX}	$\overline{CAS}$ to $\overline{XACK}$ Time	$5\ t_P - 25$	$5\ t_P + 20$	ns
t_{ACK}	$\overline{XACK}$ Leading Edge to $\overline{CAS}$ Trailing Edge Time	10		ns
t_{XW}	$\overline{XACK}$ Pulse Width	$2\ t_P - 25$		ns
t_{LL}	REFRQ Pulse Width	20		ns
t_{CHS}	$\overline{RD}$, $\overline{WR}$, $\overline{PCS}$ Active Hold to $\overline{RAS}$	0		ns
t_{WW}	$\overline{WR}$ to $\overline{WE}$ Propagation Delay	8	50	ns
t_{AL}	S_1 to ALE Setup Time	40		ns
t_{LA}	S_1 to ALE Hold Time	$2\ t_P + 40$		ns
t_{PL}	External Clock Low Time	15		ns
t_{PH}	External Clock High Time	20		ns
t_{PH}	External Clock High Time for $V_{CC} = 5V \pm 5\%$	17		ns

Notes:
1. t_P minimum determines maximum oscillator frequency.
   t_P maximum determines minimum frequency to maintain 2 ms refresh rate and t_{RP} minimum.
2. To achieve the minimum time between the $\overline{RAS}$ of a memory cycle and the $\overline{RAS}$ of a refresh cycle, such as a transparent refresh, REFRQ should be pulsed in the previous memory cycle.
3. t_{SC} is not required for proper operation which is in agreement with the other specs, but can be used to synchronize external signals with X/CLK if it is desired.
4. If t_{AS} is less than 0 then the only impact is that t_{ASR} decreases by a corresponding amount.

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 10\%$

Loading:
64 Devices

$\overline{SACK}$, $\overline{XACK}$	CL = 30 pF
$\overline{OUT_0} - \overline{OUT_6}$	CL = 320 pF
$\overline{RAS_1} - \overline{RAS_4}$	CL = 230 pF
$\overline{WE}$	CL = 450 pF
$\overline{CAS}$	CL = 640 pF

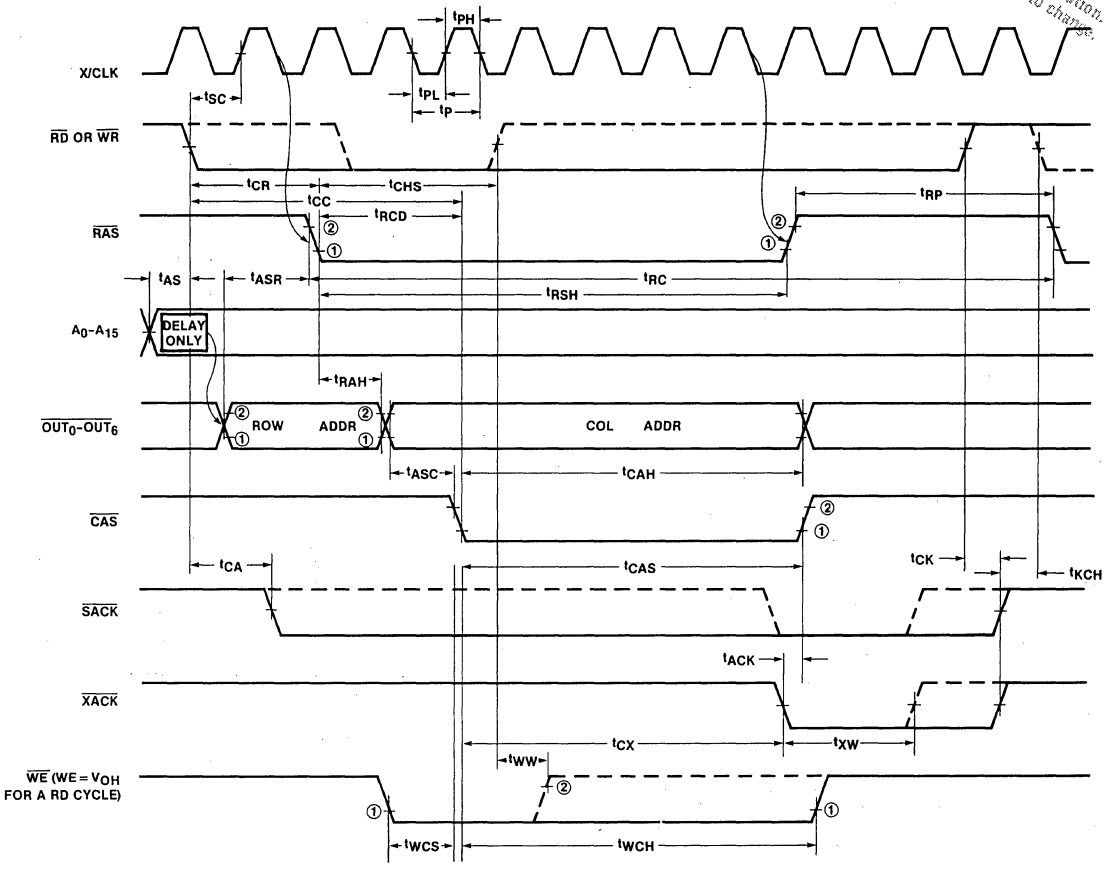
Measurements made with respect to $\overline{RAS_1} - \overline{RAS_4}$, CAS, WE, $OUT_0 - OUT_6$ are at 2.4V and 0.8V. All other pins are measured at 1.5V.

Symbol	Parameter	Min	Max	Units
t_P	Clock (Internal/External) Period (See Note 1)	40	54	ns
t_{RC}	Memory Cycle Time	$10\ t_P - 30$	$12\ t_P$	ns
t_{RAH}	Row Address Hold Time	$t_P - 10$		ns
t_{ASR}	Row Address Setup Time	t_{PH}		ns
t_{CAH}	Column Address Hold Time	$5\ t_P$		ns
t_{ASC}	Column Address Setup Time	$t_P - 35$		ns
t_{RCD}	$\overline{RAS}$ to $\overline{CAS}$ Delay Time	$2\ t_P - 10$	$2\ t_P + 45$	ns
t_{WCS}	$\overline{WE}$ Setup to $\overline{CAS}$	$t_P - 40$		ns
t_{RSH}	$\overline{RAS}$ Hold Time	$5\ t_P - 30$		ns
t_{CAS}	$\overline{CAS}$ Pulse Width	$5\ t_P - 30$		ns
t_{RP}	$\overline{RAS}$ Precharge Time (See Note 2)	$4\ t_P - 30$		ns
t_{WCH}	$\overline{WE}$ Hold Time to $\overline{CAS}$	$5\ t_P - 35$		ns
t_{REF}	Internally Generated Refresh to Refresh Time 64 Cycle 128 Cycle	 $548\ t_P$ $264\ t_P$	 $576\ t_P$ $288\ t_P$	 ns ns
t_{CR}	$\overline{RD}$, $\overline{WR}$ to $\overline{RAS}$ Delay	$t_{PH} + 30$	$t_{PH} + t_P + 75$	ns
t_{CC}	$\overline{RD}$, $\overline{WR}$ to $\overline{CAS}$ Delay	$t_{PH} + 2\ t_P + 25$	$t_{PH} + 3\ t_P + 100$	ns
t_{RFR}	REFRQ to $\overline{RAS}$ Delay	$1.5\ t_P + 30$	$2.5\ t_P + 100$	ns
t_{AS}	$A_0 - A_{15}$ to $\overline{RD}$, $\overline{WR}$ Setup Time (See Note 4)	0		ns
t_{CA}	$\overline{RD}$, $\overline{WR}$ to $\overline{SACK}$ Leading Edge		$t_P + 40$	ns
t_{CK}	$\overline{RD}$, $\overline{WR}$ to $\overline{XACK}$, $\overline{SACK}$ Trailing Edge Delay		30	ns
t_{KCH}	$\overline{RD}$, $\overline{WR}$ Inactive Hold to $\overline{SACK}$ Trailing Edge	10		ns
t_{SC}	$\overline{RD}$, $\overline{WR}$, $\overline{PCS}$ to X/CLK Setup Time (See Note 3)	15		ns
t_{CX}	$\overline{CAS}$ to $\overline{XACK}$ Time	$5\ t_P - 40$	$5\ t_P + 20$	ns
t_{ACK}	$\overline{XACK}$ Leading Edge to $\overline{CAS}$ Trailing Edge Time	10		ns
t_{XW}	$\overline{XACK}$ Pulse Width	$2\ t_P - 25$		ns
t_{LL}	REFRQ Pulse Width	20		ns
t_{CHS}	$\overline{RD}$, $\overline{WR}$, $\overline{PCS}$ Active Hold to $\overline{RAS}$	0		ns
t_{WW}	$\overline{WR}$ to $\overline{WE}$ Propagation Delay	8	50	ns
t_{AL}	S_1 to ALE Setup Time	40		ns
t_{LA}	S_1 to ALE Hold Time	$2\ t_P + 40$		ns
t_{PL}	External Clock Low Time	15		ns
t_{PH}	External Clock High Time	22		ns
t_{PH}	External Clock High Time for $V_{CC} = 5V \pm 5\%$	18		ns

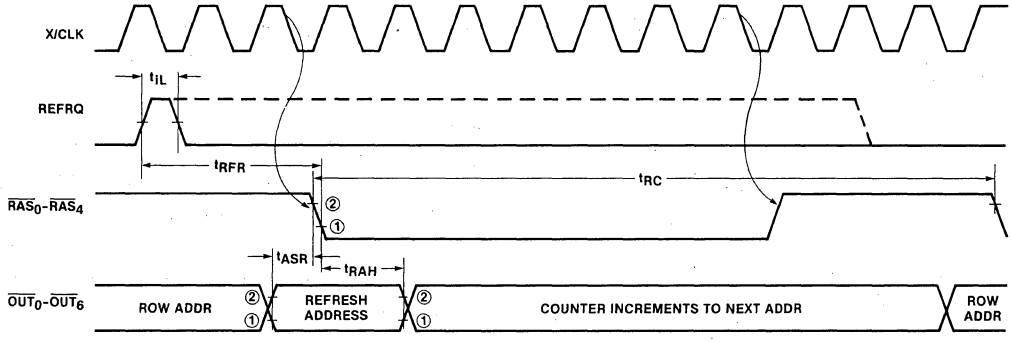
Notes:
1. t_P minimum determines maximum oscillator frequency.
   t_P maximum determines minimum frequency to maintain 2 ms refresh rate and t_{RP} minimum.
2. To achieve the minimum time between the $\overline{RAS}$ of a memory cycle and the $\overline{RAS}$ of a refresh cycle, such as a transparent refresh, REFRQ should be pulsed in the previous memory cycle.
3. t_{SC} is not required for proper operation which is in agreement with the other specs, but can be used to synchronize external signals with X/CLK if it is desired.
4. If t_{AS} is less than 0 then the only impact is that t_{ASR} decreases by a corresponding amount.

8202 TIMING

NORMAL READ OR WRITE CYCLE

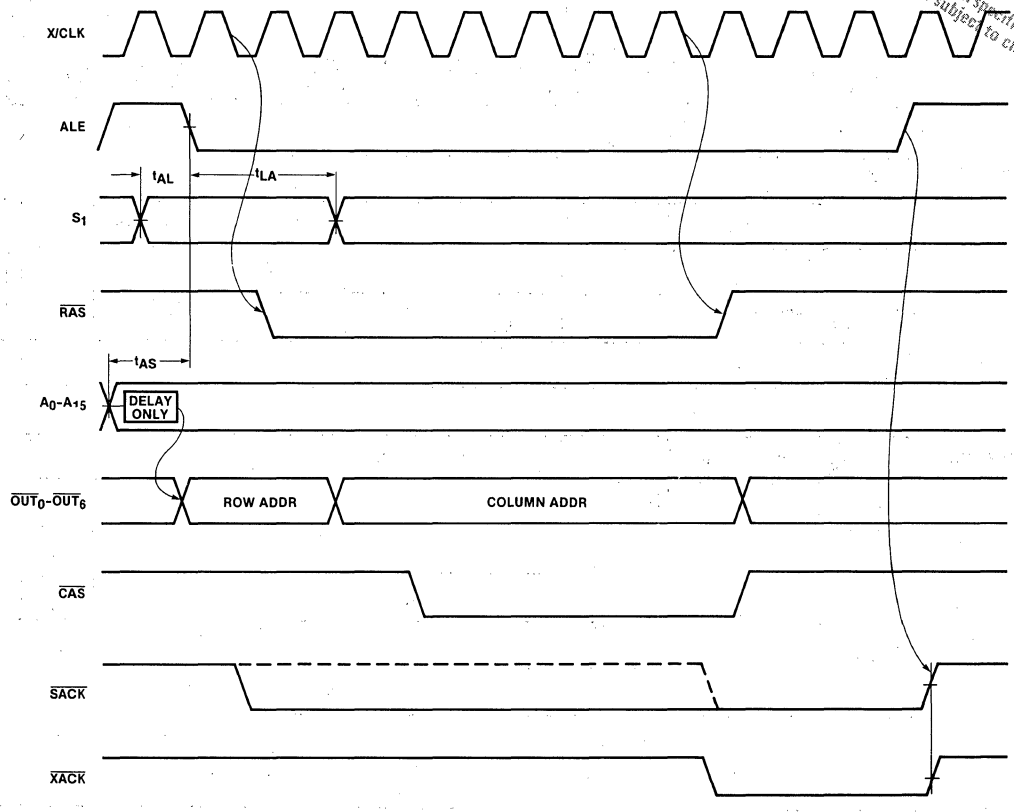


REFRESH CYCLE



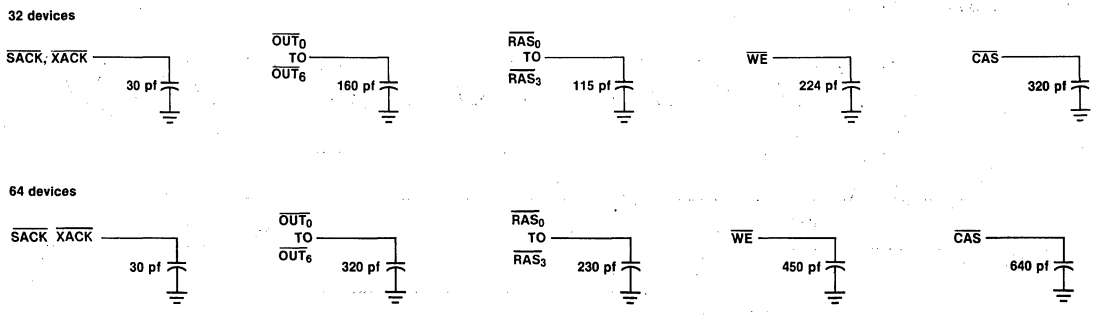
(CAS = VOH) IF THE REFRESH CYCLE IS INTERNALLY TRIGGERED THEN IGNORE REFRQ.

ADVANCED READ MODE USING THE SIMPLIFIED 8085 INTERFACE OPTION



OTHER TIMING PARAMETERS ARE THE SAME AS NORMAL MODE
WRITE CYCLE IS THE SAME AS NORMAL MODE EXCEPT THAT $\overline{XACK}$ AND $\overline{SACK}$ GO INACTIVE ON THE RISING EDGE OF ALE

OUTPUT TEST LOAD CIRCUIT



intel®

8205

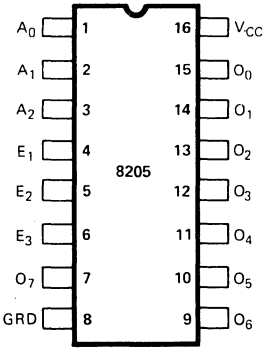
HIGH SPEED 1 OUT OF 8 BINARY DECODER

- **I/O Port or Memory Selector**
- **Simple Expansion — Enable Inputs**
- **High Speed Schottky Bipolar Technology — 18ns Max. Delay**
- **Directly Compatible with TTL Logic Circuits**
- **Low Input Load Current — .25 mA max., 1/6 Standard TTL Input Load**
- **Minimum Line Reflection — Low Voltage Diode Input Clamp**
- **Outputs Sink 10 mA min.**
- **16-Pin Dual-In-Line Ceramic or Plastic Package**

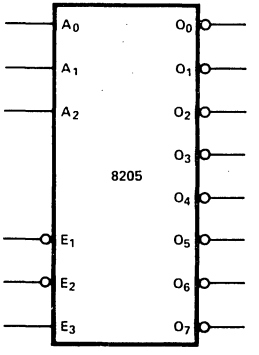
The 8205 decoder can be used for expansion of systems which utilize input ports, output ports, and memory components with active low chip select input. When the 8205 is enabled, one of its eight outputs goes "low", thus a single row of a memory system is selected. The 3 chip enable inputs on the 8205 allow easy system expansion. For very large systems, 8205 decoders can be cascaded such that each decoder can drive eight other decoders for arbitrary memory expansions.

The Intel® 8205 is packaged in a standard 16 pin dual-in-line package; and its performance is specified over the temperature range of 0°C to +75°C, ambient. The use of Schottky barrier diode clamped transistors to obtain fast switching speeds results in higher performance than equivalent devices made with a gold diffusion process.

PIN CONFIGURATION



LOGIC SYMBOL



PIN NAMES

$A_0 - A_2$	ADDRESS INPUTS
$\overline{E_1} - \overline{E_3}$	ENABLE INPUTS
$\overline{O_0} - \overline{O_7}$	DECODED OUTPUTS

ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

intel®

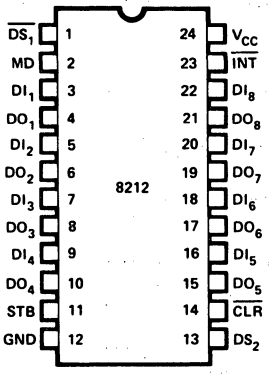
8212
8-BIT INPUT/OUTPUT PORT

- **Fully Parallel 8-Bit Data Register and Buffer**
- **Service Request Flip-Flop for Interrupt Generation**
- **Low Input Load Current — .25mA Max.**
- **Three State Outputs**
- **Outputs Sink 15mA**
- **3.65V Output High Voltage for Direct Interface to 8008, 8080A, or 8085A CPU**
- **Asynchronous Register Clear**
- **Replaces Buffers, Latches and Multiplexers in Microcomputer Systems**
- **Reduces System Package Count**

The 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.

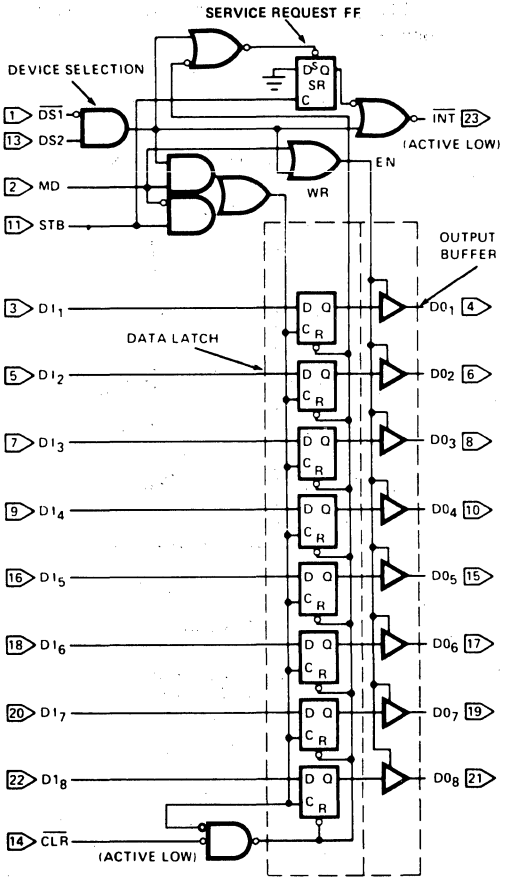
PIN CONFIGURATION



PIN NAMES

DI_1-DI_8	DATA IN
DO_1-DO_8	DATA OUT
$\overline{DS_1}$-DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

LOGIC DIAGRAM



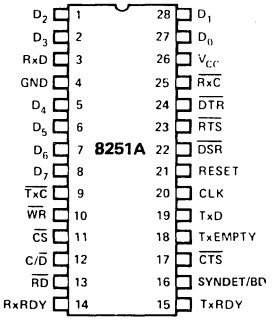
intel®

8251A
PROGRAMMABLE COMMUNICATION INTERFACE

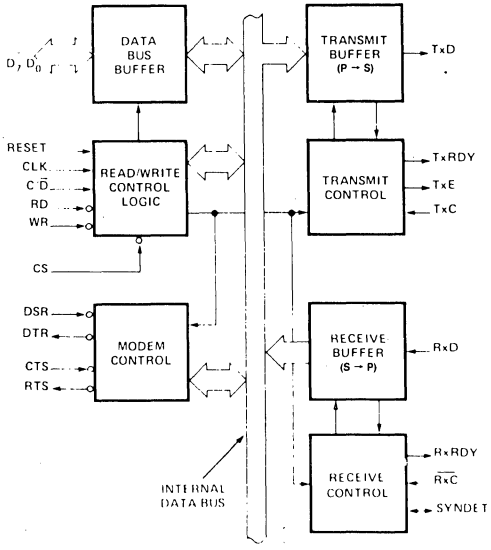
- **Synchronous and Asynchronous Operation**
- **Synchronous 5-8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion**
- **Asynchronous 5-8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling.**
- **Synchronous Baud Rate — DC to 64K Baud**
- **Asynchronous Baud Rate — DC to 19.2K Baud**
- **Full Duplex, Double Buffered, Transmitter and Receiver**
- **Error Detection — Parity, Overrun and Framing**
- **Fully Compatible with 8080/8085 CPU**
- **28-Pin DIP Package**
- **All Inputs and Outputs are TTL Compatible**
- **Single +5V Supply**
- **Single TTL Clock**

The Intel® 8251A is the enhanced version of the industry standard, Intel® 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's new high performance family of microprocessors such as the 8085. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is constructed using N-channel silicon gate technology.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

D_7-D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
$\overline{RD}$	Read Data Command
$\overline{WR}$	Write Data or Control Command
$\overline{CS}$	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char. from 8080)

$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
$\overline{RTS}$	Request to Send Data
$\overline{CTS}$	Clear to Send Data
TxE	Transmitter Empty
V_{CC}	+5 Volt Supply
GND	Ground

FEATURES AND ENHANCEMENTS

8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors that includes the new 8085 CPU and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.
- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the $\overline{RD}$ and $\overline{WR}$ do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Synchronous Baud rate from DC to 64K.
- Fully compatible with Intel's new industry standard, the MCS-85.

8251A BASIC FUNCTIONAL DESCRIPTION

General

The 8251A is a Universal Synchronous/Asynchronous Receiver/Transmitter designed specifically for the 80/85 Microcomputer Systems. Like other I/O devices in a Microcomputer System, its functional configuration is programmed by the system's software for maximum flexibility. The 8251A can support virtually any serial data technique currently in use including bi-sync.

In a communication environment an interface device must convert parallel format system data into serial format for transmission and convert incoming serial format data into parallel system data for reception. The interface device must also delete or insert bits or characters that are functionally unique to the communication technique. In essence, the interface should appear "transparent" to the CPU, a simple input or output of byte-oriented system data.

Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8251A to the system Data Bus. Data is transmitted or received by the buffer upon execution of INput or OUTput instructions of the CPU. Control words, Command words and Status information are also transferred through the Data Bus Buffer. The command status and data in, and data out are separate 8-bit registers to provide double buffering.

This functional block accepts inputs from the system Control bus and generates control signals for overall device operation. It contains the Control Word Register and Command Word Register that store the various control formats for the device functional definition.

RESET (Reset)

A "high" on this input forces the 8251A into an "Idle" mode. The device will remain at "Idle" until a new set of control words is written into the 8251A to program its functional definition. Minimum RESET pulse width is 6 t_{CY} (clock must be running).

CLK (Clock)

The CLK input is used to generate internal device timing and is normally connected to the Phase 2 (TTL) output of the 8224 Clock Generator. No external inputs or outputs are referenced to CLK but the frequency of CLK must be greater than 30 times the Receiver or Transmitter data bit rates.

$\overline{WR}$ (Write)

A "low" on this input informs the 8251A that the CPU is writing data or control words to the 8251A.

$\overline{RD}$ (Read)

A "low" on this input informs the 8251A that the CPU is reading data or status information from the 8251A.

C/$\overline{D}$ (Control/Data)

This input, in conjunction with the $\overline{WR}$ and $\overline{RD}$ inputs, informs the 8251A that the word on the Data Bus is either a data character, control word or status information.
1 = CONTROL/STATUS 0 = DATA

$\overline{CS}$ (Chip Select)

A "low" on this input selects the 8251A. No reading or writing will occur unless the device is selected. When $\overline{CS}$ is high, the Data Bus in the float state and $\overline{RD}$ and $\overline{WR}$ will have no effect on the chip.

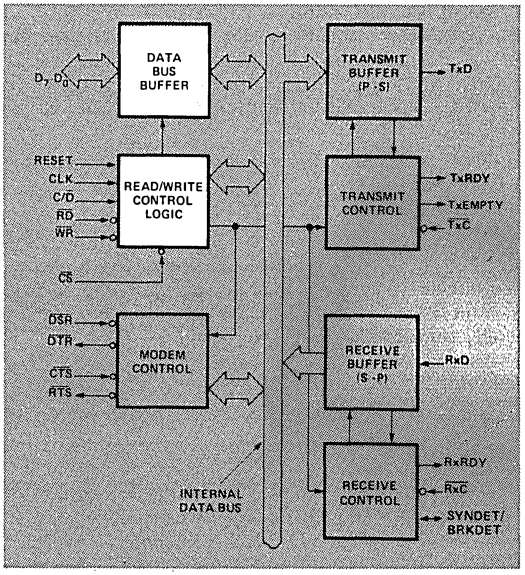


Figure 1. 8251A Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	8251A DATA ⇒ DATA BUS
0	1	0	0	DATA BUS ⇒ 8251A DATA
1	0	1	0	STATUS ⇒ DATA BUS
1	1	0	0	DATA BUS ⇒ CONTROL
X	1	1	0	DATA BUS ⇒ 3-STATE
X	X	X	1	DATA BUS ⇒ 3-STATE

Modem Control

The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any modem. The modem control signals are general purpose in nature and can be used for functions other than modem control, if necessary.

$\overline{\text{DSR}}$ (Data Set Ready)

The $\overline{\text{DSR}}$ input signal is a general purpose, 1-bit inverting input port. Its condition can be tested by the CPU using a Status Read operation. The $\overline{\text{DSR}}$ input is normally used to test modem conditions such as Data Set Ready.

$\overline{\text{DTR}}$ (Data Terminal Ready)

The $\overline{\text{DTR}}$ output signal is a general purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{\text{DTR}}$ output signal is normally used for modem control such as Data Terminal Ready or Rate Select.

$\overline{\text{RTS}}$ (Request to Send)

The $\overline{\text{RTS}}$ output signal is a general purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{\text{RTS}}$ output signal is normally used for Modem control such as Request to Send.

$\overline{\text{CTS}}$ (Clear to Send)

A "low" on this input enables the 8251A to transmit serial data if the Tx Enable bit in the Command byte is set to a "one." If either a Tx Enable off or CTS off condition occurs while the Tx is in operation, the Tx will transmit all the data in the USART, written prior to Tx Disable command before shutting down.

Transmitter Buffer

The Transmitter Buffer accepts parallel data from the Data Bus Buffer, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin on the falling edge of $\overline{\text{TxC}}$. The transmitter will begin transmission upon being enabled if $\overline{\text{CTS}}$ = 0. The TxD line will be held in the marking state immediately upon a master Reset or when Tx Enable/$\overline{\text{CTS}}$ off or TxEMPTY.

Transmitter Control

The transmitter Control manages all activities associated with the transmission of serial data. It accepts and issues signals both externally and internally to accomplish this function.

TxRDY (Transmitter Ready)

This output signals the CPU that the transmitter is ready to accept a data character. The TxRDY output pin can be used as an interrupt to the system, since it is masked by Tx Disabled, or, for Polled operation, the CPU can check TxRDY using a Status Read operation. TxRDY is automatically reset by the leading edge of $\overline{\text{WR}}$ when a data character is loaded from the CPU.

Note that when using the Polled operation, the TxRDY status bit is *not* masked by Tx Enabled, but will only indicate the Empty/Full Status of the Tx Data Input Register.

TxE (Transmitter Empty)

When the 8251A has no characters to transmit, the TxEMPTY output will go "high". It resets automatically upon receiving a character from the CPU. TxEMPTY can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplexed operational mode. TxEMPTY is independent of the Tx Enable bit in the Command instruction.

In SYNChronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be or are being transmitted automatically as "fillers". TxEMPTY does not go low when the SYNC characters are being shifted out.

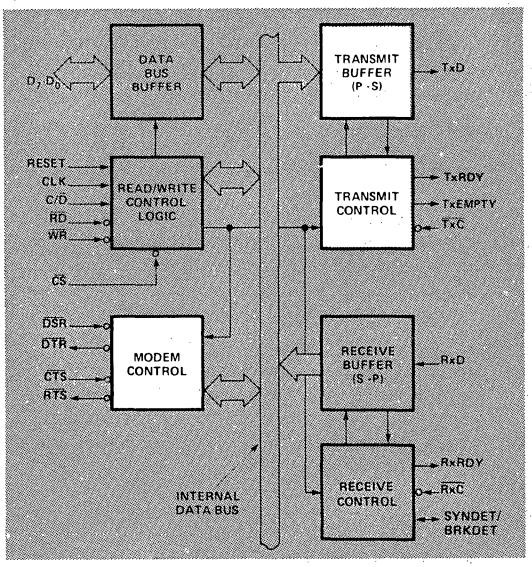


Figure 2. 8251A Block Diagram Showing Modem and Transmitter Buffer and Control Functions

$\overline{\text{TxC}}$ (Transmitter Clock)

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the Baud Rate (1x) is equal to the $\overline{\text{TxC}}$ frequency. In Asynchronous transmission mode the baud rate is a fraction of the actual $\overline{\text{TxC}}$ frequency. A portion of the mode instruction selects this factor; it can be 1, 1/16 or 1/64 the $\overline{\text{TxC}}$.

For Example:

If Baud Rate equals 110 Baud,
$\overline{\text{TxC}}$ equals 110 Hz (1x)
$\overline{\text{TxC}}$ equals 1.76 kHz (16x)
$\overline{\text{TxC}}$ equals 7.04 kHz (64x).

The falling edge of $\overline{\text{TxC}}$ shifts the serial data out of the 8251A.

Receiver Buffer

The Receiver accepts serial data, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU. Serial data is input to RxD pin, and is clocked in on the rising edge of $\overline{RxC}$.

Receiver Control

This functional block manages all receiver-related activities which consist of the following features:

The RxD initialization circuit prevents the 8251A from mistaking an unused input line for an active low data line in the "break condition". Before starting to receive serial characters on the RxD line, a valid "1" must first be detected after a chip master Reset. Once this has been determined, a search for a valid low (Start bit) is enabled. This feature is only active in the asynchronous mode, and is only done once for each master Reset.

The False Start bit detection circuit prevents false starts due to a transient noise spike by first detecting the falling edge and then strobing the nominal center of the Start bit (RxD = low).

The Parity Toggle F/F and Parity Error F/F circuits are used for parity error detection and set the corresponding status bit.

The Framing Error Flag F/F is set if the Stop bit is absent at the end of the data byte (asynchronous mode), and also sets the corresponding status bit.

RxRDY (Receiver Ready)

This output indicates that the 8251A contains a character that is ready to be input to the CPU. Rx RDY can be connected to the interrupt structure of the CPU or, for Polled operation, the CPU can check the condition of RxRDY using a Status Read operation.

Rx Enable off both masks and holds RxRDY in the Reset Condition. For Asynchronous mode, to set RxRDY, the Receiver must be Enabled to sense a Start Bit and a complete character must be assembled and transferred to the Data Output Register. For Synchronous mode, to set RxRDY, the Receiver must be enabled and a character must finish assembly and be transferred to the Data Output Register.

Failure to read the received character from the Rx Data Output Register prior to the assembly of the next Rx Data character will set overrun condition error and the previous character will be written over and lost. If the Rx Data is being read by the CPU when the internal transfer is occurring, overrun error will be set and the old character will be lost.

$\overline{RxC}$ (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the Baud Rate (1x) is equal to the actual frequency of $\overline{RxC}$. In Asynchronous Mode, the Baud Rate is a fraction of the actual $\overline{RxC}$ frequency. A portion of the mode instruction selects this factor; 1, 1/16 or 1/64 the $\overline{RxC}$.

For Example:

Baud Rate equals 300 Baud, if
$\overline{RxC}$ equals 300 Hz (1x)
$\overline{RxC}$ equals 4800 Hz (16x)
$\overline{RxC}$ equals 19.2 kHz (64x).

Baud Rate equals 2400 Baud, if
$\overline{RxC}$ equals 2400 Hz (1x)
$\overline{RxC}$ equals 38.4 kHz (16x)
$\overline{RxC}$ equals 153.6 kHz (64x).

Data is sampled into the 8251A on the rising edge of $\overline{RxC}$.

NOTE: In most communications systems, the 8251A will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the same. Both $\overline{TxC}$ and $\overline{RxC}$ will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

SYNDET (SYNC Detect)/BRKDET (Break Detect)

This pin is used in SYNChronous Mode for SYNDET and may be used as either input or output, programmable through the Control Word. It is reset to output mode low upon RESET. When used as an output (internal Sync mode), the SYNDET pin will go "high" to indicate that the 8251A has located the SYNC character in the Receive mode. If the 8251A is programmed to use double Sync characters (bisync), then SYNDET will go "high" in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

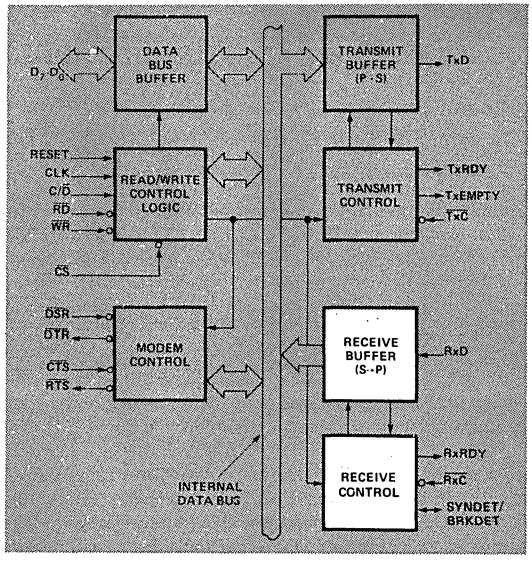


Figure 3. 8251A Block Diagram Showing Receiver Buffer and Control Functions

When used as an input (external SYNC detect mode), a positive going signal will cause the 8251A to start assembling data characters on the rising edge of the next $\overline{RxC}$. Once in SYNC, the "high" input signal can be removed. When External SYNC Detect is programmed, the Internal SYNC Detect is disabled.

BREAK DETECT (Async Mode Only)

This output will go high whenever the receiver remains low through two consecutive stop bit sequences (including the start bits, data bits, and parity bits). Break Detect may also be read as a Status bit. It is reset only upon a master chip Reset or Rx Data returning to a "one" state.

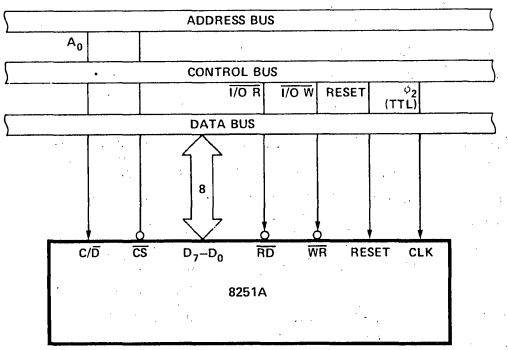


Figure 4. 8251A Interface to 8080 Standard System Bus

DETAILED OPERATION DESCRIPTION

General

The complete functional definition of the 8251A is programmed by the system's software. A set of control words must be sent out by the CPU to initialize the 8251A to support the desired communications format. These control words will program the: BAUD RATE, CHARACTER LENGTH, NUMBER OF STOP BITS, SYNCHRONOUS or ASYNCHRONOUS OPERATION, EVEN/ODD/OFF PARITY, etc. In the Synchronous Mode, options are also provided to select either internal or external character synchronization.

Once programmed, the 8251A is ready to perform its communication functions. The TxRDY output is raised "high" to signal the CPU that the 8251A is ready to receive a data character from the CPU. This output (TxRDY) is reset automatically when the CPU writes a character into the 8251A. On the other hand, the 8251A receives serial data from the MODEM or I/O device. Upon receiving an entire character, the RxRDY output is raised "high" to signal the CPU that the 8251A has a complete character ready for the CPU to fetch. RxRDY is reset automatically upon the CPU data read operation.

The 8251A cannot begin transmission until the Tx Enable (Transmitter Enable) bit is set in the Command Instruction and it has received a Clear To Send ($\overline{CTS}$) input. The TxD output will be held in the marking state upon Reset.

Programming the 8251A

Prior to starting data transmission or reception, the 8251A must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251A and must immediately follow a Reset operation (internal or external).

The control words are split into two formats:

1. Mode Instruction
2. Command Instruction

Mode Instruction

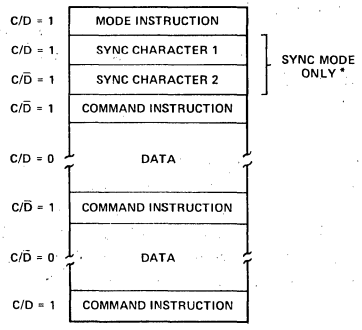
This format defines the general operational characteristics of the 8251A. It must follow a Reset operation (internal or external). Once the Mode Instruction has been written into the 8251A by the CPU, SYNC characters or Command Instructions may be inserted.

Command Instruction

This format defines a status word that is used to control the actual operation of the 8251A.

Both the Mode and Command Instructions must conform to a specified sequence for proper device operation. The Mode Instruction must be inserted immediately following a Reset operation, prior to using the 8251A for data communication.

All control words written into the 8251A after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251A at any time in the data block during the operation of the 8251A. To return to the Mode Instruction format, the master Reset bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251A back into the Mode Instruction format. Command Instructions must follow the Mode Instructions or Sync characters.



* The second SYNC character is skipped if MODE instruction has programmed the 8251A to single character Internal SYNC Mode. Both SYNC characters are skipped if MODE instruction has programmed the 8251A to ASYNC mode.

Figure 5. Typical Data Block

Mode Instruction Definition

The 8251A can be used for either Asynchronous or Synchronous data communication. To understand how the Mode Instruction defines the functional operation of the 8251A, the designer can best view the device as two separate components sharing the same package, one Asynchronous the other Synchronous. The format definition can be changed only after a master chip Reset. For explanation purposes the two formats will be isolated.

NOTE: When parity is enabled it is not considered as one of the data bits for the purpose of programming the word length. The actual parity bit received on the Rx Data line cannot be read on the Data Bus. In the case of a programmed character length of less than 8 bits, the least significant Data Bus bits will hold the data; unused bits are "don't care" when writing data to the 8251A, and will be "zeros" when reading the data from the 8251A.

Asynchronous Mode (Transmission)

Whenever a data character is sent by the CPU the 8251A automatically adds a Start bit (low level) followed by the data bits (least significant bit first), and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TxD output. The serial data is shifted out on the falling edge of $\overline{TxC}$ at a rate equal to 1, 1/16, or 1/64 that of the $\overline{TxC}$, as defined by the Mode Instruction. BREAK characters can be continuously sent to the TxD if commanded to do so.

When no data characters have been loaded into the 8251A the TxD output remains "high" (marking) unless a Break (continuously low) has been programmed.

Asynchronous Mode (Receive)

The RxD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center (16X or 64X mode only). If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter thus locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RxD pin with the rising edge of $\overline{RxC}$. If a low level is detected as the STOP bit, the Framing Error flag will be set. The STOP bit signals the end of a character. Note that the *receiver* requires only *one* stop bit, regardless of the number of stop bits programmed. This character is then loaded into the parallel I/O buffer of the 8251A. The RxRDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the OVERRUN Error flag is raised (thus the previous character is lost). All of the error flags can be reset by an Error Reset Instruction. The occurrence of any of these errors will not affect the operation of the 8251A.

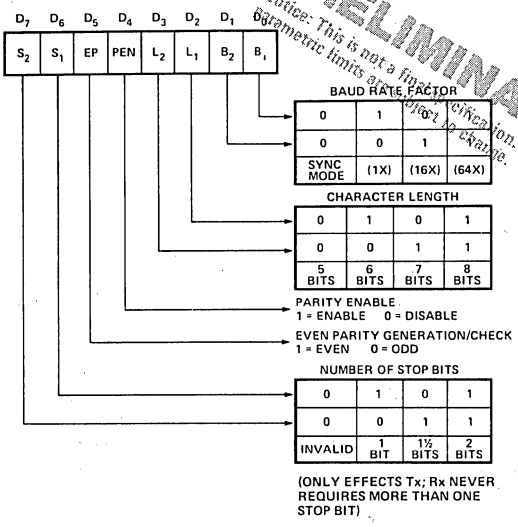


Figure 6. Mode Instruction Format, Asynchronous Mode

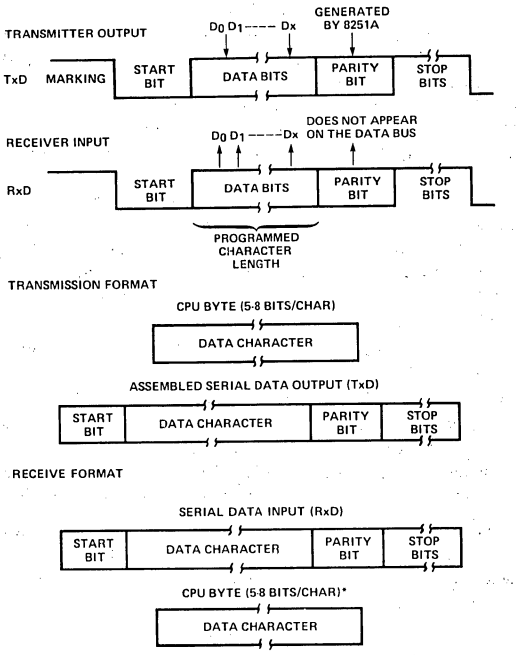
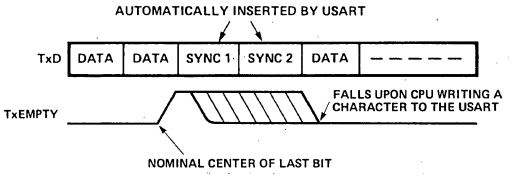


Figure 7. Asynchronous Mode

Synchronous Mode (Transmission)

The TxD output is continuously high until the CPU sends its first character to the 8251A which usually is a SYNC character. When the $\overline{CTS}$ line goes low, the first character is serially transmitted out. All characters are shifted out on the falling edge of $\overline{TxC}$. Data is shifted out at the same rate as the $\overline{TxC}$.

Once transmission has started, the data stream at the TxD output must continue at the $\overline{TxC}$ rate. If the CPU does not provide the 8251A with a data character before the 8251A Transmitter Buffers become empty, the SYNC characters (or character if in single SYNC character mode) will be automatically inserted in the TxD data stream. In this case, the TxEMPTY pin is raised high to signal that the 8251A is empty and SYNC characters are being sent out. TxEMPTY does not go low when the SYNC is being shifted out (see figure below). The TxEMPTY pin is internally reset by a data character being written into the 8251A.



Synchronous Mode (Receive)

In this mode, character synchronization can be internally or externally achieved. If the SYNC mode has been programmed, ENTER HUNT command should be included in the first command instruction word written. Data on the RxD pin is then sampled in on the rising edge of $\overline{RxC}$. The content of the Rx buffer is compared at every bit boundary with the first SYNC character until a match occurs. If the 8251A has been programmed for two SYNC characters, the subsequent received character is also compared; when both SYNC characters have been detected, the USART ends the HUNT mode and is in character synchronization. The SYNDET pin is then set high, and is reset automatically by a STATUS READ. If parity is programmed, SYNDET will not be set until the middle of the parity bit instead of the middle of the last data bit.

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDET pin, thus forcing the 8251A out of the HUNT mode. The high level can be removed after one $\overline{RxC}$ cycle. An ENTER HUNT command has no effect in the asynchronous mode of operation.

Parity error and overrun error are both checked in the same way as in the Asynchronous Rx mode. Parity is checked when not in Hunt, regardless of whether the Receiver is enabled or not.

The CPU can command the receiver to enter the HUNT mode if synchronization is lost. This will also set all the used character bits in the buffer to a "one", thus preventing a possible false SYNDET caused by data that happens to be in the Rx Buffer at ENTER HUNT time. Note that the SYNDET F/F is reset at each Status Read, regardless of whether internal or external SYNC has been programmed. This does not cause the 8251A to return to the HUNT mode. When in SYNC mode, but not in HUNT, Sync Detection is still functional, but only occurs at the "known" word boundaries. Thus, if one Status Read indicates SYNDET and a second Status Read also indicates SYNDET, then the programmed SYNDET characters have been received since the previous Status Read. (If double character sync has been programmed, then both sync characters have been contiguously received to gate a SYNDET indication.) When external SYNDET mode is selected, internal Sync Detect is disabled, and the SYNDET F/F may be set at any bit boundary.

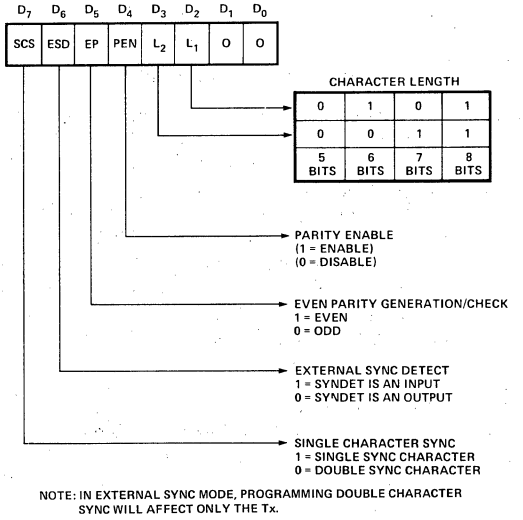


Figure 8. Mode Instruction Format, Synchronous Mode

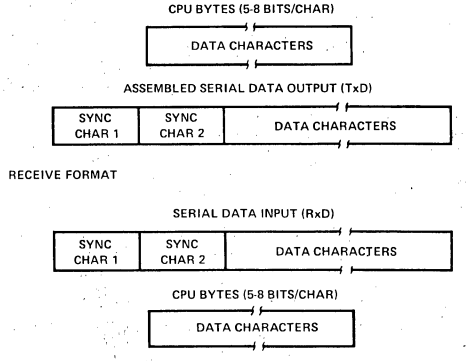
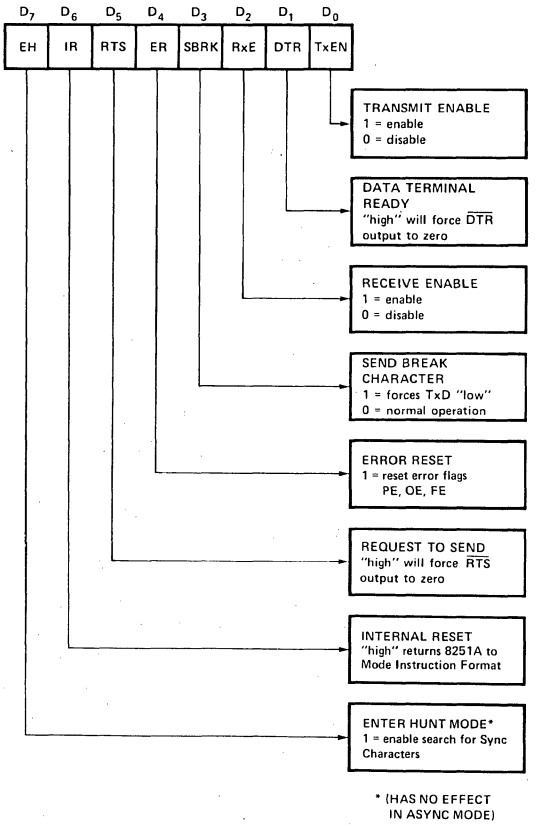


Figure 9. Data Format, Synchronous Mode

COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the Sync Characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command Instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, if necessary, then all further "control writes" ($C/\overline{D}$ = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.



Note: Error Reset must be performed whenever RxEnable and Enter Hunt are programmed.

Figure 10. Command Instruction Format

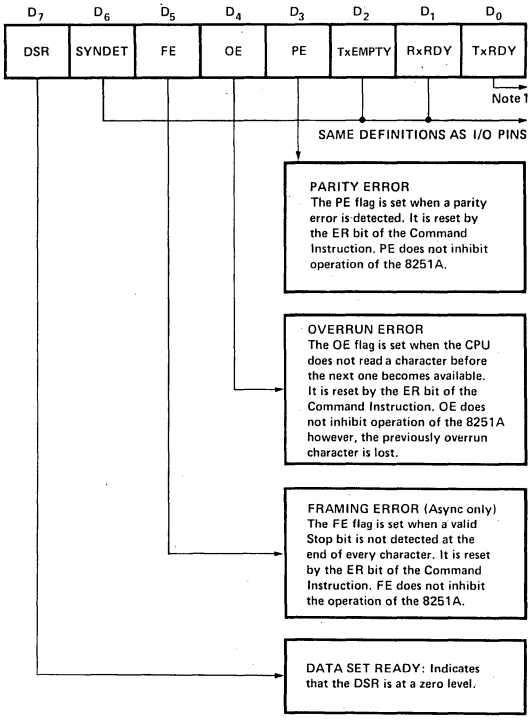
STATUS READ DEFINITION

In data communication systems it is often necessary to examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251A has facilities that allow the programmer to "read" the status of the device at any time during the functional operation. (The status update is inhibited during status read).

A normal "read" command is issued by the CPU with $C/\overline{D}$ = 1 to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251A can be used in a completely Polled environment or in an interrupt driven environment. TxRDY is an exception.

Note that status update can have a maximum delay of 28 clock periods from the actual event affecting the status.



Note 1: TxRDY status bit has different meanings from the TxRDY output pin. The former is not conditioned by $\overline{CTS}$ and TxEN; the latter is conditioned by both $\overline{CTS}$ and TxEN.

i.e. TxRDY status bit = DB Buffer Empty

TxRDY pin out = DB Buffer Empty $\cdot$ (CTS=0) $\cdot$ (TxEN=1)

Figure 11. Status Read Format

APPLICATIONS OF THE 8251A

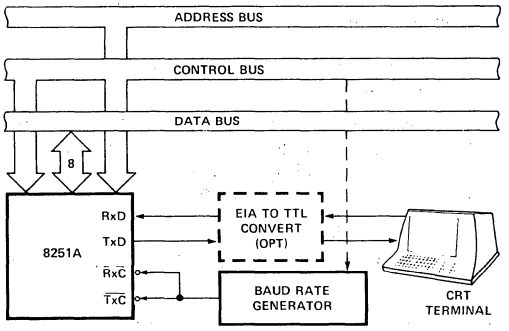


Figure 12. Asynchronous Serial Interface to CRT Terminal, DC—9600 Baud

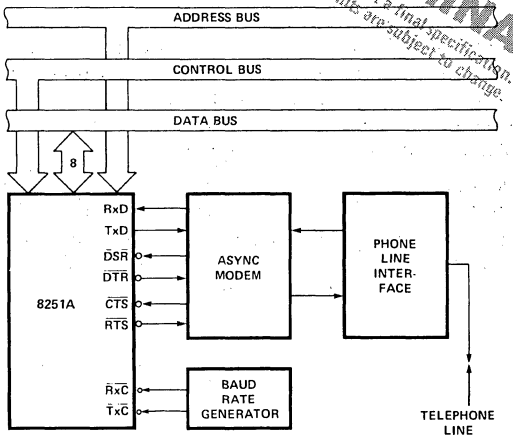


Figure 14. Asynchronous Interface to Telephone Lines

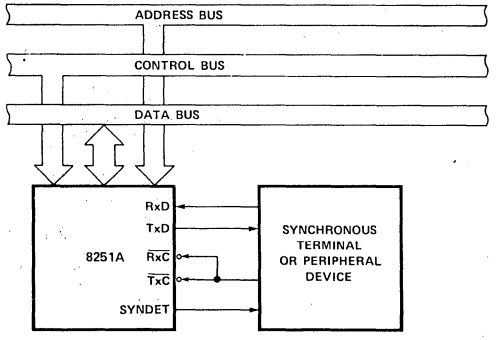


Figure 13. Synchronous Interface to Terminal or Peripheral Device

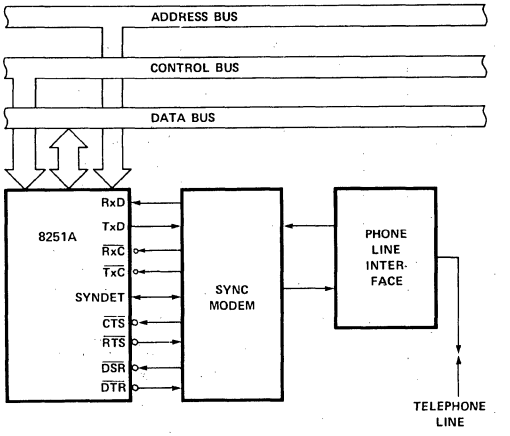


Figure 15. Synchronous Interface to Telephone Lines

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . .–65°C to +150°C
Voltage On Any Pin
With Respect to Ground . . . . . –0.5V to +7V
Power Dissipation . . . . . 1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	–0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC}	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = –400 μA
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} TO 0.45V
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} TO 0.45V
I_{CC}	Power Supply Current		100	mA	All Outputs = High

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance		10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND

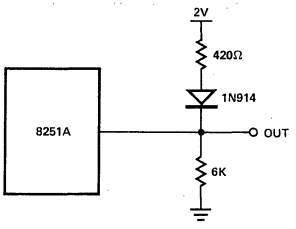


Figure 16. Test Load Circuit

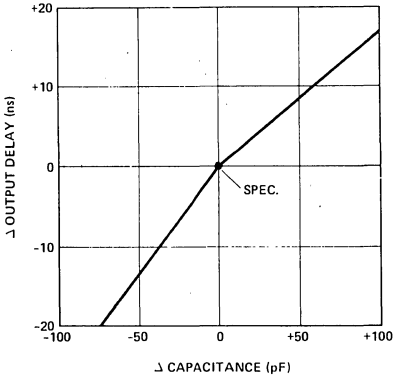


Figure 17. Typical Δ Output Delay vs. Δ Capacitance (pF)

A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Bus Parameters (Note 1)

Read Cycle:

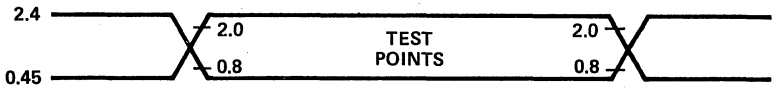
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{AR}	Address Stable Before $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	50		ns	Note 2
t_{RA}	Address Hold Time for $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	50		ns	Note 2
t_{RR}	$\overline{READ}$ Pulse Width	250		ns	
t_{RD}	Data Delay from $\overline{READ}$		250	ns	3, C_L = 150 pF
t_{DF}	$\overline{READ}$ to Data Floating	10	100	ns	

Write Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{AW}	Address Stable Before $\overline{WRITE}$	50		ns	
t_{WA}	Address Hold Time for $\overline{WRITE}$	50		ns	
t_{WW}	$\overline{WRITE}$ Pulse Width	250		ns	
t_{DW}	Data Set Up Time for $\overline{WRITE}$	150		ns	
t_{WD}	Data Hold Time for $\overline{WRITE}$	30		ns	
t_{RV}	Recovery Time Between WRITES	6		t_{CY}	Note 4

NOTES:
1. AC timings measured V_{OH} = 2.0, V_{OL} = 0.8, and with load circuit of Figure 1.
2. Chip Select ($\overline{CS}$) and Command/Data (C/$\overline{D}$) are considered as Addresses.
3. Assumes that Address is valid before $\overline{RD}$↓.
4. This recovery time is for Mode Initialization only. Write Data is allowed only when TxRDY = 1. Recovery Time between Writes for Asynchronous Mode is 8 t_{CY} and for Synchronous Mode is 16 t_{CY}.

Input Waveforms for AC Tests



Other Timings:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{CY}	Clock Period	320	1350	ns	Notes 5, 6
t_{ϕ}	Clock High Pulse Width	140	t_{CY-90}	ns	
$t_{\overline{\phi}}$	Clock Low Pulse Width	90		ns	
t_R, t_F	Clock Rise and Fall Time	5	20	ns	
t_{DTx}	TxD Delay from Falling Edge of $\overline{TxC}$		1	μs	
t_{SRx}	Rx Data Set-Up Time to Sampling Pulse	2		μs	
t_{HRx}	Rx Data Hold Time to Sampling Pulse	2		μs	
f_{Tx}	Transmitter Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{TPW}	Transmitter Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{TPD}	Transmitter Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
f_{Rx}	Receiver Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{RPW}	Receiver Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{RPD}	Receiver Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
t_{TxRDY}	TxRDY Pin Delay from Center of last Bit		8	t_{CY}	Note 7
$t_{TxRDY\ CLEAR}$	TxRDY ↓ from Leading Edge of $\overline{WR}$		180	ns	Note 7
t_{RxRDY}	RxRDY Pin Delay from Center of last Bit		24	t_{CY}	Note 7
$t_{RxRDY\ CLEAR}$	RxRDY ↓ from Leading Edge of $\overline{RD}$		150	ns	Note 7
t_{IS}	Internal SYNDET Delay from Rising Edge of $\overline{RxC}$		24	t_{CY}	Note 7
t_{ES}	External SYNDET Set-Up Time Before Falling Edge of $\overline{RxC}$		16	t_{CY}	Note 7
$t_{TxEMPTY}$	TxEMPTY Delay from Center of Last Bit		20	t_{CY}	Note 7
t_{WC}	Control Delay from Rising Edge of WRITE (TxEn, $\overline{DTR}$, $\overline{RTS}$)		8	t_{CY}	Note 7
t_{CR}	Control to READ Set-Up Time ($\overline{DSR}$, $\overline{CTS}$)		20	t_{CY}	Note 7

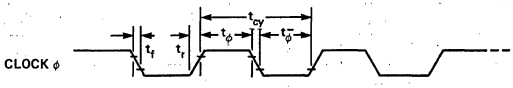
5. The TxC and RxC frequencies have the following limitations with respect to CLK.
   For 1x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(30\ t_{CY})$
   For 16x and 64x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(4.5\ t_{CY})$

6. Reset Pulse Width = 6 t_{CY} minimum; System Clock must be running during Reset.

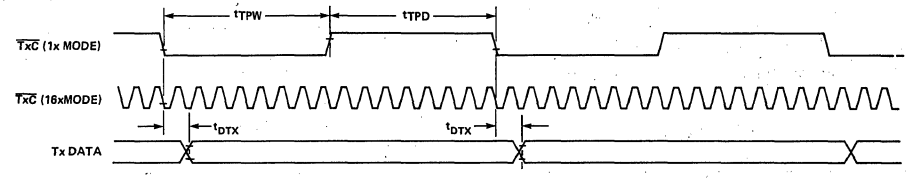
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.

WAVEFORMS

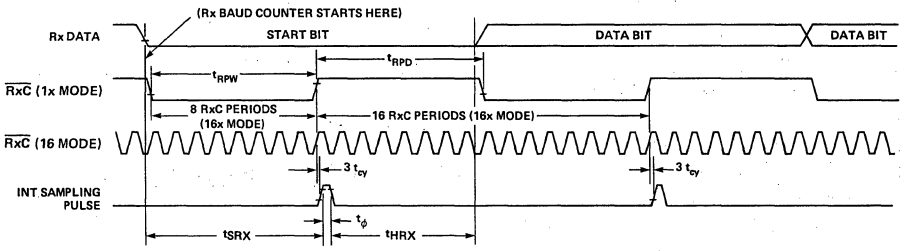
System Clock Input



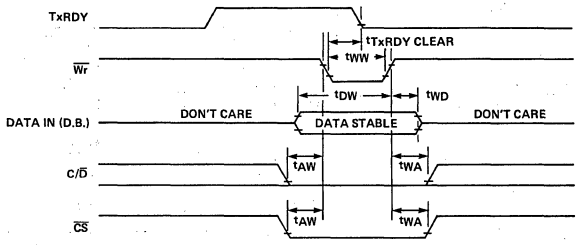
Transmitter Clock & Data



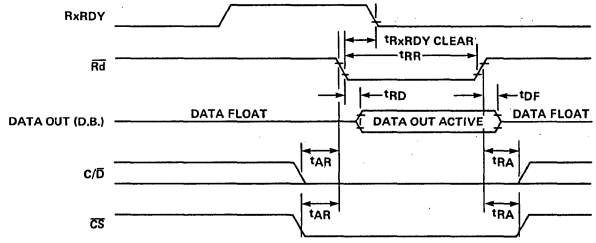
Receiver Clock & Data



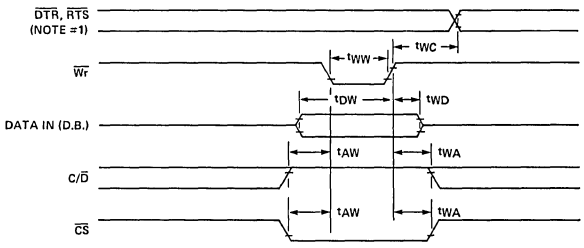
Write Data Cycle (CPU → USART)



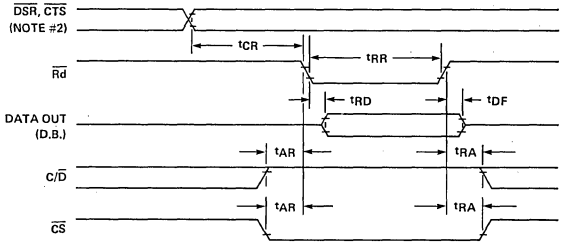
Read Data Cycle (CPU ← USART)



Write Control or Output Port Cycle (CPU → USART)

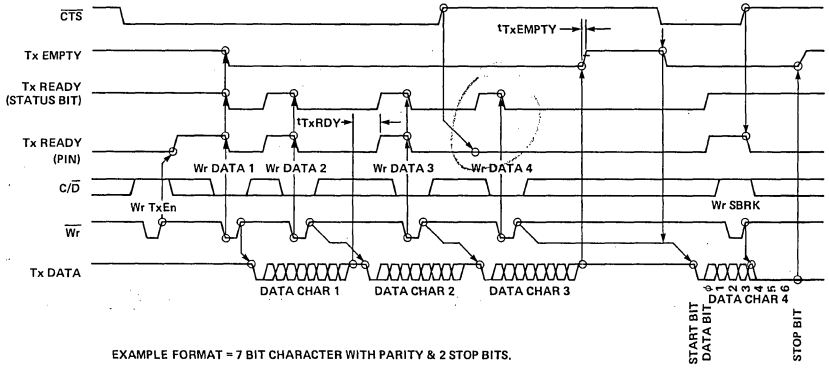


Read Control or Input Port (CPU ← USART)



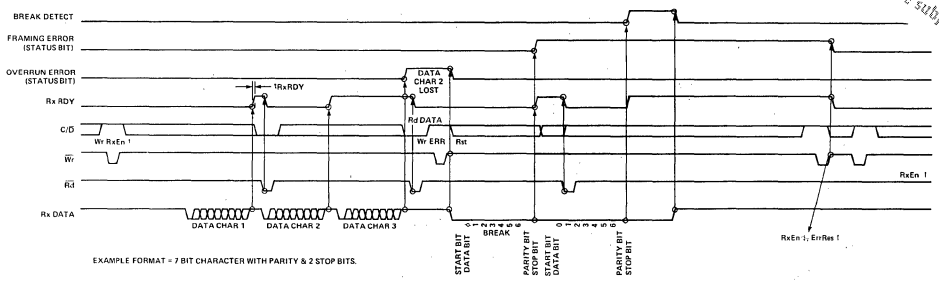
NOTE #1: T_{WC} INCLUDES THE RESPONSE TIMING OF A CONTROL BYTE.
NOTE #2: T_{CR} INCLUDES THE EFFECT OF CTS ON THE TxENBL CIRCUITRY.

Transmitter Control & Flag Timing (ASYNC Mode)



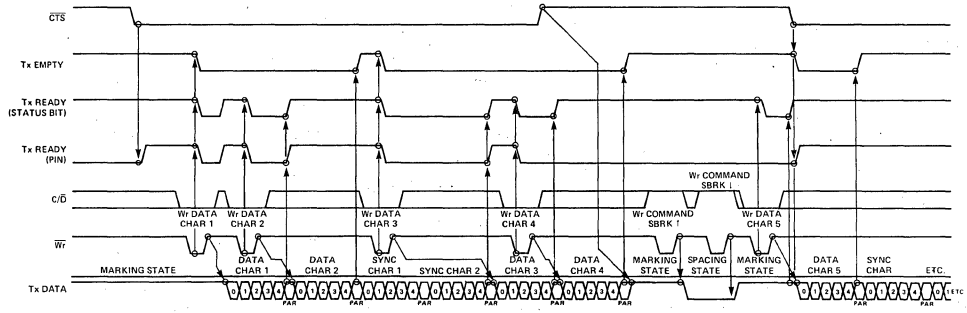
EXAMPLE FORMAT = 7 BIT CHARACTER WITH PARITY & 2 STOP BITS.

Receiver Control & Flag Timing (ASYNC Mode)



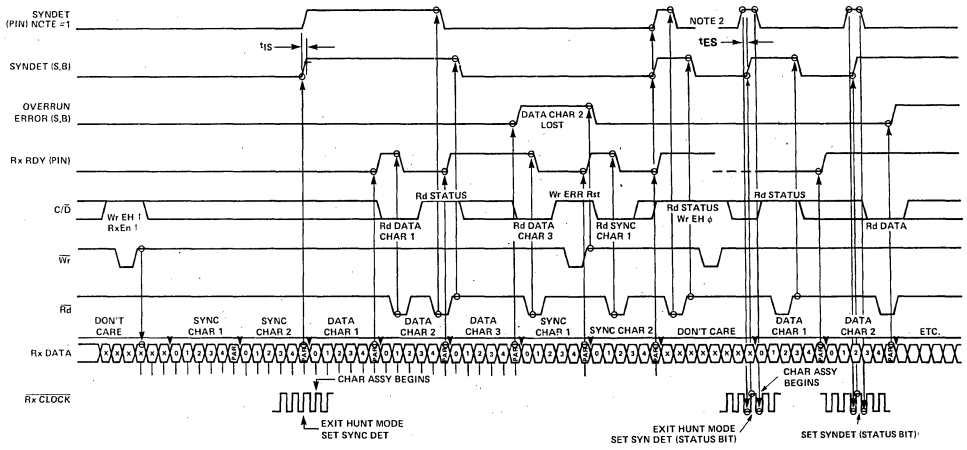
EXAMPLE FORMAT = 7 BIT CHARACTER WITH PARITY & 2 STOP BITS.

Transmitter Control & Flag Timing (SYNC Mode)



EXAMPLE FORMAT = 5 BIT CHARACTER WITH PARITY, 2 SYNC CHARACTERS.

Receiver Control & Flag Timing (SYNC Mode)



NOTE #1: INTERNAL SYNC, 2 SYNC CHARACTERS, 5 BITS, WITH PARITY
NOTE #2: EXTERNAL SYNC, 5 BITS, WITH PARITY

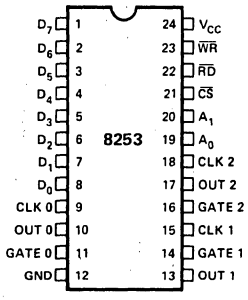
8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- **MCS—85™ Compatible 8253-5**
- **3 Independent 16-Bit Counters**
- **DC to 2 MHz**
- **Programmable Counter Modes**
- **Count Binary or BCD**
- **Single +5V Supply**
- **24-Pin Dual In-Line Package**

The Intel® 8253 is a programmable counter/timer chip designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable.

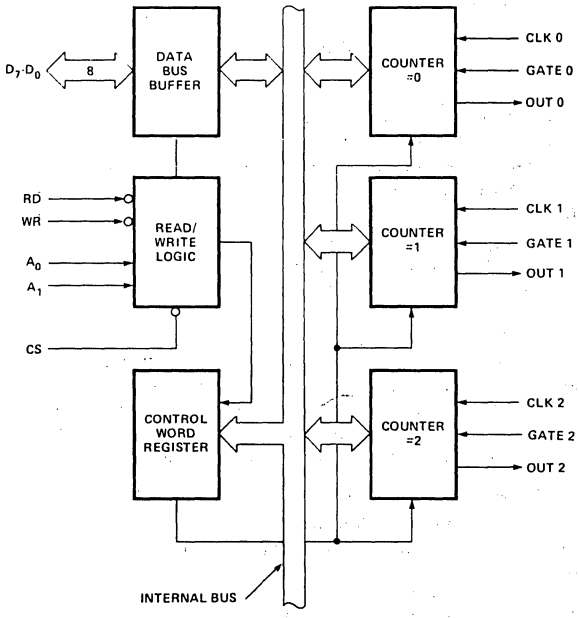
PIN CONFIGURATION

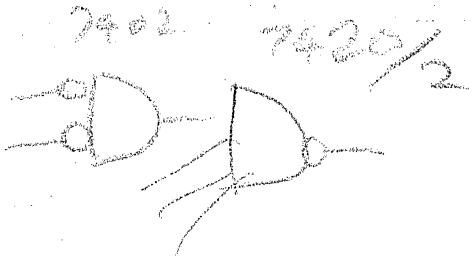


PIN NAMES

D_7-D_0	DATA BUS (8-BIT)
CLK N	COUNTER CLOCK INPUTS
GATE N	COUNTER GATE INPUTS
OUT N	COUNTER OUTPUTS
$\overline{RD}$	READ COUNTER
WR	WRITE COMMAND OR DATA
$\overline{CS}$	CHIP SELECT
A_0-A_1	COUNTER SELECT
V_{CC}	+5 VOLTS
GND	GROUND

BLOCK DIAGRAM





FUNCTIONAL DESCRIPTION

General

The 8253 is a programmable interval timer/counter specifically designed for use with the Intel™ Microcomputer systems. Its function is that of a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in systems software, the programmer configures the 8253 to match his requirements, initializes one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks. It is easy to see that the software overhead is minimal and that multiple delays can easily be maintained by assignment of priority levels.

Other counter/timer functions that are non-delay in nature but also common to most microcomputers can be implemented with the 8253.

- Programmable Rate Generator
- Event Counter
- Binary Rate Multiplier
- Real Time Clock
- Digital One-Shot
- Complex Motor Controller

Data Bus Buffer

This 3-state, bi-directional, 8-bit buffer is used to interface the 8253 to the system data bus. Data is transmitted or received by the buffer upon execution of INput or OUTput CPU instructions. The Data Bus Buffer has three basic functions.

1. Programming the MODES of the 8253.
2. Loading the count registers.
3. Reading the count values.

Read/Write Logic

The Read/Write Logic accepts inputs from the system bus and in turn generates control signals for overall device operation. It is enabled or disabled by CS so that no operation can occur to change the function unless the device has been selected by the system logic.

$\overline{RD}$ (Read)

A "low" on this input informs the 8253 that the CPU is inputting data in the form of a counters value.

$\overline{WR}$ (Write)

A "low" on this input informs the 8253 that the CPU is outputting data in the form of mode information or loading counters.

A0, A1

These inputs are normally connected to the address bus. Their function is to select one of the three counters to be operated on and to address the control word register for mode selection.

$\overline{CS}$ (Chip Select)

A "low" on this input enables the 8253. No reading or writing will occur unless the device is selected. The $\overline{CS}$ input has no effect upon the actual operation of the counters.

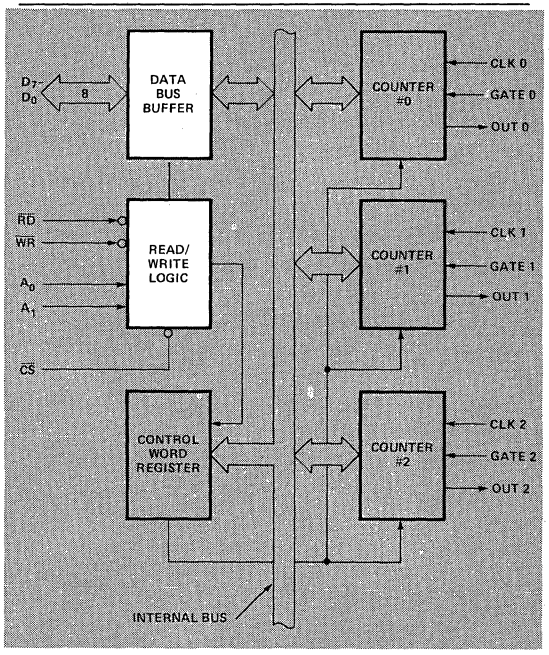


Figure 1. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter No. 0
0	0	1	0	1	Read Counter No. 1
0	0	1	1	0	Read Counter No. 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3-State

Control Word Register

The Control Word Register is selected when A0, A1 are 11. It then accepts information from the data bus buffer and stores it in a register. The information stored in this register controls the operational MODE of each counter, selection of binary or BCD counting and the loading of each count register.

The Control Word Register can only be written into; no read operation of its contents is available.

Counter #0, Counter #1, Counter #2

These three functional blocks are identical in operation so only a single Counter will be described. Each Counter consists of a single, 16-bit, pre-settable, DOWN counter. The counter can operate in either binary or BCD and its input, gate and output are configured by the selection of MODES stored in the Control Word Register.

The counters are fully independent and each can have separate Mode configuration and counting operation, binary or BCD. Also, there are special features in the control word that handle the loading of the count value so that software overhead can be minimized for these functions.

The reading of the contents of each counter is available to the programmer with simple READ operations for event counting applications and special commands and logic are included in the 8253 so that the contents of each counter can be read "on the fly" without having to inhibit the clock input.

8253 SYSTEM INTERFACE

The 8253 is a component of the Intel™ Microcomputer Systems and interfaces in the same manner as all other peripherals of the family. It is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A0, A1 connect to the A0, A1 address bus signals of the CPU. The $\overline{CS}$ can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel® 8205 for larger systems.

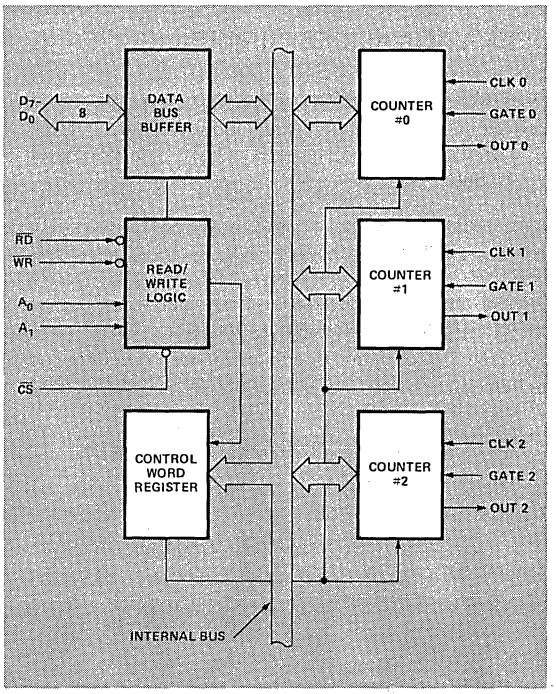


Figure 2. Block Diagram Showing Control Word Register and Counter Functions

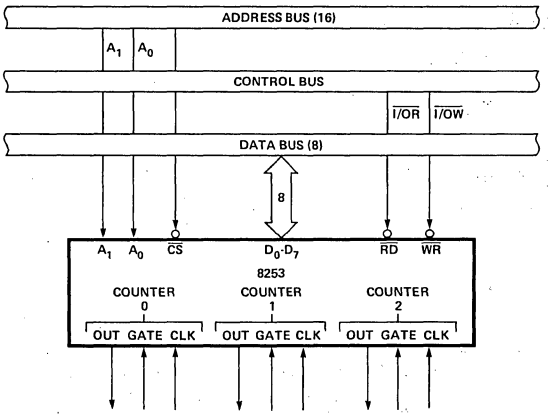


Figure 3. 8253 System Interface

OPERATIONAL DESCRIPTION

General

The complete functional definition of the 8253 is programmed by the systems software. A set of control words must be sent out by the CPU to initialize each counter of the 8253 with the desired MODE and quantity information. These control words program the MODE, Loading sequence and selection of binary or BCD counting.

Once programmed, the 8253 is ready to perform whatever timing tasks it is assigned to accomplish.

The actual counting operation of each counter is completely independent and additional logic is provided on-chip so that the usual problems associated with efficient monitoring and management of external, asynchronous events or rates to the microcomputer system have been eliminated.

Programming the 8253

All of the MODES for each counter are programmed by the systems software by simple I/O operations.

Each counter of the 8253 is individually programmed by writing a control word into the Control Word Register. (A0, A1 = 11)

Control Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Definition of Control

SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

RL — Read/Load:

RL1	RL0	
0	0	Counter Latching operation (see READ/WRITE Procedure Section)
1	0	Read/Load most significant byte only.
0	1	Read/Load least significant byte only.
1	1	Read/Load least significant byte first, then most significant byte.

M — MODE:

M2	M1	M0	
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

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

Counter Loading

The count register is not loaded until the count value is written (one or two bytes, depending on the mode selected by the RL bits), followed by a rising edge and a falling edge of the clock. Any read of the counter prior to that falling clock edge may yield invalid data.

MODE Definition

MODE 0: Interrupt on Terminal Count. The output will be initially low after the mode set operation. After the count is loaded into the selected count register, the output will remain low and the counter will count. When terminal count is reached the output will go high and remain high until the selected count register is reloaded with the mode or a new count is loaded. The counter continues to decrement after terminal count has been reached.

Rewriting a counter register during counting results in the following:

(1) Write 1st byte stops the current counting.
(2) Write 2nd byte starts the new count.

MODE 1: Programmable One-Shot. The output will go low on the count following the rising edge of the gate input.

The output will go high on the terminal count. If a new count value is loaded while the output is low it will not affect the duration of the one-shot pulse until the succeeding trigger. The current count can be read at any time without affecting the one-shot pulse.

The one-shot is retriggerable, hence the output will remain low for the full count after any rising edge of the gate input.

MODE 2: Rate Generator. Divide by N counter. The output will be low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses the present period will not be affected, but the subsequent period will reflect the new value.

The gate input, when low, will force the output high. When the gate input goes high, the counter will start from the initial count. Thus, the gate input can be used to synchronize the counter.

When this mode is set, the output will remain high until after the count register is loaded. The output then can also be synchronized by software.

MODE 3: Square Wave Rate Generator. Similar to MODE 2 except that the output will remain high until one half the count has been completed (for even numbers) and go low for the other half of the count. This is accomplished by decrementing the counter by two on the falling edge of each clock pulse. When the counter reaches terminal count, the state of the output is changed and the counter is reloaded with the full count and the whole process is repeated.

If the count is odd and the output is high, the first clock pulse (after the count is loaded) decrements the count by 1. Subsequent clock pulses decrement the clock by 2. After timeout, the output goes low and the full count is reloaded. The first clock pulse (following the reload) decrements the counter by 3. Subsequent clock pulses decrement the count by 2 until timeout. Then the whole process is repeated. In this way, if the count is odd, the output will be high for $(N+1)/2$ counts and low for $(N-1)/2$ counts.

MODE 4: Software Triggered Strobe. After the mode is set, the output will be high. When the count is loaded, the counter will begin counting. On terminal count, the output will go low for one input clock period, then will go high again.

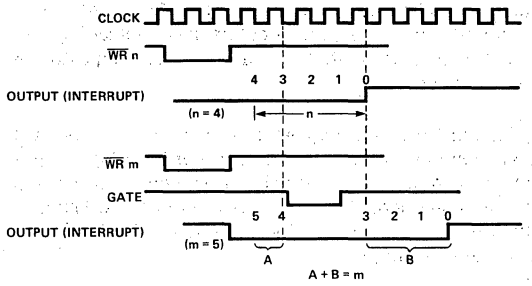
If the count register is reloaded between output pulses the present period will not be affected, but the subsequent period will reflect the new value. The count will be inhibited while the gate input is low. Reloading the counter register will restart counting beginning with the new number.

MODE 5: Hardware Triggered Strobe. The counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of any trigger.

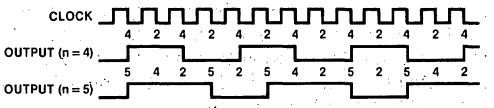
Modes \ Signal Status	Low Or Going Low	Rising	High
0	Disables counting	—	Enables counting
1	—	1) Initiates counting 2) Resets output after next clock	—
2	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
4	Disables counting	—	Enables counting
5	—	Initiates counting	—

Figure 4. Gate Pin Operations Summary

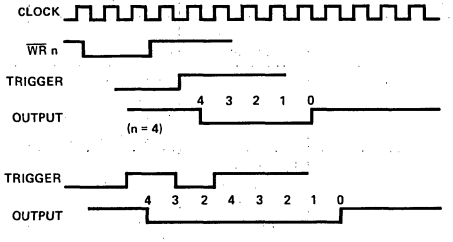
MODE 0: Interrupt on Terminal Count



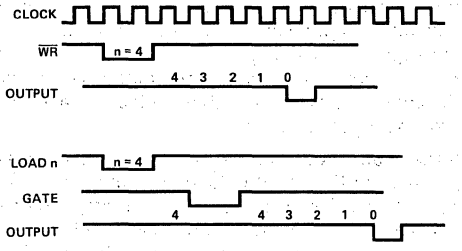
MODE 3: Square Wave Generator



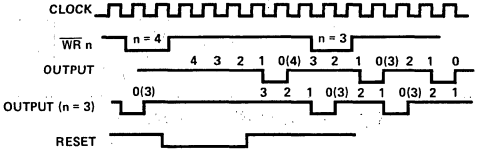
MODE 1: Programmable One-Shot



MODE 4: Software Triggered Strobe



MODE 2: Rate Generator



MODE 5: Hardware Triggered Strobe

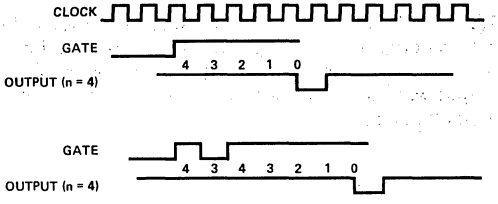


Figure 5. 8253 Timing Diagrams

8253 READ/WRITE PROCEDURE

Write Operations

The systems software must program each counter of the 8253 with the mode and quantity desired. The programmer must write out to the 8253 a MODE control word and the programmed number of count register bytes (1 or 2) prior to actually using the selected counter.

The actual order of the programming is quite flexible. Writing out of the MODE control word can be in any sequence of counter selection, e.g., counter #0 does not have to be first or counter #2 last. Each counter's MODE control word register has a separate address so that its loading is completely sequence independent. (SC0, SC1)

The loading of the Count Register with the actual count value, however, must be done in exactly the sequence programmed in the MODE control word (RL0, RL1). This loading of the counter's count register is still sequence independent like the MODE control word loading, but when a selected count register is to be loaded it must be loaded with the number of bytes programmed in the MODE control word (RL0, RL1). The one or two bytes to be loaded in the count register do not have to follow the associated MODE control word. They can be programmed at any time following the MODE control word loading as long as the correct number of bytes is loaded in order.

All counters are down counters. Thus, the value loaded into the count register will actually be decremented. Loading all zeroes into a count register will result in the maximum count (2^{16} for Binary or 10^4 for BCD). In MODE 0 the new count will not restart until the load has been completed. It will accept one of two bytes depending on how the MODE control words (RL0, RL1) are programmed. Then proceed with the restart operation.

	MODE Control Word Counter n
LSB	Count Register byte Counter n
MSB	Count Register byte Counter n

Note: Format shown is a simple example of loading the 8253 and does not imply that it is the only format that can be used.

Figure 6. Programming Format

			A1	A0
No. 1		MODE Control Word Counter 0	1	1
No. 2		MODE Control Word Counter 1	1	1
No. 3		MODE Control Word Counter 2	1	1
No. 4	LSB	Count Register Byte Counter 1	0	1
No. 5	MSB	Count Register Byte Counter 1	0	1
No. 6	LSB	Count Register Byte Counter 2	1	0
No. 7	MSB	Count Register Byte Counter 2	1	0
No. 8	LSB	Count Register Byte Counter 0	0	0
No. 9	MSB	Count Register Byte Counter 0	0	0

Note: The exclusive addresses of each counter's count register make the task of programming the 8253 a very simple matter, and maximum effective use of the device will result if this feature is fully utilized.

Figure 7. Alternate Programming Formats

Read Operations

In most counter applications it becomes necessary to read the value of the count in progress and make a computational decision based on this quantity. Event counters are probably the most common application that uses this function. The 8253 contains logic that will allow the programmer to easily read the contents of any of the three counters without disturbing the actual count in progress.

There are two methods that the programmer can use to read the value of the counters. The first method involves the use of simple I/O read operations of the selected counter. By controlling the A0, A1 inputs to the 8253 the programmer can select the counter to be read (remember that no read operation of the mode register is allowed A0, A1-11). The only requirement with this method is that in order to assure a stable count reading the actual operation of the selected counter must be inhibited either by controlling the Gate input or by external logic that inhibits the clock input. The contents of the counter selected will be available as follows:

first I/O Read contains the least significant byte (LSB).

second I/O Read contains the most significant byte (MSB).

Due to the internal logic of the 8253 it is absolutely necessary to complete the entire reading procedure. If two bytes are programmed to be read then two bytes must be read before any loading WR command can be sent to the same counter.

Read Operation Chart

A1	A0	RD	
0	0	0	Read Counter No. 0
0	1	0	Read Counter No. 1
1	0	0	Read Counter No. 2
1	1	0	Illegal

Reading While Counting

In order for the programmer to read the contents of any counter without effecting or disturbing the counting operation the 8253 has special internal logic that can be accessed using simple WR commands to the MODE register. Basically, when the programmer wishes to read the contents of a selected counter "on the fly" he loads the MODE register with a special code which latches the present count value into a storage register so that its contents contain an accurate, stable quantity. The programmer then issues a normal read command to the selected counter and the contents of the latched register is available.

MODE Register for Latching Count

A0, A1 = 11

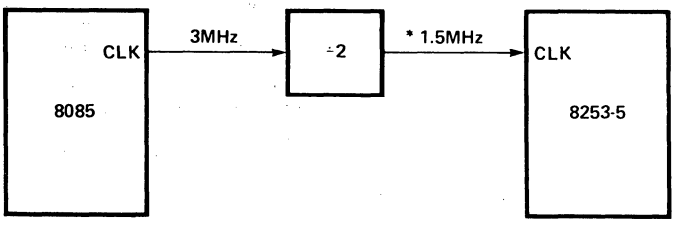
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1,SC0 — specify counter to be latched.

D5,D4 — 00 designates counter latching operation.

X — don't care.

The same limitation applies to this mode of reading the counter as the previous method. That is, it is mandatory to complete the entire read operation as programmed. This command has no effect on the counter's mode.



*If an 8085 clock output is to drive an 8253-5 clock input, it must be reduced to 2 MHz or less.

Figure 8. MCS-85™ Clock Interface*

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... −65°C to +150°C
Voltage On Any Pin
With Respect to Ground ..... −0.5 V to +7 V
Power Dissipation ..... 1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ±5%)

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC}+.5V	V	
V_{OL}	Output Low Voltage		0.45	V	Note 1
V_{OH}	Output High Voltage	2.4		V	Note 2
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V
I_{CC}	V_{CC} Supply Current		140	mA	

Note 1: 8253, I_{OL} = 1.6 mA; 8253-5, I_{OL} = 2.2 mA.
Note 2: 8253, I_{OH} = −150 μA; 8253-5, I_{OH} = −400 μA.

CAPACITANCE T_A = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	fc = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}

A.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Bus Parameters (Note 1)

Read Cycle:

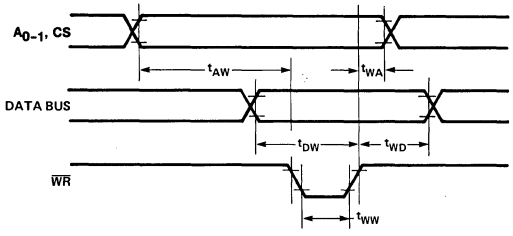
SYMBOL	PARAMETER	8253 MIN.	8253 MAX.	8253-5 MIN.	8253-5 MAX.	UNIT
t_{AR}	Address Stable Before $\overline{READ}$	50		30		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		5		ns
t_{RR}	$\overline{READ}$ Pulse Width	400		300		ns
t_{RD}	Data Delay From $\overline{READ}$[2]		300		200	ns
t_{DF}	$\overline{READ}$ to Data Floating	25	125	25	100	ns
t_{RV}	Recovery Time Between READ and Any Other Control Signal	1		1		μs

Write Cycle:

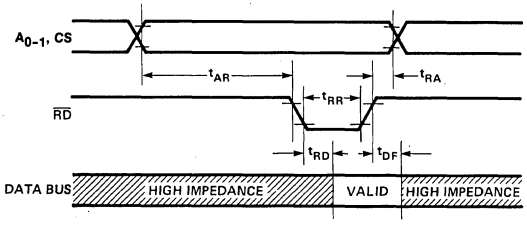
SYMBOL	PARAMETER	8253 MIN.	8253 MAX.	8253-5 MIN.	8253-5 MAX.	UNIT
t_{AW}	Address Stable Before $\overline{WRITE}$	50		30		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	30		30		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		300		ns
t_{DW}	Data Set Up Time for $\overline{WRITE}$	300		250		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		30		ns
t_{RV}	Recovery Time Between $\overline{WRITE}$ and Any Other Control Signal	1		1		μs

Notes: 1. AC timings measured at V_{OH} = 2.2, V_{OL} = 0.8
2. Test Conditions: 8253, C_L = 100pF; 8253-5: C_L = 150pF.

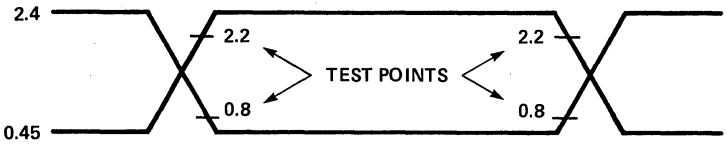
Write Timing:



Read Timing:



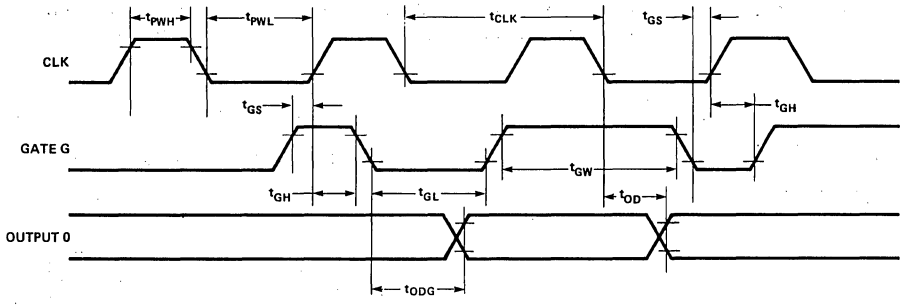
Input Waveforms for A.C. Tests:



Clock and Gate Timing:

SYMBOL	PARAMETER	8253 MIN.	8253 MAX.	8253-5 MIN.	8253-5 MAX.	UNIT
t_{CLK}	Clock Period	380	dc	380	dc	ns
t_{PWH}	High Pulse Width	230		230		ns
t_{PWL}	Low Pulse Width	150		150		ns
t_{GW}	Gate Width High	150		150		ns
t_{GL}	Gate Width Low	100		100		ns
t_{GS}	Gate Set Up Time to CLK↑	100		100		ns
t_{GH}	Gate Hold Time After CLK↑	50		50		ns
t_{OD}	Output Delay From CLK↓[1]		400		400	ns
t_{ODG}	Output Delay From Gate↓[1]		300		300	ns

Note 1: Test Conditions: 8253: C_L = 100pF; 8253-5: C_L = 150pF.

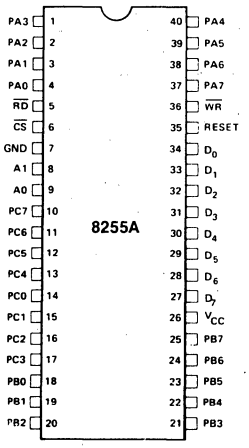


8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- **MCS-85™ Compatible 8255A-5**
- **24 Programmable I/O Pins**
- **Completely TTL Compatible**
- **Fully Compatible with Intel® Microprocessor Families**
- **Improved Timing Characteristics**
- **Direct Bit Set/Reset Capability Easing Control Application Interface**
- **40-Pin Dual In-Line Package**
- **Reduces System Package Count**
- **Improved DC Driving Capability**

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

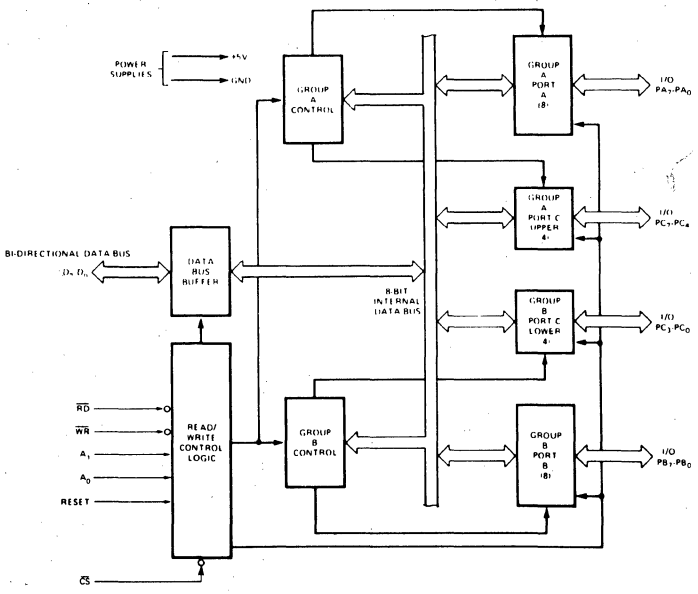
PIN CONFIGURATION



PIN NAMES

D_7–D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	Ø VOLTS

8255A BLOCK DIAGRAM



8255A FUNCTIONAL DESCRIPTION

General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel® microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

($\overline{CS}$)

Chip Select. A "low" on this input pin enables the communiction between the 8255A and the CPU.

($\overline{RD}$)

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

($\overline{WR}$)

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A_0 and A_1)

Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A_0 and A_1).

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A ⇒ DATA BUS
0	1	0	1	0	PORT B ⇒ DATA BUS
1	0	0	1	0	PORT C ⇒ DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS ⇒ PORT A
0	1	1	0	0	DATA BUS ⇒ PORT B
1	0	1	0	0	DATA BUS ⇒ PORT C
1	1	1	0	0	DATA BUS ⇒ CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS ⇒ 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS ⇒ 3-STATE

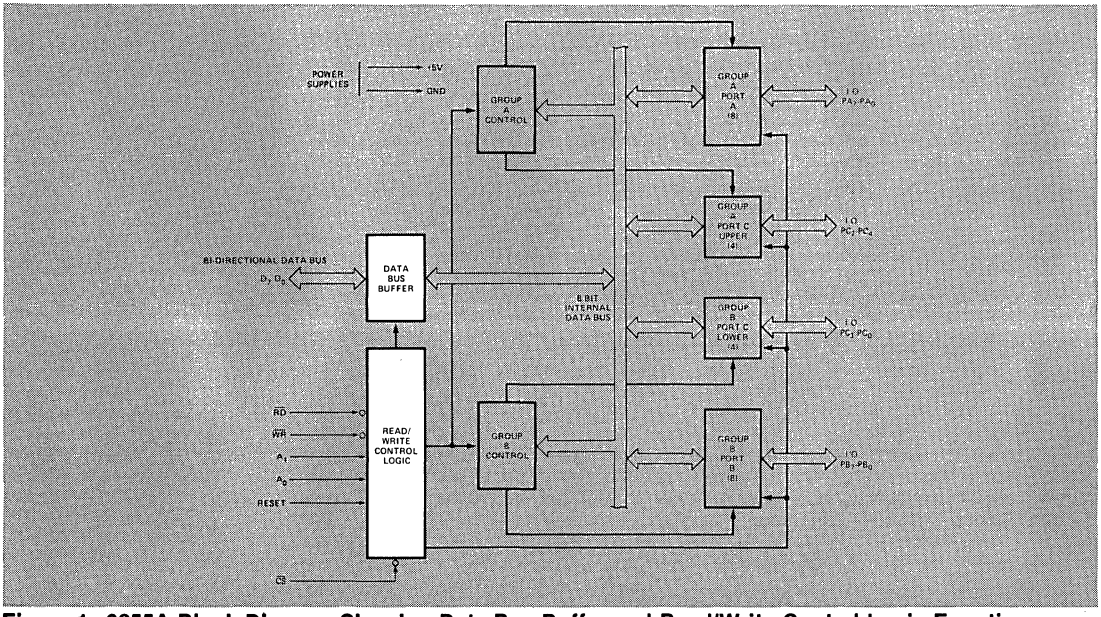


Figure 1. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

(RESET)

Reset. A "high on this input clears the control register and all ports (A, C, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A – Port A and Port C upper (C7-C4)
Control Group B – Port B and Port C lower (C3-C0)

The Control Word Register can **Only** be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.

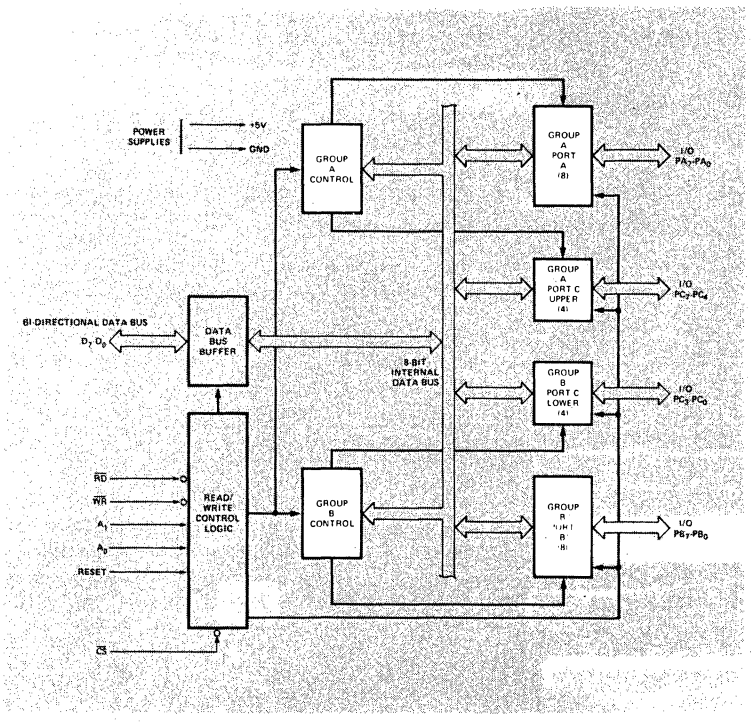
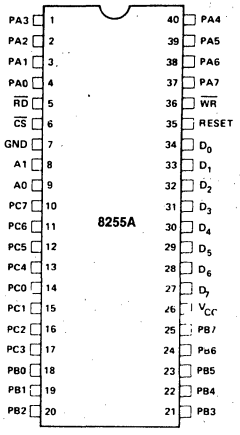


Figure 2. 8225A Block Diagram Showing Group A and Group B Control Functions

PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 — Basic Input/Output
Mode 1 — Strobed Input/Output
Mode 2 — Bi-Directional Bus

When the reset input goes "high" all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.
time.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

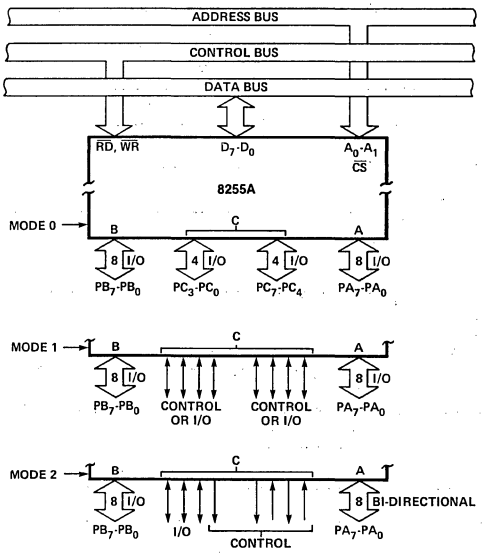


Figure 3. Basic Mode Definitions and Bus Interface

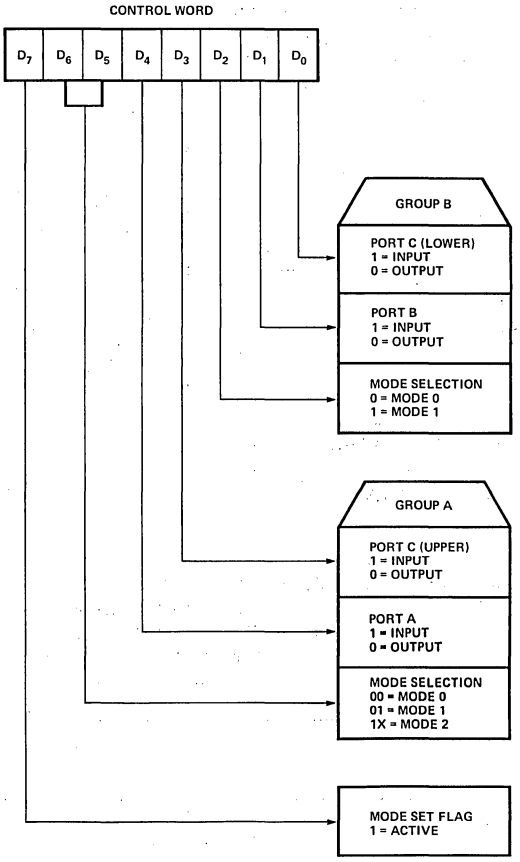


Figure 4. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

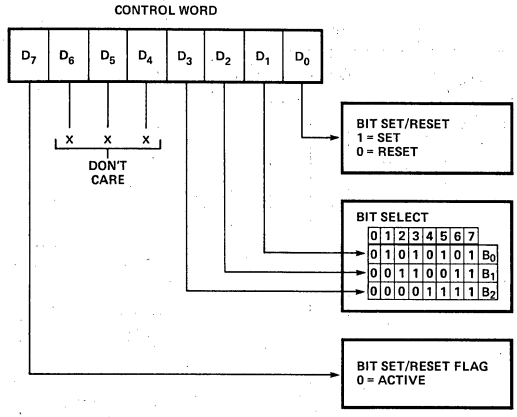


Figure 5. Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET) — INTE is SET — Interrupt enable
(BIT-RESET) — INTE is RESET — Interrupt disable

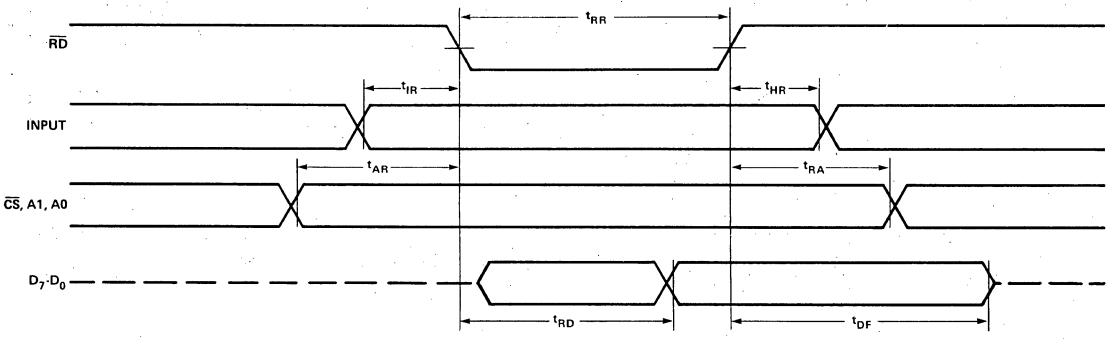
Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

Operating Modes

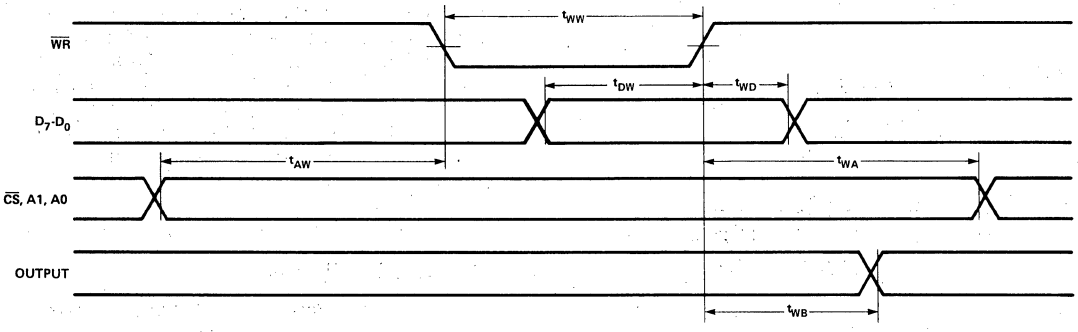
MODE 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.



MODE 0 (Basic Input)

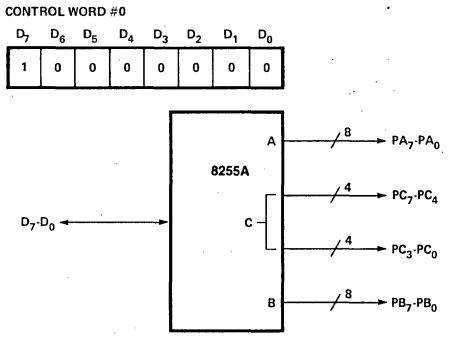


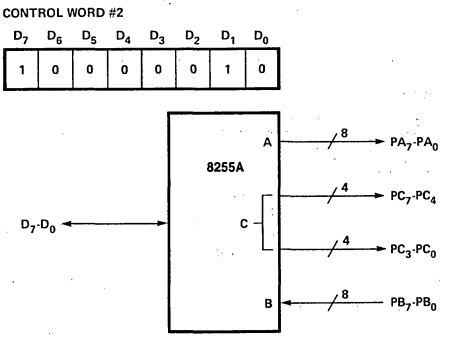
MODE 0 (Basic Output)

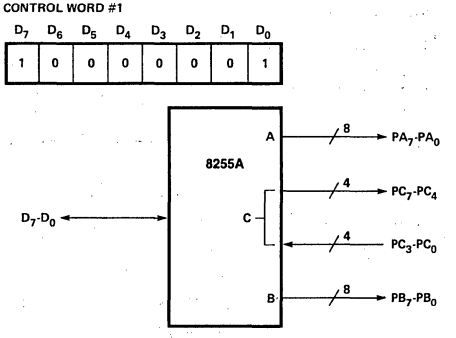
MODE 0 Port Definition

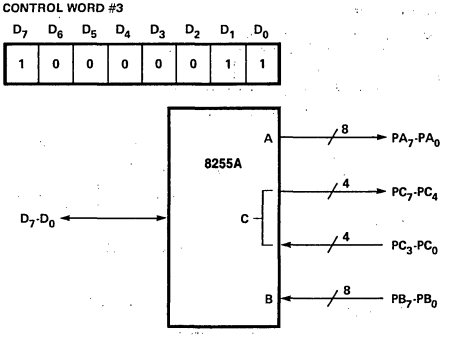
A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE 0 Configurations



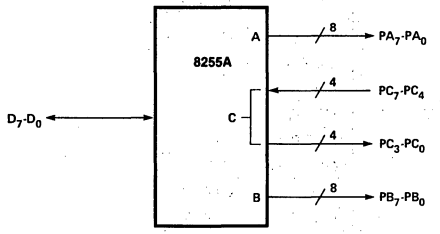






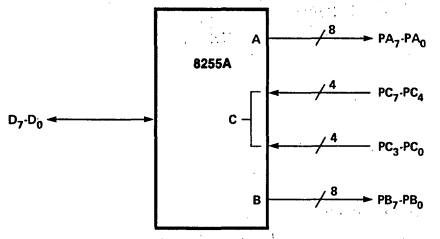
CONTROL WORD #4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



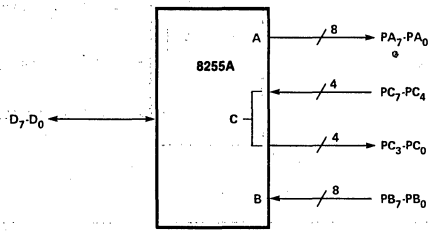
CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



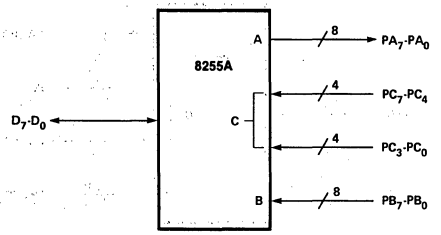
CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0



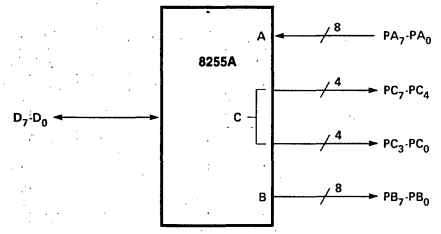
CONTROL WORD #7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



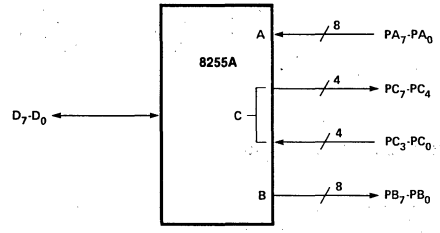
CONTROL WORD #8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0



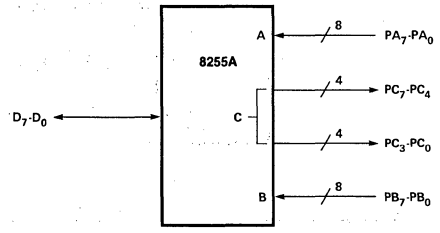
CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



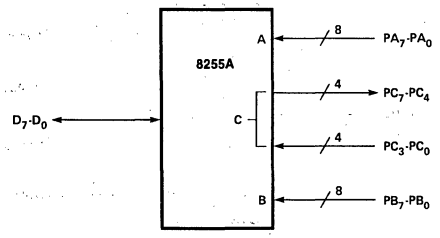
CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0



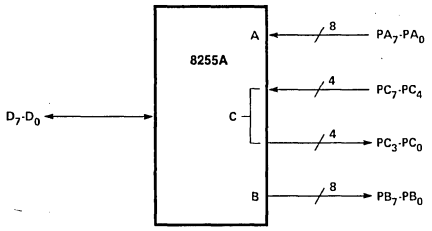
CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1



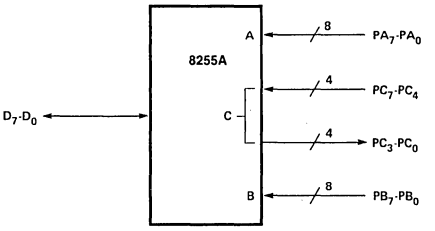
CONTROL WORD #12

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	0



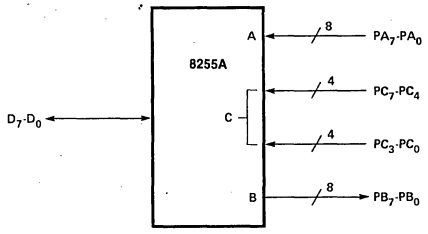
CONTROL WORD #14

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	0



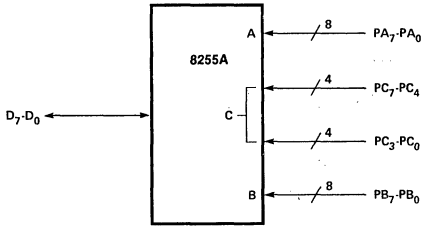
CONTROL WORD #13

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	1



CONTROL WORD #15

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	1



Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and Port B use the lines on port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

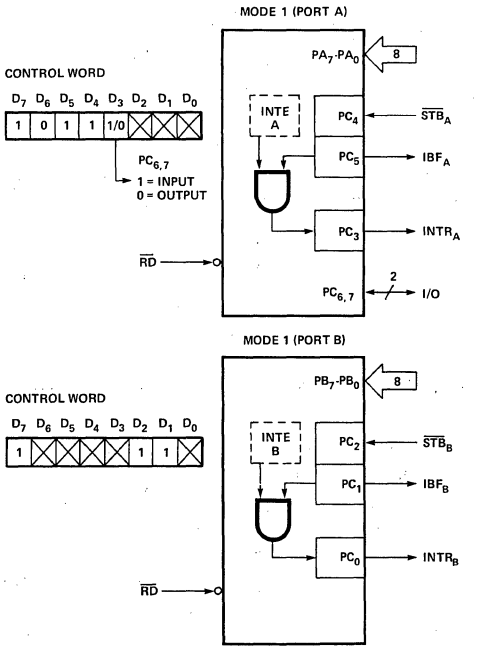


Figure 6. MODE 1 Input

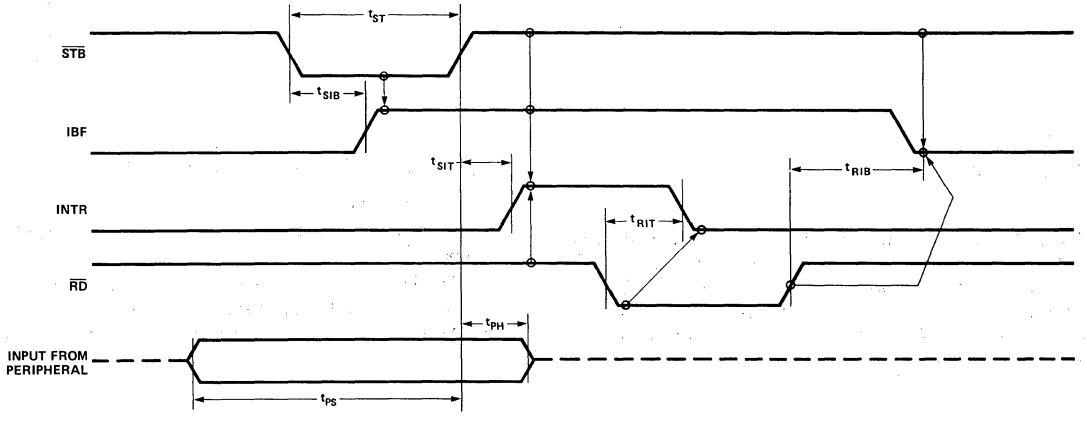


Figure 7. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{OBF}$ (Output Buffer Full F/F). The OBF output will go "low" to indicate that the CPU has written data out to the specified port. The OBF F/F will be set by the rising edge of the WR input and reset by ACK Input being low.

$\overline{ACK}$ (Acknowledge Input). A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when ACK is a "one", OBF is a "one" and INTE is a "one". It is reset by the falling edge of WR.

INTE A

Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.

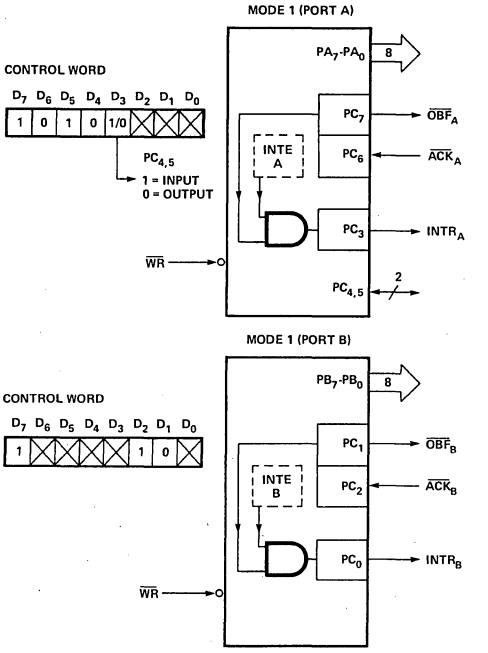


Figure 8. MODE 1 Output

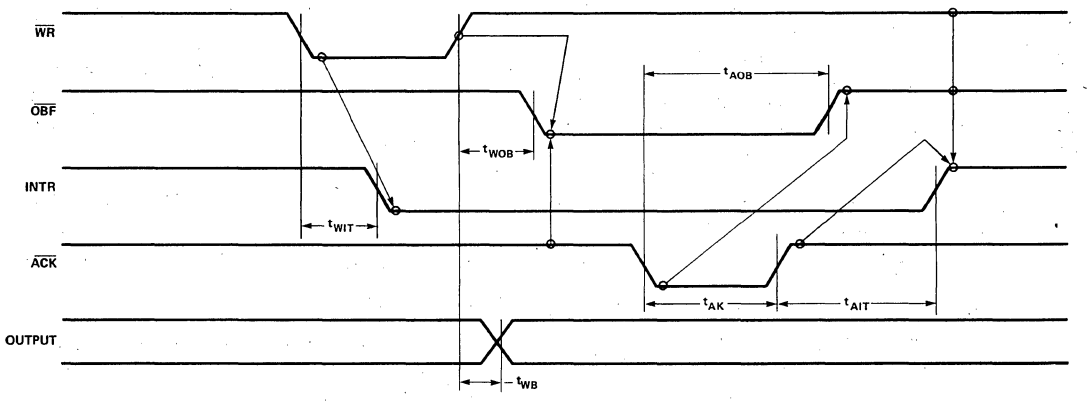
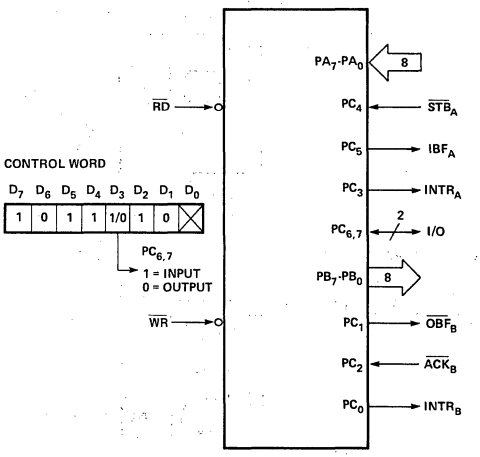


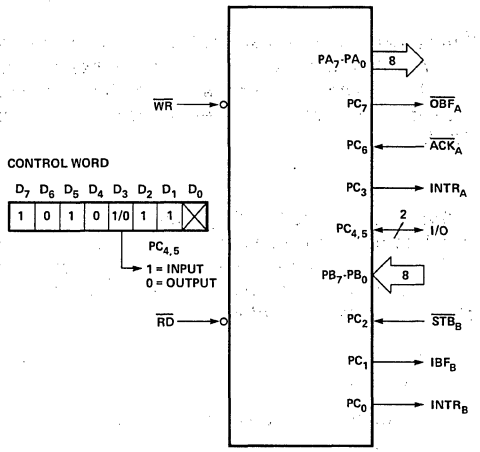
Figure 9. Mode 1 (Strobed Output)

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.



PORT A – (STROBED INPUT)
PORT B – (STROBED OUTPUT)



PORT A – (STROBED OUTPUT)
PORT B – (STROBED INPUT)

Figure 10. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Ful). The OBF output will go "low" to indicate that the CPU has written data out to port A.

$\overline{ACK}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with OBF). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input)

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

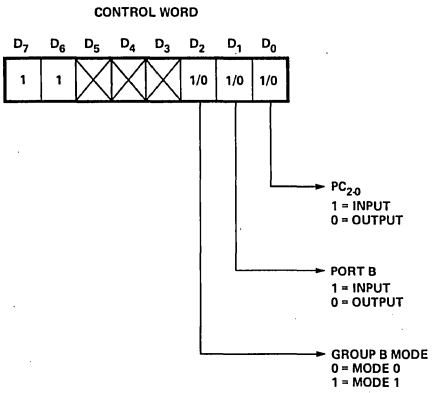


Figure 11. MODE Control Word

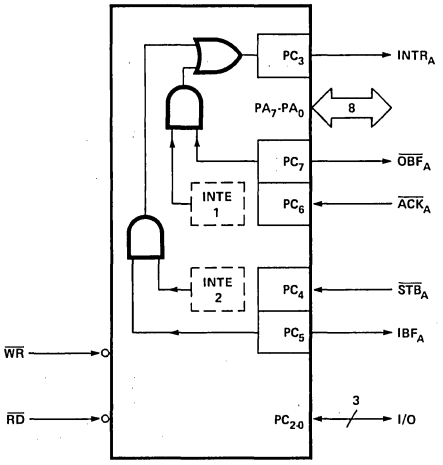


Figure 12. MODE 2

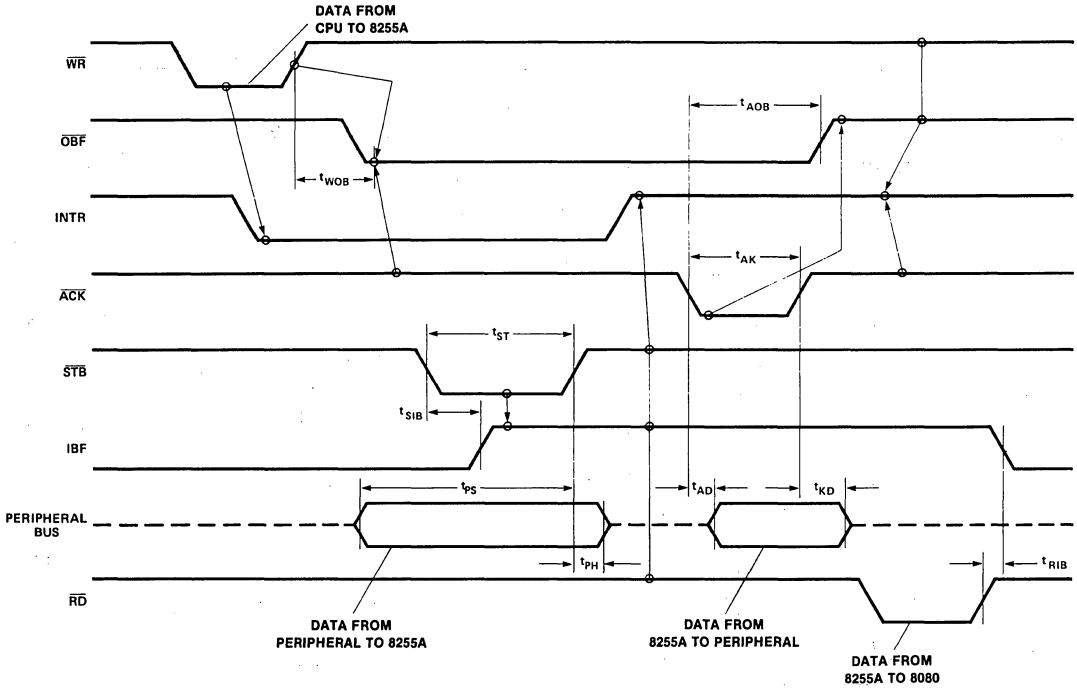


Figure 13. MODE 2 (Bidirectional)

NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

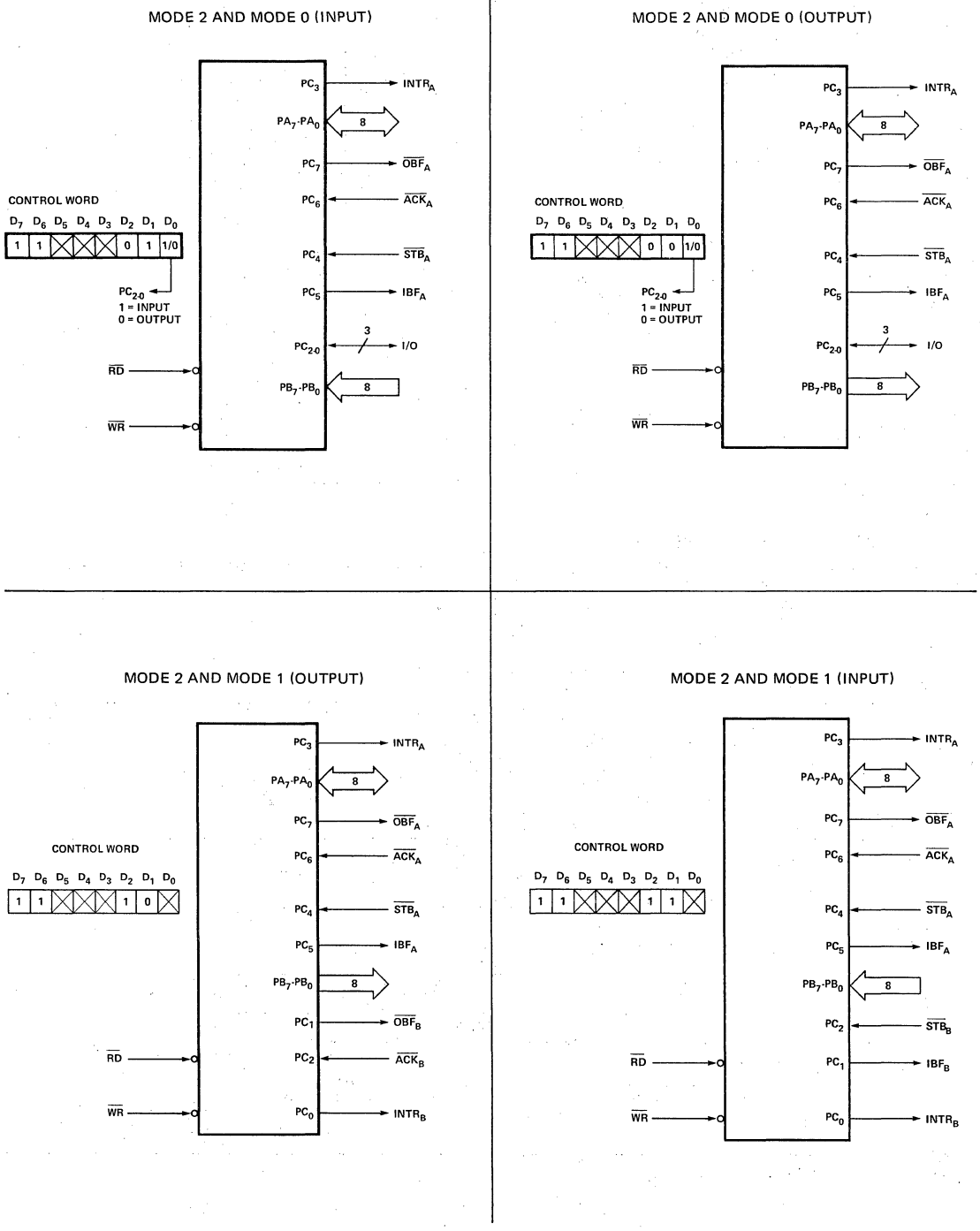


Figure 14. MODE 2 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2
	IN	OUT	IN	OUT	GROUP A ONLY
PA_0	IN	OUT	IN	OUT	↔
PA_1	IN	OUT	IN	OUT	↔
PA_2	IN	OUT	IN	OUT	↔
PA_3	IN	OUT	IN	OUT	↔
PA_4	IN	OUT	IN	OUT	↔
PA_5	IN	OUT	IN	OUT	↔
PA_6	IN	OUT	IN	OUT	↔
PA_7	IN	OUT	IN	OUT	↔
PB_0	IN	OUT	IN	OUT	—
PB_1	IN	OUT	IN	OUT	—
PB_2	IN	OUT	IN	OUT	—
PB_3	IN	OUT	IN	OUT	—
PB_4	IN	OUT	IN	OUT	—
PB_5	IN	OUT	IN	OUT	—
PB_6	IN	OUT	IN	OUT	—
PB_7	IN	OUT	IN	OUT	—
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$
PC_5	IN	OUT	IBF_A	I/O	IBF_A
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$

MODE 0 OR MODE 1 ONLY

Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs –
All input lines can be accessed during a normal Port C read.

If Programmed as Outputs –
Bits in C upper (PC_7-PC_4) must be individually accessed using the bit set/reset function.

Bits in C lower (PC_3-PC_0) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

Source Current Capability on Port B and Port C

Any set of eight output buffers, selected randomly from Ports B and C can source 1mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

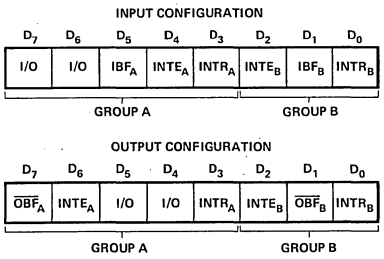


Figure 15. MODE 1 Status Word Format

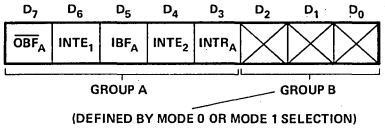


Figure 16. MODE 2 Status Word Format

APPLICATIONS OF THE 8255A

The 8255A is a very powerful tool for interfacing peripheral equipment to the microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

Each peripheral device in a microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255A is programmed by the I/O service routine and becomes an extension of the system software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the detailed operational description, a control word can easily be developed to initialize the 8255A to exactly "fit" the application. Figures 17 through 23 present a few examples of typical applications of the 8255A.

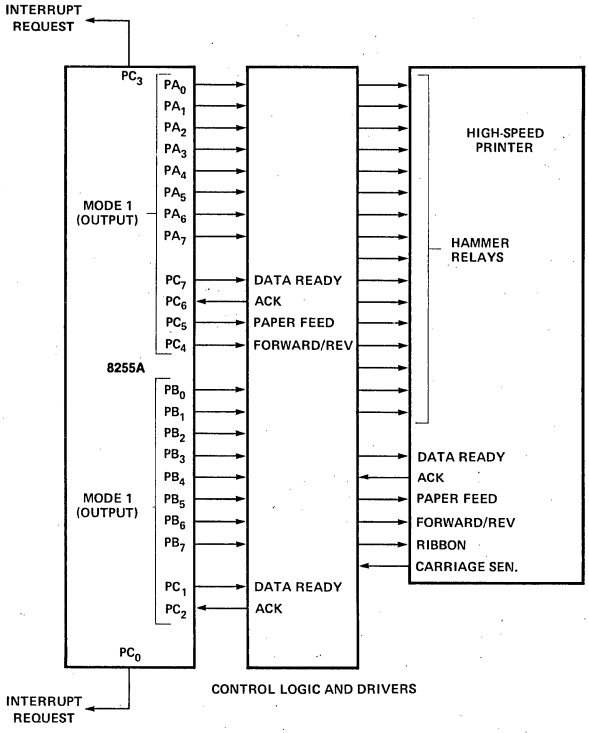


Figure 17. Printer Interface

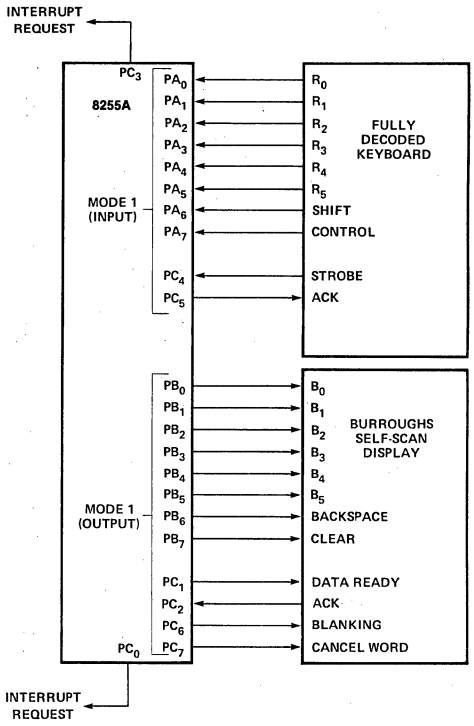


Figure 18. Keyboard and Display Interface

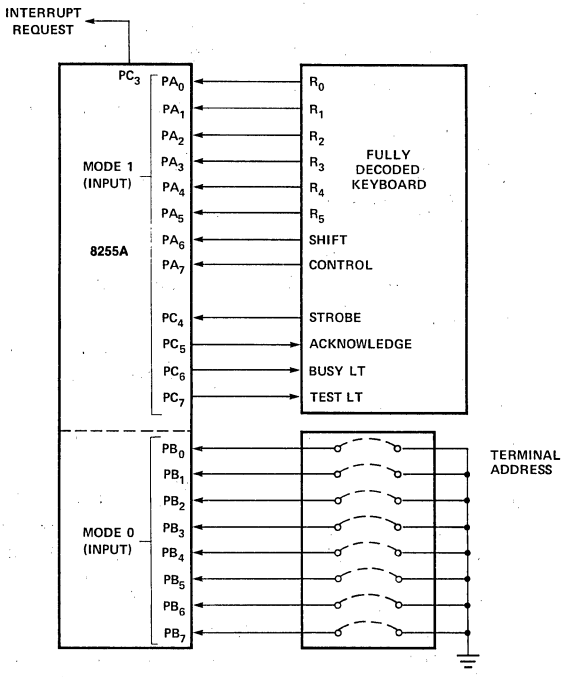


Figure 19. Keyboard and Terminal Address Interface

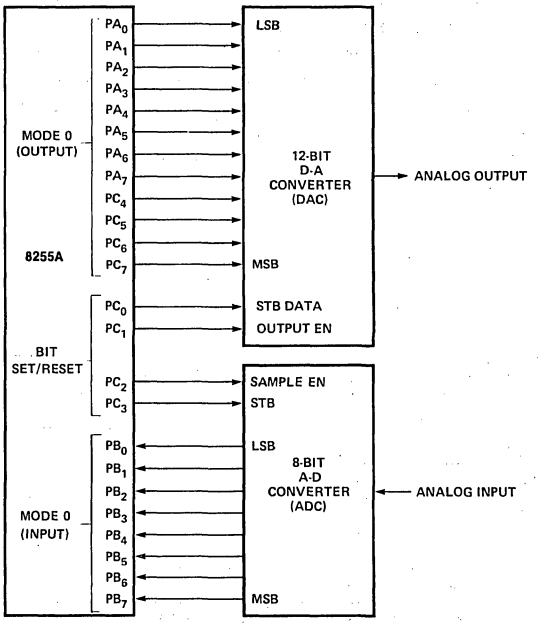


Figure 20. Digital to Analog, Analog to Digital

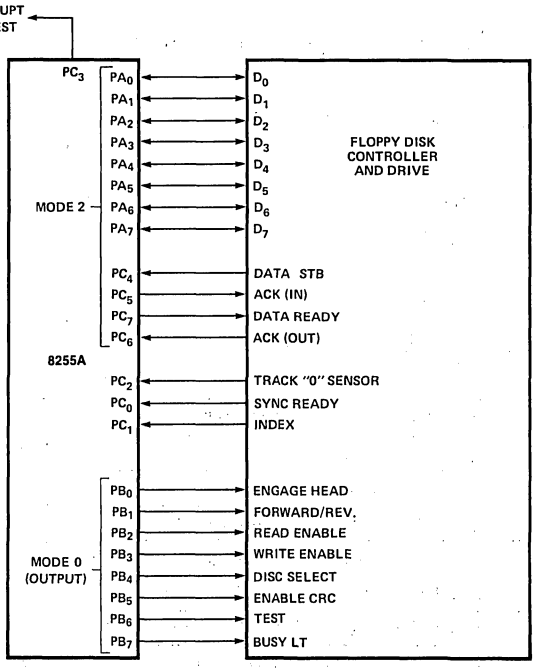


Figure 22. Basic Floppy Disc Interface

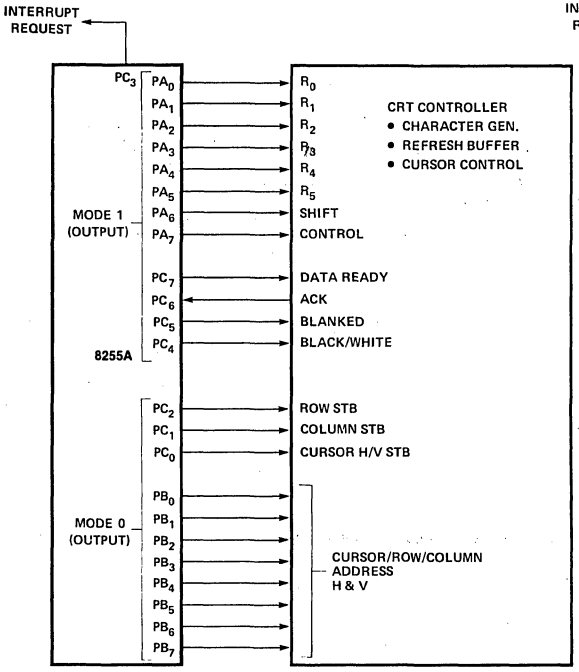


Figure 21. Basic CRT Controller Interface

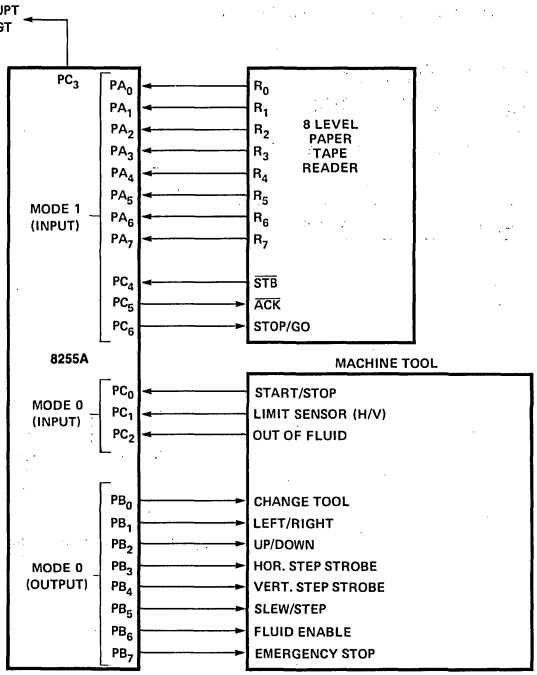


Figure 23. Machine Tool Controller Interface

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . –65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . . –0.5V to +7V
Power Dissipation . . . . . 1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±5%; GND = 0V

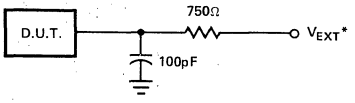
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL} (DB)	Output Low Voltage (Data Bus)		0.45	V	I_{OL} = 2.5mA
V_{OL}(PER)	Output Low Voltage (Peripheral Port)		0.45	V	I_{OL} = 1.7mA
V_{OH}(DB)	Output High Voltage (Data Bus)	2.4		V	I_{OH} = -400μA
V_{OH}(PER)	Output High Voltage (Peripheral Port)	2.4		V	I_{OH} = -200μA
I_{DAR}[1]	Darlington Drive Current	-1.0	-4.0	mA	R_{EXT} = 750Ω; V_{EXT}= 1.5V
I_{CC}	Power Supply Current		120	mA	
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V

Note 1: Available on any 8 pins from Port B and C.

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{IN}	Input Capacitance			10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND



*V_{EXT} is set at various voltages during testing to guarantee the specification.

Figure 24. Test Load Circuit (for dB)

A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = +5V ±5%; GND = 0V

NOTE:
The 8255A-5 specifications are not final. Some parametric limits are subject to change.

Bus Parameters

Read:

SYMBOL	PARAMETER	8255A MIN.	8255A MAX.	8255A-5 MIN.	8255A-5 MAX.	UNIT
t_{AR}	Address Stable Before READ	0		0		ns
t_{RA}	Address Stable After READ	0		0		ns
t_{RR}	READ Pulse Width	300		300		ns
t_{RD}	Data Valid From READ[1]		250		200	ns
t_{DF}	Data Float After READ	10	150	10	100	ns
t_{RV}	Time Between READs and/or WRITEs	850		850		ns

Write:

SYMBOL	PARAMETER	8255A MIN.	8255A MAX.	8255A-5 MIN.	8255A-5 MAX.	UNIT
t_{AW}	Address Stable Before WRITE	0		0		ns
t_{WA}	Address Stable After WRITE	20		20		ns
t_{WW}	WRITE Pulse Width	400		300		ns
t_{DW}	Data Valid to WRITE (T.E.)	100		100		ns
t_{WD}	Data Valid After WRITE	30		30		ns

Other Timings:

SYMBOL	PARAMETER	8255A MIN.	8255A MAX.	8255A-5 MIN.	8255A-5 MAX.	UNIT
t_{WB}	WR = 1 to Output[1]		350		350	ns
t_{IR}	Peripheral Data Before RD	0		0		ns
t_{HR}	Peripheral Data After RD	0		0		ns
t_{AK}	ACK Pulse Width	300		300		ns
t_{ST}	STB Pulse Width	500		500		ns
t_{PS}	Per. Data Before T.E. of STB	0		0		ns
t_{PH}	Per. Data After T.E. of STB	180		180		ns
t_{AD}	ACK = 0 to Output[1]		300		300	ns
t_{KD}	ACK = 1 to Output Float	20	250	20	250	ns
t_{WOB}	WR = 1 to OBF = 0[1]		650		650	ns
t_{AOB}	ACK = 0 to OBF = 1[1]		350		350	ns
t_{SIB}	STB = 0 to IBF = 1[1]		300		300	ns
t_{RIB}	RD = 1 to IBF = 0[1]		300		300	ns
t_{RIT}	RD = 0 to INTR = 0[1]		400		400	ns
t_{SIT}	STB = 1 to INTR = 1[1]		300		300	ns
t_{AIT}	ACK = 1 to INTR = 1[1]		350		350	ns
t_{WIT}	WR = 0 to INTR = 0[1]		850		850	ns

Notes: 1. Test Conditions: 8255A: C_L = 100pF; 8255A-5: C_L = 150pF.
2. Period of Reset pulse must be at least 50μs during or after power on. Subsequent Reset pulse can be 500 ns min.

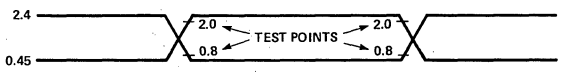


Figure 25. Input Waveforms for A.C. Tests

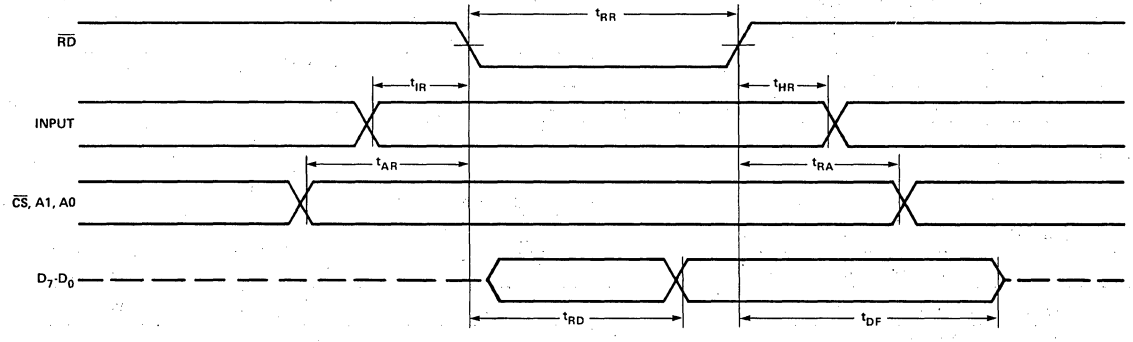


Figure 26. MODE 0 (Basic Input)

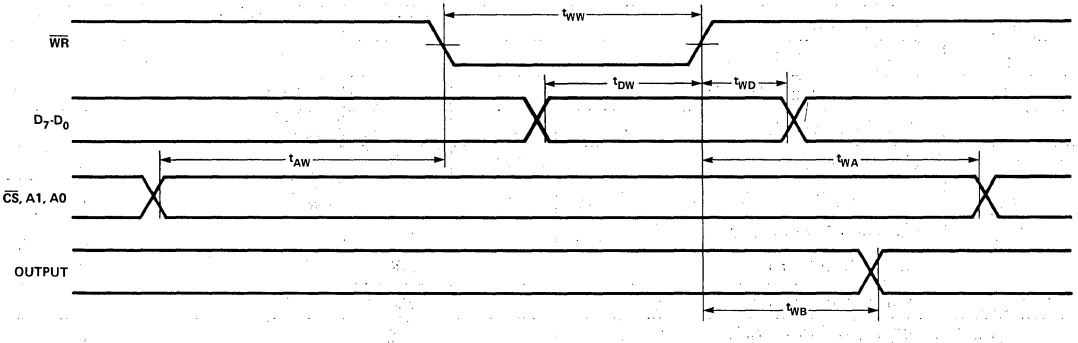


Figure 27. MODE 0 (Basic Output)

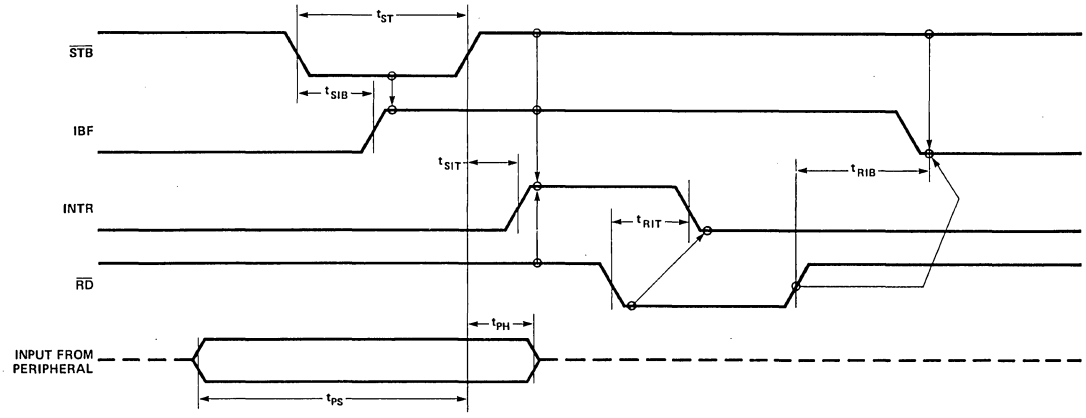


Figure 28. MODE 1 (Strobed Inut)

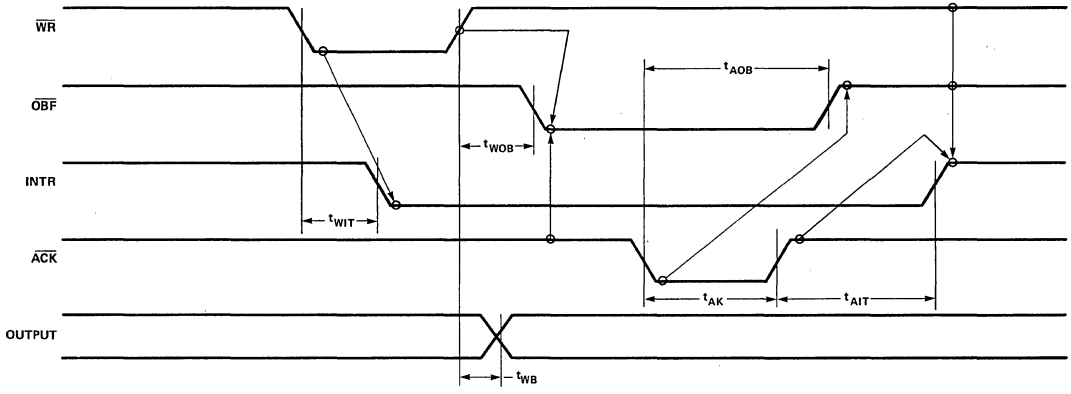


Figure 29. MODE 1 (Strobed Output)

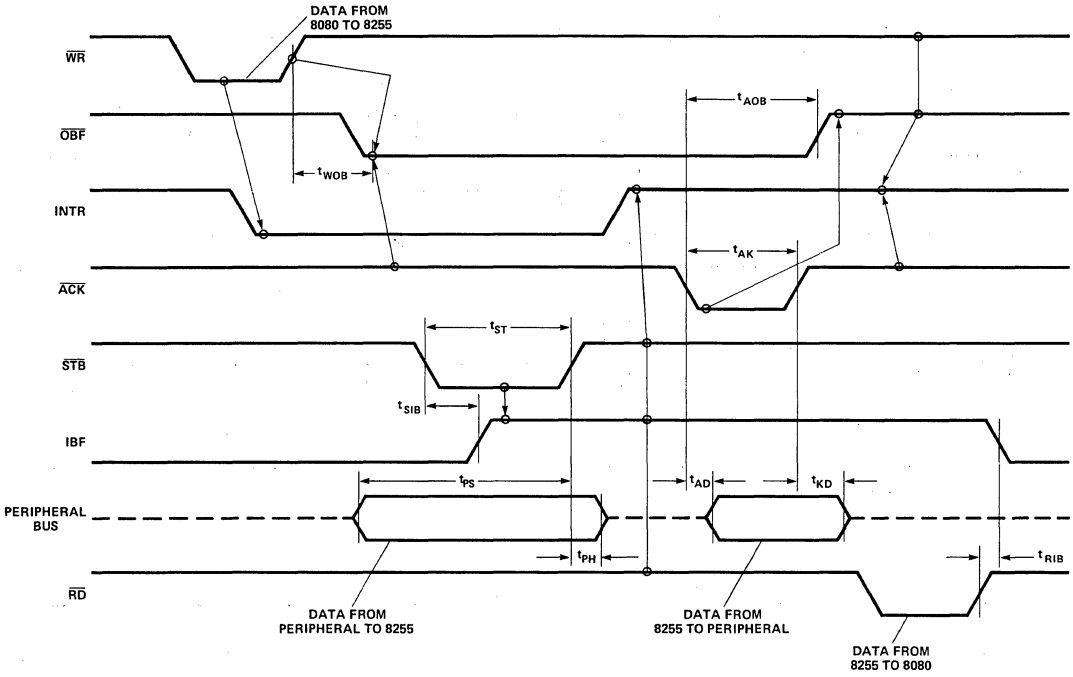


Figure 30. MODE 2 (Bidirectional)

NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

intel®

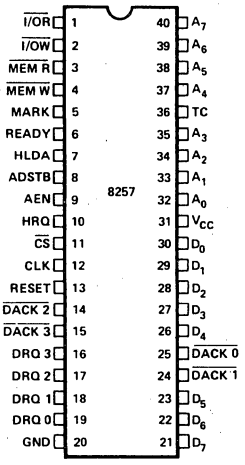
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8257/8257-5
PROGRAMMABLE DMA CONTROLLER

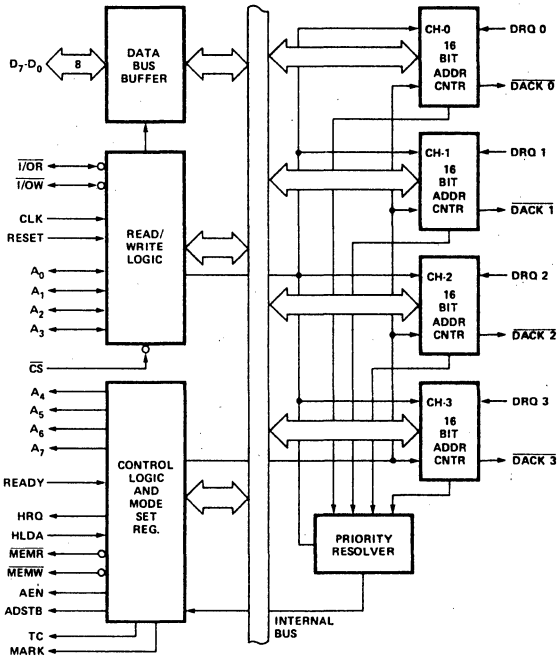
- **MCS-85™ Compatible 8257-5**
- **4-Channel DMA Controller**
- **Priority DMA Request Logic**
- **Channel Inhibit Logic**
- **Terminal Count and Modulo 128 Outputs**
- **Single TTL Clock**
- **Single +5V Supply**
- **Auto Load Mode**

The Intel® 8257 is a 4-channel direct memory access (DMA) controller. It is specifically designed to simplify the transfer of data at high speeds for the Intel® microcomputer systems. Its primary function is to generate, upon a peripheral request, a sequential memory address which will allow the peripheral to read or write data directly to or from memory. Acquisition of the system bus in accomplished via the CPU's hold function. The 8257 has priority logic that resolves the peripherals requests and issues a composite hold request to the CPU. It maintains the DMA cycle count for each channel and outputs a control signal to notify the peripheral that the programmed number of DMA cycles is complete. Other output control signals simplify sectored data transfers. The 8257 represents a significant savings in component count for DMA-based microcomputer systems and greatly simplifies the transfer of data at high speed between peripherals and memories.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

D_7-D_0	DATA BUS
A_7-A_0	ADDRESS BUS
$\overline{I/OR}$	I/O READ
$\overline{I/OW}$	I/O WRITE
$\overline{MEMR}$	MEMORY READ
$\overline{MEMW}$	MEMORY WRITE
CLK	CLOCK INPUT
RESET	RESET INPUT
READY	READY
HRQ	HOLD REQUEST (TO 8080A)
HLDA	HOLD ACKNOWLEDGE (FROM 8080A)

AEN	ADDRESS ENABLE
ADSTB	ADDRESS STROBE
TC	TERMINAL COUNT
MARK	MODULO 128 MARK
DRQ_3-DRQ_0	DMA REQUEST INPUT
$\overline{DACK_3}$-$\overline{DACK_0}$	DMA ACKNOWLEDGE OUT
$\overline{CS}$	CHIP SELECT
V_{CC}	+5 VOLTS
GND	GROUND

FUNCTIONAL DESCRIPTION

General

The 8257 is a programmable, Direct Memory Access (DMA) device which, when coupled with a single Intel® 8212 I/O port device, provides a complete four-channel DMA controller for use in Intel® microcomputer systems. After being initialized by software, the 8257 can transfer a block of data, containing up to 16,384 bytes, between memory and a peripheral device directly, without further intervention required of the CPU. Upon receiving a DMA transfer request from an enabled peripheral, the 8257:

1. Acquires control of the system bus.
2. Acknowledges that requesting peripheral which is connected to the highest priority channel.
3. Outputs the least significant eight bits of the memory address onto system address lines A_0-A_7, outputs the most significant eight bits of the memory address to the 8212 I/O port via the data bus (the 8212 places these address bits on lines A_8-A_{15}), and
4. Generates the appropriate memory and I/O read/write control signals that cause the peripheral to receive or deposit a data byte directly from or to the addressed location in memory.

The 8257 will retain control of the system bus and repeat the transfer sequence, as long as a peripheral maintains its DMA request. Thus, the 8257 can transfer a block of data to/from a high speed peripheral (e.g., a sector of data on a floppy disk) in a single "burst". When the specified number of data bytes have been transferred, the 8257 activates its Terminal Count (TC) output, informing the CPU that the operation is complete.

The 8257 offers three different modes of operation: (1) DMA read, which causes data to be transferred from memory to a peripheral; (2) DMA write, which causes data to be transferred from a peripheral to memory; and (3) DMA verify, which does not actually involve the transfer of data. When an 8257 channel is in the DMA verify mode, it will respond the same as described for transfer operations, except that no memory or I/O read/write control signals will be generated, thus preventing the transfer of data. The 8257, however, will gain control of the system bus and will acknowledge the peripheral's DMA request for each DMA cycle. The peripheral can use these acknowledge signals to enable an internal access of each byte of a data block in order to execute some verification procedure, such as the accumulation of a CRC (Cyclic Redundancy Code) checkword. For example, a block of DMA verify cycles might follow a block of DMA read cycles (memory to peripheral) to allow the peripheral to verify its newly acquired data.

Block Diagram Description

1. DMA Channels

The 8257 provides four separate DMA channels (labeled CH-0 to CH-3). Each channel includes two sixteen-bit registers: (1) a DMA address register, and (2) a terminal count register. Both registers must be initialized before a channel is enabled. The DMA address register is loaded with the address of the first memory location to be accessed. The value loaded into the low-order 14-bits of the terminal count register specifies the number of DMA cycles minus one before the Terminal Count (TC) output is activated. For instance, a terminal count of 0 would cause the TC output to be active in the first DMA cycle for that channel. In general, if N = the number of desired DMA cycles, load the value N-1 into the low-order 14-bits of the terminal count register. The most significant two bits of the terminal count register specify the type of DMA operation for that channel.

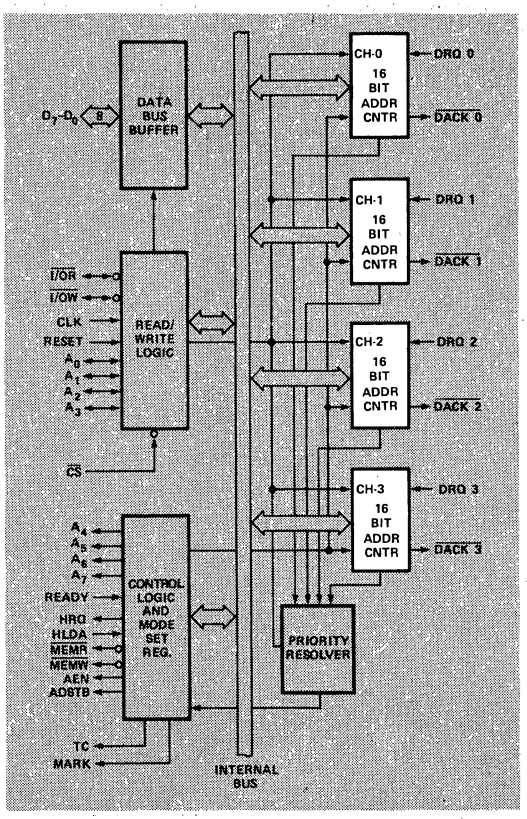


Figure 1. 8257 Block Diagram Showing DMA Channels

These two bits are not modified during a DMA cycle, but can be changed between DMA blocks.

Each channel accepts a DMA Request (DRQn) input and provides a DMA Acknowledge (DACKn) output.

(DRQ 0-DRQ 3)

DMA Request: These are individual asynchronous channel request inputs used by the peripherals to obtain a DMA cycle. If not in the rotating priority mode then DRQ 0 has the highest priority and DRQ 3 has the lowest. A request can be generated by raising the request line and holding it high until DMA acknowledge. For multiple DMA cycles (Burst Mode) the request line is held high until the DMA acknowledge of the last cycle arrives.

($\overline{\text{DACK 0 - DACK 3}}$)

DMA Acknowledge: An active low level on the acknowledge output informs the peripheral connected to that channel that it has been selected for a DMA cycle. The $\overline{\text{DACK}}$ output acts as a "chip select" for the peripheral device requesting service. This line goes active (low) and inactive (high) once for each byte transferred even if a burst of data is being transferred.

2. Data Bus Buffer

This three-state, bi-directional, eight bit buffer interfaces the 8257 to the system data bus.

(D_0-D_7)

Data Bus Lines: These are bi-directional three-state lines. When the 8257 is being programmed by the CPU, eight-bits of data for a DMA address register, a terminal count register or the Mode Set register are received on the data bus. When the CPU reads a DMA address register, a terminal count register or the Status register, the data is sent to the CPU over the data bus. During DMA cycles (when the 8257 is the bus master), the 8257 will output the most significant eight-bits of the memory address (from one of the DMA address registers) to the 8212 latch via the data bus. These address bits will be transferred at the beginning of the DMA cycle; the bus will then be released to handle the memory data transfer during the balance of the DMA cycle.

BIT 15	BIT 14	TYPE OF DMA OPERATION
0	0	Verify DMA Cycle
0	1	Write DMA Cycle
1	0	Read DMA Cycle
1	1	(Illegal)

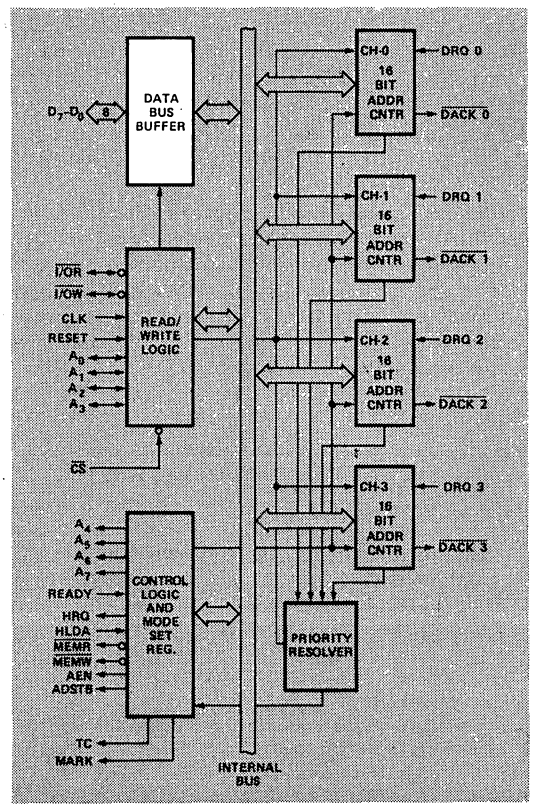


Figure 2. 8257 Block Diagram Showing Data Bus Buffer

3. Read/Write Logic

When the CPU is programming or reading one of the 8257's registers (i.e., when the 8257 is a "slave" device on the system bus), the Read/Write Logic accepts the I/O Read ($\overline{I/OR}$) or I/O Write ($\overline{I/OW}$) signal, decodes the least significant four address bits, (A_0-A_3), and either writes the contents of the data bus into the addressed register (if $\overline{I/OW}$ is true) or places the contents of the addressed register onto the data bus (if $\overline{I/OR}$ is true).

During DMA cycles (i.e., when the 8257 is the bus "master"), the Read/Write Logic generates the I/O read and memory write (DMA write cycle) or I/O Write and memory read (DMA read cycle) signals which control the data link with the peripheral that has been granted the DMA cycle.

Note that during DMA transfers Non-DMA I/O devices should be de-selected (disabled) using "AEN" signal to inhibit I/O device decoding of the memory address as an erroneous device address.

($\overline{I/OR}$)

I/O Read: An active-low, bi-directional three-state line. In the "slave" mode, it is an input which allows the 8-bit status register or the upper/lower byte of a 16-bit DMA address register or terminal count register to be read. In the "master" mode, $\overline{I/OR}$ is a control output which is used to access data from a peripheral during the DMA write cycle.

($\overline{I/OW}$)

I/O Write: An active-low, bi-directional three-state line. In the "slave" mode, it is an input which allows the contents of the data bus to be loaded into the 8-bit mode set register or the upper/lower byte of a 16-bit DMA address register or terminal count register. In the "master" mode, $\overline{I/OW}$ is a control output which allows data to be output to a peripheral during a DMA read cycle.

(CLK)

Clock Input: Generally from an Intel® 8224 Clock Generator device. (ϕ2 TTL) or Intel® 8085A CLK output.

(RESET)

Reset: An asynchronous input (generally from an 8224 or 8085 device) which disables all DMA channels by clearing the mode register and 3-states all control lines.

(A_0-A_3)

Address Lines: These least significant four address lines are bi-directional. In the "slave" mode they are inputs which select one of the registers to be read or programmed. In the "master" mode, they are outputs which constitute the least significant four bits of the 16-bit memory address generated by the 8257.

($\overline{CS}$)

Chip Select: An active-low input which enables the I/O Read or I/O Write input when the 8257 is being read or programmed in the "slave" mode. In the "master" mode, $\overline{CS}$ is automatically disabled to prevent the chip from selecting itself while performing the DMA function.

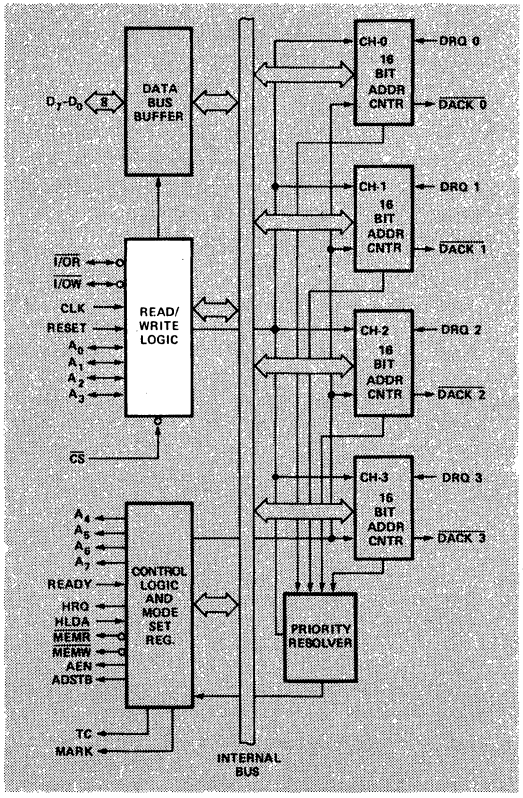


Figure 3. 8257 Block Diagram Showing Read/Write Logic Function

4. Control Logic

This block controls the sequence of operations during all DMA cycles by generating the appropriate control signals and the 16-bit address that specifies the memory location to be accessed.

(A_4-A_7)

Address Lines: These four address lines are three-state outputs which constitute bits 4 through 7 of the 16-bit memory address generated by the 8257 during all DMA cycles.

(READY)

Ready: This asynchronous input is used to elongate the memory read and write cycles in the 8257 with wait states if the selected memory requires longer cycles.

(HRQ)

Hold Request: This output requests control of the system bus. In systems with only one 8257, HRQ will normally be applied to the HOLD input on the CPU.

(HLDA)

Hold Acknowledge: This input from the CPU indicates that the 8257 has acquired control of the system bus.

($\overline{MEMR}$)

Memory Read: This active-low three-state output is used to read data from the addressed memory location during DMA Read cycles.

($\overline{MEMW}$)

Memory Write: This active-low three-state output is used to write data into the addressed memory location during DMA Write cycles.

(ADSTB)

Address Strobe: This output strobes the most significant byte of the memory address into the 8212 device from the data bus.

(AEN)

Address Enable: This output is used to disable (float) the System Data Bus and the System Control Bus. It may also be used to disable (float) the System Address Bus by use of an enable on the Address Bus drivers in systems to inhibit non-DMA devices from responding during DMA cycles. It may be further used to isolate the 8257 data bus from the System Data Bus to facilitate the transfer of the 8 most significant DMA address bits over the 8257 data I/O pins without subjecting the System Data Bus to any timing constraints for the transfer. When the 8257 is used in an I/O device structure (as opposed to memory mapped), this AEN output should be used to disable the selection of an I/O device when the DMA address is on the address bus. The I/O device selection should be determined by the DMA acknowledge outputs for the 4 channels.

(TC)

Terminal Count: This output notifies the currently selected peripheral that the present DMA cycle should be the last cycle for this data block. If the TC STOP bit in the Mode Set register is set, the selected channel will be automatically disabled at the end of that DMA cycle. TC is activated when the 14-bit value in the selected channel's terminal count register equals zero. Recall that the low-order 14-bits of the terminal count register should be loaded with the values (n-1), where n = the desired number of the DMA cycles.

(MARK)

Modulo 128 Mark: This output notifies the selected peripheral that the current DMA cycle is the 128th cycle since the previous MARK output. MARK always occurs at 128 (and all multiples of 128) cycles from the end of the data block. Only if the total number of DMA cycles (n) is evenly divisable by 128 (and the terminal count register was loaded with n-1), will MARK occur at 128 (and each succeeding multiple of 128) cycles from the beginning of the data block.

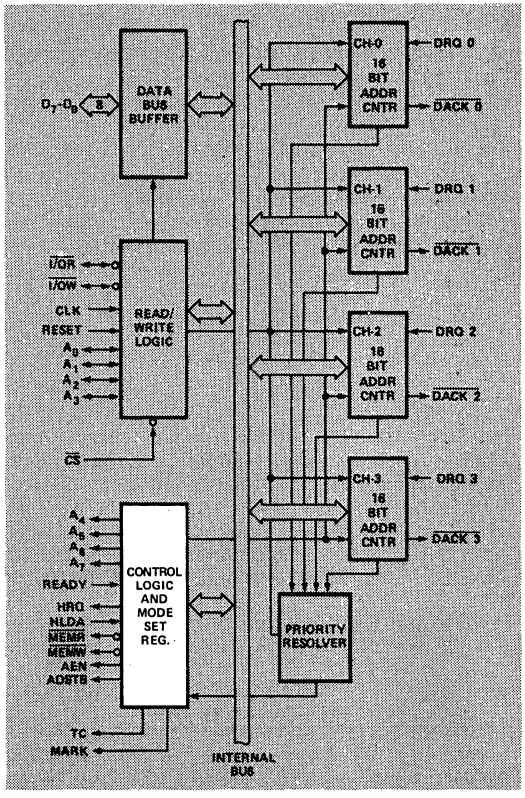
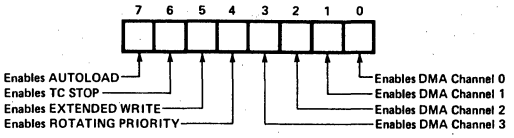


Figure 4. 8257 Block Diagram Showing Control Logic and Mode Set Register

5. Mode Set Register

When set, the various bits in the Mode Set register enable each of the four DMA channels, and allow four different options for the 8257:

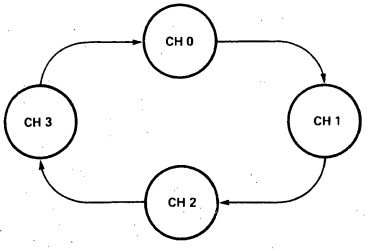


The Mode Set register is normally programmed by the CPU after the DMA address register(s) and terminal count register(s) are initialized. The Mode Set Register is cleared by the RESET input, thus disabling all options, inhibiting all channels, and preventing bus conflicts on power-up. A channel should not be left enabled unless its DMA address and terminal count registers contain valid values; otherwise, an inadvertent DMA request (DRQn) from a peripheral could initiate a DMA cycle that would destroy memory data.

The various options which can be enabled by bits in the Mode Set register are explained below:

Rotating Priority Bit 4

In the Rotating Priority Mode, the priority of the channels has a circular sequence. After each DMA cycle, the priority of each channel changes. The channel which had just been serviced will have the lowest priority.



If the ROTATING PRIORITY bit is not set (set to a zero), each DMA channel has a fixed priority. In the fixed priority mode, Channel 0 has the highest priority and Channel 3 has the lowest priority. If the ROTATING PRIORITY bit is set to a one, the priority of each channel changes after each DMA cycle (not each DMA request). Each channel moves up to the next highest priority assignment, while the channel which has just been serviced moves to the lowest priority assignment:

	CHANNEL→ JUST SERVICED	CH-0	CH-1	CH-2	CH-3
Priority → Assignments	Highest	CH-1	CH-2	CH-3	CH-0
	↑	CH-2	CH-3	CH-0	CH-1
	↓	CH-3	CH-0	CH-1	CH-2
	Lowest	CH-0	CH-1	CH-2	CH-3

Note that rotating priority will prevent any one channel from monopolizing the DMA mode; consecutive DMA cycles will service different channels if more than one channel is enabled and requesting service. There is no overhead penalty associated with this mode of operation. All DMA operations began with Channel 0 initially assigned to the highest priority for the first DMA cycle.

Extended Write Bit 5

If the EXTENDED WRITE bit is set, the duration of both the $\overline{MEMW}$ and $\overline{I/OW}$ signals is extended by activating them earlier in the DMA cycle. Data transfers within microcomputer systems proceed asynchronously to allow use of various types of memory and I/O devices with different access times. If a device cannot be accessed within a specific amount of time it returns a "not ready" indication to the 8257 that causes the 8257 to insert one or more wait states in its internal sequencing. Some devices are fast enough to be accessed without the use of wait states, but if they generate their READY response with the leading edge of the $\overline{I/OW}$ or $\overline{MEMW}$ signal (which generally occurs late in the transfer sequence), they would normally cause the 8257 to enter a wait state because it does not receive READY in time. For systems with these types of devices, the Extended Write option provides alternative timing for the I/O and memory write signals which allows the devices to return an early READY and prevents the unnecessary occurrence of wait states in the 8257, thus increasing system throughput.

TC Stop Bit 6

If the TC STOP bit is set, a channel is disabled (i.e., its enable bit is reset) after the Terminal Count (TC) output goes true, thus automatically preventing further DMA operation on that channel. The enable bit for that channel must be re-programmed to continue or begin another DMA operation. If the TC STOP bit is not set, the occurrence of the TC output has no effect on the channel enable bits. In this case, it is generally the responsibility of the peripheral to cease DMA requests in order to terminate a DMA operation.

Auto Load Bit 7

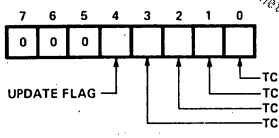
The Auto Load mode permits Channel 2 to be used for repeat block or block chaining operations, without immediate software intervention between blocks. Channel 2 registers are initialized as usual for the first data block; Channel 3 registers, however, are used to store the block re-initialization parameters (DMA starting address, terminal count and DMA transfer mode). After the first block of DMA cycles is executed by Channel 2 (i.e., after the TC output goes true), the parameters stored in the Channel 3 registers are transferred to Channel 2 during an "update" cycle. Note that the TC STOP feature, described above, has no effect on Channel 2 when the Auto Load bit is set.

If the Auto Load bit is set, the initial parameters for Channel 2 are automatically duplicated in the Channel 3 registers when Channel 2 is programmed. This permits repeat block operations to be set up with the programming of a single channel. Repeat block operations can be used in applications such as CRT refreshing. Channels 2 and 3 can still be loaded with separate values if Channel 2 is loaded before loading Channel 3. Note that in the Auto Load mode, Channel 3 is still available to the user if the Channel 3 enable bit is set, but use of this channel will change the values to be auto loaded into Channel 2 at update time. All that is necessary to use the Auto Load feature for chaining operations is to reload Channel 3 registers at the conclusion of each update cycle with the new parameters for the next data block transfer.

Each time that the 8257 enters an update cycle, the update flag in the status register is set and parameters in Channel 3 are transferred to Channel 2, non-destructively for Channel 3. The actual re-initialization of Channel 2 occurs at the beginning of the next channel 2 DMA cycle after the TC cycle. This will be the first DMA cycle of the new data block for Channel 2. The update flag is cleared at the conclusion of this DMA cycle. For chaining operations, the update flag in the status register can be monitored by the CPU to determine when the re-initialization process has been completed so that the next block parameters can be safely loaded into Channel 3.

6. Status Register

The eight-bit status register indicates which channels have reached a terminal count condition and includes the update flag described previously.



The TC status bits are set when the Terminal Count (TC) output is activated for that channel. These bits remain set until the status register is read or the 8257 is reset. The UPDATE FLAG, however, is not affected by a status register read operation. The UPDATE FLAG can be cleared by resetting the 8257, by changing to the non-auto load mode (i.e., by resetting the AUTO LOAD bit in the Mode Set register) or it can be left to clear itself at the completion of the update cycle. The purpose of the UPDATE FLAG is to prevent the CPU from inadvertently skipping a data block by overwriting a starting address or terminal count in the Channel 3 registers before those parameters are properly auto-loaded into Channel 2.

The user is cautioned against reading the TC status register and using this information to reenable channels that have not completed operation. Unless the DMA channels are inhibited a channel could reach terminal count (TC) between the status read and the mode write. DMA can be inhibited by a hardware gate on the HRQ line or by disabling channels with a mode word before reading the TC status.

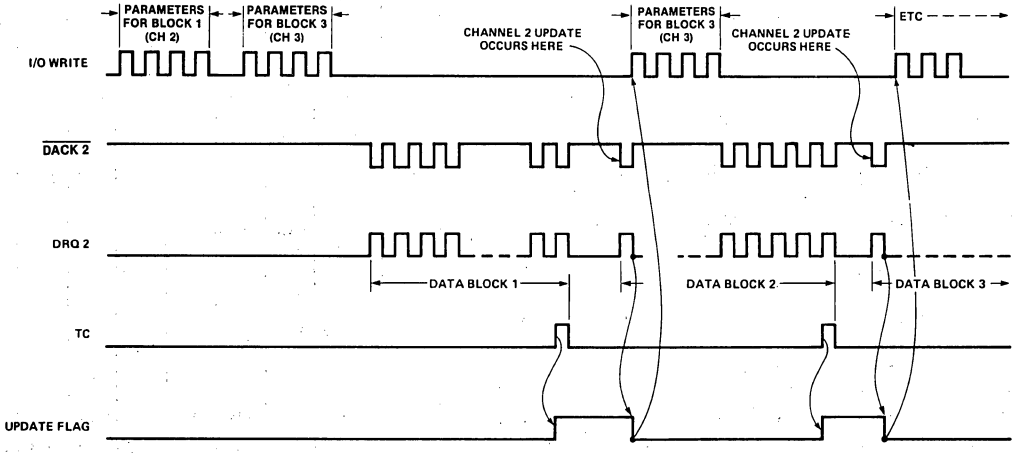


Figure 5. Autoload Timing

OPERATIONAL SUMMARY

Programming and Reading the 8257 Registers

There are four pairs of "channel registers": each pair consisting of a 16-bit DMA address register and a 16-bit terminal count register (one pair for each channel). The 8257 also includes two "general registers": one 8-bit Mode Set register and one 8-bit Status register. The registers are loaded or read when the CPU executes a write or read instruction that addresses the 8257 device and the appropriate register within the 8257. The 8228 generates the appropriate read or write control signal (generally I/OR or I/OW while the CPU places a 16-bit address on the system address bus, and either outputs the data to be written onto the system data bus or accepts the data being read from the data bus. All or some of the most significant 12 address bits A_4-A_{15} (depending on the systems memory, I/O configuration) are usually decoded to produce the chip select ($\overline{CS}$) input to the 8257. An I/O Write input (or Memory Write in memory mapped I/O configurations, described below) specifies that the addressed register is to be programmed, while an I/O Read input (or Memory Read) specifies that the addressed register is to be read. Address bit 3 specifies whether a "channel register" (A_3 = 0) or the Mode Set (program only)/Status (read only) register (A_3 = 1) is to be accessed.

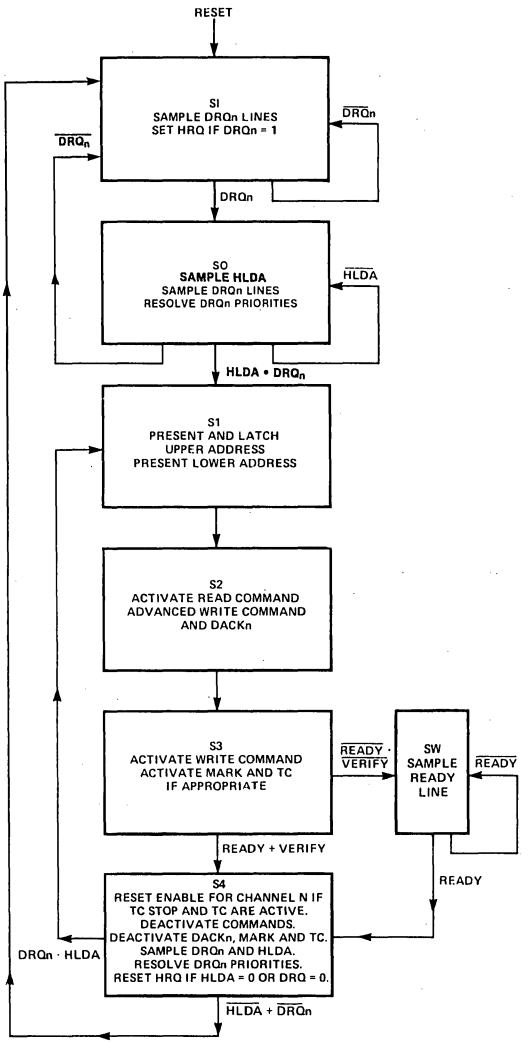
The least significant three address bits, A_0-A_2, indicate the specific register to be accessed. When accessing the Mode Set or Status register, A_0-A_2 are all zero. When accessing a channel register bit A_0 differentiates between the DMA address register (A_0 = 0) and the terminal count register (A_0 = 1), while bits A_1 and A_2 specify one of the four channels. Because the "channel registers" are 16-bits, two program instruction cycles are required to load or read an entire register. The 8257 contains a first/last (F/L) flip flop which toggles at the completion of each channel program or read operation. The F/L flip flop determines whether the upper or lower byte of the register is to be accessed. The F/L flip flop is reset by the RESET input and whenever the Mode Set register is loaded. To maintain proper synchronization when accessing the "channel registers" all channel command instruction operations should occur in pairs, with the lower byte of a register always being accessed first. Do not allow $\overline{CS}$ to clock while either $\overline{I/OR}$ or $\overline{I/OW}$ is active, as this will cause an erroneous F/L flip flop state. In systems utilizing an interrupt structure, interrupts should be disabled prior to any paired programming operations to prevent an interrupt from splitting them. The result of such a split would leave the F/L F/F in the wrong state. This problem is particularly obvious when other DMA channels are programmed by an interrupt structure.

CONTROL INPUT	$\overline{CS}$	$\overline{I/OW}$	$\overline{I/OR}$	A_3
Program Half of a Channel Register	0	0	1	0
Read Half of a Channel Register	0	1	0	0
Program Mode Set Register	0	0	1	1
Read Status Register	0	1	0	1

8257 Register Selection

REGISTER	BYTE	ADDRESS INPUTS				F/L	*BI-DIRECTIONAL DATA BUS							
		A_3	A_2	A_1	A_0		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CH-0 DMA Address	LSB	0	0	0	0	0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	MSB	0	0	0	0	1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8
CH-0 Terminal Count	LSB	0	0	0	1	0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
	MSB	0	0	0	1	1	Rd	Wr	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8
CH-1 DMA Address	LSB	0	0	1	0	0	Same as Channel 0							
	MSB	0	0	1	0	1								
CH-1 Terminal Count	LSB	0	0	1	1	0								
	MSB	0	0	1	1	1								
CH-2 DMA Address	LSB	0	1	0	0	0	Same as Channel 0							
	MSB	0	1	0	0	1								
CH-2 Terminal Count	LSB	0	1	0	1	0								
	MSB	0	1	0	1	1								
CH-3 DMA Address	LSB	0	1	1	0	0	Same as Channel 0							
	MSB	0	1	1	0	1								
CH-3 Terminal Count	LSB	0	1	1	1	0								
	MSB	0	1	1	1	1								
MODE SET (Program only)	—	1	0	0	0	0	AL	TCS	EW	RP	EN3	EN2	EN1	EN0
STATUS (Read only)	—	1	0	0	0	0	0	0	0	UP	TC3	TC2	TC1	TC0

*A_0-A_{15}: DMA Starting Address, C_0-C_{13}: Terminal Count value (N-1), Rd and Wr: DMA Verify (00), Write (01) or Read (10) cycle selection, AL: Auto Load, TCS: TC STOP, EW: EXTENDED WRITE, RP: ROTATING PRIORITY, EN3-EN0: CHANNEL ENABLE MASK, UP: UPDATE FLAG, TC3-TC0: TERMINAL COUNT STATUS BITS.



1 DRQn refers to any DRQ line on an enabled DMA channel.

Figure 6. DMA Operation State Diagram

DMA OPERATION

Single Byte Transfers

A single byte transfer is initiated by the I/O device raising the DRQ line of one channel of the 8257. If the channel is enabled, the 8257 will output a HRQ to the CPU. The 8257 now waits until a HLDA is received insuring that the system bus is free for its use. Once HLDA is received the $\overline{DACK}$ line for the requesting channel is activated (LOW). The $\overline{DACK}$ line acts as a chip select for the requesting I/O device. The 8257 then generates the read and write commands and byte transfer occurs between the selected I/O device and memory. After the transfer is complete, the $\overline{DACK}$ line is set HIGH and the HRQ line is set LOW to indicate to the CPU that the bus is now free for use. DRQ must remain HIGH until $\overline{DACK}$ is issued to be recognized and must go LOW before S4 of the transfer sequence to prevent another transfer from occuring. (See timing diagram.)

Consecutive Transfers

If more than one channel requests service simultaneously, the transfer will occur in the same way a burst does. No overhead is incurred by switching from one channel to another. In each S4 the DRQ lines are sampled and the highest priority request is recognized during the next transfer. A burst mode transfer in a lower priority channel will be overridden by a higher priority request. Once the high priority transfer has completed control will return to the lower priority channel if its DRQ is still active. No extra cycles are needed to execute this sequence and the HRQ line remains active until all DRQ lines go LOW.

Control Override

The continuous DMA transfer mode described above can be interrupted by an external device by lowering the HLDA line. After each DMA transfer the 8257 samples the HLDA line to insure that it is still active. If it is not active, the 8257 completes the current transfer, releases the HRQ line (LOW) and returns to the idle state. If DRQ lines are still active the 8257 will raise the HRQ line in the third cycle and proceed normally. (See timing diagram.)

Not Ready

The 8257 has a Ready input similar to the 8080A and the 8085A. The Ready line is sampled in State 3. If Ready is LOW the 8257 enters a wait state. Ready is sampled during every wait state. When Ready returns HIGH the 8257 proceeds to State 4 to complete the transfer. Ready is used to interface memory or I/O devices that cannot meet the bus set up times required by the 8257.

Speed

The 8257 uses four clock cycles to transfer a byte of data. No cycles are lost in the master to master transfer maximizing bus efficiency. A 2MHz clock input will allow the 8257 to transfer at a rate of 500K bytes/second.

Memory Mapped I/O Configurations

The 8257 can be connected to the system bus as a memory device instead of as an I/O device for memory mapped I/O configurations by connecting the system memory control lines to the 8257's I/O control lines and the system I/O control lines to the 8257's memory control lines.

This configuration permits use of the 8080's considerably larger repertoire of memory instructions when reading or loading the 8257's registers. Note that with this connection, the programming of the Read (bit 15) and Write (bit 14) bits in the terminal count register will have a different meaning:

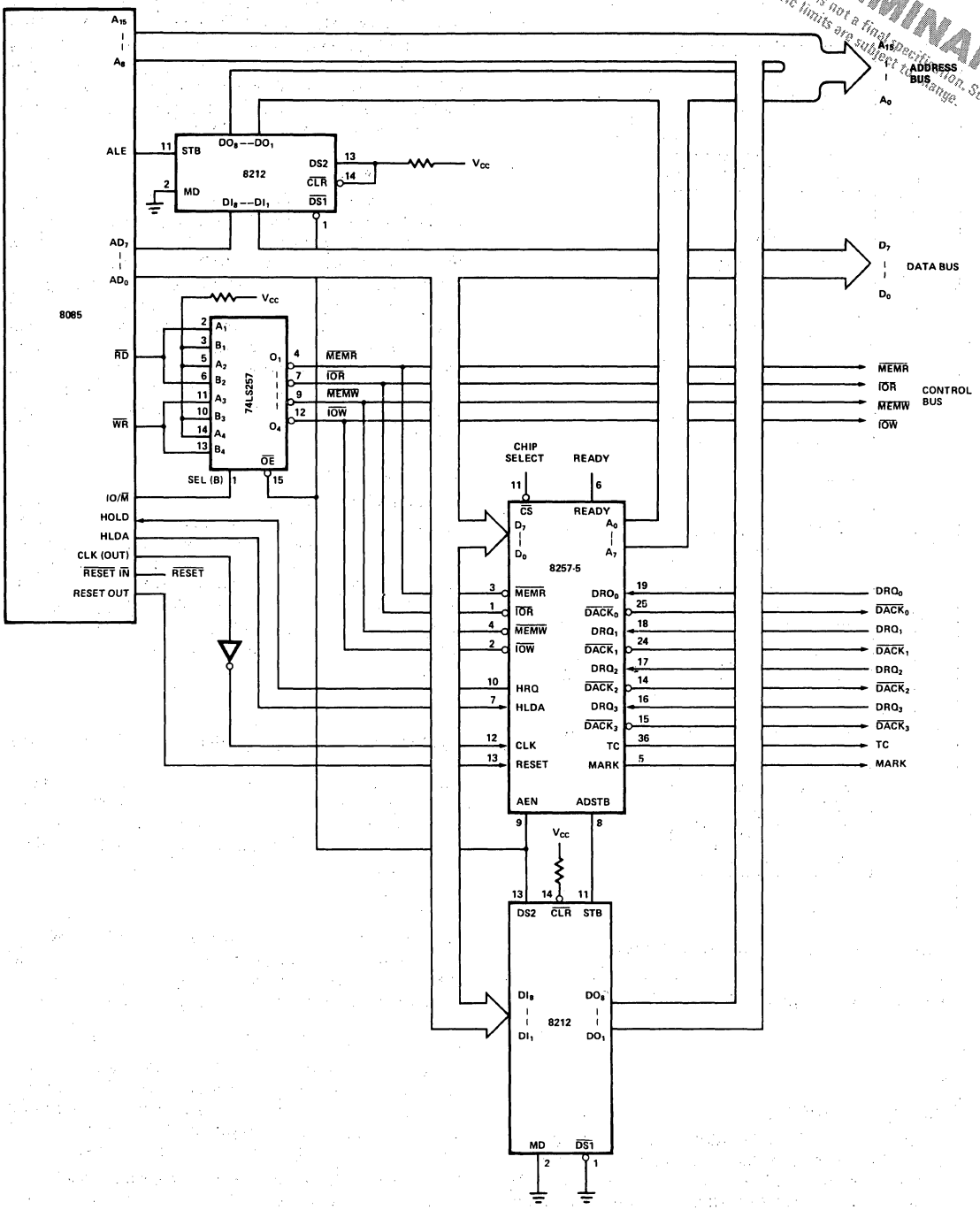


Figure 11. Detailed System Interface Schematic

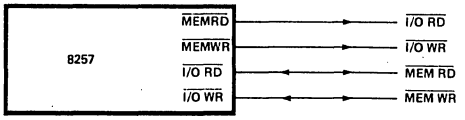


Figure 7. System Interface for Memory Mapped I/O

BIT 15 READ	BIT 14 WRITE	
0	0	DMA Verify Cycle
0	1	DMA Read Cycle
1	0	DMA Write Cycle
1	1	Illegal

Figure 8. TC Register for Memory Mapped I/O Only

SYSTEM APPLICATION EXAMPLES

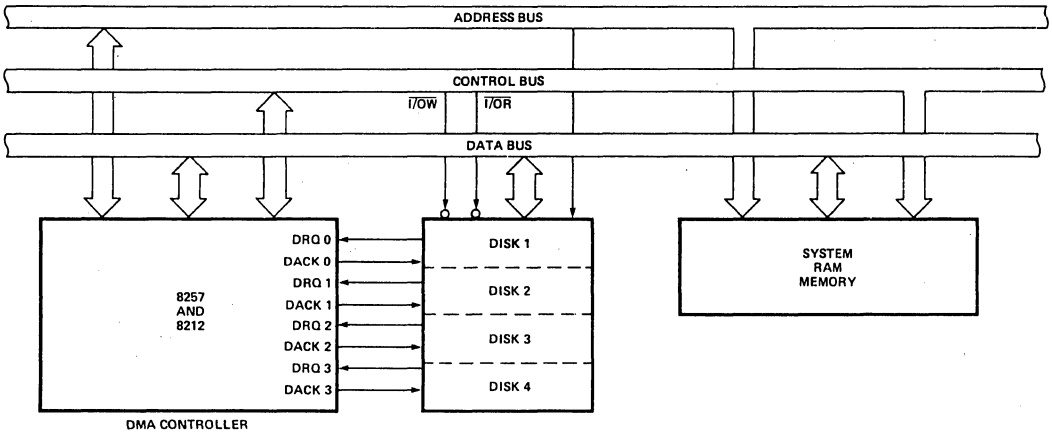


Figure 9. Floppy Disk Controller (4 Drives)

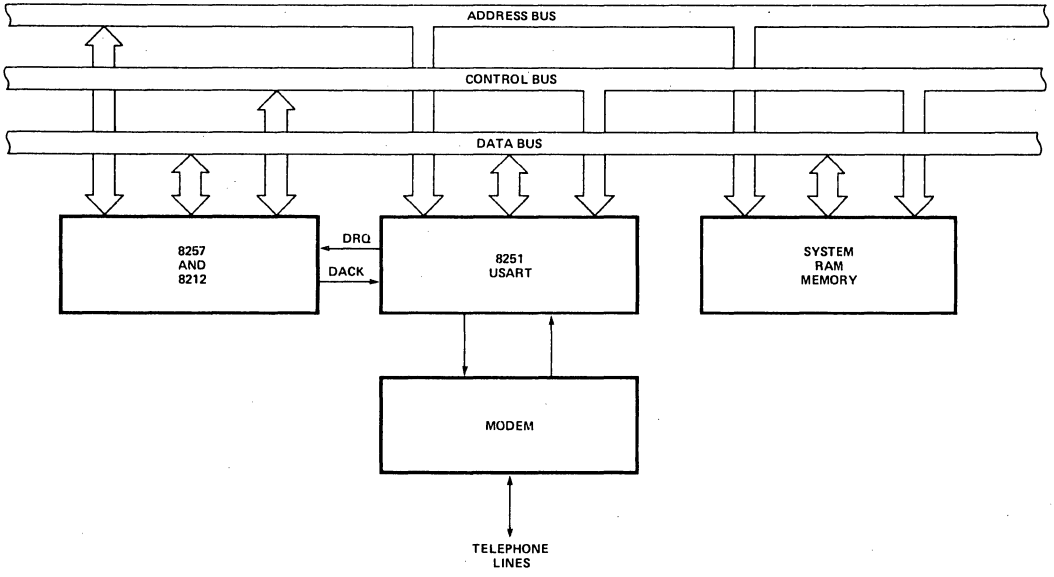


Figure 10. High-Speed Communication Controller

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . .–65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . . –0.5V to +7V
Power Dissipation . . . . . 1 Watt

**COMMENT:* Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 5%, GND = 0V

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	Volts	
V_{IH}	Input High Voltage	2.0	V_{CC}+.5	Volts	
V_{OL}	Output Low Voltage		0.45	Volts	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage	2.4	V_{CC}	Volts	I_{OH}=-150μA for AB, DB and AEN I_{OH}=-80μA for others
V_{HH}	HRQ Output High Voltage	3.3	V_{CC}	Volts	I_{OH} = -80μA
I_{CC}	V_{CC} Current Drain		120	mA	
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Leakage During Float		±10	μA	V_{OUT} = V_{CC} to 0V

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{IN}	Input Capacitance			10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND

A.C. CHARACTERISTICS: PERIPHERAL (SLAVE) MODE

T_A = 0°C to 70°C, V_{CC} = 5.0V ±5%; GND = 0V (Note 1).

8080 Bus Parameters

Read Cycle:

Symbol	Parameter	8257 Min.	8257 Max.	8257-5 Min.	8257-5 Max.	Unit	Test Conditions
T_{AR}	Adr or $\overline{CS}$↓ Setup to $\overline{RD}$↓	0		0		ns	
T_{RA}	Adr or $\overline{CS}$↑ Hold from $\overline{RD}$↑	0		0		ns	
T_{RD}	Data Access from $\overline{RD}$↓	0	300	0	200	ns	(Note 2)
T_{DF}	DB→Float Delay from $\overline{RD}$↑	20	150	20	100	ns	
T_{RR}	$\overline{RD}$ Width	250		250		ns	

Write Cycle:

Symbol	Parameter	8257 Min.	8257 Max.	8257-5 Min.	8257-5 Max.	Unit	Test Conditions
T_{AW}	Adr Setup to $\overline{WR}$↓	20		20		ns	
T_{WA}	Adr Hold from $\overline{WR}$↑	0		0		ns	
T_{DW}	Data Setup to $\overline{WR}$↑	200		200		ns	
T_{WD}	Data Hold from $\overline{WR}$↑	0		0		ns	
T_{WW}	$\overline{WR}$ Width	200		200		ns	

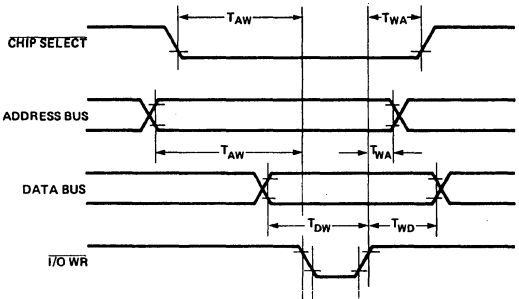
Other Timing:

Symbol	Parameter	8257 Min.	8257 Max.	8257-5 Min.	8257-5 Max.	Unit	Test Conditions
T_{RSTW}	Reset Pulse Width	300		300		ns	
T_{RSTD}	Power Supply↑ (V_{CC}) Setup to Reset↓	500		500		μs	
T_r	Signal Rise Time		20		20	ns	
T_f	Signal Fall Time		20		20	ns	
T_{RSTS}	Reset to First $\overline{I/OWR}$	2		2		t_{CY}	

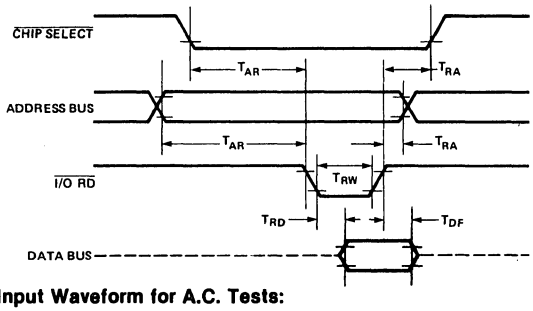
Notes: 1. All timing measurements are made at the following reference voltages unless specified otherwise: Input "1" at 2.0V, "0" at 0.8V
Output "1" at 2.0V, "0" at 0.8V
2. 8257: C_L = 100pF, 8257-5: C_L = 150pF.

8257 PERIPHERAL MODE TIMING DIAGRAMS

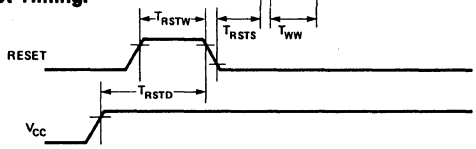
Write Timing:



Read Timing:



Reset Timing:



Input Waveform for A.C. Tests:

A.C. CHARACTERISTICS: DMA (MASTER) MODE $T_A = 0°C$ to $70°C$, $V_{CC} = +5V \pm 5\%$, GND = 0V.

Timing Requirements

SYMBOL	PARAMETER	8257 MIN.	8257 MAX.	8257-5 MIN.	8257-5 MAX.	UNIT
T_{CY}	Cycle Time (Period)	0.320	4	0.320	4	μs
T_θ	Clock Active (High)	120	$.8T_{CY}$	80	$.8T_{CY}$	ns
T_{QS}	DRQ↑ Setup to θ↓ (SI, S4)	120		30		ns
T_{QH}	DRQ↓ Hold from HLDA↑[4]	0		0		ns
T_{HS}	HLDA↑ or ↓Setup to θ↓(SI, S4)	100		100		ns
T_{RS}	READY Setup Time to θ↑ (S3, Sw)	30		30		ns
T_{RH}	READY Hold Time from θ↑ (S3, Sw)	20		20		ns

Note: 4. Tracking Parameter.

Tracking Parameters

Signals labeled as Tracking Parameters (footnotes 4-7 under A.C. Specifications) are signals that follow similar paths through the silicon die. The propagation speed of these signals varies in the manufacturing process but the relationship between all these parameters is constant. The variation is less than or equal to 50 ns.

Suppose the following timing equation is being evaluated,

$$T_{A(MIN)} + T_{B(MAX)} \leq 150 \text{ ns}$$

and only minimum specifications exist for T_A and T_B. If $T_{A(MIN)}$ is used, and if T_A and T_B are tracking parameters, $T_{B(MAX)}$ can be taken as $T_{B(MIN)}$ + 50 ns.

$$T_{A(MIN)} + (T_{B(MIN)}{}^{*} + 50 \text{ ns}) \leq 150 \text{ ns}$$

*if T_A and T_B are tracking parameters

A.C. CHARACTERISTICS: DMA (MASTER) MODE $T_A = 0°C$ to $70°C$, $V_{CC} = +5V \pm 5\%$, GND = 0V

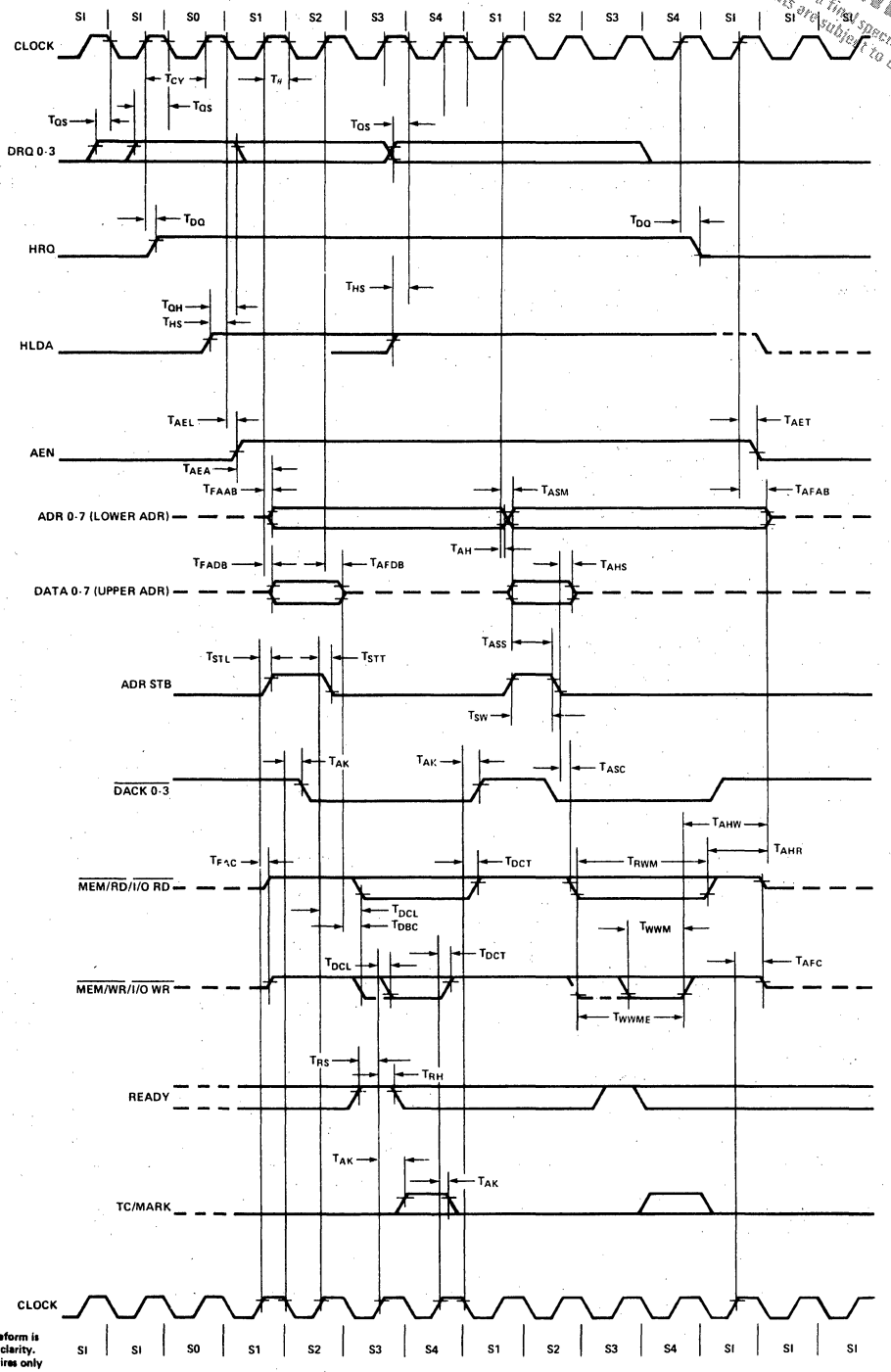
Timing Responses

SYMBOL	PARAMETER	8257 MIN.	8257 MAX.	8257-5 MIN.	8257-5 MAX.	UNIT
T_{DQ}	HRQ↑ or ↓Delay from θ↑(SI,S4) (measured at 2.0V)[1]		160		160	ns
T_{DQ1}	HRQ↑ or ↓Delay from θ↑(SI,S4) (measured at 3.3V)[3]		250		250	ns
T_{AEL}	AEN↑ Delay from θ↓(S1)[1]		300		300	ns
T_{AET}	AEN↓ Delay from θ↑(SI)[1]		200		200	ns
T_{AEA}	Adr(AB)(Active) Delay from AEN↑(S1)[4]	20		20		ns
T_{FAAB}	Adr(AB)(Active) Delay from θ↑(S1)[2]		250		250	ns
T_{AFAB}	Adr(AB)(Float) Delay from θ↑(SI)[2]		150		150	ns
T_{ASM}	Adr(AB)(Stable) Delay from θ↑(S1)[2]		250		250	ns
T_{AH}	Adr(AB)(Stable) Hold from θ↑(S1)[2]	T_{ASM}-50		T_{ASM}-50		ns
T_{AHR}	Adr(AB)(Valid) Hold from $\overline{Rd}$↑(S1,SI)[4]	60		60		ns
T_{AHW}	Adr(AB)(Valid) Hold from $\overline{Wr}$↑(S1,SI)[4]	300		300		ns
T_{FADB}	Adr(DB)(Active) Delay from θ↑(S1)[2]		300		300	ns
T_{AFDB}	Adr(DB)(Float) Delay from θ↑(S2)[2]	T_{STT}+20	250	T_{STT}+20	170	ns
T_{ASS}	Adr(DB) Setup to AdrStb↓(S1-S2)[4]	100		100		ns
T_{AHS}	Adr(DB)(Valid) Hold from AdrStb↓(S2)[4]	50		50		ns
T_{STL}	AdrStb↑ Delay from θ↑(S1)[1]		200		200	ns
T_{STT}	AdrStb↓ Delay from θ↑(S2)[1]		140		140	ns
T_{SW}	AdrStb Width (S1-S2)[4]	T_{CY}-100		T_{CY}-100		ns
T_{ASC}	$\overline{Rd}$↓ or $\overline{Wr}$(Ext)↓ Delay from AdrStb↓(S2)[4]	70		70		ns
T_{DBC}	$\overline{Rd}$↓ or $\overline{Wr}$(Ext)↓ Delay from Adr(DB) (Float)(S2)[4]	20		20		ns
T_{AK}	DACK↑ or ↓Delay from θ↓(S2,S1) and TC/Mark↑ Delay from θ↑(S3) and TC/Mark↓ Delay from θ↑ (S4)[1,5]		250		250	ns
T_{DCL}	$\overline{Rd}$↓ or $\overline{Wr}$(Ext)↓ Delay from θ↑(S2) and $\overline{Wr}$↓ Delay from θ↑(S3)[2,6]		200		200	ns
T_{DCT}	$\overline{Rd}$↑ Delay from θ↓(S1,SI) and $\overline{Wr}$↑ Delay from θ↑(S4)[2,7]		200		200	ns
T_{FAC}	$\overline{Rd}$ or $\overline{Wr}$ (Active) from θ↑(S1)[2]		300		300	ns
T_{AFC}	$\overline{Rd}$ or $\overline{Wr}$ (Float) from θ↑(SI)[2]		150		150	ns
T_{RWM}	$\overline{Rd}$ Width (S2-S1 or SI)[4]	$2T_{CY} + T_\theta$-50		$2T_{CY} + T_\theta$-50		ns
T_{WWM}	$\overline{Wr}$ Width (S3-S4)[4]	T_{CY}-50		T_{CY}-50		ns
T_{WWME}	$\overline{Wr}$(Ext) Width (S2-S4)[4]	$2T_{CY}$-50		$2T_{CY}$-50		ns

Notes: 1. Load = 1 TTL. 2. Load = 1 TTL + 50pF. 3. Load = 1 TTL + (R_L = 3.3K), V_{OH} = 3.3V. 4. Tracking Parameter. 5. $\Delta T_{AK} < 50$ ns. 6. $\Delta T_{DCL} < 50$ ns. 7. $\Delta T_{DCT} < 50$ ns.

DMA MODE WAVEFORMS

CONSECUTIVE CYCLES AND BURST MODE SEQUENCE



NOTE: The clock waveform is duplicated for clarity. The 8257 requires only one clock input.

Figure 12. Consecutive Cycles and Burst Mode Sequence

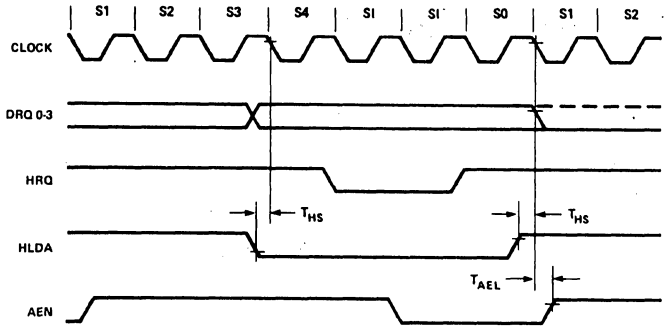


Figure 13. Control Override Sequence

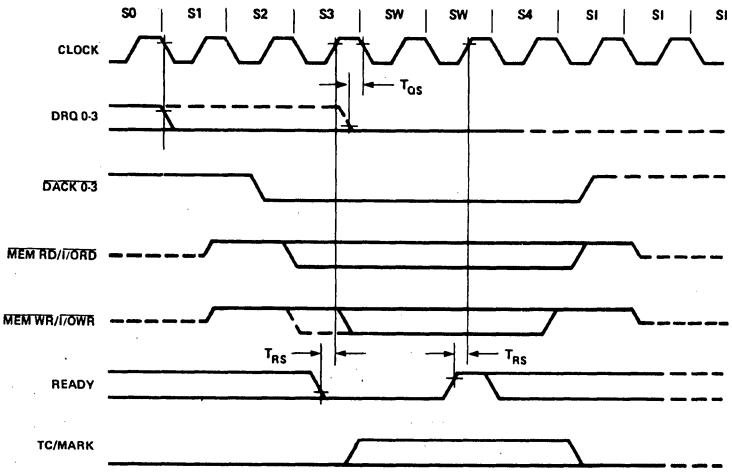


Figure 14. Not Ready Sequence

intel®

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8259A
PROGRAMMABLE INTERRUPT CONTROLLER

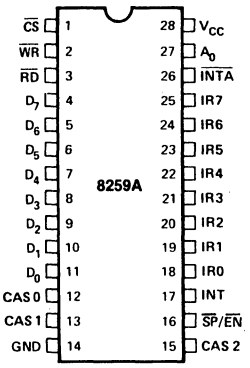
- **MCS-86™ Compatible**
- **MCS-80/85™ Compatible**
- **Eight-Level Priority Controller**
- **Expandable to 64 Levels**
- **Programmable Interrupt Modes**
- **Individual Request Mask Capability**
- **Single +5V Supply (No Clocks)**
- **28-Pin Dual-In-Line Package**

The Intel® 8259A Programmable Interrupt Controller handles up to eight vectored priority interrupts for the CPU. It is cascadable for up to 64 vectored priority interrupts without additional circuitry. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single +5V supply. Circuitry is static, requiring no clock input.

The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements.

The 8259A is fully upward compatible with the Intel® 8259. Software originally written for the 8259 will operate the 8259A in all 8259 equivalent modes (MCS-80/85, Non-Buffered, Edge Triggered).

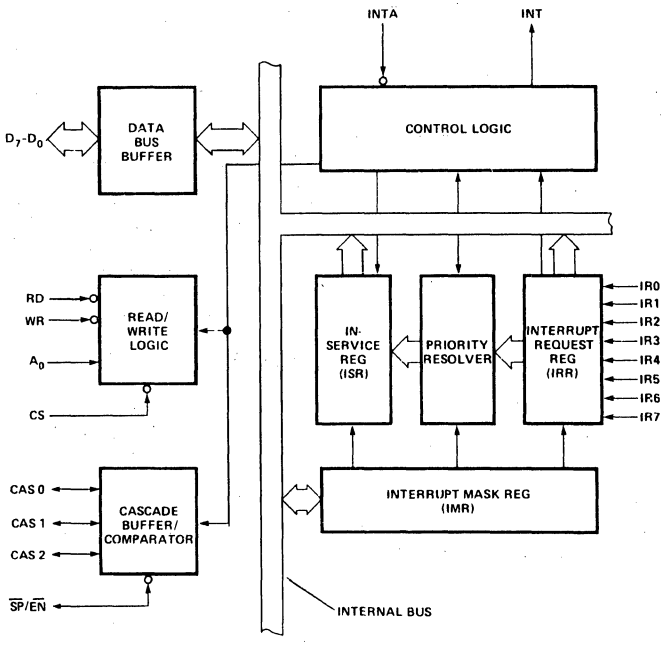
PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A_0	COMMAND SELECT ADDRESS
$\overline{CS}$	CHIP SELECT
CAS2-CAS0	CASCADE LINES
$\overline{SP}/\overline{EN}$	SLAVE PROGRAM INPUT/ENABLE
INT	INTERRUPT OUTPUT
$\overline{INTA}$	INTERRUPT ACKNOWLEDGE INPUT
IR0-IR7	INTERRUPT REQUEST INPUTS

BLOCK DIAGRAM



INTERRUPTS IN MICROCOMPUTER SYSTEMS

Microcomputer system design requires that I/O devices such as keyboards, displays, sensors and other components receive servicing in an efficient manner so that large amounts of the total system tasks can be assumed by the microcomputer with little or no effect on throughput.

The most common method of servicing such devices is the *Polled* approach. This is where the processor must test each device in sequence and in effect "ask" each one if it needs servicing. It is easy to see that a large portion of the main program is looping through this continuous polling cycle and that such a method would have a serious, detrimental effect on system throughput, thus limiting the tasks that could be assumed by the microcomputer and reducing the cost effectiveness of using such devices.

A more desirable method would be one that would allow the microprocessor to be executing its main program and only stop to service peripheral devices when it is told to do so by the device itself. In effect, the method would provide an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine that will service the requesting device. Once this servicing is complete, however, the processor would resume exactly where it left off.

This method is called *Interrupt*. It is easy to see that system throughput would drastically increase, and thus more tasks could be assumed by the microcomputer to further enhance its cost effectiveness.

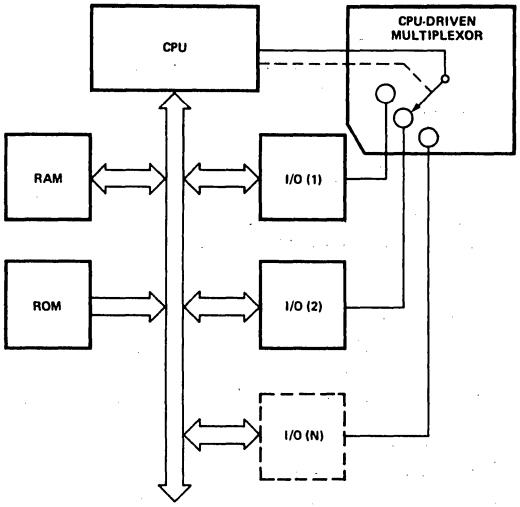
The Programmable Interrupt Controller (PIC) functions as an overall manager in an Interrupt-Driven system environment. It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest importance (priority), ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.

Each peripheral device or structure usually has a special program or "routine" that is associated with its specific functional or operational requirements; this is referred to as a "service routine". The PIC, after issuing an Interrupt to the CPU, must somehow input information into the CPU that can "point" the Program Counter to the service routine associated with the requesting device. This "pointer" is an address in a vectoring table and will often be referred to, in this document, as vectoring data.

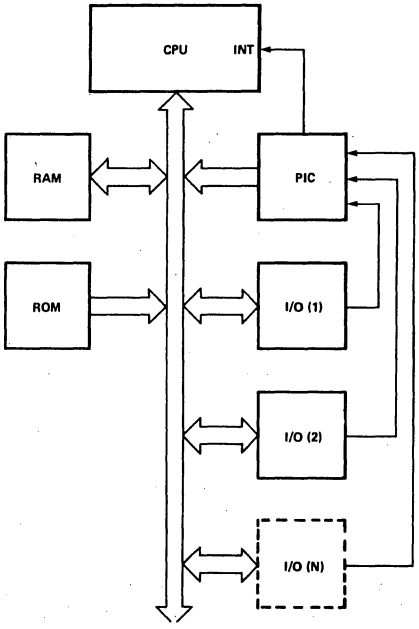
8259A BASIC FUNCTIONAL DESCRIPTION

GENERAL

The 8259A is a device specifically designed for use in real time, interrupt driven microcomputer systems. It manages eight levels or requests and has built-in features for expandability to other 8259A's (up to 64 levels). It is programmed by the system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 8259A can be configured to match his system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt structure can be defined as required, based on the total system environment.



Polled Method



Interrupt Method

INTERRUPT REQUEST REGISTER (IRR) AND IN-SERVICE REGISTER (ISR)

The interrupts at the IR input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all the interrupt levels which are requesting service; and the ISR is used to store all the interrupt levels which are being serviced.

PRIORITY RESOLVER

This logic block determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during $\overline{INTA}$ pulse.

INTERRUPT MASK REGISTER (IMR)

The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

INT (INTERRUPT)

This output goes directly to the CPU interrupt input. The V_{OH} level on this line is designed to be fully compatible with the 8080A, 8085A and 8086 input levels.

$\overline{INTA}$ (INTERRUPT ACKNOWLEDGE)

$\overline{INTA}$ pulses will cause the 8259A to release vectoring information onto the data bus. The format of this data depends on the system mode (μPM) of the 8259A.

DATA BUS BUFFER

This 3-state, bidirectional 8-bit buffer is used to interface the 8259A to the system Data Bus. Control words and status information are transferred through the Data Bus Buffer.

READ/WRITE CONTROL LOGIC

The function of this block is to accept OUTput commands from the CPU. It contains the Initialization Command Word (ICW) registers and Operation Command Word (OCW) registers which store the various control formats for device operation. This function block also allows the status of the 8259A to be transferred onto the Data Bus.

$\overline{CS}$ (CHIP SELECT)

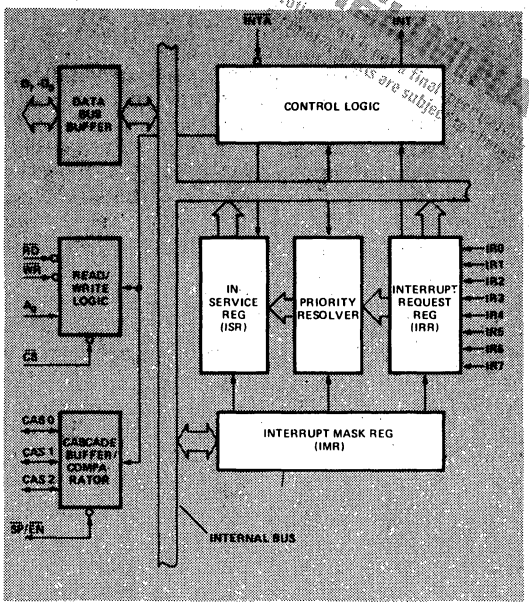
A LOW on this input enables the 8259A. No reading or writing of the chip will occur unless the device is selected.

