EK-DEUNA-TM-PRE

DIGITAL ETHERNET UNIBUS Network Adapter Technical Manual

digital

EK-DEUNA-TM-PRE

DIGITAL ETHERNET UNIBUS Network Adapter Technical Manual

PRELIMINARY

Prepared by Educational Services
of
Digital Equipment Corporation

Preliminary Edition, December 1982

Printed in U.S.A.

This document was set on DIGITAL's DECset Integrated Publishing System.

CONTENTS

Page

CHAPTER 3 LINK MODULE FUNCTIONAL DESCRIPTION

FIGURES

FIGURES (Cont)

TABLES

CHAPTER 1
INTRODUCTION

1.1 SCOPE

This chapter provides an introduction to the DIGITAL ETHERNET UNIBUS Network Adapter (DEUNA). A brief overview of the ETHERNET local area network is included, followed by a description of the DEUNA, its operation, and specifications. Additional documents related to this manual are listed for the reader who wishes more information about the ETHERNET, the DEUNA, or local area networks.

1.2 ETHERNET OVERVIEW

The ETHERNET is a local area network that provides a communications facility for high-speed data exchange among computers and other digital devices located within a moderately sized geographic area. It is intended primarily for use in such areas as office automation, distributed data processing, terminal access, and other situations requiring economical connection to a local communication medium carrying traffic at high-peak data rates.

The primary characteristics of ETHERNET include:

Topology	Branching bus.
Medium	Shielded coaxial cable, Manchester encoded digital base-band signaling.
Data Rate	10 million bits per second.
Maximum Separation of Nodes	2.8 kilometers (1.74 miles).
Maximum Number of Nodes	1,024
Network Control	Multiaccess -- fairly distributed to all nodes.
Access Control	Carrier Sense, Multiple Access with Collision Detect (CSMA/CD).
Allocation	Packet length from 64 to 1518 bytes (includes variable data field of from 46 to 1500 bytes).

The ETHERNET, like other local area networks, falls into a middle ground between long-distance, low-speed networks that carry data for hundreds or thousands of kilometers and specialized, very high-speed interconnections that are generally limited to tens of meters. Using a branching bus topoloy, ETHERNET provides a local area communications network allowing a 10M bits/s data rate over a coaxial cable at a distance of up to 2.8 km (1.74 mi).

A single ETHERNET can connect up to 1,024 nodes together for a local point-to-point/multipoint network. An example of a typical large-scale ETHERNET configuration is shown in Figure 1-1.

Rules for configuring ETHERNET are derived from certain limits that are imposed on the physical channel to ensure the optimal performance of the network. The maximum configuration for an ETHERNET is as follows:

- A segment of coaxial cable can be a maximum of 500 meters (1640.5 feet) in length. Each segment must be terminated at both ends in its characteristic impedance.
- Up to 100 nodes can be connected to any segment of the cable. Nodes on a cable segment must be spaced at least 2.5 meters (8.2 feet) apart.
- The maximum length of coaxial cable between any two nodes is 1,500 meters (4921.5 feet).
- The maximum length of the transceiver cable between a transceiver and a controller is 50 meters (164.05 feet).
- A maximum of 1,000 meters (3281 feet) of point-to-point link is allowed for extending the network.
- Repeaters can be used to continue signals from one cable segment of the ETHERNET to another. A maximum of two repeaters can be placed in the path between any two nodes.

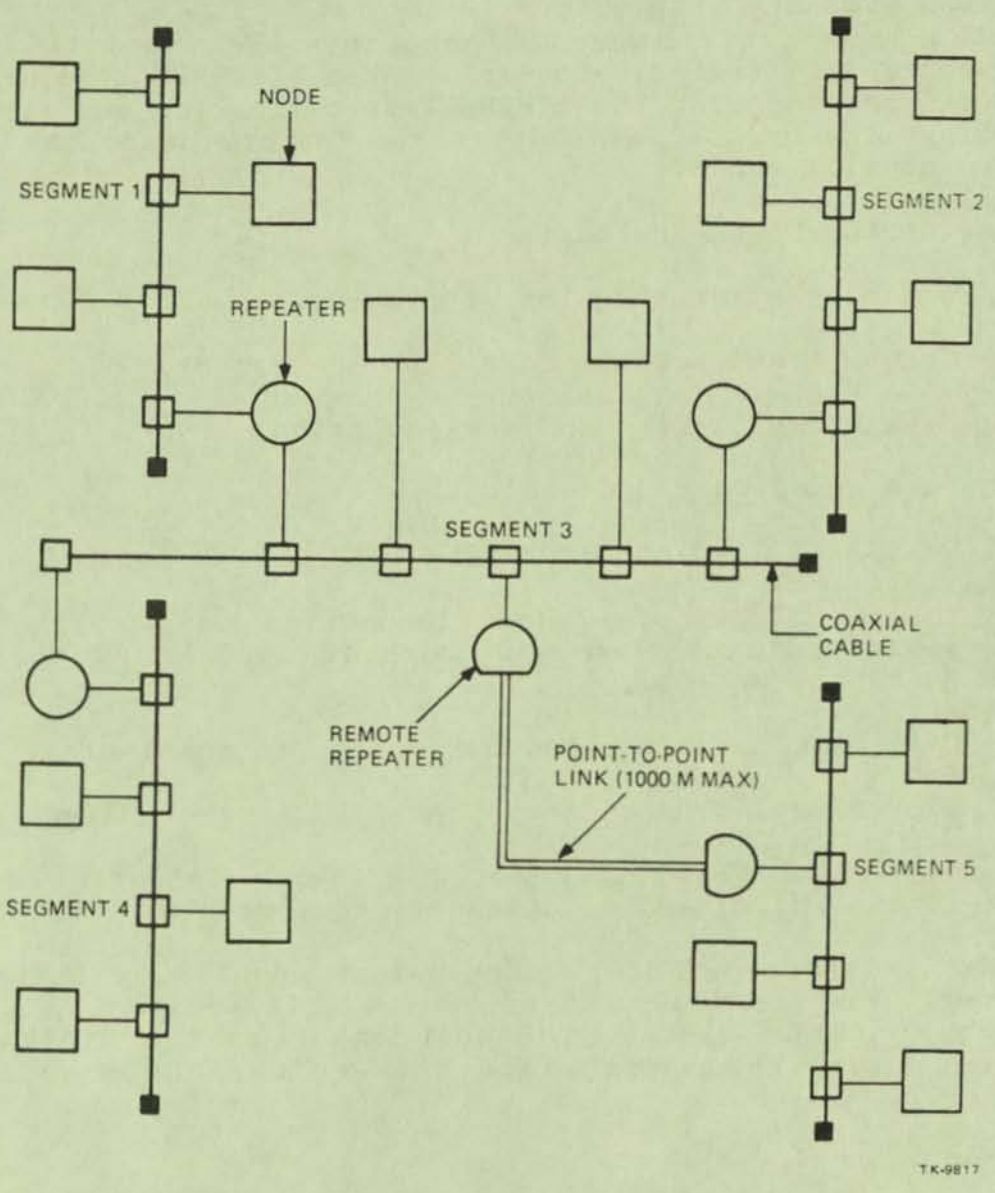


Figure 1-1 Typical Large-Scale ETHERNET Configuration

1.3 DEUNA GENERAL DESCRIPTION

The DEUNA is a data communications controller used to interface VAX-11 and PDP-11 family computers to the ETHERNET local area network. It complies with the ETHERNET specification and allows communication with up to 1024 addressable devices using the ETHERNET shielded coaxial cable.

Features of the DEUNA include:

- 10M bits/s transmission and reception,
- Transmit and receive data link management,
- Data encapsulation and decapsulation,
- Data encoding and decoding,
- Down-line loading and remote load detect capabilities,
- Internal ROM based microdiagnostics to facilitate diagnosis and maintenance of both the DEUNA and the DIGITAL H4000 transceiver,
- Collision detection and automatic retransmission,
- 32-bit Cyclic Redundancy Check (CRC) error detection, and
- 32 KB (16 KW) buffer for continuous datagram reception, transmission, and maintenance requirements.

The DEUNA is comprised of two hex-height modules, a bulkhead interconnect panel, and associated cables. It physically and electrically connects to the ETHERNET cable via the DIGITAL H4000 transceiver and the appropriate transceiver cable as shown in Figure 1-2.

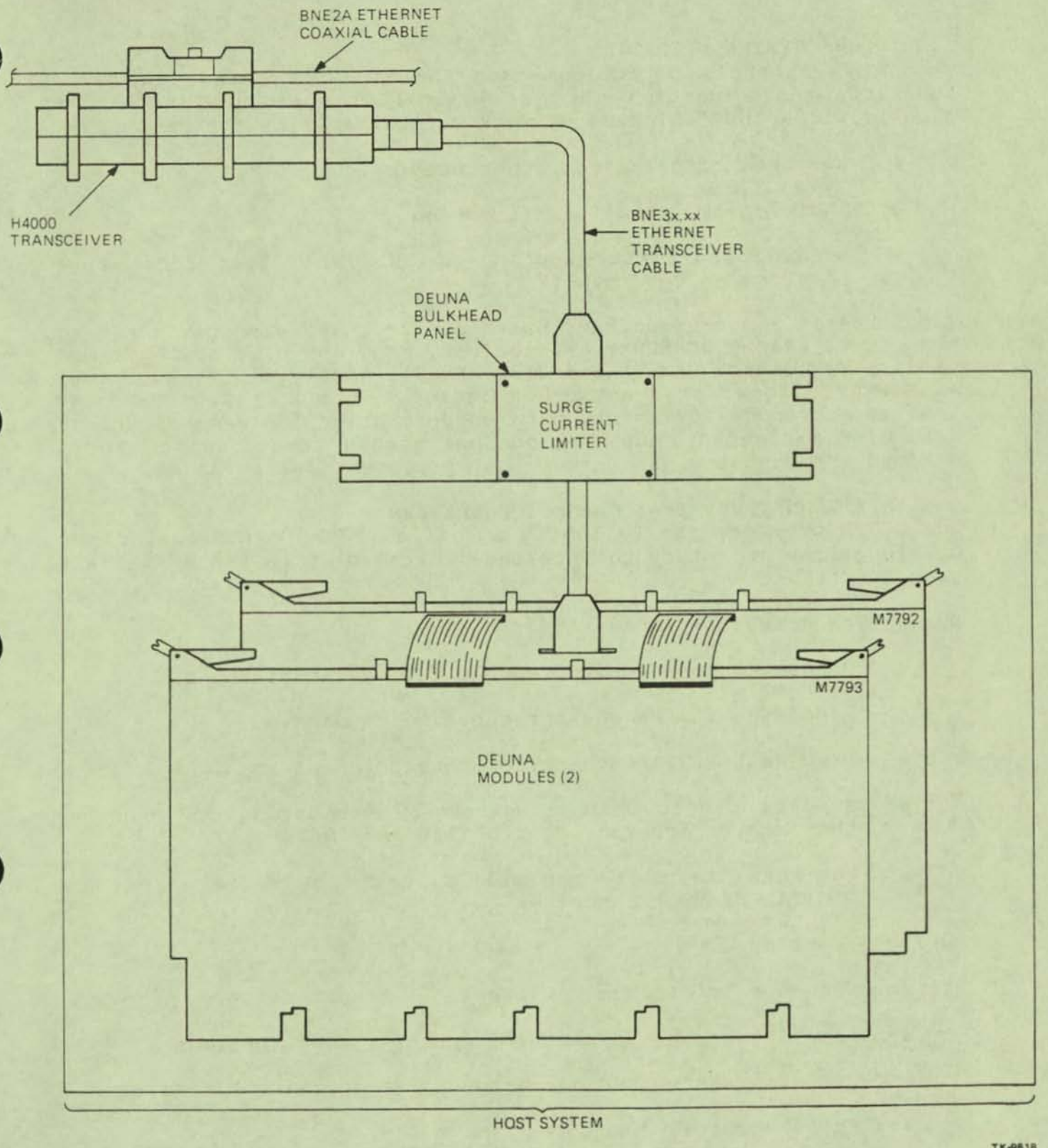


Figure 1-2 DEUNA to ETHERNET Connection

1.4 DEUNA SYSTEM OPERATION

The DEUNA controller performs both the ETHERNET data link layer functions and a portion of the physical channel functions. It also provides the following network maintainability features.

- Loopback of data from other stations.
- Individual system identification.
- Loading and remote booting of UNIBUS systems from other stations on the network.

The DEUNA is a microprocessor based device that, when connected to the DIGITAL H4000 ETHERNET transceiver, provides all the logic necessary to connect VAX-11 and UIBUS PDP-11 family minicomputers to an ETHERNET local area network (Figures 1-3 and 1-4). The controller microcode implements data encapsulation and decapsulation, data link management, and all channel access functions to ensure maximum throughput with minimum host processor intervention.

1.4.1 ETHERNET Physical Channel Functions

The DEUNA provides the following specific ETHERNET physical channel functions necessary to interface to the DIGITAL H4000 ETHERNET transceiver:

During Transmission

- Generates the 64-bit preamble for synchronization.
- Generates the Manchester encoding of data.
- Provides parallel-to-serial conversion of the frame.
- Ensures proper channel access by monitoring and sensing the carrier from any stations' transmission.
- Monitors the self-test collision detect signal from the DIGITAL H4000 transceiver.

During Reception

- Senses carrier from any stations' transmission.
- Provides serial-to-parallel conversion of the frame.
- Performs Manchester decoding of the incoming bit streams.
- Buffers received frames.
- Synchronizes to the preamble and removes it prior to processing.

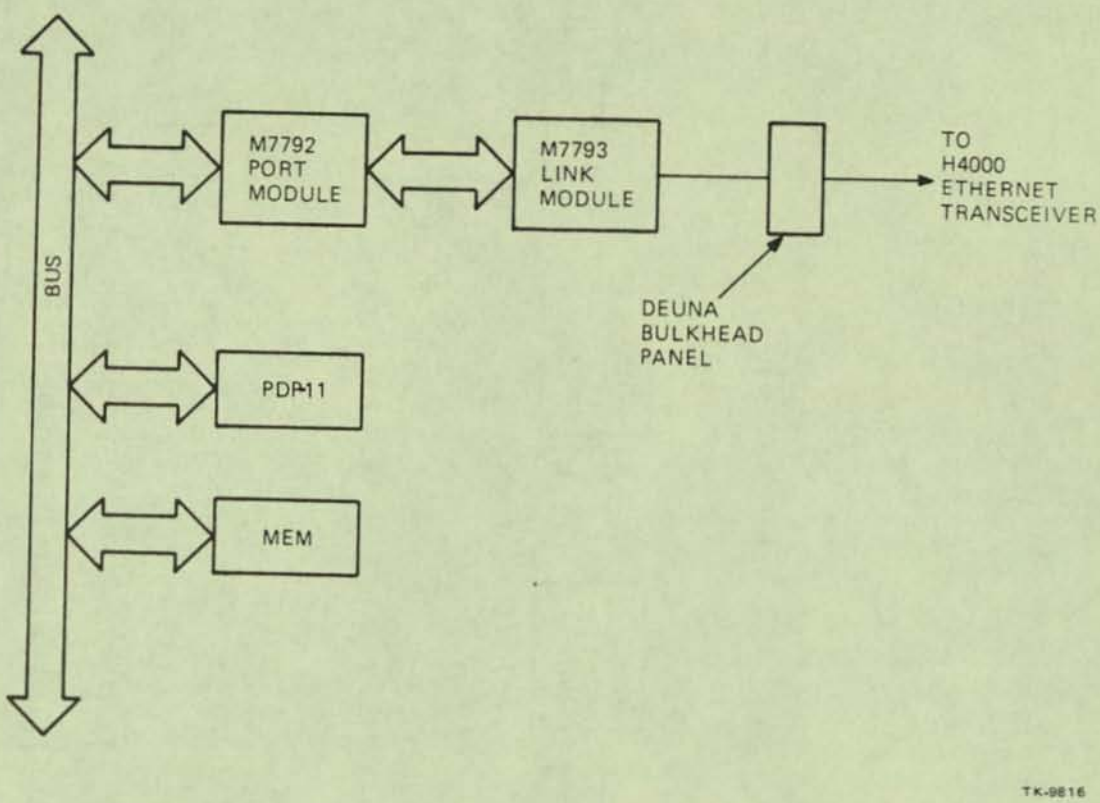


Figure 1-3 PDP-11 Host System Block Diagram

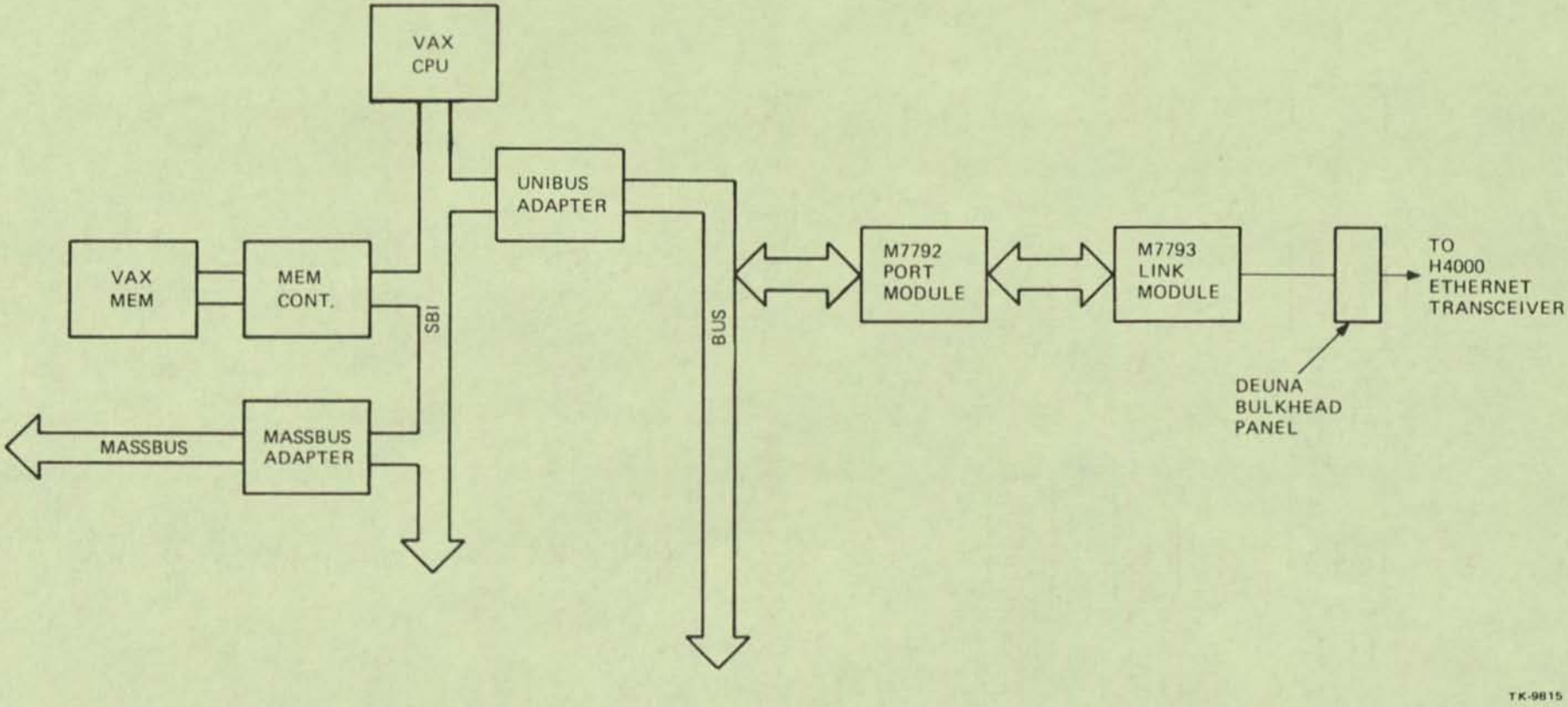


Figure 1-4 VAX-11 Host System Block Diagram

1.4.2 ETHERNET Data Link Functions

The DEUNA provides the following specific ETHERNET data link layer functions.

- Calculates the 32-bit CRC value and places it in the frame sequence field upon transmission
- Attempts automatic retransmission upon collision detection
- Checks incoming frames for proper CRC value
- Performs all address filtration

1.4.3 Data Encapsulation

The ETHERNET frame format for the transmission of data packets is shown in Figure 1-5. Each frame begins with a 64-bit preamble, that is used for synchronization by the receiving station, and ends with a 32-bit frame check sequence. Frames are separated by a specified minimum spacing period of 9.6 microseconds.

The destination address field contains the address(es) of the station(s) where the packet is sent. The address may represent: the physical or logical address of a particular station or group of stations; a multicast, or group address, associated with a set of stations; and a broadcast address for broadcast to all stations on the network.

The source address field specifies the physical address of the transmitting station. Each DEUNA has a unique 48-bit address value determined during manufacture. This value is called the default physical address. The system software can override this value and insert a more apropriate logical address into the source address field upon transmission.

The type field is specified for use by high-level network protocols and it indicates how the content of the data field is to be interpreted. The type field indicates the higher level architecture that can further decapsulate the data.

The data field may have between 46 and 1500 bytes of data. The DEUNA can be initialized to automatically insert null characters if the amount of data is less than the minimum 46 byte data size.

The frame check sequence contains a 32-bit Cyclic Redundancy Check (CRC) value that is determined and inserted by the DEUNA during transmission.

1.4.4 Data Decapsulation

The DEUNA continuously monitors the signals transmitted by the DIGITAL H4000 transceiver. After sensing a carrier, the preamble sequence of the received frame is used by the controller for synchronization. It then processes the destination address field through a hardware comparator to determine whether or not the incoming frame is intended for its station. The DEUNA accepts only

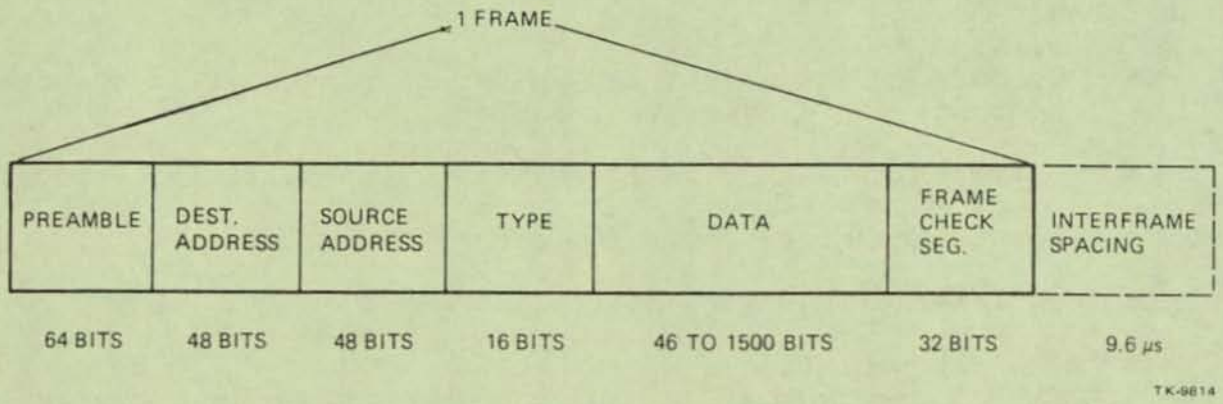


Figure 1-5 Format of an ETHERNET Data Packet

frames that have a destination address that matches one of the following types of address.

1. The physical address of the station
2. The broadcast address for all stations
3. One of the 10 multicast group addresses that the user may assign to the DEUNA
4. Any multicast address
5. All addresses, when desired

The DEUNA performs a hardware comparison of the 48-bit destination address to determine if there is a match with the station's physical address or with one of the ten user designated logical multicast addresses. If necessary, all multicast addresses may be passed to higher level software for decoding when more than ten multicast address groups are required by the user.

To assist in network management functions and to aid in fault diagnosis, the DEUNA can operate in a mode that effectively disregards the internal address filter logic. This allows all frames received from the network to be accepted. The DEUNA verifies the integrity to the received data by recalculating the 32-bit CRC value and comparing it with the CRC that is obtained from the received frame.

1.4.5 Link Management

The method utilized by the ETHERNET for channel access is called carrier sense, multiple access with collision detect (CSMA/CD). The DEUNA controls all of the link management functions necessary to successfully place or remove a frame of data on the ETHERNET network. These functions include:

Carrier Deference	The DEUNA monitors the physical channel for traffic and when the channel is busy, refers to the passing frame by delaying any transmission of its own.
Collision Detection	Collisions occur when two of more controllers attempt to transmit data simultaneously on the channel. The DEUNA monitors the collision sense signal generated by the DIGITAL H4000 transceiver. When a collision is detected, the DEUNA contiues to transmit to ensure that all network stations detect the collision.

Collision Backoff Retransmission	When a controller has attempted transmission and encountered a collision on the channel, it attempts a retransmission a short time later. The schedule for retransmission is determined by a controlled randomization process. The DEUNA attempts to transmit a total of sixteen times and reports an error if it is not successful.

1.4.6 Diagnostics and Maintenance

The DEUNA utilizes both microdiagnostics and extensive system and network diagnostics to greatly minimize the time to isolate and diagnose a network communication fault. On-board self-test microdiagnostics automatically perform a test of the major DEUNA component logic both upon powerup and at the user's discretion. Light emitting diodes on the edge of the port module (M7792) provide an indication of a specific module problem.

The DEUNA does not allow itself to transmit significantly longer than the maximum ETHERNET frame transmit period. It contains an automatic control to prevent monopolizing the ETHERNET channel. The controller can differentiate between normal frame collisions on the physical channel and cable shorts or cable opens. A built-in Time Domain Reflectometry (TDR) circuit is utilized to determine the type of cable fault and its approximate location.

The controller continuously monitors the power applied to the DIGITAL H4000 transceiver to ensure compliance with the transceiver requirements. In addition, the H4000 provides a positive functional verification (heartbeat) after every attempted transmission which indicates its proper operation, including the collision sense circuitry.

Comprehensive system diagnostics provide loopback capability through the DEUNA, transceiver, or the ETHERNET network itself. The DEUNA allows remote stations to loopback through it once the DEUNA has successfully passed the the on-board self-test microdiagnostic. This provides both a local and remote station diagnostic capability. Network error conditions are detected and statistics tabulated for use by higher level network management applications.

1.5 DEUNA SPECIFICATIONS

The DEUNA specifications are outlined in Table 1-1.

Table 1-1 DEUNA Specifications

Specification	Description
Performance	
Operating Mode	Half-duplex
Data Format	ETHERNET specification
Date Rate	10M bits/s
Network Specifications	1024 stations maximum
UNIBUS Conductor Loading	
Module Pair	4 dc loads 2 ac loads
DC Power Requirements	
Port Module	+5 V, 7.0 A
Link Module	+5 V, 9.0 A
	-15 V, 2.0 A (for H4000 transceiver)
Physical Size	
Port and Link Modules	Height (hex): 21.4 cm (8.4 in) Length: 39.8 cm (15.7 in)
Cable Interface Panel	Height: 10.6 cm (4.0 in) Length: 10.6 cm (4.0 in)
Transceiver Cables available in 5 m (16.4 ft), 10 m (32.8 ft), or 20 m (65.6 ft) lengths.	
BNE3A-XX	Low loss PVC jacket/straight connector
BNE3B-XX	Low loss PVC jacket/right angle connector
BNE3C-XX	Low loss TEFLON* jacket/straight connector
BNE3D-XX	Low loss TEFLON* jacket/right angle connector

*TEFLON is a trademark of Dupont de Nemours & Co., Inc.

Table 1-1 DEUNA Specifications (Cont)

Specification	Description
Operating Environment	
Temperature	5°C to 50°C (41°F to 122°F)
Relative Humidity	10 to 90% (noncondensing)
Wet Bulb Temperature	32°C (90°F) maximum
Altitude	Sea level to 2.4 km (8,000 ft)
Shipping Environment	
Temperature	-40°C to 0°C (-40°F to 151°F)
Relative Humidity	0 to 90% (noncondensing)
Altitude	Sea level to 9 km (30,000 ft)

1.6 RELATED DOCUMENTS

Table 1-2 provides a list of documents related to this manual.

Table 1-2 Related Hardware and Software Documents

Title	Document Numbers
DEUNA User's Guide	EK-DEUNA-UG
H4000 Technical Description	(TBS)
The ETHERNET, A Local Area Network, Data Link Layer and Physical Layer Specifications	AA-K759A-TK
ETHERNET Installation	(TBS)
Introduction to Local Area Networks	EB-22714-18
PDP-11 Bus Handbook	EB-17525

DIGITAL personnel may order hardcopy documents from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532

Attn: Publishing and Circulation Services (NR03/W3)
Order Processing Section

Customers may order hardcopy documents from:

Digital Equipment Corporation
Accessories and Supplies Group
Cotton Road
Nashua, New Hampshire 03060

For information call: 1-800-257-1710

Information concerning microfiche libraries may be obtained from:

Digital Equipment Corporation
Micropublishing Group (BUO/E46)
12 Crosby Drive
Bedford, MA 01730

CHAPER 2
PORT MODULE
FUNCTIONAL DESCRIPTION

2.1 OVERVIEW

The port module (M7792) is the microprocessor controlled interface between the UNIBUS bus and the link module of the DEUNA. The logic on the port module is divided into three basic functional areas.

1. UNIBUS Interface -- This section contains the UNIBUS transceiver, port control and status registers (PCSRs), DMA control, UNIBUS interrupt control logic, and UNIBUS control.

2. Microprocessor Section -- This section is made up of the T11 microprocessor, 8K words of ROM for microprogram storage, 4K words of writable control store (WCS), internal register address decode, and timer.

3. Link Memory Control -- This section contains the link memory arbitration logic, control for the 16K words of link memory and the port-to-link interconnect.

The port module is a hex-height module that is installed in a small peripheral controller (SPC) slot of a UNIBUS backplane.

A functional block diagram of the port module is shown in Figure 2-1. The letters in the lower right corner of each block of the diagram indicate the page in the engineering drawings where the logic for that block is located.

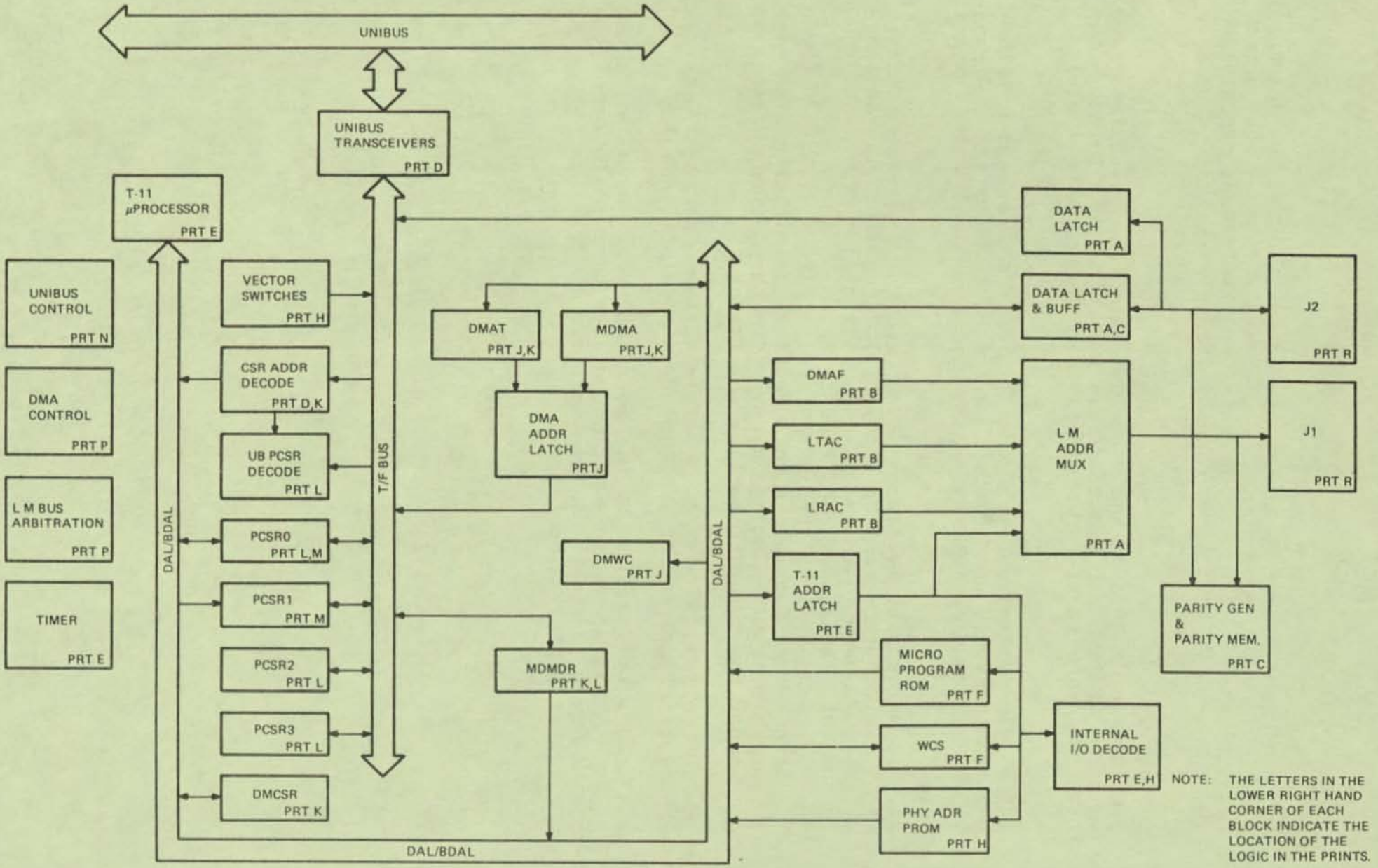


Figure 2-1 PORT Module Functional Block Diagram

2.2 UNIBUS INTERFACE

The UNIBUS interface logic on the port module is used to control the transfer of data between the host processor and the DEUNA. This logic generates the signals required of a bus master and bus slave on the UNIBUS. The DEUNA functions as bus master when data is to be transferred to or from the host processors memory via direct memory access (DMA). The DEUNA performs DMAs for the transfer of:

- Data received from the ETHERNET and
- Data to be transmitted on the ETHERNET.

The DEUNA functions as a bus slave when the host processor accesses the port control and status registers (PCSRs) for the transfer of control and status information.

NOTE

For a detailed explanation of UNIBUS architecture and protocol, refer to the PDP-11 Bus Handbook (EB-17525).

The port also controls the UNIBUS ACLO signal. It does this by setting a bit in the link mode register on the link module (see Section 3.3.2). This is used to get control of the host processor during a down-line load.

2.2.1 DMA Control

The DMA control logic is divided into two sections:

1. RX DMA -- Used when a message has been received from the ETHERNET and is ready to be transferred to the host processors memory.
2. T11 UNIBUS DMA -- Used when the T11 has to:
   - Read the ring structures in host memory,
   - Read data buffers in host memory for transmission on the ETHERNET, and
   - Write status information into the data buffers in host memory when the transmission is finished.

The control of each of these processes is implemented via programmed array logic (PAL) with the T11 UNIBUS DMA having a higher priority than the RX DMA process. This priority is established because the T11 UNIBUS DMA process transfers its data in smaller segments and therefore does not use the UNIBUS for long periods of time. This results in little effect on the RX DMA process and helps to maximize thoughput.

A description of each of the PALs used in the DMA control is contained in the engineering drawings for the DEUNA.

2.2.1.1 RX DMA -- The DEUNA transfers received messages to host memory via the UNIBUS using DMAs. This is done asynchronous to the process or processes going on in the DEUNA. The port microprocessor (T11) starts the DMA transfers by loading a group of registers with the necessary address and word count information. Once this information is loaded, the T11 starts the DMA process by setting a bit in the DMA control and status register (DMCSR). This starts the DMA transfers under the control of the RX DMA PAL and the UNIBUS control logic.

The following registers, located on the port module, are used for the RX DMA process:

- DMCSR -- DMA control and status register,
- DMAT -- DMA-to-address register,
- DMAF -- DMA-from-address register, and
- DMWC -- DMA word count register.

The T11 controls the transfer of data from the buffers located on the link module to the host memory in the following manner:

1. The T11 loads the DMAT, DMAF, and the DMWC with the proper information.
2. The T11 sets the DMA GO bit in the DMCSR.
3. The DMA logic takes over and moves the data via NPRs to the host memory.
4. When all the data is transferred or when the RX DMA logic receives an error, it interrupts the T11.

The RX DMA logic only transfers words on the UNIBUS. The host software is responsible for throwing away the extra byte when transferring an odd byte buffer.

A description and layout of each of the registers used is given in the following sections.

2.2.1.1.1 DMA Control and Status Register (DMCSR) -- The DMCSR is used by the T11 to enable the DMA logic and to report DMA status to the T11. Figure 2-2 shows the DMCSR bit format and Table 2-1 gives a description of each of the bits.

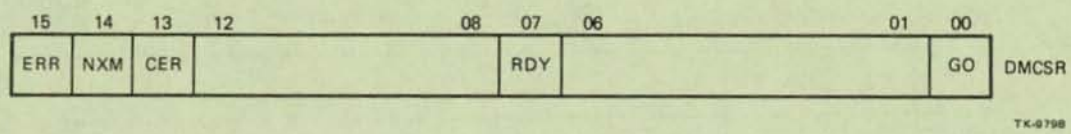


Figure 2-2 DMCSR Bit Format

Table 2-1 DMCSR Bit Descriptions

Bit	Field	Description
DMCSR<ØØ>	GO	Go Bit -- This bit is set after the address and word count are loaded. On setting the DMA, the engine begins to arbitrate for the UNIBUS and starts data transfer to host memory.
DMCSR<Ø7>	RDY	Ready Bit -- This bit creates an interrupt to the T11 to indicate that the word count has expired and the current DMA process is complete.
DMCSR<13>	CER	Collision Error -- When set indicates that the heartbeat from the H4ØØØ or similar transceiver was not detected.
DMCSR<14>	NXM	Non-Existent Memory -- When set causes the DMA logic to interrupt the T11. Indicates a UNIBUS timeout to the address contained in the DMA-to-address register.
DMCSR<15>	ERR	DMA Logic Error -- Set when UPE or NXM are set.

2.2.1.1.2 DMA to Address Registers (DMATØ and DMAT1) -- The DMAT registers are loaded by the T11 with the starting address of the receive buffer in host memory. DMATØ contains the lower 16 bits of the address. DMAT1 contains the upper 2 bits of the address. These registers are a 17-bit counter that is incremented by two after each NPR cycle.

NOTE

Bit Ø of DMATØ is always a Ø. This is because the RX DMA logic only performs word transfers.

Figure 2-3 shows the format of each of the registers.

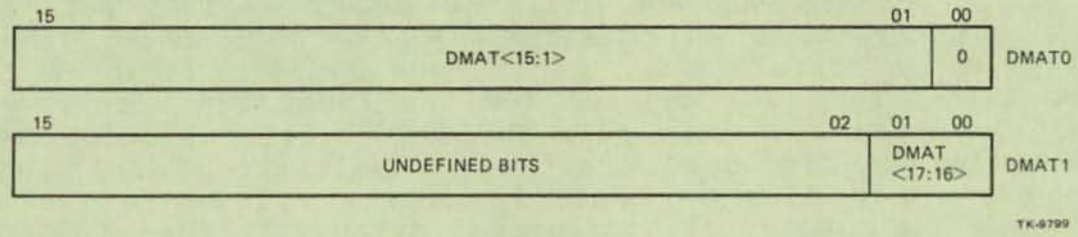


Figure 2-3 DMAT Bit Format

2.2.1.1.3 DMA-from-Address Register (DMAF) -- The DMAF register contains the receive buffer address in the link memory from which data is to be transferred. It is made up of a register for the upper four bits and a counter for the lower ten bits. The T11 loads the upper four bits from the link completed buffer address FIFO (refer to Section 2.4.3 for an explanation of the LCBAF). When the upper four bits are loaded, the lower ten bits are cleared. The counter is incremented by two after each NPR cycle. The address cannot overflow into the next buffer. Figure 2-4 shows the bit format of the DMAF register.

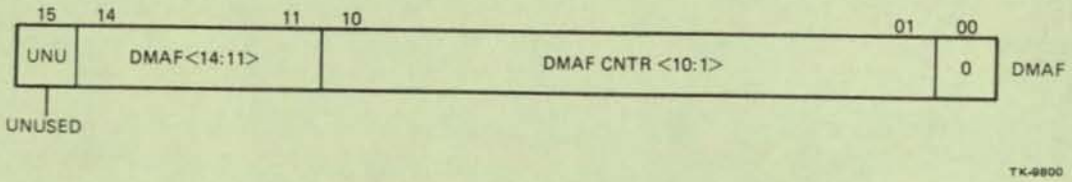


Figure 2-4 DMAF Bit Format and Descriptions

2.2.1.1.4 DMA Word Count Register (DMWC) -- The DMWC is loaded by the T11 with the number of words to be transferred to host memory by the DMA logic. The DMWC is implemented in a counter. After each NPR cycle it is decremented by two. When the register goes to zero, the DMA GO bit in the DMCSR is cleared thereby stopping the DMA logic. The RDY bit in the DMCSR is set causing an interrupt to the T11. Figure 2-5 shows the bit format of the DMWC register.

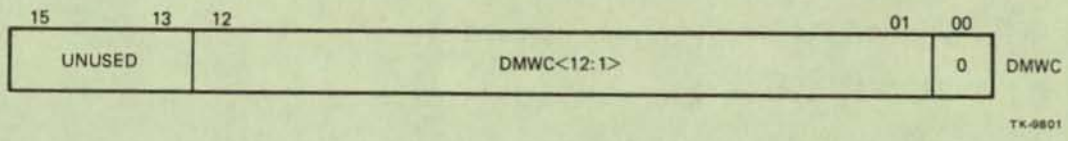


Figure 2-5 DMWC Register Bit Format and Bit Descriptions

2.2.1.2 T11 UNIBUS DMA -- The T11 UNIBUS DMA is used by the port microprocessor to access the host memory in order to perform the following functions:

- Read ring structure data,
- Read data buffers from host memory for transmission by the link, and
- Write status information to data buffers upon completion of transmission.

A T11 UNIBUS DMA transactions occurs in the following sequence:

1. The T11 loads the UNIBUS address registers, MDMAØ and MDMA1.
2. The T11 either reads or writes the microprocessor DMA data register. (MDMDRØ incrementing or MDMDR1 decrementing. Refer to Section 2.2.3.1.)
3. The reading or writing of the data register causes the microprocessor DMA to acquire the UNIBUS and transfer the data to/from host memory. During the DMA process, the T11 is stalled until the transfer is complete.

2.2.1.2.1 Microprocessor DMA Address Registers (MDMAØ and MDMA1) -- The micoprocessor DMA address registers are made up of a 17-bit counter that contains the address in host memory to or from which the data is to be transferred. MDMAØ contains the lower 16 bits of the address and MDMA1 contains the upper 2 bits of the address.

NOTE
Bit Ø of MDMAØ is always a Ø because the DMA logic only performs word transfers.

Figure 2-6 shows the Microprocessor DMA Address Register Bit Format.

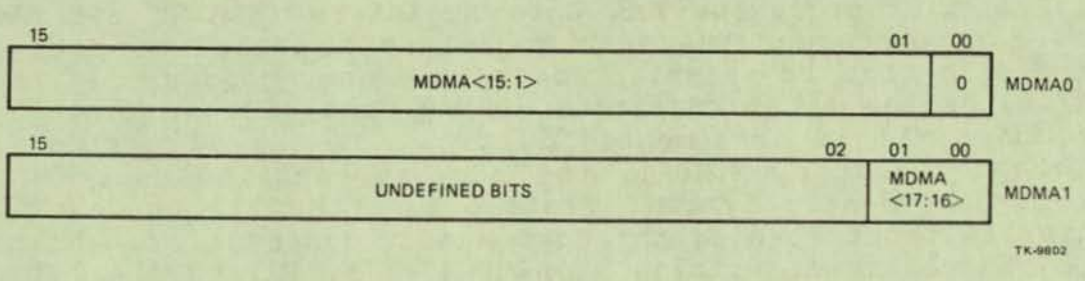


Figure 2-6 Microprocessor DMA Address Register Bit Format and Description

2.2.1.2.2 Microprocessor DMA Data Registers (MDMDRØ and MDMDR1) -- The microprocessor DMA data registers are used as data ports for the data that is transferred to/from host memory. If the T11 reads/writes the first register, MDMDRØ, the address contained in MDMAØ and MDMA1 is incremented by two. If the T11 reads/writes the second register, MDMDR1, the address contained in MDMAØ and MDMA1 is decremented by two. This allows the T11 to do multiple transfers without loading the host memory address for each transfer. The reading or writing of MDMDRØ or MDMDR1 by the T11 generates an NPR request to the UNIBUS. Refer to Figures 2-7 and 2-8 for MDMDRØ and MDMDR1 bit formats.

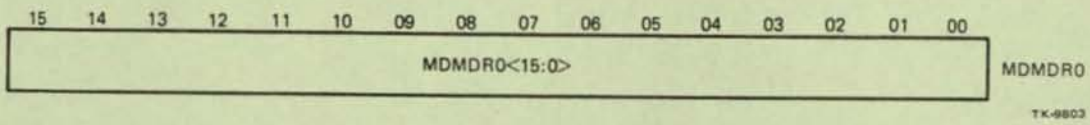


Figure 2-7 Data Register MDMDRØ (Incrementing)

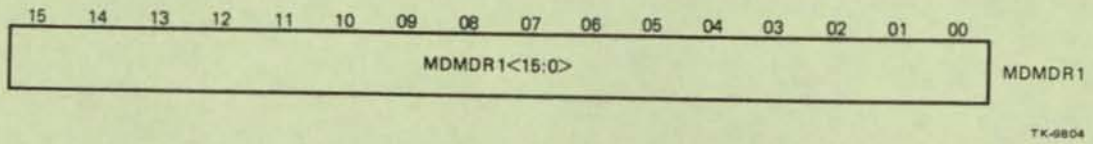


Figure 2-8 Microprocessor DMA Data Register MDMDR1

2.2.2 Port Control and Status Registers

The port control and status registers (PCSR) are used by the port module to receive commands from the host processor and report the results of the command along with other status information (interrupts, etc.).

There are four PCSRs, each with a specific function. The following sections show the format of the PCSRs and give a description of their function.

For a more detailed explanation of functions performed by the PCSRs, refer to Chapter 4, "Programming", of the DEUNA User's Guide (EK-DEUNA-UG).

2.2.2.1 Port Control and Status Register Ø (PCSRØ) -- Figure 2-9 shows the format of PCSRØ and Table 2-2 lists the functions of the bits.

15	14	13	12	11	10	09	08	07	06	05	04	03 00	
SERI	PCEI	RXI	TXI	DNI	RCBI	0	USCI	INTR	INTE	RSET	0	PORT_COMMAND	PCSR0
RWCL	RWCL	RWCL	RWCL	RWCL	RWCL	0	RWCL	R	R/W	W	0	R/W	PORT DRIVER ACCESS
W	W	W	W	W	W	0	W	W	R	R	0	R	PORT ACCESS
0	0	0	0	0	0	0	0	0	0	0	0	U	POWER UP STATE

TERMS

RWCL	READ ACCESS, WRITE ONE TO CLEAR
R/CL	READ ACCESS, CLEAR
R	READ ONLY, IGNORED WHEN WRITTEN
R/W	READ/WRITE
W	WRITE ONLY, READ AS ZERO
U	UNDEFINED

TK-6058

Figure 2-9 PCSRØ Format

Table 2-2 PCSRØ Bit Descriptions

Bits	Name	Description
<15>	SERI	Status Error Interrupt -- Indicates the presence of an error condition flagged in status register accessible by the port command function. Set by the DEUNA, cleared by the port-driver through the read and clear status port function.
<14>	PCEI	Port Command Error Interrupt -- Indicates the occurrence of either a function error or a UNIBUS timeout during the execution of a port command. Bit 7 of PCSR1 distinguishes between the two error conditions. Set by the DEUNA, cleared by the port-driver.
<13>	RXI	Receive Ring Interrupt -- Attention bit for ring updates. Set the by the DEUNA cleared by the port-driver. Indicates, when set, that the DEUNA has placed a message(s) on the ring.
<12>	TXI	Transmit Ring Interrupt -- Attention bit for ring updates. Set by the DEUNA, cleared by the port-driver. Indicates, when set, that transmission has been suspended. All messages it found on the transmit ring have been set, or an error was encountered during a transmission.
<11>	DNI	Done Interrupt -- Interrupts when the DEUNA completes a port command. (Note: the port command NO-OP does not cause the DNI bit to set.) Set by the DEUNA, cleared by the port-driver.
<10>	RCBI	Receive Buffer Unavailable Interrupt -- Interrupts when the DEUNA discards an incoming message due to receive ring buffers being unavailable. Once set by the DEUNA, RCBI is not set again until after the DEUNA has received a PDMD port command and has discarded a subsequent message. Set by the DEUNA, cleared by the port-driver.
<09>	zero	

Table 2-2 PCSRØ Bit Descriptions (Cont)

Bits	Name	Description
<Ø8>	USCI	Unsolicited State Change Interrupt -- Interrupts when the DEUNA performs the following actions: Fatal Error -- A transition into the NI and UNIBUS halted state from the ready, running, UNIBUS halted, or NI halted states. This state change is caused by the DEUNA detecting an internal fatal error, that is, internal parity error. Communication Processor Boot -- A transition into the primary load state caused by the reception of a remote boot request of the communication processor (DEUNA microcode). Communication Processor Boot -- A transition into the ready state from the primary load state following the reception of the memory load with transfer address message, as part of a remote boot request. The three conditions are distinguished by examining the state field of PCSR1. Set by the DEUNA, cleared by the port-driver.
<Ø7>	INTR	Interrupt Summary -- The logical OR of PCSRØ <15:Ø8>. Set by the DEUNA.
<Ø6>	INTE	Interrupt Enable -- Set or cleared by the port-driver, unchanged by the DEUNA.

NOTE

In order to overcome synchronization problems with the port command field when writing the INTE bit, the DEUNA hardware locks the port command field during write accesses that change the INTE bit from a one to a zero or change the INTE bit from a zero to a one. Issuing the DEUNA, a port command, and changing the state of the INTE bit must occur in two different write accesses.

Table 2-2 PCSRØ Bit Descriptions (Cont)

Bits	Name	Description	
<Ø5>	RSET	DEUNA Reset -- Clears the DEUNA and returns it to the power up state when written with a one byte port-driver. This bit is write-only. After a successful reset, PCSRØ <11> (DNI) = 1 and PCSRØ <Ø7> (INTR) = 1.	
<Ø4>	zero		
<Ø3:ØØ>	PORT_COMMAND		
	Ø Ø Ø Ø	NO-OP	No operation
	Ø Ø Ø 1	GET PCBB	Instructs the DEUNA to fetch the address of the port control block from PCSRs 2 and 3. The DEUNA accesses PCSRs over the UNIBUS conductor, and retains a copy of the address internally. If the address of the port control block is changed, this command must be repeated to inform the DEUNA.
	Ø Ø 1 Ø	GET CMD	Instructs the DEUNA to fetch and execute a command found in the first word of the port control block. The address of the port control block was obtained through the get PCBB command.
	Ø Ø 1 1	SELF TEST	Instructs the DEUNA to enter the reset state and execute self-test.
	Ø 1 Ø Ø	START	Enables transmission and reception of frames from the port-driver. This command is ignored by the DEUNA if it is in the running state. Clears any current buffer status that

Table 2-2 PCSRØ Bit Descriptions (Cont)

Bits	Name	Description	
			the DEUNA has stored internally and resets the ring pointers to the base addresses of the rings.
	Ø 1 Ø 1	BOOT	Instructs the DEUNA to enter the primary load state and initiate the down-line load of additional DEUNA microcode.
	Ø 1 1 Ø	Not Used	Reserved code, causes a NO-OP.
	Ø 1 1 1	Not Used	Reserved code, causes a NO-OP.
	1 Ø Ø Ø	PDMD	Polling Demand -- Instructs the DEUNA to check the descriptor rings. The DEUNA polls the receive descriptor ring only if it had not previously acquired a free buffer.
	1 Ø Ø 1	Not Used	Reserved code, causes a NO-OP, sets DNI.
	1 Ø 1 Ø	Not Used	Reserved code, causes a NO-OP, sets DNI.
	1 Ø 1 1	Not Used	Reserved code, causes a NO-OP, sets DNI.
	1 1 Ø Ø	Not used	Reserved code, causes a NO-OP, sets DNI.
	1 1 Ø 1	Not Used	Reserved code, causes a NO-OP, sets DNI.
	1 1 1 Ø	Not Used	Reserved code, causes a NO-OP, sets DNI.
	1 1 1 1	STOP	Suspends the operation of the DEUNA to transition to the ready state. Causes no action if the DEUNA is not in the running state.

2.2.2.2 Port Control and Status Register 1 (PCSR1) -- Figure 2-10 shows the format of PCSR1 and Table 2-3 lists the functions of the bits.

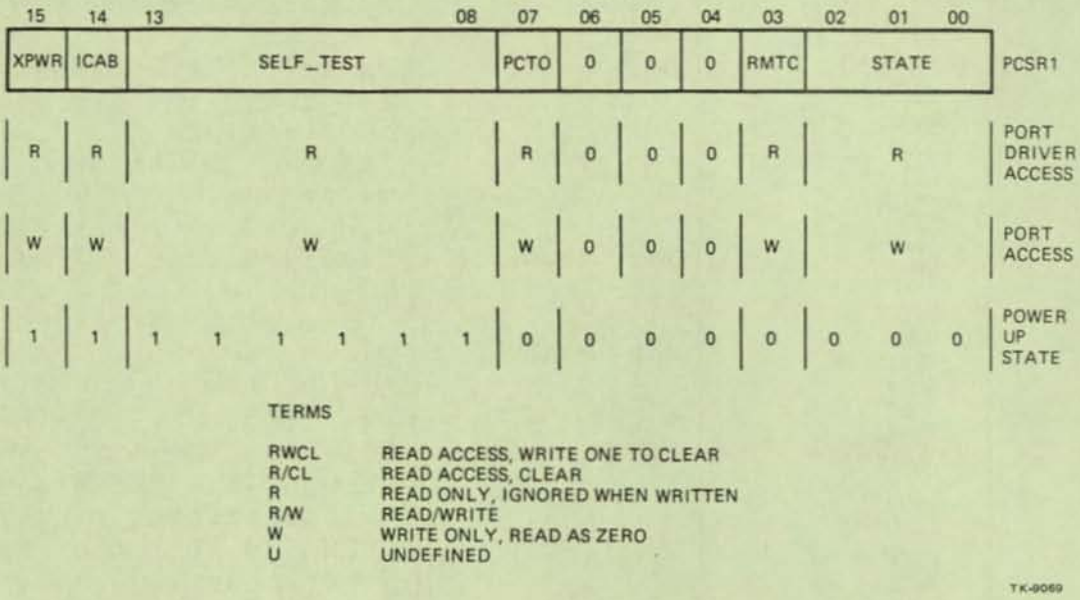


Figure 2-10 PCSR1 Format

Table 2-3 PCSR1 Bit Descriptions

Bits	Name	Description	
<15>	XPWR	Transceiver Power OK -- A zero indicates that a failure exists in either the transceiver power supply or the fuse on the link module.	
<14>	ICAB	Port/Link Cabling OK -- A zero indicates that the interconnecting cable between the port and link modules has a seating problem.	
<13:08>	SELF-TEST	Self-Test Error Code -- The encoded test of the DEUNA failed during self-test. A code of zero indicates no failure.	
<07>	PCTO	Port Command Timeout -- A UNIBUS timeout was encountered while executing a port command. Valid only after the PCEI bit of PCSR0 is set by the DEUNA. This bit is used to distinguish between a DEUNA failure to complete a port command due to a UNIBUS timeout or a function error.	
<06:04>	Zeros		
<03:00>	STATE	0 0 0 0	Reset
		0 0 0 1	Primary Load
		0 0 1 0	Ready
		0 0 1 1	Running
		0 1 0 0	Not Used
		0 1 0 1	UNIBUS Halted
		0 1 1 0	NI Halted
		0 1 1 1	NI and UNIBUS Halted
			Fatal internal error, that is parity error. An interrupt condition. When the DEUNA is in this state, the FATI bit of PCSR0 is also set. Cleared by the port-driver setting the RSET bit.
		1 1 1 1	Secondary Loader

2.2.2.3 Port Control and Status Register 2 (PCSR2) -- Figure 2-11 shows the format of PCSR2 and Table 2-4 lists the functions of the bits.

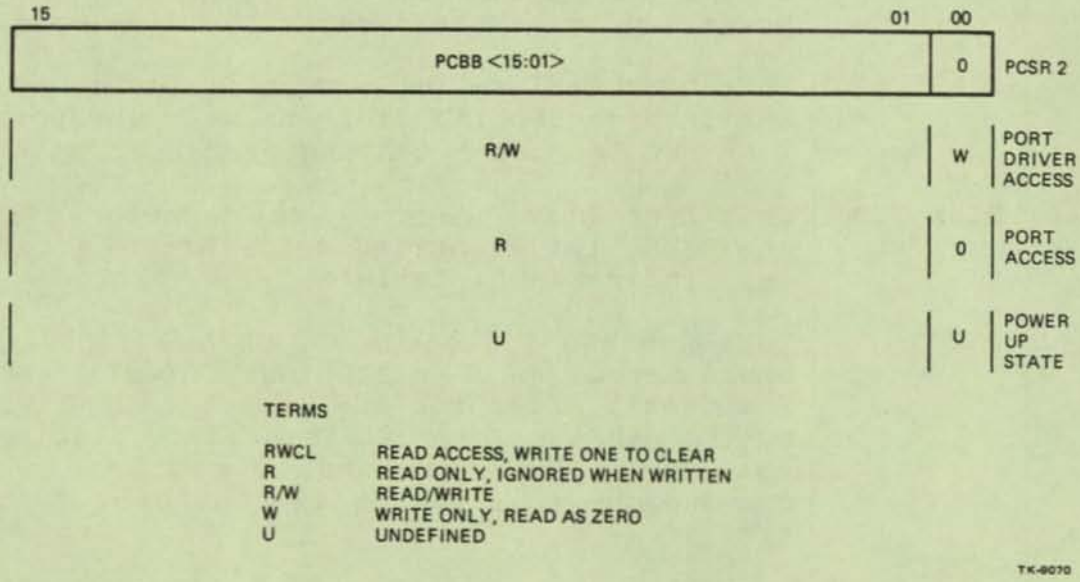


Figure 2-11 PCSR2 Format

Table 2-4 PCSR2 Bit Description

Bits	Name	Description
<15:00>	PCBB	The low order 16 bits of the address of the port control block base. The PCBB is read by the port as an even number.

2.2.2.4 Port Control and Status Register 3 (PCSR3) -- Figure 2-12 shows the format of PCSR3 and Table 2-5 lists the function of the bits.

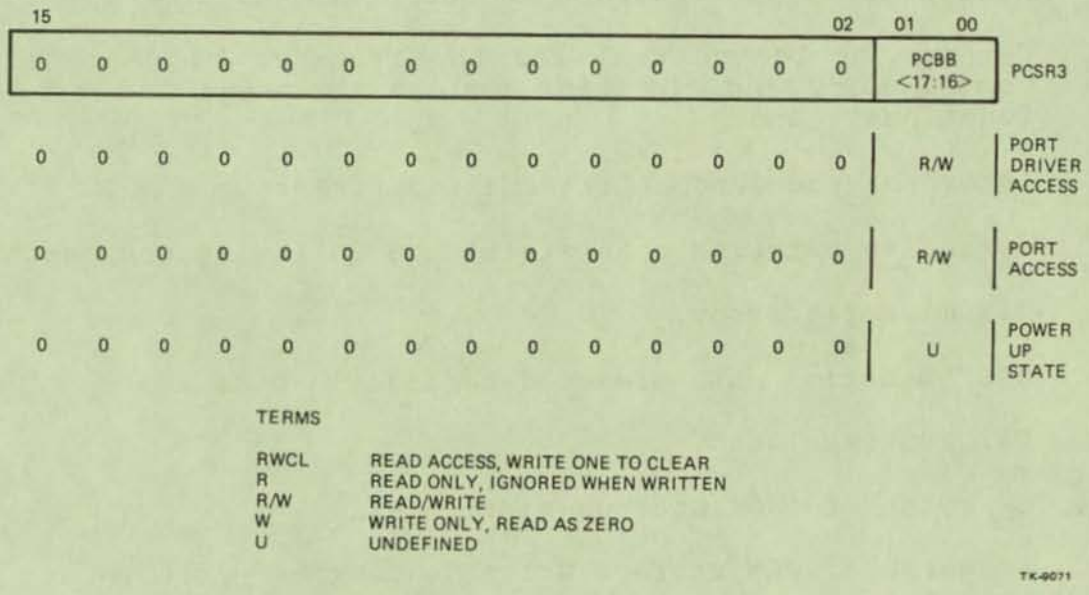


Figure 2-12 PCSR3 Format

Table 2-5 PCSR3 Bit Description

Bits	Name	Description
<15:02>		Zeros
<01:00>	PCBB	The high order 2 bits of the address of the port control block base.

2.3 MICROPROCESSOR SECTION

The function of the microprocessor section of the port module is to:

- Manage the ring structure in host memory,
- Set up the DMA control for the transfer of data between host memory and the link module (receive and transmit functions), and
- Interpret received or transmitted packets.

The microprocessor section consists of the following components:

1. T11 microprocessor,
2. DAL/BDAL-time multiplexed data/address bus,
3. T11 address latch,
4. 8K words of PROM storage-microcode,
5. 4K words of RAM storage-writable control store (WCS), and
6. Internal I/O decode-used when T11 has to access a register on the PORT module.

2.3.1 Microprocessor

The DEUNA uses a microprocessor located on the port module to control its operation. The microprocessor used is a DCT11-AA (T11). The T11 is a single chip microprocessor that uses the LSI-11 instruction set. The T11 communicates to the port module over a time multiplexed bidirectional bus called the data address lines (DAL). It also receives process and status information via a separate set of interrupt inputs. Each interrupt and its function is listed in Table 2-6.

The T11 can access a total of 32K words of memory. This address space is divided into areas for:

- Microprogram storage,
- Writable control store (WCS),

- Transmit and receive buffer space, and
- Input/output control.

Figure 2-13 shows the configuration of the T11's address space.

For more information on the operation of the T11 microprocessor, refer to the DCT11-AA Microprocessor User's Guide (EK-DCT11-UG).

Table 2-6 T11 Interrupts

Interrupt	Signal Name	Description
Receive Miss	MISS INTR	There is no receive buffer available for an incoming message.
Memory Parity	LNK MEM PAR ERR	There is a parity error in the link buffer memory.
PCSR Write	PCSR INTR	The host processor has written a command into PCSRØ.
UNIBUS Error	UBERR INTR	There is a UNIBUS timeout.
Transmit Done	XMIT DONE	The link has finished transmitting a buffer.
Receive Buffer Done	RCV BUFF DONE	There is a receive buffer waiting to be sent to host memory.
Timer	TIMER INTR	Interrupts T11 every second for timing information.
DMA Ready	DM INTR	DMA machine ready to be started.

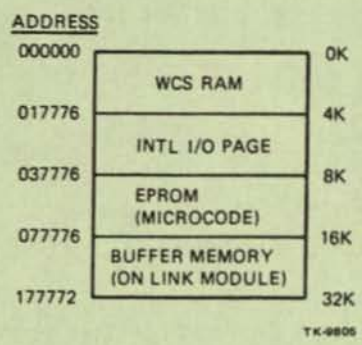


Figure 2-13 T11 Address Space

2.3.2 Internal Registers

The internal registers of the port module are used by the T11 for setting up and controlling the operation of the DEUNA. These registers reside in the I/O page of the T11.

The I/O decode logic of the port enables the selected internal register when it is addressed by the T11. This logic monitors the output of the T11 address latch and the read and write signals generated by the T11.

Table 2-7 gives a list of the internal register addresses and the type of access allowed.

Table 2-7 Internal Register Address Assignments

Address	Name	Access	Description
21000	PCSR0	R/W	Port control and status reg. 0
21002	DMCSR	R/W	DMA control and status reg.
21004	DMAT0	R/W	DMA-to-address register 0
21006	DMAT1	R/W	DMA-to-address register 1
21010	MDMA0	R/W	MicroCPU DMA-to-adrs. reg. 0
21012	MDMA1	R/W	MicroCPU DMA-to-adrs. reg. 1
21014	MDMDR0	R	MicroCPU DMA data reg. 0
21016	MDMDR1	R	MicroCPU DMA data reg. 1
21020	PCSR1	WO	Port control and status reg. 1
21022	DMAF	WO	DMA-from-address register
21024	DMWC	WO	DMA word count register
21026	MDMDR0	WO	Read Inc UB data port
21030	LTAC	WO	Link transmit adrs. counter reg.
21032	LRBAF	WO	Link rec. buffer address FIFO
21034	LCSR	WO	Link control and status reg.
21036	MDMDR1	WO	Write Dec. UB data port
21040	PCSRSW	RO	Port switchpack reg.
21042	UNUSED	RO	

Table 2-7 Internal Register Address Assignments (Cont)

Address	Name	Access	Description
21Ø44	LCBAF	RO	Link completed buffer add. FIFO
21Ø46	PCSR1	RO	Port cntl. and status reg. 1
21Ø5Ø	UNUSED	RO	
21Ø52	UNUSED	RO	
21Ø54	UNUSED	RO	
21Ø56	UNUSED	RO	
21Ø6Ø	PHYADØ	RO	Physical address byte Ø
21Ø62	PHYAD1	RO	Physical address byte 1
21Ø64	PHYAD2	RO	Physical address byte 2
21Ø66	PHYAD3	RO	Physical address byte 3
21Ø7Ø	PHYAD4	RO	Physical address byte 4
21Ø72	PHYAD5	RO	Physical address byte 5
21Ø74	PHYAD6	RO	Physical address byte 6
21Ø76	PHYAD7	RO	Physical address byte 7

2.3.3 Default Station Address (Physical Address)

The microprocessor section of the DEUNA contains a PROM which the T11 can read on power up to get the default address of the node. When the T11 reads the physical address from the PROM it transfers it to the station address RAM on the link module.

The physical address in the station address RAM can be changed by the host by a change physical address command.

2.3.4 Physical Address Registers

These registers are used to read the physical address from the physical address PROM. Figure 2-14 shows the configuration of these registers.

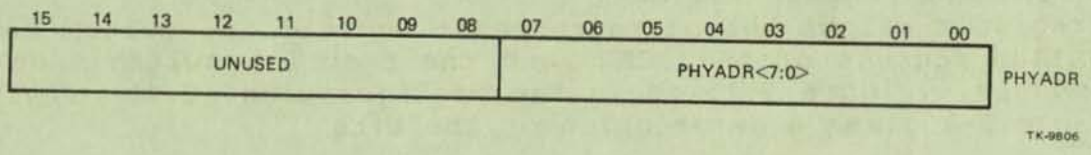


Figure 2-14 Physical Address Register Bit Configuration

2.3.5 Port Switchpack Register

This register allows the microprocessor to read the switch selected UNIBUS address of the PCSRs and the function switches on the port module. Figure 2-15 shows the configuration of the register and Table 2-8 gives a description of the bits.

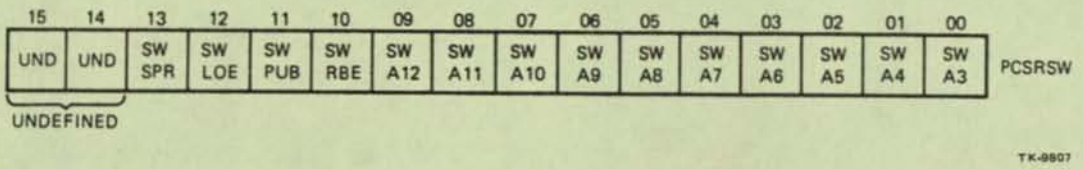


Figure 2-15 Port Switchpack Register Bit Configuration

Table 2-8 Port Switchpack Register Bit Descriptions

Bit	Field	Description
PCSRSW<09:00>	SW AXX	UNIBUS address address
PCSRSW<10>	SW RBE	Remote boot enable switch
PCSRSW<11>	SW PUB	Power-up boot switch
PCSRSW<12>	SW LOE	Loop-on-self-test error switch
PCSRSW<13>	SW SPR	Spare switch
PCSRSW<15:14>	UND	Undefined

2.3.6 Timer

The timer is made up of a one shot that generates an interrupt to the T11 every second. This allows the T11 to time events through the use of software routines.

2.3.7 Internal Buses

The port uses three sets of internal buses for the transfer of information within the DEUNA. These buses are:

1. DAL/BDAL (Data/Address Lines, Buffered Data/Address Lines).
   - Time multiplexed -- carry data during part of the timing cycle and address during the other part of the timing cycle.
   - BDAL is a buffered extension of the DAL for loading purposes.
2. T/F BUS (To/From Bus) -- transfers data between the UNIBUS bus and the DEUNA.
3. LMD BUS (Link Memory Data Bus) -- data bus to link memory buffers.
4. LINK MEM A (Link Memory Address Bus) -- address bus to link memory buffers.

2.4 LINK MEMORY CONTROL

The link memory section is the part of the port module which communicates with the link module. This section contains control for the 16K words of RAM that are located on the link module (parity generation and memory are on the port module). This memory is divided into 16 buffers that are used to buffer packets of data being transmitted to or received from the ETHERNET via the link module.

The link memory section contains the logic necessary to:

1. Arbitrate for use of the link memory,

2. Keep track of which buffers are available for use, and

3. Generate the memory addresses for writing or reading data from memory.

2.4.1 Link Memory Arbitration

Link memory is accessed by four different processes:

1. Link transmit state machine,

2. Link receive state machine,

3. DMA control, and

4. T11.

Arbitration for use of link memory by any of these processes is performed by the link memory arbitration PAL. A description for the PAL is given in the DEUNA engineering drawings.

2.4.2 Link Transmit Address Counter (LTAC)

The link transmit address counter is used when a transmit buffer is to be transmitted onto the ETHERNET.

The link transmit address counter consists of two sections:

1. Link Transmit Address Counter Register -- loaded by T11 with the four-bit buffer address.

2. Link Transmit Address Counter -- this is a 10-bit counter that is used to generate the lower 10 bits of the transmit buffer address.

They are configured as shown in Figure 2-16.

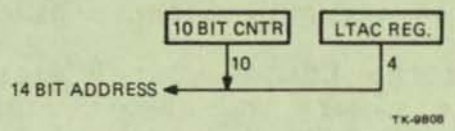


Figure 2-16 LTAC Configuration

When a transmit buffer in link memory is to be transmitted onto the ETHERNET, the following action takes place.

1. The T11 loads the LTAC register with the four-bit buffer address. This clears the 10-bit counter and notifies the transmit state machine on the link module that there is a buffer to be transmitted.

2. Transmit state machine increments counter by two after reading the word to be transmitted until the buffer is empty.

The transmit state machine can clear the 10-bit counter if it needs to do a transmit retry.

Figure 2-17 shows the bit configuration of the LTAC.

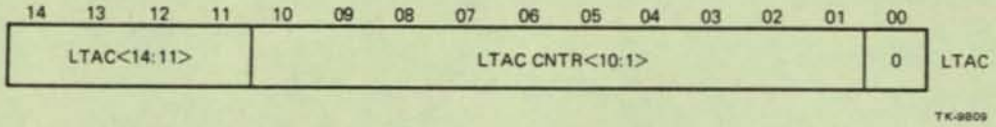


Figure 2-17 LTAC Bit Configuration

2.4.3 Link Receive Address Counter

The link receive address counter is used to generate the buffer addresses for messages received from the ETHERNET by the DEUNA.

The link receive address counter is made up of three sections.

1. The link receive buffer address FIFO (LRBAF),
2. The link completed buffer address FIFO (LCBAF), and
3. The link receive address counter.

They are configured as shown in Figure 2-18.

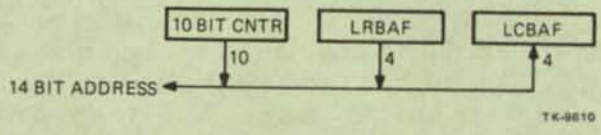


Figure 2-18 Receive Address Counter Configuration

The LRBAF and the LCBAF are four-bit by 64 location FIFOs. The LRBAF has the upper four bits of all available buffer addresses placed into it by the T11. When a receive buffer is needed by the link, the following functions are performed.

1. The address counter is cleared.

2. The buffer address at the output of the LRBAF and the output of the counter are used to generate the link memory address.

3. The counter is incremented until the buffer is completed.

4. When the buffer is completed by the link it advances the LRBAF which loads the address of the completed buffer into the LCBAF and clears the counter. The address bubbles through the FIFO.

5. When a buffer address is available at the output of the LCBAF the T11 is notified that there is a completed receive buffer. This is done by generating an interrupt (RCV BUF DONE) to the T11.

6. The T11 then processes the completed buffer and returns the buffer address to the LRBAF.

Figure 2-10 shows the configuration of the LRBAF and the LCBAF.

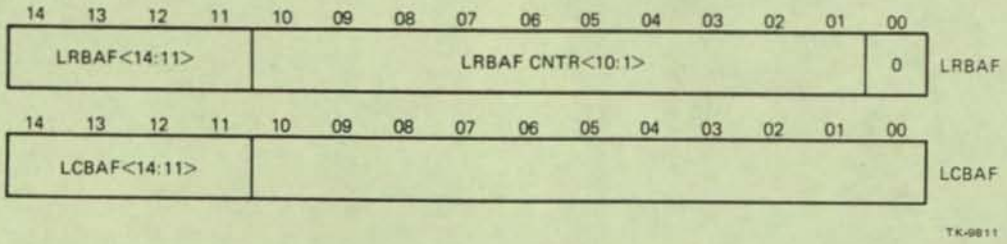


Figure 2-19 LRBAF, LCBAF Bit Configuration

2.4.4 T11 Addressing of Link Buffer Memory

When the T11 addresses link buffer memory, the memory arbitor arbitrates for use of the memory. When the T11 receives use of the memory the requested data is passed to the T11. During the data transfer the T11 is stalled.

2.4.5 Port-to-Link Interface

The DEUNA is comprised of two modules which have to be UNIBUS SPC compatible. This does not allow for a backplane interconnect. Therefore, the DEUNA port and link modules are connected by a Berg type connector over the handles cables. The signals on these cables comprise the port link interconnect.

The signals on these cables are broken down into five classes:

1. Link memory bus signals,
2. Link memory address control signals,
3. Link command register signals,
4. Link discrete status signals, and
5. Clock and initialize signals.

The following sections describe the port-to-link interface signals.

2.4.5.1 Link Memory Bus --

Signal	Source	Description
BUS LMD <15:00>	BIDIR	Link Data Bus -- Sixteen bidirectional data lines between the link and port modules.
LINK MEM A <14:01>	PORT	Link Memory Address Bus -- Fourteen address lines between the port and link modules.
BUS READ	BIDIR	Read/Write -- Used to indicate the direction of the transfer.
RX REQUEST	LINK	Receiver Request -- Used by the link receive state machine to request the link memory.
RX ACK	PORT	Receiver Acknowledge -- Used by the port to acknowledge the link request.
TX REQUEST	LINK	Transmit Request -- Used by the link transmit state machine to request the link memory.

TX ACK	PORT	Transmit Acknowledge -- Used by the port to acknowledge the link request.

2.4.5.2 Link Memory Address Control Signals -- A detailed description of these signals is given in Chapter 3 of this manual.

Signal	Source	Definition
INC TX POINTER	LINK	Increment Transmit Pointer -- Used by the link to increment the transmit address pointer.
RES TX POINTER	LINK	Restore Transmit Pointer -- Used by the link to restore the transmit address counter to the beginning address.
INC RX POINTER	LINK	Increment Receiver Pointer -- Used by the link to increment the receive address counter.
RES RX POINTER	LINK	Restore Receiver Pointer -- Used by the link to restore the receive pointer to the beginning.
ADV RX POINTER	LINK	Advance Receiver Pointer -- Used by the link to get the next receive address buffer.

2.4.5.3 Command Register Control --

Signal	Source	Description
CMDW	PORT	Command Register Write -- Enables the command register to be written from the link memory bus.
CMDE	PORT	Command Register Execute -- Tells the link to execute the command in the link command register.

2.4.5.4 Link Discrete Status --

Signal	Source	Description
CERR	LINK	Collision Test Error -- Indicates the collision output failed to activate during the collision test following a transmission (heartbeat).

SET MISS	LINK	Missed Packet -- Indicates that the link failed to write a received packet into link memory because a buffer was unavailable.
TATT	PORT	Transmitter Attention -- Tells the link that the port has completed a buffer for transmission.
TX DONE	LINK	Transmit Done -- Indication to the port that the link has finished transmitting a buffer.
ICAB1 ICAB2	LINK LINK	Installed Cable 1 & 2 -- Used by the port to re-ensure the interconnecting cable is plugged in properly.
FUSE CHECK	LINK	Transceiver Power -- Used by the port to check that the power to the transceiver is available.
RX FREE BUF	PORT	Receiver Buffer Free -- Used by the link to find out if there are any free receive buffers.
WR RESET	PORT	Reset -- Used by the link to do a UNIBUS reset.

2.4.5.5 Clock and Reset --

Signal	Source	Description
10MHZ	LINK	Clock -- 10 MHz square wave.
INIT	PORT	Buffered Initialize -- Buffered UNIBUS INIT.

CHAPTER 3
LINK MODULE
FUNCTIONAL DESCRIPTION

3.1 INTRODUCTION

The link module (M7793) is the interface between the DEUNA and the ETHERNET transceiver. It is microprogram controlled and provides the following functions.

- Physical channel interface
- Parallel-to-serial conversion of data on transmit
- Serial-to-parallel conversion of data on receive
- Collision detection and retry
- CRC generation and checking
- Station address detection
- Link memory bus control

The link in connection with the port provides the logic necessary to interface the UNIBUS Bus with the ETHERNET.

A functional block diagram of the link module is shown in Figure 3-1. The letters, in parenthesis, on the block diagram give the location of the logic for that functional block in the engineering drawings.

3.2 LINK MEMORY BUS

The link memory bus provides the communication path between the link module and the port module. The bus is made up of 54 lines that are divided into four signal groups.

- Memory Bus
- Discrete Control
- Discrete Status
- Clock

Tables 3-1, 3-2, 3-3, and 3-4 list the link memory bus signal names, their source, and a description of their function.

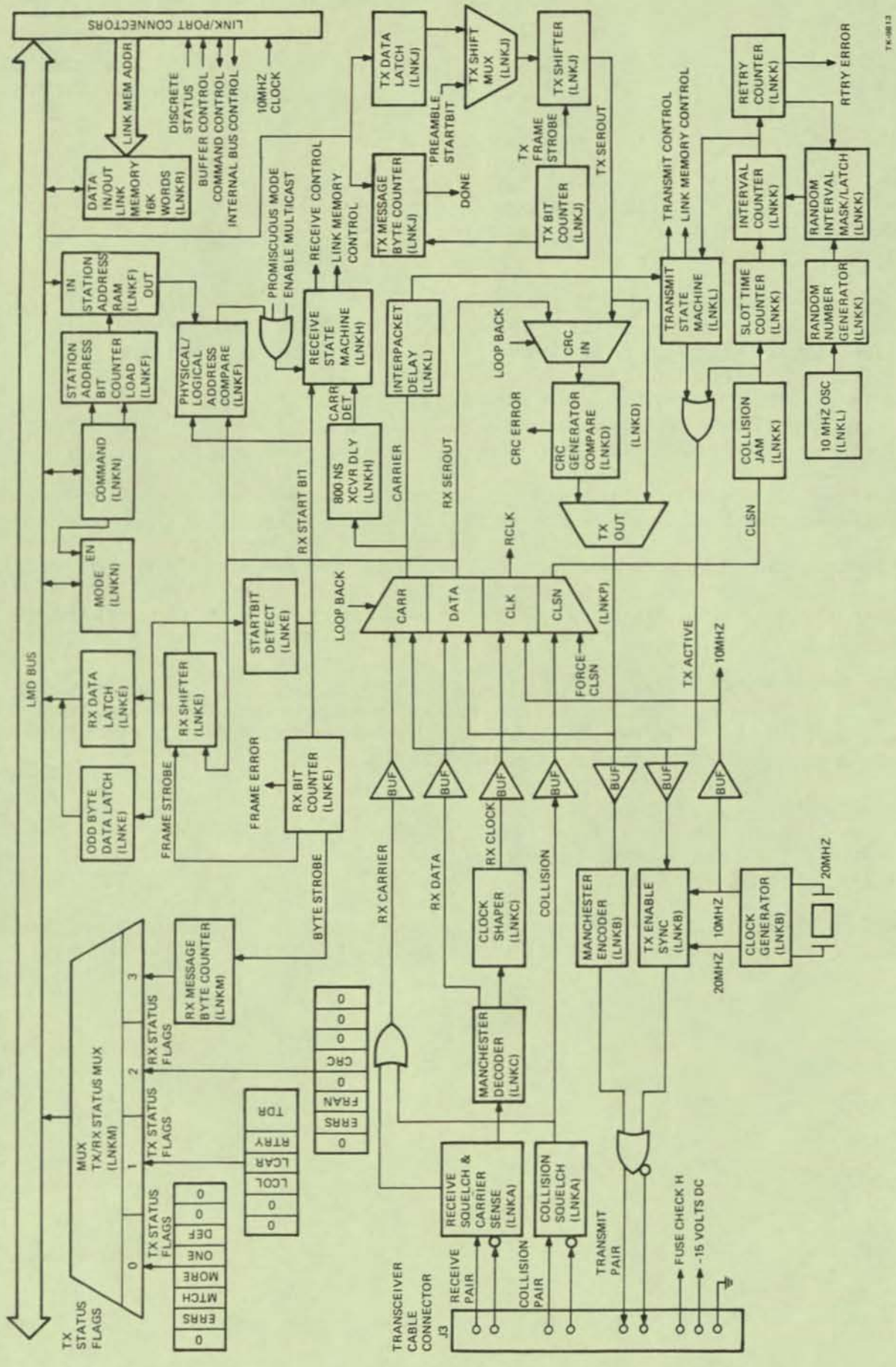


Figure 3-1 Link Module Functional Block Diagram

Table 3-1 Memory Bus Signals

Signal	Source	Description
LINK MEM <14:01>	Port	Link Memory Address Bus - Fourteen address lines used to address the memory buffers on the link module.
BUS LMD <15:00>	BIDIR	Link Memory Data Bus - Sixteen bidirectional data lines between the link and port module.
TX REQUEST	Link	Transmit Data Request - Used by the link to start arbitration for the Link Memory Bus.
RX REQUEST	Link	Receive Data Request - Used by the link to start arbitration for the link memory data bus.
TX ACK	Port	Transmit Data Acknowledge - Used by the port to inform the link that it has granted the link memory bus for a transmit operation.
RX ACK	Port	Receive Data Acknowledge - Used by the port to inform the link that it has granted the link memory bus receive operation.
BUS READ	BIDIR	Read/Write - Used to indicate the direction of the transfer. When set data is transferred from a link memory buffer.

Table 3-2 Discrete Control Bus Signals

Signal	Source	Description
INIT	Link	Synchronized Initialize - Clock synchronized power up initialize.
WR RESET	Port	Software Reset - Comes from port PCSR0.
CMDW	Port	Command Register Write - Enables the command register on the link to be written from the link memory bus. This signal is valid for 100 ns.
TATT	Port	Transmitter Attention - The port notifies the link that a transmit buffer is ready for transmission. Set by the port cleared by TX DONE.

Table 3-2 Discrete Control Bus Signals (Cont)

Signal	Source	Description
RES TX POINTER	Link	Reset Transmit Pointer - Tells the port to reset the transmit address pointer on the port. This signal is valid for 100 ns.
INC TX POINTER	Link	Increment Transmit Pointer - Tells the port to increment the transmit address pointer on the port. This signal is valid for 100 ns.
RES RX POINTER	Link	Reset Receive Pointer - Tells the port to reset the receiver address pointer on the port. This signal is valid for 100 ns.
INC RX POINTER	Link	Increment Receiver Pointer - Tells the port to increment the receiver address pointer on the port. This signal is valid for 100 ns.
ADV RX POINTER	Link	Advance Receiver Pointer - Tells the port to advance the receive buffer address pointer on the port. This signal is valid for 100 ns.
CABLE VERIFY IN	Port	Cable Verify Input - This circuit provides a closed loop electrical path with cable verify output that is used to indicate that the cable between the link and the port is installed and connected properly.
CABLE VERIFY OUT	Port	Cable Verify Output

Table 3-3 Discrete Status Bus Signals

Signal	Source	Description
CERR	Link	Collision Test Error - The transceiver collision output failed to activate during the collision test following transmission (heartbeat). Set during a collision test error. This signal is valid for 100 ns. This signal is valid for the H4000 or equivalent transceiver.
MISS	Link	Missed Packet - Receiver failed to write a packet addressed to the port into the link memory because a buffer was unavailable. This signal is valid for 100 ns.

Table 3-3 Discrete Status Bus Signals (Cont)

Signal	Source	Description
TX DONE	Link	Transmit Done - Indication to the port that the link has finished transmitting a buffer. This signal is valid for 100 ns.
FUSE CHECK	Link	Transceiver Power OK - A ONE indicates that a failure exists in either the transceiver power supply or in the cabling to the bulkhead assembly.
FREE RX BUFF	Port	Free Receiver Buffer - A buffer is available in the link memory to put an incoming packet. Set by the port, cleared by ADV RX pointer.

