



THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN IS PROPRIETARY TO AND THE EXCLUSIVE PROPERTY OF HONEYWELL INFORMATION SYSTEMS INC. THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN IS FOR THE USE OF HONEYWELL AUTHORIZED RECIPIENTS ONLY FOR THE MAINTENANCE AND OPERATION OF HONEYWELL PRODUCTS AND MUST BE MAINTAINED IN STRICTEST CONFIDENCE. IT MUST NOT BE REPRODUCED IN WHOLE OR IN PART. THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN SHALL NOT BE DISCLOSED TO ANY OTHER PARTY WITHOUT THE PRIOR WRITTEN CONSENT OF HONEYWELL.

SERIES 60 LEVEL 6

**TYPE MDC9101
MULTIPLE DEVICE
CONTROLLER
MANUAL**

Doc. No. 71010220-300 Order. No. FL19, Rev. 2

Honeywell

THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN IS PROPRIETARY TO AND THE EXCLUSIVE PROPERTY OF HONEYWELL INFORMATION SYSTEMS INC. THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN IS FOR THE USE OF HONEYWELL AUTHORIZED RECIPIENTS ONLY FOR THE MAINTENANCE AND OPERATION OF HONEYWELL PRODUCTS AND MUST BE MAINTAINED IN STRICTEST CONFIDENCE. IT MUST NOT BE REPRODUCED IN WHOLE OR IN PART. THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN SHALL NOT BE DISCLOSED TO ANY OTHER PARTY WITHOUT THE PRIOR WRITTEN CONSENT OF HONEYWELL.

SERIES 60 LEVEL 6

TYPE MDC9101 MULTIPLE DEVICE CONTROLLER MANUAL

Doc. No. 71010220-300 Order. No. FL19, Rev. 2

RECORD OF REVISIONS

REVISION	DATE	AUTHORITY	AFFECTED PAGES
-100	Dec. 1975	---	--
-200	April 1976	BLCO 60625	--
-300	March 1977	BLCO 60215	--
		BLCO 60222	
		BLCO 60734	

Hardware Publications, M&TO, Billerica, MA 01821

Printed in the United States of America
 All rights reserved

0220/FL19

CONTENTS

Section		Page
I	INTRODUCTION	1-1
	1.1 Scope and Purpose	1-1
	1.2 Unit Description	1-2
	1.3 Attachable Adapters and Devices	1-2
	1.4 Specifications	1-2
	1.5 Reference Documents	1-3
II	THEORY OF OPERATION - OVERVIEW	2-1
	2.1 General Functional Description	2-1
	2.2 Software	2-2
	2.3 Firmware	2-16
	2.4 Hardware	2-16
	2.4.1 Megabus/MDC Interface	2-16
	2.4.2 MDC/Adapter Interface	2-16
	2.4.3 Microprogram Control Store	2-24
	2.4.4 Arithmetic Logic Unit (ALU)/Accumulator (ACU)	2-24
	2.4.5 Scratch-Pad Memory (SPM)	2-25
	2.4.6 Megabus Logic	2-25
	2.4.7 Adapter/Channel Control Logic	2-25
	2.4.8 Test Multiplexer	2-26
	2.4.9 Clock Logic	2-26
	2.5 Output Data Operation	2-26
	2.5.1 Software Initiation	2-27
	2.5.2 Firmware Control Through Hardware Implementation	2-27
	2.6 Input Data Operation	2-28
III	THEORY OF OPERATION - INTERMEDIATE	3-1
	3.1 Microprogram Control Store Functional Components	3-1
	3.1.1 Subroutine Return Address Register (SRAR)	3-1
	3.1.2 Microprogram Address Selector (UPAS)	3-2
	3.1.3 Microprogram Address Counter (UPAC)	3-2

CONTENTS

Section		Page
	3.1.4 Microprogram Control Store (UPCS)	3-2
	3.1.5 Diagnostic Instruction (Test) Gate	3-2
	3.1.6 Microprogram Instruction Register (UPIR)	3-4
	3.1.7 Op-Code Decoder (OPCOD)	3-4
	3.1.8 Scan Logic	3-4
3.2	Arithmetic Logic Unit and Accumulator Functional Components	3-6
	3.2.1 A-Operand Multiplexer (ALUAX)	3-6
	3.2.2 B-Operand Multiplexer (ALUBX)	3-8
	3.2.3 Arithmetic Logic Unit (ALU)	3-9
	3.2.4 Accumulator (ACU)	3-9
3.3	Scratch Pad Memory Functional Components	3-11
	3.3.1 Scratch Pad Memory Index Control Flip-Flop (SPMICF)	3-11
	3.3.2 Scratch Pad Memory Index Register (SPMIR)	3-11
	3.3.3 Scratch Pad Memory Address Counter (SPMAC)	3-12
	3.3.4 Scratch Pad Memory Address Selector (SPMAS)	3-12
	3.3.5 Scratch Pad Memory (SPM)	3-12
3.4	Megabus Logic Functional Components	3-14
	3.4.1 Bus Data Register (BDR)	3-14
	3.4.1.1 Parallel Load Operation	3-14
	3.4.1.2 Byte-Serial Load Operation	3-14
	3.4.2 Bus Data Register Control	3-16
	3.4.3 Master Cycle Logic and Priority Network	3-17
	3.4.4 Slave Response Logic	3-18
3.5	Channel Control Functional Description	3-22
	3.5.1 Channel Ready Signals	3-22
	3.5.2 Adapter Enable Gate	3-22
	3.5.3 Adapter Control Signals	3-22
	3.5.4 Adapter Register Control Signals	3-22
	3.5.5 Request Logic	3-22
3.6	Test Multiplexer Functional Description	3-24

HONEYWELL PROPRIETARY AND CONFIDENTIAL

CONTENTS

Section		Page
3.7	Clock and Clock Control Functional Description	3-25
3.8	Data Flow and Control Paths	3-27
3.8.1	Initiation of Output Operation	3-27
3.8.2	Output Operation Execution	3-28
3.8.3	Termination of Output Operation	3-29
3.8.4	Initiation of Input Operation	3-29
3.8.5	Input Operation Execution	3-29
3.8.6	Termination of Input Operation	3-30
IV	THEORY OF OPERATION - CYCLE FLOW	4-1
4.1	Microinstructions	4-1
4.1.1	Miscellaneous Commands	4-1
4.1.2	Device Adapter Commands	4-2
4.1.3	Megabus Logic Commands	4-2
4.1.4	ALU Commands	4-2
4.1.5	Constant Commands	4-2
4.1.6	Memory Commands	4-3
4.1.7	Test and Return Commands	4-3
4.1.8	Branch Commands	4-3
4.2	Scratch Pad Memory (SPM)	4-3
4.3	Flow Charts	4-10
4.3.1	Flow Chart Symbology	4-10
4.3.2	Flow Chart Organization	4-17
4.3.3	Usage	4-18
4.4	Firmware Cycle Flow	4-19
4.4.1	Quality Logic Test	4-20
4.4.2	Setup Routine	4-20
4.4.3	Wait Routine	4-20
4.4.4	Bus Request Routine	4-26
4.4.5	Interrupt Routine	4-26
4.4.6	Resume Interrupt Routine	4-26
4.4.7	Point Routine	4-31
4.4.8	DMA Out Routine	4-31
4.4.9	DMA In Routine	4-31

HONEYWELL PROPRIETARY AND CONFIDENTIAL

ILLUSTRATIONS

Figure		Page
1-1	MDC System Block Diagram	1-3
1-2	MDC Interfaces	1-3
2-1	Megabus Configuration for I/O Output Command and MDC Response	2-4
2-2	Megabus Configuration for I/O Input Command and MDC Response	2-4
2-3	Megabus Configuration for MDC Read Main Memory Request	2-5
2-4	Megabus Configuration for MDC Write Main Memory Request	2-5
2-5	Output Control Word	2-6
2-6	Output Interrupt Control	2-6
2-7	Output Task Word	2-6
2-8	I/O Load Output Address	2-7
2-9	I/O Load Output Range	2-7
2-10	Output Configuration Word	2-7
2-11	Input Interrupt Control	2-8
2-12	Input Task Word	2-9
2-13	Input Memory Byte Address	2-10
2-14	Input Memory Module Address	2-10
2-15	Input Range	2-11
2-16	Input Configuration Word	2-12
2-17	Input Status Word 1	2-13
2-18	Input Status Word 2	2-14
2-19	Input Identification Code	2-15
2-20	MDC Hardware Major Block Diagram	2-17
2-21	Megabus/MDC Interface	2-18
2-22	MDC/Adapters Interface	2-21
3-1	MDC Hardware Major Block Diagram	3-3
3-2	Microprogram Control Store Functionality	3-5
3-3	Scan Logic	3-6
3-4	ALU and ACU Functionality	3-7
3-5	Scratch Pad Memory Functionality	3-13
3-6	Bus Data Register	3-15
3-7	Master Cycle Logic and Priority Network	3-20
3-8	Slave Response Logic	3-21
3-9	Channel Controls	3-23
3-10	Test Multiplexer	3-25
3-11	Clock Logic	3-26
4-1	Microinstruction Field Format	4-4
4-2	MDC Firmware Flow	4-5
4-3	Control Word Bit Significance	4-8
4-4	Channel Monitor Byte Bit Structure	4-8
4-5	DMA Flag Byte Bit Structure	4-9
4-6	Resume Interrupt Control Parameter Byte Bit Structure	4-9

CONTENTS

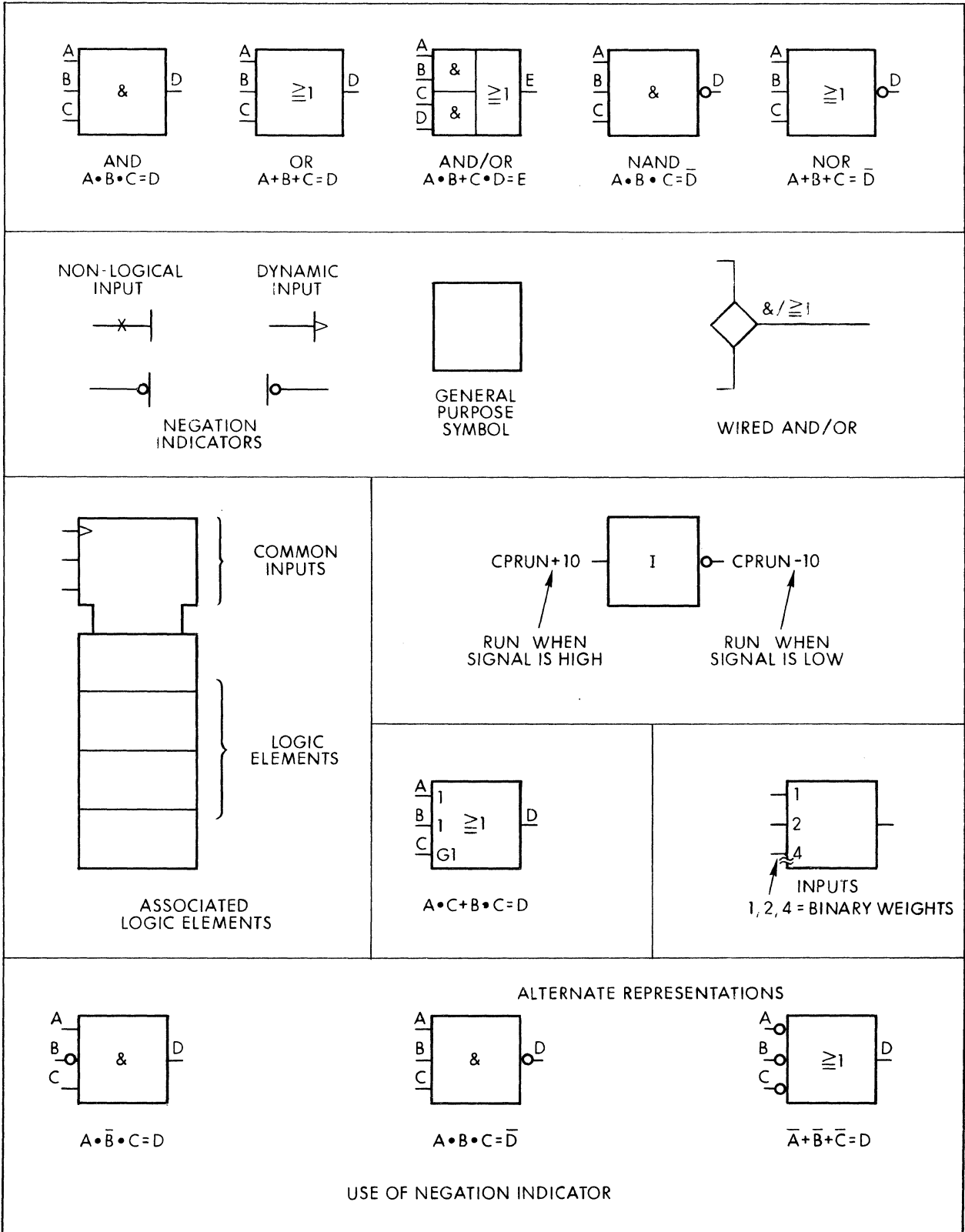
Figure		Page
4-7	Firmware Overview Flow Chart	4-19
4-8	Quality Logic Test	4-21
4-9	Setup Routine	4-24
4-10	Wait Routine	4-25
4-11	Bus Request Routine	4-27
4-12	Interrupt Routine	4-29
4-13	Resume Interrupt	4-30
4-14	Point Routine	4-32
4-15	DMA Out Routine	4-33
4-16	DMA In Routine	4-34

TABLES

1-1	Available MDC Device Adapters	1-4
1-2	Available MDC Devices	1-4
1-3	Reference Documents	1-5
2-1	I/O Commands	2-3
2-2	Megabus/MDC Interface Signal Lines	2-19
2-3	MDC/Adapter Interface Signal Lines	2-22
2-4	MDC/Adapter Interface Line and Mnemonic Cross-Reference List	2-23
3-1	AOP Multiplexer Input Selection	3-7
3-2	BOP Multiplexer Input Selection	3-8
3-3	ALU Functionality	3-10
3-4	ALU Carry Flip-Flop Indications	3-11
3-5	Bus Data Register Application	3-16
3-6	Cycle Parameters	3-17
4-1	Single Scratch Pad Memory Quadrant Topology	4-7
4-2	Scratch Pad Memory Word Description	4-10
4-3	MDC Device Identification Codes	4-12
4-4	Cycle Flow Chart Symbols	4-12



LOGIC SYMBOLY





INTRODUCTION

1.1 SCOPE AND PURPOSE

This product manual describes the functionality and operation of the Type MDC9101 Multiple Device Controller (MDC). Programming considerations are included to aid in understanding the hardware and firmware* descriptions. Detailed programming information is contained in the Minicomputer Handbook (Order No. AS22) or the Peripherals Handbook (Order No. AT04).

Operational theory within this manual is designed to acquaint the user with the major functional hardware and firmware elements in order to aid in analyzing the MDC operation at the detail presented in the logic block diagrams (LBDs).

Detailed theory for the MDC hardware logic and firmware is related directly to the maintenance philosophy of board replacement rather than on-site board repair.

This product manual is comprised of this section and Sections II through IV as follows:

- Section II - Theory of Operation - Overview

This section contains an introduction to the interfaces which the MDC has with Series 60 Level 6 Megabus** network and with the four types of attachable device adapters. It contains a brief description of the interrelationship between software and the MDC hardware and firmware and supplies an operational overview of the functional areas.

*This manual supports Firmware Rev. 77.

**Trademark of Honeywell Information Systems Inc.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

- Section III - Theory of Operation - Intermediate

This section contains the intermediate theory of operation for each area discussed in Section II. Function names are included in many places to aid entry into LBDs supplied in the Type MDC9101 Multiple Device Controller Reference Manual, Order No. FM89.

- Section IV - Theory of Operation - Cycle Flow

This section contains a description of the MDC micro-operations. An overview flow chart illustrates MDC cycle groupings with their entries and exits. Intermediate flow charts illustrate operations within each cycle grouping.

1.2 UNIT DESCRIPTION

The MDC (Figure 1-1) is a microprogrammed peripheral device controller board (BF4MDC), which is attached to the Series 60 Level 6 Megabus. The MDC performs general purpose control functions, such as:

- Execution of Megabus sequences
- Command decoding
- Data transfer to or from device adapters
- Status and control register storage, and
- Direction of the general flow of command execution.

Through the use of device adapters, the MDC controls up to a maximum of four devices. Each device adapter contains hardware unique to a particular type of device. Information from the MDC to the device adapters and from the device adapters to the MDC is transferred in byte form.

The MDC supports two interfaces. They are the Megabus/MDC interface and the MDC/device adapter interface (see Figure 1-2). For a detailed description of these interfaces, refer to subsections 2.4.1 and 2.4.2.

1.3 ATTACHABLE ADAPTERS AND DEVICES

The MDC controls the devices listed in Table 1-1 via the adapters listed in Table 1-2. The device adapters supply information to the devices in the form required by the devices (see device adapter manuals for device-specific information).

The diskette adapter is a double-sized module that can support one or two diskettes. The diskette adapter occupies two of the four available device adapter positions on the MDC. Two diskette adapters can be connected to the MDC.

The card reader adapter, the console adapter, and the printer adapter are all single-sized modules. Each single-sized module can support one device and occupies only one adapter position on the MDC.

1.4 SPECIFICATIONS

The MDC is a solid state module that consists of dual in-line packages (DIPs) mounted on a multilayer Series 60 Level 6 Controller. The MDC has eight 25-pin in-line connectors on it for mounting the device adapters.

1.5 REFERENCE DOCUMENTS

The reference documents listed in Table 1-3 contain information on the MDC at a systems level.

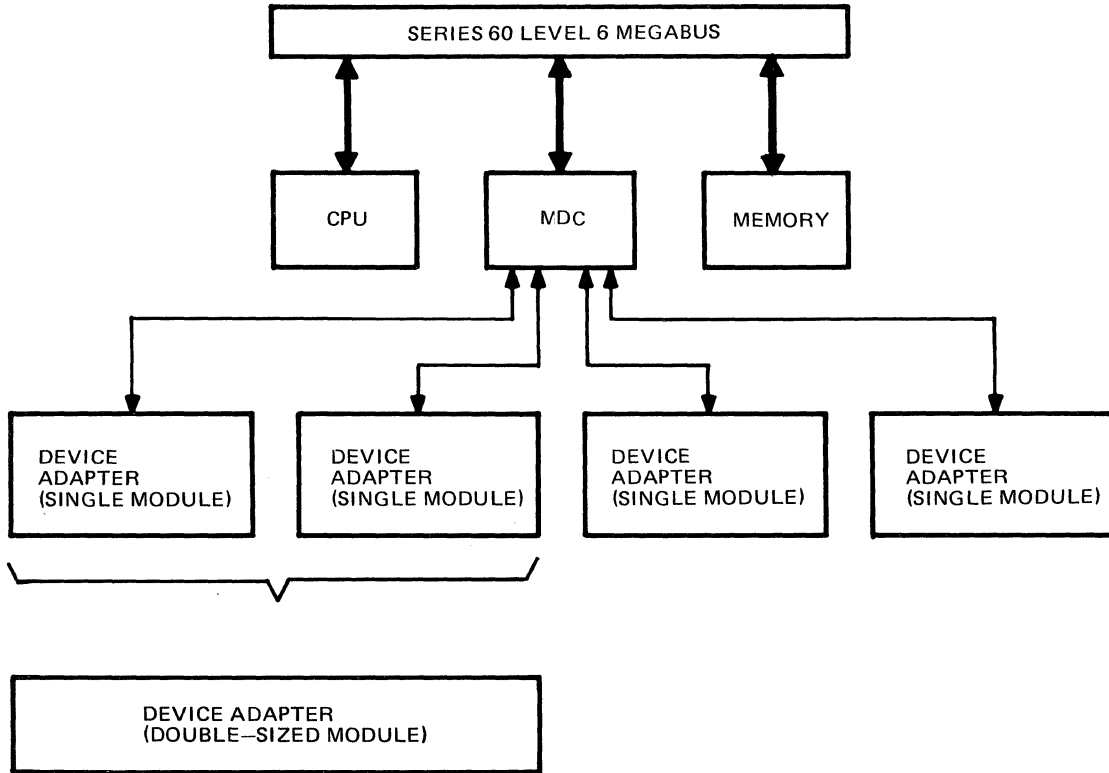


Figure 1-1 MDC System Block Diagram

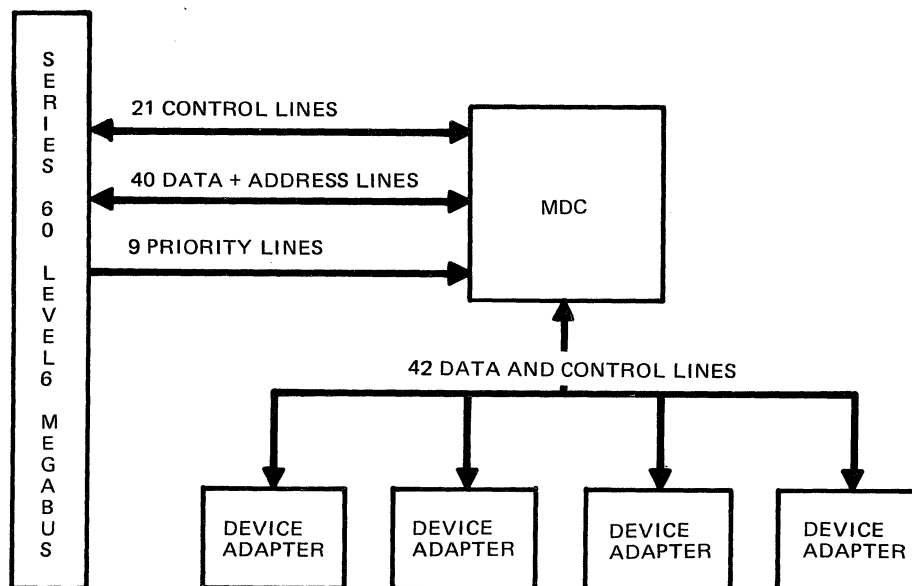


Figure 1-2 MDC Interfaces

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 1-1 Available MDC Device Adapters

TYPE NO.	DESCRIPTION
DIM9101	Diskette Adapter
KCM9101	Console Adapter
CRM9101	Card Reader Adapter
PRM9101	Printer Adapter

Table 1-2 Available MDC Devices

DEVICE	TYPE NO.	DESCRIPTION
Diskette	DIU9101	Single Diskette
Console	DKU9101	CRT (TTY) Keyboard Console 64 characters
	DKU9102	CRT (TTY) Keyboard Console 96 characters
	TTU9101	ASR-33 Teletype Console
	TTU9102	KSR-33 Teletype Console
	TTU9103	ASR-33 With Auto-Shutdown
	TTU9104	KSR-33 With Auto-Shutdown
	TWU9101	30 CPS Keyboard Typewriter Console (KSR) (TTL) w/Keypad
Card Reader ¹	CRU9101	300 CPM
	CRU9102	300 CPM With Mark Sense
	CRU9103	500 CPM
	CRU9104	500 CPM With Mark Sense
Line Printer ²	PRU9103	240 LPM, 96 Characters
	PRU9104	300 LPM, 64 Characters
	PRU9105	480 LPM, 96 Characters
	PRU9106	600 LPM, 64 Characters
Serial Printer	PRU9101	64 Characters
	PRU9102	96 Characters

NOTE

- (1) CRF 9101 = 51-Column Card Option Available
- (2) PRF 9102 = Vertical Format Unit Option Available

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 1-3 Reference Documents

DOCUMENT	DOC. NUMBER	ORDER NO.
Model 6/34/36 Systems Manual	71010200-200	FL35, Rev. 1
Model 6/34/36 CP Manual	71010201-200	FL36, Rev. 1
Type KCM9101 Console Adapter Manual	71010225-200	FL17, Rev. 1
Type PRM9101 Printer Adapter Manual	71010221-200	FL18, Rev. 1
Type CRM9101 Card Reader Adapter Manual	71010222-200	FL13, Rev. 1
Type DIM9101 Diskette Adapter Manual	71010226-200	FL16, Rev. 1
Power System Manual	71010290-200	FL34, Rev. 1
Type MDC9101 Multiple Device Controller Reference Manual (Assy. No. 60127882- 002)	71010370-100	FL26
Type MDC9101 Multiple Device Controller Reference Manual (Assy. No. 60130148- 001)	71010378-100	FM42
Minicomputer Handbook	--	AS22
Peripherals Handbook	--	AT04

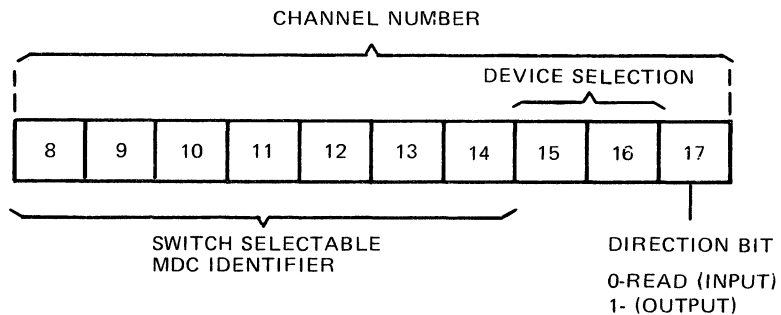


II THEORY OF OPERATION - OVERVIEW

2.1 GENERAL FUNCTIONAL DESCRIPTION (see Figure 1-1)

The multiple device controller (MDC), which accommodates a variable configuration of up to four device adapters is a firmware-operated peripheral device controller. The MDC is capable of controlling two diskette adapters or one diskette adapter and one or two other device adapters, such as a console adapter, a card reader adapter, or a printer adapter.

To perform an operation on a specific device, software must determine the channel to which the device is attached. A channel is a composite of the MDC, the device adapter, and the device. The software addresses the channel by means of a channel number. The following example indicates the format of a typical channel number, as seen on the address lines of the Megabus.



HONEYWELL PROPRIETARY AND CONFIDENTIAL

Each device attached to the MDC by way of an adapter has a specific configuration of device selection bits required to address it.

The MDC contains the firmware and hardware common to all the channels. Device-specific functionality is provided by pluggable device adapters and by firmware routines. The firmware routines are physically located within the microprogram control store on the MDC board.

The MDC multiplexes information to the correct channel and determines priority for channel activity. The highest priority is given to Megabus requests, then to adapter requests, and then to interrupts which are being retried. During normal operation only one channel is active at any given time. When initializing, all adapters are cleared simultaneously.

2.2 SOFTWARE

The MDC operations are a direct result of an input or output instruction from the central processor (CP). Table 2-1 lists the input, output, and diagnostic instructions applicable to the device adapter subsystem. These instructions read or write address, range, and control information to or from a device-specific segment of the scratch pad memory (SPM).

For output instructions (writing into the SPM), the general format of the Megabus address and data lines is as shown in Figure 2-1. The address lines reflect the channel number of the controller being addressed and a function code indicating the purpose of the information on the data lines. The data lines may have control information, address, or control words to be loaded into the MDC scratch pad memory.

For input instructions (reading from the SPM) the general format of the Megabus address and data lines is as shown in Figure 2-2. The address lines reflect the channel number of the controller being addressed and the appropriate function code. The data lines have the number of the channel initiating the command. In response to the input instruction, the MDC loads the bus address lines with the channel number of the unit that is to receive the information on the data lines. The Megabus configuration for an MDC read main memory request is illustrated in Figure 2-3 and an MDC write main memory request is depicted in Figure 2-4. Figures 2-5 through 2-19 show the Megabus configuration for the various I/O type instructions.

When the output I/O load address function code has been sent to the SPM, the state of the direction bit (see Figure 2-1, Output command) is checked in order to determine the data transfer path (to or from the device). For information pertaining to the instructions that each adapter is capable of performing, refer to the appropriate adapter manual.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-1 I/O Commands

TYPE COMMAND	FUNCTION CODE (HEX)	INSTRUCTION	ADAPTER APPLICATION
Output	01	Control Word	Diskette, Console, Card Reader & Printer
	03	Interrupt Control	Diskette, Console, Card Reader & Printer
	07	Task Word	Diskette, Console & Printer
	09	I/O Load Address	Diskette, Console, Card Reader & Printer
	0D	I/O Load Range	Diskette, Console, Card Reader & Printer
	11	Configuration Word A	Diskette, Console, & Card Reader
	13	Configuration Word B	Diskette & Console
Input	02	Interrupt Control	Diskette, Console, Card Reader & Printer
	06	Task Word	Diskette, Console & Printer
	08	Memory Byte Address	Diskette, Console, Card Reader & Printer
	0A	Memory Module Address	Diskette, Console, Card Reader & Printer
	0C	Range	Diskette, Console, Card Reader & Printer
	10	Configuration Word A	Diskette, Card Reader & Console
	12	Configuration Word B	Diskette & Console
	18	Status Word 1	Diskette, Console, Card Reader & Printer
	26	Identification Code	Diskette, Console, Card Reader & Printer

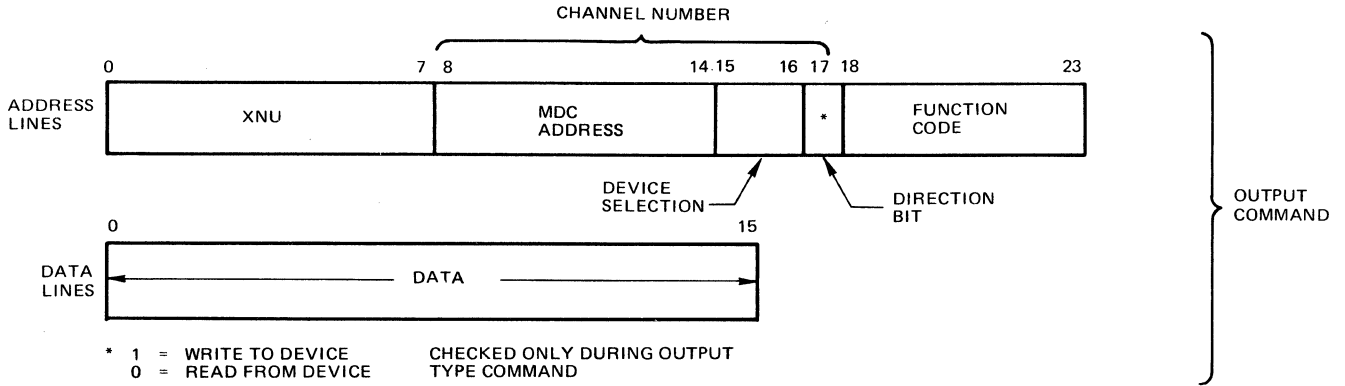


Figure 2-1 Megabus Configuration for I/O Output Command

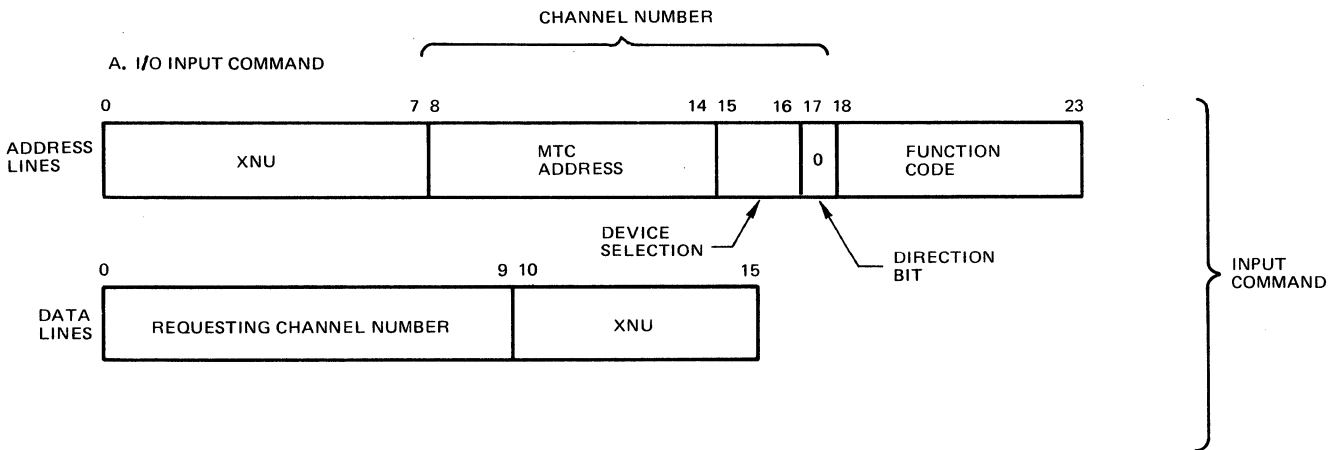


Figure 2-2 Megabus Configuration for I/O Input Command and MDC Response (Sheet 1 of 2)

