EK-0KS10-TM-002

KS10-BASED DECSYSTEM-2020 TECHNICAL MANUAL

digital equipment corporation • marlboro, massachusetts

1st Edition, October 1978
2nd Edition (Revised), September 1979

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)

FIGURES (Cont)

TABLES

PREFACE

This technical manual is designed to support the maintenance and training effort for the volume phase of the DECSYSTEM-2020 project. The first release of the manual supported the field test phase of the project and was for use by Digital Field Service personnel already trained and experienced on KL10-based systems. This revision of the manual contains additional overview material for use in training new-hires and Digital Field Service personnel not experienced on 10/20 systems. Also, some sections in Chapter 5, (the console description, for example) have been expanded at the request of Educational Services and Field Service/Product Support.

Related KS10 documents are as follows.

Title	Document No.
KS10-Based DECSYSTEM-2020 Installation Manual	EK-0KS10-IN-001
KS10 Maintenance Guide, Volume I	EK-OKS10-MG-001

CHAPTER 1
OVERVIEW

1.1 INTRODUCTION

The KS10 mainframe is the hardware base for the DECSYSTEM-2020, the current low-end member of the DECsystem-10 and DECSYSTEM-20 families of 36-bit computer systems. It contains a microprogrammed central processor unit (CPU) that executes the DEC 10/20 instruction set and supports both the TOPS-10 and TOPS-20 operating systems. It also contains a metal oxide semiconductor (MOS) memory with up to 512K 36-bit word capacity, an 8080A microprocessor console, and various peripheral device controllers depending on the system configuration. (Figure 1-1 shows a typical KS10 system configuration.) Peripherals connecting to the KS10 are selected Unibus and Massbus devices. System minimum/maximum configurations are as follows.

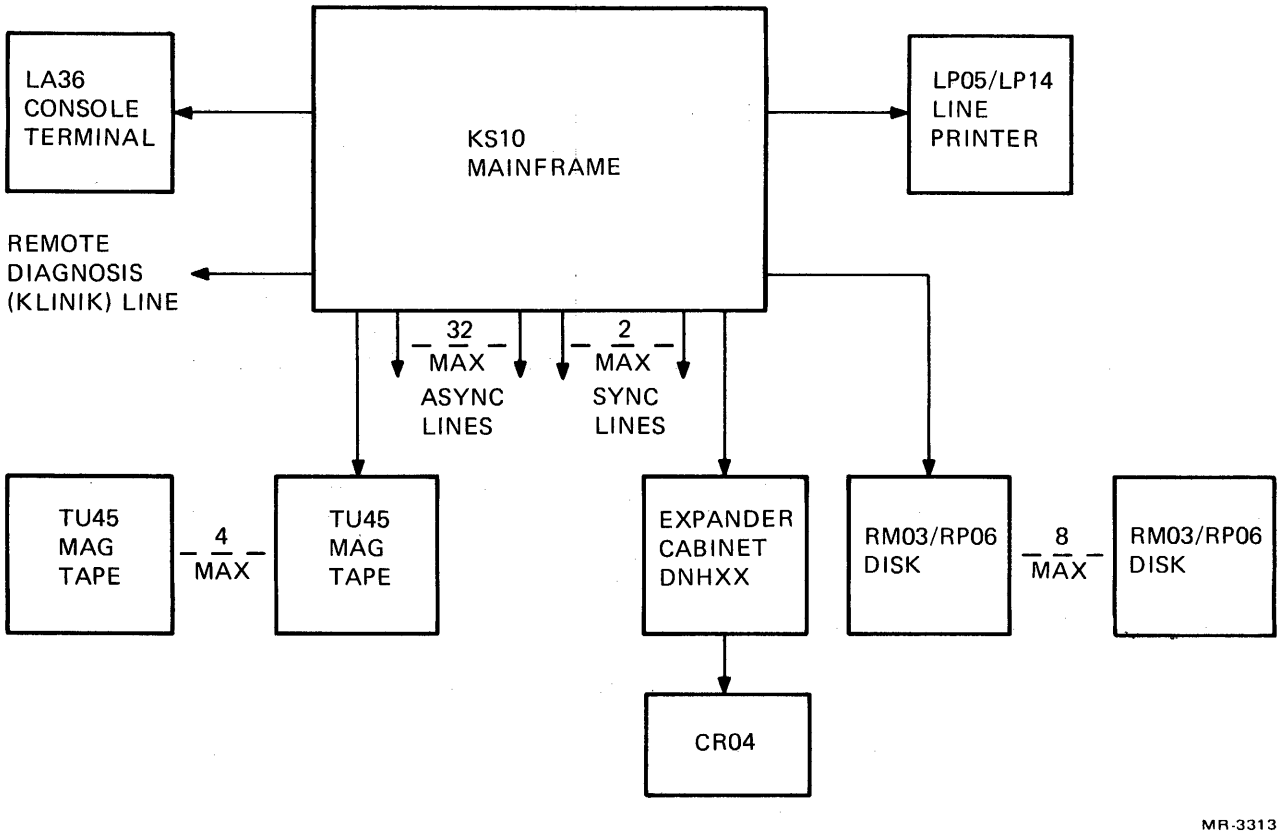


Figure 1-1 Basic KS10 System Configuration

Unit(s)	Minimum System	Maximum System
KS10	1	1
Memory (internal to KS10)	128K	512K
RM03/RP06 Disk Drive	1	8
TU45 Magnetic Tape Drive	0 (See note)	4
LP05/LP14 Line Printer	0	1
CR04 Card Reader (Requires DNHXX Expander Cabinet)	0	1
Asynchronous Lines	8	32
Synchronous Lines	0	2

NOTE
Systems serviced by DIGITAL require at least one tape drive to ensure two independent load paths for diagnostics and other system software.

1.2 SYSTEM HARDWARE

Figure 1-2 shows in detail the typical KS10 system configuration. The diagram indicates quantities and types of system buses and modules for the various mainframe components. The numbers in parentheses are module slot numbers.

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 (I/O) devices. 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 CPU consists of four extended-hex modules: two data path modules (DPE and DPM), and two control-store modules (CRA and CRM). The CPU uses low-power Schottky TTL and the AM2901 4-bit data path slice. Other features include the following.

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

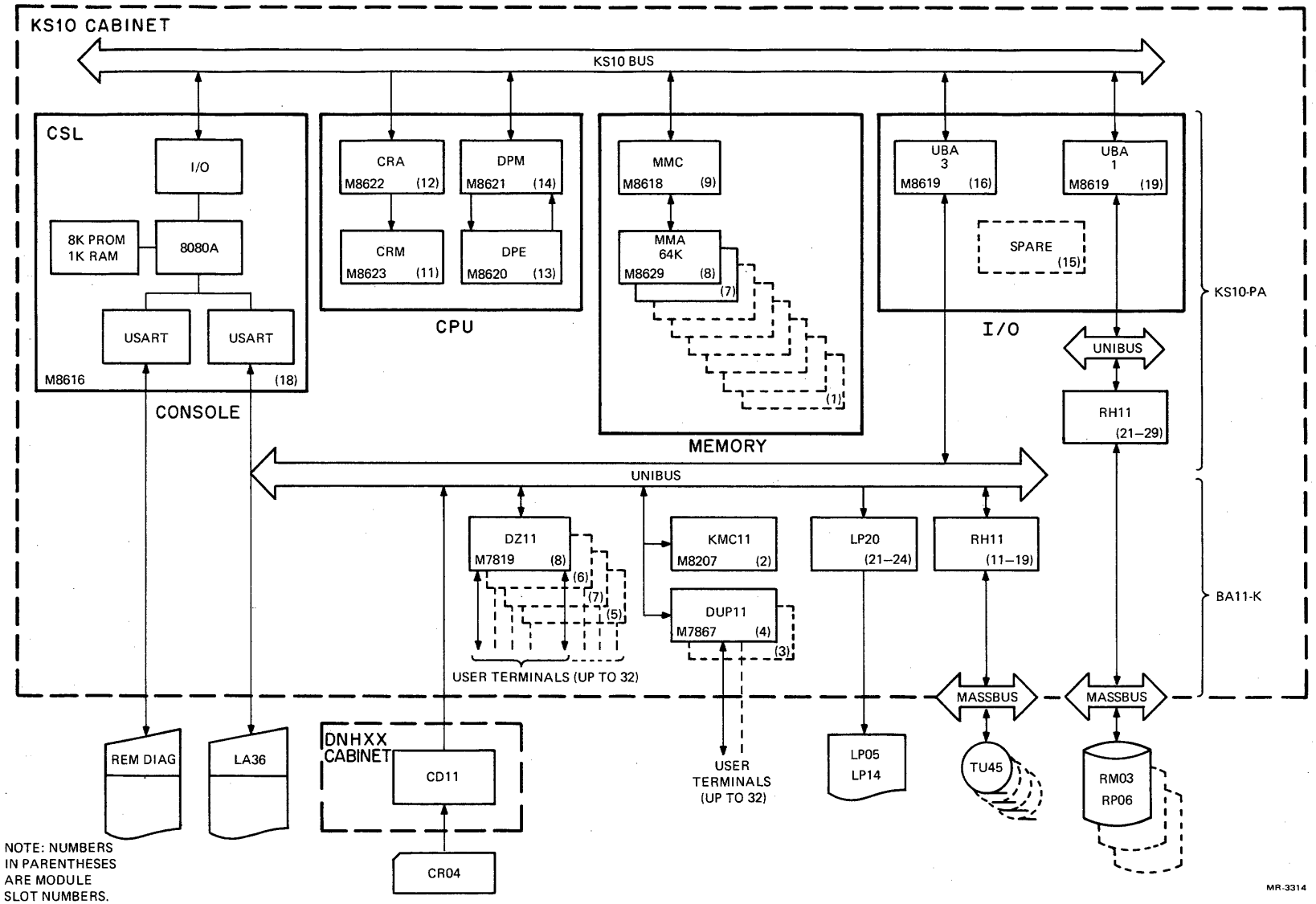


Figure 1-2 Detailed KS10 System Configuration,

The KS10 memory system consists of a single extended-hex control module (MMC) that connects to the backplane bus and from two to eight extended-hex storage array modules (MMAs). Each storage module contains 64K of MOS memory. Memory features include the following.

- 0.9 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 (CSL) that uses an 8080A microprocessor to perform console and diagnostic functions. Two USART interfaces are provided: one for console (CTY) operation and one for remote (KLINIK) line operation. The KLINIK connection operates in parallel with the CTY to allow diagnosis of the system via a remote link. The console module also contains the system clock and the arbitrator for the KS10 bus.

The KS10 I/O devices interface to the system through Unibus adapters (UBAs). Each adapter 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 typical KS10 end-user 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 drives, line printer, card reader, and synchronous and asynchronous communications lines.

Characteristics and features of the I/O devices supported on the KS10 are as follows.

Disk Drives

RP06 (RH11 controller)

- Average access time of 36.3 ms
- Average seek time of 28 ms
- Formatted capacity of 176 megabytes
- Maximum data transfer rate of 166K 36-bit words per second
- Sector size of 128 36-bit words
- Removable (20-surface) disk pack

RM03 (RH11 controller)

- Average access time of 38.3 ms
- Average seek time of 30 ms
- Formatted capacity of 67 megabytes
- Maximum data transfer rate of 250K 36-bit words per second

- Sector size of 128 36-bit words
- Removable (5-surface) disk pack

Magnetic Tape Drive

TU45 (RH11 controller)

- Tape speed of 1.9 meters (75 inches) per second
- Recording density of 32/64 characters per millimeter (800/1600 characters per inch), 9-track format on industry-standard 1/2-inch magnetic tape
- Maximum data transfer rate of 120K characters (bytes) per second

Line Printers

LP05 (LP20 controller)

- 132 columns
- EDP font, 96 or 64 characters depending on model number (model V or W)
- 230 lines per minute with 96 characters
- 300 lines per minute with 64 characters

LP14 (LP20 controller)

- 132 columns
- EDP font, 96 or 64 characters - operator-selectable
- 650 lines per minute with 96 characters
- 890 lines per minute with 64 characters

Card Reader

CR04 (CD11 controller in DNHXX expander cabinet)

- 285 cards per minute, card hopper capacity of 550 cards (models C, D).
- 1200 cards per minute, card hopper capacity of 2200 cards (models K, L)

Synchronous Communications Interface

- DUP11 controllers (1 per line)
- KMC11 NPR microprocessor (1 per system)
- Bit rate of 2000-19,200 bits per second

- DDCMP data protocol
- Two lines per system maximum

Asynchronous Communications Interface

- DZ11 controllers (one per eight lines)
- RS-232-C interface standard with baud rates of 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800
- 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
- 64 character silo receive buffer (alarm at 16 characters)

1.3 KS10 PHYSICAL DESCRIPTION

The KS10 is compactly configured in a single-width corporate hiboy cabinet (H9502H-7). This cabinet, shown in Figure 1-3, houses the KS10-PA card cage, BA11-K Unibus option drawer, power system, Massbus transition plate, asynchronous terminal distribution panel, and operator's switch panel.

1.3.1 KS10-PA Card Cage

The KS10-PA 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 1-4. This assembly contains the KS10 CPU, the MOS memory (128K words minimum, 512K words maximum), two Unibus adapters (UBAs), and the RH11 Unibus disk controller. Module utilization is shown in Figure 1-5.

1.3.2 BA11-K Unibus Option Drawer

The BA11-K Unibus option drawer (Figure 1-4) 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. RH11 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 1-6.

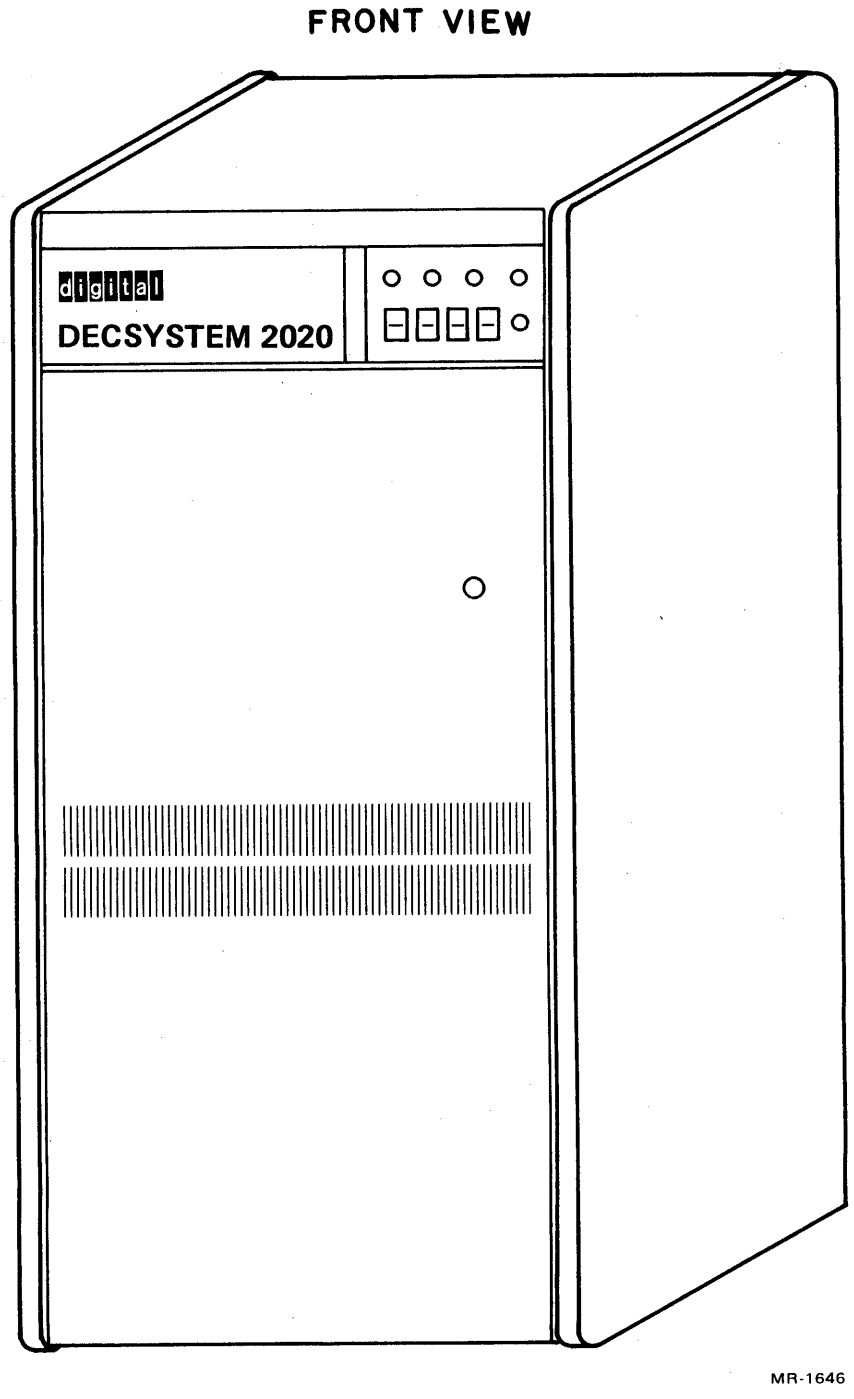


Figure 1-3 KS10 Cabinet

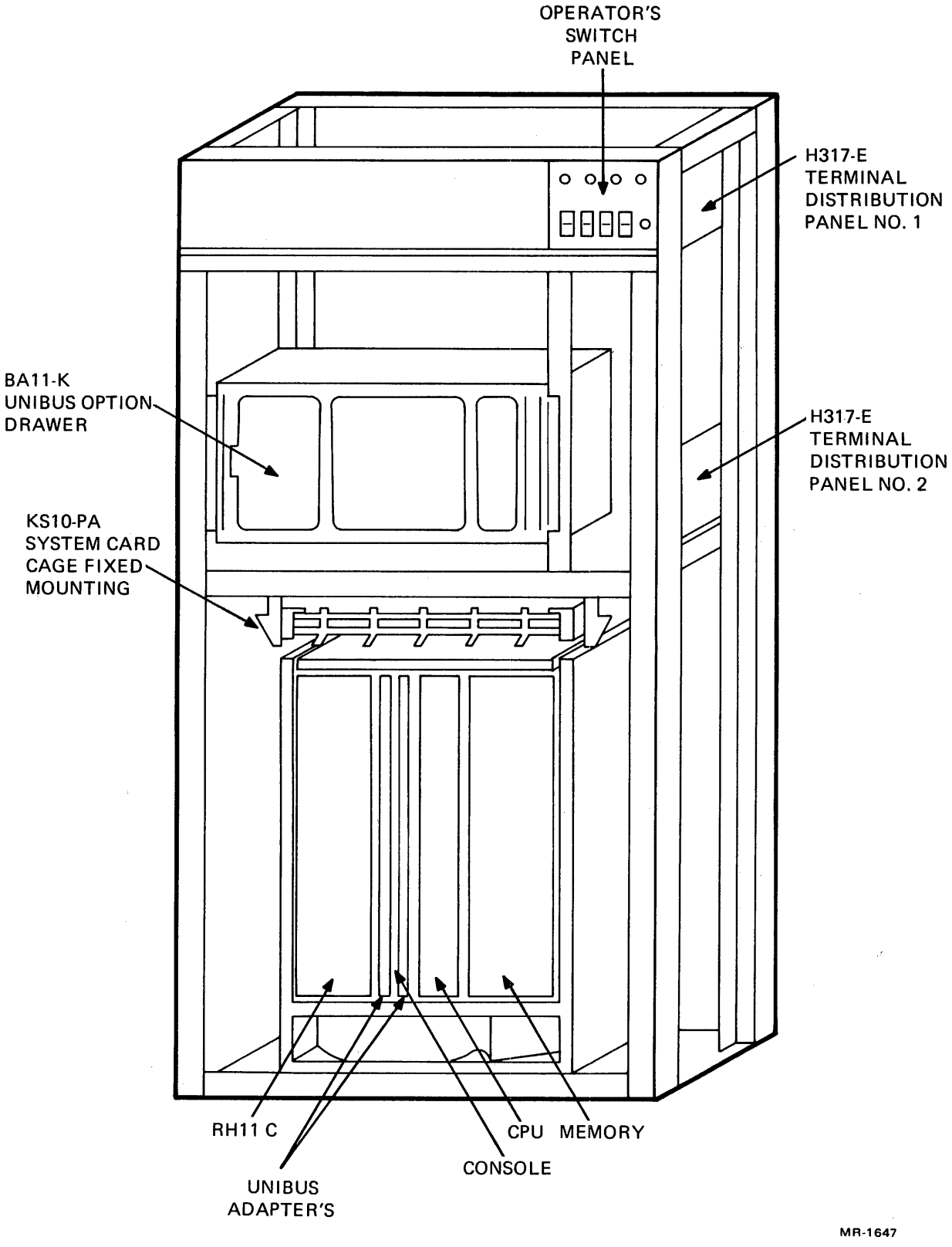


Figure 1-4 KS10 Cabinet (Front View – Skins Removed)

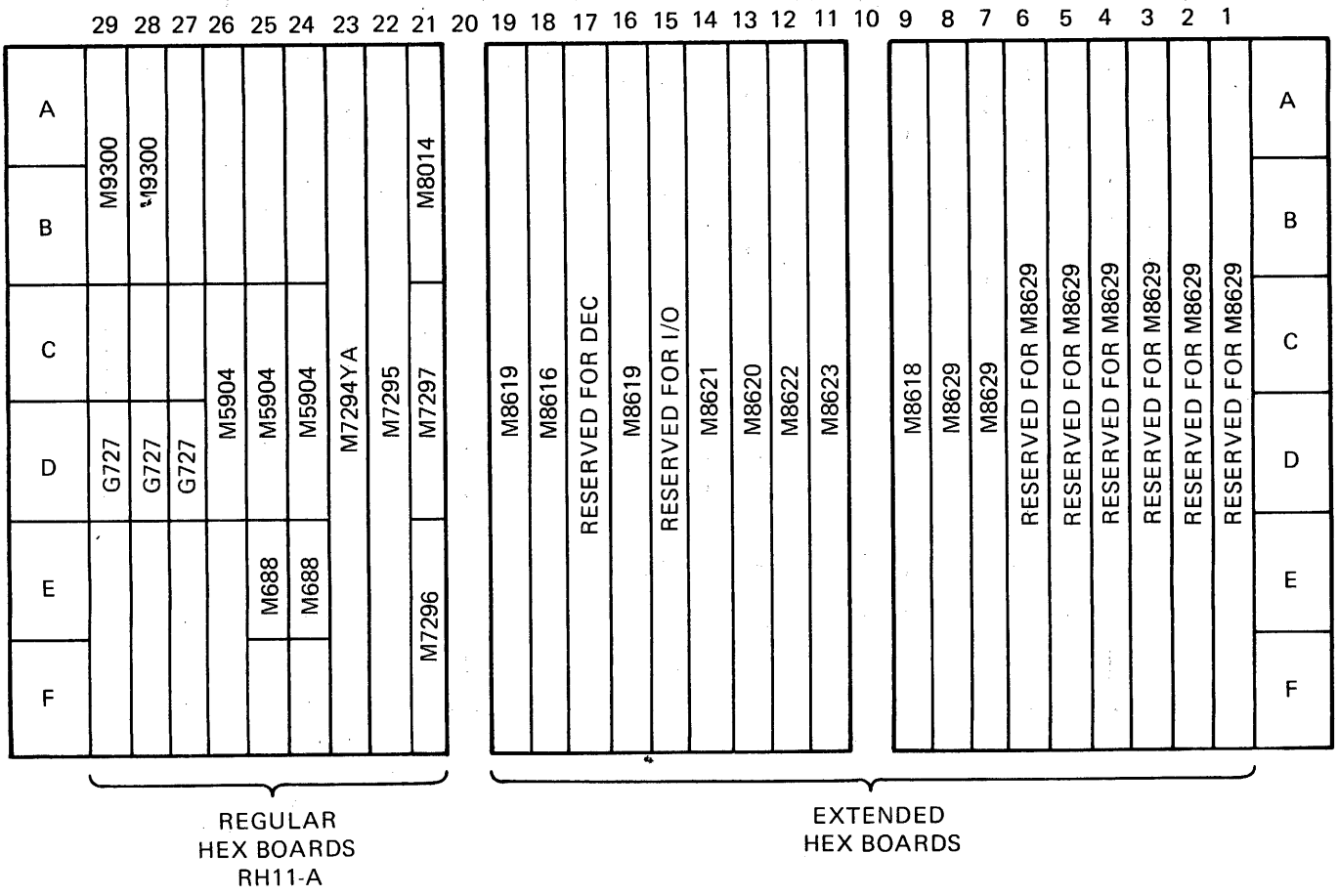
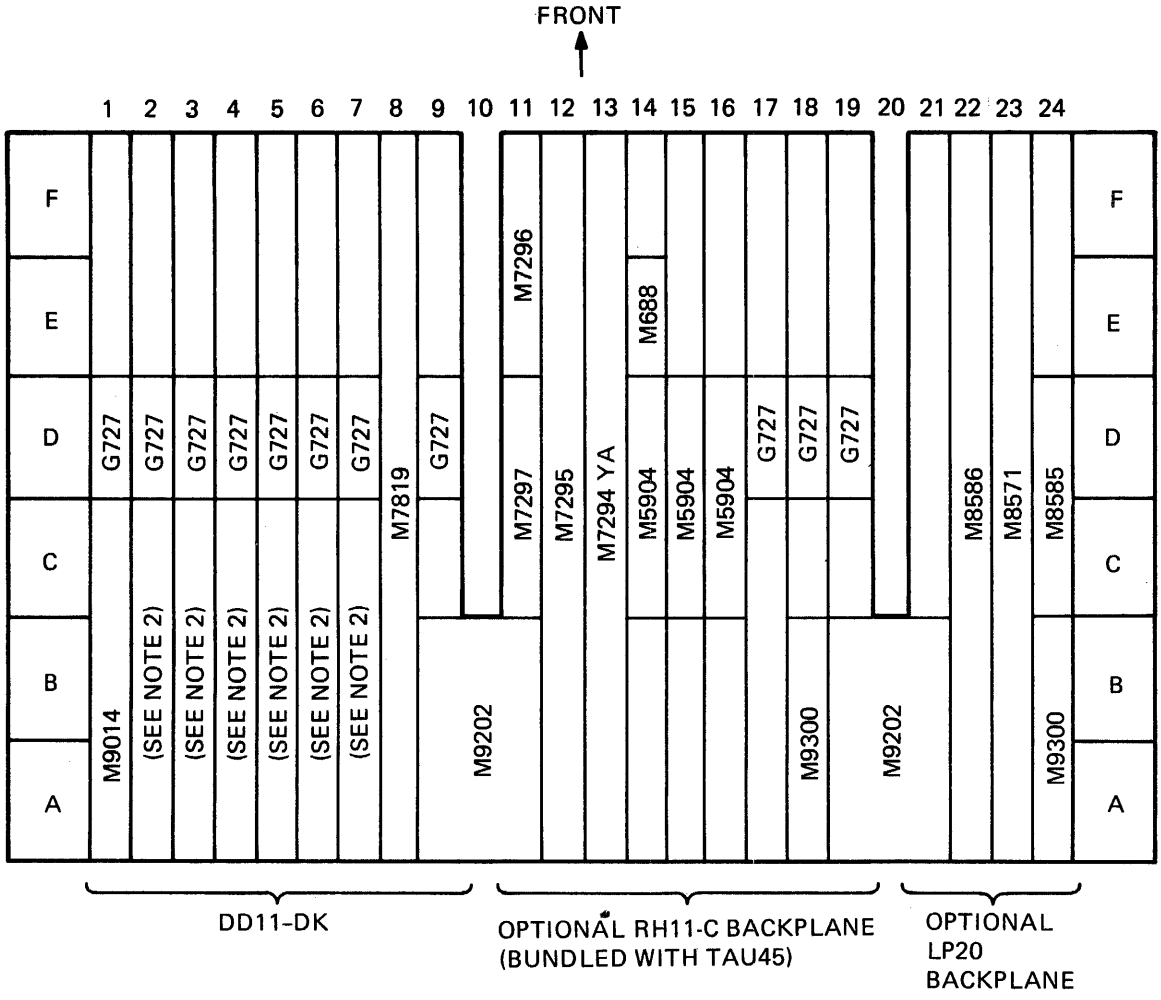


Figure 1-5 KS10-PA Card Cage Module Utilization List (MUL)



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				M8204 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-3315

Figure 1-6 BA11-K Drawer (KS10 Cabinet) MUL

1.3.3 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 KS10-PA, and the H765 switcher power supply for powering the BA11-K. Component designations for 60 Hz and 50 Hz machines are as follows.

KS10-AA (115 V, 60 Hz)	KS10-AB (230 V, 50 Hz)
861C	861B
LH Power Supply (H7130C)	LH Power Supply (H7130D)
H765A (powers BA11-K)	H765B (powers BA11-K)

NOTE TO DIGITAL
IN-HOUSE FIELD SERVICE PERSONNEL

Some in-house KS10 systems 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 other machines, there are differences in power harness wiring. When replacing a power supply, always install the same color as was removed.

1.3.4 Massbus Transition Plate

The Massbus transition plate is located at the top of the KS10 cabinet as shown in Figure 1-7. 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
2. Tape Massbus
3. Line printer Massbus

The two communication cable connectors are allocated as follows.

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

1.3.5 Asynchronous Terminal Distribution Panel (H317-E)

The KS10 is configured with a minimum of one H317-E (up to 16 lines) and a maximum of two H317-Es (32 lines). It is configured with EIA communication only.

Minimum Configuration

Lines 0–7:

One DZ11 module (8-line multiplexer)
One H317-E terminal distribution panel
One BC05W-8 cable

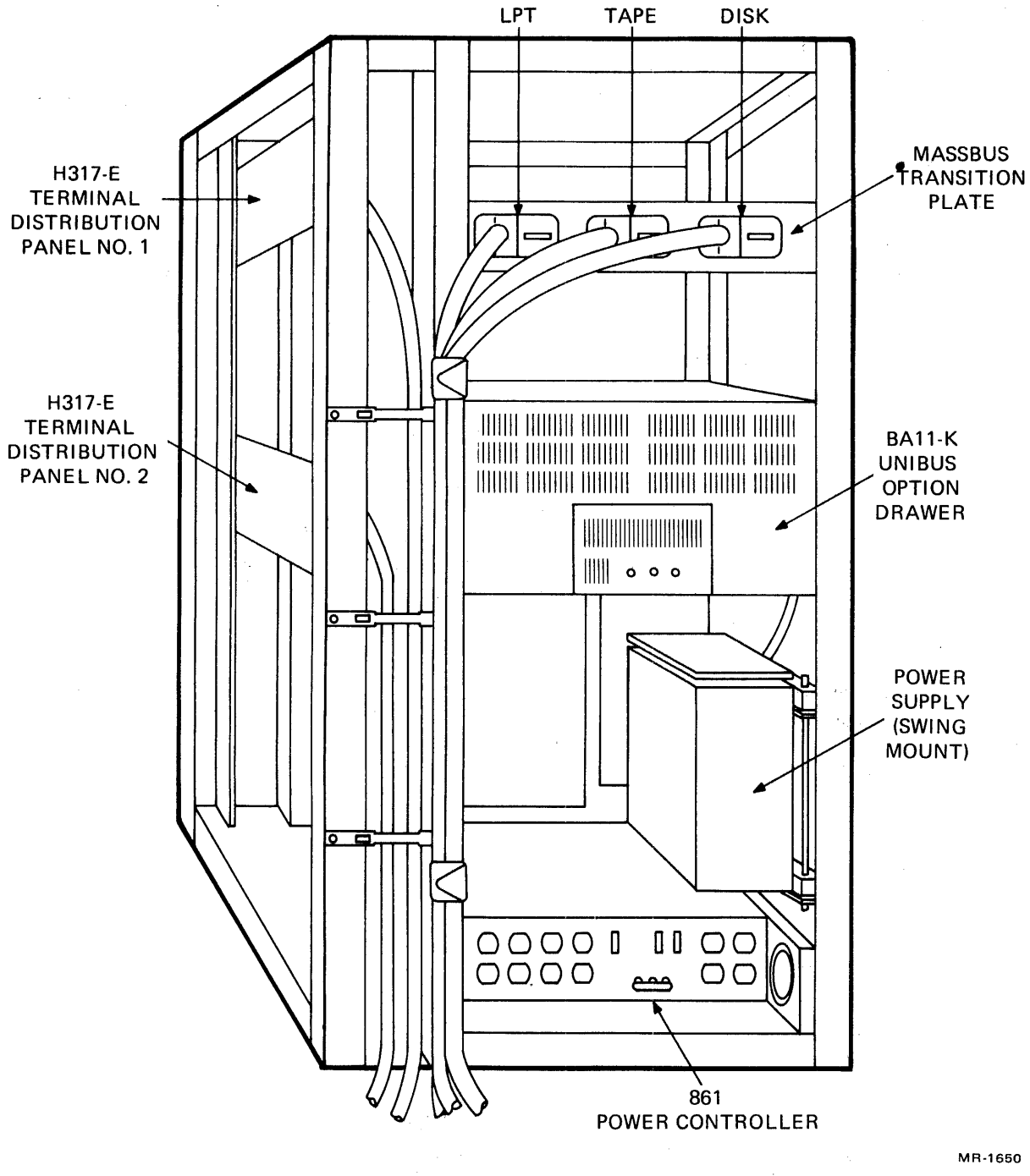


Figure 1-7 KS10 Cabinet (Rear View – Skins Removed)

Optional Expansion

Lines 8–15 (defined as a DZ11BA):

One DZ11 module (8-line multiplexer)
One BC05W-8 cable

Lines 16–23 (defined as a DZ11AA):

One DZ11 module (8-line multiplexer)
One BC05W-8 cable
One H317-E terminal distribution panel

Lines 24–32 (defined as a DZ11BA):

One DZ11 module (8-line multiplexer)
One BC05W-8 cable

1.3.6 Operator's Switch Panel

The operator's switch panel is located at the top front in the KS10 cabinet (Figure 1-3). Switch and indicator functions are described in Paragraph 4.1.

1.4 DNHXX PHYSICAL DESCRIPTION

The DNHXX expander cabinet is required to house the CD11 card reader controller when a CR04 card reader is installed on a KS10 system. Like the KS10 cabinet, it is a single-width corporate hi-boy cabinet (H9502H-7). It contains only a BA11-K drawer and an 861 power controller, however. (Component locations are shown in Figures 1-8 and 1-9.) The BA11-K holds just the CD11 on standard end-user systems. (Module utilization is shown in Figure 1-10.) Power system component designations for the 60 Hz and 50 Hz machines are as follows.

DNHXX-AA (120 V, 60 Hz)	**DNHXX-AB (230 V, 50 Hz)**
861C	861B
H765A (powers BA11-K)	H765B (powers BA11-K)

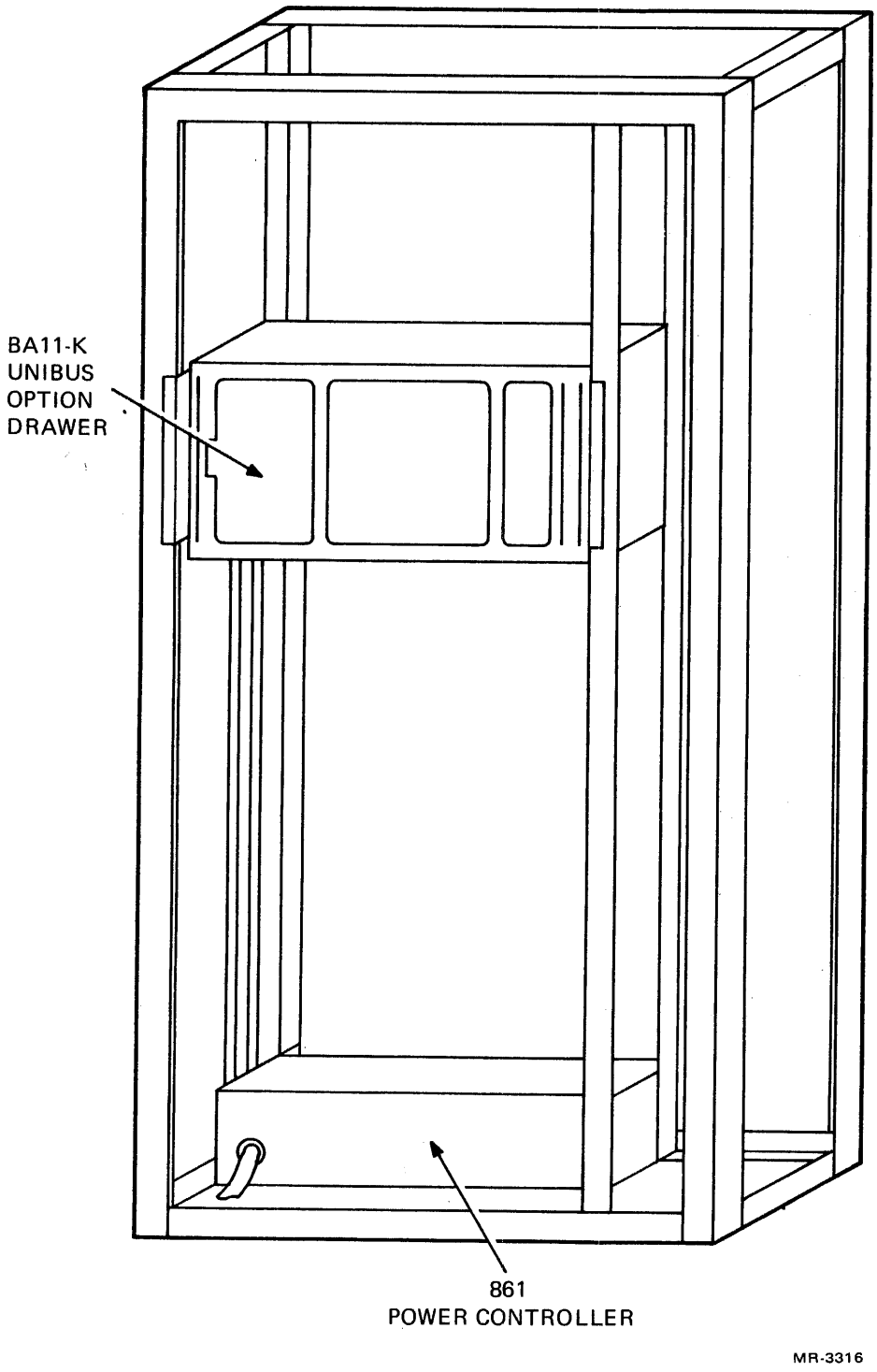


Figure 1-8 DNHXX Cabinet (Front View – Skins Removed)

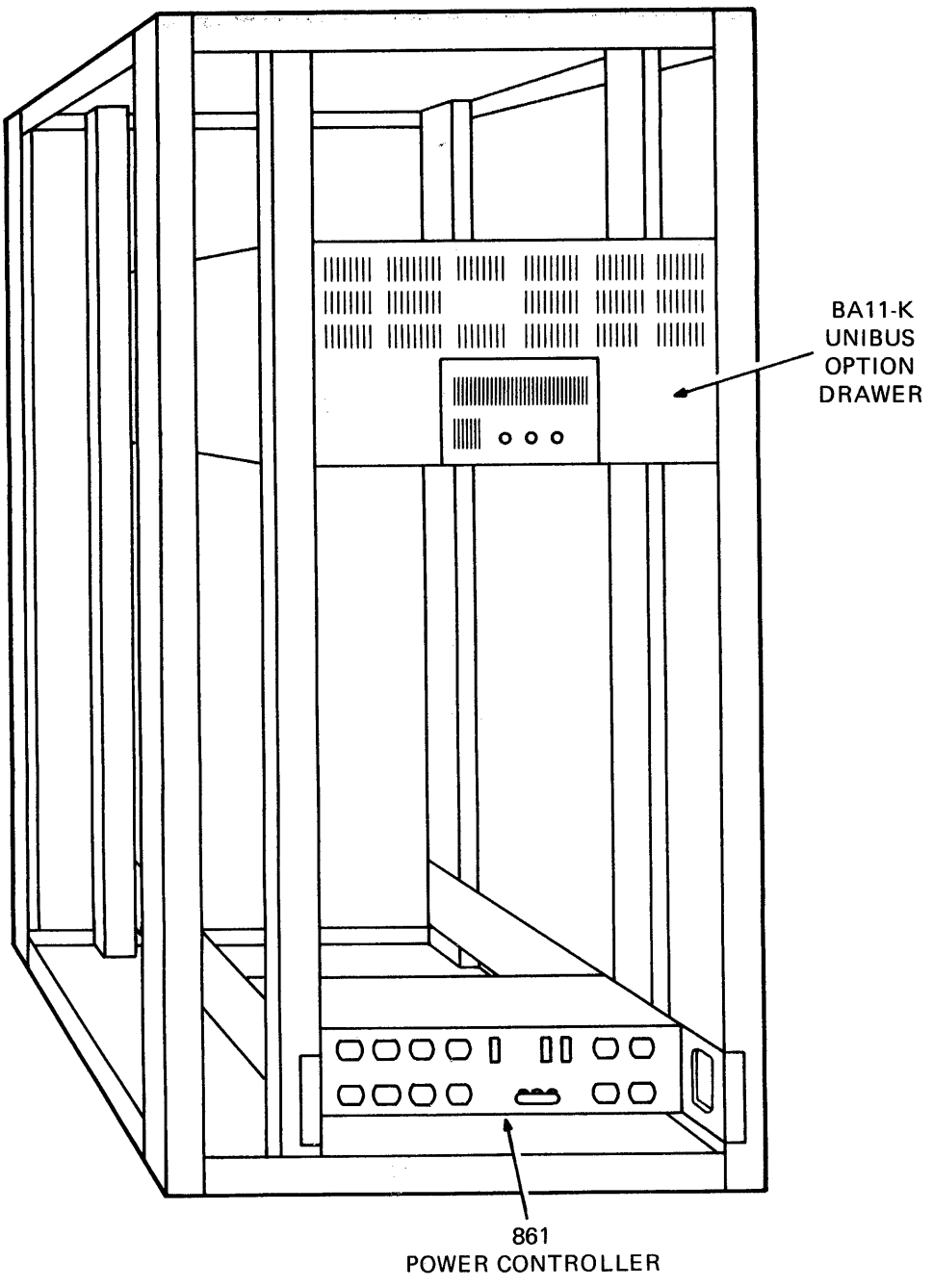
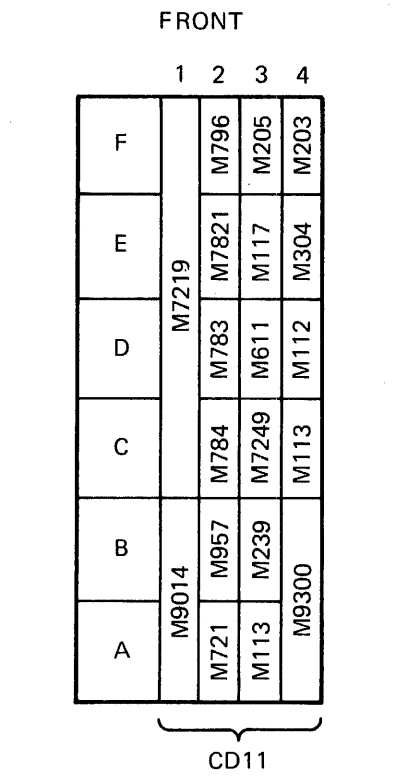


Figure 1-9 DNHXX Cabinet (Rear View – Skins Removed)



NOTE:
VIEW IS FROM MODULE SIDE.

MR-3318

Figure 1-10 BA11-K Drawer (DNHXX Cabinet) MUL

CHAPTER 2
SITE PREPARATION AND PLANNING

2.1 SITE PLANNING

Refer to Chapter 1 (Paragraphs 1.1–1.5) of the *DECSYSTEM-20 Site Preparation Guide* (EK-DEC20-SP) 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 environmental specifications for DECSYSTEM-20 systems (including KS10 systems) are listed in Table 2-1. The environmental specifications for individual KS10 system components are 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/h (3.6° F/h)
Humidity Rate of Change	2%/h
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 CABLING

Figure 2-1 shows cabling for 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 Figures 2-2 and 2-3, which show interconnections of the asynchronous and synchronous communications lines.

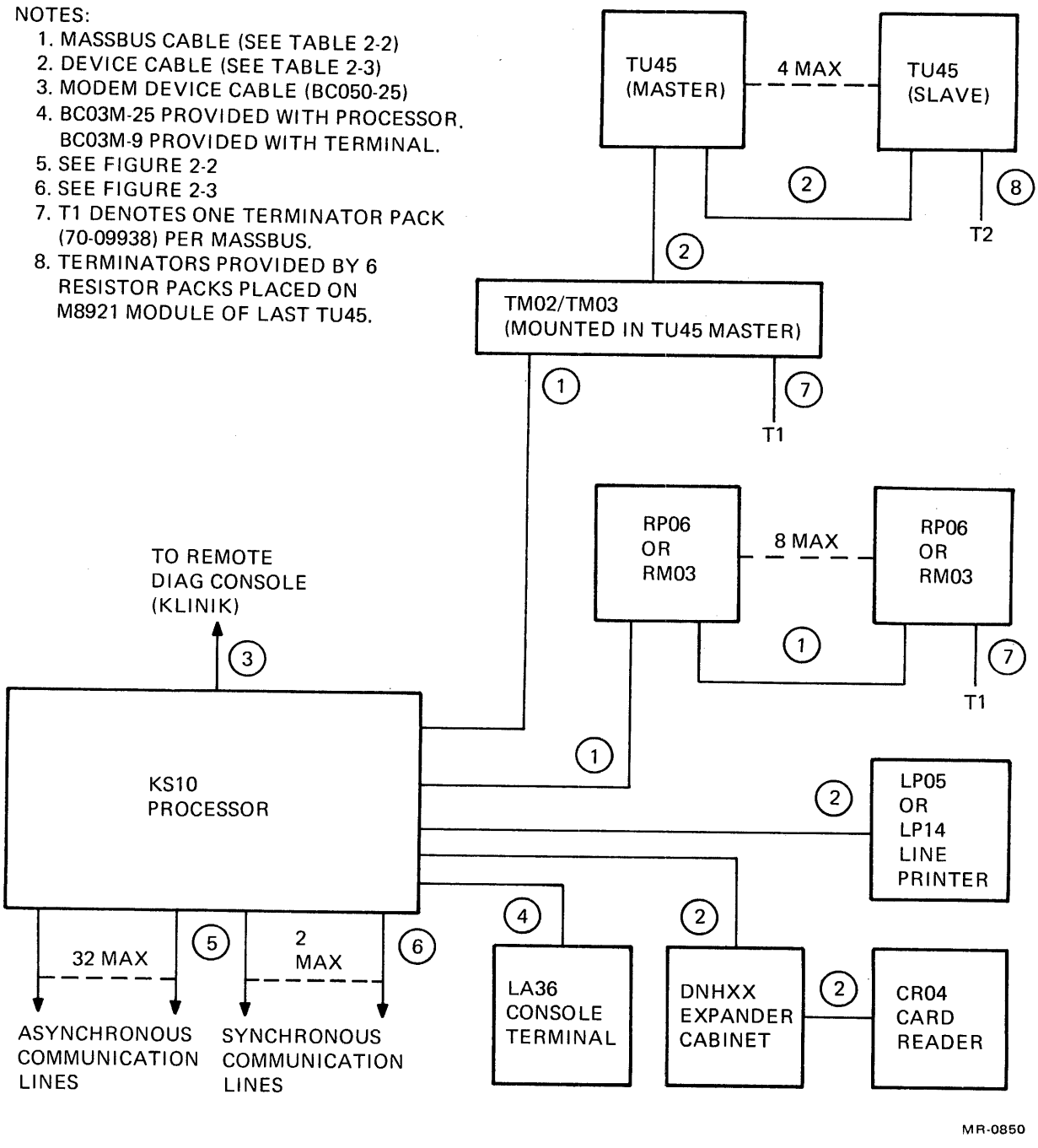


Figure 2-1 Typical KS10 System Configuration

Table 2-2 Massbus Cabling

From	To	Cable*	Available Length	
			Meters	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
			7.5	25†

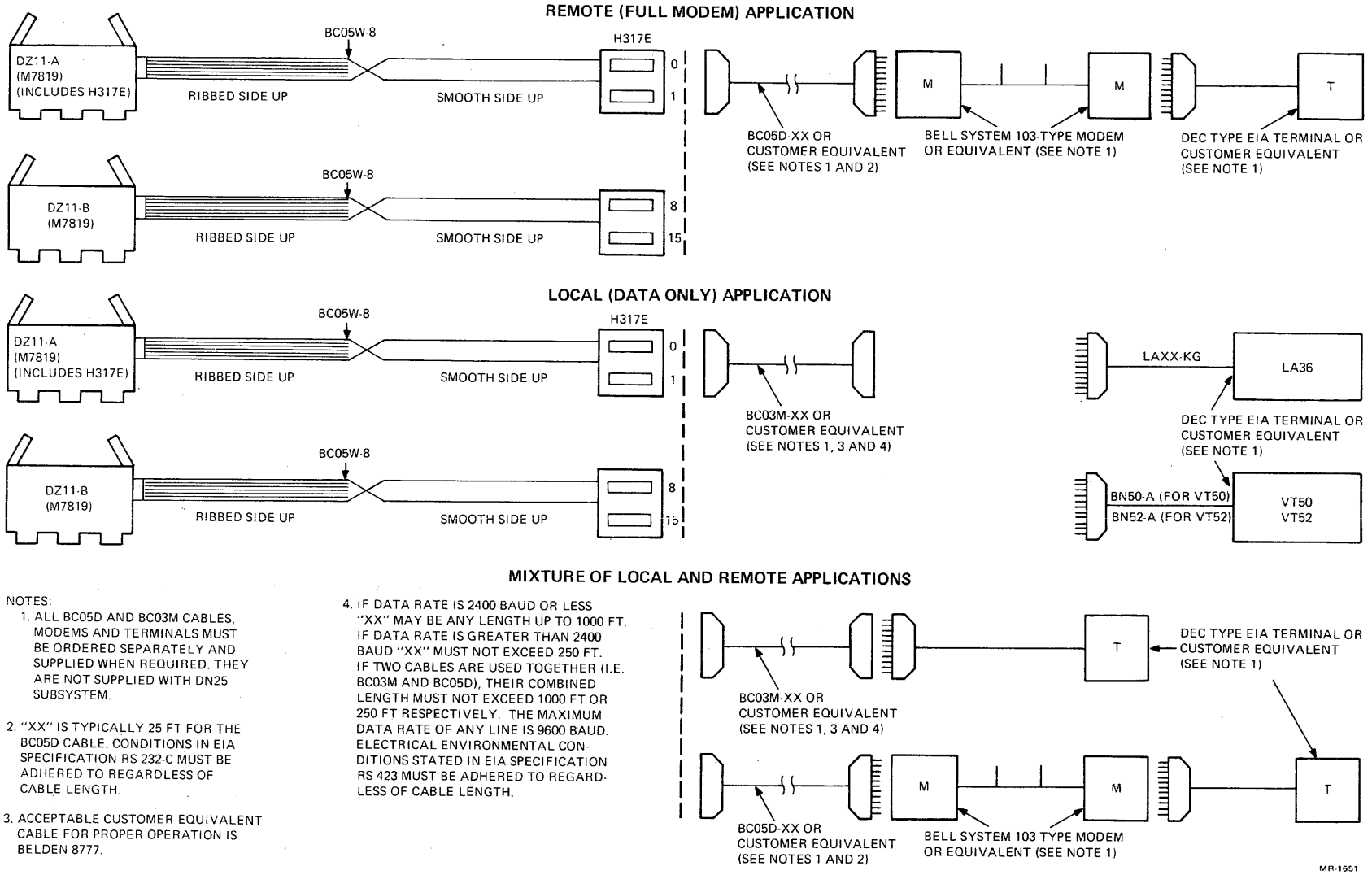
*ZIF = zero insertion force

† A 7.5 m (25 ft) BC06S cable is provided with the DNHXX to allow reconfiguration of the TU45A-E (master) drive. Reconfiguration may be necessary because the DNHXX must be installed adjacent to the KS10 cabinet.