$\overline{WR}$ (WRITE)

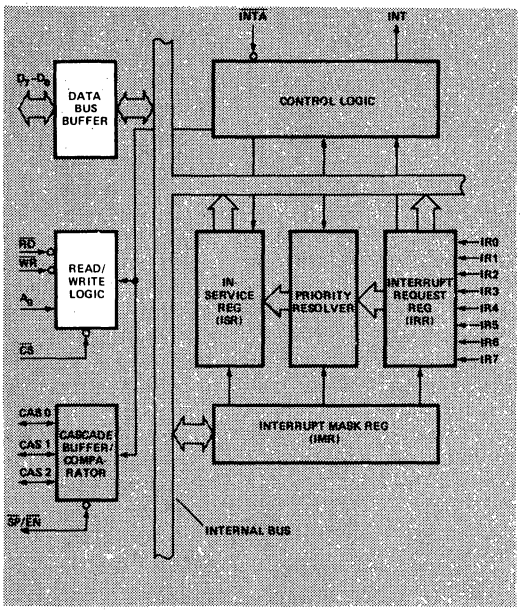
A LOW on this input enables the CPU to write control words (ICWs and OCWs) to the 8259A.

$\overline{RD}$ (READ)

A LOW on this input enables the 8259A to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR), or the Interrupt level onto the Data Bus.



8259A Block Diagram



8259A Block Diagram

A_0

This input signal is used in conjunction with $\overline{WR}$ and $\overline{RD}$ signals to write commands into the various command registers, as well as reading the various status registers of the chip. This line can be tied directly to one of the address lines.

THE CASCADE BUFFER/COMPARATOR

This function block stores and compares the IDs of all 8259A's used in the system. The associated three I/O pins (CAS0-2) are outputs when the 8259A is used as a master and are inputs when the 8259A is used as a slave. As a master, the 8259A sends the ID of the interrupting slave device onto the CAS0-2 lines. The slave thus selected will send its preprogrammed subroutine address onto the Data Bus during the next one or two consecutive $\overline{INTA}$ pulses. (See section "Cascading the 8259A".)

INTERRUPT SEQUENCE

The powerful features of the 8259A in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The normal sequence of events during an interrupt depends on the type of CPU being used.

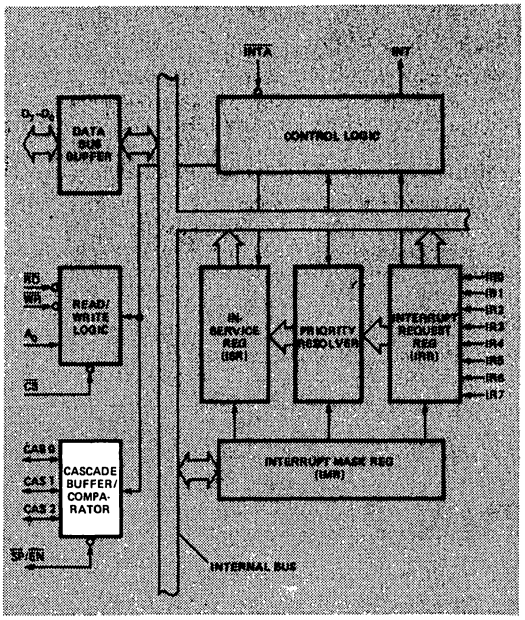
The events occur as follows in an MCS-80/85 system:

1. One or more of the INTERRUPT REQUEST lines (IR7-0) are raised high, setting the corresponding IRR bit(s).
2. The 8259A evaluates these requests, and sends an INT to the CPU, if appropriate.
3. The CPU acknowledges the INT and responds with an $\overline{INTA}$ pulse.
4. Upon receiving an $\overline{INTA}$ from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 8259A will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7-0 pins.
5. This CALL instruction will initiate two more $\overline{INTA}$ pulses to be sent to the 8259A from the CPU group.
6. These two $\overline{INTA}$ pulses allow the 8259A to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first $\overline{INTA}$ pulse and and the higher 8-bit address is released at the second $\overline{INTA}$ pulse.
7. This completes the 3-byte CALL instruction released by the 8259A. In the AEOI mode the ISR bit is reset at the end of the third $\overline{INTA}$ pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt sequence.

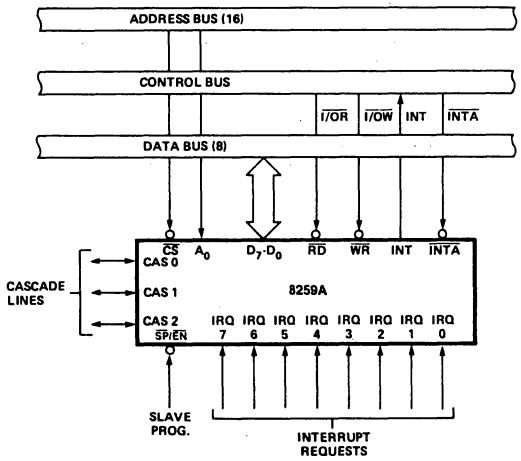
The events occurring in an MCS-86 system are the same until step 4.

4. Upon receiving an $\overline{INTA}$ from the CPU group, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive the Data Bus during this cycle.
5. The MCS-86 CPU will initiate a second $\overline{INTA}$ pulse. During this pulse, the 8259A releases an 8-bit pointer onto the Data Bus where it is read by the CPU.
6. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second $\overline{INTA}$ pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt request is present at step 4 of either sequence (i.e., the request was too short in duration) the 8259A will issue an interrupt level 7. Both the vectoring bytes and the CAS lines will look like an interrupt level 7 was requested.



8259A Block Diagram



8259A Interface to Standard System Bus

INTERRUPT SEQUENCE OUTPUTS

MCS-80/85 SYSTEM

This sequence is timed by three $\overline{INTA}$ pulses. During the first $\overline{INTA}$ pulse the CALL opcode is enabled onto the data bus.

Content of First Interrupt Vector Byte

	D7	D6	D5	D4	D3	D2	D1	D0
CALL CODE	1	1	0	0	1	1	0	1

During the second $\overline{INTA}$ pulse the lower address of the appropriate service routine is enabled onto the data bus. When Interval = 4 bits A_5–A_7 are programmed, while A_0–A_4 are automatically inserted by the 8259A. When Interval = 8 only A_6 and A_7 are programmed, while A_0–A_5 are automatically inserted.

Content of Second Interrupt Vector Byte

IR	Interval = 4							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
5	A7	A6	A5	1	0	1	0	0
4	A7	A6	A5	1	0	0	0	0
3	A7	A6	A5	0	1	1	0	0
2	A7	A6	A5	0	1	0	0	0
1	A7	A6	A5	0	0	1	0	0
0	A7	A6	A5	0	0	0	0	0

IR	Interval = 8							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	1	1	1	0	0	0
6	A7	A6	1	1	0	0	0	0
5	A7	A6	1	0	1	0	0	0
4	A7	A6	1	0	0	0	0	0
3	A7	A6	0	1	1	0	0	0
2	A7	A6	0	1	0	0	0	0
1	A7	A6	0	0	1	0	0	0
0	A7	A6	0	0	0	0	0	0

During the third INTA pulse the higher address of the appropriate service routine, which was programmed as byte 2 of the initialization sequence (A_8 – A_{15}), is enabled onto the bus.

Content of Third Interrupt Vector Byte

D7	D6	D5	D4	D3	D2	D1	D0
A15	A14	A13	A12	A11	A10	A9	A8

MCS-86 SYSTEM

MCS-86 mode is similar to MCS-80 mode except that only two Interrupt Acknowledge cycles are issued by the processor and no CALL opcode is sent to the processor. The first interrupt acknowledge cycle is similar to that of MCS-80/85 systems in that the 8259A uses it to internally freeze the state of the interrupts for priority resolution and as a master it issues the interrupt code on the cascade lines at the end of the $\overline{INTA}$ pulse. On this first cycle it does not issue any data to the processor and leaves its data bus buffers disabled. On the second interrupt acknowledge cycle in MCS-86 mode the master (or slave if so programmed) will send a byte of data to the processor with the acknowledged interrupt code composed as follows (note the state of the ADI mode control is ignored and A_5–A_{11} are unused in MCS-86 mode):

Content of Interrupt Vector Byte for MCS-86 System Mode

	D7	D6	D5	D4	D3	D2	D1	D0
IR7	A15	A14	A13	A12	A11	1	1	1
IR6	A15	A14	A13	A12	A11	1	1	0
IR5	A15	A14	A13	A12	A11	1	0	1
IR4	A15	A14	A13	A12	A11	1	0	0
IR3	A15	A14	A13	A12	A11	0	1	1
IR2	A15	A14	A13	A12	A11	0	1	0
IR1	A15	A14	A13	A12	A11	0	0	1
IR0	A15	A14	A13	A12	A11	0	0	0

PROGRAMMING THE 8259A

The 8259A accepts two types of command words generated by the CPU:

1. *Initialization Command Words (ICWs):* Before normal operation can begin, each 8259A in the system must be brought to a starting point — by a sequence of 2 to 4 bytes timed by $\overline{WR}$ pulses. This sequence is described in Figure 1.

2. *Operation Command Words (OCWs):* These are the command words which command the 8259A to operate in various interrupt modes. These modes are:
   a. Fully nested mode
   b. Rotating priority mode
   c. Special mask mode
   d. Polled mode

The OCWs can be written into the 8259A anytime after initialization.

INITIALIZATION

GENERAL

Whenever a command is issued with A0 = 0 and D4 = 1, this is interpreted as Initialization Command Word 1 (ICW1). ICW1 starts the initialization sequence during which the following automatically occur.

a. The Interrupt Mask Register is cleared.

b. IR 7 input is assigned priority 7.

c. The slave mode address is set to 7.

d. Special Mask Mode is cleared and Status Read is set to IRR.

e. If IC4 = 0, then all functions selected in ICW4 are set to zero. (Non-Buffered mode*, no Auto-EOI, MCS-80/85 system, non SFNM).

***Note:** Master/Slave in ICW4 is only used in the buffered mode.

A_0	D_4	D_3	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0			0	1	0	IRR, ISR or Interrupting Level → DATA BUS (Note 1)
1			0	1	0	IMR → DATA BUS
						OUTPUT OPERATION (WRITE)
0	0	0	1	0	0	DATA BUS → OCW2
0	0	1	1	0	0	DATA BUS → OCW3
0	1	X	1	0	0	DATA BUS → ICW1
1	X	X	1	0	0	DATA BUS → OCW1, ICW2, ICW3, ICW4 (Note 2)
						DISABLE FUNCTION
X	X	X	1	1	0	DATA BUS — 3-STATE (NO OPERATION)
X	X	X	X	X	1	DATA BUS — 3-STATE (NO OPERATION)

Notes: 1. Selection of IRR, ISR or Interrupting Level is based on the content of OCW3 written before the READ operation.
2. On-chip sequencer logic queues these commands into proper sequence.

8259A Basic Operation

INITIALIZATION COMMAND WORDS 1 AND 2 (ICW1, ICW2)

A_5-A_{15}: *Page starting address of service routines.* In an MCS 80/85 system, the 8 request levels will generate CALLs to 8 locations equally spaced in memory. These can be programmed to be spaced at intervals of 4 or 8 memory locations, thus the 8 routines will occupy a page of 32 or 64 bytes, respectively.

The address format is 2 bytes long (A_0-A_{15}). When the routine interval is 4, A_0-A_4 are automatically inserted by the 8259A, while A_5-A_{15} are programmed externally. When the routine interval is 8, A_0-A_5 are automatically inserted by the 8259A, while A_6-A_{15} are programmed externally.

The 8-byte interval will maintain compatibility with current software, while the 4-byte interval is best for a compact jump table.

In an MCS-86 system A_{15}-A_{11} are inserted in the five most significant bits of the vectoring byte and the 8259A sets the three least significant bits according to the interrupt level. A_{10}-A_5 are ignored and ADI (Address interval) has no effect.

LTIM: If LTIM = 1, then the 8259A will operate in the level interrupt mode. Edge detect logic on the interrupt inputs will be disabled.

ADI: CALL address interval. ADI = 1 then interval = 4; ADI = 0 then interval = 8.

SNGL: Single. Means that this is the only 8259A in the system. If SNGL = 1 no ICW3 will be issued.

IC4: If this bit is set — ICW4 has to be read. If ICW4 is not needed, set IC4 = 0.

INITIALIZATION COMMAND WORD 3 (ICW3)

This word is read only when there is more than one 8259A in the system and cascading is used, in which case SNGL = 0. It will load the 8-bit slave register. The functions of this register are:

a. In the master mode (either when SP = 1, or in buffered mode when M/S = 1 in ICW4) a "1" is set for each slave in the system. The master then will release byte 1 of the call sequence (for MCS-80/85 system) and will enable the corresponding slave to release bytes 2 and 3 (for MCS-86 only byte 2) through the cascade lines.

b. In the slave mode (either when $\overline{SP}$ = 0, or if BUF = 1 and M/S = 0 in ICW4) bits 2-0 identify the slave. The slave compares its cascade input with these bits and, if they are equal, bytes 2 and 3 of the call sequence (or just byte 2 for MCS-86) are released by it on the Data Bus.

INITIALIZATION COMMAND WORD 4 (ICW4)

SFNM: If SFNM = 1 the special fully nested mode is programmed.

BUF: If BUF = 1 the buffered mode is programmed. In buffered mode $\overline{SP}/\overline{EN}$ becomes an enable output and the master/slave determination is by M/S.

M/S: If buffered mode is selected: M/S = 1 means the 8259A is programmed to be a master, M/S = 0 means the 8259A is programmed to be a slave. If BUF = 0, M/S has no function.

AEOI: If AEOI = 1 the automatic end of interrupt mode is programmed.

μPM: Microprocessor mode: μPM = 0 sets the 8259A for MCS-80/85 system operation, μPM = 1 sets the 8259A for MCS-86 system operation.

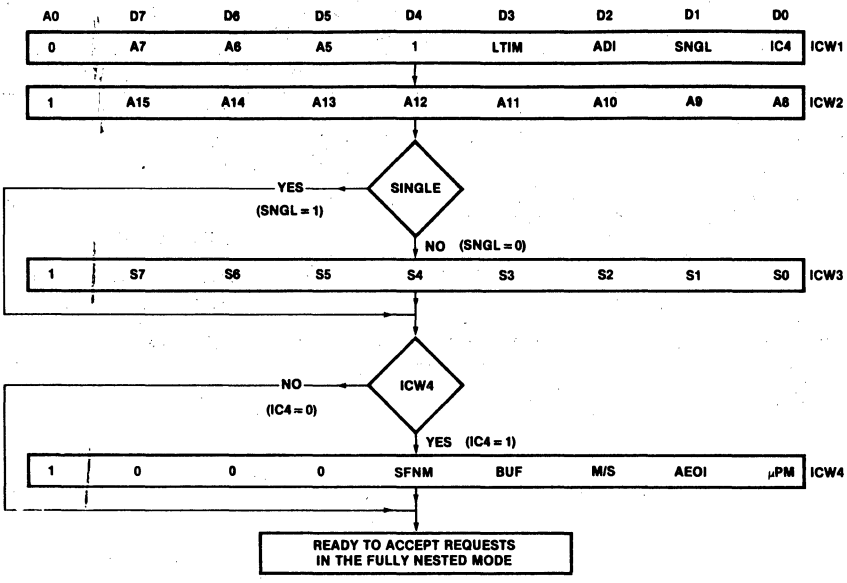
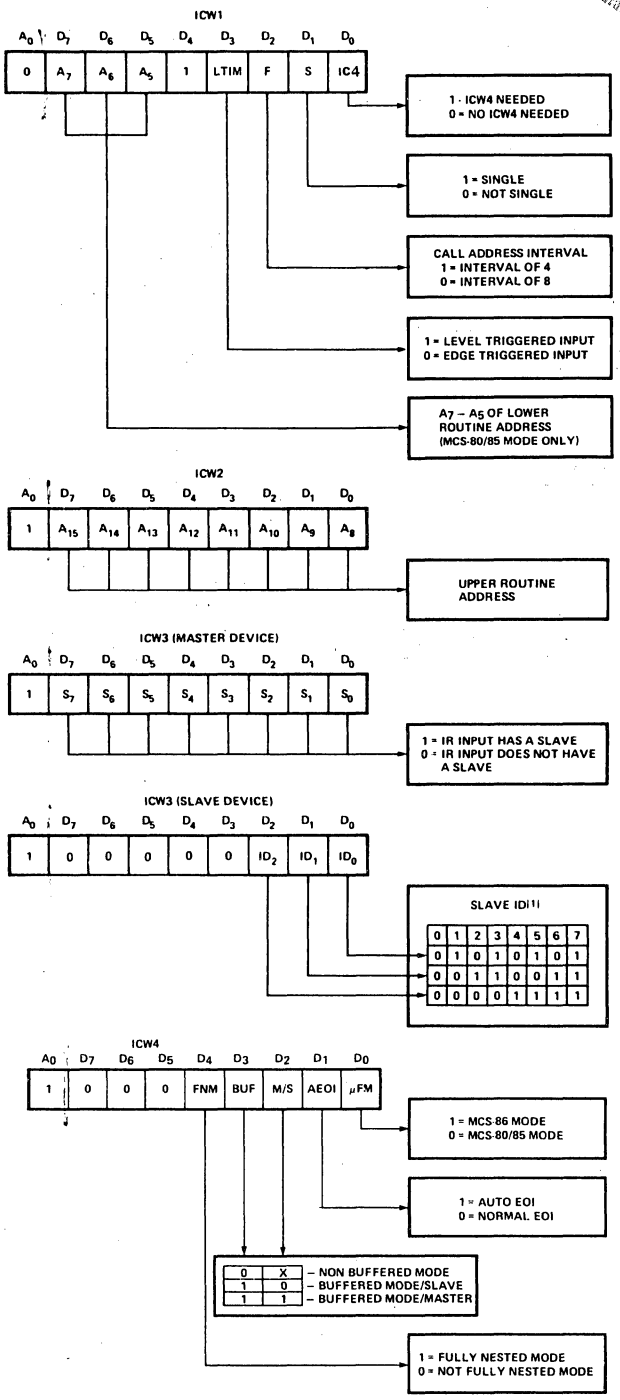


Figure 1. Initialization Sequence



NOTE 1: SLAVED ID IS EQUAL TO THE CORRESPONDING MASTER IR INPUT.
NOTE 2: X INDICATED "DON'T CARE".

Initialization Command Word Format

OPERATION COMMAND WORDS (OCWs)

After the Initialization Command Words (ICWs) are programmed into the 8259A, the chip is ready to accept interrupt requests at its input lines. However, during the 8259A operation, a selection of algorithms can command the 8259A to operate in various modes through the Operation Command Words (OCWs).

OPERATION CONTROL WORDS (OCWs)

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

OCW2

0	R	SEOI	EOI	0	0	L2	L1	L0

OCW3

0	0	SSMM	SMM	0	1	P	SRIS	RIS

OPERATION CONTROL WORD 1 (OCW1)

OCW1 sets and clears the mask bits in the interrupt Mask Register (IMR). $M_7 - M_0$ represent the eight mask bits. M = 1 indicates the channel is masked (inhibited), M = 0 indicates the channel is enabled.

OPERATION CONTROL WORD 2 (OCW2)

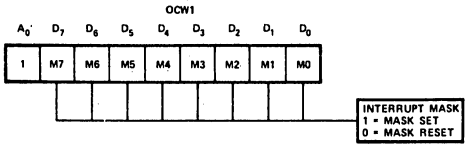
R, SEOI, EOI — These three bits control the Rotate and End of Interrupt modes and combinations of the two. A chart of these combinations can be found on the Operation Command Word Format.

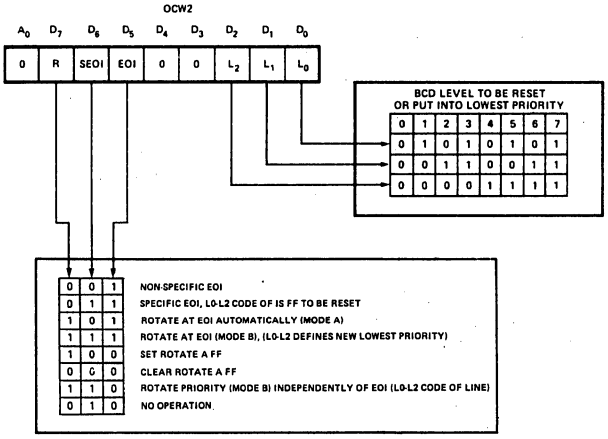
L_2, L_1, L_0 — These bits determine the interrupt level acted upon when the SEOI bit is active.

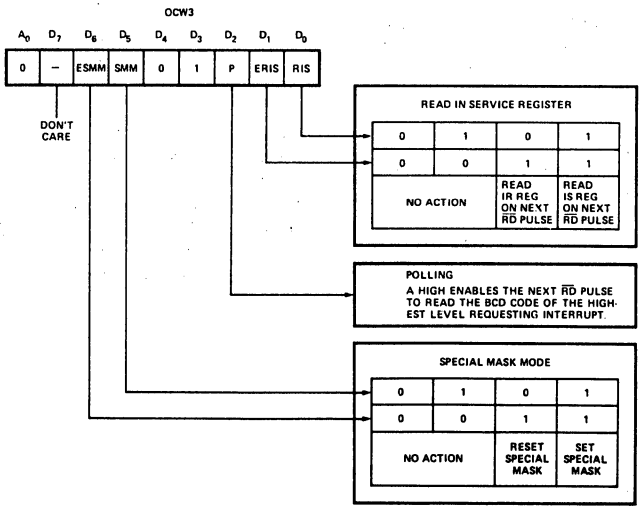
OPERATION CONTROL WORD 3 (OCW3)

ESMM — Enable Special Mask Mode. When this bit is set to 1 it enables the SMM bit to set or reset the Special Mask Mode. When ESMM = 0 the SMM bit becomes a "don't care".

SMM — Special Mask Mode. If ESMM = 1 and SMM = 1 the 8259A will enter Special Mask Mode. If ESMM = 1 and SMM = 0 the 8259A will revert to normal mask mode. When ESMM = 0, SMM has no effect.







Operation Command Word Format

INTERRUPT MASKS

Each Interrupt Request input can be masked individually by the Interrupt Mask Register (IMR) programmed through OCW1. Each bit in the IMR masks one interrupt channel if it is set (1). Bit 0 masks IR0, Bit 1 masks IR1 and so forth. Masking an IR channel does not affect the other channels operation.

SPECIAL MASK MODE

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty here is that if an Interrupt Request is acknowledged and an End of Interrupt command did not reset its IS bit (i.e., while executing a service routine), the 8259A would have inhibited all lower priority requests with no easy way for the routine to enable them

That is where the Special Mask Mode comes in. In the special Mask Mode, when a mask bit is set in OCW1, it inhibits further interrupts at that level *and enables* interrupts from *all other* levels (lower as well as higher) that are not masked.

Thus, any interrupts may be selectively enabled by loading the mask register.

The special Mask Mode is set by OCW3 where: SSMM = 1, SMM = 1, and cleared where SSMM = 1, SMM = 0.

BUFFERED MODE

When the 8259A is used in a large system where bus driving buffers are required on the data bus and the cascading mode is used, there exists the problem of enabling buffers.

The buffered mode will structure the 8259A to send an enable signal on $\overline{SP}/\overline{EN}$ to enable the buffers. In this mode, whenever the 8259A's data bus outputs are enabled, the $\overline{SP}/\overline{EN}$ output becomes active.

This modification forces the use of software programming to determine whether the 8259A is a master or a slave. Bit 3 in ICW4 programs the buffered mode, and bit 2 in ICW4 determines whether it is a master or a slave.

FULLY NESTED MODE

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority form 0 through 7 (0 highest). When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service register (ISO-7) is set. This bit remains set until the microprocessor issues an End of Interrupt (EOI) command immediately before returning from the service routine, or if AEOI (Automatic End of Interrupt) bit is set, until the trailing edge of the last INTA. While the IS bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged only if the microprocessor internal Interrupt enable flip-flop has been re-enabled through software).

After the initialization sequence, IR0 has the highest priority and IR7 the lowest. Priorities can be changed, as will be explained, in the rotating priority mode.

THE SPECIAL FULLY NESTED MODE

This mode will be used in the case of a big system where cascading is used, and the priority has to be conserved within each slave. In this case the fully nested mode will be programmed to the master (using ICW4). This mode is similar to the normal nested mode with the following exceptions:

a. When an interrupt request from a certain slave is in service this slave is not locked out from the master's priority logic and further interrupt requests from higher priority IR's within the slave will be recognized by the master and will initiate interrupts to the processor. (In the normal nested mode a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.)

b. When exiting the Interrupt Service routine the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-Service register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent.

POLL

In this mode the microprocessor internal Interrupt Enable flip-flop is reset, disabling its interrupt input. Service to devices is achieved by programmer initiative using a Poll command.

The Poll command is issued by setting P = "1" in OCW3. The 8259A treats the next $\overline{RD}$ pulse to the 8259A (i.e., $\overline{RD}$ = 0, $\overline{CS}$ = 0) as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupt is frozen from $\overline{WR}$ to $\overline{RD}$.

The word enabled onto the data bus during $\overline{RD}$ is:

D7	D6	D5	D4	D3	D2	D1	D0
I	—	—	—	—	W2	W1	W0

W0-W2: Binary code of the highest priority level requesting service.

I: Equal to a "1" if there is an interrupt.

This mode is useful if there is a routine command common to several levels so that the $\overline{INTA}$ sequence is not needed (saves ROM space). Another application is to use the poll mode to expand the number of priority levels to more than 64.

END OF INTERRUPT (EOI)

The In Service (IS) bit can be reset either automatically following the trailing edge of the last in sequence $\overline{INTA}$ pulse (when AEOI bit in ICW1 is set) or by a command word that must be issued to the 8259A before returning from a service routine (EOI command). An EOI command must be issued twice, once for the master and once for the corresponding slave if slaves are in use.

There are two forms of EOI command: Specific and Non-Specific. When the 8259A is operated in modes which preserve the fully nested structure, it can determine which IS bit to reset on EOI. When a Non-Specific EOI command is issued the 8259A will automatically reset the highest IS bit of those that are set, since in the nested mode the highest IS level was necessarily the last level acknowledged and serviced.

However, when a mode is used which may disturb the fully nested structure, the 8259A may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt (SEOI) must be issued which includes as part of the command the IS level to be reset. EOI is issued whenever EOI = 1, in OCW2, where L0-L2 is the binary level of the IS bit to be reset. Note that although the Rotate command can be issued together with an EOI where EOI = 1, it is not necessarily tied to it.

It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the 8259A is in the Special Mask Mode.

AUTOMATIC END OF INTERRUPT (AEOI) MODE

If AEOI = 1 in ICW4, then the 8259A will operate in AEOI mode continuously until reprogrammed by ICW4. In this mode the 8259A will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse (third pulse in MCS-80/85, second in MCS-86). Note that from a system standpoint, this mode should be used only when a nested multilevel interrupt structure is not required within a single 8259A.

To achieve automatic rotation (Rotate Mode A) within AEOI, there is a special rotate flip-flop. It is set by OCW2 with R = 1, SEOI = 0, EOI = 0, and cleared with R = 0, SEOI = 0, EOI = 0.

ROTATING PRIORITY MODE A (AUTOMATIC ROTATION) FOR EQUAL PRIORITY DEVICES

In some applications there are a number of interrupting devices of equal priority. In this mode a device, after being serviced, receives the lowest priority, so a device requesting an interrupt will have to wait, in the worst case until each of 7 other devices are serviced at most *once*. For example, if the priority and "in service" status is:

Before Rotate (IR4 the highest priority requiring service)

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
"IS" Status	0	1	0	1	0	0	0	0

	Lowest Priority							Highest Priority
Priority Status	7	6	5	4	3	2	1	0

After Rotate (IR4 was serviced, all other priorities rotated correspondingly)

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
"IS" Status	0	1	0	0	0	0	0	0

	Highest Priority							Lowest Priority
Priority Status	2	1	0	7	6	5	4	3

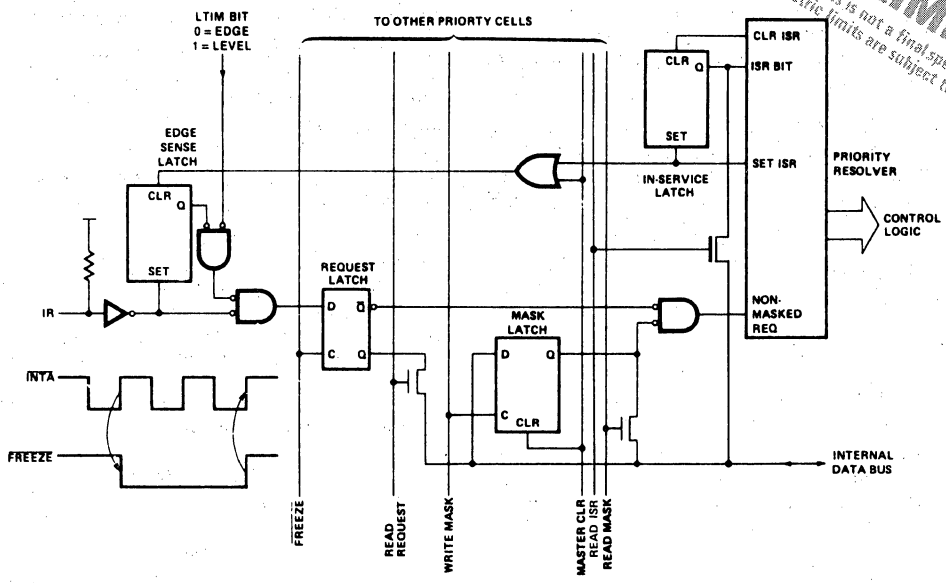
The Rotate command mode A is issued in OCW2 where: R = 1, EOI = 1, SEOI = 0. Internal status is updated by an End of Interrupt (EOI or AEOI) command. If R = 1, EOI = 0, SEOI = 0, a "Rotate-A" flip-flop is set. This is useful in AEOI, and described under Automatic End of Interrupt.

ROTATING PRIORITY MODE B (ROTATION BY SOFTWARE)

The programmer can change priorities by programming the bottom priority and thus fixing all other priorities; i.e., if IR5 is programmed as the bottom priority device, then IR6 will have the highest one.

The Rotate command is issued in OCW2 where: R = 1, SEOI = 1; L0-L2 is the binary priority level code of the bottom priority device.

Observe that in this mode internal status is updated by software control during OCW2. However, it is independent of the End of Interrupt (EOI) command (also executed by OCW2). Priority changes can be executed during an EOI command or independently.



NOTES

1. MASTER CLEAR ACTIVE ONLY DURING ICW1
2. FREEZE/ IS ACTIVE DURING INTA/ AND POLL SEQUENCES ONLY
3. TRUTH TABLE FOR D-LATCH

C	D	Q	OPERATION
1	Di	Di	FOLLOW
0	X	Qn-1	HOLD

Priority Cell — Simplified Logic Diagram

LEVEL TRIGGERED MODE

This mode is programmed using bit 3 in ICW1.

If LTIM ='1', an interrupt request will be recognized by a 'high' level on IR Input, and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued or the CPU interrupt is enabled to prevent a second interrupt from occurring.

The above figure shows a conceptual circuit to give the reader an understanding of the level sensitive and edge sensitive input circuitry of the 8259A. Be sure to note that the request latch is a transparent D type latch.

READING THE 8259A STATUS

The input status of several internal registers can be read to update the user information on the system. The following registers can be read by issuing a suitable OCW3 and reading with $\overline{RD}$.

Interrupt Mask Register: 8-bit register whose content specifies the interrupt request lines being masked. acknowledged. The highest request level is reset from the IRR when an interrupt is acknowledged. (Not affected by IMR.)

In-Service Register (ISR): 8-bit register which contains the priority levels that are being serviced. The ISR is updated when an End of Interrupt command is issued.

Interrupt Mask Register: 8-bit register which contains the interrupt request lines which are masked.

The IRR can be read when, prior to the $\overline{RD}$ pulse, a $\overline{WR}$ pulse is issued with OCW3 (ERIS = 1, RIS = 0.)

The ISR can be read in a similar mode when ERIS = 1, RIS = 1 in the OCW3.

There is no need to write an OCW3 before every status read operation, as long as the status read corresponds with the previous one; i.e., the 8259A "remembers" whether the IRR or ISR has been previously selected by the OCW3. This is not true when poll is used.

After initialization the 8259A is set to IRR.

For reading the IMR, no OCW3 is needed. The output data bus will contain the IMR whenever $\overline{RD}$ is active and A0 = 1.

Polling overrides status read when P = 1, ERIS = 1 in OCW3.

SUMMARY OF 8259A INSTRUCTION SET

Inst. #	Mnemonic		A0	D7	D6	D5	D4	D3	D2	D1	D0	Operation Description	
1	ICW1	A	0	A7	A6	A5	1	0	1	1	0		Format = 4, single, edge triggered
2	ICW1	B	0	A7	A6	A5	1	1	1	1	0		Format = 4, single, level triggered
3	ICW1	C	0	A7	A6	A5	1	0	1	0	0	Byte 1 Initialization	Format = 4, not single, edge triggered
4	ICW1	D	0	A7	A6	A5	1	1	1	0	0		Format = 4, not single, level triggered
5	ICW1	E	0	A7	A6	0	1	0	0	1	0	No ICW4 Required	Format = 8, single, edge triggered
6	ICW1	F	0	A7	A6	0	1	1	0	1	0		Format = 8, single, level triggered
7	ICW1	G	0	A7	A6	0	1	0	0	0	0		Format = 8, not single, edge triggered
8	ICW1	H	0	A7	A6	0	1	1	0	0	0		Format = 8, not single, level triggered
9	ICW1	I	0	A7	A6	A5	1	0	1	1	1		Format = 4, single, edge triggered
10	ICW1	J	0	A7	A6	A5	1	1	1	1	1		Format = 4, single, level triggered
11	ICW1	K	0	A7	A6	A5	1	0	1	0	1	Byte 1 Initialization	Format = 4, not single, edge triggered
12	ICW1	L	0	A7	A6	A5	1	1	1	0	1		Format = 4, not single, level triggered
13	ICW1	M	0	A7	A6	0	1	0	0	1	1	ICW4 Required	Format = 8, single, edge triggered
14	ICW1	N	0	A7	A6	0	1	1	0	1	1		Format = 8, single, level triggered
15	ICW1	O	0	A7	A6	0	1	0	0	0	1		Format = 8, not single, edge triggered
16	ICW1	P	0	A7	A6	0	1	1	0	0	1		Format = 8, not single, level triggered
17	ICW2		1	A15	A14	A13	A12	A11	A10	A9	A8	Byte 2 initialization	
18	ICW3	M	1	S7	S6	S5	S4	S3	S2	S1	S0	Byte 3 initialization — master	
19	ICW3	S	1	0	0	0	0	0	S2	S1	S0	Byte 3 initialization — slave	
20	ICW4	A	1	0	0	0	0	0	0	0	0	No action, redundant	
21	ICW4	B	1	0	0	0	0	0	0	0	1	Non-buffered mode, no AEOI, MCS-86	
22	ICW4	C	1	0	0	0	0	0	0	1	0	Non-buffered mode, AEOI, MCS-80/85	
23	ICW4	D	1	0	0	0	0	0	0	1	1	Non-buffered mode, AEOI, MCS-86	
24	ICW4	E	1	0	0	0	0	0	1	0	0	No action, redundant	
25	ICW4	F	1	0	0	0	0	0	1	0	1	Non-buffered mode, no AEOI, MCS-86	
26	ICW4	G	1	0	0	0	0	0	1	1	0	Non-buffered mode, AEOI, MCS-80/85	
27	ICW4	H	1	0	0	0	0	0	1	1	1	Non-buffered mode, AEOI, MCS-86	
28	ICW4	I	1	0	0	0	0	1	0	0	0	Buffered mode, slave, no AEOI, MCS-80/85	
29	ICW4	J	1	0	0	0	0	1	0	0	1	Buffered mode, slave, no AEOI, MCS-86	
30	ICW4	K	1	0	0	0	0	1	0	1	0	Buffered mode, slave, AEOI, MCS-80/85	
31	ICW4	L	1	0	0	0	0	1	0	1	1	Buffered mode, slave, AEOI, MCS-86	
32	ICW4	M	1	0	0	0	0	1	1	0	0	Buffered mode, master, no AEOI, MCS-80/85	
33	ICW4	N	1	0	0	0	0	1	1	0	1	Buffered mode, master, no AEOI, MCS-86	
34	ICW4	O	1	0	0	0	0	1	1	1	0	Buffered mode, master, AEOI, MCS-80/85	
35	ICW4	P	1	0	0	0	0	1	1	1	1	Buffered mode, master, AEOI, MCS-86	
36	ICW4	NA	1	0	0	0	1	0	0	0	0	Fully nested mode, MCS-80, non-buffered, no AEOI	
37	ICW4	NB	1	0	0	0	1	0	0	0	1	ICW4 NB through ICW4 ND are identical to ICW4 B through ICW4 D with the addition of Fully Nested Mode	
38	ICW4	NC	1	0	0	0	1	0	0	1	0		
39	ICW4	ND	1	0	0	0	1	0	0	1	1		
40	ICW4	NE	1	0	0	0	1	0	1	0	0	Fully Nested Mode, MCS-80/85, non-buffered, no AEOI	
41	ICW4	NF	1	0	0	0	1	0	1	0	1	ICW4 NF through ICW4 NP are identical to ICW4 F through ICW4 P with the addition of Fully Nested Mode	
42	ICW4	NG	1	0	0	0	1	0	1	1	0		
43	ICW4	NH	1	0	0	0	1	0	1	1	1		
44	ICW4	NI	1	0	0	0	1	1	0	0	0		
45	ICW4	NJ	1	0	0	0	1	1	0	0	1		
46	ICW4	NK	1	0	0	0	1	1	0	1	0		
47	ICW4	NL	1	0	0	0	1	1	0	1	1		
48	ICW4	NM	1	0	0	0	1	1	1	0	0		
49	ICW4	NN	1	0	0	0	1	1	1	0	1		
50	ICW4	NO	1	0	0	0	1	1	1	1	0		
51	ICW4	NP	1	0	0	0	1	1	1	1	1		
52	OCW1		1	M7	M6	M5	M4	M3	M2	M1	M0	Load mask register, read mask register	
53	OCW2	E	0	0	0	1	0	0	0	0	0	Non-specific EOI	
54	OCW2	SE	0	0	1	1	0	0	L2	L1	L0	Specific EOI. L0-L2 code of IS FF to be reset	
55	OCW2	RE	0	1	0	1	0	0	0	0	0	Rotate at EOI Automatically (Mode A)	
56	OCW2	RSE	0	1	1	1	0	0	L2	L1	L0	Rotate at EOI (mode B). L0-L2 code of line	
57	OCW2	R	0	1	0	0	0	0	0	0	0	Set Rotate A FF	
58	OCW2	CR	0	0	0	0	0	0	0	0	0	Clear Rotate A FF	
59	OCW2	RS	0	1	1	0	0	0	L2	L1	L0	Rotate priority (mode B) independently of EOI	
60	OCW3	P	0	0	0	0	0	1	1	0	0	Poll mode	
61	OCW3	RIS	0	0	0	0	0	1	0	1	1	Read IS register	

SUMMARY OF 8259A INSTRUCTION SET (Cont.)

Inst. #	Mnemonic	A0	D7	D6	D5	D4	D3	D2	D1	D0	Operation Description
62	OCW3 RR	0	0	0	0	0	1	0	1	0	Read request register
63	OCW3 SM	0	0	1	1	0	1	0	0	0	Set special mask mode
64	OCW3 RSM	0	0	1	0	0	1	0	0	0	Reset special mask mode

Note: 1. In the master mode $\overline{SP}$ pin = 1, in slave mode $\overline{SP}$ = 0

Cascading

The 8259A can be easily interconnected in a system of one master with up to eight slaves to handle up to 64 priority levels.

A typical MCS-80/85 system is shown in Figure 2. The master controls, through the 3 line cascade bus, which one of the slaves will release the corresponding address.

As shown in Figure 2, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the device routine address during bytes 2 and 3 of INTA. (Byte 2 only for MCS-86). The IRO input should not be connected to a slave 8259A unless IR1-IR7 also have slaves attached.

The cascade bus lines are normally low and will contain the slave address code from the trailing edge of the first $\overline{INTA}$ pulse to the trailing edge of the third pulse. It is obvious that each 8259A in the system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI command must be issued twice: once for the master and once for the corresponding slave. An address decoder is required to activate the Chip Select ($\overline{CS}$) input of each 8259A.

The cascade lines of the Master 8259A are activated for any interrupt input, even if no slave is connected to that input.

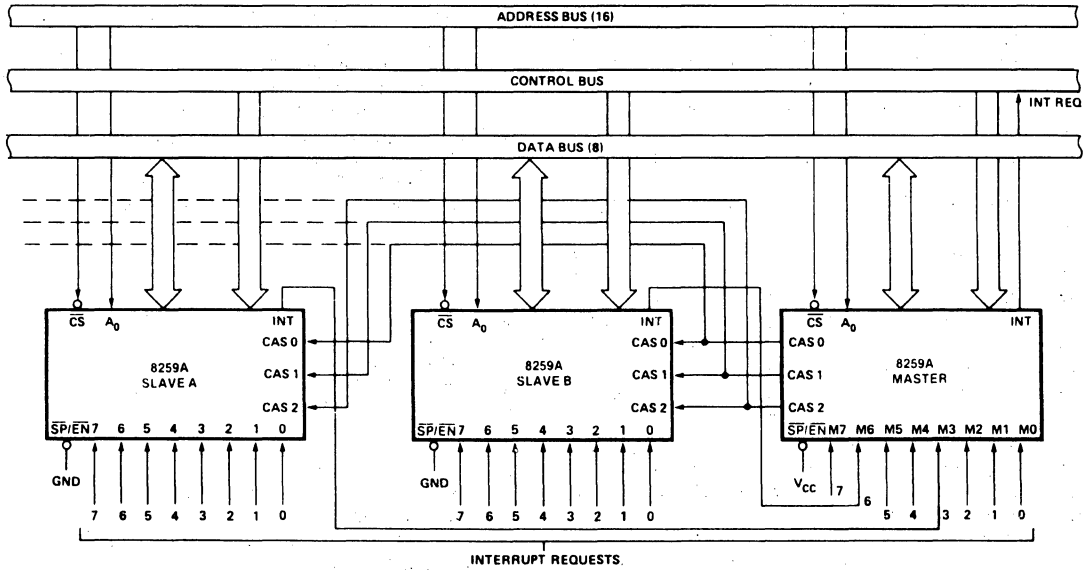


Figure 2. Cascading the 8259A

PIN FUNCTIONS

Name	I/O	Pin #	Function
V_{CC}		28	+5V supply.
GND		14	Ground.
D_{0-7}	I/O	11-4	Bidirectional data bus, used for: a) programming the mode of the 8259A (programming is done by software); b) the microprocessor can read the status of the 8259A; c) the 8259A will send vectoring data to the microprocessor when an interrupt is acknowledged.
IR_{0-7}	I	18-25	Interrupt Requests: These are asynchronous inputs. A positive-going edge will generate an interrupt request. Thus a request can be generated by raising the line and holding it high until acknowledged, or by a negative pulse. In level triggered mode, no edge is required. These lines are active HIGH.
$\overline{RD}$	I	3	Read (generally from 8228 in MCS-80 system or from 8086 in MCS-86 system).
$\overline{WR}$	I	2	Write (generally from 8228 in MCS-80 sytem or from 8086 in MCS-86 system).
$\overline{INTA}$	I	26	Interrupt Acknowledge (generally from 8228 in MCS-80 system, 8086 in MCS-86 system). The 8228 generates three distinct $\overline{INTA}$ pulses when a CALL is inserted, the 8086 produces two distinct $\overline{INTA}$ pulses during an interrupt cycle.
$\overline{CS}$	I	1	Chip Select: RD and WR are enabled by Chip Select, whereas Interrupt Acknowledge is independent of Chip Select.
A0	I	27	Usually the least significant bit of the microprocessor address output (A1 in MCS-86 system). When A0 = 1 the Interrupt Mask Register can be loaded or read. When A0 = 0 the 8259A mode can be programmed or its status can be read. $\overline{CS}$ is active LOW.
INT	O	17	Goes directly to the microprocessor interrrupt input. This output will have high V_{OH} to match the 8080 3.3V V_{IH}. INT is active HIGH.
C0-C2	I/O	12 13 15	Three cascade lines, outputs in master mode and inputs in slave mode. The master issues the binary code of the acknowledged interrupt level on these lines. Each slave compares this code with its own.
$\overline{SP}/\overline{EN}$	I/O	16	$\overline{SP}/\overline{EN}$ is a dual function pin. In the buffered mode $\overline{SP}/\overline{EN}$ is used to enable bus transceivers ($\overline{EN}$). In the non-buffered mode $\overline{SP}/\overline{EN}$ determines if this 8259A is a master or a slave. If $\overline{SP}$ = 1 the 8259A is master; $\overline{SP}$ = 0 indicates a slave.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... −40°C to 85°C
Storage Temperature ..... −65°C to +150°C
Voltage On Any Pin
With Respect to Ground ..... −0.5V to +7V
Power Dissipation ..... 1 Watt

*COMMENT
Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ± 10% (8259-A), V_{CC} = 5V ± 10% (8259A)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	-.5	V		
V_{IH}	Input High Voltage	2.0	V_{CC} + .5V	V	
V_{OL}	Output Low Voltage		.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
$V_{OH(INT)}$	Interrupt Output High Voltage	3.5 2.4		C V	I_{OH} = −100 μA I_{OH} = −400 μA
I_{LI}	Input Load Current		10	μA	V_{IN} = V_{CC} to 0V
I_{LOL}	Output Leakage Current		-10	μA	V_{OUT} = 0.45V
I_{CC}	V_{CC} Supply Current		85	mA	
I_{LIR}	IR Input Load Current		-300 10	μA μA	V_{IN} = 0 V_{IN} = V_{CC}

8259A A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$ $V_{CC} = 5V \pm 5\%$ (8259A-8) $V_{CC} = 5V \pm 10\%$ (8259A)

TIMING REQUIREMENTS

Symbol	Parameter	8259A-8 Min.	8259A-8 Max.	8259A Min.	8259A Max.	Units	Test Conditions
TAHRL	A0/$\overline{CS}$ Setup to $\overline{RD}$/$\overline{INTA}$↓	50		0		ns	
TRHAX	A0/$\overline{CS}$ Hold after $\overline{RD}$/$\overline{INTA}$↑	5		0		ns	
TRLRH	$\overline{RD}$ Pulse Width	420		235		ns	
TAHWL	A0/$\overline{CS}$ Setup to$\overline{WR}$↓	50		0		ns	
TWHAX	A0/$\overline{CS}$ Hold after $\overline{WR}$↑	20		0		ns	
TWLWH	$\overline{WR}$ Pulse Width	400		290		ns	
TDVWH	Data Setup to $\overline{WR}$↑	300		240		ns	
TWHDX	Data Hold after $\overline{WR}$↑	40		0		ns	
TJLJH	Interrupt Request Width (Low)	100		100		ns	See Note 1
TCVIAL	Cascade Setup to Second or Third $\overline{INTA}$↓ (Slave Only)	55		55		ns	
TRHRL	End of $\overline{RD}$ to Next Command	300		160		ns	
TWHRL	End of $\overline{WR}$ to Next Command	370		190		ns	

Note: 1. This is the low time required to clear the input latch in the edge triggered mode.

TIMING RESPONSES

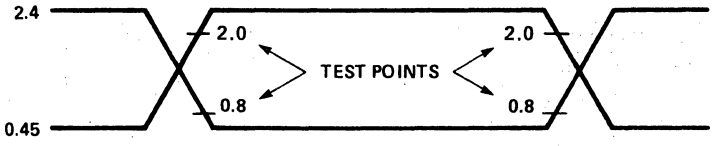
Symbol	Parameter	8259A-8 Min.	8259A-8 Max.	8259A Min.	8259A Max.	Units	Test Conditions
TRLDV	Data Valid from $\overline{RD}$/$\overline{INTA}$↓		300		200	ns	C of Data Bus Max. test C = 100 pF Min. test C = 15 pF C_{INT} = 100 pF C_{ENABLE} = 15pF
TRHDZ	Data Float after $\overline{RD}$/$\overline{INTA}$↑	10	200		100	ns	
TJHIH	Interrupt Output Delay		400		350	ns	
TIALCV	Cascade Valid from First $\overline{INTA}$↓ (Master Only)		565		565	ns	
TRLEL	Enable Active from $\overline{RD}$↓ or $\overline{INTA}$↓		160		125	ns	
TRHEH	Enable Inactive from $\overline{RD}$↑ or $\overline{INTA}$↑		325		150	ns	
TAHDV	Data Valid from Stable Address		350		200	ns	
TCVDV	Cascade Valid to Valid Data		300		300	ns	

CAPACITANCE

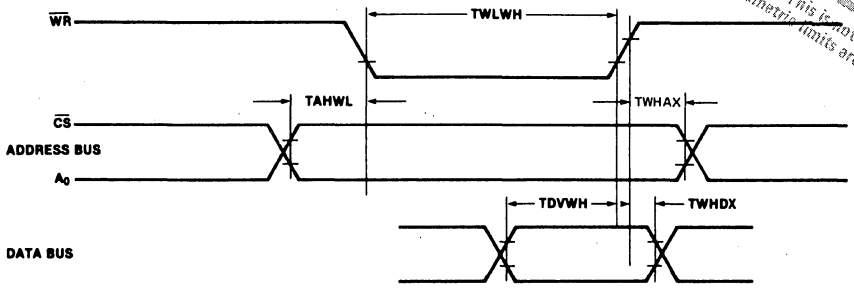
$T_A = 25°C$; $V_{CC} = GND = 0V$

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	fc = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}

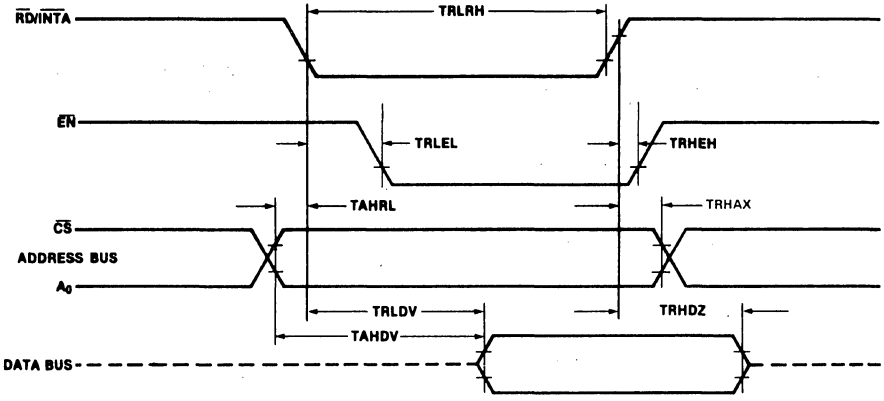
Input and Output Waveforms for A.C. Tests



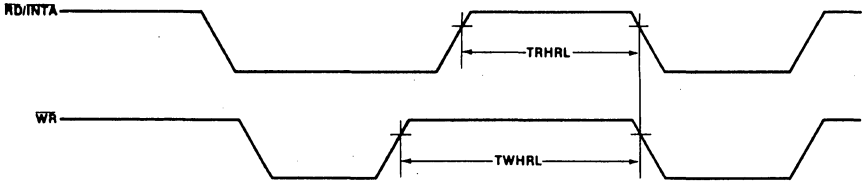
WRITE MODE



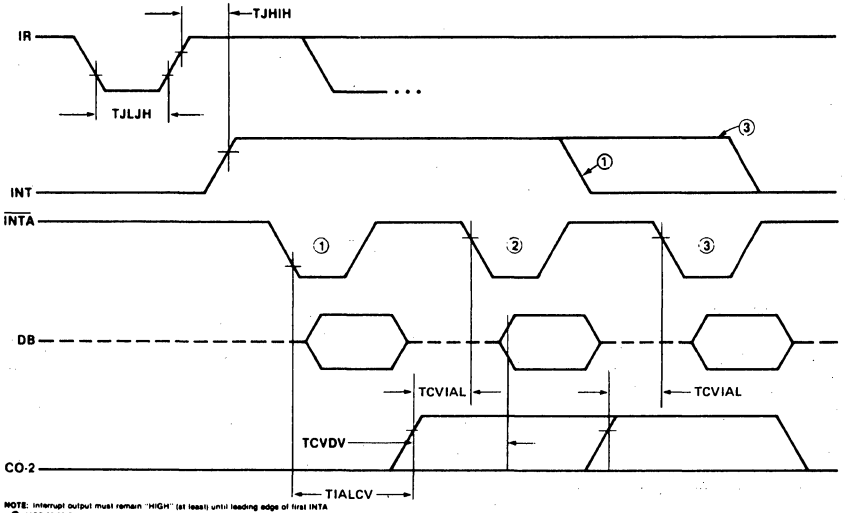
READ/INTA MODE



OTHER TIMING



INTA SEQUENCE



NOTE: Interrupt output must remain "HIGH" (at least) until leading edge of first INTA
② MCS 80/85 Systems only
① Cycle 1 in MCS 86 Systems, the Data Bus is not active

intel®

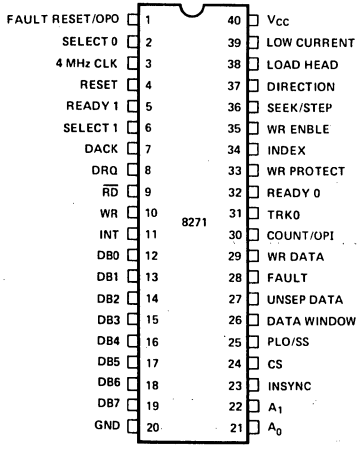
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8271
PROGRAMMABLE FLOPPY DISK CONTROLLER

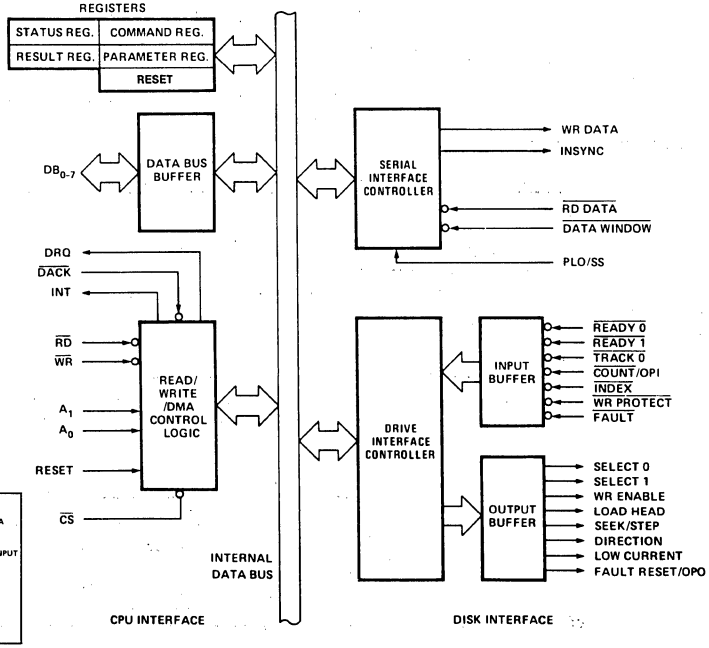
- **IBM 3740 Soft Sectored Format Compatible**
- **Programmable Record Lengths**
- **Multi-Sector Capability**
- **Maintain Dual Drives with Minimum Software Overhead Expandable to 4 Drives**
- **Automatic Read/Write Head Positioning and Verification**
- **Internal CRC Generation and Checking**
- **Programmable Step Rate, Settle-Time, Head Load Time, Head Unload Index Count**
- **Fully MCS-80 and MCS-85 Compatible**
- **Single +5V Supply**
- **40-Pin Package**

The Intel® 8271 Programmable Floppy Disk Controller (FDC) is an LSI component designed to interface one to 4 floppy disk drives to an 8-bit microcomputer system. Its powerful control functions minimize both hardware and software overhead normally associated with floppy disk controllers.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

$DB_7 - DB_0$	DATA BUS (BI-DIRECTIONAL)	PLO/SS	PLO/SINGLE SHOT
CLK	CLOCK INPUT (TTL)	DATA WINDOW	DATA WINDOW
SELECT 1, 0	SELECT 1, 0	UNSEP DATA	UNSEPARATED DATA
FAULT RESET/OPO	FAULT RESET/OPTIONAL OUTPUT	FAULT	FAULT
RESET	CHIP RESET	WR DATA	WRITE DATA
READY 1, 0	READY 1, 0	COUNT/OPI	COUNT/OPTIONAL INPUT
DACK	DMA ACKNOWLEDGE	TRK 0	TRACK 0
DRQ	DMA REQUEST	WR PROTECT	WRITE PROTECT
RD	CPU READ INPUT	INDEX	INDEX
WR	CPU WRITE INPUT	WR ENABLE	WRITE ENABLE
INT	INTERRUPT	SEEK/STEP	SEEK/STEP
$A_1, 0$	REGISTER SELECT	DIRECTION	DIRECTION
INSYNC	READ DATA INSYNC	LOAD HEAD	LOAD HEAD
CS	CHIP SELECT	LOW CURRENT	LOW CURRENT

8271 BASIC FUNCTIONAL DESCRIPTION

General

The 8271 Floppy Disk Controller (FDC) interfaces either two single or one dual floppy drive to an eight bit microprocessor and is fully compatible with Intel's new high performance MCS-85 microcomputer system. With minimum external circuitry, this innovative controller supports most standard, commonly-available flexible disk drives including the mini-floppy.

The 8271 FDC supports a comprehensive soft sectored format which is IBM 3740 compatible and includes provision for the designating and handling of bad tracks. It is a high level controller that relieves the CPU (and user) of many of the control tasks associated with implementing a floppy disk interface. The FDC supports a variety of high level instructions which allow the user to store and retrieve data on a floppy disk without dealing with the low level details of disk operation.

In addition to the standard read/write commands, a scan command is supported. The scan command allows the user program to specify a data pattern and instructs the FDC to search for that pattern on a track. Any application that is required to search the disk for information (such as point of sale price lookup, disk directory search, etc.), may use the scan command to reduce the CPU overhead. Once the scan operation is initiated, no CPU intervention is required.

Hardware Description

The 8271 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Pin Name	Pin No.	I/O	Description
V_{CC}	(40)		+5V supply
GND	(20)		Ground
Clock	(3)	I	A square wave clock
Reset	(4)	I	A high signal on the reset input forces the 8271 to an idle state. The 8271 remains idle until a command is issued by the CPU. The output signals of the drive interface are forced inactive (LOW). Reset must be active for 10 or more clock cycles.
$\overline{CS}$	(24)	I	The I/O Read and I/O Write inputs are enabled by the chip select signal.
DB_7-DB_0	(19-12)	I/O	The Data Bus lines are bidirectional, three-state lines (8080 data bus compatible).
$\overline{WR}$	(10)	I	The Write signal is used to signal the control logic that a transfer of data from the data bus to the 8271 is required.
$\overline{RD}$	(9)	I	The Read signal is used to signal the control logic that a transfer of data from the 8271 to the data bus is required.
INT	(11)	O	The interrupt signal indicates that the 8271 requires service.
A_1-A_0	(22-21)	I	These two lines are CPU Interface Register select lines.
DRQ	(8)	O	The DMA request signal is used to request a transfer of data between the 8271 and memory.
$\overline{DACK}$	(7)	I	The DMA acknowledge signal notifies the 8271 that a DMA cycle has been granted. For non-DMA transfers, this signal should be driven in the manner of a "Chip Select".
Select 1– Select 0	(6) (2)	O	These lines are used to specify the selected drive. These lines are set by the command byte.
Fault Reset/ OPO	(1)	O	The optional fault reset output line is used to reset an error condition which is latched by the drive. If this line is not used for a fault reset it can be used as an optional output line. This line is set with the write special register command.
Write Enable	(35)	O	This signal enables the drive write logic.
Seek/Step	(36)	O	This multi-function line is used during drive seeks.
Direction	(37)	O	The direction line specifies the seek direction. A high level on this pin steps the R/W head toward the spindle (step-in), a low level steps the head away from the spindle (step-out).
Load Head	(38)	O	The load head line causes the drive to load the Read/Write head against the diskette.
Low Current	(39)	O	This line notifies the drive that track 43 or greater is selected.
$\overline{Ready\ 1}$, $\overline{Ready\ 0}$	(5) (32)	I	These two lines indicate that the specified drive is ready.
$\overline{Fault}$	(28)	I	This line is used by the drive to specify a file unsafe condition.
$\overline{Count/OPI}$	(30)	I	If the optional seek/direction/count seek mode is selected, the count pin receives pulses to step the R/W head to the desired track. Otherwise, this line can be used as an optional input.
$\overline{Write\ Protect}$	(33)	I	This signal specifies that the diskette inserted is write protected.
$\overline{TRK0}$	(31)	I	This signal indicates when the R/W head is positioned over track zero.
$\overline{Index}$	(34)	I	The index signal gives an indication of the relative position of the diskette.
PLO/SS	(25)	I	This pin is used to specify the type of data separator used. Phase-Locked Oscillator/Single Shot.
Write Data	(29)	O	Composite write data.

Pin Name	Pin No.	I/O	Description
$\overline{\text{Unseparated Data}}$	(27)	I	This input is the unseparated data and clocks.
$\overline{\text{Data Window}}$	(26)	I	This is a data window established by a single-shot or phase-locked oscillator data separator.
INSYNC	(23)	O	This line is high when 8271 has attained input data synchronization, by detecting 2 bytes of zeros followed by an expected Address Mark. It will stay high until the end of the ID or data field.

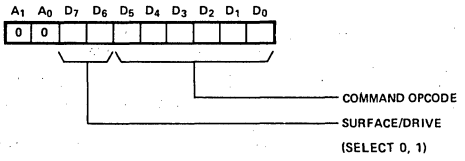
CPU Interface Description

This interface minimizes CPU involvement by supporting a set of high level commands and both DMA and non-DMA type data transfers and by providing hierarchical status information regarding the result of command execution.

The CPU utilizes the control interface (see the Block diagram) to specify the FDC commands and to determine the result of an executed command. This interface is supported by five Registers which are addressed by the CPU via the A_1, A_0, $\overline{RD}$ and $\overline{WR}$ signals. If an 8080 based system is used, the $\overline{RD}$ and $\overline{WR}$ signals can be driven by the 8228's $\overline{I/OR}$ and $\overline{I/OW}$ signals. The registers are defined as follows:

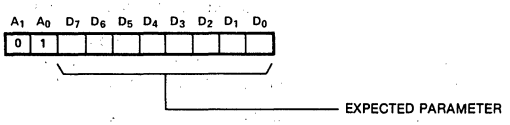
Command Register

The CPU loads an appropriate command into the Command Register which has the following format:



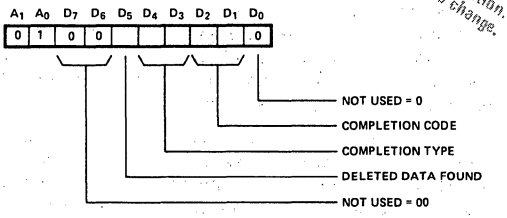
Parameter Register

Accepts parameters of commands that require further description; up to five parameters may be required, example:



Result Register

The Result Register is used to supply the outcome of FDC command execution (such as a good/bad completion) to the CPU. The standard Result byte format is:



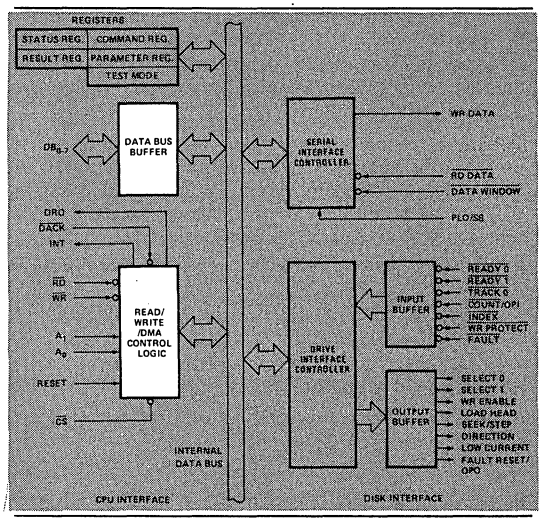
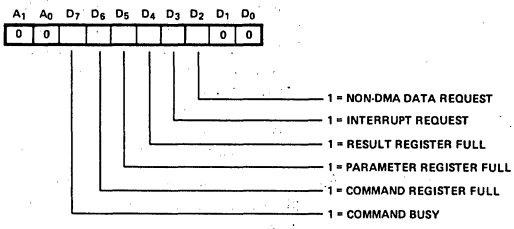


Figure 1. 8271 Block Diagram Showing CPU Interface Functions

Status Register

Reflects the state of the FDC.



Reset Register

Allows the 8271 to be reset by the program. Reset must be active for 11 or more chip clocks.

INT (Interrupt Line)

Another element of the control interface is the Interrupt line (INT). This line is used to signal the CPU that an FDC operation has been completed. It remains active until the result register is read.

DMA Operation

The 8271 can transfer data in either DMA or non DMA mode. The data transfer rate of a floppy disk drive is high enough (one byte every 32 μsec) to justify DMA transfer. In DMA mode the elements of the DMA interface are:

DRQ: DMA Request:
The DMA request signal is used to request a transfer of data between the 8271 and memory.

$\overline{\text{DACK}}$: DMA Acknowledge:
The DMA acknowledge signal notifies the 8271 that a DMA cycle has been granted.

$\overline{\text{RD}}$, $\overline{\text{WR}}$: Read, Write
The read and write signals are used to specify the direction of the data transfer.

DMA transfers require the use of a DMA controller such as the Intel®8257. The function of the DMA controller is to provide sequential addresses and timing for the transfer at a starting address determined by the CPU. Counting of data block lengths is performed by the FDC.

To request a DMA transfer, the FDC raises DRQ. DACK and RD enable DMA data onto the bus (independently of CHIP SELECT). DACK and WR transfer DMA data to the FDC. If a data transfer request (read or write) is not serviced within 31 μsec, the command is cancelled, a late DMA status is set, and an interrupt is generated. In DMA mode, an interrupt is generated at the completion of the data block transfer.

When configured to transfer data in non-DMA mode, the CPU must pass data to the FDC in response to the non-DMA data requests indicated by the status word. The data is passed to and from the chip by asserting the DACK and the RD or WR signals. Chip select should be inactive (HIGH).

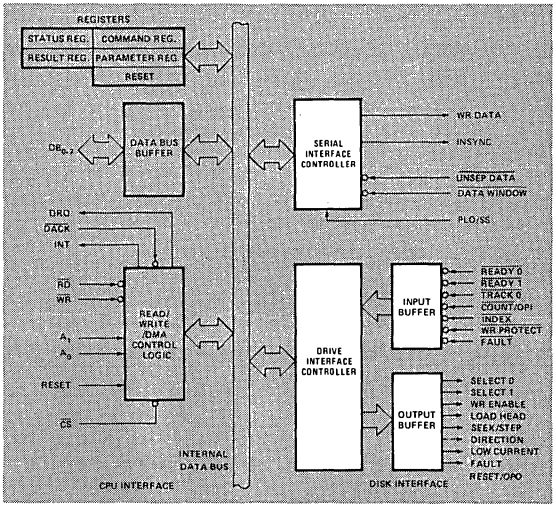


Figure 2. 8271 Block Diagram Showing Disk Interface Functions

Disk Drive Interface

The 8271 disk drive interface supports the high level command structure described in the Command Description section. The 8271 maintains the location of bad tracks and the current track location for two drives. However, with minor software support, this interface can support four drives by expanding the two drive select lines (select 0, select 1) with the addition of minimal support hardware.

The FDC Disk Drive Interface has the following major functions.

READ FUNCTIONS

Utilize the user supplied data window to obtain the clock and data patterns from the unseparated read data.

Establish byte synchronization.

Compute and verify the ID and data field CRCs.

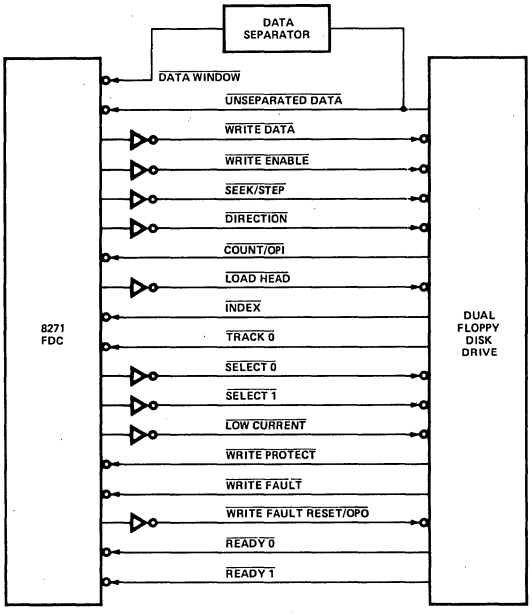
WRITE FUNCTIONS

Encode composite write data.

Compute the ID and data field CRCs and append them to their respective fields.

CONTROL FUNCTIONS

Generate the programmed step rate, head load time, head settling time, head unload delay, and monitor drive functions.



NOTE: INPUTS TO CHIP MAY REQUIRE RECEIVERS (AT LEAST PULL UP/DOWN PAIRS).

Figure 3. 8271 Disk Drive Interface

Data Separation

The 8271 needs only a data window to separate the data from the composite read data as well as to detect missing clocks in the Address Marks.

The window generation logic may be implemented using either a single-shot separator or a phase-locked oscillator.

Single-Shot Separator

The single-shot separator approach is the lowest cost solution.

The FDC samples the value of Data Window on the leading edge of Unseparated Data and determines whether the delay from the previous pulse was a half or full bit-cell (high input = full bit-cell, low input = half bit-cell).
PLO/SS should be tied to Ground.

Insync Pin

This pin gives an indication of whether the 8271 is synchronized with the serial data stream during read operations. This pin can be used with a phase-locked oscillator for soft and hard locking.

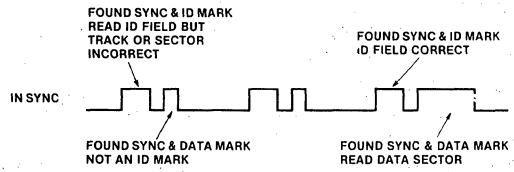
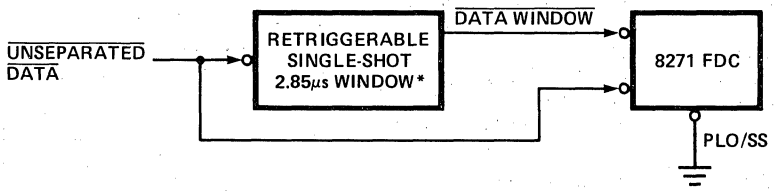


Figure 4. Insync Waveform



*FOR MINI-FLOPPY DATA WINDOW = 5.7μsec

Figure 5. Single-Shot Data Separator Block Diagram

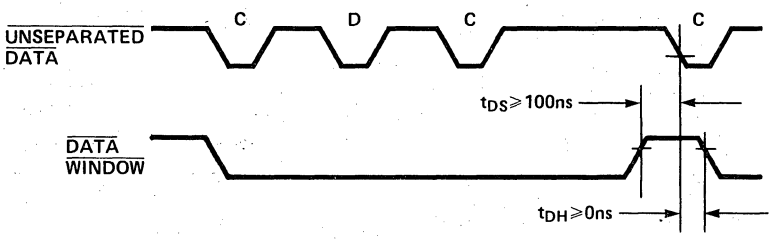


Figure 6. Single-Shot Data Window Timing

Phase-Locked Oscillator Separator

The FDC samples the value of Data Window on the leading edge of Unseparated Data and determines whether the pulse represents a Clock or Data Pulse.

PLO/SS should be tied to V_{CC} (+5V).

Insync may be used to provide soft and hard locking control for the phase-locked oscillator.

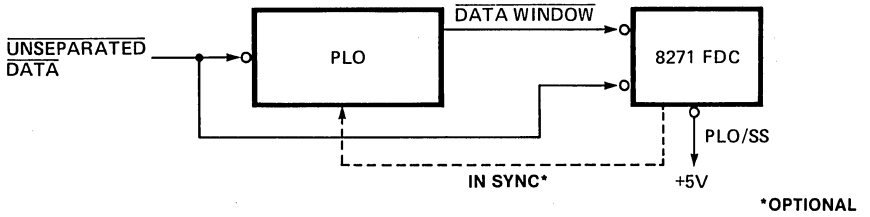
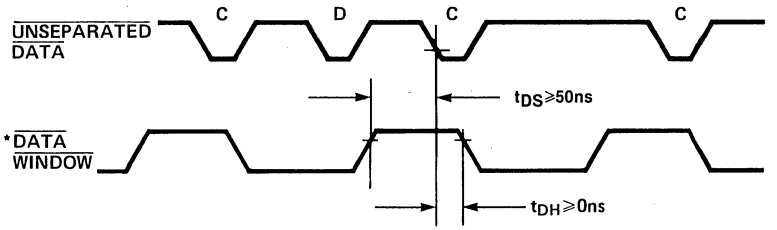


Figure 7. PLO Data Separator Block Diagram



*DATA WINDOW MAY BE 180° OUT OF PHASE IN PLO DATA SEPARATION MODE.

Figure 8. PLO Data Window Timing

Disk Drive Control Interface

The disk drive control interface performs the high level and programmable flexible disk drive operations. It custom tailors many varied drive performance parameters such as the step rate, settling time, head load time, and head unload index count. The following is the description of the control interface.

Write Enable

The Write Enable controls the read and write functions of a flexible disk drive. When Write Enable is a logical one, it enables the drive write electronics to pass current through the Read/Write head. When Write Enable is a logical zero, the drive Write circuitry is disabled and the Read/Write head detects the magnetic flux transitions recorded on a diskette. The write current turn-on is as follows.

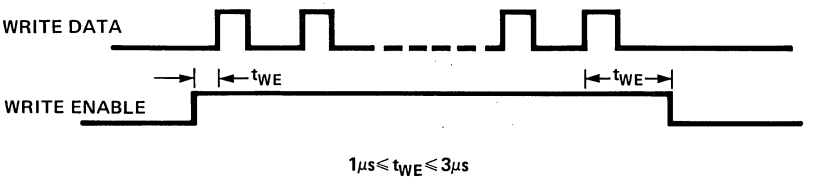


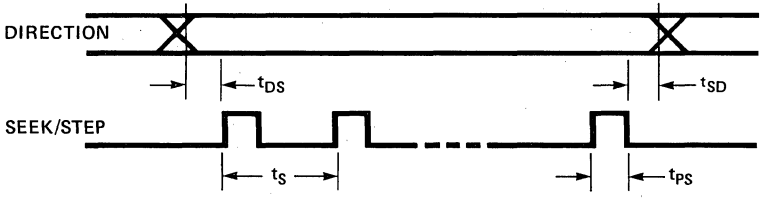
Figure 9. Write Enable Timing

Seek Control

Seek Control is accomplished by Seek/Step, Direction, and Count pins and can be implemented two ways to provide maximum flexibility in the subsystem design. One instance is when the programmed step rate is not equal to zero. In this case, the 8271 uses the Seek/Step and Direction pins (the Seek/Step pin becomes a Step pin). Programmable Step timing parameters are shown.

Another instance is when the programmable step rate is equal to zero, in which case the 8271 holds the seek line high until the appropriate number of user-supplied step pulses have been counted on the count input pin.

The Direction pin is a control level indicating the direction in which the R/W head is stepped. A logic high level on this line moves the head toward the spindle (step-in). A logic low level moves the head away from the spindle (step-out).

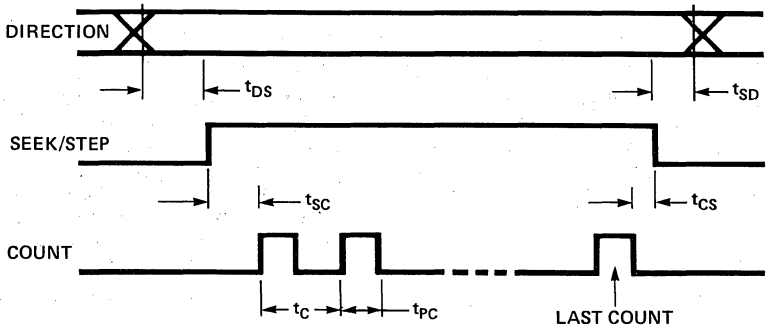


$t_{PS}=t_{DS}=t_{SD}=10\mu s$

STANDARD: $1ms \leqslant t_S \leqslant 255ms$

MINI-FLOPPY: $2ms \leqslant t_s \leqslant 510ms$

Figure 10. Seek Timing



$t_{DS}=t_{SD}=t_{CS}=10\mu s$

$t_{SC} \geqslant 1\mu s$

$t_{PC} \geqslant 20\mu s$

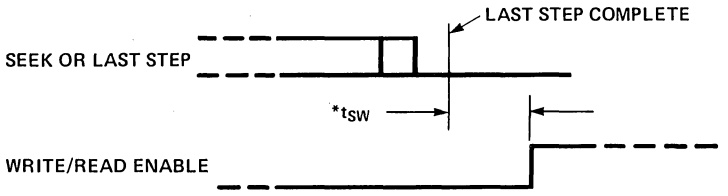
$t_C \geqslant 1ms$

Figure 11. Seek/Step/Count Timing

Head Seek Settling Time

The 8271 allows the head settling time to be programmed from 0 to 255ms, in increments of 1ms.

The head settling time is defined as the interval of time from completion of the last step to the time when reading or writing on the diskette is possible (R/W Enable). The R/W head is assumed loaded.



STANDARD: $0 \leq {}^*t_{SW} \leq 255ms$

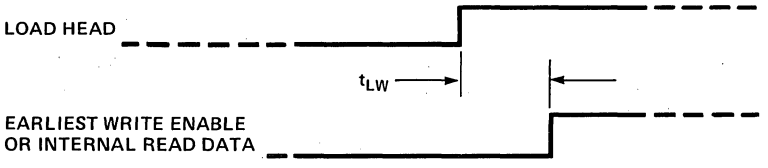
MINI-FLOPPY: $0 \leq {}^*t_{SW} \leq 510ms$

*R/W HEAD IS ASSUMED LOADED.

Figure 12. Head Load Settling Timing

Load Head

When active, load head output pin causes the drive's read/write head to be loaded on the diskette. When the head is initially loaded, there is a programmed delay (0 to 60ms in 4ms increments) prior to any read or write operation. Provision is also made to unload the head following an operation within a programmed number of diskette revolutions.



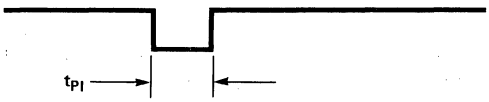
STANDARD: $0 \leq t_{LW} \leq 60ms$

MINI-FLOPPY: $0 \leq t_{LW} \leq 120ms$

Figure 13. Head Load to Read/Write Timing

Index

The Index input is used to determine "Sector not found" status and to initiate format track/read ID commands and head unload Index and Count operations.



$t_{PI} \geqslant 0.5\mu s$

Figure 14. Index Timing

Track 0

This input pin indicates that the diskette is at track 0. During any seek operation, the stepping out of the actuator ceases when the track 0 pin becomes active.

Select 1, 0

Only one drive may be selected at a time. The Input/Output pins that must be externally qualified with Select 0 and Select 1 are:

Unseparated Data
Data Window
Write Enable
Seek/Step
Count/Optional Input
Load Head
Track 0
Low Current
Write Protect
Write Fault
Fault Reset/Optional Output
Index

When a new set of select bits is specified by a new command or the FDC finishes the index count before head unload, the following pins will be set to the 0 state:

Write Enable (35)
Seek/Step (36)
Direction (37)
Load Head (38)
Low Head Current (39)

The select pins will be set to the state specified by the command or both are set to zero following the index count before head unload.

Low Current

This output pin is active whenever the physical track location of the selected drive is greater than 43. Generally this signal is used to enable compensation for the lower velocities encountered while recording on the inner tracks.

Write Protect

The 8271 will not write to a disk when this input pin is active and will interrupt the CPU if a Write attempt is made. Operations which check Write Protect are aborted if the Write Protect line is active.

This signal normally originates from a sensor which detects the presence or absence of the Write Protect hole in the diskette jacket.

Write Fault and Write Fault Reset

The Write Fault input is normally latched by the drive and indicates any condition which could endanger data integrity. The 8271 interrupts the CPU anytime Write Fault is detected during an operation and immediately resets the Write Enable, Seek/Step, Direction, and Low Current signals. The write fault condition can be cleared by using the write fault reset pin. If the drive being used does not support write fault, then this pin should be connected to V_{CC} through a pull-up resistor.

Ready 1, 0

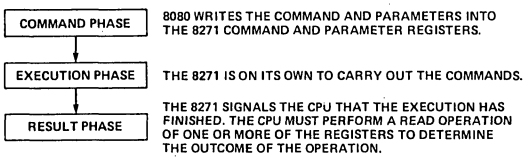
These two pins indicate the functional status of the disk drives. Whenever an operation is attempted on a drive which is not ready, an interrupt is generated. The interface continually monitors this input during an operation and if a Not Ready condition occurs, immediately terminates the operation. Note that the 8271 latches the Not Ready condition and it can only be reset by the execution of a Read Drive Status command. For drives that do not support a ready signal, either one can be derived with a one shot and the index pulse, or the ready inputs can be grounded and Ready determined through some software means.

PRINCIPLES OF OPERATION

As an 8080 peripheral device, the 8271 accepts commands from the CPU, executes them and provides a RESULT back to the 8080 CPU at the end of command execution. The communication with the CPU is established by the activation of $\overline{CS}$ and $\overline{RD}$ or $\overline{WR}$. The A_1, A_0 inputs select the appropriate registers on the chip:

$\overline{DACK}$	$\overline{CS}$	A_1	A_0	$\overline{RD}$	$\overline{WR}$	Operation
1	0	0	0	0	1	Read Status
1	0	0	0	1	0	Write Command
1	0	0	1	0	1	Read Result
1	0	0	1	1	0	Write Parameter
1	0	1	0	1	0	Write Reset Reg.
0	1	X	X	1	0	Write Data
0	1	X	X	0	1	Read Data
0	0	X	X	X	X	Not Allowed

The FDC operation is composed of the following sequence of events.



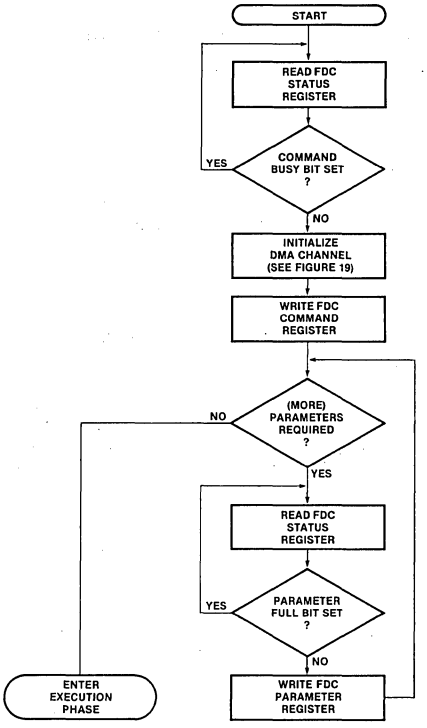
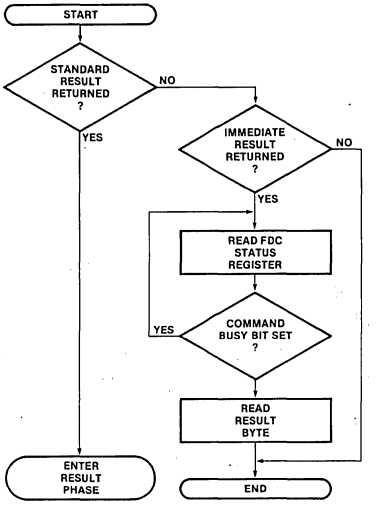


Figure 15. Passing the Command and Parameters to the 8271

The Command Phase

The software writes a command to the command register. As a function of the command issued, from zero to five parameters are written to the parameter register. Refer to diagram showing a flow chart of the command phase. Note that the flow chart shows that a command may not be issued if the FDC status register indicates that the device is busy. Issuing a command while another command is in progress is illegal. The flow chart also shows a parameter buffer full check. The FDC status indicates the state of the parameter buffer. If a parameter is issued while the parameter buffer is full, the previous parameter is over written and lost.



NOTE:

STANDARD RESULT RETURNED CAN BE DETERMINED BY MASKING OUT THE DRIVE SELECT BITS OF THE COMMAND BYTE (BITS 7 AND 6) AND CHECKING FOR A VALUE OF LESS THAN $2C_{16}$ (IF LESS THAN $2C_{16}$, STANDARD RESULT IS RETURNED).

IMMEDIATE RESULT RETURNED CAN BE DETERMINED BY ADDITIONALLY MASKING OUT BITS 5 AND 4 OF THE COMMAND BYTE AND CHECKING FOR A VALUE OF C_{16} OR GREATER (IF C_{16} OR GREATER, IMMEDIATE RESULT RETURNED).

Figure 16. Checking for Result Type Following 8271 Command and Parameters

The Execution Phase

During the execution phase the operation specified during the command phase is performed. During this phase, there is no CPU involvement if the system utilizes DMA for the data transfers. The execution phase of each command is discussed within the detailed command descriptions. The following table summarizes many of the basic execution phase characteristics.

EXECUTION PHASE BASIC CHARACTERISTICS

The following table summarizes the various commands with corresponding execution phase characteristics.

COMMANDS	1 Deleted Data	2 Head	3 Ready	4 Write/ Protect	5 Seek	6 Seek Check	7 Result	8 Completion Interrupt
SCAN DATA	SKIP	LOAD	√	x	YES	YES	YES	YES
SCAN DATA AND DEL DATA	XFER	LOAD	√	x	YES	YES	YES	YES
WRITE DATA	x	LOAD	√	√	YES	YES	YES	YES
WRITE DEL DATA	x	LOAD	√	√	YES	YES	YES	YES
READ DATA	SKIP	LOAD	√	x	YES	YES	YES	YES
READ DATA AND DEL DATA	XFER	LOAD	√	x	YES	YES	YES	YES
READ ID	x	LOAD	√	x	YES	NO	YES	YES
VERIFY DATA AND DEL DATA	XFER	LOAD	√	x	YES	YES	YES	YES
FORMAT TRACK	x	LOAD	√	√	YES	NO	YES	YES
SEEK	x	LOAD	y	x	YES	NO	YES	YES
READ DRIVE STATUS	x	–	x	x	NO	NO	NOTE 5	NO
SPECIFY	x	–	x	x	NO	NO	NO	NO
RESET	x	UNLOAD	x	x	NO	NO	NO	NO
R SP REGISTERS	x	–	x	x	NO	NO	NOTE 6	NO
W SP REGISTERS	x	–	x	x	NO	NO	NO	NO

Note: 1. "x" → DON'T CARE 2. "√" → check 3. "–" → No change 4. "y" → Check at end of operation 5. See "READ DRIVE STATUS" command. 6. See "READ SPECIAL REGISTER" command.

Table 1. Execution Phase Basic Characteristics

Explanation of the execution phase characteristics table.

1. Deleted Data Processing

   If deleted data is encountered during an operation that is marked skip in the table, the deleted data record is not transferred into memory, but the record is counted. For example, if the command and parameters specify a read of five records and one of the records was written with a deleted data mark, four records are transferred to memory. The deleted data flag is set in the result byte. However, if the operation is marked transfer, all data is transferred to memory regardless of the type of data mark.

2. Head

   The Head column in the table specifies whether the Read/Write head will be loaded or not. If the table specifies load, the head is loaded after it is positioned over the track. The head loaded by a command remains loaded until the user specified number of index pulses have occurred.

3. Ready

   The Ready column indicates if the ready line (Ready 1, Ready 0) associated with the selected drive is checked. A not ready state is latched by the 8271 until the user executes a read status command.

4. Write Protect

   The operations that are marked check Write Protect are immediately aborted if Write Protect line is active at the beginning of an operation.

5. Seek

   Many of the 8271 commands cause a seek to the desired track. A current track register is maintained for each drive or surface.

6. Seek Check

   Operations that perform Seek Check verify that selected data in the ID field is correct before the 8271 accesses the data field.

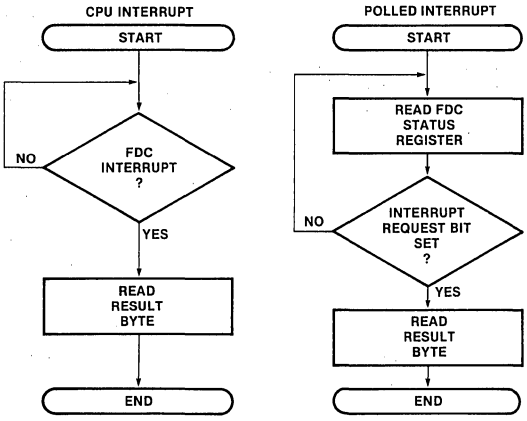


Figure 17. Getting the Result

The Result Phase

During the Result Phase, the FDC notifies the CPU of the outcome of the command execution. This phase may be initiated by:

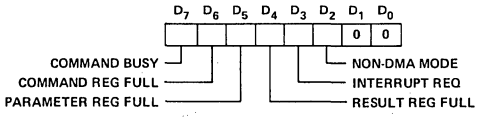
1. The successful completion of an operation.
2. An error detected during an operation.

PROGRAMMING

A_1	A_0	CS RD	CS WR
0	0	Status Reg	Command Reg
0	1	Result Reg	Parameter Reg
1	0	—	Reset Reg
1	1	—	—

STATUS REGISTER

FDC Status



Bit 7: Command Busy

The command busy bit is set on writing to the command register. Whenever the FDC is busy processing a command, the command busy bit is set to a one. This bit is set to zero after the command is completed.

Bit 6: Command Full

The command full bit is set on writing to the command buffer and cleared when the FDC begins processing the command.

Bit 5: Parameter Full

This bit indicates the state of the parameter buffer. This bit is set when a parameter is written to the FDC and reset after the FDC has accepted the parameter.

Bit 4: Result Full

This bit indicates the state of the result buffer. It is valid only after Command Busy bit is low. This bit is set when the FDC finishes a command and is reset after the result byte is read by the CPU. The data in the result buffer is valid only after the FDC has completed a command. Reading the result buffer while a command is in progress yields no useful information.

Bit 3: Interrupt Request

This bit reflects the state of the FDC INT pin. It is set when FDC requests attention as a result of the completion of an operation or failure to complete an intended operation. This bit is cleared by reading the result register.

Bit 2: Non-DMA Data Request

When the FDC is utilized without a DMA controller, this bit is used to indicate FDC data requests. Note that in the non-DMA mode, an interrupt is generated (interrupt request bit is set) with each data byte written to or read from the diskette.

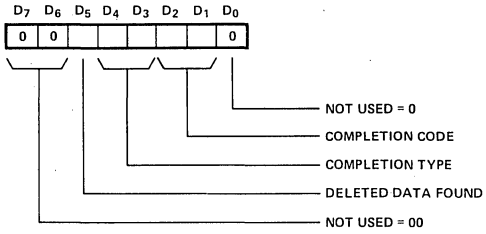
Bits 1 and 0:

Not used (zero returned).

After reading the Status Register, the CPU then reads the Result Register for more information.

THE RESULT REGISTER

This byte format facilitates the use of an address table to look up error routines and messages. The standard result byte format is:



Bits 7 and 6:

Not used (zero returned).

Bit 5:

Deleted Data Found: This bit is set when deleted data is encountered during a transaction.

Bits 4 and 3: Completion Type

The completion type field provides general information regarding the outcome of an operation.

The completion type field provides general information regarding the outcome of an operation.

Completion Type	Event
00	Good Completion — No Error
01	System Error — recoverable errors;
10	operator intervention probably required for recovery.
11	Command/Drive Error — either a program error or drive hardware failure.

Bits 2 and 1: Completion Code

The completion code field provides more detailed information about the completion type (See Table).

Completion Type	Completion Code	Event
00	00	Good Completion/ Scan Not Met
00	01	Scan Met Equal
00	10	Scan Met Not Equal
00	11	——
01	00	Clock Error
01	01	Late DMA
01	10	ID CRC Error
01	11	Data CRC Error
10	00	Drive Not Ready
10	01	Write Protect
10	10	Track 0 Not Found
10	11	Write Fault
11	00	Sector Not Found
11	01	——
11	10	——
11	11	——

It is important to note the hierarchical structure of the result byte. In very simple systems where only a GO-NO GO result is required, the user may simply branch on a zero result (a zero result is a good completion). The next level of complexity is at the completion type interface. The completion type supplies enough information so that the software may distinguish between fatal and non-fatal errors. If a completion type 01 occurs, ten retries should be performed before the error is considered unrecoverable.

The Completion Type/Completion Code interface supplies the greatest detail about each type of completion. This interface is used when detailed information about the transaction completion is required.

Bit 0:

Not used (zero returned).

Definition	Interpretation
Successful Completion/ Scan Not Met	The diskette operation specified was completed without error. If scan operation was specified, the pattern scanned was not found on the track addressed.
Scan Met Equal	The data pattern specified with the scan command was found on the track addressed with the specified comparison, and the equality was met.
Scan Met Not Equal	The data pattern specified with the scan command was found with the specified comparison on the track addressed, but the equality was not met.
Clock Error	During a diskette read operation, a clock bit was missing (dropped). Note that this function is disabled when reading any of the ID address marks (which contain missing clock pulses). If this error occurs, the operation is terminated immediately and an interrupt is generated.
Late DMA	During either a diskette read or write operation, the data channel did not respond within the allotted time interval to prevent data from being overwritten or lost. This error immediately terminates the operation and generates an interrupt.
ID Field CRC Error	The CRC word (two bytes) derived from the data read in an ID field did not match the CRC word written in the ID field when the track was formatted. If this error occurs, the associated diskette operation is prevented and no data is transferred.
Data Field CRC Error	During a diskette read operation, the CRC word derived from the data field read did not match the data field CRC word previously written. If this error occurs, the data read from the sector should be considered invalid.
Drive Not Ready	The drive addressed was not ready. This indication is caused by any of the following conditions: 1. Drive not powered up 2. Diskette not loaded 3. Non-existent drive addressed 4. Drive went not ready during an operation Note that this completion code is cleared only through an FDC read drive status command.
Write Protect	A diskette write operation was specified on a write protected diskette. The intended write operation is prevented and no data is written on the diskette.
Track 00 Not Found	During a seek to track 00 operation, the drive failed to provide a track 00 indication after being stepped 255 times.
Write Fault	This error is dependent on the drive supported and indicates that the fault input to the FDC has been activated by the drive.
Sector Not Found	Either the sector addressed could not be found within one complete revolution of the diskette (two index marks encountered) or the track address specified did not match the track address contained in the ID field. Note that when the track address specified and the track address read do not match, the FDC automatically increments its track address register (stepping the drive to the next track) and again compares the track addresses. If the track addresses still do not match, the track address register is incremented a second time and another comparison is made before the sector not found completion code is set.

Table 2. Completion Code Interpretation

INITIALIZATION

Reset Command

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PAR:	1	0	0	0	0	0	0	0	0	1
PAR:	1	0	0	0	0	0	0	0	0	0

Function: The Reset command emulates the action of the reset pin. It is issued by outputting a one followed by a zero to the Reset register.

1. The drive control signals are forced low.
2. An in-progress command is aborted.
3. The FDC status register flags are cleared.
4. The FDC enters an idle state until the next command is issued.

Reset must be active for 10 or more clock cycles.

SPECIFY COMMAND

Many of the interface characteristics of the FDC are specified by the systems software. Prior to initiating any drive operation command, the software must execute the three specify commands. There are two types of specify commands selectable by the first parameter issued.

First Parameter	Specify Type
$0D_H$	Initialization
10_H	Load bad Tracks Surface '0'
18_H	Load bad Tracks Surface '1'

The Specify command is used prior to performing any diskette operation (including formatting of a diskette) to define the drive's inherent operating characteristics and also is used following a formatting operation or installation of another diskette to define the locations of bad tracks. Since the Specify command only loads internal registers within the 8271 and does not involve an actual diskette operation, command processing is limited to only Command Phase. Note that once the operating characteristics and bad tracks have been specified for a given drive and diskette, redefining these values need only be done if a diskette with unique bad tracks is to be used or if the system is powered down.

Initialization:

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	0	1	1	0	1	0	1
PAR:	0	1	0	0	0	0	1	1	0	1
PAR:	0	1	STEP RATE*							
PAR:	0	1	HEAD SETTLING TIME*							
PAR:	0	1	INDEX CNT BEFORE HEAD UNLOAD*				HEAD LOAD TIME*			

*Note: Mini-floppy parameters are doubled.

Parameter 0 — $0D_H$ = Select Specify Initialization.
Parameter 1 — D_7-D_0 = Step Rate (0-255ms in 1ms steps).
Parameter 2 — D_7-D_0 = Head Settling Time (0-255ms in 1 ms steps). {0 – 510ms in 2ms steps} () = standard, {} = mini
Parameter 3 — D_7-D_4 = Index Count — Specifies the number of Revolutions (0-14) which are to occur before the FDC automatically unloads the R/W head. If 15 is specified, the head remains loaded.
D_3-D_0 = Head Load Time (0-60ms in steps of 4ms). {0 – 120ms in 8ms steps} () = standard, {} = mini

Load Bad Tracks

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	0	1	1	0	1	0	0
PAR:	0	1	0	0	0	1	1/0	0	0	0
PAR:	0	1	BAD TRACK NO. 1							
PAR:	0	1	BAD TRACK NO. 2							
PAR:	0	1	CURRENT TRACK							

Parameter 0: 10_H = Load Surface zero bad tracks
18_H = Load Surface one bad track

Parameter 1:
Bad track address number 1 (Physical Address).

It is recommended to program both bad tracks and current track to FF_H during initialization.

SEEK COMMAND

The seek command moves the head to the specified track without loading the head or verifying the track.

The seek operation uses the specified bad tracks to compute the physical track address. This feature insures that the seek operation positions the head over the correct track.

When a seek to track zero is specified, the FDC steps the head until the track 00 signal is detected.

If the track 00 signal is not detected within $(FF)_H$ steps, a track 0 not found error status is returned.

A seek to track zero is used to position the read/write head when the current head position is unknown (such as after a power up).

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	SEL 1	SEL 0	1	0	1	0	0	1
PAR:	0	1	TRACK ADDRESS 0-255							

Seek operations are not verified. A subsequent read or write operation must be performed to determine if the correct track is located.

READ DRIVE STATUS COMMAND

This command is used to interrogate the drive status. Upon completion the result register will hold the final drive status.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	SEL 1	SEL 0	1	0	1	1	0	0

RESULT: EACH BIT INDICATES CURRENT STATE OF INPUT PINS.

A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	—	RDY 1*	WR FAULT	INDEX	WR PROT	RDY 0*	TRACK 0	CNT/OPI

IF A DRIVE NOT READY RESULT IS RETURNED, THE READ STATUS MUST BE ISSUED TO CLEAR THE CONDITION.

*Note the two ready bits are zero latching. Therefore, to clear the drive not ready condition, assuming the drive is ready, and to detect it via software, one must issue this command twice.

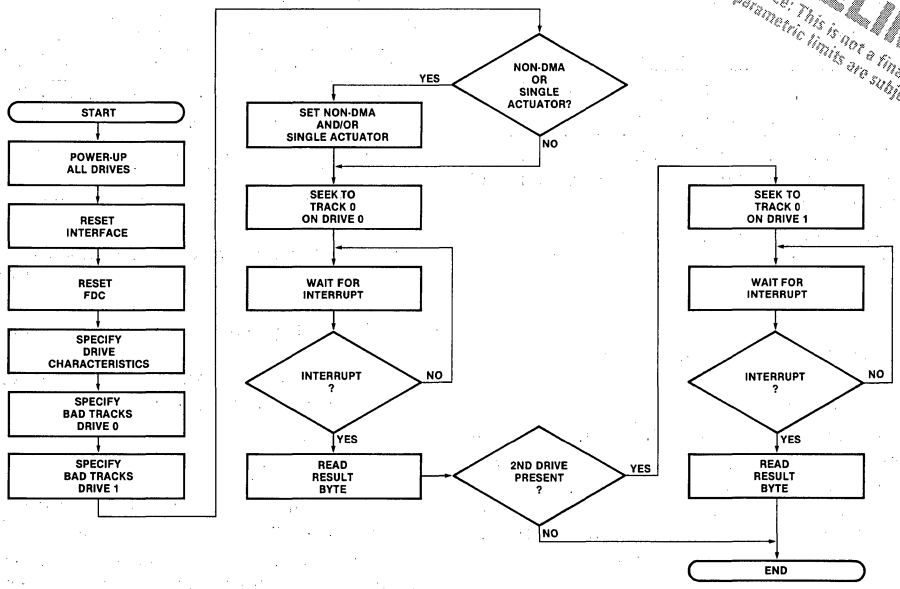


Figure 18. Initialization of the 8271 by the User

Read/Write Special Registers

This command is used to access special registers within the 8271.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	SEL 1	SEL 0	COMMAND OPCODE					
PAR:	0	1	REGISTER ADDRESS							

Command code:

$3D_H$ Read Special Register
$3A_H$ Write Special Register

For both commands, the first parameter is the register address; for Write commands a second parameter specifies data to be written. Only the Read Special Register command supplies a result.

Description	Register Address in Hex	Comment
Scan Sector Number	06	See Scan Description
Scan MSB of Count	14	See Scan Description
Scan LSB of Count	13	See Scan Description
Surface 0 Current Track	12	
Surface 1 Current Track	1A	
Mode Register	17	See Mode Register Description
Drive Control Output Port	23	See Drive Output Port Description
Drive Control Input Port	22	See Drive Input Port Description
Surface 0 Bad Track 1	10	
Surface 0 Bad Track 2	11	
Surface 1 Bad Track 1	18	
Surface 1 Bad Track 2	19	

Table 3. Special Registers

Mode Register Write Parameter Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	0	0	0		

= 0 DMA MODE, = 1 NON DMA

= 0 DOUBLE, = 1 SINGLE ACTUATOR

Bits 6 & 7

Must be one.

Bits 5-2

(Not used). Must be set to zero.

***Bit 1**

Double/Single Actuator: Selects single or double actuator mode. If the single actuator mode is selected, the FDC assumes that the physical track location of both disks is always the same. This mode facilitates control of a drive which has a single actuator mechanism to move two heads.

***Bit 0**

Data Transfer Mode: This bit selects the data transfer mode. If this bit is a zero, the FDC operates in the DMA mode (DMA Request/ACK). If this bit is a one, the FDC operates in non-DMA mode. When the FDC is operating in DMA mode, interrupts are generated at the completion of commands. If the non-DMA mode is selected, the FDC generates an interrupt for every data byte transferred.

*Bits 0 and 1 are initialized to zero.

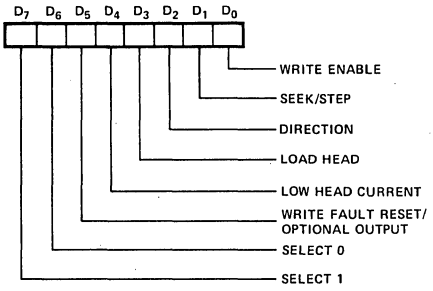
Non-DMA Transfers in DMA Mode

If the user desires, he may retain the use of interrupts generated upon command completions. This mode is accomplished by selecting the DMA capability, but using the DMA REQ/ACK pins as effective INT and CS signals, respectively.

Drive Control Input Port

Reading this port will give the CPU exactly the data that the FDC sees at the corresponding pins. Reading this port will update the drive not ready status, but will not clear the status. (See Read Drive Status Command for Bit locations.)

Drive Control Output Port Format



Each of these signals correspond to the chip pin of the same name. On standard-sized drives with write fault detection logic, bit 5 is set to generate the write fault reset signal. This signal is used to clear a write fault indication within the drive. On mini-sized drives, this bit can be used to turn on or off the drive motor prior to initiating a drive operation. A time delay after turn on may be necessary for the drive to come up to speed. The register must be read prior to writing the register in order to save the states of the remaining bits. When the register is subsequently written to modify bit 5, the remaining bits must be restored to their previous states.

IBM DISKETTE GENERAL FORMAT INFORMATION

The IBM Flexible Diskette used for data storage and retrieval is organized into concentric circular paths or TRACKS. There are 77 tracks on either one or both sides (surfaces) of the diskette. On double-sided diskettes, the corresponding top and bottom tracks are referred to as a CYLINDER. Each track is further divided into fixed length sections or SECTORS. The number of sectors per track — 26, 15 or 8 — is determined when a track is formatted and is dependent on the sector length — 128, 256 or 512 bytes respectively — specified.

All tracks on the diskette are referenced to a physical index mark (a small hole in the diskette). Each time the hole passes a photodetector cell (one revolution of the diskette), an Index pulse is generated to indicate the logical beginning of a track. This index pulse is used to initiate a track formatting operation.

Track Format

Each Diskette Surface is divided into 77 tracks with each track divided into fixed length sectors. A sector can hold a whole record or a part of a record. If the record is shorter than the sector length, the unused bytes are filled with binary zeros. If a record is longer than the sector length, the record is written over as many sectors as its length requires. The sector size that provides the most efficient use of diskette space can be chosen depending upon the record length required.

Tracks are numbered from 00 (outer-most) to 76 (inner-most) and are used as follows:

TRACK 00 reserved as System Label Track
TRACKS 01 through 74 used for data
TRACKS 75 and 76 used as alternates.

Each sector consists of an ID field (which holds a unique address for the sector) and a data field.

The ID field is seven bytes long and is written for each sector when the track is formatted. Each ID field consists of an ID field Address Mark, a Cylinder Number byte which identifies the track number, a Head Number byte which specifies the head used (top or bottom) to access the sector, a Record Number byte identifying the sector number (1 through 26 for 128 byte sectors), an N-byte specifying the byte length of the sector and two CRC (Cyclic Redundancy Check) bytes.

The Gaps separating the index mark and the ID and data fields are written on a track when it is formatted. These gaps provide both an interval for switching the drive electronics from reading or writing and compensation for rotational speed and other diskette-to-diskette and drive-to-drive manufacturing tolerances to ensure that data written on a diskette by one system can be read by another (diskette interchangeability).

IBM Format Implementation Summary

Track Format

The disk has 77 tracks, numbered physically from 00 to 76, with track 00 being the outermost track. There are logically 75 data tracks and two alternate tracks. Any two tracks may be initialized as bad tracks. The data tracks are numbered logically in sequence from 00 to 74, skipping over bad tracks (alternate tracks replace bad tracks). Note: In IBM format track 00 cannot be a bad track.

Sector Format

Each track is divided into 26, 15, or 8 sectors of 128, 256, or 512 bytes length respectively. The first sector is numbered 01, and is physically the first sector after the physical index mark. The logical sequence of the remaining sectors may be nonsequential physically. The location of these is determined at initialization by CPU software.

Each sector consists of an ID field and a data field. All fields are separated by gaps. The beginning of each field is indicated by 6 bytes of $(00)_H$ followed by a one byte address mark.

Address Marks

Address Marks are unique bit patterns one byte in length which are used to identify the beginning of ID and Data fields. Address Mark bytes are unique from all other data

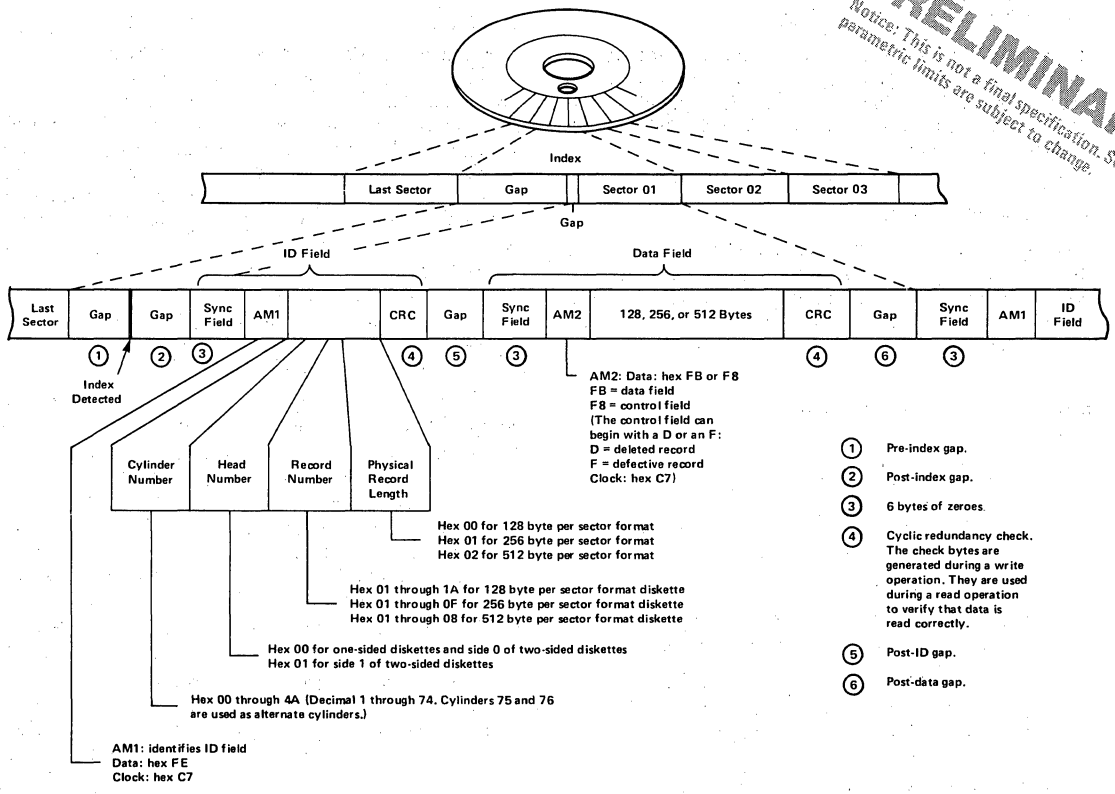


Figure 19. Track Format

bytes in that certain bit cells do not contain a clock bit (all other data bytes have clock bits in every bit cell.) There are four different types of Address Marks used. Each of these is used to identify different types of fields.

Index Address Mark

The Index Address Mark is located at the beginning of each track and is a fixed number of bytes in front of the first record.

ID Address Mark

The ID Address Mark byte is located at the beginning of each ID field on the diskette.

Data Address Mark

The Data Address Mark byte is located at the beginning of each non-deleted Data Field on the diskette.

Deleted Data Address Mark

The Deleted Data Address Mark byte is located at the beginning of each deleted Data Field on the diskette.

Address Mark Summary	Clock Pattern	Data Pattern
Index Address Mark	D7	FC
ID Address Mark	C7	FE
Data Address Mark	C7	FB
Deleted Data Address Mark	C7	F8
Bad Track ID Address Mark	C7	FE

ID Field

MARK	C	H	R	N	CRC	CRC

C = Cylinder (Track) Address, 00-74
H = Head Address
R = Record (Sector) Address, 01-26
N = Record (Sector) Length, 00-02
Note: Sector Length = 128×2^N bytes
CRC = 16 Bit CRC Character (See Below)

Data Field

MARK	DATA	CRC	CRC

Data is 128, 256, or 512 bytes long.

Note: All marks, data, ID characters and CRC characters are recorded and read most significant bit first.

CRC Character

The 16-bit CRC character is generated using the generator polynominal $X^{16} + X^{12} + X^5 + 1$, normally initialized to $(FF)_H$. It is generated from all characters (except the CRC in the ID or data field), including the data (not the clocks) in the address mark. It is recorded and read most significant bit first.

Data Format

Data is written (general case) in the following manner:

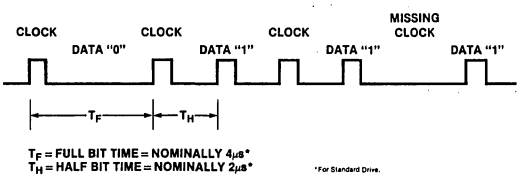


Figure 20. Data Format

References

"The IBM Diskette for Standard Data Interchange," IBM Document GA21-9182-0. "System 32," Chapter 8, IBM Document GA21-9176-0.

Bad Track Format

The Bad Track Format is the same as the good track format except that the bad track ID field is initialized as follows:

$C = H = R = N = (FF)_H$

When formatting, bad track registers should be set to FF_H for the drive during the formatting, thus specifying no bad tracks. Thus, all tracks are left available for formatting.

The track following the bad track(s) should be one higher in number than track before the bad track(s).

Upon completion of the format the bad tracks should be set up using the write special register command. The 8271 will then generate an extra step pulse to cross the bad track, locating a new track that now happens to be an extra track out.

Format Track

Format Command

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	SEL 1	SEL 0	1	0	0	0	1	1
PAR:	0	1	TRACK ADDRESS							
PAR:	0	1	GAP 3 SIZE MINUS 6							
PAR:	0	1	RECORD LENGTH			NO. OF SECTORS/TRACK				
PAR:	0	1	GAP 5 SIZE MINUS 6							
PAR:	0	1	GAP 1 SIZE MINUS 6							

The format command can be used to initialize a disk track compatible with the IBM 3740 format. A Shugart "IBM Type" mini-floppy format may also be generated.

The Format command can be used to initialize a diskette, one track at a time. When format command is used, the program must supply ID fields for each sector on the track. During command execution, the supplied ID fields (track head sector addresses and the sector length) are written sequentially on the diskette. The ID address marks originate from the 8271 and are written automatically as the first byte of each ID field. The CRC character is written in the last two bytes of the ID field and is derived from the data written in the first five bytes. During the formatting operation, the data field of each sector is filled with data pattern $(E5)_H$. The CRC, derived from the data pattern is also appended to the last byte.

1. The parameter 2 ($D_7 - D_5$) of the Format command specify record length, the bits are coded the same way as in the Read Data commands.
2. The programmable gap sizes (gap 3, gap 5, and gap 1) must be programmed such that the 6 bytes of zero (sync) are subtracted from the intended gap size i.e., if gap 1 is intended to be 16 bytes long, programmed length must be 16 – 6 = 10 bytes (of FF_H's).

Mini-Floppy Disk Format

The mini-floppy disk format differs from the standard disk format in the following ways:

1. Gap 5 and the Index Address mark have been eliminated.
2. There are fewer sectors/tracks.

GAPS

The following is the gap size and description summary:

Gap 1	Programmable
Gap 2	17 Bytes
Gap 3	Programmable
Gap 4	Variable
Gap 5	Programmable

The last six bytes of gaps 1,2,3 and 5 are $(00)_H$, all other bytes in the gaps are $(FF)_H$. The Gap 1,3 and 5 count specified by the user are the number of bytes of $(FF)_H$. Gap 4 is written until the leading edge of the index pulse. If a Gap 5 size of zero is specified, the Index Mark is not written.

Gap 1: N bytes FF's 6 bytes 0's for sync	This gap separates the index address mark of the index pulse from the first ID mark. It is used to protect the first ID field from a write on the last physical sector of the current track.
Gap 2: 11 bytes FF's 6 bytes 0's for sync	This gap separates the ID field from the data mark and field such that during a write only the data field will be changed even if the write gate turns on early, due to drive speed changes.
Gap 3: N bytes FF's 6 bytes 0's for sync	This gap separates a data area from the next ID field. It is used so that during drive speed changes the next ID mark will not be overwritten, thus causing loss of data.
Gap 4: FF's only	This gap fills out the rest of the disk and is used for slack during formatting. During drive speed variations this gap will shrink or grow if the disk is re-formatted.
Gap 5: N bytes FF's 6 bytes 0's for sync	This gap separates the last sector from the Index Address mark and is used to assure that the index address mark is not destroyed by writing on the last physical data sector on the track.

The number of FF bytes is programmable for gaps 1, 3 and 5.

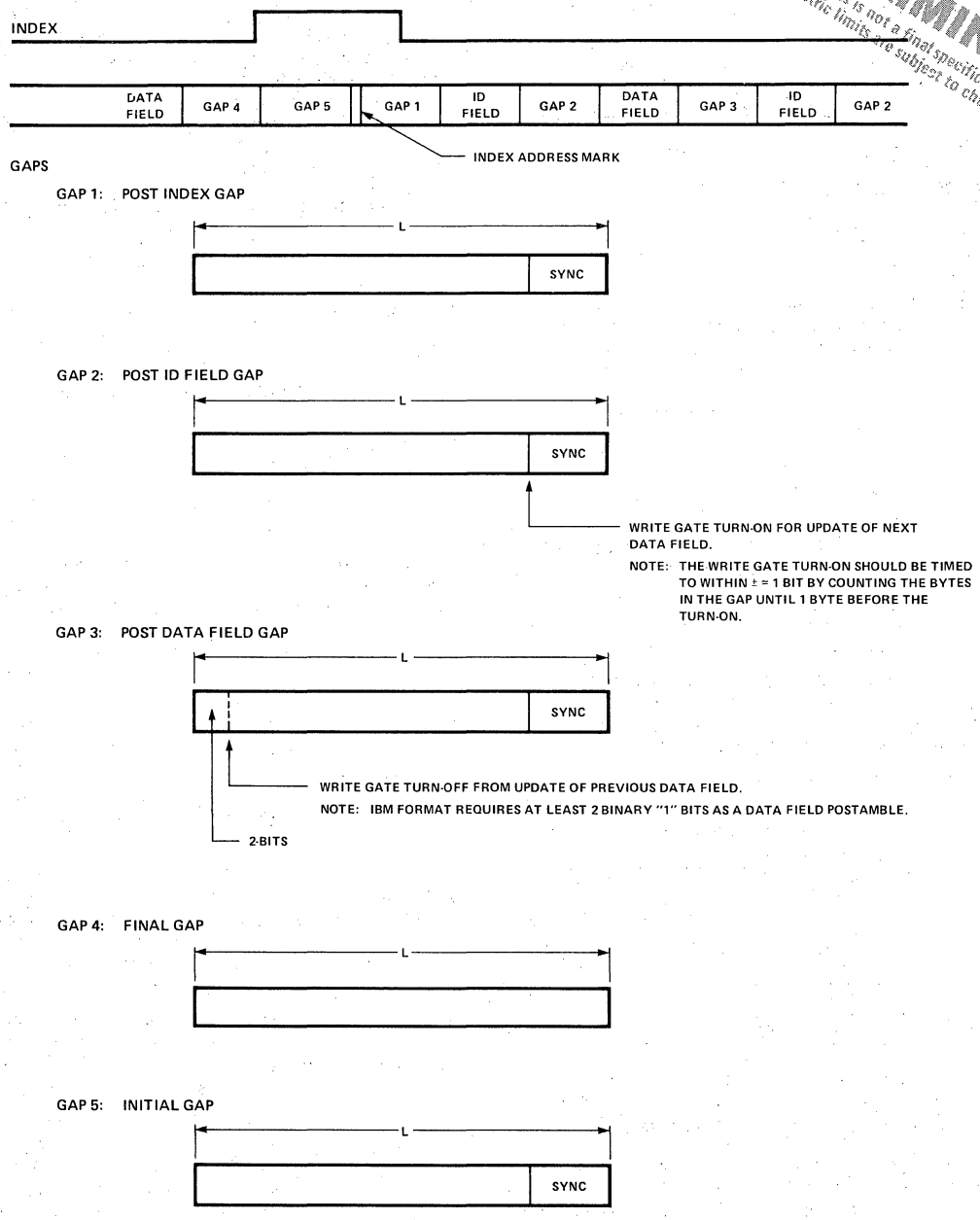
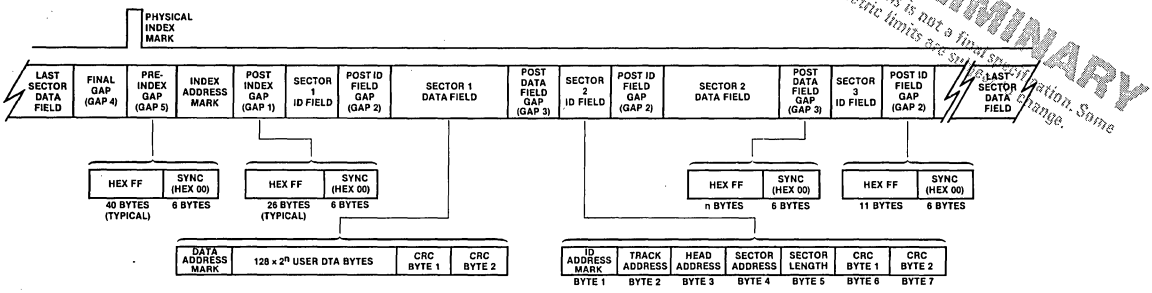


Figure 21. Track Format

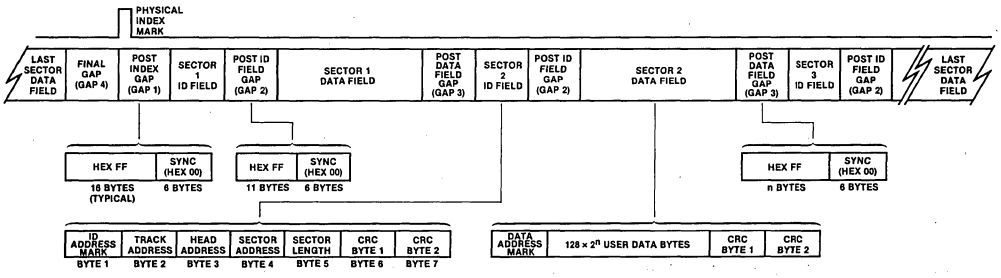


NUMBER OF SECTORS	NUMBER OF BYTES										
	GAP 1		ID FIELD	GAP 2		DATA FIELD	GAP 3		GAP 4	GAP 5	
	*ONES	SYNC		ONES	SYNC		*ONES	SYNC		*ONES	SYNC
26	26	6	7	11	6	131	27	6	275	40	6
15	26	6	7	11	6	259	48	6	129	40	6
8	26	6	7	11	6	515	90	6	146	40	6
4	26	6	7	11	6	1027	224	6	236	40	6
2	26	6	7	11	6	2051	255	6	719	40	6
1	26	6	7	11	6	4099	0	0	1007	40	6

*Program Specified

5208 Bytes Per Track

Figure 22. Standard Diskette Track Format



NUMBER OF SECTORS	NUMBER OF BYTES								
	GAP 1		ID FIELD	GAP 2		DATA FIELD	GAP 3		GAP 4
	*ONES	SYNC		ONES	SYNC		*ONES	SYNC	
18	16	6	7	11	6	131	11	6	24
10	16	6	7	11	6	259	21	6	30
5	16	6	7	11	6	515	74	6	88
2	16	6	7	11	6	1027	255	6	740
1	16	6	7	11	6	2051	0	0	1028

*Program Specified

3125 Bytes Per Track

Figure 23. Mini-Diskette Track Format

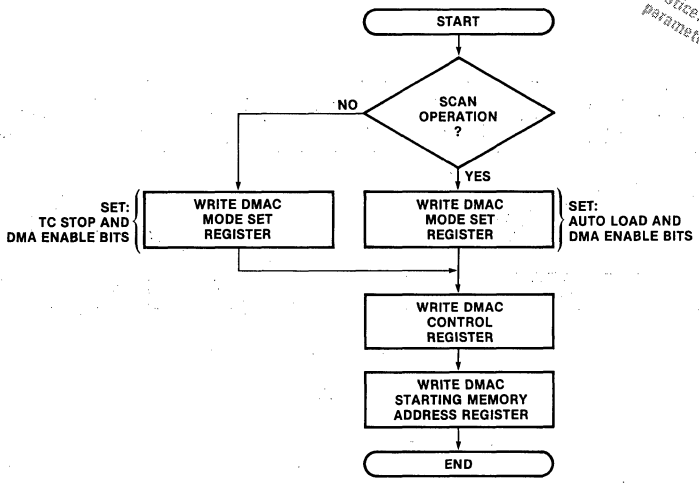


Figure 23. User DMA Channel Initialization Flowchart

Read ID Command

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	SEL 1	SEL 0	0	1	1	0	1	1
PAR:	0	1	TRACK ADDRESS							
PAR:	0	1	0	0	0	0	0	0	0	0
PAR:	0	1	NUMBER OF ID FIELDS							

The Read ID command transfers the specified number of ID fields into memory (beginning with the first ID field after Index). The CRC character is checked but not transferred.

These fields are entered into memory in the order in which they are physically located on the disk, with the first field being the one starting at the index pulse.

Data Processing Commands

All the routine Read/Write commands examine specific drive status lines before beginning execution, perform an implicit seek to the track address and load the drive's read/write head. Regardless of the type of command (i.e., read, write or verify), the 8271 first reads the ID field(s) to verify that the correct track has been located (see sector not found completion code) and also to locate the addressed sector. When a transfer is complete (or cannot be completed), the 8271 sets the interrupt request bit in the status register and provides an indication of the outcome of the operation in the result register.

If a CRC error is detected during a multisector transfer, processing is terminated with the sector in error. The address of the failing sector number can be determined by examining the Scan Sector Number register using the Read Special Register command.

Full power of the multisector read/write commands can be realized by doing DMA transfer using Intel® 8257 DMA Controller, For example, in a 128 byte per sector multisector write command, the entire data block (containing 128 bytes times the number of sectors) can be located in a disk memory buffer. Upon completion of the command phase, the 8271 begins execution by accessing the desired track, verifying the ID field, and locating the data field of the first record to be written. The 8271 then DMA-accesses the first sector and starts counting and writing one byte at a time until all 128 bytes are written. It then locates the data field of the next sector and repeats the procedure until all the specified sectors have been written. Upon completion of the execution phase the 8271 enters into the result phase and interrupts the CPU for availability of status and completion results. Note that all read/write commands, single or multisector are executed without CPU intervention.

Note, execution of multi-sector operations are faster if the sectors are *not* interleaved.

128 Byte Single Record Format

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	SEL 1	SEL 0	COMMAND OPCODE					
PAR:	0	1	TRACK ADDR 0-255							
PAR:	0	1	SECTOR 0-255							

Commands	Opcode
READ DATA	12
READ DATA AND DELETED DATA	16
WRITE DATA	0A
WRITE DELETED DATA	0E
VERIFY DATA AND DELETED DATA	1E

Variable Length/Multi-Record Format

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	SEL 1	SEL 0	COMMAND OPCODE					
PAR:	0	1	TRACK ADDR 0-255							
PAR:	0	1	SECTOR 0-255							
PAR:	0	1	LENGTH			NO. OF SECTORS				

D_7-D_5 of Parameter 2 determine the length of the disk record.

0 0 0	128 Bytes
0 0 1	256 Bytes
0 1 0	512 Bytes
0 1 1	1024 Bytes
1 0 0	2048 Bytes
1 0 1	4096 Bytes
1 1 0	8192 Bytes
1 1 1	16,384 Bytes

Commands	Opcode
READ DATA	13
READ DATA AND DELETED DATA	17
WRITE DATA	0B
WRITE DELETED DATA	0F
VERIFY DATA AND DELETED DATA	1F
SCAN DATA	00
SCAN DATA AND DELETED DATA	04

Read Commands

Read Data, Read Data and Deleted Data.

Function

The read command transfers data from a specified disk record or group of records to memory. The operation of this command is outlined in execution phase table.

Write Commands

Write Data, Write Deleted Data.

Function

The write command transfers data from memory to a specified disk record or group of records.

Verify Command

Verify Data and Deleted Data.

Function

The verify command is identical to the read data and deleted data command except that the data is not transferred to memory. This command is used to check that a record or a group of records has been written correctly by verifying the CRC character.

Scan Commands

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	SEL 1	SEL 0	0	0	0	S.DATA S.DELD	0	0
PAR:	0	1	TRACK ADDR 0-255							
PAR:	0	1	SECTOR 0-255							
PAR:	0	1	LENGTH			NO. OF SECTORS				
PAR:	0	1	SCAN TYPE		STEP SIZE					
PAR:	0	1	FIELD LENGTH (KEY)							

Command $D_2 = 0$ Scan Data
$D_2 = 1$ Scan Data and Deleted Data

Scan Commands, Scan Data and Scan Data and Deleted Data, are used to search a specific data pattern or "key" from memory. The 8271 FDC operation during a scan is unique in that data is read from memory and from the diskette simultaneously.

During the scan operation, the key is compared repetitively (using the 8257 DMA Controller in auto load mode) with the data read from the diskette (e.g., an eight byte key would be compared with the first eight bytes (1-8) read from the diskette, the second eight bytes (9-16), the third eight bytes (17-24), etc.). The scan operation is concluded when the key is located or when the specified number of sectors have been searched without locating the key. When concluded, the 8271 FDC requests an interrupt. The program must then read the result register to determine if the scan was successful (if the key was located). If successful, several of the FDC's special registers can be examined (read special registers command) to determine more specific information relating to the scan (i.e., the sector number in which the key was located, and the number of bytes within the sector that were not compared when the key was located).

The 8271 does not do a sliding scan, it does a fixed block linear search. This means the key in memory is compared to an equal length block in a sector; when these blocks meet the scan conditions the scan will stop. Otherwise, the scan continues until all the sectors specified have been searched.

The following factors regarding key length must be considered when establishing a key in memory.

1. When searching multiple sectors, the length of the key must be evenly divisible into the sector length to prevent the key from being split at subsequent sector boundaries. Since the character FF_H is not compared, the key in memory can be padded to the required length using this character. For example, if the actual pattern compared on the diskette is twelve characters in length, the field length should be sixteen and four bytes of FF_H

would be appended to the key. Consequently, the last block of sixteen bytes compared within the first sector would end at the sector boundary and the first byte of the next sector would be compared with the first byte of the key. Splitting data over sector boundarys will not work properly since the FDC expects the start of key at each sector boundary.

2. Since the first byte of the key is compared with the first byte of the sector, when the pattern does not begin with the first byte of the sector, the key must be offset using the character FF_{16}. For example, if the first byte of a nine byte pattern begins on the fifth byte of the sector, four bytes of FF_{16} are prefixed to the key (and three bytes of FF_{16} are appended to the key to meet the length requirement) so that the first actual comparison begins on the fifth byte.

The Scan Commands require five parameters:

Parameter 0, Track Address

Specifies the track number containing the sectors to be scanned. Legal values range from 00_H to $4C_H$ (0 to 76) for a standard diskette and from 00_H to 22_H (0 to 34) for a mini-sized diskette.

Parameter 1, Sector Address

Specifies the first sector to be scanned. The number of sectors scanned is specified in parameter 2, and the order in which sectors are scanned is specified in parameter 3.

Parameter 2, Sector Length/Number of Sectors

The sector length field (bits 7-5) specifies the number of data bytes allocated to each sector (see parameter 2, routine read and write commands for field interpretation). The number of sectors field (bits 4-0) specifies the number of sectors to be scanned. The number specified ranges from one sector to the physical number of sectors on the track.

Parameter 3

D_7-D_6: Indicate scan type

00-EQ Scan for each character within the field length (key) equal to the corresponding character within the disk sector. The scan stops after the first equal condition is met.

01-GEQ Scan for each character within the disk sector greater than or equal to the corresponding character within the field length (key). The scan stops after the first greater than or equal condition is met.

10-LEQ Scan for each character within the disk sector less than or equal to the corresponding character within the field length (key). The scan stops after the first less than or equal condition is met.

D_5-D_0: Step Size: The Step Size field specifies the offset to the next sector in a multisector scan. In this case, the next sector address is generated by adding the Step Size to the current sector address.

Parameter 4, Field Length

Specifies the number of bytes to be compared (length of key). While the range of legal values is from 1 to 255, the field length specified should be evenly divisible into the sector length to prevent the key from being split at sector boundaries, if the multisector scan commands are used.

Scan Command Results

More detailed information about the completion of Scan Commands may be obtained by executing Read Special Register commands.

Read Special Register

Parameter (Hex)	Results
06	The sector number of the sector in which the specified scan data pattern was located.
14	MSB Count — The number of 128 byte blocks remaining to be compared in the current sector when the scan data pattern was located. This register is decremented with each 128 byte block read.
13	LSB Count — The number of bytes remaining to be compared in the current sector when the scan data pattern is located. This register is initialized to 128 and is decremented with each byte compared.

Upon a scan met condition, the equation below can be used to determine the last byte in the located pattern.

Pointer = sector length – ((Register 14H)*128 + (Register 13H))

8271 Scan Command Example

Assume there are only 2 records on track 0 with the following data:

Record 01: 01 02 03 04 05 06 07 08 000....00
Record 02: 01 02 AA 55 00 00 00 00 00

Command	Field [1] Length	Starting Sector #	# of Sectors	Key[2]	Completion Code[3]	Special Registers[4] R06	R14	R13	Comment
* SCAN EQ	2	1	1	01,02	SME	01	0	127D	Met in first field
SCAN EQ	2	1	1	02,03	SNM	X	X	X	Not met
SCAN EQ	2	1	1	FF[5],05	SNM	X	X	X	Not met with don't care
* SCAN EQ	2	1	1	FF[5],06	SME	01	0	123D	Met with don't care
* SCAN EQ	2	1	2	AA,55	SME	02	0	125D	Met in Record 02
* SCAN EQ	2	2	1	01,02	SME	02	0	127D	Starting sector ≠ 1
* SCAN EQ	4	1	1	05,06,07,08	SME	01	0	121D	Field, Key length = 4
* SCAN GEQ	4	1	1	05,06,07,08	SME	01	0	121D	GEQ-SME
* SCAN GEQ	4	1	1	05,04,07,08	SMNE	01	0	121D	GEQ-SMNE
* SCAN GEQ	4	1	2	00,03,AA,44 [6]	SNM	X	X	X	GEQ-SNM
* SCAN LEQ	4	1	1	01,03,FF,04	SMNE	01	0	125D	LEQ-SMNE
* SCAN LEQ	4	1	1	01,02,FF,04	SME	01	0	125D	LEQ-SME

NOTES:

1. Field Length — Each record is partitioned into a number of fields equal to the record size divided by the field length. Note that the record size should be evenly divisable by the field length to insure proper operation of multi record scan. Also, maximum field length = 256 bytes.

2. Key — The key is a string of bytes located in the user system memory. The key length should equal the field length. By programming the 8257 DMA Controller into the auto load mode, the key will be recursively read in by the chip (once per field).

3. Completion Code — Shows how Scan command was met or not met.
   SNM — SCAN Not Met — 0 0 (also Good Complete)
   SME — SCAN Met Equal — 0 1
   SMNE — SCAN Met Not Equal — 1 0

4. Special Registers
   R06 — This register contains the record number where the scan was met.
   R14 — This register contains the MSB count and is decremented every 128 characters.

Length (ℓ) (D7-D5 of PAR 2)	Record Size	R14 = $2^{\ell} - 1$ (Initialize at Beginning of Record)
000	128 Bytes	0
001	256 Bytes	1
010	512 Bytes	3
011	1024 Bytes	7
•	•	•
•	•	•
•	•	•

   R13 — This register contains a modulo 128 LSB count which is initialized to 128 at beginning of each record. This count is decremented after each character is compared except for the last character in a pattern match situation.

5. The OFF_H character in the key is treated as a don't care character position.

6. The Scan comparison is done on a byte by byte basis. That is, byte 1 of each field is compared to byte 1 of the key, byte 2 of each field is compared to byte 2 of the key, etc.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . – 65°C to + 150°C
Voltage on Any Pin with
Respect to Ground . . . . . – 0.5V to + 7V
Power Dissipation . . . . . 1 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = +5.0V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	$(V_{CC} + 0.5)$	V	
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 2.0$ mA for Data Bus Pins $I_{OL} = 1.7$ mA for All Other Pins
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -220\ \mu A$
I_{IL}	Input Load Current		± 10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OZ}	Off-State Output Current		± 10	μA	$V_{OUT} = V_{CC}$ to 0V
I_{CC}	V_{CC} Supply Current		180	mA	

CAPACITANCE

$T_A = 25°C$, $V_{CC} = GND = 0V$

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	$t_c = 1$ MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = +5.0V \pm 5\%$

Read Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{RD}$	0		ns	Note 2
t_{CA}	Select Hold from $\overline{RD}$	0		ns	Note 2
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	Data Delay from Address		250	ns	Note 2
t_{RD}	Data Delay from $\overline{RD}$		150	ns	$C_L = 150$ pF, Note 2
t_{DF}	Output Float Delay	20	100	ns	$C_L = 20$ pF for Minimum; 150 pF for Maximum
t_{DC}	DACK Setup to $\overline{RD}$	25		ns	
t_{CD}	DACK Hold from $\overline{RD}$	25		ns	
t_{KD}	Data Delay from DACK		250	ns	

Write Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{WR}$	0		ns	
t_{CA}	Select Hold from $\overline{WR}$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$	150		ns	
t_{WD}	Data Hold from $\overline{WR}$	0		ns	
t_{DC}	DACK Setup to $\overline{WR}$	25		ns	
t_{CD}	DACK Hold from $\overline{WR}$	25		ns	

DMA

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CQ}	Request Hold from $\overline{WR}$ or $\overline{RD}$ (for Non-Burst Mode)		150	ns	

Other Timing

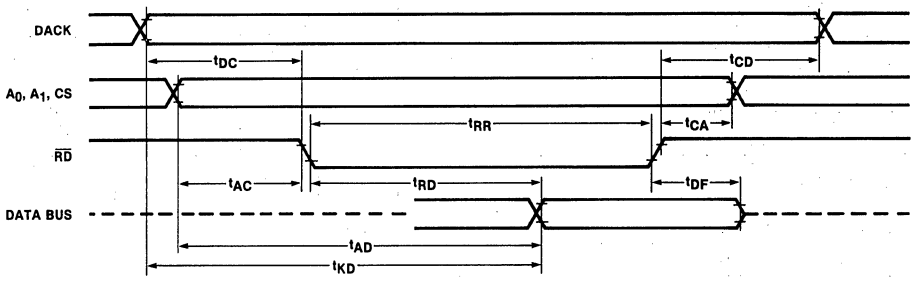
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{RSTW}	Reset Pulse Width	10		t_{CY}	
t_r	Input Signal Rise Time		20	ns	
t_f	Input Signal Fall Time		20	ns	
t_{RSTS}	Reset to First $\overline{IOWR}$	2		t_{CY}	
t_{CY}	Clock Period	250			Note 3
t_{CL}	Clock Low Period	110		ns	
t_{CH}	Clock High Period	122		ns	
t_{DS}	Data Window Setup to Unseparated Clock and Data	50		ns	
t_{DH}	Data Window Hold from Unseparated Clock and Data	0		ns	

NOTES:

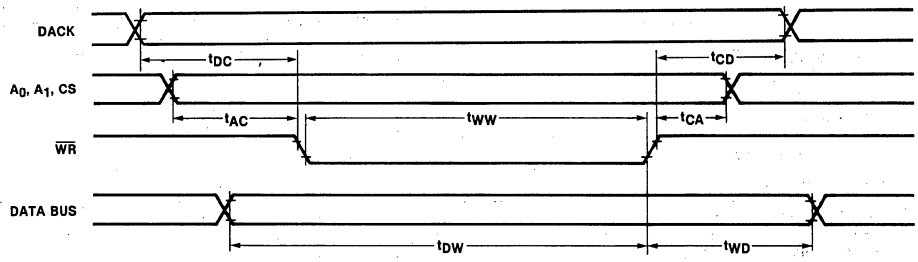
1. All timing measurements are made at the reference voltages unless otherwise specified: Input "1" at 2.0V, "0" at 0.8V
   Output "1" at 2.0V, "0" at 0.8V
2. t_{AD}, t_{RD}, t_{AC}, and t_{CA} are not concurrent specs.
3. Standard Floppy: $t_{CY} = 250$ ns $\pm 0.4\%$ Mini-Floppy: $t_{CY} = 500$ ns $\pm 0.4\%$

WAVEFORMS

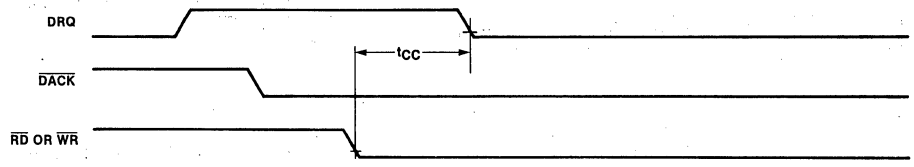
Read Waveforms

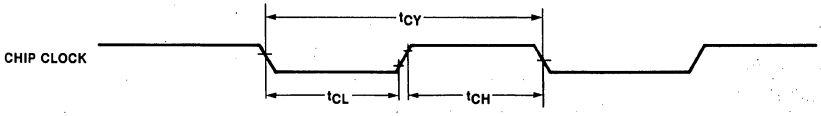


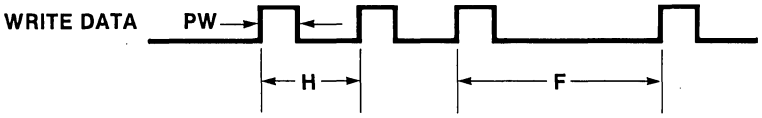
Write Waveforms



DMA Waveforms

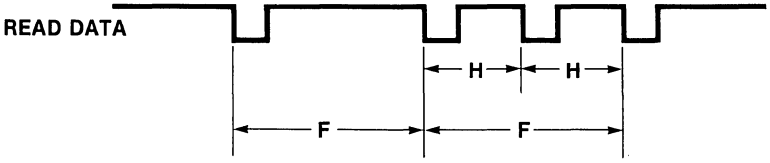






PULSE WIDTH PW = t_{CY} ± 30 ns	*t_{CY} = 250 ns ± 0.4%	**t_{CY} = 500 ns ± 0.4%
H (HALF BIT CELL) = 8 t_{CY}	250 ns ± 30 ns	500 ns ± 30 ns
F (FULL BIT CELL) = 16 t_{CY}	2.0 μs ± 8 ns	4.0 μs ± 16 ns
	4.0 μs ± 16 ns	8.0 μs ± 32 ns

Figure 24. Write Data



*t_{CY} = 250 ns **t_{CY} = 500 ns

F = 16 t_{CY} ± 8 t_{CY}
H = 8 t_{CY} ± 4 t_{CY}

Figure 25. Read Data

*STANDARD FLEXIBLE DISK DRIVE TIMING
**MINI-FLOPPY TIMING

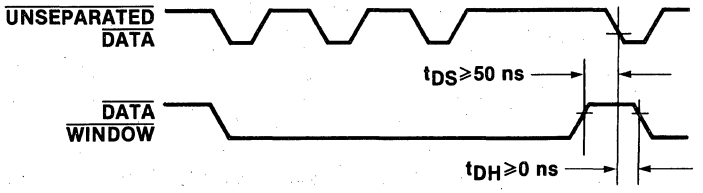
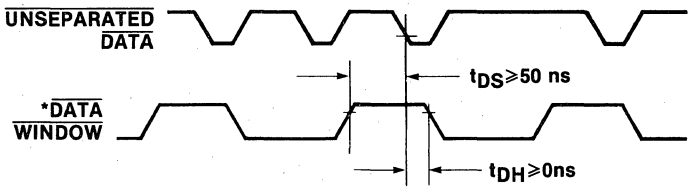


Figure 26. Single-Shot Data Separator



*DATA WINDOW MAY BE 180° OUT OF PHASE IN PLO DATA SEPARATION MODE.

Figure 27. PLO Data Separator

intel®

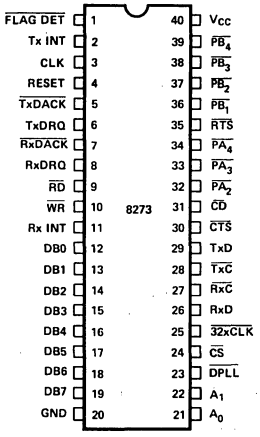
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8273
PROGRAMMABLE HDLC/SDLC PROTOCOL CONTROLLER

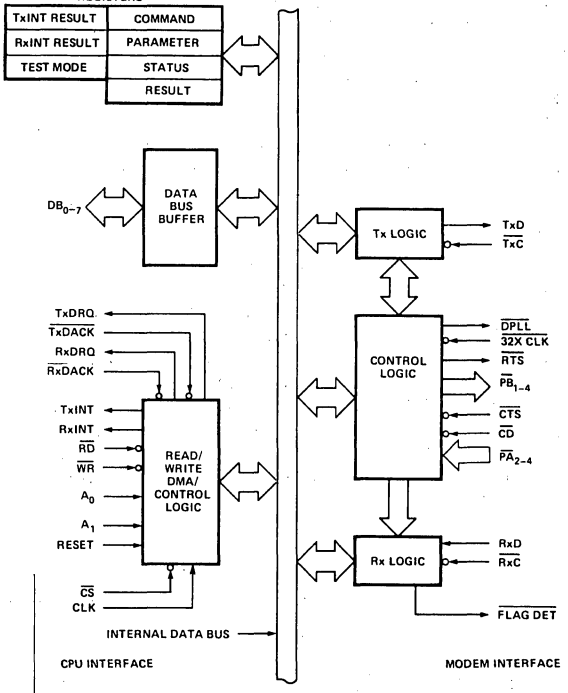
- **HDLC/SDLC Compatible**
- **Frame Level Commands**
- **Full Duplex, Half Duplex, or Loop SDLC Operation**
- **Up to 64K Baud Transfers**
- **Two User Programmable Modem Control Ports**
- **Automatic FCS (CRC) Generation and Checking**
- **Programmable NRZI Encode/Decode**
- **N-Bit Reception Capability**
- **Digital Phase Locked Loop Clock Recovery**
- **Minimum CPU Overhead**
- **Fully Compatible with 8080/8085 CPUs**
- **Single +5V Supply**
- **40-Pin Package**

The Intel® 8273 Programmable HDLC/SDLC Protocol Controller is a dedicated device designed to support the ISO/C-CITT's HDLC and IBM's SDLC communication line protocols. It is fully compatible with Intel's new high performance microcomputer systems such as the MCS-85™. A frame level command set is achieved by a unique microprogrammed dual processor chip architecture. The processing capability supported by the 8273 relieves the system CPU of the low level real-time tasks normally associated with controllers.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

DB0–DB7	DATA BUS (8 BITS)	$\overline{CS}$	CHIP SELECT
$\overline{FLAG\ DET}$	FLAG DETECT	$\overline{32xCLK}$	32 TIMES CLOCK
TxINT	TRANSMITTER INTERRUPT	Rx D	RECEIVER DATA
CLK	CLOCK INPUT	$\overline{Rx\ C}$	RECEIVER CLOCK
RESET	RESET	$\overline{Tx\ C}$	TRANSMITTER CLOCK
$\overline{Tx\ DACK}$	TRANSMITTER DMA ACKNOWLEDGE	Tx D	TRANSMITTER DATA
TxDRQ	TRANSMITTER DMA REQUEST	$\overline{CTS}$	CLEAR TO SEND
$\overline{RD}$	READ INPUT	$\overline{CD}$	CARRIER DETECT
$\overline{WR}$	WRITE INPUT	$\overline{PA_2}$–$\overline{PA_4}$	GP INPUT PORTS
$\overline{Rx\ DACK}$	RECEIVER DMA ACKNOWLEDGE	$\overline{PB_1}$–$\overline{PB_4}$	GP OUTPUT PORTS
Rx DRQ	RECEIVER DMA REQUEST	$\overline{RTS}$	REQUEST TO SEND
Rx INT	RECEIVER INTERRUPT	V_{CC}	+5 VOLT SUPPLY
A0–A1	COMMAND REGISTER SELECT ADDRESS	GND	GROUND
$\overline{DPLL}$	DIGITAL PHASE LOCKED LOOP		

A BRIEF DESCRIPTION OF HDLC/SDLC PROTOCOLS

General

The High Level Data Link Control (HDLC) is a standard communication link protocol established by International Standards Organization (ISO). HDLC is the discipline used to implement ISO X.25 packet switching systems.

The Synchronous Data Link Control (SDLC) is an IBM communication link protocol used to implement the System Network Architecture (SNA). Both the protocols are bit oriented, code independent, and ideal for full duplex communication. Some common applications include terminal to terminal, terminal to CPU, CPU to CPU, satellite communication, packet switching and other high speed data links. In systems which require expensive cabling and interconnect hardware, any of the two protocols could be used to simplify interfacing (by going serial), thereby reducing interconnect hardware costs. Since both the protocols are speed independent, reducing interconnect hardware could become an important application.

Network

In both the HDLC and SDLC line protocols, according to a pre-assigned hierarchy, a PRIMARY (Control) STATION controls the overall network (data link) and issues commands to the SECONDARY (Slave) STATIONS. The latter comply with instructions and respond by sending appropriate RESPONSES. Whenever a transmitting station must end transmission prematurely it sends an ABORT character. Upon detecting an abort character, a receiving station ignores the transmission block called a FRAME. Time fill between frames can be accomplished by transmitting either continuous frame preambles called FLAGS or an abort character. A time fill within a frame is not permitted. Whenever a station receives a string of more that fifteen consecutive ones, the station goes into an IDLE state.

Frames

A single communication element is called a FRAME which can be used for both Link Control and data transfer purposes. The elements of a frame are the beginning eight bit FLAG (F) consisting of one zero, six ones, and a zero, an eight bit ADDRESS FIELD (A), an eight bit CONTROL FIELD (C), a variable (N-bit) INFORMATION FIELD (I), a sixteen bit FRAME CHECK SEQUENCE (FCS), and an eight bit end FLAG (F), having the same bit pattern as the beginning flag. In HDLC the Address (A) and Control (C) bytes are extendable. The HDLC and the SDLC use three types of frames; an Information Frame is used to transfer data, a Supervisory Frame is used for control purposes, and a Non-sequenced Frame is used for initialization and control of the secondary stations.

Frame Characteristics

An important characteristic of a frame is that its contents are made code transparent by use of a zero bit insertion and deletion technique. Thus, the user can adopt any format or code suitable for his system — it may even be a computer word length or a "memory dump". The frame is bit oriented that is, bits, not characters in each field, have specific meanings. The Frame Check Sequence (FCS) is an error detection scheme similar to the Cyclic Redundancy Checkword (CRC) widely used in magnetic disk storage devices. The Command and Response information frames contain sequence numbers in the control fields identifying the sent and received frames. The sequence numbers are used in Error Recovery Procedures (ERP) and as implicit acknowledgement of frame communication, enhancing the true full-duplex nature of the HDLC/SDLC protocols.

In contrast, BISYNC is basically half-duplex (two way alternate) because of necessity to transmit immediate acknowledgement frames. HDLC/SDLC therefore saves propagation delay times and have a potential of twice the throughput rate of BISYNC.

It is possible to use HDLC or SDLC over half duplex lines but there is a corresponding loss in throughput because both are primarily designed for full-duplex communication. As in any synchronous system, the bit rate is determined by the clock bits supplied by the modem, protocols themselves are speed independent.

A byproduct of the use of zero-bit insertion-deletion technique is the non-return-to-zero invert (NRZI) data transmission/reception compatibility. The latter allows HDLC/SDLC protocols to be used with asynchronous data communication hardware in which the clocks are derived from the NRZI encoded data.

References

IBM Synchronous Data Link Control General Information, IBM, GA27-3093-1.
Standard Network Access Protocol Specification, DATAPAC, Trans-Canada Telephone System CCG111
Recommendation X.25, ISO/CCITT March 2, 1976.
IBM 3650 Retail Store System Loop Interface OEM Information, IBM, GA 27-3098-0
Guidebook to Data Communications, Training Manual, Hewlett-Packard 5955-1715
IBM Introduction to Teleprocessing, IBM, GC 20-8095-02
System Network Architecture, Technical Overview, IBM, GA 27-3102
System Network Architecture Format and Protocol, IBM GA 27-3112

OPENING FLAG (F)	ADDRESS FIELD (A)	CONTROL FIELD (C)	INFORMATION FIELD (I)	FRAME CHECK SEQUENCE (FCS)	CLOSING FLAG (F)
01111110	8 BITS	8 BITS	VARIABLE LENGTH (ONLY IN I FRAMES)	16 BITS	01111110

Figure 1. Frame Format

FUNCTIONAL DESCRIPTION

General

The Intel® 8273 HDLC/SDLC controller is a microcomputer peripheral device which supports the International Standards Organization (ISO) High Level Data Link Control (HDLC), and IBM Synchronous Data Link Control (SDLC) communications protocols. This controller minimizes CPU software by supporting a comprehensive frame-level instruction set and by hardware implementation of the low level tasks associated with frame assembly/disassembly and data integrity. The 8273 can be used in either synchronous or asynchronous applications. In asynchronous applications the data can be programmed to be encoded/decoded in NRZI code. The clock is derived from the NRZI data using a digital phase locked loop. The data transparency is achieved by using a zero-bit insertion/deletion technique. The frames are automatically checked for errors during reception by verifying the Frame Check Sequence (FCS); the FCS is automatically generated and appended before the final flag in transmit.

The 8273 recognizes and can generate flags (01111110), Abort, Idle, and GA (EOP) characters.

The 8273 can assume either a primary (control) or a secondary (slave) role. It can therefore be readily implemented in an SDLC loop configuration as typified by the IBM 3650 Retail Store System by programming the 8273 into a one-bit delay mode. In such a configuration, a two wire pair can be effectively used for data transfer between controllers and loop stations. The digital phase locked loop output pin can be used by the loop station without the presence of an accurate Tx clock.

Hardware Description

The 8273 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Pin Name (No.)	I/O	Description
V_{CC} (40)		+5V Supply
GND (20)		Ground
RESET (4)	I	A high signal on this pin will force the 8273 to an idle state. The 8273 will remain idle until a command is issued by the CPU. The modem interface output signals are forced high. Reset must be true for a minimum of 10 TCY.
$\overline{CS}$ (24)	I	The $\overline{RD}$ and $\overline{WR}$ inputs are enabled by the chip select input.
DB_7-DB_0 (19-12)	I/O	The Data Bus lines are bidirectional three-state lines which interface with the system Data Bus.
$\overline{WR}$ (10)	I	The Write signal is used to control the transfer of either a command or data from CPU to the 8273.
$\overline{RD}$ (9)	I	The Read signal is used to control the transfer of either a data byte or a status word from the 8273 to the CPU.
TxINT (2)	O	The Transmitter interrupt signal indicates that the transmitter logic requires service.
RxINT (11)	O	The Receiver interrupt signal indicates that the Receiver logic requires service.
TxDRQ (6)	O	Requests a transfer of data between memory and the 8273 for a transmit operation.
RxRDQ (8)	O	Requests a transfer of data between the 8273 and memory for a receive operation.
$\overline{TxDACK}$ (5)	I	The Transmitter DMA acknowledge signal notifies the 8273 that the TxDMA cycle has been granted.
$\overline{RxDACK}$ (7)	I	The Receiver DMA acknowledge signal notifies the 8273 that the RxDMA cycle has been granted.
A_1-A_0 (22-21)	I	These two lines are CPU Interface Register Select lines.
TxD (29)	O	This line transmits the serial data to the communication channel.
$\overline{TxC}$ (28)	I	The transmitter clock is used to synchronize the transmit data.
RxD (26)	I	This line receives serial data from the communication channel.
$\overline{RxC}$ (27)	I	The Receiver Clock is used to synchronize the receive data.
$\overline{32X\ CLK}$ (25)	I	The 32X clock is used to provide clock recovery when an asynchronous modem is used. In loop configuration the loop station can run without an accurate 1X clock by using the 32X CLK in conjunction with the DPLL output. (This pin must be grounded when not used).
$\overline{DPLL}$ (23)	O	Digital Phase Locked Loop output can be tied to RxC and/or TxC when 1X clock is not available. DPLL is used with 32X CLK.
$\overline{FLAG\ DET}$ (1)	O	Flag Detect signals that a flag (01111110) has been received by an active receiver.
$\overline{RTS}$ (35)	O	Request to Send signals that the 8273 is ready to transmit data.
$\overline{CTS}$ (30)	I	Clear to Send signals that the modem is ready to accept data from the 8273.
$\overline{CD}$ (31)	I	Carrier Detect signals that the line transmission has started and the 8273 may begin to sample data on RxD line.
$\overline{PA_{2-4}}$ (32-34)	I	General purpose input ports. The logic levels on these lines can be Read by the CPU through the Data Bus Buffer.
$\overline{PB_{1-4}}$ (36-39)	O	General purpose output ports. The CPU can write these output lines through Data Bus Buffer.
CLK (3)	I	A square wave TTL clock.

CPU Interface

The CPU interface is optimized for the MCS-80/85™ bus with an 8257 DMA controller. However, the interface is flexible, and allows either DMA or non-DMA data transfers, interrupt or non-interrupt driven. It further allows maximum line utilization by providing early interrupt mechanism for buffered (only the information field can be transferred to memory) Tx command overlapping. It also provides separate Rx and Tx interrupt output channels for efficient operation. The 8273 keeps the interrupt request active until all the associated interrupt results have been read.

The CPU utilizes the CPU interface to specify commands and transfer data. It consists of seven registers addressed via $\overline{CS}$, A_1, A_0, $\overline{RD}$ and $\overline{WR}$ signals and two independent data registers for receive data and transmit data. A_1, A_0 are generally derived from two low order bits of the address bus. If an 8080 based CPU is utilized, the $\overline{RD}$ and $\overline{WR}$ signals may be driven by the 8228 $\overline{I/OR}$ and $\overline{I/OW}$. The table shows the seven register select decoding:

A_1	A_0	$\overline{TxDACK}$	$\overline{RxDACK}$	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	Register
0	0	1	1	0	1	0	Command
0	0	1	1	0	0	1	Status
0	1	1	1	0	1	0	Parameter
0	1	1	1	0	0	1	Result
1	0	1	1	0	1	0	Reset
1	0	1	1	0	0	1	TxINT Result
1	1	1	1	0	1	0	—
1	1	1	1	0	0	1	RxINT Result
X	X	0	1	1	1	0	Transmit Data
X	X	1	0	1	0	1	Receive Data

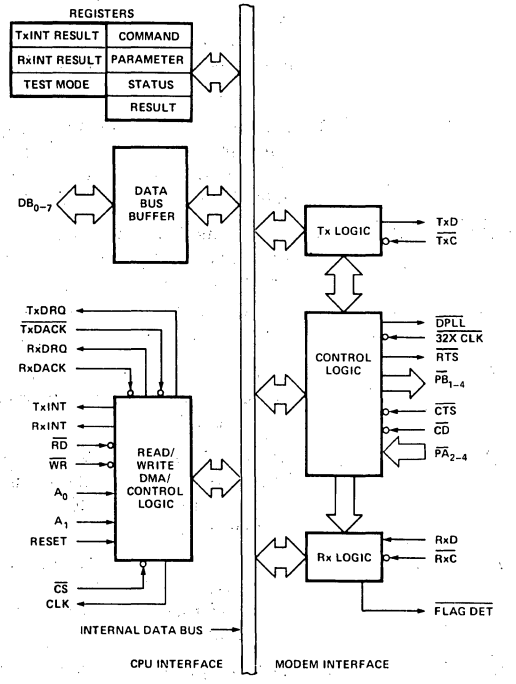


Figure 2. 8273 Block Diagram Showing CPU Interface Functions

Register Description

Command

Operations are initiated by writing an appropriate command in the Command Register.

Parameter

Parameters of commands that require additional information are written to this register.

Result

Contains an immediate result describing an outcome of an executed command.

Transmit Interrupt Result

Contains the outcome of 8273 transmit operation (good/bad completion).

Receive Interrupt Result

Contains the outcome of 8273 receive operation (good/bad completion), followed by additional results which detail the reason for interrupt.

Status

The status register reflects the state of the 8273 CPU Interface.

DMA Data Transfers

The 8273 CPU interface supports two independent data interfaces: receive data and transmit data. At high data transmission speeds the data transfer rate of the 8273 is great enough to justify the use of direct memory access (DMA) for the data transfers. When the 8273 is configured in DMA mode, the elements of the DMA interfaces are:

TxDRQ: Transmit DMA Request

Requests a transfer of data between memory and the 8273 for a transmit operation.

$\overline{TxDACK}$: Transmit DMA Acknowledge

The $\overline{TxDACK}$ signal notifies the 8273 that a transmit DMA cycle has been granted. It is also used with $\overline{WR}$ to transfer data to the 8273 in non-DMA mode.

RxDRQ: Receive DMA Request

Requests a transfer of data between the 8273 and memory for a receive operation.

$\overline{\text{RxDACK}}$: Receive DMA Acknowledge

The $\overline{\text{RxDACK}}$ signal notifies the 8273 that a receive DMA cycle has been granted. It is also used with $\overline{\text{RD}}$ to read data from the 8273 in non-DMA mode.

$\overline{\text{RD}}$, $\overline{\text{WR}}$: Read, Write

The $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals are used to specify the direction of the data transfer.

DMA transfers require the use of a DMA controller such as the Intel 8257. The function of the DMA controller is to provide sequential addresses and timing for the transfer, at a starting address determined by the CPU. Counting of data block lengths is performed by the 8273.

To request a DMA transfer the 8273 raises the appropriate DMA REQUEST. DMA ACKNOWLEDGE and READ enables DMA data onto the bus (independently of CHIP SELECT). DMA ACKNOWLEDGE and WRITE transfers DMA data to the 8273 (independent of CHIP SELECT).

It is also possible to configure the 8273 in the non-DMA data transfer mode. In this mode the CPU module must pass data to the 8273 in response to non-DMA data requests indicated by the status word.

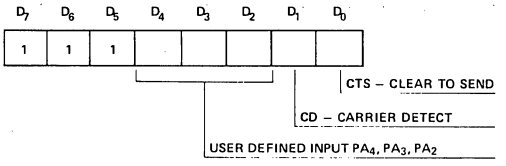
Modem Interface

The 8273 Modem interface provides both dedicated and user defined modem control functions. All the control signals are active low so that EIA RS-232C inverting drivers (MC 1488) and inverting receivers (MC 1489) may be used to interface to standard modems. For asynchronous operation, this interface supports programmable NRZI data encode/decode, a digital phase locked loop for efficient clock extraction from NRZI data, and modem control ports with automatic $\overline{\text{CTS}}$, $\overline{\text{CD}}$ monitoring and $\overline{\text{RTS}}$ generation. This interface also allows the 8273 to operate in PRE-FRAME SYNC mode in which the 8273 prefixes 16 transitions to a frame to synchronize idle lines before transmission of the first flag.

It should be noted that all the 8273 port operations deal with logical values, for instance, bit D0 of Port A will be a one when $\overline{\text{CTS}}$ (Pin 30) is a physical zero (logical one).

Port A — Input Port

During operation, the 8273 interrogates input pins $\overline{\text{CTS}}$ (Clear to Send) and $\overline{\text{CD}}$ (Carrier Detect). $\overline{\text{CTS}}$ is used to condition the start of a transmission. If during transmission $\overline{\text{CTS}}$ is lost the 8273 generates an interrupt. During reception, if $\overline{\text{CD}}$ is lost, the 8273 generates an interrupt.



The user defined input bits correspond to the 8273 PA_4, PA_3 and PA_2 pins. The 8273 does not interrogate or manipulate these bits.

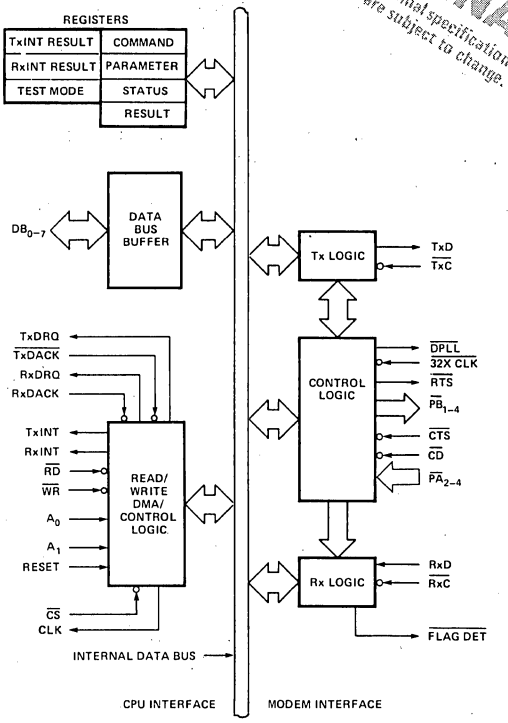
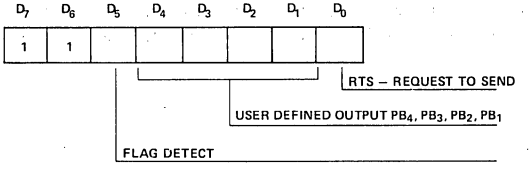


Figure 3. 8273 Block Diagram Showing Control Logic Functions

Port B - Output Port

During normal operation, if the CPU sets $\overline{\text{RTS}}$ active, the 8273 will not change this pin; however, if the CPU sets $\overline{\text{RTS}}$ inactive, the 8273 will activate it before each transmission and deactivate it one byte time after transmission. While the receiver is active the flag detect pin is pulsed each time a flag sequence is detected in the receive data stream. Following an 8273 reset, all pins of Port B are set to a high, inactive level.



The user defined output bits correspond to the state of PB_4-PB_1 pins. The 8273 does not interrogate or manipulate these bits.

Serial Data Logic

The Serial data is synchronized by the user transmit ($\overline{TxC}$) and receive ($\overline{RxC}$) clocks. The leading edge of $\overline{TxC}$ generates new transmit data and the trailing edge of $\overline{RxC}$ is used to capture receive data. The NRZI encoding/decoding of the receive and transmit data is programmable.

The diagnostic features included in the Serial Data logic are programmable loop back of data and selectable clock for the receiver. In the loop-back mode, the data presented to the TxD pin is internally routed to the receive data input circuitry in place of the RxD pin, thus allowing a CPU to send a message to itself to verify operation of the 8273.

In the selectable clock diagnostic feature, when the data is looped back, the receiver may be presented incorrect sample timing by the external circuitry. The user may select to substitute the $\overline{TxC}$ pin for the $\overline{RxC}$ input on-chip so that the clock used to generate the loop back data is used to sample it. Since TxD is generated off the leading edge of $\overline{TxC}$ and RxD is sampled on the trailing edge, the selected clock allows bit synchronism.

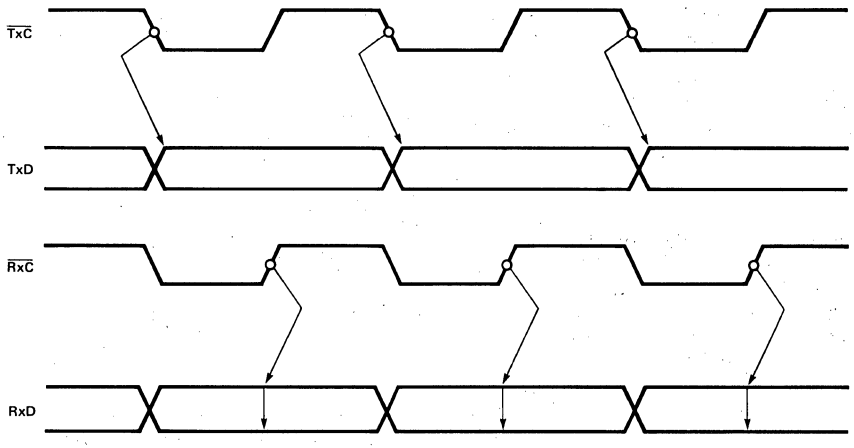


Figure 4. Transmit/Receive Timing

Asynchronous Mode Interface

Although the 8273 is fully compatible with the HDLC/SDLC communication line protocols, which are primarily designed for synchronous communication, the 8273 can also be used in asynchronous applications by using this interface. The interface employs a digital phase locked loop (DPLL) for clock recovery from a receive data stream and programmable NRZI encoding and decoding of data. The use of NRZI coding with SDLC transmission guarantees that within a frame, data transitions will occur at least every five bit times — the longest sequence of ones which may be transmitted without zero-bit insertion. The DPLL should be used only when NRZI coding is used since the NRZI coding will transmit zero sequence as line transitions. The digital phase locked loop also facilitates full-duplex and half-duplex asynchronous implementation with, or without modems.

Digital Phase Locked Loop

In asynchronous applications, the clock is derived from the receiver data stream by the use of the digital phase locked loop (DPLL). The DPLL requires a clock input at 32 times the required baud rate. The receive data (RxD) is sampled with this $\overline{32X\ CLK}$ and the 8273 DPLL supplies a sample pulse nominally centered on the RxD bit cells. The DPLL has a built-in "stiffness" which reduces sensitivity to line noise and bit distortion. This is accomplished by making phase error adjustments in discrete increments. Since the nominal pulse is made to occur at 32 counts of the $\overline{32X\ CLK}$, these counts are subtracted or added to the nominal, depending upon which quadrant of the four error quadrants the data edge occurs in. For example if an RxD edge is detected in quadrant A1, it is apparent that the $\overline{DPLL}$ sample "A" was placed too close to the trailing edge of the data cell; sample "B" will then be placed at T = ($T_{nominal}$ - 2 counts) = 30 counts of the $\overline{32X\ CLK}$ to move the sample pulse "B" toward the nominal center of the next bit cell. A data edge occuring in quadrant B1 would cause a smaller adjustment of phase with T = 31 counts of the $\overline{32X\ CLK}$. Using this technique the $\overline{DPLL}$ pulse will converge to nominal bit center within 12 data bit times, worst case, with constant incoming RxD edges.

A method of attaining bit synchronism following a line idle is to use PRE-FRAME SYNC mode of transmission.

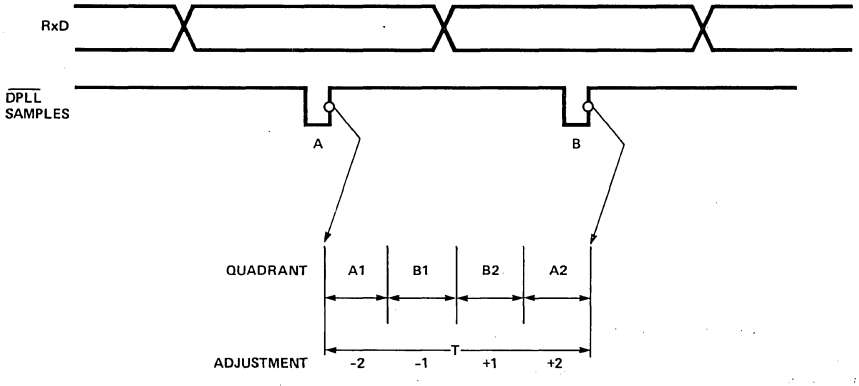
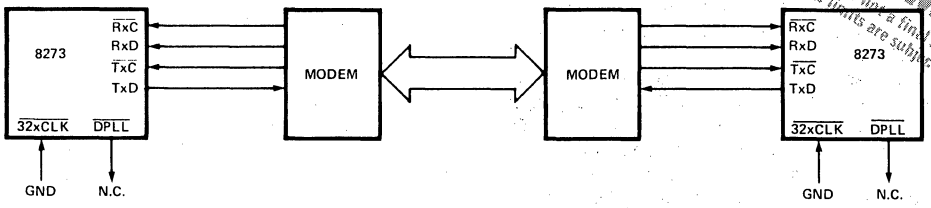
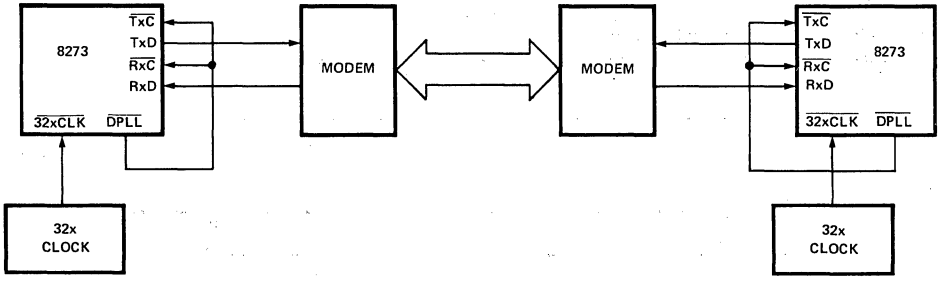


Figure 5. DPLL Sample Timing

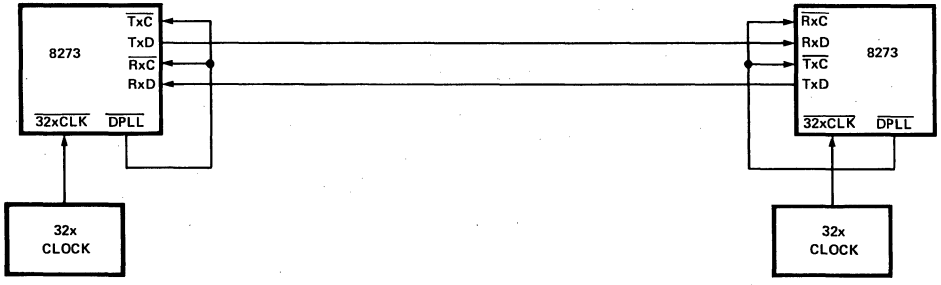
Synchronous Modem — Duplex or Half Duplex Operation



Asynchronous Modems — Duplex or Half Duplex Operation



Asynchronous — No Modems — Duplex or Half Duplex



SDLC Loop

The DPLL simplifies the SDLC loop station implementation. In this application, each secondary station on a loop data link is a repeater set in one-bit delay mode. The signals sent out on the loop by the loop controller (primary station) are relayed from station to station then, back to the controller. Any secondary station finding its address in the A field captures the frame for action at that station. All received frames are relayed to the next station on the loop.

Loop stations are required to derive bit timing from the incoming NRZI data stream. The DPLL generates sample Rx clock timing for reception and uses the same clock to implement Tx clock timing.

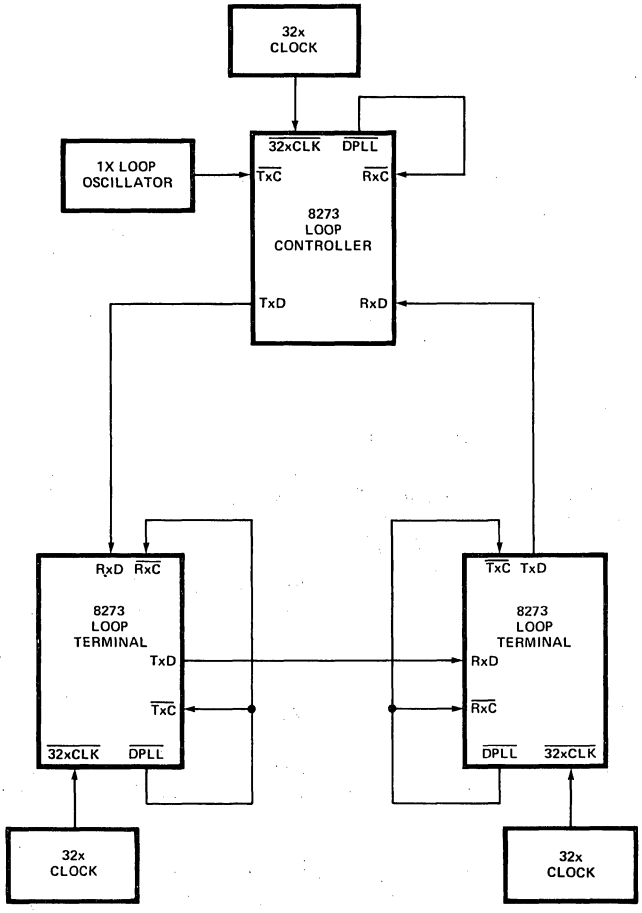


Figure 6. SDLC Loop Application

PRINCIPLES OF OPERATION

The 8273 is an intelligent peripheral controller which relieves the CPU of many of the rote tasks associated with constructing and receiving frames. It is fully compatible with the MCS-80/85™ system bus. As a peripheral device, it accepts commands from a CPU, executes these commands and provides an Interrupt and Result back to the CPU at the end of the execution. The communication with the CPU is done by activation of $\overline{CS}$, $\overline{RD}$, $\overline{WR}$ pins, while the A_1, A_0 select the appropriate registers on the chip as described in the Hardware Description Section.

The 8273 operation is composed of the following sequence of events:

COMMAND PHASE	CPU WRITES COMMAND AND PARAMETERS INTO THE 8273 COMMAND AND PARAMETER REGISTERS.
EXECUTION PHASE	THE 8273 IS ON ITS OWN TO CARRY OUT THE COMMAND.
RESULT PHASE	THE 8273 SIGNALS THE CPU THAT THE EXECUTION HAS FINISHED. THE CPU MUST PERFORM A READ OPERATION OF ONE OR MORE OF THE REGISTERS.

The Command Phase

During the command phase, the software writes a command to the command register. The command bytes provide a general description of the type of operation requested. Many commands require more detailed information about the command. In such a case up to four parameters are written into the parameter register. The flowchart of the command phase indicates that a command may not be issued if the Status Register indicates that the device is busy. Similarly if a parameter is issued when the Parameter Buffer shows full, incorrect operation will occur.

The 8273 is a duplex device and both transmitter and receiver may each be executing a command or passing results at any given time. For this reason separate interrupt pins are provided. However, the command register must be used for one command sequence at a time.

Status Register

The status register contains the status of the 8273 activity. The description is as follows.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CBSY	CBF	CPBF	CRBF	RxINT	TxINT	RxIRA	TxIRA

Bit 7 CBSY (Command Busy)

Indicates in-progress command, set for CPU poll when Command Register is full, reset upon command phase completion. It is improper to write a command when CBSY is set; it results in incorrect operation.

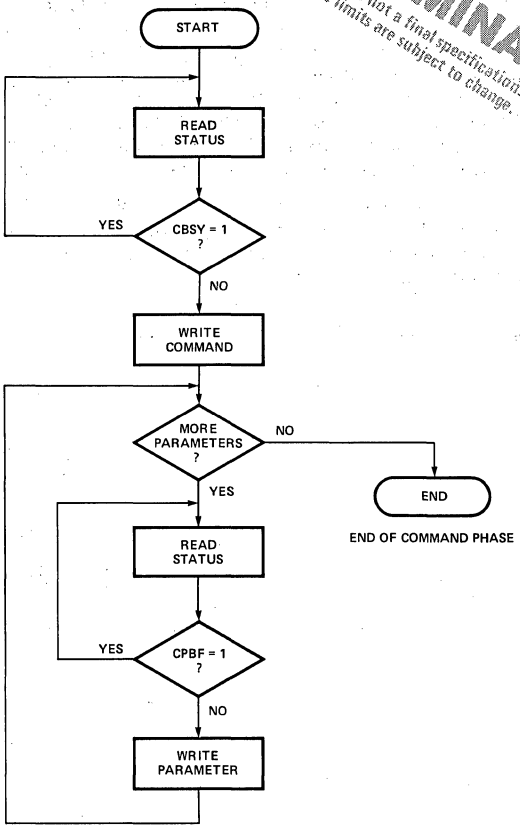


Figure 7. Command Phase Flowchart

Bit 6 CBF (Command Buffer Full)

Indicates that the command register is full, it is reset when the 8273 accepts the command byte but does not imply that execution has begun.

Bit 5 CPBF (Command Parameter Buffer Full)

CPBF is set when the parameter buffer is full, and is reset by the 8273 when it accepts the parameter. The CPU may poll CPBF to determine when additional parameters may be written.

Bit 4 CRBF (Command Result Buffer Full)

Indicates that an executed command immediate result is present in the Result Register. It is set by 8273 and reset when CPU reads the result.

Bit 3 RxINT (Receiver Interrupt)

RxINT indicates that the receiver requires CPU attention. It is identical to RxINT (pin 11) and is set by the 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has received a data byte from the 8273 in a Non-DMA data transfer.

Bit 2 TxINT (Transmitter Interrupt)

The TxINT indicates that the transmitter requires CPU attention. It is identical to TxINT (pin 2). It is set by 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has transferred transmit data byte to the 8273 in a Non-DMA transfer.

Bit 1 RxIRA (Receiver Interrupt Result Available)

The RxIRA is set by the 8273 when an interrupt result byte is placed in the RxINT register. It is reset after the CPU has read the RxINT register.

Bit 0 TxIRA (Transmitter Interrupt Result Available)

The TxIRA is set by the 8273 when an interrupt result byte is placed in the TxINT register. It is reset when the CPU has read the TxINT register.

The Execution Phase

Upon accepting the last parameter, the 8273 enters into the Execution Phase. The execution phase may consist of a DMA or other activity, and may or may not require CPU intervention. The CPU intervention is eliminated in this phase if the system utilizes DMA for the data transfers, otherwise, for non-DMA data transfers, the CPU is interrupted by the 8273 via TxINT and RxINT pins, for each data byte request.

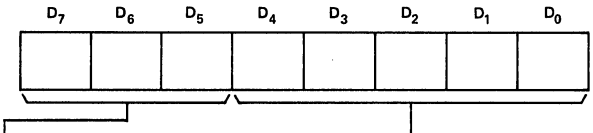
The Result Phase

During the result phase, the 8273 notifies the CPU of the execution outcome of a command. This phase is initiated by:

1. The successful completion of an operation
2. An error detected during an operation.

To facilitate quick network software decisions, two types of execution results are provided:

1. An Immediate Result
2. A Non-Immediate Result

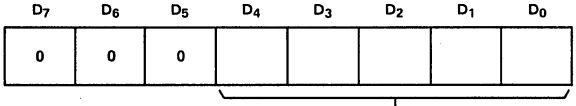


D_7	D_6	D_5	
1	1	1	All 8 bits received
0	0	0	D_0 received
1	0	0	D_1-D_0 received
0	1	0	D_2-D_0 received
1	1	0	D_3-D_0 received
0	0	1	D_4-D_0 received
1	0	1	D_5-D_0 received
0	1	1	D_6-D_0 received

*Partial Byte Received

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Receiver Interrupt Result Code	Rx Status After INT
*	*	*	0	0	0	0	0	A_1 match or general receive	Active
*	*	*	0	0	0	0	1	A_2 match	Active
0	0	0	0	0	0	1	1	CRC error	Active
0	0	0	0	0	1	0	0	Abort detected	Active
0	0	0	0	0	1	0	1	Idle detect	Disabled
0	0	0	0	0	1	1	0	EOP detected	Disabled
0	0	0	0	0	1	1	1	Frame less than 32 bits	Active
0	0	0	0	1	0	0	0	DMA overrun detected	Disabled
0	0	0	0	1	0	0	1	Memory buffer overflow	Disabled
0	0	0	0	1	0	1	0	Carrier detect failure	Disabled
0	0	0	0	1	0	1	1	Receive interrupt overrun	Disabled

Figure 8. Rx Interrupt Result Byte Format



D_4	D_3	D_2	D_1	D_0	
0	1	1	0	0	Early transmit interrupt
0	1	1	0	1	Frame transmit complete
0	1	1	1	0	DMA underrun
0	1	1	1	1	Clear to Send (CTS) error
1	0	0	0	0	Abort complete

Figure 10. Tx Interrupt Result Byte Format

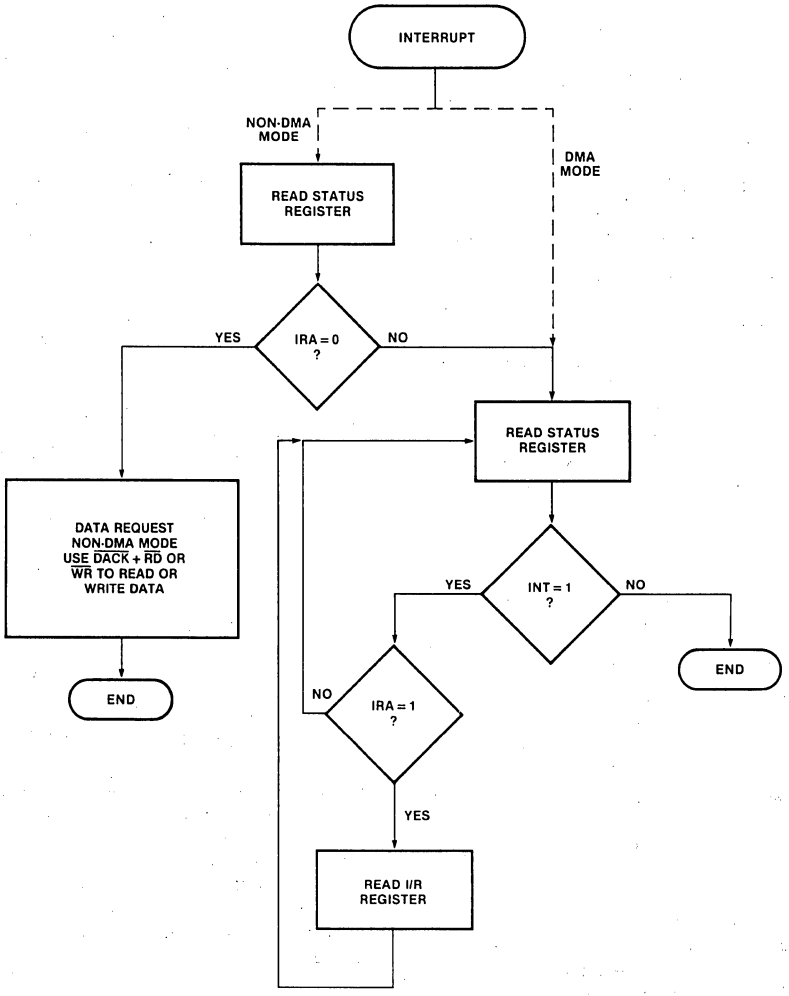
Immediate result is provided by the 8273 for commands such as Read Port A and Read Port B which have information ($\overline{CTS}$, $\overline{CD}$, $\overline{RTS}$, etc.) that the network software needs to make quick operational decisions.

A command which cannot provide an immediate result will generate an interrupt to signal the beginning of the Result phase. The immediate results are provided in the Result Register; all non-immediate results are available upon device interrupt, through Tx Interrupt Result Register TxI/R or Rx Interrupt Result Register RxI/R. The result may consist of a one-byte interrupt code indicating the condition for the interrupt and, if required, one or more bytes which detail the condition.

Tx and Rx Interrupt Result Registers

The Result Registers have a result code, the three high order bits D_7-D_5 of which are set to zero for all but the receive command. This command result contains a count that indicates the number of bits received in the last byte. If a partial byte is received, the high order bits of the last data byte are indeterminate.

All results indicated in the command summary must be read during the result phase.



RESULT PHASE FLOWCHART — INTERRUPT RESULTS

DETAILED COMMAND DESCRIPTION

General

The 8273 HDLC/SDLC controller supports a comprehensive set of high level commands which allows the 8273 to be readily used in full-duplex, half-duplex, synchronous, asynchronous and SDLC loop configuration, with or without modems. These frame-level commands minimize CPU and software overhead. The 8273 has address and control byte buffers which allow the receive and transmit commands to be used in buffered or non-buffered modes.

In buffered transmit mode, the 8273 transmits a flag automatically, reads the Address and Control buffer registers and transmits the fields, then via DMA, it fetches the information field. The 8273, having transmitted the information field, automatically appends the Frame Check Sequence (FCS) and the end flag. Correspondingly, in buffered read mode, the Address and Control fields are stored in their respective buffer registers and only Information Field is transferred to memory.

In non-buffered transmit mode, the 8273 transmits the beginning flag automatically, then fetches and transmits the Address, Control and Information fields from the memory, appends the FCS character and an end flag. In the non-buffered receive mode the entire contents of a frame are sent to memory with the exception of the flags and FCS.

HDLC Implementation

HDLC Address and Control field are extendable. The extension is selected by setting the low order bit of the field to be extended to a one, a zero in the low order bit indicates the last byte of the respective field.

Since Address/Control field extension is normally done with software to maximize extension flexibility, the 8273 does not create or operate upon contents of the extended HDLC Address/Control fields. Extended fields are transparently passed by the 8273 to user as either interrupt results or data transfer requests. Software must assemble the fields for transmission and interrogate them upon reception.

However, the user can take advantage of the powerful 8273 commands to minimize CPU/Software overhead and simplify buffer management in handling extended fields. For instance buffered mode can be used to separate the first two bytes, then interrogate the others from buffer. Buffered mode is perfect for a two byte address field.

The 8273 when programmed, recognizes protocol characters unique to HDLC such as Abort, which is a string of seven or more ones (01111111). Since Abort character is the same as the GA (EOP) character used in SDLC Loop applications, Loop Transmit and Receive commands are not recommended to be used in HDLC. HDLC does not support Loop mode.

Initialization Set/Reset Commands

These commands are used to manipulate data within the 8273 registers. The Set commands have a single parameter which is a mask that corresponds to the bits to be set. (They perform a logical-OR of the specified register with the mask provided as a parameter). The Register commands have a single parameter which is a mask that has a zero in the bit positions that are to be reset. (They perform a logical-AND of the specified register with the mask).

Set One-Bit Delay (CMD Code A4)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	1	0	0	1	0	0
PAR:	0	1	1	0	0	0	0	0	0	0

When one bit delay is set, 8273 retransmits the received data stream one bit delayed. This mode is entered at a receiver character boundary, and should only be used by Loop Stations.

Reset One-Bit Delay (CMD Code 64)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	1	0	0	1	0	0
PAR:	0	1	0	1	1	1	1	1	1	1

The 8273 stops the one bit delayed retransmission mode.

Set Data Transfer Mode (CMD Code 97)

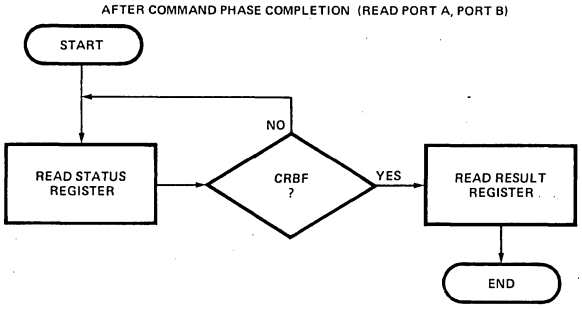
	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	0	1	0	1	1	1
PAR:	0	1	0	0	0	0	0	0	0	1

When the data transfer mode is set, the 8273 will interrupt when data bytes are required for transmission or are available from a receive. If a transmit interrupt occurs and the status indicates that there is no Transmit Result (TxIRA = 0), the interrupt is a transmit data request. If a receive interrupt occurs and the status indicates that there is no receive result (RxIRA = 0), the interrupt is a receive data request.

Reset Data Transfer Mode (CMD Code 57)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	0	1	0	1	1	1
PAR:	0	1	1	1	1	1	1	1	1	0

If the Data Transfer Mode is reset, the 8273 data transfers are performed through the DMA requests without interrupting the CPU.



RESULT PHASE FLOWCHART — IMMEDIATE RESULTS

Figure 9. Rx Interrupt Service

Set Operating Mode (CMD Code 91)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	0	1	0	0	0	1
PAR:	0	1	0	0						

1 = FLAG STREAM MODE
1 = PREFRAME SYNC MODE
1 = BUFFERED MODE
1 = EARLY INTERRUPT MODE
1 = EOP INTERRUPT MODE
1 = HDLC MODE

Reset Operating Mode (CMD Code 51)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	0	1	0	0	0	1
PAR:	0	1	1	1						

Any mode switches set in CMD code 91 can be reset using this command by placing zeros in the appropriate positions.

(D5) HDLC Mode

In HDLC mode, a bit sequence of seven ones (01111111) is interpreted as an abort character. Otherwise, eight ones (011111111) signal an abort.

(D4) EOP Interrupt Mode

In EOP interrupt mode, an interrupt is generated whenever an EOP character (01111111) is detected by an active receiver. This mode is useful for the implementation of an SDLC loop controller in detecting the end of a message stream after a loop poll.

(D3) Transmitter Early Interrupt Mode (Tx)

The early interrupt mode is specified to indicate when the 8273 should generate an end of frame interrupt. When set, an early interrupt is generated when the last data character has been passed to the 8273. If the user software responds with another transmit command before the final flag is sent, the final flag interrupt will not be generated and a new frame will immediately begin when the current frame is complete. This permits frames to be separated by a single flag. If no additional Tx commands are provided, a final interrupt will follow.

If this bit is zero, the interrupt will be generated only after the final flag has been transmitted.

(D2) Buffered Mode

If the buffered mode bit is set to a one, the first two bytes (normally the address (A) and control (C) fields) of a frame are buffered by the 8273. If this bit is a zero the address and control fields are passed to and from memory.

(D1) Preframe Sync Mode

If this bit is set to a one the 8273 will transmit two characters before the first flag of a frame.
To guarantee sixteen line transitions, the 8273 sends two bytes of data $(00)_H$ if NRZI is set or data $(55)_H$ if NRZI is not set.

(D0) Flag Stream Mode

If this bit is set to a one, the following table outlines the operation of the transmitter.

TRANSMITTER STATE	ACTION
Idle	Send Flags immediately.
Transmit or Transmit-Transparent Active	Send Flags after the transmission complete
Loop Transmit Active	Ignore command.
1 Bit Delay Active	Ignore command.

If this bit is reset to zero the following table outlines the operation of the transmitter.

TRANSMITTER STATE	ACTION
IDLE	Send Idles on next character boundary.
Transmit or Transmit-Transparent Active	Send Idles after the transmission is complete.
Loop Transmit Active	Ignore command.
1 Bit Delay Active	Ignore command.

Set Serial I/O Mode (CMD Code A0)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	1	0	0	0	0	0
PAR:	0	1	0	0	0	0	0			

1 = NRZ1 MODE
1 = TxC → RxC
1 = LOOP BACK TxD → RxD

Reset Serial I/O Mode (CMD Code 60)

This command allows bits set in CMD code A0 to be reset by placing zeros in the appropriate positions.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	1	0	0	0	0	0
PAR:	0	1	1	1	1	1	1			

(D2) Loop Back

If this bit is set to a one, the transmit data is internally routed to the receive data circuitry.

(D1) TxC → RxC

If this bit is set to a one, the transmit clock is internally routed to the receive clock circuitry. It is normally used with the loop back bit (D2).

(D0) NRZI Mode

If this bit is set to a one, NRZI encoding and decoding of transmit and receive data is provided. If this bit is a zero, the transmit and receive data is treated as a normal positive logic bit stream.

NRZI encoding specifies that a zero causes a change in the polarity of the transmitted signal and a one causes no polarity change. NRZI is used in all asynchronous operations. Refer to IBM document GA27-3093 for details.

Reset Device Command

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TMR:	1	0	0	0	0	0	0	0	0	1
TMR:	1	0	0	0	0	0	0	0	0	0

An 8273 reset command is executed by outputing a $(01)_H$ followed by $(00)_H$ to the reset register (TMR). See 8273 AC timing characteristics for Reset pulse specifications.

The reset command emulates the action of the reset pin.

1. The modem control signals are forced high (inactive level).
2. The 8273 status register flags are cleared.
3. Any commands in progress are terminated immediately.
4. The 8273 enters an idle state until the next command is issued.
5. The Serial I/O and Operating Mode registers are set to zero and DMA data register transfer mode is selected.
6. The device assumes a non-loop SDLC terminal role.

Receive Commands

The 8273 supports three receive commands: General Receive, Selective Receive, and Selective Loop Receive.

General Receive (CMD Code C0)

General receive is a receive mode in which frames are received regardless of the contents of the address field.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	0	0	0	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							

NOTES:

1. If buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received.
2. If non-buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received plus two (the count includes the address and control bytes).
3. The frame check sequence (FCS) is not transferred to memory.
4. Frames with less than 32 bits between flags are ignored (no interrupt generated) if the buffered mode is specified.
5. In the non-buffered mode an interrupt is generated when a less than 32 bit frame is received, since data transfer requests have occurred.
6. The 8273 receiver is always disabled when an Idle is received after a valid frame. The CPU module must issue a receive command to re-enable the receiver.
7. The intervening ABORT character between a final flag and an IDLE does not generate an interrupt.
8. If an ABORT Character is not preceded by a flag and is followed by an IDLE, an interrupt will be generated for the ABORT followed by an IDLE interrupt one character time later. The reception of an ABORT will disable the receiver.

Selective Receive (CMD Code C1)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	0	0	0	1
PAR:	0	1	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)							

Selective receive is a receive mode in which frames are ignored unless the address field matches any one of two address fields given to the 8273 as parameters.

When selective receive is used in HDLC the 8273 looks at the first character, if extended, software must then decide if the message is for this unit.

Selective Loop Receive (CMD Code C2)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	0	0	1	0
PAR:	0	0	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)							

Selective loop receive operates like selective receive except that the transmitter is placed in flag stream mode automatically after detecting an EOP (01111111) following a valid received frame. The one bit delay mode is also reset at the end of a selective loop receive.

Receive Disable (CMD Code C5)

Terminates an active receive command immediately.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	0	1	0	1

PAR: NONE

Transmit Commands

The 8273 supports three transmit commands: Transmit Frame, Loop Transmit, Transmit Transparent.

Transmit Frame (CMD Code C8)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	0	0	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							
PAR:	0	1	ADDRESS FIELD OF TRANSMIT FRAME (A)							
PAR:	0	1	CONTROL FIELD OF TRANSMIT FRAME (C)							

Transmits one frame including: initial flag, frame check sequence, and the final flag.

If the buffered mode is specified, the L0, L1, frame length provided as a parameter is the length of the information field and the address and control fields must be input.

In unbuffered mode the frame length provided must be the length of the information field plus two and the address and control fields must be the first two bytes of data. Thus only the frame length bytes are required as parameters.

Loop Transmit (CMD Code CA)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	0	1	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							
PAR:	0	1	ADDRESS FIELD OF TRANSMIT FRAME (A)							
PAR:	0	1	CONTROL FIELD OF TRANSMIT FRAME (C)							

Transmits one frame in the same manner as the transmit frame command except:

1. This command should be given only in one-bit delay mode.
2. If the flag stream mode is not active transmission will begin after a received EOP has been converted to a flag.
3. If the flag stream mode is active transmission will begin at the next flag boundary for buffered mode or at the third flag boundary for non-buffered mode.
4. At the end of a loop transmit the one-bit delay mode is entered and the flag stream mode is reset.

Transmit Transparent (CMD Coded C9)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	0	0	1
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							

The 8273 will transmit a block of raw data without protocol, i.e., no zero bit insertion, flags, or frame check sequences.

Abort Transmit Commands

An abort command is supported for each type of transmit command. The abort commands are ignored if a transmit command is not in progress.

Abort Transmit Frame (CMD Code CC)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	1	0	0
PAR:	NONE									

After an abort character (eight contiguous ones) is transmitted, the transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

Abort Loop Transmit (CMD Code CE)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	1	1	0
PAR:	NONE									

After a flag is transmitted the transmitter reverts to one bit delay mode.

Abort Transmit Transparent (CMD Code CD)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	1	0	1
PAR:	NONE									

The transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

Modem Control Commands

The modem control commands are used to manipulate the modem control ports.

When read Port A or Port B commands are executed the result of the command is returned in the result register. The Bit Set Port B command requires a parameter that is a mask that corresponds to the bits to be set. The Bit Reset Port B command requires a mask that has a zero in the bit positions that are to be reset.

Read Port A (CMD Code 22)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	0	1	0	0	0	1	0

PAR: NONE

Read Port B (CMD Code 23)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	0	1	0	0	0	1	1

PAR: NONE

Set Port B Bits (CMD Code A3)

This command allows user defined Port B pins to be set.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	1	0	0	0	1	1
PAR:	0	1	0	0						

RTS – REQUEST TO SEND
USER DEFINED
FLAG DETECT

(D_5) Flag Detect

This bit can be used to set the flag detect pin. However, it will be reset when the next flag is detected.

(D_4-D_1) User Defined Outputs

These bits correspond to the state of the PB_4-PB_1 output pins.

(D_0) Request to Send

This is a dedicated 8273 modem control signal, and reflects the same logical state of RTS pin.

Reset Port B Bits (CMD Code 63)

This command allows Port B user defined bits to be reset.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	1	0	0	0	1	1
PAR:	0	1	1	1						

RTS – REQUEST TO SEND
USER DEFINED
FLAG DETECT

This command allows Port B (D_4-D_1) user defined bits to be reset. These bits correspond to Output Port pins (PB_4-PB_1).

8273 Command Summary

Command Description	Command (HEX)	Parameter	Results	Result Port	Completion Interrupt
Set One Bit Delay	A4	Set Mask	None	—	No
Reset One Bit Delay	64	Reset Mask	None	—	No
Set Data Transfer Mode	97	Set Mask	None	—	No
Reset Data Transfer Mode	57	Reset Mask	None	—	No
Set Operating Mode	91	Set Mask	None	—	No
Reset Operating Mode	51	Reset Mask	None	—	No
Set Serial I/O Mode	A0	Set Mask	None	—	No
Reset Serial I/O Mode	60	Reset Mask	None	—	No
General Receive	C0	B0,B1	RIC,R0,R1,(A,C)(2)	RXI/R	Yes
Selective Receive	C1	B0,B1,A1,A2	RIC,R0,R1,(A,C)(2)	RXI/R	Yes
Selective Loop Receive	C2	B0,B1,A1,A2	RIC,R0,R1,(A,C)(2)	RXI/R	Yes
Receive Disable	C5	None	None	—	No
Transmit Frame	C8	L0,L1,(A,C)(1)	TIC	TXI/R	Yes
Loop Transmit	CA	L0,L1,(A,C)(1)	TIC	TXI/R	Yes
Transmit Transparent	C9	L0,L1	TIC	TXI/R	Yes
Abort Transmit Frame	CC	None	TIC	TXI/R	Yes
Abort Loop Transmit	CE	None	TIC	TXI/R	Yes
Abort Transmit Transparent	CD	None	TIC	TXI/R	Yes
Read Port A	22	None	Port Value	Result	No
Read Port B	23	None	Port Value	Result	No
Set Port B Bit	A3	Set Mask	None	—	No
Reset Port B Bit	63	Reset Mask	None	—	No

Notes: 1. Issued only when in buffered mode. 2. Read as results only in buffered mode.

8273 Command Summary Key

B0 — Least significant byte of the receive buffer length.
B1 — Most significant byte of the receive buffer length.
L0 — Least significant byte of the Tx frame length.
L1 — Most significant byte of the Tx frame length.
A1 — Receive frame address match field one.
A2 — Receive frame address match field two.
A — Address field of received frame. If non-buffered mode is specified, this result is not provided.
C — Control field of received frame. If non-buffered mode is specified this result is not provided.
RXI/R — Receive interrupt result register.
TXI/R — Transmit interrupt result register.
R0 — Least significant byte of the length of the frame received.
R1 — Most significant byte of the length of the frame received.
RIC — Receiver interrupt result code.
TIC — Transmitter interrupt result code.

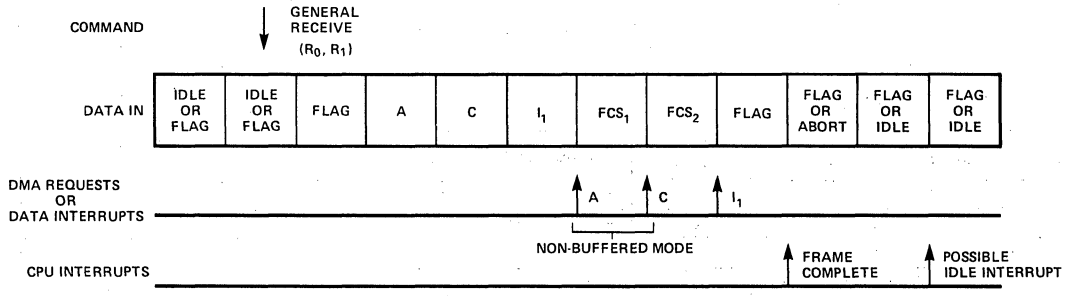


Figure 12. Typical Frame Reception

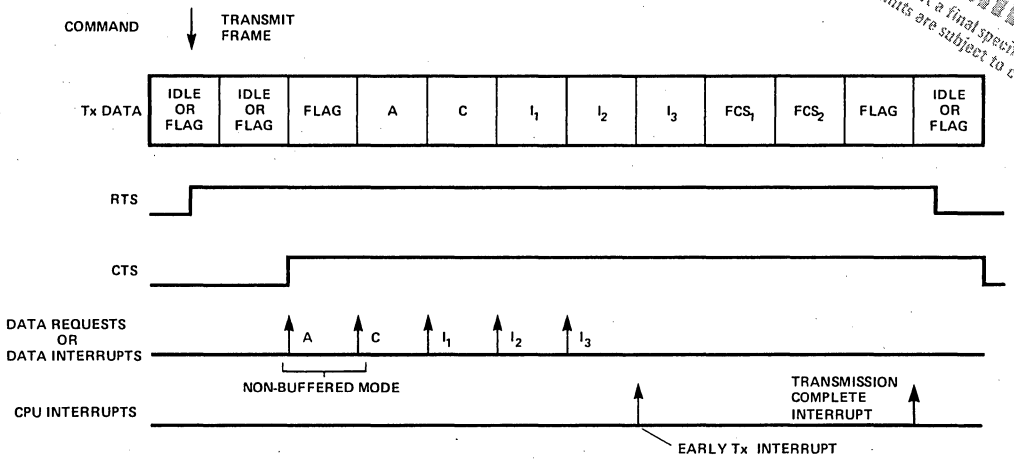


Figure 13. Typical Frame Transmission

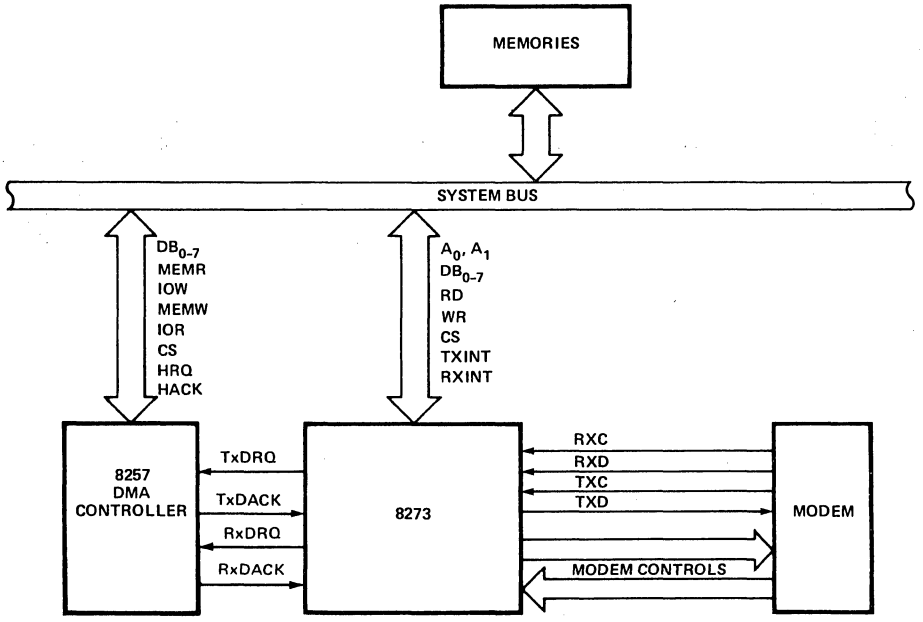


Figure 14. 8273 System Diagram

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... -65°C to +150°C
Voltage on Any Pin With
Respect to Ground ..... -0.5V to +7V
Power Dissipation ..... 1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5.0V ±5%

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.0 mA for Data Bus pins I_{OL} = 1.7 mA for all other pins
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −200 μA
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OZ}	Off-State Output Current		±10	μA	V_{OUT} = V_{CC} to 0V
I_{CC}	V_{CC} Supply Current		180	mA	

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	t_c = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = +5.0V \pm 5\%$

Read Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{RD}$	0		ns	Note 3
t_{CA}	Select Hold from $\overline{RD}$	0		ns	Note 3
t_{RR}	RD Pulse Width	0		ns	
t_{AD}	Data Delay from Address		250	ns	Note 3
t_{RD}	Data Delay from $\overline{RD}$		150	ns	$C_L = 150$ pF, Note 3
t_{DF}	Output Float Delay	20	100	ns	$C_L = 20$ pF for Minimum; 150 pF for Maximum
t_{DC}	DACK Setup to $\overline{RD}$	25		ns	
t_{CD}	DACK Hold from $\overline{RD}$	25		ns	
t_{KD}	Data Delay from DACK		250	ns	

Write Cycle

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{WR}$	0		ns	
t_{CA}	Select Hold from $\overline{WR}$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$	150		ns	
t_{WD}	Data Hold from $\overline{WR}$	0		ns	
t_{DC}	DACK Setup to $\overline{WR}$	25		ns	
t_{CD}	DACK Hold from $\overline{WR}$	25		ns	

DMA

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CQ}	Request Hold from $\overline{WR}$ or $\overline{RD}$ (for Non-Burst Mode)		150	ns	

Other Timing

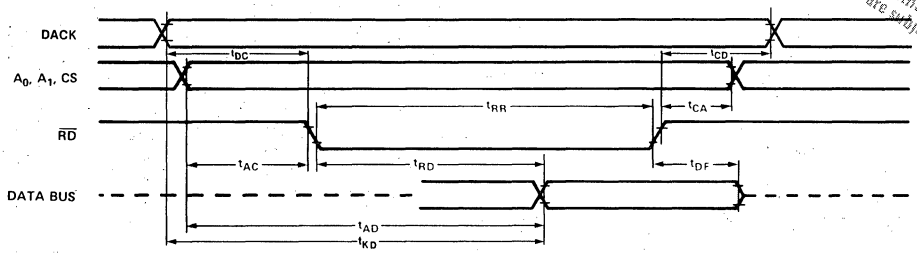
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{RSTW}	Reset Pulse Width	10		t_{CY}	
t_r	Input Signal Rise Time		20	ns	
t_f	Input Signal Fall Time		20	ns	
t_{RSTS}	Reset to First $\overline{IOWR}$	2		t_{CY}	
t_{CY}	Clock	250		ns	Note 2
t_{CL}	Clock Low	110		ns	
t_{CH}	Clock High	122		ns	
t_{DCL}	Data Clock Low	200		ns	
t_{DCH}	Data Clock High	200		ns	
t_{DCY}	Data Clock	15625		ns	Note 2
t_{TD}	Transmit Data Delay		100	ns	
t_{DS}	Data Setup Time	100		ns	
t_{DH}	Data Hold Time	0		ns	
t_{DPLL}	$\overline{DPLL}$ Output Low	200		ns	
t_{FLD}	$\overline{FLAG\ DET}$ Output Low	$8 \cdot t_{CY} \pm 50$		ns	

NOTES:

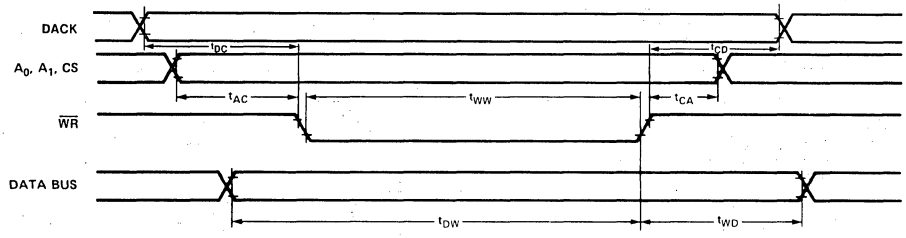
1. All timing measurements are made at the reference voltages unless otherwise specified: Input "1" at 2.0V, "0" at 0.8V
   Output "1" at 2.0V, "0" at 0.8V
2. 64K baud maximum operating rate.
3. t_{AD}, t_{RD}, t_{AC}, and t_{CA} are not concurrent specs.

WAVEFORMS

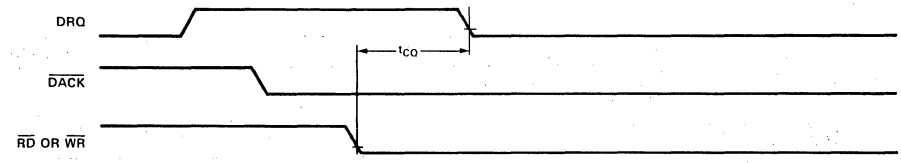
Read Waveforms

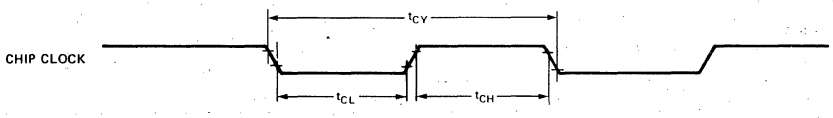


Write Waveforms

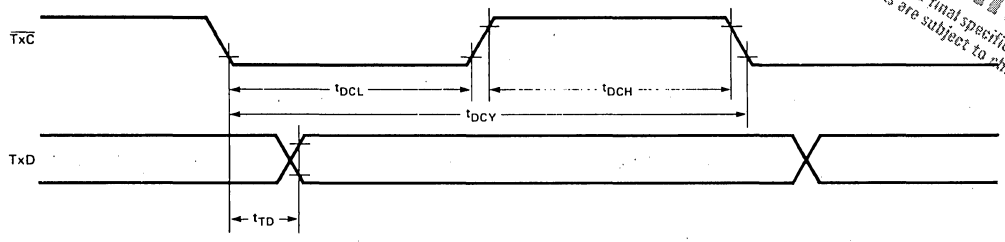


DMA Waveforms

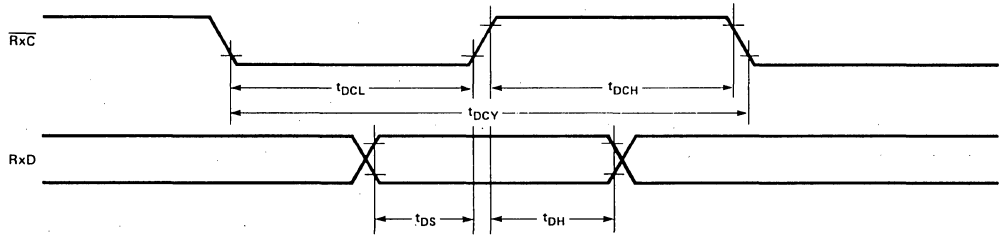




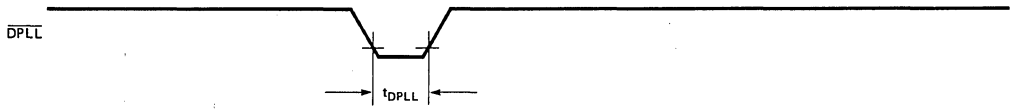
Transmit Data Waveforms



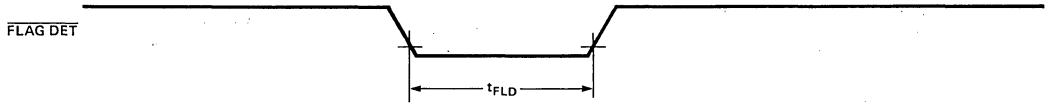
Receive Data Waveforms



DPLL Output Waveform



Flag Detect Output Waveform



intel®

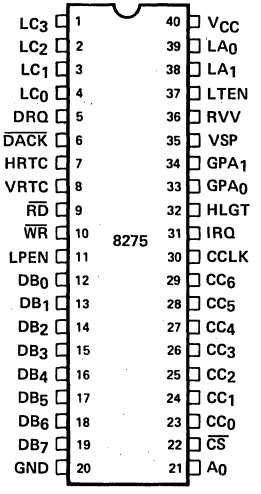
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8275
PROGRAMMABLE CRT CONTROLLER

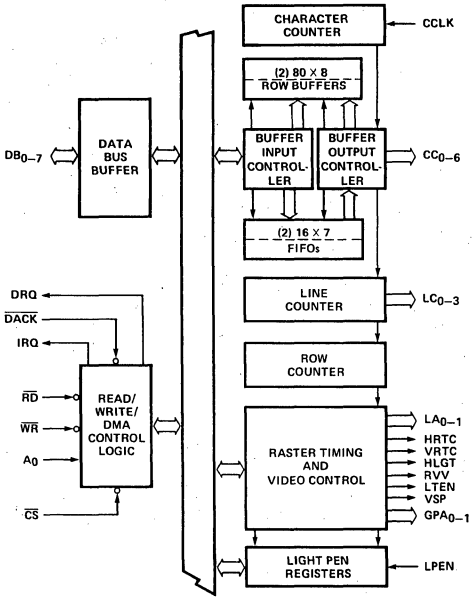
- **Programmable Screen and Character Format**
- **6 Independent Visual Field Attributes**
- **11 Visual Character Attributes (Graphic Capability)**
- **Cursor Control (4 Types)**
- **Light Pen Detection and Registers**
- **Fully MCS-80™ and MCS-85™ Compatible**
- **Dual Row Buffers**
- **Programmable DMA Burst Mode**
- **Single +5V Supply**
- **40-Pin Package**

The Intel® 8275 Programmable CRT Controller is a single chip device to interface CRT raster scan displays with Intel® microcomputer systems. Its primary function is to refresh the display by buffering the information from main memory and keeping track of the display position of the screen. The flexibility designed into the 8275 will allow simple interface to almost any raster scan CRT display with a minimum of external hardware and software overhead.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

DB_{0-1}	BI-DIRECTIONAL DATA BUS	LC_{0-3}	LINE COUNTER OUTPUTS
DRQ	DMA REQUEST OUTPUT	LA_{0-1}	LINE ATTRIBUTE OUTPUTS
$\overline{DACK}$	DMA ACKNOWLEDGE INPUT	HRTC	HORIZONTAL RETRACE OUTPUT
IRQ	INTERRUPT REQUEST OUTPUT	VRTC	VERTICAL RETRACE OUTPUT
$\overline{RD}$	READ STROBE INPUT	HLGT	HIGHLIGHT OUTPUT
$\overline{WR}$	WRITE STROBE INPUT	RVV	REVERSE VIDEO OUTPUT
A_0	REGISTER ADDRESS INPUT	LTEN	LIGHT ENABLE OUTPUT
CS	CHIP SELECT INPUT	VSP	VIDEO SUPPRESS OUTPUT
CCLK	CHARACTER CLOCK INPUT	GPA_{0-1}	GENERAL PURPOSE ATTRIBUTE OUTPUTS
CC_{0-6}	CHARACTER CODE OUTPUTS	LPEN	LIGHT PEN INPUT

PIN DESCRIPTIONS

Pin #	Pin Name	I/O	Pin Description
1 2 3 4	LC_3 LC_2 LC_1 LC_0	O	Line count. Output from the line counter which is used to address the character generator for the line positions on the screen.
5	DRQ	O	DMA request. Output signal to the 8257 DMA controller requesting a DMA cycle.
6	$\overline{DACK}$	I	DMA acknowledge. Input signal from the 8257 DMA controller acknowledging that the requested DMA cycle has been granted.
7	HRTC	O	Horizontal retrace. Output signal which is active during the programmed horizontal retrace interval. During this period the VSP output is high and the LTEN output is low.
8	VRTC	O	Vertical retrace. Output signal which is active during the programmed vertical retrace interval. During this period the VSP output is high and the LTEN output is low.
9	$\overline{RD}$	I	Read input. A control signal to read registers.
10	$\overline{WR}$	I	Write input. A control signal to write commands into the control registers or write data into the row buffers during a DMA cycle.
11	LPEN	I	Light pen. Input signal from the CRT system signifying that a light pen signal has been detected.
12 13 14 15 16 17 18 19	DB_0 DB_1 DB_2 DB_3 DB_4 DB_5 DB_6 DB_7	I/O	Bi-directional three-state data bus lines. The outputs are enabled during a read of the C or P ports.
20	Ground		Ground

Pin #	Pin Name	I/O	Pin Description
40	V_{CC}		+5V power supply
39 38	LA_0 LA_1	O	Line attribute codes. These attribute codes have to be decoded externally by the dot/timing logic to generate the horizontal and vertical line combinations for the graphic displays specified by the character attribute codes.
37	LTEN	O	Light enable. Output signal used to enable the video signal to the CRT. This output is active at the programmed underline cursor position, and at positions specified by attribute codes.
36	RVV	O	Reverse video. Output signal used to indicate the CRT circuitry to reverse the video signal. This output is active at the cursor position if a reverse video block cursor is programmed or at the positions specified by the field attribute codes.
35	VSP	O	Video suppression. Output signal used to blank the video signal to the CRT. This output is active: — during the horizontal and vertical retrace intervals. — at the top and bottom lines of rows if underline is programmed to be number 8 or greater. — when an end of row or end of screen code is detected. — When a DMA underrun occurs. — at regular intervals (1/16 frame frequency for cursor, 1/32 frame frequency for character and field attributes) — to create blinking displays as specified by cursor, character attribute, or field attribute programming.
34 33	GPA_1 GPA_0	O	General purpose attribute codes. Outputs which are enabled by the general purpose field attribute codes.
32	HLGT	O	Highlight. Output signal used to intensify the display at particular positions on the screen as specified by the character attribute codes or field attribute codes.
31	IRQ	O	Interrupt request.
30	CCLK	I	Character clock (from dot/timing logic).
29 28 27 26 25 24 23	CC_6 CC_5 CC_4 CC_3 CC_2 CC_1 CC_0	O	Character codes. Output from the row buffers used for character selection in the character generator.
22	$\overline{CS}$	I	Chip select. The read and write are enabled by $\overline{CS}$.
21	A_0	I	Port address. A high input on A_0 selects the "C" port or command registers and a low input selects the "P" port or parameter registers.