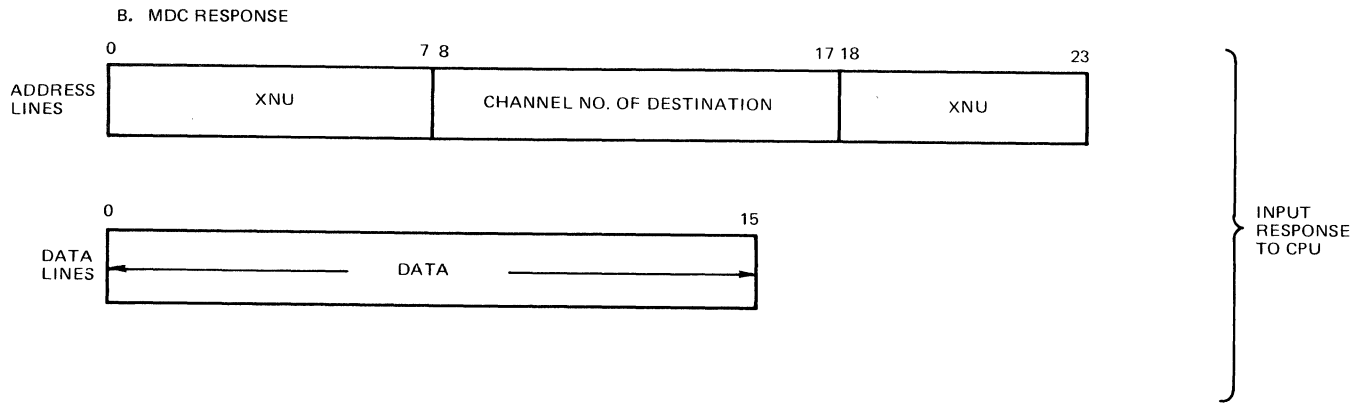


Figure 2-2 Megabus Configuration for I/O Input Command and MDC Response (Sheet 2 of 2)