Table 3-4 Clock Signal

Signal	Source	Description
10 MHz	Link	Clock - The link clock is a 100 nanosecond square wave derived from a free running 10 MHz clock located in the ECL section of the link.

3.3 LINK REGISTERS

The operation of the link module is controlled by the port module though the use of two registers. The two registers are the link command register and the mode register. These registers are used to initialize, start, stop, and select the mode of operation of the link module. In addition to the command and mode registers, the link contains the station address RAM. The station address RAM is used to hold the addresses of the node for decoding by the address detection logic.

3.3.1 Command Register

The link command register is used by the port module to initialize, start, and stop the link module. This register is accessed by the port microprocessor by asserting CMDW H on the link memory bus. This register is write only by the port and is set to all zeros on power up or when initialized.

Figure 3-2 shows the format of the register and Table 3-5 describes the function of each bit.

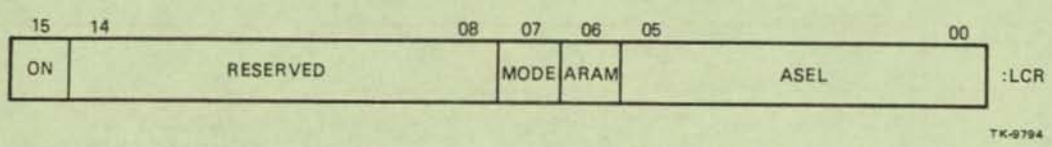


Figure 3-2 Format of Link Command Register

Table 3-5 Link Command Register Bit Descriptions

Bits	Field	Description
<15>	ON	Enable Link Module - When set, this bit enables both the receive and the transmit state machines. Set and cleared by the port. Powers on in the zero state.
<14:8>	Reserved	
<7>	Mode	Enable Mode Register - When set, this bit enables the write access of the mode register over the link memory data bus when CMDE H is asserted by the port. Set and cleared by the port.
<6>	ARAM	Enable Station Address RAM - When set, this bit allows the station address RAM to be written when CMDE H is asserted by the port. Set and cleared by the port.
<5:Ø>	ASEL	Address Select - Specifies the memory location within the station address RAM containing the physical and logical address: the data section of the station address begins at location ASEL=2Ø (octal). Set and cleared by the port.

NOTE
The first word, ARAMØ, of the Station Address RAM begins at location ASEL=2Ø (octal). This is due to the binary counter logic used in the address comparator section.

3.3.2 Link Mode Register

The port uses the mode register to control the transmit and receive operations of the link module. It is written when the mode bit of the link command register is set and bus signal CMDE is asserted. The register is set to all zeros on power up or when the link is initialized.

Figure 3-2 shows the format of the register and Table 3-6 describes the function of each bit.

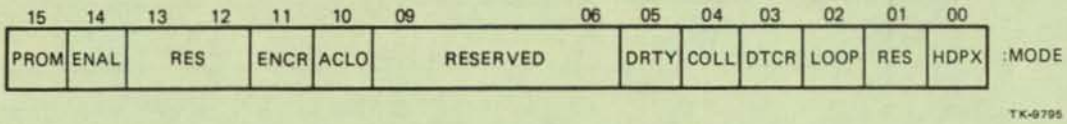


Figure 3-3 Link Mode Register Format

Table 3-6 Bit Descriptions for Link Mode Register

Bits	Field	Description
<15>	PROM	Instructs the link to accept all incoming frames regardless of the destination address field. Written and cleared by the port.
<14>	ENAL	Instructs the link to accept all incoming frames with multicast destinations. Written and cleared by the port.
<13:12>	RES	Reserved
<11>	ENCR	Enable Collision Test Error. When set, any collison test errors will be reported back to the port. Set and cleared by the port.
<10>	ACLO	Enable ACLO. When set, ACLO asserts ACLO on the UNIBUS Bus and disables INIT on the DEUNA. Set by the port cleared by the link.
<9:6>	RESERVED	
<5>	DRTY	Disable Retry Logic. When set, the link attempts only one transmission of a packet. This is a maintenance self-test function. Written and cleared by the port.
<4>	COLL	Simulate a collision on the wire during loopback mode. This is a maintenance self-test function. Written and cleared by the port.
<3>	DTCR	Disable Transmit CRC Logic. If DTCR=1, the CRC logic is dedicated to the receiver. If DTCR=0, the CRC logic is dedicated to the transmitter. This feature is used as a loopback maintenance function. Written and cleared by the port.
<2>	LOOP	Enable Loopback. When set, this bit enables loopback internal to the link, and the CRC logic dedicated to the receiver or transmitter as selected by DTCR. Written and cleared by the port.
<1>	RES	Reserved

Table 3-6 Bit Descriptions for Link Mode Register (Cont)

Bits	Field	Description
<Ø>	HDPX	Half-Duplex Mode. Indicates when clear that the link will receive messages transmitted to itself over the wire. Messages received in this manner do not undergo CRC check and a CRC error status is returned with them. Indicates when set that the link will not receive messages transmitted to itself. However, the link recognizes the transmitted messages as being addressed to itself and sets the MTCH bit in the transmit ring following the transmission attempt. Set and cleared by the port. Cleared upon power up.

3.3.3 Station Address RAM (ARAM)

The station address RAM contains the physical, logical, and broadcast addresses of the node. There can be a maximum of 12 addresses. Each address is 48 bits in length. These addresses are loaded by the port and read by the receive state machine.

Data is written to the ARAM over the link memory bus when the ARAM bit of the command register is set and the port asserts CMDE H.

Figure 3-4 shows the format of the station address RAM and Table 3-7 describes the register bits.

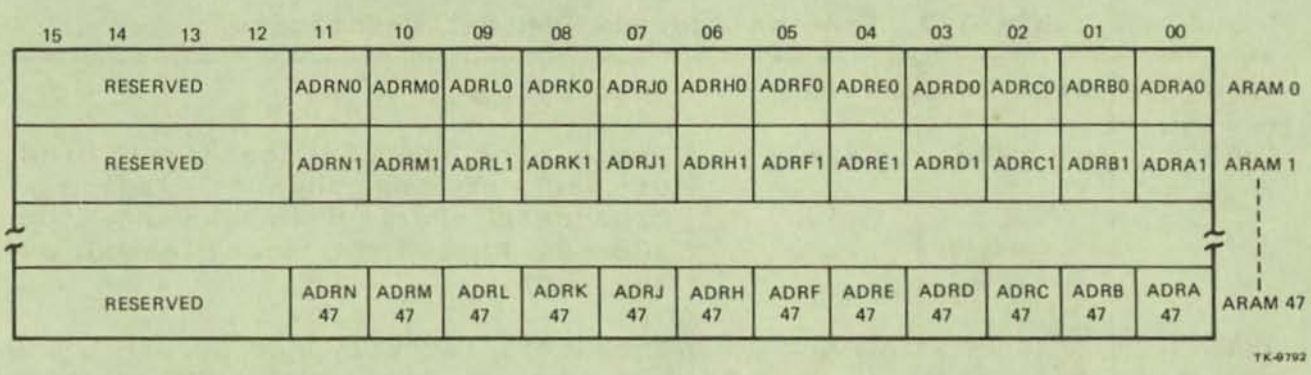


Figure 3-4 Station Address RAM Format

Table 3-7 Station Address RAM Bit Descriptions

Word	Bits	Field	Description
ARAM 0	<11:00>	ADRX0	These bits specify the first bits of each of the physical/logical/ broadcast address in the station address RAM. Set and cleared by the port.
ARAM 1	<11:00>	ADRX1	These bits specify the second bits of each of the physical/logical/ broadcast addresses in the station address RAM. Set and cleared by the port.
ARAM 2-ARAM 47	<11:00>	ADRX2-47	These bits specify the 2nd to 47th bits of each of the physical/logical/broadcast addresses in the station address RAM. Set and cleared by the port.

3.4 PHYSICAL CHANNEL INTERFACE

The physical channel is implemented in ECL technology and directly interfaces to the ETHERNET transceiver. The physical channel provides Manchester encoding and decoding of all serial data.

3.4.1 Transceiver Signals

The transceiver signals are those signals required by the H4000 transceiver. The following signals the ones used to communicate between the transceiver and link.

1. Collision Presence -- This signal is used to notify the transmit and retry logic of the link of a collision on the ETHERNET.

2. Receive -- This is the data received from the ETHERNET.

3. Transmit -- This is the data to be transmitted from the link.

4. Power -- Power required for the operation of the transceiver.

3.4.2 Receiver

3.4.2.1 Receiver Squelch and Carrier Sense -- Carrier sense is asserted when one or more stations are attempting transmission on the cable, regardless of whether the station sensing carrier is transmitting at that time. Carrier sense will turn on and remain on as long as data is present on the cable.

The carrier sense signal passes through the carrier MUX and is delayed 800 ns to allow proper synchronization of the preamble.

The delayed carrier signal is used as an input to:

- CRC checker,
- Receive shifter,
- Start bit detector, and
- Receive state machine.

The nondelayed carrier signal at the output of the carrier MUX is used as an input to:

- Time Domain Reflectometer (TDR), and
- Interpacket gap counter.

3.4.2.2 **Manchester Decoder** -- The Manchester decoder is used to separate the incoming phase encoded bit stream from the coaxial cable into a data stream and a clock signal. The Manchester data output is used as an input to the CRC checker and the RX shifter. The RX clock generated by the Manchester decoder is used as inputs to the clock shaper, CRC checker, and the RX shifter.

3.4.2.3 **Clock Shaper** -- The clock shaper is used to reshape the Manchester decoder clock output to ensure a minimum clock period and pulse width. The clock shaper protects the receive clock from distortion due to noise at the receive input.

3.4.2.4 **Collision Squelch** -- The collision squelch is similar in operation to the receive squelch. Its output is ORed with the output of the receive squelch circuitry.

Collision is asserted when two or more stations are attempting transmission on the coaxial cable, regardless of whether the station sensing collision is transmitting at that time. The collision squelch is used as an input to the TDR counter, collision jam, and the carrier multiplexer.

This signal is synchronized to the 10 MHz system clock by a dual rank synchronizer before entering any logic operating off the system clock.

3.4.3 **Transmitter**

The transmitter section of the physical channel interface on the link performs the encoding of data and enables the transmitter. This logic is comprised of the Manchester encoder and transmit enable circuitry.

3.4.3.1 **Manchester Encoder** -- The Manchester encoder is used to translate physically separate signals of lock and data into a single, self synchronizing serial bit stream, suitable for transmission on the coaxial cable. The inputs to the Manchester encoder are a 10 MHz clock and the output of the TX shifter. The Manchester encoder is controlled by the transmit state machine, and collision jam.

3.4.3.2 **Transmit Enable Sync** -- The TX enable sync logic enables the transmission of data when either the transmission slottime counter has expired and at the end of an interpacket delay.

TX enable sync is controlled by the transmit state machine and collision jam.

3.5 TRANSMIT SECTION

The transmit section of the link module prepares data for transmission onto the ETHERNET. After transmission, this logic will report status on the data transmitted. In order to accomplish this, the transmit section performs the following functions.

- Buffering of transmit data and status information between the host processor and the physical channel
- Parallel-to-serial data conversion
- Preamble generation
- CRC generation

The following paragraphs explain the functional sections of the transmit logic.

3.5.1 Data Section (Link Memory Buffers)

The link memory transmit buffer is made up of two sections, the data section and the status section.

There are two ways that the link module interacts with a link memory transmit buffer.

- Data Section -- This is the data being transmitted. It is written by the port and read by the link.
- Transmit Status Information Section -- Upon successful completion of transmission of a frame or after 16 unsuccessful attempts to transmit a frame, the link will write status information to the link memory transmit buffer.

Figure 3-5 shows the format of the transmit buffer before transmission and Table 3-8 gives a description of the buffer bits.

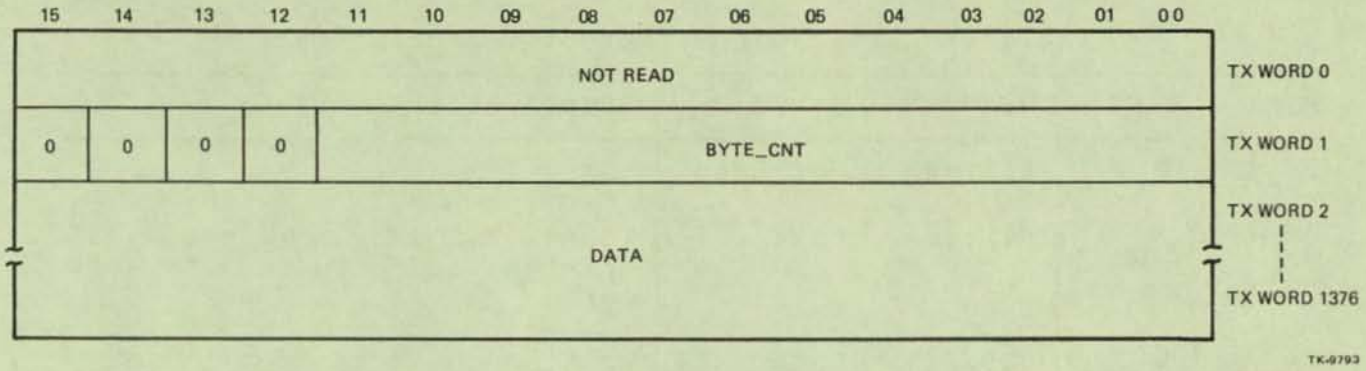


Figure 3-5 Transmit Buffer Format Before Transmission

Table 3-8 Transmit Buffer Bit Descriptions

Word	Bits	Field	Description
TX Word Ø	<15:ØØ>	Not Read	
RX Word 1	<11:ØØ>	BYTE CNT	Transmit Byte Count register. Written and cleared by the port.
TX Word 2- TX Word 1376	<15:ØØ>	Data	Written by the port. When transmitting an odd number of bytes, data found in bits <Ø7:ØØ> in the last entry location of the buffer is sent last.

3.5.2 TX Data Latch

The TX data latch is used to transfer transmit data from the link memory data bus to the TX shift multiplexer. The TX data latch is controlled by the transmit state machine and link memory bus controller.

3.5.3 TX Message Byte Counter

The TX byte counter is implemented as a 12-bit counter that is loaded by the transmit state machine from information contained in the link memory buffer. The TX byte counter contains the number of data bytes to be transmitted over the physical channel and is decremented to zero by 1Ø MHz clock. The count output of the TX byte count register is an input to the transmit state machine.

3.5.4 TX Frame and Byte Sync

The TX frame and byte sync signals provide a 1ØØ ns pulse signal every 16- and 8-clock periods respectively. The TX frame and byte sync signals are implemented as an UP counter and a terminal count detect circuit. These signals are initialized by the transmit state machine. During the odd byte case TX frame is advanced eight bits just before sending the four byte CRC. TX frame and byte sync are controlled by the 1Ø MHz clock and the TX byte count register.

3.5.5 TX Shift MUX

The TX shift multiplexer is used to selectively transfer a 16-bit word of either preamble or transmit data to the TX shifter. The TX shift multiplexer is controlled by the transmit state machine.

Sel 1	Output
Sel TX Data L=Ø	Transmit Data
Sel TX Data L=1	Preamble Data

TX Multiplexer Selection Chart

3.5.6 TX Shifter

The TX shifter converts parallel data from the TX shift multiplexer into a serial output data stream that goes to the CRC generator and the Manchester encoder. The TX shifter is parallel loaded and is controlled by the TX clock and the transmit state machine.

3.5.7 TX Output MUX

The TX output MUX is used to select the output of the TX shifter or the CRC generator for output to the transmitter.

3.5.8 TX Status Information

The transmit status information is written into the link memory transmit buffer by the link either after a successful attempt to transmit a frame or after 16 attempts to transmit a frame have failed. The first two words of the transmit buffer are used to store this information. Figure 3-6 shows the format of these words in relation to the rest of the buffer. Table 3-9 describes each of the status bits.

For information about the transmit data buffer before transmission, refer to paragraph 3.5.1 of this chapter.

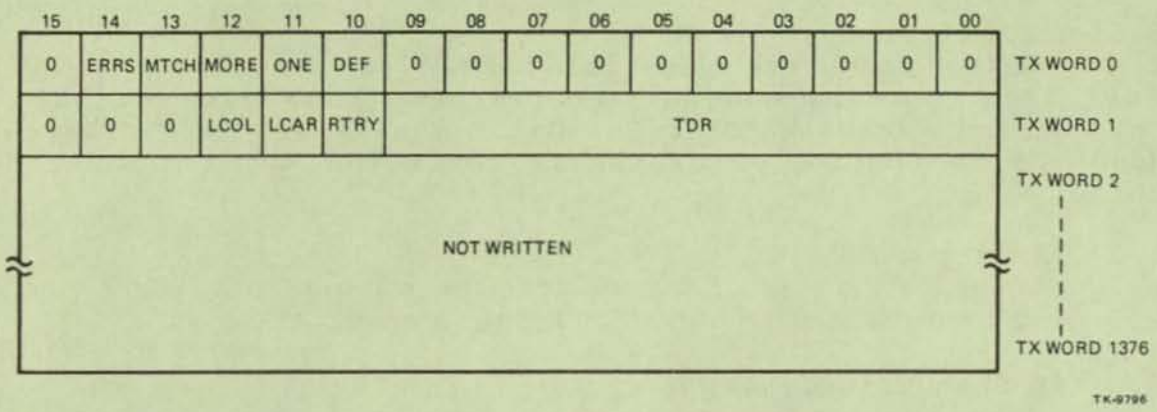


Figure 3-6 Transmit Buffer Format

Table 3-9 Transmit Status Bit Descriptions

Word	Bits	Field	Description
TX Word Ø	<14>	ERRS	Error Summary - The logical OR of LCOL, LCAR, or RTRY was set. Written and cleared by the link.
TX Word Ø	<13>	MTCH	Station Match - Set by the link when the destination address of the message matches one of the addresses of the UNA.
TX Word Ø	<12>	MORE	Multiple Retries Needed - Set when more than one and less than 16 retries were needed to transmit a frame. Written and cleared by the link.
TX Word Ø	<11>	ONE	One Collision - Set when exactly one retry was needed to transmit a frame. Written and cleared by the link.
TX Word Ø	<1Ø>	DEF	Deferred - Set when the transmitter experienced no collisions but had to defer while trying to transmit a frame. Written and cleared by the link.
TX Word 1	<12>	LCOL	Late Collision - A collision has occurred after the slot time of the channel has elapsed. Written and cleared by the link.
TX Word 1	<11>	LCAR	Loss of Carrier - Carrier was either not present on the channel during transmission or transceiver power was not present. Written and cleared by the link.
TX Word 1	<1Ø>	RTRY	Retry - Transmitter has failed in 16 attempts to transmit the frame due to collisions on the medium. Written and cleared by the link.
TX Word 1	<9:Ø>	TDR	Time Domain Reflectometry Value - Valid only when RTRY or LCAR is set. Written and cleared by the link. All ones indicates an overflow condition.

Table 3-9 Transmit Status Bit Descriptions (Cont)

Word	Bits	Field	Description
TX Word 2- TX Word 1376	<15:00>	NOT WRITTEN	

3.5.9 Transmit State Machine

The transmit state machine controls are:

- The link data path during transmission, and
- The access of the buffers in the link memory.

The transmit state machine is implemented in PALs and consists of the following states:

1. Transmit Enable State - Entered by the transmit state machine when the port asserts TATT. Exited after carrier has gone away and the interpacket gap timer has elapsed.

2. Preamble/Start Bit - Entered after the transmit enable state. The preamble consists of 64 bits of alternating ones and zeros ending in a double one. The preamble is loaded into the TX shifter as four 16-bit words to be shifted serially out onto the wire.

   If the transmitter is enabled and there are no collisions on the wire, the transmit state machine will increment the TX pointer and then load the transmit byte count during the loading of the first word of preamble.

3. Data State - Entered after the fourth word of preamble is loaded into the TX shifter. During this state, data is transferred from the link memory data bus to the TX shifter to be serially shifted onto the wire. This state remains active until the TX byte count register has expired or a collision occurs.

4. CRC State - Entered after the data state if the DTCR bit is not set and exited after 32 bits of CRC are transmitted or a collision occurs.

5. Write Status - Entered after the CRC state. During this state the transmit state machine writes the transmit status into the link memory buffer residing on the port. If there are no collisions and no collision errors, then the transmit state machine resets the TX pointer, write status Word 0, and write status Word 1.

6. Retry - Entered if there is a collision on the wire. During this state the transmit state machine continues transmitting, a process known as jamming, for 32-bit times. At the end of enforcing the jam, the transmit state machine delays for attempting to retransmit again. This delay is based upon some multiple number of slot times. This state is further described in the RETRY section.

7. Done.

3.6 RETRY LOGIC

The retry logic controls the scheduling of the retransmission of packets when a collision has occurred. This logic uses the binary exponential backoff algorithm. Basically the algorithm waits a generally increasing random number of slot times before retransmission. The random number must be between Ø and 2**K, where K is the min(n,1Ø) for the nth transmission.

3.6.1 Collision Jam

Collision jam keeps the transmitter on for 32-bit times after a collision is detected and the preamble has finished transmitting.

Collision jam is asserted by the leading edge of collision detect and is used as an input to the retry slot time counter, the carrier multiplexer, and the TX enable sync.

3.6.2 Slot Time Counter

The slot time counter is a 51.2 microseconds modulus counter. The slot time counter begins its count upon recognition that the retry state machine is in the backoff state. The output of the slot time counter is used as an input to the retry interval counter.

3.6.3 1Ø MHz Oscillator

The 1Ø MHz oscillator is implemented as an RC voltage controlled oscillator. The oscillator provides the clock for the random number generator.

An RC oscillator is used so that the probability of the retry logic of other nodes on the ETHERNET becoming synchronized is decreased.

3.6.4 Random Number Generator

The random number generator is implemented as a 1Ø-bit binary counter that continuously counts from power-up and is never reset. The 1Ø outputs of the random number generator are the inputs to the random interval mask/latch.

3.6.5 Random Interval Mask/Latch

The random interval mask is combinational logic which masks out bits in the random number according to the number of retries needed to successfully transmit a packet. The mask ensures that the random number is between Ø and 2**K, where K is the min(n,1Ø) for the nth transmission. Inputs to the random interval mask are

the 10 output lines from the random number generator, and the retry counter. The output from the mask is latched into the random interval latch.

3.6.6 Interval Counter

The retry interval counter is a binary counter that counts the number of slot times that have elapsed. Counting ceases when the number of slot time intervals is equal to the number of random slot times provided by the random interval mask/latch.

3.6.7 Retry Counter

The retry counter counts the number of retransmissions that have occurred. The retry counter is incremented by the interval counter and reset by the transmit state machine. The outputs of the retry counter are the inputs to the transmit status register, and the retry interval mask.

3.6.8 Retry State Machine

The retry state machine, not shown on the block diagram, is used to control the retry process during a collision. The retry state machine is implemented in a PAL and consists of the following states.

1. Jam State - This state is entered if a collision is encountered during transmission of data on the wire. During this state, the transmit section remains transmitting for 32-bit times if the collision occurred during the data state. If, however, the collision occurred during the preamble state, the transmitter will continue transmitting the preamble and then jam for 32 bits. During the jam state, the CRC is disabled.

2. Backoff State - This state is entered after the jam state.

   At the end of enforcing jam, the transmitter delays before attempting to retransmit again. This delay is an integral multiple of slot times. The number of slot times to delay before the nth retransmission attempt is chosen as a uniformly distributed random integer r in the range of 0<= r <= 2**k where k=min(N,10). If all 16 attempts to transmit fail, the event is reported back as a RTRY error.

3. Force Collision - This is a maintenance self-test function and is valid if the loop and COLL bits in the mode register are both set and the function is reset by clearing these bits.

   Force test allows the microprogrammer to single step through the collision retry algorithm one attempt at a time by simulating a collision on the wire without being physically linked to it. Each attempt to transmit forces a collision internally to the link module. The transmit

state machine then goes through the collision jam and retry states. The retry counter is then incremented and the transmit state machine then writes the appropriate status information to link memory.

3.6.9 Time Domain Reflectometry

The TDR counter is ten bits wide modulus counter. It is cleared by the transmit state machine and counts upon the recognition of carrier during transmission. Counting ceases either due to a collision, loss of carrier, or if it has reached its modulus. The value of the TDR is written into memory by the microprocessor. TDR is used to determine the location of suspected cable faults.

3.7 RECEIVE SECTION

The receive logic on the link module is used to:

- Convert serial data to parallel data
- Count the number of bytes received
- Write the received data into the link memory buffers
- Write status information and message length into the receive buffers

3.7.1 Data Section (Link Memory Buffers)

The link memory receive buffer is located in link memory and is written only by the link. It contains the data and status information provided by the physical channel and the receiver state machine. Figure 3-7 shows the format of the link memory receive buffer. Table 3-10 describes the status and data bits of the buffer.

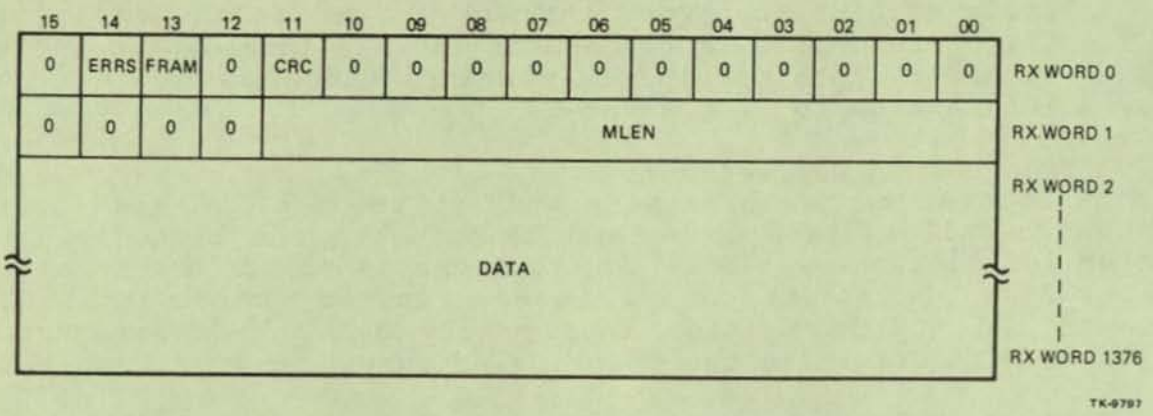


Figure 3-7 Receive Buffer Format

Table 3-10 Receive Buffer Status and Data Bit Description

Word	Bits	Field	Description
RX Word 0	<14>	ERRS	Error Summary - The logical OR of FRAM, CRC. Written and cleared by the link.
RX Word 0	<13>	FRAM	Frame Error - Indicates that the incoming frame contained a non integer multiple of 8 bits and the CRC value at the last 8-bit boundary was in error. Written and cleared by the link.
RX Word 0	<11>	CRC	Cyclical Redundancy Check - Frame check error, data is not valid. Written and cleared by the link.
RX Word 1	<11:0>	MLEN	Receiver Byte Count Register - Written by the link. This register latches at all ones indicating a babbling node on the network or broken byte count detect logic.
RX Word 2- RX Word 1376	<15:00>	DATA	Written by the link. During the odd byte case, data would be found in bits <7:00> of the last word written.

3.7.2 Receive MUX

The RX multiplexer is used to select data from the ETHERNET or the output of the TX output MUX. The RX multiplexer is controlled by a loopback signal.

Sel 1	Output
Loopback=0	Receive Data
Loopback=1	Transmit Data

Multiplexer Output Selection Chart

3.7.3 Receive Shifter

The RX shifter is a 16-bit wide device that frames the incoming serial bit stream into a word stream. A normal reception sequence consists of the continuous shifting of the alternating "ones" and "zeros" that comprise the preamble through the shifter. Upon the recognition of the double "one" pattern that indicates the start of data, the data is then framed into the RX shifter.

The RX shifter is controlled by the receive state machine and RX clock. The output of the RX shifter is transferred to the RX data latch.

3.7.4 RX Data Latch

The RX data latch is used to transfer receive data from the RX shifter to the link memory data bus. The RX latch is implemented as three 8-bit counters whose inputs are the outputs of the RX shifter. Two of these latches work together to transfer data found on 16-bit boundaries to the link memory data bus. The third latch, strobed every 8 bits, is used during the odd byte case to transfer the last byte of data to the link memory data bus. Data strobed into the RX latch is transferred onto the link memory data bus using the handshake provided by the link memory bus controller. The RX latch is controlled by the receive state machine, carrier, bit Ø of the RX byte count register (to detect odd bytes), and the link memory data bus control logic.

3.7.5 RX Frame and Byte Sync

The RX frame and byte sync signals provide a 1ØØ ns pulse signal every 16- and 8-clock period respectively. The RX frame and byte sync signals are implemented as an up counter and a terminal count detect circuit. These signals are initialized by the recognition of start bit. The frame and byte sync are further gated with RCLK L to minimize skew.

3.7.6 RX Byte Counter

The RX byte count is implemented as a 12-bit counter that may be accessed over the link memory data bus. The RX byte counter contains the number of data bytes that are received from the physical channel.

The RX byte counter is incremented by RX clock and is controlled by the receive state machine. The counter will latch up to all ones for an overflow condition. The output of the RX byte counter is passed to the link memory data bus by the TX/RX status multiplexer.

3.7.7 Receive State Machine

Control of the link data path during reception is provided by the receive state machine. The receiver state machine is implemented in PALs and consists of the following states.

1. Receiver Enabled - Entered by the receiver state machine upon setting the on bit in the command register, or upon completing the transfer of an incoming frame, or after the bad packet state, or after the miss state. The receiver state machine stays in this state until carrier is no longer present on the wire.

2. No Carrier - Entered after receive enabled state when carrier is no longer present on the wire and exited when the carrier signal comes up.

3. Carrier - Entered after the no carrier state upon presence of carrier on the wire. During this state the receive state machine looks for a valid preamble, a free receiver buffer, and checks for a runt, no address match, or a start bit.

4. Pointer Reset - Entered after the no carrier state upon presence of carrier on the wire. This state resets the receiver address pointer on the link memory. Exited after one clock period.

5. Pointer Increment - Entered after the pointer reset state. This state increments the receiver address pointer on the link memory to point to the data section of the buffer.

6. Data Request - Entered during the carrier state after recognition of a valid start bit. Exited upon loss of carrier and a bad packet, or status write and a valid packet. During this state, the receive state machine transfers data from the wire to link memory and increments the RX pointers.

7. Bad Packet - Entered after the carrier state if either the packet was less than 64 bytes (runt packet) or the packet did not pass address recognition.

8. Valid Packet - Entered after the carrier state if the packet passed address detection was not a runt packet and there was a free receiver buffer to put the packet in.

9. Miss - Entered after the carrier state if the packet passed address detection, was not a runt packet, and there was no free receiver buffer available.

10. Write Status - Entered after the valid packet state. During this state, status information is written to the link memory buffer.

11. End of Reception - Entered after the write status state. Exited after one clock period.

3.7.8 Interpacket Delay

The interpacket delay prevents the transmission of data for at least 9.6 microseconds after the last carrier detect.

The interpacket delay is asserted by the trailing edge of carrier and is used as an input to the transmit state machine.

3.8 STATION ADDRESS DECODE

The station address detect logic checks the destination address of the incoming packet to determine if the packet is addressed to this node. A packet passes address detection if at least one of the following is true:

1. Logical address match: the destination address of the packet exactly matches one of the 11 possible logical addresses of the node.

2. Physical address match: the destination address of the packet exactly matches the physical address of the node.

3. Promiscuous mode: this mode accepts all packets regardless of the destination address.

4. Enable all multicast: this mode accepts all packets with multicast address regardless of the destination address.

The station address match is then used by the receive state machine. This signal is synchronized to the 10 MHz system clock by a dual rank synchronizer before entering any logic operating off the system clock.

3.8.1 Physical/Logical Address Detection

Physical/logical address detection is done by serial comparing each bit of the destination address on the wire against the contents of the 48*12 station address RAM. The serial compares of the physical and logical addresses are all done in parallel and are enabled by the receiver state machine.

The physical/logical address is written into the station address RAM by 48 sequential memory writes over the link memory data bus.

3.8.2 Promiscuous Mode

In this mode the receiver logic will accept all packets that are sent, regardless of the destination field of the packet.

3.8.3 Enable All Multicast

This mode accepts all packets with multicast addresses regardless of the destination address.

3.9 CRC LOGIC

The CRC logic implements the 32-bit CRC using the AUTODIN-II polynomial as the generating polynomial. The generation and checking of the CRC is done using a 32-bit register implemented in PALs which acts as a shift register, XOR gates, and combinational logic for control.

CRC logic is half-duplex during transmission, reception, and loopback. During loopback the CRC logic is dedicated to the transmit section of the link unless DTCR is set in the link mode register. (If DTCR is set, the CRC logic is dedicated to the receiver.)

For checking the CRC at the end of a packet, a residue detector is used to monitor the data as it shifts through the CRC generator. The residue detector is strobed on 8-bit boundaries. If there are no CRC errors, the output of the CRC to the residue detector is the value of:

11000111 00000100 11011101 01111011

(Where the leftmost bit corresponds to the X**31 term of the polynomial and the rightmost to the X**0 term.) Any other value indicates an error.

The input to the CRC generator is either the transmit data stream or the receive data stream. The CRC generator/checker is controlled by the transmit state machine, receive state machine, RX clock, TX clock, and loopback.

NOTE
Output of the CRC PALs are asserted "low".

3.10 TX/RX STATUS

The TX/RX status multiplexer is used to transfer status information from TX Word 0, TX Word 1, RX Word 0, or RX Word 1 to the link memory data bus for writing into the apropriate link memory buffer that resides on the port. The TX/RX status multiplexer is enabled and controlled by the link memory data bus control logic and the receive and transmit state machines.

Sel 1	Sel 0	Output
0	0	TX Status Word 0
0	1	TDR, TX Status Word 1
1	0	RX Status Word 0
1	1	RX byte count

TX/RX Status Multiplexer Selection Chart

3.11 LINK MEMORY

The link memory section is the part of the link module that communicates with the port module. This section contains 16K words of RAM which is used by the link module to buffer packets that are to be received or transmitted on the ETHERNET. This 16K of memory is broken down into sixteen 1536 byte buffers. The first four bytes of each buffer are used to convey status information about the packet.

Addressing of link memory is provided by the port module over one of two over-the-top cables connecting the port to the link.

This memory is arbitrated for and accessed by four different processes:

1. Link transmit state machine.
2. Link receive state machine.
3. DMA engine (described in the UNA Port Module Functional Description).

4. T11 (described in the UNA Port Module Function Description).

The link memory arbitrator resides on the port module.

3.12 LINK MEMORY BUS CONTROLLER

The link memory bus controller is a simple state machine that provides the necessary handshake involved in transferring data between link memory and the transmitter or receiver.

CHAPTER 4
MICROCODE

4.1 OVERVIEW

The microcode provides the microcode instructions necessary to control the T11 microprocessor contained on the port module. This code in conjunction with the T11 is responsible for data encapsulation and decapsulation, data link management, and all channel access functions. This allows for maximum data throughput with a minimum of intervention by the host processor.

In order to understand how the microcode of the DEUNA functions, it is necessary to understand how the DEUNA is programmed. Information on how the DEUNA is programmed can be found in Chapter 3 of the DEUNA User's Guide (EK-DEUNA-UG).

4.2 STRUCTURE

The microcode of the DEUNA is structured as a series of concurrent, cooperating processes that are executed under the control of a supervisor program. These processes are created at the time the DEUNA is powered up and are entirely self-contained. Each process is capable of performing its specific function without assistance from any other process.

4.3 SUPERVISOR

The supervisor is made up of the routines that are needed to:

- Control the scheduling of the different processes used in the DEUNA, and
- Maintain the status and data needed for the operation of the DEUNA.

There are two different types of routines executed by the supervisor, interrupt routines and subroutines.

1. Interrupt Routines -- These routines are executed as a result of a specific interrupt generated by the hardware of the DEUNA. These routines will normally run to completion at the level of the interrupt.
2. Subroutines -- These routines are called by a specific process while that process is running. These routines are accessed by way of a dispatch table contained in ROM. This table is written into the WCS of the DEUNA during initialization.

4.3.1 Initialization

The initialize routine is the first supervisor routine to be executed after the completion of self test.

The function of the initialize routine is to:

1. Reset the hardware of the DEUNA to a known state;

2. Build the supervisor dispatch tables in Writeable Control Store (WCS);

3. Create the data structures in WCS required by the micro-code;

4. Clear all the internal counters, the multicast list, mode register, and descriptor ring lengths;

5. Load the physical and broadcast address into the station address RAM on the link module;

6. Enable all hardware interrupts;

7. Load the address of the receive buffers and allocate the transmit buffers; and

8. Start the null process (this executes at priority zero).

4.3.2 Scheduling

The supervisor performs the scheduling of processes through the use of a request mask. When the T-11 receives an interrupt requesting a particular process to be run, the interrupt service routine sets a bit in the request mask. The next time the null process runs it will scan the request mask to see if any low priority processes are scheduled to be run.

All the processes will execute at the CPU priority of zero with the exception of the datagram receive process. As a result there is no context switching between low priority processes. This means that each process, with the exception of the datagram receive process, will run to completion before the request mask is scanned again. The receive process runs at the priority of the hardware interrupt.

When a process has completed it will return to the supervisor by executing an RTI instruction or calling the supervisor command complete routine.

Table 4-1 gives a list of the processes and the order of execution (priority).

Table 4-1 Priority of Processes

Process	Priority
Datagram Receive	1
Port Command	2
Timer	3
Loop and Maintenance	4
Datagram Transmit	5
Null	6

4.3.3 Datagram Receive Process

The datagram receive process is used to transfer receive datagrams from the receive buffers on the link to host memory. This process is the highest priority process because it has the greatest impact on the throughput of the Ethernet and the DEUNA.

The receive process is started by the buffer filled interrupt or by the START port command. The process is ended when:

1. The datagram was written into host memory
2. Status information was written into the descriptor
3. A new buffer descriptor was read from the ring entry

The receive process executes at a hardware priority level of five and can only be interrupted by DMA done, power failure, or errors. Because the amount of processing performed is short (get buffer, start DMA machine), it is possible for other processes to run between the time the DMA machine is started and the DMA done interrupt is generated.

The receive process is initiated in two ways:

- A datagram was received and an interrupt was sent to the T-11.
- A poll demand or start command was received from the host. Either causes an interrupt to generate, and the receive process to start.

The receive process performs the following:

1. Poll receive ring to get a buffer in host memory.
2. Load and start DMA machine.

3. When DMA is done, execute an RTI instruction or run the null process.

Figure 4-1 and Figure 4-2 show the function of the microcode for the receive process.

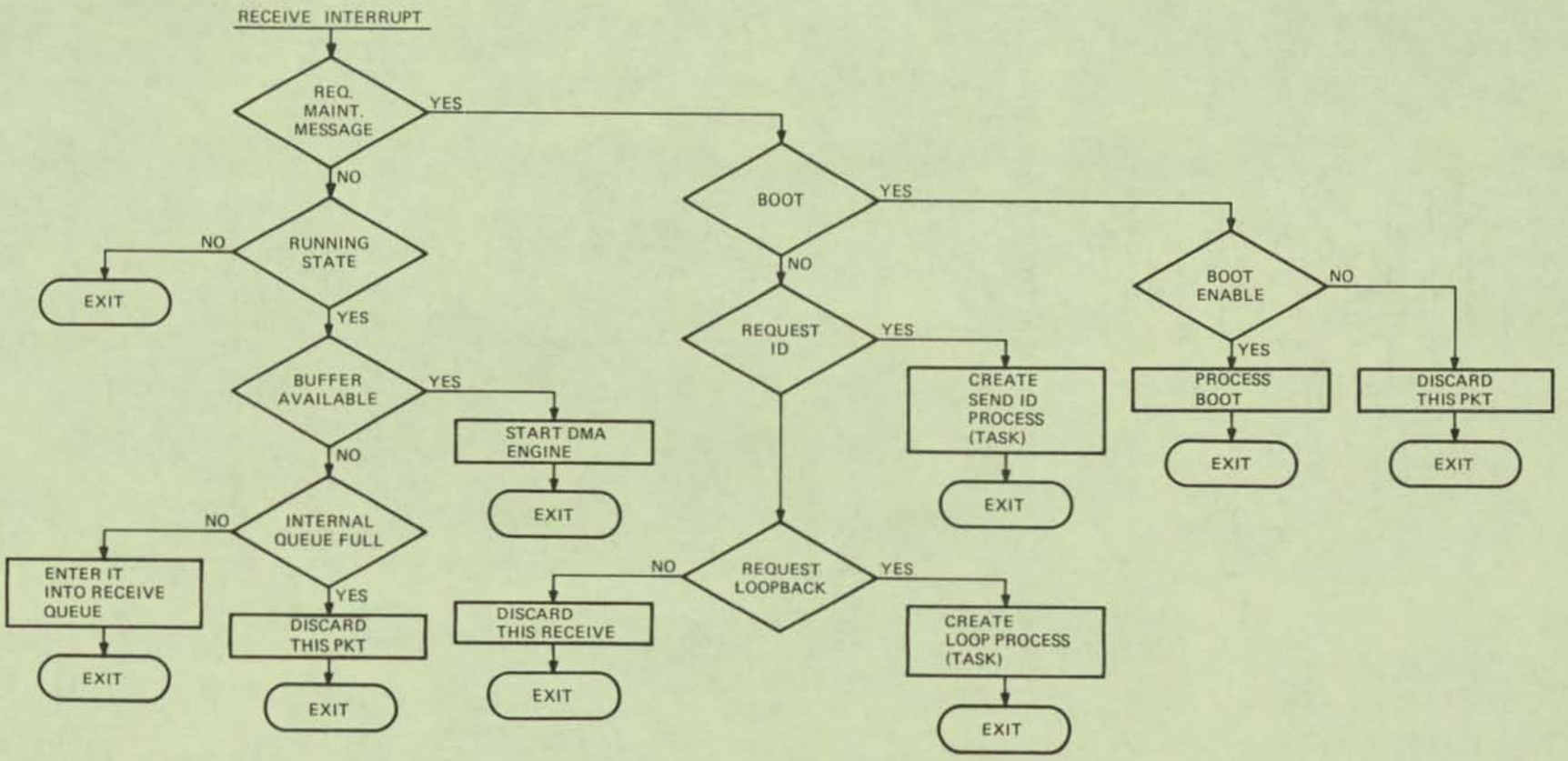


Figure 4-1 Receive Flow Diagram

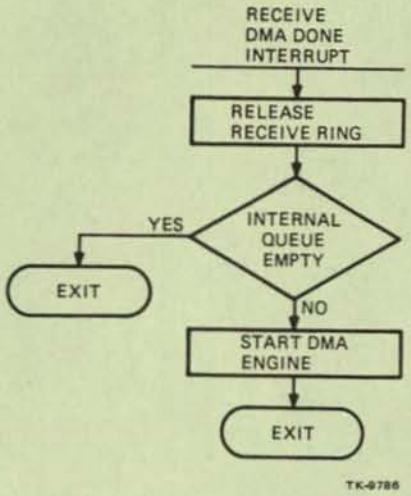


Figure 4-2 Receive DMA Done Flow Diagram

4.3.4 Command Execution Process

The command execution process is used to receive commands from the host processor. The host sends commands to the DEUNA via a structure in host memory called the port control block (PCB). The host tells the DEUNA that it has placed a command in the PCB by writing to PCSRØ. This causes an interrupt to be generated. When the interrupt is received by the T-11, the supervisor will read the command from the PCB and schedule the requested process for execution.

The command process reads the low byte of PCSRØ and uses the code in bits <Ø3:ØØ> to select one of the port command routines.

Figure 4-3 shows the different command processes.

4.3.4.1 Port Commands -- The following port commands are used by the DEUNA.

1. Get PCBB -- The DEUNA reads the address of the PCBB from PCSR2 and PCSR3 and stores it in the WCS.

2. Get CMD -- Requests execution of the ancilliary command process.

3. Self-Test -- Invokes the internal ROM based diagnostic feature of the DEUNA. All datagram activity are aborted and the DEUNA returns to the ready state.

4. START -- The transmit and receive processes are activated and the ring pointers are reset to the base of the rings.

5. BOOT -- The UNA enters the primary load state and requests a program from the load server address.

6. POLL Demand -- The transmit and receive processes are activated if not already active. The transmit and receive rings in host memory are polled.

7. STOP -- The DEUNA completes the current transmit and receive operations and does not fetch any more ring entries until a START command is received.

   This command is implemented by:

   a. Clearing the status flag that indicates the DEUNA is in the running mode, and

   b. Setting the state of both rings to inactive.

   This also causes any datagrams in the link memory buffers to be lost.

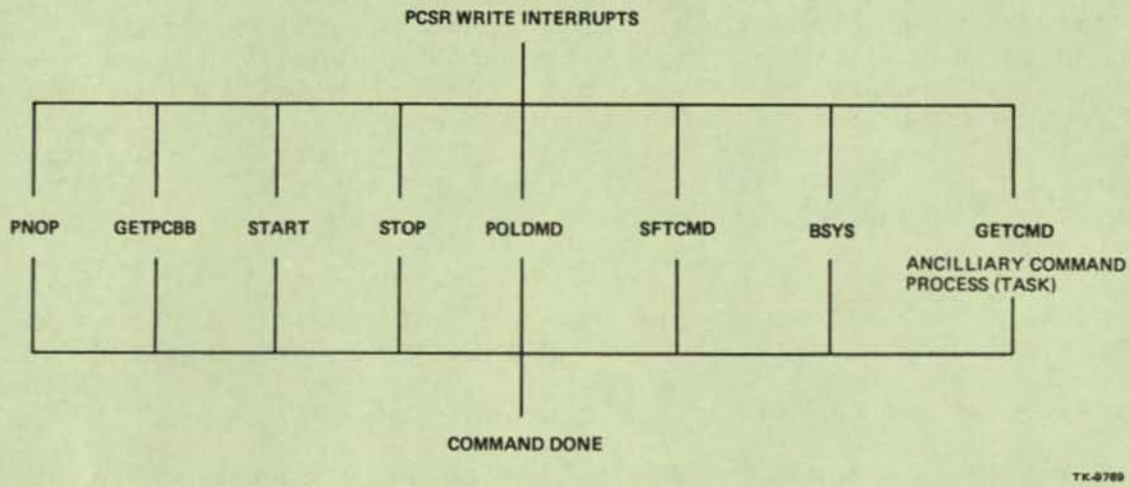


Figure 4-3 Port Command Processes

4.3.4.2 Ancilliary Commands -- The ancilliary commands are subroutines that are called by the get CMD port command. Each of the subroutines executes its specific task and then exits to:

- The command done supervisor routine, or
- Set an error flag and call the appropriate function error routine.

The ancilliary commands executed by the DEUNA are as follows.

1. No Operation (NOP) -- Calls the command done routine.

2. Load and Start at Address (LDSTA) -- Executes a JSR PC instruction directly to the address specified by the PCB.

3. Read Default Physical Address (RDEFPA) -- The physical address contained in ROM on the port module is written to the PCB.

4. Write Physical Address (WRTPA) -- The physical address specified by the PCB is placed in the location reserved for the current physical address (PHYADR). The formatting routine is called to build the data format needed by the address filter of the link module and the data is loaded into the link.

   The link must be halted to execute this command.

5. Read Physical Address (READPA) -- The current physical address is written to the PCB.

6. Write Multicast List (WRTMLT) -- The multicast list is read and stored in a table in WCS. This list along with the broadcast and physical addresses is formatted and written into the address filter in the link.

   The link must be halted to execute this command.

7. Read Multicast List (RDMLT) -- The multicast list is written to the UNIBUS data block specified by the PCB.

8. Read Ring Format (RDRFMT) -- The ring format block of the DEUNA is written to the UNIBUS data block specified by the PCB.

9. Write Ring Format (WTRFMT) -- The ring format is read from the UNIBUS data block and written into WCS of the DEUNA. To maximize performance, the address of the last entry in each ring is calculated. These addresses along with the length of the rings in bytes are saved. The address of both of the rings is written into the ring descriptor for the next entry to be fetched from each of the rings (receive and transmit).

The DEUNA can not be in the running state when this command is executed.

10. Read Counters (RDCNTR) and Read and Clear Counters (RCLCNT) -- The counters that are maintained in WCS are written to host memory. If the command is a read and clear command, the counters are read and then cleared.

11. Dump Internal Memory (DMPMEM) -- A block of data contained in the memory of the DEUNA is specified by the command and transferred to a data buffer in host memory.

12. Load Internal Memory (LDMEM) -- A specified block of data in host memory is copied into the memory on the DEUNA.

13. Read/Write System ID Parameters (RDPARM), (WTPARM) -- The system parameters list is copied from either:

    - A data buffer in host memory to the DEUNA, or
    - The DEUNA to a data buffer in host memory.

14. Read Load Server Address (RDSERV) -- The load server address currently in use by the DEUNA is written into the PCB.

15. Write Load Server Address (WTSERV) -- The load server address in the PCB is written to the DEUNA.

4.3.5 Timer Process

The timer process is executed every second in response to an interrupt generated by the timer on the port module of the DEUNA. The timer is used to:

1. Send an ID message to the ETHERNET every 10 minutes,

2. Keep track of seconds since the counters maintained by the DEUNA were last zeroed. This keeps track of activity in the DEUNA, and

3. Provide timing for various boot operations.

4.3.6 Loop and Maintenance Process

The loop and maintenance process is used to loop data back onto the network, send system ID messages, and perform system boots. The processes are handled as follows:

1. Loop Messages -- Loop service is provided by the microcode to verify that the DEUNA is properly connected to the network and is able to receive and transmit messages.

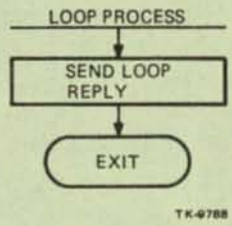


Figure 4-4 Loop Process Flow Diagram

The microcode screens the received messages, in internal memory, to see if there are any loop type messages. (All messages that are not loop type messages are handled as normal datagrams.) If a loop type message is found and is error free it is handled as follows:

a. The microcode modifies some of the address fields.

b. Places the receive buffer into a transmit buffer.

c. Transmits the message.

d. The receive buffer is returned to the receive free buffer queue.

These type of messages are not passed to the host for processing by higher level software.

Figure 4-1 and Figure 4-4 show the function of the microcode for the loop process.

2. System Identification Messages -- When enabled, the microcode will build and transmit a system identification message. This message is transmitted to the network every 8 to 10 minutes to identify the node to the network. This address is also sent if a request station ID message is received. A request station ID message is not processed as a datagram.

   Figure 4-1 and 4-5 shows the function of microcode for the system ID process.

3. Boot Messages -- When enabled, the microcode monitors the incoming receive messages for a boot message. If a boot message is received, the following action takes place:

   a. Datagram service is turned off,

   b. A request program load message is sent to the requesting station, and

   c. The WCS is down-line loaded and program execution is started out of the WCS.

   This procedure may be used to load remote console code or to load the system secondary loader. If the system is to be booted, as determined by the boot message, the microcode will halt the system by asserting ACLO and starting the power fail sequence before it transmits the program request message.

   Figure 4-1 shows the function of the microcode for the boot functions.

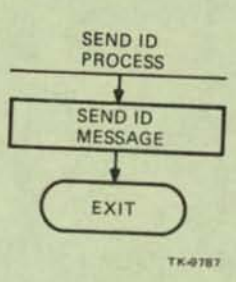


Figure 4-5 Station ID Flow Diagram

4. Power-up Boot -- If the microcode is enabled to do a power-up boot, the microcode will:

   a. Halt the system,

   b. Start power fail sequence,

   c. Transmit program request message, and

   d. Wait for secondary loader.

   If the system boot port command is received, the microcode will handle the request the same way except it does not halt the system.

5. Remote Boot -- For a remote boot from the system ROM (not located on DEUNA), the microcode asserts ACLO.

4.3.7 Transmit Datagram Process

The function of the transmit process is to read a datagram located in host memory and load it into a buffer in link memory for transmission onto the ETHERNET.

The transmit process is activated when the DEUNA receives a poll demand and will be deactivated when the DEUNA comes to a ring entry that is not owned by it.

The transmit process functions as follows.

1. A poll demand or start command generate an interrupt which starts the transmit process.

2. A transmit buffer in link memory is allocated.

3. A ring entry is fetched from the host and is stored in the transmit descriptor of the DEUNA called NEXT.

4. The data described by the ring entry is loaded into the transmit buffer in link memory.

5. The link is given the address of the buffer to be transmitted on the ETHERNET.

6. The link transmit function is started.

7. The ring descriptor in the DEUNA is renamed CURRENT.

8. When the link has finished transmitting the buffer, a transmit done interrupt is generated.

9. The transmit status from the link is stored in the ring entry addressed by the CURRENT ring descriptor in the DEUNA.

10. The ring entry is released and the CURRENT descriptor is marked empty.

11. A return is executed and the NULL process will run. If the transmit process is still the highest process in the request mask the transmit ring will be polled and the process repeated.

Figure 4-6 and Figure 4-7 shows the function of the microcode for the transmit process.

4.3.8 Null Process

The null process scans the request mask to see if any low priority process is scheduled to run. All the low priority processes run sequentially. Each process runs to completion before the request mask is scanned again.

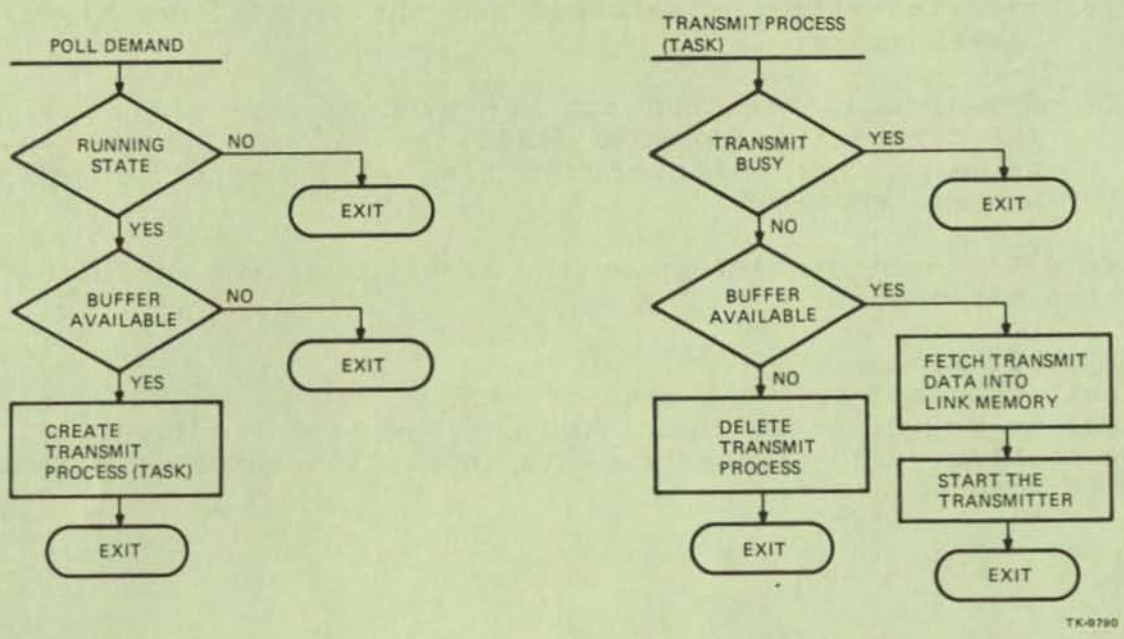


Figure 4-6 Transmit Flow Diagram

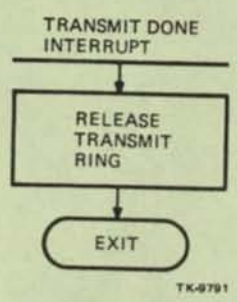


Figure 4-7 Transmit Done Flow Diagram

Digital Equipment Corporation • Bedford, MA 01730

KS10-BASED DECSYSTEM-2020 TECHNICAL MANUAL

Company Confidential

FOR FIELD SERVICE DISTRIBUTION ONLY

digital equipment corporation • marlboro, massachusetts

EK-0KS10-TM-001

KS10-BASED DECSYSTEM-2020 TECHNICAL MANUAL

Company Confidential

FOR FIELD SERVICE DISTRIBUTION ONLY

digital equipment corporation • marlboro, massachusetts

1st Edition, October 1978

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

FIGURES

FIGURES (Cont)

FIGURES (Cont)

TABLES

PREFACE

This technical manual is designed to support the field test phase of the DECSYSTEM-2020 project. It is written for service personnel already competent on KL10-based systems. (Field test phase machines are to be serviced by experienced DECsystem-10 people.) A complete set of hardware documentation, directed to all DIGITAL service personnel, is currently being prepared to support the volume phase of the DECSYSTEM-2020 project. For example, an Installation Guide and a Maintenance Handbook are scheduled for release in 1978. A Technical Description and a Diagnostic User's Guide are scheduled for early 1979.

CHAPTER 1
INTRODUCTION

The DECSYSTEM-2020 is the hardware base for the new low-end member of the DECsystem-10 and DECSYSTEM-20 families of computers. The KS10 processor design combines a highly efficient microprogrammed architecture with low power/high density LS/TTL circuitry. Also, the system supports both the TOPS-10 and TOPS-20 operating systems. This capability, together with the compact/low cost hardware configuration, provides a new kind of mainframe computer; that is, a machine with large computer software power in the mini-midi computer price range.

1.1 OVERVIEW

The configuration for the end-user version of the KS10-based 2020 system is shown in Figure 1-1 and listed in Table 1-1. (Note that systems supported by DIGITAL Field Service require a magtape.)

The heart of the KS10 is an internal backplane bus, called the *KS10 bus,* that provides a control and data path between the processor, memory, console, and peripheral devices (via Unibus adapters). It is a multiplexed 2-cycle bus that allows command and address information to be transmitted by one bus device to another during one bus cycle; data is then transferred to/from the addressed device during a following bus cycle.

The *KS10 processor* consists of four extended hex modules: data path modules DPE and DPM, and control-store modules CRA and CRM. The processor uses low power Schottky TTL and features the AM2901 4-bit data path slice. Other features include:

- A 512-word virtual-address cache memory
- Eight blocks of sixteen fast general purpose registers
- Parity checking in micro-store, on data paths, and on backplane bus
- Fast byte operations on 7-bit ASCII characters
- A 2K word (96-bits/word) writable RAM micro-store with address provision for 4K words
- Basic micro-instruction cycle time of 300 ns.

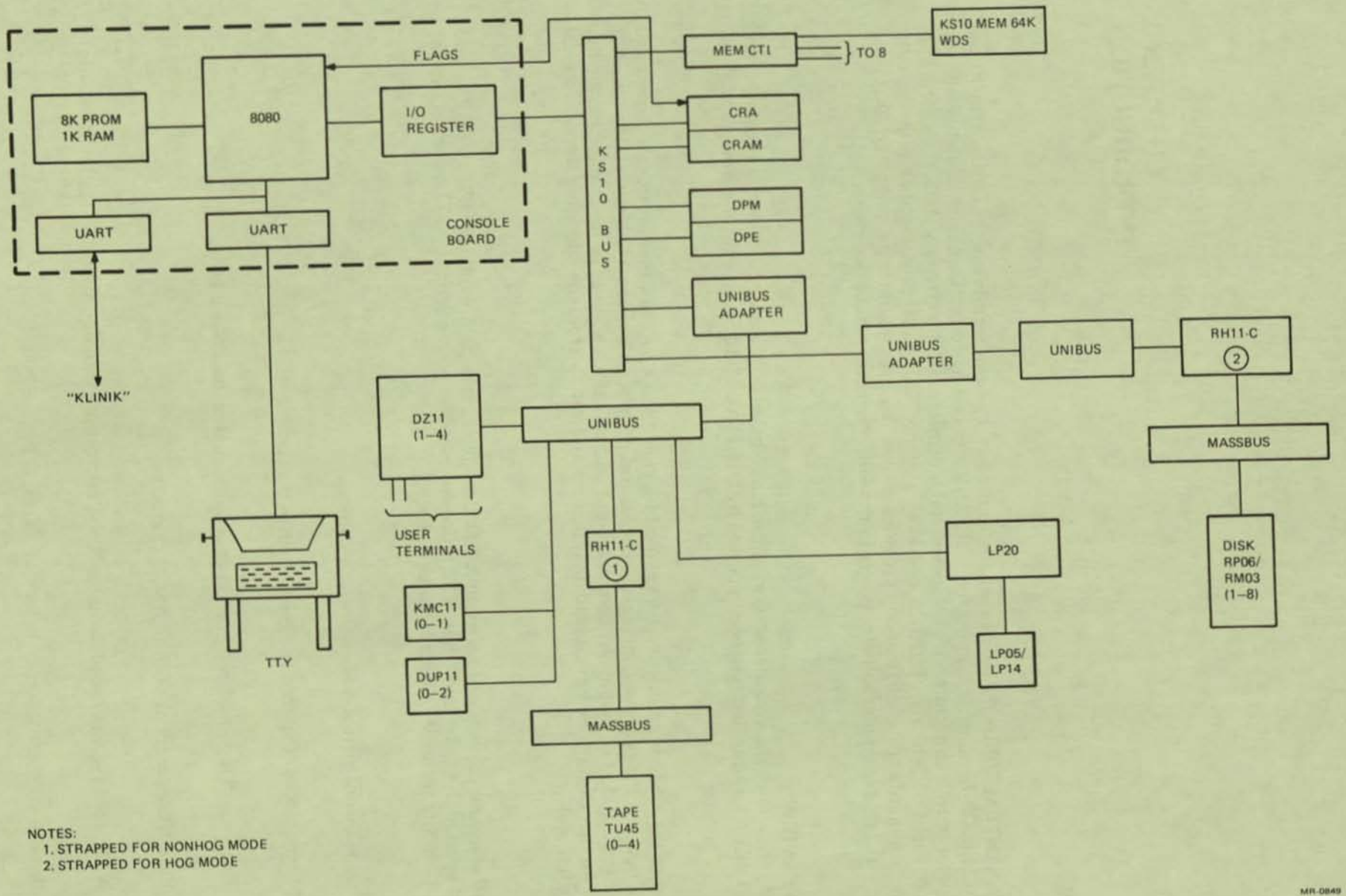


Figure 1-1 DECSYSTEM-2020 System Configuration

Table 1-1 2020 Configurations

Item	Min System	Typical System	Max System
CPU	1	1	1
MEMORY	128K words	256K words	512K words
RM03 or RP06	1	2	8
TU45	0 (see note)	1	4
SYNC LINES	0	1	2
LP05/LP14	0	1	1
TERMINAL LINES	8	16	32

NOTE
OEM-serviced systems only. DIGITAL Software/Hardware Support will not maintain systems that do not have TU45 Magtape.

The *KS10 memory system* consists of a single extended hex control module that connects to the backplane bus and to 2–8 storage (array) modules. Each storage module contains 64K of MOS memory. Memory features include:

- 1.050 μs cycle time
- Single bit error correction
- Double bit error detection
- 128K words minimum capacity and up to 512K words maximum capacity.

The *console* consists of a single extended hex module that uses an 8080 microprocessor to perform console and diagnostic functions. Two UART interfaces are provided: one for console (CTY) operation and one for KLINIK operation. The KLINIK connection operates in parallel with the CTY to allow diagnosis of the system via a remote link.

KS10 peripheral devices are selected Unibus devices that interface to the system through Unibus adapters (UBAs). A *UBA* is a single extended hex module connecting to both the backplane bus and a Unibus. Up to three UBAs may be installed in the KS10 although two UBAs are standard in the end-user 2020 configuration. One UBA (and Unibus) is reserved for disks only. The second UBA (and Unibus) is used for all other devices; that is, for tape, line printer, and synchronous and asynchronous communications lines. Characteristics and features of the devices supported on the UBAs are as follows.

DISKS

RP06

- Average access time of 36.3 ms
- Average seek time of 28 ms
- Formatted capacity of 176 M bits
- Maximum data transfer rate of 166K 36-bit words/second
- Sector size of 128 36-bit words
- Removable (20-surface) disk pack
- 18-bit (NPR) data transfers over Unibus; 36-bit (NPR) data transfers over KS10 (back-plane) bus

RM03

- Average access time of 38.3 ms
- Average seek time of 30 ms
- Formatted capacity of 67 M bits
- Maximum data transfer rate of 250K 36-bit words/second
- Sector size of 128 36-bit words
- Removable (5-surface) disk pack
- 18-bit (NPR) data transfers over Unibus; 36-bit (NPR) data transfers over KS10 (back-plane) bus

TAPE

TM03/TU45

- Tape speed of 75 IPS
- Recording density of 800/1600 BPI, 9-track format on industry-standard ½-inch magnetic tape
- Maximum data transfer rate of 120K characters (bytes)/second
- 18-bit (NPR) data transfers over Unibus; 18-bit (NPR) data transfers over KS10 (back-plane) bus

SYNCHRONOUS COMMUNICATIONS INTERFACE

- DUP11 single-line controller (1 per line)
- Bit rate of 2000–19200 BPS
- DDCMP data protocol
- KMC11 NPR microprocessor (one per system)
- Two lines/system (maximum)
- 8- or 16-bit (NPR) data transfers over Unibus; 8- or 16-bit (NPR) data transfers over KS10 (backplane) bus

ASYNCHRONOUS COMMUNICATIONS INTERFACE

- DZ11 8-line controllers
- RS 232C interface standard with baud rates of 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, and 9600
- Line units available in 8-line groups: 8, 16, 24, or 32 lines per system
- Character lengths of 5, 6, 7, or 8 bits with 1, 1.5, or 2 stop bits and either odd or even parity
- Carrier, ring, data, terminal ready, and break MODEM control
- Full duplex
- Sixty-four character silo receive buffer (alarm at 16 characters)
- 8-bit (register I/O) data transfers over Unibus; 8-bit (register I/O) data transfers over KS10 (backplane) bus

1.2 PHYSICAL DESCRIPTION

The KS10 is compactly configured in a single width corporate high-boy cabinet (H7502H-7). This cabinet, shown in Figure 1-2, houses the KS10PA card cage, BA11K drawer, power system, MASSBUS transition plate, asynchronous communication panel, and operator's switch panel.

1.2.1 KS10PA

The KS10PA assembly is a hybrid style card cage; that is, it contains both extended hex and standard hex modules. It is located in the lower front portion of the KS10 cabinet as shown in Figure 1-3. This assembly contains the KS10 CPU, the MOS memory (128K words minimum, 512K words maximum), two Unibus adapters (UBAs), and the RH11C Unibus disk controller. Module utilization is shown in Figure 1-4.

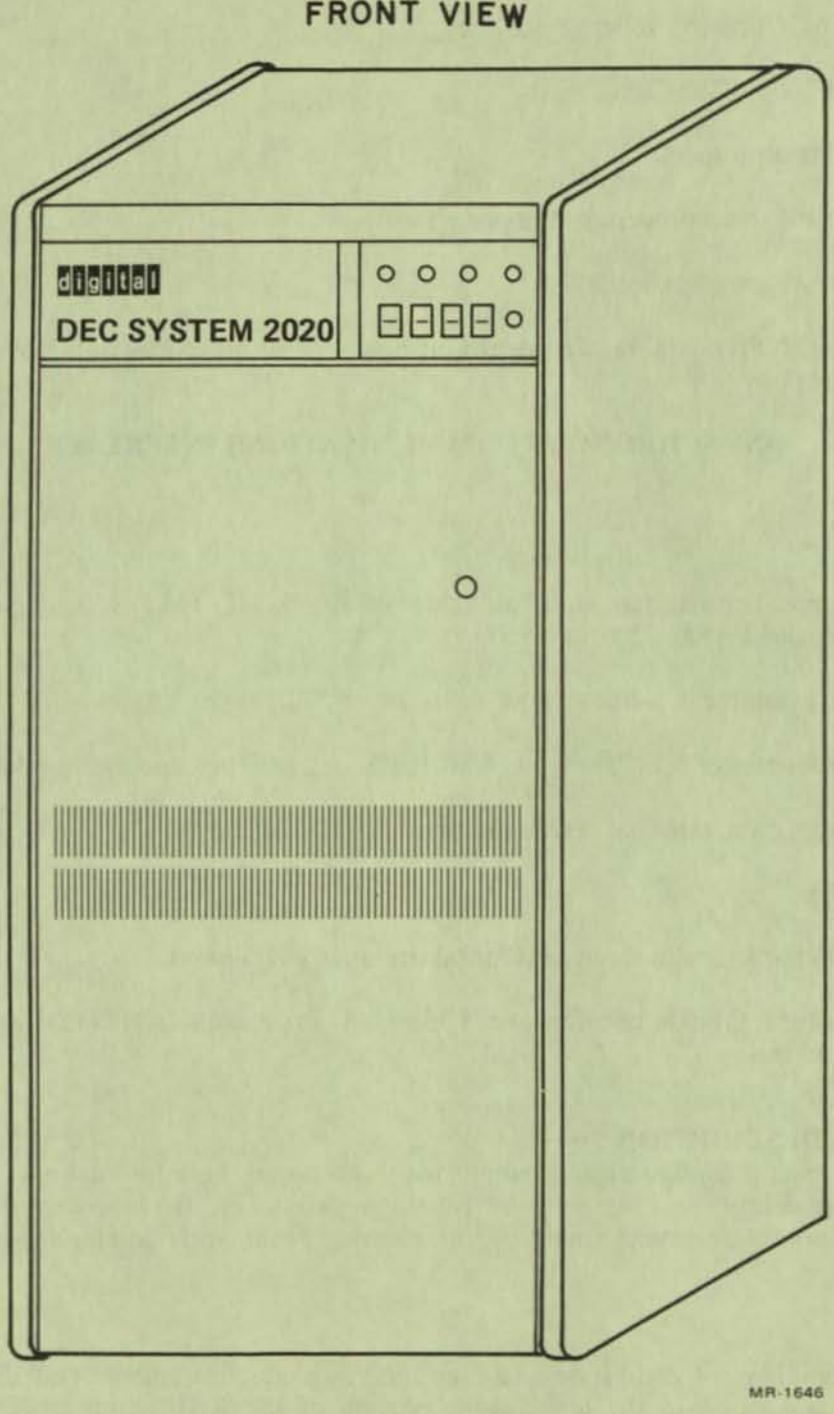


Figure 1-2 KS10 Cabinet

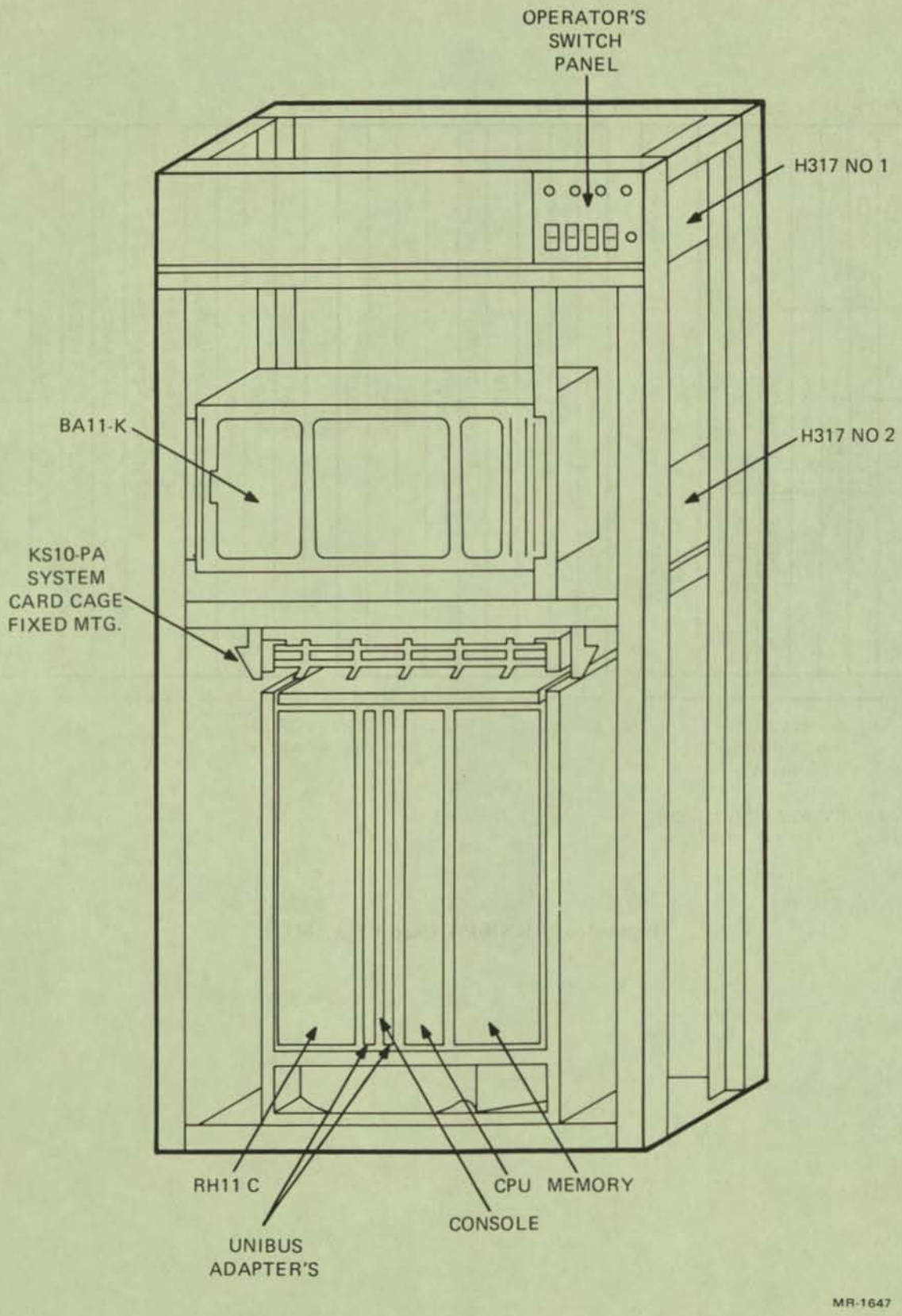
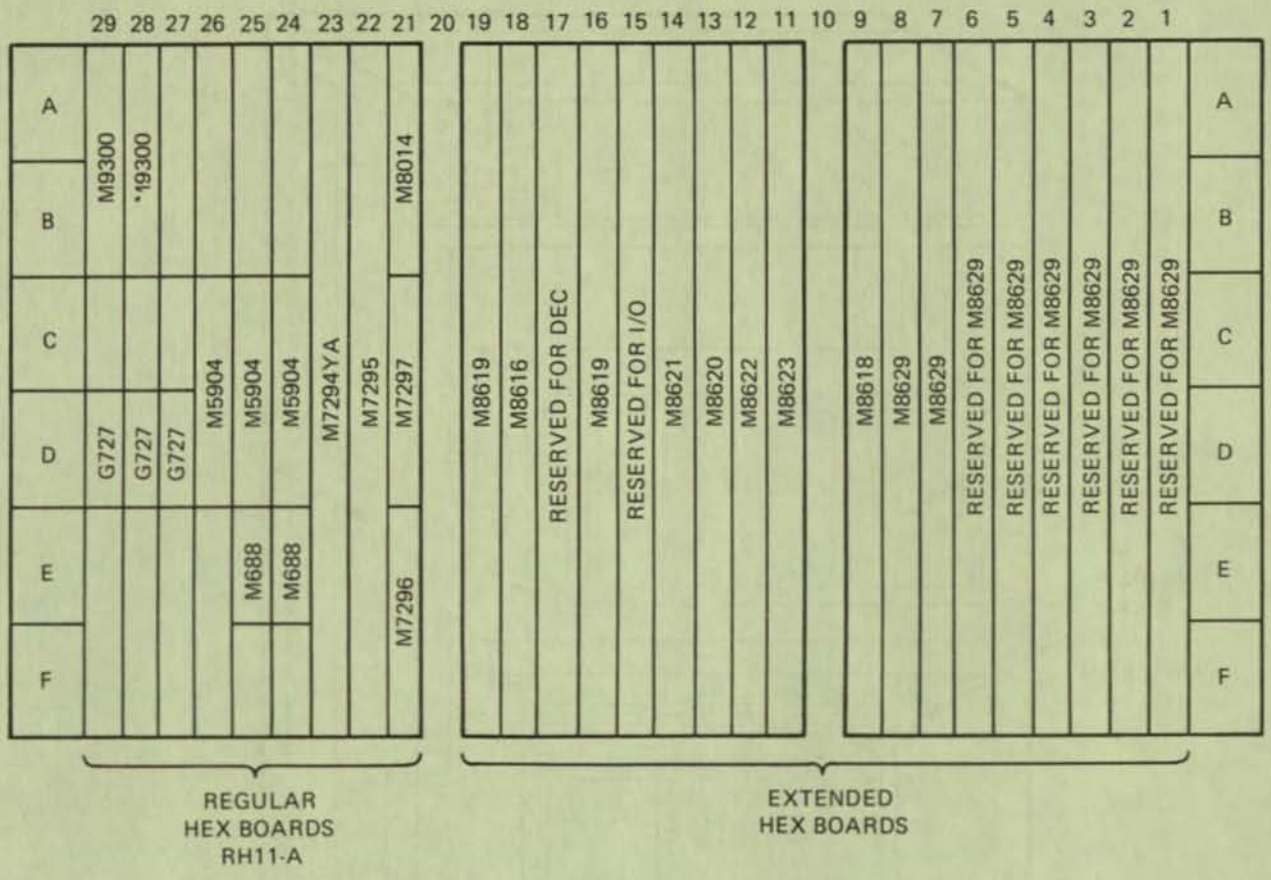


Figure 1-3 KS10 Cabinet (Front View - Skins Removed)



NOTE: VIEW IS FROM MODULE SIDE

MR-1648

Figure 1-4 KS10PA Card Cage, MUL

1.2.2 BA11-K

The BA11-K drawer (Figure 1-3) contains the KS10 system's I/O peripheral controllers. It has dedicated locations for the following.

1. DZ11 asynchronous communications controllers: 1 minimum (8 lines), 4 maximum (32 lines)
2. DUP11/KMC11 synchronous communications controller: 0 minimum, 2 DUP11s maximum (2 lines)
3. LP20 line printer controller: 0 minimum, 1 maximum
4. RH11C magnetic tape system controller: 0 minimum, 1 maximum. (This option is bundled into the TAU45 tape system.)

BA11-K module utilization is shown in Figure 1-5.

1.2.3 Power System

The major components in the KS10 power system are the 861 power control for ac power distribution, the LH switcher power supply for powering the KS10PA, and the H765 switcher power supply for powering the BA11-K. Component designations for 60 Hz and 50 Hz machines are as follows.

KS10AA (115 V, 60 Hz)	**KS10AB (230 V, 50 Hz)**
861C	861B
LH Power Supply (H7130C)	LH Power Supply (H7130D)
H765A (powers BA11K)	H765B (powers BA11K)

NOTE TO DIGITAL
IN-HOUSE FIELD SERVICE PERSONNEL

The first of the in-house KS10 systems will contain the H7130A (60 Hz) and H7130B (50 Hz) power supplies. Although input and output power specifications for these A and B (blue) models are the same as for the C and D (silver) models that are installed in all other machines, there are differences in power harness wiring. Thus, in the event of failure, do not replace one type of supply with one of a different color.

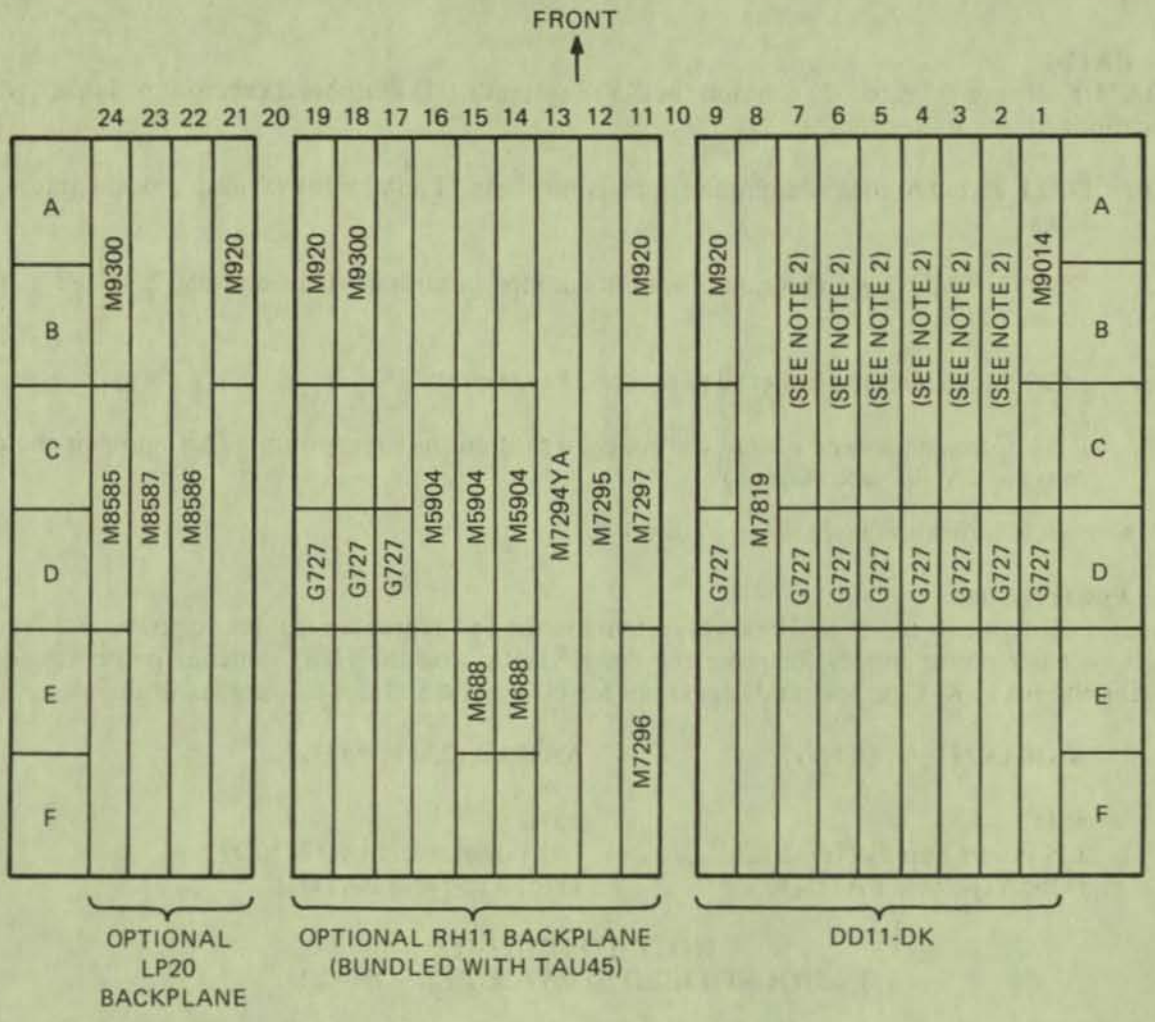
1.2.4 MASSBUS Transition Plate

The MASSBUS transition plate is located at the top of the KS10 cabinet as shown in Figure 1-6. It is a connection plate that holds three MASSBUS connectors plus two 25-pin communications cable connectors. The MASSBUS connectors are allocated from right to left as follows.

1. Disk MASSBUS channel
2. Tape MASSBUS channel
3. Line printer channel

The two communication cable connectors are allocated as follows.