Table 2-3 Device Cabling

From	To	Cable	Available Length	
			Meters	Feet
CPU	LP05/LP14	7011426 (AMP ZIF to Winchester)	7.5* 30	25* 100
CPU	DNHXX	BC11A (M9014 to M9014)	3.9	13
DNHXX	CR04	7008764 (AMP to M957)	7.5	25
TM02/ TM03	TU45	BC06R (BERG to BERG – cabled internally)	3	10
TU45	TU45	BC06R (BERG to BERG)	3	10

* The 7.5 m (25 ft) 7011426 cable is the standard length that will be provided if no cable information is provided 60 days prior to scheduled shipment.



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

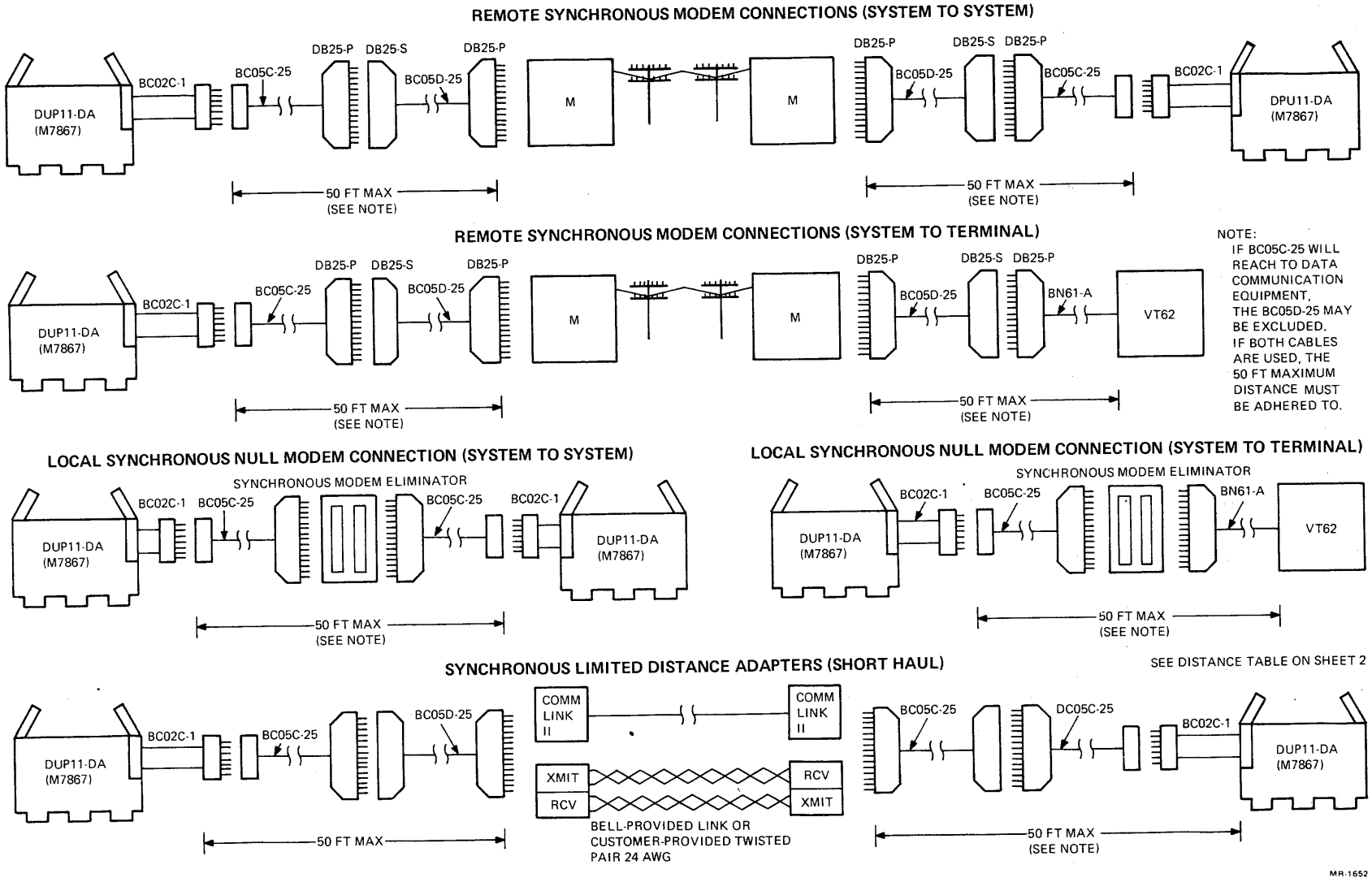


Figure 2-3 KS10 Synchronous Communications Lines

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. They are arranged in alphanumeric sequence by device designations as follows.

CR04-C/D
CR04-K/L
DNHXX-AA/AB
KS10-AA/AB Processor
LA36
LP05-V/W
LP14-C/D
RM03
RP06-A/B
TU45A-E (Master)
TU45A-E (Slave)

CR04-C/D

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
TT	27 kg 60 lb	28 cm 11 in	49 cm 19 in	35.5 cm 14 in	VE	N/A

POWER (AC)

AC Voltage			Frequency Tolerance	Phase(s)	Steady State Current (RMS)	Surge Current	Surge Duration
Low	Nom	High					
104 208	120 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	5 A 2.5 A	13 A 6.5 A	6 s 6 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
5 ms	439 kg·cal/hr 1740 Btu/hr	510	0.60	2.7 m 9.0 ft	NEMA 5-15P NEMA 6-15P	0.545 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
15° to 32° C 59° to 90° F	5° to 50° C 40° to 120° F	20 - 80%	0 - 95%	7° C/hr 12° F/hr	2%/hr	75 ft^3/min (rear)

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
15° to 32° C 59° to 90° F	4° to 49° C 40° to 120° F	20-80%	0-95%	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	7.5 m 25 ft	N/A

CR04-K/L

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	91 kg 200 lb	102 cm 40 in	64 cm 25 in	102 cm 40 in	VE	N/A

POWER (AC)

AC Voltage			Frequency Tolerance	Phase(s)	Steady State Current (RMS)	Surge Current	Surge Duration
Low	Nom	High					
104 208	120 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	9.0 A 4.5 A	25 A 12.5 A	6 s 6 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
5 ms	473 kg·cal/hr 1877 Btu/hr	550	1.1	4.5 m 15 ft	NEMA 5-15P NEMA 6-15P	0.655 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
15° to 32° C 59° to 90° F	5° to 50° C 40° to 120° F	20 - 80%	0 - 95%	7° C/hr 12° F/hr	2%/hr	150 ft^3/min (Rear)

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
15° to 32° C 59° to 90° F	4° to 49° C 40° to 120° F	20 - 80%	0 - 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	7.5 m 25 ft	N/A

DNHXX-AA/AB

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	181 kg 400 lb	152 cm 60 in	69 cm 27 in	76 cm 30 in	H9502H-7	N/A

POWER (AC)

AC Voltage			Frequency Tolerance	Phase(s)	Steady State Current (CRMS)	Surge Current	Surge Duration
Low	Nom	High					
104 208	120 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	1.3 A 0.65 A	3.3 A 1.7 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	121 kg-cal/hr 480 Btu/hr	140.5	0.156	4.5 m 15 ft	NEMA L5–30P NEMA L6–20P	1.15 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
15° to 32° C 59° to 90° C	-40° to 66° C -40° to 151°F	20 – 80%	0 – 95%	7° C/hr 12° F/hr	2%/hr	500 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

KS10-AA/AB Processor

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	120 230	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	1.18	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)

LA36

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	46 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					
90 180	120 230	132 264	60 Hz ± 1 50 Hz ± 1	1 1	2.0 A 1.0 A	30 A 15 A	1 s 1 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
	175 kg-cal/hr 696 Btu/hr	204	0.24	2.4 m 8 ft	NEMA 5-15P NEMA 6-15P	0.107 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
10° to 40° C 50° to 104° F	-40° to 66° C -40° to 151°F	10–90%	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

LP05-V/W

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	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		Steady State	Surge	Surge
Low	Nom	High	Tolerance	Phase(s)	Current (RMS)	Current	Duration
104 208	120 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	4.5 A 2.3 A	10 A 5 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	395 kg·cal/hr 1570 Btu/hr	460	0.54	2.7 m 9 ft	NEMA 5-15P NEMA 6-15P	0.55 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	20–80%	0–95%	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	20–80%	0–95%	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

LP14-C/D

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	120 230	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	670 kg-cal/hr 2662 Btu/hr	780	0.84	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	20–80%	0–95%	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	20–80%	0–95%	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

RM03

MECHANICAL

Mounting Code	Weight	Height	Width	Depth	Cab Type If Used	Skid Type
FS	195 kg 430 lb	99 cm 39 in	56 cm 22 in	79 cm 31 in	H9691 (modified)	

POWER (AC)

AC Voltage			Frequency		Steady State	Surge	Surge
Low	Nom	High	Tolerance	Phase(s)	Current (CRMS)	Current	Duration
102 213	120 230	128 257	60 Hz ± 1 50 Hz ± 1	1 1	7.0 A 3.5 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)
8 ms	614 kg-cal/hr 2436 Btu/hr	714	0.84	1.8 m 6 ft	NEMA 5–15P NEMA 6–15P	1.218 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
15° to 32° C 59° to 90° F	-40° to 70° C -40° to 158°	20 – 80%	5 – 95%	7° C/hr 12° F/hr	2%/hr	

ENVIRONMENTAL (MEDIA)

Temperature		Relative Humidity		Rate of Change	
Operating	Storage	Operating	Storage	Temp	Rel. Humid.
10° to 57° C 50° to 135° F	-40° to 65° C -40° to 150° F	8 – 80%	8 – 80%	0.1° C/min 0.2° F/min	

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	272 kg 600 lb	118 cm 46.5 in	81 cm 32 in	81 cm 32 in	VE	12-10568-02

POWER (AC)

AC Voltage			Frequency Tolerance	Phase(s)	Steady State Current (RMS)	Surge Current	Surge Duration
Low	Nom	High					
104 208	120 240	127 254	60 Hz ± 1 50 Hz ± 1	3-wye 3-wye	12.6 A (total) 6.3 A (total)	30 A 15 A	10 s 10 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
10 ms	1105 kg-cal/hr 4384 Btu/hr	1285	1.5 (total)	4.5 m 15 ft	NEMA # L21-20P	4.36 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
15° to 32° C 59° to 90° F	10° to 44° C 50° to 110° 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 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

TU45A-E (Master)

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 Tolerance	Phase(s)	Steady State Current (RMS)	Surge Current	Surge Duration
Low	Nom	High					
104 208	120 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	8.5 A 4.3 A	14 A 7 A	1 s 1 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
5 ms	757 kg-cal/hr 3003 Btu/hr	880	1.02	4.5 m 15 ft	NEMA L5-30P NEMA L6-20P	3.16 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
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	400 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)

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 Tolerance	Phase(s)	Steady State Current (RMS)	Surge Current	Surge Duration
Low	Nom	High					
104 208	120 230	127 254	60 Hz ± 1 50 Hz ± 1	1 1	6.8 A 3.4 A	14 A 7 A	1 s 1 s

POWER (AC)

Interrupt Tolerance (Max)	Heat Dissipation	Watts	KVA	PWR Cord Length	PWR Cord Conn Type	Leakage Current (Max)
5 ms	605 kg-cal/hr 2402 Btu/hr	704	0.82	4.5 m 15 ft	NEMA L5-30P NEMA L6-20P	3.16 mA

ENVIRONMENTAL (DEVICE)

Temperature		Relative Humidity		Rate of Change		Air Volume Inlet
Operating	Storage	Operating	Storage	Temp	Rel. Humid.	
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	400 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	N/A	3 m 10 ft	N/A

(3) BC06R

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.

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

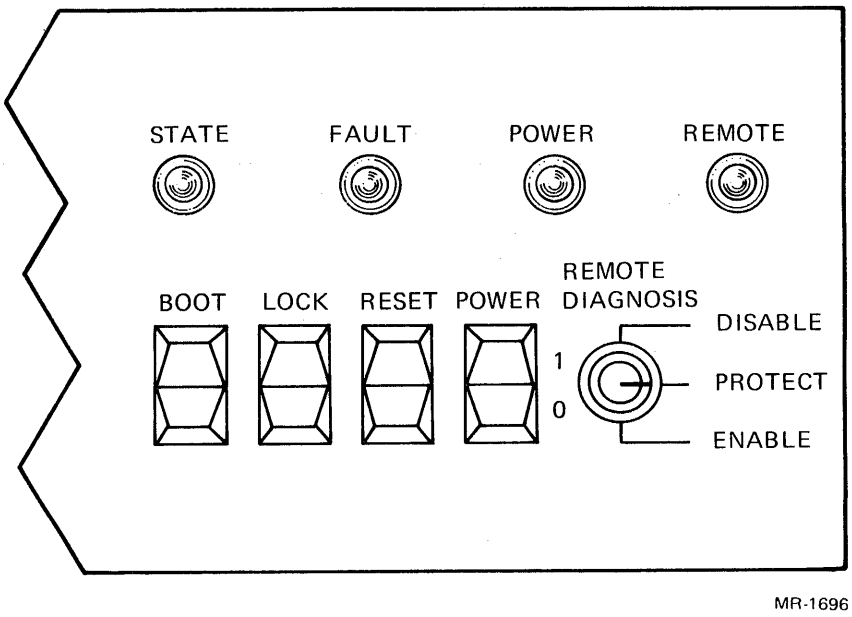


Figure 4-1 KS10 Switch and Indicator Panel

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 BA11-K power supplies.) When system is powered on, system will auto-boot unless CTY character is typed within 30 seconds.
RESET	Resets all KS10 system components (including the 8080 console hardware). System will auto-boot unless CTY character is typed within 30 seconds.
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

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 INSTRUCTION SET

The KS10 instruction set consists of 396 system-level instructions. Each instruction is microprogrammed; that is, it is actually executed by a series of microinstructions (called the microcode) in the CPU's control-store. (The microcode is loaded from disk or tape into the CPU's 2K-word control RAM during the system bootstrap operation.) The system-level instruction set may be grouped logically as follows.

- Full Word Data Transmission Instructions – Move one or more full (36-bit) words of data from one memory location to another. The instructions may also perform minor arithmetic operations, such as forming the negative (2's complement) or the absolute value of the word being processed.

- Half-Word Data Transmission Instructions – Move a half-word and possibly modify the contents of the other half of the destination location. The instructions within the group differ by the direction in which they move the selected half-word, and by the way in which they modify the other half of the destination location.

- Byte Manipulation Instructions – Pack or unpack bytes of any length anywhere within a word. The byte instructions utilize a byte pointer which allows addressing of any size byte in any position within a word.

- Fixed-Point Arithmetic Instructions – Perform scaling (multiplying by a power of 2), negating (forming 2's complement), addition, subtraction, multiplication, and division on numbers in single- and double-precision floating-point format.

- Logic Instructions – Provide the capability of shifting and rotating, as well as performing the complete set of 16 Boolean functions on two variables.

- Test Instructions – The arithmetic test instructions may jump or skip depending on the result of an arithmetic test. Also, they may first perform an arithmetic operation on the test word. The logical test instructions may skip depending on the condition of bits selected (and/or modified) by a mask.

- Program control and Stack Instructions – These include several types of jump instructions and subroutine control instructions. Pushdown stacks are handled by instructions (PUSH and POP) which, through a stack pointer, process data on a "first-in-last-out" basis. Subroutine entry and return is accomplished by jump instructions (PUSHJ and POPJ) that insert return addresses on a pushdown stack. These instructions are vital to the efficient operation of the timeshared monitor and all of the reentrant systems programs.

- String Instructions – Move, compare, and edit strings (successive bytes) of data. The instructions may be used on a variety of encoded data, including ASCII, EBCDIC, etc. The ability to detect special characters in the source string is implemented.

- Input/Output (I/O) Instructions – One group, the external I/O instructions, controls the movement of information to and from the system's peripheral devices. Both byte (8-bit) and full word (36-bit) instructions are implemented. Another group, the internal I/O instructions, controls paging, priority interrupts and other system-oriented operations by accessing selected registers in the CPU.

All KS10 instructions are 36-bit words with the format shown in Figure 4-2. Bits 0–8 specify the instruction code; bits 9–12 usually address an accumulator (AC) but are sometimes used for special control purposes or instruction code extensions. The rest of the instruction word supplies information for calculating the effective address (E) which is used to fetch the operand or alter program flow. Bit 13 (I) specifies the type of addressing (direct or indirect). Bits 14–17 specify an index register (X) for use in address modification (a 0 indicates no indexing), and bits 18–35 (Y) contain an 18-bit address.

Similar to the KL10, the KS10 implements a feature that allows expansion of the basic instruction set by an extension of the basic format to two words (refer to Figure 4-3). An EXTEND instruction (instruction code = 123 octal), when executed, points to another instruction located at the EXTEND instruction's calculated effective address. The operation codes of these extended instructions (that is, those preceded and pointed to by the EXTEND) may then form another complete instruction set in addition to the basic set. Only a few of the available extended instruction codes are currently implemented, such as those for the string instructions.

Many of the instruction codes not assigned as specific instructions are executed as unimplemented user operations (UUOs) wherein the word given as an instruction is trapped and must be interpreted by a routine included for this purpose by the programmer. Those UUOs reserved for use by the monitor are called monitor UUOs (MUUOs), while user UUOs are called local UUOs (LUUOs). Instructions that are illegal trap in the same manner as MUUOs.

In addition to the trap capability provided by UUOs, the KS10 has a trapping mechanism for direct handling of arithmetic overflow and underflow conditions, pushdown list overflow conditions, and page failures. This trap capability avoids recourse to the program interrupt system.

Instruction set mnemonics are constructed so as to closely specify the machine operation performed. For example, the mnemonic for the instruction to move full-words of data takes the form shown in Figure 4-4. It consists of a basic operator and two groups of modifiers as follows.

1. MOV is the basic operator specifying a full-word move.

2. Modifier 1 specifies how the word is to be modified when it is moved.

   E = No modification
   N = Take 2's complement
   M = Take magnitude
   S = Swap left and right halves

3. Modifier 2 specifies from where the word is to be fetched, and where it is to be moved.

   (Blank) = No memory to accumulator (the value of Y to AC).
   I = Take 18-bit address as operand (memory to AC-immediate mode)
   M = Accumultor to memory
   S = Memory to memory (self)

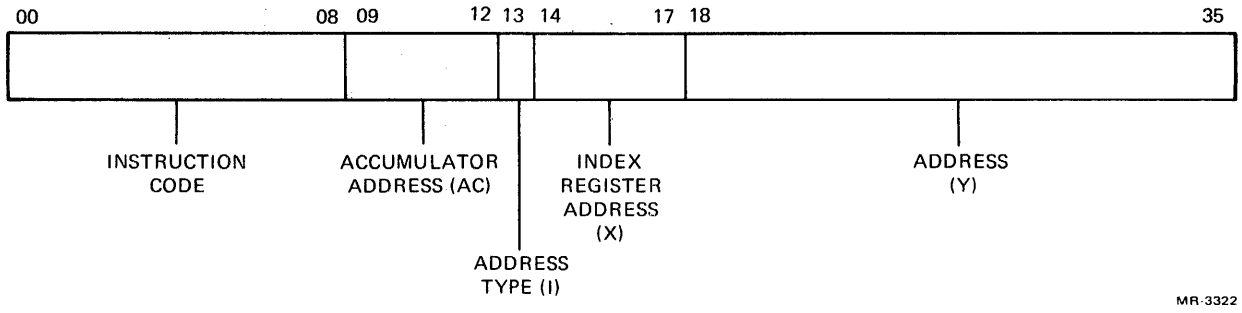


Figure 4-2 Instruction Format

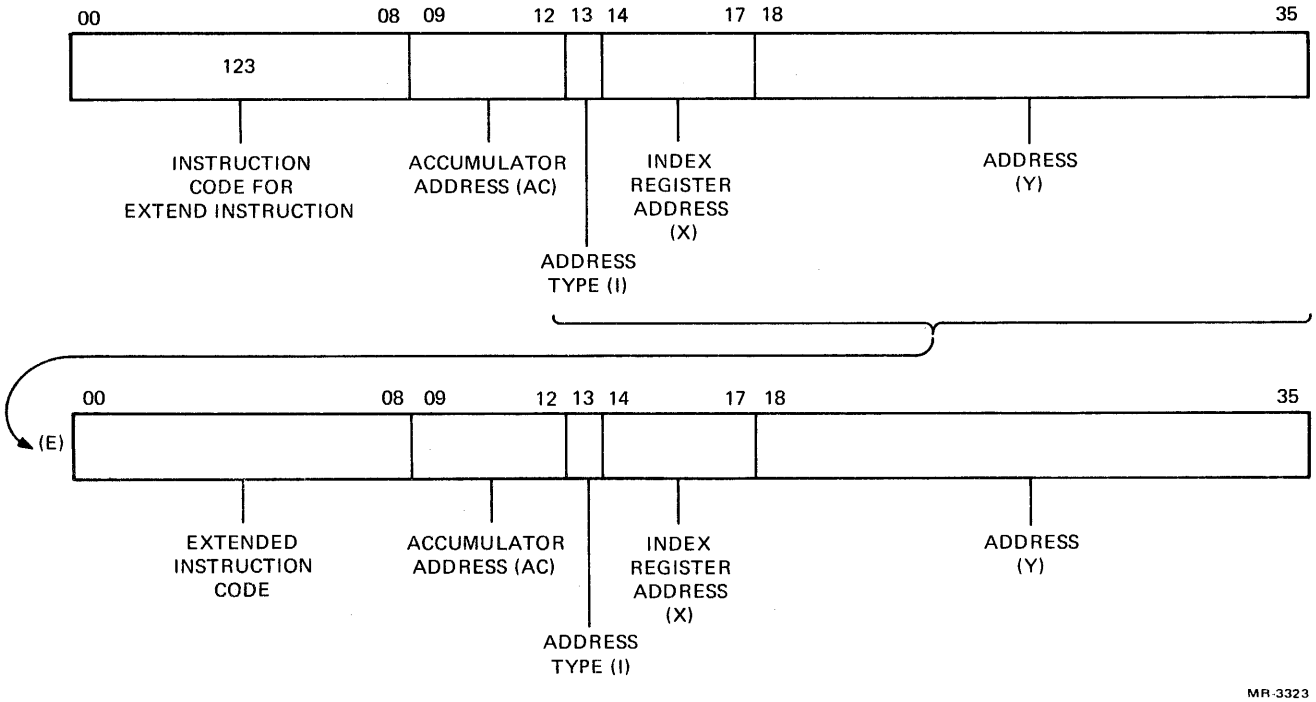


Figure 4-3 Extended Instruction Format

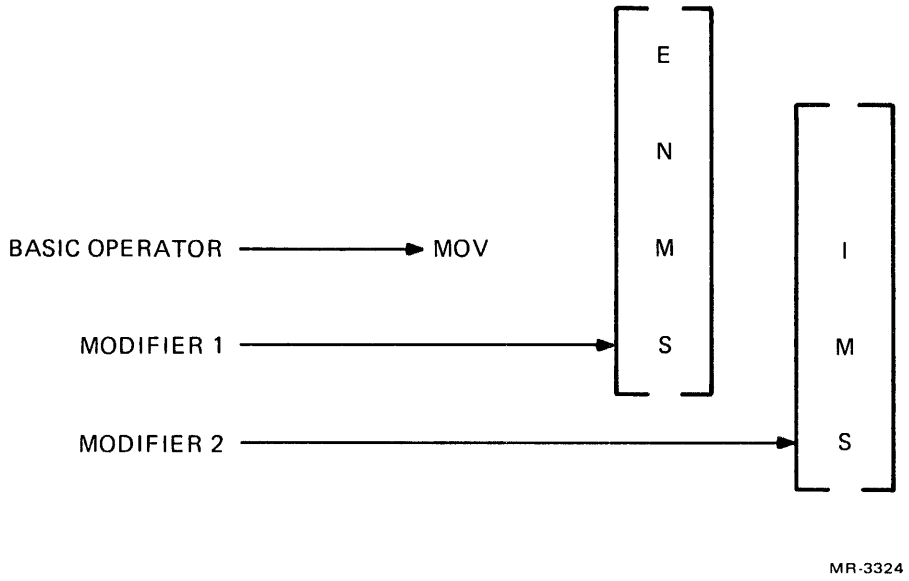


Figure 4-4 Move Instruction Mnemonic Construction

Therefore, with one broad instruction class (MOV), a total of 16 instructions may be formed. For example:

MOVE AC, ADR	Move the contents of ADR to specified AC
MOVEI AC, 5	Move the number 5 to AC
MOVEM AC, ADR	Move the contents of specified AC to ADR
MOVNM AC, ADR	Move complemented contents of AC to ADR

All 16 move instructions, together with their instruction codes and an algebraic representation for their operation, are shown in Figure 4-5.

NOTE

A complete specification for the KS10 instruction set is given in Volume I of the *Hardware Reference Manual* (EK-10/20-HR). Furthermore, Appendix A in the same document summarizes the data by listing all instruction codes and mnemonics, and by showing the algebraic representation for the complete instruction set.

MNEMONIC	INSTRUCTION CODE (OCTAL)	SOURCE TO DESTINATION
MOVE	200	(E) → (AC)
MOVEI	201	0, E → (AC)
MOVEM	202	(AÇ) → (E)
MOVES	203	IF AC = 0,: NO-OP IF AC ≠ 0: (E) → (AC)
MOVS	204	$(E)_S$ → (AC)
MOVSI	205	E, 0, → (AC)
MOVSM	206	$(AC)_S$ → (E)
MOVSS	207	$(E)_S$ → E IF AC ≠ 0: (E) → (AC)
MOVN	210	-(E) → (AC)
MOVNI	211	-[0, E] → (AC)
MOVNM	212	-(AC) → (E)
MOVNS	213	-(E) → (E) IF AC ≠ 0: (E) → (AC)
MOVM	214	|(E)| → (AC)
MOVMI	215	0, E → (AC)
MOVMM	216	|(AC)| → (E)
MOVMS	217	|(E)| → (E) IF AC ≠ 0: (E) → (AC)

SYMBOL	MEANING
AC	THE ACCUMULATOR ADDRESS IN BITS 9–12 OF THE INSTRUCTION WORD.
E	THE RESULT OF THE EFFECTIVE ADDRESS CALCULATION.
(X)	THE CONTENTS OF X.
$(X)_S$	THE CONTENTS OF X WITH THE LEFT AND RIGHT HALVES SWAPPED.
A, B	A 36-BIT WORD WITH THE 18-BIT QUANTITY A IN ITS LEFT HALF AND THE 18-BIT QUANTITY B IN ITS RIGHT HALF (EITHER A OR B MAY BE 0).
-	THE ARITHMETIC OPERATOR FOR NEGATION (OR SUBTRACTION).
| |	THE ARITHMETIC OPERATOR FOR ABSOLUTE VALUE (MAGNITUDE).

MR-3325

Figure 4-5 Move Instructions

4.3 NUMBER SYSTEM

Data words can be interpreted by the program as 36-bit unsigned binary numbers, or the left and right halves of a word can be taken as separate 18-bit numbers. Arithmetic operands used by the fixed-point arithmetic instructions use 2's complement representations to do binary arithmetic. As shown in Figure 4-6, bit 0 (the leftmost bit) represents the sign: 0 for positive, 1 for negative. In a positive number, the remaining 35 bits represent the magnitude in ordinary binary notation. The negative of a number is obtained by taking its 2's complement. Zero is represented by a word containing all 0s

NOTE

The 2's complement is formed by taking the logical complement (that is, complementing each bit in the word including the sign bit) and adding 1 to the result. For example, the 2's complement of +100 (octal) is the logical complement 777777 777677 plus 1, which equals 777777 777700 (-100 octal).

Two common conventions are to regard a number as an integer (binary point at the right) or as a proper fraction (binary point at the left). In these two cases, the range of numbers represented by a single word is -2^{35} to $2^{35}-1$, or -1 to $1-2^{-35}$. Because multiplication and division make use of double-length numbers, there are special instructions for performing these operations to yield results that can be represented by a single word.

As shown in Figure 4-7, the format for double-length fixed-point numbers is an extension of the single-length format. The magnitude (or its 2's complement) is the 70-bit string in bits 1–35 of the high- and low-order words. Bit 0 of the high-order word is the sign and bit 0 of the low-order word is made equal to the sign. The range for double-length integer and proper fractions is thus -2^{70} to $2^{70}-1$, or -1 to $1-2^{-70}$.

The KS10 also processes both single- and double-precision floating-point numbers. Format for single-precision operands is shown in Figure 4-8. A single-precision floating-point instruction interprets bit 0 as the sign, but interprets the rest of the word as an 8-bit exponent and a 27-bit fraction. Normalized single-precision floating-point numbers have a fraction that ranges in magnitude from 1/2 to $1-2^{-27}$.

NOTE

A floating-point number is considered normalized if the magnitude of the fraction is greater than or equal to 1/2 and less than 1. The KS10 may not give the correct result if the program supplies an operand that is not normalized, or that has a zero fraction with a nonzero exponent.

A double-precision operand consists of the sign, an 8-bit exponent, and a 62-bit fraction as shown in Figure 4-9. (The high-order word has the same format as that used for a single-precision number.) Increasing the length of a number to two words does not significantly change the range, but rather increases the precision. (The fraction ranges in magnitude from 1/2 to $1-2^{62}$ for double-precision numbers.) In any format, the magnitude range of the normalized fraction is from 1/2 to 1, decreased by the value of the least significant bit. In all formats, the exponent range is from -128 to +127.

4.4 EFFECTIVE ADDRESS CALCULATION

Bits 13–35 have the same format in all KS10 instructions. As shown in Figure 4-2, I (bit 13) is the indirect bit, X (bits 14-17) is the index register address, and Y (bits 18-35) is the address field. The index register address (1–17 octal) is actually the address of an accumulator.

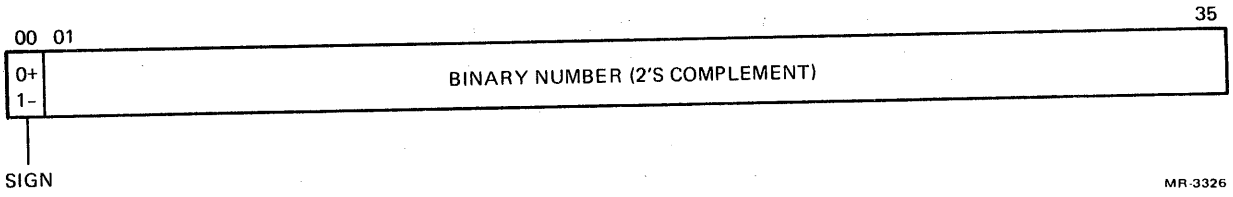


Figure 4-6 Single-Length Fixed-Point Operand

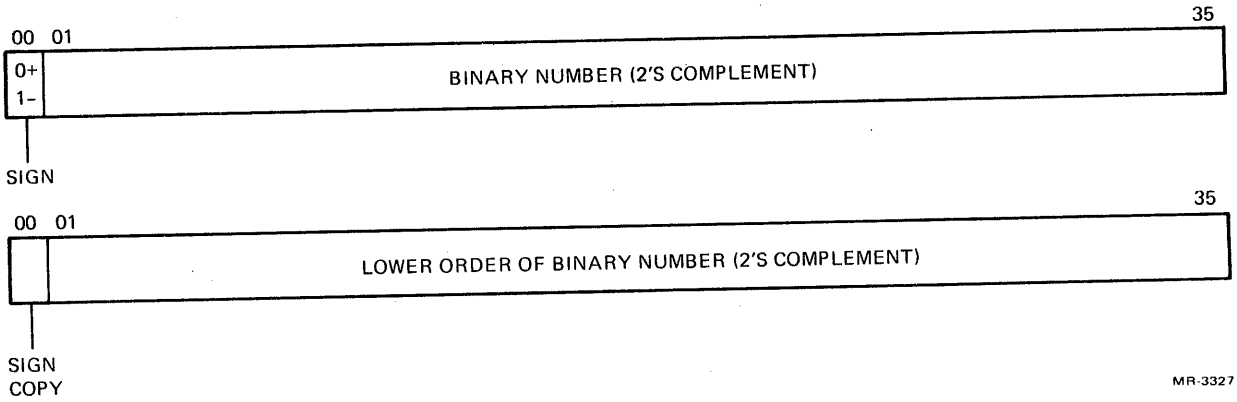


Figure 4-7 Double-Length Fixed-Point Operand

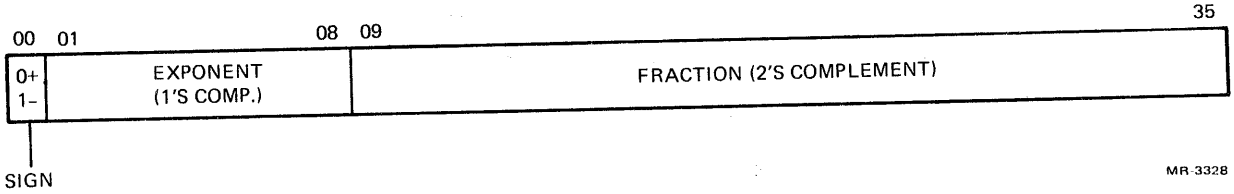


Figure 4-8 Single-Precision Floating-Point Operand

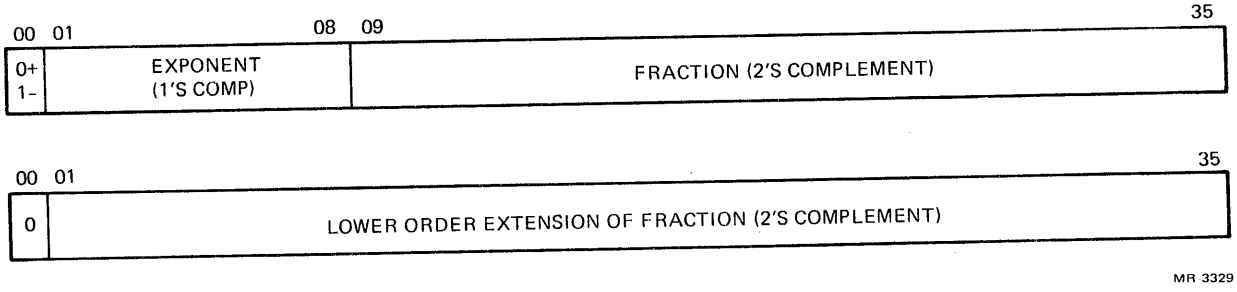


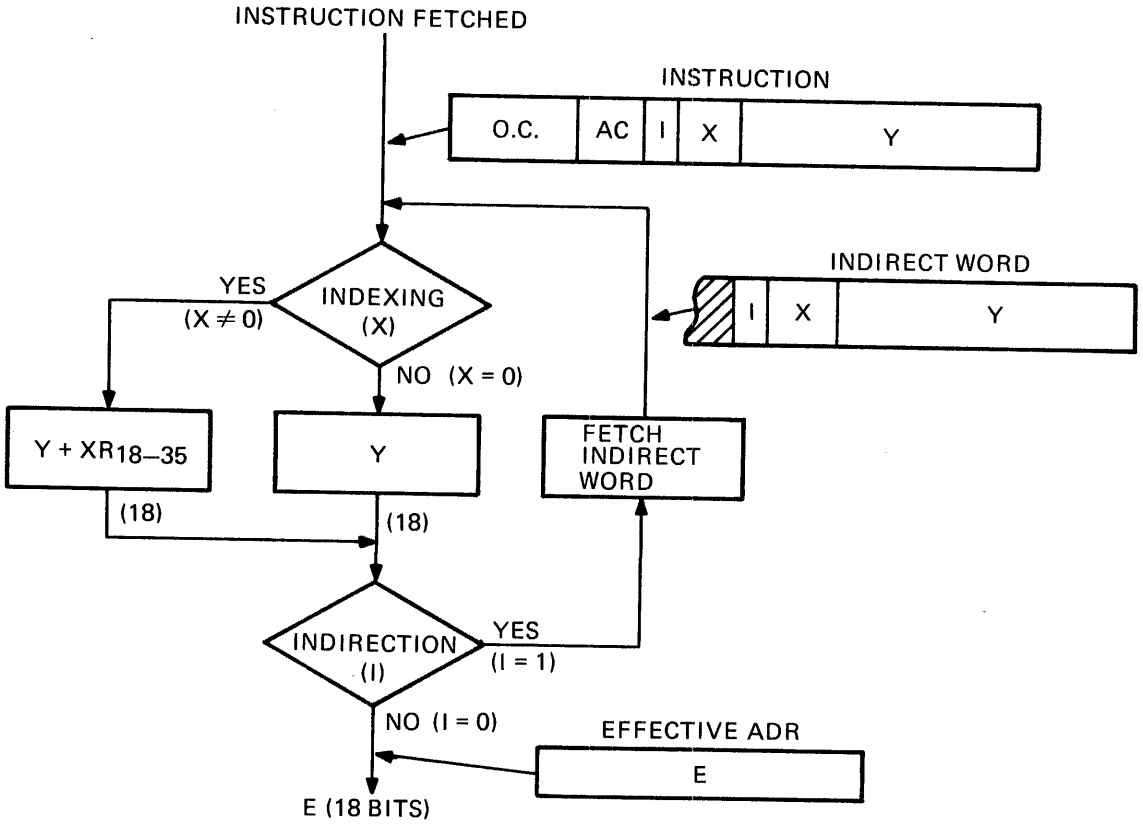
Figure 4-9 Double-Precision Floating-Point Operand

Following the instruction fetch by the CPU, the effective address (E) for all KS10 instructions except the external I/O instructions is calculated from I, X, and Y as described below. The process is diagrammed in Figure 4-10.

1. First, if there is indexing ($X \neq 0$), the right half (bits 18–35) of the index register contents are added to the 18-bit value of Y. If there is no indexing ($X = 0$), Y is not modified.

2. Next, if there is no indirection ($I = 0$), the result of the first step, which is Y or Y + XR (bits 18–35), becomes the 18-bit value for E. If there is indirection ($I = 1$), Y or Y + XR (bits 18–35) is used as an address to fetch another word (called an indirect word) from memory.

3. The indirect word is processed in the same manner as the instruction. As in the previous steps, the X and Y parts of the word determine the value of E if there is no indirection. If there is indirection, yet another indirect word is fetched from memory. The effective address calculation continues until a word is fetched having the indirect bit equal to 0, in which case the X and Y fields of the word determine the value of E.



NOTES:
AC = ACCUMULATOR ADDRESS
O.C. = OP CODE
E = EFFECTIVE ADDRESS
I = INDIRECT BIT
X = INDEX REGISTER ADDRESS
XR = INDEX REGISTER CONTENTS
Y = ADDRESS FIELD

MR-3330

Figure 4-10 KS10 Effective Address Calculation (Not Valid for External I/O Instructions)

The following examples illustrate the effective address calculation.

Example 1 - Direct Addressing

MOVE AC, 1000

With no indexing or indirection, the MOVE instruction's effective address is equal to Y; that is, E = 1000.

Example 2 - Indexed Addressing

MOVE AC,1000 (5) where the contents of index register 5 = 100.

The instruction uses indexing; thus the effective address is the index register contents added to Y; that is, Y = 1100.

Example 3 - Indirect Addressing

MOVE AC, @ 1000 where the contents of 1000 = 4000

The indirect bit is set, which causes an indirect word fetch. The contents are then used as the effective address; that is, E = 4000.

Example 4 - Indexed and Indirect Addressing

MOVE AC, @ 1000 (5) where the contents of index register 5 = 100 and the contents of 1100 = 5000.

Both indexing and indirection are used. The result of the indexing (1100) is the address of the indirect word. The contents then become the effective address; that is, E = 5000.

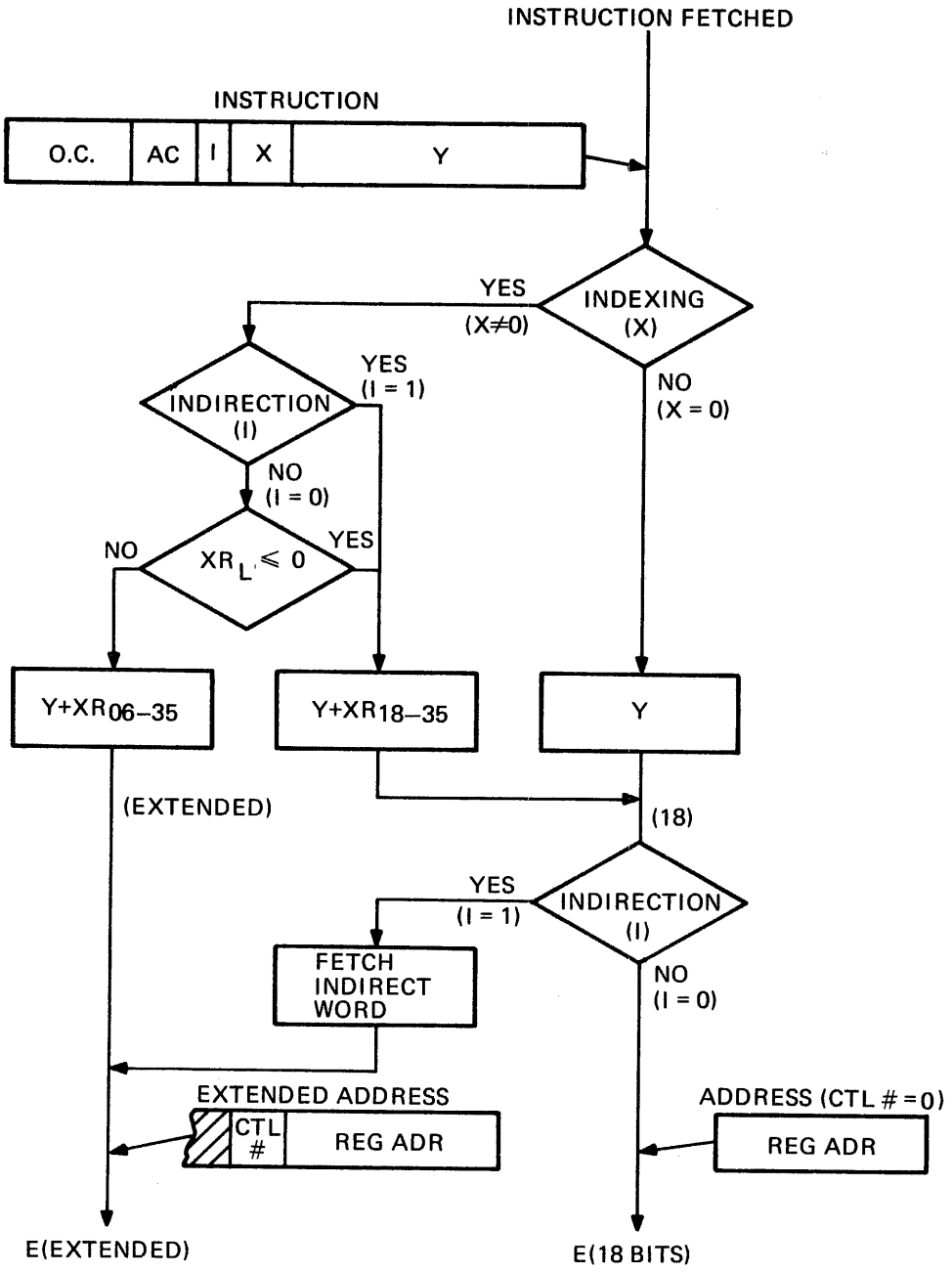
The effective address calculation for the external I/O instructions (instruction codes 710_8-727_8) is diagrammed in Figure 4-11. The calculation differs from other KS10 instructions in that the result, an I/O address, can be either an 18-bit or an extended (greater than 18-bit) address. An extended address is necessary because an I/O address consists of a 4-bit controller number plus an 18-bit register address as shown in Figure 4-12. Controller numbers 1 and 3 select a Unibus adapter (UBA), in which case the register address selects a register in a Unibus device connecting to the addressed UBA, or it selects a register in the UBA itself. Controller number 0 is used to address KS10 registers not associated with a Unibus (for example, memory status register).

NOTE

If the controller number is to be 0, the I/O address need not be extended. That is, an 18-bit effective address calculation results in the hardware forcing the controller number to 0.

The effective address calculation for the external I/O instructions is as follows.

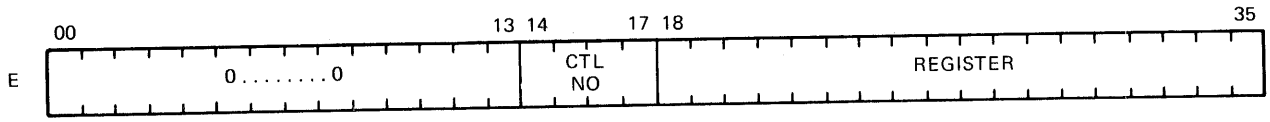
1. As for the other KS10 instructions, Y is not modified if there is no indexing (X = 0). Also, if there is indexing (X ≠ 0), Y is added to the right half (bits 18–35) of the index register, but only if the left half of the index register is negative (bit 00 = 1) or if the indirect bit is set (I = 1). If the indirect bit is not set (I = 0), and if the left half of the index register is positive (bit 00 = 0), Y is added to bits 06–35 of the index register to generate an extended effective address.



NOTES:
- AC = ACCUMULATOR ADDRESS
- O.C. = OP CODE
- E = EFFECTIVE ADDRESS
- I = INDIRECT BIT
- X = INDEX REGISTER ADDRESS
- XR = INDEX REGISTER CONTENTS
- Y = ADDRESS FIELD

MR-3331

Figure 4-11 KS10 Effective Address Calculation for External I/O Instructions



CONTROLLER NUMBER	REGISTER ADDRESS (OCTAL)	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 4-12 I/O Address Format

2. Next, if an extended effective address has not been generated in the first step, the resulting 18-bit address (Y or Y + XR 18–35) is used as the effective address provided there is no indirection (I = 0). If there is indirection (I = 1), the 18-bit address is used to fetch an indirect word from memory.

3. When an indirect word is fetched from memory, bits 14–35 of the contents are unconditionally used to generate an extended address. Unlike the address calculation for the rest of the KS10 instruction set, the indirect word is not treated like another instruction word. There is only one level of indirection employed for external I/O instructions.

As can be seen, to address controllers other than 0 (which require an extended address), instructions using indexed or indirect addressing must be used. Examples of indexed and indirect addressing follow. In both examples the status register (register address = 763100 octal) in UBA 1 (controller no. = 1) is read into an AC with the RDI0 external I/O instruction.

NOTE
UBA and Unibus device register addresses are given in Appendix B.

Example 1 – Indexed Addressing

RDIO AC, 763100 (5) where the contents of index register = 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 – Indirect Addressing

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.

4.5 MACHINE MODES

A program running in the KS10 operates in one of two modes: executive (exec) mode or user mode.

In exec mode, all implemented instructions are legal. The operating system operates in exec mode and is thus able to control all systems resources and the state of the processor; that is, it handles system I/O and priority interrupts, constructs page maps, and handles the general management of the system for all users.

In user mode, certain instructions that can compromise system integrity or affect other users are illegal. For example, I/O instructions are illegal, causing a trap to the operating system. Users are required to issue UUOs for system services such as I/O.

4.6 PROCESS TABLES

Special tables are set up in memory by the TOPS-10 or TOPS-20 operating system to aid in the system management of both exec and user processes. The operating system keeps an executive process table (EPT) for its own use, and a user process table (UPT) for each user on the system. Each table is a single page (512 words). EPT and UPT configurations for both TOPS-10 and TOPS-20 are shown in Figures 4-13 and 4-14.

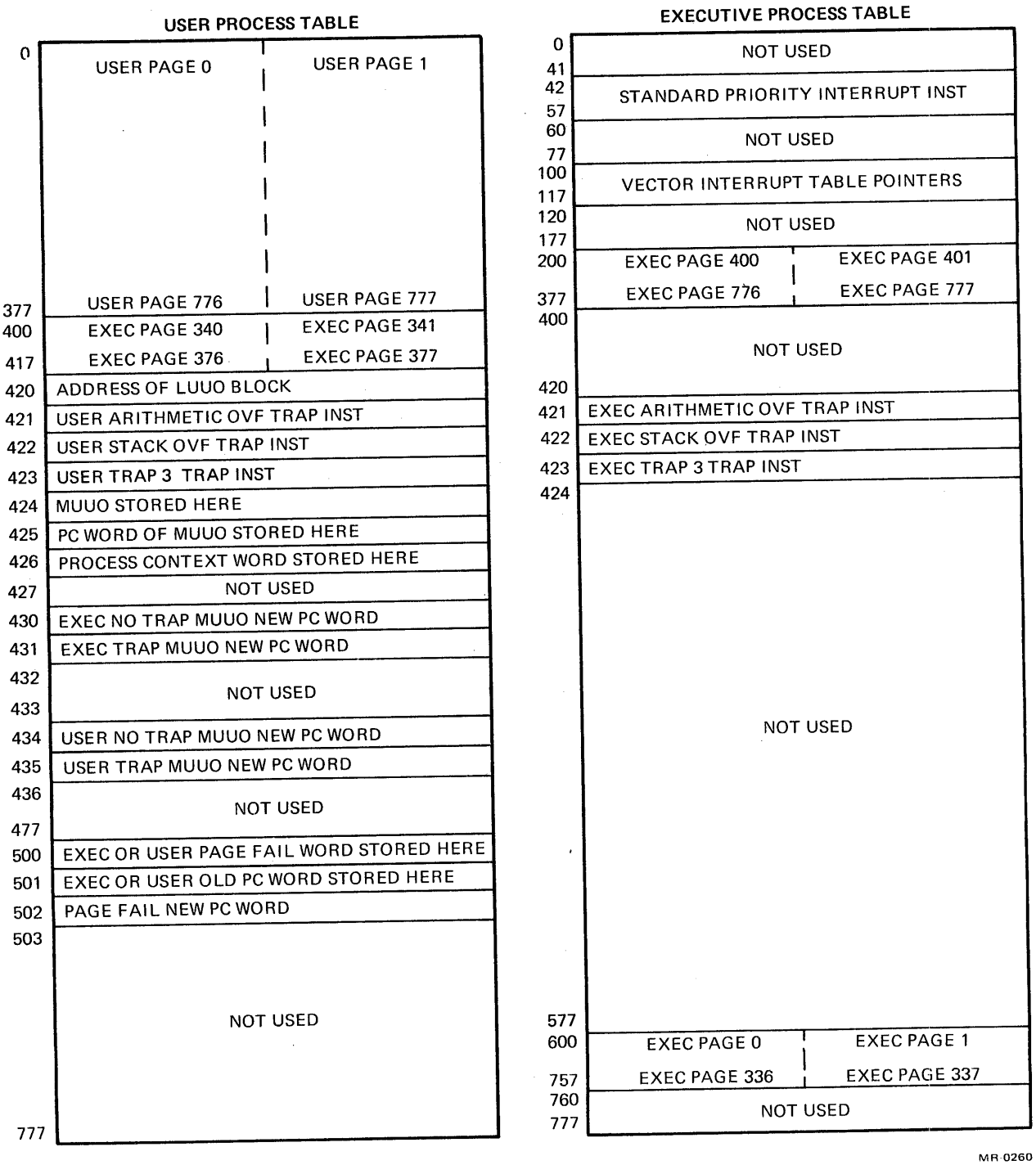


Figure 4-13 KS10 EPT/UPT (TOPS-10 Paging)

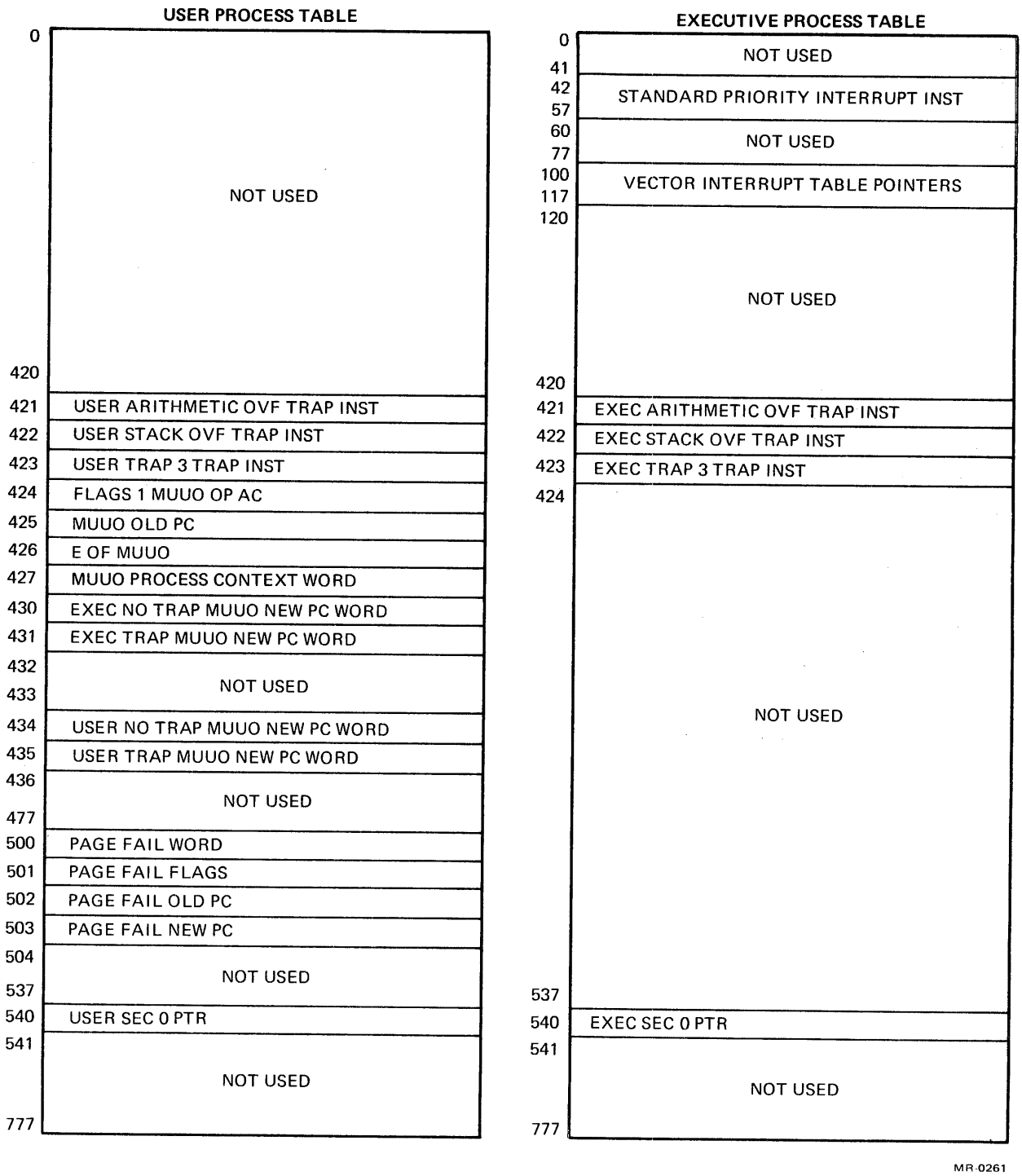


Figure 4-14 KS10 EPT/UPT (TOPS-20 Paging)

The process tables contain page maps, trap and interrupt instructions, pointers, and other information referenced by (and acting as a bridge between) the hardware and software. Parts of the table are not allocated for hardware reference (labeled "not used" in the figures) and are available to the operating system for various systems management functions. For example, in each UPT, the operating system generally keeps a stack for use with the process, the job tables, and the various user statistics such as memory space and billing information.