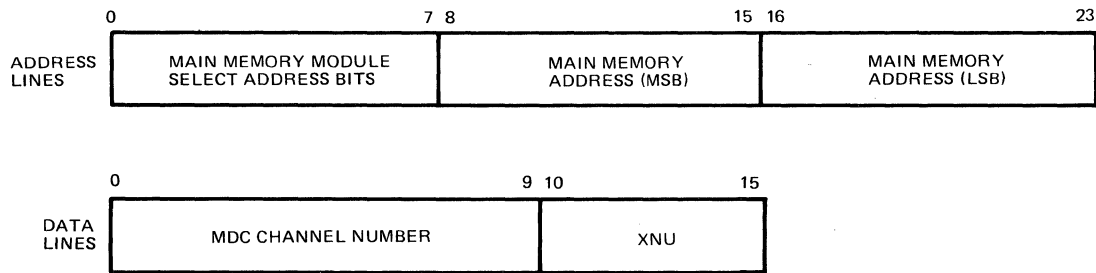


Figure 2-3 Megabus Configuration for MDC Read Main Memory Request

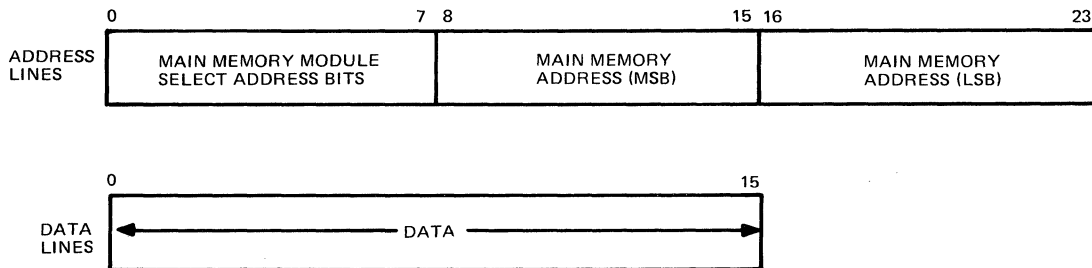


Figure 2-4 Megabus Configuration for MDC Write Main Memory Request

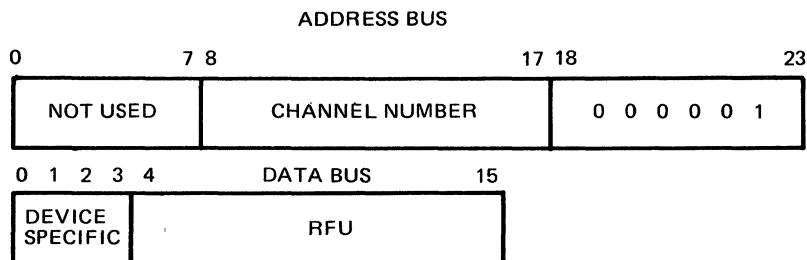


Figure 2-5 Output Control Word

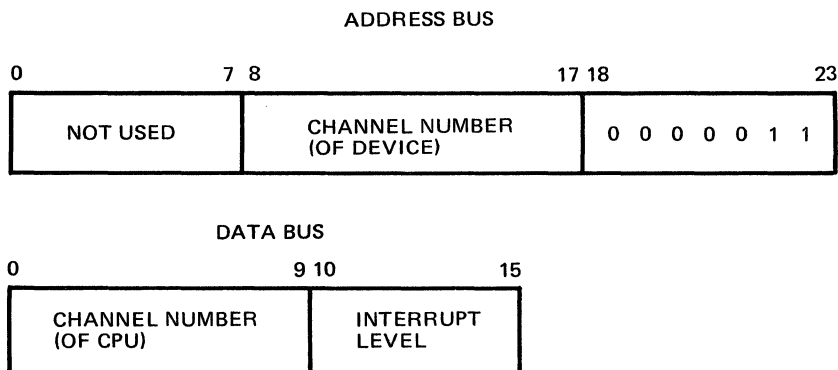


Figure 2-6 Output Interrupt Control

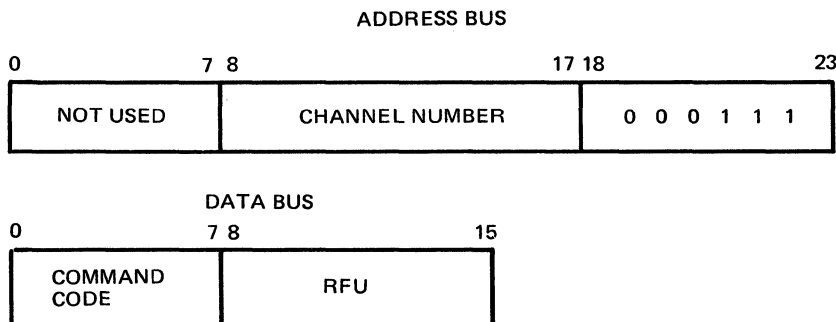


Figure 2-7 Output Task Word

HONEYWELL PROPRIETARY AND CONFIDENTIAL

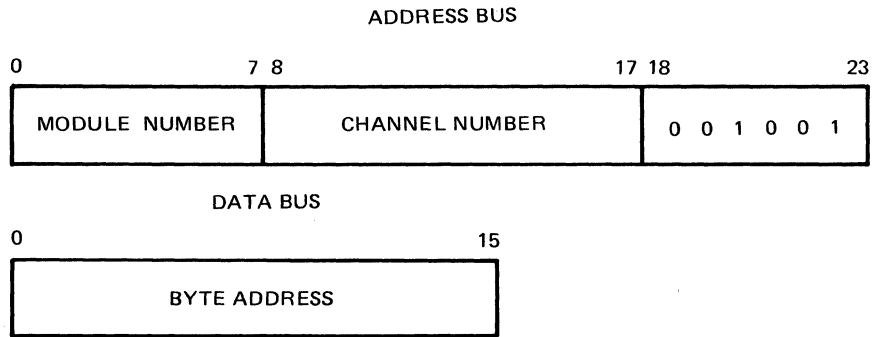


Figure 2-8 I/O Load Output Address

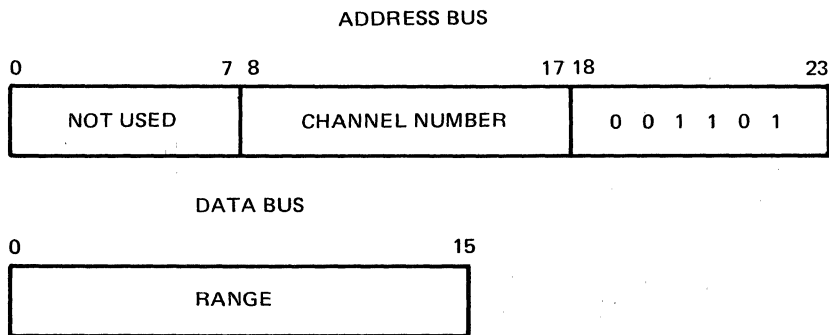


Figure 2-9 I/O Load Output Range

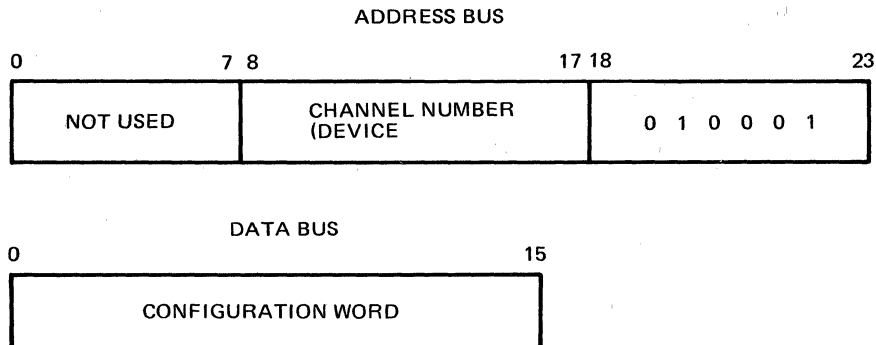


Figure 2-10 Output Configuration Word

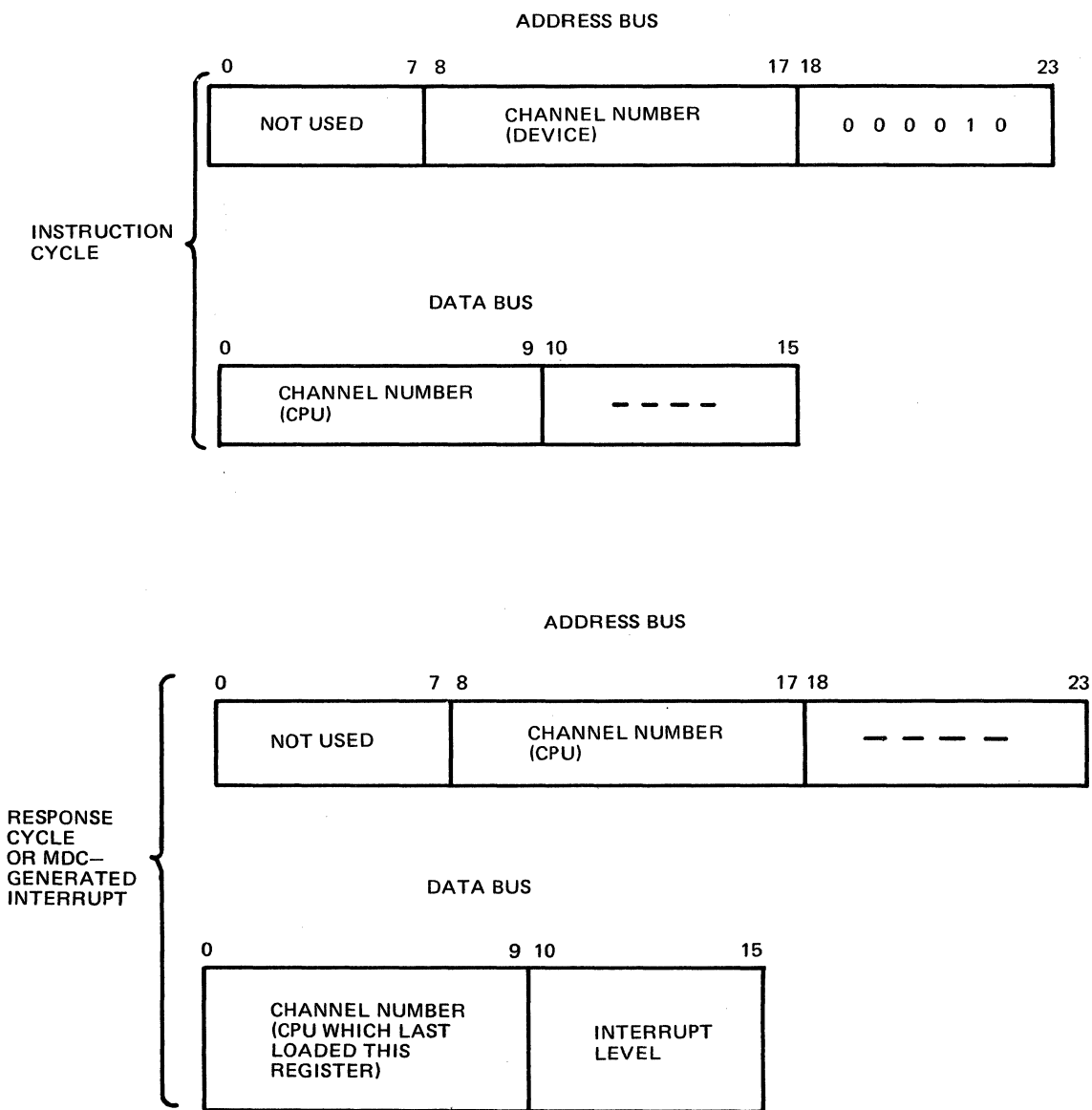


Figure 2-11 Input Interrupt Control

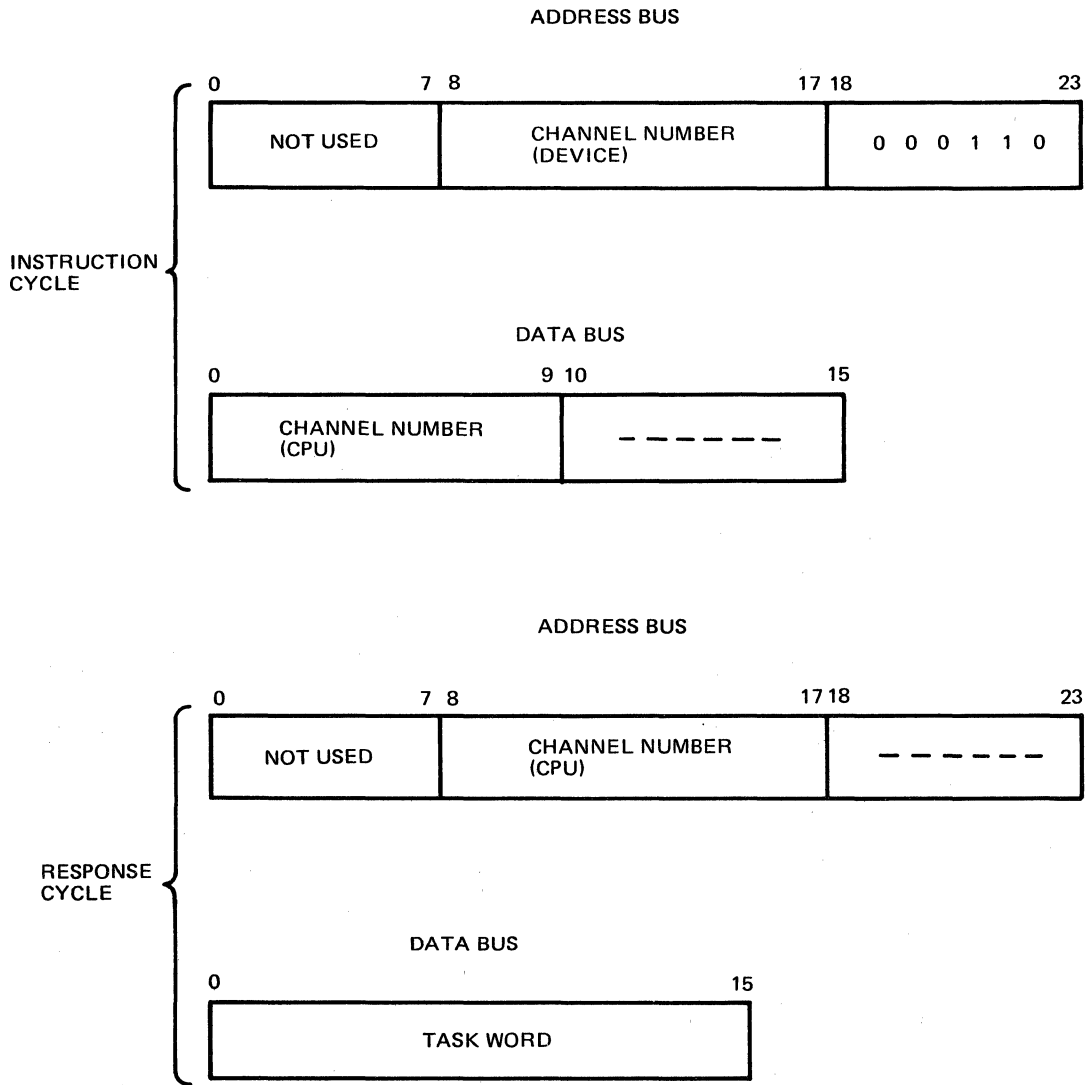


Figure 2-12 Input Task Word

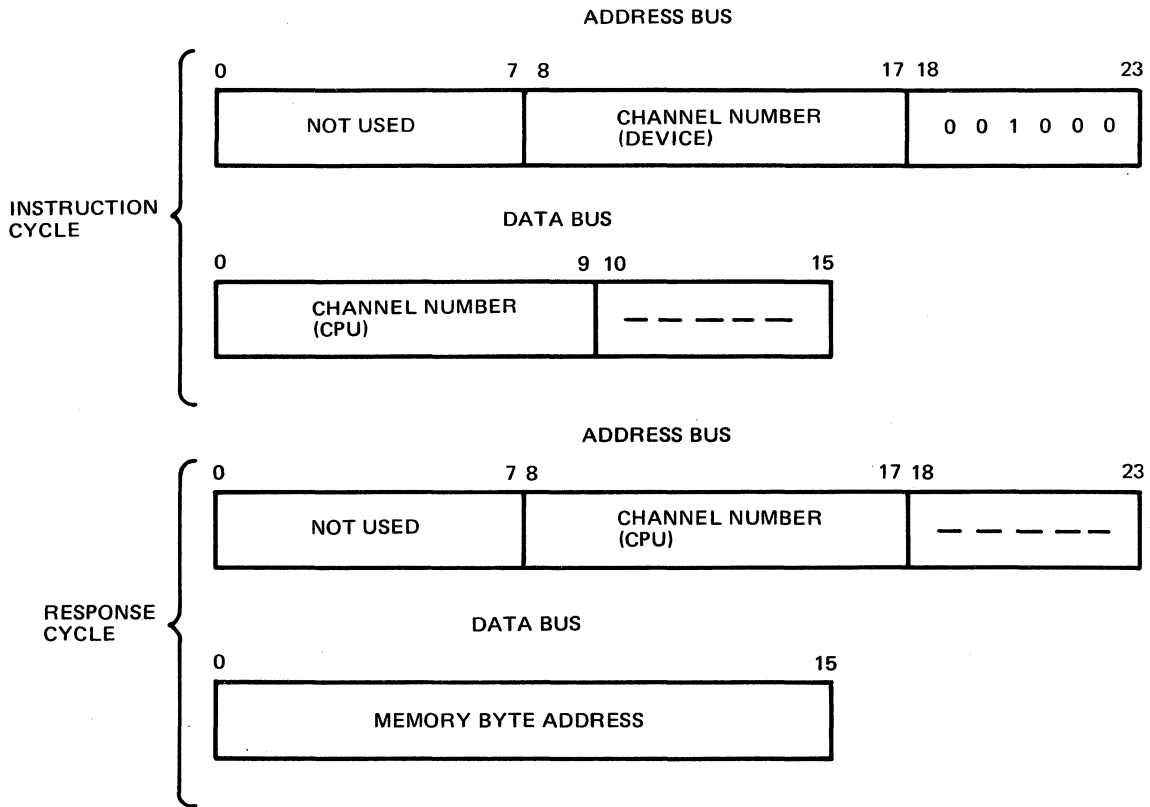


Figure 2-13 Input Memory Byte Address

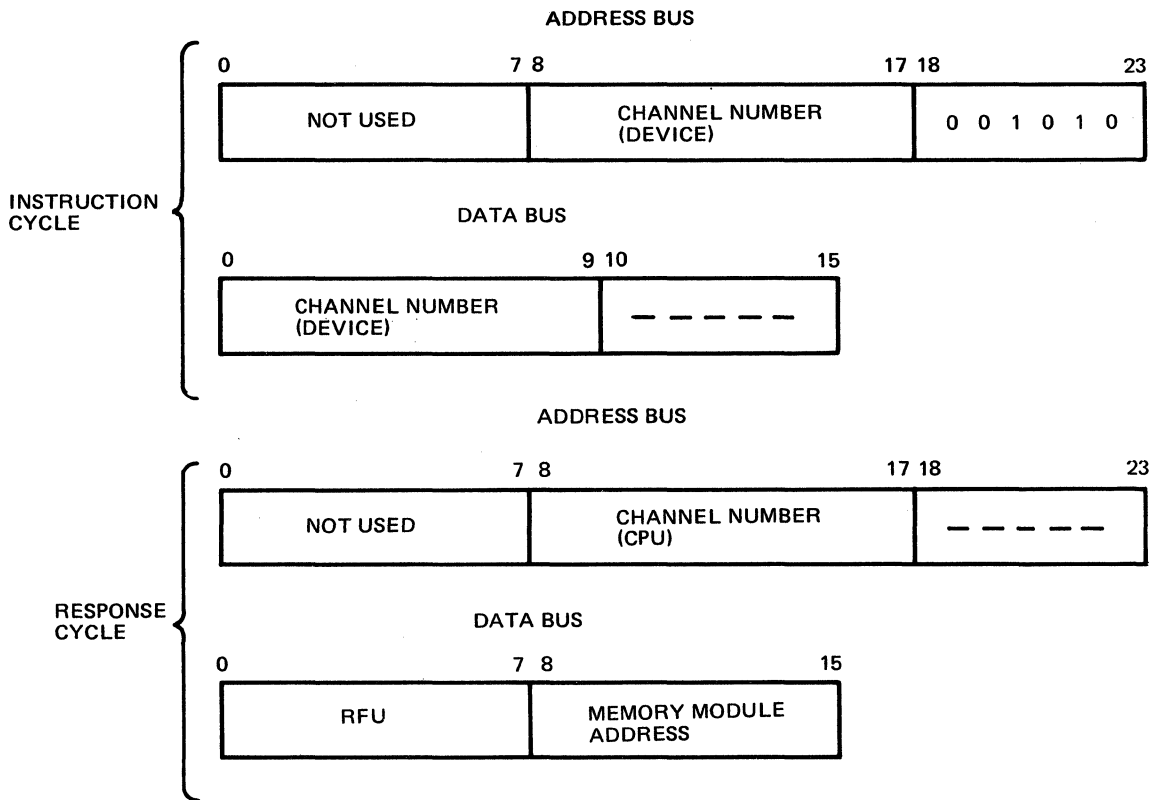


Figure 2-14 Input Memory Module Address

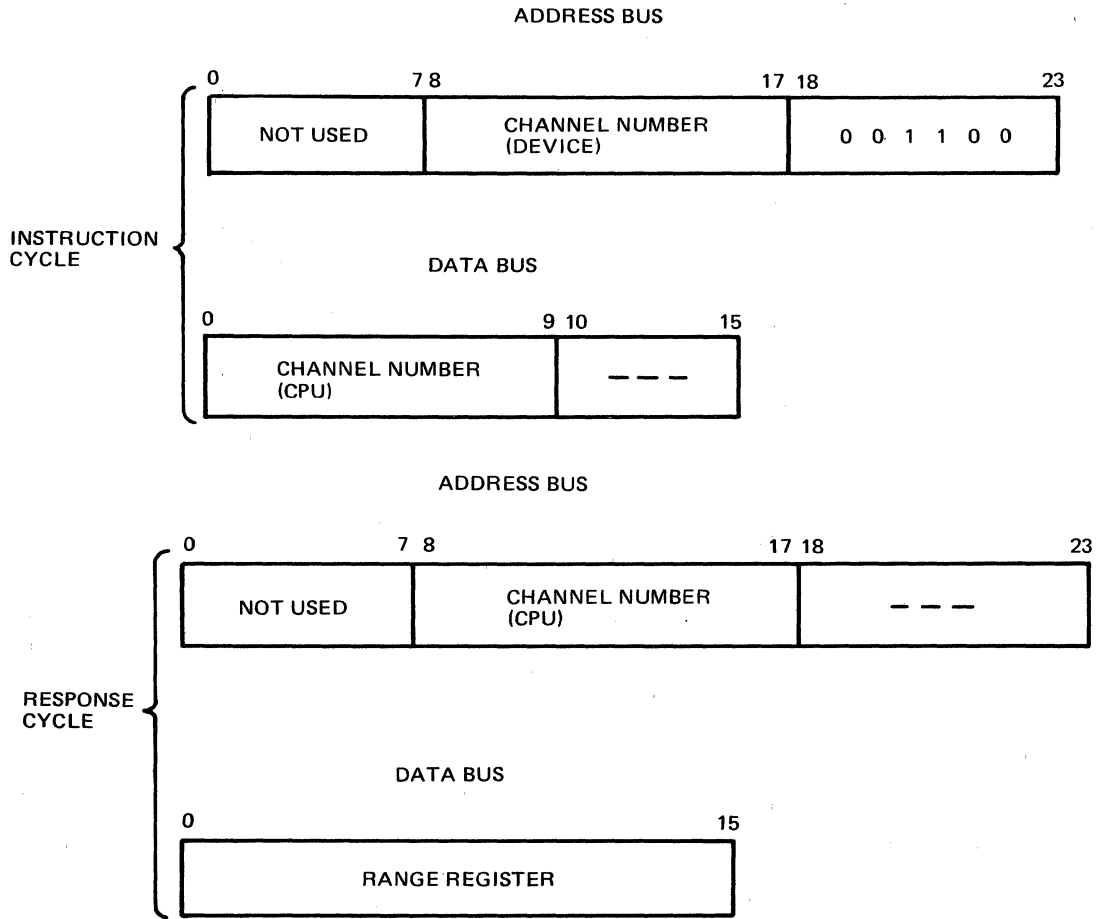


Figure 2-15 Input Range

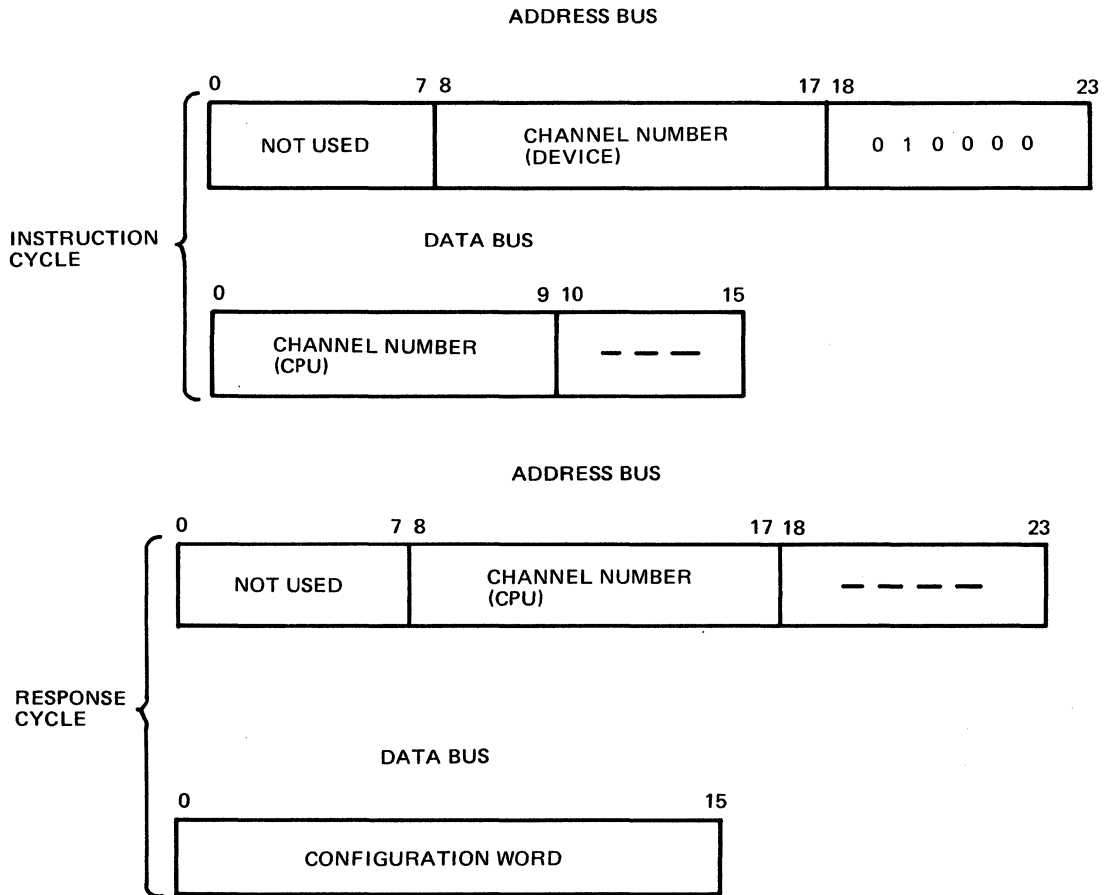


Figure 2-16 Input Configuration Word

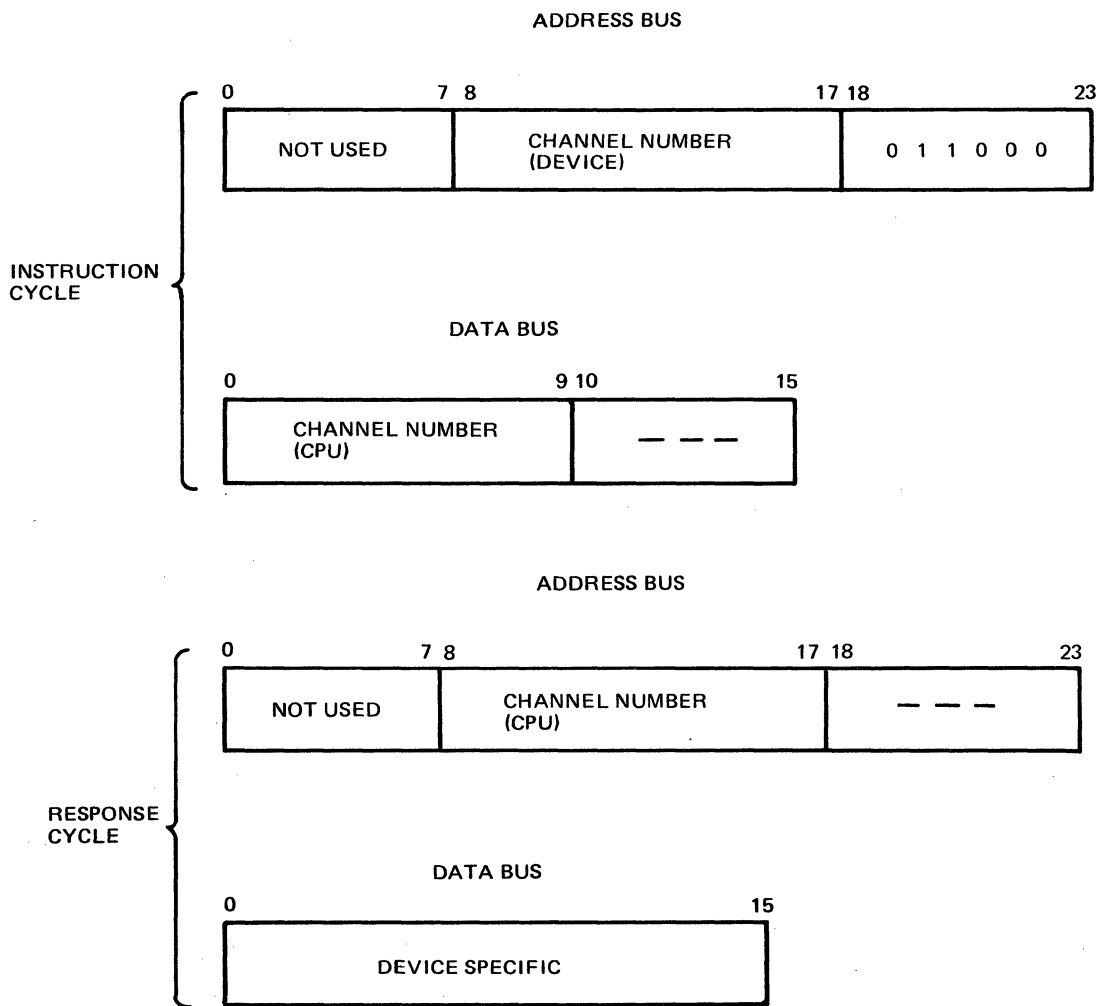


Figure 2-17 Input Status Word 1

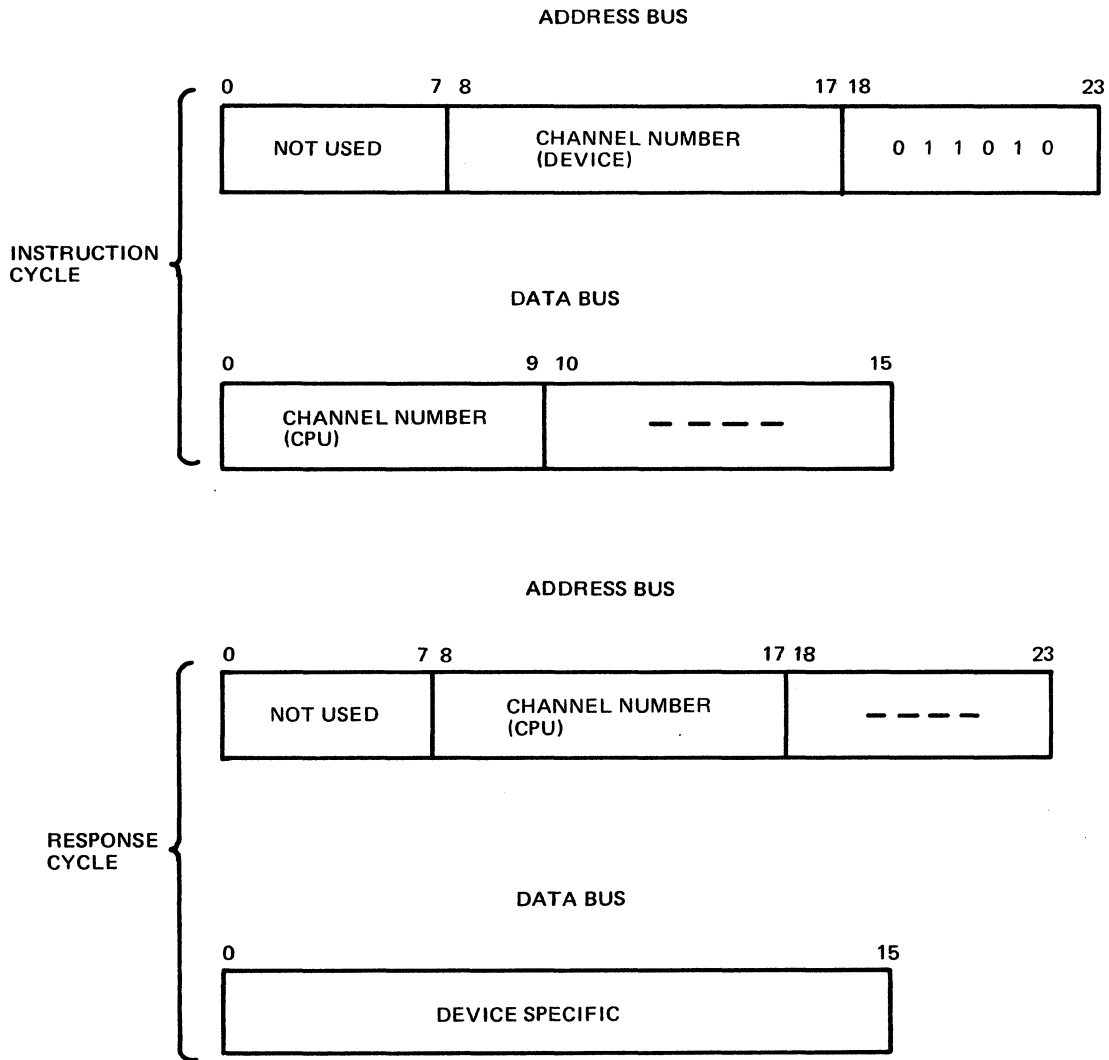


Figure 2-18 Input Status Word 2

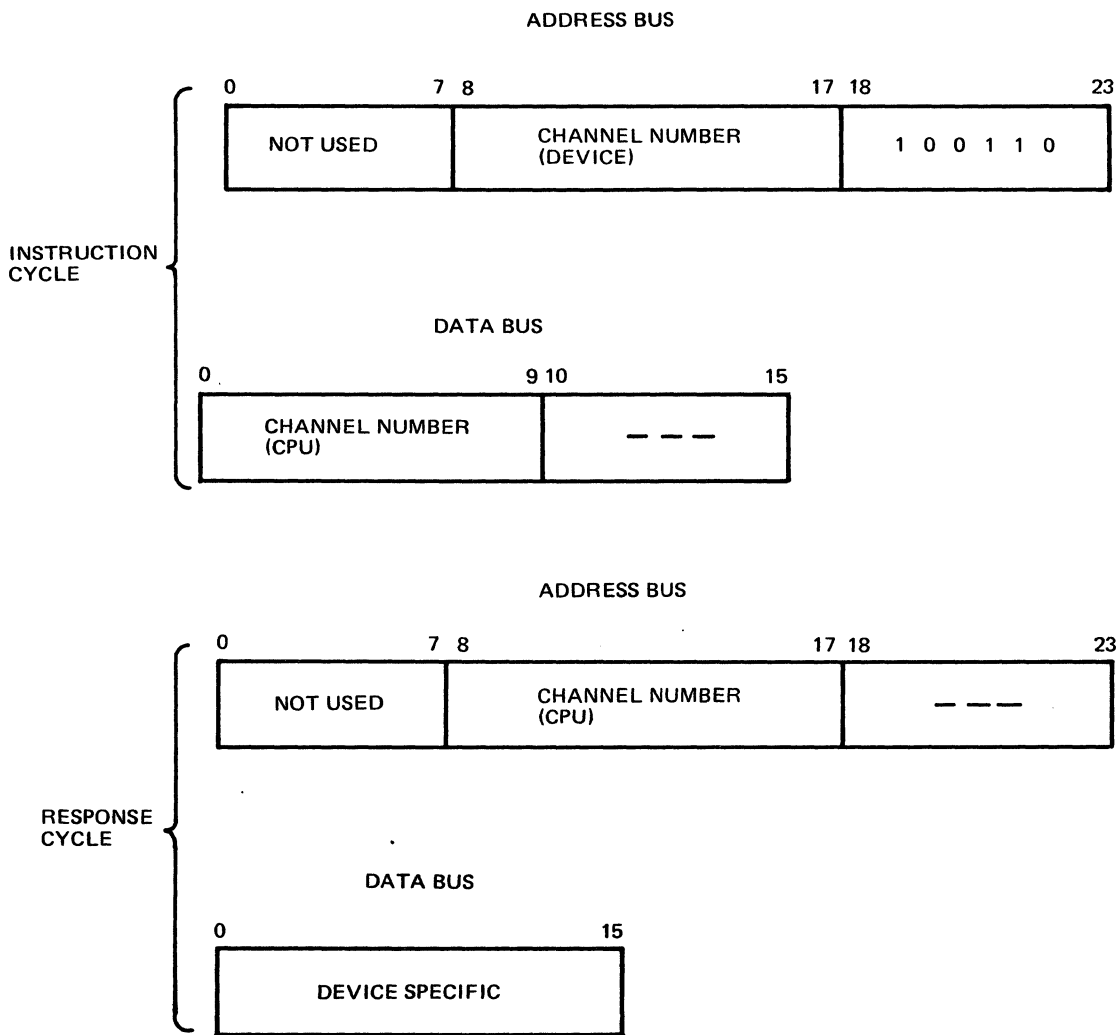


Figure 2-19 Input Identification Code

2.3 FIRMWARE

Firmware, a sequence of microinstructions resident in the microprogram control store, is the primary control element of the MDC. The main function of the firmware is to interpret external and internal events or conditions and to react in a prescribed manner (i.e., setting or resetting of hardware functions). Efficient data transfer is also a result of firmware control of hardware components in the data path.

Each time the MDC is cleared, a series of firmware operations called the Quality Logic Test is performed. These operations are used to test whether the MDC hardware is functioning properly. Upon successful completion of the Quality Logic Test, the firmware proceeds to set up the hardware for execution of software instructions. This is accomplished by setting up control information in the read/write memory and enabling the available adapters. After the adapters are enabled, the firmware enters a routine and waits for a request from the Megabus or from one of the adapters. When a request occurs, the firmware analyzes status and priority before proceeding to the appropriate firmware operation for processing the request. (Refer to Section IV for detailed information.)

2.4 HARDWARE

The MDC hardware (see Figure 2-20) is organized into seven logic areas: microprogram control store, arithmetic logic unit and accumulator, scratch-pad memory, Megabus logic, adapter logic, test multiplexer, and clock logic. The descriptions in the following subsections are of an overview nature, and all pertain to the interfaces and logic blocks depicted in Figure 2-20.

2.4.1 Megabus/MDC Interface

The MDC/Level 6 Megabus interface is the control and transfer link between the MDC and any other unit in the system. It provides a path for address, data, and control information. This interface also supplies the paths for determining priority of a request from any attached controller. Figure 2-21 identifies all the interface lines, indicating direction, usage, and mnemonic of each. Table 2-2 describes each signal line used by the Megabus/MDC interface.

2.4.2 MDC/Adapter Interface

A diagram of the MDC/adapter interface interconnection is shown in Figure 2-22. This figure identifies the interface lines, their direction, mnemonics, and usage. It depicts the lines as seen by the MDC (not the adapter, where the mnemonics for many of the lines differ). Table 2-3 describes each signal line used by the MDC/Adapter Interface. For a cross-reference of signal names, refer to Table 2-4.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

This interface provides a common link with all four adapters and gives the firmware the capability of selecting the proper adapter and performing the designated operation. It provides the paths necessary to supply the adapters with data, control, and timing pulses; it also supplies the MDC with requests and data from the adapter.

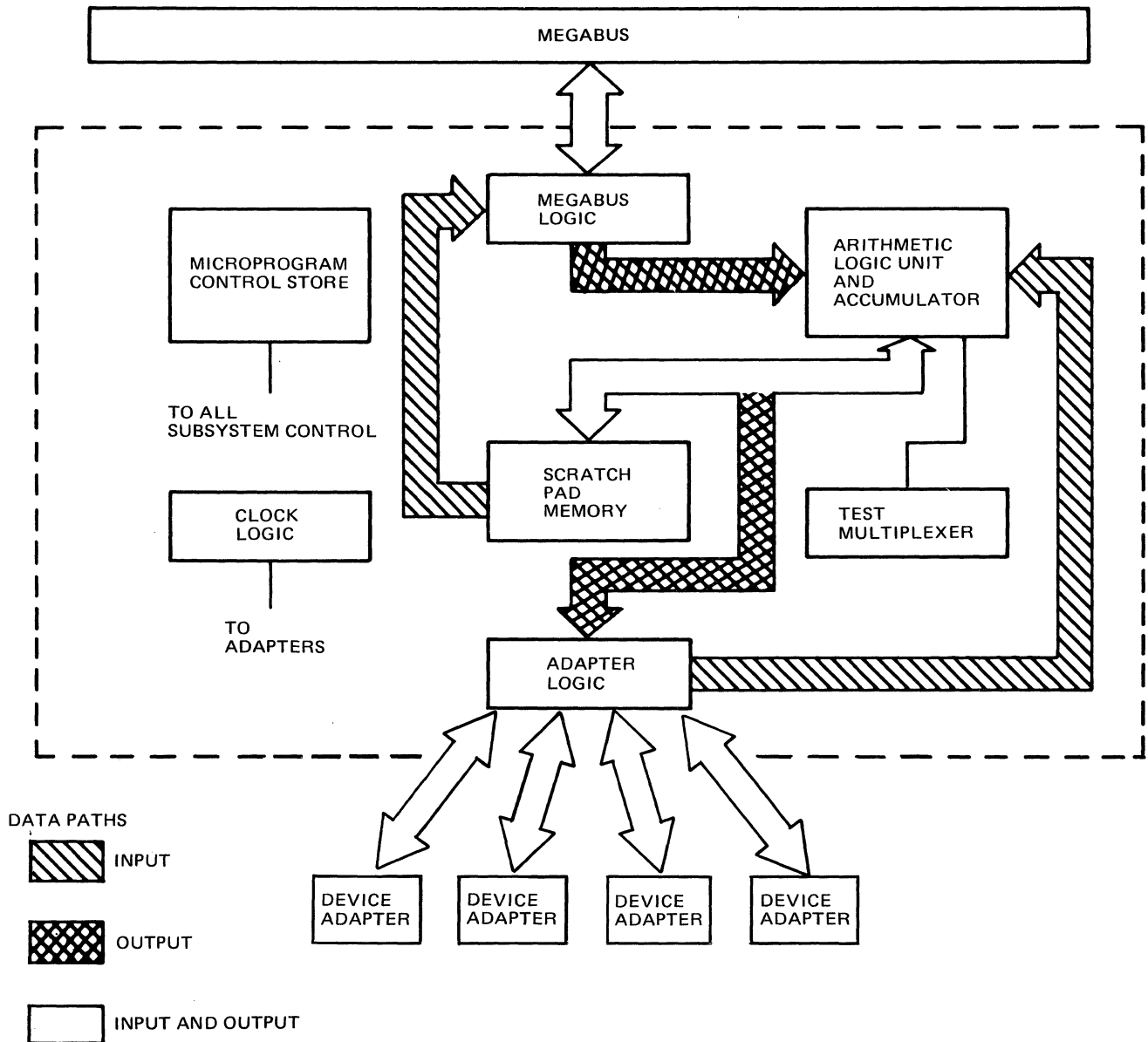
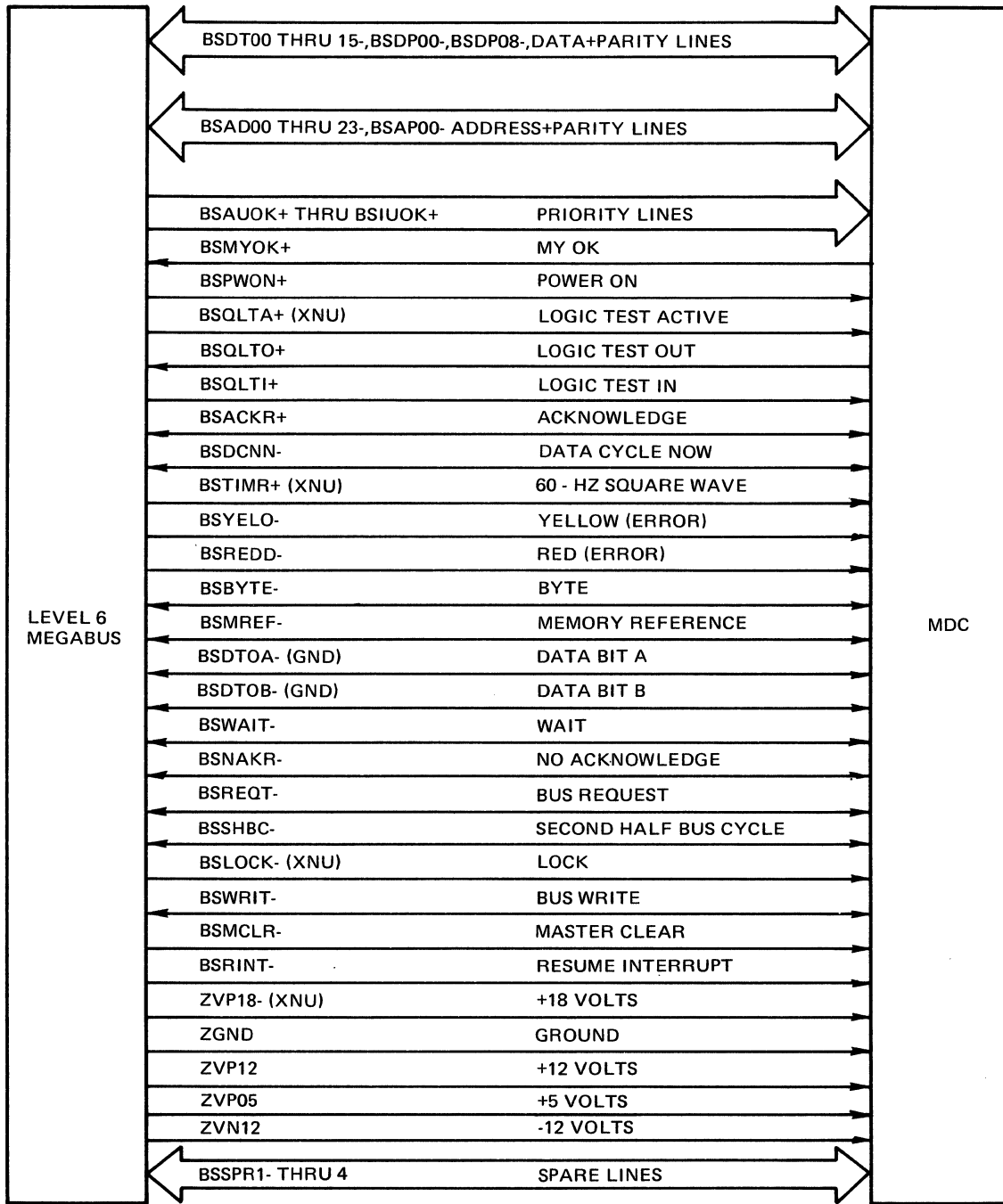


Figure 2-20 MDC Hardware Major Block Diagram



NOTE: XNU = NOT USED

Figure 2-21 Megabus/MDC Interface

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-2 Megabus/MDC Interface Signal Lines
(Sheet 1 of 2)

TERM/MNEMONIC	DESCRIPTION
Data Bits 0 through 7 (BSDT00- → 07-)	These lines represent the most significant byte of data.
Data Bits 8 through 15 (BSDT08- → 15-)	These lines represent the least significant byte of data.
Data Parity-Left Byte (BSPD00-)	This signal contains odd parity for data bits 0 through 7.
Data Parity-Right Byte (BSPD08-)	This signal contains odd parity for data bits 8 through 15.
Address Bus Bits 0 through 23 (BSAD00- → 23-)	These lines contain an address to be used by memory or by a controller or central processor.
Address Parity (BSAP00-)	This signal contains odd parity for the most significant byte of the address bus, bits 0 through 7.
Priority Lines (BSAUOK+ through BSIUOK+)	These lines are used to establish priority of the units attached to the bus.
My OK (BSMYOK+)	This signal indicates the unit that is presently using the bus.
Power On (BSPWON+)	This signal is true when all power supplies in the system are operating correctly.
Logic Test In (BSQLTI+)	This signal initiates the Internal Logic Test in a unit attached to the bus.
Logic Test Out (BSQLTO+)	This signal indicates the unit has successfully completed running its Internal Logic Test and is used as the logic-test-in signal for the next unit attached to the bus.
Acknowledge (BSACKR+)	This signal indicates that the information on the bus has been accepted.
Data Cycle Now (BSDCNN-)	This signal indicates that the information on the bus is valid.
Yellow (BSYELO-)	This signal indicates that the accompanying transferred information is correct but that a correction operation was performed.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-2 Megabus/MDC Interface Signal Lines
(Sheet 2 of 2)

TERM/MNEMONIC	DESCRIPTION
Red (BSREDD-)	This signal indicates that the accompanying transferred information is in error.
Byte (BSBYTE-)	This signal indicates that the current transfer is a byte transfer rather than a word transfer.
Memory Reference (BSMREF-)	This signal indicates that the address leads contain a memory address.
Wait (BSWAIT-)	This signal indicates that the transfer will be accepted when the bus data register is available.
No Acknowledge (BSNAKR-)	This signal indicates that the information on the bus has been refused.
Bus Request (BSREQT-)	This signal indicates that one or more units on the bus have requested a bus cycle.
Second Half Bus Cycle (BSSHBC-)	This signal identifies the second bus cycle in response to a memory read request.
Bus Write (BSWRIT-)	This signal indicates that information on the bus is ready to be transferred.
Master Clear (BSMCLR-)	This signal initializes the units attached to the bus.
Resume Interrupt (BSRINT-)	This signal is a 200-nanosecond pulse which is issued by the central processor when it is capable of receiving interrupts again.

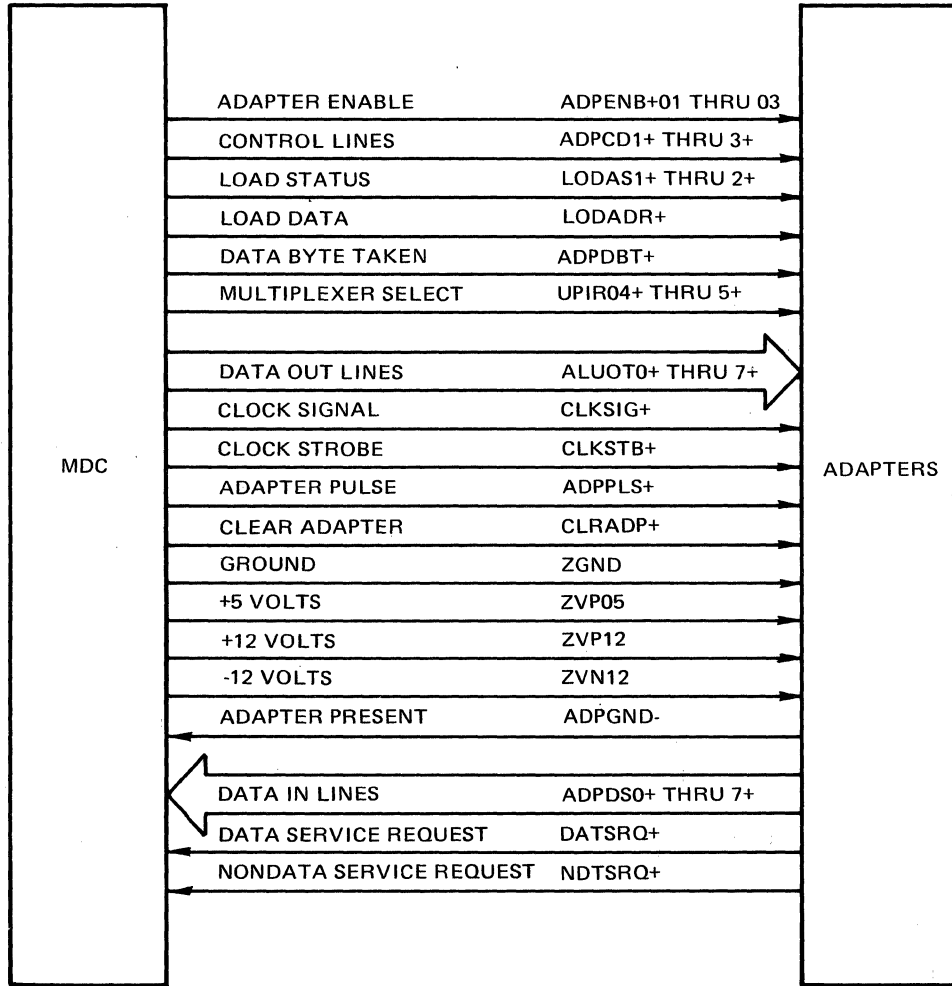


Figure 2-22 MDC/Adapters Interface

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-3 MDC/Adapter Interface Signal Lines

TERM/MNEMONIC	DESCRIPTION
Adapter Enable (ADPENB+01 through +04)	The diskette adapter uses two of the four lines as device selection bits. The other device adapters use these lines as enables.
Control Lines (ADPCD1+ through 3+)	These signals are used for adapter-specific operations.
Load Status (LODAS1+ through 2+)	These signals are used to load data in the device adapter's status registers.
Load Data (LODADR+)	This signal is used to load data in the device adapter's data register.
Data Byte Taken (ADPDBT+)	This signal notifies the device adapter that the contents of its data register have been stored.
Multiplexer Select (UPIR04+ through 05+)	These signals are the selection bits for the device adapter's input multiplexer.
Data Out Lines (ALUOT0+ through 7+)	These lines from the MDC to the device adapter represent the information to be used by the adapter.
Clock Signal (CLKSIG+)	A clock function from the MDC to the device adapter.
Clock Strobe (CLKSTB+)	A clock function from the MDC to the device adapter.
Clear Adapter (CLRADP+)	This signal clears the major logic areas of the adapter.
Adapter Present (ADPGND-)	This signal notifies the MDC that an adapter is present by indicating a ground connection.
Data In Lines (ADPDS0+ → 7+)	These lines from the device adapter to the MDC represent the information to be used by the MDC.
Data Service Request (DATSRQ+)	This signal notifies the MDC that the device adapter has a data byte ready or that a data byte is required.
Nondata Service Request (NDTSRQ+)	This signal notifies the MDC that the device adapter has a nondata service request to be processed (e.g., a change in device state).
Adapter Pulse (ADPPLS+)	This signal is used only by the printer device adapter to generate a strobe for the printer.

Table 2-4 MDC/Adapter Interface Line and Mnemonic Cross-Reference List

MDC		ADAPTER			
		DISKETTE	READER	CONSOLE	PRINTER
FUNCTION	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC	MNEMONIC
Adapter Enable Control Lines ↓	ADPENB+01→+04 ADPCD1+ ADPCD2+ ADPCD3+	ADPEN+ /ADPENB+ ADPCD1+ ADPCD2+ ADPCD3+	ADPENX XNU XNU XNU	ADPENX XNU XNU XNU	ADPENX XNU XNU XNU
Load Status 1	LODAS1+	LODAS1+	ADPCN1+	ADPCN1+	ADPCN1+
Load Status 2	LODAS2+	LODAS2+	ADPCN2+	ADPCN2+	ADPCN2+
Load Adapter Data	LODADR+	LODADR+	XNU	ADPDAT+	ADPDAT+
Data Byte Taken	ADPDBT+	ADPDBT+	ADPDBT+	ADPDBT+	XNU
Data Out	ALUOT0→+7+	ALUOT0→+7+	ALUOT0→+7+	ALUOT0→+7+	ALUOT0→+7+
Multiplexer Select ↓	UPIR04+ UPIR05+	ADPMS0+ ADPMS1+	ADPIC0+ ADPIC1+	ADPIC0+ ADPIC1+	ADPIC0+ ADPIC1+
Data Request	DATSRQ+	DATSRQ+	DATSRQ+	DATSRQ+	DATSRQ+
Nondata Request	NDTSRQ+	NDTSRQ+	NDTSRQ+	NDTSRQ+	NDTSRQ+
Data In	ADPDS0→+7+	ADPDS0→+7+	INFOI0→+7+	INFOI0→+7+	INFOI0→+7+
Clock Signal	CLKSIG+	CLKSIG+	XNU	CLKSIG+	XNU
Clock Strobe	CLKSTB+	CLKSTB+	CLKSTB+	CLKSTB+	CLKSTB+
Adapter Pulse	ADPPLS+	XNU	XNU	XNU	ADPPLS+
Adapter Present	ADPGND-	ADPGND-	ZGND	ZGND	ZGND
Clear Adapter	CLRADP+	ADPCLR+	ADPCLR+	ADPCLR+	ADPCLR+
Ground	ZGND	ZGND	ZGND	ZGND	ZGND
+5 Volts	ZVP05	ZVP05	ZVP05	ZVP05	ZVP05
-12 Volts	ZVN12	ZVN12	ZVN12	ZVN12	ZVN12
+12 Volts	ZVP12	ZVP12	ZVP12	ZVP12	ZVP12

Note: XNU = Not used

2.4.3 Microprogram Control Store (Figure 2-20)

The Microprogram Control Store (UPCS) provides permanent storage for resident control firmware and diagnostic microprograms. The UPCS is enabled during normal operation, when the MDC is not in test mode. It utilizes its ability to vary address sequencing in order to execute appropriate routines, depending on priority information, test conditions, or channel activity.

The UPCS is configured of either 2K or 4K PROM chips. The internal array of each 2K PROM chip is 512 locations by 4 bits wide, providing a maximum of 2,048 bits. The 2K PROM chips are aligned in rows of 4 to create a UPCS output 16 bits wide. The UPCS can be expanded to a maximum of 2K word locations.

The internal array of each 4K PROM chip is 1,024 locations by 4 bits wide, providing a maximum of 4,096 bits. The 4K PROM chips are aligned in rows of 4 to establish a UPCS output 16 bits wide. The UPCS is expandable to a maximum of 4K word locations.

NOTE

The hex rotary switch on the controller is set to 0 for 2K PROM chips or to F for 4K PROM chips.

2.4.4 Arithmetic Logic Unit (ALU)/Accumulator (ACU)
(Figure 2-20)

The ALU is the focal point of all data operations within the MDC, between the MDC and the device adapter(s), and between the MDC and the Megabus. Two input multiplexers (A-operand and B-operand) to the ALU allow various combinations of registers to be used as operands within the ALU.

The ALU performs eight-bit arithmetic and logic operations on the outputs of the two multiplexers and determines the destination of the result as a function of the firmware. ALU status is generated as a result of an ALU operation and is stored in ALU status flip-flops for interrogation or until the initiation of the subsequent ALU firmware command.

The ALU also performs bit operations on one input with a data constant supplied by firmware. The constant can be loaded into the ALU by way of the B-operand multiplexer (MUX).

Through the use of multiple byte procedures, word mode operations are performed by firmware and the ALU. In a multiple byte procedure, ALU status (i.e., carry-out) is set each time a byte is operated on, and this status is used to determine the operation required for the subsequent byte. The ALU is capable of performing a left shift operation.

The ACU is an eight-bit register used for temporary storage of the ALU outputs. The ACU outputs are fed to the A-operand multiplexer for distribution of information throughout the MDC. During data transfers, the ACU outputs are fed to the A-operand multiplexer as part of the data path to and from the device adapters.

2.4.5 Scratch-Pad Memory (SPM) (Figure 2-20)

The SPM is a 256-location-by-8-bit-deep read/write memory which is used for storage of information that is required by or generated by each channel (i.e., data, status, commands, etc.) The memory is segmented into four parts (64 locations each) with one part designated for each channel. The SPM is addressed by eight bits; the two high-order bits select the correct segment for the channel which is active, and the remaining six bits select a relative location within the segment.

Data from the ALU operand mux is written into the SPM during a firmware Memory Write command at the relative location specified by the address bits. Because the SPM uses the ALU operand multiplexer as the data input, all firmware-visible registers can be implemented as an input local register. The data out of the SPM is delivered to either one of the ALU operand multiplexers and can be distributed throughout the MDC from there.

2.4.6 Megabus Logic (Figure 2-20)

The Megabus logic provides an interconnection between the Megabus and each channel on the MDC. Although some of the logic elements are dedicated to an individual channel, the Megabus hardware is common to all the channels.

When Megabus cycles designated for the MDC are detected by a Megabus logic decoder, the response logic is enabled. If the channel being addressed has an adapter installed, a response is generated; otherwise, no response is made, indicating the absence of a device on the desired channel. If the channel is available, the response causes the information on the Megabus address and data lines to be stored. The states of certain control lines are also stored, and the Megabus logic goes busy to inhibit further transfers to the MDC until firmware has dispensed with the stored information.

When the MDC requests a Megabus transfer, firmware loads the address and data registers and sets the proper control lines prior to Megabus cycle initiation. The response of the slave unit is stored by the Megabus logic, and the Megabus logic remains busy until firmware detects completion of the cycle.

Control signals for synchronization of the MDC Megabus requests (and Megabus requests by other units) are also located in the Megabus logic. Finally, the Megabus logic determines the MDC priority with reference to other units when there are simultaneous requests for Megabus activity.

2.4.7 Adapter/Channel Control Logic (Figure 2-20)

A channel consists of the MDC, a device adapter, and a device. Complete channel control results from a combination of hardware and firmware.

Each channel can provide two types of service requests: data or nondata. These service requests are supplied to a channel request encoder where the presence of a data service request signifies

that a data byte read from the device is available or that a data byte must be transferred to the device. The nondata service request indicates a change of device state or the presence of a device condition which requires attention (e.g., head of form on the printer).

The request encoder indicates when a channel request is active. It also sets up the priorities so that any Megabus request has priority over a channel data service request. A similar request from channel 0 would have higher priority than that from any of the higher channel numbers.

Firmware has the ability to test the output of the priority encoder to determine if a request has occurred. It also can test the encoder to ascertain if the request with the highest priority is a Megabus request or a channel request.

The priority encoder has two outputs which are a binary function of the highest priority channel. These lines are used to enable a single adapter at a time corresponding to the appropriate channel. One exception is when Master Clear is active: all the adapters are enabled simultaneously to receive the Master Clear signal.

2.4.8 Test Multiplexer (Figure 2-20)

Subsystem conditions, including status, errors, and register bits, are visible to firmware for examination via the test multiplexer. Firmware operations which require the bypassing of a micro-operation specify a signal and its condition for which a skip should occur. The test multiplexer provides the facilities for comparing the state of the specified signal with the state specified by the firmware. If the two conditions are equal, the instruction register is cleared for one cycle.

2.4.9 Clock Logic (Figure 2-20)

The MDC divides the output of an 8-MHz oscillator to derive a 4-MHz clock cycle. The MDC is supplied with both the assertion and negation of the clock, each having a 250-nanosecond cycle time. Both signals are utilized by the MDC to implement logic in the first 125-nanosecond period and the second 125-nanosecond period of a single clock cycle. A third type of clock pulse is utilized by the MDC as a strobe pulse for the SPM, the adapters, and various other MDC registers. This strobe occurs during the latter part of the second 125-nanosecond period of a single clock cycle and is only 35 to 55 nanoseconds in duration.

2.5 OUTPUT DATA OPERATION

The following subsections describe the MDC hardware and firmware implemented when an output data operation is to be performed on a selected channel. Discussion centers around the data transfer path, as well as the firmware control, timing, and data verification checks required to execute the output data operation.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Two functional components are required to complete an output data operation: software initiation and firmware control through hardware implementation.

2.5.1 Software Initiation

Software initiates the output data operation by using the output function codes listed in Table 2-1 and the bus formats illustrated in Figures 2-5 through 2-10. This is accomplished by sending control information from the central processor (CP) to the MDC for storage in the SPM: control information, the configuration words (FC = 11 hex), the memory address (FC = 09 hex), and the range (FC = 0D hex).

At this point, software involvement is terminated if the addressed device is capable of only one functional operation (i.e., printer). If the addressed device is capable of performing multiple functional operations (i.e., diskette), software is required to make an additional transfer, the task word (FC = 07 hex), to the MDC for storage in the SPM.