1. CTY (BC03L to BC03M)
2. KLINIK remote maintenance port (BC03L to BC05D)



NOTES:

1. VIEW IS FROM MODULE SIDE.
2. OPTION VARIATIONS ARE LISTED BELOW.

OPTION VARIATIONS

SLOTS	ASYNC LINES 8–15	ASYNC LINES 16–23	ASYNC LINES 24–32	SYNC FIRST	SYNC SECOND
2				M8207 KMC11	
3					M7867 DUP11
4				M7867 DUP11	
5			M7819 DZ11		
6	(M7819 DZ11)	M7819 DZ11			
7	M7819 DZ11				

3. M7819 FOR ASYNC LINES 8–15 IS INSTALLED IN SLOT 6 WHEN CONFIGURATION EQUALS 0–23 LINES.

MR-1649

Figure 1-5 BA11-K Drawer, MUL

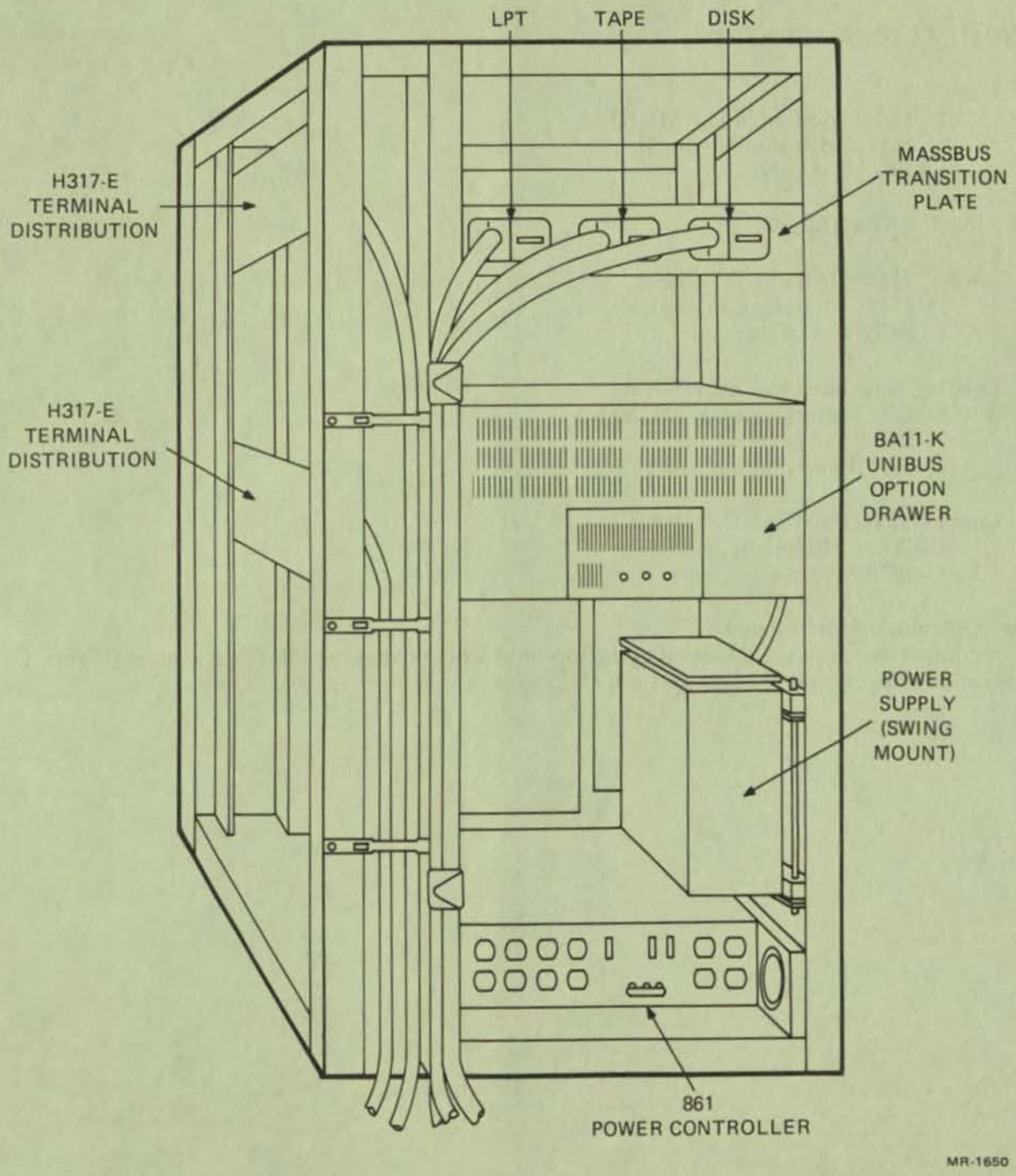


Figure 1-6 KS10 Cabinet (Rear View – Skins Removed)

1.2.5 Asynchronous Communications Panel (H317E)

The KS10 is configured with a minimum of 1-H317E (up to 16 lines) and a maximum of 2-H317Es (32 lines). It is configured with EIA communication only.

MINIMUM CONFIGURATION

Lines 0–7:
- 1 - DZ11 Module (8-line MUX)
- 1 - H317E distribution panel
- 1 - BC05W-8 cable

OPTIONAL EXPANSION

Lines 8–15 (defined as a DZ11BA):
- 1 - DZ11 Module (8-line MUX)
- 1 - BC05W-8 cable

Line 16–23 (defined as a DZ11AA):
- 1 - DZ11 Module (8-line MUX)
- 1 - BC05W-8 cable
- 1 - H317E distribution panel

Line 24–32 (defined as a DZ11BA):
- 1 - DZ11 Module (8-line MUX)
- 1 - BC05W-8 cable

1.2.6 Operator's Switch Panel

The operator's switch panel is located at the top-most front position in the KS10 cabinet (Figure 1-2). Switch and indicator functions are given in Chapter 4.

CHAPTER 2
SITE PREPARATION AND PLANNING

2.1 SITE PLANNING

Refer to Chapter 1 (Paragraph 1.1–1.4) of the DECSYSTEM-20 Site Preparation Guide for the following information:

- Schedule of site preparation prior to system delivery
- Summary of site preparation functions and responsibilities
- Site consideration and selection
- Building requirements.

2.2 ENVIRONMENTAL REQUIREMENTS

The recommended environmental specifications for DECSYSTEM-20 systems (including KS10 systems) are listed in Table 2-1. The environmental specifications for individual KS10 system components are given on data sheets at the end of this chapter. Heat dissipation and air flow rate of internal fans are also given. To estimate cooling and other environmental requirements, refer to Paragraph 1.6 of the DECSYSTEM-20 Site Preparation Guide.

Table 2-1 Recommended KS10 System Environmental Specifications

PARAMETER	SPECIFICATION
Temperature	18° C to 24° C (65° F to 75° F)
Humidity	40% to 60%
Temperature Rate of Change	2° C/hr (3.6° F/hr)
Humidity Rate of Change	2%/hr
Voltage Tolerance	120/208 V ± 10% for single phase/ three phase (60 Hz)
	240/380 V ± 10% for single phase/ three phase (50 Hz)
Frequency Tolerance	60 Hz ± 1 Hz 50 Hz ± 1 Hz

NOTE

Compliance to the environmental specifications above may be required if the system is under a DIGITAL Maintenance Agreement.

2.3 SYSTEM CONFIGURATION

Figure 2-1 shows a typical KS10 system configuration. Reference is made on the figure to Tables 2-2 and 2-3, which provide MASSBUS and device cable data, and to Figure 2-2 and 2-3, which show interconnections of the asynchronous and synchronous communications lines.

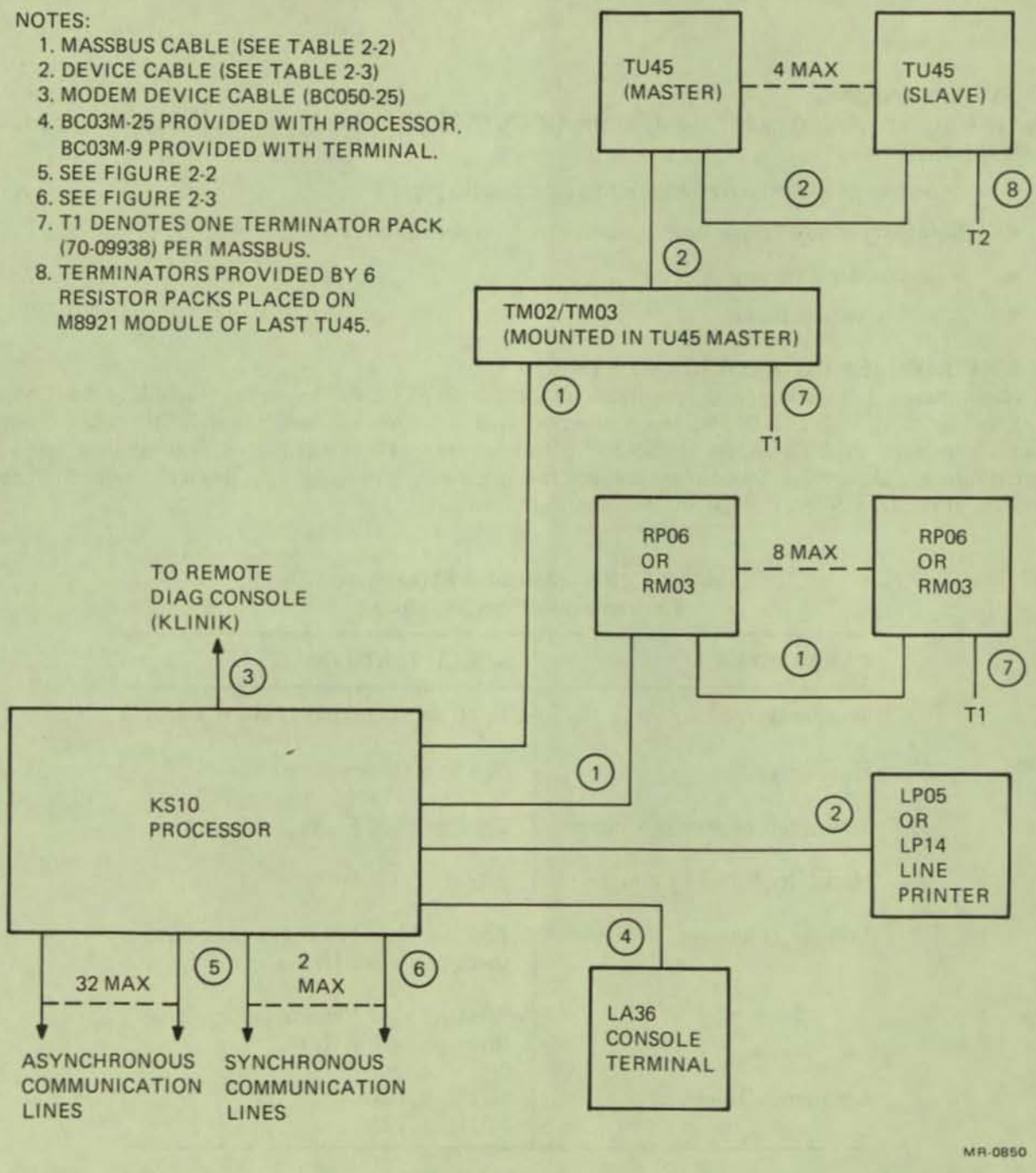
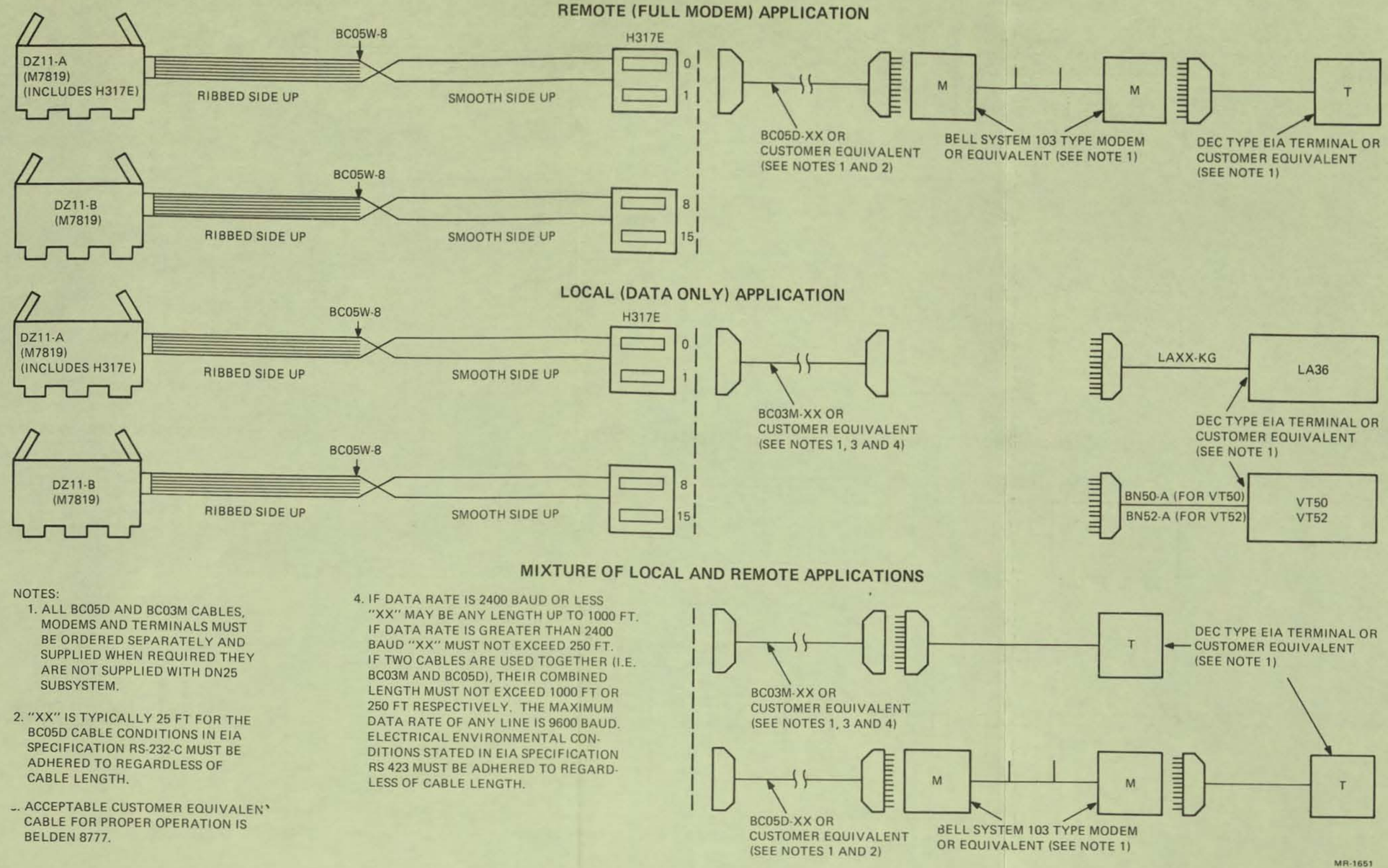


Figure 2-1 Typical KS10 System Configuration



NOTES:

1. ALL BC05D AND BC03M CABLES, MODEMS AND TERMINALS MUST BE ORDERED SEPARATELY AND SUPPLIED WHEN REQUIRED THEY ARE NOT SUPPLIED WITH DN25 SUBSYSTEM.

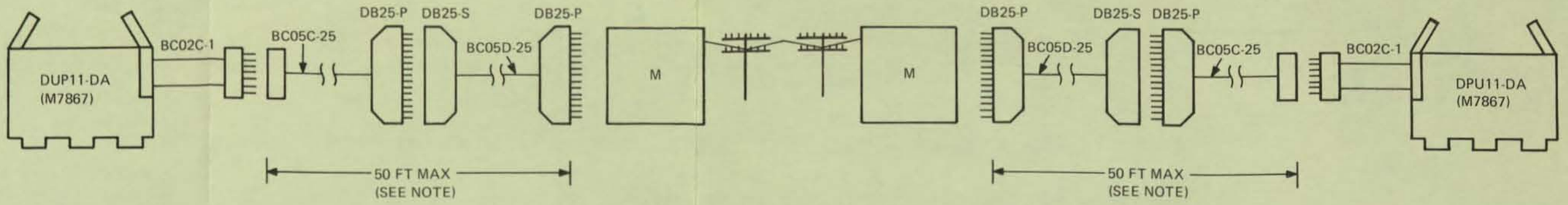
2. "XX" IS TYPICALLY 25 FT FOR THE BC05D CABLE CONDITIONS IN EIA SPECIFICATION RS-232-C MUST BE ADHERED TO REGARDLESS OF CABLE LENGTH.

3. ACCEPTABLE CUSTOMER EQUIVALENT CABLE FOR PROPER OPERATION IS BELDEN 8777.

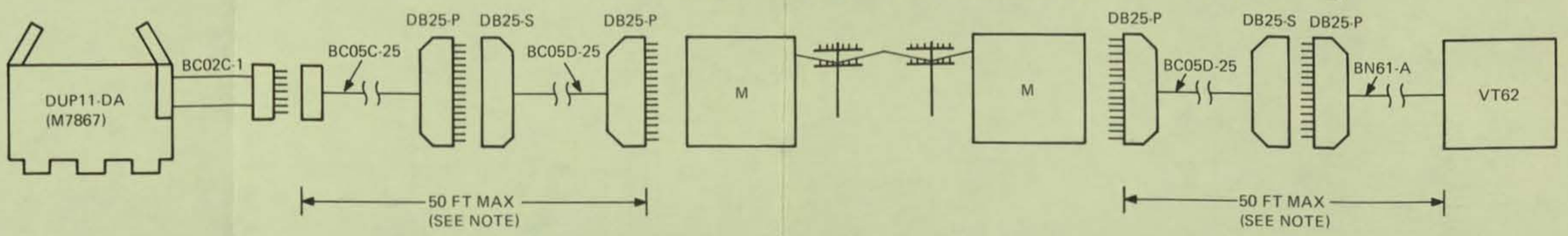
4. IF DATA RATE IS 2400 BAUD OR LESS "XX" MAY BE ANY LENGTH UP TO 1000 FT. IF DATA RATE IS GREATER THAN 2400 BAUD "XX" MUST NOT EXCEED 250 FT. IF TWO CABLES ARE USED TOGETHER (I.E. BC03M AND BC05D), THEIR COMBINED LENGTH MUST NOT EXCEED 1000 FT OR 250 FT RESPECTIVELY. THE MAXIMUM DATA RATE OF ANY LINE IS 9600 BAUD. ELECTRICAL ENVIRONMENTAL CONDITIONS STATED IN EIA SPECIFICATION RS 423 MUST BE ADHERED TO REGARDLESS OF CABLE LENGTH.

Figure 2-2 KS10 Asynchronous Communications Lines

REMOTE SYNCHRONOUS MODEM CONNECTIONS (SYSTEM TO SYSTEM)

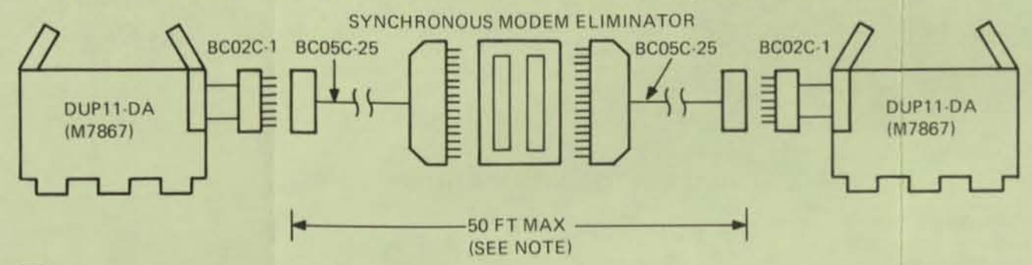


REMOTE SYNCHRONOUS MODEM CONNECTIONS (SYSTEM TO TERMINAL)

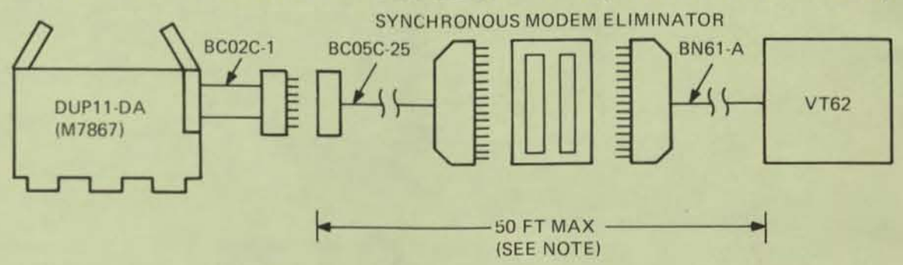


NOTE:
IF BC05C-25 WILL REACH TO DATA COMMUNICATION EQUIPMENT, THE BC05D-25 MAY BE EXCLUDED, IF BOTH CABLES ARE USED, THE 50 FT MAXIMUM DISTANCE MUST BE ADHERED TO.

LOCAL SYNCHRONOUS NULL MODEM CONNECTION (SYSTEM TO SYSTEM)

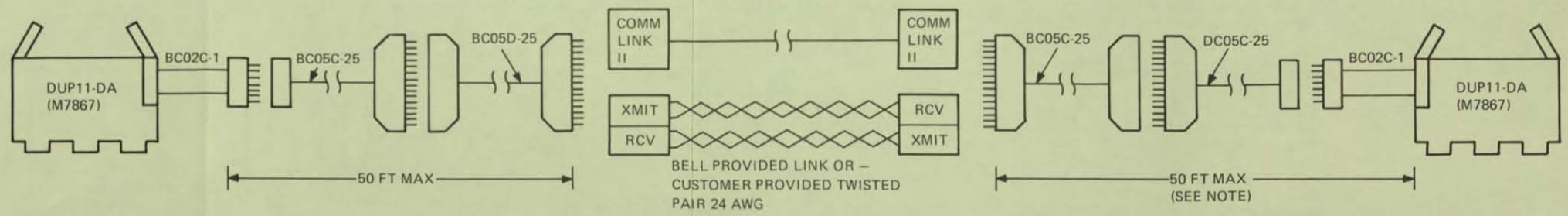


LOCAL SYNCHRONOUS NULL MODEM CONNECTION (SYSTEM TO TERMINAL)



SYNCHRONOUS LIMITED DISTANCE ADAPTERS (SHORT HAUL)

SEE DISTANCE TABLE ON SHEET 2



MR-1652

Figure 2-3 KS10 Synchronous Communications Lines

Table 2-2 Massbus Cabling

FROM	TO	CABLE	AVAILABLE METERS	LENGTH FEET
CPU	RP06	BC06S (AMP ZIF to AMP ZIF)	4.5	15
CPU	RM03	BC06S (AMP ZIF to AMP ZIF)	7.5	25
RP06	RP06	BC06S (AMP ZIF to AMP ZIF)	0.6/0.75	2/2.5
RM03	RM03	BC06S (AMP ZIF to AMP ZIF)	4.5	15
RP06	RM03	BC06S (AMP ZIF to AMP ZIF)	4.5	15
CPU	TM02/TM03	BC06S (AMP ZIF to AMP ZIF)	4.5	15

Table 2-3 Device Cabling

FROM	TO	CABLE	AVAILABLE METERS	LENGTH FEET
CPU	LP05/LP14	7011426 (AMP ZIF to Winchester)	7.5 30	25* 100
TM02/ TM03	TU45	BC06R (BERG to BERG-cabled internally)	3	10
TU45	TU45	BC06R (BERG to BERG)	3	10

NOTE

An asterisk (*) denotes the standard length that will be provided if no cable information is provided 60 days prior to scheduled shipment.

2.4 PRIMARY POWER (AC)

Primary power specifications for KS10 system components are provided on data sheets at the end of this chapter. Refer to Chapter 1 (Paragraphs 1.7–1.8) of the DECSYSTEM-20 Site Preparation Guide for the following information:

- Definition of data sheet parameters (surge current, leakage current, etc.)
- Description of power regulation systems
- Phase balancing, grounding, and service outlet requirements
- Description of receptacles and plugs specified (on data sheets) for KS10 system components.

2.5 OPTION DATA SHEETS

Option data sheets for the various KS10 system components are contained in this section and are arranged in alphanumeric sequence by device designations as follows.

1. KS10-AA/AB Processor
2. LA36
3. LP05
4. LP14
5. RM03
6. RP06
7. TU45A (Master)
8. TU45A (Slave)

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	267 kg 590 lb	152 cm 60 in	69 cm 27 in	76 cm 30 in	H9502H-7	N/A

POWER (AC)

AC Voltage			Frequency		Steady State	Surge	Surge
Low	Nom	High	Tolerance	Phase(s)	Current (RMS)	Current *	Duration
104 208	115 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	9.90 A 4.95 A	25.0 A 12.5 A	6 cycles 6 cycles

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
16 ms	920 kg-cal/hr 3652 Btu/hr	1070	1.14	4.5 m 15 ft	NEMA L5-30P NEMA L6-20P	4.93 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	Inlet
15° to 32° C 59° to 90° F	-40° to 66° C -40° to 151° F	20-80%	0-95%	7° C/hr 12° F/hr	2%/hr	1100 ft^3/min

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
N/A	N/A	N/A	N/A	N/A	N/A

MAXIMUM CABLE LENGTH AND TYPE(S)

Memory	I/O Bus	Massbus	Device	Other
N/A	N/A	See Table 2-2	See Table 2-3	N/A (internal)

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
VE	46.4 kg 102 lb	85 cm 33.5 in	70 cm 27.5 in	61 cm 24 in	VE	86.4 cm X 128.3 cm 34" X 50-1/2"

POWER (AC)

AC Voltage			Frequency Tolerance	Phase(s)	Steady State Current (RMS)	Surge Current	Surge Duration
Low	Nom	High					
104 208	115 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	2.0 A 1.0 A	60 A 20 A	

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
	309 kg·cal/hr 1230 Btu/hr	300-print 160-idle	0.35	2.4 m 8 ft	NEMA L5-30P NEMA L6-20P	0.107 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	Inlet
15° to 32° C 59° to 90° F	-40° to 66° C -40° to 151°F	20 - 80%	0 - 95%	7° C/hr 12° F/hr	2%/hr	100 ft^3/min

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
15° to 32° C 59° to 90° F	15° to 32° C 59° to 90° F	20 - 80%	20 - 80%		

MAXIMUM CABLE LENGTH AND TYPE(S)

Memory	I/O Bus	Massbus	Device*	Other
N/A	N/A	N/A	3 m 9 ft	N/A

*3 m (9 ft) for EIA Interface
5 m (15 ft) for 20 mA Loop

LA36

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
VE	155 kg 340 lb	113 cm 44.5 in	84 cm 33 in	66 cm 26 in	VE	84 cm X 103 cm 33 in X 40-1/2 in

POWER (AC)

AC Voltage			Frequency Tolerance	Phase(s)	Steady State Current (RMS)	Surge Current	Surge Duration
Low	Nom	High					
104 208	115 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	4.5 A 2.3 A	98 A 49 A	2 s 2 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
5 ms	450 kg-cal/hr 1800 Btu/hr	459	0.525	4.0 m 13 ft	NEMA 5-15P NEMA 6-15P	0.55 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
10° to 38° C 50° to 100° F	-18° to 66° C 0° to 150° F	10 - 90%	5 - 95%	7° C/hr 12° F/hr	2%/hr	300 ft^3/hr

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
10° to 38° C 50° to 100° F	-18° to 66° C 0° to 150° F	10 - 90%	5 - 95%	7° C/hr 12° F/hr	2%/hr

MAXIMUM CABLE LENGTH AND TYPE(S)

Memory	I/O Bus	Massbus	Device	Other
N/A	N/A	N/A	30 m 100 ft	N/A

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
VE	198 kg 435 lb	114 cm 45 in	84 cm 33 in	70 cm 27 in	VE	86.4 cm X 128.3 cm 34 in X 50-1/2 in

POWER (AC)

AC Voltage			Frequency		Steady State	Surge	Surge
Low	Nom	High	Tolerance	Phase(s)	Current (RMS)	Current	Duration
104 208	115 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	7 A 3.5 A	140 A 70 A	2 s 2 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
5 ms	710 kg·cal/hr 2815 Btu/hr	780	0.825	3.7 m 12 ft	NEMA 5-15P NEMA 6-15P	0.394 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	Inlet
10° to 38° C 50° to 100° F	-18° to 66° C 0° to 150° F	10-90%	5-95%	7° C/hr 12° F/hr	2%/hr	300 ft^3/min

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
10° to 38° C 50° to 100° F	-18° to 66° C 0° to 150° F	10-90%	5-95%	7° C/hr 12° F/hr	2%/hr

MAXIMUM CABLE LENGTH AND TYPE(S)

Memory	I/O Bus	Massbus	Device	Other
N/A	N/A	N/A	30 m 100 ft	N/A

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	195 kg 430 lb	99 cm 39 in	54.6 cm 21.5 in	84 cm 33 in	H9691 (modified)	

POWER (AC)

AC Voltage			Frequency		Steady State	Surge	Surge
Low	Nom	High	Tolerance	Phase(s)	Current (RMS)	Current	Duration
104 208	115 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	11 A 5.5 A	30 A 15 A	14 s 14 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
	560 kg·cal/hr 2220 Btu/hr	650	0.73	3.7 m 12 ft	NEMA 5-15P NEMA 6-15P	1.218 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	Inlet
15° to 32° C 59° to 90° F	-40° to 66° C -40° to 151° F	20 - 80%	0 - 95%	7° C/hr 12° F/hr	2%/hr	

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.

MAXIMUM CABLE LENGTH AND TYPE(S)

Memory	I/O Bus	Massbus *	Device	Other
N/A	N/A	48 m 160 ft	N/A	N/A

*Total system (maximum)

RM03

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	275 kg 600 lb	119 cm 47 in	84 cm 33 in	81 cm 32 in	VE	12-10568-02

POWER (AC)

AC Voltage			Frequency		Steady State	Surge	Surge
Low	Nom	High	Tolerance	Phase(s)	Current (RMS)	Current	Duration
See Note 1			60 Hz ± 1 50 Hz ± 1	3-wye 3-wye	6 A/Phase 3 A/Phase	30 A/Phase 15 A/Phase	15 s 15 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
25 ms	1800 kg·cal/hr 7200 Btu/hr	2100	2.16	4.5 m 15 ft	NEMA # L21-20P	4.36 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	Inlet
16° to 32° C 60° to 90° F	10° to 44° C 50° to 110° F	20-80%	10-90%	7° C/hr 12° F/hr	2%/hr	550 ft^3/min

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
15° to 50° C 60° to 120° F	40° to 65° C -40° to 150° F	8-80%	8-80%	7° C/hr 12° F/hr	2%/hr

MAXIMUM CABLE LENGTH AND TYPE(S)

Memory	I/O Bus	Massbus *	Device	Other
N/A	N/A	48 m 160 ft	N/A	N/A

*Total system (maximum)

RP06-A, B

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	290 kg 640 lb	152 cm 60 in	69 cm 27 in	76 cm 30 in	H9502	N/A

POWER (AC)

AC Voltage			Frequency		Steady State	Surge	Surge
Low	Nom	High	Tolerance	Phase(s)	Current (RMS)	Current	Duration
104 208	115 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	8.5 A 4.3 A	20 A 10 A	4 cycles 4 cycles

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
5 ms	840 kg cal/hr 3300 Btu/hr	880	0.98	4.5 m 15 ft	NEMA L5-30P NEMA L6-20P	3.16 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	Inlet
16° to 32° C 60° to 90° F	-40° to 60° C -40° to 140° F	20-80%	5-95%	7° C/hr 12° F/hr	N/A	500 ft^3/min

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
16° to 32° C 60° to 90° F	-40° to 60° C -40° to 140° F	20-80%	5-95%	N/A	N/A

MAXIMUM CABLE LENGTH AND TYPE(S)

Memory	I/O Bus	Massbus	Device	Other
N/A	N/A	30 m 100 ft	1.8 m 6 ft	N/A

TU45A-E (Master)

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	272 kg 600 lb	152 cm 60 in	69 cm 27 in	76 cm 30 in	H9502	N/A

POWER (AC)

AC Voltage			Frequency		Steady State	Surge	Surge
Low	Nom	High	Tolerance	Phase(s)	Current (RMS)	Current	Duration
104 208	115 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	6.8 A 3.4 A	15 A 7.5 A	4 cycles 4 cycles

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
N/A	680 kg·cal/hr 2690 Btu/hr	704	0.78	4.5 m 15 ft	NEMA L5-30P NEMA L6-20P	3.16 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	Inlet
16° to 32° C 60° to 90° F	-40° to 60° C -40° to 140° F	20–80%	5–95%	7° C/hr 12° F/hr	N/A	500 ft^3/min

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
16° to 32° C 60° to 90° F	-40° to 60° C -40° to 140° F	20–80%	5–95%	N/A	N/A

MAXIMUM CABLE LENGTH AND TYPE(S)

Memory	I/O Bus	Massbus	Device	Other
N/A	N/A	30 m 100 ft	3 m 10 ft	N/A

TU45A-E (Slave)

CHAPTER 3
INSTALLATION

Refer to KS10-Based DECSYSTEM-2020 Installation Manual (Document number EK-0KS10-IN).

CHAPTER 4
OPERATION/PROGRAMMING

4.1 CONTROLS AND INDICATORS

The KS10 switch and indicator panel is shown in Figure 4-1. There are five switches, three of which provide for powering-up, resetting, and bootstrapping the system. The fourth switch serves as an interlock to prevent an inadvertent reset or bootstrap by the operator once the system is in operation. The last switch controls the remote diagnosis link to the system. Switch functions are listed in Table 4-1.

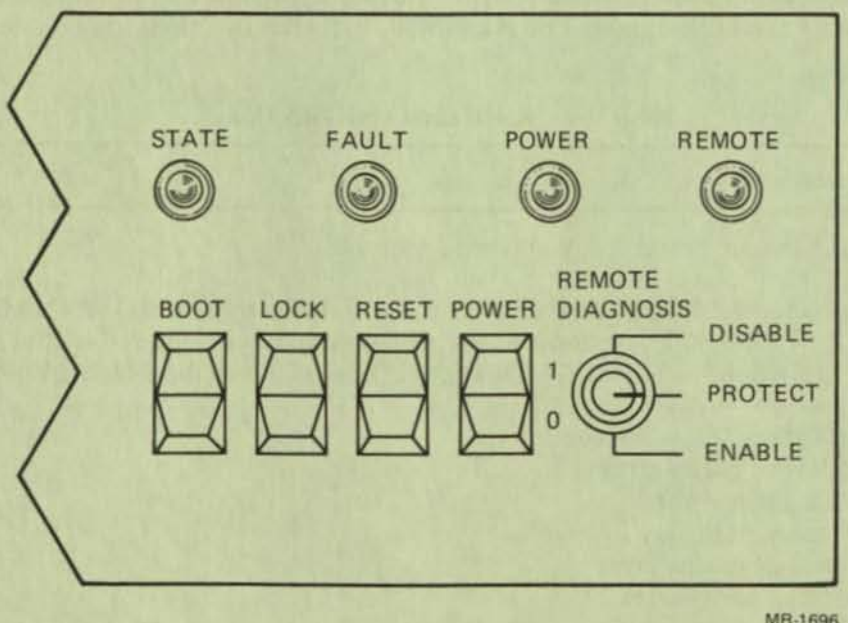


Figure 4-1 KS10 Switch and Indicator Panel

Table 4-1 KS10 Switch Functions

Switch	Function
POWER	Turns ac power on/off. (Causes 861 power control to apply/remove line power to the CPU and BA11K power supplies.)
RESET	Resets all KS10 system components (including the 8080 console hardware).
BOOT	Bootstraps the system. Performs same function as BT console command.
LOCK	Electrically interlocks the RESET and BOOT switches so that they have no effect. Also prevents the operator from switching the CTY from user mode to CTY mode (disables "control-\" command).
REMOTE DIAGNOSIS	Three-position key-operated switch that controls access by the remote diagnosis (KLINIK) line. DISABLE position – Prevents access to the system. PROTECT position – Allows access to the system with password. ENABLE position – Allows free access to the system without password protection.

The panel also has four indicators. One indicates power-on. The other three, which are under control of the 8080 console program, indicate the system's run state, when a system fault has been detected, and when the system's remote diagnosis line is enabled. Indicator functions are detailed in Table 4-2.

Table 4-2 KS10 Indicator Functions

Indicator	Function
POWER	Lights when dc power (-5 V and +12 V) is on.
REMOTE	Lights when KLINIK line is enabled; that is, when the REMOTE DIAGNOSIS switch is in the PROTECT position and the password has been entered by the operator (at the CTY), or when the REMOTE DIAGNOSIS switch is in the ENABLE position.
FAULT	Lights for the following conditions: 1. KS10 bus parity error 2. UBA parity error 3. Memory parity error 4. Data path parity error 5. Console parity error 6. CRA parity error 7. CRM parity error 8. Memory refresh error 9. Boot command fails to start machine.
STATE	Lights when KS10 microcode is loaded and running. Indicator blinks (1 second on, 1 second off) when system monitor has been loaded and is maintaining "keep-alive" dialogue with console.

4.2 KS10 DIFFERENCES (KS10 vs KL10)

For the purpose of this document, some aspects of KS10 operation and programming are best described in relation to the KL10. There are only a few differences in the instruction set and the KS10 uses the same operating system as the KL10 with minor modifications. The following differences between the KS10 and KL10 do exist, however.

4.2.1 Public Mode

Because TOPS-20 does not support public mode, it has not been implemented in the KS10. All instructions behave as if the machine is in concealed mode.

4.2.2 Addressing

Only section 0 addressing is implemented in the KS10; that is, like the KL10-B (KL10-PA processor), virtual memory space is 256K words. Extended addressing, 32 sections of 256K words as implemented by the KL10-E (KL10-PV processor), is not currently supported by the KS10. It does support the model B instructions XJRSTF, XJEN, XPCW, and SFM.

4.2.3 Interrupt Handling

KS10 priority interrupt operation is the same as the KL10-B (KL10-PA processor) except for the following.

1. Only the JSR or XPCW instruction is allowed as an interrupt instruction; that is, as the first instruction executed as a result of an interrupt. Any other instruction will halt the processor (Paragraph 4.4).

2. The only interrupt function implemented for devices is the dispatch function (i.e., interrupt vector). Unlike the KL10-B, which dispatches to and executes the instruction in the EPT location specified by the vector address, the KS10 first references an EPT location determined by the UBA number (EPT + 100 + CONTROLLER #). It then uses this word as the exec-virtual address of a table and executes the instruction at TABLE + VECTOR/4.

3. The KS10 implements two levels of PIA for I/O (Unibus) devices; one PIA can have a higher priority than the other. The PI level (1–7) assigned to Unibus devices interrupting on BR levels 7 and 6 is set by loading a high level PIA (bits 30–32 of UBA status register). The PI level (1–7) for devices interrupting on BR levels 5 and 4 is set by loading a low level PIA (bits 33–35 of UBA status register).

Table 4-3 lists the hard-wired interrupt vectors and BR levels for the various KS10 I/O (Unibus) devices in a fully configured system. It also indicates which PIA, high or low level, is associated with each device.

4.2.4 Paging

Both TOPS-10 and TOPS-20 paging are implemented in the KS10. The paging mode is selected by bit 21 in the WREBR instruction (Paragraph 4.3.1).

Table 4-3 I/O (Unibus) Device Vectors and BR Levels

Device	UBA #	PIA	Interrupt Vector	BR
RH11 #1 (RP06)	1	HI	254	6
RH11 #3 (TU45)	3	HI	224	6
LP20 #1	3	LO	750	4
DZ11 #1	3	LO	340	5
DZ11 #2	3	LO	350	5
DZ11 #3	3	LO	360	5
DZ11 #4	3	LO	370	5
KMC11 #1	3	LO	540	5
DUP11 #1	3	LO	570	5
DUP11 #2	3	LO	600	5

4.2.5 KS10 Instruction Set

The KS10 has the same instruction set as the KL10-B (i.e., Model PA Processor-section 0 addressing only) except for the following.

1. The single-precision (without rounding) floating point instructions that facilitate software double-precision operations are **not** supported on the KS10 and will trap as MUUOs. These are:

   a. UFA (Unnormalized Floating Add)
   b. DFN (Double Floating Negate)
   c. FADL (Floating Add Long)
   d. FSBL (Floating Subtract Long)
   e. FMPL (Floating Multiply Long)
   f. FDVL (Floating Divide Long).

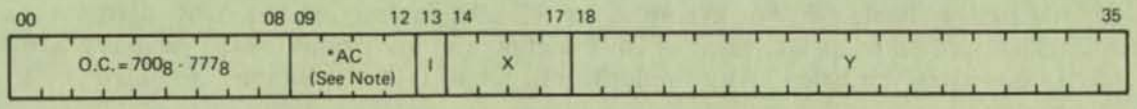
2. The KS10 checks several MBZ (must be zero) fields in the extended instruction set that are not checked by the KL10-B. Any nonzero fields cause an MUUO trap.

3. In KI paging mode, if a MAP instruction is done to a page with A = 0 in the page table entry, the KS10 returns the address it was given. The KL10 returns zero as an address.

4. All KL10 I/O instructions have been replaced by a new I/O instruction set for the KS10. Because the KS10 I/O instructions do not specify a device code, instruction format has been changed to conform to the basic instruction format of op code, AC, and effective address. The KS10 I/O instructions are described in Paragraph 4.3.

4.3 KS10 I/O INSTRUCTIONS

KS10 I/O instructions have the same basic format as the rest of the KS10 instruction set. (Format is shown in Figure 4-2.) I/O instruction op codes are in the range 700–777 (octal). Op code assignments are shown in Table 4-4.



O.C. = OP CODE (TABLE 4-4)

AC = ACCUMULATOR

I = INDIRECT ADDRESSING BIT

X = INDEX REGISTER

Y = ADDRESS

*NOTE:
AC FIELD USED AS OP-CODE EXTENSION FOR APR I/O INSTRUCTIONS (TABLE 4-5).

MR-0228

Figure 4-2 I/O Instruction Format

Table 4-4 I/O Instruction Op Codes (Octal)

	0	1	2	3	4	5	6	7
700	APR0	APR1	APR2	-	UMOVE	UMOVEM	-	-
710	TIOE	TION	RDIO	WRIO	BSIO	BCIO	-	-
720	TIOEB	TIONB	RDIOB	WRIOB	BSIOB	BCIOB	-	-
730	-	-	-	-	-	-	-	-
740	-	-	-	-	-	-	-	-
750	-	-	-	-	-	-	-	-
760	-	-	-	-	-	-	-	-
770	-	-	-	-	-	-	-	-

4.3.1 Internal (APR) I/O Instructions

The internal I/O instructions (APR0-2; op codes 700–702) use the AC field as an extension of the op code as indicated in Table 4-5. For example, the RDEBR instruction (an APR instruction with op code = 701) is specified by an AC value of 24. Function and bit format for the various KS10 internal I/O instructions are given below. Any similarities to KL10 I/O instructions are noted.

Table 4-5 AC Field Assignments (Octal) for APR I/O Instructions

AC	700	701	702
00	APRID	-	RDSPB
04	-	RDUBR	RDCSB
10	-	CLRPT	RDPUR
14	-	WRUBR	RDCSTM
20	WRAPR	WREBR	RDTIME
24	RDAPR	RDEBR	RDINT
30	-	-	RDHSB
34	-	-	-
40	-	-	WRSPB
44	-	-	WRCSB
50	-	-	WRPUR
54	-	-	WRCSTM
60	WRPI	-	WRTIME
64	RDPI	-	WRINT
70	-	-	WRHSB
74	-	-	-

- **APRID (70000)** - The APRID instruction, similar in function to the APRID instruction for the KL10, reads the KS10 microcode version number and CPU serial number, The information is stored in E. Bit format is shown in Figure 4-3.

- **WRAPR (70020)** - WRAPR is an immediate mode instruction used to control the processor. It is analogous to the CONO APR instruction used in the KL10. Bit format is shown in Figure 4-4.

- **RDAPR (70024)** - The RDAPR instruction stores APR status is E. It corresponds to the CONI APR instruction used in the KL10. Bit format is shown in Figure 4-5.

- **WRPI (70060)** - The WRPI instruction is identical to the KL10 CONO PI instruction except that bits 18–20 (write even parity) are not implemented. Bit format is shown in Figure 4-6.

- **RDPI (70064)** - The RDPI instruction is identical to the KL10 CONI PI instruction except that bits 18–20 read no status (write parity is not implemented). Bit format is shown in Figure 4-7.

- **RDUBR (70104)** - The RDUBR instruction, which is similar to the KL10 DATAI PAG instruction, reads the user base register (UBR) and stores the information in E. The word stored is in exactly the same format as used by the WRUBR instruction. In order to allow the word to be used directly (by a WRUBR), bits 0 and 2 are set to one in the result. Bit format is shown in Figure 4-8.

- **CLRPT (70110)** - The CLRPT instruction is similar to the KL10 CLRPT instruction. It clears the hardware page table so that the next reference to the word at E will cause a refill cycle. There is only one entry in the page table for any virtual page. Clearing the mapping information for a page clears both the EXEC and USER mapping. Bit format is shown in Figure 4-9.

- **WRUBR (70114)** - The WRUBR instruction, which is similar to the KL10 DATAO PAG instruction, loads the UBR with the word at E. Bit format is shown in Figure 4-10.

- **WREBR (70120)** - The WREBR instruction loads the executive base register (EBR) from E. It is similar in function to the KL10 CONO PAG instruction. Bit format is shown in Figure 4-11.

- **RDEBR (70124)** - The RDEBR instruction reads the EBR into the right half of E. It is comparable to the KL10 CONI PAG instruction. Bit format is when in Figure 4-12.

- **RDSPB (70200)** - The RDSPB instruction reads the shared pointer table (SPT) base register and stores the value in E. Bit format is shown in Figure 4-13.

- **RDCSB (70204)** - The RDCSB instruction reads the core status table (CST) base register and stores the value in E. Bit format is shown in Figure 4-14.

- **RDPUR (70210)** - The RDPUR instruction reads the process use register (PUR) and stores the value in E. Bit format is shown in Figure 4-15.

- **RDCSTM (70214)** - The RDCSTM instruction reads the CST mask register and stores the value in E. Bit format is shown in Figure 4-16.

- **RDTIME (70220)** - The RDTIME instruction is similar to the RDTIME instruction for the KL10. It reads the time base and stores the double-word value in E and E + 1. The time base upcounts at 4.096 mHz. Bit format is shown in Figure 4-17.

- **RDINT (70224)** -The RDINT instruction reads the current value of the inerval timer period register and stores the value in E. Bit format is shown in Figure 4-18.

- **RDHSB (70230)** - The RDHSB instruction stores the value of the halt status block address at E. Bit format is shown in Figure 4-19.

- **WRSPB (70240)** - The WRSPB instruction loads the SPT base register from E. Bit format is shown in Figure 4-20.

- **WRCSB (70244)** - The WRCSB instruction loads the CST base register from E. Bit format is shown in Figure 4-21.

- **WRPUR (70250)** - The WRPUR instruction loads the PUR from E. (Bit format is shown in Figure 4-22.) The PUR contains the AGER in the left-most bits. These bits are cleared by ANDing the CST entry with the CST mask; then the entire PUR is ORed with the CST entry.

- **WRCSTM (70254)** - The WRCSTM instruction loads the CST mask register from E. The CST mask register should contain a zero for every bit in the AGER and a one in all other bit positions. Bit format is shown in Figure 4-23.

- **WRTIME (70260)** - The WRTIME instruction loads the double-word at E and E + 1 into the time base. (Bit format is shown in Figure 4-14.) The Time Base up-counts at 4.096 mHz.

- **WRINT (70264)** – The WRINT instruction loads the interval timer period register from E. The binary number (n) that is loaded determines the interval; that is, the interval (period) = n milliseconds. Bit format for the instruction is shown in Figure 4-25.

- **WRHSB (70270)** – The WRHSB instruction loads the signed word at E as the address of the halt status block (Paragraph 4.7.2). If the word is negative or zero, no halt status will subsequently be stored. If the word is positive, the halt status block (20 words) will be stored starting at the specified address. Initially, when the microcode is loaded and started, the halt status block address is set to a value of (+) 376000. Bit format for the WRHSB instruction is shown in Figure 4-26.

4.3.2 External I/O Instructions

The external I/O instructions read, write, modify, and test registers in KS10 devices external to the CPU. The effective address (E) for these instructions specify an I/O address; the specified AC holds register read/write data or mask data (for test or modification) depending on the instruction type. Both full-word (normal) instructions and byte instructions are implemented. The full-word instructions transfer 36 bits of data and use the full contents of an AC. The byte instructions, which are employed only when addressing Unibus device registers, transfer only eight bits of data and use only the eight right-most bits in an AC. The various external I/O instructions are described below.

The I/O address generated by the external I/O instructions' effective address consists of a controller number and a register address. Bit format and the general ranges of controller number and register address assignments are given in Figure 4-27. The specific I/O addresses for a fully configured KS10 are listed in Table 4-6.

- **TIOE and TIOEB (710 and 720)** – The TIOE (or TIOEB) instruction fetches one word (or byte) from the I/O address specified by E, and ANDs the word (or byte) with the contents of the specified AC. The instruction skips if the result of the AND is zero. The contents of the AC are not modified.

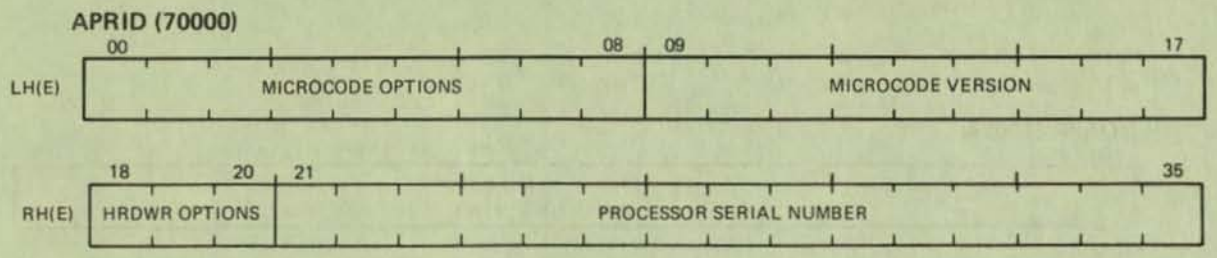
- **TION and TIONB (711 and 721)** – The TION (or TIONB) instruction performs the same function as the TIOE (or TIOEB) instruction except that the instruction skips if the result of the AND is **not** zero.

- **RDIO and RDIOB (712 and 722)** – The RDIO (or RDIOB) instruction fetches the word (or byte) from the I/O address specified by E and stores the word (or byte) right-justified in the specified AC.

- **WRIO and WRIOB (713 and 723)** – The WRIO (or WRIOB) instruction takes the word (or byte) contained in the specified AC and transfers the word (or byte) to the I/O address specified by E.

- **BSIO and BSIOB (714 and 724)** – The BSIO (or BSIOB) instruction fetches the word (or byte) from the I/O address specified by E, ORs the word (or byte) with the contents of the specified AC, and then transfers the result back to the I/O address. The instruction(s) may be used to set selected bits in Unibus device registers. The contents of the AC are not modified.

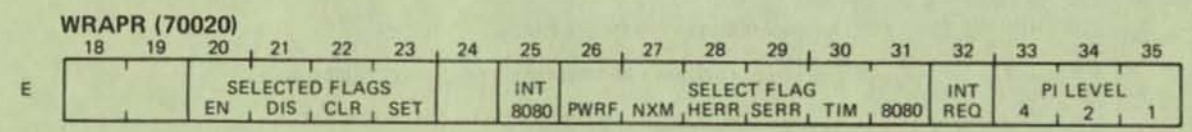
- **BCIO and BCIOB (715 and 725)** – The BC10 (or BC10B) instruction is similar to BS10 (or BS10B) instruction except that the word (or byte) read from I/O address is ANDed with the complement of the AC contents. The instruction(s) may be used to clear selected bits in Unibus device registers. The contents of the AC are not modified.



BIT(S)	FUNCTION
0-8	RESERVED FOR MICROCODE VERSION
9-17	MICROCODE VERSION NUMBER
18-20	HARDWARE OPTIONS (BITS CURRENTLY = 0)
21-35	PROCESSOR SERIAL NUMBER

MR-0229

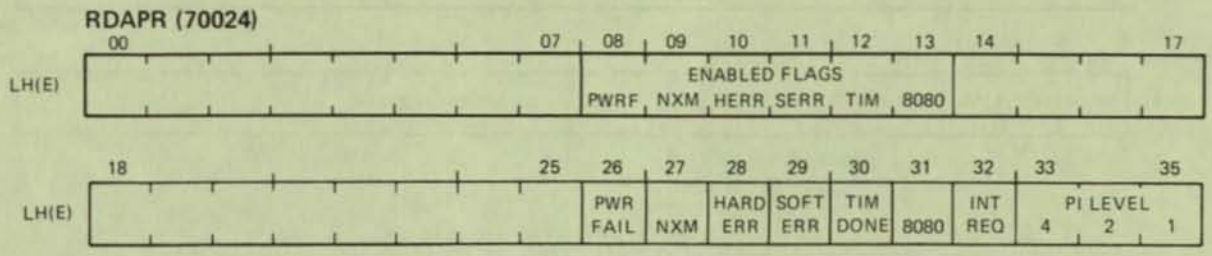
Figure 4-3 APRID Instruction



BIT(S)	FUNCTION
20	ENABLE CONDITIONS SELECTED BY BITS 26-31 TO CAUSE INTERRUPTS
21	DISABLE INTERRUPTS FOR CONDITIONS SELECTED BY BITS 26-31
22	CLEAR FLAGS INDICATED BY BITS 26-31
23	SET FLAGS INDICATED BY BITS 26-31
25	INTERRUPT 8080 CONSOLE
26	POWER FAIL
27	NON-EXISTENT MEMORY ERROR
28	HARD MEMORY ERROR (CANNOT BE CORRECTED BY ECC)
29	SOFT MEMORY ERROR (CORRECT DATA PLACED ON BUS)
30	INTERVAL TIMER
31	8080 CONSOLE
32	GENERATE INTERRUPT REQUEST
33-35	PIA

MR-0230

Figure 4-4 WRAPR Instruction



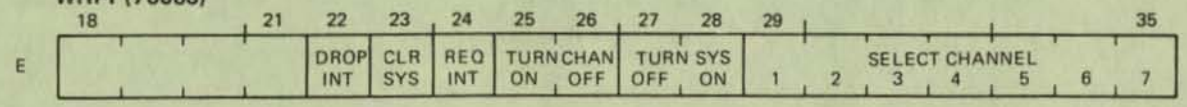
BIT(S)	FUNCTION
08	POWER FAIL ENABLED
09	NON-EXISTENT MEMORY ERROR ENABLED
10	HARD MEMORY ERROR INTERRUPT ENABLED
11	SOFT MEMORY ERROR INTERRUPT ENABLED
12	INTERVAL TIMER ENABLED
13	8080 CONSOLE INTERRUPT ENABLED
26	POWER FAIL ERROR
* 27	NON-EXISTENT MEMORY ERROR
* 28	HARD MEMORY ERROR (CANNOT BE CORRECTED BY ECC)
29	SOFT MEMORY ERR (CORRECT DATA PLACED ON BUS)
30	INTERVAL TIMER DONE
31	8080 CONSOLE INTERRUPT
32	INTERRUPT REQUESTED
33-35	PIA

*NOTE:
PAGE FAIL OCCURS IF ERROR IS RESULT OF CPU MEMORY REQUEST, NXM FLAG ALSO SETS IN UNIBUS DEVICE IF ERROR IS RESULT OF UNIBUS NPR REQUEST.

MR-0231

Figure 4-5 RDAPR Instruction

WRPI (70060)

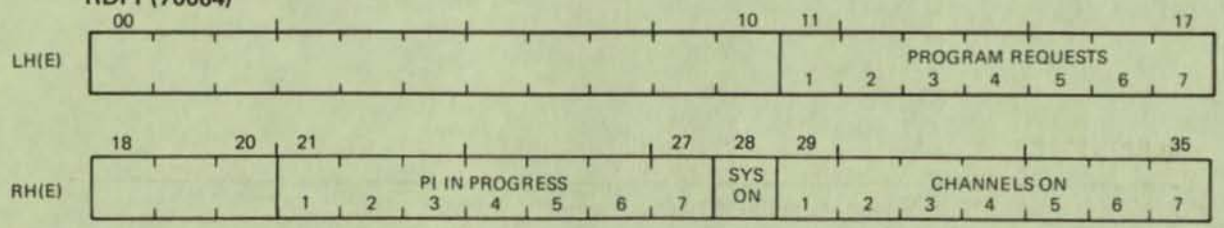


BIT(S)	FUNCTION
22	DROP PROGRAM REQUESTS ON SELECTED CHANNELS
23	CLEAR PI SYSTEM
24	INITIATE INTERRUPTS ON THE SELECTED CHANNELS
25	TURN ON THE SELECTED CHANNELS
26	TURN OFF THE SELECTED CHANNELS
27	DEACTIVATE THE PI SYSTEM
28	ACTIVATE THE PI SYSTEM
29-35	SELECT CHANNELS FOR BITS 22, 24, 25, AND 26

MR 0232

Figure 4-6 WRPI Instruction

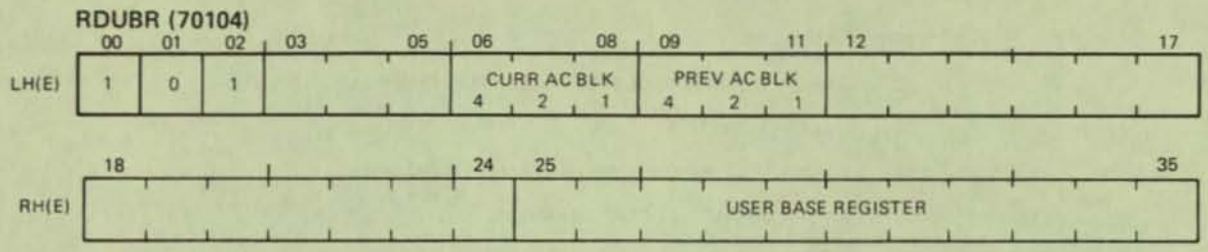
RDPI (70064)



BIT(S)	FUNCTION
11-17	PROGRAM REQUESTS ON CHANNELS 1-7
21-27	INTERRUPTS HOLDING (IN PROGRESS) ON CHANNELS 1-7
28	PI SYSTEM ON
29-35	ACTIVE CHANNELS 1-7

MR 0233

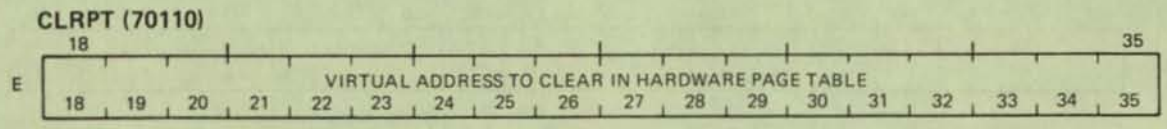
Figure 4-7 RDPI Instruction



BIT(S)	FUNCTION
0	1
1	0
2	1
6-8	CURRENT AC BLOCK
9-11	PREVIOUS AC BLOCK
25-35	USER BASE REGISTER

MR-0234

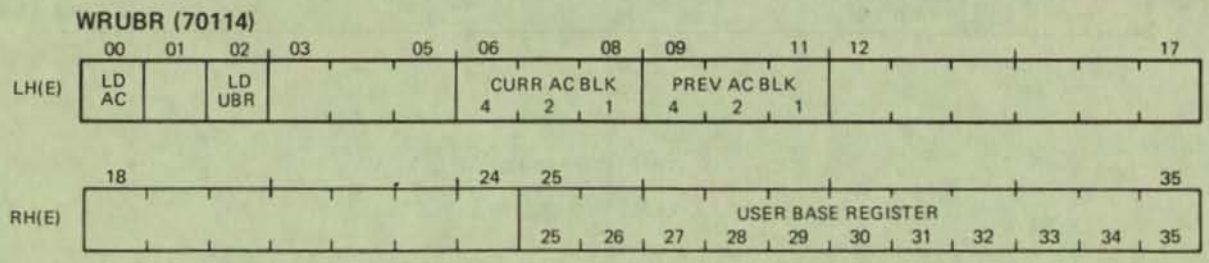
Figure 4-8 RDUBR Instruction



BIT(S)	FUNCTION
18-35	VIRTUAL ADDRESS TO CLEAR IN HARDWARE PAGE TABLE

MR-0235

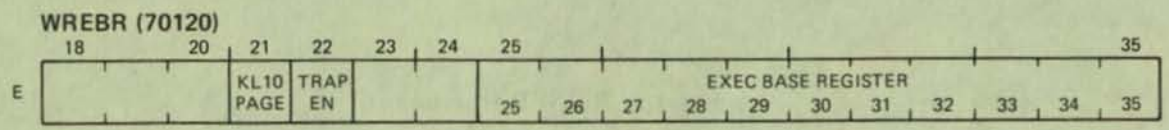
Figure 4-9 CLRPT Instruction



BIT(S)	FUNCTION
0	LOAD AC BLOCK NUMBERS
2	LOAD UBR
6-8	CURRENT AC BLOCK
9-11	PREVIOUS AC BLOCK
25-35	USER BASE REGISTER (PHYSICAL PAGE NUMBER OF UPT)

MR-0236

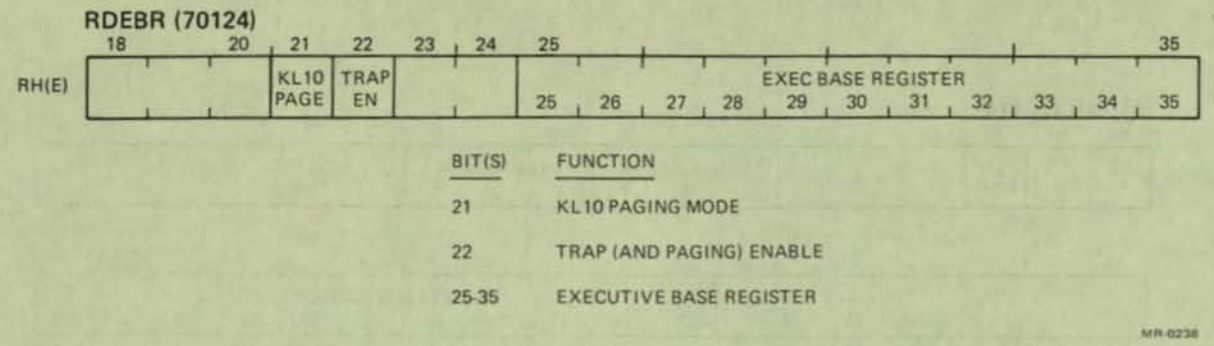
Figure 4-10 WRUBR Instruction



BIT(S)	FUNCTION
21	KL10 PAGING MODE
22	TRAP (AND PAGING) ENABLE
25-35	EXECUTIVE BASE REGISTER (PHYSICAL PAGE NUMBER OF EPT)

MR-0237

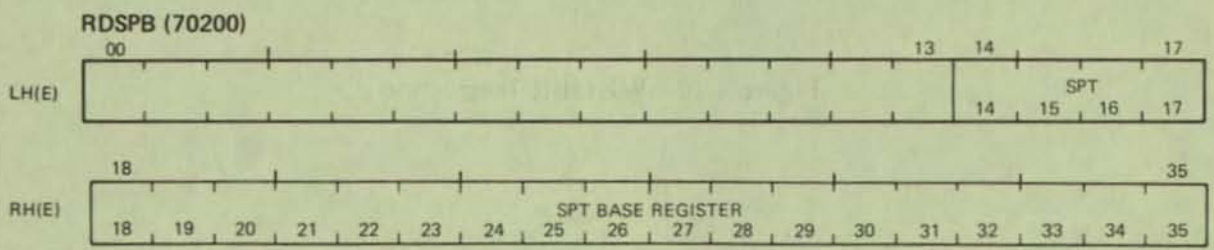
Figure 4-11 WREBR Instruction



BIT(S)	FUNCTION
21	KL10 PAGING MODE
22	TRAP (AND PAGING) ENABLE
25-35	EXECUTIVE BASE REGISTER

MR-0238

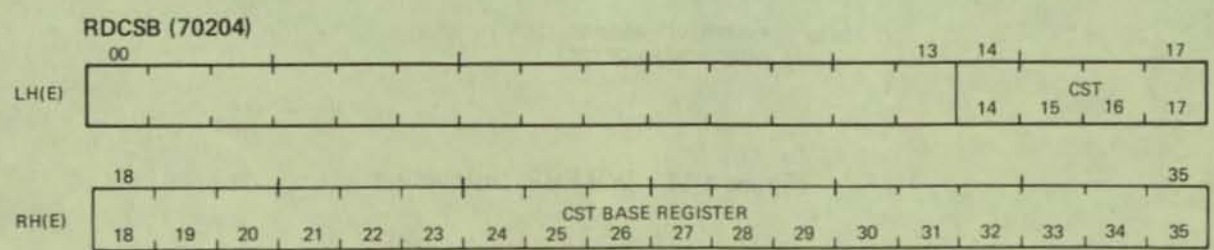
Figure 4-12 RDEBR Instruction



BIT(S)	FUNCTION
14-35	SPT (SHARED POINTER TABLE) BASE REGISTER

MR-0239

Figure 4-13 RDSPB Instruction



BIT(S)	FUNCTION
14-35	CST (CORE STATUS TABLE) BASE REGISTER

MR-0240

Figure 4-14 RDCSB Instruction

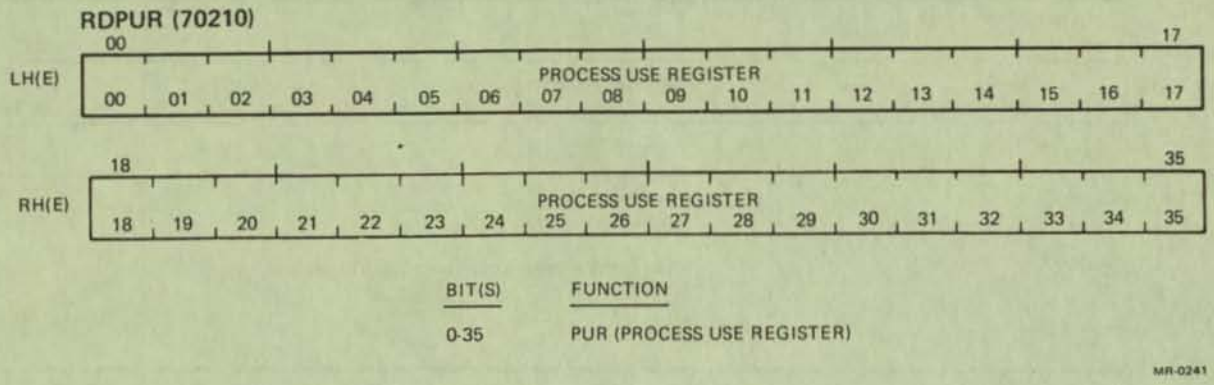


Figure 4-15 RDPUR Instruction

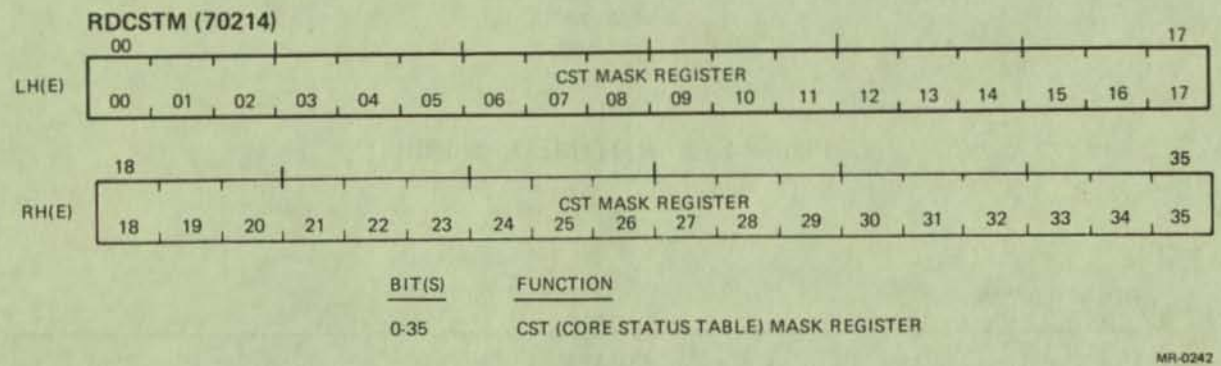


Figure 4-16 RDCSTM Instruction

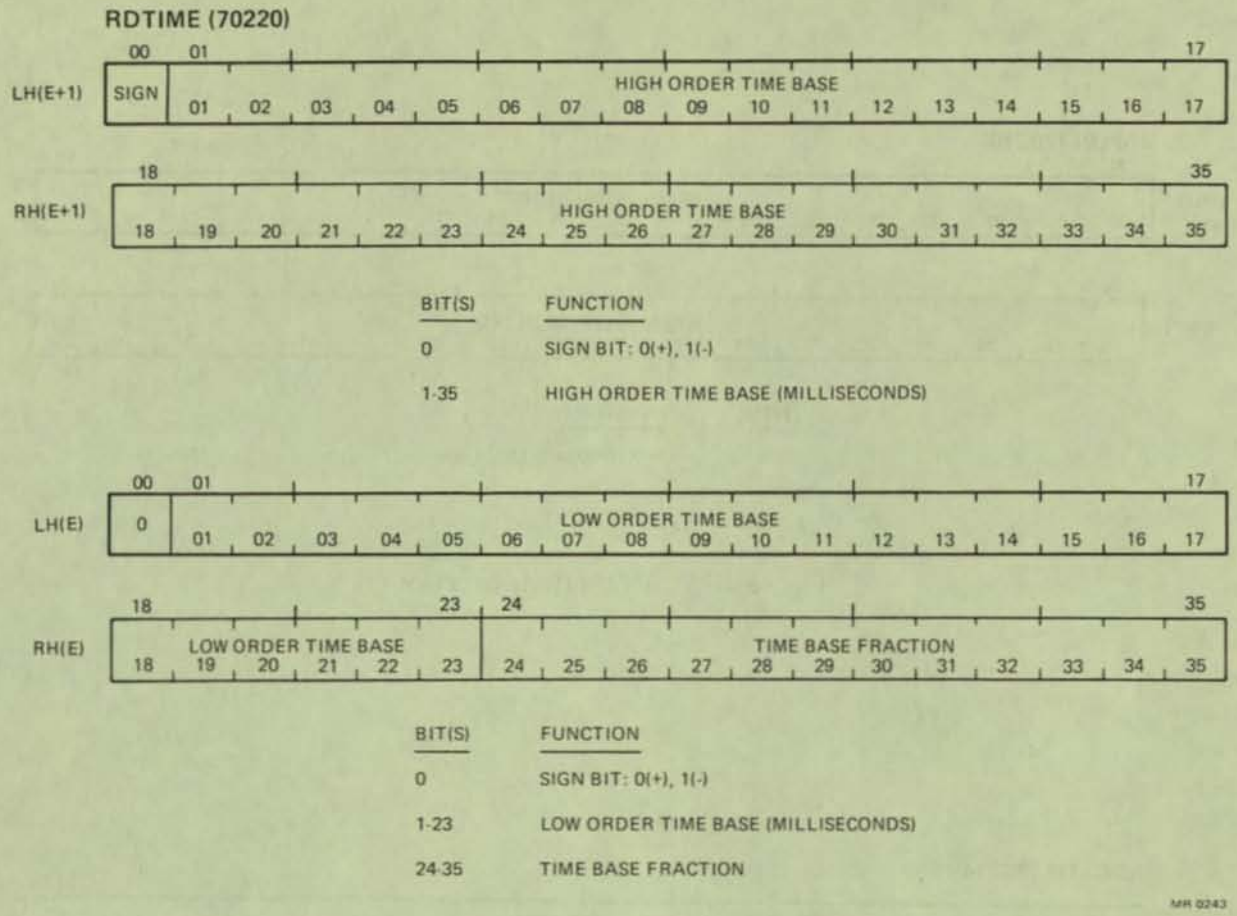


Figure 4-17 RDTIME Instruction

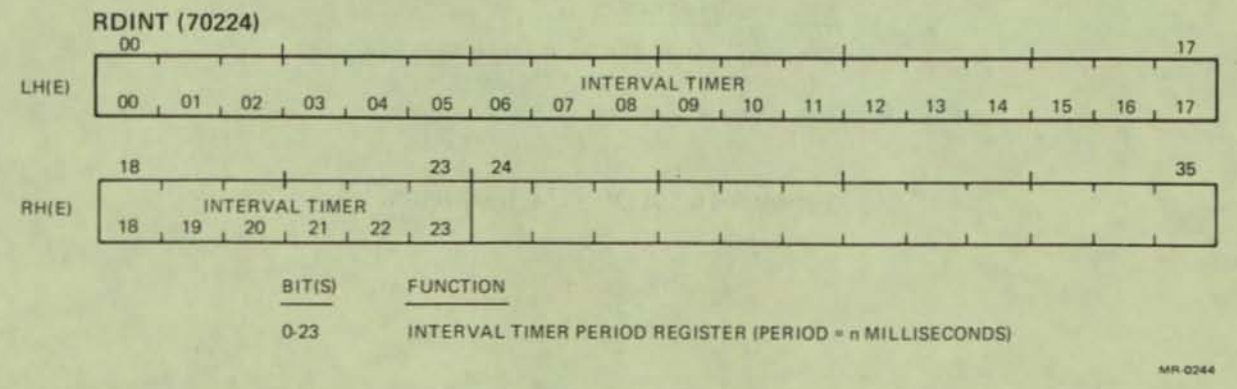
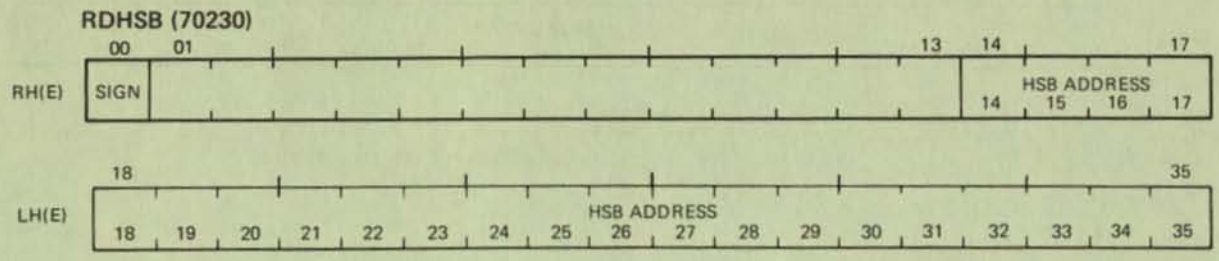


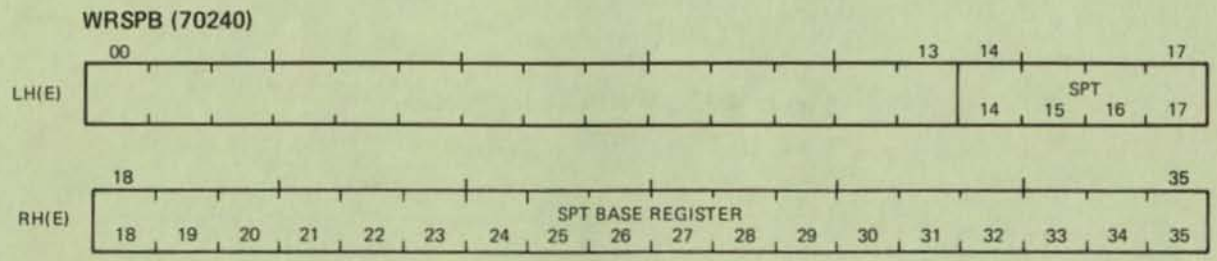
Figure 4-18 RDINT Instruction



BIT(S)	FUNCTION
0	SIGN BIT = 0 = STORE HALT STATUS SIGN BIT = 1 = DO NOT STORE HALT STATUS
14-35	HSB (HALT STATUS BLOCK) ADDRESS (BIT 00 = 0)

MR-0245

Figure 4-19 RDHSB Instruction



BIT(S)	FUNCTION
14-35	SPT (SHARED POINTER TABLE) BASE REGISTER

MR-0246

Figure 4-20 WRSPB Instruction

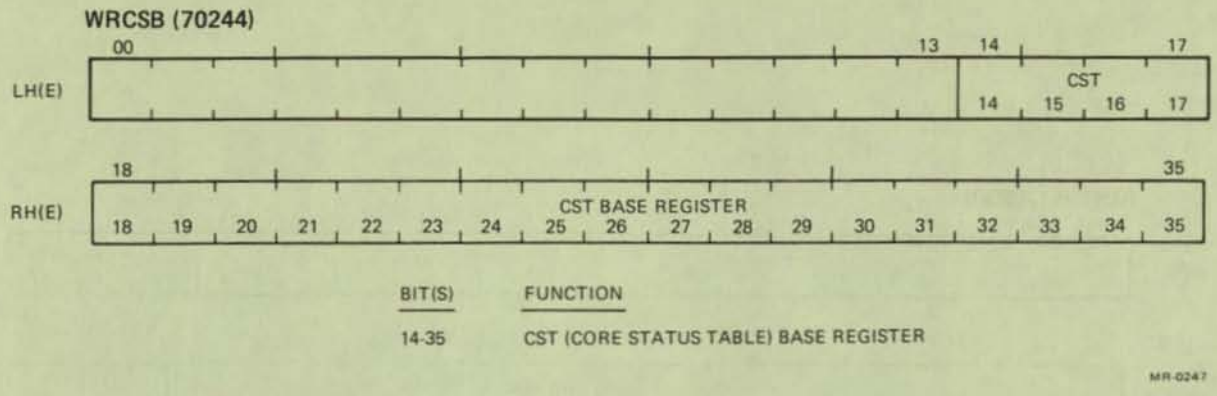


Figure 4-21 WRCSB Instruction

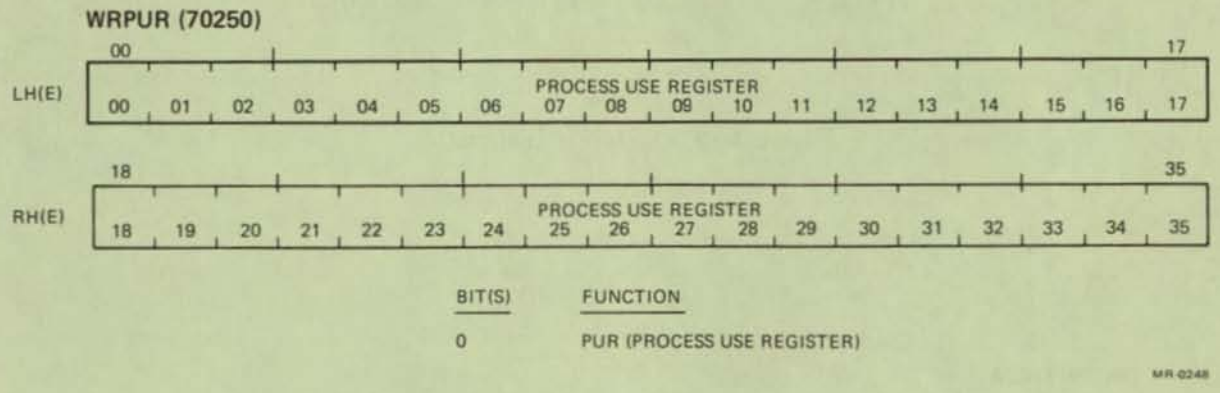


Figure 4-22 WRPUR Instruction

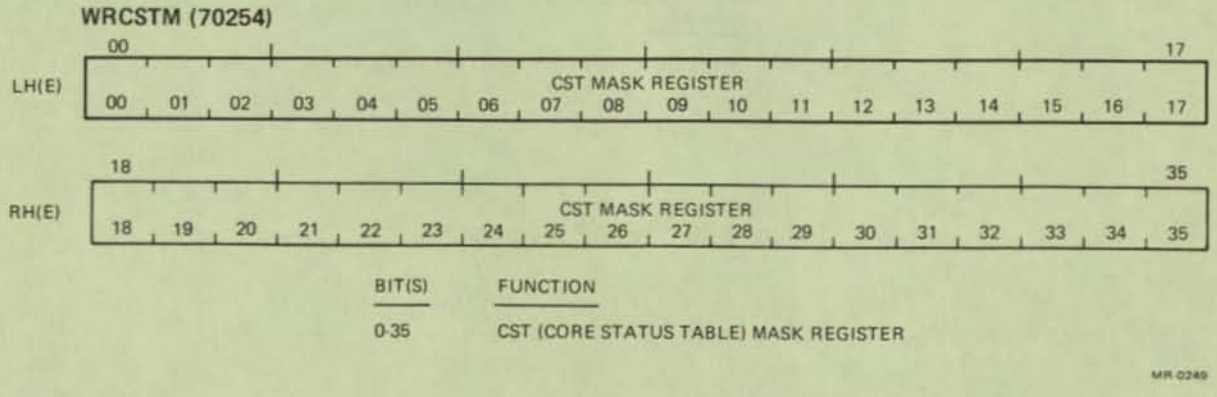


Figure 4-23 WRCSTM Instruction

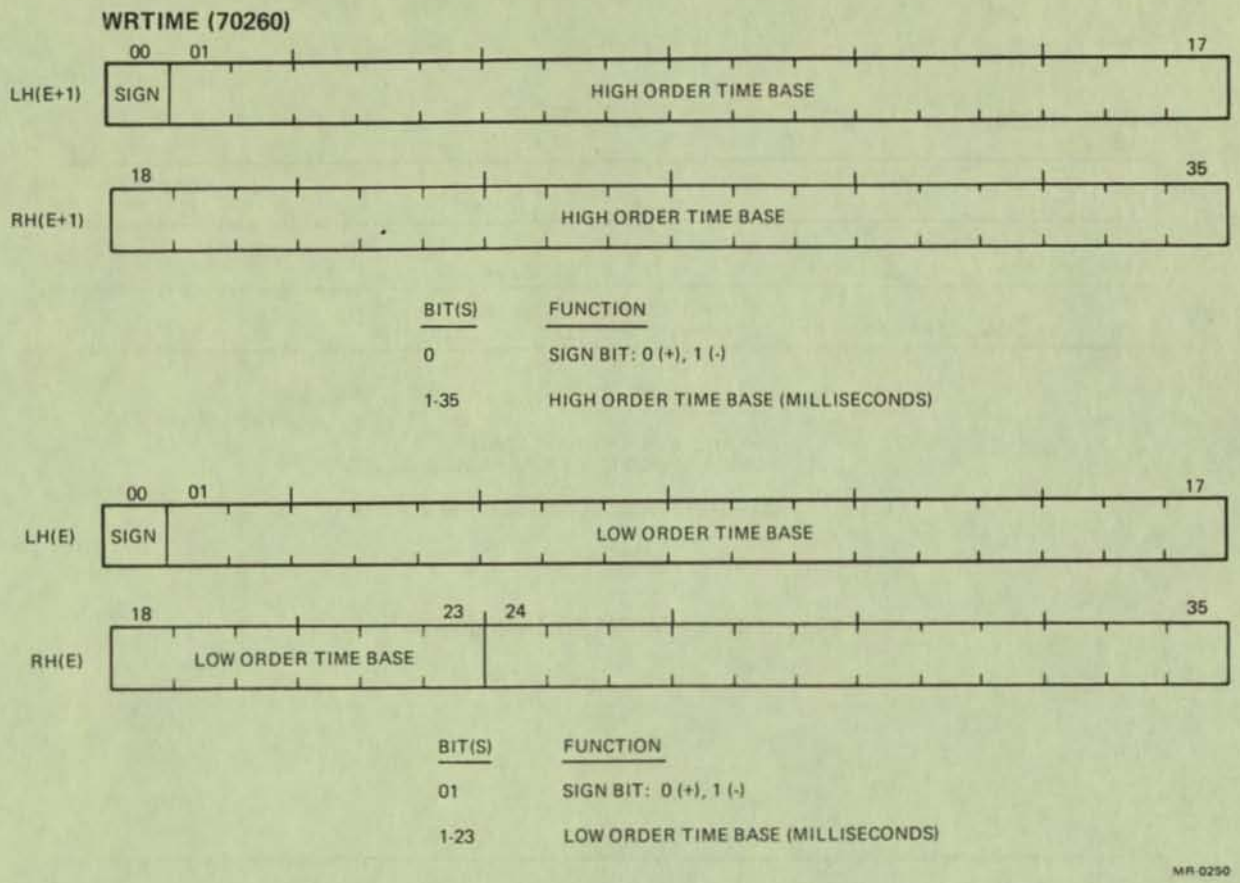


Figure 4-24 WRTIME Instruction

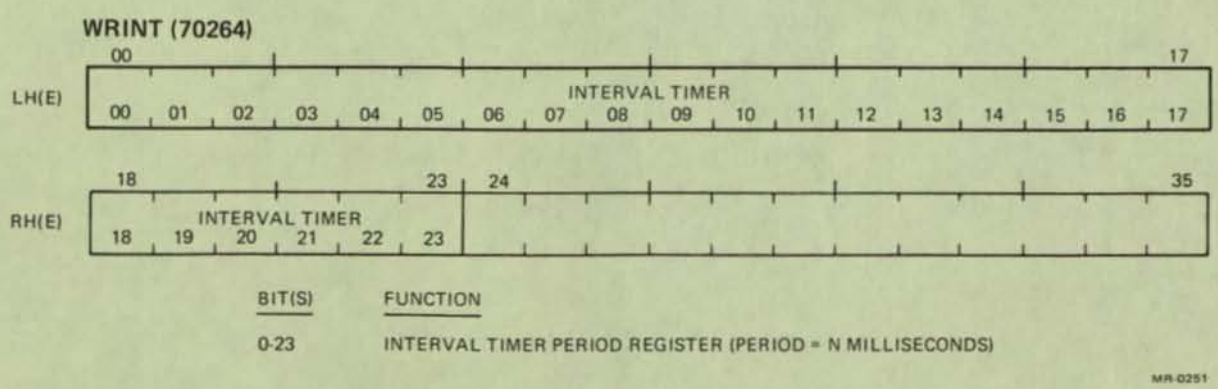
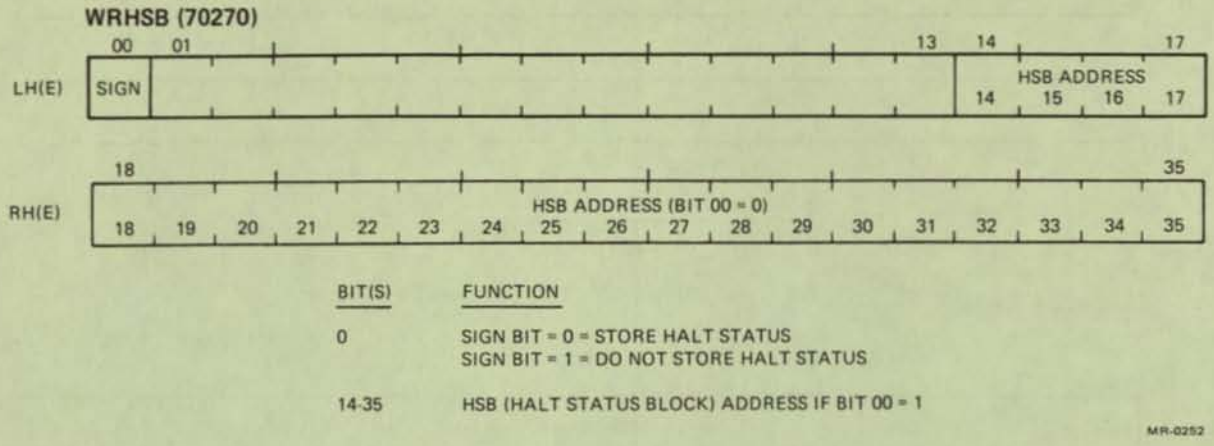


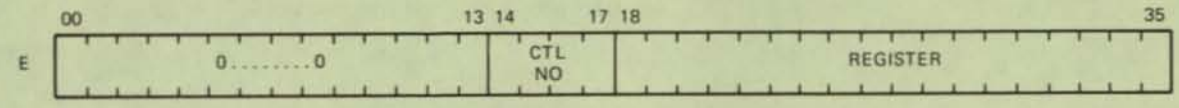
Figure 4-25 WRINT Instruction



BIT(S)	FUNCTION
0	SIGN BIT = 0 = STORE HALT STATUS SIGN BIT = 1 = DO NOT STORE HALT STATUS
14-35	HSB (HALT STATUS BLOCK) ADDRESS IF BIT 00 = 1

MR-0252

Figure 4-26 WRHSB Instruction



CONTROLLER NUMBER	REGISTER ADDRESS	REGISTER(S)
0	0-077777	NOT USED
0	100000	MEMORY STATUS REGISTER
0	100000-1-177777	NOT USED
0	200000	CONSOLE INSTRUCTION REGISTER
0	20000001-777777	NOT USED
1	0-377777	NOT USED
1	400000-777777	UNIBUS 1 (UBA AND DEVICE) REGISTERS
3	0-377777	NOT USED
3	400000-777777	UNIBUS 3 (UBA AND DEVICE) REGISTERS
2, 4-17		NOT USED

MR-0253

Figure 4-27 I/O Address Format

Table 4-6 External I/O Addresses

Register(s)	KS10 Bus Device	Unibus Device	CTL #	Register Address
Memory Status Register	Memory Cont.		0	100000
Console Instruction Register	Console		0	200000
UBA Paging RAM	UBA1		1	763000-77
UBA Status Register	UBA1		1	763100
UBA Maintenance Register	UBA1		1	763101
		Unibus 1		
	UBA1	RH11 # 1 (RP06)	1	776700*
UBA Paging RAM	UBA3		3	763000-77
UBA Status Register	UBA3		3	763100
UBA Maintenance Register	UBA3		3	763101
		Unibus 3		
	UBA3	RH11 # 3 (TU45)	3	772440*
	UBA3	LP20 # 1	3	775400*
	UBA3	DZ11 # 1	3	760010*
	UBA3	DZ11 # 2	3	760020*
	UBA3	DZ11 # 3	3	760030*
	UBA3	DZ11 # 4	3	760040*
	UBA3	KMC11 # 1	3	760540*
	UBA3	DUP11 # 1	3	760300*
	UBA3	DUP11 # 2	3	760310*

NOTE
An asterisk (*) indicates address is a base address.