FUNCTIONAL DESCRIPTION

Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8275 to the system Data Bus.

This functional block accepts inputs from the System Control Bus and generates control signals for overall device operation. It contains the Command, Parameter, and Status Registers that store the various control formats for the device functional definition.

A_0	OPERATION	REGISTER
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

$\overline{RD}$ (Read)

A "low" on this input informs the 8275 that the CPU is reading data or status information from the 8275.

$\overline{WR}$ (Write)

A "low" on this input informs the 8275 that the CPU is writing data or control words to the 8275.

$\overline{CS}$ (Chip Select)

A "low" on this input selects the 8275. No reading or writing will occur unless the device is selected. When $\overline{CS}$ is high, the Data Bus in the float state and $\overline{RD}$ and $\overline{WR}$ will have no effect on the chip.

DRQ (DMA Request)

A "high" on this output informs the DMA Controller that the 8275 desires a DMA transfer.

$\overline{DACK}$ (DMA Acknowledge)

A "low" on this input informs the 8275 that a DMA cycle is in progress.

IRQ (Interrupt Request)

A "high" on this output informs the CPU that the 8275 desires interrupt service.

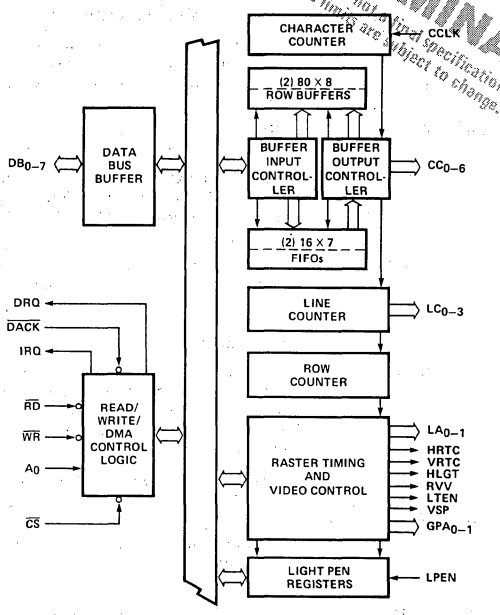


Figure 1. 8275 Block Diagram Showing Data Bus Buffer and Read/Write Functions

A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	Write 8275 Parameter
0	1	0	0	Read 8275 Parameter
1	0	1	0	Write 8275 Command
1	1	0	0	Read 8275 Status
X	1	1	0	Three-State
X	X	X	1	Three-state

Character Counter

The Character Counter is a programmable counter that is used to determine the number of characters to be displayed per row and the length of the horizontal retrace interval. It is driven by the CCLK (Character Clock) input, which should be a derivative of the external dot clock.

Line Counter

The Line Counter is a programmable counter that is used to determine the number of horizontal lines (Sweeps) per character row. Its outputs are used to address the external character generator ROM.

Row Counter

The Row Counter is a programmable counter that is used to determine the number of character rows to be displayed per frame and length of the vertical retrace interval.

Light Pen Registers

The Light Pen Registers are two registers that store the contents of the character counter and the row counter whenever there is a rising edge on the LPEN (Light Pen) input.

Note: Software correction is required.

Raster Timing and Video Controls

The Raster Timing circuitry controls the timing of the HRTC (Horizontal Retrace) and VRTC (Vertical Retrace) outputs. The Video Control circuitry controls the generation of LA_{0-1} (Line Attribute), HGLT (Highlight), RVV (Reverse Video), LTEN (Light Enable), VSP (Video Suppress), and GPA_{0-1} (General Purpose Attribute) outputs.

Row Buffers

The Row Buffers are two 80 character buffers. They are filled from the microcomputer system memory with the character codes to be displayed. While one row buffer is displaying a row of characters, the other is being filled with the next row of characters.

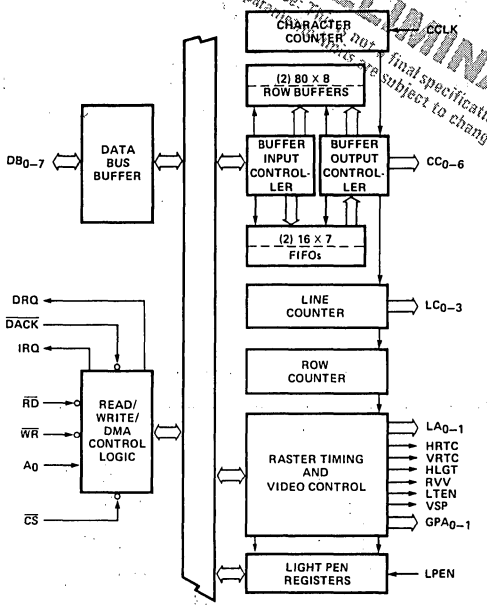


Figure 2. 8275 Block Diagram Showing Counter and Register Functions

FIFOs

There are two 16 character FIFOs in the 8275. They are used to provide extra row buffer length in the Transparent Attribute Mode (see Detailed Operation section).

Buffer Input/Output Controllers

The Buffer Input/Output Controllers decode the characters being placed in the row buffers. If the character is a character attribute, field attribute or special code, these controllers control the appropriate action. (Examples: An "End of Screen–Stop DMA" special code will cause the Buffer Input Controller to stop further DMA requests. A "Highlight" field attribute will cause the Buffer Output Controller to activate the HGLT output.)

SYSTEM OPERATION

The 8275 is programmable to a large number of different display formats. It provides raster timing, display row buffering, visual attribute decoding, cursor timing, and light pen detection.

It is designed to interface with the 8257 DMA Controller and standard character generator ROMs for dot matrix decoding. Dot level timing must be provided by external circuitry.

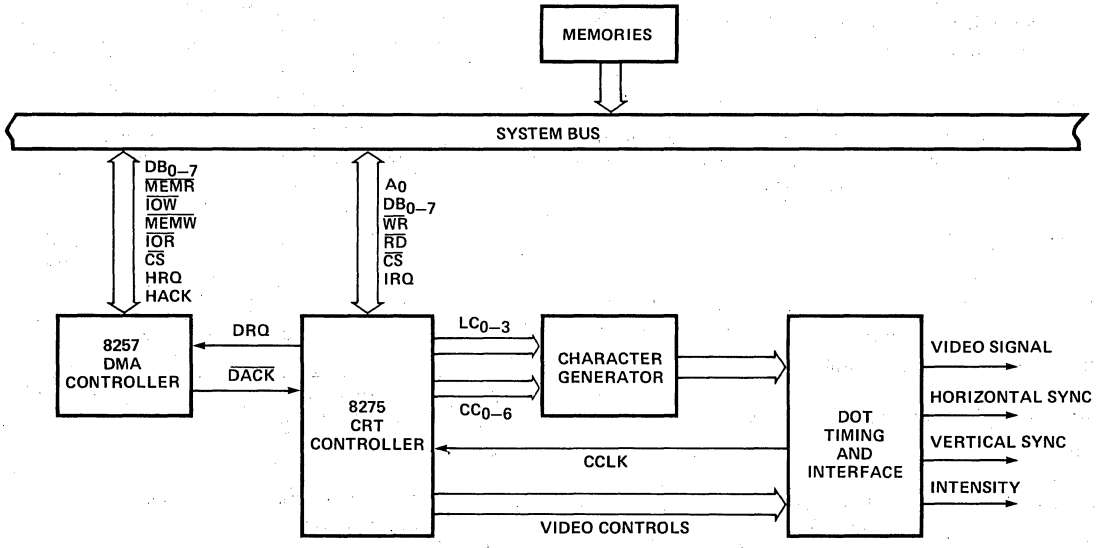


Figure 3. 8275 Systems Block Diagram Showing Systems Operation

General Systems Operational Description

The 8275 provides a "window" into the microcomputer system memory.

Display characters are retrieved from memory and displayed on a row by row basis. The 8275 has two row buffers. While one row buffer is being used for display, the other is being filled with the next row of characters to be displayed. The number of display characters per row and the number of character rows per frame are software programmable, providing easy interface to most CRT displays. (See Programming Section.)

The 8275 requests DMA to fill the row buffer that is not being used for display. DMA burst length and spacing is programmable. (See Programming Section.)

The 8275 displays character rows one line at a time.

The number of lines per character row, the underline position, and blanking of top and bottom lines are programmable. (See Programming Section.)

The 8275 provides special Control Codes which can be used to minimize DMA or software overhead. It also provides Visual Attribute Codes to cause special action or symbols on the screen without the use of the character generator (see Visual Attributes Section).

The 8275 also controls raster timing. This is done by generating Horizontal Retrace (HRTC) and Vertical Retrace (VRTC) signals. The timing of these signals is programmable.

The 8275 can generate a cursor. Cursor location and format are programmable. (See Programming Section.)

The 8275 has a light pen input and registers. The light pen input is used to load the registers. Light pen registers can be read on command. (See Programming Section.)

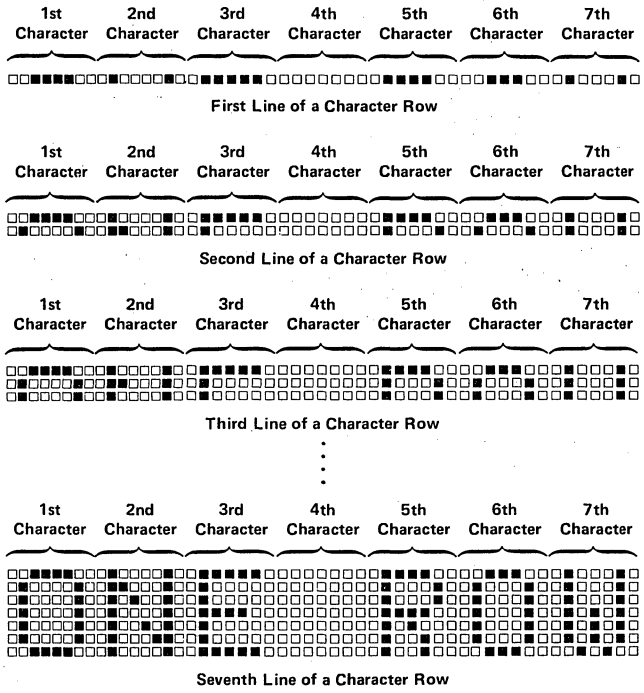


Figure 4. Display of a Character Row

Display Row Buffering

Before the start of a frame, the 8275 requests DMA and one row buffer is filled with characters.

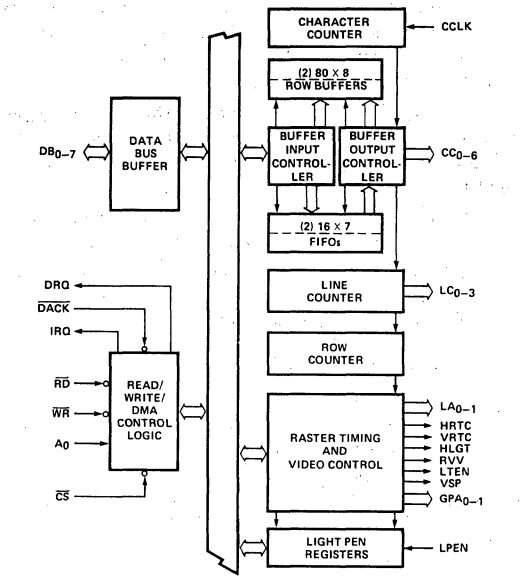


Figure 5. First Row Buffer Filled

When the first horizontal sweep is started, character codes are output to the character generator from the row buffer just filled. Simultaneously, DMA begins filling the other row buffer with the next row of characters.

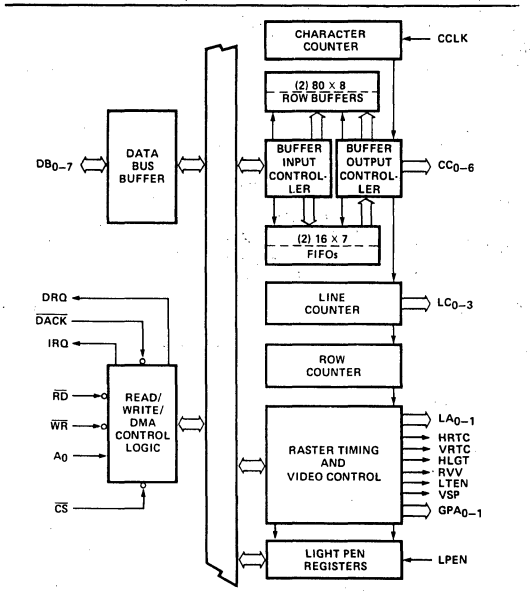


Figure 6. Second Buffer Filled, First Row Displayed

After all the lines of the character row are scanned, the roles of the two row buffers are reversed and the same procedure is followed for the next row.

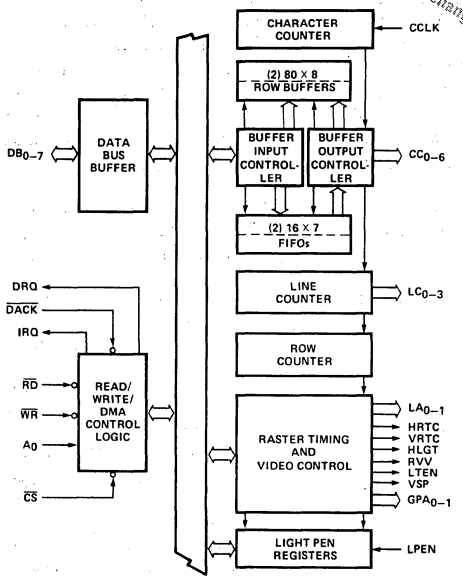


Figure 7. First Buffer Filled with Third Row, Second Row Displayed

This is repeated until all of the character rows are displayed.

Display Format

Screen Format

The 8275 can be programmed to generate from 1 to 80 characters per row, and from 1 to 64 rows per frame.

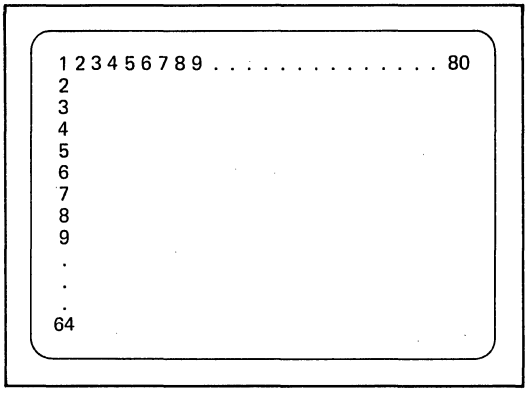


Figure 8. Screen Format

The 8275 can also be programmed to blank alternate rows. In this mode, the first row is displayed, the second blanked, the third displayed, etc. DMA is not requested for the blanked rows.

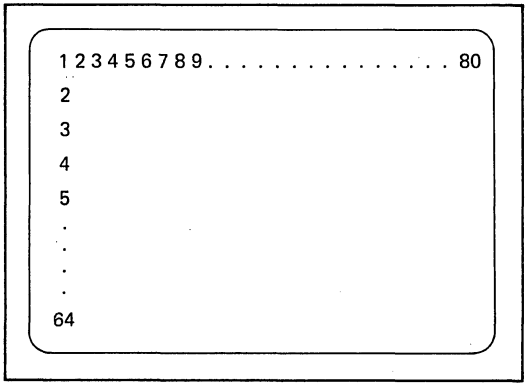


Figure 9. Blank Alternate Rows Mode

Row Format

The 8275 is designed to hold the line count stable while outputting the appropriate character codes during each horizontal sweep. The line count is incremented during horizontal retrace and the whole row of character codes are output again during the next sweep. This is continued until the whole character row is displayed.

The number of lines (horizontal sweeps) per character row is programmable from 1 to 16.

The output of the line counter can be programmed to be in one of two modes.

In mode 0, the output of the line *counter* is the same as the line *number*.

In mode 1, the line *counter* is offset by one from the line *number*.

Note: In mode 1, while the *first* line (line number 0) is being displayed, the *last* count is output by the line counter (see examples).

Line Number										Line Counter Mode 0	Line Counter Mode 1
0	□	□	□	□	□	□	□	□	□	0000	1111
1	□	□	□	□	■	□	□	□	□	0001	0000
2	□	□	□	■	□	■	□	□	□	0010	0001
3	□	□	■	□	□	□	■	□	□	0011	0010
4	□	■	□	□	□	□	□	■	□	0100	0011
5	□	■	□	□	□	□	□	■	□	0101	0100
6	□	■	■	■	■	■	■	■	□	0110	0101
7	□	■	□	□	□	□	□	■	□	0111	0110
8	□	■	□	□	□	□	□	■	□	1000	0111
9	□	■	□	□	□	□	□	■	□	1001	1000
10	□	□	□	□	□	□	□	□	□	1010	1001
11	□	□	□	□	□	□	□	□	□	1011	1010
12	□	□	□	□	□	□	□	□	□	1100	1011
13	□	□	□	□	□	□	□	□	□	1101	1100
14	□	□	□	□	□	□	□	□	□	1110	1101
15	□	□	□	□	□	□	□	□	□	1111	1110

Figure 10. Example of a 16-Line Format

Line Number								Line Counter Mode 0	Line Counter Mode 1
0	□	□	□	□	□	□	□	0000	1001
1	□	□	□	■	□	□	□	0001	0000
2	□	□	■	□	■	□	□	0010	0001
3	□	■	□	□	□	■	□	0011	0010
4	□	■	□	□	□	■	□	0100	0011
5	□	■	■	■	■	■	□	0101	0100
6	□	■	□	□	□	■	□	0110	0101
7	□	■	□	□	□	■	□	0111	0110
8	□	□	□	□	□	□	□	1000	0111
9	□	□	□	□	□	□	□	1001	1000

Figure 11. Example of a 10-Line Format

Mode 0 is useful for character generators that leave address zero blank and start at address 1. Mode 1 is useful for character generators which start at address zero.

Underline placement is also programmable (from line *number* 0 to 15). This is independent of the line *counter* mode.

If the line *number* of the underline is greater than 7 (line *number* MSB = 1), then the top and bottom lines will be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □ □	0000	1011
1	□ □ □ □ ■ □ □ □ □	0001	0000
2	□ □ □ ■ □ ■ □ □ □	0010	0001
3	□ □ ■ □ □ □ ■ □ □	0011	0010
4	□ ■ □ □ □ □ □ ■ □	0100	0011
5	□ ■ □ □ □ □ □ ■ □	0101	0100
6	□ ■ ■ ■ ■ ■ ■ ■ □	0110	0101
7	□ ■ □ □ □ □ □ ■ □	0111	0110
8	□ ■ □ □ □ □ □ ■ □	1000	0111
9	□ ■ □ □ □ □ □ ■ □	1001	1000
10	■ ■ ■ ■ ■ ■ ■ ■ ■	1010	1001
11	□ □ □ □ □ □ □ □ □	1011	1010

Top and Bottom Lines are Blanked

Figure 12. Underline in Line Number 10

If the line *number* of the underline is less than or equal to 7 (line *number* MSB = 0), then the top and bottom lines will *not* be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ ■ □ □ □	0000	0111
1	□ □ ■ □ ■ □ □	0001	0000
2	□ ■ □ □ □ ■ □	0010	0001
3	□ ■ □ □ □ ■ □	0011	0010
4	□ ■ ■ ■ ■ ■ □	0100	0011
5	□ ■ □ □ □ ■ □	0101	0100
6	□ ■ □ □ □ ■ □	0110	0101
7	■ ■ ■ ■ ■ ■ ■	0111	0110

Top and Bottom Lines are not Blanked

Figure 13. Underline in Line Number 7

If the line *number* of the underline is greater than the maximum number of lines, the underline will not appear.

Blanking is accomplished by the VSP (Video Suppression) signal. Underline is accomplished by the LTEN (Light Enable) signal.

Dot Format

Dot width and character width are dependent upon the external timing and control circuitry.

Dot level timing circuitry should be designed to accept the parallel output of the character generator and shift it out serially at the rate required by the CRT display.

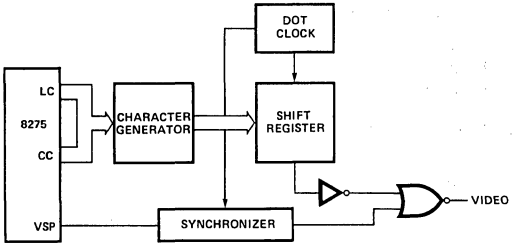


Figure 14. Typical Dot Level Block Diagram

Dot width is a function of dot clock frequency.

Character width is a function of the character generator width.

Horizontal character spacing is a function of the shift register length.

Note: Video control and timing signals must be synchronized with the video signal due to the character generator access delay.

Raster Timing

The character counter is driven by the character clock input (CCLK). It counts out the characters being displayed (programmable from 1 to 80). It then causes the line counter to increment, and it starts counting out the horizontal retrace interval (programmable from 2 to 32). This is constantly repeated.

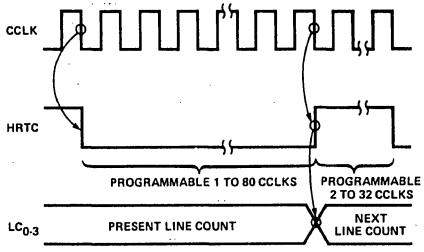


Figure 15. Line Timing

The line counter is driven by the character counter. It is used to generate the line address outputs (LC_{0-3}) for the character generator. After it counts all of the lines in a character row (programmable from 1 to 16), it increments the row counter, and starts over again. (See Character Format Section for detailed description of Line Counter functions.)

The row counter is an internal counter driven by the line counter. It controls the functions of the row buffers and counts the number of character rows displayed.

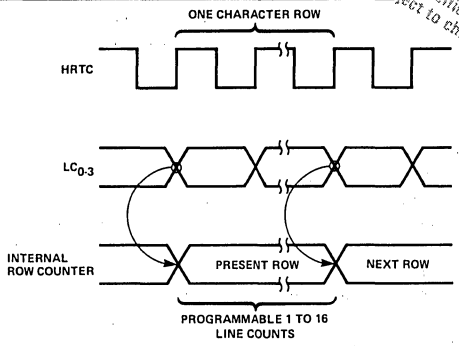


Figure 16. Row Timing

After the row counter counts all of the rows in a frame (programmable from 1 to 64), it starts counting out the vertical retrace interval (programmable from 1 to 4).

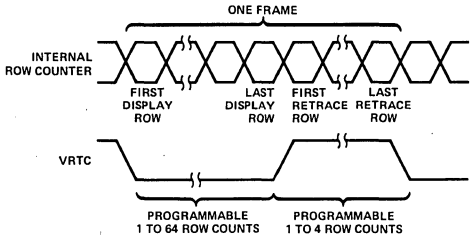


Figure 17. Frame Timing

The Video Suppression Output (VSP) is active during horizontal and vertical retrace intervals.

Dot level timing circuitry must synchronize these outputs with the video signal to the CRT Display.

DMA Timing

The 8275 can be programmed to request burst DMA transfers of 1 to 8 characters. The interval between bursts is also programmable (from 0 to 55 character clock periods ±1). This allows the user to tailor his DMA overhead to fit his system needs.

The first DMA request of the frame occurs one *row time* before the end of vertical retrace. DMA requests continue as programmed, until the row buffer is filled. If the row buffer is filled in the middle of a burst, the 8275 terminates the burst and resets the burst counter. No more DMA requests will occur until the *beginning* of the *next* row. At that time, DMA requests are activated as programmed until the other buffer is filled.

If, for any reason, there is a DMA underrun, a flag in the status word will be set.

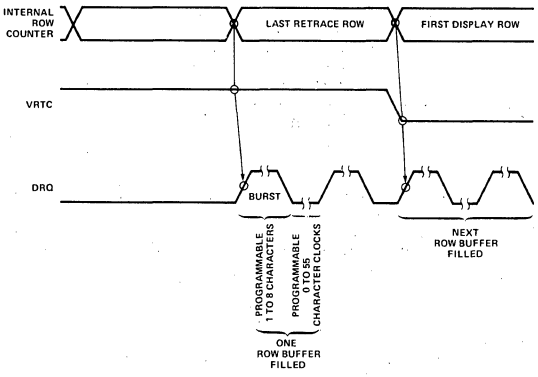


Figure 18. DMA Timing

The DMA controller is typically initialized for the next frame at the end of the current frame.

Interrupt Timing

The 8275 can be programmed to generate an interrupt request at the end of each frame. This can be used to reinitialize the DMA controller. If the 8275 interrupt enable flag is set, an interrupt request will occur at the *beginning* of the *last display row.*

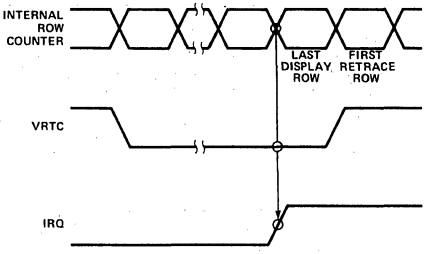


Figure 19. Beginning of Interrupt Request

IRQ will go inactive after the status register is read.

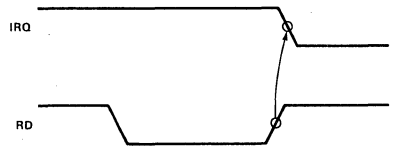


Figure 20. End of Interrupt Request

A reset command will also cause IRQ to go inactive, but this is not recommended during normal service.

Another method of reinitializing the DMA controller is to have the DMA controller itself interrupt on terminal count. With this method, the 8275 interrupt enable flag should not be set.

Note: Upon power-up, the 8275 Interrupt Enable Flag may be set. As a result, the user's cold start routine should write a reset command to the 8275 before system interrupts are enabled.

VISUAL ATTRIBUTES AND SPECIAL CODES

The characters processed by the 8275 are 8-bit quantities. The character code outputs provide the character generator with 7 bits of address. The Most Significant Bit is the extra bit and it is used to determine if it is a normal display character (MSB = 0), or if it is a Visual Attribute or Special Code (MSB = 1).

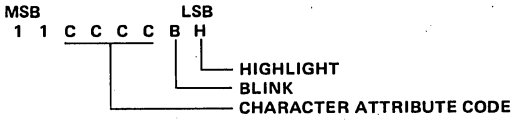
There are two types of Visual Attribute Codes. They are Character Attributes and Field Attributes.

Character Attribute Codes

Character attribute codes are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA_{0-1}), the Video Suppression output (VSP), and the Light Enable output. The dot level timing circuitry can use these signals to generate the proper symbols.

Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT).

Character Attributes



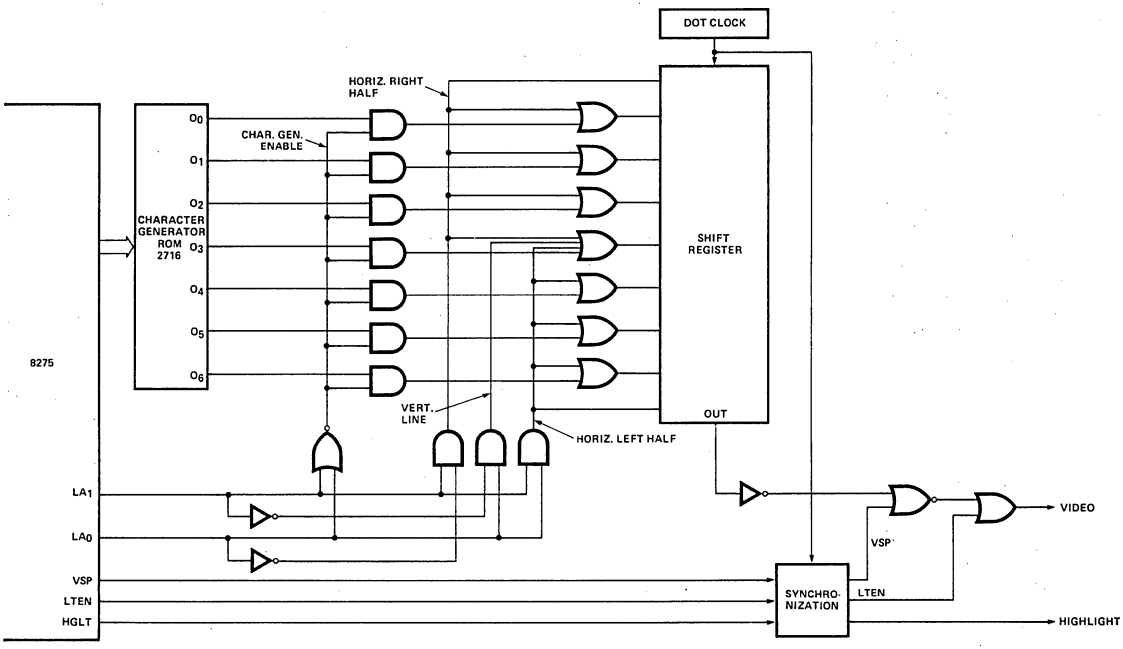


Figure 21. Typical Character Attribute Logic

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"		OUTPUTS LA_1	LA_0	VSP	LTEN	SYMBOL	DESCRIPTION
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

Blinking is active when B = 1.

Highlight is active when H = 1.

Special Codes

Four special codes are available to help reduce memory, software, or DMA overhead.

Special Control Character

```
MSB                 LSB
1 1 1 1   0 0 S S
                  └──── SPECIAL CONTROL CODE
```

S S	FUNCTION
0 0	End of Row
0 1	End of Row-Stop DMA
1 0	End of Screen
1 1	End of Screen-Stop DMA

The End of Row Code (00) activates VSP and holds it to the end of the line.

The End of Row-Stop DMA Code (01) causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the Row Buffer. It affects the display in the same way as the End of Row Code (00).

The End of Screen Code (10) activates VSP and holds it to the end of the frame.

The End of Screen-Stop DMA Code (11) causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the Row Buffer. It affects the display in the same way as the End of Screen Code (10).

If the Stop DMA feature is not used, all characters after an End of Row character are ignored, except for the End of Screen character, which operates normally. All characters after an End of Screen character are ignored.

Note: If a Stop DMA character is not the last character in a burst or row, DMA is not stopped until after the next character is read. In this situation, a dummy character must be placed in memory after the Stop DMA character.

Field Attributes

The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the code up to, and including, the character which precedes the *next* field attribute code, or up to the end of the frame. The field attributes are reset during the vertical retrace interval.

There are six field attributes:

1. *Blink* — Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. *Highlight* — Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. *Reverse Video* — Characters following the code are caused to appear with reverse video by activating the Reverse Video output (RVV).
4. *Underline* — Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).

5,6. *General Purpose* — There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. GPA_{0-1} are active high outputs.

Field Attribute Code

```
MSB                 LSB
1 O U R G G B H
                    └──── HIGHLIGHT
                  └────── BLINK
              └──────────── GENERAL PURPOSE
            └──────────────── REVERSE VIDEO
          └────────────────── UNDERLINE
```

H = 1 FOR HIGHLIGHTING
B = 1 FOR BLINKING
R = 1 FOR REVERSE VIDEO
U = 1 FOR UNDERLINE
GG = GPA_1, GPA_0

The 8275 can be programmed to provide visible or invisible field attribute characters.

If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character.

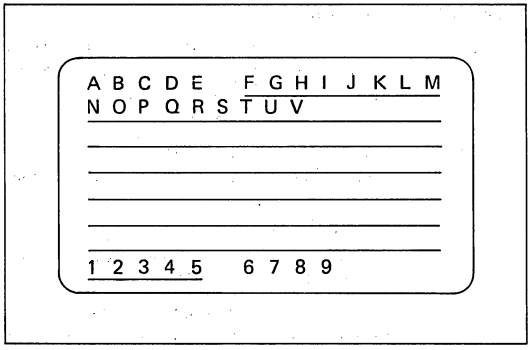


Figure 22. Example of the Visible Field Attribute Mode (Underline Attribute)

If the 8275 is programmed in the invisible field attribute mode, the 8275 FIFO is activated.

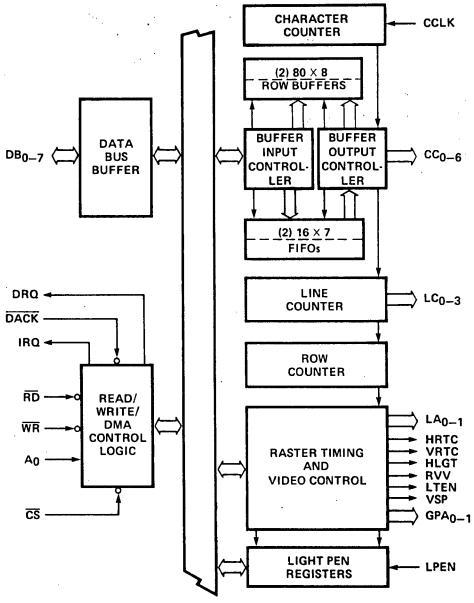


Figure 23. Block Diagram Showing FIFO Activation

Each row buffer has a corresponding FIFO. These FIFOs are 16 characters by 7 bits in size.

When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the *next* character in the proper FIFO.

When a field attribute is placed in the Buffer Output Controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CC_{0-6}). The chosen Visual Attributes are also activated.

Since the FIFO is 16 characters long, no more than 16 field attribute characters may be used per line in this mode. If more are used, a bit in the status word is set and the first characters in the FIFO are written over and lost.

Note: Since the FIFO is 7 bits wide, the MSB of any characters put in it are stripped off. Therefore, a Visual Attribute or Special Code must *not* immediately follow a field attribute code. If this situation does occur, the Visual Attribute or Special Code will be treated as a normal display character.

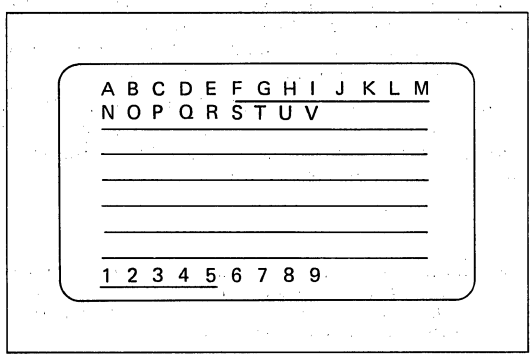


Figure 24. Example of the Invisible Field Attribute Mode (Underline Attribute)

Field and Character Attribute Interaction

Character Attribute Symbols are affected by the Reverse Video (RRV) and General Purpose (GPA_{0-1}) field attributes. They are not affected by Underline, Blink or Highlight field attributes; however, these characteristics can be programmed *individually* for Character Attribute Symbols.

Cursor Timing

The cursor location is determined by a cursor row register and a character position register which are loaded by command to the controller. The cursor can be programmed to appear on the display as:

1. a blinking underline
2. a blinking reverse video block
3. a non-blinking underline
4. a non-blinking reverse video block

The cursor blinking frequency is equal to the screen refresh frequency divided by 16.

If a non-blinking reverse video *cursor* appears in a non-blinking reverse video *field*, the cursor will appear as a normal video block.

If a non-blinking underline *cursor* appears in a non-blinking underline *field*, the cursor will not be visible.

Light Pen Detection

A light pen consists of a micro switch and a tiny light sensor. When the light pen is pressed against the CRT screen, the micro switch enables the light sensor. When the raster sweep reaches the light sensor, it triggers the light pen output.

If the output of the light pen is presented to the 8275 LPEN input, the row and character position coordinates are stored in a pair of registers. These registers can be read on command. A bit in the status word is set, indicating that the light pen signal was detected. The LPEN input must be a 0 to 1 transition for proper operation.

Note: Due to internal and external delays, the character position coordinate will be off by at least three character positions. This has to be corrected in software.

Device Programming

The 8275 has two programming registers, the Command Register (CREG) and the Parameter Register (PREG). It also has a Status Register (SREG). The Command Register can only be written into and the Status Registers can only be read from. They are addressed as follows:

A_0	OPERATION	REGISTER
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

The 8275 expects to receive a command and a sequence of 0 to 4 parameters, depending on the command. If the proper number of parameter bytes are not received before another command is given, a status flag is set, indicating an improper command.

Instruction Set

The 8275 instruction set consists of 8 commands.

COMMAND	NO. OF PARAMETER BYTES
Reset	4
Start Display	0
Stop Display	0
Read Light Pen	2
Load Cursor	2
Enable Interrupt	0
Disable Interrupt	0
Preset Counters	0

In addition, the status of the 8275 (SREG) can be read by the CPU at any time.

1. Reset Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Reset Command	0	0	0	0	0	0	0	0
Parameters	Write	0	Screen Comp Byte 1	S	H	H	H	H	H	H	H
	Write	0	Screen Comp Byte 2	V	V	R	R	R	R	R	R
	Write	0	Screen Comp Byte 3	U	U	U	U	L	L	L	L
	Write	0	Screen Comp Byte 4	M	F	C	C	Z	Z	Z	Z

Action – After the reset command is written, DMA requests stop, 8275 interrupts are disabled, and the VSP output is used to blank the screen. HRTC and VRTC continue to run. HRTC and VRTC timing are random on power-up.

As parameters are written, the screen composition is defined.

Parameter – S Spaced Rows

S	FUNCTIONS
0	Normal Rows
1	Spaced Rows

Parameter – HHHHHHH Horizontal Characters/Row

H	H	H	H	H	H	H	NO. OF CHARACTERS PER ROW
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	2
0	0	0	0	0	1	0	3
			.				.
			.				.
			.				.
1	0	0	1	1	1	1	80
1	0	1	0	0	0	0	Undefined
			.				.
			.				.
			.				.
1	1	1	1	1	1	1	Undefined

Parameter – VV Vertical Retrace Row Count

V	V	NO. OF ROW COUNTS PER VRTC
0	0	1
0	1	2
1	0	3
1	1	4

Parameter – RRRRRR Vertical Rows/Frame

R	R	R	R	R	R	NO. OF ROWS/FRAME
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
			.			.
			.			.
			.			.
1	1	1	1	1	1	64

Parameter – UUUU Underline Placement

U	U	U	U	LINE NUMBER OF UNDERLINE
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
		.		.
		.		.
		.		.
1	1	1	1	16

Parameter – LLLL Number of Lines per Character Row

L	L	L	L	NO. OF LINES/ROW
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
		.		.
		.		.
		.		.
1	1	1	1	16

Parameter – M Line Counter Mode

M	LINE COUNTER MODE
0	Mode 0 (Non-Offset)
1	Mode 1 (Offset by 1 Count)

Parameter – F Field Attribute Mode

F	FIELD ATTRIBUTE MODE
0	Transparent
1	Non-Transparent

Parameter – CC Cursor Format

C	C	CURSOR FORMAT
0	0	Blinking reverse video block
0	1	Blinking underline
1	0	Nonblinking reverse video block
1	1	Nonblinking underling

Parameter – ZZZZ Horizontal Retrace Count

Z	Z	Z	Z	NO. OF CHARACTER COUNTS PER HRTC
0	0	0	0	2
0	0	0	1	4
0	0	1	0	6
		.		.
		.		.
		.		.
1	1	1	1	32

Note: uuuu MSB determines blanking of top and bottom lines (1 = blanked, 0 = not blanked).

2. Start Display Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Start Display	0	0	1	S	S	S	B	B
No parameters											

S S S BURST SPACE CODE

S S S	NO. OF CHARACTER CLOCKS BETWEEN DMA REQUESTS
0 0 0	0
0 0 1	7
0 1 0	15
0 1 1	23
1 0 0	31
1 0 1	39
1 1 0	47
1 1 1	55

B B BURST COUNT CODE

B B	NO. OF DMA CYCLES PER BURST
0 0	1
0 1	2
1 0	4
1 1	8

Action — 8275 interrupts are enabled, DMA requests begin, video is enabled, Interrupt Enable and Video Enable status flags are set.

3. Stop Display Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Stop Display	0	1	0	0	0	0	0	0
No parameters											

Action — Disables video, interrupts remain enabled, HRTC and VRTC continue to run, Video Enable status flag is reset, and the "Start Display" command must be given to re-enable the display.

4. Read Light Pen Command

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Read Light Pen	0	1	1	0	0	0	0	0
Parameters	Read	0	Char. Number	(Char. Position in Row)							
	Read	0	Row Number	(Row Number)							

Action — The 8275 is conditioned to supply the contents of the light pen position registers in the next two read cycles of the parameter register. Status flags are not affected.

Note: Software correction of light pen position is required.

5. Load Cursor Position:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Load Cursor	1	0	0	0	0	0	0	0
Parameters	Write	0	Char. Number	(Char. Position in Row)							
	Write	0	Row Number	(Row Number)							

Action — The 8275 is conditioned to place the next two parameter bytes into the cursor position registers. Status flags not affected.

6. Enable Interrupt Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Enable Interrupt	1	0	1	0	0	0	0	0
No parameters											

Action — The interrupt enable status flag is set and interrupts are enabled.

7. Disable Interrupt Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Disable Interrupt	1	1	0	0	0	0	0	0
No parameters											

Action — Interrupts are disabled and the interrupt enable status flag is reset.

8. Preset Counters Command:

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB							LSB
Command	Write	1	Preset Counters	1	1	1	0	0	0	0	0
No parameters											

Action — The internal timing counters are preset, corresponding to a screen display position at the top left corner. Two character clocks are required for this operation. The counters will remain in this state until any other command is given.

This command is useful for system debug and synchronization of clustered CRT displays on a single CPU.

Status Flags

	OPERATION	A_0	DESCRIPTION	DATA BUS MSB LSB
Command	Read	1	Status Word	0 IE IR LP IC VE DU FO

IE – (Interrupt Enable) Set or reset by command. It enables vertical retrace interrupt. It is automatically set by a "Start Display" command and reset with the "Reset" command.

IR – (Interrupt Request) This flag is set at the beginning of display of the last row of the frame if the interrupt enable flag is set. It is reset after a status read operation.

LP – This flag is set when the light pen input (LPEN) is activated and the light pen registers have been loaded. This flag is automatically reset after a status read.

IC – (Improper Command) This flag is set when a command parameter string is too long or too short. The flag is automatically reset after a status read.

VE – (Video Enable) This flag indicates that video operation of the CRT is enabled. This flag is set on a "Start Display" command, and reset on a "Stop Display" or "Reset" command.

DU – (DMA Underrun) This flag is set whenever a data underrun occurs during DMA transfers. Upon detection of DU, the DMA operation is stopped and the screen is blanked until after the vertical retrace interval. This flag is reset after a status read.

FO – (FIFO Overrun) This flag is set whenever the FIFO is overrun. It is reset on a status read.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . -65°C to +150°C
Voltage On Any Pin
With Respect to Ground . . . . . -0.5V to +7V
Power Dissipation . . . . . 1 Watt

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5V ±5%

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5V	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400 μA
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V
I_{CC}	V_{CC} Supply Current		160	mA	

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

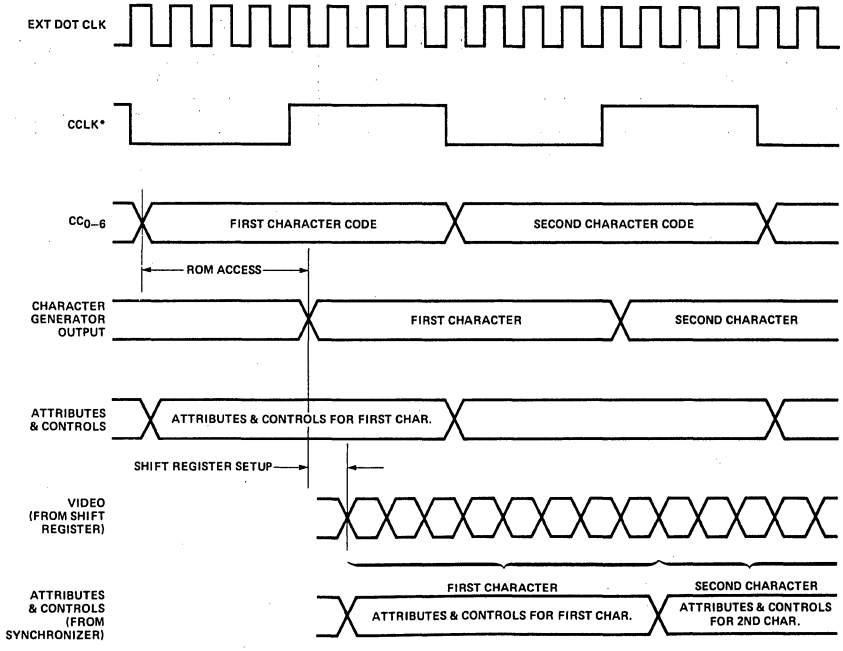
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to V_{SS}.

Other Timing:

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_{CC}	Character Code Output Delay		150	ns	C_L = 50 pF
t_{HR}	Horizontal Retrace Output Delay		150	ns	C_L = 50 pF
t_{LC}	Line Count Output Delay		250	ns	C_L = 50 pF
t_{AT}	Control/Attribute Output Delay		250	ns	C_L = 50 pF
t_{VR}	Vertical Retrace Output Delay		250	ns	C_L = 50 pF
t_{IR}	IRQ↑ from CCLK↓		250	ns	C_L = 50 pF
t_{RI}	IRQ↓ from Rd↑		250	ns	C_L = 50 pF
t_{KQ}	DRQ↑ from CCLK↓		250	ns	C_L = 50 pF
t_{WQ}	DRQ↑ from WR↑		250	ns	C_L = 50 pF
t_{RQ}	DRQ↓ from WR↓		200	ns	C_L = 50 pF
t_{LR}	DACK↓ to WR↓	0		ns	
t_{RL}	WR↑ to DACK↑	0		ns	
t_{PR}	LPEN Rise		50	ns	
t_{PH}	LPEN Hold	100		ns	

Note: Timing measurements are made at the following reference voltages: Output "1" = 2.0V, "0" = 0.8V.

WAVEFORMS



*CCLK IS A MULTIPLE OF THE DOT CLOCK AND AN INPUT TO THE 8275.

Figure 25. Typical Dot Level Timing

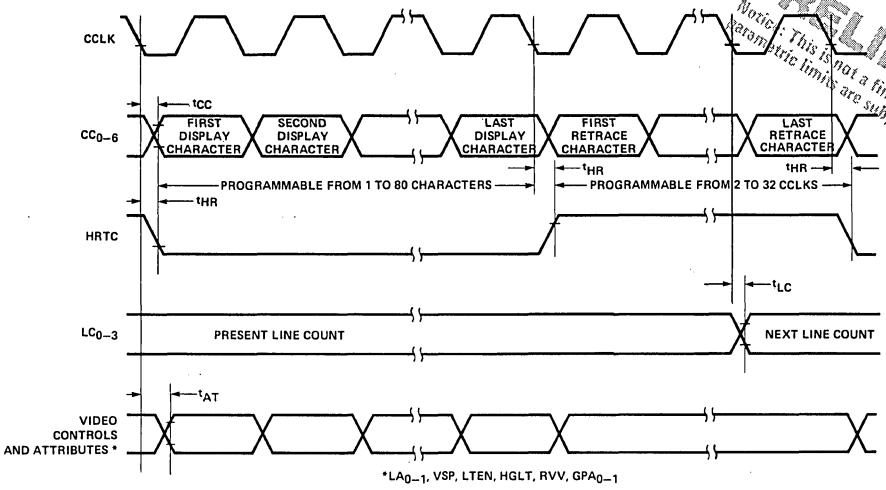


Figure 26. Line Timing

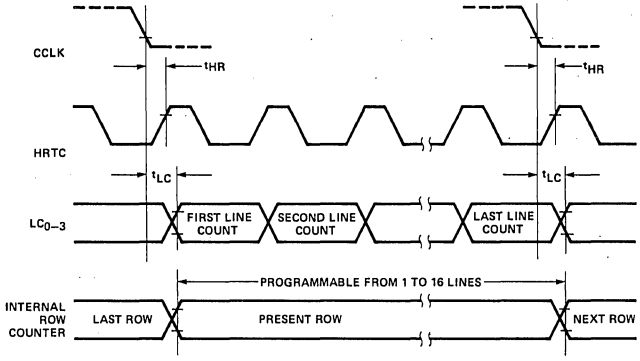


Figure 27. Row Timing

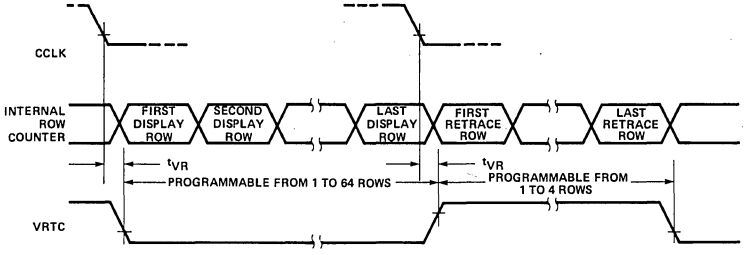


Figure 28. Frame Timing

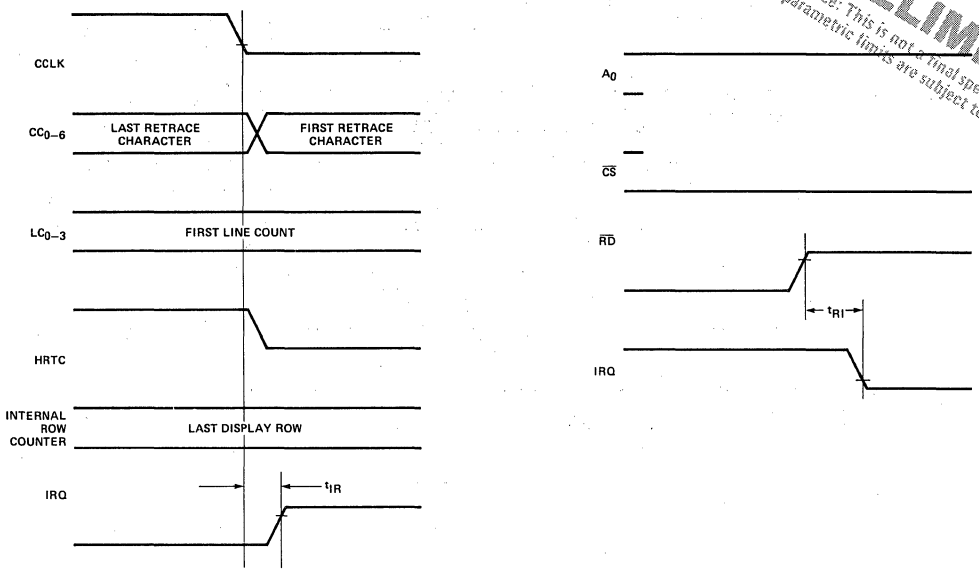


Figure 29. Interrupt Timing

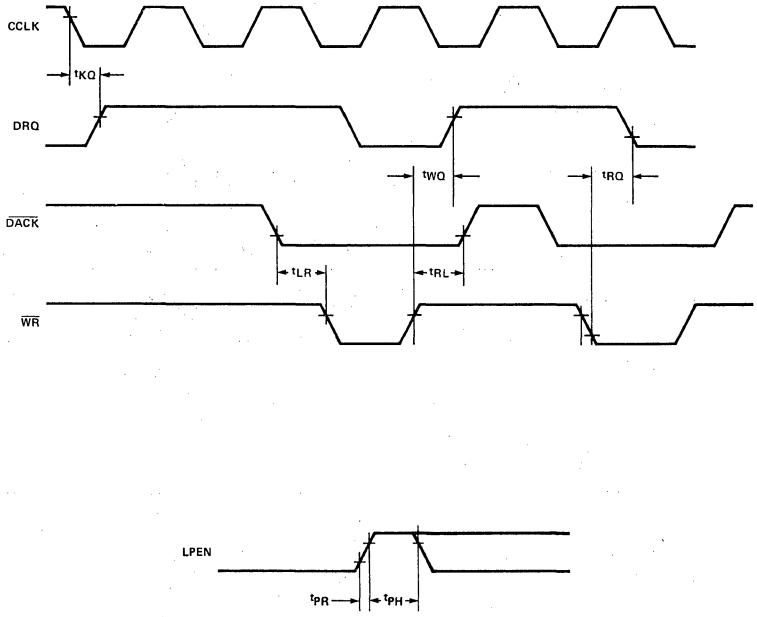


Figure 30. DMA Timing

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$; $V_{CC} = 5.0V \pm 5\%$; GND = 0V

Bus Parameters (Note 1)

Read Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_{AR}	Address Stable Before READ	0		ns	
t_{RA}	Address Hold Time for READ	0		ns	
t_{RR}	READ Pulse Width	250		ns	
t_{RD}	Data Delay from READ		200	ns	$C_L = 150$ pF
t_{DF}	READ to Data Floating	20	100	ns	

Write Cycle:

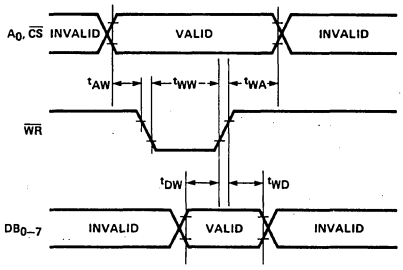
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_{AW}	Address Stable Before WRITE	0		ns	
t_{WA}	Address Hold Time for WRITE	0		ns	
t_{WW}	WRITE Pulse Width	250		ns	
t_{DW}	Data Setup Time for WRITE	150		ns	
t_{WD}	Data Hold Time for WRITE	0		ns	

Clock Timing:

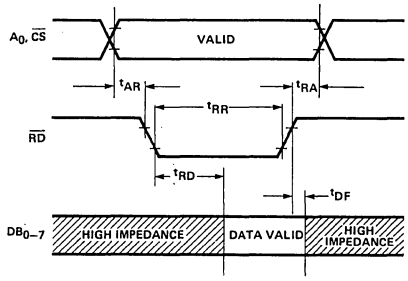
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_{CLK}	Clock Period	320		ns	
t_{KH}	Clock High	120		ns	
t_{KL}	Clock Low	120		ns	
t_{KR}	Clock Rise	5	30	ns	
t_{KF}	Clock Fall	5	30	ns	

Note 1: AC timings measured at $V_{OH} = 2.0$, $V_{OL} = 0.8$

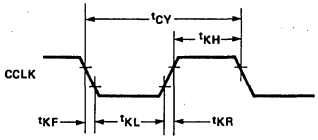
Write Timing



Read Timing



Clock Timing



Input Waveforms (For A.C. Tests)

intel®

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

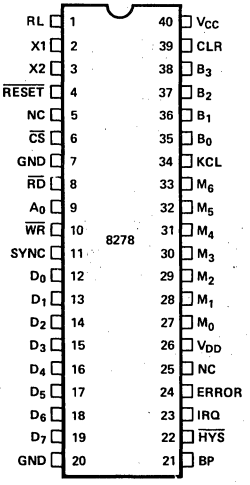
8278
PROGRAMMABLE KEYBOARD INTERFACE

- **Simultaneous Keyboard and Display Operations**
- **Interface Signals for Contact and Capacitive Coupled Keyboards**
- **128-Key Scanning Logic**
- **10.7 msec Matrix Scan Time for 128 Keys and 6 MHz Clock**
- **8-Character Keyboard FIFO**
- **N-Key Rollover with Programmable Error Mode on Multiple New Closures**
- **16-Character 7-Segment Display Interface**
- **Right or Left Entry Display RAM**
- **Depress/Release Mode Programmable**
- **Interrupt Output on Key Entry**

The Intel® 8278 is a general purpose programmable keyboard and display interface device designed for use with 8-bit microprocessors such the MDS-80™ and MCS-85™. The keyboard portion can provide a scanned interface to 128-key contact or capacitive-coupled keyboards. The keys are fully debounced with N-key rollover and programmable error generation on multiple new key closures. Keyboard entries are stored in an 8-character FIFO with overrun status indication when more than 8 characters are entered. Key entries set an interrupt request output to the master CPU.

The display portion of the 8278 provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric displays and simple indicators may be used. The 8278 has a 16X4 display RAM which can be loaded or interrogated by the CPU. Both right entry calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

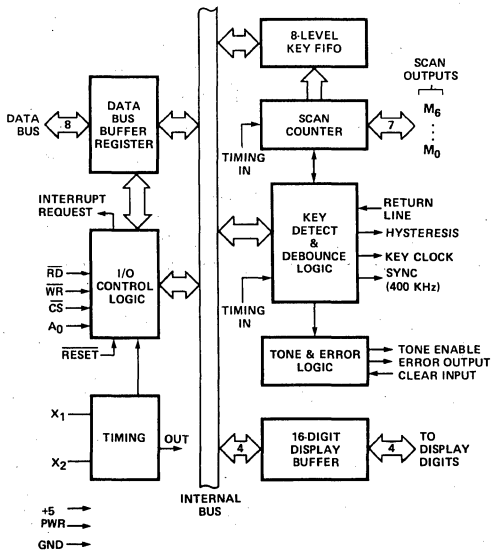
PIN CONFIGURATION



PIN NAMES

D_7–D_0	DATA BUS
$\overline{RD}$, $\overline{WR}$	READ, WRITE STROBES
$\overline{CS}$	CHIP SELECT
A_0	CONTROL/DATA SELECT
$\overline{RESET}$	RESET INPUT
X_1, X_2	FREQ. REFERENCE INPUT
SYNC	HIGH FREQUENCY OUTPUT CLOCK
RL	KEYBOARD RETURN LINE
CLR	CLEAR ERROR
KCL	KEY CLOCK
M_6–M_0	MATRIX SCAN LINES
B_3–B_0	DISPLAY OUTPUTS
ERROR	ERROR SIGNAL
$\overline{IRQ}$	INTERRUPT REQUEST
$\overline{HYS}$	HYSTERESIS
BP	TONE ENABLE

BLOCK DIAGRAM



PIN DESCRIPTION

The 8278 is packaged in a 40-pin DIP. The following is a brief functional description of each pin.

Signal	Pin No.	Description
D_0-D_7	12-19	Three-state, bi-directional data bus lines used to transfer data and commands between the CPU and the 8278.
$\overline{WR}$	10	Write strobe which enables the master CPU to write data and commands between the CPU and the 8278.
$\overline{RD}$	8	Read strobe which enables the master CPU to read data and status from the 8278 internal registers.
$\overline{CS}$	6	Chip select input used to enable reading and writing to the 8278.
A_0	9	Address input used by the CPU to indicate control or data.
$\overline{RESET}$	4	A low signal on this pin resets the 8278.
X_1, X_2	2,3	Inputs for crystal, L-C or external timing signal to determine internal oscillator frequency.
IRQ	23	Interrupt Request Output to the master CPU. In the keyboard mode the IRQ line goes low with each FIFO read and returns high if there is still information in the FIFO or an ERROR has occurred.
M_0-M_6	27-33	Matrix scan outputs. These outputs control a decoder which scans the key matrix columns and the 16 display digits. Also, the Matrix scan outputs are used to multiplex the return lines from the key matrix.
RL	1	Input from the multiplexer which indicates whether the key currently being scanned is closed.
$\overline{HYS}$	22	Hysteresis output to the analog detector. (Capacitive keyboard configuration). A "0" means the key currently being scanned has already been recorded.
KCL	34	Key clock output to the analog detector (capacitive keyboard configuration) used to reset the detector before scanning a key.
SYNC	11	High frequency (400 KHz) output signal used in the key scan to detect a closed key (capacitive keyboard configuration).
B_0-B_3	35-38	These four lines contain binary coded decimal display information synchronized to the keyboard column scan. The outputs are for multiplexed digital displays.
ERROR	24	Error signal. This line is high whenever two new key closures are detected during a single scan or when too many characters are entered into the keyboard FIFO. It is reset by a system RESET pulse or by a "1" input on the CLR pin or by the CLEAR ERROR command.
CLR	39	Input used to clear an ERROR condition in the 8278.
BP	21	Tone enable output. This line is high for 10ms following a valid key closure; it is set high and remains high during an ERROR condition.
V_{CC}, V_{DD}	40,26	+5 volt power input: +5V ± 10%.
GND	20,7	Signal ground.

PRINCIPLES OF OPERATION

The following is a description of the major elements of the Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 1.

I/O Control and Data Buffers

The I/O control section uses the $\overline{CS}$, A_0, $\overline{RD}$, and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers (see Table 1). All data flow to and from the 8278 is enabled by $\overline{CS}$. The 8-bits of information being transferred by the CPU is identified by A_0. A logic one means information is command or status. A logic zero means the information is data. $\overline{RD}$ and $\overline{WR}$ determine the direction of data flow through the Data Bus Buffer (DBB). The DBB register is a bi-directional 8-bit buffer register which connects the internal 8278 bus buffer register to the external bus. When the chip is not selected ($\overline{CS}$ = 1) the DBB is in the high impedance state. The DBB acts as an input when ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$) = (1, 0, 0) and an output when ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$) = (0, 1, 0).

$\overline{CS}$	A_0	$\overline{WR}$	$\overline{RD}$	Condition
0	0	1	0	Read DBB Data
0	1	1	0	Read STATUS
0	0	0	1	Write Data to DBB
0	1	0	1	Write Command to DBB
1	X	X	X	Disable 8278 Bus is High Impedance

Scan Counter

The scan counter provides the timing to scan the keyboard and display. The four MSB's (M_3-M_6) scan the display digits and provide column scan to the keyboard via a 4 to 16 decoder. The three LSB's (M_0-M_2) are used to multiplex the row return lines into the 8278.

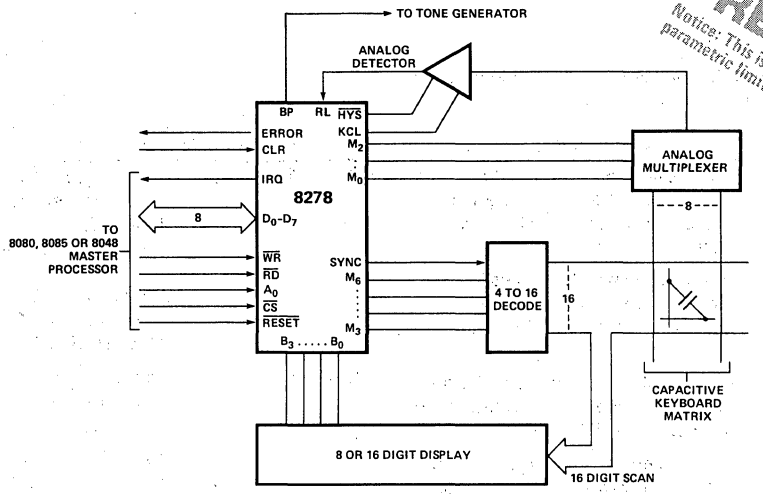


Figure 1. System Configuration for Capacitive-Coupled Keyboard

Keyboard Debounce and Control

The 8278 system configuration is shown in Figure 2. The rows of the matrix are scanned and the outputs are multiplexed by the 8278. When a key closure is detected, the debounce logic waits about 12 msec to check if the key remains closed. If it does,the address of the key in the matrix is transferred into a FIFO buffer.

FIFO and FIFO Status

The 8278 contains an 8X8 FIFO character buffer. Each new entry is written into a successive FIFO location and each is then read out in the order of entry. A FIFO status register keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or key entries will be recognized as an error. The status can be read by a $\overline{RD}$ with $\overline{CS}$ low and A_0 high. The status logic also provides a IRQ signal to the master processor whenever the FIFO is not empty.

Display Address Registers and Display RAM

The display Address registers hold the address of the word currently being written or read by the CPU and the 4-bit nibble being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The display RAM can be directly read by the CPU after the correct mode and address is set. Data entry to the display can be set to either left or right entry.

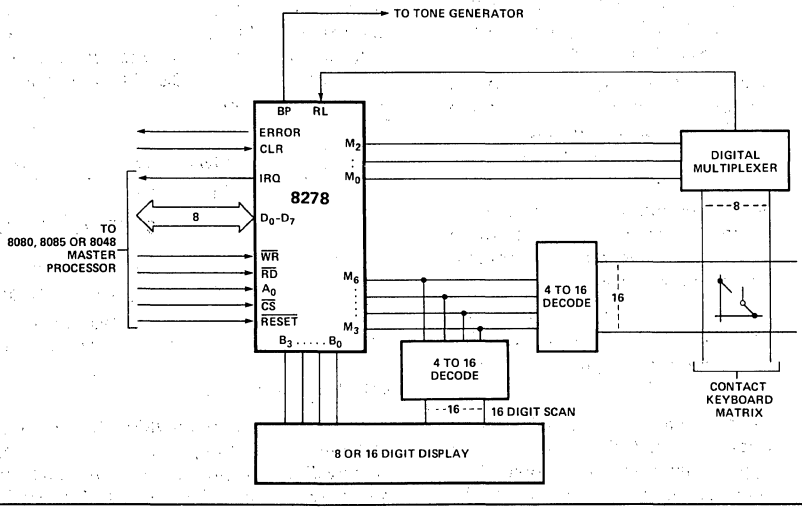
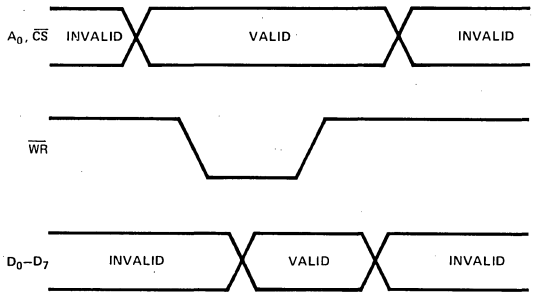


Figure 2. System Configuration for Contact Keyboard

8278 COMMANDS

The 8278 operating mode is programmed by the master CPU using the A_0, WR, and D_0-D_7 inputs as shown below:



The master CPU presents the proper command on the D_0-D_7 data lines with A_0=1 and then sends a $\overline{WR}$ pulse. The command is latched by the 8278 on the rising edge of the $\overline{WR}$ and is decoded internally to set the proper operating mode.

COMMAND SUMMARY

Keyboard/Display Mode Set

CODE	0	0	0	N	E	I	D	K

where the mode set bits are defined as follows:

K — the keyboard mode select bit
0 — normal key entry mode
1 — special function mode: Entry on key closure and on key release

D — the display entry mode select bit
0 — left display entry
1 — right display entry

I — the interrupt request (IRQ) output enable bit.
0 — enable IRQ output
1 — disable IRQ output

E — the error mode select bit
0 — error on multiple key depression
1 — no error on multiple key depression

N — the number of display digits select
0 — 16 display digits
1 — 8 display digits

NOTE: The default mode following a RESET input is all bits zero:

0	0	0	0	0	0	0	0

Read FIFO Command

CODE	0	1	0	0	0	0	0	0

Read Display Command

CODE	0	1	1	AI	A_3	A_2	A_1	A_0

Where AI indicates Auto Increment and A_3-A_0 is the address of the next display character to be read out.

AI=1 AUTO increment
AI=0 no AUTO increment

Write Display Command

CODE	1	0	0	AI	A_3	A_2	A_1	A_0

Where AI indicates Auto Increment and A_3-A_0 is the address of the next display character to be written.

Clear/Blank Command

CODE	1	0	1	UD	BD	CD	CF	CE

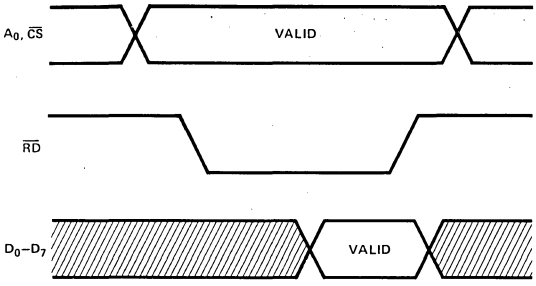
Where the command bits are defined as follows:

CE = Clear ERROR
CF = Clear FIFO
CD = Clear Display RAM to all High
BD = Blank Display to all High (Display RAM unaffected)
UD = Unblank Display

The display is cleared and blanked following a Reset.

8278 Status Read

The status register in the 8278 can be read by the master CPU using the A_0, $\overline{RD}$, and D_0-D_7 inputs as shown below:



The 8278 places 8-bits of status information on the D_0-D_7 lines following (A_0, $\overline{CS}$, $\overline{RD}$) = 1, 0 , 0 inputs from the master.

Status Format

S_3	S_2	S_1	S_0	B	KE	IBF	OBF
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Where the status bits are defined as follows:

OBF = Output Buffer Full Flag
IBF = Input Buffer Full Flag
KE = Keyboard Error Flag (multiple depression)
B = BUSY Flag
S_3-S_0 = FIFO Status

Status Description

The S_3-S_0 status bits indicate the number of entries (0 to 8) in the 8-level FIFO. A FIFO overrun will lock status at 1111. The overrun condition will prevent further key entries until cleared.

A multiple key closure error will set the KE flag and prevent further key entries until cleared.

The IBF and OBF flags signify the status of the 8278 data buffer registers used to transfer information (data, status or commands) to and from the master CPU.

The IBF flag is set when the master CPU writes Data or Commands to the 8278. The IBF flag is cleared by the 8278 during its response to the Data or Command.

The OBF flag is set when the 8278 has output data ready for the master CPU. This flag is cleared by a master CPU Data READ.

The Busy flag in the status register is used as a LOCK-OUT signal to the master processor during response to any command or data write from the master.

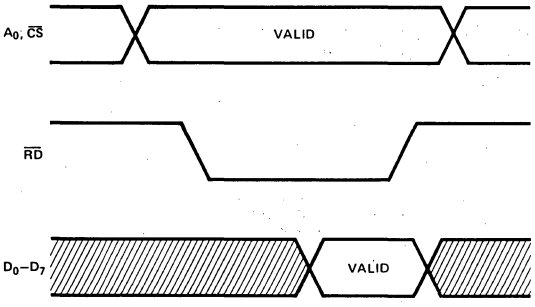
The master must test the Busy flag before each read (during a sequence) to be sure that the 8278 is ready with valid DATA.

The ERROR and TONE outputs from the 8278 are set high for either type of error. Both types of error are cleared by the CLR input, by the CLEAR ERROR command, or by a reset. The FIFO and Display buffers are cleared independently of the Errors.

FIFO status is used to indicate the number of characters in the FIFO and to indicate whether an error has occurred. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO. The character read will be the last one entered. FIFO status will remain at 0000 and the error condition will not be set.

8278 Data Read

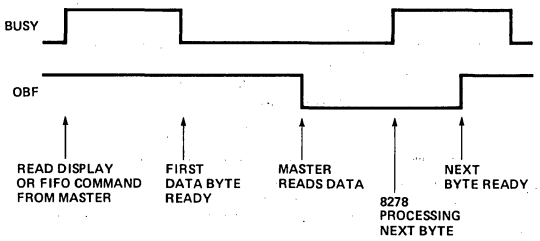
The master CPU can read DATA from the 8278 FIFO or Display buffers by using the A_0, $\overline{RD}$, and D_0-D_7 inputs as follows:



The master sends a $\overline{RD}$ pulse with A_0=0 and CS=0 and the 8278 responds by outputing data on lines D_0-D_7. The data is strobed by the trailing edge of $\overline{RD}$.

Data Read Sequence

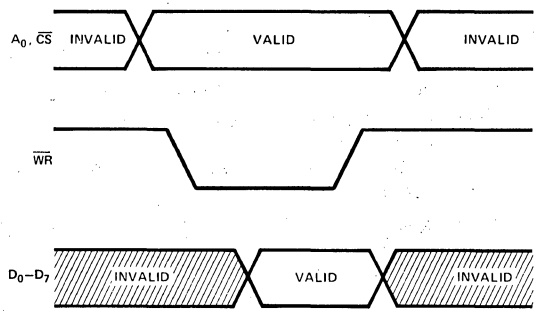
Before reading data, the master CPU must send a command to select FIFO or Display data. Following the command, the master must read STATUS and test the BUSY flag and the OBF flag to verify that the 8278 has responded to the previous command. A typical DATA READ sequence is as follows:



After the first read following a Read Display or Read FIFO command, successive reads may occur as soon as OBF rises.

8278 Data Write

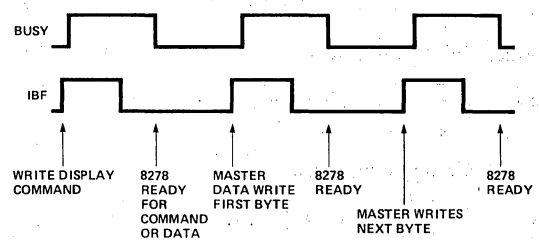
The master CPU can write DATA to the 8278 Display buffers by using the A_0, $\overline{WR}$ and D_0-D_7 inputs as follows:



The master CPU presents the Data on the D_0-D_7 lines with A_0=0 and then sends a $\overline{WR}$ pulse. The data is latched by the 8278 on the rising edge of $\overline{WR}$.

Data Write Sequence

Before writing data to the 8278, the master CPU must first send a command to select the desired display entry mode and to specify the address of the next data byte. Following the commands, the master must read STATUS and test the BUSY flag (B) and IBF flag to verify that the 8278 has responded. A typical sequence is shown below:



INTERFACE CONSIDERATIONS

Scanned Keyboard Mode

With N-key rollover each key depression is treated independently from all others. When a key is depressed the debounce logic waits for a full scan of 128 keys and then checks to see if the key is still down. If it is, the key is entered into the FIFO.

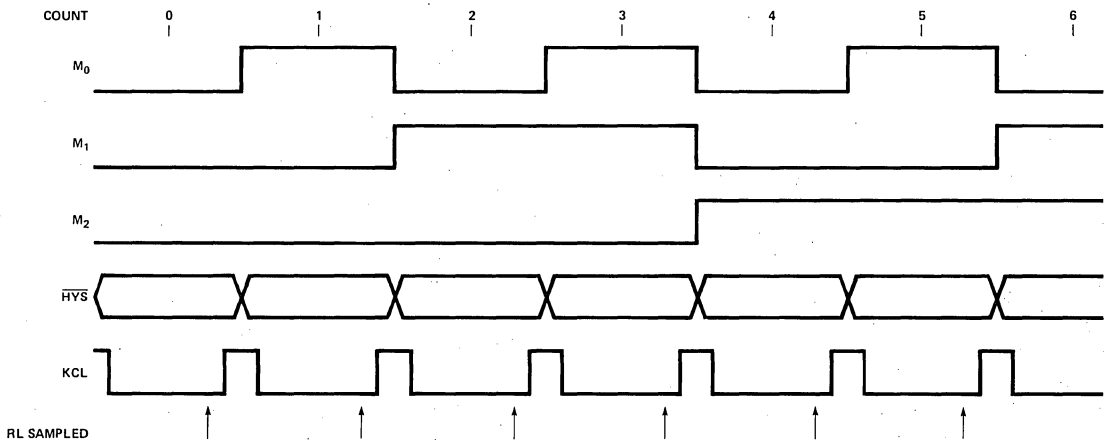


Figure 3. Keyboard Timing

If two key closures occur during the same scan the ERROR output is set, the KE flag is set in the Status word, the TONE output is activated and IRQ is set, and no further inputs are accepted. This condition is cleared by a high signal on the CLEAR input or by a system $\overline{RESET}$ input or by the CLEAR ERROR command.

In the special function mode both the key closure and the key release cause an entry to the FIFO. The release is entered with the MSB=1.

Any key entry triggers the TONE output for 10ms.

The $\overline{HYS}$ and KCL outputs enable the analog multiplexer and detector to be synchronized for interface to capacitive coupled keyboards.

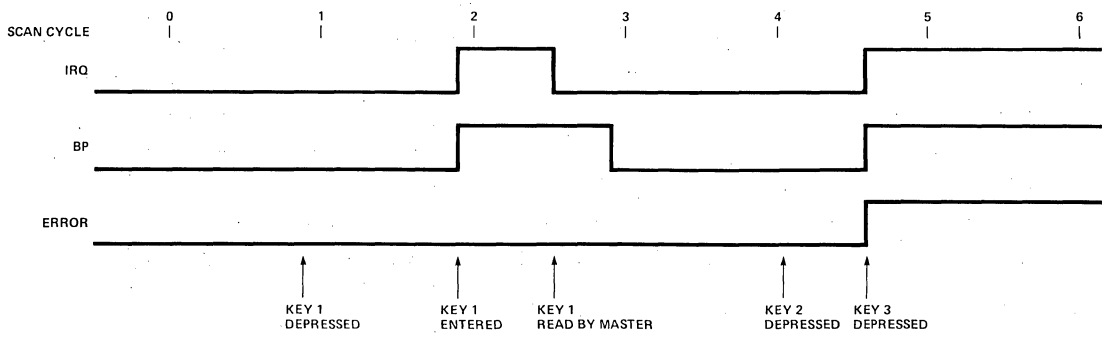


Figure 4. Key Entry and Error Timing

Data Format

In the scanned keyboard mode, the code entered into the FIFO corresponds to the position or address of the switch in the keyboard. The MSB is relevant only for special function keys in which code "0" signifies closure and "1" signifies release. The next four bits are the column count which indicates which column the key was found in. The last three bits are from the row counter.

KEY CODING

BIT	7	6	5	4	3	2	1	0
	SF	M_6	M_5	M_4	M_3	M_2	M_1	M_0

SF: 1 FOR SPECIAL FUNCTION MODE AND KEY RELEASED; 0 FOR KEY DEPRESSED

Display

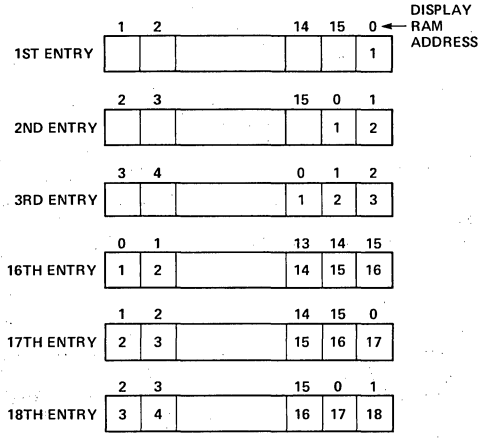
Display data is entered into a 16x4 display register and may be entered from the left, from the right or into specific locations in the display register. A new data character is put out on B_0-B_3 each time the M_6-M_3 lines change (i.e., once every 0.75ms with a 6 MHz crystal). Data is blanked during the time the column select lines change by raising the display outputs. Output data is positive true.

Left Entry

The left entry mode is the simplest display format in that each display position in the display corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 is the right-most display character. Entering characters from position zero causes the display to fill from the left. The 17th character is entered back in the left-most position and filling again proceeds from there.

Right Entry

Right entry is the method used by most electronic calculators. The first entry is placed in the right-most display character. The next entry is also placed in the right-most character after the display is shifted left one character. The left-most character is shifted off the end and is lost.



Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended. A Clear Display command should be given before display data is entered if the number of data characters is not equal to 16 (or 8) in this mode.

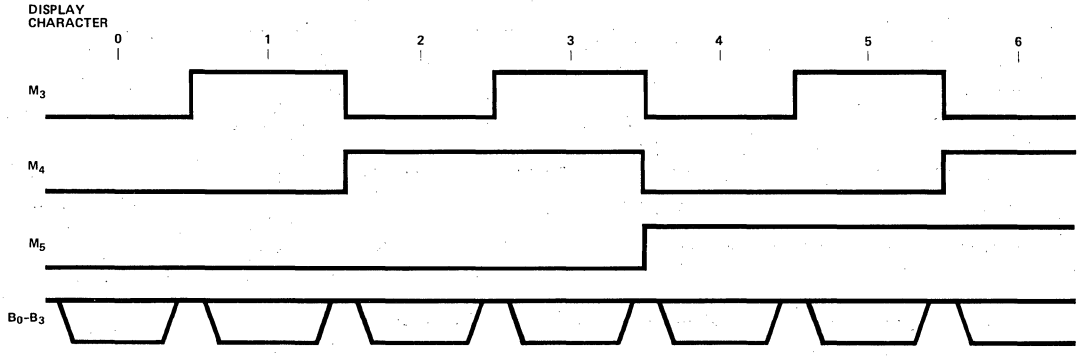


Figure 5. Display Timing

Auto Increment

In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Left Entry — Auto Increment mode has no undesirable side effects and the result is predictable:

	0	1	2	3	4	5	6	7 ← DISPLAY RAM ADDRESS
1ST ENTRY	1							

	0	1	2	3	4	5	6	7
2ND ENTRY	1	2						

	0	1	2	3	4	5	6	7
COMMAND 10010101	1	2						

ENTER NEXT AT LOCATION 5 AUTO INCREMENT

	0	1	2	3	4	5	6	7
3RD ENTRY	1	2				3		

	0	1	2	3	4	5	6	7
4TH ENTRY	1	2				3	4	

In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except that the address sequence is interrupted:

	1	2	3	4	5	6	7	0 ← DISPLAY RAM ADDRESS
1ST ENTRY								1

	2	3	4	5	6	7	0	1
2ND ENTRY							1	2

	2	3	4	5	6	7	0	1
COMMAND 10010101							1	2

ENTER NEXT AT LOCATION 5 AUTO INCREMENT

	3	4	5	6	7	0	1	2
3RD ENTRY			3			1	2	

	4	5	6	7	0	1	2	3
4TH ENTRY		3	4		1	2		

Starting at an arbitrary location operates as shown below:

	0	1	2	3	4	5	6	7 ← DISPLAY RAM ADDRESS
COMMAND 10010101								

ENTER NEXT AT LOCATION 5 AUTO INCREMENT

	1	2	3	4	5	6	7	0
1ST ENTRY					1			

	2	3	4	5	6	7	0	1
2ND ENTRY				1	2			

8TH ENTRY	4	5	6	7	8	1	2	3

9TH ENTRY	5	6	7	8	9	2	3	4

Entry appears to be from the initial entry point.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... -65°C to +150°C
Voltage on Any Pin With
Respect to Ground ..... −0.5V to +7V
Power Dissipation ..... 1.5 Watt

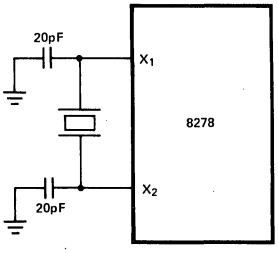
**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

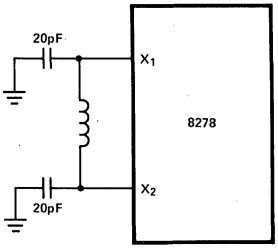
Commercial: $T_A = 0°C$ to $70°C$; $V_{CC} = +5V \pm 5\%$; $V_{SS} = 0V$

Symbol	Parameter	Min.	Max.	Units	Condition
V_{IL}	Input Low Voltage (All Inputs Except X_1, X_2	-0.5	0.8	V	
V_{IH1}	Input High Voltage (All Inputs Except X_1, X_2, $\overline{RESET}$	2.0	V_{CC}	V	
V_{IH2}	$\overline{RESET}$ High Voltage	3.0	V_{CC}	V	
V_{OL1}	Output Low Voltage (D_0-D_7)		0.45	V	$I_{OL} = 2.0mA$
V_{OL2}	Output Low Voltage (All Other Outputs)		0.45	V	$I_{OL} = 1.6mA$
V_{OH1}	Output High Voltage (D_0-D_7)	2.4		V	$I_{OH} = -400\mu A$
V_{OH2}	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50\mu A$
I_{IL}	Input Leakage Current (All Inputs Except $\overline{RESET}$)		±10	μA	$V_{IN} = V_{CC}$
I_{OL}	Output Leakage Current (D_0-D_7)		±10	μA	$V_{IN} = V_{SS} + 0.45V$ or $V_{IN} = V_{CC}$
$I_{DD} + I_{CC}$	Total Supply Current		135	mA	$V_{CC} = 5.5V$
I_{DD}	V_{DD} Supply Current		25	mA	$V_{CC} = 5.5V$
I_{LI}	Low Input Source Current ($\overline{RESET}$)		0.2	mA	$V_{IL} = 0.8V$

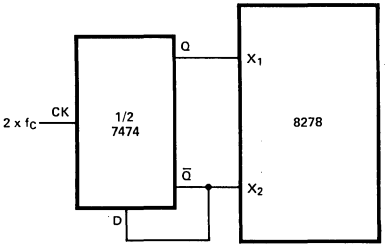
8278 CLOCK OPTIONS



1-6 MHz
CRYSTAL



40 μh-130 μh
INDUCTOR



EXTERNAL
CLOCK

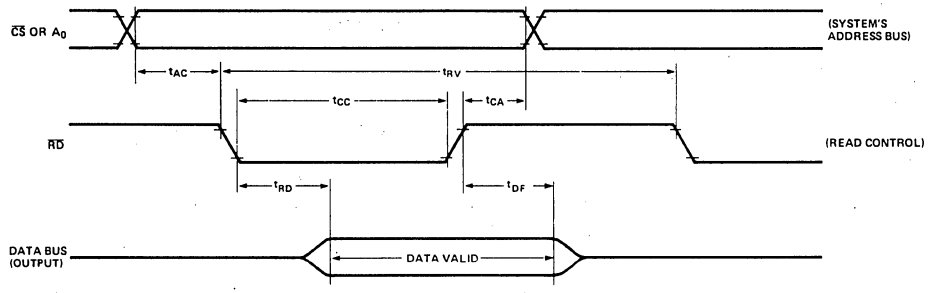
A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$; $V_{CC} = +5V \pm 10\%$; $V_{SS} = 0V$

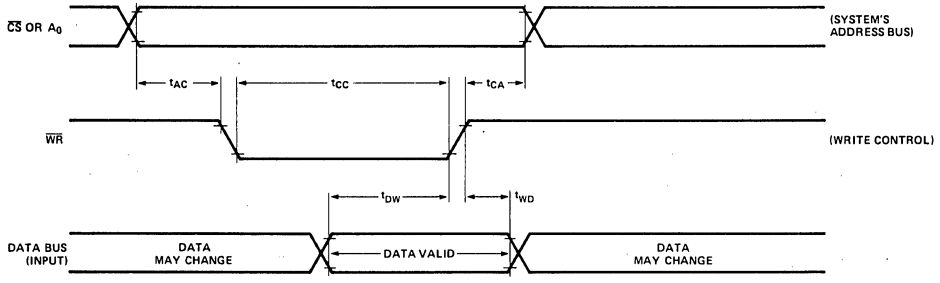
Symbol	Parameter	Min.	Max.	Units	Condition
t_{AC}	Address ($\overline{CS}$, A_0) Setup to Control ($\overline{RD}$, $\overline{WR}$)	0		ns	D_0-D_7, C_L = 150pF
t_{CA}	Address Hold from Control	0		ns	
t_{CC}	Control Pulse Width	250		ns	
t_{DW}	Data in Setup to $\overline{WR}$ T.E.	150		ns	
t_{WD}	Data in Hold After $\overline{WR}$ T.E.	0		ns	
t_{RD}	$\overline{RD}$ L.E. to Data Out Valid		150	ns	
t_{DF}	$\overline{RD}$ T.E. to Data Out Float	10	100	ns	
t_{MCY}	Matrix Cycle Time		10.7	ms	With 6MHz Crystal
t_{RV}	Recovery Time Between Reads and/or Writes	1		μs	

WAVEFORMS

Read Operation — Data Bus Buffer Register



Write Operation — Data Bus Buffer Register



intel®

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

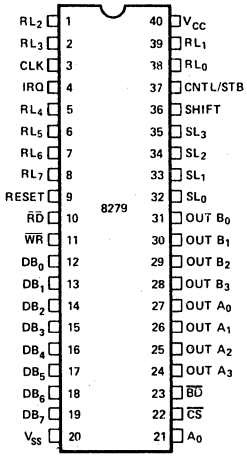
8279/8279-5
PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- **MCS-85™ Compatible 8279-5**
- **Simultaneous Keyboard Display Operations**
- **Scanned Keyboard Mode**
- **Scanned Sensor Mode**
- **Strobed Input Entry Mode**
- **8-Character Keyboard FIFO**
- **2-Key Lockout or N-Key Rollover with Contact Debounce**
- **Dual 8- or 16-Numerical Display**
- **Single 16-Character Display**
- **Right or Left Entry 16-Byte Display RAM**
- **Mode Programmable from CPU**
- **Programmable Scan Timing**
- **Interrupt Output on Key Entry**

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16X8 display RAM which can be organized into dual 16X4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

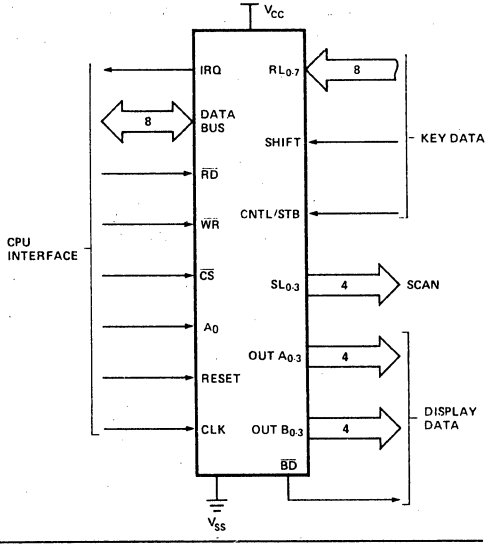
PIN CONFIGURATION



PIN NAMES

DB_{0-7}	I/O	DATA BUS (BI-DIRECTIONAL)
CLK	I	CLOCK INPUT
RESET	I	RESET INPUT
$\overline{CS}$	I	CHIP SELECT
$\overline{RD}$	I	READ INPUT
$\overline{WR}$	I	WRITE INPUT
A_0	I	BUFFER ADDRESS
IRQ	O	INTERRUPT REQUEST OUTPUT
SL_{0-3}	O	SCAN LINES
RL_{0-7}	I	RETURN LINES
SHIFT	I	SHIFT INPUT
CNTL/STB	I	CONTROL/STROBE INPUT
OUT A_{0-3}	O	DISPLAY (A) OUTPUTS
OUT B_{0-3}	O	DISPLAY (B) OUTPUTS
$\overline{BD}$	O	BLANK DISPLAY OUTPUT

LOGIC SYMBOL



HARDWARE DESCRIPTION

The 8279 is packaged in a 40 pin DIP. The following is a functional description of each pin.

No. Of Pins	Designation	Function
8	DB_0-DB_7	Bi-directional data bus. All data and commands between the CPU and the 8279 are transmitted on these lines.
1	CLK	Clock from system used to generate internal timing.
1	RESET	A high signal on this pin resets the 8279. After being reset the 8279 is placed in the following mode: 1) 16 8-bit character display —left entry. 2) Encoded scan keyboard—2 key lockout. Along with this the program clock prescaler is set to 31.
1	$\overline{CS}$	Chip Select. A low on this pin enables the interface functions to receive or transmit.
1	A_0	Buffer Address. A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
2	$\overline{RD}$, $\overline{WR}$	Input/Output read and write. These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
1	IRQ	Interrupt Request. In a keyboard mode, the interrupt line is high when there is data in the FIFO/Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM. In a sensor mode, the interrupt line goes high whenever a change in a sensor is detected.
2	V_{SS}, V_{CC}	Ground and power supply pins.
4	SL_0-SL_3	Scan Lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
8	RL_0-RL_7	Return line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pullups to keep them high until a switch closure pulls one low. They also serve as an 8-bit input in the Strobed Input mode.
1	SHIFT	The shift input status is stored along with the key position on key closure in the Scanned Keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low.
1	CNTL/STB	For keyboard modes this line is used as a control input and stored like status on a key closure. The line is also the strobe line that enters the data into the FIFO in the Strobed Input mode. (Rising Edge). It has an active internal pullup to keep it high until a switch closure pulls it low.
4 4	OUT A_0-OUT A_3 OUT B_0-OUT B_3	These two ports are the outputs for the 16 x 4 display refresh registers. The data from these outputs is synchronized to the scan lines (SL_0-SL_3) for multiplexed digit displays. The two 4 bit ports may be blanked independently. These two ports may also be considered as one 8 bit port.
1	$\overline{BD}$	Blank Display. This output is used to blank the display during digit switching or by a display blanking command.

PRINCIPLES OF OPERATION

The following is a description of the major elements of the 8279 Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 1.

I/O Control and Data Buffers

The I/O control section uses the $\overline{CS}$, A_0, $\overline{RD}$ and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers. All data flow to and from the 8279 is enabled by $\overline{CS}$. The character of the information, given or desired by the CPU, is identified by A_0. A logic one means the information is a command or status. A logic zero means the information is data. $\overline{RD}$ and $\overline{WR}$ determine the direction of data flow through the Data Buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ($\overline{CS}$ = 1), the devices are in a high impedance state. The drivers input during $\overline{WR} \bullet \overline{CS}$ and output during $\overline{RD} \bullet \overline{CS}$.

Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by presenting the proper command on the data lines with A_0 = 1 and then sending a $\overline{WR}$. The command is latched on the rising edge of $\overline{WR}$.

FUNCTIONAL DESCRIPTION

Since data input and display are an integral part of many microprocessor designs, the system designer needs an interface that can control these functions without placing a large load on the CPU. The 8279 provides this function for 8-bit microprocessors.

The 8279 has two sections: keyboard and display. The keyboard section can interface to regular typewriter style keyboards or random toggle or thumb switches. The display section drives alphanumeric displays or a bank of indicator lights. Thus the CPU is relieved from scanning the keyboard or refreshing the display.

The 8279 is designed to directly connect to the microprocessor bus. The CPU can program all operating modes for the 8279. These modes include:

Input Modes

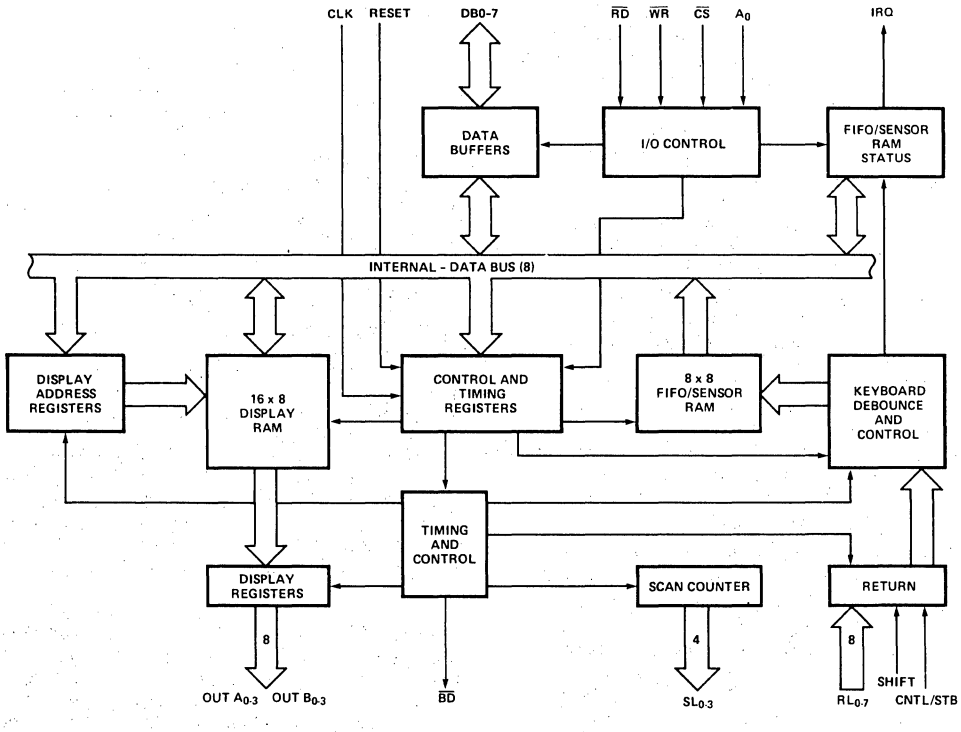
- Scanned Keyboard — with encoded (8 x 8 key keyboard) or decoded (4 x 8 key keyboard) scan lines. A key depression generates a 6-bit encoding of key position. Position and shift and control status are stored in the FIFO. Keys are automatically debounced with 2-key lockout or N-key rollover.
- Scanned Sensor Matrix — with encoded (8 x 8 matrix switches) or decoded (4 x 8 matrix switches) scan lines. Key status (open or closed) stored in RAM addressable by CPU.
- Strobed Input — Data on return lines during control line strobe is transferred to FIFO.

Output Modes

- 8 or 16 character multiplexed displays that can be organized as dual 4-bit or single 8-bit ($B_0 = D_0$, $A_3 = D_7$).
- Right entry or left entry display formats.

Other features of the 8279 include:

- Mode programming from the CPU.
- Clock Prescaler
- Interrupt output to signal CPU when there is keyboard or sensor data available.
- An 8 byte FIFO to store keyboard information.
- 16 byte internal Display RAM for display refresh. This RAM can also be read by the CPU.



The command is then decoded and the appropriate function is set. The timing control contains the basic timing counter chain. The first counter is a ÷ N prescaler that can be programmed to yield an internal frequency of 100 kHz which gives a 5.1 ms keyboard scan time and a 10.3 ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan times.

Scan Counter

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant 2 bits and provides a decoded 1 of 4 scan. Note than when the keyboard is in decoded scan, so is the display. This means that only the first 4 characters in the Display RAM are displayed.

In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.

Return Buffers and Keyboard Debounce and Control

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a closed switch, it waits about 10 msec to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL are transferred to the FIFO. In the scanned Sensor Matrix modes, the contents of the return lines is directly transferred to the corresponding row of the Sensor RAM (FIFO) each key scan time. In Strobed Input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse.

FIFO/Sensor RAM and Status

This block is a dual function 8 x 8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by an $\overline{RD}$ with $\overline{CS}$ low and A_0 high. The status logic also provides an IRQ signal when the FIFO is not empty. In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected.

Display Address Registers and Display RAM

The Display Address Registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The Display RAM can be directly read by the CPU after the correct mode and address is set. The addresses for the A and B nibbles are automatically updated by the 8279 to match data entry by the CPU. The A and B nibbles can be entered independently or as one word, according to the mode that is set by the CPU. Data entry to the display can be set to either left or right entry. See Interface Considerations for details.

SOFTWARE OPERATION

8279 commands

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with $\overline{CS}$ low and A_0 high and are loaded to the 8279 on the rising edge of $\overline{WR}$.

Keyboard/Display Mode Set

	MSB							LSB
Code:	0	0	0	D	D	K	K	K

Where DD is the Display Mode and KKK is the Keyboard Mode.

DD

0	0	8 8-bit character display — Left entry
0	1	16 8-bit character display — Left entry*
1	0	8 8-bit character display — Right entry
1	1	16 8-bit character display — Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

KKK

0	0	0	Encoded Scan Keyboard — 2 Key Lockout*
0	0	1	Decoded Scan Keyboard — 2-Key Lockout
0	1	0	Encoded Scan Keyboard — N-Key Rollover
0	1	1	Decoded Scan Keyboard — N-Key Rollover
1	0	0	Encoded Scan Sensor Matrix
1	0	1	Decoded Scan Sensor Matrix
1	1	0	Strobed Input, Encoded Display Scan
1	1	1	Strobed Input, Decoded Display Scan

Program Clock

Code:	0	0	1	P	P	P	P	P

All timing and multiplexing signals for the 8279 are generated by an internal prescaler. This prescaler divides the external clock (pin 3) by a programmable integer. Bits PPPPP determine the value of this integer which ranges from 2 to 31. Choosing a divisor that yields 100 kHz will give the specified scan and debounce times. For instance, if Pin 3 of the 8279 is being clocked by a 2 MHz signal, PPPPP should be set to 10100 to divide the clock by 20 to yield the proper 100 kHz operating frequency.

Read FIFO/Sensor RAM

Code:	0	1	0	AI	X	A	A	A

X = Don't Care

The CPU sets up the 8279 for a read of the FIFO/Sensor RAM by first writing this command. In the Scan Key-

*Default after reset.

board Mode, the Auto-Increment flag (AI) and the RAM address bits (AAA) are irrelevant. The 8279 will automatically drive the data bus for each subsequent read ($A_0 = 0$) in the same sequence in which the data first entered the FIFO. All subsequent reads will be from the FIFO until another command is issued.

In the Sensor Matrix Mode, the RAM address bits AAA select one of the 8 rows of the Sensor RAM. If the AI flag is set (AI = 1), each successive read will be from the subsequent row of the sensor RAM.

Read Display RAM

Code:

0	1	1	AI	A	A	A	A

The CPU sets up the 8279 for a read of the Display RAM by first writing this command. The address bits AAAA select one of the 16 rows of the Display RAM. If the AI flag is set (AI = 1), this row address will be incremented after each following read *or write* to the Display RAM. Since the same counter is used for both reading and writing, this command sets the next read *or write* address and the sense of the Auto-Increment mode for both operations.

Write Display RAM

Code:

1	0	0	AI	A	A	A	A

The CPU sets up the 8279 for a write to the Display RAM by first writing this command. After writing the command with $A_0 = 1$, all subsequent writes with $A_0 = 0$ will be to the Display RAM. The addressing and Auto-Increment functions are identical to those for the Read Display RAM. However, this command does not affect the source of subsequent Data Reads; the CPU will read from whichever RAM (Display or FIFO/Sensor) which was last specified. If, indeed, the Display RAM was last specified, the Write Display RAM will, nevertheless, change the next Read location.

Display Write Inhibit/Blanking

The IW Bits can be used to mask nibble A and nibble B in applications requiring separate 4-bit display ports. By setting the IW flag (IW = 1) for one of the ports, the port becomes marked so that entries to the Display RAM from the CPU do not affect that port. Thus, if each nibble is input to a BCD decoder, the CPU may write a digit to the Display RAM without affecting the other digit being displayed. It is important to note that bit B_0 corresponds to bit D_0 on the CPU bus, and that bit A_3 corresponds to bit D_7.

If the user wishes to blank the display, the BL flags are available for each nibble. The last Clear command issued determines the code to be used as a "blank." This code defaults to all zeros after a reset. Note that both BL flags must be set to blank a display formatted with a single 8-bit port.

Clear

The C_D bits are available in this command to clear all rows of the Display RAM to a selectable blanking code as follows:

C_D	C_D	C_D	
0	X		All Zeros (X = Don't Care)
1	0		AB = Hex 20 (0010 0000)
1	1		All Ones

Enable clear display when = 1 (or by C_A = 1)

During the time the Display RAM is being cleared (~160 μs), it may not be written to. The most significant bit of the FIFO status word is set during this time. When the Display RAM becomes available again, it automatically resets.

If the C_F bit is asserted ($C_F = 1$), the FIFO status is cleared and the interrupt output line is reset. Also, the Sensor RAM pointer is set to row 0.

C_A, the Clear All bit, has the combined effect of C_D and C_F; it uses the C_D clearing code on the Display RAM and also clears FIFO status. Furthermore, it resynchronizes the internal timing chain.

End Interrupt/Error Mode Set

Code:

1	1	1	E	X	X	X	X

X = Don't care.

For the sensor matrix modes this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset).

For the N-key rollover mode — if the E bit is programmed to "1" the chip will operate in the special Error mode. (For further details, see Interface Considerations Section.)

Status Word

The status word contains the FIFO status, error, and display unavailable signals. This word is read by the CPU when A_0 is high and $\overline{CS}$ and $\overline{RD}$ are low. See Interface Considerations for more detail on status word.

Data Read

Data is read when A_0, $\overline{CS}$ and $\overline{RD}$ are all low. The source of the data is specified by the Read FIFO or Read Display commands. The trailing edge of $\overline{RD}$ will cause the address of the RAM being read to be incremented if the Auto-Increment flag is set. FIFO reads always increment (if no error occurs) independent of AI.

Data Write

Data that is written with A_0, $\overline{CS}$ and $\overline{WR}$ low is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command. Auto-Incrementing on the rising edge of $\overline{WR}$ occurs if AI set by the latest display command.

INTERFACE CONSIDERATIONS

Scanned Keyboard Mode, 2-Key Lockout

There are three possible combinations of conditions that can occur during debounce scanning. When a key is depressed, the debounce logic is set. Other depressed keys are looked for during the next two scans. If none are encountered, it is a single key depression and the key position is entered into the FIFO along with the status of CNTL and SHIFT lines. If the FIFO was empty, IRQ will be set to signal the CPU that there is an entry in the FIFO. If the FIFO was full, the key will not be entered and the error flag will be set. If another closed switch is encountered, no entry to the FIFO can occur. If all other keys are released before this one, then it will be entered to the FIFO. If this key is released before any other, it will be entirely ignored. A key is entered to the FIFO only once per depression, no matter how many keys were pressed along with it or in what order they were released. If two keys are depressed within the debounce cycle, it is a simultaneous depression. Neither key will be recognized until one key remains depressed alone. The last key will be treated as a single key depression.

Scanned Keyboard Mode, N-Key Rollover

With N-key Rollover each key depression is treated independently from all others. When a key is depressed, the debounce circuit waits 2 keyboard scans and then checks to see if the key is still down. If it is, the key is entered into the FIFO. Any number of keys can be depressed and another can be recognized and entered into the FIFO. If a simultaneous depression occurs, the keys are recognized and entered according to the order the keyboard scan found them.

Scanned Keyboard — Special Error Modes

For N-key rollover mode the user can program a special error mode. This is done by the "End Interrupt/Error Mode Set" command. The debounce cycle and key-validity check are as in normal N-key mode. If during a single debounce cycle, two keys are found depressed, this is considered a simultaneous multiple depression, and sets an error flag. This flag will prevent any further writing into the FIFO and will set interrupt (if not yet set). The error flag could be read in this mode by reading the FIFO STATUS word. (See "FIFO STATUS" for further details.) The error flag is reset by sending the normal CLEAR command with $C_F = 1$.

Sensor Matrix Mode

In Sensor Matrix mode, the debounce logic is inhibited. The status of the sensor switch is inputted directly to the Sensor RAM. In this way the Sensor RAM keeps an image of the state of the switches in the sensor matrix. Although debouncing is not provided, this mode has the advantage that the CPU knows how long the sensor was closed and when it was released. A keyboard mode can only indicate a validated closure. To make the software easier, the designer should functionally group the sensors by row since this is the format in which the CPU will read them.

The IRQ line goes high if any sensor value change is detected at the end of a sensor matrix scan. The IRQ line is cleared by the first data read operation if the Auto-Increment flag is set to zero, or by the End Interrupt command if the Auto-Increment flag is set to one.

Note: Multiple changes in the matrix Addressed by (SL_{0-3} = 0) may cause multiple interrupts. (SL_0 = 0 in the Decoded Mode). Reset may cause the 8279 to see multiple changes.

Data Format

In the Scanned Keyboard mode, the character entered into the FIFO corresponds to the position of the switch in the keyboard plus the status of the CNTL and SHIFT lines (non-inverted). CNTL is the MSB of the character and SHIFT is the next most significant bit. The next three bits are from the scan counter and indicate the row the key was found in. The last three bits are from the column counter and indicate to which return line the key was connected.

MSB							LSB
CNTL	SHIFT	SCAN			RETURN		

SCANNED KEYBOARD DATA FORMAT

In Sensor Matrix mode, the data on the return lines is entered directly in the row of the Sensor RAM that corresponds to the row in the matrix being scanned. Therefore, each switch postion maps directly to a Sensor RAM position. The SHIFT and CNTL inputs are ignored in this mode. Note that switches are not necessarily the only thing that can be connected to the return lines in this mode. Any logic that can be triggered by the scan lines can enter data to the return line inputs. Eight multiplexed input ports could be tied to the return lines and scanned by the 8279.

MSB							LSB
RL_7	RL_6	RL_5	RL_4	RL_3	RL_2	RL_1	RL_0

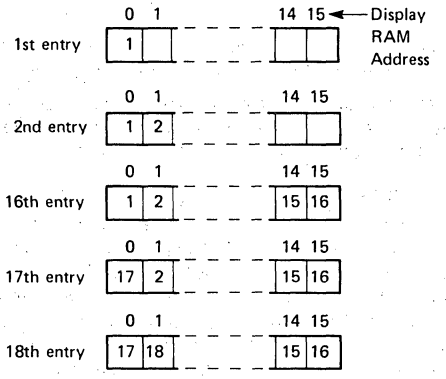
In Strobed Input mode, the data is also entered to the FIFO from the return lines. The data is entered by the rising edge of a CNTL/STB line pulse. Data can come from another encoded keyboard or simple switch matrix. The return lines can also be used as a general purpose strobed input.

MSB							LSB
RL_7	RL_6	RL_5	RL_4	RL_3	RL_2	RL_1	RL_0

Display

Left Entry

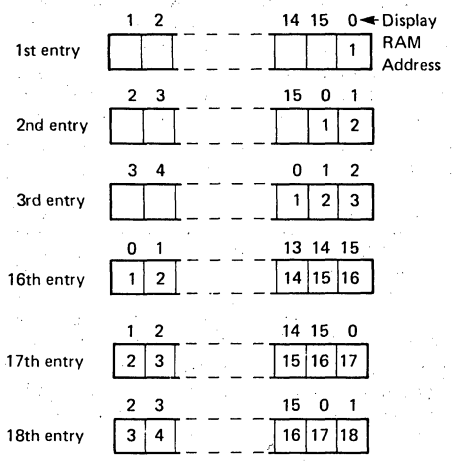
Left Entry mode is the simplest display format in that each display position directly corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 (or address 7 in 8 character display) is the right most display character. Entering characters from position zero causes the display to fill from the left. The 17th (9th) character is entered back in the left most position and filling again proceeds from there.



LEFT ENTRY MODE
(AUTO INCREMENT)

Right Entry

Right entry is the method used by most electronic calculators. The first entry is placed in the right most display character. The next entry is also placed in the right most character after the display is shifted left one character. The left most character is shifted off the end and is lost.

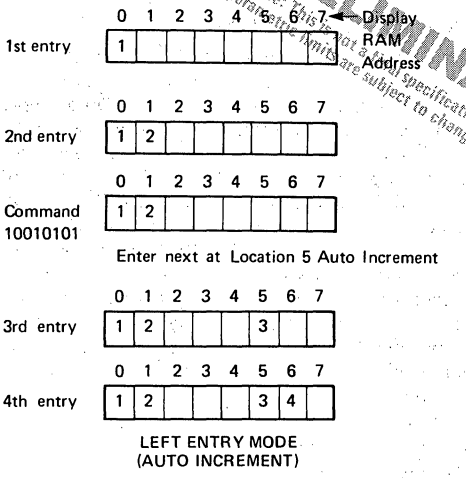


RIGHT ENTRY MODE
(AUTO INCREMENT)

Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended.

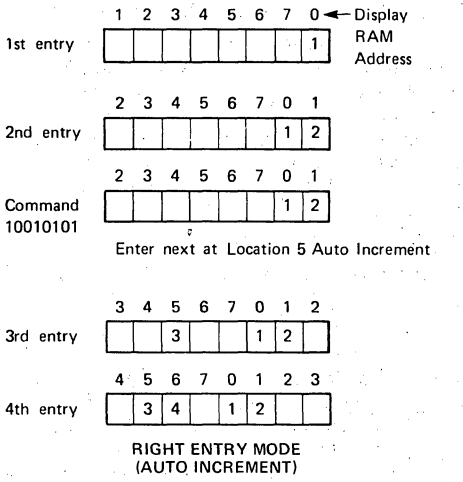
Auto Increment

In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Auto Increment mode has no undesirable side effects and the result is predictable:



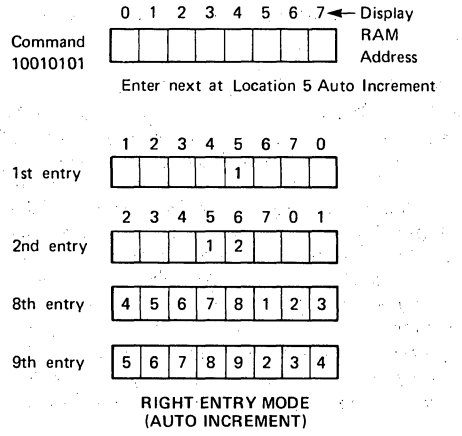
LEFT ENTRY MODE
(AUTO INCREMENT)

In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except if the address sequence is interrupted:



RIGHT ENTRY MODE
(AUTO INCREMENT)

Starting at an arbitrary location operates as shown below:



RIGHT ENTRY MODE
(AUTO INCREMENT)

Entry appears to be from the initial entry point.

8/16 Character Display Formats

If the display mode is set to an 8 character display, the on duty-cycle is double what it would be for a 16 character display (e.g., 5.1 ms scan time for 8 characters vs. 10.3 ms for 16 characters with 100 kHz internal frequency).

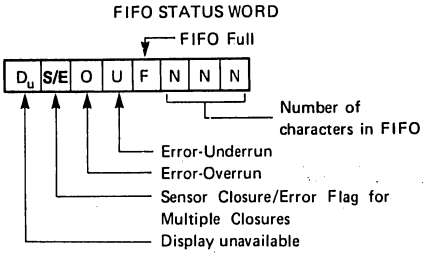
G. FIFO Status

FIFO status is used in the Keyboard and Strobed Input modes to indicate the number of characters in the FIFO and to indicate whether an error has occurred. There are two types of errors possible: overrun and underrun. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO.

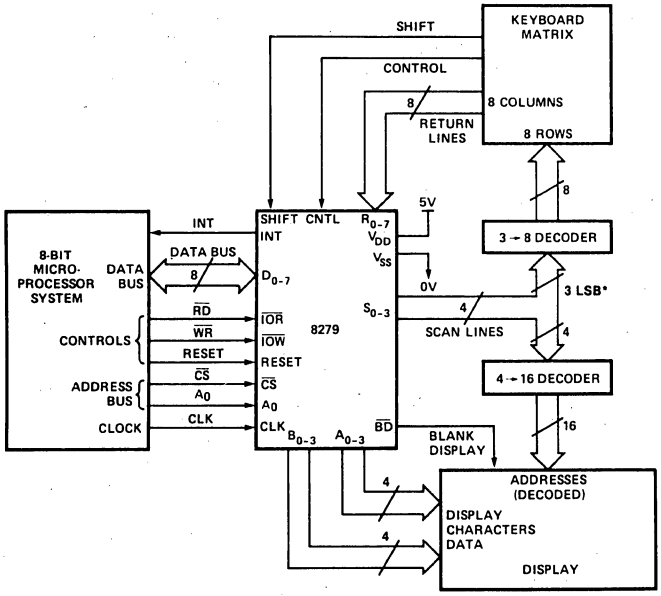
The FIFO status word also has a bit to indicate that the Display RAM was unavailable because a Clear Display or Clear All command had not completed its clearing operation.

In a Sensor Matrix mode, a bit is set in the FIFO status word to indicate that at least one sensor closure indication is contained in the Sensor RAM.

In Special Error Mode the S/E bit is showing the error flag and serves as an indication to whether a simultaneous multiple closure error has occurred.



APPLICATIONS



*Do not drive the keyboard decoder with the MSB of the scan lines.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature . . . . . 0°C to 70°C
Storage Temperature . . . . . -65°C to 125°C
Voltage on any Pin with
Respect to Ground . . . . . -0.5V to +7V
Power Dissipation . . . . . .1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{SS} = 0V$, $V_{CC} = +5V \pm 5\%$, $V_{CC} = +5V \pm 10\%$ (8279-5)

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL1}	Input Low Voltage for Return Lines	-0.5	1.4	V	
V_{IL2}	Input Low Voltage for All Others	-0.5	0.8	V	
V_{IH1}	Input High Voltage for Return Lines	2.2		V	
V_{IH2}	Input High Voltage for All Others	2.0		V	
V_{OL}	Output Low Voltage		0.45	V	Note 1
V_{OH1}	Output High Voltage on Interrupt Line	3.5		V	Note 2
V_{OH2}	Other Outputs	2.4			
I_{IL1}	Input Current on Shift, Control and Return Lines		+10 -100	μA μA	$V_{IN} = V_{CC}$ $V_{IN} = 0V$
I_{IL2}	Input Leakage Current on All Others		±10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OFL}	Output Float Leakage		±10	μA	$V_{OUT} = V_{CC}$ to 0V
I_{CC}	Power Supply Current		120	mA	

Notes:
8279, I_{OL} = 1.6mA; 8279-5, I_{OL} = 2.2mA.
8279, I_{OH} = -100μA; 8279-5, I_{OH} = -400μA.

CAPACITANCE

SYMBOL	TEST	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{in}	Input Capacitance	5	10	pF	$V_{in}=V_{CC}$
C_{out}	Output Capacitance	10	20	pF	$V_{out}=V_{CC}$

A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{SS} = 0V, (Note 1)

Bus Parameters

Read Cycle:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
t_{AR}	Address Stable Before $\overline{READ}$	50		0		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		0		ns
t_{RR}	$\overline{READ}$ Pulse Width	420		250		ns
t_{RD}[2]	Data Delay from $\overline{READ}$		300		150	ns
t_{AD}[2]	Address to Data Valid		450		250	ns
t_{DF}	$\overline{READ}$ to Data Floating	10	100	10	100	ns
t_{RCY}	Read Cycle Time	1		1		μs

Write Cycle:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
t_{AW}	Address Stable Before $\overline{WRITE}$	50		0		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	20		0		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		250		ns
t_{DW}	Data Set Up Time for $\overline{WRITE}$	300		150		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		0		ns
t_{WCY}	Write Cycle Time	1		1		μs

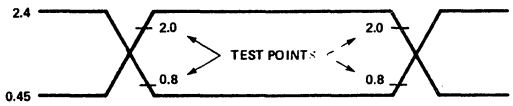
Notes:
1. 8279, V_{CC} = +5V ±5%; 8279-5, V_{CC} = +5V ±10%.
2. 8279, C_L = 100pF; 8279-5, C_L = 150pF.

Other Timings:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
$t_{\phi W}$	Clock Pulse Width	230		120		nsec
t_{CY}	Clock Period	500		320		nsec

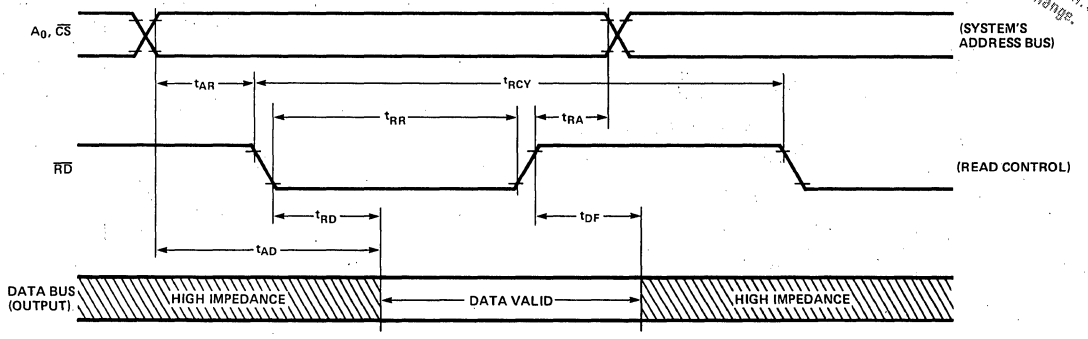
Keyboard Scan Time:	5.1 msec	Digit-on Time:	480 μsec
Keyboard Debounce Time:	10.3 msec	Blanking Time:	160 μsec
Key Scan Time:	80 μsec	Internal Clock Cycle:	10 μsec
Display Scan Time:	10.3 msec		

Input Waveforms For A.C. Tests

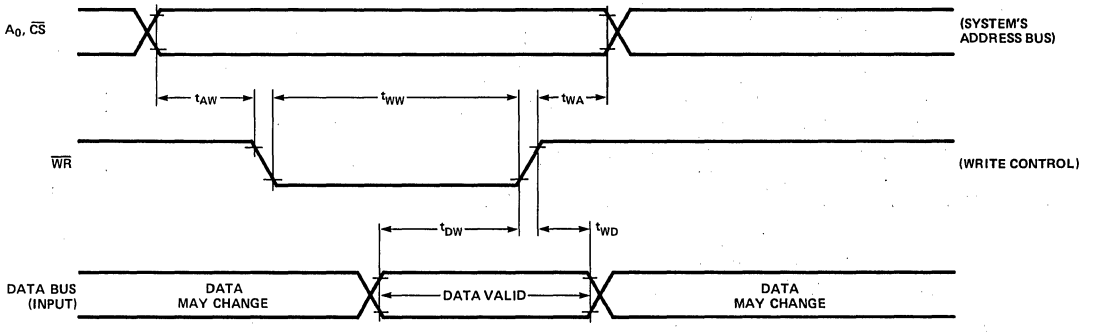


WAVEFORMS

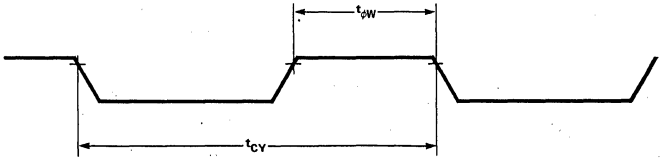
Read Operation



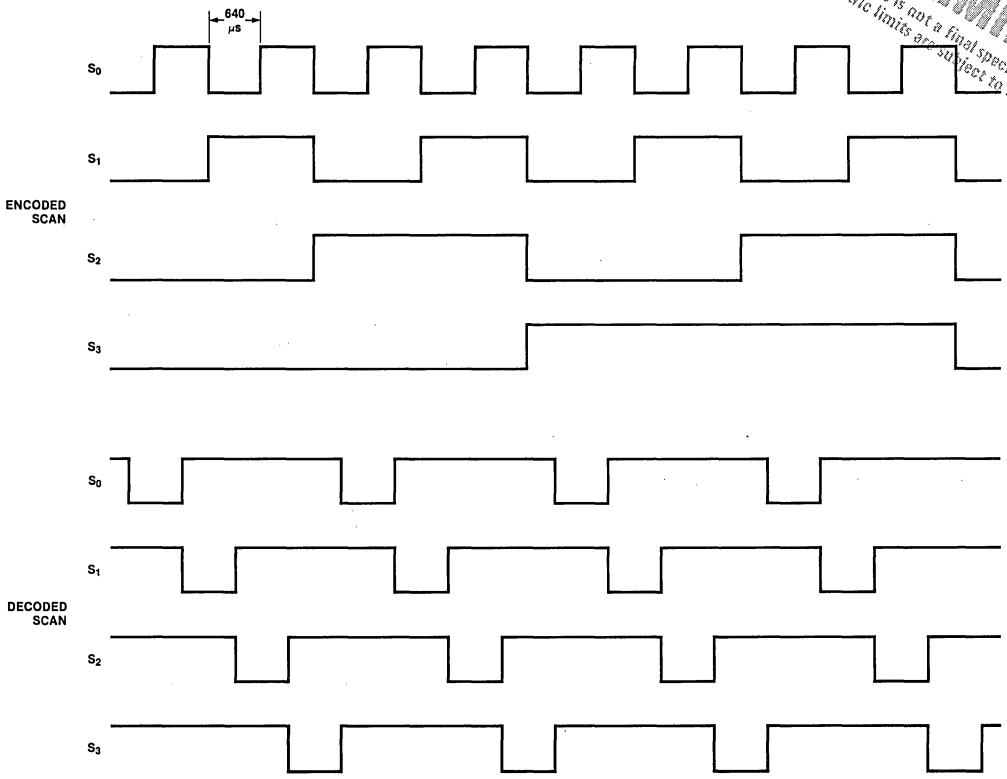
Write Operation



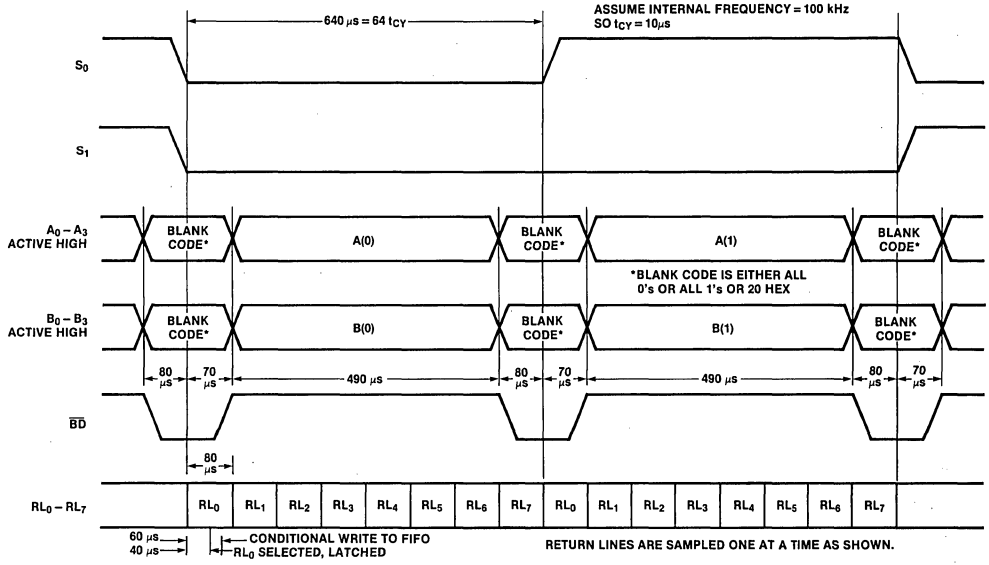
Clock Input



SCAN WAVEFORMS



DISPLAY WAVEFORMS



NOTE: SHOWN IS ENCODED SCAN LEFT ENTRY
S_2-S_3 ARE NOT SHOWN BUT THEY ARE SIMPLY S_1 DIVIDED BY 2 AND 4

intel®

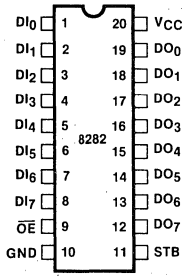
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

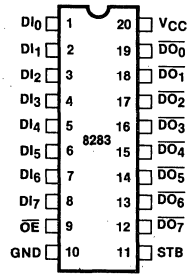
8282/8283
OCTAL LATCH

- **Fully Parallel 8-Bit Data Register and Buffer**
- **Transparent during Active Strobe**
- **Supports 8080, 8085, 8048, and 8086 Systems**
- **High Output Drive Capability for Driving System Data Bus**
- **3-State Outputs**
- **20-Pin Package with 0.3" Center**
- **No Output Low Noise when Entering or Leaving High Impedance State**

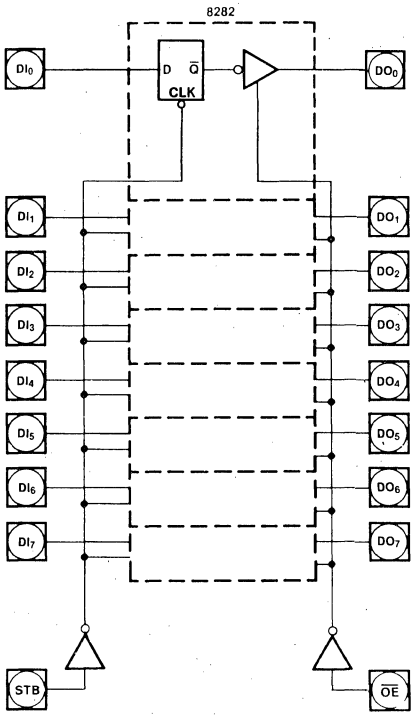
The 8282 and 8283 are 8-bit bipolar latches with 3-state output buffers. They can be used to implement latches, buffers, or multiplexers. The 8283 inverts the input data at its outputs while the 8282 does not. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with these devices.

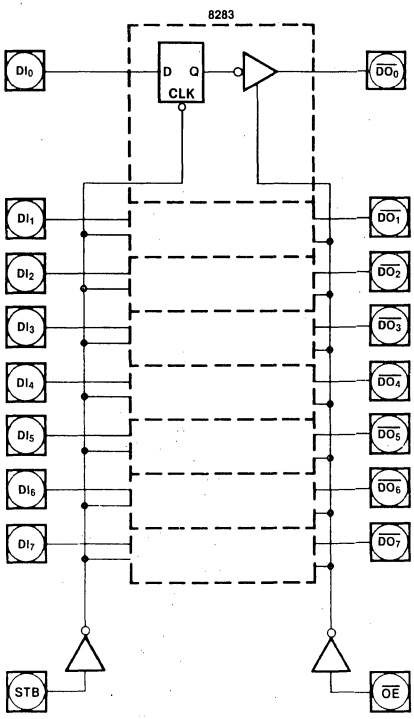
PIN CONFIGURATIONS





LOGIC DIAGRAMS





PIN NAMES

DI_0-DI_7	DATA IN
DO_0-DO_7	DATA OUT
$\overline{OE}$	OUTPUT ENABLE
STB	STROBE

intel®

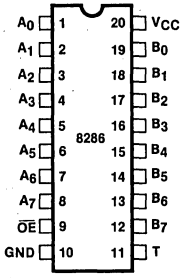
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

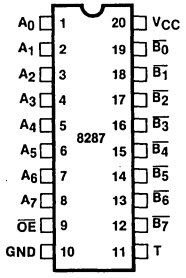
8286/8287
OCTAL BUS TRANSCEIVER

- **Data Bus Buffer Driver for MCS-86™, MCS-80™, MCS-85™, and MCS-48™ Families**
- **High Output Drive Capability for Driving System Data Bus**
- **Fully Parallel 8-Bit Transceivers**
- **3-State Outputs**
- **20-Pin Package with 0.3" Center**
- **No Output Low Noise when Entering or Leaving High Impedance State**

The 8286 and 8287 are 8-bit bipolar transceivers with 3-state outputs. The 8287 inverts the input data at its outputs while the 8286 does not. Thus, a wide variety of applications for buffering in microcomputer systems can be met.

PIN CONFIGURATIONS

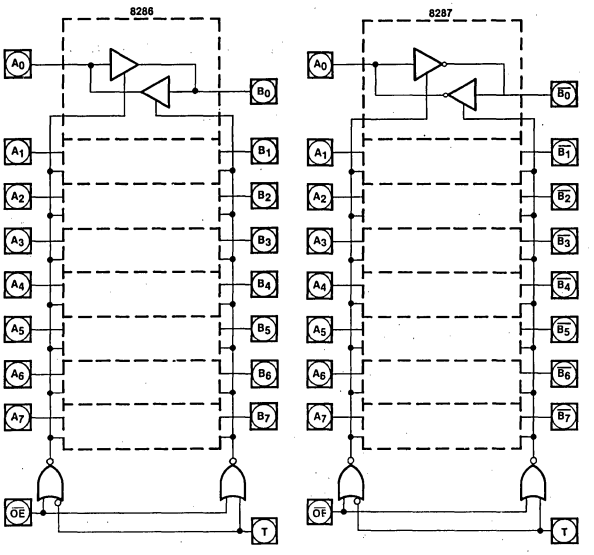




PIN NAMES

A_0-A_7	LOCAL BUS DATA
B_0-B_7	SYSTEM BUS DATA
$\overline{OE}$	OUTPUT ENABLE
T	TRANSMIT

LOGIC DIAGRAMS



intel®

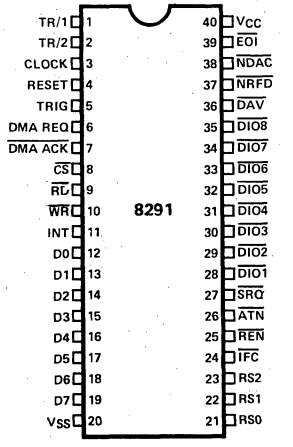
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8291
GPIB TALKER/LISTENER

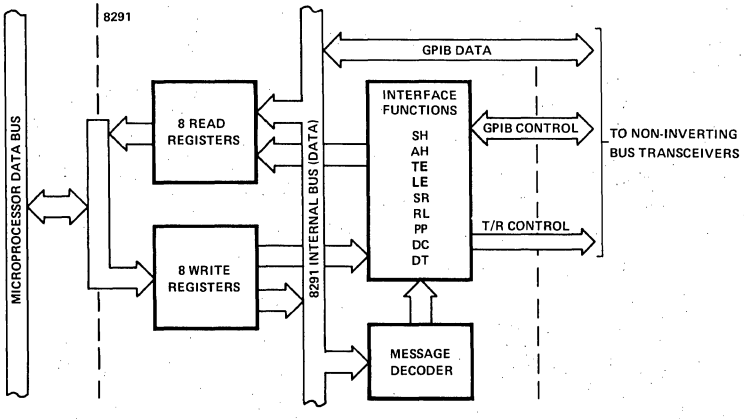
- Designed to Interface Microprocessors (e.g., 8080, 8085, 8086, 8048) to an IEEE Standard 488 Digital Interface Bus
- Programmable Data Transfer Rate
- Complete Source and Acceptor Handshake
- Complete Talker and Listener Functions with Extended Addressing
- Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local Functions
- Selectable Interrupts
- On-Chip Primary and Secondary Address Recognition
- Automatic Handling of Addressing and Handshake Protocol
- Provision for Software Implementation of Additional Features
- 1 – 8 MHz Clock Range
- 16 Registers (8 Read, 8 Write), 2 for Data Transfer, the Rest for Interface Function Control, Status, etc.
- Directly Interfaces to External Non-Inverting Transceivers for Connection to the GPIB
- Provides Three Addressing Modes, Allowing the Chip to be Addressed Either as a Major or a Minor Talker/Listener with Primary or Secondary Addressing
- DMA Handshake Provision Allows for Bus Transfers without CPU Intervention
- Trigger Output Pin
- On-Chip EOS (End of Sequence) Message Recognition Facilitates Handling of Multi-Byte Transfers

The 8291 GPIB Talker/Listener is a microprocessor-controlled chip designed to interface microprocessors (e.g., 8048, 8080, 8085, 8086) to an IEEE Standard 488 Instrumentation Interface Bus. It implements all of the Standard's interface functions except for the controller.

PIN CONFIGURATION



BLOCK DIAGRAM

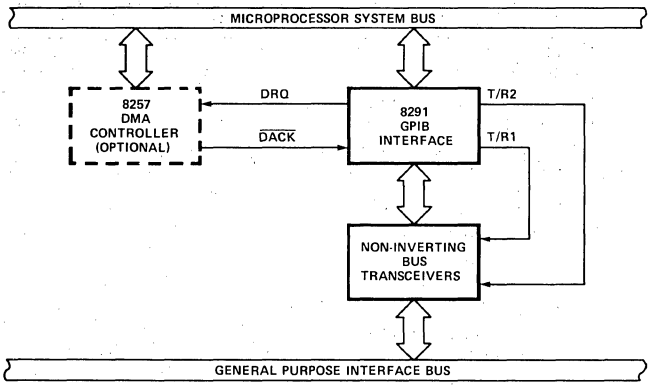


PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
D_0-D_7	I/O	12-19	Data bus port, to be connected to microprocessor data bus.
RS_0-RS_2	I	21-23	Register select inputs, to be connected to three non-multiplexed microprocessor address bus lines. Select which of the 8 internal read (write) registers will be read from (written into) with the execution of $\overline{RD}$ ($\overline{WR}$).
$\overline{CS}$	I	8	Chip select. When low, enables reading from or writing into the register selected by RS_0-RS_2.
$\overline{RD}$	I	9	Read strobe. When low, selected register contents are read by the CPU.
$\overline{WR}$	I	10	Write strobe. When low, data is written into the selected register.
INT ($\overline{INT}$)	O	11	Interrupt request to the microprocessor, set high for request and cleared when the appropriate register is accessed by the CPU. May be software configured to be active low.
DMA REQ	O	6	DMA request, normally low, set high to indicate byte output or byte input, in DMA mode; reset by DMA ACK.
$\overline{DMA\ ACK}$	I	&	DMA acknowledge. When low, resets DMA REQ and selects data in/data out register for DMA data transfer (actual transfer done by $\overline{RD}$/$\overline{WR}$ pulse).
TRIG	O	5	Trigger output, normally low; generates a triggering pulse corresponding to the GET command.
CLOCK	I	3	External clock input, used for internal time delays generator. May be any speed in 1-8 MHz range.
RESET	I	4	Reset input. When high, forces the device into an "Idle" (initialization) mode. The device will remain at "Idle" until released by the microprocessor.
$\overline{DIO_1}$-$\overline{DIO_8}$	I/O	28-35	8-bit GPIB data port, used for bidirectional data byte transfer between 8291 and GPIB via non-inverting external line transceivers.
$\overline{DAV}$	I/O	36	Data valid; GPIB handshake control line. Indicates the availability and validity of information on the DIO lines.
$\overline{NRFD}$	I/O	37	Not ready for data; GPIB handshake control line. Indicates the condition of readiness of device(s) connected to the bus to accept data.
$\overline{NDAC}$	I/O	38	Not data accepted; GPIB handshake control line. Indicates the condition of acceptance of data by the device(s) connected to the bus.
$\overline{ATN}$	I	26	Attention; GPIB command line. Specifies how data on DIO lines are to be interpreted.
$\overline{IFC}$	I	24	Interface clear; GPIB command line. Places the interface functions in a known quiescent state.
$\overline{SRQ}$	O	27	Service request; GPIB command line. Indicates the need for attention and requests an interruption of the current sequence of events on the GPIB.
$\overline{REN}$	I	25	Remote enable; GPIB command line. Selects (in conjunction with other messages) remote or local control of the device.
$\overline{EOI}$	I/O	39	End or identify; GPIB command line. Indicates the end of a multiple byte transfer sequence or, in conjunction with ATN, addresses the device during a polling sequence.
T/R1	O	1	External transceivers control line. Set high to indicate output data/signals on the DIO_1-DIO_8 and DAV lines and input signals on the NRFD and NDAC lines (active source handshake). Set low to indicate input data/signals on the DIO_1-DIO_8 and DAV lines and output signals on the NRFD and NDAC lines (active acceptor handshake).
T/R2	O	2	External transceivers control line. Set high to indicate output signals on the EOI line. Set low to indicate expected input signal on the EOI line during parallel poll.
V_{CC}	P.S.	40	Positive power supply (5V ± 10%).
GND	P.S.	20	Potential ground circuit.

Note: all signals on the 8291 pins are specified with positive logic. However, IEEE 488 specifies negative logic on its 16 signal lines.

8291 SYSTEM DIAGRAM



THE GENERAL PURPOSE INTERFACE BUS (GPIB)

The General Purpose Interface Bus (GPIB) is defined in the IEEE Standard 488-1975 "Digital Interface for Programmable Instrumentation." Although a knowledge of this standard is assumed, Figure 1 provides the bus structure for quick reference. Also, Tables 1 and 2 reference the interface state mnemonics and the interface messages respectively. Modified state diagrams for the 8291 are presented in Appendix A.

GENERAL DESCRIPTION

The 8291 is a microprocessor controlled device designed to interface microprocessors e.g., 8048, 8080, 8085, 8086 to the GPIB. It implements all of the interface functions defined in the IEEE 488 Standard. If an implementation of the Standard's Controller function is desired, it can be connected with an Intel® 8292 to form a complete interface.

The 8291 handles communication between a microprocessor controlled device and the GPIB. Its capabilities include data transfer, handshake protocol, talker/listener addressing procedures, device clearing and triggering, service request, and both serial and parallel polling schemes. In most procedures, it does not disturb the microprocessor unless a byte is waiting on input or a byte sent on output (output buffer empty).

The 8291 architecture includes 16 registers. Eight of these registers may be written into by the microprocessor. The other eight registers may be read by the microprocessor. One each of these read and write registers is for direct data transfers. The rest of the write registers control the various features of the chip, while the rest of the read registers provide the microprocessor with a monitor of GPIB states, various bus conditions, and device conditions.

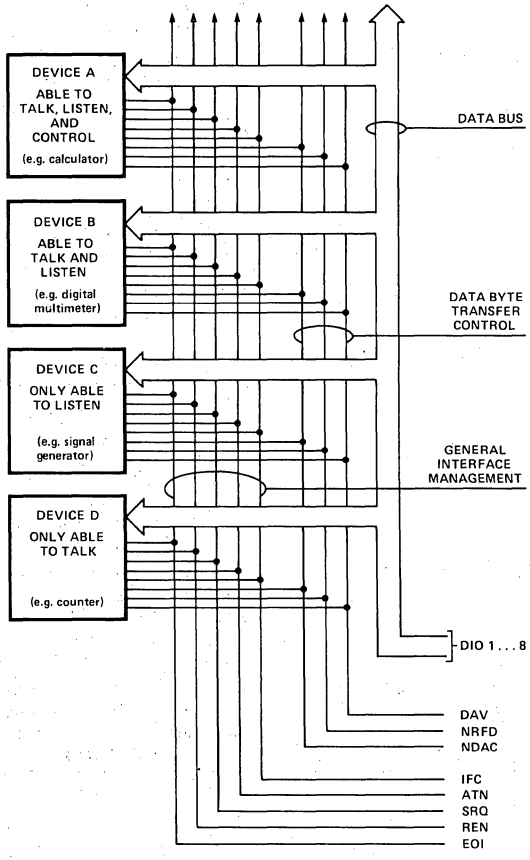


Figure 1. Interface Capabilities and Bus Structure.

GPIB Addressing

Each device connected to the GPIB must have at least one address whereby the controller device in charge of the bus can configure it to talk, listen, or send status. An 8291 implementation of the GPIB offers the user three addressing modes from which the device can be initialized for each application. The first of these modes allows for the device to have two separate primary addresses. The second mode allows the user to implement a single talker/listener with a two byte address (primary address + secondary address). The third mode again allows for two distinct addresses but in this instance, they can each have a two-byte address. However, this mode requires that the secondary addresses be passed to the microprocessor for verification. These three addressing schemes are described in more detail in the discussion of the Address registers.

TABLE 1.
IEEE 488 INTERFACE STATE MNEMONICS

Mnemonic	State Represented
ACDS	Accept Data State
ACRS	Acceptor Ready State
AIDS	Acceptor Idle State
ANRS	Acceptor Not Ready State
APRS	Affirmative Poll Response State
AWNS	Acceptor Wait for New Cycle State
CACS	Controller Active State
CADS	Controller Addressed State
CAWS	Controller Active Wait State
CIDS	Controller Idle State
CPPS	Controller Parallel Poll State
CPWS	Controller Parallel Poll Wait State
CSBS	Controller Standby State
CSNS	Controller Service Not Requested State
CSRS	Controller Service Requested State
CSWS	Controller Synchronous Wait State
CTRS	Controller Transfer State
DCAS	Device Clear Active State
DCIS	Device Clear Idle State
DTAS	Device Trigger Active State
DTIS	Device Trigger Idle State
LACS	Listener Active State
LADS	Listener Addressed State
LIDS	Listener Idle State
LOCS	Local State
LPAS	Listener Primary Addressed State
LPIS	Listener Primary Idle State
LWLS	Local With Lockout State
NPRS	Negative Poll Response State

Mnemonic	State Represented
PACS	Parallel Poll Addressed to Configure State
PPAS	Parallel Poll Active State
PPIS	Parallel Poll Idle State
PPSS	Parallel Poll Standby State
PUCS	Parallel Poll Unaddressed to Configure State
REMS	Remote State
RWLS	Remote With Lockout State
SACS	System Control Active State
SDYS	Source Delay State
SGNS	Source Generate State
SIAS	System Control Interface Clear Active State
SIDS	Source Idle State
SIIS	System Control Interface Clear Idle State
SINS	System Control Interface Clear Not Active State
SIWS	Source Idle Wait State
SNAS	System Control Not Active State
SPAS	Serial Poll Active State
SPIS	Serial Poll Idle State
SPMS	Serial Poll Mode State
SRAS	System Control Remote Enable Active State
SRIS	System Control Remote Enable Idle State
SRNS	System Control Remote Enable Not Active State
SRQS	Service Request State
STRS	Source Transfer State
SWNS	Source Wait for New Cycle State
TACS	Talker Active State
TADS	Talker Addressed State
TIDS	Talker Idle State
TPIS	Talker Primary Idle State

------ The Controller function is implemented on the Intel® 8292.

TABLE 2.
IEEE 488 INTERFACE MESSAGE REFERENCE LIST

Mnemonic	Message	Interface Function(s)
LOCAL MESSAGES RECEIVED (By Interface Functions)		
*gts	go to standby	C
ist	individual status	PP
lon	listen only	L, LE
lpe	local poll enable	PP
nba	new byte available	SH
pon	power on	SH,AH,T,TE,L,LE,SR,RL,PP,C
rdy	ready	AH
*rpp	request parallel poll	C
*rsc	request system control	C
rsv	request service	SR
rtl	return to local	RL
*sic	send interface clear	C
*sre	send remote enable	C
*tca	take control asynchronously	C
*tcs	take control synchronously	AH, C
ton	talk only	T, TE
REMOTE MESSAGES RECEIVED		
ATN	Attention	SH,AH,T,TE,L,LE,PP,C
DAB	Data Byte	(Via L, LE)
DAC	Data Accepted	SH
DAV	Data Valid	AH
DCL	Device Clear	DC
END	End	(via L, LE)
GET	Group Execute Trigger	DT
GTL	Go to Local	RL
IDY	Identify	L,LE,PP
IFC	Interface Clear	T,TE,L,LE,C
LLO	Local Lockout	RL
MLA	My Listen Address	L,LE,RL,T,TE
MSA	My Secondary Address	TE,LE,RL
MTA	My Talk Address	T,TE,L,LE
OSA	Other Secondary Address	TE
OTA	Other Talk Address	T, TE
PCG	Primary Command Group	TE,LE,PP
†PPC	Parallel Poll Configure	PP
†[PPD]	Parallel Poll Disable	PP
†[PPE]	Parallel Poll Enable	PP
*PPR_N	Parallel Poll Response N	(via C)
†PPU	Parallel Poll Unconfigure	PP
REN	Remote Enable	RL
RFD	Ready for Data	SH
RQS	Request Service	(via L, LE)
[SDC]	Select Device Clear	DC
SPD	Serial Poll Disable	T, TE
SPE	Serial Poll Enable	T, TE
*SQR	Service Request	(via C)
STB	Status Byte	(via L, LE)
*TCT or [TCT]	Take Control	C
UNL	Unlisten	L, LE

*These messages are handled only by Intel's 8292.
†Undefined commands which may be passed to the microprocessor.

TABLE 2. (Cont'd)
IEEE 488 INTERFACE MESSAGE REFERENCE LIST

Mnemonic	Message	** Interface Function(s)
REMOTE MESSAGES SENT		
ATN	Attention	C
DAB	Data Byte	(via T, TE)
DAC	Data Accepted	AH
DAV	Data Valid	SH
DCL	Device Clear	(via C)
END	End	(via T)
GET	Group Execute Trigger	(via C)
GTL	Go to Local	(via C)
IDY	Identify	C
IFC	Interface Clear	C
LLO	Local Lockout	(via C)
MLA or [MLA]	My Listen Address	(via C)
MSA or [MSA]	My Secondary Address	(via C)
MTA or [MTA]	My Talk Address	(via C)
OSA	Other Secondary Address	(via C)
OTA	Other Talk Address	(via C)
PCG	Primary Command Group	(via C)
PPC	Parallel Poll Configure	(via C)
[PPD]	Parallel Poll Disable	(via C)
[PPE]	Parallel Poll Enable	(via C)
PPR_N	Parallel Poll Response N	PP
PPU	Parallel Poll Unconfigure	(via C)
REN	Remote Enable	C
RFD	Ready for Data	AH
RQS	Request Service	T, TE
[SDC]	Selected Device Clear	(via C)
SPD	Serial Poll Disable	(via C)
SPE	Serial Poll Enable	(via C)
SRQ	Service Request	SR
STB	Status Byte	(via T, TE)
TCT	Take Control	(via C)
UNL	Unlisten	(via C)

**All Controller messages must be sent via Intel's 8292.

8291 Registers

A bit-by-bit map of the 16 registers on the 8291 is presented in Table 3. A more detailed explanation of each of these registers and their functions follows. The access of these registers by the microprocessor is accomplished by using the $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, and RS_0-RS_2 pins.

Register	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	RS_0-RS_2
All Read Registers	0	0	1	CCC
All Write Registers	0	1	0	CCC
Don't Care	1	X	X	XXX

TABLE 3. 8291 REGISTERS

READ REGISTERS								REGISTER SELECT CODE RS2	RS1	RS0	WRITE REGISTERS							
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	0	0	0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
DATA IN											DATA OUT							
CPT	APT	GET	END	DEC	ERR	BO	BI	0	0	1	CPT	APT	GET	END	DEC	ERR	BO	BI
INTERRUPT STATUS 1											INTERRUPT MASK 1							
INT	SPAS	LLO	REM	SPASC	LLOC	REMC	ADSC	0	1	0	0	0	DMAO	DMAI	SPASC	LLOC	REMC	ADSC
INTERRUPT STATUS 2											INTERRUPT MASK 2							
S8	SRQS	S6	S5	S4	S3	S2	S1	0	1	1	S8	rsv	S6	S5	S4	S3	S2	S1
SERIAL POLL STATUS											SERIAL POLL MODE							
ton	lon	EOI	LPAS	TPAS	LA	TA	MJMN	1	0	0	TO	LO	0	0	0	0	ADM1	ADM0
ADDRESS STATUS											ADDRESS MODE							
CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	1	0	1	CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0
COMMAND PASS THROUGH											AUX MODE							
X	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0	1	1	0	ARS	DT	DL	AD5	AD4	AD3	AD2	AD1
ADDRESS 0											ADDRESS 0/1							
X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1	1	1	1	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0
ADDRESS 1											EOS							

Data Registers

DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

DATA-IN REGISTER (0R)

DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

DATA-OUT REGISTER (0W)

The data-in register is used to move data from the GPIB to the microprocessor or to memory when the 8291 is addressed to listen. Incoming information is separately latched by this register, and its contents are not destroyed by a write to the data-out register. The RFD (Ready for Data) message is held false until the byte is removed from the data in register, either by the microprocessor or by DMA. The 8291 then completes the handshake automatically. In RFD/DAV holdoff mode (see Auxiliary Register A), the handshake is not finished until a command is sent telling the 8291 to release the holdoff. In this way, the same byte may be read several times, or an over anxious talker may be held off until all available data has been processed.

When the 8291 is addressed to talk, it uses the data-out register to move data onto the GPIB. Upon a write to this register, the 8291 initiates and completes the handshake while sending the byte out over the bus. When the RFD/DAV holdoff mode is in effect, data is held until the release command is issued. Also, a read of the data-in register does not destroy the information in the data-out register.

Interrupt Registers

CPT	APT	GET	END	DEC	ERR	BO	BI

INTERRUPT STATUS 1 (1R)

CPT	APT	GET	END	DEC	ERR	BO	BI

INTERRUPT MASK 1 (1W)

INT	SPAS	LLO	REM	SPASC	LLOC	REMC	ADSC

INTERRUPT STATUS 2 (2R)

0	0	DMAO	DMAI	SPASC	LLOC	REMC	ADSC

INTERRUPT MASK 2 (2W)

The 8291 can be configured to generate an interrupt to the microprocessor upon the occurrence of any of 12 conditions or events on the GPIB. Upon receipt of an interrupt, the microprocessor must read the Interrupt Status registers to determine which event has occurred, and then execute the appropriate service routine (if necessary). Each of the 12 interrupt status bits has a matching mask bit in the interrupt mask registers. These mask bits are used to select the events that will cause the INT pin to be asserted. Writing a logic "1" into any of these bits enables the corresponding interrupt status bits to generate an interrupt. Bits in the Interrupt Status registers are set regardless of the states of the mask bits. The Interrupt Status registers are then cleared upon being read or when a local pon (power-on) message is executed. If an event occurs while one of the Interrupt Status registers is being read, the event is typically held until after its register is cleared and then placed in the register.

The mnemonics for each of the bits in these registers and a brief description of their respective functions appears in Table 4. This table also indicates how each of the interrupt bits is set.

TABLE 4. Interrupt Bits

Indicates Undefined Commands	CPT	An undefined command has been received.
Set by (TPAS + LPAS)•SCG•ACDS•MODE 3	APT	A secondary address must be passed through to the microprocessor for recognition.
Set by DTAS	GET	A group execute trigger has occurred.
Set by (EOS + EOI)•LACS	END	An EOS or EOI message has been received.
Set by DCAS	DEC	Device Clear Active State has occurred.
Set by TACS•nba•DAC•RFD	ERR	Interface error has occurred; no listeners are active.
TACS•(SWNS + SGNS)	BO	A byte has been output.
Set by LACS•ACDS	BI	A byte has been input.
Shows status of the INT pin	INT	These are status only. They will not generate interrupts, nor do they have corresponding mask bits.
The device has been enabled for a serial poll	SPAS	
The device is in local lock out state. (LWLS+RWLS)	LLO	
The device is in a remote state. (REMS+RWLS)	REM	
SPAS ⇄ $\overline{\text{SPAS}}$	SPASC	Serial Poll Active State change interrupt
LLO ⇄ NO LLO	LLOC	Local lock out change interrupt.
Remote ⇄ Local	RLC	Remote/Local change interrupt.
Addressed ⇄ Unaddressed	ADSC	Address status change interrupt.*

*In ton (talk-only) and lon (listen-only) modes, no ADSC interrupt is generated.

The BO and BI interrupts enable the user to perform data transfer cycles. BO indicates that a byte has been sent to the GPIB and a new data byte may be written into the Data Out register. It is set by the occurrence of TACS • (SWNS + SGNS). Hence, it is reset when a data byte is written into the Data Out register, when ATN is asserted on the GPIB, or when the device stops being addressed to talk. Similarly, BI is set when an input byte is accepted into the 8291 and reset when the microprocessor reads the Data In register. BO and BI are also reset by pon (power-on local message) and by a read of the Interrupt Status 1 register. However, if it is so desired, data transfer cycles may be performed without reading the Interrupt Status 1 register if all interrupts except for BO or BI are masked; BO and BI will automatically reset after each byte is transferred.

If the 8291 is used without DMA, the BO and BI interrupts may be enabled through the DMA REQ pin. The DMAO and DMAI bits in the Interrupt Mask 2 register would be the corresponding mask bits for this feature. Thus, implementing this feature, with BO and BI masked from the INT pin, allows for servicing of these interrupts without reading the Interrupt Status registers.

The ERR bit is set to indicate the bus error condition where the 8291 is an active talker, tries sending a byte to the GPIB, but there are no active listeners (e.g., all devices on the GPIB are in AIDS). The logical equivalent of (nba • TACS • DAC • RFD) will set this bit.

The DEC bit is set whenever DCAS has occurred. The user must define a known state to which all device functions will return in DCAS. Typically this state will be a power-on state. However, the state of the device functions at DCAS is at the designer's discretion. It should be noted that DCAS has no effect on the interface functions which are returned to a known state by the IFC (interface clear) message or the pon local message.

The End Interrupt bit may be used by the microprocessor to detect that a multi-byte transfer has been completed. The bit will be set when the 8291 is an active listener (LACS) and either EOS or EOI is received. EOS will generate an interrupt when the byte in the Data In register matches the byte in the EOS register. Otherwise the interrupt will be generated when a true input is detected at the EOI pin of the 8291.

The GET interrupt bit is used by the microprocessor to detect that DTAS has occurred. It is set by the 8291 when the GET message is received while it is addressed to listen. The TRIG output pin of the 8291 is also asserted when the GET message is received. Thus, the basic operation of the device may be started without involving the microprocessor.

The APT interrupt bit indicates to the processor that a secondary address is available in the CPT register for validation. This interrupt will only occur if Mode 3 addressing is in effect. (Refer to the section on addressing.) In Mode 2, secondary addresses will be recognized on the 8291. They will be ignored in Mode 1.

The CPT interrupt bit flags the occurrence of an undefined command and of all secondary commands following an undefined command. The Command pass through feature is enabled by the BO bit of Auxiliary register B.

$$UDC = [UCG + ACG(TADS \bullet \overline{PPC} + LADS \bullet \overline{TCT})] \bullet undefined \bullet BO$$

where:

ACG — Addressed Command Group
UCG — Universal Command Group
SCG — Secondary Command Group

Any message not decoded by the 8291 (not included in the state diagrams in Appendix B) becomes an undefined command. Note from the logic equation that any addressed command is automatically ignored when the 8291 is not addressed.

Undefined commands are read by the CPU from the Command Pass Through Register of the 8291. Until this register is read, the 8291 will hold off the handshake (only if the CPT feature is enabled).

An especially useful feature of the 8291 is its ability to generate interrupts from state transitions in the interface functions. In particular, the lower 4 bits of the Interrupt Status 2 register, if enabled by the corresponding mask bits, will cause an interrupt upon changes in the following states as defined in IEEE 488:

Bit 0	ADSC	change in LIDS or TIDS or MJMN
Bit 1	RLC	change in LOCS or REMS
Bit 2	LLOC	change in LWLS or RWLS
Bit 3	SPASC	change in SPAS

The upper 4 bits of the Interrupt Status 2 register are available to the processor as status bits. Thus, if one of the bits 1-3 generates an interrupt indicating a state change has taken place, the corresponding status bit (bits 5-7) may be read to determine what the new state is. To determine the nature of a change in addressed status (bit 0) the Address Status Register is available to be read. And finally, bit 7 monitors the state of the 8291 INT pin. Logically, it is an OR of all unmasked interrupt status bits. One should note that bits 4-7 of the Interrupt Status 2 Register do not generate interrupts, but are available only to be read as status bits by the processor.

Bits 4 and 5 (DMAI, DMAO) of the Interrupt Mask 2 Register are available to enable direct data transfers between memory and the GPIB, DMAI (DMA in) enables the DMA REQ (DMA request) pin of the 8291 to be asserted upon the occurrence of BI. Similarly, DMAO (DMA out) enables the DMA REQ pin to be asserted upon the occurrence of BO. One might note that the DMA REQ pin may be used as a second interrupt output pin, monitoring BI and/or BO and masked by DMAI and DMAO. One should note that the DMA REQ pin is not affected by a read of the Interrupt Status 1 Register. It is reset whenever a byte is written to the Data Out Register or read from the Data in Register.

To ensure that an interrupt status bit will not be cleared without being read, and will not remain uncleared after being read, the 8291 implements a special interrupt

handling procedures. When an unmasked interrupt bit is set in either of the Interrupt Status Registers, the input of the registers are blocked until the set bit is read and reset by the microprocessor. Thus, potential problems arise when interrupt status changes while the register is being blocked. However, the 8291 stores all new interrupts in a temporary register and transfers them to the appropriate interrupt Status Register after the interrupt has been reset. In the Interrupt Status 1 Register and in ADSC bit, this transfer takes place only if the corresponding bits were read as zeroes. For the other status change bits in the Interrupt Status 2 Register, the transfer will always take place. However, even number of changes in these status bits during blocking time will cause no interrupt.

Serial Poll Registers

S8	SRQS	S6	S5	S4	S3	S2	S1

SERIAL POLL STATUS (3R)

S8	rsv	S6	S5	S4	S3	S2	S1

SERIAL POLL MODE (3W)

The Serial Poll Mode Register is used to establish the status byte that the 8291 sends out on the GPIB data lines when it receives the SPE (Serial Poll Enable) message. Bit 6 of this register is reserved for the rsv (request service) local message. Setting this bit to 1 causes the 8291 to assert its SRQ line, indicating its need for attention from the controller-in-charge of the GPIB. When service has been granted, the bit should be cleared by the microprocessor. The other bits of this register are available for sending status information over the GPIB. Sometime after the microprocessor initiates a request for service by setting bit 6, the controller of the GPIB sends the SPE message and then addresses the 8291 to talk. At this point, one byte of status is returned by the 8291 via the Serial Poll Mode Register.

The Serial Poll Status Register is available for reading the status byte in the Serial Poll Mode Register. The processor may check the status of a request for service by polling bit 6 of this register, which corresponds to SRQS (Service Request State). When a Serial Poll is conducted and the controller-in-charge reads the status byte, the SRQS bit is cleared. The SRQ line is tied to this bit, so that a request for service is terminated when the 8291's status byte is read. The rsv bit of the Serial Poll Mode Register must then be cleared by the microprocessor.

Address Registers

ton	lon	EOI	LPAS	TPAS	LA	TA	MJMN

ADDRESS STATUS (4R)

TO	LO	0	0	0	0	ADM1	ADM0

ADDRESS MODE (4W)

X	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0

ADDRESS 0 (6R)

ARS	DT	DL	AD5	AD4	AD3	AD2	AD1

ADDRESS 0/1 (6W)

X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1

ADDRESS 1 (7R)

The Address Mode Register is used to select one of the five modes of addressing available on the 8291. It determines the way in which the 8291 uses the information in the Address 0 and Address 1 registers:

—In Mode 1, the contents of the Address 0 Register constitute the "Major" talker/listener address while the Address 1 Register represents the "Minor" talker/listener address. In applications where only one address is needed, the major talker/listener is used, and the minor talker/listener should be disabled. Loading an addres via the Address 0/1 Register into Address Registers 0 and 1 enables the major and minor talker/listener functions respectively.

—In Mode 2 the 8291 recognizes two sequential address bytes: a primary followed by a secondary. Both address bytes must be received in order to enable the device to talk or listen. In this manner, Mode 2 addressing implements the extended talker and listener functions as defined in IEEE 488.

To use Mode 2 addressing the primary address must be loaded into the Address 0 Register, and the Secondary address is placed in the Address 1 Register. With both primary and secondary addresses residing on chip, the 8291 can handle all addressing sequences without processor intervention.

—In Mode 3, the 8291 handles addressing just as it does in Mode 1, except that each Major or Minor primary address must be followed by a secondary address. All secondary addresses must be verified by the microprocessor when Mode 3 is used. When the 8291 is in TPAS or LPAS (talker/listener primary addresses state), and it does not recognize the byte on the DIO lines, an APT interrupt is generated (see section on Interrupt Registers) and the byte is available in the CPT (Command Pass-Through) Register. As part of its interrupt service routine, the microprocessor must read the CPT Register and write one of the following responses to the Auxiliary Mode Register:

1. 07H implies a non-valid secondary address
2. 0FH implies a valid secondary address

Setting the "ton" bit generates the local ton (talk-only) message and sets the 8291 to a talk-only mode. This mode allows the device to operate as a talker in an interface system without a controller.

Setting the "lon" bit generates the local lon (listen-only) message and sets the 8291 to a listen-only mode. This mode allows the device to operate as a listener in an interface system without a controller.

The mode of addressing implemented by the 8291 may be selected by writing one of the following bytes to the Address Mode Register:

Register Contents	Mode
10000000	Enable talk only mode (ton)
01000000	Enable listen only mode (lon)
11000000	The 8291 may talk to itself
00000001	Mode 1, (Primary-Primary)
00000010	Mode 2 (Primary-Secondary)
00000011	Mode 3 (Primary/APT-Primary/APT)

The Address Status Register contains information used by the microprocessor to handle its own addressing. This information includes status bits that monitor the address state of each talker/listener, "ton" and "lon" flags which indicate the talk only and listen only states, and an EOI bit which, when set, signifies that the END message came with the last data byte. LPAS and TPAS indicate that the listener or talker primary address has been received. The microprocessor can then use these bits when the secondary address is passed through to determine whether the 8291 is addressed to talk or listen. The LA (listener addressed) bit will be set when the 8291 is in LACS (Listener Active State) or in LADS (Listener Addressed State). Similarly, the TA (Talker Addressed bit will be set to indicate TACS or TADS, but also to indicate SPAS (Serial Poll Active State). The MJMN bit is used to determine whether the information in the other bits applies to the Major or Minor talker/listener. It is set to "1" when the Minor talker/listener is addressed. It should be noted that only one talker/listener may be active at any one time. Thus, the MJMN bit will indicate which, if either, of the talker/listeners is addressed or active.

The Address 0/1 Register is used for specifying the device's addresses according to the format selected in the Address Mode Register. Five bit addresses may be loaded into the Address 0 and Address 1 registers by writing into the Address 0/1 Register. The ARS bit is used to select which of these registers the other seven bits will be loaded into. The DT and DL bits may be used to disable the talker or listener function at the address signified by the other five bits. When Mode 1 addressing is used and only one primary address is desired, both the talker and the listener should be disabled at the Minor address.

As an example of how the Address 0/1 Register might be used, consider an example where two primary addresses are needed in the device. The Major primary address will be selectable only as a talker and the Minor primary address will be selectable only as a listener. This configuration of the 8291 is formed by the following sequence of writes by the microprocessor:

Operation	CS	RD	WR	Data	RS2-RS0
1. Select addressing Mode 1	0	1	0	00000001	100
2. Load major address into Address 0 Register with listener function disabled.	0	1	0	001AAAAA	110
3. Load minor address into Address 1 Register with talker function disabled.	0	1	0	110BBBBB	110

At this point, the addresses AAAAA and BBBBB are stored in the Address 0 and Address 1 registers respectively, and are available to be read by the microprocessor. Thus, it is not necessary to store any address information elsewhere. Also, with the information stored in the Address 0 and Address 1 registers, processor intervention is not required to recognize addressing by the controller. Only in Mode 3, where secondary addresses are passed through, must the processor intervene in the addressing sequence.

Command Pass Through Register

CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0

COMMAND PASS THROUGH (5R)

The Command Pass Through Register is used to transfer undefined 8-bit remote message codes from the GPIB to the microprocessor. When the CPT feature is enabled (bit B0 in Auxiliary Register B), any message not decoded by the 8291 becomes an undefined command. When Mode 3 addressing is used secondary addresses are also passed through the CPT Register. In either case, the 8291 will holdoff the handshake until the microprocessor reads this register and issues the VSCMD auxiliary command.

The CPT and APT interrupts flag the availability of undefined commands and secondary addresses in the CPT Register. The details of these interrupts are explained in the section on Interrupt Registers.

An added feature of the 8291 is its ability to handle undefined secondary commands following undefined primaries. Thus, the number of available commands for user definition or future IEEE 488 definition is significantly increased; one undefined primary command followed by a sequence of as many as 32 secondary commands can be processed. However, it is recommended that users do not define their own commands since such definition would violate IEEE 488.

The recommended use of the 8291's undefined command capabilities is for a controller-configured Parallel Poll. The PPC message is an undefined primary command typically followed by PPE, an undefined secondary command. For details on this procedure, refer to the section on Parallel Poll Protocol.

Auxiliary Mode Register

CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0

AUX MODE (5W)

CNT0—2:CONTROL BITS
COM0—:COMMAND BITS

The Auxiliary Mode Register contains a three-bit control field and a five-bit command field. It is used for several purposes on the 8291:

1. To load "hidden" auxiliary registers on the 8291.
2. To issue commands from the microprocessor to the 8291.
3. To preset an internal counter used to generate T1, delay in the Source Handshake function, as defined in IEEE 488.

Table 4 summarizes how these tasks are performed with the Auxiliary Mode Register. Note that the three control bits determine how the five command bits are interpreted.

TABLE 4

CODE		COMMAND
CONTROL BITS	COMMAND BITS	
000	0CCCC	Execute auxiliary command CCCC
001	0FFFF	Preset internal counter to match external clock frequency of FFFF MHZ (FFFF - binary representation of 1 to 8 MHz)
100	DDDDD	Write DDDDD into auxiliary register A
101	0DDDD	Write DDDD into auxiliary register B
011	$USP_3P_2P_1$	Configure/unconfigure parallel poll $SP_3P_2P_1$ as defined in Std. 488. (Configure if U = 0, Unconfigure if U = 1). This command is the local poll enable (lpe) message when U = 0.

AUXILIARY COMMANDS

Auxiliary commands are executed by the 8291 whenever 0000CCCC is written into the Auxiliary Mode Register, where CCCC is the 4-bit command code.

4-Bit Code	Description
0000	Immediate Execute pon — This command resets the 8291 to a power up state (local pon message as defined in IEEE 488). The following conditions constitute the power up state: 1. All talkers and listeners are disabled. 2. No interrupt status bits are set. The 8291 is designed to power up in certain states as specified in the IEEE 488 state diagrams. Thus, the following states are in effect in the power up state: SIDS, AIDS, TIDS, LIDS, NPRS, LOCS, and PPIS. The "0000" pon is an immediate execute command (a pon pulse). It is also used to release the "initialize" state generated by either an external reset pulse or the "0010" Chip Reset command.
0010	Chip Reset (Initialize) — This command has the same effect as a pulse applied to the Reset pin. (Refer to the section on Reset Procedure.)
0011	Finish Handshake — This command finishes a handshake that was stopped because of a holdoff on RFD or DAV. (Refer to Auxiliary Register A.)
0100	Trigger — A "Group Execute Trigger" is forced by this command. It has the same effect as a GET command issued by the controller-in-charge of the GPIB, but does not cause a GET interrupt.
0101	rtl[1] — This command corresponds to the local rtl message as defined in IEEE 488. The 8291 will go to a local state if local lockout is not in effect.
0110	Send EOI — The EOI line of the 8291 may be asserted with this command. The command causes EOI to go true with the next byte transmitted. The EOI line is then cleared upon completion of the handshake for that byte.
0111, 1111	Non-Valid/Valid Secondary Address or Command (VSCMD) — This command informs the 8291 that the secondary address received by the microprocessor was valid or invalid (0111 → invalid, 1111 → valid). If Mode 3 addressing is used, the processor must field each extended address and respond to it, or the GPIB will hang up. Note that the COM3 bit is the invalid/valid flag. The valid (1111) command is also used to tell the 8291 to continue from the command-pass-through state (immediate execute command).
0001, 1001	Parallel Poll Flag (local "ist" message) — This command sets (1001) or clears (0001) the parallel poll flag. A "1" is sent over the assigned data line (PPR-Parallel Poll Response true) only if the parallel poll flag matches the sense bit from the lpe local message (or indirectly from the PPE message). For a more complete description of the Parallel Poll features and procedures refer to the section on Parallel Poll Protocol.

1. Subsequently the 8291 will include "set rtl" and "clear rtl" commands.

INTERNAL COUNTER

The internal counter determines the delay time allowed for the settling of data on the DIO lines. This delay time is defined as T_1 in IEEE 488 and appears in the Source Handshake state diagram between SDYS and STRS. As such, DAV is asserted T_1 *after* the DIO lines are driven. Consequently, T_1 is a major factor in determining the data transfer rate of the 8291 over the GPIB (T_1 = TWROV2-TWROI5).

When open-collector transceivers are used for connection to the GPIB, T_1 is defined by IEEE 488 to be 2μsec. By writing 0010FFFF into the Auxiliary Mode Register, the counter is preset to match a f_C MHz clock input, where FFFF is the binary representation of N_F ($1 \leq N_F \leq 8$, $N_F = (FFFF)_2$). When $N_F = f_C$, a 2μsec T_1 delay will be generated before each DAV asserted.

$$T_{1(\mu sec)} = \frac{2N_F}{f_C} + t_{SYNC}, \quad 1 \leq N_F \leq 8$$

t_{SYNC} is a synchronization error, greater than zero and smaller than the larger of T clock high and T clock low. (For a 50% duty cycle clock, t_{SYNC} is less than half the clock cycle).

If it is necessary that T_1 be different from 2μsec, N_F may be set to a value other than f_C. In this manner, data transfer rates may be programmed for a given system. In small systems, for example, where transfer rates exceeding GPIB specifications are required, one may set $N_F < f_C$ and decrease T_1.

When tri-state transceivers are used, IEEE 488 allows a higher transfer rate (lower T_1). Use of the 8291 with such transceivers is enabled by setting B2 in Auxiliary Register B. In this case, setting $N_F = f_C$ causes a T_1 delay of 2μsec to be generated for the first byte transmitted — all subsequent bytes will have a delay of 500 nsec.

$$T_1(\text{High Speed})\ \mu sec = \frac{N_F}{2f_C} + t_{SYNC}$$

Thus, setting $N_F = 1$ using a 4 MHz clock will generate for a 50% duty cycle clock (t_{SYNC}<125 nsec.):

$$T_1 = \frac{1}{2.4} + 0.125 = 0.250\ \mu sec = 250\ nsec$$

AUXILIARY REGISTER A

Auxiliary Register A is a "hidden" 5-bit register which is used to enable some of the 8291 features. Whenever a 100 $A_4A_3A_2A_1A_0$ byte is written into the Auxiliary Register, it is loaded with the data $A_4A_3A_2A_1A_0$. Setting the respective bits to "1" enables the following features:

A_0 — RFD/DAV Holdoff on all Data: If the 8291 is listening, RFD will not be sent true until the "finish handshake" auxiliary command is issued by the microprocessor. If the 8291 is talking, DAV is not sent true until the "finish handshake" command is given. In both cases, the holdoff will be in effect for each data byte.

A_1 — RFD/DAV Holdoff on End: This feature enables the holdoff on EOI or EOS (if enabled). However, no holdoff will be in effect on any other data bytes.

A_2 — End on EOS Received: Whenever the byte in the Data In Register matches the byte in the EOS Register, the End interrupt bit will be set in the Interrupt Status 1 Register.

A_3 — Output EOI on EOS Sent: Any occurrence of data in the Data Out Register matching the EOS Register causes the EOI line to be sent true along with the data.

A_4 — EOS Binary Compare: Setting this bit causes the EOS Register to function as a full 8-bit word. When it is not set, the EOS Register is a 7-bit word (for ASCII characters).

If $A_0 = A_1 = 1$, a special "continuous Acceptor Handshake cycling" mode is enabled. This mode should be used only in a controller system configuration, where both the 8291 and the 8292 are used. It provides a continuous cycling through the Acceptor Handshake state diagram, requiring no local messages from the microprocessor; the rdy local message is automatically generated when in ANRS. As such, the 8291 Acceptor Handshake serves as the controller Acceptor Handshake. Thus, the controller cycles through the Acceptor Handshake without delaying the data transfer in progress. When the t_{CS} local message is executed, the 8291 is taken out of the "continuous AH cycling" mode, the GPIB hangs up in ANRS, and a BI interrupt is generated to indicate that control may be taken. A simpler procedure may be used when a "t_{CS} on end of block" is executed; the 8291 may stay in "continuous AH cycling". Upon the end of a block (EOI or EOS received), a holdoff is generated, the GPIB hangs up in ANRS, and control may be taken.

AUXILIARY REGISTER B

Auxiliary Register B is a "hidden" 4-bit register which is used to enable some of the features of the 8291. Whenever a 1010$B_3B_2B_1B_0$ is written into the Auxiliary Mode Register, it is loaded with the data $B_3B_2B_1B_0$. Setting the respective bits to "1" enables the following features:

B_0 — Enable Undefined Command Pass Through: This feature allows any commands not recognized by the 8291 to be handled in software. If enabled, this feature will cause the 8291 to holdoff the handshake when an undefined command is received. The microprocessor must then read the command from the Command Pass Through Register and send the VSCMD auxiliary command. Until the VSCMD command is sent, the handshake holdoff will be in effect.

B_1 — Send EOI in SPAS: This bit enables EOI to be sent with the status byte; EOI is sent true in Serial Poll Active State. Otherwise, EOI is sent false in SPAS.

B_2 — Enable High Speed Data Transfer: This feature may be enabled when tri-state external transceivers are used. The data transfer rate is limited by T_1 (delay time generated in the Source Handshake function), which is defined according to the type of transceivers used. When the "High Speed" feature is enabled, T_1 = 2 microseconds is generated for the first byte transmitted after each true to false transition of ATN. For all subsequent bytes, T_1 = 500 nanoseconds. Refer to the Internal Counter section for an explanation of T_1 duration as a function of B_2 and of clock frequency.

B_3 — Enable Active Low Interrupt: Setting this bit causes the polarity of the INT pin to be reversed, providing an output signal compatible with Intel's MCS-48™. Interrupt registers are not affected by this bit.

PARALLEL POLL PROTOCOL

Writing a $011USP_3P_2P_1$ into the Auxiliary Mode Register will configure (U=0) or unconfigure (U=1) the 8291 for a parallel poll. When U=0, this command is the "lpe" (local poll enable) local message as defined in IEEE 488. The "S" bit is the sense in which the 8291 is configured; only if the Parallel Poll Flag ("ist" local message) matches this bit will the Parallel Poll Response, PPR_N, be sent true. The bits $P_3P_2P_1$ specify which of the eight data lines PPR_N will be sent over. Thus, once the 8291 has been configured for Parallel Poll, whenever it senses both EOI and ATN true, it will automatically compare its PP flag with the sense bit and send PPR_N true or false according to the comparison.

If a PP_2* implementation is desired, the "lpe" and "ist" local messages are all that are needed. Typically, the user will configure the 8291 for Parallel Poll immediately after initialization. During normal operation the microprocessor will set or clear the Parallel Poll Flag (ist) according to the device's need for service. Consequently the 8291 will be set up to give the proper response to IDY (EOI • ATN) without directly involving the microprocessor.

If a PP1* implementation is desired, the undefined command features of the 8291 must be used. In PP1, the 8291 is indirectly configured for Parallel Poll by the active controller on the GPIB. The sequence at the 8291 being configured is as follows:

1. The PPC message is received true. Being an undefined command, it is loaded into the Command Pass Through Register, and a CPT interrupt is sent to the microprocessor. The handshake is automatically held off.

2. The microprocessor reads the CPT Register and sends VSCMD to the 8291, releasing the handshake.

3. Having received an undefined primary command, the 8291 is set up to receive an undefined secondary command, the PPE message. This message is also received into the CPT Register, the handshake is held off, and the CPT interrupt is generated.

4. The microprocessor reads the PPE message and decodes the $SP_3P_2P_1$ information. It then sends the appropriate "lpe" local message to the 8291. Finally, the microprocessor sends VSCMD and the handshake is released.

*As defined in IEEE Standard 488.

End of Sequence (EOS) Register

EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0

EOS REGISTER

The EOS Register and its features offer an alternative to the "Send EOI" auxiliary command. A seven or eight bit byte (ASCII or binary) may be placed in the register to flag the end of a block or read. The type of EOS byte to be used is selected in Auxiliary Register bit A_4.

If the 8291 is a listener, and the "End on EOS Received" is enabled at bit A_2, then an End interrupt is generated in the Interrupt Status 1 Register whenever the byte in the Data-In Register matches the byte in the EOS Register.

If the 8291 is a talker, and the "Output EOI on EOS Sent" is enabled at bit A_3, then the EOI line is sent true with the next data byte whenever the contents of the Data Out Register match the EOS register.

Reset Procedure

The 8291 is reset to an initialization state either by a pulse applied to its Reset pin, or by a reset auxiliary command (02H written into the Auxiliary Command Register). The following conditions are caused by a reset pulse (or local reset command):

1. A "pon" local message as defined by IEEE 488 is held true until the initialization state is released.
2. The Interrupt Status Registers are cleared.
3. Auxiliary Registers A and B are cleared.
4. The Serial Poll Mode Register is cleared.
5. The Parallel Poll Flag is cleared.
6. The EOI bit in the Address Status Register is cleared.
7. N_F in the Internal Counter is set to 8 MHz. This setting causes the longest possible t_1 delay to be generated in the Source Handshake (16 μsec for 1 MHz clock).

The initiallization state is released by an "immediate execute pon" command (00H written into the Auxiliary Command Register).

The suggested initialization sequence is:

1. Apply a reset pulse or send the reset auxiliary command.
2. Set the desired initial conditions by writing into the Interrupt Mask, Serial Poll Mode, Address Mode, Address 0/1, and EOS Registers. Auxiliary Registers A and B, and the internal counter should also be initialized.
3. Send the "immediate execute pon" auxiliary command to release the initialization state.
4. If a PP_2 Parallel Poll implementation is to be used the "lpe" local message may be sent, configuring the 8291 for a Parallel Poll Response on an assigned line. (Refer to the section on Parallel Poll Protocol.)

Using DMA

The 8291 may be connected to the Intel 8257 DMA Controller for DMA operation. The DMA REQ pin of the 8291 requests a DMA byte transfer from the 8257. It is set by BO or BI flip flops, masked by the DMAO and DMAI bits in the Interrupt Mask 2 Register. (After reading, the INT1 register BO and BI interrupts will be cleared but not BO and BI in DREQ equation.)

The $\overline{\text{DMA ACK}}$ pin is driven by the 8257 in response to the DMA request. When $\overline{\text{DMA ACK}}$ is true (active low) it sets CS = RS0 = RS1 = RS2 = 0 such that the RD and WR signals sent by the 8257 refer to the Data In and Data Out Registers. Also, the DMA request line is reset by $\overline{\text{DMA ACK}}$.

DMA input sequence:

1. A data byte is accepted from the GPIB by the 8291.
2. A BI interrupt is generated and DMA REQ is set.
3. $\overline{\text{DMA ACK}}$ is asserted by the 8257 and DMA REQ is reset.
4. $\overline{\text{RD}}$ is driven by the 8257 and the contents of the Data In Register are transferred to MCS bus.
5. The 8291 sends RFD true on the GPIB and proceeds with the Acceptor Handshake protocol.

DMA output sequence:

1. A BO interrupt is generated (indicating that the Data Out Register is empty) and DMA REQ is asserted.
2. $\overline{\text{DMA ACK}}$ is asserted by the 8257 and DMA REQ is reset.
3. $\overline{\text{WR}}$ is driven by the 8257 and a byte is transferred from the MCS bus into the Data Out Register.
4. The 8291 sends DAV true on the GPIB and proceeds with the Source Handshake protocol.

It should be noted that each time the device is addressed, the Address Status Register should be read, and the 8257 should be initialized accordingly. (Refer to the 8257 data sheet available in Intel's Peripheral Design Handbook.)

System Configuration

Microprocessor Bus Connection

The 8291 is 8080, 8048, 8085 and 8086 compatible. The three address pins (RS_0, RS_1, RS_2) should be connected to the non-multiplexed address bus (for example: A_8, A_9, A_{10}). In case of 8080, any address lines may be used.

External Transceivers Connection

8291 IEEE bus pins are TTL compatible. For IEEE Std. bus connection, external transceivers are required. 8291 supplies Transmit/Receive control pins: T/R1 controls DIO_{1-8}, NRFD, NADC and DAV transceivers, T/R2 controls EOI transceiver. IFC, ATN, REN are always inputs and SRQ is always an output.

Logically, TR1 = TACS + SPAS + PPAS;
TR2 = TACS + SPAS.

Refer to 8292 Data Sheet for 8291/8292 system configuration.

DEVICE ELECTRICAL CHARACTERISTICS

D.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5V ± 10%

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL}=2mA (4mA for TR1 pin)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA (-150μA for SRQ pin)
V_{OH-INT}	Interrupt Output High Voltage	2.4		V	I_{OH}=-400μA
		3.5		V	I_{OH}=-50μA
I_{IL}	Input Leakage		10	μA	V_{IN}=0V to V_{CC}
I_{LOL}	Output Leakage Current		-10	μA	V_{OUT}=0.45V
I_{LOH}	Output Leakage Current		10	μA	V_{OUT}=V_{CC}
I_{CC}	V_{CC} Supply Current		180	mA	T_A=0°C

A.C. CHARACTERISTICS

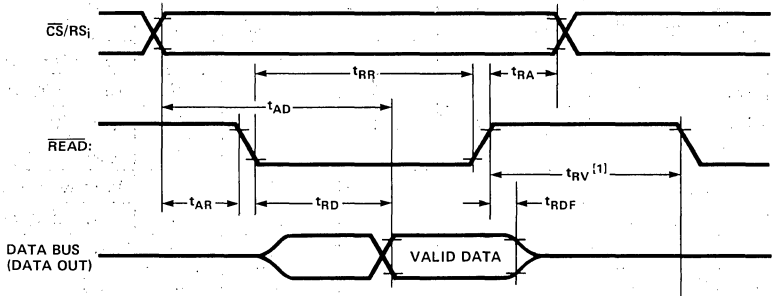
V_{CC} = 5V ± 10%, Commercial: T_A = 0°C to 70°C

Symbol	Parameter	Min.	Max.	Unit
t_{AR}	Address Stable Before $\overline{READ}$	0		nsec[1]
t_{RA}	Address Hold After $\overline{READ}$	0		nsec[1]
t_{RR}	$\overline{READ}$ width	250		nsec[2]
t_{AD}	Address Stable to Data Valid		250	nsec[1]
t_{RD}	$\overline{READ}$ to Data Valid		100	nsec[2]
t_{RDF}	Data Float After $\overline{READ}$	0	60[2]	nsec
t_{AW}	Address Stable Before $\overline{WRITE}$	0		nsec[1]
t_{WA}	Address Hold After $\overline{WRITE}$	0		
t_{WW}	$\overline{WRITE}$ Width	250		nsec[1]
t_{DW}	Data Set Up Time to the Trailing Edge of $\overline{WRITE}$	150		nsec[1]
t_{WD}	Data Hold Time After $\overline{WRITE}$	0		nsec[1]
t_{AKRQ}	$\overline{DACK}$↓ to DREQ↓		130	nsec
t_{DKDA6}	$\overline{DACK}$↓ to Up Data Valid		200	nsec

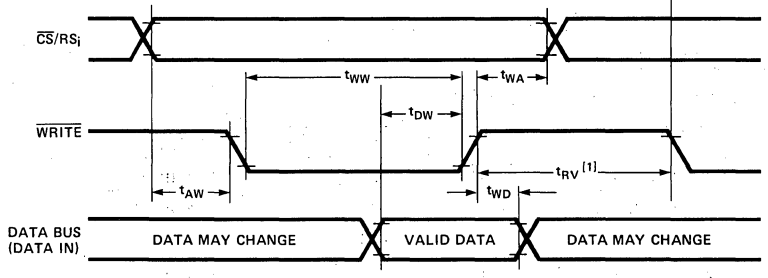
Notes:
1. 8080 System C_{Lmax} = 100pF; C_{Lmin} = 15pF; 3 MHz clock.
2. 8085 System C_L = 150pF; 4 MHz clock.

TIMING WAVEFORMS

READ

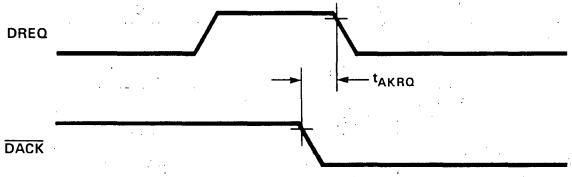


WRITE



NOTES: 1. t_{RV} IS THE TIME BETWEEN READ OR WRITE OPERATIONS WITH THE CHIP SELECTED (CHIP RECOVERY TIME).

DMA



GPIB TIMINGS[1]

Symbol	Parameter	Max.	Unit	Test Conditions
TEOT13	$\overline{EOI}$↓ to TR1↑	90	nsec	PPSS, $\overline{ATN}$=0.45V
TEODI6	$\overline{EOI}$↓ to $\overline{DIO}$ Valid	130	nsec	PPSS, $\overline{ATN}$=0.45V
TEOT12	$\overline{EOI}$↓ to TR1↓	130	nsec	PPSS, $\overline{ATN}$=0.45V
TATND4	$\overline{ATN}$↓ to $\overline{NDAC}$↓	130	nsec	TACS, AIDS
TATT14	$\overline{ATN}$↓ to TR1↓	130	nsec	TACS, AIDS
TATT24	$\overline{ATN}$↓ to TR2↓	130	nsec	TACS, AIDS
TDNVD3-C	$\overline{DAV}$↓ to $\overline{NDAC}$↑	350	nsec	AH, CACS
TNDDV1	$\overline{NDAC}$↑ to $\overline{DAV}$↑	300	nsec	SH, STRS
TNRDV2	$\overline{NRFD}$↑ to $\overline{DAV}$↓	300	nsec	SH, T1 True
TNDDR1	$\overline{NDAC}$↑ to DREQ↑	350	nsec	SH
TDVDR3	$\overline{DAV}$↓ to DREQ↑	350	nsec	AH, LACS, $\overline{ATN}$=2.4V
TDVND2-C	$\overline{DAV}$↑ to $\overline{NDAC}$↓	350	nsec	AH,LACS
TDVNR1-C	$\overline{DAV}$↑ to $\overline{NRFD}$↑	350	nsec	AH, LACS, rdy=True
TRDNR3	$\overline{RD}$↓ to $\overline{NRFD}$↑	500	nsec	AH, LACS
TWRDI5	$\overline{WR}$ to $\overline{DIO}$ Valid	200	nsec	SH, TACS, RS = 0.4V
TWRDV2	$\overline{WR}$↑ to $\overline{DAV}$↓	760	nsec	$\overline{NRFD}$ = 2.4V, RS = 0.4V, SH, TACS, High Speed Transfers Enabled, $N_F = f_C = 8$ MHz

Notes:
1. All GPIB timings are at the pins of the 8291.

Appendix A

MODIFIED STATE DIAGRAMS

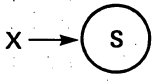
Figure A.1 presents the interface function state diagrams. It is derived from IEEE Std. state diagrams, with the following changes:

A. Controller function omitted.
B. Addressing modes included in T,L state diagrams.

Note that in Mode 3, MSA, OSA are generated only after secondary address validity check by the microprocessor (APT interrupt).

C. All remote messages sent true in each state are indicated.
D. All remote multiline messages decoded are conditioned by ACDS. The multiplication by ACDS is not drawn to simplify the diagrams.

E. The symbol



indicates:

1. When event X occurs, the function will return to state S.
2. X overrides any other transition condition in the function.

Statement 2 simplifies the diagram, avoiding the explicit use of $\overline{X}$ to condition all transitions from S to other states.

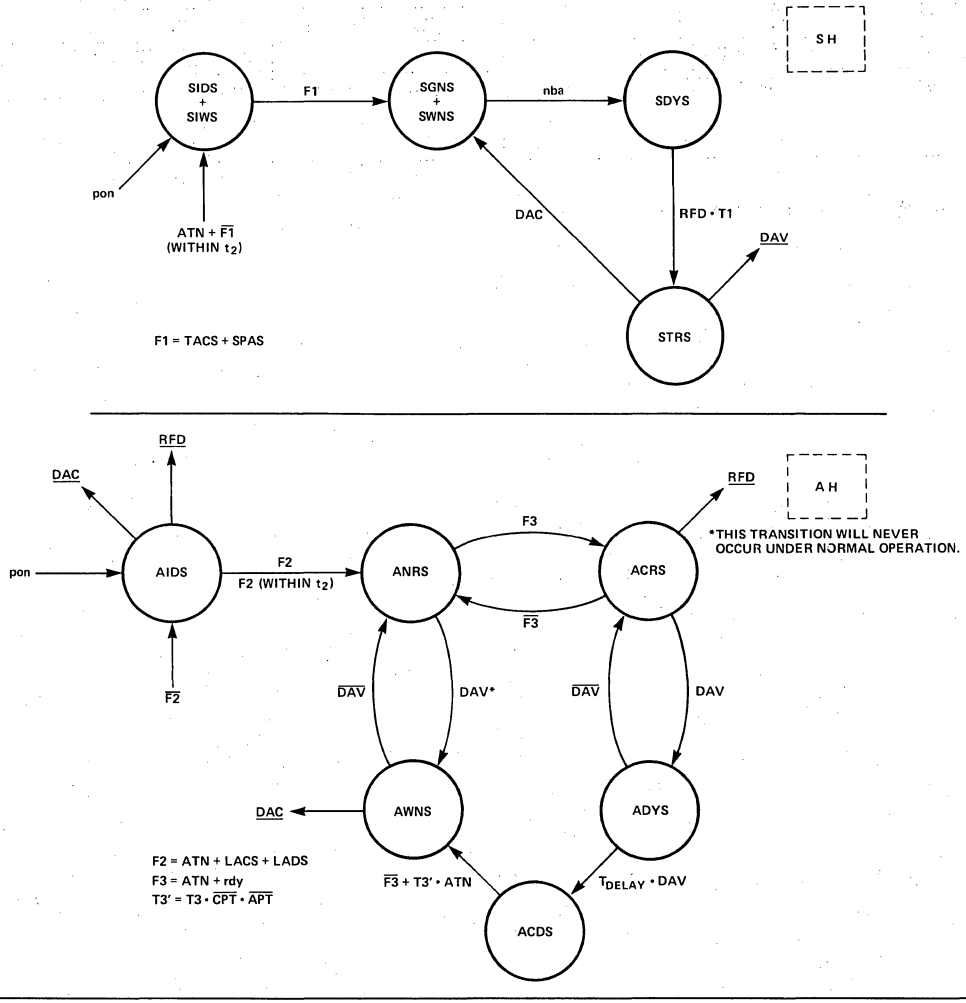


Figure A.1. 8291 State Diagrams (Continued next page)

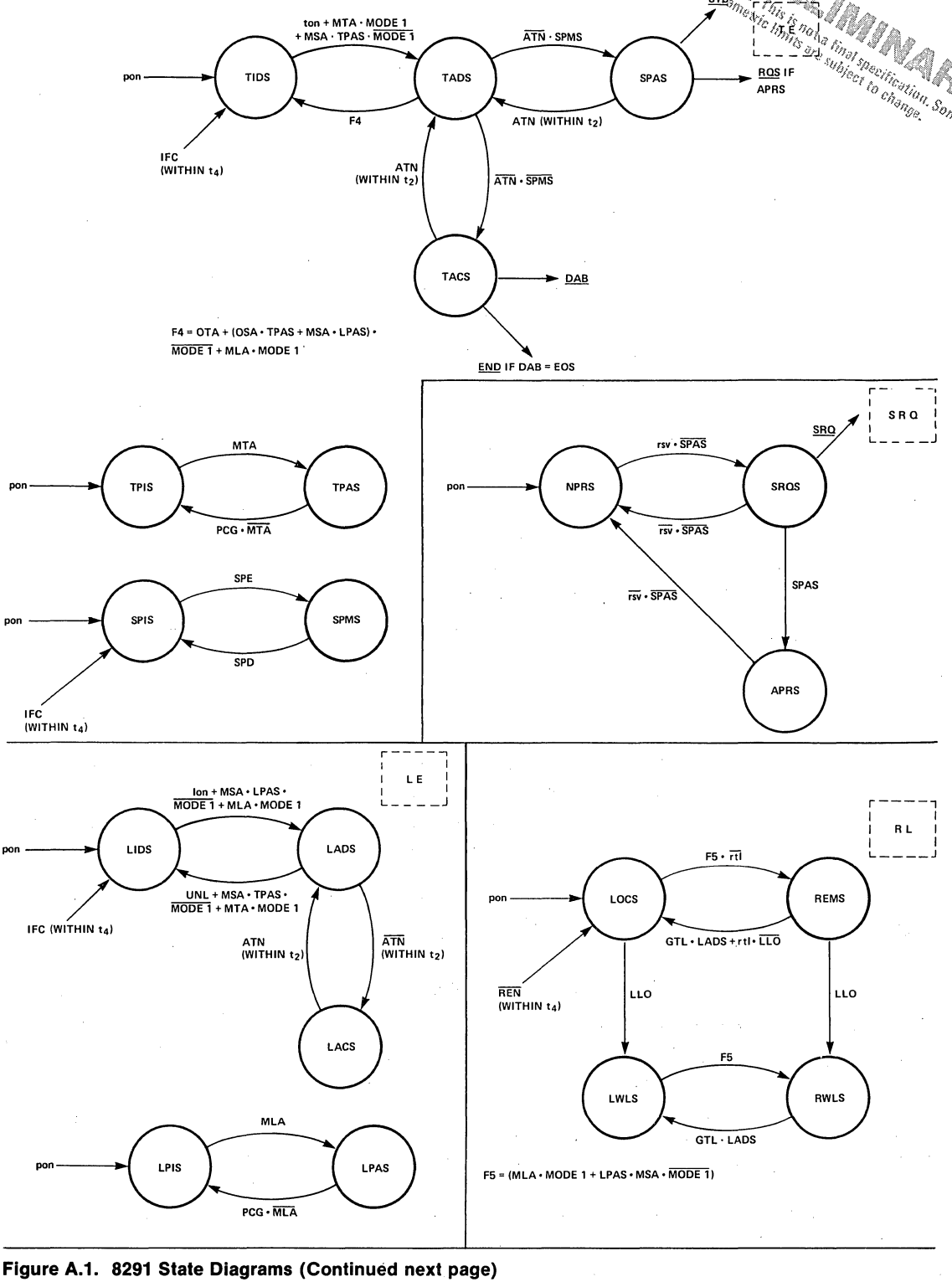


Figure A.1. 8291 State Diagrams (Continued next page)

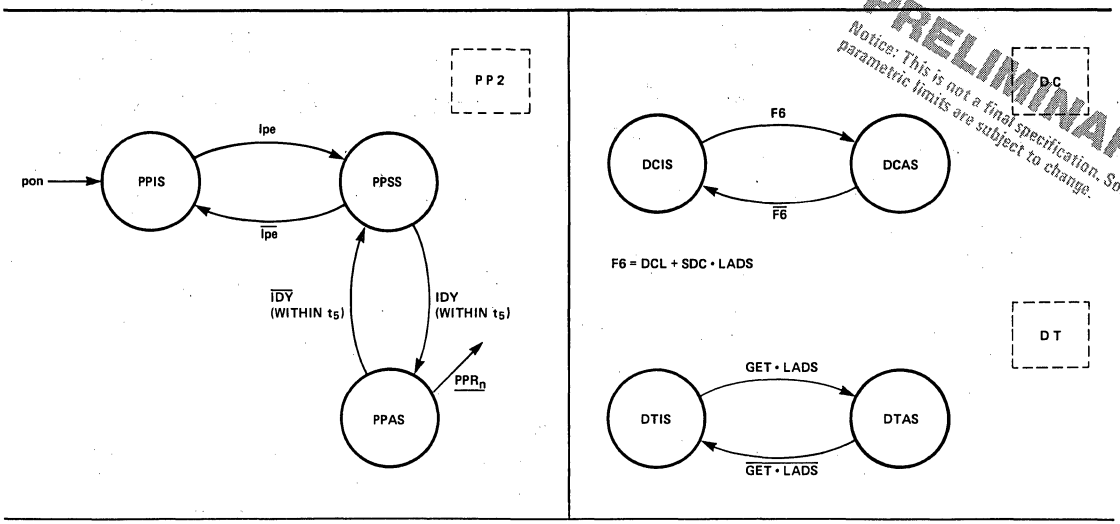


Figure A.1. 8291 State Diagrams

Appendix B

IEEE 488 TIME VALUES

Time Value Identifier*	Function (Applies to)	Description	Value
T_1	SH	Settling Time for Multiline Messages	≥ 2μs†
t_2	LC,$\overline{IC}$,SH,AH,T,L	Response to ATN	≤ 200ns
T_3	AH	Interface Message Accept Time ‡	> 0 δ
t_4	T,TE,L,LE,C,CE	Response to IFC or REN False	< 100μs
t_5	PP	Response to ATN+EOI	≤ 200ns
T_6	C	Parallel Poll Execution Time	≥ 2μs
T_7	C	Controller Delay to Allow Current Talker to see ATN Message	≥ 500ns
T_8	C	Length of IFC or REN False	> 100μs
T_9	C	Delay for EOI**	≥ 1.5μs††

* Time values specified by a lower case t indicate the maximum time allowed to make a state transition. Time values specified by an upper case T indicate the minimum time that a function must remain in a state before exiting.

† If three-state drivers are used on the DIO, DAV, and EOI lines, T_1 may be:
1. ≥ 1100ns
2. Or ≥ 700ns if it is known that within the controller ATN is driven by a three-state driver.
3. Or ≥ 500ns for all subsequent bytes following the first sent after each false transition of ATN (the first byte must be sent in accordance with (1) or (2).
4. Or ≥ 350ns for all subsequent bytes following the first sent after each false transition of ATN under conditions specified in Section 5.2.3 and warning note. See IEEE Standard 488.

‡ Time required for interface functions to accept, not necessarily respond to interface messages.

δ Implementation independent.

** Delay required for EOI, NDAC, and NRFD signal lines to indicate valid states.

†† ≥ 600ns for three-state drivers.

Appendix C
THE THREE WIRE HANDSHAKE

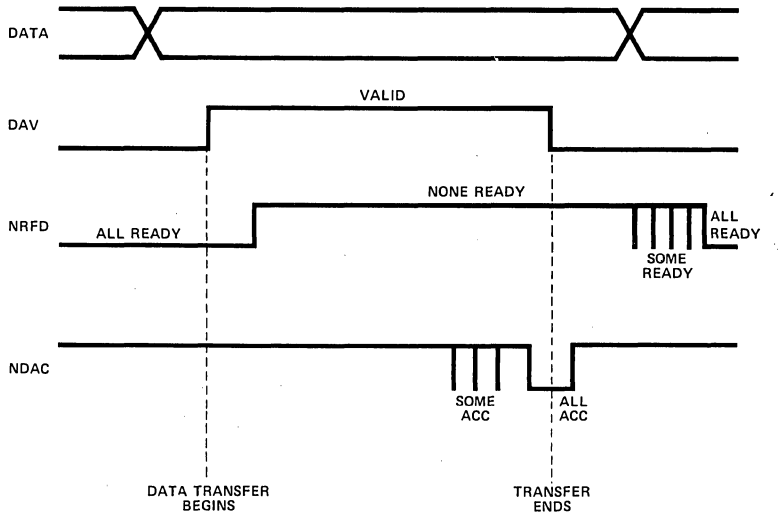
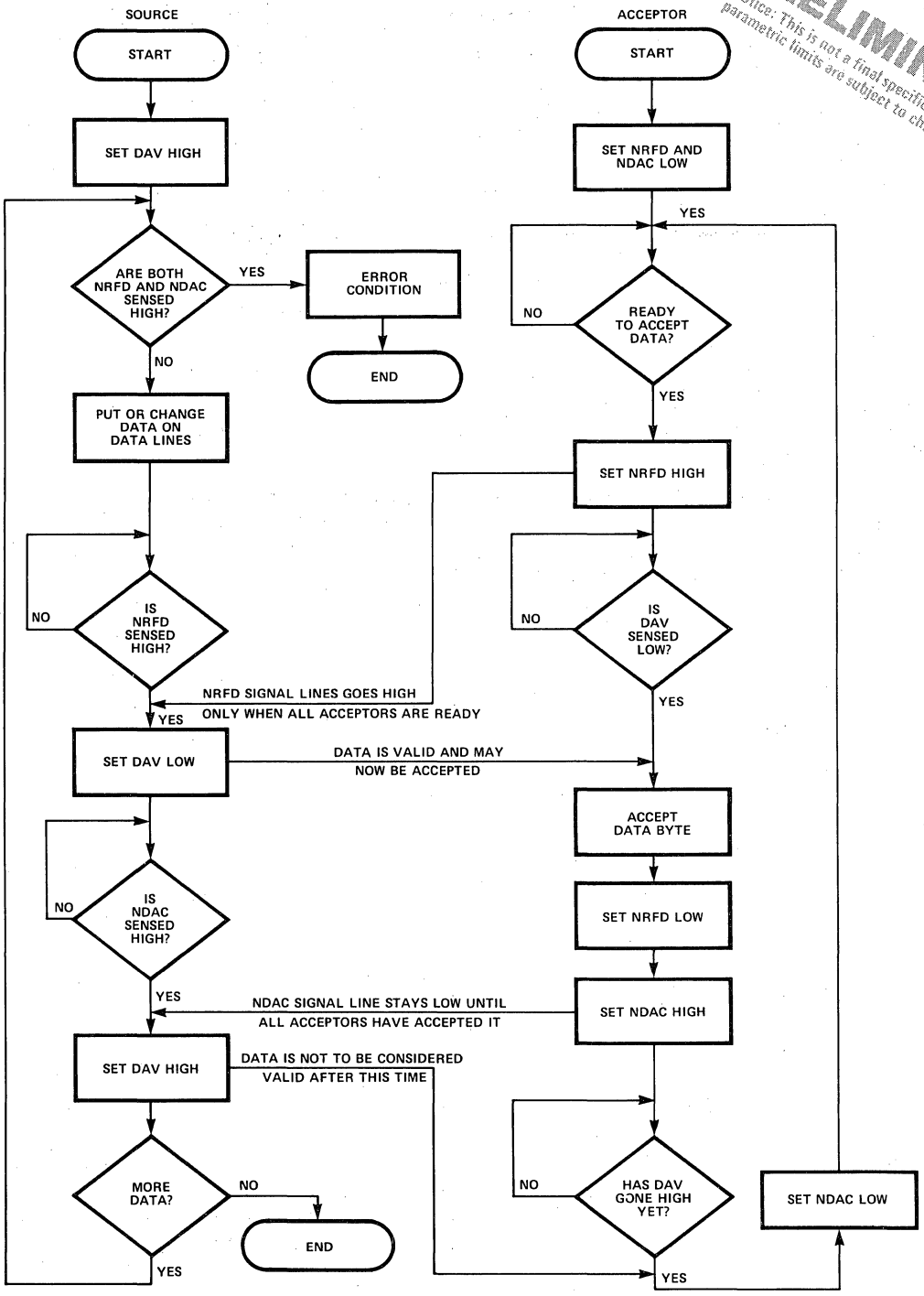


Figure C.1. 3-Wire Handshake Timing.

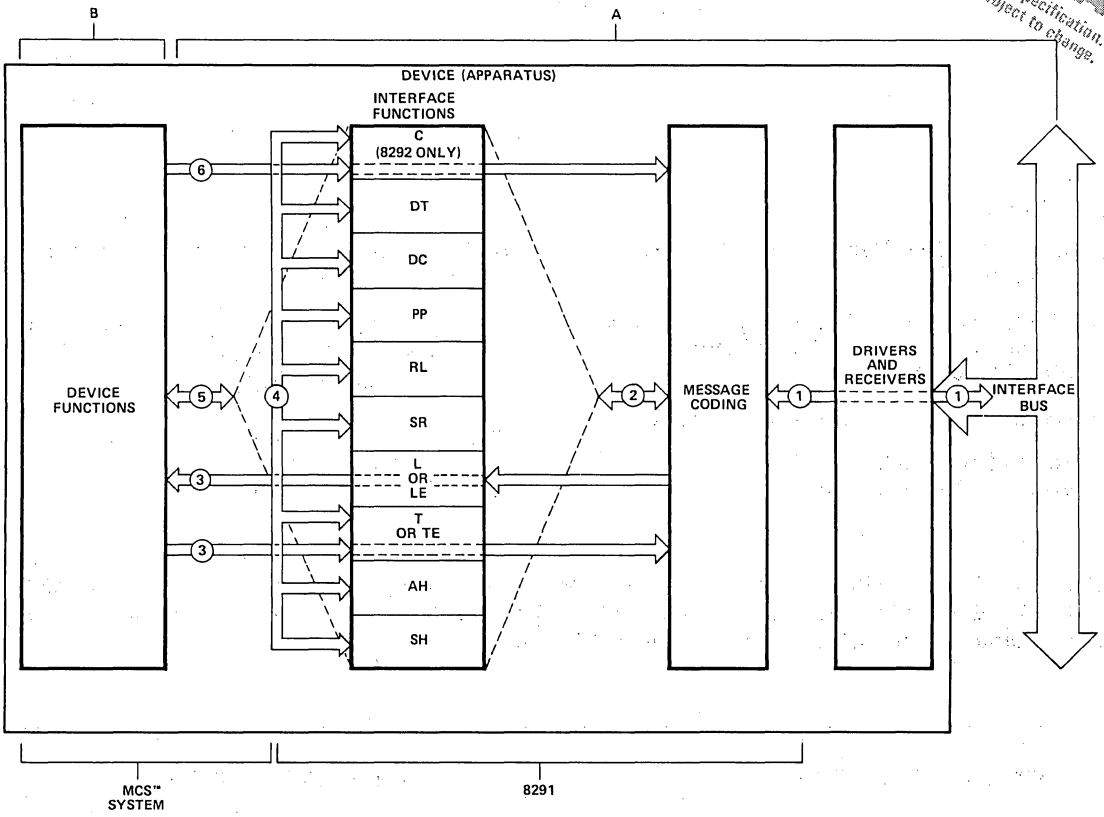


FLOW DIAGRAM OUTLINES SEQUENCE OF EVENTS DURING TRANSFER OF DATA BYTE. MORE THAN ONE LISTENER AT A TIME CAN ACCEPT DATA BECAUSE OF LOGICAL AND CONNECTION OF NRFD AND NDAC LINES.

Figure C.2. Handshake Flowchart.

Appendix D

FUNCTIONAL PARTITIONS



A – CAPABILITY DEFINED BY THE 488-1975 STANDARD.
B – CAPABILITY DEFINED BY THE DESIGNER.
1 – INTERFACE BUS SIGNAL LINES.
2 – REMOTE INTERFACE MESSAGES TO AND FROM INTERFACE FUNCTIONS.
3 – DEVICE DEPENDENT MESSAGES TO AND FROM DEVICE FUNCTIONS.
4 – STATE LINKAGES BETWEEN INTERFACE FUNCTIONS.
5 – LOCAL MESSAGES BETWEEN DEVICE FUNCTIONS AND INTERFACE FUNCTIONS (MESSAGES TO INTERFACE FUNCTIONS ARE DEFINED, MESSAGES FROM INTERFACE FUNCTIONS EXIST ACCORDING TO THE DESIGNER'S CHOICE).
6 – CONTROL MESSAGES (8292 ONLY).

Figure D.1. Functional Partition Within a Device.

intel®

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

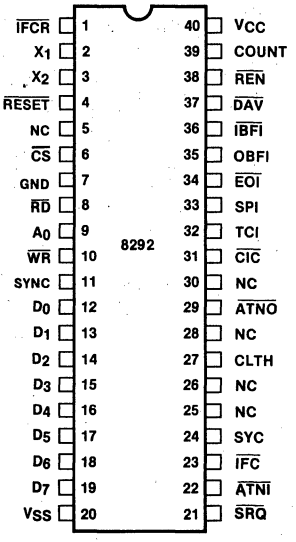
8292
GPIB CONTROLLER

FEATURES:

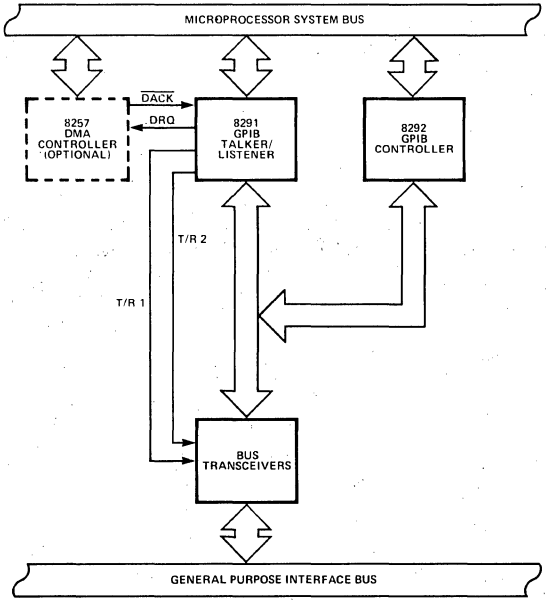
- Complete IEEE Standard 488 Controller Function.
- Interface Clear (IFC) Sending Capability Allows for Seizure of Control and/or Initialization of the Bus.
- Responds to Service Requests (SRQ).
- Sends (REN), Allowing Instruments to Switch to Remote Control.
- Complete Implementation of Transfer Control Protocol.
- Synchronous Control Seizure Prevents the Destruction of any Data Transmission in Progress.
- Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller.

The 8292 GPIB CONTROLLER is a microprocessor-controlled chip designed to connect with the 8291 GPIB TALKER/LISTENER to implement the full IEEE Standard 488 controller function, including transfer control protocol. The 8292 is a pre-programmed UPI-41A.™

PIN CONFIGURATION



8291, 8292 SYSTEM DIAGRAM



PIN DESCRIPTION

Symbol	I/O	Pin No.	Function
D_0-D_7	I/O	12-19	8 bidirectional lines used for communication between the central processor and the 8292's data bus buffers and status register.
A_0	I	9	Address Line—Used to select between the data bus and the status register during read operations and to distinguish between data and commands written into the 8292 during write operations.
$\overline{CS}$	I	6	Chip Select Input—Used to select the 8292 from other devices on the common data bus.
$\overline{RD}$	I	8	I/O write input which allows the master CPU to write to the 8292.
$\overline{WR}$	I	10	I/O read input which allows the master CPU to read from the 8292.
$\overline{RESET}$	I	4	Used to initialize the chip to a known state during power on.
$\overline{DAV}$	I/O	37	DAV Handshake Line—Used only during parallel poll, configures to force the 8291 to accept the parallel poll status bits.
$\overline{ATNI}$	I	22	Attention In—Used by the 8292 to monitor the GPIB ATN control line. It is used during "take control synchronously" execution and during the transfer control procedure.
$\overline{CIC}$	O	31	Controller In Charge—Controls the S/R input of the SRQ bus transceiver. It can also be used to indicate that the 8292 is in charge of the bus.
$\overline{EOI}$	I/O	34	End Or Identify—One of the GPIB management lines, as defined by IEEE Std. 488-1975. Used with ATN as Identify Message during parallel poll.
$\overline{IFC}$	I/O	23	Interface Clear—One of the GPIB management lines, as defined by IEEE Std. 488-1975, places all devices in a known quiescent state.
SYC	I	24	System Controller—Monitors the system controller switch.
OBFI	O	35	Output Buffer Full—Used as an interrupt to the central processor while the output buffer of the 8292 is full. The feature can be enabled and disabled by the interrupt mask register.
$\overline{IBFI}$	O	36	Input Buffer Not Full—Used to interrupt the central processor while the input buffer of the 8292 is empty. This feature is enabled and disabled by the interrupt mask register.
$\overline{ATNO}$	O	29	Attention Out—Controls the ATN control line of the bus through external logic for tcs (take control synchronously) purpose. (ATN is a GPIB control line, as defined by IEEE Std. 488-1975.)
$\overline{SRQ}$	I	21	Service Request—One of the IEEE control lines. Sampled by the 8292 when it is controller in charge, if true—SPI interrupt to the monitor will be generated.
$\overline{REN}$	O	38	The Remote Enable bus signal selects remote or local control of the device on the bus. A GPIB bus management line, as defined by IEEE Std. 488-1975.
TCI	O	32	Task Complete Interrupt—Interrupt to the control processor used to indicate that the task requested was completed by the 8292 and the information requested is ready in the data bus.
SPI	O	33	Special Interrupt—Used as an interrupt on events not initiated by the central processor.
CLTH	O	27	CLEAR LATCH Output—Used to clear the $\overline{IFCR}$ after recognized by the 8292. Usually low (except after hardware Reset), will be pulsed low when $\overline{IFCR}$ is recognized by the 8292.
$\overline{IFCR}$	I	1	IFC Received (latched)—The 8292 monitors the IFC Line (when not system controller) through this pin.
COUNT	I	39	Count Input—When enabled by the proper command the internal counter will count external events through this pin. High to low transition will increment the internal counter by one. The pin is sampled once per three internal instruction cycles (7.5 μsec when using 6 MHz XTAL). It can be used for byte counting when connected to NDAC line, or for block counting when connected to the EOI line.
X_1,X_2	I	2,3	Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
SYNC	O	11	8041A instruction cycle synchronization signal; it is an output clock with a frequency of XTAL ÷ 15.
V_{CC}	P.S.	40	+ 5V supply input.
V_{SS}	P.S.	7,20	Circuit ground potential.

intel®

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8294
DATA ENCRYPTION UNIT

- **Certified by National Bureau of Standards**
- **80 Byte/Sec Data Conversion Rate**
- **64-Bit Data Encryption Using 56-Bit Key**
- **DMA Interface**
- **3 Interrupt Outputs to Aid in Loading and Unloading Data**
- **7-Bit User Output Port**
- **Single 5V ± 10% Power Supply**
- **Peripheral to MCS-86™, MCS-85™, MCS-80™ and MCS-48™ Processors**
- **Implements Federal Information Processing Data Encryption Standard**
- **Encrypt and Decrypt Modes Available**

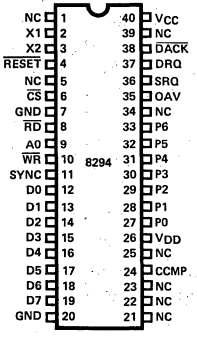
DESCRIPTION

The Intel® 8294 Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294; however, the 56-bit key is user-defined and may be changed at any time.

The 56-bit key and 64-bit message data are transferred to and from the 8294 in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 80 bytes/second. The 8294 also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294 implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.