After the last software transfer is completed, the remaining control of the output data operation is maintained by the MDC firmware.

2.5.2 Firmware Control Through Hardware Implementation (see Figure 2-20)

Firmware is made aware that an output data operation is to be performed when a bus request is detected which is outputting control information. The control information on the data lines as a result of the bus request from the CP is stored in the SPM in the locations specified by the function code.

With software initiation of the output data operation, either 4 or 5 bus transfers to the MDC are executed (refer to subsection 2.5.1). The control information transferred across the bus is the control word, the configuration words, the memory address, the range and, if necessary, the task word. As each transfer occurs, the firmware stores the control information into the SPM of the MDC. Control of the remaining portion of the output data operation (the data transfer) is maintained by the firmware.

The firmware performs all the output data operations (write or print), status verification, valid channel, range determination, and device capabilities. The task is initiated by firmware's loading of the adapter's control registers with the control information previously stored in the SPM as a result of the software bus request. The adapter will then generate requests for data and the MDC firmware accesses memory, receives the data and transfers the data to the adapter using the hardware path shown in Figure 2-20. All data integrity checks performed on the information sent to the adapters is accomplished by the adapter hardware or the device specific firmware routines.

After detection of the data transfer termination (end of range), the firmware reports the operation's status to the software by generating a bus transfer. Using the reported status, software then establishes if the operation was successful. Retries of unsuccessful operations are performed according to software parameters.

2.6 INPUT DATA OPERATION

Software and firmware involvement in the input data operation is identical to the output data operation (refer to subsection 2.5). The variations of the input data operation are the hardware data paths between the Megabus and the adapter. The output data operation transfers data from the Megabus logic through the ALU and the SPM back to the ALU and then to the adapter, whereas the input data operation transfers information from the adapter through the ALU, the SPM, and the Megabus logic to the Megabus.

III THEORY OF OPERATION - INTERMEDIATE

The MDC is divided into seven logic areas. These areas and the data, instructions, and control flows are illustrated in Figure 3-1. The seven logic areas are the microprogram control store, the arithmetic logic unit and accumulator, the scratch-pad memory and addressing, the test multiplexer, Megabus logic, adapter logic, and clock logic. Each area is described in detail under its appropriate subsection heading.

3.1 MICROPROGRAM CONTROL STORE FUNCTIONAL COMPONENTS (Figure 3-2)

3.1.1 Subroutine Return Address Register (SRAR)

The SRAR is used by firmware to store a microprogram control store (UPCS) address which will be branched to at some later time by a firmware routine or subroutine. To load a location with an address, the firmware issues a Load Return Address command, which generates the write enable function LDSRAR. This causes the address defined by bits 4-15 of the microprogram instruction register (UPIR) to be stored in the SRAR at the location specified by the SPMIR.

The SRAR is a four-word by 12-bit register file memory. Each word represents the return address of a particular adapter channel. The SRAR word that is being read or written is determined by the output of the scratch pad memory index register (SPMIR0- and SPMIR1-). The scratch pad memory index register (SPMIR) contains the number of one of the four device channels in the binary code (see subsection 3.3.2).

HONEYWELL PROPRIETARY AND CONFIDENTIAL

When firmware performs a Return Branch command, the contents of the location defined by the SPMIR is gated to the SRAR output. The output is then transferred by way of the microprogram address selector (UPAS) to the microprogram address counter (UPAC). The Return Branch command then presets the UPAC to an address previously stored in the SRAR.

3.1.2 Microprogram Address Selector (UPAS)

One of the two inputs to the UPAS is used as a preset for the UPAC. When the microprogram control store bit 2 (UPCS02+) is high, a state indicating a Go To command, the output of the UPAS (12 bits) reflects bits 4 through 15 of the UPCS. When UPCS02+ is low, the output of the UPAS reflects the twelve bits of the SRAR.

3.1.3 Microprogram Address Counter (UPAC)

The microprogram address counter (UPAC) is a 12-bit counter which is incremented once at the start of every clock cycle, except in the instance of a clear or load operation. The UPAC is cleared by CLRBD- , which is active only during an MDC Initialize (Master Clear); it is loaded when a load UPAC operation (LODUPA-) is active. The LODUPA- function causes the UPAC to reflect the address on the microprogram address selector output. The load UPAC operation is performed when a Go To command or a Return Branch command is decoded in the address control logic.

The low-order nine bits (3 through 11) of the UPAC are gated directly to the address lines of the UPCS. The high-order three bits (0-2) are fed to the Microprogram Control Enable (UPCSE1- to UPCSE8-), which is a 3-bit to 1-of-8-line decoder. The eight UPCSE functions are used to enable one of the eight rows within the microprogram control store.

3.1.4 Microprogram Control Store (UPCS)

For a description of the UPCS (shown in Figure 3-2), refer to subsection 2.4.3 of this manual.

3.1.5 Diagnostic Instruction (Test) Gate

When a control word issued to the MDC by software specifies the diagnostic mode, the MDC firmware executes a Set Test Mode (STMCM) command. This command sets the TSTMOD flip-flop, which in turn puts the MDC clock in the step mode and disables the microprogram control store outputs. At this time the diagnostic instruction (test) gate is enabled, allowing the transfer of information from the bus data register to the ORing network of the microprogram control store. Any subsequent transfer to the MDC while the TSTMOD flip-flop is set generates one clock cycle. During this cycle the data on the Megabus is transferred by the test mode gate and the ORing network to the microprogram instruction register. It is then executed as if it were a resident microinstruction.

The TSTMOD flip-flop remains set until software issues a Reset Test Mode (RTMCM) instruction or until a Master Clear signal occurs.

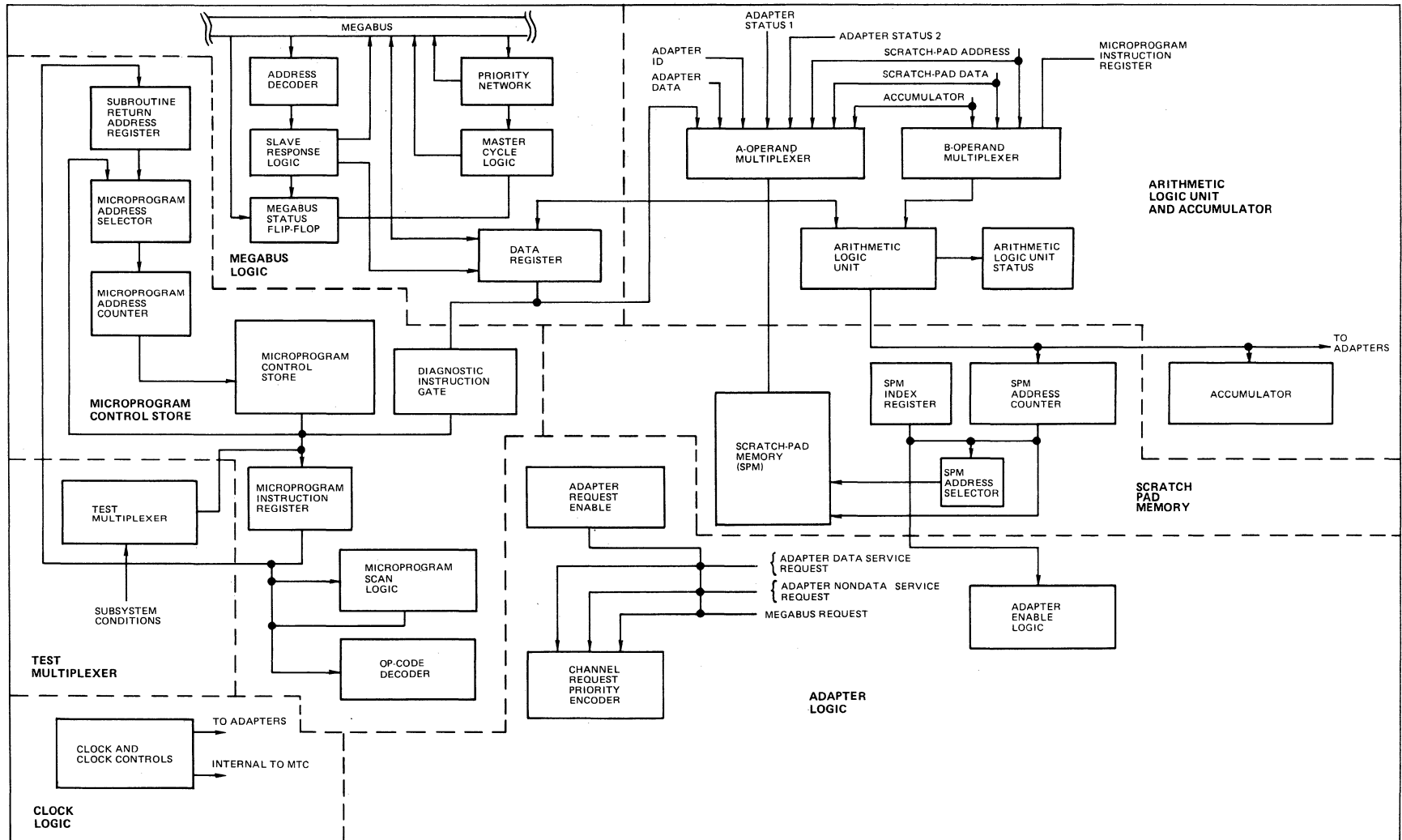


Figure 3-1 MDC Hardware Major Block Diagram

3.1.6 Microprogram Instruction Register (UPIR)

The microprogram instruction register (UPIR) is a 16-bit wide register used to store the output of the microprogram control store (UPCS) or the test gate for one clock cycle during a microinstruction execution. The UPIR is loaded at the leading edge of each cycle by CLKSIG+ unless the signal Clear Microprogram Instruction Register (CLRUPI) is active, which causes a reset to Zero. CLRUPI is active during the Master Clear operation and during a skipped cycle due to a successful test instruction.

3.1.7 Op-Code Decoder (OPCOD)

The high-order three bits (UPIR00+ to 02+) of the UPIR are fed to the op-code decoder, which performs a 3-bit to 1-of-8-lines decode. These lines indicate what type of microinstruction is being performed. The op-code decoder is enabled unless the MDC is in the process of performing a UPCS scan or unless the UPIR is being cleared by the CLRUPI function. The eight output lines of the op-code decoder, in conjunction with various other UPIR bits, implement and control the MDC and device adapter hardware.

To increase the speed of operation during the firmware routines, the op code of the Branch (Go To) command is decoded directly from the output of the UPCS rather than from the UPIR. For the same reason, the address for the Go To command is taken from the output of the UPCS. In the case of all other command types, the op code is decoded from the output of the UPIR.

3.1.8 Scan Logic (Figure 3-3)

The scan logic in the MDC ensures the integrity of the UPCS. This logic performs a longitudinal check on each bit position of the UPCS outputs and causes the MDC clock to halt if an error is detected. If no errors are detected, the MDC goes on to execute the firmware portion of the Quality Logic Test.

The scan mode flip-flop (SCNMOD) sets with Master Clear (MSTCLR-), disabling the op-code decoder (see Figure 3-2) and inhibiting Branch commands. With the commands inhibited, the MDC loads the UPIR with the output of the UPCS and increments the UPAC to the next location at each sequential clock cycle. This process continues from location 000 to location FFF, at which time a carry-out of the UPAC (UPAC2C-) is detected. The UPAC then returns to zero, beginning another scan.

The scan bit selectors (SCNBSH and SCNBSL) select one bit of the UPIR at the address defined by the output of the scan bit address counter. The selected UPIR bit provides the input to the scan bit sum flip-flop (SCNBSM). The SCNBSM half-adds the selected bit at the start of each clock cycle with its previous contents. At the end of one scan through the UPCS, the scan bit sum flip-flop has added the specified bit of each location with an expected result of zero (no error). If the sum is not zero at the termination of each scan, the scan error gate output (SCNERR+) is high and sets the scan error flip-flop (UPIERR). The UPIERR causes the MDC clock to halt, and the contents of the scan bit address counter will indicate which bit of the UPCS is in error.

If an error is not detected at the end of a scan, the scan bit address counter will increment due to the overflow of the UPAC, and the UPAC will wrap around to location zero. The scan operation is repeated with the next bit of the UPIR as an input to the scan bit sum flip-flop. The scan operation continues until an error is detected or until all bits of the UPIR have been checked, at which time the scan bit address counter (SCNBAC) will overflow and reset the scan mode flip-flop. When the scan mode flip-flop is reset, the op-code decoder and Branch commands are re-enabled, and normal execution of firmware begins.

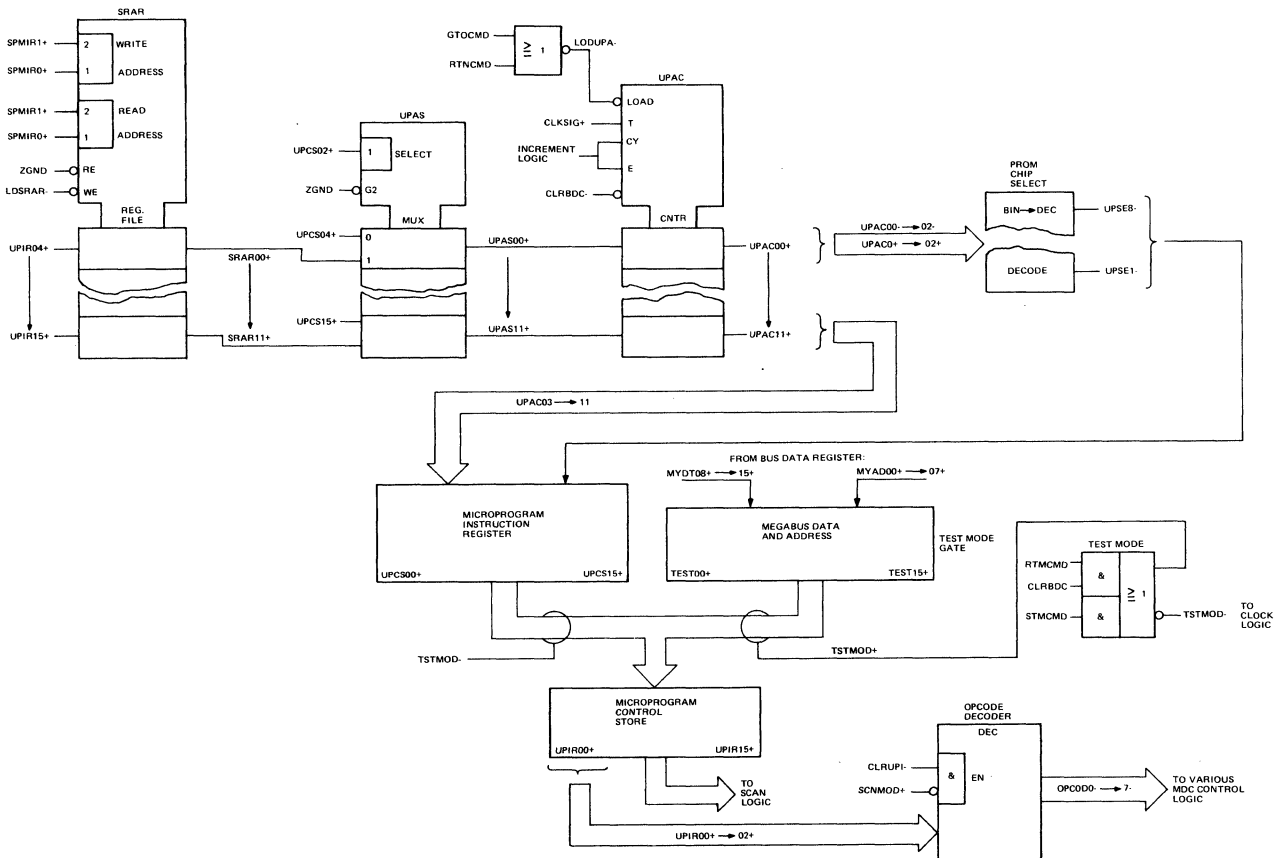


Figure 3-2 Microprogram Control Store Functionality

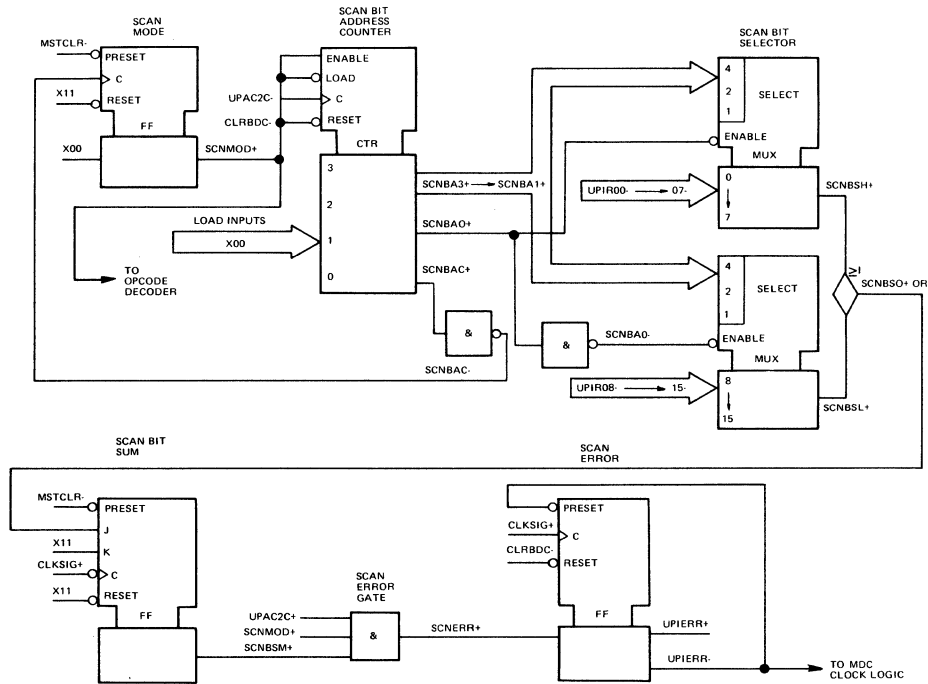


Figure 3-3 Scan Logic

3.2 ARITHMETIC LOGIC UNIT AND ACCUMULATOR FUNCTIONAL COMPONENTS

3.2.1 A-Operand Multiplexer (ALUAX) (Figure 3-4)

The A-operand multiplexer (AOP mux) can select one of eight types of data fields according to a three-bit multiplexer address defined by bits 3 to 5 of the UPIR. Table 3-1 shows the various bit configurations and the register selected by each configuration.

For address configurations equal to 0 through 3, UPIR bit 3 reset, the output from the accumulator, scratch pad data, scratch pad address, or bus data is selected as an input to the AOP multiplexer.

For address configurations equal to 4 through 7, UPIR bit 3 set, the output of the adapter's data selector is used as an input to the AOP multiplexer. Bits 4 and 5 of the UPIR then select one of the four adapter registers visible to the MDC: data, ID, status 1, or status 2.

Only the one adapter which is active will have its data enabled at the selector output. The outputs of the adapter's data selectors are Ored on the MDC, and the Ored outputs are fed only to the AOP multiplexer.

The loads of the 8-bit output of the AOP multiplexer are the ALU, the test multiplexer, the scratch pad memory, and the bus data register.

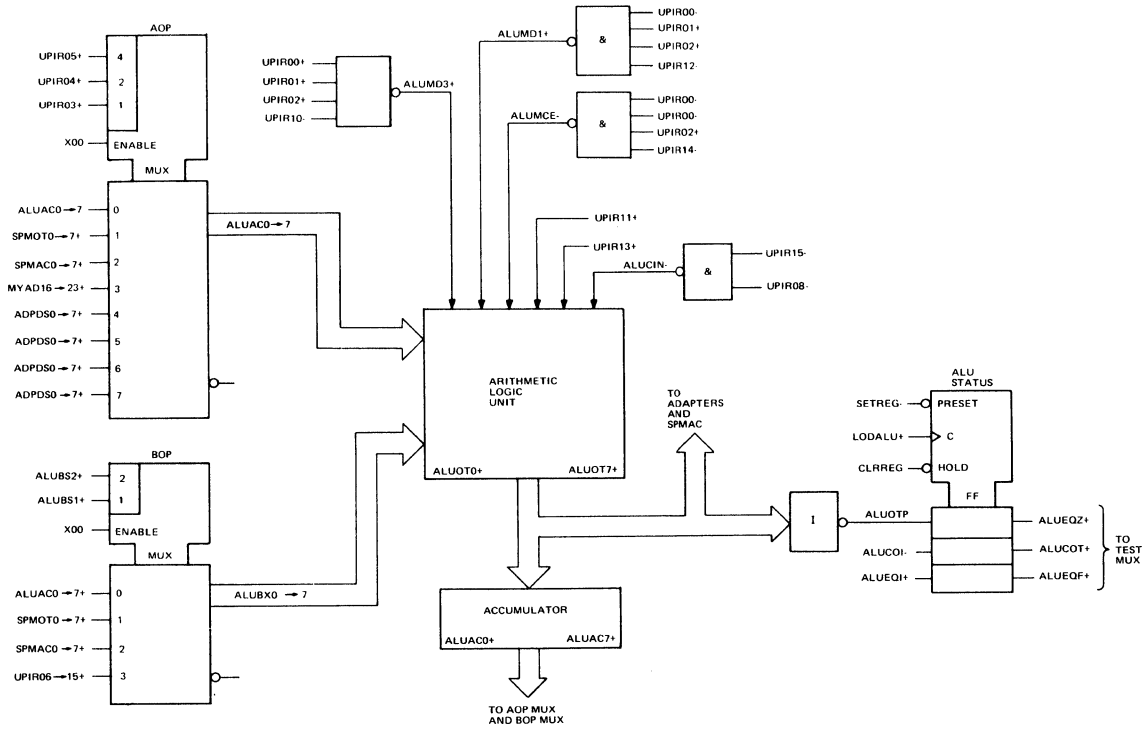


Figure 3-4 ALU and ACU Functionality

Table 3-1 AOP Multiplexer Input Selection

UPIR ADDRESS CONFIGURATION BITS			SELECTED REGISTER	MNEMONIC
03	04	05		
0	0	0	Accumulator	ALUAC0+ → 7+
0	0	1	Scratch Pad Data	SPMOT0+ → 7+
0	1	0	Scratch Pad Address	SPMAC0+ → 7+
0	1	1	Bus Data	MYAD16+ → 23+*
1	0	0	Adapter Data	ADPDS0+ → 7+
1	0	1	Adapter ID	ADPDS0+ → 7+
1	1	0	Adapter Status 1	ADPDS0+ → 7+
1	1	1	Adapter Status 2	ADPDS0+ → 7+

*The data seen at this point depends on the number of shifts performed on the bus data register.

3.2.2 B-Operand Multiplexer (ALUBX) (Figure 3-4)

The B-operand multiplexer (BOP mux) can select one of four data fields as a B-input to the ALU. The BOP is defined during ALU commands by UPIR bits 6 and 7. When performing Constant commands, the BOP multiplexer selects the output of the UPIR bits 6 to 10, 12, 14, and 15 for the input to the ALU. These particular UPIR bits represent the data constant to be operated with during the constant command.

The address for the data field to be loaded into the BOP mux is a 2-function decode (ALUBS1, ALUBS2) of the operation code (UPIR bits 0+ through 2+) and UPIR bits 6- and 7-. Table 3-2 shows the four possible configurations of ALUBS1+ and ALUBS2+ and the data input selected by each configuration.

Table 3-2 BOP Multiplexer Input Selection

UPIR BIT DECODE		SELECTED DATA INPUT	MNEMONIC
Address Functions ALUBS1•ALUBS2			
0	0	Accumulator	ALUAC0 → 7
0	1	Scratch Pad Data	SPMOT0 → 7
1	0	Scratch Pad Address	SPMAC0 → 7
1	1	UPIR Constant	UPIR06 → 10,12,14,15

3.2.3 Arithmetic Logic Unit (ALU) (Figure 3-4)

The ALU can perform an 8-bit arithmetic or logic operation on the data supplied by the AOP and BOP multiplexers. The type of operation to be performed is determined by the four mode signals (ALUMD3+, UPIR11+, ALUMD1+, UPIR13+), the carry enable function (ALUMCE-), and the carry input (ALUCIN-) to the ALU. Table 3-3 shows the relationship between these functions and the operation to be accomplished by the ALU.

These mode signals, as well as carry enable and carry in, are explicitly defined by bits of the UPIR during ALU commands having an op code of 3. The firmware can also set Carry Enable and specify that the carry-in is to reflect the previous ALU command carry-out as defined by the flip-flop ALUCOT.

When an ALU command is not being performed, the mode signals (ALUMD3+, ALUMD1+) and the carry-enable signal (ALUMCE-) default to a One state.

The two remaining ALU mode lines (UPIR11+, UPIR13+) do not have a default state but rather vary according to the output of the UPIR bits 11 and 13. These two bits select one of four operations which can be performed when executing a Constant command.

The ALU data output is fed to the accumulator, to the scratch pad memory address counter, and to the device adapters. In addition, the ALU outputs are inputs to a set of inverters whose wire-ANDed outputs provide the data for the ALU equal zero (ALUEQZ) flip-flop. This status flip-flop, along with the ALU equal FF (ALUEQF) flip-flop and the ALU carryout (ALUCOT) flip-flop, is set during the ALU operation by the signal LODALU+. In this manner, the ALU status flip-flops will be set or reset during an ALU command and will remain valid for firmware interrogation until the subsequent ALU operation.

The ALUEQZ and ALUEQF status flip-flops indicate an all-Zeros ALU output and an all-Ones ALU output, respectively. The ALUCOT flip-flop indicates a carry-out of the ALU, which serves to indicate the relative magnitudes of the AOP and BOP fields during an ALU subtract operation. Table 3-4 shows the relationship between the ALU carry-in, the ALU carry-out, and the size of the AOP and BOP fields.

For diagnostic purposes, each of the ALU status flip-flops can be set independently of the ALU outputs by the function SETREG-. They can also be cleared (reset) by the signal CLRREG-. Both SETREG- and CLRREG- can be enabled by firmware commands, with CLRREG- also being active during Master Clear. During a Master Clear operation, CLRREG- causes the ALU status flip-flops to be zeroed. After the first ALU command, and after Master Clear, these flip-flops will reflect the proper state indicated by the ALU outputs.

3.2.4 Accumulator (ACU) (Figure 3-4)

The ACU, which can be cleared by the function CLRREG-, is an 8-bit register that provides temporary storage of the ALU output. The ACU will be loaded at CLKSTB- time during the following operations:

HONEYWELL PROPRIETARY AND CONFIDENTIAL

- When performing a Load Constant command with the scratch pad memory (SPM) or the ACU defined as the A-operand.
- When performing an ALU command with the destination of the result specified as either the A-operand (SPM or ACU) or the B-operand (SPM or ACU).

The outputs of the ACU are fed only to the AOP and BOP multiplexers.

Table 3-3 ALU Functionality

HEX CONFIGURATION	POSITIVE LOGIC		
	ALUMCE- = 1 = H LOGIC OPERATIONS	ALUMCE- = 0 = L, ARITHMETIC OPERATIONS	
		ALUCIN- = 1	ALUCIN- = 0
L L L L (0000)	$F = \bar{A}$	$F = A$	$F = A \text{ plus } 1$
L L L H (0001)	$F = \overline{A + B}$	$F = A + B$	$F = (A + B) \text{ plus } 1$
L L H L (0010)	$F = \overline{AB}$	$F = A + \bar{B}$	$F = (A + \bar{B}) \text{ plus } 1$
L L H H (0011)	$F = 0$	$F = \text{minus } 1 \text{ (2's compl)}$	$F = \text{ZERO}$
L H L L (0100)	$F = \overline{AB}$	$F = A \text{ plus } \overline{AB}$	$F = A \text{ plus } \overline{AB} \text{ plus } 1$
L H L H ((0101)	$F = \bar{B}$	$F = (A + B) \text{ plus } \overline{AB}$	$F = (A + B) \text{ plus } \overline{AB} \text{ plus } 1$
† L H H L (0110)	$F = A + B$	$F = A \text{ minus } B \text{ minus } 1$	$F = A \text{ minus } B$
L H H H (0111)	$F = \overline{AB}$	$F = \overline{AB} \text{ minus } 1$	$F = \overline{AB}$
H L L L (1000)	$F = \bar{A} + B$	$F = A \text{ plus } AB$	$F = A \text{ plus } AB \text{ plus } 1$
†† H L L H (1001)	$F = \overline{A + B}$	$F = A \text{ plus } B$	$F = A \text{ plus } B \text{ plus } 1$
H L H L (1010)	$F = B$	$F = (A + \bar{B}) \text{ plus } AB$	$F = (A + \bar{B}) \text{ plus } AB \text{ plus } 1$
H L H H (1011)	$F = AB$	$F = AB \text{ minus } 1$	$F = AB$
H H L L (1100)	$F = 1$	$F = A \text{ plus } A^*$	$F = A \text{ plus } A \text{ plus } 1$
H H L H (1101)	$F = A + \bar{B}$	$F = (A + B) \text{ plus } A$	$F = (A + B) \text{ plus } A \text{ plus } 1$
H H H L (1110)	$F = A + B$	$F = (A + \bar{B}) \text{ plus } A$	$F = (A + \bar{B}) \text{ plus } A \text{ plus } 1$
H H H H (1111)	$F = A$	$F = A \text{ minus } 1$	$F = A$

†Subtract Mode

††Add Mode

*Each bit is shifted to the next more significant position.

NOTE

With $\overline{C_E}$ at a logical ONE, $\overline{C_{in}}$ is irrelevant.

Table 3-4 ALU Carry Flip-Flop Indications

ALU CARRY-IN (ALUCIN)	ALU CARRY-OUT (ALUCOT)	AOP AND BOP RELATIONSHIP
1	0	$A \leq B$
1	1	$A > B$
0	1	$A \geq B$
0	0	$A < B$

3.3 SCRATCH PAD MEMORY FUNCTIONAL COMPONENTS

3.3.1 Scratch Pad Memory Index Control Flip-Flop (SPMICF) (Figure 3-5)

The SPMICF is set or reset as a result of a Set or Reset Index command (SRICMD+) according to the state of the UPIR bit 9 (UPIR09+). The SPMICF is also reset as a result of the CLRREG-function being active.

The output of the SPMICF determines the source from where the scratch pad address will be loaded. When the SPMICF is reset, the memory is accessed by all eight bits (absolute address) of the scratch pad memory address counter (SPMAC). However, when the SPMICF is set, the memory is addressed by way of the six low-order bits of the SPMAC, and the two high-order bits (relative address) are taken from the contents of the scratch pad memory index register (SPMIR).

3.3.2 Scratch Pad Memory Index Register (SPMIR) (Figure 3-5)

The MDC can be configured with up to four devices (channels). Although every device can be busy simultaneously, the MDC multiplexes the channel activity. The SPMIR (SPMIR0+, SPMIR1+) is used to specify which channel is active at any given time. The contents of the SPMIR are utilized as the two high-order bits to address the scratch pad memory or the subroutine return address register (see Figure 3-2). The SPMIR also determines which adapter is enabled for proper firmware command execution.

The two-bit SPMIR can be loaded with a specific channel number from the firmware (UPIR12+, UPIR13+). It can also be loaded with the channel number designated by the channel request priority encoder (CRENX0+, CRENX1+). The input to the SPMIR is selected as a result of bit 11 of the instruction register (UPIR11+) and is loaded whenever the function LODSPI is active.

3.3.3 Scratch Pad Memory Address Counter (SPMAC)
(Figure 3-5)

The SPMAC is an 8-bit counter which is parallel loaded with the output of the ALU or sequentially incremented by a Write Increment Address command (WIACMD). The ALU output (ALUOT0+ through 7+) is loaded into the SPMAC at CLKSTB+ time whenever LODSPA+ is active. The control signal LODSPA+ is generated for those ALU commands (AOPSPA-, BOPSPA-) which specify that the SPMAC be loaded with the ALU output. It is also generated during Constant commands, which utilize the SPMAC as the A-operand.

The SPMAC is incremented whenever the control signal WIACMD+ is active at CLKSTB+ time, causing the contents of the SPMAC to be increased by one. The increment (WIAFLP) flip-flop is set during the execution of a WIACMD and resets at the completion of the subsequent command. In this manner the write is performed during the initial cycle of a WIACMD, and the increment occurs one cycle later.

The CLRREG- signal, which is generated by a Master Clear and under firmware control, resets the SPMAC to zero and also resets the WIAFLP flip-flop.

3.3.4 Scratch Pad Memory Address Selector
(SPMAS) (Figure 3-5)

Firmware normally indexes the scratch pad memory, thereby dividing the memory into quadrants, one for each available channel. Each quadrant of memory has the same topology (refer to Table 4-1). However, the activity within a specific quadrant may vary according to device type.

The SPMAS determines the two most significant bits of the scratch pad address. When the SPMICF is set, the SPMAS selects the contents of the SPMIR as the two high-order bits of the address. If the SPMICF is reset, the output from the SPMAS reflects the two high-order bits of the SPMAC.

3.3.5 Scratch Pad Memory (SPM) (Figure 3-5)

The SPM is a 256-location by 8-bit read/write memory which stores information that is required by or generated by each channel. Data from the AOP multiplexer is written in the SPM during the Memory Write command at the relative location defined by bits 0 and 1 of the SPMAS and bits 2 through 7 of the SPMAC. Since the input data to the SPM is received from the AOP mux, all firmware-visible registers can be implemented as an input local register. (Refer to subsection 3.4.1.2.) The data out of the SPM is delivered to the AOP and the BOP multiplexers.