Note that an extended address (greater than 18 bits) is required to address controllers other than zero. The effective address calculation for external I/O instructions (diagramed in Figure 4-28) is as follows.

1. If there is no indexing or indirection, Y is used as the effective address.
2. If there is indirection (or indirection and indexing), Y (or Y indexed by bits 18–35 of the index register) is used as an address for an indirect word fetch and the contents of the indirect word (bits 14–35) are used as the effective address.
3. If there is indexing (and no indirection) and the left half of the index register is less than or equal to zero, Y indexed by bits 18–35 of the index register is used as the effective address.
4. If there is indexing (and no indirection) and the left half of the index register is positive, Y indexed by bits 06–35 of the index register is used as the effective address.

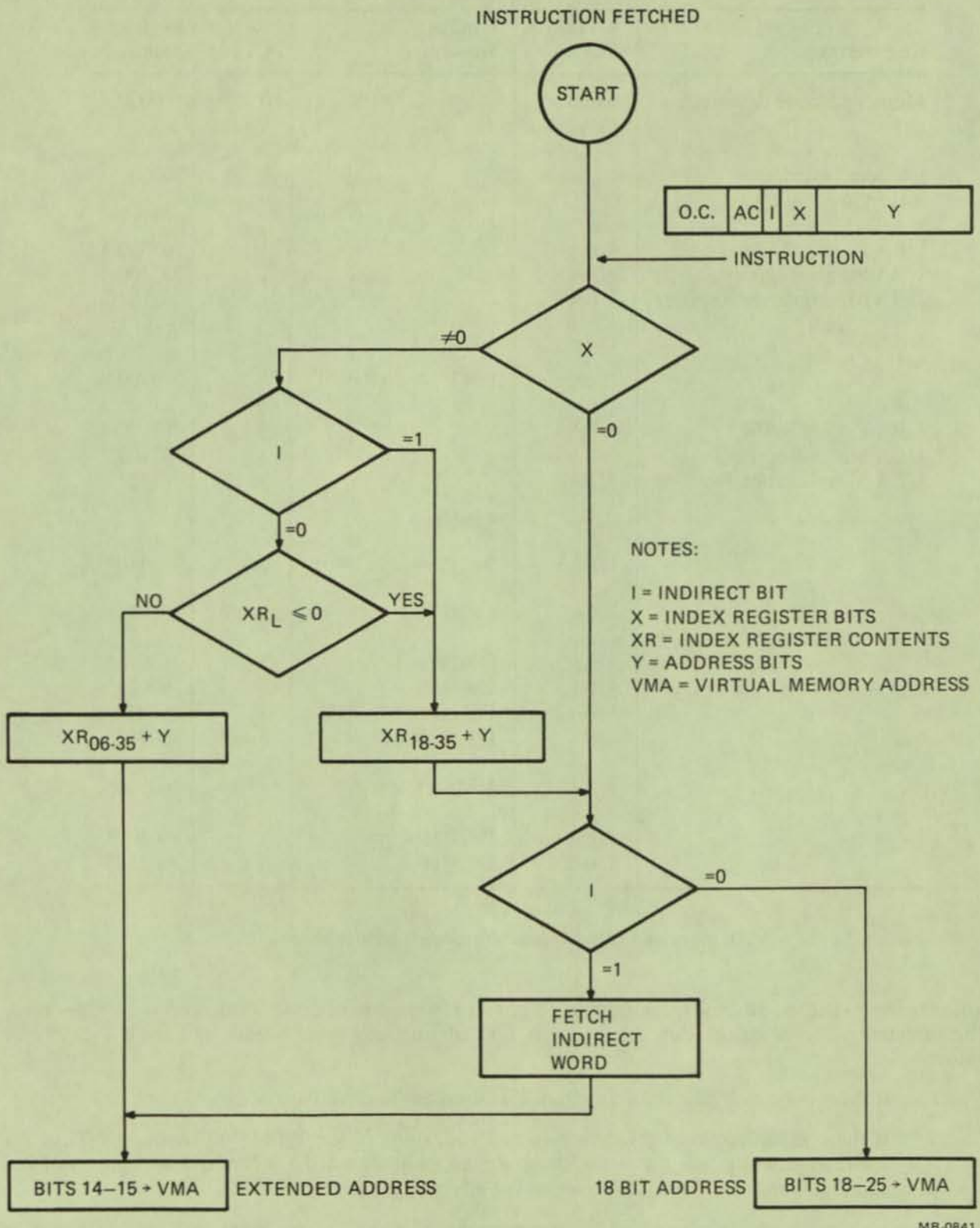


Figure 4-28 Effective Address Calculation for External I/O Instructions

The operations described above are summarized in Table 4-7. The result of the effective address calculation, which can be either an 18-bit or an extended I/O address is also indicated.

Table 4-7 Summary of Effective Address Calculation for External I/O Instructions

Indirection	Indexing	XRL	Effective Address	Comments
NO	NO	-	Y	18-bit Address
YES	NO	-	(Y)	Extended Address
YES	YES	-	(Y + XR18–35)	Extended Address
NO	YES	≤0	Y + XR18–35	18-bit Address
NO	YES	>0	Y + XR06–35	Extended Address

It can be seen that to address controllers other than zero, which require an extended address, instructions using indexing or indirect addressing must be used. Examples of each follow, both of which read the status register (register address = 763100) in UBA #1 (controller number = 1) into an AC with the RDIO external I/O instruction.

Example 1 (using indexing):

RDIO AC, 763100(X) where the contents of index register X = 1000000

The index register contents (the controller number) is added to Y (the register address) to give the extended I/O address 1736100.

Example 2 (using indirection):

RDIO AC, @ 100 where the contents of 100 = 1763100

An indirect word fetch of location 100 is made and the contents are used to generate the extended I/O address 1763100.

4.3.3 PXCT Extensions

The UMOVE and UMOVEM instructions have been originated for the KS10 to save time and space in the monitor.

- **UMOVE (704)** – The UMOVE (move from previous context) instruction performs the same functions as:

  PXCT 4, [MOVE AC,E]

- **UMOVEM (705)** – The UMOVEM (move to previous context) instruction performs the same function as:

  PXCT 4, [MOVEM AC,E]

4.4 KS10 PROCESSOR STATUS WORDS

Whenever the KS10 processor halts, it writes a halt status word, the PC, and (optionally) a halt status block of 18 words in memory.

The halt status word contains a code indicating the type of halt; the halt status block contains a read-out of several CPU registers (HR, VMA, etc.) at the time of the halt. Other KS10 status words include the page fail word, which is written into the UPT following a page failure; and the PC word (PC with flags), which is stored in an AC or memory location by certain system level instructions.

4.4.1 Halt Status Word

A halt status code is stored in physical memory location 0 (not AC0) whenever the KS10 processor halts. Codes in the range 0–77 (octal) indicate normal halts; codes in the range 100–177 (octal) indicate software failures; codes of 1000 (octal) or greater indicate microcode or software failures. Bit format and halt code definitions are given in Figure 4-29.

4.4.2 PC

A processor halt causes the PC to be stored right justified in physical memory location 1 (not AC1).

4.4.3 Halt Status Block

If the halt status block address is positive, a processor halt causes the contents of several processor registers to be stored in a block of KS10 memory starting at the specified address. Figure 4-30 shows the information stored in each location. If the halt status block address is negative, the halt status block is not stored. The WRHSB instruction (Paragraph 4.3.1) allows the program to load any address value. Initially, when the microcode is started, the halt status block address is set to a value of (+) 376000 and the halt status block is stored.

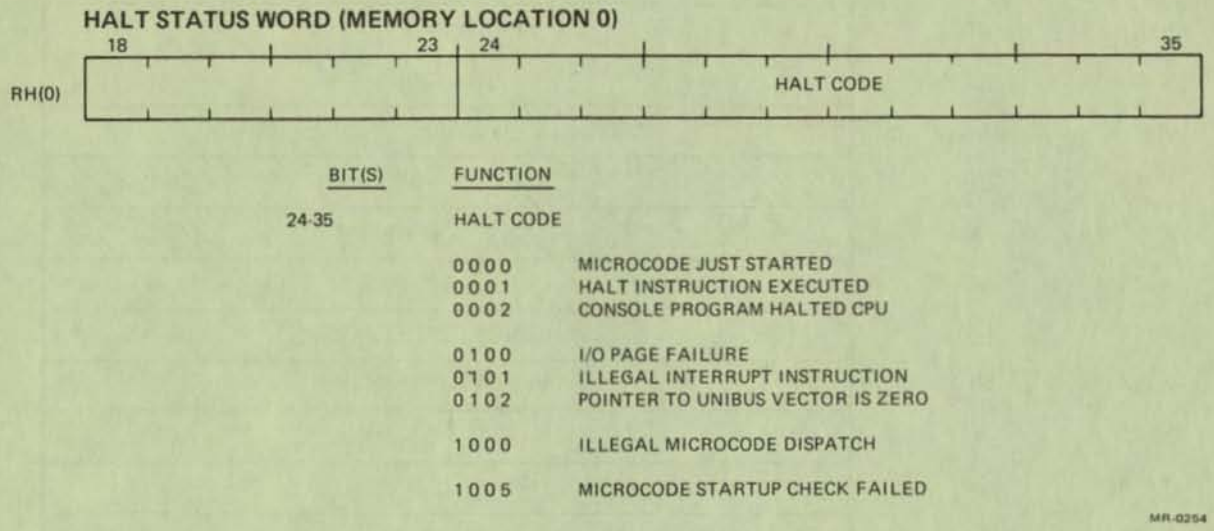


Figure 4-29 Halt Status Word

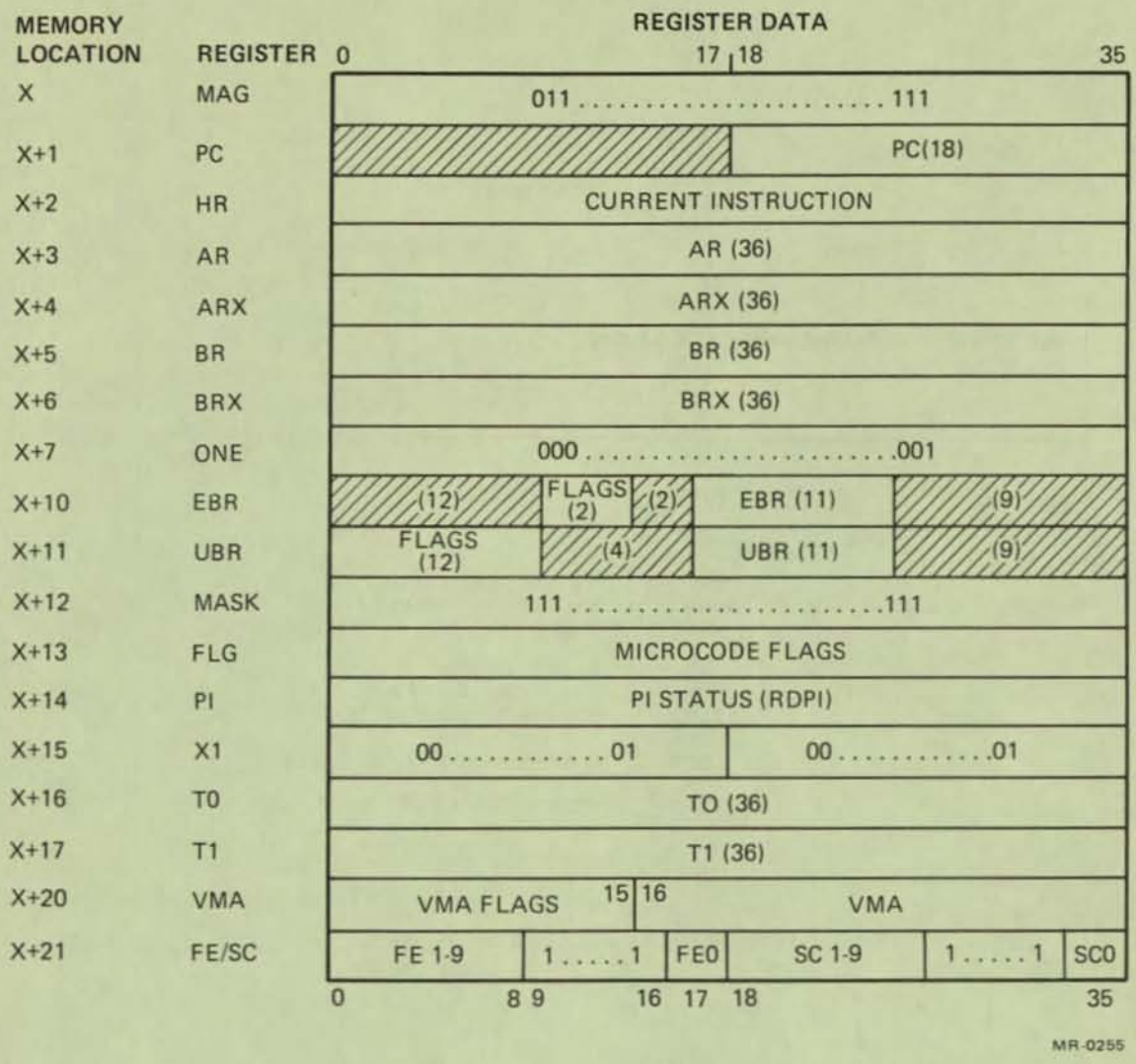
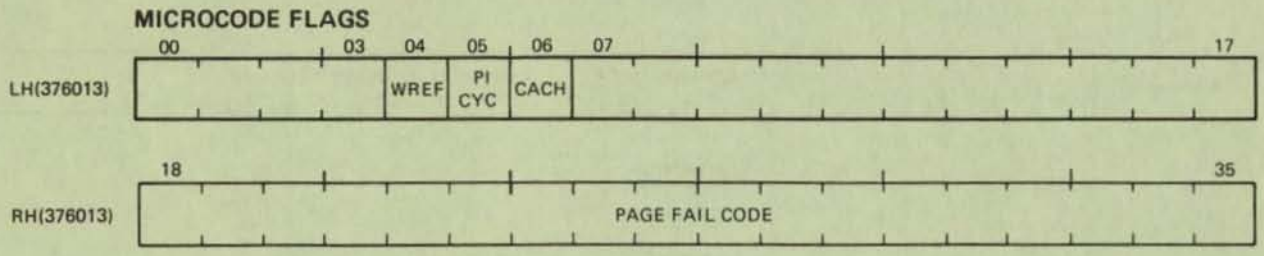


Figure 4-30 Halt Status Block

The first 16 memory locations of the halt status block hold the register data read from the 16-word RAMs associated with the 2901 microprocessor circuits. Significant status information includes the PC also stored in memory location, current instruction, EBR, UBR, microcode flags, and PI system status. (PI system status is the same as that read by RDPI instruction.) Another processor status word, the VMA contents (plus flags), is stored in the next to last location of the halt status block. Bit definitions for the microcode flag and VMA words are given below.

- **Microcode Flags** – In the event of a page failure, three flags and a page fail code are stored as part of the halt status block in X + 13. The page fail code, which is the contents of the microword's magic number field, specifies the operation for which the page failure occurred. Status word bit format and page fail code definitions are given in Figure 4-31.

- **VMA** – The virtual memory address (VMA) and VMA flags are stored in location X + 20 of the halt status block. Bit format and definitions are given in Figure 4-32.

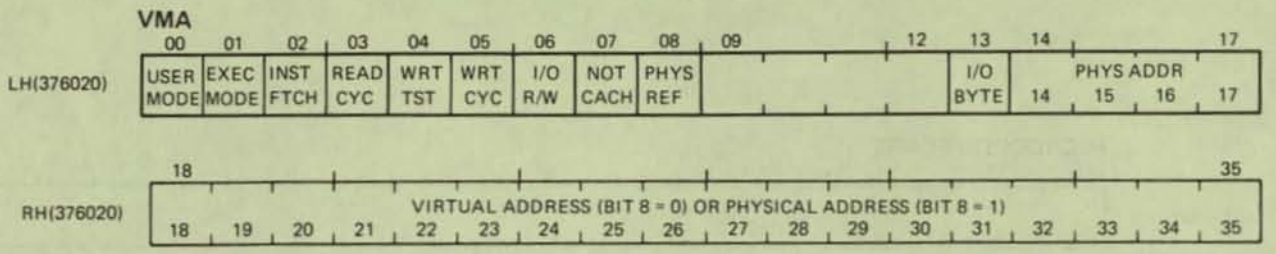


BIT(S)	FUNCTION
4	WRITE REFERENCE BIT FROM PAGE MAP
5	PI CYCLE
6	LOOK IN CACHE BIT FROM PAGE MAP
18-35	PAGE FAIL CODE

000000	SIMPLE INSTRUCTIONS
000001	BLT IN PROGRESS
400002	MAP IN PROGRESS
000003	MOVE STRING SOURCE IN PROGRESS
000004	MOVE STRING FILL IN PROGRESS
000005	MOVE STRING DESTINATION IN PROGRESS
000006	FILLING DESTINATION
000007	EDIT SOURCE
000010	EDIT DESTINATION
000011	CONVERTING DECIMAL TO BINARY
000012	COMPARING DESTINATION

MR-0256

Figure 4-31 Microcode Flags



BIT(S)	FUNCTION
0	USER MODE
1	EXEC MODE
2	INSTRUCTION FETCH
3	READ CYCLE
4	WRITE TEST
5	WRITE CYCLE
6	I/O READ OR WRITE
7	DO NOT LOOK IN CACHE
8	PHYSICAL REFERENCE
13	I/O BYTE INSTRUCTION
14-17	BITS 14-17 OF PHYSICAL ADDRESS (OR 0s)
18-35	BITS 18-35 OF VIRTUAL ADDRESS (BIT 8 = 0) OR PHYSICAL ADDRESS (BIT 8 = 1)

MR-0257

Figure 4-32 VMA

4.4.4 PC Word

Several of the jump instructions (e.g., JSR) save the PC and various processor flags in a memory location or an AC. Bit format for this PC word is shown in Figure 4-33.

4.4.5 Page Fail Word

Following all page failures, except for in-out failures, the processor causes a page fail trap and stores a page fail word in location 500 (octal) of the UPT. Bit format is shown in Figure 4-34.

4.5 KS10 EPT/UPT

Executive process table (EPT) and user process table (UPT) configurations for the KS10 are shown in Figures 4-35 and 4-36.

4.6 OPERATOR CONSOLE

Local operator control of the KS10 is by a set of commands typed at the console terminal (CTY). The CTY connects directly to the 8080-based console hardware via a serial line. A second serial line, that operates in parallel with the first line, may also be connected to the console hardware to allow control of the KS10 by a remote diagnosis link. Other (user only) terminals connect to the KS10 via the Unibus (DZ11s).

The commands typed at the CTY, or entered from the remote diagnosis link, are implemented by the program running in the console module's 8080 microprocessor. The program is resident in PROM and valid at power-up.

The CTY operates in either CTY mode or user mode.

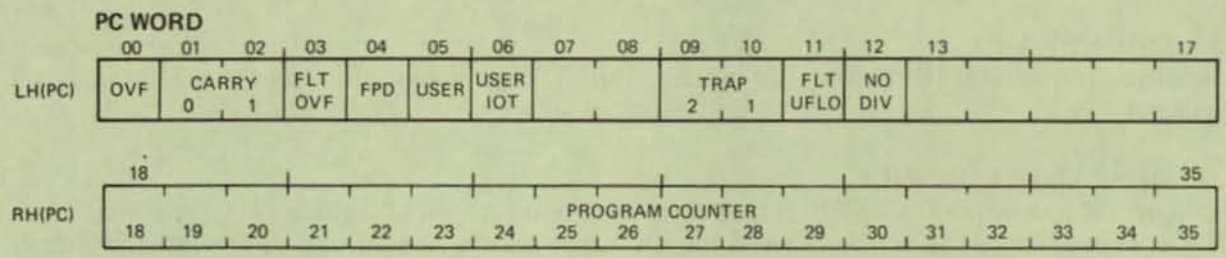
NOTE

The remote diagnosis link operates in one of several modes as explained in Paragraph 4.7. These KLINIK line modes should not be confused with the two 8080 program modes stated above and described below. They are not directly related to each other.

In CTY mode, commands are directed to (and executed by) the 8080 console hardware. An operator may perform the following major functions.

1. Reset and bootstrap system
2. Load and check microcode
3. Deposit and examine memory
4. Read and write I/O device registers
5. Read and write KS10 bus
6. Start and stop CPU clock
7. Single-step the CPU clock
8. Execute a given instruction
9. Halt the machine
10. Start the machine at a given location
11. Single-instruct a program

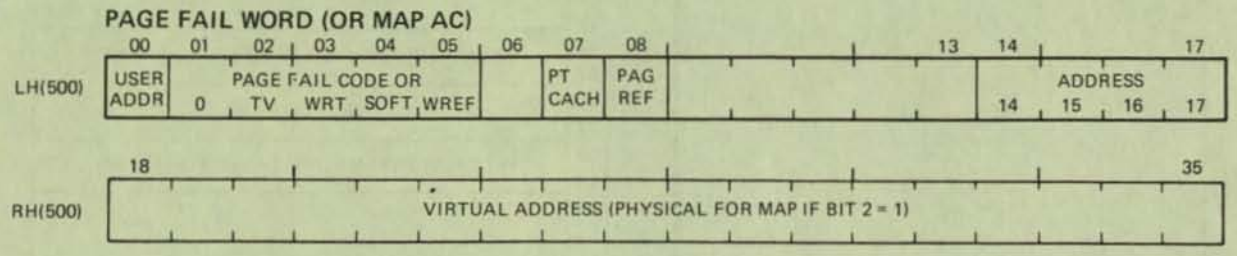
In user mode, the CTY is a user terminal and (with one exception) the console passes all characters directly to and from the program running in the KS10 CPU without echoing or interpreting the characters in any way. The exception is a "control\", which causes the console program to switch the CTY from user mode to CTY mode.



BIT(S)	FUNCTION
0	OVERFLOW
1	CARRY 0
2	CARRY 1
3	FLOATING OVERFLOW
4	FIRST PART DONE
5	USER MODE
6	USER IOT (ALSO PCU)
9	TRAP 2
10	TRAP 1
11	FLOATING UNDERFLOW
12	NO DIVIDE
18-35	PROGRAM COUNTER

MR-0258

Figure 4-33 PC Word



BIT(S)	FUNCTION
0	USER ADDRESS
2-5 (BIT 1 = 0)	
2	TRANSLATION VALID
3	WRITABLE (KL PAGING MODE = 0) WRITTEN (KL PAGING MODE = 1)
4	SOFTWARE (KL PAGING MODE = 0) WRITABLE (KL PAGING MODE = 1)
5	WRITE REFERENCE
2-5 (BIT 1 = 1)	PAGE FAIL CODE
	20 AN I/O INSTRUCTION SELECTED A NONEXISTENT DEVICE OR REGISTER. (BITS 14-35 = I/O ADDRESS)
	25 PAGE TABLE PARITY ERROR
	36 HARD MEMORY ERROR
	37 NXM
7	PAGE TABLE CACHE
8	PAGED REFERENCE
18-35	VIRTUAL ADDRESS (PHYSICAL FOR MAP IF BIT 2 = 1)

MR-0259

Figure 4-34 Page Fail Word

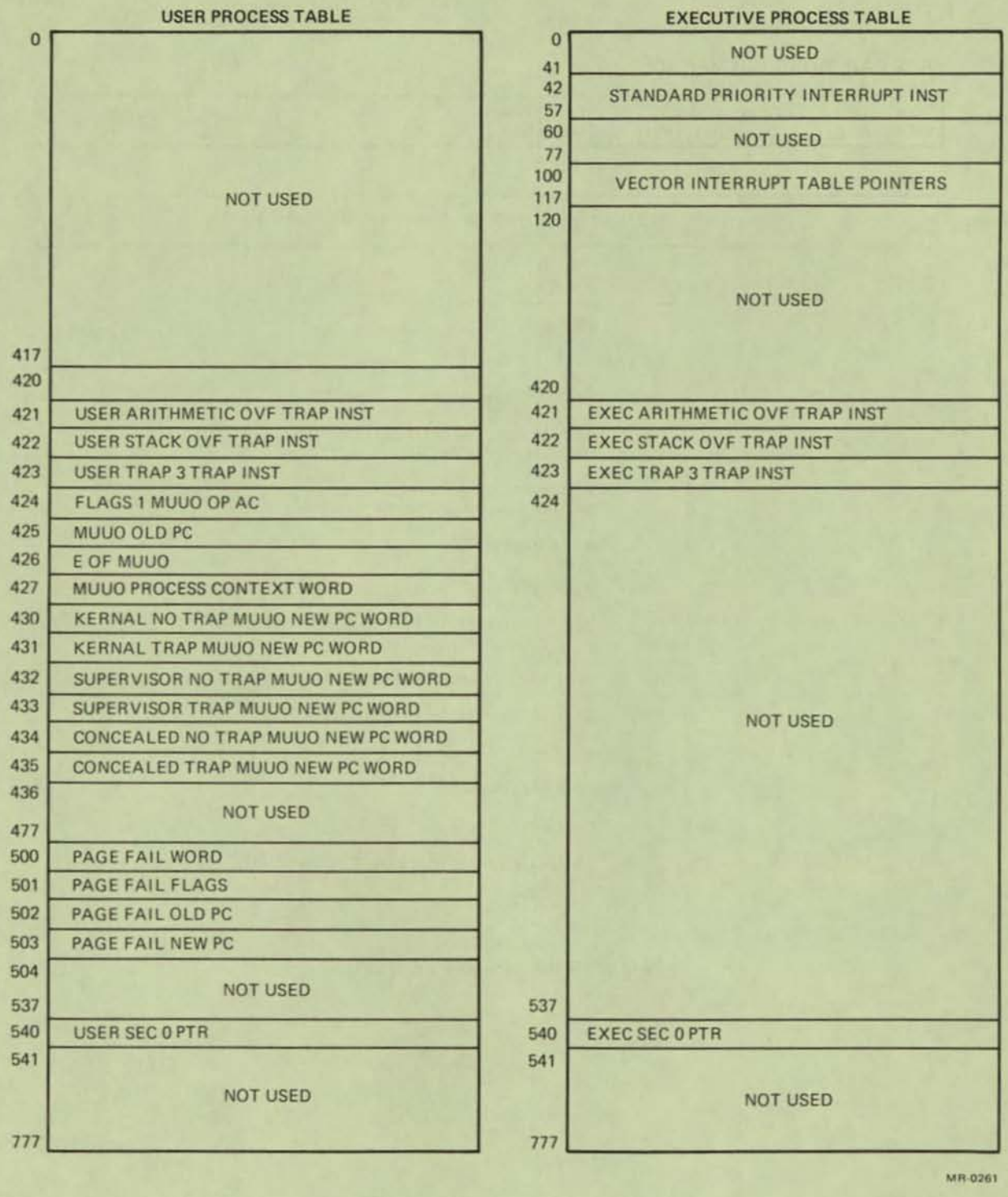


Figure 4-35 KS10 EPT/UPT (TOPS-20 Paging)

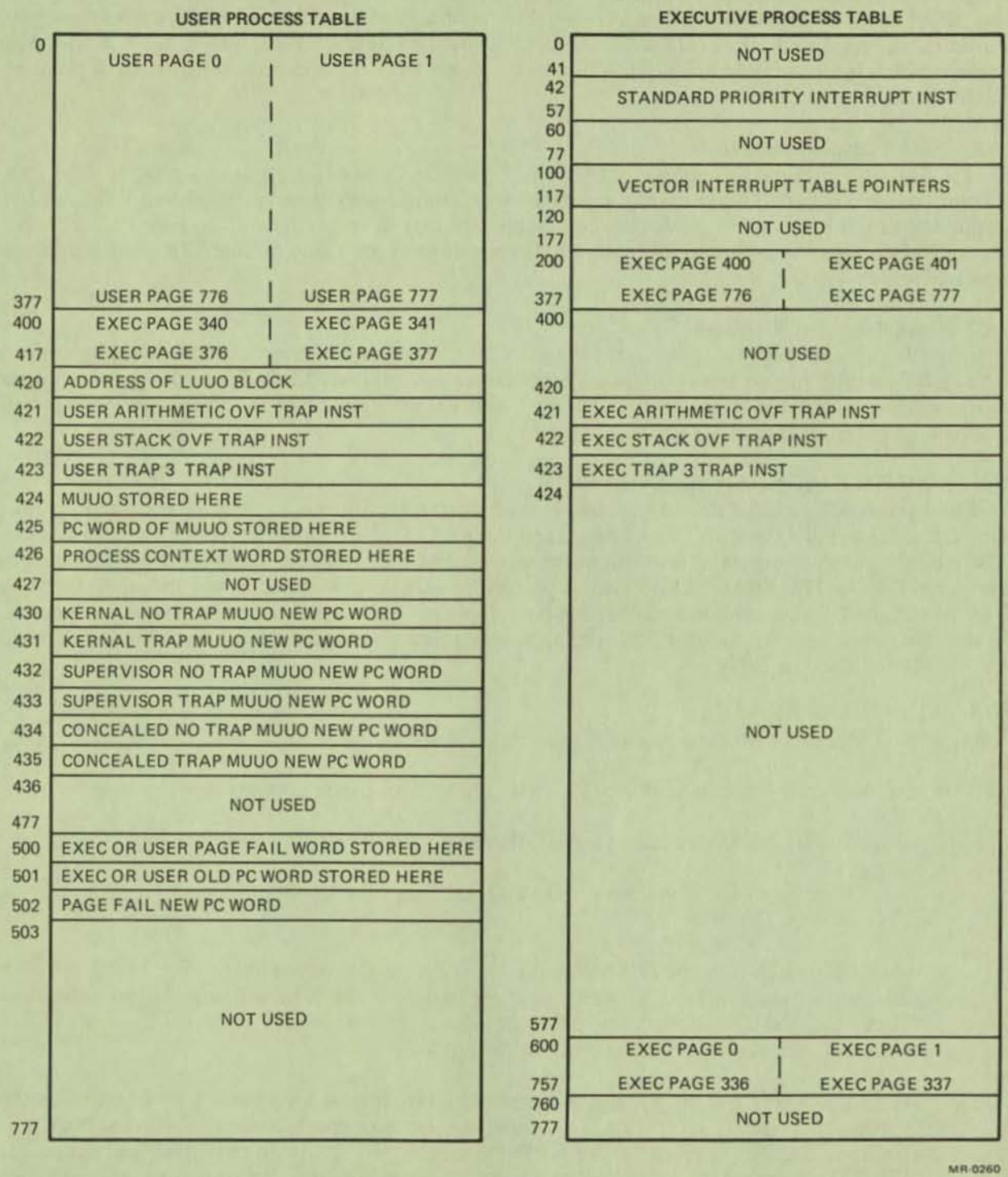


Figure 4-36 KS10 EPT/UPT (TOPS-10 Paging)

The console program initializes to CTY mode at power-up. When in CTY mode, starting or continuing KS10 program execution (ST or CO commands) or a "control-Z" switches the console program to user mode. As stated previously, a "control-\" in user mode causes a return to CTY mode. Also, an error which lights the FAULT indicator causes a return to CTY mode, as does any KS10 processor halt instruction.

4.6.1 8080 Console Commands

The CTY mode command prompt is the characters "KS10" followed by a greater than sign ("KS10>"). A command, or a string of commands separated by commas, may then be typed and followed by a carriage return (CR). The CR causes the command, or string of commands, to be executed. The various console commands are listed in Table 4-8. Error printouts are listed in Table 4-9. Other messages are listed in Table 4-10.

4.6.2 Console Status Registers

The console program reads and prints (at the CTY) the contents of certain 8080 registers in response to the EB (examine bus) command and a system parity error (?PAR ERR). The EB command prints registers 100–103 and 300–303. Registers 100, 303, and 103 are printed when the system parity error is detected. Register bit format is shown in Figure 4-37.

4.7 REMOTE DIAGNOSIS (KLINIK) LINE

As stated previously, a serial line to facilitate remote diagnosis connects to the 8080 console in parallel with the serial line for the CTY. This line, called the KLINIK line, operates under the control of the 8080 console program in one of four operating modes. The mode is dependent on the position of the front panel REMOTE DIAGNOSIS switch (Paragraph 4.1) and whether or not the operator at the CTY has entered a password or enabled/disabled duplicate CTY operation. The password is entered by the PW command; duplicate CTY operation is enabled/disabled by the KL command. Console commands are listed in Table 4-8.

4.7.1 KLINIK Line Modes

The four KLINIK line operating modes (0–3) are as follows.

1. Mode 0 (console unavailable to KLINIK line) – This mode is in effect when:

   a. The front panel switch is set to DISABLE.

   b. The front panel switch is set to PROTECT and the operator has **not** typed a password at the CTY.

2. When the switch is in the DISABLE position or with the switch in the PROTECT position and no password entered, any character entered over the KLINIK line will be echoed as "?NA" (i.e., Not Available). Once the operator types a password at the CTY using the PW command, the KLINIK line is switched to mode 1.

   Mode 1 (console waiting for password on KLINIK line) – This mode is in effect when the front panel switch is set to PROTECT and the operator **has** typed a password at the CTY. The first character entered by the KLINIK user in mode 1 is thrown away and echoed as "PW." Characters following the first are then compared against the password entered by the operator. If there is no match, that is, if the KLINIK user has entered the wrong password, the console responds with the error message ?IL (i.e., Illegal Password). The KLINIK user is allowed three chances to enter the correct password. (The line is hung up after the third consecutive miss.) Once a correct password is entered, the message "OK" is printed and the KLINIK line is switched to mode 2.

Table 4-8 Console Mode Commands

Load Commands

LA xx — Set KS10 memory address xx (0000000–1777777).

LC xx — Set CRAM address xx (0000–3777).

LF xx — Load diagnostic write function xx (0–7). The function specifies a 12-bit group within a CRAM address.

LF	CRAM Bits
0	00–11
1	12–23
2	24–35
3	36–47
4	48–59
5	60–71
6	72–83
7	84–95

LI xx — Set I/O address xx. The address consists of a control number and a register address. I/O addresses accessible from the console are listed below. Note that the address of the console instruction register is not included. If the console attempts to access its own instruction register, no response occurs.

CTL	Register Address	Register(s)
0	100000	Memory status register
1,3	763000–77	UBA paging RAM
1,3	763100	UBA status register
1,3	763101	UBA maintenance register
1,3	7xxxxx	Unibus device registers

LK xx — Set 8080 memory address xx. (PROM address = 00000–17777; RAM address = 20000–21777).

Note that the values loaded by the load commands listed above are addresses for use as arguments by associated deposit/examine commands. The values are **not** the contents of an address.

Deposit Commands

DB xx — Deposit xx (36 bits) onto KS10 bus.

* DC xx — Deposit xx (96 bits) into CRAM. Address previously loaded by LC command.

* DF xx — Deposit xx (12-bit group) into CRAM. Address and diagnostic function previously loaded by LC and LF commands.

DI xx — Deposit xx (16, 18 or 36 bits) into an I/O register. Address previously loaded by LI command.

Table 4-8 Console Mode Commands (Cont)

Deposit Commands (Cont)	
DK xx	Deposit xx (8 bits) into 8080 memory. Address previously loaded by LK command. (Data cannot be deposited in PROM addresses; only in RAM addresses.)
DM xx	Deposit xx (36 bits) into KS10 memory. Address previously loaded by LA command.
DN xx	Deposit xx into next (KS10, 8080, I/O, CRAM) address.
Examine Commands	
EB	Examine KS10 Bus. Prints contents of console registers 100–103 and 300–303 (Paragraph 4.6.2).
* EC	Examine contents of CRAM control register.
* EC xx	Examine contents of CRAM address xx.
* EI	Examine contents of I/O register. Address previously loaded by LI command.
* EI xx	Examine contents of I/O address xx.
* EJ	Examine current CRAM address, next CRAM address, jump address, and subroutine return address.
EK	Examine contents of 8080 memory. Address previously loaded by LK command.
EK xx	Examine contents of 8080 memory address xx.
EM	Examine contents of KS10 memory. Address previously loaded by LA command.
EM xx	Examine contents of KS10 memory address xx.
EN	Examine contents of next (KS10, 8080, I/O, CRAM) address.
Start/Stop Clock Commands	
CH	Halt CPU clock.
* CP	Pulse CPU clock.
* CP xx	Pulse CPU clock xx times.
* CS	Start CPU clock.

Table 4-8 Console Mode Commands (Cont)

Start/Stop Microcode Commands

* PM	Pulse microcode. Performs a CP command to execute a microinstruction followed by an EJ command to print current CRAM address, next CRAM address, jump address, and subroutine return address.
* SM	Reset and start microcode at CRAM address 0.
* SM xx	Reset and start microcode at CRAM address xx.
* TR	Trace. Repeats PM command until any CTY key is depressed.
* TR xx	Trace. Repeats PM command until CRAM address xx is reached or until any CTY key is depressed.

Start/Stop Program Commands

HA	Halt KS10 program. Microcode enters halt loop.
CO	Continue KS10 program execution. Console program enters user mode.
SH	Shut down command. Deposits nonzero data into KS10 memory location 30 to allow orderly shut down of the monitor.
SI	Single instruct. Executes next KS10 instruction.
ST xx	Start KS10 program at address xx. Console program enters user mode.

Select Device Commands

DS	Select disk for bootstrap or microcode verification. Console program asks for UBA number (default = 1), RH11 base address (default = 776700), and disk unit number (default = 0) as follows: >>UBA? 1 <CR> >>RHBASE? 776700 <CR> >>UNIT? 0 <CR> The default value for the RH11 base address is currently the only value permitted. Also, a carriage return in response to any question retains the current value.

Table 4-8 Console Mode Commands (Cont)

Select Device Commands (Cont)	
MS	Select tape for bootstrap or for microcode verification. Console program asks for UBA number (default = 3), RH11 base address (default = 772440), tape unit number (default = 0), tape density (default = 1600 BPI), and slave number (default = 0) as follows: >>UBA? 3 <CR> >>RHBASE? 772440 <CR> >>TCU? 0<CR> >>DENS? 1600 <CR> >>SLV? 0 <CR> The default value for the RH11 base address is currently the only value permitted. Also, a carriage return in response to any question retains the current value.
Boot Commands	
BT	Bootstrap the KS10 from disk. Loads and starts microcode and monitor boot program from drive 0 on UBA1 (default address) or drive selected by last DS command; starts KS10 at memory address 1000. The BT command is performed automatically 15 seconds after power-up. A "control-C" aborts the automatic boot process.
BT 1	Same as BT command except that diagnostic boot program (not monitor boot program) is loaded and started.
BC	Check the KS10 boot path.
LB	Load the monitor boot program from the disk selected last. Does not load microcode. Program must be started at 1000.
LB 1	Same as LB command except that diagnostic boot program (not monitor boot program) is loaded. Program must be started at 1000.
MB	Load the monitor boot program from the tape selected last. Does not load microcode. Program must be started at 1000.
MT	Bootstrap the KS10 from tape. Loads and starts microcode and monitor boot program from tape unit 0, slave unit 0 on UBA3 (default address) or drive selected by last MS command; starts KS10 at memory address 1000.
Verify Microcode Commands	
VD	Verify CRAM against disk. Compares microcode in CRAM with microcode found on disk unit 0 on UBA1 (default address) or disk selected by last DS command.
VT	Verify CRAM against tape. Compares microcode in CRAM with microcode found on tape unit 0, slave unit 0 on UBA3 (default address) or tape selected by last MS command.

Table 4-8 Console Mode Commands (Cont)

Mark/Unmark Microcode Commands

* MK xx	Mark microcode word (set bit 95) at CRAM address xx.
* UM xx	Unmark microcode word (clear bit 95) at CRAM address xx.

Master Reset Command

MR	Master reset. Issue bus reset.

Execute Command

EX xx	Execute the single KS10 systems-level instruction xx.

Enable/Disable Commands

CE xx	Enable (xx = 1) or disable (xx = 0) cache.
PE xx	Enable or disable parity detection as follows:

xx	
0	Disable all parity detection.
4	Enable KS10 bus parity detection.
5	Enable DPE/DPM parity detection.
6	Enable CRA/CRM parity detection.
7	Enable all parity detection.

SC xx	Enable (xx=1) or disable (xx=0) automatic recovery from soft CRAM parity errors.
TE xx	Enable (xx = 1) or disable (xx = 0) CPU interval timer interrupts.
TP xx	Enable (xx = 1) or disable (xx = 0) CPU traps. Following an enable/disable command with a carriage return gives the current value.

Read Cram Commands

* RC	Read CRAM data. Performs diagnostic read functions 0–17 to read CRAM addresses and contents (of current address) as follows:

xx	
0	CRAM bits 00–11
1	Next CRAM address
2	CRAM subroutine return address
3	Current CRAM address
4	CRAM bits 12–23
5	CRAM bits 24–35 (Copy A)
6	CRAM bits 24–35 (Copy B)
7	0s
10	Parity bits A–F
11	KS10 Bus bits 24–35
12	CRAM bits 36–47 (Copy A)
13	CRAM bits 36–47 (Copy B)
14	CRAM bits 48–59
15	CRAM bits 60–71
16	CRAM bits 72–83
17	CRAM bits 84–95

Table 4-8 Console Mode Commands (Cont)

Zero Memory Command

ZM	Zero memory. Deposit 0s into all KS10 memory locations.

Repeat Command

RP	Repeat last command, or last command string, until any CTY key is depressed.
RP xx	Repeat last command, or last command string, xx times.

Lamp Test Command

LT	Blink indicators. Momentarily lights (1–2 seconds) and turns off (1–2 seconds) STATE, FAULT, and REMOTE indicators. The indicators are then returned to their original state.

Password Command

PW xx	Set password xx (xx = maximum of 6 alpha-numeric characters).
	Following a PW command with a carriage return clears the password storage area.

KLINIK Commands

KL xx	Enable remote link with access to system to operate in mode 2 but not in mode 3 (xx = 0). Enable remote link with access to system to operate in mode 2 or in mode 3 (xx = 1).
	Following a KL command with a carriage return gives the current value.
TT	Force KLINIK line from mode 3 to mode 2.

Special Control Characters

control-C	Abort current command. Console returns command prompt.
control-O	Inhibit CTY output (type-outs).
control-S	Inhibit CTY output and stop 8080 console program until control-Q is typed at CTY.
control-Q	Enable CTY output and continue 8080 console program.
control-U	Delete current line.
control-Z	Enter user mode.
control-\	Enter CTY mode. (KLINIK line: enter mode 3.)

Table 4-8 Console Mode Commands (Cont)

Special Control Characters (Cont)

RUBOUT Delete last character.

NOTES

1. **An asterisk (*) indicates that the CPU clock must be stopped in order to execute the command.**
2. **More than one command may be entered on a line (separated by commas) and executed as a command string.**
3. **Commands (except for special control characters) and command strings are followed by a carriage return (CR) to cause command execution. (Special control characters are executed when typed.)**

Table 4-9 8080 Console Error Messages

Message	Meaning
?A/B	A not equal to B. (A and B copies of a microcode field did not match.)
?BC xx	BC command failed. (Refer to KS10 Maintenance Handbook for definition of error code xx.)
?BFO	Buffer overflow. (Too many characters typed; console's 80 character input buffer is full.)
?BN	Bad number. (Character typed is not an octal number.)
?BT xx	BT command failed. (Refer to KS10 Maintenance Handbook for definition of error code xx.)
?BUS	Bad KS10 bus. (All bus lines not zero after power-up or reset.)
?C CYC	Command/address cycle failed. (KS10 bus data failure detected during DB command; good and bad data printed.)
?CHK xx	PROM checksum error. (Bad checksum for PROM chip xx where xx = 1, 2, 3, or 4.)
?D CYC	Data cycle failed. (KS10 bus data failure detected during DB command; good and bad data printed.)

Table 4-9 8080 Console Error Messages (Cont)

Message	Meaning
?DNC	Did not complete. (HA or SM command did not cause microcode to enter halt loop.)
?DNF	Did not finish. (ST, CO, or EX command did not complete.)
?FRC	Forced reload. (Monitor has requested reload; 8080 halts the KS10, reloads the pre-boot program, and starts in KS10 memory location 1000.)
?IA	Illegal address. (Address typed is out of range.)
?IL	Illegal command. (Command typed is not valid.)
?IL	Incorrect password. (Password entered via KLINIK line does not match password entered at CTY.)
?KA	Keep-alive error. (During timesharing, the monitor failed to update the keep-alive count for a period of approximately 15 seconds.)
?MRE	Memory refresh error. (Incomplete KS10 MOS memory cycle. Error occurs when memory must be refreshed in hung state.)
?NA	Not available. (Console not enabled to receive KLINIK line input.)
?NBR	No bus response. (Console did not receive GRANT after requesting KS10 Bus.)
?NDA	No data acknowledge. (Console did not receive DATA CYCLE signal after a data request.)
?NR-SCE	Nonrecoverable soft CRAM error. This message is followed by the standard "?PAR ERR" message.
?NXM	Nonexistent memory. (Deposit or examine command referenced nonexistent KS10 MOS memory location.)
?PAR ERR xx	System parity error. (CPU clock stopped due to system parity; xx = contents of the following console status registers in the order indicated: 100, 303, 103 Refer to Paragraph 4.6.2 for status register bit format.)
?PWL	Password length error. (Password longer than six alphanumeric characters.)
?RA	Requires argument. (Command typed requires an argument.)
?RUNNING	Clock running. (Command typed requires CPU clock to be stopped.)
?UI	Unknown interrupt. (Console received interrupt but CTY or KLINIK line has no character.)
%SCE	Soft CRAM error. 8080 is attempting to recover by reloading the CRAM and continuing the instruction that got the parity error.

Table 4-10 Other 8080 Console Messages

Message	Meaning
BT AUTO	Beginning automatic boot procedure after power-up.
BT SW	Beginning boot procedure as a result of BOOT switch being pressed (LOCK switch in UNLOCK position).
BUS 0-35	Message header for EB command.
CYC	Cycle type for DB command.
ENABLED	Entering CTY mode from user mode. (CTY mode is entered as a result of a "control-\" in user mode with LOCK switch in UNLOCK position.)
HLTD	Halt in KS10 processor program execution.
KS10>	Command prompt.
OFF	Current state is off. (Response to CE, TE, TP, and KL commands when current state of enable is requested and it is a 0.)
ON	Current state is on. (Response to CE, TE, TP, and KL commands when current state of enable is requested and it is a 1.)
RCVD	Data received from bus. (Indicates bus data received if failure occurred during EB command.)
SENT	Data sent to bus. (Indicates bus data transmitted if failure detected during DB command.)
USR MOD	Entering user mode. (User mode is entered as a result of a "control-Z" or the successful completion of a CO, ST, BT, or MT command.)
>>UBA?	Query for UBA number.
>>UNIT?	Query for unit number.
>>TCU	Query for tape controller unit number.
>>RHBASE?	Query for RH11 base register address.
>>DENS?	Query for tape density.
>>SLV?	Query for tape slave number.

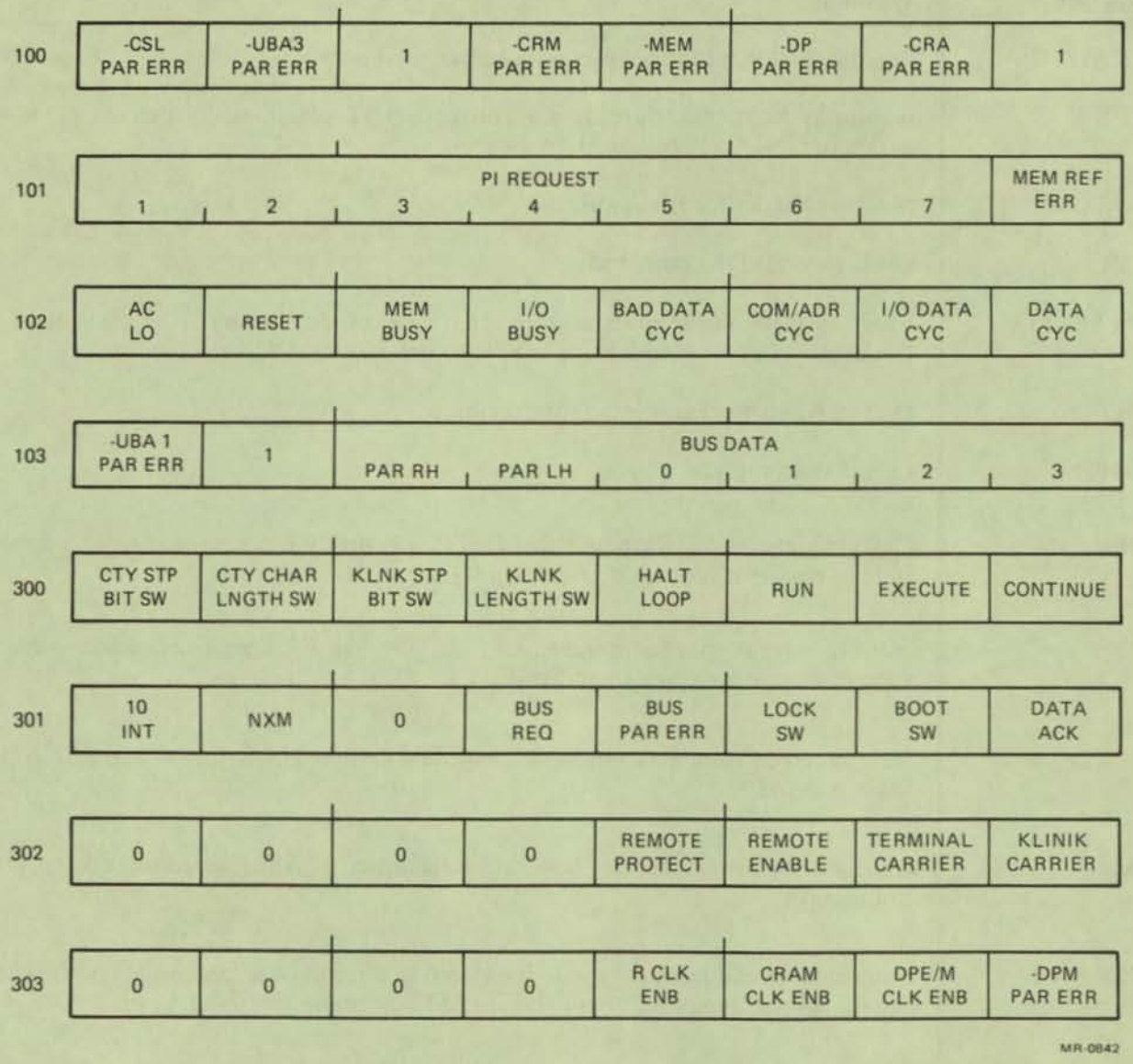


Figure 4-37 Console Status Registers

3. Mode 2 (timesharing user line) – This mode, in which the KLINIK line operates as a timesharing user line, is in effect when:

   a. The front panel switch is set to ENABLE.

   b. The front panel switch is set to PROTECT and the password entered over the KLINIK line matches that typed by the operator.

   Except for a "control-\", all characters entered on the KLINIK line in mode 2 are passed directly to the program running in the KS10 CPU; the characters are not echoed or interpreted in any way by the 8080 console. A "control-\", however, moves the KLINIK line to mode 3 provided the operator has enabled the mode change by typing "KL1" at the CTY.

4. Mode 3 (duplicate CTY) – This mode can only be entered from mode 2 ("control-\" on KLINIK line), and only when entry is enabled by the operator ("KL1" command at the CTY). Once in mode 3, the KLINIK line has expanded capability in that it may perform 8080 console functions (Table 4-8). Characters entered over the line are interpreted exactly like characters typed at the CTY (inputs actually ORed together at the 8080 input buffer), and output from either the 8080 or a running KS10 program go to both the KLINIK line and the CTY. The KLINIK line may be forced back to mode 2 by a "TT" or "KL0" command.

4.7.2 KLINIK Line Operation

To summarize KLINIK line operation, the operator (after determining the need for remote diagnosis) calls the DIGITAL Remote Diagnosis Center giving his number and password. He also switches the REMOTE DIAGNOSIS switch to PROTECT and enters the password using the PW command (KLINIK line switches from mode 0 to mode 1). The Remote Diagnosis Center then responds by entering the correct password on the KLINIK line (line switches from mode 1 to mode 2).

The KLINIK line user now becomes a user on the system. In this mode of operation, the customer may continue to use a degraded but otherwise operational system while the Remote Diagnosis Center runs SYSERR and user mode diagnostics to trouble-shoot the problem. If it becomes necessary to take the system from the customer, or if the system is down, the KLINIK line can become a duplicate CTY by entering a "control-\" after the operator has enabled the mode change (mode 2 to mode 3) with the KL command. The Remote Diagnosis Center then has complete control of the system for trouble-shooting purposes.

CHAPTER 5
TECHNICAL DESCRIPTION

The organizational structure of the KS10 can be viewed as a hierarchy of buses, each of which is a data path shared by a number of different logical elements. The overall system comprises five or more major units or subsystems that communicate with one another over a backpanel bus. Each subsystem in turn is made up of secondary components grouped around one or more secondary buses or data paths. In almost all cases these buses are bidirectional. At the system level, the backpanel bus has a single set of data lines over which any subsystem can communicate with any other, although there are limitations on which subsystems can initiate transactions with which others and on which directions of data flow are available between a given pair of subsystems. In the memory or an in-out subsystem, data flow is bidirectional but occurs between a single controller and any one of a number of storage modules or peripheral devices. The processor, which controls the normal operation of the entire system, is based on a unidirectional data path that is in several parts, with many side loops and with logic elements lying in the various parts of the path.

The first section of this chapter discusses the overall flow of data in the system as a whole and in the processor in particular, and describes those characteristics common to all subsystems, mainly the timing and the manner of communication over the backpanel bus. The remaining sections give detailed descriptions of the bus and the various subsystems, with several sections devoted to the major parts of the processor.

5.1 ORGANIZATION AND TIMING

Figure 5-1 shows the physical arrangement of the KS10 boards in the rack. The slot numbers are at the tops of the columns representing the boards – just below the three-letter board designations. Since the boards correspond rather well to the logical elements that make up the system, the drawing also shows the elementary logical organization of a KS10-based DECSYSTEM-20. The system is made up of a number of subsystems, all of which are connected to a backpanel bus, which handles the movement of data among them. The minimal system has five subsystems: processor, memory, console, and two in-out subsystems, each based on a Unibus. The processor comprises the four boards in slots 11–14, and its connection to the bus is at the DPM board. The memory control board MMC and the memory array boards at its right make up the memory; essentially MMC is the interface between the backpanel bus and the memory array, as it is connected both to that bus and to a memory bus over which all transfers occur between control and memory. Similarly the Unibus adapter board UBA in slot 19 serves as the interface between the backpanel bus and the Unibus to the disks. The second adapter is mounted in slot 16 for interfacing another Unibus, which handles all of the other peripheral devices. There are of course many other interboard connections apart from the backpanel bus, especially among the four boards that make up the processor, and between the console and all other subsystems. The console board provides the connecting link to the operator via a terminal for all operating and diagnostic purposes. Moreover the console board contains the system clock and controls access to the backpanel bus: hence it supplies timing, arbitration, bootstrap and diagnostic signals to all other subsystems, including a myriad of signals to the individual processor boards (only the memory array boards lack direct connections to the console board, as they communicate solely via the memory bus with MMC).

The four boards that make up the processor are organized in two pairs. The data path boards, DPM and DPE, handle the execution of the instructions in the program under control of the microcontroller, which comprises the CRA and CRM boards. DPE contains the full word arithmetic logic, the program flags, the instruction register, and the RAM file containing the fast memory (AC blocks), the virtual cache, and a microcode workspace; DPM contains the 10-bit arithmetic logic for step counting and byte manipulation, the memory addressing logic, the cache directory, and the transceivers that connect to the backpanel bus. CRM is devoted solely to the microcode control RAM, about a third of which is on CRA, which also contains the microcode addressing logic. The DP boards supply many conditions that can be tested by CRA for sequencing the microcode; in particular each instruction code selects a location in a dispatch ROM that, although located on DPE, is part of the microcontroller. The selected dispatch word, together with the AC and index fields of the instruction word, provides CRA with information for dispatching to appropriate control RAM locations for calculating the effective address, fetching the operands, executing the instruction, and storing the result. These activities are carried out on DPM and DPE under control of the microinstruction bits supplied by CRA and CRM. Some bits are also supplied to CSL for handling console functions and to control the clock – its period can be lengthened whenever necessary for the data path operations.

The data path is in three major parts. Contained entirely on DPE is a D bus, which supplies data to the arithmetic logic, the RAM file, and the instruction register. Via DP the output of the arithmetic logic is available to the D bus and to the various elements on the DPM board, including the backpanel bus transceivers for transmission to memory or the I/O equipment. Data from various DPM elements, including words from memory and the DP data with its halves swapped, is available via DBM, which is an input to the mixer for the D bus.

The hardware on each board is shown in a set of circuit schematics, code CS. In every set, the final one, two or even three prints are devoted entirely to power and ground connections, capacitors, spare pins, and sometimes connector layouts and terminators. Each schematic has a number in the form W-X-YZ where W is the letter "M" followed by the board number, X is the revision number of the board, usually 0, and Y is the three-letter mnemonic board designation, such as DPE or CRA. Z is a number or letter indicating the individual drawing in the series; the numbers are used first, followed by letters in alphabetical order when there are more than nine drawings. If a schematic is revised after being signed off by the engineer, a revision letter appears at the right of the drawing number. Throughout the text and in the block diagrams, individual prints are referenced simply by the YZ part of the drawing number.

The YZ designations are also used as prefixes in signal names to show the signal source. The board designations (Y) are used in the names of signals that are generated on the console and supplied to the other boards. Signals for Unibus adapters 1 and 3 are designated by the mnemonics shown in parenthesis in Figure 5-1, namely ADPT0 and ADPT2. The adapter numbers 1 and 3 are actually the subsystem numbers used for addressing the adapters over the backpanel bus. The print set for the memory includes two hardware flowcharts, code FD, drawings M8618-0-MMCF1 and 2.

5.1.1 Processor Logical Organization

The bold line across the center of Figure 5-2 divides the two major functional areas of the processor: the data path above, the microcontroller below. The dashed lines are the boundaries of the individual boards.

The execution of a program is performed by the data path elements under control of the microcontroller. To get an instruction or data from storage, the data path address logic supplies address and command information via the bus transceivers and the backpanel bus to memory. The memory responds by sending the requested word over the bus, and the processor holds it in the memory buffer (MB) contained in the bus transceivers. From MB each word is available via DBM and the D bus to the arithmetic unit. All full word arithmetic and logical operations, including calculation of effective

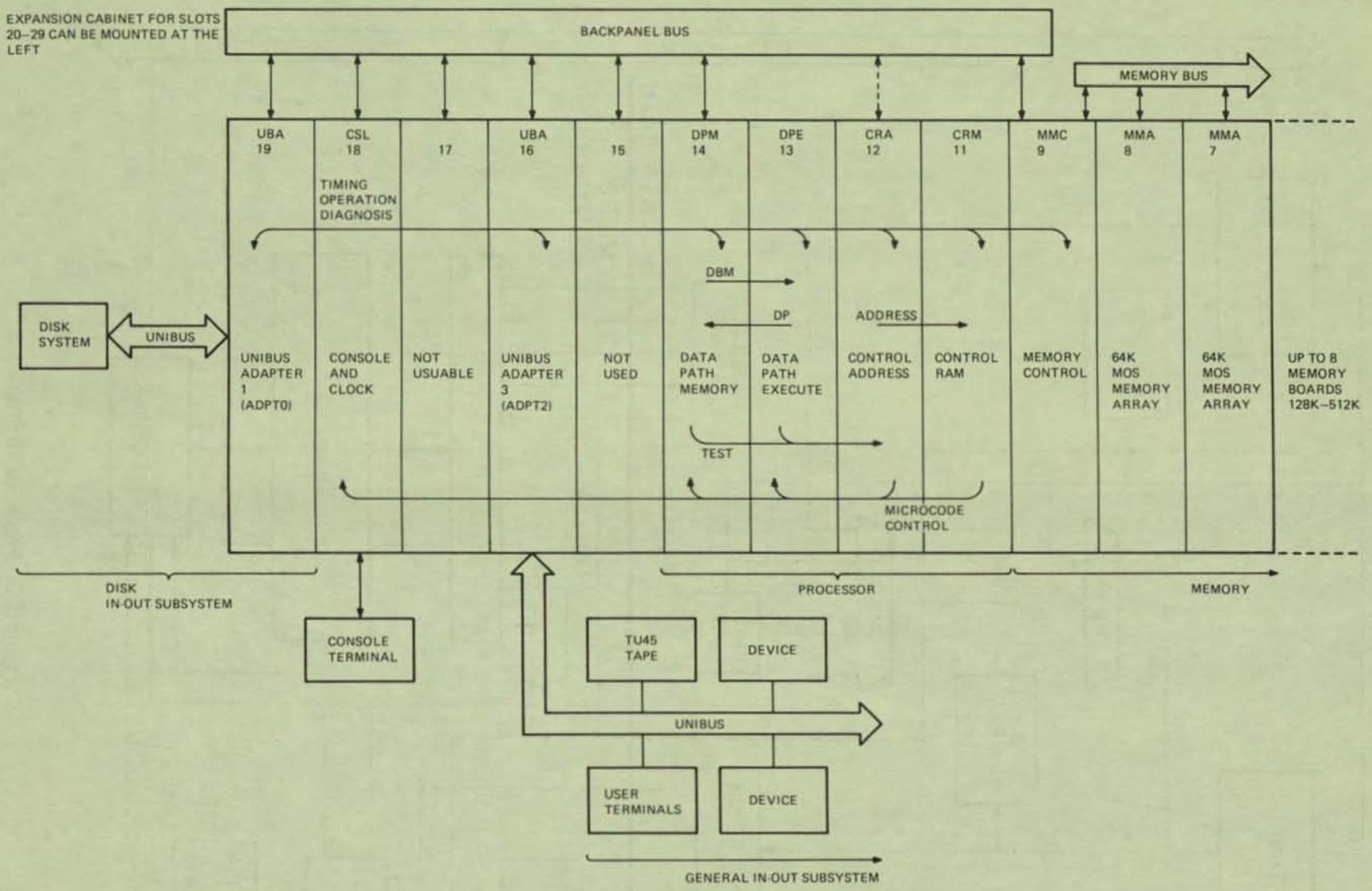


Figure 5-1 DECSYSTEM-2020 Layout

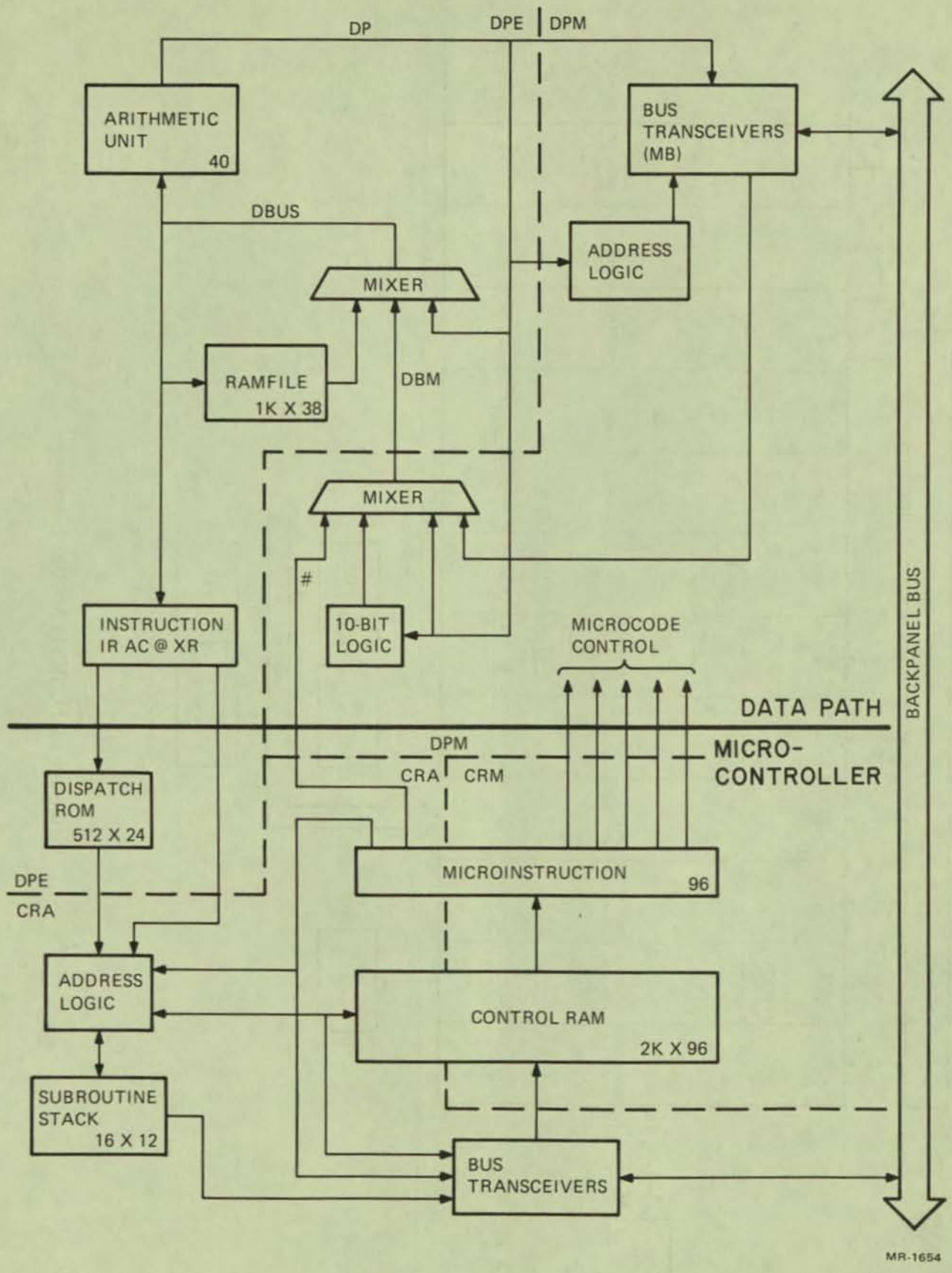


Figure 5-2 Processor Organization

addresses, are performed in this unit, which is based on ten Am2901 microprocessor slices. Together these slices contain not only an adder and a shifter, but an entire register file that holds the arithmetic registers, the program count PC, the effective address, general purpose registers, and various constants and control words. Many operations are entirely internal to the arithmetic unit, using only words from the register file and placing the results in that file. But operations can also combine an external operand with one from the register file, and the result can be kept in the register file or be available elsewhere via the data path DP.

Every word fetched from storage goes to the arithmetic unit, but it is also placed in the cache part of the RAM file, so that the next time it is needed it is available directly to the D bus from the cache without requiring a transfer over the backpanel bus. If a word fetched from memory (storage or cache) is an instruction, its left half is loaded into the instruction register where it is available for execution by the microcontroller. The result of an operation may go from the arithmetic unit via DP and the D bus to an accumulator in the RAM file. But to send a word to memory requires that the arithmetic unit first supply address and command information via the address logic and bus transceivers over the backpanel bus to memory, and then supply the word to be written, again via transceivers and bus. Any word sent to storage is also written via the D bus into the cache so that it is subsequently available without going to storage. The same kinds of bus operations for reading and writing in storage are also used for handling I/O functions, but in the I/O case no data is placed in the cache.

The computational and manipulative capability of the arithmetic unit is expanded by use of the DBM mixer and its associated logic on the DPM board. Via the mixer the number field of a microinstruction word is available to the DPE board, and the mixer can swap the two halves of a word supplied to it via DP. The 10-bit logic includes an adder and two registers, SC and FE. This logic provides for computations on exponents and insertion of bytes in words supplied via DP.

To control its own sequence the microcontroller address logic selects a location in the control RAM, and the contents of that location via the microinstruction register supply information back to the address logic for selecting the next location. This information includes not only an actual address but specification of conditions for skipping or dispatching and commands to call or return from a subroutine, for which the address logic employs a stack.

The two functional parts of the processor work together to perform the program. The 96-bit microinstruction register not only supplies a number field as a quantity to the data path, but its individual bits feed a multitude of control lines that govern operations in all portions of the data path. In the opposite direction the data path supplies various conditions to the microcontroller address logic for skipping and dispatching, but principally it supplies the individual program instruction words, taken from memory for execution by the microcontroller.

Following power turn on, the console boots the system by using some of the backpanel bus data lines, connected to transceivers on the CRA board, to load the microcode into the control RAM in 12-bit segments. Following loading and starting by the console, the microcode initializes the machine, setting up various constants and tables that it stores in a workspace in the RAM file, and it then enters the halt loop in which it can respond to commands from the console. There are various general procedures that the microcode executes, procedures separate from actually executing program instructions, but in all cases these are associated in some way with the running of a program. To execute program instructions, microwords in the microinstruction register control the data path to determine the address for retrieval of the instruction, save that address plus one as the program count in the PC location in the register file, send that address to memory, and upon receipt of the instruction word, place its left half in the instruction register. From this register the AC, index and indirect fields are available to the microcontroller address logic, and the instruction code selects a location in the dispatch ROM, which in turn supplies a dispatch word. Using these quantities the address logic sequences the microcode, through skipping and dispatching, to control the data path to carry out all of the operations necessary to execute the instruction.

Figure 5-3 shows the processor data path in greater detail. The ten 2901s actually form a 40-bit arithmetic unit, with two extra bits at each end. Often these extra bits (in the register file and the adder) are disabled or ignored, but the processor does make use of them at the left end of the adder for sign and overflow considerations. The adder operates on two words supplied by the A and B mixers, which in turn can receive (in various combinations) a word from the D bus, one or two words from the register file (selected by the A and B addresses), zero, or a word from the Q register. The adder output can go to any register in the file, direct or shifted one place left or right. The word in the Q register can also be shifted either way, or it can be replaced by the adder output.

The DPM address logic includes the virtual memory address register VMA, the page table, and the cache directory. Addresses to VMA and mappings to the page table are both supplied via DP. When a virtual reference is to storage, the page table supplies bits 16–26 of the physical memory address and VMA supplies the rest. For a virtual memory write reference, the word on DP is written both in storage and in the cache part of the RAM file, and its address is placed in the cache directory. For a virtual read reference, a match of the address in VMA with the address in the directory indicates the word is in the cache - a "cache hit" - and it is read from there rather than from storage. If a read reference fails to hit the cache, the word read from storage is written in the cache for later use. The cache occupies the top half of the RAM file (512 locations); the AC blocks occupy the bottom 128 locations, and the remaining 384 are available to the microcode for a workspace. Associated with VMA are various flags for specifying the type of transaction for which the backpanel bus will be used (VMA supplies command/address information for I/O instructions as well as for memory access). Both VMA and the flags are available to the data path via DBM. Other logic associated with the DBM mixer provides, besides the functions already mentioned, error and diagnostic information, and a reading of a 10-bit counter that does a 4096 count in each millisecond.

Whenever VMA is loaded from DP, a copy of the right ten bits is kept on the DPE board, for use along with AC, XR and the microword number field in addressing RAM file locations. This copy and associated program flags are directly available to the D bus.

5.1.2 System Timing

The CSL board has an oscillator and divider that generates two 6.66 MHz clock trains, and T and R clocks (transmit and receive), whose relationship is shown by the top two lines in Figure 5-4. These clocks are supplied via the backpanel to the other boards; all boards use the T clock, but the R clock is used only by those connected to the bus. From the basic clock supplied by CSL, each board derives its own clock signals, many of which are gated to produce ticks only when particular conditions are met. The clock is shown on print CSL1, and the clock circuitry for the various boards can be found on CRA2, CRM2, DPE5, DPMA, MMCC, CSL9 (T clock), CSLA (R clock), UBA1 (T clock) and UBA6 (R clock).

On the processor boards the clock is gated to define the processor cycle, which is a set of clock periods used for the execution of a single microinstruction. The cycle is defined by a CSL clock enable that passes a single tick of the T clock to terminate one cycle and begin the next. This is shown in the lower part of Figure 5-4, where the nomenclature applicable to the DPE board is used as an example. The top gate in the clock circuitry at the left on DPE5 passes an entire T clock train, inverted but otherwise unchanged, which is available for operations apart from the actual microinstruction execution. The other gates generate seven DPE5 clock signals, each of which is limited to a single tick of 75 ns duration that occurs while the enable from CSL is low. Similar circuitry on the other boards acts in the same way so as to synchronize all operations on the various boards. Throughout the text the phrase "T clock" refers to signals such as DPE5 T CLK H and any of the other clock signals equivalent to it that are generated on each board. Those particular T clock ticks that determine the processor cycle, namely DPE5 CLK H and equivalent signals, are referred to as the "cycle clock". Almost all operations in the system, including transmission over the bus, are synchronized to a rising edge of the T clock. The R clock is used mostly for latching information received from the bus, but it is also used for

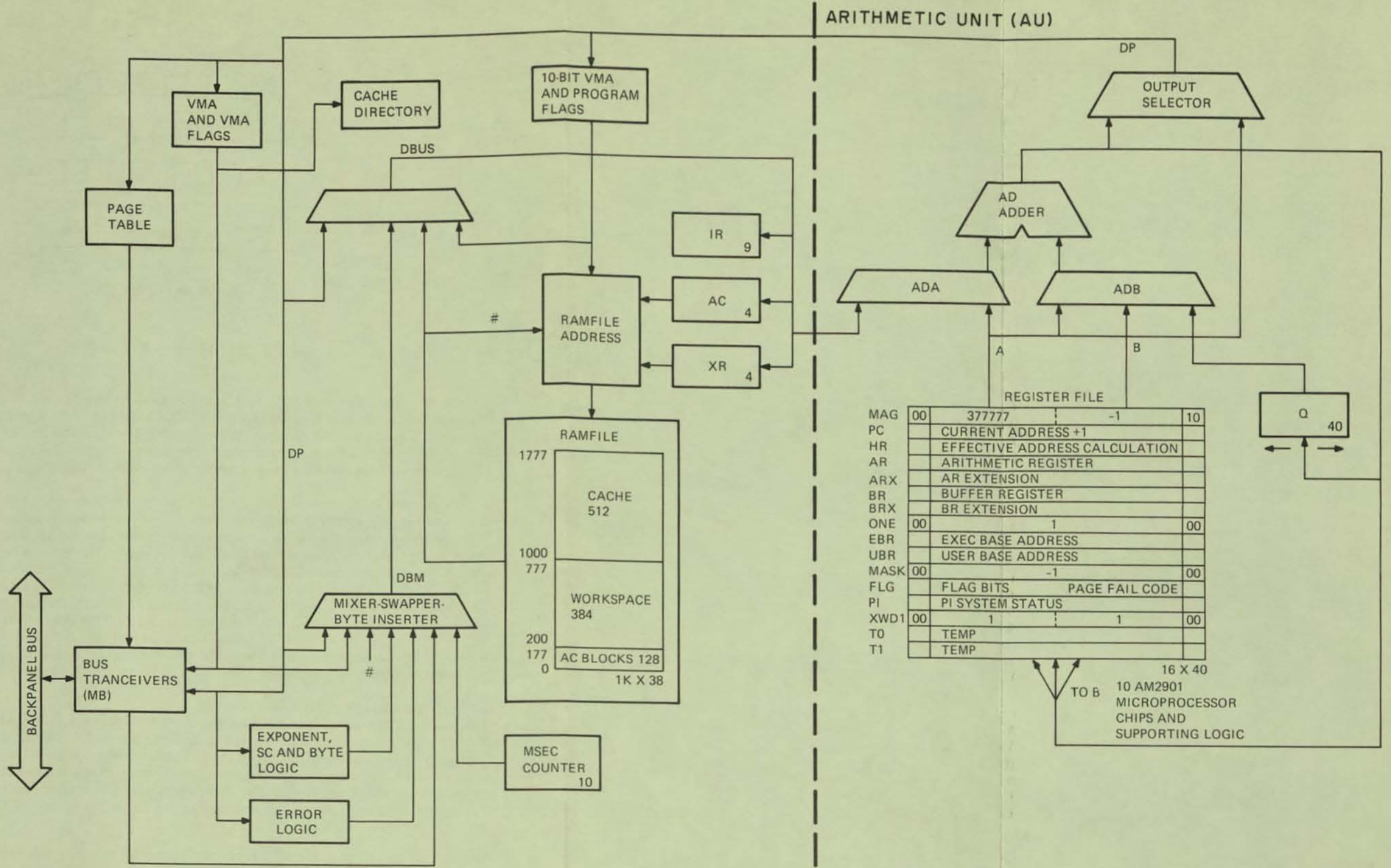


Figure 5-3 Processor Data Flow

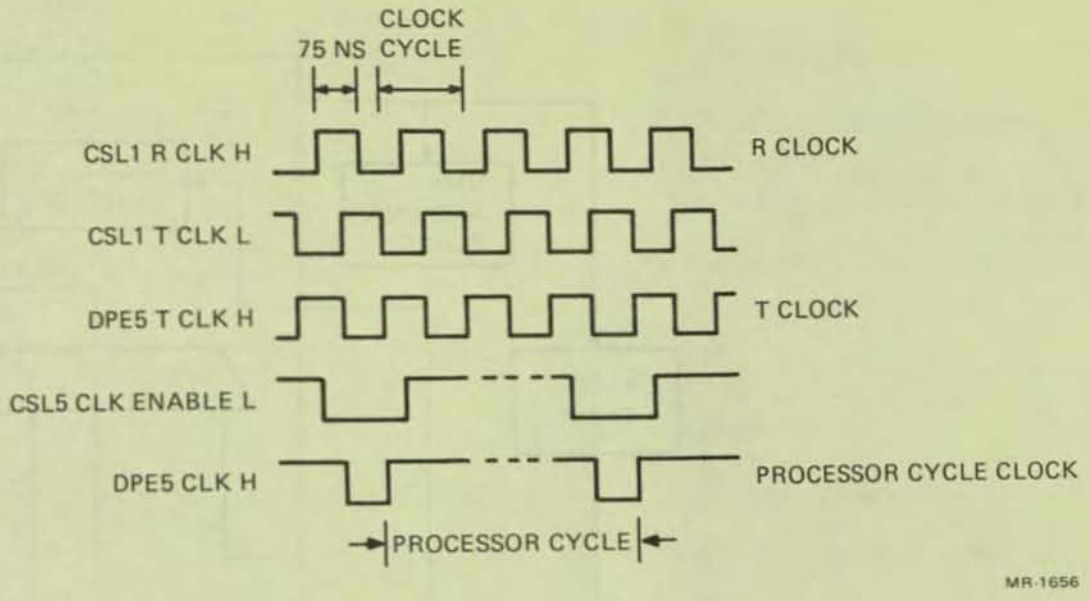


Figure 5-4 Clocks

some timing in memory control and bus arbitration. As can be seen in the circuitry for those boards connected to the bus (such as DPMA and MMCC), certain of the signals that have "T CLK" in their names are actually the OR function of the T and R clocks. These are the signals applied to the transmit clock inputs of the bus transceivers; transmission is triggered at the rising edge of the T clock, but the input must be enabled for three quarters of the clock period.

Except in a fast shift, the processor cycle always encompasses at least two periods of the T clock, but the microcode can increase this by one to three periods whenever additional time is needed, such as when getting a word from the workspace or an accumulator other than AC. The cycle can also be extended by various hardware conditions such as waiting for memory, but in any event the cycle clock always occurs at the end of a processor cycle. In some cases, cycle clocks are gated by conditions besides the enable so that a particular clock signal may not occur at all in a given cycle. For example there are separate gated clocks for the left and right halves of the arithmetic logic, so that operations can be performed on one half of a word without affecting the other. Even the T clock for a board may be gated, as in the case of the CRA board where the first tick in each cycle manipulates the subroutine stack and sets up some skip conditions for possible testing by the microinstruction. In a fast shift, a single microinstruction is repeated at the full T clock rate with the microcontroller clock disabled.

5.1.3 Intrasystem Data Flow

Every subsystem maintains one connection to the backpanel bus. The memory connection is at MMC, the processor connection, at DPM. The dotted line between CRA and the bus in Figure 5-1 does not represent a true bus connection. With the rest of the system quiescent, the console can boot the microcode, and for diagnosing the microcontroller it can read the microcode, various addresses, and the outputs of the CRM parity nets. For these operations the console "borrows" a dozen bus data lines, but no use is made of the arbitration logic or the bus control lines – CSL controls all events directly via other backpanel signals.

Any subsystem except memory can request access to or use of the bus. The memory is limited to responding to memory requests over the bus from other subsystems. A subsystem that gains access to the bus becomes the bus master, and the unit it addresses is the slave. To gain access the unit makes a bus request to the bus arbitrator, which is located on the CSL board (upper right, CSL1). At the completion of the current bus operation (if any), the arbitrator grants the bus to the requesting unit. If

there is more than one request up at the same time, the grant goes to the unit of highest priority in the order console, adapter 1, adapter 3, processor, as determined by priority encoder E103, whose output is decoded to assert the appropriate grant signal. When granting the bus, the arbitrator disables the grant circuit for one cycle: this gives the master a minimum of two bus cycles, as the grant procedure is carried out during the final cycle of any bus transaction (assuming another request is already up). If a master makes a single transfer using only a single data cycle, that transfer may take place in either of the allotted two cycles. Upon receiving a grant, the master typically uses the bus for a command/address cycle, in which the master selects which unit is to be the slave (unless this is predetermined by other circumstances) and supplies the slave with information as to the type of data transfer to be made and the address of the memory location or peripheral device. When the master does execute a command/address cycle, the arbitrator automatically disables the grant circuit for a second cycle, thus giving the same master a minimum of three. Moreover in any bus operation involving memory, the memory itself freezes the arbitrator so that the master gets as many cycles as are needed to complete the entire operation. Data read from memory is always held on the bus for at least two cycles – three when there is an error, whether correctable or not. Other data transfers are completed in a single cycle. The easiest way to understand how the bus works is to consider the various kinds of transfers that can be made between different subsystems.

1. The processor can communicate with memory over the bus for read or write access to the memory array. Upon receipt of the command/address cycle, MMC puts up a memory busy signal that holds the bus until the operation is complete. The data transfer occurs when the read data has been retrieved from the array, or the microcode makes the write data available to the bus. In the write case the data is usually sent right after the command/address cycle.