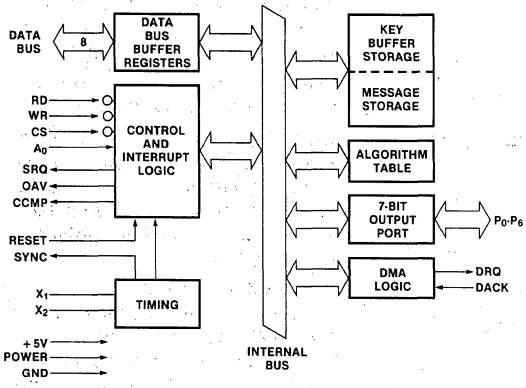
PIN CONFIGURATION



PIN NAMES

PIN NAME	FUNCTION
D_7-D_0	DATA BUS
$\overline{RD}$,$\overline{WR}$	READ,WRITE STROBES
$\overline{CS}$	CHIP SELECT
A_0	CONTROL/DATA SELECT
$\overline{RESET}$	RESET INPUT
X_1,X_2	FREQUENCY REFERENCE INPUT
SYNC	HIGH FREQUENCY OUTPUT
DRQ,$\overline{DACK}$	DMA REQUEST,DMA ACKNOWLEDGE
SRQ,OAV,CCMP	INTERRUPT REQUEST OUTPUTS
P_6-P_0	OUTPUT PORT LINES
V_{CC},V_{DD},GND	+5V POWER,GND
NC	NO CONNECTION

BLOCK DIAGRAM



Pin #	Pin Name	I/O	Pin Description
1	NC	—	No connection.
2	X1	I	Inputs for crystal, L-C or external timing signal to determine internal oscillator frequency.
3	X2		
4	$\overline{RESET}$	I	A low signal to this pin resets the 8294.
5	NC	—	No connection.
6	$\overline{CS}$	I	A low signal to this pin enables reading and writing to the 8294.
7	GND	—	This pin must be tied to ground.
8	$\overline{RD}$	I	An active low read strobe at this pin enables the CPU to read data and status from the internal DEU registers.
9	A_0	I	Address input used by the CPU to select DEU registers during read and write operations.
10	$\overline{WR}$	I	An active low write strobe at this pin enables the CPU to send data and commands to the DEU.
11	SYNC	O	High frequency (Clock ÷ 15) output. Can be used as a strobe for external circuitry.
12	D_0	I/O	Three-state, bi-directional data bus lines used to transfer data between the CPU and the 8294.
13	D_1		
14	D_2		
15	D_3		
16	D_4		
17	D_5		
18	D_6		
19	D_7		
20	GND	—	This pin must be tied to ground.

Pin #	Pin Name	I/O	Pin Description
40	V_{CC}	—	+5 volt power input: +5V ± 10%.
39	NC	—	No connection.
38	$\overline{DACK}$	I	DMA acknowledge. Input signal from the 8257 DMA Controller acknowledging that the requested DMA cycle has been granted.
37	DRQ	O	DMA request. Output signal to the 8257 DMA Controller requesting a DMA cycle.
38	SRQ	O	Service Request. Interrupt to the CPU indicating that the 8294 is awaiting data or commands at the input buffer. SRQ = 1 implies IBF = 0.
35	OAV	O	Output Available. Interrupt to the CPU indicating that the 8294 has data or status available in its output buffer. OAV = 1 implies OBF = 1.
34	NC	—	No connection.
33	P6	O	User output port lines. Output lines available to the user via a CPU command which can assert selected port lines. These lines have nothing to do with the encryption function. At power-on, each line is in a 1 state.
32	P5		
31	P4		
30	P3		
29	P2		
28	P1		
27	P0		
26	V_{DD}	—	+5V power input. (+5V ± 10%) Low power standby pin.
25	NC	—	No connection.
24	CCMP	O	Conversion Complete. Interrupt to the CPU indicating that the encryption/decryption of an 8-byte block is complete.
23	NC	—	No connection.
22	NC	—	No connection.
21	NC	—	No connection.

BASIC FUNCTIONAL DESCRIPTION

OPERATION

The data conversion sequence is as follows:

1. A Set Mode command is given, enabling the desired interrupt outputs.
2. An Enter New Key command is issued, followed by 8 data inputs which are retained by the DEU for encryption/decryption. Each byte must have odd parity.
3. An Encrypt Data or Decrypt Data command sets the DEU in the desired mode.

After this, data conversions are made by writing 8 data bytes and then reading back 8 converted data bytes. Any of the above commands may be issued between data conversions to change the basic operation of the DEU; e.g., a Decrypt Data command could be issued to change the DEU from encrypt mode to decrypt mode without changing either the key or the interrupt outputs enabled.

INTERNAL DEU REGISTERS

Four internal registers are addressable by the master processor: 2 for input, and 2 for output. The following table describes how these registers are accessed.

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	A_0	Register
1	0	0	0	Data input buffer
0	1	0	0	Data output buffer
1	0	0	1	Command input buffer
0	1	0	1	Status output buffer
X	X	1	X	Don't care

The functions of each of these registers are described below.

Data Input Buffer — Data written to this register is interpreted in one of three ways, depending on the preceding command sequence.

1. Part of a key.
2. Data to be encrypted or decrypted.
3. A DMA block count.

Data Output Buffer — Data read from this register is the output of the encryption/decryption operation.

Command Input Buffer — Commands to the DEU are written into this register. (See command summary below.)

Status Output Buffer — DEU status is available in this register at all times. It is used by the processor for poll-driven command and data transfer operations.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	X	X	X	KPE	CF	DEC	IBF	OBF

OBF Output Buffer Full; OBF = 1 indicates that output from the encryption/decryption function is available in the Data Output Buffer. It is reset when the data is read.

IBF Input Buffer Full; A write to the Data Input Buffer or to the Command Input Buffer sets IBF = 1. The DEU resets this flag when it has accepted the input byte. Nothing should be written when IBF = 1.

DEC Decrypt; indicates whether the DEU is in an encrypt or a decrypt mode. DEC = 1 implies the decrypt mode. DEC = 0 implies the encrypt mode.

CF Completion Flag; This flag may be used to indicate any or all of three events in the data transfer protocol.

1. It may be used in lieu of a counter in the processor routine to flag the end of an 8-byte transfer.
2. It must be used to indicate the validity of the KPE flag.
3. It may be used in lieu of the CCMP interrupt to indicate the completion of a DMA operation.

KPE Key Parity Error; After a new key has been entered, the DEU uses this flag in conjunction with the CF flag to indicate correct or incorrect parity.

COMMAND SUMMARY

1 — Enter New Key

OP CODE:

0	1	0	0	0	0	0	0

MSB LSB

This command is followed by 8 data byte inputs which are retained in the key buffer (RAM) to be used in encrypting and decrypting data. These data bytes must have odd parity represented by the LSB.

2 — Encrypt Data

OP CODE:

0	0	1	1	0	0	0	0

MSB LSB

This command puts the 8294 into the encrypt mode.

3 — Decrypt Data

OP CODE:

0	0	1	0	0	0	0	0

MSB LSB

This command puts the 8294 into the decrypt mode.

4 — Set Mode

OP CODE:

0	0	0	0	A	B	C	D

MSB LSB

where:

A is the OAV (Output Available) interrupt enable
B is the SRQ (Service Request) interrupt enable
C is the DMA (Direct Memory Access) transfer enable
D is the CCMP (Conversion Complete) interrupt enable

This command determines which interrupt outputs will be enabled. A "1" in bits A, B, or D will enable the OAV, SRQ, or CCMP interrupts respectively. A "1" in bit C will allow DMA transfers. When bit C is set the OAV and SRQ interrupts should also be enabled (bits A,B = 1). Following the command in which bit C, the DMA bit, is set, the 8294 will expect one data byte to specify the number of 8-byte blocks to be converted using DMA.

5 — Write to Output Port

OP CODE:

1	P_6	P_5	P_4	P_3	P_2	P_1	P_0
MSB							LSB

This command causes the 7 least significant bits of the command byte to be latched as output data on the 8294 output port. The initial output data is 1111111. Use of this port is independent of the encryption/decryption function.

PROCESSOR/DEU INTERFACE PROTOCOL

ENTERING A NEW KEY

The timing sequence for entering a new key is shown in Figure 1. A flowchart showing the CPU software to accommodate this sequence is given in Figure 2.

After the Enter New Key command is issued, 8 data bytes representing the new key are written to the data input buffer (most significant byte first). After the eighth byte is accepted by the DEU, CF goes true (CF = 1). The CF bit goes false again when KPE is valid. The CPU can then check the KPE flag. If KPE = 1, a parity error has been detected and the DEU has not accepted the key. Each byte is checked for odd parity, where the parity bit is the LSB of each byte.

Since the CF bit is used in this protocol to indicate the validity of the KPE flag, it may not be used to flag the end of the 8 byte key entry. CF = 1 only as long as KPE is invalid. Therefore, the CPU might not detect that CF = 1 and the key entry is complete before KPE becomes valid. Thus, a counter should be used, as in Figure 2, to flag the end of the new key entry. Then, CF is used to indicate a valid KPE flag.

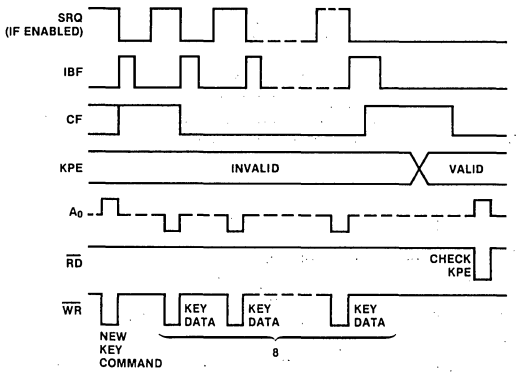


Figure 1. Entering a New Key

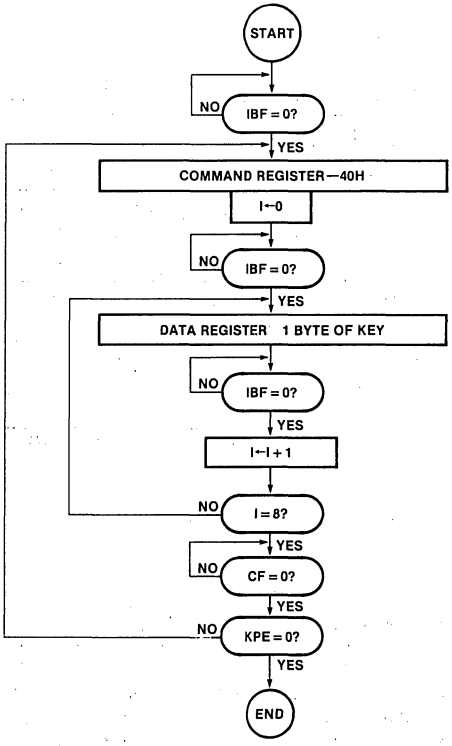


Figure 2. Flowchart for Entering a New Key

ENCRYPTING OR DECRYPTING DATA

Figure 3 shows the timing sequence for encrypting or decrypting data. The CPU writes 8 data bytes to the DEU's data input buffer for encryption/decryption. CF then goes true (CF = 1) to indicate that the DEU has accepted the 8-byte block. Thus, the CPU may test for IBF = 0 and CF = 1 to terminate the input mode, or it may use a software counter. When the encryption/-decryption is complete, the CCMP and OAV interrupts are asserted and the OBF flag is set true (OBF = 1). OAV and OBF are set false again after each of the converted data bytes is read back by the CPU. The CCMP interrupt is set false, and remains false, after the first read. After 8 bytes have been read back by the CPU, CF goes false (CF = 0). Thus, the CPU may test for CF = 0 to terminate the read mode. Also, the CCMP interrupt may be used to initiate a service routine which performs the next series of 8 data reads and 8 data writes.

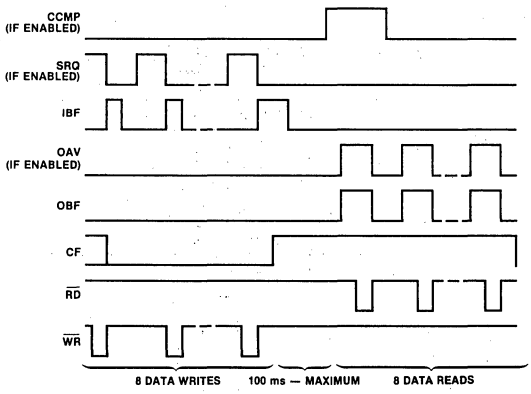


Figure 3. Encrypting/Decrypting Data

Figure 4 offers two flowcharts outlining the alternative means of implementing the data conversion protocol. Either the CF flag or a software counter may be used to end the read and write modes.

SRQ = 1 implies IBF = 0, OAV = 1 implies OBF = 1. This allows interrupt routines to do data transfers without checking status first. However, the OAV service routine must detect and flag the end of a data conversion.

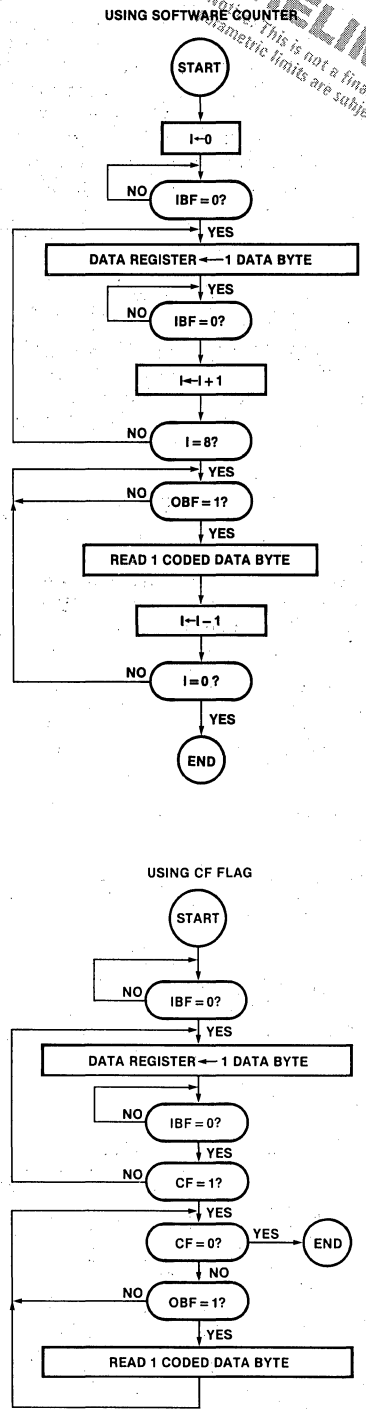


Figure 4. Data Conversion Flowcharts

USING DMA

The timing sequence for data conversions using DMA is shown in Figure 5. This sequence can be better understood when considered in conjunction with the hardware DMA interface in Figure 6. Note that the use of the DMA feature requires 3 external AND gates and 2 DMA channels (one for input, one for output). Since the DEU has only one DMA request pin, the SRQ and OAV outputs are used in conjunction with two of the AND gates to create separate DMA request outputs for the 2 DMA channels. The third AND gate combines the two active-low $\overline{DACK}$ inputs.

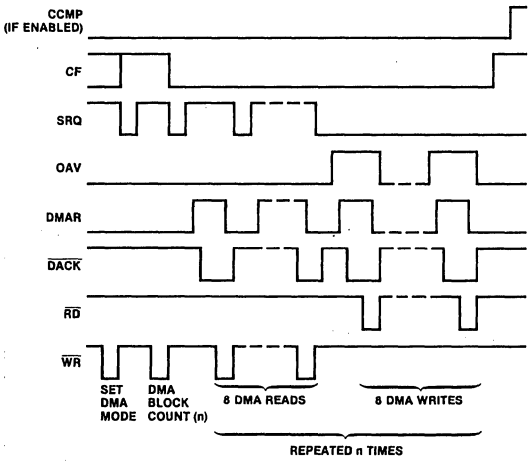
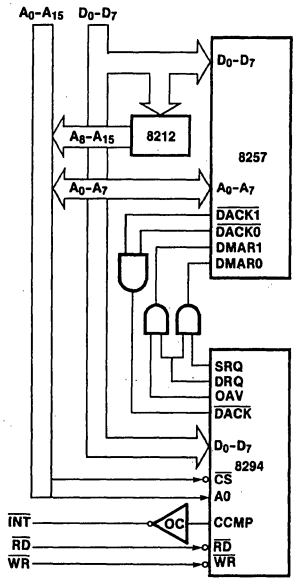


Figure 5. DMA Sequence



DMAR0 IS FOR MEMORY TO DEU DATA TRANSFER
DMAR1 IS FOR DEU TO MEMORY DATA TRANSFER
USE OF CCMP IS OPTIONAL

Figure 6. DMA Interface

To initiate a DMA transfer, the CPU must first initialize the two DMA channels as shown in the flowchart in Figure 7. It must then issue a Set Mode command to the DEU enabling the OAV, SRQ, and DMA outputs. The CCMP interrupt may be enabled or disabled, depending on whether that output is desired. Following the Set Mode command, there must be a data byte giving the number of 8-byte blocks of data (n<256) to be converted. The DEU then generates the required number of DMA requests to the 2 DMA channels with no further CPU intervention. When the requested number of blocks has been converted, the DEU will set CF and assert the CCMP interrupt (if enabled). CCMP then goes false again with the next write to the DEU (command or data). Upon completion of the conversion, the DMA mode is disabled and the DEU returns to the encrypt/decrypt mode. The enabled interrupt outputs, however, will remain enabled until another Set Mode command is issued.

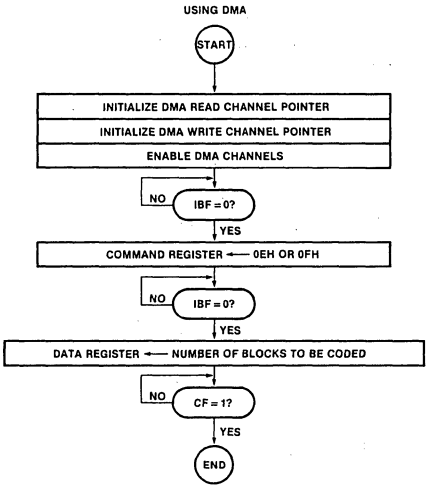


Figure 7. DMA Flowchart

SINGLE BYTE COMMANDS

Figure 8 shows the timing and protocol for single byte commands. Note that any of the commands is effective as a pacify command in that they may be entered at any time, except during a DMA conversion. The DEU is thus set to a known state. However, if a command is issued out of sequence, an additional protocol is required (Figure 9). The CPU must wait until the command is accepted (IBF = 0). A data read must then be issued to clear anything the preceding command sequence may have left in the Data Output Buffer.

CPU/DEU INTERFACES

Figures 10 through 13 illustrate four interface configurations used in the CPU/DEU data transfers. In all cases SRQ will be true (if enabled) and IBF will be false when the DEU is ready to accept data or commands.

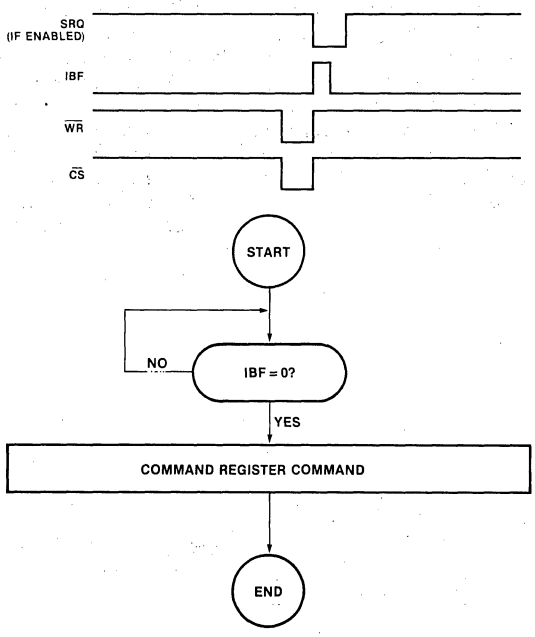


Figure 8. Single Byte Commands

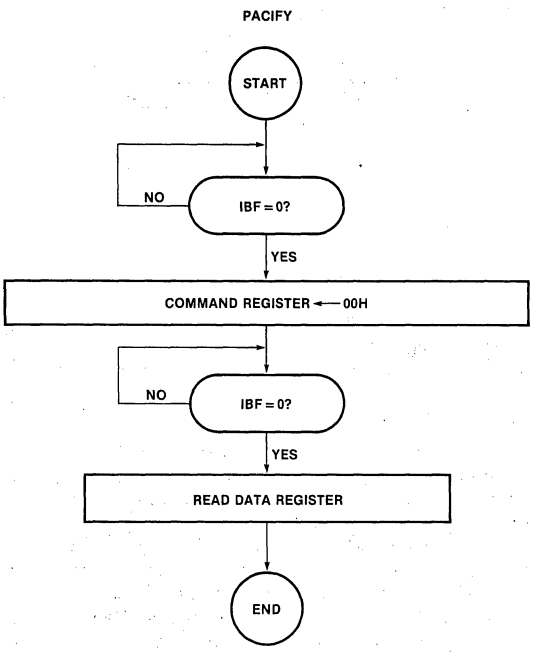


Figure 9. Pacify Protocol

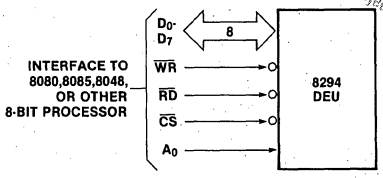


Figure 10. Polling Interface

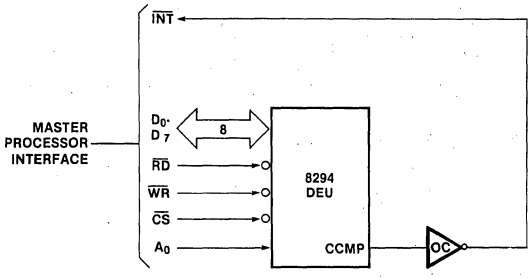


Figure 11. Single Interrupt Interface

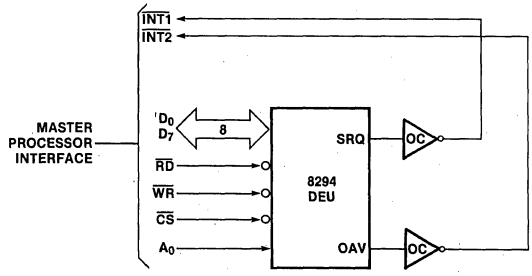
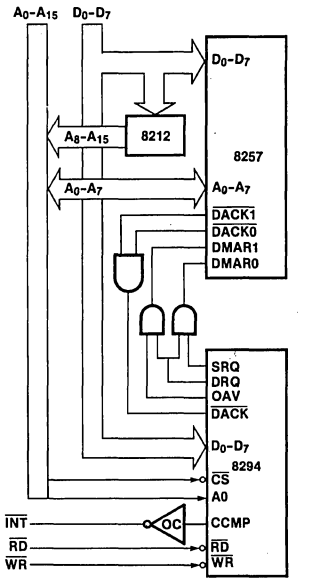


Figure 12. Dual Interrupt Interface



DMAR0 IS FOR MEMORY TO DEU DATA TRANSFER
DMAR1 IS FOR DEU TO MEMORY DATA TRANSFER
USE OF CCMP IS OPTIONAL

Figure 13. DMA Interface

OSCILLATOR AND TIMING CIRCUITS

The 8294's internal timing generation is controlled by a self-contained oscillator and timing circuit. A choice of crystal, L-C or external clock can be used to derive the basic oscillator frequency.

The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 14.

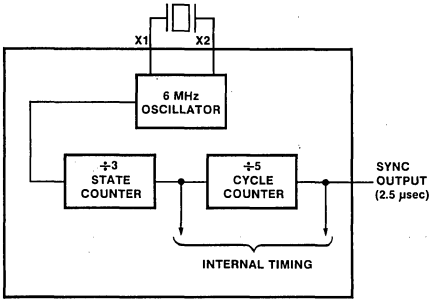
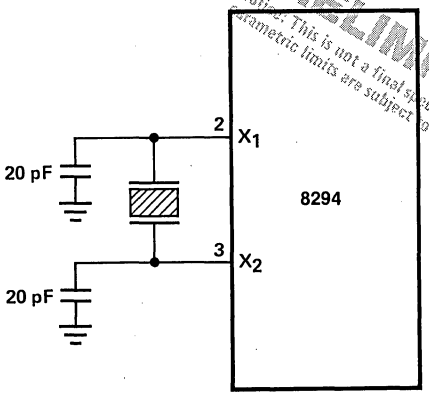


Figure 14. Oscillator Configuration

OSCILLATOR

The on-board oscillator is a series resonant circuit with a frequency range of 1 to 6 MHz. Pins X1 and X2 are input and output (respectively) of a high gain amplifier stage. A crystal or inductor and capacitor connected between X1 and X2 provide the feedback and proper phase shift for oscillation. Recommended connections for crystal or L-C are shown in Figure 15.



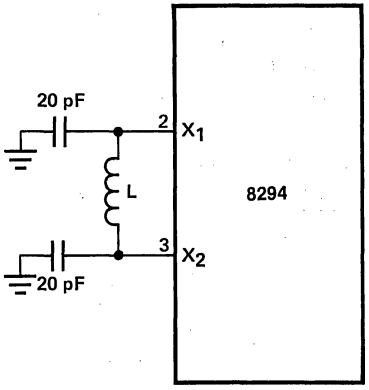


Figure 15. Recommended Crystal and L-C Connections

A recommended range of inductance and capacitance combinations is given below:

L = 130 μH corresponds to 3 MHz
L = 40 μH corresponds to 5 MHz

An external clock signal can also be used as a frequency reference to the 8294; however, the levels are *not* compatible. The signal must be in the 1 MHz-6 MHz frequency range and must be connected to pins X1 and X2 by buffers with a suitable pull-up resistor to guarantee that a logic "1" is above 3.0 volts. Two recommended connections are shown in Figure 16.

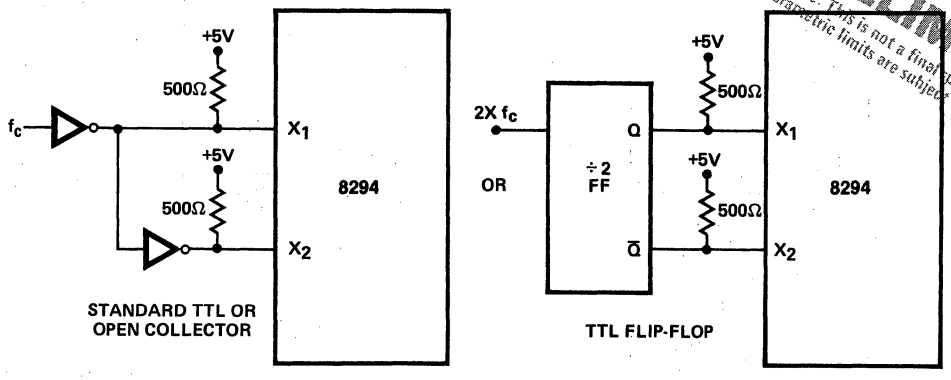


Figure 16. Recommended Connections for External Clock Signal

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... −65°C to +150°C
Voltage on Any Pin With
Respect to Ground ..... 0.5V to +7V
Power Dissipation ..... 1.5 Watt

*COMMENT
Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C TO 70°C, V_{CC} = V_{DD} = +5V ± 10% V_{SS} = 0V

Symbol	Parameter	Limits Min.	Limits Typ.	Limits Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage (All Except X_1, X_2, $\overline{RESET}$)	−0.5		0.8	V	
V_{IH_1}	Input High Voltage (All Except X_1, X_2 $\overline{RESET}$)	2.0		V_{CC}	V	
V_{IH2}	Input High Voltage (X_1,X_2 $\overline{RESET}$)	3.0		V_{CC}	V	
V_{OL_1}	Output Low Voltage (D_0-D_7, Sync)			0.45	V	I_{OL} = 2.0 mA
V_{OL2}	Output Low Voltage All Other Outputs			0.45	V	I_{OL} = 1.6 mA
V_{OH1}	Output High Voltage (D_0-D_7)	2.4			V	I_{OH} = −400 μA
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = −50 μA
I_{IL}	Input Leakage Current $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, $\overline{A_0}$,			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OZ}	Output Leakage Current (D_0-D_7, High Z State)			±10	μA	$V_{SS}+0.45 \leq V_{IN} \leq V_{CC}$
I_{DD}	V_{DD} Supply Current		10	25	mA	
$I_{DD}+I_{CC}$	Total Supply Current		65	135	mA	
I_{LI1}	Low Input Load Current Pins 24, 27-38			0.4	mA	V_{IL}=0.8V
I_{LI2}	Low Input Load Current $\overline{RESET}$			0.2	mA	V_{IL} = 0.8V

A.C. CHARACTERISTICS $T_A = 0°C$ TO $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$,A_0 Setup to $\overline{RD}$ ↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$ ↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		150	ns	
t_{RD}	$\overline{RD}$ ↓ to Data Out Delay		150	ns	
t_{RDF}	$\overline{RD}$ ↑ to Data Float Delay	10		ns	
			100	ns	
t_{RV}	Recovery Time Between Reads and/or Write	1		μs	
t_{CY}	Cycle Time	2.5		μs	6 MHz Crystal

DBB WRITE

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}$ ↓	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}$ ↑	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$ ↑	150		ns	
t_{WD}	Data Hold to $\overline{WR}$ ↑	0		ns	

DMA AND INTERRUPT TIMING

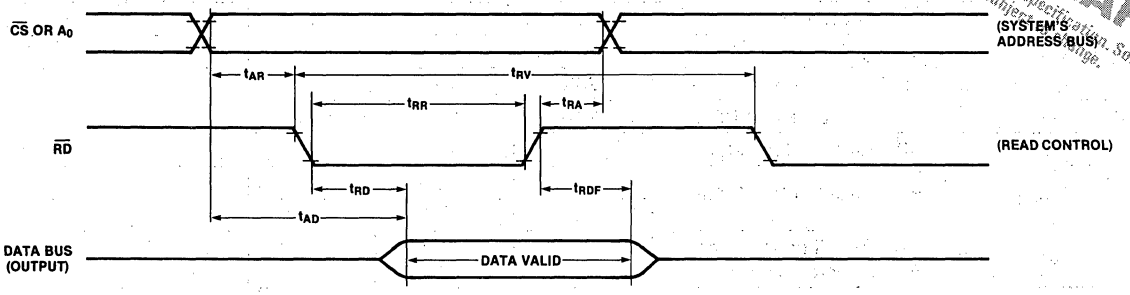
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{KC}	$\overline{DACK}$ Setup to Control	50		ns	
t_{CK}	$\overline{DACK}$ Hold After Control	0		ns	
t_{CR}	Control L.E. to DRQ T.E.		150	ns	
t_{CI}	Control T.E. to Interrupt T.E.		$t_{CY} + 500$	ns	

A.C. TEST CONDITIONS

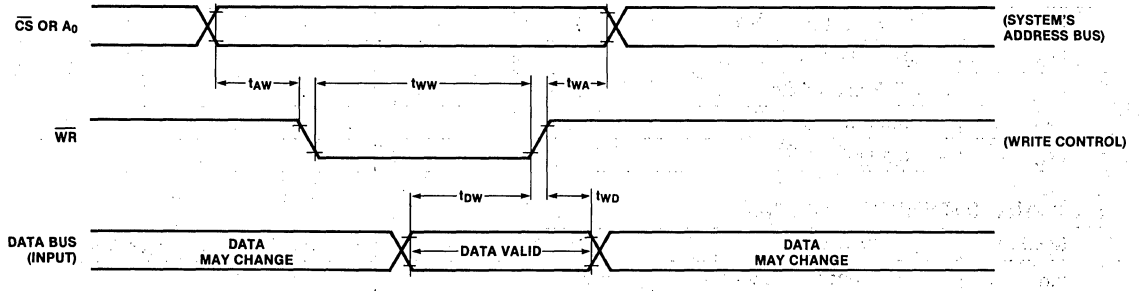
D_7–D_0 Outputs $C_L = 150$ pF

WAVEFORMS

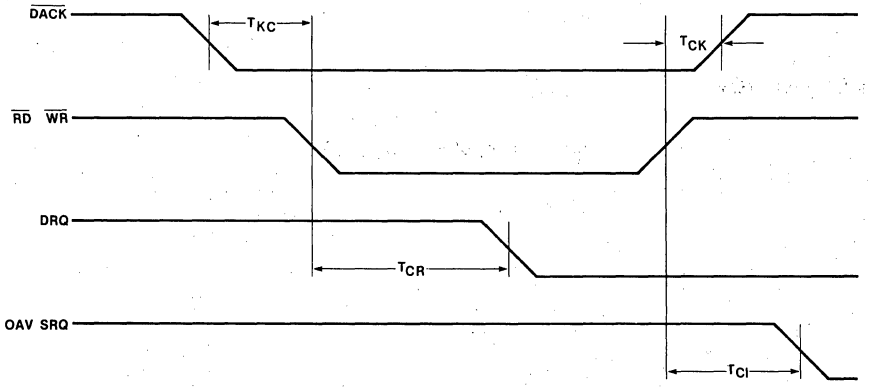
1. READ OPERATION — OUTPUT BUFFER REGISTER.



2. WRITE OPERATION — INPUT BUFFER REGISTER.



DMA AND INTERRUPT TIMING



intel®

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

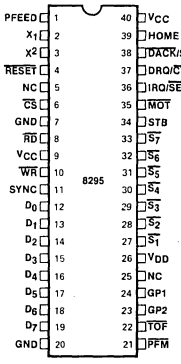
8295
DOT MATRIX PRINTER CONTROLLER

- Interfaces Dot Matrix Printers to MCS-48™, MCS-80/85™, MCS-86™ Systems
- 40 Character Buffer On Chip
- Serial or Parallel Communication with Host
- DMA Transfer Capability
- Programmable Character Density (10 or 12 Chararcters/Inch)
- Programmable Print Intensity
- Single or Double Width Printing
- Programmable Multiple Line Feeds
- 3 Tabulations
- 2 General Purpose Outputs

The Intel® 8295 Dot Matrix Printer Controller provides an interface for microprocessors to the LRC 7040 Series dot matrix impact printers. It may also be used as an interface to other similar printers.

The chip may be used in a serial or parallel communication mode with the host processor. In parallel mode, data transfers are based on polling, interrupts, or DMA. Furthermore, it provides internal buffering of up to 40 characters and contains a 7 x 7 matrix character generator accommodating 64 ASCII characters.

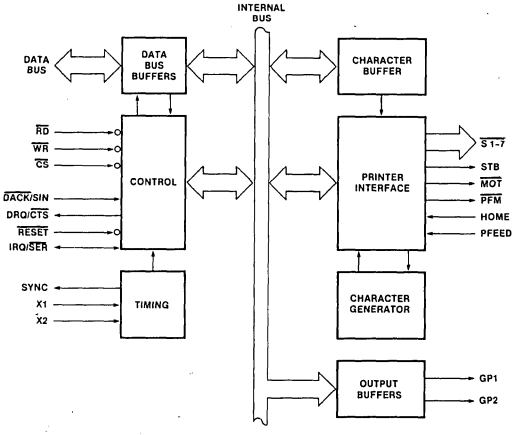
PIN CONFIGURATION



PIN NAMES

PIN NAME	FUNCTION
D_0-D_7	DATA BUS
RD, WR	READ, WRITE STROBES
CS	CHIP SELECT
RESET	RESET INPUT
X_1, X_2	FREQUENCY REFERENCE INPUTS
SYNC	HIGH FREQUENCY OUTPUT
MOT, PFM	MAIN, PAPER FEED MOTOR DRIVES
DRQ, DACK	DMA REQUEST, ACKNOWLEDGE
SIN, CTS	SERIAL INPUT, CLEAR-TO-SEND
IRQ/SER	INTERRUPT REQUEST, SERIAL GROUND
S_1-S_7	SOLENOID DRIVE OUTPUTS
PFEED	PAPER FEED INPUT
HOME, TOF	HOME, TOP-OF-FORM INPUTS
STB	SOLENOID STROBE OUTPUT
GP1, GP2	GENERAL PURPOSE OUTPUTS
VCC, VDD, GND	+5V POWER, GND

BLOCK DIAGRAM



PIN DESCRIPTION

Name	I/O	Pin #	Description
PFEED	I	1	Paper feed input switch.
X1 X2	I	2 3	Inputs for a crystal to set internal oscillator frequency. For proper operation use 6 MHz crystal.
$\overline{RESET}$	I	4	Reset input, active low. After reset the 8295 will be set for 12 characters/inch single width printing, solenoid strobe at 320 msec.
NC	—	5	No connection.
$\overline{CS}$	I	6	Chip select input used to enable the $\overline{RD}$ and $\overline{WR}$ inputs except during DMA.
GND	—	7	This pin must be tied to ground.
$\overline{RD}$	I	8	Read input which enables the master CPU to read data and status. In the serial mode this pin must be tied to V_{CC}.
V_{CC}	—	9	+5 volt power input: +5V ± 10%.
$\overline{WR}$	I	10	Write input which enables the master CPU to write data and commands to the 8295. In the serial mode this pin must be tied to V_{SS}.
SYNC	O	11	2.5 μs clock output. Can be used as a strobe for external circuitry.
D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7	I/O	12 13 14 15 16 17 18 19	Three-state bidirectional data bus buffer lines used to interface the 8295 to the host processor in the parallel mode. In the serial mode $D_0 - D_2$ sets up the baud rate.
GND	—	20	This pin must be tied to ground.
V_{CC}	—	40	+5 volt power input: +5V ± 10%.

Name	I/O	Pin #	Description
HOME	I	39	Home input switch, used by the 8295 to detect that the print head is in the home position.
$\overline{DACK}$/SIN	I	38	In the parallel mode used as DMA acknowledgement; in the serial mode, used as input for data.
DRQ/$\overline{CTS}$	O	37	In the parallel mode used as DMA request output pin to indicate to the 8257 that a DMA transfer is requested; in the serial mode used as clear-to-send signal.
IRQ/$\overline{SER}$	O	36	In parallel mode it is an interrupt request input to the master CPU; in serial mode it should be strapped to V_{SS}.
$\overline{MOT}$	O	35	Main motor drive, active low.
STB	O	34	Solenoid strobe output. Used to determine duration of solenoids activation.
$\overline{S_7}$ $\overline{S_6}$ $\overline{S_5}$ $\overline{S_4}$ $\overline{S_3}$ $\overline{S_2}$ $\overline{S_1}$	O	33 32 31 30 29 28 27	Solenoid drive outputs; active low.
V_{DD}	—	26	+5V power input (+5V ± 10%). Low power standby pin.
NC	—	25	No connection.
GP1 GP2	O O	24 23	General purpose output pins.
$\overline{TOF}$	I	22	Top of form input, used to sense top of form signal for type T printer.
$\overline{PFM}$	O	21	Paper feed motor drive, active low.

FUNCTIONAL DESCRIPTION

The 8295 interfaces microcomputers to the LRC 7040 Series dot matrix impact printers, and to other similar printers. It provides internal buffering of up to 40 characters. Printing begins automatically when the buffer is full or when a carriage return character is received. It provides a modified 7x7 matrix character generator. The character set includes 64 ASCII characters.

Communication between the 8295 and the host processor can be implemented in either a serial or parallel mode. The parallel mode allows for character transfers into the buffer via DMA cycles. The serial mode features selectable data rates from 110 to 4800 baud.

The 8295 also offers two general purpose output pins which can be set or cleared by the host processor. They can be used with various printers to implement such functions as ribbon color selection, enabling form release solenoid, and reverse document feed.

COMMAND SUMMARY

Hex Code	Description
00	Clear GP1. This command brings the GP1 pin to a logic low state. After power on it is automatically set high.
01	Clear GP2. Same as the above but for GP2.
02	Set GP1. Sets GP1 pin to a logic high state, inverse of command 00.
03	Set GP2. Same as above but for GP2. Inverse command 01.
04	Software Reset. This is a pacify command. This command is not effective immediately after commands requiring a parameter, as the Reset command will be interpreted as a parameter.
05	Print 10 characters/in. density.
06	Print 12 characters/in. density.
07	Print double width characters. This command prints characters at twice the normal width, that is, at either 17 or 20 characters per line.
08	Enable DMA mode; must be followed by two bytes specifying the number of data characters to be fetched. Least significant byte accepted first.
09	Tab character.
0A	Line feed.
0B	Multiple Line Feed; must be followed by a byte specifying the number of line feeds.
0C	Top of Form. Enables the line feed output until the Top of Form input is activated.
0D	Carriage Return. Signifies end of a line and enables the printer to start printing.
0E	Set Tab #1, followed by tab position byte.
0F	Set Tab #2, followed by tab position byte. Should be greater than Tab #1.
10	Set Tab #3, followed by tab position byte. Should be greater than Tab #2.
11	Print Head Home on Right. On some printers the print head home position is on the right. This command would enable normal left to right printing with such printers.
12	Set Strobe Width; must be followed by strobe width selection byte. This command adjusts the duration of the strobe activation.

PROGRAMMABLE PRINTING OPTIONS

CHARACTER DENSITY

The character density is programmable at 10 or 12 characters/inch (32 or 40 characters/line). The 8295 is automatically set to 12 characters/inch at power-up. Invoking the Print Double-Width command halves the character density (5 or 6 characters/inch). The 10 char/in or 12 char/in command must be re-issued to cancel the Double-Width mode. Different character density modes may not be mixed within a single line of printing.

PRINT INTENSITY

The intensity of the printed characters is determined by the amount of time during which the solenoid is on. This on-time is programmable via the Set Strobe-Width command. A byte following this command sets the solenoid on-time according to Table 1. Note that only the three least significant bits of this byte are important.

D7—D3	D2	D1	D0	Solenoid On (microsec)
x	0	0	0	200
x	0	0	1	240
x	0	1	0	280
x	0	1	1	320
x	1	0	0	360
x	1	0	1	400
x	1	1	0	440
x	1	1	1	480

Table 1.

TABULATIONS

Up to three tabulation positions may be specified with the 8295. The column position of each tabulation is selected by issuing the Set Tab commands, each fol-

lowed by a byte specifying the column. The tab positions will then remain valid until new Set Tab commands are issued.

Sending a tab character (09H) will automatically fill the character buffer with blanks up to the next tab position. The character sent immediately after the tab character will thus be stored and printed at that position.

CPU TO 8295 INTERFACE

Communication between the CPU and the 8295 may take place in either a serial or parallel mode. However, the selection of modes is inherent in the system hardware; it is not software programmable. Thus, the two modes cannot be mixed in a single 8295 application.

PARALLEL INTERFACE

Two internal registers on the 8295 are addressable by the CPU: one for input, one for output. The following table describes how these registers are accessed.

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Register
1	0	0	Input Data Register
0	1	0	Output Status Register

Input Data Register—Data written to this register is interpreted in one of two ways, depending on how the data is coded.

1. A command to be executed (0XH or 1XH).
2. A character to be stored in the character buffer for printing (2XH, 3XH, 4XH, or 5XH). See the character set, Table 2.

Output Status Register—8295 status is available in this register at all times.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	x	x	PA	DE	x	x	IBF	x

PA—Parameter Required; PA = 1 indicates that a command requiring a parameter has been received. After the necessary parameters have been received by the 8295, the PA flag is cleared.

DE—DMA Enabled; DE = 1 whenever the 8295 is in DMA mode. Upon completion of the required DMA transfers, the DE flag is cleared.

IBF—Input Buffer Full; IBF = 1 whenever data is written to the Input Data Register. No data should be written to the 8295 when IBF = 1.

A flow chart describing communication with the 8295 is shown in Figure 1.

The interrupt request output (IRQ, Pin 36) is available on the 8295 for interrupt driven systems. This output is asserted true whenever the 8295 is ready to receive data.

To improve bus efficiency and CPU overhead, data may be transferred from main memory to the 8295 via DMA cycles. Sending the Enable DMA command (08H) activates the DMA channel of the 8295. This command must be followed by two bytes specifying the length of the data string to be transferred (least significant byte first). The 8295 will then assert the required DMA requests to the 8257 DMA controller without further CPU intervention. Figure 2 shows a block diagram of the 8295 in DMA mode.

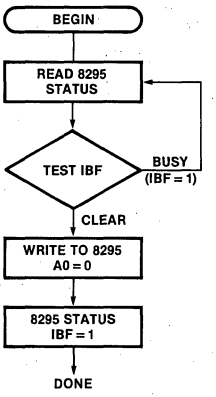


Figure 1. Host to 8295 Protocol Flowchart

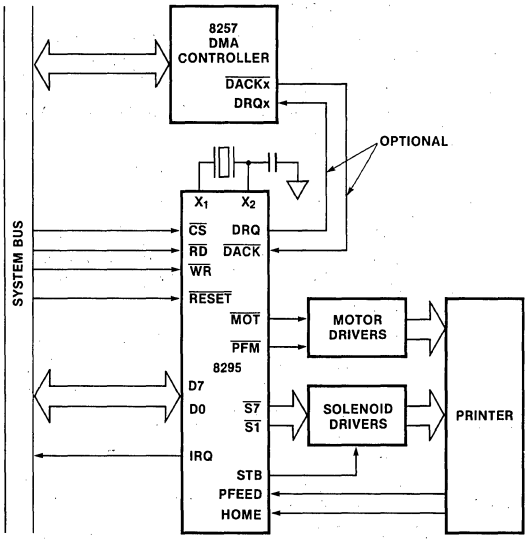


Figure 2. Parallel System Interface

Data transferred in the DMA mode may be either commands or characters or a mixture of both. The procedure is as follows:

1. Set up the 8257 DMA controller channel by sending a starting address and a block length.
2. Set up the 8295 by issuing the "Enable DMA" command (08H) followed by two bytes specifying the block length (least significant byte first).

The DMA enabled flag (DE) will be true until the assigned data transfer is completed. Upon completion of the transfer, the flag is cleared and the interrupt request (IRQ) signal is asserted. The 8295 then returns to the non-DMA mode of operation.

SERIAL INTERFACE

The 8295 may be hardware programmed to operate in a serial mode of communication. By connecting the IRQ/$\overline{SER}$ pin (pin 36) to logic zero, the serial mode is enabled immediately upon power-up. The serial Baud rate is also hardware programmable; by strapping pins 14, 13, and 12 according to Table 2, the rate is selected. $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ must be strapped as shown in Figure 3.

Pin 14	Pin 13	Pin 12	Baud Rate
0	0	0	110
0	0	1	150
0	1	0	300
0	1	1	600
1	0	0	1200
1	0	1	2400
1	1	0	4800
1	1	1	4800

Table 2.

The serial data format is shown in Figure 3. The CPU should wait for a clear to send signal ($\overline{CTS}$) from the 8295 before sending data.

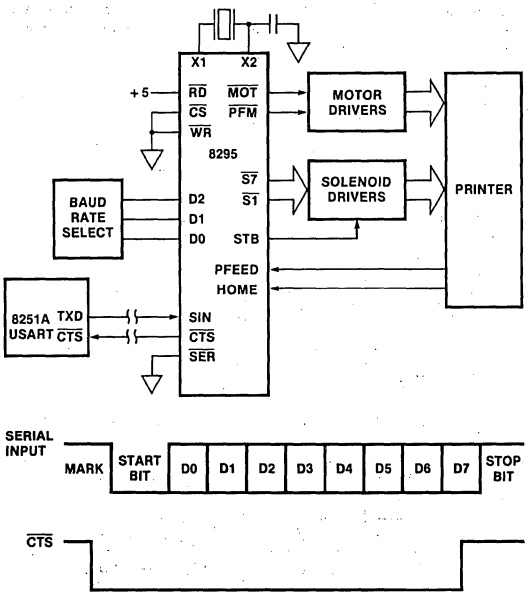


Figure 3. Serial Interface to UART (8251A)

8295 TO PRINTER INTERFACE

The strobe output signal of the 8295 determines the duration of the solenoid outputs, which hold the data to the printer. These solenoid outputs cannot drive the printer solenoids directly. They should be buffered through solenoid drivers as shown in Figure 4. Recommended solenoid and motor driver circuits may be found in the printer manufacturer's interface guide.

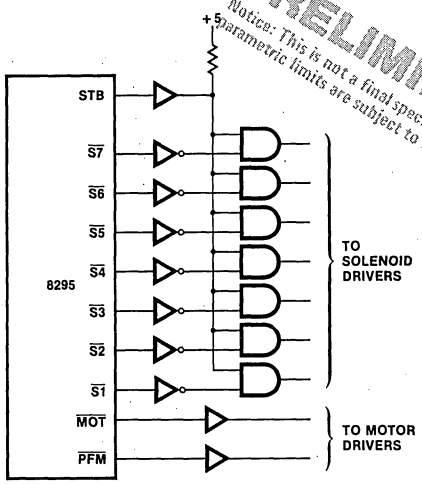


Figure 4. 8295 To Printer Solenoid Interface

OSCILLATOR AND TIMING CIRCUITS

The 8295's internal timing generation is controlled by a self-contained oscillator and timing circuit. A 6 MHz crystal is used to derive the basic oscillator frequency. The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 5. The recommended crystal connection is shown in Figure 6.

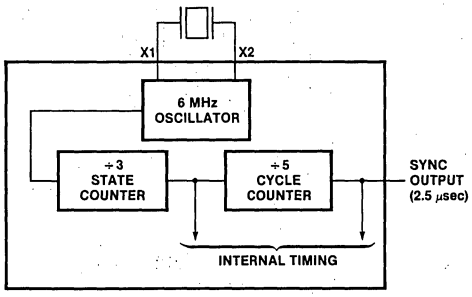


Figure 5. Oscillator Configuration

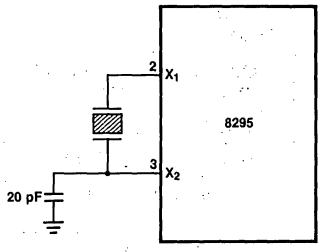


Figure 6. Recommended Crystal Connection

8295 CHARACTER SET

Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.
20	space	30	0	40	@	50	P
21	!	31	1	41	A	51	Q
22	"	32	2	42	B	52	R
23	#	33	3	43	C	53	S
24	$	34	4	44	D	54	T
25	%	35	5	45	E	55	U
26	&	36	6	46	F	56	V
27	,	37	7	47	G	57	W
28	(	38	8	48	H	58	X
29	)	39	9	49	I	59	Y
2A	*	3A	:	5A	J	5A	Z
2B	+	3B	;	4B	K	5B	[
2C	'	3C	<	4C	L	5C	\
2D	-	3D	=	4D	M	5D	]
2E	.	3E	>	4E	N	5E	↑
2F	/	4F	?	4F	O	5F	—

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias.....0°C to 70°C
Storage Temperature.....−65° to +150°C
Voltage on Any Pin With
Respect to Ground.....0.5V to +7V
Power Dissipation.....1.5 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$ $V_{SS} = 0V$

Symbol	Parameter	Limits Min.	Limits Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage (All Except X_1, X_2, $\overline{RESET}$)	−0.5	0.8	V	
V_{IH_1}	Input High Voltage (All Except X_1, X_2 $\overline{RESET}$)	2.2	V_{CC}	V	
V_{IH2}	Input High Voltage (X_1, X_2 $\overline{RESET}$)	3.0	V_{CC}	V	
V_{OL_1}	Output Low Voltage ($D_0 - D_7$, Sync)		0.45	V	$I_{OL} = 2.0$ mA
V_{OL2}	Output Low Voltage All Other Outputs		0.45	V	$I_{OL} = 1.6$ mA
V_{OH1}	Output High Voltage ($D_0 - D_7$)	2.4		V	$I_{OH} = -400\ \mu A$
V_{OH2}	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50\ \mu A$
I_{IL}	Input Leakage Current $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, $\overline{A_0}$		±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OZ}	Output Leakage Current ($D_0 - D_7$, High Z State)		±10	μA	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$
I_{DD}	V_{DD} Supply Current		15	mA	
$I_{DD} + I_{CC}$	Total Supply Current		125	mA	
I_{LI1}	Low Input Load Current Pins 24, 27-38		0.5	mA	$V_{IL} = 0.8V$
I_{LI2}	Low Input Load Current $\overline{RESET}$		0.2	mA	$V_{IL} = 0.8V$

A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

DBB READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$ ↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$ ↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	
t_{RD}	$\overline{RD}$ ↓ to Data Out Delay		225	ns	
t_{RDF}	$\overline{RD}$ ↑ to Data Float Delay	10	100	ns	
T_{RV}	Recovery Time Between Reads and/or Write	300		μs	
t_{CY}	Cycle Time	2.5	15	μs	

DBB WRITE

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}$ ↓	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}$ ↑	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$↑	150		ns	
t_{WD}	Data Hold to $\overline{WR}$ ↑	0		ns	

DMA AND INTERRUPT TIMING

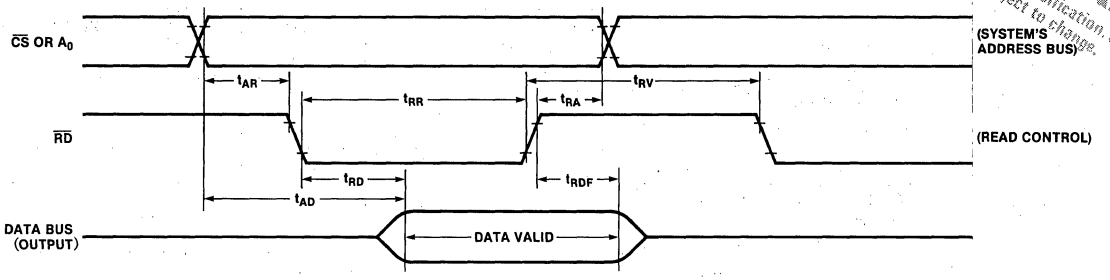
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{KC}	$\overline{DACK}$ Setup to Control	0		ns	
t_{CK}	$\overline{DACK}$ Hold After Control	0		ns	
t_{CRQ}	$\overline{WR}$ to DRQ Cleared		200	ns	

A.C. TEST CONDITIONS

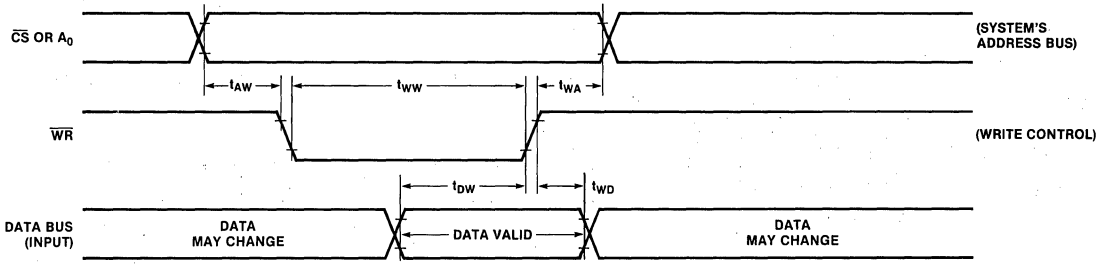
$D_7 - D_0$ Outputs $C_L = 150$ pF

WAVEFORMS

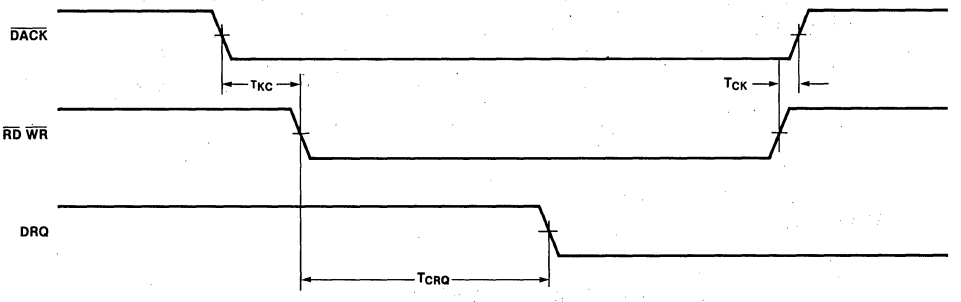
1. READ OPERATION — OUTPUT BUFFER REGISTER.



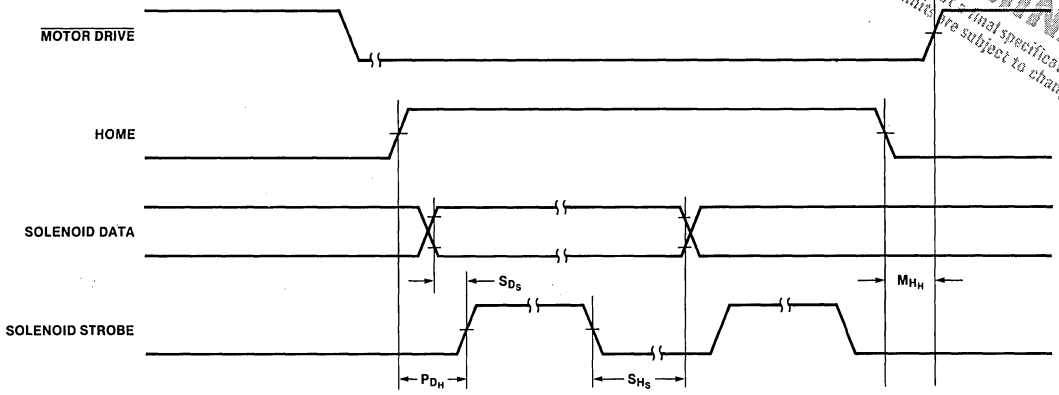
2. WRITE OPERATION — INPUT BUFFER REGISTER.

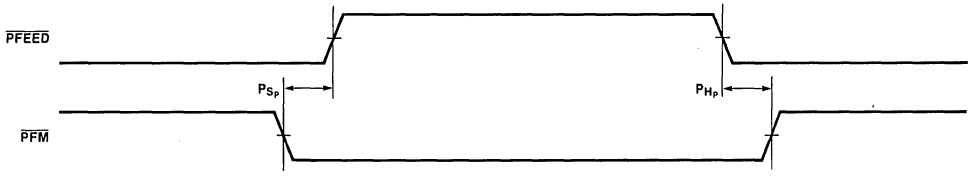


DMA AND INTERRUPT TIMING



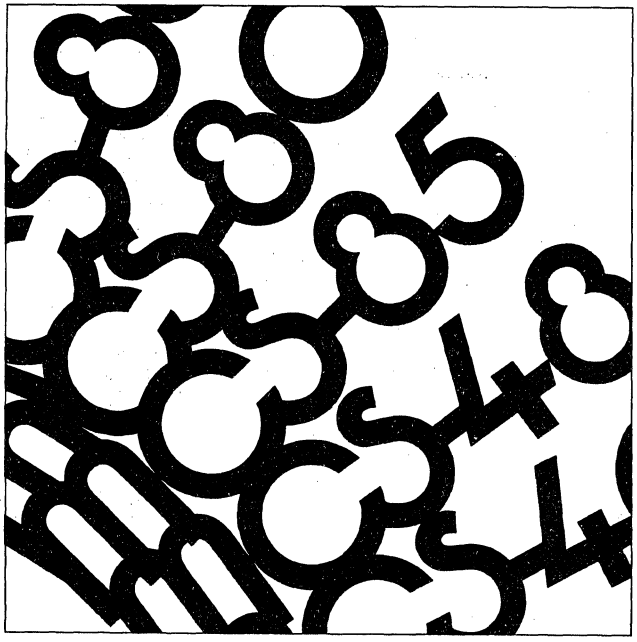
PRINTER INTERFACE TIMING AND WAVEFORMS





Symbol	Parameter	Typical
P_{DH}	Print delay from home inactive	1.8 ms
S_{DS}	Solenoid data setup time before strobe active	25 μs
S_{HS}	Solenoid data hold after strobe inactive	>1 ms
M_{HA}	Motor hold time after home active	3.2 ms
P_{SP}	PFEED setup time after PFM active	58 ms
P_{HP}	PFM hold time after PFEED active	9.75 ms

SECTION 2
PERIPHERAL
APPLICATION NOTES



Introduction to the UPI-41A

by John Beaston and Robin Jigour

INTRODUCTION

Since the introduction in 1974 of the second generation of microprocessors, such as the 8080, a wide range of peripheral interface devices have appeared. At first, these devices solved application problems of a general nature; i.e., parallel interface (8255), serial interface (8251), timing (8253), interrupt control (8259). However, as the speed and density of LSI technology increased, more and more intelligence was incorporated into the peripheral devices. This allowed more specific application problems to be solved, such as floppy disk control (8271), CRT control (8275), and data link control (8273). The advantage to the system designer of this increased peripheral device intelligence is that many of the peripheral control tasks are now handled externally to the main processor in the peripheral hardware rather than internally in the main processor software. This reduced main processor overhead results in increased system throughput and reduced software complexity.

In spite of the number of peripheral devices available, the pervasiveness of the microprocessor has been such that there is still a large number of peripheral control applications not yet satisfied by dedicated LSI. Complicating this problem is the fact that new applications are emerging faster than the manufacturers can react in developing new, dedicated peripheral controllers. To address this problem, a new microcomputer-based Universal Peripheral Interface (UPI-41A) device was developed.

In essence, the UPI-41A acts as a slave processor to the main system CPU. The UPI contains its own processor, memory, and I/O, and is completely user programmable; that is, the entire peripheral control algorithm can be programmed locally in the UPI, instead of taxing the master processor's main memory. This distributed processing concept allows the UPI to handle the real-time tasks such as encoding keyboards, controlling printers, or multiplexing displays, while the main processor is handling non-real-time dependent tasks such as buffer management or arithmetic. The UPI relies on the master only for initialization, elementary commands, and data transfers. This technique results in an overall increase in system efficiency since both processors — the master CPU and the slave UPI — are working in parallel.

This application note presents three UPI-41A applications which are roughly divided into two groups: applications whose complexity and UPI code space requirements allow them to either stand alone or be incorporated as just one task in a "multi-tasking" UPI, and applications which are complete UPI applications in themselves. Applications in the first group are a simple LED display and sensor matrix controllers. A combination serial/parallel I/O device is an application in the second group. Each application illustrates different UPI config-

UPI-41 vs. UPI-41A

The UPI-41A is an enhanced version of the UPI-41. It incorporates several architectural features not found on the "non-A" device:

- Separate Data In and Data Out data bus buffer registers
- User-definable STATUS register bits
- Programmable master interrupts for the OBF and $\overline{IBF}$ flags
- Programmable DMA interface to external DMA controller.

The separate Data In (DBBIN) and Data Out (DBBOUT) registers greatly simplify the master/UPI protocol compared to the UPI-41. The master need only check IBF before writing to DBBIN and OBF before reading DBBOUT. No data bus buffer lock-out is required.

The most significant nibble of the STATUS register, undefined in the UPI-41, is user-definable in UPI-41A. It may be loaded directly from the most significant nibble of the Accumulator (MOV STS, A). These extra four STATUS bits are useful for transferring additional status information to the master. This application note uses this feature extensively.

A new instruction, EN FLAGS, allows OBF and $\overline{IBF}$ to be reflected on Port 2 bit 4 and Port 2 bit 5 respectively. This feature enables interrupt-driven data transfers when these pins are interrupt sources to the master.

By executing an EN DMA instruction Port 2 bit 6 becomes a DRQ (DMA Request) output and Port 2 bit 7 becomes $\overline{DACK}$ (DMA Acknowledge). Setting DRQ requests a DMA cycle to an external DMA controller. When the cycle is granted, the DMA controller returns DACK plus either RD (Read) or WR (Write). DACK automatically forces $\overline{CS}$ and A0 low internally and clears DRQ. This selects the appropriate data buffer register (DBBOUT for DACK and RD, DBBIN for DACK and WR) for the DMA transfer.

Like the "non-A", the UPI-41A is available in both ROM (8041A) and EPROM (8741A) Program Memory versions. This application note deals exclusively with the UPI-41A since the applications use the "A"s enhanced features.

urations and features. However, before the application details are presented, a section on the UPI/master protocol requirements is included. These protocol requirements are key to UPI software development. It is suggested that the reader not already familiar with the architecture and instruction set of the UPI-41A read the "Intel UPI-41 User's Manual" before proceeding with this document. For convenience, the UPI block diagram and instruction set summary are reproduced in Figures 1 and 2.

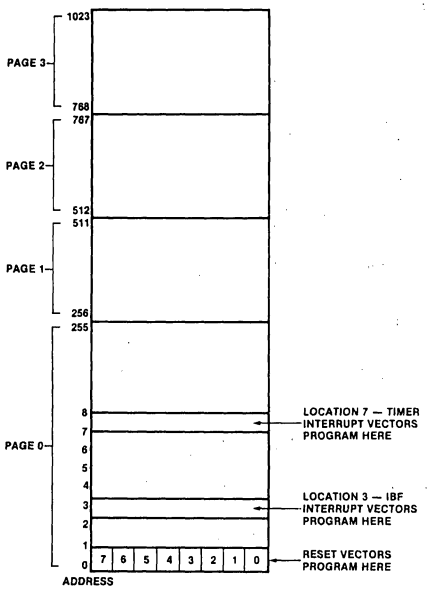


Figure 1A. Program Memory Map

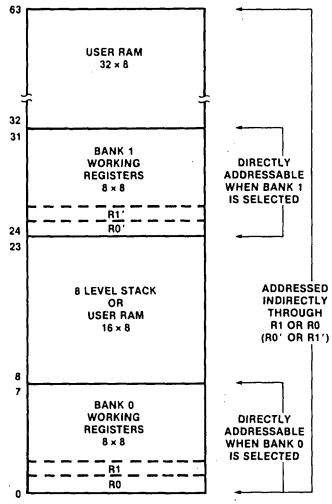


Figure 1B. Data Memory Map

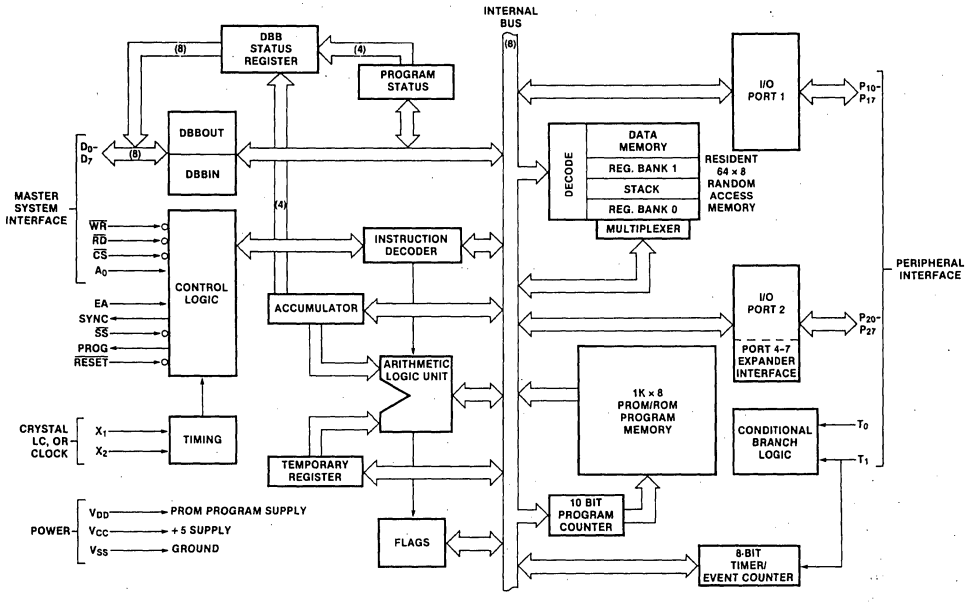


Figure 1C. UPI-41A Block Diagram

UPI INSTRUCTION SET

Mnemonic	Description	Bytes	Cycles
ACCUMULATOR			
ADD A,Rr	Add register to A	1	1
ADD A,@Rr	Add data memory to A	1	1
ADD A,#data	Add immediate to A	2	2
ADDC A,Rr	Add immed to A with carry	1	1
ADDC A,@Rr	Add immed to A with carry	1	1
ADDC A,#data	Add immed. to A with carry	2	2
ANL A,Rr	AND register to A	1	1
ANL A,@Rr	AND data memory to A	1	1
ANL A,#data	AND immediate to A	2	2
ORL A,Rr	OR register to A	1	1
ORL A,@Rr	OR data memory to A	1	1
ORL A,#data	OR immediate to A	2	2
XRL A,Rr	Exclusive OR register to A	1	1
XRL A,@Rr	Exclusive OR data memory to A	1	1
XRL A,#data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap digits of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
INPUT/OUTPUT			
IN A,Pp	Input port to A	1	2
OUTL Pp,A	Output A to port	1	2
ANL Pp,#data	AND immediate to port	2	2
ORL Pp,#data	OR immediate to port	2	2
IN A,DBB	Input DBB to A, clear IBF	1	1
OUT DBB,A	Output A to DBB, set OBF	1	1
MOVD A,Pp	Input Expander port to A	1	2
MOVD Pp,A	Output A to Expander port	1	2
ANLD Pp,A	AND A to Expander port	1	2
ORLD Pp,A	OR A to Expander port	1	2
DATA MOVES			
MOV A,Rr	Move register to A	1	1
MOV A,@Rr	Move data memory to A	1	1
MOV A,#data	Move immediate to A	2	2
MOV Rr,A	Move A to register	1	1
MOV @Rr,A	Move A to data memory	1	1
MOV Rr,#data	Move immediate to register	2	2
MOV @Rr,#data	Move immediate to data memory	2	2
MOV A,PSW	Move PSW to A	1	1
MOV PSW,A	Move A to PSW	1	1
XCH A,Rr	Exchange A and register	1	1
XCH A,@Rr	Exchange A and data memory	1	1
XCHD A,@Rr	Exchange digit of A and register	1	1
MOVP A,@A	Move to A from current page	1	2
MOVP3, A,@A	Move to A from page 3	1	2
TIMER/COUNTER			
MOV A,T	Read Timer/Counter	1	1
MOV T,A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1

Mnemonic	Description	Bytes	Cycles
CONTROL			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
REGISTERS			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1
SUBROUTINE			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
FLAGS			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
MOV STS, A	A_4-A_7 to Bits 4-7 of Status	1	1
BRANCH			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R,addr	Decrement register and skip	2	2
JC addr	Jump on Carry = 1	2	2
JNC addr	Jump on Carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 Flag = 1	2	2
JF1 addr	Jump on F1 Flag = 1	2	2
JTF addr	Jump on Timer Flag = 1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag = 0	2	2
JOBF addr	Jump on OBF Flag = 1	2	2
JBb addr	Jump on Accumulator Bit	2	2

Figure 2. UPI-41A Instruction Set Summary

UPI/MASTER PROTOCOL

As in most closely coupled multiprocessor systems, the various processors communicate via a shared resource. This shared resource is typically specific locations in RAM or in registers through which status and data are passed. In the case of a master processor and a UPI-41A, the shared resource is 3 separate, master-addressable, registers internal to the UPI. These registers are the STATUS register (STATUS), the Data Bus Buffer Input register (DBBIN), and the Data Bus Output register (DBBOUT). [Data Bus Buffer direction is relative to the UPI]. To illustrate this register interface, consider the 8085A/UPI system in Figure 3.

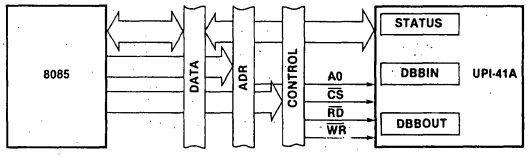


Figure 3. Register Interface

Looking into the UPI from the 8085A, the 8085A sees only the three registers mentioned above. If the 8085A wishes to issue a command to the UPI, it does so by writing the command to the DBBIN register according to the decoding of Figure 4. Data for the UPI is also passed via the DBBIN register. (The UPI differentiates commands and data by examining the A0 pin. Just how this is done is covered shortly.) Data from the UPI for the 8085A is passed in the DBBOUT register. The 8085A may interrogate the UPI's status by reading the UPI's STATUS register. Four bits of the STATUS register act as flags and are used to handshake data and commands into and out of the UPI. The STATUS register format is shown in Figure 5.

Bit 0 is OBF (Output Buffer Full). This flag indicates to the master when the UPI has placed data in the DBBOUT register. OBF is set when the UPI writes to DBBOUT and is reset when the master reads DBBOUT. The master finds meaningful data in the DBBOUT register only when OBF is set.

The Input Buffer Full (IBF) flag is bit 1. The UPI uses this flag as an indicator that the master has written to the DBBIN register. The master uses IBF to indicate when the UPI has accepted a particular command or data byte. The master should examine IBF before outputting anything to the UPI. IBF is set when the master writes to DBBIN and is reset when the UPI reads DBBIN. The master must wait until IBF = 0 before writing new data or commands to DBBIN. Conversely, the UPI must ensure IBF = 1 before reading DBBIN.

The third STATUS register bit is F0 (Flag 0). This is general purpose flag that the UPI can set, reset, and test. It is typically used to indicate a UPI error or busy condition to the master.

Flag 1 (F1) is the final dedicated STATUS bit. Like F0 the UPI can set, reset, and test this flag. However, in addition, F1 reflects the state of the A0 pin whenever the master writes to the DBBIN register. The UPI uses this flag to delineate between master command and data writes to DBBIN.

$\overline{CS}$	A0	$\overline{RD}$	$\overline{WR}$	REGISTER
0	0	0	1	READ DBBOUT
0	1	0	1	READ STATUS
0	0	1	0	WRITE DBBIN (DATA)
0	1	1	0	WRITE DBBIN (COMMAND)
1	X	X	X	NO ACTION

Figure 4. Register Decoding

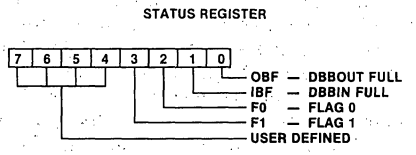


Figure 5. Status Register Format

The remaining four STATUS register bits are user definable. Typical uses of these bits are as status indicators for individual tasks in a multitasking UPI or as UPI generated interrupt status. These bits find a wide variety of uses in the upcoming applications.

Looking into the 8085A from the UPI, the UPI sees the two DBB registers plus the IBF, OBF, and F1 flags. The UPI can write from its accumulator to DBBOUT or read DBBIN into the accumulator. The UPI cannot read OBF, IBF, or F1 directly, but these flags may be tested using conditional jump instructions. The UPI should make sure that OBF is reset before writing new data into DBBOUT to ensure that the master has read previous DBBOUT data. IBF should also be tested before reading DBBIN since DBBIN data is valid only when IBF is set. As was mentioned earlier, the UPI uses F1 to differentiate between command and data contents in DBBIN when IBF is set. The UPI may also write the upper 4-bits of its accumulator to the upper 4-bits of the STATUS register. These bits are thus user definable.

The UPI can test the flags at any time during its internal program execution. It essentially "polls" the STATUS register for changes. If faster response is needed to master commands and data, the UPI's internal interrupt structure can be used. If IBF interrupts are enabled, a master write to DBBIN (either command or data) sets IBF which generates an internal CALL to location 03H in program memory. At this point, working register contents can be saved using bank switching, the accumulator saved in a spare working register, and the DBBIN register read and serviced. The interrupt logic for the IBF interrupt is shown in Figure 6. A few observations concerning this logic are appropriate. Note that if the master writes to DBBIN while the UPI is still servicing the last IBF interrupt (a Return (RETR) instruction has not been executed), the IBF Interrupt Pending line is made high which causes a new CALL to 03H as soon as the first RETR is executed. No EN I (Enable Interrupt) instruction is needed to rearm the interrupt logic as is needed in an 8080 or 8085A system; the RETR performs this function. Also note that executing a DIS I to disable further IBF interrupts does not clear a pending interrupt. Only a CALL to location 03H or RESET clears a pending IBF interrupt.

Keeping in mind that the actual master/UPI protocol is dependent on the application, probably the best way to illustrate correct protocol is by example. Let's consider using the UPI as a simple parallel I/O device. (This is a trivial application but it embodies all of the important protocol considerations.) Since the UPI may be either interrupt or non-interrupt driven internally, both cases are considered.

Let's take the easiest configuration first; using the UPI Port 1 as an 8-bit output port. From the UPI's point-of-view, this is an input-only application since all that is required is that the UPI input data from the master. Once the master writes data to the UPI, the UPI reads the DBBIN register and transfers the data to Port 1. No testing for commands vs data is needed since the UPI "knows" it only performs one task — no commands are needed.

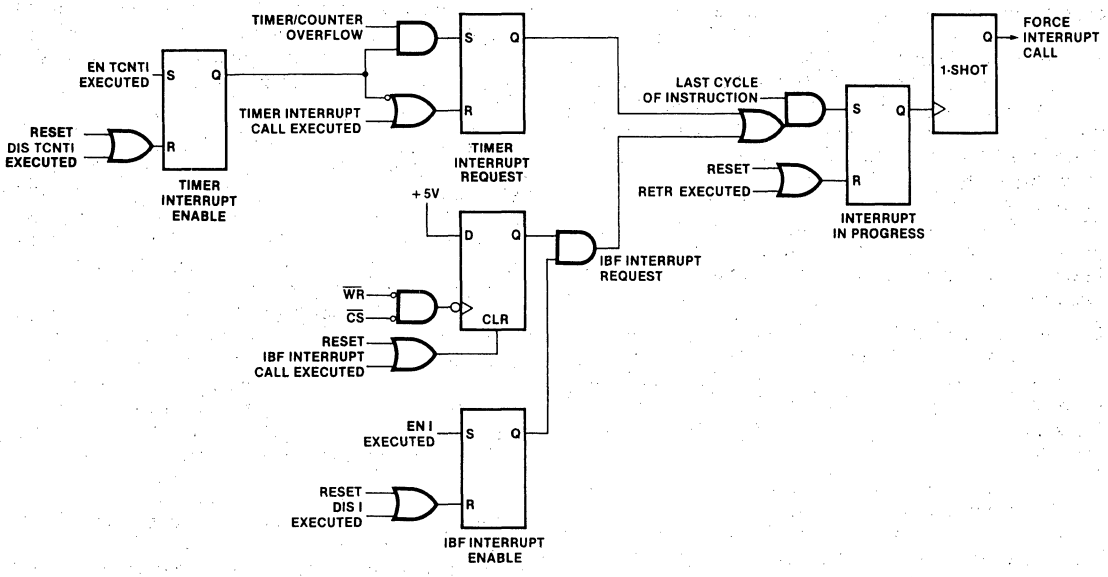


Figure 6. UPI-41A Interrupt Structure

Non-interrupt driven UPI software is shown in Figure 7A while Figure 7B shows interrupt based software. For Figure 7A, the UPI simply waits until it sees IBF go high indicating the master has written a data byte to DBBIN. The UPI then reads DBBIN, transfers it to Port 1, and returns to waiting for the next data. For the interrupt-driven UPI, Figure 7B, once the EN I instruction is executed, the UPI simply waits for the IBF interrupt before handling the data. The UPI could handle other tasks during this waiting time. When the master writes the data to DBBIN, an IBF interrupt is generated which performs a CALL to location 03H. At this point the UPI reads DBBIN (no testing of IBF is needed since an IBF interrupt implies that IBF is set), transfers the data to Port 1, and executes an RETR which returns program flow to the main program.

Software for the master 8085A is included in Figure 7C. The only requirement for the master to output data to the UPI is that it check the UPI to be sure the previous data had been taken before writing new data. To accomplish this the master simply reads the STATUS register looking for IBF = 0 before writing the next data.

```
;
; UPI INPUT ONLY EXAMPLE - PORT 1 USED AS OUTPUT PORT
;        UPI POLLS IBF FOR DATA
;
RESET:   JNIBF   RESET    ; WAIT ON IBF FOR INPUT
         IN      A, DBB   ; INPUT THERE, SO READ IT
         OUTL    P1, A    ; TRANSFER DATA TO PORT 1
         JMP     RESET    ; GO WAIT FOR NEXT DATA
```

Figure 7A. Single Output Port Example — Polling

```
;
; UPI INPUT ONLY EXAMPLE - PORT 1 USED AS OUTPUT PORT
;        DATA INPUT IS INTERRUPT-DRIVEN ON IBF
;
RESET:   EN      I         ; ENABLE IBF INTERRUPTS
         JMP     RESET + 1 ; LOOP WAITING FOR INPUT
IBFINT:  IN      A, DBB    ; READ DATA FROM DBBIN
         OUTL    P1, A     ; TRANSFER DATA TO PORT 1
         RETR              ; RETURN WITH RESTORE
```

Figue 7B. Single Output Port Example — Interrupt

```
;
; 8085 SOFTWARE FOR UPI INPUT-ONLY EXAMPLE
;        DATA FOR OUTPUT IS PASSED IN REG. C
;
UPIOUT:  IN      STATUS    ; READ UPI STATUS
         ANI     IBF       ; LOOK AT IBF
         JNZ     UPIOUT    ; WAIT FOR IBF = 0
         MOV     A, C      ; GET DATA FROM C
         OUT     DBBIN     ; OUTPUT DATA TO DBBIN
         RET               ; DONE, RETURN
```

Figure 7C. 8085A Code for Single Output Port Example

Figure 8A illustrates the case where UPI Port 2 is used as an 8-bit input port. This configuration is termed UPI output-only as the master does not write (input) to the UPI but simply reads either the STATUS or the DBBOUT registers. In this example only the OBF flag is used. OBF signals the master that the UPI has placed new port data in DBBOUT. The UPI loops testing OBF. When OBF is clear, the master has read the previous data and UPI then reads its input port (Port 2) and places this data in DBBOUT. It then waits on OBF until the master reads

DBBOUT before reading the input port again. When the master wishes to read the input port data, Figure 8B, it simply checks for OBF being set in the STATUS register before reading DBBOUT. While this technique illustrates proper protocol, it should be noted that it is not meant to be a good method of using the UPI as an input port since the master would never get the newest status of the port.

The above examples can easily be combined. Figure 9 shows UPI software to use Port 1 as an output port simultaneously with Port 2 as an input port. The program starts with the UPI checking IBF to see if the master has written data destined for the output port into DBBIN. If IBF is set, the UPI reads DBBIN and transfers the data to the output port (Port 1). If IBF is not set or once the data is transferred to the output port if it was, OBF is tested. If OBF is reset (indicating the master has read DBBOUT), the input port (Port 2) is read and transferred to DBBOUT. If OBF is set, the master has yet to read DBBOUT so the program just loops back to test IBF.

```
;
; UPI OUTPUT ONLY EXAMPLE - PORT 2 USED AS INPUT PORT
;        PORT DATA IS AVAILABLE IN DBBOUT
;
RESET:   JOBF    RESET    ; LOOP IF OBF = 1 (DATA NOT READ)
         IN      A, P2    ; DBBOUT CLEAR, READ PORT
         OUT     DBB, A   ; TRANSFER PORT DATA TO DBBOUT
         JMP     RESET    ; WAIT FOR MASTER TO READ DATA
```

Figure 8A. Single Input Port Example

```
;
; 8085 SOFTWARE FOR UPI OUTPUT-ONLY EXAMPLE
;        INPUT DATA RETURNED IN REG. A
;
UPIIN:   IN      STATUS   ; READ UPI STATUS
         ANI     OBF      ; LOOK AT OBF
         JZ      UPIIN    ; WAIT UNTIL OBF = 1
         IN      DBBOUT   ; READ DBBOUT
         RET              ; RETURN WITH DATA IN A
```

Figure 8B. 8085A Single Input Port Code

```
;
;UPI INPUT/OUTPUT EXAMPLE - PORT 1 OUTPUT, PORT 2 INPUT
;
RESET:   JNIBF   OUT1     ; IF IBF = 0, DO OUTPUT
         IN      A, DBB   ; IF IBF = 1, READ DBBIN
         OUTL    P1, A    ; TRANSFER DATA TO PORT 1
OUT1:    JOBF    RESET    ; IF OBF = 1, GO TEST IBF
         IN      A, P2    ; IF OBF = 0, READ PORT 2
         OUT     DBB, A   ; TRANSFER PORT DATA TO DBBOUT
         JMP     RESET    ; GO CHECK FOR INPUT
```

Figure 9. Combination Output/Input Port Example

The master software is identical to the separate input/output examples; the master must test IBF and OBF before writing output port data into DBBIN or before reading input port data from DBBOUT respectively.

In all of the three examples above, the UPI treats information from the master solely as data. There has been no need to check if DBBIN information is a command rather than data since the applications do not require commands. But what if both Port 1 and 2 were used as output ports? The UPI needs to know into which port to put the data. Let's use a command to select which port.

Recall that both commands and data pass through DBBIN. The state of the A0 pin at the time of the write to DBBIN is used to distinguish commands from data. By convention, DBBIN writes with A0 = 0 are for data, and those with A0 = 1 are commands. When DBBIN is written into, F1 (Flag 1) is set to the state of A0. The UPI tests F1 to determine if the information in the DBBIN register is data or a command.

For the case of two output ports, let's assume that the master selects the desired port with a command prior to writing the data. (We could just use F1 as a port select but that would not illustrate the subtle differences between commands and data.) Let's define the port select commands such that bit 1 = 1 if the next data is for Port 1 (Write Port 1 = 0000 0010) and bit 2 = 1 if the next data is for Port 2 (Write Port 2 = 0000 0100). (The number of the set bit selects the port.) Any other bits are ignored. This assignment is completely arbitrary; we could use any command structure, but this one has the advantage of being simple.

Note that the UPI must "remember" from DBBIN write to write which port has been selected. Let's use F0 (Flag 0) for this purpose. If a Write Port 1 command is received, F0 is reset. If the command is Write Port 2, F0 is set. When the UPI finds data in DBBIN, F0 is interrogated and the data is loaded into the previously selected port. The UPI software is shown in Figure 10A.

```
;
; UPI DUAL OUTPUT PORT EXAMPLE - BOTH PORT 1 AND 2 OUTPUTS
;        COMMAND SELECTS DESIRED PORT
;        WRITE PORT 1 - 0000 0010 (02H)
;        WRITE PORT 2 - 0000 0100 (04H)
;
;        FLAG 0 USED TO REMEMBER WHICH PORT WAS SELECTED
;        BY LAST COMMAND.
;
RESET:   JNIBF   RESET    ; WAIT FOR MASTER INPUT
         IN      A, DBB   ; READ INPUT
         JF1     CMD      ; IF F1 = 1, COMMAND INPUT
         JF0     PORT2    ; INPUT IS DATA, TEST F0
         OUTL    P1, A    ; F0 = 0, SO OUTPUT TO PORT 1
         JMP     RESET    ; WAIT FOR NEXT INPUT
PORT2:   OUTL    P2, A    ; F0 = 1, SO OUTPUT TO PORT 2
         JMP     RESET    ; WAIT FOR NEXT INPUT
CMD:     JB1     PT1      ; TEST COMMAND BITS (BIT 1)
         JB2     PT2      ; TEST BIT 2
         JMP     RESET    ; NEITHER BIT SET, WAIT FOR INPUT
PT1:     CLR     F0       ; PORT 1 SELECTED, CLEAR F0
         JMP     RESET    ; WAIT FOR INPUT
PT2:     CLR     F0       ; PORT 2 SELECTED, SET F0
         CPL     F0
         JMP     RESET    ; WAIT FOR INPUT
```

Figure 10A. Dual Output Port Example

Initially, the UPI simply waits until IBF is set indicating the master has written into DBBIN. Once IBF is set, DBBIN is read and F1 is tested for a command. If F1 = 1, the DBBIN byte is a command. Assuming a command, bit 1 is tested to see if the command selected port 1. If so, F0 is cleared and the program returns to wait for the data. If bit 1 = 0, bit 2 is tested. If bit 2 is set, Port 2 is selected so F0 is set. The program then loops back waiting for the next master input. This input is the desired port data. If bit 2 was not set, F0 is not changed and no action is taken.

When IBF = 1 is again detected, the input is again tested for command or data. Since it is necesarily data, DBBIN is read and F0 is tested to determine which port was previously selected. The data is then output to that port, following which the program waits for the next input. Note that since F0 still selects the previous port, the next input could be more data for that port. The port selection command could be thought of as a port select flip-flop control; once a selection is made, data may be repeatedly written to that port until the other port is selected. Master software, Figure 10B, simply must check IBF before writing either a command or data to DBBIN. Otherwise, the master software is straightforward.

For the sake of completeness, UPI software for implementing two input ports is given in Figure 11. This case is simpler than the dual output case since the UPI can assume that all writes to DBBIN are port selection commands so no command/data testing is required. Once the Port Read command is input, the selected port is read and the port data is placed in DBBOUT. Note that in this case F0 is used as a UPI error indicator. If the master happened to issue an invalid command (a command without either bit 1 or 2 set), F0 is set to notify the master that the UPI did not know how to interpret the command. F0 is also set if the master commanded a port read before it had read DBBOUT from the previous command. The UPI simply tests OBF just prior to loading DBBOUT and if OBF = 1, F0 is set to indicate the error.

All of the above examples are, in themselves, rather trivial applications of the UPI although they could easily be incorporated as one of several tasks in a UPI handling multiple small tasks. We have covered them primarily to introduce the UPI concept and to illustrate some master/UPI protocol. Before moving on to more realistic UPI applications, let's discuss two UPI features that do not directly relate to the master/UPI protocol but greatly enhance the UPI's data transfer capability.

In addition to the OBF and IBF bits in the STATUS register, these flags can also be made available directly on two port pins. These port pins can then be used as interrupt sources to the master. By executing an EN FLAGS instruction, Port 2 pin 4 reflects the condition of OBF and Port 2 pin 5 reflects the inverted condition of IBF ($\overline{IBF}$). These dedicated outputs can then be enabled or disabled via their respective port bit values; i.e., P24 reflects OBF as long as an instruction is executed which sets P24 (i.e. ORL P2,#10H). The same action applies to the $\overline{IBF}$ output except P25 is used. Thus P24 may serve as a DATA AVAILABLE interrupt output. Likewise for P25 as a READY-TO-ACCEPT-DATA interrupt. This greatly simplifies interrupt-driven master-slave data transfers.

```
;
; 8085 SOFTWARE FOR DUAL OUTPUT PORT EXAMPLE
;        THIS ROUTINE WRITES DATA IN REG. C TO PORT 1
;        (SAME ROUTINE FOR PORT 2 - JUST CHANGE COMMAND)
;
PORT1:   IN     STATUS     ; READ UPI STATUS
         ANI    IBF        ; LOOK AT IBF
         JNZ    PORT1      ; WAIT UNTIL IBF = 0
         MVI    A, 00000010B    ; LOAD WRITE PORT1 CMD
         OUT    UPICMD     ; OUTPUT TO UPI COMMAND PORT
P1:      IN     STATUS     ; READ UPI STATUS AGAIN
         ANI    IBF        ; LOOK AT IBF
         JNZ    P1         ; WAIT UNTIL COMMAND ACCEPTED
         MOV    A, C       ; GET DATA FROM C
         OUT    DBBIN      ; OUTPUT TO DBBIN
         RET               ; DONE, RETURN
```

Figure 10B. 8085A Dual Output Port Example Code

```
;
; UPI DUAL INPUT PORT EXAMPLE - BOTH PORT 1 AND 2 INPUTS
;        COMMAND SELECTS WHICH PORT IS TO BE READ
;        FLAG 0 USED AS ERROR FLAG
;
RESET:   JNIBF  RESET      ; WAIT FOR INPUT
         CLR    F0         ; CLEAR ERROR FLAG
         IN     A, DBB     ; READ INPUT (COMMAND)
         JB1    PT1        ; TEST BIT 1 (PORT1)
         JB2    PT2        ; TEST BIT 2 (PORT2)
ERROR:   CPL    F0         ; ERROR - COMPLEMENT F0
         JMP    RESET      ; WAIT FOR INPUT
PT1:     IN     A, P1      ; READ PORT 1
         JOBF   ERROR      ; TEST OBF BEFORE LOADING DBBOUT
         OUT    DBB, A     ; LOAD PORT1 DATA INTO DBBOUT
         JMP    RESET      ; WAIT FOR INPUT
PT2:     IN     A, P2      ; READ PORT 2
         JOBF   ERROR      ; TEST OBF BEFORE LOADING DBBOUT
         OUT    DBB, A     ; LOAD PORT2 DATA INTO DBBOUT
         JMP    RESET      ; WAIT FOR INPUT
```

Figure 11. Dual Input Port Example

The UPI also supports a DMA transfer interface. If an EN DMA instruction is executed, Port 2 pin 6 becomes a DMA Request (DRQ) output and P27 becomes a high impedance DMA Acknowledge ($\overline{DACK}$) input. Any instruction which would normally set P26 now sets DRQ. DRQ is cleared when $\overline{DACK}$ is low and either $\overline{RD}$ or $\overline{WR}$ is low. When $\overline{DACK}$ is low, $\overline{CS}$ and A0 are forced low internally which allows data bus transfers between DBBOUT or DBBIN to occur, depending upon whether $\overline{WR}$ or $\overline{RD}$ is true. Of course, the function requires the use of an external DMA controller.

Now that we have discussed the aspects of the UPI protocol and data transfer interfaces, let's move on to the actual applications.

EXAMPLE APPLICATIONS

Each of the following three sections present the hardware and software details of a UPI application. Each application utilizes one of the protocols mentioned in the last section. The first example is a simple 8-digit LED display controller. This application requires only that the UPI perform input operations from the DBBIN; DBBOUT is not used. The reverse is true for the second

application: a sensor matrix controller. The final application involves both DBBOUT and DBBIN operations: a combination serial/parallel I/O device.

The core master processor system with which these applications were developed is the iSBC 80/30 single board computer. This board provides an especially convenient UPI environment since it contains a dedicated socket specifically interfaced for the UPI-41A. The 80/30 uses the 8085A as the master processor. The I/O and peripheral compliment on the 80/30 include 12 vectored priority interrupts (8 on an 8259 Programmable Interrupt Controller and 4 on the 8085A itself), an 8253 Programmable Interval Timer supplying three 16-bit programmable timers (one is dedicated as a programmable baud rate generator), a high speed serial channel provided by a 8251 Programmable USART, and 24 parallel I/O lines implemented with an 8255A Programmable Parallel Interface. The memory compliment contains 16K bytes of RAM using 2117 16K bit Dynamic RAMs and the 8202 Dynamic RAM Controller, and up to 8K bytes of ROM/EPROM with sockets compatible with 2716, 2758, or 2332 devices. The 80/30's RAM uses a dual port architecture. That is, the memory can be considered a global system resource, accessible from the on-board 8085A as well as from remote CPUs and other devices via the MULTIBUS. The 80/30 contains MULTIBUS control logic which allows up to 16 80/30s or other bus masters to share the same system bus. (More detailed information on the iSBC 80/30 and other iSBC products may be found in the latest Intel Systems Data Catalog.)

A block diagram of the iSBC 80/30 is shown in Figure 12. Details of the UPI interface are shown in Figure 13. This interface decodes the UPI registers in the following format:

Register	Operations
Read STATUS	IN E5H
Write DBBIN (command)	OUT E5H
Read DBBOUT (data)	IN E4H
Write DBBIN (data)	OUT E4H

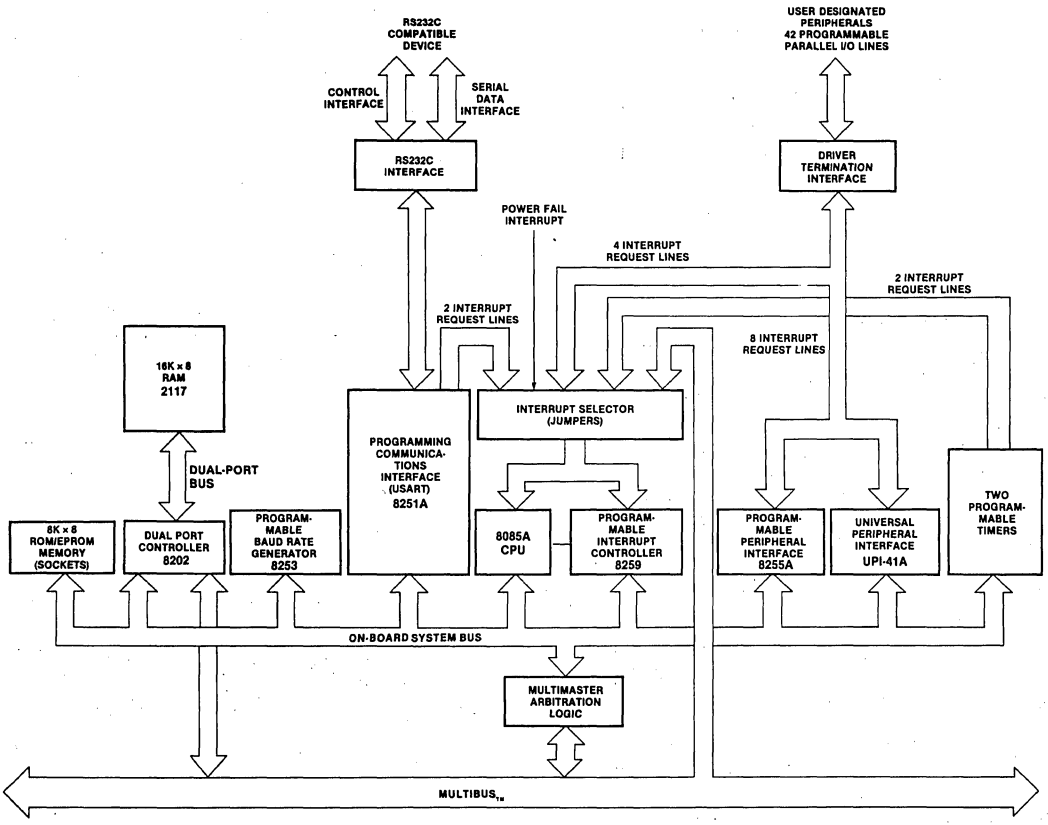


Figure 12. iSBC 80/30 Block Diagram

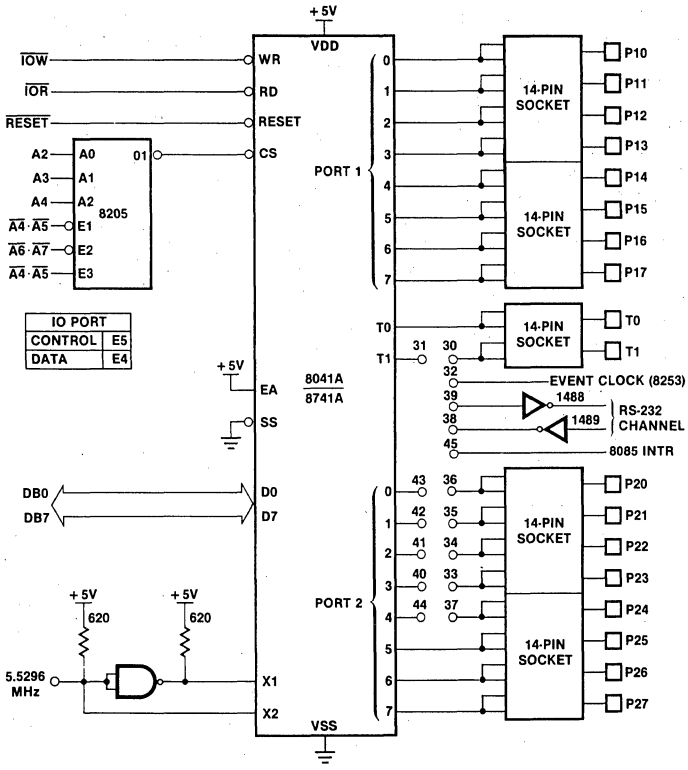


Figure 13. UPI Interface on iSBC 80/30

8-Digit Multiplexed LED Display

The traditional method of interfacing an LED display with a microprocessor is to use a data latch along with a BDC-to-7-segment decoder for each digit of the display. Thus two ICs, seven current limiting resistors, and about 45 connections are required for each digit. These requirements are, of course, multiplied by the total number of digits desired. The obvious disadvantages of this method are high parts count and high power dissipation since each digit is "ON" continuously. Instead, a scheme of time multiplexing the display can be used to decrease both parts count and power dissipation.

Display multiplexing basically involves connecting the same segment (a, b, c, d, e, f, or g) of each digit in parallel and driving the common digit element (anode or cathode) of each digit separately. This is shown schematically in Figure 14. The various digits of the display are not all on at once; rather, only one digit at a time is energized. As each digit is energized, the appropriate segments for that digit are turned on. Each digit is enabled in this way, in sequence, at a rate fast enough to ensure that each digit appears to be "ON" continuously. This implies that the display must be "refreshed" at periodic intervals to keep the digits flicker-free. If the CPU had to handle this task, it would have to suspend normal processing, go update the display, and then return to its normal flow. This extra burden is ideally handled by a UPI. The master CPU could simply give characters to the UPI and let the UPI do the actual segment decoding, display multiplexing, and refreshing.

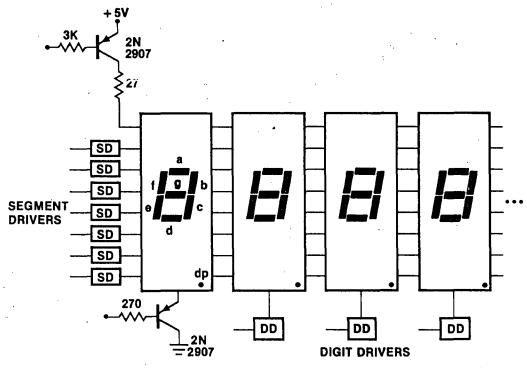


Figure 14. LED Multiplexing

As an example of this technique, Figure 15 shows the UPI controlling an 8-digit LED display. All digit segments are connected in parallel and are driven through segment drivers by the UPI's Port 1. The lower 3 bits of Port 2 are inputs to a 3-to-8 decoder which selects an individual digit through a digit driver. A fourth Port 2 line is used as a decoder enable input. The remaining Port 2 lines plus the T0 and T1 inputs are available for other tasks.

Internally, the UPI uses the counter/timer in the interval timer mode to define the interval between display refreshes. Once the timer is loaded with the desired interval and started, the UPI is free to handle other tasks. It is only when a timer overflow interrupt occurs that the UPI handles the short display multiplexing routine. The display multiplexing can be considered a background task which is entirely interrupt-driven. The amount of time spent multiplexing is such that there is ample time to handle a non-timer task in the UPI foreground. (We'll discuss this timing shortly.)

When a timer interrupt occurs, the UPI turns off all digits via the decoder enable. The next digit's segment contents are retrieved from the internal data memory and output via Port 1 to the segment drivers. Finally, the next digit's location is placed on Port 2 (P20–P22) and the decoder enabled. This displays the digit's segment information until the next interrupt. The timer is then restarted for the next interval. This process continues repeatedly for each digit in sequence.

As a prelude to discussing the UPI software, let's examine the internal data memory structure used in this application, Figure 16. This application requires only 14 of the 64 total data memory locations. The top eight locations are dedicated to the Display Map; one location for each digit. These locations contain the segment and decimal point information for each character. Just how characters are loaded into this section of memory is covered shortly. Register R7 of Register Bank 1 is used as the temporary Accumulator store during the interrupt service routines. Register R3 stores the digit number of the next digit to be displayed. R2 is a temporary storage register for characters during the character input routine. R0 is the offset pointer pointing to the Display Map location of the next digit. That makes 12 locations so far. The remaining two locations are the two stack locations required to store the return address plus status during the timer and input interrupt service routines. The remaining unused locations, all of Register Bank 0, 14 bytes of stack, 4 in Register Bank 1, and 24 general purpose RAM locations, are all available for use by any foreground task.

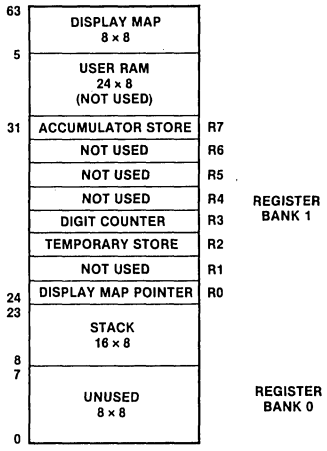


Figure 16. LED Display Controller Data Memory Allocation

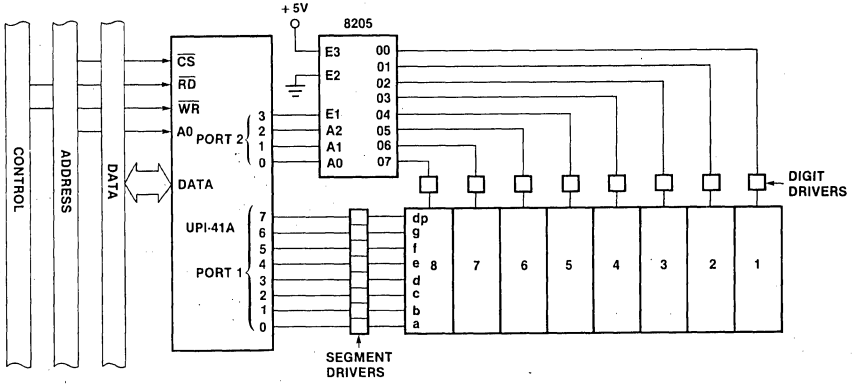


Figure 15. UPI Controlled 8-Digit LED Display

The UPI software consists of only three short routines. One, INIT, is used strictly during initialization. DISPLA is the multiplexing routine called at a timer interrupt. INPUT is the character input handler called at an IBF interrupt. The flow charts for these routines are shown in Figures 17A thru 17C.

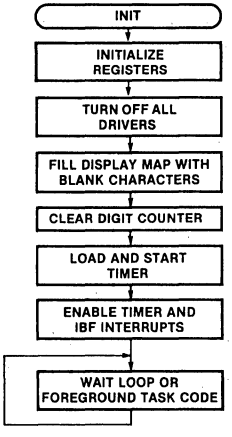


Figure 17A. INIT Routine Flow

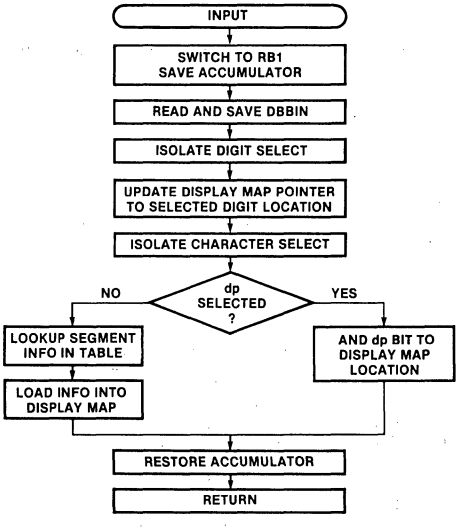


Figure 17B. INPUT Routine Flow

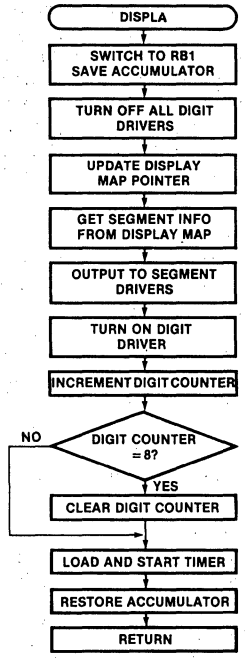


Figure 17C. DISPLA Routine Flow

INIT initializes the UPI by simply turning off all segment and digit drivers, filling the Display Map with blank characters, loading and starting the timer, and enabling both timer and IBF interrupts. Although the flow chart shows the program looping at this point, it is here that the code for any foreground task is inserted. The only restrictions on this foreground task are that it not use I/O lines dedicated to the display and that it not require dedicated use of the timer. It could share the timer if precautions are taken to ensure that the display will still be refreshed at the required interval.

The INPUT routine handles the character input. It is called when an IBF interrupt occurs. After the usual swapping of register banks and saving of the accumulator, DBBIN is read and stored in register R2. DBBIN contains the Display Data Word. The format for this word, Figure 18, has two fields: Digit Select and Character Select. The Digit Select field selects the digit number into which the character from the Character Select field is placed. Notice that the character set is not limited strictly to numerics, some alphanumeric capability is provided. Once DBBIN is read, the offset for the selected digit is computed and placed in the Display Map Pointer R0. Next the segment information for the selected character is found through a look-up table starting in page 3 of the program memory. This segment information is then stored at the location pointed at by the Display Map Pointer. If the Character Select field specified a decimal point, the segment correponding the decimal point is ANDed into the present segment information for that digit. After the accumulator is restored, execution is returned to the main program.

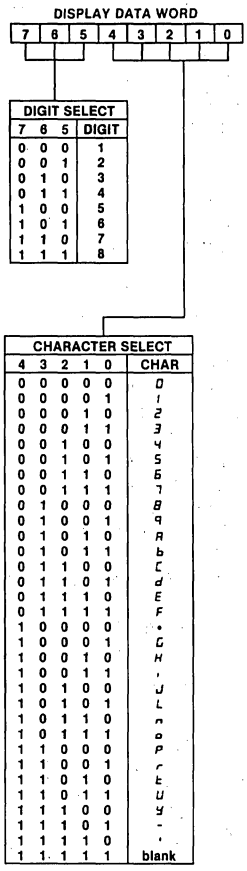


Figure 18. LED Display Controller Display Data Word Format

The DISPLA routine simply implements the multiplexing actions described earlier. It is called whenever a timer interrupt occurs. After saving pre-interrupt status by switching register banks and storing the Accumulator, all digit drivers are turned off. The Display Map Pointer is then updated using the Current Digit Register to point at that digit's segment information in the Display Map. This information is output to Port 1; the segment drivers. The number of the current digit, R3, is then sent to the digit select decoder and the decoder is enabled. This turns on the current digit. The digit counter is incremented and tested to see if all eight digits have been refreshed. If so, the digit counter is reset to zero. If not, nothing is done. Finally, the timer is loaded and restarted, the Accumulator is restored, and the routine returns execution to the main program. Thus DISPLA refreshes one digit each time it is CALLed by the timer interrupt. The digit remains on until the next time DISPLA is executed.

The UPI software listing is included as Appendix A1. Appendix A2 shows the 8085A test routine used to display the contents of a display buffer on the display. The 8085A software takes care of the display digit numbering. Since the application is input-only for the UPI, the only protocol required is that the master must test IBF before writing a Display Data Word into DBBIN.

On the iSBC 80/30, the UPI frequency is at 5.5296 MHz. To obtain a flicker-free display, the whole display must be refreshed at a rate of 50 Hz or greater. If we assume a 50 Hz refresh rate and an 8-digit display, this means the DISPLA routine must be CALLed 50 x 8 or 400 times/sec. This translates, using the timer interval of 87 μs at 5.5296 MHz, to a timer count of 227. (Recall from the UPI-41 User's Manual that the timer is an "8-bit up-counter".) Hence the TIME equate of 227D in the UPI listing. Obviously, different frequency sources or display lengths would require that this equate be modified.

With the UPI running at 5.5296 MHz, the instruction cycle time is 2.713 μs. The DISPLA routine requires 28 instruction cycles, therefore, the routine executes in 76 μs. Since DISPLA is CALLed 400 times/sec, the total time spent refreshing the display during one second is then 30 ms or 3% of the total UPI time. This leaves 97.0% for any foreground tasks that could be added.

While the basic UPI software is useful just as it stands, there are several enhancements that could be incorporated depending on the application. Auto-incrementing of the digit location could be added to the input routine to alleviate the need for the master to keep track of digit numbers. This could be (optionally) either right-handed or left-handed entry a la TI or HP calculators. The character set could be easily modified by simply changing the lookup table. The display could be expanded to 16 digits at the expense of one additional Port 2 digit select line, the replacement of the 3-to-8 decoder with a 4-to-16 decoder, and 8 more Display Map locations.

Now let's move on to a slightly more complex application that is UPI output-only — a sensor matrix controller.

Sensor Matrix Controller

Quite often a microprocessor system is called upon to read the status of a large number of simple SPST switches or sensors. This is especially true in a process or industrial control environment. Alarm systems are also good examples of systems with a large sensor population. If the number of sensors is small, it might be reasonable to dedicate a single input port pin for each sensor. However, as the number of sensors increase, this technique becomes very wasteful. A better arrangement is to configure the sensors in a matrix organization like that shown in Figure 19. This arrangement of 16 sensors requires only 4 input and 4 output lines; half the number needed if dedicated inputs were used. The line saving becomes even more substantial as the number of sensors increases.

In Figure 19, the basic operation of the matrix involves scanning individual row select lines in sequence while reading the column return lines. The state of any particular sensor can then be determined by decoding the row and column information. The typical configuration

pulls up the column return lines and the selected row is held low. Deselected rows are held high. Thus a return line remains high for an open sensor on the selected row and is pulled low for a closed sensor. Diode isolation is used to prevent a phantom closure which would occur when a sensor is closed on a selected row and there are two or more closures on a deselected row. Germanium diodes are used to provide greater noise margin at the return line input.

If the main processor was required to control such a matrix it would periodically have to output at the row port and then read the column return port. The processor would need to maintain in memory a map of the previous state of the matrix. A comparison of the new return information to the old information would then be made to determine whether a sensor change had occurred. Any changes would be processed as needed. A row counter and matrix map pointer also require maintenance each scan. Since in most applications sensors change very slowly compared to most processing actions, the processor probably would scan the rows only periodically with other tasks being processed between scans.

Rather than require the processor to handle the rather mundane tasks of scanning, comparing, and decoding the matrix, why not use a dedicated processor? The UPI is perfect.

Figure 20 shows a UPI configuration for controlling up to 128 sensors arranged in a 16 × 8 matrix. The 4-to-16 line decoder is used as the row selector to save port pins and provides the expansion to 128 sensors over the maximum of 64 sensors if the port had been used directly. It also helps increase the port drive capability. The column return lines go directly into Port 1. Features of this design include complete matrix management. As the UPI scans the matrix it compares its present status to the previous scan. If any change is detected, the location of the change is decoded and loaded, along with the sensor's present state, into DBBOUT. This byte is called a Change Word. The Master processor has only to read one byte to determine the status and coordinate of a changed sensor. If the master had not read a previous Change Word in DBBOUT (OBF = 1) before a new sensor change is detected, the new Change Word is loaded into an internal FIFO. This FIFO buffers up to 40 changes before it fills. The status of the FIFO and OBF is made available to the master either by polling the UPI STATUS register, Figure 21A, or as interrupt sources on port pins P24 and P25 respectively, Figure 20. The FIFO NOT EMPTY pin and bit are true as long as there are changes not yet read in the FIFO. As long as the FIFO is not empty, the UPI monitors OBF and loads new Change Words from the FIFO into DBBOUT. Thus, the UPI provides complete FIFO management.

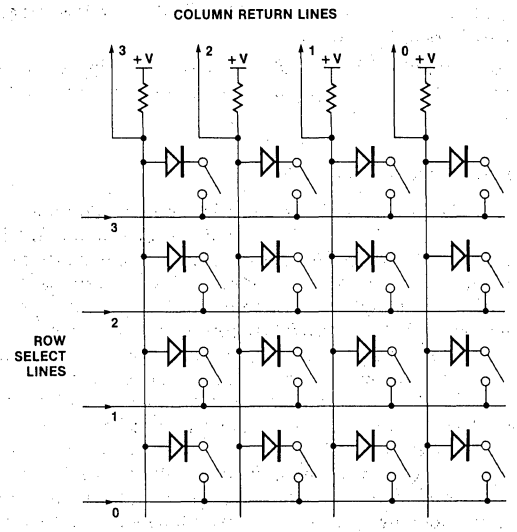


Figure 19. 4 × 4 Sensor Matrix

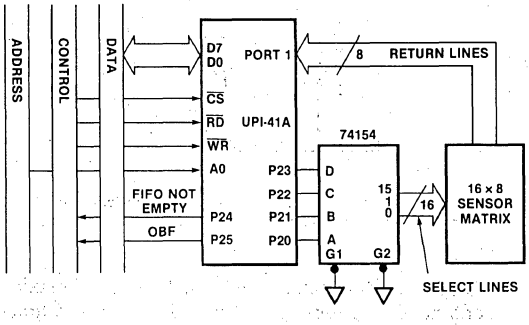


Figure 20. 128 Sensor Matrix Controller

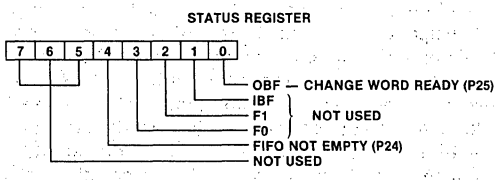


Figure 21A. Sensor Matrix Status Register Format

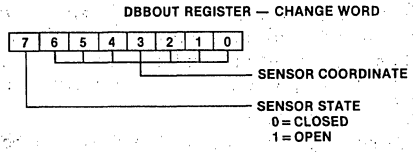


Figure 21B. Sensor Matrix Change Word Format

Internally, the matrix scanning software is programmed to run as a foreground task. This allows the timer/counter to be used by any background task although the hardware configuration leaves only 2 inputs (T0 and T1) plus 2 I/O port pins available. Also, to add a background task, the FIFO would have to be made smaller to accommodate the needed register and data memory space. (It would be possible however to turn the table here and make the scanning software timer/counter interrupt-driven where the timer times the scan interval.)

The data memory organization for this application is shown in Figure 22. The upper 16 bytes form the Matrix Map and store the sensor states from the previous scan; one bit for each sensor. The Change Word FIFO occupies the next 40 locations. (The top and bottom addresses of this FIFO are treated as equate variables in the program so that the FIFO size may easily be changed to accommodate the register needs of other tasks.) Register R0 serves as a pointer into the matrix map area for comparisons and updates of the sensor status. R1 is a general FIFO pointer. The FIFO is implemented as a circular buffer with In and Out pointer registers which are stored in R4 and R5 respectively. These registers are moved into FIFO pointer R1 for actual transfers into or out of the FIFO. R2 is the Row Select Counter. It stores the number of the row being scanned.

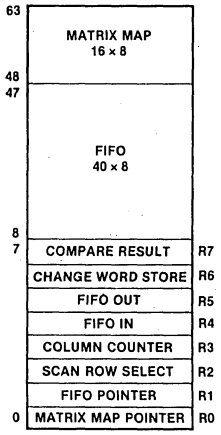


Figure 22. Sensor Matrix Data Memory Map

Register R3 is the Column Counter. This counter is normally set to 00H; however, when a change is detected somewhere in a particular row, it is used to inspect each sensor status bit individually for a change. When a changed sensor bit is found, the Row Select Counter and Column Counter are combined to give the sensor's matrix coordinate. This coordinate is temporarily stored in the Change Word Store, register R6. Register R7 is the Compare Result. As each row is scanned, the return information is Exclusive-OR'd with the return information from the previous scan of that row. The result of this operation is stored in R7. If R7 is zero, there have been no changes on that row. A non-zero result indicates at least one changed sensor.

The basic program operation is shown in the flow chart of Figure 23. At RESET, the software initializes the working registers, the ports, and clears the STATUS register. To get a starting point from which to perform the sensor comparisons, the current status of the matrix is read and stored in the Matrix Map. At this point, the UPI begins looking for changed sensors starting with the first row.

Before delving further into the flow, let's pause to describe the general format of the operation. The UPI scans the matrix one row at a time. If no changes are detected on a particular row, the UPI simply moves to the next row after checking the status of DBBOUT and the FIFO. If a change is detected, the UPI must check each bit (sensor) within the row to determine the actual sensor location. (More than one sensor on the scanned row could have changed.) Rather than test all 8 bits of the row before checking the DBBOUT and FIFO status again, the UPI performs the status check in between each of the bit tests. This ensures the fastest response to the master reading previous Change Words from DBBOUT and the FIFO.

With this general overview in mind, let's go first thru the flow chart assuming we are scanning a row where no changes have occurred. Starting at the Scan-and-Compare section, the UPI first checks if the entire matrix has been scanned. If it has, the various pointers are reset. If not, the address of the next row is placed on Port 20 thru 23. This selects the desired row. The state of the row is then read on Port 1; the column return lines. This present state is compared to the previous state by retrieving the previous state from the matrix map and performing an Exclusive-OR with the present state. Since we are assuming that no change has occurred, the result is zero. No coordinate decoding is needed and the flow branches to the FIFO-DBBOUT Management section.

The FIFO-DBBOUT Management section simply maintains the FIFO and loads DBBOUT whenever Change Words are present in the FIFO and DBBOUT is clear (OBF = 0). The section first tests if the FIFO is full. (If we assume our "no-change" row is the first row scanned, the FIFO obviously would not be full.) If it is, the UPI waits until OBF = 0, at which point the next Change Word is retrieved from the FIFO and placed in DBBOUT. This "unfills" the FIFO making room for more Change Words. At this point, the Column Counter, R3, is checked. For rows with no changes, the Column

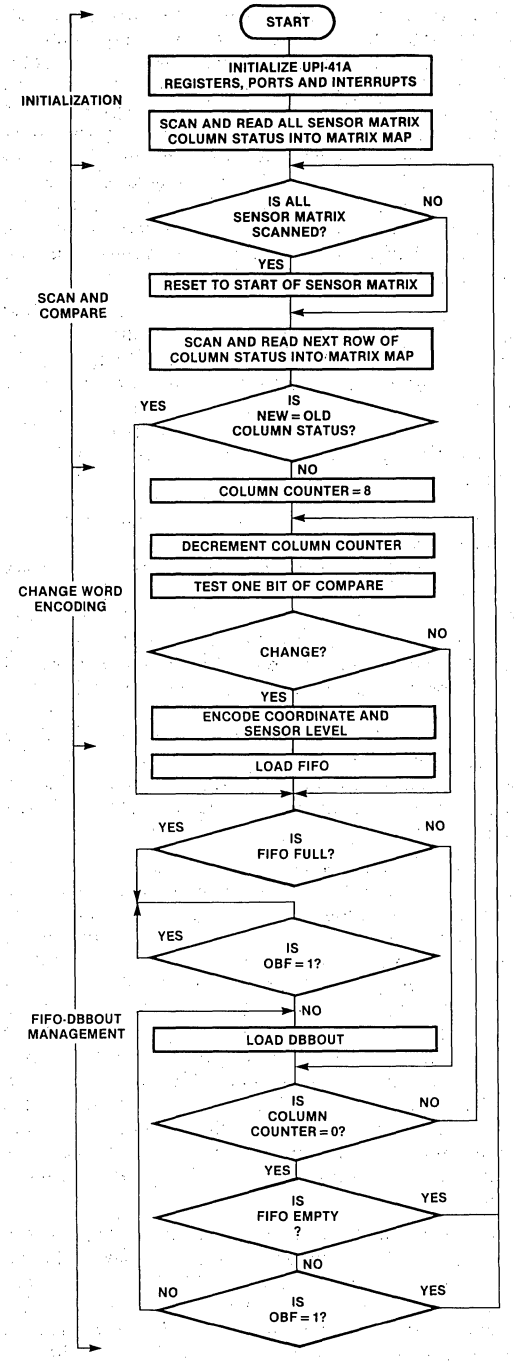


Figure 23. Sensor Matrix Controller Flow Chart

Counter is always zero so the test simply falls through. (We cover the case for changes shortly.) Now the FIFO is tested for being empty. If it is, there is no sense in any further tests so the flow simply goes back up to scan the next row. If the FIFO is not empty, DBBOUT is tested again through OBF. If a Change Word is in DBBOUT waiting for the master to read it, nothing can be done and the flow likewise branches up for the next row. However, if the DBBOUT is free and remembering that the previous test showed that the FIFO was not empty, DBBOUT is loaded with the next Change Word and the last two conditional tests repeat.

Now let's assume the next row contains several changed sensors. Like before, the row is selected, the return lines read, and the sensor status compared to the previous scan. Since changes have occurred, the Exclusive-OR result is now non-zero. Any 1s in the result reflect the positions of the changed sensors. This non-zero result is stored in the Compare Result register, R7. At this point, the Column Counter is preset to 8. To determine the changed sensors' locations, the Compare Result register is shifted bit-by-bit to the left while decrementing the Column Counter. After each shift, bit 7 of the result is tested. If it is a one, a changed sensor has been found. The Column Counter then reflected the sensor's matrix column position while the Scan Row Select register holds it row position. These registers are then combined in R6, the Change Word Store, to form the sensor's matrix coordinate section of the Change Word. The 8th bit of the Change Word Store is coded with the sensor's present state (Figure 21). This byte forms the complete Change Word. It is loaded into the next available FIFO position. If bit 7 of the Compare Result had been a zero, that particular sensor had not changed and the coordinate decoding is not performed.

In between each shift, test, and coordinate encode (if necessary), the FIFO-DBBOUT Management is performed. It is the Column Counter test within this section that routes the flow back up to the Change Word Encoding section if the entire Compare Result (row) has not been shifted and tested.

The FIFO is implemented as a circular buffer with IN and OUT pointers (R4 and R5 respectively). The operations of the FIFO is best understood using an example, Figure 24. This series of figures show how the FIFO, DBBOUT, and OBF interact as changes are detected and Change Words are read by the master. The letters correspond to sequential Change Words being loaded into the FIFO. Note that the figures show only a 4 × 8 FIFO however, the principles are the same in the 40 × 8 FIFO.

Figure 24A shows the condition where no Change Words have been loaded into the FIFO or DBBOUT. In Figure 24B a change, "A", has been detected, decoded, and loaded into the FIFO at the location equal to the value of the FIFO-IN pointer. The FIFO-IN pointer is then incremented and the FIFO-OUT pointer is reset to the bottom of the FIFO since it had reached the FIFO top. Now that a Change Word is in the FIFO, OBF is checked to see if DBBOUT is empty. Because OBF = 0, DBBOUT is empty and the Change Word is loaded from the FIFO location pointed at by the FIFO-OUT pointer. This is shown in Figure 24C. Loading DBBOUT automatically

sets OBF. OBF remains set until the master reads DBBOUT. Figures 24D and 24E show two more Change Words loaded into the FIFO. In Figure 24F the first Change Word is finally read by the master resetting OBF. This allows the next Change Word to be loaded into DBBOUT. Note that each time the FIFO is loaded, the FIFO-IN pointer increments. Each time DBBOUT is read the FIFO-OUT pointer increments unless there are no more Change Words in the FIFO. Both pointers wrap-around to the bottom once they reach the FIFO top. The remaining figures show more Change Words being loaded into the FIFO. When the entire FIFO fills and DBBOUT can not be loaded (OBF = 1), scanning stops until the master reads DBBOUT making room for more Change Words.

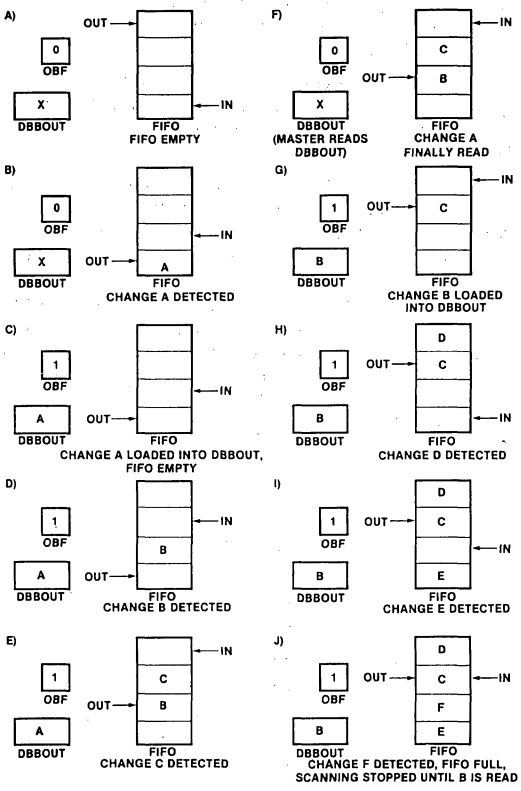


Figure 24A-J. FIFO Operation Example

As was mentioned earlier, two interrrupt outputs to the master are available: Change Word Ready (P25, OBF) and FIFO NOT EMPTY (P24). The Change Word Ready interrupt simply reflects OBF and is handled automatically by the UPI since an EN FLAGS instruction is executed during initialization. The FIFO NOT EMPTY interrupt is generated and cleared as appropriate, each pass through the FIFO management code.

No debouncing is provided although it could be added. Rather, the scan time is left as an equate variable so that it could be varied to account for both debounce time and expected sensor change rates. The minimum scan time for this application is 2 msec when using a 6 MHz clock. Since the matrix controller is coded as a foreground task, scan time simply uses a software delay loop.

The UPI software is included as Appendix B1. Appendix B2 is 8085A test software which builds a Change Word buffer starting at BUFSRT. This software simply polls the STATUS register looking for Change Word Ready to go true. DBBOUT is then read and loaded into the buffer. Now let's move on to an application which combines both the foreground and background concepts.

Combination I/O Device

The final UPI application was designed especially to add additional serial and parallel I/O ports to the iSBC 80/30. This UPI simulates a full-duplex UART (Universal Asynchronous Receiver/Transmitter) combined with an 8-bit parallel I/O port. Features of the UART include: software selectable baud rates (110, 300, 600, or 1200 baud), double buffering for both the transmitter and receiver, and receiver testing for false state bit, framing, and overrun errors. For parallel I/O, one 8-bit port is programmable for either input or output. The output port is statically latched and the input port is sampled.

Figure 25 shows the interface of this combination I/O device to the dedicated UPI socket on the iSBC 80/30. The only external requirement is a 76.8 kHz source which serves as the baud rate standard. The internal baud rates are generated as multiples of this external clock. This clock is obtained from one of the 8253 counters. Otherwise, an RS-232 driver and receiver already available for UPI use in serial I/O applications. Sockets are also provided for termination of the parallel port.

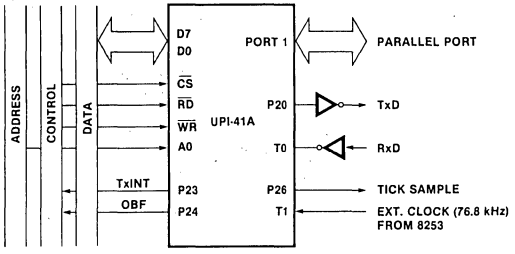


Figure 25. Combination I/O Device

There are three commands for this application. Their format is shown in Figure 26. The CONFIGURE command specifies the serial baud rate and the parallel I/O direction. Normally this command is issued once during system initialization. The I/O command causes a parallel I/O operation to be performed. If the parallel port direction is output, the UPI expects the data byte immediately following an I/O command to be data for the output port. If the port is in the input direction, an I/O command causes the port to be read and the data placed in DBBOUT. The RESET ERROR command resets the serial receiver error bits in the STATUS register.

The STATUS register format is shown in Figure 27. Looking at each bit, bit 0 (OBF) is the DATA AVAILABLE flag. It is set whenever the UPI places data into DBBOUT. Since the data may come from either the receiver or the parallel input port, the F0 and F1 flags (bits 2 and 3) code the source. Thus, when the master finds OBF set, it must decode F0 and F1 to determine the source.

Bit 1 (IBF) functions as a busy bit. When IBF is set, no writes to DBBIN are allowed. Bit 5 is the TxINT (Transmitter Interrupt) bit. It is asserted whenever the transmitter buffer register is empty. The master uses this bit to determine when the transmitter is ready to accept a data character.

COMMAND FORMAT

7	6	5	4	3	2	1	0	
								CONFIGURE COMMAND
0	0	0	A	B	C	D	P	A — 1200 BAUD SELECT B — 600 BAUD SELECT C — 300 BAUD SELECT D — 110 BAUD SELECT P — PARALLEL I/O DIRECTION 0—INPUT 1—OUTPUT
1	0	0	0	0	0	0	0	I/O COMMAND
1	1	0	0	0	0	0	0	RESET ERROR COMMAND

Figure 26. Combination I/O Command Format

STATUS FORMAT

7	6	5	4	3	2	1	0
FRAMING ERROR	OVERRUN ERROR		Tx INTERRUPT	NOT USED	F1	F0	IBF — BUSY / OBF — DATA AVAILABLE

F0	F1	OPERATION (IBF = 1)
0	0	NO OPERATION
0	1	PARALLEL I/O DATA
1	0	SERIAL I/O DATA
1	1	COMMAND ERROR

Figure 27. STATUS Register Format

Bits 6 and 7 are receiver error flags. The framing error flag, bit 6, is set whenever a character is received with an invalid stop bit. Bit 7, overrun error, is set if a character is received before the master has read a previous character. If an overrun occurs, the previous character is overwritten and lost. Once an error occurs, the error flag remains set until reset by a RESET ERROR command. A set error flag does not inhibit receiver operation however.

Figure 28 shows the port pin definition for this application. Port 1 is the parallel I/O port. The UART uses Port 2 and the Test inputs. P20 is the transmitter data out pin. It is set for a mark and reset for a space. P23 is a transmitter interrupt output. This pin has the same timing as the TxINT bit in the STATUS register. It is normally used in interrupt-driven systems to interrupt the master processor when the transmitter is ready to accept a new data character.

The OBF flag is brought out on P24 as a master interrupt when data is available in DBBOUT. P26 is a diagnostic pin which pulses at four times the selected baud rate. (More about this pin later.) The receiver data input uses the T0 input. One of the Port 2 pins could have been used, however, the software can test the T0 in one instruction without first reading a port.

The T1 input is the baud rate external source. The UART divides this input to determine the timing needed for the selected baud rate. The input is a non-synchronous 76.8 kHz source.

Internally, when the CONFIGURE command is received and the selected baud rate is determined, the internal timer/counter is loaded with a baud rate constant and started in the event counter mode. Timer/counter interrupts are then enabled. The baud rate constant is selected to provide a counter interrupt at four times the desired baud rate. At each interrupt, both the transmitter and receiver are handled. Between interrupts, any new commands and data are recognized and executed.

PORT PIN DEFINITION

PORT	BIT	FUNCTION
1	0-7	PARALLEL I/O
2	0	Tx DATA
	1	NOT USED
	2	NOT USED
	3	Tx INTERRUPT
	4	OBF INTERRUPT
	5	NOT USED
	6	NOT USED (TICK SAMPLE)
	7	NOT USED
T0		Rx DATA
T1		EXTERNAL CLOCK (76.8 kHz)

Figure 28. Combination I/O Port Definition

As a prelude to discussing the flow charts, Figure 29 shows the register definition. Register Bank 0 serves the UART receiver and parallel I/O while Register Bank 1 handles the UART transmitter and commands. Looking at RB0 first, R3 is the receiver status register, RxSTS. Reflected in the bits of this register is the current receiver status in sequential order. Figure 30 shows this bit definition. Bit 0 is the Rx flag. It is set whenever a possible start bit is received. Bit 1 signifies that the start bit is good and character construction should begin with the next received bit. Bit 1 is the Good Start flag. Bit 2 is the Byte Finished flag. When all data bits of a character are received, this flag is set. When all the bits, data and stop bits are received, the assembled character is loaded into the holding register (R4 in Figure 29) bit 3, the Data Ready flag, is set. The foreground routine which looks for commands and data continuously, looks at this bit to determine when the receiver has received a character. Bits 4 and 5 signify any error conditions for a particular character.

The parallel I/O port software uses bits 6 and 7. Bit 6 codes the I/O direction specified by the last CONFIGURE command. Bit 7 is set whenever an I/O command is received. The foreground routine tests this bit to determine when an I/O operation has been requested by the master.

As was mentioned, R4 is the receiver holding register. Assembled characters are held in this register until the foreground routine finds DBBOUT free, at which time the data is transferred from R4 to DBBOUT. R5 is the receiver tick counter. Recall that counter interrupts occur at four times the baud rate. Therefore, once a start bit is found, the receiver only needs to look at the data every four interrupts or tick counts. R5 holds the current tick count.

63	USER RAM (NOT USED)		
32			
31	AC TEMP. STORE	R7	
30	COMMAND STORE	R6	
29	Tx STATUS—TxSTS	R5	
28	Tx BUFFER	R4	REGISTER BANK 1
27	Tx SERIALIZER	R3	
26	Tx TICK COUNTER	R2	
25	BAUD RATE CONSTANT	R1	
24	NOT USED	R0	
23	STACK (ONE LEVEL USED)		
8			
7	STATUS STORE	R7	
6	Rx DESERIALIZER	R6	
5	Rx TICK COUNTER	R5	
4	Rx HOLDING	R4	REGISTER BANK 0
3	Rx STATUS—RxSTS	R3	
2	NOT USED	R2	
1	NOT USED	R1	
0	NOT USED	R0	

Figure 29. Combination I/O Register Map

RxSTS FORMAT

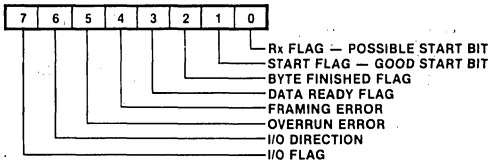


Figure 30. RxSTS Register

R6 is the receiver de-serializing register. Data characters are assembled in this register. R6 is preset to 80H when a good start bit is received. As each bit is sampled every four timer ticks, they are rotated into the leftmost bit of R6. The software knows the character assembly is complete when the original preset bit rotates into the carry.

An image of the upper 4 bits of the STATUS register is stored in R7. These bits are the TxINT, Framing and Overrun bits. This image is needed since the UPI may load the upper 4 STATUS register bits from its accumulator; however, it cannot read STATUS directly.

In Register Bank 1 (Figure 29), R1 holds the baud rate constant which is found from decoding the baud rate select bits of the CONFIGURE command. The counter is reloaded with this constant every timer tick. Like the receiver, the transmitter only needs to update the transmitter output every four ticks. R2 holds the transmitter tick count. The value of R2 determines which portion of the data is being transmitted; start bit, data bits, or stop bit. The transmit serializer is R3. R3 holds the data character as each character bit is transmitted.

R4 is the transmitter holding register. It provides the double buffering for the transmitter. While transmitting one character, it is possible to load the next character into R4 via DBBIN. The TxINT bit in STATUS and pin on Port 2 reflect the "fullness" of R4. If the holding register is empty, the interrupt bit and pin are set. They are reset when the master writes a new data byte for the transmitter into DBBIN. The transmitter Status register (TxSTS) is R5. Like RxSTS, TxSTS contains flag bits which indicate the current state of the transmitter. This flag bit format is shown in Figure 31.

TxSTS bit 0 is the Tx flag. It is set whenever the transmitter is transmitting a character. It is set from the beginning of the start bit until the end of the stop bit. Bit 1 is the Tx Request flag. This bit is set by the foreground routine when it transfers a new character from DBBIN to the Tx Holding register, R4. The transmitter software uses this flag to tell if new data is available. It is reset when the transmitter transfers the character from the holding register to the serializer.

Bit 2 is the Pipelined Tx Data Bit. The transmitter uses a pipelining technique which sets up the next output level in bit 2 after processing the current timer tick. The output level is always changed at the same point after a timer tick interrupt. This technique ensures that no bit timing distortion results from different length processing paths through the receiver and transmitter routines.

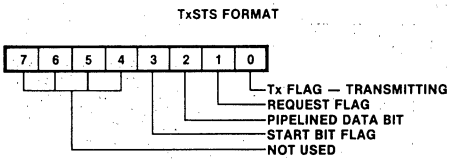


Figure 31. TxSTS Register

Bit 3 of TxSTS is the Start Bit flag. It is set by the transmitter when the start bit space is set up in the Pipelined Data Bit. This allows the transmitter to differentiate between the start bit and data bits on following timer ticks.

The flow charts for this application are shown in Figures 32A-F. At reset, the INIT routine is executed which initializes the registers and port pins. After initialization, IBF and OBF are tested in MNLOOP. These flags are tested continually in this loop. If IBF is set, F1 is tested for command or data and execution is transferred to the appropriate routine (CMD or DATA). If IBF = 0, OBF is checked. If OBF = 0 (DBBOUT is free), the Rx Data Ready and I/O flags in RxSTS are tested. If Rx Data Ready is set, the received data is retrieved from the Rx Holding register and transferred to DBBOUT. Any error flags associated with that data are also transferred to STATUS. If the I/O flag is set and the I/O direction is input, Port 1 is read and the data transferred to DBBOUT. In either case, F0 and F1 are set to indicate the data source.

If IBF is set by a command write to DBBIN, CMD reads the command and decodes the desired operation. If an

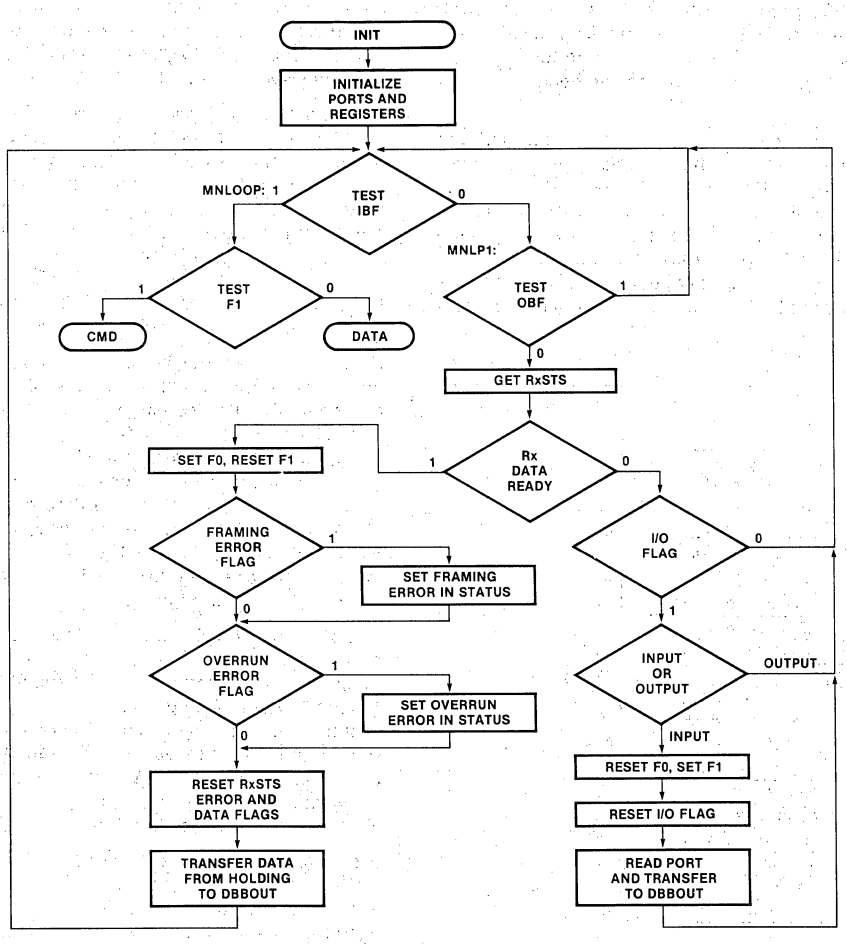


Figure 32A. INIT Flow Chart

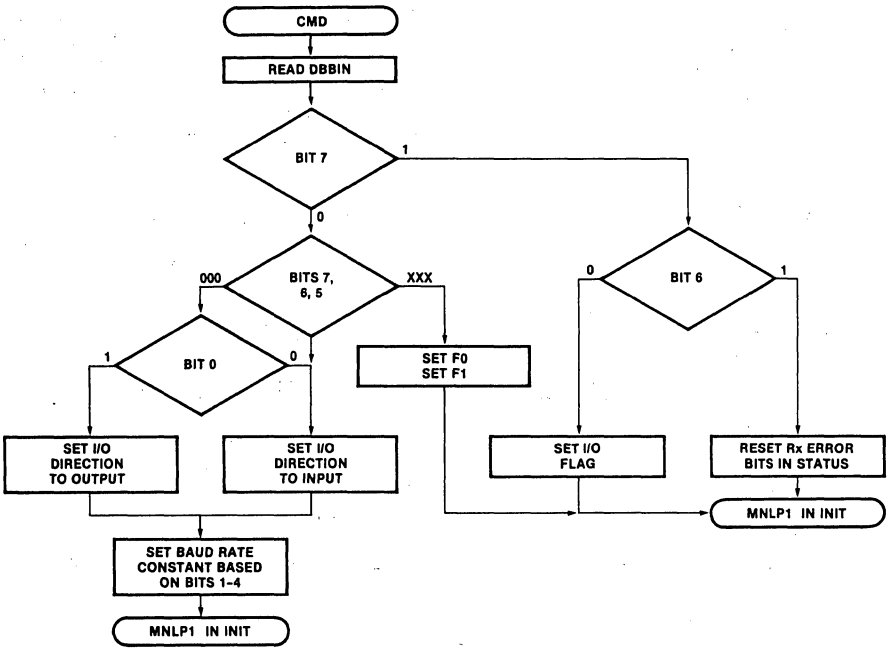


Figure 32B. CMD Flow Chart

I/O operation is specified, the I/O flag is set to indicate to the MNLOOP and DATA routines that an I/O operation is to be performed. If the command is a CONFIGURE command, the constant for the selected baud rate is loaded into both Baud Rate Constant register and the timer/counter. The timer/counter is started in the event counter mode and timer/counter interrupts are enabled. In addition, the I/O port is initialized to all 1's if the I/O direction bit specifies an input port. If the command is a RESET ERROR command, the two error flags in STATUS are cleared.

If the IBF flag is set by a data write, the DATA routine reads DBBIN and places the data in the appropriate place. If the I/O flag is set, the data is for the output port so the port is loaded. If the I/O flag is reset, the data is for the UART transmitter. Data for the transmitter resets the TxINT bit and pin plus sets the Tx Request flag in TxSTS. The data is transferred to the Tx Holding register, R4.

Once a CONFIGURE command is received and the counter started, timer/counter interrupts start occurring at four times the selected baud rate. These interrupts cause a vector to the TIMINT routine, Figure 32D. A 76.8 kHz counter input provides a 13.02 μs counter resolution. Since it requires several UPI instruction cycles to reload the counter, the counter is set to two counts less than the desired baud rate and the counter is reloaded in TIMINT synchronous with the second low-going transition after the interrupt. Once the counter is reloaded, an output port (P26) is toggled to give an external indication of internal counter interval. This is a helpful diagnostic feature. After the tick sample output, the pipelined transmitter data in TxSTS is output to the TxD pin. Although this occurs every timer tick, the pipelined data is changed only every fourth tick.

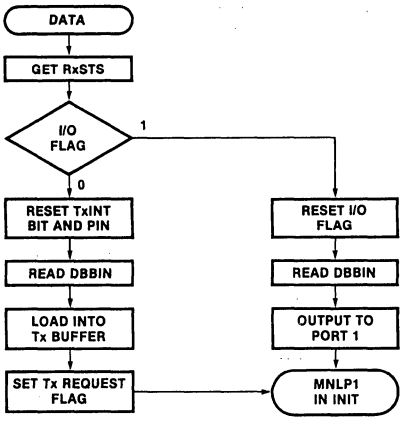


Figure 32C. Data Flow Chart

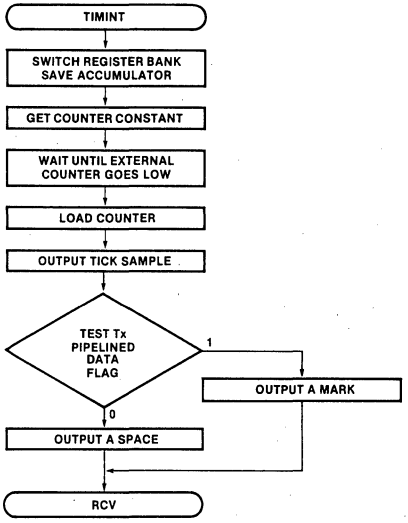


Figure 32D. TIMINT Flow Chart

The receiver is now handled, Figure 32E. The RX flag in RxSTS is examined to see if the receiver is currently in the process of receiving a character. If it is not, the RxD input is tested for a space condition which might indicate a possible start bit. If the input is a mark, no start bit is possible and execution branches to the transmitter flow, XMIT. If the input is a space, the Rx flag is set before proceeding with XMIT.

If the Rx flag is found set when entering RCV, the receiver is in the process of receiving a character. If so, the Start Bit flag is then tested to determine if a good start bit so the Start Bit flag is set, the Rx Tick Counter is initialized to four, and the Rx De-serializer initialized to 80H. A mark indicates a bad start bit so the Rx flag is reset to abort the reception.
start bit so the Start Bit flag is set, the Rx tick counter is initialized to four, and the Rx deserializer initialized to 80H. A mark indicates a bad start bit so the Rx flag is reset to abort the reception.

If the Start Bit flag is set, the program is somewhere in the middle of the received character. Since the data should be sampled every fourth timer tick, the Tick Counter is decremented and tested for zero. If non-zero no sample is needed and execution continues with

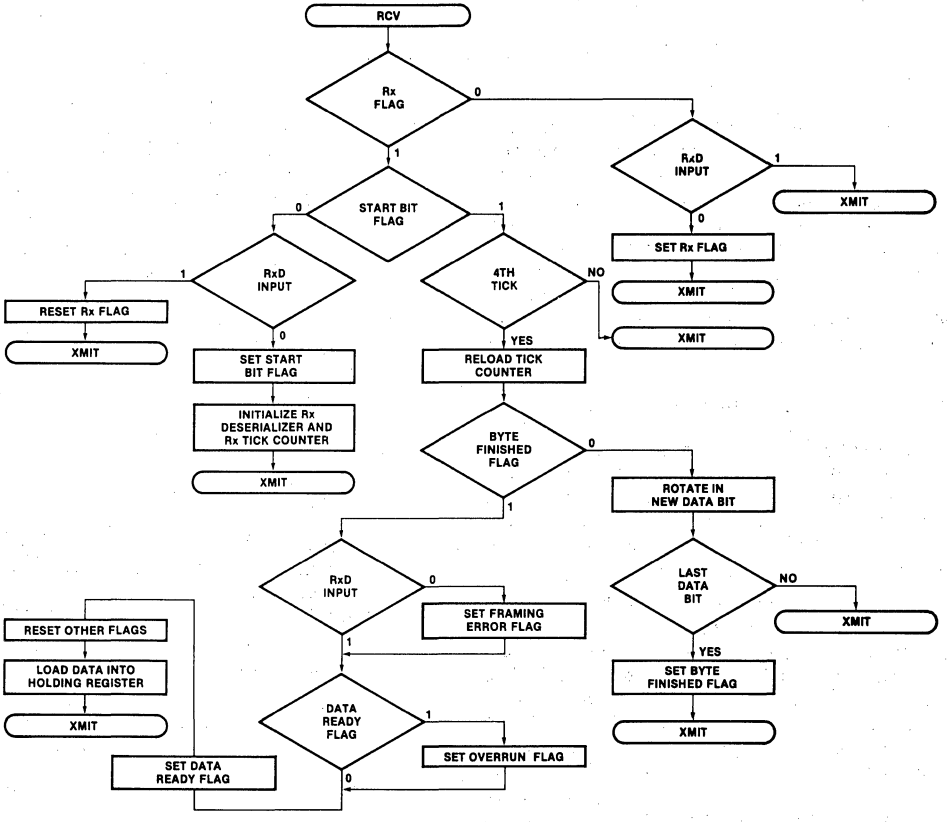


Figure 32E. RCV Flow Chart

XMIT. If zero, the tick counter is reset to four. Now the Byte Finished flag is tested to determine if the data sample is a data or stop bit. If reset, the sample is a data bit. The sample is done and the new bit rotated into the Rx deserializer. If this rotate sets the carry, that data bit was the last so the Byte Finished flag is set. If the carry is reset, the data bit is not the last so execution simply continues with XMIT.

Had the Byte Finished flag been set, this sample is for the stop bit. The RxD input is tested and if a space, the Framing Error flag is set. Otherwise, it is reset. Next, the Rx Data Ready flag is tested. If it is set, the master has not read the previous character so the Overrun Error flag is set. Then the Rx Data Ready flag is set and the received data character is transferred into the Rx Holding register. The Rx, Start Bit, and Byte Finished flags are reset to get ready for the next character.

Execution of the transmitter routine, XMIT, follows the receiver, Figure 32F. The transmitter starts by checking the Start Bit flag in TxSTS. Recall that the actual transmit data is output at the beginning of the timer routine. The Start Bit flag indicates whether the current timer tick interrupt started the start bit. If it is set, the pipelined data output earlier in the routine was the start of the start bit so the flag is reset and the Tx tick counter is initialized. Nothing else is done this timer tick so the routine returns to the foreground.

If the Start Bit flag is reset, the Tx tick counter is incremented and tested. The test is performed module 4. If the counter mod 4 is not zero, it has not been four ticks since the transmitter was handled last so the routine simply returns. If the counter mod 4 is zero, it is time to handle the transmitter and the Tx flag is tested.

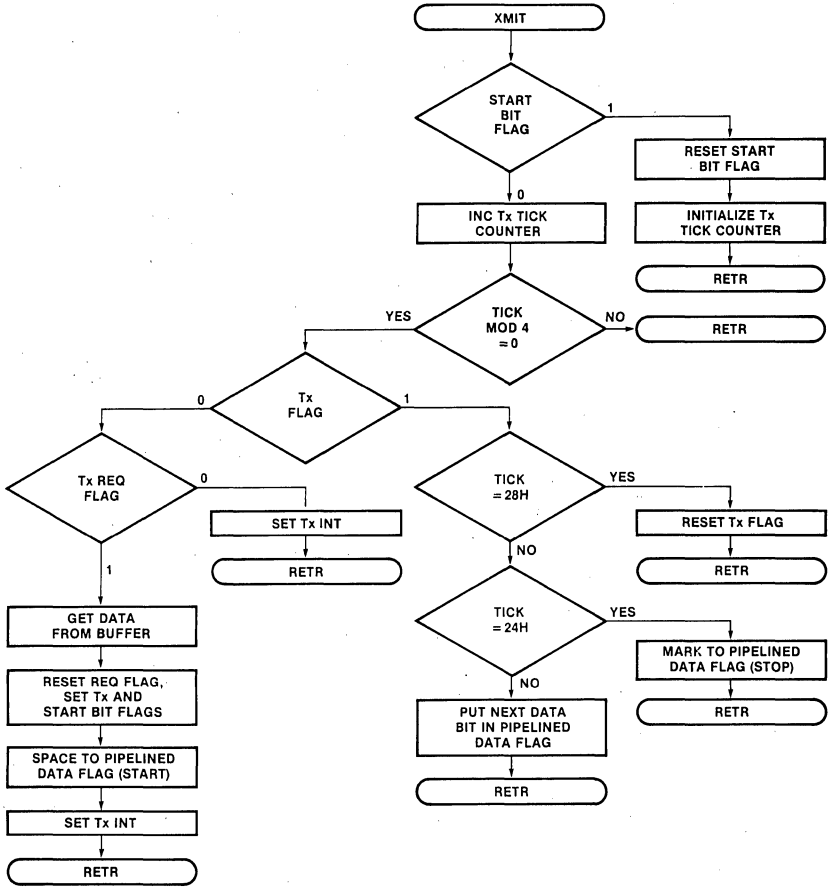


Figure 32F. XMIT Flow Chart

The Tx flag indicates whether the transmitter is active. If the transmitter is inactive, no character is currently being transmitted so the Tx Request flag is tested to see if a new character is waiting in the Tx buffer. If no character is waiting (Tx Request Flag = 0), the Tx interrupt pin and bit are set before returning to the foreground. If there is a character waiting, it is retrieved from the buffer and placed in the Tx serializer. The Tx Request flag is reset while the Tx and Start Bit flags are set. A space is placed in the Tx Pipelined Data bit so a start bit will be output on the next tick. Since the Tx buffer is now empty, the Tx interrupt bit and pin are set to indicate the availability of the buffer to the master. The routine then returns to the foreground.

If the tick counter mod 4 is zero and the Tx flag indicates the transmitter is in the middle of a character, the tick counter is checked to see what transmitter operation is needed. If the counter is 28H (40D), all data bits plus the stop bits are complete. The character is therefore done and the Tx flag is reset. If the counter is 24H (36D), the data bits are complete and the next output should be a mark for the stop bit so a mark is loaded into the Tx Pipelined Data bit.

If neither of the above conditions are met for the counter, the transmitter is some place in the data field, so the next data bit is rotated out of the Tx serializer into the Pipelined Data bit. The next tick outputs this bit.

At this point the program execution is returned to the foreground.

That completes the discussion of the combination I/O device flow charts. The UPI software listing is shown in Appendix C1. Appendix C2 is example 8085A driver software.

Several observations concerning the drivers are appropriate. Notice that since the receiver and input port of the UPI use the OBF flag and interrupt output, the interrupt and flag are cleared when the master reads DBBOUT. This is not true for the transmitter. There is always some time after a master write of new transmitter data before the transmitter interrupt bit and pin are cleared. Thus in an interrupt-driven system, edge-sensitive interrupts should be used. For polled-systems, the software must wait after writing new data for IBF = 0 before re-examining the Tx Interrupt flag in STATUS.

Notice that this application uses none of the user Data Memory above Register Bank 1 and only 361 bytes of Program Memory. This leaves the door open for many improvements. Improvements that come to mind are increased buffering of the transmit or received data, modem control pins, and parallel port handshaking inputs.

This completes our discussion of specific UPI applications. Before concluding, let's look briefly at two debug techniques used during the development of these applications that you might find useful in your own designs.

DEBUG TECHNIQUES

Since the UPI is essentially a single-chip microcomputer, the classical data, address, and control buses are not available to the outside world during normal operation. This fact normally makes debugging a UPI design difficult; however, certain "tricks" can be included in the UPI software to ease this task.

If a UPI is handling multiple tasks, it is usually easier to code and debug each task individually. This is fairly standard procedure. Since each task usually utilizes only a subset of the total number of I/O pins, coding only one task leaves some I/O pins free. Port output instructions can then be added in the task code being debugged which toggle these unused pins to determine which section of task code is being executed at any particular time. The task can also be made to "wait" at various points by using an extra pin as an input and adding code to loop until a particular input condition is met.

One example of using an extra pin as an output is included in the combination serial/parallel device code. During initial development the receiver was not receiving characters correctly. Since this could be caused by incorrect sampling, three lines of code were added to toggle bit 6 of Port 2 at each tick of the sample clock. This code is at lines 184 and 185 of the listing. Thus by looking at the location of the tick sample pulse with respect to the received bit, the UPI sampling interval can be observed. The tick sample time was incorrect and the code was modified accordingly. Similar techniques could be applied at other locations in the program.

The EPROM version of the UPI (8741A) also contains another feature to aid in debug: the capability to single step thru a program. The user may step thru the program instruction-by-instruction. The address of the next instruction to be fetched is available on Port 1 and the lower 2 bits of Port 2. Figure 33 shows the timing used in the discussion below. When the Single Step input, $\overline{SS}$, is brought low, the internal processor responds by stopping during the fetch portion of the next instruction. This action is acknowledged by the processor raising the SYNC output. The address of the instruction to be fetched is then placed on the port pins. This state may be held indefinitely. To step to the next instruction, $\overline{SS}$ is raised high, which causes SYNC to go low, which is then used to return $\overline{SS}$ low. This allows the processor to advance to the next instruction. If $\overline{SS}$ is left high, the processor continues to execute at normal speed until $\overline{SS}$ goes low.

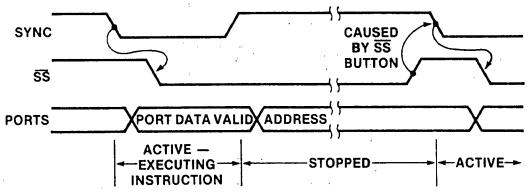


Figure 33. Single Step Timing

To preserve port functionality, port data is valid while SYNC is low. Figure 34 shows the external circuitry required to implement single step while preserving port functionality. S1 is the RUN/STOP switch. When in the RUN position, the 7474 is held preset so $\overline{SS}$ is high and the UPI executes normally. When switched to STOP, the preset is removed and the next low-going transition of SYNC causes the 7474 to clear, lowering $\overline{SS}$. While SYNC is low, the port data is valid and the current instruction is executing. Low SYNC is also used to enable the tri-state buffers when the ports are used as inputs. When execution is complete, SYNC goes high. This transition latches the valid port data in the 74LS374s. SYNC going high also signifies that the address of the next instruction will appear on the port pins. This state can be held indefinitely with the address data displayed on the LEDs.

When the S2 is depressed, the 7474 is set which causes $\overline{SS}$ to go high. This allows the processor to fetch and execute the instruction whose address was displayed. SYNC going low during execution, clears the 7474 lowering $\overline{SS}$. Thus the processor again stops when execution is complete and the next fetch is started.

All UPI functions continue to operate while single stepping (the processor is actuallly executing NOPs internally while stopped). Both IBF and timer/counter interrupts can be serviced. The only change is that the interval timer is prescaled on single stepped instructions and, of course, will not indicate the correct intervals in real time. The total number of instructions which would have been executed during a given interval is the same however.

The single step circuitry can be used to step through a complete program; however, this might be a time-consuming job if the program is long or if only a portion is to be examined. The circuitry could easily be modified to incorporate the output toggling technique to determine when to run and stop. If you would like to step thru a particular section of code, an extra port pin could replace switch S1. Extra instructions would then be added to lower the port when entering the code section and raise the port when exiting the section. The program would then stop when that section of code is reached allowing it to be stepped through. At the end of the section, the program would execute at normal speed.

CONCLUSION

Well, that's it. Machine readable (floppy disk or paper tape) source listings of UPI software for these applications are available in Insite, the Intel library of user-donated programs. Also available in Insite are the source listings for some of Intel's pre-programmed UPI products. These products are:

- 8278 Keyboard Display Controller
- 8295 Dot Matrix Printer Controller.

Other pre-programmed UPIs are the 8294 Data Encryption Unit and the 8292 GPIB (IEEE-488) Controller.

For information about Insite, write to:

Insite
Intel Corp.
3065 Bowers Ave.
Santa Clara, Ca 95051

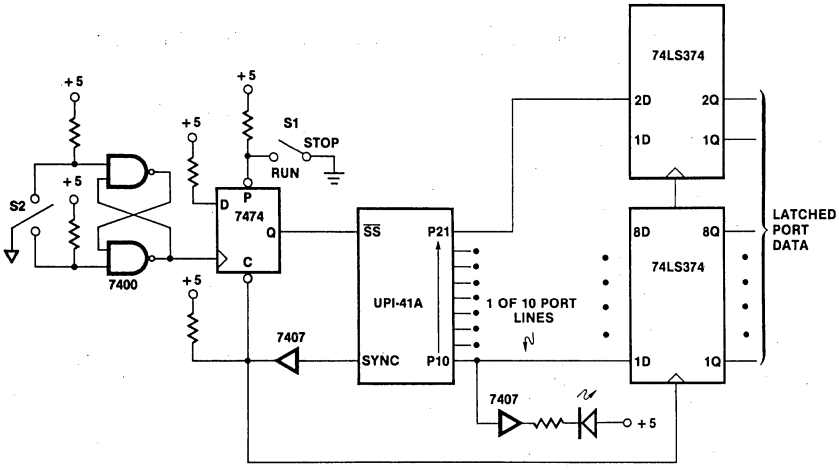


Figure 34. Single Step External Circuitry

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0


 LOC  OBJ        SEQ        SOURCE STATEMENT

                   1 ;        ********************************************
                   2 ;        *    UPI-41 8-DIGIT LED DISPLAY CONTROLLER    *
                   3 ;        ********************************************
                   4 ;
                   5 ;
                   6 ;
                   7 ;THIS PROGRAM USES THE UPI-41 AS A LED DISPLAY CONTROLLER
                   8 ;WHICH SCANS AND REFRESHES EIGHT SEVEN-SEGMENT LED DISPLAYS.
                   9 ;THE CHARACTERS ARE DEFINED BY INPUT FROM A MASTER CPU IN THE
                  10 ;FORM OF ONE EIGHT BIT WORD PER DIGIT-CHARACTER SELECTION.
                  11 ;
                  12 ;
                  13 ;
                  14 ;*****************************************************************
                  15 ;
                  16 ;REGISTER DEFINITIONS:
                  17 ;        REGISTER              RB1                    RB0
                  18 ;        --------              ---                    ---
                  19 ;        R0            DISPLAY MAP POINTER            NOT USED
                  20 ;        R1            NOT USED                       NOT USED
                  21 ;        R2            DATA WORD AND CHARACTER STORAGE NOT USED
                  22 ;        R3            DIGIT COUNTER                  NOT USED
                  23 ;        R4            NOT USED                       NOT USED
                  24 ;        R5            NOT USED                       NOT USED
                  25 ;        R6            NOT USED                       NOT USED
                  26 ;        R7            ACCUMULATOR STORAGE            NOT USED
                  27 ;*****************************************************************
                  28 ;
                  29 ;PORT PIN DEFINITIONS:
                  30 ;        PIN           PORT 1 FUNCTION        PORT 2 FUNCTION
                  31 ;        ---           ---------------        ---------------
                  32 ;        P0-7          SEGMENT DRIVER CONTROL DIGIT DRIVER CONTROL
                  33 ;
                  34 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 35 ;*****************************************************************
                 36 ;DISPLAY DATA WORD BIT DEFINITION:
                 37 ;       BIT            FUNCTION
                 38 ;       ---            --------
                 39 ;       0-4            CHARACTER SELECT
                 40 ;       5-7            DIGIT SELECT
                 41 ;
                 42 ;CHARACTER SELECT:
                 43 ;              D4  D3  D2  D1  D0  CHARACTER
                 44 ;              0   0   0   0   0     0
                 45 ;              0   0   0   0   1     1
                 46 ;              0   0   0   1   0     2
                 47 ;              0   0   0   1   1     3
                 48 ;              0   0   1   0   0     4
                 49 ;              0   0   1   0   1     5
                 50 ;              0   0   1   1   0     6
                 51 ;              0   0   1   1   1     7
                 52 ;              0   1   0   0   0     8
                 53 ;              0   1   0   0   1     9
                 54 ;              0   1   0   1   0     A
                 55 ;              0   1   0   1   1     B
                 56 ;              0   1   1   0   0     C
                 57 ;              0   1   1   0   1     D
                 58 ;              0   1   1   1   0     E
                 59 ;              0   1   1   1   1     F
                 60 ;              1   0   0   0   0     .
                 61 ;              1   0   0   0   1     G
                 62 ;              1   0   0   1   0     H
                 63 ;              1   0   0   1   1     I
                 64 ;              1   0   1   0   0     J
                 65 ;              1   0   1   0   1     L
                 66 ;              1   0   1   1   0     N
                 67 ;              1   0   1   1   1     O
                 68 ;              1   1   0   0   0     P
                 69 ;              1   1   0   0   1     R
                 70 ;              1   1   0   1   0     T
                 71 ;              1   1   0   1   1     U
                 72 ;              1   1   1   0   0     Y
                 73 ;              1   1   1   0   1     -
                 74 ;              1   1   1   1   0     '
                 75 ;              1   1   1   1   1     "BLANK"
                 76
                 77 ;
                 78 ;DIGIT SELECT:
                 79 ;              D7  D6  D5   DIGIT NUMBER
                 80 ;              0   0   0        1
                 81 ;              0   0   1        2
                 82 ;              0   1   0        3
                 83 ;              0   1   1        4
                 84 ;              1   0   0        5
                 85 ;              1   0   1        6
                 86 ;              1   1   0        7
                 87 ;              1   1   1        8
                 88 ;*****************************************************************
                 89 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 90 ;*****************************************************************
                 91 ;                        EQUATES
                 92 ;THE FOLLOWING CODE DESIGNATES "TIME" AS A VARIABLE. THIS
                 93 ;ADJUSTS THE AMOUNT OF CYCLES THE TIMER COUNTS BEFORE
                 94 ;A TIMER INTERRUPT OCCURS AND REFRESHES THE DISPLAY. APPROXIMATELY
                 95 ;50 TIMES PER SECOND.
FFF1             96 TIME    EQU     -0FH    ;TIMER VALUE 2.5MSEC
                 97 ;*****************************************************************
                 98 ;                        INTERRUPT BRANCHING
                 99 ;THIS PORTION OF MEMORY IS DEDICATED FOR USE OF RESET AND
                100 ;INTERRUPT BRANCHING. WHEN THE INTERRUPTS ARE ENABLED THE
                101 ;CODE AT THE FOLLOWING DESIGNATED SPOTS ARE EXECUTED WHEN A
                102 ;RESET OR A INTERRUPT OCCURS.
0000            103        ORG     0               ;
0000 0409       104        JMP     START           ;RESET
0002 00         105        NOP                     ;
0003 0438       106        JMP     INPUT           ;IBF INTERRUPT
0005 00         107        NOP                     ;
0006 00         108        NOP                     ;
0007 041F       109        JMP     DISPLA          ;TIMER INTERRUPT
                110 ;*****************************************************************
                111 ;                        INITIALIZATION
                112 ;THE FOLLOWING CODE SETS UP THE UPI-41 AND DISPLAY HARDWARE
                113 ;INTO OPERATIONAL FORMAT.  THE DISPLAY IS TURNED OFF, THE DISPLAY
                114 ;MAP IS FILLED WITH "BLANK" CHARACTERS, THE TIMER SET AND THE
                115 ;INTERRUPTS ARE ENABLED.
                116 ;
0009 D5         117 START:  SEL     RB1             ;
000A 8A08       118        ORL     P2,#08H         ;TURN DIGIT DRIVERS OFF
000C B838       119        MOV     R0,#38H         ;DISPLAY MAP POINTER,BOTTOM OF DISPLAY MAP
000E 23FF       120 BLKMAP: MOV    A,#0FFH         ;FF="BLANK"
0010 A0         121        MOV     @R0,A           ;BLANK TO DISPLAY MAP
0011 18         122        INC     R0              ;INCREMENT DISPLAY MAP POINTER
0012 F8         123        MOV     A,R0            ;DISPLAY MAP POINTER TO ACCUMULATOR
0013 B20E       124        JB5     BLKMAP          ;BLANK DISPLAY MAP TILL FILLED
0015 BB00       125        MOV     R3,#00H         ;SET DIGIT COUNTER TO 0
0017 23F1       126        MOV     A,#TIME         ;TIMER VALUE
0019 62         127        MOV     T,A             ;LOAD TIMER
001A 55         128        STRT    T               ;START TIMER
001B 25         129        EN      TCNTI           ;ENABLE TIMER INTERRUPT
001C 05         130        EN      I               ;ENABLE IBF INTERRUPT
                131 ;*****************************************************************
                132 ;                        USER PROGRAM
                133 ;A USERS PROGRAM WOULD INITIALIZE AT THIS POINT.  THE FOLLOWING
                134 ;CODE IS USED TO TAKE THE PLACE OF A POSSIBLE USER PROGRAM.
                135 ;
                136 ;
001D 041D       137 LOOP:   JMP     LOOP            ;WAIT FOR INTERRUPT
                138 ;*****************************************************************
                139 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                140 ;*****************************************************************
                141 ;                          DISPLAY ROUTINE
                142 ; THIS PORTION OF THIS PROGRAM IS AN INTERRUPT ROUTINE WHICH IS
                143 ;ACTED UPON WHEN THE TIMER COUNT IS COMPLETED. THE ROUTINE UPDATES
                144 ;ONE DISPLAY DIGIT FROM THE DISPLAY MAP PER INTERRUPT SEQUENTIALLY,
                145 ;THUS EIGHT TIMER INTERRUPTS WILL HAVE REFRESHED THE ENTIRE DISPLAY.
                146 ;REGISTER BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON
                147 ;ENTERING THE ROUTINE. ONCE THE DISPLAY HAS BEEN REFRESHED THE TIMER
                148 ;IS RESET AND THE ACCUMULATOR AND PRE-INTERRUPT REGISTER BANK IS RESTORED.
                149 ;
001F D5         150 DISPLA: SEL    RB1          ;REGISTER BANK 1
0020 AF         151         MOV    R7,A         ;SAVE ACCUMULATOR
0021 8A08       152         ORL    P2,#08H      ;TURN DIGIT DRIVERS OFF
0023 FB         153         MOV    A,R3         ;DIGIT COUNTER TO ACCUMULATOR
0024 4338       154         ORL    A,#38H       ;"OR" TO GET DISPLAY MAP ADDRESS
0026 A8         155         MOV    R0,A         ;DISPLAY MAP POINTER
0027 F0         156         MOV    A,@R0        ;GET CHARACTER FROM DISPLAY MAP
0028 39         157         OUTL   P1,A         ;OUTPUT CHARACTER TO SEGMENT DRIVERS
0029 FB         158         MOV    A,R3         ;DIGIT COUNTER VALUE TO ACCUMULATOR
002A 3A         159         OUTL   P2,A         ;OUTPUT TO DIGIT DRIVERS
002B 1B         160         INC    R3           ;INCREMENT DIGIT COUNTER
002C D307       161         XRL    A,#07H       ;CHECK IF AT LAST DIGIT
002E 9632       162         JNZ    SETIME       ;RESET TIMER IN NOT LAST DIGIT
0030 B800       163         MOV    R3,#00H      ;RESET DIGIT COUNTER
0032 23F1       164 SETIME: MOV    A,#TIME      ;TIMER VALUE
0034 62         165         MOV    T,A          ;LOAD TIMER
0035 55         166         STRT   T            ;START TIMER
0036 FF         167         MOV    A,R7         ;RESTORE ACCUMULATOR
0037 93         168         RETR                ;RETURN
                169 ;*****************************************************************
                170 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                171 ;
                172 ;*********************************************************************
                173 ;                INPUT CHARACTER AND DIGIT ROUTINE
                174 ; THIS PORTION OF THE PROGRAM IS AN INTERRUPT ROUTINE WHICH
                175 ;IS ACTED UPON WHEN THE IBF BIT IS SET. THE ROUTINE GETS THE
                176 ;DISPLAY DATA WORD FROM THE DBB AND DEFINES BOTH THE DIGIT AND
                177 ;THE CHARACTER TO BE DISPLAYED. THIS IS DONE BY MEANS OF A
                178 ;CHARACTER LOOP-UP TABLE AND A DISPLAY MAP FOR DIGIT AND CHARACTER
                179 ;LOCATION. SPECIAL CONSIDERATION IS TAKEN FOR A DECIMAL POINT WHICH IS
                180 ;SIMPLY ADDED TO THE EXISTING CHARACTER IN THE DISPLAY MAP. REGISTER
                181 ;BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON ENTERING
                182 ;THE ROUTINE. ONCE THE DATA WORD HAS BEEN FULLY DEFINED THE ACCUMULATOR
                183 ;AND THE PRE-INTERRUPT REGISTER BANK IS RESTORED.
                184 ;
0038 D5         185 INPUT:  SEL     RB1          ;REGISTER BANK 1
0039 AF         186         MOV     R7,A         ;SAVE ACCUMULATOR
003A 22         187         IN      A,DBB        ;GET DATA
003B AA         188         MOV     R2,A         ;SAVE DATA WORD
003C 47         189         SWAP    A            ;DEFINE DIGIT LOCATION
003D 77         190         RR      A            ;
003E 5307       191         ANL     A,#07H       ;
0040 4338       192         ORL     A,#38H       ;
0042 A8         193         MOV     R0,A         ;DIGIT LOCATION IN DIGIT POINTER
0043 FA         194         MOV     A,R2         ;SAVED DATA WORD TO ACCUMULATOR
0044 531F       195         ANL     A,#1FH       ;DEFINE CHARACTER LOOK-UP-TABLE LOC.
0046 E3         196         MOVP3   A,@A         ;GET CHARACTER
0047 AA         197         MOV     R2,A         ;SAVE CHARACTER
0048 D37F       198         XRL     A,#7FH       ;IS CHARACTER DECIMAL POINT
004A C650       199         JZ      DPOINT       ;
004C FA         200         MOV     A,R2         ;SAVED CHARACTER TO ACCUMULATOR
004D A0         201         MOV     @R0,A        ;CHARACTER TO DISPLAY MAP
004E 0453       202         JMP     RETURN       ;
0050 FA         203 DPOINT: MOV     A,R2         ;SAVED CHARACTER TO ACCUMULATOR
0051 50         204         ANL     A,@R0        ;"AND" WITH OLD CHARACTER
0052 A0         205         MOV     @R0,A        ;BACK TO DISPLAY MAP
0053 FF         206 RETURN: MOV     A,R7         ;RESTORE ACCUMULATOR
0054 93         207         RETR
                208 ;*********************************************************************
                209 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                210 ;*****************************************************************
                211 ;                        LOOK-UP TABLE
                212 ; THIS LOOK-UP TABLE ORIGINATES IN PAGE 3 OF THE UPI-41 PROGRAM
                213 ;MEMORY.  IT IS USED TO DEFINE THE CORRECT LEVEL OF EACH SEGMENT
                214 ;AND DECIMAL POINT FOR A SELECTED CHARACTER FROM THE INPUT ROUTINE.
                215 ;INVERSE LOGIC IS USED BECAUSE OF THE SPECIFIC DRIVER CIRCUITRY, THUS
                216 ;A 1 ON A GIVEN SEGMENT MEANS IT IS OFF AND A 0 MEANS IT IS ON.
                217 ;
                218                                ;*******SEGMENTS********
0300            219         ORG     300H           ;DP  G  F  E  D  C  B  A
0300 C0         220 CH0:    DB      0C0H           ;1   1  0  0  0  0  0  0
0301 F9         221 CH1:    DB      0F9H           ;1   1  1  1  1  0  0  1
0302 A4         222 CH2:    DB      0A4H           ;1   0  1  0  0  1  0  0
0303 B0         223 CH3:    DB      0B0H           ;1   0  1  1  0  0  0  0
0304 99         224 CH4:    DB      99H            ;1   0  0  1  1  0  0  1
0305 92         225 CH5:    DB      92H            ;1   0  0  1  0  0  1  0
0306 82         226 CH6:    DB      82H            ;1   0  0  0  0  0  1  0
0307 F8         227 CH7:    DB      0F8H           ;1   1  1  1  1  0  0  0
0308 80         228 CH8:    DB      80H            ;1   0  0  0  0  0  0  0
0309 98         229 CH9:    DB      98H            ;1   0  0  1  1  0  0  0
030A 88         230 CHA:    DB      88H            ;1   0  0  0  1  0  0  0
030B 83         231 CHB:    DB      83H            ;1   0  0  0  0  0  1  1
030C C6         232 CHC:    DB      0C6H           ;1   1  0  0  0  1  1  0
030D A1         233 CHD:    DB      0A1H           ;1   0  1  0  0  0  0  1
030E 86         234 CHE:    DB      86H            ;1   0  0  0  0  1  1  0
030F 8E         235 CHF:    DB      8EH            ;1   0  0  0  1  1  1  0
0310 7F         236 CHDP:   DB      7FH            ;0   1  1  1  1  1  1  1
0311 C2         237 CHG:    DB      0C2H           ;1   1  0  0  0  0  1  0
0312 89         238 CHH:    DB      89H            ;1   0  0  0  1  0  0  1
0313 FB         239 CHI:    DB      0FBH           ;1   1  1  1  1  0  1  1
0314 E1         240 CHJ:    DB      0E1H           ;1   1  1  0  0  0  0  1
0315 C7         241 CHL:    DB      0C7H           ;1   1  0  0  0  1  1  1
0316 AB         242 CHN:    DB      0ABH           ;1   0  1  0  1  0  1  1
0317 A3         243 CHO:    DB      0A3H           ;1   0  1  0  0  0  1  1
0318 8C         244 CHP:    DB      8CH            ;1   0  0  0  1  1  0  0
0319 AF         245 CHR:    DB      0AFH           ;1   0  1  0  1  1  1  1
031A 87         246 CHT:    DB      87H            ;1   0  0  0  0  1  1  1
031B C1         247 CHU:    DB      0C1H           ;1   1  0  0  0  0  0  1
031C 91         248 CHY:    DB      91H            ;1   0  0  1  0  0  0  1
031D BF         249 CHDASH: DB      0BFH           ;1   0  1  1  1  1  1  1
031E FD         250 CHAPOS: DB      0FDH           ;1   1  1  1  1  1  0  1
031F FF         251 BLANK:  DB      0FFH           ;1   1  1  1  1  1  1  1
                252 ;*****************************************************************
                253         END

USER SYMBOLS
BLANK  031F   BLKMAP 000E   CH0    0300   CH1    0301   CH2    0302   CH3    0303   CH4    0304   CH5    0305
CH6    0306   CH7    0307   CH8    0308   CH9    0309   CHA    030A   CHAPOS 031E   CHB    030B   CHC    030C
CHD    030D   CHDASH 031D   CHDP   0310   CHE    030E   CHF    030F   CHG    0311   CHH    0312   CHI    0313
CHJ    0314   CHL    0315   CHN    0316   CHO    0317   CHP    0318   CHR    0319   CHT    031A   CHU    031B
CHY    031C   DISPLA 001F   DPOINT 0050   INPUT  0038   LOOP   001D   RETURN 0053   SETIME 0032   START  0009
TIME   FFF1

ASSEMBLY COMPLETE,  NO ERRORS
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 1 ;
                 2 ;8085A SUBROUTINE TO DISPLAY THE 8-DIGIT BUFFER STARTING
                 3 ;AT THE LOCATION POINTED AT BY MSGSRT ON THE UPI-CONTROLLED
                 4 ;LED DISPLAY.
                 5 ;
                 6 ;INPUTS:MSGSRT - MESSAGE START LOCATION POINTER
                 7 ;DESTROYS: A, F/F'S
                 8 ;CALLS: OUTCHR
                 9 ;
4000            10          ORG     4000H
00E5            11 STATUS   EQU     0E5H            ;UPI STATUS PORT
0002            12 IBF      EQU     02H             ;UPI IBF FLAG MASK
00E4            13 DBBIN    EQU     0E4H            ;UPI DBBIN PORT
                14 ;
4000 E5         15 DSPLAY:  PUSH    H               ;SAVE HL
4001 C5         16          PUSH    B               ;SAVE BC
4002 2A2840     17          LHLD    MSGSRT          ;LOAD HL WITH MESSAGE START ADR
4005 0600       18          MVI     B,00H           ;INITIALIZE DIGIT COUNTER
4007 7E         19 S1:      MOV     A,M             ;GET CHR FROM BUFFER
4008 E61F       20          ANI     1FH             ;MAKE IT 5 BITS
400A 80         21          ADD     B               ;ADD IN DIGIT COUNTER
400B 4F         22          MOV     C,A             ;SAVE TOTAL IN C
400C CD1D40     23          CALL    OUTCHR          ;OUTPUT CHR PLUS LOCATION TO UPI
400F 78         24          MOV     A,B             ;GET DIGIT COUNTER
4010 C620       25          ADI     20H             ;INC FOR NEXT DIGIT
4012 DA1A40     26          JC      EXIT            ;DONE IF CARRY SET
4015 47         27          MOV     B,A             ;RESTORE DIGIT COUNTER
4016 23         28          INX     H               ;INC MESSAGE POINTER
4017 C30740     29          JMP     S1              ;GO GET NEXT CHR
                30 ;
401A C1         31 EXIT:    POP     B               ;RESTORE BC
401B E1         32          POP     H               ;RESTORE HL
401C C9         33          RET                     ;RETURN
                34 ;
                35 ;SUBROUTINE TO OUTPUT CHR TO UPI
                36 ;
401D DBE5       37 OUTCHR:  IN      STATUS          ;READ UPI STATUS
401F E602       38          ANI     IBF             ;LOOK AT IBF
4021 C21D40     39          JNZ     OUTCHR          ;WAIT UNTIL IBF=0
4024 79         40          MOV     A,C             ;GET CHR
4025 D3E4       41          OUT     DBBIN           ;OUTPUT CHR TO UPI DBBIN
4027 C9         42          RET                     ;RETURN
                43 ;
0002            44 MSGSRT:  DS      02H             ;LOCATION OF MESSAGE START POINTER
                45 ;
                46 END
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0


 LOC  OBJ        SEQ        SOURCE STATEMENT

                  1 ;        ******************************************
                  2 ;        *      UPI-41A SENSOR MATRIX CONTROLLER     *
                  3 ;        ******************************************
                  4 ;
                  5 ;       THIS PROGRAM USES THE UPI-41A AS A SENSOR MATRIX CONTROLLER.
                  6 ;IT HAS MONITORING CAPABILITIES OF UP TO 128 SENSORS.  THE COORDINATE
                  7 ;AND SENSOR STATUS OF EACH DETECTED CHANGE IS AVAILABLE TO THE MASTER
                  8 ;MICROPROCESSOR IN A SINGLE BYTE.  A 40X8 FIFO QUEUE IS PROVIDED FOR
                  9 ;DATA BUFFERING.  BOTH HARDWARE OR POLLED INTERRUPT METHODS CAN BE USED
                 10 ;TO NOTIFY THE MASTER OF A DETECTED SENSOR CHANGE.
                 11 ;
                 12 ;*****************************************************************
                 13 ;
                 14 ;REGISTER DEFINITIONS:
                 15 ;       REGISTER            RBQ                 RBI
                 16 ;       --------            ---                 ---
                 17 ;         R0          MATRIX MAP POINTER       NOT USED
                 18 ;         R1          FIFO POINTER             NOT USED
                 19 ;         R2          SCAN ROW SELECT          NOT USED
                 20 ;         R3          COLUMN COUNTER           NOT USED
                 21 ;         R4          FIFO-IN                  NOT USED
                 22 ;         R5          FIFO-OUT                 NOT USED
                 23 ;         R6          CHANGE WORD              NOT USED
                 24 ;         R7          COMPARE                  NOT USED
                 25 ;
                 26 ;*****************************************************************
                 27 ;
                 28 ;PORT PIN DEFINITIONS:
                 29 ;
                 30 ;PIN       PORT 1 FUNCTION      PIN       PORT 2 FUNCTION
                 31 ;---       ---------------      ---       ---------------
                 32 ;P0-7      COLUMN LINE INPUTS   P0-3      ROW SELECT OUTPUTS
                 33 ;                               P4        FIFO NOT EMPTY INTERRUPT
                 34 ;                               P5        OBF INTERRUPT
                 35 ;                               P6-7      NOT USED
                 36 ;
                 37 ;*****************************************************************
                 38 ;
                 39 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 40 ;*****************************************************************
                 41 ;
                 42 ;CHANGE WORD BIT DEFINITION:
                 43 ;
                 44 ;                BIT                 FUNCTION
                 45 ;                ---                 --------
                 46 ;                D0-6                SENSOR COORDINATE
                 47 ;                D7                  SENSOR STATUS
                 48 ;
                 49 ;*****************************************************************
                 50 ;
                 51 ;STATUS REGISTER BIT DEFINITION:
                 52 ;
                 53 ;                BIT                 FUNCTION
                 54 ;                ---                 --------
                 55 ;                D0                  OBF
                 56 ;                D1-3                IBF, F0, F1 (NOT USED)
                 57 ;                D4                  FIFO NOT EMPTY
                 58 ;                D5-7                USED DEFINED (NOT USED)
                 59 ;
                 60 ;*****************************************************************
                 61 ;
                 62 ;                        EQUATES
                 63 ;
                 64 ;THE FOLLOWING CODE DESIGNATES THREE VARIABLES; SCANTM,FIFOBA
                 65 ;AND FIFOTA.  SCANTM ADJUSTS THE LENGTH OF A DELAY BETWEEN
                 66 ;SCANNING SWITCH.  THIS SIMULATES DEBOUNCE FUNCTIONS.  FIFOBA
                 67 ;IS THE BOTTOM ADDRESS OF THE FIFO.  FIFOTA IS THE TOP ADDRESS
                 68 ;OF THE FIFO.  THIS MAKES IT POSSIBLE TO HAVE A FIFO 3 TO 40
                 69 ;BYTES IN LENGTH.
                 70 ;
                 71 ;*****************************************************************
                 72 ;
000F             73 SCANTM  EQU     0FH            ;SCAN TIME ADJUST
0008             74 FIFOBA  EQU     08H            ;FIFO BOTTOM ADDRESS
002F             75 FIFOTA  EQU     2FH            ;FIFO TOP ADDRESS
                 76 ;
                 77 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 78 ;*****************************************************************
                 79 ;
                 80 ;                          INITIALIZATION
                 81 ;
                 82 ;THE PROGRAM STARTS AT THE FOLLOWING CODE UPON RESET.  WITHIN
                 83 ;THIS INITIALIZATION SECTION THE REGISTERS THAT MAINTAIN THE MATRIX
                 84 ;MAP,FIFO AND ROW SCANNING ARE SET UP.  PORT 1 IS SET HIGH FOR USE
                 85 ;AS AN INPUT PORT FOR THE COLUMN STATUS.  BIT 4 OF STATUS REGISTER IS
                 86 ;WRITTEN TO CONVEY A FIFO EMPTY CONDITION.  THE INITIAL COLUMN STATUS
                 87 ;OF ALL THE ROWS IN THE SENSOR MATRIX IS THEN READ INTO THE MATRIX
                 88 ;MAP.  ONCE THE MATRIX MAP IS FILLED THE OBF INTERRUPT (PORT 2-4) IS
                 89 ;ENABLED.
                 90 ;
                 91 ;*****************************************************************
                 92 ;
0000             93          ORG     0
0000 B83F        94 INITMX:  MOV     R0,#3FH        ;MATRIX MAP POINTER REGISTER,TOP ADDRESS
0002 BA0F        95          MOV     R2,#0FH        ;SCAN ROW SELECT REGISTER,TOP ROW
0004 BC08        96          MOV     R4,#FIFOBA     ;FIFO INPUT ADDRESS REGISTER,BOTTOM OF FIFO
0006 BD2F        97          MOV     R5,#FIFOTA     ;FIFO OUTPUT ADDRESS REGISTER,TOP OF FIFO
0008 89FF        98          ORL     P1,#0FFH       ;INITIALIZE PORT 1 HIGH FOR INPUTS
000A 2300        99          MOV     A,#00H         ;INITIALIZE STATUS REGISTER, FIFO EMPTY
000C 90         100          MOV     STS,A          ;WRITE TO STATUS REGISTER, BITS 4-7
000D FA         101 FILLMX:  MOV     A,R2           ;SCAN ROW SELECT TO ACCUMULATOR
000E 3A         102          OUTL    P2,A           ;OUTPUT SCAN ROW SELECT TO PORT 2
000F 09         103          IN      A,P1           ;INPUT COLUMN STATUS PORT 1
0010 A0         104          MOV     @R0,A          ;LOAD MATRIX MAP WITH COLUMN STATUS
0011 FA         105          MOV     A,R2           ;CHECK SCAN ROW SELECT REGISTER VALUE FOR 0
0012 C618       106          JZ      OBFINT         ;IF 0 ENABLE OBF INTERRUPT
0014 C8         107          DEC     R0             ;DECREMENT TO NEXT MATRIX MAP ADDRESS
0015 CA         108          DEC     R2             ;DECREMENT TO SCAN NEXT ROW
0016 040D       109          JMP     FILLMX         ;FILL NEXT MATRIX MAP ADDRESS
0018 BA10       110 OBFINT:  MOV     R2,#10H        ;BIT 4 HIGH IN ROW SCAN SELECT REGISTER
001A FA         111          MOV     A,R2           ;ROW SCAN SELECT VALUE TO ACCUMULATOR
001B 3A         112          OUTL    P2,A           ;INITIALIZE PORT 2, BIT 4 FOR "EN FLAGS"
001C F5         113          EN      FLAGS          ;ENABLE OBF INTERRUPT PORT 2, BIT 4
                114 ;
                115 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                116 ;*****************************************************************
                117 ;
                118 ;                        SCAN AND COMPARE
                119 ;
                120 ;THE FOLLOWING CODE IS THE SCAN AND COMPARE SECTION OF THE PROGRAM.
                121 ;UPON ENTERING THIS SECTION A CHECK IS MADE TO SEE IF THE ENTIRE MATRIX
                122 ;HAS BEEN SCANNED.  IF SO THE REGISTERS THAT MAINTAIN THE MATRIX MAP AND ROW
                123 ;SCANNING ARE RESET TO THE BEGINNING OF THE SENSOR MATRIX.  IF THE ENTIRE
                124 ;MATRIX HASNT BEEN SCANNED THE REGISTERS INCREMENT TO SCAN THE NEXT ROW.
                125 ;FROM THIS POINT ON THE ROW SCAN SELECT REGISTER IS USED FOR TWO FUNCTIONS.
                126 ;BITS 0-3 FOR SCANNING AND BITS 4 AND 5 FOR THE EXTERNAL INTERRUPTS.  THUSLY
                127 ;ALL USAGE OF THE REGISTERS IS DONE BY LOGICALLY MASKING IT SO AS TO ONLY
                128 ;AFFECT THE FUNCTION DESIRED.  ONCE THE REGISTERS ARE RESET, ONE ROW OF THE
                129 ;SENSOR MATRIX IS SCANNED.  A DELAY IS EXECUTED TO ADJUST FOR SCAN TIME
                130 ;(DEBOUNCE).  A BYTE OF COLUMN STATUS IS THEN READ INTO THE MATRIX MAP.
                131 ;AT THE TIME THE NEW COLUMN STATUS IS COMPARED TO THE OLD.  THE RESULT IS
                132 ;STORED IN THE COMPARE REGISTER.  THE PROGRAM IS THEN ROUTED ACCORDING TO
                133 ;WHETHER OR NOT A CHANGE WAS DETECTED.
                134 ;
                135 ;*****************************************************************
                136 ;
001D FA         137 ADJREG: MOV    A,R2          ;SCAN ROW SELECT TO ACCUMULATOR
001E 530F       138         ANL    A,#0FH        ;CHECK FOR 0 SCAN VALUE ONLY,NOT INTERRUPT
0020 C626       139         JZ     RSETRG        ;IF 0 RESET REGISTERS
0022 C8         140         DEC    R0            ;DECREMENT MATRIX MAP POINTER
0023 CA         141         DEC    R2            ;DECREMENT SCAN ROW SELECT
0024 042C       142         JMP    SCANMX        ;SCAN MATRIX
0026 B83F       143 RSETRG: MOV    R0,#3FH       ;RESET MATRIX MAP POINTER REGISTER,TOP ADDRESS
0028 FA         144         MOV    A,R2          ;SCAN ROW SELECT TO ACCUMULATOR
0029 430F       145         ORL    A,#0FH        ;RESET SCAN ROW SELECT,NO INTERRUPT CHANGE
002B AA         146         MOV    R2,A          ;SCAN ROW SELECT REGISTER
002C FA         147 SCANMX: MOV    A,R2          ;SCAN ROW SELECT TO ACCUMULATOR
002D 3A         148         OUTL   P2,A          ;OUTPUT SCAN ROW SELECT TO PORT 2
002E BB0F       149         MOV    R3,#SCANTM    ;SET DELAY FOR OUTPUT SCAN TIME
0030 EB30       150 DELAY2: DJNZ   R3,DELAY2     ;DELAY
0032 09         151         IN     A,P1          ;INPUT COLUMN STATUS FROM PORT 1 TO ACCUMULATOR
0033 20         152         XCH    A,@R0         ;STORE NEW COLUMN STATUS SAVE OLD IN ACCUMULATOR
0034 D0         153         XRL    A,@R0         ;COMPARE OLD WITH NEW COLUMN STATUS
0035 AF         154         MOV    R7,A          ;SAVE COMPARE RESULT IN COMPARE REGISTER
0036 C669       155         JZ     CHFFUL        ;IF THE SAME, CHECK IF FIFO IS FULL
                156 ;
                157 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                158 ;*****************************************************************
                159 ;
                160 ;                        CHANGE WORD ENCODING
                161 ;
                162 ;THE FOLLOWING CODE IS THE CHANGE WORD ENCODING SECTION.  THIS
                163 ;SECTION IS ONLY EXECUTED IF A CHANGE WAS DETECTED.  THE COLUMN COUNTER
                164 ;IS SET AND DECREMENTED TO DESIGNATE EACH OF THE 8 COLUMNS.  THE COMPARE
                165 ;REGISTER IS LOOKED AT ONE BIT AT A TIME TO FIND THE EXACT LOCATION OF
                166 ;THE CHANGE(S).  WHEN A CHANGE IS FOUND IT IS ENCODED BY GIVING IT A
                167 ;COORDINATE FOR ITS LOCATION.  THIS IS DONE BY COMBINING THE PRESENT VALUE
                168 ;IN THE ROW SCAN SELECT REGISTER AND THE COLUMN COUNTER.  THE ACTUAL STATUS
                169 ;OF THAT SENSOR IS ESTABLISHED BY LOOKING AT THE CORRESPONDING BYTE IN
                170 ;THE MATRIX MAP.  THIS STATUS IS COMBINED WITH THE COORDINATE TO ESTABLISH
                171 ;THE CHANGE WORD.  THE CHANGE WORD IS THEN STORED IN THE CHANGE WORD REGISTER.
                172 ;
                173 ;*****************************************************************
                174 ;
0038 B808       175          MOV    R3,#08H        ;SET COLUMN COUNTER REGISTER TO 8
003A CB         176 RRLOOK:  DEC    R3             ;DECREMENT COLUMN COUNTER
003B F0         177          MOV    A,@R0          ;COLUMN STATUS TO ACCUMULATOR
003C 77         178          RR     A              ;ROTATE COLUMN STATUS RIGHT
003D A0         179          MOV    @R0,A          ;ROTATED COLUMN STATUS BACK TO MATRIX MAP
003E FF         180          MOV    A,R7           ;COMPARE REGISTER VALUE TO ACCUMULATOR
003F 77         181          RR     A              ;ROTATE COMPARE VALUE RIGHT
0040 AF         182          MOV    R7,A           ;ROTATED COMPARE VALUE TO COMPARE REGISTER
0041 F245       183          JB7    ENCODE         ;TEST BIT 7 IF CHANGE DETECTED ENCODE CHANGE WORD
0043 0469       184          JMP    CHFFUL         ;IF NO CHANGE IS DETECTED CHECK FOR FIFO FULL
0045 FA         185 ENCODE:  MOV    A,R2           ;SCAN ROW SELECT TO ACCUMULATOR  0000XXXX
0046 530F       186          ANL    A,#0FH         ;ROTATE ONLY SCAN VALUE
0048 E7         187          RL     A              ;ROTATE LEFT                     000XXXX0
0049 E7         188          RL     A              ;ROTATE LEFT                     00XXXX00
004A E7         189          RL     A              ;ROTATE LEFT                     0XXXX000
004B 4B         190          ORL    A,R3           ;ESTABLISH MATRIX COORDINANT     0XXXXXXX
                191                                ;(OR) COLUMN COUNTER VALUE WITH ACCUMULATOR
004C AE         192          MOV    R6,A           ;SAVE COORDINANT IN CHANGE WORD REGISTER
004D F0         193          MOV    A,@R0          ;COLUMN STATUS FROM MATRIX MAP TO ACCUMULATOR
004E 5380       194          ANL    A,#80H         ;0 ALL BITS BUT BIT 7
0050 4E         195          ORL    A,R6           ;(OR) SENSOR STATUS WITH COORDINATE FOR COMPLETED CHANGE WORD
0051 AE         196          MOV    R6,A           ;SAVE CHANGE WORD                XXXXXXXX
                197 ;
                198 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                199 ;*****************************************************************
                200 ;
                201 ;                    FIFO-DBBOUT MANAGEMENT
                202 ;
                203 ;THE FOLLOWING CODE IS THE FIFO-DBBOUT MANAGEMENT SECTION OF THE
                204 ;PROGRAM.  THIS SECTION TAKES AN ENCODED CHANGE WORD AND LOADS IT INTO
                205 ;THE FIFO.  THE FIFO NOT EMPTY INTERRUPT IS THEN SET AND THE FIFO-IN
                206 ;POINTER GETS UPDATED. A FIFO FULL CONDITION IS THEN CHECKED FOR AND
                207 ;ROUTED ACCORDINGLY.  IF BOTH THE FIFO AND OBF HAVE CHANGE WORDS THE
                208 ;PROGRAM LOCKS UP UNTIL THIS HAS CHANGED.  IF THE FIFO ISNT FULL COLUMN
                209 ;COUNTER= 0, FIFO EMPTY AND OBF CONDITIONS ARE CHECKED.  THE FIFO-OUT
                210 ;POINTER IS SET AND DBBOUT IS LOADED IF THE FIFO ISNT EMPTY AND OBF ISNT
                211 ;SET.  IF THIS ISNT THE SITUATION, PROGRAM FLOW IS ROUTED BACK TO THE
                212 ;THE SCAN AND COMPARE SECTION TO SCAN THE NEXT ROW.
                213 ;
                214 ;*****************************************************************
                215 ;
0052 FC         216 LOADFF: MOV    A,R4          ;FIFO INPUT ADDRESS TO ACCUMULATOR
0053 A9         217         MOV    R1,A          ;FIFO POINTER USED FOR INPUT
0054 FE         218         MOV    A,R6          ;CHANGE WORD TO ACCUMULATOR
0055 A1         219         MOV    @R1,A         ;LOAD FIFO AT FIFO INPUT ADDRESS
0056 2310       220 STATNE: MOV    A,#10H        ;BIT 4 FOR FIFO NOT EMPTY
0058 90         221         MOV    STS,A         ;WRITE TO STATUS REGISTER, FIFO NOT EMPTY
0059 8A20       222 INTRH1: ORL    P2,#20H       ;FIFO NOT EMPTY INTERRUPT PORT 2-5 HIGH
005B FA         223         MOV    A,R2          ;ROW SCAN SELECT TO ACCUMULATOR
005C 4320       224         ORL    A,#20H        ;SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
005E AA         225         MOV    R2,A          ;ROW SCAN SELECT REGISTER
005F 232F       226 ADJFIN: MOV    A,#FIFOTA     ;FIFO TOP ADDRESS TO ACCUMULATOR
0061 DC         227         XRL    A,R4          ;COMPARE WITH CURRENT FIFO INPUT ADDRESS
0062 C667       228         JZ     RSFFIN        ;IF THE SAME RESET FIFO INPUT REGISTER
0064 1C         229         INC    R4            ;NEXT FIFO INPUT ADDRESS
0065 0469       230         JMP    CHFFUL        ;CHECK FIFO FULL
0067 BC08       231 RSFFIN: MOV    R4,#FIFOBA    ;RESET FIFO INPUT REGISTER,BOTTOM OF FIFO
0069 FC         232 CHFFUL: MOV    A,R4          ;FIFO INPUT ADDRESS TO ACCUMULATOR
006A DD         233         XRL    A,R5          ;COMPARE INPUT WITH OUTPUT FIFO ADDRESS
006B 967D       234         JNZ    CHCNTR        ;IF NOT SAME CHECK COLUMN COUNTER VALUE
006D 866D       235 CHOBF1: JOBF   CHOBF1        ;IF OBF IS 1 THEN CHECK OBF
006F 232F       236 ADJFOT: MOV    A,#FIFOTA     ;FIFO TOP ADDRESS TO ACCUMULATOR
0071 DD         237         XRL    A,R5          ;COMPARE TOP TO OUTPUT FIFO ADDRESS
0072 C677       238         JZ     RSFFOT        ;IF THE SAME RESET FIFO OUTPUT REGISTER
0074 1D         239         INC    R5            ;NEXT FIFO OUTPUT ADDRESS
0075 0479       240         JMP    LOADDB        ;LOAD DBBOUT
0077 BD08       241 RSFFOT: MOV    R5,#FIFOBA    ;RESET FIFO OUTPUT ADDRESS TO BOTTOM OF FIFO
0079 FD         242 LOADDB: MOV    A,R5          ;OUTPUT FIFO ADDRESS TO ACCUMULATOR
007A A9         243         MOV    R1,A          ;FIFO POINTER USED FOR OUTPUT
007B F1         244         MOV    A,@R1         ;CHANGE WORD TO ACCUMULATOR
007C 02         245         OUT    DBB,A         ;CHANGE WORD TO DBBOUT
007D FB         246 CHCNTR: MOV    A, R3         ;COLUMN COUNTER TO ACCUMULATOR
007E 963A       247         JNZ    RRLOOK        ;IF NOT 0 FINISH CHANGE WORD ENCODING
0080 2308       248 CHFFEM: MOV    A,#FIFOBA     ;FIFO BOTTOM ADDRESS TO ACCUMULATOR
0082 DC         249         XRL    A,R4          ;COMPARE FIFO INPUT ADDRESS WITH FIFO BOTTOM ADDRESS
0083 C68C       250         JZ     ADJFEM        ;IF THE SAME,ADJUST TO CHECK FOR FIFO EMPTY
0085 FC         251         MOV    A,R4          ;FIFO INPUT ADDRESS TO ACCUMULATOR
0086 07         252         DEC    A             ;DECREMENT FIFO INPUT ADDRESS IN ACCUMULATOR
0087 DD         253         XRL    A,R5          ;COMPARE INPUT TO OUTPUT FIFO ADDRESSES
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0088 C691        254        JZ      STATMT        ;IF SAME,WRITE STATUS REGISTER FOR FIFO EMPTY
008A 049C        255        JMP     CHOBF2        ;CHECK OBF
008C 232F        256 ADJFEM: MOV    A,#FIFOTA     ;FIFO TOP ADDRESS TO ACCUMULATOR
008E D0          257        XRL     A,R5          ;COMPARE TOP TO OUTPUT FIFO ADDRESS
008F 969C        258        JNZ     CHOBF2        ;IF NOT SAME THEN FIFO IS NOT EMPTY, CHECK OBF
0091 2300        259 STATMT: MOV    A,#00H        ;CLEAR BIT 0 FOR FIFO EMPTY
0093 90          260        MOV     STS,A         ;WRITE TO STATUS REGISTER
0094 9ADF        261 INTRLO: ANL    P2,#0DFH      ;FIFO EMPTY, INTERRUPT PORT 2-5 LOW
0096 FA          262        MOV     A,R2          ;SCAN ROW SELECT TO ACCUMULATOR
0097 53DF        263        ANL     A,#0DFH       ;SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
0099 AA          264        MOV     R2,A          ;SCAN ROW SELECT REGISTER
009A 041D        265        JMP     ADJREG        ;ADJUST REGISTERS
009C 861D        266 CHOBF2: JOBF   ADJREG        ;IF OBF=1 THEN ADJUST REGISTERS
009E 046F        267        JMP     ADJFOT        ;ADJUST FIFO OUT ADDRESS TO LOAD DBBOUT
                 268 ;
                 269        END

USER SYMBOLS
ADJFEM 008C   ADJFIN 005F   ADJFOT 006F   ADJREG 001D   CHCNTR 007D   CHFFEM 0080   CHFFUL 0069   CHOBF1 006D
CHOBF2 009C   DELAY2 0030   ENCODE 0045   FIFOBA 0008   FIFOTA 002F   FILLMX 000D   INITMX 0000   INTRH1 0059
INTRLO 0094   LOADDB 0079   LOADFF 0052   OBFINT 0018   RRLOOK 003A   RSETRG 0026   RSFFIN 0067   RSFFOT 0077
SCANMX 002C   SCANTM 000F   STATMT 0091   STATNE 0056

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, X108        MODULE   PAGE    1
8085A/UPI SENSOR MATRIX CONTROLLER

 LOC  OBJ         SEQ        SOURCE STATEMENT

                    1 ;
                    2 ;SUBROUTINE TO READ ALL CHANGES IN THE UPI AND BUILD A BUFFER
                    3 ;STARTING AT BUFSRT.  REG. B CONTAINS THE NUMBER OF CHANGES
                    4 ;UPON EXIT.  THE MAXIMUM NUMBER OF CHANGES IN ANY ONE CALL
                    5 ;IS 255.
                    6 ;
                    7 ;INPUTS: NOTHING
                    8 ;OUTPUTS: CHANGE WORD BUFFER AT BUFSRT
                    9 ;         CHANGE WORD COUNT IN REG. B
                   10 ;CALLS: NOTHING
                   11 ;
4000               12         ORG     4000H
00E5               13 STATUS  EQU     0E5H            ;UPI STATUS PORT
00E4               14 DBBOUT  EQU     0E4H            ;UPI DBBOUT PORT
0010               15 FIFO    EQU     10H             ;FIFO NOT EMPTY MASK
0001               16 OBF     EQU     01H             ;OBF MASK
4300               17 BUFSRT  EQU     4300H           ;BUFFER START LOCATION
                   18 ;
4000 210043        19 START:  LXI     H,BUFSRT        ;INITIALIZE BUFFER POINTER
4003 0600          20         MVI     B,00H           ;CLEAR CHANGE WORD COUNTER
4005 DBE5          21 POLL1:  IN      STATUS          ;READ UPI STATUS
4007 E611          22         ANI     FIFO OR OBF     ;TEST FIFO NOT EMPTY AND OBF
4009 C8            23         RZ                      ;RETURN IF ZERO
400A DBE5          24         IN      STATUS          ;READ UPI STATUS
400C E601          25         ANI     OBF             ;TEST OBF FLAG
400E CA0540        26         JZ      POLL1           ;WAIT IF NOT READY
4011 DBE4          27         IN      DBBOUT          ;READ CHANGE WORD
4013 77            28         MOV     M,A             ;LOAD BUFFER WITH CHANGE WORD
4014 23            29         INX     H               ;INC BUFFER POINTER
4015 04            30         INR     B               ;INC CHANGE WORD COUNTER
4016 C8            31         RZ                      ;EXIT IF COUNTER = 256
4017 C30540        32         JMP     POLL1           ;CHECK IF MORE CHANGE WORDS
                   33 ;
                   34 END
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0
AP-41 COMBINATION I/O DEVICE

 LOC  OBJ        SEQ        SOURCE STATEMENT

                   1 $MOD42
                   2 ;*************************UNIOD*****************************************
                   3 ;
                   4 ;THIS UPI-41 PROGRAM IMPLEMENTS A FULL-DUPLEX UART WITH ON-CHIP
                   5 ;BAUD RATE GENERATION IN COMBINATION WITH AN 8-BIT PARALLEL I/O
                   6 ;PORT.  THE BAUD RATE IS SELECTABLE FROM 110 TO 1200 BAUD.  THE
                   7 ;PARALLEL I/O PORT IS PROGRAMMABLE FOR EITHER INPUT OR OUTPUT.
                   8 ;
                   9 ;INTERRUPT OUTPUTS ARE AVAILABLE FOR DATA AVAILABLE ON THE RECEIVER
                  10 ;AND PARALLEL INPUT.  THE STATUS REGISTER MUST BE READ TO DETERMINE
                  11 ;WHICH SOURCE CAUSED THE INTERRUPT.  THE FLAGS F0 AND F1 CODE THE
                  12 ;INTERRUPT SOURCE.  F0 AND F1 ALSO GIVE AN INDICATION OF COMMAND
                  13 ;ERRORS.
                  14 ;
                  15 ;*******************************************************************
                  16 ;
                  17 ;REGISTER DEFINITION
                  18 ;              RB0                    RB1
                  19 ;              ---                    ---
                  20 ;       0      NOT USED               NOT USED
                  21 ;       1      NOT USED               BAUD RATE CONSTANT
                  22 ;       2      NOT USED               TX TICK COUNTER
                  23 ;       3      RX STATUS (RXSTS)      TX SERIALIZER
                  24 ;       4      RX HOLDING             TX BUFFER
                  25 ;       5      RX TICK COUNTER        TX STATUS (TXSTS)
                  26 ;       6      RX DESERIALIZER        COMMAND STORE
                  27 ;       7      STATUS REG STORE       ACC.  INTERRUPT SAVE
                  28 ;
                  29 ;*******************************************************************
                  30 ;
                  31 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 32 ;
                 33 ;*****************************************************************
                 34 ;
                 35 ;COMMANDS
                 36 ;
                 37 ;       CONFIGURE: 0 0 0 A B C D P
                 38 ;                                    A - 1200 BAUD SELECT
                 39 ;                                    B - 600 BAUD SELECT
                 40 ;                                    C - 300 BAUD SELECT
                 41 ;                                    D - 110 BAUD SELECT
                 42 ;                                    E - PARALLEL I/O DIRECTION
                 43 ;                                          0 - INPUT
                 44 ;                                          1 - OUTPUT
                 45 ;
                 46 ;       I/O:        1 0 0 0 0 0 0 0   (PERFORM I/O OPERATION)
                 47 ;       RESET ERROR:1 1 0 0 0 0 0 0    (RESET RX ERROR IN STATUS)
                 48 ;
                 49 ;*****************************************************************
                 50 ;
                 51 ;STATUS REGISTER DEFINITION
                 52 ;
                 53 ;       BIT            DEFINITION
                 54 ;       ---            ----------
                 55 ;        0             OBF - DATA AVAILABLE
                 56 ;        1             IBF - BUSY
                 57 ;        2             F0
                 58 ;        3             F1
                 59 ;        4             NOT USED
                 60 ;        5             TXINT - TX INTERRUPT
                 61 ;        6             FRAMING ERROR
                 62 ;        7             OVERRUN ERROR
                 63 ;
                 64 ;       F0      F1     OPERATION
                 65 ;       --      --     ---------
                 66 ;       0       0      X
                 67 ;       0       1      PARALLEL I/O DATA AVAILABLE
                 68 ;       1       0      SERIAL I/O DATA AVAILABLE
                 69 ;       1       1      COMMAND ERROR
                 70 ;
                 71 ;*****************************************************************
                 72 ;
                 73 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 74 ;
                 75 ;*****************************************************************
                 76 ;
                 77 ;STATUS REGISTER DEFINITIONS
                 78 ;
                 79 ;                        RXSTS                    TXSTS
                 80 ;                        -----                    -----
                 81 ;                0       RX FLAG - SPACE          TX FLAG - TRANSMITTING CHR
                 82 ;                1       START FLAG - GOOD START  REQUEST BYTE - CHR IN BUFFER
                 83 ;                2       BYTE FINISHED            TX PIPELINED DATA BIT
                 84 ;                3       DATA READY               START BIT FLAG
                 85 ;                4       FRAMING ERROR            NOT USED
                 86 ;                5       OVERRUN ERROR            NOT USED
                 87 ;                6       IO DIRECTION             NOT USED
                 88 ;                7       IO FLAG                  NOT USED
                 89 ;
                 90 ;*****************************************************************
                 91 ;
                 92 ;PORT 2 DEFINITIONS
                 93 ;
                 94 ;        BIT             DEFINITION
                 95 ;        ---             ----------
                 96 ;         0              TX DATA
                 97 ;         1              NOT USED
                 98 ;         2              NOT USED
                 99 ;         3              TX INTERRUPT
                100 ;         4              OBF INTERRUPT (RX OR I/O DATA AVAILABLE)
                101 ;         5              NOT USED
                102 ;         6              NOT USED (TICK SAMPLE)
                103 ;         7              NOT USED
                104 ;
                105 ;*****************************************************************
                106 ;
                107 ;MISC.
                108 ;
                109 ;        RX DATA         T0 INPUT
                110 ;        EXT CLOCK       T1 INPUT 76.8KHZ (1.2288MHZ/16)
                111 ;
                112 ;*****************************************************************
                113 ;
                114 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                115 ;
                116 ;*****************************************************************
                117 ;
                118 ;SYSTEM EQUATES:
                119 ;
0001            120 RXFLG   EQU     01H             ;RECEIVE FLAG IN RXSTS
0002            121 SRTFLG  EQU     02H             ;START BIT FLAG IN RXSTS
0004            122 BFFLG   EQU     04H             ;BYTE FINISHED FLAG IN RXSTS
0008            123 DATRDY  EQU     08H             ;DATA READY FLAG IN RXSTS
0010            124 FRAMER  EQU     10H             ;FRAMING ERROR FLAG IN RXSTS
0020            125 OVRUN   EQU     20H             ;OVERRUN ERROR FLAG IN RXSTS
0040            126 IODIR   EQU     40H             ;I/O DIRECTION FLAG IN RXSTS
0080            127 IOFLG   EQU     80H             ;I/O REQUEST FLAG IN RXSTS
0001            128 TXFLG   EQU     01H             ;TX FLAG IN TXSTS
0002            129 REQFLG  EQU     02H             ;REQUEST BYTE FLAG IN TXSTS
0040            130 TICOUT  EQU     40H             ;TICK SAMPLE BIT IN PORT 2
0080            131 RXINTL  EQU     80H             ;RX DESERIALIZER INITIALIZATION
0004            132 TICSRT  EQU     04H             ;TICK INITIALIZATION
007F            133 ASCMSK  EQU     7FH             ;ASCII MASK
0003            134 TXTIC   EQU     03H             ;TX TICK MOD MASK
0028            135 TXEND   EQU     40D             ;TICK COUNT AT END OF TX CHARACTER
0024            136 STPEND  EQU     36D             ;TICK COUNT AT END OF TX DATA
0004            137 MARK    EQU     04H             ;MARK OUTPUT
00FB            138 SPACE   EQU     0FBH            ;SPACE OUTPUT
0000            139 ZERO    EQU     00H             ;GENERAL CLEAR
0008            140 TXINT   EQU     08H             ;TX INTERRUPT OUTPUT IN PORT 2
0020            141 TXBIT   EQU     20H             ;TX INTERRUPT BIT IN STATUS
0020            142 TIMCON  EQU     32D             ;TIMER CONSTANT RAM LOCATION
003F            143 RSTERR  EQU     3FH             ;RESET ERROR MASK FOR STATUS
0040            144 FESTS   EQU     40H             ;FRAMING ERROR BIT IN STATUS
0080            145 OVSTS   EQU     80H             ;OVERRUN ERROR BIT IN STATUS
0001            146 MKOUT   EQU     01H             ;MARK OUTPUT TO PORT
00FE            147 SPOUT   EQU     0FEH            ;SPACE OUTPUT TO PORT
0008            148 SBIT    EQU     08H             ;TX START BIT FLAG
0003            149 RXSTS   EQU     R3              ;RX STATUS REGISTER
0005            150 TXSTS   EQU     R5              ;TX STATUS REGISTER
                151 ;
                152 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                153 ;*****************************************************************
                154 ;
                155 ;RESET VECTOR LOCATION
                156 ;
                157 ;*****************************************************************
                158 ;
0000            159        ORG     0000H
                160 ;
0000 C5         161 RESET:  SEL     RB0            ;GET INTO RB0 AT RESET
0001 4400       162        JMP     INIT           ;GO TO INITIALIZATION
                163 ;
                164 ;*****************************************************************
                165 ;
                166 ;TIMER INTERRUPT LOCATION - TIMER IS SET TO 4 TIMES THE BAUD RATE.  THE
                167 ;RECEIVER AND TRANSMITTER ARE SERVICED EVERY FOUR TIMER TICKS.  SOFTWARE
                168 ;DELAY LOOP IS USED FOR TIMING FINE-TUNING.  RB1 R1 POINTS AT DELAY
                169 ;CONSTANT AT INTERRUPT.  R1-1 POINTS AT TIMER CONSTANT.
                170 ;
                171 ;*****************************************************************
                172 ;
0007            173        ORG     0007H
                174 ;
0007 D5         175 TIMINT: SEL     RB1            ;INTERRUPT PROCESSING IN RB1
0008 AF         176        MOV     R7,A           ;SAVE ACCUMULATOR IN R7
0009 F9         177        MOV     A,R1           ;GET TIMER CONSTANT
000A 00         178        NOP                    ;DELAY TO GET INTO T1 HIGH
000B 560B       179 INT1:   JT1     INT1           ;WAIT UNTIL T1 IS LOW
000D 62         180        MOV     T,A            ;THEN LOAD COUNTER
                181 ;
                182 ;TICK SAMPLE OUTPUT
                183 ;
000E 9ABF       184        ANL     P2,#NOT TICOUT
0010 8A40       185        ORL     P2,#TICOUT
                186 ;
                187 ;*****************************************************************
                188 ;
                189 ;TRANSMITTER OUTPUT - TIME CRITICAL TASKS DONE FIRST.  DATA BIT OUTPUT
                190 ;PIPELINED IN TXSTS BIT 2 IS OUTPUT NOW.
                191 ;
                192 ;*****************************************************************
                193 ;
0012 FD         194 TXOUT:  MOV     A,TXSTS        ;GET TX STATUS
0013 5219       195        JB2     MOUT           ;TEST PIPELINED DATA
0015 9AFE       196        ANL     P2,#SPOUT      ;OUTPUT SPACE IF RESET
0017 041B       197        JMP     RCV            ;DO RECEIVER
0019 8A01       198 MOUT:   ORL     P2,#MKOUT      ;OUTPUT MARK IS SET
                199 ;
                200 ;*****************************************************************
                201 ;
                202 ;START OF RECEIVER FLOW -  RXSTS REGISTER
                203 ;HOLDS RECEIVER STATUS.
                204 ;
                205 ;*****************************************************************
                206 ;
001B C5         207 RCV:    SEL     RB0            ;SWITCH TO RX BANK
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