The decode of a Memory Write command is ANDed with the clock signal CLKSTB+, which combination produces the write pulse MWTCMD-. The MWTCMD- is approximately 40 ns wide and occurs just prior to the end of a memory write cycle, guaranteeing data setup and hold times. (Refer to subsection 3.7 for a description of the clock signals.)

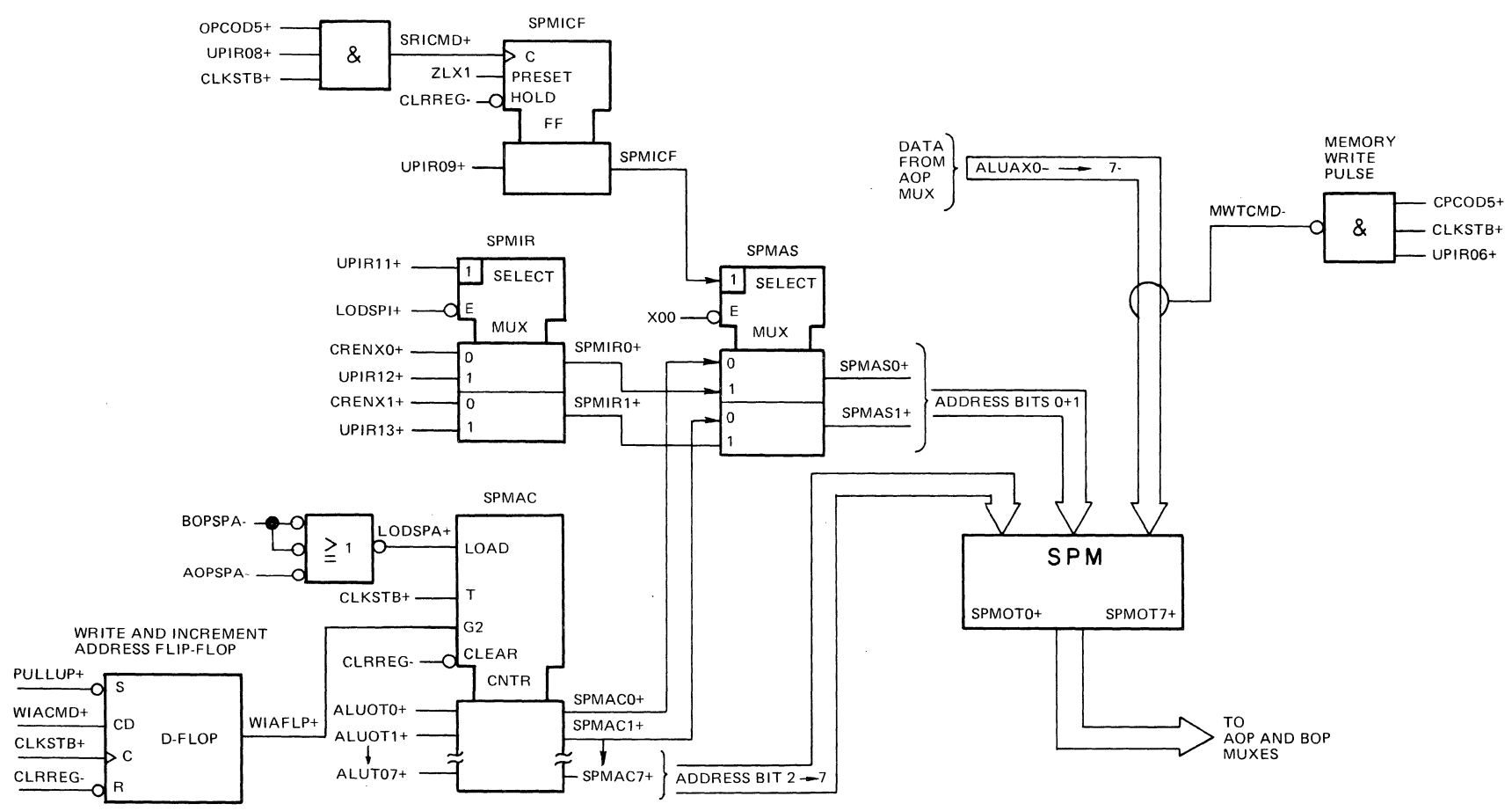


Figure 3-5 Scratch Pad Memory Functionality

3.4 MEGABUS LOGIC FUNCTIONAL COMPONENTS

3.4.1 Bus Data Register (BDR)

(Figure 3-6)

The bus data register consists of a 40-bit register for address and data storage, a 5-bit register for parity storage, parity generating and checking logic, and a series of flip-flops which control register access by the MDC and/or the Megabus. The BDR, primarily an input/output buffer for the Megabus, can be parallel loaded by the Megabus or byte-serial-loaded by the MDC.

3.4.1.1 Parallel Load Operation (Figure 3-6)

The BDR is cleared and then parallel loaded with address, data, and parity from the Megabus each time the MDC is capable of acknowledging (ACK) a Megabus cycle. The register, parallel loaded during any Megabus cycle during which a load is permissible, occurs concurrently with address decoding and Megabus cycle response generation. The information must be loaded into the BDR in this manner because the data on the Megabus becomes invalid at the leading edge of the slave's response.

The Clear BDR (CLRBDR-) signal clears the BDR prior to the time the parallel loading occurs. This clearing prevents ORing the incoming data and the present register contents. CLRBDR- becomes active whenever the BDR is to be loaded (LODBDR+) and remains active until the data on the Megabus becomes valid.

3.4.1.2 Byte-Serial Load Operation (Figure 3-6)

For loading/unloading from the MDC, the BDR is configured as a 5-byte shift register, each byte containing eight data bits and a parity bit. The MDC loads and unloads the BDR one byte at a time with the AOP-multiplexer-visible registers both receiving and supplying data. The Shift Bus Data Register (SFTBDR+) control signal is active whenever a Shift Data Register command (SDRCMD-) is executed, causing the register to shift one byte position. During the shift operation for unloading the BDR, the contents of the low order byte (byte 4) is reflected in the AOP multiplexer; each of the other bytes (bytes 0 through 3) is shifted down one byte position. At CLKSTB+ time the output of the AOP multiplexer is reloaded into byte position 0, effectively performing an end-around-byte shift. When performing a load BDR operation, the same procedure is followed except that the output of the AOP multiplexer will reflect one of the seven other firmware-visible registers (refer to subsection 3.2.1).

The parity bit for a Load BDR operation is computed on the output of the AOP multiplexer and written into the parity register in the position corresponding to byte 0. During a BDR unload, the parity is also generated on the output of the AOP multiplexer (data from the BDR). This parity is compared with the parity output of the BDR for detection of a Bus Parity Error (BSPYER+). If the two compared inputs are not equal, the BSPYER+ is set, and the bus parity error flip-flop (BSPYCK+) sets at CLKBPC+ time. Table 3-5 indicates when the MDC is master (initiator of bus request) and when it is the slave (receiver of bus request) for each byte of the BDR during a Megabus cycle.

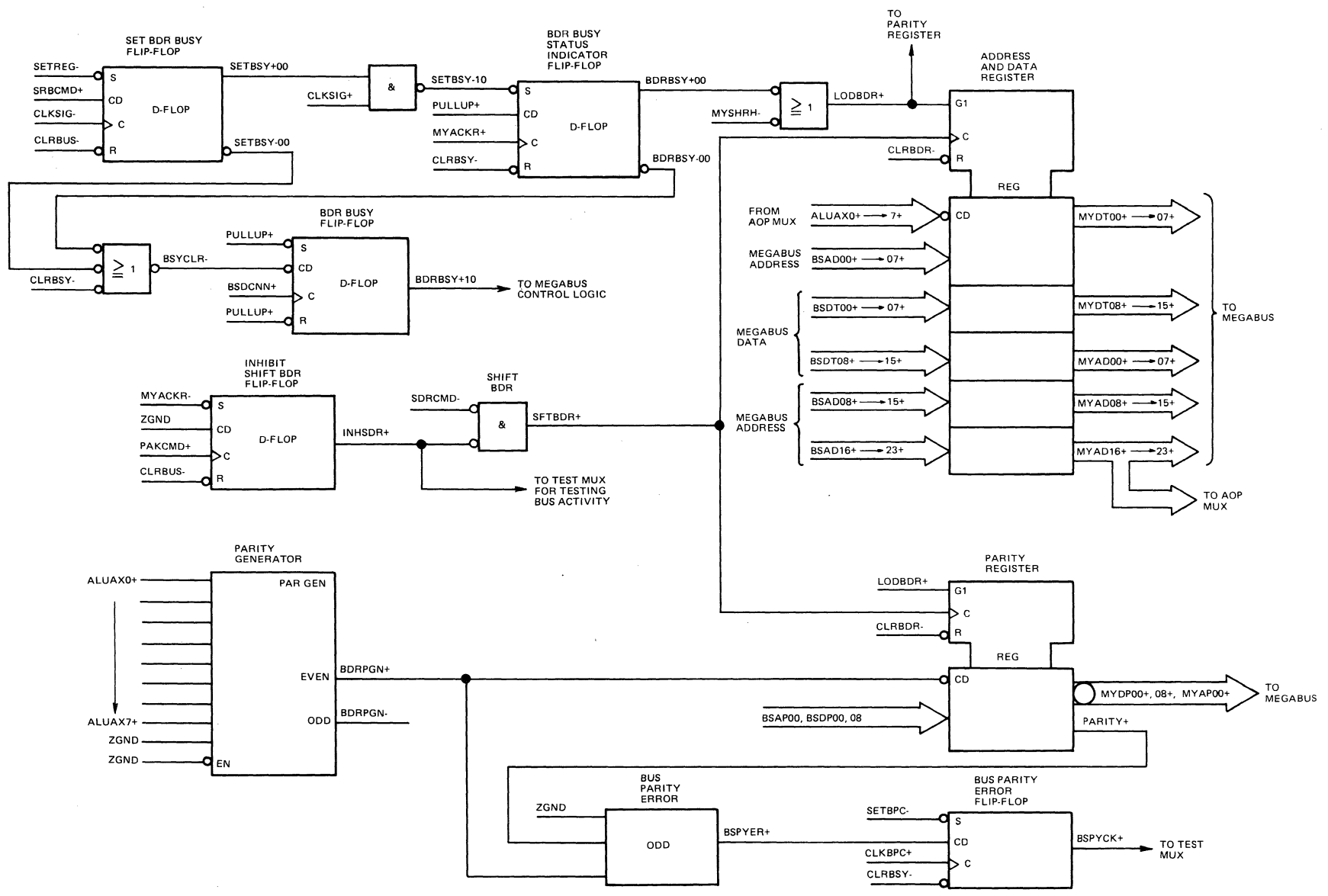


Figure 3-6 Bus Data Register

Table 3-5 Bus Data Register Application

BYTE	MDC	
	MASTER	SLAVE
0	MSB Data	MSB Address
1	LSB Data	MSB Data
2	MSB Address	LSB Data
3	MID Address	MID Address
4	LSB Address	LSB Address

NOTE

The shift input is loaded into byte 0 from the AOP multiplexer. The register output is obtained from byte 4 and delivered to the AOP multiplexer.

3.4.2 Bus Data Register Control (Figure 3-6)

The bus data register is used as an input/output buffer for the Megabus and is, therefore, subject to simultaneous access by the MDC and by incoming Megabus cycles. Controls are required to limit register access to one user at a time. To accomplish this, five cycle parameters (see Table 3-6) and several gates serve to inhibit or enable various register operations.

NOTE

In Figure 3-6, the mnemonic for the bus data register busy flip-flop is BDRBSY+10, and the mnemonic for the bus data register busy status indicator flip-flop is BDRBSY+00.

The bus data register busy (BDRBSY+10) flip-flop is implemented to control Megabus access to the BDR. This flip-flop is clocked at the leading edge of each Megabus cycle by the signal BSDCNN, and the Megabus interface logic is busy if any of the following conditions are met:

1. The BDR is busy as indicated by the bus data register busy status indicator (BDRBSY+00) flip-flop.
2. The MDC firmware is in the process of setting the BDR busy flip-flop as indicated by the set BDR busy (SETBSY) flip-flop.
3. The MDC is in the process of resetting the BDR busy flip-flop as indicated by the CLRBSY- signal.

The BDR busy flip-flop being set inhibits the Megabus from accessing the BDR and will normally cause a Wait response to be generated for the Megabus.

The BDR busy status indicator flip-flop can be set by the MDC firmware or by the Megabus hardware. The output of this flip-flop

is examined at the leading edge of each Megabus cycle to determine if an Acknowledge signal can be generated. The output also generates the load pulse for the BDR (LODBDR+).

When the MDC firmware wants to utilize the BDR, it must first set the BDR busy status indicator flip-flop in order to inhibit the clearing of the BDR and to inhibit Megabus access. To accomplish this, a Set Register Busy command is executed, which sets the SETBSY flip-flop for one clock cycle. The SETBSY flip-flop being set will enable the BDR Busy status indicator flip-flop at the initiation of the subsequent clock cycle, disabling the LODBDR+ function.

The Megabus can set the BDR busy status indicator flip-flop with the leading edge of the Acknowledge (ACK) signal generated by the MDC. In either case, whether this flip-flop is set by firmware through the SETBSY+00 or with the leading edge of ACK, the BDR busy status indicator flip-flop stays set until CLRBSY- is activated. The signal CLRBSY- can be activated by the firmware through a firmware command or by the Megabus as a result of a Master Clear.

An ACK to the Megabus by the MDC sets the inhibit shift BDR (INHSDR) flip-flop. This flip-flop inhibits the firmware from performing operations on the BDR until a Processor Acknowledge command (PAK) is executed by firmware, acknowledging the Megabus cycle. A PAK command resets the INHSDR flip-flop and allows the firmware to unload the BDR. If an ACK is generated while the firmware is trying to set the BDR busy, the INHSDR flip-flop will be set, keeping the firmware from accessing the BDR.

Table 3-6 Cycle Parameters

AOP BITS	PARAMETERS
0	Cycle Request
1	Memory Reference
2	Response Required
3	Second Half-Read Cycle
4	Byte Mode
5	RFU
6	RFU
7	RFU

3.4.3 Master Cycle Logic and Priority Network (Figure 3-7)

When the MDC (Master) initiates a transfer over the Megabus to the main memory or to the CP, it first sets the BDR busy status indicator flip-flop and loads the register with address and data. It then loads the cycle parameters into an AOP register (refer to subsection 3.2.1) and issues a Cycle command. The master cycle logic has control of the Megabus cycle once the Cycle command is executed, and firmware must test the master cycle status flip-flops to determine when the Megabus cycle is complete. When a Cycle command is executed, the cycle parameters are set according to the outputs of the AOP multiplexer. In addition, the cycle request (CYCREQ) flip-flop sets, initiating a Megabus request. Table 3-6 shows the usage of the cycle parameters.

When the CYCREQ flip-flop is set and when the Megabus is not busy (BSBUSY-), the Set Request (SETREQ-) signal becomes active, causing the Request (MYREQT) flip-flop to set. The request flip-flop goes to one of the inputs of the function Set Data Cycle Now (SETDCN-). This Megabus request inhibits other units on the Megabus from initiating a new request and causes the output of the MDC priority network (BSMYOK+) to go low.

The priority network consists of nine input signals (BSAUOK+ through BSIUOK+) and one output signal (MYDCNN-). The MDC has priority when all the inputs are high, indicating that no other unit with higher priority has a Megabus request active. The priority network is sampled when the Megabus is inactive, as determined by the CLRDCN- and BSDCNB+ functions. The priority network output being high to other units on the Megabus with lower priority shows that the MDC does not have a request active and that the unit generating the BSIUOK+ coming into the MDC does not have a request active.

With the request flip-flop set and with the completion of any previous Megabus cycle (BSDCNB+), the MDC will set its data cycle now (MYDCNN) flip-flop provided it has the highest priority on the Megabus. When all conditions of SETDCN are met, the MYDCNN flip-flop sets and stays set until the slave unit responds. The response of the slave (CLRDCN-) clears the MYDCNN flip-flop, resetting the request unless the response is a wait, in which case the request flip-flop remains set.

An ACK response (ACKRSP) by the slave to a first half read cycle will set the second half read history (MYSHRH) flip-flop. The MYSHRF flip-flop is used to generate the load and clear signals to the BDR during the second half read cycle. This is necessary because the BDR becomes busy prior to the first half read cycle. The busy condition inhibits other units on the Megabus from accessing the MDC's BDR between first and second half read cycles and allows the MDC to acknowledge the second half bus cycle. The MYSHRH flip-flop aids in differentiating between the slave's response and other cycles addressed to the MDC.

3.4.4 Slave Response Logic (Figure 3-8)

The MDC has the capability to respond to a Megabus cycle in four ways which are prioritized and have the following significance:

1. No response indicates that the addressed channel has no associated device adapter installed.
2. The NAK response indicates that the addressed channel is not in a ready condition. A channel may be busy doing a device operation.
3. A wait response indicates that the addressed channel is ready but that the BDR is busy. The master unit should retry the Megabus cycle.
4. An ACK response indicates that the addressed channel is ready and that the MDC has stored the information on the Megabus in the BDR.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Each device adapter has an output level (ADPGND-01 through -04) which indicates that the adapter is installed. The level from each adapter goes to the adapter present multiplexer, which selects the level corresponding to the addressed channel. The output of the adapter present multiplexer (ADPGND-10) is required by the cycle response logic. A high output indicates that an adapter is not installed, inhibiting a response by the MDC.

There are four channel ready flip-flops which firmware uses to store the status of each channel at the leading edge of each Megabus cycle. The outputs of these flip-flops (CHNRDY-11 through -14) are gated to the channel ready multiplexer. The state of the addressed channel, as reflected by the channel ready flip-flop is selected by the channel ready multiplexer and fed to the cycle response logic. The address of the channel is augmented by the Second Half Bus Cycle signal (BSSHBC+), causing the addressed channel to appear ready for any second half read cycle. The channel can also be forced to the ready state whenever the function code on the address bus is a One.

The register busy latch stores the status of the BDR at the start of each Megabus cycle. As described in subsection 3.4.1, the BDR is considered busy if it is being set busy, already busy, or being reset from the busy state. The output of the register busy latch goes to the cycle response logic and is used to generate a wait response.

The last input to the cycle response is the output of the address decoder which determines if the MDC is being addressed by the Megabus. The decoder, consisting of two hexadecimal rotary switches and gating, compares the address on the Megabus against the address of the MDC. The address of the MDC is determined by the setting of the switches and is not visible to the MDC until the first Megabus cycle acknowledged by the MDC. The decoder output is active (low) when bits 8 through 14 of the Megabus agree with the switch settings and when the Memory Reference Signal (BSMREF+) on the Megabus is low. Bits 15 and 16 of the Megabus (BSAD15+,16+) determine which of the four MDC channels is being addressed and are not examined by the decoder. These bits are the selection bits for the adapter present and channel ready multiplexer.

The cycle response gating generates the ACK, NAK, or wait response if the MDC is addressed and the addressed channel has an adapter installed. The output of the response gate is stored in the response latch 60 ns after the cycle was initiated with BSDCND+ coming high. The output of the response latch goes to the Megabus drivers and to various logic on the MDC (i.e., the ACK response latch is used to clock the response status register).

The response status register stores pertinent Megabus signals, including write mode (BSWRIT+) and red (BSREDD+)/yellow (BSYELO+) status signals. These signals (except BSWRIT+), along with NAK response (NAKRSP+) and bus parity check (BSPYCK+), are fed to the error collector, which transmits its output (MEMERR-) to the test multiplexer.

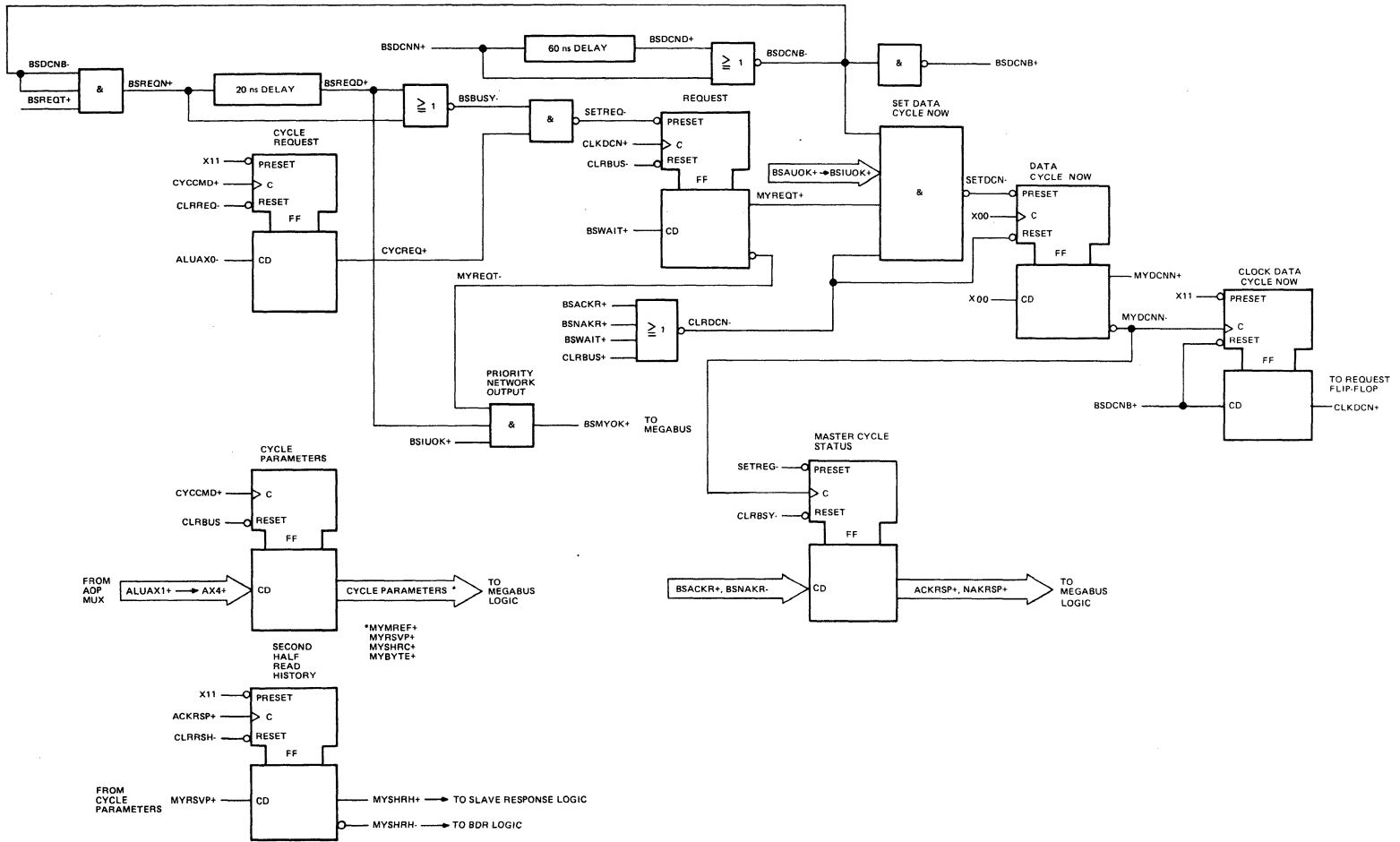


Figure 3-7 Master Cycle Logic and Priority Network

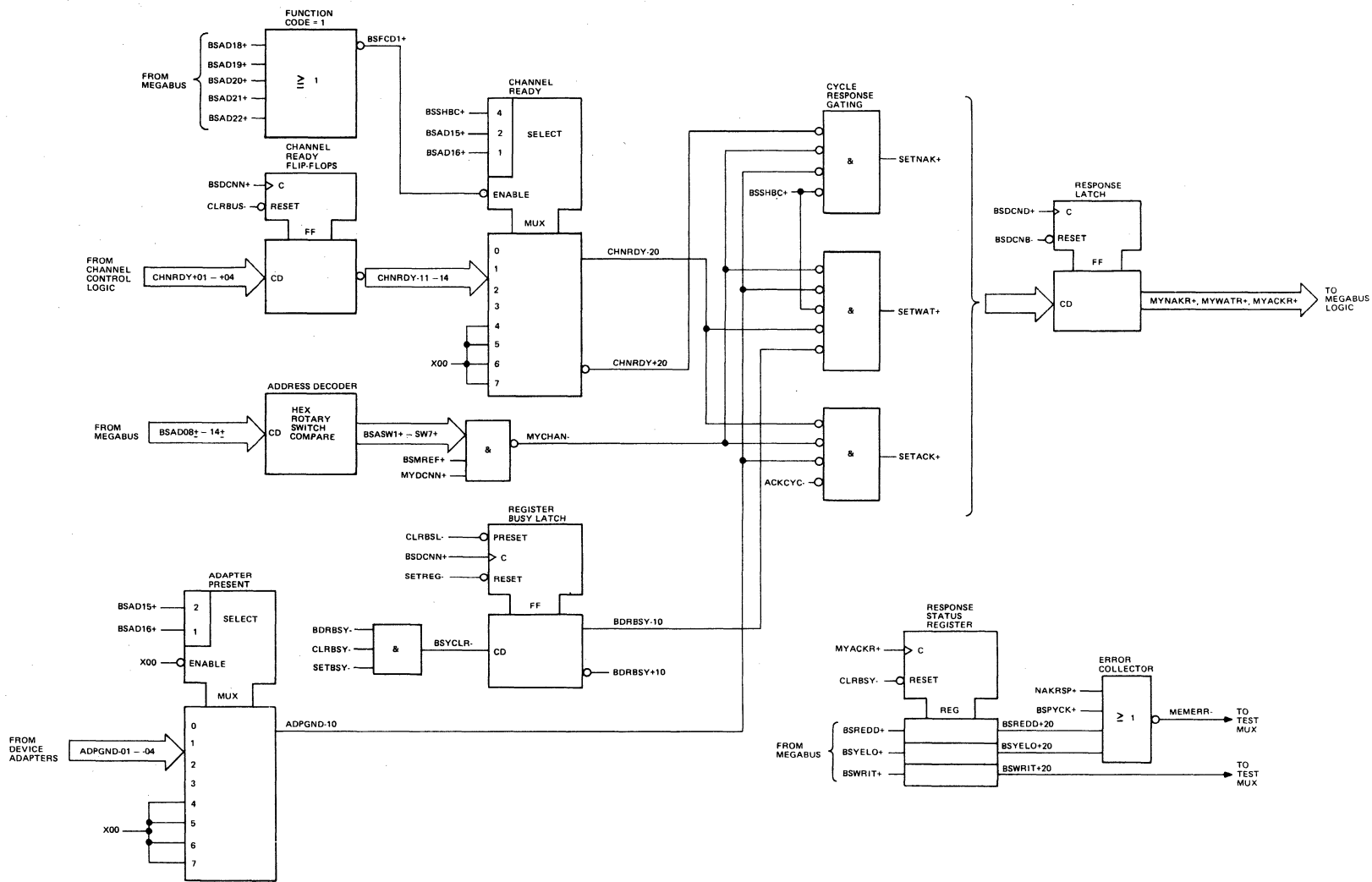


Figure 3-8 Slave Response Logic

3.5 CHANNEL CONTROL FUNCTIONAL DESCRIPTION

3.5.1 Channel Ready Signals

The channel ready (CHNRDY) register (Figure 3-9) is used to store the ready state of a device channel (set or reset). A channel is normally ready unless a device operation or a software interrupt is in progress on the device channel. The CHNRDY register is enabled by signal CHNRDE, the decode of a Megabus logic firmware command. The selection of the CHNRDY register is achieved by the scratch pad memory index register bits (SPMIR0+, 1+). The selected channel ready signal output is set or reset according to the state of the UPIR bit 12 (UPIR12+) of the firmware command which enabled the CHNRDY register. The outputs of the register (CHNRDY+01 through +04) are sent to the channel ready flip-flops (Figure 3-8).

3.5.2 Adapter Enable Gate

The adapter enable gate (Figure 3-9) generates one enable signal (ADPENB+) for each device channel, using the decoded contents of the SPMIR (SPMIE0- through 3-) to determine which device adapter is to be enabled. Those device adapters which are not enabled ignore all the control signals on the MDC/adapter interface and also disable their data selector outputs, allowing the MDC to communicate with a single device adapter at one time. The adapter enable gate outputs (ADPENB+01 through +04) do not affect the dialogue between the device and device adapter whether selected or not.

3.5.3 Adapter Control Signals

The adapter control signals (Figure 3-9) are used to control the device adapters. These control signals (ADPPLS+, ADPDBT+, ADPCD1+ through 3+) are generated from a firmware command and the UPIR bits (UPIR06- through 10-) of the same command. The reset signal (CLRADP+) sent to the device adapters, is a result of a Master Clear (MSTCLR-) or firmware command (RDACMD+).

3.5.4 Adapter Register Control Signals

The adapter register control signals (Figure 3-9) are used to load data into device adapter registers and are developed by an octal to decimal decoder. The enable (AOPENB-) for the decoder is developed by either a Constant command or by an ALU command, with the result storage designated on the device adapter register. The decoding of the device adapter register to be loaded is achieved by the UPIR bits (UPIR03+ through 05+) of the same firmware command that enabled the decode. The decoded outputs Load Data Register (LODADR-), Load Status 1 (LODAS1-), and Load Status 2 (LODAS2-) are sent to the enabled device adapter.

3.5.5 Request Logic

The request logic (Figure 3-9) determines the channel activity required by the MDC. The nondata service requests (ADPSRQ-01 through -04) and data service requests (ADPDRQ-01 through -04) from each device adapter are fed to the inputs of the priority encoder.

When the MDC receives a device service request (i.e., data or non-data) and the priority encoder is enabled by the absence of a bus request (INHSDR+ is low), the high order output (CREDRQ-) indicates the type of request active. The two low order outputs (CREBA0-, 1-) represent a binary decode of the requesting device. Since the address bits (MYAD15+, 16+) from the bus data register are high due to firmware continually clearing the BDR, the SPMIR is loaded with the requesting device number.

For a bus request, the enable input (INHSDR+) of the decoder is high and forces all the priority encoder outputs high, ignoring the inputs. The request active signal (CREREQ+) is set by INHSDR-. When the firmware determines that a request is active (CREREQ+), the bus request is detected by examination of INHSDR+. With the priority encoder outputs high, the SPMIR is loaded with the device selection bits (MYAD15+, 16+) from the BDR.

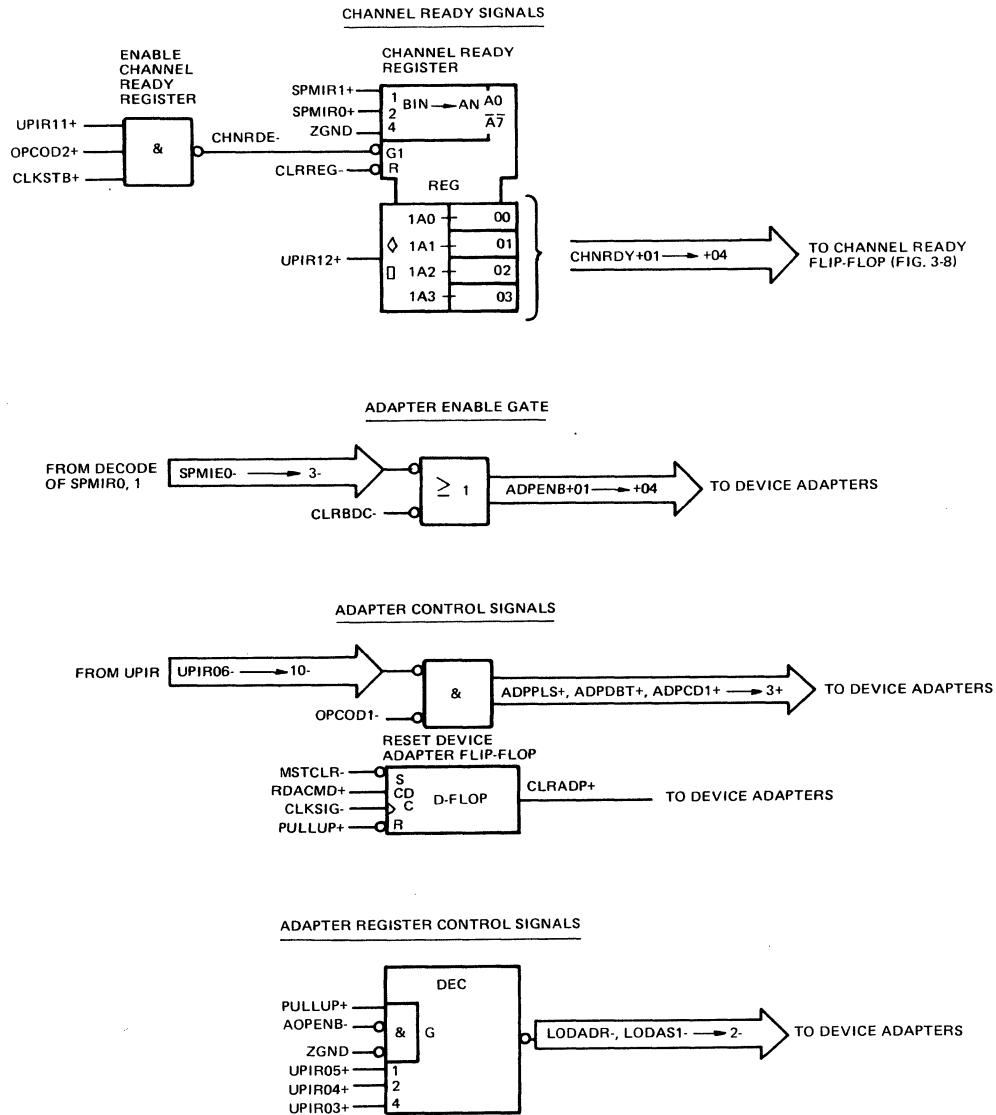


Figure 3-9 Channel Controls (Sheet 1 of 2)

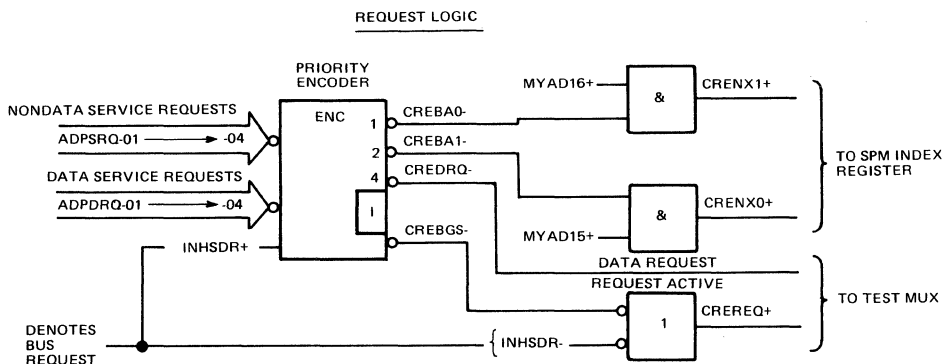


Figure 3-9 Channel Controls (Sheet 2 of 2)

3.6 TEST MULTIPLEXER FUNCTIONAL DESCRIPTION (Figure 3-10)

Firmware intelligence is derived from the capability of testing data, status, and errors within the subsystem. Test commands specify a test item and a test condition. The next sequential command is skipped if the state of the test item equals the specified test condition.