The address (physical page number) of the EPT is kept in a hardware register called the executive base register (EBR). Its value is defined during system initialization when the operating system is loaded. The address of the UPT for the user program currently running on the system is kept in a similar hardware register called the user base register (UBR). The UBR is loaded by the operating system during a context switch; that is, when switching from one user to another.

4.7 MEMORY ADDRESS MAPPING BY THE CPU

All KS10 memory, both virtual and physical, is divided into pages of 512 words each. The maximum virtual memory space addressable by a program running in the CPU (that is, resulting from an instruction's 18-bit effective address calculation) is 512 pages or 256K words. The physical memory space is currently 1024 pages or 512K words with address provision (20 bits) for future expansion to 2048 pages or 1024K words.

Both virtual and physical memory addresses consist of a page number and a line number. The virtual page number consists of the 9 most significant bits (bits 18-26) of the 18-bit virtual address; the physical page number consists of the 11 most significant bits (bits 16-26) of the 20-bit physical address. The line number, which specifies the word within the page consists of the 9 least significant bits (bits 27-35) of either the virtual or physical address.

The KS10 CPU translates a virtual address to a physical address by replacing the virtual page number with a physical page number. The line number of the virtual address is not altered; that is, a given word within a virtual page is the same word within the corresponding physical page.

NOTE

No address translation is made when the general-purpose registers (memory addresses 00-17 octal) are addressed. These hardware registers are considered to be in physical address space. They may be addressed by any program, although the same address may be in different blocks for different programs. Refer to Paragraph 4.9.

Whenever a virtual to physical address translation is required, it is carried out automatically by the paging hardware as shown in Figure 4-15. The 9-bit virtual page number selects one of 512 locations in a hardware page table (also called the pager) and the page table contents provide the corresponding 11-bit physical page number, provided the addressed page has been allocated space in physical memory. If the addressed page has been allocated space in memory, it is indicated by the state of an access bit (called the valid bit) in the page table entry. Other control bits specify whether a page can be altered or not, whether it is a user or exec mode address, and whether the cache can be used for references to it.

When the relocation data for a referenced page does not exist in the hardware page table, a page table refill cycle occurs during which the hardware reads the relocation data from page maps set up in memory for each program. The page maps are created by the operating system. In systems running TOPS-10, the page maps are in the process tables (EPT and UPT). In systems running TOPS-20, the EPT and UPT contain a pointer to the page maps (or to yet another pointer) which are elsewhere in memory.

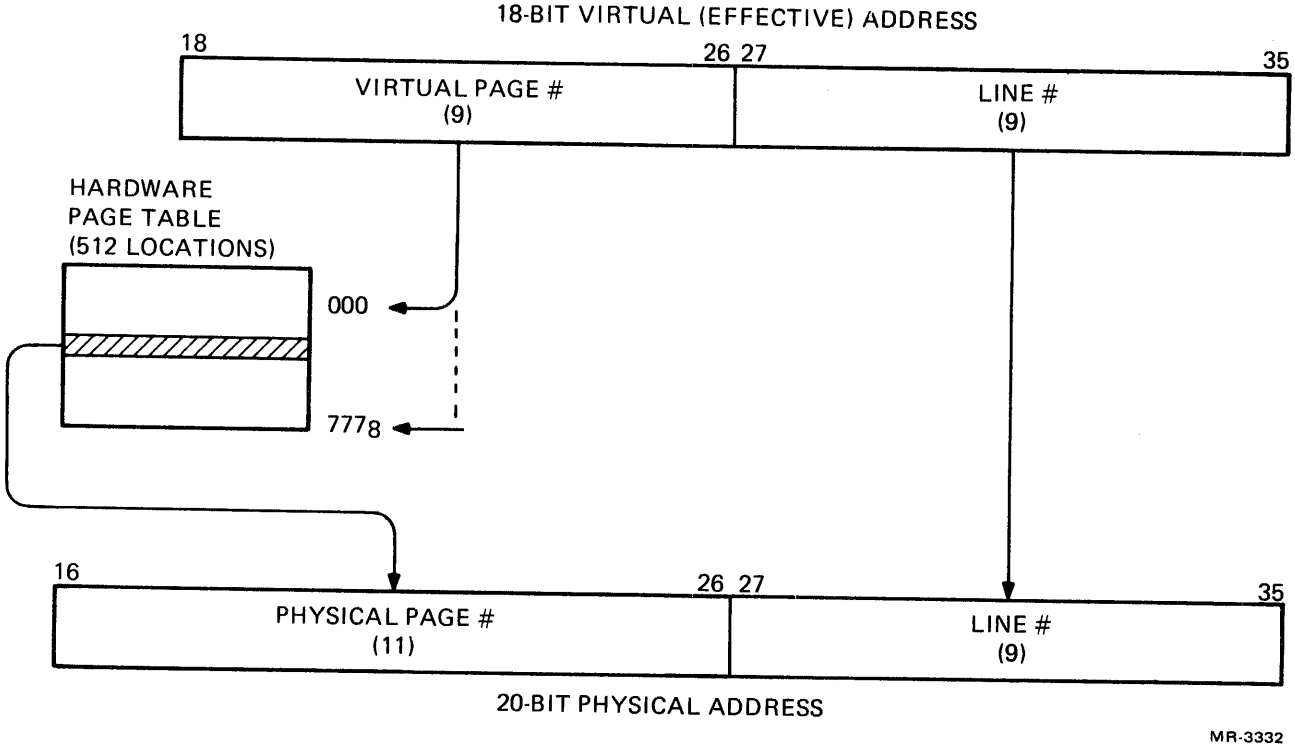


Figure 4-15 CPU Virtual to Physical Address Conversion

If valid relocation data for the referenced page is not obtained as a result of the refill cycle, a page fail trap to the operating system occurs. When the trap occurs because the page has not been allocated a place in memory (valid bit not set), the operating system may then assign a physical address and update the page mapping information accordingly (pages may be transferred to/from the disk area to manage the available address space). A retry by the program then causes a page refill and an entry to be made in the hardware pager. As the running program references memory, the hardware pager continues to be filled with mapping information. In most cases, the pager will eventually have enough entries to eliminate the need for further references to memory for paging information.

In summary, the operating system assigns the physical address space for each user (and itself) by loading the appropriate page mapping information in the process tables and (for TOPS-20*) elsewhere in memory. In addition, the operating system provides memory protection for each user (and itself) by filling the page maps – and thus the hardware pager – with only those entries which are allowed to be accessed by a given user. Advantages of paged memory management are that it does the following.

- Allows extensive sharing of data and programs on a page-by-page basis by multiple users.
- Allows access to the entire physical memory space even though the maximum user addressing capability is usually smaller.

*TOPS-20 memory management also employs a shared pages table, which holds the physical addresses for pages used by more than one user; and a core status table, which stores information about how long a page has been in physical memory and the number of processes sharing it.

- Requires that only a portion of a program need be in physical memory at any given time.
- Allows dynamic changes in a user's address space as size requirements change during execution.

4.8 MEMORY ADDRESS MAPPING BY THE UBA

The UBA converts the 18-bit virtual (Unibus) address specified by a peripheral device during an NPR transfer into a 20-bit physical (KS10 MOS memory) address. This is similar to the 18-bit virtual to 20-bit physical memory address translation in the CPU. The virtual address space addressable by the Unibus device is 64 pages or 32K words. (A KS10 page is 512 words.) The physical address space is currently 1024 pages or 512K words.

With reference to Figure 4-16, six of the seven most significant bits of the Unibus address (bits 16–11) specify the virtual page number. (Bit 17 is not used and must be 0.) The next nine least significant bits of address (bits 10–02) specify the line number, that is, the word within the page. Note that the two least significant bits of Unibus address, the word and byte bits, are not part of the memory address. The values of these bits specify the position of the Unibus data within the addressed memory word. For example, the word bit (bit 01) specifies in which half of the KS10 memory word the Unibus data (word or byte) is to be placed. The byte bit (bit 00) specifies whether the byte is high- or low-order.

Virtual to physical address translation is made by replacing the virtual page number with a physical page number. In the UBA, as in the CPU, the virtual page number selects a location in a hardware page table to supply the 11-bit physical page number. Also, there is no line number translation since a given word within a virtual page is the same word within a physical page.

The UBA's hardware page table, called the paging RAM, has 64 locations (one for each virtual page number) and is loaded by the program. In addition to the physical page number, a RAM location also contains an access bit to indicate the entry is valid. Three other control bits specify a fast or slow transfer, read reverse mode, and the masking of the high-order bits in the left and right halves of the KS10 memory word during non-18 bit transfers to the Unibus. (The latter control bit, called the disable bit, prevents nonzero memory data from causing false parity indications in the Unibus device.) The paging RAM also contains a parity bit. If a RAM parity error is detected or if the access (valid) bit is not set during a NPR transfer, it causes the associated Unibus device to time-out, set an error flag, and terminate the data transfer.

4.9 GENERAL-PURPOSE REGISTER BLOCKS

The general purpose registers, also called "AC blocks" or "fast ACs," consist of 8 blocks of fast memory with each block consisting of 16 36-bit locations. These eight sets of registers are RAM locations contained in the CPU (DPE module) and are not part of MOS memory. They are used as accumulators, index registers, or the first 16 locations in memory depending on how they are addressed by the instruction word. That is, they may be addressed by the instruction's 4-bit accumulator and index register fields, or as ordinary memory locations resulting from the instruction's effective address calculation. Access time for the general-purpose registers is approximately 300 ns.

Generally, only one block of general-purpose registers is available to a program at any given time. As indicated in Figure 4-17, the operating system usually reserves block 0 for its own use and assigns block 1 to the current user program. Blocks 2–6 are also used by the operating system, principally when handling priority interrupt (PI) activity at the various interrupt levels. Block 7 is reserved for microcode use.

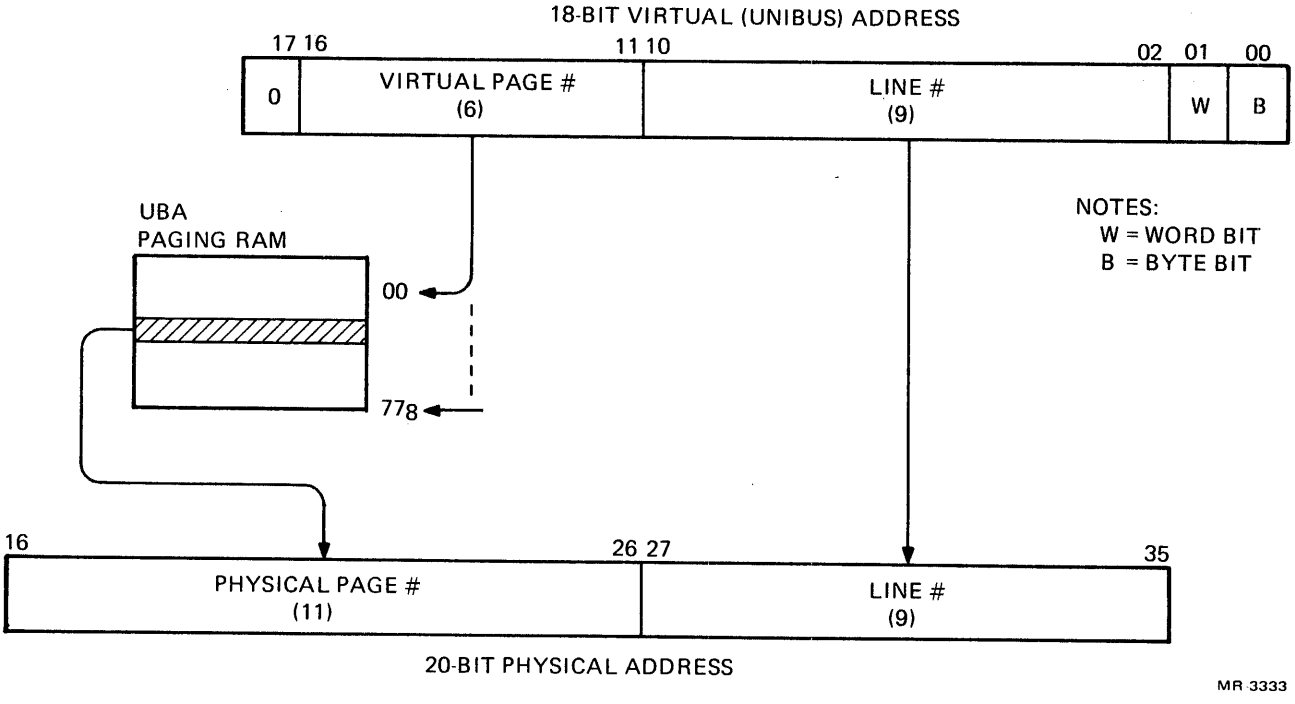


Figure 4-16 UBA Virtual to Physical Address Conversion

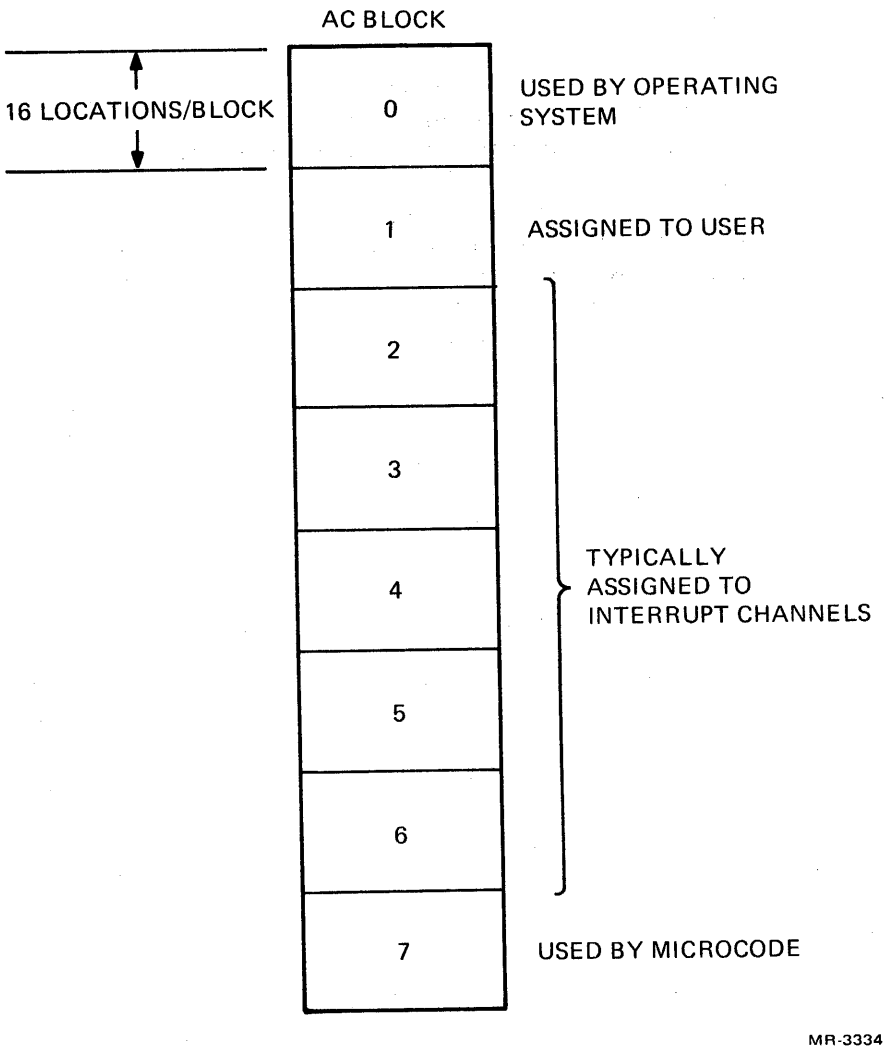


Figure 4-17 AC Block Usage

4.10 MEMORY SYSTEM

KS10 main memory consists of a single memory controller module that connects to the internal bus, plus between 2 and 8 storage array modules that utilize metal-oxide semiconductor (MOS) 16K RAMs as the storage elements. The memory is a single-port system with a memory cycle time of 0.90 μsec. It has a minimum capacity of 128K words that can be expanded to 512K words in 64K increments (one storage array module = 64K words). Word length is 36 data bits plus 7 error detection and correction bits. Memory interleaving is not implemented.

NOTE

In MOS memory implementation, data is stored by charging (or not charging) the effective capacitance of each storage cell. These charges, which are extremely small, must be periodically renewed when memory is holding data (memory refresh cycle = 15 μsecs). Also, unlike the core memories in previous 10/20 machines, all stored information is lost if power is removed from the system.

The seven error correction and detection bits stored with each data word provide for single-bit error corrections and double-bit error detection when data is read from memory. If the error is one of the 36 data bits, the bad bit is automatically corrected by the hardware. If the error is one of the seven check bits, no corrective action is required. When a double-bit error occurs, the hardware cannot determine which two bits are bad. However, it does detect the error condition causing a page fail trap by the CPU. The operating system may then examine the failing address and take the appropriate action.

The KS10 uses a 512 word high-speed RAM cache or buffer memory to speed overall operation. It is the same RAM that is used for the general-purpose registers, and access time is approximately 300 ns. Memory read data is found in the cache a high percentage of the time (90 to 95 per cent in some instances), thus greatly reducing the effective memory-access time for the system.

The cache is loaded in the following manner. When a data word is read from or written to main (MOS) memory, the word is also stored in a cache location addressed by the line number. (Refer to Figure 4-18.) A cache directory, another high-speed RAM addressed by the line number, is also written at the same time to store the page number corresponding to the line number together with a valid bit if the memory address is a virtual address. (Physical addresses are invalid. The cache is written when a physical reference is made but no later use can be made of the data.)

Data is taken from the cache and not from main memory (refer to Figure 4-19) if the memory reference is a memory read and the page numbers in the memory address and the cache directory match; that is, if the word has previously been accessed in memory (read or written) and the word is in the cache for a quick retrieval. This is called a cache hit. Of course the valid bit in the directory must also be set. In addition, the cache must be enabled and the referenced page must be cacheable and have a valid mapping as described in Paragraph 4.7.

4.11 PRIORITY INTERRUPT SYSTEM

Priority interrupt (PI) requests on the KS10 system are generated by the various Unibus peripheral devices as well as the processor itself. The requests are handled on eight levels or channels (0–7) arranged in a priority sequence. Channel 1 has the highest priority; channel 7 has the lowest priority. (A channel number equal to 0 inhibits interrupt activity.) A PI level is assigned by the program by loading a 3-bit PI channel number assignment (PIA) in the appropriate control register. For example, the PIA for the processor is loaded by writing the processor's control and status register with an internal I/O instruction (i.e., WRAPR). The instruction also allows the program to generate an interrupt on command and these "software interrupts" are generated by the operating system for user scheduling purposes, time of day, etc. Interrupts by the processor cause a jump in program execution to 40 + 2n in the EPT, where n is the interrupt channel (1–7).

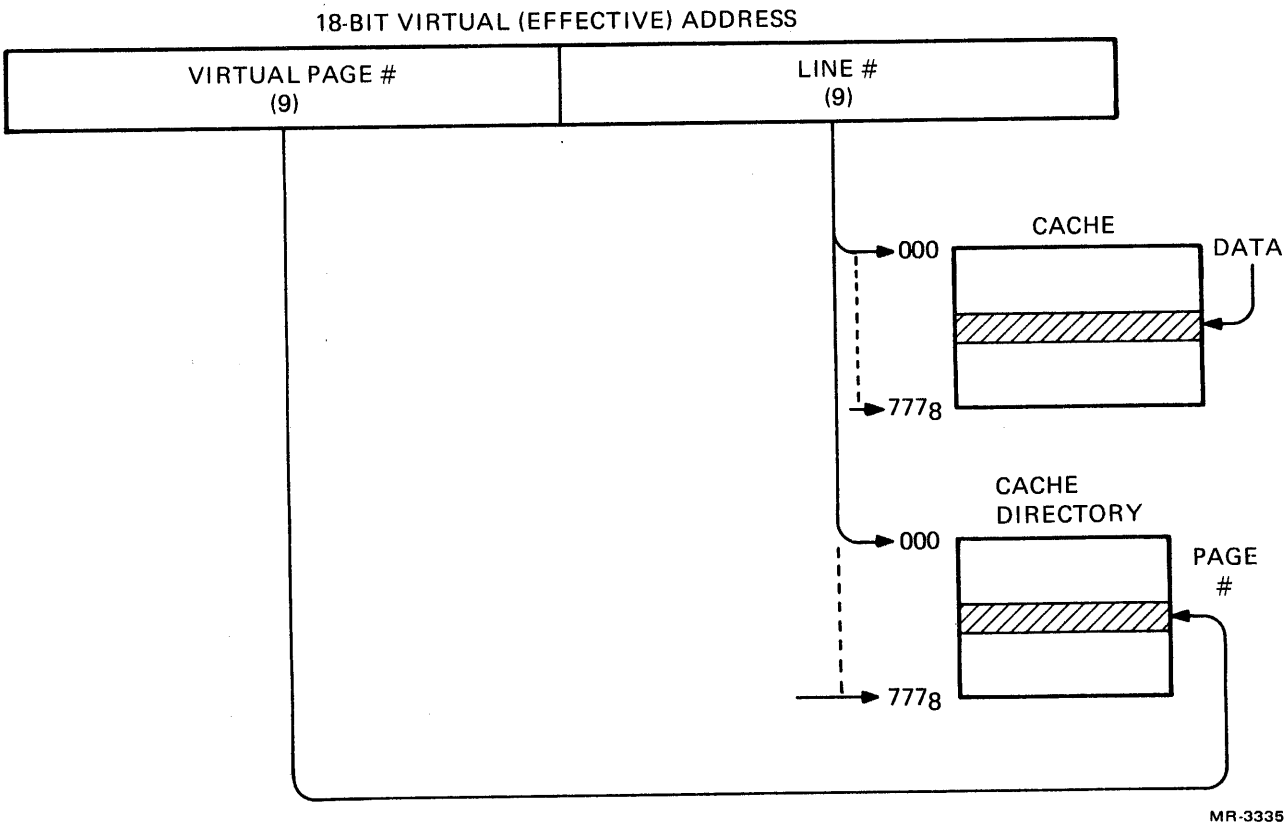


Figure 4-18 Loading the Cache

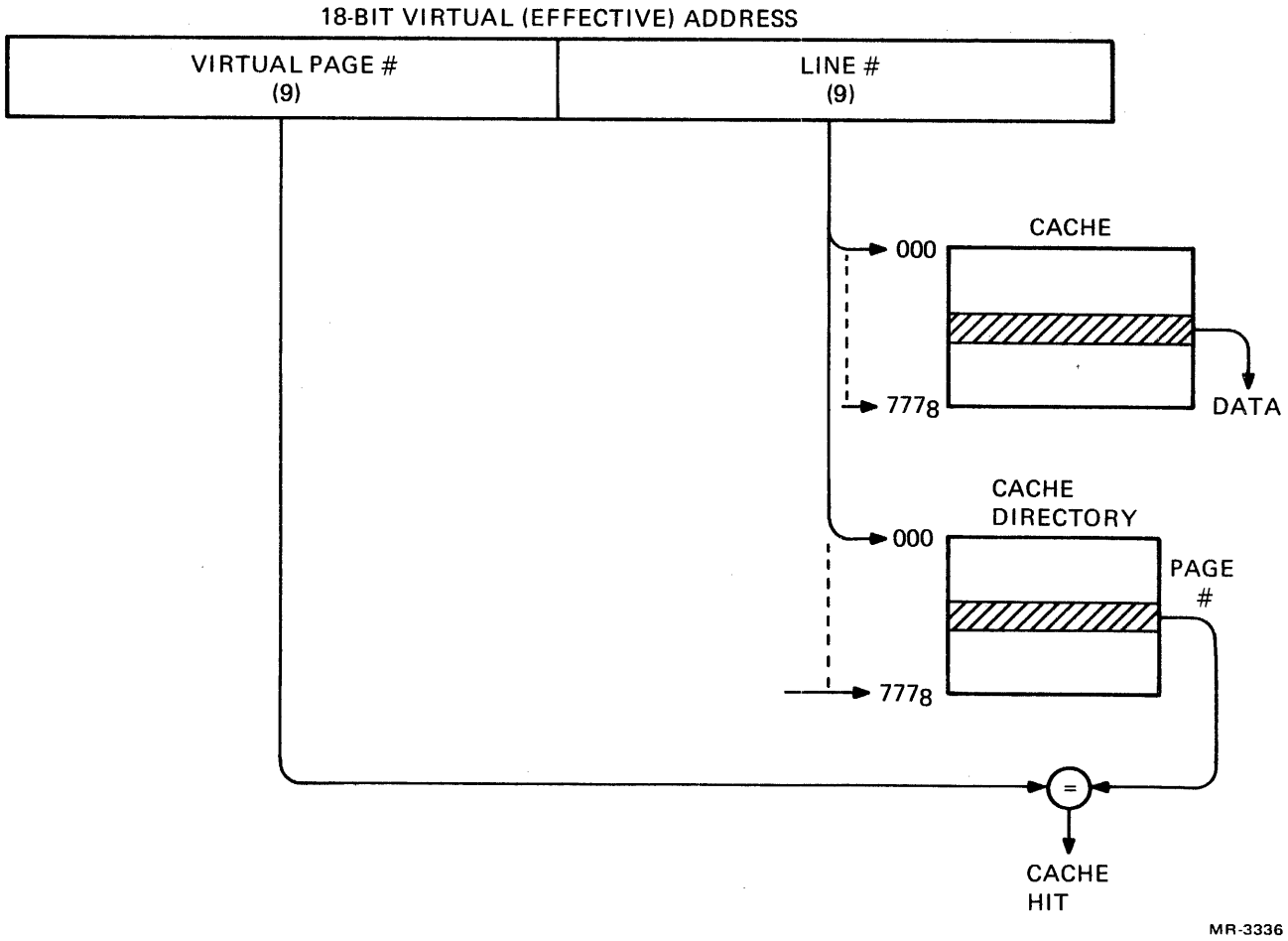


Figure 4-19 Reading the Cache (Cache Hit)

The PIA for a peripheral device is loaded by writing the 3-bit channel number in the UBA's control and status register with an external I/O instruction (i.e., WRIO). Two levels of PIA are provided, thus allowing one group of Unibus devices to interrupt on one PI channel, and a second group to interrupt on another. When conditions are met for a Unibus device interrupt (end of transfer, read error, etc.), a vector is transferred from the device on the Unibus, through the UBA, and over the KS10 bus to the CPU. (Vector addresses for Unibus devices are given in Appendix B.) The CPU, after first referencing an EPT location determined by the UBA's controller number (EPT + 100 octal + CNTRL #) to obtain the address (T) of a table, uses the vector to execute the instruction at T + VECTOR/4.

4.12 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. Two important status words within the halt status block are the microcode flag word and the VMA word. 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.

- **Halt Status Word** – A processor halt causes a halt status code to be stored in physical memory location 0 (not AC 0). 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 4-20.

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

- **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 4-21 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 (Appendix A) allows the program to load any address value x. Initially, when the microcode is started, the halt status block address is set to a value of x = +376000 (octal) and the halt status block is stored in memory locations 376000–376021. 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 1), the current instruction, the EBR and UBR, the microcode flags, and the PI system status. The PI system status is the same as that read by the RDPI instruction (Appendix A). The VMA contents (plus flags) are also stored in the next to last halt status block location.

- **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 (octal). The page fail code specifies the operation for which the page failure occurred. Status word bit format and page fail code definitions are given in Figure 4-22.

- **VMA** – The virtual memory address (VMA) and VMA flags are stored in location x + 20 (octal) of the halt status block. Bit format and definitions are given in Figure 4-23.

- **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 4-24.

- **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 4-25.

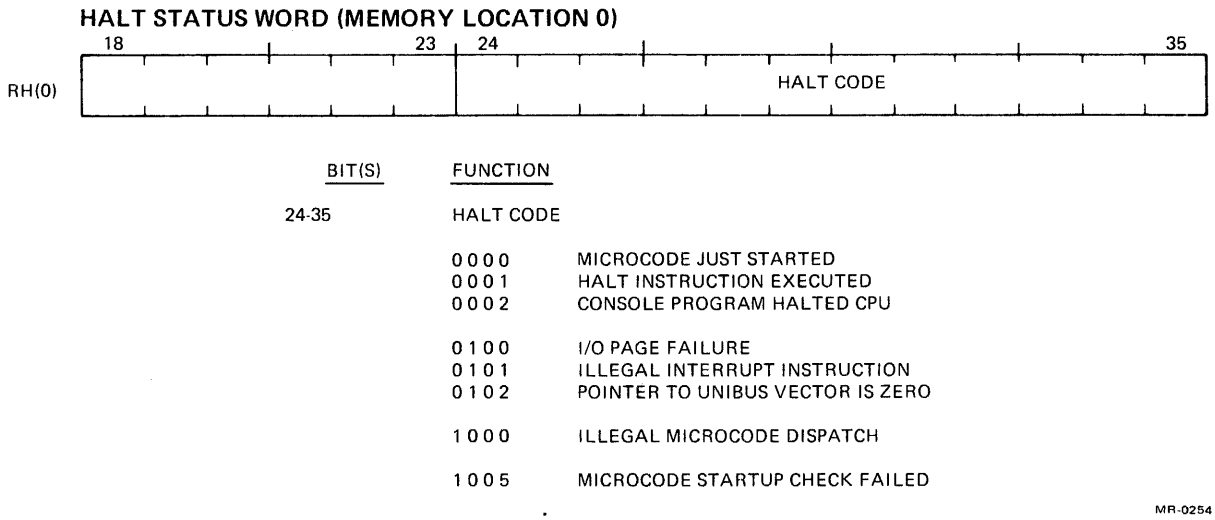


Figure 4-20 Halt Status Word

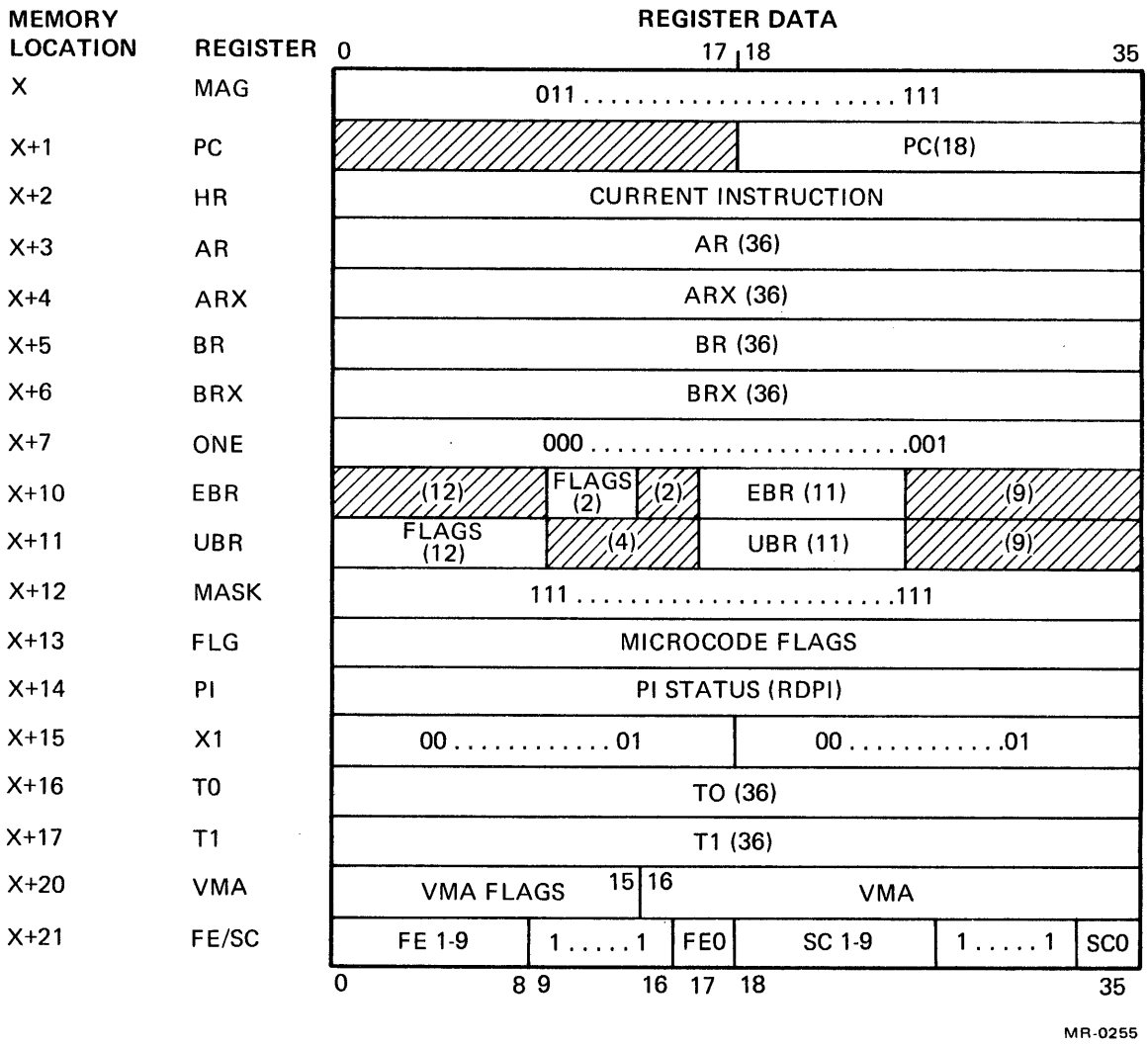
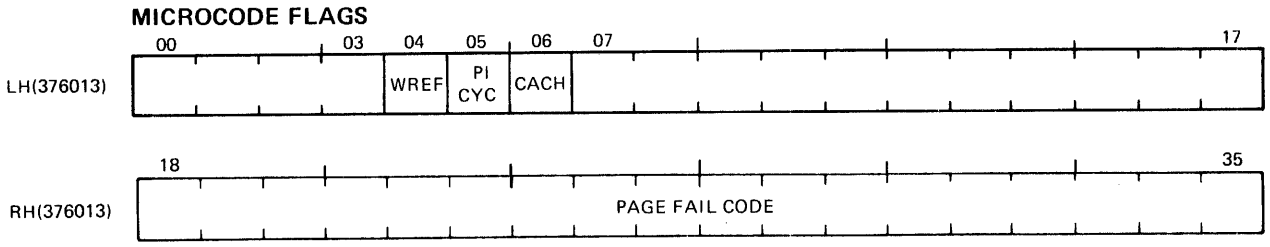


Figure 4-21 Halt Status Block

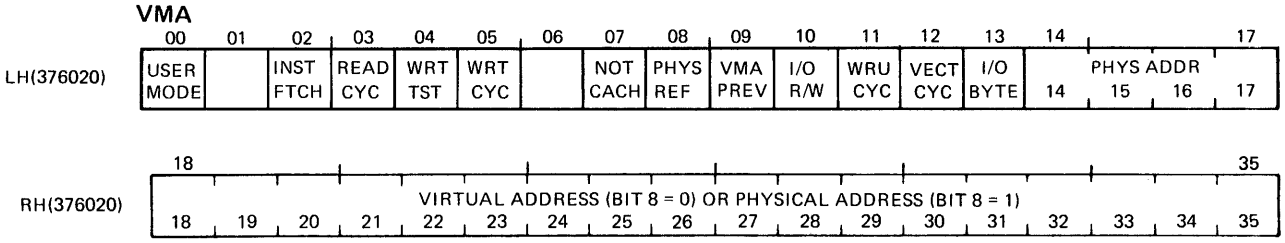


BIT(S)	FUNCTION
4	WRITE REFERENCE BIT FROM PAGE MAP
5	PI CYCLE
6	LOOK IN CACHE BIT FROM PAGE MAP
18-35	PAGE FAIL CODE

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

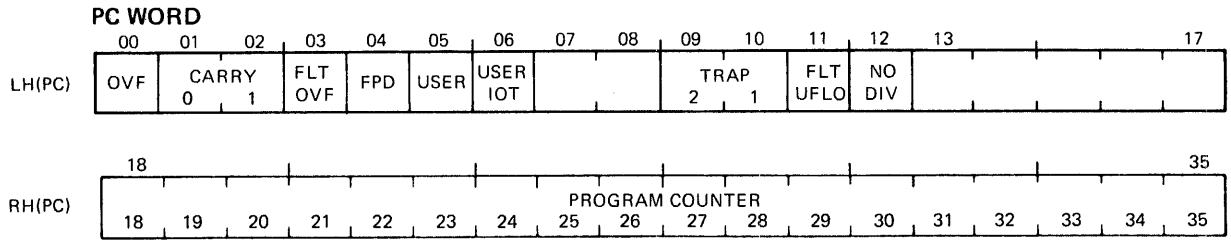
Figure 4-22 Microcode Flags



BIT(S)	FUNCTION
0	USER MODE
2	INSTRUCTION FETCH
3	READ CYCLE
4	WRITE TEST
5	WRITE CYCLE
7	DO NOT LOOK IN CACHE
8	PHYSICAL REFERENCE
9	VMA PREVIOUS
10	I/O READ OR WRITE
11	WRU CYCLE
12	VECTOR CYCLE
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

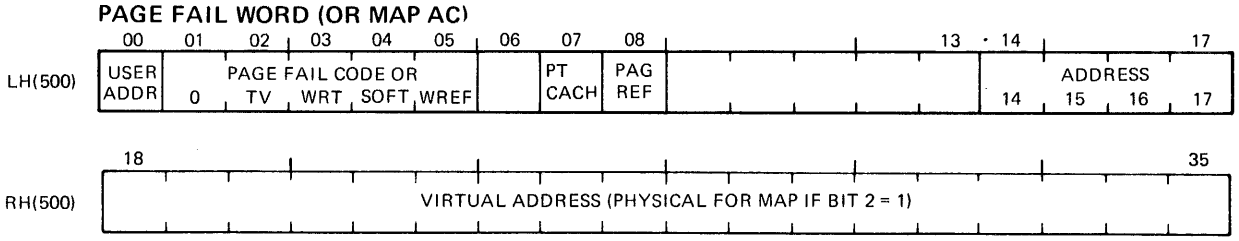
Figure 4-23 VMA



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 4-24 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)
	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 4-25 Page Fail Word

4.13 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, which 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 DZ11 asychronous communications controllers.

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 also operates in more than one mode as explained in Paragraph 4.14. These KLINIK line modes should not be confused with the console (CTY and user) modes. 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 provided the front panel LOCK switch is not on.

The console program initializes to CTY mode at power-up. When in CTY mode, if the operator starts or continues KS10 program execution (ST or CO commands) or enters a "control-Z", the console program switches 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.

The CTY mode command prompt consists of 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 4-3. Error printouts are listed in Table 4-4. Other messages are listed in Table 4-5.

Table 4-3 Console Mode Commands

Command	Description
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. **Control No. — Register Address (Octal) — 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).
LR xx	Set 8080 register address xx.
	NOTE **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.

* An asterisk (*) indicates that the CPU clock must be stopped in order to execute the command.

Table 4-3 Console Mode Commands (Cont)

Command	Description
Deposit Commands (Cont)	
* 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.
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.
DR xx	Deposit xx into 8080 register. Address previously loaded by LR command.
Examine Commands	
EB	Examine KS10 bus. Prints contents of console registers 100–103 and 300–303 (octal).
* EC	Examine contents of CRAM 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.

* An asterisk (*) indicates that the CPU clock must be stopped in order to execute the command.

Table 4-3 Console Mode Commands (Cont)

Command	Description
Examine Commands (Cont)	
EN	Examine contents of next (KS10, 8080, I/O) address.
ER	Examine contents of 8080 register. Address previously loaded by LR command.
ER xx	Examine contents of 8080 register address xx.
Start/Stop Clock Commands	
CH	Halt CPU clock.
* CP	Pulse CPU clock.
* CP xx	Pulse CPU clock xx times.
* CS	Start CPU clock.
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	Shutdown command. Deposits nonzero data into KS10 memory location 30 to allow orderly shutdown of the monitor.
SI	Single instruct. Executes next KS10 instruction.
ST xx	Start KS10 program at address xx. Console program enters user mode.

* An asterisk (*) indicates that the CPU clock must be stopped in order to execute the command.

Table 4-3 Console Mode Commands (Cont)

Command	Description
Select Device Commands	
DS	Select disk for bootstrap or microcode verification. Console program asks for UBA number (default = 1), RH11 base address (default = 776700 octal), 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.
MS	Select tape for bootstrap or for microcode verification. Console program asks for UBA number (default = 3), RH11 base address (default = 772440 octal), tape unit number (default = 0), tape density (default = 1600 bits/in), and slave number (default = 0) as follows: >>UBA? 3 <CR> >>RHBASE? 772440 <CR> >>TCU? 0 <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	
BC	Check the KS10 boot path.
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 (octal). The BT command is performed automatically 30 seconds after power-up. Also, if the boot fails, it is automatically retried every 15 seconds thereafter. 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.
LB	Load the monitor boot program from the disk selected last. Does not load microcode. Program must be started at 1000 (octal).
LB 1	Same as LB command except that diagnostic boot program (not monitor boot program) is loaded. Program must be started at 1000 (octal).

Table 4-3 Console Mode Commands (Cont)

Command	Description
Boot Commands (Cont)	
MB	Load the monitor boot program from the tape selected last. Does not load microcode. Program must be started at 1000 (octal).
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 (octal).
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.
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 Meaning** 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.

* An asterisk (*) indicates that the CPU clock must be stopped in order to execute the command.

Table 4-3 Console Mode Commands (Cont)

Command	Description
Enable/Disable Commands (Cont)	
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.
	NOTE **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 Read Function** 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
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.

* An asterisk (*) indicates that the CPU clock must be stopped in order to execute the command.

Table 4-3 Console Mode Commands (Cont)

Command	Description
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.
control-\\	Enter mode 3 (KLINIK line).
RUB-OUT	Delete last character.
	NOTES **1. More than one command may be entered on a line (separated by commas) and executed as a command string.** **2. 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 4-4 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 Guide* (EK-OKS10-MG) 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 Guide* for definition of error code xx.)
?BUS	Bad KS10 bus. (All bus lines not 0 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.)
?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) or 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.)

Table 4-4 8080 Console Error Messages (Cont)

Message	Meaning
?NR-SCE	Nonrecoverable CRAM error. This message is followed by 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 (octal)
?PWL	Password length error. (Password is 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 4-5 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.)

Table 4-5 Other 8080 Console Messages (Cont)

Message	Meaning
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.

The console program reads and prints at the CTY the contents of certain 8080 registers in response to the EB examine bus (EB) command or a system parity error command (?PAR ERR). The EB command prints registers 100–103 (octal) and 300–303 (octal) in addition to the bus data registers 00–03. Registers 100, 303, and 103 (octal) are printed when the system parity error is detected. Register bit format is shown in Figure 4-26.

4.14 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 control of the 8080 console program in one of four operating modes. The mode depends 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 4-3.

The following list describes the four KLINIK line operating modes (0–3). Refer to Figure 4-27.

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.

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

2. 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 PW (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.

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 4-3). 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. It may be returned to mode 0 by clearing the password area; that is, by entering a PW command with no argument.

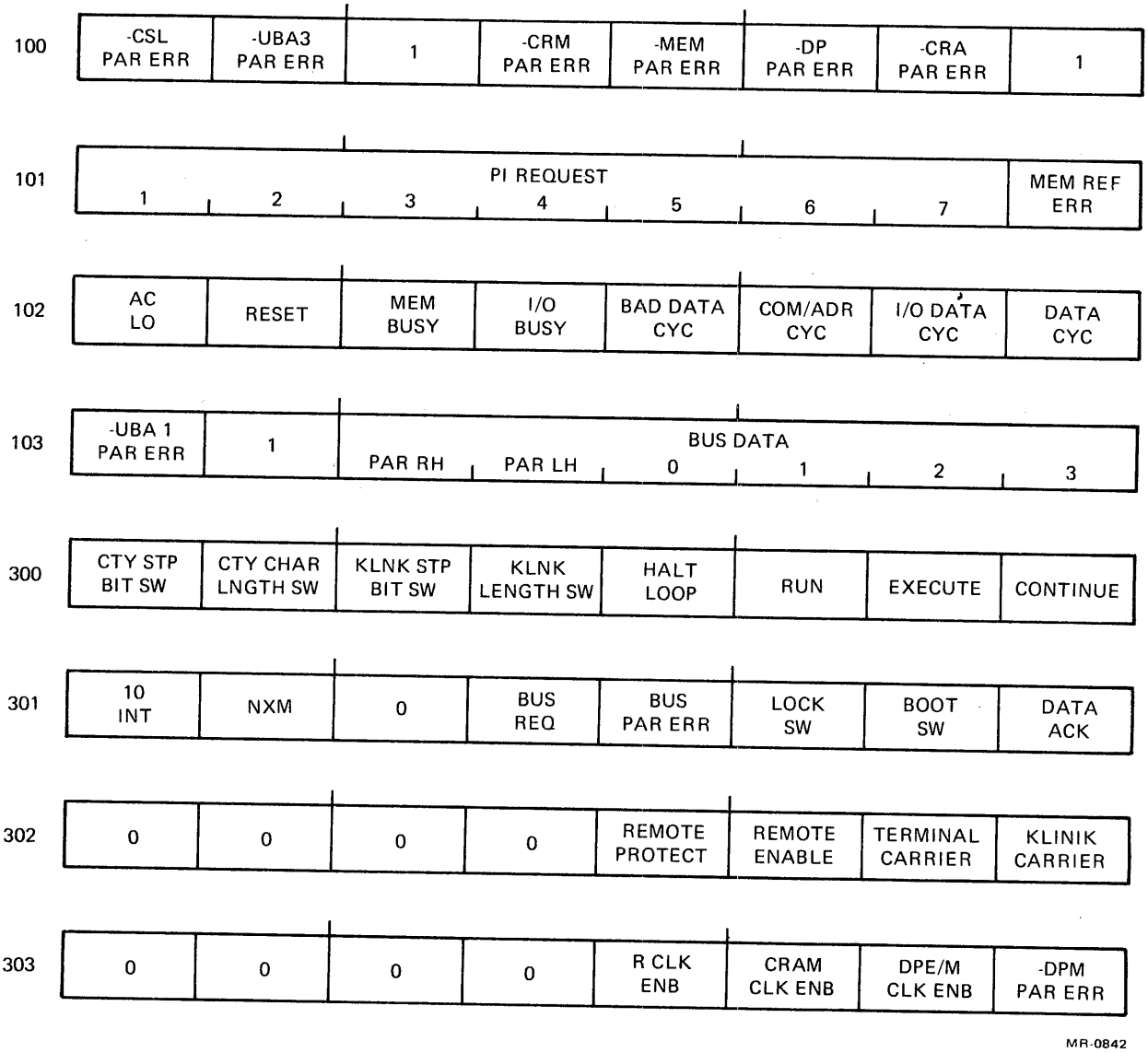


Figure 4-26 Console Status Registers

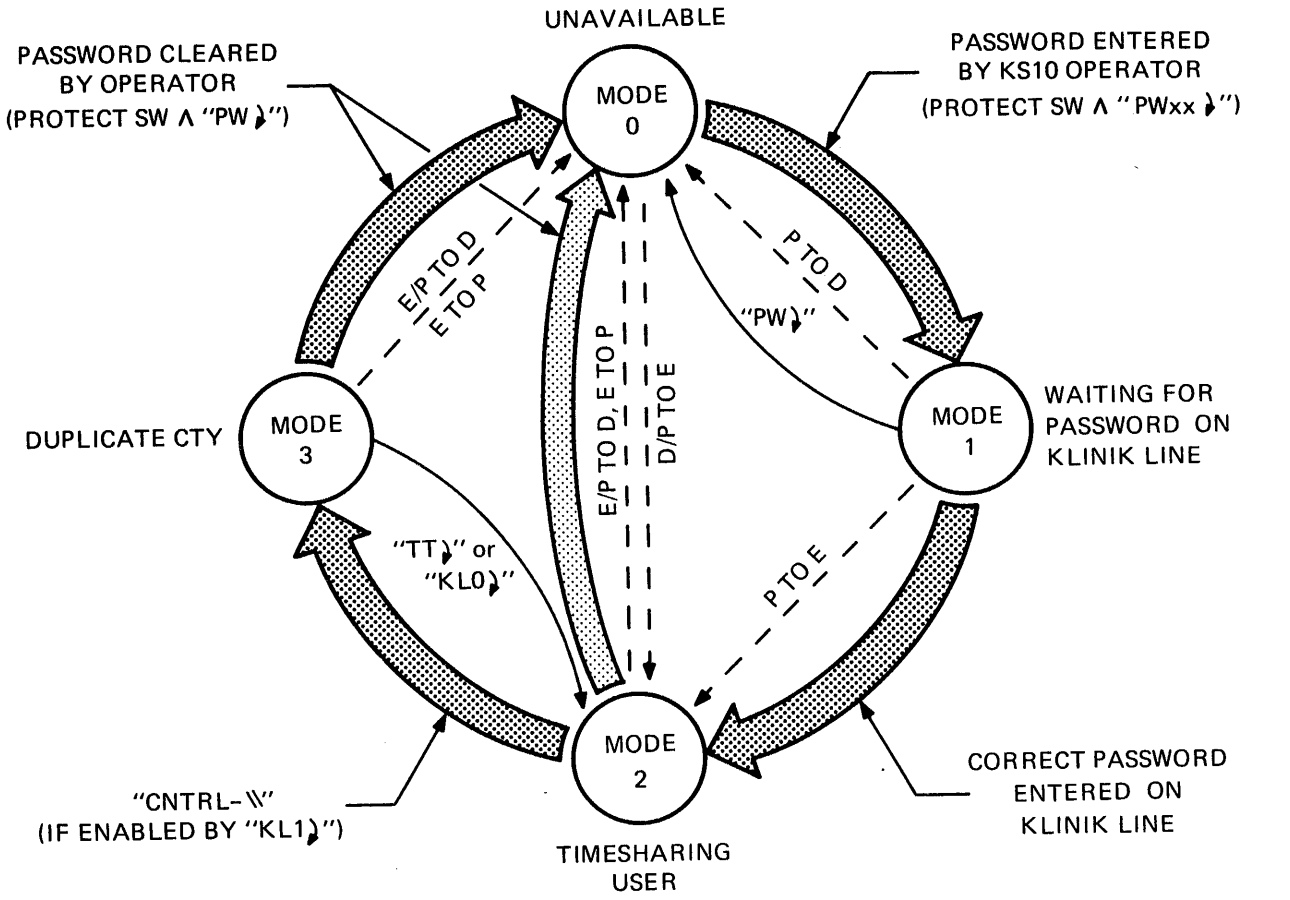


Figure 4-27 KLINIK Line Modes

To summarize KLINIK line operation, the operator, after determining the need for remote diagnosis, calls the DIGITAL Remote Diagnosis Center and gives 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 troubleshoot 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

5.1 INTRODUCTION

A functional block diagram of the KS10 is shown in Figure 5-1. The major KS10 components are the console, CPU, MOS memory, and Unibus adapters (UBAs). They are interconnected by the KS10 (backplane) bus as shown in the block diagram.