001C FB         208        MOV     A,RXSTS        ;GET RXSTS
001D 1226       209        JB0     RCV1           ;TEST RECEIVE FLAG
                210                               ;0 - NO CHR BEING RECEIVED
                211                               ;1 - POSSIBLE START BIT, DO TEST
001F 3668       212        JT0     XMIT           ;TEST RXD INPUT
                213                               ;0 - SPACE, SET RX FLAG
                214                               ;1 - MARK, GO CHECK XMIT
0021 4301       215        ORL     A,#RXFLG       ;SPACE - SET RX FLAG
0023 AB         216        MOV     RXSTS,A        ;RESTORE RXSTS
0024 0468       217        JMP     XMIT           ;GO HANDLE XMTR
                218 ;
                219 ;START BIT TEST
                220 ;
0026 3238       221 RCV1:  JB1     RCV3           ;FIRST TEST START BIT FLAG
0028 3633       222        JT0     RCV2           ;TEST RXD INPUT
                223                               ;0 - SPACE, GOOD START BIT
                224                               ;1 - MARK, BAD START BIT, IGNORE
002A 4302       225        ORL     A,#SRTFLG      ;GOOD START - SET START BIT FLAG
002C AB         226        MOV     RXSTS,A        ;RESTORE RXSTS
002D BE80       227        MOV     R6,#RXINTL     ;SETUP RX DESERIALIZER
002F BD04       228        MOV     R5,#TICSRT     ;LOAD RX TICK COUNTER
0031 0468       229        JMP     XMIT           ;GO HANDLE XMTR
                230 ;
                231 ;BAD START BIT - RESET FLAGS
                232 ;
0033 53FE       233 RCV2:  ANL     A,#NOT RXFLG   ;RESET RECEIVE FLAG
0035 AB         234        MOV     RXSTS,A        ;RESTORE RXSTS
0036 0468       235        JMP     XMIT           ;GO HANDLE XMTR
                236 ;
                237 ;IN MIDDLE OF CHR - SAMPLE EVERY 4 TIMER TICKS
                238 ;
0038 ED68       239 RCV3:  DJNZ    R5,XMIT        ;WAIT UNTIL 4TH TICK
003A BD04       240        MOV     R5,#TICSRT     ;RELOAD RX TICK COUNTER
003C 524D       241        JB2     RCV5           ;TEST BYTE FINISHED FLAG
                242                               ;0 - MIDDLE OF CHR, CONTINUE
                243                               ;1 - DONE WITH STOP BITS
003E 97         244        CLR     C              ;CLEAR CARRY BEFORE ROTATE
003F 2642       245        JNT0    RCV4           ;TEST RXD INPUT
0041 A7         246        CPL     C              ;RXD IS MARK, SET CARRY
0042 FE         247 RCV4:  MOV     A,R6           ;GET DESERIALIZER
0043 67         248        RRC     A              ;ROTATE IN NEW BIT
0044 AE         249        MOV     R6,A           ;RESTORE DESERIALIZER
0045 E668       250        JNC     XMIT           ;TEST CARRY AFTER ROTATE
                251                               ;0 - MIDDLE OF CHR
                252                               ;1 - STOP BIT COMING NEXT
0047 FB         253        MOV     A,RXSTS        ;GET RXSTS
0048 4304       254        ORL     A,#BFFLG       ;SET BYTE FINISHED FLAG
004A AB         255        MOV     RXSTS,A        ;RESTORE RXSTS
004B 0468       256        JMP     XMIT           ;GO HANDLE XMTR
                257 ;
                258 ;BYTE FINISHED - DO STOP BIT TEST
                259 ;
004D 2660       260 RCV5:  JNT0    RCV8           ;TEST RXD INPUT
                261                               ;0 - SPACE, INVALID STOP BIT
                262                               ;1 - MARK, VALID STOP BIT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

004F 53EF        263        ANL     A,#NOT FRAMER   ;NO FRAMING ERROR, RESET FLAG
                 264 ;
                 265 ;OVERRUN TEST - IF RX DATA READY STILL SET, OVERRUN ERROR
                 266 ;
0051 7264        267 RCV6:  JB3     RCV9            ;IF DATA READY STILL SET, ERROR
0053 53DF        268        ANL     A,#NOT OVRUN    ;NO OVERRUN, RESET FLAG
                 269 ;
                 270 ;CLEAN UP RXSTS AT CHR COMPLETE
                 271 ;
0055 4308        272 RCV7:  ORL     A,#DATRDY       ;SET DATA READY
0057 53F8        273        ANL     A,#NOT (RXFLG OR SRTFLG OR BFFLG)       ;RESET OTHER FLAGS
0059 AB          274        MOV     RXSTS,A         ;RESTORE RXSTS
005A FE          275        MOV     A,R6            ;GET DESERIALIZER REG
005B 537F        276        ANL     A,#ASCMSK       ;MAKE IT 7 BITS
005D AC          277        MOV     R4,A            ;PUT DATA INTO HOLDING REG
005E 0468        278        JMP     XMIT            ;GO HANDLE XMTR
                 279 ;
                 280 ;BAD STOP - SET FRAMING ERROR FLAG
                 281 ;
0060 4310        282 RCV8:  ORL     A,#FRAMER       ;SET FRAMING ERROR FLAG
0062 0451        283        JMP     RCV6            ;CONTINUE
                 284 ;
                 285 ;OVERRUN ERROR - SET OVERRUN FLAG
                 286 ;
0064 4320        287 RCV9:  ORL     A,#OVRUN        ;SET OVERRUN FLAG
0066 0455        288        JMP     RCV7            ;CONTINUE
                 289 ;
                 290 ;*****************************************************************
                 291 ;
                 292 ;START OF TRANSMITTER FLOW - TRANSMITTER IS SERVICED EVERY 4 TICKS.
                 293 ;THE TX TICK COUNTER SERVES AS THE TX BIT COUNTER.  TRANSMITTER STATUS
                 294 ;IS HELD IN THE TXSTS REGISTER.
                 295 ;
                 296 ;*****************************************************************
                 297 ;
0068 D5          298 XMIT:  SEL     RB1             ;BE SURE WE'RE IN RB1
0069 FD          299        MOV     A,TXSTS         ;GET TX STATUS
006A 72B3        300        JB3     SRTBIT          ;THIS IS START OF START BIT
006C 1A          301        INC     R2              ;INC TX TICK COUNTER
006D 2303        302        MOV     A,#TXTIC        ;TEST TICK COUNTER MOD 4
006F 5A          303        ANL     A,R2
0070 96B0        304        JNZ     RETURN          ;IF NON-ZERO, MIDDLE OF BIT
0072 FD          305        MOV     A,TXSTS         ;ZERO, GET TXSTS
0073 37          306        CPL     A               ;COMPLEMENT FOR 0 TEST
0074 129C        307        JB0     XMT4            ;TEST TX FLAG
                 308                                ;0 - NOT TXING, CHECK FOR NEW CHR
                 309                                ;1 - CURRENTLY IN CHR
0076 2328        310        MOV     A,#TXEND        ;CHECK FOR END OF DATA AND STOP
0078 DA          311        XRL     A,R2            ;XOR WITH CURRENT TICK COUNT
0079 9681        312        JNZ     XMT1            ;NOT DONE, CONTINUE
007B FD          313        MOV     A,TXSTS         ;DONE, GET TXSTS
007C 53FE        314        ANL     A,#NOT TXFLG    ;RESET TX FLAG
007E AD          315        MOV     TXSTS,A         ;RESTORE TXSTS
007F 04B0        316        JMP     RETURN          ;GO EXIT
                 317 ;
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                318 ;CHECK IF IT'S TIME FOR STOP BIT
                319 ;
0081 2324       320 XMT1:   MOV    A,#STPEND       ;CHECK FOR STOP BIT TIME
0083 DA         321         XRL    A,R2            ;COMPARE WITH TICK COUNTER
0084 968C       322         JNZ    XMT2            ;NOT TIME, DO NEXT BIT
                323 ;
                324 ;TRANSMIT STOP BIT
                325 ;
0086 FD         326         MOV    A,TXSTS         ;GET TX STATUS
0087 4304       327         ORL    A,#MARK         ;SETUP PIPELINED STOP BIT
0089 AD         328         MOV    TXSTS,A         ;RESTORE TX STATUS
008A 04B0       329         JMP    RETURN          ;RETURN
                330 ;
                331 ;IN MIDDLE OF CHR - TRANSMIT NEXT BIT
                332 ;
008C FB         333 XMT2:   MOV    A,R3            ;GET TX SERIALIZER
008D 67         334         RRC    A               ;ROTATE NEXT BIT INTO CARRY
008E AB         335         MOV    R3,A            ;RESTORE SERIALIZER
008F FD         336         MOV    A,TXSTS         ;GET TX STATUS FOR PIPELINED DATA
0090 F697       337         JC     XMT3            ;OUTPUT A MARK IF 1
0092 53FB       338         ANL    A,#SPACE        ;RESET TXDATA BIT
0094 AD         339         MOV    TXSTS,A         ;RESTORE TX STATUS
0095 04B0       340         JMP    RETURN          ;GO EXIT
0097 4304       341 XMT3:   ORL    A,#MARK         ;SET TXDATA BIT
0099 AD         342         MOV    TXSTS,A         ;RESTORE TX STATUS
009A 04B0       343         JMP    RETURN          ;GO EXIT
                344 ;
                345 ;TEST REQUEST FLAG SINCE NOT CURRENTLY TRANSMITTING
                346 ;
009C 32A8       347 XMT4:   JB1    XMT5            ;TEST TX REQUEST FLAG
                348                                ;0 - NO CHR WAITING IN BUFFER
                349                                ;1 - CHR WAITING IN BUFFER
009E FC         350         MOV    A,R4            ;CHR WAITING, GET IT FROM HOLDING
009F AB         351         MOV    R3,A            ;PUT IN SERIALIZER
00A0 FD         352         MOV    A,TXSTS         ;GET TXSTS
00A1 53FD       353         ANL    A,#NOT REQFLG   ;RESET REQUEST FLAG
00A3 4309       354         ORL    A,#TXFLG OR SBIT        ;SET TX AND START BIT FLAGS
00A5 53FB       355         ANL    A,#SPACE        ;SETUP TXDATA FOR START BIT
00A7 AD         356         MOV    TXSTS,A         ;RESTORE TXSTS
                357 ;
                358 ;TX BUFFER EMPTY - SET TXINT PIN AND BIT
                359 ;
00A8 8A08       360 XMT5:   ORL    P2,#TXINT       ;SET TXINT PIN
00AA C5         361         SEL    RB0             ;SWITCH FOR STS
00AB FF         362         MOV    A,R7            ;GET STS
00AC 4320       363         ORL    A,#TXBIT        ;SET TXINT BIT
00AE AF         364         MOV    R7,A            ;RESTORE STS
00AF 90         365         MOV    STS,A           ;LOAD STATUS
                366 ;
                367 ;*****************************************************************
                368 ;
                369 ;EXIT FOR TIMER INTERRUPT ROUTINE POINT
                370 ;
                371 ;*****************************************************************
                372 ;
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0080 D5          373 RETURN: SEL     RB1            ;MAKE SURE WE'RE IN RB1
0081 FF          374         MOV     A,R7           ;RESTORE A
0082 93          375         RETR                   ;RETURN WITH RESTORE
                 376 ;
                 377 ;*****************************************************************
                 378 ;
                 379 ;GET HERE IF INTERRUPT IS FIRST FOR START BIT - CLEAR START BIT FLAG IN
                 380 ;TXSTS AND SETUP TX TICK COUNTER.
                 381 ;
                 382 ;*****************************************************************
                 383 ;
0083 53F7        384 SRTBIT: ANL     A,#NOT SBIT    ;RESET START BIT FLAG IN TXSTS
0085 AD          385         MOV     TXSTS,A        ;RESTORE TX STATUS
0086 BA01        386         MOV     R2,#01H        ;INITIALIZE TX TICK COUNTER
0088 04B0        387         JMP     RETURN         ;RETURN
                 388 ;
                 389 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                390 ;
                391 ;*****************************************************************
                392 ;
                393 ;COMMAND RECOGNIZER - GET HERE FROM IBF WRITE WITH F1 SET.  COMMAND
                394 ;IS STORED IN R6.  BAUD RATE SELECTION BITS ARE EVALUATED RIGHT TO LEFT.
                395 ;THE FIRST SET BIT FOUND DETERMINES THE BAUD RATE.  IF AN INVALID COMMAND
                396 ;IS DETECTED, BOTH F1 AND F0 ARE SET AND NO ACTION IS TAKEN.
                397 ;THE TIMER BAUD RATE CONSTANT IS SET TO TWO COUNTS LESS THAN THE DESIRED
                398 ;NUMBER.
                399 ;
                400 ;*****************************************************************
                401 ;
0100            402        ORG     0100H
                403 ;
0100 D5         404 CMD:   SEL     RB1          ;SELECT RB1
0101 22         405        IN      A,DBB        ;READ COMMAND
0102 AE         406        MOV     R6,A         ;SAVE COMMAND IN R6
0103 F227       407        JB7     IOER         ;IF BIT 7 SET, IO OPERATION
0105 53E0       408        ANL     A,#0E0H      ;TEST TOP 3 BITS
0107 963A       409        JNZ     ERROR        ;IF NON-ZERO, ERROR
0109 C5         410        SEL     RB0          ;IO FLAG IN RB0
010A 1221       411        JB0     CMD2         ;IF BIT 0=1, OUTPUT PORT
010C 89FF       412        ORL     P1,#0FFH     ;INPUT PORT, SET ALL HIGH
010E FB         413        MOV     A,RXSTS      ;GET RXSTS
010F 53BF       414        ANL     A,#NOT IODIR ;RESET IO DIRECTION FLAG
0111 AB         415        MOV     RXSTS,A      ;RESTORE RXSTS
0112 D5         416 CMD1:  SEL     RB1          ;BAUD RATE CONSTANTS IN RB1
0113 B920       417        MOV     R1,#TIMCON   ;POINT AT TIMER CONSTANT LOCATION
0115 FE         418        MOV     A,R6         ;GET COMMAND
0116 323E       419        JB1     B110         ;110 BAUD SELECTED
0118 5242       420        JB2     B300         ;300 BAUD SELECTED
011A 7246       421        JB3     B600         ;600 BAUD SELECTED
011C 924A       422        JB4     B1200        ;1200 BAUD SELECTED
011E B5         423        CPL     F1           ;RESET F1
011F 4414       424        JMP     MNLP1        ;DONE, JUMP BACK TO MAIN LOOP
                425 ;
                426 ;PORT IS SELECTED AS OUTPUT PORT - SET IO DIRECTION FLAG
                427 ;
0121 FB         428 CMD2:  MOV     A,RXSTS      ;GET RXSTS
0122 4340       429        ORL     A,#IODIR     ;SET IO DIRECTION FLAG
0124 AB         430        MOV     RXSTS,A      ;RESTORE RXSTS
0125 2412       431        JMP     CMD1         ;CONTINUE
                432 ;
                433 ;HERE WITH EITHER IO OR RESET ERROR COMMAND
                434 ;
0127 D231       435 IOER:  JB6     ERRST        ;IF BIT 6 SET, RESET ERROR FLAGS
0129 C5         436        SEL     RB0          ;IO FLAG IN RXSTS
012A FB         437        MOV     A,RXSTS      ;GET RXSTS
012B 4380       438        ORL     A,#IOFLG     ;SET IO FLAG
012D AB         439        MOV     RXSTS,A      ;RESTORE RXSTS
012E B5         440        CPL     F1           ;RESET F1
012F 4414       441        JMP     MNLP1        ;DONE, JUMP BACK TO MAIN LOOP
                442 ;
                443 ;RESET ERROR COMMAND
                444 ;
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0131 C5          445 ERRST:  SEL     RB0            ;STS IN RB0
0132 FF          446         MOV     A,R7           ;GET STS
0133 533F        447         ANL     A,#RSTERR      ;RESET ERROR FLAGS
0135 AF          448         MOV     R7,A           ;RESTORE STS
0136 90          449         MOV     STS,A          ;LOAD STATUS
0137 B5          450         CPL     F1             ;RESET F1
0138 4414        451         JMP     MNLP1          ;DONE, BACK TO MAIN LOOP
                 452 ;
                 453 ;COMMAND ERROR - SET BOTH F1 AND F0
                 454 ;
013A 85          455 ERROR:  CLR     F0             ;SET F0
013B 95          456         CPL     F0
013C 4414        457         JMP     MNLP1          ;DONE, BACK TO MAIN LOOP
                 458 ;
                 459 ;110 BAUD CONSTANTS
                 460 ;
013E B954        461 B110:   MOV     R1,#-(174D-2D) ;LOAD 110 BAUD CONSTANT
0140 244C        462         JMP     STTIMR         ;GO START TIMER
                 463 ;
                 464 ;300 BAUD CONSTANTS
                 465 ;
0142 B9C2        466 B300:   MOV     R1,#-(64D-2D)  ;LOAD 300 BAUD CONSTANT
0144 244C        467         JMP     STTIMR         ;GO START COUNTER
                 468 ;
                 469 ;600 BAUD CONSTANTS
                 470 ;
0146 B9E2        471 B600:   MOV     R1,#-(32D-2D)  ;LOAD 600 BAUD CONSTANT
0148 244C        472         JMP     STTIMR         ;GO START COUNTER
                 473 ;
                 474 ;1200 BAUD CONSTANTS
                 475 ;
014A B9F2        476 B1200:  MOV     R1,#-(16D-2D)  ;LOAD 1200 BAUD CONSTANT
                 477 ;
                 478 ;START COUNTER
                 479 ;
014C F9          480 STTIMR: MOV     A,R1           ;GET COUNTER CONSTANT
014D 62          481         MOV     T,A            ;LOAD COUNTER
014E 45          482         STRT    CNT            ;START COUNTER
014F 25          483         EN      TCNTI          ;ENABLE TIMER INTERRUPTS
0150 B5          484         CPL     F1             ;RESET F1
0151 4414        485         JMP     MNLP1          ;DONE, BACK TO MAIN LOOP
                 486 ;
                 487 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                488 ;
                489 ;*****************************************************************
                490 ;
                491 ;DATA ROUTINE - GET HERE WITH IBF WRITE WITH F1 RESET.  THIS ROUTINE
                492 ;FIRST TESTS IF THE I/O FLAG IS SET IN THE RXSTS REGISTER.  IF SO, THE DATA
                493 ;IS FOR THE OUTPUT PORT.  OTHERWISE, THE DATA IS FOR THE TRANSMITTER AND
                494 ;IS PLACED IN THE TX BUFFER REGISTER.  THE TXINT BIT AND PIN ARE RESET.
                495 ;
                496 ;*****************************************************************
                497 ;
0153 C5         498 DATA:  SEL    RB0            ;DATA HANDLED MOSTLY IN RB0
0154 FB         499        MOV    A,RXSTS        ;GET RXSTS
0155 F267       500        JB7    IODATA         ;IF IO FLAG SET, DATA IN FOR I/O
0157 FF         501        MOV    A,R7           ;GET STS
0158 53DF       502        ANL    A,#NOT TXBIT   ;RESET TXINT BIT IN STS
015A AF         503        MOV    R7,A           ;RESTORE STS
015B 90         504        MOV    STS,A          ;LOAD STATUS
015C 9AF7       505        ANL    P2,#NOT TXINT  ;RESET TXINT PIN
015E D5         506        SEL    RB1            ;TXSTS IN RB1
015F 22         507        IN     A,DBB          ;READ DATA
0160 AC         508        MOV    R4,A           ;PUT DATA IN TX BUFFER
0161 FD         509        MOV    A,TXSTS        ;GET TXSTS
0162 4302       510        ORL    A,#REQFLG      ;SET REQUEST FLAG IN TXSTS
0164 AD         511        MOV    TXSTS,A        ;RESTORE TXSTS
0165 4414       512        JMP    MNLP1          ;BACK TO MAIN LOOP
                513 ;
                514 ;IO DATA ROUTINE
                515 ;
0167 537F       516 IODATA: ANL   A,#NOT IOFLG   ;RESET IO FLAG
0169 AB         517        MOV    RXSTS,A        ;RESTORE RXSTS
016A 22         518        IN     A,DBB          ;READ IO DATA FROM DBBIN
016B 39         519        OUTL   P1,A           ;OUTPUT TO PORT 1
016C 4414       520        JMP    MNLP1          ;DONE, BACK TO MAIN LOOP
                521 ;
                522 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                523 ;
                524 ;*********************************************************************
                525 ;
                526 ;INITIALIZATION - GET HERE AT RESET.  THIS ROUTINE RESETS THE INTERRUPT
                527 ;OUTPUTS AND ENABLES THEM, AND CLEARS THE APPROPRIATE STATUS AND DATA
                528 ;REGISTERS.
                529 ;
                530 ;*********************************************************************
                531 ;
0200            532        ORG     0200H
                533 ;
0200 9AF7       534 INIT:  ANL     P2,#0F7H              ;RESET TXINT PIN
0202 F5         535        EN      FLAGS            ;ENABLE INTERRUPTS OUTPUT
0203 2300       536        MOV     A,#ZERO          ;CLEAR A
0205 AB         537        MOV     RXSTS,A          ;CLEAR RXSTS
0206 AD         538        MOV     R5,A             ;CLEAR RX TICK COUNTER
0207 AF         539        MOV     R7,A             ;CLEAR STS
0208 D5         540        SEL     RB1              ;SWITCH BANKS
0209 AE         541        MOV     R6,A             ;CLEAR CONFIGURE STORE
020A BD04       542        MOV     TXSTS,#MARK      ;SETUP PIPELINED TX DATA
                543 ;
                544 ;*********************************************************************
                545 ;
                546 ;MAIN LOOP - IBF AND OBF ARE HANDLED IN THIS LOOP.  IF IBF=1, THE
                547 ;APPROPRIATE COMMAND OR DATA ROUTINE IS ACCESSED.  IF IBF=0, THEN OBF
                548 ;IS TESTED.  IF OBF=1, IBF IS TESTED AGAIN.  AS SOON AS OBF=0, RXSTS
                549 ;IS EXAMINED TO SEE IF DATA IS WAITING FOR OUTPUT.  WHEN RX DATA
                550 ;READY IS SET, F0 IS SET AND F1 IS CLEARED, AND THE DATA IS TRANSFERRED
                551 ;FROM THE RX HOLDING REGISTER INTO DBBOUT AFTER TESTING FOR ERROR
                552 ;FLAGS.  ANY ERROR FLAGS SET ARE TRANSFERRED TO THE STATUS REGISTER.
                553 ;IF THE I/O FLAG IS SET, THE PORT IS READ AND THE DATA TRANSFERRED TO
                554 ;DBBOUT.
                555 ;
                556 ;*********************************************************************
                557 ;
020C D614       558 MNLOOP: JNIBF  MNLP1            ;IF IBF=0, TEST OBF
020E 7612       559        JF1     CMDJ1            ;IBF=1, TEST F1 FOR COMMAND
0210 2453       560        JMP     DATA             ;F1=0, JUMP TO DATA ROUTINE
0212 2400       561 CMDJ1:  JMP    CMD              ;OUT-OF-PAGE COMMAND JUMP
0214 860C       562 MNLP1:  JOBF   MNLOOP           ;WAIT UNTIL DBBOUT IS FREE
0216 C5         563        SEL     RB0              ;RXSTS IN RB0
0217 FB         564        MOV     A,RXSTS          ;GET RXSTS
0218 721E       565        JB3     RXRDY            ;TEST RX DATA READY FLAG
021A F23C       566        JB7     IOFLAG           ;TEST IO FLAG
021C 440C       567        JMP     MNLOOP           ;LOOP
                568 ;
                569 ;RX DATA READY - TRANSFER TO DBBOUT
                570 ;
021E 85         571 RXRDY:  CLR    F0               ;SET F0
021F 95         572        CPL     F0
0220 A5         573        CLR     F1               ;RESET F1
0221 922E       574        JB4     RXF              ;CHECK FRAMING ERROR FLAG
0223 FB         575 RXRDY1: MOV    A,RXSTS          ;GET RXSTS
0224 B235       576        JB5     RXO              ;CHECK FOR OVERRUN ERROR
0226 FB         577 RXRDY2: MOV    A,RXSTS          ;GET RXSTS AGAIN
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0227 53C7       578        ANL    A,#NOT (DATRDY OR FRAMER OR OVRUN)     ;RESET SOME FLAGS
0229 AB         579        MOV    RXSTS,A        ;RESTORE RXSTS
022A FC         580        MOV    A,R4           ;GET DATA FROM HOLDING REG
022B 02         581        OUT    DBB,A          ;PUT IN DBBOUT
022C 440C       582        JMP    MNLOOP         ;LOOP
                583 ;
                584 ;FRAMING ERROR FLAG SET
                585 ;
022E FF         586 RXF:   MOV    A,R7           ;GET STS
022F 4340       587        ORL    A,#FESTS       ;SET FRAMING ERROR FLAG
0231 AF         588        MOV    R7,A           ;RESTORE STS
0232 90         589        MOV    STS,A          ;LOAD STATUS
0233 4423       590        JMP    RXRDY1         ;CONTINUE
                591 ;
                592 ;OVERRUN ERROR FLAG SET
                593 ;
0235 FF         594 RXO:   MOV    A,R7           ;GET STS
0236 4380       595        ORL    A,#OVSTS       ;SET OVERRUN ERROR FLAG
0238 AF         596        MOV    R7,A           ;RESTORE STS
0239 90         597        MOV    STS,A          ;LOAD STATUS
023A 4426       598        JMP    RXRDY2         ;CONTINUE
                599 ;
                600 ;IO FLAG SET - TEST DIRECTION
                601 ;
023C FB         602 IOFLAG: MOV   A,RXSTS        ;GET RXSTS
023D D20C       603        JB6    MNLOOP         ;PORT IS OUTPUT - NO ACTION
023F 85         604        CLR    F0             ;RESET F0
0240 A5         605        CLR    F1             ;SET F1
0241 B5         606        CPL    F1
0242 537F       607        ANL    A,#NOT IOFLG   ;RESET IO FLAG
0244 AB         608        MOV    RXSTS,A        ;RESTORE RXSTS
0245 09         609        IN     A,P1           ;READ PORT 1
0246 02         610        OUT    DBB,A          ;PUT DATA IN DBBOUT
0247 440C       611        JMP    MNLOOP         ;LOOP
                612 ;
                613        END
```

USER SYMBOLS

ASCMSK	007F	B110	013E	B1200	014A	B300	0142	B600	0146	BFFLG	0004	CMD	0100	CMD1	0112
CMD2	0121	CMDJ1	0212	DATA	0153	DATRDY	0008	ERROR	013A	ERRST	0131	FESTS	0040	FRAMER	0010
INIT	0200	INT1	000B	IODATA	0167	IODIR	0040	IOER	0127	IOFLAG	023C	IOFLG	0080	MARK	0004
MKOUT	0001	MNLOOP	020C	MNLP1	0214	MOUT	0019	OVRUN	0020	OVSTS	0080	RCV	001B	RCV1	0026
RCV2	0033	RCV3	0038	RCV4	0042	RCV5	004D	RCV6	0051	RCV7	0055	RCV8	0060	RCV9	0064
REQFLG	0002	RESET	0000	RETURN	00B0	RSTERR	003F	RXF	022E	RXFLG	0001	RXINTL	0080	RXO	0235
RXRDY	021E	RXRDY1	0223	RXRDY2	0226	RXSTS	0003	SBIT	0008	SPACE	00FB	SPOUT	00FE	SRTBIT	00B3
SRTFLG	0002	STPEND	0024	STTIMR	014C	TICOUT	0040	TICSRT	0004	TIMCON	0020	TIMINT	0007	TXBIT	0020
TXEND	0028	TXFLG	0001	TXINT	0008	TXOUT	0012	TXSTS	0005	TXTIC	0003	XMIT	0068	XMT1	0081
XMT2	008C	XMT3	0097	XMT4	009C	XMT5	00A8	ZERO	0000						

ASSEMBLY COMPLETE, NO ERRORS

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 1 ;
                 2 ;TEST ROUTINE WHICH OUTPUTS THE ASCII CHARACTER SET TO THE
                 3 ;UPI TRANSMITTER AND DISPLAYS ON THE 80/30 CONSOLE ANY
                 4 ;CHARACTERS RECEIVED BY THE UPI RECEIVER.
                 5 ;
                 6 ;INPUTS: NOTHING
                 7 ;OUTPUTS: CHARACTERS TO CONSOLE
                 8 ;CALLS: NOTHING
                 9 ;
4000            10          ORG     4000H
00DF            11 MODE53   EQU     0DFH            ;8253 CONTROL PORT
00DC            12 CNT0     EQU     0DCH            ;8253 CNT 0 PORT
00E5            13 CMD      EQU     0E5H            ;UPI COMMAND PORT
00E5            14 STATUS   EQU     0E5H            ;UPI STATUS PORT
00E4            15 DBBIN    EQU     0E4H            ;UPI DBBIN PORT
00E4            16 DBBOUT   EQU     0E4H            ;UPI DBBOUT PORT
0020            17 TXINT    EQU     20H             ;TXINT MASK
0001            18 OBF      EQU     01H             ;OBF MASK
0002            19 IBF      EQU     02H             ;IBF MASK
00ED            20 STAT51   EQU     0EDH            ;8251 STATUS PORT
00EC            21 DATA51   EQU     0ECH            ;8251 DATA PORT
0001            22 TXRDY    EQU     01H             ;8251 TXRDY MASK
                23 ;
4000 3E36       24 START:   MVI     A,36H           ;8253 CNT0 MODE WORD
4002 D3DF       25          OUT     MODE53          ;8253 CONTROL PORT
4004 3E10       26          MVI     A,10H           ;DIVIDE BY 16D
4006 D3DC       27          OUT     CNT0            ;8253 CNT0 PORT LSB
4008 3E00       28          MVI     A,00H           ;
400A D3DC       29          OUT     CNT0            ;8253 CNT0 PORT MSB
400C 0620       30          MVI     B,20H           ;INITIALIZE OUTPUT CHR
400E 3E10       31          MVI     A,10H           ;CONFIGURE COMMAND - 1200 BAUD
4010 D3E5       32          OUT     CMD             ;UPI COMMAND PORT
4012 DBE5       33 POLL1:   IN      STATUS          ;READ UPI STATUS
4014 E621       34          ANI     TXINT OR OBF    ;TEST TXINT AND OBF
4016 CA1240     35          JZ      POLL1           ;WAIT UNTIL ONE IS SET
4019 DBE5       36          IN      STATUS          ;READ UPI STATUS AGAIN
401B E601       37          ANI     OBF             ;WAS IT OBF?
401D C23840     38          JNZ     RX              ;YES, GO DO RECEIVER
                39                                  ;NO, MUST BE TRANSMITTER
4020 78         40          MOV     A,B             ;GET NEXT CHR FOR OUTPUT
4021 D3E4       41          OUT     DBBIN           ;OUTPUT TO UPI DBBIN
4023 FE5A       42          CPI     'Z'             ;WAS IT LAST CHR?
4025 CA3340     43          JZ      NEWB            ;YES, RESET REG. B
4028 04         44          INR     B               ;OTHERWISE, INC B
4029 DBE5       45 POLL2:   IN      STATUS          ;TEST IF IBF STILL SET
402B E602       46          ANI     IBF             ;TEST IBF
402D C22940     47          JNZ     POLL2           ;WAIT UNTIL IBF=0
4030 C31240     48          JMP     POLL1           ;BEFORE LOOKING AT STATUS AGAIN
                49 ;
4033 0620       50 NEWB:    MVI     B,20H           ;RESET REG. B
4035 C32940     51          JMP     POLL2           ;GO BACK
                52 ;
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

4038 DBE4         53 RX:    IN      DBBOUT        ;READ DBBOUT FOR RECEIVED CHR
403A 4F           54        MOV     C,A           ;SAVE IT IN C
403B DBED         55 RX1:   IN      STAT51        ;READ 8251 STATUS
403D E601         56        ANI     TXRDY         ;TEST TXRDY
403F CA3B40       57        JZ      RX1           ;WAIT UNTIL READY
4042 79           58        MOV     A,C           ;GET CHR
4043 D3EC         59        OUT     DATA51        ;OUTPUT CHR TO CONSOLE
4045 C31240       60        JMP     POLL1         ;GO TEST UPI AGAIN
                  61 ;
                  62 END

PUBLIC SYMBOLS


EXTERNAL SYMBOLS


USER SYMBOLS
CMD    A 00E5   CNT0   A 00DC   DATA51 A 00EC   DBBIN  A 00E4   DBBOUT A 00E4   1BF    A 0002   MODE53 A 00DF
NEWB   A 4033   OBF    A 0001   POLL1  A 4012   POLL2  A 4029   RX     A 4038   RX1    A 403B   START  A 4000
STAT51 A 00ED   STATUS A 00E5   TXINT  A 0020   TXRDY  A 0001

ASSEMBLY COMPLETE,  NO ERRORS
```

Using the 8202 Dynamic RAM Controller

by Bruce McCormick

INTRODUCTION

Dynamic RAMs have always been somewhat of a problem for system designers. The problems associated with refreshing, timing, and interfacing have caused many RAM intensive microprocessor applications to utilize only static memory. This is unfortunate because dynamic memory is more attractive on the basis of cost, power consumption, and bit density. The Intel 8202 was designed to eliminate the bulk of the problems associated with dynamic RAMs and to make interfacing simple and cost effective.

The 8202 provides for address decoding, multiplexing and all the timing needed to refresh 16 pin dynamic (multiplexed address) RAMs. All refresh operations are handled by the 8202 and refreshing is *not* dependent on the state of the controlling system. This means that long reset times, single stepping, or extended bus holds will not destroy RAM contents, as would be the case with processor controlled refreshing. In essence, the 8202 has the ability to make dynamic memory look static to the user.

This note will describe in detail the Intel 8202 and illustrate its use. Broken into three major sections, each successive section discusses a different aspect on using the 8202. The first section describes the functionality of the 8202 and illustrates its use in general. The second section is more specific and discusses, as an overview, the interface, decoupling and speed selection of the memory array. The last section illustrates 8085A application examples and an interconnect of the 8086 max mode system with 128K bytes of dynamic RAM.

For those readers who are unfamiliar with dynamic RAM or would like to "refresh" their memory, some basics about dynamic RAM are attached in Appendix 1. Appendix 2 details the basics on laying out a MOS array, a critical area of importance when trying to develop a trouble-free system. Also attached is Appendix 3, an exhaustive analysis of 8202's compatibility with the Intel industry standard dynamic RAMs. If you've never worked with dynamic RAM before, it is highly recommended that you obtain and read the dynamic RAM sections of the Intel Memory Design Handbook. Also, for a comparison and supplemental background, you may want to refer to Intel Application Report #1, "Simplify Your Dynamic RAM/Microprocessor Interface," and Application Note #38, "Application Techniques for the Intel 8085A Bus." These notes illustrate the complexities that *used* to be involved when interfacing to dynamic RAM. An 8202 data sheet will prove very beneficial when reading this note.

FUNCTIONAL DESCRIPTION

Overview

The 8202 is a bipolar device packaged in a 40 pin dual-in-line package. This Dynamic RAM controller performs all the system control support needed to operate and refresh up to 16K bytes of 2104A (4K × 1), 64K bytes of 2117, 2118 (16K × 1) in an 8080A, 8085A system or 128K bytes for 8086. To accomplish this the 8202 has the following features:

- Provides address bus multiplexing, compatible with address multiplexed RAMs.
- Provides sufficient output drive for up to 4 banks of 8 2104A's, 2117's or 2118's. All necessary $\overline{CAS}$ (Column Address Strobe), $\overline{RAS}$ (Row Address Strobe), $\overline{WE}$ (Write Enable) and $\overline{CS}$ (Chip Select) signals are generated.
- Provides failsafe refresh. Memory is refreshed in a distributed manner ($\overline{RAS}$ only refresh) *without* using the inefficient method of asking the processor to relinquish the bus. The processor is asked to wait if it requests a memory cycle when a refresh cycle is being performed.
- Provides externally controlled refresh option for those who want sync or hidden refresh.
- Allows direct interface with the 8080A bus, the demultiplexed 8085A bus, the 8086 maximum mode bus, and DMA controllers.
- Allows convenient and compatible debugging with ICE modules and μScope.
- Allows all memory and refresh requests to be asynchronously requested.

A block diagram of the 8202 is given in Figure 1 illustrating how these features are integrated. Each function is described below.

Oscillator

The Intel 8202 generates its timing from a crystal* controlled internal shift register technique. This method provides highly accurate control of the fine resolution of timing required for dynamic RAM. This is far superior to a monostable multivibrator approach where glitches and unit to unit timing accuracies are difficult to control.

Arbiter

The arbiter resolves all conflicts between any cycles that are requested. These cycle requests can be generated from one of four places:

A. Read cycle request — $\overline{RD}$/S1 input
B. Write cycle request — $\overline{WR}$ input
C. External refresh request — REFRQ/ALE
D. Internal refresh request (refresh timer shown in block diagram)

If a refresh cycle is in progress and another cycle is requested, the requesting device is asked to wait until the present cycle is completed. After completion of the present refresh cycle a response from a system acknowledge output, called $\overline{SACK}$, will notify the

* An external TTL clock can also be used.

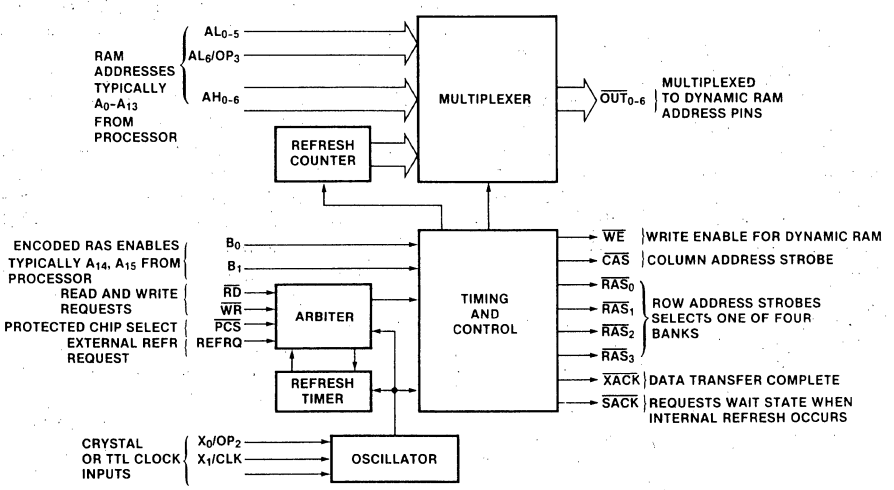


Figure 1. 8202 Block Diagram

requesting device of availability for use. If a read or write request occurs simultaneously with a refresh request, the read or write cycle will be performed first, then the refresh cycle. Read and write cycle requests should not occur simultaneously during normal operation. If the 8202 is deselected, only an internal or external refresh cycle request will be accepted; however, it will continue with the present memory cycle if one is being performed. (Hence the chip select input is called protected chip select, $\overline{PCS}$, because the current cycle is always completed regardless of any other pending request.)

Refresh Timer and Counter

The refresh timer increments on each pulse from the clock input until it reaches a preset number that causes an internal refresh request to occur. Note that this causes the refresh rate to be 8202 clock cycle dependent. External refresh requests will cause the refresh timer to reset, but not disable it.

The internal counter contains the $\overline{RAS}$ address that will be used during the refresh. The counter is incremented after each refresh, resetting to zero after all $\overline{RAS}$ addresses have been refreshed.

Multiplexer

The multiplexer is controlled by the timing and control logic. It presents to the address bus one of the following;

1. The refresh counter when there is a refresh cycle
2. AL_{0-6} on a $\overline{RAS}$ pulse (AL_{0-5}, AH_6 for $\overline{CS}$ on 4K 2104A's)
3. AH_{0-6} on a $\overline{CAS}$ pulse (AH_6 is still used as $\overline{CS}$ for 2104A)

The outputs from the multiplexer are inverted from the address inputs. This is immaterial to the dynamic RAM array and does *not* require inversion for proper system operation.

Timing and Control

The timing and control logic allows either a read, write or refresh cycle to occur. After any read or write cycle request, $\overline{SACK}$ (System ACKnowledge) goes active if the cycle was not requested during a refresh cycle. If it was, $\overline{SACK}$ is delayed until $\overline{XACK}$, thereby requesting wait states from the cycle requester.

Support Functions

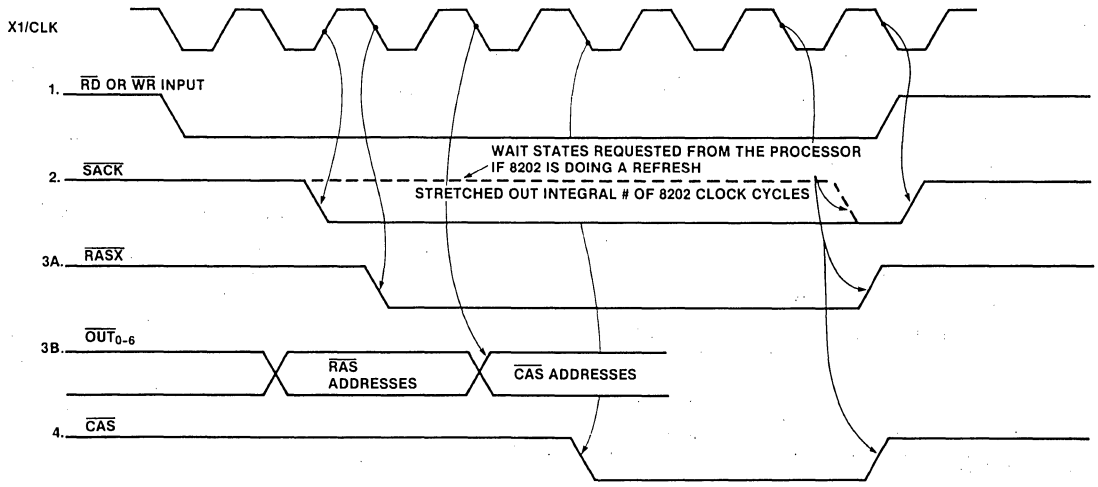
Generation of $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$ and $\overline{CS}$

All the dynamic RAM controlling signals are taken from taps off an internal traveling ones shift register. This internal shift register is driven by the clock given at the 8202 X_0, X_1 inputs, whether it is crystal or external drive. The taps on this internal shift register provide integral increments of the clock period, giving accurate control of the $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$ and $\overline{CS}$ signals, as illustrated in Figure 2. Timing begins after the internal shift register clock is synchronized with the processor, DMA or refresh request control signals.

Cycle Start

Externally, starting a cycle can occur from one of three inputs: $\overline{RD}$, $\overline{WR}$ or a high pulse on the REFRQ input. Internally, memory cycles can also be inititated from an internal refresh request. Arbitration between memory or refresh cycles is performed internally in the 8202.

To perform this arbitration, a refresh request (internal or external using REFRQ) is clocked in on the rising edge



SEQUENCE OF EVENTS: 1. READ OR WRITE REQUESTS A MEMORY CYCLE. 2. SACK RETURNS, TELLING PROCESSOR THAT 8202 IS READY. (IF REFRESH IS BEING PERFORMED, IT WILL STAY HIGH UNTIL COMPLETION. 3A, 3B. ADDRESSES ARE SET UP FOR RAS, THEN RAS GOES LOW. 4. COLUMN ADDRESSES ARE SET UP AND CAS GOES LOW TO STROBE INTO DYNAMIC RAM.

Figure 2. Timing Signals Generated by the Intel 8202 When a Cycle is Requested.

of the X_{in} clock input as shown in Figure 3 (note 1). The internal refresh logic is then activated on the following rising edge of the clock (note 5), if no memory cycle is currently in progress. At this time, the refresh address (taken from the internal refresh counter) is multiplexed out to the memory and refresh control is set up (note 6, 7). If a memory cycle is in progress, the refresh request is latched until serviced (at the end of the current memory cycle).

Conversely, if no refresh is in progress, a $\overline{RD}$ or $\overline{WR}$ could be setting up while the clock is low (note 2). On the next rising edge of the clock a cycle is started. If $\overline{RD}$, $\overline{WR}$ setup time was not met, the request will synchronize with the next cycle (if no refresh is requested). If both a refresh and a memory cycle were requested simultaneously, the cycle performed will be a memory cycle as this will be recognized a half cycle earlier (please refer to Figure 3 for clarification). If a refresh request was made during a memory cycle, it is postponed until after the memory cycle is completed, as refresh requests are *latched* until serviced.

When refresh cycles occur, any requested memory cycle is delayed through the use of the $\overline{SACK}$ output. This output is typically NANDed with other system acknowledge signals to the ready pin of the processor, asking the processor to wait until the refresh is completed. If no dynamic RAM memory cycle is requested during a refresh, the processor operation will not be disturbed as $\overline{SACK}$ only responds after a cycle is requested from the 8202. This is a more efficient method than using the hold input as overhead cycles are needed to recognize hold, generate HLDA and later regain control of the system bus.

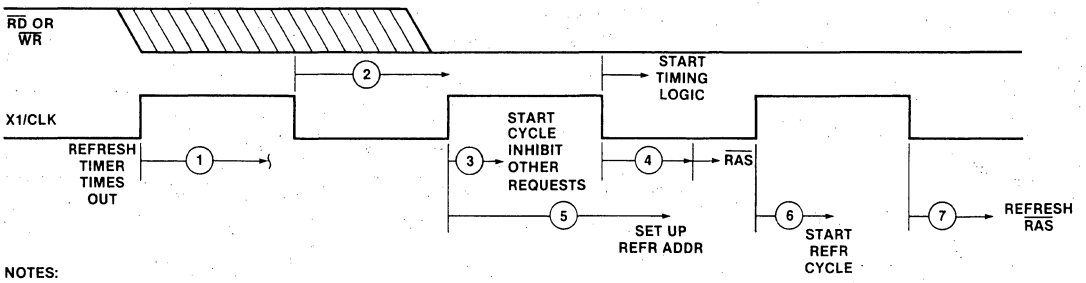
With this type of arbitration, there is less contention between a refresh or memory cycle request as it is eliminated through the synchronization with different 8202 clock edges. This arbitration method also results in the 8202 clock, system clock, refresh requests, and memory read and write requests being totally asynchronous with respect to each other. Reads and writes, however, should not be requested simultaneously.

Another area of concern with arbiters is the hangup that can occur when an input edge corresponds exactly with the clock edge. All systems have this problem but the 8202 is designed to statistically eliminate this problem. Compared to a standard TTL flip flop, the 8202 (if the input edge was exactly in sync with the clock edge) is estimated to be 10-15 orders of magnitude improvement in mean time to failure (i.e., years). This is accomplished by the fact that internally, the 8202 has 3 flip flops and 1 latch in series to resolve the arbitration. For hangup to occur, all four logic elements must hangup — statistically not feasible.

Refresh Modes

As was shown in the block diagram in Figure 1, the refresh logic is broken into two blocks: The refresh address counter and the timer. The counter contains the address of the next RAM location to be refreshed while the timer controls as to when a refresh is needed. This refresh timer is basically a frequency divider that divides the input clock frequency down to the refresh rate required.

With this type of refresh logic, internal refresh control is done in a distributed manner, i.e., a refresh cycle is per-



NOTES:

1 REFRESH TIMER TIMES OUT ON A POSITIVE CLOCK EDGE.

2 A $\overline{RD}$ OR $\overline{WR}$ SETS UP TO REQUEST A CYCLE WHEN THE CLOCK IS LOW.

3 A MEMORY CYCLE STARTS WHEN THE XIN GOES HIGH.

4 TIMING LOGIC IS STARTED AND THE $\overline{RAS}$ STROBE IS GENERATED.

5 IF THERE IS NO CYCLE IN PROGRESS, THE CYCLE MODE IS SWITCHED TO REFRESH FROM THE RISING CLOCK EDGE. A REFRESH CYCLE IS NOW REQUESTED. REFRESH ADDRESSES ARE SET UP.

6 REFRESH CYCLE IS STARTED.

7 TIMING LOGIC HAS STARTED AND THE REFRESH $\overline{RAS}$ STROBE IS GENERATED.

Figure 3. Cycle Start Timing Diagram

formed approximately every 2 ms/# $\overline{RAS}$ addresses period of time. It is performed as a $\overline{RAS}$ only refresh, the preferred approach as this does not turn on all the internal memory circuitry as would a $\overline{RAS}/\overline{CAS}$ refresh. Less circuitry that is on means less power consumption, and less noise spiking in the system. Remembering that 4K dynamic RAMs require 64 cycle refresh for each 2 ms of time and 16K dynamic RAMs require 128 cycle (as they are organized as 128 rows × 128 columns) it was necessary to have different refresh rates for the different RAMs. The 8202 knows it should perform a 64 cycle versus a 128 cycle refresh due to the 4K/16K option selected on input pin AL_6. Looking at the 8202 refresh specifications, the exact amount of delay between refresh requests can be determined as follows:

For 2104A Configuration:

Min	Max
548 × EXTERNAL 8202 Clock period	576 × EXTERNAL 8202 Clock period

For 2117, 2118 Configuration:

Min	Max
264 × EXTERNAL 8202 Clock period	288 × EXTERNAL 8202 Clock period

The refresh timing is generated after the internal refresh counter has incremented the above number of cycles. It may seem odd that there is a range of counts between min and max. This was designed as such to reduce power consumption. The lower three bits in the counter are indeterminant when reset, which occurs every time the counter reaches terminal count. The maximums above were selected to give proper refresh for the 2104A and 2117 at the 18.432 MHz frequency associated with the 8080A system and above to 25 MHz (top operating frequency of the 8202). To illustrate this, the following example is given:

For 8080A clock frequency of 18.432 MHz

Clock Period = 1/18.432 MHz = 54.25 ns

For 2104A

Max time between refresh = 576 (54.25 ns) = 31.25 μsec

which is the same as 2 ms/64 = 31.25 μsec.

For 2117

Max time between refresh = 288 (54.25 ns) = 15.625 μsec.

which is the same as 2 ms/128 = 15.625 μsec.

For frequencies above 18.432 MHz a similar exercise can be performed to determine the internal refresh rate. Note that these internally generated refresh cycles are generated religiously if you have power and a clock supplied to the 8202. Regardless of reset, processor being in a hold or halt state, or even DMA transfers, the 8202 is going to refresh memory when it needs it.

External Refresh

An input is provided (REFRQ/ALE) that allows a user to externally command the 8202 to perform a refresh. This input is high level sensitive and only requires a 20 ns pulse width for a request. After the refresh cycle is started the internal refresh counter is reset, but it is not

disabled! This feature guarantees data integrity in the memory array if the external requests fail to occur, such as might be the case in a processor controlled refresh system. To use the external refresh pin to *totally* avoid having an asynchronous, internal refresh requests that may generate random wait states, you must provide the 8202 with an external refresh pulse more often than the 8202 will want to do the refresh. This rate is dependent on the clock frequency of the 8202 and can be calculated with the *min* equations in the last section.

Hidden or Sync Refresh

A user may want to squeeze out a little more processor efficiency by doing an external refresh that is hidden in opcode decode. If not enough time is available during opcode decode, and program memory is not in dynamic RAM, the whole fetch cycle can be used to refresh RAM by pulsing the REFRQ/ALE pin when fetch occurs. Refreshing as often as an opcode fetch occurs has the advantage of gaining a little processor efficiency, with the disadvantage of increasing memory array power usage. The designer should trade off the relative importance of these two aspects when considering external or sync refresh.

Addressing and Control to the System Bus

Processor addressing to the 8202 is simple and direct, with the 8202 taking care of the address multiplexing necessary for the dynamic RAM. To illustrate the addressing schemes, a few examples are given. In Figure 4 the simplest is shown, mapping a 16 bit address bus onto 64K of memory.

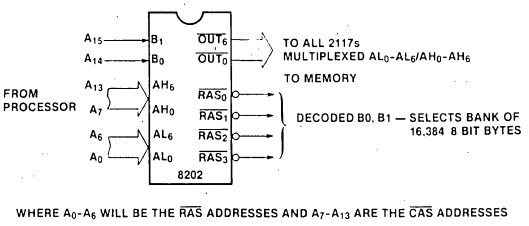


Figure 4. 8202 Addressing

With this simple configuration, all the address lines from the 8080A bus or 8085A demultiplexed bus are presented to the 8202 for 64K bytes of dynamic RAM. If using this addressing scheme, it would be necessary to memory overlay the program memory to start the processor. With this simple method of addressing, memory will be organized as shown in Figure 5.

If the system requires it, the addresses to the 8202 can be scrambled *anyway* desired (be careful with B_0, B_1). This may seem confusing at first but RAM means what it says, *R*andom *A*ccess *M*emory! Many RAMs don't have a logical mapping of address space even internally. Even though the program will go through consecutive locations, scrambling the addresses to the 8202 will present no problems as it will map it into the dynamic RAM. You may have noticed that the output addresses are inverted from the inputs. Taking advantage of the irrelevance of the addresses, the 8202 inverts its outputs to provide for faster operation. When isolating a *physical* location in RAM (software only sees relative addresses) you must take this into consideration with any address scrambling you may do. To do refresh efficiently, however, you must step through the lower $\overline{RAS}$ addresses of the RAM to refresh every row. Again the user need not be concerned as the 8202 automatically does this.

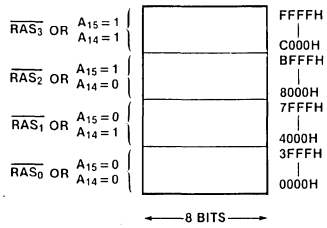


Figure 5. 64K 2117 Memory Organization

If not all the memory is wanted, say only 48K bytes instead of 64K so that program memory (such as ROM or EPROM) could reside in 0000H-3FFFH, all that is needed is to gate the prohibited memory space of A_{14} and A_{15} and input this to the 8202 PCS pin (Figure 6). This $\overline{PCS}$ pin is a Protected Chip Select pin. When it is low, $\overline{WR}$ and $\overline{RD}/S_1$ inputs are enabled. If high, it will not recognize any read or write cycles, but it is protected against terminating a cycle in progress and allows any future refresh cycles to occur. As in this case mentioned here, its purpose is to disable the 8202 when accessing the first quadrant of address space. With the $\overline{PCS}$ pin you can segment your memory as desired by decoding the proper enabling signal.

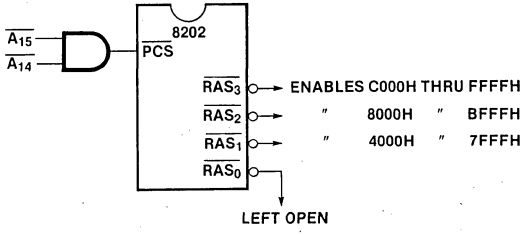


Figure 6. Deselecting during Prohibitive Address Space

The control interface to the 8202 is just as simple. In most applications all that is needed is a read or write command. The falling edge of these signals start the memory cycle to be performed. During write cycles it is up to the user that data is provided at the proper time to the memory array. This is discussed in more detail in the application and processor interface sections.

The 8202 generates two control outputs in addition to all the dynamic RAM controlling signals. One, which has already been discussed, is $\overline{SACK}$ (System ACKnowledge). The other signal provided is called $\overline{XACK}$ and can be used to latch data buffers with the information from the dynamic RAM. This $\overline{XACK}$ signal disappears after 2 clock cycles if read or write and $\overline{PCS}$ terminate beforehand, otherwise it remains low until read or write and $\overline{PCS}$ disappear. $\overline{XACK}$ can also be used as an alternative $\overline{SACK}$ to guarantee that wait states occur in some specific designs.

$\overline{SACK}$, $\overline{XACK}$ and READY

This is one area that is easily confused and requires special attention. $\overline{SACK}$ (System ACKnowledge) is a system concept that is similar to a DMA acknowledge when a DMA request is made in a system with DMA. If a memory cycle request is made, the 8202 will respond on $\overline{SACK}$ when it is available for use. $\overline{SACK}$ signifies when a cycle request has been accepted. It is delayed when there is a refresh cycle in progress until the refresh is complete. $\overline{SACK}$ *does not* tell you whether or not wait states are needed to be access compatible with the memory! Since $\overline{SACK}$ is gated into the READY input of the processor (this is more efficient than a request/request granted type of structure) you must make sure you don't mix the two concepts of acknowledge and memory wait states.

How to treat $\overline{SACK}$ can divided into two topic areas for its understanding; one, when a memory wait state is not needed to guarantee memory access and two; when it is.

1. Wait state not needed

To guarantee a wait state free environment (except during refresh) $\overline{SACK}$ must return in time to guarantee READY set up at the processor. $\overline{SACK}$ will only respond after both $\overline{PCS}$ and a control signal has been presented to the 8202. If the set up time is not met, you may or may not have a wait state. To avoid this you may want to try to use advanced controls to gain additional response time.

The common configuration that has been shown up until this point in time for $\overline{SACK}$ is to NAND it with other system acknowledges to the READY input of the processor. This concept only works if every addressable adress space has associated with it a system acknowledge, which would be common in a general bus environment. If this is not the case, a little care must be taken for the logic on the READY input. Take for example:

Single board, a few memory mapped peripherals, ROM and RAM, with the only system acknowledge coming from an 8202

If a user was to take this case and simply invert (the "logical" thing to do with only one $\overline{SACK}$) $\overline{SACK}$ into READY, he would get wait states till ad nauseum when any address space except the 8202 was accessed. The user may then be tempted to not invert $\overline{SACK}$ and input this into READY. Sorry, but this will not perform the function properly either. The correct way is shown in Figure 7 where PCS is OR'ed with positive $\overline{SACK}$ (or by using DeMorgan's theorem, $\overline{PCS}$ can be inverted and NANDed with $\overline{SACK}$). Since READY is active high, the processor will always see the address space as "ready" (because $\overline{PCS}$ will be high) until it accesses the 8202 address space. At this point in time, READY will not be active until $\overline{SACK}$ returns.

2. Wait state is needed to be memory access compatible

If $\overline{SACK}$ returns in time to guarantee ready setup and a wait state is needed, it will be necessary to delay $\overline{SACK}$ or gate it with some form of wait state generator. Example schematics of wait state generators for the various Intel processors are published in the various Intel Ap Notes and User Manuals. Another technique would be to use $\overline{XACK}$ (Transfer ACKnowledge) instead of $\overline{SACK}$ to produce the refresh and memory wait states needed. Depending on the buffering delays, processor and 8202 cycle speeds, $\overline{XACK}$ may introduce more than one wait state.

If $\overline{SACK}$ does not return in time to guarantee READY set up, it is sometimes very tempting to use this as your wait state generator. *Don't*! This will *not* always guarantee a wait state. READY set up is specified as a minimum for guaranteed recognition, not a maximum for guaranteed nonrecognition. Again $\overline{XACK}$ (in addition to its duty of latching data buffers) could be the answer for a simple low cost wait state generator. The same discussion in the previous section about not having $\overline{SACK}$s associated with the entire memory space also applies here. (If you had trouble understanding this section, reread it, it will be well worth your time.)

Clocking the 8202

The 8202 can be clocked in a variety of ways. The crystal inputs can be used with series resonant, fundamental crystals up to 25 MHz. The specifications for this crystal and their explanation are given in Intel Application Note #35, Crystals: Specifications for Intel Components. When working with such high frequencies, certain precautions should be observed. The crystal should be as close to the 8202 inputs as possible to reduce the parasitic effects associated with leads and board traces. The crystal holder or "can" should be grounded to the ground plane if possible. A trimming decoupling capacitor on the order of 10 pF is helpful to prevent DC across the crystal and trim the frequency.

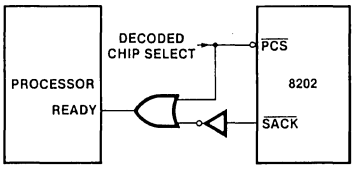


Figure 7. Ready Logic when Not All Address Space has $\overline{\text{SACK}}$s

Tying the X_0 input to + 12V through a 1K ohm resistor allows for operation of an external clock driver. Again this clock driver should be as close to the 8202 as possible! If it is necessary to bus the clock signal to other components, you will want to buffer it (with a buffer that can easily handle 25 MHz) and possibly devote a layer on your board just for the clock to prevent clock and signal interaction. Unlike other NMOS components, the 8202 clock input can be driven directly with TTL levels (i.e., no pullup resistors) when this option is selected.

Power Up Reset

The 8202 is designed such that no external reset is necessary. To understand this, it is helpful to look at the five groups of logic which could have required a reset, as listed below:

1. Refresh timer logic
2. Refresh address logic
3. $\overline{\text{XACK}}$ and $\overline{\text{SACK}}$ logic
4. Control timing logic
5. Cycle start logic

1. The refresh timer can't count more cycles than it is set for. Therefore, after power up, a refresh cycle will occur within the required amount of time for the dynamic RAM. If it is desired to provide some synchronization with external refresh (to insure that an internal refresh will not occur in a wait state free system, for example) pulsing the REFRQ refresh request input will cause a refresh cycle to start and reset the timer.

2. In reference to the refresh address logic, dynamic RAM does not care which row in the array is refreshed first, just as long as all of them are before 2 ms. As a result, it doesn't matter as to what location is used for a starting point after power up. However, pulling both read and write inputs low at the same time will cause the refresh address counter to reset to zero. This may be useful for testing purposes but it is *not* recommended for normal operation.

3,4,5. $\overline{\text{XACK}}$, $\overline{\text{SACK}}$ and control timing logic are all reset when $\overline{\text{RD}}$, $\overline{\text{WR}}$ are both high and no cycle is requested. After power up, three memory cycles amount of time must elapse before a valid memory cycle can occur. This is due to the reset nature of the cycle start logic. The cycle start logic consists of several flip-flops, each of which is able to request a start of a cycle. If one of these is set at powerup, a cycle will start. Hence, it is possible that three "memory" cycles could occur, an internal and external requested refresh and a pseudo memory cycle. At the end of these pseudo initial cycles, the control timing and cycle start logic will reset itself and become available for normal use.

MEMORY SPEED SELECTION AND LAYOUT

Compatible Memories

The 8202 was defined to give the widest possible range of compatibility with memory devices, as will be noted shortly. This section strictly discusses the 8202 compatibility with the dynamic RAM, not the processor compatibility. This will be dealt with in an upcoming section.

An exhaustive approach can be performed to show the extent of compatibility of the 8202 to the actual memory device by comparing each individual specification. This has been done for you in defining the 8202 (and is reproduced in Appendix 3) with the Intel industry standard 2104A and 2117 dynamic RAM parts.

TABLE 1

MEMORY	$\overline{\text{RAS}}$ ACCESS	$\overline{\text{CAS}}$ ACCESS	8202 compatible freq
16K (2117)			
2117-2	150 ns	100 ns	18.432 MHz–25 MHz
2117-3	200 ns	135 ns	18.432 MHz–25 MHz
2117-4	250 ns	165 ns	18.432 MHz–22.222 MHz
4K (2104A)*			
S6047	150 ns	100 ns	18.432 MHz–25 MHz
S6048	200 ns	135 ns	18.432 MHz–25 MHz
S6049	250 ns	165 ns	18.432 MHz–22.222 MHz

***NOTE:** Do not use 2104A-1, 2, 3, 4 with the 8202. The 8202 does a $\overline{\text{RAS}}$ only refresh which requires the use of the S6047-6049 speed selections.

Since the 8202 specs, for the most part, are frequency dependent, equations can be set up to determine the lowest and highest frequency possible for the compatibility of the specs. Doing this, the 8202 frequencies (at the X_0, X_1 clock inputs) can be obtained for the above dynamic RAMs and the results are summarized in Table 1. The upper limit of 25 MHz is the frequency limit of the 8202 itself, not the dynamic RAMs. The lower limit of 18.432 MHz is established by the refresh rate of the 8202. At 8202 frequencies below this limit, the dynamic RAM will not be refreshed adequately, as it will be below the 128 row/2 ms or 64 row/2 ms rate necessary, respectively.

With the 2117-4 and S6049, the upper frequency limit is established by the t_{RAH} (row address hold time). To illustrate the frequency cut off due to this spec, let's calcu-

late the allowable 8202 frequency range using t_{RAH} as the limiting factor:

2117-4 t_{RAH} is 35 ns

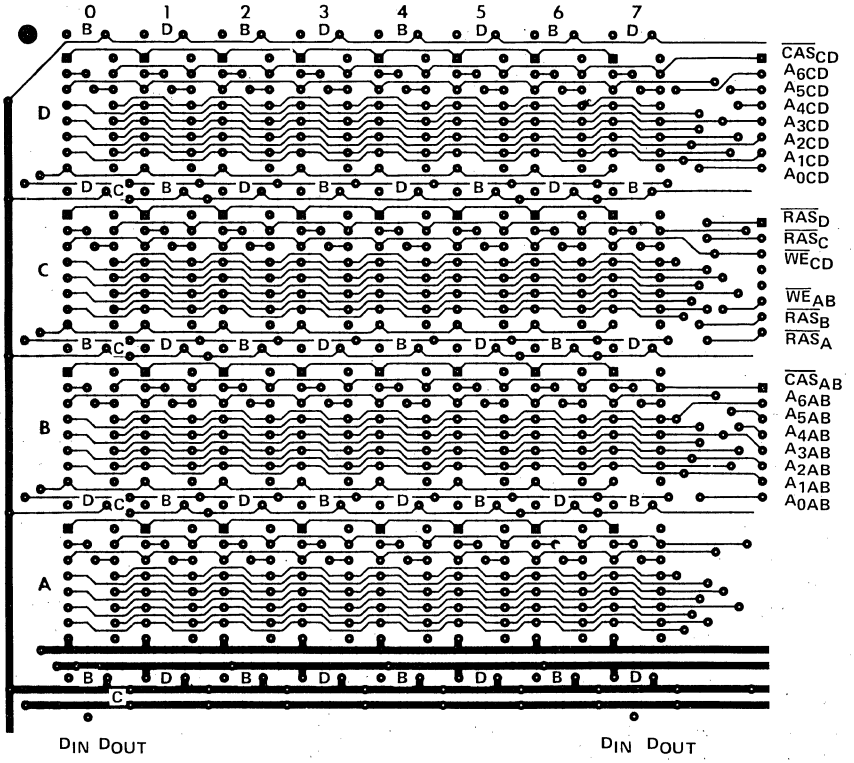
Setting the 8202 min spec $t_{RAH} = t_P - 10$ (where $t_P = 1/8202$ clock frequency) the upper limit can be established.

$t_P - 10$ ns = 35 ns
$t_P = 45$ ns Therefore f8202 upper = $1/t_P = 22.222$ MHz

You may want to check a current data sheet and re-establish this limit (check other parameters also). Note that this range of 18.432 MHz–22.222 MHz allows you to work with slower memories and includes the frequency 22.1184 MHz, a convenient source for BAUD rates. (There are also 2109s available which have the same pin-out as the 2117 but are 8K × 1 memories.) Again, please note that this is the frequency range of compatibility with the 8202, not the processor. Noting that specs change from time to time, a complete analysis is given in Apendix 3. Take the current specifications that you have for your memory component and compare them to the ones used in Appendix 3.

Layout, Decoupling

The need for adequate decoupling when working with dynamic RAM cannot be overstressed. Because of its dynamic nature, internal circuitry is turned off when deselected to limit the use of power. Large switching currents occur when turning on a group of dynamic RAMs all at once, which must be furnished by the power supply and bypass/decoupling capacitors. Any memory cycle will turn on at least 8 components, with a refresh using up to 32 components. Therefore, large bulk decoupling capacitors should be used to prevent power supply droop, with many small capacitors to filter out the high frequency noise by minimizing the current spiking as much as possible. Reproduced in Figure 8 is a suggested 2117, 2104A layout (with 2104A, AH6 is $\overline{CS}$). To make this layout compatible with the 8202, the $\overline{CAS}$, $\overline{WE}$ and address lines should be tied together to form one $\overline{WE}$, $\overline{CAS}$ and set of address lines.



DECOUPLING CAPACITORS
B = 0.1 μF TO V_{DD} TO V_{SS}
D = 0.1 μF V_{BB} TO V_{SS}
C = 0.01 μF V_{CC} TO V_{SS}

SAMPLE P.C. BOARD LAYOUT EMPLOYING VERTICAL AND HORIZONTAL GRIDDING ON ALL POWER SUPPLIES

BOARD ORGANIZATION: 64K WORDS BY 8-BITS

Figure 8. Recommended 2104A, 2117 Array Layout

Figure 8 also illustrates the proper method for gridding and placement of decoupling for a dynamic memory board. Locations B, C, and D in the array are strategic locations for placing capacitors to minimize noise to the memories. Gridding of the power supplies, not normally done with standard TTL boards, usually requires a multilayer board (double layer shown). Gridding is imperative with MOS dynamic RAMs. Gridding helps lower the effective impedance of the supply bus and works toward eliminating power supply drops throughout the array. (Refer to primer in Appendix 2.)

The 8202 itself should be decoupled by capacitors. It is recommended that three capacitors (the bulk capacitor could be at where the power enters the board) be used, as close to the component power pin as possible, with the following values:

Quantity	Value
1	4.7 microfarad (for supply droop)
1	.1 microfarad (for filtering)
1	.001 microfarad (for filtering)

It is necessary to have these capacitors to insure proper operation of the 8202, especially under heavy loading conditions. In addition to the decoupling capacitors, it is sometimes helpful to use series resistors with the 8202 outputs to reduce reflections and match impedance levels with that of the dynamic RAM. There already exists 20 ohms internal in the 8202 for this purpose.

Checking Out the Memory Array

After you blessed out the tech and draftsmen for the layout mistakes and wiring errors, it's time to see how well you did your homework. The first step is to determine if a simple write then read can be performed. If this doesn't happen correctly, you usually haven't found all the wiring errors. Afterwards, a simple test can be performed to determine if you have any gross noise or fundamental problems.

An example test pattern that does functional and some level of address checking is a checkerboard/checkerboard bar pattern. This pattern, which is an alternating 1's and 0's test, can check if there is a "stuck" bit, if the addressing is stepping from one location to another and basically if the entire memory array is functional. Writing all 1's or 0's will not be sufficient as addressing is not really checked in this case. A flow chart for this type of test is given in Figure 9. This chart steps through the entire memory by first writing the pattern and then reading, comparing the read to what was written. If there is an error, both the address and erred data are stored in static RAM. (If you are not using the full memory space). If you are using the full address space, you may want to check out the array in sections. A second pass can be performed with the opposite data pattern.

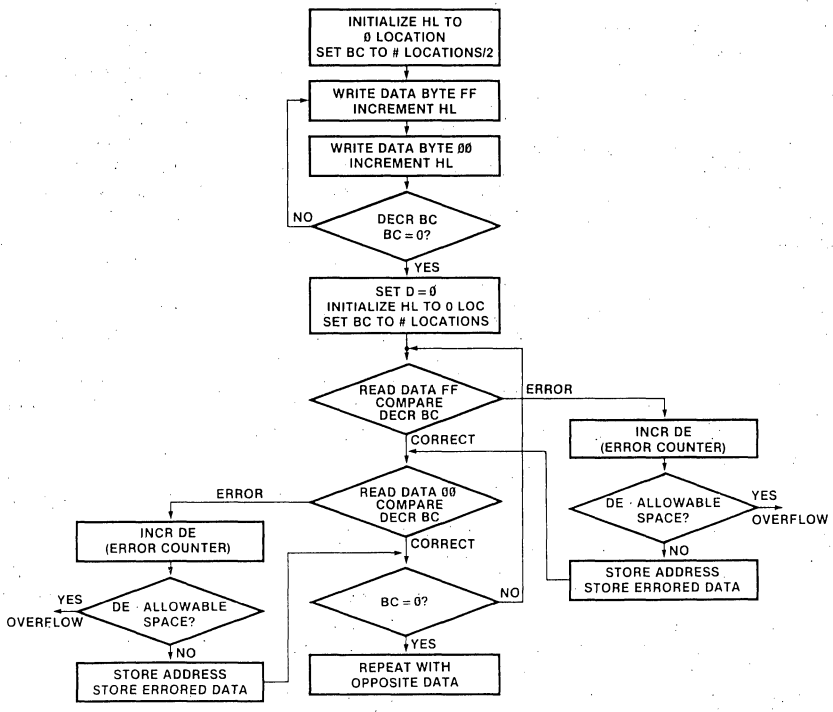


Figure 9. Example 8080A, 8085A/8202 Test

There are several items that must be taken into consideration when interpreting the information obtained from the above test. One is that the data byte written covers 8 components, making it imperative to determine which bit corresponds to which column of components. If there is one totally failing RAM, it will cause 16,384 bytes to be wrong (if using 16Ks). The other consideration is the addressing scheme. Remember that $\overline{RAS}$'s are decoded from the B_0, B_1 inputs and the output addresses of the 8202 are *inverted* from the input. Therefore the software addresses are not the same as your hardware addresses. With these considerations in mind, let's continue on to some helpful hints as to where to look if problems arise.

No errors — Great! Time for a beer. But the true test is over temperature, supply tolerance and time under random read/write conditions. If you are using Intel components, they were thoroughly tested individually before they were shipped to you.

Single bit — This may be the result of a bad component, but may not. Retest the memory to see if the error repeats. If so, *move* the failing component to another location and see if the error follows. If it doesn't, you have a random error. If it does, replace the component and try again.

Random errors — First off, many errors that are claimed to be random really aren't. Obtain the total information of failures (even if they are different from one pass to another, work with one or two passes) and determine if there is any consistency. A few helpful hints can be followed to isolate most problems.

(1) Is there a consistent address involved? Most problems of this nature are due to noise on the address lines. By using a scope, determine if the address levels are met before the required set up time to $\overline{RAS}$ and $\overline{CAS}$. Is there ringing in the address lines? Series damping resistors on the order of 10 ohms can help reduce this problem.

(2) Make sure that all address and control lines make it to the RAM array and are not shorted together anywhere. Surprisingly, this is often a source of random errors.

(3) Ground noise may be the problem. Look at this with the scope grounded properly. Make sure ground is well gridded as suggested in Appendix 2. After noise levels are cleaned up, try again.

(4) If the errors are truly random, you will want to look at the V_{DD} bus in dynamic RAM array. Is it free from droop (especially during refresh) and severe noise spikes? Supply droop can be minimized by using large bulk decoupling capacitors. Also look at the V_{BB} and V_{CC} lines. Intel 12V dynamic RAMs are relatively insensitive to V_{CC} as only the output buffer is connected to the +5V supply. V_{DD} is the main supply (this is not true with the 2118). V_{BB}, however, is connected to the substrate of the device and affects operation of every transistor in the RAM device. It is, therefore, very important that V_{DD} and V_{BB} noise is minimized. Power supply gridding (as shown in the Appendix) and liberal use of strategically located capacitors will help you here. Contrary to 12V dynamic RAMs, the 8202 is driven from the V_{CC} line. It is just as important that noise here is reduced. Make sure the signals entering the 8202 are buffered if necessary and the 8202 is well decoupled. Referring to Intel's Memory Design Handbook may also provide some clues to the problem.

With the possibility of 1 megabyte systems with the 8086, it sometimes becomes attractive to use some of the techniques that large mainframe computer manufacturers use to correct random errors. One such technique is error correction coding. Through clever coding it is possible to isolate a bit in error and correct it, in significantly less bits than the data byte itself. For complete coverage of these techniques you may want to refer to a textbook on the Hamming Code for a start. Also, Intel will be publishing an Application Note on ECC in the near future.

APPLICATION EXAMPLES

8085A

The processor interface to the 8202 should also be taken into consideration when selecting a memory device. The 8080A interface will not be covered, as this component has considerable technical documentation and is widely understood by the industrial market. It is, however, worthwhile to consider the higher performance 8085A and the recently announced 16 bit 8086 microprocessors. It is interesting to note that the 8202 coupled with these processors provides one of the the lowest cost, highest performance dynamic RAM solution possible on the market. Another solution has been processor controlled refresh, which is not dependable in a system environment where the bus can be tied up for long lengths of time. With this in mind, let's look at the 8085A interface with the 8202.

In the 8085A system you must be conscious of two things. One is that data is guaranteed to be valid 40 ns after the leading edge of WR (20 ns in 8085A-2). Secondly, READY needs to be set up prior to $\overline{RD}$ or $\overline{WR}$ control signals going low for the 8085A to work without wait states (check data sheet).

Looking at the 8202 data sheet, the 8202 responds on its $\overline{SACK}$ line $40 + t_P$ ns (1/8202 frequency) after a memory cycle is requested. What does this mean? This means that if you use $\overline{RD}$ or $\overline{WR}$ from the 8085A for the 8202 controls, you may or may not have 1 wait state inserted by the $\overline{SACK}$ line coming back. To guarantee that a wait state will be included, a wait state generator could be included locally on the processor board, which assumes a specific and dedicated system environment, or on the slave board, assuming a general bus such as the Intel MULTIBUS. The $\overline{SACK}$ (or $\overline{XACK}$ could be used) line from the 8202 will still have to be involved so that extra wait states, if needed, will be introduced when the 8202

is performing an internally initiated refresh. With a few simple tricks (or one that the 8202 provides) you can avoid wait states on read or both read and write cycles if desired.

To determine the extent of compatibility of the 8202 and dynamic RAM with the processor, an extension of the spec comparisons can be done. This analysis should be done to determine the optimal operating frequency of the 8202 and the processor. The timings that should be checked are as follows:

1. Read access
2. Data valid to write
3. Cycle time
4. Ready set up
5. Turn off delay

When determining read access for the memory, you will find in most cases that t_{CAC} is the relevant parameter instead of t_{RAC}. To describe further:

Read Access

After subtracting off the buffering delay (to and from memory) and the 8202's control to $\overline{CAS}$ delay, you will have the amount of time available for $\overline{CAS}$ access. Compare this to the 2117 data sheet to initially select the unit. The 5V 2118 offers some higher speed selections.

Data Valid to Write

If using advanced write signals to help eliminate wait states, you must take into consideration when write data will be valid. Dynamic RAM, unlike static and most peripherals, wants data to be valid at the leading edge of $\overline{WE}$.

Cycle Time

In most instances, cycle time will be no problem, unless you start the cycle late in the CPU cycle. Take into consideration how quickly other bus masters will be able to get ahold of the bus and request a cycle.

READY Set-Up

READY timing is tricky. If the $\overline{SACK}$ line coming back does not meet the minimum timing of the processor READY set up, it does not mean that you have a guaranteed wait state. As mentioned with the 8085A, if the memory requires a wait state to be access time compatible, you must insure that a wait state occurs. This can be done with some logic (see Application Note #38) or $\overline{XACK}$ oftentimes will provide the necessary signal.

Turn Off Delay

Will the memory let go of the bus in time for the processor to use it? If not, a buffer could buy you the extra time you need by isolating the memory bus when the system bus needs to be floated.

8085A Interface Examples

There are many options with interfacing the 8202 to the 8085A microprocessor. Each option has its merits in performance versus cost tradeoff. Hopefully without introducing confusion, several of these options will be illustrated to provide the designer a range of choices.

These examples are discussed below with the relative tradeoffs given in Table 2.

TABLE 2

Performance	Cost
Example 1	
10-20% reduction in processor speed (software dependent) due to wait states Up to 64K bytes addressable space Internal failsafe refresh	0 Hardware overhead Allows full 8202 compatible range for memory. Therefore low cost 2117-4 can be used. Plenty of margin for buffering
Example 2	
Eliminates all read wait states (in most systems, the vast majority) except when refresh occurs Up to 32K bytes of address space Internal, failsafe refresh	0 Hardware overhead Full 8202 compatible range possible Some margin available for buffering
Example 3	
Same as 2 except 64K address space	1 inverter, 1 D flip-flop, otherwise same as 2
Example 4	
Same as 3 except no write wait states	3 flip-flops, 5 gates, otherwise same as 2
Example 5	
Virtually nonexistant wait states Internal and external refreshing used	2 flip-flops + couple gates, otherwise same as 2

Example 1 — Figure 10 illustrates a direct interface to the 8085A. 64K bytes of memory are shown but one 16K block could just as easily been 2109's to allow for 8K of program memory space (ROM, EPROM) with 56K bytes of dynamic RAM. This system has at least one wait state with every read and write (note the use of $\overline{XACK}$ instead of $\overline{SACK}$). Since internal refreshing is used in this schematic, wait states will also be interjected when a memory cycle is requested during a refresh. If one was to assume that every transfer involved the 8202 and the average instruction was 7 clock cycles, a reduction in processor efficiency of 13-15% could be expected. For most applications this is not a detrimental factor.

Example 2 — To eliminate read wait states (which are usually the vast majority of the cycles that are performed) and still have a direct interface, one can take advantage of some of the 8202 options. An advanced read can be done by tying $\overline{RD}/S_1$ input to the 8085A S_1 output and latching it with ALE on the REFRQ/ALE input. To use this option B_1/OP_1 needs to be tied to +12V through a 5K ohm resistor. Since this input is used to enable two of the four $\overline{RAS}$ lines, two of them will be disabled ($\overline{RAS_0}$ and $\overline{RAS_1}$ as the 8202 will interpret B_1/OP_1 as a high), halving the available address space. If halts can occur in your software, you may want to qualify S_1 with $IO/\overline{M}$ so that reads do not occur in a HALT state (see 8085A data sheet for $IO/\overline{M}$, S_1 and S_0 decoding). An advanced write cannot be generated the same way as data will not be valid at the RAM array before the $\overline{WE}$ from the 8202 is presented. A little bus analysis can convince you of this point.

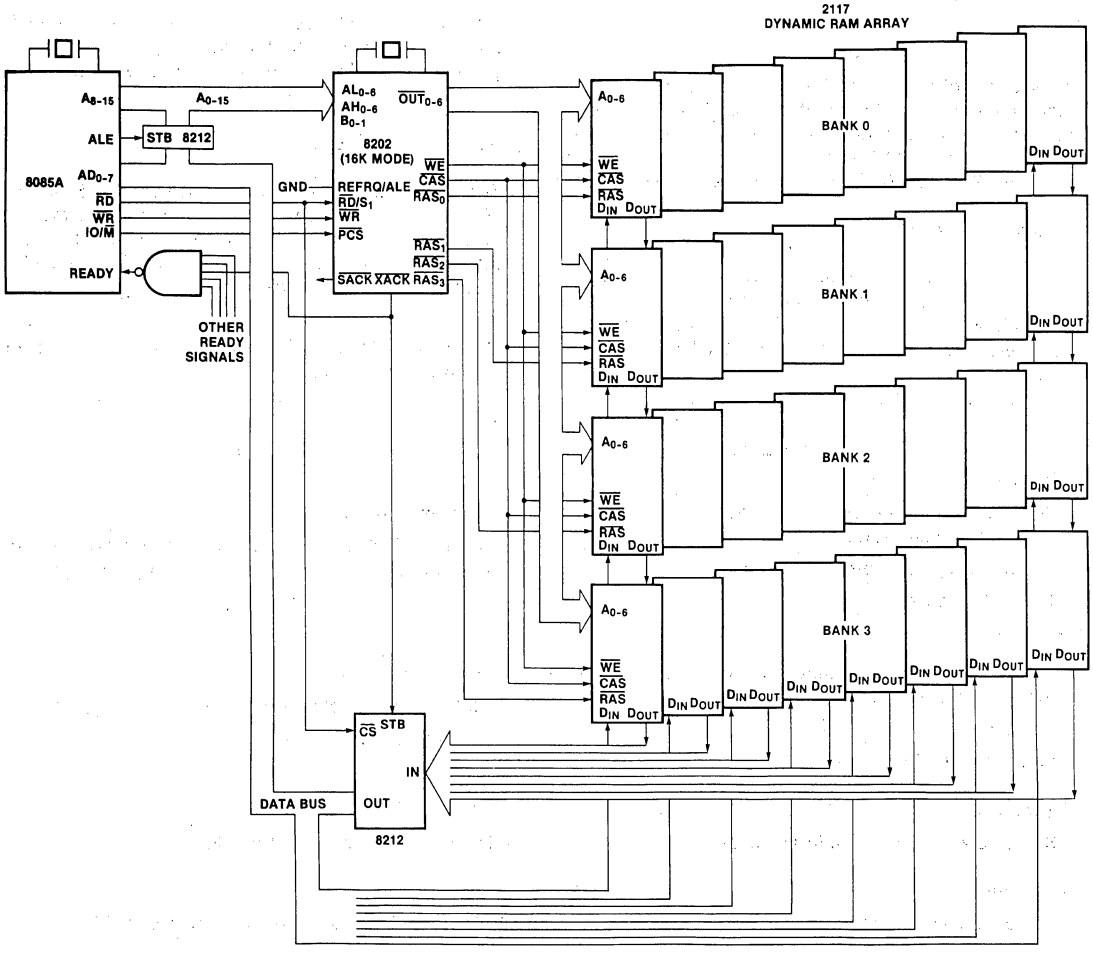


Figure 10. 8085A — 2117 Interface

Example 3 — Obviously, some people would prefer to have advanced read and the complete memory space. This is accomplished easily enough by generating the advanced read externally as illustrated in Figure 11. For the price of one D flip-flop, the entire memory space is available with no read wait states (if memory access is compatible with the processor access). You will also note that when using advanced read, $\overline{SACK}$ is shown as the READY input versus $\overline{XACK}$. With advanced read it is possible (depending on the amount of buffer delay involved) to be memory access compatible. Since the write cycle is still being initiated by the 8085A $\overline{WR}$, there may or not be a write wait state (as READY is not guaranteed to be set up). Make sure that the write timings will be compatible with the memory you are working with, with or without a wait state.

Example 4 — For doing an 8085A write without wait states, a little more finesse is required. To understand the following approach, it is necessary to understand a particular function of the 8202. The 8202 looks at its $\overline{WR}$ and $\overline{RD}$ inputs to know when to start a memory cycle. If $\overline{WR}$ is the requestor, a memory write cycle is started. At the proper time, the 8202 will allow $\overline{WR}$ to pass through as $\overline{WE}$ to the dynamic RAM. If one was to shut off $\overline{WR}$ to the 8202 after the cycle has started and then give it a write again after data is valid (in other words, a double write pulse) he can delay the $\overline{WE}$ to the dynamic RAM*. You've got to be careful when doing this. One problem that might occur is that instead of a write cycle, a read-modify-write cycle could get started. This will end up with an aborted read. Dynamic RAM doesn't care about this as long as you perform a write to the same location within the same cycle and don't expect to have read valid data. From a system point of view, the data out and data in lines of the 2117 (or any output floating memory) could no longer be tied together as data out is no longer guaranteed to be tri-stated. If you are considering this approach and are confused about this last point, think about it for a while. Using this concept allows one to avoid wait states in an 8085A write. To summarize, this is accomplished by one; giving the 8202 an advanced write so that $\overline{SACK}$ will return in time for READY and two; delaying $\overline{WE}$ to the array until data is valid.

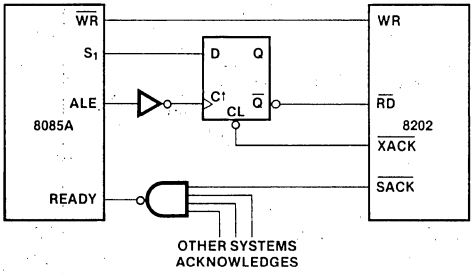
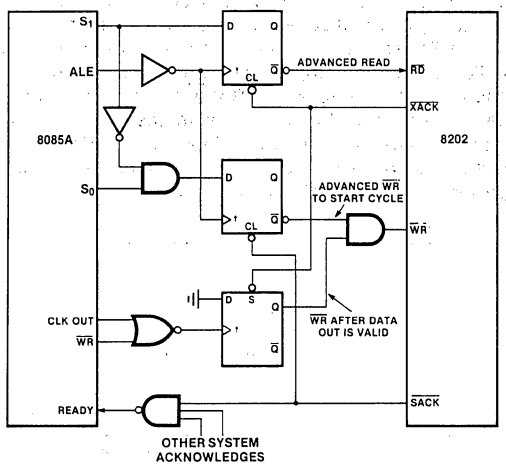


Figure 11. 8085A Interface — No READ wait states, wait states during WRITE and REFRESH

*Another technique would be to ignore the WE from the 8202 and provide your own when data is valid.

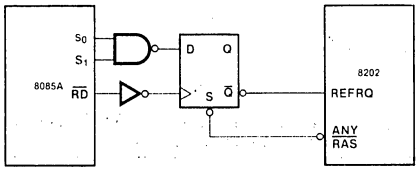
A circuit illustrating this technique, along with advanced read, is shown in Figure 12. Note that an advanced write cycle is started when S_0, S_1, and falling edge of ALE occurs and is cleared as soon as $\overline{SACK}$ acknowledges it. The normal $\overline{WR}$ is delayed by CLKOUT (until output data is valid as the 8202 propagates a $\overline{WE}$ very quickly) and provides the $\overline{WE}$ to the dynamic RAM at the proper time. The footnote in Figure 12 is particularly important. If you can be guaranteed that no wait states will occur due to internal refresh, you can ignore the $\overline{SACK}$ line. If memory access, cycle times are met, it is not necessary to generate an advanced write to avoid write wait states, as ready will no longer be controlled by $\overline{SACK}$. Before making this assumption, determine if reset, burst DMA, or other lengthy bus holds can occur in your system.

Example 5, Hidden Refresh — As mentioned before, it is possible to do a hidden refresh that will increase CPU efficiency as wait states will be reduced. An example of an external refresh that is in sync with opcode decode is shown in Figure 13.



NOTE: IF YOU CAN BE *GUARANTEED* THAT EVERY REFRESH WILL BE AN EXTERNAL REFRESH, YOU CAN IGNORE THE SACK LINE IN THE ABOVE CONFIGURATION AND USE WR DIRECTLY FROM 8085A.

Figure 12. 8085A — 8202 Interface with No Read or Write Wait States



HIDDEN SYNC REFRESH WITH INSTRUCTION DECODE OF OPCODE FETCH. CAN BE USED WITH OR WITHOUT ADVANCED READ.

Figure 13. Hidden Sync Refresh

This schematic illustrates using the S_0 and S_1 lines from the 8085A and gated into a D flip-flop to request an external refresh. When both S_0 and S_1 are high (which denotes a fetch cycle is occurring) the D flip-flop will be able to clock in the refresh request. Note that the leading edge of read is used to clock the D flip-flop. This allows fetch to occur from dynamic RAM with an external refresh cycle immediately afterwards during opcode decode. This external refresh request pulse can occur simultaneously with the memory cycle request. Using the leading edge of the read from the 8085A will insure that the memory cycle will start before requesting a refresh cycle. (Remember refresh requests are latched until serviced). Note that this is only necessary if instructions reside in dynamic RAM. If program memory resides in static RAM, ROM or EPROM, the entire fetch cycle could be used to perform a refresh. This can be implemented in a similar fashion as Figure 13, except ALE could clock a negative edge triggered flip-flop instead of RD in Figure 13. A $\overline{RAS}$ from the 8202 is a convenient signal that can be used to clear the refresh request flip-flop.

Example 5, DMA — A DMA controller interface to the 8202 is very similar to the processor interface. An example giving a partial schematic is shown in Figure 14 with the 8085A. The control lines from the 8257 could be wire OR'ed to the demultiplexed control lines of the 8085A and gated to the 8202. The $\overline{SACK}$ signal from the 8202 serves the same function as with the processor and must be gated into the ready input of the 8257-5. This will cause wait states to occur when a refresh is being performed and a DMA transfer is requested.

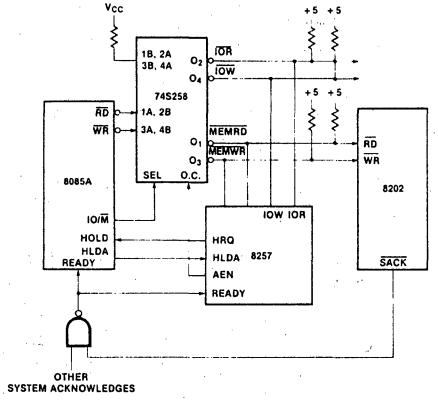


Figure 14. Partial Schematic Showing $\overline{RD}$, $\overline{WR}$ and READY Interconnect in DMA System

Unlike in the 8085A system, the 8257 only requires the ready line to be set a minimum of 90 ns after the read or extended write control line goes low. This means that an automatic wait state will not included on reads or extended writes. (Over certain frequency ranges of 8202, see t_{CA} spec on 8202. Also affected by the clock frequency of the 8202). Therefore you must be careful about DMA read access from the 8202, will it be compatible with the DMA transfer?

Just as important is the write cycle. Remember dynamic RAM wants data to be set up at the leading edge of the write enable signal. If advanced write is used to start the write cycle, data must be set up at the RAM (consider buffer propagation delay) after the cycle is started. After a certain period of time the 8202 will give the dynamic RAM a $\overline{WE}$ signal.

If you are using an external advanced read or write option as shown in an earlier figure with the 8085A, you will want to place a tri-state buffer in between the advanced read flip-flop and the 8202. This is to insure that the lines are floated when the DMA controller asks for the bus. The control lines should be tied to +5 volts through 1K ohm resistor on any input to the 8202 that may float to prevent unwanted cycles.

Pulling together some of the previous examples (1-4) and the discussion on hidden refresh and DMA, example 5 can be shown as a high performance, low cost system. To illustrate, say a system had an 8275 CRT controller, 3 MHz 8085A processor, 8257-5 DMA controller, 8202 and wanted to use 2117-4 dynamic RAMs without wait states. If opcode fetches occur more often than the 8202 will want to do them (see refresh under support section earlier in the Note) you can devise a system using external refresh without write wait states. Figure 15 illustrates only looking at the $\overline{SACK}$ line when a read is being performed. A wait state here could only occur if an internal refresh request happened on the first cycle after a bus hold. Figure 15 also illustrates the use of advanced read and external refresh, so that read, write and the vast majority of the refresh wait states are eliminated.

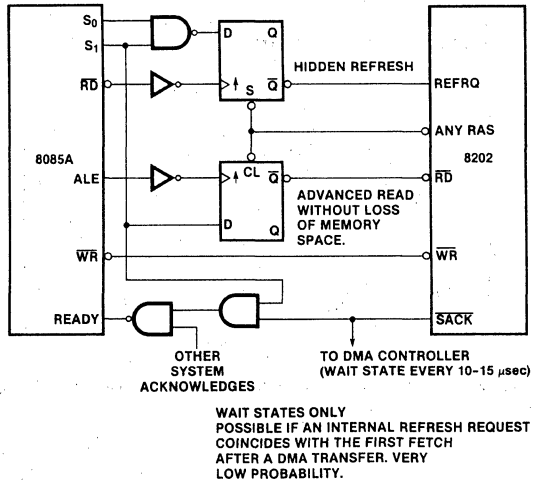


Figure 15. Example 5

8086

The 8086 is a high performance 16 bit microcomputer that has the capability of operating on an 8 bit byte or 16 bit data word. Since the data word is 16 bits long, we could use two 8202's in sync (each controlling xk × 8 bits) or one 8202 controlling 16 bits with the aid of a buffer. The buffer would be needed when trying to drive 33 to 64 MOS devices because of the additional dynamic RAM input capacitance. Up to 64 devices could be used as now the memory will be organized as 64K *words* (or 128K bytes, depending on your point of view). The 2117, 2104A input CAS, for an example of excessive capacitance loading, is a signal that must interface with all the components (versus RAS which only sees a max of 16 components) and is max at 10 pF each, or 640 pF in a 128K byte system. As this exceeds the drive capability of the 8202, this signal must be buffered. Because of the tight timing on address set up to $\overline{CAS}$ it is advisable that the addresses are buffered with the same physical component to maintain roughly the same amount of delay. Address setup to $\overline{RAS}$ will still be guaranteed as long as the designer provides the addresses to the 8202 30-40 ns before the control signal (which is easy in the 8086 system).

In choosing either two 8202s or one buffered 8202, I picked the buffered 8202 to illustrate. Note that this buffering is only needed if more than 64K bytes (or 32K words) are being used, or only one 8202 is being used. For the 128K byte/64K word system shown in Figure 16, buffering is imperative, as noted above.

Referring to Figure 16, you will notice that the 7 address outputs and $\overline{CAS}$ signal are all buffered by a noninverting octal bus driver. The driver shown has a typical propagation delay of 6 ns and an I_{OL} sink current of 68 MA. Distributed buffering could be used instead if desired. Note that you must be careful about not violating any dynamic RAM timing parameters when using buffers. Delaying the address lines too much with respect to $\overline{RAS}$ or $\overline{CAS}$ could possibly violate the address setup time required for dynamic RAM. In addition, any delay of $\overline{RAS}$ or $\overline{CAS}$ will take away some of the available access time for the memory.

Also notice in Figure 16 that the $\overline{WE}$ signal is qualified by $\overline{BHE}$ (byte high enable) or $\overline{A_0}$, (lowest address line on the address bus). This allows for the full software capabilities of the 8086 to be able to write to an 8 bit byte or 16 bit word.

Another consideration in this array is the layout. The 8202 is shown driving the array from the center out. This would allow for better signal distribution to the RAM array. Remember, many noise problems are due to signal noise, any signal being driven from only one side will experience some degradation by the time it reaches the last component.

The schematic shown also illustrates using A_1–A_{16} for the address lines. (A_0 denotes whether writing a high or low order byte. If reading a 16 bit word, A_0 is immaterial). The remaining addresses A_{17}–A_{19} can be decoded by a 1 of 8 binary decoder (such as the Intel 8205) to generate $\overline{PCS}$. Since this application will most likely be off board, A_{17}–A_{19} should be double buffered along with the rest of the addresses, data, and control bus. These double buffers must be taken into consideration when determining the memory device selection and operating frequency of the 8202. To interface this memory array to the 8086, I chose the 8086 max mode system as the example to illustrate.

The 8086 max mode system combines some of the features of both the 8080A and 8085A families and adds some of its own. It has a multiplexed address/data bus like the 8085A, which is demultiplexed when working with dynamic RAM. Also, in the 8086 max mode, a system controller (8288) provides for a set of MULTIBUS compatible control signals.

Figure 17 illustrates how the max mode 8086, running at 5 MHz, can be interfaced to the memory array shown earlier. This figure illustrates a general bus environment (with partial MULTIBUS signals given) where the memory board generates its own chip selects and everything is double buffered. This schematic allows for optimal system flexibility with reduced system cost through the use of inexpensive dynamic RAMs. Access time compatibility is guaranteed through the use of $\overline{XACK}$ requesting wait states from the 8284 clock generator. Wait states in the 8086 family do not reduce processor efficiency as much as other processors as the 8086 has an instruction queue that is constantly making full use of the bus. This queue takes advantage of instruction decode and long internal instructions to fetch additional instructions and acts as a fast "cache" memory for the 8086.

The configuration shown in Figure 17 has two wait states on normal read and write cycles. This can be reduced to one by using $\overline{SACK}$ instead of $\overline{XACK}$ and some gating logic to delay it to the 8284 until one wait state occurs.

Using the general bus concept, it would be easy to expand the address space of the 8086 to its full capabilities of one megabyte. If all of this was dynamic RAMs, it would only take 8 memory cards, each with one 8202 and 128K bytes of RAM and one processor card. A processor as powerful as the 8086 with one megabyte of memory in about the space of a bread box, almost frightening, isn't it?

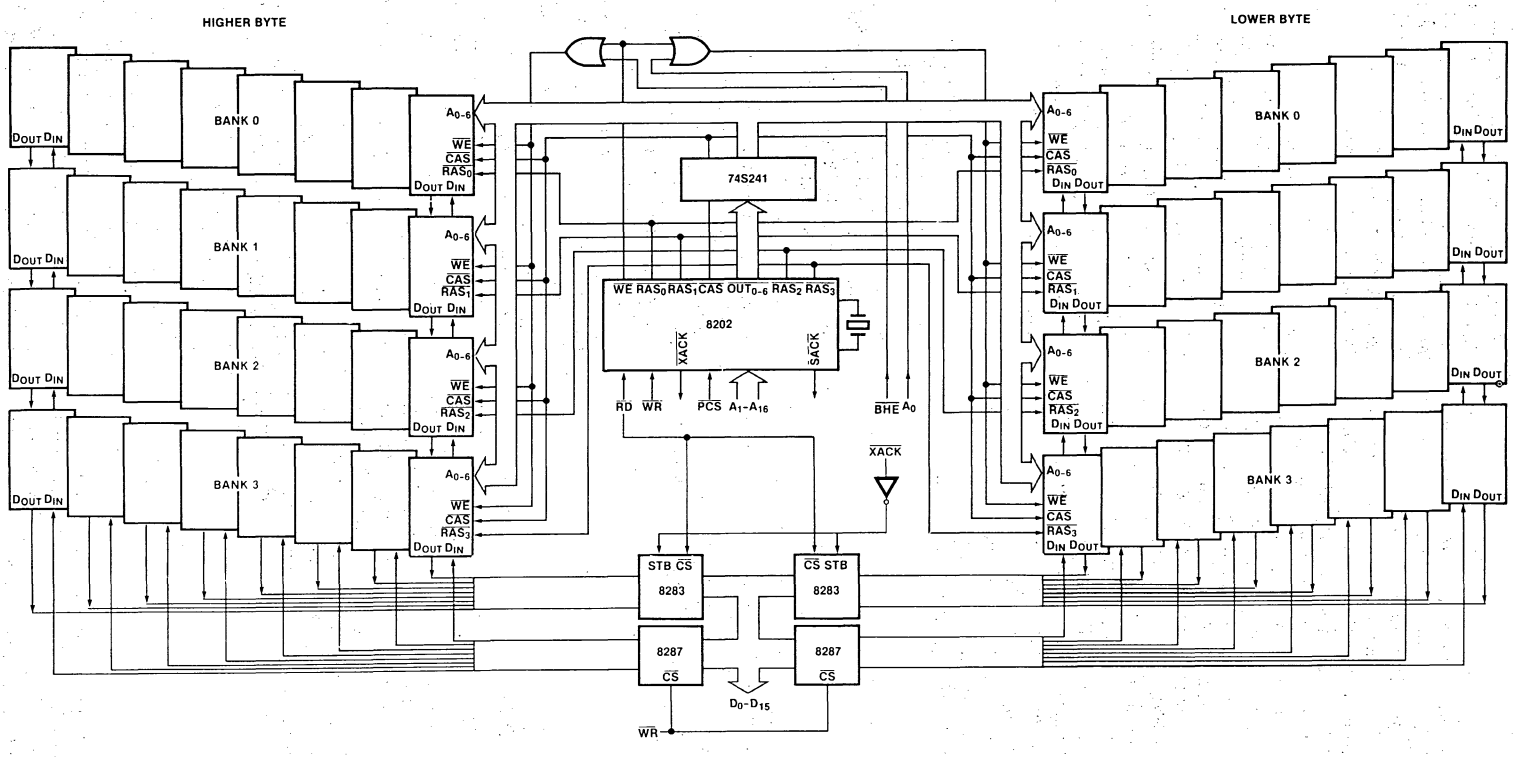


Figure 16. 8202 with 64K 16 Bit Words

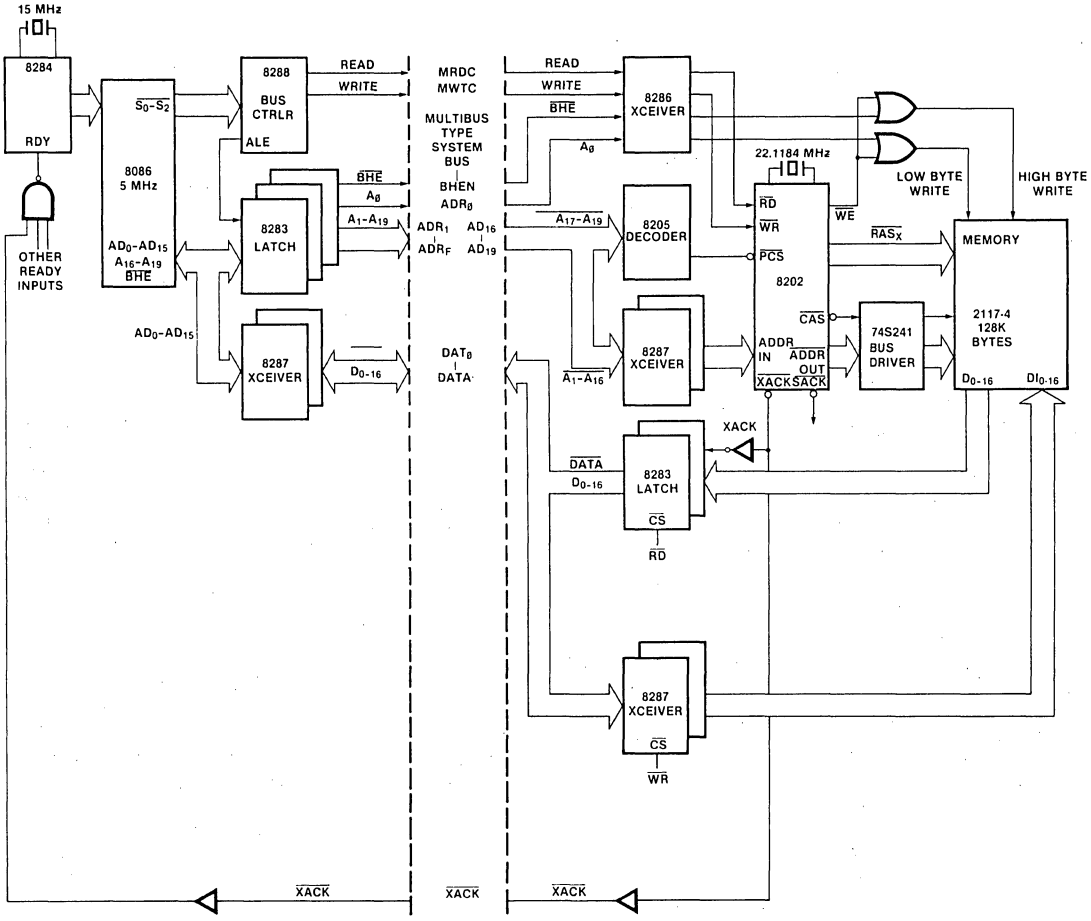


Figure 17. 5 MHz 8086/8202/128K Byte System — Double Data, Control and Address Buffering (Note: Bus driver on 8202 is not needed if less than 64K bytes are used)

SUMMARY

One of the main concerns with buying dynamic RAM has been its difficulty to use. Common complaints and comments are: "You've got to multiplex the address and you have to hassle with complicated clocks and refresh. Sure the memory is less expensive, denser and uses less power, but it isn't worth it."

Well to those of you who have avoided dynamic RAM so far, you don't have any excuses anymore. With the 8202, as far as the processor knows, the memory is static, and the design hassles are gone.

APPENDIX 1. DYNAMIC RAM OVERVIEW

Why would anyone want dynamic RAM? Why not stay with static? First, physical size — dynamic RAM generally has a four to one advantage over static RAM in terms of how much memory can be placed in one component. With each static RAM memory cell taking about 4 transistors versus one or two for a dynamic RAM cell, die size will be smaller for dynamic RAM. Smaller die size means more memory in one component and *less expensive* memory.

Another attribute associated with dynamic RAM is its lower power consumption. Storing information on a capacitor type cell (as is done in dynamic RAM) and leaving it until you need it consumes much less power than a flip-flop that is continuously on (as in most static RAMs). Dynamic RAM is, therefore, *less expensive* to operate.

What do you have to pay for having denser, lower cost memory? Milton Friedman claims that there is no such as a free lunch. Well, you have to refresh the part. In other words, the information on the storage cell must be periodically recharged before it degrades to an indiscernible level. The beauty of the 8202 is that it does this for you, regardless of resets, single stepping, bus holds or DMA. Even though the lunch isn't free, you've got someone else paying for it! To illustrate the dramatic reduction in layout area and operating power/bit, a comparison of dynamic RAM vs static is shown in Table 1.

TABLE 1

Device	No. of Bits	BITS/IN² Board Area	Operating Power/Bit
2102A static	1K	2K	0.16 mw
2114 static	4K	7K	0.10 mw
2104A dynamic	4K	8K	0.02 mw
2117,2118 dynamic	16K	32K	0.01 mw

The Intel 2104A and 2117 dynamic RAMs listed above are both packaged in a 16 pin configuration as shown in Figure 18 (A_6 is $\overline{CS}$ for 2104A). These devices are organized as 4096 or 16384 bits × 1 respectively. With this type of organization there is only one data input and output associated with each device. To use this type of memory in an 8 bit microcomputer system, it is necessary to place 8 of these devices in parallel, all controlled by the same control signals. This allows the user to have memory organized in integer groups of 4K or 16K bytes (8K, numbered 2109, is also available).

Addressing

If one was to look at the number of address pins in the above dynamic RAM pinout he would come to the conclusion that there was only half as many address pins as needed. Addressing for the 2014A and 2117 is performed in a multiplexed fashion. One half of the addresses are given to the RAM on the leading edge of $\overline{RAS}$ (Row Address Strobe) and the remaining half on the same address bus, on the leading edge of $\overline{CAS}$ (Column Address Strobe), as shown in Figure 19. This saves pins on the RAM device to allow for a compact memory layout.

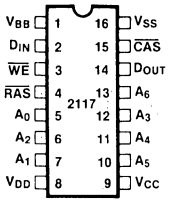


Figure 18. Pinout for Multiplexed Address Dynamic RAMs

Read Access

For multiplexed address dynamic RAMs, two read access specs are given; t_{RAC} (access time from $\overline{RAS}$) and t_{CAC} (access time from $\overline{CAS}$). Output data is valid t_{RAC} nanoseconds from $\overline{RAS}$ as long as the $\overline{RAS}$ to $\overline{CAS}$ timing specification is met. If $\overline{RAS}$ to $\overline{CAS}$ timing is longer than the specified maximum, data is valid t_{CAC} nanoseconds after $\overline{CAS}$.

Write

For a normal write, input data must be valid at a certain time with respect to $\overline{CAS}$ (or $\overline{WE}$ if using read-modify-write write cycle) and prior to write going low. Dynamic RAM differs from static in this respect as write data is latched on the leading edge of write versus trailing in static RAM systems. Some microprocessors do not guarantee data to be valid at the leading edge of $\overline{WR}$ which must be taken into consideration when interfacing to dynamic RAM. It is not possible for a dynamic RAM controller to guarantee that data is set up at the proper time for a valid write to occur, making the user responsible for this. This point is discussed in more detail in the body of the Application Note.

Refresh

The memory cell that the addresses select is a capacitor that needs to be recharged (refreshed) periodically. To understand refresh and how to do it, it is first best to *conceptually* understand the way a dynamic RAM memory array is layed out. The 2104A can be viewed as a 64 row by 64 row column matrix arrangement, with the 2117 as 128 rows by 128 columns. The addresses presented to the memory array on the falling edge of $\overline{RAS}$ are decoded and used to select one of the rows, whereas the addresses presented on the falling edge of $\overline{CAS}$ select one of the columns. The 64 cells (128 in 2117) in the selected row are all individually connected to one of 64 columns (see Figure 20). Then only one selected column is connected to the input/output (d_{in}/d_{out}) line to

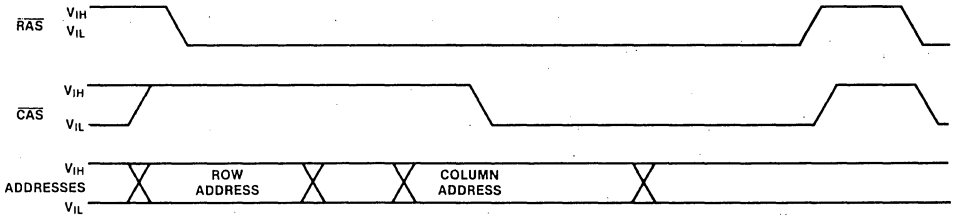


Figure 19. Dynamic RAM Clocking

present the information to the outside world. However, each column in the array has associated with it circuitry called as sense amplifier. This sense amplifier serves a dual purpose; one is, when selected, to drive the output buffer with the cell information and the other is to refresh or enhance the information on the cell, whether that column is selected or not.

As can be seen by this, with each unique $\overline{RAS}$ address, one row is refreshed. To refresh the entire memory in the most efficient manner, a "memory cycle" must be performed for all the possible combinations of $\overline{RAS}$ addresses ($2^6 = 64$ for 2104A and $2^7 = 128$ for 2117). Note that a "memory cycle" can be a read, write or special refresh cycle. Specifications for refresh (generally 2 ms) give the maximum amount of time that can elapse from when a row is refreshed to when it is refreshed again. Two general methods of refreshing have developed to meet this specification, distributed and burst refresh. Distributed refresh is represented by a refresh cycle being performed every 2 ms/#rows period of time. Burst refresh has all the rows refreshed consecutively in one block of time every 2 ms. Both theoretically take the same amount of time, neglecting overhead of the processor or controller. The 8202 performs a *distributed* refresh, at least one cycle every 15+ microseconds on the 2117 and 31+ microseconds for the 2104A (approximately 2 ms/128 and 2 ms/64 respectively). For further information, please refer to the Intel 2104A, 2117 data sheets and the Intel Memory Design Handbook.

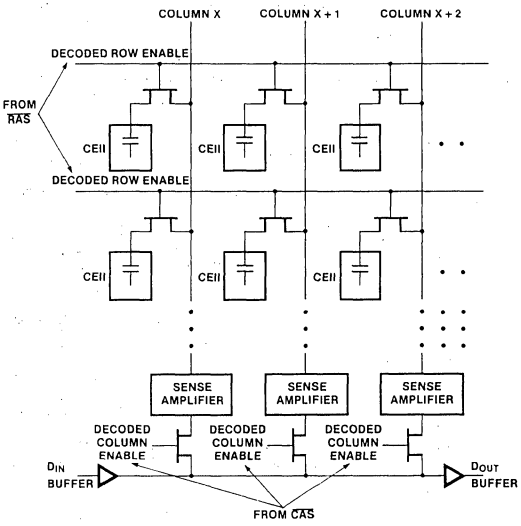
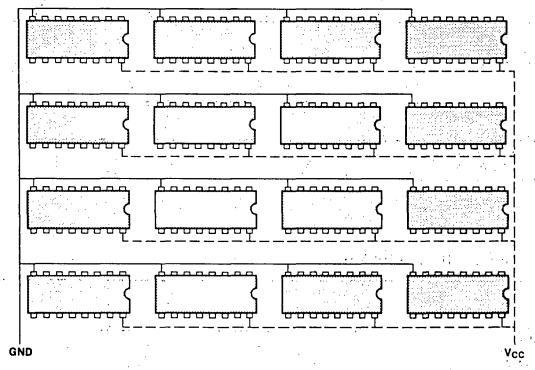


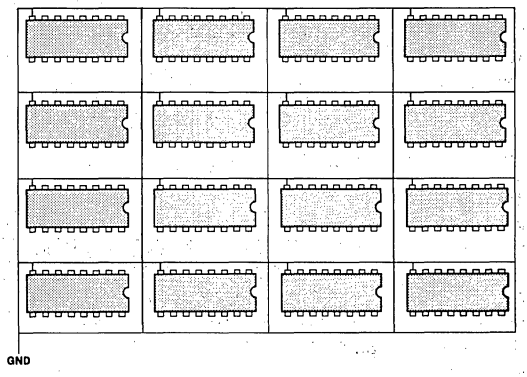
Figure 20. Typical Dynamic RAM Memory



NOTES:

TYPICAL TTL AND BREADBOARD POWER DISTRIBUTION

- Usually single sided.
- Decoupling sparse if at all.
- Totally inadequate for memory use.



How do we correct this inadequacy?

STEP ONE

- "GRID" Ground.
- Requires double sided board.
- Run vertical traces on one side of the board — horizontal traces on the other.
- Interconnect at every crossing.
- This decreases inductance.

Remember — Traces are pieces of transmission line.

— Connecting N transmission lines in parallel decreases impedance to 1/N of original value. Lower impedance means less coupling to signal traces.

FORMULAS!

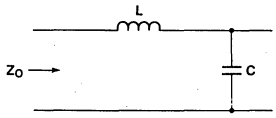
Z_O = TRANSMISSION LINE IMPEDANCE

$$= \sqrt{\frac{L}{C}}$$

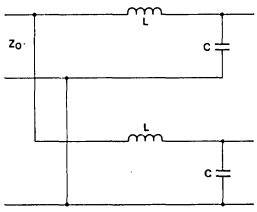
L = INDUCTANCE/ UNIT LENGTH
C = CAPACITANCE/ UNIT LENGTH

SO

EQUIVALENT TRANSMISSION LINE CIRCUIT



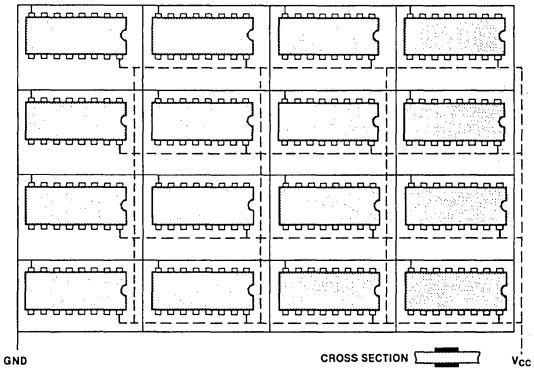
CONNECTING TWO LINES IN PARALLEL



AND $Z_{O'} = \sqrt{\frac{L}{\frac{2}{2C}}} = \frac{1}{2}\sqrt{\frac{L}{C}} = \frac{1}{2} Z_O$

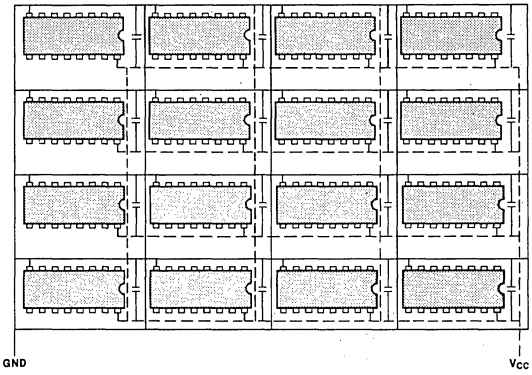
THE MORE YOU CONNECT —
THE LOWER THE EFFECTIVE IMPEDANCE

NOTES:



STEP TWO

- Grid the power supply — same as the ground grid.
- Impedance is better controlled — conductors are above each other.
- Same argument as for gridding the ground applies — lowers impedance and reduces coupling to signal traces.



STEP THREE

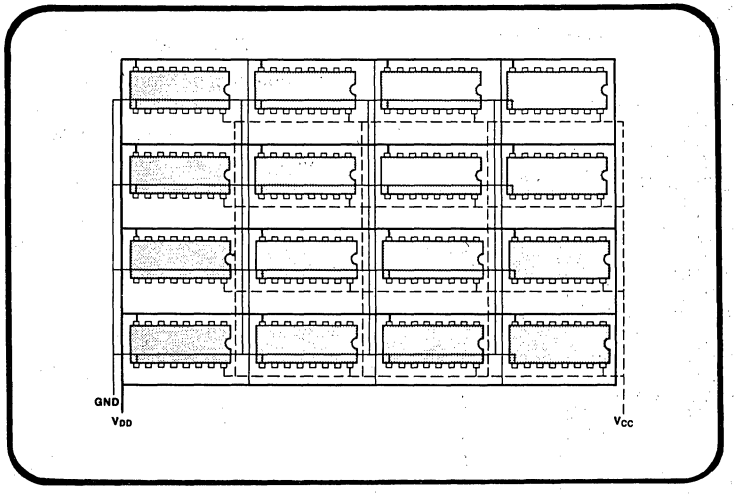
- Add decoupling capacitors.
- And now a few words about decoupling capacitors.

00203A

NOTES:

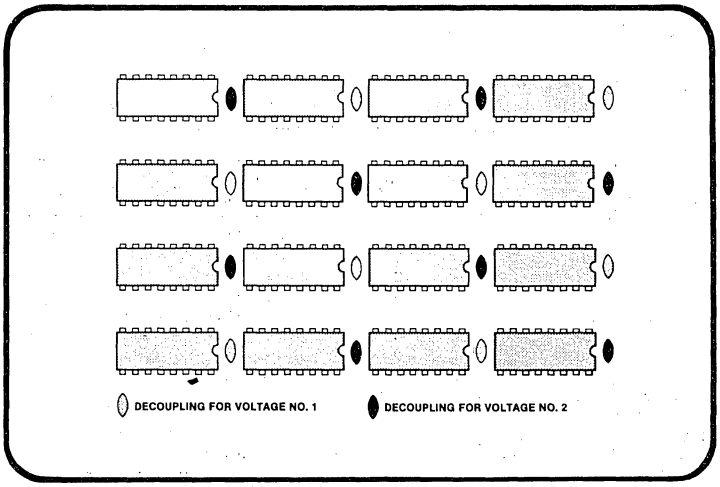
DECOUPLING CAPACITORS

Decoupling capacitors smooth out transients (spikes) found on voltage supply lines, by providing a source of energy to supply the extra current ($i = c\ dv/dt$) required. To see how this happens, let's analyze what happens when a device requires extra current, as in the case of a CS signal causing some output buffers and decoders to turn on. The decoupling capacitors that are located throughout the board, usually small, high frequency .1μF ceramic types, instantaneously provide the extra current required. The large electrolytic capacitors, usually located near the card edge or the edge of the array, replenish the charge to the small high frequency capacitors by way of the low impedance gridded distribution system. The large capacitors, usually tantalum electrolyte types, are slower in frequency response and therefore cannot instantaneously provide the charge required at the device. Finally, the power supply, which has a very low frequency response, replenishes the charge top the large (sometimes called "bulk") decoupling capacitors.



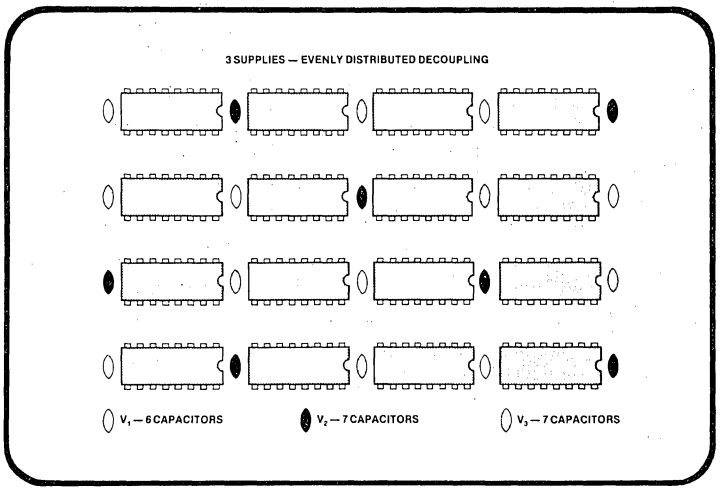
GOTCHA YOU SAY —
I NEED 2 POWER SUPPLIES!!!

Easy. Lay in a separate grid using the same techniques as for the previous ground and supply distribution.

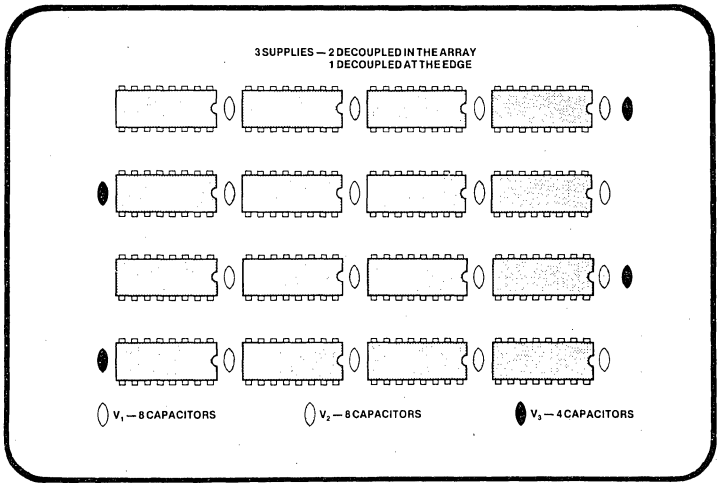


NOTES:

- OK. So now we've got 2 power supplies gridded.
  Where do the decoupling caps go?
- This is just one of several schemes. The idea is to get the minimum distance from any capacitor to any voltage pin on a device.
- With some devices a scheme that places the low current (actually the low di/dt) decoupling near the edges is preferred, as it allows for more decoupling internal to the array.



- Three supplies you say? No problem.



00203A

MEMORY COMPATIBILITY WITH 8202

(Does not include processor compatibility)

Under column labeled "Note": ok signifies that there is no problem with this parameter below the maximum 8202 frequency of 25 MHz. Any area where there exists user considerations is numbered and explained below. Blank spaces refer to specs that the user must guarantee with the processor interface.

2104A		S6047			S6048			S6049			8202		
parameter	description	min	max	note	min	max	note	min	max	note	par	min	max
tREF	Time betw refr		2ms	1		2ms	1		2ms	1	tREF	548tP	576tP
tRP	$\overline{RAS}$ precharge	100		ok	120		ok	120		ok	tRP	4tP–30	
tCP	$\overline{CAS}$ precharge	60		ok	80		ok	110		ok	>tRP		
tRCD	$\overline{RAS}$ to $\overline{CAS}$	20	50	2	25	65	2	35	85	2	tRCD	2tP–10	2tP+30
tRSH	$\overline{RAS}$ hold	100		ok	135		ok	165		ok	tRSH	5tP–30	
tAR	$\overline{RAS}$ to addr	95		ok	120		ok	160		ok	>tCAH		
tASR	Row addr setup	0		ok	0		ok	0		ok	tASR	tpH	
tASC	Col addr setup	–10		ok	–10		ok	–10		ok	tASC	tP–35	
tRAH	Row addr hold	20		ok	25		ok	35		3	tRAH	tP–10	
tCAH	Col addr hold	45		ok	55		ok	75		ok	tCAH	5tP	
tCAC	$\overline{CAS}$ access		100			135			165		processor depend		
tRAC	$\overline{RAS}$ access		150			200			250		processor depend		
READ CYCLE													
tRC	Read, Write cycle	320		ok	375		ok	410		ok	tRC	10tp–30	12tP
tRAS	$\overline{RAS}$ pulse width	150	10K	ok	200	10K	ok	250	10K	ok	>tRSH		
tCAS	$\overline{CAS}$ pulse width	100		ok	135		ok	165		ok	tCAS	5tP	
tRCS	READ cmd setup	0		ok	0		ok	0		ok	tRC-tWCH guarantees		
tRCH	READ cnd hold	0		ok	0		ok	0		ok	by definition		
WRITE CYCLE													
tWCS	Write cmd set	0		ok	0		ok	0		ok	tWCS	tP–40	
tWCH	Write cmd hold	45		ok	55		ok	75		ok	tWCH	5tP	
tWCR	Write hold to $\overline{RAS}$	95		ok	120		ok	160		ok	>tWCH		
tWP	Write pulse	45		ok	55		ok	75		ok	>tWCH		
tRWL	Write to $\overline{RAS}$ lead	50		ok	70		ok	85		ok	>tCAS		
tCWL	Write to $\overline{CAS}$ lead	50		ok	70		ok	85		ok	>tCAS		
tCS	Data in setup	0			0			0			processor depend		
tDH	Data in hold	55			65			75			processor depend		
tDHR	Data in to $\overline{RAS}$	95			120			160			processor depend		

Notes:

All numbers given above are in ns unless otherwise specified. All mathematical operations are from left to right.

1. 2 ms/64 rows = 555 (tP) Therefore tP = 54.253 ns
   f8202 lower = 1/tP = 18.432 ns

2. CAS access limited

3. tRAH = 35 = tP – 10 Therefore tP = 45 ns
   f8202 upper = 1/tP = 22.22222 MHz

MEMORY COMPATIBILITY WITH 8202

(Does not include processor compatibility)

Under column labeled "Note": ok signifies that there is no problem with this parameter below the maximum 8202 frequency of 25 MHz. Any area where there exists user considerations is numbered and explained below. Blank spaces refer to specs that the user must guarantee with the processor interface.

2117		2117-2			2117-3			2117-4			8202		
parameter	description	min	max	note	min	max	note	min	max	note	par	min	max
tREF	Time betw refr		2ms	1		2ms	1		2ms	1	tREF	264tP	288tP
tRP	$\overline{RAS}$ precharge	100		ok	120		ok	150		ok	tRP	4tP–30	
tCP	$\overline{CAS}$ precharge	25		ok	25		ok	25		ok	>tRP		
tRCD	$\overline{RAS}$ to $\overline{CAS}$	20	50	2	25	65	2	35	85	2	tRCD	2tP–10	2tP+30
tRSH	$\overline{RAS}$ hold	100		ok	135		ok	165		ok	tRSH	5tP–30	
tAR	$\overline{RAS}$ to addr	95		ok	120		ok	160		ok	>tCAH		
tASR	Row addr setup	0		ok	0		ok	0		ok	tASR	tpH	
tASC	Col addr setup	–10		ok	–10		ok	–10		ok	tASC	tP–35	
tRAH	Row addr hold	20		ok	25		ok	35		3	tRAH	tP–10	
tCAH	Col addr hold	45		ok	55		ok	75		ok	tCAH	5tP	
tCAC	$\overline{CAS}$ access		100			135			165		processor depend		
tRAC	$\overline{RAS}$ access		150			200			250		processor depend		
READ CYCLE													
tRC	Read, Write cycle	320		ok	375		ok	410		ok	tRC	10tP–30	12tP
tRAS	$\overline{RAS}$ pulse width	150	10K	ok	200	10K	ok	250	10K	ok	>tRSH		
tCAS	$\overline{CAS}$ pulse width	100		ok	135		ok	165		ok	tCAS	5tP	
tRCS	READ cmd setup	0		ok	0		ok	0		ok	tRC-tWCH guarantees		
tRCH	READ cnd hold	0		ok	0		ok	0		ok	by definition		
WRITE CYCLE													
tWCS	Write cmd set	–20		ok	–20		ok	–20		ok	tWCS	tP–40	
tWCH	Write cmd hold	45		ok	55		ok	75		ok	tWCH	5tP	
tWCR	Write hold to $\overline{RAS}$	95		ok	120		ok	160		ok	>tWCH		
tWP	Write pulse	45		ok	55		ok	75		ok	>tWCH		
tRWL	Write to $\overline{RAS}$ lead	60		ok	80		ok	100		ok	>tCAS		
tCWL	Write to $\overline{CAS}$ lead	60		ok	80		ok	100		ok	>tCAS		
tCS	Data in setup	0			0			0			processor depend		
tDH	Data in hold	45			55			75			processor depend		
tDHR	Data in to $\overline{RAS}$	95			120			160			processor depend		

Notes:

All numbers given are in ns unless otherwise specified. All mathematical operations are from left to right.

1. 2 ms/128 rows = 288 tP Therefore tP = 54.253 ns
   f8202 lower = 18.432 ns

2. CAS access limited

3. Same as 2104A tRAH

Using The 8251 Universal Synchronous/Asynchronous Receiver/Transmitter

by Lionel Smith

NEW PRODUCT INFORMATION
8251A

The industry standard USART, the Intel® 8251 has now been improved and is called 8251A. It is packed with features and enhancements as described below. Using the 8251A considerably simplifies programming and minimizes CPU overhead even further.

FEATURES AND ENHANCEMENTS

8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors that includes the new 8085 CPU and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and ehhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.
- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the RD and WR do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Baud rate from DC to 64K.
- Fully compatible with Intel's new industry standard, the MCS-85.

INTRODUCTION

The Intel 8251 is a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) which is capable of operating with a wide variety of serial communication formats. Since many peripheral devices are available with serial interfaces, the 8251 can be used to interface a microcomputer to a broad spectrum of peripherals, as well as to a serial communications channel. The 8251 is part of the MCS-80™ Microprocessor Family, and as such it is capable of interfacing to the 8080 system with a minimum of external hardware.

This application note describes the 8251 as a component and then explains its use in sample applications via several examples. A specific use of the 8251 to facilitate communication between two MCS-80 systems is discussed in detail from both the hardware and software viewpoints. The first two sections of this application note describe the 8251 first from a functional standpoint and then on a detailed level. The function of each input and output pin is fully defined. The next section describes the various operating modes and how they can be selected, and finally, a sample design is discussed using the 8251 as a data link between the MCS-80 systems.

COMMUNICATION FORMATS

Serial communications, either on a data link or with a local peripheral, occurs in one of two basic formats; asynchronous or synchronous. These formats are similar in that they both require framing information to be added to the data to enable proper detection of the character at the receiving end. The major difference between the two formats is that the asynchronous format requires framing information to be added to each character, while the synchronous format adds framing information to blocks of data, or messages. Since the synchronous format is more efficient than the asynchronous format but requires more complex decoding, it is typically found on high-speed data links, while the asynchronous format is used on lower speed lines.

The asynchronous format starts with the basic data bits to be transmitted and adds a "START" bit to the front of them and one or more "STOP" bits behind them as they are transmitted. The START bit is a logical zero, or SPACE, and is defined as the positive voltage level by RS-232-C. The STOP bit is a logical one, or MARK, and is defined as the negative voltage level by RS-232-C. In current loop applications current flow normally indicates a MARK and lack of current a SPACE. The START bit tells the receiver to start assembling a character and allows the receiver to synchronize itself with the transmitter. Since this synchronization only has to last for the duration of the character (the next character will contain a new START bit), this method works quite well assuming a properly designed receiver. One or more STOP bits are added to the end of the character to ensure that the START bit of the next character will cause a transition on the communication line and to give the receiver time to "catch up" with the transmitter if its basic clock happens to be running slightly slower than that of the transmitter. If, on the other hand, the receiver clock happens to be running slightly faster than the transmitter clock, the receiver will perceive gaps between characters but will still correctly decode the data. Because of this tolerance to minor frequency deviations, it is not necessary that the transmitter and receiver clocks be locked to the identical frequency for successful asynchronous communication.

The synchronous format, instead of adding bits to each character, groups characters into records and adds framing characters to the record. The framing characters are generally known as SYN characters and are used by the receiver to determine where the character boundaries are in a string of bits. Since synchronization must be held over a fairly long stream of data, bit synchronization is normally either extracted from the communication channel by the modem or supplied from an external source.

An example of the synchronous and asynchronous formats is shown in Figure 1. The synchronous format shown is fairly typical in that it requires two SYN characters at the start of the message. The asynchronous format, also typical, requires a START bit preceding each character and a single STOP bit following it. In both cases, two 8-bit characters are to be transmitted. In the asynchronous mode 10*n bits are used to transmit n characters and in the synchronous mode 8N + 16 bits are used. For the example shown the asynchronous mode is actually more efficient, using 20 bits versus 32. To transmit a thousand characters in the asynchronous mode, however, takes 10,000 bits versus 8,016 for the synchronous format mode. For long messages the synchronous format becomes much more efficient than the asynchronous format; the crossover point for the examples shown in Figure 1 is eight characters, for which both formats require 80 bits.

In addition to the differences in format between synchronous and asynchronous communication, there are differences with regards to the type of modems that can be used. Asynchronous modems typically employ FSK (Frequency Shift Keying) techniques which simply generate one audio tone for a MARK and another for a SPACE. The receiving modem detects these tones on the telephone

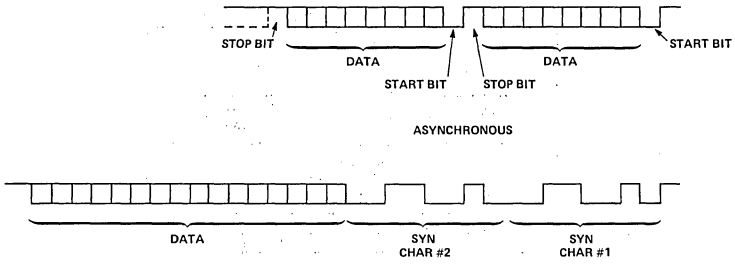


Figure 1. Transmission Formats

line, converts them to logical signals, and presents them to the receiving terminal. Since the modem itself is not concerned with the transmission speed, it can handle baud rates from zero to its maximum speed. Synchronous modems, in contrast to asynchronous modems, supply timing information to the terminal and require data to be presented to them in synchronism with this timing information. Synchronous modems, because of this extra clocking, are only capable of operating at certain preset baud rates. The receiving modem, which has an oscillator running at the same frequency as the transmitting modem, phase locks its clock to that of the transmitter and interprets changes of phase as data.

In some cases it is desirable to operate in a hybrid mode which involves transmitting data with the asynchronous format using a synchronous modem. This occurs when an increase in operating speed is required without a change in the basic protocol of the system. This hybrid technique is known as isosynchronous and involves the generation of the start and stop bits associated with the asynchronous format, while still using the modem clock for bit synchronization.

The 8251 USART has been designed to meet a broad spectrum of requirements in the synchronous, asynchronous, and isosynchronous modes. In the synchronous mode the 8251 operates with 5, 6, 7, or 8-bit characters. Even or odd parity can be optionally appended and checked. Synchronization can be achieved either externally via added hardware or internally via SYN character detection. SYN detection can be based on one or two characters which may or may not be the same. The single or double SYN characters are inserted into the data stream automatically if the software fails to supply data in time. The automatic generation of SYN characters is required to prevent the loss of synchronization. In the asynchronous mode the 8251 operates with the same data and parity structures as it does in the synchronous mode. In addition to appending a START bit to this data, the 8251 appends 1, 1½, or 2 STOP bits. Proper framing is checked by the receiver and a status flag set if an error occurs. In the asynchronous mode the USART can be programmed to accept clock rates of 16 or 64 times the required baud rate. Isosynchronous operation is a special case of asynchronous with the multiplier rate programmed as one instead of 16 or 64. Note that X1 operation is only valid if the clocks of the receiver and transmitter are synchronized.

The 8251 USART can transmit the three formats in half or full duplex mode and is double-buffered internally (i.e., the software has a complete character time to respond to a service request). Although the 8251 supports basic data set control signals (e.g., DTR and RTS), it does not fully support the signaling described in EIA-RS-232-C. Examples of unsupported signals are Carrier Detect (CF), Ring Indicator (CE), and the secondary channel signals. In some cases an additional port will be required to implement these signals. The 8251 also does not interface to the voltage levels required by EIA-RS-232-C; drivers and receivers must be added to accomplish this interface.

BLOCK DIAGRAM

A block diagram of the 8251 is shown in Figure 2. As can be seen in the figure, the 8251 consists of five major sections which communicate with each other on an internal data bus. The five sections are the receiver, transmitter, modem control, read/write control, and I/O Buffer. In order to facilitate discussion, the I/O Buffer has been shown broken down into its three major subsections: the status buffer, the transmit data/command buffer, and the receive data buffer.

Receiver

The receiver accepts serial data on the RxD pin and converts it to parallel data according to the appropriate format. When the 8251 is in the asynchronous mode and it is ready to accept a character

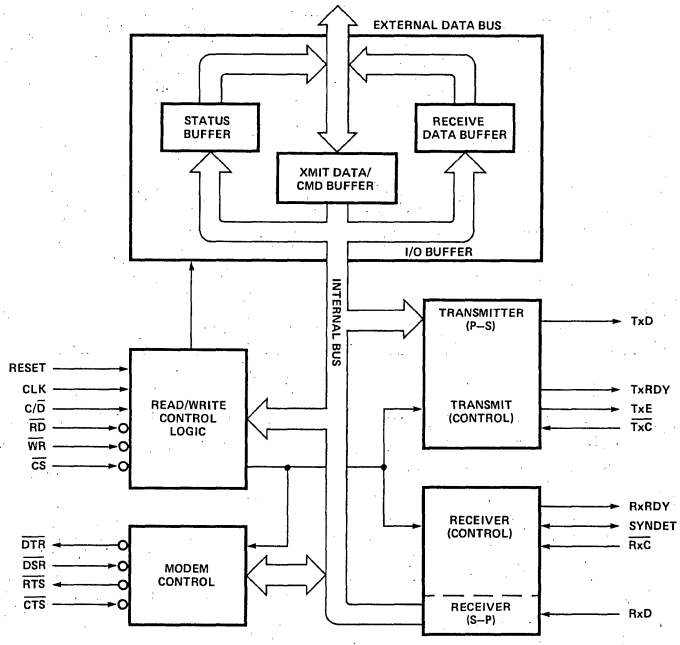


Figure 2. 8251 Block Diagram

(i.e., it is not in the process of receiving a character), it looks for a low level on the RxD line. When it sees the low level, it assumes that it is a START bit and enables an internal counter. At a count equivalent to one-half of a bit time, the RxD line is sampled again. If the line is still low, a valid START bit has probably been received and the 8251 proceeds to assemble the character. If the RxD line is high when it is sampled, then either a noise pulse has occurred on the line or the receiver has become enabled in the middle of the transmission of a character. In either case the receiver aborts its operation and prepares itself to accept a new character. After the successful reception of a START bit the 8251 clocks in the data, parity, and STOP bits, and then transfers the data on the internal data bus to the receive data register. When operating with less than 8 bits, the characters are right-justified. The RxRDY signal is asserted to indicate that a character is available.

In the synchronous mode the receiver simply clocks in the specified number of data bits and transfers them to the receiver buffer register, setting RxRDY. Since the receiver blindly groups data bits into characters, there must be a means of synchronizing the receiver to the transmitter so that the proper character boundaries are maintained in the serial data stream. This synchronization is achieved in the HUNT mode.

In the HUNT mode the 8251 shifts in data on the RxD line one bit at a time. After each bit is received, the receiver register is compared to a register holding the SYN character (program loaded). If the two registers are not equal, the 8251 shifts in another bit and repeats the comparison. When the registers compare as equal, the 8251 ends the HUNT mode and raises the SYNDET line to indicate that it has achieved synchronization. If the USART has been programmed to operate with two SYN characters the process is as described above, except that two contiguous characters from the line must compare to the two stored SYN characters before synchronization is declared. Parity is not checked. If the USART has been programmed to accept external synchronization, the SYNDET pin is used as an input to synchronize the receiver. The timing necessary to do this is discussed in the SIGNALS section of this note. The USART enters the HUNT mode when it is initialized into the synchronous mode or when it is commanded to do so by the command instruction. Before the receiver is operated, it must be enabled by the RxE bit (D_2) of the command instructions. If this bit is not set the receiver will not assert the RxRDY bit.

Transmitter

The transmitter accepts parallel data from the processor, adds the appropriate framing information, serializes it, and transmits it on the TxD pin. In the asynchronous mode the transmitter always

adds a START bit; depending on how the unit is programmed, it also adds an optional even or odd parity bit, and either 1, 1½, or 2 STOP bits. In the synchronous mode no extra bits (other than parity, if enable) are generated by the transmitter unless the computer fails to send a character to the USART. If the USART is ready to transmit a character and a new character has not been supplied by the computer, the USART will transmit a SYN character. This is necessary since synchronous communications, unlike asynchronous communications, does not allow gaps between characters. If the USART is operating in the dual SYN mode, both SYN characters will be transmitted before the message can be resumed. The USART will not generate SYN characters until the software has supplied at least one character; i.e., the USART will fill 'holes' in the transmission but will not initiate transmission itself. The SYN characters which are to be transmitted by the USART are specified by the software during the initialization procedure. In either the synchronous or asynchronous modes, transmission is inhibited until TxEnable and the $\overline{CTS}$ input are asserted.

An additional feature of the transmitter is the ability to transmit a BREAK. A BREAK is a period of continuous SPACE on the communication line and is used in full duplex communication to interrupt the transmitting terminal. The 8251 USART will transmit a BREAK condition as long as bit 3 (SBRK) of the command register is set.

Modem Control

The modem control section provides for the generation of $\overline{RTS}$ and the reception of $\overline{CTS}$. In addition, a general purpose output and a general purpose input are provided. The output is labeled $\overline{DTR}$ and the input is labeled $\overline{DSR}$. $\overline{DTR}$ can be asserted by setting bit 2 of the command instruction; $\overline{DSR}$ can be sensed as bit 7 of the status register. Although the USART itself attaches no special significance to these signals, DTR (Data Terminal Ready) is normally assigned to the modem, indicating that the terminal is ready to communicate and DSR (Data Set Ready) is a signal from the modem indicating that it is ready for communications.

I/O Control

The Read/Write Control Logic decodes control signals on the 8080 control bus into signals which gate data on and off the USART's internal bus and controls the external I/O bus (DB_0–DB_7). The truth table for these operations is as follows:

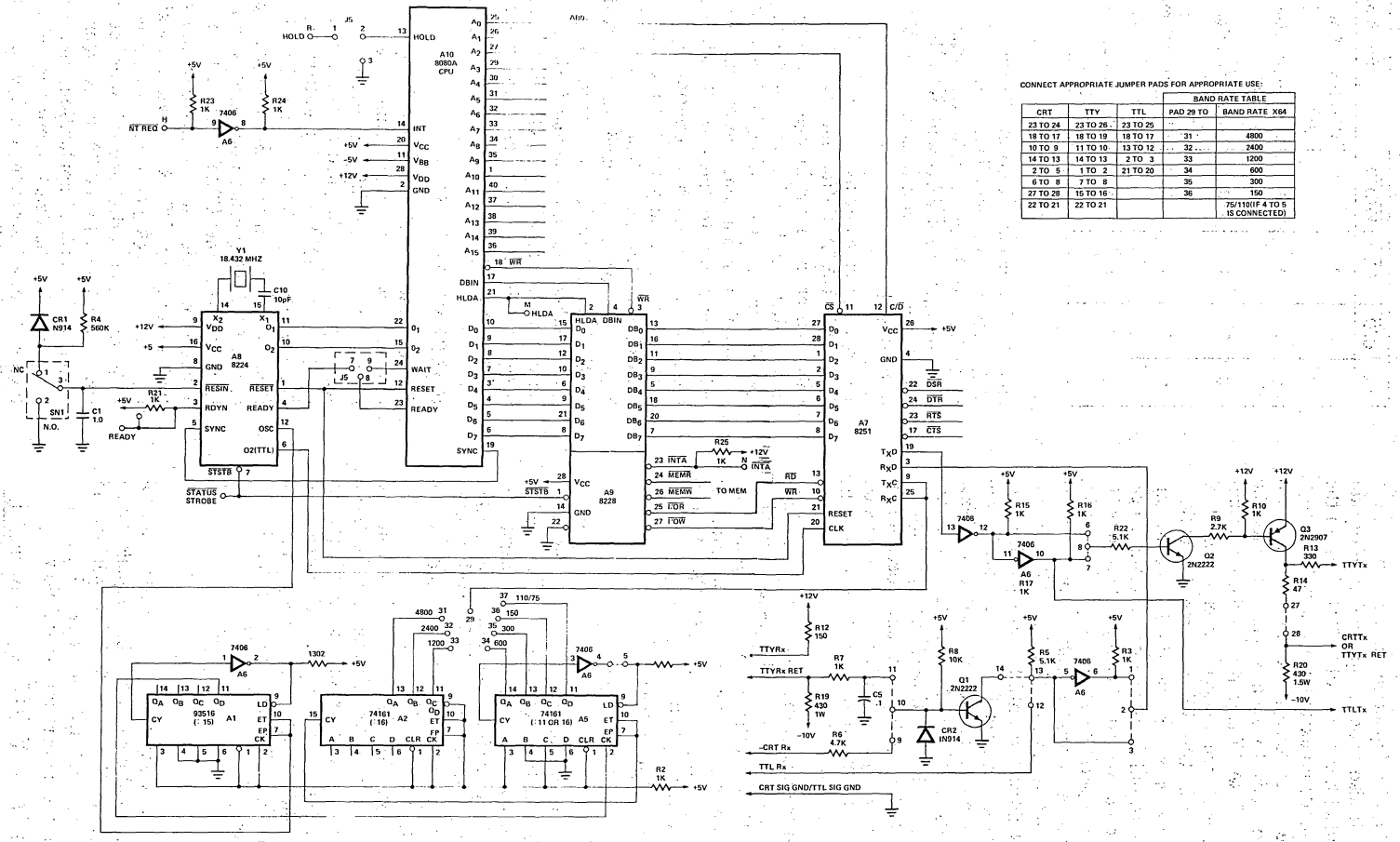
If neither $\overline{READ}$ or $\overline{WRITE}$ is a zero, then the USART will not perform an I/O function. $\overline{READ}$ and $\overline{WRITE}$ being a zero at the same time is an illegal state with undefined results. The Read/Write Control Logic contains synchronization circuits so that the $\overline{READ}$ and $\overline{WRITE}$ pulses can occur at any time with respect to the clock inputs to the USART.

CE	C/$\overline{D}$	$\overline{READ}$	$\overline{WRITE}$	Function
0	0	0	1	CPU Reads Data from USART
0	1	0	1	CPU Reads Status from USART
0	0	1	0	CPU Writes Data to USART
0	1	1	0	CPU Writes Command to USART
1	X	X	X	USART Bus Floating (NO-OP)

The I/O buffer contains the STATUS buffer, the RECEIVE DATA buffer and the XMIT DATA/CMD buffer as shown in Figure 2. Note that although there are two registers which store data for transfer to the CPU (STATUS and RECEIVE DATA), there is only one register which stores data being transferred to the USART. The sharing of the input register for both transmit data and commands makes it important to ensure that the USART does not have data stored in this register before sending a command to the device. The TxRDY signal can be monitored to accomplish this. Neither data nor commands should be transferred to the USART if TxRDY is low. Failure to perform this check can result in erroneous data being transmitted.

INTERFACE SIGNALS

The interface signals of the 8251 USART can be broken down into two groups – a CPU-related group and a device-related group. The CPU-related signals have been designed to optimize the attachment of the 8251 to a MCS-80™ system. The device-related signals are intended to interface a modem or like device. Since many peripherals (TTY, CRT, etc.) can be obtained with a modem-like interface, the USART has a broad range of applications which do not include a modem. Note that although the USART provides a logical interface to an EIA-RS-232 device, it does not provide EIA compatible drive, and this must be added via circuitry external to the 8251. As an example of a peripheral interface application and to aid in understanding the signal descriptions which follow, Figure 3 shows a system configured to interface with a TTY or CRT.



CONNECT APPROPRIATE JUMPER PADS FOR APPROPRIATE USE:

CRT	TTY	TTL	BAND RATE TABLE	
			PAD 29 TO	BAND RATE X64
23 TO 24	23 TO 26	23 TO 25		
18 TO 17	18 TO 19	18 TO 17	31	4800
10 TO 9	11 TO 10	13 TO 12	32	2400
14 TO 13	14 TO 13	2 TO 3	33	1200
2 TO 5	1 TO 2	21 TO 20	34	600
6 TO 8	7 TO 8		35	300
27 TO 28	15 TO 16		36	150
22 TO 21	22 TO 21			75/110(IF 4 TO 5 IS CONNECTED)

Figure 3. Terminal Interface

CPU-Related Signals

V_{CC} (26)	I	+5 Volt Supply
GND (4)	I	+5 Volt Common
CLK (20)	I	The CLK input generates internal device timing. No external inputs or outputs are referenced to CLK, but the frequency of CLK must be greater than 30 times the Receiver or Transmitter clock inputs for synchronous mode or 4.5 times the clock inputs for an asynchronous mode. An additional constraint is imposed by the electrical specifications (ref. Appendix B) which require the period of CLK be between 0.42 μsec and 1.35 μsec. The CLK input can generally be connected to the Phase 2 (TTL) output of the 8224 clock generator.
RESET (21)	I	A high on this input performs a master reset on the 8251. The device returns to the idle mode and will remain there until reinitialized with the appropriate control words.
DB_7–DB_0 (8,7,6,5,2,1, 28,27)	I/O	The DB signals form a three-state bus which can be connected to the CPU data bus. Control, status, and data are transferred on this bus. Note that the CPU always remains in control of the bus and all transfers are initiated by it.
$\overline{CS}$ (11)	I	*Chip Select.* A low on this input enables communication between the USART and the CPU. Chip Select should go low when the USART is being addressed by the CPU.
C/$\overline{D}$ (12)	I	*Control/Data.* During a read operation this pin selects either status or data to be input to the CPU (high=status, low=data). During a write operation this pin causes the USART to interpret the data on the bus as a command if it is high or as data if it is low.
$\overline{RD}$ (13)	I	A low on this input causes the USART to gate either status or data onto the data bus.
$\overline{WR}$ (10)	I	A low on this input causes the USART to accept data on the data bus as either a command or as a data character.
TxRDY (15)	O	*Transmitter Ready.* This output signals the CPU that the USART is ready to accept a data character or command. It can be used as an interrupt to the system or, for polled operation, the CPU can check TxRDY using the status read operation. Note, however, that while the TxRDY status bit will be asserted whenever the XMIT DATA/CMD buffer is empty, the TxRDY output will be asserted only if the buffer is empty and the USART is enabled to transmit (i.e., $\overline{CTS}$ is low and TxEN is high). TxRDY will be reset when the USART receives a character from the program.
TxE (18)	O	*Transmitter Empty.* A high output on this line indicates that the parallel to serial converter in the transmitter is empty. In the synchronous mode, if the CPU has failed to load a new character in time, TxE will go high momentarily as SYN characters are loaded into the transmitter to fill the gap in transmission.
RxRDY (14)	O	*Transmitter Ready.* This output goes high to indicate that the 8251 has received a character on its serial input and is ready to transfer it to the CPU. Although the receiver runs continuously, RxRDY will only be asserted if the RxE (Receive Enable) bit in the command register has been set. RxRDY can be connected to the interrupt structure or, for polled operation, the CPU can check the condition of RxRDY using a status read operation. RxRDY will be reset when the character is read by the CPU.

SYNDET (16) I/O *Synch Detect.* This line is used in the synchronous mode only. It can be either an input or output, depending on whether the initialization program sets the USART for external or internal synchronization. SYNDET is reset to a zero by RESET. When in the internal synchronization mode, the USART uses SYNDET as an output to indicate that the device has detected the required SYN character(s). A high output indicates synchronization has been achieved. If the USART is programmed to operate with double SYN characters, SYNDET will go high in the middle of the last bit of the second SYN character. SYNDET will be reset by a status read operation. When in the external synchronization mode a positive-going input on the SYNDET line will cause the 8251 to start assembling characters on the next falling edge of $\overline{RxC}$. The high input should be maintained at least for one RxC cycle following this edge.

Device-Related Signals

$\overline{DTR}$ (24) O *Data Terminal Ready.* This is a general purpose output signal which can be set low by programming a '1' in command instruction bit 1. This signal allows additional device control.

$\overline{DSR}$ (22) I *Data Set Ready.* This is a general purpose input signal. The status of this signal can be tested by the CPU through a status read. This pin can be used to test device status and is read as bit 7 of the status register.

$\overline{RTS}$ (23) O *Request to Send.* This is a general purpose output signal equivalent to $\overline{DTR}$. RTS is normally used to request that the modem prepare itself to transmit (i.e., establish carrier). $\overline{RTS}$ can be asserted (brought low) by setting bit 5 in the command instruction.

$\overline{CTS}$ (17) I *Clear to Send.* A low on this input enables the USART to transmit data. CTS is normally generated by the modem in response to a $\overline{RTS}$.

$\overline{RxC}$ (25) I *Receiver Clock.* This clock controls the data rate of characters to be received by the USART. In the synchronous mode RxC is equivalent to the baud rate, and is supplied by the modem. In asynchronous mode $\overline{RxC}$ is 1, 16, or 64 times the baud rate. The clock division is preselected by the mode control instruction. Data is sampled by the USART on the rising edge of $\overline{RxC}$.

RxD (3) I *Receiver Data.* Characters are received serially on this pin and assembled into parallel characters. RxD is high true (i.e., High = MARK or ONE).

$\overline{TxC}$ (9) I *Transmitter Clock.* This clock controls the rate at which characters are transmitted by the USART. The relationship between clock rate and baud rate is the same as for RxC. Data is shifted out of the USART on the falling edge of TxC.

TxD (19) O *Transmit Data.* Parallel characters sent by the CPU are transmitted serially by the USART on this line. TxD is high true (i.e., High = MARK or ONE).

MODE SELECTION

The 8251 USART is capable of operating in a number of modes (e.g., synchronous or asynchronous). In order to keep the hardware as flexible as possible (both at the chip and end product level), these operating modes are selected via a series of control outputs to the USART. These mode control outputs must occur between the time the USART is reset and the time it is utilized for data transfer. Since the USART needs this information to structure its internal logic it is essential to complete the initialization before any attempts are made at data transfer (including reading status).

A flowchart of the initialization process appears in Figure 4. The first operation which must occur following a reset is the loading of the mode control

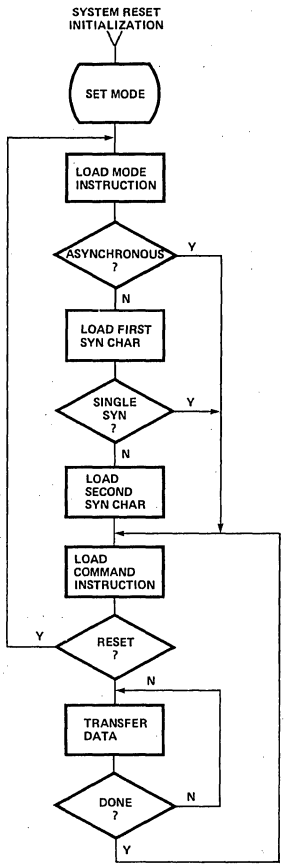


Figure 4. Initialization Flowchart

register. The mode control register is loaded by the first control output ($C/\overline{D}$=1, $\overline{RD}$=1, $\overline{WR}$=0, $\overline{CS}$=0) following a reset. The format of the mode control instruction is shown in Figure 5. The instruction can be considered as four 2-bit fields. The first 2-bit field (D_1 D_0) determines whether the USART is to operate in the synchronous (00) or asynchronous mode. In the asynchronous mode this field also controls the clock scaling factor. As an example, if D_1 and D_0 are both ones, the $\overline{RxC}$ and $\overline{TxC}$ will be divided by 64 to establish the baud rate. The second field, D_3–D_2, determines the number of data bits in the character and the third, D_5–D_4, controls parity generation. Note that the parity bit (if enabled) is added to the data bits and is not considered as part of them when setting up the character length. As an example, standard ASCII transmission, which is seven data bits plus even parity, would be specified as:

X X 1 1 1 0 X X

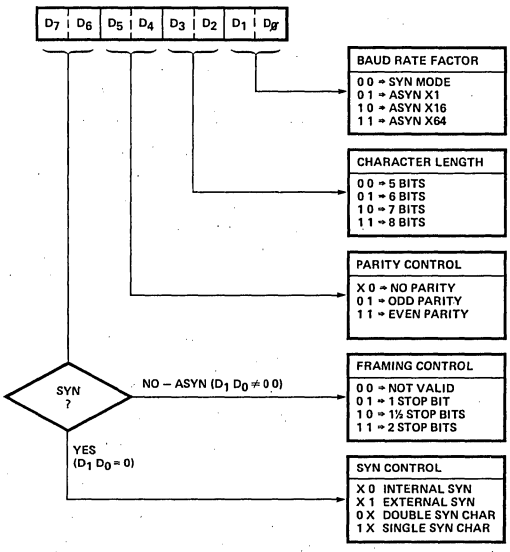


Figure 5. Mode Instruction Format

The last field, D_7–D_6, has two meanings, depending on whether operation is to be in the synchronous or asynchronous mode. For the asynchronous mode (i.e., D_1 $D_0 \neq 00$), it controls the number of STOP bits to be transmitted with the character. Since the receiver will always operate with only one STOP bit, D_7 and D_6 only control the transmitter. In the synchronous mode (D_1 D_0 = 00), this field controls the synchronizing process. Note that the choice of single or double SYN characters is independent of the choice of internal or external synchronization. This is because even though the receiver may operate with external synchronization logic, the transmitter must still know whether to send one or two SYN characters should the CPU fail to supply a character in time.

Following the loading of the mode instruction the appropriate SYN character (or characters) must be loaded if synchronous mode has been specified. The SYN character(s) are loaded by the same control output instruction used to load the mode instruction. The USART determines from the mode instruction whether no, one, or two SYN characters are required and uses the control output to load SYN characters until the required number are loaded.

At completion of the load of SYN characters (or after the mode instruction in the asynchronous mode), a command character is issued to the USART. The command instruction controls the operation of the USART within the basic framework established by the mode instruction. The format of the command instruction is shown in

Figure 6. Note that if, as an example, the USART is waiting for a SYN character load and instead is issued an internal reset command, it will accept the command as a SYN character instead of resetting. This situation, which should only occur if two independent programs control the USART, can be avoided by outputting three all zero characters as commands before issuing the internal reset command. The USART indicates its state in a status register which can be read under program control. The format of the status register read is shown in Figure 7.

When operating the receiver it is important to realize that RxE (bit 2 of the command instruction) only inhibits the assertion of RxRDY; it does not inhibit the actual reception of characters. Because the receiver is constantly running, it is possible for it to contain extraneous data when it is enabled. To avoid problems this data should be read from the USART and discarded. The read should be done immediately following the setting of Receive Enable in the asynchronous mode, and following the setting of Enter Hunt in the synchronous mode. It is not necessary to wait for RxRDY before executing the dummy read.

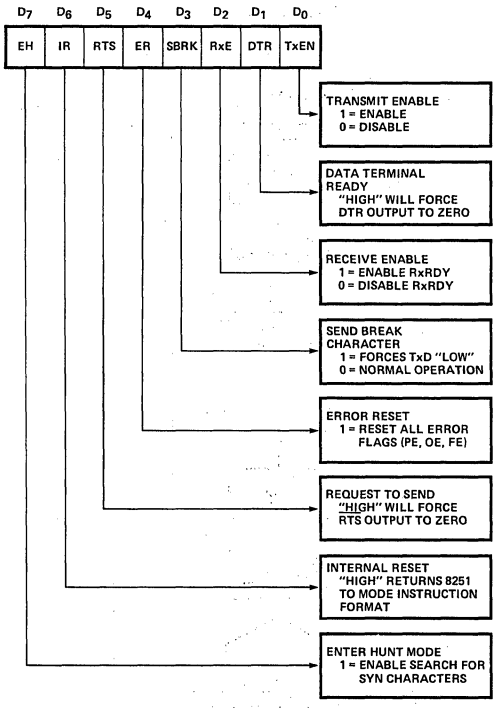


Figure 6. Command Instruction Format

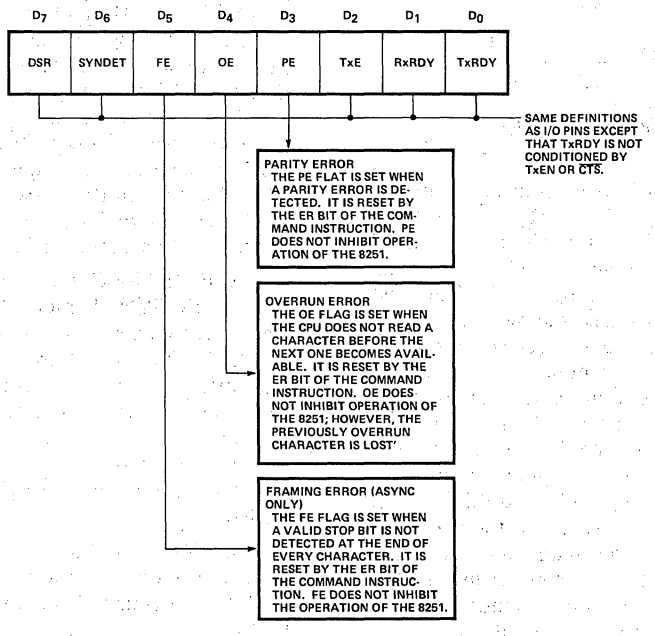


Figure 7. Status Register Format

PROCESSOR DATA LINK

The ability to change the operating mode of the USART by software makes the 8251 an ideal device to use to implement a serial communication link. A terminal initially configured with a simple asynchronous protocol can be upgraded to a synchronous protocol such as IBM Binary Synchronous Communication by a software only upgrade. In order to demonstrate the use of the 8251 USART, the remainder of this document will describe the implementation of an interrupt-driven, full duplex communication link on the Intel MDS™ system. With minor modifications, the program developed could be used on the Intel SBC-80/10™ OEM card, thus implementing a data link between the two systems. Such a facility can be used to down-load programs, run diagnostics, and maintain common data bases in multiprocessor systems.

The factors which must be considered in the design of such a link include the desired transmission rate and format, the error checking requirements, the desirability of full duplex operation, and the physical implementation of the link. The basic requirement of the system described here is that it allow an Intel SBC-80/10 OEM card to be loaded from an MDS development system, either locally or on the switched telephone network. An additional constraint is that the modem used on the switched network be readily available and inexpensive. These requirements led to the choice of a modem such as the Bell 103A to implement the link. These modems, which support full duplex communication at up to 300 baud, are readily available from a number of sources at reasonable cost. These modems are also available in acoustically coupled versions which do not require permanent installation on the telephone network. Interface to the 103A modem is accomplished with nine wires: Protective Ground, Signal Ground, Transmitted Data, Received Data, Clear to Send, Data Set Ready, Data Terminal Ready, Carrier Detector, and Ringing Indicator.

The utilization of the interface signals to the modem is as follows:

Protective Ground	Protective Ground is used to bond the chassis ground of the modem to that of the terminal.
Signal Ground	Signal Ground provides a common ground reference between the modem and the terminal.
Transmitted Data	Transmitted Data is used to transfer serial data from the terminal to the modem.
Received Data	Received Data is used to transfer serial data from the modem to the terminal.
Clear to Send	Clear to Send indicates that the modem has established a connection with a remote modem and is ready to transmit data.
Data Set Ready	Data Set Ready indicates that the modem is connected to the telephone line and is in the data mode.
Data Terminal Ready	Data Terminal Ready is a signal from the terminal which permits the modem to enter the data mode.
Carrier Detector	Carrier Detector is identical to Clear to Send in the 103 modem and will not be used in this interface.
Ringing Indicator	Ringing Indicator indicates that the modem is receiving a ringing signal from the telephone system. This signal will not be used in the interface, since it is possible for the terminal to assert Data Terminal Ready whenever it is ready for the modem to "answer the telephone". The modem uses Data Set Ready to indicate that it has answered the call.

A block diagram showing the connections between the MDS and the SBC-80/10 through the modems is shown in Figure 8. Figure 9 shows the portion of the MDS monitor board devoted to the USARTs and Figure 10 shows the equivalent section of the SBC-80/10 board. Note that several signals on the MDS to not have the proper EIA defined voltage levels, and for this reason the adapter shown in Figure 11 was added to the MDS. The 390 pF capacitor was added to the 1488 driver to bring the rise time within EIA imposed limits of 30 volts/μsec. In Figure 7 the signal labels within the MDS and SBC-80/10 blocks correspond to the labels on the schematics, the signal labels within the modem blocks correspond to EIA conventions, and the signal labels on the wires between the blocks are abbreviations for the English language names of the signals.

As an example of how the USART clocks can be generated, circuits A27, A16, and A15 of Figure 9 form a divider of the OSC signal. The OSC signal has a frequency of 18.432 MHz and is generated by the 8224 which generates system timing for the 8080A. The 18.432 MHz signal results in a state time of 488 ns versus the normal 500 ns for the 8080A. (This does not violate 8080A specifications.) The 18.432 MHz signal can be divided by

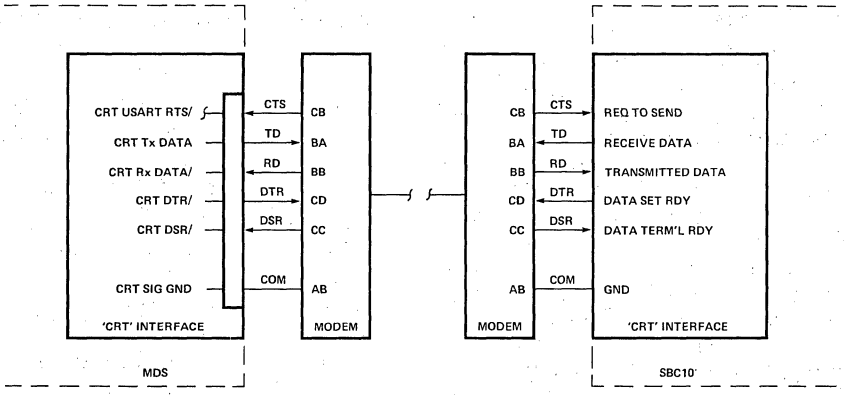


Figure 8. System Block Diagram

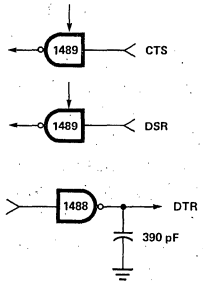


Figure 9. EIA Adapter

30 and then 64 to give a 9600 baud communication standard. The 9600 baud signal can be further divided to give 4800, 2400, 1200, 600, and 300 baud signals. The 1200 baud signal can be divided by 11 to give a 109.1 baud signal which is within 1% of the 110 baud standard signal rate. Note that because of constraints on the CLK input 9600 baud operation is not possible in the X64 mode. The divide by 64 can be accomplished by dividing by 4 with a counter and then 16 within the USART.

In order to keep the system as general purpose as possible, it was decided to transmit 8-bit data characters with an appended odd parity bit. Having a full 8-bit byte available for data enables the transmission of codes such as ASCII (which is 7-level with an additional parity bit) to be transmitted and received transparently in the system. Also, of course, it allows 8-bit bytes from the 8080A memory to be transferred in one transmission character. If error checking beyond the parity check is required, it could be added to the data record to be transmitted in the form of redundant check characters.

Before the software design of the system could be undertaken, it was necessary to decide whether service requests from the USART would be handled on a polled or interrupt driven mode. Polled operation normally results in more compact code but it requires that whatever programs are running concurrently with a transmission or reception must periodically either check the status of the USART or call a routine that does. Since it was not possible to determine what program might be running during a receive or transmit operation, it was decided to operate in an interrupt driven mode.

The program which operates the 8251 must be instructed as to what data it should transmit or receive from some other program resident in the 8080 system. To facilitate the discussion of the operation of the software, the following definitions will be made:

> USRUN is the program which controls the operation of the 8251.
>
> USER is a program which utilizes USRUN in order to effect a data transmission.

USER passes commands and parameters to USRUN by means of the control block shown in Figure 12. The first byte of the block contains the command which USER wants USRUN to execute. Valid contents of this byte are "C" which causes USRUN to initialize itself and the 8251, "R" which causes the execution of the data input (or READ) operation, and "W" which causes a data output (WRITE) operation. The second byte of the control block is used by USRUN to inform USER of the status of the requested operation. The third and fourth bytes specify the starting address of a buffer set up by USER which contains the data for a transmit operation or which will be used by USRUN to store received data. The fifth and sixth bytes are concatenated to form a positive binary

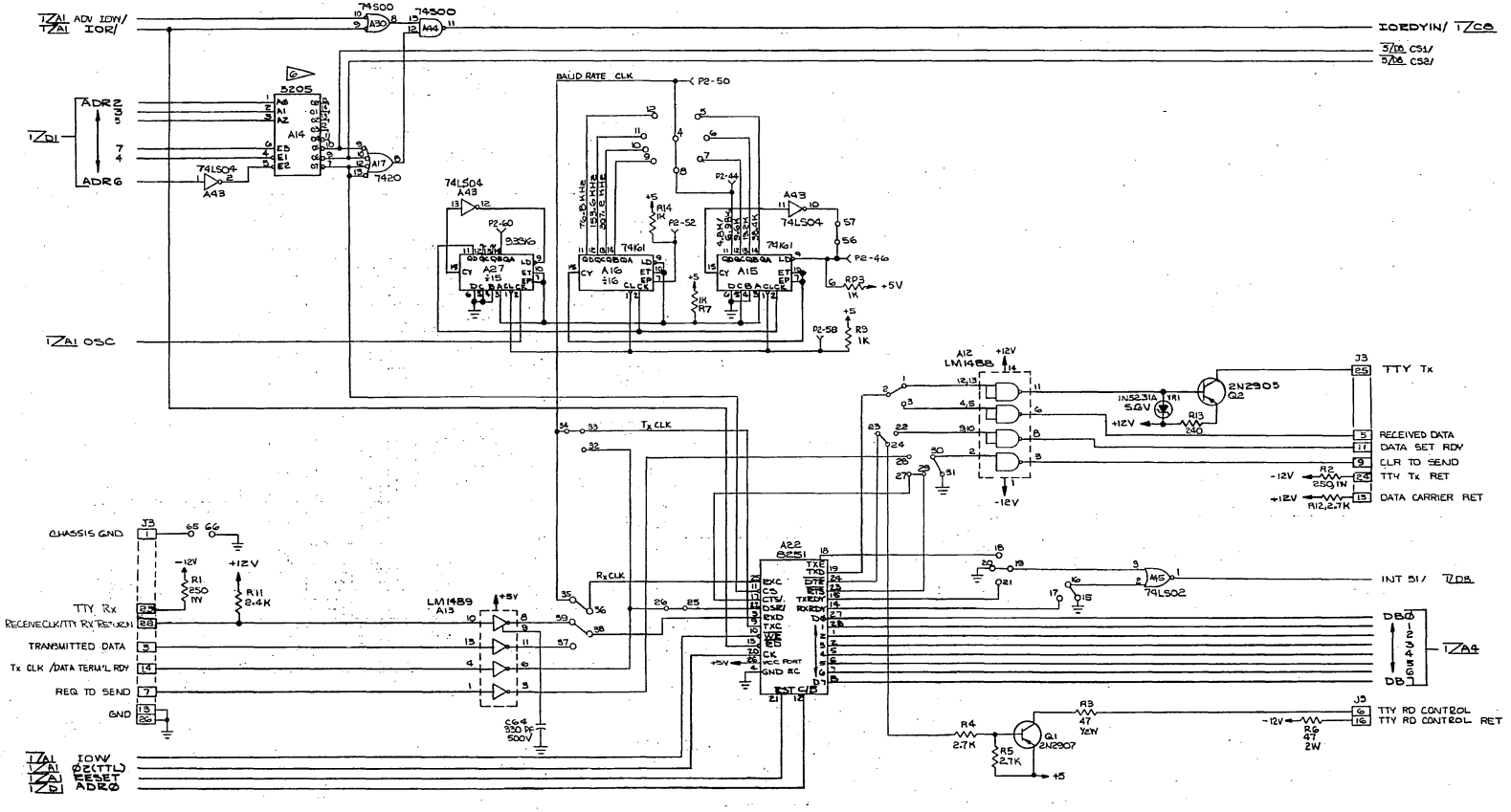


Figure 10. SBC 80/10 Serial I/O

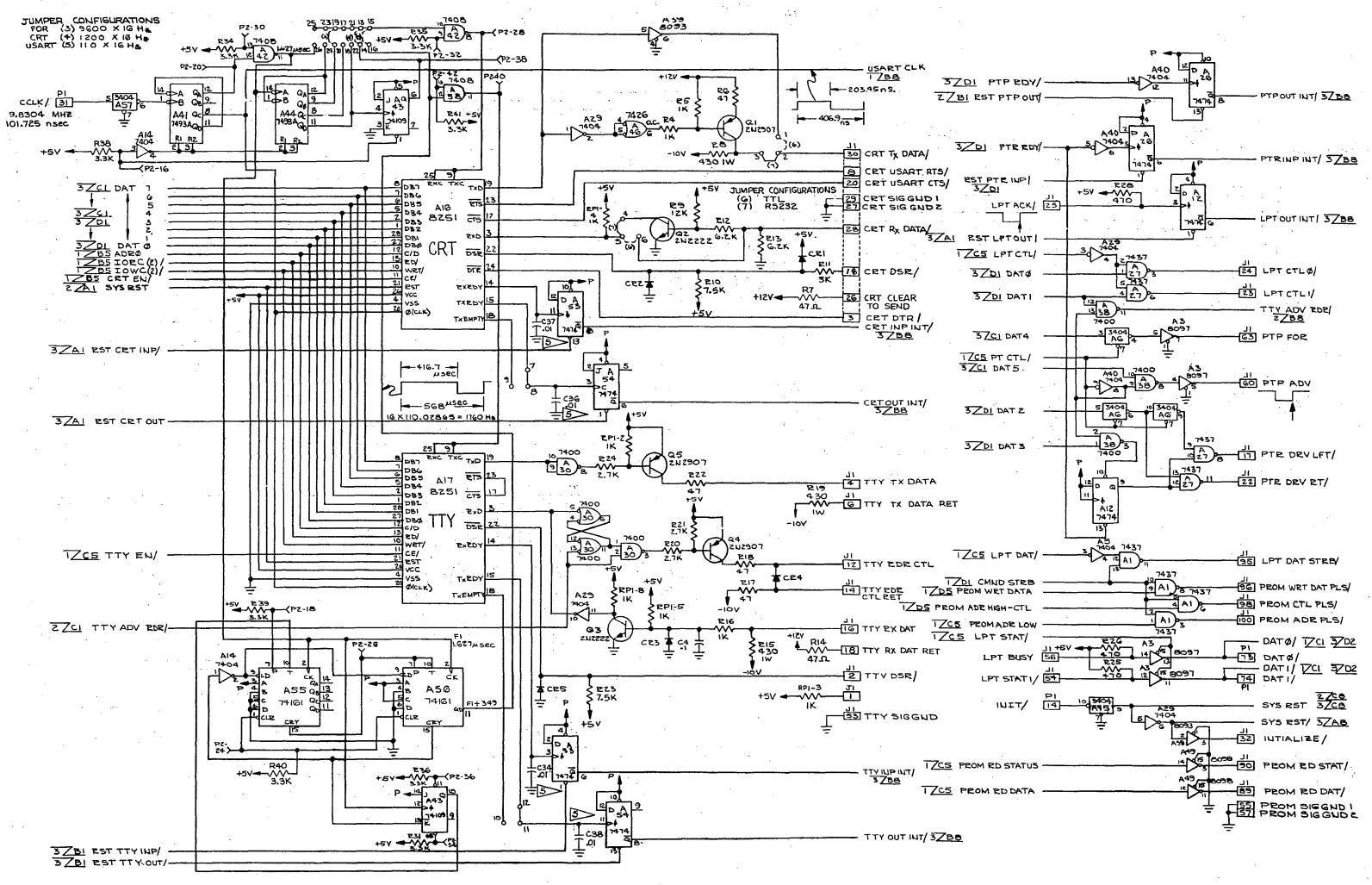


Figure 11. MDS Monitor Module

number which specifies how many bytes of data USER wants transferred. The seventh and eighth bytes are concatenated and used by USRUN to count the number of bytes that have been transferred. When the required number of characters have been transferred, or if USRUN terminates a READ or WRITE due to an abnormal condition, then USRUN calls a subroutine at an address defined by the ninth and tenth bytes of the command block. This subroutine, which is provided by USER, must determine the state of the process and then take appropriate action.

Since USRUN must be capable of operation in a full duplex mode (i.e., be able to receive and transmit simultaneously), it keeps the address of two control blocks; one for a READ operation and one for a WRITE. The address of the controlling command block is kept in RAM locations labeled RCBA for the READ operation 28d TCBA for the WRITE operation. If RCBA (Receive Control Block Address) or TCBA (Transmit Control Block Address) is zero, it indicates that the corresponding operation is in an idle status.

Flowcharts of USRUN appear in Figure 13 and the listings appear in Figure 14. The first section of the flowcharts (Figures 13.1 and 13.2) consists of two subroutines which are used as convenient tools for operating on the control blocks. These routines are labeled LOADA and CLEAN. LOADA is entered with the address of a control block in registers H and L. Upon return registers D and E have been set equal to the address in the buffer which is the target of the next data transfer (i.e., D,E = BAD+ CCT); and CCT (transferred byte count) has then been incremented. In addition, the B register is set to zero if the number of bytes that have been transferred is equal to the number requested (i.e., CCT = RCT). CLEAN, the second routine, is also entered with the address of a command block in the H and L registers. In addition, the Accumulator holds the status which will be placed in the STATUS byte of the command block. On exit the STATUS byte has been updated and the address of the completion routine has been placed in H and L.

Upon interrupt, control of the MCS-80 system is transferred to VECTOR (Figure 13.3). Vector is a program which saves the state of the system, gets the status of the USART and jumps to the RISR (Receive Interrupt Service Routine) or the TISR (Transmit Interrupt Service Routine), depending on which of the two ready flags is active. If neither ready flag is active, VECTOR restores the status of the running program, enables interrupts, and returns. (Interrupts are automatically disabled by the hardware upon an interrupt.) This exit from VECTOR, which is labeled VOUT, is used from other

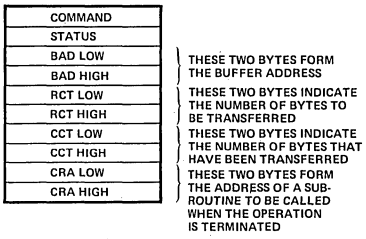


Figure 12. Control Block

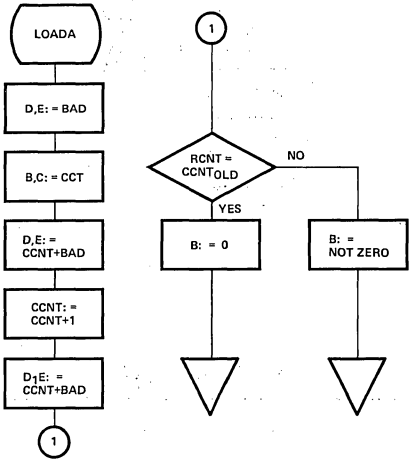


Figure 13.1. LOADA Subroutine

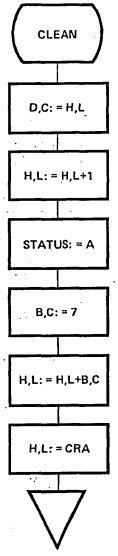


Figure 13.2. CLEAN Subroutine

portions of USRUN if return from the interrupt mode is required.

In addition to handling normal data transfers, TISR (Figure 13.4) checks a location in memory named TCMD in order to determine if the receive program wishes to send a command to the USART. Since the transmit data and command must share a buffer within the USART, any command output must occur when TxRDY is asserted. If TCMD is zero, TISR proceeds with the data transfer. If TCMD is non-zero, TISR calls TUTE (Transmit Utility, Figure 13.5) which, depending on the value

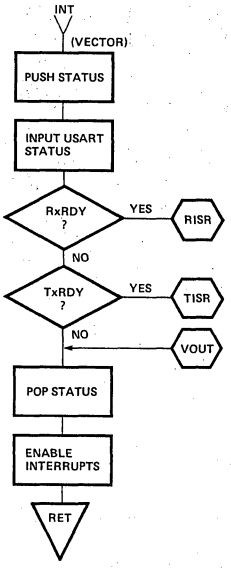


Figure 13.3. Interrupt Entry

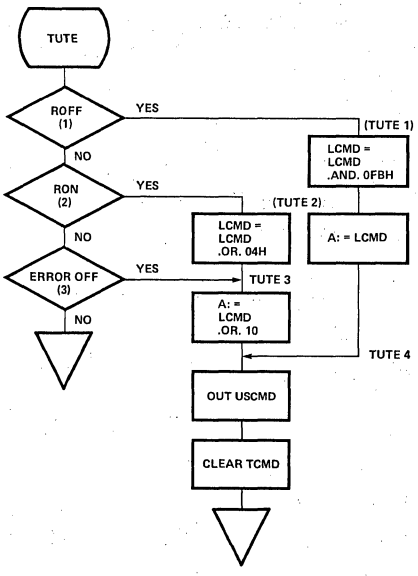


Figure 13.4. Transmit Interrupt Service Routine

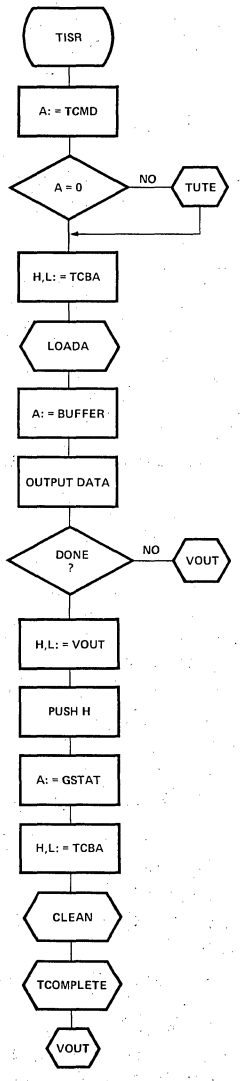


Figure 13.5. Transmit Utility Routine

in TCMD, turns off the receiver, turns on the receiver, or clears error conditions. Note that the error flags (parity, framing, and overrun) are always cleared by the software when the receiver is first enabled.

The flowchart of the RISR is shown in Figure 13.6. Note that in addition to terminating whenever the required number of characters have been received, the RISR also terminates if one of the error flags becomes set or if the received character matches a character found in a table pointed to by the label ETAB. This table, which starts at ETAB and continues until an all "ones" entry is found, can be used by USER to define special characters, such as EOT (End Of Transmission), which will terminate a READ operation. The remainder of Figure 13 (13.7) shows the decoding of the commands to USRUN. The listings also include a test USER which exercises USRUN. This program sets up a 256-byte transmit buffer and transfers it to a similar input buffer by means of a local loop. When both the READ and WRITE operations are complete, the test USER checks to insure that the two buffers are identical. If the buffers differ, the MDS monitor is called; if the data is correct, the test is repeated.

CONCLUSION

The 8251 USART has been described both as a device and as a component in a system. Since not only modems but also many peripheral devices have a serial interface, the 8251 is an extremely useful component in a microcomputer system. A particular advantage of the device is that it is capable of operating in various modes without requiring hardware modifications to the system of which it is a part. As with any complex subsystem, however, the 8251 USART must be carefully applied so that it can be utilized to full advantage in the overall system. It is hoped that this application note will aid in the designer in the application of the 8251 USART. As a further aid to the application of the 8251, the appendix of this document includes a list of design hints based on past experience with the 8251.

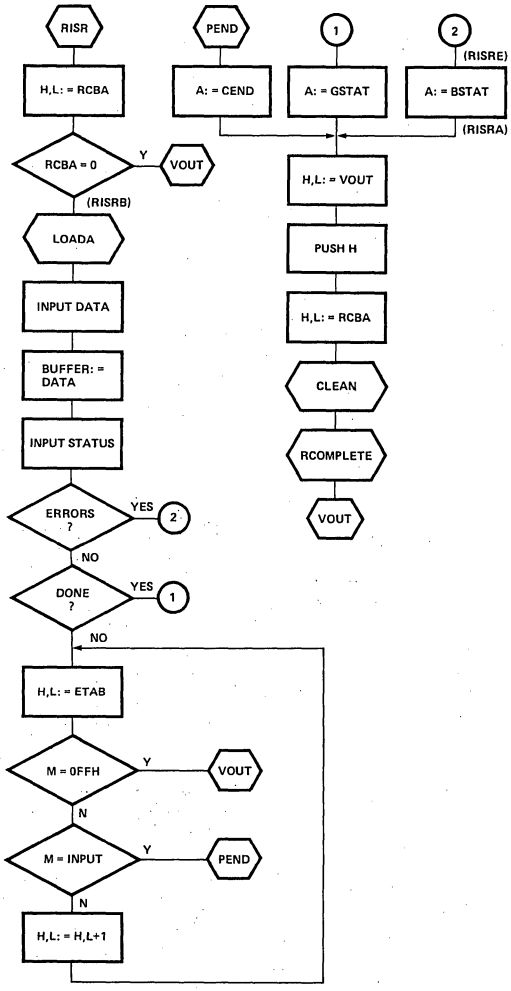


Figure 13.6. Receive Interrupt Service Routine

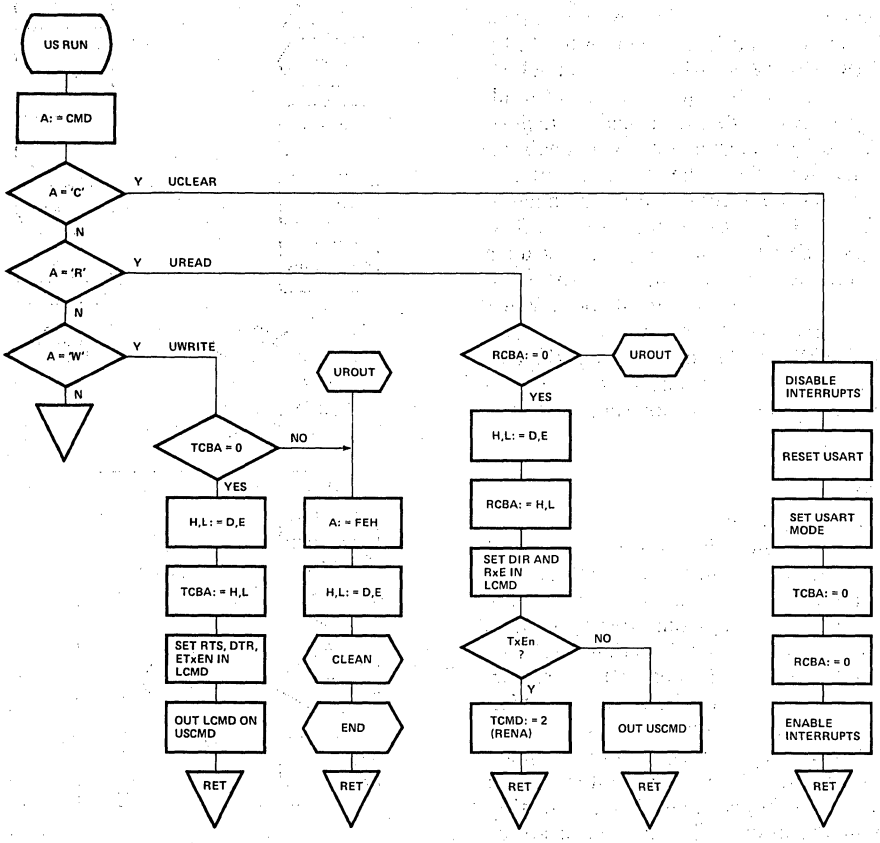


Figure 13.7. URUN Command Decode

Figure 14. Program Listing

```
                ;*****
                ;
                ;       SYSTEM ORIGIN STATEMENT
                ;
                ;*****

4000                    ORG     4000H


                ;*****
                ;
                ;       DATA STORAGE FOR TEST USER
                ;
                ;*****

4000            BUFIN:  DS      100H    ;INPUT BUFFER
4100            BUFOUT: DS      100H    ;OUTPUT BUFFER
4200 5200       RBLOCK: DB      'R',00H ;RECEIVE CONTROL BLOCK
4202 0040       RBAD:   DW      BUFIN
4204 FF00       RRCT:   DW      0FFH
4206 0000       RCCT:   DW      00H
4208 1742       RCRA:   DW      RCR
420A 5700       TBLOCK: DB      'W',00H ;TRANSMIT CONTROL BLOCK
420C 0041       TBAD:   DW      BUFOUT
420E FF00       TRCT:   DW      0FFH
4210 0000       TCCT:   DW      00H
4212 2742       TCRA:   DW      TCR
4214 4300       GBLOCK: DB      'C',00H
4216 00         FLAG:   DB      00H


                ;*****
                ;
                ;       COMPLETION ROUTINES
                ;
                ;*****


4217 AF         RCR:    XRA     A       ;CLEAR A
4218 323B42             STA     RCBA    ;TURN OFF RECEIVE
421B 323C42             STA     RCBA+1
421E 3A1642             LDA     FLAG    ;GET FLAG
4221 E60F               ANI     0FH     ;CLEAR UPPER FOUR BITS
4223 321642             STA     FLAG    ;RESTORE FLAG
4226 C9                 RET
4227 AF         TCR:    XRA     A       ;CLEAR A
4228 323942             STA     TCBA    ;TURN OFF TRANSMIT
422B 323A42             STA     TCBA+1
422E 3A1642             LDA     FLAG    ;GET FLAG
4231 E6F0               ANI     0F0H    ;CLEAR LOWER FOUR BITS
4233 321642             STA     FLAG    ;RESTORE FLAG
4236 C9                 RET             ;THEN RETURN
```

```
                ;*****
                ;
                ;       SYSTEM EQUATES
                ;
                ;*****

00F5            USTAT   EQU     0F5H    ;USART STATUS ADDRESS
00F5            USCMD   EQU     0F5H    ;USART CMD ADDRESS
00F4            USDAI   EQU     0F4H    ;USART DATA INPUT ADDRESS
00F4            USDAO   EQU     0F4H    ;USART DATA OUTPUT ADDRESS
0000            GSTAT   EQU     00H     ;GOOD STATUS
00FF            BSTAT   EQU     0FFH    ;BAD STATUS
0001            CEND    EQU     01H


                ;*****
                ;
                ;       SYSTEM DATA TABLE
                ;
                ;*****

4237 00         LCMD:   DB      00H     ;CURRENT OPERATING COMMAND
4238 00         TCMD:   DB      00H     ;IF NON ZERO A COMMAND TO BE SENT
4239 0000       TCBA:   DW      00H     ;ADDRESS OF XMIT CBLOCK
423B 0000       RCBA:   DW      00H     ;ADDRESS OF RECEIVE CBLOCK
423D FF         MTAB:   DB      0FFH    ;END CHARACTER TABLE
```

```
                ;*****
                ;
                ;       LOAD ADDRESS ROUTINE
                ;       LOADA IS ENTERED WITH THE ADDRESS OF A CONTROL
                ;       BLOCK IN H,L. ON EXIT D,E CONTAINS THE ADDRESS
                ;       WHICH IS THE TARGET OF THE NEXT DATA TRANSFER (BAD+CCNT)
                ;       AND B HAS BEEN SET TO ZERO IF THE REQUESTED NUMBER OF
                ;       TRANSFERS HAS BEEN ACCOMPLISHED. CCNT IS INCREMENTED
                ;       AFTER THE TARGET ADDRESS HAS BEEN CALCULATED.
                ;
                ;*****

423E 23         LOADA:  INX     H       ;D,E GETS BUFFER ADDRESS
423F 23                 INX     H
4240 5E                 MOV     E,M
4241 23                 INX     H
4242 56                 MOV     D,M     ;DONE
4243 23                 INX     H       ;B,C GETS COMPLETED COUNT (CCNT)
4244 23                 INX     H
4245 23                 INX     H
4246 4E                 MOV     C,M
4247 23                 INX     H
4248 46                 MOV     B,M     ;DONE
4249 EB                 XCHG            ;D,E GETS BAD+CCNT
424A 09                 DAD     B
424B EB                 XCHG            ;DONE
424C 03                 INX     B       ;CCNT GETS INCREMENTED
424D 70                 MOV     M,B
424E 2B                 DCX     H
424F 71                 MOV     M,C     ;DONE
4250 0B                 DCX     B       ;DOES OLD CCNT=RCNT?
4251 2B                 DCX     H
4252 7E                 MOV     A,M
4253 90                 SUB     B
4254 47                 MOV     B,A
4255 C0                 RNZ             ;NO-RETURN WITH B NOT ZERO
4256 2B                 DCX     H
4257 7E                 MOV     A,M
4258 91                 SUB     C
4259 47                 MOV     B,A
425A C9                 RET             ;RETURN WITH B=0 IF RCNT=CCNT
```

```
                ;*****
                ;
                ;       CLEAN-UP ROUTINE
                ;       CLEAN IS ENTERED WITH THE ADDRESS OF A CONTROL
                ;       BLOCK IN H,L AND A NEW STATUS TO BE
                ;       ENTERED INTO IT IN A. ON EXIT THE ADDRESS OF THE
                ;       CONTROL BLOCK IS IN D,E; THE STATUS OF THE BLOCK
                ;       HAS BEEN UPDATED; AND THE ADDRESS OF THE COMPLETION
                ;       ROUTINE IS IN H,L.
                ;
                ;*****

425B 5D         CLEAN:  MOV     E,L     ;SAVE THE ADRESS OF THE COMMAND BLOCK
425C 54                 MOV     D,H
425D 23                 INX     H       ;POINT AT STATUS
425E 77                 MOV     M,A     ;SET STATUS EQUAL TO A
425F 010700             LXI     B,7     ;SET INDEX TO SEVEN
4262 09                 DAD     B       ;POINT AT COMPLETION ADDRESS
4263 7E                 MOV     A,M     ;GET LOWER ADDRESS
4264 23                 INX     H       ;POINT AT UPPER ADDRESS
4265 66                 MOV     H,M     ;H GETS HIGH ADDRESS BYTE
4266 6F                 MOV     L,A     ;L GETS LOW ADDRESS BYTE
4267 C9                 RET

                ;*****
                ;
                ;       INTERUPT VECTOR ROUTINE
                ;       VECTOR SAVES THE STATUS OF THE RUNNING PROGRAM
                ;       THEN READS THE STATUS OF THE USART TO DETERMINE
                ;       IF A RECEIVE OR TRANSMIT INTERUPT OCCURRED.
                ;       VECTOR THEN CALLS THE APPROPRIATE SERVICE ROUTINE.
                ;       IF NEITHER INTERUPTS OCCURRED THEN VECTOR RESTORES
                ;       THE STATUS OF THE RUNNING PROGAM. THE SERVICE
                ;       ROUTINES USE THE EXIT CODE, LABLED VOUT, TO EFFECT
                ;       THEIR EXIT FROM INTERUPT MODE.
                ;
                ;*****

4268 F5         VECTOR: PUSH    PSW     ;PUSH STATUS INTO THE STACK
4269 C5                 PUSH    B
426A D5                 PUSH    D
426B E5                 PUSH    H
426C DBF5               IN      USTAT   ;GET USART ADDRESS
426E DBFA               IN      0FAH    ;MDS-GET MONITOR CARD INT. STATUS
4270 0F                 RRC             ;ROTATE TWO PLACES
4271 0F                 RRC             ;SO THAT CARRY=RXRDY
4272 DA8842             JC      RISR    ;IF RXRDY GO TO SERVICE ROUTINE
4275 07                 RLC             ;IF NOT ROTATE BACK
4276 07                 RLC             ;LEAVING TXRDY IN CARRY
4277 DAD442             JC      TISR    ;IF TXRDY THEN GO TO SERVICE ROUTINE
427A 3EFC               MVI     A,0FCH  ;MDS-CLEAR OTHER LEVEL THREE INTERUPTS
427C D3F3               OUT     0F3H    ;MDS
427E E1         VOUT:   POP     H       ;ELSE EXIT FROM INTERUPT MODE
427F D1                 POP     D
4280 C1                 POP     B
4281 3E20               MVI     A,20H   ;MDS-RESTORE CURRENT LEVEL
4283 D3FD               OUT     0FDH    ;MDS
4286 FB                 EI              ;ENABLE INTERUPTS
4287 C9                 RET
```

```
                ;*****
                ;
                ;       RECEIVE INTERUPT SERVICE ROUTINE;
                ;               RISR PROCESSES A RECEIVE INTERUPT
                ;               AT THE END OF RECEIVE THE USER SUPPLIED
                ;               COMPLETION ROUTINE IS CALLED AND THEN AN
                ;               EXIT IS TAKEN THROUGH VOUT OF THE
                ;               VECTOR
                ;*****

4288 2A3B42     RISR:   LHLD    RCBA
428B 3E82               MVI     A,82H   ;MDS-CLEAR RECEIVE INTERUPT
428D D3F3               OUT     0F3H    ;MDS
428F 2C                 INR     L
4290 2D                 DCR     L
4291 C29942             JNZ     RISRB
4294 24                 INR     H
4295 25                 DCR     H
4296 CA7E42             JZ      VOUT
4299 CD3E42     RISRB:  CALL    LOADA   ;READY-SET UP ADDRESS
429C DBF4               IN      USDAI   ;GET INPUT DATA
429E 12                 STAX    D       ;AND PUT IN THE BUFFER
429F 4F                 MOV     C,A     ;SAVE INPUT DATA IN C
42A0 DBF5               IN      USTAT   ;GET STATUS AGAIN
42A2 E638               ANI     38H     ;MASK FOR ERROR FIELD
42A4 C2B942             JNZ     RISRE   ;NOT ZERO-TAKE ERROR EXIT
42A7 04                 INR     B       ;B WAS 00 IF DONE
42A8 05                 DCR     B
42A9 C2BE42             JNZ     EXCHAR  ;NOT DONE-EXIT
'2AC 3E00               MVI     A,GSTAT ;A GETS GOOD STATUS
 2AE 217E42     RISRA:  LXI     H,VOUT  ;GET RETURN ADDRESS
→2B1 E5                 PUSH    H       ;AND PUSH IT INTO THE STACK
42B2 2A3B42             LHLD    RCBA    ;POINT H,L AT THE CMD BLOCK
42B5 CD5B42             CALL    CLEAN   ;CALL CLEANUP ROUTINE
42B8 E9                 PCHL            ;EFFECTIVELY CALLS COMPLETION ROUTINE
                                        ;RETURN IS TO VOUT BECAUSE OF PUSH H
42B9 3EFF       RISRE:  MVI     A,BSTAT ;A GETS BAD STATUS
42BB C3AE42             JMP     RISRA   ;OTHERWISE EXIT IS NORMAL
42BE 213D42     EXCHAR: LXI     H,MTAB  ;TEST CHARACTER AGAINST EXIT TABLE
42C1 7E         EXA:    MOV     A,M
42C2 FEFF               CPI     0FFH    ;END OF TABLE
42C4 CA7E42             JZ      VOUT
42C7 B9                 CMP     C
42C8 CACF42             JZ      PEND    ;MATCH-TERMINATE READ
42CB 23                 INX     H
42CC C3C142             JMP     EXA
42CF 3E01       PEND:   MVI     A,CEND
42D1 C3AE42             JMP     RISRA
```

```
                ;*****
                ;
                ;       TRANSMIT INTERUPT SERVICE ROUTINE
                ;       TISR PROCCESSES TRANSMITTER INTERUPTS
                ;       WHEN THE END OF A TRANSMISSION IS
                ;       DETECTED THE USER SUPPLIED COMPLETION
                ;       ROUTINE IS CALLED AND THEN AN EXIT IS
                ;       TAKEN THROUGH VOUT OF VECTOR
                ;
                ;*****

42D4 3A3842     TISR:   LDA     TCMD    ;GET POTENTIAL COMMAND
42D7 B7                 ORA     A       ;DESIGNATE ON IT
42D8 C40443             CNZ     TUTE    ;DO UTILITY COMMAND
42DB 3E81               MVI     A,081H  ;MDS-CLEAR XMIT INTERUPTS
42DD D3F3               OUT     0F3H    ;MDS
42DF 2A3942             LHLD    TCBA
42E2 2C                 INR     L       ;MAKE SURE HAVE VALID CONTROL BLOCK
42E3 2D                 DCR     L
42E4 C2EC42             JNZ     TISRA   ;GOOD
42E7 24                 INR     H
42E8 25                 DCR     H
42E9 CA7E42             JZ      VOUT    ;NON VALID BLOCK (H,L=0)
42EC CD3E42     TISRA:  CALL    LOADA   ;SET UP ADDRESS
42EF 1A                 LDAX    D       ;GET DATA FROM BUFFER
42F0 D3F4               OUT     USDAO   ;AND OUTPUT IT
42F2 04                 INR     B       ;B WAS 00 IF DONE
42F3 05                 DCR     B
42F4 C27E42             JNZ     VOUT    ;NOT DONE-EXIT FROM SERVICE ROUTINE
42F7 217E42             LXI     H,VOUT  ;SET UP RETURN ADDRESS
42FA E5                 PUSH    H       ;AND PUSH IT INTO THE STACK
42FB 3E00               MVI     A,GSTAT ;A GETS GOOD STATUS
42FD 2A3942             LHLD    TCBA    ;POINT H,L AT COMMAND BLOCK
4300 CD5B42             CALL    CLEAN   ;CALL CLEANUP ROUTINE
4303 E9                 PCHL            ;CALL COMPLETION ROUTINE
                                        ;RETURN WILL BE TO VOUT
4304 FE01       TUTE:   CPI     01      ;RECEIVER OFF
4306 CA2443             JZ      TUTE1
4309 FE02               CPI     02      ;RECEIVER ON
430B CA1443             JZ      TUTE2
430E FE03               CPI     03      ;CLEAR ERRORS
4310 CA1C43             JZ      TUTE3
4313 C9                 RET
4314 3A3742     TUTE2:  LDA     LCMD
4317 F604               ORI     04
4319 323742             STA     LCMD
431C 3A3742     TUTE3:  LDA     LCMD
431F F610               ORI     10H
4321 D3F5       TUTE4:  OUT     USCMD
4323 C9                 RET
4324 3A3742     TUTE1:  LDA     LCMD
4327 E6FB               ANI     0FBH
4329 323742             STA     LCMD
432C C32143             JMP     TUTE4
```

```
                ;*****
                ;
                ;       USART COMMAND BLOCK INTERPRETER
                ;       USRUN IS CALLED BY USER WITH THE ADDRESS
                ;       OF THE COMMAND BLOCK IN H,L. USRUN EXAMINES
                ;       THE BLOCK AND INTIALIZES THE REQUESTED OPERATION
                ;
                ;*****

432F 1A         USRUN:  LDAX    D       ;GET THE CMD FROM THE BLOCK
4330 FE43               CPI     'C'     ;IS IT A CLEAR COMMAND?
4332 CA4043             JZ      UCLEAR  ;YES GO TO CLEAR ROUTINE
4335 FE52               CPI     'R'     ;IS IT A READ COMMAND?
4337 CA5D43             JZ      UREAD   ;YES-GO TO READ ROUTINE
433A FE57               CPI     'W'     ;IS IT A WRITE COMMAND?
433C CA9D43             JZ      UWRITE  ;GO TO WRITE ROUTINE
433F C9                 RET             ;NOT A GOOD COMMAND-RETURN
4340 F3         UCLEAR: DI              ;DISABLE INTERUPTS
4341 AF                 XRA     A       ;CLEAR A
4342 D3F5               OUT     USCMD   ;OUTPUT THREE TIMES TO ENSURE
4344 D3F5               OUT     USCMD   ;THAT THE USART IS IN A KNOWN STATE
4346 D3F5               OUT     USCMD
4348 3E40               MVI     A,40H   ;CODE TO RESET USART
434A D3F5               OUT     USCMD   ;OUTPUT ON CMD CHANNEL
434C 3E5E               MVI     A,05EH  ;CE IMPLIES ASYN MODE (X16)
                                        ;           8 DATA BITS
                                        ;           ODD PARITY
                                        ;           1 STOP BIT
434E D3F5               OUT     USCMD   ;OUTPUT ON CMD CHANNEL
4350 AF                 XRA     A       ;CLEAR A, SET ZERO
4351 213942             LXI     H,TCBA  ;CLEAR TCBA AND RCBA
4354 77                 MOV     M,A
4355 23                 INX     H
4356 77                 MOV     M,A
4357 23                 INX     H
4358 77                 MOV     M,A
4359 23                 INX     H
435A 77                 MOV     M,A
435B FB                 EI              ;ENABLE INTERUPTS
435C C9                 RET             ;AND RETURN TO USER
                                        ;
                                        ;
435D 213B42     UREAD:  LXI     H,RCBA  ;CHECK READ IDLE
4360 7E                 MOV     A,M
4361 B7                 ORA     A
4362 C26B43             JNZ     UROUT
4365 23                 INX     H
4366 7E                 MOV     A,M
4367 B7                 ORA     A
4368 CA7743             JZ      URDA    ;READ IS IDLE-PROCEDE
436B 3EFE       UROUT:  MVI     A,0FEH  ;ALREADY RUNNING-ERROR STATUS
436D 217643             LXI     H,URDB  ;SET UP RETURN ADDRESS
4370 E5                 PUSH    H       ;PUSH IT INTO STACK
4371 EB                 XCHG            ;H GETS COMMAND BLOCK ADDRESS
4372 CD5B42             CALL    CLEAN   ;CALL CLEANUP ROUTINE
4375 E9                 PCHL            ;EFFECTIVELY CALLS END ROUTINE
4376 C9         URDB:   RET             ;RETURN TO USER

4377 EB         URDA:   XCHG            ;H GETS COMMAND BLOCK ADDRESS
4378 223B42             SHLD    RCBA    ;RCBA GETS COMMAND BLOCK ADDRESS
437B 3A3742             LDA     LCMD    ;GET LAST COMMAND
437E F616               ORI     16H     ;SET RXE AND DTR AND RESET ERRORS
4380 323742             STA     LCMD    ;AND RETURN TO MEMORY
4383 0F                 RRC             ;SET CARRY EQUAL TO TXE
```

```
4384 D28C43            JNC     URDC
4387 3E02              MVI     A,2
4389 323842            STA     TCMD
438C 07        URDC:   RLC
438D D3F5              OUT     USCMD   ;OUTPUT CMD
438F DBF4              IN      USDAI   ;CLEAR USART OF LEFT OVER CHARACTERS
4391 DBF4              IN      USDAI
4393 3E82              MVI     A,82H   ;MDS-CLEAR RECEIVE INTERUPT
4395 D3F3              OUT     0F3H    ;MDS
4397 3EF6              MVI     A,0F6H  ;MDS-ENABLE LEVEL THREE
4399 D3FC              OUT     0FCH    ;MDS
439B FB                EI              ;ENABLE INTERUPTS
439C C9                RET             ;RETURN TO USER


439D 213942    UWRITE: LXI     H,TCBA  ;CHECK WRITE IDLE
43A0 7E                MOV     A,M
43A1 B7                ORA     A
43A2 C26B43            JNZ     UROUT   ;BUSY-EXIT
43A5 23                INX     H
43A6 7E                MOV     A,M
43A7 C26B43            JNZ     UROUT   ;BUSY-EXIT
43AA EB                XCHG            ;OK-H GETS COMMAND BLOCK ADDRESS
43AB 223942            SHLD    TCBA    ;TCBA GETS COMMAND BLOCK ADDRESS
43AE 3A3742            LDA     LCMD    ;GET LAST COMMAND
43B1 F623              ORI     023H    ;SET RTS,DTR, AND TXEN
43B3 323742            STA     LCMD
43B6 D3F5              OUT     USCMD
43B8 3EF6              MVI     A,0F6H  ;MDS-ENABLE LEVEL THREE INTERUPTS
43BA D3FC              OUT     0FCH    ;MDS
43BC FB                EI              ;ENABLE SYSTEM INTERUPTS
43BD C9                RET             ;AND RETURN
```

```
                ;*****
                ;
                ;       USER IS A TEST PROGRAM WHICH EXERCISES USRUN
                ;
                ;*****

43BE 3EC3       USER:   MVI     A,0C3H  ;MDS-SET INTERUPT VECTOR
43C0 321800             STA     018H
43C3 216842             LXI     H,VECTOR
43C6 221900             SHLD    019H
43C9 3E43               MVI     A,'C'   ;SET GENERAL BLOCK TO A 'C'
43CB 111442             LXI     D,GBLOCK
43CE 12                 STAX    D
43CF CD2F43             CALL    USRUN
43D2 210040             LXI     H,BUFIN ;CLEAR INPUT BUFFER
43D5 AF                 XRA     A
43D6 77                 MOV     M,A
43D7 2C                 INR     L
43D8 C2D643             JNZ     $-2
43DB 210041             LXI     H,BUFOUT ;INITIALIZE OUTPUT BUFFER
43DE 75                 MOV     M,L
43DF 2C                 INR     L
43E0 C2DE43             JNZ     $-2
43E3 65                 MOV     H,L     ;REINTIALIZE CONTROL BLOCKS
43E4 2E52               MVI     L,'R'
43E6 220042             SHLD    RBLOCK
43E9 2E57               MVI     L,'W'
43EB 220A42             SHLD    TBLOCK
43EE 6C                 MOV     L,H
43EF 220642             SHLD    RCCT
43F2 221042             SHLD    TCCT
43F5 110042             LXI     D,RBLOCK ;START READ
43F8 CD2F43             CALL    USRUN
43FB 110A42             LXI     D,TBLOCK ;START WRITE
43FE CD2F43             CALL    USRUN
4401 3EFF               MVI     A,0FFH  ;LOOP WAITING COMPLETION
4403 321642             STA     FLAG    ;FLAG WILL BE SET BY COMPLETION ROUTINES
4406 3A1642             LDA     FLAG
4409 B7                 ORA     A
440A C20644             JNZ     $-4
440D 210040             LXI     H,BUFIN ;TEST INPUT BUFFER=OUTPUT BUFFER
4410 7E         COMLP:  MOV     A,M
4411 24                 INR     H
4412 BE                 CMP     M
4413 C21E44             JNZ     COMER
4416 25                 DCR     H
4417 2C                 INR     L
4418 C21044             JNZ     COMLP
441B C3BE43             JMP     USER    ;GOOD COMPARE-REPEAT TEST
441E C7         COMER:  RST     0       ;ERROR-RETURN TO MONITOR

0000                    END
```

BSTAT	00FF	BUFIN	4000	BUFOU	4100	CEND	0001
CLEAN	425B	COMER	441E	COMLP	4410	EXA	42C1
EXCHA	42BE	FLAG	4216	GBLOC	4214	GSTAT	0000
LCMD	4237	LOADA	423E	MTAB	423D	PEND	42CF
RBAD	4202	RBLOC	4200	RCBA	423B	RCCT	4206
RCR	4217	RCRA	4208	RISR	4288	RISRA	42AE
RISRB	4299	RISRE	42B9	RRCT	4204	TBAD	420C
TBLOC	420A	TCBA	4239	TCCT	4210	TCMD	4238
TCR	4227	TCRA	4212	TISR	42D4	TISRA	42EC
TRCT	420E	TUTE	4304	TUTE1	4324	TUTE2	4314
TUTE3	431C	TUTE4	4321	UCLEA	4340	URDA	4377
URDB	4376	URDC	438C	UREAD	435D	UROUT	436B
USCMD	00F5	USDAI	00F4	USDAO	00F4	USER	43BE
USRUN	432F	USTAT	00F5	UWRIT	439D	VECTO	4268
VOUT	427E						

APPENDIX A
8251 DESIGN HINTS

1. Output of a command to the USART destroys the integrity of a transmission in progress if timed incorrectly.

   Sending a command into the USART will overwrite any character which is stored in the buffer waiting for transfer to the parallel-to-serial converter in the device. This can be avoided by waiting for TxRDY to be asserted before sending a command if transmission is taking place. Due to the internal structure of the USART, it is also possible to disturb the transmission if a command is sent while a SYN character is being generated by the device. (The USART generates a SYN if the software fails to respond to TxRDY.) If this occurrence is possible in a system, commands should be transferred only when a positive-going edge is detected on the TxRDY line.

2. RxE only acts as a mask to RxRDY; it does not control the operation of the receiver.

   When the receiver is enabled, it is possible for it to already contain one or two characters. These characters should be read and discarded when the RxE bit is first set. Because of these extraneous characters the proper sequence for gaining synchronization is as follows:

   1. Disable interrupts
   2. Issue a command to enter hunt mode, clear errors, and enable the receiver (EH,ER,RxE= 1)
   3. Read USART data (it is not necessary to check status)
   4. Enable interrupts

   The first RxRDY that occurs after the above sequence will indicate that the SYN character or characters have been detected and the next character has been assembled and is ready to be read.

3. Loss of CTS or dropping TxEnable will immediately clamp the serial output line.

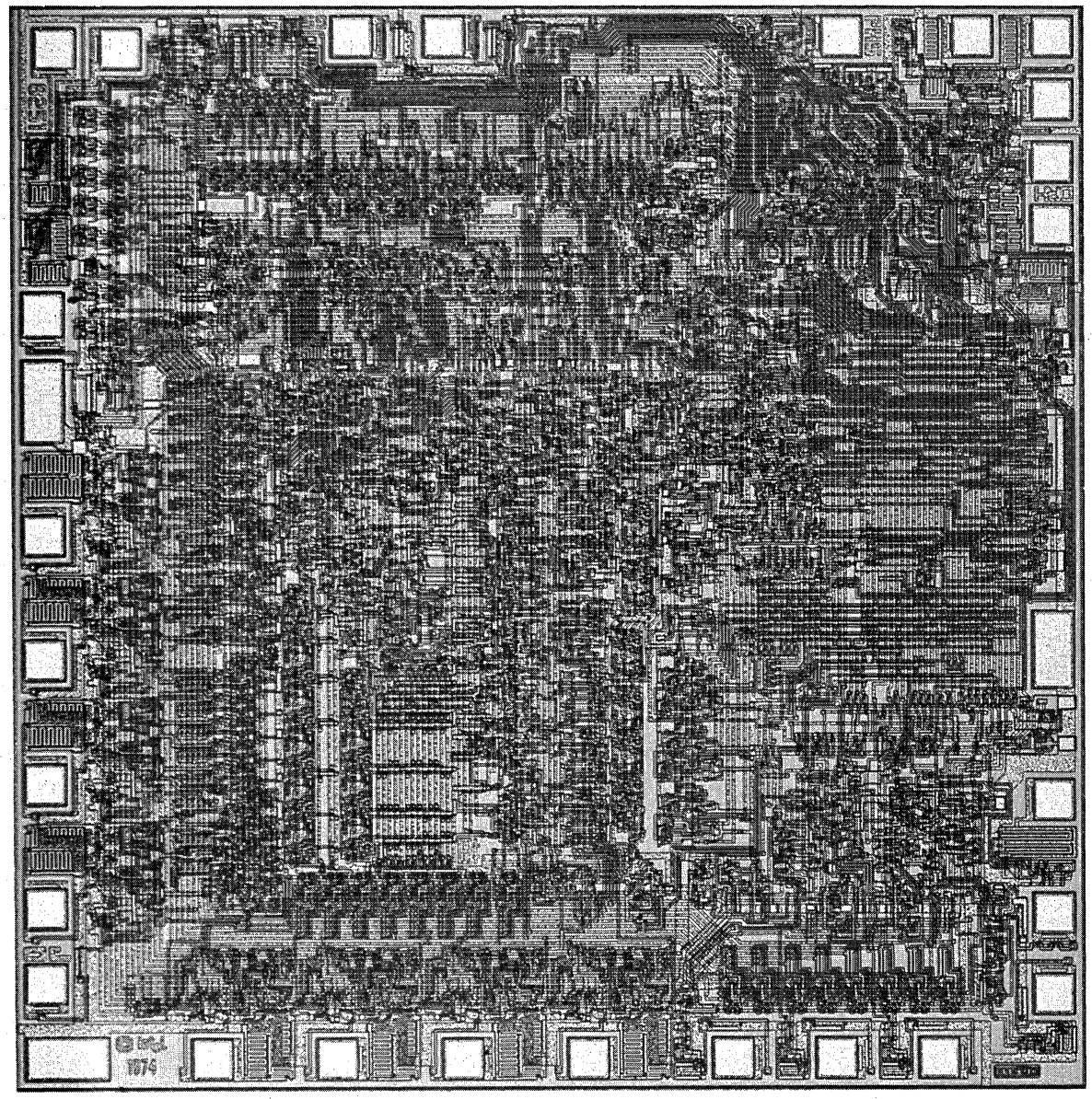
   TxEnable and RTS should remain asserted until the transmission is complete. Note that this implies that not only has the USART completed the transfer of all bits of the last character, but also that they have cleared the modem. A delay of 1 msec following a proper occurrence of TxEmpty is usually sufficient (see item 4). An additional problem can occur in the synchronous mode because the loss of TxEnable clamps the data in at a SPACE instead of the normal MARK. This problem, which does not occur in the asynchronous mode, can be corrected by an external gate combining RTS and the serial output data.