All test items specified by a 5-bit field go to one of the test multiplexers whose outputs are TSTXB0+ through TSTXB3+. These multiplexers use the three low-order bits of the UPIR 5-bit field to select one of eight inputs. The high-order two bits of the test item field select the test item inputted to the multiplexer (TSTMUX). The state of the test item, as defined by the output of TSTMUX, is compared against the test condition, and the Test Valid (TSTVLD+) signal goes high if the comparison is true. A valid test is recorded in the clear microprogram instruction register (CLRUPI) flip-flop, which clears the UPIR during the next clock pulse (performs a NOP). If the test is not valid, the UPIR is not cleared, and the next sequential instruction is executed normally.

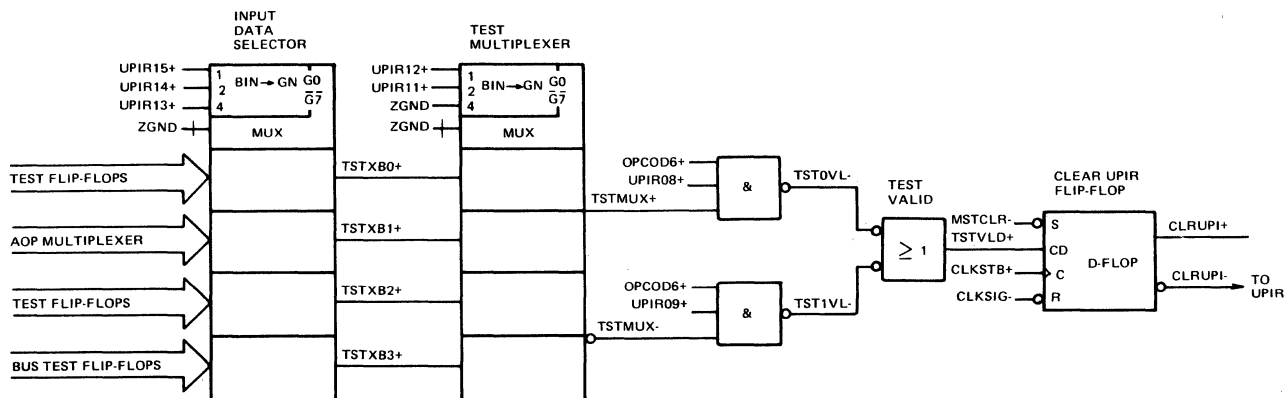


Figure 3-10 Test Multiplexer

3.7 CLOCK AND CLOCK CONTROL FUNCTIONAL DESCRIPTION (Figure 3-11)

The output of an 8 MHz crystal oscillator (CLKOSC+) is divided to obtain a 250 ns clock cycle. Two clock signals (CLKSIG+) and a clock strobe (CLKSTB+), developed during each clock cycle, control the MDC and adapter hardware.

The clock signal (CLKSIG) flip-flop is a J-K flip-flop which changes state at the negative edge of the CLKOSC+00 signal as long as the inputs are high. When the clock halt (CLKHLT) flip-flop is set, the J-input goes low, causing the clock signal to be reset or to remain reset. The assertion and negation of the CLKSIG flip-flop are inverted for distribution throughout the MDC. The negation clock signal, after inversion, is fed to a 100 ns delay line. The 40 ns tap (CLKDLY+03) and the 90 ns tap (CLKDLY+08) generate strobe CLKSTB+, which occurs during the later part of each clock cycle. The clock, a minimum of 35 ns duration, occurs at a point in the cycle which will meet data setup and hold times sufficient for writing into the scratchpad memory. CLKSTB also clocks registers within the MDC and device adapters.

The clock halt (CLKHLT) flip-flop can be set when:

1. Detecting a critical error and stopping the MDC
2. Entering the test mode
3. Detecting a microprogram control store error.

The CLKHLT flip-flop is set by execution of a firmware Halt command (HLTCMD). In the test mode the Halt condition is recirculated by the signal RECHLT- until a Master Clear signal occurs. When the test mode (TSTMOD) flip-flop is on, the RECHLT- function goes inactive each time the MDC acknowledges a Megabus cycle. The next transition of the oscillator (CLKOSC) causes the CLKHLT flip-flop to be reset, putting a One at the J-input to the clock signal (CLKSIG) flip-flop, which will be set at the second transition of CLKOSC+. CLKSIG+ feeds back to the CLKHLT flip-flop, causing it to be set during the third transition of CLKOSC-. In this manner one clock cycle is generated in the test mode for each Megabus transfer to the MDC.

The TSTMOD flip-flop is set by the firmware command Set Test Mode (STMCMD) and remains set due to the Recirculate Test Mode (RECMOD) until a Master Clear signal occurs or a command is received on the Megabus which resets the test mode and the clock halt flip-flops. In addition to controlling the stepping of the clock, the test mode flip-flop controls the input to the UPIR. When set, TSTMOD disables the microprogram control store input to the UPIR and enables the test mode input. (Refer to subsection 3.1.5 of this manual.)

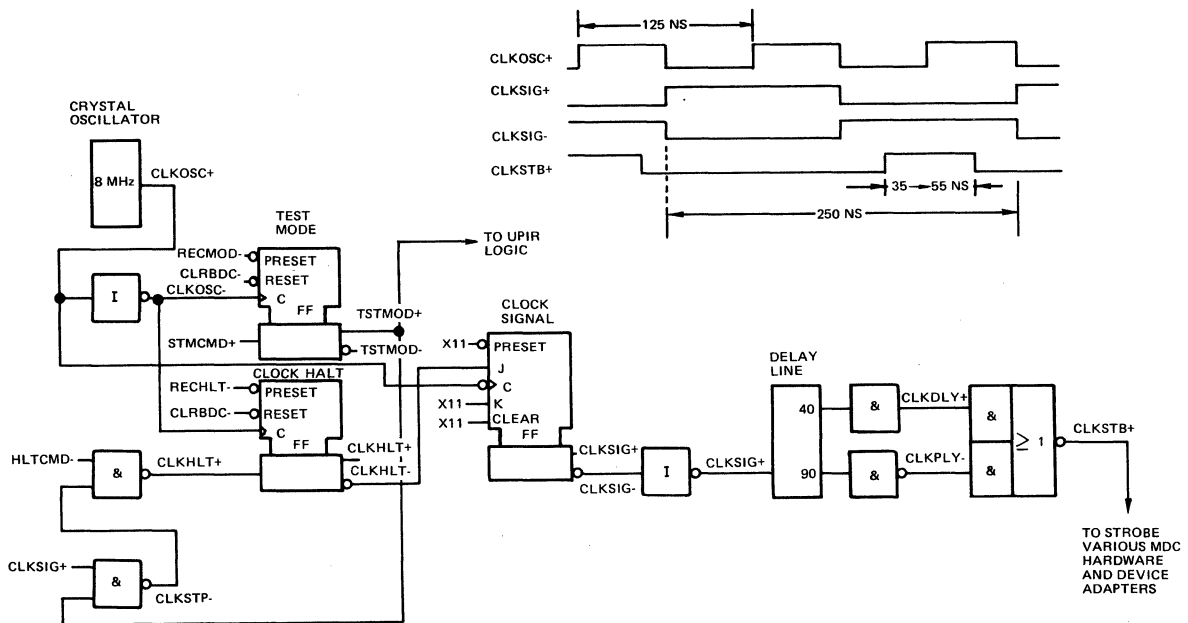


Figure 3-11 Clock Logic

3.8 DATA FLOW AND CONTROL PATHS

The following subsections discuss the data and control paths illustrated in Figure 3-1. During the description of the specific hardware and firmware operations necessary to execute an output and input operation, references are made to the intermediate block diagrams. The basic hardware application is primarily the same for both operations (output and input) with the firmware acting as the determining factor as to how the transfers will differ. The discussion commences with initiation of the operation by software, followed by operation execution through firmware control of hardware implementation and termination of data transfer.

3.8.1 Initiation of Output Operation

Software initiates an output data operation by transferring control information from the CPU to the MDC via the Megabus, using I/O commands. The MDC Megabus logic determines if the operation is destined for the MDC (see Figure 3-8). This is accomplished by continuously examining the Megabus address bits 8 through 14 (BSAD08+ through 14+) and comparing them to the settings of the hex rotary switches on the MDC. When there is an equal compare between the address bits and the switch settings, and a bus request (BUSREQT+) is present, the MDC hardware generates the proper response. This response is an acknowledge (MYACKR+) provided the BDR is not busy and the device, selected by the device selection bits (BSAD15+ and 16+), is available (CHNRDY-20).

The firmware periodically scans the request logic (see Figure 3-9) to determine which one of the three types of requests (bus request, data service request, or nondata service request) is active. Upon detecting a bus request (INHSDR+) with the MDC generating an ACK response (MYACKR+), the BDR (see Figure 3-6) is inhibited from shifting, then parallel loaded (LODBDR+) with the three bytes of address (BSAD00+ through 23+) and two bytes of data (BSDT00+ through 15+) from the Megabus and set busy by the firmware. The firmware shifts the BDR (SFTBDR+), allowing a byte (MYAD16+ through 23+) representative of the least significant byte (LSB) of address (BSAD16+ through 23+) to be transferred through the AOP multiplexer and stored in a relative location of the SPM (see Figure 3-5). This SPM relative address, CHN1 (for SPM topology refer to Table 4-1), is developed by the low order six bits of the SPMAC (SPMAC2+ through 7+), which represent a constant supplied by firmware, and the high order two bits (SPMAS0+ and 1+), which are provided by the SPMIR (SPMIR0+ and 1+). The high order two bits represent the device selection bits (CRENX0+ and 1+) of the channel address. The SPM location CHN1 contains the output type function code (see Table 2-1), which is used in conjunction with the SPMIR, to develop the proper relative address in the SPM, for storing the two bytes of data from the BDR. As each of the two data bytes (MYDT00+ through 07+ and MYDT08+ through 15+) is shifted and stored in the SPM, a parity check (BSPYCK+) is performed (see Figure 3-6).

The total number of output information transfers from the Megabus to the SPM depends upon the device type. After the two bytes from the BDR data lines have been loaded in the SPM, firmware

examines the function code to see if it is an I/O load output address function code (see Table 2-1). When this type of function code is detected, the direction bit (see Figure 2-1) is examined and should be set to One, indicating an output data operation (write to device). As each transfer is received across the Megabus, the firmware examines the function code to determine if it is the startup function code, indicating the last bus transfer has taken place and the output data operation is ready to be executed.

3.8.2 Output Operation Execution

The firmware maintains control of the output data operation. When the request logic (see Figure 3-9) receives a data service request (ADPDRQ-) from the selected device, a data request (CREDRQ-) is generated, provided a bus request (INHSDR+) is not active. The firmware recognizes the data service request and examines the Megabus priority network (see Figure 3-7) to see that no other unit with higher priority (BSAUOK+ through BSIUOK+) is using the Megabus. The signal BSMYOK+ is generated and sent to the Megabus to inhibit access to the Megabus by other units, while the MDC is using it.

The firmware sets the BDR busy status indicator flip-flop (see Figure 3-6) and loads the BDR five times serially, one byte at a time from the AOP mux (ALUAX0+ through 7+). The Megabus is parallel loaded from the BDR with the five bytes configured as a read main memory request format (see Figure 2-2). The firmware increments the main memory address and decrements the range, checking for the end of range to indicate the last data transfer from the main memory.

The firmware loads the cycle parameters (see Figure 3-7), memory reference (MYMREF+), response required (MYRSVP+), second half read cycle (MYSHRC+), and byte mode (MYBYTE+) from the AOP mux (ALUAX1+ through 4+) to the Megabus by issuing a Cycle command (CYCCMD+), which also initiates a bus cycle. During the second half read cycle, the main memory loads the Megabus with the data requested and responds with an acknowledge to the MDC (ACKRSP+) to clock (see Figure 3-7) the second half read history flip-flop (MYSHRH±). The assertion output (MYSHRH+) is used to develop an acknowledge cycle (ACKCYC-), which generates cycle response (SETACK+) in the slave response logic (see Figure 3-8) for the MDC's acknowledge response (MYACKR+). The negation output (MYSHRH-) is used by the BDR logic (see Figure 3-6) to parallel load the BDR (LODBDR+) with the five bytes of information on the Megabus from main memory. The firmware shifts the BDR twice, ignoring the two bytes of Megabus address (BSAD08+ through 23+) and presents the LSB of data (BSDT08+ through 15+) at the BDR output (MYAD16+ through 23+). The LSB of data is checked for the correct parity and transferred through the AOP mux for storage in the SPM relative location DTAL. The BDR is shifted again, with the MSB of data (BSDT00+ through 7+) at its output, which is checked for correct parity and loaded into the ACU via the AOP mux (see Figure 3-4). Firmware presents the MSB of data to the device from the ACU (ALUOT0+ through 7+) via the MDC/device adapter data interface lines. The

LSB of data is transferred to the device by firmware unloading SPM relative location DTA1 into the ACU via the AOP mux and presenting the data byte over the same data interface lines (ALUOT0+ through 7+). In an output data operation, the firmware is capable of requesting one or two bytes of data during each read main memory request cycle. The entire data transfer operation is repeated until the end of range is detected by firmware.

3.8.3 Termination of Output Operation

When the firmware detects the end of range, it issues a Shift Data Register command (SDRCMD-) five times, which serially loads the BDR (see Figure 3-6) with five bytes of information configured to an Input Interrupt Control format (see Figure 2-8) for an MDC-generated interrupt. The firmware parallel loads the Megabus with this format, loads the cycle parameters (see Figure 3-7), and when it determines that no other unit with higher priority has access to the Megabus, it initiates (CYCCMD+) a bus cycle. When the CPU acknowledges (BSACKR+) the information sent to it by the MDC, the firmware issues a Set Channel Ready command to enable the Channel Ready Register (see Figure 3-9) and allow the Channel Ready (CHNRDY+) signal to be set, indicating it is now available. This completes the operation, and the MDC is now ready to accept another task.

3.8.4 Initiation of Input Operation

The software initiation of an input data operation is similar to the initiation of an output operation (see subsection 3.8.1). The main difference is that when firmware examines the direction bit (see Figure 2-1) of the I/O load output address function code (see Table 2-1), it should now be a Zero, indicating an input data operation (read from the device).

3.8.5 Input Operation Execution

The firmware maintains control of the input data operation. When the request logic (see Figure 3-9) receives a data service request (ADPDRQ-) from the selected device, a data request (CREDRQ-) is generated, provided a bus request (INHSDR+) is not present. The firmware recognizes the data service request and loads the data (see Figure 3-4) from the device adapter (ADPDS0+ through 7+) through the AOP mux (ALUAX0+ through 7+), the ALU (ALUOT0+ through 7+), the ACU (ALUAC0+ through 7+) and back (see Figure 3-5) to the AOP mux for storage in the SPM (SPMOT0+ through 7+).

The firmware examines the Megabus priority network (see Figure 3-7) to see that no other unit with higher priority (BSAUOK+ through BSIUOK+) is using the Megabus. The signal BSMYOK+ is generated and sent to the Megabus to inhibit access to the Megabus by other units while the MDC is using it. The BDR is set busy (see Figure 3-6) and loaded five times serially, one byte at a time, from the SPM through the AOP MUX (ALUAX0+ through 7+). The Megabus is parallel loaded from the BDR with the five bytes of information configured

as a write main memory request format (see Figure 2-4). The firmware, which can write one or two bytes of data into the main memory, increments the address and decrements the range, checking for the end of range to indicate the last data transfer to the main memory.

Firmware loads the cycle parameters (see Figure 3-7) from the AOP mux (ALUAX1+ through 4+) to the Megabus by issuing a Cycle command (CYCCMD+), which also initiates a bus cycle. After the main memory unloads the data from the Megabus, it responds with an acknowledgement to the MDC (BSACKR+). The firmware repeats the entire data transfer operation until it detects the end of range.

3.8.6 Termination of Input Operation

The termination of an input data operation is identical to the termination of an output operation (see subsection 3.8.3).

IV THEORY OF OPERATION - CYCLE FLOW

4.1 MICROINSTRUCTIONS

A microinstruction and its purpose are defined by the bit structures within a firmware word (refer to subsection 2.4.3 of this manual). A combination of one or more microinstructions is collectively referred to as a firmware command (see Figure 4-1). Sequences of firmware commands utilized to perform a particular function are known as microprograms or firmware routines. The MDC has eight types of firmware commands, each of which is subdivided into various bit configurations (microinstructions) to perform a specific operation. The eight types of firmware commands are:

- Miscellaneous
- Device Adapter
- Megabus Logic
- ALU
- Constant
- Memory
- Test
- Branch

Each of the major categories of firmware commands is identified by a particular op code, the coding of bits 0, 1, and 2 of the microprogram control store word.

4.1.1 Miscellaneous Commands

Miscellaneous commands, which have an op code of 000, are used primarily to perform clear and set operations on registers and flip-flops in the MDC and the device adapters. These commands are

comprised, typically, of a single microinstruction which is specified by bits 3-15 of the microprogram control stored word. Table A, Figure 4-2, lists the microinstructions within the miscellaneous category, their mnemonics, and the binary and hexadecimal values for each word.

4.1.2 Device Adapter Commands

Device adapter commands have an op code of 001 and are used to generate control signals going from the MDC to the device adapter. The device adapter enabled as a result of the scratch pad memory index register (refer to subsection 3.3.2) is the only one of the four possible adapters which reacts to and executes the command.

Device adapter commands use microinstructions which utilize the ALU A-operand multiplexer in conjunction with device-specific microinstructions to develop a complete firmware command. Table B, Figure 4-2, is a listing of the device adapter commands, their mnemonics, and the hexadecimal and binary values for each word.

4.1.3 Megabus Logic Commands

Megabus logic commands have an op code of 010 and are a result of a combination of microinstructions particular to the bus command and ALU A-operand microinstructions. The resulting firmware word is utilized to perform control functions on hardware associated with the Series 60 Level 6 Megabus. Table C, Figure 4-2, lists the Megabus logic commands, their mnemonics, and the hexadecimal and binary values for each word.

4.1.4 ALU Commands

A-operand microinstructions, B-operand microinstructions, and microinstructions specific to the ALU are used to fully define one ALU command. ALU commands, having an op code of 011, perform specific logic or arithmetic operations on the contents of the A- and/or B-operand registers. The result is then stored in the A-operand or B-operand as determined by the ALU microinstructions. Mode, carry enable, and carry in are also specified by the ALU microinstructions. Table D, Figure 4-2, is a list of the microinstructions and their mnemonics. The A-operand and B-operand selection fields (see Tables E and F, Figure 4-2) are also shown, and the binary value is given for those bits pertaining to the particular command.

4.1.5 Constant Commands

Constant commands have an op code of 100 and are comprised of a combination of ALU A-operand microinstructions and two types of constant microinstructions. The two types of constant microinstructions specify a data constant to be used and the type of ALU operation to be performed. Table G, Figure 4-2, lists the types of Constant commands, their mnemonics, and their binary bit configurations.

4.1.6 Memory Commands

Memory commands are used to write into the scratch pad memory and to control or alter the scratch pad address. The memory commands have an op code of 101 and are combinations of ALU A-operand microinstructions and specific memory microinstructions. Table H, Figure 4-2, lists the memory commands, their binary and hexadecimal values, and their mnemonics.

4.1.7 Test and Return Commands

Test commands (op code 110) allow firmware to determine the state of certain hardware elements in the MDC and device adapters. The parameters that the firmware is capable of testing, their mnemonics, and their hexadecimal values are listed in Table K, Figure 4-2.

Return commands have the same op code (110) as test commands and allow firmware to load the microprogram address counter with an address that was previously stored in a subroutine return address register. Table J, Figure 4-2, lists test and return commands, their binary and hexadecimal values, and their mnemonics.

4.1.8 Branch Commands

Branch commands have an op code of 111 and are utilized to load the microprogram address counter or the index subroutine return address register with an address within the firmware word. Table L, Figure 4-2, lists the branch commands, their binary and hexadecimal values, and their mnemonics.

4.2 SCRATCH PAD MEMORY (SPM)

The MDC firmware cycle flow can be more easily understood if it is remembered that the scratch pad memory is divided into four quadrants, with one quadrant dedicated to a particular channel. A quadrant has 64 addressable locations, each of which stores 8-bit bytes of data or control information. The topology of a single quadrant of the SPM is shown in Table 4-1, which defines the relative address (hex), the mnemonic, and the contents for each of the 64 locations. The four quadrants are organized in the same manner. When a location of the SPM is unique to an adapter or is specifically for software usage, the byte is described in the appropriate manual.

Information stored as a 16-bit word requires two SPM locations, one for the most significant byte (MSB), bits 0 through 7, and one for the least significant byte (LSB), bits 8 through 15.

Table 4-2 shows the control words which have specific significance for the MDC firmware, and Figures 4-3 through 4-6 indicate the bit structure of the bytes unique to the MDC.

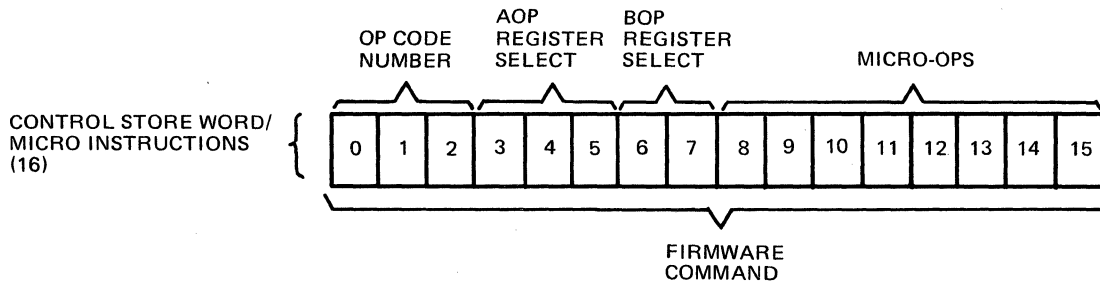


Figure 4-1 Microinstruction Field Format

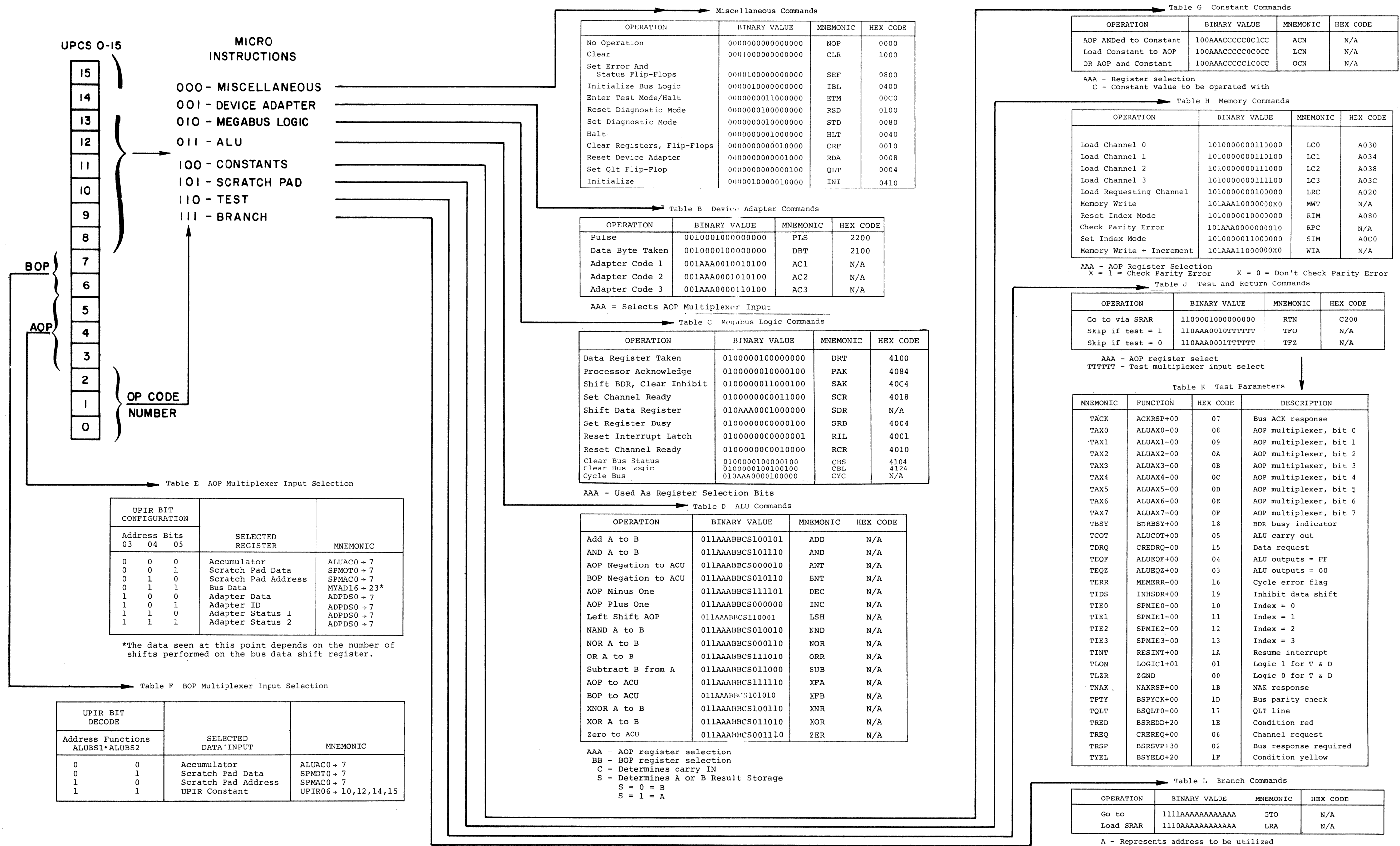


Figure 4-2 MDC Firmware Flow

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 4-1 Single Scratch Pad Memory Quadrant Topology

RELATIVE ADDRESS (Hex)	MNEMONIC	CONTENTS
00	CWD1	Control Word, LSB
01	CWD2	Control Word, MSB
02	ILC1	Interrupt Level, LSB
03	ILC2	Interrupt Level, MSB
04	SFC1	Startup Function Code
05	SWRV**	Firmware Revision
06	TSK1*	Task, LSB
07	TSK2*	Task, MSB
08	ADR1	Address, LSB
09	ADR2	Address, MSB
0A	MOD1	Module Address
0B	ENQB	Enqueue Buffer
0C	RNG1	Range, LSB
0D	RNG2	Range, MSB
0E	Spare	
0F	Spare	
10	CNF1*	Configuration Word 1, LSB
11	CNF2*	Configuration Word 2, MSB
12	CNF3*	Configuration Word 3, LSB
13	CNF4*	Configuration Word 4, MSB
14	Spare	
15	Spare	
16	Spare	
17	Spare	
18	STS1*	Status Word One, LSB
19	STS2*	Status Word One, MSB
1A	STS3*	Status Word Two, LSB
1B	STS4*	Status Word Two, MSB
1C	POLQ*	Poll Queue
1D	POLP*	Poll Pointer
1E	TSKQ*	Task Queue
1F	TSKP*	Task Pointer
20	DTA1	Data, LSB
21	DTA2	Data, MSB
22	CMSK	Channel Mask
23	Spare	
24	MON1	Channel Monitor
25	DM1	DMA Control
26	DID1*	Device ID, LSB
27	DID2*	Device ID, MSB
28	CHN1	Channel Number, LSB
29	CHN2	Channel Number, MSB
2A	CPC1	CP Address, LSB
2B	CPC2	CP Address, MSB
2C	IDF1	Interrupt Vector, LSB
2D	IDF2	Interrupt Vector, MSB
2E	WL01*	Work Location 1
↓	↓	↓
3C	WL15*	Work Location 15
3D	RICP	Resume Interrupt Control
3E	TMW1**	Test Mode Work LOC1
3F	TMW2**	Test Mode Work LOC2
CB***	QLTI	Quality Logic Test Pass/Fail Flag