The KS10 bus consists of 36 data lines (and 2 parity lines) plus the control lines necessary to perform memory read/write operations, I/O register read/write operations, and priority interrupt (PI) operations. (The data lines are multiplexed in that command/address information is transferred during one bus cycle, and data is transferred during another.) The console, CPU, and UBA all read and write memory over the bus. The console and CPU read and write the I/O registers. PI operations on the bus take place between the CPU and UBA.

5.1.1 Console

The M8616 console module (CSL) performs the following functions.

1. Provides the operator and maintenance interface to the system
2. Generates and controls the system clocks
3. Arbitrates the KS10 bus

The module contains an 8-bit 8080 microprocessor system and an associated CPU control and KS10 bus interface. The 8080 system consists of an 8080A microprocessor, an 8K PROM that stores the console program, a 1K RAM used to store arguments and variables, and two USARTs to connect the operator's console terminal (CTY) and the remote diagnosis (KLINIK) line to the system. The 8080 components interconnect via the 8-bit CSL data bus.

One of the functions of the console program resident in PROM (and executed by the 8080A microprocessor) is to control and service the USARTs. That is, 8-bit characters are transferred from the RAM to the USARTs and the data is serialized for output to the CTY or KLINIK lines. Conversely, serial input data from the CTY or KLINIK lines is assembled by the USARTs into 8-bit characters, each causing an 8080A interrupt. In response to the interrupts, the microprocessor transfers the characters from the USARTs to the RAM for processing.

The main function of the console program is to decode and implement the console commands entered via the USARTs. Many commands are executed as a result of the console initiating one or more KS10 bus functions. The module's KS10 bus interface provides both 36-bit to 8-bit, and 8-bit to 36-bit, data buffering to allow MOS memory and the I/O registers throughout the system to be read or written over the bus.

Other console commands assert control lines that connect directly to the CPU. For example, the start/stop program commands assert RUN, EXECUTE, and CONTINUE lines to control KS10 program execution. Diagnostic functions also assert special control lines, although 18 of the 36 KS10 bus data lines are used to transfer the diagnostic data. The diagnostic functions load and examine microcode in the CPU's microcontroller. They are initiated during system bootstrap and by console commands.

The system clocks generated on the console module are the basic block train (T clock and R clocks). The console also generates the enable levels for the basic processor clocks in the CPU modules. The enable levels may be turned on or off by the console program or by certain hardware parity errors.

The KS10 bus arbitrator is on the console module, but it is not controlled by the console program. The modules connecting to the bus (including the console itself) must first request the use of the bus before initiating a bus function. The arbitrator monitors all requests and grants the bus on a priority basis.

5.1.2 CPU

The CPU consists of the M8622 and M8623 microcontroller modules (CRA and CRM) and the M8620 and M8621 data path modules (DPE and DPM). The microcontroller stores and sequences the KS10 microcode loaded by the console. The data path, in turn, responds to the control signals generated by the sequencing microcode to implement the basic CPU functions.

Aside from the diagnostic logic that is required to load and examine the microcode, the main elements in the microcontroller are the 96-bit × 2K control RAM (CRAM) which stores the microcode, the CRAM register which holds each 96-bit microinstruction as it is read from CRAM and executed, and the skip and dispatch logic (some of which is located on the data path modules) which determines the next CRAM address. Part of the skip and dispatch logic is the 512-location dispatch ROM (DROM). It is addressed by the op-code stored in an instruction register (IR) to provide dispatch and control information specific to the individual system-level instruction being executed.

Also, part of the skip and dispatch logic is a subroutine stack and associated circuitry that stores the current CRAM address and allows for temporarily interrupting the normal sequencing in order to jump to (call) a subroutine elsewhere in the microcode. The stack is a 16-location RAM, thus providing for several levels of subroutine nesting; that is, the calling of one subroutine from another.

The basic element in the data path is the 2901 4-bit processor slice. A total of 10 2901s are used in parallel to provide a 40-bit processing element that performs all of the primary arithmetic and logic functions of the CPU. The 2901s contain an ALU (AD), Q register, input and output control logic, and a 16-word RAM. The RAM contains working locations and constants plus several of the main CPU control registers (PC, AR/ARX, EBR, UBR, etc.).

Another primary element in the data path is the RAM file. It is a 28-bit (26 data bits plus 2 parity bits) × 1K RAM that contains the CPU's 8 blocks of 16 ACs, 384 words of workspace, and the 512-word cache.

The data path also contains the 10-bit logic which performs computations on exponents, counts steps in shift and arithmetic operations, and performs byte manipulation. The 10-bit logic includes the SCAD, SC, and FE registers.

The major data path components interfacing directly to the KS10 bus are the VMA register which holds the virtual memory address (and I/O address) associated with KS10 bus operations, and the hardware page table (a 16-bit × 512 RAM) that stores the virtual page number of the memory address.

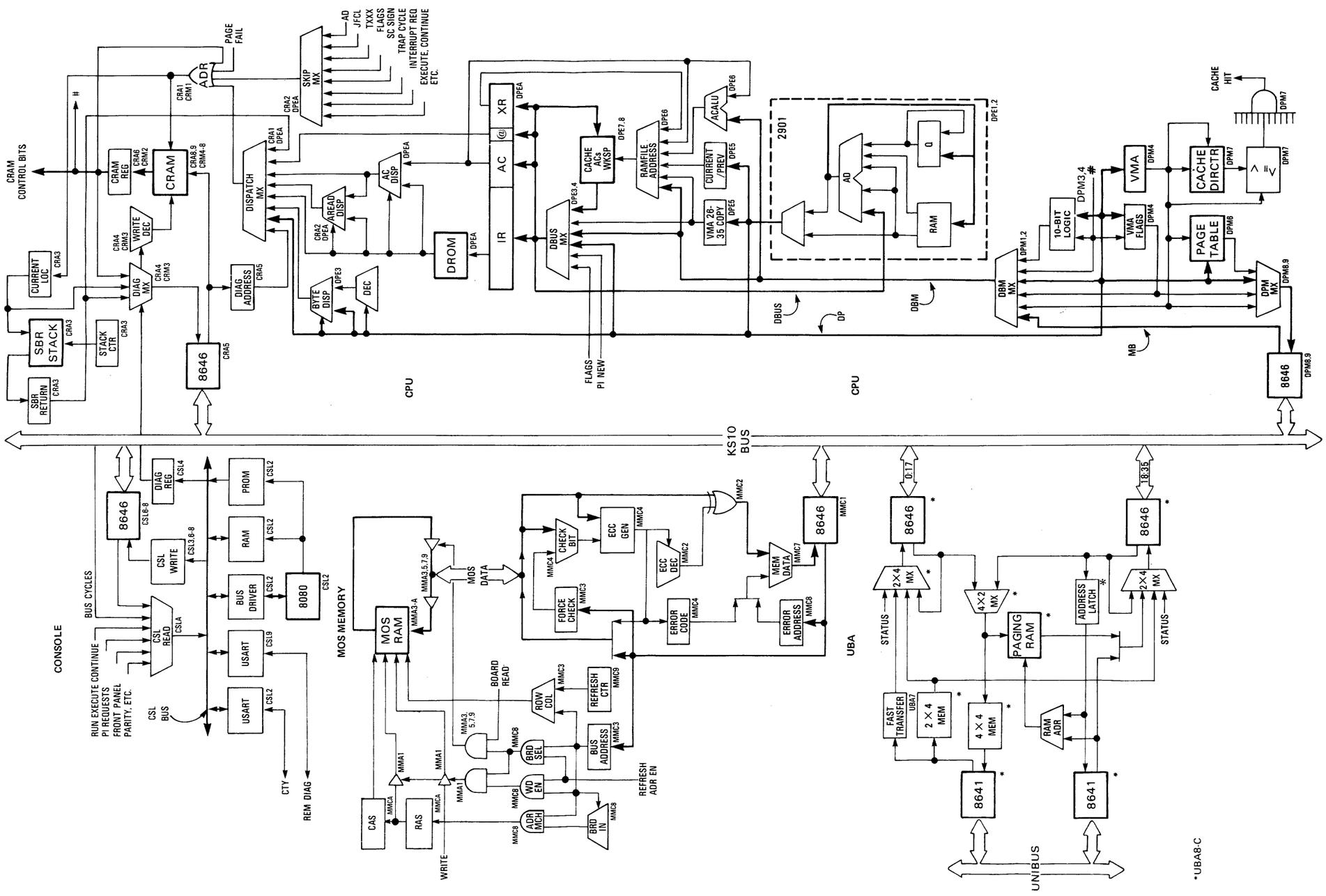


Figure 5-1 KS10 Functional Block Diagram

The flow of information through the data path is over three data buses: the DBus, DP, and DBM. The DBus supplies data to the 2901 arithmetic unit, the RAM file, and the IR. The DBus receives data via the DBus mixer from the RAM file, PC flags, new PI level, VMA 27-35, and the other two data buses (DP and DBM).

By way of DP, the 2901 outputs (which are the only source of data for DP) are available to the DBus, the 10-bit logic, and the various elements in the KS10 bus interface. This includes the KS10 bus transceivers, which makes DP data available for writing to memory and I/O devices over the KS10 bus.

DBM supplies data to the DBus and RAM file. It receives data from the 10-bit logic, the magic number field in the current microinstruction (loaded in the CRAM register), DP and DP swapped, the VMA, and the memory buffer (MB) which is contained in the KS10 bus transceivers.

Figure 5-2 shows the data path in greater detail. Every word fetched from storage goes to the 2901 arithmetic unit, but it is also placed in the cache part of the RAM file. As a result, the next time the word is referenced, it is available directly to the DBus from the cache without requiring a KS10 bus memory read operation. If a word fetched from storage (memory or cache) is an instruction, its left half is loaded into the IR where it is available for execution by the microcode.

The result of an operation may go from the arithmetic unit via DP and the DBus to an accumulator in the RAM file. But to write a word in memory requires that the arithmetic unit first supply a command/address for transmission on the KS10 bus. It then supplies the memory write data. As for words read from storage, words sent to storage are also written into the cache so that they are readily available for subsequent memory read references. Words are deposited in cache via the DBus. The data flow during KS10 bus I/O functions is similar to that for memory functions, except that data is not placed in cache.

5.1.3 MOS Memory

The primary storage for the KS10 is a 128K to 512K MOS memory. It consists of from 2 to 8 M8629 MOS array boards (MMA) and an M8618 memory controller module (MMC). Each array module contains 172 1-bit × 16K MOS chips to store 64K 43-bit words. A word contains 36 data bits plus 7 check bits. Only the memory controller module connects to the KS10 bus, thus providing the interface to memory for the rest of the system. The CPU, console, and UBA all read and write MOS memory over the KS10 bus. In addition, the UBA may gain a read-pause-write access during NPR data transfers.

The memory subsystem has two basic paths for information flow: an address path and a data path. When a command/address is transmitted on the KS10 bus to initiate a memory access, the bus address is stored in the memory controller and then used to generate the necessary MOS array select signals during the ensuing operation. The select signals (that is, board select, row column address, etc.), together with 43 data lines, make up the MOS data bus interface that connects the memory controller to all the array modules.

The 43 data lines on the MOS data bus interface are part of the basic data path for the memory system. During a memory write, the 36-bit data word to be written into the MOS array is transmitted on the KS10 bus after transmission of the command/address. This data is gated from the KS10 bus directly to the MOS data bus interface and then written into the array along with the seven check bits. The check bits, which allow 1-bit error correction and 2-bit error detection, are generated by an error correction code (ECC) generator in conjunction with a check bit multiplexer.

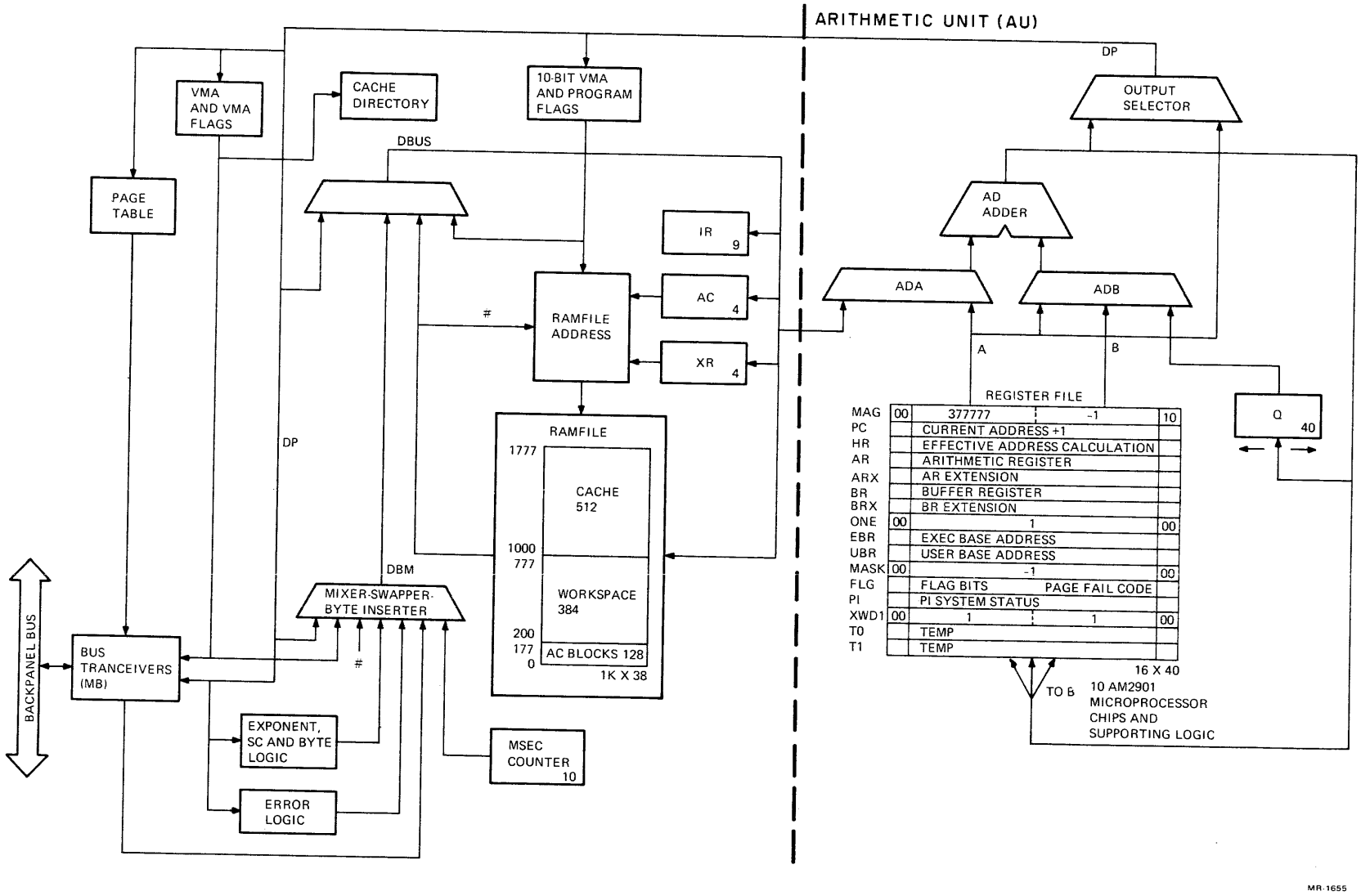


Figure 5-2 Processor Data Flow

If the command/address received by the controller initiates a memory read, the 36 bits of read data and the seven check bits are read from the MOS array and asserted on the MOS data bus interface. The data bits are also transmitted on the KS10 bus. In addition, the data bits and (this time) the check bits are gated to the ECC generator. If a non-zero ECC is generated, it indicates an error. If the error is in only one bit, the ECC value is decoded to produce a signal that complements the failing bit as it is being transmitted on the KS10 bus. A 2-bit error is flagged but not corrected.

5.1.4 Unibus Adapter

An M8619 Unibus adapter module (UBA) provides the interface between the KS10 bus and the KS10 I/O devices connected to a single Unibus. Two UBAs, and thus two Unibus connections, are standard for current KS10 configurations. A high-speed disk connects to one UBA; all other I/O devices connect to the other. A UBA performs the following functions.

1. Arbitrates the associated Unibus
2. Allows NPR transfers of Unibus data directly to/from KS10 memory
3. Allows access to the I/O registers in the Unibus devices
4. Transfers vector addresses to the KS10 CPU following Unibus device interrupts

Because the Unibus may be used by the UBA and all the connecting I/O devices, an arbitrator is required to determine which device obtains control of the bus. The UBA arbitrator grants the Unibus to a device for NPR and vector transfers. The UBA controls the bus for I/O register read/write operations.

Similar to MOS memory, the UBA has an address path and a data path for information flow. The main component in the address path is the paging RAM. During NPR transfers, which provide direct access to KS10 memory without intervention by the KS10 CPU, the paging RAM converts the address transmitted on the 18 Unibus address lines into a 20-bit KS10 memory address. The UBA then transmits this address to the MOS memory (as part of a command/address) and initiates either a memory read or a memory write (or read-pause-write) depending on the direction of the NPR transfer.

The major components in the data path are two groups of 4 × 4 memories used to buffer the Unibus data. One group is used during NPR read (from memory) operations. Following transmission of a KS10 bus command/address by the UBA, the memory read data transmitted on the bus by the memory is first latched by the UBA, and then the appropriate Unibus word (16 or 18 bits) or byte (8 bits) within the 36-bit memory word is stored in the 4 × 4 buffers. (The position of the Unibus data within the KS10 memory word is specified by the two low-order bits of the Unibus address.) The buffer outputs transmit the Unibus data directly to the device over the 18 Unibus data lines.

The other group of 4 × 4 memories act as a buffer for Unibus data during the NPR write (to memory) operations. The Unibus word or byte is loaded directly into the buffers and transmitted on the KS10 lines as memory write data following transmission of the command/address. Mixers in the data path position the Unibus data within the KS10 memory word as required. The memory operation initiated by the UBA is a memory write operation if the Unibus data is the first data loaded in the KS10 memory location. Otherwise, a read-pause-write access is initiated. That is, the Unibus data already written in the memory word is read and recirculated by the UBA's data path mixers; the recirculated data is then written together with the new Unibus data into the same KS10 memory address.

The 4 × 4 memories in the data path not only store NPR data; they also store control/status information transferred to/from the Unibus devices as a result of KS10 bus I/O register write/read operations. The address path for I/O transfers differs from that for NPR transfers, however. The I/O address (part of the command/address received by the UBA to initiate the operation) is simply latched and then transmitted on the Unibus address lines.

Interrupt vector addresses transmitted on the Unibus are also buffered in the 4 × 4 memories. A vector is stored in response to the Unibus interrupt (bus request) signals (BR levels) that first causes a KS10 CPU interrupt. (The BR levels assert interrupt lines on the KS10 bus to interrupt the CPU.) The CPU follows by initiating a vector read operation on the KS10 bus, the command/address specifying the interrupting UBA. The UBA then loads the Unibus device vector and transmits it on the KS10 bus for collection by the CPU.

Because more than one UBA may assert the same KS10 bus interrupt line, the vector read operation is preceded by a KS10 bus operation that determines which UBAs are interrupting on the PI channel number being served. (A UBA signals that it is interrupting on the specified channel by asserting the assigned KS10 bus data line corresponding to its controller number.) If both UBAs are interrupting on the same PI channel), the CPU reads the vector from UBA1 which has the highest priority.

5.2 SYSTEM TIMING

The console (CSL) module generates the basic system clocks together with the enable levels necessary to control the CPU clock. (The CPU clock defines the basic processor cycle as described in Paragraph 5.4.) Clocks and enable levels are distributed via the KS10 backplane as shown in Figure 5-3.

NOTE

The special timing signals asserted by the CSL when loading and reading microcode (shown as dotted lines in Figure 5-3) are not described here. Refer to Paragraph 5.3.8.

5.2.1 Basic Clocks

The basic clocks are the 6.66 MHz T and R (transmit and receive) clocks. Both clocks are normally free-running and have a period of 150 ns. Their timing relationship is shown in Figure 5-4.

T clocks are distributed (in parallel) to all modules on the KS10 backplane. The leading edge of a T clock is used to change the data transmitted on the KS10 (internal) bus. The principal function of T clock, however, is to generate other clocks within the various modules. Some of these clocks are generated only when particular conditions are met. For example, the CPU clock is generated from T clock only when a CPU clock enable is true.

R clock, unlike T clock, is not used by all the modules on the KS10 backplane. R clocks are used principally to gate information off the KS10 bus, and thus they are distributed (in parallel) to only those modules that normally receive bus data.

A 26.666 MHz crystal oscillator and a 2-flip-flop frequency divider are used on the CSL module to generate the two 6.66 MHz T and R clock trains. The frequency divider outputs, CSL1 MASTER T CLK and CSL1 MASTER R CLK, are passed through adjustable delays to generate the parallel clock outputs. The delays, one per set of 4 output gates, are set to minimize the skew that occurs because of differences in IC propagation delays.

NOTE

The clock skew adjustments on the M8616 CSL module are factory adjustments and should not be attempted in the field.

Although T and R clocks are normally free-running, the 8080 console program may stop and start the clocks by first setting CSL6 MAINT ENB to disconnect the crystal oscillator, and then by setting and clearing CSL6 CLK 0 to clock the frequency divider. This allows the clocks to be single-stepped during stimulus-response (STIRS) operations.

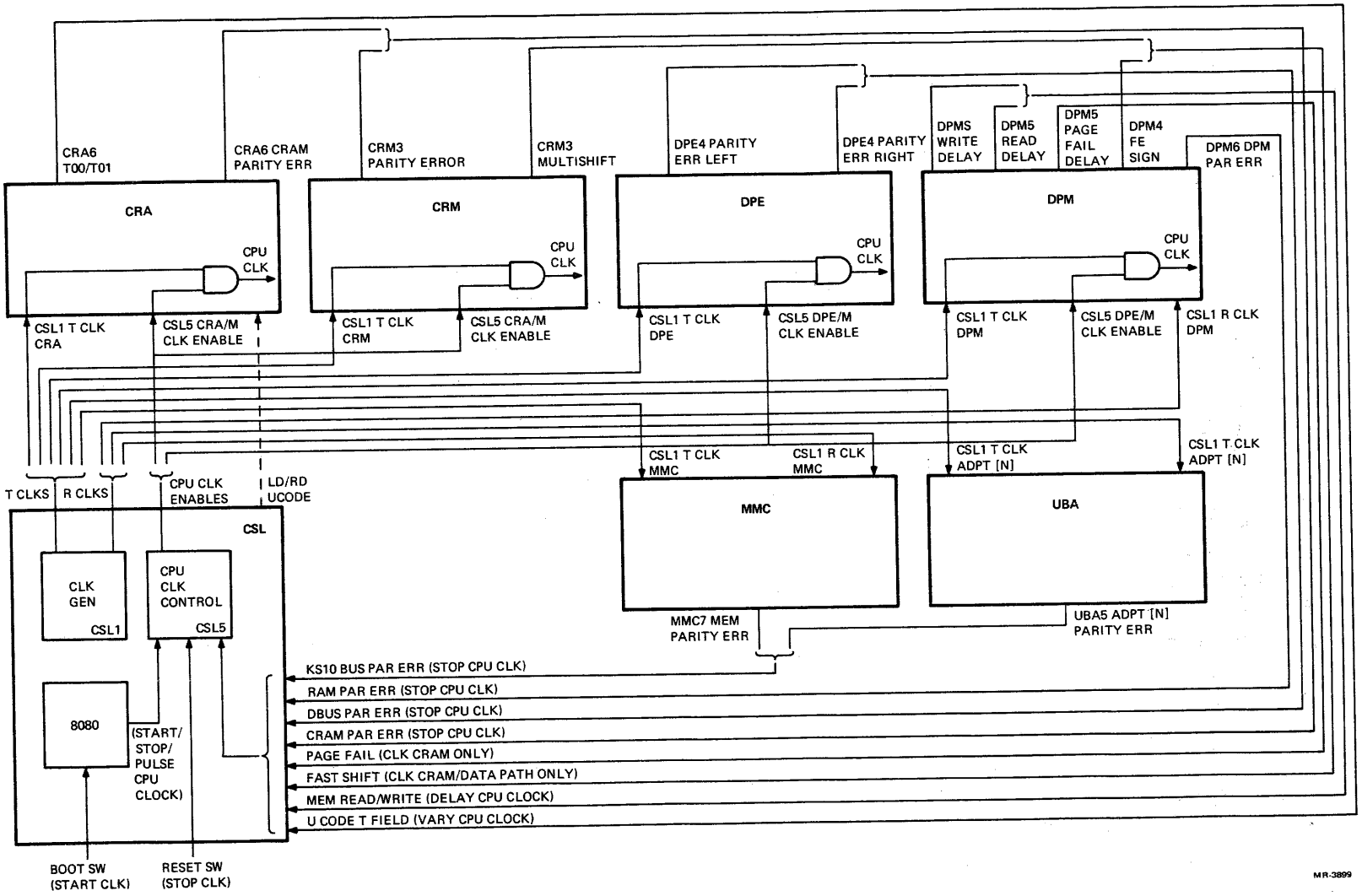


Figure 5-3 Clock Distribution and CPU Clock Control

5.2.2 CPU Clock Control

The console module (CSL) controls the assertion of CPU clocks by generating enable levels that are ANDed with T clock in the four CPU modules. There are two enable levels, one for the CRA and CRM microprocessor modules (CSL5 CRA/M CLK ENABLE), and one for the DPE andd DPM data path modules (CSL5 DPE/M CLK ENABLE). The enables are normally 150 ns in duration, each 150 ns assertion passing a single T clock to produce the CPU clock train. The trailing edge of a CPU clock terminates one processor cycle and begins the next. (Timing is shown in Figure 5-4.) By controlling the enables, the console module may:

- Start the CPU clock
- Stop the CPU clock
- Pulse the CPU clock
- Vary the CPU clock
- Delay the CPU clock
- Clock the microprocessor while inhibiting the data path
- Clock the data path while inhibiting the microprocessor.

The enables and thus the CPU clock are controlled mainly by CSL5 ENABLE. This flip-flop asserts both CRA/M CLK ENABLE and DPE/M CLK ENABLE except for the special cases when one half of the machine (either the microprocessor or the data path) is clocked but the other half is not. The ENABLE flip-flop operates in the following manner.

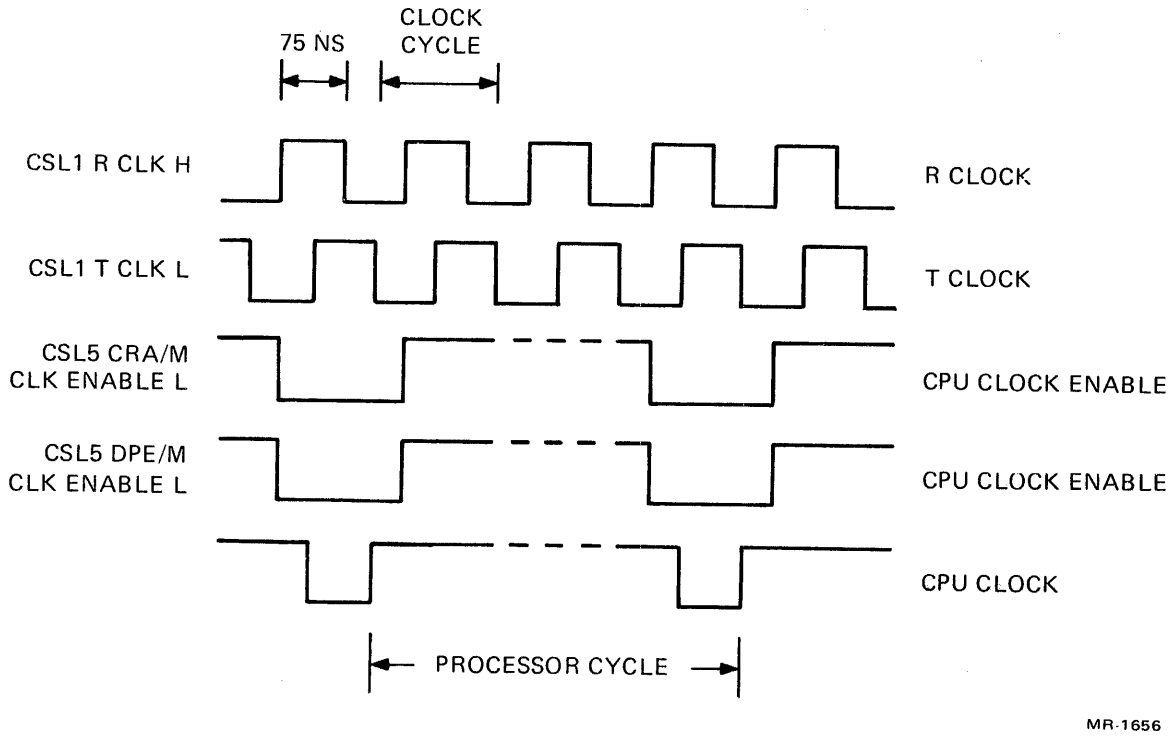


Figure 5-4 System Clocks

To start the CPU clock, the ENABLE flip-flop is set by CSL5 CLK RUN, which in turn is controlled by the 8080 console program. CLK RUN is set by any of the following:

1. The CS (start CPU clock) console command
2. The SM (start microcode) console command
3. All of the bootstrap operations, whether invoked by console command or the BOOT switch on the front panel.

With CLK RUN = 1, the ENABLE flip-flop is continuously set and cleared by R clock to give the series of 150 ns pulses that generate the CPU clock. Clearing CLK RUN then stops the CPU clock. It is cleared by any of the following:

1. The RESET switch on the front panel
2. The MR (master reset) console command
3. The CH (halt CPU clock) console command
4. The CE (cache enable/disable) console command.

The CPU clock is also stopped by various error conditions throughout the system. CSL3 PE is ANDed with CLK RUN to stop the assertion of the ENABLE flip-flop whenever any of the following conditions occur.

1. A KS10 bus parity error is detected by a Unibus adapter (UBA module), the memory controller (MMC module), or the console itself (CSL module).
2. A DBus parity error is detected in the CPU data path (DPE module).
3. A CRAM parity error is detected in the CPU microprocessor (CRA or CRM module).
4. A RAM (cache directory or hardware pager) parity error is detected in the CPU data path (DPM module).

NOTE

The stopping of the CPU clock by one or more of the above error conditions may be disabled by means of the PE console command.

Another function of the ENABLE flip-flop is to allow the CPU clock to be single-stepped. The console program first sets CSL5 SINGLE CLK to allow ENABLE to set. ENABLE then clears SINGLE CLK, resulting in a single CPU clock being produced. SINGLE CLK is set by any of the following:

1. The CP (pulse CPU clock) console command
2. The PM (pulse microcode) console command
3. The TR (trace) console command
4. The EC (examine CRAM register) console command when an argument is specified
5. The MK and UM (mark and unmark microcode) console commands.

In addition to starting and stopping the CPU clock, the enable flip-flop controls the period between outputs, thus changing the length of the processor cycle. This period is normally a function of the current microinstruction; that is, a CPU clock causes (among other things) a microinstruction to be loaded in the CRAM register and the value of this microinstruction's T field determines the time before the next CPU clock. To accomplish this, the T field (CRA6 T00 and T01) connects to a mixer,

and a 2-stage counter sequences the mixer's select levels to set CSL5 T COUNT DONE after the number of T clocks specified by the value of T. The T COUNT DONE signal, which is ANDed with the other ENABLE flip-flop inputs (for example, CLK RUN), then allows the next CPU clock to be generated. A value of T = 00 results in a CPU clock period equal to two T clocks. By changing T, the microcode can increase the period by one to three T clocks whenever additional time is needed, such as when getting a word from the workspace or an accumulator other than AC.

T Field	CPU Clock Period
00	300 ns
01	450 ns
10	600 ns
11	750 ns

The ENABLE flip-flop also controls the CPU clock period by simply delaying the clock. This occurs whenever the CPU must temporarily halt processing because a memory read/write data transfer has not completed. Either CSL5 READ DLY or WRT DLY, ANDed with T COUNT DONE and the other ENABLE flip-flop inputs, temporarily stops the CPU clock until memory read data has been received by the CPU (DPM5 READ DELAY ∧ DATA CYCLE asserted on KS10 bus) or until memory write data is about to be asserted on the KS10 bus by the CPU (DPM5 WRITE DELAY ∧ BUS GRANT received by the CPU).

As mentioned previously, the ENABLE flip-flop generates both CPU clock enables, except for special cases when the microprocessor or data path is to be clocked independently. One such case is when a page fail occurs. CSL5 PAGE FAIL inhibits the DPE/M CLK ENABLE level (and thus the data path clock), but the CRA/M CLK ENABLE level (and thus the microprocessor clock) is asserted. This allows the microprocessor to start executing page fail microcode. During this interval, CSL5 PAGE FAIL START asserts ENABLE to generate a CRAM clock.

Another special case occurs during fast-shift operations by the CPU (CRM2 MULTI SHIFT = 1). First, only the microprocessor clock is enabled as preparations are made to begin the shift operations. Then, with the FE register preset to the desired shift count (DPM4 FE SIGN = 1), only the data path clock is enabled to fast-shift the data being processed. Also, CSL5 FS is asserted, which holds the enable flip-flop set and causes a CPU clock to be generated at the full T clock rate (the period equals 150 ns).

5.3 KS10 (BACKPLANE) BUS

THE KS10 bus is a synchronous backplane bus internal to the KS10 processor. It 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, UBA1 and UBA3. The bus can accomodate another I/O controller to allow for future expansion.) The KS10 bus performs the following major functions.

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

- Diagnostic Data Transfer – Transfers diagnostic data to/from the microcontroller under control of the console. Data transferred to the microcontroller includes the microcode, which is loaded during system bootstrap.

- 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 also two parity bits associated with the data lines, one for data lines 0–17 and one for data lines 18–35. 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 module connecting to the bus 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 module can initiate a transfer of information over the KS10 bus, it must first request and then be granted the bus to become bus master. There is a bus request line and a corresponding grant line for each requesting 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 module.

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

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

Table 5-1 KS10 Bus Signal Summary

Signal	Description
REQUEST (one per device)	Asserted by the module requesting the bus.
GRANT (one per device)	Asserted by the bus arbitrator when the module requesting the bus has been granted the bus. (Module becomes bus master.)
COM/ADR CYCLE	Asserted by the bus master when transmitting command/address on data lines. Asserted for one bus cycle.
DATA CYCLE	Asserted by the bus master when transmitting memory write data or I/O register write data on the data lines. Asserted by memory controller (slave) when transmitting memory read data on data lines. Asserted for one bus cycle.
BAD DATA CYCLE	Asserted by the memory controller (slave) when transmitting uncorrectable memory read data on the data lines. Asserted for one bus cycle coincident with DATA CYCLE.

Table 5-1 KS10 Bus Signal Summary (Cont)

Signal	Description
I/O DATA CYCLE	Asserted by the bus master or slave when transmitting I/O register read data on the data lines, or by the UBA (slave) when transmitting an interrupt vector on the data lines. Asserted for one bus cycle.
MEM BUSY	Asserted by the memory controller (slave) after receiving memory read, write, or read-pause-write command. Negated when memory is ready to accept another command. This signal disables the bus arbitrator.
I/O BUSY	Always asserted by addressed module after receiving an I/O register write command. Also asserted by addressed module 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 module requests the bus and generates a data cycle to transfer the data at a later time.)
CLR BUSY	Asserted by the CPU (after a time-out) to negate I/O BUSY in UBA after a nonexistent register has been referenced.
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 modules 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 initialize modules connecting to 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).

5.3.1 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 5-6. In the KS10, T CLK ORed with R CLK usually 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. If the TRN ENABLE input is false (high), no data is loaded in the flip-flops and 0s are transmitted on the bus. When the REC LATCH input is true (low), the data currently on the bus and received by the 8646 is stored in the gated latch circuits. The latches remain closed (that is, the received data output pins will not change) until the REC LATCH input goes false. When the REC LATCH input is false, the latches are open and the data currently on the bus is asserted on the received data output pins.

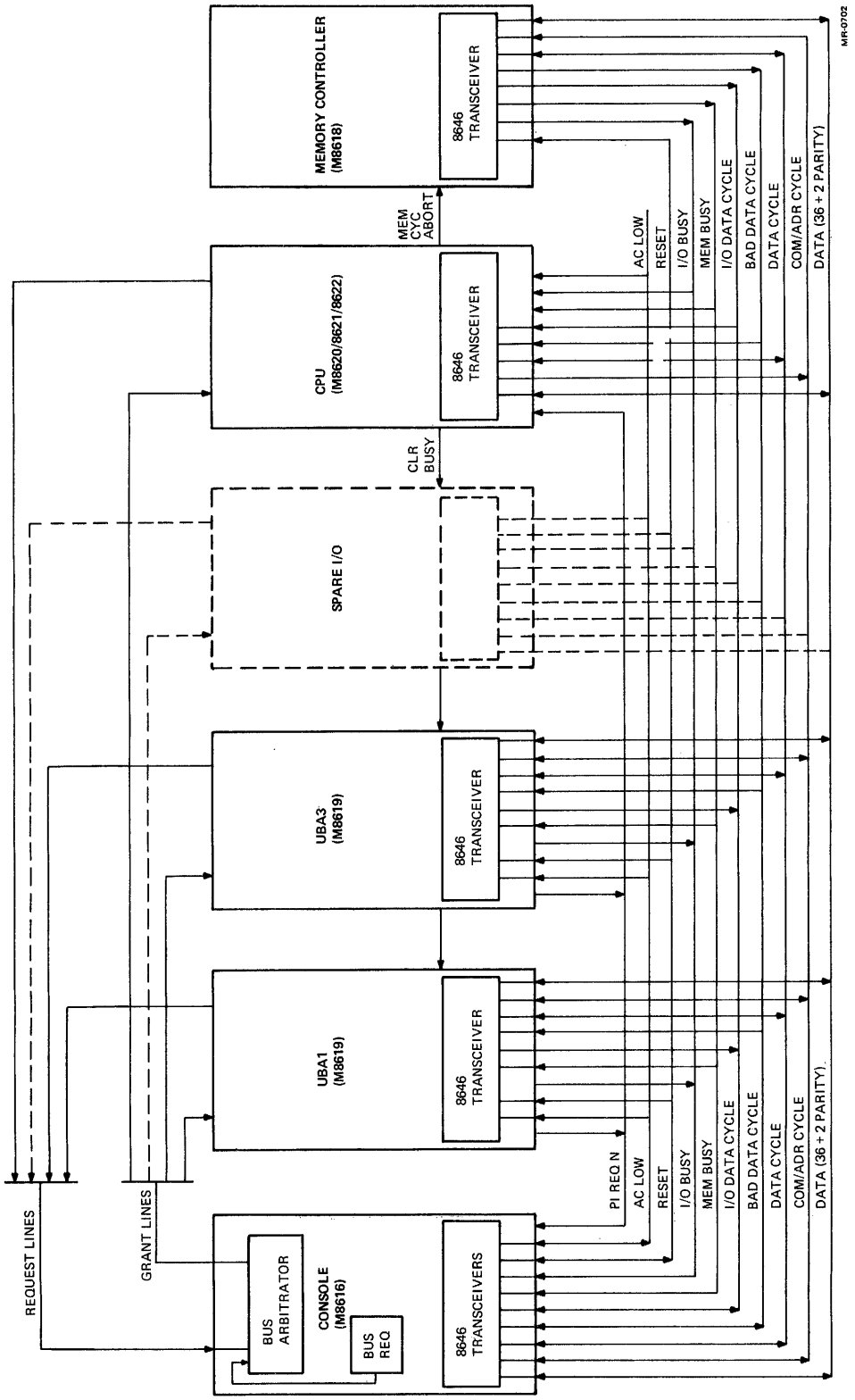


Figure 5-5 KS10 (Backplane) Bus

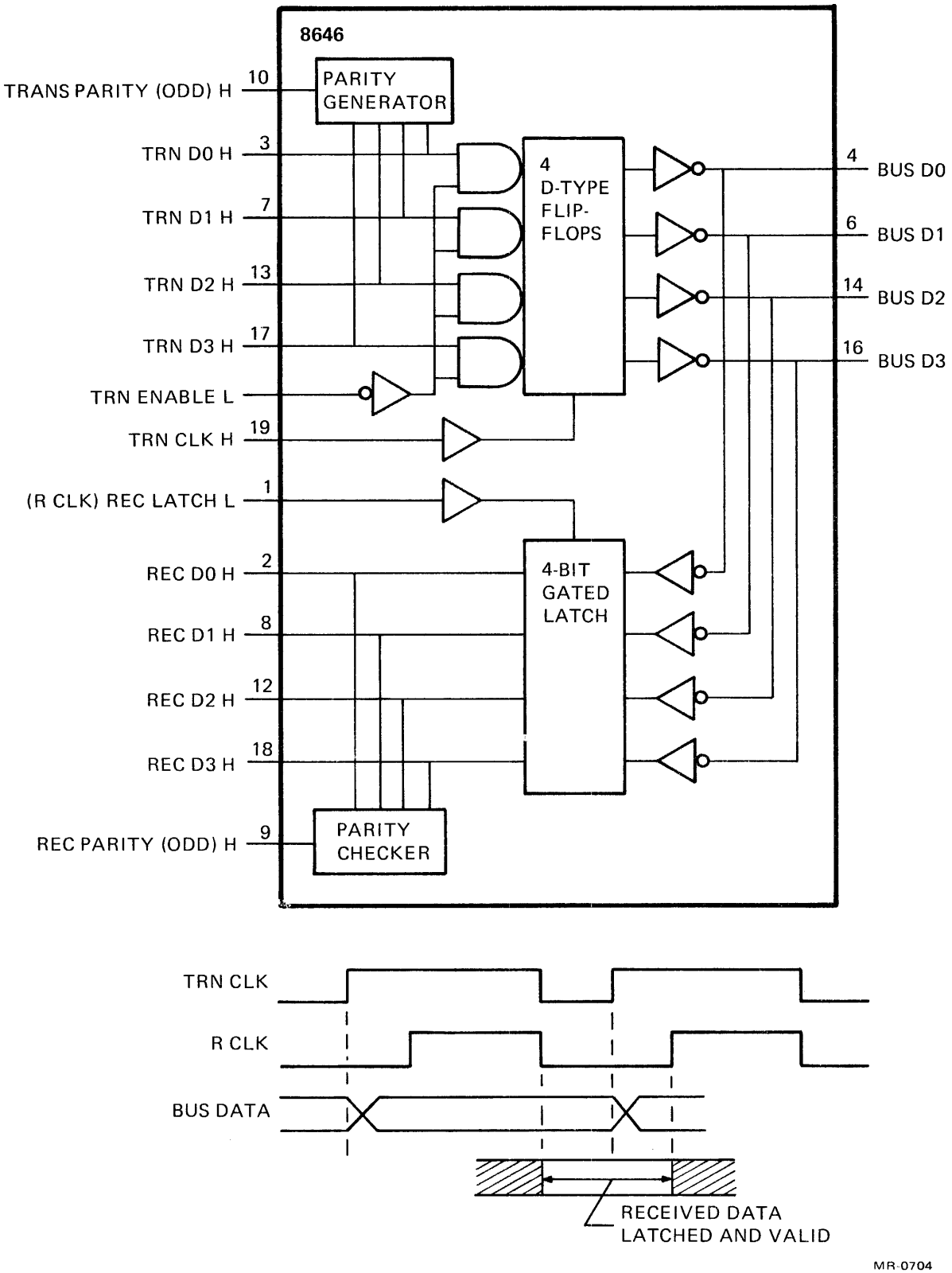


Figure 5-6 8646 Bus Transceiver

Bus transceivers that connect to the KS10 bus data lines utilize the internal parity generator and parity checker. Little additional logic is required 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.

5.3.2 Bus Arbitration

Modules request the use of the bus by asserting a bus REQUEST line at the leading edge of T clock, the start of a bus cycle. Assuming no higher priority requests, the bus arbitrator circuitry (on CSL1) grants the bus by asserting the requesting module's GRANT line at the completion of the current bus operation. If there is no bus operation currently in progress, the GRANT signal will be asserted during the same bus cycle that the bus REQUEST signal is asserted. When GRANT is received, the module negates its REQUEST line at the leading edge of the next T clock and assumes control of the bus as bus master.

When there is more than one bus REQUEST line asserted at once, the bus is granted to the module with the highest priority. Bus priority is determined by a priority encoder circuit whose output is decoded to assert the appropriate GRANT line. Bus priority, highest to lowest, is as follows.

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

NOTE

The memory controller module does not make bus requests. The memory is limited to responding to other bus modules (as a slave) and is never bus master.

After granting the bus, the arbitrator is always disabled (that is, prevented from honoring another request) for one bus cycle. This actually gives the bus master a minimum of two bus cycles, as the bus may be used during the next bus grant procedure assuming another bus REQUEST line is asserted. In other words, the bus grant procedure takes place during the final cycle of any bus operation currently in progress.

Once granted the bus, a bus master may perform a single data cycle (using one of the two alloted bus cycles), not use the bus, perform a diagnostic operation, or (more typically) perform a command address cycle. If a command address cycle is generated, bus signal COM/ADR CYCLE causes the arbitrator to be disabled for an additional bus cycle, thus giving the bus master a minimum of three bus cycles. In addition, if the command/address is initiating a memory operation to a legal address, the arbitrator is disabled during the third cycle by MEM BUSY. This bus signal is asserted by the memory controller module in response to the command/address, and it remains asserted, freezing the arbitrator until the 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 for at least two bus cycles.

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

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

REQUEST/GRANT timing for all three cases is shown in Figure 5-7.

5.3.3 Bus Usage

As stated previously, a module may do the following after it has been granted the bus:

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

Not using the bus after requesting and being granted the bus is equivalent to giving up the bus; another bus request must be made to become bus master. For the current KS10 configuration and barring a malfunction, the only module 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 DPMC MEM CYC ABORT to terminate MOS memory operation.

The only module 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 module 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.

Diagnostic operations are performed only by the console module. All transfers take place to/from the microcontroller in the CPU. Only the data line portion of the bus is used. The various diagnostic operations are discussed in Paragraph 5.3.8.

A module 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 bus devices for each operation. For example, the first entry indicates that the console, CPU, and UBAs all write data into memory.

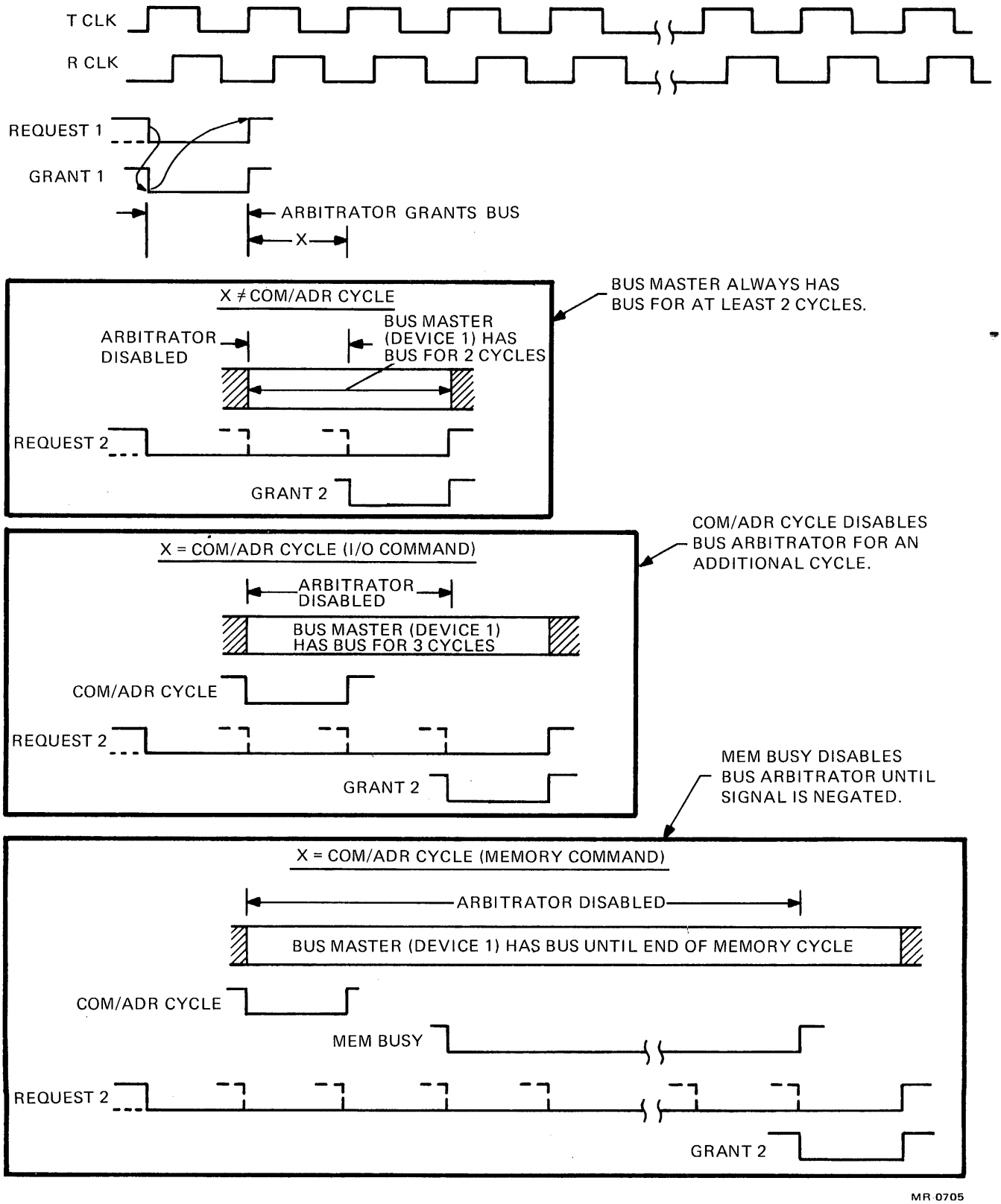


Figure 5-7 Request/Grant, Bus Timing Diagram

Table 5-2 Bus Operations

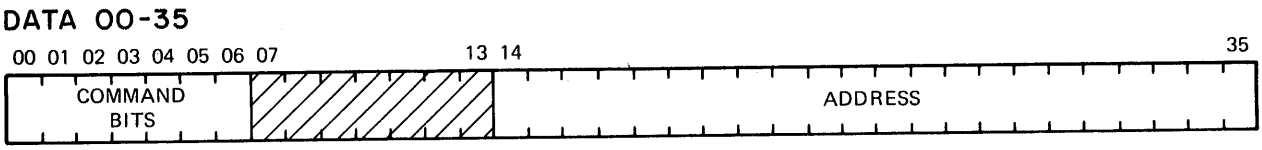
Operation	Initiated By (Master)	Directed To (Slave)
Memory Write	CPU	Memory
	Console	Memory
	UBA	Memory
Memory Read	CPU	Memory
	Console	Memory
	UBA	Memory
Memory Read-Pause-Write	UBA	Memory
I/O Register Write (byte operations directed to UBA only)	CPU	UBA
	CPU	Memory (status register)
	Console	UBA
	Console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

5.3.4 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 as previously described. 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 5-8. Data lines 00–06, 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 5-9.