4. Extraneous transitions can occur on TxEmpty while data (including USART generated SYNs) is transferred to the parallel-to-serial converter.

   This situation can be avoided by ensuring that TxEmpty occurs during several consecutive status reads before assuming that the transmitter is truly in the empty state.

5. A BREAK (i.e., long space) detected by the receiver results in a string of characters which have framing errors.

   If reception is to be continued after a BREAK, care must be taken to ensure that valid data is being received; special care must be taken with the last character perceived during a BREAK, since its value, including any framing error associated with it, is indeterminate.



8255A Programmable Peripheral Interface Applications

by Alan Ebright

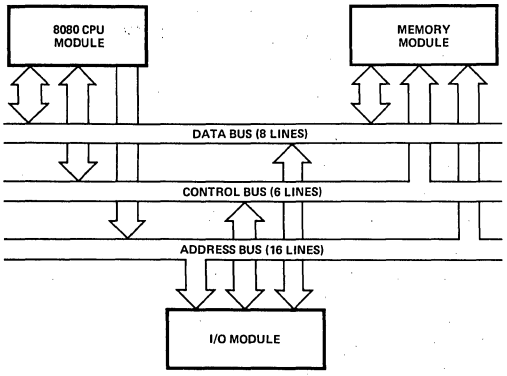
INTRODUCTION

Microprocessor-based system designs are a cost-effective solution to a wide variety of problems. When a system designer is presented with the task of selecting a microprocessor for a design, the capabilities of the microprocessor should not be the only consideration. The microprocessor should be an element of a compatible family of devices. The MCS-80 component family is a group of compatible devices which have been designed to directly address and solve the problems of microprocessor-based system design. One member of the MCS-80 component family is Intel's 8255A programmable peripheral interface chip. This device replaces a significant percentage of the logic required to support a variety of byte oriented Input/Output interfaces. Through the use of the 8255A, the I/O interface design task is significantly simplified, the design flexibility is increased, and the number of components required is reduced.

This application note presents detailed design examples from both the hardware and software points of view. Since the 8255A is an extremely flexible device, it is impossible to list all of the applications and configurations of the device. A number of designs are presented which may be modified to fulfill specific user interface requirements.

Detailed design examples are discussed within the context of the 8080 system shown in Figure 1. The basic 8080 system is composed of the CPU module, memory module, and the I/O module. CPU module and memory module design are discussed within other Intel publications. This application note deals exclusively with I/O module design.

It is assumed that the reader is familiar with the MCS-80 User's Manual and/or the MCS-85 User's Manual, particularly the 8255A device description.



OVERVIEW OF THE 8255A

The 8255A block diagram shown in Figure 2 has been divided into three sections: 8080 CPU Module Interface, Peripheral Interface, and the Internal Logic.

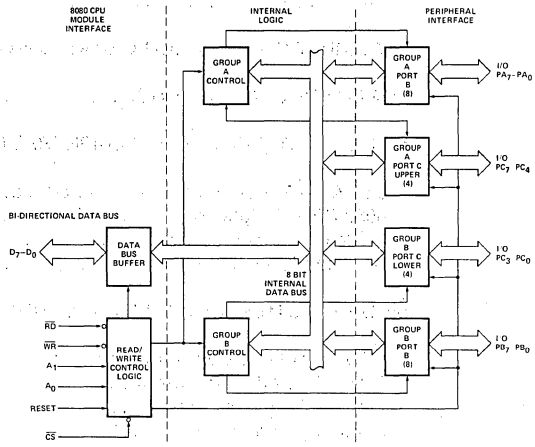


Figure 2. 8255A Block Diagram

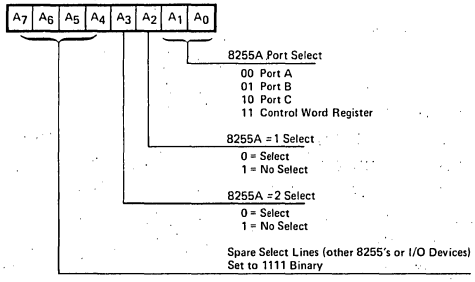
8080 CPU MODULE INTERFACE

The 8255A is a compatible member of the MCS-80 component family and, therefore, may be directly interfaced to the 8080. Figure 3 displays one method of interconnecting the 8255A and an 8080 CPU module. The 8080 CPU module consists of the 8080A CPU, the 8224 Clock Generator, and the 8228 System Controller. The system shown in Figure 3 utilizes a linear select scheme which dedicates an address line as an exclusive enable (chip select) for each specific I/O device. The chip select signal is used to enable communication between the selected 8255A and the 8080 CPU. I/O Ports A, B, C, or the Control Word Register are selected by the two port select signals (A_1, A_0). These signals (A_1 and A_0) are driven by the least significant bits of the address bus. The I/O port select characters required by this configuration are shown in Figure 4.

When a system utilizing the linear select scheme is implemented, a maximum of six I/O devices may be selected. If more than six I/O devices must be addressed, the six device select bits must be encoded to generate a maximum of 64 device select lines. Note that when large systems are implemented, bus loading considerations may require that bus drivers be included in the CPU module. The MCS-80 component family contains parts which are designed to perform this function (8216, 8226).

The 8255A I/O read ($\overline{RD}$) and I/O write ($\overline{WR}$) signals may be directly driven by the 8228. This results in an isolated I/O architecture where 8080 Input/Output instructions are used to reference an independent I/O address space. An alternate approach is memory mapped I/O. This architecture treats an area of memory as the I/O address space. The memory mapped I/O architecture utilizes 8080 memory reference instructions to access the I/O address space. Interfacing with the 8080 is outlined in Chapter 3 of the "8080 Microcomputer User's Manual".

The most important feature of the 8255A to 8080 CPU Module Interface is that for small system designs the 8255A may be interfaced directly to the standard MCS-80 component family with no external logic. Minimum external logic is required in large system designs.



Port Selected	Hexadecimal Port Select Character (Used with IN or OUT Instructions)
Port A 8255A = 1	F8
Port B 8255A = 1	F9
Port C 8255A = 1	FA
Control Word Register 8255A = 1	FB
Port A 8255A = 2	F4
Port B 8255A = 2	F5
Port C 8255A = 2	F6
Control Word Register 8255A = 2	F7

Figure 4. I/O Port Select Characters

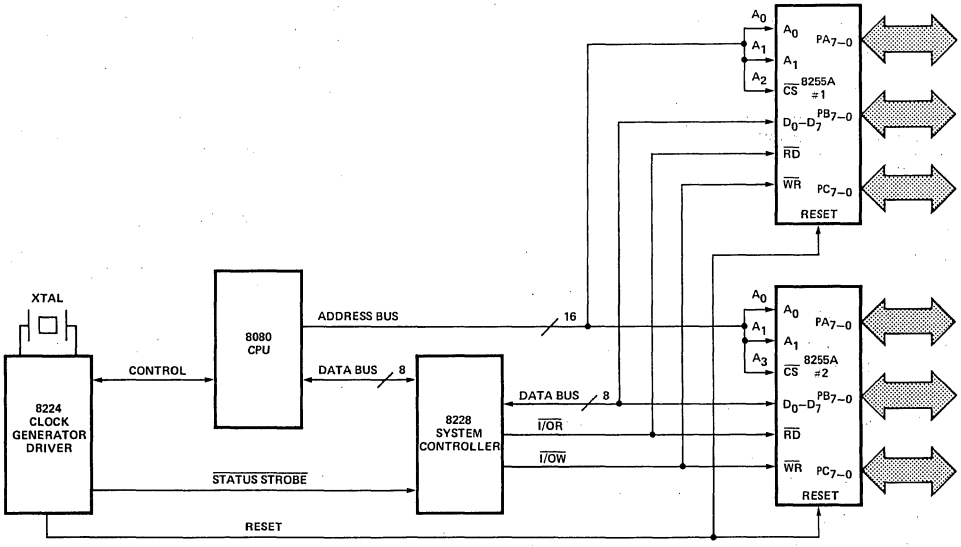


Figure 3. Linear Select 8255A Interconnect

PERIPHERAL INTERFACE SECTION

The peripheral interface section contains 24 peripheral interface lines, buffers, and control logic. The characteristics and functions of the interface lines are determined by the operating mode selected under program control. The flexibility of the 8255A is due to the fact that the device is programmable. Three modes of operation may be selected under program control: Mode 0 – Basic Input/Output, Mode 1 – Strobed Input/Output with interrupt support, and Mode 2 – Bidirectional bus with interrupt support. Through selecting the correct operating mode, the interface lines may be configured to fulfill specific interface requirements. The characteristics of the interface lines within each mode must be understood so that the designer may utilize the 8255A to achieve the most efficient design. Table I lists the basic features of the peripheral interface lines within each mode group. Figure 5 shows the grouping of the peripheral interface lines within each mode.

Table I. Features of Peripheral Interface Lines

Mode 0 – Basic Input/Output
Two 8-bit ports Two 4-bit ports with bit set/reset capability Outputs are latched Inputs are not latched
Mode 1 – Strobed Input/Output
One or two strobed ports Each Mode 1 port contains: 8-bit data port 3 control lines Interrupt support logic Any port may be input or output If one Mode 1 port is used, the remaining 13 lines may be configured in Mode 0. If two Mode 1 ports are used, the remaining 2 bits may be input or output with bit set/reset capability.
Mode 2 – Strobed Bidirectional Bus
One bidirectional bus which contains: 8-bit bidirectional bus supported by Port A 5 control lines Interrupt support logic Inputs and outputs are latched The remaining 11 lines may be configured in either Mode 0 or Mode 1.

One feature of Port C is important to note. Each Port C bit may be individually set and reset. Through the use of this feature, device strobes may be easily generated by software without utilizing external logic. The Mode 1 and Mode 2 configurations use a number of the Port C lines for interrupt control lines. Thus, the 8255A contains a large portion of the logic required to implement an interrupt driven I/O interface. This feature simplifies interrupt driven hardware design and saves a significant amount of the external logic that is normally required when less powerful I/O chips are used. In fact, the design examples contained in this application note describe how interrupt driven interfaces may be designed such that the only interrupt control logic required is that contained in the 8255A.

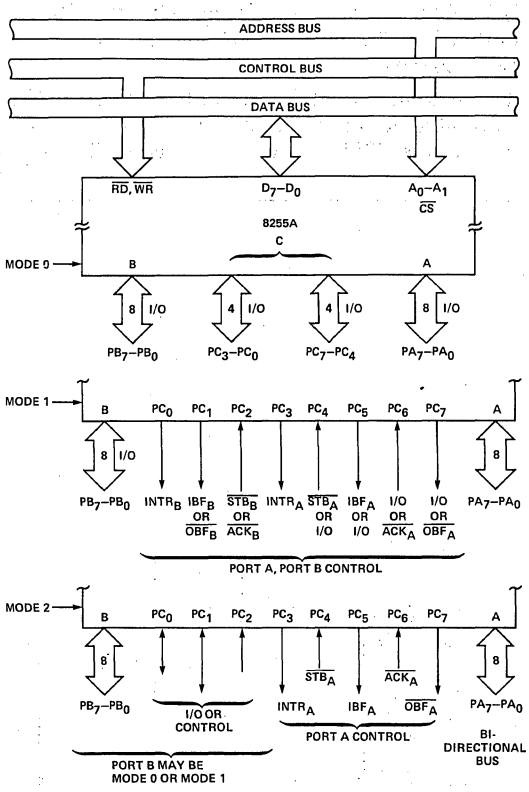


Figure 5. Grouping of Peripheral Interface Lines

INTERNAL LOGIC SECTION

The internal logic section manages the transfer of data and control information on the internal data bus (refer to Figure 2). If the port select lines (A_1 and A_0) specify Ports A, B, or C, the operation is an I/O port data transfer. The internal logic will select the specified I/O port and perform the data transfer between the I/O port and the CPU interface. As was previously mentioned, both the functional configuration of each port and bit set/reset on Port C are controlled by the system's software. When the control word register is selected, the internal logic performs the operation described by the control word. The control word contains an opcode field which defines which of the two functions are to be performed (mode definition or bit set/reset).

Mode Definition

When the opcode field (Bit 7) of the control word is equal to a one, the control word is interpreted by the 8255A as a mode definition control word. The mode definition control word (shown in Figure 6) is used to specify the configuration of the 24 8255A peripheral interface lines. The system's software may specify the modes of Port A and Port B independently. Port C may be treated independently or divided into two portions as required by the Port A and Port B mode definitions.

Example #1: This example demonstrates how a mode control word is constructed and issued to an 8255A. The mode control word is passed to the device through the use of an output instruction that references an 8080 I/O port address. The value of the I/O port address is determined by the 8080 CPU interface implemented. This example references the I/O port addresses realized by the simple 8080 to 8255A interface shown in Figure 3.

If an 8255A is to be configured through the use of the mode control word interface as:

Port A	Mode 0 Input
Port B	Mode 1 Output
Port C	Bits PC_7–PC_4 Output
Port C	Bit 3 Input

The following mode control word is used:

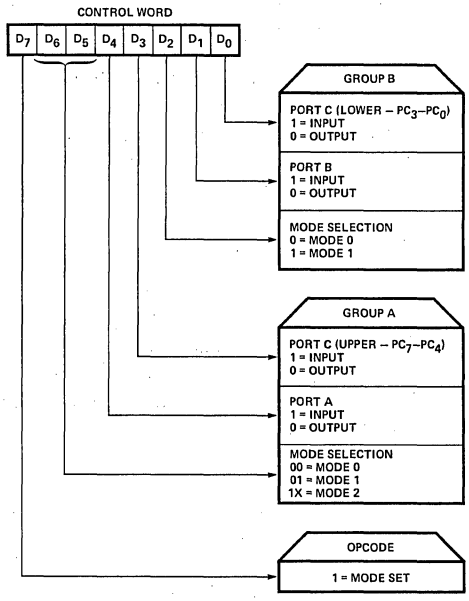
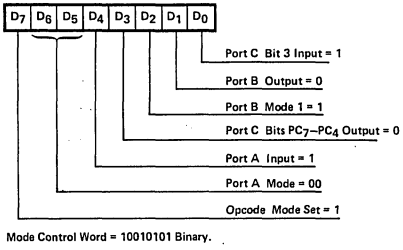


Figure 6. Mode Definition Control Word



The assembly language program is:

```
CWR         EQU   0FBH          ;8255A #1 CONTROL WORD REGISTER
;*****
;       ISSUE MODE CONTROL WORD
;*****
            MVI   A,10010101B   ;GET MODE CONTROL WORD
            OUT   CWR           ;OUTPUT TO 8255A #1 CONTROL WORD
                                ;REGISTER
```

Bit Set/Reset

When the opcode field (Bit 7) of the control word is equal to a zero, the control word is interpreted by the 8255A as a Port C bit set/reset command word (see Figure 7). Through the use of the bit set/reset command, any of the 8 bits on Port C may be independently set or reset. Note that control word bits 6–4 are not used. Bits 6–4 should be set to zero.

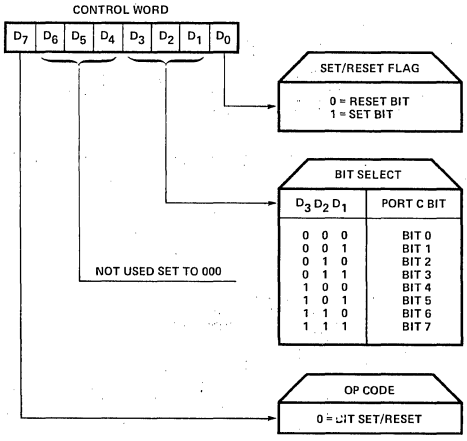
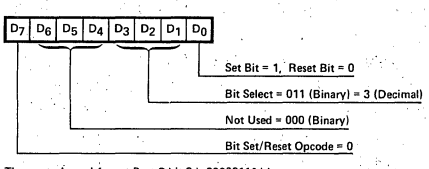


Figure 7. Bit Set/Reset Control Word

Example #2: This example demonstrates how a Port C bit set/reset control word is constructed and issued to an 8255A. The bit set/reset control word is passed to the device through the use of an output instruction that references an 8080 I/O port address. The value of the I/O port address is determined by the 8080 CPU interface implemented. This example references the I/O port addresses realized by the simple 8080 to 8255A interface shown in Figure 3.

Control word (see Figure 7).



The control word for set Port C bit 3 is 00000111 binary.
The control word for reset Port C bit 3 is 00000110 binary.

The assembly language program is:

```
CWR     EQU     0FBH            ;8255A #1 CONTROL WORD REGISTER
;*****
;       SET BIT 3
;*****
        MVI     A, 00000111B    ; GET SET BIT 3 CONTROL WORD
        OUT     CWR             ; OUTPUT TO 8255A #1 CONTROL WORD REGISTER
;*****
;       RESET BIT 3
;*****
        MVI     A, 00000110B    ; GET RESET BIT 3 CONTROL WORD
        OUT     CWR             ; OUTPUT TO 8255A #1 CONTROL WORD REGISTER
```

NOTE: An MVI instruction is used to load the reset bit 3 control word into the A register. Since it is known that the set bit control word is already in the A register, a "DCR A" Instruction could be used to generate the correct control word and save one byte of code.

00000111 – 1 = 00000110 (RESET BIT 3 CONTROL WORD)

Example #3: This example demonstrates one simple method of performing a bit set/reset operation on Ports A and B. The state of any output port may be determined by reading the port. The assembly language program which may be used to set/reset Port A or B bits is:

```
PORTA   EQU     0F8H            ;8255A #1 PORT A
;*****
;       SET BIT 0
;*****
        IN      PORTA           ; GET STATE OF PORT
        ORI     01H             ; SET BIT 0
        OUT     PORTA           ; OUTPUT TO PORT
;*****
;       RESET BIT 0
;*****
        IN      PORTA           ; GET STATE OF PORT
        ANI     0FEH            ; RESET BIT 0
        OUT     PORTA           ; OUTPUT TO PORT
```

INTERRUPT CONTROL LOGIC STATUS WORDS

As previously mentioned, the 8255A Mode 1 and Mode 2 configurations support interrupt control logic. If a read of Port C is issued when the 8255A is configured in Mode 1, the software will receive the Mode 1 status word shown in Figure 8. The bits in the status word correspond to the state of the associated Port C lines (buffer full, interrupt request, etc.). The INTE bit shown in the status word corresponds to the interrupt enable flip-flop contained in the 8255A. This signal is not available externally. The structure of the Mode 1 status word varies as a function of the mode of the 8255A. Example #4 shows the status word which results from reading Port C from an 8255A which is configured with Port A Mode 1 input and Port B Mode 1 output.

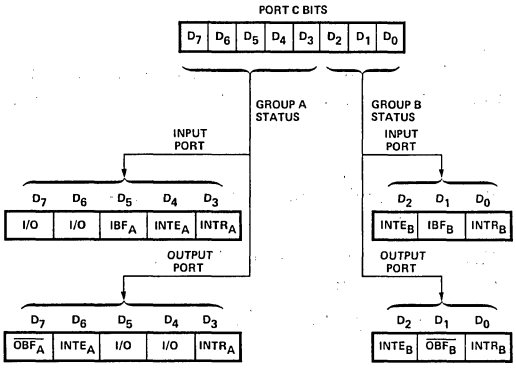


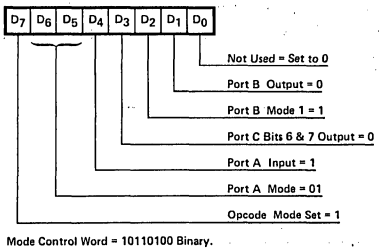
Figure 8. Mode 1 Status Word

Example #4 – MODE 1 STATUS WORD

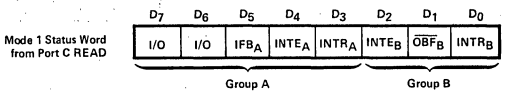
If an 8255A is to be configured through the use of the mode control word interface as:

Port A	Mode 1 Input
Port B	Mode 1 Output
Port C	Bits 6 & 7 Output

The following mode control word is used:



After the 8255A mode control word has been issued, a READ of Port C will obtain the following Mode 1 status word:



NOTE: The Port C I/O bits D_7 and D_6 should be modified through the use of the Port C bit set/reset command word. If a write to Port C is issued, the $INTE_A$ and $INTE_B$ bits may be inadvertently modified by the user. The IBF_A, $INTR_A$, $\overline{OBF}_B$, and $INTR_B$ bits will not be modified by either a write to Port C or a bit set/reset command. These four bits always reflect the state of the interrupt control logic.

Note that the Mode 2 status word (shown in Figure 9) differs from the Mode 1 status word. The format of the status word data bits D_2–D_0 are defined by the specification of the Port B configuration. Example #5 shows the structure of the Mode 2 status word when the 8255A is configured with Port A Mode 2 (bidirectional bus) and Port B Mode 1 input.

The Mode 1 and Mode 2 status words reflect the state of the interrupt logic supported by the 8255A.

Example #6 demonstrates how the interrupt enable bits are controlled through the use of the Port C bit set/reset feature. The application examples provide a more detailed explanation of the use of the Port C status word in the Mode 1 and Mode 2 configurations.

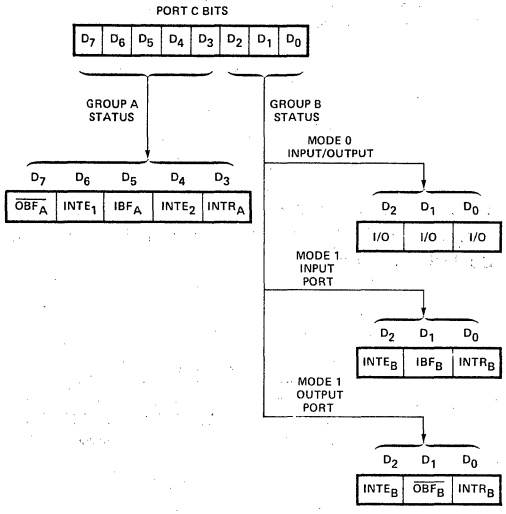


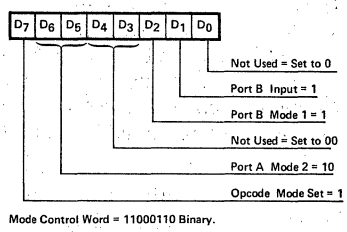
Figure 9. Mode 2 Status Word

Example #5 – MODE 2 STATUS WORD

If the 8255A is to be configured as follows:

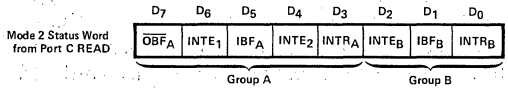
Port A	Mode 2 Bidirectional Bus
Port B	Mode 1 Input

The following mode control word is used:



Mode Control Word = 11000110 Binary.

After the 8255A mode control word has been issued, a read of Port C will obtain the following Mode 2 status word:



Example #6 – MODE 2 INTERRUPT ENABLE/DISABLE

The Mode 2 status word shown in Figure 9 contains two interrupt enable bits:

$INTE_1$ – Bit 6 – Enable output interrupts
$INTE_2$ – Bit 4 – Enable input interrupts

Bit set/reset control words may be constructed which may be used to control the INTE bits.

Set Bit 6 (Enable Output Interrupts) = 00001101 Binary

Reset Bit 6 (Disable Output Interrupts) = 00001100 Binary

Set Bit 4 (Enable Input Interrupts) = 00001001 Binary

Reset Bit 4 (Disable Input Interrupts) = 00001000 Binary

The control words shown were constructed from the standard bit set/reset format shown in Figure 7.

The value of CWR used in the following program example corresponds to the 8080 configuration shown in Figure 3.

```
CWR     EQU     0FBH            ; 8255A #1 CONTROL WORD REGISTER
; *****
        ENABLE INTERRUPTS FOR MODE 2 OUTPUT (SET PORT C BIT 6)
; *****
        MVI     A, 00001101B    ; GET SET BIT 6 CONTROL WORD
        OUT     CWR             ; OUTPUT TO 8255 #1 CONTROL WORD REGISTER
; *****
;       DISABLE INTERRUPTS FOR MODE 2 OUTPUT (RESET PORT C BIT 6)
; *****
        MVI     A, 00001100B    ; GET RESET BIT 6 CONTROL WORD
        OUT     CWR             ; OUTPUT TO 8255A #1 CONTROL WORD REGISTER
```

SOFTWARE CONSIDERATIONS

Regardless of the mode selected, the software must always issue the correct mode control word after a reset of the device. Generally, an initialization routine is constructed which issues the correct mode control word, sets up the initial state of the control lines, and initializes any program internal data.

Many of the software requirements of the 8255A vary as a function of the mode selected. The simplest mode supported by the device is Mode 0 (Basic Input/Output). Generally, Mode 0 is used for simple status driven device interfaces (no interrupts). Figure 10 illustrates sample software that could be used to support such interfaces. Most devices support a BUSY or READY signal which is used to determine when the device is ready to input or output data and a DATA STROBE which is used to request data transfer (DATA STROBE may easily be generated with the Port C bit set/reset feature). In the Mode 0 configuration, Ports A and B are used to input/output byte oriented data. Port C is used to input 8255A status, peripheral status and to drive peripheral control lines.

When the Mode 1 and Mode 2 configurations are used, the software is generally required to support interrupts. Software routines written for an interrupt driven environment tend to be more complex than status driven routines. The added complexity is due to the fact that interrupt driven systems are constructed such that other software tasks are run while the I/O transaction is in progress. A software routine that controls a peripheral device is generally referred to as a device driver. One method of implementing an interrupt driven device driver is to partition the device driver into a "Command Processor" and an "Interrupt Service Routine". The command processor is the module that validates and initiates user program requests to the device driver. A common method of passing information between the various software programs is to have the requesting routine provide a device control block in memory. A sample device control block is shown in Table II.

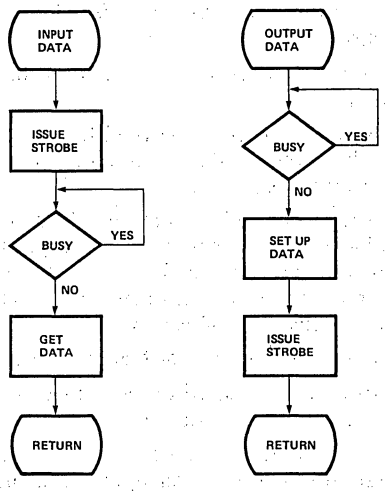


Figure 10. Sample Status Driven Software Flowchart

Table II. Sample Device Control Block

NAME	DESCRIPTION
Status	This 1-byte field is used to transmit the status of the I/O transaction (busy, complete, etc.).
Opcode	This 1-byte field defines the type of I/O (READ, WRITE, etc.).
Buffer Address	This 2-byte field specifies the source/destination of the data block.
Character Count	This 1-byte field is a count of the number of characters involved in the transaction.
Character Transferred Count	This 1-byte count of the number of characters which were actually transferred.
Completion Address	This 2-byte field is the address of the user supplied completion routine which will be called after the I/O has been performed.

The command processor validates the transaction and initiates the operation described by the control block. Control is then returned to the requestor so that other processing may proceed. The interrupt service routine processes the remainder of the transaction.

The interrupt service routine supports the following functions:

1. The state of the machine (registers, status, etc.) must be saved so that it may be restored after the interrupt is processed.
2. The source of the interrupt must be determined. The hardware may support a register which indicates the interrupting device, or the software may poll the devices through interrogating the Port C status word of each 8255A.
3. Data must be passed to or from the device.
4. Control must be passed to the requesting routine at the completion of the I/O.
5. The state of the machine must be restored before returning to the interrupted program.

Figures 11 and 12 are simplified flowcharts of one of the many methods of implementing command processor and interrupt service routine modules.

The rest of this application note presents specific application examples. All of the 8080 assembly language programs supplied with the application examples use the standard Intel 8080 assembly language mnemonics. The programs discussed use the program equate statement to specify all hardware related data. Equate statements are used so that all references to an I/O port may be changed through a simple reassignment of the port address in the equate statement.

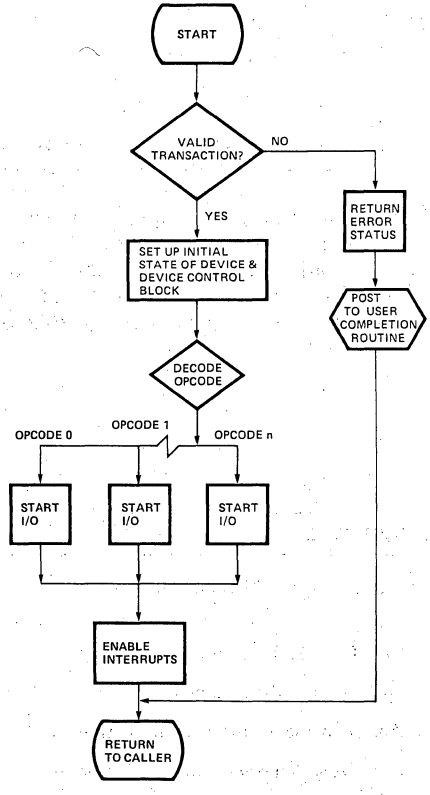


Figure 11. Command Processor

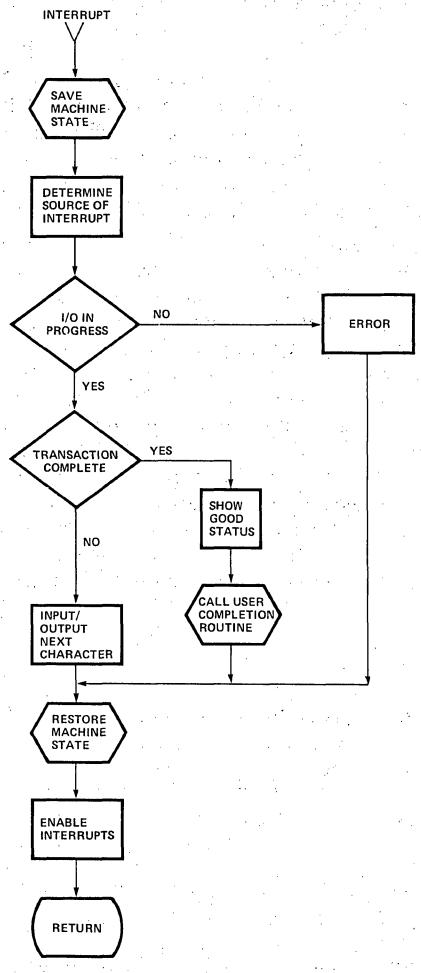


Figure 12. Interrupt Service Routine

MODE 0 – STATUS DRIVEN PERIPHERAL INTERFACE

This design example shows how a single 8255A in Mode 0 may be used to develop a status driven interface (no interrupts) for the Centronics 306 character printer, the Remex paper tape punch, and the Remex paper tape reader.

8255A To Peripheral Hardware Interface

The first step in the design is to examine the specification for the peripheral devices and identify the control and data signals which must be supported by the interface. Table III lists the signals which were chosen to be supported by the 8255A interface. All three of the devices support the standard BUSY/DATA STROBE interface discussed previously (see Figure 10). Figure 13 is a block diagram of the interface design. The 8255A Port A is configured as a Mode 0 output port which is used to support the printer and the paper tape punch data bus. Port B is configured as a Mode 0 input port and is used to input the paper tape reader data. Three of the Port C lower bits (PC_2–PC_0) configured in input mode are used to input the device busy indications. Three of the Port C upper bits (PC_6–PC_4) configured in output mode are used to support the device strobe signals required by each device.

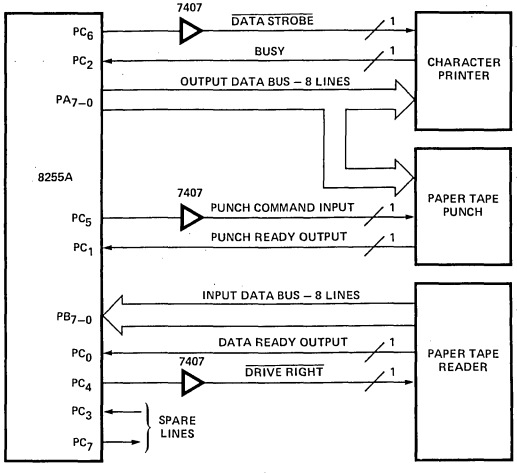
The drive requirements of the interface lines are a function of the peripheral interface circuitry, the length of the interface cable, and the environment in which the unit is running. In this particular design example, all output lines from the 8255A to the peripherals were buffered through a 7407 buffer/driver. The input lines from the peripherals were fed directly into the Port C and Port B inputs.

Table III. Peripheral Interface Signals

CHARACTER PRINTER	
Name:	DATA 0–DATA 7
Definition:	Input data levels. A high signal represents a binary 1 and a low signal represents a binary 0. These eight lines are the data lines to the printer.
Name:	$\overline{\text{DATA STROBE}}$
Definition:	A 0.5 μsec pulse (minimum) used to transfer data from the 8255A to the printer.
Name:	BUSY
Definition:	The level indicating that the printer cannot receive data.
PAPER TAPE PUNCH	
Name:	TRACKS 1–8 DATA INPUT
Definition:	Input data levels. A high signal causes a hole to be punched on the associated track. These eight lines are the data lines to the printer.
Name:	PUNCH COMMAND INPUT
Definition:	A true condition moves the tape and initiates punching the tape. This signal is actually a data strobe.
Name:	PUNCH READY OUTPUT
Definition:	True signal indicates that the punch is ready to accept a punch command. This is the punch busy line.
PAPER TAPE READER	
Name:	DATA TRACK OUTPUTS
Definition:	True signal indicates data track hole. These eight lines are the data lines from the punch.
Name:	$\overline{\text{DRIVE RIGHT}}$
Definition:	True signal drives the tape to the right and reads a character. This signal is actually the data strobe (initiate read signal).
Name:	DATA READY OUTPUT
Definition:	True signal indicates data track outputs are in "On character" condition. This signal is the reader busy line.

8080 CPU Module To 8255A Interface

The schematic of the completed hardware design is shown in Figure 14. The CPU module design shown is the design which was implemented for Intel's SDK 80 kit board. The 8255A is addressed through the use of an isolated I/O architecture utilizing a linear select scheme. Address bits A_1 and A_0 are used to select the 8255A port. Address bit A_3 is the exclusive enable for 8255A #1. Examination of the schematic shows that all of the 8255A interface lines are directly driven by the CPU module.



NOTE:
1. OUTPUT DATA BUS BUFFERED WITH 7407.
2. ALL 8255A OUTPUT LINES ARE PULLED UP TO +5 V AT THE PERIPHERAL.

Figure 13. Interface Block Diagram

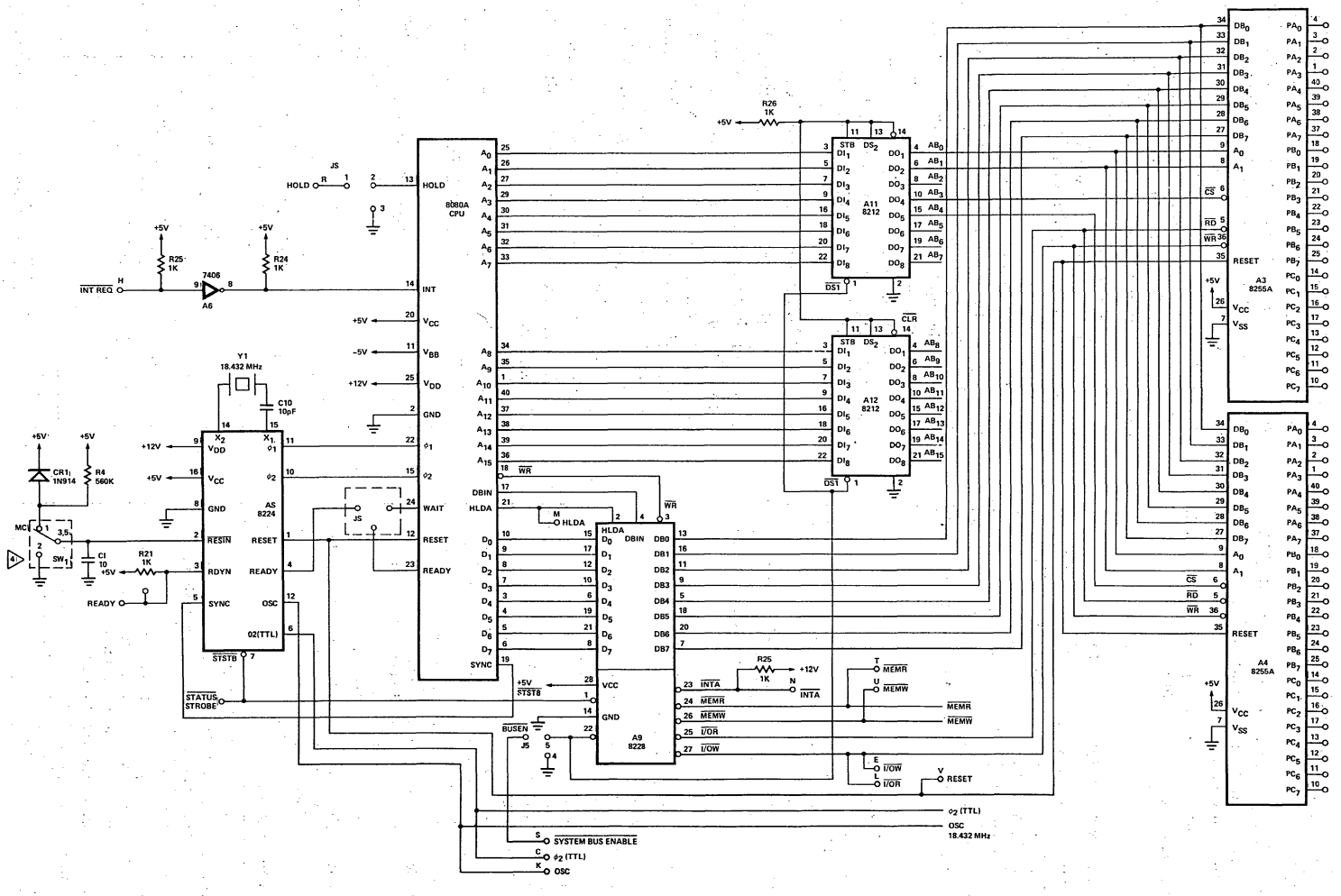
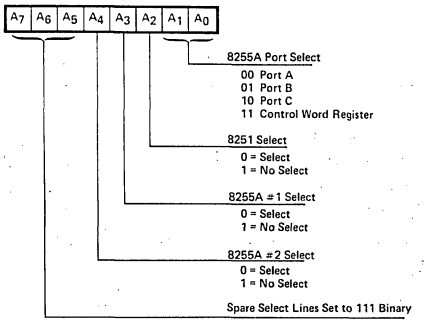


Figure 14. SDK 80 Schematic

Mode 0 Interface Software

An initialization routine and three device drivers (one for each peripheral device) are required to support the peripheral interface. The I/O port addresses implemented by the hardware are shown in Figure 15. The unused chip select bits are set to one so that chip select conflicts will not result if the unused bits are required by an expanded system.



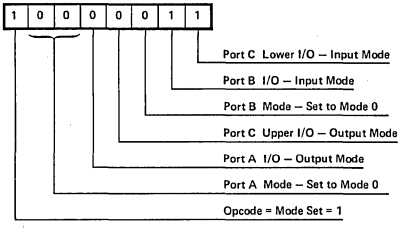
Port Selected	Port Select Character (In Hexadecimal)
Port A 8255A #1	F4
Port B 8255A #1	F5
Port C 8255A #1	F6
Control Word Register 8255A #1	F7
Port A 8255A #2	EC
Port B 8255A #2	ED
Port C 8255A #2	EE
Control Word Register 8255A #2	EF

Figure 15. I/O Port Addresses

Note that the initialization routine issues the mode control word (shown in Figure 16). It also sets the low true DATA STROBE signals to an inactive (high) state.

Hardware Requirements:

Port A – Output Mode 0
Port C Upper Bits C_7–C_4 – Output Mode 0
Port C Lower Bits C_3–C_0 – Input Mode 0
Port B – Input Mode 0



Mode Control Word = 100000011 Binary = 83 HEX.

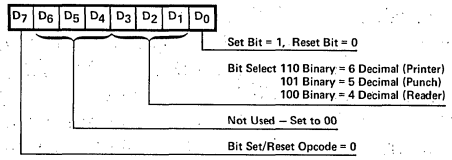
Figure 16. Mode Control Word

```
ISIS 8080 MACRO ASSEMBLER, V1.0                 PAGE 1
MODE ZERO EXAMPLE

                        TITLE 'MODE ZERO EXAMPLE'
                ;*****
                ;
                ;       CHARACTER PRINTER, PAPER TAPE PUNCH, PAPER TAPE READER
                ;       MODE ZERO EXAMPLE
                ;
                ;*****
                ;
                ;*****
                ;       PROGRAM EQUATES
                ;*****
00F4            PORTA   EQU     0F4H    ; 8255 PORT A
                PORTB   EQU     0F5H    ; 8255 PORT B
00F6            PORTC   EQU     0F6H    ; 8255 PORT C
00F7            CWR     EQU     0F7H    ; 8255 CONTROL WORD REGISTER
                ;*****
                ;
                ;       INITIALIZATION CONTROL WORD
                ;
                ;               USED TO CONFIGURE THE 8255 AS FOLLOWS:
                ;
                ;                       PORT A - OUTPUT MODE ZERO
                ;                       PORT B - INPUT MODE ZERO
                ;                       PORT C (UPPER) - OUTPUT
                ;                       PORT C (LOWER) - INPUT
                ;
                ;*****
0083            ICW     EQU     10000011B       ; INITIALIZATION CONTROL WORD
                ;*****
                ;
                ;       SET/RESET CONTROL WORDS FOR GENERATION OF DATA STROBES
                ;       ON PORT C.
                ;
                ;*****
000D            LPSON   EQU     00001101B       ; PRINTER DATA STROBE ON
000C            LPSOF   EQU     00001100B       ; PRINTER DATA STROBE OFF
000B            PNSON   EQU     00001011B       ; PUNCH DATA STROBE ON
000A            PNSOF   EQU     00001010B       ; PUNCH DATA STROBE OFF
0009            RDSON   EQU     00001001B       ; READER DATA STROBE ON
0008            RDSOF   EQU     00001000B       ; READER DATA STROBE OFF
                ;*****
                ;       BIT MASK FOR DEVICE BUSY CHECK
                ;*****
0004            LPBSY   EQU     04H     ; LINE PRINTER BUSY
0002            PNBSY   EQU     02H     ; PUNCH BUSY
0001            RDBSY   EQU     01H     ; READER BUSY
```

```
ISIS 8080 MACRO ASSEMBLER, V1.0                 PAGE 2
MODE ZERO EXAMPLE - INITIALIZATION ROUTINE

                ;*****
                ;       PROGRAM ORIGIN
                ;*****
3000                    ORG     03000H
                ;*****
                ;       INITIALIZATION ROUTINE
                ;       A REGISTER MODIFIED
                ;*****
                INIT:
3000 3E83               MVI     A,ICW   ; GET INITIALIZATION CONTROL WORD
3002 D3F7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
                ;*****
                ;       SET ALL LOW TRUE DATA STROBES ON
                ;*****
3004 3E0D               MVI     A,LPSON ; GET CONTROL WORD TO TURN ON PRINTER DATA STROBE
3006 D3F7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
3008 3E09               MVI     A,RDSON ; GET CONTROL WORD TO TURN ON READER DATA STROBE
300A D3F7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
300C C9                 RET             ; RETURN TO CALLER
```

The three peripheral drivers which follow all have the basic structure discussed previously. Consider the printer routine. Here the user routine places an ASCII data character in the C-register and passes control to the LPST location through a subroutine call. The printer driver interrogates the status of the printer by reading Port C. If the printer is busy, the routine will loop until the printer is idle. When the printer is ready to accept a data character, the character is placed on the Port A lines and a DATA STROBE is generated. After generating the DATA STROBE, the driver executes a subroutine return to the caller.

The DATA STROBE signals to the devices are generated through the use of the Port C bit set/reset feature. The bit set/reset control words used are shown in Figure 17.



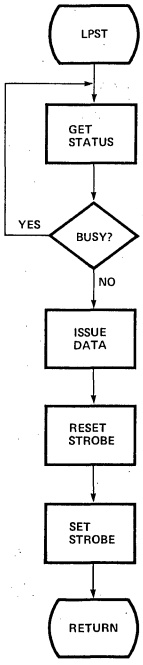
The control word for set Printer DATA STROBE (PC 6) = 00001101 binary.
The control word for reset Printer DATA STROBE (PC 6) = 00001100 binary.
The control word for set Punch DATA STROBE (PC 5) = 00001011 binary.
The control word for reset Punch DATA STROBE (PC 5) = 00001010 binary.
The control word for set Reader DATA STROBE (PC 4) = 00001001 binary.
The control word for reset Reader DATA STROBE (PC 4) = 00001000 binary.

Figure 17. Bit Set/Reset Control Words

Summary/Conclusions

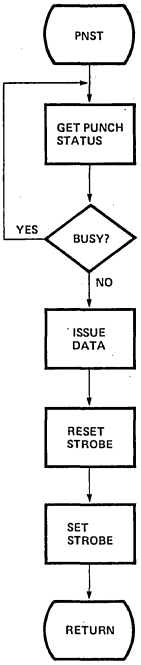
This design example discussed the basic hardware and software required to handle a simple device interface. The 8255A will easily accommodate a more complex interface design which utilizes additional interface lines supported by the peripheral. For instance, one of the spare Port C output lines may be used to control the punch direction. Support of this additional feature would require minor modification of the device driver so that the punch direction line could be specified by the user routine.

Through consideration of this example, the use of the 8255A in Mode 0 should become evident.



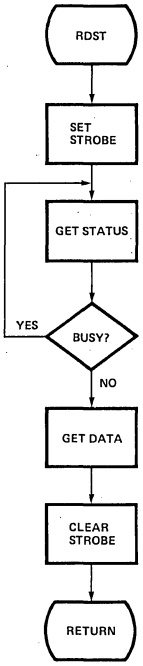
PRINTER DRIVER

```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 3
MODE ZERO EXAMPLE - CHARACTER PRINTER DRIVER

                ;*****
                ;
                ;       CHARACTER PRINTER DRIVER
                ;
                ;               INPUTS : CHARACTER TO PRINT IN C-REG
                ;               OUTPUTS: CHARACTER TO PRINTER
                ;
                ;       A REGISTER MODIFIED
                ;
                ;*****
                LPST:
300D DBF6               IN      PORTC   ; GET STATUS OF PRINTER
300F E604               ANI     LPBSY   ; SEE IF BUSY
                        JNZ     LPST    ; IF BUSY - JUMP TO LPST (WAIT LOOP)
                ;*****
                ;       PRINTER IS IDLE - OUTPUT A CHARACTER
                ;*****
3014 79                 MOV     A,C     ; GET DATA BYTE SUPPLIED BY CALLER
3015 D3F4               OUT     PORTA   ; OUTPUT DATA TO DATA LINES
3017 3E0C               MVI     A,LPSOF ; GET DATA STROBE CONTROL WORD
3019 D3F7               OUT     CWR     ; RESET DATA STROBE (LOW TRUE SIGNAL)
301B 3C                 INR     A       ; GENERATE SET DATA STROBE CONTROL WORD
301C D3F7               OUT     CWR     ; SET DATA STROBE
301E C9                 RET             ; RETURN TO CALLER
```



PUNCH DRIVER

```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 4
MODE ZERO EXAMPLE - PAPER TAPE PUNCH DRIVER

                ;*****
                ;
                ;       PAPER TAPE PUNCH DRIVER
                ;
                ;               INPUTS : DATA TO PUNCH IN C-REGISTER
                ;               OUTPUTS: DATA TO PUNCH
                ;
                ;       A REGISTER MODIFIED
                ;
                ;*****
                PNST:
301F DBF6               IN      PORTC   ; GET STATUS OF PUNCH
3021 E602               ANI     PNBSY   ; SEE IF BUSY
3023 C21F30             JNZ     PNST    ; IF BUSY - JUMP TO PNST (WAIT LOOP)
                ;*****
                ;       PUNCH IS IDLE - OUTPUT A CHARACTER
                ;*****
3026 79                 MOV     A,C     ; GET DATA BYTE SUPPLIED BY CALLER
                        OUT     PORTA   ; OUTPUT DATA TO DATA LINES
3029 3E0B               MVI     A,PNSON ; SET DATA STROBE CONTROL WORD
302B D3F7               OUT     CWR     ; SET DATA STROBE
302D 3D                 DCR     A       ; GENERATE RESET DATA STROBE CONTROL WORD
302E D3F7               OUT     CWR     ; RESET DATA STROBE
3030 C9                 RET             ; RETURN TO CALLER
```



READER DRIVER

```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 5
MODE ZERO EXAMPLE - PAPER TAPE READER DRIVER

                ;*****
                ;
                ;       PAPER TAPE READER DRIVER
                ;
                ;               INPUTS : DATA FROM READER
                ;               OUTPUTS: CHARACTER TO USER IN C-REGISTER
                ;
                ;       A AND C REGISTER MODIFIED
                ;
                ;*****
                RDST:
3031 3E08               MVI     A,RDSOF ; GET STROBE CONTROL WORD (LOW TRUE SIGNAL)
3033 D3F7               OUT     CWR     ; SET DATA STROBE
                RDLP:
3035 DBF6               IN      PORTC   ; GET STATUS OF DEVICE
3037 E601               ANI     RDBSY   ; SEE IF BUSY
3039 C23530             JNZ     RDLP    ; IF BUSY - LOOP UNTIL IDLE
                ;*****
                ;       READER NOT BUSY - GET CHAR AND CLEAR STROBE
                ;*****
303C DBF5               IN      PORTB   ; GET CHARACTER
303E 0E07               MVI     C,A     ; SAVE CHARACTER
3040 3E09               MVI     A,RDSON ; GET STROBE SET CONTROL WORD (LOW TRUE SIGNAL)
3042 D3F7               OUT     CWR     ; TURN OFF STROBE
3044 C9                 RET             ; RETURN TO CALLER
                ;*****
                ;       END OF MODE ZERO EXAMPLE
                ;*****
0000                    END
```

MODE 1 INTERRUPT DRIVEN PRINTER INTERFACE

The status driven interface described in the previous example required the software driver to poll the device status for completion. An alternate approach is to construct the device interface such that an interrupt is used to signal the completion of the operation. When an interrupt driven interface is utilized, the time that was dedicated to polling can be used to perform other functions and the effective processor through-put is increased. This example demonstrates how an 8255A configured in Mode 1 may be used to develop an interrupt driven interface for the Centronics 306 character printer.

CPU Module To 8255A Interface

The 8080 bus interface implemented for this example is the same as the Mode 0 example with the addition of interrupt support. Interrupt support is implemented through the use of a special feature of the 8228 System Controller. If only one interrupt vector is required (such as in small systems), the 8228 can automatically assert an RST 7 instruction onto the data bus at the proper time. This option is selected by connecting the $\overline{INTA}$ output of the 8228 to the +12-volt supply through a 1K ohm series resistor.

The Mode 1 interrupt support logic of the 8255A provides an interrupt request line for each port. The 8255A interrupt request line ($INTR_A$) must be connected to the INT line of the 8080. A 10K ohm pullup resistor is used to insure that the V_{IH} requirements of the 8080 are met.

8255A To Peripheral Interface

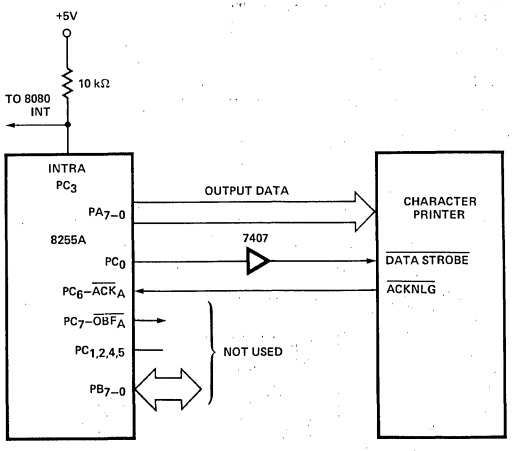
The interrupt driven configuration control signal interface to the printer is different than the status driven interface. Instead of a BUSY/DATA STROBE interface, a DATA STROBE/ACK interface is supported. The ACK signal notifies the 8255A that a character transferred to the printer by a DATA STROBE has been accepted. After an ACK is issued the printer is considered idle. The block diagram shown in Figure 18 displays the interface signals used.

The Mode 1 interrupt driven peripheral support signals used are:

PA_7–PA_0 – Output Data
Used to support the printer data port.

$\overline{OBF}$ – Output Buffer Full
This line goes low when data is placed in the output buffer. The OBF signal may be used as a data strobe signal when interfacing to peripherals which do not require a pulsed input. The Centronics 306 requires a pulsed DATA STROBE signal. This signal is supported by Port C bit 0.

$\overline{ACK}$ – ACKnowledge
This line is used to signal the 8255A that the device has accepted the data. This line is supported by the printer ACKNLG signal.



NOTES:
1. DATA BUS BUFFERED WITH 7407.
2. ALL 8255A OUTPUT LINES ARE PULLED UP TO +5 V AT THE PERIPHERAL.

Figure 18. Interface Block Diagram

Mode 1 Software Driver

The software driver implemented for this example utilizes the typical interrupt driven software structure outlined previously. The initialization routine issues the mode control word (shown in Figure 19) to the 8255A after reset of the device. The initialization routine also places a jump to the interrupt service routine in the interrupt location for RST 7. The command processor is started by the user routine through a subroutine call to PSTRT, with the address of the control block in the D and E registers (the control block format is shown in Table IV). The command processor insures that an I/O operation is not already in progress, starts the I/O, enables interrupts, and returns to the caller so that other processing may proceed.

After a character is placed in the output buffer, the DATA STROBE signal is generated through the use of the Port C bit set/reset feature. When the ACK is generated by the printer, the buffer full indication is cleared and the 8255A generates an interrupt. If interrupts are enabled, the interrupt request is serviced by the 8080 CPU through disabling processor interrupts and then executing the instruction at location 38 hexadecimal in program memory. The interrupt service routine saves the processor state and polls the 8255A to determine the source of the interrupt. Once the interrupting device is located, the control block is used to locate the next data character for transfer to the 8255A output buffer. After the entire buffer has been printed, the interrupt service routine passes control to the user-supplied completion routine. Before returning from the interrupt, the state of the processor is restored.

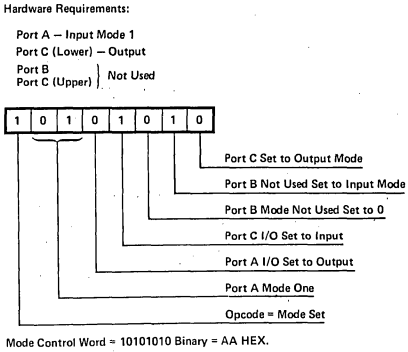


Figure 19. Mode Control Word

Table IV. Printer Software Control Block

NAME	POSITION	DEFINITION
Status	Byte 0	A 1-byte field which defines the completion status of an I/O. 00 = Good completion 01 = Error – command already in progress
Buffer Address	Byte 1, 2	Pointer to the start of the data to print.
Character Count	Byte 3	Count of the number of characters to print.
Character Transferred Count	Byte 4	The number of characters transferred.
Completion Address	Byte 5, 6	Address of a user supplied routine which will be called after the I/O has been performed.

NOTES:

1. An opcode field is not required because WRITE is the only operation performed.
2. The control block must reside above location FF Hex.

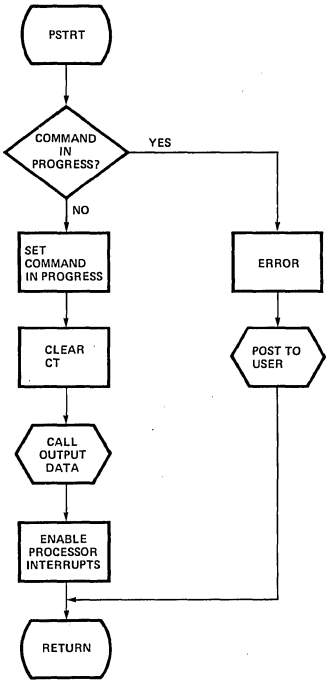
There are a number of error conditions which may occur, such as an interrupt from a device which does not have a control block in progress, or an interrupt when polling establishes that no device requires service. Neither of these errors should occur, but if they do, the driver should perform in a consistent fashion. The recovery routines implemented to handle error conditions are determined by the particular applications environment.

Summary/Conclusions

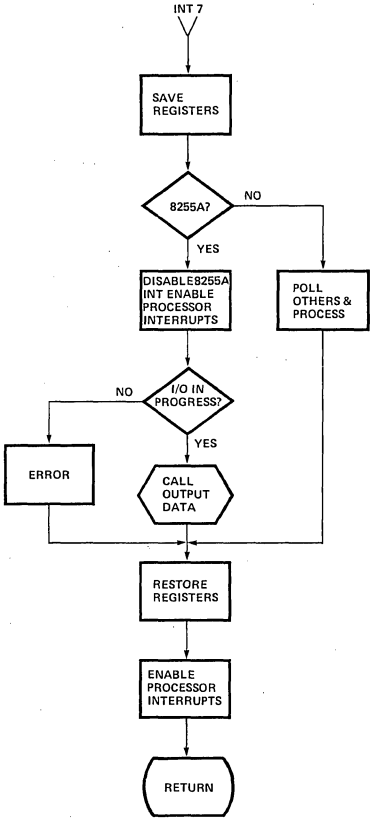
When utilized in a small system design, the 8255A interrupt support logic provides all of the capabilities required to implement an interrupt driven hardware interface without the use of external logic. In larger system designs, the designer may chose to use additional hardware to determine the source of interrupt requests without software polling. The software design required by an interrupt driven system is inherently more complex than the status driven interface. If an interrupt driven system is required the added complexity is a small price to pay for a significant increase in system through-put.

```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 1
MODE ONE EXAMPLE

                        TITLE 'MODE ONE EXAMPLE'
                ;*****
                ;
                ;       CHARACTER PRINTER - INTERRUPT DRIVEN
                ;       MODE ONE EXAMPLE
                ;
                ;*****
                ;
                ;*****
                ;       PROGRAM EQUATES
                ;*****
00F4            PORTA   EQU     0F4H    ; 8255 PORT A
00F5            PORTB   EQU     0F5H    ; 8255 PORT B
00F6            PORTC   EQU     0F6H    ; 8255 PORT C
00F7            CWR     EQU     0F7H    ; 8255 CONTROL WORD REGISTER
0038            RST7    EQU     038H    ; RESTART 7 ADDRESS
                ;
                ;*****
                ;
                ;       INITIALIZATION CONTROL WORD
                ;
                ;               USED TO CONFIGURE THE 8255 AS FOLLOWS:
                ;
                ;                       PORT A - OUTPUT MODE 1
                ;                       PORT B - INPUT MODE 0 (NOT USED)
                ;                       PORT C LOWER - OUTPUT
                ;
                ;*****
00AA            ICW     EQU     10101010B       ; INITIALIZATION CONTROL WORD
                ;*****
                ;       SET/RESET CONTROL WORDS
                ;*****
0001            STBON   EQU     00000001B       ; SET STROBE
0000            STBOF   EQU     00000000B       ; RESET STROBE
                ;*****
                ;       8255 ENABLE/DISABLE INTERRUPT CONTROL WORDS
                ;*****
000D            IEN     EQU     00001101B       ; ENABLE INTERRUPTS
000C            IDN     EQU     00001100B       ; DISABLE INTERRUPTS
                ;*****
                ;       DEVICE STATUS EQUATES
                ;*****
0080            LPBSY   EQU     080H    ; BUFFER FULL (LINE PRINTER BUSY)
0008            INTRA   EQU     08H     ; INTERRUPT REQUEST
```

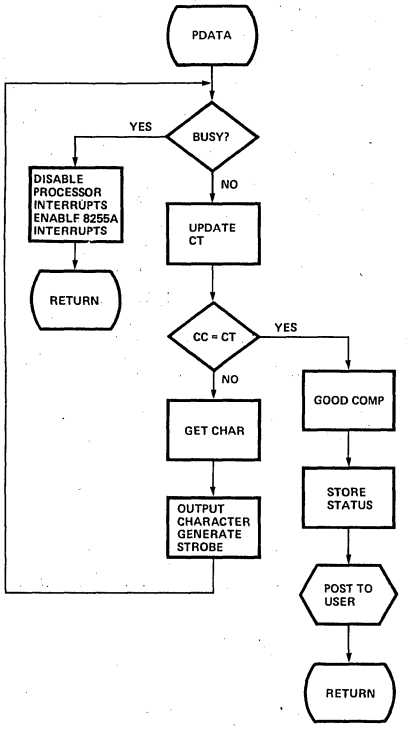
```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 2
MODE ONE EXAMPLE

                ;*****
                ;       CONTROL BLOCK EQUATES
                ;*****
0000            CBST    EQU     00H             ; STATUS BYTE
0001            CBUF    EQU     01H             ; BUFFER ADDRESS
0003            CBCC    EQU     03H             ; CHARACTER COUNT
0004            CBCT    EQU     04H             ; CHARACTER TRANSFERED COUNT
0005            CBCMP   EQU     05H             ; COMPLETION SERVICE ADDRESS
                ;*****
                ;       COMPLETION STATUS EQUATES
                ;*****
0000            STGD    EQU     00H             ; GOOD COMPLETION
0001            STE1    EQU     01H             ; ERROR - COMMAND ALREADY IN PROGRESS
                ;*****
                ;       PROGRAM ORIGIN
                ;*****
3000                    ORG     03000H
                ;*****
                ;
                ;       INITIALIZATION ROUTINE
                ;
                ;       A,H,L REGISTERS MODIFIED
                ;
                ;*****
                INIT:
3000 3EAA               MVI     A,ICW   ; GET MODE CONTROL WORD
3002 D3F7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
3004 3E01               MVI     A,STBON ; GET SET DATA STROBE CONTROL WORD
3006 D3F7               OUT     CWR     ; SET DATA STROBE (LOW TRUE SIGNAL)
                ;*****
                ;       SET UP RESTART 7 LOCATION WITH JUMP TO PINT
                ;*****
3008 3EC3               MVI     A,0C3H  ; GET "JMP"
300A 323800             STA     RST7    ; PLACE IN RST7 LOCATION
300D 213030             LXI     H,PINT  ; GET ADDRESS OF INTERRUPT SERVICE ROUTINE
3010 223900             SHLD    RST7+1  ; STORE ADDRESS
3013 C9                 RET             ; RETURN TO CALLER
```




```
                ;*****
                ;
                ;       COMMAND PROCESSOR
                ;
                ;               INPUTS: CONTROL BLOCK ADDRESS IN D AND E REGISTERS
                ;
                ;               OUTPUTS: START I/O OR ERROR STATUS IN CONTROL BLOCK
                ;
                ;       A,H,L REGISTERS MODIFIED
                ;
                ;*****
                PSTRT:
3014 3AA230             LDA     PIPRG+1 ; GET PRINT IN PROGRESS BLOCK ADDRESS
3017 A7                 ANA     A       ; SEE IF ZERO (PRINT IN PROGRESS)
                                        ; IF BLOCK ADDRESS NOT EQUAL TO ZERO THEN
                                        ; PRINT IN PROGRESS
3018 C22B30             JNZ     PSTE    ; IF YES - BRANCH TO ERROR
301B EB                 XCHG
301C 22A130             SHLD    PIPRG   ; SAVE CONTROL BLOCK ADDRESS
301F EB                 XCHG
3020 210400             LXI     H,CBCT  ; GET INDEX TO CT
3023 19                 DAD     D       ; COMPUTE ADDRESS OF CT
3024 3600               MVI     M,00H   ; CLEAR CT
3026 CD5830             CALL    PDATA   ; START I/O
3029 FB                 EI              ; ENABLE PROCESSOR INTERRUPTS
302A C9                 RET             ; RETURN TO CALLER
                ;*****
                ;       ERROR - TRANSACTION ALREADY IN PROGRESS
                ;*****
                PSTE:
302B 3E01               MVI     A,STE1  ; GET ERROR STATUS CODE
302D C39430             JMP     POST    ; PASS CONTROL TO COMPLETION ROUTINE
```




```
                ;*****
                ;       PRINTER INTERRUPT SERVICE ROUTINE
                ;       ALL REGISTERS SAVED AND RESTORED
                ;*****
                PINT:
3030 F5                 PUSH    PSW     ; SAVE PSW
3031 C5                 PUSH    B       ; SAVE REGISTER PAIR B AND C
3032 D5                 PUSH    D       ; SAVE REGISTER PAIR D AND E
                        PUSH    H       ; SAVE REGISTER PAIR H AND L
                ;*****
                ;       POLL INTERRUPT SOURCE - SEE IF 8255
                ;*****
3034 DBF6               IN      PORTC   ; GET STATUS OF DEVICE
3036 E608               ANI     INTRA   ; SEE IF INT
3038 CA5230             JZ      PPOLL   ; NO - BRANCH TO POLL OTHER DEVICES IF ANY
303B 3E0C               MVI     A,IDN   ; GET 8255 INT DISABLE CONTROL WORD
303D D3F7               OUT     CWR     ; DISABLE DEVICE INTERRUPTS
303F FB                 EI              ; ENABLE PROCESSOR INTERRUPTS
3040 2AA130             LHLD    PIPRG   ; GET CONTROL BLOCK ADDRESS
3043 AF                 XRA     A       ; CLEAR A REG
3044 BC                 CMP     H       ; SEE IF PRINT IN PROGRESS
3045 CA5530             JZ      PIER1   ; NO - BRANCH TO ERROR ROUTINE
3048 EB                 XCHG
3049 CD5830             CALL    PDATA   ; PRINT DATA
                ;*****
                ;       RESTORE REGISTERS AND RETURN FROM INTERRUPT
                ;*****
                PRTN:
304C E1                 POP     H       ; RESTORE REGISTER PAIR H AND L
304D D1                 POP     D       ; RESTORE REGISTER PAIR D AND E
304E C1                 POP     B       ; RESTORE REGISTER PAIR B AND C
304F F1                 POP     PSW     ; RESTORE PSW
3050 FB                 EI              ; ENABLE PROCESSOR INTERRUPTS
3051 C9                 RET             ; RETURN TO INTERRUPTED PROCESS
                ;*****
                ;       POLL OTHER DEVICES IF ANY
                ;               IF NO OTHER DEVICES TO POLL - USER SUPPLIED ERROR
                ;               RECOVERY ROUTINE.
                ;*****
                PPOLL:
3052 C34C30             JMP     PRTN    ; RETURN
                ;*****
                ;       ERROR - INTERRUPT FROM IDLE DEVICE
                ;               USER SUPPLIED ERROR RECOVERY ROUTINE
                ;*****
                PIER1:
3055 C34C30             JMP     PRTN    ; RETURN
```




```
                ;*****
                ;
                ;       PRINTER OUTPUT DATA ROUTINE
                ;
                ;       CONTROL BLOCK ADDRESS IN D AND E REG
                ;
                ;*****
                PDATA:
3058 DBF6               IN      PORTC   ; GET STATUS OF DEVICE
305A E680               ANI     LPBSY   ; SEE IF BUSY (BUFFER FULL)
305C CA8430             JZ      PD10    ; IF BUSY - BRANCH
305F 210400             LXI     H,CBCT  ; GET INDEX TO CT
3062 19                 DAD     D       ; COMPUTE ADDRESS OF CT
3063 7E                 MOV     A,M     ; GET CT
3064 34                 INR     M       ; INC CT
3065 2B                 DCX     H       ; DEC TO CC
3066 BE                 CMP     M       ; SEE IF EQUAL
3067 CA8A30             JZ      PCOMP   ; IF EQUAL - DONE GO TELL USER
306A 210100             LXI     H,CBUF  ; GET INDEX TO BUFFER ADDRESS
306D 19                 DAD     D       ; COMPUTE ADDRESS OF BUFFER ADDRESS
306E D5                 PUSH    D       ; SAVE D AND E REGISTERS
306F 5E                 MOV     E,M     ; GET LSB OF BUFFER ADDRESS
3070 23                 INX     H       ; INC TO NEXT BYTE
3071 56                 MOV     D,M     ; GET BUFFER MSB
3072 2600               MVI     H,00H   ; CLEAR H REG
3074 6F                 MOV     L,A     ; GET CT
3075 19                 DAD     D       ; COMPUTE CHARACTER ADDRESS
3076 7E                 MOV     A,M     ; GET CHARACTER
3077 D3F4               OUT     PORTA   ; OUTPUT CHARACTER TO PRINTER
3079 3E00               MVI     A,STBOF ; RESET DATA STROBE (LOW TRUE SIGNAL)
307B D3F7               OUT     CWR
307D 3C                 INR     A       ; GENERATE SET CONTROL WORD
307E D3F7               OUT     CWR     ; SET DATA STROBE
3080 D1                 POP     D       ; RESTORE CONTROL BLOCK ADDRESS
3081 C35830             JMP     PDATA   ; LOOP UNTIL BUSY

                ;*****
                ;       PRINTER BUSY - RETURN
                ;*****
                PD10:
3084 F3                 DI              ; DISABLE INTERRUPTS
3085 3E0D               MVI     A,IEN   ; ENABLE DEVICE INTERRUPTS
3087 D3F7               OUT     CWR     ; SET INTERRUPT ENABLE
3089 C9                 RET             ; RETURN TO CALLER
                ;*****
                ;       POST GOOD COMPLETION TO USER
                ;*****
                PCOMP:
308A 3E00               MVI     A,STGD  ; GET GOOD STATUS CODE
                        CALL    POST    ; POST TO USER
308F AF                 XRA     A       ; CLEAR A REG
3090 32A230             STA     PIPRG+1 ; CLEAR COMMAND IN PROGRESS ADDRESS
3093 C9                 RET             ; RETURN TO CALLER


                ;*****
                ;
                ;       POST TO USER COMPLETION ROUTINE
                ;
                ;               INPUTS : STATUS CODE IN A REG
                ;                        CONTROL BLOCK ADDRESS IN D AND E REG
                ;               OUTPUTS: PASSES CONTROL TO USER COMPLETION ADDRESS
                ;                        SPECIFIED IN CONTROL BLOCK
                ;                        WITH CONTROL BLOCK ADDRESS IN D AND E
                ;
                ;               A,H,L,B,C REG MODIFIED
                ;
                ;*****
                POST:
3094 EB                 XCHG
3095 77                 MOV     M,A     ; UPDATE STATUS
3096 EB                 XCHG
3097 210500             LXI     H,CBCMP ; GET INDEX TO COMPLETION ADDRESS
309A 19                 DAD     D       ; COMPUTE ADDRESS
309B 4E                 MOV     C,M     ; GET LSB OF COMPLETION ADDRESS
309C 23                 INX     H       ; INC TO NEXT BYTE
309D 46                 MOV     B,M     ; GET MSB  OF COMPLETION ADDRESS
309E C5                 PUSH    B       ; PUSH ADDRESS INTO STACK
309F C9                 RET             ; PASS CONTROL TO USER ROUTINE
                ;*****
                ;       DATA AND TABLES
                ;*****
30A1 0000       PIPRG:  DW      0       ; IN PROGRESS CONTROL BLOCK ADDRESS
                                        ; IF DATA = 0 NO CONTROL BLOCK IN PROGRESS
                                        ; IF DATA NOT EQUAL TO ZERO CONTROL BLOCK IN PROGRESS
                ;*****
                ;       END OF MODE ONE EXAMPLE
                ;*****
0000                    END
```

MODE 2 – 8080 TO 8080 INTERFACE

Due to the drastic reduction of hardware costs, system designs which utilize multiple CPU Modules are becoming more common. An 8080 may be configured as a master CPU and used to control multiple 8080 slave modules which act as intelligent I/O controllers. When multiple CPUs are utilized, a method of processor intercommunication must be supported. Figure 20 is a block diagram of one method of implementing a master/slave interface through the use of the 8255A Mode 2 bidirectional bus.

Hardware Discussion

Two complete 8080 systems are required for this example. Intel's SBC 80/10 OEM board is used as the master CPU module and Intel's SDK 80 board is used as the slave CPU. The SBC 80/10 supports an 8255A which is configured in Mode 2. The 8255A is selected through the use of a decoded select scheme. Through the use of the 8228 RST 7 interrupt feature, a simple interrupt structure is supported. The SDK 80 is configured without interrupts for this example. The external logic required for this example is associated with the slave CPU. Simple logic is implemented which allows the slave CPU to generate the $\overline{ACK}$ and $\overline{STB}$ signals required to READ from and WRITE to the 8255A bidirectional bus with a single I/O instruction.

The system shown in Figure 20 utilizes SSI logic to read the 8255A IBF and OBF signals. If two spare 8255A input lines are available they could be used to input the IBF and OBF signals and eliminate the SSI logic.

Software Discussion

Two sets of software are required to support the processor to processor interface. The master resident software which follows conforms to the simple interrupt driven software structure outlined previously. The initialization routine issues the Mode 2 control word to the 8255A after device reset. The command processor accepts READ/WRITE control blocks which provide a simple user interface for transferring data to/from the slave CPU. The master software is capable of processing both a read and a write control block simultaneously. The slave resident software shown at the end of this example utilizes the status driven approach.

Summary/Conclusions

It is important to note that this design may be expanded to include more slave CPUs by simply adding another 8255A to the master module for each slave. The software drivers discussed address only the passing of data between the two processors. Specific applications generally dictate a software protocol be implemented for information transfer.

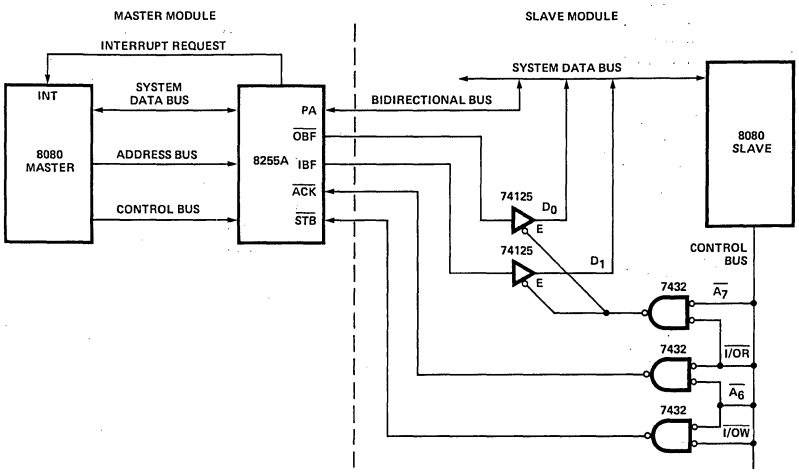
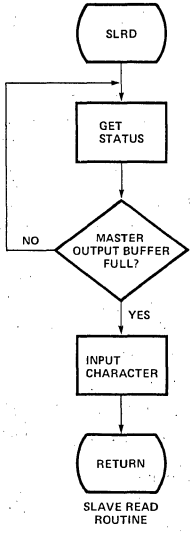
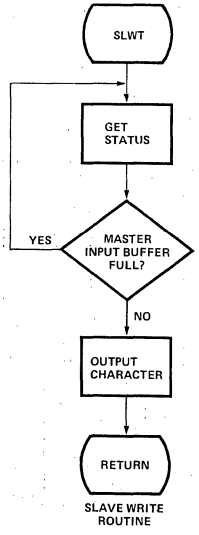


Figure 20. Interface Block Diagram




```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 1
MODE TWO EXAMPLE - SLAVE SOFTWARE

                        TITLE 'MODE TWO EXAMPLE - SLAVE SOFTWARE'
                  ;*****
                  ;
                  ;     8080 MASTER TO 8080 SLAVE INTERFACE
                  ;            - SLAVE SOFTWARE -
                  ;            MODE TWO EXAMPLE
                  ;
                  ;*****
                  ;
                  ;*****
                  ;     PROGRAM EQUATES
                  ;*****
00BF              PDATA EQU     0BFH    ; INTERPROCESSOR DATA PORT
007F              PSTS  EQU     07FH    ; STATUS
                  ;*****
                  ;     BUFFER STATUS MASKS
                  ;*****
0001              OBF   EQU     01H     ; OUTPUT BUFFER FULL
0002              IBF   EQU     02H     ; INPUT BUFFER FULL
                  ;*****
                  ;     PROGRAM ORIGIN
                  ;*****
3000                    ORG     03000H
                  ;*****
                  ;     SLAVE READ ROUTINE
                  ;
                  ;            INPUTS:  NONE
                  ;            OUTPUTS: CHARACTER READ IN C-REGISTER
                  ;
                  ;     A,C REG MODIFIED
                  ;
                  ;*****
                  SLRD:
3000 DB7F               IN      PSTS    ; GET STATUS
3002 E601               ANI     OBF     ; SEE IF BUFFER FULL
3004 C20030             JNZ     SLRD    ; NO - LOOP UNTIL FULL
3007 DBBF               IN      PDATA   ; GET CHARACTER
3009 4F                 MOV     C,A     ; PLACE IN C-REG
300A C9                 RET             ; RETURN TO CALLER
```

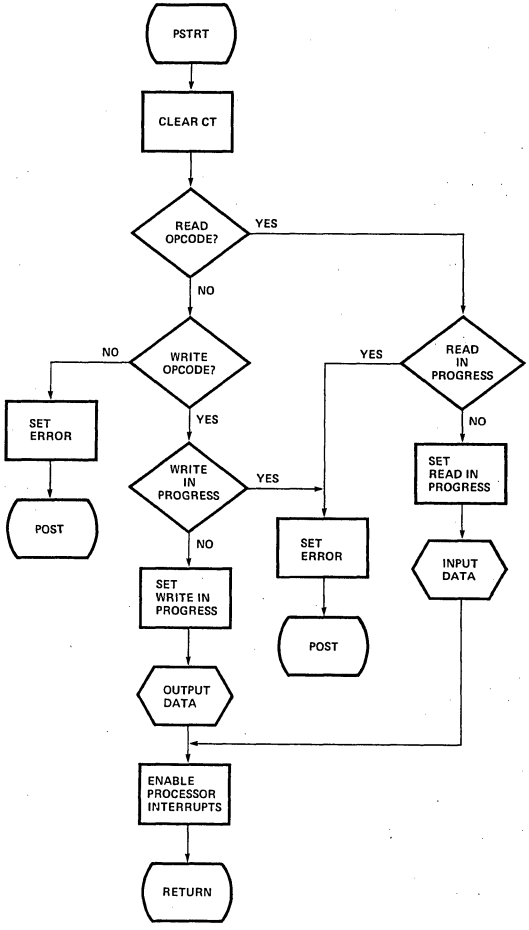



```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 2
MODE TWO EXAMPLE - SLAVE SOFTWARE

                  ;*****
                  ;
                  ;     SLAVE WRITE ROUTINE
                  ;
                  ;            INPUTS:  CHARACTER TO WRITE IN C-REGISTER
                  ;            OUTPUTS: NONE
                  ;
                  ;     A REG MODIFIED
                  ;
                  ;*****
                  SLWT:
300B DB7F               IN      PSTS    ; GET STATUS
300D E602               ANI     IBF     ; SEE IF BUFFER FULL
300F C20B30             JNZ     SLWT    ; YES - LOOP UNTIL EMPTY
3012 79                 MOV     A,C     ; GET DATA CHARACTER
3013 D3BF               OUT     PDATA   ; OUTPUT DATA
3015 C9                 RET             ; RETURN TO CALLER
                  ;*****
                  ;     END OF SLAVE SOFTWARE DRIVER
                  ;*****
0000                    END
```

```
                        TITLE 'MODE TWO EXAMPLE - MASTER SOFTWARE'
                ;*****
                ;
                ;       8080 MASTER TO 8080 SLAVE INTERFACE
                ;               - MASTER SOFTWARE -
                ;               MODE TWO EXAMPLE
                ;
                ;*****
                ;
                ;*****
                ;
                ;       PROGRAM EQUATES
                ;*****
00E4            PORTA   EQU     0E4H    ; 8255 PORT A
                PORTB   EQU     0E5H    ; 8255 PORT B
00E6            PORTC   EQU     0E6H    ; 8255 PORT C
00E7            CWR     EQU     0E7H    ; 8255 CONTROL WORD REGISTER
0038            RST7    EQU     038H    ; RESTART 7 ADDRESS
                ;
                ;*****
                ;
                ;       INITIALIZATION CONTROL WORD
                ;
                ;               USED TO CONFIGURE THE 8255 AS FOLLOWS:
                ;
                ;                       PORT A - MODE 2 BIDIRECTIONAL BUS
                ;                       PORT B - INPUT MODE 0 (NOT USED)
                ;                       REMAINING PORT C LINES - INPUT MODE (NOT USED)
                ;
                ;*****
00CB            ICW     EQU     11001011B       ; INITIALIZATION CONTROL WORD
                ;*****
                ;       8255 ENABLE/DISABLE INTERRUPT CONTROL WORDS
                ;*****
000D            IENI    EQU     00001101B       ; ENABLE INPUT INTERRUPTS
0009            IENO    EQU     00001001B       ; ENABLE OUTPUT INTERRUPTS
000C            IDNI    EQU     00001100B       ; DISABLE INPUT INTERRUPTS
0008            IDNO    EQU     00001000B       ; DISABLE OUTPUT INTERRUPTS
                ;*****
                ;       STATUS EQUATES
                ;*****
0008            INTRA   EQU     08H     ; INTERRUPT REQUEST
0080            OBFA    EQU     80H     ; OUTPUT BUFFER FULL
0020            IBFA    EQU     20H     ; INPUT BUFFER FULL
```



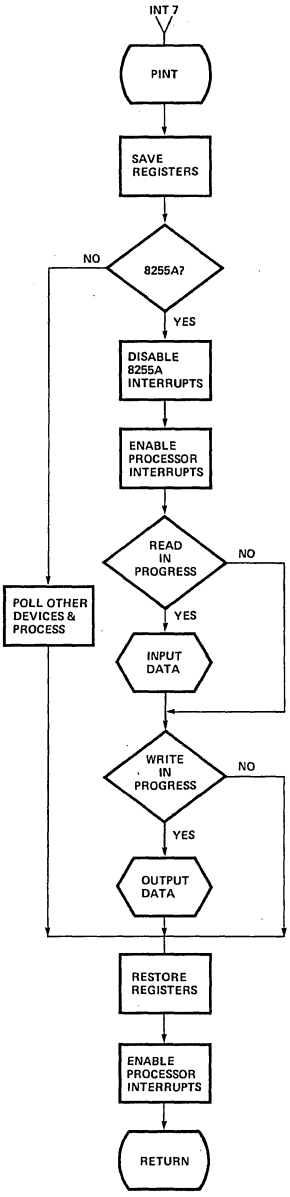
```
                ;*****
                ;       CONTROL BLOCK EQUATES
                ;*****
0000            CBST    EQU     00H             ; STATUS BYTE
                CBOP    EQU     01H             ; OPCODE = 0 READ
                                                ;        = 1 WRITE
0002            CBUF    EQU     02H             ; BUFFER ADDRESS
0004            CBCC    EQU     04H             ; CHARACTER COUNT
0005            CBCT    EQU     05H             ; CHARACTER TRANSFERED COUNT
0006            CBCMP   EQU     06H             ; COMPLETION SERVICE ADDRESS
                ;*****
                ;       OPCODE EQUATES
                ;*****
0000            OPRD    EQU     00H     ; READ OPCODE
0001            OPWT    EQU     01H     ; WRITE OPCODE
                ;*****
                ;       COMPLETION STATUS EQUATES
                ;*****
0000            STGD    EQU     00H             ; GOOD COMPLETION
0001            STE1    EQU     01H             ; ERROR - COMMAND ALREADY IN PROGRESS
0002            STE2    EQU     02H             ; ERROR - INVALID OPCODE
                ;*****
                ;       SET UP INTERRUPT VECTOR
                ;*****
0038                    ORG     RST7
0038 C34630             JMP     PINT            ; JUMP TO INTERRUPT SERVICE ROUTINE
                ;*****
                ;       PROGRAM ORIGIN
                ;*****
3000                    ORG     03000H
                ;*****
                ;
                ;       INITIALIZATION ROUTINE
                ;
                ;       A REGISTER MODIFIED
                ;
                ;*****
                INIT:
3000 3ECB               MVI     A,ICW   ; GET MODE CONTROL WORD
3002 D3E7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
3004 C9                 RET             ; RETURN TO CALLER
```




```
                ;*****
                ;
                ;       COMMAND PROCESSOR
                ;
                ;               INPUTS: CONTROL BLOCK ADDRESS IN D AND E REGISTERS
                ;
                ;               OUTPUTS: START I/O OR ERROR STATUS IN CONTROL BLOCK
                ;       A,H,L   MODIFIED
                ;
                ;*****
                PSTRT:
3005 210500             LXI     H,CBCT  ; GET INDEX TO CT
3008 19                 DAD     D       ; COMPUTE ADDRESS OF CT
3009 3600               MVI     M,OPRD  ; CLEAR CT
300B 210100             LXI     H,CBOP  ; GET INDEX TO OPCODE
300E 19                 DAD     D       ; COMPUTE ADDRESS
                        MOV     A,M     ; GET OPCODE
3010 FE00               CPI     00H     ; SEE IF READ
3012 CA2430             JZ      PSRD    ; YES - GO PROCESS READ
3015 FE01               CPI     OPWT    ; SEE IF WRITE
3017 CA3530             JZ      PSWT    ; YES - GO PROCESS WRITE
                ;*****
                ;       ERROR - INVALID OPCODE
                ;*****
301A 3E02               MVI     A,STE2  ; GET ERROR STATUS CODE
                        JMP     POST    ; CALL COMPLETION ROUTINE
                ;*****
                ;       ERROR - TRANSACTION ALREADY IN PROGRESS
                ;*****
                PSTE:
301F 3E01               MVI     A,STE1  ; GET ERROR STATUS CODE
3021 C3DC30             JMP     POST    ; CALL COMPLETION ROUTINE
                ;*****
                ;       PROCESS READ COMMAND
                ;*****
                PSRD:
3024 3AEA30             LDA     PRGRD+1 ; GET READ IN PROGRESS ADDRESS
3027 A7                 ANA     A       ; SEE IF READ IN PROGRESS (TEST FOR ZERO)
3028 C21F30             JNZ     PSTE    ; IF YES - BRANCH
302B EB                 XCHG
302C 22E930             SHLD    PRGRD   ; SAVE CONTROL BLOCK ADDRESS
302F EB                 XCHG
3030 CD7C30             CALL    PIN     ; START I/O
3033 FB                 EI              ; ENABLE INTERRUPTS
3034 C9                 RET             ; RETURN TO CALLER
```



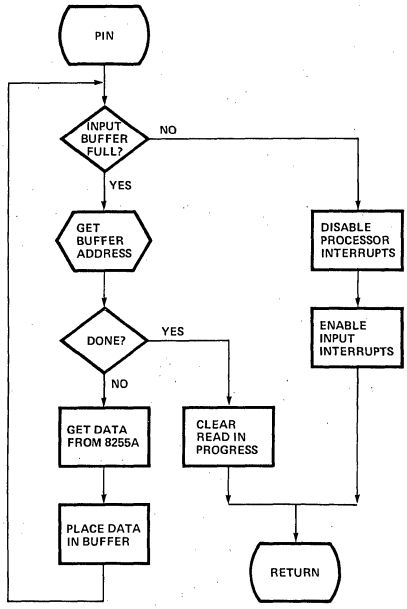
```
                ;*****
                ;       PROCESS WRITE COMMAND
                ;*****
                PSWT:
3035 3AEC30             LDA     PRGWT+1 ; GET WRITE IN PROGRESS ADDRESS
3038 A7                 ANA     A       ; SEE IF WRITE IN PROGRESS (TEST FOR ZERO)
3039 C21F30             JNZ     PSTE    ; IF YES - BRANCH
303C EB                 XCHG
303D 22EB30             SHLD    PRGWT   ; SAVE CONTROL BLOCK ADDRESS
3040 EB                 XCHG
3041 CD9C30             CALL    POUT    ; START I/O
3044 FB                 EI              ; ENABLE INTERRUPTS
3045 C9                 RET             ; RETURN TO CALLER
```



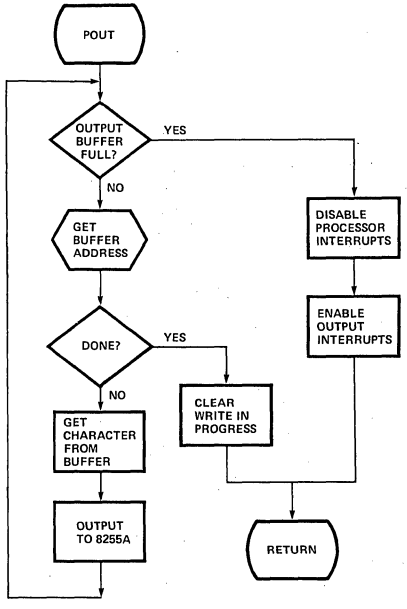

```
                ;*****
                ;
                ;       INTERRUPT SERVICE ROUTINE
                ;      ALL REGISTERS SAVED AND RESTORED
                ;
                ;*****
                PINT:
 3046 F5                PUSH    PSW     ; SAVE PSW
 3047 C5                PUSH    B       ; SAVE REGISTER PAIR B AND C
 3048 D5                PUSH    D       ; SAVE REGISTER PAIR D AND E
                PUSH    H       ; SAVE REGISTER PAIR H AND L
                ;*****
                ;       POLL INTERRUPT SOURCE - SEE IF 8255
                ;*****
 304A DBE6              IN      PORTC   ; GET STATUS OF DEVICE
 304C E608              ANI     INTRA   ; SEE IF INT
 304E CA7630            JZ      PPOLL   ; NO - BRANCH TO POLL OTHER DEVICES IF ANY
 3051 3E0C              MVI     A,IDNI  ; GET INPUT INT DISABLE CONTROL WORD
 3053 D3E7              OUT     CWR     ; DISABLE DEVICE INTERRUPTS
 3055 3E08              MVI     A,IDNO  ; GET OUTPUT INT DISABLE CONTROL WORD
 3057 D3E7              OUT     CWR     ; DISABE DEVICE INTERRUPTS
 3059 FB                EI              ; ENABLE PROCESSOR INTERRUPTS
 305A 2AE930            LHLD    PRGRD   ; GET READ CONTROL BLOCK
 305D AF                XRA     A       ; CLEAR A REG
 305E BC                CMP     H       ; SEE IF READ IN PROGRESS
 305F CA6530            JZ      PINT1   ; NO - BRANCH
 3062 CD7C30            CALL    PIN     ; DO INPUT
                PINT1:
 3065 2AEB30            LHLD    PRGWT   ; GET WRITE CONTROL BLOCK
 3068 AF                XRA     A       ; CLEAR A REG
 3069 BC                CMP     H       ; SEE IF WRITE IN PROGRESS
 306A CA7030            JZ      PRTN    ; NO - BRANCH
 306D CD9C30            CALL    POUT    ; DO OUTPUT
                ;*****
                ;       RESTORE REGISTERS AND RETURN FROM INTERRUPT
                ;*****
                PRTN:
 3070 E1                POP     H       ; RESTORE REGISTER PAIR H AND L
 3071 D1                POP     D       ; RESTORE REGISTER PAIR D AND E
 3072 C1                POP     B       ; RESTORE REGISTER PAIR B AND C
 3073 F1                POP     PSW     ; RESTORE PSW
 3074 FB                EI              ; ENABLE PROCESSOR INTERRUPTS
 3075 C9                RET             ; RETURN TO INTERRUPTED PROCESS
```



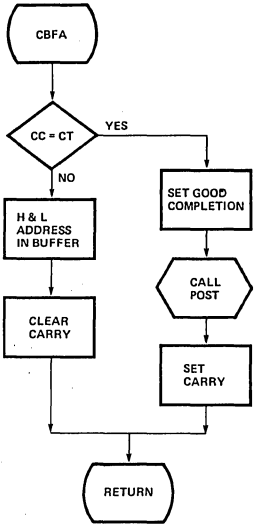
```
                ;*****
                ;
                ;       POLL OTHER DEVICES IF ANY
                ;               IF NO OTHER DEVICES TO POLL - USER SUPPLIED ERROR
                ;               RECOVERY ROUTINE.
                ;
                ;*****
                PPOLL:
 3076 C37030            JMP     PRTN    ; RETURN
                ;*****
                ;       ERROR - INTERRUPT FROM IDLE DEVICE
                ;               USER SUPPLIED ERROR RECOVERY ROUTINE
                ;*****
                PIER1:
 3079 C37030            JMP     PRTN    ; RETURN
```




```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 7
INPUT DATA ROUTINE

                ;*****
                ;
                ;       INPUT DATA ROUTINE
                ;*****
                PIN:
307C DBE6               IN      PORTC   ; GET STATUS OF DEVICE
307E E620               ANI     IBFA    ; SEE IF INPUT BUFFER FULL
3080 CA9630             JZ      PRTI    ; NO - BRANCH
3083 CDBC30             CALL    CBFA    ; GET ADDRESS IN BUFFER
3086 DA8F30             JC      PIDON   ; IF DONE - BRANCH
3089 DBE4               IN      PORTA   ; GET DATA
308B 77                 MOV     M,A     ; PLACE IN BUFFER
308C C37C30             JMP     PIN     ; LOOP
                ;*****
                ;       END OF INPUT TRANSACTION
                ;*****
                PIDON:
308F AF                 XRA     A       ; CLEAR A
3090 32EA30             STA     PRGRD+1 ; CLEAR READ IN PROGRESS
3093 C39630             JMP     PRTI    ; RETURN
                ;*****
                ;       RETURN FROM INPUT
                ;*****
                PRTI:
3096 F3                 DI              ; DISABLE PROCESSOR INTERRUPTS
3097 3E0D               MVI     A,IENI  ; GET ENABLE INPUT INTERRUPTS CONTROL WORD
3099 D3E7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
309B C9                 RET             ; RETURN TO CALLER
```



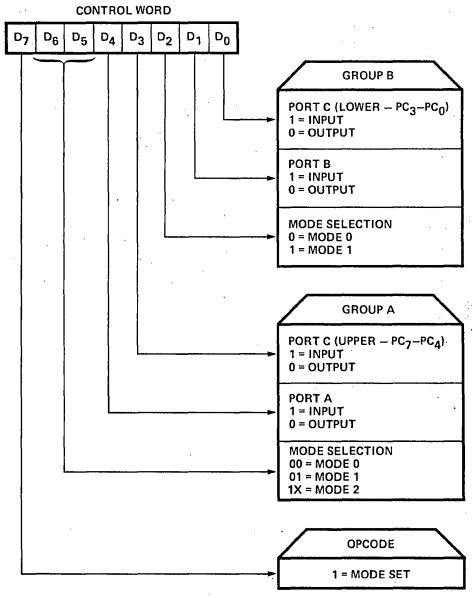

```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 8
OUTPUT DATA ROUTINE

                ;*****
                ;       OUTPUT DATA ROUTINE
                ;*****
                POUT:
309C DBE6               IN      PORTC   ; GET PORTC STATUS
309E E620               ANI     IBFA    ; SEE IF OUTPUT BUFFER FULL
                        JNZ     PRTO    ; YES - BRANCH
30A3 CDBC30             CALL    CBFA    ; SET UP ADDRESS OF DATA
30A6 DAAF30             JC      PODON   ; IF DONE - BRANCH
30A9 7E                 MOV     A,M     ; GET DATA FROM BUFFER
30AA D3E4               OUT     PORTA   ; OUTPUT DATA
30AC C39C30             JMP     POUT    ; LOOP
                ;*****
                ;       END OF OUTPUT TRANSACTION
                ;*****
                PODON:
30AF AF                 XRA     A       ; CLEAR A REG
30B0 32EC30             STA     PRGWT+1 ; CLEAR WRITE IN PROGRESS
30B3 C3B630             JMP     PRTO    ; RETURN
                ;*****
                ;       RETURN FROM OUTPUT
                ;*****
                PRTO:
30B6 F3                 DI              ; DISABLE PROCESSOR INTERRUPTS
30B7 3E09               MVI     A,IENO  ; GET ENABLE OUTPUT INTERRUPTS CONTROL WORD
30B9 D3E7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
30BB C9                 RET             ; RETURN TO CALLER
```



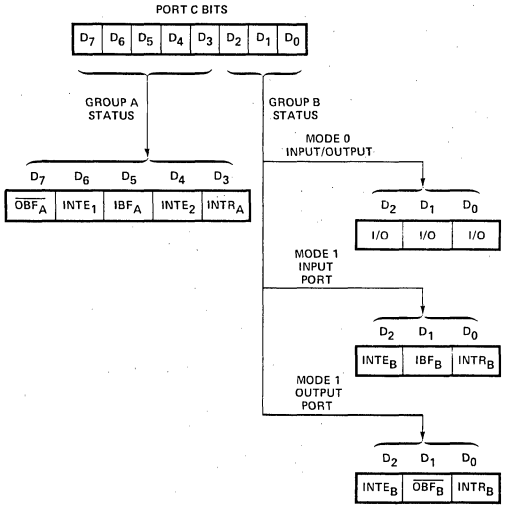
Setup Buffer Address Subroutine

```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 9
COMPUTE BUFFER ADDRESS ROUTINE

                ;*****
                ;
                ;       COMPUTE BUFFER ADDRESS ROUTINE
                ;*****
                CBFA:
30BC 210500             LXI     H,CBCT  ; GET INDEX TO CT
30BF 19                 DAD     D       ; COMPUTE ADDRESS OF CT
30C0 7E                 MOV     A,M     ; GET CT
30C1 34                 INR     M       ; INC CT
30C2 2B                 DCX     H       ; DEC TO CC
30C3 BE                 CMP     M       ; SEE IF EQUAL
30C4 CAD530             JZ      PCOMP   ; IF EQUAL - DONE GO TELL USER
30C7 210200             LXI     H,CBUF  ; GET INDEX TO BUFFER ADDRESS
30CA 19                 DAD     D       ; COMPUTE ADDRESS OF BUFFER ADDRESS
30CB D5                 PUSH    D       ; SAVE D AND E REGISTERS
30CC 5E                 MOV     E,M     ; GET LSB OF BUFFER ADDRESS
30CD 23                 INX     H       ; INC TO NEXT BYTE
30CE 56                 MOV     D,M     ; GET BUFFER MSB
30CF AC                 XRA     H       ; CLEAR H REG
30D0 6F                 MOV     L,A     ; GET CT
30D1 19                 DAD     D       ; COMPUTE CHARACTER ADDRESS
30D2 D1                 POP     D       ; RESTORE CONTROL BLOCK ADDRESS
30D3 AF                 XRA     A       ; CLEAR CARRY
30D4 C9                 RET             ; RETURN TO CALLER
```

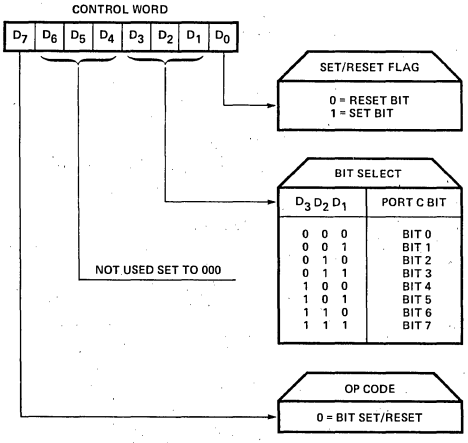
```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 10
POST TO USER COMPLETION ROUTINE

                ;*****
                ;
                ;       POST GOOD COMPLETION TO USER
                ;*****
                PCOMP:
30D5 3E00               MVI     A,STGD  ; GET GOOD STATUS CODE
30D7 CDDC30             CALL    POST    ; CALL USER ROUTINE
30DA 37                 STC             ; SET CARRY
30DB C9                 RET             ; RETURN TO CALLER
                ;*****
                ;       POST TO USER COMPLETION ROUTINE
                ;
                ;               INPUTS : STATUS CODE IN A REG
                ;                        CONTROL BLOCK ADDRESS IN D AND E REG
                ;               OUTPUTS: PASSES CONTROL TO USER COMPLETION ADDRESS
                ;                        SPECIFIED IN CONTROL BLOCK
                ;*****
                POST:
30DC EB                 XCHG
30DD 77                 MOV     M,A     ; UPDATE STATUS
30DE EB                 XCHG
30DF 210600             LXI     H,CBCMP ; GET INDEX TO COMPLETION ADDRESS
30E2 19                 DAD     D       ; COMPUTE ADDRESS
30E3 4E                 MOV     C,M     ; GET LSB OF COMPLETION ADDRESS
30E4 23                 INX     H       ; INC TO NEXT BYTE
30E5 46                 MOV     B,M     ; GET MSB BYTE OF COMPLETION ADDRESS
30E6 C5                 PUSH    B       ; PUSH ADDRESS INTO STACK
30E7 C9                 RET             ; PASS CONTROL TO USER ROUTINE
30E8 C9                 RET             ; RETURN TO CALLER
                ;*****
                ;
                ;       DATA AND TABLES
                ;
                ;               IF DATA NON ZERO CONTROL BLOCK IN PROGRESS
                ;
                ;*****

30E9 0000       PRGRD:  DW      0       ; IN PROGRESS READ CONTROL BLOCK
30EB 0000       PRGWT:  DW      0       ; IN PROGRESS WRITE CONTROL BLOCK
                ;*****
                ;       END OF MASTER SOFTWARE DRIVER
                ;*****
0000                    END
```



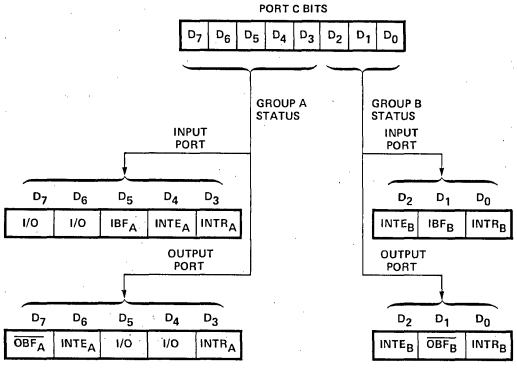
MODE CONTROL WORD



MODE 1 STATUS WORD

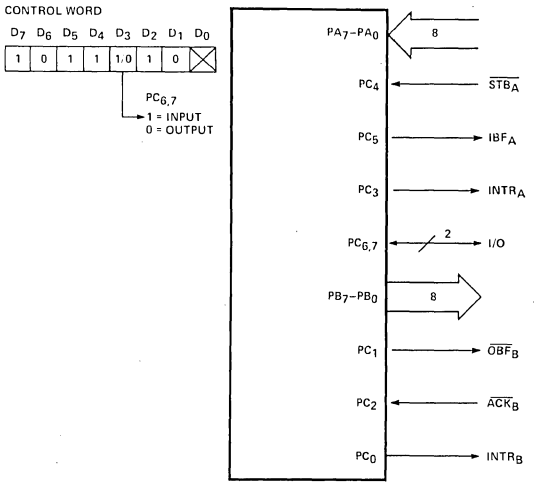


BIT SET/RESET CONTROL WORD

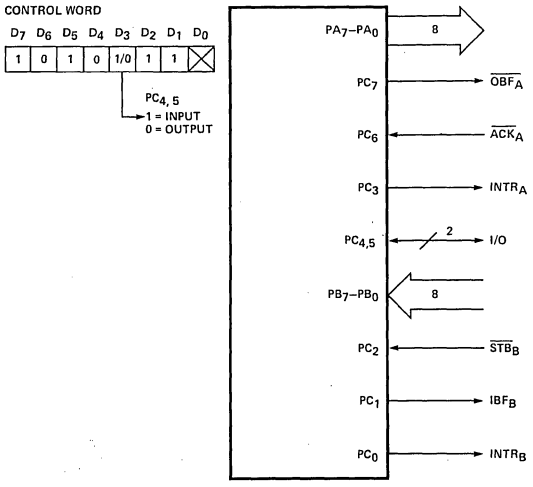


MODE 2 STATUS WORD

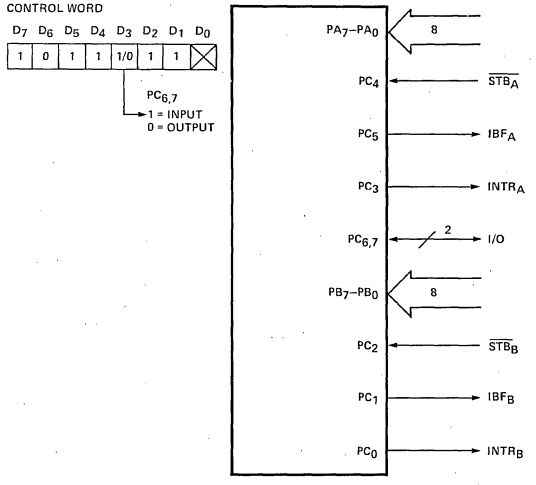
MODE 1 CONFIGURATIONS



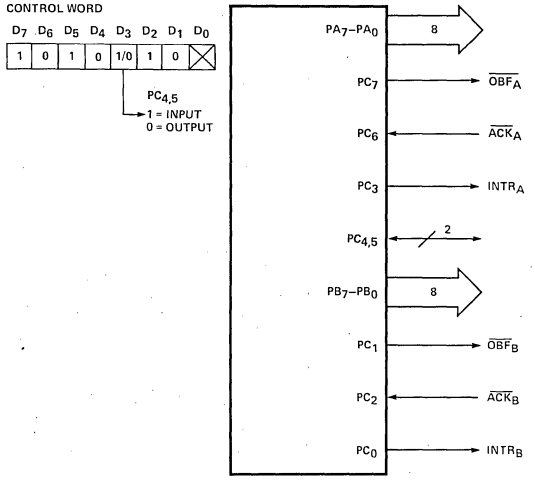
PORT A – STROBED INPUT
PORT B – STROBED OUTPUT



PORT A – STROBED OUTPUT
PORT B – STROBED INPUT

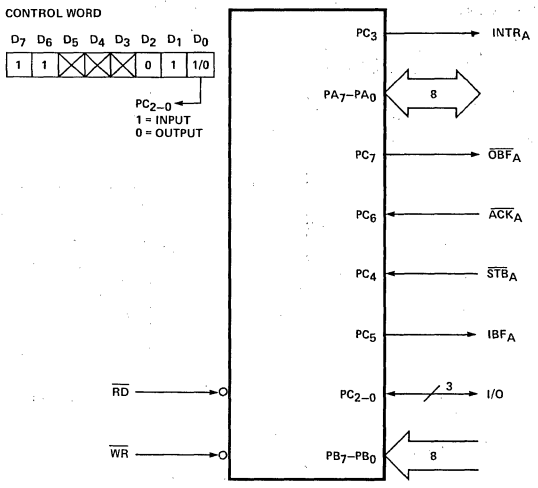


PORT A – STROBED INPUT
PORT B – STROBED INPUT

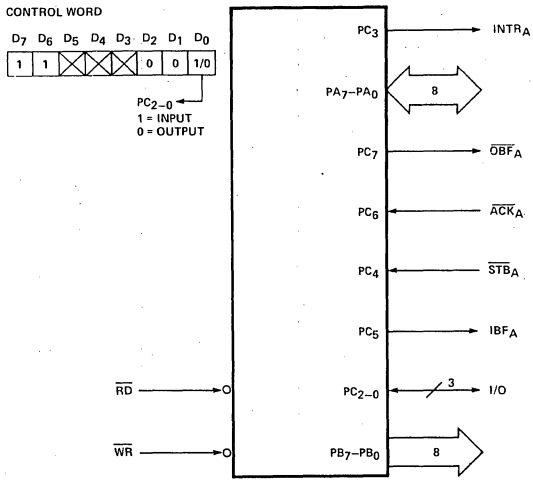


PORT A – STROBED OUTPUT
PORT B – STROBED OUTPUT

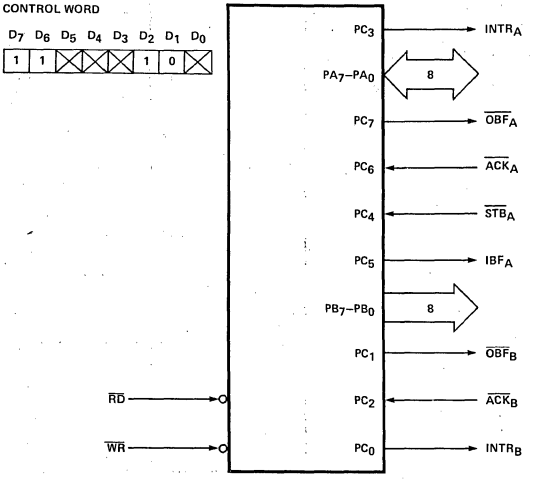
MODE 2 CONFIGURATIONS



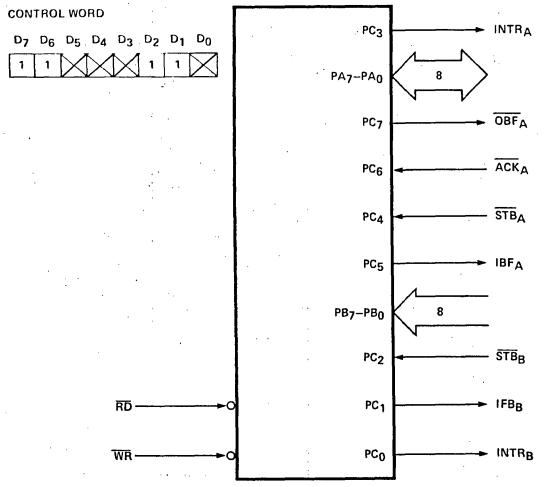
PORT A – MODE 2
PORT B – MODE 0 INPUT



PORT A – MODE 2
PORT B – MODE 0 OUTPUT



PORT A – MODE 2
PORT B – MODE 1 OUTPUT



PORT A – MODE 2
PORT B – MODE 1 INPUT

Using the 8273 SDLC/HDLC Protocol Controller

by John Beaston

INTRODUCTION

The Intel 8273 is a Data Communications Protocol Controller designed for use in systems utilizing either SDLC or HDLC (Synchronous or High-Level Data Link Control) protocols. In addition to the usual features such as full duplex operation, automatic Frame Check Sequence generation and checking, automatic zero bit insertion and deletion, and TTL compatibility found on other single component SDLC controllers; the 8273 features a frame level command structure, a digital phase locked loop, SDLC loop operation, and diagnostics.

The frame level command structure is made possible by the 8273's unique internal dual processor architecture. A high-speed bit processor handles the serial data manipulations and character recognition. A byte processor implements the frame level commands. These dual processors allow the 8273 to control the necessary byte-by-byte operation of the data channel with a minimum of CPU (Central Processing Unit) intervention. For the user this means the CPU has time to take on additional tasks. The digital phase locked loop (DPLL) provides a means of clock recovery from the received data stream on-chip. This feature, along with the frame level commands, makes SDLC loop operation extremely simple and flexible. Diagnostics in the form of both data and clock loopback are available to simplify board debug and link testing. The 8273 is a dedicated function peripheral in the MCS-80/85 Microcomputer family and as such, it interfaces to the 8080/8085 system with a minimum of external hardware.

This application note explains the 8273 as a component and shows its use in a generalized loop configuration and a typical 8085 system. The 8085 system was used to verify the SDLC operation of the 8273 on an actual IBM SDLC data communications link.

The first section of this application note presents an overview of the SDLC/HDLC protocols. It is fairly tutorial in nature and may be skipped by the more knowledgeable reader. The second section describes the 8273 from a functional standpoint with explanation of the block diagram. The software aspects of the 8273, including command examples, are discussed in the third section. The fourth and fifth sections discuss a loop SDLC configuration and the 8085 system respectively.

SDLC/HDLC OVERVIEW

SDLC is a protocol for managing the flow of information on a data communications link. In other words, SDLC can be thought of as an envelope — addressed, stamped, and containing an s.a.s.e. — in which information is transferred from location to location on a data communications link. (Please note that while SDLC is discussed specifically, all comments also apply to HDLC except where noted.) The link may be either point-to-point or multi-point, with the point-to-point configuration being either switched or nonswitched. The information flow may use either full or half duplex exchanges. With this many configurations supported, it is difficult to find a synchronous data communications application where SDLC would not be appropriate.

Aside from supporting a large number of configurations, SDLC offers the potential of a 2× increase in throughput over the presently most prevalent protocol: Bi-Sync. This performance increase is primarily due to two characteristics of SDLC: full duplex operation and the implied acknowledgement of transferred information. The performance increase due to full duplex operation is fairly obvious since, in SDLC, both stations can communicate simultaneously. Bi-Sync supports only half-duplex (two-way alternate) communication. The increase from implied acknowledgement arises from the fact that a station using SDLC may acknowledge previously received information while transmitting different information. Up to 7 messages may be outstanding before an acknowledgement is required. These messages may be acknowledged as a block rather than singly. In Bi-Sync, acknowledgements are unique messages that may not be included with messages containing information and each information message requires a separate acknowledgement. Thus the line efficiency of SDLC is superior to Bi-Sync. On a higher level, the potential of a 2× increase in performance means lower cost per unit of information transferred. Notice that the increase is not due to higher data link speeds (SDLC is actually speed independent), but simply through better line utilization.

Getting down to the more salient characteristics of SDLC; the basic unit of information on an SDLC link is that of the frame. The frame format is shown in Figure 1. Five fields comprise each frame: flag, address, control, information, and frame check sequence. The flag fields (F) form the boundary of the frame and all other fields are positionally related to one of the two flags. All frames start with an opening flag and end with a closing flag. Flags are used for frame synchronization. They also may serve as time-fill characters between frames. (There are no intraframe time-fill characters in SDLC as there are in Bi-Sync.) The opening flag serves as a reference point for the address (A) and control (C) fields. The frame check sequence (FCS) is referenced from the closing flag. All flags have the binary configuration 01111110 (7EH).

SDLC is a bit-oriented protocol, that is, the receiving station must be able to recognize a flag (or any other special character) at any time, not just on an 8-bit boundary. This, of course, implies that a frame may be N-bits in length. (The vast majority of applications tend to use frames which are multiples of 8 bits long, however.)

OPENING FLAG	ADDRESS FIELD (A)	CONTROL FIELD (C)	INFORMATION FIELD (I)	FRAME CHECK SEQUENCE (FCS)	CLOSING FLAG
01111110	8 BITS	8 BITS	ANY LENGTH 0 TO N BITS	16 BITS	01111110

Figure 1. SDLC Frame Format

The fact that the flag has a unique binary pattern would seem to limit the contents of the frame since a flag pattern might inadvertently occur within the frame. This would cause the receiver to think the closing flag was received, invalidating the frame. SDLC handles this situation through a technique called zero bit insertion. This techniques specifies that within a frame a binary 0 be inserted by the transmitter after any succession of five contiguous binary 1s. Thus, no pattern of 01111110 is ever transmitted by chance. On the receiving end, after the opening flag is detected, the receiver removes any 0 following 5 consecutive 1s. The inserted and deleted 0s are not counted for error determination.

Before discussing the address field, an explanation of the roles of an SDLC station is in order. SDLC specifies two types of stations: primary and secondary. The primary is the control station for the data link and thus has responsibility of the overall network. There is only one predetermined primary station, all other stations on the link assume the secondary station role. In general, a secondary station speaks only when spoken to. In other words, the primary polls the secondaries for responses. In order to specify a specific secondary, each secondary is assigned a unique 8-bit address. It is this address that is used in the frame's address field.

When the primary transmits a frame to a specific secondary, the address field contains the secondary's address. When responding, the secondary uses its own address in the address field. The primary is never identified. This ensures that the primary knows which of many secondaries is responding since the primary may have many messages outstanding at various secondary stations. In addition to the specific secondary address, an address common to all secondaries may be used for various purposes. (An all 1s address field is usually used for this "All Parties" address.) Even though the primary may use this common address, the secondaries are expected to respond with their unique address. The address field is always the first 8 bits following the opening flag.

The 8 bits following the address field form the control field. The control field embodies the link-level control of SDLC. A detailed explanation of the commands and responses contained in this field is beyond the scope of this application note. Suffice it to say that it is in the control field that the implied acknowledgement is carried out through the use of frame sequence numbers. None of the currently available SDLC single chip controllers utilize the control field. They simply pass it to the processor for analysis. Readers wishing a more detailed explanation of the control field, or of SDLC in general, should consult the IBM documents referenced on the front page overleaf.

In some types of frames, an information field follows the control field. Frames used strictly for link management may or may not contain one. When an information field is used, it is unrestricted in both content and length. This code transparency is made possible because of the zero bit insertion mentioned earlier and the bit-oriented nature of SDLC. Even main memory core dumps may be transmitted because of this capability. This feature is unique to bit-oriented protocols. Like the control field, the information field is not interpreted by the SDLC device; it is merely transferred to and from memory to be operated on and interpreted by the processor.

The final field is the frame check sequence (FCS). The FCS is the 16 bits immediately preceding the closing flag. This 16-bit field is used for error detection through a Cyclic Redundancy Checkword (CRC). The 16-bit transmitted CRC is the complement of the remainder obtained when the A, C, and I fields are "divided" by a generating polynomial. The receiver accumulates the A, C, and I fields and also the FCS into its internal CRC register. At the closing flag, this register contains one particular number for an error-free reception. If this number is not obtained, the frame was received in error and should be discarded. Discarding the frame causes the station to not update its frame sequence numbering. This results in a retransmission after the station sends an acknowledgement from previous frames. [Unlike all other fields, the FCS is transmitted MSB (Most Significant Bit) first. The A, C, and I fields are transmitted LSB (Least Significant Bit) first.] The details of how the FCS is generated and checked is beyond the scope of this application note and since all single component SDLC controllers handle this function automatically, it is usually sufficient to know only that an error has or has not occurred. The IBM documents contain more detailed information for those readers desiring it.

The closing flag terminates the frame. When the closing flag is received, the receiver knows that the preceding 16 bits constitute the FCS and that any bits between the control field and the FCS constitute the information field.

SDLC does not support an interframe time-fill character such as the SYN character in Bi-Sync. If an unusual condition occurs while transmitting, such as data is not available in time from memory or CTS (Clear-to-Send) is lost from the modem, the transmitter aborts the frame by sending an Abort character to notify the receiver to invalidate the frame. The Abort character consists of eight contiguous 1s sent without zero bit insertion. Intraframe time-fill consists of either flags, Abort characters, or any combination of the two.

While the Abort character protects the receiver from transmitted errors, errors introduced by the transmission medium are discovered at the receiver through the FCS check and a check for invalid frames. Invalid frames are those which are not bounded by flags or are too short, that is, less than 32 bits between flags. All invalid frames are ignored by the receiver.

Although SDLC is a synchronous protocol, it provides an optional feature that allows its use on basically asynchronous data links — NRZI (Non-Return-to-Zero-Inverted) coding. NRZI coding specifies that the signal condition does not change for transmitting a binary 1, while a binary 0 causes a change of state. Figure 2 illustrates NRZI coding compared to the normal NRZ. NRZI coding guarantees that an active line will have a transition at least every 5-bit times; long strings of zeroes cause a transition every bit time, while long strings of 1s are broken up by zero bit insertion. Since asynchronous

operation requires that the receiver sampling clock be derived from the received data, NRZI encoding plus zero bit insertion make the design of clock recovery circuitry easier.

All of the previous discussion has applied to SDLC on either point-to-point or multi-point data networks, SDLC (but not HDLC) also includes specification for a loop configuration. Figure 3 compares these three configurations. IBM uses this loop configuration in its 3650 Retail Store System. It consists of a single loop controller station with one or more down-loop secondary stations. Communications on a loop rely on the secondary stations repeating a received message down loop with a delay of one bit time. The reason for the one bit delay will be evident shortly.

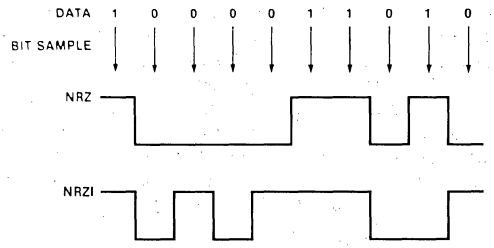


Figure 2. NRZI vs NRZ Encoding

Loop operation defines a new special character: the EOP (End-of-Poll) character which consists of a 0 followed by 7 contiguous, non-zero bit inserted, ones. After the loop controller transmits a message, it idles the line (sends all 1s). The final zero of the closing flag plus the first 7 1s of the idle form an EOP character. While repeating, the secondaries monitor their incoming line for an EOP character. When an EOP is detected, the secondary checks to see if it has a message to transmit. If it does, it changes the seventh 1 to a 0 (the one bit delay allows time for this) and repeats the modified EOP (now alias flag). After this flag is transmitted, the secondary terminates its repeater function and inserts its message (with multiple preceding flags if necessary). After the closing flag, the secondary resumes its one bit delay repeater function. Notice that the final zero of the secondary's closing flag plus the repeated 1s from the controller form an EOP for the next down-loop secondary, allowing it to insert a message if it desires.

One might wonder if the secondary missed any messages from the controller while it was inserting its own message. It does not. Loop operation is basically half-duplex. The controller waits until it receives an EOP before it transmits its next message. The controller's reception of the EOP signifies that the original message has propagated around the loop followed by any messages inserted by the secondaries. Notice that secondaries cannot communicate with one another directly, all secondary-to-secondary communication takes place by way of the controller.

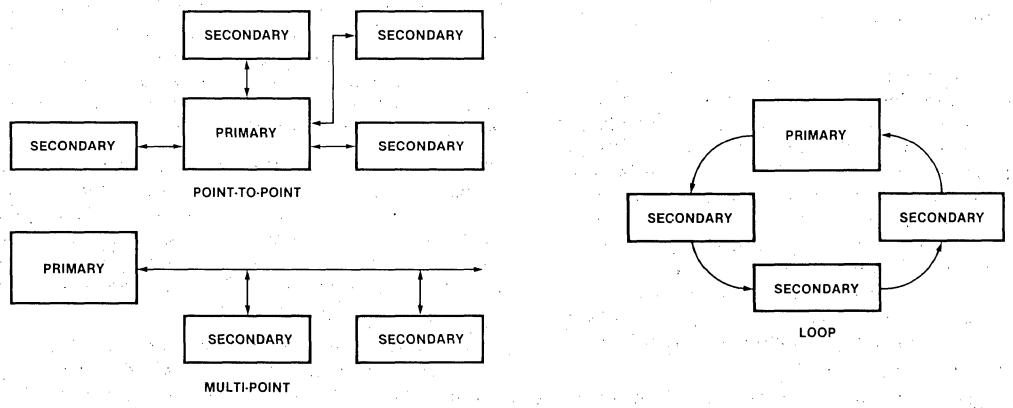


Figure 3. Network Configurations

Loop protocol does not utilize the normal Abort character. Instead, an abort is accomplished by simply transmitting a flag character. Down loop, the receiver sees the abort as a frame which is either too short (if the abort occurred early in the frame) or one with an FCS error. Either results in a discarded frame. For more details on loop operation, please refer to the IBM documents referenced earlier.

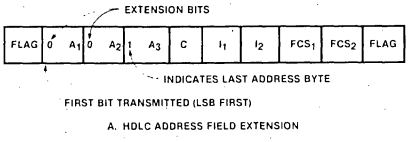
Another protocol very similar to SDLC which the 8273 supports is HDLC (High-Level Data Link Control). There are only three basic differences between the two: HDLC offers extended address and control fields, and the HLDC Abort character is 7 contiguous 1s as opposed to SDLC's 8 contiguous 1s.

Extended addressing, beyond the 256 unique addresses possible with SDLC, is provided by using the address field's least significant bit as the extended address modifier. The receiver examines this bit to determine if the octet should be interpreted as the final address octet. As long as the bit is 0, the octet that contains it is considered an extended address. The first time the bit is a 1, the receiver interprets that octet as the final address octet. Thus the address field may be extended to any number of octets. Extended addressing is illustrated in Figure 4a.

A similar technique is used to extend the control field although the extension is limited to only one extra control octet. Figure 4b illustrates control field extension.

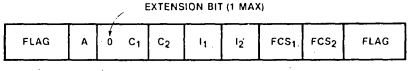
Those readers not yet asleep may have noticed the similarity between the SDLC loop EOP character (a 0 followed by 7 1s) and the HDLC Abort (7 1s). This possible incompatibility is neatly handled by the HDLC protocol not specifying a loop configuration.

This completes our brief discussion of the SDLC/HDLC protocols. Now let us turn to the 8273 in particular and discuss its hardware aspects through an explanation of the block diagram and generalized system schematics.



A. HDLC ADDRESS FIELD EXTENSION

Figure 4a



B. HDLC CONTROL FIELD EXTENSION

Figure 4b

BASIC 8273 OPERATION

It will be helpful for the following discussions to have some idea of the basic operation of the 8273. Each operation, whether it is a frame transmission, reception or port read, etc., is comprised of three phases: the Command, Execution, and Result phases. Figure 5 shows the sequence of these phases. As an illustration of this sequence, let us look at the transmit operation.

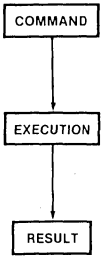


Figure 5. 8273 Operational Phases

When the CPU decides it is time to transmit a frame, the Command phase is entered by the CPU issuing a Transmit Frame command to the 8273. It is not sufficient to just instruct the 8273 to transmit. The frame level command structure sometimes requires more information such as frame length and address and control field content. Once this additional information is supplied, the Command phase is complete and the Execution phase is entered. It is during the Execution phase that the actual operation, in this case a frame transmission, takes place. The 8273 transmits the opening flag, A and C fields, the specified number of I field bytes, inserts the FCS, and closes with the closing flag. Once the closing flag is transmitted, the 8273 leaves the Execution phase and begins the Result phase. During the Result phase the 8273 notifies the CPU of the outcome of the command by supplying interrupt results. In this case, the results would be either that the frame is complete or that some error condition causes the transmission to be aborted. Once the CPU reads all of the results (there is only one for the Transmit Frame command), the Result phase and consequently the operation, is complete. Now that we have a general feeling for the operation of the 8273, let us discuss the 8273 in detail.

HARDWARE ASPECTS OF THE 8273

The 8273 block diagram is shown in Figure 6. It consists of two major interfaces: the CPU module interface and the modem interface. Let's discuss each interface separately.

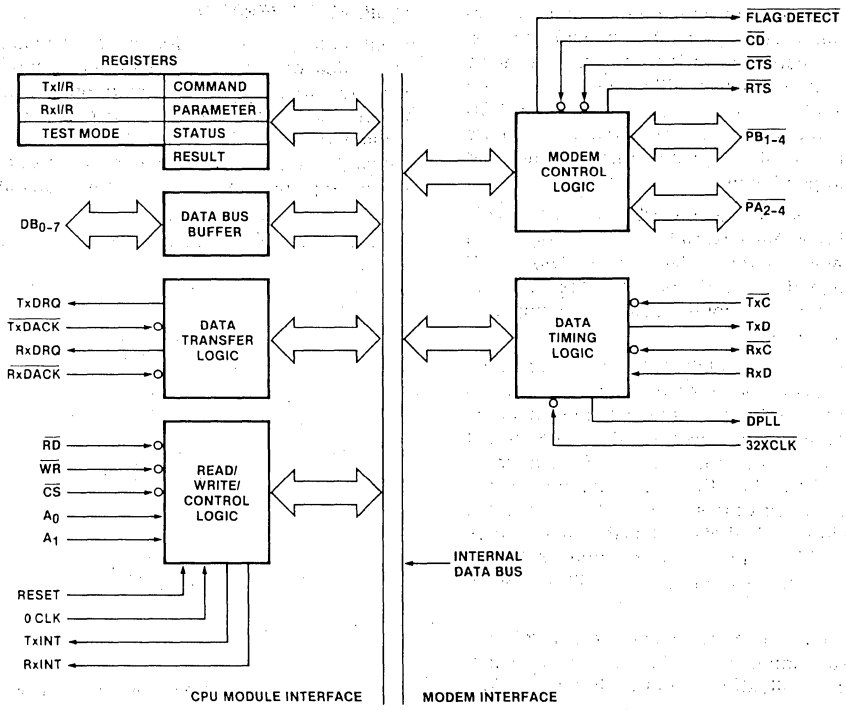


Figure 6. 8273 Block Diagram

CPU Interface

The CPU interface consists of four major blocks: Control/Read/Write logic (C/R/W), internal registers, data transfer logic, and data bus buffers.

The CPU module utilizes the C/R/W logic to issue commands to the 8273. Once the 8273 receives a command and executes it, it returns the results (good/bad completion) of the command by way of the C/R/W logic. The C/R/W logic is supported by seven registers which are addressed via the A_0, A_1, $\overline{RD}$, and $\overline{WR}$ signals, in addition to $\overline{CS}$. The A_0 and A_1 signals are generally derived from the two low order bits of the CPU module address bus while $\overline{RD}$ and $\overline{WR}$ are the normal I/O Read and Write signals found on the system control bus. Figure 7 shows the address of each register using the C/R/W logic. The function of each register is defined as follows:

ADDRESS INPUTS		CONTROL INPUTS	
A_1	A_0	$\overline{CS} \cdot \overline{RD}$	$\overline{CS} \cdot \overline{WR}$
0	0	STATUS	COMMAND
0	1	RESULT	PARAMETER
1	0	TxI/R	TEST MODE
1	1	RxI/R	—

Figure 7. 8273 Register Selection

Command — 8273 operations are initiated by writing the appropriate command byte into this register.

Parameter — Many commands require more information than found in the command itself. This additional information is provided by way of the parameter register.

Immediate Result (Result) — The completion information (results) for commands which execute immediately are provided in this register.

Transmit Interrupt Result (TxI/R) — Results of transmit operations are passed to the CPU in this register.

Receiver Interrupt Result (RxI/R) — Receive operation results are passed to the CPU via this register.

Status — The general status of the 8273 is provided in this register. The Status register supplies the handshaking necessary during various phases of the 8273 operation.

Test Mode — This register provides a software reset function for the 8273.

The commands, parameters, and bit definition of these registers are discussed in the following software section. Notice that there are not specific transmit or receive data registers. This feature is explained in the data transfer logic discussion.

The final elements of the C/R/W logic are the interrupt lines (RxINT and TxINT). These lines notify the CPU module that either the transmitter or the receiver requires service; i.e., results should be read from the appropriate interrupt result register or a data transfer is required. The interrupt request remains active until all the associated interrupt results have been read or the data transfer is performed. Though using the interrupt lines relieves the CPU module of the task of polling the 8273 to check if service is needed, the state of each interrupt line is reflected by a bit in the Status register and non-interrupt driven operation is possible by examing the contents of these bits periodically.

The 8273 supports two independent data interfaces through the data transfer logic; receive data and transmit data. These interfaces are programmable for either DMA or non-DMA data transfers. While the choice of the configuration is up to the system designer, it is based on the intended maximum data rate of the communications channel. Figure 8 illustrates the transfer rate of data bytes that are acquired by the 8273 based on link data rate. Full-duplex data rates above 9600 baud usually require DMA. Slower speeds may or may not require DMA depending on the task load and interrupt response time of the processor.

Figure 9 shows the 8273 in a typical DMA environment. Notice that a separate DMA controller, in this case the Intel 8257, is required. The DMA controller supplies the timing and addresses for the data transfers while the 8273 manages the requesting of transfers and the actual counting of the data block lengths. In this case, elements of the data transfer interface are:

TxDRQ: Transmit DMA Request — Asserted by the 8273, this line requests a DMA transfer from memory to the 8273 for transmit.

TxDACK: Transmit DMA Acknowledge — Returned by the 8257 in response to TxDRQ, this line notifies the 8273 that a request has been granted, and provides access to the transmitter data register.

RxDRQ: Receiver DMA Request — Asserted by the 8273, it requests a DMA transfer from the 8273 to memory for a receive operation.

TxDACK: Receiver DMA Acknowledge — Returned by the 8257, it notifies the 8273 that a receive DMA cycle has been granted, and provides access to the receiver data register.

RD: Read — Supplied by the 8257 to indicate data is to be read from the 8273 and placed in memory.

WR: Write — Supplied by the 8257 to indicate data is to be written to the 8273 from memory.

To request a DMA transfer the 8273 raises the appropriate DMA request line; let us assume it is a transmitter request (TxDRQ). Once the 8257 obtains control of the system bus by way of its HOLD and HLDA (hold acknowledge) lines, it notifies the 8273 that TxDRQ has been granted by returning $\overline{\text{TxDACK}}$ and $\overline{\text{WR}}$. The $\overline{\text{TxDACK}}$ and $\overline{\text{WR}}$ signals transfer data to the 8273 for a transmit, independent of the 8273 chip select pin ($\overline{\text{CS}}$). A similar sequence of events occurs for receiver requests. This "hard select" of data into the transmitter or out of the receiver alleviates the need for the normal transmit and receive data registers addressed by a combination of address lines, CS, and WR or RD. Competitive devices that do not have this "hard select" feature require the use of an external multiplexer to supply the correct inputs for register selection during DMA. (Do not forget that the SDLC controller sees both the addresses and control signals supplied by the DMA controller during DMA cycles.) Let us look at typical frame transmit and frame receive sequences to better see how the 8273 truly manages the DMA data transfer.

Before a frame can be transmitted, the DMA controller is supplied, by the CPU, the starting address for the desired information field. The 8273 is then commanded to transmit a frame. (Just how this is done is covered later during our software discussion.) After the command, but before transmission begins, the 8273 needs a little more information (parameters). Four parameters are required for the transmit frame command: the address field byte, the control field byte, and two bytes which are the least significant and most significant bytes of the information field byte length. Once all four parameters are loaded, the 8273 makes RTS (Request-to-Send) active and waits for CTS (Clear-to-Send) to go active. Once CTS is active, the 8273 starts the frame transmission. While the 8273 is transmitting the opening flag, address field, and control field; it starts making transmitter DMA requests. These requests continue at character (byte) boundaries until the pre-loaded number of bytes of information field have been transmitted. At this point the requests stop, the FCS and closing flag are transmitted, and the TxINT line is raised, signaling the CPU that the frame transmission is complete. Notice that after the initial command and parameter loading, absolutely no CPU intervention was required (since DMA is used for data transfers) until the entire frame was transmitted. Now let's look at a frame reception.

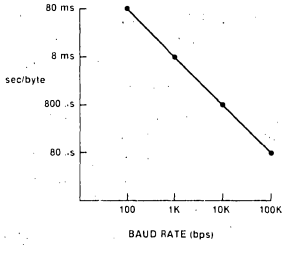


Figure 8. Byte Transfer Rate vs Baud Rate

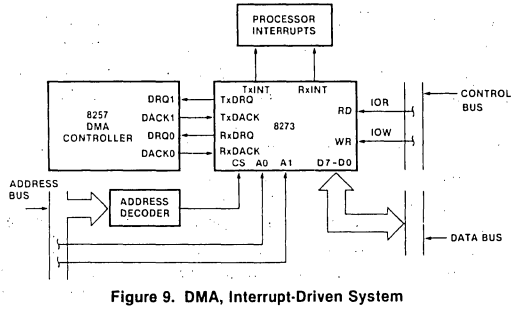


Figure 9. DMA, Interrupt-Driven System

The receiver operation is very similar. Like the initial transmit sequence, the DMA controller is loaded with a starting address for a receiver data buffer and the 8273 is commanded to receive. Unlike the transmitter, there are two different receive commands: General Receive, where all received frames are transferred to memory, and Selective Receive, where only frames having an address field matching one of two preprogrammed 8273 address fields are transferred to memory. Let's assume for now that we want to general receive. After the receive command, two parameters are required before the receiver becomes active: the least significant and most significant bytes of the receiver buffer length. Once these bytes are loaded, the receiver is active and the CPU may return to other tasks. The next frame appearing at the receiver input is transferred to memory using receiver DMA requests. When the closing flag is received, the 8273 checks the FCS and raises its RxINT line. The CPU can then read the results which indicate if the frame was error-free or not. (If the received frame had been longer than the pre-loaded buffer length, the CPU would have been notified of that occurrence earlier with a receiver error interrupt. The command description section contains a complete list of error conditions.) Like the transmit example, after the initial command, the CPU is free for other tasks until a frame is completely received. These examples have illustrated the 8273's management of both the receiver and transmitter DMA channels.

It is possible to use the DMA data transfer interface in a non-DMA interrupt-driven environment. In this case, 4 interrupt levels are used: one each for TxINT and RxINT, and one each for TxDRQ and RxDRQ. This configuration is shown in Figure 10. This configuration offers the advantages that no DMA controller is required and data requests are still separated from result (completion) requests. The disadvantages of the configuration are that 4 interrupt levels are required and that the CPU must actually supply the data transfers. This, of course, reduces the maximum data rate compared to the configuration based strictly on DMA. This system could use an Intel 8259 8-level Priority Interrupt Controller to supply a vectored CALL (subroutine) address based on requests on its inputs. The 8273 transmitter and receiver make data requests by raising the respective DRQ line. The CPU is interrupted by the 8259 and vectored to a data transfer routine. This routine either writes (for transmit) or reads (for receive) the 8273 using the respective TxDACK or RxDACK line. As in the case above, the DACK lines serve as "hard" chip selects into and out of the 8273. ($\overline{\text{TxDACK}}$ + $\overline{\text{WR}}$ writes data into the 8273 for transmit. $\overline{\text{RxDACK}}$ + $\overline{\text{RD}}$ reads data from the 8273 for receive.) The CPU is notified of operation completion and results by way of TxINT and RxINT lines. Using the 8273, and the 8259, in this way, provides a very effective, yet simple, interrupt-driven interface.

Figure 11 illustrates a system very similar to that described above. This system utilizes the 8273 in a non-DMA data transfer mode as opposed to the two DMA approaches shown in Figures 9 and 10. In the non-DMA case, data transfer requests are made on the TxINT and RxINT lines. The DRQ lines are not used. Data transfer requests are separated from result requests by a bit in the Status register. Thus, in response to an interrupt, the CPU reads the Status register and branches to either a result or a data transfer routine based on the status of one bit. As before, data transfers are made via using the DACK lines as chip selects to the transmitter and receiver data registers.

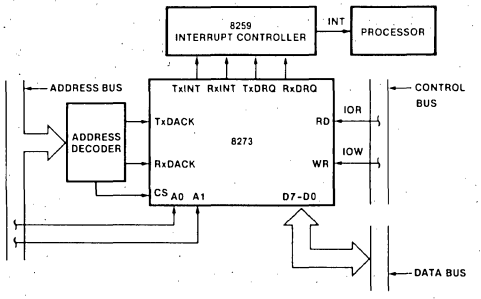


Figure 10. Interrupt-Based DMA System

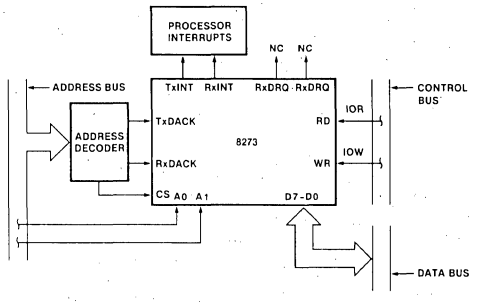


Figure 11. Non-DMA Interrupt-Driven System

Figure 12 illustrates the simplest system of all. This system utilizes polling for all data transfers and results. Since the interrupt pins are reflected in bits in the Status register, the software can read the Status register periodically looking for one of these to be set. If it finds an INT bit set, the appropriate Result Available bit is examined to determine if the "interrupt" is a data transfer or completion result. If a data transfer is called for, the DACK line is used to enter or read the data from the 8273. If the interrupt is a completion result, the appropriate result register is read to determine the good/bad completion of the operation.

The actual selection of either DMA or non-DMA modes is controlled by a command issued during initialization. This command is covered in detail during the software discussion.

The final block of the CPU module interface is the Data Bus Buffer. This block supplies the tri-state, bidirectional data bus interface to allow communication to and from the 8273.

Modem Interface

As the name implies, the modem interface is the modem side of the 8273. It consists of two major blocks: the modem control block and the serial data timing block.

The modem control block provides both dedicated and user-defined modem control functions. All signals supported by this interface are active low so that EIA inverting drivers (MC1488) and inverting receivers (MC1489) may be used to interface to standard modems.

Port A is a modem control input port. Its representation on the data bus is shown in Figure 13. Bits D_0 and D_1 have dedicated functions. D_0 reflects the logical state of the $\overline{CTS}$ (Clear-to-Send) pin. [If $\overline{CTS}$ is active (low), D_0 is a 1.] This signal is used to condition the start of a transmission. The 8273 waits until $\overline{CTS}$ is active before it starts transmitting a frame. While transmitting, if $\overline{CTS}$ goes inactive, the frame is aborted and the CPU is interrupted. When the CPU reads the interrupt result, a $\overline{CTS}$ failure is indicated.

D_1 reflects the logical state of the $\overline{CD}$ (Carrier Detect) pin. $\overline{CD}$ is used to condition the start of a frame reception. $\overline{CD}$ must be active in time for a frame's address field. If $\overline{CD}$ is lost (goes inactive) while receiving a frame, an interrupt is generated with a $\overline{CD}$ failure result. $\overline{CD}$ may go inactive between frames.

Bits D_2 thru D_4 reflect the logical state of the $\overline{PA_2}$ thru $\overline{PA_4}$ pins respectively. These inputs are user defined. The 8273 does not interrogate or manipulate these bits. Bits D_5, D_6, and D_7 are not used and each is read as a 1 for a Read Port A command.

Port B is a modem control output port. Its data bus representation is shown in Figure 14. As in Port A, the bit values represent the logical condition of the pins. D_0 and D_5 are dedicated function outputs. D_0 represents the $\overline{RTS}$ (Request-to-Send) pin. $\overline{RTS}$ is normally used to notify the modem that the 8273 wishes to transmit. This function is handled automatically by the 8273. If $\overline{RTS}$ is inactive (pin is high) when the 8273 is commanded to transmit, the 8273 makes it active and then waits for $\overline{CTS}$ before transmitting the frame. One byte time after the end of the frame, the 8273 returns $\overline{RTS}$ to its inactive state. However, if $\overline{RTS}$ was active when a transmit command is issued, the 8273 leaves it active when the frame is complete.

Bit D_5 reflects the state of the $\overline{\text{Flag Detect}}$ pin. This pin is activated whenever an active receiver sees a flag character. This function is useful to activate a timer for line activity timeout purposes.

Bits D_1 thru D_4 provide four user-defined outputs. Pins $\overline{PB_1}$ thru $\overline{PB_4}$ reflect the logical state of these bits. The 8273 does not interrogate or manipulate these bits. D_6 and D_7 are not used. In addition to being able to output to Port B, Port B may be read using a Read Port B command. All Modem control output pins are forced high on reset. (All commands mentioned in this section are covered in detail later.).

The final block to be covered is the serial data timing block. This block contains two sections: the serial data logic and the digital phase locked loop (DPLL).

Elements of the serial data logic section are the data pins, TxD (transmit data output) and RxD (receive data input), and the respective data clocks, $\overline{TxC}$ and $\overline{RxC}$. The transmit and receive data is synchronized by the $\overline{TxC}$ and $\overline{RxC}$ clocks. Figure 15 shows the timing for these signals. The leading edge (negative transition) of $\overline{TxC}$ generates new transmit data and the trailing edge (positive transition) of $\overline{RxC}$ is used to capture the receive data.

It is possible to reconfigure this section under program control to perform diagnostic functions; both data and clock loopback are available. In data loopback mode, the TxD pin is internally routed to the RxD pin. This allows simple board checkout since the CPU can send an SDLC message to itself. (Note that transmitted data will still appear on the TxD pin.)

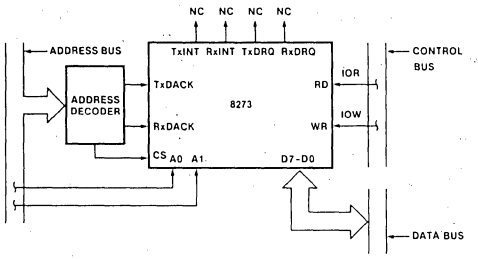


Figure 12. Polled System

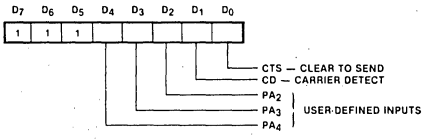


Figure 13. Port A (Input) Bit Definition

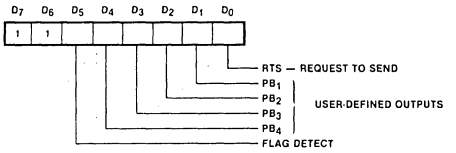


Figure 14. Port B (Output) Bit Definition

When data loopback is utilized, the receiver may be presented incorrect sample timing ($\overline{RxC}$) by the external circuitry. Clock loopback overcomes this problem by allowing the internal routing of $\overline{TxC}$ and $\overline{RxC}$. Thus the same clock used to transmit the data is used to receive it. Examination of Figure 15 shows that this method ensures bit synchronism. The final element of the serial data logic is the Digital Phase Locked Loop.

The DPLL provides a means of clock recovery from the received data stream. This feature allows the 8273 to interface without external synchronizing logic to low cost asynchronous modems (modems which do not supply clocks). It also makes the problem of clock timing in loop configurations trivial.

To use the DPLL, a clock at 32 times the required baud rate must be supplied to the $\overline{32 \times CLK}$ pin. This clock provides the interval that the DPLL samples the received data. The DPLL uses the 32× clock and the received data to generate a pulse at the $\overline{DPLL}$ output pin. This $\overline{DPLL}$ pulse is positioned at the nominal center of the received data bit cell. Thus the $\overline{DPLL}$ output may be wired to $\overline{RxC}$ and/or $\overline{TxC}$ to supply the data timing. The exact position of the pulse is varied depending on the line noise and bit distortion of the received data. The adjustment of the $\overline{DPLL}$ position is determined according to the rules outlined in Figure 16.

Adjustments to the sample phase of $\overline{DPLL}$ with respect to the received data is made in discrete increments. Referring to Figure 16, following the occurrence of $\overline{DPLL}$ pulse A, the DPLL counts $\overline{32 \times CLK}$ pulses and examines the received data for a data edge. Should no edge be detected in 32 pulses, the $\overline{DPLL}$ positions the next $\overline{DPLL}$ pulse (B) at 32 clock pulses from pulse A. Since no new phase information is contained in the data stream, the sample phase is assumed to be at nominal 1× baud rate. Now assume a data edge occurs after $\overline{DPLL}$ pulse B. The distance from B to the next pulse C is influenced according to which quadrant (A_1, B_1, B_2, or A_2) the data edge falls in. (Each quadrant represents 8 $\overline{32 \times CLK}$ times.) For example, if the edge is detected in quadrant A_1, it is apparent that pulse B was too close to the data edge and the time to the next pulse must be shortened. The adjustment for quadrant A_1 is specified as −2. Thus, the next $\overline{DPLL}$ pulse, pulse C, is positioned 32 − 2 or 30 $\overline{32 \times CLK}$ pulses following $\overline{DPLL}$ pulse B. This adjustment moves pulse C closer to the nominal bit center of the next received data cell. A data edge occurring in quadrant B_2 would have caused the adjustment to be small, namely 32 + 1 or 33 $\overline{32 \times CLK}$ pulses. Using this technique, the $\overline{DPLL}$ pulse converges to the nominal bit center within 12 data transitions, worse case — 4-bit times adjusting through quadrant A_1 or A_2 and 8-bit times adjusting through B_1 or B_2.

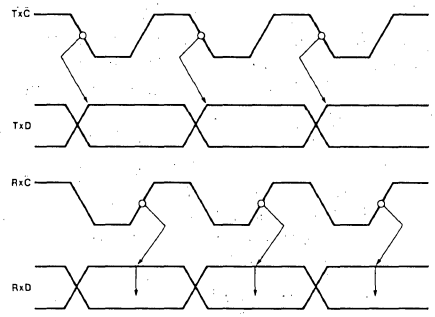


Figure 15. Transmit/Receive Timing

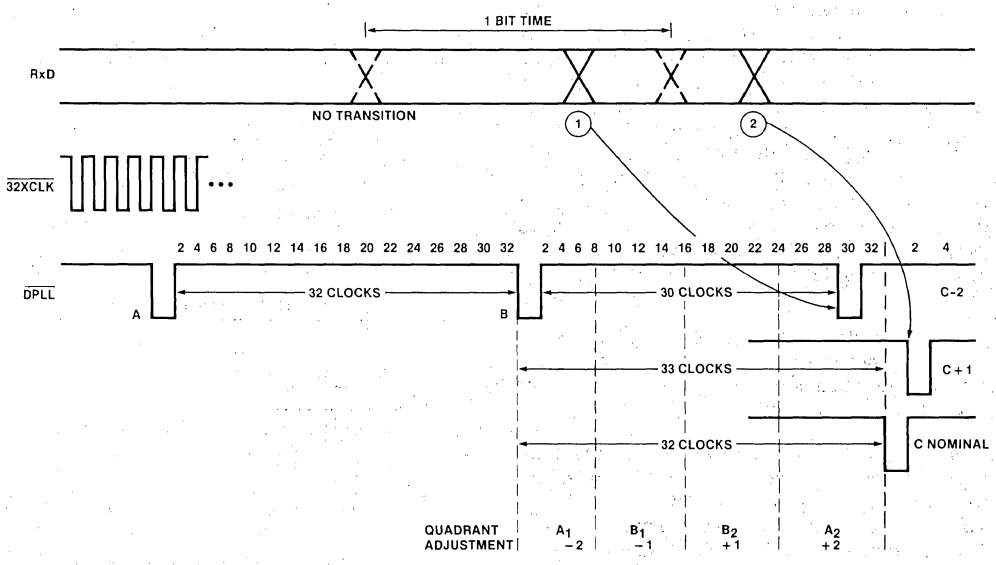


Figure 16. DPLL Phase Adjustments

When the receive data stream goes idle after 15 ones, DPLL pulses are generated at 32 pulse intervals of the 32x CLK. This feature allows the DPLL pulses to be used as both transmitter and receiver clocks.

In order to guarantee sufficient transitions of the received data to enable the $\overline{DPLL}$ to lock, NRZI encoding of the data is recommended. This ensures that, within a frame, data transitions occur at least every five bit times — the longest sequence of 1s which may be transmitted with zero bit insertion. It is also recommended that frames following a line idle be transmitted with preframe sync characters which provide a minimum of 12 transitions. This ensures that the $\overline{DPLL}$ is generating $\overline{DPLL}$ pulses at the nominal bit centers in time for the opening flag. (Two 00H characters meet this requirement by supplying 16 transitions with NRZI encoding. The 8273 contains a mode which supplies such a preframe sync.)

Figure 17 illustrates 8273 clock configurations using either synchronous or asynchronous modems. Notice how the $\overline{DPLL}$ output is used for both $\overline{TxC}$ and $\overline{RxC}$ in the asynchronous case. This feature eliminates the need for external clock generation logic where low cost) asynchronous modems are used and also allows direct connection of 8273s for the ultimate in low cost data links. The configuration for loop applications is discussed in a following section.

This completes our discussion of the hardware aspects of the 8273. Its software aspects are now discussed.

SOFTWARE ASPECTS OF THE 8273

The software aspects of the 8273 involve the communication of both commands from the CPU to the 8273 and the return of results of those commands from the 8273 to the CPU. Due to the internal processor architecture of the 8273, this CPU-8273 communication is basically a form of interprocessor communication. Such communication usually requires a form of protocol of its own. This protocol is implemented through use of handshaking supplied in the 8273 Status register. The bit definition of this register is shown in Figure 18.

CBSY: Command Busy — CBSY indicates when the 8273 is in the command phase. CBSY is set when the CPU writes a command into the Command register, starting the Command phase. It is reset when the last parameter is deposited in the Parameter register and accepted by the 8273, completing the Command phase.

CBF: Command Buffer Full — When set, this bit indicates that a byte is present in the Command register. This bit is normally not used.

CPBF: Command Parameter Buffer Full — This bit indicates that the Parameter register contains a parameter. It is set when the CPU deposits a parameter in the Parameter register. It is reset when the 8273 accepts the parameter.

CRBF: Command Result Buffer Full — This bit is set when the 8273 places a result from an immediate type command in the Result register. It is reset when the CPU reads the result from the Result register.

RxINT: Receiver Interrupt — The state of the RxINT pin is reflected by this bit. RxINT is set by the 8273 whenever the receiver needs servicing. RxINT is reset when the CPU reads the results or performs the data transfer.

TxINT: Transmitter Interrupt — This bit is identical to RxINT except action is initiated based on transmitter interrupt sources.

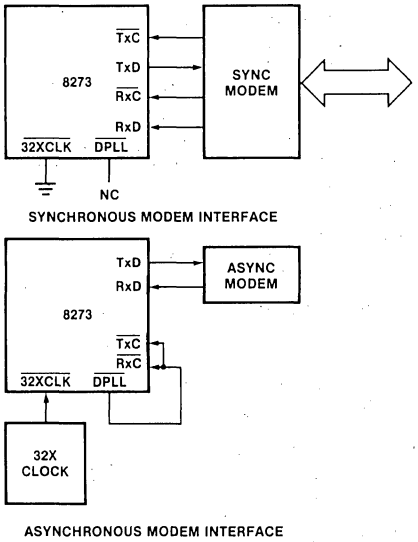


Figure 17. Serial Data Timing Configuration

RxIRA: Receiver Interrupt Result Available — RxIRA is set when the 8273 places an interrupt result byte into the RxI/R register. RxIRA is reset when the CPU reads the RxI/R register.

TxIRA: Transmitter Interrupt Result Available — TxIRA is the corresponding Result Available bit for the transmitter. It is set when the 8273 places an interrupt result byte in the TxI/R register and reset when the CPU reads the register.

The significance of each of these bits will be evident shortly. Since the software requirements of each 8273 phase are essentially independent, each phase is covered separately.

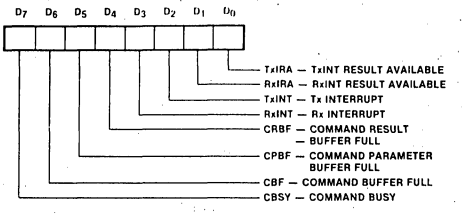


Figure 18. Status Register Format

Command Phase Software

Recalling the Command phase description in an earlier section, the CPU starts the Command phase by writing a command byte into the 8273 Command register. If further information about the command is required by the 8273, the CPU writes this information into the Parameter register. Figure 19 is a flowchart of the Command phase. Notice that the CBSY and CPBF bits of the Status register are used to handshake the command and parameter bytes. Also note that the chart shows that a command may not be issued if the Status register indicates the 8273 is busy (CBSY = 1). If a command is issued while CBSY = 1, the original command is overwritten and lost. (Remember that CBSY signifies the command phase is in progress and not the actual execution of the command.) The flowchart also includes a Parameter buffer full check. The CPU must wait until CPBF = 0 before writing a parameter to the Parameter register. If a parameter is issued while CPBF = 1, the previous parameter is overwritten and lost. An example of command output assembly language software is provided in Figure 20a. This software assumes that a command buffer exists in memory. The buffer is pointed at by the HL register. Figure 20b shows the command buffer structure.

The 8273 is a full duplex device, i.e., both the transmitter and receiver may be executing commands or passing interrupt results at any given time. (Separate Rx and Tx interrupt pins and result registers are provided for this reason.) However, there is only one Command register. Thus, the Command register must be used for only one command sequence at a time and the transmitter and receiver may never be simultaneously in a command phase. A detailed description of the commands and their parameters is presented in a following section.

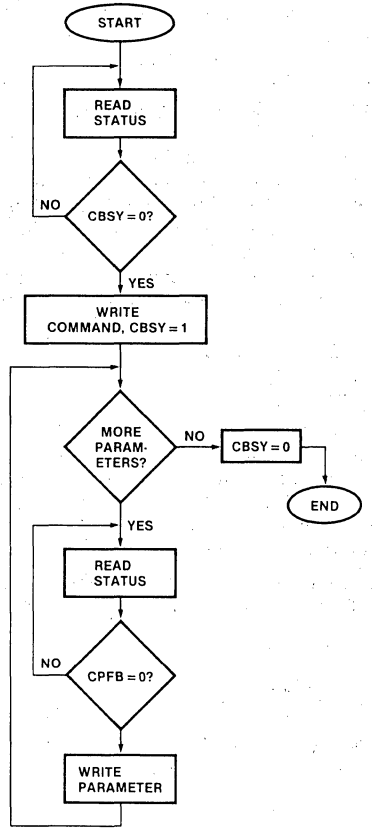


Figure 19. Command Phase Flowchart

```
;FUNCTION: COMMAND DISPATCHER
;INPUTS: HL - COMMAND BUFFER ADDRESS
;OUTPUTS: NONE
;CALLS: NONE
;DESTROYS: A,B,H,L,F/F'S
;DESCRIPTION: CMDOUT ISSUES THE COMMAND + PARAMETERS
;IN THE COMMAND BUFFER POINTED AT BY HL
;
CMDOUT: LXI     H,CMDBUF;POINT HL AT BUFFER
        MOV     B,M     ;1ST ENTRY IS PAR. COUNT
        INX     H       ;POINT AT COMMAND BYTE
CMD1:   IN      STAT73  ;READ 8273 STATUS
        RLC     ;ROTATE CBSY INTO CARRY
        JC      CMD1    ;WAIT UNTIL CBSY=0
        MOV     A,M     ;MOVE COMMAND BYTE TO A
        OUT     COMM73  ;PUT COMMAND IN COMMAND REG
CMD2:   MOV     A,B     ;GET PARAMETER COUNT
        ANA     A       ;TEST IF ZERO
        RZ      ;IF 0 THEN DONE
        INX     H       ;NOT DONE, SO POINT AT NEXT PAR
        DCR     B       ;DEC PARAMETER COUNT
CMD3:   IN      STAT73  ;READ 8273 STATUS
        ANI     CPBF    ;TEST CPBF BIT
        JNZ     CMD3    ;WAIT UNTIL CPBF IS 0
        MOV     A,M     ;GET PARAMETER FROM BUFFER
        OUT     PARM73  ;OUTPUT PAR TO PARAMETER REG
        JMP     CMD2    ;CHECK IF MORE PARAMETERS
```

Figure 20A. Command Phase Software

+4	PARAMETER 3	
+3	PARAMETER 2	
+2	PARAMETER 1	
+1	COMMAND	
CMDBUF:	PARAMETER COUNT	← HL

Figure 20B. Command Buffer Format

Execution Phase Software

During the Execution phase, the operation specified by the Command phase is performed. If the system utilizes DMA for data transfers, there is no CPU involvement during this phase, so no software is required. If non-DMA data transfers are used, either interrupts or polling is used to signal a data transfer request.

For interrupt-driven transfers the 8273 raises the appropriate INT pin. When responding to the interrupt, the CPU must determine whether it is a data transfer request or an interrupt signaling that an operation is complete and results are available. The CPU determines the cause by reading the Status register and interrogating the associated IRA (Interrupt Result Available) bit (TxIRA for TxINT and RxIRA for RxINT). If the IRA = 0, the interrupt is a data transfer request. If the IRA = 1, an operation is complete and the associated Interrupt Result register must be read to determine the completion status (good/bad/etc.). A software interrupt handler implementing the above sequence is presented as part of the Result phase software.

When polling is used to determine when data transfers are required, the polling routine reads the Status register looking for one of the INT bits to be set. When a set INT bit is found, the corresponding IRA bit is examined. Like in the interrupt-driven case, if the IRA = 0, a data transfer is required. If IRA = 1, an operation is complete and the Interrupt Result register needs to be read. Again, example polling software is presented in the next section.

Result Phase Software

During the Result phase the 8273 notifies the CPU of the outcome of a command. The Result phase is initiated by either a successful completion of an operation or an error detected during execution. Some commands such as reading or writing the I/O ports provide immediate results, that is, there is essentially no delay from the issuing of the command and when the result is available. Other commands such as frame transmit, take time to complete so their result is not available immediately. Separate result registers are provided to distinguish these two types of commands and to avoid interrupt handling for simple results.

Immediate results are provided in the Result register. Validity of information in this register is indicated to the CPU by way of the CRBF bit in the Status register. When the CPU completes the Command phase of an immediate command, it polls the Status register waiting until CRBF = 1. When this occurs, the CPU may read the Result register to obtain the immediate result. The Result register provides only the results from immediate commands.

Example software for handling immediate results is shown in Figure 21. The routine returns with the result in the accumulator. The CPU then uses the result as is appropriate.

All non-immediate commands deal with either the transmitter or receiver. Results from these commands are provided in the TxI/R (Transmit Interrupt Result) and RxI/R (Receive Interrupt Result) registers respectively. Results in these registers are conveyed to the CPU by the TxIRA and RxIRA bits of the Status register. Results of non-immediate commands consist of one byte result interrupt code indicating the condition for the interrupt and, if required, one or more bytes supplying additional information. The interrupt codes and the meaning of the additional results are covered following the detailed command description.

Non-immediate results are passed to the CPU in response to either interrupts or polling of the Status register. Figure 22 illustrates an interrupt-driven result handler. (Please note that all of the software presented in this application note is not optimized for either speed or code efficiency. They are provided as a guide and to illustrate concepts.) This handler provides for interrupt-driven data transfers as was promised in the last section. Users employing DMA-based transfers do not need the lines where the IRA bit is tested for zero. (These lines are denoted by an asterisk in the comments column.) Note that the INT bit is used to determine when all results have been read. All results must be read. Otherwise, the INT bit (and pin) will remain high and further interrupts may be missed. These routines place the results in a result buffer pointed at by RCRBUF and TxRBUF.

A typical result handler for systems utilizing polling is shown in Figure 23. Data transfers are also handled by this routine. This routine utilizes the routines of Figure 22 to handle the results.

At this point, the reader should have a good conceptual feel about how the 8273 operates. It is now time for the particulars of each command to be discussed.

```
;FUNCTION: IMDRLT
;INPUTS: NONE
;OUTPUTS: RESULT REGISTER IN A
;CALLS: NONE
;DESTROYS: A,F/F'S
;DESCRIPTION: IMDRLT IS CALLED AFTER A CMDOUT FOR AN
;IMMEDIATE COMMAND TO READ THE RESULT REGISTER
;
IMDRLT: IN      STAT73  ;READ 8273 STATUS
        ANI     CRBF    ;TEST IF RESULT REG READY
        JZ      IMDRLT  ;WAIT IF CRBF=0
        IN      RESL73  ;READ RESULT REGISTER
        RET     ;RETURN
```

Figure 21. Immediate Result Handler

```
;FUNCTION: RXI - INTERRUPT DRIVEN RESULT/DATA HANDLER
;INPUTS: RCRBUF, RCVPNT
;CALLS: NONE
;OUTPUTS: RCRBUF, RCVPNT
;DESTROYS: NOTHING
;DESCRIPTION: RXI IS ENTERED AT A RECEIVER INTERRUPT.
;THE INTERRUPT IS TESTED FOR DATA TRANSFER (IRA=0)
;OR RESULT (IRA=1).  FOR DATA TRANSFER, THE DATA IS
;PLACED IN A BUFFER AT RCVPNT. RESULTS ARE PLACED IN
;A BUFFER AT RCRBUF.
;A FLAG(RXFLAG) IS SET IF THE INTERRUPT WAS A RESULT.
;(DATA TRANSFER INSTRUCTIONS ARE DENOTED BY (*) AND
;MAYBE ELIMINATED BY USERS USING DMA.
;
RXI:    PUSH    H       ;SAVE HL
        PUSH    PSW     ;SAVE PSW
        PUSH    B       ;SAVE B
        IN      STAT73  ; (*) READ 8273 STATUS
        ANI     RXIRA   ; (*) TEST IRA BIT
        JZ      RXI2    ; (*) IF 0, DATA TRANSFER NEEDED
RXI1:   LHLD    RCRBUF  ;GET RESULT BUFFER POINTER
        IN      STAT73  ;READ 8273 STATUS AGAIN
        ANI     RXINT   ;TEST INT BIT
        JZ      RXI4    ;IF 0, THEN DONE
        IN      STAT73  ;READ 8273 STATUS AGAIN
        ANI     RXIRA   ;TEST IRA AGAIN
        JZ      RXI1    ;LOOP UNTIL RESULT IS READY
        IN      RXIR73  ;READY, READ RXI/R
        MOV     M,A     ;STORE RESULT IN BUFFER
        INX     H       ;BUMP RESULT POINTER
        SHLD    RCRBUF  ;RESTORE BUFFER POINTER
        JMP     RXI1    ;GO BACK TO SEE IF MORE
RXI2:   SHLD    RCVPNT  ; (*) GET DATA BUFFER POINTER
        IN      RCVDAT  ; (*) READ DATA VIA RXDACK
        MOV     M,A     ; (*) STORE DATA IN BUFFER
        INX     H       ; (*) BUMP DATA POINTER
        JMP     RXI3    ; (*) DONE
RXI4:   MVI     A,01H   ;SET RX FLAG TO SHOW COMPLETION
        STA     RXFLAG  ;COMPLETION
RXI3:   POP     B       ;RESTORE BC
        POP     PSW     ;RESTORE PSW
        POP     H       ;RESTORE HL
        EI      ;ENABLE INTERRUPTS
        RET     ;DONE

;FUNCTION: TXI - INTERRUPT DRIVEN RESULT/DATA HANDLER
;INPUTS: TXRBUF, TXPNT, TXFLAG
;OUTPUTS: TXRBUF, TXPNT, TXFLAG
;CALLS: NONE
;DESTROYS: NOTHING
;DESCRIPTION: TXI IS ENTERED AT A TRANSMITTER INTERRUPT.
;THE INTERRUPT IS TESTED BY WAY OF THE IRA BIT TO SEE
;IF A DATA TRANSFER OR RESULT COMPLETION HAS OCCURED.
;FOR DATA TRANSFERS (IRA=0), THE DATA IS OBTAINED FROM
;A BUFFER LOCATION POINTED AT BY TXPNT.  FOR COMPLETION,
;(IRA=1), THE RESULTS ARE READ AND PLACED AT A RESULT
;BUFFER POINTED AT BY TXRBUF, AND THE TXFLAG IS SET
;TO INDICATE TO THE MAIN PROGRAM THAT A OPERATION IS
;COMPLETE. TX OPERATIONS HAVE ONLY ONE RESULT.
;DATA TRANSFER INSTRUCTIONS ARE DENOTED BY (*). THESE
;MAYBE REMOVED BY USERS USING DMA.
;
TXI:    PUSH    H       ;SAVE HL
        PUSH    PSW     ;SAVE PSW
        IN      STAT73  ; (*) READ 8273 STATUS
        ANI     TXIRA   ; (*) TEST TXIRA BIT
        JZ      TXI2    ; (*) IF 0, DATA TRANSFER
        IN      TXIR73  ;1, THEN READ TXIR
        LHLD    TXRBUF  ;GET RESULT BUFFER POINTER
        MOV     M,A     ;STORE RESULT IN BUFFER
        INX     H       ;BUMP RESULT POINTER
        SHLD    TXRBUF  ;RESTORE RESULT POINTER
        MVI     A,01H   ;SET TXFLAG TO SHOW COMPLETION
        STA     TXFLAG  ;SET FLAG
TXI1:   POP     PSW     ;RESTORE PSW
        POP     H       ;RESTORE HL
        EI      ;ENABLE INTERRUPTS
        RET     ;DONE
TXI2:   LHLD    TXPNT   ; (*) GET DATA POINTER
        MOV     A,M     ; (*) GET DATA FROM BUFFER
        OUT     TXDATA  ; (*) OUTPUT TO 8273 VIA TXDACK
        INX     H       ; (*) BUMP DATA POINTER
        SHLD    TXPNT   ; (*) RESTORE POINTER
        JMP     TXI1    ; (*) RETURN AFTER RESTORE
```

Figure 22. Interrupt-Driven Result Handlers with Non-DMA Data Transfers

```
;FUNCTION: POLOP
;INPUTS: NONE
;OUTPUTS: C=0 (NO STATUS), =1 (RX COMPLETION),
;    =2 (TX COMPLETION), =3 (BOTH)
;CALLS: TXI, RXI
;DESTROYS: B,C
;DESCRIPTION: POLOP IS CALLED TO POLL THE 8273 FOR
;DATA TRANSFERS AND COMPLETION RESULTS.  THE
;ROUTINES TXI AND RXI ARE USED FOR THE ACTUAL
;TRANSFERS AND BUFFER WORK.  POLOP RETURNS
;THE STATUS OF THEIR ACTION.
;
POLOP:  PUSH    PSW     ;SAVE PSW
        MVI     C,00H   ;CLEAR C
POLOP1: IN      STAT73  ;READ 8273 STATUS
        ANI     INT     ;ARE TXINT OR RXINT SET?
        JZ      PEXIT   ;NO, EXIT
        IN      STAT73  ;READ 8273 STATUS
        ANI     RXINT   ;TEST RX INT
        JNZ     RXIC    ;YES, GO SERVICE RX
        CALL    TXI     ;MUST BE TX, GO SERVICE IT
        LDA     TXFLAG  ;GET TX FLAG
        CPI     01H     ;WAS IT A COMPLETION? (01)
        JNZ     PEXIT   ;NO, SO JUST EXIT
        INR     C       ;YES, UPDATE C
        INR     C
        JMP     POLOP1  ;TRY AGAIN
;
RXIC:   CALL    RXI     ;GO SERVICE RX
        LDA     RXFLAG  ;GET RX FLAG
        CPI     01H     ;WAS IT A COMPLETION? (01)
        JNZ     PEXIT   ;NO, SO JUST EXIT
        INR     C       ;YES, UPDATE C
        JMP     POLOP1  ;TRY AGAIN
;
PEXIT:  POP     PSW     ;RESTORE PSW
        RET     ;RETURN WITH COMP. STATUS IN C
```

Figure 23. Polling Result Handler

8273 COMMAND DESCRIPTION

In this section, each command is discussed in detail. In order to shorten the notation, please refer to the command key in Table 1. The 8273 utilizes five different command types: Initialization/Configuration, Receive, Transmit, Reset, and Modem Control.

Initialization/Configuration Commands

The Initialization/Configuration commands manipulate registers internal to the 8273 that define the various operating modes. These commands either set or reset specified bits in the registers depending on the type of command. One parameter is required. Set commands perform a logical OR operation of the parameter (mask) and the internal register. This mask contains 1s where register bits are to be set. A 0 in the mask causes no change in the corresponding register bit. Reset commands perform a logical AND operation of the parameter (mask) and the internal register, i.e., the mask is 0 to reset a register bit and a 1 to cause no change. Before presenting the commands, the register bit definitions are discussed.

TABLE 1. COMMAND SUMMARY KEY

B_0, B_1	—	LSB AND MSB OF RECEIVE BUFFER LENGTH
R_0, R_1	—	LSB AND MSB OF RECEIVED FRAME LENGTH
L_0, L_1	—	LSB AND MSB OF TRANSMIT FRAME LENGTH
A_1, A_2	—	MATCH ADDRESSES FOR SELECTIVE RECEIVE
RIC	—	RECEIVER INTERRUPT RESULT CODE
TIC	—	TRANSMITTER INTERRUPT RESULT CODE
A	—	ADDRESS FIELD OF RECEIVED FRAME
C	—	CONTROL FIELD OF RECEIVED FRAME

Operating Mode Register (Figure 24)

D_7-D_6: *Not Used* — These bits must not be manipulated by any command; i.e., D_7-D_6 must be 0 for the Set command and 1 for the Reset command.

D_5: *HDLC Abort* — When this bit is set, the 8273 will interrupt when 7 1s (HDLC Abort) are received by an active receiver. When reset, an SDLC Abort (8 1s) will cause an interrupt.

D_4: *EOP Interrupt* — Reception of an EOP character (0 followed by 7 1s) will cause the 8273 to interrupt the CPU when this bit is set. Loop controller stations use this mode as a signal that a polling frame has completed the loop. No EOP interrupt is generated when this bit is reset.

D_3: *Early Tx Interrupt* — This bit specifies when the transmitter should generate an end of frame interrupt. If this bit is set, an interrupt is generated when the last data character has been passed to the 8273. If the user software issues another transmit command within two byte times, the final flag interrupt does not occur and the new frame is transmitted with only one flag of separation. If this restriction is not met, more than one flag will separate the frames and a frame complete interrupt is generated after the closing flag. If the bit is reset, only the frame complete interrupt occurs. This bit, when set, allows a single flag to separate consecutive frames.

D_2: *Buffered Address and Control* — When set, the address and control fields of received frames are buffered in the 8273 and passed to the CPU as results after a received frame interrupt (they are not transferred to memory with the information field). On transmit, the A and C fields are passed to the 8273 as parameters. This mode simplifies buffer management. When this bit is reset, the A and C fields are passed to and from memory as the first two data transfers.

D_1: *Preframe Sync* — When set, the 8273 prefaces each transmitted frame with two characters before the opening flag. These two characters provide 16 transitions to allow synchronization of the opposing receiver. To guarantee 16 transitions, the two characters are 55H-55H for non-NRZI mode (see Serial I/O Register description) or 00H-00H for NRZI mode. When reset, no preframe characters are transmitted.

D_0: *Flag Stream* — When set, the transmitter will start sending flag characters as soon as it is idle; i.e., immediately if idle when the command is issued or after a transmission if the transmitter is active when this bit is set. When reset, the transmitter starts sending Idle characters on the next character boundary if idle already, or at the end of a transmission if active.

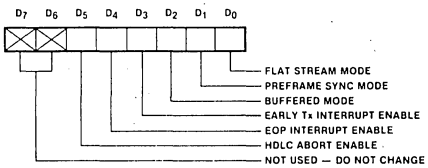


Figure 24. Operating Mode Register

Serial I/O Mode Register (Figure 25)

D_7-D_3: *Not Used* — These bits must be 0 for the Set command and 1 for the Reset command.

D_2: *Data Loopback*— When set, transmitted data (TxD) is internally routed to the receive data circuitry. When reset, TxD and RxD are independent.

D_1: *Clock Loopback* — When set, $\overline{TxC}$ is internally routed to $\overline{RxC}$. When reset, the clocks are independent.

D_0: *NRZI (Non-Return to Zero Inverted)* — When set, the 8273 assumes the received data is NRZI encoded, and NRZI encodes the transmitted data. When reset, the received and transmitted data are treated as a normal positive logic bit stream.

Data Transfer Mode Register (Figure 26)

D_7-D_1: *Not Used* — These bits must be 0 for the Set command and 1 for the Reset command.

D_0: *Interrupt Data Transfer* — When set, the 8273 will interrupt the CPU when data transfers are required (the corresponding IRA Status register bit will be 0 to signify a data transfer interrupt rather than a Result phase interrupt). When reset, 8273 data transfers are performed through DMA requests on the DRQ pins without interrupting the CPU.

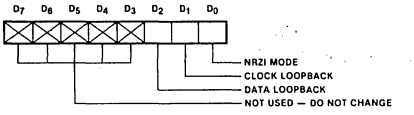


Figure 25. Serial I/O Mode Register

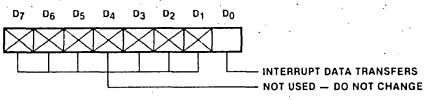


Figure 26. Data Transfer Mode Register

One Bit Delay Register (Figure 27)

D_7: *One Bit Delay* — When set, the 8273 retransmits the received data stream one bit delayed. This mode is entered and exited at a received character boundary. When reset, the transmitted and received data are independent. This mode is utilized for loop operation and is discussed in a later section.

D_6-D_0: *Not Used* — These bits must be 0 for the Set command and 1 for the Reset command.

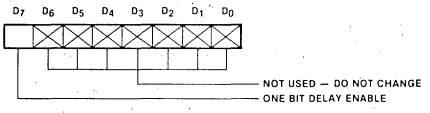


Figure 27. One Bit Delay Mode Register

Figure 28 shows the Set and Reset commands associated with the above registers. The mask which sets or resets the desired bits is treated as a single parameter. These commands do not interrupt nor provide results during the Result phase. After reset, the 8273 defaults to all of these bits reset.

REGISTER	COMMAND	HEX CODE	PARAMETER
ONE BIT DELAY MODE	SET	A4	SET MASK
	RESET	64	RESET MASK
DATA TRANSFER MODE	SET	97	SET MASK
	RESET	57	RESET MASK
OPERATING MODE	SET	91	SET MASK
	RESET	51	RESET MASK
SERIAL I/O MODE	SET	A0	SET MASK
	RESET	60	RESET MASK

Figure 28. Initialization/Configuration Command Summary

Receive Commands

The 8273 supports three receive commands plus a receiver disable function.

General Receive

When commanded to General Receive, the 8273 passes all frames either to memory (DMA mode) or to the CPU (non-DMA mode) regardless of the contents of the frame's address field. This command is used for primary and loop controller stations. Two parameters are required: B_0 and B_1. These parameters are the LSB and MSB of the receiver buffer size. Giving the 8273 this extra information alleviates the CPU of the burden of checking for buffer overflow. The 8273 will interrupt the CPU if the received frame attempts to overfill the allotted buffer space.

Selective Receive

In Selective Receive, two additional parameters besides B_0 and B_1 are required: A_1 and A_2. These parameters are two address match bytes. When commanded to Selective Receive, the 8273 passes to memory or the CPU only those frames having an address field matching either A_1 or A_2. This command is usually used for secondary stations with A_1 being the secondary address and A_2 is the "All Parties" address. If only one match byte is needed, A_1 and A_2 should be equal. As in General Receive, the 8273 counts the incoming data bytes and interrupts the CPU if B_0, B_1 is exceeded.

Selective Loop Receive

This command is very similar in operation to Selective Receive except that One Bit Delay mode must be set and that the loop is captured by placing transmitter in Flag Stream mode automatically after an EOP character is detected following a selectively received frame. The details of using the 8273 in loop configurations is discussed in a later section so please hold questions until then.

The handling of interrupt results is common among the three commands. When a frame is received without error, i.e., the FCS is correct and $\overline{CD}$ (Carrier Detect) was active throughout the frame or no attempt was made to overfill the buffer; the 8273 interrupts the CPU following the closing flag to pass the completion results. These results, in order, are the receiver interrupt result code (RIC), and the byte length of the information field of the received frame (R_0, R_1). If Buffered mode is selected, the address and control fields are passed as two additional results. If Buffered mode is not selected, the address and control fields are passed as the first two data transfers and R_0, R_1 reflect the information field length plus two.

Receive Disable

The receiver may also be disabled using the Receive Disable command. This command terminates any receive operation immediately. No parameters are required and no results are returned.

The details for the Receive command are shown in Figure 29. The interrupt result code key is shown in Figure 30. Some explanation of these result codes is appropriate.

The interrupt result code is the first byte passed to the CPU in the RxI/R register during the Result phase. Bits D_4-D_0 define the cause of the receiver interrupt. Since each result code has specific implications, they are discussed separately below.

COMMAND	HEX CODE	PARAMETERS	RESULTS* RxI/R
GENERAL RECEIVE	C0	B_0, B_1	RIC, R_0, R_1, A, C
SELECTIVE RECEIVE	C1	B_0, B_1, A_1, A_2	RIC, R_0, R_1, A, C
SELECTIVE LOOP RECEIVE	C2	B_0, B_1, A_1, A_2	RIC, R_0, R_1, A, C
DISABLE RECEIVER	C5	NONE	NONE

*A AND C ARE PASSED AS RESULTS ONLY IN BUFFERED MODE.

Figure 29. Receiver Command Summary

RIC D_7-D_0	RECEIVER INTERRUPT RESULT CODE	Rx STATUS AFTER INT
* 00000	A_1 MATCH OR GENERAL RECEIVE	ACTIVE
* 00001	A_2 MATCH	ACTIVE
000 00011	CRC ERROR	ACTIVE
000 00100	ABORT DETECTED	ACTIVE
000 00101	IDLE DETECTED	DISABLED
000 00110	EOP DETECTED	DISABLED
000 00111	FRAME < 32 BITS	ACTIVE
000 01000	DMA OVERRUN	DISABLED
000 01001	MEMORY BUFFER OVERFLOW	DISABLED
000 01010	CARRIER DETECT FAILURE	DISABLED
000 01011	RECEIVER INTERRUPT OVERRUN	DISABLED

*D_7-D_5	PARTIAL BYTE RECEIVED
111	ALL 8 BITS OF LAST BYTE
000	D_0
100	D_1-D_0
010	D_2-D_0
110	D_3-D_0
001	D_4D-$_0$
101	D_5-D_0
011	D_6-D_0

Figure 30. Receiver Interrupt Result Codes (RIC)

The first two result codes result from the error-free reception of a frame. If the frame is received correctly after a General Receive command, the first result is returned. If either Selective Receive command was used (normal or loop), a match with A_1 generates the first result code and a match with A_2 generates the second. In either case, the receiver remains active after the interrupt; however, the internal buffer size counters are not reset. That is, if the receive command indicated 100 bytes were allocated to the receive buffer (B_0, B_1) and an 80-byte frame was received correctly, the maximum next frame size that could be received without recommanding the receiver (resetting B_0 and B_1) is 20 bytes. Thus, it is common practice to recommand the receiver after each frame reception. DMA and/or memory pointers are usually updated at this time. (Note that users who do not wish to take advantage of the 8273's buffer management features may simply use B_0, B_1 = 0FFH for each receive command. Then frames of 65K bytes may be received without buffer overflow errors.)

The third result code is a CRC error. This indicates that a frame was received in the correct format (flags, etc.); however, the received FCS did not check with the internally generated FCS. The frame should be discarded. The receiver remains active. (Do not forget that even though an error condition has been detected, all frame information up until that error has either been transferred to memory or passed to the CPU. This information should be invalidated. This applies to all receiver error conditions.) Note that the FCS, either transmitted or received, is never available to the CPU.

The Abort Detect result occurs whenever the receiver sees either an SDLC (8 1s) or an HDLC (7 1s), depending on the Operating Mode register. However, the intervening Abort character between a closing flag and an Idle does not generate an interrupt. If an Abort character (seen by an active receiver within a frame) is not preceded by a flag and is followed by an Idle, an interrupt will be generated for the Abort, followed by an Idle interrupt one character time later. The Idle Detect result occurs whenever 15 consecutive 1s are received. After the Abort Detect interrupt, the receiver remains active. After the Idle Detect interrupt, the receiver is disabled and must be recommanded before further frames may be received.

If the EOP Interrupt bit is set in the Operating Mode register, the EOP Detect result is returned whenever an EOP character is received. The receiver is disabled, so the Idle following the EOP does not generate an Idle Detect interrupt.

The minimum number of bits in a valid frame between the flags is 32. Fewer than 32 bits indicates an error. If Buffered mode is selected, such frames are ignored, i.e., no data transfers or interrupts are generated. In non-Buffered mode, a < 32-bit frame generates an interrupt with the < 32-bit Frame result since data transfers may already have disturbed the 8257 or interrupt handler. The receiver remains active.

The DMA Overrun result results from the $\overline{DMA}$ controller being too slow in extracting data from the 8273, i.e., the $\overline{RxDACK}$ signal is not returned before the next received byte is ready for transfer. The receiver is disabled if this error condition occurs.

The Memory Buffer Overflow result occurs when the number of received bytes exceeds the receiver buffer length supplied by the B_0 and B_1 parameters in the receive command. The receiver is disabled.

The Carrier Detect Failure result occurs when the $\overline{CD}$ pin goes high (inactive) during reception of a frame. The $\overline{CD}$ pin is used to qualify reception and must be active by the time the address field starts to be received. If $\overline{CD}$ is lost during the frame, a $\overline{CD}$ Failure interrupt is generated and the receiver is disabled. No interrupt is generated if $\overline{CD}$ goes inactive between frames.

If a condition occurs requiring an interrupt be generated before the CPU has finished reading the previous interrupt results, the second interrupt is generated after the current Result phase is complete (the RxINT pin and status bit go low then high). However, the interrupt result for this second interrupt will be a Receive Interrupt Overrun. The actual cause of the second interrupt is lost. One case where this may occur is at the end of a received frame where the line goes idle. The 8273 generates a received frame interrupt after the closing flag and then 15-bit times later, generates an Idle Detect interrupt. If the interrupt service routine is slow in reading the first interrupt's results, the internal RxI/R register still contains result information when the Idle Detect interrupt occurs. Rather than wiping out the previous results, the 8273 adds a Receive Interrupt Overrun result as an extra result. If the system's interrupt structure is such that the second interrupt is not acknowledged (interrupts are still disabled from the first interrupt), the Receive Interrupt Overrun result is read as an extra result, after those from the first interrupt. If the second interrupt is serviced, the Receive Interrupt Overrun is returned as a single result. (Note that the INT pins supply the necessary transitions to support a Program-

mable Interrupt Controller such as the Intel 8259. Each interrupt generates a positive-going edge on the appropriate INT pin and the high level is held until the interrupt is completely serviced.) In general, it is possible to have interrupts occurring at one character time intervals. Thus the interrupt handling software must have at least that much response and service time.

The occurrence of Receive Interrupt Overruns is an indication of marginal software design; the system's interrupt response and servicing time is not sufficient for the data rates being attempted. It is advisable to configure the interrupt handling software to simply read the interrupt results, place them into a buffer, and clear the interrupt as quickly as possible. The software can then examine the buffer for new results at its leisure, and take appropriate action. This can easily be accomplished by using a result buffer flag that indicates when new results are available. The interrupt handler sets the flag and the main program resets it once the results are retrieved.

Both SDLC and HDLC allow frames which are of arbitrary length (>32 bits). The 8273 handles this N-bit reception through the high order bits (D_7-D_5) of the result code. These bits code the number of valid received bits in the last received information field byte. This coding is shown in Figure 30. The high order bits of the received partial byte are indeterminate. [The address, control, and information fields are transmitted least significant bit (A_0) first. The FCS is complemented and transmitted most significant bit first.]

Transmit Commands

The 8273 transmitter is supported by three Transmit commands and three corresponding Abort commands.

Transmit Frame

The Transmit Frame command simply transmits a frame. Four parameters are required when Buffered mode is selected and two when it is not. In either case, the first two parameters are the least and the most significant bytes of the desired frame length (L_0, L_1). In Buffered mode, L_0 and L_1 equal the length in bytes of the desired information field, while in the non-Buffered mode, L_0 and L_1 must be specified as the information field length plus two. (L_0 and L_1 specify the number of data transfers to be performed.) In Buffered mode, the address and control fields are presented to the transmitter as the third and fourth parameters respectively. In non-Buffered mode, the A and C fields must be passed as the first two data transfers.

When the Transmit Frame command is issued, the 8273 makes $\overline{RTS}$ (Request-to-Send) active (pin low) if it was not already. It then waits until $\overline{CTS}$ (Clear-to-Send) goes active (pin low) before starting the frame. If the Preframe Sync bit in the Operting Mode register is set, the transmitter prefaces two characters (16 transitions) before the opening flag. If the Flag Stream bit is set in the Operating Mode register, the frame (including Preframe Sync if selected) is started on a flag boundary. Otherwise the frame starts on a character boundary.

At the end of the frame, the transmitter interrupts the CPU (the interrupt results are discussed shortly) and returns to either Idle or Flag Stream, depending on the Flag Stream bit of the Operating Mode register. If $\overline{RTS}$ was active before the transmit command, the 8273 does not change it. If it was inactive, the 8273 will deactivate it within one character time.

Loop Transmit

Loop Transmit is similar to Frame Transmit (the parameter definition is the same). But since it deals with loop configurations, One Bit Delay mode must be selected.

If the transmitter is not in Flag Stream mode when this command is issued, the transmitter waits until after a received EOP character has been converted to a flag (this is done automatically) before transmitting. (The one bit delay is, of course, suspended during transmit.) If the transmitter is already in Flag Stream mode as a result of a selectively received frame during a Selective Loop Receive command, transmission will begin at the next flag boundary for Buffered mode or at the third flag boundary for non-Buffered mode. This discrepancy is to allow time for enough data transfers to occur to fill up the internal transmit buffer. At the end of a Loop Transmit, the One Bit Delay mode is re-entered and the flag stream mode is reset. More detailed loop operation is covered later.

Transmit Transparent

The Transmit Transparent command enables the 8273 to transmit a block of raw data. This data is without SDLC protocol, i.e., no zero bit insertion, flags, or FCS. Thus it is possible to construct and transmit a Bi-Sync message for front-end processor switching or to construct and transmit an SDLC message with incorrect FCS for diagnostic purposes. Only the L_0 and L_1 parameters are used since there are not fields in this mode. (The 8273 does not support a Receive Transparent command.)

Abort Commands

Each of the above transmit commands has an associated Abort command. The Abort Frame Transmit command causes the transmitter to send eight contiguous ones (no zero bit insertion) immediately and then revert to either idle or flag streaming based on the Flag Stream bit. (The 8 1s as an Abort character is compatible with both SDLC and HDLC.)

For Loop Transmit, the Abort Loop Transmit command causes the transmitter to send one flag and then revert to one bit delay. Loop protocol depends upon FCS errors to detect aborted frames.

The Abort Transmit Transparent simply causes the transmitter to revert to either idles or flags as a function of the Flag Stream mode specified.

The Abort commands require no parameters, however, they do generate an interrupt and return a result when complete.

A summary of the Transmit commands is shown in Figure 31. Figure 32 shows the various transmit interrupt result codes. As in the receiver operation, the transmitter generates interrupts based on either good

completion of an operation or an error condition to start the Result phase.

The Early Transmit Interrupt result occurs after the last data transfer to the 8273 if the Early Transmit Interrupt bit is set in the Operating Mode register. If the 8273 is commanded to transmit again within two character times, a single flag will separate the frames. (Buffered mode must be used for a single flag to separate the frames. If non-Buffered mode is selected, three flags will separate the frames.) If this time constraint is not met, another interrupt is generated and multiple flags or idles will separate the frames. The second interrupt is the normal Frame Transmit Complete interrupt. The Frame Transmit Complete result occurs at the closing flag to signify a good completion.

The DMA Underrun result is analogous to the DMA Overrun result in the receiver. Since SDLC does not support intraframe time fill, if the DMA controller or CPU does not supply the data in time, the frame must be aborted. The action taken by the transmitter on this error is automatic. It aborts the frame just as if an Abort command had been issued.

Clear-to-Send Error result is generated if $\overline{CTS}$ goes inactive during a frame transmission. The frame is aborted as above.

The Abort Complete result is self-explanatory. Please note however that no Abort Complete interrupt is generated when an automatic abort occurs. The next command type consists of only one command.

COMMAND	HEX CODE	PARAMETERS*	RESULTS TxI/R
TRANSMIT FRAME	C8	L_0, L_1, A, C	TIC
ABORT	CC	NONE	TIC
LOOP TRANSMIT	CA	L_0, L_1, A, C	TIC
ABORT	CE	NONE	TIC
TRANSMIT TRANSPARENT	C0	L_0, L_1	TIC
ABORT	CD	NONE	TIC

*A AND C ARE PASSED AS PARAMETERS IN BUFFERED MODE ONLY.

Figure 31. Transmitter Command Summary

TIC D_7-D_0	TRANSMITTER INTERRUPT RESULT CODE	Tx STATUS AFTER INT
000 01100	EARLY Tx INTERRUPT	ACTIVE
000 01101	FRAME Tx COMPLETE	IDLE OR FLAGS
000 01110	DMA UNDERRUN	ABORT
000 01111	CLEAR TO SEND ERROR	ABORT
000 10000	ABORT COMPLETE	IDLE OR FLAGS

Figure 32. Transmitter Interrupt Result Codes

Reset Command

The Reset command provides a software reset function for the 8273. It is a special case and does not utilize the normal command interface. The reset facility is provided in the Test Mode register. The 8273 is reset by simply outputting a 01H followed by a 00H to the Test Mode register. Writing the 01 followed by the 00 mimicks the action required by the hardware reset. Since the 8273 requires time to process the reset internally, at least 10 cycles of the ØCLK clock must occur between the writing of the 01 and the 00. The action taken is the same as if a hardware reset is performed, namely:

1. The modem control outputs are forced high inactive).
2. The 8273 Status register is cleared.
3. Any commands in progress cease.
4. The 8273 enters an idle state until the next command is issued.

Modem Control Commands

The modem control ports were discussed earlier in the Hardware section. The commands used to manipulate these ports are shown in Figure 33. The Read Port A and Read Port B commands are immediate. The bit definition for the returned byte is shown in Figures 13 and 14. Do not forget that the returned value represents the logical condition of the pin, i.e., pin active (low) = bit set.

PORT	COMMAND	HEX CODE	PARAMETER	REG RESULT
A INPUT	READ	22	NONE	PORT VALUE
B OUTPUT	READ	23	NONE	PORT VALUE
	SET	A3	SET MASK	NONE
	RESET	63	RESET MASK	NONE

Figure 33. Modem Control Command Summary

The Set and Reset Port B commands are similar to the Initialization commands in that they use a mask parameter which defines the bits to be changed. Set Port B utilizes a logical OR mask and Reset Port B uses a logical AND mask. Setting a bit makes the pin active (low). Resetting the bit deactivates the pin (high).

To help clarify the numerous timing relationships that occur and their consequences, Figures 34 and 35 are provided as an illustration of several typical sequences. It is suggested that the reader go over these diagrams and re-read the appropriate part of the previous sections if necessary.

HLDC CONSIDERATIONS

The 8273 supports HDLC as well as SDLC. Let's discuss how the 8273 handles the three basic HDLC/SDLC differences: extended addressing, extended control, and the 7 1s Abort character.

Recalling Figure 4A, HDLC supports an address field of indefinite length. The actual amount of extension used is determined by the least significant bit of the characters immediately following the opening flag. If the LSB is 0, more address field bytes follow. If the LSB is 1, this byte is the final address field byte. Software must be used to determine this extension.

If non-Buffered mode is used, the A, C, and I fields are in memory. The software must examine the initial characters to find the extent of the address field. If Buffered mode is used, the characters corresponding to the SDLC A and C fields are transferred to the CPU as interrupt results. Buffered mode assumes the two characters following the opening flag are to be transferred as interrupt results regardless of content or meaning. (The 8273

does not know whether it is being used in an SDLC or an HDLC environment.) In SDLC, these characters are necessarily the A and C field bytes, however in HDLC, their meaning may change depending on the amount of extension used. The software must recognize this and examine the transferred results as possible address field extensions.

Frames may still be selectively received as is needed for secondary stations. The Selective Receive command is still used. This command qualifies a frame reception on the first byte following the opening flag matching either of the A_1 or A_2 match byte parameters. While this does not allow qualification over the complete range of HDLC addresses, it does perform a qualification on the first address byte. The remaining address field bytes, if any, are then examined via software to completely qualify the frame.

Once the extent of the address field is found, the following bytes form the control field. The same LSB test used for the address field is applied to these bytes to determine the control field extension, up to two bytes maximum. The remaining frame bytes in memory represent the information field.

The Abort character difference is handled in the Operating Mode register. If the HDLC Abort Enable bit is set, the reception of seven contiguous ones by an active receiver will generate an Abort Detect interrupt rather than eight ones. (Note that both the HDLC Abort Enable bit and the EOP Interrupt bit must not be set simultaneously.)

Now let's move on to the SDLC loop configuration discussion.

LOOP CONFIGURATION

Aside from use in the normal data link applications, the 8273 is extremely attractive in loop configuration due to the special frame-level loop commands and the Digital Phase Locked Loop. Toward this end, this section details the hardware and software considerations when using the 8273 in a loop application.

The loop configuration offers a simple, low-cost solution for systems with multiple stations within a small physical location, i.e., retail stores and banks. There are two primary reasons to consider a loop configuration. The interconnect cost is lower for a loop over a multi-point configuration since only one twisted pair or fiber optic cable is used. (The loop configuration does not support the passing of distinct clock signals from station to station.) In addition, loop stations do not need the intelligence of a multi-point station since the loop protocol is simpler. The most difficult aspects of loop station design are clock recovery and implementation of one bit delay (both are handled neatly by the 8273).

Figure 36 illustrates a typical loop configuration with one controller and two down-loop secondaries. Each station must derive its own data timing from the received data stream. Recalling our earlier discussion of the DPLL, notice that $\overline{TxC}$ and $\overline{RxC}$ clocks are provided by the DPLL output. The only clock required in the secondaries is a simple, non-synchronized clock at 32 times the desired baud rate. The controller requires both 32× and 1× clocks. (The 1× is usually implemented by dividing the 32× clock with a 5-bit divider. However, there is no synchronism requirement between these clocks so any convenient implementation may be used.)

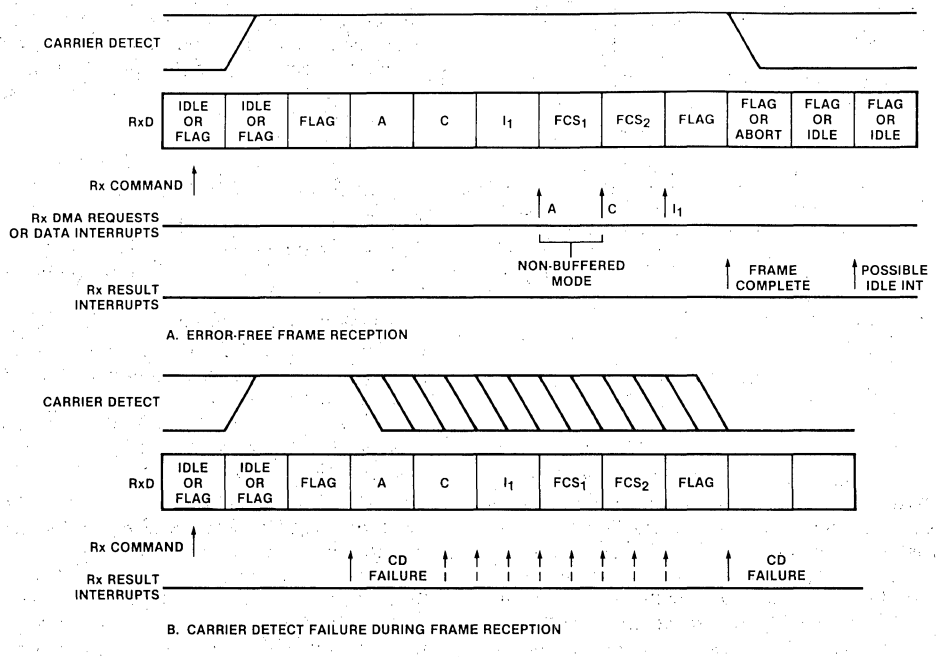


Figure 34. Sample Receiver Timing Diagrams

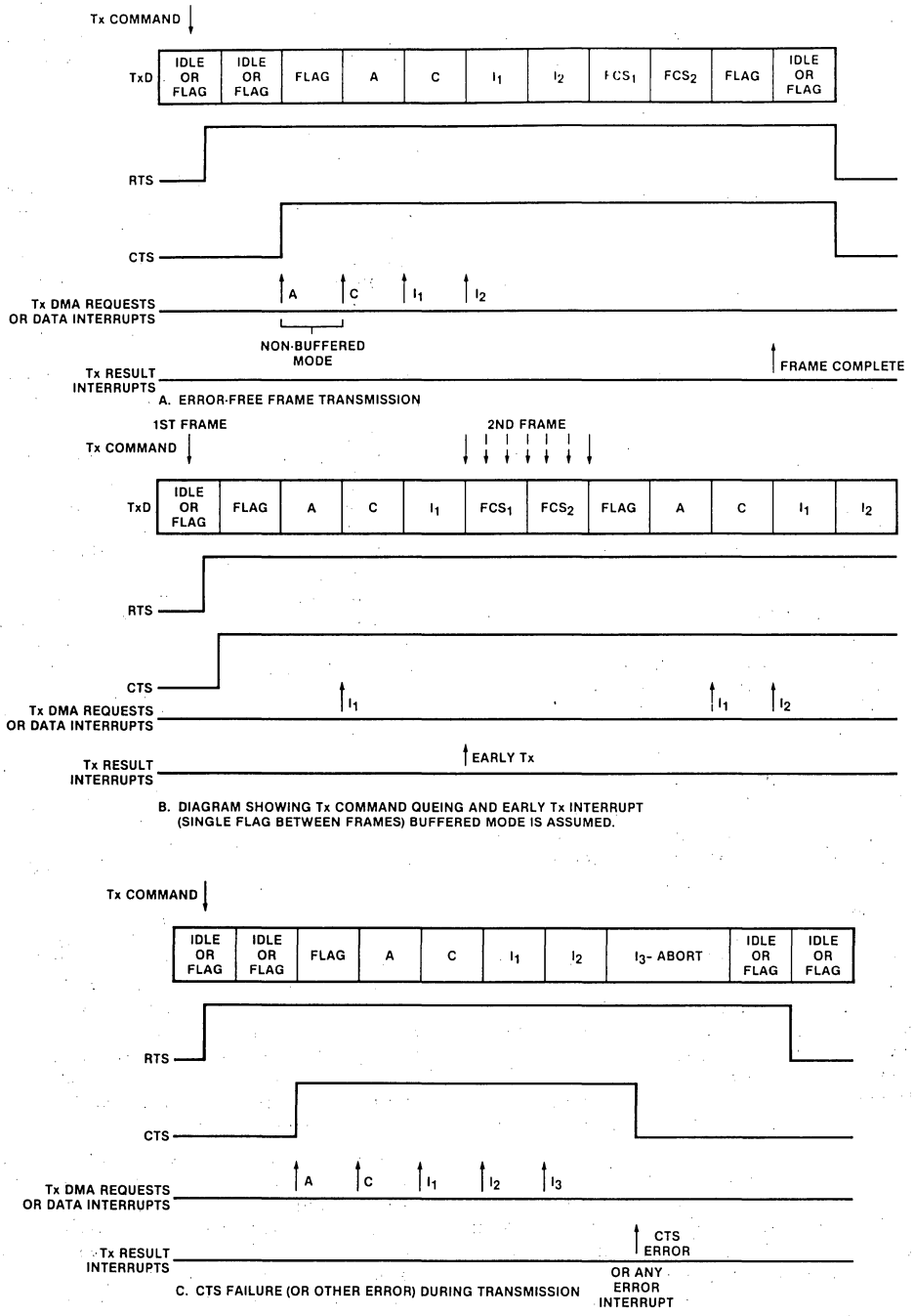


Figure 35. Sample Transmitter Timing Diagrams

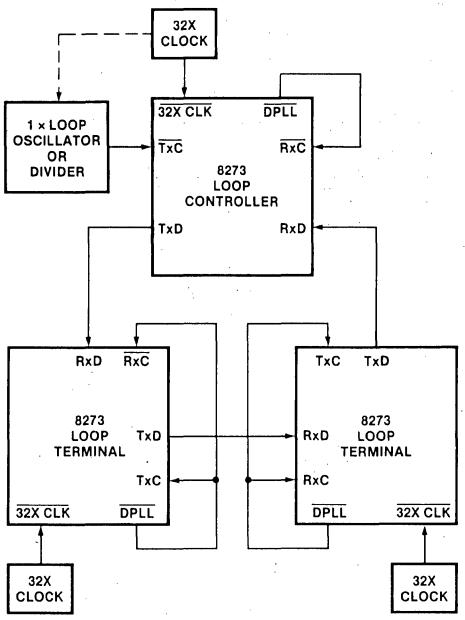


Figure 36. SDLC Loop Application

A quick review of loop protocol is appropriate. All communication on the loop is controlled by the loop controller. When the controller wishes to allow the secondaries to transmit, it sends a polling frame (the control field contains a poll code) followed by an EOP (End-of-Poll) character. The secondaries use the EOP character to capture the loop and insert a response frame as will be discussed shortly.

The secondaries normally operate in the repeater mode, retransmitting received data with one bit time of delay. All received frames are repeated. The secondary uses the one bit time of delay to capture the loop.

When the loop is idle (no frames), the controller transmits continuous flag characters. This keeps transitions on the loop for the sake of down-loop phase locked loops. When the controller has a non-polling frame to transmit, it simply transmits the frame and continues to send flags. The non-polling frame is then repeated around the loop and the controller receives it to signify a complete traversal of the loop. At the particular secondary addressed by the frame, the data is transferred to memory while being repeated. Other secondaries simply repeat it.

If the controller wants to poll the secondaries, it transmits a polling frame followed by all 1s (no zero bit insertion). The final zero of the closing frame plus the first seven 1s form an EOP. While repeating, the secondaries monitor their incoming line for an EOP. When an EOP is received, the secondary checks if it has any response for the controller. If not, it simply continues repeating. If the secondary has a response, it changes the seventh EOP one into a zero (the one bit time of delay allows time for this) and repeats it, forming a flag for the down-loop stations. After this flag is transmitted, the secondary terminates its repeater function and inserts its response frame (with multiple preceding flags if necessary). After the closing flag of the response, the secondary re-enters its repeater function, repeating the up-loop controller 1s. Notice that the final zero of the response's closing flag plus the repeated 1s from the controller form a new EOP for the next down-loop secondary. This new EOP allows the next secondary to insert a response if it desires. This gives each secondary a chance to respond.

Back at the controller, after the polling frame has been transmitted and the continuous 1s started, the controller waits until it receives an EOP. Receiving an EOP signifies to the controller that the original frame has propagated around the loop followed by any responses inserted by the secondaries. At this point, the controller may either send flags to idle the loop or transmit the next frame. Let's assume that the loop is implemented completely with the 8273s and describe the command flows for a typical controller and secondary.

The loop controller is initialized with commands which specify that the NRZI, Preframe Sync, Flag Stream, and EOP Interrupt modes are set. Thus, the controller encodes and decodes all data using NRZI format. Preframe Sync mode specifies that all transmitted frames be prefaced with 16 line transitions. This ensures that the minimum of 12 transitions needed by the DPLLs to lock after an all 1s line have occurred by the time the secondary sees a frame's opening flag. Setting the Flag Stream mode starts the transmitter sending flags which idles the loop. And the EOP Interrupt mode specifies that the controller processor will be interrupted whenever the active receiver sees an EOP, indicating the completion of a poll cycle.

When the controller wishes to transmit a non-polling frame, it simply executes a Frame Transmit command. Since the Flag Stream mode is set, no EOP is formed after the closing flag. When a polling frame is to be transmitted, a General Receive command is executed first. This enables the receiver and allows reception of all incoming frames; namely, the original polling frame plus any response frames inserted by the secondaries. After the General Receive command, the frame is transmitted with a Frame Transmit command. When the frame is complete, a transmitter interrupt is generated. The loop controller processor uses this interrupt to reset Flag Stream mode. This causes the transmitter to start sending all 1s. An EOP is formed by the last flag and the first 7 1s. This completes the loop controller transmit sequence.

At any time following the start of the polling frame transmission the loop controller receiver will start receiving frames. (The exact time difference depends, of course, on the number of down-loop secondaries due to each inserting one bit time of delay.) The first received frame is simply the original polling frame. However, any additional frames are those inserted by the secondaries. The loop controller processor knows all frames have been received when it sees an EOP Interrupt. This interrupt is generated by the 8273 since the EOP Interrupt mode was set during initialization. At this point, the transmitter may be commanded either to enter Flag

Stream mode, idling the loop, or to transmit the next frame. A flowchart of the above sequence is shown in Figure 37.

The secondaries are initialized with the NRZI and One Bit Delay modes set. This puts the 8273 into the repeater mode with the transmitter repeating the received data with one bit time of delay. Since a loop station cannot transmit until it sees and EOP character, any transmit command is queued until an EOP is received. Thus whenever the secondary wishes to transmit a response, a Loop Transmit command is issued. The 8273 then waits until it receives an EOP. At this point, the receiver changes the EOP into a flag, repeats it, resets One Bit Delay mode stopping the repeater function, and sets the transmitter into Flag Stream mode. This captures the loop. The transmitter now inserts its message. At the closing flag, Flag Stream mode is reset, and One Bit Delay mode is set, returning the 8273 to repeater function and forming an EOP for the next down-loop station. These actions happen automatically after a Loop Transmit command is issued.

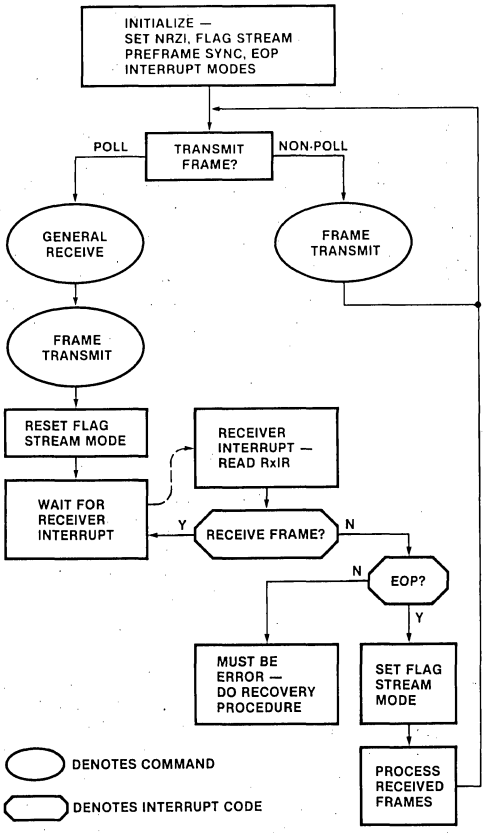


Figure 37. Loop Controller Flowchart

When the secondary wants its receiver enabled, a Selective Loop Receive command is issued. The receiver then looks for a frame having a match in the Address field. Once such a frame is received, repeated, and transferred to memory, the secondary's processor is interrupted with the appropriate Match interrupt result and the 8273 continues with the repeater function until an EOP is received, at which point the loop is captured as above. The processor should use the interrupt to determine if it has a message for the controller. If it does, it simply issues a Loop Transmit command and things progress as above. If the processor has no message, the software must reset the Flag Stream mode bit in the Operating Mode register. This will inhibit the 8273 from capturing the loop at the EOP. (The match frame and the EOP may be separated in time by several frames depending on how many up-loop stations inserted messages of their own.) If the timing is such that the receiver has already captured the loop when the Flag Stream mode bit is reset, the mode is exited on a flag boundary and the frame just appears to have extra closing flags before the EOP. Notice that the 8273 handles the queuing of the transmit commands and the setting and resetting of the mode bits automatically. Figure 38 illustrates the major points of the secondary command sequence.

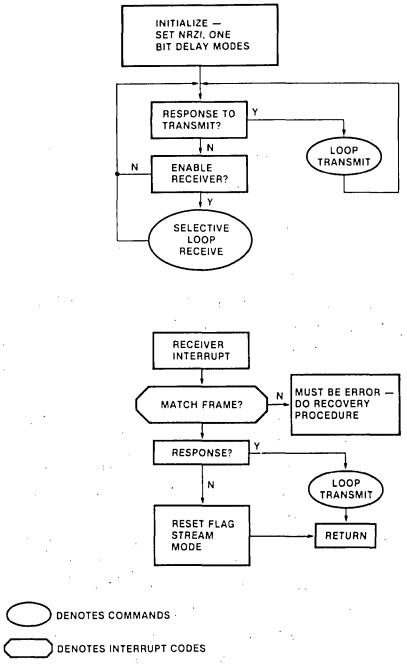


Figure 38. Loop Secondary Flowchart

When an off-line secondary wishes to come on-line, it must do so in a manner which does not disturb data on the loop. Figure 39 shows a typical hardware interface. The line labeled Port could be one of the 8273 Port B outputs and is assumed to be high (1) initially. Thus up-loop data is simply passed down-loop with no delay; however, the receiver may still monitor data on the loop. To come on-line, the secondary is initialized with only the EOP Interrupt mode set. The up-loop data is then monitored until an EOP occurs. At this point, the secondary's CPU is interrupted with an EOP interrupt. This signals the CPU to set One Bit Delay mode in the 8273 and then to set Port low (active). These actions switch the secondary's one bit delay into the loop. Since after the EOP only 1s are traversing the loop, no loop disturbance occurs. The secondary now waits for the next EOP, captures the loop, and inserts a "new on-line" message. This signals the controller that a new secondary exists and must be acknowledged. After the secondary receives its acknowledgement, the normal command flow is used.

It is hopefully evident from the above discussion that the 8273 offers a very simple and easy to implement solution for designing loop stations whether they are controllers or down-loop secondaries.

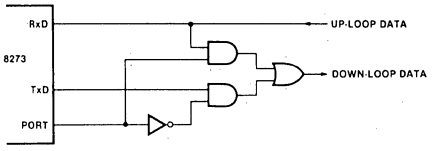


Figure 39. Loop Interface

APPLICATION EXAMPLE

This section describes the hardware and software of the 8273/8085 system used to verify the 8273 implementation of SDLC on an actual IBM SDLC Link. This IBM link was gratefully volunteered by Raytheon Data Systems in Norwood, Mass. and I wish to thank them for their generous cooperation. The IBM system consisted of a 370 Mainframe, a 3705 Communications Processor, and a 3271 Terminal Controller. A Comlink II Modem supplied the modem interface and all communications took place at 4800 baud. In addition to observing correct responses, a Spectron D601B Datascope was used to verify the data exchanges. A block diagram of the system is shown in Figure 40. The actual verification was accomplished by the 8273 system receiving and responding to polls from the 3705. This method was used on both point-to-point and multi-point configurations. No attempt was made to implement any higher protocol software over that of the poll and poll responses since such software would not affect the verification of the 8273 implementation. As testimony to the ease of use of the 8273, the system worked on the first try.

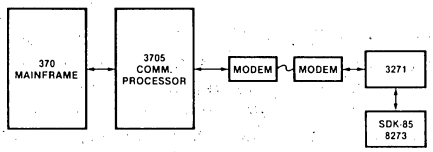


Figure 40. Raytheon Block Diagram

An SDK-85 (System Design Kit) was used as the core 8085 system. This system provides up to 4K bytes of ROM/EPROM, 512 bytes of RAM, 76 I/O pins, plus two timers as provided in two 8755 Combination EPROM/I/O devices and two 8155 Combination RAM/I/O/Timer devices. In addition, 5 interrupt inputs are supplied on the 8085. The address, data, and control buses are buffered by the 8212 and 8216 latches and bidirectional bus drivers. Although it was not used in this application, an 8279 Display Driver/Keyboard Encoder is included to interface the on-board display and keyboard. A block diagram of the SDK-85 is shown in Figure 41. The 8273 and associated circuitry was constructed on the ample wire-wrap area provided for the user.

The example 8273/8085 system is interrupt-driven and uses DMA for all data transfers supervised by an 8257 DMA Controller. A 2400 baud asynchronous line, implemented with an 8251A USART, provides communication between the software and the user. 8253 Programmable Interval Timer is used to supply the baud rate clocks for the 8251A and 8273. (The 8273 baud rate clocks were used only during initial system debug. In actual operation, the modem supplied these clocks via the RS-232 interface.) Two 2142 1K × 4 RAMs provided 512 bytes of transmitter and 512 bytes of receiver buffer memory. (Command and result buffers, plus miscellaneous variables are stored in the 8155s.) The RS-232 interface utilized MC1488 and MC1489 RS-232 drivers and receivers. The schematic of the system is shown in Figure 42.

One detail to note is the DMA and interrupt structure of the transmit and receive channels. In both cases, the receiver is always given the higher priority (8257 DMA channel 0 has priority over the remaining channels and the 8085 RST 7.5 interrupt input has priority over the RST 6.5 input.) Although the choice is arbitrary, this technique minimizes the chance that received data could be lost due to other processor or DMA commitments.

Also note that only one 8205 Decoder is used for both the peripherals' and the memorys' Chip Selects. This was done to eliminate separate memory and I/O decoders since it was known beforehand that neither address space would be completely filled.

The 4 MHz crystal and 8224 Clock Generator were used only to verify that the 8273 operates correctly at that maximum spec speed. In a normal system, the 3.072 MHz clock from the 8085 would be sufficient. (This fact was verified during initial checkout.)