2. An adapter can gain read or write access to memory as a nonprocessor request in the same way the processor makes a data access to memory (in the write case there is never a wait – the adapter always has the data ready). Moreover an adapter can also gain read-pause-write access, for inserting a byte into a memory word. In this case the memory holds the bus busy until two data transfers have occurred.

3. In response to a PI request, the processor does a command/address cycle that specifies an interrupt level and asks for the identification of any subsystem (adapter) that is requesting an interrupt on that level. The appropriate adapters identify themselves during the following data cycle.

4. The processor can select a specific adapter either to get an interrupt vector (from one that has requested an interrupt) or to do an I/O instruction. For an output instruction the data from the processor follows immediately, freeing the bus, but an I/O busy signal from the adapter causes the processor (microcode) to wait until the data has been sent on to the addressed device. For an interrupt vector or an input instruction, the cycles available for data are thrown away, and the bus is then free for use by other subsystems while the I/O busy signal causes the processor to wait for the desired information.

5. An adapter can use the bus for a single I/O data cycle in which it sends an interrupt vector or data to the processor in response to a previous request.

6. The processor can do an I/O instruction to memory to read or write memory status. Although the data cycle always occurs within the three allotted the master, the memory asserts I/O busy on a status write and does the read status as an I/O data cycle, because the procedure at the processor is handled by the same microcode that does I/O instructions to the adapters.

7. The processor can do an I/O instruction to read the console register. The data is returned immediately in an I/O data cycle.

8. Via the bus the console can communicate with any other subsystem: it can gain data access to memory for an examine or deposit; it can do an I/O instruction to memory or an adapter (which in the latter case may elicit a later I/O data cycle); and it can order the processor to get a word from the console register (via the bus) and execute it as an instruction.

A detailed description of KS10 bus operation follows.

5.2 KS10 (BACKPLANE) Bus

The KS10 bus is a synchronous backplane bus internal to the KS10 processor that provides a control and data path between the console, CPU, memory, and I/O controllers. (The only I/O controllers currently on the bus are the two UBAs.) The bus can accommodate another I/O controller (to allow for future expansion) and it performs the following major functions.

1. Memory Data Transfer - Transfers data to/from MOS memory via the memory controller under the control of the CPU, console, or a UBA (NPR data transfers).

2. I/O Register Data Transfer - Transfers data to/from I/O device registers under control of the CPU or console. An I/O device is considered any device external to the CPU. Thus, not only are the Unibus devices connected to a UBA considered to be I/O devices, but also the UBA itself as well as the memory (controller) and console.

3. PI Handling - Transmits PI requests generated by the UBAs and transfers the interrupting controller (UBA) numbers and interrupt vectors from the UBAs to the CPU under control of the CPU.

4. System Synchronization - Provides a continuous clock train that is used by all bus devices to sequence logic and to synchronize operation with the rest of the system.

5. System Reset and Power Fail Indicator - Allows the console to reset the system and to signal ac power failure to the devices on the bus.

The KS10 bus data path is 36 bits wide. There are two parity bits associated with the data lines, one for data lines 0–17 and one for data lines 18–35. Each device checks for correct (even) parity when it receives information over the bus. If bad (odd) parity is detected, the CPU clock is stopped.

The number of control lines on the bus is minimized in that command/address information is transmitted over the data lines in addition to memory and I/O register data. For example, if a device is to write memory, it asserts command bits and the memory address on the data lines for one bus cycle. This cycle is called a command/address cycle. Then, during a following bus cycle, it transmits the 36-bit data word to be written in memory. This cycle is called a data cycle.

Before any device can transfer information over the KS10 bus, it must first request and then be granted the bus. There is a bus request line and a corresponding grant line for each device. The bus arbitrator, located on the console module, monitors all requests, resolves request priority, and (whenever the bus is free) grants the bus by asserting the grant line for the highest priority device.

KS10 bus signals and information flow are shown in Figure 5-5. Bus signals are terminated at both ends of the wire run (Z = 120 ohms). The majority of signals are terminated at the console module on one end and at the memory controller on the other end. Bus logic levels are as follows.

Logic Level	Voltage
0	+3.4 V
1	0 V to +0.8 V

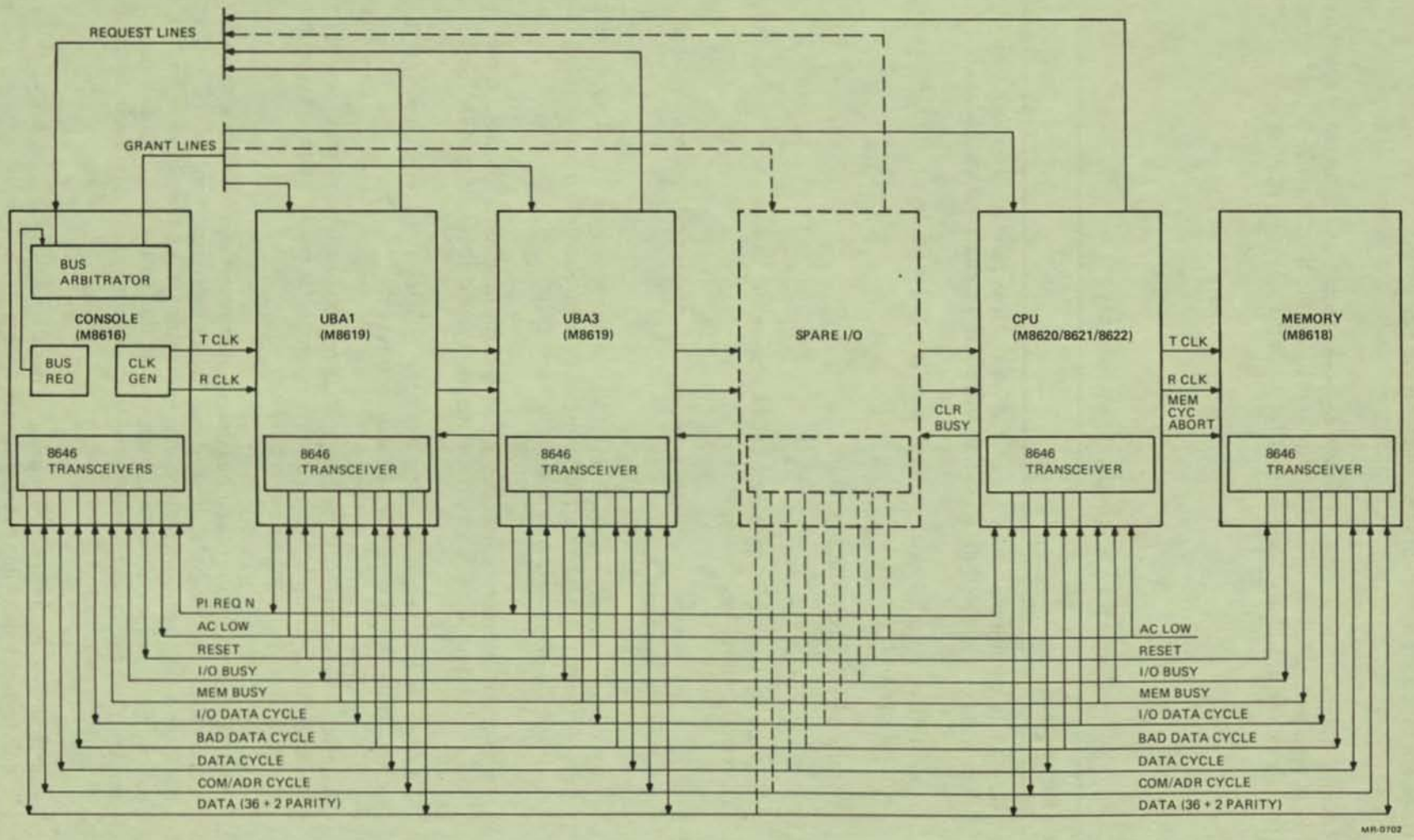


Figure 5-5 KS10 (Backplane) Bus

Table 5-1 summarizes the functions of the various signals on the KS10 bus.

5.2.1 Bus Timing

T CLK and R CLK are system clocks generated on the console module and distributed to all devices on the bus. T CLK is used to transmit data and control signals on the bus; R CLK is used to receive data and control signals on the bus. Both clocks have a 150 ns period. Timing relationship is shown in Figure 5-6.

Table 5-1 KS10 Bus Signal Summary

Signal	Description
T CLK	6.66 MHz continuous clock generated on console module. Leading edge defines start of bus cycle. Used to clock data and control signals transmitted on bus.
R CLK	6.66 MHz continuous clock generated on console module. Used to latch data and control signals received on bus.
REQUEST (one per device)	Asserted by device requesting bus.
GRANT (one per device)	Asserted by bus arbitrator when device requesting bus has been granted the bus. (Device becomes bus master.)
COM/ADR CYCLE	Asserted by bus master when transmitting command/address on data lines. Asserted for one bus cycle.
DATA CYCLE	Asserted by bus master when transmitting memory write data or I/O register write data on the data lines. Asserted by memory controller when transmitting memory read data on data lines. Asserted for one bus cycle.
BAD DATA CYCLE	Asserted by memory controller when transmitting uncorrectable memory read data on the data lines. Asserted for one bus cycle coincident with DATA CYCLE.
I/O DATA CYCLE	Asserted by bus device when transmitting I/O register read data or an interrupt vector on the data lines. Asserted for one bus cycle.
MEM BUSY	Asserted by the memory controller after receiving memory read, write, or read-pause-write command. Negated when memory is ready to accept another command. This signal disables bus arbitrator.
I/O BUSY	Always asserted by addressed bus device after receiving an I/O register write command. Also asserted by addressed bus device after receiving an I/O register read command (or a read interrupt vector command) when the device is NOT going to supply the register read data (or the vector) during the bus cycles allotted the bus master. (The bus device requests the bus and generates a data cycle to transfer the data at a later time.)
CLR BUSY	Asserted by CPU (after a time-out) to negate I/O BUSY in UBA after a nonexistent device register has been referenced.

Table 5-1 KS10 Bus Signal Summary (Cont)

Signal	Description
MEM CYC ABORT	Asserted by CPU to terminate memory reference (cache hit or AC reference).
PI REQ n (n = 1–7)	Asserted by UBAs to request CPU priority interrupt on channel n.
DATA 00–35	Bidirectional data lines used to transfer command/address and read/write data between devices on the bus.
PARITY LEFT	Transfers computed (even) parity for data lines 00–17.
PARITY RIGHT	Transfers computed (even) parity for data lines 18–35.
RESET	Generated by CSL to clear all I/O devices on the bus. Triggered by front panel RESET switch. Also occurs automatically 2 ms after AC LO.
AC LO	Generated by CSL when ac power is failing (power failure detected by H7130 power supply).

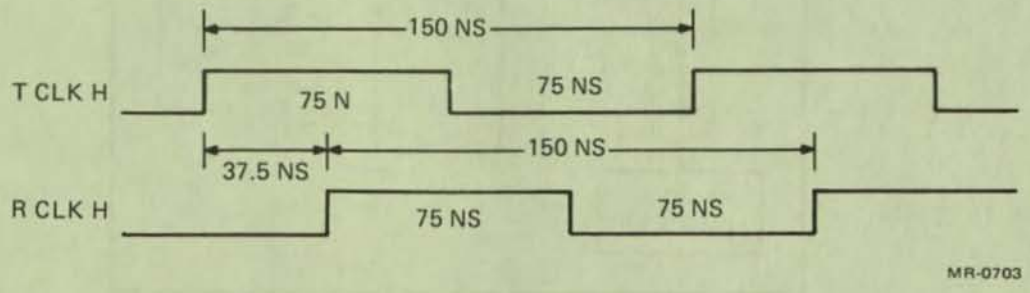


Figure 5-6 T CLK/R CLK Timing Diagram

5.2.2 8646 Bus Transceiver

System modules connecting to the KS10 bus use 8646 transceiver latch circuits to transmit and receive information on the data lines and a majority of the control lines. Each 8646 can transmit and receive four bus signals. In addition, the circuit determines parity for both input and output data.

A circuit schematic for the 8646 is shown in Figure 5-7. In KS10 devices, T CLK (ORed with R CLK) connects to the TRN CLK input, and R CLK (and sometimes additional latching logic) connects to the REC LATCH input. If the TRN ENABLE input is true (low), input data is clocked by the TRN CLK input into four D-type flip-flops and asserted on the bus. This data is asserted until the next TRN CLK input (150 ns later) when the flip-flops are clocked again. To negate all four transmitted outputs, the TRN ENABLE input is made false (high) causing the next TRN CLK to load 0s in the flip-flops. Bus signals not transmitted by a device (only received) have the corresponding 8646 inputs permanently false (wired to ground).

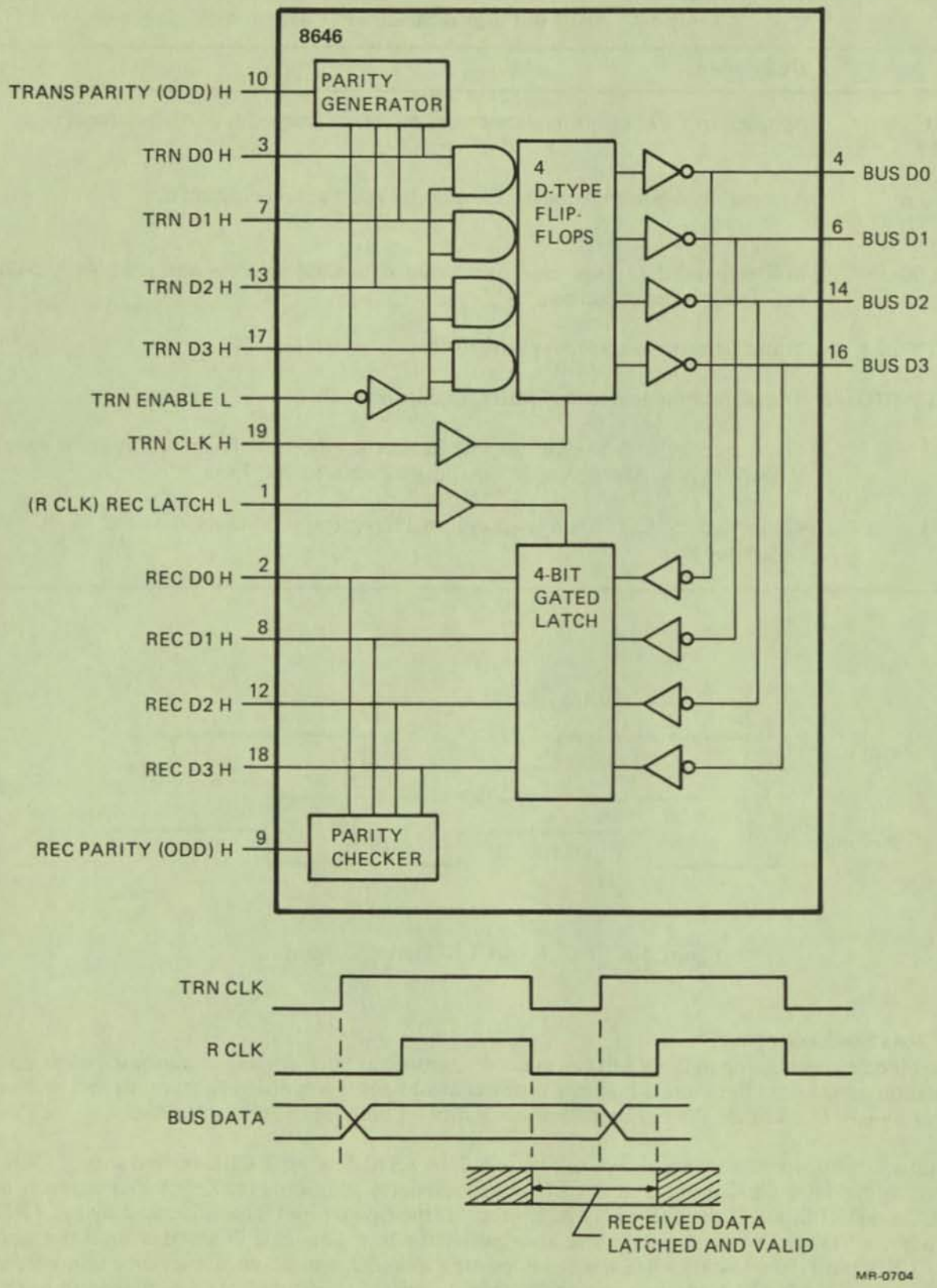


Figure 5-7 8646 Bus Transceiver

The 8646 uses a 4-bit latch circuit to receive and buffer information on the bus. When the REC LATCH input (usually R CLK) is false (high), the latches remain open and the data currently on the bus at the latch inputs is asserted at the 8646 data output pins. When R CLK drives the REC LATCH input true (low), the bus data is latched and the data output pins will not change for the duration of the clock (75ns). During this time, the latched data may be gated and clocked by T CLK and the next R CLK in the bus device, even though bus data is changing at the latch inputs.

Bus transceivers that connect to the KS10 bus data lines utilize the internal parity generator and parity checker. Little additional logic is required in a device to generate the two bus parity bits (PARITY LEFT and PARITY RIGHT) transmitted on the bus, or to check the parity of the entire 36 bits of data received on the bus.

5.2.3 Bus Arbitration

A device may request the bus at any time by asserting its REQUEST line at the leading edge of T CLK, the start of a bus cycle. The bus arbitrator on the console module will then grant the requesting device the bus by asserting the device GRANT line whenever the bus is free and if a higher priority device is not also requesting the bus. Bus priority, highest to lowest, is as follows.

1. Console
2. UBA1
3. UBA3
4. CPU

NOTE

The memory controller does not make bus requests.

Assuming there are no higher priority requests and the bus is not already being used by another device, the GRANT signal will be asserted by the arbitrator during the same bus cycle that REQUEST is asserted. When GRANT is received by a device, the device negates REQUEST (at leading edge of T CLK) and assumes control of the bus as bus master. With reference to Figure 5-8, the device then has the bus for the next two cycles, three cycles, or an unspecified number of cycles depending on what action it takes during the first cycle. A bus master may do the following, each affecting the arbitrator in a different way.

1. No Command/Address Cycle – Bus arbitrator grants bus for **two** cycles. Actions taken by the bus master for which the first cycle is not a command/address cycle are as follows:

   a. Data Cycle – I/O device requests and is granted the bus and then initiates a data cycle to send data to another device to complete an I/O register read operation or an interrupt vector read operation.

   b. No Action – Device requests and is granted the bus and then does not use the bus. Not using granted cycles is equivalent to giving up the bus; another bus request must be made to become bus master.

2. Command/Address Cycle (I/O Operation) – The arbitrator monitors the COM/ADR CYCLE bus signal and grants the bus master **three** bus cycles (an extra cycle) if the first cycle is a command/address cycle. When the command/address specifies an I/O operation, no additional bus signal will disable the arbitrator. All I/O operations, except those named above in 1(a) which require another bus request to transfer data, complete within the three allotted cycles.

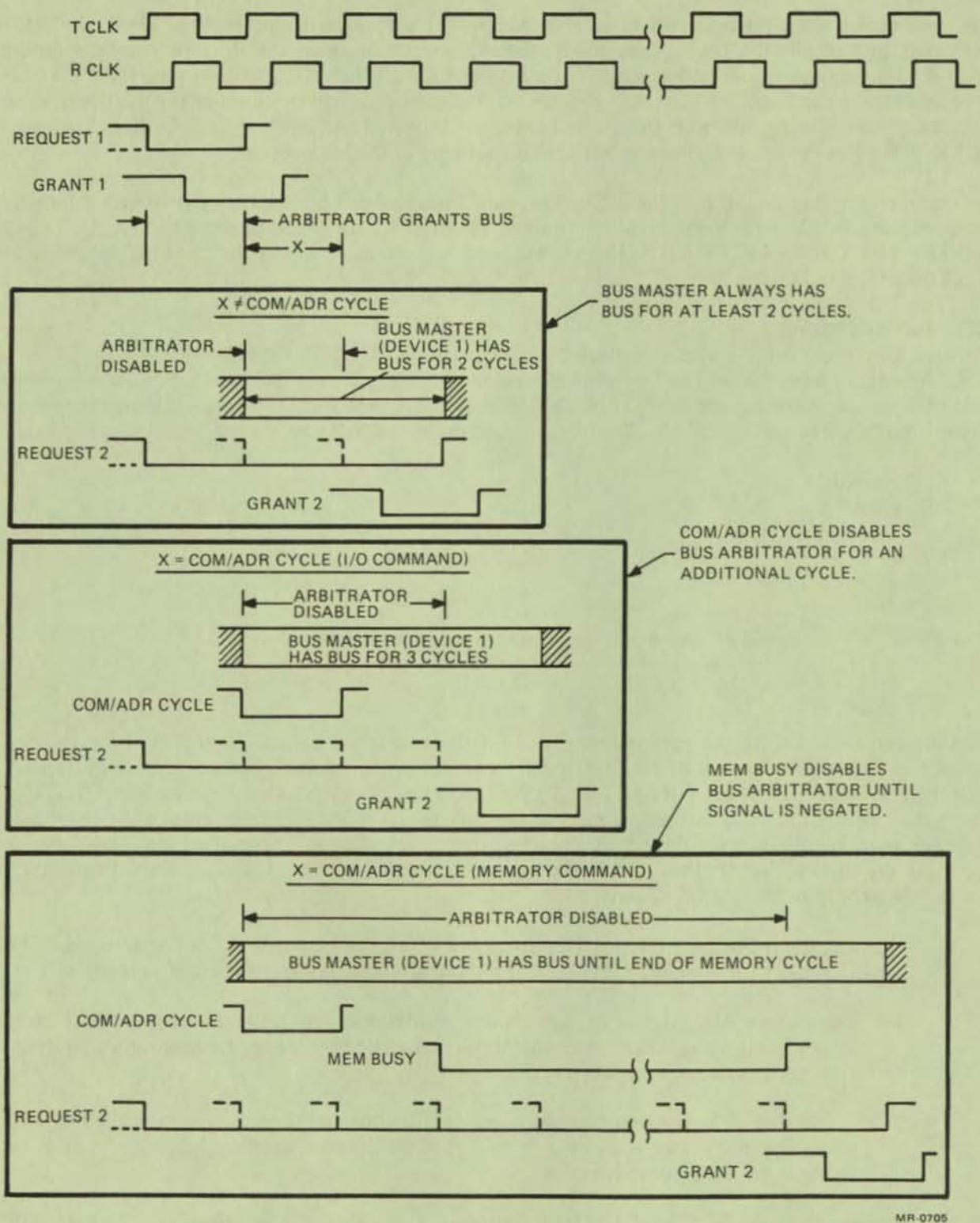


Figure 5-8 Request/Grant, Bus Timing Diagram

3. Command/Address Cycle (Memory Operation) - As for a command/address cycle specifying an I/O operation, the COM/ADR CYCLE signal causes the arbitrator to grant the bus master three cycles. However, when a memory operation is specified, the arbitrator is disabled during the third bus cycle by bus signal MEM BUSY. This signal is asserted by the memory controller in response to the command, and it is negated when the memory controller is ready to accept another command. Thus, the bus master has the bus for an **unspecified** number of cycles; that is, for the duration of the memory operation.

The operation of the bus arbitrator may be summarized as follows.

1. The bus master is always granted the bus at least two bus cycles.

2. If the first cycle is a command/address cycle, the bus master is granted the bus at least three cycles.

3. If the bus master initiates a memory operation, it is granted the bus until the operation completes.

5.2.4 Bus Usage

As discussed in Paragraph 5.2.3, a device may do the following after it has been granted the bus:

1. Not use the bus.
2. Initiate a data cycle.
3. Initiate a command/address cycle.

For the current KS10 configuration and barring a malfunction, the only device that does not use the bus after a bus request is made is the CPU. To save time, the CPU always requests the bus for every memory reference. Then, if there is a cache hit or if the reference is to an AC, MOS memory need not be referenced and the CPU initiates no bus action.

NOTE

In some cases, a command/address cycle may be generated before a cache hit is detected. The CPU then asserts MEM CYC ABORT to terminate MOS memory operation.

The only device that does a data cycle after becoming bus master (without first performing a command/address cycle) is a UBA. The data cycle actually completes a previously initiated I/O register read operation by the CPU or console, or an interrupt vector read operation by the CPU. When first addressed, a UBA does not furnish register data or an interrupt vector within the three bus cycles allotted the device initiating the operation. Instead, the bus is requested again, this time by the UBA. When the bus is granted, a data cycle is generated to transfer the data.

A device normally uses the bus by first generating a command/address cycle. The command/address cycle, in turn, initiates one of the following eight bus operations.

1. Memory Write
2. Memory Read
3. Memory Read-Pause-Write
4. I/O Register Write
5. I/O Register Write (Byte)
6. I/O Register Read
7. Controller Number Read
8. Interrupt Vector Read

Table 5-2 lists the initiating and responding devices for each operation. For example, the first entry indicates that the console, CPU, and UBAs all write data into memory.

Table 5-2 Bus Operations

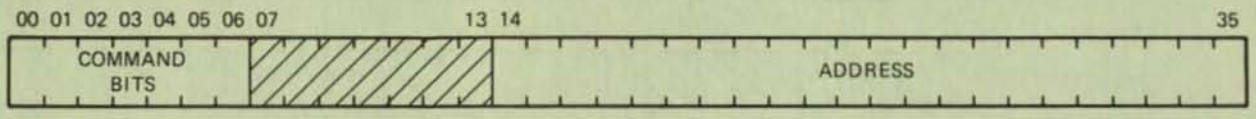
Operation	Initiated By	Directed To
Memory Write	CPU	Memory
	Console	Memory
	UBA	Memory
Memory Read	CPU	Memory
	Console	Memory
	UBA	Memory
Memory Read-Pause-Write	UBA	Memory
I/O Register Write	CPU	UBA
(byte operations directed	CPU	Memory
to UBA only)	Console	UBA
	Console	Memory
I/O Register Read	CPU	Console
	CPU	UBA
	CPU	Memory
	Console	UBA
	Console	Memory
Controller Number Read	CPU	UBA
Interrupt Vector Read	CPU	UBA

5.2.5 Command/Address Cycle

After being granted the bus, the bus master initiates a bus operation by transmitting a command/address on the data lines during the first allotted bus cycle. The bus master also asserts the COM/ADR CYCLE control line. COM/ADR CYCLE is monitored by the bus arbitrator to give the bus master an extra bus cycle (Paragraph 5.2.3). Its principal function, however, is to cause the other devices on the bus to decode the transmitted command/address information. If addressed, a device will then respond to the specified command.

The basic command/address bit format on the data lines is shown in Figure 5-9. Data lines 0–6, the command bits, specify the bus operation to be performed; data lines 14–35 carry the address information specific to the command. The command/address bits are given in Figure 5-10.

The seven command bits (bit 3 is not used) specify the nine different bus operations in the following manner. But 0 determines whether the operation is a memory data transfer or an I/O data transfer; that is, bit 0 = 0 specifies a memory function and bit 0 = 1 specifies an I/O function. Bits 1 and 2, the read and write bits respectively, act in conjunction with bit 0 to further specify the type of operation. For example, if bit 0 = 0 and bit 1 (read) = 1, the operation is a memory read function. All three memory operations (read/write/read-pause-write) and the two I/O register operations (read/write) are specified by these first three commands bits (0–2).



COMMAND BITS	FUNCTION
00 (=1)	I/O FUNCTION
00 (=0)	MEMORY FUNCTION
01	READ (I/O OR MEMORY). BITS 14–35 SPECIFY ADDRESS.
02	WRITE (I/O OR MEMORY). BITS 14–35 SPECIFY ADDRESS.
03	NOT USED.
04	READ INTERRUPTING DEVICE NUMBER. BITS 15–17 SPECIFY PI CHANNEL.
05	READ INTERRUPT VECTOR. BITS 14–17 SPECIFY I/O CONTROLLER.
06	BYTE TRANSFER.
ADDRESS BITS	
14–17	I/O CONTROLLER ADDRESS
18–35	I/O REGISTER ADDRESS
14–35	MEMORY ADDRESS
15–17	PI CHANNEL NUMBER

MR-0706

Figure 5-9 Basic KS10 Command/Address Format

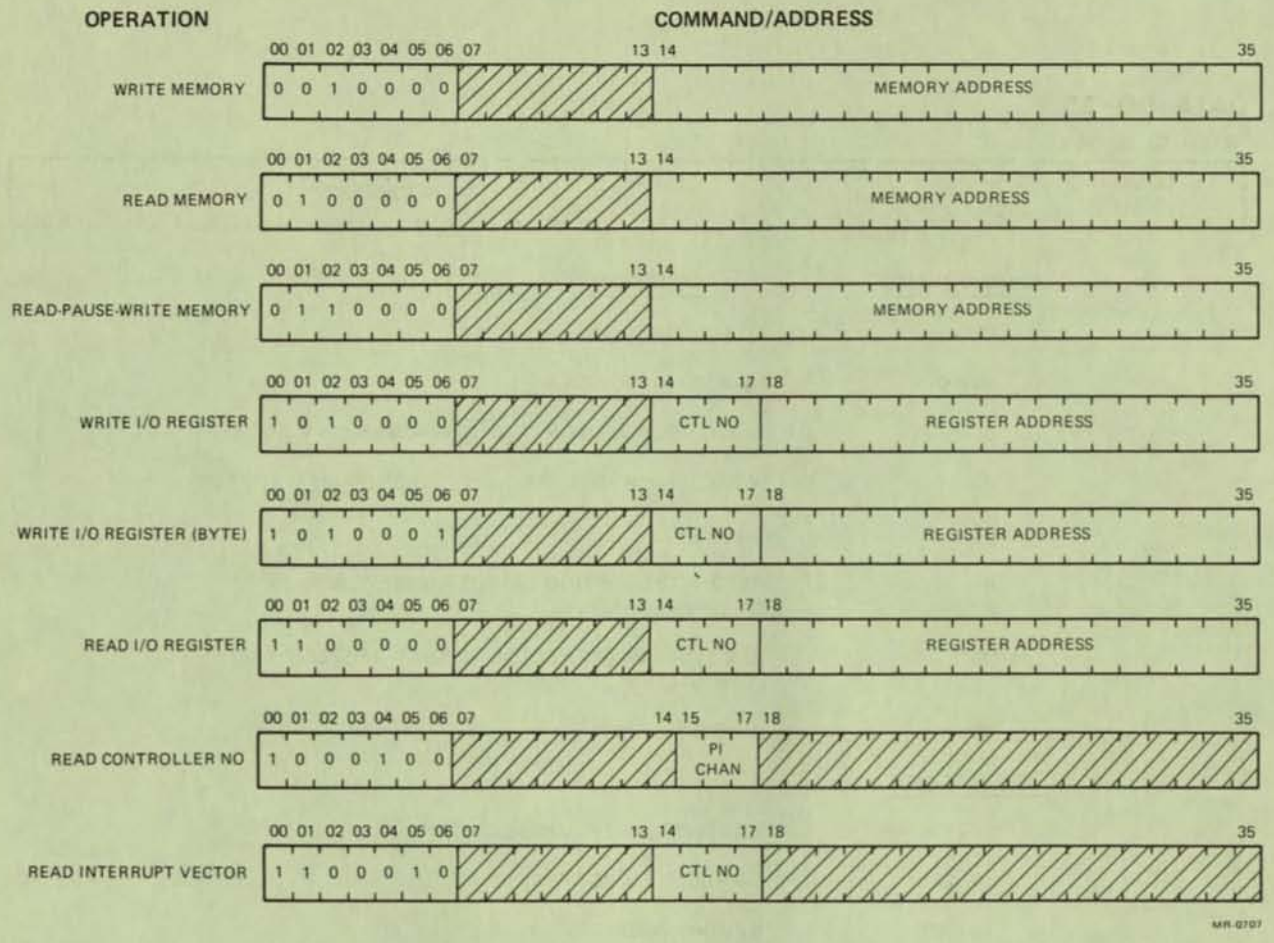


Figure 5-10 Command/Address Bits for KS10 Bus Operations

Bits 4 and 5 are used to specify the two PI operations performed over the KS10 bus. The CPU asserts one of the bits (bit 4 = 1) to read the interrupting controller number. It asserts the other bit (bit 5 = 1) to read the interrupt vector. Bit 0 = 1 for both PI operations. Bit 1 (read) = 1 for the vector read.

Bit 6, the byte transfer bit, has significance only for I/O register write operations that address Unibus device registers. Unibus devices allow full-word (16 bit) or byte (8 bit) transfers of register data and bit 6 is used to specify the transfer mode.

The 22 data lines (14–35) reserved for address information transfer either a memory address (bit 0 = 0) or an I/O address (bit 0 = 1). For memory functions, the least significant 20 bits of the address field are currently used (maximum memory configuration = 512K). For I/O register read/write functions, the I/O address consists of a controller number (bits 14–17) and a register address (bits 18–35). For the PI function that reads the interrupt vector (bit 5 = 1), only the controller number is significant. For the other PI function (bit 4 = 1), which reads the interrupting controller number, the I/O address consists of a 3-bit PI channel number (1–7) on data lines 15–17. The various KS10 I/O controller numbers and register addresses are given in Table 4-6.

5.2.6 Bus Memory Operation

The CPU, console, and UBA all reference MOS memory over the KS10 bus. Once granted the bus, the device making the reference first transmits a command/address on the data lines to specify a memory operation (bit 0 = 0), the memory address (bits 14–35), and the type of memory operation; that is, a read (bit 1 = 1), a write (bit 2 = 1), or a read-pause-write (bit 1 = 1, bit 2 = 1). When the memory controller receives the command address and the address is valid (in-bounds), it asserts MEM BUSY to freeze the bus arbitrator. The device making the reference then has the bus until the end of the memory operation, at which time, MEM BUSY is negated to unlock the arbitrator and allow the next bus operation to take place.

If the operation initiated by the command/address is a memory write operation, write data may be asserted on the data lines during any cycle following the command/address cycle (up to 7.5 μs maximum. The device making the reference initiates the data cycle by asserting the write data and the DATA CYCLE control signal. DATA CYCLE is used by the memory controller to strobe the write data from the bus and to start a memory write cycle. When the write cycle completes and the data has been stored in the MOS array, the memory controller negates MEM BUSY to end the operation. Bus timing for the memory write operation is shown in Figure 5-11.

If the operation is a memory read operation, the memory controller starts a memory read cycle after receiving the command/address. When data is read from the MOS array, it is transmitted on the data lines and the ECC (error correction code) is checked for error. If there is no error, the memory controller initiates a data cycle during the next bus cycle; that is, it continues to assert the data lines and it generates the DATA CYCLE control signal. DATA CYCLE acts as a data strobe (as for the write operation) and it is used by the device initiating the memory reference to gate the read data from the bus. When there is an ECC error, the memory controller attempts to correct the read data and delays the data cycle for one bus cycle; that is, the corrected or uncorrected data is transmitted for the next two cycles and DATA CYCLE is asserted during the second cycle. In either case, error or no error, the read data is on the data lines for one full cycle before DATA CYCLE is generated. This is to allow extra propagation time before the data is gated and clocked in the CPU's 2901 microprocessor circuits. When the data read from the array is uncorrectable, the memory controller flags the data as invalid by asserting the BAD DATA CYCLE control line in addition to DATA CYCLE. Bus timing is shown in Figure 5-12.

During a memory read-pause-write operation, data is first read from the specified address with read data and DATA CYCLE asserted on the bus as previously described for the memory read operation.

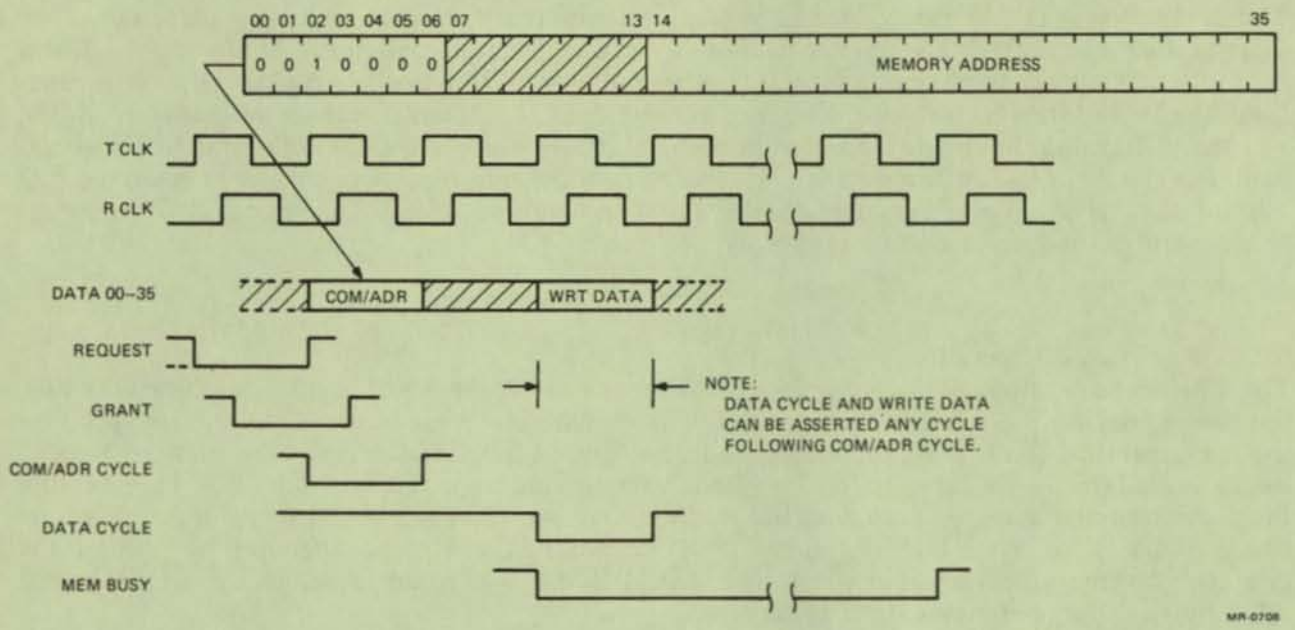


Figure 5-11 Write Memory, Bus Timing Diagram

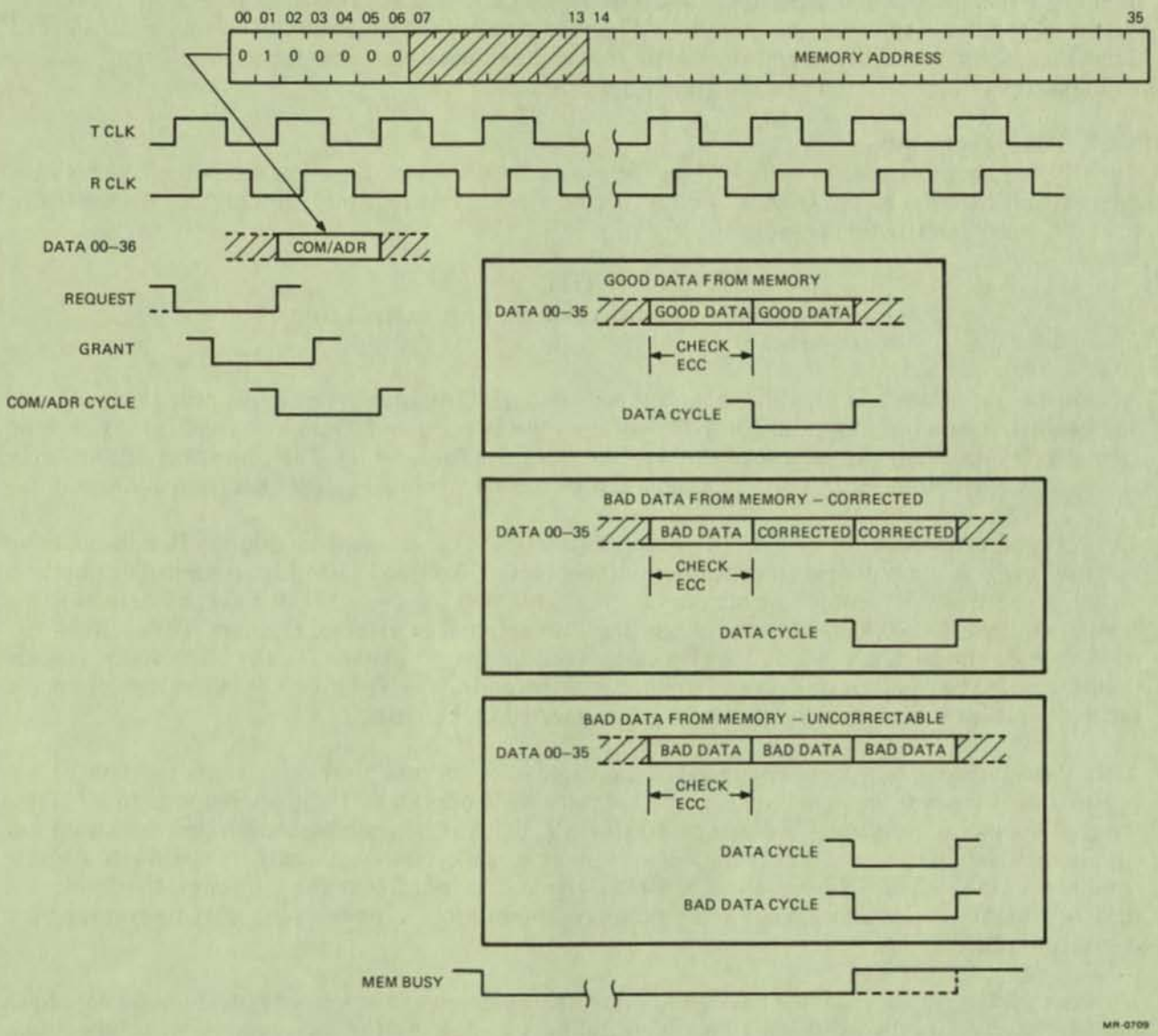


Figure 5-12 Read Memory, Bus Timing Diagram

Then, following the read operation, the memory stays active (MEM BUSY = 1) and performs a memory write cycle when write data and DATA CYCLE are asserted on the bus as previously described for the memory write operation. The read-pause-write operation allows a device to read data from memory, modify it, and then write the modified data back into the same memory address all in one operation. Bus timing is shown in Figure 5-13.

5.2.7 Bus I/O Operation

The CPU and console read or write I/O registers over the KS10 bus. Both can access the I/O registers internal and external to the UBA as well as the memory status registers. In addition, the CPU can read the console instruction register.

NOTE

The console cannot read its own instruction register.

After being granted the bus, the CPU or console accesses an I/O register by first transmitting a command/address on the data lines to specify an I/O operation (bit 0 = 1), an I/O address (bits 14–35), and the type of I/O operation; that is, a read (bit 1 = 1) or a write (bit 2 = 1). The command/address may also specify a byte transfer (bit 6 = 1) when a UBA external (Unibus) register has been addressed.

The I/O address consists of a controller number (bits 14–17) and a register address (bit 18–35). The memory and console both have a controller number = 0. UBA1 and UBA3 have controller numbers 1 and 3 respectively. Except for the memory status register (address = 100000) and console instruction register (address = 200000), all I/O registers are UBA internal or external registers. The internal registers include the 64 UBA paging RAM locations (addresses = 763000-77), the UBA status register (address = 763100) and the UBA maintenance register (address = 763101). The external registers are the addressable registers in the Unibus devices connected to the UBA.

After the addressed bus controller receives the command/address, it always asserts the I/O BUSY control line whenever the operation is an I/O register write operation. If the operation is an I/O register read operation, only the UBA asserts I/O BUSY. (This is because bus controllers which do not supply read data during the requesting device's allotted bus cycles must assert I/O BUSY to flag the condition.) Unlike MEM BUSY, the I/O BUSY signal does not freeze the arbitrator. Consequently, devices initiating I/O register reads and writes have the bus for only two cycles after transmitting the command/address.

During an I/O register write, the device initiating the operation can assert write data on the data lines during either of the two following command/address cycles. As for the memory write operation, DATA CYCLE is also asserted when the write data is transmitted on the bus. DATA CYCLE is used by the addressed controller to strobe the write data from the bus and to store the information in the addressed register. Although the bus data cycle completes the bus operation, storing the write data may take additional time. For example, the UBA must initiate a DATO operation or DATOB operation (command bit 6 = 1) over the Unibus in order to transfer the information to an external register address. Bus timing for the I/O register write operation is shown in Figure 5-14.

During an I/O register read, bus operation differs depending on which controller register is addressed. If the memory or console I/O registers are addressed, read data is asserted on the data lines by the controller two cycles after the command/address is received. This leaves a free cycle between the command/address and data cycles as shown in the upper part of Figure 5-15. If a UBA internal or external register is addressed, read data is not asserted on the bus during the bus cycles allotted to the device initiating the operation. Instead, as shown in the lower part of Figure 5-15, the UBA requests the bus at some later time and transmits the read data whenever the bus is granted. The

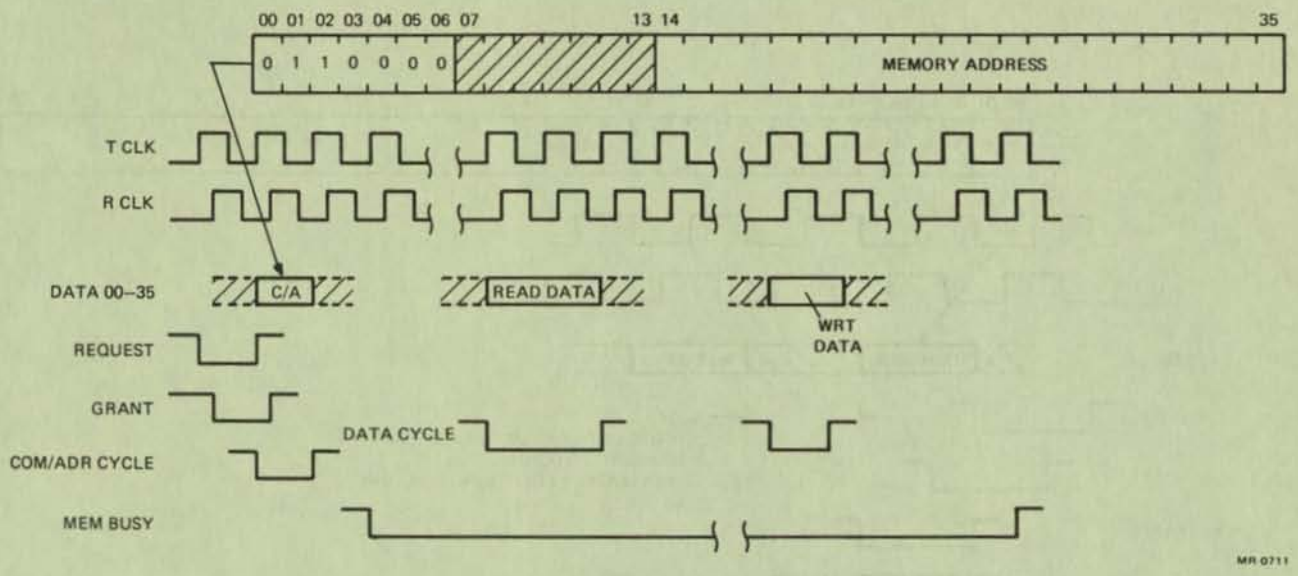


Figure 5-13 RPW, Bus Timing Diagram

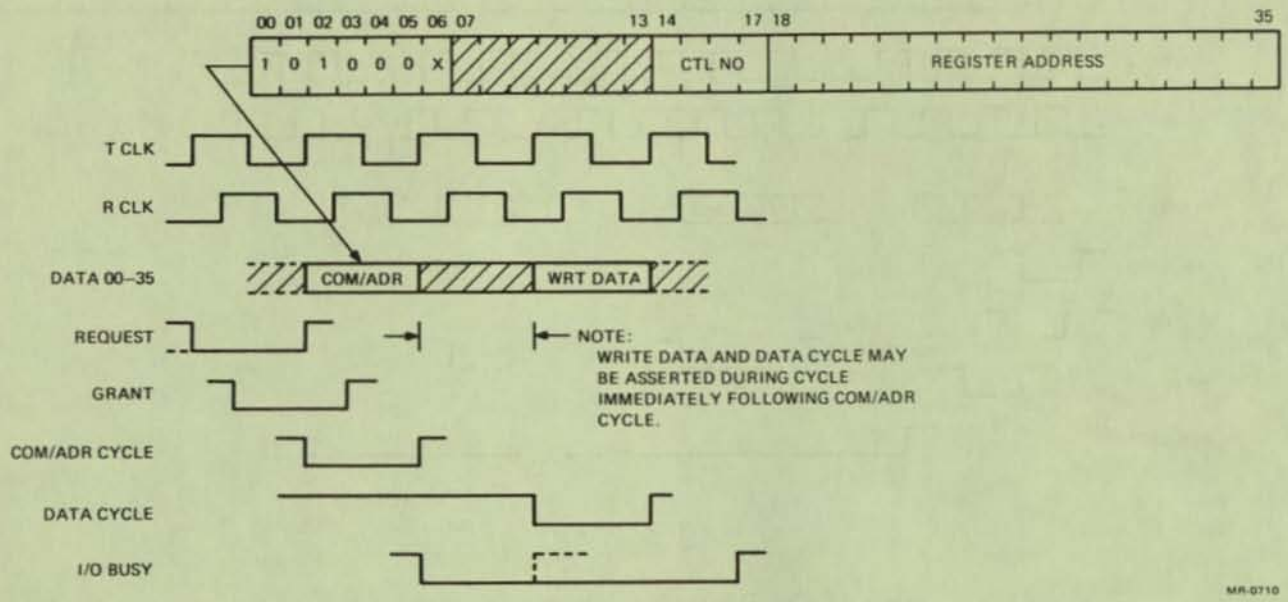


Figure 5-14 Write I/O Register, Bus Timing Diagram

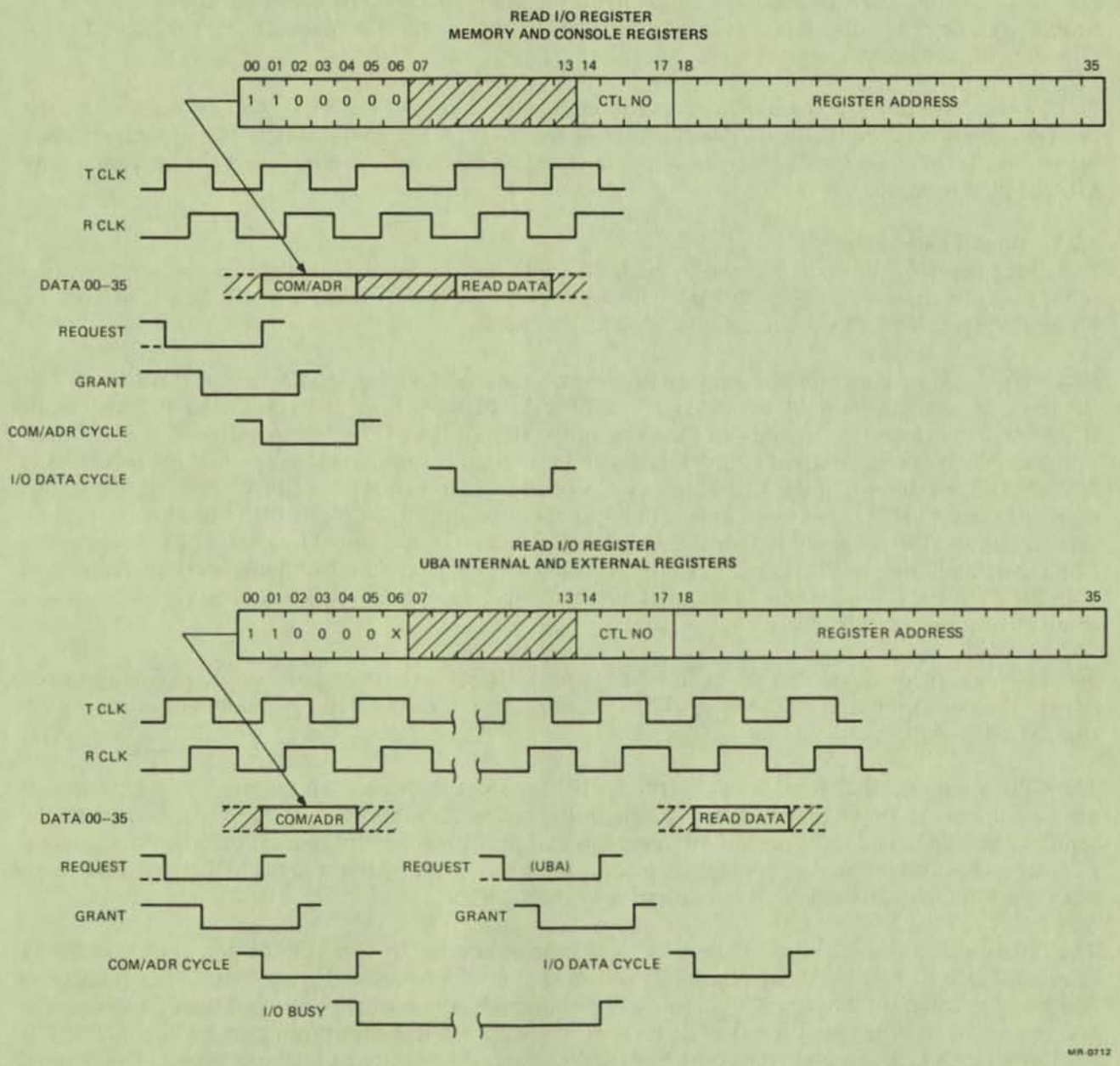


Figure 5-15 Read I/O Register, Bus Timing Diagram

reason for this is that the UBA must initiate a Unibus DATI operation to retrieve data from an external register, and the register data cannot possibly be supplied during the two bus cycles immediately following the command/address cycle. Although UBA internal registers could be read during these two bus cycles, the UBA control logic implements the same operation (bus request to transfer data) to simplify the design. For internal register addresses, the bus request is made during the second cycle following the command/address cycle.

During both types of I/O register read operations, control line I/O DATA CYCLE is asserted on the bus coincident with the read data. Similar to the DATA CYCLE signal asserted during memory read operations, I/O DATA CYCLE serves as a data strobe so that the device initiating the operation may gate the data from the bus.

5.2.8 Bus PI Operation

Part of the control information stored in the UBA status register are 3-bit high level and low level priority interrupt channel numbers (PIAs). The high level PIA is associated with BR7 and BR6 on the Unibus; the low level PIA is associated with BR5 and BR4.

When conditions are met for initiating an interrupt, a Unibus device asserts its assigned BR level. The BR level, in turn, causes the UBA to assert one of seven PI REQ lines (1–7) on the KS10 bus. The PI REQ line that is asserted depends on the value of the stored PIA (1–7) corresponding to the BR level. For example, if BR7 is asserted on the Unibus and the channel number stored in the high level PIA is 2, PI REQ 2 is asserted on the KS10 bus. As can be seen, with two levels of PIA, the UBA can assert more than one PI REQ at any one time. That is, in the preceding example, if BR5 was also asserted on the Unibus and the channel number stored in the low level PIA was equal to 4, the UBA would assert PI REQ 4 in addition to PI REQ 2. For the case when there are both high and low level interrupts and both PIAs are equal to the same PI channel number value, a single PI REQ would be asserted but as a result of two asserted BR levels.

When a high or low level PIA is set equal to 0, no PI REQ level is asserted on the KS10 bus even though the corresponding BR level is true. This provides a means for programmers to inhibit interrupt activity for a device.

The CPU monitors all PI REQ levels on the KS10 bus. More than one request line may be asserted at any one time (i.e., up to four with two UBAs in the system) and more than one UBA can assert the same request line. The CPU detects all interrupts and resolves interrupt request priority on a channel number basis (lowest channel has highest priority). When it is ready to serve the highest priority channel, it performs the first of two PI operations over the KS10 bus.

The first PI operation initiated by the CPU is to determine the UBA or UBAs interrupting on the PI channel that is to be served. Bus timing is shown in the upper part of Figure 5-16. After requesting and being granted the bus, the CPU asserts the command/address to specify that the operation is an I/O controller number read (bit 0 = 1, bit 4 = 1) for controllers interrupting on PI channel n (bits 15–17). When a UBA receives the command/address, and if it is interrupting, it compares the channel number value received on the data lines with the stored PIA. If a match occurs, a UBA asserts one of the data lines to indicate its physical address; that is, UBA1 asserts data line 19 and UBA3 asserts data line 21. The CPU strobes the data lines (during the second bus cycle following the command/address cycle), resolves controller number priority (UBA1 has highest priority), and then performs a second bus operation to read the interrupt vector from the highest priority UBA.

Bus timing for the second PI operation is shown in the lower part of Figure 5-16. The command/address specifies that an interrupt vector is to be read (bit 0 = 1, bit 1 = 1, bit 5 = 1) from controller n (bits 14–17). When the addressed UBA receives the command/address, it initiates a priority transfer control and interrupt sequence over the Unibus to read the vector from the interrupting device. The

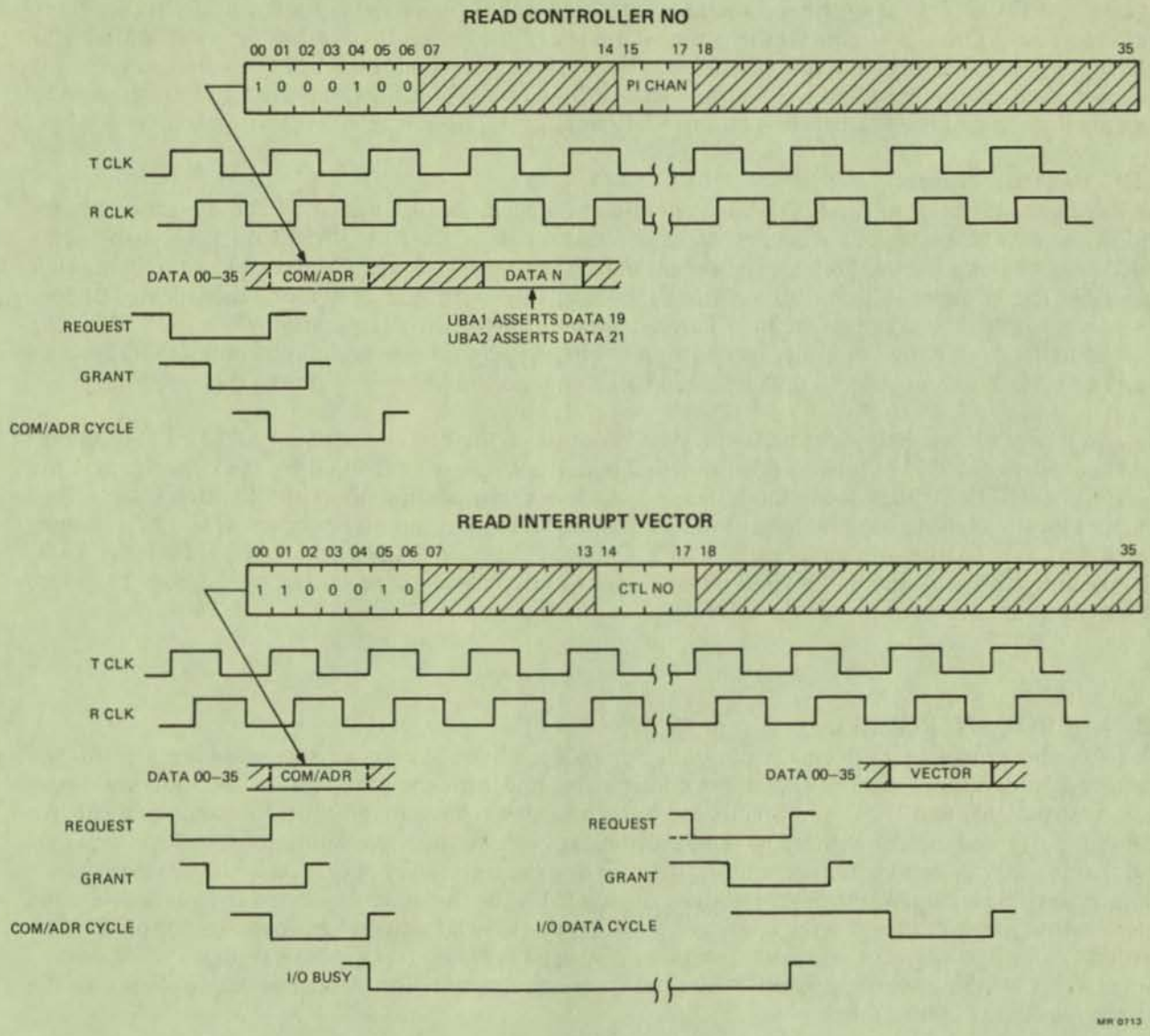


Figure 5-16 PI Operation, Bus Timing Diagram

vector is read from the device interrupting on the highest level BR associated with the specified PI channel. For example, if both BR7 and BR6 are asserted and the high PIA is being served, the vector is read from the device interrupting on BR7. Because the vector cannot be read during the two bus cycles allotted the CPU after the command/address cycle, bus operation is similar to the I/O register read operation. The UBA requests the bus at some later time, when the Unibus priority transfer and interrupt operation completes, and then asserts the vector address on the KS10 bus data lines when it has been granted the bus. The UBA also asserts I/O DATA CYCLE, which the CPU uses to strobe the data lines to end the PI operation on the KS10 bus.

5.2.9 Bus Parity Error

All devices connecting to the KS10 bus generate and check data line parity. Each device computes and transmits two parity bits whenever data is transmitted on the bus. PARITY LEFT is the computed parity (even) for the 18 least significant data lines (00–17). PARITY RIGHT is the computed parity for the 18 most significant data lines (18–35). Also, with one exception, each device checks parity when data is received on the bus. The exception is during the PI operation when the CPU reads the bus to determine the controller or controllers interrupting on a specified channel. Because more than one controller may assert a data line, the CPU ignores data line parity during this operation.

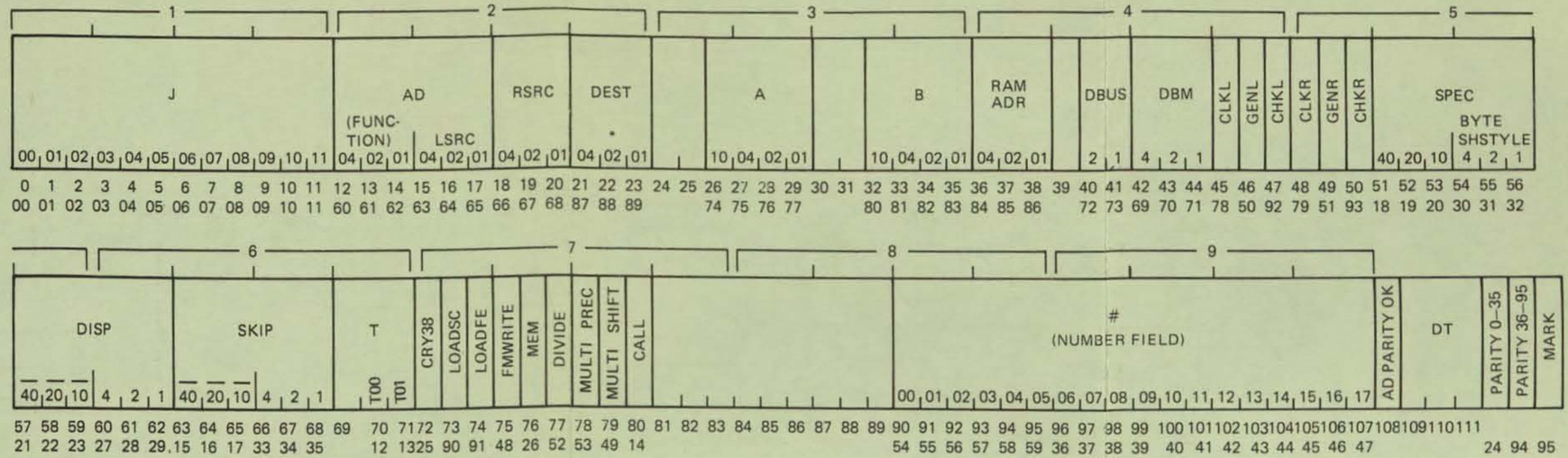
When a device detects bad (odd) parity for data received on the bus, it asserts a PARITY ERROR signal that causes the CPU clock to be stopped. The CPU clock is controlled by the console, and the PARITY ERROR signals from the various bus devices (including the console itself) are ORed together on the console module to set flip-flop CSL3 PE(1) when an error occurs. CSL3 PE, in turn, clears CSL5 ENABLE which negates CSL5 CRA/M CLK ENABLE and CSL5 DPE/M CLK ENABLE to stop the clock in all CPU modules. The parity error is also sensed by the 8080 program, which prints an error message at the CTY.

5.3 MICROCONTROLLER

The way the processor performs a program depends both on the processor hardware and on the microcode it executes. Most of the microcode is associated with the execution of the individual program instructions, and these are not treated here. The descriptive material in this and the next two sections is devoted almost entirely to the hardware, plus those microcode procedures of a more general nature, such as sequencing the microcode from one program-level operation to the next and handling priority interrupts and page failures. Associated with the microcode are two quantities, the microinstruction word itself, referred to as the "microword," and a dispatch word that supplies information for the execution of individual program instructions. The first two parts of this section discuss the structure of these words (Figure 5-17), and the rest of the section describes the hardware of the microcontroller (Figure 5-18).

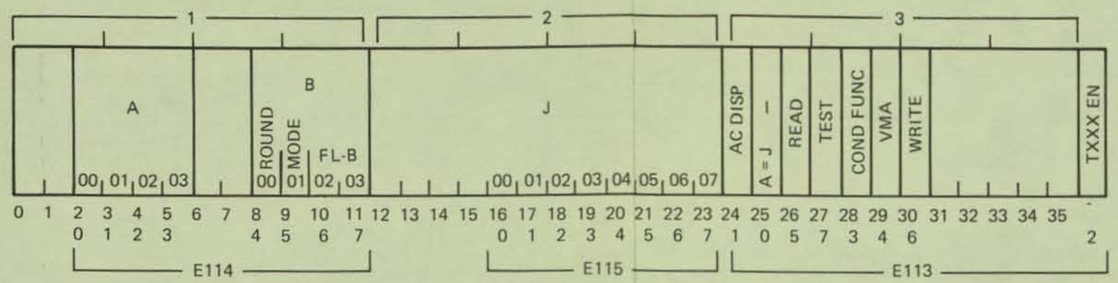
5.3.1 Microword

The upper part of Figure 5-17 shows the format of the control RAM microword. Of the two rows of numbers below the boxes, the upper lists the numbers of the bits as determined by the microcode assembler, and the lower lists their physical numbers according to their positions in the control RAM. Bits lacking physical numbers are either simply not used or are in special macro default fields, which are given in macro definitions but which do not appear in the assembly listing, as they are used only to default other fields that are in the actual microword. In the field definitions given in the microcode listing, the letter D means default to a constant, whereas F means default to a function, such as a macro default field. Bits that have only a physical number are created by the software that sets up the physical microwords. These include a mark bit for scope synching and two even parity bits, one for that part of the RAM contained on the CRA board (bits 0–35) and another for the rest of the RAM contained on the CRM board. The numbers above the boxes show the correlation between the parts of the microword as illustrated and the 4-digit groups that make up the words in the microcode listing. Preceding each word in the listing is the address of its location in the control RAM.



*INVERTED RELATIVE TO 2901 SPEC

MICROWORD



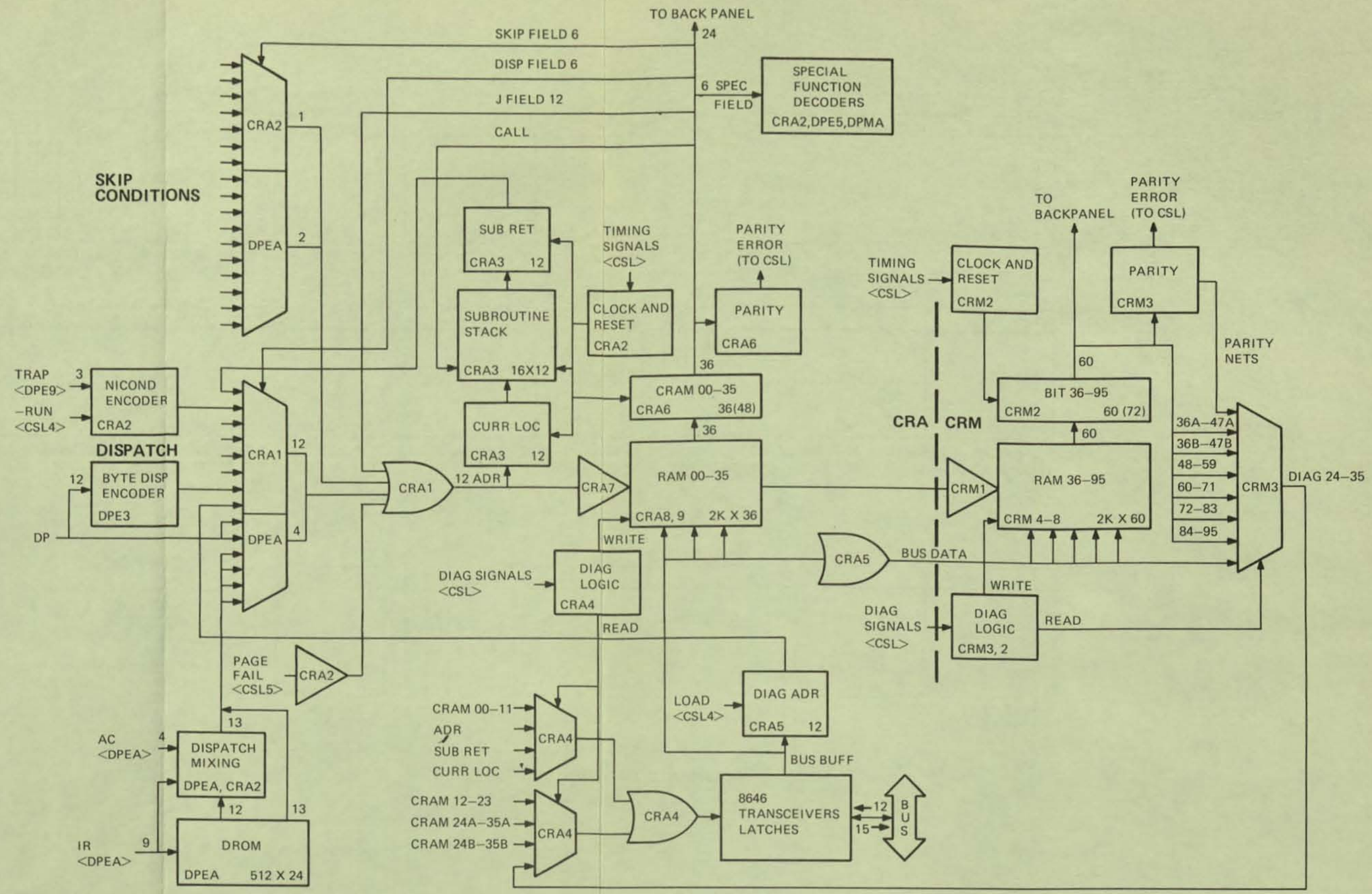
DISPATCH WORD

NUMBER FIELD

ARBITRARY NUMBER
SCAD, SCAD A, SCAD B, S #
STATE REGISTER
WORKSPACE IN RAMFILE
PC FLAGS
MEMORY OR I/O FUNCTION
BUS FUNCTIONS
PI (2 SETS)
AC + # SELECTION
APRID DATA
HALT CODES
FLG BITS

MR-1657

Figure 5-17 Microword Formats



MR-1658

Figure 5-18 Microcontroller

The labels used for the fields of the microword are those defined for the microcode assembly language. Multiple labels indicate bits used for more than one purpose depending on the circumstances – the number field is used for many purposes as listed at the lower right in the drawing. For bits labeled "inverted," a 0 rather than a 1 selects the defined function. The hardware signal names are very similar to the microcode labels and the reader should have no trouble identifying them; signal names for the bits on the CRA and CRM boards, respectively, are listed on CRA6 and CRM2, matched to the physical bit numbers. For those fields comprising more than a single bit, the signal names for the individual bits are numbered; the numbers are written on the bits inside the boxes in the format drawing.

The rest of this section explains the various groups of microword bits to serve as an introduction to the microcode listing. No attempt is made here to identify all the different quantities that can be selected by each field, as that information is given in complete detail at the beginning of the listing.

0–11 The address of the RAM location from which the next microword will be taken, perhaps modified by a skip or dispatch, or even supplanted altogether by a subroutine return, some other dispatch, or a page fail condition.

12–35 These fields govern the full word arithmetic unit. From the point of view of the microcode, there are 64 adder functions selected by the six AD bits, where the left three specify the function as defined by the 2901 spec, and the right three specify the source operand. The 9-bit instruction specification is completed by the three destination bits. Note that the middle bit in the destination code is inverted before being applied to the 2901s; hence the microcode configurations 0–7 correspond respectively to the destination control codes 2,3,0,1,6,7,4,5 as given in the 2901 spec. Note also that the terms "right" and "left" as used in the 2901 spec are opposite from their KS10 meanings, and inputs and outputs are numbered in the opposite order.

Physically, the adder is controlled as two separate left and right halves insofar as the operand source is concerned, and the right three AD bits select only the left source. The RSRC field enables the programmer to select a different right source in order to perform operations in which one half of an operand is manipulated as desired while the other half is (for example) simply cleared or left unchanged. If no right source selection is made, however, the RSRC field defaults to the value given for LSRC.

The A and B fields supply the A and B addresses for the 2901 register file. Of course a specified address has no effect unless the selected 2901 instruction calls for its use. Only B can select a register for loading.

36–38 The source of the RAM file address. Note that in this field a VMA selection means an ordinary memory reference, which may be virtual or physical and which may turn out to be a cache or AC reference, whereas a RAM selection means an absolute reference to any RAM file location via the right ten VMA bits.

40–41 The source of data for the D bus via the D bus mixer.

42–44 If the D bus field selects DBM as the source, this field selects the source of data into the mixer that feeds the DBM input to the D bus mixer.

45–50 These are two sets of three bits that separately control certain operations in the left and right halves of the main data path. Bits 45 and 48 separately clock the two halves of the arithmetic unit, so that operations can be performed in one half while the other is unaffected. Note that this means there is no change at all in the other half: to load an arbitrary destination with a word half modified and half unchanged requires clocking both halves with separate left and right source selections.

For parity purposes there are an even parity bit and a valid bit associated with each half word in the register file. Whenever a location in the file is loaded, the two associated parity bits are set up from the parity signals generated for the two half words on the D bus, and the valid bits are set up from bits 46 and 49 of the microword. Hence by means of the valid bits, the microprogrammer can label the parity bits according to whether they actually do represent true even parity for the stored half words. The parity signals generated for the D bus are correct for a word stored if the operation performed by the arithmetic unit is parity-conserving, as is the case for a simple transfer or ANDing with the mask, and the source of the data word is the RAM file or DBM. Of course parity storage is also valid if the operation is simply the transfer of the contents of register A to register B and the D bus mixer selects DP. For convenience in handling these control bits, a macro definition can put a 1 in bit 108 to indicate that the macro operation conserves parity: in a microword containing the macro, the GENL and GENR fields default to the value given by bit 108 unless the programmer overrides it.

Bits 47 and 50 enable parity checking on the left and right halves of the D bus. A parity check should always be made when the source of the D bus data is the RAM file or the backpanel bus (MB). When the D bus mixer selects DP, the parity check should be made only if the AU operation conserves the parity given by the bits associated with the file location selected by the B field (as that is the source of the parity indication against which the check is made). Even then the requested check for either half is overriden by the corresponding valid bit being off, indicating that the stored bit does not represent true parity for that register. No check should ever be made when the source is VMA or any DBM selection other than MB.

51–56 This field selects among a number of special functions such as loading IR, manipulating flags, and sweeping the cache. Additionally, if the AD function being performed involves shifting, the right three bits control the connections at the adder extremities for the type of shifting; and if the DBM field selects the data path input to the DBM mixer, the same three bits select the position (if any) for insertion of a 7-bit byte in the word. The right three bits are decoded together, and the left three individually select groups of eight functions. Hence functions can be combined. The same right three configurations select loading IR and XR, so they can be loaded together. (IR includes AC, and XR includes the indirect bit.) Similarly the right code that selects arithmetic shifting also selects (via the 40 bit) the ASH overflow test, which is used for left shifting.

57–62 This field selects a quantity to be ORed with J field bits 0–7 or 8–11 or both to select the location of the next microword to be executed. To jump to a specific location such as that given by the J field of the dispatch ROM or a return address from the subroutine stack, the microword J field must be zero.

63–68 This field selects a skip condition, which if satisfied, causes a 1 to be ORed into bit 11 of the address for selecting the next control RAM location. Thus the microcode can jump to an even location with the possibility of skipping that word and going directly to the next odd location. The skip field can select one, two or three conditions from among three sets of six, where the skip occurs if any selected condition is satisfied.

70–71 The processor cycle is extended beyond two clock ticks by the number of ticks specified by this field. For convenience the T field defaults to the value given by bits 109–111, which can be specified in a macro definition. Thus a macro can indicate when extra time is needed for the operation it produces, but the programmer can override this specification if the extra time is not needed because of the circumstances in which the macro is used.

72 Inserts a carry into the LSB of the adder.

73 Loads the step counter from SCAD as set up by the number field.

74 Loads FE from SCAD as set up by the number field.

75 Writes the contents of the D bus into the RAM file at the location specified by bits 36–38.

76 Starts or completes a memory or I/O function (usually a bus transaction) whose characteristics are specified by the number field.

77 This microinstruction is doing a divide.

78 This microinstruction is doing a multiprecision step in DFAD, DFSB or divide.

79 Causes this microinstruction to be executed as a no-op if FE bit 0 is already 0, but otherwise causes it to be repeated until FE overflows (bit 0 becomes 0). This feature is used for fast shifting.

80 Pushes the current location on the stack to effect a subroutine call.

90–107 This field supplies information for a variety of functions selected by other fields. The kinds of information are listed in the format drawing.

5.3.2 Dispatch Word

The 9-bit instruction code in IR automatically selects one of the 512 locations in the dispatch ROM and makes its contents available to the skip and dispatch logic. Dispatching on the given information occurs mostly in the part of the microcode labeled "The Instruction Loop", which appears at the beginning of the listing just after the power-up sequence. The format of the words supplied by the dispatch ROM is shown in the lower part of Figure 5-17 using the same conventions as for the microword given above. However the dispatch ROM bits are not numbered physically, so chip locations and outputs are given instead.

2–5 This field specifies the kind of operand fetching to be done for the instruction, and for a simple read indicates whether the next instruction can then be fetched immediately.

4–7 This field specifies the test condition in all test instructions, the modification of the masked bits for logical testing, and in all other instructions it specifies the disposition of the results except that in floating point it also indicates whether there is rounding and whether the operation is additive or multiplicative. The extra physical bit, TXXX EN, shown at the right end of the word, is a duplicate of bit 9 of the B field.

12–23 The address of the control RAM location at which execution of the instruction begins. This 8-bit field selects a location in the range 1400–1777.

24 Causes the AC field of the instruction word to replace the right four bits of the J field so that a jump to begin instruction execution will actually dispatch to one of 16 locations where the instruction code is expanded to 13 bits.

25 Causes an immediate dispatch on the J field when the microword calls for the standard AREAD dispatch on the A field. This is used for instructions that require no memory access or special setup (e.g., MOVEI, JFCL).

26 Starts a memory read when the microword selects an AREAD memory function.

27 Starts a write test (for page fail) when the microword selects an AREAD memory function.

28 Starts a memory cycle when the microword selects a BWRITE conditional memory function.

29 Loads VMA when the microword selects an AREAD memory function.

30 Starts a memory write when the microword selects an AREAD memory function.

5.3.3 Control RAM

The 2048-word control RAM (Figure 5-18, upper right) is made up of a pair of 1K RAM chips for each microword bit. Bits 0–35 are on the CRA board and are shown on prints CRA8,9; bits 36–95 are on the CRM board and are shown on CRM4–8. An entire microword is selected by selecting a single bit from each pair of chips. Selection is made by an address supplied by the skip and dispatch logic (Paragraph 5.3.4) and applied to the two parts of the RAM through the drivers on CRA7 and CRM1. Associated with the two parts of the RAM are two parts of a register that holds each microword while its bits are controlling the events that constitute its execution and are supplying an address for use in selecting the location of the next microword. These two parts are the CRAM register on CRA6 and the BIT register on CRM2. (In each there are duplicate flip-flops for the 12-bit segment that sustains the heaviest use.) At the end of each processor cycle the clock triggers the events for one microinstruction and loads the next into the register from the RAM. However if a 1 in the multishift bit disables the microcontroller clocks (on CSL) without affecting the data path clocks, the same microinstruction is repeated. On the other hand, when FE 00 is 0 in a fast shift, the data path clocks are disabled but not the microcontroller clocks, so a no-go results and the next microword is loaded. The parity nets for checking the CRA part of a word are at the upper left on CRA6, and those for the CRM part are across the top and in the lower right corner of CRM3. The outputs of the nets go directly to CSL to stop the processor clock should an error occur.

The J field is used solely by the microcontroller address logic; all other CRAM and BIT signals are available via the backpanel to other boards, although most of the skip, dispatch and special function bits are used on CRA. Most of the bits that control the 2901s are applied directly to those chips, although a few are also used elsewhere in the arithmetic logic. Most other multibit fields are applied to mixers to select among various sets of inputs, such as the data for DBM or the address for the RAM file. The right three special bits are applied to mixers for selecting shift inputs at the 2901s, but are otherwise applied to decoders for generating specific functions, where the individual decoders are enabled by 1s in the 40, 20 and 10 bits, or for byte insertion, by the appropriate configuration of the DBM field. In some cases, duplicate decoders are employed in order to get a function signal as close to the target logic as possible, and in a couple of cases individual function signals are duplicated for use in two different places. Decoders enabled by the 40 and 20 bits are at the upper left in DPE5; at the upper right in DPMA are a decoder for the 10 bit and a duplicate of that for the 20 bit (note that except for the memory wait function, these decoders are enabled only during the low period of the cycle clock); and a duplicate for the 10 bit is at the right on CRA2.

5.3.4 Skip and Dispatch Logic

The logic that determines the location from which the next microword will be taken is shown in the left quarter of Figure 5-18 and appears mostly on prints DPEA and CRA1,2. Each address is supplied to the control RAM through the OR gates above the two rows of mixers on CRA1. From the 6-bit microword dispatch field, individual inputs to the mixers are selected by the right three bits and the different sets are enabled by single bits among the left three. The upper row on CRA1 has 4-bit mixers for the left eight address bits, and these are enabled by the 20 dispatch bit. The lower row, enabled by the 10 bit, contains 8-bit mixers for the right four address bits. A similar set of mixers for the right four bits but enabled by the 40 bit appears at the upper right on DPEA; the outputs of this set are applied directly to the lower row of OR gates on CRA1 as the DPEA DISP signals.

The OR gates on CRA1 combine the outputs of the several sets of mixers with the J field from the CRAM register. Hence there are two ways to address a single, arbitrary location in the control RAM: with the dispatch mixers disabled, the microcode can jump to the address given by the J field; with J zero, dispatch mixers for all 12 bits can supply a specific number, such as a diagnostic or subroutine return address. But the microcontroller can dispatch within a range of four, eight or 16 locations, starting at that given by the J field, by ORing a variable quantity into the right four address bits through the mixers enabled by the 10 or 40 bit. Note that for an individual mixer to have any effect the corresponding bit in the J field must be 0; a 1 in the J bit overrides any selection made by the mixer.

At the lower right on CRA1 the OR gates for address bit 11 also receive the outputs of the three skip mixers. This arrangement allows the microcode to give an even J with the possibility of going instead to the next odd location on the satisfaction of any of three independently specifiable skip conditions. The skip mixers function from the skip field of the microword in exactly the same way as the dispatch mixers. The mixers for the 40 and 20 bits (which handle mostly flag and arithmetic conditions) are in the upper left corner on DPEA, and the mixer for the 10 bit is at the upper right on CRA2. Note that the signals that can be selected for skipping are all inherently synchronized to processor operations except for conditions 4–7 in the CRA2 mixer. These four conditions are therefore synchronized to the cycle by means of the flip-flops in E115 (D3). One of these signals, I/O LATCH, is the OR function, by way of a flip-flop in E416 (A6), of the two bus signals that represent response to an I/O instruction. The synchronization is handled via the bottom gate in the clock logic at the left and the top flip-flop in E416. The skip condition flip-flops are set up at the end of every processor cycle through assertion by the clock enable of the signal DISP & SKIP EN. The same T clock also sets the top flip-flop in E416 to generate FIRST CYCLE, which really means the first tick in the processor cycle. If microword bit T01 is 1, indicating a three-tick cycle, the asynchronous skip conditions are updated at E115 so they will be fresher when used at the end of the cycle.

Finally, note that the page fail signal from the console is fed into all of the address OR gates. Hence it can override any selection made by the J field or the skip and dispatch mixers, and force selection of location 7777.