The seven command bits (bit 03 is not used) specify the nine different bus operations in the following manner. Bit 00 determines whether the operation is a memory data transfer or an I/O data transfer; that is, bit 00 = 0 specifies a memory function and bit 00 = 1 specifies an I/O function. Bits 01 and 02, the read and write bits respectively, act in conjunction with bit 00 to further specify the type of operation. For example, if bit 00 = 0 and bit 01 (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 command bits (00–02).



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 5-8 Basic KS10 Command/Address Format

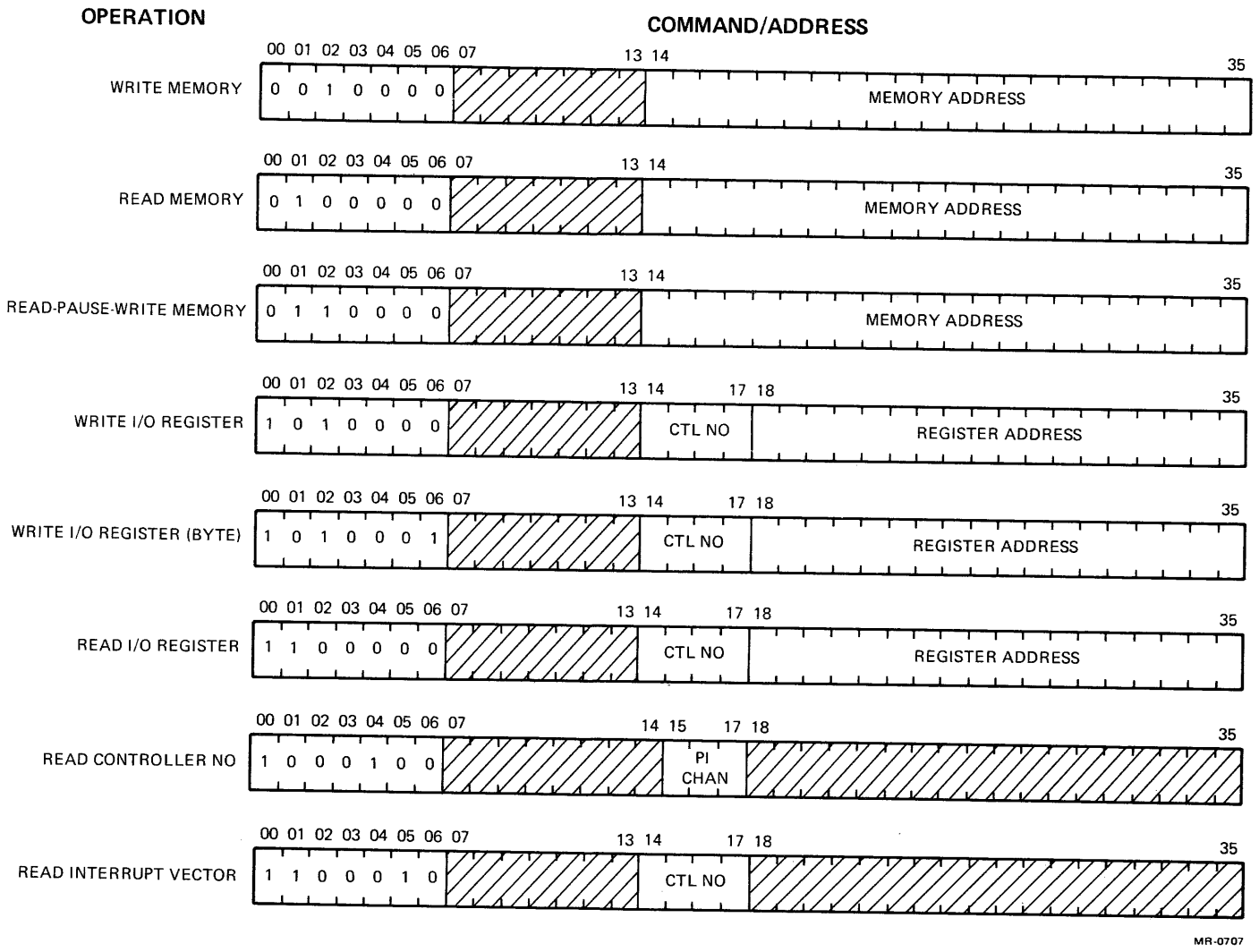


Figure 5-9 Command/Address Bits

Bits 04 and 05 are used to specify the two PI operations performed over the KS10 bus. The CPU asserts one of the bits (bit 04 = 1) to read the interrupting controller number. It asserts the other bit (bit 05 = 1) to read the interrupt vector. Bit 00 = 1 for both PI operations. Bit 01 (read) = 1 for the vector read.

Bit 06, 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 06 is used to specify the transfer mode.

The 22 data lines (14–35) reserved for address information transfer either a memory address (bit 00 = 0) or an I/O address (bit 00 = 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 05 = 1), only the controller number is significant. For the other PI function (bit 04 = 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 Appendix B.

5.3.5 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 00 = 0), the memory address (bits 14–35), and the type of memory operation; that is, a read (bit 01 = 1), a write (bit 02 = 1), or a read-pause-write (bit 01 = 1, bit 02 = 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 module 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 5-10.

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 module 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 5-11.

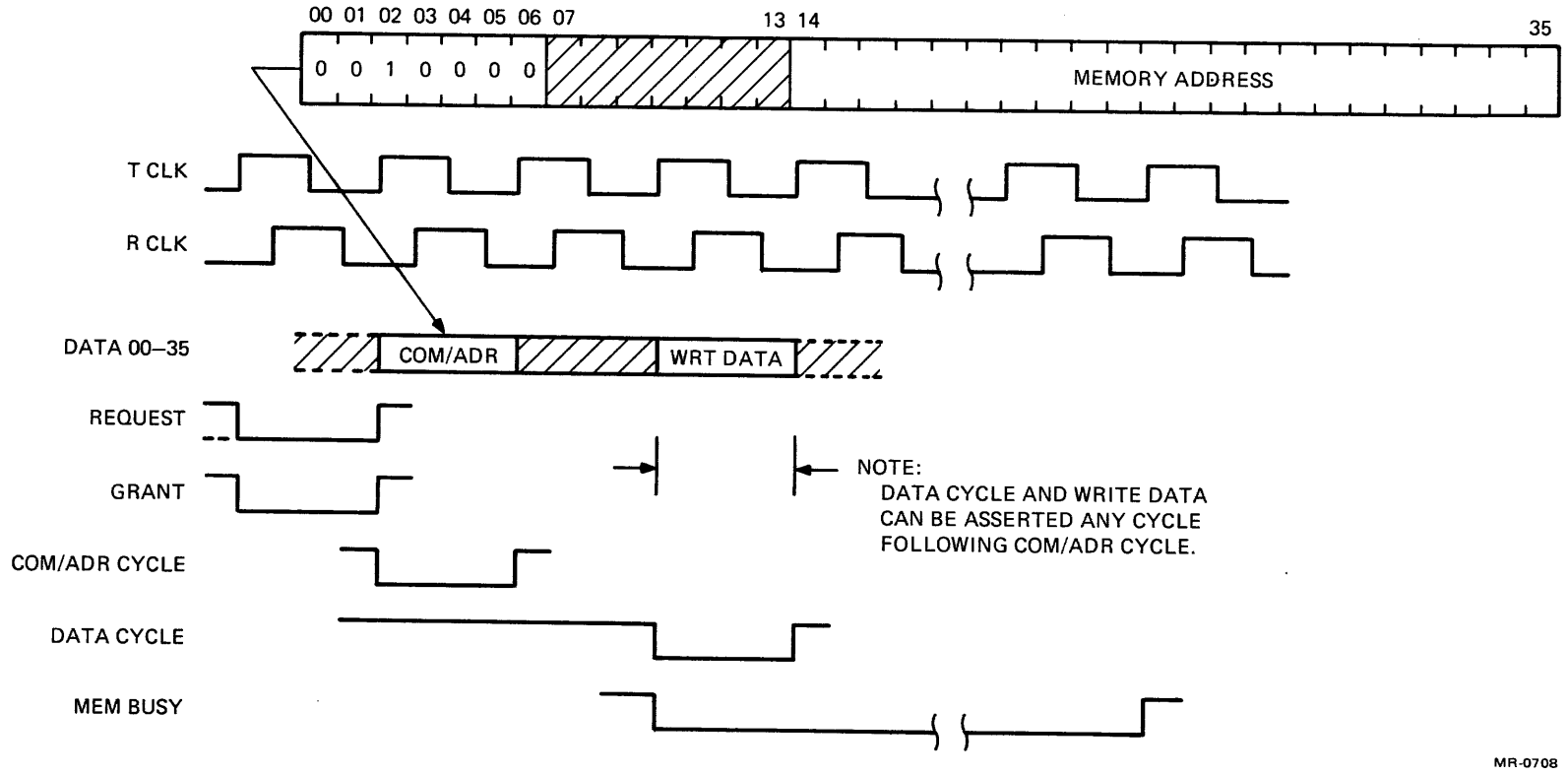


Figure 5-10 Memory Write, Bus Timing Diagram

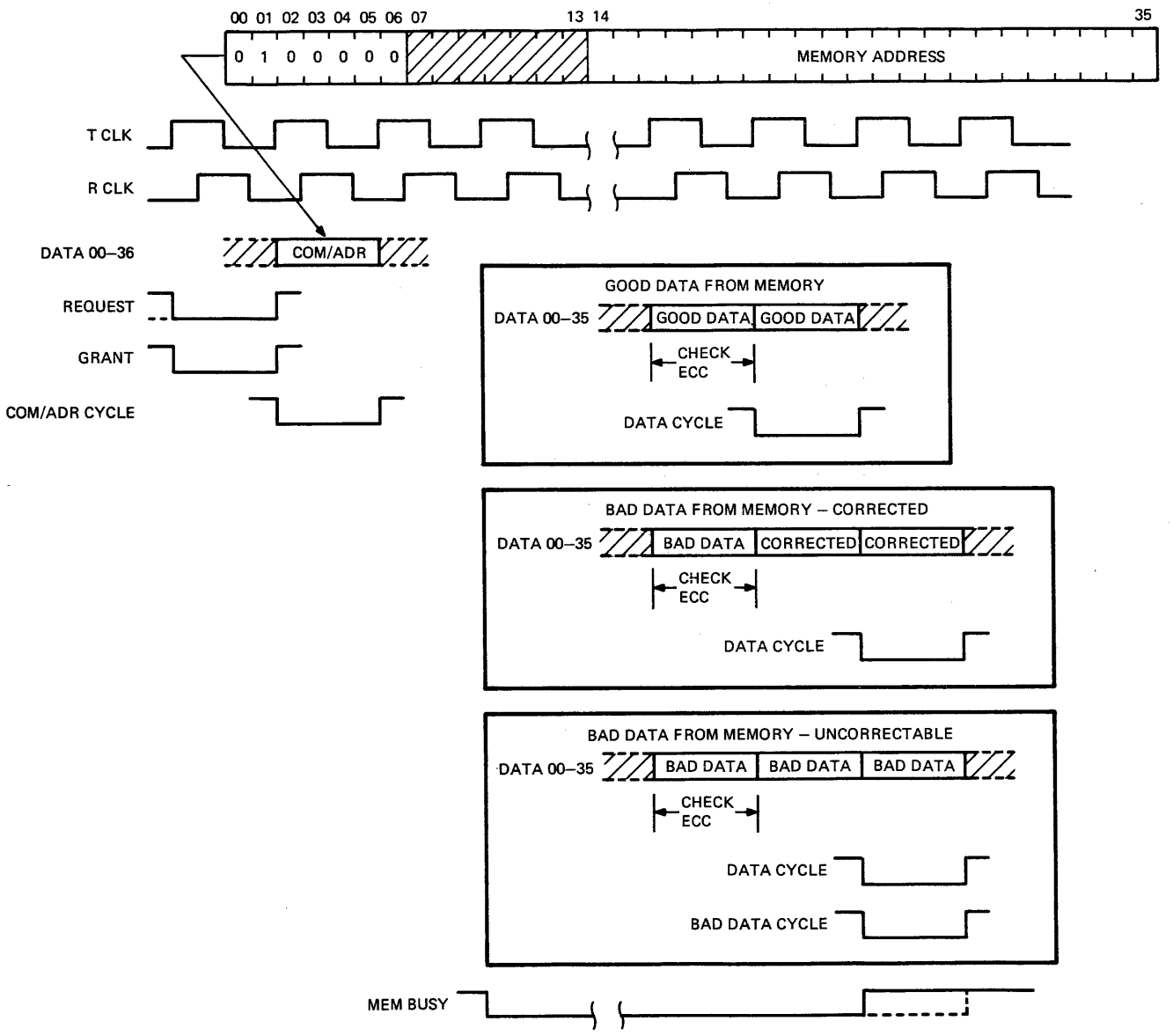


Figure 5-11 Memory Read, Bus Timing Diagram

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. 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 module 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 5-12.

5.3.6 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 00 = 1), an I/O address (bits 14–35), and the type of I/O operation; that is, a read (bit 01 = 1) or a write (bit 02 = 1). The command/address may also specify a byte transfer (bit 06 = 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 (bits 18–35). The memory and console both have a controller number of 0. UBA1 and UBA3 have controller numbers 1 and 3 respectively. Except for the memory status register (address = 100000_8) and console instruction register (address = 200000_8), all I/O registers are UBA internal or external registers. The internal registers include the 64 UBA paging RAM locations (addresses = 763000–776307_8), the UBA status register (address = 763100_8) and the UBA maintenance register (address = 763101_8). 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 alloted 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 module 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 06 = 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 5-13.

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 5-14. If a UBA internal or external register is addressed, read data is not asserted on the bus during the bus cycles allotted to the module initiating the operation. Instead, as shown in the lower part of Figure 5-14, the UBA requests the bus at some later time and transmits the read data whenever the bus is granted. The reason for this

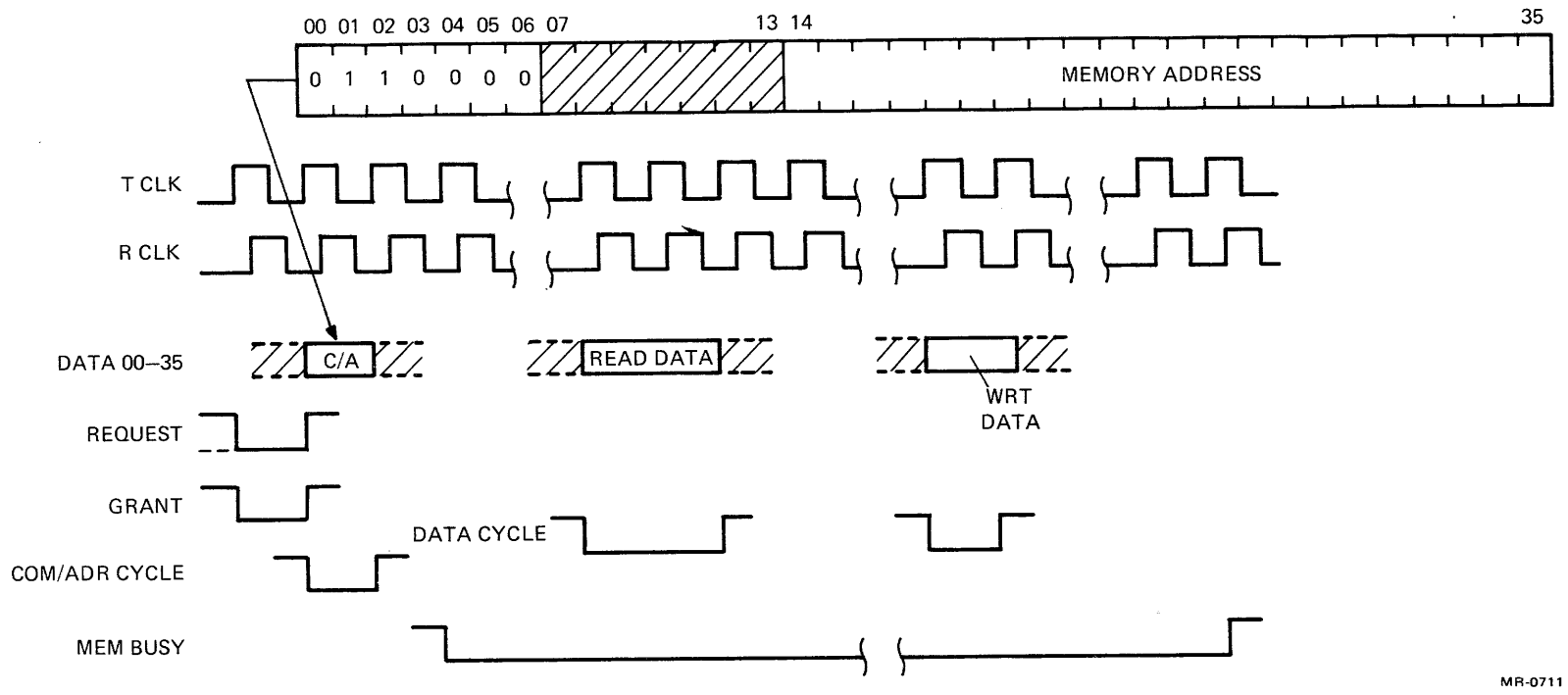


Figure 5-12 Memory Read-Pause-Write, Bus Timing Diagram

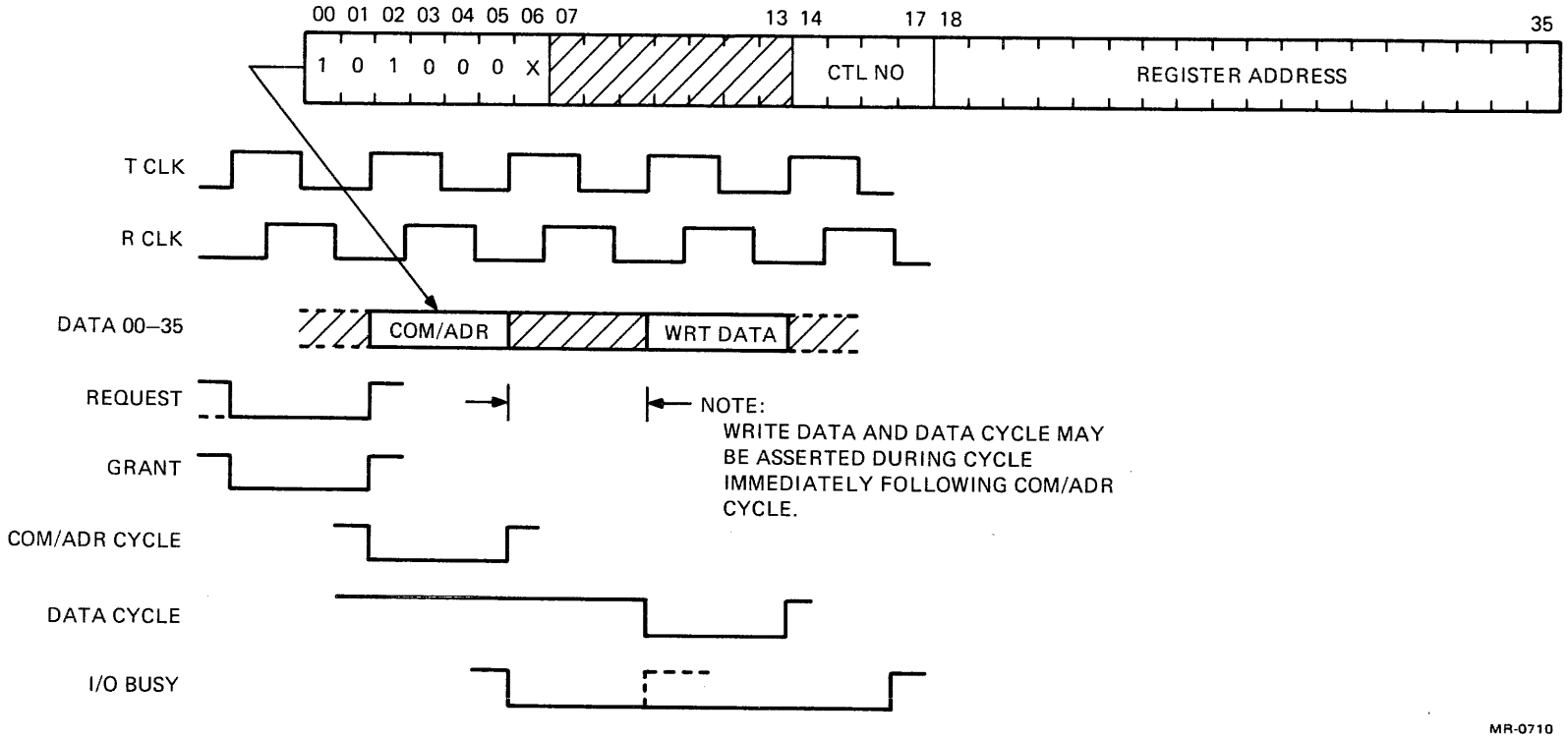


Figure 5-13 I/O Register Write, Bus Timing Diagram

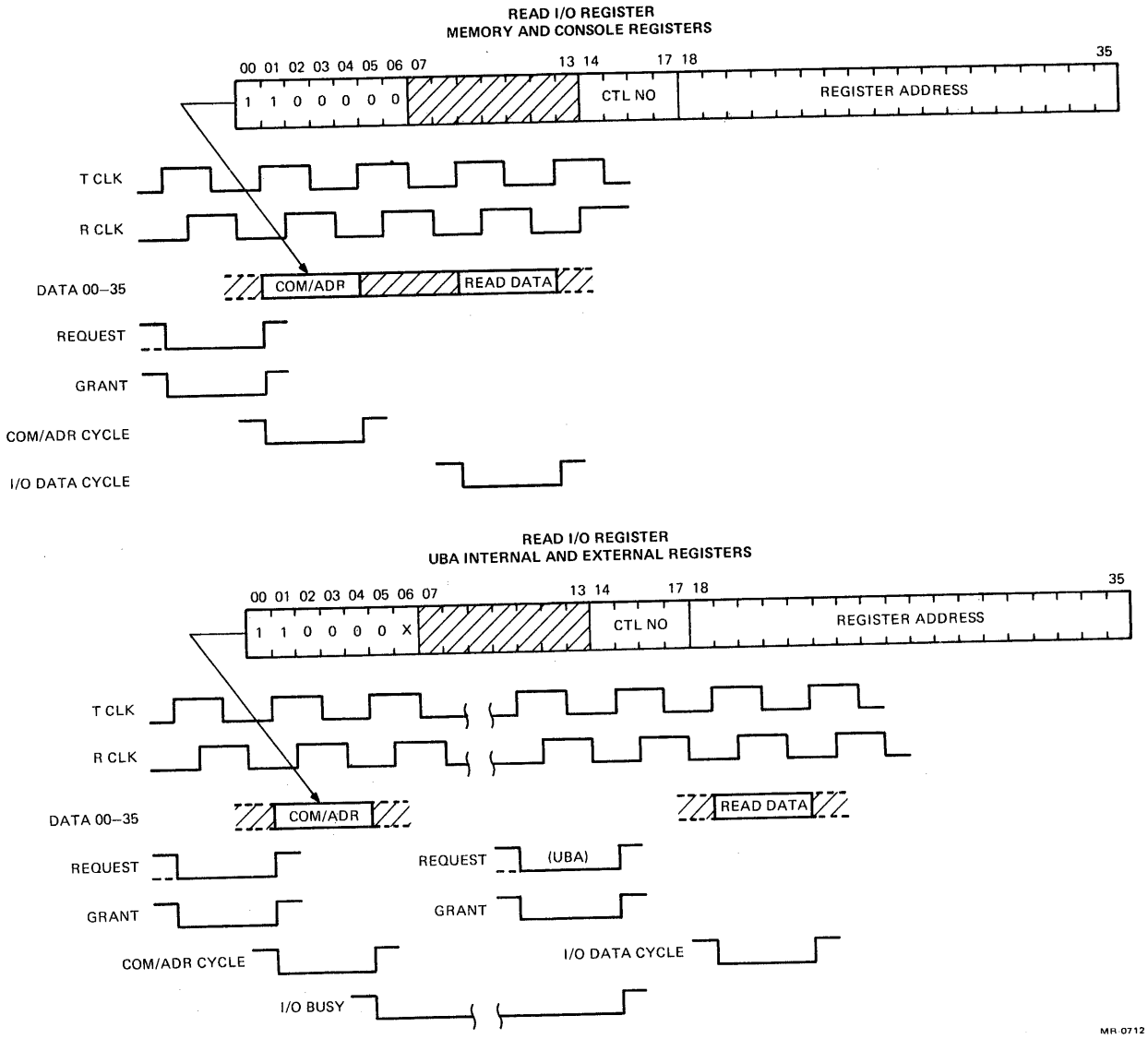


Figure 5-14 I/O Register Read, Bus Timing Diagram

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 module initiating the operation may gate the data from the bus.

5.3.7 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 (that is, 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 5-15. 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 00 = 1, bit 04 = 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 5-15. The command/address specifies that an interrupt vector is to be read (bit 00 = 1, bit 01 = 1, bit 05 = 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 vector is read from the device interrupting on the highest level BR associated with the

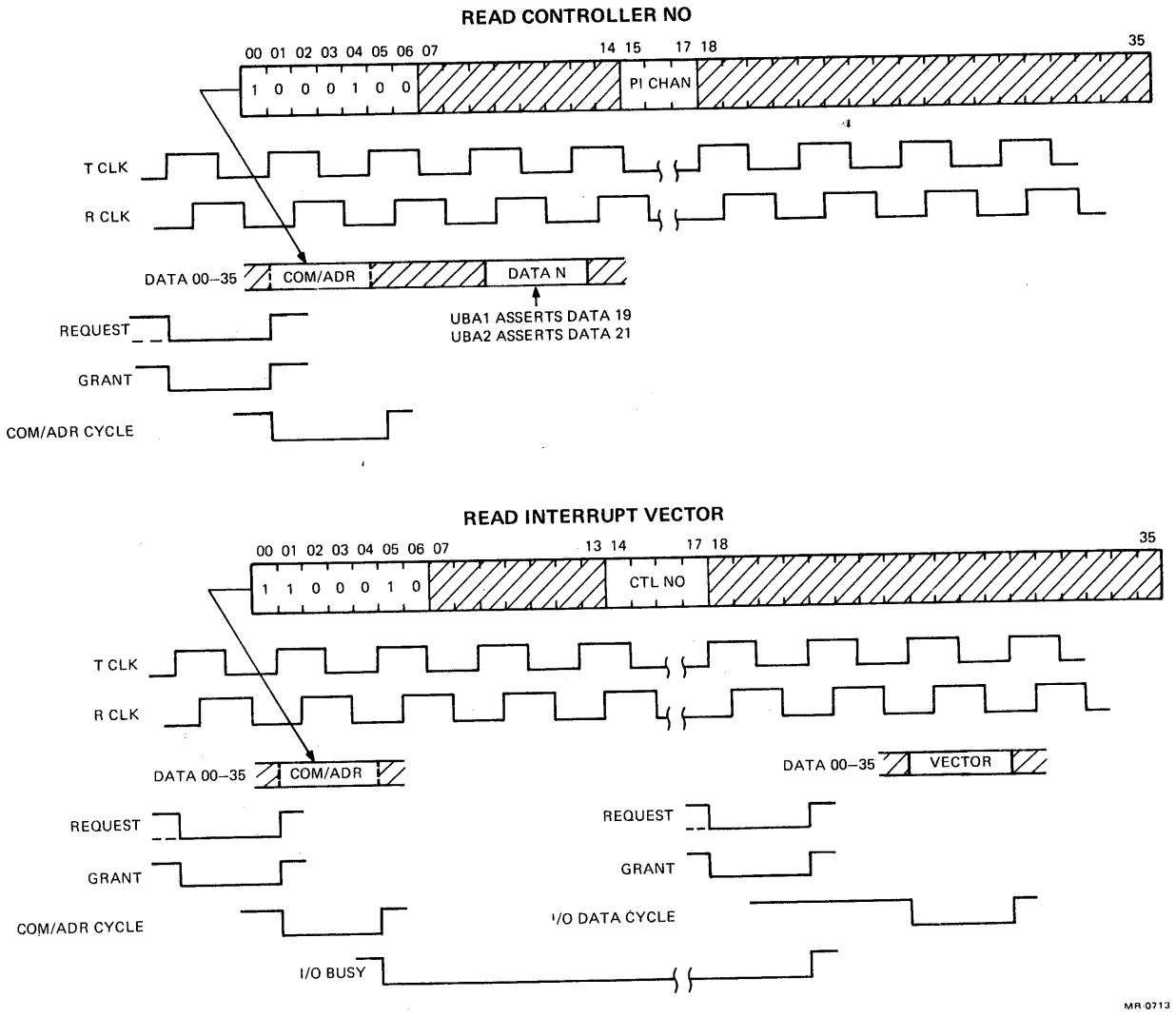


Figure 5-15 PI Operation, Bus Timing Diagram

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.

5.3.8 Diagnostic Operations

The console may perform diagnostic operations on the microcontroller in the CPU. Only 12 of the 36 KS10 bus data lines are used to transfer diagnostic data. These are data lines 24–35, which connect to the CRA module. Data lines 21–23 also connect to the CRA module, but they are used only for parity purposes. That is, when data is transmitted to the console, odd parity for each of the three 4-bit groups of data is also transmitted to ensure even bus parity. (PARITY LEFT and PARITY RIGHT do not connect to CRA.) No other data lines are used. Also, the standard bus control lines that gate and strobe bus data (such as DATA CYCLE) are not used. Instead, the console generates special control signals to load the diagnostic data, some of which are actually asserted after the interval that the console is bus master.

There are three types of diagnostic operations. The diagnostic write function is used to load microcode in the CRAM. Another diagnostic operation, the CRAM address load operation, must precede a write function in order to specify which of the 2048 CRAM locations are to be written. (Once the CRAM address is loaded, eight diagnostic write operations are required to load the specified 96-bit CRAM location.) Lastly, the diagnostic read operation is used to read either the CRAM address bits, the current location register, the subroutine return register, the CRAM parity checker outputs, the 8646 outputs, or the CRAM register. (Eight diagnostic read operations are required to read the 96-bit CRAM register.) All the diagnostic operations are controlled by the 8080 console program. They are invoked by the console commands (Table 4-3) and during system bootstrap.

Bus timing for the CRAM address load operation is shown in Figure 5-16. The console first requests the bus to become bus master. After being granted the bus, it transmits the 12-bit address on the data lines and asserts CSL4 CRA R CLK. This signal is used by the CRA module to latch the CRAM address on the data lines into its 8646 bus transceivers. To end the operation, the console asserts CSL4 CRAM ADR LOAD which loads the latched 8646 outputs into the diagnostic address register, also in CRA.

Bus timing for the diagnostic write operation is shown in Figure 5-17. Operation is similar to the CRAM address load in that the console asserts the data lines and then the latching signal CSL4 CRA R CLK after becoming bus master. Diagnostic data (that is, microcode) is transmitted on the data lines however. Also, CSL4 CRAM WRITE (instead of CRAM ADR LOAD) is asserted together with a diagnostic function (0–7) on the four diagnostic lines CSL4 DIAG 10, 4, 2, and 1. The diagnostic function selects which 12-bit group of microcode is to be loaded, and CSL4 CRAM WRITE causes the information to be written in the CRAM.

During the diagnostic read operation, the console also asserts the diagnostic lines. Timing is shown in Figure 5-18. The diagnostic function ($0–17_8$) transmitted on the lines selects the diagnostic data to be read from the microcontroller. The console then requests the bus and asserts CSL4 CRA T CLK ENABLE when the bus is granted. This signal connects to the 8646s in the CRA module and causes the selected diagnostic information to be transmitted on the bus data lines. The console then strobes the data lines to end the bus operation.

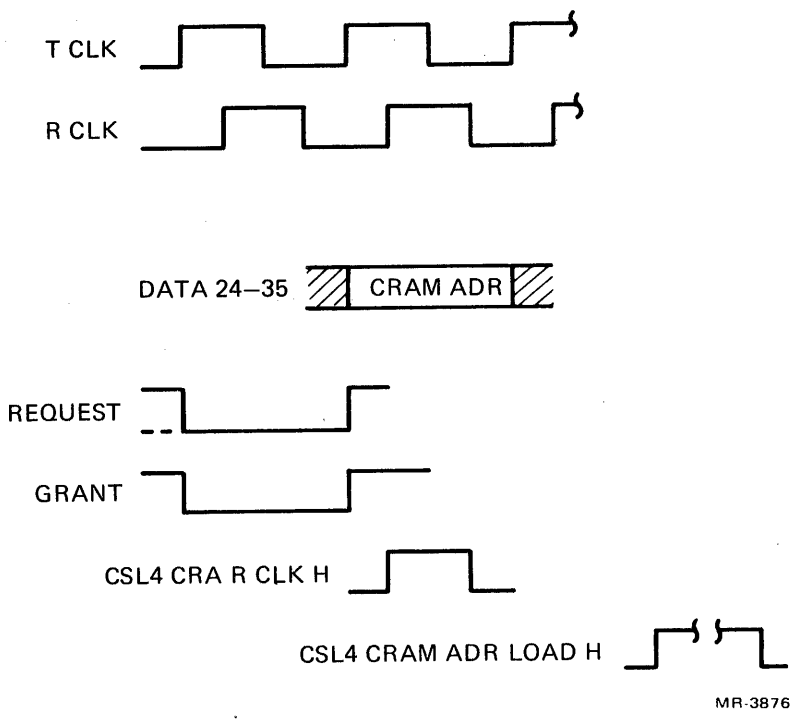


Figure 5-16 CRAM Address Load, Bus Timing Diagram

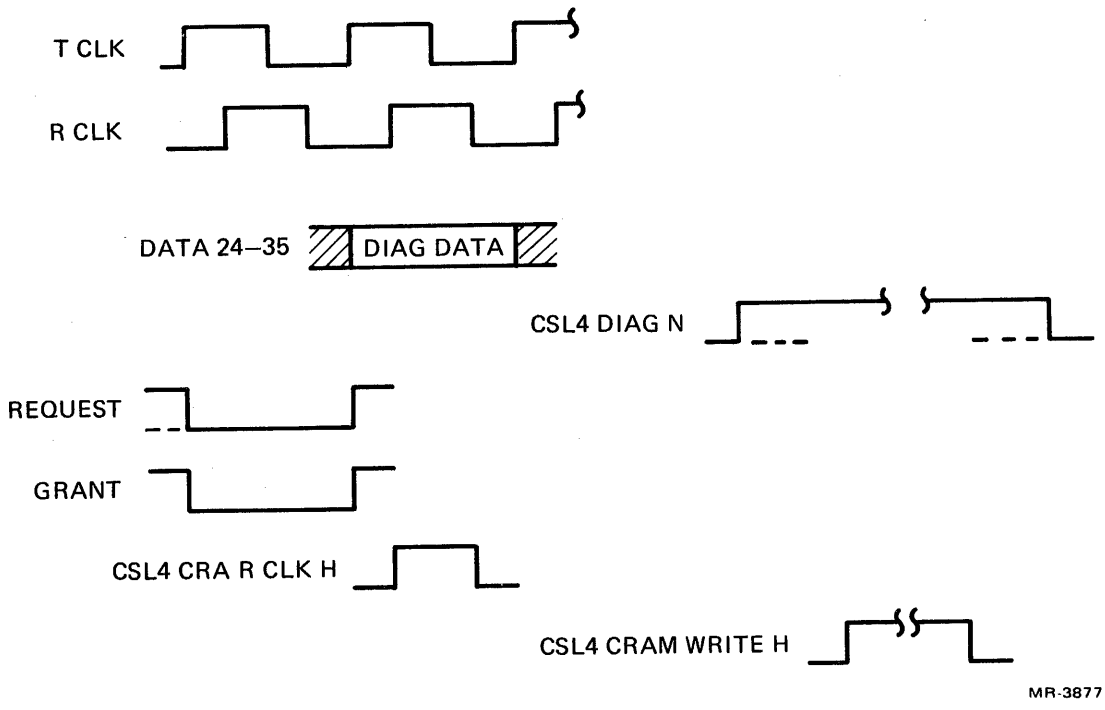


Figure 5-17 Diagnostic Write, Bus Timing Diagram

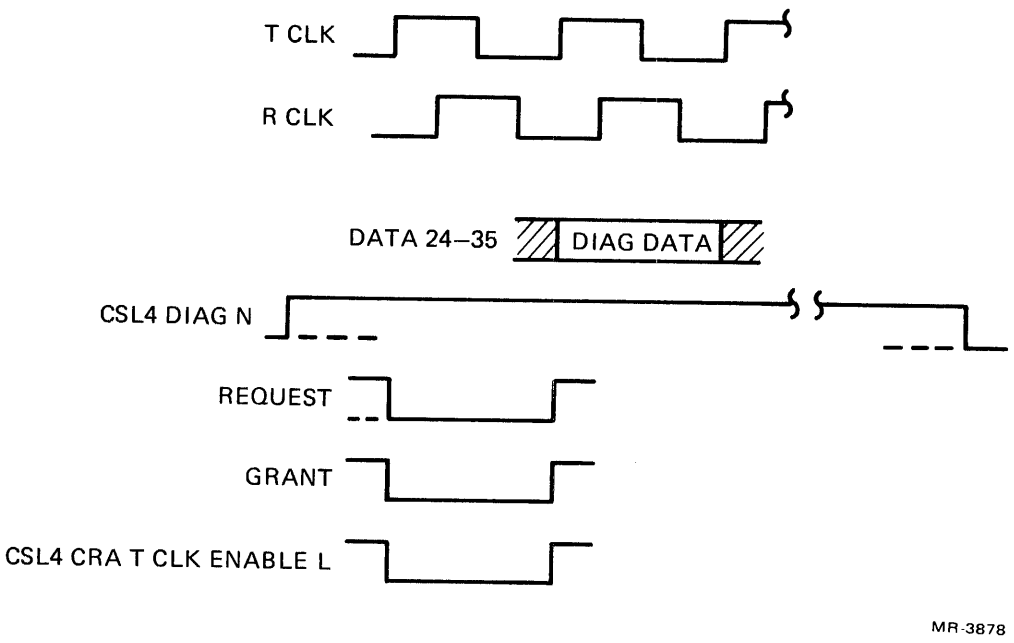


Figure 5-18 Diagnostic Read, Bus Timing Diagram

5.3.9 Bus Parity Error

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.

5.4 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, which are not treated here. The descriptive material that follows is devoted almost entirely to the hardware. It also includes those microcode procedures of a 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. These are discussed in Paragraphs 5.4.1 and 5.4.2. The microcontroller hardware is described in Paragraphs 5.4.3–5.4.6. A block diagram is shown in Figure 5-19.

5.4.1 Microword

The upper part of Figure 5-20 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). These bits are used only to default to other fields that are in the actual microword. In the field definitions given in the

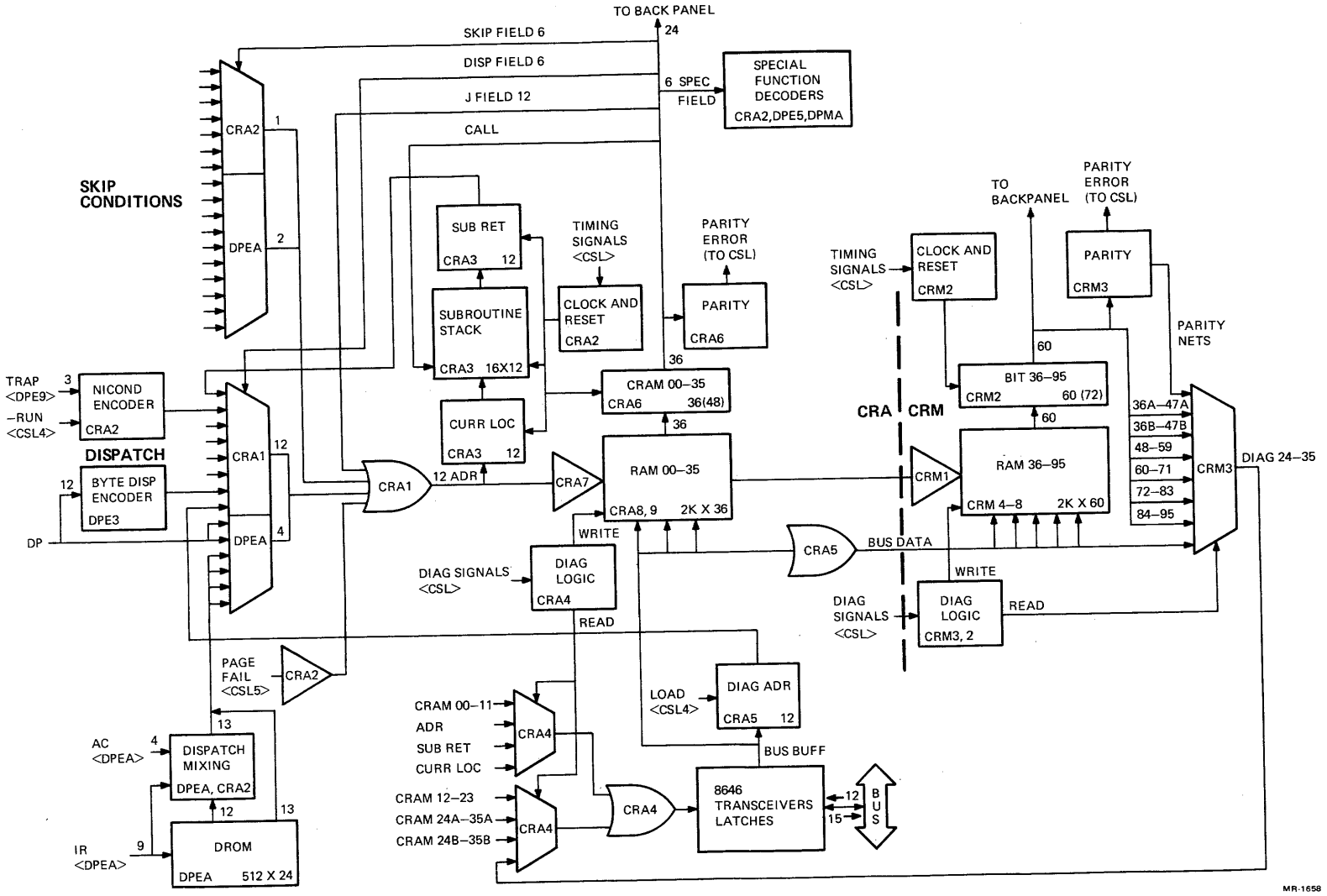
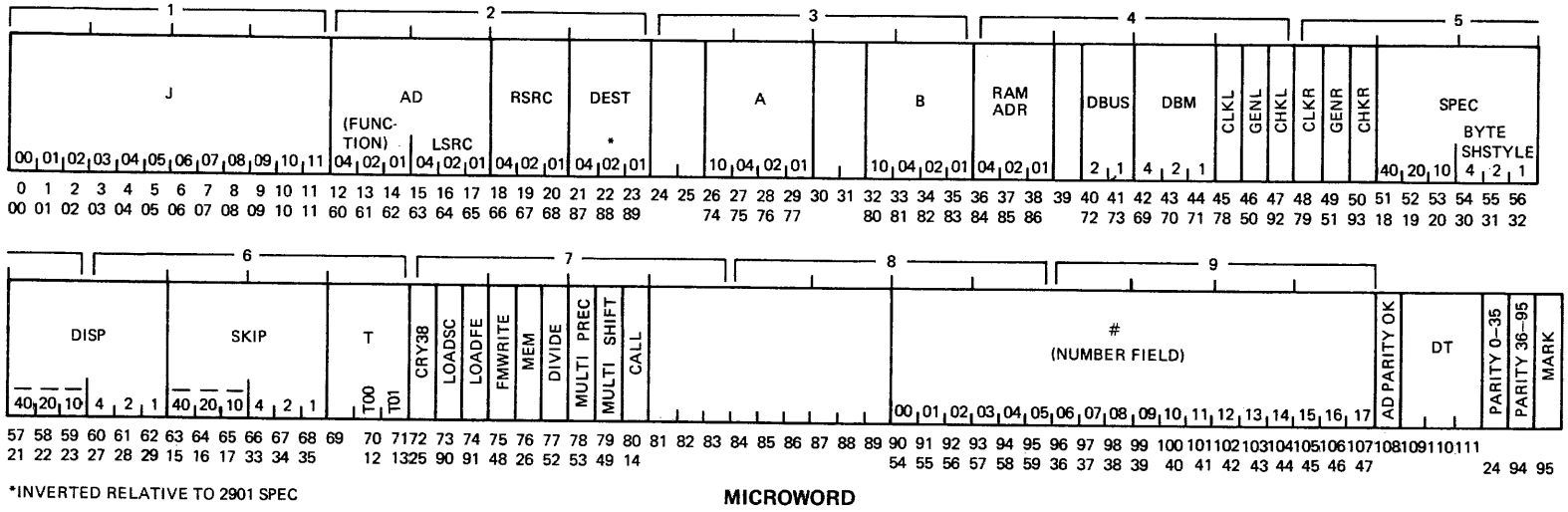
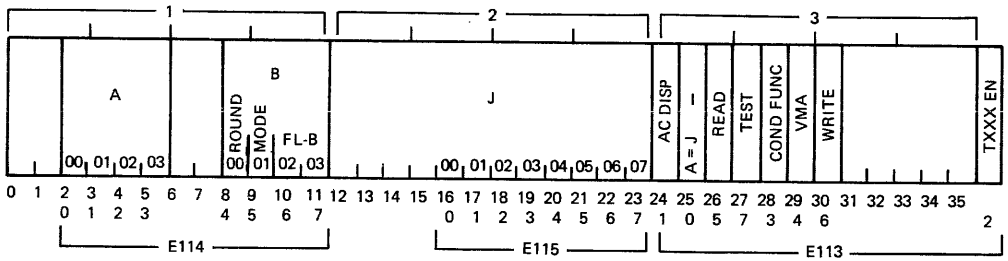


Figure 5-19 Microcontroller, Block Diagram



*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 5-20 Microword Formats

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.

The labels used for the fields of the microword are those defined for the microcode assembly language. Multiple labels indicate bits that are used for more than one purpose depending on the circumstances – the number field is used for many purposes as listed at the lower right of Figure 5-20. 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 and matched to the physical bit numbers.

The rest of this Paragraph explains the various groups of microword bits. It serves 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 This is 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 performed by the 2901, and the right three specify the source operand. The 9-bit instruction specification is completed by the three destination bits.

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 This field is 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 This is the source of data for the DBus via the DBus mixer.

42–44 If the DBus field selects DBM as the source, this field selects the source of data into the mixer that feeds the DBM input to the DBus 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 DBus, 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 DBus 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 DBus 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 DBus. A parity check should always be made when the source of the DBus 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 overridden 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 bit 40) 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 that this field specifies. 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 This bit inserts a carry into the LSB of the adder.

73 This bit loads the step counter from SCAD as set up by the number field.

74 This bit loads FE from SCAD as set up by the number field.

75 This bit writes the contents of the DBus into the RAM file at the location specified by bits 36–38.

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

77 This bit indicates the microinstruction is doing a divide.

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

79 This bit 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 This bit 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 Figure 5-20.

5.4.2 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 5-20 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. For a simple read it indicates whether the next instruction can then be fetched immediately.

4–7 This field specifies the test condition in all test instructions. It specifies 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 This 8-bit field specifies the address of the control RAM location at which execution of the instruction begins. It selects a location in the range 1400–1777$_8$.

24 This bit 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 This bit 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 (for example, MOVEI, JFCL).

26 This bit starts a memory read when the microword selects an AREAD memory function.

27 This bit starts a write test (for page fail) when the microword selects an AREAD memory function.

28 This bit starts a memory cycle when the microword selects a BWRITE conditional memory function.

29 This bit loads VMA when the microword selects an AREAD memory function.

30 This bit starts a memory write when the microword selects an AREAD memory function.

5.4.3 Control RAM

The 2048-word control RAM 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, and 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 5.4.4) 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 (at the same time) 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 the microcontroller clocks are not, 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 some 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). A duplicate for the 10 bit is at the right on CRA2.

5.4.4 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 5-19 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 value for 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 the last CRAM location (3777_8).

5.4.4.1 Dispatch ROM – The left half of each instruction is loaded from the DBus 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 $1400–1777_8$. In the normal situation, J bits 4–7 go to address bits 8–11 through the dispatch mixer at the top of DPEA in the same way as do 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 A2 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.

5.4.4.2 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, making the J dispatch unnecessary.

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 5.5.4) 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.

5.4.5 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.

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. At the same time as 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.

5.4.6 Booting and Diagnosis

The logic through which the console directly manipulates the microcontroller is shown across the bottom of Figure 5-19. 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 microinstructions, 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 is clear, J is 0, 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.

5.5 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 handles all the internal operations for the execution of an instruction – arithmetic and logic operations and data manipulation. The memory data path handles all aspects of communication over the backpanel bus for both memory and I/O instructions. It determines whether a memory access should be made to the RAM file (a cache or AC reference) and thus whether action is required of 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. This paragraph deals with the execute part of the path; the memory part is discussed in Paragraph 5.6. Figure 5-21 is a block diagram of the execute path for the internal structure of the register and RAM files, however, refer to Figure 5-2.

5.5.1 Arithmetic Unit

The heart of the main data path is the arithmetic unit (shown on prints DPE1,2) and most of the logic is in the 2901s themselves. Just as 36-bit words are centered in the 40-bit register file, the DBus 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. Instead of direct carry connections, however, 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 (that is, 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 microword 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 as the extra bits can be masked out. 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 5-22 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 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 step 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. 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 operation (using the right nine slices). 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.

5.5.2 Main Path

The output of the arithmetic unit is available via DP to many processor elements, including the mixers for the DBus on prints DPE3,4. These mixers can also select either 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 DBus to either the arithmetic unit, the RAM file, or the instruction register at the lower left on DPEA. All of the instruction bits can be loaded together by two special functions. One function handles both the IR and AC fields. 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 a value of zero for use by the skip and dispatch logic.

The remaining DBus 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 the 16 × 4 RAM that implement the parity arrangement described in the discussion of microword bits 45–50 in Paragraph 5.4.1. 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 DBus check enable signals stem from the corresponding valid bits.