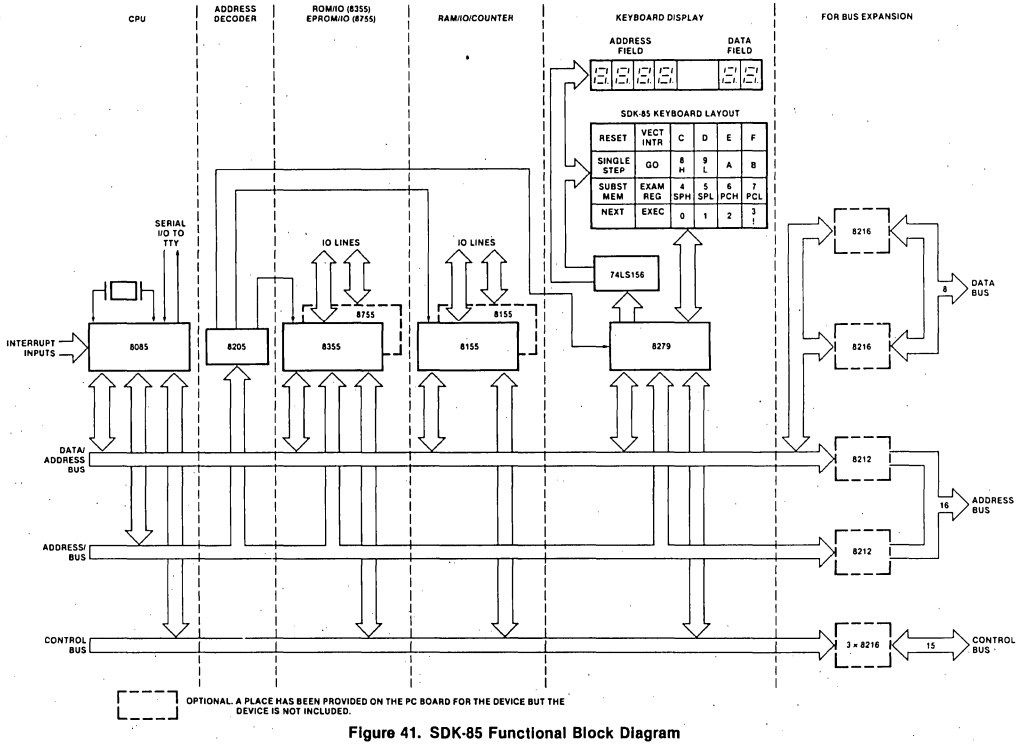


Figure 41. SDK-85 Functional Block Diagram

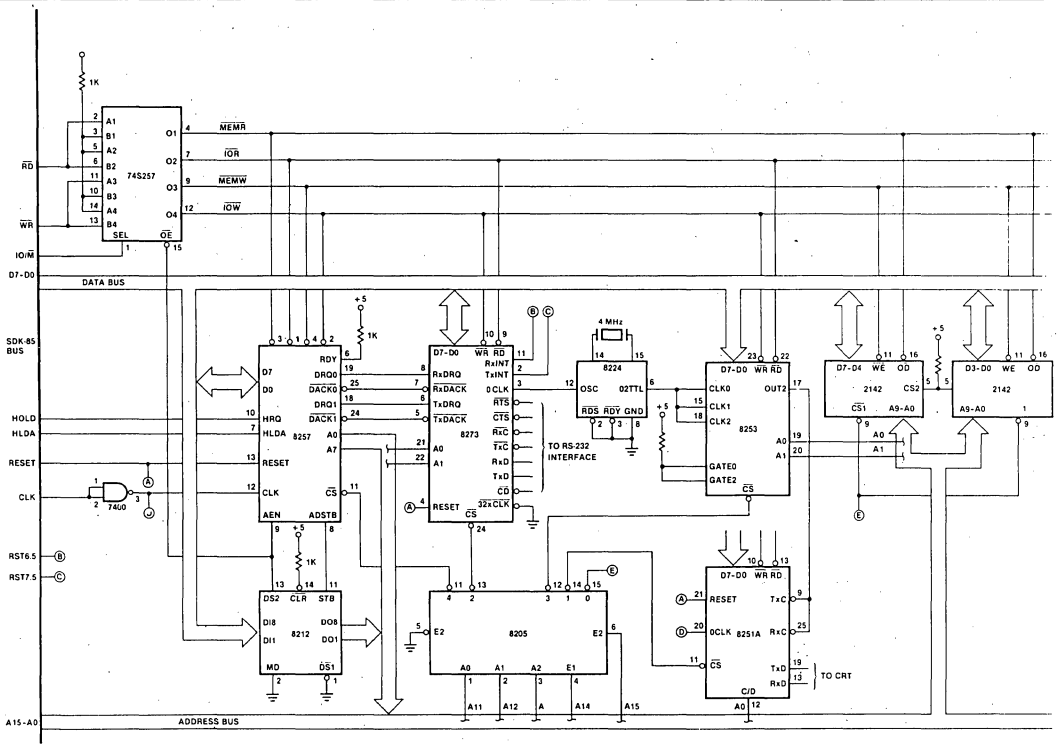


Figure 42. 8273/SDK-85 System

The software consists of the normal monitor program supplied with the SDK-85 and a program to input commands to the 8273 and to display results. The SDK-85 monitor allows the user to read and write on-board RAM, start execution at any memory location, to single-step through a program, and to examine any of the 8085's internal registers. The monitor drives either the on-board keyboard/LED display or a serial TTY interface. This monitor was modified slightly in order to use the 8251A with a 2400 baud CRT as opposed to the 110 baud normally used. The 8273 program implements monitor-like user interface. 8273 commands are entered by a two-character code followed by any parameters required by that command. When 8273 interrupts occur, the source of the interrupt is displayed along with any results associated with it. To gain a flavor of how the user/program interface operates, a sample output is shown in Figure 43. The 8273 program prompt character is a "–" and user inputs are underlined.

The "SO 05" implements the Set Operating Mode command with a parameter of 05H. This sets the Buffer and Flag Stream modes. "SS 01" sets the 8273 in NRZI mode using the Set Serial I/O Mode command. The next command specifies General Receiver with a receiver buffer size of 0100H bytes ($B_0 = 00$, $B_1 = 01$). The "TF" command causes the 8273 to transmit a frame containing an address field of C2H and control field of 11H. The information field is 001122. The "TF" command has a special format. The L_0 and L_1 parameters are computed from the number of information field bytes entered.

After the TF command is entered, the 8273 transmits the frame (assuming that the modem protocol is observed). After the closing flag, the 8273 interrupts the 8085. The 8085 reads the interrupt results and places them in a buffer. The software examines this buffer for new results and if new results exist, the source of the interrupt is displayed along with the results.

In this example, the 0DH result indicates a Frame Complete interrupt. There is only one result for a transmitter interrupt, the interrupt's trailing zero results were included to simplify programming.

The next event is a frame reception. The interrupt results are displayed in the order read from the 8273. The E0H indicates a General Receive interrupt with the last byte of the information field received on an 8-bit boundary. The 03 00 (R_0, R_1) results show that there are 3H bytes of information field received. The remaining two results indicate that the received frame had a C2H address field and a 34H control field. The 3 bytes of information field are displayed on the next line.

```
8273 MONITOR V1.2

–  SO 05
–  SS 01
–  GR 00 01
–  TF C2 11 00 11 22
–
TxINT  –  0D 00 00 00 00
–
RxINT  –  E0 03 00 C2 34
FF EE DD
–
```

Figure 43. Sample 8273 Monitor I/O

Figures 44 through 51 show the flowcharts used for the 8273 program development. The actual program listing is included as Appendix A. Figure 44 is the main status poll loop. After all devices are initialized and a prompt character displayed, a loop is entered at LOOPIT. This loop checks for a change of status in the result buffer or if a keyboard character has been received by the 8251 or if a poll frame has been received. If any of these conditions are met, the program branches to the appropriate routine. Otherwise, the loop is traversed again.

The result buffer is implemented as a 255-byte circular buffer with two pointers: CNADR and LDADR. CNADR is the console pointer. It points to the next result to be displayed LDADR is the load pointer. It points to the next empty position in the buffer into which the interrupt handler places the next result. The same buffer is used for both transmitter and receiver results. LOOPIT examines these pointers to detect when CNADR is not equal to LDADR indicating that the buffer contains results which have not been displayed. When this occurs, the program branches to the DISPLY routine.

DISPLY determines the source of the undisplayed results by testing the first result. This first result is necessarily the interrupt result code. If this result is 0CH or greater, the result is from a transmitter interrupt. Otherwise it is from a receiver source. The source of the result code is then displayed on the console along with the next four results from the buffer. If the source was a transmitter interrupt, the routine merely repoints the pointer CNADR and returns to LOOPIT. For a receiver source, the receiver data buffer is displayed in addition to the receiver interrupt results before returning to LOOPIT.

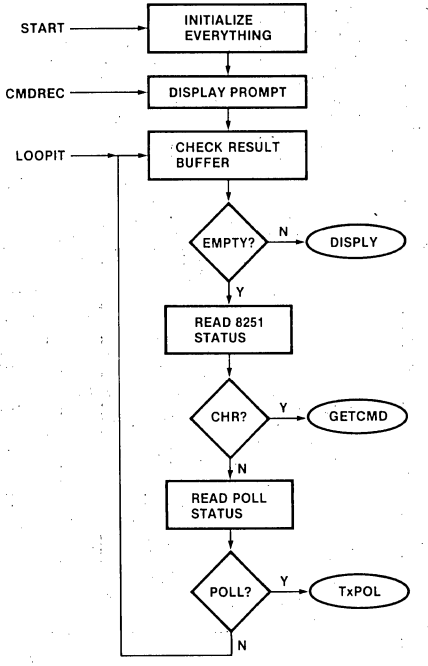


Figure 44. Main Status Poll Loop

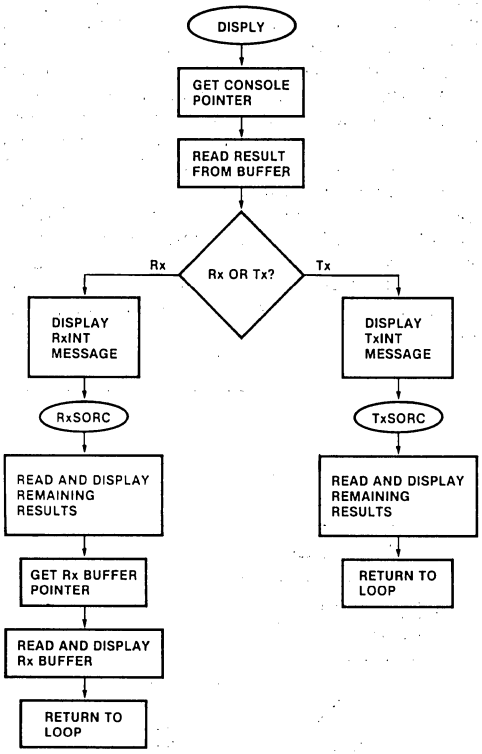


Figure 45. DISPLY Subroutine

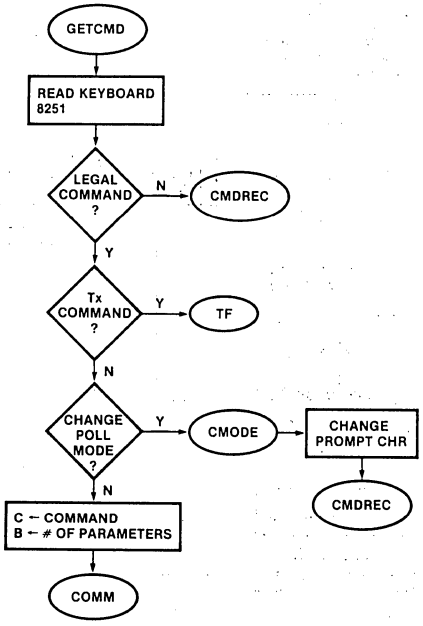


Figure 46. GETCMD Subroutine

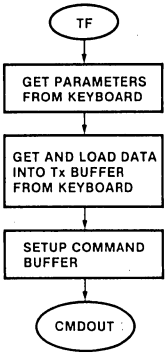


Figure 47. TF Subroutine

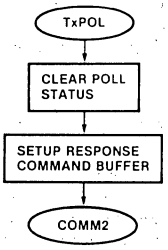


Figure 48. TxPOL Subroutine

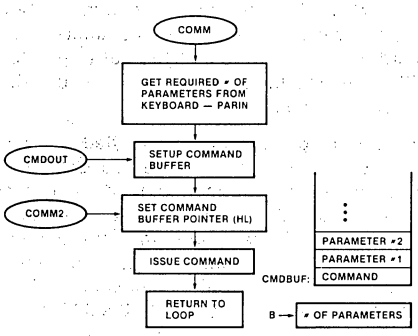


Figure 49. COMM Subroutine with Command Buffer Format

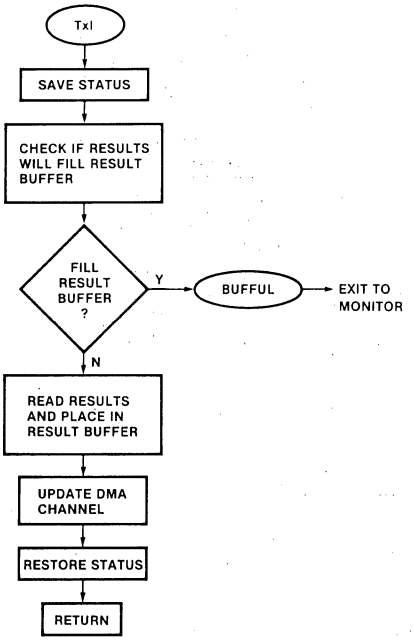


Figure 50. TxI (Transmitter Interrupt) Routine

If the result buffer pointers indicate an empty buffer, the 8251A is polled for a keyboard character. If the 8251 has a character, GETCMD is called. There the character is read and checked if legal. Illegal characters simply cause a reprompt. Legal characters indicate the start of a command input. Most commands are organized as two characters signifying the command action; i.e., GR — General Receive. The software recognizes the two character command code and takes the appropriate action. For non-Transmit type commands, the hex equivalent of the command is placed in the C register and the number of parameters associated with that command is placed in the B register. The program then branches to the COMM routine.

The COMM routine builds the command buffer by reading the required number of parameters from the keyboard and placing them at the buffer pointed at by CMDBUF. The routine at COMM2 then issues this command buffer to the 8273.

If a Transmit type command is specified, the command buffer is set up similarly to the the COMM routine; however, since the information field data is entered from the keyboard, an intermediate routine, TF, is called. TF loads the transmit data buffer pointed at by TxBUF. It counts the number of data bytes entered and loads this number into the command buffer as L_0, L_1. The command is then issued to the 8273 by jumping to CMDOUT.

One command does not directly result in a command being issued to the 8273. This command, Z, operates a software flip-flop which selects whether the software will respond automatically to received polling frames. If the Poll-Response mode is selected, the prompt character is changed to a '+'. If a frame is received which contains a prearranged poll control field, the memory location POLIN is made nonzero by the receiver interrupt handler. LOOPIT examines this location and if it is nonzero, causes a branch to the TxPOL routine. The TxPOL routine clears POLIN, sets a pointer to a special command buffer at CMDBUF1, and issues the command by way of the COMM2 entry in the COMM routine. The special command buffer contains the appropriate response frame for the poll frame received. These actions only occur when the Z command has changed the prompt to a '+'. If the prompt is normal '−', polling frames are displayed as normal frames and no response is transmitted. The Poll-Response mode was used during the IBM tests.

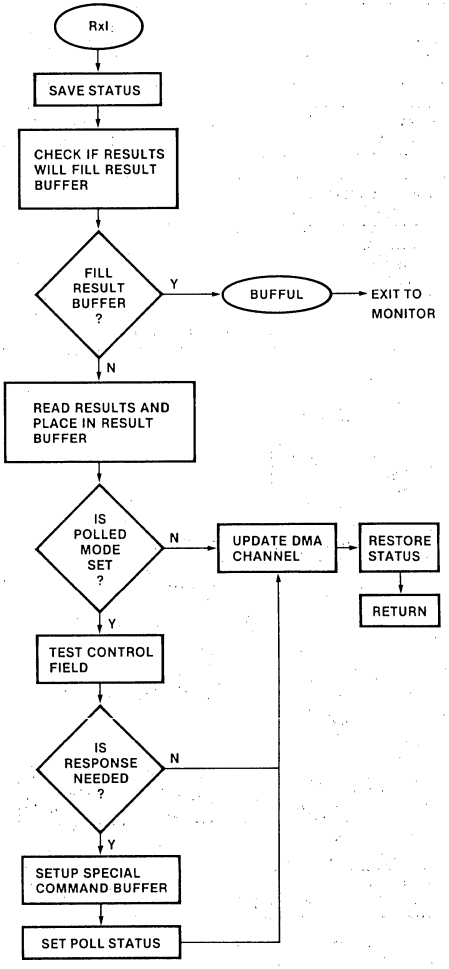


Figure 51. RxI (Recevier Interrupt) Routine

The final two software routines are the transmitter and receiver interrupt handlers. The transmit interrupt handler, TxI, simply saves the registers on the stack and checks if loading the result buffer will fill it. If the result buffer will overfill, the program is exited and control is passed to the SDK-85 monitor. If not, the results are read from the TxI/R register and placed in the result buffer at LDADR. The DMA pointers are then reset, the registers restored, and interrupts enabled. Execution then returns to the pre-interrupt location.

The receiver interrupt handler, RxI, is only slightly more complex. As in TxI, the registers are saved and the possibility of overfilling the result buffer is examined. If the result buffer is not full, the results are read from RxI/R and placed in the buffer. At this point the prompt character is examined to see if the Poll-Response mode is selected. If so, the control field is compared with two possible polling control fields. If there is a match, the special command buffer is loaded and the poll indicator, POLIN, is made nonzero. If no match occurred, no action is taken. Finally, the receiver DMA buffer pointers are reset, the processor status restored, and interrupts are enabled. The RET instruction returns execution to the pre-interrupt location.

This completes the discussion of the 8273/8085 system design.

CONCLUSION

This application note has covered the 8273 in some detail. The simple and low cost loop configuration was explored. And an 8273/8085 system was presented as a sample design illustrating the DMA/interrupt-driven interface. It is hoped that the major features of the 8273, namely the frame-level command structure and the Digital Phase Locked Loop, have been shown to be a valuable asset in an SDLC system design.

APPENDIX A

```
ASM80 :F1:RAYT73.SRC


ISIS-II 8080/8085 MACRO ASSEMBLER, X108        MODULE   PAGE    1


 LOC  OBJ         SEQ        SOURCE STATEMENT

                    1 $NOPAGING MOD85  NOCOND
 0000               2 TRUE    EQU     00H            ;00 FOR RAYTHEON
                    3 ;                              ;FF FOR SELF-TEST
 0000               4 TRUE1   EQU     00H            ;00 FOR NORMAL RESPONSE
                    5 ;                              ;FF FOR LOOP RESPONSE
 0000               6 DEM     EQU     00H            ;00 FOR NO DEMO
                    7 ;                              ;FF FOR DEMO
                    8 ;
                    9 ;
                   10 ;GENERAL 8273 MONITOR WITH RAYTHEON POLL MODE ADDED
                   11 ;
                   17 ;
                   18 ;
                   19 ;COMMAND SUPPORTED ARE: RS - RESET SERIAL I/O MODE
                   20 ;                       SS - SET SERIAL I/O MODE
                   21 ;                       RO - RESET OPERATING MODE
                   22 ;                       SO - SET OPERATING MODE
                   23 ;                       RD - RECEIVER DISABLE
                   24 ;                       GR - GENERAL RECEIVE
                   25 ;                       SR - SELECTIVE RECEIVE
                   26 ;                       TF - TRANSMIT FRAME
                   27 ;                       AF - ABORT FRAME
                   28 ;                       SP - SET PORT B
                   29 ;                       RP - RESET PORT B
                   30 ;                       RB - RESET ONE BIT DELAY (PAR = 7F)
                   31 ;                       SB - SET ONE BIT DELAY (PAR = 80)
                   32 ;                       SL - SELECTIVE LOOP RECEIVE
                   33 ;                       TL - TRANSMIT LOOP
                   34 ;                       Z  - CHANGE MODES FLIP/FLOP
                   38 ;
                   39 ;*************************************************************************
                   40 ;
                   41 ;NOTE:  'SET' COMMANDS IMPLEMENT LOGICAL 'OR' FUNCTIONS
                   42 ;       'RESET' COMMANDS IMPLEMENT LOGICAL 'AND' FUNCTIONS
                   43 ;
                   44 ;*************************************************************************
                   45 ;
                   46 ;BUFFERED MODE MUST BE SELECTED WHEN SELECTIVE RECEIVE IS USED.
                   47 ;
                   48 ;COMMAND FORMAT IS: 'COMMAND (2 LTRS)' 'PAR.#1' 'PAR.#2' ETC.
                   49 ;
                   50 ;THE TRANSMIT FRAME COMMAND FORMAT IS: 'TF' 'A' 'C' 'BUFFER CONTENTS'.
                   51 ;       NO LENGTH COUNT IS NEEDED.  BUFFER CONTENTS IS ENDED WITH A CR.
                   52 ;
                   53 ;*************************************************************************
                   54 ;
                   55 ;POLLED MODE:   WHEN POLLED MODE IS SELECTED (DENOTED BY A '+' PROMPT), IF
```

```
                 ;                A SNRM-P OR RR(0)-P IS RECEIVED, A RESPONSE FRAME OF NSA-F
                 ;                OR RR(0)-F IS TRANSMITTED.  OTHER COMMANDS OPERATE NORMALLY.
                 ;
                 ;*****************************************************************************
                 ;
                 ;8273 EQUATES
                 ;
0090             STAT73  EQU     90H             ;STATUS REGISTER
0090             COMM73  EQU     90H             ;COMMAND REGISTER
0091             PARM73  EQU     91H             ;PARAMETER REGISTER
0091             RESL73  EQU     91H             ;RESULT REGISTER
0092             TXIR73  EQU     92H             ;TX INTERRUPT RESULT REGISTER
0093             RXIR73  EQU     93H             ;RX INTERRUPT RESULT REGISTER
0092             TEST73  EQU     92H             ;TEST MODE REGISTER
0020             CPBF    EQU     20H             ;PARAMETER BUFFER FULL BIT
0004             TXINT   EQU     04H             ;TX INTERRUPT BIT IN STATUS REGISTER
0008             RXINT   EQU     08H             ;RX INTERRUPT BIT IN STATUS REGISTER
0001             TXIRA   EQU     01H             ;TX INT RESULT AVAILABLE BIT
0002             RXIRA   EQU     02H             ;RX INT RESULT AVAILABLE BIT
                 ;
                 ;8253 EQUATES
                 ;
009B             MODE53  EQU     9BH             ;8253 MODE WORD REGISTER
009C             CNT053  EQU     9CH             ;COUNTER 0 REGISTER
009D             CNT153  EQU     9DH             ;COUNTER 1 REGISTER
009E             CNT253  EQU     9EH             ;COUNTER 2 REGISTER
000C             COBR    EQU     000CH           ;CONSOLE BAUD RATE (2400)
0036             MDCNT0  EQU     36H             ;MODE FOR COUNTER 0
00B6             MDCNT2  EQU     0B6H            ;MODE FOR COUNTER 2
2017             LKBR1   EQU     2017H           ;8273 BAUD RATE LSB ADR
2018             LKBR2   EQU     2018H           ;8273 BAUD RATE MSB ADR
                 ;
                 ;BAUD RATE TABLE:       BAUD RATE       LKBR1   LKBR2
                 ;                       *********       *****   *****
                 ;                         9600           2E      00
                 ;                         4800           5C      00
                 ;                         2400           B9      00
                 ;                         1200           72      01
                 ;                          600           E5      02
                 ;                          300           C9      05
                 ;
                 ;
                 ;8257 EQUATES
                 ;
00A8             MODE57  EQU     0A8H            ;8257 MODE PORT
00A0             CH0ADR  EQU     0A0H            ;CH0 (RX) ADR REGISTER
00A1             CH0TC   EQU     0A1H            ;CH0 TERMINAL COUNT REGISTER
00A2             CH1ADR  EQU     0A2H            ;CH1 (TX) ADR REGISTER
00A3             CH1TC   EQU     0A3H            ;CH1 TERMINAL COUNT REGISTER
00A8             STAT57  EQU     0A8H            ;STATUS REGISTER
8200             RXBUF   EQU     8200H           ;RX BUFFER START ADDRESS
8000             TXBUF   EQU     8000H           ;TX BUFFER START ADDRESS
0062             DRDMA   EQU     62H             ;DISABLE RX DMA CHANNEL, TX STILL ON
41FF             RXTC    EQU     41FFH           ;TERMINAL COUNT AND MODE FOR RX CHANNEL
0063             ENDMA   EQU     63H             ;ENABLE BOTH TX AND RX CHANNELS-EXT. WR, TX STOP
0061             DTDMA   EQU     61H             ;DISABLE TX DMA CHANNEL, RX STILL ON
81FF             TXTC    EQU     81FFH           ;TERMINAL COUNT AND MODE FOR TX CHANNEL
                 ;
```

```
                 118 ;8251A EQUATES
                 119 ;
0089             120 CNTL51  EQU     89H             ;CONTROL WORD REGISTER
0089             121 STAT51  EQU     89H             ;STATUS REGISTER
0088             122 TXD51   EQU     88H             ;TX DATA REGISTER
0088             123 RXD51   EQU     88H             ;RX DATA REGISTER
00CE             124 MDE51   EQU     0CEH            ;MODE 16X,2 STOP,NO PARITY
0027             125 CMD51   EQU     27H             ;COMMAND, ENABLE TX&RX
0002             126 RDY     EQU     02H             ;RXRDY BIT
                 127 ;
                 128 ;MONITOR SUBROUTINE EQUATES
                 129 ;
061F             130 GETCH   EQU     061FH           ;GET CHR FROM KEYBOARD, ASCII IN CH
05F8             131 ECHO    EQU     05F8H           ;ECHO CHR TO DISPLAY
075E             132 VALDG   EQU     075EH           ;CHECK IF VALID DIGIT, CARRY SET IF VALID
05BB             133 CNVBN   EQU     05BBH           ;CONVERTS ASCII TO HEX
05EB             134 CRLF    EQU     05EBH           ;DISPLAY CR, HENCE LF TOO
06C7             135 NMOUT   EQU     06C7H           ;CONVERT BYTE TO 2 ASCII CHR AND DISPLAY
                 136 ;
                 137 ;MISC EQUATES
                 138 ;
20C0             139 STKSRT  EQU     20C0H           ;STACK START
0003             140 CNTLC   EQU     03H             ;CNTL-C EQUIVALENT
0008             141 MONTOR  EQU     0008H           ;MONITOR
2000             142 CMDBUF  EQU     2000H           ;START OF COMMAND BUFFER
2020             143 CMDBF1  EQU     2020H           ;POLL MODE SPECIAL TX COMMAND BUFFER
000D             144 CR      EQU     0DH             ;ASCII CR
000A             145 LF      EQU     0AH             ;ASCII LF
20D4             146 RST75   EQU     20D4H           ;RST7.5 JUMP ADDRESS
20CE             147 RST65   EQU     20CEH           ;RST6.5 JUMP ADDRESS
2010             148 LDADR   EQU     2010H           ;RESULT BUFFER LOAD POINTER STORAGE
2013             149 CNADR   EQU     2013H           ;RESULT BUFFER CONSOLE POINTER STORAGE
2800             150 RESBUF  EQU     2800H           ;RESULT BUFFER START - 255 BYTES
0093             151 SNRMP   EQU     93H             ;SNRM-P CONTROL CODE
0011             152 RR0P    EQU     11H             ;RR(0)-P CONTROL CODE
0073             153 NSAF    EQU     73H             ;NSA-F CONTROL CODE
0011             154 RR0F    EQU     11H             ;RR(0)-F CONTROL CODE
2015             155 PRMPT   EQU     2015H           ;PRMPT STORAGE
2016             156 POLIN   EQU     2016H           ;POLL MODE SELECTION INDICATOR
2027             157 DEMODE  EQU     2027H           ;DEMO MODE INDICATOR
                 161 ;
                 162 ;*****************************************************************************
                 163 ;
                 164 ;RAM STORAGE DEFINITIONS:
                 165 ;       LOC             DEF
                 166 ;       ---             ---
                 167 ;       2000-200F       COMMAND BUFFER
                 168 ;       2010-2011       RESULT BUFFER LOAD POINTER
                 169 ;       2013-2014       RESULT BUFFER CONSOLE POINTER
                 170 ;       2015            PROMPT CHARACTER STORAGE
                 171 ;       2016            POLL MODE INDICATOR
                 172 ;       2017            BAUD RATE LSB FOR SELF-TEST
                 173 ;       2018            BAUD RATE MSB FOR SELF-TEST
                 177 ;       2019            SPARE
                 179 ;       2020-2026       RESPONSE COMMAND BUFFER FOR POLL MODE
                 180 ;       2800-28FF       RESULT BUFFER
                 181 ;
                 182 ;*****************************************************************************
```

```
                 183 ;
                 184 ;PROGRAM START
                 185 ;
                 186 ;INITIALIZE 8253, 8257, 8251A, AND RESET 8273.
                 187 ;ALSO SET NORMAL MODE, AND PRINT SIGNON MESSAGE
                 188 ;
0800             189        ORG    800H
                 190
0800 31C020      191 START: LXI    SP,STKSRT       ;INITIALIZE SP
0803 3E36        192        MVI    A,MDCNT0        ;8253 MODE SET
0805 D39B        193        OUT    MODE53          ;8253 MODE PORT
0807 3A1720      194        LDA    LKBR1           ;GET 8273 BAUD RATE LSB
080A D39C        195        OUT    CNT053          ;USING COUNTER 0 AS BAUD RATE GEN
080C 3A1820      196        LDA    LKBR2           ;GET 8273 BUAD RATE MSB
080F D39C        197        OUT    CNT053          ;COUNTER 0
0811 CD1A0B      198        CALL   RXDMA           ;INITIALIZE 8257 RX DMA CHANNEL
0814 CD350B      199        CALL   TXDMA           ;INITIALIZE 8257 TX DMA CHANNEL
0817 3E01        200        MVI    A,01H           ;OUTPUT 1 FOLLOWED BY A 0
0819 D392        201        OUT    TEST73          ;TO TEST MODE REGISTER
081B 3E00        202        MVI    A,00H           ;TO RESET THE 8273
081D D392        203        OUT    TEST73
081F 3E2D        204        MVI    A,'-'           ;NORMAL MODE PROMPT CHR
0821 321520      205        STA    PRMPT           ;PUT IN STORAGE
0824 3E00        206        MVI    A,00H           ;TX POLL RESPONSE INDICATOR
0826 321620      207        STA    POLIN           ;0 MEANS NO SPECIAL TX
0829 332720      208        STA    DEMODE          ;CLEAR DEMO MODE
082C 21A30C      212        LXI    H,SIGNON                ;SIGNON MESSAGE ADR
082F CD920C      213        CALL   TYMSG           ;DISPLAY SIGNON
                 214 ;
                 215 ;MONITOR USES JUMPS IN RAM TO DIRECT INTERRUPTS
                 216 ;
0832 21D420      217        LXI    H,RST75         ;RST7.5 JUMP LOCATION USED BY MONITOR
0835 01000C      218        LXI    B,RXI           ;ADDRESS OF RX INT ROUTINE
0838 36C3        219        MVI    M,0C3H          ;LOAD 'JMP' OPCODE
083A 23          220        INX    H               ;INC POINTER
083B 71          221        MOV    M,C             ;LOAD RXI LSB
083C 23          222        INX    H               ;INC POINTER
083D 70          223        MOV    M,B             ;LOAD RXI MSB
083E 21CE20      224        LXI    H,RST65         ;RST6.5 JUMP LOCATION USED BY MONITOR
0841 01CE0C      225        LXI    B,TXI           ;ADDRESS OF TX INT ROUTINE
0844 36C3        226        MVI    M,0C3H          ;LOAD 'JMP' OPCODE
0846 23          227        INX    H               ;INC POINTER
0847 71          228        MOV    M,C             ;LOAD TXI LSB
0848 23          229        INX    H               ;INC POINTER
0849 70          230        MOV    M,B             ;LOAD TXI MSB
084A 3E18        231        MVI    A,18H           ;GET SET TO RESET INTERRUPTS
084C 30          232        SIM                    ;RESET INTERRUPTS
084D FB          233        EI                     ;ENABLE INTERRUPTS
                 234 ;
                 235 ;INITIALIZE BUFFER POINTER
                 236 ;
                 237 ;
084E 210028      238        LXI    H,RESBUF        ;SET RESULT BUFFER POINTERS
0851 221320      239        SHLD   CNADR           ;RESULT CONSOLE POINTER
0854 221020      240        SHLD   LDADR           ;RESULT LOAD POINTER
                 241 ;
                 242 ;MAIN PROGRAM LOOP - CHECKS FOR CHANGE IN RESULT POINTERS, USART STATUS,
                 243 ;      OR POLL STATUS
```

```
                 ;
0857 CDEB05      CMDREC: CALL    CRLF        ;DISPLAY CR
085A 3A1520              LDA     PRMPT       ;GET CURRENT PROMPT CHR
085D 4F                  MOV     C,A         ;MOVE TO C
085E CDF805              CALL    ECHO        ;DISPLAY IT
0861 2A1320      LOOPIT: LHLD    CNADR       ;GET CONSOLE POINTER
0864 7D                  MOV     A,L         ;SAVE POINTER LSB
0865 2A1020              LHLD    LDADR       ;GET LOAD POINTER
0868 BD                  CMP     L           ;SAME LSB?
0869 C2390A              JNZ     DISPY       ;NO, RESULTS NEED DISPLAYING
086C DB89                IN      STAT51      ;YES, CHECK KEYBOARD
086E E602                ANI     RDY         ;CHR RECEIVED?
0870 C27D08              JNZ     GETCMD      ;MUST BE CHR SO GO GET IT
0873 3A1620              LDA     POLIN       ;GET POLL MODE STATUS
0876 A7                  ANA     A           ;IS IT 0?
0877 C24C09              JNZ     TXPOL       ;NO, THEN POLL OCCURRED
087A C36108              JMP     LOOPIT      ;YES, TRY AGAIN
                 ;
                 ;
                 ;COMMAND RECOGNIZER ROUTINE
                 ;
                 ;
087D CD1F06      GETCMD: CALL    GETCH       ;GET CHR
0880 CDF805              CALL    ECHO        ;ECHO IT
0883 79                  MOV     A,C         ;SETUP FOR COMPARE
0884 FE52                CPI     'R'         ;R?
0886 CAAF08              JZ      RDWN        ;GET MORE
0889 FE53                CPI     'S'         ;S?
088B CAD708              JZ      SDWN        ;GET MORE
088E FE47                CPI     'G'         ;G?
0890 CAFF08              JZ      GDWN        ;GET MORE
0893 FE54                CPI     'T'         ;T?
0895 CA0E09              JZ      TDWN        ;GET MORE
0898 FE41                CPI     'A'         ;A?
089A CA2209              JZ      ADWN        ;GET MORE
089D FE5A                CPI     'Z'         ;Z?
089F CA3109              JZ      CMODE       ;YES, GO CHANGE MODE
08A2 FE03                CPI     CNTLC       ;CNTL-C?
08A4 CA0800              JZ      MONTOR      ;EXIT TO MONITOR
08A7 0E3F        ILLEG:  MVI     C,'?'       ;PRINT ?
08A9 CDF805              CALL    ECHO        ;DISPLAY IT
08AC C35708              JMP     CMDREC      ;LOOP FOR COMMAND
                 
08AF CD1F06      RDWN:   CALL    GETCH       ;GET NEXT CHR
08B2 CDF805              CALL    ECHO        ;ECHO IT
08B5 79                  MOV     A,C         ;SETUP FOR COMPARE
08B6 FE4F                CPI     'O'         ;O?
08B8 CA5D09              JZ      ROCMD       ;RO COMMAND
08BB FE53                CPI     'S'         ;S?
08BD CA6709              JZ      RSCMD       ;RS COMMAND
08C0 FE44                CPI     'D'         ;D?
08C2 CA7109              JZ      RDCMD       ;RD COMMAND
08C5 FE50                CPI     'P'         ;P?
08C7 CAD809              JZ      RPCMD       ;RP COMMAND
08CA FE52                CPI     'R'         ;R?
08CC CA0008              JZ      START       ;START OVER
08CF FE42                CPI     'B'         ;B?
08D1 CA7B09              JZ      RBCMD       ;RB COMMAND
```

```
08D4 C3A708      311          JMP     ILLEG          ;ILLEGAL, TRY AGAIN
                 312
08D7 CD1F06      313 SDWN:    CALL    GETCH          ;GET NEXT CHR
08DA CDF805      314          CALL    ECHO           ;ECHO IT
08DD 78          315          MOV     A,B            ;SETUP FOR COMPARE
08DE FE4F        316          CPI     'O'            ;O?
08E0 CAA609      317          JZ      SOCMD          ;SO COMMAND
08E3 FE53        318          CPI     'S'            ;S?
08E5 CAB009      319          JZ      SSCMD          ;SS COMMAND
08E8 FE52        320          CPI     'R'            ;R?
08EA CABA09      321          JZ      SRCMD          ;SR COMMAND
08ED FE50        322          CPI     'P'            ;P?
08EF CAE209      323          JZ      SPCMD          ;SP COMMAND
08F2 FE42        324          CPI     'B'            ;B?
08F4 CA8509      325          JZ      SBCMD          ;SB COMMAND
08F7 FE4C        326          CPI     'L'            ;L?
08F9 CA8F09      327          JZ      SLCMD          ;SL COMMAND
08FC C3A708      328          JMP     ILLEG          ;ILLEGAL, TRY AGAIN
                 329
08FF CD1F06      330 GDWN:    CALL    GETCH          ;GET NEXT CHR
0902 CDF805      331          CALL    ECHO           ;ECHO IT
0905 78          332          MOV     A,B            ;SETUP FOR COMPARE
0906 FE52        333          CPI     'R'            ;R?
0908 CAC409      334          JZ      GRCMD          ;GR COMMAND
090B C3A708      335          JMP     ILLEG          ;ILLEGAL, TRY AGAIN
                 336
090E CD1F06      337 TDWN:    CALL    GETCH          ;GET NEXT CHR
0911 CDF805      338          CALL    ECHO           ;ECHO IT
0914 78          339          MOV     A,B            ;SETUP FOR COMPARE
0915 FE46        340          CPI     'F'            ;F?
0917 CAEC09      341          JZ      TFCMD          ;TF COMMAND
091A FE4C        342          CPI     'L'            ;L?
091C CA9909      343          JZ      TLCMD          ;TL COMMAND
091F C3A708      344          JMP     ILLEG          ;ILLEGAL, TRY AGAIN
                 345
0922 CD1F06      346 ADWN:    CALL    GETCH          ;GET NEXT CHR
0925 CDF805      347          CALL    ECHO           ;ECHO IT
0928 78          348          MOV     A,B            ;SETUP FOR COMPARE
0929 FE46        349          CPI     'F'            ;F?
092B CACE09      350          JZ      AFCMD          ;AF COMMAND
092E C3A708      351          JMP     ILLEG          ;ILLEGAL, TRY AGAIN
                 352 ;
                 353 ;RESET POLL MODE RESPONSE - CHANGE PROMPT CHR AS INDICATOR
                 354 ;
0931 F3          355 CMODE:   DI                     ;DISABLE INTERRUPTS
0932 3A1520      356          LDA     PRMPT          ;GET CURRENT PROMPT
0935 FE2D        357          CPI     '-'            ;NORMAL MODE?
0937 C24309      358          JNZ     SW             ;NO, CHANGE IT
093A 3E2B        359          MVI     A,'+'          ;NEW PROMPT
093C 321520      360          STA     PRMPT          ;STORE NEW PROMPT
093F FB          365          EI                     ;ENABLE INTERRUPTS
0940 C35708      366          JMP     CMDREC         ;RETURN TO LOOP
0943 3E2D        367 SW:      MVI     A,'-'          ;NEW PROMPT CHR
0945 321520      368          STA     PRMPT          ;STORE IT
0948 FB          369          EI                     ;ENABLE INTERRUPTS
0949 C35708      370          JMP     CMDREC         ;RETURN TO LOOP
                 371 ;
                 372 ;
```

```
                373 ;TRANSMIT ANSWER TO POLL SETUP
                374 ;
094C 3E00       382 TXPOL:  MVI     A,00H           ;CLEAR POLL INDICATOR
094E 321620     384         STA     POLIN           ;INDICATOR ADR
0951 216108     385         LXI     H,LOOPIT        ;SETUP STACK FOR COMMAND OUTPUT
0954 E5         386         PUSH    H               ;PUT RETURN TO CMDREC ON STACK
0955 0604       387         MVI     B,04H           ;GET # OF PARAMETERS READY
0957 212020     388         LXI     H,CMDBF1        ;POINT TO SPECIAL BUFFER
095A C3FF0A     389         JMP     COMM2           ;JUMP TO COMMAND OUTPUTER
                390 ;
                391 ;
                392 ;
                393 ;COMMAND IMPLEMENTING ROUTINES
                394 ;
                395 ;
                396 ;RO - RESET OPERATING MODE
                397 ;
095D 0601       398 ROCMD:  MVI     B,01H           ;# OF PARAMETERS
095F 0E51       399         MVI     C,51H           ;COMMAND
0961 CDE50A     400         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
0964 C35708     401         JMP     CMDREC          ;GET NEXT COMMAND
                402 ;
                403 ;RS - RESET SERIAL I/O MODE COMMAND
                404 ;
0967 0601       405 RSCMD:  MVI     B,01H           ;# OF PARAMETERS
0969 0E60       406         MVI     C,60H           ;COMMAND
096B CDE50A     407         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
096E C35708     408         JMP     CMDREC          ;GET NEXT COMMAND
                409 ;
                410 ;RD - RECEIVER DISABLE COMMAND
                411 ;
0971 0600       412 RDCMD:  MVI     B,00H           ;# OF PARAMETERS
0973 0EC5       413         MVI     C,0C5H          ;COMMAND
0975 CDE50A     414         CALL    COMM            ;ISSUE COMMAND
0978 C35708     415         JMP     CMDREC          ;GET NEXT COMMAND
                416 ;
                417 ;RB - RESET ONE BIT DELAY COMMAND
                418 ;
097B 0601       419 RBCMD:  MVI     B,01H           ;# OF PARAMETERS
097D 0E64       420         MVI     C,64H           ;COMMAND
097F CDE50A     421         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
0982 C35708     422         JMP     CMDREC          ;GET NEXT COMMAND
                423 ;
                424 ;SB - SET ONE BIT DELAY COMMAND
                425 ;
0985 0601       426 SBCMD:  MVI     B,01H           ;# OF PARAMETERS
0987 0EA4       427         MVI     C,0A4H          ;COMMAND
0989 CDE50A     428         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
098C C35708     429         JMP     CMDREC          ;GET NEXT COMMAND
                430 ;
                431 ;SL - SELECTIVE LOOP RECEIVE COMMAND
                432 ;
098F 0604       433 SLCMD:  MVI     B,04H           ;# OF PARAMETERES
0991 0EC2       434         MVI     C,0C2H          ;COMMAND
0993 CDE50A     435         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
0996 C35708     436         JMP     CMDREC          ;GET NEXT COMMAND
                437 ;
                438 ;TL - TRANSMIT LOOP COMMAND
```

```
                 ;
0999 210020      TLCMD:  LXI     H,CMDBUF        ;SET COMMAND BUFFER POINTER
099C 0602                MVI     B,02H           ;LOAD PARAMETER COUNTER
099E 36CA                MVI     M,0CAH          ;LOAD COMMAND INTO BUFFER
09A0 210220              LXI     H,CMDBUF+2      ;POINT AT ADR AND CNTL POSITIONS
09A3 C3F609              JMP     TFCMD1          ;FINISH OFF COMMAND IN TF ROUTINE
                 ;
                 ;SO - SET OPERATING MODE COMMAND
                 ;
09A6 0601        SOCMD:  MVI     B,01H           ;# OF PARAMETERS
09A8 0E91                MVI     C,91H           ;COMMAND
09AA CDE50A              CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
09AD C35708              JMP     CMDREC          ;GET NEXT COMMAND
                 ;
                 ;SS - SET SERIAL I/O COMMAND
                 ;
09B0 0601        SSCMD:  MVI     B,01H           ;# OF PARAMETERS
09B2 0EA0                MVI     C,0A0H          ;COMMAND
09B4 CDE50A              CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
09B7 C35708              JMP     CMDREC          ;GET NEXT COMMAND
                 ;
                 ;SR - SELECTIVE RECEIVE COMMAND
                 ;
09BA 0604        SRCMD:  MVI     B,04H           ;# OF PARAMETERS
09BC 0EC1                MVI     C,0C1H          ;COMMAND
09BE CDE50A              CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
09C1 C35708              JMP     CMDREC          ;GET NEXT COMMAND
                 ;
                 ;GR - GENERAL RECEIVE COMMAND
                 ;
09C4 0602        GRCMD:  MVI     B,02H           ;NO PARAMETERS
09C6 0EC0                MVI     C,0C0H          ;COMMAND
09C8 CDE50A              CALL    COMM            ;ISSUE COMMAND
09CB C35708              JMP     CMDREC          ;GET NEXT COMMAND
                 ;
                 ;AF - ABORT FRAME COMMAND
                 ;
09CE 0600        AFCMD:  MVI     B,00H           ;NO PARAMETERS
09D0 0ECC                MVI     C,0CCH          ;COMMAND
09D2 CDE50A              CALL    COMM            ;ISSUE COMMAND
09D5 C35708              JMP     CMDREC          ;GET NEXT COMMAND
                 ;
                 ;RP - RESET PORT COMMAND
                 ;
09D8 0601        RPCMD:  MVI     B,01H           ;# OF PARAMETERS
09DA 0E63                MVI     C,63H           ;COMMAND
09DC CDE50A              CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
09DF C35708              JMP     CMDREC          ;GET NEXT COMMAND
                 ;
                 ;SP - SET PORT COMMAND
                 ;
09E2 0601        SPCMD:  MVI     B,01H           ;# OF PARAMETERS
09E4 0EA3                MVI     C,0A3H          ;COMMAND
09E6 CDE50A              CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
09E9 C35708              JMP     CMDREC          ;GET NEX COMMAND
                 ;
                 ;TF - TRANSMIT FRAME COMMAND
                 ;
```

```
09EC 210020      TFCMD:  LXI     H,CMDBUF        ;SET COMMAND BUFFER POINTER
09EF 0602                MVI     B,02H           ;LOAD PARAMETER COUNTER
09F1 36C8                MVI     M,0C8H          ;LOAD COMMAND INTO BUFFER
09F3 210220              LXI     H,CMDBUF+2      ;POINT AT ADR AND CNTL POSITIONS
09F6 78          TFCMD1: MOV     A,B             ;TEST PARAMETER COUNT
09F7 A7                  ANA     A               ;IS IT 0?
09F8 CA070A              JZ      TBUFL           ;YES, LOAD TX DATA BUFFER
09FB CDAD0A              CALL    PARIN           ;GET PARAMETER
09FE DAA708              JC      ILLEG           ;ILLEGAL CHR RETURNED
0A01 23                  INX     H               ;INC COMMAND BUFFER POINTER
0A02 05                  DCR     B               ;DEC PARAMETER COUNTER
0A03 77                  MOV     M,A             ;LOAD PARAMETER INTO COMMAND BUFFER
0A04 C3F609              JMP     TFCMD1          ;GET NEXT PARAMETER

0A07 210080      TBUFL:  LXI     H,TXBUF         ;LOAD TX DATA BUFFER POINTER
0A0A 010000              LXI     B,0000H         ;CLEAR BC - BYTE COUNTER
0A0D C5          TBUFL1: PUSH    B               ;SAVE BYTE COUNTER
0A0E CDAD0A              CALL    PARIN           ;GET DATA, ALIAS PARAMETER
0A11 DA1B0A              JC      ENDCHK          ;MAYBE END IF ILLEGAL
0A14 77                  MOV     M,A             ;LOAD DATA BYTE INTO BUFFER
0A15 23                  INX     H               ;INC BUFFER POINTER
0A16 C1                  POP     B               ;RESTORE BYTE COUNTER
0A17 03                  INX     B               ;INC BYTE COUNTER
0A18 C30D0A              JMP     TBUFL1          ;GET NEXT DATA
0A1B FE0D        ENDCHK: CPI     CR              ;RETURNED ILLEGAL CHR CR?
0A1D CA240A              JZ      TBUFFL          ;YES, THEN TX BUFFER FULL
0A20 C1                  POP     B               ;RESTORE B TO SAVE STACK
0A21 C3A708              JMP     ILLEG           ;ILLEGAL CHR
0A24 C1          TBUFFL: POP     B               ;RESTORE BYTE COUNTER
0A25 210120              LXI     H,CMDBUF+1      ;POINT INTO COMMAND BUFFER
0A28 71                  MOV     M,C             ;STORE BYTE COUNT LSB
0A29 23                  INX     H               ;INC POINTER
0A2A 70                  MOV     M,B             ;STORE BYTE COUNT MSB
0A2B 0604                MVI     B,04H           ;LOAD PARAMETER COUNT INTO B
0A2D 21360A              LXI     H,TFRET         ;GET RETURN ADR FOR THIS ROUTINE
0A30 C5                  PUSH    B               ;PUSH ONCE
0A31 E3                  XTHL                    ;PUT RETURN ON STACK
0A32 C5                  PUSH    B               ;PUSH IT SO CMDOUT CAN USE IT
0A33 C3FB0A              JMP     CMDOUT          ;ISSUE COMMAND
0A36 C35708      TFRET:  JMP     CMDREC          ;GET NEXT COMMAND
                 ;
                 ;
                 ;ROUTINE TO DISPLAY RESULT IN RESULT BUFFER WHEN LOAD AND CONSOLE
                 ;POINTERS ARE DIFFERENT.
                 ;
                 ;
0A39 1605        DISPY:  MVI     D,05H           ;D IS RESULT COUNTER
0A3B 2A1320              LHLD    CNADR           ;GET CONSOLE POINTER
0A3E E5                  PUSH    H               ;SAVE IT
0A3F 7E                  MOV     A,M             ;GET RESULT IC
0A40 E61F                ANI     1FH             ;LIMIT TO RESULT CODE
0A42 FE0C                CPI     0CH             ;TEST IF RX OR TX SOURCE
0A44 DA620A              JC      RXSORC          ;CARRY, THEN RX SOURCE
0A47 21C30C      TXSORC: LXI     H,TXIMSG        ;TX INT MESSAGE
0A4A CD920C              CALL    TYMSG           ;DISPLAY IT
0A4D E1          DISPY2: POP     H               ;RESTORE CONSOLE POINTER
0A4E 7E          DISPY1: MOV     A,M             ;GET RESULT
0A4F CDC706              CALL    NMOUT           ;CONVERT AND DISPLAY
```

```
0A52 0E20        555          MVI    C,' '          ;SP CHR
0A54 CDF805      556          CALL   ECHO           ;DISPLAY IT
0A57 2C          557          INR    L              ;INC BUFFER POINTER
0A58 15          558          DCR    D              ;DEC RESULT COUNTER
0A59 C24E0A      559          JNZ    DISPY1         ;NOT DONE
0A5C 221320      560          SHLD   CNADR          ;UPDATE CONSOLE POINTER
0A5F C35708      561          JMP    CMDREC         ;RETURN TO LOOP
                 562 ;
                 563 ;
                 564 ;RECEIVER SOURCE - DISPLAY RESULTS AND RECEVIE BUFFER CONTENTS
                 565 ;
                 566 ;
0A62 21B80C      567 RXSORC: LXI    H,RXIMSG       ;RX INT MESSAGE ADR
0A65 CD920C      568          CALL   TYMSG          ;DISPLAY MESSAGE
0A68 E1          569          POP    H              ;RESTORE CONSOLE POINTER
0A69 7E          570 RXS1:    MOV    A,M            ;RETRIEVE RESULT FROM BUFFER
0A6A CDC706      571          CALL   NMOUT          ;CONVERT AND DISPLAY IT
0A6D 0E20        572          MVI    C,' '          ;ASCII SP
0A6F CDF805      573          CALL   ECHO           ;DISPLAY IT
0A72 2C          574          INR    L              ;INC CONSOLE POINTER
0A73 15          575          DCR    D              ;DEC RESULT COUNTER
0A74 7A          576          MOV    A,D            ;GET SET TO TEST COUNTER
0A75 FE04        577          CPI    04H            ;IS THE RESULT R0?
0A77 CAA20A      578          JZ     R0PT           ;YES, GO SAVE IT
0A7A FE03        579          CPI    03H            ;IS THE RESULT R1?
0A7C CAA70A      580          JZ     R1PT           ;YES, GO SAVE IT
0A7F A7          581 RXS2:    ANA    A              ;TEST RESULT COUNTER
0A80 C2690A      582          JNZ    RXS1           ;NOT DONE YET, GET NEXT RESULT
0A83 221320      583          SHLD   CNADR          ;DONE, SO UPDATE CONSOLE POINTER
0A86 CDEB05      584          CALL   CRLF           ;DISPLAY CR
0A89 210082      585          LXI    H,RXBUF        ;POINT AT RX BUFFER
0A8C C1          586          POP    B              ;RETRIEVE RECEIVED COUNT
0A8D 78          587 RXS3:    MOV    A,B            ;IS COUNT 0?
0A8E B1          588          ORA    C              ;
0A8F CA5708      589          JZ     CMDREC         ;YES, GO BACK TO LOOP
0A92 7E          590          MOV    A,M            ;NO, GET CHR
0A93 C5          591          PUSH   B              ;SAVE BC
0A94 CDC706      592          CALL   NMOUT          ;CONVERT AND DISPLAY CHR
0A97 0E20        593          MVI    C,' '          ;ASCII SP
0A99 CDF805      594          CALL   ECHO           ;DISPLAY IT TO SEPARATE DATA
0A9C C1          595          POP    B              ;RESTORE BC
0A9D 0B          596          DCX    B              ;DEC COUNT
0A9E 23          597          INX    H              ;INC POINTER
0A9F C38D0A      598          JMP    RXS3           ;GET NEXT CHR
                 599
0AA2 4E          600 R0PT:    MOV    C,M            ;GET R0 FOR RESULT BUFFER
0AA3 C5          601          PUSH   B              ;SAVE IT
0AA4 C37F0A      602          JMP    RXS2           ;RETURN
                 603
0AA7 C1          604 R1PT:    POP    B              ;GET R0
0AA8 46          605          MOV    B,M            ;GET R1 FOR RESULT BUFFER
0AA9 C5          606          PUSH   B              ;SAVE IT
0AAA C37F0A      607          JMP    RXS2
                 608 ;
                 609 ;
                 610 ;
                 611 ;PARAMETER INPUT - PARAMETER RETURNED IN E REGISTER
                 612 ;
```

```
                  ;
0AAD C5           PARIN:  PUSH    B           ;SAVE BC
0AAE 1601                 MVI     D,01H       ;SET CHR COUNTER
0AB0 CD1F06               CALL    GETCH       ;GET CHR
0AB3 CDF805               CALL    ECHO        ;ECHO IT
0AB6 79                   MOV     A,C         ;PUT CHR IN A
0AB7 FE20                 CPI     ' '         ;SP?
0AB9 C2E00A               JNZ     PARIN1      ;NO, ILLEGAL, TRY AGAIN
0ABC CD1F06       PARIN3: CALL    GETCH       ;GET CHR OF PARAMETER
0ABF CDF805               CALL    ECHO        ;ECHO IT
0AC2 CD5E07               CALL    VALDG       ;IS IT A VALID CHR?
0AC5 D2E00A               JNC     PARIN1      ;NO, TRY AGAIN
0AC8 CDBB05               CALL    CNVBN       ;CONVERT IT TO HEX
0ACB 4F                   MOV     C,A         ;SAVE IT IN C
0ACC 7A                   MOV     A,D         ;GET CHR COUNTER
0ACD A7                   ANA     A           ;IS IT 0?
0ACE CADC0A               JZ      PARIN2      ;YES,DONE WITH THIS PARAMETER
0AD1 15                   DCR     D           ;DEC CHR COUNTER
0AD2 AF                   XRA     A           ;CLEAR CARRY
0AD3 79                   MOV     A,C         ;RECOVER 1ST CHR
0AD4 17                   RAL                 ;ROTATE LEFT 4 PLACES
0AD5 17                   RAL
0AD6 17                   RAL
0AD7 17                   RAL
0AD8 5F                   MOV     E,A         ;SAVE IT IN E
0AD9 C3BC0A               JMP     PARIN3      ;GET NEXT CHR
0ADC 79           PARIN2: MOV     A,C         ;2ND CHR IN A
0ADD B3                   ORA     E           ;COMBINE BOTH CHRS
0ADE C1                   POP     B           ;RESTORE BC
0ADF C9                   RET                 ;RETURN TO CALLING PROGRAM
0AE0 79           PARIN1: MOV     A,C         ;PUT ILLEGAL CHR IN A
0AE1 37                   STC                 ;SET CARRY AS ILLEGAL STATUS
0AE2 C1                   POP     B           ;RESTORE BC
0AE3 C9                   RET                 ;RETURN TO CALLING PROGRAM
                  ;
                  ;
                  ;JUMP HERE IF BUFFER FULL
                  ;
0AE4 CF           BUFFUL: DB      0CFH        ;EXIT TO MONITOR
                  ;
                  ;
                  ;COMMAND DISPATCHER
                  ;
                  ;
0AE5 210020       COMM:   LXI     H,CMDBUF    ;SET POINTER
0AE8 C5                   PUSH    B           ;SAVE BC
0AE9 71                   MOV     M,C         ;LOAD COMMAND INTO BUFFER
0AEA 78           COMM1:  MOV     A,B         ;CHECK PARAMETER COUNTER
0AEB A7                   ANA     A           ;IS IT 0?
0AEC CAFB0A               JZ      CMDOUT      ;YES, GO ISSUE COMMAND
0AEF CDAD0A               CALL    PARIN       ;GET PARAMETER
0AF2 DAA708               JC      ILLEG       ;ILLEGAL CHR RETURNED
0AF5 23                   INX     H           ;INC BUFFER POINTER
0AF6 05                   DCR     B           ;DEC PARAMETER COUNTER
0AF7 77                   MOV     M,A         ;PARAMETER TO BUFFER
0AF8 C3EA0A               JMP     COMM1       ;GET NEXT PARAMETER
0AFB 210020       CMDOUT: LXI     H,CMDBUF    ;REPOINT POINTER
0AFE C1                   POP     B           ;RESTORE PARAMETER COUNT
```

```
0AFF DB90        671 COMM2:  IN      STAT73        ;READ 8273 STATUS
0B01 07          672         RLC                   ;ROTATE CBSY INTO CARRY
0B02 DAFF0A      673         JC      COMM2         ;WAIT FOR OK
0B05 7E          674         MOV     A,M           ;OK, MOVE COMMAND INTO A
0B06 D390        675         OUT     COMM73        ;OUTPUT COMMAND
0B08 78          676 PAR1:   MOV     A,B           ;GET PARAMETER COUNT
0B09 A7          677         ANA     A             ;IS IT 0?
0B0A C8          678         RZ                    ;YES, DONE, RETURN
0B0B 23          679         INX     H             ;INC COMMAND BUFFER POINTER
0B0C 05          680         DCR     B             ;DEC PARAMETER COUNT
0B0D DB90        681 PAR2:   IN      STAT73        ;READ STATUS
0B0F E620        682         ANI     CPBF          ;IS CPBF BIT SET?
0B11 C20D0B      683         JNZ     PAR2          ;WAIT TIL ITS 0
0B14 7E          684         MOV     A,M           ;OK, GET PARAMETER FROM BUFFER
0B15 D391        685         OUT     PARM73        ;OUTPUT PARAMETER
0B17 C3080B      686         JMP     PAR1          ;GET NEXT PARAMETER
                 687 ;
                 688 ;
                 689 ;INITIALIZE AND ENABLE RX DMA CHANNEL
                 690 ;
                 691 ;
0B1A 3E62        692 RXDMA:  MVI     A,DRDMA       ;DISABLE RX DMA CHANNEL
0B1C D3A8        693         OUT     MODE57        ;8257 MODE PORT
0B1E 010082      694         LXI     B,RXBUF       ;RX BUFFER START ADDRESS
0B21 79          695         MOV     A,C           ;RX BUFFER LSB
0B22 D3A0        696         OUT     CH0ADR        ;CH0 ADR PORT
0B24 78          697         MOV     A,B           ;RX BUFFER MSB
0B25 D3A0        698         OUT     CH0ADR        ;CH0 ADR PORT
0B27 01FF41      699         LXI     B,RXTC        ;RX CH TEERMINAL COUNT
0B2A 79          700         MOV     A,C           ;RX TERMINAL COUNT LSB
0B2B D3A1        701         OUT     CH0TC         ;CH0 TC PORT
0B2D 78          702         MOV     A,B           ;RX TERMINAL COUNT MSB
0B2E D3A1        703         OUT     CH0TC         ;CH0 TC PORT
0B30 3E63        704         MVI     A,ENDMA       ;ENABLE DMA WORD
0B32 D3A8        705         OUT     MODE57        ;8257 MODE PORT
0B34 C9          706         RET                   ;RETURN
                 707 ;
                 708 ;
                 709 ;INITIALIZE AND ENABLE TX DMA CHANNEL
                 710 ;
                 711 ;
0B35 3E61        712 TXDMA:  MVI     A,DTDMA       ;DISABLE TX DMA CHANNEL
0B37 D3A8        713         OUT     MODE57        ;8257 MODE PORT
0B39 010080      714         LXI     B,TXBUF       ;TX BUFFER START ADDRESS
0B3C 79          715         MOV     A,C           ;TX BUFFER LSB
0B3D D3A2        716         OUT     CH1ADR        ;CH1 ADR PORT
0B3F 78          717         MOV     A,B           ;TX BUFFER MSB
0B40 D3A2        718         OUT     CH1ADR        ;CH1 ADR PORT
0B42 01FF81      719 TXDMA1: LXI     B,TXTC        ;TX CH TERMINAL COUNT
0B45 79          720         MOV     A,C           ;TX TERMINAL COUNT LSB
0B46 D3A3        721         OUT     CH1TC         ;CH1 TC PORT
0B48 78          722         MOV     A,B           ;TX TERMINAL COUNT MSB
0B49 D3A3        723         OUT     CH1TC         ;CH1 TC PORT
0B4B 3E63        724         MVI     A,ENDMA       ;ENABLE DMA WORD
0B4D D3A8        725         OUT     MODE57        ;8257 MODE PORT
0B4F C9          726         RET                   ;RETURN
                 727 ;
                 728 ;
```

```
                 ;INERRUPT PROCESSING SECTION
                 ;
0C00             ORG     0C00H
                 ;
                 ;
                 ;RECEIVER INTERRUPT - RST 7.5 (LOC 3CH)
                 ;
0C00 E5          RXI:    PUSH    H               ;SAVE HL
0C01 F5                  PUSH    PSW             ;SAVE PSW
0C02 C5                  PUSH    B               ;SAVE BC
0C03 D5                  PUSH    D               ;SAVE DE
0C04 3E62                MVI     A,DRDMA         ;DISABLE RX DMA
0C06 D3A8                OUT     MODE57          ;8257 MODE PORT
0C08 3E18                MVI     A,18H           ;RESET RST7.5 F/F
0C0A 30                  SIM
0C0B 1604                MVI     D,04H           ;D IS RESULT COUNTER
0C0D 2A1020              LHLD    LDADR           ;GET LOAD POINTER
0C10 E5                  PUSH    H               ;SAVE IT
0C11 E5                  PUSH    H               ;SAVE IT AGAIN
0C12 45                  MOV     B,L             ;SAVE LSB
0C13 2A1320              LHLD    CNADR           ;GET CONSOLE POINTER
0C16 04          RXI1:   INR     B               ;BUMP LOAD POINTER LSB
0C17 78                  MOV     A,B             ;GET SET TO TEST
0C18 BD                  CMP     L               ;LOAD=CONSOLE?
0C19 CAE40A              JZ      BUFFUL          ;YES, BUFFER FULL
0C1C 15                  DCR     D               ;DEC COUNTER
0C1D C2160C              JNZ     RXI1            ;NOT DONE, TRY AGAIN
0C20 1605                MVI     D,05H           ;RESET COUNTER
0C22 E1                  POP     H               ;RESTORE LOAD POINTER
0C23 DB90        RXI2:   IN      STAT73          ;READ STATUS
0C25 E608                ANI     RXINT           ;TEST RX INT BIT
0C27 CA390C              JZ      RXI3            ;DONE, GO FINISH UP
0C2A DB90                IN      STAT73          ;READ STATUS AGAIN
0C2C E602                ANI     RXIRA           ;IS RESULT READY?
0C2E CA230C              JZ      RXI2            ;NO, TEST AGAIN
0C31 DB93                IN      RXIR73          ;YES, READ RESULT
0C33 77                  MOV     M,A             ;STORE IN BUFFER
0C34 2C                  INR     L               ;INC BUFFER POINTER
0C35 15                  DCR     D               ;DEC COUNTER
0C36 C3230C              JMP     RXI2            ;GET MORE RESULTS
0C39 7A          RXI3:   MOV     A,D             ;GET SET TO TEST
0C3A A7                  ANA     A               ;ALL RESULTS?
0C3B CA450C              JZ      RXI4            ;YES, SO FINISH UP
0C3E 3600                MVI     M,00H           ;NO, LOAD 0 TIL DONE
0C40 2C                  INR     L               ;BUMP POINTER
0C41 15                  DCR     D               ;DEC COUNTER
0C42 C3390C              JMP     RXI3            ;GO AGAIN
0C45 221020      RXI4:   SHLD    LDADR           ;UPDATE LOAD POINTER
0C48 3A1520              LDA     PRMPT           ;GET MODE INDICATOR
0C4B FE2D                CPI     '-'             ;NORMAL MODE?
0C4D CA850C              JZ      RXI6            ;YES, CLEAN UP BEFORE RETURN
                 ;
                 ;       POLL MODE SO CHECK CONTROL BYTE
                 ;       IF CONTROL IS A POLL, SET UP SPECIAL TX COMMAND BUFFER
                 ;       AND RETURN WITH POLL INDICATOR NOT 0
                 ;
0C50 E1          POP     H               ;GET PREVIOUS LOAD ADR POINTER
0C51 7E          MOV     A,M             ;GET IC BYTE FROM BUFFER
```

```
0C52 E61E        787          ANI    1EH          ;LOOK AT GOOD FRAME BITS
0C54 C2890C      788          JNZ    RXI5         ;IF NOT 0, INTERRUPT WASN'T FROM A GOOD FRAME
0C57 2C          789          INR    L            ;BYPASS R0 AND R1 IN BUFFER
0C58 2C          790          INR    L
0C59 2C          791          INR    L
0C5A 56          792          MOV    D,M          ;GET ADR BYTE AND SAVE IT IN D
0C5B 2C          793          INR    L
0C5C 7E          794          MOV    A,M          ;GET CNTL BYTE FROM BUFFER
0C5D FE93        795          CPI    SNRMP        ;WAS IT SNRM-P?
0C5F CA6C0C      796          JZ     T1           ;YES, GO SET RESPONSE
0C62 FE11        797          CPI    RR0P         ;WAS IT RR(0)-P?
0C64 C2890C      798          JNZ    RXI5         ;YES, GO SET RESPONSE, OTHERWISE RETURN
0C67 1E11        799          MVI    E,RR0F       ;RR(0)-P SO SET RESPONSE TO RR(0)-F
0C69 C36E0C      800          JMP    TXRET        ;GO FINISH LOADING SPECIAL BUFFER
0C6C 1E73        801 T1:      MVI    E,NSAF       ;SNRM-P SO SET RESPONSE TO NSA-F
0C6E 212020      802 TXRET:   LXI    H,CMDBF1     ;SPECIAL BUFFER ADR
0C71 36C8        806          MVI    M,0C8H       ;LOAD TX FRAME COMMAND
0C73 23          808          INX    H            ;INC POINTER
0C74 3600        809          MVI    M,00H        ;L0=0
0C76 23          810          INX    H            ;INC POINTER
0C77 3600        811          MVI    M,00H        ;L1=0
0C79 23          812          INX    H            ;INC POINTER
0C7A 72          813          MOV    M,D          ;LOAD RCVD ADR BYTE
0C7B 23          814          INX    H            ;INC POINTER
0C7C 73          815          MOV    M,E          ;LOAD RESPONSE CNTL BYTE
0C7D 3E01        816          MVI    A,01H        ;SET POLL INDICATOR NOT 0
0C7F 321620      817          STA    POLIN        ;LOAD POLL INDICATOR
0C82 C3890C      818          JMP    RXI5         ;RETURN
                 819
0C85 E1          820 RXI6:    POP    H            ;CLEAN UP STACK IF NORMAL MODE
0C86 C3890C      821          JMP    RXI5         ;RETURN
                 822
0C89 CD1A0B      823 RXI5:    CALL   RXDMA        ;RESET DMA CHANNEL
0C8C D1          824          POP    D            ;RESTORE REGISTERS
0C8D C1          825          POP    B
0C8E F1          826          POP    PSW
0C8F E1          827          POP    H
0C90 FB          828          EI                  ;ENABLE INTERRUPTS
0C91 C9          829          RET                 ;RETURN
                 830 ;
                 831 ;
                 832 ;MESSAGE TYPER - ASSUMES MESSAGE STARTS AT HL
                 833 ;
                 834 ;
0C92 C5          835 TYMSG:   PUSH   B            ;SAVE BC
0C93 7E          836 TYMSG2:  MOV    A,M          ;GET ASCII CHR
0C94 23          837          INX    H            ;INC POINTER
0C95 FEFF        838          CPI    0FFH         ;STOP?
0C97 CAA10C      839          JZ     TYMSG1       ;YES, GET SET FOR EXIT
0C9A 4F          840          MOV    C,A          ;SET UP FOR DISPLAY
0C9B CDF805      841          CALL   ECHO         ;DISPLAY CHR
0C9E C3930C      842          JMP    TYMSG2       ;GET NEXT CHR
0CA1 C1          843 TYMSG1:  POP    B            ;RESTORE BC
0CA2 C9          844          RET                 ;RETURN
                 845 ;
                 846 ;
                 847 ;SIGNON MESSAGE
                 848 ;
```

```
0CA3 0D          849 SIGNON: DB      CR,'8273 MONITOR  V1.1',CR,0FFH
0CA4 38323733
0CA8 204D4F4E
0CAC 49544F52
0CB0 20205631
0CB4 2E31
0CB6 0D
0CB7 FF
                 850 ;
                 851 ;
                 852 ;
                 853 ;RECEIVER INTERRUPT MESSAGES
                 854 ;
                 855 ;
0CB8 0D          856 RXIMSG: DB      CR,'RX INT - ',0FFH
0CB9 52582049
0CBD 4E54202D
0CC1 20
0CC2 FF
                 857 ;
                 858 ;TRANSMITTER INTERRUPT MESSAGES
                 859 ;
0CC3 0D          860 TXIMSG: DB      CR,'TX INT - ',0FFH
0CC4 54582049
0CC8 4E54202D
0CCC 20
0CCD FF
                 861 ;
                 862 ;
                 863 ;TRANSMITTER INTERRUPT ROUTINE
                 864 ;
0CCE E5          865 TXI:    PUSH    H               ;SAVE HL
0CCF F5          866         PUSH    PSW             ;SAVE PSW
0CD0 C5          867         PUSH    B               ;SAVE BC
0CD1 D5          868         PUSH    D               ;SAVE DE
0CD2 3E61        869         MVI     A,DTDMA         ;DISABLE TX DMA
0CD4 D3A8        870         OUT     MODE57          ;8257 MODE PORT
0CD6 1604        871         MVI     D,04H           ;SET COUNTER
0CD8 2A1020      872         LHLD    LDADR           ;GET LOAD POINTER
0CDB E5          873         PUSH    H               ;SAVE IT
0CDC 45          874         MOV     B,L             ;SAVE LSB IN B
0CDD 2A1320      875         LHLD    CNADR           ;GET CONSOLE POINTER
0CE0 04          876 TXI1:   INR     B               ;INC POINTER
0CE1 78          877         MOV     A,B             ;GET SET TO TEST
0CE2 BD          878         CMP     L               ;LOAD=CONSOLE?
0CE3 CAE40A      879         JZ      BUFFUL          ;YES, BUFFER FULL
0CE6 15          880         DCR     D               ;NO, TEST NEXT LOCATION
0CE7 C2E00C      881         JNZ     TXI1            ;TRY AGAIN
0CEA E1          882         POP     H               ;RESTORE LOAD POINTER
0CEB DB92        883         IN      TXIR73          ;READ RESULT
0CED 77          884         MOV     M,A             ;STORE IN BUFFER
0CEE 2C          885         INR     L               ;INR POINTER
0CEF 3600        886         MVI     M,00H           ;EXTRA RESULT SPOTS 0
0CF1 2C          887         INR     L
0CF2 3600        888         MVI     M,00H
0CF4 2C          889         INR     L
0CF5 3600        890         MVI     M,00H
0CF7 2C          891         INR     L
```

```
0CF8 3600        892        MVI     M,00H
0CFA 2C          893        INR     L
0CFB 221020      894        SHLD    LDADR          ;UPDATE LOAD POINTER
0CFE CD350B      899        CALL    TXDMA          ;RESET DMA CHANNEL
0D01 D1          900        POP     D              ;RESTORE DE
0D02 C1          901        POP     B              ;RESTORE BC
0D03 F1          902        POP     PSW            ;RESTORE PSW
0D04 E1          903        POP     H              ;RESTORE HL
0D05 FB          904        EI                     ;ENABLE INTERRUPTS
0D06 C9          905        RET                    ;RETURN
                 906 ;
                 907 ;
                 952 ;
                 953 ;
                 954        END
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

```
ADWN   A 0922   AFCMD  A 09CE   BUFFUL A 0AE4   CH0ADR A 00A0   CH0TC  A 00A1   CH1ADR A 00A2   CH1TC  A 00A3
CMD51  A 0027   CMDBF1 A 2020   CMDBUF A 2000   CMDOUT A 0AFB   CMDREC A 0857   CMODE  A 0931   CNADR  A 2013
CNT053 A 009C   CNT153 A 009D   CNT253 A 009E   CNTL51 A 0089   CNTLC  A 0003   CNVBN  A 05BB   COBR   A 000C
COMM   A 0AE5   COMM1  A 0AEA   COMM2  A 0AFF   COMM73 A 0090   CPBF   A 0020   CR     A 000D   CRLF   A 05EB
DEM    A 0000   DEMODE A 2027   DISPY  A 0A39   DISPY1 A 0A4E   DISPY2 A 0A4D   DRDMA  A 0062   DTDMA  A 0061
ECHO   A 05F8   ENDCHK A 0A1B   ENDMA  A 0063   GDWN   A 08FF   GETCH  A 061F   GETCMD A 087D   GRCMD  A 09C4
ILLEG  A 08A7   LDADR  A 2010   LF     A 000A   LKBR1  A 2017   LKBR2  A 2018   LOOPIT A 0861   MDCNT0 A 0036
MDCNT2 A 00B6   MDE51  A 00CE   MODE53 A 009B   MODE57 A 00A8   MONTOR A 0008   NMOUT  A 06C7   NSAF   A 0073
PAR1   A 0B08   PAR2   A 0B0D   PARIN  A 0AAD   PARIN1 A 0AE0   PARIN2 A 0ADC   PARIN3 A 0ABC   PARM73 A 0091
POLIN  A 2016   PRMPT  A 2015   R0PT   A 0AA2   R1PT   A 0AA7   RBCMD  A 097B   RDCMD  A 0971   RDWN   A 08AF
RDY    A 0002   RESBUF A 2800   RESL73 A 0091   ROCMD  A 095D   RPCMD  A 09D8   RR0F   A 0011   RR0P   A 0011
RSCMD  A 0967   RST65  A 20CE   RST75  A 20D4   RXBUF  A 8200   RXD51  A 0088   RXDMA  A 0B1A   RXI    A 0C00
RXI1   A 0C16   RXI2   A 0C23   RXI3   A 0C39   RXI4   A 0C45   RXI5   A 0C89   RXI6   A 0C85   RXIMSG A 0CB8
RXINT  A 0008   RXIR73 A 0093   RXIRA  A 0002   RXS1   A 0A69   RXS2   A 0A7F   RXS3   A 0A8D   RXSORC A 0A62
RXTC   A 41FF   SBCMD  A 0985   SDWN   A 08D7   SIGNON A 0CA3   SLCMD  A 098F   SNRMP  A 0093   SOCMD  A 09A6
SPCMD  A 09E2   SRCMD  A 09BA   SSCMD  A 09B0   START  A 0800   STAT51 A 0089   STAT57 A 00A8   STAT73 A 0090
STKSRT A 20C0   SW     A 0943   T1     A 0C6C   TBUFFL A 0A24   TBUFL  A 0A07   TBUFL1 A 0A0D   TDWN   A 090E
TEST73 A 0092   TFCMD  A 09EC   TFCMD1 A 09F6   TFRET  A 0A36   TLCMD  A 0999   TRUE   A 0000   TRUE1  A 0000
TXBUF  A 8000   TXD51  A 0088   TXDMA  A 0B35   TXDMA1 A 0B42   TXI    A 0CCE   TXI1   A 0CE0   TXIMSG A 0CC3
TXINT  A 0004   TXIR73 A 0092   TXIRA  A 0001   TXPOL  A 094C   TXRET  A 0C6E   TXSORC A 0A47   TXTC   A 81FF
TYMSG  A 0C92   TYMSG1 A 0CA1   TYMSG2 A 0C93   VALDG  A 075E
```

ASSEMBLY COMPLETE. NO ERRORS

CRT Terminal Design Using The Intel® 8275 and 8279

by John Murray and George Alexy

1. INTRODUCTION

The purpose of this application note is to provide the reader with the conceptual and factual tools needed to apply the 8275 Programmable CRT Controller and 8279 Programmable Keyboard/Display Interface in CRT system design. The 8275 Controller is designed to interface CRT raster scan displays with Intel® Microcomputer Products. Its primary functions include refreshing the CRT display by buffering information from display memory and generating horizontal and vertical timing signals used for CRT synchronization. The programmable features of the 8275 allow it to be interfaced to almost any raster scan display with a minimum of external hardware. In addition, visual attribute features allow the implementation of specialized graphic display functions and display enhancement operations. The 8279 Keyboard Interface provides key scanning, debounce, and buffering features required for interfacing CRT terminal keyboards to the system processor. Two key or N-key rollover is provided. The use of these devices in a microcomputer based CRT terminal yields substantial savings in component count, printed circuit board area, and power consumption.

The application note is divided into five sections:

1. Introduction
2. CRT System Design Concepts
3. Component Description
4. CRT System Design Example
5. Appendix

Readers desiring an overview of CRT system design should consider reading the first three sections of the application note. Individuals requiring an in-depth knowledge of CRT system design should read the first three sections, then proceed to the design example. The design example consists of a description of the design of a complete CRT terminal. Both hardware and software aspects of the design are included. It will be assumed in Section 4 that the reader is familiar with the 8275, 8279, and 8257 data sheets, and the operation of the 8080A microprocessor.

2. CRT SYSTEM DESIGN CONCEPTS

2.1 CRT OPERATION

In order to fully understand the CRT terminal design process, it is necessary to consider the fundamentals of CRT operation. A typical CRT Monitor is shown in Figure 2-1. The CRT consists of an evacuated glass structure having a phosphorescent coating on the inner surface of the rectangular frontal region (screen). A filament contained in the narrow cylindrical region (neck) of the CRT heats the cathode, causing the cathode to give off electrons by thermionic emission. Heating is accomplished by applying a low voltage source across the filament leads. A high voltage source applied between the cathode and the screen electrode (anode) accelerates the electrons toward the screen. The electron beam, upon striking the phosphorescent inner surface of the screen, produces light. To control the point at which the beam strikes the screen, two primary deflection techniques are utilized. The first technique, electromagnetic deflection, involves applying a current through a deflection coil placed around the neck of the CRT. The resulting magnetic field forces the electron beam to be deflected in proportion to the magnitude of the applied current. Electrostatic deflection involves placing deflection electrodes in the neck of the CRT perpendicular to the electron beam. An applied voltage changes the position of the beam accordingly.

2.2 MONITOR OPERATION

A CRT monitor consists of a CRT and the electronics required for positioning the beam in the desired manner. A block diagram of the control electronics contained within a typical CRT monitor is provided in Figure 2-2.

The horizontal oscillator is designed to move the electron beam horizontally across the CRT screen and then return the beam rapidly to its original position. As the beam is moved horizontally, the vertical oscillator causes the beam to be deflected vertically. The net result of these operations is to move the beam in a manner shown in Figure 2-3. If the intensity of the electron beam is modulated in a controlled manner as the beam sweeps across the screen, it is possible to display pictorial information on the CRT screen surface. It will be assumed that the monitor in question will be used for displaying alphanumeric characters or graphic symbols. In this case, the electron beam will be turned on to display a light region on the screen and turned off to display a dark region. Display information appearing at the video input to the CRT is applied through the video amplifier to a control grid located in the neck of the CRT. The magnitude of the video signal determines whether the electron beam will be on or off.

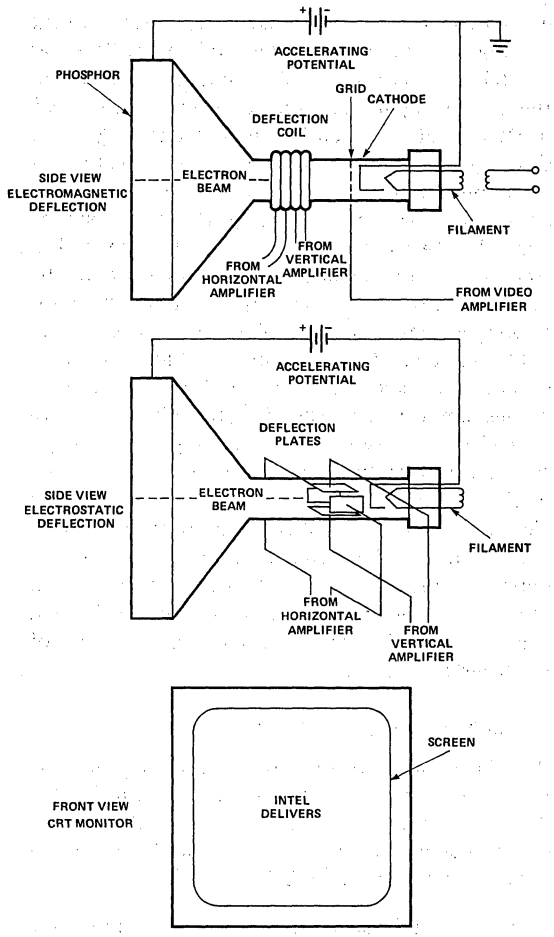


Figure 2-1. CRT Monitor

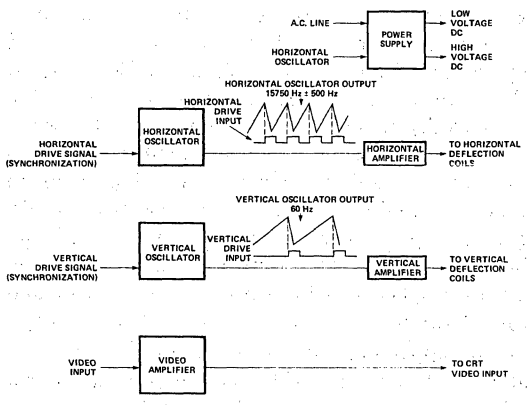


Figure 2-2. CRT Monitor Electronics

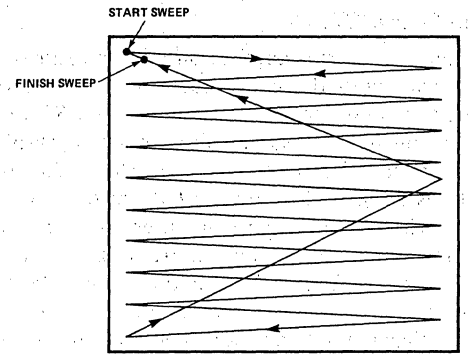


Figure 2-3. CRT Monitor Raster

2.3 CRT TERMINAL DESCRIPTION

A CRT terminal consists basically of a CRT monitor, monitor control electronics, memory for storing display information, logic to control information transfer to and from external devices and between internal devices, and a keyboard. The fundamental operations performed by a CRT terminal consist of the display of information contained in internal memory on the CRT screen, communication with manual data entry devices such as keyboards or light pens, and communication with external intelligent devices such as computers or data communication terminals. Typical CRT terminal communication functions are illustrated in Figure 2-4.

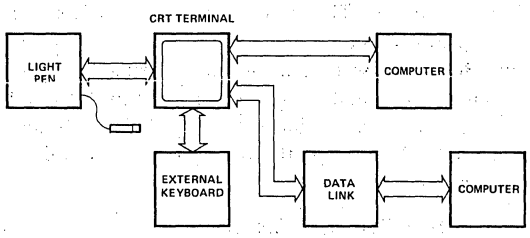


Figure 2-4. CRT Terminal Communications

2.4 CRT TERMINAL IMPLEMENTATION

A typical microprocessor-based CRT terminal is presented in block diagram form in Figure 2-5. The terminal consists of the CRT monitor, monitor electronics, memory for storing the information to be displayed, a serial communication device, keyboard, keyboard interface device, CRT controller, central processor and associated program memory, and a DMA device. The primary function of the CRT controller is to refresh the display. It does this by controlling the periodic transfer of information from display memory to the CRT screen. The central processor unit (CPU) coordinates the transfer of information to and from the terminal peripheral devices and external devices. When information from an external device is received by the terminal, the central processor performs character recognition and handling functions, display memory management functions, and cursor control functions. The CPU also interrogates the keyboard interface device. If a key depression is detected by the keyboard interface device, the CPU responds by transmitting the ASCII character representing the key to the terminal serial output line via the serial communication device. A direct memory access (DMA) device is required in the system to effect the necessary memory to screen data transfer rate.

The CRT terminal control functions under consideration may be implemented with LSI devices at a considerable cost savings over earlier terminal designs using MSI and SSI components. This cost savings is complemented by an increase in the number of features which can be incorporated in terminal designs. The additional features stem from the programmable nature of the devices. In addition, utilizing a microprocessor as the terminal controller allows considerable intelligence to be built into the terminal for decision making, computational, and control functions. The design example presented in Section 4 of the application note illustrates the use of the 8275 Programmable CRT Controller and 8279 Keyboard Controller in a typical terminal design. In the following section, the 8275 and 8279 are considered in depth.

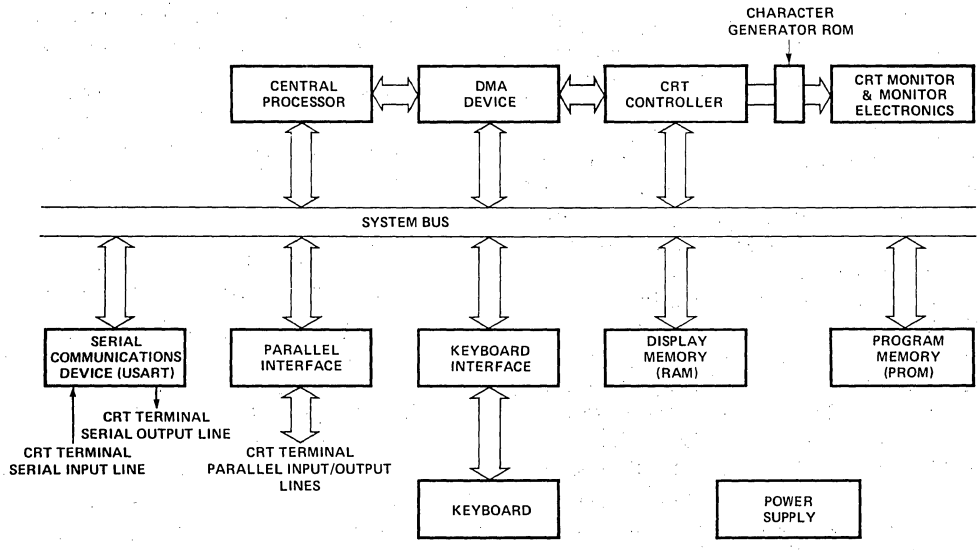


Figure 2-5. CRT Terminal Block Diagram

3. COMPONENT DESCRIPTION

3.1 8275

The block diagram and pin configuration for the 8275 Programmable CRT Controller are presented in Figure 3-1. The 8275 provides the following general capabilities:

1. *CRT Display Refreshing* – The 8275, having been programmed to a specific screen format, generates a series of DMA request signals, resulting in the transfer of a row of characters from display memory, via the 8257 DMA Controller, to the 8275's row buffers. The 8275 presents the character codes to an external character generator ROM. The 8275 character code outputs CC0–CC6 are used for this purpose. External dot timing logic is then utilized to transfer the parallel output data from the character generator ROM, serially, to the video input of the CRT. The character rows are displayed on the CRT one line at a time. Line count outputs LC0–LC3 are applied to the character generator ROM to perform the line selection function. The display process is graphically illustrated in Figure 3-2. The entire process is repeated for each display row. At the beginning of the last display row, the 8275 issues an interrupt via the INT output line. The 8275 interrupt output will normally be connected to the interrupt input of the system central processor. The interrupt causes the CPU to execute an interrupt service subroutine. The service subroutine typically re-initializes DMA controller parameters for the next display refresh cycle, polls the system keyboard controller, and/or executes other appropriate functions. A block diagram of a CRT system implemented with the 8275 CRT Controller is provided in Figure 3-3. Proper CRT refreshing requires that certain 8275 parameters be programmed prior to the beginning of display operation. The 8275 has two types of programming registers, the Command Registers (CREG) and the Parameter Registers (PREG). It also has a Status Register (SREG). The Command Registers may only be written to and the Status Registers may only be read. The 8275 expects to receive a command followed by a sequence of from 0 to 4 parameters, depending on the command. The 8275 instruction set consists of 8 commands:

COMMAND	NO. OF PARAMETER BYTES	NOTES
RESET	4	Display format parameters required
START DISPLAY	0	DMA operation parameters included in command
STOP DISPLAY	0	—
READ LIGHT PEN	2	—
LOAD CURSOR	2	Cursor X,Y position parameters required
ENABLE INTERRUPT	0	—
DISABLE INTERRUPT	0	—
PRESET COUNTERS	0	Clears all internal counters

In order to establish the format of the display, the 8275 provides a number of user programmable display format parameters. Display formats having from 1 to 80 characters per row, 1 to 64 rows per screen, and from 1 to 16 horizontal lines per row are available.

In addition to transferring characters from memory to the CRT screen, the 8275 features cursor position control. The cursor position may be programmed, via X and Y cursor position registers, to any character position on the display. The user may select from 4 cursor formats. Blinking or non-blinking underline and reverse video block cursors are available.

2. *CRT Timing* – The 8275 provides two timing outputs, HRTC and VRTC, which are utilized in synchronizing CRT horizontal and vertical oscillators to the 8275 refresh cycle. In addition, whenever HRTC or VRTC are active, a third timing output, VSP (Video Suppress) is true, providing a blanking signal to the dot timing logic. The dot timing logic will normally inhibit the video output to the CRT during the time when video suppress signal is true. An additional timing output, LTEN (Light Enable) is used to provide the ability to force the video output high regardless of the state of VSP. This feature is utilized by

the 8275 to place a cursor on the screen and to control attribute functions. Attributes will be considered in the next section.

The HLGT (Highlight) output allows an attribute function to increase the CRT beam intensity to a level greater than normal. The fifth timing signal, RVV (Reverse Video) will, when enabled, cause the system video output to be inverted.

3. *Special Functions* –

VISUAL ATTRIBUTES – Visual attributes are special codes which, when retrieved from display memory by the 8275, affect the visual characteristics of a character position or field of characters. Two types of visual attributes exist, character attributes and field attributes.

Character Attribute Codes: Character attribute codes are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA0–LA1), the Video Suppression output (VSP), and the Light Enable output. The dot timing logic uses these signals to generate the proper symbols. Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT). Character attributes were designed to produce the graphic symbols shown in Figure 3-4.

Field Attribute Codes: The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the field attribute code up to, and including, the character which precedes the next field attribute code, or up to the end of the frame.

There are six field attributes:

1. *Blink* – Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. *Highlight* – Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. *Reverse Video* – Characters following the code are caused to appear in reverse video format by activating the Reverse Video output (RVV).
4. *Underline* – Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).
5. *General Purpose* – There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. These attributes may be used to select colors or perform other desired control functions.

The 8275 can be programmed to provide visible or invisible field attribute characters as shown in Figure 3-5. If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character. If the 8275 is programmed in the invisible field attribute mode, the 8275 row buffer FIFOs are activated. The FIFOs effectively lengthen the row buffers by 16 characters, making room for up to 16 field attribute characters per display row. The FIFOs are 16 characters by 7 bits in size. When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the next character in the proper FIFO. When a field attribute is placed in the buffer output controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CC0–6). The chosen attributes are also activated.

LIGHT PEN DETECTION – A light pen consists fundamentally of a switch and light sensor. When the light pen is pressed against the CRT screen, the switch enables the light sensor. When the raster sweep coincides with the light sensor position on the display, the light pen output is acti-

vated. If the output of the light pen is presented to the 8275 LPEN input, the row and character position coordinates are stored in two 8275 internal registers. These registers can be read on command by the microprocessor.

SPECIAL CODES – Four special codes may be used to help reduce memory, software, or DMA overhead. These codes are placed in character positions in display memory.

1. *End of Row Code* –

   Activates VSP. VSP remains active until the end of the line is reached. While VSP is active, the screen is blanked.

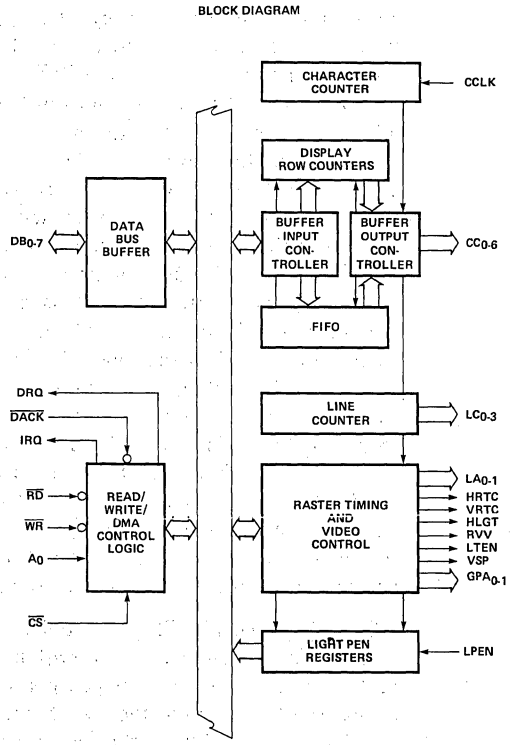
2. *End of Row-Stop DMA Code* – Causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the row buffer. It affects the display in the same way as the End of Row Code.

3. *End of Screen Code* –

   Activates VSP. VSP remains active until the end of the frame is reached.

4. *End of Screen-Stop DMA Code* – Causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the row buffer. It affects the display in the same way as the End of Screen Code.

PROGRAMMABLE DMA BURST CONTROL – The 8275 can be programmed to request single byte DMA transfers or DMA burst transfers of 2, 4, or 8 characters per burst. The interval between bursts is also programmable. This allows the user to tailor his DMA overhead to fit his system needs.



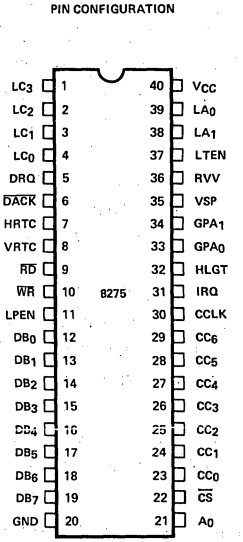


Figure 3-1. 8275 Block Diagram/Pin Configuration

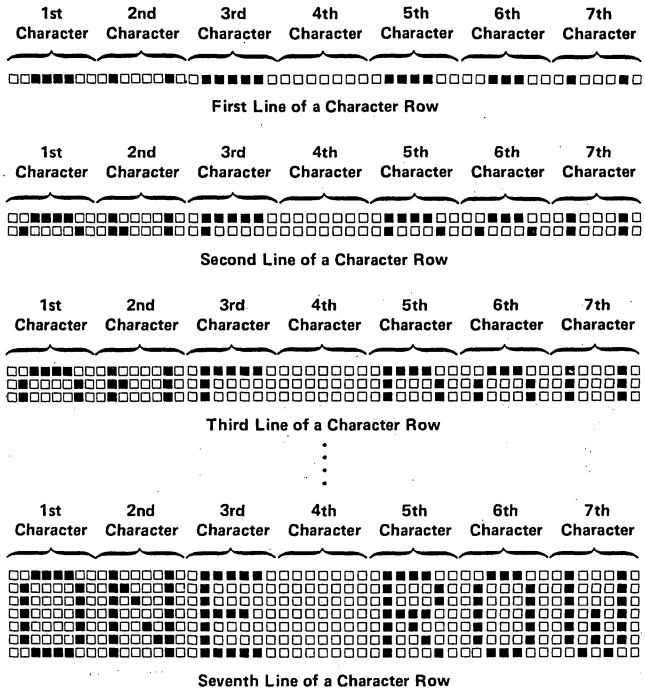


Figure 3-2. 8275 Row Display

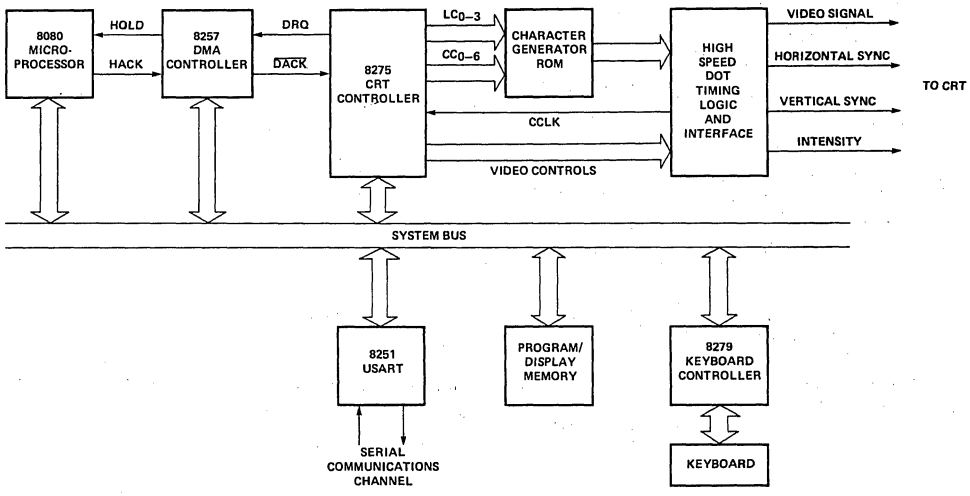


Figure 3-3. CRT System Block Diagram

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"		OUTPUTS LA_1	LA_0	VSP	LTEN	SYMBOL	DESCRIPTION
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

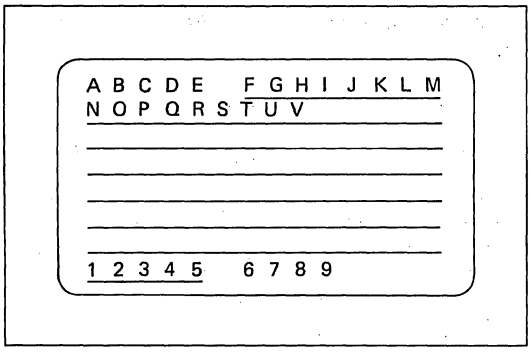
*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

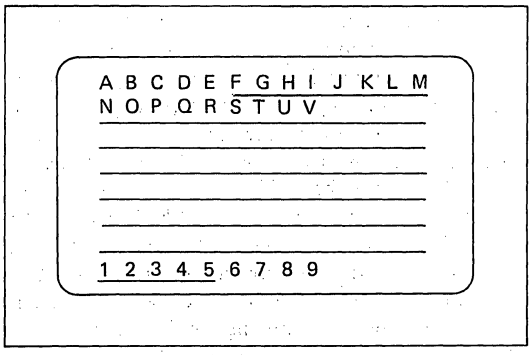
Blinking is active when B = 1.

Highlight is active when H = 1.

Figure 3-4. Character Attributes



EXAMPLE OF THE VISIBLE FIELD ATTRIBUTE MODE (UNDERLINE ATTRIBUTE)



EXAMPLE OF THE INVISIBLE FIELD ATTRIBUTE MODE (UNDERLINE ATTRIBUTE)

Figure 3-5. Field Attribute Examples

3.2 8279

The 8279 Programmable Keyboard/Display Interface block diagram and pin configuration are shown in Figure 3-6. The 8279 will be utilized in the CRT design example for performing keyboard scanning, key debounce, and data bus interface functions. Only features associated with these functions will be described in this section. The reader is referred to the 8279 data sheet for information on display control, sensor matrix mode operation, and strobed input mode operation. A detailed description of the 8279 keyboard scanning, debounce, and data bus interface functions follows.

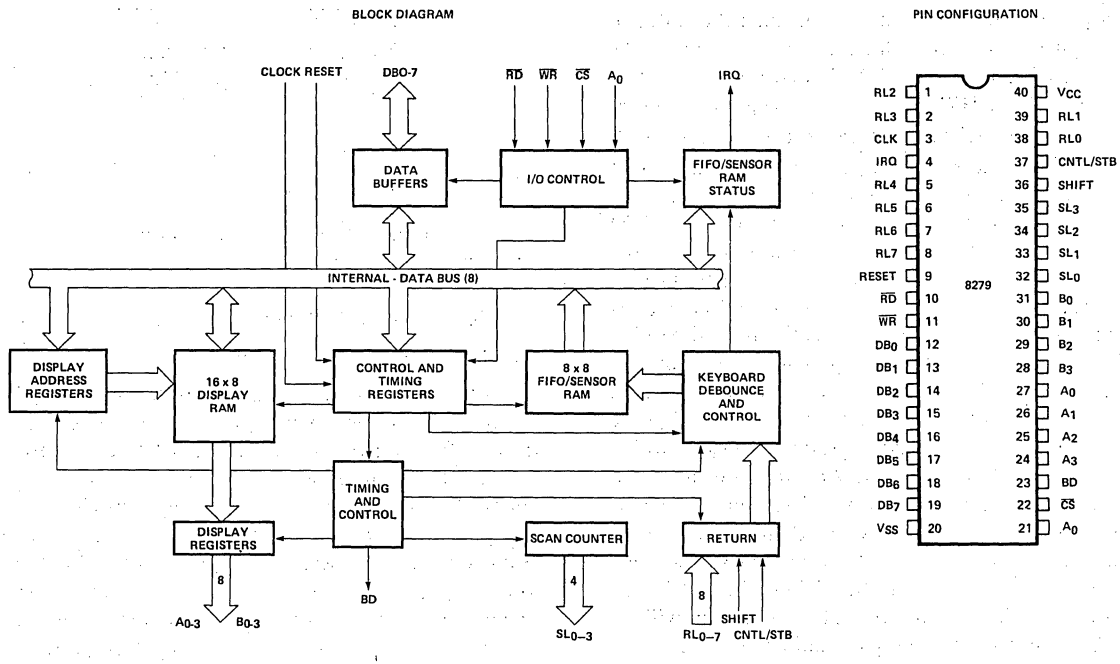


Figure 3-6. 8279 Pin Configuration and Block Diagram

The primary functions of the 8279 in the CRT system application include scanning the 64 key keyboard, determining if a key has been depressed, and, when polled by the system processor, transmitting the address of the key in the keyboard matrix to the master processor. Alternately, the interrupt line from the 8279 may be used to inform the CPU of a key depression. A block diagram of the 8279 interface, as implemented in the CRT system design example, is provided in Figure 3-7. The keyboard controller initiates the keyboard scanning process by transmitting keyboard scan line selection information over output lines SL_0–SL_2. The data may be encoded or decoded depending on the mode programmed. Assuming encoded mode is selected, the SL_0–SL_2 lines are connected to the input of a 3-line to 8-line decoder as shown in Figure 3-7. The decoder outputs are connected to the keyboard row inputs. Only one decoder output will be enabled for a given set of input conditions. The keyboard column outputs are connected to the 8279 return line inputs RL_0–RL_7. The eight return lines are buffered and latched by the 8279. These lines are scanned by the internal logic of the 8279, looking for a key depression in the selected row. If the debounce circuit detects a key depression, it waits approximately 10 ms to determine if the key remains down. If it does, the address of the key in the matrix plus the status of the shift and control lines are transferred to the 8279 FIFO. The FIFO data format is shown in Figure 3-8. The FIFO will hold up to eight data bytes; that is, up to eight key depressions may occur prior to a CPU initiated read operation. The number of characters entered into the FIFO is indicated by the character count contained within the FIFO status word. When a key depression is detected, the 8279 interrupt line goes high, and the FIFO status is modified to reflect the number of characters contained in the FIFO. The CPU may determine the occurrence of a key depression in one of two ways: The 8279 interrupt line may be connected to the interrupt input line of the CPU, forcing the CPU to call an interrupt service routine which reads the FIFO character. An alternate approach requires the CPU to periodically poll the 8279, reading the FIFO status word. If the FIFO character count is non-zero, indicating that at least one character is present in the FIFO, the CPU then reads the FIFO contents. This approach will be utilized in the CRT design example. A read operation places the contents of the FIFO on the system data bus and decrements the FIFO character count, contained within the FIFO status word, by one.

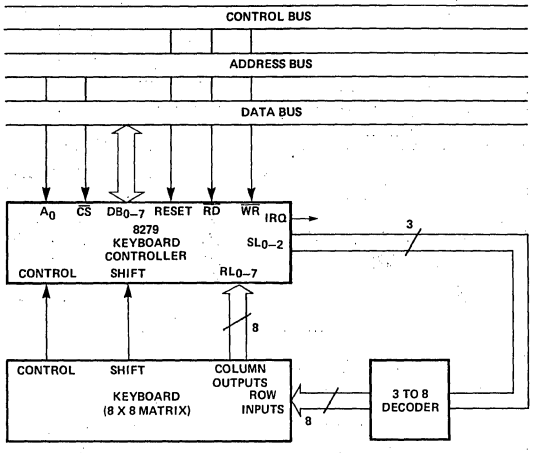


Figure 3-7. 8279 Interface

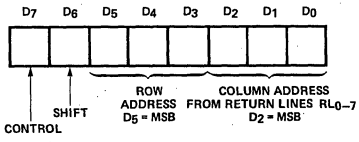


Figure 3-8. FIFO Data Byte Format

4. CRT SYSTEM DESIGN EXAMPLE

4.1 SCOPE OF THE PROJECT

A fully operational, microcomputer-based CRT terminal was designed and constructed utilizing the 8275 CRT Controller and 8279 Keyboard Controller as the basic system elements. The terminal incorporates the majority of the functions found in existing dedicated computer terminals. An Intel® 8080A microprocessor was utilized as the CPU in the design. The recently announced Intel® 8085 microprocessor constitutes an ideal processor for future CRT terminal designs. LSI devices were utilized in the design whenever possible in order to minimize component count.

4.2 SYSTEM SPECIFICATIONS

The specifications for the CRT terminal design are as follows:

Display Format

- 80 characters/display row
- 25 display rows

Character Format (Figure 4-1)

- 5X7 character contained within a 7X10 matrix, 1st and 10th lines blanked, 1st and 7th columns blanked, 9th line cursor position, blinking underline cursor.

Characters Recognized

- Displayable characters: 64 ASCII upper-case alphanumeric characters
- Control characters:
  Line feed, Control J
  Carriage return, Control M
  Back space, Control H
- Escape Sequences:
  Cursor up, ESC, A
  Cursor down, ESC, B
  Cursor right, ESC, C
  Cursor left, ESC, D
  Clear screen, ESC, E
  Home, ESC, H
  Erase to end of screen, ESC, J
  Erase line, ESC, K

Characters Transmitted

- 64 ASCII upper-case alphanumeric characters
- ASCII Control Character set
- ASCII Escape Sequence set

Program Memory

- 2K bytes, 2716 EPROM

Display/Buffer/Stack Memory

- 2K bytes, 2114 static RAM

Data Rate

- 4800 BAUD maximum using 8080A

CRT Monitor

- Ball Bros TV-12, 12 MHz B.W.

Keyboard

- Microswitch hall effect keyboard, open collector outputs

Scrolling Capability

- Scroll up feature implemented with 8257 DMA Controller

Screen Refresh Rate

- 60 Hz

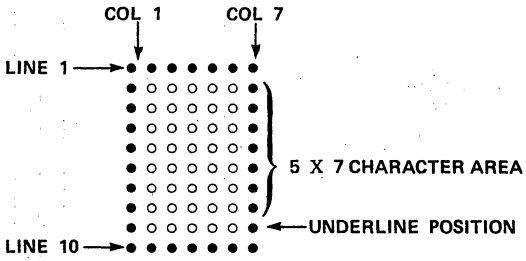


Figure 4-1. Character Format

4.3 SYSTEM HARDWARE DESIGN

4.3.1 General Considerations

A block diagram of the CRT terminal is presented in Figure 4-2. The diagram includes only essential system features. A detailed schematic of the CRT terminal is contained in the appendix. The terminal was constructed using an Intel® SDK-80 microcomputer kit and an Intel® SBC 905 prototyping board. The standard 8080 bus structure incorporated in the SDK-80 kit allowed the CRT terminal to be implemented with minimum buffering.

In the ensuing discussion of CRT terminal operation, it will be assumed that the terminal normally communicates with a remote device, such as an Intel® MDS microcomputer development system. Communication will take place in the full duplex mode. The CRT terminal, upon transmitting a character to the remote device, will remain idle until a character is received from the external device. Transmission of a character to the remote device is initiated by depressing a key on the keyboard. Character transmission to the CRT terminal from the remote device is assumed to be asynchronous with respect to terminal operation.

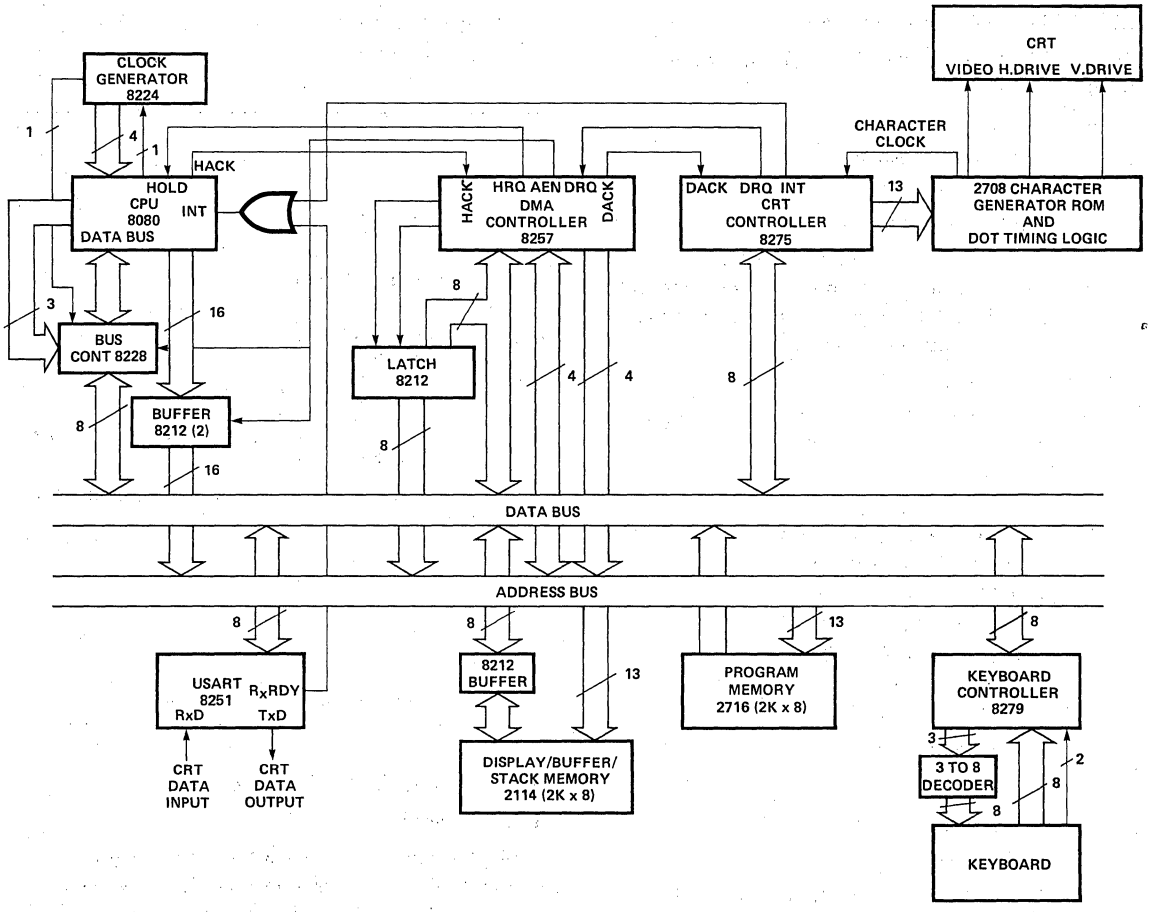


Figure 4-2. CRT Terminal Block Diagram

4.3.2 Operation

The 8080A CPU initializes each peripheral to the appropriate mode of operation following system reset. Upon receiving a character from a remote device, the 8251 USART issues an interrupt to the CPU. The CPU calls the interrupt service subroutine, which polls both the 8275 and 8251 to determine the source of the interrupt. Having determined that the 8251 issued the interrupt, the CPU calls the READ/STORE USART character subroutine, reads the USART character, and stores the character in buffer memory. The character recognition subroutine is called next. This routine determines whether the character is a displayable character, a control character, or a character in an escape sequence. Assuming the character is a displayable character, the CPU places the character in display memory at the location corresponding to the present cursor position, advances the cursor, modifies the display memory pointers, and, if required, performs the operations necessary for scrolling. If the received character is a control character or escape sequence character requiring cursor and display memory pointer changes, these functions are carried out. Escape sequences which involve erasing a portion of the display are also handled via the appropriate subroutines.

In order to place characters contained in display memory on the CRT display screen, the 8275 CRT Controller must first transfer the display characters, via the 8257 DMA Controller, to the 8275's row buffers. It should be noted that the 8257 DMA Controller is required to achieve the data transfer

rate necessary for CRT refreshing. Display characters are then transferred from the 8275 row buffers to the character code outputs CC0–CC5. The character code outputs are applied to the character generator address lines A3–A8 (Figure 4-3). Line count outputs LC0–LC2 from the 8275 are applied to character generator address lines A0–A2. It should be noted that the 8275 displays character rows one line at a time. The line count outputs are utilized to determine which line of the character selected by A3–A8 will be displayed. Following the transfer of the first line to the dot timing logic, the line count is incremented and the second line of the character row is selected. The process continues until the last line of the row under consideration is transferred to the dot timing logic.

The dot timing logic latches the 6-bit character code and 3-bit line count from the 8275 on positive transitions of the character clock and transfers this information to the character generator ROM. In systems requiring a greater number of lines/character, the fourth line count output would also be used. The 7-bit ROM output corresponds to the 7 dots which make up a line segment for a particular character. The ROM output is loaded into a parallel input-serial output shift register. The shift register is clocked at the dot clock rate (11.34 MHz) continuously. The shift register output constitutes the video input to the CRT. The character code outputs select the character to be displayed at a given character position in the display row. The character set consists of 2^6=64 ASCII upper case alphanumeric characters.

The row by row transfer of character data from display memory to the 8275 continues until the beginning of the last display row. At this time the 8275 issues an interrupt to the CPU. The CPU polls both the 8275 and 8251. Having determined that the interrupt originated with the 8275, the CPU calls the 8275 interrupt subroutine. The 8275 interrupt subroutine re-initializes the 8257 DMA Controller starting address and terminal count parameters and polls the 8279 Keyboard Controller to determine if a key depression has occurred. If a key has been depressed, the CPU reads the key position data from the 8279, performs a table lookup, and transmits the appropriate ASCII character to the CRT data output via the 8251 USART. It should be noted that interrupts are generated by the 8275 every 16.67 ms for a 60 Hz screen refresh rate.

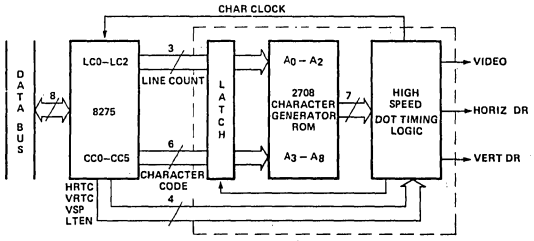


Figure 4-3. Character Generator/Dot Timing Logic Block Diagram

4.3.3 System Timing

The CRT terminal display raster is shown in Figure 4-4. It can be seen from the figure that a display row is composed of 10 lines. The Total Line Time consists of the display portion of the line plus the Horizontal Blanking Time. Row Time is equal to the number of lines per row multiplied by the Total Line Time. The Total Screen Time (1/Refresh Rate) is equal to the Row Time multiplied by the number of display rows plus the Row Time intervals associated with vertical blanking. Specifications for the BALL BROS. monitor show that there are constraints on the Vertical Blanking Time, Horizontal Blanking Time, and Horizontal Oscillator Repetition Rate. These constraints are summarized in Table 4-1.

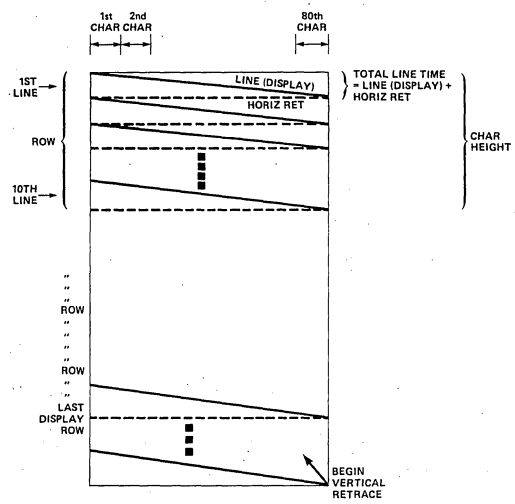


Figure 4-4. CRT Display Raster

Table 4-1

PARAMETER	RANGE
Vertical Blanking Time (VRTC)	900 μsec nominal
Vertical Drive Pulsewidth	300 μsec ≤ PW ≤ 1.4 ms
Horizontal Blanking Time (HRTC)	11 μsec nominal
Horizontal Drive Pulsewidth	25 μsec ≤ PW ≤ 30 μsec
Horizontal Repetition Rate	15,750 ±500 pps

Given the constraints in Table 4-1 and the Refresh Rate specification of 60 Hz, the Vertical Retrace Row Count and Horizontal Retrace Character Count parameters required by the 8275 CRT Controller may be calculated:

$$\text{Total Screen Time} = \frac{1}{\text{Refresh rate}} = \frac{1}{60\ \text{Hz}}$$
$$= 0.01667\ \text{sec}$$

Also,

$$\text{Total Screen Time} = (\text{Row Time})\ (\#\ \text{of Display Rows}) + \text{Vertical Blanking Time (VRTC)}$$

Vertical Blanking Time (VRTC) must be an integral number of Row Times (between 1 and 4).

Therefore,

$$0.016667\ \text{sec} = (\text{Row Time})\ (25) + \text{VRTC}$$
$$= (\text{Row Time})\ (25) + N\ (\text{Row Time})$$

If N is selected to be 2, the following result is obtained:

$$\text{Row Time} = 6.17284 \times 10^{-4}\ \text{sec}$$

Therefore,

$$\text{VRTC} = (2)(\text{Row Time}) = 12.3457 \times 10^{-4}\ \text{sec}$$
$$= 1.23457\ \text{ms}$$

Since the Vertical Blanking Time, nominally 900 μsec, falls within the constraints for the Vertical Drive Pulsewidth, the VRTC output from the 8275 may be used directly for the Vertical Drive Pulse. The 8275 will be programmed for a Vertical Retrace Row Count of 2.

In order to calculate the Horizontal Retrace Character Count, it is necessary to consider the row format as defined in the specifications. Figure 4-5 shows three adjacent characters in a row. The row, as shown, is composed of 10 Lines/Row and 7 Dots/Line/Character. Given that the Row Time is 617.284 μsec, the Total Line Time may be calculated as follows:

$$\text{Total Line Time} = \frac{\text{Row Time}}{\#\ \text{Lines/Row}}$$
$$= \frac{617.284 \times 10^{-6}\text{sec}}{10}$$
$$= 61.7284 \times 10^{-6}\ \text{sec}$$
$$= 61.7284\ \mu\text{sec}$$

The Total Line Time is composed of the display portion of the line plus the Horizontal Blanking Time (HRTC).

$$\text{Total Line Time} = 61.7284 \times 10^{-6}\ \text{sec}$$
$$= 80\left(\frac{\text{Character Time}}{\text{line}}\right) + \text{HRTC}$$

Horizontal Blanking Time (HRTC) must be an integral number of Character Times/Line.

Then

$$61.7284 \times 10^{-6}\ \text{sec} = 80\left(\frac{\text{Character Time}}{\text{line}}\right) + M\left(\frac{\text{Character Time}}{\text{line}}\right)$$

If M is selected to be 20, the following result is obtained:

$$\left(\frac{\text{Character Time}}{\text{line}}\right) = \frac{61.7284 \times 10^{-6}}{80 + 20}$$
$$= 6.1728 \times 10^{-7}\ \text{sec}$$
$$= 617.284\ \text{ns}$$

This value defines the period of the 8275 character clock.

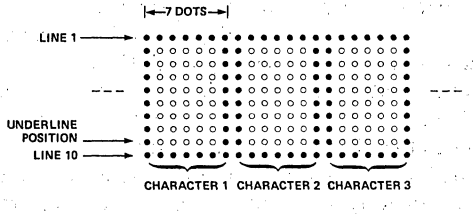


Figure 4-5. Row Format

The Horizontal Blanking Time (HRTC) is calculated as follows:

$$\text{HRTC} = 20\,(617.284\text{ ns})$$
$$= 12.3456\ \mu\text{sec (nominal value 11 }\mu\text{sec)}$$

The 8275 will be programmed for a Horizontal Retrace Character Count of 20. Since the specifications call for a Horizontal Drive Pulsewidth of 25–30 μsec, an external oneshot is required. The oneshot is triggered by the leading edge of HRTC.

Using the value for the Character Time/Line, the Dot Clock Rate may be established. It should be noted that the clock is used to shift data from the parallel in-serial out shift register (contained in the dot timing logic) to the CRT video input. The system character clock is also derived from the Dot Clock.

The dot clock is calculated as follows:

$$\left(\frac{\text{Dot Time}}{\text{line}}\right) = \frac{\left(\frac{\text{Character Time}}{\text{line}}\right)}{\#\text{ dots/character}}$$
$$= \frac{6.17284 \times 10^{-7}}{7}\text{ sec}$$
$$= 8.8183 \times 10^{-8}\text{ sec}$$
$$= 88.183\text{ ns}$$

$$\text{Dot Clock Frequency} = \frac{1}{\frac{\text{Dot Time}}{\text{Line}}} = 11.34\text{ MHz}$$

The Horizontal Oscillator Repetition Rate may be calculated as follows:

$$f_{\text{Horiz}} = \frac{1}{\text{Total Line Time}} = \frac{1}{61.7284 \times 10^{-6}\text{ sec}}$$
$$= 16{,}200\text{ Hz}$$

This value falls within the system specification of 15,750 ±500 pps.

4.3.4 Dot Timing Logic

The primary function of the dot timing logic, illustrated in Figure 4-6, is to transfer the output of the character generator ROM to the video input of the CRT. Due to the high data transfer rate (11.34 MHz), logic external to the 8275 is required for this function. The data transfer operation is accomplished as follows: The character generator ROM output is applied to the parallel input lines of the 74166 shift register, the shift register is loaded synchronously with respect to the positive-going edge of the character clock, and data is clocked out of the 74166 serial input at the dot clock frequency. The 74166 output is applied, through appropriate gating logic, to the CRT video input. In addition to the previously described functions, the dot timing logic provides the timing signals required for transferring characters from the 8275 character code and line count outputs to the character generator ROM, implements the video suppress and light enable gating functions, and generates the system dot and character clocks.

In order to understand the dot timing logic design process, it is necessary to refer to Figure 4-6 and Figure 4-7.

It can be seen from the timing waveforms of Figure 4-7 that the character code output from the 8275 will be valid 150 ns (worst case) after the negative-going edge of the character clock. The character generator ROM output will be valid, assuming a direct connection between the 8275 and the ROM, 450 ns (worst case) after the character code appears at the address inputs. Total delay from the negative-going edge of the character clock until ROM output data becomes available is then 600 ns. Given the character clock width of 617 ns and external logic propagation delays and setup times, it becomes difficult to latch the ROM output for the first display character during the first character clock period. In order to alleviate this situation, a data pipelining technique is utilized. The timing for this technique is shown in Figure 4-7. A latch, introduced between the 8275 and the character generator ROM as shown in Figure 4-6, samples character code and line count data from the 8275 1/2 dot clock (45 ns) after the positive-going edge of the character clock. Data from the latch is applied to the character generator ROM address lines yielding, after a 450 ns delay (worst case), the appropriate 7-bit code at the ROM output. ROM data is loaded into the 74166 shift register on the next positive-going edge of the character clock. This technique effectively delays the video output from the shift register by 1½ character clocks, but eliminates the difficulties in sampling the ROM data within the first character clock period. Due to the video delay associated with this technique, it is also necessary to delay all signals affecting the video output and CRT timing. These signals include HRTC, VRTC, VSP, and

LTEN. The delay is accomplished using a two-stage shift register constructed with edge triggered D flip-flops (74175). The system dot clock (11.34 MHz) is obtained by dividing the 22.68 MHz output from the 8224 clock generator by two. The dot clock is utilized to clock the 74166 output shift register and is divided by 7, using a 74S163 counter, to produce the system character clock. It should be noted that the use of a bipolar character generator PROM such as the Intel® 3604 or 3608 will reduce the external dot timing logic package count due to the reduced access time.

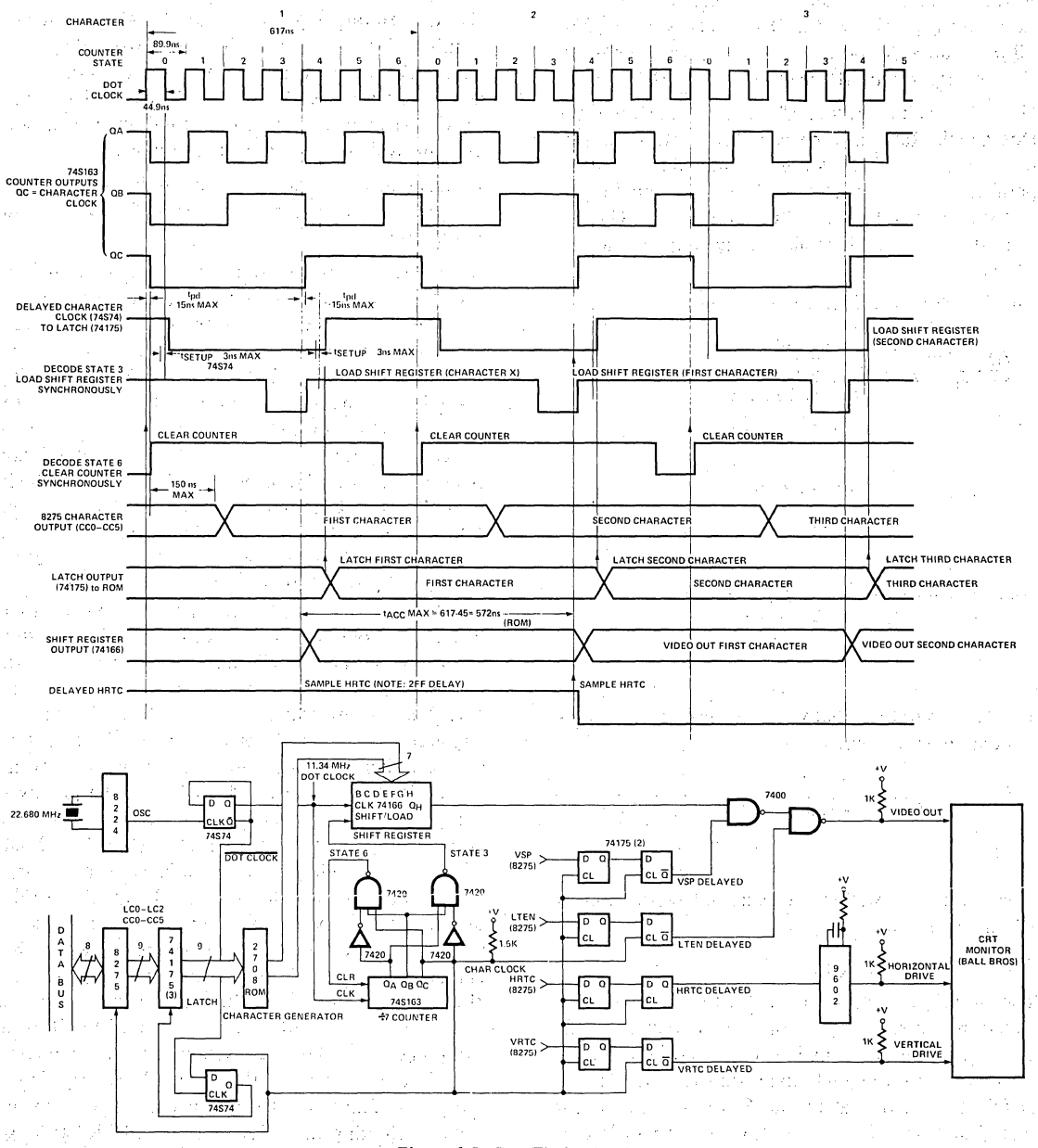


Figure 4-6. Dot Timing Logic

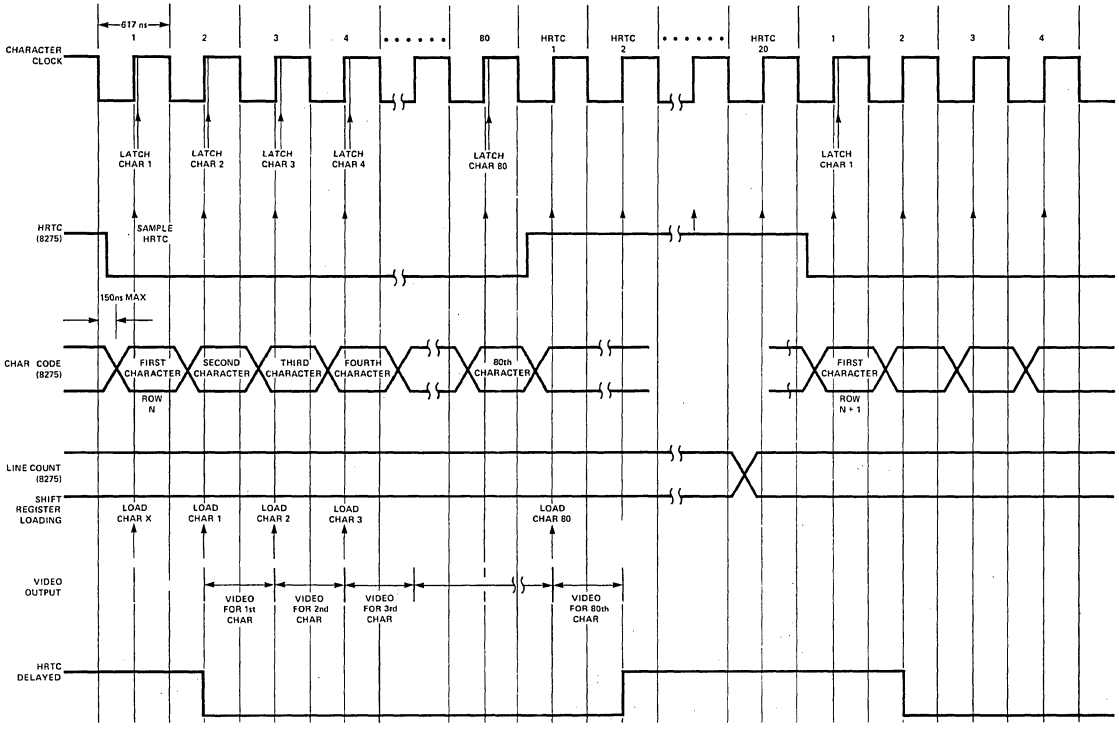


Figure 4-7. CRT System Timing

4.3.5 Keyboard Interface Design

The keyboard interface, Figure 4-8, consists of the 8279 Keyboard Controller and the decoding logic necessary for scanning the keyboard matrix. The 8279 SL_0–SL_2 output lines are decoded by the 74S138 decoder. The eight output lines from the decoder select 1 of 8 keyboard matrix rows for testing by the 8279. The keyboard matrix column output lines are connected to the 8279 return lines, RL_0–RL_7. Open collector outputs presented by individual keys within the matrix eliminate the need for isolation diodes when two keys in a given column are depressed. Two-key rollover was chosen as the operating mode for the 8279.

4.3.6 System Memory Design

The system memory, illustrated in Figure 4-9, consists of one 2716 EPROM used for program storage and four 2114 RAMs used for display memory, buffer memory, and system stack. The 2114 4K static RAM was chosen for the design because of its 1K X 4 organization, ease of use, and availability. Buffering between RAM memory and the system data bus was used to minimize bus loading.

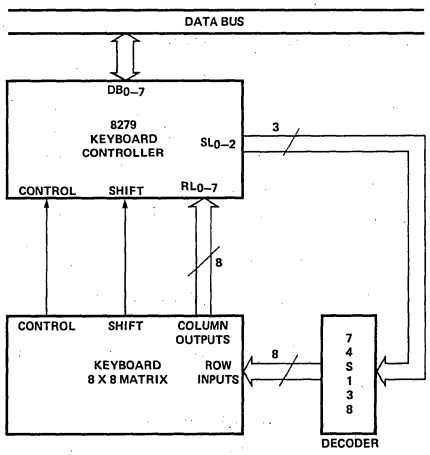


Figure 4-8. Keyboard Interface

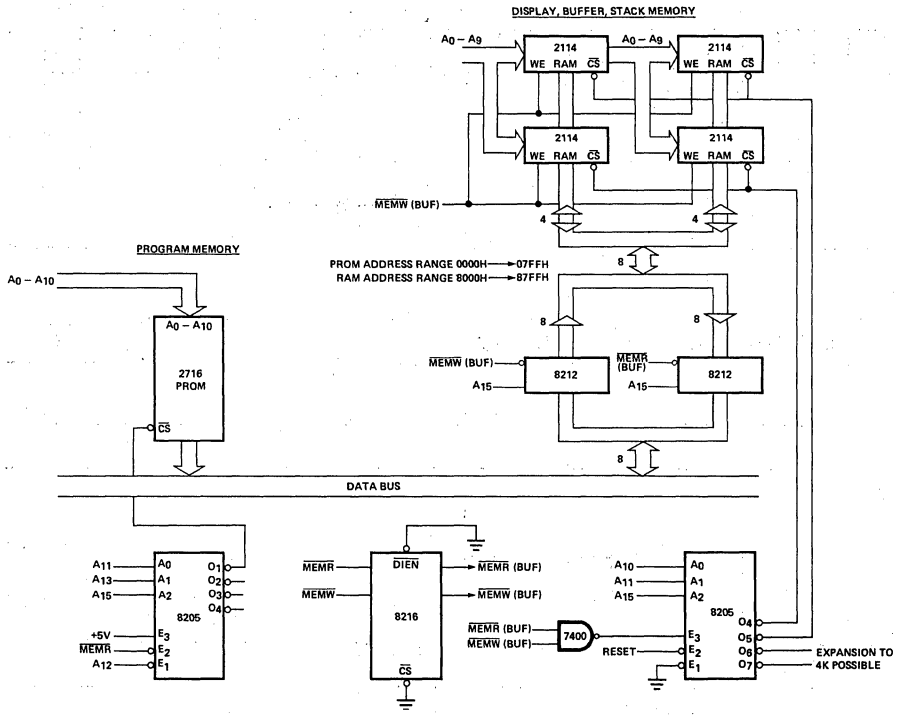


Figure 4-9. System Memory

4.4 SYSTEM SOFTWARE DESIGN

4.4.1 General Considerations

The approach taken in presenting the system software design is as follows: First, the software development process will be outlined. A discussion of system software operation will then be undertaken. Software operation will be followed by a detailed presentation of system subroutines.

4.4.2 Software Development

Software development was accomplished using the following tools:

1. Intel® MDS microcomputer development system
2. Intel® dual floppy disc system
3. Intel® ICE-80 In-Circuit Emulator
4. Intel® ISIS II disc operating system

The MDS was utilized in conjunction with the dual floppy disc system for program editing, assembly, relocation, and loading functions.

The ICE module was used extensively for loading assembled routines into the prototype system RAM and debugging program errors. While in the emulation mode, the ICE processor controlled the operation of the CRT system. During debugging, emulation proceeded normally until certain user specified break conditions occurred, at which time ICE entered the interrogation mode. During interrogation mode all processor functions, including DMA, ceased, allowing the user to access and display CPU register contents, status, and up to 44 previous machine cycles, system memory contents, and I/O device data.

4.4.3 Operation

The fundamental operations performed by the CRT system software are presented in Figure 4-10. Extensive use of subroutines in implementing major software functions resulted in readily understandable software. Debugging operations were also simplified as a result of the software structure. At

system reset, the central processor interrupt system is disabled, the program counter is set to zero, and peripheral reset functions are carried out. Following reset, the system software initializes all peripherals, clears buffer memory, initializes special buffer locations, fills display memory with space codes, and enables interrupts. The processor then loops until an interrupt arrives from the 8275 or 8251. When the processor detects the occurrence of an interrupt, the instruction being executed is completed, an RST 7 vector is placed on the system data bus, and the RST 7 call instruction is executed, forcing a jump to the starting address of the 8275/8251 interrupt polling routine. Once the polling routine establishes the source of the interrupt, program flow continues along one of the two possible paths shown in Figure 4-10. An 8275 interrupt causes the 8257 DMA Controller to be re-initialized, the 8279 Keyboard Controller to be serviced, and, if a key depression has occurred, a character to be transmitted to the terminal output. An interrupt from the 8251 will first cause the USART character to be read and stored in memory. The system software then examines the character to determine whether it is a displayable character, a control code, or the first or second character in an escape sequence. After determining the nature of the character, an appropriate subroutine is called. Following the completion of the routines associated with an 8275/8251 interrupt, interrupts are re-enabled and a return instruction executed. The CPU then loops until the receipt of an interrupt. In order to appreciate the operation of the system software in detail, it is necessary to consider the following items:

1. System memory organization.
2. The relationship between character position on the screen and screen pointers Row Count, Column Count, and memory pointer Top.
3. The relationship between memory pointers Row Count, Column Count and the 8275 cursor X and Y position registers.
4. Scrolling concepts, including the relation between scrolling, display memory, and the memory pointer Top.

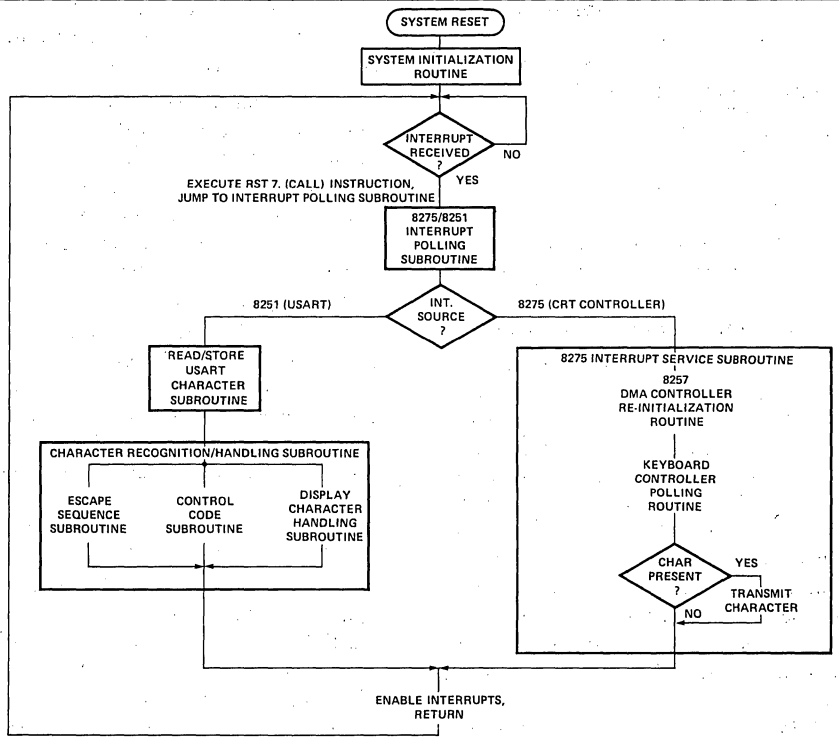


Figure 4-10. CRT Software Operations

System Memory Organization

System memory organization is shown in Figure 4-11. It should be noted that an additional 2K block of RAM was utilized for program memory (rather than PROM) during the software development/debug phase of system design.

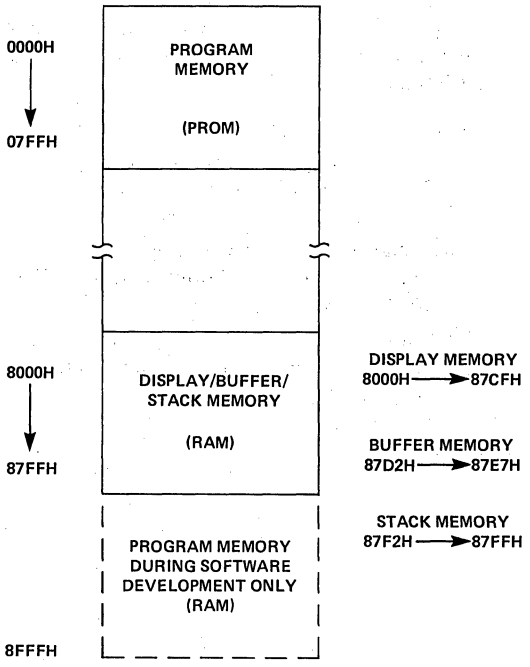


Figure 4-11. System Memory Organization

Character Position/Screen Pointer Relationships

To define the location of a character on the screen, two pointers, Row Count and Column Count, were created in memory. The relationship between character location on the screen and the two pointers is illustrated in Figure 4-12. Row Count and Column Count are stored in memory locations RCTAD and CCTAD, respectively. Row Count represents the position of the first character in a given row. For the first row, Row Count = 0000H. For the second row, Row Count = 0050H. Column Count represents the specific column in which the character is located. Character position on the screen may be calculated by adding the Row Count to the Column Count; e.g., the highlighted character in Figure 4-12 is located at AOH + 03H = A3H.

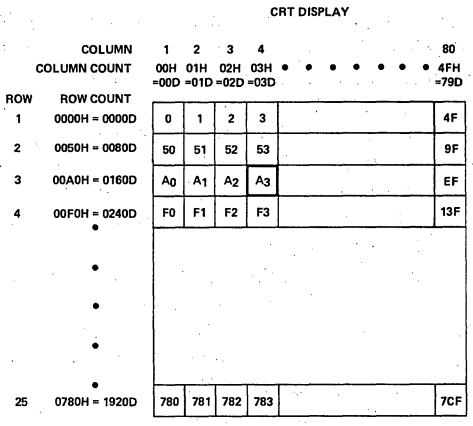


Figure 4-12. Character Location/Pointer Relationship

Memory Pointer/8275 Cursor Position Register Relationship

It was necessary to establish a relationship between Row Count and Column Count pointers and the 8275 Cursor X and Y Position registers for the cursor generated by the 8275 to be loaded at the appropriate position on the screen. This relationship is summarized in Table 4-2.

The value transferred to the 8275 for the Cursor X Position is identical to the Column Count. A new parameter, Cursor Y Position, stored at memory location CURSY, was also established. For a given Row Count value, a value for Cursor Y Position is defined. This value is transferred to the 8275 Cursor Y Position register.

It is necessary to introduce an additional parameter, Top, which will be used in conjunction with Row Count and Column Count to determine the location in display memory at which an incoming display character will be stored. The location at which a given character will be stored (assuming no more than 2000 characters have been entered since initialization) is calculated by adding TOP + Row Count + Column Count, where TOP is assumed to be 8000H, the starting location of display memory shown in Figure 4-11. Following system initialization, characters will be entered in display memory starting at memory location 8000H. The 2000th character will be entered at location 87CFH. Upon entering the 2001st character, a scrolling condition exists and TOP will be modified to point to memory address 8050H. An in-depth discussion of scrolling is presented in the next section.

Table 4-2

SCREEN POINTER/8275 CURSOR X,Y POSITION REGISTER RELATIONSHIP

ROW	ROW COUNT VALUE	CURSOR Y POSITION REGISTER VALUE	COLUMN	COLUMN COUNT VALUE	CURSOR X POSITION REGISTER VALUE
1	0000H	00H	1	00H	00H
2	0050H	01H	2	01H	01H
3	00A0H	02H	3	02H	02H
4	00F0H	03H	4	03H	03H
25	0780H = 1920D	18H = 24D	80	4FH = 79D	4FH = 79D

Scrolling

Scrolling is implemented in the CRT system design by shifting the entire display up by 1 row when a scrolling condition occurs. Scrolling will occur when certain cursor manipulation functions are exercised or when a character is entered in the last CRT display position, indicating a full memory page condition exists. Character entry will be used as the vehicle for explaining scrolling in the following discussion.

Characters are normally entered sequentially in display memory. When the 2000th character has been entered, display memory capacity has been attained; i.e., a full page condition exists. At this point, scrolling will take place. For scrolling to take place, DMA channel 2, the channel used to extract characters from display memory, must be re-initialized to the appropriate starting address and terminal count values. The memory pointer TOP will be used to establish the starting address for channel 2. Prior to scrolling, TOP = 8000H, the starting address of display memory. Each scrolling operation causes 80D (50H) to be added to TOP, moving the pointer, as shown in Figure 4-13b, to the beginning of the following row in display memory. It should be recalled that TOP, in conjunction with Row Count and Column Count determines the insertion address for incoming display characters. The net effect of modifying TOP is to shift the information being displayed on the CRT up by 1 row; i.e., scrolling is accomplished. Prior to scrolling, the terminal count value for DMA channel 2 is equal in magnitude to the display memory length −1 or 87CFH − 8000H. The actual value sent to the terminal count register is 87CFH − 8000H + 8000H. The addition of 8000H sets bit 14 in the terminal count register to a 1, indicating a DMA read operation. If scrolling is to be implemented, the terminal count value must be modified to 87CFH − TOP + 8000H. Characters transferred by channel 2 include those characters located from the address specified by TOP to the end of display memory. In order to transfer the characters from the beginning of display memory through the address immediately prior to TOP, the autoload feature of the 8257 DMA controller is utilized. When DMA channel 2 reaches terminal count, following the transfer of characters from TOP to the end of display memory, the starting address and terminal count parameters stored in the DMA channel 3 registers are loaded into channel 2. DMA operations resume in channel 2 using the channel 3 parameters. To accomplish the desired channel 3 operations, it is only necessary to re-initialize the channel 3 starting address to the beginning address of display memory, and the terminal count value to 87CFH, the maximum terminal count for a 2000-byte display memory space. These processes are performed during DMA re-initialization following an 8275 interrupt. New text entry following scrolling is illustrated in Figure 4-13. BOTTOM, a parameter corresponding to the address of the first character in the last row to be displayed, is utilized during clear to end of screen operations.

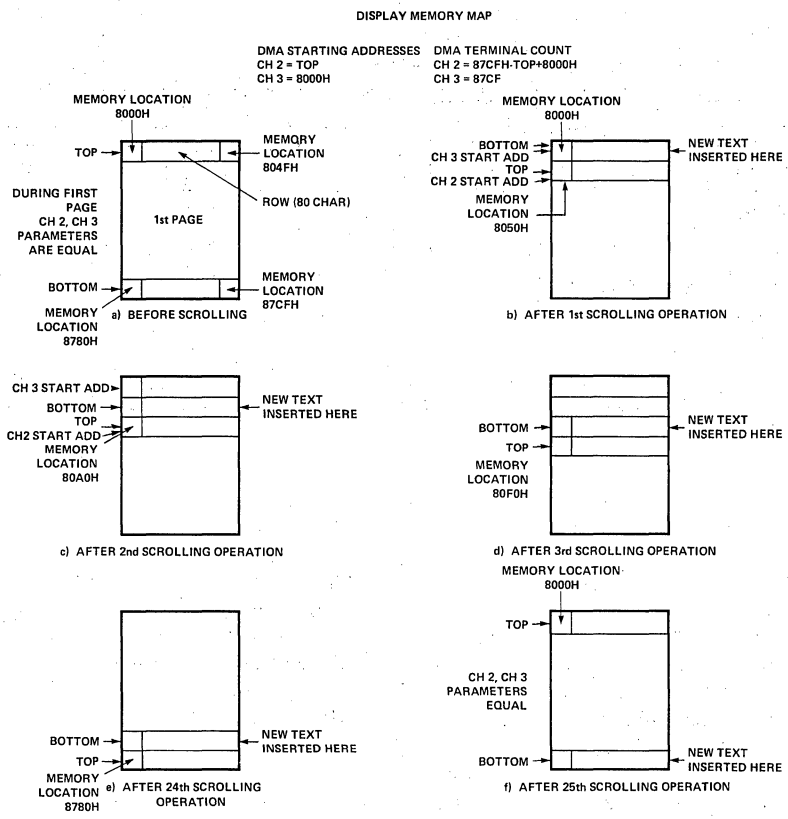


Figure 4-13. Pointer Manipulation During Scrolling

4.4.4 System Subroutines

System Initialization Routine (CRTGO)

The system initialization routine, Figure 4-14, establishes a starting point for system operation. The 8251 USART is initialized to transmit to and receive characters from an external device. The 8279 Keyboard Controller, at system reset, comes up in the two-key rollover mode. It is therefore only necessary to set up the Keyboard Controller internal operating frequency during initialization. Assuming a desired internal operating frequency of approximately 100 kHz and a 2.048 MHz system clock, the frequency divider chain is programmed to divide by 21. The 8275 initialization parameters are determined from the original CRT system specifications and vertical retrace Row Count/ Horizontal Retrace Character Count calculations previously performed. The delayed line number feature allows the use of only 3 line count outputs to determine which of 10 possible lines in a character row will be displayed. Given that the underline placement position is set to the ninth row, the top and bottom lines of the character are automatically blanked, leaving, effectively, 8 unique lines for display. The 8275 cursor position registers are initialized to zero, forcing the cursor to the upper left-hand corner of the display. The preset counters command resets all 8275 counters to zero and stops the 8275 counters until another command is issued. The 8275 is then started by a start display command. An interrupt will be generated from the 8275 approximately 15 ms later. Interrupts are enabled following the 8275 start command. Interrupts were disabled prior to this time to insure that the central processor did not react to erroneous interrupts from the 8275 generated prior to 8275 initialization. The processor, following initialization, waits in a loop until the arrival of an interrupt from the 8275 or 8251.

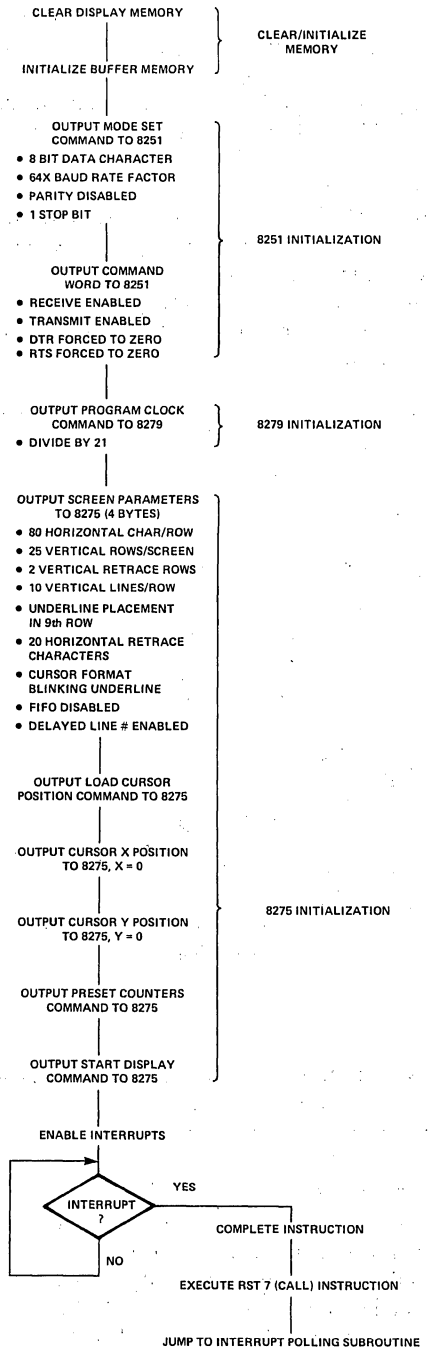


Figure 4-14. System Initialization Routines

Interrupt Polling Subroutine (Poll)

The interrupt polling subroutine, Figure 4-15, tests to determine the source of the interrupt. If the interrupt originated with the 8275, the 8275 interrupt service subroutine is called. Following completion of the subroutine, interrupts are re-enabled, and a return executed. An interrupt issued from the 8251 forces subroutine calls to the read/store USART character subroutine and the character recognition/handling subroutine. Interrupts are re-enabled at the completion of the character recognition/handling routine. A return operation follows.

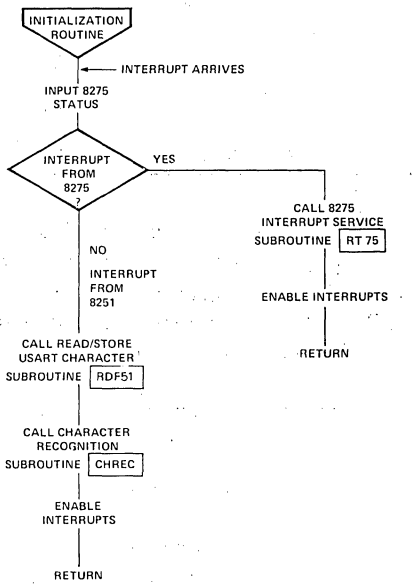


Figure 4-15. Interrupt Polling Subroutine (POLL)

8275 Interrupt Service Subroutine (RT 75)

The 8275 interrupt service subroutine, Figure 4-16, re-initializes the 8257 DMA Controller, then tests the 8279 FIFO status. If a character has been transmitted from the keyboard to the Keyboard Controller, a table lookup operation is performed to obtain the correct ASCII code for the character, and the character is transmitted.

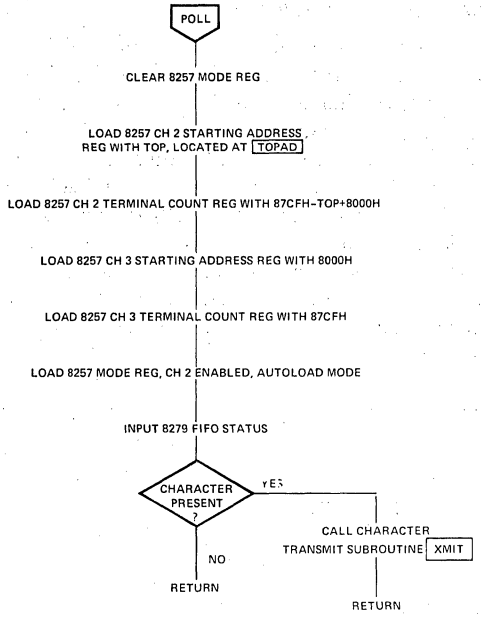


Figure 4-16. 8275 Interrupt Service Subroutine (RT75)

USART Read/Store Subroutine (RDF 51)

The read/store USART character subroutine, Figure 4-17, moves a character from the USART to the CPU, masks off the upper-most bit, and stores the character in system buffer memory.

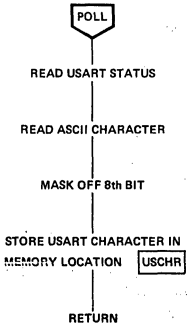


Figure 4-17. READ/STORE USART Character Subroutine (RDF51)

Character Recognition/Handling Subroutine (CHREC)

The character recognition/handling subroutine, Figure 4-18, examines the masked USART character to determine whether the character is a displayable character, control code, or the first or second character in an escape sequence. A call to the appropriate subroutine follows the decision-making process. If the character is the first character in an escape sequence, the escape sequence flag is set and the processor loops until a second character is received. The character immediately following the ESC character is examined by the escape code handling subroutine and a jump to an escape code routine follows. If the character is a displayable character or control code, the appropriate subroutine is called.

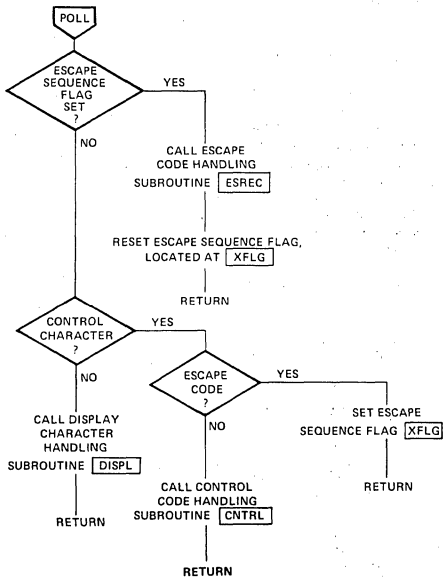


Figure 4-18. Character Recognition/Handling Subroutine (CHREC)

Escape Sequence Subroutine (ESREC)

The escape sequence subroutine, Figure 4-19, performs a masking operation on the USART character, shifts the result by one bit position, and adds this value to the base address of the escape sequence lookup table, BSETI. The lookup table contains starting addresses for each of the escape sequence routines. This address is jammed into the program counter and the routine executed. A summary of escape sequence functions is given in Appendix 5.2.

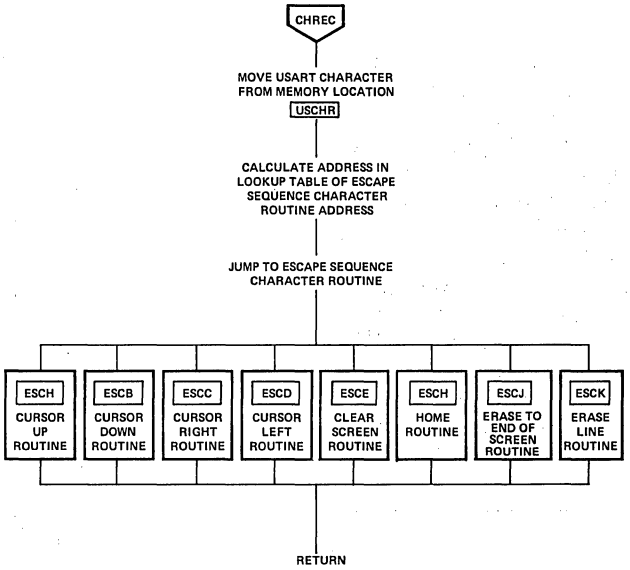


Figure 4-19. Escape Sequence Subroutine (ESREC)

Control Code Subroutine (CNTRL)

The control code subroutine, Figure 4-20, involves, conceptually, the same procedures executed by the escape sequence subroutine. A summary of control code functions is given in Appendix 5.2.

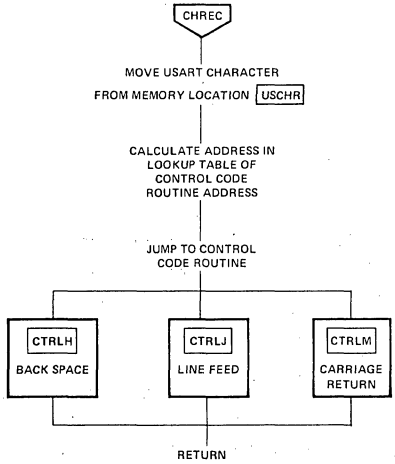


Figure 4-20. Control Code Subroutine (CNTRL)

Display Character Handling Subroutine (DISPL)

The display character handling subroutine, Figure 4-21, determines if the cursor is located in the last column of the row, the last display position, or elsewhere and calls the appropriate subroutines.

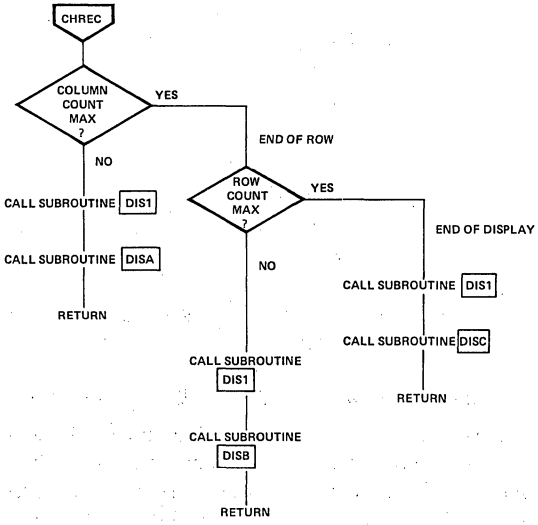


Figure 4-21. Display Character Handling Subroutine (DISPL)

Display Subroutine One (DIS1)

Display subroutine one, Figure 4-22, calculates the location in memory at which the display character is to be inserted. If the location calculation results in an address outside of the display memory bounds, appropriate compensation action is taken. Prior to inserting the display character in memory, the first character position in the row in which the character will be located is examined. If an End of Row character (EOR) is found, the row in question will be blanked by the 8275. It is necessary to clear the row by filling it with space codes (Fill Subroutine), then insert the display character in the desired location. If no EOR character is found, insertion proceeds without further software intervention.

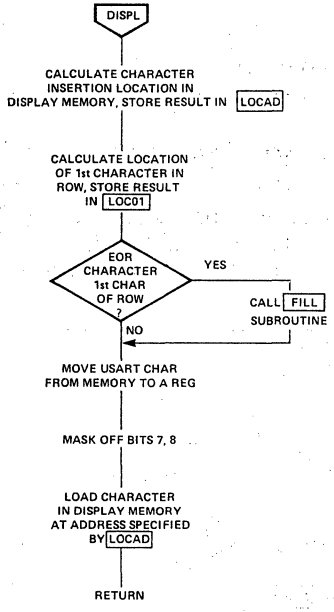


Figure 4-22. Display Subroutine 1 (DIS1)

Display Subroutines A, B, C (DISA, DISB, DISC)

Display subroutines A, B, and C, Figure 4-23, modify the appropriate display memory pointers. The modifications are based on the present cursor location, as determined by subroutine DISPL. The resulting cursor position data is transferred to the 8275 Cursor X and Y Position registers. If DISC is called, a scrolling operation occurs.

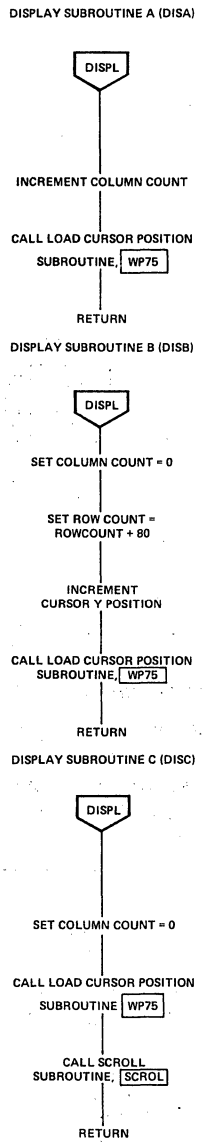


Figure 4-23. Display Subroutines – A (DISA), B (DISB), C (DISC)

Cursor Up Routine (ESCA)

The cursor up routine, Figure 4-24, determines if the cursor is located in the first display row. If it is, the Row Count and Column Count values are modified, and the cursor is moved to the last display row with no change in X position. If the cursor is not in the top row, the row up subroutine is called.

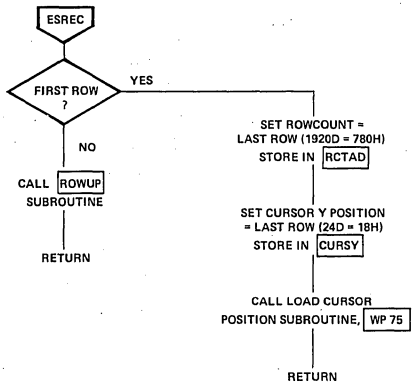


Figure 4-24. Cursor Up Routine (ESCA)

Cursor Down Routine (ESCB)

The cursor down routine, Figure 4-25, determines if the cursor is located in the last display row. If it is, the scroll subroutine is called. No modification of cursor position is called for. If the cursor is not located in the last display row, the row down subroutine is called.

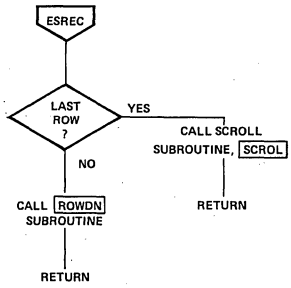


Figure 4-25. Cursor Down Routine (ESCB)

Cursor Right Routine (ESCC)

The cursor right routine tests the cursor location and moves the cursor as described in Figure 4-26. If the cursor is in the last display position, a scrolling operation occurs. 8275 Cursor X and Y Position registers are updated accordingly.

Cursor Left Routine (ESCD)

The cursor left routine tests the cursor location and moves the cursor as described in Figure 4-27.

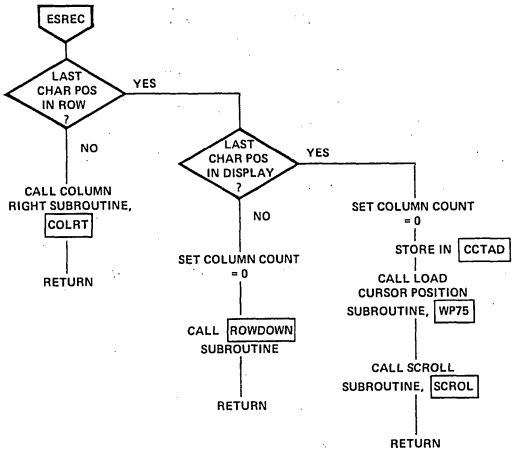


Figure 4-26. Cursor Right Routine (ESCC)

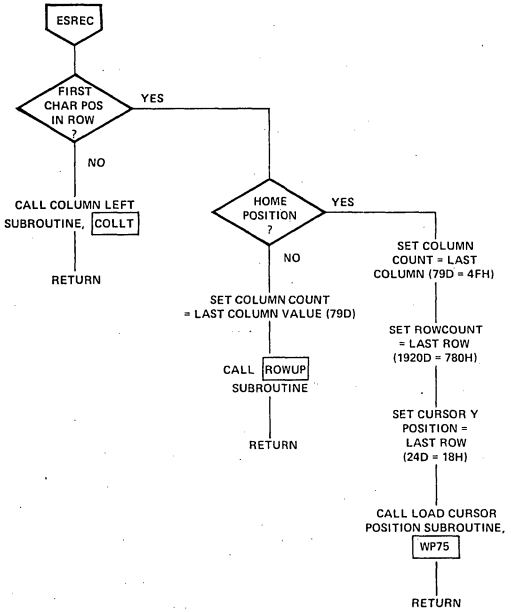


Figure 4-27. Cursor Left Routine (ESCD)

Clear Screen Routine (ESCE)

Several possibilities existed for implementing the clear screen function. The simplest of these techniques involves filling the display memory with space codes. This technique, although conceptually simple, requires several milliseconds to implement.

The End-of-Row character (EOR) recognized by the 8275 allows the clear screen feature to be executed in a considerably shorter time span. During the clear screen routine, Figure 4-28, EOR characters are placed in the first character position of each row in display memory. Since the EOR character blanks the entire display row when placed in the first character position of the row, the use of EOR characters in each row blanks the entire screen. All pointers are cleared during the clear screen operation.

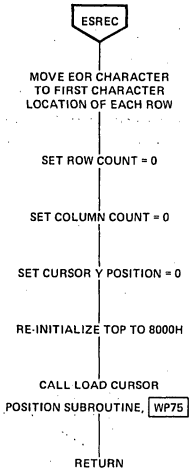


Figure 4-28. Clear Screen Routine (ESCE)

Home Routine (ESCH)

The home routine, Figure 4-29, resets the Row Count, Column Count and Cursor Y Position buffers to zero, but does not affect the value of TOP.

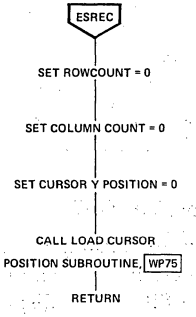


Figure 4-29. Home Routine (ESCH)

Erase to End of Screen Routine (ESCJ)

The erase to end of screen routine, Figure 4-30, inserts End of Row characters (EOR) in display memory in the same fashion as the clear screen routine. The fundamental difference between the routines is that the erase to end of screen routine must insert EOR characters selectively. Only rows from the present display row until the last display row, pointed to by BOTTOM, receive EOR characters. It should be noted that the pointer BOTTOM changes dynamically with scrolling operations.

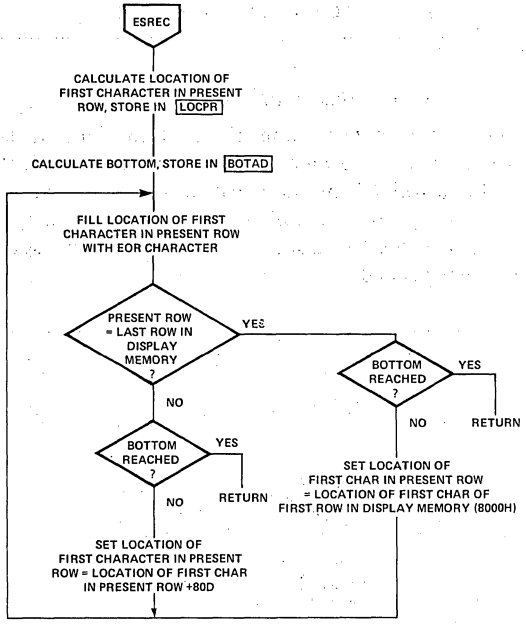


Figure 4-30. Erase to End of Screen Routine (ESCJ)

Erase Line Routine (ESCK)

The erase line routine, Figure 4-31, calculates the location of the first character in the current display row, stores the location in buffer memory, and calls the fill subroutine, which fills the row with space codes.

Backspace Routine (CTRLH)

See cursor left routine.

Line Feed Routine (CTRLJ)

See cursor down routine.

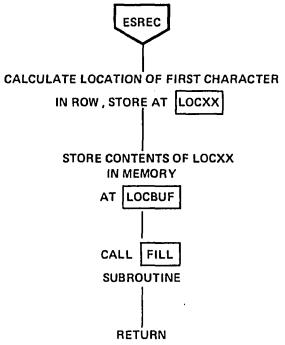


Figure 4-31. Erase Line Routine (ESCK)

Carriage Return Routine (CTRLM)

The carriage return routine, Figure 4-32, clears the column count and updates the 8275 cursor position registers.

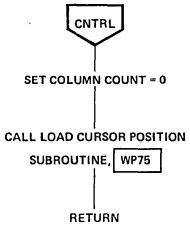


Figure 4-32. Carriage Return Routine (CTRLM)

Row Up, Row Down Subroutines (ROW UP, ROW DOWN)

The row up subroutine, Figure 4-33, subtracts 80D from the Row Count value, decrements the Cursor Y Position pointer, and updates the 8275 Cursor Position registers. The row down subroutine, Figure 4-34, differs in that 80D is added to Row Count.

Column Right, Column Left Subroutines (COLRT, COLLT)

The column right subroutine, Figure 4-35, increments the Column Count pointer and updates the 8275 cursor position registers. The column left subroutine, Figure 4-36, differs in that the Column Count is decremented.

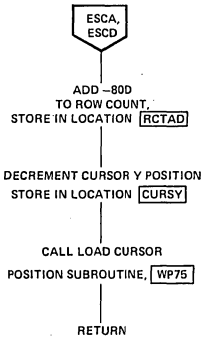


Figure 4-33. Row Up Subroutine (ROWUP)

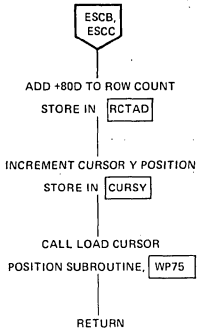


Figure 4-34. Row Down Subroutine (ROWDN)

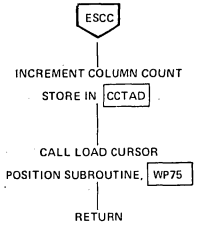


Figure 4-35. Column Right Subroutine (COLRT)

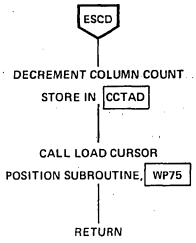


Figure 4-36. Column Left Subroutine (COLLT)

Scroll Subroutine (SCROL)

The scroll subroutine, Figure 4-37, fills the row in display memory pointed to by TOP with space characters via the fill subroutine, then modifies the value of TOP. TOP is utilized by the 8275 service subroutine in re-initializing the 8257 DMA controller.

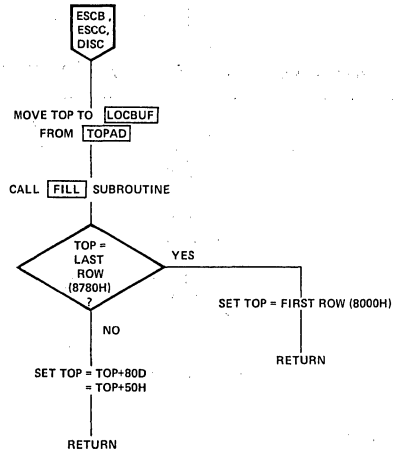


Figure 4-37. Scroll Subroutine (SCROL)

Fill Subroutine (FILL)

The fill subroutine, Figure 4-38, calculates the location of the last character in the current display row, plus one character position, by adding 80D = 50H to the location of the first character in the current display row. The current stack pointer value is saved, then the stack pointer is loaded with the location of the last character in the current display row, plus one character position. The B and C registers of the CPU are loaded with space characters and 40 PUSH B operations performed. This technique provides a rapid means (275 μsec) of filling a given row with space codes.

Load Cursor Position Subroutine (WP 75)

The load cursor position subroutine, Figure 4-39, transfers the contents of the Column Count and cursor Y position pointers to the 8275 cursor X position and cursor Y position registers, respectively.

The relationship between system subroutines is presented in Appendix 5.3. Software timing considerations are covered in Appendix 5.4.

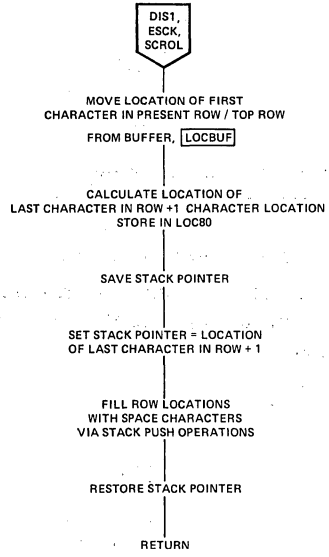


Figure 4-38. Fill Subroutine (FILL)

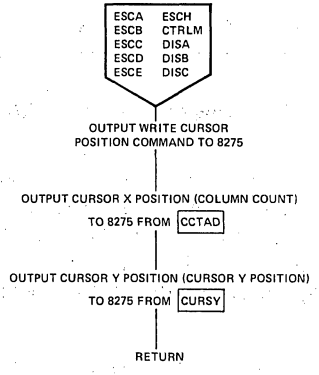
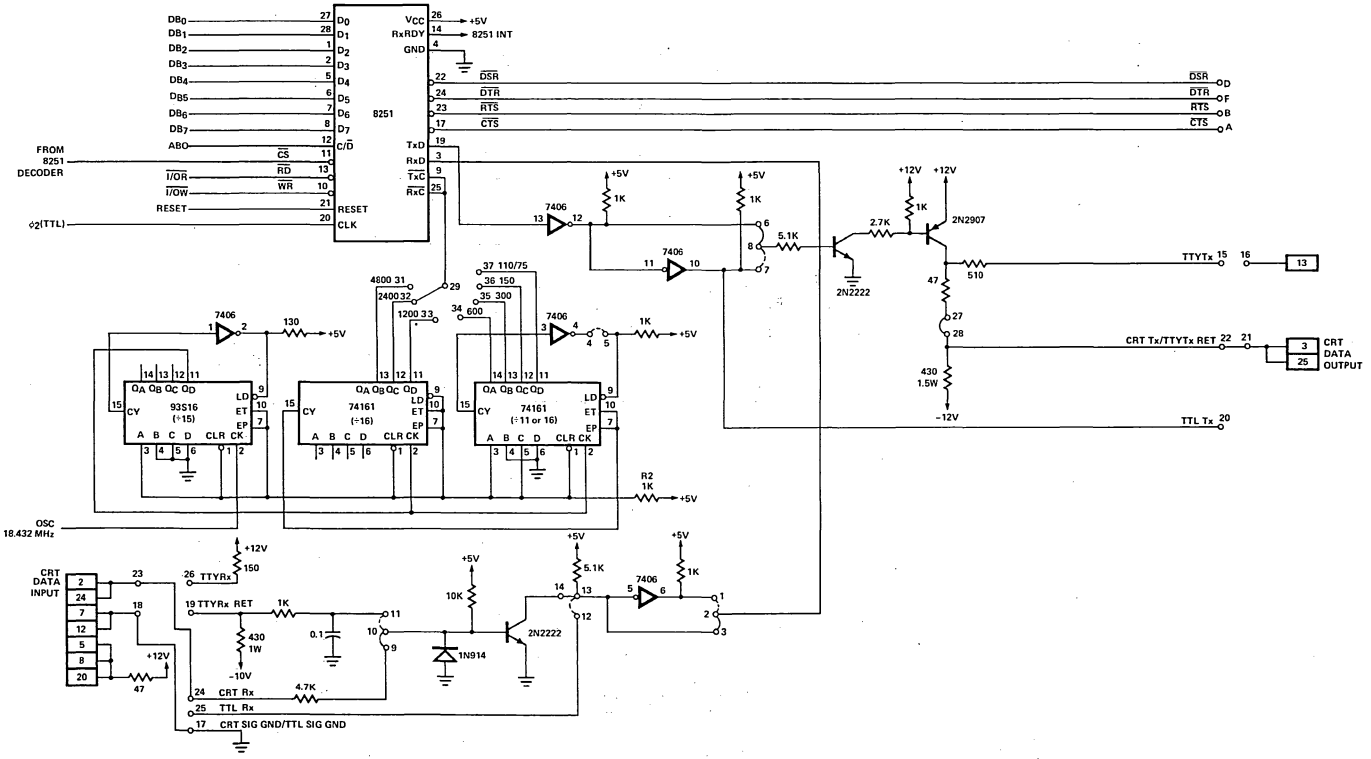
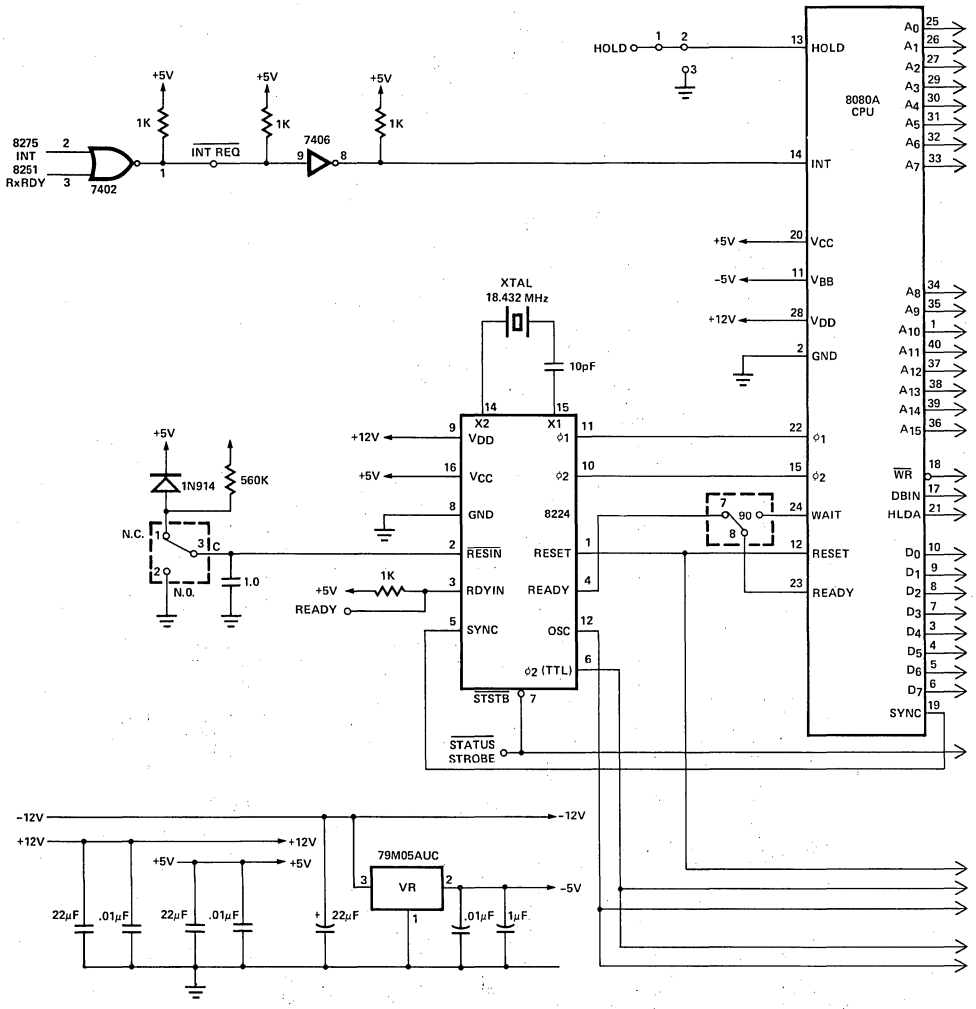
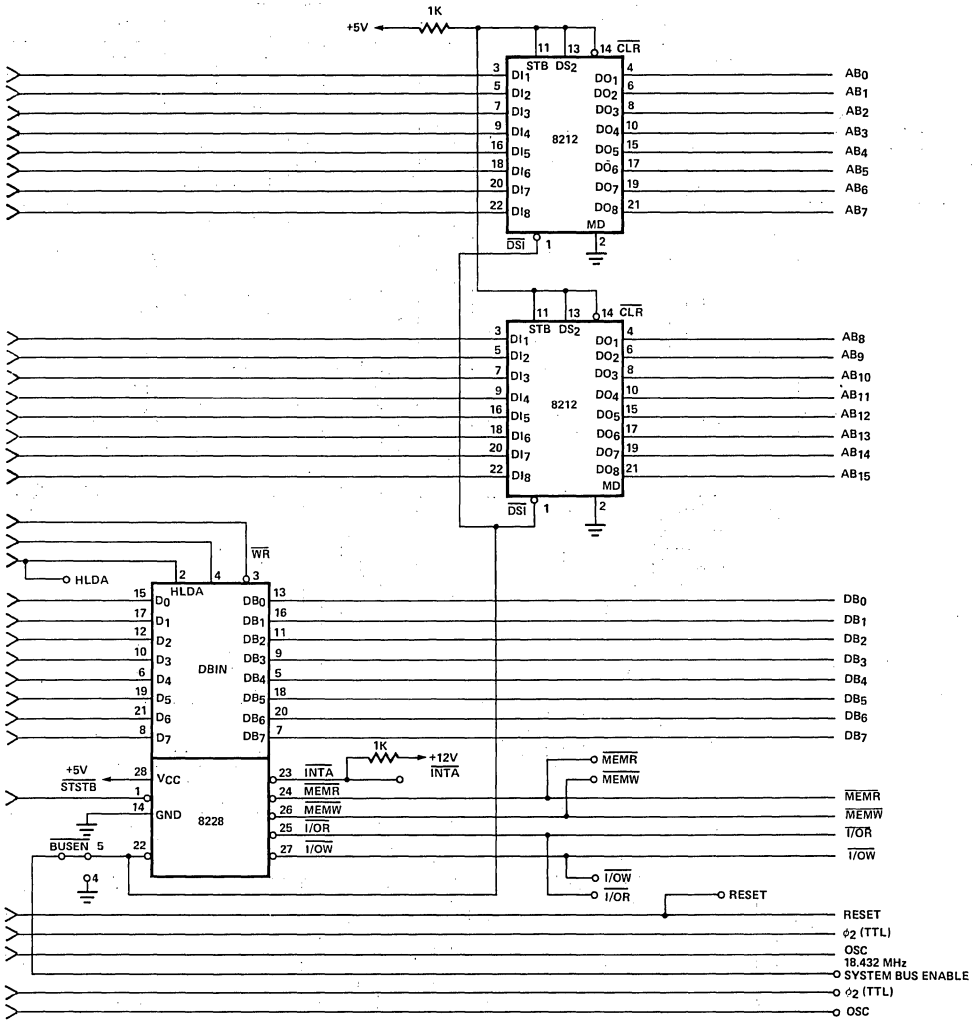


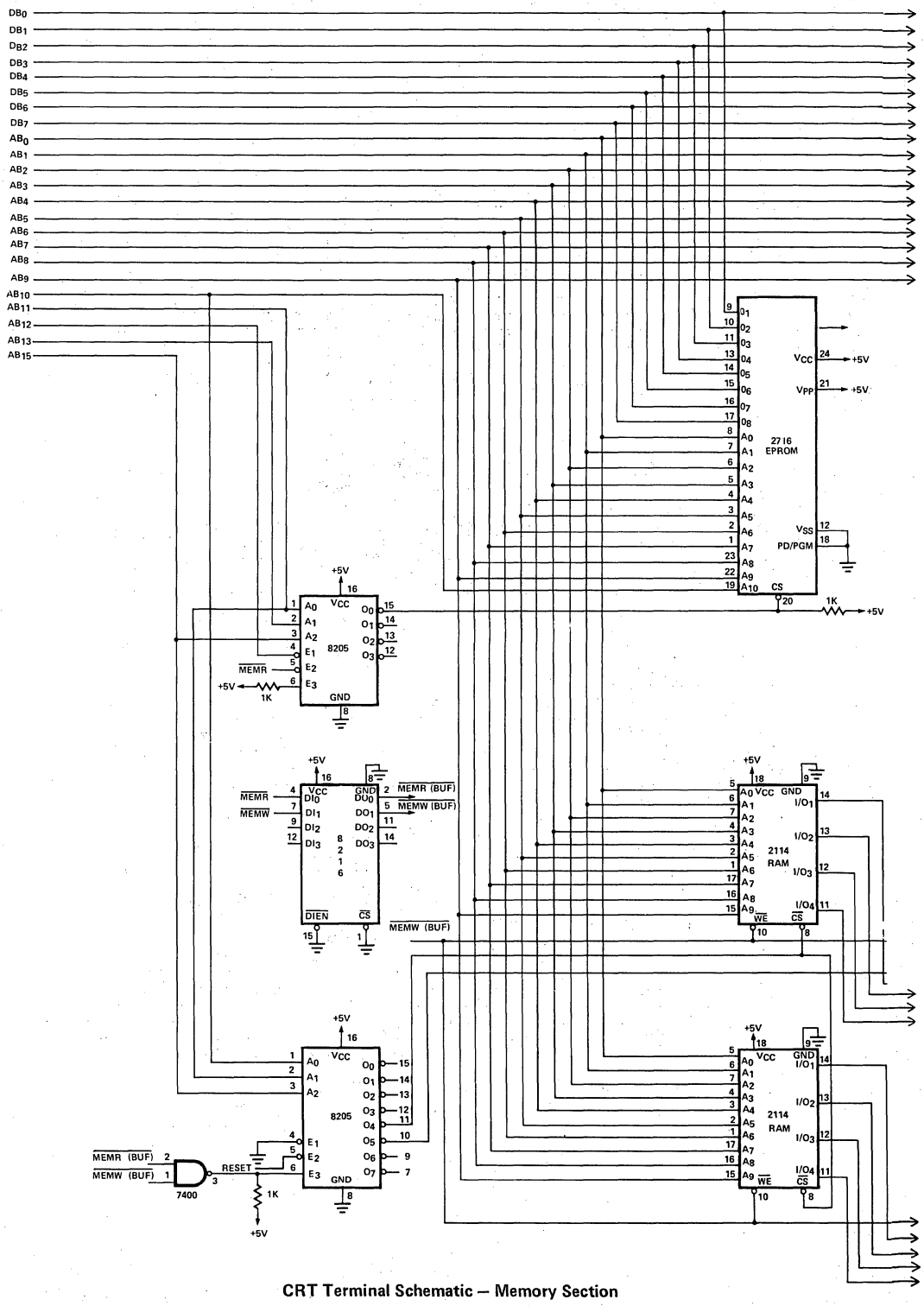
Figure 4-39. Load Cursor Position Subroutine (WP75)


8251
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
A0
FROM 8251 DECODER
CS
I/OR
I/OW
RD
WR
RESET
CLK
C/D
Vcc
RxRDY
GND
+5V
8251 INT
TxD
RxD
TxC
RxC
DSR
DTR
RTS
CTS
7406
1K
5.1K
2.7K
2N2907
2N2222
510
47
27
28
430
1.5W
-12V
+12V
TTYTx
CRT Tx/TTYTx RET
CRT DATA OUTPUT
TTL Tx
93S16
(÷15)
74161
(÷16)
(÷11 or 16)
4800 31
2400 32
1200 33
37 110/75
36 150
35 300
34 600
130
R2
OSC
18.432 MHz
CRT DATA INPUT
TTYRx
TTYRx RET
150
430
1W
-10V
0.1
10K
1N914
4.7K
CRT Rx
TTL Rx
CRT SIG GND/TTL SIG GND

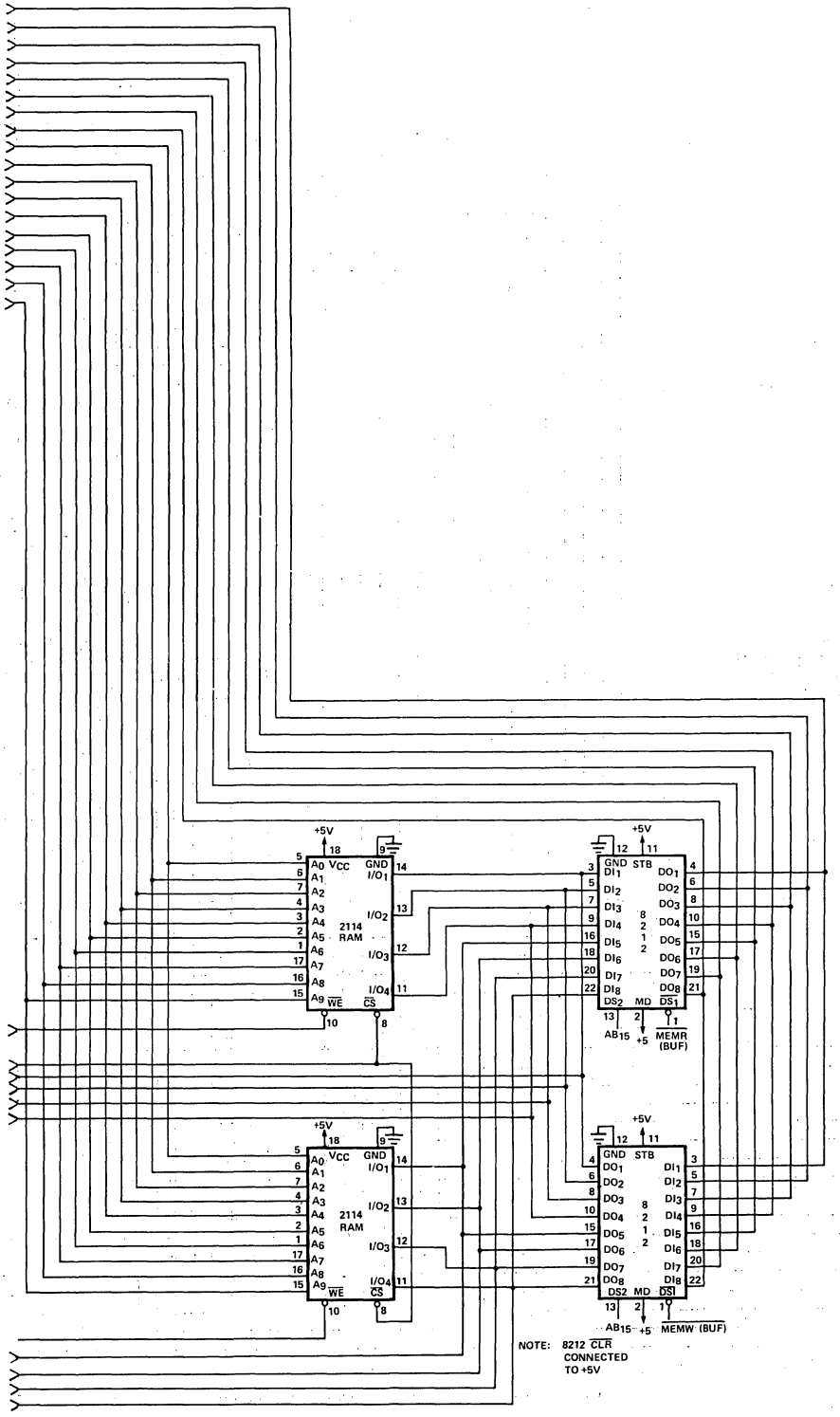


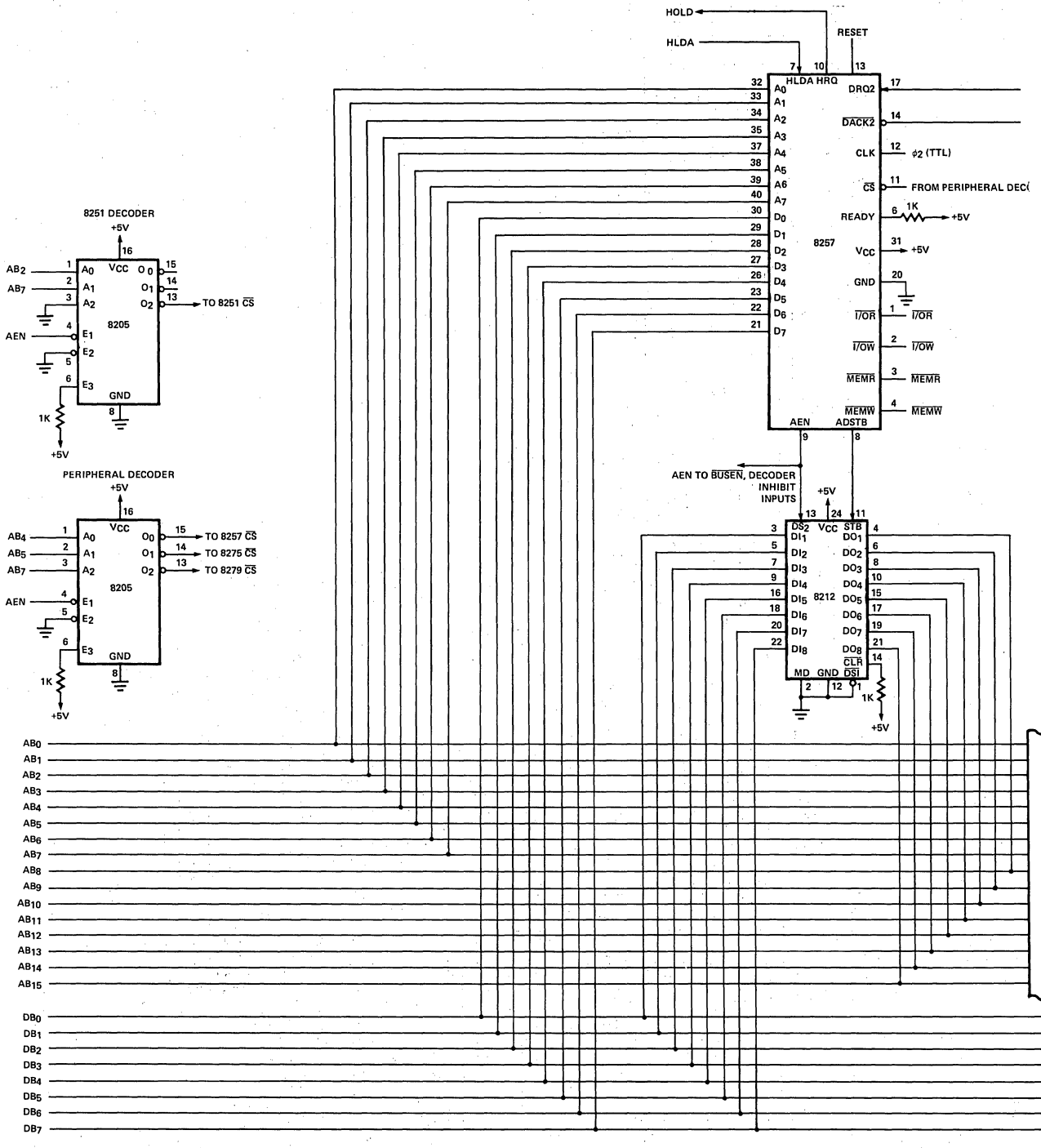
CRT Terminal Schematic — CPU Section


+5V
1K
8212
STB
DS2
CLR
DI1
DI2
DI3
DI4
DI5
DI6
DI7
DI8
DO1
DO2
DO3
DO4
DO5
DO6
DO7
DO8
MD
DSI
AB0
AB1
AB2
AB3
AB4
AB5
AB6
AB7
AB8
AB9
AB10
AB11
AB12
AB13
AB14
AB15
WR
HLDA
DBIN
D0
D1
D2
D3
D4
D5
D6
D7
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
8228
VCC
GND
STSTB
BUSEN
INTA
MEMR
MEMW
I/OR
I/OW
+12V
RESET
ϕ2 (TTL)
OSC
18.432 MHz
SYSTEM BUS ENABLE

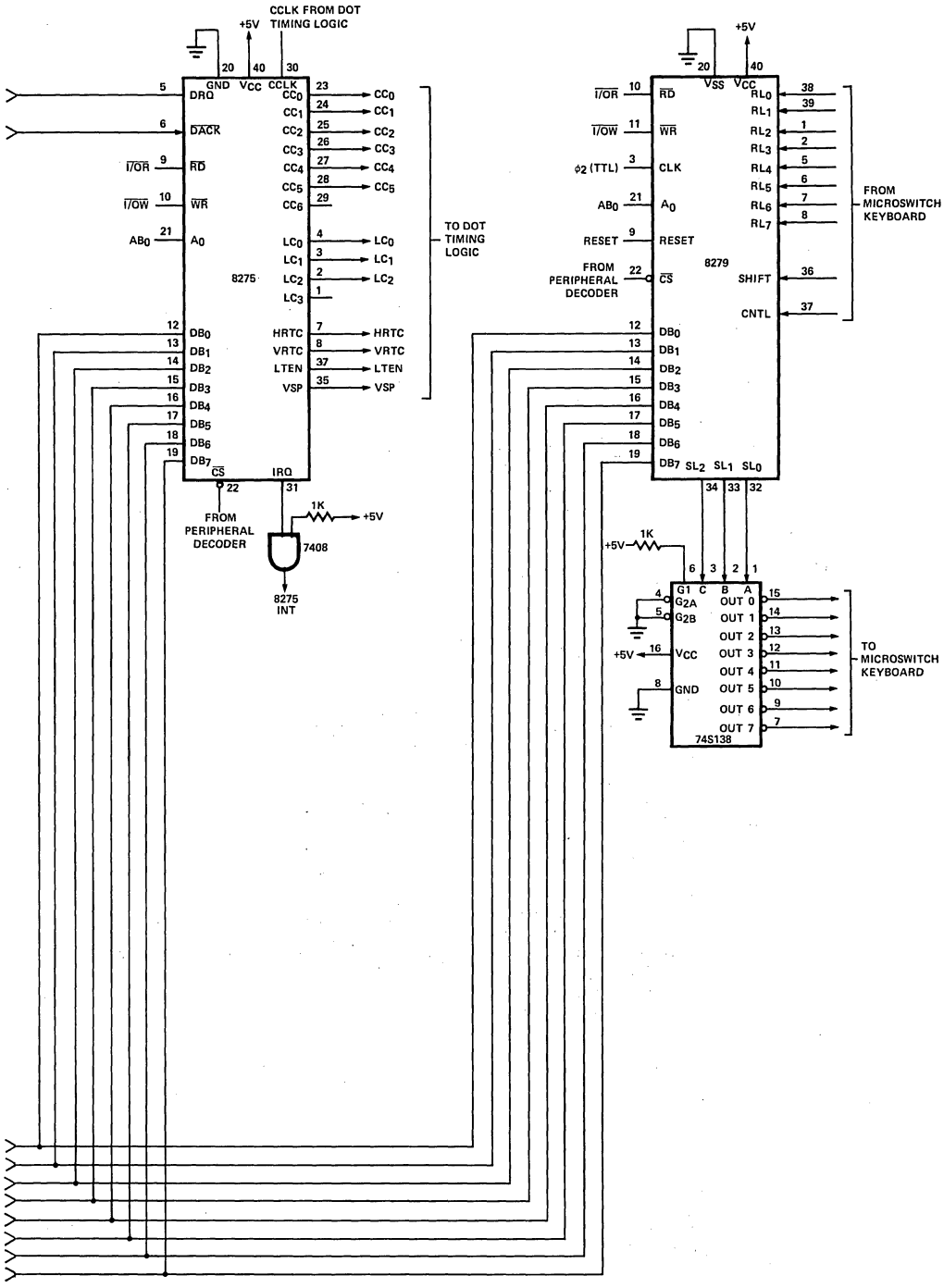


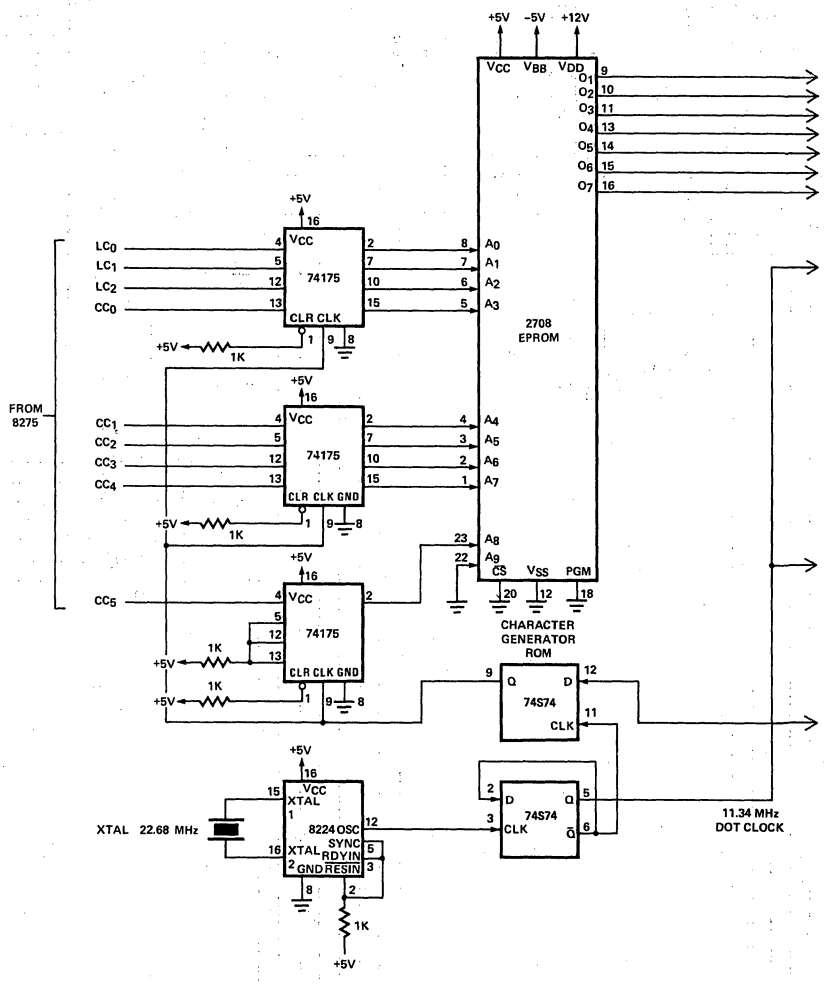
CRT Terminal Schematic — Memory Section


+5V
2114
RAM
A0 VCC GND
A1 I/O1
A2
A3 I/O2
A4
A5
A6 I/O3
A7
A8
A9 WE CS I/O4
8212
GND STB
DI1 DO1
DI2 DO2
DI3 DO3
DI4 DO4
DI5 DO5
DI6 DO6
DI7 DO7
DI8 DO8
DS2 MD DS1
AB15
MEMR (BUF)
MEMW (BUF)
NOTE: 8212 CLR CONNECTED TO +5V

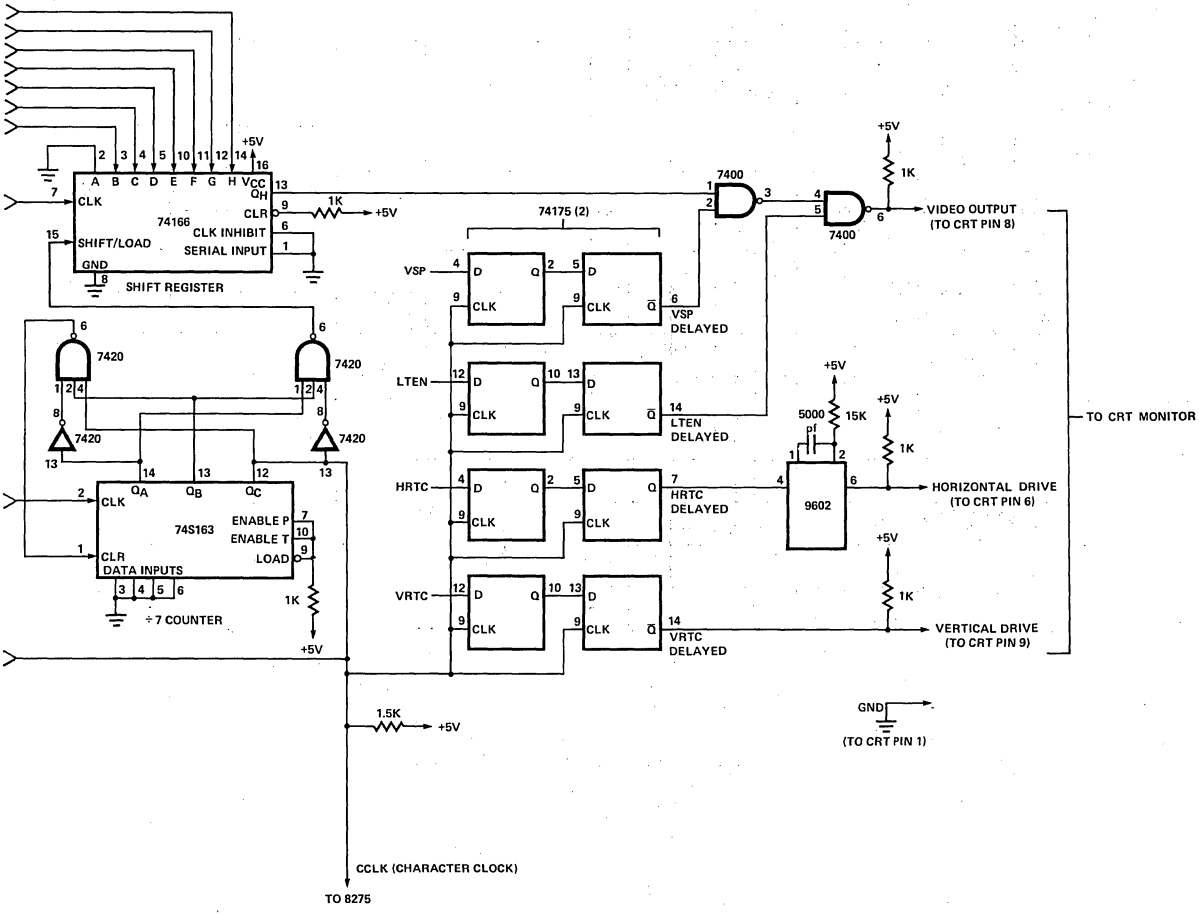


CRT Terminal Schematic — Peripherals Section


CCLK FROM DOT TIMING LOGIC
+5V
GND
VCC
CCLK
DRQ
DACK
I/OR
I/OW
AB0
RD
WR
A0
CC0
CC1
CC2
CC3
CC4
CC5
CC6
LC0
LC1
LC2
LC3
HRTC
VRTC
LTEN
VSP
8275
TO DOT TIMING LOGIC
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
CS
IRQ
FROM PERIPHERAL DECODER
1K
7408
8275 INT
VSS
RL0
RL1
RL2
RL3
RL4
RL5
RL6
RL7
RESET
CLK
ϕ2 (TTL)
SHIFT
CNTL
8279
FROM MICROSWITCH KEYBOARD
SL2
SL1
SL0
G1
C
B
A
G2A
G2B
OUT 0
OUT 1
OUT 2
OUT 3
OUT 4
OUT 5
OUT 6
OUT 7
74S138
TO MICROSWITCH KEYBOARD



CRT Terminal Schematic — Dot Timing Logic Section

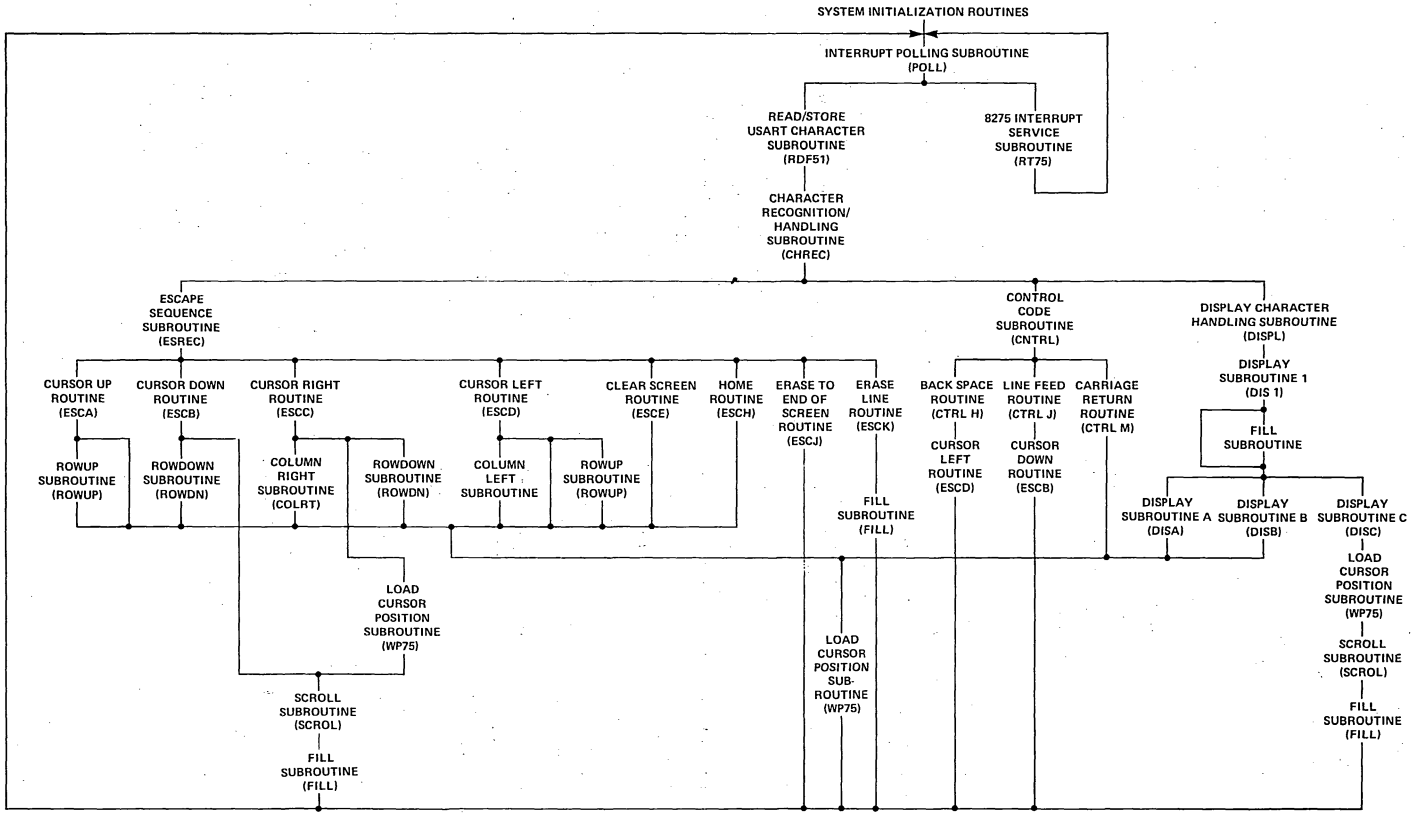

74166
SHIFT REGISTER
CLK
SHIFT/LOAD
GND
CLR
CLK INHIBIT
SERIAL INPUT
+5V
1K
7420
74S163
÷7 COUNTER
ENABLE P
ENABLE T
LOAD
DATA INPUTS
74175 (2)
VSP
VSP DELAYED
LTEN
LTEN DELAYED
HRTC
HRTC DELAYED
VRTC
VRTC DELAYED
7400
VIDEO OUTPUT (TO CRT PIN 8)
5000 pf
15K
9602
HORIZONTAL DRIVE (TO CRT PIN 6)
VERTICAL DRIVE (TO CRT PIN 9)
TO CRT MONITOR
GND (TO CRT PIN 1)
1.5K
CCLK (CHARACTER CLOCK)
TO 8275

Appendix 5.2
ESCAPE/CONTROL/DISPLAY CHARACTER SUMMARY

	CONTROL CHARACTERS				DISPLAYABLE CHARACTER						ESCAPE SEQUENCE			
BIT	0_0	0_0	0_1	0_1	1_0	1_0	1_1	1_1	0_1	0_1	1_0	1_0	1_1	1_1
0000	NUL @	DLE P	SP	ϕ	@	P								
0001	SOH A	DCI Q	!	I	A	Q					↑ A			
0010	STX B	DC2 R	"	2	B	R					↓ B			
0011	ETX C	DC3 S	#	3	C	S					→ C			
0100	EOT D	DC4 T	$	4	D	T					← D			
0101	ENQ E	NAK U	%	5	E	U					CLR E			
0110	ACK F	SYN V	&	6	F	V								
0111	BEL G	ETB W	'	7	G	W								
1000	BS H	CAN X	(	8	H	X					HOME H			
1001	HT I	EM Y	)	9	I	Y								
1010	LF J	SUB Z	*	:	J	Z					EOS I			
1011	VT K	ESC [	+	;	K	[					EL J			
1100	FF L	FS /	,	<	L	\								
1101	CR M	GS	–	=	M	]								
1110	SO N	RS ^	.	>	N	^								
1111	S1 O	US –	/	?	O	–								

NOTE: Shaded blocks = functions terminal will react to. Others can be generated but are ignored up on receipt.

Appendix 5.3
SUBROUTINE INTERRELATIONSHIPS



Appendix 5.4
SOFTWARE TIMING

Subroutine execution times are summarized in the flowchart provided in Figure 5-1. The values shown represent the number of clock cycles required for the execution of a given routine. The actual routine execution time is obtained by multiplying the number of clock cycles/routine by the time/clock cycle. For a 2.048 MHz system clock, the time/clock cycle is 0.4883 μsec. It should be noted that the values indicated represent worst-case execution times. In order to appreciate the meaning of the subroutine execution times, it is necessary to consider two factors:

1. The time available for the CPU to execute instructions between DMA operations.
2. The maximum rate at which data characters are presented to the CPU for processing.

CPU availability during a complete display frame is illustrated in Figure 5-2. Available CPU processing time, per character, at 4800 baud, during the DMA active portion of the display frame, is illustrated in Figure 5-3. It can be seen from Figure 5-3 that 1443 μsec are available for processing each character during the DMA active portion of the frame. Total CPU processing time during the DMA inactive portion of the frame may be seen from Figure 5-2 to be 1234 μsec. This value encompasses the time to process the 8275 interrupt and perform character handling functions.

Using the information contained in Figure 5-1, the maximum execution time* for a given character handling routine is 802 μsec. Since this value is less than 1.443 msec, proper timing is assured. Using the maximum character handling routine execution time and the time required for 8275 interrupt processing, the maximum CPU availability requirement during the DMA inactive portion of the frame may be calculated. This value corresponds to 802 μsec + 253 μsec (8275 interrupt processing) or 1055 μsec. Since this value is less than 1234 μsec, proper timing is assured.

*see notes, Figure 5-1.

Appendix 5.5
VISUAL ATTRIBUTE IMPLEMENTATION CONSIDERATIONS

In order to utilize the visual attribute features of the 8275, it is necessary to modify the CRT system hardware and software functions accordingly.

Hardware modifications necessary to implement character attributes are illustrated in Figure 5-4. The attribute outputs LA0–LA1 selectively control the data transferred to the output shift register.

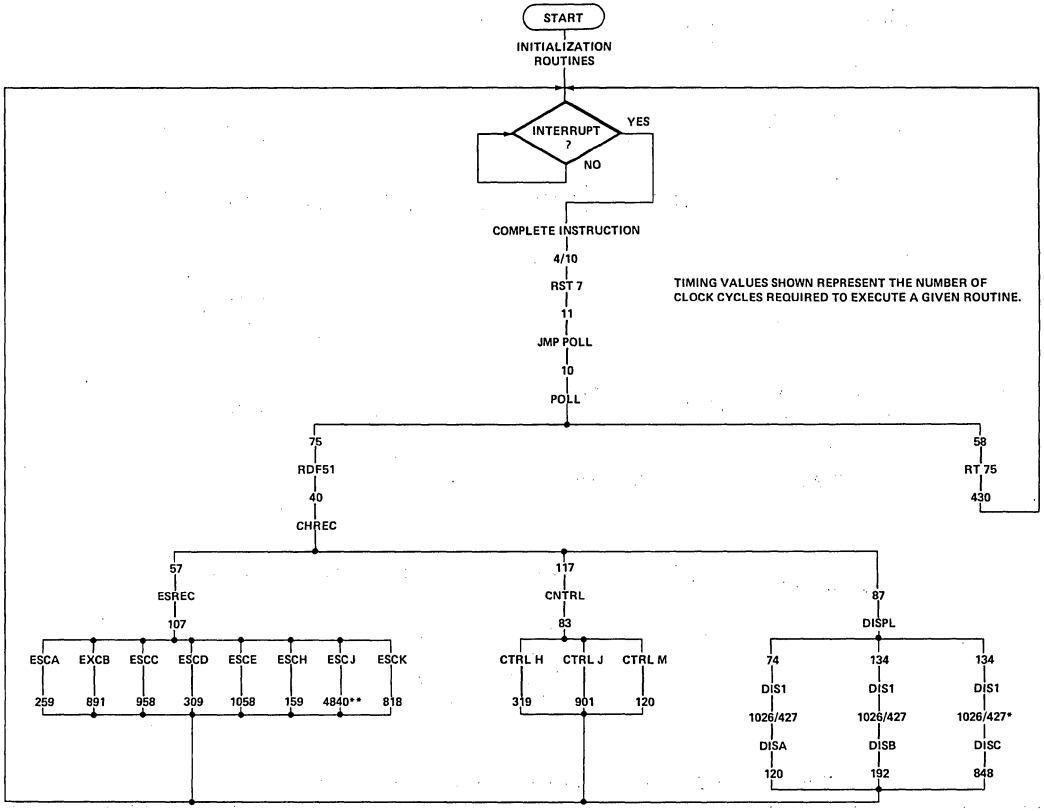
The software memory management scheme presented in the Application Note must be modified in order to accommodate attribute features. An outline of the software considerations involved when using the attribute features is presented as follows:

1. Attributes, as described in the 8275 Data Sheet, occupy character locations in display memory. Since the number of attributes per display row may be variable, the linear mapping relationship between character position on the screen and memory pointers Top, Row Count, and Column Count no longer exists. It is necessary to keep track of the number of attribute characters in each row and their specific location when modifying pointer values.

2. The increased number of character locations required will force the user to incorporate additional display RAM.

3. Since the total number of characters in display memory may be variable when attributes are utilized, it is necessary to modify the starting address and terminal count values for the DMA channels as required.

4. Character insertion and deletion operations may be handled through block transfer operations or through the use of extended display memory row segments.



* UNDER NORMAL OPERATING CONDITIONS, 427 CLOCK CYCLES REPRESENTS THE WORST CASE EXECUTION TIME FOR THIS ROUTINE.

** IT IS NECESSARY FOR THE REMOTE DEVICE TO WAIT APPROXIMATELY 2.5 ms FOLLOWING THE TRANSMISSION OF AN ESCJ CHARACTER BEFORE RESUMING TRANSMISSION.

Figure 5-1. Subroutine Execution Times Flowchart

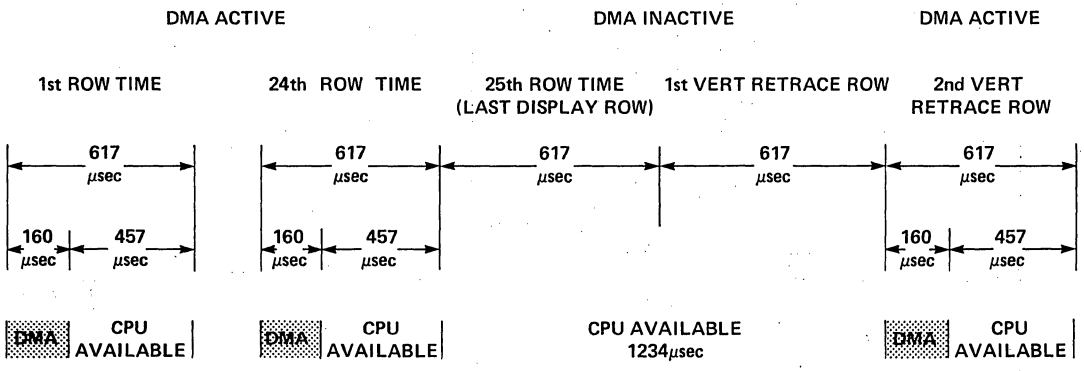
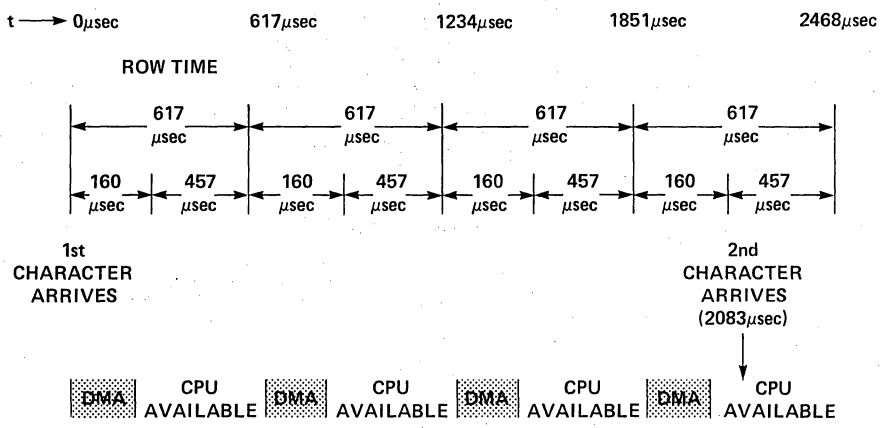


Figure 5-2. CPU Availability



TIME BETWEEN CHARACTERS = 2.083 ms = 2083μsec

TOTAL CPU TIME AVAILABLE BETWEEN CHARACTERS	= (457μsec X 3) + (2082 – 2011μsec) = 1443μsec

BAUD RATE = 4800 BAUD
10 BITS/CHARACTER

Figure 5-3. CPU Availability/Character at 4800 Baud (DMA Active)

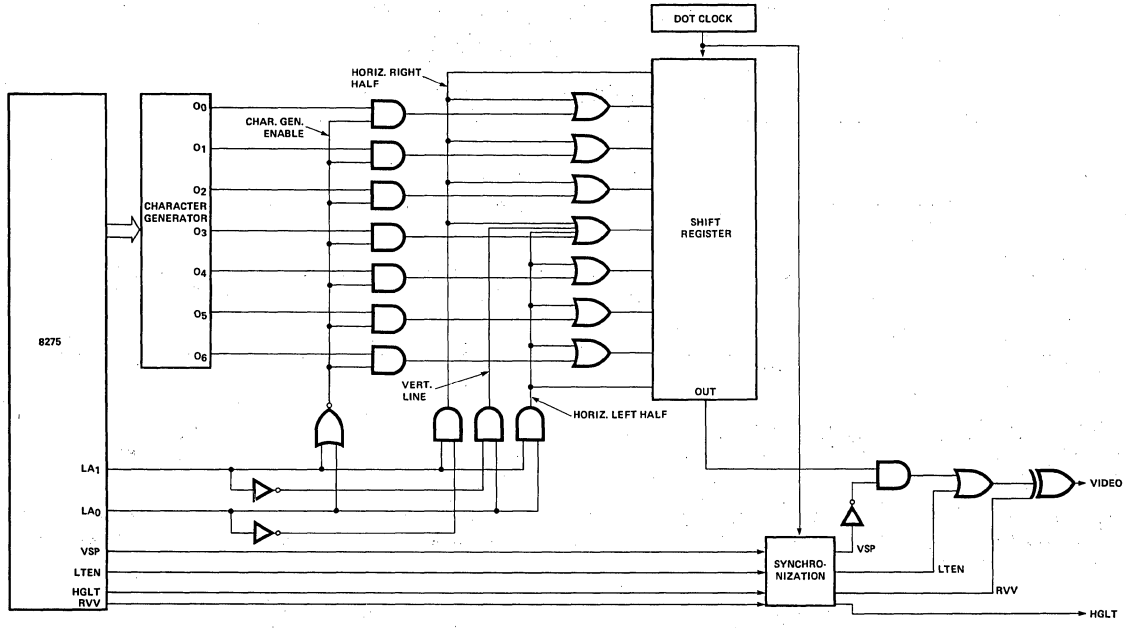


Figure 5-4. Typical Character Attribute Logic

Appendix 5.6
SOFTWARE LISTINGS

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                    1 ;8275/8279 CRT SYSTEM SOFTWARE
                    2          ;
                    3          ;SYSTEM EQUATES
                    4          ;
00FB                5 CNCTL    EQU     0FBH            ;8251 CONTROL ADDRESS
00FA                6 CNIN     EQU     0FAH            ;8251 INPUT DATA ADD
00FA                7 CNOUT    EQU     0FAH            ;8251 OUTPUT DATA ADD
006F                8 KCOM     EQU     6FH             ;8279 COMMAND ADDRESS
006E                9 KDAT     EQU     6EH             ;8279 DATA ADDRESS
005F               10 CRCOM    EQU     5FH             ;8275 COMMAND ADDRESS
005E               11 CRDAT    EQU     5EH             ;8275 DATA ADDRESS
0044               12 PC2SA    EQU     44H             ;8257 CH 2 START ADD PORT
0045               13 PC2TC    EQU     45H             ;8257 CH 2 TERM COUNT PORT
0046               14 PC3SA    EQU     46H             ;8257 CH 3 STARTING ADD PORT
0047               15 PC3TC    EQU     47H             ;8257 CH 3 TERM COUNT PORT
0000               16 MDC57    EQU     00H             ;8257 MODE CLEAR
0084               17 MDS57    EQU     84H             ;8257 MODE SET (AUTOLOAD, CH 2 ENABLED)
0048               18 PMD57    EQU     48H             ;8257 MODE SET PORT
                   19          ;
                   20          ;SYSTEM INITIALIZATION ROUTINES
                   21          ;
                   22          ;
0000 C34000        23          JMP     CRTGO           ;JUMP TO START OF MAIN ROUTINE
                   24          ;
0038               25          ORG     0038H
                   26          ;
0038 C3C900        27          JMP     POLL            ;JUMP TO START OF INT SERVICE ROUTINE
                   28          ;
0040               29          ORG     0040H
                   30          ;
0040 F3            31 CRTGO:   DI                      ;DISABLE INTERRUPTS
0041 31FF87        32          LXI     SP, 87FFH       ;LOAD STACK POINTER
                   33          ;
                   34          ;MEMORY CLEAR ROUTINE
                   35          ;
0044 210080        36          LXI     H,8000H         ;LOAD H&L WITH START ADD OF DISPLAY MEM
0047 3E20          37 THETA:   MVI     A, 20H          ;LOAD A WITH SPACE CHAR CODE
0049 77            38          MOV     M,A             ;LOAD SPACE CHAR IN MEM
004A 7D            39          MOV     A,L             ;MOVE LOW ADD BYTE TO A
004B FECF          40          CPI     0CFH            ;COMPARE WITH 0CFH
004D CA5400        41          JZ      NXT1            ;IF COMPARRISON JMP TO NXT1
0050 23            42          INX     H               ;INCREMENT H&L
0051 C34700        43          JMP     THETA           ;JMP TO THETA, CONT LOADING MEMORY
0054 7C            44 NXT1:    MOV     A,H             ;MOVE UP ADD BYTE TO A
0055 FE87          45          CPI     87H             ;COMPARE WITH 87H
0057 CA5E00        46          JZ      NXT2            ;IF COMPARRISON,ADD=LAST DISPLAY
                   47                                  ;ADD,THEREFORE,JMP TO NXT2
005A 23            48          INX     H               ;INCREMENT H&L
005B C34700        49          JMP     THETA           ;JMP TO THETA, CONT LOADING MEMORY
                   50          ;
                   51          ;POINTER/BUFFER CLEAR ROUTINE
                   52          ;

005E 210000        53 NXT2:    LXI     H,0000H         ;ZERO H&L
0061 22D387        54          SHLD    RCTAD           ;ZERO ROW COUNT
0064 22E287        55          SHLD    LOCBUF          ;ZERO BUFFER
0067 22D887        56          SHLD    LOCAD           ;ZERO CHARACTER LOCATION
006A 22DA87        57          SHLD    LOC01           ;ZERO LOC OF 1ST CHAR IN ROW
006D 22DC87        58          SHLD    LOC80           ;ZERO LOC OF 80TH CHAR IN ROW
0070 22DE87        59          SHLD    LOCXX           ;ZERO PRESENT LOC OF 1ST CHAR IN ROW
0073 22E087        60          SHLD    LOCPR           ;ZERO PRESENT LOC OF 1ST CHAR IN ROW
0076 210080        61          LXI     H,8000H         ;LOAD H&L WITH 8000H
0079 22D687        62          SHLD    TOPAD           ;SET TOP = 8000H
007C 218087        63          LXI     H,8780H         ;LOAD H&L WITH 8780H
007F 22E687        64          SHLD    BOTAD           ;SET BOT = 8780H
0082 3E00          65          MVI     A,00H           ;ZERO A
0084 32D287        66          STA     CCTAD           ;ZERO COLUMN COUNT
0087 32D587        67          STA     CURSY           ;ZERO CURSOR Y POINTER
008A 32E487        68          STA     XFLG            ;ZERO ESC SEQ FLAG
008D 32E587        69          STA     USCHR           ;ZERO USART CHAR BUFFER
                   70          ;
                   71          ;8251 INITIALIZATION ROUTINE
                   72          ;
0090 3E4F          73          MVI     A,4FH           ;MODE SET VALUE TO A
0092 D3FB          74          OUT     CNCTL           ;OUPUT VALUE
0094 3E27          75          MVI     A,27H           ;COMMAND WORD TO A
0096 D3FB          76          OUT     CNCTL           ;OUTPUT VALUE
                   77          ;
                   78          ;8279 INITIALIZATION ROUTINE
                   79          ;
0098 3E35          80          MVI     A,35H           ;OUTPUT PROG CLOCK, DIV BY 21
009A D36F          81          OUT     KCOM
                   82          ;
                   83          ;8275 INITIALIZATION ROUTINE
                   84          ;
```

```
009C 3E00          MVI     A,00H          ;RESET AND STOP DISPLAY
009E D35F          OUT     CRCOM
00A0 3E4F          MVI     A,4FH          ;SCREEN PARAM BYTE 1
00A2 D35E          OUT     CRDAT
00A4 3E58          MVI     A,58H          ;              BYTE 2
00A6 D35E          OUT     CRDAT
00A8 3E89          MVI     A,89H          ;              BYTE 3
00AA D35E          OUT     CRDAT
00AC 3ED9          MVI     A,0D9H         ;              BYTE 4
00AE D35E          OUT     CRDAT
00B0 3E80          MVI     A,80H          ;LOAD CURSOR POSITION
00B2 D35F          OUT     CRCOM
00B4 3E00          MVI     A,00H          ;CURSOR X POSITION
00B6 D35E          OUT     CRDAT
00B8 3E00          MVI     A,00H          ;CURSOR Y POSITION
00BA D35E          OUT     CRDAT
00BC 3EE0          MVI     A,0E0H         ;PRESET COUNTERS
00BE D35F          OUT     CRCOM
00C0 3E23          MVI     A,23H          ;START DISPLAY
00C2 D35F          OUT     CRCOM
00C4 FB            EI                     ;ENABLE INTERRUPTS
00C5 00      LOOP: NOP
00C6 C3C500        JMP     LOOP
                   ;
                   ;
                   ;8275/8251 INTERRUPT POLLING ROUTINE
                   ;
                   ;
00C9 DB5F    POLL: IN      CRCOM          ;READ 8275 STATUS, CLEARING INT
00CB E620          ANI     20H            ;MASK STATUS, SAVE INT REQ BIT
00CD CAD500        JZ      AGGIE          ;IF STATUS=1, SERVICE 8275
                   ;
00D0 CD7304  GIGEM: CALL   RT75           ;CALL 8275 INT SERVICE SUBROUTINE
00D3 FB            EI                     ;ENABLE INTERRUPTS
00D4 C9            RET                    ;RETURN
                   ;
00D5 CDDD00  AGGIE: CALL   RDF51          ;CALL READ USART CHAR ROUTINE
00D8 CDE500        CALL    CHREC          ;CALL CHARACTER RECOG/HANDLING ROUTINE
00DB FB            EI                     ;ENABLE INTERRUPTS
00DC C9            RET                    ;RETURN
                   ;
                   ;USART READ/STORE CHAR SUBROUTINE
                   ;
00DD DBFA    RDF51: IN     CNIN           ;READ ASCII CHAR FROM USART, RESETTING RXRDY
00DF E67F          ANI     7FH            ;MASK BIT 8,SAVE BITS 1-7
00E1 32E587        STA     USCHR          ;STORE USART CHAR IN MEMORY
00E4 C9            RET                    ;RETURN
                   ;
                   ;CHARACTER RECOGNITION/HANDLING SUBROUTINE
                   ;
00E5 3AE487  CHREC: LDA    XFLG           ;LOAD A WITH ESC SEQ FLAG
00E8 E6FF          ANI     0FFH           ;SET/RESET ZERO BIT
00EA CAF100        JZ      NXTX           ;IF ONE,CHAR=2ND CHAR IN ESC SEQ
00ED CD0F01        CALL    ESREC          ;CALL ESC SEQ SUBROUTINE
00F0 C9            RET                    ;RETURN
00F1 3AE587  NXTX: LDA     USCHR          ;LOAD USART CHAR IN A
00F4 E660          ANI     60H            ;MASK BITS 1-5,&8,SAVING BITS 6&7
00F6 CAFD00        JZ      NXTY           ;IF ZERO CHAR=CONTROL CHAR
                                          ;IF ONE CHAR=DISPLAY CHAR
00F9 CD4B03        CALL    DISPL          ;CALL DISPLAY CHAR SUBROUTINE
00FC C9            RET                    ;RETURN
00FD 3AE587  NXTY: LDA     USCHR          ;LOAD  USART CHAR IN A
0100 E610          ANI     10H            ;MASK OFF BITS,SAVE BIT 5
0102 C20901        JNZ     NXTZ           ;IF ZERO CONT CHAR=CONT CODE
                                          ;IF ONE CONT CHAR=ESC CODE
0105 CD2701        CALL    CNTRL          ;CALL CONTROL CODE SUBROUTINE
0108 C9            RET                    ;RETURN
0109 21E487  NXTZ: LXI     H,XFLG         ;LOAD H&L WITH ADD OF ESC SEQ FLAG
010C 3601          MVI     M,01H          ;SET ESC SEQ FLAG
010E C9            RET                    ;RETURN
                   ;
                   ;ESCAPE SEQUENCE SUBROUTINE
                   ;
010F 3E00    ESREC: MVI     A,00H          ;ZERO A
0111 32E487        STA     XFLG           ;RESET ESC SEQ FLAG
0114 3AE587        LDA     USCHR          ;LOAD USART CHAR IN A
0117 E60F          ANI     0FH            ;MASK BITS 5-8
0119 07            RLC                    ;SHIFT LEFT,YIELDING OFFSET
011A 21D004        LXI     H,BSET1        ;LOAD BASE ADD OF TABLE 1 IN H&L
011D 110000        LXI     D,0000H        ;ZERO D&E
0120 5F            MOV     E,A            ;LOAD OFFSET IN E
0121 19            DAD     D              ;ADD OFFSET TO BASE, RESULT IN H&L
0122 5E            MOV     E,M            ;MOVE LOW BYTE OF ROUTINE ADD TO E
0123 23            INX     H              ;INCREMENT COMPUTED ADDRESS
0124 56            MOV     D,M            ;MOVE UP BYTE OF ROUTINE ADD TO D
0125 EB            XCHG                   ;EXCHANGE D&E WTIH H&L
0126 E9            PCHL                   ;LOAD PC WITH ROUTINE ADD, JMP TO ROUTINE
                   ;
                   ;CONTROL CODE SUBROUTINE
```

```
                  ;
0127 3AE587  CNTRL: LDA     USCHR        ;LOAD USART CHAR IN A
012A E606           ANI     06H          ;MASK CHAR, SAVE BITS 2-3
012C 21F004         LXI     H,BSET2      ;LOAD BASE ADD OF TABLE 2 IN H&L
012F 110000         LXI     D,0000H      ;CLEAR D&E
0132 5F             MOV     E,A          ;LOAD OFFSET IN E
0133 19             DAD     D            ;ADD OFFSET TO BASE, RESULT IN H&L
0134 5E             MOV     E,M          ;MOVE LOW BYTE OF ROUTINE ADD TO E
0135 23             INX     H            ;INCREMENT COMPUTED ADDRESS
0136 56             MOV     D,M          ;MOVE UP BYTE OF ROUTINE ADD TO D
0137 EB             XCHG                 ;EXCHANGE D&E WITH H&L
0138 E9             PCHL                 ;LOAD PC WITH ROUTINE ADD, JMP TO ROUTINE
                  ;
                  ;CURSOR UP ROUTINE
                  ;
0139 2AD387  ESCA:  LHLD    RCTAD        ;LOAD ROWCOUNT IN H&L
013C 7D             MOV     A,L          ;MOVE LOW BYTE OF ROWCOUNT TO A
013D FE00           CPI     00H          ;COMPARE BYTE WITH 00H
013F CA4601         JZ      ALPHA        ;IF BYTE=0 CONTINUE COMPARRISON
0142 CD0803         CALL    ROWUP        ;CALL ROWUP SUBROUTINE
0145 C9             RET                  ;RETURN
0146 7C      ALPHA: MOV     A,H          ;MOVE UP BYTE OF ROWCOUNT TO A
0147 FE00           CPI     00H          ;COMPARE BYTE WITH 00H
0149 CA5001         JZ      BETA         ;IF BYTE=0,ROWCOUNT=FIRST ROW
014C CD0803         CALL    ROWUP        ;CALL ROWUP SUBROUTINE
014F C9             RET                  ;RETURN
0150 218007  BETA:  LXI     H,0780H      ;LOAD H&L WITH ROWCOUNT=LAST ROW VALUE (1920D)
0153 22D387         SHLD    RCTAD        ;STORE 0780H IN ROWCOUNT BUFFER
0156 3E18           MVI     A,18H        ;LOAD A WITH CURSOR Y POS=LAST ROW VALUE (240)
0158 32D587         STA     CURSY        ;STORE 18H IN CURSOR Y POS BUFFER
015B CD3C03         CALL    WP75         ;CALL LOAD CURSOR POSITION SUBROUTINE
015E C9             RET
                  ;
                  ;CURSOR DOWN ROUTINE
                  ;
015F 2AD387  ESCB:  LHLD    RCTAD        ;LOAD ROWCOUNT IN H&L
0162 7D             MOV     A,L          ;MOVE LOW BYTE OF ROWCOUNT TO A
0163 FE80           CPI     80H          ;COMPARE BYTE WITH 80H
0165 CA6C01         JZ      GAMMA        ;IF BYTE=80H, CONTINUE COMPARRISON
0168 CD1A03         CALL    ROWDN        ;CALL ROWDOWN SUBROUTINE
016B C9             RET                  ;RETURN
016C 7C      GAMMA: MOV     A,H          ;MOVE UP BYTE OF ROWCOUNT TO A
016D FE07           CPI     07H          ;COMPARE BYTE WITH 07H
016F CA7601         JZ      DELTA        ;IF BYTE=07H, ROWCOUNT=LAST ROW
0172 CD1A03         CALL    ROWDN        ;CALL ROWDOWN SUBROUTINE
0175 C9             RET                  ;RETURN
0176 CD3C03  DELTA: CALL    WP75         ;CALL LOAD CURSOR POSITION SUBROUTINE
0179 CD0B04         CALL    SCROL        ;CALL SCROLL SUBROUTINE
017C C9             RET                  ;RETURN
                  ;
                  ;CURSOR RIGHT ROUTINE
                  ;
017D 3AD287  ESCC:  LDA     CCTAD        ;LOAD COLUMN COUNT IN A
0180 FE4F           CPI     4FH          ;COMPARE BYTE WITH 4FH
0182 CA8901         JZ      ZETA         ;IF BYTE=4FH, COLUMN COUNT =LAST
                                         ;CHARACTER POS IN ROW
0185 CD3403         CALL    COLRT        ;CALL COLUMN RIGHT SUBROUTINE
0188 C9             RET                  ;RETURN
0189 2AD387  ZETA:  LHLD    RCTAD        ;LOAD ROWCOUNT IN H&L
018C 7D             MOV     A,L          ;MOVE LOW BYTE OF ROWCOUNT TO A
018D FE80           CPI     80H          ;COMPARE BYTE WITH 80H
018F C29B01         JNZ     CCTOA        ;IF BYTE=80H, CONTINUE COMPARRISON
0192 7C             MOV     A,H          ;MOVE UP BYTE OF ROWCOUNT TO A
0193 FE07           CPI     07H          ;COMPARE BYTE WITH 07H
0195 C29B01         JNZ     CCTOA        ;IF BYTE=07H,ROWCOUNT=LAST ROW
0198 C3A401         JMP     CCTOB        ;JUMP TO CCTOB
019B 3E00    CCTOA: MVI     A,00H        ;ZERO A
019D 32D287         STA     CCTAD        ;ZERO COLUMN COUNT
01A0 CD1A03         CALL    ROWDN        ;CALL ROWDOWN SUBROUTINE
01A3 C9             RET                  ;RETURN
01A4 3E00    CCTOB: MVI     A,00H        ;ZERO A
01A6 32D287         STA     CCTAD        ;ZERO COLUMN COUNT BUFFER
01A9 CD3C03         CALL    WP75         ;CALL LOAD CURSOR POSITION SUBROUTINE
01AC CD0B04         CALL    SCROL        ;CALL SCROLL SUBROUTINE
01AF C9             RET                  ;RETURN
                  ;
                  ;CURSOR LEFT ROUTINE
                  ;
01B0 3AD287  ESCD:  LDA     CCTAD        ;LOAD COLUMN COUNT IN A
01B3 FE00           CPI     00H          ;COMPARE BYTE WITH 00H
01B5 CABC01         JZ      NXTA         ;IF BYTE=0,COLUMN COUNT =FIRST CHAR POS IN ROW
01B8 CD2C03         CALL    COLLT        ;CALL COLUMN LEFT SUBROUTINE
01BB C9             RET                  ;RETURN
01BC 2AD387  NXTA:  LHLD    RCTAD        ;LOAD ROWCOUNT IN H&L
01BF 7D             MOV     A,L          ;LOAD LOW BYTE OF ROWCOUNT IN A
01C0 FE00           CPI     00H          ;COMPARE BYTE WITH 00H
01C2 C2CE01         JNZ     CCTMA        ;IF BYTE=0,CONTINUE COMPARRISON
01C5 7C             MOV     A,H          ;LOAD UP BYTE OF ROWCOUNT IN A
01C6 FE00           CPI     00H          ;COMPARE BYTE WITH ZERO
01C8 C2CE01         JNZ     CCTMA        ;IF BYTE=0,HOME POS CONDITION EXISTS
```

```
01CB C3D701          JMP     CCTMB          ;JUMP TO CCTMB
01CE 3E4F    CCTMA:  MVI     A,4FH          ;LOAD A WITH 4FH
01D0 32D287          STA     CCTAD          ;SET COLUMN COUNT=4FH=79D
01D3 CD0803          CALL    ROWUP          ;CALL ROWUP SUBROUTINE
01D6 C9              RET                    ;RETURN
01D7 218007  CCTMB:  LXI     H,0780H        ;LOAD H&L WITH ROWCOUNT=780H=1920D
01DA 22D387          SHLD    RCTAD          ;SET ROWCOUNT =1920D
01DD 3E4F            MVI     A,4FH          ;LOAD A WITH 4FH
01DF 32D287          STA     CCTAD          ;SET COLUMN COUNT=4FH=79D
01E2 3E18            MVI     A,18H          ;LOAD A WITH 18H
01E4 32D587          STA     CURSY          ;SET CURSOR Y POINTER=18H=24D
01E7 CD3C03          CALL    WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
01EA C9              RET                    ;RETURN
                     ;
                     ;HOME ROUTINE
                     ;
01EB 210000  ESCH:   LXI     H,0000H        ;ZERO H&L
01EE 22D387          SHLD    RCTAD          ;SET ROWCOUNT=0
01F1 3E00            MVI     A,00H          ;ZERO A
01F3 32D287          STA     CCTAD          ;SET COLUMN COUNT=0
01F6 32D587          STA     CURSY          ;SET CURSOR Y POINTER=0
01F9 CD3C03          CALL    WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
01FC C9              RET                    ;RETURN
                     ;
                     ;ERASE LINE ROUTINE
                     ;
01FD 2AD687  ESCK:   LHLD    TOPAD          ;LOAD TOP IN H&L
0200 EB              XCHG                   ;STORE TOP IN D&E
0201 2AD387          LHLD    RCTAD          ;LOAD ROWCOUNT IN H&L
0204 19              DAD     D              ;ADD TOP+ROWCOUNT, RESULT IN H&L
0205 22DE87          SHLD    LOCXX          ;STORE RESULT IN MEM
                     ;
0208 3E87            MVI     A,87H          ;LOAD 87H IN A
020A BC              CMP     H              ;COMPARE H WITH 87H
020B D21402          JNC     FRODO          ;IF NO CARRY, CONTINUE
020E CD2A02          CALL    COMRX          ;IF CARRY,CALL COMPENSATION ROUTINE
0211 C32002          JMP     BILBO          ;JUMP TO BILBO
0214 C22002  FRODO:  JNZ     BILBO          ;IF NOT EQUAL END COMPARRISON
0217 3ECF            MVI     A,0CFH         ;LOAD CFH IN A
0219 BD              CMP     L              ;COMPARE L WITH CFH
021A D22002          JNC     BILBO          ;IF NO CARRY,LOCXX LESS THAN OR EQ TO 87CFH
021D CD2A02          CALL    COMRX          ;IF CARRY, CALL COMPENSATION ROUTINE
0220 2ADE87  BILBO:  LHLD    LOCXX          ;LOAD LOC OF FIRST CHAR IN ROW IN H&L
0223 22E287          SHLD    LOCBUF         ;STORE LOCXX IN BUFFER
0226 CD3204          CALL    FILL           ;CALL FILL ROW WITH SP CHAR SUBROUTINE
0229 C9              RET                    ;RETURN
                     ;
                     ;COMPENSATION SUBROUTINE COMRX
                     ;
022A 2ADE87  COMRX:  LHLD    LOCXX          ;LOAD LOCXX IN H&L
022D 1130F8          LXI     D,0F830H       ;LOAD COMPENSATION VALUE IN D&E
0230 19              DAD     D              ;ADD D&E TO H&L
0231 22DE87          SHLD    LOCXX          ;STORE RESULT IN LOCXX
0234 C9              RET
                     ;
                     ;CLEAR SCREEN ROUTINE
                     ;
0235 3EF0    ESCE:   MVI     A,0F0H         ;MOVE EOR CHAR TO A
0237 0619            MVI     B,19H          ;MOVE LOOP CTR START VALUE =19H=25D TO B
0239 115000          LXI     D,50H          ;MOVE 80D=50H TO D&E
023C 210080          LXI     H,8000H        ;MOVE 8000H TO H&L
                     ;
023F 77      LOADX:  MOV     M,A            ;MOVE EOR CHARACTER TO MEM
0240 19              DAD     D              ;ADD 80D=50H TO ADDRESS IN H&L
0241 05              DCR     B              ;DECREMENT B
0242 C23F02          JNZ     LOADX          ;CONTINUE LOOPING IF B NOT ZERO
                     ;
0245 210000          LXI     H,0000H        ;ZERO H&L
0248 22D387          SHLD    RCTAD          ;ZERO ROWCOUNT
024B 210080          LXI     H,8000H
024E 22D687          SHLD    TOPAD
0251 218087          LXI     H,8780H
0254 22E687          SHLD    BOTAD
0257 3E00            MVI     A,00H          ;ZERO A
0259 32D287          STA     CCTAD          ;ZERO COLUMN COUNT
025C 32D587          STA     CURSY          ;ZERO CURSOR Y POS
025F 32E487          STA     XFLG
0262 CD3C03          CALL    WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
0265 C9              RET
                     ;
                     ;ERASE TO END OF SCREEN ROUTINE
                     ;
0266 2AD687  ESCJ:   LHLD    TOPAD          ;LOAD TOP IN H&L
0269 EB              XCHG                   ;STORE TOP IN D&E
026A 2AD387          LHLD    RCTAD          ;LOAD ROW COUNT IN H&L
026D 19              DAD     D              ;ADD TOP+ROWCOUNT, YIELDING LOC OF
                                            ;FIRST CHAR IN PRESENT ROW
026E 22E087          SHLD    LOCPR          ;STORE LOCATION IN MEM
                     ;
```

```
0271 3E87          353          MVI     A,87H          ;LOAD 87H IN A
0273 BC            354          CMP     H              ;COMPARE H WITH 87H
0274 D27D02        355          JNC     VAR            ;IF NO CARRY, CONTINUE COMPARRISON
0277 CDEE02        356          CALL    COMRY          ;CALL COMPENSATION ROUTINE
027A C38902        357          JMP     FIN            ;JUMP TO FIN
027D C28902        358 VAR:     JNZ     FIN            ;IF NOT EQUAL END COMPARRISON
0280 3ECF          359          MVI     A,0CFH         ;LOAD CFH IN A
0282 BD            360          CMP     L              ;COMPARE L WITH CFH
0283 D28902        361          JNC     FIN            ;IF NO CARRY,LOCPR LESS THAN OR EQ TO 87CFH
0286 CDEE02        362          CALL    COMRY          ;CALL COMPENSATION ROUTINE
                   363          ;
                   364          ;
0289 2AD687        365 FIN:     LHLD    TOPAD          ;LOAD TOP IN H&L
028C 7D            366          MOV     A,L            ;MOVE L TO A
028D FE00          367          CPI     00H            ;COMPARE BYTE TO 00H
028F C2A102        368          JNZ     TROLL          ;IF NO COMPARRISON, JUMP TO TROLL
0292 7C            369          MOV     A,H            ;MOVE H TO A
0293 FE80          370          CPI     80H            ;COMPARE BYTE WITH 80H
0295 C2A102        371          JNZ     TROLL          ;IF NO COMPARRISON,JUMP TO TROLL
0298 218087        372          LXI     H,8780H        ;IF COMPARRISON,SET BOT=8780H
029B 22E687        373          SHLD    BOTAD
029E C3AB02        374          JMP     GNOME          ;JUMP TO GNOME
02A1 11B0FF        375 TROLL:   LXI     D,0FFB0H       ;LOAD -80D=0FFB0H IN D&E
02A4 2AD687        376          LHLD    TOPAD          ;LOAD TOP IN H&L
02A7 19            377          DAD     D              ;ADD -80D TO TOP
02A8 22E687        378          SHLD    BOTAD
                   379          ;
02AB 3EF0          380 GNOME:   MVI     A,0F0H         ;LOAD A WITH EOR CHAR (LOOP START)
                   381          ;
02AD 2AE087        382          LHLD    LOCPR          ;LOAD LOCPR IN H&L
02B0 77            383          MOV     M,A            ;MOVE EOR CHAR TO MEM
                   384          ;
02B1 7D            385          MOV     A,L            ;MOVE L TO A
02B2 FE80          386          CPI     80H            ;COMPARE YTE WITH 80H
02B4 C2D502        387          JNZ     WIZAR          ;IF NO COMPARRISON,JMP TO WIZAR
02B7 7C            388          MOV     A,H            ;MOVE H TO A
02B8 FE87          389          CPI     87H            ;COMPARE BYTE WITH 87H
02BA C2D502        390          JNZ     WIZAR          ;IF NO COMPARRISON, JMP TO WIZAR
                   391                                 ;IF COMPARRISON,PROCEED TO GZONK
02BD EB            392 GZONK:   XCHG                   ;STORE PRESENT LOC IN D&E
02BE 2AE687        393          LHLD    BOTAD          ;LOAD BOT IN H&L
02C1 7D            394          MOV     A,L            ;MOVE L TO A
02C2 BB            395          CMP     E              ;COMPARE E WITH A
02C3 C2CC02        396          JNZ     FUN            ;IF NO COMP, JUMP TO FUN
02C6 7C            397          MOV     A,H            ;MOVE H TO A
02C7 BA            398          CMP     D              ;COMPARE D WITH A
02C8 C2CC02        399          JNZ     FUN            ;IF NO COMP, JMP TO FUN
                   400                                 ;IFCOMPARRISON,RETURN
02CB C9            401          RET                    ;RETURN
02CC 210080        402 FUN:     LXI     H,8000H        ;LOAD H&L WITH 8000H
02CF 22E087        403          SHLD    LOCPR          ;SET LOCPR =8000H
02D2 C3AB02        404          JMP     GNOME
                   405          ;
02D5 EB            406 WIZAR:   XCHG                   ;STORE LOCPR IN D&E
02D6 2AE687        407          LHLD    BOTAD          ;LOAD BOT IN H&L
02D9 7D            408          MOV     A,L            ;MOVE L TO A
02DA BB            409          CMP     E              ;COMPARE E WITH A
02DB C2E402        410          JNZ     NUF            ;IF NO COMP,JMP TO NUF
02DE 7C            411          MOV     A,H            ;MOVE H TO A
02DF BA            412          CMP     D              ;COMPARE D WITH A
02E0 C2E402        413          JNZ     NUF            ;IF NO COMP, JMP TO NUF
                   414                                 ;IF COMPARRISON,RETURN
02E3 C9            415          RET                    ;RETURN
02E4 215000        416 NUF:     LXI     H,50H          ;LOAD 80D=50H IN H&L
02E7 19            417          DAD     D              ;ADD 80D TO LOCPR (LOCPR IN D&E)
02E8 22E087        418          SHLD    LOCPR          ;STORE LOCPR IN MEM
02EB C3AB02        419          JMP     GNOME          ;JUMP TO GNOME
                   420          ;
                   421          ;COMPENSATION SUBROUTINE COMRY
                   422          ;
02EE 2AE087        423 COMRY:   LHLD    LOCPR          ;LOAD LOCPR IN H&L
02F1 1130F8        424          LXI     D,0F830H                ;LOAD COM VALUE IN D&E
02F4 19            425          DAD     D              ;ADD COMPENSATION TO LOCPR
02F5 22E087        426          SHLD    LOCPR          ;STORE LOCPR IN MEM
02F8 C9            427          RET                    ;RETURN
                   428          ;LINE FEED ROUTINE
                   429          ;
02F9 C35F01        430 CTRLJ:   JMP     ESCB
                   431          ;
                   432          ;CARRIAGE RETURN ROUTINE
                   433          ;
02FC 3E00          434 CTRLM:   MVI     A,00H          ;ZERO A
02FE 32D287        435          STA     CCTAD          ;SET COLUMN COUNT=0
0301 CD3C03        436          CALL    WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
0304 C9            437          RET                    ;RETURN
                   438          ;
                   439          ;BACK SPACE ROUTINE
                   440          ;
0305 C3B001        441 CTRLH:   JMP     ESCD
```

```
                        ;
                        ;ROWUP SUBROUTINE
                        ;
0308 2AD387   ROWUP:    LHLD    RCTAD       ;LOAD ROWCOUNT IN H&L
030B 11B0FF             LXI     D,0FFB0H    ;MOVE -80D=0FFB0H (2'S COMP) TO D&E
030E 19                 DAD     D           ;ADD -80D TO ROWCOUNT
030F 22D387             SHLD    RCTAD       ;STORE RESULT IN ROWCOUNT BUFFER

0312 21D587             LXI     H,CURSY     ;LOAD CURSOR Y POINTER ADDRESS IN H&L
0315 35                 DCR     M           ;DECREMENT CURSOR Y POINTER
0316 CD3C03             CALL    WP75        ;CALL LOAD CURSOR POSITION SUBROUTINE
0319 C9                 RET                 ;RETURN
                        ;
                        ;ROWDOWN SUBROUTINE
                        ;
031A 2AD387   ROWDN:    LHLD    RCTAD       ;LOAD ROWCOUNT IN H&L
031D 115000             LXI     D,50H       ;MOVE +80D=50H TO D&E
0320 19                 DAD     D           ;ADD +80D TO ROWCOUNT
0321 22D387             SHLD    RCTAD       ;STORE RESULT IN ROWCOUNT
0324 21D587             LXI     H,CURSY     ;LOAD CURSOR Y POINTER ADDRESS IN H&L
0327 34                 INR     M           ;INCREMENT CURSOR Y POINTER
0328 CD3C03             CALL    WP75        ;CALL LOAD CURSOR POSITION SUBROUTINE
032B C9                 RET                 ;RETURN
                        ;
                        ;COLUMN LEFT SUBROUTINE
                        ;
032C 21D287   COLLT:    LXI     H,CCTAD     ;LOAD COLUMN COUNT ADDRESS IN H&L
032F 35                 DCR     M           ;DECREMENT COLUMN COUNT
0330 CD3C03             CALL    WP75        ;CALL LOAD CURSOR POSITION SUBROUTINE
0333 C9                 RET                 ;RETURN
                        ;
                        ;COLUMN RIGHT SUBROUTINE
                        ;
0334 21D287   COLRT:    LXI     H,CCTAD     ;LOAD COLUMN COUNT ADDRESS IN H&L
0337 34                 INR     M           ;INCREMENT COLUMN COUNT
0338 CD3C03             CALL    WP75        ;CALL LOAD CURSOR POSITION SUBROUTINE
033B C9                 RET                 ;RETURN
                        ;
                        ;LOAD CURSOR POSITION SUBROUTINE
                        ;
033C 3E80     WP75:     MVI     A,80H       ;LOAD A WITH 80H, LOAD CURSOR POSITION COMMAND
033E D35F               OUT     CRCOM
0340 3AD287             LDA     CCTAD       ;LOAD A WITH CURSOR X POSITION
0343 D35E               OUT     CRDAT
0345 3AD587             LDA     CURSY       ;LOAD A WITH CURSOR Y POSITION
0348 D35E               OUT     CRDAT
034A C9                 RET                 ;RETURN
                        ;
                        ;
                        ;DISPLAY CHARACTER HANDLING SUBROUTINE
                        ;
034B 3AD287   DISPL:    LDA     CCTAD       ;LOAD COLUMN COUNT IN H&L
034E FE4F               CPI     4FH         ;COMPARE BYTE WITH 4FH=79D
0350 CA5A03             JZ      CTA         ;IF BYTE=4FH,COLUMN COUNT=LAST CHAR-
                                            ;ACTER IN ROW
0353 CD7E03             CALL    DIS1        ;CALL DIS1 SUBROUTINE
0356 CDBB03             CALL    DISA        ;CALL DISA SUBROUTINE
0359 C9                 RET                 ;RETURN
035A 2AD387   CTA:      LHLD    RCTAD       ;LOAD ROWCOUNT IN H&L
035D 7D                 MOV     A,L         ;LOAD LOW BYTE OF ROWCOUNT IN H$L
035E FE80               CPI     80H         ;COMPARE BYTE WITH 80H
0360 CA6A03             JZ      CTB         ;IF BYTE=80H,CONTINUE COMPARRISON
0363 CD7E03             CALL    DIS1        ;CALL DIS1 SUBROUTINE
0366 CDC303             CALL    DISB        ;CALL DISB SUBROUTINE
0369 C9                 RET                 ;RETURN
036A 7C       CTB:      MOV     A,H         ;MOVE UP BYTE OF ROWCOUNT TO H&L
036B FE07               CPI     07H         ;COMPARE BYTE WITH 07H
036D CA7703             JZ      CTC         ;IF BYTE=07H,END OF DISPLAY COND EXISTS
0370 CD7E03             CALL    DIS1        ;CALL DIS1 SUBROUTINE
0373 CDC303             CALL    DISB        ;CALL DISB SUBROUTINE
0376 C9                 RET                 ;RETURN
0377 CD7E03   CTC:      CALL    DIS1        ;CALL DIS1 SUBROUTINE
037A CDDA03             CALL    DISC        ;CALL DISC SUBROUTINE
037D C9                 RET                 ;RETURN
                        ;
                        ;SUBROUTINE DIS1
                        ;
037E 2AD687   DIS1:     LHLD    TOPAD       ;LOAD TOP IN H&L
0381 EB                 XCHG                ;STORE TOP IN D&E
0382 2AD387             LHLD    RCTAD       ;LOAD ROWCOUNT IN H&L
0385 19                 DAD     D           ;ADD TOP+ROWCOUNT,  RESULT IN H&L
0386 22DA87             SHLD    LOC01       ;STORE LOCATION OF FIRST CHAR IN ROW
0389 EB                 XCHG                ;STORE TOP+ROWCOUNT IN D&E
038A 210000             LXI     H,0000H     ;ZERO H&L
038D 3AD287             LDA     CCTAD       ;LOAD COLUMN COUNT IN A
0390 6F                 MOV     L,A         ;MOVE COLUMN COUNT TO L
0391 19                 DAD     D           ;CALCULATE LOCATION=
                                            ;TOP+ROWCOUNT+COLUMN COUNT,RESULT IN H&L
0392 22D887             SHLD    LOCAD       ;STORE LOCATION IN MEMORY
0395 3E87               MVI     A,87H       ;LOAD 87H IN A
```

```
0397 BC          532          CMP     H              ;COMPARE H WITH 87H
0398 D2A103      533          JNC     NXTCM          ;IF NO CARRY,CONTNUE COMPARRISON
039B CDE603      534          CALL    COMRT          ;IF CARRY, CALL COMPENSATION ROUTINE
039E C3AD03      535          JMP     XSTAD          ;JUMP TO XSTAD
03A1 C2AD03      536 NXTCM:   JNZ     XSTAD          ;IF NOT EQUAL,END COMPARRISON
03A4 3ECF        537          MVI     A,0CFH         ;LOAD 0CFH IN A
03A6 BD          538          CMP     L              ;COMPARE L WITH 0CFH
03A7 D2AD03      539          JNC     XSTAD          ;IF NO CARRY, LOCATION LESS THAN
                 540                                 ;OR EQUAL TO 87CFH
03AA CDE603      541          CALL    COMRT          ;IF CARRY, CALL COMPENSATION ROUTINE
03AD CDFB03      542 XSTAD:   CALL    EORT           ;CALL END OF ROW CHAR TEST ROUTINE
03B0 21E587      543          LXI     H,USCHR        ;LOAD USART CHAR ADD IN H&L
03B3 7E          544          MOV     A,M            ;MOVE USART CHAR TO A
03B4 E63F        545          ANI     3FH            ;MASK OFF UPPER 2 BITS OF CHAR
03B6 2AD887      546          LHLD    LOCAD          ;LOAD LOCATION IN H&L
03B9 77          547          MOV     M,A            ;MOVE CHARACTER TO CHARACTER
                 548                                 ;LOCATION IN DISPLAY MEMORY
03BA C9          549          RET                    ;RETURN
                 550          ;
                 551          ;SUBROUTINE DISA
                 552          ;
03BB 21D287      553 DISA:    LXI     H,CCTAD        ;LOAD COLUMN COUNT ADD IN H&L
03BE 34          554          INR     M              ;INCREMENT COLUMN COUNT
03BF CD3C03      555          CALL    WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
03C2 C9          556          RET                    ;RETURN
                 557          ;
                 558          ;SUBROUTINE DISB
                 559          ;
03C3 3E00        560 DISB:    MVI     A,00H          ;ZERO A
03C5 32D287      561          STA     CCTAD          ;ZERO COLUMN COUNT
03C8 2AD387      562          LHLD    RCTAD          ;LOAD ROWCOUNT IN H&L
03CB 115000      563          LXI     D,50H          ;LOAD 80D=50H IN D&E
03CE 19          564          DAD     D              ;ADD +80 TO ROWCOUNT
03CF 22D387      565          SHLD    RCTAD          ;STORE ROWCOUNT IN MEMORY
03D2 21D587      566          LXI     H,CURSY        ;LOAD CURSOR Y POSITION ADDRESS IN H&L
03D5 34          567          INR     M              ;INCREMENT CURSOR Y POSITION
03D6 CD3C03      568          CALL    WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
03D9 C9          569          RET                    ;RETURN
                 570          ;
                 571          ;SUBROUTINE DISC
                 572          ;
03DA 3E00        573 DISC:    MVI     A,00H          ;ZERO A
03DC 32D287      574          STA     CCTAD          ;ZERO COLUMN COUNT
03DF CD3C03      575          CALL    WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
03E2 CD0B04      576          CALL    SCROL
03E5 C9          577          RET                    ;RETURN
                 578          ;
                 579          ;ADDRESS COMPENSATION SUBROUTINE
                 580          ;
03E6 2AD887      581 COMRT:   LHLD    LOCAD          ;LOAD CHARACTER LOCATION
03E9 1130F8      582          LXI     D,0F830H       ;LOAD COMPENSATION VALUE IN D&E
03EC 19          583          DAD     D              ;ADD COMPENSATION TO LOCATION
03ED 22D887      584          SHLD    LOCAD          ;STORE MODIFIED LOCATION IN MEMORY
                 585          ;
03F0 2ADA87      586          LHLD    LOC01          ;LOAD LOCATION OF FIRST CHAR
                 587                                 ;IN ROW IN H&L
03F3 1130F8      588          LXI     D,0F830H       ;LOAD COMPENSATION VALUE IN H&L
03F6 19          589          DAD     D              ;ADD COMPENSATION TO LOC01
03F7 22DA87      590          SHLD    LOC01          ;STORE MODIFIED LOC01 IN MEMORY
03FA C9          591          RET                    ;RETURN
                 592          ;
                 593          ;END OF ROW TEST ROUTINE
                 594          ;
03FB 2ADA87      595 EORT:    LHLD    LOC01          ;LOAD LOCATION OF FIRST CHAR
                 596                                 ;IN ROW IN H&L
03FE 7E          597          MOV     A,M            ;MOVE FIRST CHAR IN ROW TO A REG
03FF FEF0        598          CPI     0F0H           ;COMPARE CHAR WITH 0F0 (END OF ROW CHAR)
0401 C20A04      599          JNZ     XIT            ;IF NO COMPARRISON, EXIT
0404 22E287      600          SHLD    LOCBUF         ;STORE FIRST CHAR IN ROW ADD IN LOCBUF
0407 CD3204      601          CALL    FILL           ;CALL FILL ROW WITH SPACE CODES SUBROUTINE
040A C9          602 XIT:     RET                    ;RETURN
                 603          ;
                 604          ;SCROLL SUBROUTINE
                 605          ;
040B 2AD687      606 SCROL:   LHLD    TOPAD          ;LOAD TOP IN H&L
040E 22E287      607          SHLD    LOCBUF         ;STORE FIRST CHAR IN ROW ADD IN LOCBUF
0411 CD3204      608          CALL    FILL           ;CALL FILL ROW WITH SPACE CODES SUBROUTINE
0414 2AD687      609          LHLD    TOPAD          ;MOVE TOP TO H&L
0417 7D          610          MOV     A,L            ;MOVE LOWER BYTE OF TOP TO A
0418 FE80        611          CPI     80H            ;COMPARE TOP WITH MAX VALUE
041A C22A04      612          JNZ     DUCK           ;IF NO COMPARRISON EXISTS, CONTINUE SCROL
041D 7C          613          MOV     A,H            ;MOVE UPPER BYTE OF TOP TO A
041E FE87        614          CPI     87H            ;COMPARE TOP WITH MAX VALUE
0420 C22A04      615          JNZ     DUCK           ;IF NO COMPARRISON EXISTS, CONTINUE SCROL
                 616                                 ;IF COMPARRISON, TOP=MAX VALUE=8780H
0423 210080      617          LXI     H,8000H        ;IF COMPARRISON, MODIFY TOP TO TOP=8000H
0426 22D687      618          SHLD    TOPAD          ;STORE MODIFIED TOPAD IN MEMORY
0429 C9          619          RET                    ;RETURN
042A 115000      620 DUCK:    LXI     D,50H          ;MOVE 80D=50H TO D&E
```

```
042D 19          621          DAD     D              ;ADD 80D=50H TO TOP
042E 22D687      622          SHLD    TOPAD          ;STORE MODIFIED TOPAD IN MEMORY
0431 C9          623          RET                    ;RETURN
                 624          ;
                 625          ;FILL SUBROUTINE
                 626          ;
0432 2AE287      627 FILL:    LHLD    LOCBUF         ;LOAD LOCATION OF FIRST CHAR IN ROW
                 628                                 ;OR FIRST CHAR IN TOP ROW IN H&L
0435 115000      629          LXI     D,50H          ;LOAD 80D=50H IN D&E
0438 19          630          DAD     D              ;CALCULATE LOCATION OF LAST CHAR IN ROW
0439 22DC87      631          SHLD    LOC80          ;STORE LOCATION OF LAST CHAR IN ROW IN MEMORY
043C 012020      632          LXI     B,2020H        ;LOAD SPACE CHARACTERS IN B&C
043F 210000      633          LXI     H,0000H        ;ZERO H&L
0442 39          634          DAD     SP             ;ADD SP TO H&L, TRANSFERRING SP TO H&L
0443 EB          635          XCHG                   ;STORE STACK POINTER IN D&E
0444 2ADC87      636          LHLD    LOC80          ;LOAD LOCATION OF LAST CHAR IN ROW IN H&L
0447 F9          637          SPHL                   ;LOAD LAST CHAR LOCATION IN SP
                 638          ;
0448 C5          639          PUSH B                 ;EXECUTE THE LIST OF PUSH B COMMANDS TO
0449 C5          640          PUSH B                 ;FILL THE LINE WITH BLANK CHARACTERS
044A C5          641          PUSH B
044B C5          642          PUSH B
044C C5          643          PUSH B
044D C5          644          PUSH B
044E C5          645          PUSH B
044F C5          646          PUSH B
0450 C5          647          PUSH B
0451 C5          648          PUSH B
0452 C5          649          PUSH B
0453 C5          650          PUSH B
0454 C5          651          PUSH B
0455 C5          652          PUSH B
0456 C5          653          PUSH B
0457 C5          654          PUSH B
0458 C5          655          PUSH B
0459 C5          656          PUSH B
045A C5          657          PUSH B
045B C5          658          PUSH B
045C C5          659          PUSH B
045D C5          660          PUSH B
045E C5          661          PUSH B
045F C5          662          PUSH B
0460 C5          663          PUSH B
0461 C5          664          PUSH B
0462 C5          665          PUSH B
0463 C5          666          PUSH B
0464 C5          667          PUSH B
0465 C5          668          PUSH B
0466 C5          669          PUSH B
0467 C5          670          PUSH B
0468 C5          671          PUSH B
0469 C5          672          PUSH B
046A C5          673          PUSH B
046B C5          674          PUSH B
046C C5          675          PUSH B
046D C5          676          PUSH B
046E C5          677          PUSH B
046F C5          678          PUSH B
0470 EB          679          XCHG                   ;STACK POINTER TRANSFERRED TO H&L
0471 F9          680          SPHL                   ;RESTORE STACK
0472 C9          681          RET                    ;RETURN
                 682          ;
                 683          ;
                 684          ;
                 685          ;
                 686          ;8275 INTERRUPT SERVICE SUBROUTINE
                 687          ;
                 688          ;8257 REINITIALIZATION
                 689          ;
0473 3E00        690 RT75:    MVI     A,MDC57        ;MOVE MODE CLEAR COMMAND TO A
0475 D348        691          OUT     PMD57          ;OUTPUT MODE CLEAR COMMAND TO 8257
                 692          ;
0477 2AD687      693          LHLD    TOPAD          ;LOAD TOP IN H&L
047A 7D          694          MOV     A,L            ;LOAD CH 2 START ADD, LOW BYTE, IN A
047B D344        695          OUT     PC2SA          ;OUTPUT CH 2 START ADD TO 8257
047D 7C          696          MOV     A,H            ;LOAD CH 2 START ADD, UP BYTE, IN A
047E D344        697          OUT     PC2SA          ;OUTPUT CH 2 START ADD TO 8257
                 698          ;
0480 7D          699          MOV     A,L            ;LOAD LOW BYTE OF TOP IN A
0481 2F          700          CMA                    ;COMPLEMENT A
0482 6F          701          MOV     L,A            ;LOAD COMPLEMENTED VALUE IN L
0483 7C          702          MOV     A,H            ;LOAD UP BYTE OF TOP IN A
0484 2F          703          CMA                    ;COMPLEMENT A
0485 67          704          MOV     H,A            ;LOAD COMPLEMENTED VALUE IN H
0486 23          705          INX     H              ;INCREMENT H&L, YIELDING 2'S COMPLEMENT
                 706                                 ;OF TOP IN A
0487 11CF87      707          LXI     D,87CFH        ;LOAD 87CFH IN D&E
048A 19          708          DAD     D              ;ADD H&L TO D&E, YIELDING 87CFH-TOP
048B 110080      709          LXI     D,8000H        ;LOAD D&E WITH 8000H
048E 19          710          DAD     D              ;ADD 8000H TO 87CF-TOP
```

```
048F 7D          711          MOV     A,L          ;MOVE LOW BYTE OF CH 2 TC TO A
0490 D345        712          OUT     PC2TC        ;OUTPUT CH 2 TC TO 8257
0492 7C          713          MOV     A,H          ;MOVE UP BYTE OF CH 2 TC TO A
0493 D345        714          OUT     PC2TC        ;OUTPUT CH 2 TC TO 8257
                 715          ;
0495 210080      716          LXI     H,8000H      ;LOAD 8000HIN H&L
0498 7D          717          MOV     A,L          ;MOVE LOW BYTE OF CH 3 START ADD TO A
0499 D346        718          OUT     PC3SA        ;OUTPUT CH 3 START ADD TO 8257
049B 7C          719          MOV     A,H          ;MOVE UP BYTE OF CH 3 START ADD TO A
049C D346        720          OUT     PC3SA        ;OUTPUT CH 3 START ADD TO 8257
                 721          ;
049E 21CF87      722          LXI     H,87CFH      ;LOAD CH 3 TC VALUE IN H&L
04A1 7D          723          MOV     A,L          ;MOVE L TO A
04A2 D347        724          OUT     PC3TC        ;OUTPUT CH 3 TC TO 8257
04A4 7C          725          MOV     A,H          ;MOVE H TO A
04A5 D347        726          OUT     PC3TC        ;OUTPUT CH 3 TC TO 8257
04A7 3E84        727          MVI     A,MDS57      ;LOAD A WITH MODE SET VALUE
04A9 D348        728          OUT     PMD57        ;OUTPUT MODE SET TO 8257
                 729          ;
                 730          ;
                 731          ;KEYBOARD POLLING ROUTINE
                 732          ;
04AB DB6F        733 KPOLL:   IN      KCOM         ;INPUT FIFO STATUS
04AD E607        734          ANI     07H          ;MASK STATUS, SAVE BITS 0-2
04AF CAB504      735          JZ      ZIP          ;TEST FOR CHARACTER PRESENT
04B2 CDB604      736          CALL    XMIT         ;CALL CHARACTER TRANSMIT ROUTINE
04B5 C9          737 ZIP:     RET                  ;RETURN
                 738          ;
                 739          ;CHARACTER TRANSMIT SUBROUTINE
                 740          ;
04B6 DB6E        741 XMIT:    IN      KDAT         ;INPUT FIFO CHARACTER
04B8 EEC0        742          XRI     0C0H         ;INVERT TOP 2 BITS
04BA 21F804      743          LXI     H,BSET3      ;LOAD BASE ADD OF TABLE 3 IN H&L
04BD 110000      744          LXI     D,0000H      ;ZERO D&E
04C0 5F          745          MOV     E,A          ;LOAD E WITH CHARACTER FROM FIFO
04C1 19          746          DAD     D            ;CALCULATE ADD IN LOOKUP TABLE
                 747                               ;CONTAINING ASCII CHARACTERS
                 748                               ;CORRESPONDING TO KEY POSITION IN MATRIX
04C2 DBFB        749 USZ:     IN      CNCTL        ;INPUT USART STATUS
04C4 E601        750          ANI     01H          ;MASK STATUS, SAVE TRANSMITTER READY BIT
04C6 CAC204      751          JZ      USZ          ;TEST READY BIT
04C9 7E          752          MOV     A,M          ;MOVE ASCII CHAR TO A
04CA E67F        753          ANI     7FH          ;MASK BIT 7
04CC D3FA        754          OUT     CNOUT        ;OUTPUT CHAR FROM USART
04CE C9          755          RET                  ;RETURN
                 756          ;
                 757          ;DUMY ROUTINE DEFINITION
                 758          ;
04CF C9          759 DUMY:    RET                  ;RETURN
                 760          ;
                 761          ;
                 762          ;
                 763          ;
                 764          ;TABLE DEFINITION AREA
                 765          ;
                 766          ;
                 767          ;
04D0 CF04        768 BSET1:   DW      DUMY
04D2 3901        769          DW      ESCA
04D4 5F01        770          DW      ESCB
04D6 7D01        771          DW      ESCC
04D8 B001        772          DW      ESCD
04DA 3502        773          DW      ESCE
04DC CF04        774          DW      DUMY
04DE CF04        775          DW      DUMY
04E0 EB01        776          DW      ESCH
04E2 CF04        777          DW      DUMY
04E4 6602        778          DW      ESCJ
04E6 FD01        779          DW      ESCK
04E8 CF04        780          DW      DUMY
04EA CF04        781          DW      DUMY
04EC CF04        782          DW      DUMY
04EE CF04        783          DW      DUMY
                 784          ;
                 785          ;
04F0 0503        786 BSET2:   DW      CTRLH
04F2 F902        787          DW      CTRLJ
04F4 FC02        788          DW      CTRLM
04F6 CF04        789          DW      DUMY
                 790          ;
                 791          ;
04F8 30          792 BSET3:   DB      30H          ;DUMMY CHARACTER
04F9 30          793          DB      30H
04FA 30          794          DB      30H
04FB 30          795          DB      30H
04FC 30          796          DB      30H
04FD 30          797          DB      30H
04FE 30          798          DB      30H
04FF 30          799          DB      30H
```

```
0500 30          800        DB        30H
0501 30          801        DB        30H
0502 30          802        DB        30H
0503 30          803        DB        30H
0504 30          804        DB        30H
0505 30          805        DB        30H
0506 30          806        DB        30H
0507 30          807        DB        30H
0508 1B          808        DB        1BH          ; ESC
0509 0A          809        DB        0AH          ; LF
050A 2C          810        DB        2CH          ; ,
050B 0D          811        DB        0DH          ; CR
050C 20          812        DB        20H          ; SP
050D 7F          813        DB        7FH          ; DEL
050E 2E          814        DB        2EH          ; .
050F 2F          815        DB        2FH          ; /
0510 5A          816        DB        5AH          ; Z
0511 58          817        DB        58H          ; X
0512 4D          818        DB        4DH          ; M
0513 56          819        DB        56H          ; V
0514 30          820        DB        30H
0515 43          821        DB        43H          ; C
0516 4E          822        DB        4EH          ; N
0517 42          823        DB        42H          ; B
0518 30          824        DB        30H          ; 0
0519 2D          825        DB        2DH          ; -
051A 4F          826        DB        4FH          ; O
051B 4C          827        DB        4CH          ; L
051C 39          828        DB        39H          ; 9
051D 3A          829        DB        3AH          ; :
051E 50          830        DB        50H          ; P
051F 3B          831        DB        3BH          ; ;
0520 53          832        DB        53H          ; S
0521 44          833        DB        44H          ; D
0522 4B          834        DB        4BH          ; K
0523 47          835        DB        47H          ; G
0524 41          836        DB        41H          ; A
0525 46          837        DB        46H          ; F
0526 4A          838        DB        4AH          ; J
0527 48          839        DB        48H          ; H
0528 57          840        DB        57H          ; W
0529 45          841        DB        45H          ; E
052A 49          842        DB        49H          ; I
052B 54          843        DB        54H          ; T
052C 51          844        DB        51H          ; Q
052D 52          845        DB        52H          ; R
052E 55          846        DB        55H          ; U
052F 59          847        DB        59H          ; Y
0530 32          848        DB        32H          ; 2
0531 33          849        DB        33H          ; 3
0532 38          850        DB        38H          ; 8
0533 35          851        DB        35H          ; 5
0534 31          852        DB        31H          ; 1
0535 34          853        DB        34H          ; 4
0536 37          854        DB        37H          ; 7
0537 36          855        DB        36H          ; 6
0538 30          856        DB        30H
0539 30          857        DB        30H
053A 30          858        DB        30H
053B 30          859        DB        30H
053C 30          860        DB        30H
053D 30          861        DB        30H
053E 30          862        DB        30H
053F 30          863        DB        30H
0540 30          864        DB        30H
0541 30          865        DB        30H
0542 30          866        DB        30H
0543 30          867        DB        30H
0544 30          868        DB        30H
0545 30          869        DB        30H
0546 30          870        DB        30H
0547 30          871        DB        30H
0548 30          872        DB        30H
0549 30          873        DB        30H
054A 3C          874        DB        3CH          ; <
054B 30          875        DB        30H
054C 30          876        DB        30H
054D 30          877        DB        30H
0517 42          823        DB        42H          ; B
0518 30          824        DB        30H          ; 0
0519 2D          825        DB        2DH          ; -
051A 4F          826        DB        4FH          ; O
051B 4C          827        DB        4CH          ; L
051C 39          828        DB        39H          ; 9
051D 3A          829        DB        3AH          ; :
051E 50          830        DB        50H          ; P
051F 3B          831        DB        3BH          ; ;
0520 53          832        DB        53H          ; S
0521 44          833        DB        44H          ; D
0522 4B          834        DB        4BH          ; K
```

```
0523 47          835        DB      47H          ; G
0524 41          836        DB      41H          ; A
0525 46          837        DB      46H          ; F
0526 4A          838        DB      4AH          ; J
0527 48          839        DB      48H          ; H
0528 57          840        DB      57H          ; W
0529 45          841        DB      45H          ; E
052A 49          842        DB      49H          ; I
052B 54          843        DB      54H          ; T
052C 51          844        DB      51H          ; Q
052D 52          845        DB      52H          ; R
052E 55          846        DB      55H          ; U
052F 59          847        DB      59H          ; Y
0530 32          848        DB      32H          ; 2
0531 33          849        DB      33H          ; 3
0532 38          850        DB      38H          ; 8
0533 35          851        DB      35H          ; 5
0534 31          852        DB      31H          ; 1
0535 34          853        DB      34H          ; 4
0536 37          854        DB      37H          ; 7
0537 36          855        DB      36H          ; 6
0538 30          856        DB      30H
0539 30          857        DB      30H
053A 30          858        DB      30H
053B 30          859        DB      30H
053C 30          860        DB      30H
053D 30          861        DB      30H
053E 30          862        DB      30H
053F 30          863        DB      30H
0540 30          864        DB      30H
0541 30          865        DB      30H
0542 30          866        DB      30H
0543 30          867        DB      30H
0544 30          868        DB      30H
0545 30          869        DB      30H
0546 30          870        DB      30H
0547 30          871        DB      30H
0548 30          872        DB      30H
0549 30          873        DB      30H
054A 3C          874        DB      3CH          ; <
054B 30          875        DB      30H
054C 30          876        DB      30H
054D 30          877        DB      30H
054E 3E          878        DB      3EH          ; >
054F 3F          879        DB      3FH          ; ?
0550 30          880        DB      30H
0551 30          881        DB      30H
0552 5D          882        DB      5DH          ; ]
0553 30          883        DB      30H
0554 30          884        DB      30H
0555 30          885        DB      30H
0556 30          886        DB      30H
0557 30          887        DB      30H
0558 30          888        DB      30H
0559 3D          889        DB      3DH          ; =
055A 30          890        DB      30H
055B 5C          891        DB      5CH          ; \
055C 29          892        DB      29H          ; )
055D 2A          893        DB      2AH          ; *
055E 30          894        DB      30H
055F 2B          895        DB      2BH          ; +
0560 30          896        DB      30H
0561 30          897        DB      30H
0562 5B          898        DB      5BH          ; [
0563 30          899        DB      30H
0564 30          900        DB      30H
0565 30          901        DB      30H
0566 30          902        DB      30H
0567 30          903        DB      30H
0568 30          904        DB      30H
0569 30          905        DB      30H
056A 30          906        DB      30H
056B 30          907        DB      30H
056C 30          908        DB      30H
056D 30          909        DB      30H
056E 30          910        DB      30H
056F 30          911        DB      30H
0570 22          912        DB      22H          ; "
0571 23          913        DB      23H          ; #
0572 28          914        DB      28H          ; (
0573 25          915        DB      25H          ; %
0574 21          916        DB      21H          ; !
0575 24          917        DB      24H          ; $
0576 27          918        DB      27H          ; '
0577 26          919        DB      26H          ; &
0578 30          920        DB      30H
0579 30          921        DB      30H
057A 30          922        DB      30H
057B 30          923        DB      30H
057C 30          924        DB      30H
```

```
057D 30          925        DB        30H
057E 30          926        DB        30H
057F 30          927        DB        30H
0580 30          928        DB        30H
0581 30          929        DB        30H
0582 30          930        DB        30H
0583 30          931        DB        30H
0584 30          932        DB        30H

0585 30          933        DB        30H
0586 30          934        DB        30H
0587 30          935        DB        30H
0588 30          936        DB        30H
0589 30          937        DB        30H
058A 30          938        DB        30H
058B 30          939        DB        30H
058C 30          940        DB        30H
058D 30          941        DB        30H
058E 30          942        DB        30H
058F 30          943        DB        30H
0590 1A          944        DB        1AH          ; SUB
0591 18          945        DB        18H          ; CAN
0592 0D          946        DB        0DH          ; CR
0593 16          947        DB        16H          ; SYN
0594 30          948        DB        30H
0595 03          949        DB        03H          ; ETX
0596 0E          950        DB        0EH          ; SO
0597 02          951        DB        02H          ; STX
0598 30          952        DB        30H
0599 1F          953        DB        1FH          ; US
059A 0F          954        DB        0FH          ; SI
059B 0C          955        DB        0CH          ; FF
059C 30          956        DB        30H
059D 30          957        DB        30H
059E 10          958        DB        10H          ; DLE
059F 30          959        DB        30H
05A0 13          960        DB        13H          ; DC3
05A1 04          961        DB        04H          ; EOT
05A2 0B          962        DB        0BH          ; VT
05A3 07          963        DB        07H          ; BEL
05A4 01          964        DB        01H          ; SOH
05A5 06          965        DB        06H          ; ACK
05A6 0A          966        DB        0AH          ; LF
05A7 08          967        DB        08H          ; BS
05A8 17          968        DB        17H          ; ETB
05A9 05          969        DB        05H          ; ENQ
05AA 09          970        DB        09H          ; HT
05AB 14          971        DB        14H          ; DC4
05AC 11          972        DB        11H          ; DC1
05AD 12          973        DB        12H          ; DC2
05AE 15          974        DB        15H          ; NAK
05AF 19          975        DB        19H          ; EM
05B0 30          976        DB        30H
05B1 30          977        DB        30H
05B2 30          978        DB        30H
05B3 30          979        DB        30H
05B4 30          980        DB        30H
05B5 30          981        DB        30H
05B6 30          982        DB        30H
05B7 30          983        DB        30H
05B8 30          984        DB        30H
05B9 30          985        DB        30H
05BA 30          986        DB        30H
05BB 30          987        DB        30H

05BC 30          988        DB        30H
05BD 30          989        DB        30H
05BE 30          990        DB        30H
05BF 30          991        DB        30H
05C0 30          992        DB        30H
05C1 30          993        DB        30H
05C2 30          994        DB        30H
05C3 30          995        DB        30H
05C4 30          996        DB        30H
05C5 30          997        DB        30H
05C6 30          998        DB        30H
05C7 30          999        DB        30H
05C8 30          1000       DB        30H
05C9 30          1001       DB        30H
05CA 30          1002       DB        30H
05CB 30          1003       DB        30H
05CC 30          1004       DB        30H
05CD 30          1005       DB        30H
05CE 30          1006       DB        30H
05CF 30          1007       DB        30H
05D0 30          1008       DB        30H
05D1 30          1009       DB        30H
05D2 1D          1010       DB        1DH          ; GS
05D3 30          1011       DB        30H
05D4 30          1012       DB        30H
05D5 30          1013       DB        30H
```

```
05D6 30          DB      30H
05D7 30          DB      30H
05D8 30          DB      30H
05D9 30          DB      30H
05DA 30          DB      30H
05DB 1C          DB      1CH             ; FS
05DC 30          DB      30H
05DD 30          DB      30H
05DE 30          DB      30H
05DF 30          DB      30H
05E0 30          DB      30H
05E1 30          DB      30H
05E2 1B          DB      1BH             ; ESC
05E3 30          DB      30H
05E4 30          DB      30H
05E5 30          DB      30H
05E6 30          DB      30H
05E7 30          DB      30H
05E8 30          DB      30H
05E9 30          DB      30H
05EA 30          DB      30H
05EB 30          DB      30H
05EC 30          DB      30H
05ED 30          DB      30H
05EE 30          DB      30H
05EF 30          DB      30H
05F0 30          DB      30H
05F1 30          DB      30H
05F2 30          DB      30H
05F3 30          DB      30H
05F4 30          DB      30H
05F5 30          DB      30H
05F6 30          DB      30H
05F7 30          DB      30H
                 ;
                 ;
                 ;
                 ;
                 ;DATA SEGMENT AREA
                 ;
87D2             ORG     87D2H
                 ;
0001             CCTAD:  DS      1
0002             RCTAD:  DS      2
0001             CURSY:  DS      1
0002             TOPAD:  DS      2
0002             LOCAD:  DS      2
0002             LOC01:  DS      2
0002             LOC80:  DS      2
0002             LOCXX:  DS      2
0002             LOCPR:  DS      2
0002             LOCBUF: DS      2
0001             XFLG:   DS      1
0001             USCHR:  DS      1
0002             BOTAD:  DS      2
                 END
```