5.3.4.1 Dispatch ROM – The left half of each instruction is loaded from the D bus into the IR, AC, indirect and XR registers at the left on DPEA. Each instruction code from IR selects a location in the dispatch ROM, made up of the three 512 × 8-bit chips at right center. The outputs from the left chip are used as individual control signals or skip conditions, as in the net at C5 where TXXX EN is combined with AD = 0 to decide on a skip in the microcode to execute a skip or jump in an instruction. The microcode can dispatch on the A and B fields from the center chip by way of the bottom two inputs to the dispatch mixers at the top of the print; the latter occurs in the "Store Answers" part of the instruction loop and elsewhere for specific instruction groups. Dispatching for instruction execution is on the J field from the right chip but this is somewhat roundabout. J bits 0–3 are input to address bits 4–7 through the upper mixer on CRA1, and the same dispatch function puts 1s in bits 2 and 3 so dispatching is in the range 1400–1777. In the normal situation J bits 4–7 go to address bits 8–11 through the dispatch mixer at the top of DPEA as the DPEA J signals available from mixer E118 (A3). But on an AC dispatch for JRST or an I/O instruction, the AC address is substituted for the DROM J bits.

The standard AREAD dispatch on the A field for the "Fetch Arguments" part of the instruction loop uses control RAM locations 40–57, but a 1 in the I bit of the dispatch word can cause an immediate dispatch on the J field. This is accomplished through two mixers, E119 at DPEA A 2 and E420 at CRA2 C3. For the standard dispatch, AREAD bits 8–11 from E119 are equivalent to the DROM A field and are supplied to the address through input 3 of the mixers at the top of DPEA. E420 sets AREAD 04–07 to 0010, which selects the desired range through the upper mixers on CRA1. For an immediate dispatch, the I bit, which is the A = J signal, substitutes J 08–11 in AREAD 08–11, substitutes J 00–03 in AREAD 04–07, and inserts 1s directly into address bits 2 and 3 via mixer E517 (CRA1 C6) to make the range the same as that used for an ordinary J dispatch.

5.3.4.2 Other Dispatch Procedures – The next instruction condition or NICOND dispatch appears at the very beginning of the "Start Next Instruction" part of the microcode instruction loop. The dispatch is handled through the lower mixers (input 4) on CRA1 and the signals are generated, except for the most significant, through the priority encoder at E216 (CRA2 B3). Only five encoder inputs are used, and at the end of any program-level microcode operation they provide for dispatching to the next operation in the priority order trap 3, trap 2, trap 1, halt, and the ground at input 7 provides for going on to the next program instruction if none of the other conditions intervenes. The dispatch in the microcode actually has two sets of five locations distinguished by NOCOND 08, which simply indicates whether a memory cycle is in progress. This condition has no effect on traps or a halt, as the microinstructions in each pair of dispatch locations distinguished only by the memory condition are identical. But it does affect the next normal instruction and indicates whether the instruction must still be fetched or is already being prefetched.

The D6 input of the mixers associated with the 10 bit provides for dispatching to every other location among 16 for the effective address calculation, which immediately follows the NICOND dispatch in the microcode listing. Here again there are two categories of dispatching depending on whether or not there is indexing or indirection, one specifically for the instruction JRST 0, and one for all other instructions. The special case is the most frequently used instruction in the entire PDP-10 set, and the AND gate at A4 on DPEA saves a processor cycle by detecting JRST 0, directly from IR, obviating the J dispatch.

The most common byte size used is seven bits. The KS10 saves considerable time by having hardware for manipulation of 7-bit bytes with zero alignment built right in. Most of this hardware is associated with the DBM mixer and the 10-bit logic (Paragraph 5.4) but the microcontroller has a mechanism for dispatching on byte position; that is, on which byte in the word is being processed. The three byte dispatch signals available at the D5 inputs to the lower CRA1 dispatch mixers are provided by the decoder-mixer combination at the lower left on DPE3. When DP carries a byte pointer, the decoder is enabled by a size indication of 7, and the circuit translates the zero-alignment byte positions into byte dispatch configurations as follows.

Byte	Position	Dispatch Code
1	29	001
2	22	010
3	15	100
4	8	101
5	1	111

The single D7 input to the lower CRA1 mixers (at E122) provides for a skip of two locations (actually to the next even location) from the microword J field when SCAD is negative. Similarly the arithmetic condition at E121D2 provides a four skip that is used in multiplication. The remaining inputs to the mixers at the top of DPEA provide for dispatching on various sets of bits in a word on DP, in one case combined with arithmetic conditions.

5.3.5 Subroutine Stack

The binary counter and RAMs in the middle row on CRA3 provide a standard stack for microcode subroutine calling. Position in the stack is determined by the value in the counter, which goes up for pushing and down for popping. The top of the stack is defined as the location whose address is one greater than the number in the counter, and the stack therefore allows a depth of subroutine nesting of 15 levels. The address lines to the RAMs carry the current value in the counter unless SELECT NEXT enables the gates at the right of the counter, in which case they carry an address two greater. Following the initial reset of the counter from the console, the top of the stack is at location 1.

At the end of every processor cycle, the cycle clock loads the address of the next control RAM location into the current location register at the bottom of the drawing. Halfway through the first tick the stack write signal (through the top gate in the clock circuitry on CRA2) loads the current location into the RAM position one above the current top of the stack as selected by the select-next gates. This is done at the beginning of every microinstruction – if it turns out to be unnecessary, the stored address is just thrown away when the next current location is loaded in its place. However, if the microinstruction is a call or there is a page failure (the call or return signal from CRA2 A5 includes the page fail condition from the console), the counter is incremented so the temporary save location now becomes the top of the stack. Simultaneously the saved address is loaded into the register at the top of the drawing so it is available for a subsequent return. On the other hand, if the microinstruction is a return (and thus makes use of the SBR RET address), the select-next gates are disabled, and at the same time the cycle clock decrements the counter, the address from the top of the stack is moved to SBR RET for a subsequent return from the level in which the just-executed subroutine was nested.

5.3.6 Booting and Diagnosis

The logic through which the console directly manipulates the microcontroller is shown across the bottom of Figure 5-18. The reset signals are located on the same prints as the clock circuitry. Note in particular (CRA2 A3) that the reset for the stack is separate from that for the microinstruction register, so the console can clear the register, and then inspect locations or single step, without bothering the stack.

To bootstrap the microcode, control signals from the console bring in data 12 bits at a time from the backpanel bus via the transceivers on CRA5. Loading each location in the control RAM requires nine transfers, the first for an address, which is loaded into the register at the top of the drawing, and eight more for the 96-bit microword in 12-bit segments. With the microinstruction register clear, J is zero, the skip field selects no condition, and the dispatch field selects the diagnostic address through the mixers on CRA1. By means of the gates and decoders at the bottom of CRA4 the console can select and write three segments in the addressed location in the CRA part of the control RAM, and similar logic at the lower left of CRM3 handles the selection and writing of five segments in the CRM part.

The same selection signals, but with the write replaced by a read (CRM2 B3), can read any 12-bit segment of the CRM part of the microinstruction register, the contents of the transceiver latches, or the output of the CRM parity nets through the mixers on CRM3. The same signal that enables the CRM read mixers, CSL4 DIAG 10 H, disables the upper mixers on CRA4 and selects the output of the CRM3 mixers as the input to the lower CRA4 set. When that signal has the opposite polarity however, the signals that select the sections of the CRA part of the control RAM select 12-bit CRA inputs to one or the other set of mixers on CRA4. The quantities selected can be any part of the CRAM, the contents of the current location or subroutine return register, or the address supplied to the control RAM by the skip and dispatch logic. The output of either mixer set is available through the OR gates at the top of the drawing to the TRN inputs to the transceivers on CRA5. Note that the parity signals generated for the transmitted data by the three transceivers that handle the 12 bits are themselves transmitted through a fourth transceiver on an additional three data lines to make the parity on the bus even.

When the console first starts the microcode, it executes the "Power-up Sequence," which is at the beginning of the listing. In the register file this sequence sets up the constants, clears control words, and also clears temporary registers to avoid parity errors. In the workspace it sets up a table of powers of 10 for binary-to-decimal conversion, clears locations for the time base and flag enables, and saves the address of the halt status block. Finally it clears the flags, enters executive mode, and enters the halt loop.

5.4 DATA PATH EXECUTE

Although the activities of the two data path boards are intertwined, the logic can reasonably be divided into two parts: the execute data path, which handles all the internal operations for the execution of an instruction – arithmetic and logic operations, data manipulation; and the memory data path, which handles all aspects of communication over the backpanel bus for both memory and I/O instructions, including determining whether a memory access should be made also or instead to the RAM file (a cache or AC reference) and thus require action by the execute data path. Although the boards are labeled DPE and DPM, the logical and physical boundaries do not coincide, and both paths include elements on both boards. Here we deal with the execute part of the path; the memory part is discussed in Paragraph 5.5. Figure 5-19 is a block diagram of the execute path, but for the internal structure of the register and RAM files refer to Figure 5-3.

5.4.1 Arithmetic Unit

The heart of the main data path is the arithmetic unit (shown on prints DPE1,2) and most of the logic is the 2901s themselves. Just as 36-bit words are centered in the 40-bit register file, the D bus inputs are centered in the ten slices, with the extra pair of bits at the left receiving copies of bit 0 (the sign), and the extra pair at the right receiving 0s. The chips are interconnected for left and right shifting, but instead of direct carry connections the carry function is handled through look-ahead logic supplied by the 2902s at the bottom on DPE2. Note that the carry from the right half to the left (i.e., into bit 17) is controlled by the microcode. Also under microcode control are the separate clocks for the two halves via the middle gates in the clock circuitry at the left on DPE5. The bits from the appropriate micro word fields, with separate left and right source selections, are applied directly to the chips with two exceptions: the 02 destination bit, which must be inverted and distinguishes between left and right shifting in those functions that do shift; and the 01 function bit, which distinguishes between add and subtract when the 04 and 02 bits are both 0.

Having words centered in the 40-bit adder is appropriate for some one-word shifts and for additive operations, as the sign is available at the left end, or for LSH the extra bits can be masked out; and moreover the result of an arithmetic operation can never exceed 40 bits, so DP SIGN always has the correct sign even when DP 00 is wrong because of overflow. On the other hand, for arithmetic shifting and multilength operations it is not suitable to have words centered, as there is then a hole in the middle of a double length operand in AD and Q (which can be shifted together), and the connection between the sign and bit 1 is buried in the leftmost chip. Hence before performing such operations, the microcode must move the operands to the right, frequently placing them entirely in the right nine chips, which are then used as a 36-bit AU. That such action is expected is evidenced by the signals that serve as inputs to the shift logic at the bottom on DPE1 and by the fact that the carry out of bit 2 is an input to several logic nets and is available for testing by the microcode. The net in the lower right corner on DPE5 performs the necessary inversion of the 02 destination bit and also supplies the left and right shift signals to the shift logic on DPE1. When shifting is called for, the gates at the far left supply shift connections that are constant for a given direction, and the tristate mixers decode the right half of the special function field to set up those connections that vary depending on the type of shift. The tristate logic is necessary because the 2901 pins that receive inputs for shifting in one direction supply outputs for the other, at which time the corresponding tristate circuits are disabled so their outputs neither drive nor load the signal lines significantly. Figure 5-20 shows the various kinds of shift arrangements for shift instructions and arithmetic subroutines, where the short boxes represent the left slice and the long boxes the other nine. The indicated use is for the main shift activity, and the

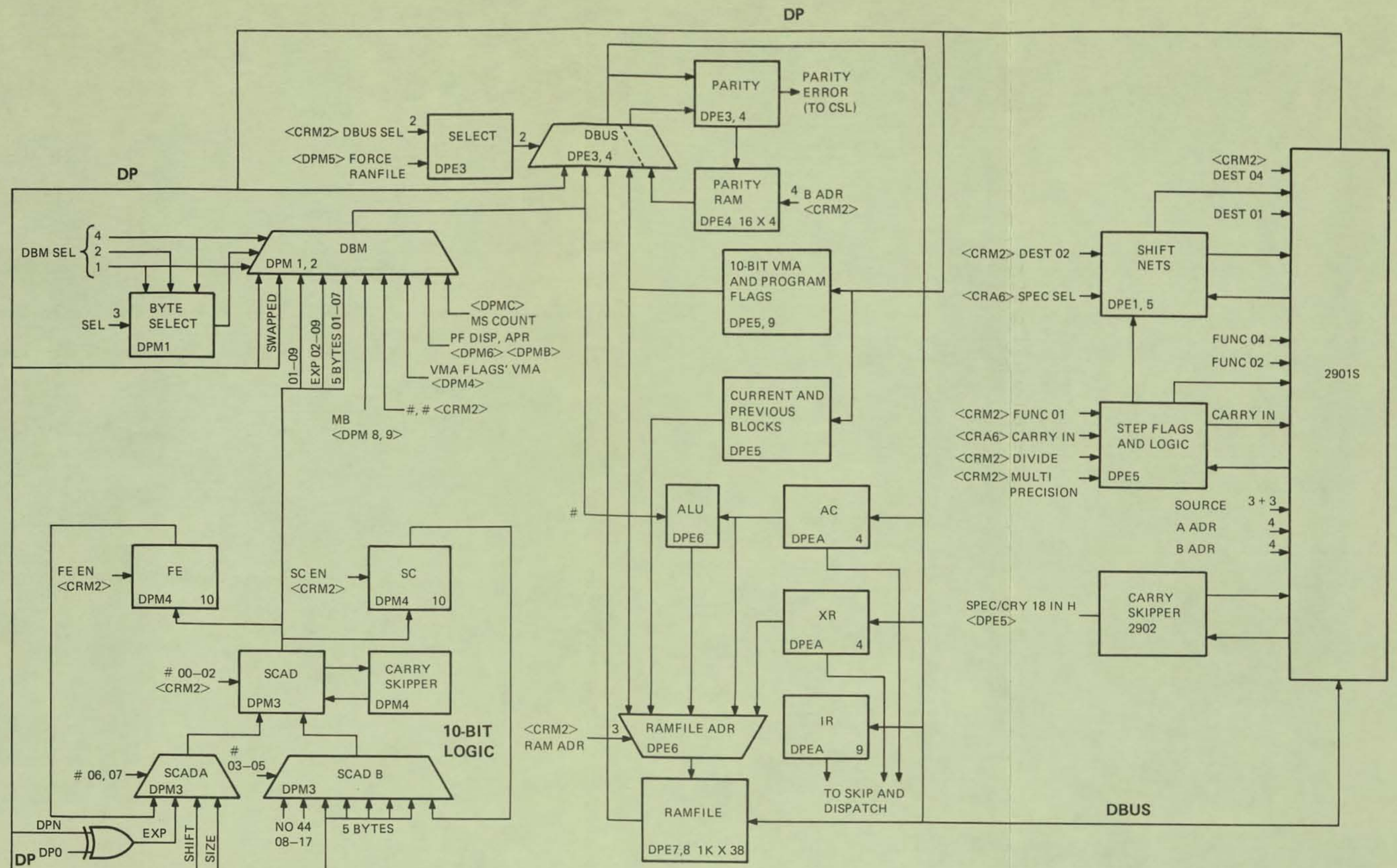


Figure 5-19 Data Path Execute

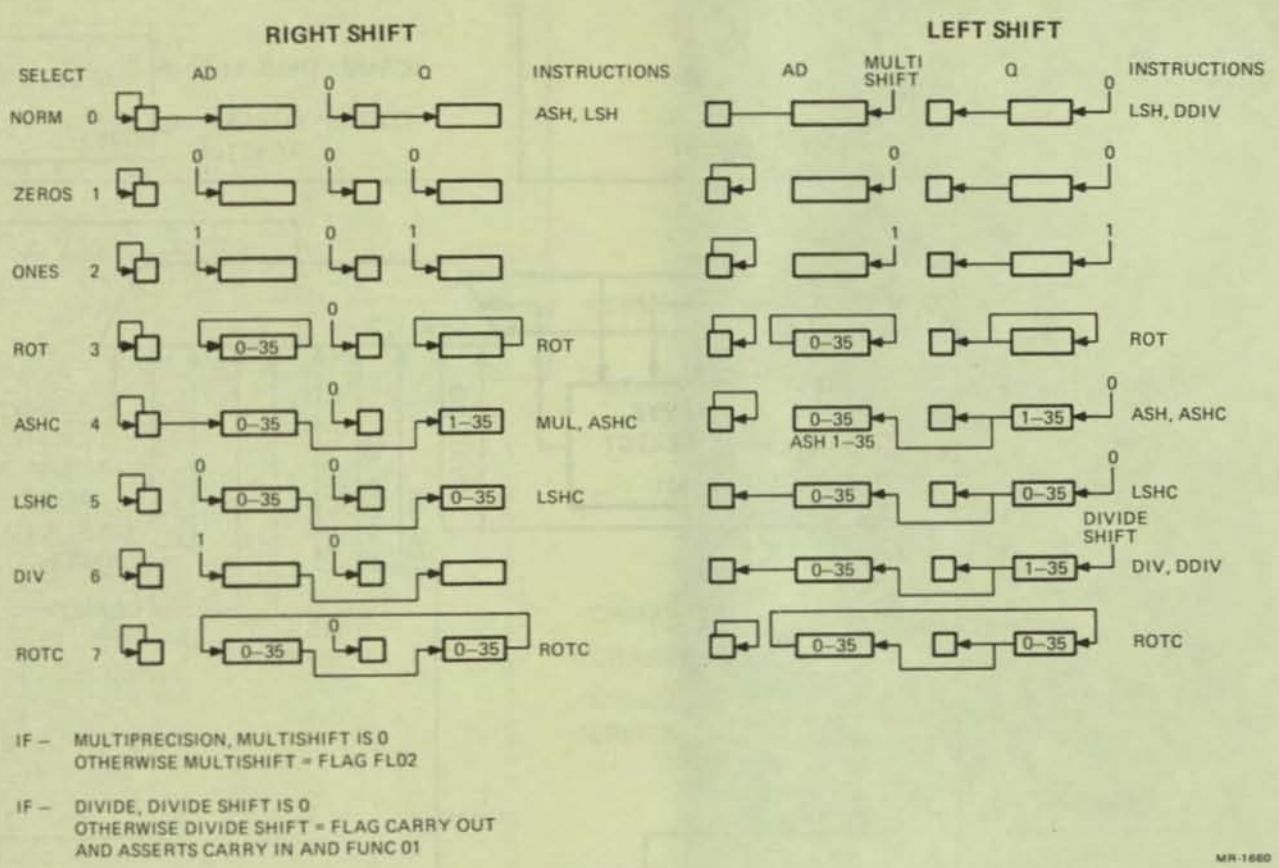


Figure 5-20 Shift Configurations

numbers inside the boxes indicate the initial position of the operands for the type of shift, if different from the normal. Before the main shifting activity, the microcode must of course move the operands from their normal positions to the ones given, making use of whatever shift arrangement is appropriate.

Note that two of the bit inputs to the shift logic do not come directly from the 2901s. These two signals plus the carry into the right end of the adder and the above mentioned 01 function bit are supplied by the gates at the right on DPE5. Through the top two gates, 1s in the corresponding microword bits do assert the function and carry signals, but the rest of the logic is for assisting the microcode in division and certain multiprecision operations. The flip-flops save information from one step for use in the next or from operations on lower order words for use on higher order. In division for example, the carry out in one step means that in the next a 1 must be shifted into the partial quotient and the divisor must be subtracted from the dividend; hence FLAG CARRY OUT being set causes a divide step to assert DIVIDE SHIFT for input to the shift nets and implements a subtraction by generating a carry in and asserting the 01 function bit. This simple hardware feature saves a great deal of microcode time: instead of requiring the microcode to use a skip to decide whether to add or subtract in each divide step, it simply calls for an add in every step, and when the carry is present the add changes automatically to a subtract accompanied by the carry in required for 2's complement arithmetic. In a similar way the microword multiprecision bit carries over a subtraction from one step to the next, inserts a carry in a high order operation if there was a carry out of the low order (using the right nine slices), and bits shifted left out of the second position can be inserted at the right in the next normal shift step via MULTI SHIFT. This last signal, which has absolutely nothing to do with the microword multishift bit, supplies 0s in LSH. FLAG QR 37 provides multiplier bits for dispatching in the multiply subroutine.

5.4.2 Main Path

The output of the arithmetic unit is available via DP to many processor elements, including the mixers for the D bus on prints DPE3,4. These mixers can also select the output of the RAM file, the output of the DBM mixer on the DPM board, or a word made up of the program flags, the number of the PI level on which a new request has been accepted, and the right 10 VMA bits that are kept on the DPE board for accessing the RAM file. Input selection is made according to the microword DBUS field through the gates at bottom center on DPE3. But note that a microword selection of DBM can be forced to a RAM file selection instead; this occurs when DBM is selected for MB and the memory request turns out to be a cache hit or an AC reference. The selected word can be sent over the D bus to the arithmetic unit, the RAM file or the instruction register, which is at the lower left on DPEA. All of the instruction bits can be loaded together by two special functions, of which one handles both the IR and AC fields, and the other handles the XR and indirect fields as well as a bit that indicates the instruction is being executed by a PXCT and should do its indexing in the previous context. XR and AC are decoded for zero for use by the skip and dispatch logic.

The remaining D bus logic is for parity operations, and includes the standard nets for generating even parity bits and checking parity. It also includes, on DPE4, the E714 flip-flops and 16 × 4 RAM that implement the parity arrangement described in the discussion of microword bits 45–50 in Paragraph 5.3.1. The RAM contains two validity bits and two parity bits for each location in the register file and is written according to the B field selection whenever the destination code loads a register. (Writing occurs at the leading edge of the cycle clock, 75 ns before the 2901s are clocked, through the bottom gate in the clock circuitry at the left on DPE5.) The E714 flip-flops allow generation of a left or right parity error signal for the console when the corresponding check indicates bad parity, but only if the microinstruction enables the parity check and the hardware provides the appropriate DBUS CHK EN signal. These enable signals are supplied through an extra mixer for each half of the bus. The signals for both halves are always false on VMA selection and always true on RAM file or DBM selection (in the last case the microinstruction should enable parity checking only if MB is the source, because the parity bits that accompany the DBM selection are those supplied by the backpanel bus).

When DP is the source, the parity bits are those supplied by the RAM location selected by the B field, and the D bus check enable signals stem from the corresponding valid bits.

The final part of the main path is the DBM mixer, which appears on DPM1,2. Inputs, as selected by the microword DBM field, can be any of those listed in the table at the lower right on DPM1. Selection of "bytes" provides 0 in bit 35 and five copies of SCAD 01-07 in the other 35 bits. Reading the exponent puts a 0 in bit 0, the exponent from SCAD 02-09 in bits 2-9, and fills the rest of the left half from DP but reads the current value of the MSEC counter in the right half. The number field is duplicated on the two halves of DBM. The bits of the microword DBM field are applied to buffers for multiple drive lines for the mixers. Because the drive lines for the 4 and 1 bits typically each drive a third of the mixers, the 2 bit has five lines, each corresponding to a 7-bit byte. These lines are further gated by the configurations of the right half of the special function field through an E412 decoder that is enabled by DBM select code 1 or 3. When the code is 1 all of the drive lines for the select 2 bit are off as required. For code 3, the select 2 drive lines that are on cause selection of the DP input, but a function number from 1 to 5 turns off the corresponding select 2 line, causing one set of seven mixers to select the input for code 1 instead of code 3. This inserts a 7-bit byte from SCAD 01-07 in the selected position with the rest of the word made up from DP. Byte 5 is handled as eight bits but the final mixer receives DP 35 for either code.

5.4.3 RAM File

The 38 RAMs on DPE7,8 provide storage for 1024 words with an even parity bit for each half. The word contained in the location selected by the address inputs is available at the RAM outputs, and a falling edge at the write input replaces it with the contents of the D bus. The write signal, which occurs at the falling edge of the cycle clock, is produced through the gates at upper center on DPE5 upon command from either a microinstruction or the memory data path (Paragraph 5.5.4). Other DPE5 logic for the RAM file is the E308 flip-flops at lower center that hold the numbers of the current and previous fast memory blocks as given by the program, and the upper right flip-flops that hold the DPE copy of the right 10 VMA bits. The loading of both VMA and its partial copy is produced through the gate at A4 when the microcode gives a memory function that requests it or initiates a cache sweep.

The ALU at the left on DPE6 can generate numerous functions but is used principally to add the least significant four bits of the number field to the instruction AC field to generate addresses for the block of accumulators used in extend instructions. The rest of the logic is mixers for selecting the RAM file address according to the source specified by the microword RAMADR field as given by the table at the lower left. The generation of the address is logically in three parts corresponding to the three rows of mixers. The bottom row selects the obvious source for the least significant four bits directly according to the microword field. The middle row selects those three bits that for fast memory references correspond to the block designation. This requires an extra mixer at the left through which address bits 04 and 02 select other functions to make the address selection. The obvious selection is made for a cache, VMA or number reference, or the current block for an accumulator; however an index reference may be to either the current or previous block, and substitution of an AC reference for memory may also be to either block. An address selection code less than 4 always means fast memory, so a 0 in the 04 microword bit disables the top row of mixers altogether. Codes 6 and 7 make the standard selection, but again the source for use of the RAM file for a virtual reference depends on whether it is an AC or cache reference: for the former the mixers put out all 0s, but for the latter they combine two VMA bits with a 1 in the most significant position, as the cache occupies the top half of the RAM file.

5.4.4 Ten-Bit Logic

This logic is a small scale arithmetic unit controlled by the microword number field in the same way that the AD and other fields control the 2901s. Of course those other fields always control the AU, whereas the 10-bit logic is manipulated by the number field only when that field is not being used for something else. This smaller arithmetic unit performs computations on exponents, counts steps in shift and arithmetic operations, and manipulates 7-bit bytes with zero alignment, which can therefore be handled much more efficiently than other sizes.

The 10-bit logic comprises the two sets of mixers and adder on DPM3 and the carry skipper and SC and FE registers at the bottom on DPM4. The adder is made up of ALUs, but these are limited to the seven functions listed in the table at the upper left because selection is made by only three bits. The SCAD outputs are available to the two registers, which are themselves inputs to the adder via the mixers. SCAD also goes to the main data path in both byte and exponent positions via DPM. Both rows of mixers on DPM3 handle 10-bit quantities, but the lower one requires eight inputs for only seven positions, and bits 0, 8 and 9 are handled by the 4 × 2 mixer at the left end. Most of the inputs to both sets are from DP, but they involve different parts of DP for different purposes. The upper set can receive FE, the exponent part of a word from DP always in positive form via the XOR gates at the left, the effective number of shifts in a shift or rotate instruction, and the size part of a byte pointer. The lower adder can receive SC, the right ten bits of the number field, octal 44 for generating an initial byte pointer, and a 7-bit byte from any position. Note that the inputs for 44 also receive DP 06 at the right mixer so as not to disrupt the size field when a position field is inserted in a byte pointer.

5.4.5 Program Flags

DPE9 shows the program flags and the multitude of gates through which they are set and cleared. There are essentially two ways in which the flags are manipulated: by conditions resulting from arithmetic and other operations in the hardware, and direct manipulations by the microcode for saving and restoring or, for example, setting the First Part Done flag for later control of its own activities. Direct microcode control and ordinary carry-overflow testing is via a single special function with selection by the number field as listed at the upper left on the print. Individual special functions take care of ASH and exponent testing so the same microinstruction can use the number field for the 10-bit logic. Hardware conditions come into play on the selection of various tests by the microcode; the large number of such conditions is listed in detail with the discussion of the program flags in Paragraph 2.9 of the System Reference Manual (DEC-10-XSRMA-A-D).

There are however a few special considerations that should be mentioned. Because AU is 40 bits, the net at the upper right corner detects overflow from a discrepancy between DP SIGN and DP 00, and determines the presence of carry 1 by overflow being opposite carry 0 (which is available as CARRY OUT); these signals are derived from DP signals and are therefore valid only if the adder is doing an arithmetic function and its output is on DP. Note that the gate that detects overflow in arithmetic shifting (C7) checks for opposing states of DP bits 1 and 2 but these are actually bits 0 and 1 of the word being shifted. Decoding of trap signals from the trap flags at the lower right requires trap enables from both the processor and the console.

5.5 DATA PATH MEMORY

This data path is actually for both memory and I/O operations, and most of what is discussed here is therefore also related to communication over the bus. All I/O operations require use of the bus. But a requested memory function uses the bus only when a word must actually be transferred to or from a storage module; that is, memory functions use the bus except when a memory access turns out to be an AC reference or a cache hit, an attempted access results in a page failure, or the memory function is simply a write test (i.e., a check whether a page failure would result were a write function to be given). Of the many DPM signals whose names contain MEM or MEMORY, some really are for memory whereas others control both memory and I/O operations. This same ambiguity occurs in the microcode definitions. In an attempt to limit confusion in the test, the term "memory" will be used only to refer to memory, and "DPM" will be used in general circumstances applicable to both memory and I/O.

Every DPM function, whether memory or I/O and whether requiring the bus or not, is set up by a microinstruction with a 1 in bit 76 (physical bit 26). In line with the standard terminology, the bit is labeled MEM and the print signal from it is MEMORY FUNCTION. The setup information is supplied by the number field, where bits 0–11 select the type of memory cycle (i.e., read, write test, write)

and specify other associated characteristics; that is, user or executive space, virtual or physical reference, and so forth. Bits 12–17 perform more general control functions, such as starting the cycle and loading VMA. In particular there is a bit that can substitute bits from the data path for selecting the function type and characteristics; a 1 in this DP function bit causes the hardware to make the selection according to DP bits 0–13 instead of number bits 0–11. Setup for an I/O function must always be made from DP, because only DP can supply the bus command bits unique to an I/O function. Use of DP for a memory function is generally to remake a request following a page failure. For references in instructions, another of the general control bits can cause the selection of the memory cycle type to be made according to bits in the dispatch ROM in place of number bits.

Handling a memory or I/O function requires two microinstructions. The first sets up and starts the function, and the hardware associated with the bus then goes on independently of the microcode, requesting the bus, waiting for the grant, doing the command/address cycle, and even doing the data cycle if the function is read. Of course the hardware stops short of all this on a memory function that does not need the bus, but otherwise it ends either with the word read in MB or waiting to send a word on write. The second microinstruction the microcode gives is simply a wait. If the independent functions are already complete, there is no delay and the microcode just takes the word read or gives the word to be written (where the latter action triggers the data cycle). If the independent hardware functions are not finished, the processor enters a microcode delay until they are.

Figure 5-21 is a block diagram of the memory data path, with some necessary simplification and omissions.

5.5.1 Memory and I/O Setup

The hardware for setting up a memory or I/O operation is principally on DPM5 and the upper two-thirds of DPM4; it appears at the bottom and at the left in Figure 5-21. Across the center of DPM4 is the VMA register, which is loaded by the first in the pair of microinstructions that must be given to start and complete a DPM function (memory or I/O). Included in the leftmost chip of the VMA are two flags, one indicating that a sweep is being done, and the other that VMA is extended. A sweep is simply invalidation of the entire contents of the page table or cache directory, and it automatically sets the top two E214 flip-flops, which would otherwise be duplicates of VMA 18 and 27; this mechanism speeds up a sweep by allowing it to handle two table or directory locations at a time. The enable for VMA is produced by the sweep set as well as by the DPM function conditions (DPE5 bottom center). VMA EXTENDED allows VMA 14–17 to be sent over the bus either for use in a physical address or for a subsystem number in an I/O operation.

The other flags associated with VMA are in three categories, two of which are at the top of DPM4 and the third at the upper left on DPM5. The four E511 flags (EPM4 D4) can be set up only from DP and are for specifying those characteristics unique to an I/O function. The four at the left in E508 are for an instruction fetch and specifying several address characteristics (logically VMA EXTENDED should be regarded as in this group). Note that when a memory function is selected from DP, user space and previous context are selected directly by bits 0 and 9 as this is generally to redo a previously-defined function following page failure. The original selection however is dependent on a number of conditions. In particular, note that when the User Flag is on, user space is always selected for an instruction fetch; and it is selected for other memory functions unless executive space is being forced or the processor is executing an instruction supplied from the console. Selection of the previous context (which can also force user space in executive mode) is handled by the mixer and flip-flops at the far right on DPMA according to both bits 9–11 of the number field and the selection made by the AC field of the instruction. The remaining VMA flags at the upper left on DPM5 are for inhibiting the cache and selecting the type of cycle, where the three flags for the latter may be set from dispatch ROM bits as well as from bits of the number field or DP. Note that all flags on DPM4 are set up when VMA is loaded, but those for the cache and cycle type are enabled by MEM EN, which comes from the net at B3 and indicates that a DPM function is actually being started. This is so the cycle type can

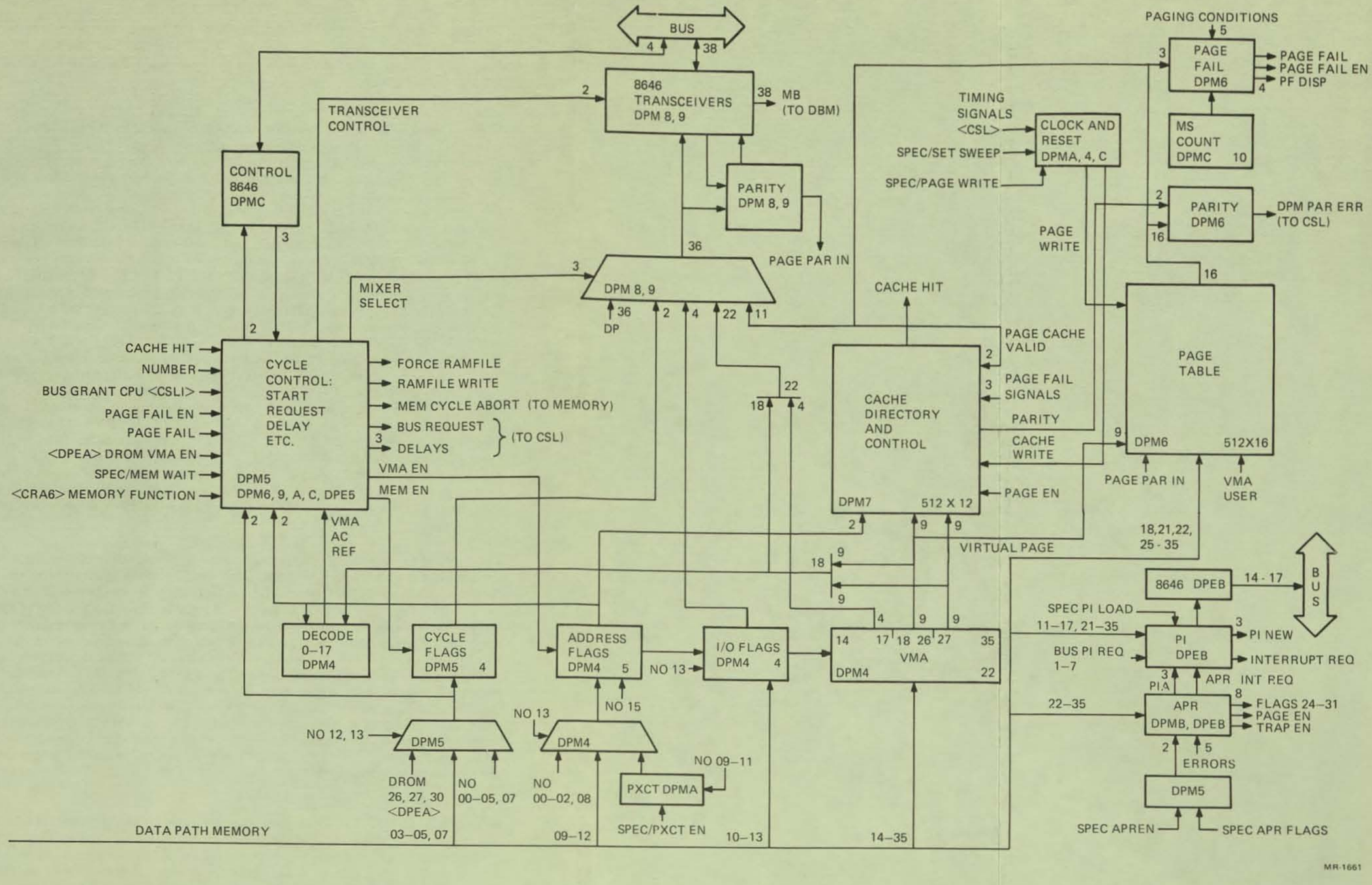


Figure 5-21 Data Path Memory

be changed without disturbing the address, as for a write following a read or write test. The easiest way to understand the role the flags play in a DPM function and how the function is selected is to read Table 5-3, which explains the use of the number field and DP bits when a microinstruction gives the DPM function.

Once a function has been set up, operations are handled by the two sets of flip-flops in the lower half on DPM5. Note that these flip-flops are triggered directly by the T clock rather than the processor cycle clock so they can be manipulated independently of the microcode. To synchronize initial setup with the microinstruction that calls the function, the logic makes use of sync signals that are equivalent to the enable for the processor clock (refer to the clock circuit at the left on DPMA). The first microinstruction in the required pair performs several operations besides setting up VMA and the flags. If it loads VMA, it also sets VMA JUST LOADED at the lower right corner on DPM5; this flip-flop remains on for just one clock tick, and it prevents the logic from taking action on conditions generated spuriously by state transitions during setup. A write test does not actually make use of a real DPM cycle. However MEM EN produces START CYCLE at B2 for either a read or a write but not if a read-pause-write cycle is already in progress. (The logic includes provision for read-pause-write but it is never used, as the microcode makes only separate read and write requests.) START CYCLE sets the appropriate delay enable in E205, makes a bus request via the top flip-flop in E306 at the left, and sets MEMORY CYCLE in E405 at the left on DPM6 (MEM WAIT comes from MEM EN). From this point the hardware works independently of the microcode. When the console arbitrator grants the bus, the request is dropped and the bus operations are performed as explained in Paragraph 5.5.2. Of course even when START CYCLE is given, there may be no actual bus operation: conditions such as an AC reference, a cache hit, a page failure, an interrupt, or a timeout of the millisecond count – any of these may kill the bus request (via STOP MAIN MEMORY at D2) and the delay enables. The second microinstruction may regenerate MEM EN from the number field or give a special memory wait function. Either produces MEM WAIT (C3) to clear MEMORY CYCLE, and produces MEMORY DELAY to start the read or write delay if the corresponding enable is still on. A delay for either function stops the processor clock at the console board until the bus operation is completed. Note that the sync does not enter this logic – MEMORY DELAY comes on immediately (see DPMA D1).

The way the hardware and the microcode resynchronize depends on the kind of function and when the second microinstruction is given. For a read function the hardware does the command/address cycle and waits for the response. The response may be an identification for a who-are-you cycle, a word from memory, or a word from an I/O register. Only the first two of these are necessarily completed in one use of the bus: for a UBA I/O read the bus will have been freed, and the processor waits until the adapter gets the bus to do an I/O data cycle. In any event when the word comes over the bus it is loaded into MB and the delay enable is killed. If the second microinstruction has already been given, the delay ends and MB is read. Otherwise the hardware waits and the second microinstruction reads MB without delay. On a write the bus grant kills the delay enable as the command/address cycle begins. If the second microinstruction has already been given, the delay terminates and the word is sent immediately. Otherwise the hardware waits, and when the second microinstruction does come the word is sent without delay. Note however that there is no provision for holding the bus beyond three cycles during an I/O function. Hence for an I/O write the microcode must give the second microinstruction immediately.

For a virtual reference in which the in-section part of VMA (bits 18–35) is in the range 0–17, the net at the upper right corner on DPM4 indicates an AC reference. During the second microinstruction, this causes selection of the appropriate source for the RAM file address as explained in Paragraph 5.4.3; for a read it forces selection of the RAM file in place of MB at DBM via the gate at DPM5 D2; and for a write it produces the RAM file write signal at DPMA D6.

Table 5-3 Selection of Memory and I/O Functions

Bit	Number Field	DP
0	Force user mode	User mode
1	Force executive mode	
2	Instruction fetch	Instruction fetch
3	Read cycle	Read
4	Write test	Write test
5	Write cycle	Write cycle
6		
7	Inhibit cache	Inhibit cache
8	Physical reference	Physical reference
9	Previous context mode	Previous context
10	Previous context mode	I/O function
11	Previous context mode	WRU (who are you?) cycle
12	AREAD – select function according to DROM bits 26, 27, and 30 in place of number field bits 3, 4, and 5; load VMA if DROM bit 29 is 1 (without disabling No. 14); ignore No. 07	Vector cycle
13	DP function – ignore No. 00–11 and select function according to DP 00–13 (note: No. 12 must be 0)	Output byte cycle
14	Load VMA and VMA flags from DP	
15	Extend VMA – put VMA 14–17 on bus in command/address cycle	
16	Start cycle, or start wait (i.e., synchronize microcode to bus operation)	
17	Start cycle if DROM bit 28 (COND FUNC) is 1	

The console single step switch being on prevents the processor from holding the bus from the first to the second DPM microinstruction. When SS MODE is true, read and write cannot be enabled together (read-pause-write is split into two separate functions). START CYCLE for write sets up the whole operation, but instead of setting BUS REQUEST it sets DLYD WRITE REQ just below it. Then when MEMORY DELAY comes on, the bus request is made and the entire operation takes place in a single microcode step. Note that in case the switch goes off between the two microinstructions, the fact that a single step cycle is in progress is remembered by the second E405 flip-flop on DPM6.

5.5.2 Bus Operation

The bus grant from the console arrives at the processor at the upper left corner on DPMC. If FAST ABORT is false, indicating the processor has not determined that the function should be voided or the bus is not needed (C7), the grant asserts COM/ADR EN. This signal is applied to the top flip-flop in the COM/ADR counter just at the right and the top transceiver at B2, and through the net at DPMA A2 it supplied the transmitter enable for the bus data transceivers on DPM8,9. Hence the next T clock counts the first bus cycle, puts the command/address control signal on the bus, and since BUS REQUEST is still on at this time, it loads the command/address information into the transmitter flip-flops through the mixers below the transceivers. At the same time it also clears BUS REQUEST. If VMA is extended, VMA bits 14–17 are included in the address; and if the function is a virtual memory reference, address bits 16–26 are supplied by the page table instead of VMA. Note that for bits 16–26, the parity generators get the mixer outputs directly; this is to compensate for paging time. If the processor belatedly determines during the command/address cycle that the function should be voided or the bus is not needed, STOP MAIN MEMORY comes on (DPM5 D2); this produces MEM CYCLE ABORT (DPMC C7) to shut down the storage cycle in the memory subsystem and disable the transmit logic (DPMA A2) to prevent any further attempt to send information over the bus.

The next T clock clears the transmitters and sets the appropriate flag at DPMC D3 to identify the cycle as memory or I/O. For a write function the generation of MEMORY DELAY enables the transmit circuit (DPMA A2) so the processor cycle clock in the second microinstruction transmits the word held on DP. At the same time WRITE CYCLE indicates a bus data cycle through the control signal transceiver chip at the lower right on DPMC. For a read, every R clock temporarily latches the receivers via the gate at the bottom of the clock circuitry on DPMA, but the termination of the read delay enable holds the latch. The strobe that ends the enable for a memory transfer or I/O instruction (DPM5) comes from the bus signal for a data cycle or I/O data cycle via the control 8646.

The remaining flip-flops on the COM/ADR counter are for special situations. The second T clock sets COM/ADR + 1, which sets up the write transfer for a single step cycle. For a WRU cycle the adapter identification must be sent back within the allotted three cycles, and the T clock following the setting of COM/ADR +2 terminates the read delay enable. For any bus memory cycle the same T clock sets the nonexistent memory error flag at DPMC D3 if the memory has not yet returned MEM BUSY.

5.5.3 Paging

Paging information, including address space, page use bits and physical page number (for 1024K of memory), is available for each virtual reference from the page table at the top on DPM6. The table is kept in pairs of 256 × 4 RAMs whose locations are selected by the virtual page numbers from VMA. So long as PAGE EN is set (DPMB A6), the net at the lower right on DPM9 indicates a paged reference whenever the microcode indicates the address for a memory function is virtual. When an addressed location in the table does not contain a mapping appropriate to the reference being made, the microcode refill procedure loads the desired mapping from DP 1, 21, 22, 25–35 and the User flag. Writing in the page table is handled as a special function via the decoder at DPMA 2A; the page write signal at D6 is produced via the flip-flop at DPMC B3. Each table entry includes an odd parity bit, where the parity for the DP bits is supplied by the same chip that generates parity for bus transmission on DPM9 (note that PAGE WRITE EN cuts out DP bits 19, 20, 23 and 24). Parity checking of the paging information is made by the circuits at the lower right on DPM6.

In each pair of RAMs the left is enabled by a 0 in VMA 18, and the right is enabled by a 1 in that bit or by a sweep function. Thus to invalidate the entire table, the microcode gives both the sweep and page write special functions (DPMA) with a 0 in DP 18, and invalidates two locations at a time by running through all configurations of VMA 19–26 keeping a 0 in VMA 18.

The logic at the lower left on DPM6 detects a page failure. But note that via the bottom two flip-flops and the E404 gate, certain interrupts and errors are handled as page failures. In a virtual memory read – all I/O is physical – if there is either an interrupt request or an MSEC count timeout when MEMORY CYCLE is set, INT OR ERR is asserted. The various conditions – interrupt, error or real page failure – produce STOP MAIN MEMORY and FAST ABORT to kill the bus request or the function, and they are encoded into a set of four signals on which the microcode can dispatch to handle the situation. Interrupt and errors have precedence, and the priority encoder ensures indication of at most one real page fail condition, and then only when paging is enabled on a virtual non-AC reference. Any condition produces the page fail enable, which holds up the processor clock via the top delay gate (lower right, DPM5) when the second DPM microinstruction is given. During the delay the console transfers control to the microcode page fail handler. The conditions indicated by the various configurations of the dispatch bits, which are available as DP 18–21 via DBM, are as follows.

PF DISP 10-01	Condition
0000	Interrupt or timeout
0010	Bad data
0100	Nonexistent memory
1000	Not writable on write test
1010	Mapping not valid
1011	Wrong address space

Note that the order of the real page fail conditions in terms of dispatch numbers is not the same as their priority order; namely the write test condition has lower priority than the other two.

5.5.4 Cache

The cache holds one memory word for each configuration of VMA bits 27–35, for a total complement of 512 words. The cache directory also contains 512 locations selected by VMA 27–35, but here the information in each location identifies the virtual page and address space of the word contained in the corresponding cache location. The structure of the cache, which occupies the top half of the RAM file, and the way it is addressed are discussed with the RAM file in Paragraph 5.4.3.

Whenever the processor actually writes a word in or reads a word from main memory, it generates both RAM FILE WRITE and CACHE WRITE through the gates at the top left on DPMA. The first of these signals writes the word in the cache. CACHE WRITE however writes in the corresponding location of the directory, which comprises the three pairs of 256 × 4 RAMs on DPM7. The information written is the virtual page number part of VMA (bits 18–27), the user flag to indicate the address space, an odd parity bit, and the inverse of VMA PHYSICAL, which serves as a valid bit. Hence the information in the directory is valid only when the word written in the cache results from a virtual reference. The cache is written on a physical reference, but no later use can be made of the data.

With the cache enabled from the console, the contents of the directory location selected by VMA 27–35 are regularly compared with the corresponding information currently in those elements that initially supply the directory entry (but note that CACHE VALID is simply compared with a 1 since the entry must be valid to be of any use). If the two quantities are identical and all of the other inputs to the large AND gate at the upper right are true, a cache hit is indicated. The necessary conditions are that the processor is making a virtual non-AC memory read reference, that paging is enabled and there is no failure or error, that the microcode is not inhibiting the cache, and that the page is cacheable and has a valid mapping. Except for the source of the RAM file address, a cache hit acts just like an AC read reference as described at the end of Paragraph 5.5.1. Detection of even parity in a directory entry stops the clock via the same signal used by the page table (DPM6 B1).

To invalidate the cache directory the microcode gives the special sweep function, which generates CACHE WRITE. The combination of the sweep and a 0 in VMA 27 enables both RAMs in each pair, so by having VMA PHYSICAL set, the microcode can invalidate the entire directory two locations at a time by running through the VMA 28–35 configurations. There is no special function for CACHE WRITE, as it is expected the cache will be swept whenever the page table is swept. The microcode can sweep just the cache, however, and does so whenever it invalidates even a single page table entry.

5.5.5 Error Logic

At the lower left on DPMC is a 10-bit counter, which is driven by a 4.096 MHz clock and therefore overflows every millisecond. If enabled from the console, overflow sets the 1 MSEC flip-flop in E502, which in turn sets 1MS at the left on DPM6 to cause a page failure at the next virtual memory read reference.

Failure of a memory to respond to a request within three bus cycles sets NXM ERR at the upper right on DPMC, and the flag just below it is set if the memory returns bad data as indicated through the control 8646. Either of these flags being set causes a page failure through the logic at the lower left on DPM6 and also sets a corresponding APR flag on DPMB. Other APR flags are set by an interrupt from the console, an indication over the bus that AC power is failing, or that a read error has occurred in memory but memory control was able to send corrected data. The setting of any APR flag can request an APR interrupt if the program has set the corresponding enable in the APR register at the bottom of the print.

Both the APR flags and their enables are controlled by the program via bits on DP. Clock signals for the flags and the register are provided by special functions via the bottom two E306 flip-flops at the lower left on DPM5. Besides the flag enables, the APR register includes flags through which the monitor enables trapping and paging, and a flag that allows the microcode to trigger an APR interrupt request directly. Moreover part of the register, containing the PI assignment and a copy of TRAP EN, is on DPEB. The trap and page enable flags and all of the APR flags are available to the microcode via the right half of DBM (in the same set of inputs that includes the page fail dispatch code and the APR interrupt request signal). The APR register cannot be read, but the microcode keeps a copy of it in the left half of the FLG location in the register file.

5.5.6 Priority Interrupt

Almost all of the PI logic is on DPEB. By means of the three sets of flip-flops at the bottom, the microcode via DP can select which levels are active, make soft (i.e., program-initiated) interrupt requests, turn the system on and off, and specify on which levels interrupts are currently being held. A UBA or the processor APR logic can request an interrupt on its assigned level by placing a signal on the appropriate line of the seven in the PI request bus. These bus lines are input at the upper left to flip-flops through which the cycle clock synchronizes the request to the microcode. Through the AND and OR gates just at the right of the request flip-flops, the logic automatically recognizes any soft request but recognizes only those hard requests made on active levels. Both the recognized requests and the signals for current interrupts are applied to priority encoders, which indicate the highest priority new request and current interrupt, but note that the request encoder is enabled only if the PI system is on.

If a new request has priority over all the current interrupts, the compare circuit at D3 generates an interrupt enable, which in turn produces an interrupt request for the microcode through the top flip-flop at the upper left. When the microcode responds to the request it can read the new level through the D bus mixer as bits 19–21 of the same word that contains the 10-bit VMA and program flags. The state of the system is kept at all times in the PI location in the register file. From it and the new PI level, the microcode can make up a new current configuration, and the new level is then available as the output of the current priority encoder.

The microcode then manipulates the DPM function logic to do a WRU cycle to determine the source of the request. When the bus is granted, the gate at DPMC B7 enables PI transmission during the WRU command/address cycle. PI XMIT EN places the number of the current level on bus data lines 15–17 through the single 8646 at the upper right on DPEB. The exclusive OR gates that feed line 14 produce even parity for this set of four lines so as not to change the parity for the left half of the bus on DPM8.

The flip-flops at the lower right are part of the APR register and contain the APR PI assignment. When an APR flag requests an interrupt, the decoder asserts the PI request line corresponding to the assigned level.

5.6 MEMORY

Figure 5-22 is a block diagram of the memory subsystem, including the memory controller, the memory bus, and the array boards. The diagram shows the signal connections among the various elements of the logic, shows the way the bus interfaces the controller to the array boards, and indicates which print contains which part of the logic. The text in the following paragraphs is not geared specifically to the block diagram, but rather to a flowchart of the events that implement access to memory from the KS10 bus. Before going into the flow, however, there are two parts of the hardware that require separate discussion: the configuration of the memory array, and the procedure for error detection and correction.

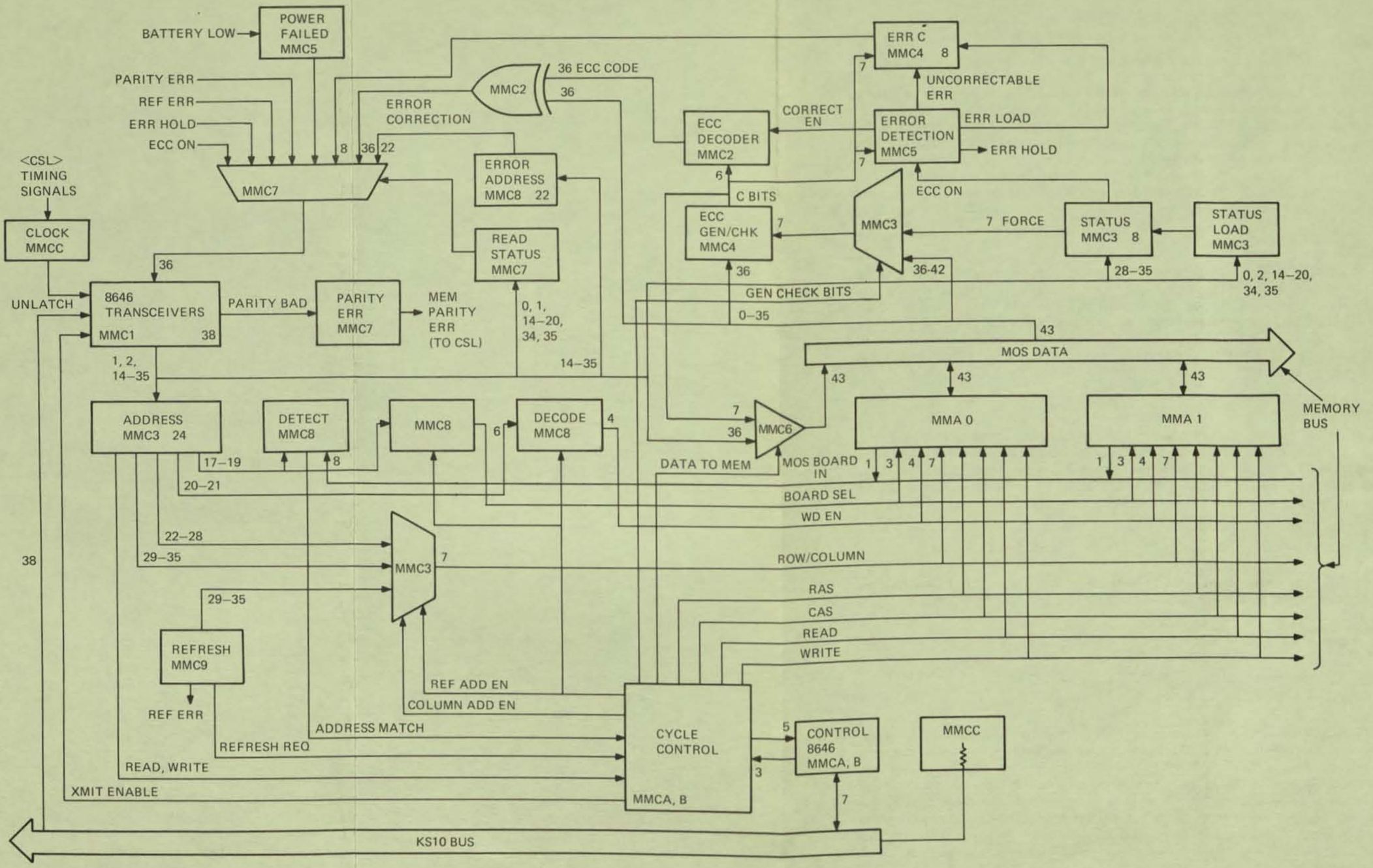


Figure 5-22 Memory Subsystem

Each 36-bit data word from the KS10 bus is stored in main memory as a 43-bit word including a 7-bit error correction code. The entire memory array is made up of two to eight MMA boards, each containing 64K words organized in four word groups. A group comprises 43 RAM chips, where a single chip contains one bit from each of the 16K words in the group. The organization of the prints showing the array is as follows (the memory bus data connections are with the RAMs for words groups 0 and 1).

Print	Word Groups	Bits
MMA3	0,1	Right odd (23–41)
MMA4	2,3	Right odd (23–41)
MMA5	0,1	Left odd (1–21)
MMA6	2,3	Left odd (1–21)
MMA7	0,1	Right even (22–42)
MMA8	2,3	Right even (22–42)
MMA9	0,1	Left even (0–20)
MMAA	2,3	Left even (0–20)

Selection of a single word in one set of 43 chips is made by the memory address (bits 14–35) in the word supplied to the controller over the KS10 bus during a command/address cycle. The roles the different parts of the address play in making the selections are as follows.

Bits	Select
14–16	Controller (at present all bits must be zero to select the single controller)
17–19	MMA board
20–21	Word group (one of four sets of chips) on board
22–28	Row in chips
29–35	Column in chips

The three bits for board selection are supplied by six lines, high and low, and are decoded on the board; the selected board returns a signal indicating that it is present. The controller decodes the two bits for the word group and supplies four select lines for the individual groups. The row and column addresses are supplied on the same set of seven lines and are applied to the same set of chip inputs – the two quantities are distinguished by an accompanying row or column address strobe. For writing, the controller places the data on the memory bus and sends the write signal to the chips; for reading, it just sends a read signal that places the selected board output on the data lines. The first two MMA prints show the logic for distributing the control signals to the different word groups.

Since there are fewer than 64 bits in a memory word, including both data and error correction code, only six bits are needed to correct single errors (i.e., to identify a single bit that is in error), but seven are used so that double errors can be detected as well. The logic that generates the code, both for writing with data and for handling errors in words read, is on MMC4. The error correcting code, on both write and read, is made up of the seven C bits. Each bit is produced by a pair of parity chips, which receive as inputs some set of data bits and a corresponding check bit. On write, the signal GEN CHECK BITS is true, so the check bits are equivalent to the "force" bits through the mixers at the lower left. The force bits are supplied as status by the program and are always 0 in normal operation. The C bits generated on write are stored with the data word as MOS Data Bits 36–42, and it is these extra data bits that become the check bits for generating the correction code on read. Consider a single code bit such as C4, which is generated by parity chips E719 and E710. On write, check bit C4 is 0, so if the set of data bits applied to E719 and E710 has even parity, code bit C4 is 0. This code bit is stored as MOS data bit 40, which becomes check bit C4 on read. Hence on read the check bit is still 0, and C4 of the read code is therefore also still 0. If the parity on write is odd, C4 is 1; but this places a 1 in the check bit on read, thus changing the read parity to even and again producing a 0 for C4 in the read code. This means that a single data bit changing state from write to read produces 1s in any read code bits to whose nets it is applied.

To understand how the above characteristics implement error detection and correction, it is necessary to know the way the data bits are arranged in relation to the code bits. First, each data bit serves as input to the nets for an odd number of code bits, and each such set of code bits is unique – no two data bits are associated with the same set of code bits. Moreover, if two such sets overlap, they overlap in two places; in other words, the set of code bits associated with a given pair of data bits either have no bits in common or have two bits in common. Since a change in a data bit changes the associated code bits from 0s to 1s, this means that a single error produces 1s in an odd number of code bits whereas a double error produces 1s in an even number. The error logic is at the right on MMC5, where any code bit being 1 indicates a read error, and odd parity for the entire code indicates that the error is correctable. The correction logic is on MMC2, and comprehending it requires that the reader be aware of one more characteristic of the generating bit arrangement: namely, that although all seven code bits are used in error detection, the data bits are arranged so that each possible error position corresponds to a unique configuration of the six numbered code bits, and only they are therefore needed for identifying the bit in error. Consequently when an error is correctable, the decoders at the left and bottom on MMC2 decode the six numbered bits to generate a single signal corresponding to the data bit in error. Without errors the exclusive OR gates pass the MOS data signals, but the assertion of a single ECC code signal changes the output of the corresponding gate to a state opposite the applied data bit.

Note on MMC5 that occurrence of error is always indicated, but error correction is under control of the program through ECC ON, which is supplied by write status. The "force" signals are so named because the program can force error indications by supplying 1s to them as status. When a check bit is 1 on write, the corresponding code bit is still 0 or 1 respectively for even or odd parity, but since it then replaces a 1 when read back, that bit of the read code will also be 1.

5.6.1 Data Access

All of the events involved in moving data between the KS10 bus and the memory array appear in a single flowchart (MMCF1) in the Field Maintenance Print Set. Any data sequence begins with the command/address cycle and then continues to either read or write. From read, the sequence either terminates or goes through the special path to cross over to write. Relationships among the major signals relevant to read and write respectively appear in timing diagrams, Figures 5-23 and 5-24. Both include the command/address cycle.

At most points where memory control places a word on the KS10 bus or reads a word from the bus, whether in data or status operations, it checks the output of the receiver parity net. Bad parity sets the flag at the upper right on MMC7, both for a parity error status indication and to signal the console to shut down the processor clock (see CSL3).

5.6.1.1 Command/Address – To gain access to memory to read or write data, a subsystem must become bus master and send command/address information to memory control with a 0 on bus data line 0. Events in memory control begin at the trailing edge of the R clock with the loading of the address and the read and write bits into the address register at the bottom on MMC3. If MMC is not presently holding information about some previous error, the address is also loaded into the error address register on MMC8 so that it will already be saved should an error occur in the upcoming cycle. At the end of the command/address cycle, memory control effectively enters its own independ ent sequence of operation by setting the CA cycle seen flag on MMCA, provided no belated abort signal has arrived from the master. Simultaneously the board and word select parts of the address are sent from the address register out on the memory bus by the gates on MMC8, and the row part has been supplied by the mixers at the top on MMC3. If the selected board is present, the return signal at the upper left on MMC8 produces an address match, allowing the sequence to continue.

Operations with the selected board begin at the next R clock with the setting of RAS, which sends the row address strobe. MMC then sends MEM BUSY to the console to hold the KS10 bus, switches the MMC3 mixers over to the column address, and sets CAS for the column address strobe. The next two T clocks set T4 and T5, but these timing signals are relevant only to read.

5.6.1.2 Read – Events specifically for read begin at 337.5 ns on the read path with the generation of the read signal to the array board. Following the setting of T4, memory control sets up transmission over the KS10 bus, and loads the data onto the bus at the same time it is being run through the error checking circuits. Unless there is error information already being held, the ECC code is saved at 862.5 in case it is needed. If the data is correct, the strobe is enabled and the data is held on the bus for a second cycle, which is the official bus data cycle. If the data is bad, the enabling of the strobe is held off for a cycle so that data can be on the bus for two cycles following the initial transmission of bad data. The data sent in the final two cycles may be corrected, or it may be simply the same bad data with an uncorrectable error indication. In any event, the final cycle in which data is on the bus is the

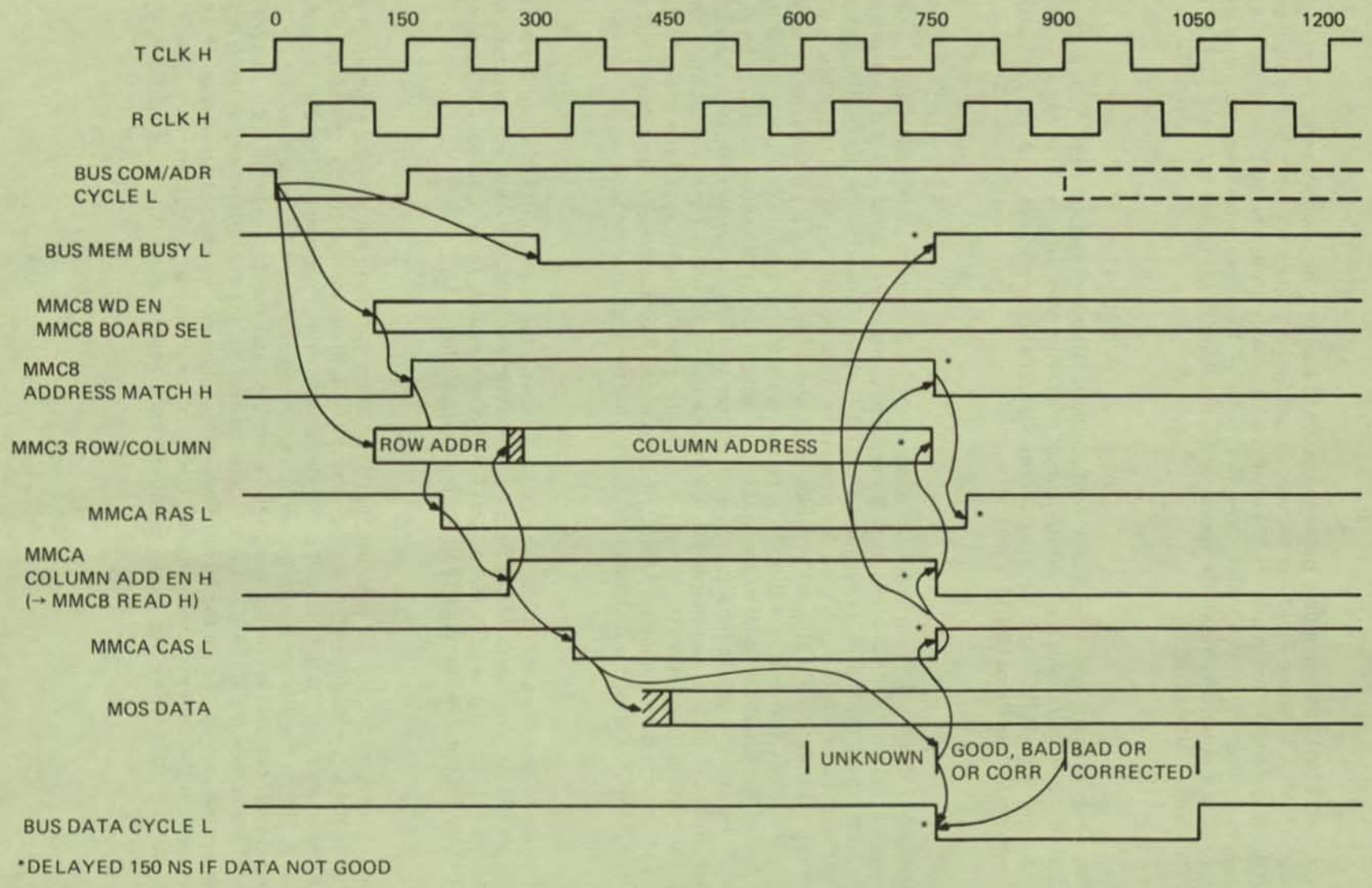


Figure 5-23 Memory Read, Timing Diagram

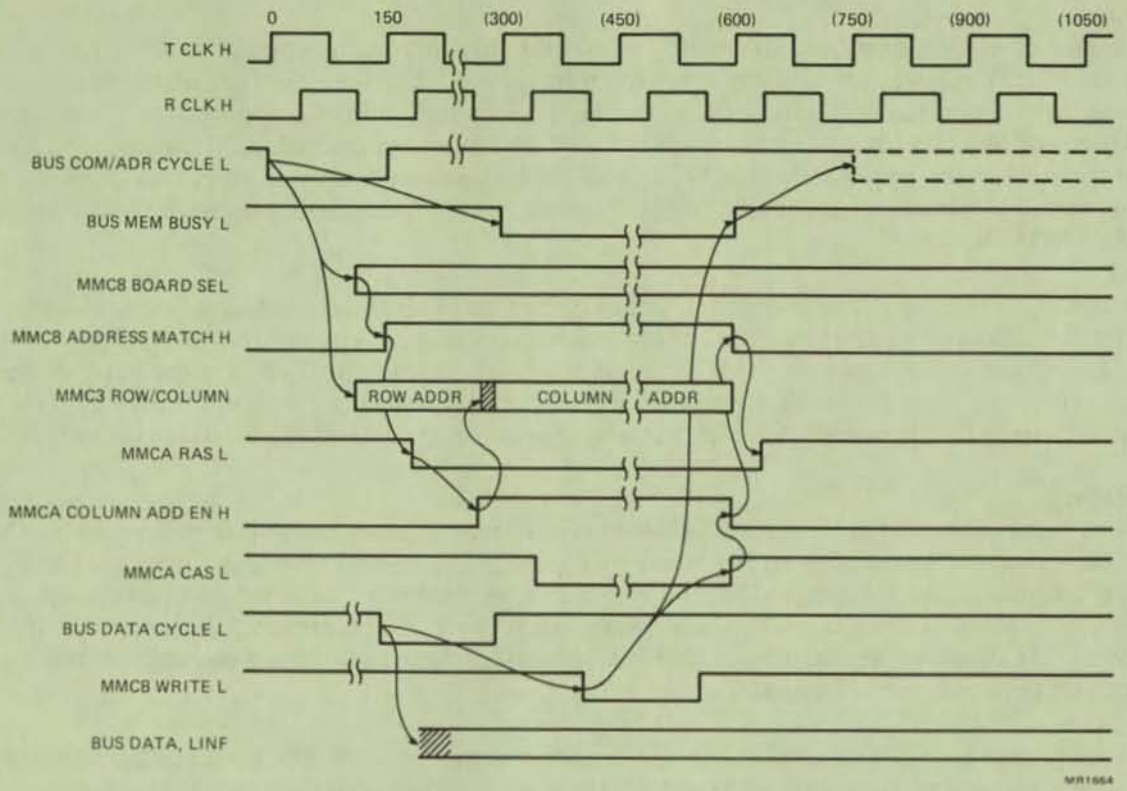


Figure 5-24 Memory Write, Timing Diagram

one labeled bus data cycle. A read error sets the error hold flag to lock the error information in the status register; hence it continues to identify the error, even if there are further data operations, until a write status clears it. Also at 900/1050 MMC begins clearing the control logic and drops MEM BUSY, so the bus arbitrator can decide who shall be master next while the memory data is still on the bus. The final cycles used for clearing out the rest of the control logic may actually overlap a new command/address cycle to begin another memory access.

5.6.1.3 Write – During the command/address part of the flow for write, MMC enables the generation of the check bits and asserts DATA TO MEM to enable the drivers for the memory bus data lines on MMC6. Although memory control disables the bus arbitrator and sends the row and column addresses to the array to select the single location, it waits for the data from the master before performing any further operations specifically for write. When the data does arrive it is latched, and MMC sends signals to the array to cause it to be written into the selected location. The rest of the sequence clears the control logic and frees the bus arbitrator.

5.6.1.4 Read-Pause-Write – In the command/address cycle, the combination of both READ and WRITE generates PAUSE. The sequence executes in the same way as read until the bus data cycle, at which time the existence of the write condition prevents the standard clearing operations, and flow continues into the pause path instead of terminating. Provided good data has been transmitted, the sequence follows the far right path to set up a write function, clear the bus, and then wait for the data from the master just as at the beginning of a normal write. However, if bad data has been transmitted, the initialization of the pause sequence simulates the write clear condition, which clears out the logic and frees the bus, using part of the normal read termination.

5.6.2 Status

The sequence of events employed in reading or writing memory status appears at the right on flowchart MMCF2. (Status register bit format is shown in Figure 5-25.) The master initiates the sequence by transmitting a command/address cycle for an I/O operation, with appropriate coding of the address bits, and selecting read or write by bits 1 and 2 as in a data operation. The command word is decoded for read at the upper left on MMC7. The read operation simply enables transmission, gates the status through the output mixers on MMC7 to load the bus, and indicates an I/O data cycle via the 8646 on MMCB.

The command for writing status is decoded at left center on MMC3 to enable loading of the force bit register on the right and enabling control of the other status flags. Memory control signals I/O BUSY over the bus at the lower right on MMCB, loads the force register, and sets up the various flags as indicated. The data must arrive in the second or third cycle, but note that a 0 on line 12 clears the power failed flag in the second cycle no matter what the configuration of status sent in the third cycle.

5.6.3 Refresh

The flow of events that occurs in refreshing the memory array appears at the left on flowchart MMC2, and the timing of the major signals is shown in Figure 5-26. In this sequence, the generation of a single row address strobe refreshes all of the words in a given row in all word groups throughout the entire memory array. Thus no matter how many array boards are present, the entire memory is refreshed by 128 strobes given to all possible row addresses. Since refreshing is required every 2 ms, the logic is set up to refresh a single row about every 15 μs.

Timing is governed by the T clock through the counters at the top on MMC9, where the left two provide the row address and the right two count off the time between refresh cycles. At each refresh, the logic subtracts 1 from the address and sets the refresh count to 99. After it counts down to zero, the hundredth T clock generates a refresh request, which causes a refresh cycle to be executed by the control logic on MMCA and MMCB as soon as it completes the current data operation, if any. The request sets REF ADD EN on MMCB C3, and this signal both selects all boards and word groups through the logic on MMC8 (by making all select lines high) and supplies the refresh address as the row address through the mixers at the top on MMC3. The next T clock sets REF SET RAS, which initiates a sequence of row address strobe, T4 and T5 to do the refresh. T5 indicates completion of the operation, and the remaining clocks clear out the logic. While the sequence is executing, including any wait for the completion of a data operation, the refresh countdown continues. If the request is still up when the count reaches 31, a refresh error is indicated by the lower flag on MMC9. Since the most likely reason for the delay is a memory hangup waiting for write data, the refresh error enters the write sequence (flowchart MMCF1) to clear the control logic by simulating the completion of a write. No memory data is lost.

5.6.4 Power Requirements

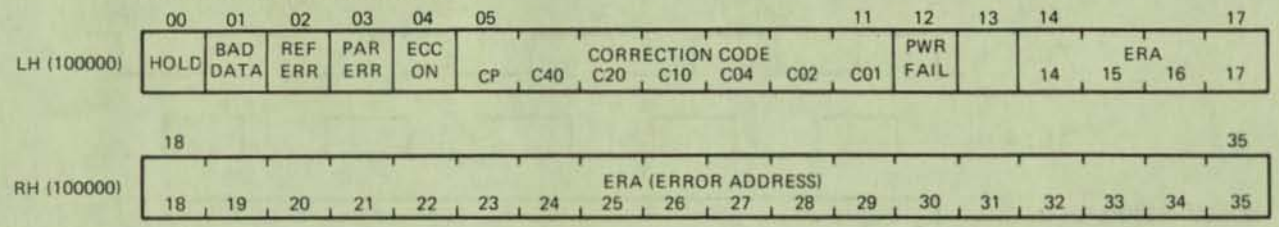
The memory power connections are listed along with the capacitors on MMAB and MMAC. On the first of these drawings, the +5 V connections with battery backup are listed at the top of the left column; those without battery backup are in the second column. The center column lists the -5 V connections on the left and the ground connections on the right. The +12 V connections are at the top of the first column on MMAC.

Single loading is as follows.

Data Input Drive Current

High	20 μA maximum
Low	-400 μA maximum

MEMORY STATUS REGISTER



	BIT(S)	FUNCTION
*	00	ERROR HOLD (ERROR CONDITION DETECTED BY CONTROLLER)
	01	BAD DATA (UNCORRECTABLE READ ERROR)
*	02	REFRESH ERROR (INCOMPLETE WRITE CYCLE)
**	03	BUS PARITY ERROR DETECTED
	04	ERROR CORRECTION ENABLED
	05–11	CORRECTION CODE BITS
*	12	POWER FAIL (LOSS OF POWER OR BATTERY BACKUP LOW)
	14–35	ERROR ADDRESS

NOTES:
*WRITING A 1 BIT CLEARS THE FLAG.
**WRITING A 1 BIT SETS THE FLAG, WRITING A 0 BIT CLEARS THE FLAG.

MR1665

Figure 5-25 Memory Status

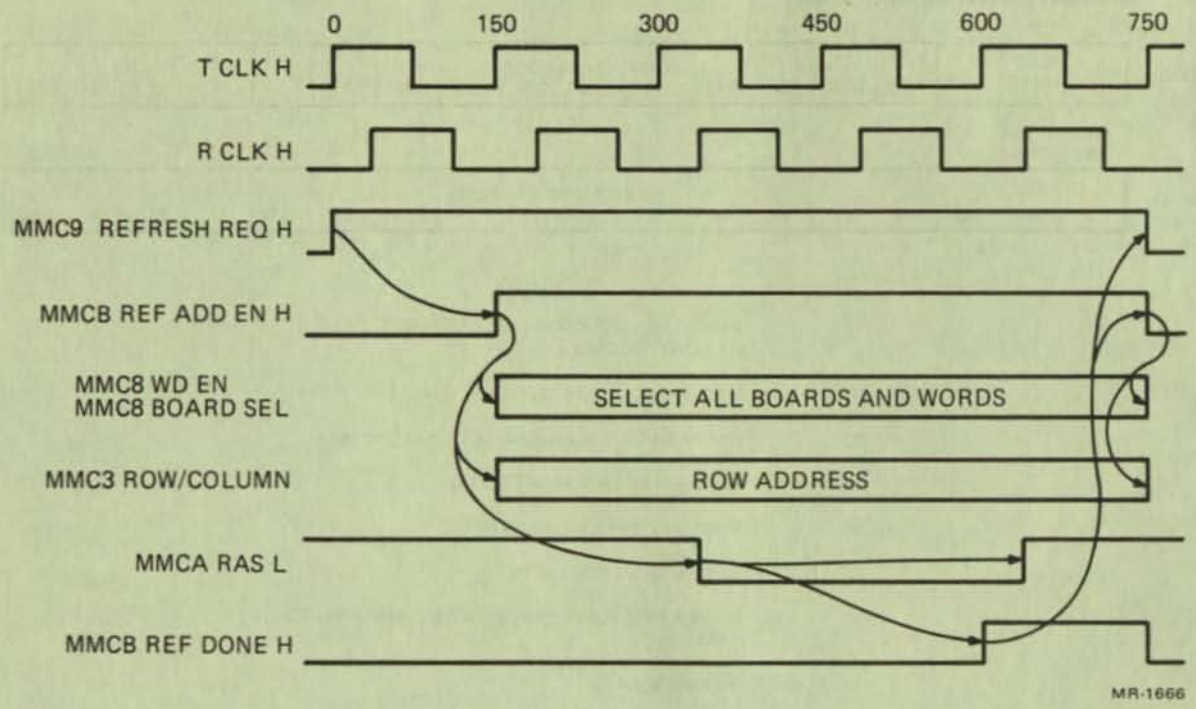


Figure 5-26 Memory Refresh, Timing Diagram

Data Output Drive Current

High	–170 mA typical
Low	170 mA typical

Control and Address Input Current

High	50 μA maximum
Low	–2 mA maximum

Power required by one memory array board is given below. Active means always reading or writing, 1 μs cycles; standby means refresh only, 15 μs between cycles.

Supply	Power in Watts	Current in Amperes
–5 V	0.2	0.035
+5 V	7.4	1.5
+12 V		
Active	20	1.6
Standby	5	0.42

Tolerance on the +12 V and –5 V supplies is ±15% at the board inputs. Tolerance on the +5 V is ±5% at the TTL chip level and is 13% at the board input. These values include peak to peak ac ripple superimposed on the dc level.

5.7 CONSOLE AND CLOCK

The console control and the system clock share the CSL board. Besides these two major pieces of logic, the board also contains the bus arbitrator discussed in Paragraph 5.1.3.

5.7.1 Console

The logic through which the operator controls the system is built around two large-scale monolithic integrated circuits, an Intel 8080A microprocessor and an 8251 universal asynchronous receiver-transmitter. Figure 5-27 is a block diagram of the console hardware, in which the 8080 communicates with the other elements of the logic by means of a data bus (DBUS 0–7) and controls the selection of such elements via an address bus (ADR 0–13). The 8080 with its associated clock generator and drive circuits appears at the left on CSL2. The 8228 has bidirectional drivers for the eight lines of the data bus. The 8224 clock and driver circuit supplies the two 1.638 MHz clocks for the microprocessor chip; it also supplies its oscillator output directly to a divide-by-six counter that supplies a 2.45 MHz clock to the baud rate generators for the UARTs for the console terminal (shown on CSL2) and a remote KLINIK terminal (CSL9).

At the beginning of every microprocessor machine cycle the 8080 places status information identifying the use of the cycle on its D outputs and sends a sync signal to the 8224. This latter chip responds by sending a strobe to the 8228, causing it to load the status into a set of latches, thus freeing the data lines for transfers during the cycle. The status information indicates the kinds of events that will occur during the cycle. From the latched bits and the 8080 control signals WR and DBIN (write and data bus in), the 8228 sets up the memory and I/O bus control signals. A memory read is for the fetch of an instruction, an operand, or an item from the stack; writing in memory may be for an operand or a stack item. (Note that the terms "memory" and "I/O" used in relation to the 8080 have nothing to do with communication over the KS10 bus to system memory or the I/O registers in the other

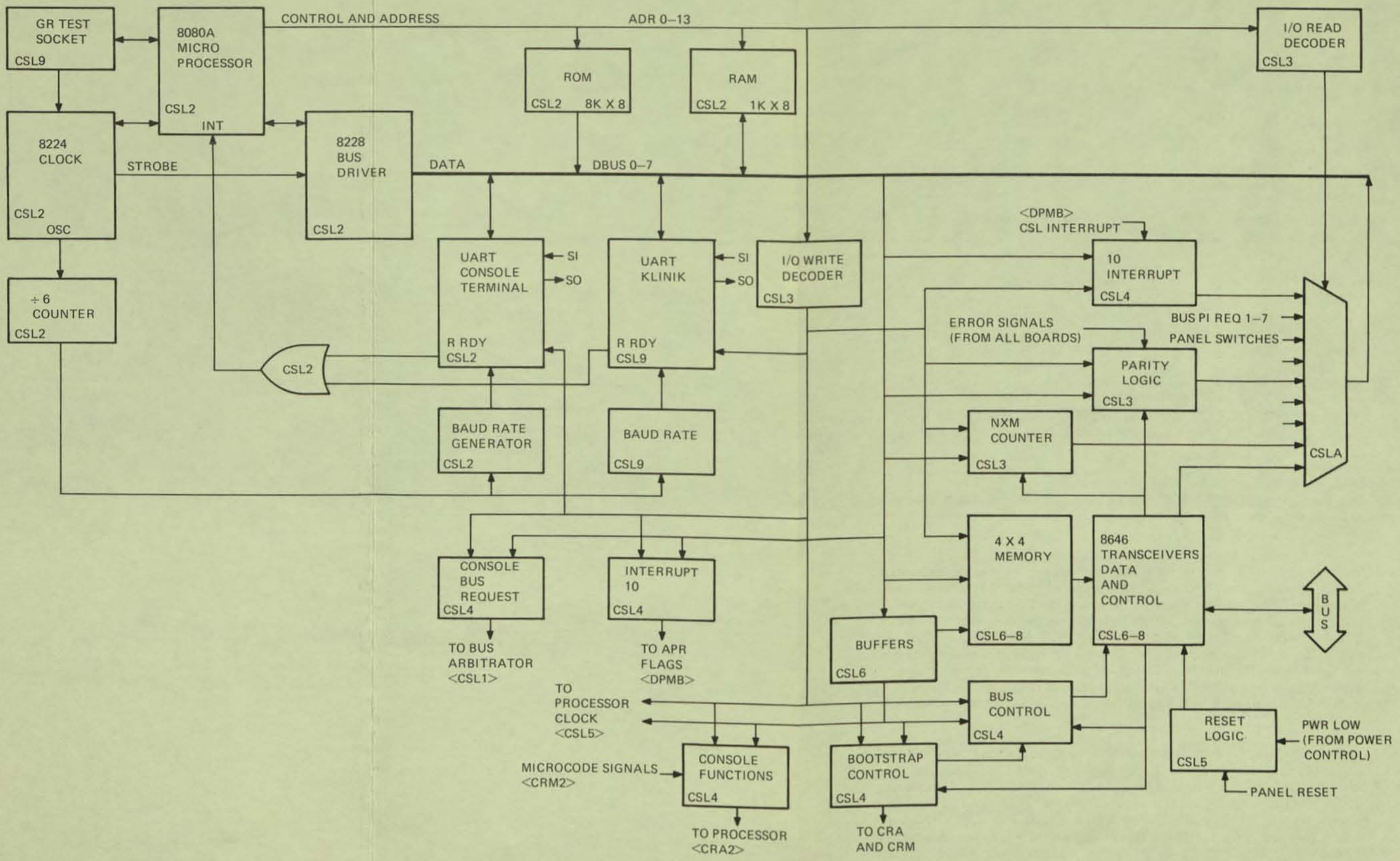


Figure 5-27 Console

subsystems that make up the KS10.) The 8080 memory appears at the right on CSL2 and comprises 2K of RAM for a stack and other general use, and 8K of PROM that contains the console microcode; that is, the program from which the 8080 runs. 8080 I/O includes actual I/O registers scattered throughout the CSL prints as well as generation of individual control functions by I/O writing at particular addresses, often gated by data bits. Of course the console register the KS10 processor can address and the register through which the console communicates with system memory or a UBA both appear as sets of 8-bit I/O registers to the 8080. These two registers are contained in the 4 × 4 memory chips (only half used) that appear at the left of the 8646 transceivers on CSL6–8.

All operations performed in the console are a function of the microcode stored in the PROM. Detailed information about the microprogram is contained in the KNS10 microcode listing. The commands the microcode can perform are listed in Paragraph 4.6.1. Here we are concerned only with the way the hardware functions to carry out the console microcode. The 8080, under the direction of the operator, controls the entire system via I/O write functions and inspects conditions that indicate the state of the system via I/O read functions. Address decoding for I/O write is handled by the 138 decoders at the right on CSL3 and the gates at A6. By loading registers located at the left on CSL3, at center and upper left on CSL4, and elsewhere, the 8080 determines which errors will be detected and the effect they will have, requests access to the KS10 bus and transmits information over it, controls the processor clock, enables the cache and other features throughout the system, and requests an interrupt in the processor by setting APR flag 31 on DPMB. Preliminary decoding of read functions is handled by the 139 on CSL3 A7 to read information from many sources through the mixers on CSLA.