*Device-specific locations
 **Utilized by software only
 ***Absolute location

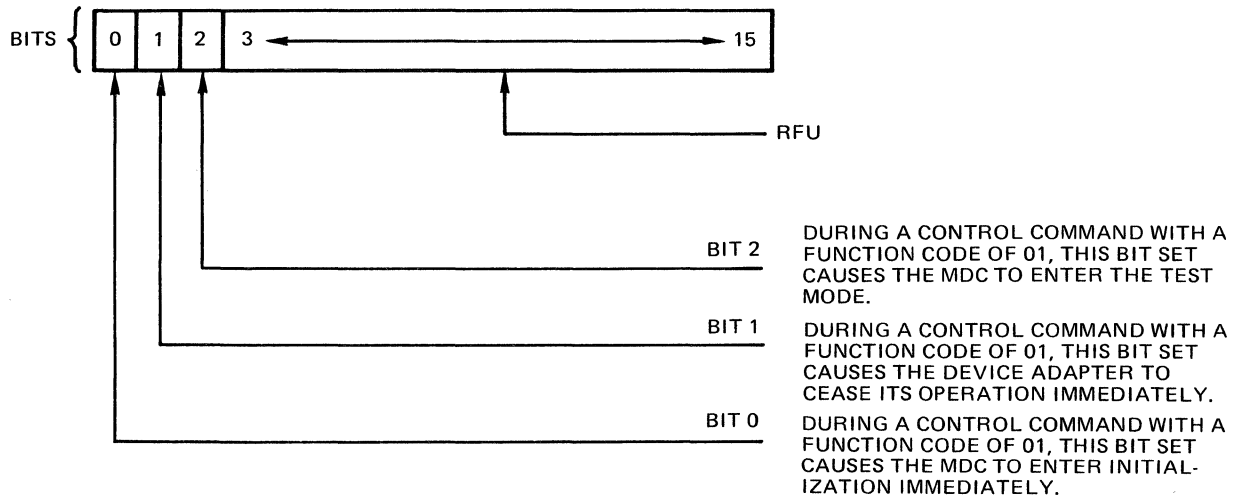


Figure 4-3 Control Word Bit Significance

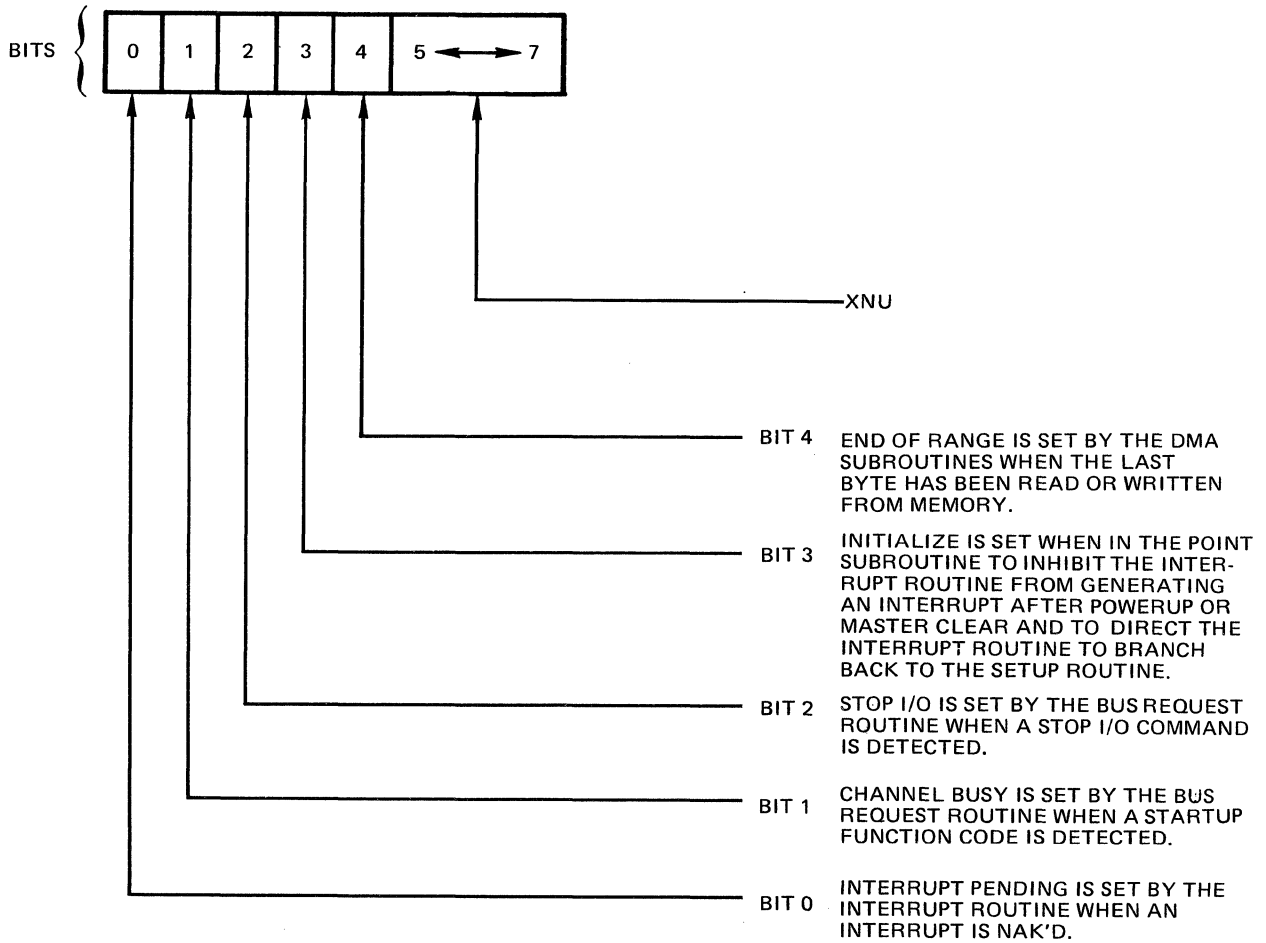


Figure 4-4 Channel Monitor Byte Bit Structure

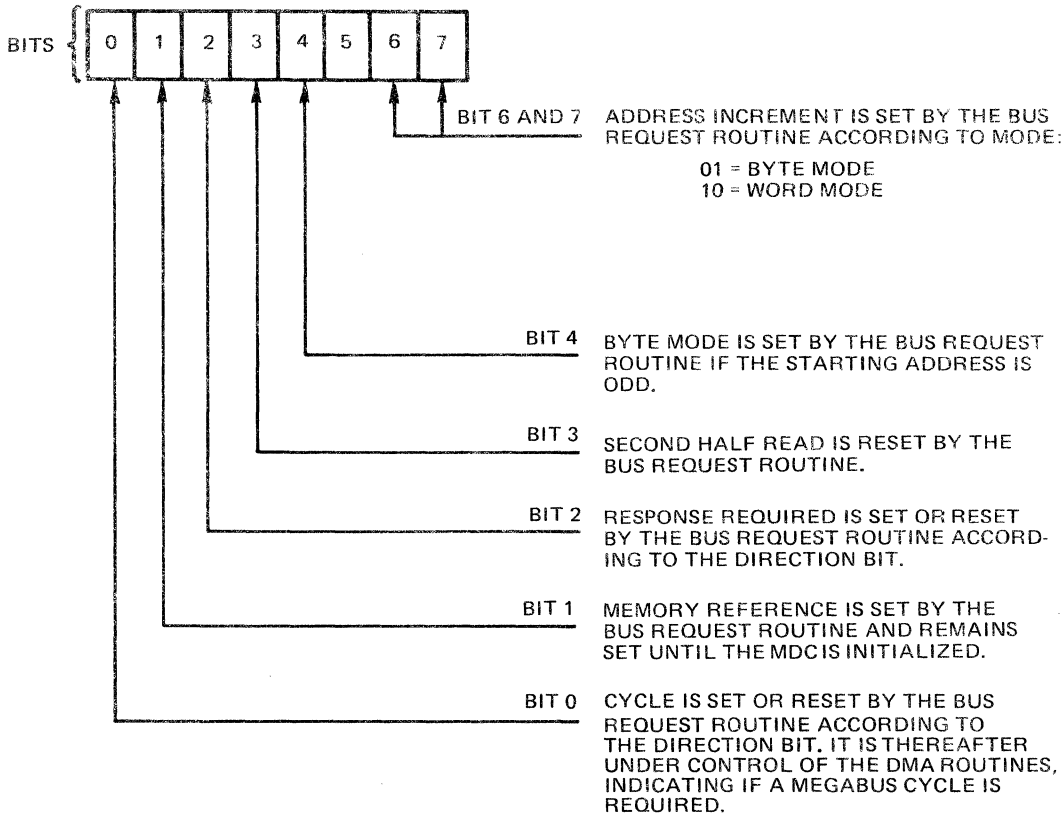


Figure 4-5 DMA Flag Byte Bit Structure

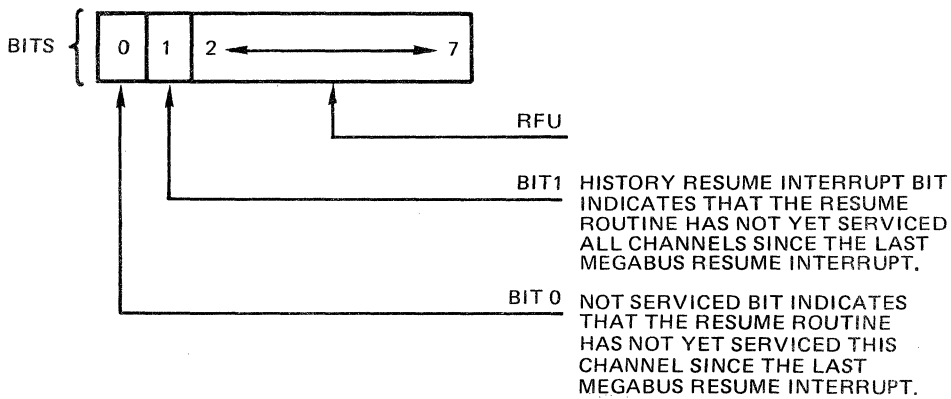


Figure 4-6 Resume Interrupt Control Parameter Byte Bit Structure

4.3 FLOW CHARTS

The firmware flow charts in this manual, which are generated at an intermediate level, are designed more as an aid in understanding firmware routine functions than as a maintenance aid. The symbols used in the flow charts are described and illustrated in subsection 4.3.1. Subsection 4.3.2 describes the flow chart organization. Information concerning the use of these flow charts and a flow chart example are detailed in subsection 4.3.3.

4.3.1 Flow Chart Symbolology

The flow charts are a composite of symbols each of which has a specific meaning application. In some instances the same symbol can have more than one meaning, depending upon its usage. A description and illustration of each flow chart symbol are found in Table 4-4.

Table 4-2 Scratch Pad Memory Word Description (Sheet 1 of 2)

MNEMONIC	TERM	DESCRIPTION
CWD1 and CWD2	Control Word	A bit-specific control word which specifies immediate channel operations. (See Figure 4-3.)
ILC1 and ILC2	Interrupt Level	The interrupt level word consists of 2 bytes, with bits 0 → 9 containing the CP's channel number. Bits 10 → 15 represent a binary value of from 0 to 63, indicating the priority of the MDC's interrupt to the CP.
SFC1	Startup Function Code	The start function code is an 07 for the diskette and an 0D for all other devices. It is used to compare with the function code from the Megabus to determine if the command requires initiation of device action.
ADR1 and ADR2	Address	This word is the main memory address sent across the Megabus to access information.
MOD1	Module Address	This byte is part of the main memory address and is used by main memory to select a 64K module.
RNG1 and RNG2	Range	The range is utilized to determine the number of data bytes to be transferred across the Megabus. This word is decremented for each byte transferred until it reaches zero.
DAT1 and DAT2	Data	These two byte locations are used for temporary storage of data to or from the device.
MON1	Channel Monitor	See Figure 4-4 for a bit-specific usage of the channel monitor byte.
DMA1	DMA Control	See Figure 4-5 for a bit-specific usage of the data management control byte.

Table 4-2 Scratch Pad Memory Word Description (Sheet 2 of 2)

MNEMONIC	TERM	DESCRIPTION
DID2	Device ID (MSB)	This is the MSB of the device identification code. It is supplied by the MDC firmware and is always a 20. The LSB of the ID code is supplied by the adapter. (See Table 4-3.)
CHN1 and CHN2	Channel Number	These two bytes supply the unique channel number of a particular MDC.
CPC1 and CPC2	CP Address	The CP address is utilized by the MDC for loading the address lines of the BIR whenever a request for data transfer to the CP occurs.
IDF1 and IDF2	Interrupt Vector	The interrupt vector word consists of two bytes, with bits 0 → 9 containing the MDC channel number. Bits 10 → 15 represent a binary value of from 0 → 63, indicating the priority of the MDC interrupt to the CP.
RICP	Resume Interrupt Control	See Figure 4-6 for a bit-specific usage of the resume interrupt control parameter byte.
ENQB	Enqueue Buffer	The enqueue buffer has one location and is accessed only in the nonindexed mode. The buffer is used as storage for an indicator byte where bits 0, 1, 2 and 3 are the 'stalled' indicator bits of the respective channels which have read/write orders pending.
CMSK	Channel Mask	The channel mask is an identifier byte stored for each channel. The mask allows a flag of a specific channel number to be set by an ORing of the channel mask with another parameter byte. The channel numbers are represented with the byte by the four MSB of the byte. These bits refer to channel numbers 0, 1, 2 and 3 respectively.
QLTI	Quality Logic Test Flag	The Quality Logic Test flag is set to 00 if the Quality Logic Test fails or to FF if the Quality Logic Test is successful.

Table 4-3 MDC Device Identification Codes

PERMISSIBLE RANGE OF NUMBERS (Hex)	DEVICE
2000 → 2007	Line Printer/Serial Printer
2008 → 200F	Card Reader
2010 → 2017	Diskette
2018 → 201F	Console

Table 4-4 Cycle Flow Chart Symbols (Sheet 1 of 6)

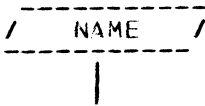
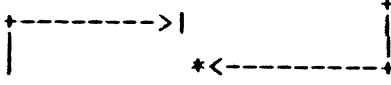

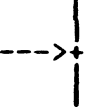

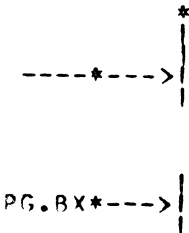


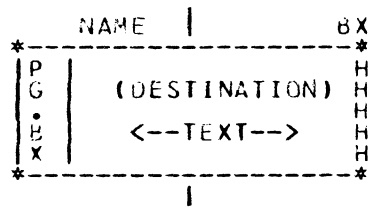
OPERATION	SYMBOL	DESCRIPTION
Initial Connector		This symbol always begins a new flow chart "cluster." The name is taken from the next symbol. A flow chart cluster is the flow path tree created during the chart drawing process.
Direct Connectors	<p>HORIZONTAL CONNECTORS</p>  <p>VERTICAL CONNECTORS</p>  <p>'T' CONNECTORS</p>  <p>CROSS CONNECTORS</p> 	These symbols are examples of mechanical representation of hand-drawn flow lines. Such lines can cross with or without connection. A tied cross point is indicated by a + symbol.

Table 4-4 Cycle Flow Chart Symbols (Sheet 2 of 6)

OPERATION	SYMBOL	DESCRIPTION
Back-referencing Connectors		<p>The * entry in a flow path line indicates that more than one reference exists at that point of entry. Coordinate information is provided from remote branch points. Only the first such reference is given. The absence of an * indicates that the branch path is unique; that is, no 'FAN-IN' of logical flow is indicated.</p>
Intermediate Connector		<p>This symbol is generated whenever the flow path is continued in the near vicinity. It arises due to physical limitations of page size.</p>
Process		<p>The symbol shows in-line computational processing. Box depth is determined by extent of the enclosed comment.</p>
Subroutine Call		<p>A subroutine is given control. Return is assumed to be in-line. Box encloses a destination name and coordinate reference which appear vertically on the left as PG.BX.</p>

HONEYWELL PROPRIETARY AND CONFIDENTIAL

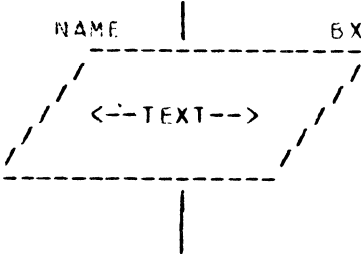
Table 4-4 Cycle Flow Chart Symbols (Sheet 4 of 6)

OPERATION	SYMBOL	DESCRIPTION
EXIT	<pre> NAME BX ----- * EXIT * ----- <--TEXT--> </pre>	<p>This symbol shows an exit flow path terminator. Accompanying text appears immediately below the symbol.</p>
EXIT or Continue	<pre> NAME BX ----- * EXIT * ----- <--TEXT--> </pre>	<p>This construction is used for EXIT instruction sequences which are executed conditionally, depending on processing elsewhere in the flow chart.</p>
HALT	<pre> NAME BX ----- * HALT * ----- <--TEXT--> </pre>	<p>This symbol shows a HALT flow path terminator. Accompanying text appears immediately below the symbol.</p>
HALT, Continue	<pre> NAME BX ----- * PAUSE * ----- <--TEXT--> </pre>	<p>This symbol represents a HALT without flow path termination. Accompanying text appears immediately below the symbol and the flow path continues downward.</p>
NOTE	<pre> NAME NOTE BX * * * * * * * * * * * * <--TEXT--> * * * * * * * * * * * * * * * * </pre>	<p>A NOTE is a descriptive or explanatory notation embedded directly in the flow path.</p>

Table 4-4 Cycle Flow Chart Symbols (Sheet 5 of 6)

OPERATION	SYMBOL	DESCRIPTION
Decision		<p>The diamond shape is the basic decision symbol; it is fixed in size. Whenever possible, any text is fitted within the box. If the text does not fit, it appears in a note which immediately precedes the diamond. SEE NOTE ABOVE appears as the text within the diamond. Notice that the left-hand path continues to another symbol not illustrated here. The decision symbol may involve two or three possible branch paths, each having a distinct label consisting of up to five characters. The in-line label continues directly from the bottom of the symbol to the very next symbol. All other labels are called out-of-line. A decision connector is generated whenever directly connecting flow paths cannot be drawn.</p>
Decision Connector		<p>This fixed-length symbol is generated for cases in which an out-of-line path cannot be directly connected by a line on the same chart page.</p>

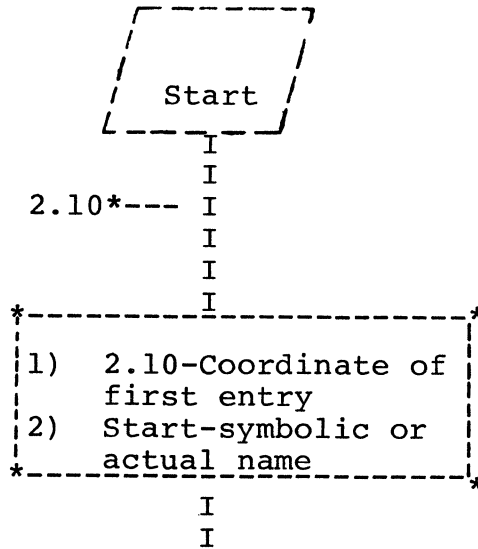
Table 4-4 Cycle Flow Chart Symbols (Sheet 6 of 6)

OPERATION	SYMBOL	DESCRIPTION
Input/Output		<p>This is the distinctive fixed size I/O symbol. If accompanying text cannot be completely contained within this symbol, it will appear as a note immediately preceding the I/O parallelogram, as previously described for the decision symbol with excessive text.</p> <p style="text-align: center;">NOTE</p> <p>This symbol is only used for descriptive purposes in the firmware flow charts.</p>

4.3.2 Flow Chart Organization

The symbols on each flow chart sheet are numbered sequentially, starting with the number 01. The number 01 is assigned to the first symbol in the leftmost column and appears outside and at the upper right-hand corner of the symbol. In addition, each sheet is assigned a number, starting with 01. The numbering of the symbols begins again with 01 for the first symbol of the new sheet. Therefore, any symbol can be identified and located by citing its number and the number of the sheet on which it appears. This information is referred to as the coordinate of the symbol and is expressed in the format PG.BX, where PG is the sheet number and BX is the symbol number. For example, a symbol whose coordinate is 03.07 is the seventh symbol on sheet 3 of the flow chart.

In some instances due to the flow chart layout, a connecting line cannot be drawn from one symbol to another. When this situation arises, a coordinate at the symbol specifies the destination and indicates that entry is from another portion of the flow chart. In the following example, the coordinate 2.10 indicates that there is a logical connection between this symbol and the tenth symbol on sheet 2. This application of a coordinate is called an in-connector. The asterisk (*) following the coordinate indicates an entry to this symbol from some other point in the flow chart.



The preceding example of an in-connector also illustrates that any flow chart symbol can be assigned a name. This name can represent up to ten digits of a firmware address or tag associated with a particular line of the firmware file, or it can represent a symbolic address for use as the destination of a branch operation. Regardless of the use of the name, it is located outside the upper left-hand corner of the symbol.

4.3.3 Usage

Figure 4-8 is an illustration of an intermediate flow chart utilizing one of the MDC firmware routines as an example. This flow chart contains a description of the overall operations being performed by one or more firmware commands being executed. With one exception, only the name (or address) associated with the addresses or tags located in the firmware listing will be shown outside the upper left-hand corner of the symbol. The exception to this is that if a symbolic destination tag (name) is required for a test or branch operation, the tag will not appear in the firmware listing.

Since the operations performed in one or more cycles may be no-effect operations (i.e., operations which are not reflected in the overall procedure being performed), only those commands necessary to justify an end result will be incorporated into the flow chart.

4.4 FIRMWARE CYCLE FLOW

The MDC firmware is divided into nine functional areas or routines. Figure 4-7, an overview flow chart of the firmware routines associated with the MDC, indicates the names, interconnecting paths between routines, and the exit paths to the specific device adapter support routines.

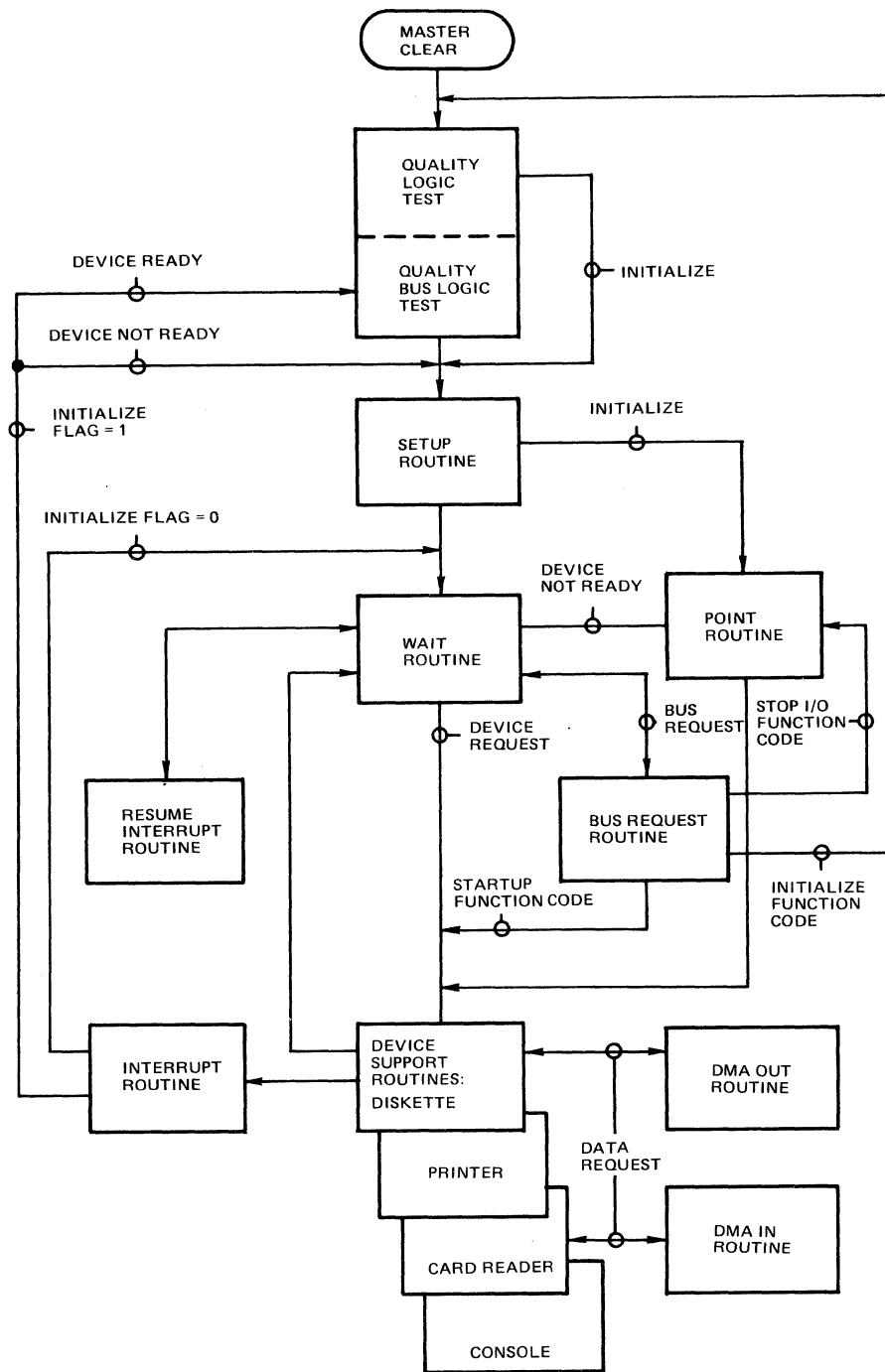


Figure 4-7 Firmware Overview Flow Chart

4.4.1 Quality Logic Test (Figure 4-8)

The Quality Logic Test (QLT) (Figure 4-8, sheets 1 and 2) is a firmware routine that functionally verifies major logic components of the MDC to determine operational capabilities. This routine is for verification of the MDC only and is not utilized to determine if the device adapters are functional.

The Quality Logic Test Bus Subroutine (QLT-Bus) portion of the QLT verifies the MDC bus logic (see Figure 4-8, sheet 3). It is used only for those channels which have a recognizable device identification code (see Table 4-3) and a device in the ready state. Entry to the QLT-Bus subroutine is from the Interrupt routine after the device support firmware has completed the device-specific initialization sequence.

The Quality Logic Test is executed as a result of a Master Clear or the initialize bit being set in an output control word which resets the QLT done flip-flop. This results in the forcing of the microprogram address counter to zero and illumination of the red LED located on the front edge of the MDC module. Upon successful completion of the test, firmware sets the QLT done flip-flop and extinguishes the LED.

4.4.2 Setup Routine (Figure 4-9)

At the completion of the QLT, the Setup routine (SETUP) is entered. The first portion of the Setup routine initializes scratch pad memory and is utilized by the QLT as a subroutine for zeroing the SPM.

The subsequent segment of the Setup routine enables each device adapter independently and branches to the Point routine. Upon execution of the Point routine, the Device Support routine, and the Interrupt routine, the Setup routine is reentered and enables the next sequential device adapter. After each adapter has been enabled and set up, the Setup routine branches to the Wait routine.

4.4.3 Wait Routine (Figure 4-10)

The Wait routine (WAIT) tests the channel request priority encoder and resume interrupt flip-flop to determine what, if any, firmware action is required.

If a channel request is active, the Wait routine loads the scratch pad memory index register with the number of the requesting channel having the highest priority. This enables the MDC logic and the device adapter associated with that channel. After the channel number is loaded, the Wait routine branches to the Bus Request routine if a Megabus transfer to the MDC has occurred. If no Megabus transfer has occurred, the Wait routine returns to the Device Support routine provided the adapter is ready.

If no channel requests are active, the Wait routine tests the resume interrupt flip-flop and branches to the Resume Interrupt routine if the flip-flop is set.

The Wait routine delays until a channel request occurs or the resume interrupt flip-flop sets and performs the prioritizing of the execution of channel requests.

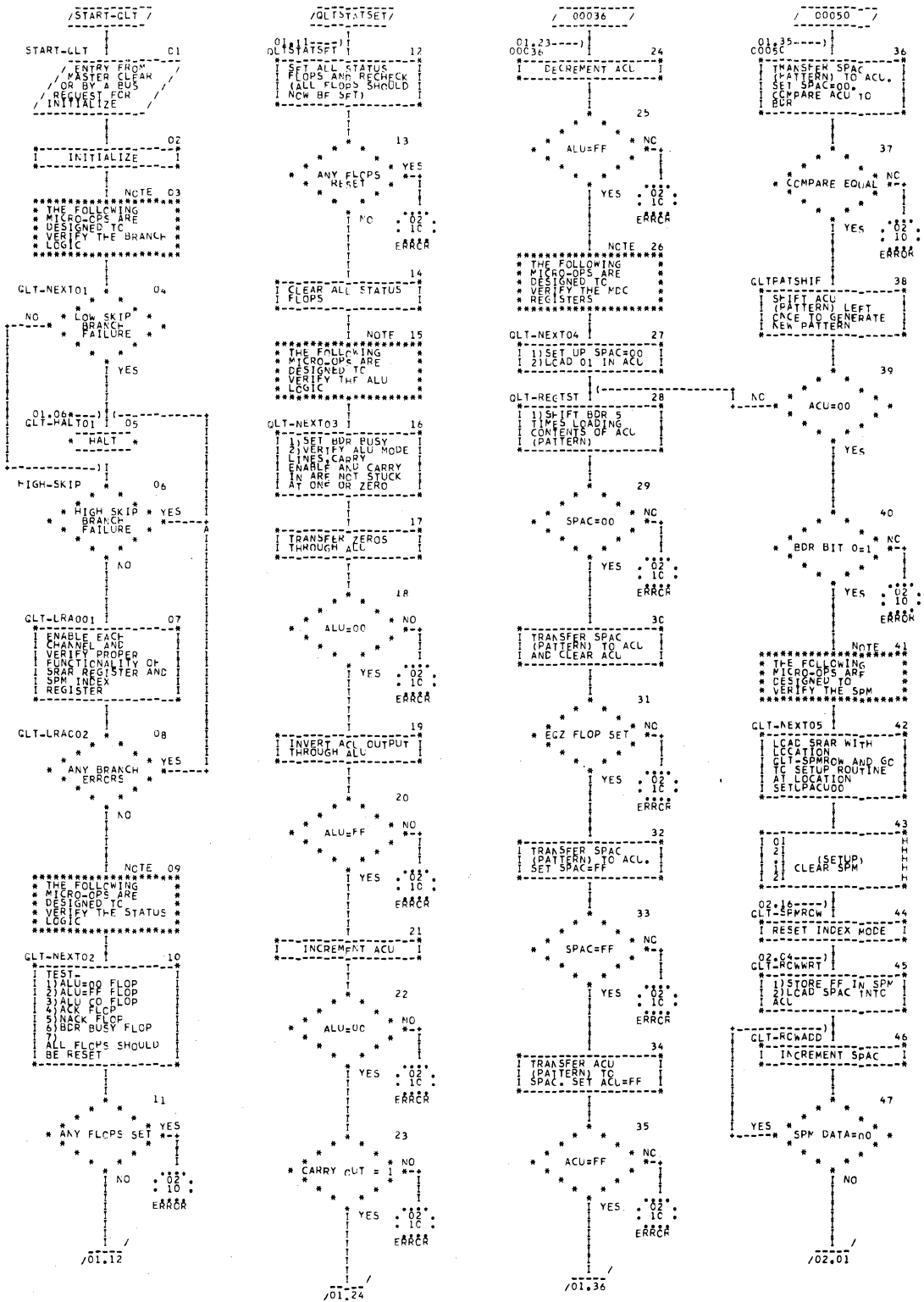


Figure 4-8 Quality Logic Test (Sheet 1 of 3)

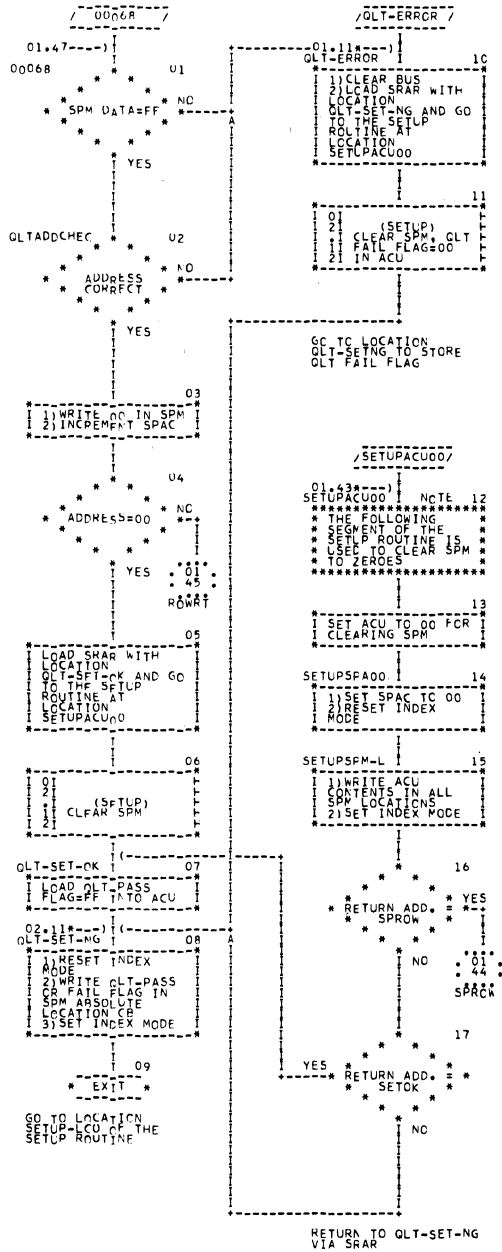


Figure 4-8 Quality Logic Test (Sheet 2 of 3)

HONEYWELL PROPRIETARY AND CONFIDENTIAL

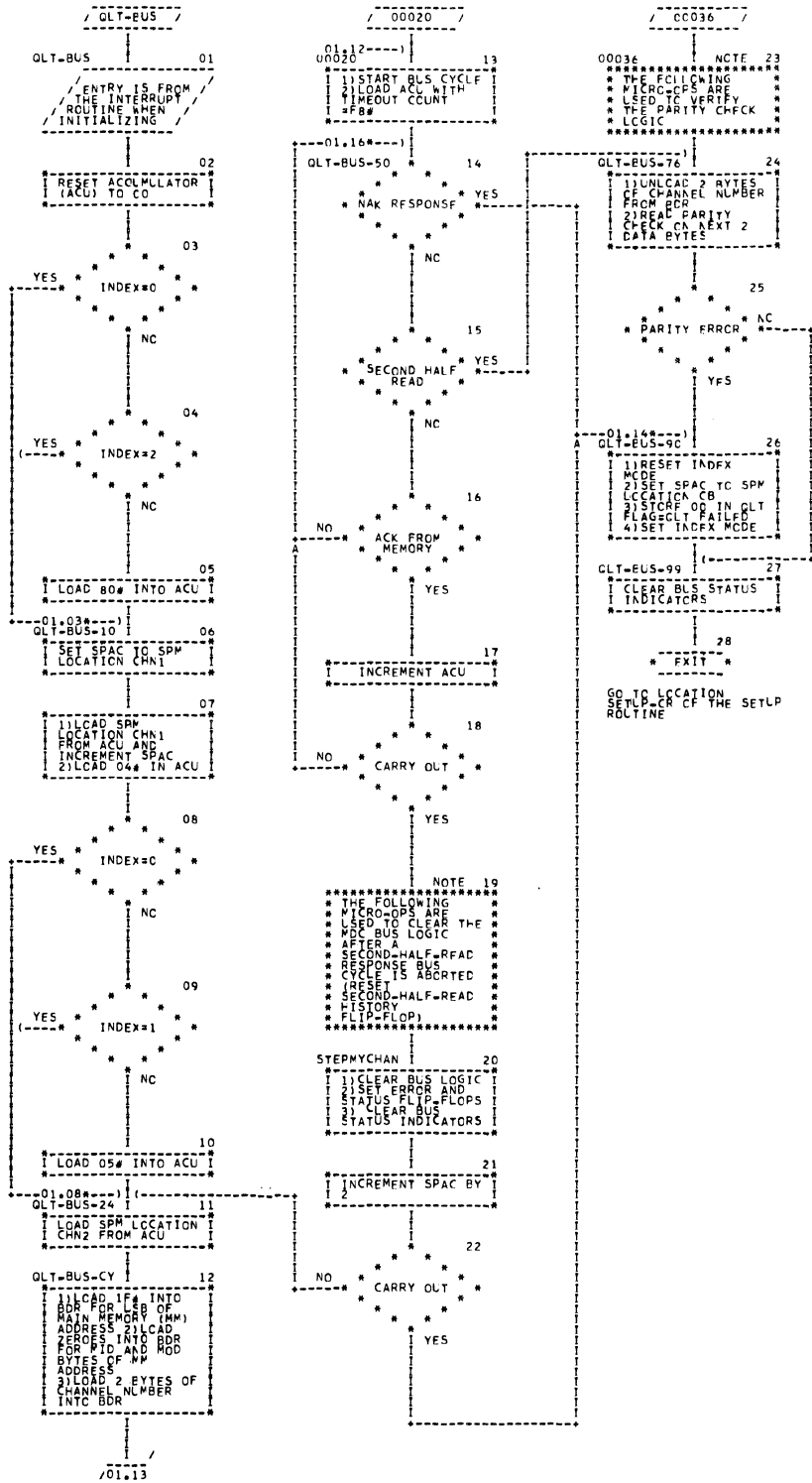


Figure 4-8 Quality Logic Test (Sheet 3 of 3)