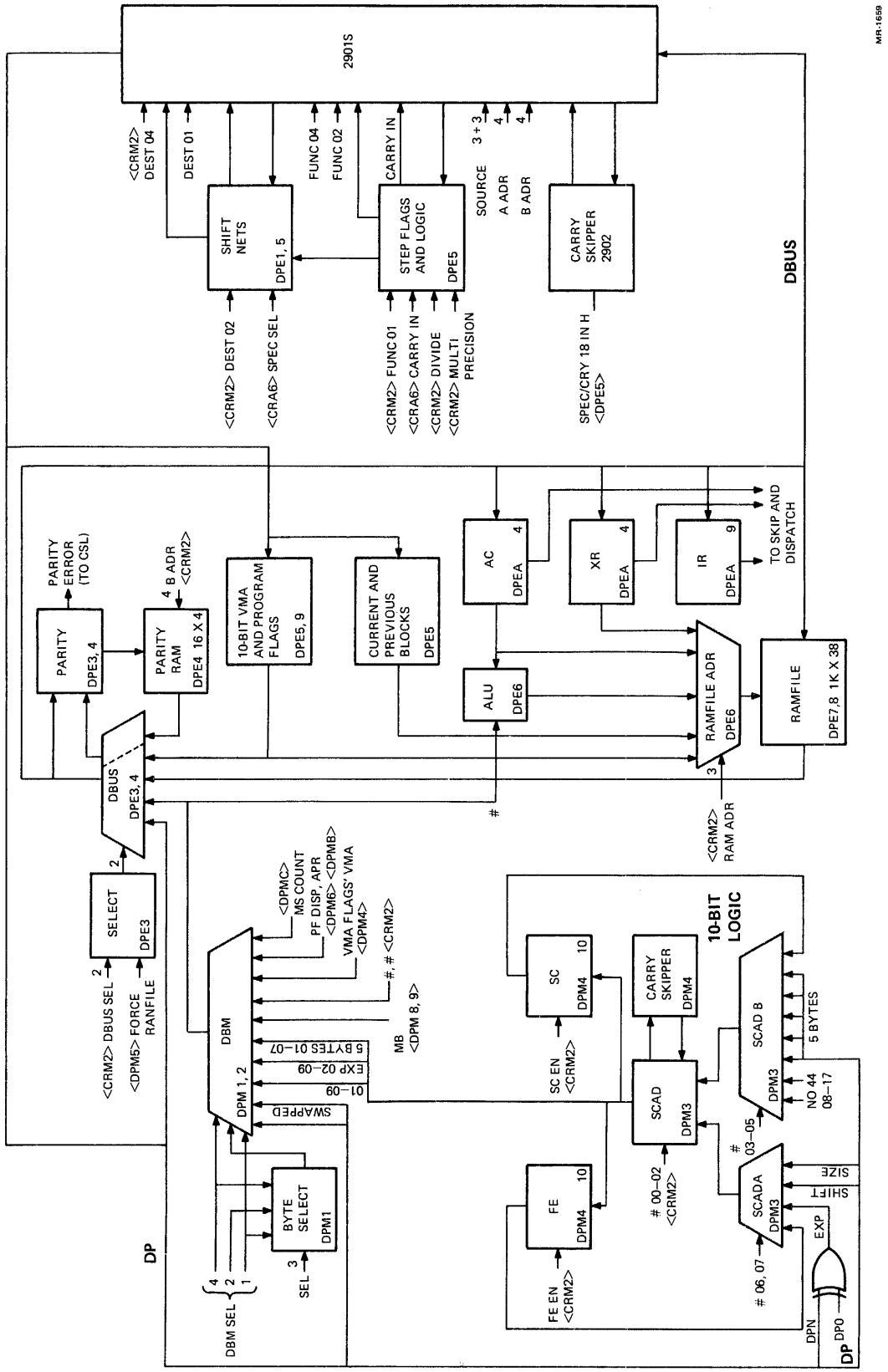


Figure 5-21 Data Path (Execute), Block Diagram

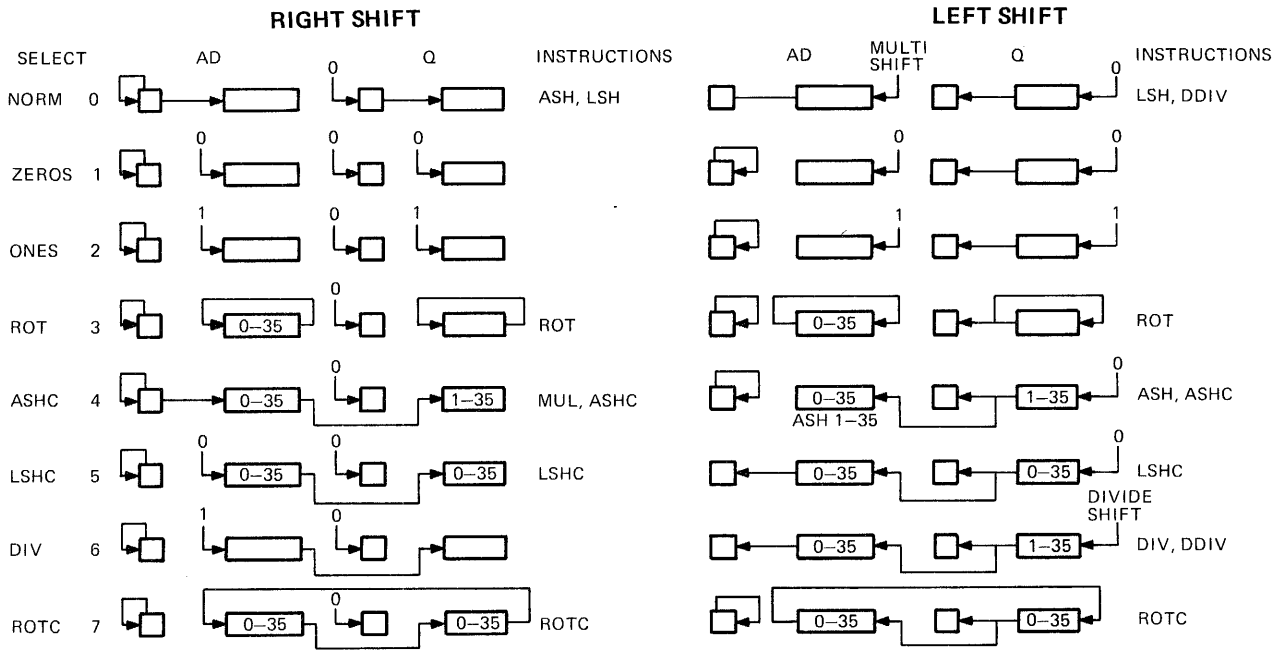


Figure 5-22 Shift Configurations

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.

5.5.3 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 DBus. 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 5.6.4). 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 ten 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.

5.5.4 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 the following: 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.

5.5.5 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 FPD (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 Volume I, Paragraph 2.9 of the *Hardware Reference Manual* (EK-10/20-HR).

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.

5.6 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, when an attempted access results in a page failure, or when the memory function is simply a write test (that is, 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 set-up information is supplied by the number field, where bits 0–11 select the type of memory cycle (that is, read, write test, write) and specify other associated characteristics such as 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 5-23 is a block diagram of the memory data path, with some necessary simplification and omissions.

5.6.1 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 5-23. 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 on DPM4 and the third at the upper left on DPM5. The four E511 flags (DPM4 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 for 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, depends 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 that the cycle type can 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.

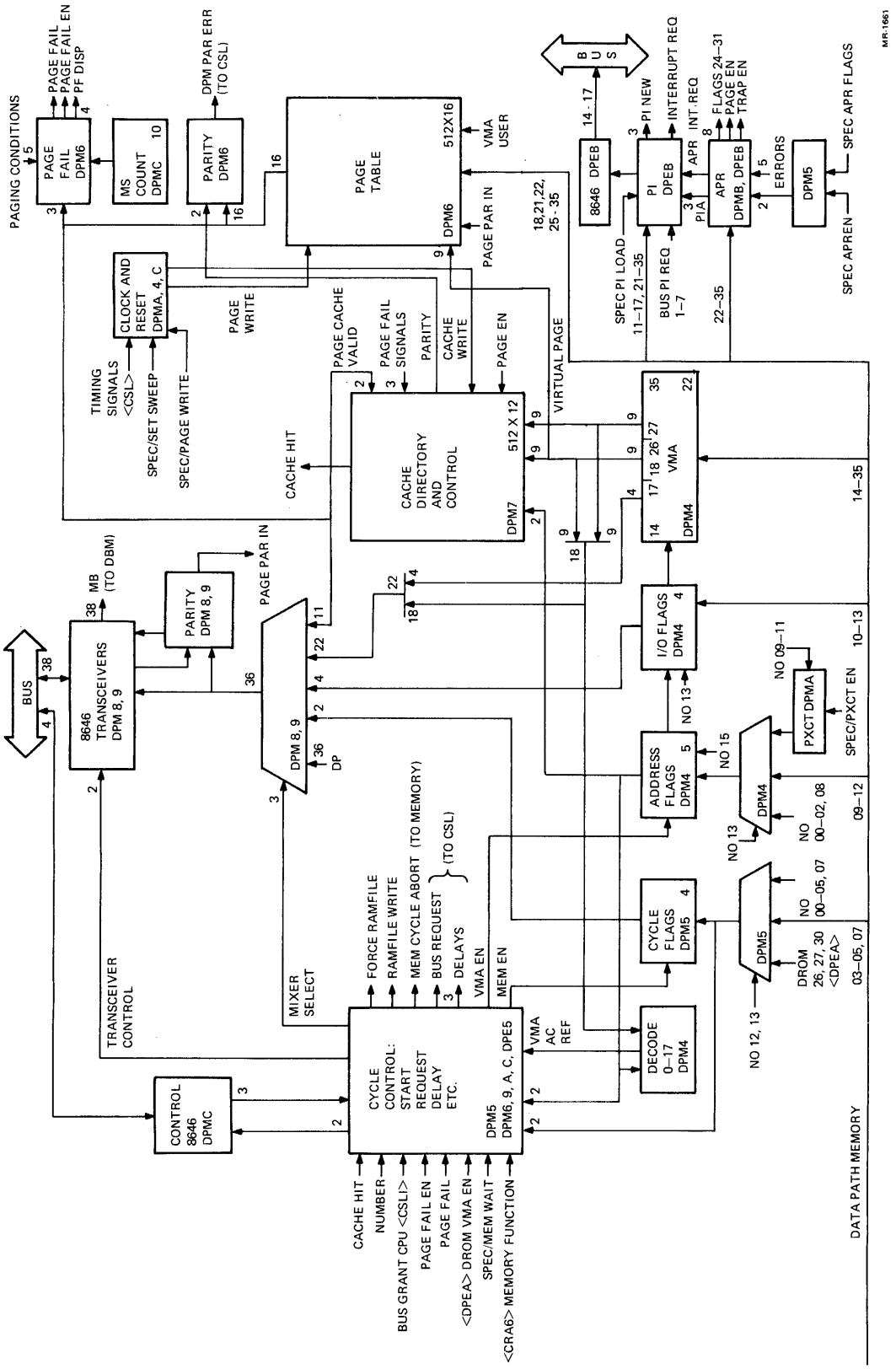


Figure 5-23 Data Path (Memory), Block Diagram

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 (that is, synchronize microcode to bus operation)	
17	Start cycle if DROM bit 28 (COND FUNC) is 1	

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 5.6.2. 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 $0–17_8$, 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 5.5.3; 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.

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.

5.6.2 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.

5.6.3 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. It invalidates two locations at a time by running through all configurations of VMA 19–26 while 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. Interrupts 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.

5.6.4 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 5.5.3.

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 consists of 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 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 5.6.1. 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.

5.6.5 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.

5.6.6 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 (that is, program-initiated) interrupt requests, turn the system on and off, and specify on which levels of 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 DBus 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.

5.7 MEMORY

An internal MOS memory is the primary storage in the KS10 system. The memory has a 0.9 μs cycle time and consists of a M8618 memory controller interfaced to the KS10 (backplane) bus, plus 2-8 M8629 storage array modules connected to the controller via MOS data bus. A block diagram is shown in Figure 5-24. (Note that only one storage array module is diagrammed.) Each array module stores 64K 43-bit words to give a total system memory capacity of 128K to 512K. The 43-bit word consists of 36 data bits plus 7 check bits. The check bits provide for 1-bit error correction and 2-bit error detection when retrieving data from memory.

Access to the MOS memory by the other modules on the KS10 backplane (that is, CPU, CSL, UBAs) is via the KS10 bus. A KS10 bus master may do the following.

1. Write memory
2. Read memory
3. Read-pause-write (RPW) memory
4. Read/write the memory controller status register

The memory and status read/write operations are diagrammed in Figures 5-25 and 5-26. (The RPW is not shown as it is a read operation followed by a write to the same address.) As indicated, the operations are initiated by a KS10 bus command/address cycle. Information is then transferred during a following data cycle. KS10 bus operation is described in Paragraph 5.3.

Another major memory operation is the refresh cycle which is initiated by the memory controller itself. The refresh cycle is required to periodically recharge the storage cells on the array modules so that stored data will not be lost.

5.7.1 Storage Organization and Addressing

A 16K (16,384) location MOS chip is the basic storage element on the M8629 (MMA) array modules. A chip stores one bit position for a range of 16K addresses, and there are 172 chips on each module to provide the total storage capacity of 64K 43-bit words. When a location is accessed for the storage or retrieval of data, 43 of the 172 MOS chips are written or read in parallel. There are 4 of these 43-chip groups on a module; print organization for the various bit positions within each group is as follows.

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)

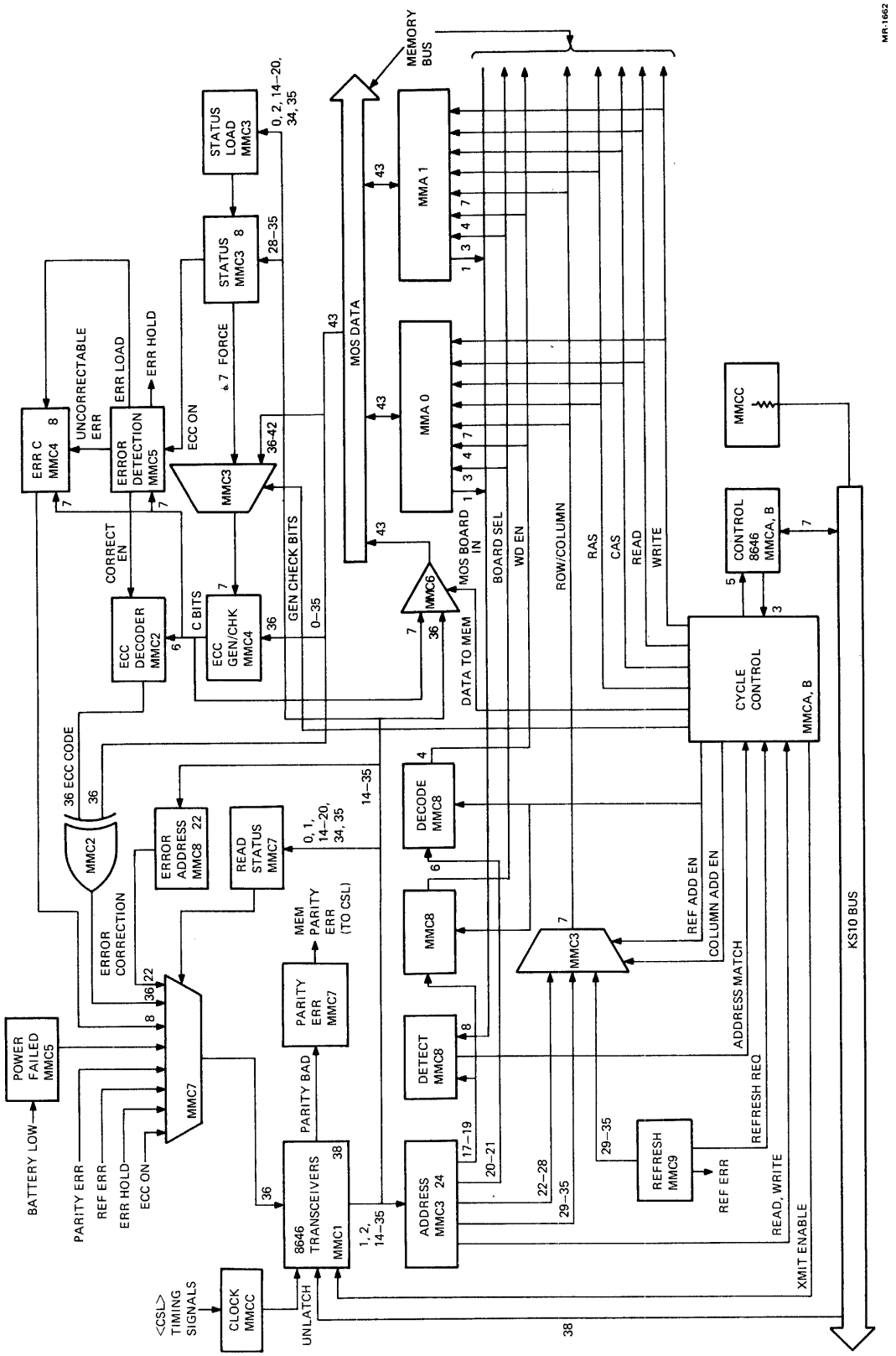


Figure 5-24 MOS Memory, Block Diagram

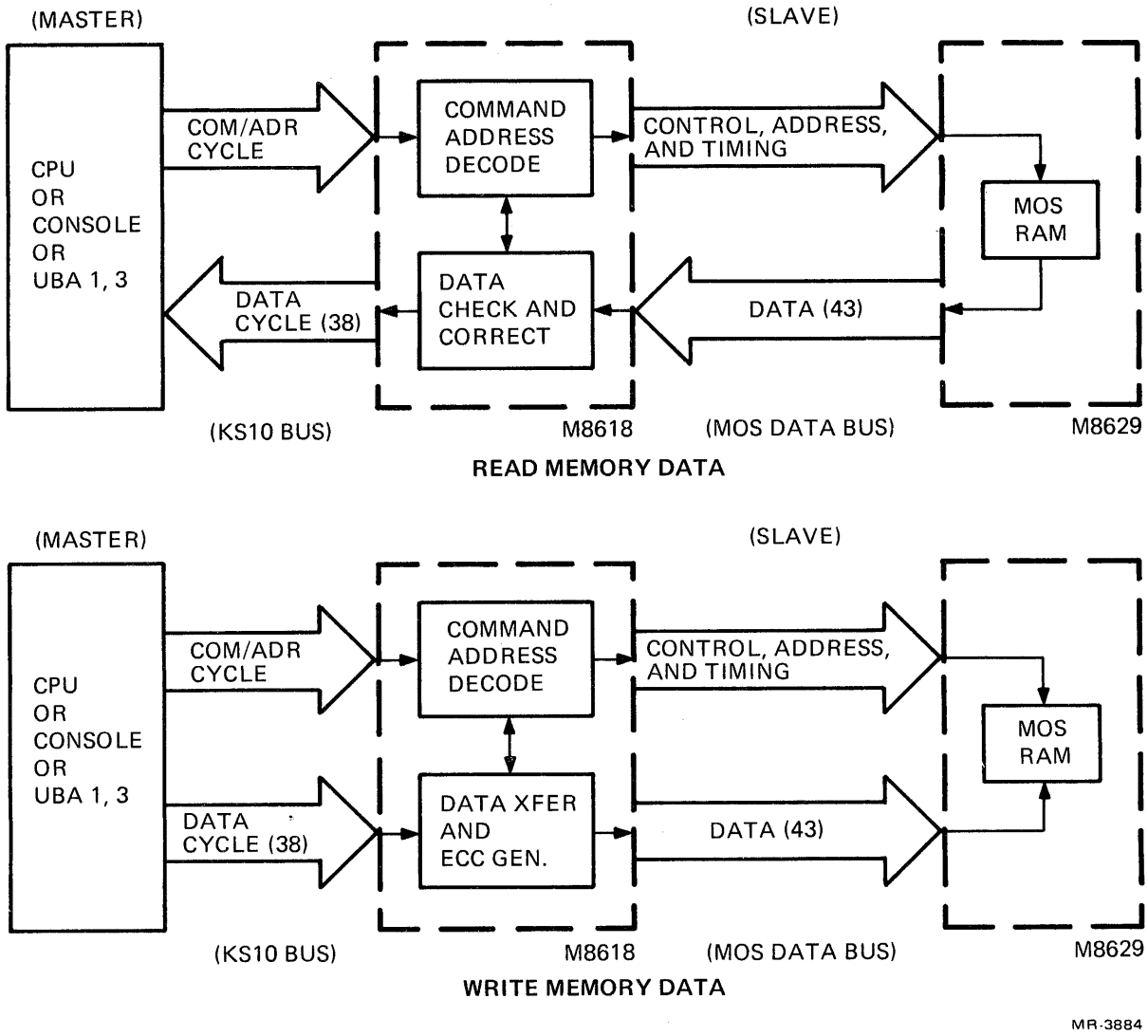


Figure 5-25 Memory Read/Write Operation

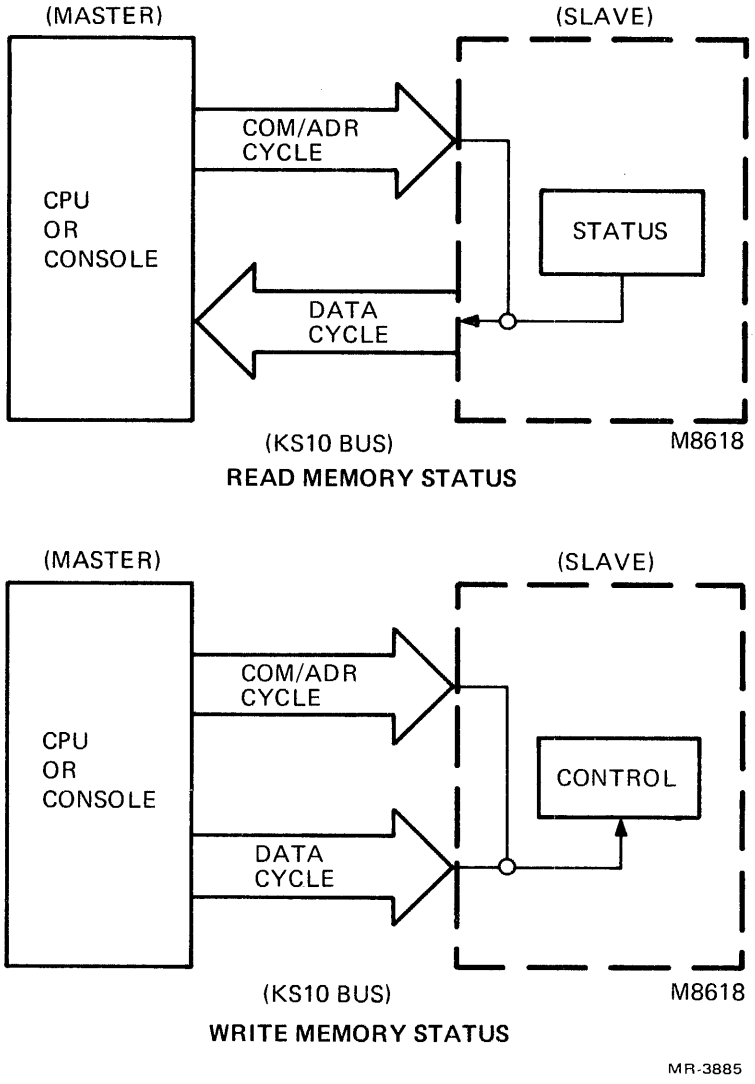


Figure 5-26 Status Read/Write Operation

Selection of a single word in one set of 43 chips in one of the array modules is made by the memory address (bits 14–35) supplied to the controller over the KS10 bus during a command/address cycle. The address bits do the following.

Address Bits	Descriptions
14–16 (3 bits)	Currently not used. Must be all 0s.
17–19 (3 bits)	Select 1 of 8 possible MOS storage array modules.
20–21 (2 bits)	Select 1 of 4 16K word groups (1 of 4 chips sets) on selected array module.
22–28 (7 bits)	Select 1 of 128 rows within the selected word group.
29–35 (7 bits)	Select 1 of 128 columns within the selected word group.

5.7.2 MOS Data Bus

The primary function of the MOS data bus interface is to transfer the memory read/write data between the memory controller and the storage array modules. The interface also provides for selecting the specified memory address in the array. That is, after receiving and decoding the address on the KS10 bus, the controller asserts signals on the interface to select the addressed array module, the word group, and one of 16K locations in the word group (by row and column). Interface signals are described in Table 5-4.

Table 5-4 MOS Data Bus Signal Summary

Signal	Description
MOS DATA 00–42	Bidirectional data lines. Transfer data to/from the MOS storage array.
BOARD SEL 4,2,1	Select addressed storage array module (0–7).
BOARD IN 0–7	These signals indicate array modules (0–7) are plugged into backplane. Each module asserts a separate BOARD IN signal.
WD 0–3 EN	Select addressed word group (0–3).
Row 22 Column 29 Row 23 Column 30 Row 24 Column 31 Row 25 Column 32 Row 26 Column 33 Row 27 Column 34	Multiplexed ROW/COL lines. Transmit ROW and COLUMN addresses to the MOS array.
RAS	Strobes the ROW address.
CAS	Strobes the COLUMN address.
READ	In conjunction with BOARD SEL signals, gates data from MOS chips onto MOS DATA lines.
WRITE	In conjunction with BOARD SEL signals, enables data on MOS DATA lines to be written into MOS chips.

5.7.3 Command/Address Load (Memory Access)

As stated previously, a KS10 bus master must perform a command/address cycle to initiate a memory-write, read, or RPW. The I/O control bit (data line 00) must be 0 and the read/write control bits (data lines 01 and 02) specify the operation.

In the memory controller, BUS COM/ADR CYCLE starts the operation by loading the read/write control bits and the memory address into the address register (MMC3 print). If not holding information from some previous error, the address is also loaded into the error register (MMC8) so that it may be saved should an error occur in the upcoming operation.

When the address register is loaded, the BOARD SEL, WD 0–3 EN, and the ROW/COL lines on the MOS data bus interface are asserted to begin selection of the memory address in the MOS array. The ROW/COL lines transmit the row address at this time.

After the address register is loaded, MMCA CA CYC SEEN sets to assert BUS MEM BUSY, thus freezing the KS10 bus arbitrator. CA CYC SEEN remains set to allow the memory access to continue provided the CPU as bus master is not aborting the operation (DPMC MEM CYCLE ABORT = 0), and provided the memory address is inbounds (MMCB ADDRESS MATCH = 1). The memory address is checked by comparing it against the BOARD IN signals on the MOS data bus interface.

Following the assertion of CA CYC SEEN, and with ADDRESS MATCH = 1, RAS is transmitted on the MOS data bus interface to strobe the row address asserted on the ROW/COL lines into the MOS array chips. Then, MMCA COLUMN ADD EN sets to transmit the column address on the ROW/COL lines, and the CAS signal is asserted to strobe column address into the addressed MOS array chips.

5.7.4 Memory Write

During a memory write operation after the command/address is transmitted on the KS10 bus, the bus master transmits the data on the bus that is to be written into the MOS storage array. The write data is transferred to the memory controller by means of a KS10 bus data cycle. That is, when BUS DATA CYCLE is received by the controller, it sets MMCB DATA HOLD which latches the controller's bus transceivers to store the write data. The data cycle can occur at any time after the command/address cycle. If generated immediately after, the latching of the write data proceeds in parallel with the controller sequence that generates RAS and CAS as described in Paragraph 5.7.3.

Once BUS DATA CYCLE is received and the write data is latched, the data (plus the 7 check bits) is asserted on the MOS data bus interface. (The generation of check bits is discussed in Paragraph 5.7.7.) BUS DATA CYCLE also sets MMCB WE, which causes the WRITE signal to be asserted on the interface. The write data is then written into the selected MOS address. The rest of the control sequence resets the control logic and frees the KS10 bus arbitrator to end the operation. Timing for the write operation is shown in Figure 5-27.

5.7.5 Memory Read

During a memory read operation, the READ signal is transmitted on the MOS data bus interface by MMCA COLUMN ADD EN just prior to the assertion of CAS. After CAS is transmitted, the READ signal (together with the BOARD SEL signal) allows the read data and check bits in the selected MOS array location to be transmitted on the MOS data bus.

In the controller, the resulting check bits are saved (unless error information is already held) and the read data on the MOS data bus is transmitted on the KS10 bus by MMCA XMIT ENABLE (asserted by T4) at the same time that the MOS data is being checked for error by the ECC circuits. (Error checking is discussed in Paragraph 5.7.7.) If the data is correct, it is asserted on the bus for a second KS10 bus cycle. MMCA STROBE EN also asserts BUS DATA CYCLE so that the KS10 bus master may latch the information from the bus. If the data is not correct, the strobe is delayed for another bus

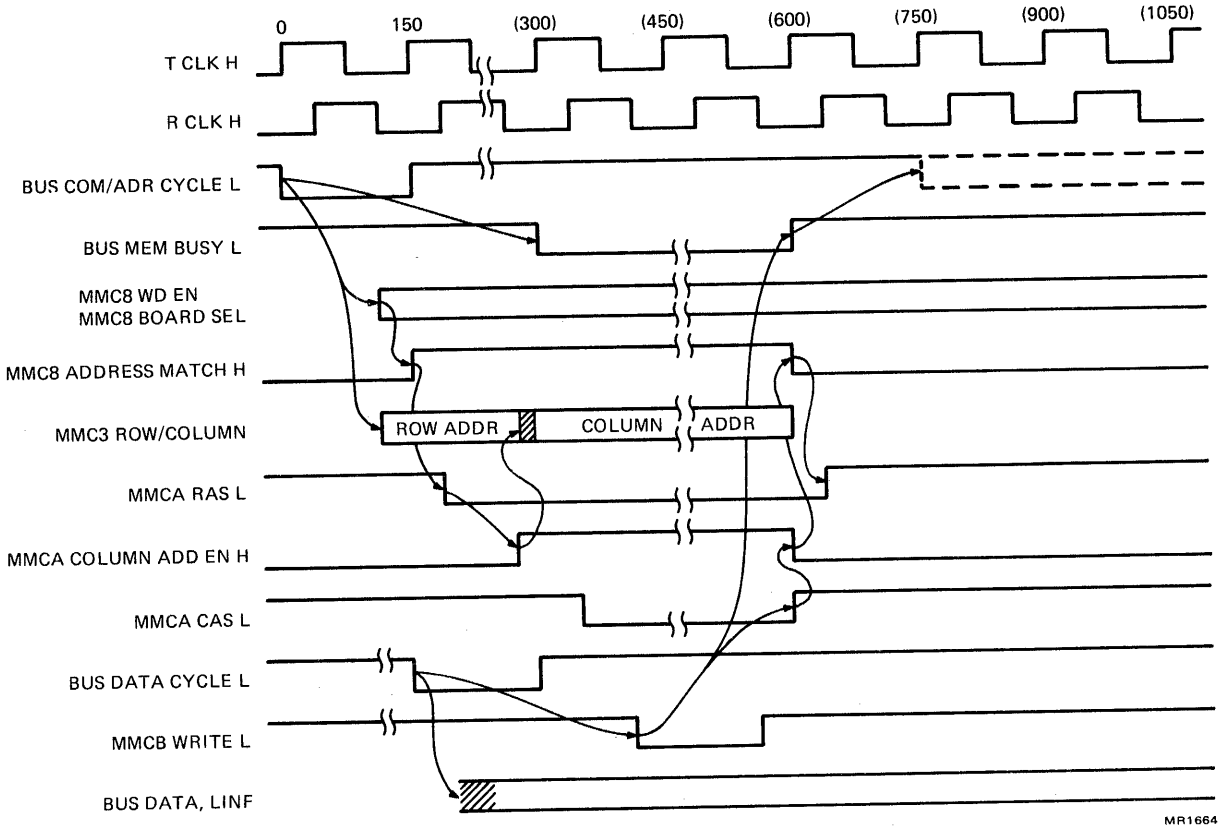


Figure 5-27 Memory Write, Timing Diagram

cycle, and then transmitted on the bus together with the corrected or bad (uncorrectable) data. If the data is uncorrectable, BUS BAD DATA CYCLE is also transmitted along with the BUS DATA CYCLE strobe to flag the error. Once a data strobe is generated, the control logic in the controller is reset and the KS10 bus arbitrator is freed ending the operation. Timing is shown in Figure 5-28.

5.7.6 Read-Pause-Write

A read-pause-write operation (RPW) is initiated when both the read and write control bits loaded during the command/address cycle are true (MMCA PAUSE = 1). First, a normal read operation occurs as described in Paragraph 5.7.5. However, PAUSE prevents the reset of the control logic at the end of read operation unless the read data is uncorrectable. (The reset occurs and aborts the RPW if the read data is uncorrectable.) If the read data is good or corrected, BUS MEM BUSY remains asserted to freeze the KS10 bus arbitrator and the controller waits (pauses) for write data. When the data is received, a normal write operation occurs as described in Paragraph 5.7.4. The completion of the write operation then terminates the RPW.

5.7.7 Error Detection and Correction

The KS10 memory word contains seven check bits in addition to the 36 bits of data. Seven bits allows single-bit error detection and correction. It also allows double-bit errors to be detected, but not corrected. Results are unspecific when there is an error in more than two bits. Error detection and correction is done in the memory controller.

The coding scheme used to generate the check bits is shown in Figure 5-29. Basically, it involves seven overlapping parity checks. The seven outputs from a parity generator/check network (MMC4 CP, C40, C20, C10, C4, C2, and C1) determine the check bit values and each is associated with a unique set of 18 data bits. For example, check bit C1 is associated with the 18 even number bits in the data word; check bit C40 is associated with bits 18–35.

The check bits that are generated for each memory word are even parity bits. That is, each check bit is generated so that it makes the overall parity of the check bit and the associated 18 data bits even. The check bits are written into the MOS memory location with the data. Then, during a following read, a parity check is made using the same parity network (now a parity checker) to verify that the checks bits and associated data still have even parity. If so, the seven outputs from the parity generator/checker will be an all-zeros check character or error correction code (ECC).

When a single-bit failure occurs, the overlapping parity checks associated with the failing bit will be odd. This generates a non-zero ECC (MMC5 READ ERROR = 1). Also, the resulting ECC will have a value that depends on the failing bit. For example, C40, C10, and C2 are associated with data bit 25. If this bit fails, the resulting ECC will have a value of 052 as shown below.

CP	C40	C20	C10	C4	C2	C1	
0	1	0	1	0	1	0	$= 052_8$

By decoding the various values of the ECC, it is possible to generate a signal to complement a failing data bit, thus correcting the error. (The circuitry is shown on the MMC2 print.) Table 5-5 gives the check bit patterns generated for correctable data bit errors and the specific bit position corrected in each case. Single check-bit errors are considered correctable errors although no correction (bit complement) is required.

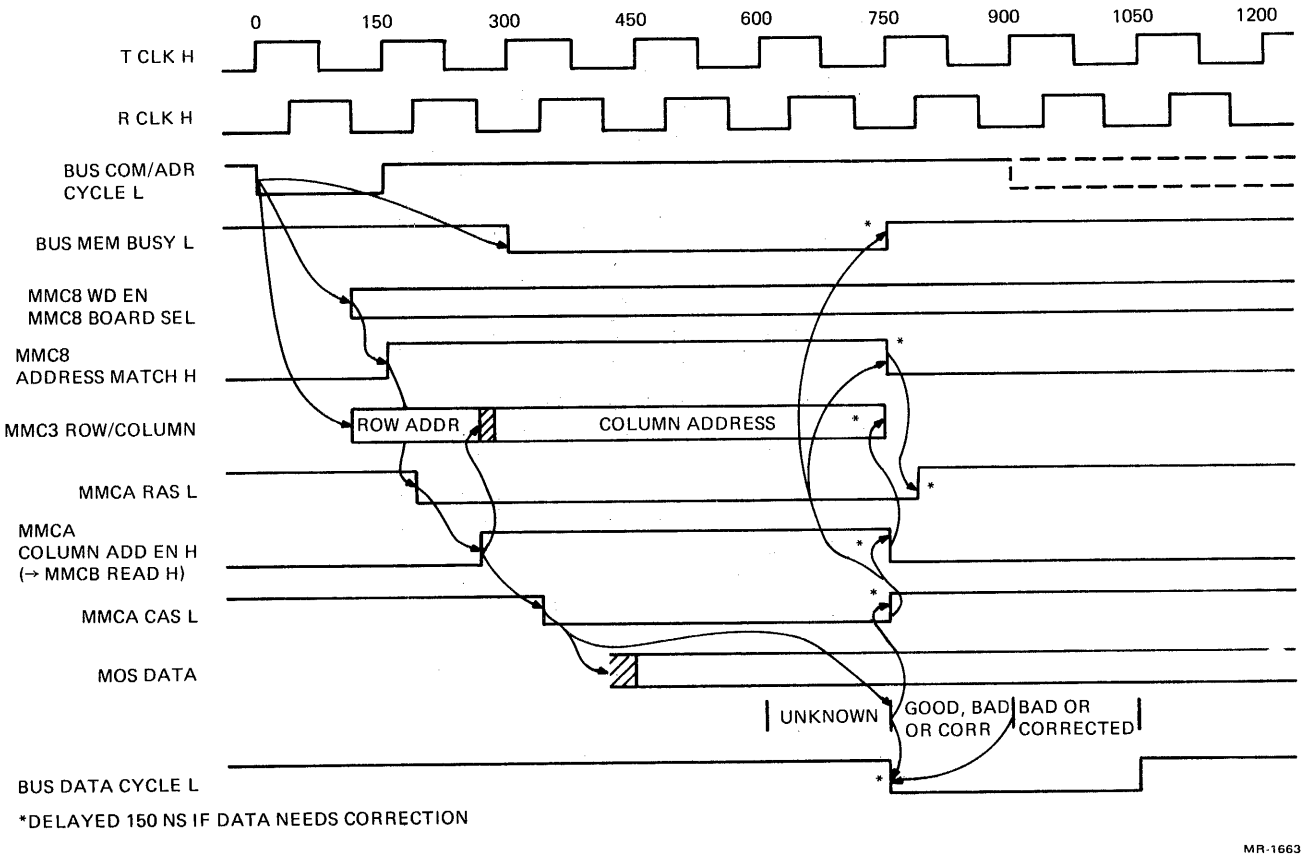


Figure 5-28 Memory Read, Timing Diagram

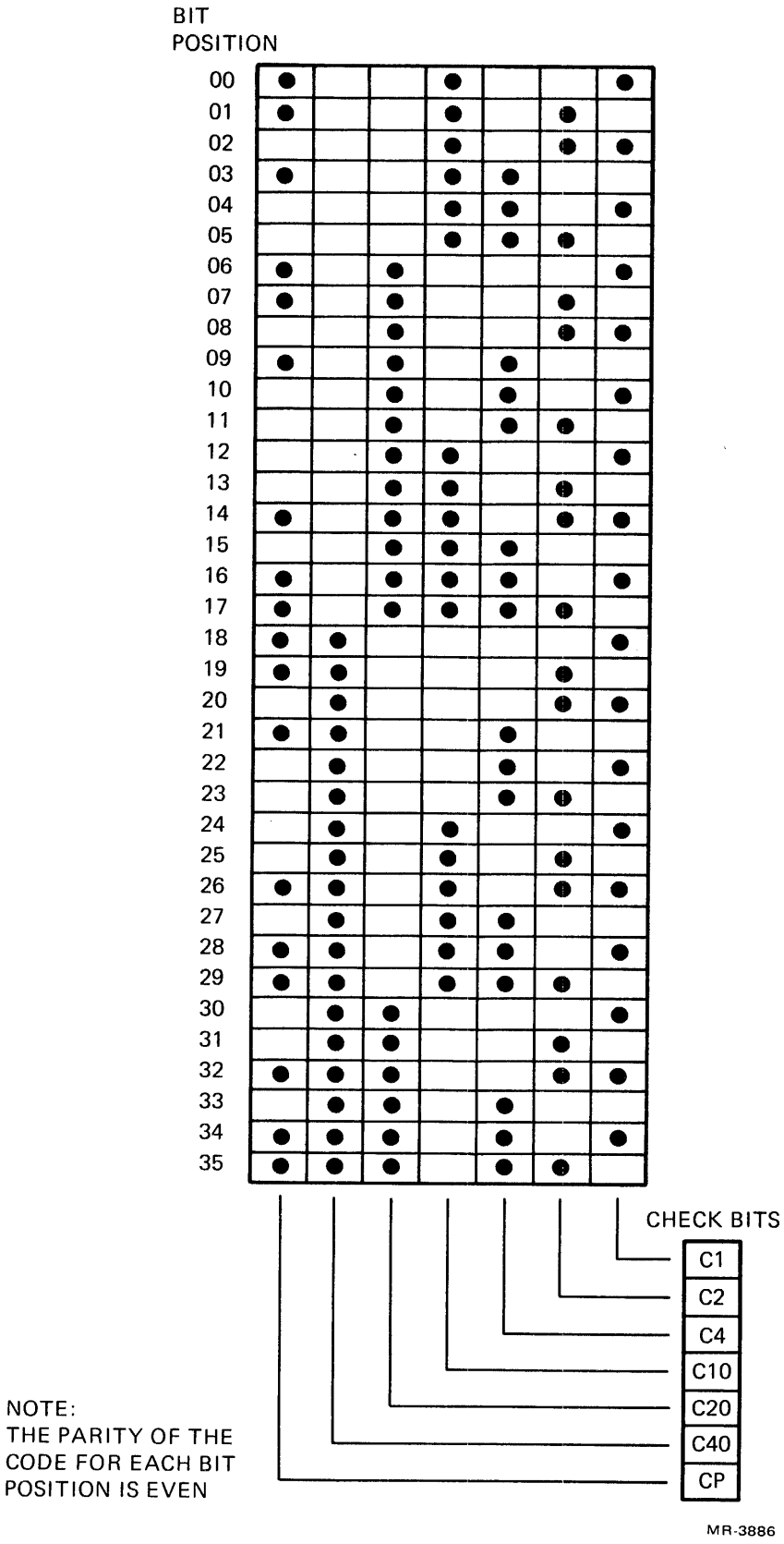


Figure 5-29 Check Bit and Data Bit Relationship

Table 5-5 Correction Codes

Check Bit Value	Data Bit Corrected	Check Bit Value	Data Bit Corrected
111	00	141	18
112	01	142	19
013	02	043	20
114	03	144	21
015	04	045	22
016	05	046	23
121	06	051	24
122	07	052	25
023	08	153	26
124	09	054	27
025	10	155	28
026	11	056	29
031	12	061	30
032	13	062	31
133	14	163	32
034	15	064	33
135	16	165	34
136	17	166	35

In addition to being decoded, the ECC bits are also checked for parity. This is necessary to distinguish between correctable and uncorrectable errors. Odd parity occurs for correctable single-bit errors, asserting MMC5 CORRECT EN which enables the ECC decoders (MMC2 print). Even parity occurs for double-bit errors, disabling the ECC decoders and asserting MMC5 UNCORRECTABLE ERR to flag the condition. The various types of failures are summarized in Table 5-6.

A special diagnostic feature permits the program to force incorrect check bits during a memory write; the error correction and detection logic may then be tested during a subsequent memory read. Seven force bits, one corresponding to each check bit, may be loaded into a register (MMC3 print) by a status write operation. The register outputs connect to the parity network that generate the check bits during the write. If a force bit is set, it causes the corresponding check bit to be incorrect. The force bits are set to 0 during normal operation.

5.7.8 Status Read/Write

Like a memory access, the status read or write operation is initiated by a command/address cycle on the KS10 bus. The bus master transmits the I/O bit on data line 00, the read or write bit on data line 01 or 02, and the I/O address for the memory status register on data lines 14–35. (The memory controller number is 0; the memory status register address is 100000_8.) If the operation is a status read, MMC7 READ STATUS is asserted to initiate a data cycle by the memory controller. That is, the signal gates and enables the transmission of the various controller status bits onto the KS10 bus data lines, and then causes BUS I/O DATA CYCLE to be asserted so that the bus master may latch the information from the bus. If the operation is status write, MMC3 STATUS IO WRITE sets to assert MMC3 STATUS LOAD EN. The bus master then initiates a data cycle and STATUS LOAD EN causes the control bits on the KS10 bus data lines to be strobed in the memory controller. The bit format for the status information read and written in the memory controller is shown in Figure 5-30.

Table 5-6 Failure Modes

Failing Bit(s)	Fault	Resulting ECC Value	Resulting ECC Parity	Action
None	-	Zero	Even	None
One data bit	Picked-up or dropped	Nonzero	Odd	Bad bit complemented before data transferred to master.
One check bit	Picked-up or dropped	Nonzero	Odd	None
Two data or check bits	-	Nonzero	Even	Data transferred to master and flagged by BUS BAD DATA cycle.
More than two data or check bits	-	Unspecified	Unspecified	Unspecified. Some 2-bit errors will be flagged as bad data; some will not.

5.7.9 Refresh Cycle

Each storage cell in a dynamic MOS storage array must be recharged at least once during a fixed time period called the refresh interval. The refresh interval for the 16K (128 rows × 128 columns) MOS chips used in the KS10 is 2 ms. A single row address strobe refreshes all the words in a given row in all word groups throughout the entire array. Thus, no matter how many array boards are present, the entire memory is refreshed by 128 strobes given to all possible row addresses. Because the refresh interval is 2 ms, a single row is refreshed about every 15 μs. Refresh cycle timing is shown in Figure 5-31.

Refresh timing is determined by two binary counters (MMC9 print). An address counter decrements through the full range of row addresses ($177–000_8$) during a refresh interval. An interval counter determines the 15 μs period (100 counts) between refresh cycles.

At each refresh, the logic subtracts 1 from the address and sets the interval count to 99. After the interval counter counts down to 0, MMC9 REFRESH REQ is set. This signal activates the required control sequence as soon as the memory is free; that is, when it is not busy doing a write, read, or RPW.

If or when the memory is inactive, the refresh request asserts MMCB REF ADD EN. This signal causes the address counter outputs (a row address) to be transmitted over the ROW/COL lines on the MOS data bus interface. It also asserts all BOARD SEL and WD EN lines to select all the MOS chips in the array. MMCB REF SET RAS then transmits RAS on the interface to refresh the selected row.

While the refresh cycle is executing, including any wait for the completion of a memory access, the interval counter continues to count (wraparound). If the refresh request is still true when the count reaches 31, a refresh error flag (MMC9 REF ERR) is set. Because the most probable reason for the extended delay is a hung memory waiting for write data, the refresh error clears the control logic by forcing the completion of a memory write. No memory data is lost.

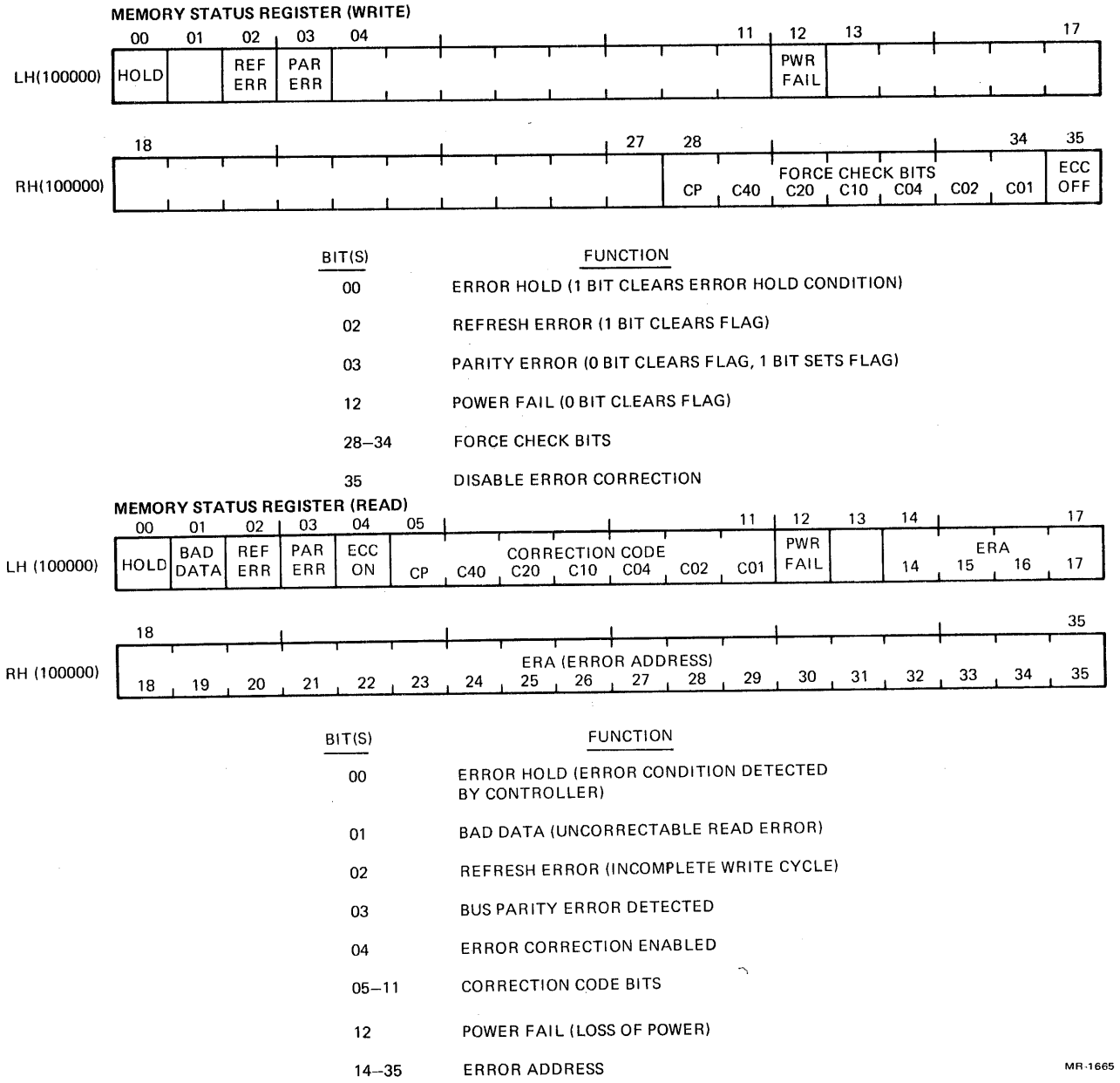


Figure 5-30 Memory Status

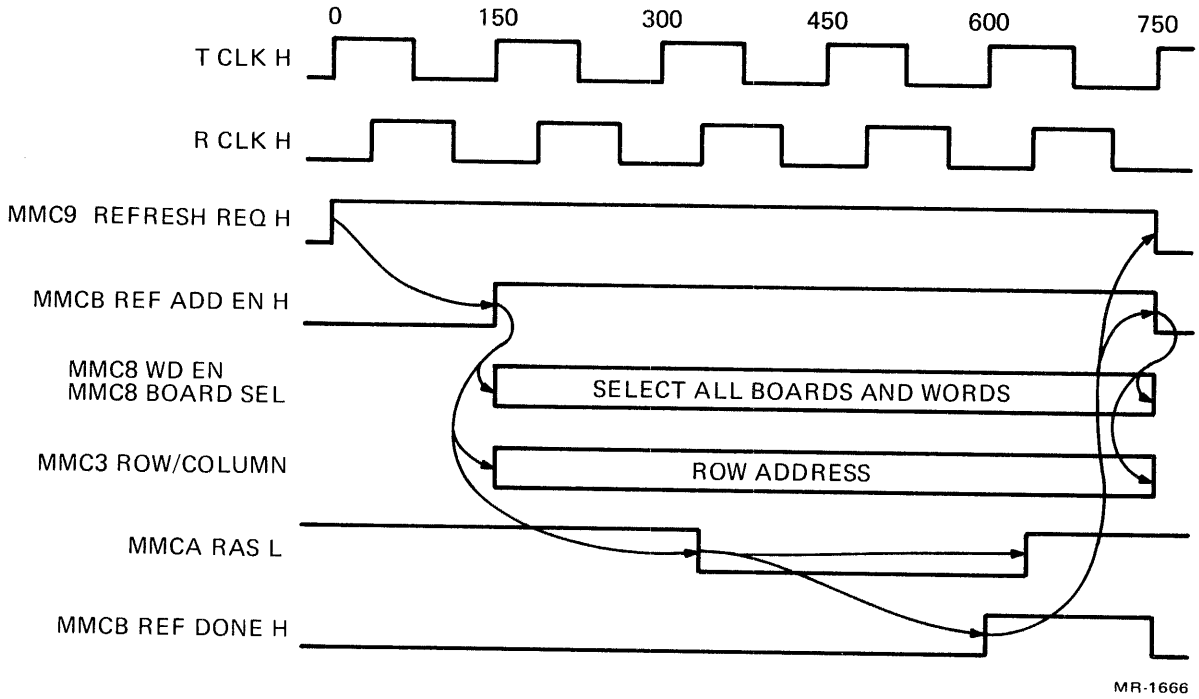


Figure 5-31 Memory Refresh, Timing Diagram

5.8 CONSOLE

The console (CSL) module is the hardware interface to the KS10 system that allows local (CTY) and remote (KLINIK) serial line control by an operator. (Console functions are described in Paragraph 4.13.) The module also performs functions not under operator control, such as generation of the system clocks and arbitration of the KS10 bus. A block diagram of the CSL module is shown in Figure 5-32. The module contains the following major components.