The logic at the top on CSL3 detects errors in data received by the console over the KS10 bus and also receives error signals from the other subsystems; under control of the 8080 these various error signals set the PE flag to stop the processor clock. In the lower left quarter on CSL4 are flip-flops that handle the console functions run, execute and continue, under control of both the 8080 and the KS10 microcode. Above these are flip-flops through which the KS10 microcode indicates when it is in the halt loop, the processor requests an interrupt in the console, and the console requests interrupt in the processor. The logic at the upper right handles normal communication over the KS10 bus: this includes bus requests and transmissions by the console, and decoding of a command/address given by the processor for the console. The registers at the upper left and logic at the lower right handle the bootstrapping of the microcode into the control RAM directly through the special bus transceivers located on the CRA board. This operation also makes use of the regular bus communication logic at the upper right.

5.7.2 Clock

The overall system clock is made up of the crystal oscillator. flip-flops, and microswitch-controlled pulse delay circuits in the left and lower three quarters on CSL1. The oscillator frequency of 26.66 MHz is divided into two 6.66 MHz pulse trains, with the R train trailing the T train by one quarter period. At the left are three deskew circuits for the system T clock lines, which drive the T clock circuits on all boards. Only two circuits are needed (lower right) for the system R clock lines, as only those boards connected to the KS10 bus have R clock circuits. Note that there is no R clock line to the CRA board as the timing signals for bootstrapping are generated directly by the bootstrap logic on CSL4.

The basic enable for the processor cycle clocks is the E501 flip-flop on CSL5 C1. From this basic enable, the net in the lower right corner generates separate enables for the microcontroller and data path boards to allow the data path to execute a fast shift while the microcontroller is static, and to allow the microcontroller to respond to a page failure while the data path is static.

The enable is turned on by an R clock whenever all inputs to any one of the four E505 AND gates above the flip-flop are true. The top gate provides for one processor cycle at a time entirely under control of the console 8080 microprocessor for single stepping. The second gate holds the enable on during a fast shift. The bottom two gates are both disabled when a parity error is detected or the 8080 specifically turns off the clock. While a read delay disables the bottom gate, the second gate from the bottom allows two T clock periods for the console logic to set up the microcontroller to handle a page failure, and it then enables the cycle clock. The bottom gate is regularly used to enable the cycle at every second T clock but it also provides for three types of delay: a stretchout of the cycle as selected by the KS10 microcode T field through the E601 mixer; a write delay until the bus is granted to the processor; and a read delay until the processor has received the data.

5.8 UNIBUS ADAPTER

The Unibus Adapter (UBA) is a KS10 I/O controller that allows Unibus peripheral devices to be connected to the KS10 system. It connects between the internal KS10 (backplane) bus and a Unibus as shown in Figure 5-28. More than one UBA may connect to the backplane bus, thus allowing more than one Unibus to be interfaced to the KS10 system. Each UBA consists of a single extended hex module (M8619) mounted in the KS10 cabinet. Physical locations are given in Chapter 1.

5.8.1 Basic Operation

A UBA controls and synchronizes the following major operations that take place between the connecting Unibus devices and the rest of the system.

1. NPR data transfers from Unibus devices to KS10 memory, and from KS10 memory to Unibus devices
2. I/O register data transfers (initiated by the CPU or console) to/from Unibus devices
3. Vector address transfers from Unibus devices to the CPU following device interrupts

5.8.1.1 NPR Data Transfers - The UBA allows the following NPR data transfers by a Unibus device to/from KS10 memory.

1. DATO to memory (16- or 18-bit word)
2. DATOB to memory (8-bit byte)
3. DATI from memory (16- or 18-bit word or 8-bit byte)

The transfers to/from memory are direct with no intervention or control by the CPU. Unibus data positioning within the KS10 memory word is shown in Figure 5-29. Correspondence to the Unibus address is indicated.

Data flow for the NPR write to memory operation (i.e., DATO or DATOB by device) and the NPR read from memory operation (i.e., DATI by device) are shown in Figures 5-30 and 5-31. Note that word transfers may be 18 bits as well as 16 bits. (Some devices such as the RH11 use the 2 Unibus parity lines in addition to the 16 data lines to transfer NPR data.) Also note that in addition to normal byte and word transfers, a fast transfer mode of operation is implemented for word transfers only. This mode, which is program selectable, is used for transfers to/from high speed I/O devices such as disks. It provides an extra 18 bits of data buffering, thus reducing the number of KS10 bus memory operations by a factor of 2. Fast transfer mode is set by loading a bit in the paging RAM as specified in Paragraph 5.8.2.

NOTE

Fast mode should not be set for more than one device on a Unibus. Simultaneous NPR data transfers on a single Unibus give unspecified results when two or more of the active devices are transferring data in fast mode.

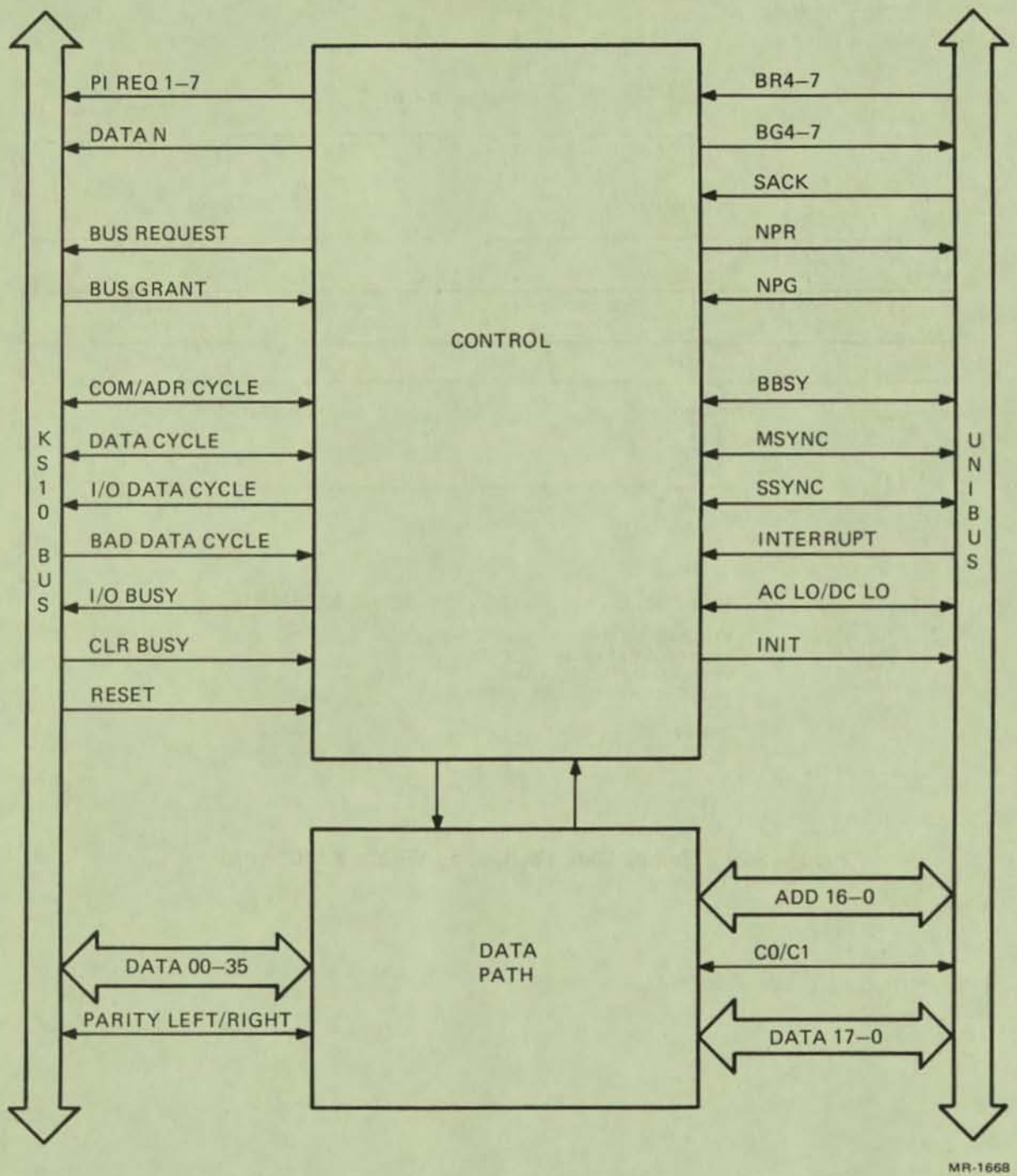
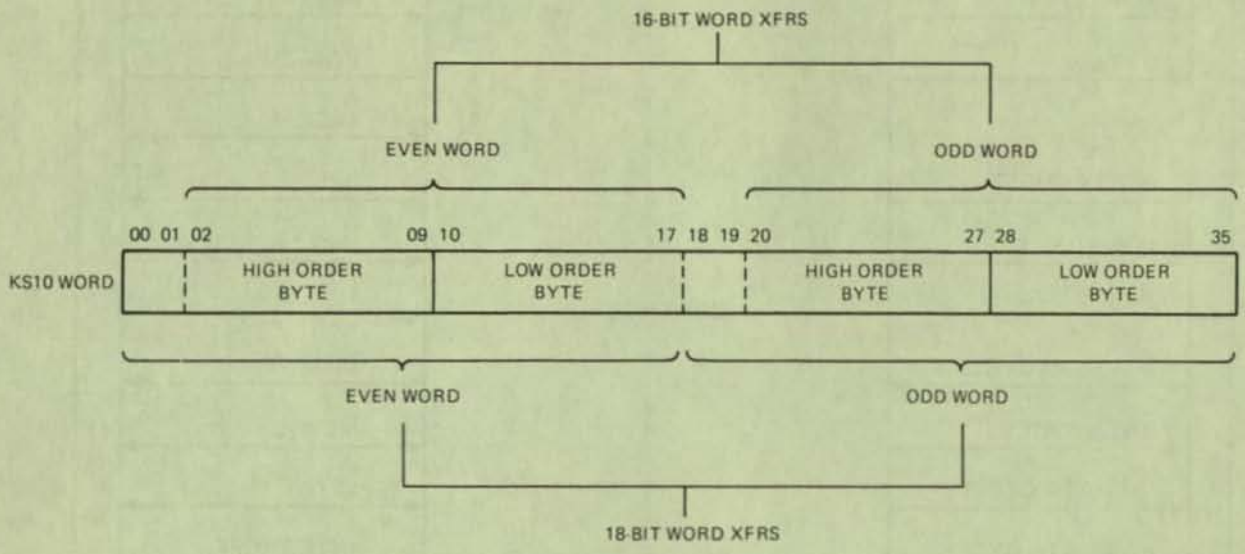


Figure 5-28 UBA, Simplified Block Diagram



UNIBUS DATA	UNIBUS ADDRESS BITS A1	A0
LOW ORDER BYTE – EVEN WORD	0	0
HIGH ORDER BYTE – EVEN WORD	0	1
LOW ORDER BYTE – ODD WORD	1	0
HIGH ORDER BYTE – ODD WORD	1	1
EVEN WORD	0	0
ODD WORD	1	0

MR 1669

Figure 5-29 Unibus Data Positioning Within KS10 Word

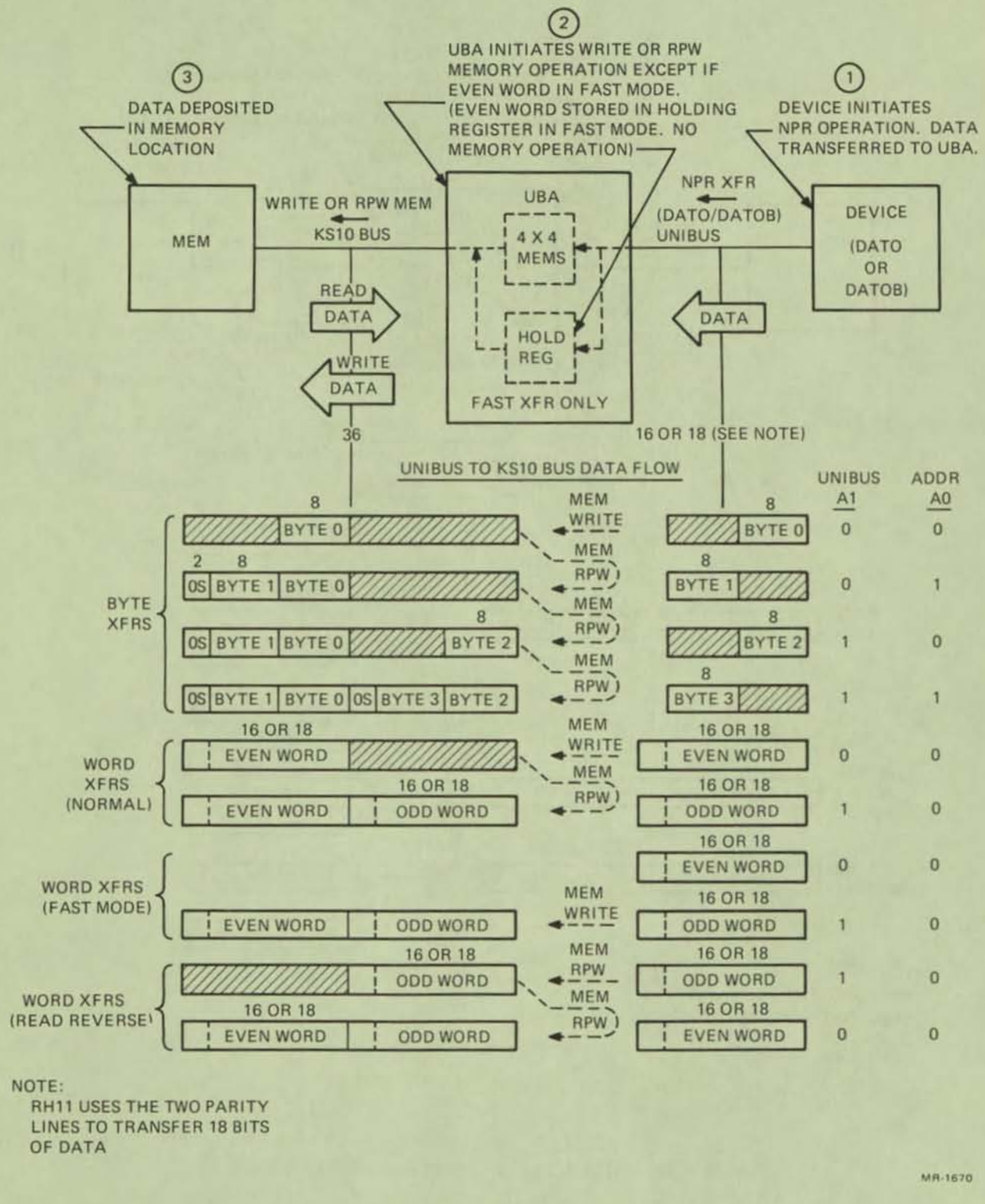


Figure 5-30 NPR Write (To Memory), Data Flow

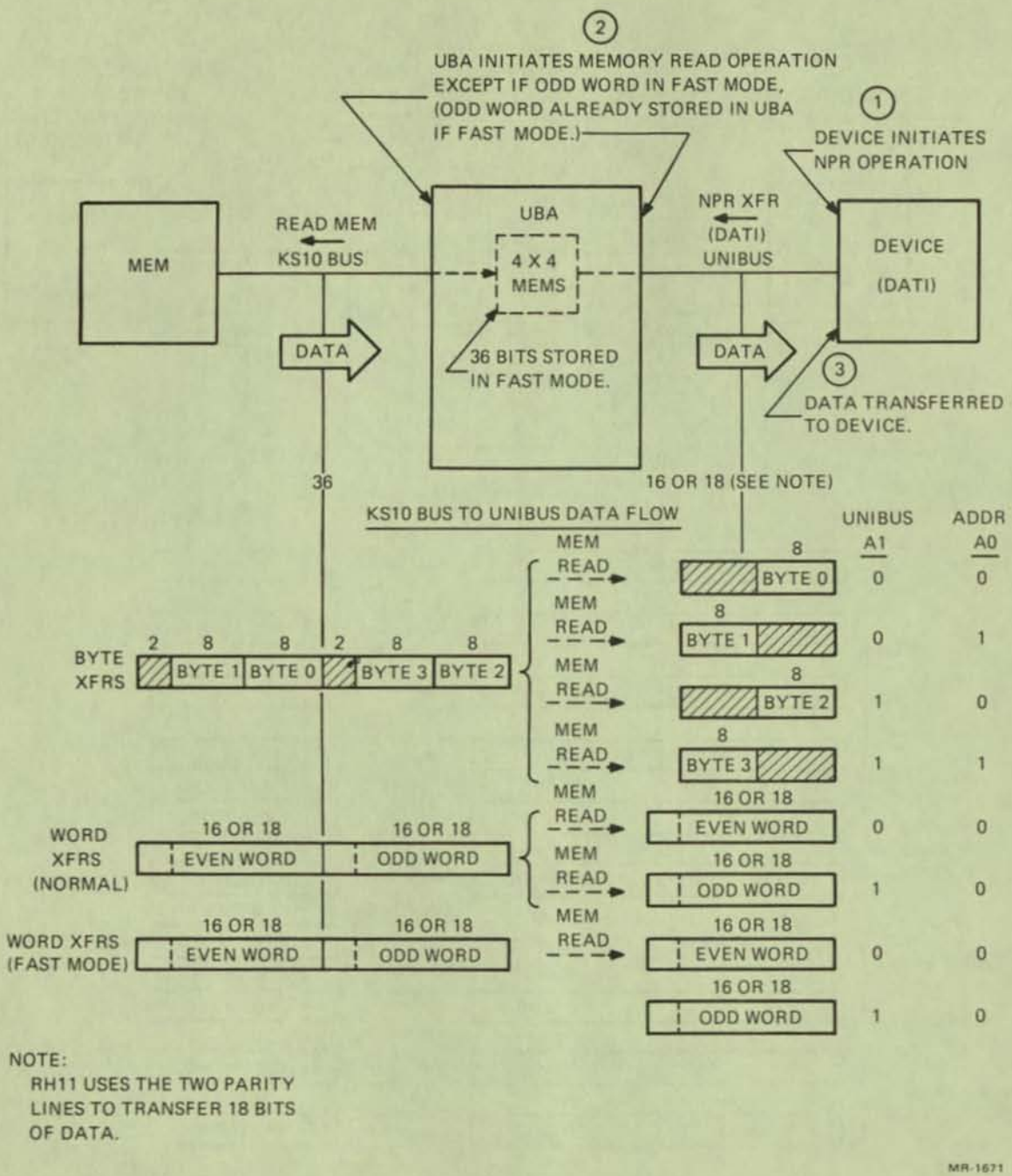


Figure 5-31 NPR Read (From Memory), Data Flow

In addition to the fast mode of operation for both NPR write to memory operations and read from memory operations, a special mode for NPR write to memory operations is implemented in the UBA to accommodate device read reverse operations from slower devices (e.g., tape). The read reverse mode is provided mainly for diagnostic program use; it is not utilized by the system monitor. As for fast mode, read reverse mode is set by loading a bit in the paging RAM as specified in Paragraph 5.8.2.

NOTE

Results are unspecified if read reverse mode and fast transfer mode are both set for the same NPR data transfer.

Basic operation during NPR data transfers is as follows.

1. The Unibus device, in response to a previously issued read/write command, initiates a Unibus NPR operation when it has read data to transfer to KS10 memory, or when it requires write data from KS10 memory.

2. For *NPR write to memory* operations, the UBA stores the read data transferred over the Unibus and then does one of the following depending on the Unibus address and the transfer type:

    a. Byte transfers – For the low order byte in an even word (byte 0 in Figure 5-30), which is the first byte loaded into a KS10 memory location during execution of a device read command, the UBA does a memory write operation on the KS10 bus to load the byte directly into memory. For all other bytes (bytes 1, 2, and 3 in Figure 5-30), the UBA does a memory read-pause-write operation. The read-pause-write operation is necessary so that the data loaded in memory during the device's previous NPR data transfer may be first read and recirculated by the UBA. The previously loaded data is then written back into memory along with the current byte.

    b. Word Transfers (Normal) – For normal even word transfers (as for byte 0 transfers), the UBA does a memory write operation over the KS10 bus to load the data directly into memory. For normal odd word transfers (as for bytes 1, 2, and 3), the UBA does a read-pause-write memory operation. This reads the previously loaded even word and then writes both the odd and even word into memory.

    c. Word Transfers (Fast Mode) – For fast mode even word transfers, the UBA initiates no memory operation. The word is loaded from the Unibus into a holding register until the next NPR data transfer (an odd word). The UBA then initiates a memory write operation to write both even and odd words into memory.

    d. Word Transfer (Read Reverse) – For transfers in read reverse mode, the UBA receives data words from the Unibus in reverse order; that is, the odd word is received and written by the UBA into a memory location first. This, and the fact that a read-pause-write memory operation is initiated for odd and even words, is the only difference in data flow between read reverse and normal operation.

3. For all *NPR read from memory* operations, except for odd words in fast mode, the UBA does a memory read operation over the KS10 bus, temporarily stores the memory data, and then transfers the byte or word specified by the Unibus address over the Unibus to the device. In fast mode, both odd and even words are stored in the UBA during the even word transfer. Thus, for the next (odd word) transfer, the data is transferred directly to the device with no memory operation being required.

5.8.1.2 I/O Register Data Transfers – The UBA allows the CPU or console to write and read the addressable I/O registers in the Unibus devices connected to the KS10 system. Transfers initiated on the Unibus by the UBA in response to KS10 bus commands are as follows.

1. DATO to device (16-bit word)
2. DATOB to device (8-bit byte)
3. DATI from device (16-bit word or 8-bit byte)

In addition to writing and reading Unibus device (external) registers, the CPU or console may also write/read registers in the UBA itself. These UBA (internal) registers are discussed in Paragraph 5.8.2.

Data flow for both I/O register write and read operations is shown in Figures 5-32 and 5-33. I/O register data transfers are initiated by the CPU as a result of the external instruction set (Paragraph 4.3.2). Both word and byte instructions can be executed. I/O register data transfers are also initiated by the console in response to commands entered via the CTY (DI and EI commands). The console does only word transfers; byte transfers are not implemented.

NOTE

All UBA internal and external I/O register addresses have the most significant register address bit (the 64K bit) equal to 1. If an I/O register data transfer is directed to a UBA and this address bit is equal to 0, it forces a UBA NPR data transfer cycle causing I/O register data to be read from, or written into, KS10 memory. This maintenance feature, called wrap-around mode, is discussed in Paragraph 5.8.7.

Basic sequence of operations for I/O register data transfers is as follows.

1. The CPU (when an I/O instruction is executed) or the console (when the appropriate command is given) initiates an I/O register write or read operation on the KS10 bus.

2. For an *I/O register write* operation, and when an external (Unibus) register is addressed, the UBA responds by loading the register data from the KS10 bus and then initiating a Unibus DATO or DATOB operation to write the word or byte into the addressed device register. When an internal (UBA) register is addressed, no Unibus action is required and the data is loaded directly into the UBA register address.

3. For an *I/O register read* operation and an external address, the UBA responds by performing a Unibus DATI operation (for both word and byte operations) to read the addressed register. Because the time required to retrieve the register data from the device is greater than the time allotted the CPU or console for the KS10 bus operation (3 bus cycles), the UBA is disconnected from the KS10 bus during the Unibus DATI operation. As a result, when the register data is finally received from the device, the KS10 bus must be requested again, this time by the UBA. (The CPU or console requested the bus originally to initiate the operation.) The UBA then does a KS10 bus data cycle to transfer the data to the CPU or console. The UBA also performs a KS10 bus data cycle when an internal (UBA) register is addressed. In this case, however, the UBA is disconnected from the KS10 bus for only a short interval if it is granted the bus immediately. This is because there is no delaying Unibus action; the data being readily available from the UBA register address.

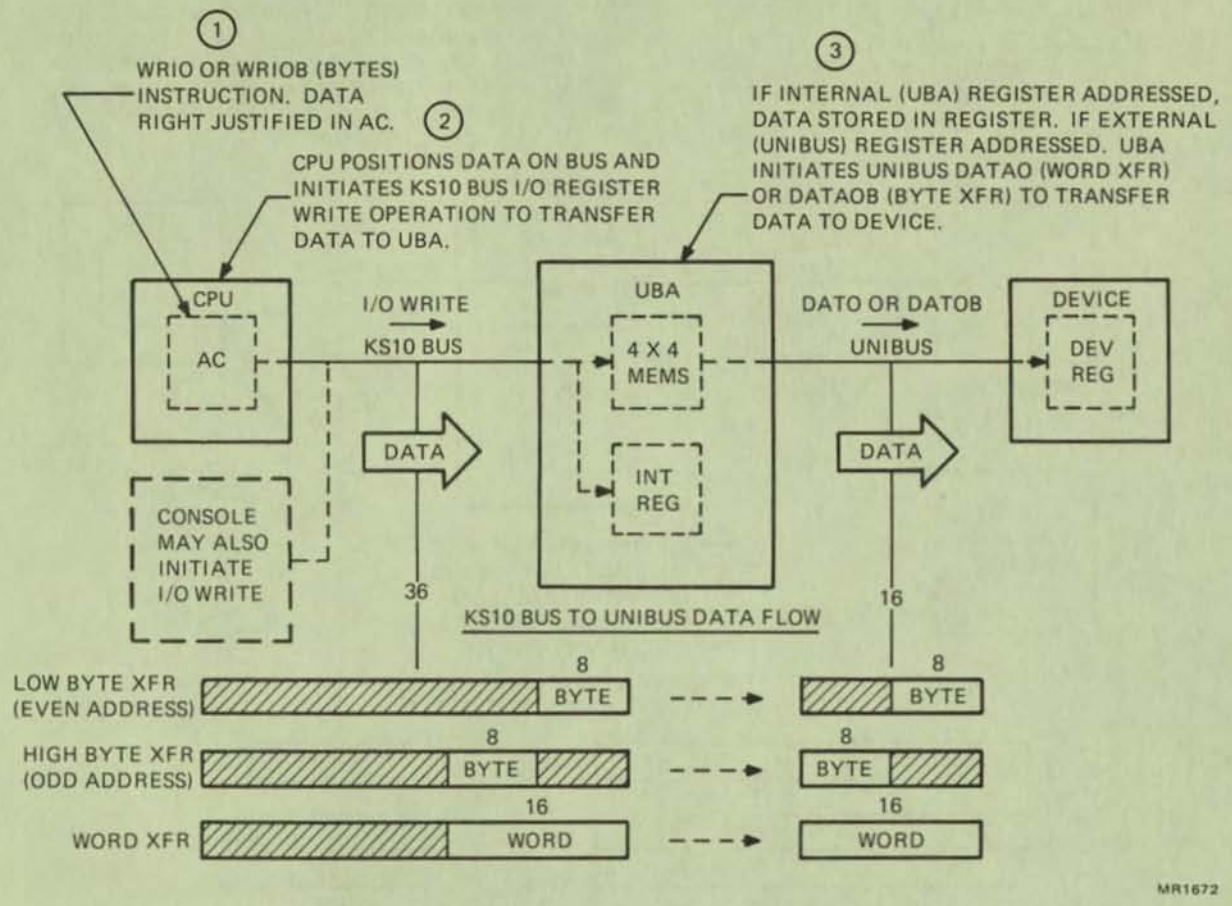


Figure 5-32 I/O Write, Data Flow

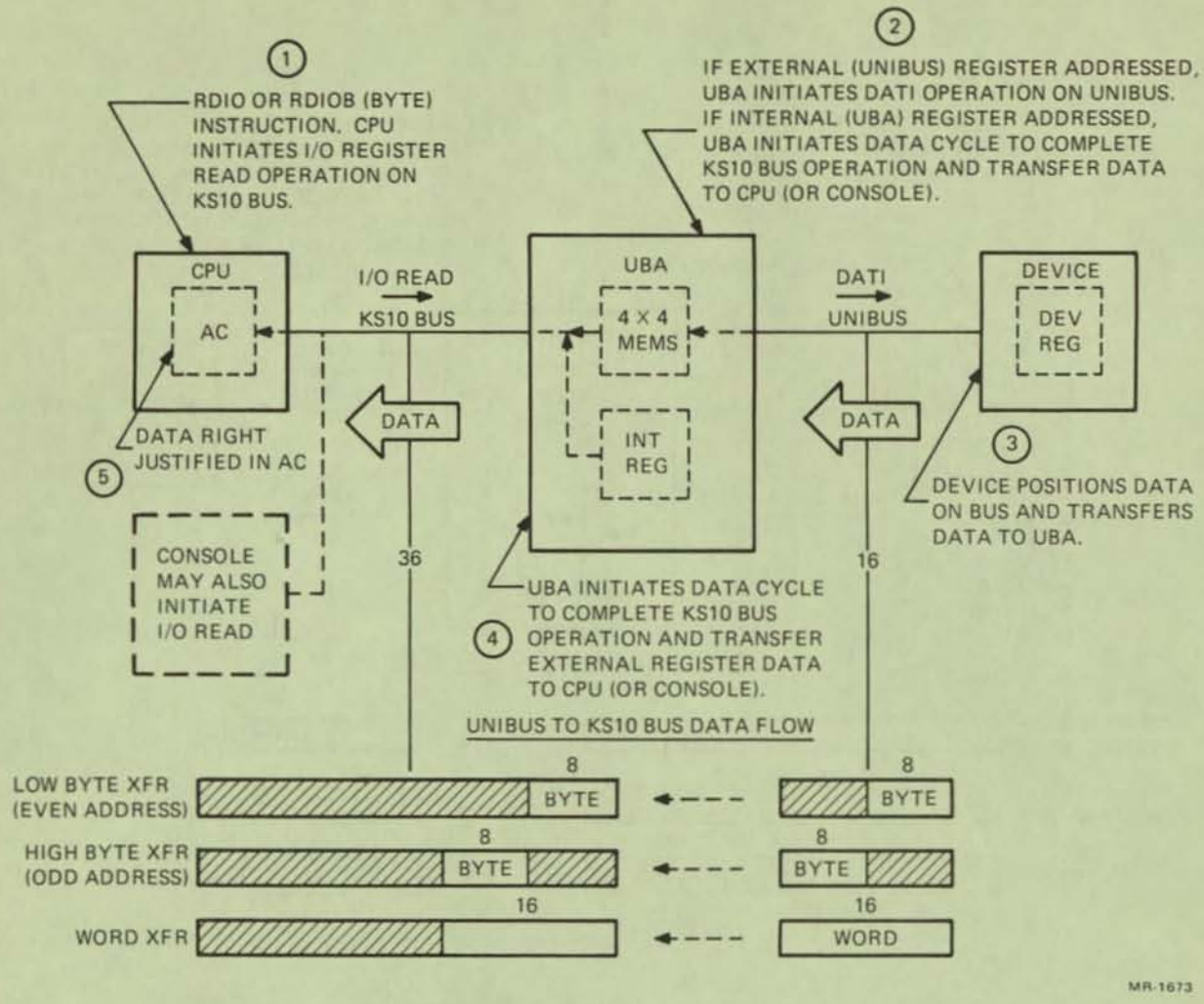


Figure 5-33 I/O Read, Data Flow

5.8.1.3 PI Operation – The UBA monitors all interrupt requests (BR levels) on the Unibus and asserts PI requests (1–7) on the KS10 bus depending on the PI channel number (PIA) loaded in the UBAs status register (Paragraph 5.8.2). Both a low level PIA (for BR4 and BR5) and a high level PIA (for BR6 and BR7) may be loaded, allowing one group of Unibus devices to interrupt at one PI request level, and a second group to interrupt at another PI request level. Following a device interrupt request, the UBA performs a second major PI function by allowing the Unibus device interrupt vector to be transferred to the CPU. Data flow is shown in Figure 5-34. Basic operation is as follows.

1. When the CPU detects a PI request on the KS10 bus, it first resolves PI channel number priority; that is, more than one PI request may be asserted on the bus by the various I/O controllers (e.g., UBAs) and the CPU selects the highest priority (lowest numbered) channel to service.

2. The CPU then performs a KS10 bus operation to read the controller numbers interrupting on the selected channel. (More than one controller may assert the same PI request line.) In response, each interrupting controller asserts a data line corresponding to its controller number. UBA1 (controller 1) asserts data line 19 and UBA3 (controller 3) asserts data line 21.

3. After reading the interrupting controller numbers for the selected PI channel, the CPU resolves controller number priority (lowest number has highest priority) and initiates another KS10 bus operation to read the interrupt vector from or (in the case of a UBA) via the selected controller. In response, an addressed UBA initiates a Unibus interrupt operation to read the vector from the highest priority Unibus device interrupting on the PI channel being serviced. (Devices may be asserting both of the BR levels associated with the PIA, and more than one device can assert the same BR level.)

4. To initiate a Unibus interrupt operation, the UBA asserts the BG level corresponding to the highest priority BR level asserted (highest numbered BR level has highest priority). For example, if the low level PIA is being served and both BR4 and BR5 are asserted, the UBA asserts BG5. Once the BG level is asserted, the first device on the Unibus asserting the associated BR level transfers the vector to the UBA. (The device electrically nearest the UBA has highest priority.) The UBA, in turn, then transfers the vector to the CPU. As for an I/O register read operation, the UBA is disconnected from the KS10 bus during the Unibus interrupt operation. Thus, when it collects the vector from the device, it must first request the KS10 bus before transferring the vector to the CPU via a data cycle.

5.8.2 UBA Status and Control Registers

The UBA has the following internal registers.

Address	Register	Read/Write
763000-77	Paging RAM	R/W
763100	Status Register	R/W
763101	Maintenance Register	W

The registers may be accessed with the external I/O instruction set (WRIO, RDIO, etc.) or by the appropriate console commands (DI and EI). Note that the maintenance register (763101) is a write-only register.

5.8.2.1 Paging RAM – The 64-location paging RAM allows a virtual address on the 18 Unibus address lines to be translated to a 20-bit physical KS10 memory address during NPR data transfers. Each RAM location contains 16 bits, 11 of which are used to specify a KS10 memory page address.

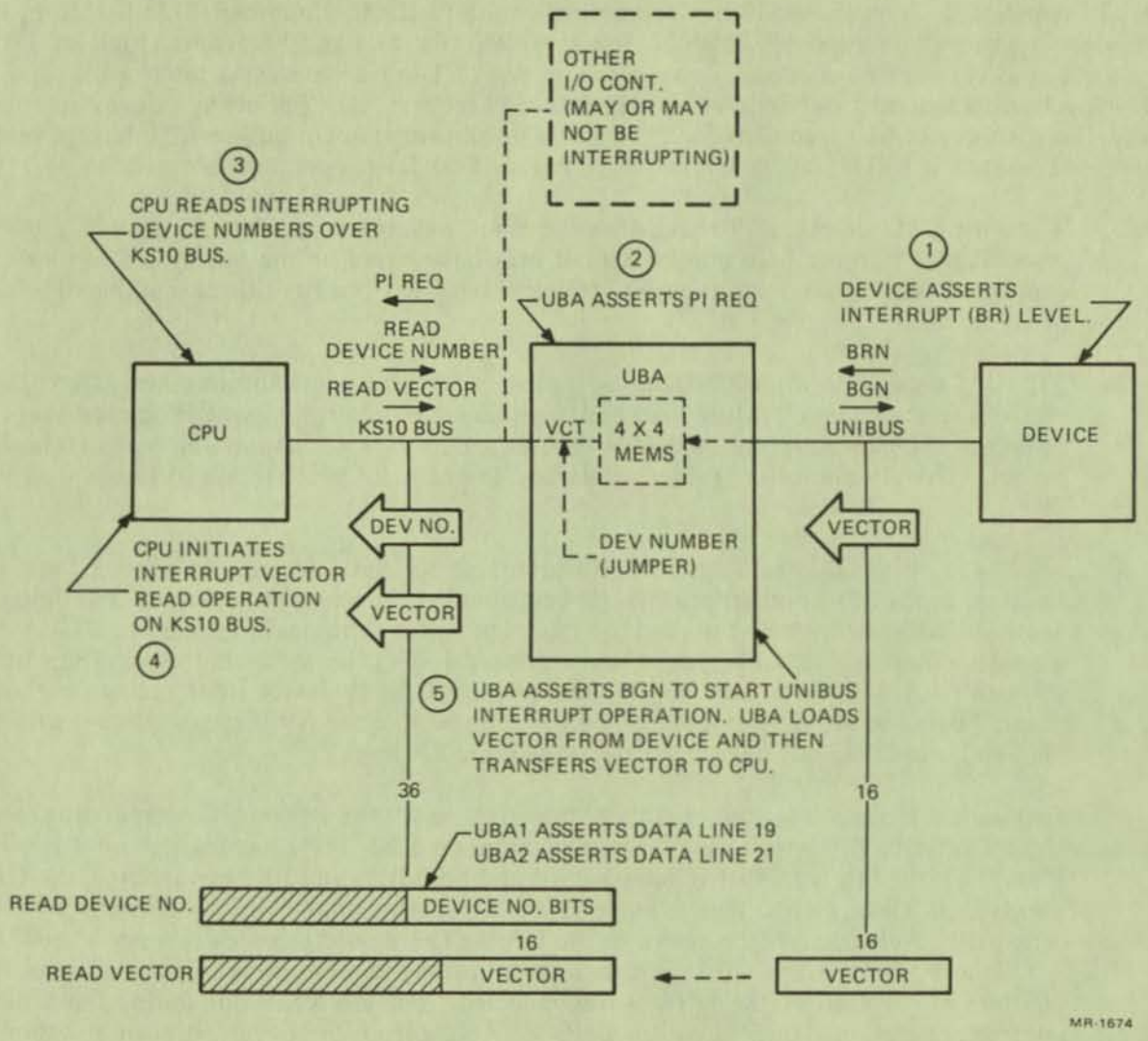


Figure 5-34 PI Operation, Data Flow

The other 5 RAM bits are used for control purposes. Bit format and definitions for the I/O instructions accessing the RAM are given in Figure 5-35. As shown, bit format when loading the RAM is not the same as when reading the RAM locations.

Unibus to memory address translation is shown in Figure 5-36. The two least significant bits of Unibus address specify the position of Unibus data within a memory location and are not used as part of the memory address; bit 1 specifies an odd or even word; bit 0 specifies a high or low order byte. (Refer to Figure 5-29.) The next 9 least significant bits of Unibus address (bits 10–2) are used directly as the 9 least significant bits of memory address (bits 27–35). This is similar to virtual to physical address translation in the CPU. The 9 bits specify one of 512 words; that is, a word within a 512 word page. To furnish the page address, 6 of the remaining 7 Unibus address bits (bits 16–11) select a paging RAM location. The contents (the 11 bit address) then supply the KS10 memory page address (memory address bits 16–26). The most significant bit of Unibus address (bit 17) is not used.

NOTE

The most significant bit of Unibus address (the 64K bit) must be made equal to 0 for NPR data transfers. If equal to 1, a memory reference is not made, causing the Unibus device to time-out, set an error flag, and terminate the device read or write operation.

The five paging RAM control bits are as follows.

1. VALID – Indicates physical page number is a valid address. Set by program when paging RAM is loaded.

2. READ REVERSE (READ-PAUSE-WRITE) – Forces read-pause-write memory cycles for all NPR write (to memory) transfers. Allows read reverse operations by a Unibus device by causing odd words previously loaded in memory to be read and recirculated by the UBA during even word transfers. (Normally, even words are the first data loaded in a memory location and they are loaded directly via a memory write cycle.)

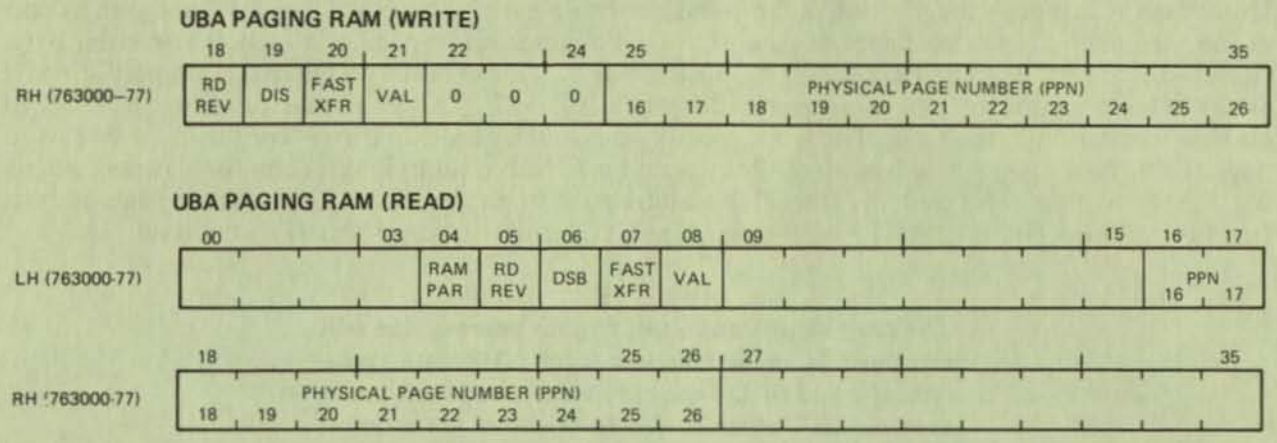
3. DISABLE – Prevents the two most significant bits of Unibus data in KS10 memory (bits 0 and 1, or bits 18 and 19) from being transferred to the Unibus data lines (17 and 16) during NPR read (from memory) operations. The two Unibus data lines, which are device parity error lines during non-18-bit transfers, are forced to 0 to prevent non-zero data in memory from causing false parity error indications in the Unibus device. The DISABLE bit must not be set for 18-bit word transfers.

4. FAST TRANSFER (36 BIT ENABLE) – Sets fast transfer mode for NPR word transfers. In this mode, both odd and even words of Unibus data (a total of 36 bits) are transferred during a single KS10 memory reference.

5. RAM PARITY – Paging RAM odd parity bit. Generated by hardware when paging RAM is loaded.

NOTE

If a RAM parity error, or the absence of a RAM valid bit, is detected during an NPR transfer, it will cause the associated Unibus device to time-out, set an error flag, and terminate the device read/write operation.



BIT(S)		FUNCTION
WRT	RD	
–	04	PAGING RAM PARITY BIT
18	05	READ REVERSE
19	06	DISABLE PARITY BIT XFR TO UNIBUS
20	07	FAST TRANSFER MODE
21	08	ADDRESS IS VALID
22–24	–	MBZ (MUST BE ZERO)
25–35	16–26	PHYSICAL PAGE NUMBER

MR-1675

Figure 5-35 Paging RAM

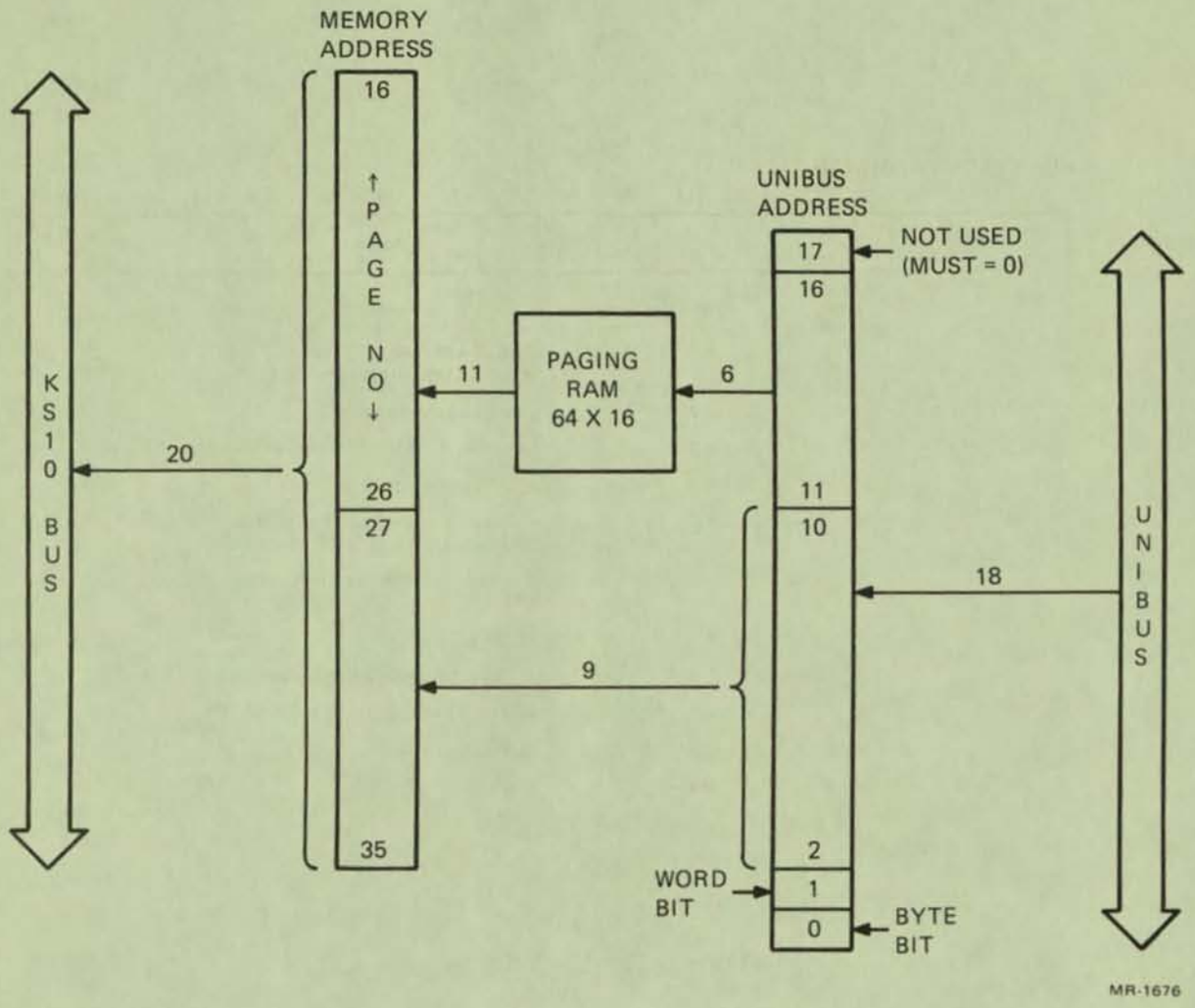


Figure 5-36 Unibus to Memory Address Translation

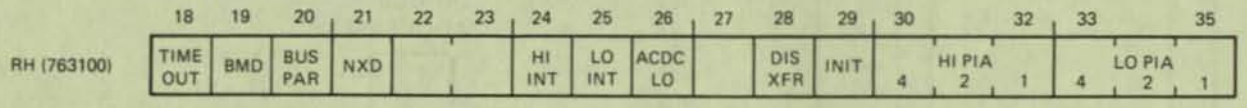
5.8.2.2 Status Register – UBA status register bit format and definitions are given in Figure 5-37. As shown, provision is made to indicate both high and low level interrupt requests (bits 24 and 25, respectively), and to load and indicate high and low level PIAs (bits 30–32 and 33–35, respectively). Provision is also made to initialize both the UBA and the Unibus devices (bit 29 = 1). In addition, there are five error flags and a DISABLE TRANSFER control bit as follows.

1. TIMEOUT (bit 18) – Indicates a Unibus arbitrator time-out (10 μs) or a nonexistent memory time-out (1.2 μs). The Unibus arbitrator time-out may be caused by:

   a. No SACK signal received from a Unibus device after the UBA granted the Unibus to a device for an NPR or interrupt vector data transfer. Usually indicates a system malfunction.

   b. No SSYNC signal received from a Unibus device after the UBA initiated a DATO, DATOB, or DATI operation. Usually indicates a nonexistent device.

UBA STATUS REGISTER



BIT(S)	READ/WRITE	FUNCTION
* 18	R/W	UNIBUS ARBITRATOR TIME-OUT OR NON-EXISTENT MEMORY ADDRESS.
* 19	R/W	BAD MEMORY DATA
* 20	R/W	KS10 (BACKPLANE) BUS PARITY ERROR
* 21	R/W	NON-EXISTENT DEVICE
24	R	HIGH LEVEL INTERRUPT PENDING (BR7 AND BR6).
25	R	LOW LEVEL INTERRUPT PENDING (BR5 AND BR4).
26	R	AC OR DC LOW
28	R/W	DISABLE TRANSFER IF BMD (BIT 19 = 1).
29	W	INITIALIZE UBA AND UNIBUS DEVICES
30–32	R/W	HIGH LEVEL PIA.
33–35	R/W	LOW LEVEL PIA.

NOTE:
*WRITING A 1 BIT CLEARS THE FLAG.

MR-1677

Figure 5-37 UBA Status

The nonexistent memory time-out is caused by the condition that no MEM BUSY signal was received after the UBA was granted the KS10 bus for a memory operation during an NPR data transfer. It usually indicates a nonexistent memory address.

The TIMEOUT error flag is cleared by writing the status register with bit 18 = 1.

2. BAD MEMORY DATA (bit 19) - Indicates uncorrectable data was read from memory during an NPR data transfer. Error may be set not only during an NPR read from memory data transfer (memory read operation), but also during an NPR write to memory data transfer (memory read-pause-write operation). Except for an NPR read from memory operation when DISABLE TRANSFER (bit 28) is **not** set, this error prevents the UBA from generating SSYNC, causing the device controller (e.g., RH11) to time-out and terminate the device read or write operation. The BAD MEMORY DATA error flag is cleared by writing the status register with bit 19 = 1.

3. BUS PARITY ERROR (bit 20) - Indicates that the UBA detected a KS10 (backplane) bus parity error when it either received or transmitted bus information. Unless disabled by a console command, a BUS PARITY error causes the console module to stop the CPU clock. The BUS PARITY ERROR flag is cleared by writing the status register with bit 20 = 1.

4. NONEXISTENT DEVICE (bit 21) - Indicates that no SSYNC signal was received from a Unibus device 10 μs after the UBA initiated a DATO, DATOB, or DATI operation. Usually indicates a nonexistent device. This error condition also sets a TIMEOUT error (bit 18). The NONEXISTENT DEVICE error flag is cleared by writing the status register with bit 21 = 1.

5. AC/DC LOW (bit 26) - Indicates assertion of Unibus AC LOW or DC LOW (condition sensed by H765 P.S.), or assertion of KS10 bus AC LOW (condition sensed by H7130 P.S.).

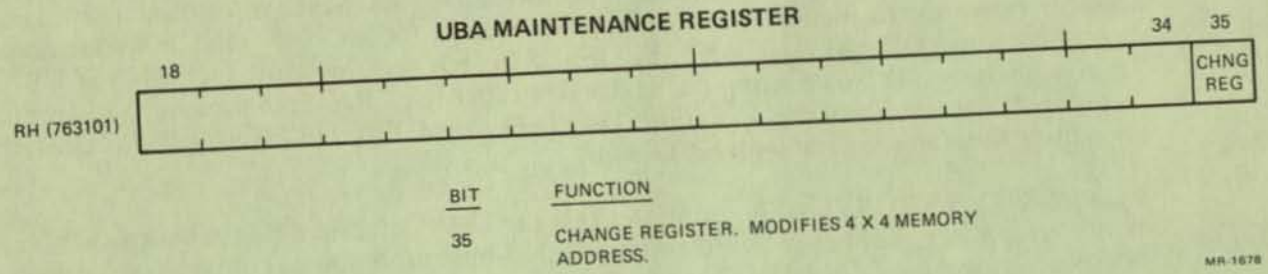
5.8.2.3 Maintenance Register - The maintenance register contains a single write-only control bit as shown in Figure 5-38. CHANGE REGISTER (bit 35), when set during an I/O register read/write or interrupt vector read operation, modifies the addressing logic for the 4 × 4 memories interfacing to the Unibus so that the data received or transmitted on the bus is stored in the 4 × 4 memory locations normally used for NPR data transfers. This is to facilitate operation in wraparound mode (Paragraph 5.8.7), but it also allows a quick check of 4 × 4 memory operation when Unibus data is in error during normal operation. For example, if it is found that after writing and reading a Unibus device register that the register data does not match, the maintenance bit may be set and the operation repeated. If the register data then agrees, it indicates a bad 4 × 4 memory location.

5.8.3 Logical Organization

With reference to Figure 5-39, the UBA consists of the following major logic elements.

1. Data path
2. NPR control
3. I/O read/write control
4. Unibus arbitrator
5. Unibus control
6. KS10 bus control

The *data path* (UBA circuit schematics UBA8, 9, A–C) consists of data mixers, KS10 bus transceiver/latches, 4 × 4 IC memory elements, and Unibus drivers and receivers that are arranged to allow NPR and I/O register data to pass between the KS10 bus and the Unibus. The data path also transfers addressing information, and it contains an address register to store I/O register addresses and the 64 × 16 bit paging RAM for NPR address translations.



BIT	FUNCTION
35	CHANGE REGISTER. MODIFIES 4 X 4 MEMORY ADDRESS.

Figure 5-38 Maintenance Register

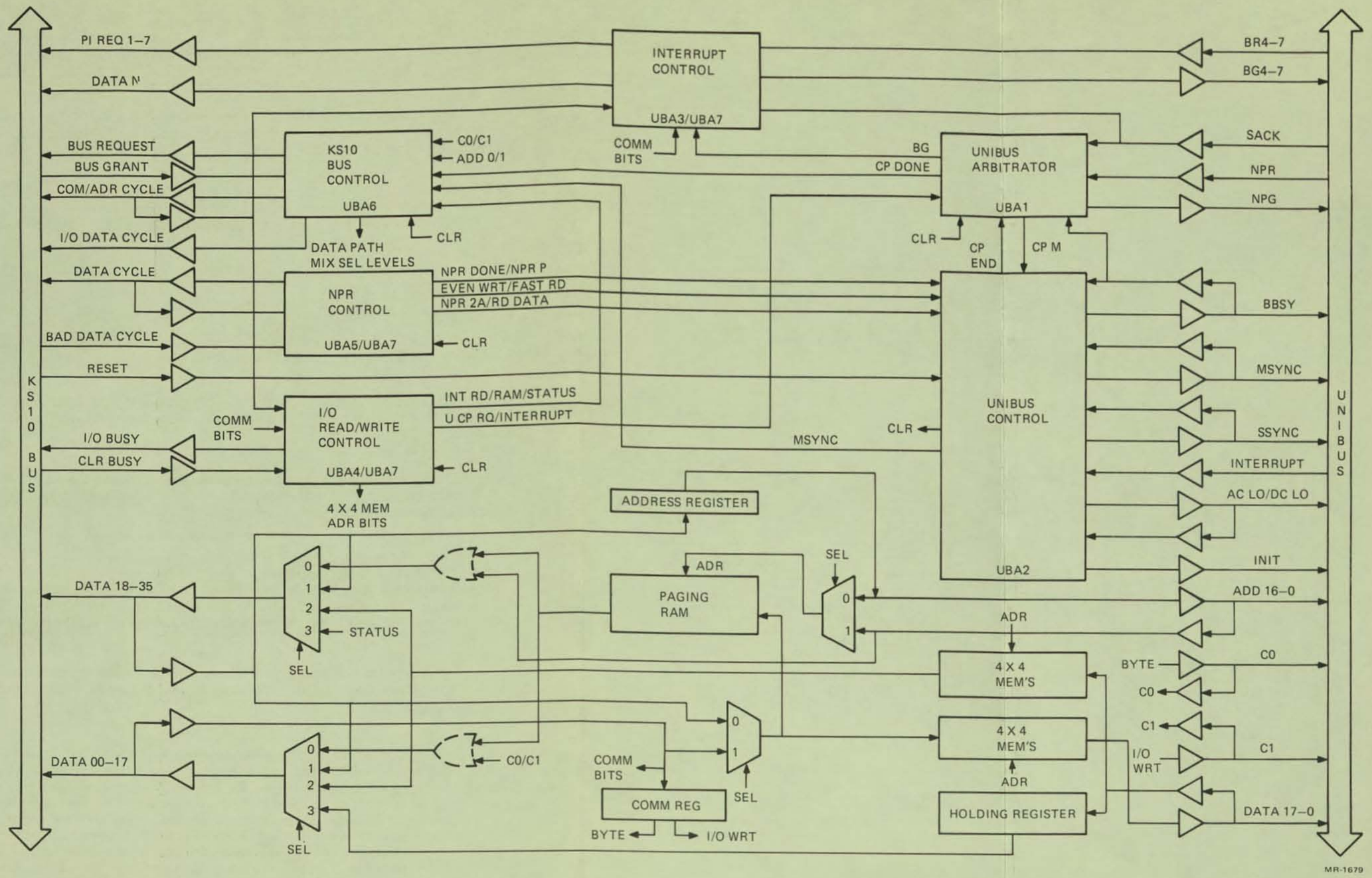


Figure 5-39 UBA, Detailed Block Diagram

The *NPR control* (UBA5 and part of UBA7) contains the control flip-flops and assorted logic to sequence NPR data transfers. It also contains the paging RAM read/write control logic and parity circuits.

The *I/O read/write control* (UBA4 and part of UBA7) contains the I/O command/address decoding logic, as well as the control logic necessary to sequence I/O register read/write operations for both external and internal register addresses. It also controls interrupt vector read operations.

The *Unibus arbitrator* (UBA1) consists of a priority encoder, latches, a counter, several one-shots, and the associated logic to detect, store, initiate, and synchronize Unibus I/O, NPR, and interrupt requests. Requests for the Unibus are honored on a priority basis (highest to lowest) as follows.

1. NPR requests
2. I/O requests
3. Interrupt requests

The arbitrator logic also detects either the successful completion of a Unibus operation or the associated error conditions (i.e., TIMEOUT and/or NXD).

The *Unibus control* (UBA2) contains the control logic associated with Unibus signals MSYNC, SSYNC, BBSY, and INIT. It also controls the transmission of Unibus data and addressing information.

The *KS10 bus control* (UBA6) contains the circuitry to request the KS10 bus, initiate bus data cycles, and initiate bus command/address cycles to read/write memory. It also contains the mixer selection logic that controls the data path mixers.

5.8.4 NPR Data Transfer Operation

The essential steps in an NPR write to memory operation and an NPR read from memory operation are shown in Figures 5-40 and 5-41. With reference to the UBA circuit schematics, operation is as follows. (Note that there is a correspondence between the steps in the figures and the steps below.)

1. A Unibus device asserts the NPR line on the Unibus to signal that it has data to transfer to KS10 memory, or that it requires data from KS10 memory. More than one device may assert NPR.

    a. Unibus arbitrator – When received by the UBA, NPR asserts an input to the Unibus arbitrator's priority encoder. This causes arbitrator output flip-flop UBA1 NPG to set, which asserts NPG on the Unibus. The arbitrator asserts NPG immediately if the arbitrator is enabled (UBA1 ARB BUSY = 0); that is, if there is no Unibus priority arbitration, I/O read/write operation, or an interrupt vector read operation in progress. A previously initiated NPR data transfer may still be in progress however (Unibus BBSY = 1). Once NPG is asserted, the arbitrator is disabled (UBA1 ARB BUSY = 1). (Appendix A contains a Unibus signal summary and a description of arbitrator operation.)

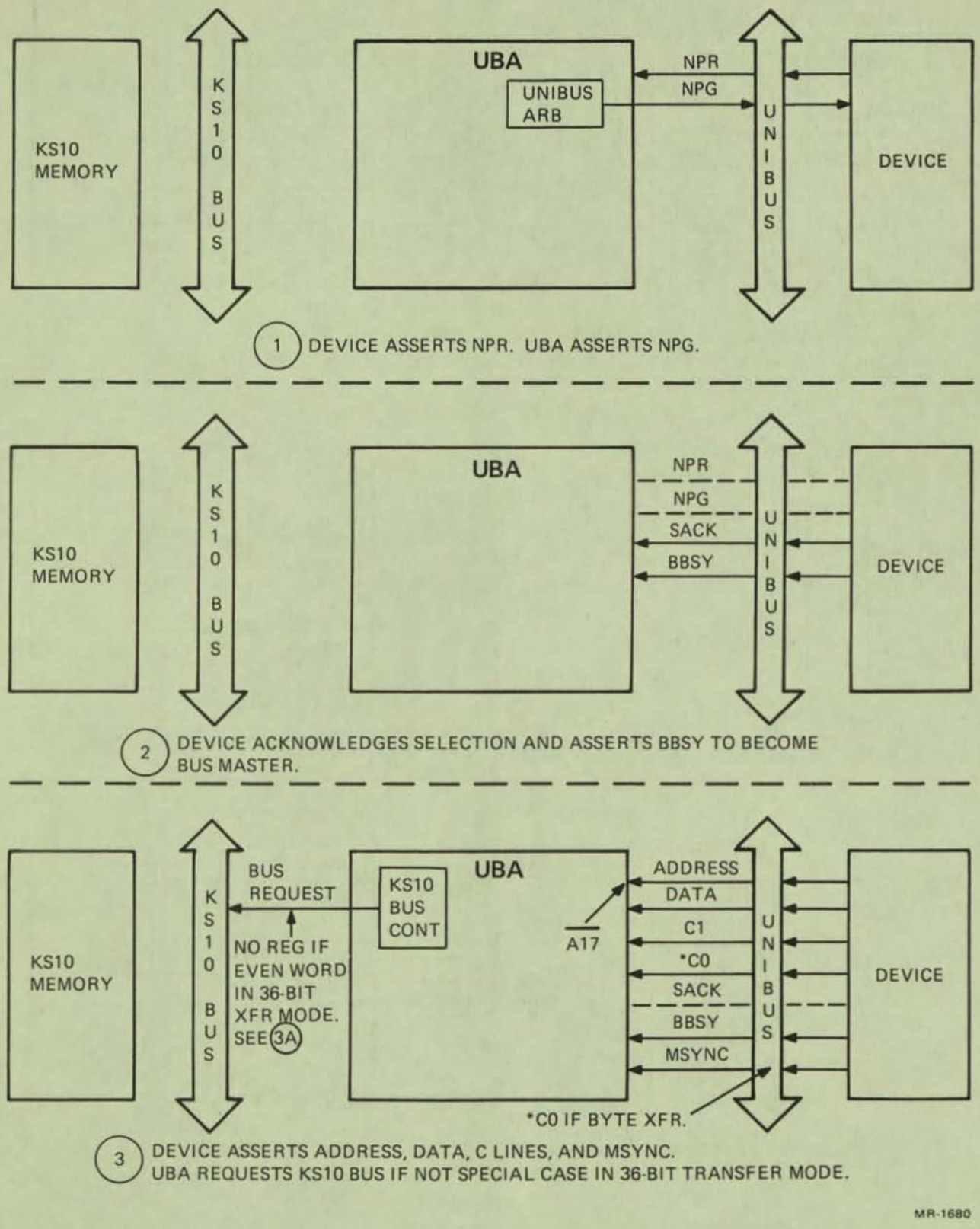


Figure 5-40 NPR Write, Bus Dialogue (Sheet 1 of 3)

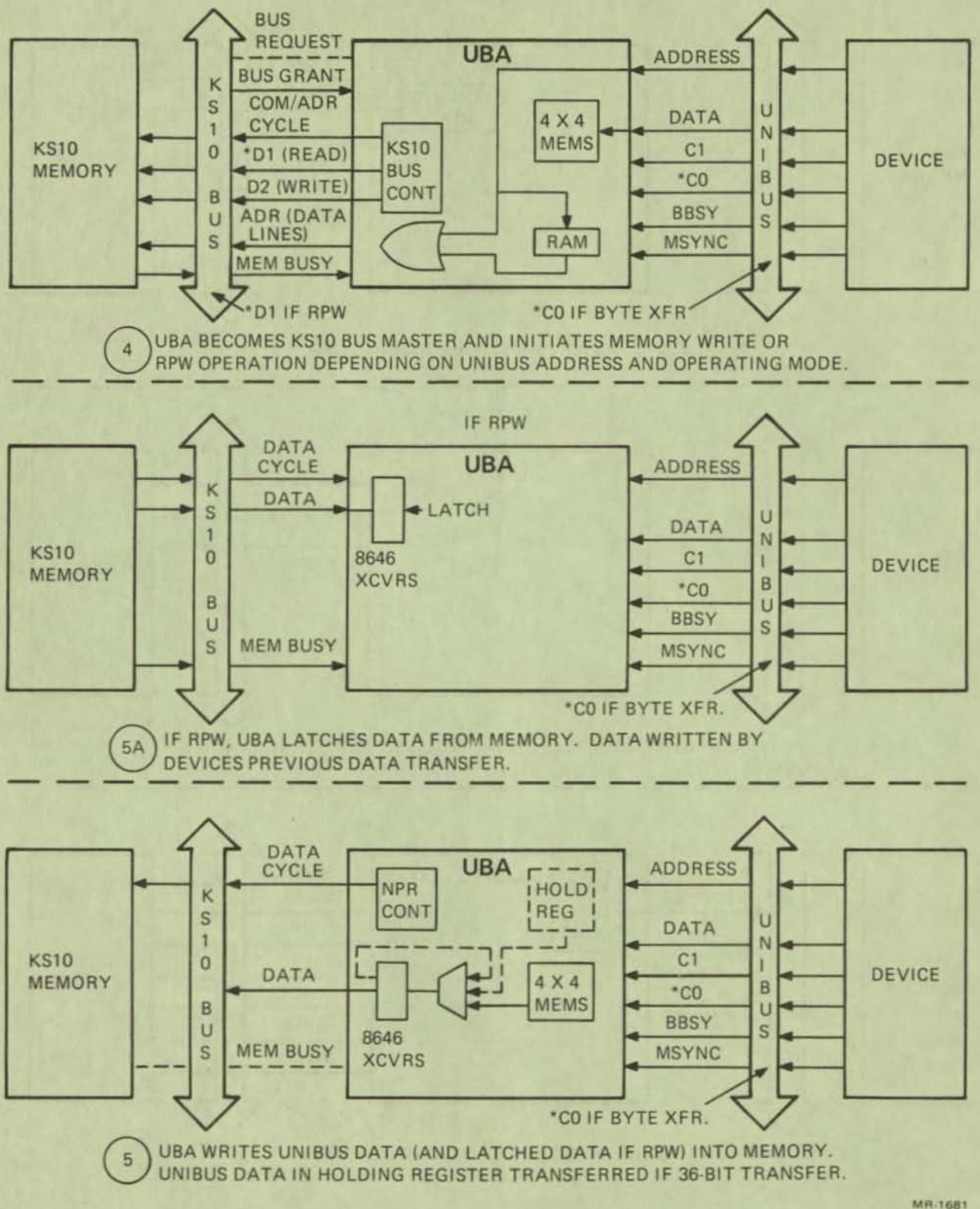


Figure 5-40 NPR Write, Bus Dialogue (Sheet 2 of 3)

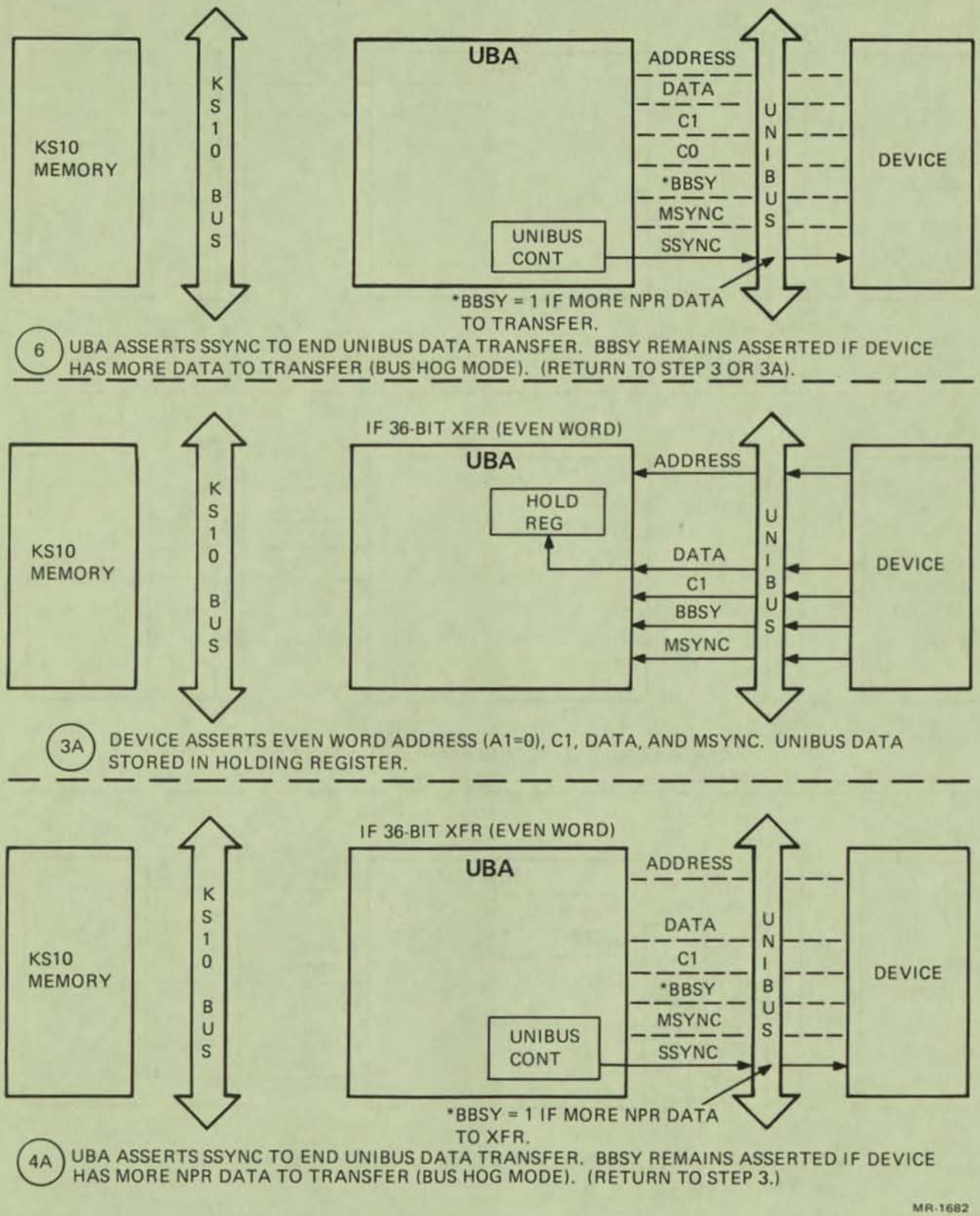


Figure 5-40 NPR Write, Bus Dialogue (Sheet 3 of 3)

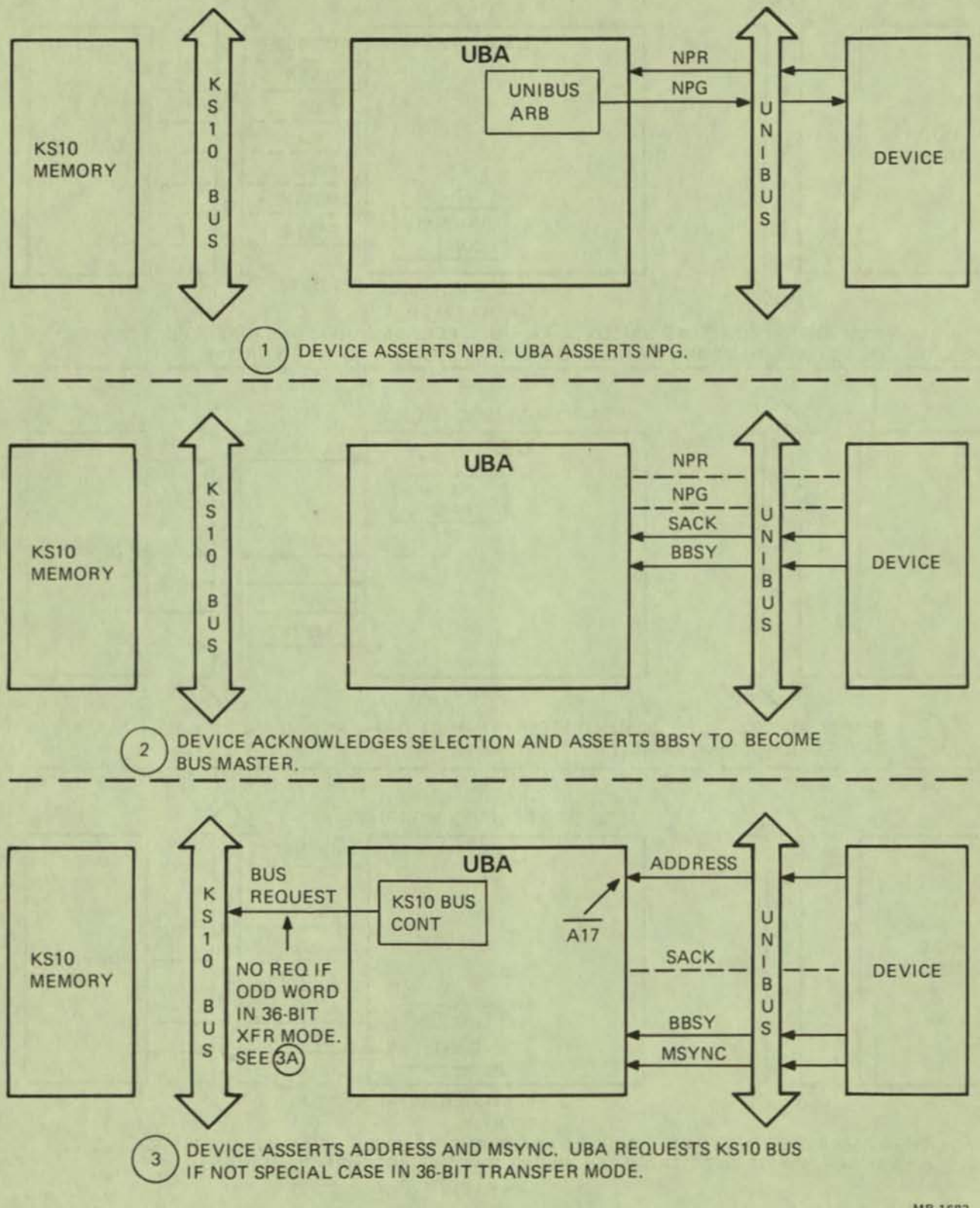


Figure 5-41 NPR Read, Bus Dialogue (Sheet 1 of 3)

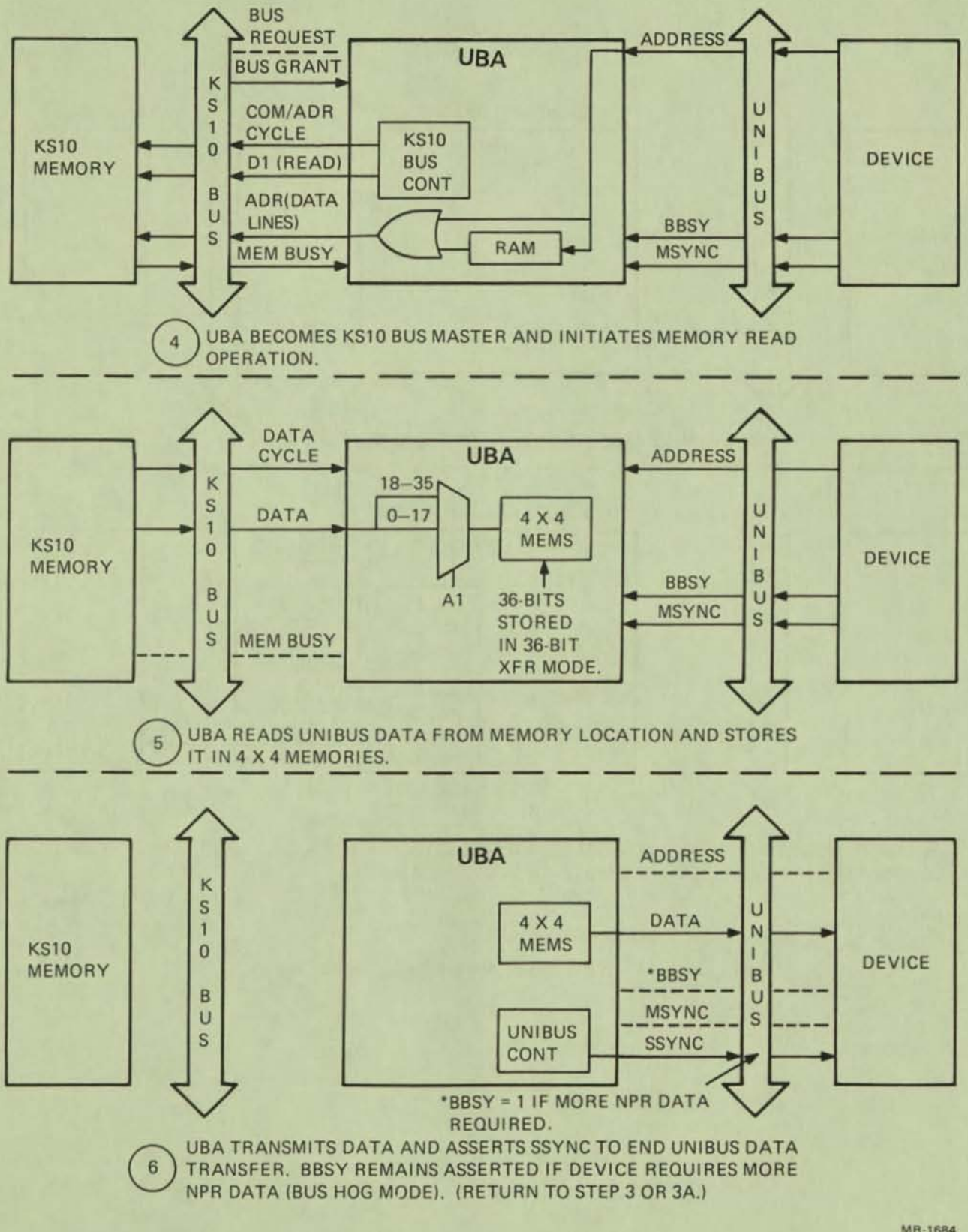


Figure 5-41 NPR Read, Bus Dialogue (Sheet 2 of 3)

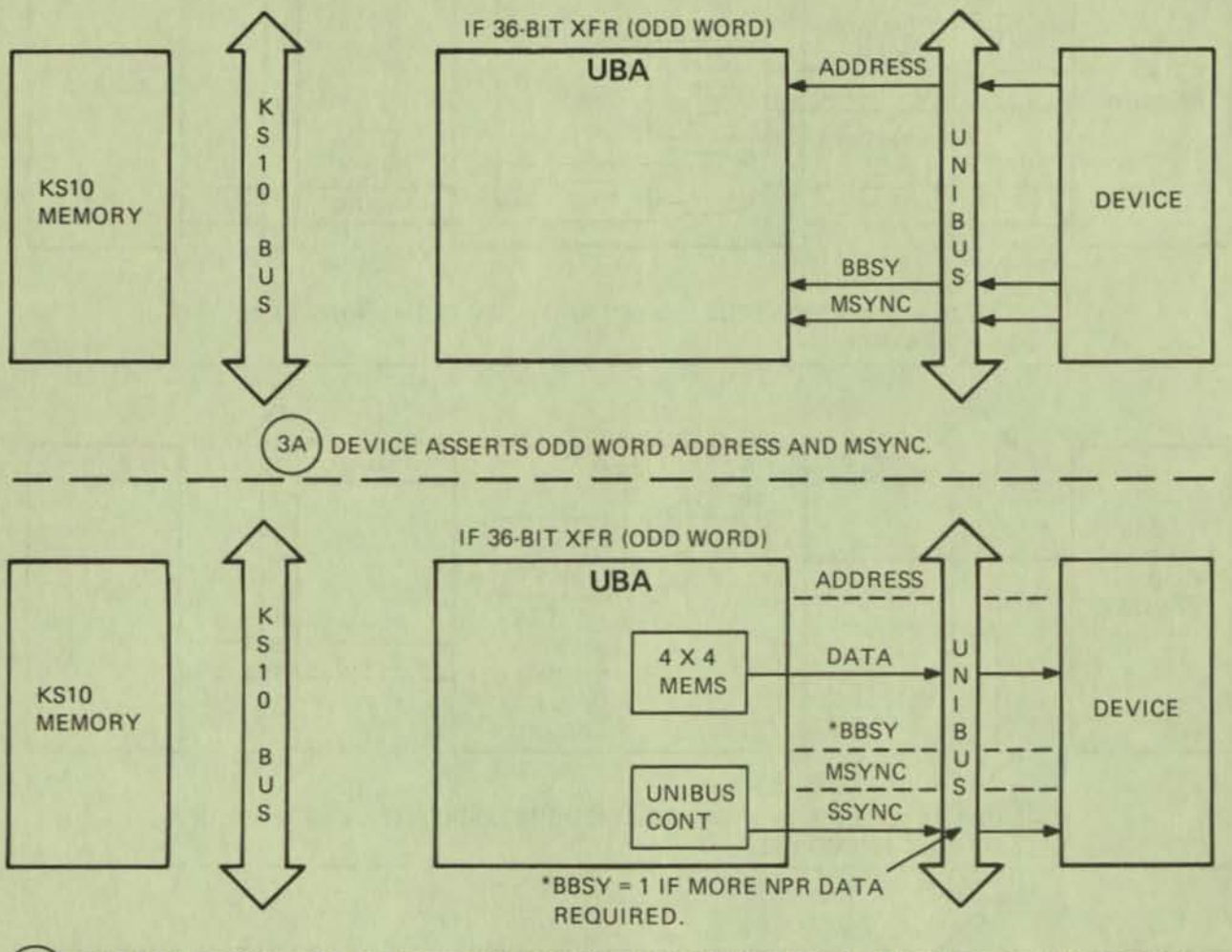


Figure 5-41 NPR Read, Bus Dialogue (Sheet 3 of 3)

2. When NPG is asserted on the Unibus, it is passed through devices not asserting NPR. However, the first device that is asserting NPR blocks the signal from proceeding down the bus, negates NPR, and asserts the Unibus SACK line to acknowledge selection. SACK causes the UBA to clear NPG.

NOTE

NPG is returned on the SACK line by the Unibus terminator if the signal is passed to the end of the bus due to a system malfunction (e.g., spurious NPR). Because SACK clears NPG, and the negation of NPG in turn clears SACK for this case only, UBA1 ARB CLR is asserted to re-enable the arbitrator and prevent a TIMEOUT error from occurring for this type of system malfunction.

After asserting SACK, and if both BBSY and SSYNC are **not** asserted on the Unibus, the device may become Unibus master by asserting BBSY. If BBSY or SSYNC is already asserted, indicating another device has the bus (NPR data transfer already active), it must wait until the Unibus is free. When the device asserting SACK becomes bus master, it negates SACK to re-enable the arbitrator in the UBA.

3. Once a device becomes Unibus master after an NPR request, it performs either a Unibus DATO or DATOB data transfer for an NPR write to memory operation, or a Unibus DATI data transfer for an NPR read from memory operation. To perform the data transfer, it transmits an address on the Unibus address (A) lines and it asserts bus control lines C0 and C1 as follows:

C0	C1	OPERATION
0	0	DATI (NPR read)
0	1	DATO (NPR write)
1	1	DATOB (NPR write-byte transfer)

The device also transmits NPR write to memory data on the Unibus data (D) lines if the data transfer is a DATO or DATOB. With the A, C, and possibly the D lines asserted, the device then asserts MSYNC on the bus to cause the following to occur in the UBA.

a. Data path – Unibus address bits A16–A11 select a paging RAM location and the page address (UBA8/9 PAGED ADR 16–26) is read out of the RAM and gated through the data path mixers (together with Unibus address bits A10–A2) to assert a 20-bit memory address at the KS10 bus transceiver inputs (data lines 16–35). The memory address is not yet transmitted on the KS10 bus.

b. KS10 bus control – MSYNC sets the first of a chain of NPR control flip-flops (UBA6 NPR MSYNC), the last of which (UBA6 NPR REQ) generates a KS10 bus request (UBA6 ADPT (N) BUS REQUEST) except for two cases in fast transfer mode. For these two cases, an even word address (A1 = 0) and a DATO by device or an odd word address (A1 = 1) and a DATI by device, the bus request is inhibited by UBA7 CLRA2 at the input to the NPR REQ flip-flop. A bus request will also be inhibited (causing a time-out by the device) if the Unibus 64K address bit is asserted (UBAC UB ADD 17 = 1), if the paging RAM valid bit is not set (UBAA PAGE VALID = 0), or if the RAM parity is incorrect (UBA5 RAM PARITY VAL = 0).

c. NPR control – As stated above, a memory request is not made for two cases in fast mode. Instead, for the DATO operation when the address is even, the Unibus data is strobed into a holding register (UBA7 DATA 00–17) by UBA7 FST D(18–35)>UB. The data is written into memory by the next NPR transfer initiated by the device (a 36-bit transfer). For the DATI operation when the address is odd, UBA7 FAST D(18–35) asserts UBA2 DATA-11B to cause the data in the 4 × 4 memories to be transmitted on the Unibus. The data was stored in the memories by the previous NPR transfer by the device (again, a 36-bit transfer). UBA7 FST (D18–35)>UB (during the DATO) and UBA7 FAST D(18–35) (during the DATI) assert UBA2 ADPTR SSYNC OUT to generate SSYNC on the Unibus and end the data transfer. (Refer to step 6.)