Printer Control with the UPI-41

by Lionel Smith

INTRODUCTION

The UPI-41 is a low-cost, single-chip microcomputer designed to be used as a universal peripheral interface device in a microcomputer system. The device is based on a completely self-contained 8-bit microcomputer with program memory, data memory, CPU, I/O, event timer, and clock oscillator, in a single 40-pin package. A bus interface is included which enables the UPI-41 to be used as a peripheral controller in MCS-48, MCS-80, MCS-85 and other 8-bit microcomputer families. The device is designed for keyboard scanning, printer control, display multiplexing and similar applications which involve interfacing peripheral devices to microcomputer systems.

The UPI-41 is fabricated with N-channel MOS technology and requires only a single 5-volt supply for operation. It has 1K words of program memory and 64 words of data memory on-chip. Both ROM (8041) and EPROM (8741) versions are available and the two are completely pin compatible. The instruction set of the UPI-41 is almost identical to that of the MCS-48. A single byte data register on the UPI-41 interfaces directly to an 8-bit master processor bus to handle asynchronous data transfer to and from the master system. A separate 4-bit register is used to indicate the status of data transfer. Two 8-bit TTL-compatible I/O ports plus two single-bit test inputs are available. I/O can be expanded further by using the 8243 I/O expander device. A separate register in the UPI-41 is used as an event counter or interval timer.

Because it is a complete microcomputer, the UPI-41 provides more power and flexibility than conventional LSI interface devices. For instance, the UPI-41 can be programmed as a peripheral interface for any of the low-cost drum or dot matrix printers currently on the market. In addition to controlling the printer, the UPI-41 can handle zero suppression, limit-checking, formatting and other computations, thereby unburdening the master processor. This type of distributed intelligence, made possible by the UPI-41, greatly enhances overall system capability while reducing cost and development time.

This application note describes how the UPI-41 can be used to implement an interface to a matrix printer. The printer chosen is fairly typical of a large class of printers which minimize total system cost by reducing the mechanical content at the expense of more sophisticated electronic requirements. The UPI-41, with its high degree of capability, is ideal for this type of application. It is suggested that the reader not already familiar with the UPI-41 read the "Intel UPI-41 User's Manual" before proceeding in this document.

THE LRC PRINTER

The LRC Model 7040 printer is a matrix printer manufactured by LRC Inc. of Riverton, Wyoming. Capable of printing up to 40 columns of alphanumeric information, this printer is mechanically simple and should be ideal for a variety of applications such as point of sale terminals and data logging. While this note concentrates on the Model 7040 printer, the techniques discussed should be applicable to a variety of similar printers which are currently available.

The printer (Figure 1) consists of four major subassemblies, the frame, the print head, the main drive, and the paper handling components. The frame is an aluminum extrusion which provides a suitable base for mounting the various components of the printer. The print head consists of seven solenoids which each drive stiff wires to impact the paper through the inked ribbon. At the solenoid end of the print head these wires are arranged in a circular fashion. Where these wires impact the printer, however, the wires are arranged in a vertical column. To see how this arrangement can be used to print alphanumeric characters refer to Figure 2. The figure shows a 5 X 7 matrix of "dots". The columns are labeled C1 through C5; the rows are labeled as Row 1 through Row 7. Each row corresponds to one of the solenoid-driven wires. The entire print head assembly is moved left to right across the paper so that at T_1 it is over C1, at T_2 it is over C2, and so on. If the correct solenoids are activated at each of these times (T_1–T_5) then a character can be formed. Figure 2 shows the character "A" formed. At T_1 solenoids one through five were active, at T_2 solenoids four and six were active, and so on until the complete character was formed. The complete character is formed by choosing the correct pattern of active solenoids for each of five instants in time.

The print head is moved across the paper by the main drive. The main drive consists of a 24-pole synchronous motor which drives a rotating plastic drum. The drum has a spiral groove molded into it. A pin attached to the print head rests in this groove so that as the drum rotates at a constant speed the print head is driven back and forth across the paper. Printing is accomplished by controlling

the activation of the solenoids as the print head is driven from left to right across the paper. When the end of the print area occurs the spiral groove reverses the direction of the head motion. As the left-hand edge of the paper is reached a cam attached to the drum activates the HOME microswitch and the groove again reverses the motion of the head. When the print head is again over the print area and travelling in the left to right direction the microswitch is deactivated. The printer controller uses the trailing edge of the signal generated by the microswitch to initiate the printing of a new line of information.

Paper feed is accomplished by a second synchronous motor which can be activated to feed paper through the mechanism. A switch is provided which is activated while the actual line feed is occurring. The control logic can use the trailing edge of the signal generated by this switch to turn off the line feed motor. A version of the printer with automatic line feed is available.

INTERFACE SIGNALS

The interface signals to the printer consists of a pair of wires for each solenoid, a pair of wires for each motor (main drive and line feed), a pair of wires returning the state of the HOME microswitch, and a pair of wires returning the state of the LINEFEED microswitch.

The solenoids must be driven from a 40 ±4 volt source. The peak current is approximately 3.6A, the average current is approximately 0.5A. A circuit providing the required drive is shown in Figure 3. The output stage, consisting of the 2N6045 Darlington transistor, the 1N4002 catching diode, and the 20-ohm damping resistor, is the

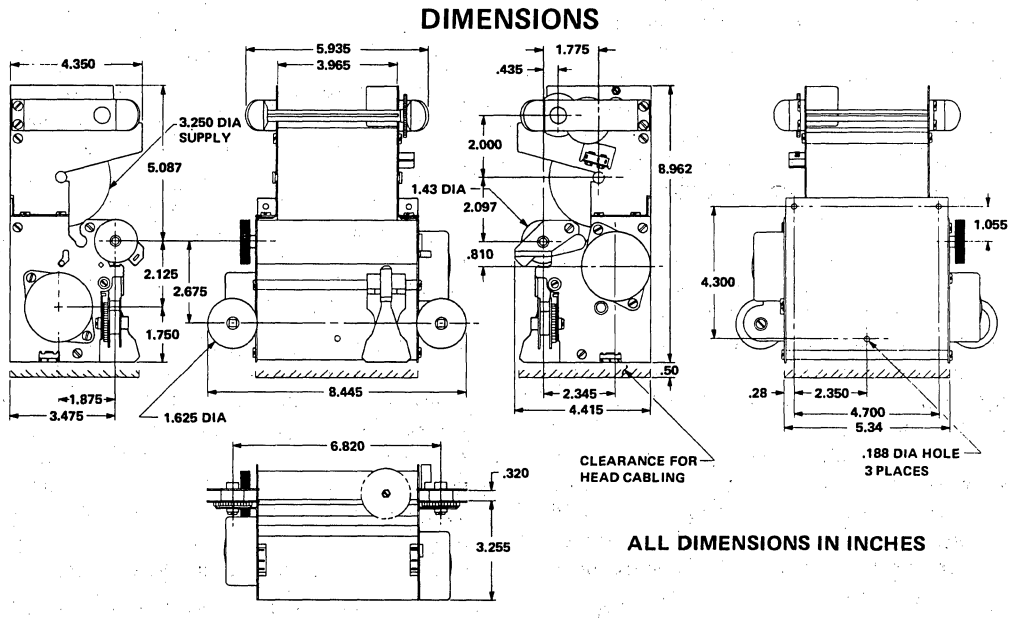


Figure 1. LRC Model 7040 Printer

one suggested by the manufacturer of the printer. The input stage is a discrete implementation of a DTL gate. Note that the base-emitter junction of the 2N6045 will protect the 2N2222A transistor from over-voltage on its collector. This circuit has several features which are important to the printer interface:

1. All solenoid power (including the power used to drive the base of the power transistor) is derived from the 40-volt supply.
2. Disconnecting the drivers from the UPI-41 or the loss of the 5-volt supply to the UPI-41 will result in the solenoids being turned off.

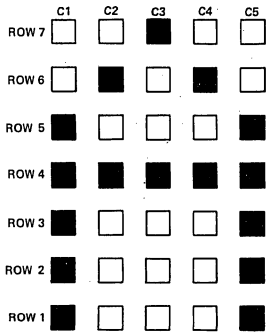


Figure 2. 5 x 7 Dot Matrix

The first feature of the drivers will minimize the impact of the printer and its interface on the 5-volt supply of the system. The second feature prevents the activation of the solenoids erroneously during power on/off cycles or during system checkout. This is an important point since the solenoids will be damaged if left activated continuously. (During the debug of the design described in this note fuses were added to the solenoid drivers to protect them from mishap.)

The two motors can each be driven as shown in Figure 4. The Monsanto MCS-6200 is an optically-coupled TRIAC which is ideal for driving the small synchronous motors in the printer. Coupled with a buffer this part provides a simple means of controlling the motor without sacrificing the isolation required for safe and reliable operation.

Figure 5 shows a UPI-41 used as an interface between an Intel® 8085 and an LRC Model 7040 printer. The drivers which have already been described have been used to interface the TTL outputs of the 8741 to the levels required by the printer. The two contact closure outputs from the printer (PAPERFEED and HOME) have been filtered and applied to the TEST0 and TEST1 inputs of the UPI-41. Bit 5 of output port 2 has been designated as an interrupt pin which will be used to request service from the 8085.

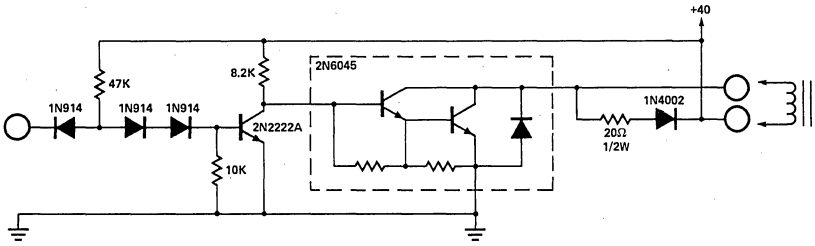


Figure 3. Solenoid Driver

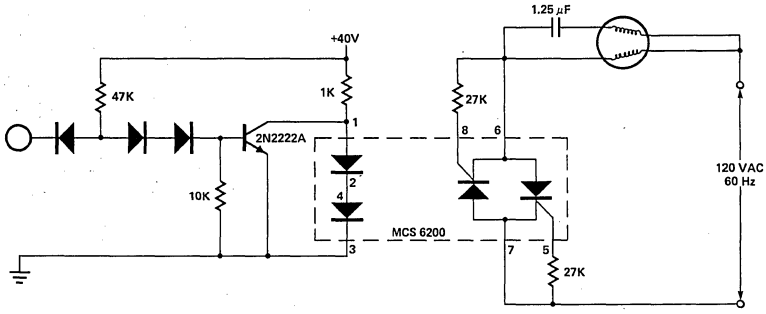


Figure 4. Motor Driver

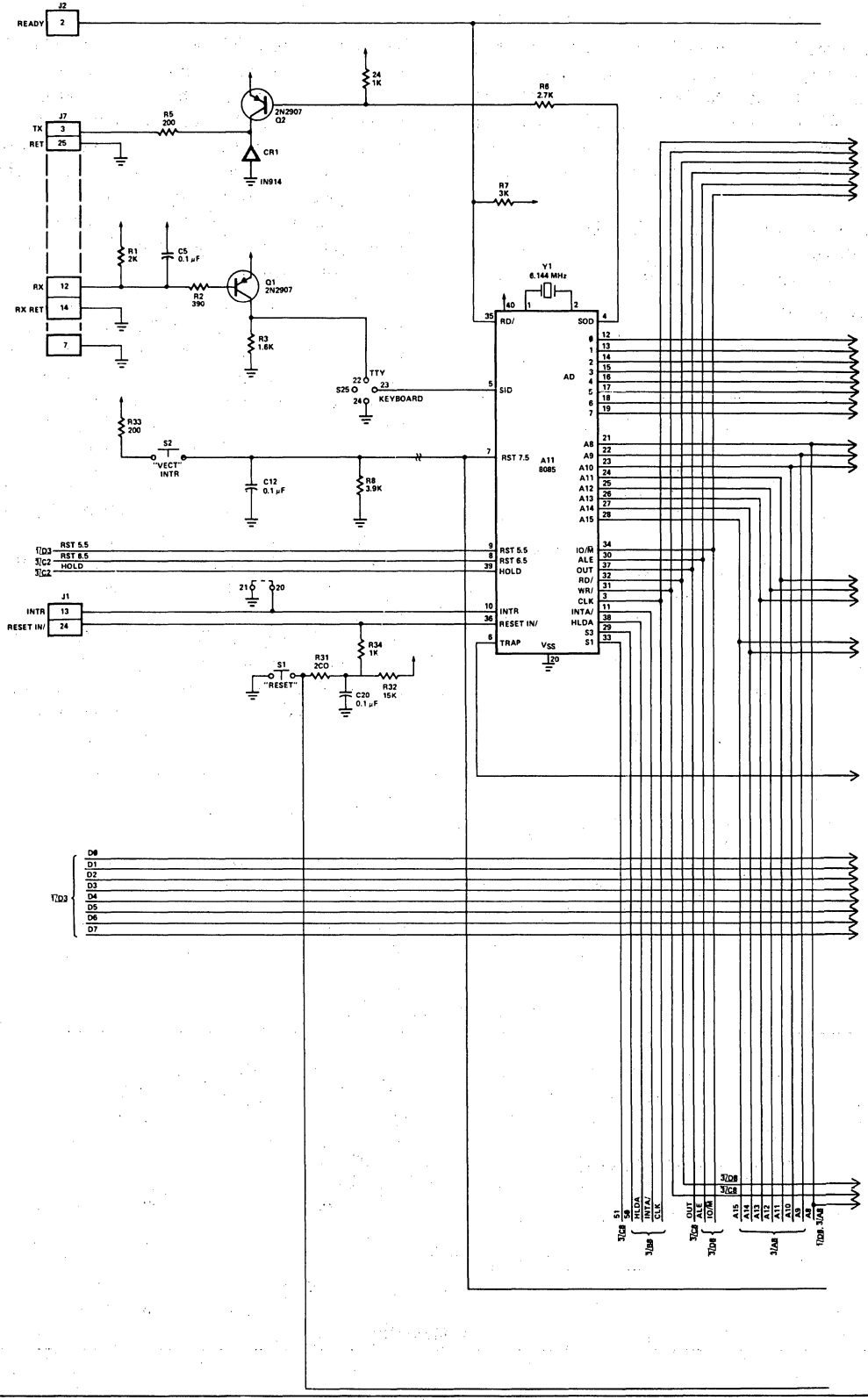
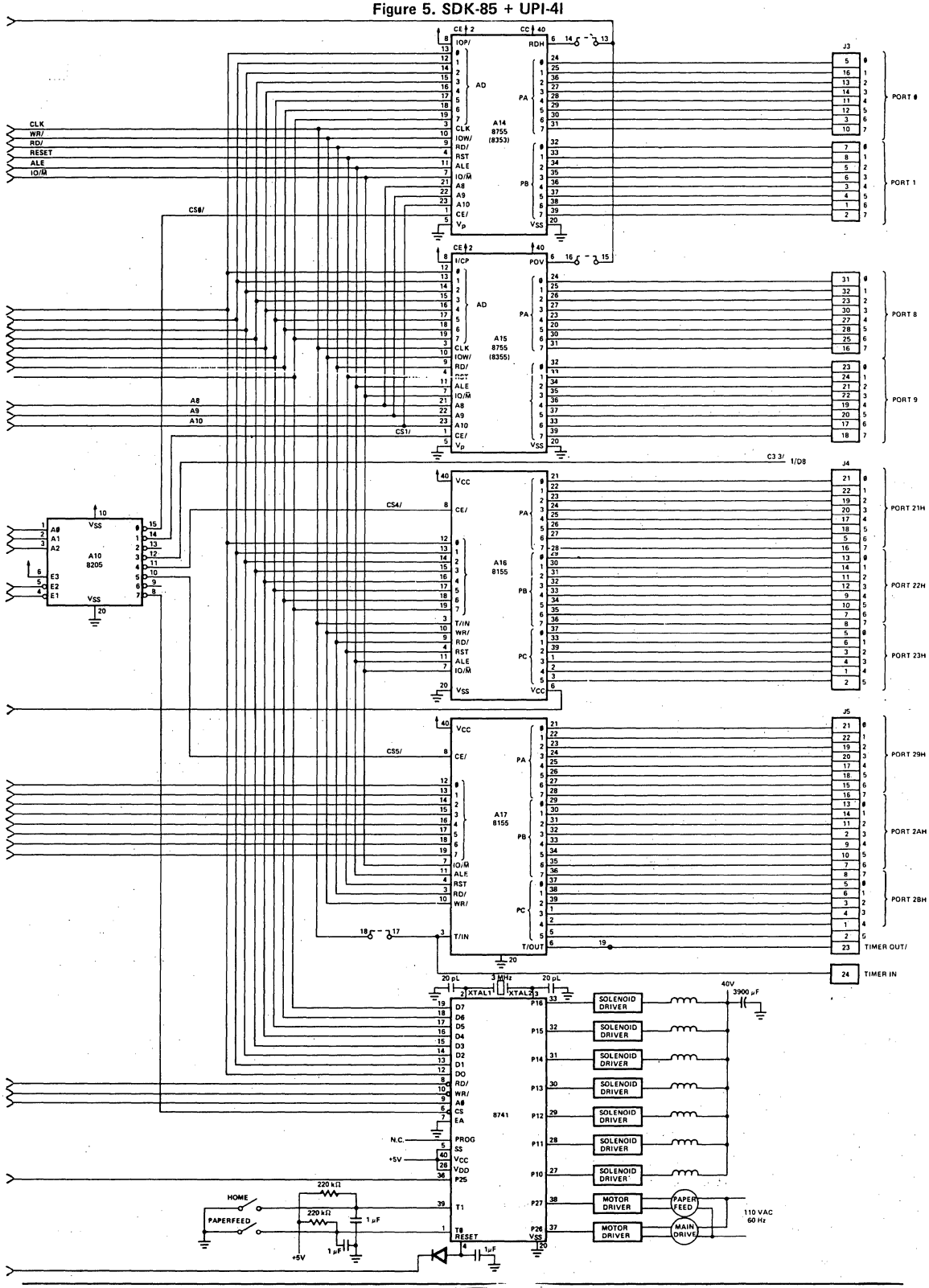

J2
READY
2
J7
TX
3
RET
25
R5
200
2N2907
Q2
CR1
IN914
24
1K
R6
2.7K
R7
3K
R1
2K
C5
0.1 μF
RX
12
RX RET
14
7
R2
390
Q1
2N2907
R3
1.6K
Y1
6.144 MHz
40
1
2
35
RD/
SOD
4
AD
TTY
22
23
S25
24
KEYBOARD
5
SID
R33
200
S2
"VECT"
INTR
7
RST 7.5
A11
8085
C12
0.1 μF
R8
3.9K
RST 5.5
RST 6.5
HOLD
9
8
39
IO/M
ALE
OUT
RD/
WR/
CLK
INTA/
HLDA
S3
S1
34
30
37
32
31
3
11
38
29
33
21
20
J1
INTR
13
RESET IN/
24
10
36
INTR
RESET IN/
6
TRAP
VSS
20
R34
1K
S1
"RESET"
R31
2CO
C20
0.1 μF
R32
15K
D0
D1
D2
D3
D4
D5
D6
D7
3/D8
3/C8
S1
S0
HLDA
INTA/
CLK
OUT
ALE
IO/M
A15
A14
A13
A12
A11
A10
A9
A8
1/D8, 3/A8

Figure 5. SDK-85 + UPI-41



TIMING

The relative timing of the interface signals to the printer is shown in Figure 6. Actual printing commences when the main drive switch signal goes into the print ready state. This edge indicates that the print head is scanning across the paper in the left to right direction and that the printer is ready to start the actual printing of characters. When this edge occurs the UPI-41 must start transmitting pulses to each of the seven solenoids. The timing for these pulses is shown on the last line of Figure 6. A pulse of about 400 microseconds is used to generate a dot on the paper; a pause of about 900 microseconds between these pulses satisfies the duty cycle restrictions of the solenoids and provides a space between dots. Since the printer does not provide any feedback to the UPI-41 which would indicate the position of the print head, it is necessary for the UPI-41 to decide when to fire each solenoid based on timing information it maintains internally. The specifications of the printer allow 310 milliseconds for the print head to traverse the print area. The maximum repetition rate at which the solenoids can be fired is once every 1.3 milliseconds. The maximum number of dots that can be printed in the available print area is then 310/1.3 = 238. After the last dot has been printed the line feed motor can be activated. The motor should remain activated until the line feed switch makes the off to on to off transition; this takes about 200 milliseconds. After the line feed motor is deactivated the next time of interest is when the main drive signal goes to the inactive state. At this point the printing of a complete line, including the necessary line feed, has been accomplished and the UPI-41 must prepare itself for the reactivation of the main drive switch. The activation of this switch will indicate that the printing of the next line can commence.

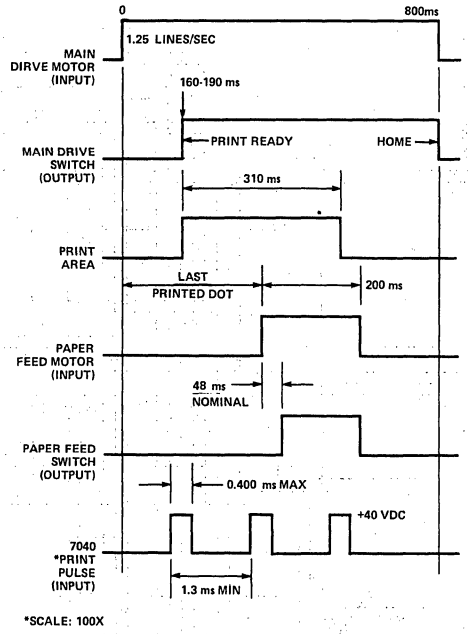


Figure 6. Printer Timing

SOFTWARE

The software system necessary to drive the LRC printer can be thought of as two main parts, each with an associated data structure. A block diagram of the system is shown in Figure 7. All the items shown above the dotted line are associated with the BUFFER MANAGER (BMGR) program part. All items shown below the dotted line are associated with a PRINTER SERVICE ROUTINE (PSR).

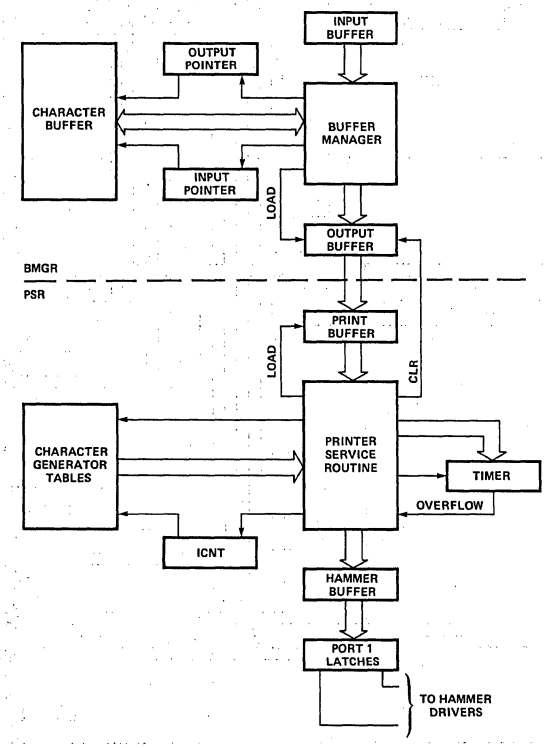


Figure 7. Software Block Diagram

The BUFFER MANAGER is responsible for all interaction with the master processor (i.e., the 8085 in Figure 5). The data structure associated with BMGR is a 40-character buffer which is used to store the characters as they are received from the master processor. BMGR maintains two pointers which are used to access the buffer; these pointers are shown as INPUT POINTER and OUTPUT POINTER in the diagram and are implemented as UPI-41 registers R_0 and R_1, respectively. The input pointer (INPNT) is kept pointing to the last character loaded into the buffer, the output pointer (OUTPNT) is kept pointing to the next character to be printed. BMGR has two major interfaces, the INPUT BUFFER, which is used to communicate with the master processor, and the register shown in the figure as OUTPUT BUFFER. This register, which is implemented with register R_3 of the UPI-41, is used to communicate with the printer service routine (PSR). A character to be printed is placed in the output buffer (OBUF). When PSR is ready to print the character it moves it from OBUF to its own buffer (PBUF) which is labeled as PRINT BUFFER in the diagram. After the character is moved the output buffer is overwritten by a predetermined value which indicates that PSR has accepted the character. BMGR will load a character into the output buffer only if it currently is equal to this value.

The printer service routine utilizes the TIMER to keep track of the current position of the print head. At the appropriate times it causes the solenoid drivers to be pulsed so that the character stream it sees in PBUF is printed. Based on the contents of PBUF and the contents of ICNT, which indicates the active column of the current character, PSR looks up the appropriate column data to be printed in the character generator tables. This data is stored in the HAMMER BUFFER until the precise time that it should be presented to the hammer drivers via the I/O bits in PORT 1. ICNT and the HAMMER BUFFER are implemented as UPI-41 registers 5 and 7, respectively.

DETAILS OF THE BUFFER MANAGER

Before BMGR can be discussed in detail, the manner in which it utilizes the character buffer must be understood. Figure 8 shows the operation of the buffer while two lines of data are input to the UPI-41 and subsequently printed. In order to keep the discussion manageable, this figure is drawn as if the printer were capable of printing only four characters per line. The two lines of characters to be printed are:

ABCD
1234

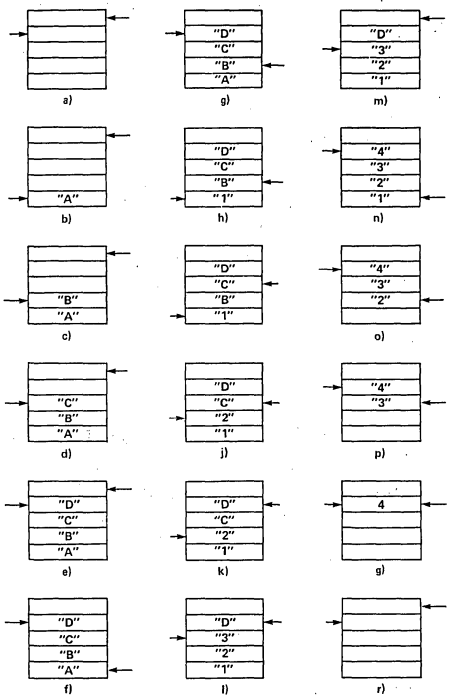


Figure 8. Buffer Operation

It should be noted that the buffer contains 5 bytes, one more than the number of print positions. The extra byte is a "phantom address" which, when pointed to by the output pointer, indicates that the section of BMGR which services the printer service routine is inactive. This state must be allowed because the actual print operation cannot begin until the complete line has been input to the buffer. If this rule were not enforced, some underrun protocol would have to be established to handle the situation of the input stream from the master processor failing to keep up with the print head.

Figure 8a shows the buffer in its initial state. The input pointer is set to the last real position in the buffer and the output pointer is set to the phantom position. Figures 8b through 8f show the operation of the pointers as the characters "A", "B", "C", and "D" are loaded. In each case the

input pointer is incremented to point to the next available location and then that location is loaded with the character. The position of the output pointer is not changed until the last position of the buffer has been loaded. When this occurs, the output pointer is set to point at the first character of the buffer. The operation of the pointers thus far can be described by the following algorithm:

```
INITIAL:
   INPOINT:=BUFFER_MAX;
   OUTPOINT:=BUFFER_MAX+1;
   ...
   LOOP:
   IF CHARACTER_AVAILABLE THEN
   BEGIN
       INPOINT:=(INPOINT+1) MOD BUFFER_LENGTH;
       BUFFER(INPOINT):=CHARACTER;
       IF INPOINT=BUFFER_MAX THEN OUTPOINT:=BUFFER_MIN;
   END;
   GOTO LOOP;
END;
```

Obviously, if this loop were allowed to continue, the buffer would be overwritten by the next line of text before the first could be printed. This can be prevented by modifying the algorithm as follows:

```
...
LOOP:
IF CHARACTER_AVAILABLE THEN
BEGIN
    TEMP:=(INPOINT+1) MOD BUFFER_LENGTH;
    IF TEMP<>OUTPOINT THEN
    BEGIN
        INPOINT:=TEMP;
        BUFFER(INPOINT):=CHARACTER;
        IF INPOINT=BUFFER_MAX THEN OUTPOINT:=BUFFER_MIN;
    END;
END;
GOTO LOOP;
```

This modification will "freeze the action" at Figure 8f until the output pointer is incremented. When this occurs the input procedure will immediately load the input data over the character that was just printed (assuming that data is available to the procedure at a higher rate than can be printed). The defined interface with the printer service routine allows a character to be removed from the buffer and placed in the output buffer whenever the output buffer contains the value placed there by the PSR, indicating that it has accepted the character that was previously in the output buffer. If this value is called EMPTY_FLAG then the complete buffer handling procedure can be defined as follows:

```
INITIAL:
INPOINT:=BUFFER_MAX;
OUTPOINT:=BUFFER_MAX+1;
   ...
   LOOP:
   IF CHARACTER_AVAILABLE THEN
   BEGIN
       TEMP:=(INPOINT+1) MOD BUFFER_LENGTH;
       IF TEMP<>OUTPOINT THEN
       BEGIN
           INPOINT:=TEMP;
           BUFFER(INPOINT):=CHARACTER;
           IF INPOINT=BUFFER_MAX THEN
             OUTPOINT:=BUFFER_MIN;
       END;
       IF OUTPUT_BUFFER=EMPTY_FLAG THEN
       BEGIN
           IF OUTPOINT<=BUFFER_MAX THEN
           BEGIN
               OUTPUT_BUFFER:=BUFFER(OUTPOINT);
               OUTPOINT:=OUTPOINT+1;
           END;
       END;
   END;
   GOTO LOOP:
```

Examination of Figures 8g through 8r will show how this algorithm maintains the buffer. If there is an open position and a character is available, it is placed in the buffer. When a complete line is in the buffer, printing is initialized by setting the output pointer to BUFFER_MIN. As the last character of a line is printed, the output pointer is incremented to point at the "phantom location" until the next line is completely entered. It should also be noted that if the input stream is faster than the print operation, then after the last character of a line is printed only one character need be input before printing can resume (see Figures 8l, m, and n). Frame r shows that after all available characters have been printed the state of the buffer is the same as it is initially. This is obviously a desirable feature.

The flowcharts for the complete BUFFER MANAGER are shown in Figures 9a and 9b. The corresponding code can be found starting at label BMGR of the program listings (see appendix). The flowcharts follow the algorithm that has been discussed very closely. Some additions have been made to implement logic not associated with the buffer. The first difference is that when a byte is in the input buffer it is tested to determine whether it is a command byte or a data character before further action is taken. Only two commands are recognized; one to set, and one to reset, the internal interrupt enable flag. This flag, which is

implemented as bit zero of PORT2 determines whether or not the UPI-41 will assert an interrupt to the master processor when it is able to accept a new character. Two additional deviations can be noted in Figure 9a; the first is that the motor of the printer will be turned on whenever a data character is received, the second is that if an end of line code (i.e., an ASCII line feed) is received, then, instead of storing it in the buffer, a mode is entered which fills the remaining buffer locations with space characters. This mode is enabled by bit one of PORT2. Note that utilizing otherwise unused bits of PORT2 for program status allows convenient testing and setting by the software and also enables external monitoring of the program operation.

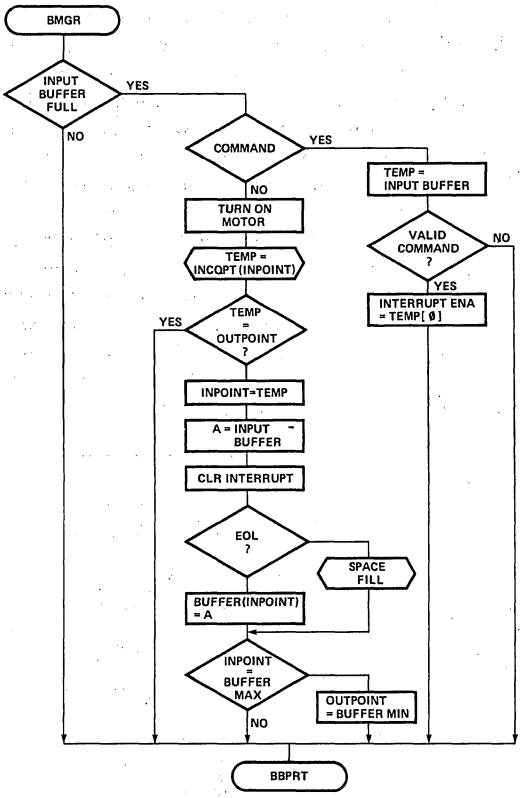


Figure 9a. Buffer Manager Flowchart

The last addition to the algorithm can be seen in Figure 9b where instead of going directly back to the start of the program after servicing the printer, a test is made to determine if the interrupt to the master processor should be asserted. This interrupt is set if the enable bit is set and there is also room in the buffer for at least one more character. After this test, control is passed back to the beginning of BMGR.

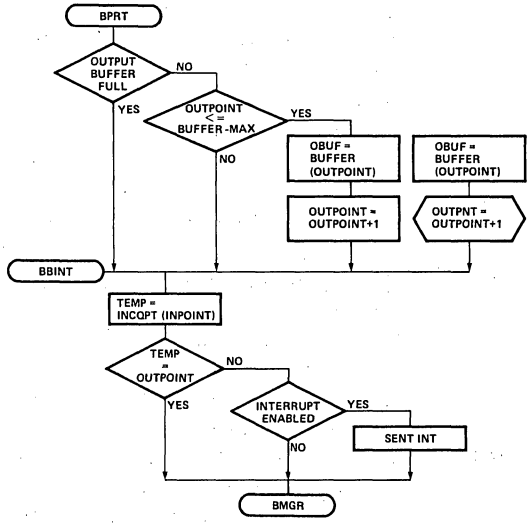


Figure 9b. Buffer Manager Flowchart

PRINTER SERVICE ROUTINES

The Printer Service Routine must convert the characters given to it by the Buffer Manager into an appropriately timed stream of pulses to the solenoids. Because the PSR is extremely time-dependent, it was implemented as an interrupt-driven routine which is given control when the timer overflow occurs. This allows exact timing of the solenoid firings without requiring software delay loops. If the timing had been generated by such loops, synchronization would have been lost when the delay loops were interrupted in order to service the master processor.

If a hardware design of a controller for the printer were being undertaken, a convenient place to start would be to generate a state transition diagram which shows all the states that can be entered and how control can transfer from state to state. This hardware design technique is often useful in software design and was, in fact, used to develop the PSR. The state diagram of the PSR is shown in Figure 10. A total of eight states are necessary to implement the printer control function. Before discussing this diagram further, each of these states must be defined.

WPA: The WPA (Wait for Print Area) state is the state in which the system waits for the input from the printer which indicates that it is ready to start the actual printing of data.

TPA: During the TPA (Test Print Area) state the system digitally filters the signal from the printer to ensure that contact bounce is not causing an erroneous indication that the print area has started.

IPO: Transfer to the IPO (Initialize Print Operation) state occurs after the positioning of the print head over the print area has been verified. During this state the system initializes itself to start printing a line of text.

ICOL: The ICOL (Inter Column) state is used to time the period between the activation of the hammers. During this state the space between the dots of the characters is generated.

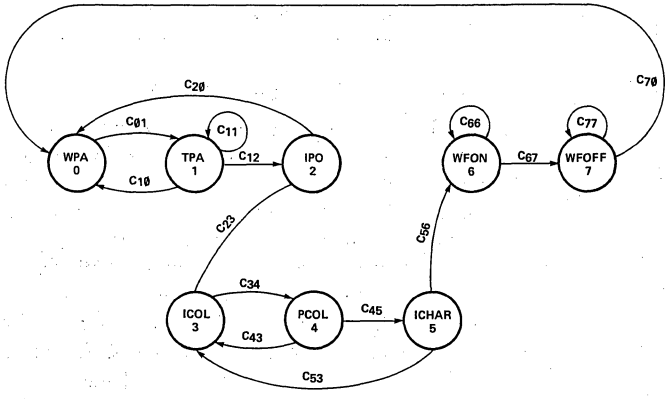
PCOL: During the PCOL (Print Column) state the hammers are energized if the particular character being printed requires a dot in the corresponding position.

ICHAR: The ICHAR (Inter Character) state is active between characters on a given line.

WFON: During the WFON (Wait for Feed On) state the system waits for the assertion of the feed pulse from the printer. This signal indicates that the process of feeding paper is occurring.

WFOFF: The system remains in the WFOFF (Wait for Feed Off) until the feed pulse goes inactive. This indicates that the required paper feed operation has been completed.

The state diagram, in addition to defining the allowable states, also defines how state to state transitions can be made. The general structure of this diagram shows that PSR is initiated by the occurrence of the timer overflow interrupt. When the interrupt occurs the contents of the HAMDAT (HAMmer DATa) register are immediately transferred to PORT1 which causes the hammer solenoids to be activated. Each of the eight possible states sets data into the register which should be output at the next timer overflow occurrence and starts the timer operating in a mode which will result in the main program (BMGR) being interrupted at the proper time. The following paragraphs describe the operation of each of the states



WPA ≡ WAIT FOR PRINT AREA
TPA ≡ TEST PRINT AREA
IPO ≡ INITIALIZE PRINT OPERATION
ICOL ≡ INTER COLUMN DELAY
PCOL ≡ PRINT COLUMN
ICHAR ≡ INTER CHARACTER DELAY
WFON ≡ WAIT FOR PAPER FEED ON
WFOFF ≡ WAIT FOR PAPER FEED OFF

C_{01} ≡ 1
C_{10} ≡ NOT (PRINT_AREA)
C_{11} ≡ (PRINT_AREA)·(FILTER()0)
C_{12} ≡ (PRINT_AREA)·(FILTER=0)
C_{20} ≡ (OBUF=EMPTY_FLAG)
C_{23} ≡ (OBUF()EMPTY_FLAG)
C_{34} ≡ 1
C_{43} ≡ (COL_COUNT()0)
C_{45} ≡ (COL_COUNT=0)
C_{53} ≡ (OBUF()EMPTY_FLAG)
C_{56} ≡ (OBUF=EMPTY_FLAG)
C_{66} ≡ (PAPERFEED=OFF)
C_{67} ≡ (PAPERFEED=ON)
C_{70} ≡ (PAPERFEED=ON)
C_{77} ≡ (PAPERFEED=OFF)

Figure 10. Print Control State Transition Diagram

in detail. The flowcharts of the routines can be found in Figure 11.

The WPA, CPA, and IPO states are all associated with the detection of the valid start of the print area. The WPA state sets the timer in the event count mode so that the edge of the print area signal can be detected, the CPA state digitally filters this input once it has been detected to ensure that noise has not caused a false input, and finally, the IPO state initializes the system to start the actual printing of data. The flowchart shows that the WPA state accomplishes the following actions:

1. Turns off the paper feed motor
2. Sets the filter count (for the CPA state)
3. Sets HAMDAT to zero
4. Sets STATE to one.

The timer is set to event count with an initial value of 0FFH. This will cause a timer overflow interrupt the next time a negative transition occurs on the TEST1 input. Since this input is tied to the signal from the PRINT AREA switch, this interrupt should occur when the start of the print area is reached. The WPA state sets the STATE register to cause the TPA state to be entered when this interrupt occurs. Each time the TPA (Test Print Area) state is activated the software checks to ensure that the print area switch is in the proper state; if it is not, then all the actions of state zero are repeated (except turning off the motor), since a false start of print area has occurred. If the test reveals that the print area switch is in the proper state, then the filter count is reduced by one and the timer is started with an initial value of 0FFH, the minimum attainable timer increment. The STATE register is set to repeat the TPA state unless the filter count has reached zero; when this occurs the IPO state is selected. The IPO state, which is responsible for the initialization of the actual print operation, first tests the output buffer register to determine if there is any data for it to print. If this test is unsuccessful the printer main drive motor is turned off, the TPA state is reinvoked and the timer is started in the event count mode so that it can detect the next start of print area. At first glance this seems somewhat fruitless since the event required cannot happen if the motor is not turning. By referring back to Figure 9, however, it can be seen that BMGR turns on the motor whenever it has a data character from the master computer. The reception of a character will always allow the PSR to find the next print area. If, when the IPO state makes its test, there is data in the output buffer then the data is moved to the print buffer and the output buffer is set to the empty value. After this is accomplished, a counter is set to the number of columns to be printed per character (seven in this case – see comment by CGEN label in program listing), the STATE register is set to the ICOL state and the timer is set to time the intercolumn time. (The intercolumn time is the time that elapses between each possible column of the character.) Before exiting from this state the first column of data for the hammbers is generated by the COLUMN routine and placed in the HAMDAT register.

The three states already discussed set the printer up so that it is ready to print. The next three states are repeated sequentially until the entire line of data has been printed. The ICOL state is probably the simplest of the states. When it is invoked the hammers have just been fired by the entry into the PSR. All that the ICOL state does is to set the timer to time the proper duration of the hammer strikes, clear the HAMDAT register, and set the STATE register to the PCOL state. The PCOL state, only slightly more complicated than the ICOL state, first decrements the column count. If the end of a character is detected (count equal zero), the HAMDAT register is cleared and the STATE register is set to invoke the ICHAR state. If the end of a character is not detected then the COLUMN routine is again used to determine the next data to be sent to the hammers and the ICOL state is reinvoked. When the ICOL state is active two things can happen, depending on whether there is more data to print. If there is data in the output buffer then a series of actions similar to those of the IPO state occur to reinitialize the printing of a character; if there is no more data in the line then the paper feed motor is turned on, HAMDAT is cleared, and the STATE register is set to the WFON state. The timer is set for approximately one millisecond so that the state of the paper feed switch can be sampled periodically by the WFON and WFOFF states.

The WFON and WFOFF states continue to set the timer to the one millisecond sample rate, the WFON state reinvokes itself until the paper feed switch input is detected and then it invokes the WFOFF state. The WFOFF state reinvokes itself until the paper feed switch is detected in the off state and then invokes the WPA state. The sole purpose of the WFON and WFOFF states is to ensure that an off to on to off transition occurs on

the paper feed switch. When this criterion is satisfied the WPA state is invoked which first turns off the paper feed motor and then proceeds to print the next line of data.

CONCLUSION

The UPI-41 has been shown to be easily capable of controlling the LRC matrix printer with no external logic other than drivers and receivers. The program listings which implement the algorithms discussed are shown in Appendix A. It should be noted that no attempt has been made to minimize the amount of code in the program; the emphasis was on clarity of operation and ease of implementation. A careful programmer should be able to significantly reduce the amount of code space needed, especially in the printer service routine which duplicates much code in each STATE. Even with this relatively loose coding the printer control function, including the complete character tables, easily fit within the memory available in the UPI-41. The extra room in memory could be used to implement such extra features as tabulation, printing prestored messages, or even limited graphic capabilities. The power and flexibility of the UPI-41 make such features easy to implement.

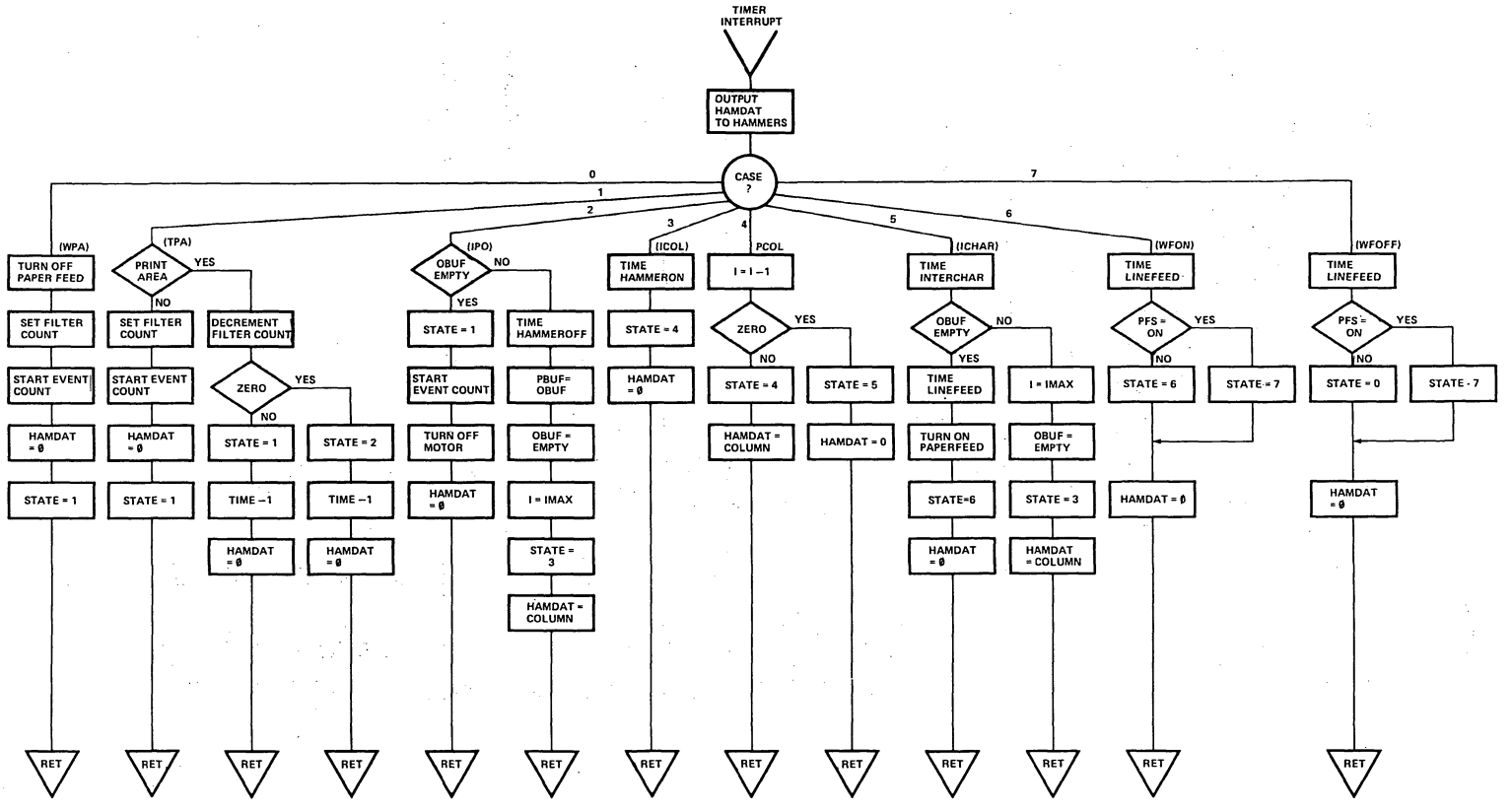


Figure 11. PSR Flowchart

```
ISIS-II 8048 ASSEMBLER, V1.1
                LRC PRINTER CONTROLLER 7/14/7

 LOC  OBJ          SEQ        SOURCE STATEMENT

                     1
                     2
                     3 ;*********************************************************************
                     4 ;
                     5 ;        UPI-41 LRC PRINTER CONTROLLER
                     6 ;
                     7 ;        THIS PROGRAM IMPLEMENTS THE CONTROL OF THE
                     8 ;        LRC PRINTER WITH THE UPI-41. DATA IS INPUT TO THE
                     9 ;        UPI-41 AS SIX BIT ASCII. COMMANDS ARE PROVIDED
                    10 ;        TO ENABLE OR DISABLE THE GENERATION OF AN
                    11 ;        INTERRUPT WHEN THE UNIT IS READY
                    12 ;        FOR ANOTHER DATA CHARACTER. THE INTERRUPT IS ENABLED
                    13 ;        BY OUTPUTING 03H TO THE CONTROL CHANNEL AND DISABLED
                    14 ;        BY OUTPUTING 02H. WHEN ENABLED THE INTERRUPT
                    15 ;        IS IMPLEMENTED AS A POSITIVE GOING EDGE ON P25.
                    16 ;
                    17 ; NOTE: A PL/M LIKE LANGUAGE WAS USED TO COMMENT
                    18 ;        THIS PROGRAM. NO COMPILER EXISTS FOR THE UPI-41.
                    19 ;        THE COMMENTS WERE 'HAND COMPILED' INTO UPI-41
                    20 ;        ASSEMBLY LANGUAGE.
                    21 ;
                    22 ;*********************************************************************
                    23
                    24
                    25
                    26 ;*************************************************************************
                    27 ;
                    28 ;        REGISTER ASSIGNMENTS
                    29 ;
                    30 ;*************************************************************************
                    31
                    32
 0007               33 HAMDAT  EQU     R7
 0006               34 STATE   EQU     R6
 0005               35 ICNT    EQU     R5
 0004               36 PBUF    EQU     R4
 0003               37 OBUF    EQU     R3
 0002               38 TESTR   EQU     R2
 0001               39 OUTPNT  EQU     R1
 0000               40 INPNT   EQU     R0
                    41
                    42
                    43
                    44 ;**********************************************************************
                    45 ;
                    46 ;        TIMER EQUATES
                    47 ;
                    48 ;**********************************************************************
                    49
 00A0               50 TICK    EQU     160
 FFFE               51 THON    EQU     -320/TICK
 FFFD               52 THOFF   EQU     -480/TICK
 FFF8               53 TINTER  EQU     -1280/TICK
 FFFA               54 TLFEED  EQU     -1000/TICK
 0004               55 FILTV   EQU     640/TICK
                    56
                    57
                    58 ;**********************************************************************
                    59 ;
                    60 ;        PROGRAM MASKS
                    61 ;
                    62 ;**********************************************************************
                    63
 00FF               64 EMTFLG  EQU     0FFH
 0007               65 IMAX    EQU     07H
 007F               66 PFEED   EQU     7FH
 00BF               67 MOTON   EQU     0BFH
 0001               68 INTENA  EQU     01H
 0002               69 FMODE   EQU     02H
 000A               70 EOL     EQU     0AH
 0021               71 EXCLAIM EQU     021H
 0020               72 SPACE   EQU     20H
 0020               73 EXREQ   EQU     20H
 0018               74 QPTMIN  EQU     18H
 0018               75 BMIN    EQU     18H
 003F               76 BMAX    EQU     3FH
                    77
                    78
                    79 $        EJECT
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                   80
                   81 ;*********************************************************************
                   82 ;
                   83 ;        START OF PROGRAM
                   84 ;
                   85 ;*********************************************************************
                   86
                   87                                  ; INITIALIZE;
                   88                                  ; INITIALIZE AND GO TO
                   89                                  ; BMGR
0000               90          ORG     Ø0H
0000 1416          91 RESET:   CALL    CASEØ
0002 3479          92          CALL    INIT
0004 25            93          EN      TCNTI
0005 2400          94          JMP     BMGR            ; CODE MUST END AT LOC 6
                   95
                   96
                   97 ;*********************************************************************
                   98 ;
                   99 ;        START OF INTERRUPT DRIVEN STATE MACHINE
                  100 ;
                  101 ;*********************************************************************
                  102
                  103                              ; DO;
                  104                              ;       HAMMERS=HAMMER$DAT;
                  105                              ;       DO CASE STATE;
0007 2F           106 TISR:   XCH     A,HAMDAT
0008 37           107         CPL     A
0009 39           108         OUTL    P1,A
000A FE           109         MOV     A,STATE
000B 030E         110         ADD     A,#CBASE
000D B3           111         JMPP    @A
000E 16           112 CBASE:  DB      CASEØ
000F 24           113         DB      CASE1
0010 4Ø           114         DB      CASE2
0011 61           115         DB      CASE3
0012 6B           116         DB      CASE4
0013 7D           117         DB      CASE5
0014 9E           118         DB      CASE6
0015 AE           119         DB      CASE7
                  120
                  121
                  122
                  123                              ;
                  124                              ;
                  125                              ;       DO;  /*CASE Ø, FEEDING LINE */
                  126                              ;          PAPER$FEED=OFF;
                  127                              ;          STATE=1;
                  128                              ;          ICNT=FILTV
                  129                              ;          WAIT(PRINT$AREA);
                  130                              ;          HAMMER$DATA=Ø;
                  131                              ;       END; /* END OF CASE Ø */
0016 8A8Ø         132 CASEØ:  ORL     P2,#(NOT PFEED)
0018 BE01         133         MOV     STATE,#1
001A BD04         134         MOV     ICNT,#FILTV
001C 23FF         135         MOV     A,#-1
001E 62           136         MOV     T,A
001F 45           137         STRT    CNT
0020 2300         138         MOV     A,#Ø
0022 2F           139         XCH     A,HAMDAT
0023 93           140         RETR
                  141                              ;       DO; /* CASE1,TESTING FOR PRINT AREA */
                  142                              ;          IF TØ=1 THEN
                  143                              ;          DO;
                  144                              ;             STATE=1;
                  145                              ;             ICNT=FILTV;
                  146                              ;             WAIT(PRINTAREA);
                  147                              ;             HAMMER$DATA=Ø;
                  148                              ;          END;
                  149
0024 4632         150 CASE1:  JNT1    C1ELS
0026 BE01         151         MOV     STATE,#1
0028 BD04         152         MOV     ICNT,#FILTV
002A 23FF         153         MOV     A,#-1
002C 62           154         MOV     T,A
002D 55           155         STRT    T
002E 2300         156         MOV     A,#Ø
0030 043E         157         JMP     C1END
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT
                   158                          ;          ELSE DO;
                   159                          ;             ICNT=ICNT-1;
                   160                          ;             IF ICNT=0 THEN STATE=2 ELSE STATE=1;
                   161                          ;             TIME(-1);
                   162                          ;             HAMMER$DATA=0;
                   163                          ;          END;
0032 BE02          164 C1ELS:  MOV     STATE,#2
0034 ED38          165         DJNZ    ICNT,C1LA
0036 BE01          166         MOV     STATE,#1
0038 23FF          167 C1LA:   MOV     A,#-1
003A 62            168         MOV     T,A
003B 55            169         STRT    T
003C 2300          170         MOV     A,#0
                   171                          ;       END; /*END OF CASE1 */
003E 2F            172 C1END:  XCH     A,HAMDAT
003F 93            173         RETR
                   174
                   175
                   176                          ;       DO;  /*CASE 2, INITIALIZE PRINT OPERATION */
                   177                          ;          IF OBUF<>EMPTY$FLAG THEN
                   178                          ;          DO;
                   179                          ;             TIME(HAMMER$OFF);
                   180                          ;             PBUF=OBUF;
                   181                          ;             OBUF=EMPTY$FLAG;
                   182                          ;             I=IMAX;
                   183                          ;             STATE=3;
                   184                          ;             HAMMER$DATA=COLUMN(PBUF,I);
                   185                          ;          END;
0040 FB            186 CASE2:  MOV     A,OBUF
0041 D3FF          187         XRL     A,#EMTFLG
0043 C655          188         JZ      C2ELS
0045 23FD          189         MOV     A,#THOFF
0047 62            190         MOV     T,A
0048 55            191         STRT    T
0049 FB            192         MOV     A,OBUF
004A AC            193         MOV     PBUF,A
004B BBFF          194         MOV     OBUF,#EMTFLG
004D BD07          195         MOV     ICNT,#IMAX
004F BE03          196         MOV     STATE,#3
0051 54E0          197         CALL    COLUMN
0053 045F          198         JMP     C2END
                   199                          ;          ELSE DO;
                   200                          ;             STATE=1;
                   201                          ;             WAIT(PRINT$AREA);
                   202                          ;             MOTOR=OFF;
                   203                          ;             HAMMER$DATA=0;
                   204                          ;          END;
0055 BE01          205 C2ELS:  MOV     STATE,#1
0057 23FF          206         MOV     A,#-1
0059 62            207         MOV     T,A
005A 45            208         STRT    CNT
005B 8A40          209         ORL     P2,#NOT MOTON
005D 2300          210         MOV     A,#0
                   211                          ;       END;  /*END OF CASE 2 */
005F 2F            212 C2END:  XCH     A,HAMDAT
0060 93            213         RETR
                   214                          ;
                   215                          ;       DO;  /*CASE 3, HAMMER OFF CYCLE */
                   216                          ;          TIME(HAMMER$ON);
                   217                          ;          HAMMER$DATA=0;
                   218                          ;          STATE=4;
                   219                          ;       END; /*END OF CASE 3 */
0061 23FE          220 CASE3:  MOV     A,#THON
0063 62            221         MOV     T,A
0064 55            222         STRT    T
0065 2300          223         MOV     A,#0
0067 BE04          224         MOV     STATE,#4
0069 2F            225         XCH     A,HAMDAT
006A 93            226         RETR
                   227 $       EJECT
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                  228                          ;        DO; /*CASE 4,PRINTING COL I OF CHAR */
                  229                          ;           TIME(HAMMER$OFF);
006B 23FD         230 CASE4:   MOV     A,#THOFF
006D 62           231          MOV     T,A
006E 55           232          STRT    T
                  233                          ;           I=I-1;
                  234                          ;           IF I=0 THEN
                  235                          ;           DO;
                  236                          ;              STATE=5;
                  237                          ;              HAMMER$DATA=0;
                  238                          ;           END
006F ED77         239          DJNZ    ICNT,C4ELS
0071 BE05         240          MOV     STATE,#5
0073 2300         241          MOV     A,#0
0075 047B         242          JMP     C4END
                  243                          ;           ELSE DO;
                  244                          ;              STATE=3;
                  245                          ;              HAMMER$DATA=COLUMN(PBUF,I);
                  246                          ;           END;
0077 BE03         247 C4ELS:   MOV     STATE,#3
0079 54E0         248          CALL    COLUMN
                  249                          ;        END; /* END OF CASE 4 */
007B 2F           250 C4END:   XCH     A,HAMDAT
007C 93           251          RETR
                  252                          ;        DO;  /*CASE 5, INTERCHARACTER SPACE */
                  253                          ;           TIME(INTER$CHAR);
007D 23F8         254 CASE5:   MOV     A,#TINTER
007F 62           255          MOV     T,A
0080 55           256          STRT    T
                  257                          ;           IF OBUF<>EMPTY$FLAG THEN
                  258                          ;           DO;
                  259                          ;              PBUF=OBUF;
                  260                          ;              OBUF=EMPTY$FLAG;
                  261                          ;              I=IMAX;
                  262                          ;              STATE=3;
                  263                          ;              HAMMER$DATA=COLUMN(PBUF,I);
                  264                          ;           END;
0081 FB           265          MOV     A,OBUF
0082 D3FF         266          XRL     A,#EMTFLG
0084 C692         267          JZ      C5ELS
0086 FB           268          MOV     A,OBUF
0087 AC           269          MOV     PBUF,A
0088 BBFF         270          MOV     OBUF,#EMTFLG
008A BD07         271          MOV     ICNT,#IMAX
008C BE03         272          MOV     STATE,#3
008E 54E0         273          CALL    COLUMN
0090 049C         274          JMP     C5END
                  275                          ;           ELSE DO;
                  276                          ;              TIME(LINE$FEED);
                  277                          ;              PAPER$FEED=ON;
                  278                          ;              STATE=6;
                  279                          ;              HAMMER$DATA=0;
                  280                          ;           END;
0092 23FA         281 C5ELS:   MOV     A,#TLFEED
0094 62           282          MOV     T,A
0095 55           283          STRT    T
0096 9A7F         284          ANL     P2,#PFEED
0098 BE06         285          MOV     STATE,#6
009A 2300         286          MOV     A,#0
                  287                          ;        END; /* END OF CASE 5*/
009C 2F           288 C5END:   XCH     A,HAMDAT
009D 93           289          RETR
                  290
                  291 $        EJECT
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                  292                          ;        DO;  /*CASE 6,WAITING FOR FEED ON */
                  293                          ;           TIME(LINE$FEED);
                  294                          ;           IF PFS=1 THEN
                  295                          ;           DO;
                  296                          ;              STATE=7;
                  297                          ;           END;
009E 23FA         298 CASE6:  MOV     A,#TLFEED
00A0 62           299         MOV     T,A
00A1 55           300         STRT    T
00A2 26A8         301         JNT0    C6ELS
00A4 BE07         302         MOV     STATE,#7
00A6 04AA         303         JMP     C6END
                  304                          ;           ELSE DO;
                  305                          ;              STATE=6;
                  306                          ;           END;
00A8 BE06         307 C6ELS:  MOV     STATE,#6
                  308                          ;           HAMMER$DATA=0;
                  309                          ;        END;  /*END OF CASE 6 */
00AA 2300         310 C6END:  MOV     A,#0
00AC 2F           311         XCH     A,HAMDAT
00AD 93           312         RETR
                  313
                  314                          ;        DO; /*CASE 7, WAITING FOR FEED OFF */
                  315                          ;           TIME(LINE$FEED);
                  316                          ;           IF PFS=0 THEN
                  317                          ;           DO;
                  318                          ;              STATE=0;
                  319                          ;           END;
00AE 23FA         320 CASE7:  MOV     A,#TLFEED
00B0 62           321         MOV     T,A
00B1 55           322         STRT    T
00B2 36B8         323         JT0     C7ELS
00B4 BE00         324         MOV     STATE,#0
00B6 04BA         325         JMP     C7END
                  326                          ;           ELSE DO;
                  327                          ;              STATE=7;
                  328                          ;           END;
                  329                          ;           HAMMER$DATA=0;
                  330                          ;        END;  /*END OF CASE 7 */
00B8 BE07         331 C7ELS:  MOV     STATE,#7
00BA 2300         332 C7END:  MOV     A,#0
00BC 2F           333         XCH     A,HAMDAT
00BD 93           334         RETR
                  335
                  336                          ;     END; /* END OF CASE BLOCK */
                  337
                  338
                  339
                  340
                  341 ;*********************************************************************
                  342 ;
                  343 ; BMGR
                  344 ;
                  345 ; THIS SEGMENT CONTROLS THE HANDSHAKING BETWEEN THE
                  346 ; CONTROLLER AND THE MASTER PROCESSOR.
                  347 ;
                  348 ;*********************************************************************
                  349
                  350
                  351                          ; /BMGR-BUFFER MANAGER*/
                  352
                  353                          ; DO;
                  354                          ;    IF IBF=FULL THEN
                  355                          ;    DO;
                  356                          ;       IF TYPE=DATA THEN
                  357                          ;       DO;
                  358                          ;         MOTOR=ON;
                  359                          ;         TEMP=INCQPT(INPNT);
                  360
                  361
0100              362         ORG     100H
                  363
0100 D647         364 BMGR:   JNIBF   BBPRT
0102 7636         365         JF1     BBCMD
0104 9ABF         366         ANL     P2,#MOTON
0106 F8           367         MOV     A,INPNT
0107 3470         368         CALL    INCQPT
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT
                   369                          ;        IF TEMP<>OUT$POINT THEN
                   370                          ;        DO;
                   371                          ;          IN$POINT=TEMP;
                   372                          ;          IF FILL$MODE=ON THEN
                   373                          ;          DO;
                   374                          ;             TEMP=SPACE;
                   375                          ;          ELSE DO;
                   376                          ;             TEMP=INPUT$BUFFER;
                   377                          ;             INTERRUPT=OFF;
                   378                          ;          END;
                   379                          ;          IF TEMP=EOL THEN
                   380                          ;          DO;
                   381                          ;             FILL$MODE=ON;
                   382                          ;             TEMP=SPACE;
                   383                          ;          END;
                   384                          ;          IF TEMP=CONTROL$CODE THEN TEMP='!';
                   385                          ;          BUFFER(IN$POINT)=TEMP AND 03FH;
                   386                          ;          IF IN$POINT=BUFFER$MAX THEN
                   387                          ;          DO;
                   388                          ;             FILL$MODE=OFF;
                   389                          ;             OUT$POINT=BUFFER$MIN;
                   390                          ;          END;
                   391                          ;        END;
                   392
                   393
0109 D9            394 BBL1:    XRL     A,OUTPNT
010A C647          395          JZ      BBPRT
010C D9            396          XRL     A,OUTPNT
010D A8            397          MOV     INPNT,A
010E 0A            398          IN      A,P2
010F 3216          399          JB1     FILL
0111 22            400          IN      A,DBB
0112 9ADF          401          ANL     P2,#NOT(EXREQ)
0114 2418          402          JMP     BBL1A
0116 2320          403 FILL:    MOV     A,#SPACE
0118 D30A          404 BBL1A:   XRL     A,#EOL
011A 9620          405          JNZ     BBL1B
011C 8A02          406          ORL     P2,#FMODE
011E 2428          407          JMP     BBL1C
0120 D30A          408 BBL1B:   XRL     A,#EOL
0122 D228          409          JB6     BBL1C
0124 B228          410          JB5     BBL1C
0126 2321          411          MOV     A,#EXCLAIM
0128 533F          412 BBL1C:   ANL     A,#03FH
012A A0            413          MOV     @INPNT,A
012B F8            414          MOV     A,INPNT
012C D33F          415          XRL     A,#BMAX
012E 9647          416          JNZ     BBPRT
0130 9AFD          417          ANL     P2,#NOT FMODE
0132 B918          418          MOV     OUTPNT,#BMIN
0134 2447          419          JMP     BBPRT
                   420                          ;        ELSE DO; /*TYPE IS COMMAND*/
                   421                          ;          INTERRUPT=OFF;
                   422                          ;          IF (PORT0 AND 3)=2 THEN INTENA=OFF;
                   423                          ;          IF(PORT0 AND 3)=3 THEN INTENA=ON;
                   424                          ;        END;
                   425
0136 22            426 BBCMD:   IN      A,DBB
0137 9ADF          427          ANL     P2,#NOT(EXREQ)
0139 5303          428          ANL     A,#3
013B 323F          429          JB1     BBL2
013D 2447          430          JMP     BBPRT
013F 1245          431 BBL2:    JB0     BBL3
0141 9AFE          432          ANL     P2,#NOT INTENA
0143 2447          433          JMP     BBPRT
0145 8A01          434 BBL3:    ORL     P2,#INTENA
                   435 $        EJECT
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                  436                          ;      IF OBUF=EMPTY$FLAG AND (OUT$POINT<>BMIN OR STATE
                  437                          ;      DO;
                  438                          ;       IF OUT$POINT<=BUFFER$MAX THEN
                  439                          ;       DO;
                  440                          ;          OBUF=BIN(OUT$POINT);
                  441                          ;          OUT$POINT=OUT$POINT+1;
                  442                          ;        END;
                  443                          ;      END;
                  444
0147 FB           445 BBPRT:   MOV     A,OBUF
0148 D3FF         446          XRL     A,#EMTFLG
014A 965E         447          JNZ     BINT
014C F9           448          MOV     A,OUTPNT
014D D318         449          XRL     A,#QPTMIN
014F 9658         450          JNZ     BBPRTA
0151 FE           451          MOV     A,STATE
0152 03FD         452          ADD     A,#-3
0154 F258         453          JB7     BBPRTA
0156 245E         454          JMP     BINT
0158 F9           455 BBPRTA:  MOV     A,OUTPNT
0159 D25E         456          JB6     BINT
015B F1           457          MOV     A,@OUTPNT
015C AB           458          MOV     OBUF,A
015D 19           459          INC     OUTPNT
                  460                          ;      TEMP=INCQPT(INPNT);
                  461                          ;      IF TEMP<>OUT$POINT THEN
                  462                          ;      DO;
                  463                          ;        IF INTENA=ON AND FMODE=OFF THEN INTERRUPT=ON;
                  464                          ;      END;
                  465                          ;    END;
                  466                          ; END
                  467
015E F8           468 BINT:    MOV     A,INPNT
015F 3470         469          CALL    INCQPT
0161 D9           470          XRL     A,OUTPNT
0162 C600         471          JZ      BMGR
                  472 BINTA:
0164 0A           473          IN      A,P2
0165 37           474          CPL     A
0166 1200         475          JB0     BMGR
0168 326C         476          JB1     SETINT
016A 2400         477          JMP     BMGR
016C 8A20         478 SETINT:  ORL     P2,#EXREQ
016E 2400         479          JMP     BMGR
                  480
                  481
                  482
                  483
                  484
                  485
                  486
                  487                          ; PROCEDURE INCQPT(A,CARRY);
                  488                          ; DO;
                  489                          ;    A=A MOD BUFFER LENGTH+BUFFER MIN;
                  490                          ;    IF A=BUFFER_MIN THEN CARRY=1;
                  491                          ; END;
0170 0301         492 INCQPT:  ADD     A,#1
0172 D275         493          JB6     ADJUST
0174 83           494          RET
0175 2318         495 ADJUST:  MOV     A,#QPTMIN
0177 A7           496          CPL     C
0178 83           497          RET
                  498
                  499
                  500                          ; PROCEDURE INIT;
                  501                          ; DO;
                  502                          ;    OBUF=EMPTY$FLAG;
                  503                          ;    OUT$POINT=BUFFER$MAX+1;
                  504                          ;    IN$POINT=BUFFER$MAX;
                  505                          ;    MOTOR=OFF;
                  506                          ;    PAPER$FEED=OFF;
                  507                          ;    FILL$MODE=OFF;
                  508                          ;    INTERRUPT$MASK=OFF;
                  509                          ;    BUS$BUFFER=0;
                  510                          ; END;
0179 BBFF         511 INIT:    MOV     OBUF,#EMTFLG
017B B940         512          MOV     OUTPNT,#BMAX+1
017D B83F         513          MOV     INPNT,#BMAX
017F 23F0         514          MOV     A,#0F0H
0181 3A           515          OUTL    P2,A
0182 22           516          IN      A,DBB
0183 83           517          RET
                  518
                  519
                  520
                  521
                  522
                  523
                  524 $        EJECT
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                   525 ;*********************************************************************
                   526 ;
                   527 ; COLUMN IS CALLED WITH ICNT EQL TO THE CURRENT COLUMN NUMBER
                   528 ; AND PBUFF EQL TO THE CHARACTER TO BE CONVERTED.
                   529 ; COLUMN RETURNS THE APPROPRIATE COLUMN OF DATA FROM THE
                   530 ; CHARACTER GENERATER TABLE. COLUMN IS LOCATED IN PAGE 2
                   531 ; FOLLOWING THE FIRST HALF OF THE TABLE.
                   532 ;
                   533 ; *********************************************************************
                   534
                   535
02E0               536          ORG      2E0H
                   537                                ; PROCEDURE COLUMN(PRINT$BUFFER,ICNT);
                   538                                ; DO;
                   539                                ;    FLAG0=NOT PRINT$BUFFER[5];
                   540                                ;    PRINT$BUFFER[5]=0;
                   541                                ;    TEMP=7*(PBUF+1)-ICNT
                   542                                ;    IF FLAG0=OFF THEN
                   543                                ;    DO;
                   544                                ;       TEMP=MP3(TEMP);
                   545                                ;       PRINT$BUFFER[5]=1;
                   546                                ;    END;
                   547                                ;    ELSE DO;
                   548                                ;       TEMP=MP2(TEMP);
                   549                                ;    END;
                   550                                ; END;
02E0 FC            551 COLUMN: MOV      A,PBUF
02E1 85            552         CLR      F0
02E2 B2E5          553         JB5      NOSET
02E4 95            554         CPL      F0
02E5 531F          555 NOSET:  ANL      A,#01FH
02E7 AC            556         MOV      PBUF,A
02E8 E7            557         RL       A
02E9 E7            558         RL       A
02EA E7            559         RL       A
02EB 37            560         CPL      A
02EC 6C            561         ADD      A,PBUF
02ED 6D            562         ADD      A,ICNT
02EE 37            563         CPL      A
02EF 0307          564         ADD      A,#7
02F1 B6F9          565         JF0      PAG2
02F3 E3            566         MOVP3    A,@A
02F4 2C            567         XCH      A,PBUF
02F5 4320          568         ORL      A,#20H
02F7 2C            569         XCH      A,PBUF
02F8 83            570         RET
02F9 A3            571 PAG2:   MOVP     A,@A
02FA 83            572         RET
                   573
                   574
                   575
                   576
                   577 ;*********************************************************************
                   578 ;
                   579 ; CHARACTER GENERATER TABLES.
                   580 ; THE FIRST HALF OF THESE TABLES IS IN PAGE 2. FOLLOWING THIS HALF
                   581 ; IS THE COLUMN SUBROUTINE. THE SECOND HALF OF THE TABLE IS
                   582 ; IN PAGE 3. THE PLACEMENT OF THESE TABLES IS TO TAKE
                   583 ; ADVANTAGE OF THE MCS-41 MOVP AND MOVP3 INSTRUCTIONS.
                   584 ;
                   585 ; THE CHARACTERS ARE FORMED BY A SEVEN BY SEVEN MATRIX
                   586 ; OF DOTS. EACH DOT POSITION CORRESPONDS
                   587 ; TO ONE HALF THE NORMAL DOT SPACING OF THE LRC PRINTER.
                   588 ; TO PREVENT EXCEEDING THE "BANDWIDTH" OF THE SOLENOIDS
                   589 ; THE CHARACTERS ARE FORMED SO THAT THE SAME SOLENOID IS
                   590 ; NOT ENERGIZED TWICE IN SUCCESSION. CONSTRUCTING THE
                   591 ; TABLE IN THIS MANNER ALLOWS THE FORMATION OF A CHARACTER IN ONLY
                   592 ; 4 PRINT COLUMNS SINCE THREE OF THE DOTS WILL APPEAR BETWEEN
                   593 ; NORMAL COLUMN POSITIONS.
                   594 ;
                   595 ; THE COMMENT FIELD OF THE TABLE SHOWS THE BIT PATTERN OF THE
                   596 ; CHARACTERS. THE SPACING OF THE PRINTER CHARACTERS CAUSES
                   597 ; DISTORTION OF THE CHARACTERS BUT THE PATTERN IS STILL DISCERNABLE.
                   598 ;
                   599 ;
                   600 ;*********************************************************************
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                601
0200            602        ORG     200H
                603
                604
0200 0C         605        DB      0CH     ;   **          ;  [AT]
0201 22         606        DB      22H     ;  *   *
0202 41         607        DB      41H     ; *     *
0203 58         608        DB      58H     ;    ** *
0204 01         609        DB      01H     ; *
0205 48         610        DB      48H     ;    *  *
0206 00         611        DB      00H     ;
                612
0207 0F         613        DB      0FH     ; ****          ;  [A]
0208 10         614        DB      10H     ;     *
0209 24         615        DB      24H     ;   *  *
020A 40         616        DB      40H     ;       *
020B 24         617        DB      24H     ;   *  *
020C 10         618        DB      10H     ;     *
020D 0F         619        DB      0FH     ; ****
                620
                621
020E 7F         622        DB      7FH     ; *******       ;  [B]
020F 00         623        DB      00H     ;
0210 49         624        DB      49H     ; *  *  *
0211 00         625        DB      00H     ;
0212 08         626        DB      08H     ;    *
0213 55         627        DB      55H     ; * * * *
0214 22         628        DB      22H     ;  *   *
                629
0215 3E         630        DB      3EH     ;  *****        ;  [C]
0216 41         631        DB      41H     ; *     *
0217 00         632        DB      00H     ;
0218 41         633        DB      41H     ; *     *
0219 00         634        DB      00H     ;
021A 41         635        DB      41H     ; *     *
021B 22         636        DB      22H     ;  *   *
                637
021C 7F         638        DB      7FH     ; *******       ;  [D]
021D 00         639        DB      00H     ;
021E 41         640        DB      41H     ; *     *
021F 00         641        DB      00H     ;
0220 00         642        DB      00H     ;
0221 41         643        DB      41H     ; *     *
0222 3E         644        DB      3EH     ;  *****
                645
0223 7F         646        DB      7FH     ; *******       ;  [E]
0224 00         647        DB      00H     ;
0225 49         648        DB      49H     ; *  *  *
0226 00         649        DB      00H     ;
0227 49         650        DB      49H     ; *  *  *
0228 00         651        DB      00H     ;
0229 41         652        DB      41H     ; *     *
                653
022A 7F         654        DB      7FH     ; *******       ;  [F]
022B 00         655        DB      00H     ;
022C 48         656        DB      48H     ;    *  *
022D 00         657        DB      00H     ;
022E 48         658        DB      48H     ;    *  *
022F 00         659        DB      00H     ;
0230 40         660        DB      40H     ;       *
                661
0231 3E         662        DB      3EH     ;  *****        ;  [G]
0232 41         663        DB      41H     ; *     *
0233 00         664        DB      00H     ;
0234 41         665        DB      41H     ; *     *
0235 04         666        DB      04H     ;   *
0236 41         667        DB      41H     ; *     *
0237 26         668        DB      26H     ;  **  *
                669
                670 $      EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0238 7F         671        DB      7FH     ; *******       ;  [H]
0239 00         672        DB      00H     ;
023A 08         673        DB      08H     ;    *
023B 00         674        DB      00H     ;
023C 08         675        DB      08H     ;    *
023D 00         676        DB      00H     ;
023E 7F         677        DB      7FH     ; *******
                678
023F 00         679        DB      00H     ;               ;  [I]
0240 41         680        DB      41H     ; *     *
0241 00         681        DB      00H     ;
0242 7F         682        DB      7FH     ; *******
0243 00         683        DB      00H     ;
0244 41         684        DB      41H     ; *     *
0245 00         685        DB      00H     ;
                686
0246 02         687        DB      02H     ;  *            ;  [J]
0247 01         688        DB      01H     ; *
0248 00         689        DB      00H     ;
0249 01         690        DB      01H     ; *
024A 00         691        DB      00H     ;
024B 01         692        DB      01H     ; *
024C 7E         693        DB      7EH     ;  ******
                694
024D 7F         695        DB      7FH     ; *******       ;  [K]
024E 00         696        DB      00H     ;
024F 04         697        DB      04H     ;   *
0250 14         698        DB      14H     ;   * *
0251 22         699        DB      22H     ;  *   *
0252 41         700        DB      41H     ; *     *
0253 00         701        DB      00H     ;
                702
0254 7F         703        DB      7FH     ; *******       ;  [L]
0255 00         704        DB      00H     ;
0256 01         705        DB      01H     ; *
0257 00         706        DB      00H     ;
0258 01         707        DB      01H     ; *
0259 00         708        DB      00H     ;
025A 01         709        DB      01H     ; *
                710
025B 7F         711        DB      7FH     ; *******       ;  [M]
025C 40         712        DB      40H     ;       *
025D 20         713        DB      20H     ;      *
025E 18         714        DB      18H     ;    **
025F 20         715        DB      20H     ;      *
0260 40         716        DB      40H     ;       *
0261 3F         717        DB      3FH     ; ******
                718
                719
0262 7F         720        DB      7FH     ; *******       ;  [N]
0263 20         721        DB      20H     ;      *
0264 10         722        DB      10H     ;     *
0265 08         723        DB      08H     ;    *
0266 04         724        DB      04H     ;   *
0267 00         725        DB      00H     ;
0268 7F         726        DB      7FH     ; *******
                727
0269 3E         728        DB      3EH     ;  *****        ;  [O]
026A 41         729        DB      41H     ; *     *
026B 00         730        DB      00H     ;
026C 41         731        DB      41H     ; *     *
026D 00         732        DB      00H     ;
026E 41         733        DB      41H     ; *     *
026F 3E         734        DB      3EH     ;  *****
                735
0270 37         736        DB      37H     ; *** **        ;  [P]
0271 00         737        DB      00H     ;
0272 48         738        DB      48H     ;    *  *
0273 00         739        DB      00H     ;
0274 00         740        DB      00H     ;
0275 48         741        DB      48H     ;    *  *
0276 30         742        DB      30H     ;     **
                743
0277 3E         744        DB      3EH     ;  *****        ;  [Q]
0278 41         745        DB      41H     ; *     *
0279 00         746        DB      00H     ;
027A 40         747        DB      40H     ;       *
027B 05         748        DB      05H     ; * *
027C 42         749        DB      42H     ;  *    *
027D 3D         750        DB      3DH     ; * ****
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT
                  751
027E 7F           752          DB      7FH     ; *******       ; [R]
027F 00           753          DB      00H     ;
0280 48           754          DB      48H     ;    *  *
0281 00           755          DB      00H     ;
0282 04           756          DB      04H     ;   *
0283 4A           757          DB      4AH     ;  * *  *
0284 31           758          DB      31H     ; *   **
                  759
0285 32           760          DB      32H     ;  *  **        ; [S]
0286 49           761          DB      49H     ; *  *  *
0287 00           762          DB      00H     ;
0288 49           763          DB      49H     ; *  *  *
0289 00           764          DB      00H     ;
028A 49           765          DB      49H     ; *  *  *
028B 26           766          DB      26H     ;  **  *
                  767
                  768
028C 40           769          DB      40H     ;       *       ; [T]
028D 00           770          DB      00H     ;
028E 40           771          DB      40H     ;       *
028F 3F           772          DB      3FH     ; ******
0290 40           773          DB      40H     ;       *
0291 00           774          DB      00H     ;
0292 40           775          DB      40H     ;       *
                  776
0293 7C           777          DB      7CH     ;  *****        ; [U]
0294 02           778          DB      02H     ;  *
0295 01           779          DB      01H     ; *
0296 00           780          DB      00H     ;
0297 01           781          DB      01H     ; *
0298 02           782          DB      02H     ;  *
0299 7C           783          DB      7CH     ;  *****
                  784
029A 78           785          DB      78H     ;   ****        ; [V]
029B 04           786          DB      04H     ;   *
029C 02           787          DB      02H     ;  *
029D 01           788          DB      01H     ; *
029E 02           789          DB      02H     ;  *
029F 04           790          DB      04H     ;   *
02A0 78           791          DB      78H     ;   ****
                  792
02A1 7E           793          DB      7EH     ;  ******       ; [W]
02A2 01           794          DB      01H     ; *
02A3 02           795          DB      02H     ;  *
02A4 0C           796          DB      0CH     ;  **
02A5 02           797          DB      02H     ;  *
02A6 01           798          DB      01H     ; *
02A7 7E           799          DB      7EH     ;  ******
                  800
02A8 41           801          DB      41H     ; *     *       ; [X]
02A9 22           802          DB      22H     ;  *   *
02AA 14           803          DB      14H     ;   * *
02AB 08           804          DB      08H     ;    *
02AC 14           805          DB      14H     ;   * *
02AD 22           806          DB      22H     ;  *   *
02AE 41           807          DB      41H     ; *     *
                  808
02AF 40           809          DB      40H     ;       *       ; [Y]
02B0 20           810          DB      20H     ;      *
02B1 10           811          DB      10H     ;     *
02B2 0F           812          DB      0FH     ; ****
02B3 10           813          DB      10H     ;     *
02B4 20           814          DB      20H     ;      *
02B5 40           815          DB      40H     ;       *
                  816
                  817 $        EJECT
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

02B6 41            818          DB      41H     ; *     *       ;  [Z]
02B7 02            819          DB      02H     ;  *
02B8 45            820          DB      45H     ; * *   *
02B9 08            821          DB      08H     ;   *
02BA 51            822          DB      51H     ; *   * *
02BB 20            823          DB      20H     ;     *
02BC 41            824          DB      41H     ; *     *
                   825
02BD 7F            826          DB      7FH     ; *******       ;  [[]
02BE 00            827          DB      00H     ;
02BF 41            828          DB      41H     ; *     *
02C0 00            829          DB      00H     ;
02C1 41            830          DB      41H     ; *     *
02C2 00            831          DB      00H     ;
02C3 41            832          DB      41H     ; *     *
                   833
02C4 40            834          DB      40H     ;       *       ;  [\]
02C5 20            835          DB      20H     ;      *
02C6 10            836          DB      10H     ;     *
02C7 08            837          DB      08H     ;    *
02C8 04            838          DB      04H     ;   *
02C9 02            839          DB      02H     ;  *
02CA 01            840          DB      01H     ; *
                   841
02CB 41            842          DB      41H     ; *     *       ;  []]
02CC 00            843          DB      00H     ;
02CD 41            844          DB      41H     ; *     *
02CE 00            845          DB      00H     ;
02CF 41            846          DB      41H     ; *     *
02D0 00            847          DB      00H     ;
02D1 7F            848          DB      7FH     ; *******
                   849
02D2 00            850          DB      00H     ;               ;  [UA]
02D3 04            851          DB      04H     ;   *
02D4 08            852          DB      08H     ;    *
02D5 10            853          DB      10H     ;     *
02D6 08            854          DB      08H     ;    *
02D7 04            855          DB      04H     ;   *
02D8 00            856          DB      00H     ;
                   857
02D9 01            858          DB      01H     ; *             ;  [_]
02DA 00            859          DB      00H     ;
02DB 01            860          DB      01H     ; *
02DC 00            861          DB      00H     ;
02DD 01            862          DB      01H     ; *
02DE 00            863          DB      00H     ;
02DF 01            864          DB      01H     ; *
                   865
                   866
                   867
                   868
                   869 ;*********************************************************************
                   870 ;
                   871 ; START OF SECOND HALF OF CGEN TABLE
                   872 ;
                   873 ;*********************************************************************
                   874
0300               875          ORG     300H
                   876
                   877
0300 00            878          DB      00H     ;               ;  [ ]
0301 00            879          DB      00H     ;
0302 00            880          DB      00H     ;
0303 00            881          DB      00H     ;
0304 00            882          DB      00H     ;
0305 00            883          DB      00H     ;
0306 00            884          DB      00H     ;
                   885
0307 00            886          DB      00H     ;               ;  [!]
0308 00            887          DB      00H     ;
0309 00            888          DB      00H     ;
030A 7D            889          DB      7DH     ; * *****
030B 00            890          DB      00H     ;
030C 00            891          DB      00H     ;
030D 00            892          DB      00H     ;
                   893
                   894 $        EJECT
```

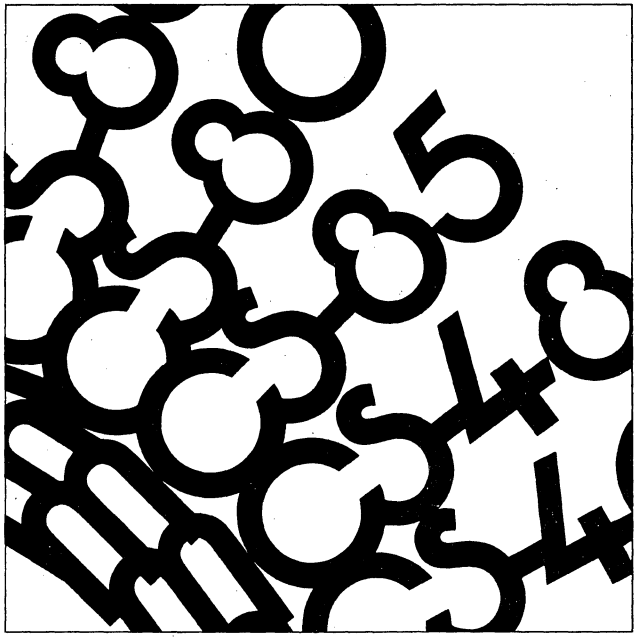
```
LOC  OBJ          SEQ        SOURCE STATEMENT

030E 00           895        DB     00H     ;                  ;  ["]
030F 20           896        DB     20H     ;      *
0310 40           897        DB     40H     ;       *
0311 00           898        DB     00H     ;
0312 20           899        DB     20H     ;      *
0313 40           900        DB     40H     ;       *
0314 00           901        DB     00H     ;
                  902
0315 14           903        DB     14H     ;   * *            ;  [#]
0316 00           904        DB     00H     ;
0317 7F           905        DB     7FH     ; *******
0318 00           906        DB     00H     ;
0319 7F           907        DB     7FH     ; *******
031A 00           908        DB     00H     ;
031B 14           909        DB     14H     ;   * *
                  910
031C 00           911        DB     00H     ;                  ;  [$]
031D 32           912        DB     32H     ;  *  **
031E 49           913        DB     49H     ; *  *  *
031F 36           914        DB     36H     ;  ** **
0320 49           915        DB     49H     ; *  *  *
0321 26           916        DB     26H     ;  **  *
0322 00           917        DB     00H     ;
                  918
0323 51           919        DB     51H     ; *   * *          ;  [%]
0324 02           920        DB     02H     ;  *
0325 54           921        DB     54H     ;   * * *
0326 08           922        DB     08H     ;    *
0327 15           923        DB     15H     ; * * *
0328 20           924        DB     20H     ;      *
0329 45           925        DB     45H     ; * *   *
                  926
032A 26           927        DB     26H     ;  **  *           ;  [&]
032B 49           928        DB     49H     ; *  *  *
032C 10           929        DB     10H     ;    *
032D 49           930        DB     49H     ; *  *  *
032E 26           931        DB     26H     ;  **  *
032F 01           932        DB     01H     ; *
0330 05           933        DB     05H     ; * *
                  934
0331 00           935        DB     00H     ;                  ;  ['']
0332 00           936        DB     00H     ;
0333 10           937        DB     10H     ;     *
0334 20           938        DB     20H     ;      *
0335 40           939        DB     40H     ;       *
0336 00           940        DB     00H     ;
0337 00           941        DB     00H     ;
                  942
                  943
0338 1C           944        DB     1CH     ;   ***            ;  [(]
0339 22           945        DB     22H     ;  *   *
033A 41           946        DB     41H     ; *     *
033B 00           947        DB     00H     ;
033C 00           948        DB     00H     ;
033D 00           949        DB     00H     ;
033E 00           950        DB     00H     ;
                  951
033F 00           952        DB     00H     ;                  ;  [)]
0340 00           953        DB     00H     ;
0341 00           954        DB     00H     ;
0342 00           955        DB     00H     ;
0343 41           956        DB     41H     ; *     *
0344 22           957        DB     22H     ;  *   *
0345 1C           958        DB     1CH     ;   ***
                  959
0346 49           960        DB     49H     ; *  *  *          ;  [*]
0347 22           961        DB     22H     ;  *   *
0348 1C           962        DB     1CH     ;   ***
0349 77           963        DB     77H     ; *** ***
034A 1C           964        DB     1CH     ;   ***
034B 22           965        DB     22H     ;  *   *
034C 49           966        DB     49H     ; *  *  *
                  967
034D 08           968        DB     08H     ;    *             ;  [+]
034E 08           969        DB     08H     ;    *
034F 08           970        DB     08H     ;    *
0350 3E           971        DB     3EH     ;  *****
0351 08           972        DB     08H     ;    *
0352 08           973        DB     08H     ;    *
0353 08           974        DB     08H     ;    *
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                   975
0354 00            976        DB      00H     ;                       ;  [,]
0355 00            977        DB      00H     ;
0356 00            978        DB      00H     ;
0357 01            979        DB      01H     ; *
0358 06            980        DB      06H     ;  **
0359 00            981        DB      00H     ;
035A 00            982        DB      00H     ;
                   983
035B 04            984        DB      04H     ;   *                   ;  [-]
035C 04            985        DB      04H     ;   *
035D 04            986        DB      04H     ;   *
035E 04            987        DB      04H     ;   *
035F 04            988        DB      04H     ;   *
0360 04            989        DB      04H     ;   *
0361 04            990        DB      04H     ;   *
                   991
                   992
0362 00            993        DB      00H     ;                       ;  [.]
0363 00            994        DB      00H     ;
0364 00            995        DB      00H     ;
0365 01            996        DB      01H     ; *
0366 00            997        DB      00H     ;
0367 00            998        DB      00H     ;
0368 00            999        DB      00H     ;
                  1000
0369 01           1001        DB      01H     ; *                     ;  [/]
036A 02           1002        DB      02H     ;  *
036B 04           1003        DB      04H     ;   *
036C 08           1004        DB      08H     ;    *
036D 10           1005        DB      10H     ;     *
036E 20           1006        DB      20H     ;      *
036F 40           1007        DB      40H     ;       *
                  1008
0370 1D           1009        DB      1DH     ; * ***                 ;  [0]
0371 22           1010        DB      22H     ;  *   *
0372 45           1011        DB      45H     ; * *   *
0373 08           1012        DB      08H     ;    *
0374 51           1013        DB      51H     ; *   * *
0375 22           1014        DB      22H     ;  *   *
0376 5C           1015        DB      5CH     ;   *** *
                  1016
0377 00           1017        DB      00H     ;                       ;  [1]
0378 21           1018        DB      21H     ; *    *
0379 40           1019        DB      40H     ;       *
037A 7F           1020        DB      7FH     ; *******
037B 00           1021        DB      00H     ;
037C 01           1022        DB      01H     ; *
037D 00           1023        DB      00H     ;
                  1024
037E 23           1025        DB      23H     ; **   *                ;  [2]
037F 44           1026        DB      44H     ;  *   *
0380 01           1027        DB      01H     ; *
0381 48           1028        DB      48H     ;    *  *
0382 01           1029        DB      01H     ; *
0383 48           1030        DB      48H     ;    *  *
0384 31           1031        DB      31H     ; *   **
                  1032
0385 42           1033        DB      42H     ;  *    *               ;  [3]
0386 01           1034        DB      01H     ; *
0387 50           1035        DB      50H     ;     * *
0388 01           1036        DB      01H     ; *
0389 50           1037        DB      50H     ;     * *
038A 29           1038        DB      29H     ; *  * *
038B 46           1039        DB      46H     ;  **   *
                  1040
                  1041 $       EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

038C 04         1042       DB      04H     ;   *           ;  [4]
038D 08         1043       DB      08H     ;    *
038E 14         1044       DB      14H     ;   * *
038F 20         1045       DB      20H     ;      *
0390 5F         1046       DB      5FH     ; ***** *
0391 00         1047       DB      00H     ;
0392 04         1048       DB      04H     ;   *
                1049
0393 72         1050       DB      72H     ;  *  ***       ;  [5]
0394 01         1051       DB      01H     ; *
0395 50         1052       DB      50H     ;    * *
0396 01         1053       DB      01H     ; *
0397 40         1054       DB      40H     ;      *
0398 11         1055       DB      11H     ; *   *
0399 4E         1056       DB      4EH     ;  ***  *
                1057
039A 17         1058       DB      17H     ; *** *         ;  [6]
039B 21         1059       DB      21H     ; *    *
039C 40         1060       DB      40H     ;      *
039D 09         1061       DB      09H     ; *  *
039E 40         1062       DB      40H     ;      *
039F 09         1063       DB      09H     ; *  *
03A0 46         1064       DB      46H     ;  **   *
                1065
03A1 40         1066       DB      40H     ;      *        ;  [7]
03A2 00         1067       DB      00H     ;
03A3 47         1068       DB      47H     ; ***   *
03A4 08         1069       DB      08H     ;    *
03A5 50         1070       DB      50H     ;    * *
03A6 20         1071       DB      20H     ;     *
03A7 40         1072       DB      40H     ;      *
                1073
03A8 36         1074       DB      36H     ;  ** **        ;  [8]
03A9 49         1075       DB      49H     ; *  *  *
03AA 00         1076       DB      00H     ;
03AB 49         1077       DB      49H     ; *  *  *
03AC 00         1078       DB      00H     ;
03AD 49         1079       DB      49H     ; *  *  *
03AE 36         1080       DB      36H     ;  ** **
                1081
03AF 30         1082       DB      30H     ;     **        ;  [9]
03B0 48         1083       DB      48H     ;   *  *
03B1 01         1084       DB      01H     ; *
03B2 48         1085       DB      48H     ;   *  *
03B3 01         1086       DB      01H     ; *
03B4 42         1087       DB      42H     ;  *    *
03B5 3C         1088       DB      3CH     ;   ****
                1089
                1090
03B6 00         1091       DB      00H     ;               ;  [:]
03B7 00         1092       DB      00H     ;
03B8 00         1093       DB      00H     ;
03B9 14         1094       DB      14H     ;   * *
03BA 00         1095       DB      00H     ;
03BB 00         1096       DB      00H     ;
03BC 00         1097       DB      00H     ;
                1098
03BD 00         1099       DB      00H     ;               ;  [;]
03BE 00         1100       DB      00H     ;
03BF 01         1101       DB      01H     ; *
03C0 02         1102       DB      02H     ;  *
03C1 14         1103       DB      14H     ;   * *
03C2 00         1104       DB      00H     ;
03C3 00         1105       DB      00H     ;
                1106
03C4 00         1107       DB      00H     ;               ;  [<]
03C5 08         1108       DB      08H     ;    *
03C6 14         1109       DB      14H     ;   * *
03C7 22         1110       DB      22H     ;  *   *
03C8 41         1111       DB      41H     ; *     *
03C9 00         1112       DB      00H     ;
03CA 00         1113       DB      00H     ;
                1114
03CB 00         1115       DB      00H     ;               ;  [=]
03CC 14         1116       DB      14H     ;   * *
03CD 00         1117       DB      00H     ;
03CE 14         1118       DB      14H     ;   * *
03CF 00         1119       DB      00H     ;
03D0 14         1120       DB      14H     ;   * *
03D1 00         1121       DB      00H     ;
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                  1122
03D2 00           1123          DB      00H     ;               ;  [>]
03D3 00           1124          DB      00H     ;
03D4 41           1125          DB      41H     ; *     *
03D5 22           1126          DB      22H     ;  *   *
03D6 14           1127          DB      14H     ;   * *
03D7 08           1128          DB      08H     ;    *
03D8 00           1129          DB      00H     ;
                  1130
03D9 00           1131          DB      00H     ;               ;  [?]
03DA 20           1132          DB      20H     ;      *
03DB 40           1133          DB      40H     ;       *
03DC 05           1134          DB      05H     ; * *
03DD 48           1135          DB      48H     ;    *  *
03DE 30           1136          DB      30H     ;     **
03DF 00           1137          DB      00H     ;
                  1138
                  1139          END
```

APPENDIX 1
ARTICLE REPRINTS



Slave microcomputer lightens main microprocessor load

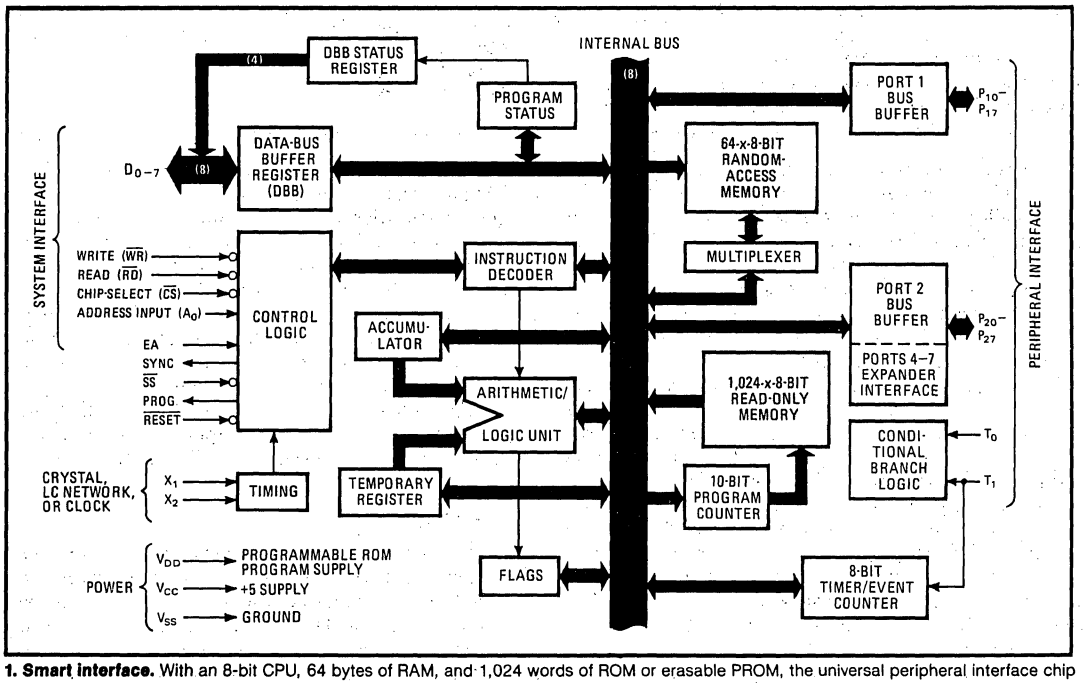
by Don Phillips and Allen Goodman, *Intel Corp., Santa Clara, Calif.*

☐ Peripheral devices for microprocessors are growing in number and complexity to the point where they are taxing the processor's time and memory. Nor do simple interface adapters that contain no intelligence of their own lighten the burden of managing such peripheral equipment as floppy disks, cathode-ray-tube displays, and keyboards. What can save the day for the central processing unit is a new class of peripheral controllers: intelligent microcomputer-based universal peripheral interface chips.

In essence, what the UPI microcomputer does is act as a slave processor to the main-system CPU. With a built-in processor and memory, it greatly eases the handling of real-time tasks such as controlling printers, encoding keyboards, and multiplexing displays. In fact, entire control algorithms can be programmed locally in the slave processor, instead of taxing the limited memory space and execution time of the main system. Moreover, the device substantially increases the overall efficiency of a system, since two processors—the central CPU and the slave UPI device—are working in parallel.

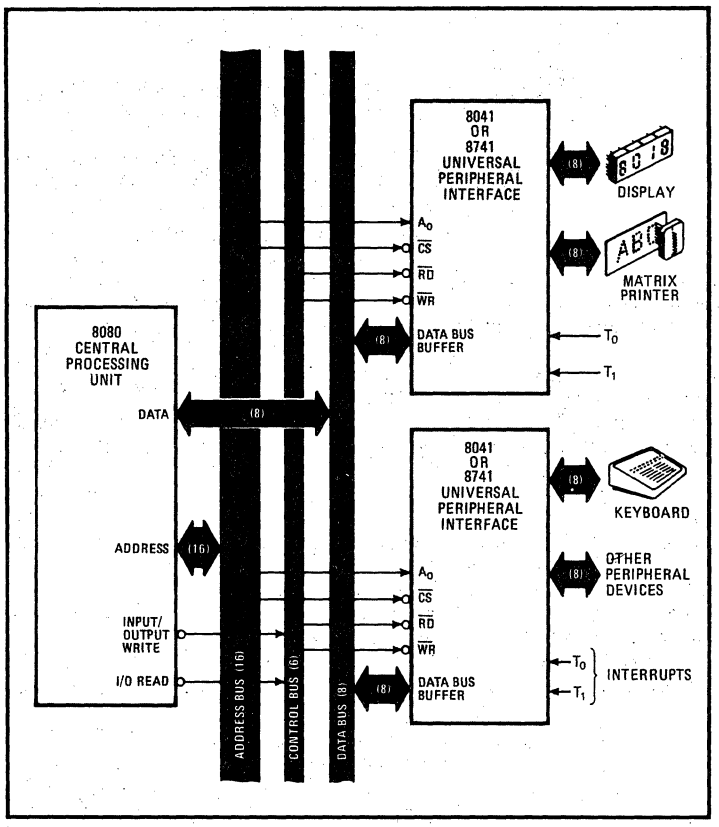
A peripheral controller

In operation, the UPI microcomputer acts as a peripheral controller rather than just an interface adapter. Its architecture, detailed in Fig. 1, is similar to the recently introduced 8048 one-chip microcomputer: it has an 8-bit CPU, 64 bytes of random-access memory, 1,024 bytes of read-only memory, a timer/counter, and 18 input/output lines. In fact, the device executes the same basic set of instructions as does the 8048, except for special tailoring of data-bus operations to better suit control applications. The difference is that the new peripheral-controlling microcomputer is designed to function as a



1. Smart interface. With an 8-bit CPU, 64 bytes of RAM, and 1,024 words of ROM or erasable PROM, the universal peripheral interface chip is an intelligent peripheral controller rather than a simple interface adapter. The architecture of the chip is similar to that of the 8048 microcomputer. It uses nearly the same instruction set, save for slight variations that improve data-bus operations.

Electronics/July 7, 1977



2. Slaves. The microcomputer-based universal peripheral interface chips—the 8741 with erasable PROM and the 8041 with mask-programmed ROM—are connected as slave processors to a main processor (here an 8080 CPU) to take over its I/O chores.

slave processor to the main-system processor.

The chip is the first microcomputer made specifically for a multiprocessor environment in which a master processor sends information to one or more slave processors that in turn control peripheral devices. To accommodate a variety of master processor types, including the 8080, the enhanced 8085, and other 8-bit processors, the chip has bus interface registers that work directly with the central processor's data bus.

Two peripheral controllers are available: the 8741 and the 8041, identical except in one respect. The 8741 has an ultraviolet-erasable, electrically programmable ROM plus the special capability of running through a program a single step at a time. It is designed for low-volume applications requiring program development, as in prototype testing and custom interfacing. The 8041 has a conventional mask-programmable ROM and features a low-power standby mode. It is intended to replace the 8741 once a system design has been set. The 8741/8041 connections for a master-slave arrangement are shown in the block diagram of Fig. 2.

The master processor and the peripheral controller communicate through an asynchronous data-bus buffer register on the UPI. Data and commands are received from the master processor through the DBB, and status and data information are returned through it to the master. The controller sends status information to the main processor from a 4-bit status register that uses four of the buffer register's eight lines.

The configuration of the DBB and status registers is shown in Fig. 3. The master processor controls data transfer to the UPI by four input lines: the address-input signal specifies whether a command or a data word is being sent; the chip-select line is an enable input that permits communication with the interface, and the read and write lines are used to stroke output and input data, respectively. The master processor uses these signals to direct the exchange of information through the DBB register, which serves as temporary storage for commands and data flowing between master and slave processors.

The four flags

The status register comprises four flags that direct the handshaking between the master and slave processors. The first is a general-purpose flag, which is set by programming in the 8041/8741 and used to prevent contention over the DBB register between master and slave processors. Another is the command/data flag that, when set, indicates that command information is being transferred. The input-buffer-full flag is set whenever the DBB register is loaded with a word from the main processor, and the output-buffer-full flag is set when the UPI loads its DBB register.

Protocol for the interface begins with the master processor writing an 8-bit character into the buffer register. This sets the IBF flag, signaling the peripheral controller with an internal interrupt. The UPI can then transfer the 8-bit data byte to its accumulator at any time under software control, which clears the IBF flag.

In transferring data in the other direction—from slave to master—the peripheral chip loads the DBB register while automatically setting the OBF flag. The master processor can then read the status register to determine that the OBF flag is set and can proceed to take in data from the buffer register, at the same time clearing the flag in preparation for the arrival of more data.

Transfer of data within the peripheral controller is asynchronous to external processor timing. The chip can thus effectively control peripheral devices while data transfers go on unhindered. Moreover, the DBB register isolates peripheral control tasks from the main processor. Task isolation is desirable in that it eases software development and debugging within a given system (by modularizing functions). In addition, it is certain to enhance data throughput, since two microprocessors are running concurrently.

Optimized for control

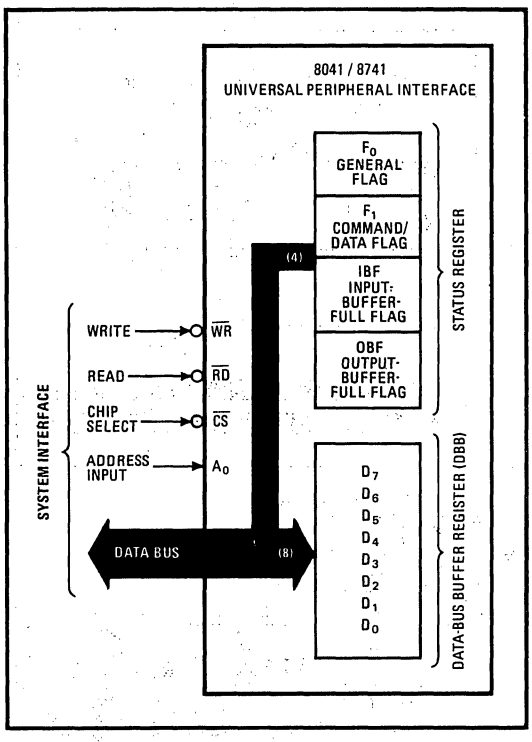
The CPU and instruction set of the 8041/8741 are designed to efficiently handle the single-bit operations required in most control applications, including I/O operations and data-bit manipulation. Two 8-bit-wide ports, compatible with transistor-transistor logic, are provided on the chip. (Sixteen additional lines may be had with the addition of an 8243 I/O expander chip, which takes up half the lines of I/O port 2.) Two inputs to the peripheral controller are provided that may be tested with conditional branch instructions in UPI software. Any port line can be set or cleared individually under software control, and any line can function as either input or output, irrespective of remaining lines.

The timer/event-counter included on the peripheral controller can be preset, read, started, or stopped under software control. In the timing mode, an internal oscillator can be set by a crystal or an LC network. In the event-counter mode, the T_1 input may be used to count switch closures or tachometer pulses, directing program flow accordingly. If the counter has been preset, a flag is available that indicates overflow, and it can signal the master processor.

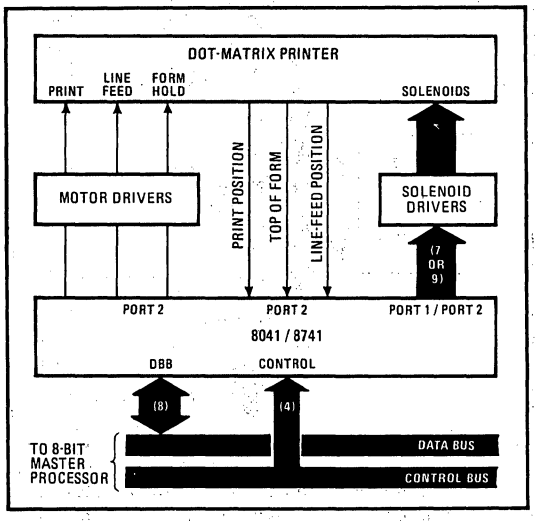
The 1,024 bytes of on-chip ROM are sufficient for most dedicated programming applications. Typically, keyboard encoding or printer control requires 500 to 700 8-bit bytes, and therefore ample program space is available for additional functions.

Of the 64 locations in the on-chip RAM, there are two 8-byte register banks, an eight-level program-counter stack, and 32 bytes of user RAM. The dual 8-byte register banks allow fast response to interrupts such as the IBF flag or time overflow. The stack also provides convenient handling of subroutine cells and storage of other data.

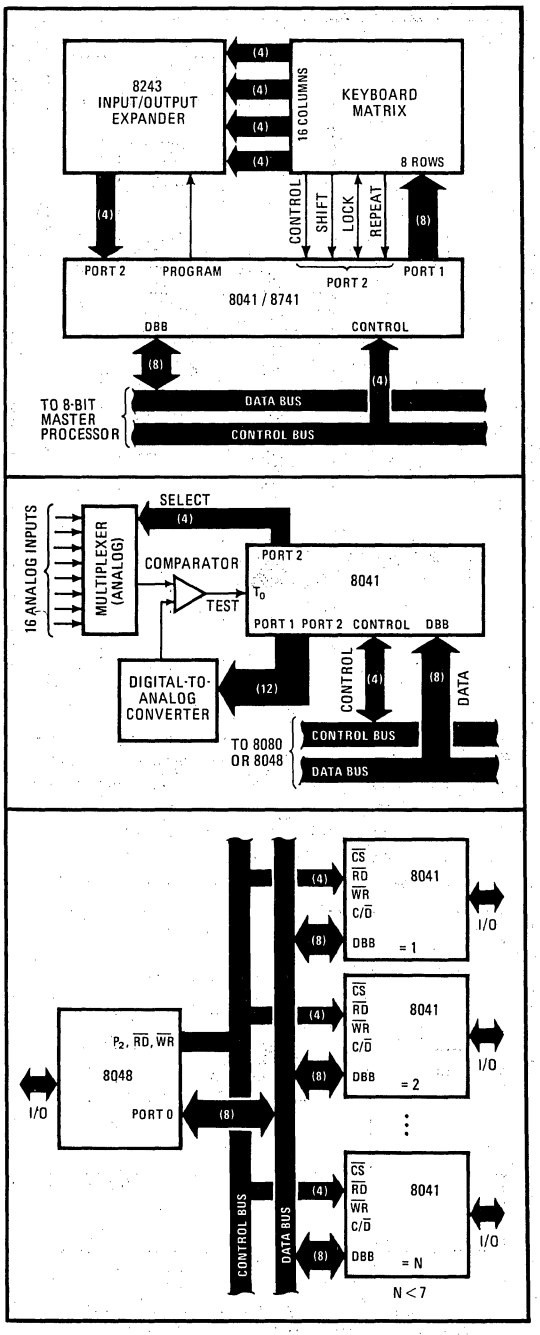
The thrust of the peripheral-controller chip is in its isolation of peripheral tasks from the main processor. Since its job is specifically for control, the main



3. Buffer to the bus. The data-bus buffer register (DBB) provides temporary storage for commands and data flowing between the UPI and a main-system processor. The status register puts four signals on the data bus that between them inform the main processor of the status of the DBB and also establish a handshaking protocol.



4. Printer control. Memory in the 8741/8041 allows the device to buffer as many as 40 characters to be printed. The main-system processor can transfer a block of data at this speed and then continue with other tasks while the UPI's bidirectional I/O ports monitor and control sequential character printing.



5. Using the UPI. Typical applications of the 8741/8041 include (a) a keyboard scanner in which an 8243 input/output expander is added to permit the encoding of as many as 128 keys, (b) a process-control subsystem implemented with an analog multiplexer and a digital-to-analog converter, and (c) a generalized distributed processing system that employs up to seven of the devices as slave processors, connecting them to a single 8048 microcomputer.

processor can therefore be left to down-load commands and transfer data, while the UPI works in real time.

One application might be the controlling of a printer peripheral to an 8080 system, as shown in Fig. 4. The entire real-time control portion of the task can be handled by the peripheral controller. With its built-in timer, it easily handles timing functions like character spacing, print position, and line feed. The UPI has ample I/O ports for a 40-column dot-matrix printer.

In this printer application, the DBB register allows for standardization of data transfer to and from the 8080-based main processing system. To do this, one typical format might be for the main processor to send a start command followed by a full line of 40 ASCII characters. The peripheral controller would then store the characters under program control in a portion of the RAM and begin execution of the printing as soon as the print head and line feed were in the proper position. In the meantime, the main processor returns to other tasks. The ROM in the 8041/8741 can be used to convert the ASCII code to dot-matrix or other formats.

In printer applications, standardization is the key feature offered by the slave peripheral controller. Without any changes in the 8080-based main processing system, the UPI can be programmed to handle any printer mechanisms—dot matrix, drum, spherical head, and so on. In this way, a designer can easily upgrade the peripherals in his system with a minimum of change in the master-processor software.

A keyboard application

Figure 5a illustrates an application in which the new chip plus an 8243 I/O expander provide a compact system for scanning and encoding as many as 128 keys from a terminal keyboard. N-key rollover and debounce are implemented by using the on-chip RAM to keep a copy of the key status after each scan. When a key closure is detected, the 8041 uses a ROM look-up table to generate the appropriate ASCII code for transfer to the master processor. As many as 16 characters can be stored and transferred in a block to the master processor.

The analog process-control subsystem illustrated in Fig. 5b can be implemented using an analog multiplexer and digital-to-analog converter along with the 8041. In this configuration, the peripheral controller can monitor and digitize eight analog inputs, perform linearization (using equations or ROM look-up tables), check for limits and zero offsets, and receive control information that could determine new limits.

Figure 5c illustrates a generalized distributed-processing system using as many as seven 8041s as slave processors connected to a single 8048 master processor. Port 2 of the 8048 provides seven chip-select lines to the peripheral controllers plus the command/data control function. This low-cost, low-speed multiprocessor configuration has many advantages over a single high-speed processor. The peripheral controllers are designed especially for control or interface applications, and each can be programmed to handle a single isolated task. This modular approach allows easy development and debugging of the system. □

MICROCOMPUTER INTERFACING: CHARACTERISTICS OF THE 8253 PROGRAMMABLE INTERVAL TIMER

Marvin L. DeJong
School of the Ozarks

Jonathan A. Titus and Christopher Titus
Tychon, Inc

Peter R. Rony and David G. Larsen
Virginia Polytechnic Institute and State University

As a preliminary discussion, some characteristics of the Intel 8253 programmable interval timer are presented. This extremely versatile input/output chip has various potential uses such as a real-time clock, event counter, and period counter, in addition to replacing software-implemented timing loops. For example, interval timers have been used in a digital cardiotachometer, a data-logging timer that employed several phototransistors to measure velocities and accelerations, and a program to sample nonperiodic waveforms for subsequent display on an oscilloscope.*

The 8253 is a 24-pin integrated circuit that requires a single 5-V supply and contains three independent 16-bit interval timers, each of which can be operated in six different modes. An interval timer is a device for measuring the time interval between two actions, or a timer that switches electrical circuits on or off for the duration

*Dr DeJong of the Dept of Mathematics/Physics at the School of the Ozarks, Point Lookout, Mo has implemented the timers in these simple, but diverse, applications.

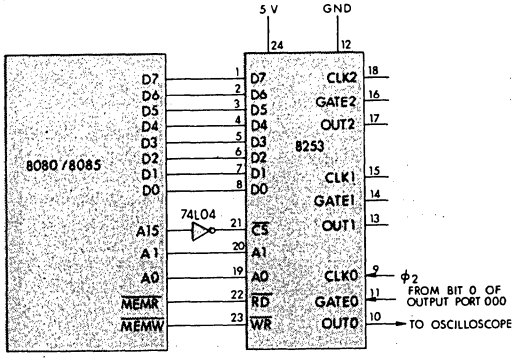


Fig 1 Interface circuit between 8253 programmable interval timer and 8080/8085 microcomputer. Timer uses four locations of memory in this memory-mapped interface circuit

TABLE 1

Addressing the 8253 Programmable Interval Timer

Control Inputs						Memory Address in Demonstration Program and Interface Circuit
$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0		
0	1	0	0	0	Load counter #0	200 000
0	1	0	0	1	Load counter #1	200 001
0	1	0	1	0	Load counter #2	200 002
0	1	0	1	1	Load control register	200 003
0	0	1	0	0	Read counter #0	200 000
0	0	1	0	1	Read counter #1	200 001
0	0	1	1	0	Read counter #2	200 002
0	0	1	1	1	No operation (3-state)	—
1	X	X	X	X	Disable chip (3-state)	—
0	1	1	X	X	No operation (3-state)	—

Note: X = don't care (logic 0 or logic 1)

of the preset time interval.[1] Fig 1 serves the dual purpose of giving the pin diagram of the 8253 chip, while showing how the chip can be interfaced with an 8080A/8085 based microcomputer system using memory-mapped input/output (I/O).[2]

Four internal registers—three interval timers and a control register—that are decoded as memory locations 200 000 through 200 003 with the aid of the address bus signals A0, A1, and A15 (see Fig 1 and Table 1) are contained on the 8253 chip. In Table 1, the $\overline{RD}$ and $\overline{WR}$ control inputs determine whether a specific register is being loaded or read. It is not possible to read the contents of the control register.

Table 2 summarizes the coding for the 8-bit control register within the chip. Bits D7 and D6 determine the selection of the interval timer; bits D5 and D4 determine the nature of the read/write operation associated with the chosen timer; bits D3, D2, and D1, the mode of operation of the timer; and bit D0, whether the timer counts down in binary or binary-coded decimal (BCD).

Fig 2 provides a block diagram for a typical counter in the chip. The microcomputer loads the 16-bit down

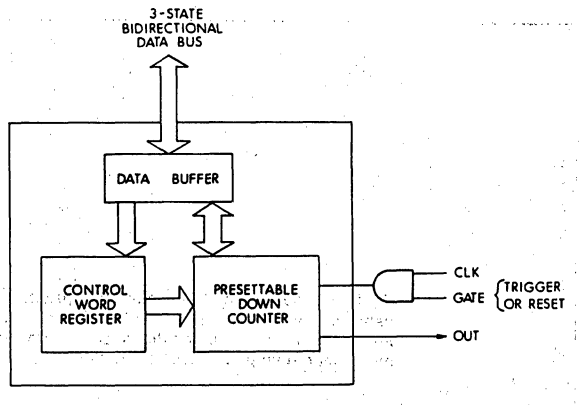


Fig 2 Functional diagram of each of three 16-bit interval timers in 8253 chip. Gate input acts alternatively as gate, trigger, or reset input, depending upon mode chosen

counter as two successive bytes, a HI and LO byte, via the bidirectional data bus, D0 through D7. If the gate line, GATE, is active, negative edge transitions at the CLK input decrement the counter. When the counter reaches zero, OUT becomes active, its actual behavior depending upon the mode programmed into the control register for the counter (see Table 2). The three 16-bit counters on the chip can each be programmed independently in any one of the six modes of operation. Counter inputs and outputs—CLK, GATE, and OUT—for the chosen counter are independent of the CLK, GATE, and OUT I/O of the remaining two counters on the chip.

In addition to the address, data, and control bus connections shown in Fig 1, the CLK0 and GATE0 inputs to counter 0 are respectively connected to the $\phi 2$ [transistor-transistor logic (TTL)] microcomputer clock output (typically 2 MHz) and to bit 0 of accumulator output port 000. Any TTL level clock with a frequency of less than 2 MHz can be used as input to CLK0, and any suitably debounced switch or source of strobe pulses can be used to control the timer at GATE0. The output of the counter, OUT0, can be connected to an oscilloscope to permit observation of each of the six timer modes of operation.

Next month's discussion will focus on the behavior of a demonstration program for the 8253 programmable peripheral interface chips, which are further described in Refs 3 and 4. This program will illustrate the loading, latching, and reading of counter 0 as well as the various output modes.

TABLE 2

Coding for 8-Bit Control Register in 8253 Chip

Bits		Control Function
D7	D6	
0	0	Control word is for counter #0
0	1	Control word is for counter #1
1	0	Control word is for counter #2
1	1	———
	D5	D4
	0	0 Latch both bytes of chosen counter for read operation
	0	1 Load or read only most significant byte (MSB) of chosen counter
	1	0 Load or read only least significant byte (LSB) of chosen counter
	1	1 Load or read LSB first, then MSB of chosen counter
D3	D2	D1
0	0	0 Mode 0: Output = 1 on zero counter
0	0	1 Mode 1: Retriggerable variable-width one-shot
X	1	0 Mode 2: Programmable rate generator
X	1	1 Mode 3: Programmable square wave generator
1	0	0 Mode 4: Delayed strobe (software triggered strobe)
1	0	1 Mode 5: Triggered strobe (hardware triggered strobe)
		D0
		0 Count down in binary
		1 Count down in BCD

Note: X = don't care (logic 0 or logic 1)

References

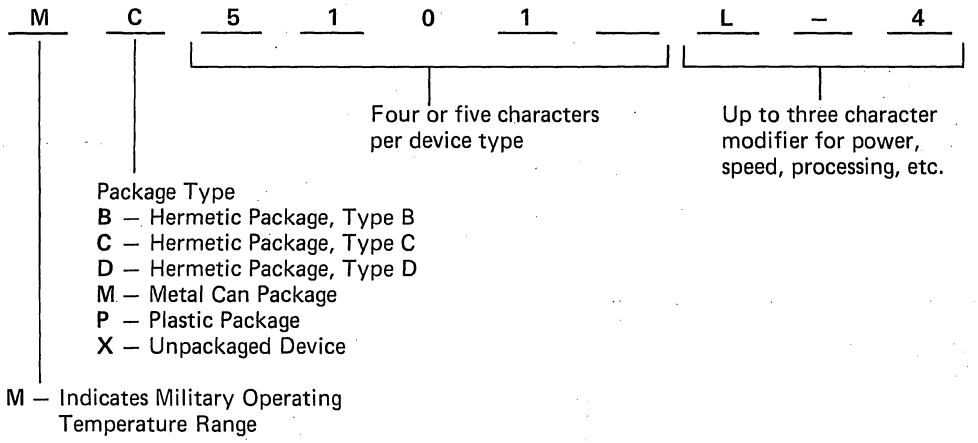
1. R. F. Graf, *Modern Dictionary of Electronics*, Howard W. Sams & Co, Indianapolis, Ind, 1972, p 298
2. D. G. Larsen, P. R. Rony, and J. A. Titus, *The Bugbook*[R] *VI. 8080A Microcomputer Programming and Interfacing*, E & L Instruments, Inc, Derby, Conn, 1977, p 21-1
3. *Intel Data Catalog 1977*, Intel Corp, 3065 Bowers Ave, Santa Clara, CA 95051, pp 10-159 (Price, $2.50)
4. A. Osborne, *An Introduction to Microcomputers, Vol II. Some Real Products*, Osborne and Associates, Berkeley, Calif, 1976, pp 4-106

This article is based, with permission, on a column appearing in *American Laboratory* magazine.

ORDERING INFORMATION

Semiconductor components are identified as follows:

Example:



Examples:

P5101L	CMOS 256 X 4 RAM, low power selection, plastic package, commercial temperature range.
C8080A2	8080A Microprocessor with 1.5 μs cycle time, hermetic package Type C, commercial temperature range.
MD3604/C	512 X 8 PROM, hermetic package Type D, military temperature range, MIL-STD-883 Level C processing.*
MC8080A/B	8080A Microprocessor, hermetic package Type C, military temperature range, MIL-STD-883 Level B processing.*

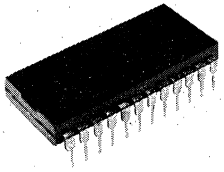
Kits, boards and systems may be ordered using the part number designations in this catalog.

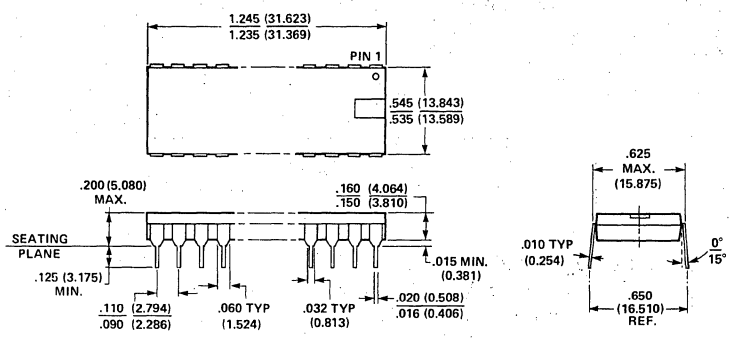
The latest Intel OEM price book should be consulted for availability of various options. These may be obtained from your local Intel representative or by writing directly to Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051.

**On military temperature devices, B suffix indicates MIL-STD-883 Level B processing. Suffix C indicates MIL-STD-883 Level C processing. "S" number suffixes must be specified when entering any order for military temperature devices. All orders requesting source inspection will be rejected by Intel.*

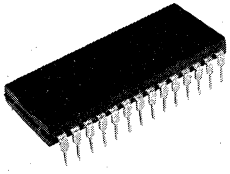
PLASTIC DUAL IN-LINE PACKAGE TYPE P

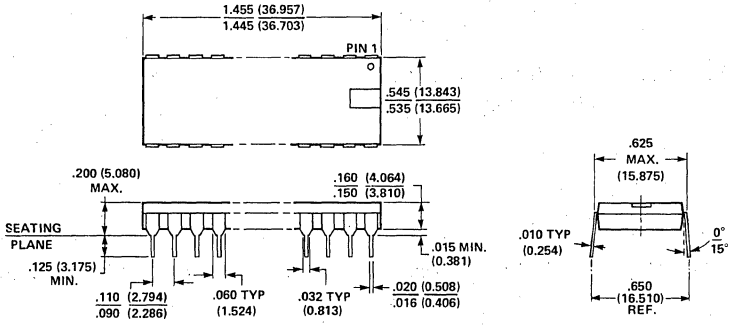
24-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P



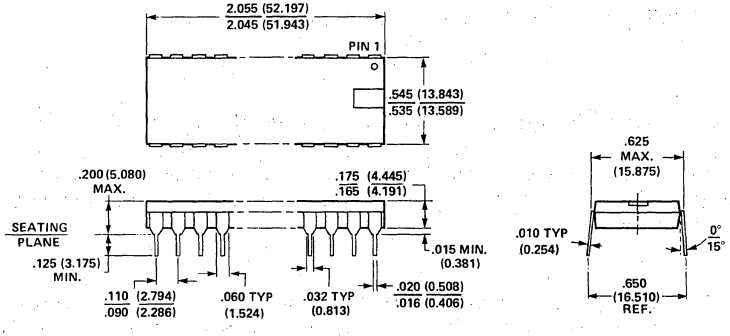


28-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P



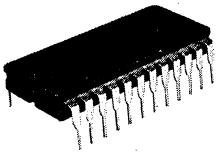


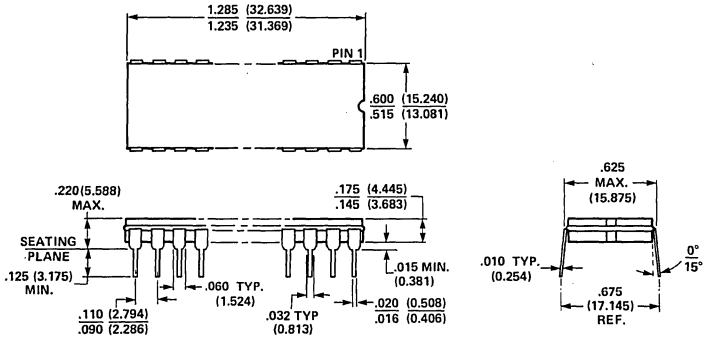
40-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P



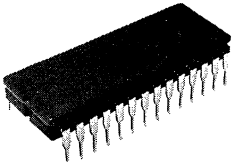
CERAMIC DUAL IN-LINE PACKAGE TYPE D

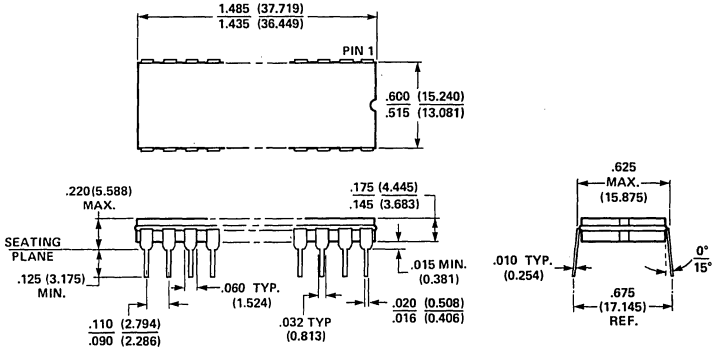
24-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D



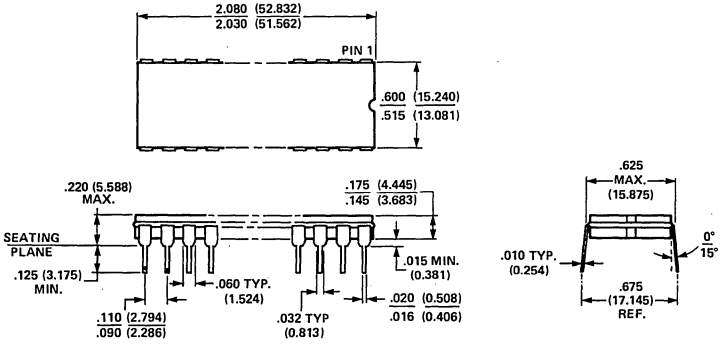


28-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D

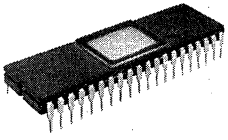


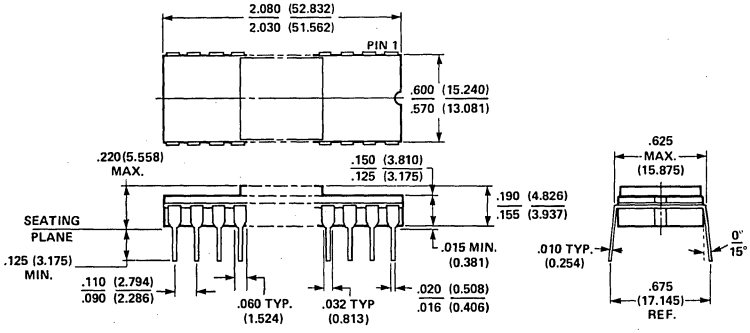


40-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D



40-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE B





NOTES

intel®

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080
TWX: 910-338-0026
TELEX: 34-6372

U.S. AND CANADIAN DISTRIBUTORS

ALABAMA

†Hamilton/Avnet Electronics
4692 Commercial Drive
Huntsville 35805
Tel: (205) 837-7210

Pioneer
1207 Putman Drive NW
Huntsville 35805
Tel: (205) 837-9300

ARIZONA

†Hamilton/Avnet Electronics
2615 South 21st Street
Phoenix 85034
Tel: (602) 275-7851

†Liberty/Arizona
8155 N. 24th Avenue
Phoenix 85021
Tel: (602) 249-2232
TELEX: 910-951-4282

CALIFORNIA

†Avnet Electronics
350 McCormick Avenue
Costa Mesa 92626
Tel: (714) 754-6111

†Hamilton/Avnet Electronics
1175 Bordeaux Drive
Sunnyvale 94086
Tel: (408) 743-3355

†Hamilton/Avnet Electronics
8917 Complex Drive
San Diego 92123
Tel: (714) 279-2421

†Hamilton Electro Sales
10912 W. Washington Boulevard
Culver City 90230
Tel: (213) 558-2121

†Liberty Electronics
124 Maryland Street
El Segundo 90245
Tel: (213) 322-5059
TWX: 910-348-7140

†Liberty/San Diego
8284 Mercury Court
San Diego 92111
Tel: (714) 565-9171
TELEX: 910-335-1590

†Elmar Electronics
2288 Charleston Road
Mountain View 94040
Tel: (415) 961-3611
TELEX: 910-379-6437

Hamilton/Avnet Electronics
17312 Eastman Street
Irvine 92714
Tel: (714) 979-6864

COLORADO

†Elmar/Denver
6777 E. 50th Avenue
Commerce City 80022
Tel: (303) 287-9611
TWX: 910-936-0770

†Hamilton/Avnet Electronics
5921 No. Broadway
Denver 80216
Tel: (303) 534-1212

CONNECTICUT

†Cramer/Connecticut
12 Beaumont Road
Wallingford 06492
Tel: (203)265-7741

†Hamilton/Avnet Electronics
643 Danbury Road
Georgetown 06829
Tel: (203) 762-0361

†Harvey Electronics
112 Main Street
Norwalk 06851
Tel: (203) 853-1515

FLORIDA

Arrow Electronics
1001 N.W. 62nd Street
Suite 402
Ft. Lauderdale 33309
Tel: (305) 776-7790

Arrow Electronics
115 Palm Bay Road, NW
Suite 10
Palm Bay 32905
Tel: (305) 725-1480

†Hamilton/Avnet Electronics
6800 Northwest 20th Ave.
Ft. Lauderdale 33309
Tel: (305) 971-2900

†Pioneer
6220 S. Orange Blossom Trail
Suite 412
Orlando 32809
Tel: (305) 859-3600

Hamilton/Avnet
3197 Tech. Drive N.
St. Petersburg 33702
Tel: (813) 576-3930

GEORGIA

Arrow Electronics
3406 Oak Cliff Road
Doraville 30340
Tel: (404) 455-4054

†Hamilton/Avnet Electronics
6700 I 85, Access Road, #11
Norcross 30071
Tel: (404) 448-0800

ILLINOIS

†Cramer/Chicago
1911 So. Busse Rd.
Mt. Prospect 60056
Tel: (312) 593-8230

†Hamilton/Avnet Electronics
3901 No. 25th Ave.
Schiller Park 60176
Tel: (317) 849-7300

Pioneer/Chicago
1551 Carmen Drive
Elk Grove Village 60006
Tel: (312) 437-9680

INDIANA

†Pioneer/Indiana
6408 Castleplace Drive
Indianapolis 46250
Tel: (317) 849-7300

Sheridan Sales
8790 Purdue Road
Indianapolis 46268
Tel: (317) 297-3146

KANSAS

†Hamilton/Avnet Electronics
9219 Quivira Road
Overland Park 66215
Tel: (913) 888-8900

MARYLAND

†Hamilton Avnet
P.O. Box 647,
BWI Airport
7235 Standard Drive
Hanover 21076
Tel: (301) 796-5684

†Pioneer/Washington
9100 Gaither Road
Gaithersburg 20760
Tel: (301) 948-0710
TWX: 710-828-0545

MASSACHUSETTS

†Cramer Electronics Inc.
85 Wells Avenue
Newton 02159
Tel: (617) 969-7700

MASSACHUSETTS (continued)

†Hamilton/Avnet Electronics
50 Tower Office Park
Woburn 01801
Tel: (617) 273-7500

MICHIGAN

†Sheridan Sales Co.
24543 Indoplex Circle
Farmington Hills 48024
Tel: (313) 477-3800

†Pioneer/Michigan
13485 Stamford
Livonia 48150
Tel: (313) 525-1800

†Hamilton/Avnet Electronics
32487 Schoolcraft Road
Livonia 48150
Tel: (313) 522-4700
TWX: 810-242-8775

MINNESOTA

†Industrial Components
5280 West 74th Street
Minneapolis 55435
Tel: (612) 831-2666

†Cramer/Bonn
5424 Edina Industrial Blvd.
Edina 55435
Tel: (612) 835-7811

†Hamilton/Avnet Electronics
7449 Cahill Road
Edina 55435
Tel: (612) 941-3801

MISSOURI

†Hamilton/Avnet Electronics
364 Brookes Drive
Hazelwood 63042
Tel: (314) 731-1144

Sheridan Sales
220 S. Hwy 67, Suite 10
Florissant 63031
Tel: (314) 837-5200

NEW JERSEY

Arrow Electronics
Pleasant Valley Avenue
Moorestown 08057
Tel: (215) 928-1800

Arrow Electronics
285 Midland Avenue
Saddlebrook 07662
Tel: (201) 797-5800

†Hamilton/Avnet Electronics
10 Industrial Road
Fairfield 07006
Tel: (201) 575-3390
TWX: 710-994-5787

†Harvey Electronics
45 Route 46
Pinebrook 07058
Tel: (201) 227-1262

†Hamilton/Avnet Electronics
113 Gaither Drive
East Gate Industrial Park
Mt. Laurel 08057
Tel: (609) 234-1233
TWX: 710-897-1405

NEW MEXICO

Alliance Electronics Inc.
11728 Linn Ave., N.E.
Albuquerque 87123
Tel: (505) 292-3360

†Hamilton/Avnet Electronics
2524 Baylor Drive, S.E.
Albuquerque 87119
Tel: (505) 765-1500

†Microcomputer System Technical Demonstrator Centers
*Microcomputer Systems Spares Order Point

U.S. AND CANADIAN SALES OFFICES

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080
TWX: 910-338-0026
TELEX: 34-6372

ALABAMA
Intel Corp.
3322 S. Parkway, Ste. 71
Holiday Office Center
Huntsville 35802
Tel: (205) 883-2430

Glen White Associates
3502 9th Avenue
Huntsville 35805
Tel: (205) 883-9394

†Pen-Tech Associates, Inc.
Holiday Office Center
3322 S. Memorial Pkwy.
Huntsville 35801
Tel: (205) 533-0090

ARIZONA
Intel Corp.
8650 N. 35th Avenue, Suite 101
Phoenix 85021
Tel: (602) 242-7205

†BFA
4426 North Saddle Bag Trail
Scottsdale 85251
Tel: (602) 994-5400

CALIFORNIA
Intel Corp.
7670 Opportunity Rd.
Suite 135
San Diego 92111
Tel: (714) 268-3563

Intel Corp.*
1651 East 4th Street
Suite 105
Santa Ana 92701
Tel: (714) 835-9642
TWX: 910-595-1114

Intel Corp.*
15335 Morrison
Suite 345
Sherman Oaks 91403
(213) 986-9510
TWX: 910-495-2045

Intel Corp.*
3375 Scott Blvd.
Santa Clara 95051
Tel: (408) 987-8086
TWX: 910-339-9279
TWX: 910-338-0255

Earle Associates, Inc.
4805 Mercury Street
Suite L
San Diego 92111
Tel: (714) 287-5441

Mac-I
2576 Shattuck Ave.
Suite 4B
Berkeley 94704
Tel: (415) 843-7625

Mac-I
P.O. Box 1420
Cupertino 95014
Tel: (408) 257-9880

Mac-I
P.O. Box 8763
Fountain Valley 92708
Tel: (714) 839-3341

Mac-I
20121 Ventura Blvd., Suite 240E
Woodland Hills 91364
Tel: (213) 347-5900

COLORADO
Intel Corp.*
6000 East Evans Ave.
Bldg. 1, Suite 260
Denver 80222
Tel: (303) 758-8086
TWX: 910-931-2289

†Westex
27972 Meadow Drive
P.O. Box 1355
Evergreen 80439
Tel: (303) 674-5255

CONNECTICUT
Intel Corp.
Peacock Alley
1 Padanaram Road, Suite 146
Danbury 06810
Tel: (203) 792-8366
TWX: 710-456-1199

FLORIDA
Intel Corp.
1001 N.W. 62nd Street, Suite 406
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

Intel Corp.
5151 Adanson Street, Suite 203
Orlando 32804
Tel: (305) 628-2393
TWX: 810-853-9219

†Pen-Tech Associates, Inc.
201 S.E. 15th Terrace, Suite F
Deerfield Beach 33441
Tel: (305) 421-4989

†Pen-Tech Associates, Inc.
111 So. Maitland Ave., Suite 202
Maitland 32751
Tel: (305) 645-3444

GEORGIA
†Pen-Tech Associates, Inc.
Suite 305 C
2101 Powers Ferry Road
Atlanta 30339
Tel: (404) 955-0293

ILLINOIS
Intel Corp.*
900 Jorie Boulevard
Suite 220
Oakbrook 60521
Tel: (312) 325-9510
TWX: 910-651-5881

†Dytek-Central, Inc.
121 So. Wilke Road
Suite 304
Arlington Heights 60005
Tel: (312) 394-3380
TWX: 910-687-2267

INDIANA
Electro Reps Inc.
941 E. 86th Street, Suite 101
Indianapolis 46240
Tel: (317) 255-4147
TWX: 810-341-3217

IOWA
Technical Representatives, Inc.
St. Andrews Building
1930 St. Andrews Drive N.E.
Cedar Rapids 52405
Tel: (319) 393-5510

KANSAS
Technical Representatives, Inc.
8245 Nieman Road, Suite #100
Lenexa 66214
Tel: (913) 888-0212, 3, & 4
TWX: 910-749-6412

KENTUCKY
†Lowry & Associates, Inc.
3351 Commodore
Lexington 40502
Tel: (606) 269-6329

MARYLAND
Intel Corp.*
7257 Parkway Drive
Hanover 21076
Tel: (301) 796-7500
TWX: 710-862-1944

Glen White Associates
57 W. Timonium Road, Suite 307
Timonium 21093
Tel: (301) 252-6360

†Mesa Inc.
11900 Parklawn Drive
Rockville 20852
Tel: Wash. (301) 881-8430
Balto. (301) 792-0021

MASSACHUSETTS
Intel Corp.*
187 Billerica Road, Suite 14A
Chelmsford 01824
Tel: (617) 667-8126
TWX: 710-343-6333

†Computer Marketing, Inc.
257 Crescent Street
Waltham 02154
Tel: (617) 894-7000

MICHIGAN
Intel Corp.
26500 Northwestern Hwy.
Suite 401
Southfield 48075
Tel: (313) 353-0920
TWX: 910-420-1212
TELEX: 2 31143

†Lowry & Associates, Inc.
135 W. North Street
Suite 4
Brighton 48116
Tel: (313) 227-7067

MINNESOTA
Intel Corp.
8200 Normandale Avenue
Suite 422
Bloomington 55437
Tel: (612) 835-6722
TWX: 910-576-2867

†Dytek North
1821 University Ave.
Room 163N
St. Paul 55104
Tel: (612) 645-5816

MISSOURI
Technical Representatives, Inc.
320 Brookes Drive, Suite 104
Hazelwood 63042
Tel: (314) 731-5200
TWX: 910-762-0618

NEW JERSEY
Intel Corp.
1 Metroplaza Office Bldg.
505 Thornall St.
Edison 08817
Tel: (201) 494-5040
TWX: 710-480-6238

NEW MEXICO
BFA Corporation
P.O. Box 1237
Las Cruces 88001
Tel: (505) 523-0601
TWX: 910-983-0543

BFA Corporation
3705 Westerfield, N.E.
Albuquerque 87111
Tel: (505) 292-1212
TWX: 910-989-1157

NEW YORK
Intel Corp.*
350 Vanderbilt Motor Pkwy.
Suite 402
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

Intel Corp.
80 Washington St.
Poughkeepsie 12601
Tel: (914) 473-2303
TWX: 510-248-0060

Intel Corp.
474 Thurston Road
Rochester 14619
Tel: (716) 328-7340
TWX: 510-253-3841

†Measurement Technology, Inc.
159 Northern Boulevard
Great Neck 11021
Tel: (516) 482-3500

T-Squared
4054 Newcourt Avenue
Syracuse 13206
Tel: (315) 463-8592
TWX: 710-541-0554

†T-Squared
2 E. Main
Victor 14564
Tel: (716) 924-9101
TELEX: 97-8289

NORTH CAROLINA
†Pen-Tech Associates, Inc.
P.O. Box 5382
Highpoint 27262
Tel: (919) 883-9125

Glen White Associates
3700 Computer Drive
Suite 330
Raleigh 27609
Tel: (919) 787-7016

OHIO
Intel Corp.*
8312 North Main Street
Dayton 45415
Tel: (513) 890-5350
TWX: 810-450-2528

Intel Corp.*
Chagrin-Brainard Bldg. #201
28001 Chagrin Blvd.
Cleveland 44122
Tel: (216) 464-2736

Lowry & Associates, Inc.
24200 Chagrin Blvd.
Suite 320
Cleveland 44122
Tel: (216) 464-8113

†Lowry & Associates, Inc.
1524 Marsetta Drive
Dayton 45432
Tel: (513) 429-9040

†Lowry & Associates, Inc.
1050 Freeway Dr., N.
Suite 209
Columbus 43229
Tel: (614) 436-2051

OREGON
ES/Chase Company
4095 SW 144th St.
Beaverton 97005
Tel: (503) 641-4111

PENNSYLVANIA
Intel Corp.*
275 Commerce Dr.
200 Office Center
Suite 212
Fort Washington 19034
Tel: (215) 542-9444
TWX: 510-661-2077

†Lowry & Associates, Inc.
Seven Parkway Center
Suite 455
Pittsburgh 15520
Tel: (412) 922-5110

†Q.E.D. Electronics
300 N. York Road
Hatboro 19040
Tel: (215) 674-9600

TENNESSEE
Glen White Associates
Rt. #12, Norwood S/D
Jonesboro 37659
Tel: (615) 477-8850

Glen White Associates
2523 Howard Road
Germantown 38138
Tel: (901) 754-0483

Glen White Associates
6446 Ridge Lake Road
Hixon 37343
Tel: (615) 842-7799

TEXAS
Intel Corp.*
2925 L.B.J. Freeway
Suite 175
Dallas 75234
Tel: (214) 241-9521
TWX: 910-860-5487

Intel Corp.*
6776 S.W. Freeway
Suite 550
Houston 77074
Tel: (713) 784-3400

Mycrosystems Marketing Inc.
13777 N. Central Expressway
Suite 405
Dallas 75243
Tel: (214) 238-7157
TWX: 910-867-4763

Mycrosystems Marketing Inc.
6610 Harwin Avenue, Suite 125
Houston 77036
Tel: (713) 783-2900

UTAH
†Westek
3788 Brockbank Drive
Salt Lake City 84117
Tel: (801) 278-6920

VIRGINIA
Glen White Associates
P.O. Box 1104
Lynchburg 24505
Tel: (804) 384-6920

Glen White Associates
Rt. #1, Box 322
Colonial Beach 22443
Tel: (804) 224-4871

WASHINGTON
Intel Corp.
300 120th Avenue N.E.
Bldg. 2, Suite 202
Bellevue 98005
Tel: (206) 453-8086

E.S./Chase Co.
P.O. Box 80903
Seattle 98198
Tel: (206) 762-4824
TWX: 910-444-2298

WISCONSIN
Intel Corp.
4369 S. Howell Ave.
Milwaukee 53207
Tel: (414) 747-0789

CANADA
Intel Corp.
Suite 233, Bell Mews
39 Highway 7, Bell Corners
Ottawa, Ontario K2H 8R2
Tel: (613) 829-9714
TELEX: 053-4419

Multilek, Inc.*
15 Grenfell Crescent
Ottawa, Ontario K2G 0G3
Tel: (613) 226-2365
TELEX: 053-4585

*Field application location
†These representatives do not offer Intel Components, only boards and systems.

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) 987-8080

Printed in U.S.A./T-3/279/30K CP