- 8080 Console Processor – Consists of an Intel 8080A microprocessor system and serial line interface (CSL2/CSL9), and a KS10 bus data and control line interface (CSL6–8). The program in the console processor implements all operator-controlled console functions. This program is resident in 8080 ROM and valid at power-up.

- CPU/Bus Control – Consists of the control logic (CSL4) necessary to interface to the KS10 bus, read system status, execute diagnostic functions in the microcontroller, start/stop KS10 program execution, and perform other miscellaneous control functions.

- Systems Clock Control – Consists of the system clock generator (CSL1) and the CPU clock control (CSL5). This circuitry is discussed in Paragraph 5.2.

- KS10 Bus Arbitrator – Arbitrates the use of the KS10 bus by the various KS10 system components including the console itself. Bus arbitration is discussed in Paragraph 5.3.2.

5.8.1 8080 Console Processor

The console processor system consists of an 8080A Microprocessor, a 8224 clock generator and driver, a 8228 bus driver, two 8251 universal synchronous/asynchronous receiver transmitters (USARTs), two 2114 1K × 4 RAMs (1K × 8 total), and four 2716 2K × 8 ROMs (8K × 8 total).

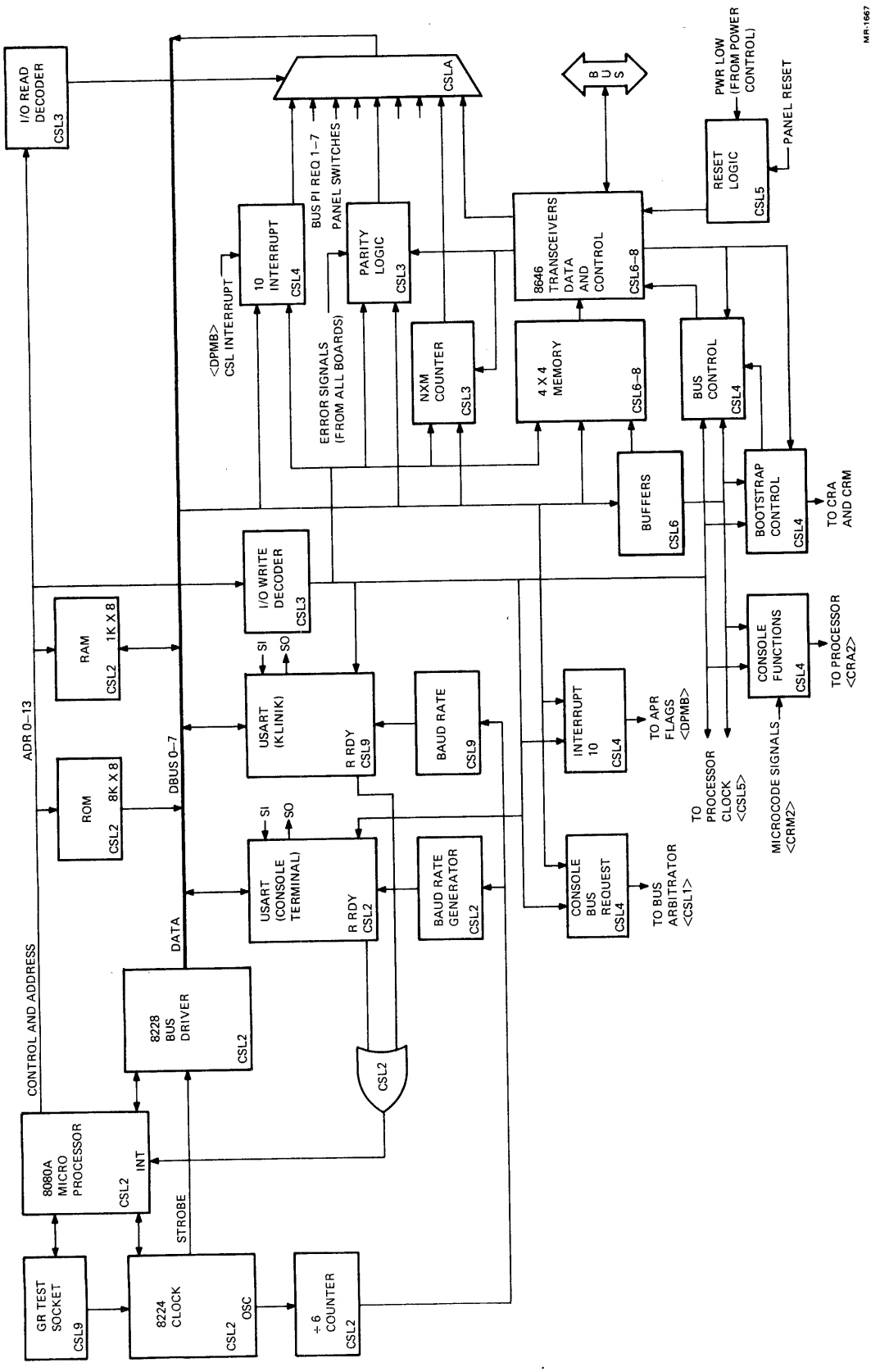


Figure 5-32 Console, Block Diagram

The Intel 8080A is an 8-bit parallel central processor unit. It has two main buses, a data bus and an address bus. The data bus is a bidirectional 3-state bus on which internal state information and data transfers occur. The address bus is a 3-state 16-bit bus over which memory and peripheral device addresses are transmitted. There are also six timing and control outputs and four control inputs on the 8080A.

The timing of the 8080A as used in the KS10 is determined by the 8224 clock generator and driver. The 8224 contains a crystal-controlled oscillator, two high-level drivers, a "divide by nine" counter, and several auxiliary logic functions. Among these auxiliary functions is the generation of the signal RESET. RESET occurs at power-up and is used to set the 8080A stack pointer to location 0. The oscillator circuit derives its basic operating frequency from an external 14.746 MHz series resonant, fundamental crystal. This oscillator is used to provide the 8080A with the two clocks necessary for its operation. These two non-overlapping clock pulses are labeled Phase I and Phase II and are at a frequency of 1.638 MHz. The 8224 also supplies 14.746 MHz clock to a "divide by six" counter which in turn supplies 2.45 MHz to the two baud rate generators.

The 8228 bus driver is a single-chip system controller and bus driver. It generates all the signals required to interface the RAM, ROM, and I/O components to the 8080A. The 8228 consists of three sections: a bidirectional bus driver, a status latch, and a gating array. The bidirectional bus driver provides isolation for the 8080A data bus from memory and I/O devices. The bidirectional bus driver receives control from signals generated by the gating array. The status latch receives "status" information from the 8080A data bus at the beginning of each machine cycle. This "status" information indicates the type of activity that will occur during the cycle. It is stored in the status latch which is connected to the gating array. The gating array generates control signals by gating the outputs of the status latch with signals from the 8080A CPU.

The two 8251 USARTs are used to convert parallel format system data into serial format for transmission to the CTY and KLINIK lines. They also invert serial data from the CTY and KLINIK lines to parallel format for use in the system. An 8251 will delete or insert bits or characters that are unique to the communication system being used. In essence, an 8251 is transparent to the CPU, generating a simple input or output of byte-oriented system data.

5.8.1.1 8080A Timing – With reference to Figure 5-33, the 8080A is supplied with a phase I clock and a phase II clock from the 8224. It is the phase I clock pulse which determines the smallest unit of processing activity known as a state. A state is defined as the interval between two successive positive-going transitions of the phase I clock pulse. Each time the 8080A CPU accesses memory or an I/O port it requires a machine cycle. Each machine cycle consists of three, four, or five states; the time required to fetch and execute an instruction is an instruction cycle.

Every instruction cycle is made up of one to five machine cycles. A full instruction cycle requires anywhere from four to eighteen states, depending on the type of instruction involved. The 8080A uses internal timing logic which takes the clock inputs and uses them to produce a SYNC pulse. The low to high transition of the phase II clock is used to trigger the SYNC pulse. The SYNC pulse is used to identify the beginning of every machine cycle.

Every 8080A machine cycle within an instruction cycle is made up of from three to five active states: T1, T2, T3, T4, T5 or TW. (Refer to Table 5-7.) The number of states being used depends on the instruction being executed. There are at least three states in every machine cycle. The transitions of the phase I and phase II clock set the reference for the events that take place in each state. If the processor has to wait for a response from a peripheral or memory with which it is communicating, then the machine cycle may also contain one or more T_W (wait) states. The processor will enter a wait state (T_W)at the end of T_2, instead of proceeding to T_3. During the three basic states, data is transferred to or from the processor.

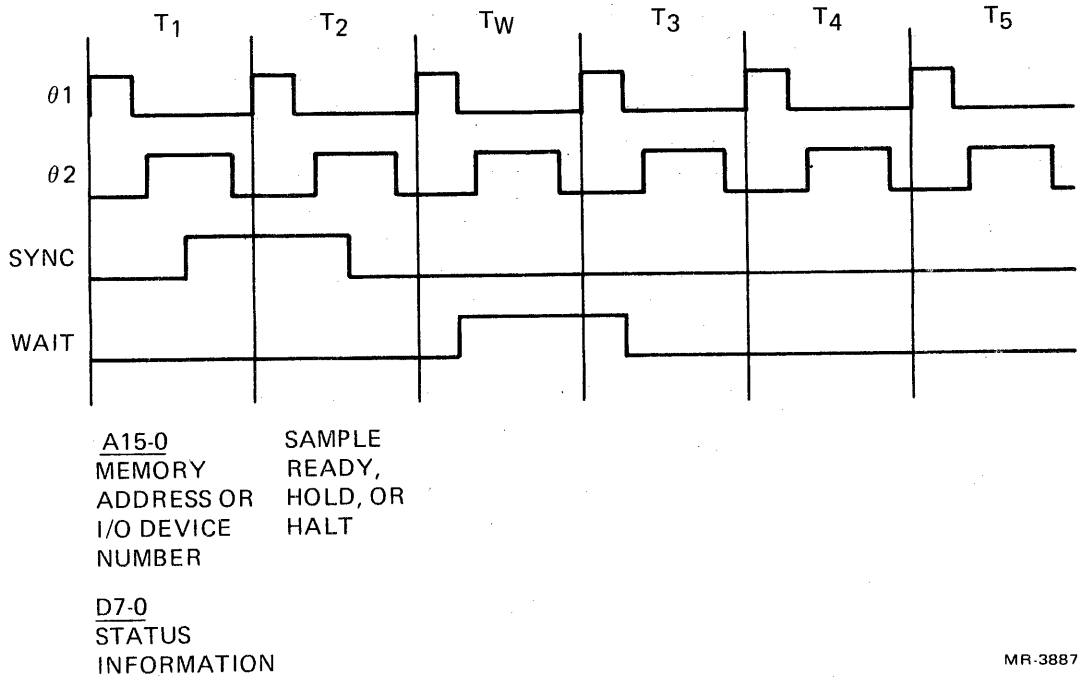


Figure 5-33 8080A, Basic Timing

Table 5-7 8080A State Definitions

State	Associated Activities
T_1	A memory address or I/O device number is placed on the address bus; status information is placed on the data bus.
T_2	The CPU samples the READY and HOLD inputs and checks for halt instruction.
T_W	Processor enters wait state if READY is low or if HALT instruction has been executed.
T_3	An instruction byte, data byte, or interrupt instruction is input to the CPU from the data bus; a data byte is output onto the data bus.
T_4 or T_5	States T_4 and T_5 are available if the execution of a particular instruction requires them; if not, the CPU may skip one or both of them.

It is difficult to generalize after the T_3 state. If the execution of a particular instruction requires them, the processor can use T_4 and T_5, but not all machine cycles require their use. The use of T_4 and T_5 depends on the particular instruction being processed. As soon as the CPU processing activities are over it will stop the machine cycle rather than continuing through the T_4 and T_5 states. Anytime after a T_3, T_4, or T_5, the CPU may exit the machine cycle and proceed directly to T_1 of the next machine cycle.

5.8.1.2 8080A Operation – There are three basic operations that take place when the 8080A processes information. First, at the beginning of every microprocessor machine cycle, the 8080A places status information identifying the use of the cycle on its data bus outputs. At the same time the 8080A sends a sync signal to the 8224 clock generator and driver. The 8224 responds by sending a strobe (STSTB) to the 8228 bus driver. The strobe causes the 8228 to load the status information into a set of latches, thus freeing the data lines for transfers during the cycle. The status information indicates the type of activity in current process. This activity can take four different forms as indicated by the 8228 outputs: PROM read, RAM write, console register read, and console register write. Once the latches have been set, the 8080A will send one of three control signals to the 8228, depending on the operation. These signals are WR (write), DBIN (data bus in), or HLDA (hold acknowledge).

During the second basic operation, the 8080A issues a binary code on the address bus to identify which particular memory location or I/O device will be involved in the current process activity. The 8080A memory comprises 1K of RAM for a stack or other general use, and 8K of ROM (or PROM) that contains the program which the 8080A runs. The I/O devices are made up of a number of registers, mixers, and memories, all of which are accessed by console register reads or writes. The 8080A internally latches the address to its outputs.

During the third basic operation, the 8080A CPU simply sends or receives data to/from the selected memory location or I/O device via the 8228 bus driver.

5.8.1.3 Console Operations – All the operations performed by the 8080A in the console board are functions of the program stored in the 8K of ROM. Communications to or from the KS10 take place via a series of 4 × 4 memories, mixers, and registers. Of the four activities (signals):

PROM read,
RAM write,
Console register read, and
Console register write;

only two are used for reading or writing information to or from the registers.

If the 8080A is going to write a register, the activity code for a console register write is sent to the 8228 bus driver. After sending the activity code, which is latched by the 8228 (output CSL2 CSL REG WRT = 1), the 8080A sends an address and finally data. The activity and address are sent to a pair of decoders. The decoders determine which console register write will take place (for example, CSL3 CSL REG WRT 100). (Table 5-8 shows which address bits are set to produce the individual console register writes.) The data sent by the 8080A in conjunction with the individual console register write signal produces the required control signals or data.

If the 8080A is going to read information, the activity code for a console register read is sent to the 8228 bus driver. After the activity code is latched by the 8228 (output CSL2 CSL REG RD = 1), the 8080A sends an address, and then receives the specified data. As for the write operation, the activity and address are decoded to specify the data to be transferred. In this case, the decoder determines which individual console register read will take place (for example, CSL3 CSL REG RD 100). (Table 5-8 shows which address bits are set to produce the individual console register reads.) The console program performs register read operations in order to read the KS10 bus and the various system status bits.

The 8080A may also access the 8K of ROM by issuing a ROM read activity code to the 8228 bus driver. After the activity code is latched by the 8228 (output CSL2 PROM RD = 1), the 8080A sends an address and then receives the addressed data as for a register read operation. The activity and address are decoded to determine which one of the four ROM chips will be enabled (for example, CSL2 4K). Once the chip has been enabled, the address selects a location within the chip. Table 5-8 indicates which address bits are set to produce the individual ROM selection.

Table 5-8 Console Register Reads and Writes

Notes	RD	WRT	CSL REG	CSL2 ADR: 13 12	11 10 9	8 7 6	5 4 3	2 1 0
	X		0	0* 0*	0* 0* 0*	0* 0 0	0* 0* 0	0 0 0
	X		100			0 1	0	0 0 0
		X	100			0 1	0	0 0 0
		X	101			0 1	0	0 0 1
		X	102			0 1	0	0 1 0
		X	104			0 1	0	1 0 0
		X	106			0 1	0	1 1 0
		X	110			0 1	1	0 0 0
		X	112			0 1	1	0 1 0
		X	114			0 1	1	1 0 0
		X	116			0 1	1	1 1 0
	X		200			1 0	0	0 0 0
		X	200			1 0	0	0 0 0
		X	202			1 0	0	0 1 0
		X	204			1 0	0	1 0 0
		X	205			1 0	0	1 0 1
		X	206			1 0	0	1 1 0
		X	210			1 0	1	0 0 0
		X	212			1 0	1	0 1 0
	X		300			1 1	0	0 0 0
	X		302			1 1	0	0 1 0
USART 1 EN	X		200			1 0	0	0 0 0
USART 2 EN	X		202			1 0	0	0 1 0
USART 1 EN		X	200			1 0	0	0 0 0
USART 2 EN		X	202			1 0	0	0 1 0

	RD	WRT	PROM RAM	CSL2 ADR: 13 12	11 10 9	8 7 6	5 4 3	2 1 0
PROM EN	X		2K	$(00000–37777)_8$				
PROM EN	X		4K	$(04000–07777)_8$				
PROM EN	X		6K	$(10000–13777)_8$				
PROM EN	X		8K	$(14000–17777)_8$				
RAM CHIP EN	X		1K	$(20000–21777)_8$				

Note that only RAM CHIP EN is required to read the RAM and it is asserted by CSL2 ADR 13 and CSL2 8 ENB.

*Indicates that column is all zeros.

In a similar manner, the 8080A may access the 1K RAM. In accessing the RAM there are two activities possible: a RAM write or a RAM read. For the write, the latched activity code in 8228 (CSL2 RAM WRT = 1) enables the RAM to be written, the address (sent next) selects a location to be written into, and the data (sent next) is loaded in the selected address. To read the RAM the only signal required is RAM CHIP ENABLE. RAM CHIP ENABLE is derived from the address (CSL2 ADR 13 and CSL2 8 ENB) following the assertion of the activity code. Table 5-8 indicates which address bits are set to produce the RAM read and write.

The 8251 universal asynchronous receiver/transmitter (USART) is used to communicate to and from the CTY. When a command is typed on the CTY it is received in serial form on the serial input (R DATA) of the 8251 UART. The UART converts the serial data to parallel and raises R RDY, the receiver ready signal. R RDY asserts the 8080A interrupt (INT) input, which informs the CPU that a character is ready.

The 8080A interrupt input is asynchronous, and a request may occur any time during an instruction cycle. Thus, it must be re-clocked by the internal logic of the 8080A to establish a proper correspondence with the driving clock. There is also an internal interrupt latch. Should an interrupt request arrive during the time the interrupt enable is high the next phase 2 clock will set this latch. The latch is set during the last state of the instruction cycle in which the request occurs. This ensures that any instruction in progress will be completed before the interrupt is processed. Before the interrupt is processed the status of the program counter is saved so that data in the counter may be replaced after the interrupt request has been processed.

After the 8251 interrupts the 8080A, the 8080A will initiate a console register read, specifically a console register read 200 (CSL3 CSL REG RD 200). The console register read 200 along with address bit 1 produces the USART enable signal (CSL3 USART 1 EN). It is the combination of these two signals (CSL2 CSL REG RD and CSL3 USART 1 EN) along with address bit 0 that allows the parallel data to be read by the 8080A. Address bit 0 informs the 8251 whether the information is data or control/status.

To transmit data in parallel form to the USART to be converted to serial form for use by the CTY, the 8080A initiates a console register write 200 (CSL3 CSL REG WRT 200). The USART enable signal (CSL3 USART 1 EN) is again asserted, this time by the console write 200 along with address bit 1. Also, similar to the serial read, it is the combination of these two signals (CSL2 CSL REG WRT and CSL3 USART 1 EN) along with address bit 0 that allows the serial data out. Also, address bit 0 informs the 8251 whether the information is data or control/status.

5.8.2 CPU/Bus Control

The 8080A program controls the KS10 bus and CPU with a group of control flip-flops set by the console register write operations. Table 5-9 lists these CPU/bus control signals, their function, and the console register write operation that asserts them.

Table 5-9 CPU/Bus Control Flip-Flops

Control Flip-Flop	Register Write	Description
CSL3 CACHE EN	100	Enables cache operation in DPM.
CSL3 RESET	100	Initializes console and asserts RESET on KS10 bus. Set by MR command.
CSL4 1 MSEC EN	100	Enables operation (service by microcode) of interval timer. Set by TE command.

Table 5-9 CPU/Bus Control Flip-Flops (Cont)

Control Flip-Flop	Register Write	Description
CSL3 PE DETECT	100	Enables system parity errors to stop the CPU clock. Set by PE command.
CSL3 CRM DETECT	100	Enables a CRAM parity error detected in CRA or CRM to stop the CPU clock. Set by PE command.
CSL3 DP PE DETECT	100	Enables a DBus parity error detected in DPE, or a RAM parity error detected in DPM, to stop the CPU clock. Set by PE command.
CSL3 STATE	101	Blinks front panel STATE indicator when microcode is loaded and running and monitor is maintaining "keep alive" dialogue with console.
CSL3 FAULT	101	Lights front panel FAULT indicator when there is a system parity error (CSL3 PE(1) = 1), a memory refresh error (MMC9 REF ERR B = 1), or a boot command failed to start the machine.
CSL3 REMOTE	101	Lights front panel REMOTE indicator if the REMOTE DIAGNOSIS switch is set to ENABLE or if the switch is set to PROTECT and a password has been entered by the operator (PW command).
CSL4 INT 10	116	Signals CPU interrupt by console.
CSL4 CRAM WRITE	204	Enables the diagnostic function decoders on CRA and CRM during diagnostic operations.
CSL4 CRAM ADR LOAD	204	Loads the diagnostic address register in CRA during diagnostic operations.
CSL4 SS MODE	204	
CSL4 DP RESET	204	Initializes DPE and DPM modules. Asserted by MR command.
CSL4 STACK RESET	204	Clears stack counter on CRS (not used).
CSL4 CRA/M RESET	204	Initializes CRA and CRM modules. Asserted by MR command.
CSL4 TRAP EN	205	Enables trap operation by CPU. Asserted by TP command.
CSL4 DIAG 10, 4, 2, and 1	205	Specify diagnostic read/write function in CRA/CRM during diagnostic operations.

Table 5-9 CPU/Bus Control Flip-Flops (Cont)

Control Flip-Flop	Register Write	Description
CSL4 CRA T CLK ENB (CSL4 CRA T CLK ENABLE)	210	Enables the 8646s on CRA to transmit data to the console during diagnostic operations.
CSL4 XMIT ADR	210	Causes the command/address or diagnostic data in 4 × 4 memory location 0 to be transmitted on the KS10 bus. Set during deposit/examine commands nor during diagnostic operations.
CSL4 XMIT DATA	210	Causes the data in 4 × 4 memory location 1 to be transmitted on the KS10 bus. Set during deposit commands or during an instruction register read operation.
CSL4 LATCH DATA	210	Set during examine commands to latch 8646s (if CSL4 CLOSE LATCHES = 1) when data is received on KS10 bus.
CSL4 CLOSE LATCHES	210	Allows data to be latched in 8646s and inhibits read of instruction register during bus operations by console.
CSL4 BUS REQ (CSL4 CONSOLE REQ (1))	210	Requests KS10 bus.
CSL3 MEM	210	Enable reception of DATA CYCLE on KS10 bus. When 0, allows check of nonexistent memory (NXM) circuit.
CSL4 R CLK ENB (CSL4 CRA R CLK)	210	Latches the 8646s in CRA during diagnostic operations.
CSL4 RUN	212	In CRA, causes the microcode to enter the halt loop.
CSL4 EXECUTE	212	In CRA, causes the microcode to execute the instruction in the console's instruction register.
CSL4 CONTINUE	212	In CRA, causes the microcode to exit from the halt loop and execute one or more system-level instructions.

5.8.2.1 KS10 Bus Functions – The CPU/bus control logic is active during the KS10 bus functions initiated by the console commands. It is also active in response to the read of the console's instruction register by the CPU. The console executes KS10 bus I/O, memory, and diagnostic read/write operations in implementing the various console commands and switch functions. It may also simply examine and deposit (transmit on) the bus with no transfer of data to/from another bus device. To perform these KS10 bus operations, the console program may perform the following functions.

1. Store information in the 4 × 4 memories connecting to the 8646 bus transceivers, and then enable the 8646s to transmit that information on the bus. Two memory locations are used so that two sets of information can be transmitted on successive bus cycles (that is, command address cycle followed by a data cycle).

2. Latch bus data in the 8646s after receiving a bus data strobe (that is, I/O DATA CYCLE, DATA CYCLE).

3. Latch bus data in the 8646s after enabling the transfer of diagnostic data from CRA.

4. Latch bus data in the CRA 8646s during the transfer of diagnostic data from CSL.

5. Latch bus data in the 8646s unconditionally (that is, EB and DB commands).

As stated previously, control is by the console program setting one or more of the control flip-flops listed in Table 5-9. This is indicated in Figure 5-34, which also shows the basic timing. The control flip-flops set for the various bus functions executed by CSL are shown in Table 5-10.

Table 5-10 Control Flip-Flops for CSL Bus Functions

Bus Function	BUS REQ	XMIT ADR	XMIT DATA	LATCH DATA	CLOSE LATCHES	MEM	CRA T CLK	CRA R CLK
I/O Register Write	X	X	X					
I/O Register Read	X	X		X	X			
Memory Write	X	X	X					
Memory Read	X	X		X	X	X		
Deposit Bus	X	X			X			
Examine Bus					X			
Diagnostic Write	X	X						X
Diagnostic Read	X						X	
Diagnostic Address Write	X	X						X

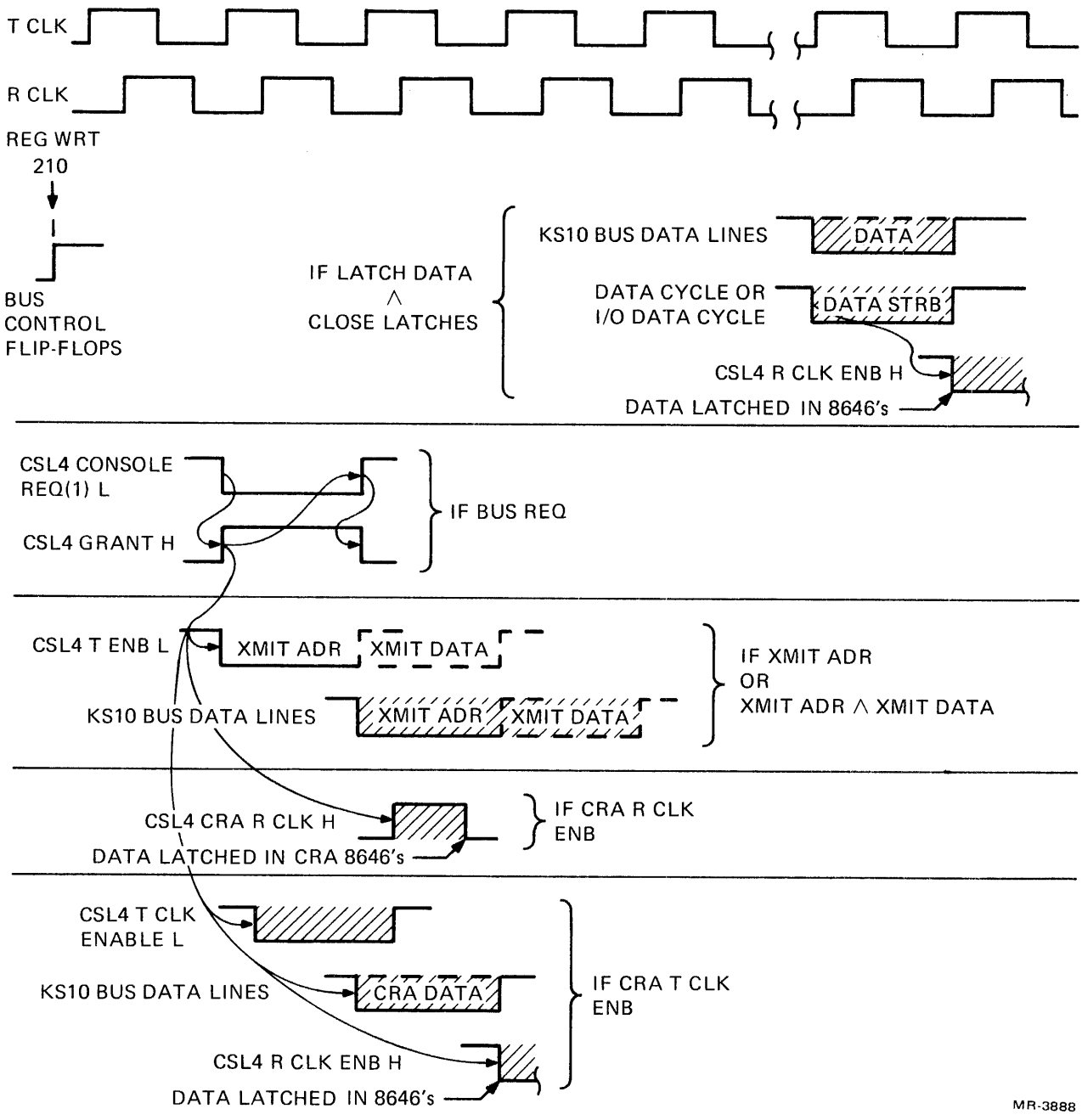


Figure 5-34 Console Bus Functions, Basic Timing

As indicated, the bus is requested (CSL4 BUS REQ = 1) for all operations except a bus examine. This delays these operations until the bus is granted by the bus arbitrator (also on the CSL module). The bus is not requested for a bus examine because the operation, the result of an (examine bus) EB command, does not use the bus. An EB command only latches the 8646s to read the state of the bus (CSL4 CLOSE LATCHES = 1).

The console transmits on the bus (CSL4 XMIT ADR = 1 or CSL4 XMIT DATA = 1) for all operations except a diagnostic read and of course the bus examine. The I/O and memory read/write operations all require transmission of a command/address, and the console program loads the 4 × 4 memories (location 0) and sets XMIT ADR, and thus CSL4 T ENB, to transmit this information on the bus. (The COM/ADR CYCLE control line input is also loaded in the 4 × 4 memory location like a data bit and transmitted at the same time).

For the deposit bus and the diagnostic address load and diagnostic write operations, no command/address is transmitted. But the console program loads the bus or diagnostic data into the same 4 × 4 memory locations and again sets XMIT ADR to transmit the information on the bus. (The deposit bus operation also unlatches and latches the 8646s, after which the console program compares the information transmitted to the information latched.)

For the I/O and memory write operations, I/O or memory data is transmitted in addition to the command/address. To do this, the console program also loads the second 4 × 4 memory location that is used (location 1), and sets both XMIT ADR and XMIT DATA. The XMIT DATA flip-flop asserts CSL4 DATA CYCLE during the second bus cycle to read the second 4 × 4 memory location and (like XMIT ADR) assert CSL4 T ENB to transmit the data and a DATA CYCLE data strobe on the bus. (DATA CYCLE is asserted by a bit loaded in the 4 × 4 memory, similar to the generation of COM/ADR CYCLE.)

When reading data transmitted from other devices, such as for the I/O or memory read operations, the console enables bus data to be latched as during a bus examine (CSL4 CLOSE LATCHES = 1) but not until the appropriate data strobe has been received (CSL4 LATCH DATA = 1). The data strobe, either DATA CYCLE or I/O DATA CYCLE) first unlatches and then latches the 8646s by setting and then clearing CSL4 R CLK ENB. During a memory read, the console may disable the detection of DATA CYCLE (CSL4 MEM = 0) in order to check the NXM error circuitry.

Although the data line portion of the KS10 bus is used for diagnostic operations, a bus data strobe such as DATA CYCLE is not transmitted along with diagnostic data by either the console or the microcontroller. As a result, the console program sets a control flip-flop (CSL4 CRA R CLK = 1) to assert CSL4 CRA R CLK during a diagnostic write or diagnostic address load operation; and it sets another control flip-flop (CSL4 CRA T CLK) to assert CSL4 CRA T CLK ENABLE during a diagnostic read. The CRA R CLK signal is used by the microcontroller to receive diagnostic data from the console. The CRA T CLK ENABLE signal is used by the microcontroller to transmit diagnostic data to the console.

5.8.2.2 Instruction Register Read Operation – When a KS10 system-level instruction is to be executed from the console with the EX (execute) command, the console program loads the specified instruction in one of the 4 × 4 memory locations (location 1) and asserts CPU control lines CSL4 EXECUTE and CSL4 CONTINUE. The CPU then fetches the instruction over the KS10 bus and executes it.

The CPU also fetches and executes an instruction, this time a JRST, when the KS10 program is started from the console with an ST (start) command. As for the execute operation, the console program loads the instruction in a 4 × 4 memory location (again location 1) and asserts CSL4 EXECUTE and CSL4 CONTINUE. It also asserts CSL4 RUN to allow the program to continue from the starting address specified by the JRST.

The CPU fetches the instruction in the instruction register by means of an I/O read operation. The register address transmitted by the CPU is 200000_8. (The console CTL number is 0.) When received by the console module, the address sets CSL4 STATUS RD which asserts CSL4 T ENB and CSL4 DATA CYCLE as when data is transmitted under the control of the console program (that is, XMIT DATA = 1). The instruction stored in the 4 × 4 memories is then transmitted on the bus, together with data strobe I/O DATA CYCLE, and collected by the CPU.

5.8.3 Console Program

The console program, which is stored in the 8K of ROM, runs continuously in the 8080A. Basic operation of the program is shown in Figure 5-35. The program begins with the initialization sequence which occurs during power-up and restart. A number of activities take place such as a KS10 reset, 8080 PROM checksum, enabling parity detection, loading default constants, enabling 8080A interrupts, and starting the auto-boot sequence. A description of the initialization sequence can be found in 5.8.3.1. The bootstrap process is described in Paragraph 5.8.3.3.

Following initialization the program clears the end-of-line flag, the current error code, and other flags. It then moves into the null job while waiting for a hardware interrupt from one of the two USARTs. In the null job, the program enters a state loop that continually monitors the front panel boot button, the KLINIK carrier, the parity error flags, the run flip-flops, the MOS memory refresh error flag, user mode, and the end-of-line flag for any changes in state. If such a change occurs the program will service it and then return to the null state loop. For example, if a parity error is detected, the program branches to a subroutine that determines if the error is recoverable, whether it is a hard or soft error, and what action should be taken in each case. Upon completion of the subroutine (if the error was not fatal) the program returns to the null state loop.

The major portion of the console program is devoted to processing commands. The processing of a command begins with a hardware interrupt which is generated by the associated USART whenever a character is entered over the CTY or KLINIK lines. The first step in the interrupt sequence is to push the current 8080A CPU status onto the 8080A stack. This part of the program is known as the interrupt handler. The interrupt handler makes a number of decisions, one of which is to determine if the processor is in user mode or console mode. If in user mode, the program moves to the 8080A to KS10 character service routine. The 8080A to KS10 character service routine writes the character in KS10 memory and then interrupts the KS10 CPU (CSL4 INT 10 = 1) so that the monitor may process the information.

If the processor is not in user mode (in console mode) the character is part of a console command. If not an end of line (EOL) character, it is stored in a buffer together with characters in the command entered previously. If it is an end of line character (that is, a carriage return), indicating that all characters in the command have been entered, the program sets the EOL flag and returns to the null job.

Once back in the null job with the EOL flag set, the program will branch to the command decode and dispatch list. At this point the characters are compared to the command list, and checked for validity. If there is a match, the console program executes the command as discussed in Paragraph 5.8.3.2. If there is no match, the program prints an error message and returns to the null state loop.

Another step in the null job is to check if the KS10 CPU is interrupting the 8080A (DPM CSL INTERRUPT = 1). Interrupts occur during the keep-alive dialogue with the KS10 CPU, and when the CPU has a character to transfer to the USART(s) and onto the serial line(s).

5.8.3.1 Power-Up/Initialization – KS10 initialization begins with the 8080A RESET generated by the 8224 clock generator and drive for the 8080A CPU. An external RC network is connected to the RESIN input. As the power supply comes up to full voltage, a sequence is started in the 8224. The slow transition of the power supply rise is sensed by an internal Schmitt trigger. This circuit takes the slow

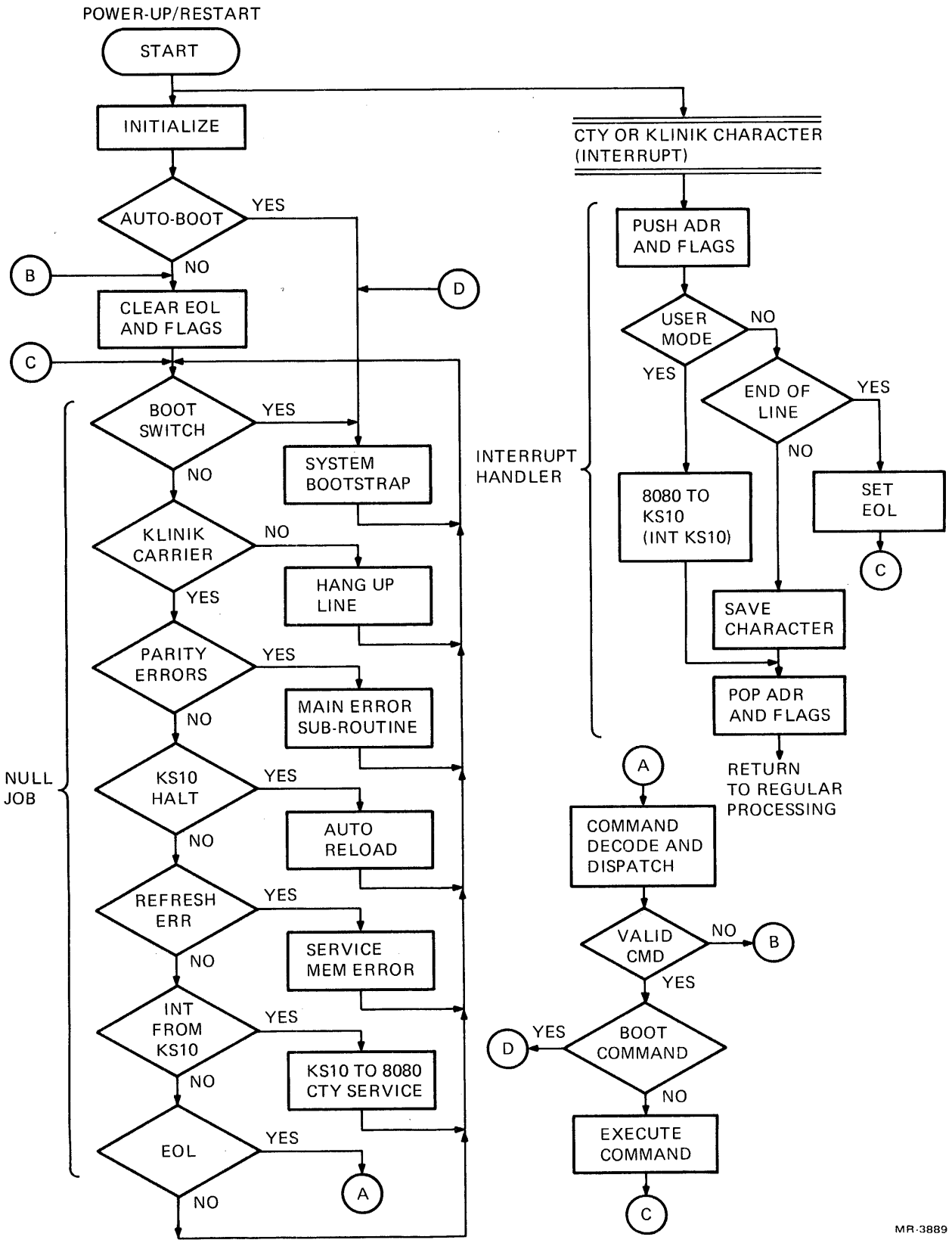


Figure 5-35 Console Program, Basic Operation

transition and converts it into a clean, fast edge when its input level rises to a predetermined value. The output of the Schmitt trigger is connected to a D-type flip-flop. The flip-flop is synchronously reset and an active high level is presented to the 8080A at its RESET input. This RESET restores the processor's internal program counter to zero and the program in the ROM begins immediately. Program operation during power-up and system initialization is shown in Figure 5-36.

The console program begins with the setting of the 8080A stack pointer. The stack pointer is loaded with the last RAM address. Once this is accomplished, a brief procedure for initializing the KS10 is started.

Initializing the KS10 starts with clearing CSL4 RUN, CSL4 EXECUTE, and CSL4 CONTINUE. (These flip-flops must be individually cleared because they are not cleared by a KS10 reset signal.) Next, KS10 reset signals CSL DP RESET and then CSL4 CRAM RESET are asserted. The first resets the processor's data path and the second resets the processor's microcontroller. KS10 BUS RESET is then generated to initialize the rest of the KS10 system. Finally, the ten interrupt flip-flop (CSL4 INT 10) is cleared.

Following the KS10 reset sequence, the console program enters console mode, initializes the 8080 USARTs, and performs an 8080 PROM checksum. The program computes the checksum for each of the 2K 8080 PROM pieces. If the PROM checksum fails, the message ?CHK will be printed along with the failing PROM number (1–4).

The loading of the default constants into the 8080 RAM is next. The default magtape UBA number is 3 and the default disk UBA number is 1. The initial default MTA RH base address is 772440_8 and the initial default DSK RH base address is 776700_8.

Next, the microcode version and the ID number are printed. Following this, an examine bus is performed to determine that the KS10 bus data lines are not asserted. If the data lines are not all 0s, indicating a machine malfunction, the message "?BUS" is printed. The 8080A interrupts are then enabled and a check is made to see if the KS10 memory has battery backup. The console program checks for battery backup by reading the memory status register. If there is battery backup and the contents of memory (the system monitor) are preserved, the console program begins a bootstrap operation that loads only the KS10 microcode. If there is no battery backup, the console program initiates the auto-boot sequence to load both the microcode and the system monitor. At this point the initialization sequence is complete.

NOTE
The KS10 does not currently employ battery backup.

5.8.3.2 Console Commands – Figure 5-37 lists the console commands together with the associated signals and KS10 bus operations required for each. The figure also indicates the order in which the signal and bus operations are generated. For example, the DC (deposit CRAM) command will first generate a CRAM RESET, followed by a CRAM address load operation, and then a diagnostic write operation.

Other commands are completely internal to the 8080 processing system and no KS10 bus operations take place. An example of an internal command is the EB (examine bus) command. This command reads the bus but does not initiate a KS10 bus operation.

Note that the commands are grouped into 19 functional command types. For example, the LC (load CRAM) command is grouped with the load commands and the DC (deposit CRAM) command with the deposit commands. Also note that in some locations on the table a 0 replaces the X. This indicates that the signal is cleared instead of being set.

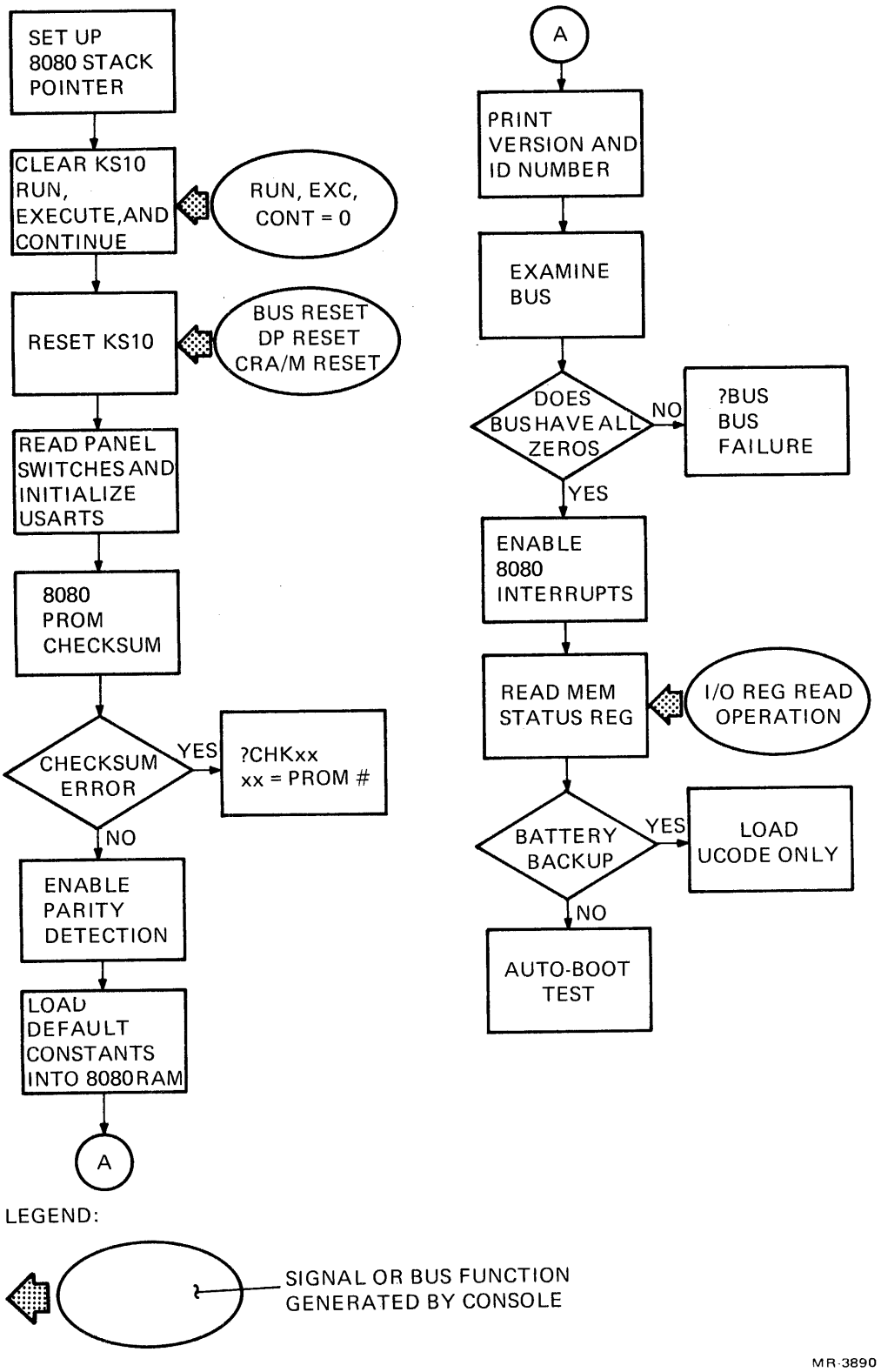


Figure 5-36 Power-Up and Initialization Sequence

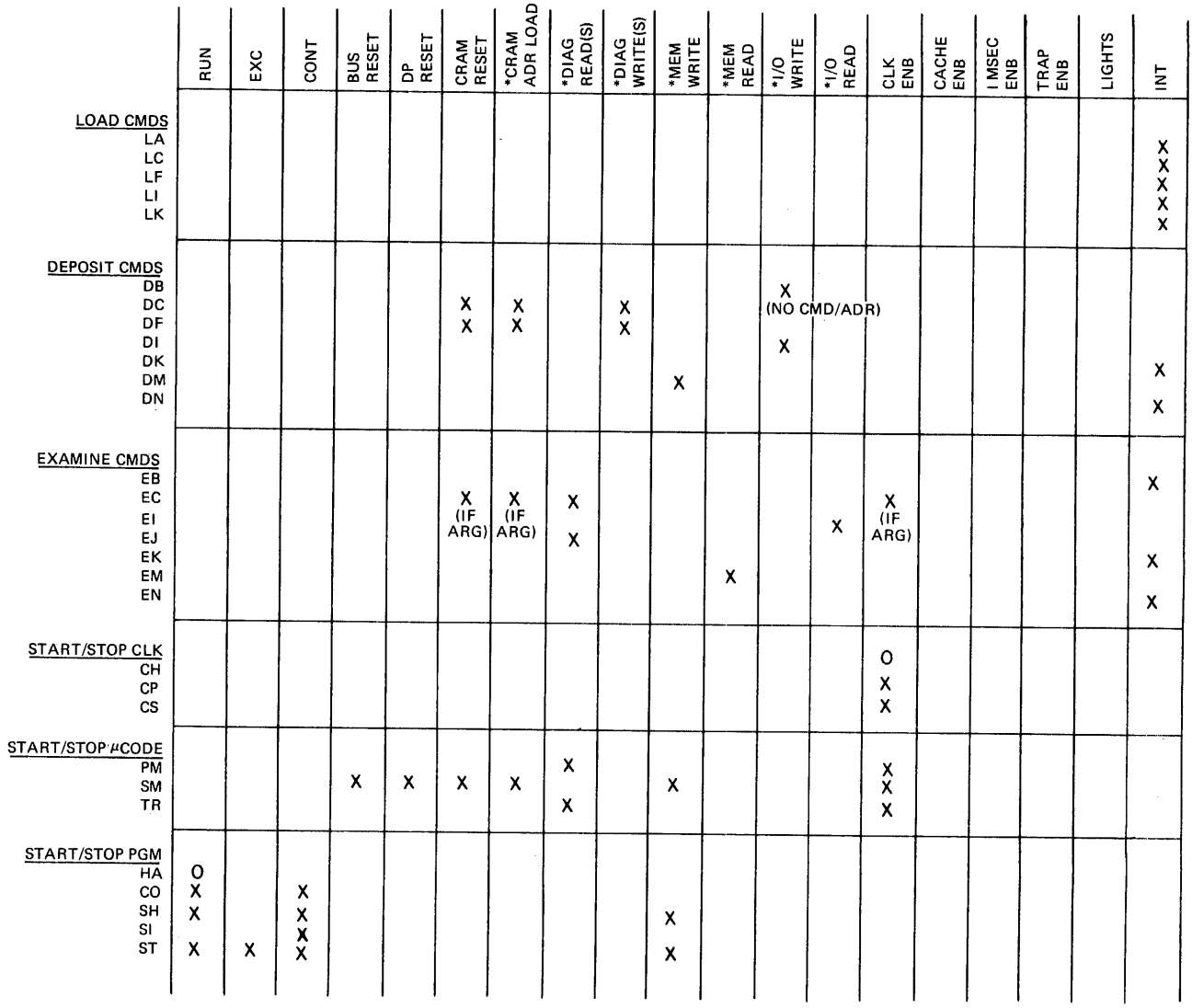


Figure 5-37 Signals and KS10 Bus Operation Initiated by Console Commands (Sheet 1 of 2)

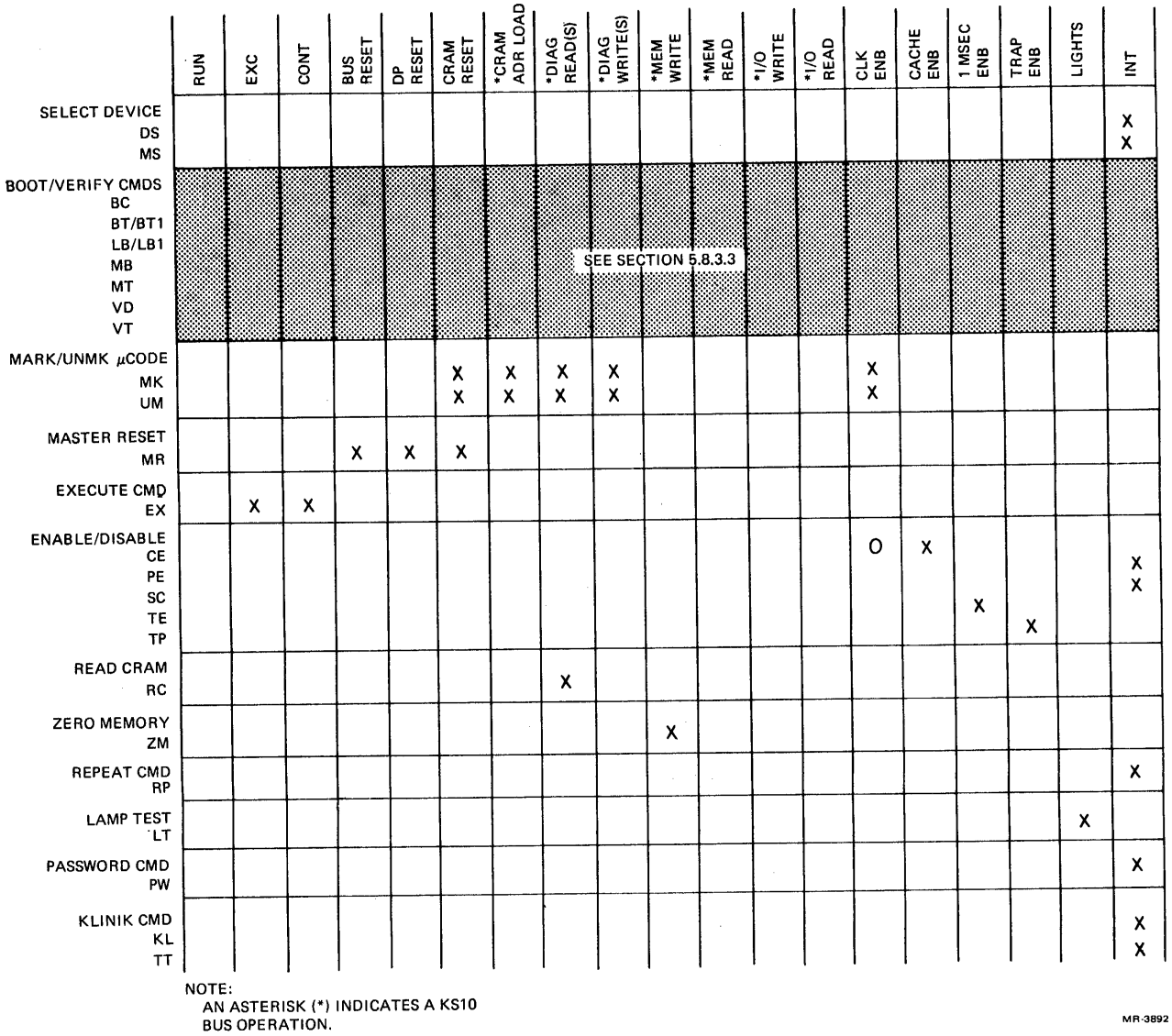


Figure 5-37 Signals and KS10 Bus Operation Initiated by Console Commands (Sheet 2 of 2)

5.8.3.3 System Bootstrap – The bootstrap sequence executed by the console program is initiated by the front panel BOOT switch, by the bootstrap commands (BT, MT, etc.), or occurs automatically 30 seconds after power-up or system reset. System bootstrap consists of the following four basic operations occurring in the sequence shown in Figure 5-38.