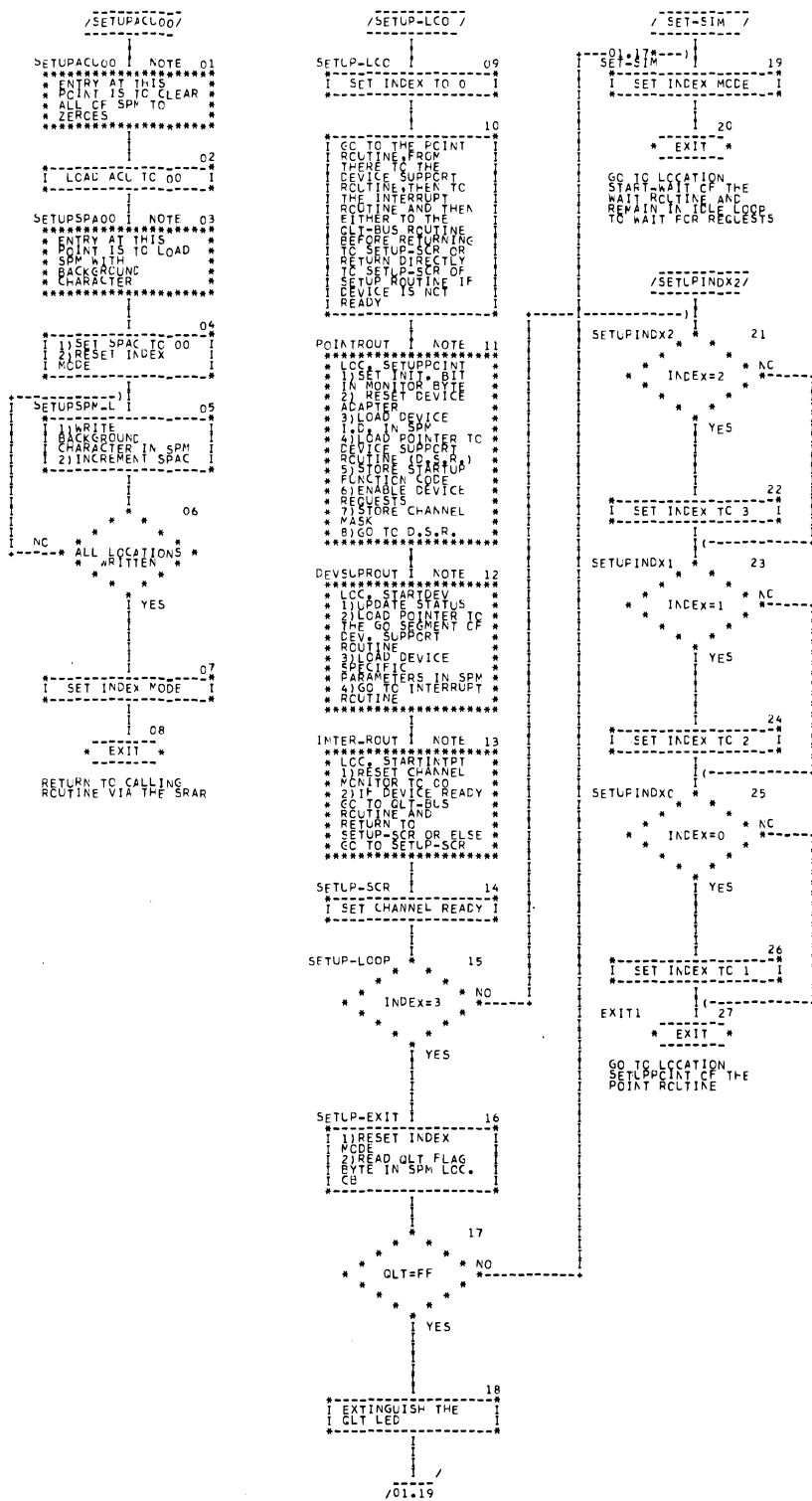


Figure 4-9 Setup Routine

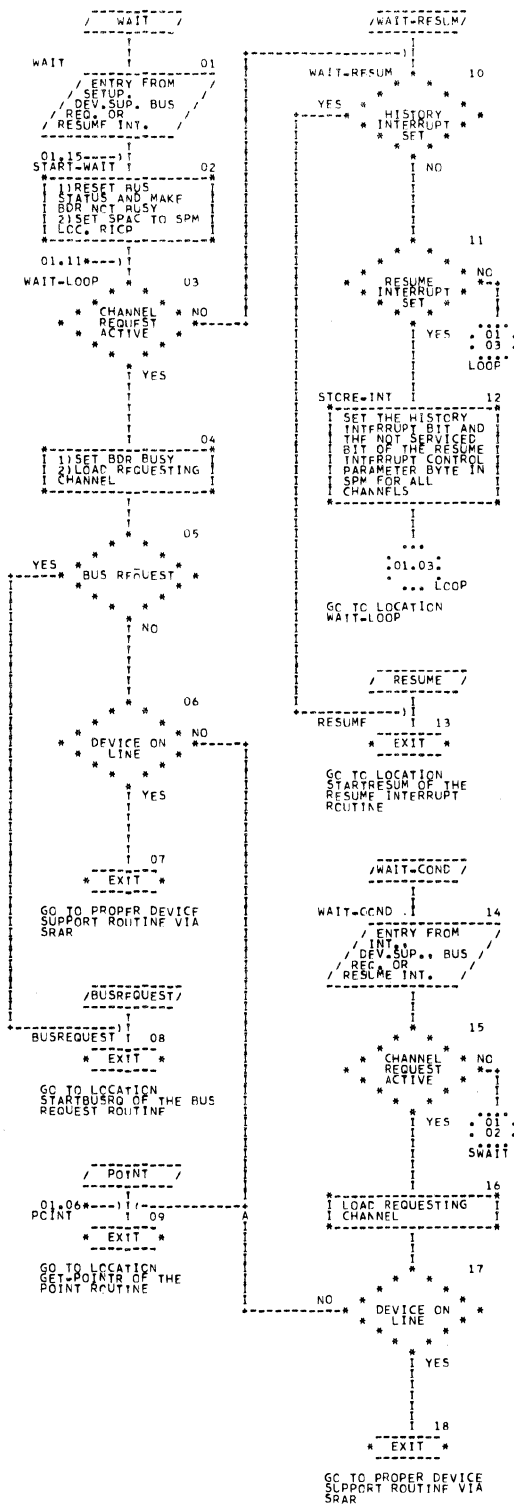


Figure 4-10 Wait Routine

4.4.4 Bus Request Routine (Figure 4-11)

The Wait routine loads the requesting channel numbers and branches to the Bus Request routine upon detection of a Megabus transfer to the MDC. Since the DMA routines, the Interrupt routine, and the Resume Interrupt routine complete all Megabus transfers which they initiate, the Megabus transfer causing the request is unsolicited. This means that the Megabus cycle was initiated by the central processor or by another controller.

If no response is required, the pertinent information from the bus data register data and address segments is stored in the scratch pad memory. This is accomplished by using the function code as the low order six bits of the SPM address. The data is then decoded, causing a branch to the Setup routine, Quality Logic Test, Device Support routine, or back to the Wait routine, depending on the function code and the data field contents.

If a response is required, the scratch pad memory is accessed at the address defined by the function code and index register. Data from the scratch pad memory is loaded in the bus data register, and the response cycle is completed prior to branching back to the Wait routine.

4.4.5 Interrupt Routine (Figure 4-12)

Device support routines branch to the Interrupt routine whenever a potential interrupt condition is detected or when initialization occurs. If initialization has occurred, the Interrupt routine branches to the QLT-Bus subroutine of the Quality Logic Test to verify some of the MDC bus logic.

If initialization has not occurred, the Interrupt routine generates an interrupt provided the interrupt level is not zero. Data to be utilized during the Megabus cycle is stored in the SPM for use by the Resume Interrupt routine and the interrupt pending flip-flop is set if the interrupt is NAK'd. The Wait routine is branched back to at the completion of the Interrupt routine.

4.4.6 Resume Interrupt Routine (Figure 4-13)

The Wait routine branches to the Resume Interrupt routine provided there are no channel requests pending and the resume interrupt flip-flop is set. Upon completion of the Resume Interrupt routine, it branches back to the Wait routine.

The Resume Interrupt routine enables each adapter sequentially and retransmits interrupts that have been previously NAK'd. No activity occurs if a channel does not have an interrupt pending, as indicated by the interrupt-pending flag in the channel monitor byte.

The Resume Interrupt routine sets the channel ready flip-flop and resets the interrupt pending flag if the retransmitted interrupt is ACK'd. The interrupt pending flag remains set and the channel remains busy if the retransmitted interrupt is not ACK'd. All pending interrupts are retried regardless of the central processor response to earlier interrupts.

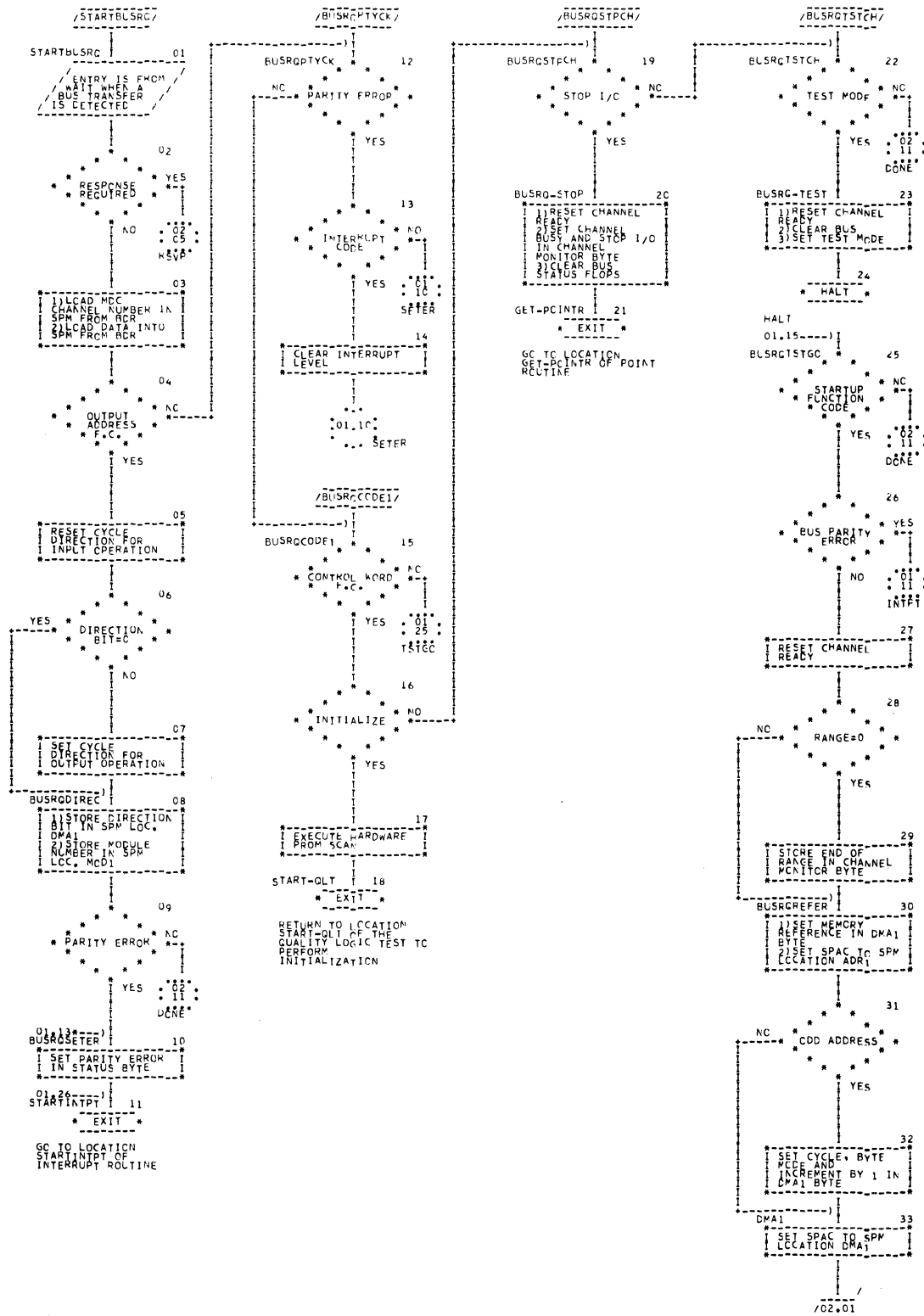


Figure 4-11 Bus Request Routine (Sheet 1 of 2)

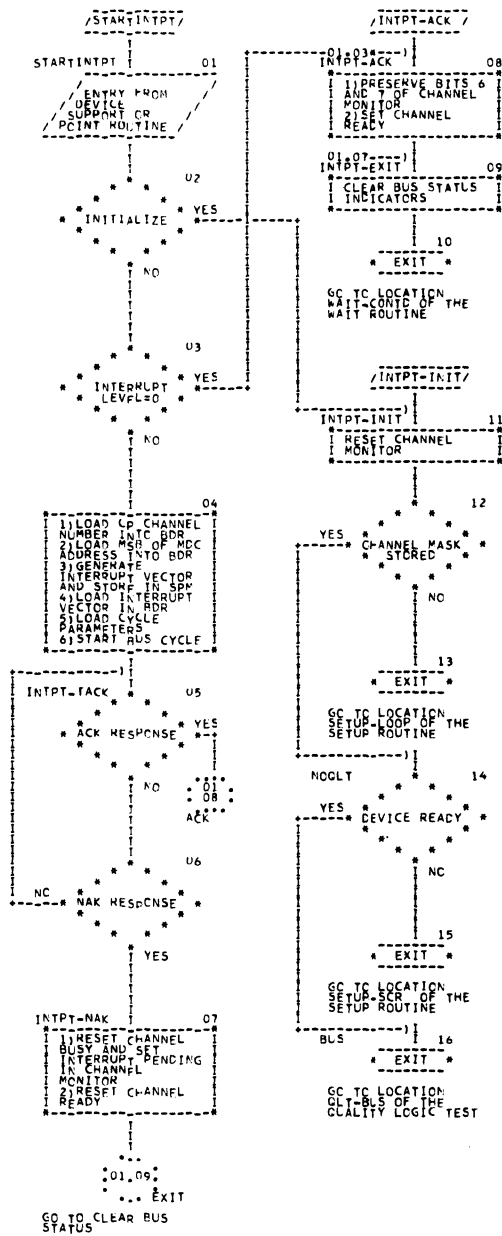


Figure 4-12 Interrupt Routine

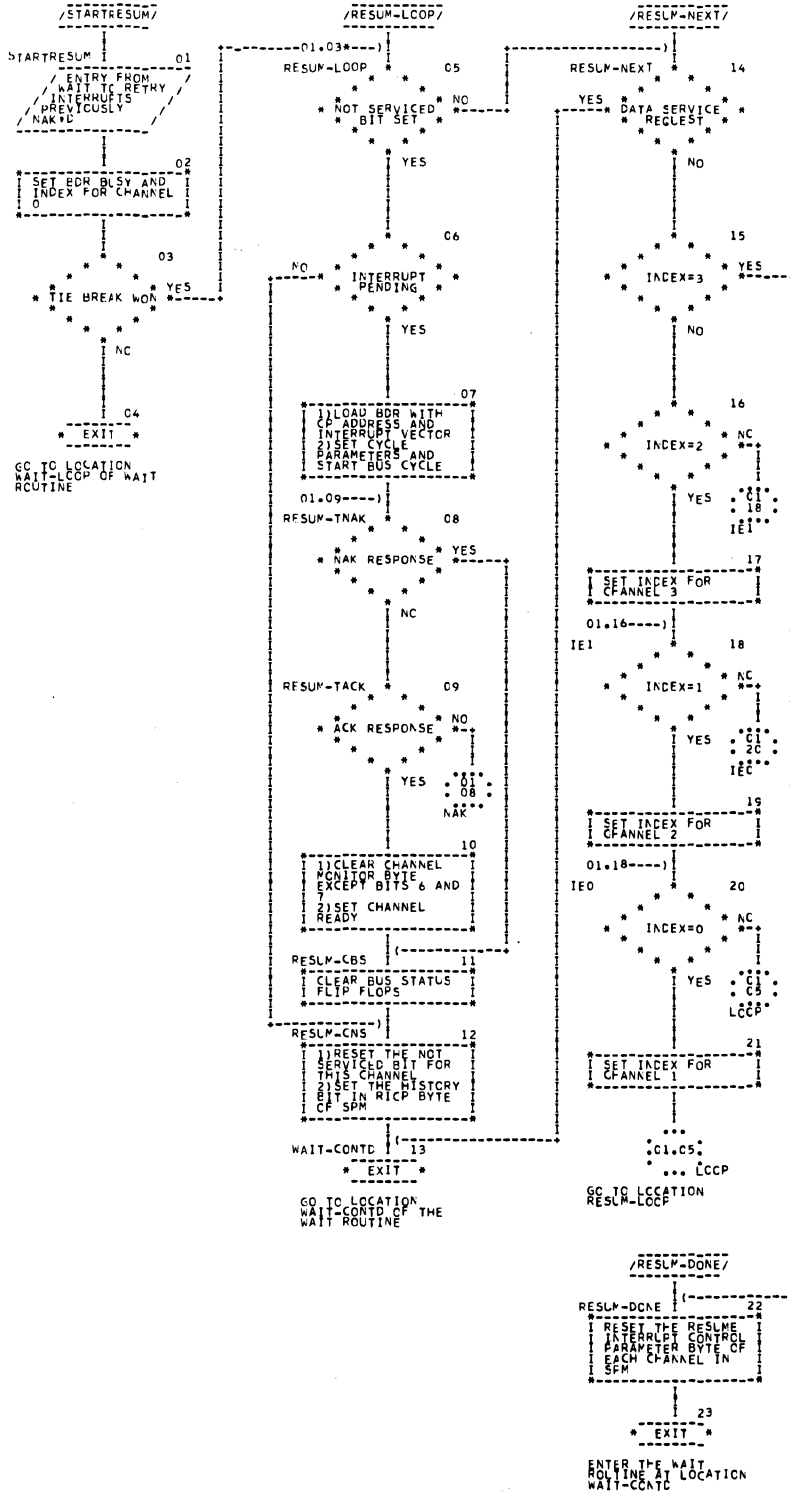


Figure 4-13 Resume Interrupt

4.4.7 Point Routine (Figure 4-14)

The Point routine has two entry points; one is utilized by the Setup routine and the other by the Wait routine and the Bus Request routine. The entry point used by the Setup routine sets the initialize flag in the channel monitor byte and then goes to the second entry point.

The second entry point stores the device identification (ID) code (see Table 4-3) in the scratch pad memory and then uses the ID code to determine the device type connected to the adapter. This is accomplished by the MDC polling each adapter. This results in the adapter transmitting its device ID code to the MDC, provided a device is present. The ID code is compared for validity with a MDC-sorted ID code. If the device type is supported, the startup function code associated with the device is stored, and the SRAR is loaded with the address of the applicable device support routine. In addition, the channel ready flip-flop is set, and device requests for that channel are enabled. The Point routine then returns to the starting address it has loaded into the SRAR.

If a device is not supported the Point routine loads the selected SRAR with the starting address of the Interrupt routine and returns to the Interrupt routine.

4.4.8 DMA Out Routine (Figure 4-15)

The DMA Out (DMAOT) routine is used by the device support routines to transfer data from the main memory to the MDC. In a device output operation, the DMAOT routine initiates a bus cycle to read the data from the main memory.

The DMAOT routine computes address and range and, when applicable, signifies end-of-range to the device support routine. This is accomplished by setting the end-of-range flag in the channel monitor byte located in the SPM. Upon completion of the DMAOT routine, a return to the device support routine is achieved by using the indexed SRAR as an address.

4.4.9 DMA In Routine (Figure 4-16)

The DMA In (DMAIN) routine is used by the device support routines to transfer data from the MDC to the main memory. When writing data in the main memory, the DMAIN routine initiates a bus cycle to accomplish the device input operation.

The DMAIN routine updates address and range and, when applicable, signifies end-of-range to the device support routine. This is indicated by the setting of the end-of-range flag in the channel monitor byte located in the SPM. Upon completion of the DMAIN routine, a return to the device support routine is achieved by using the indexed SRAR as an address.

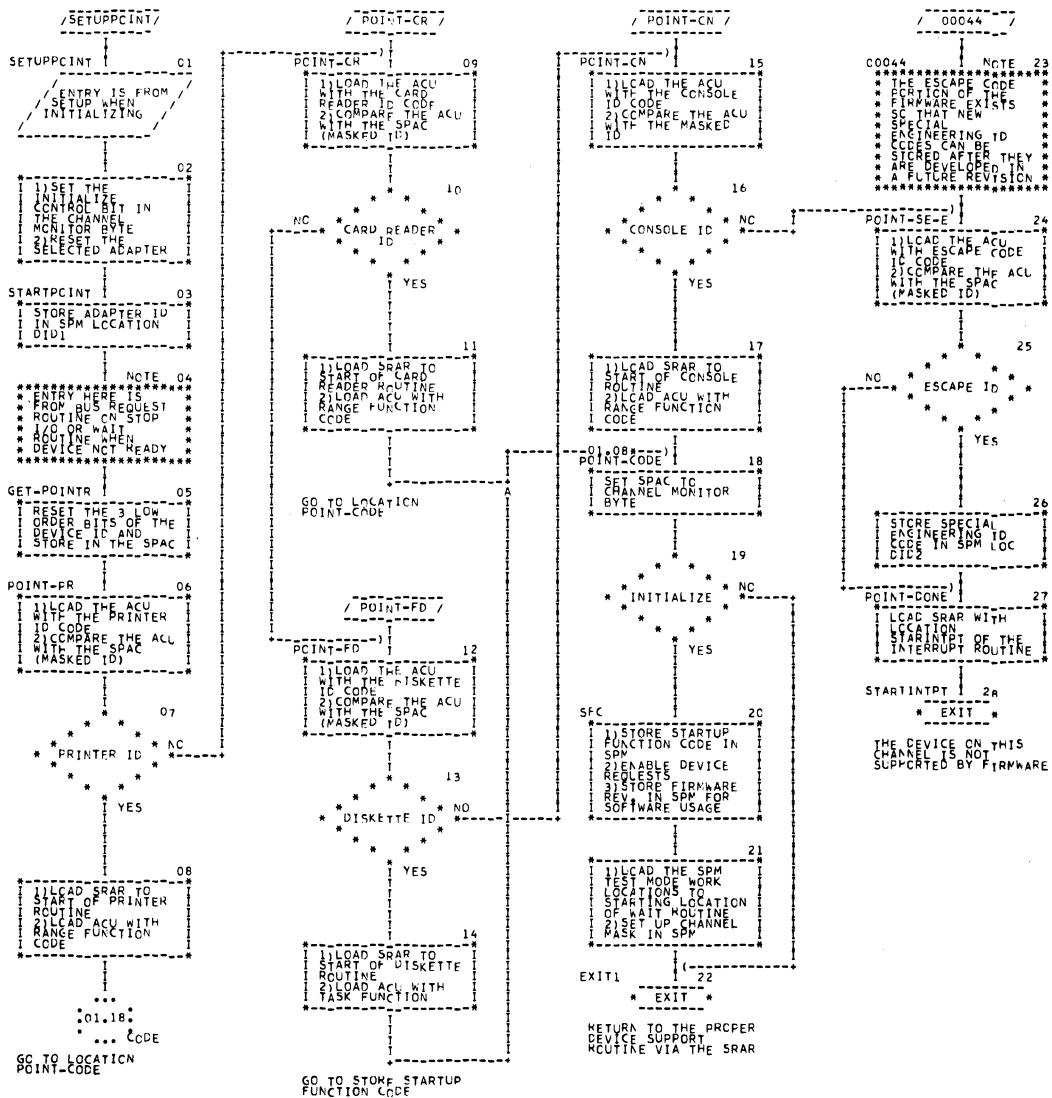


Figure 4-14 Point Routine

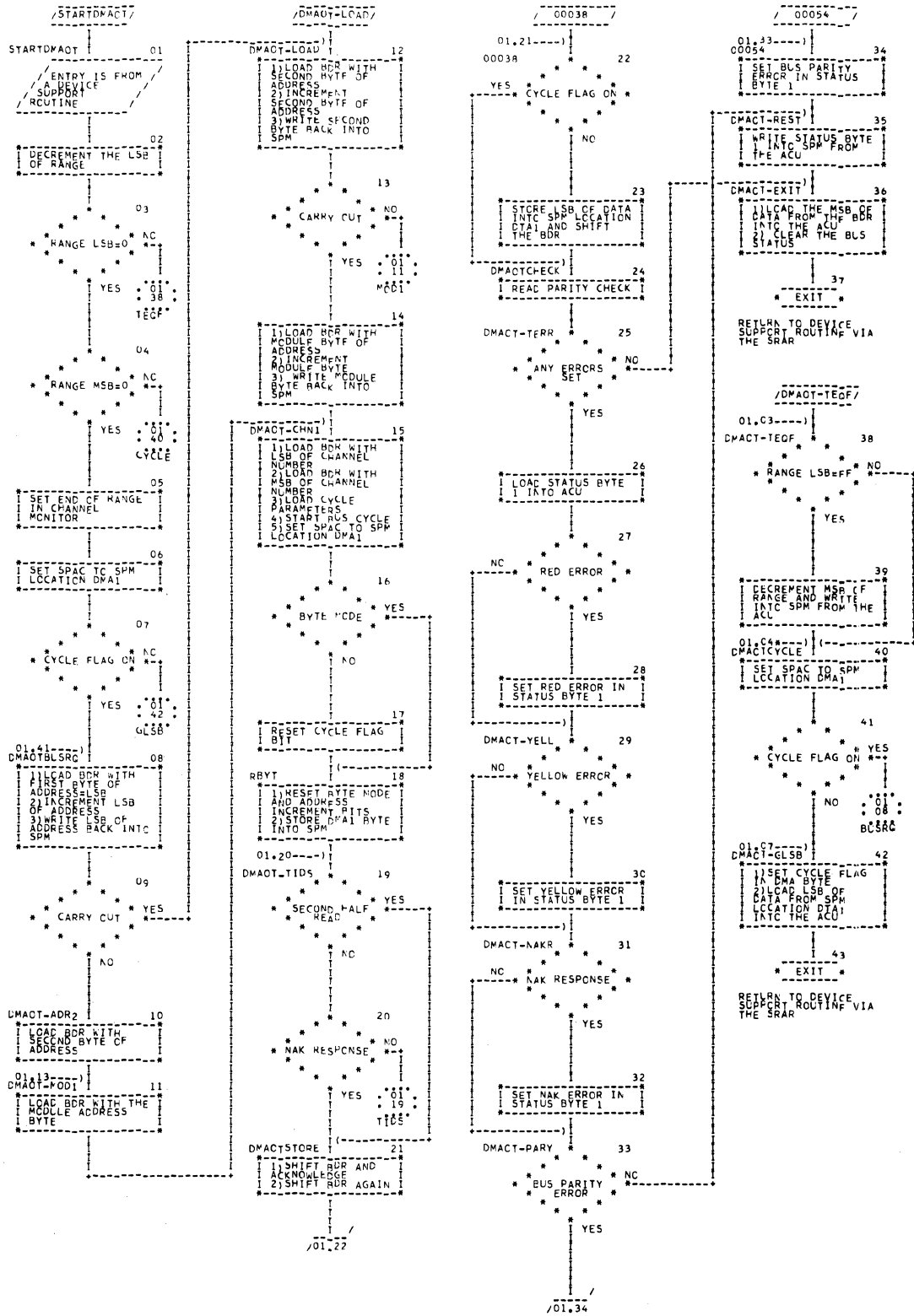


Figure 4-15 DMA Out Routine

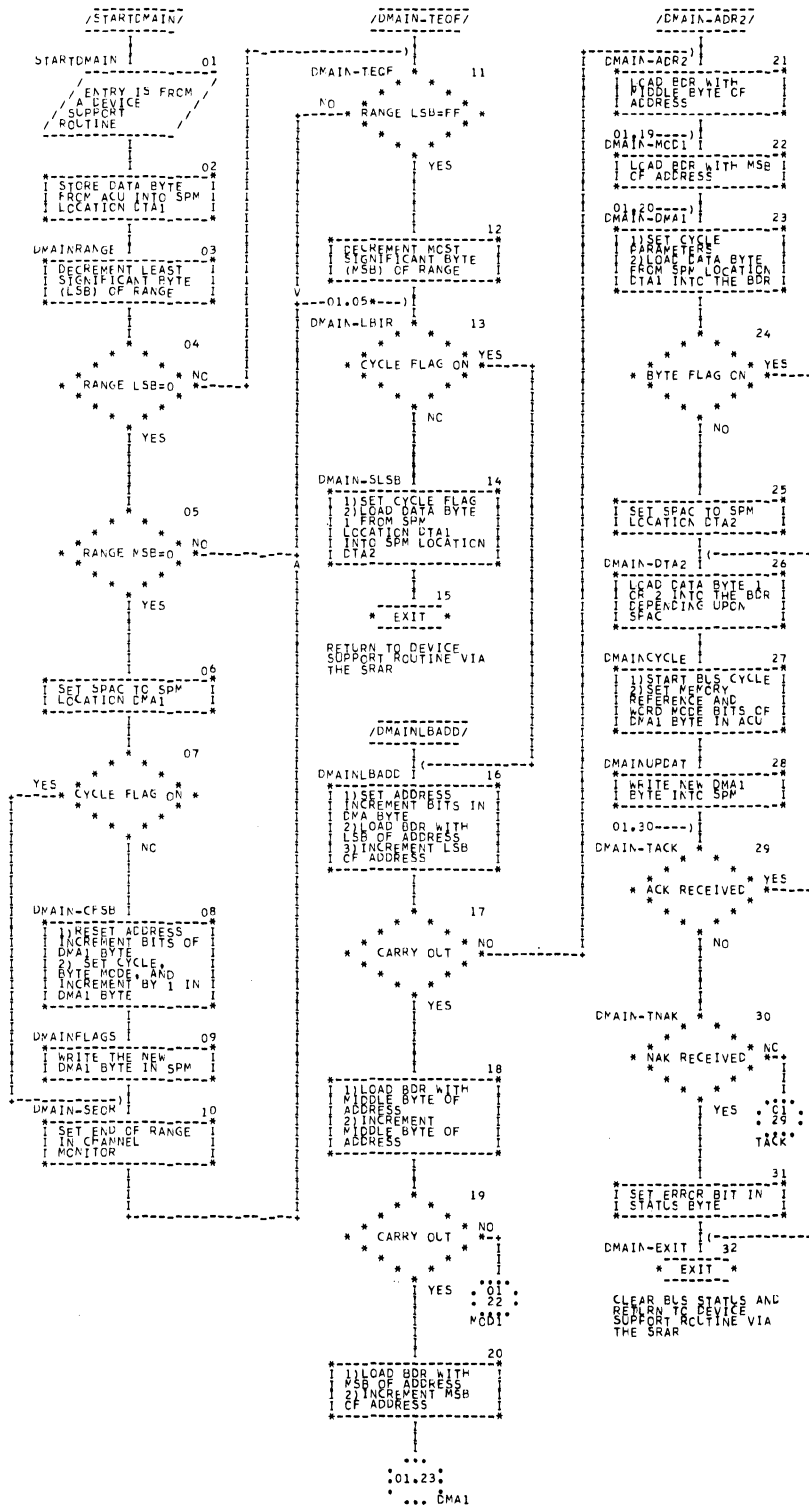


Figure 4-16 DMA In Routine

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

M&TO HARDWARE PUBLICATIONS
USER COMMENTS FORM

DOCUMENT TITLE: _____

PART NO.: _____

ORDER NO.: _____

ERRORS:

HOW DO YOU USE THIS DOCUMENT?

THEORY _____

MAINTENANCE _____

TROUBLESHOOTING _____

OTHER: _____

DOES THIS MANUAL SATISFY YOUR REQUIREMENTS?

YES

NO

IF NOT, PLEASE EXPLAIN _____

FROM: NAME _____, DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

FIRST CLASS
Permit No. 39531
Waltham, Ma.

Business Reply Mail

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

HONEYWELL INFORMATION SYSTEMS INC.
200 SMITH STREET
WALTHAM, MA. 02154

MAIL STATION 872A
HARDWARE PUBLICATIONS, BILLERICA

Honeywell



Honeywell

Honeywell Information Systems

In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

FL19, Rev. 2