4. Following the KS10 bus request, the KS10 bus arbitrator on the console module grants the UBA the bus by asserting CSLI BUS GRANT ADPT. The UBA then initiates a memory operation as follows.

a. KS10 bus control – The grant signal asserts UBA6 START CA CYCLE, which asserts UBA6 T ENB to open the inputs to the KS10 bus transceivers. Also, at the next T CLK, START CA CYCLE asserts COM/ADR CYCLE on the KS10 bus. Flip-flop UBA6 MOS REF is also set, which generates UBA2 SSYNC WRT to write the data on the Unibus data lines into the 4 × 4 memories (location 0). This stores the Unibus data for subsequent transfer to KS10 memory (if DATO or DATOB by device). In addition, UBA6 MOS REF sets state flip-flop UBA6 ADPTR MOS REF to indicate that the UBA is KS10 bus master for an NPR operation.

b. Data path – With UBA6 T ENB true, the 20-bit memory address at the KS10 bus transceiver inputs is transmitted on the KS10 bus data lines at the same time as COM/ADR CYCLE is asserted. In addition, the appropriate read/write command bits are asserted as follows.

DATA 01	LINES 02	SPECIFIES
0	1	Memory write
1	0	Memory read
1	1	Memory read-pause-write (RPW)

The command bits asserted, and thus the memory operation initiated by the UBA, depends upon the Unibus operation being performed, the Unibus address, and any special operating modes set in the UBA. (Refer to Figures 5-30 and 5-31.) For example, data line 02 (the write bit) is asserted by UBA6 C1(1) because this signal indicates a DATO or DATOB is in progress and data must be transferred to memory via a write or RPW memory operation. Data line 01 (the read bit) must also be asserted if the UBA is to do a RPW, and UBAC PAUSE (ANDed with UBA6 C1(1)) does this during all odd word transfers to memory unless in fast mode (UBA6 EN D(18–35) ANDed with UBA7 FST D(0–35)>MOS (0)), during transfer of byte 1 to memory (UBA8 UB ADD 0 ANDed UBAC C0), and during an even word transfer to memory in read reverse mode (UBAB FORCE RPW). Data line 01 is also asserted by UBA6 C1(0) to initiate a memory read operation when the device is performing a DATI.

c. KS10 bus control – During assertion of the command/address information on the KS10 bus (i.e., when UBA6 ADPTR MOS REF sets), the data path mixer select levels are asserted as necessary to set up the NPR write to memory data path for the next (data transfer) portion of the NPR operation. The levels are conditioned by Unibus address bits A0 and A1, UBA7 FST D(0–35)>MOS, and UBAB FORCE RPW (read reverse mode) as shown in Table 5-4.

Table 5-4 Data Path Mixer Selection for NPR Transfers

Select Level Inputs				Select Levels				Function (See Below)
A1	A0	FST D (0-35) >MOS	FORCE RPW	USEL 2 1	UBSEL 2 1	LSEL 2 1	LBSEL 2 1	
0	0	0	0	1 0	1 0	0 0	0 0	A
0	0	1	0	1 0	1 0	0 1	0 1	B
0	1	0	0	0 1	1 0	0 0	0 0	C
1	0	–	0	0 1	0 1	1 0	1 0	D
1	0	0	1	1 1	1 1	1 0	1 0	E
1	1	0	0	0 1	0 1	0 1	1 0	F

Function	Unibus DXX to KS10 bus data lines XX	Recirculate KS10 bus data lines XX
A	D17–0 to 0–17	
B	D17–0 to 0–17	18–35
C	D17–8 to 0–9	10–17
D	D17–0 to 18–35	0–17
E	D17–0 to 18–35 (Holding Register to 0–17)	
F	D17–8 to 18–27	0–17, 28–35

For example, in read reverse mode and for an even word address (the second entry in the table), the UBA performs an RPW memory operation during the next part of the NPR transfer to first read an odd word previously stored in memory, and then to write both the odd and even word back into memory. Thus, the mixers are conditioned by UBA6 USEL 2, UBSEL 2, LSEL 1, and LBSEL 1 to recirculate the information on KS10 bus data lines 18– 35 (odd word is in right half of the KS10 data word) and to gate the even word on the Unibus data lines to KS10 bus data lines 0–17 (even word is stored in left half of KS10 data word).

5. Once the UBA initiates a memory operation by means of a KS10 bus command/address cycle, a bus data cycle is generated either by the memory (memory read operation), by the UBA (memory write operation), or by both (memory RPW operation) to transfer Unibus data to/from memory. Operation is as follows.

   a. NPR control/data path - After assertion of the command/address, UBA6 ADPTR MOS REF sets U5A5 NPR XFER A and B to begin the second (data transfer) portion of the NPR operation. For a memory write operation (UBAC C1(1) = 1 AND UBAC PAUSE = 0), NPR 2 asserts UBA5 NPR DATA>MOS, which generates UBA6 T ENB and causes BUS DATA CYCLE to be transmitted on the KS10 bus at the next T CLK. At the same time, the Unibus data stored in the 4 × 4 memories (and in the holding register during a fast mode transfer) is transmitted on the bus. For a memory read or RPW operation, BUS DATA CYCLE is generated by the memory and transmitted on the bus coincident with the data read from the MOS array. The bus data is valid one bus cycle before (and during) the BUS DATA CYCLE signal.

   When the memory operation is a read (UBA6 C1(0) = 1), UBA5 WRT DATA>UB (which is clocked on and off continuously as long as UBA5 XFER B is set) causes the received memory data to be written into the 4 × 4 memories that output to the Unibus. Because of the previously stated bus timing, the data is valid in the 4 × 4 memories prior to receiving BUS DATA CYCLE. If not in fast mode, either the left or right half of the memory data is stored (in location 0) depending on the state of select level UBA6 NPR at the 4 × 4 memories input data mixers. The state of UBA6 NPR is a function of the Unibus address; that is, whether the address is even (A1 = 0) or odd (A1 = 1). If in fast mode, all 36 bits of memory data are stored. UBA7 36 BIT WRT goes true when BUS DATA CYCLE is received to negate UBA6 SELECT DATA>UB and cause the right half of the memory data to be stored (in location 1). As when not in fast mode, the left half is stored (in location 0) prior to receiving BUS DATA CYCLE.

   When the memory operation is an RPW, the received memory data is not stored in the UBAs 4 × 4 memories. Instead, part of the data word is recirculated in the data path mixers. (The data recirculated and the mixer select levels asserted are indicated in Table 5-4.) BUS DATA CYCLE then sets UBA5 PSE WRT GO which inhibits UBA6 R CLK A and B. This latches the recirculating memory data in the KS10 bus transceivers so that it may be written back into the addressed memory location together with the Unibus data stored in the 4 × 4 memories. To write the data, NPR Q asserts UBA5 NPR DATA>MOS. Similar to the memory write operation, NPR DATA> MOS then causes the data and BUS DATA CYCLE to be transmitted on the KS10 bus when the next T CLK occurs.

6. Following the memory write, read, or RPW operation, the Unibus data transfer operation is terminated as follows.

   a. Unibus control - At the same time that UBA5 NPR DATA>MOS is asserted to transmit data on the KS10 bus during a memory write operation, UBA5 WRT is also asserted to set UBA5 UB NPR DONE after a delay (three T CLKS). UB NPR DONE then asserts UBA2 ADPTR SSYNC OUT to transmit SSYNC on the Unibus and end the device DATO or DATOB operation. During an RPW, UBA5 PSE WRT GO asserts UBA2 ADPTR SSYNC OUT to generate SSYNC and end the DATO or DATOB operation. In this case, the SSYNC signal will not be generated (causing a timeout error by the device) if bad data had been read from memory (UBA4 BD MOS DATA = 1). To terminate a device DATI operation following a memory read, the 0 output of flip-flop UBA5 REC KS DATA is used to assert UBA2 ADPTR SSYNC OUT. (REC KS DATA is cleared by R CLK shortly after being set by BUS DATA

CYCLE.) Similar to the RPW operation, SSYNC is not generated when bad data has been read from memory, but only when the DISABLE TRANSFER control bit has been set by the program. This control bit (UBA3 DIS XFER), which causes UBA4 DIS BD NPR XFER to inhibit ADPTR SSYNC OUT when the bad data is detected, is normally set by the program except for special cases during error recovery routines. Following the memory read operation, UBA2 DATA>UNIBUS is asserted at the same time as SSYNC to transmit the memory data stored in the 4 × 4 memories onto the Unibus.

When the device receives SSYNC following the memory operation initiated by the UBA, it ends the Unibus data transfer by either strobing the data lines (DATI by device) or by negating the data lines (DATO or DATOB by device), and by negating the address lines, C lines, and MSYNC. The device also negates BBSY to relinquish Unibus mastership unless more NPR data is to be transferred immediately (i.e., device operating in bus hog mode). In this case, the device continues to assert BBSY and it begins another Unibus data transfer, as described in step 3, by asserting the next address, the next data word or byte (if DATO or DATOB), the appropriate C lines, and MSYNC.

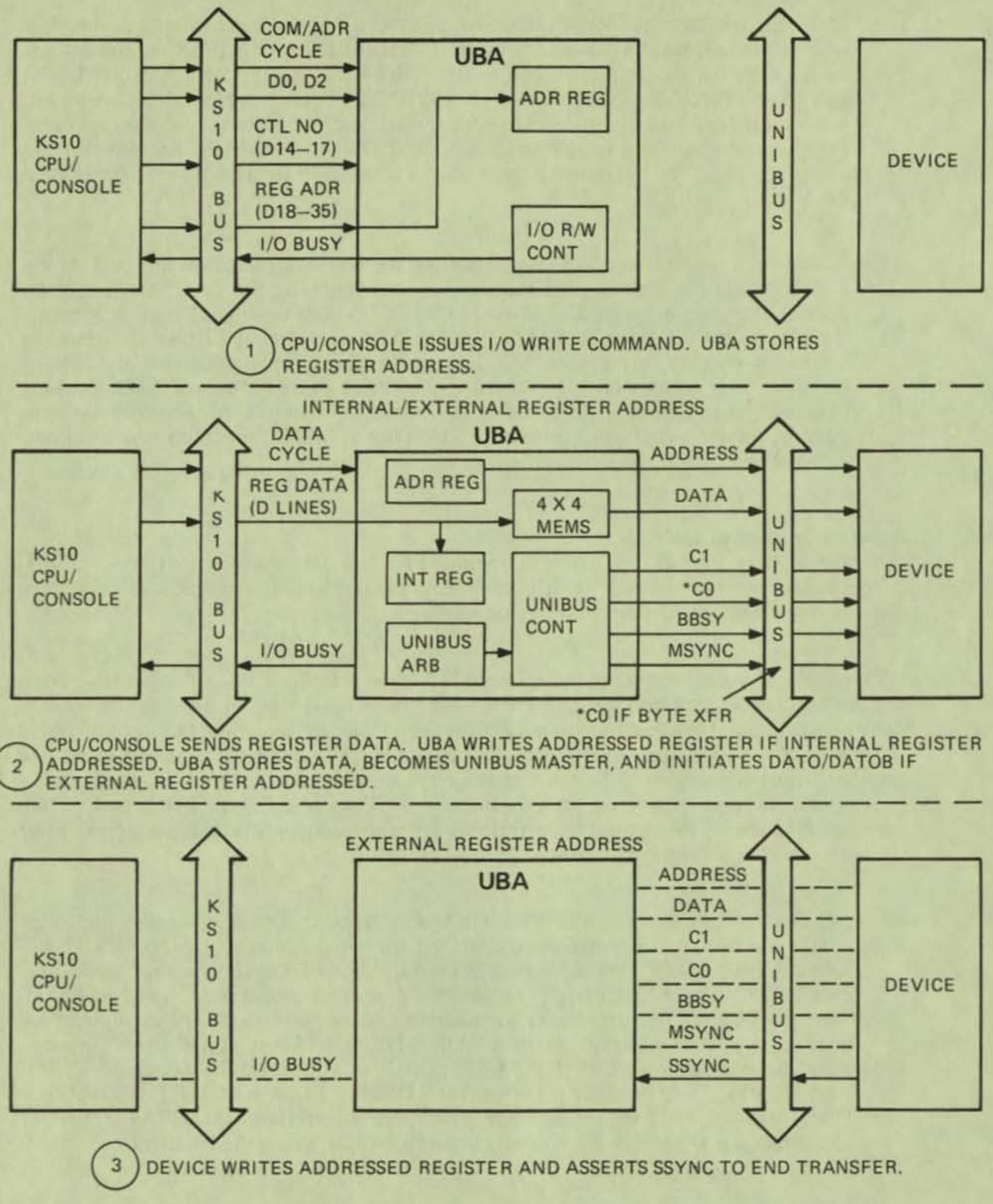
5.8.5 I/O Data Transfer Operation

Figures 5-42 and 5-43 show the basic sequence of operation for I/O data transfers to/from the UBA. A description of UBA operation follows. Refer to the UBA circuit schematics and note that the steps in the description correspond to the steps in the associated figures.

1. When ready to write or read an addressable I/O register in the UBA (an internal register) or in a Unibus device connected to the UBA (an external register), the CPU or console performs a command/address operation on the KS10 bus by asserting bus control signal COM/ADR CYCLE, data line 00 (the I/O command bit), either data line 01 or 02 (the read or write command bit), the UBA controller number on data lines 14–17, and an internal or external register address on data lines 18–35. Data line 06 (the byte transfer command bit) is also asserted together with the write command bit if a byte transfer is to be made to an external register address.

   a. I/O read/write control – When the UBA is addressed; that is, when the controller number on the data lines matches the UBAs hard-wired address, COM/ADR CYCLE asserts UBA7 COM/ADR ENABLE. Flip-flop UBA4 I/O ADD is then set by the next T CLK to strobe the output of a command/address decoder circuit and set one of four control flip-flops that specify the operation to be performed. For example, if an internal register is addressed (UBA4 ADR ADPTR REG = 1) and the write command bit is asserted (UBAC REC KSBUS BIT = 1), control flip-flop UBA4 WRT ADPTR REG is set. Similarly, UBA4 RD ADPTR REG, UBA4 WRT UB REG, or UBA4 RD UB REG are set depending on the command/address. UBA4 I/O ADD also direct sets UBA5 I/O BUSY, which asserts I/O BUSY on the KS10 bus.

   b. Data path – In addition to asserting I/O BUSY and setting the control flip-flop that specifies the operation, UBA4 I/O ADD clocks an address register that stores the register address on KS10 bus data lines 18–35. The write, read, and byte transfer command bits are also stored in flip-flops UBA9 I/O REG READ, UBA9 I/O REG WRT, and UBAA BYTE CYCLE at the same time.



Pigure 5-42 I/O Write, Bus Dialogue

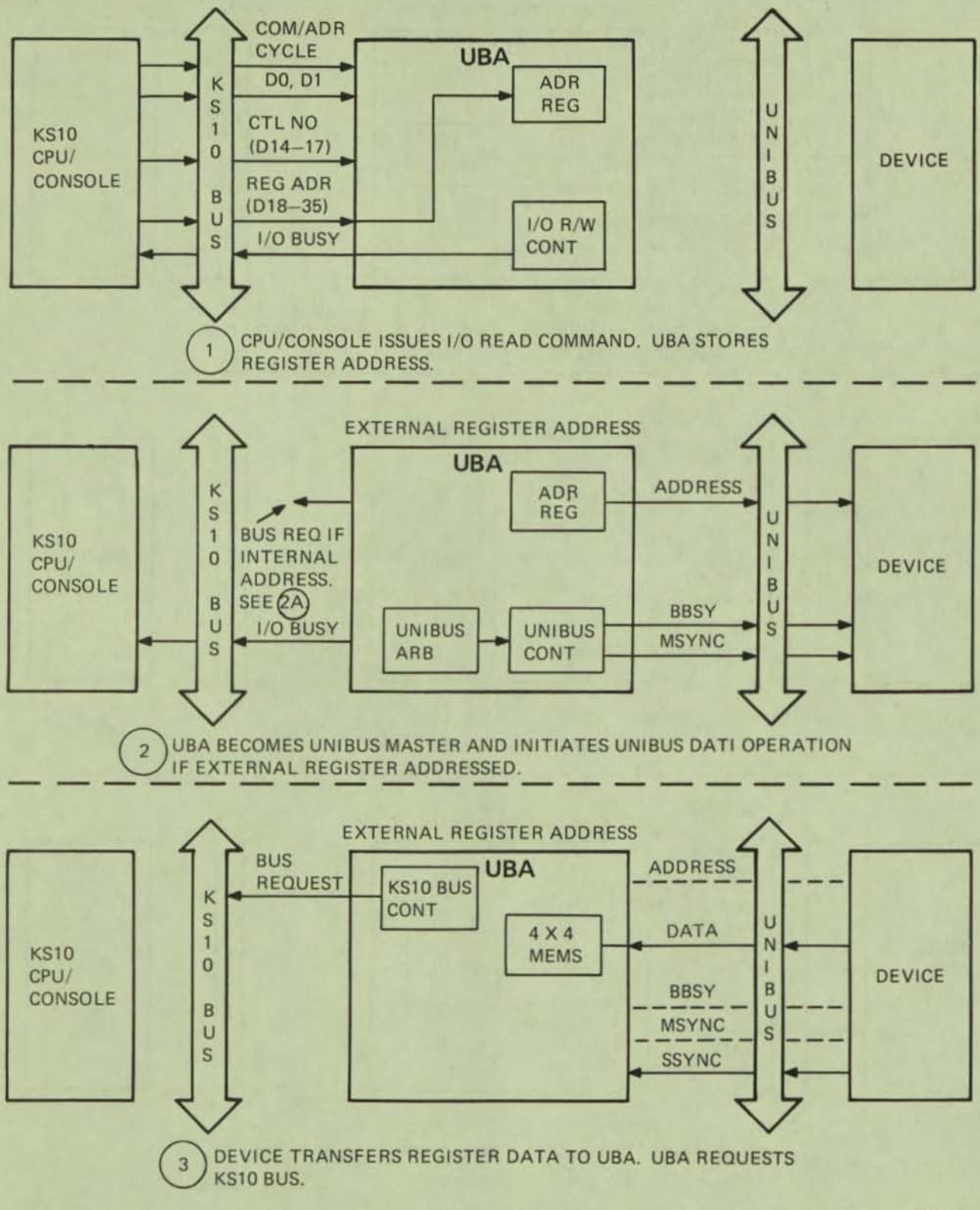


Figure 5-43 I/O Read, Bus Dialogue (Sheet 1 of 2)

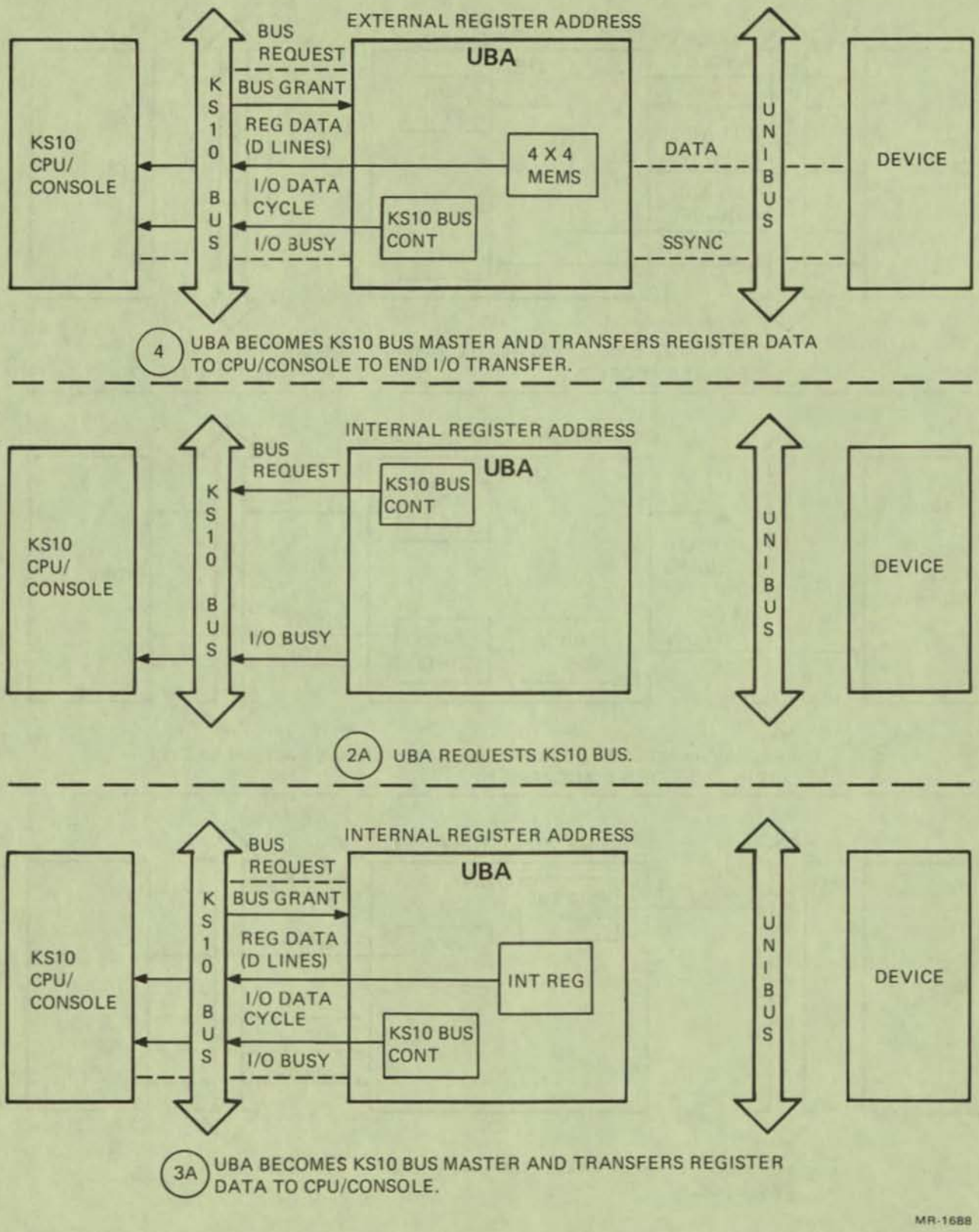


Figure 5-43 I/O Read, Bus Dialogue (Sheet 2 of 2)

2. If the I/O transfer is a register write operation, the CPU or console performs a bus data cycle after the command/address cycle (without requesting the bus again) to transfer the register data to the UBA. If the transfer is a register read operation, the UBA performs the bus data cycle, but not during the KS10 bus cycles allotted the CPU or console. (The register data cannot be read in that short a time period.) Instead, the bus is requested by the UBA and the data cycle performed at a later time when the register data is available for transfer.

   a. I/O read/write control - For an internal register write operation (UBA4 WRT ADPTR REG(1) = 1), BUS DATA CYCLE from the CPU or console sets UBA5 STA/MNT WRT. This signal, ANDed with the appropriate output from register address decoder circuits (e.g., UBA4 STATUS, etc.) acts as a data strobe to load the UBA's status and maintenance registers directly from the KS10 bus data lines. UBA5 ADPTR WRT CLR is also set by the same enable level as UBA5 STA/MNT WRT. This signal generates write pulse UBA4 WRT RAM to load the RAM from the data lines when a paging RAM location is addressed. With UBA5 STA/MNT WRT set, I/O BUSY is negated on the KS10 bus signaling the end of the I/O transfer.

   b. KS10 bus control - For an internal register read operation (UBA4 RD ADPTR REG(1) = 1), a KS10 bus request is generated and UBA6 START I/O DATA CYC is set when the UBA is granted the bus. This asserts the appropriate data path mixer select levels if the status register is addressed (UBA6 LSEL 2 and 1, and UBA6 LBSEL 2 and 1). (No select levels are asserted to read the paging RAM.) UBA6 START I/O DATA CYC also asserts UBA6 T ENB, which opens the KS10 bus transceiver inputs and (at the next T CLK) generates I/O DATA CYCLE to cause the internal register data to be transferred to the CPU or console via a KS10 bus data cycle. I/O BUSY is also negated on the bus to signal the end of the I/O transfer.

   c. Unibus arbitrator/data path - For an external register address, a Unibus data transfer operation must be initiated, and UBA4 WRT UB REG(1) or UBA4 RD UB REG(1) sets flip-flop UBA4 ADPTR UB REG to assert an input to the Unibus arbitrator. (Also, if the operation is an external register write, the data received on the KS10 bus must be stored in the 4 × 4 memories that output to the Unibus, and BUS DATA CYCLE sets UBA5 WRT DATA>UB to write the bus data into location 2.) With UBA4 ADPTR UB REG set, arbitrator output flip-flop UBA1 ADPTR UB MSTR is set to begin a Unibus data transfer if and when the arbitrator is enabled, an NPR request is not asserted by a device, and the Unibus is not active. (BBSY or SSYNC = 1).

   d. Unibus control - Once set, UBA1 ADPTR UB MSTR starts a Unibus data transfer by clocking on UBA2 ADR>UNIBUS. This flip-flop asserts BBSY on the Unibus (UBA now Unibus master) and it enables the UBAs Unibus address line transmitters so that the register address held in the address register is transmitted on the bus. Also, if the operation is a register write (UBA9 I/O REG WRT = 1), Unibus control line C1 is asserted and UBA2 DATA>UNIBUS goes true to enable the Unibus data line transmitters and causes the data previously loaded from the KS10 bus into the 4 × 4 memories to be transmitted on the bus. Unibus control line C0 is also asserted if the I/O transfer is a byte operation (UBAA BYTE CYCLE = 1). Similar to an NPR operation, the control lines specify the Unibus data transfer as follows.

C0	C1	OPERATION
0	0	DATI (I/O read)
0	1	DATO (I/O write)
1	1	DATOB (I/O write-byte transfer)

With the register address and BBSY asserted on the Unibus (together with the register data and control lines if the transfer is a DATO/DATOB), UBA2 ADR>UNIBUS sets UBA2 MSYNC after a 175 ns delay to assert MSYNC. The MSYNC line signals the Unibus device to read or write the addressed register as specified by C1 and C0.

3. After the device receives MSYNC on the Unibus, it either strobes the data lines to write the addressed register (DATO/DATOB operation) or it reads the addressed register and transmits the contents on the data lines (DATI operation). It also asserts SSYNC on the Unibus to signal that the Unibus data has been received or sent. In the UBA, the following occurs.

   a. Unibus control – SSYNC asserts UBA2 SSYNC WRT to write the information on the Unibus data lines into the 4 × 4 memories (location 2). This stores the register data transmitted by the device if the transfer is a DATI. SSYNC also clears UBA2 MSYNC to negate MSYNC on the bus, and it asserts UBA2 ADPTR END. When received by the device, the trailing edge of MSYNC causes SSYNC (and the data lines if the transfer is a DATI) to be negated on the bus.

   b. Unibus arbitrator – UBA2 ADPTR END causes the next T CLK to set UBA1 ADPTR DONE. This flip-flop ends the Unibus data transfer in the UBA by re-enabling the Unibus arbitrator and clearing UBA2 ADR>UNIBUS. BBSY and the Unibus address lines are then negated, as well as the control and data lines if the operation is a DATO/DATOB. The termination of a DATO/DATOB ends an external register write operation, and UBA2 ADPTR DONE negates I/O BUSY on the KS10 bus to indicate the I/O transfer has completed. However, for a DATI, the data collected from the Unibus by the UBA must be transferred to the CPU or console before the I/O transfer completes.

   c. KS10 bus control/data path – To transfer the data read by the Unibus DATI operation, UBA2 ADPTR DONE first asserts a KS10 bus request. When the bus is granted, UBA6 ADPTR MOS REF is set (similar to an internal register read operation) to assert the appropriate data path mixer select levels (UBA6 LSEL 2 and LBSEL 2 for an external register read) and UBA6 T ENB. When the next T CLK occurs, I/O DATA CYCLE and the register data in the 4 × 4 memories is transmitted on the KS10 bus. UBA6 ADPTR MOS REF also clears I/O BUSY on the KS10 bus to signal the end of the I/O transfer.

5.8.6 PI Operation

Figure 5-44 shows the basic steps associated with servicing a Unibus device interrupt request and transferring the interrupt vector to the CPU. With reference to the UBA circuit schematics, operation is as follows.

1. A device makes an interrupt request by asserting its assigned BR level on the Unibus (one of BR4–7). More than one BR level may be asserted at one time by the various Unibus devices, and more than one device can assert the same BR level.

   a. Interrupt control – When received by the UBA, a BR level is synchronized to T CLK and (if a vector is not already being read by the CPU) it asserts one of seven PI

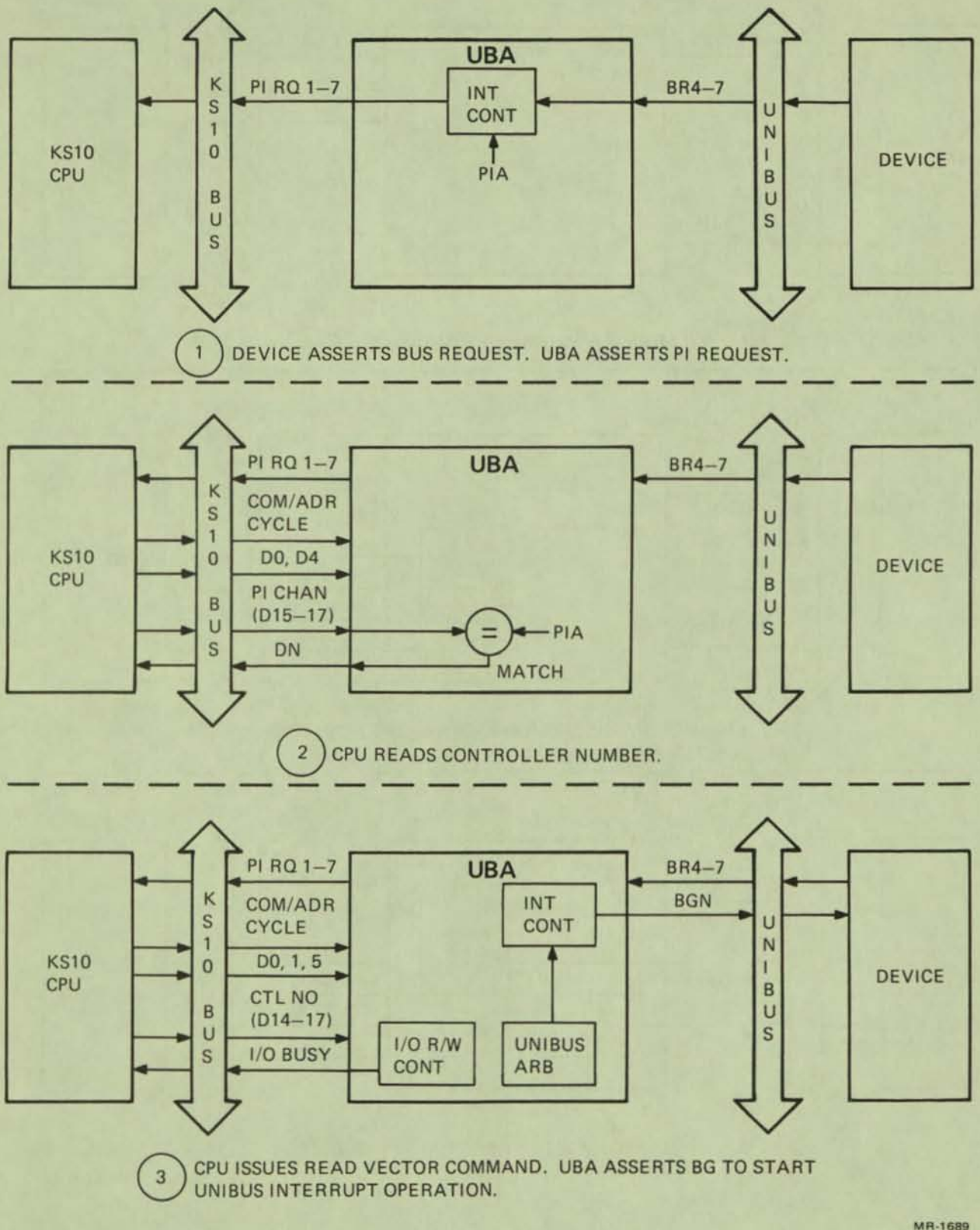


Figure 5-44 PI Operation, Bus Dialogue (Sheet 1 of 2)

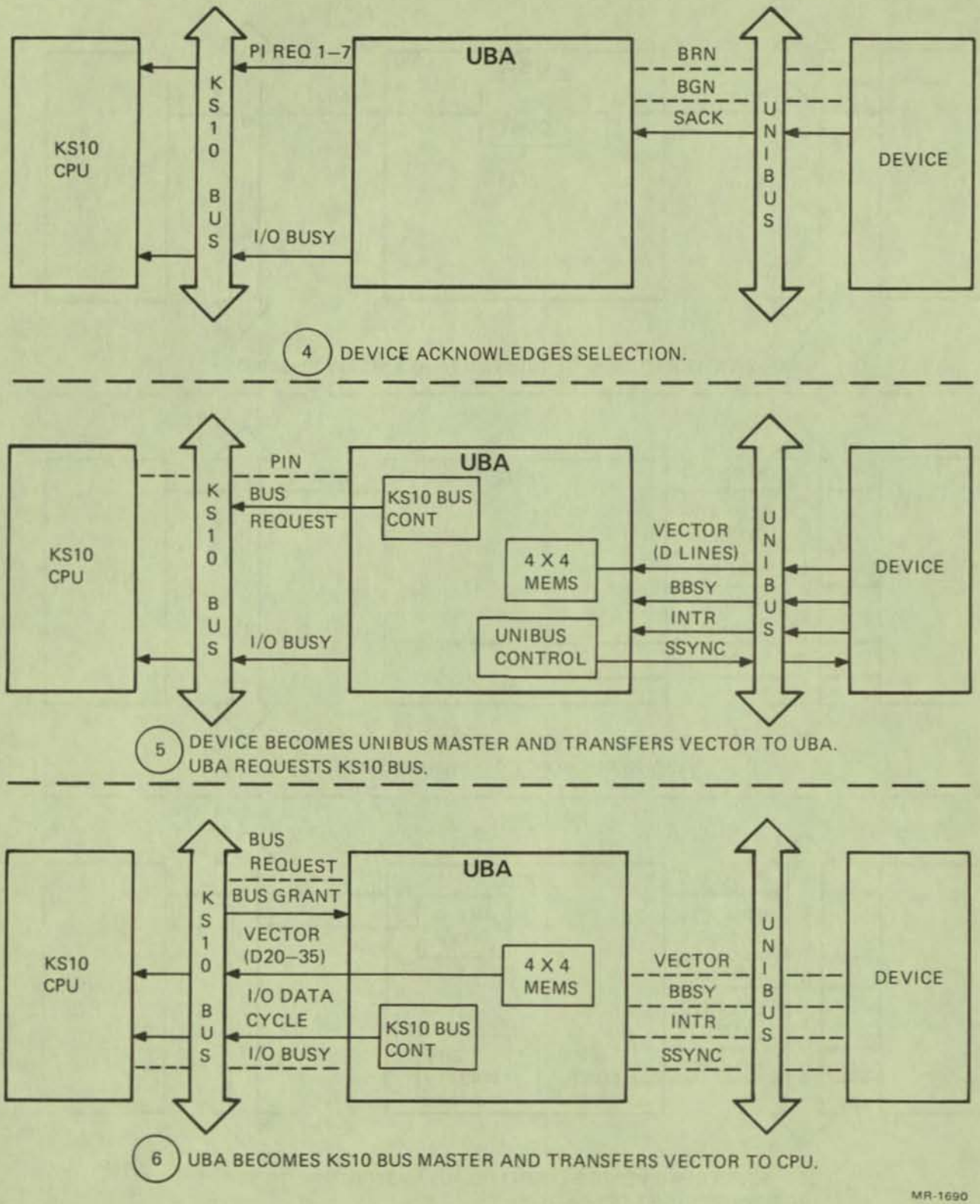


Figure 5-44 PI Operation, Bus Dialogue (Sheet 2 of 2)

requests on the KS10 bus (BUS PI REQ 1–7) depending on the associated PIA (PI channel 1–7) stored in the UBA's status register. There is one (high level) PIA associated with BR6 and 7 (UBA3 PIH2–0) and another (low level) PIA associated with BR 6 and 7 (UBA3 PIL2–0). Thus, at any one time, the UBA can assert up to two PI REQ levels. Also, two BR levels can assert a single PI REQ level.

2. The CPU, when ready to service an interrupt, resolves PI channel number priority for the PI REQ signals that are asserted on the KS10 bus, and then initiates a KS10 bus operation to read the controller numbers of the I/O controllers (i.e., the UBAs) interrupting on the highest priority channel. (More than one I/O controller can assert the same PI REQ line.) The CPU initiates the KS10 bus operation by transmitting a command/address; that is, it asserts BUS COM/ADR CYCLE, data line 00 (I/O command bit), data line 04 (read device number command bit), and the PI channel number to be serviced on data lines 15–17.

   a. Interrupt control – If interrupting on either the high level PIA (UBA3 BR6/7 INT RQ = 1) or the low level PIA (UBA3 BR4/5 INT RQ = 1) and if the PI channel to be served (on data lines 15–17) matches the corresponding PIA, a UBA responds to the command/address by asserting KS10 bus transmitter UBA7 PI REQ ADPT(N) for one bus cycle. (UBA7 BG HI is also set if a match occurs for the high level PIA.) The transmitter output is jumpered via the backplane to the KS10 bus data line corresponding to the UBAs controller number. Jumpering is such that UBA1 (located in module slot 19) asserts data line 19, and UBA3 (in slot 16) asserts data line 21. The CPU then strobes the data lines (during the second bus cycle following the command/address cycle) to end the KS10 bus operation.

3. After the CPU has read the interrupting controller number, it initiates another KS10 bus operation to read the interrupt vector from the highest priority controller. (UBA1 has higher priority than UBA3.) Again, the CPU executes a command/address cycle, asserting data line 00 (I/O command bit), data line 01 (read command bit), data line 05 (read vector command bit) and the selected controller number on data lines 14–17.

   a. I/O read/write control – As for an I/O transfer, BUS COM/ADR CYCLE asserts UBA7 COM/ADR ENB when the UBA has been addressed, and this signal sets UBA4 I/O ADD to direct set UBA5 I/O BUSY and assert the I/O BUSY signal on the KS10 bus. For the read vector command, UBA7 START VEC CYCLE is also asserted coincident with UBA7 COM/ADR ENB.

   b. Unibus arbitrator – To read the vector from an interrupting device, the UBA must allow a Unibus interrupt operation to take place. Thus, UBA7 START VEC CYCLE causes Unibus arbitrator input UBA1 VECTOR REQ to be set by UBA4 I/O ADD. The arbitrator input first latches the second rank of flip-flops synchronizing the BR levels to T CLK. (This stores the current BRs and freezes the PI REQ logic during the subsequent read vector operation.) Then, if the arbitrator is enabled and there is no request pending for a Unibus NPR or I/O transfer, the arbitrator input asserts arbitrator output flip-flop UBA1 BG to start the interrupt operation. UBA1 BG does this by asserting the Unibus BG level (one of BG4–7) that corresponds to the BR to be serviced.

c. Interrupt control – The BG level asserted by UBA1 BG depends first upon the PIA (high or low level) being served by the CPU; that is, it depends upon whether UBA7 BG HI was set or cleared when the interrupting controller numbers were read previously by the CPU. For example, if BR7 caused the PI request (i.e., UBA7 BG HI = 1), the BR7 level is gated from the second rank of synchronizing flip-flops and is clocked into flip-flop UBA3 B BG7 to assert BG7 on the Unibus. If a low level BR is also asserted, it is inhibited from asserting the corresponding BG level by UBA7 BG LOW = 0. (BG LOW is the complement of BG HI.) (Note that for the special case when there are both high and low level interrupts, and both the high and low level PIAs have the same value, BG HI will be set to give the high level interrupt the highest priority.) The second factor determining which BG level will be asserted is that the highest numbered BR (for either the high or low level PIA) has the highest priority. For example, if BG LOW = 1 and both BG4 and BG5 are asserted, gating at the input to the BG output flip-flops is such that only UBA3 B BG5 is set. In any case, there is only one BG level asserted on the Unibus to start the interrupt operation.

4. Similar to the Unibus NPG signal, the asserted BG signal is passed along the Unibus by each device not asserting the associated BR level. However, the first device that is asserting the associated BR level blocks the BG signal, negates its BR, and asserts SACK to acknowledge selection. Thus, when more than one device is asserting the same BR line, the device electrically nearest the UBA has the higher priority. In the UBA, SACK asserts UBA3 SACK CLR to clear the flip-flop asserting the BG level on the Unibus.

5. When the device detects the negation of the BG level on the Unibus, and if the Unibus is not already active, it asserts BBSY to become Unibus master, transmits the interrupt vector on the data lines, and asserts the INTERRUPT control line. It then negates SACK. The following occurs in the UBA.

   a. Interrupt control – The INTERRUPT signal, when received by the UBA, clears UBA1 VECTOR REQ to unlatch the second rank of synchronizing flip-flops holding the BR levels. With the BR level being served now negated by the device, and if no other device is asserting the same BR signal, the associated PI REQ on the KS10 bus will go false.

   b. Unibus control/data path – The INTERRUPT signal also asserts UBA2 ADPTR SSYNC OUT to cause the UBA to assert SSYNC on the Unibus. In addition, UBA2 SSYNC WRT and UBA2 ADPTR END are asserted as during an I/O transfer. The SSYNC WRT signal loads the interrupt vector on the data lines into the 4 × 4 memories (location 2). The ADPTR END signal sets UBA1 ADPTR DONE to re-enable the Unibus arbitrator. When the device receives SSYNC, it negates the data lines, BBSY, and the INTERRUPT line. The trailing edge of INTERRUPT then causes the UBA to drop SSYNC, thus ending the Unibus interrupt operation.

6. Once the interrupt vector is stored in the UBA, it must be transferred to the CPU as follows.

   a. KS10 bus control – As for an I/O register read operation, UBA1 ADPTR DONE generates a KS10 bus request. When the bus is granted, UBA6 START I/O DATA CYCLE asserts UBA6 T ENB and causes I/O DATA CYCLE and the interrupt vector in the 4 × 4 memories to be transmitted on the KS10 bus when the next T CLK occurs. UBA6 START I/O DATA CYCLE also clears I/O BUSY on the KS10 bus to signal the end of the interrupt vector transfer.

5.8.7 Wraparound Data Transfer

For maintenance purposes, I/O write data transferred from the CPU or console to the UBA and asserted on the Unibus may be looped back through the UBA via the NPR data path and written in a KS10 memory location. Also, data may be read from memory and asserted on the Unibus via the NPR data path, and then transferred to the CPU or console as I/O read data. Wraparound data transfers allow most of the UBA's data path and control logic to be checked out independent of a Unibus device.

To perform a wraparound data transfer, CHANGE NPR ADR (bit 35) in the UBA's maintenance register (763101) must be set first. This bit conditions the 4 × 4 memory addressing logic so that only the locations normally used for NPR transfers are utilized. (During a wraparound data transfer, the same 4 × 4 memory locations must be accessed by both the I/O and NPR data transfer logic or the data transmitted on the Unibus will not be looped back to the KS10 bus as required.)

Next, to initiate the wraparound data transfer, an I/O register read or write command is issued to the UBA with the most significant address bit equal to 0. The UBA decodes this address as an external address, and it is asserted on the Unibus address lines as for a normal I/O transfer. However, with the most significant address bit (A17) equal to 0, no Unibus device will respond because the address is not a valid register address. The address is a valid address for an NPR data transfer however, and together with the MSYNC signal (also asserted by the UBA), it causes an NPR operation to take place in the UBA concurrent with the I/O transfer. Operation is as follows:

1. As stated previously, an I/O transfer is initiated by the CPU or console to begin the operation. The command/address is received and the I/O transfer is started as described in Paragraph 5.8.5, and as shown in Figures 5-42 and 5-43 (step 1). The I/O address is decoded as an internal address (i.e., UBA4 ADR ADPTR REG = 0).

2. Next, the address lines (and the data and control lines if the operation is an I/O write), BBSY, and MSYNC are asserted on the Unibus by the UBA, again as described in Paragraph 5.8.5 and as shown in the associated figures (step 2). Because the Unibus signals transmitted by the UBA are also received by the UBA, and because A17 (the 64K address bit) = 0, the MSYNC signal now starts an NPR operation in the UBA (by setting UBA6 NPR MSYNC) while the I/O transfer logic is hung waiting for a Unibus SSYNC signal.

3. The UBA now performs the NPR operation as described in Paragraph 5.8.4, and as shown in Figures 5-40 and 5-41 (steps 3–6). The Unibus address (loaded by the I/O write) is translated by the paging RAM to a 20-bit KS10 memory address and, if an I/O write has initiated the wraparound transfer, an NPR write to memory operation is performed. That is, the data asserted on the Unibus (loaded by the I/O write into the 4 × 4 memories that output to the Unibus) is loaded into the 4 × 4 memories that receive data on the Unibus. (The Unibus data is loaded in location 0 instead of location 2 as in normal NPR operation.) The looped-back data is then deposited in memory via a memory write or RPW cycle. If an I/O read has initiated the wraparound data transfer, a memory read operation is performed and the data fetched from memory is stored in the 4 × 4 memories that output to the Unibus. Following the memory operations (read, write, or RPW), the UBA asserts SSYNC on the Unibus to signal the end of the NPR operation in the UBA.

   The SSYNC signal also ends the I/O write operation (and the wraparound transfer) if it has initiated the NPR transfer. As described in Paragraph 5.8.5 (step 3), the I/O write is waiting only for a Unibus SSYNC signal to terminate. If an I/O read operation has initiated the NPR operation, SSYNC completes the wraparound of data by causing the fetched memory data stored in the 4 × 4 memories transmitting on the Unibus to be stored in the 4 × 4 memories receiving data on the bus. As described in Paragraph 5.8.5 (step 4), the looped-back data is transferred to the CPU or console via a KS10 bus data cycle to complete the I/O read operation and the wraparound transfer.

5.9 KS10 POWER SYSTEM

A simplified block diagram of the KS10 power distribution system is shown in Figure 5-45. Input power requirements and specifications are given below.

Device	Line Voltage	Freq.	RMS Current	Surge Current	Surge Duration	KVA
KS10-AA	104-126 Vac	60 Hz	9.90 A	25 A	6 cycles	1.14 KVA
KS10-AB	207-253 Vac	50 Hz	4.95 A	12.5 A	6 cycles	1.14 KVA

The major power system components are as follows.

1. 861-C (60 Hz) or 861-B (50 Hz) power control.
2. H7130 power supply and 5413261 power distribution module.
3. H765A (60 Hz) or H765B (50 Hz) power supply.
4. Blower for CPU and memory.

The H7130 power supply is used to power the KS10PA card cage. The H765A power supply is used to power the BA11-KE drawer (115 Vac version); the H765B power supply is used to power the BA11-KF drawer (230 Vac version).

The location of the major power system components are given in Chapter 1 and on the KS10 Unit Assembly drawing.

5.9.1 861 Power Control

The 861 controls and distributes power in the KS10 cabinet. It performs the following functions.

1. Controls large amounts of power with low signal power.
2. Provides a convenient ac power distribution point.
3. Filters out electrical noise on the ac lines.
4. Disconnects power for servicing and in case of overload or overtemperature.

The 861 consists of four switched duplex outlets, two unswitched duplex outlets, a contactor with associated control circuitry, a circuit breaker, and a thermoswitch. All components are contained in a 19-in. rack-mounted box. The unit is supplied with 15 feet of power input cable with a suitable connector.

NOTE
Loads external to the KS10 cabinet are NOT to be plugged into the 861 power control.

Input power for the 861 is as follows.

Power Control	Voltage	Current (Maximum)	Phase
861-B	180 V – 264 V	16 A	1
861-C	90 V – 132 V	24 A	1

5.9.2 H7130 Power Supply and 5413261 Power Distribution Module

The H7130 is a multiple output power supply (+5 V, +5 VA, +12 V, -5 V, -15 V) which is used to power the CPU and MOS memory (KS10PA). The power supply is an off-line switching regulator that provides regulated ac to dc outputs under normal operating conditions. Input power is single phase line power.

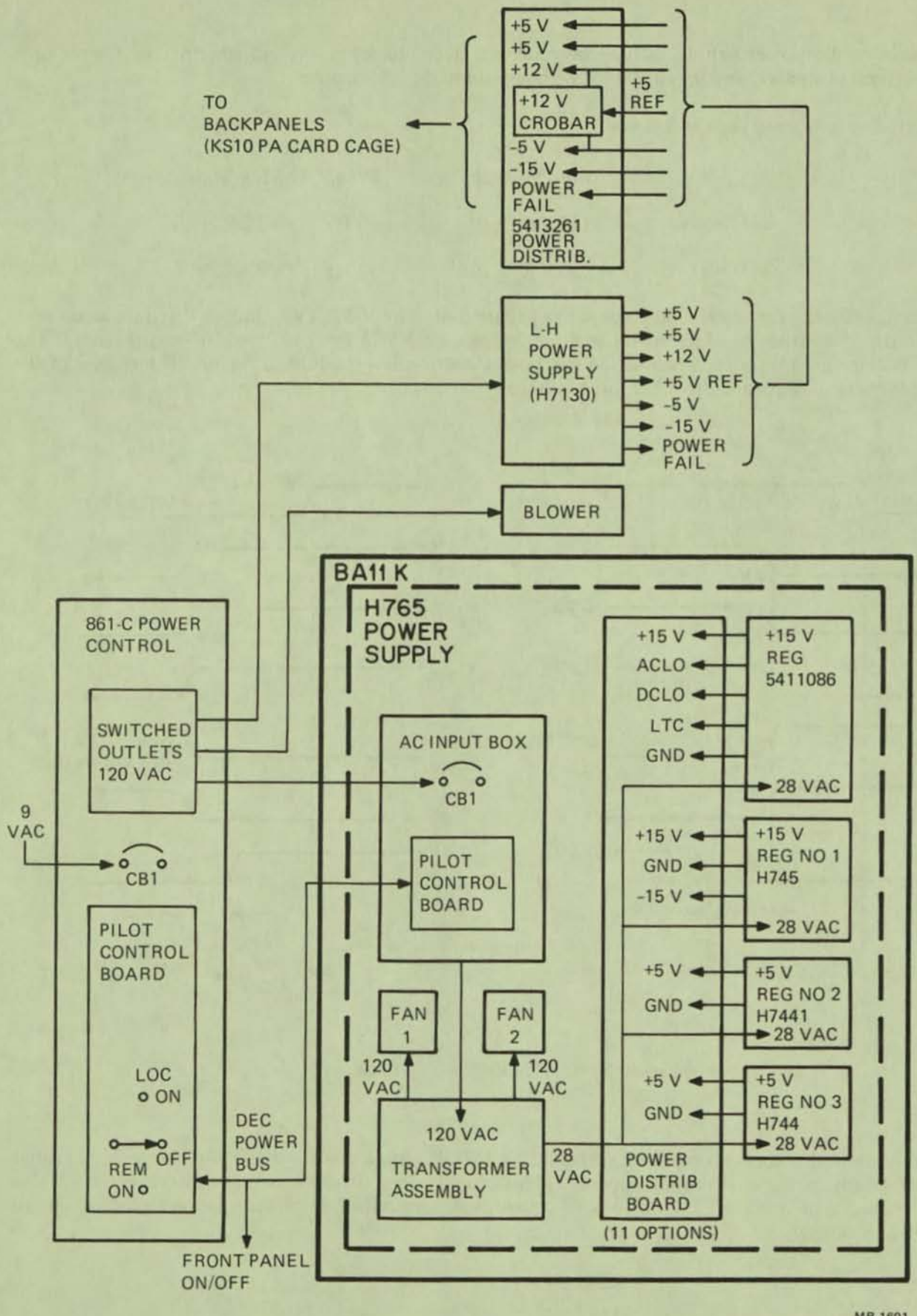


Figure 5-45 KS10 Power System

Features of the power supply include overcurrent, overvoltage, power-fail, thermal shutdown protection, and power sequencing of +12 V with respect to the –5 V output.

Electrical specifications are as follows.

	Line Voltage	Freq.	Max. Run Current
H7130C	115 Vac ±10%	60 Hz	3.75 A
H7130D	230 Vac ±10%	50 Hz	1.87 A

Power sequencing for the H7130 is shown in Figure 5-46. The +5 V (V2) and –5 V sequence up first. When the –5 V drops to –4 V, the other three voltages (+5 V (V1), +12 V, –15 V) sequence up. The –5 V is connected through a resistor on the power distribution module to the on-off terminal of the power supply to sequence the other three voltages.

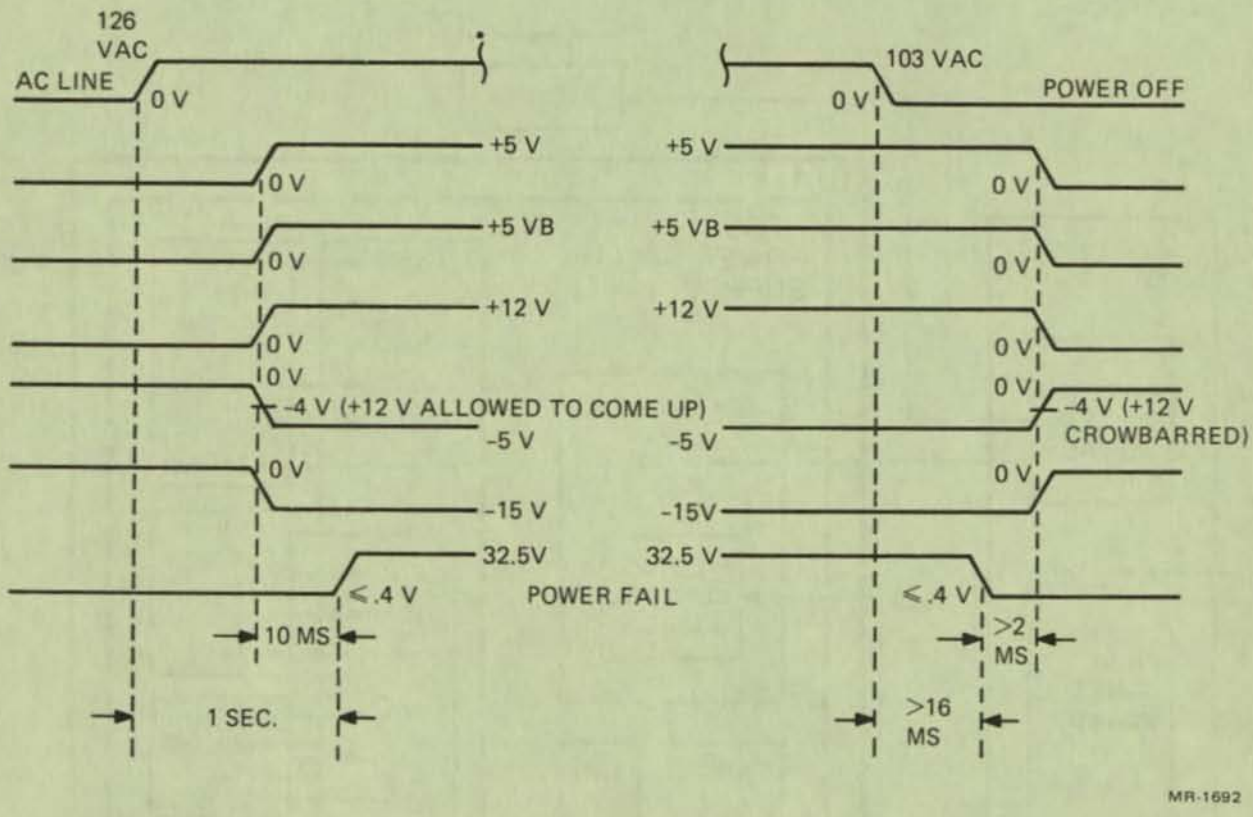


Figure 5-46 H7130 Power Sequencing

In addition to the normal operating voltages, the H7130 supplies +5 V REF to power a 12 V crobar circuit on the power distribution module. (This circuit protects MOS memory in the event –5 V is lost.) Absence of +5 V REF (and –5 V of course) will turn off all output regulators and short circuit the +12 V output.

As shown in Figure 5-45, the H7130 output voltages connect to the 5413261 power distribution module. This module does the following.

1. Provides terminal blocks for interconnecting the power supply harness to the backplane harness.

2. Provides connectors to interconnect the front panel with the backplane.

3. Interconnects the thermo-sensors to the power control.

4. Has light-emitting diodes (LEDs) which indicate if the voltages are present. The voltages indicated are +5, +5 VA, +12 V, -15 V, -5 V. The LEDs do not indicate if the voltages are within specifications.

5. Contains the 12 V crobar circuit (mentioned above) which monitors the -5 V, turns off the regulators, and shorts the +12 V to ground if the -5 V rises above -4 V.

5.9.3 H765 Power Supply (BA11-K)

The H765 power supply consists of five standard DIGITAL regulators (2-H744s, 1-H745, 1-H754 which is not used, and 1-5411086), a power control box (7009811), a power transformer, a power distribution board and two six-inch fans.

The H744 regulators each provide +5 V at 25 A. The H745 regulator provides -15 V at 10 A. The 5411086 regulator provides +15 V at 4 A. This board also generates the power fail signals AC LO and DC LO, and the line clock signal LTCL (not used in the 2020).

The 7009811 power control box contains a line cord circuit breaker, power relay, and relay control circuitry. Four three pin Mate-N-Lok connectors, two on the rear of the supply and two internal to the supply, allow low voltage control of the power on/off and emergency shutdown function. Two versions of the power control box are available, the 7009811-1 for 115 Vac operation, and the 7009811-2 for 230 Vac operation.

The power distribution board can provide dc power and control signals (AC LO, DC LO, and LTCL) to a maximum of five standard DIGITAL system units.

Electrical specifications for the H765 are as follows.

	Line Voltage	Freq.	Max. Run Current
H765-A	90–132 Vac	47–63 Hz	3.03 A
H765-B	180–264 Vac	47–63 Hz	1.52 A

Power sequencing for the H765 power supply is shown in Figure 5-47.

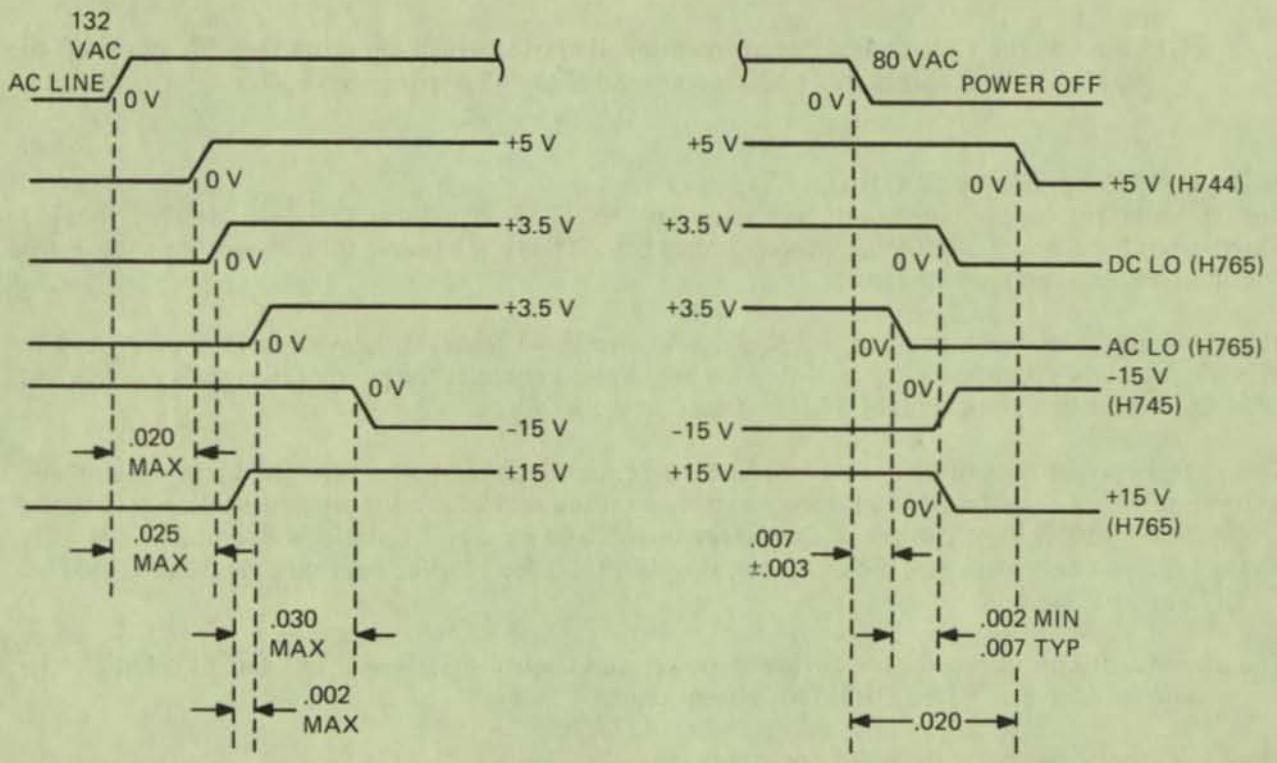
5.9.4 Blower for CPU and Memory

The blower used to provide cooling for both CPU and MOS memory operates at a voltage of 115 Vac and a current of 1.4 A.

5.9.5 Interconnection and Control

With reference to Figure 5-45, the H765, H7130, and blower are connected to the 861 power control via the switched duplex outlets. The H765 provides a minimum of 5 ms ride-through power (before indicating AC LO) during a power outage condition. The H7130 power fail signal is connected to the M8616 module and provides power sequencing of the CPU. All H765 dc outputs have a minimum hold-up time of 20 ms.

The front panel ON/OFF switch interfaces to the 861 power control via the DIGITAL power control bus. This allows all power supplies and the blower to be powered up simultaneously inside the KS10 cabinet.



NOTE:
TIMES SHOWN ARE IN SECONDS.

Figure 5-47 H765 Power Sequencing

APPENDIX A
UNIBUS

I/O devices connect to the KS10 system via a Unibus and a Unibus Adapter (UBA) as shown in Figure A-1. (The UBA, which interfaces the Unibus to the KS10 bus, is described in Paragraph 5-8.) A system may contain more than one UBA (and Unibus), thus allowing more than one string of Unibus devices to be connected. Unibus information flow is shown in Figure A-2. Unibus signals are summarized in Table A-1.

NOTE

The Unibus used on a KS10 system has the following restrictions:

1. **BR4-7 can be used only for interrupts.**
2. **An NPR device cannot do DATIP data transfers.**
3. **An NPR device is not allowed to interrupt if control of the bus was obtained by an NPR.**
4. **An NPR device cannot hog the bus for more than two memory cycles unless it is the only device connected to a UBA. (The RH11 disk controller in the standard 2020 configuration is jumpered to hog the bus for 16 memory cycles, however, it has a dedicated UBA.)**

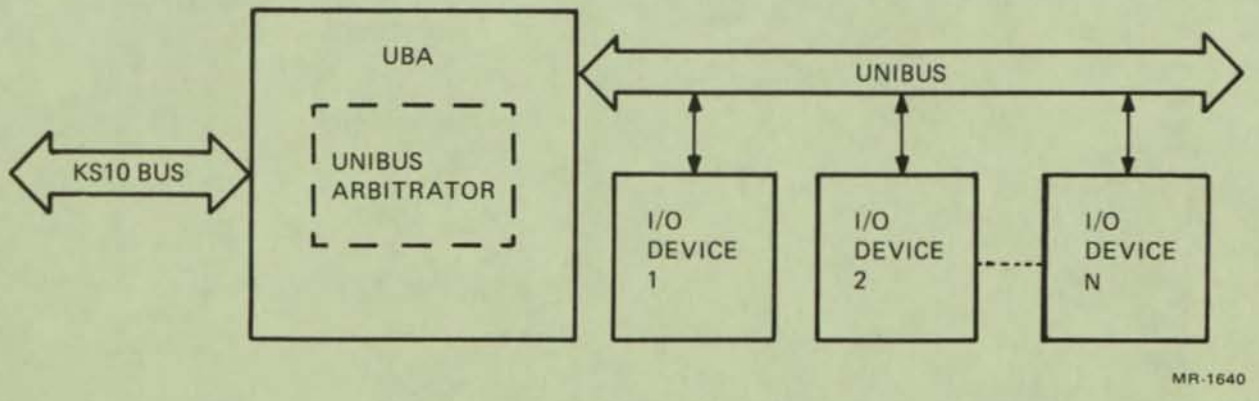


Figure A-1 KS10 Unibus Connection

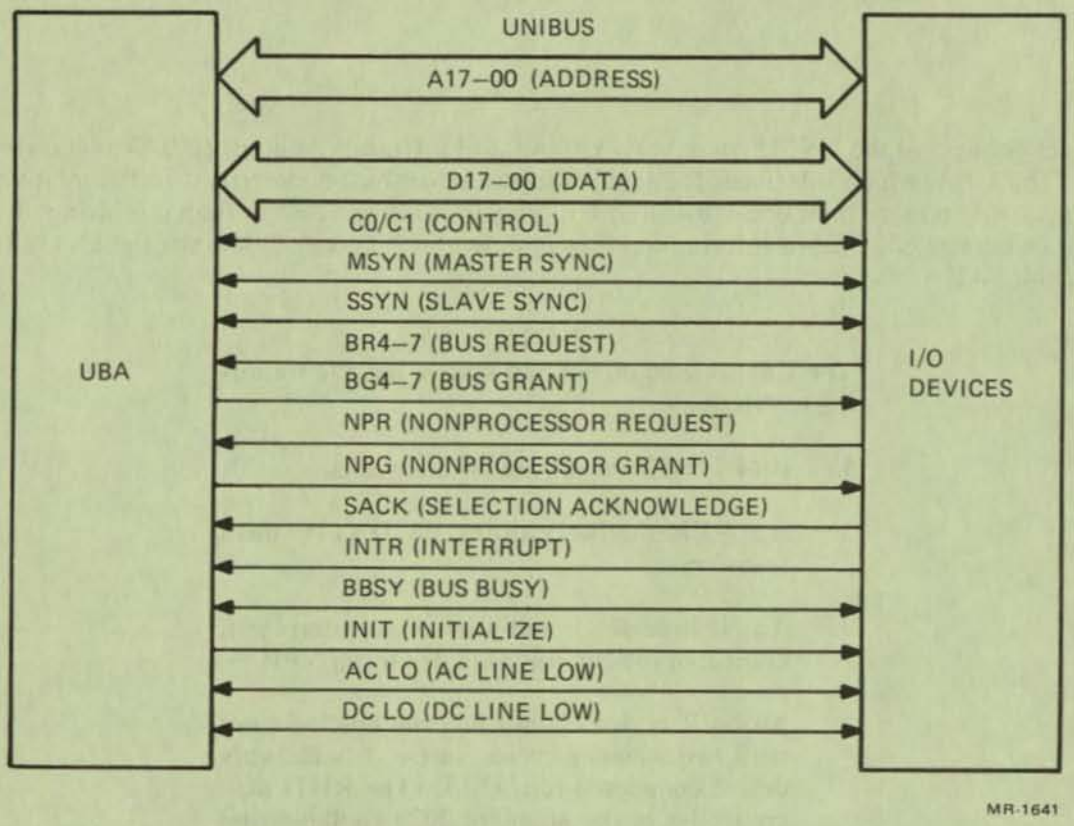


Figure A-2 Unibus Interface

Table A-1 Unibus Signal Summary

Signal(s)	Description
Address lines (A17–00)	Select slave device register (UBA master) or memory address (device master).
Data lines (D17–00)	Transfer register data (UBA master) or NPR data (device master) between master and slave.
Control lines (C0/C1)	Specify type of data transfer. **C0 C1** 0 0 DATI 0 1 DATO 1 1 DATOB
Master sync (MSYNC)	Asserted by master to initiate a data transfer.
Slave sync (SSYNC)	Asserted by slave in response to MSYNC.
Bus requests (BR4–7)	Asserted by device to request use of bus for interrupt.
Bus grants (BG4–7)	Asserted by UBA to grant use of bus for interrupt.
Nonprocessor grant (NPG)	Asserted by device to request use of bus for data transfer.
Nonprocessor grant (NPG)	Asserted by UBA to grant use of bus for data transfer.
Selection acknowledge (SACK)	Asserted by device to acknowledge bus grant (BG or NPG).
Interrupt (INTR)	Asserted by device to initiate an interrupt vector transfer.
Bus Busy (BBSY)	Asserted by master when it assumes control of the bus for data (or vector) transfer.
Initialize (INIT)	Asserted by UBA to initialize devices at power-up and in response to UBA and system reset.
AC line low	Anticipatory signal that indicates loss of ac power to Unibus device power supply (H765) or to KS10 processor power supply (H7130).
DC line low (DC LO)	Indicates loss of dc power by UBA or Unibus device power supply (H765).

The priority arbitrator for a Unibus connected to the KS10 system is located on the associated UBA. As shown in Figure A-3, it intercepts the following requests or commands.

1. NPR requests received on the Unibus
2. I/O read/write commands (to Unibus register addresses) received on the KS10 bus
3. Interrupt vector read commands received on the KS10 bus

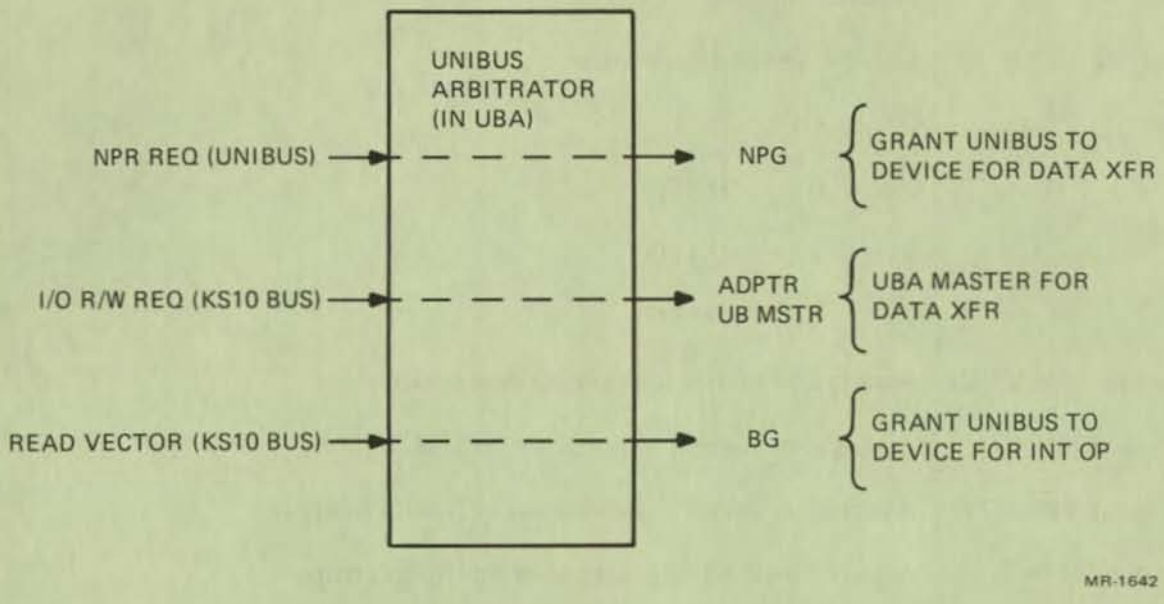


Figure A-3 Arbitrator Inputs/Outputs

In response, and *when enabled,* the arbitrator resolves request/command priority. It then performs (or allows) the Unibus priority arbitration required for the Unibus operation by asserting one of three outputs as follows:

1. NPG – Asserted and transmitted on the Unibus in response to a Unibus NPR data transfer request. NPG signals the highest priority requesting device (the one nearest the UBA) that it is the next Unibus master. The device first asserts SACK to acknowledge selection. (Bus dialogue for NPR priority arbitration is shown in Figure A-4). If and when the Unibus is inactive (BBSY $\wedge$ SSYNC = 0), the device then becomes bus master (by asserting BBSY) and performs a data transfer. (Unibus dialogue during a data transfer is shown in Figure A-5.) The arbitrator is disabled (ARB BUSY = 1) when it asserts NPG; it becomes enabled again (ARB BUSY = 0) when the device negates SACK after becoming bus master. Thus, the arbitrator is enabled to service any inputs while the NPR data transfer is in progress.

2. ADPTR UB MSTR – Asserted in response to a KS10 bus I/O command if an NPR request is not asserted (NPR request has higher priority) and if the Unibus is not already active. This arbitrator output grants the Unibus to the UBA immediately (i.e., there is no Unibus priority arbitration dialogue), and the UBA becomes bus master (asserts BBSY) and performs a data transfer operation. The arbitrator is disabled when ADPTR UB MSTR is asserted, and it is not enabled again until the end of the data transfer when SSYNC is generated by the responding device.

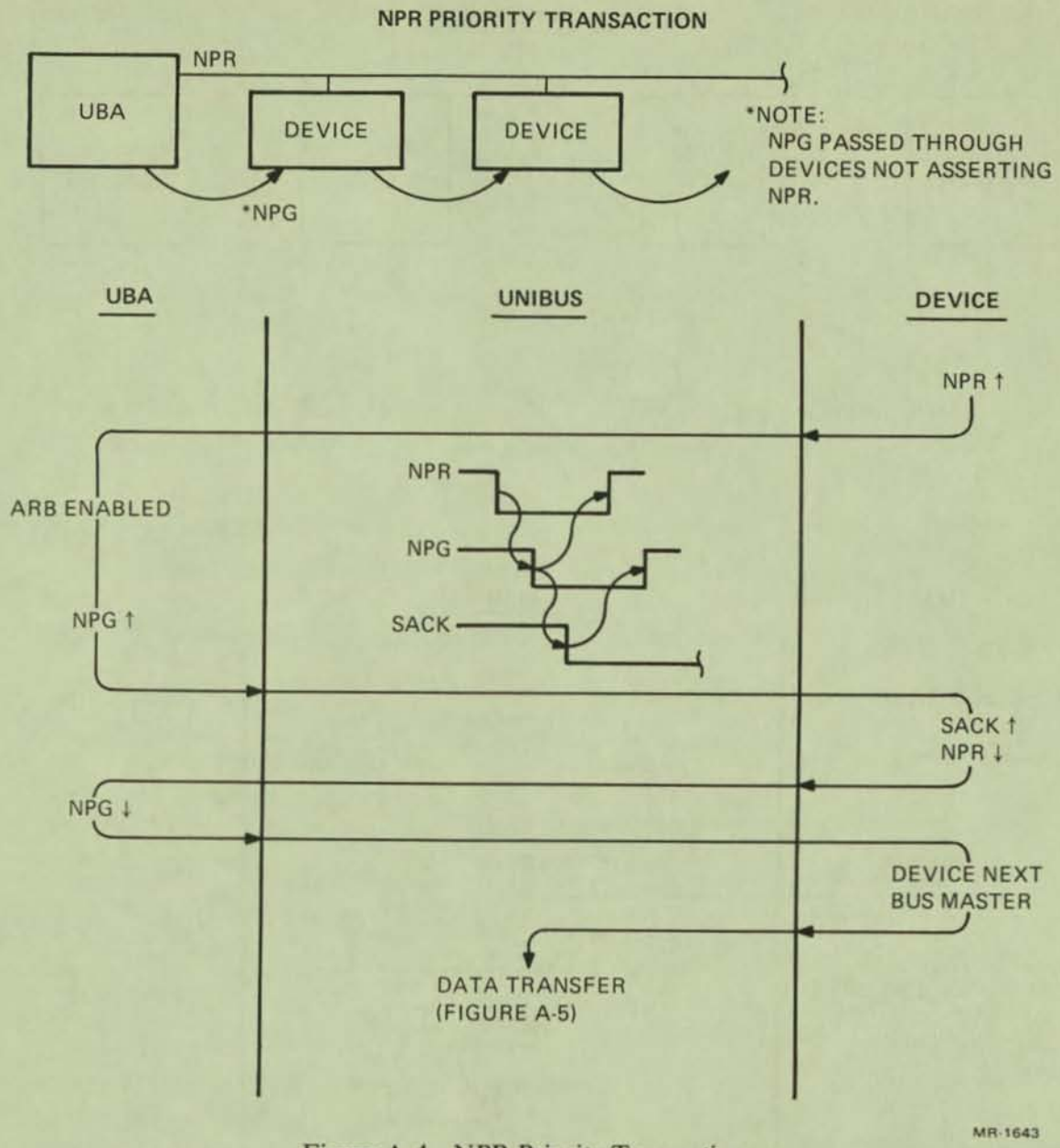


Figure A-4 NPR Priority Transaction

3. BG – Asserted in response to a KS10 bus vector read command when there is no NPR request asserted, and (if the bus is inactive) when no KS10 bus I/O command has been received. (If the bus is active, an I/O command is ignored by the arbitrator.) BG starts the Unibus priority arbitration for an interrupt operation (as shown in Figure A-6) by asserting a BG level on the Unibus corresponding to the highest priority Unibus BR level asserted by a device. (A BR level, by asserting a PI REQ on the KS10 bus, is what has caused the read vector command to be issued in the first place.) Similar to NPG, the BG level signals the highest priority requesting device (the one nearest the UBA) that it is the next bus master. The device first acknowledges selection by asserting SACK. If and when the bus is inactive, the device then becomes bus master (by asserting BBSY) and transfers the interrupt vector over the Unibus. The arbitrator is disabled when it asserts BG, and it remains disabled for the entire vector transfer; that is, until SSYNC is generated by the UBA at the end of the Unibus operation.

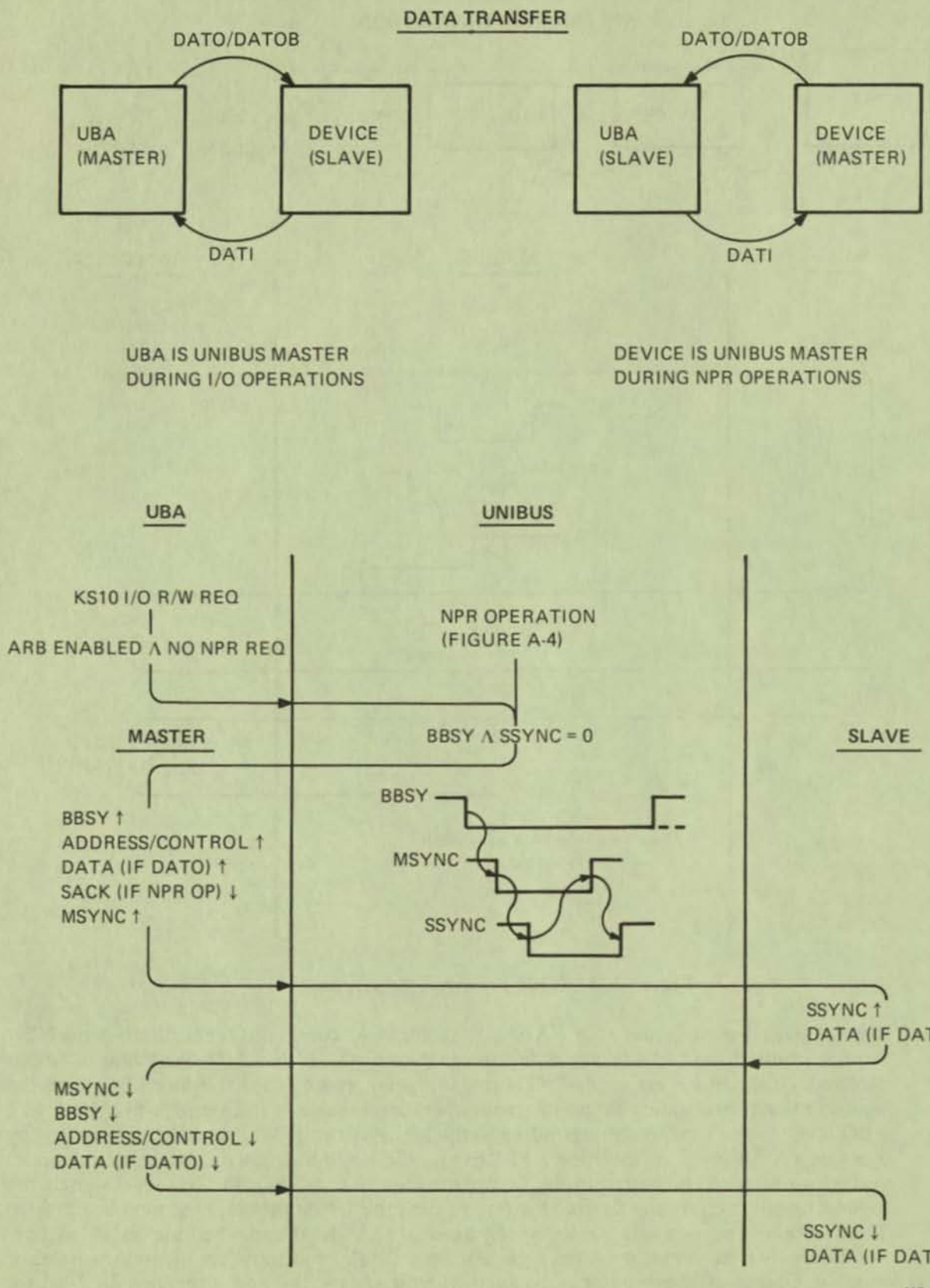


Figure A-5 Data Transfer

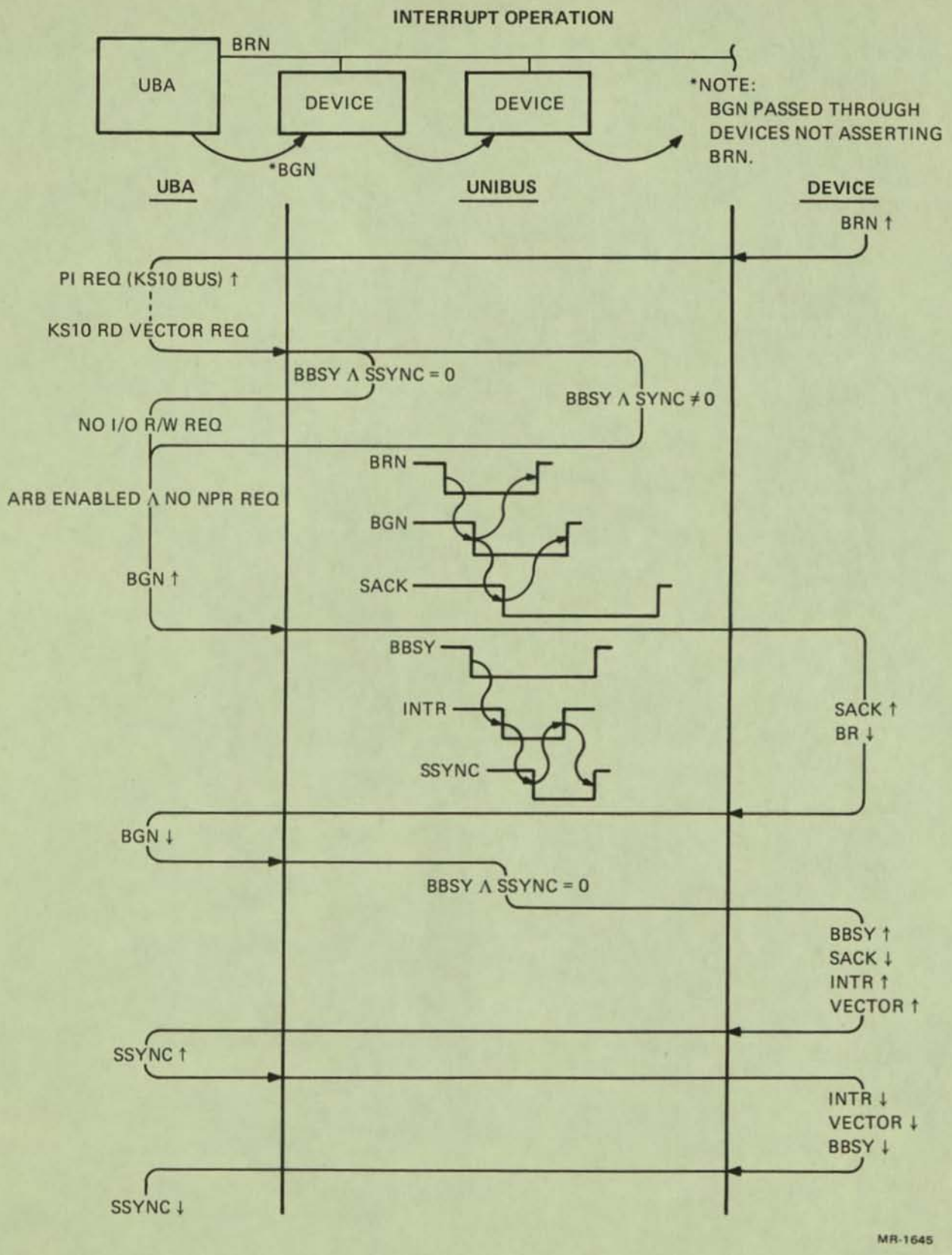


Figure A-6 Interrupt Operation

KS10-BASED DECSYSTEM-2020
TECHNICAL MANUAL
EK-0KS10-TM-001

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

_____

_____

_____

What features are most useful? _____

_____

_____

_____

What faults or errors have you found in the manual? _____

_____

_____

_____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

_____

_____

_____

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, Ma 01532
Attention: Printing and Circulation Services (NR2/M15)
Customer Services Section

Order No. EK-0KS10-TM-001

— — — — — — — — — — — Fold Here — — — — — — — — — — —

— — — — — — — — Do Not Tear - Fold Here and Staple — — — — — — — —

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
200 Forest Street (MR1-2/T17)
Marlboro, Massachusetts 01752