1. Microcode to memory
2. Microcode to CRAM
3. Boot program to memory
4. Start CPU

The first of these operations, the transferring of microcode to memory, is an NPR transfer from the disk or tape directly to memory (via the UBA). The microcode is transferred one page at a time. For the bootstrap from disk, the home block and file pointers are read prior to reading the first page of microcode. The home block is read to obtain the disk address of the file pointers; the file pointers are read to obtain the disk address of the microcode.

The second basic operation is the transfer of the microcode loaded in KS10 memory to the CRAM. After each page of microcode is loaded in memory, the console program reads one 36-bit memory location after the other into its data buffer and (after each memory read operation) transfers the data 12 bits at a time to the CRAM by means of diagnostic write functions. When all words in the page of microcode in memory have been loaded in CRAM, a new page is loaded in memory from disk or tape and the CRAM load operation repeats. The transfers of microcode to memory and then to CRAM continue until all 2K 96-bit CRAM locations have been loaded.

The third basic operation in the system bootstrap is reading of the boot program from disk or tape into memory. (This is the first operation performed for the LB and MB commands which do not load microcode, although the MB command must perform a skip over the microcode which precedes the boot program on tape.) When reading from disk, the home block and file pointers are read first as when reading microcode from disk.

The last basic operation that occurs during system bootstrap is starting the CPU; that is, starting execution of the boot program now loaded in KS10 memory. (The program is started at location 1000.) The console program switches to user mode when starting the CPU, and the operator may then bring in the system monitor and initialize the system for normal operation.

Detailed flow diagrams for the disk and tape bootstrap operations described above are given in Figures 5-39 and 5-40. Note that the console program performs several error checks during a bootstrap. Error printouts have the format ?BT XXX YYY where XXX and YYY specify the type of error. Error code definitions are given in Table 5-11.

During the verify commands, console program operation is very similar to the bootstrap sequence. Microcode is read from disk (VD command) or tape (VT command) into KS10 memory, and then read from memory into the console's data buffer. However, the console reads the CRAM (not loads the CRAM as during a bootstrap) with a series of diagnostic read functions, comparing the microcode already loaded in the machine to the microcode stored in memory. Every CRAM location is compared, thus verifying that the microcode in the machine matches that stored on the disk or the tape.

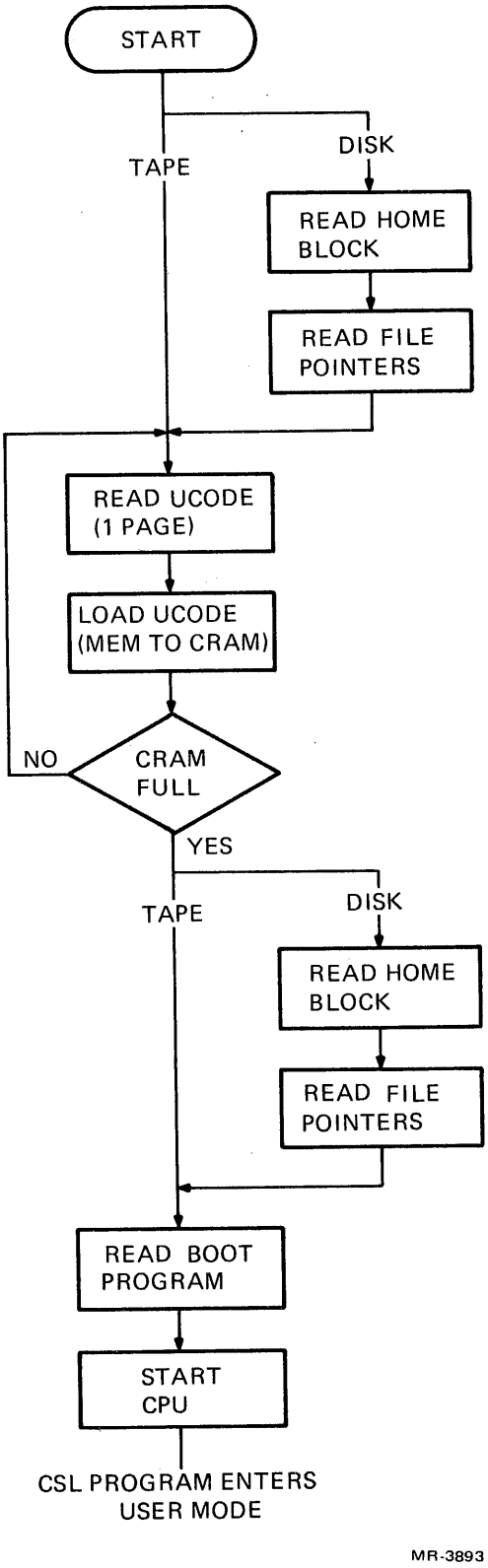


Figure 5-38 System Bootstrap, Basic Operation

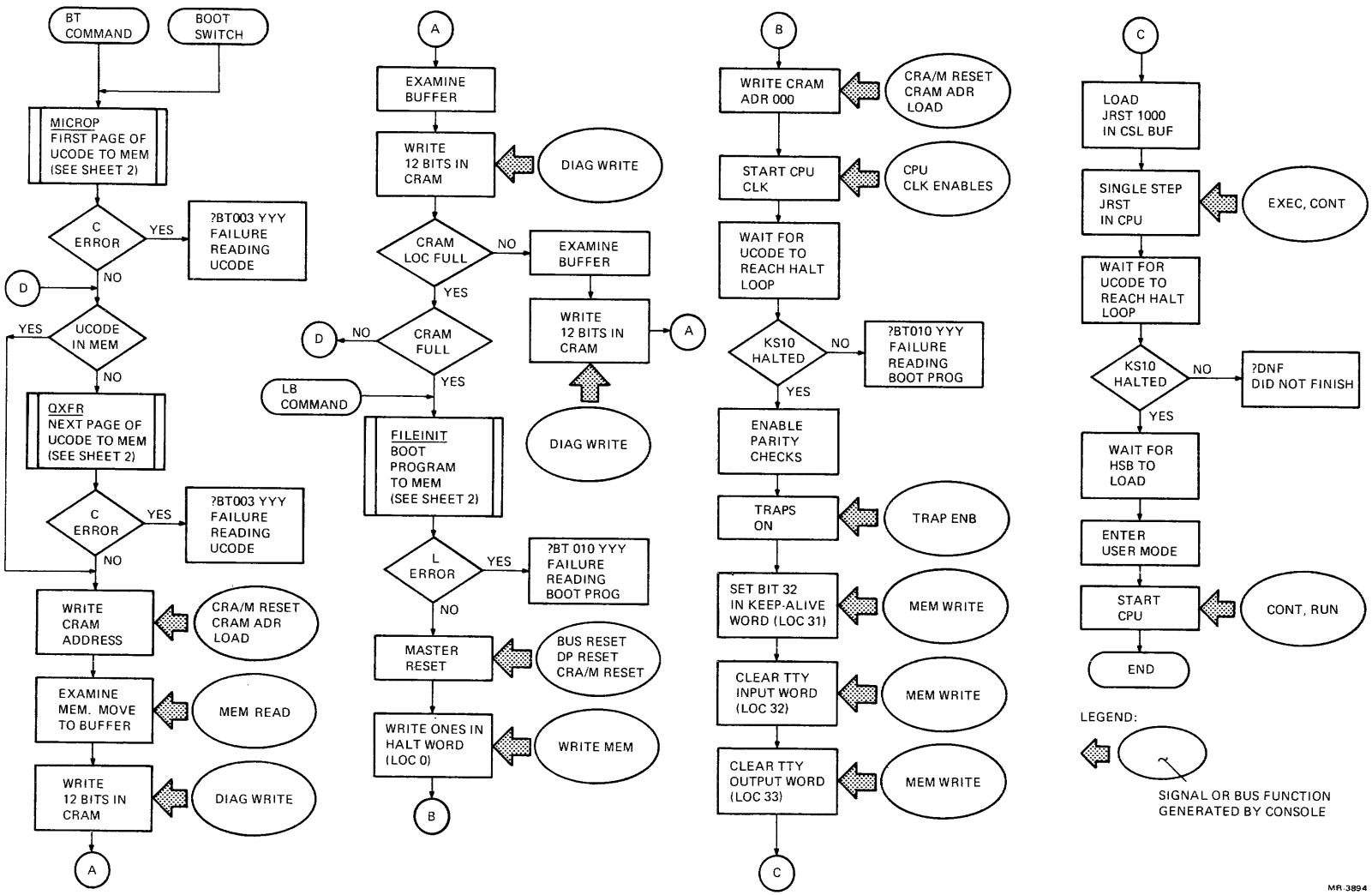


Figure 5-39 Bootstrap from Disk, Detailed Operation (Sheet 1 of 2)

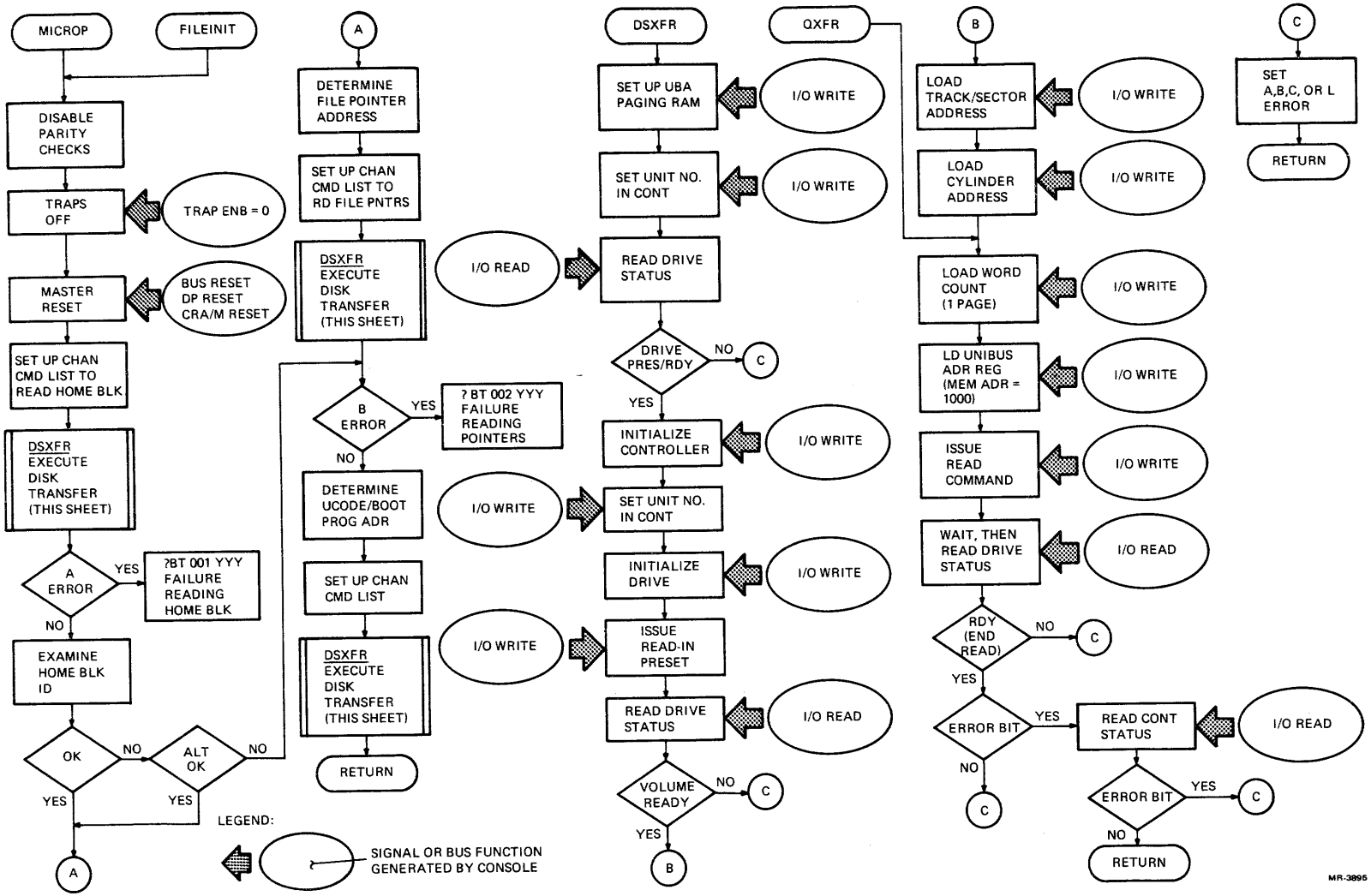


Figure 5-39 Bootstrap from Disk, Detailed Operation (Sheet 2 of 2)

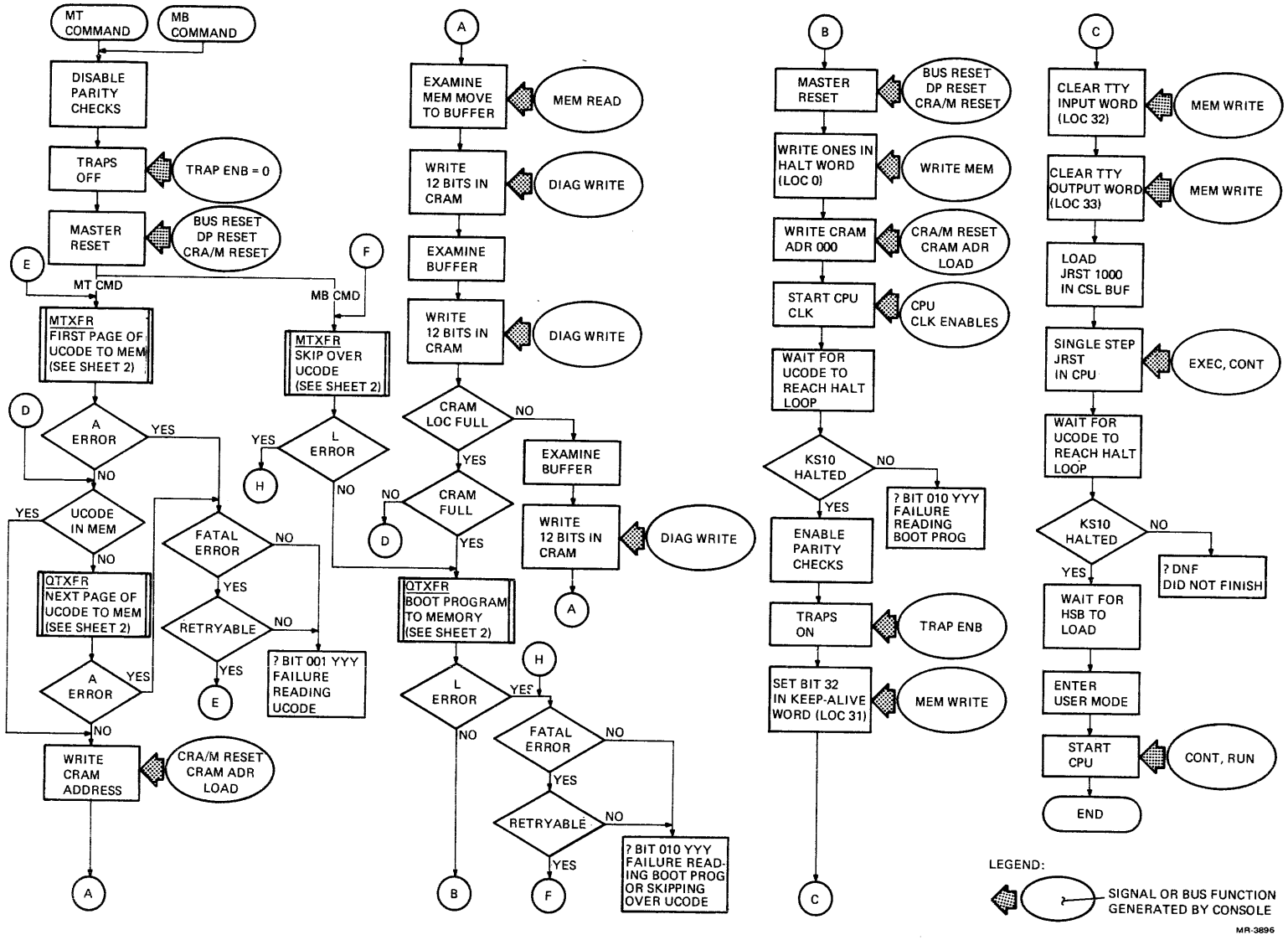


Figure 5-40 Boostrap from Tape, Detailed Operation (Sheet 1 of 2)

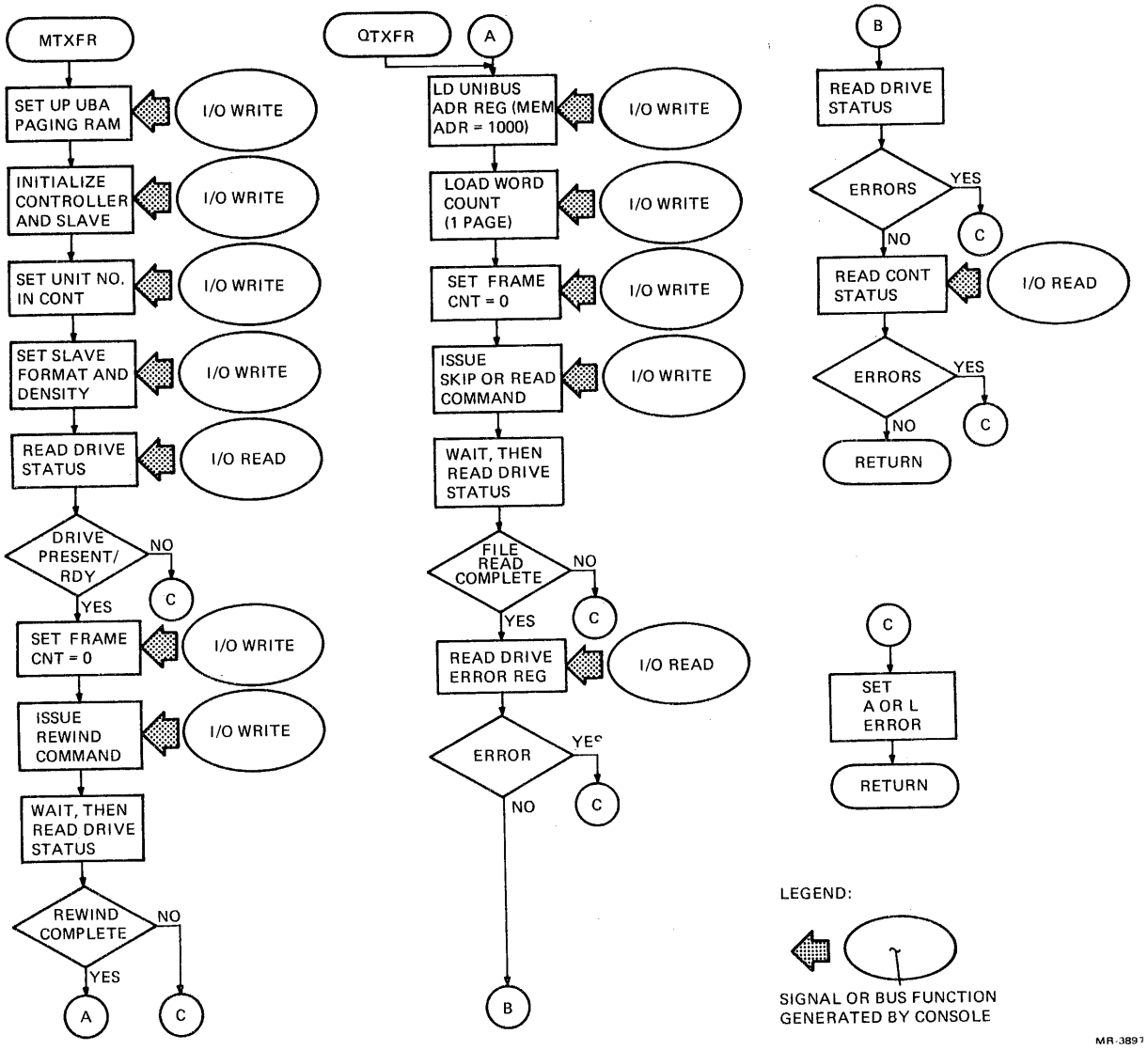


Figure 5-40 Boostrap from Tape, Detailed Operation (Sheet 2 of 2)

Table 5-11 Error Printouts During System Bootstrap

Printout	Definition
XXX = 001 (A Error)	Disk – Error encountered while trying to read the home blocks. Tape – Error encountered while trying to read the first page of microcode from tape.
XXX = 002 (B Error)	Error encountered while trying to read the page of pointers which make up the 8080A file system.
XXX = 003 (C Error)	Error encountered while trying to read a page of microcode.
XXX = 004 (D Error)	Microcode failed to start.
XXX = 010 (L Error)	Error encountered while trying to read the boot program.
YYY	Lower eight bits in the 8080A address of the failing "channel command list" operation.

5.9 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 5-41. 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.

5.9.1 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

5.9.1.1 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 5-42. Correspondence to the Unibus address is indicated.

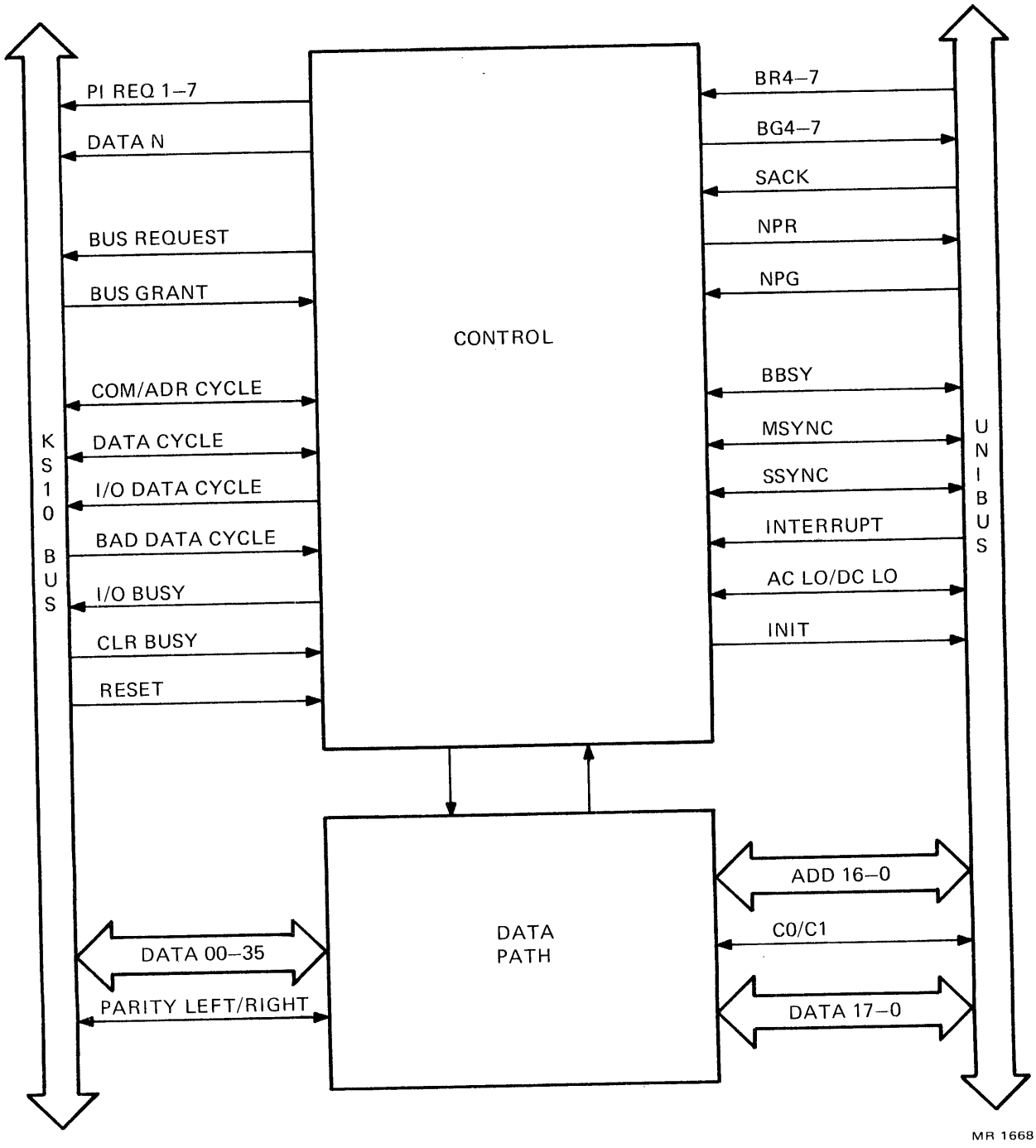
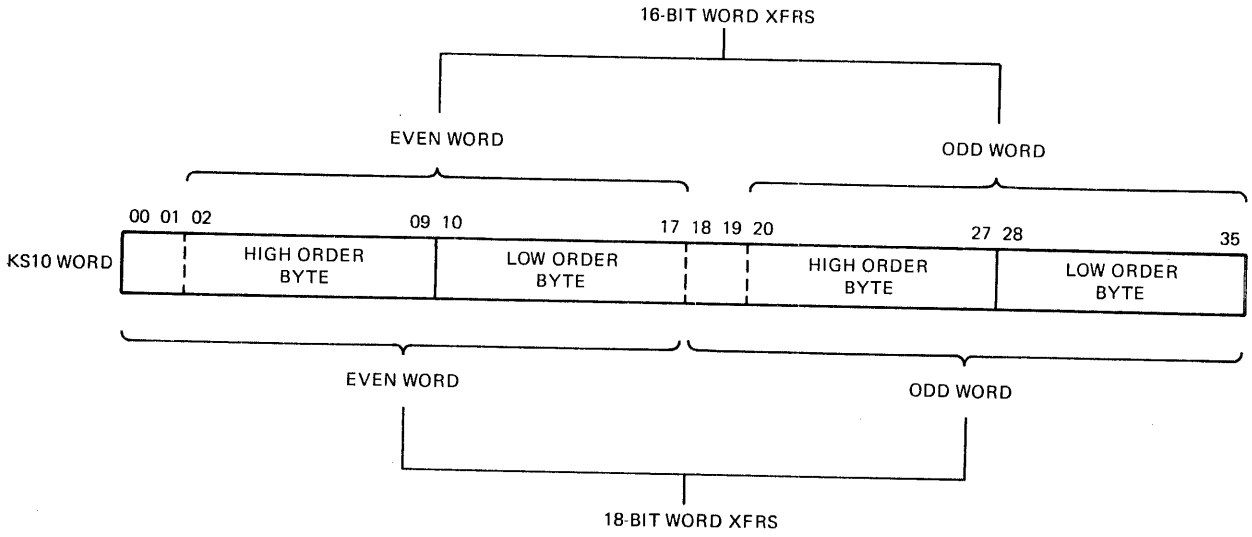


Figure 5-41 UBA, Simplified Block Diagram



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 5-42 Unibus Data Positioning Within KS10 Word

Data flow for the NPR write to memory operation (DATO or DATOB by device) and the NPR read from memory operation (i.e., DATI by device) are shown in Figures 5-43 and 5-44. 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 5.9.2.

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.

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 such as tape. The read reverse mode is provided mainly for diagnostic program use; the system monitor does not use it. As for fast mode, read reverse mode is set by loading a bit in the paging RAM as specified in Paragraph 5.9.2.

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 operations 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 5-43), 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 5-43), 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.

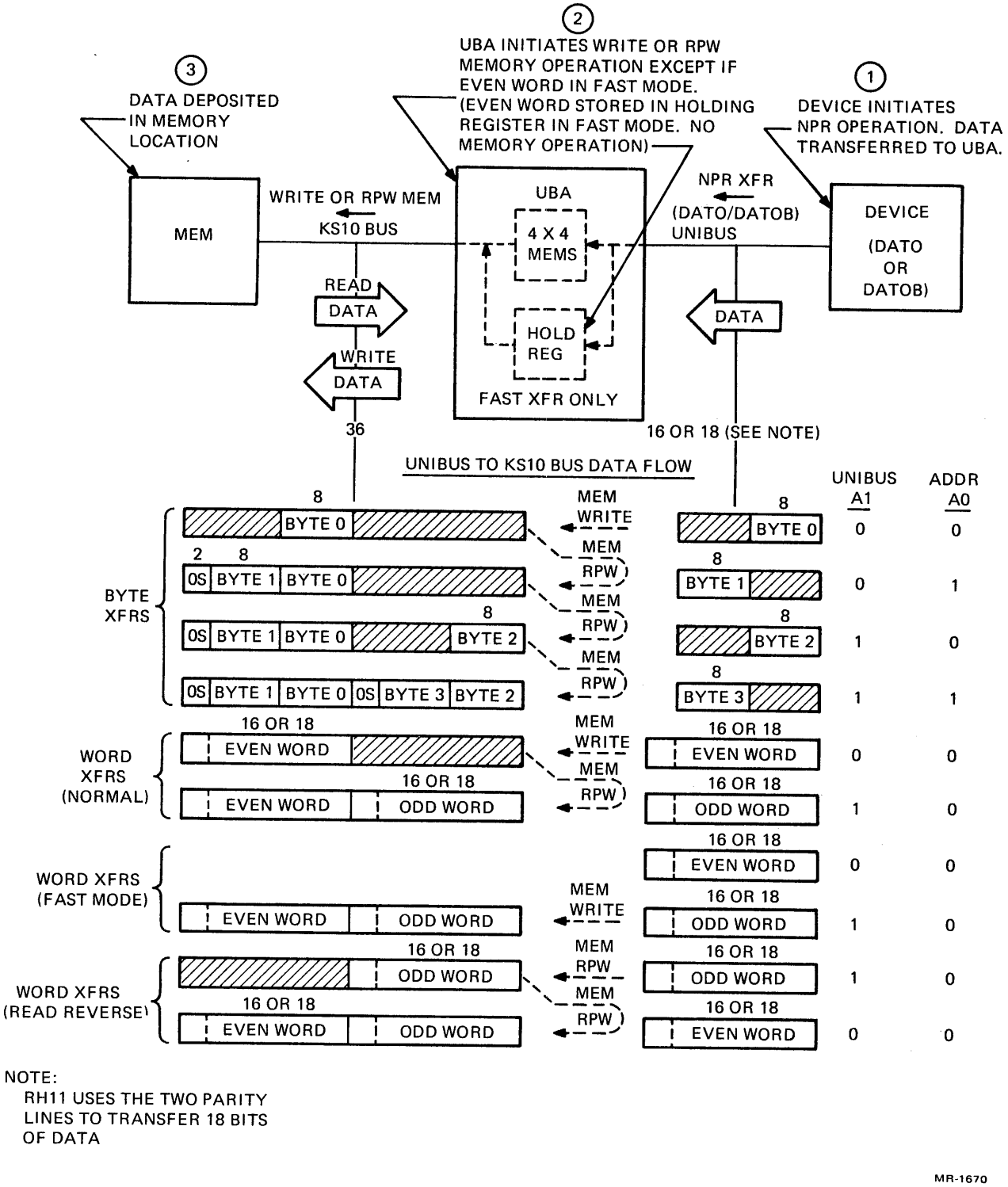


Figure 5-43 NPR Write (to Memory), Data Flow

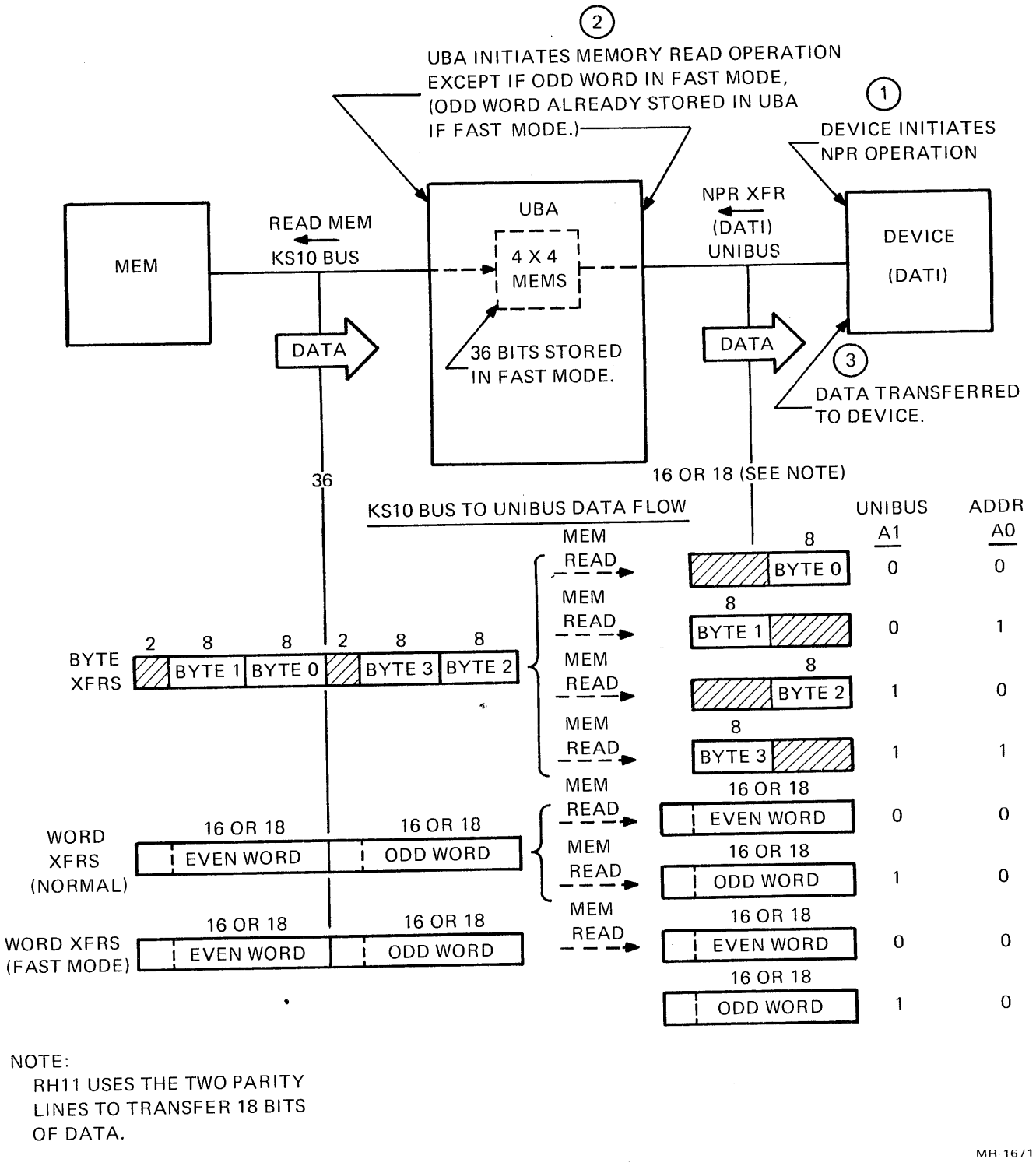


Figure 5-44 NPR Read (from Memory), Data Flow

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.

5.9.1.2 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 5.9.2.

Data flow for both I/O register write and read operations is shown in Figures 5-45 and 5-46. I/O register data transfers are initiated by the CPU as result of the external instruction set (Appendix A). Both word and byte instructions can be executed. I/O register data transfers are also initiated by the console in response to deposit and examine I/O 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. This causes I/O register data to be read from, or written into, KS10 memory. This maintenance feature, called wraparound mode, is discussed in Paragraph 5.9.7.

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.

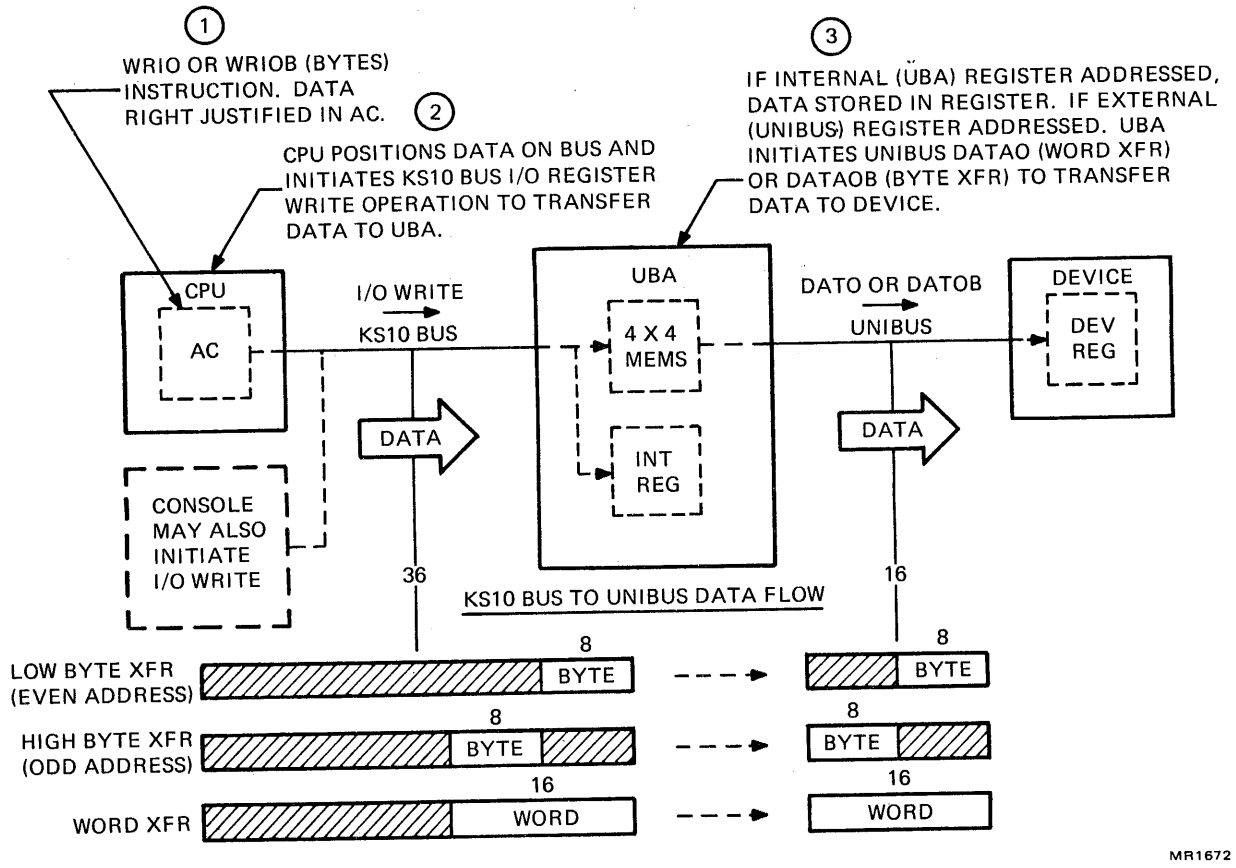


Figure 5-45 I/O Write, Data Flow

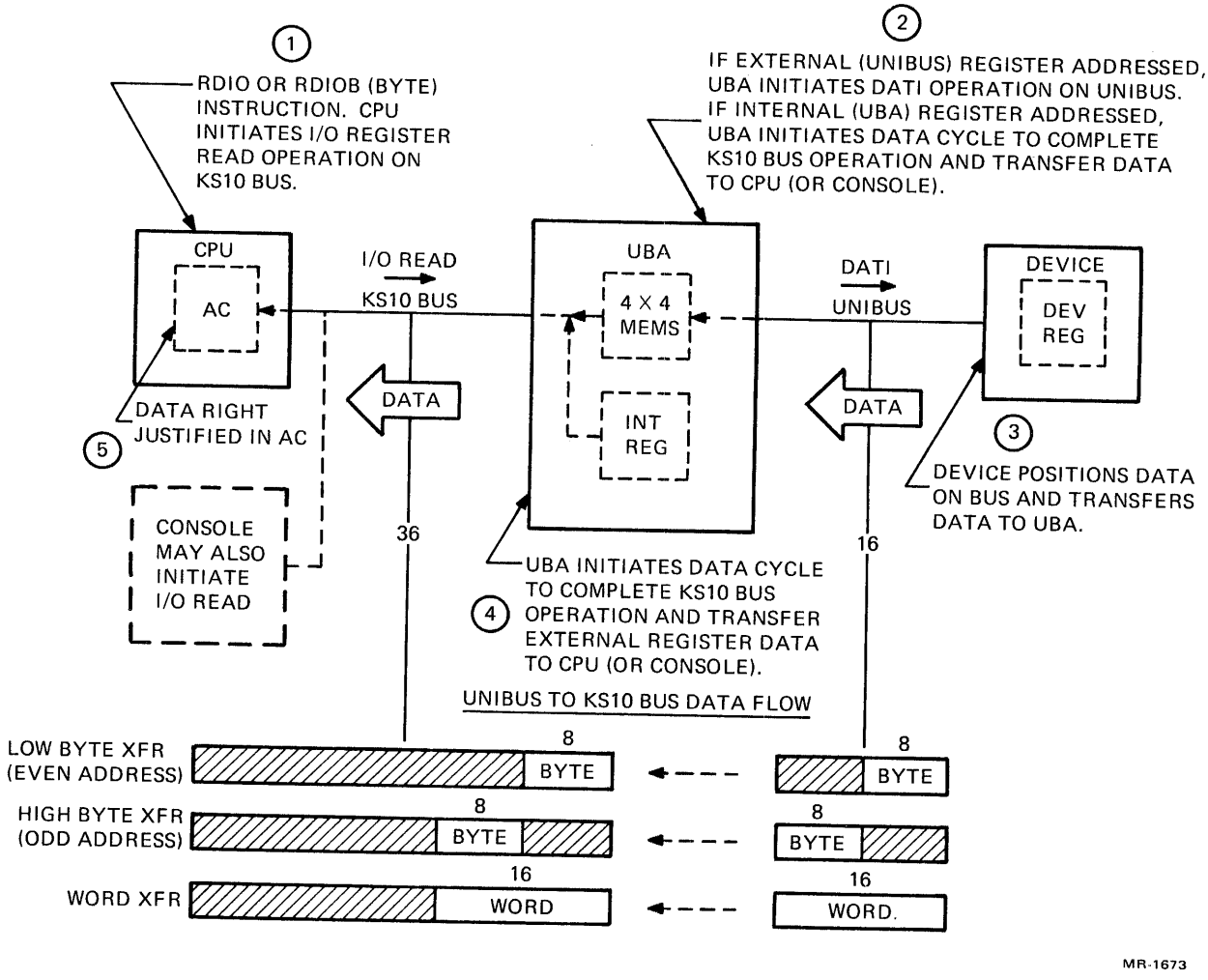


Figure 5-46 I/O Read, Data Flow

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.

5.9.1.3 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 UBA's status register (Paragraph 5.9.2). 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 5-47. 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, such as 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 correponding 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.

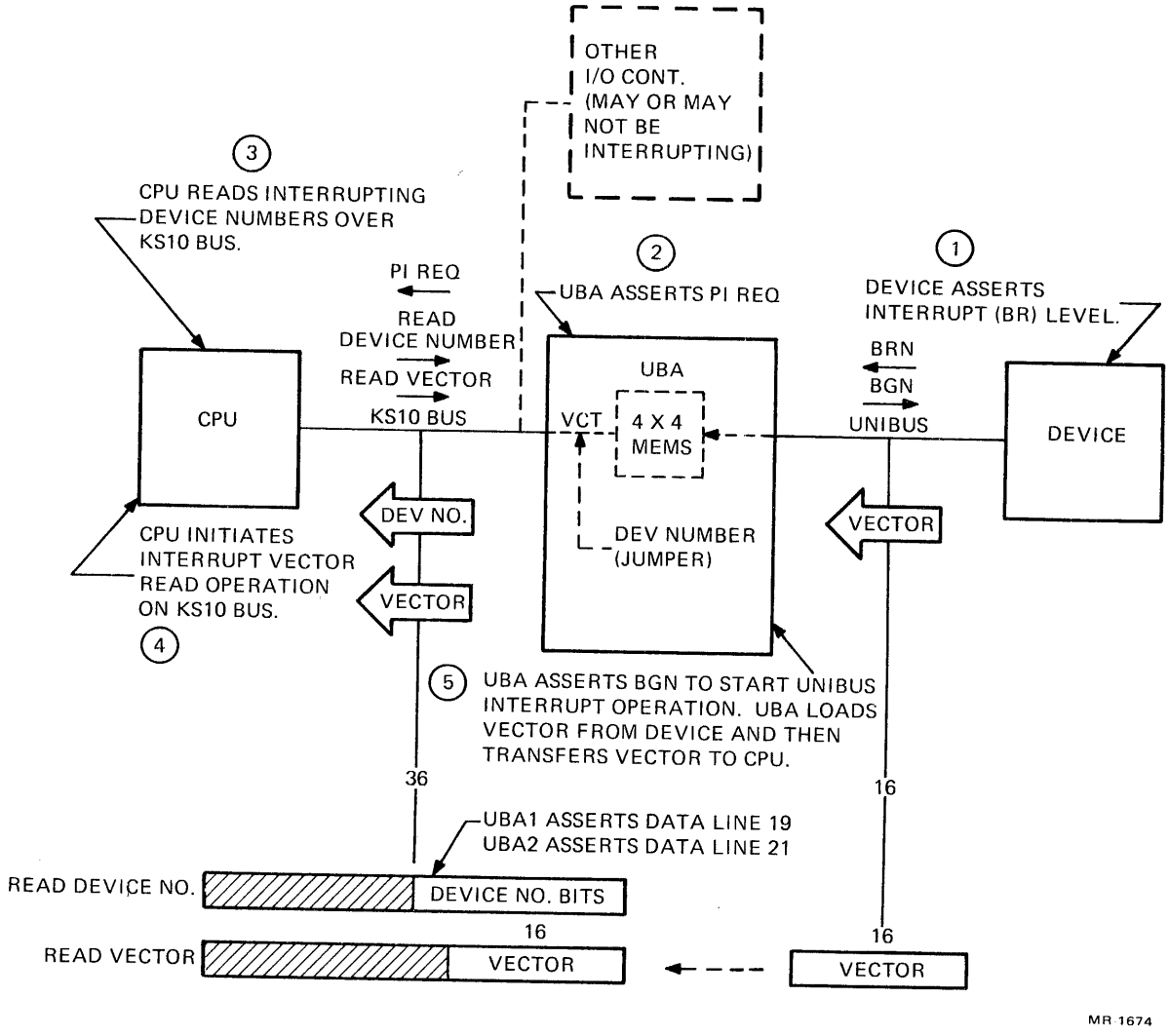


Figure 5-47 PI Operation, Data Flow

5.9.2 UBA Status and Control Registers

The UBA has the following internal registers.

Address (octal)	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_8) is a write-only register.

5.9.2.1 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. The other five RAM bits are used for control purposes. Bit format and definitions for the I/O instructions accessing the RAM are given in Figure 5-48. As shown, bit format is not the same when loading the RAM as when reading the RAM locations.

Unibus to memory address translation is shown in Figure 5-49. 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 5-42.) The next nine least significant bits of Unibus address (bits 10–2) are used directly as the nine least significant bits of memory address (bits 27–35). This is similar to virtual to physical address translation in the CPU. The nine bits specify one of 512 words; that is, a word within a 512 word page. To furnish the page address, six of the remaining seven 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 the Unibus address (bit 17) is not used.

NOTE

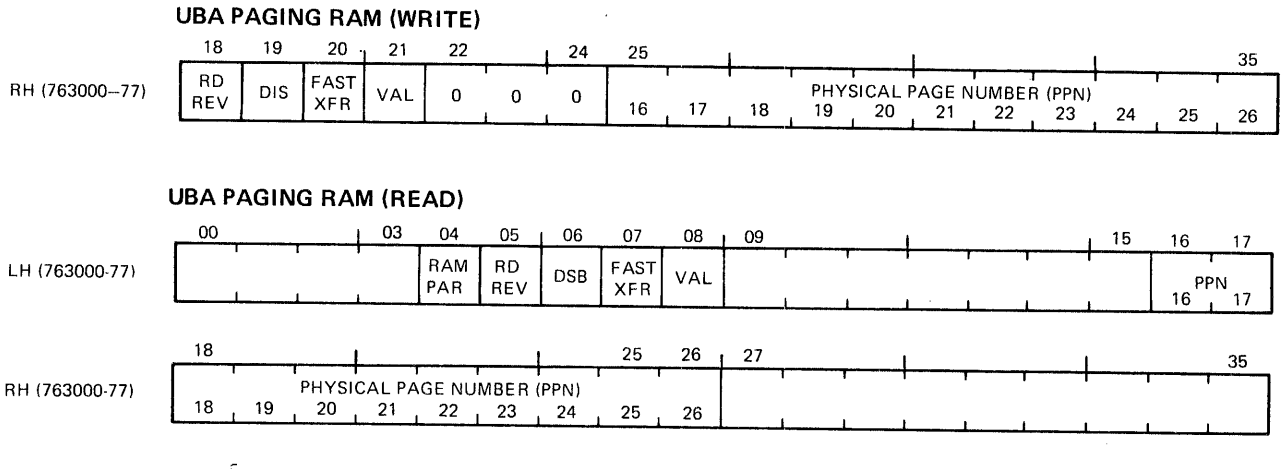
The most significant bit of Unibus address (the 64K bit) must be made equal to 0 for NPR data transfers. If equal to 1, a memory reference is not made, causing the Unibus device to time-out, set an error flag, and terminate the device read or write operation.

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.



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 5-48 Paging RAM

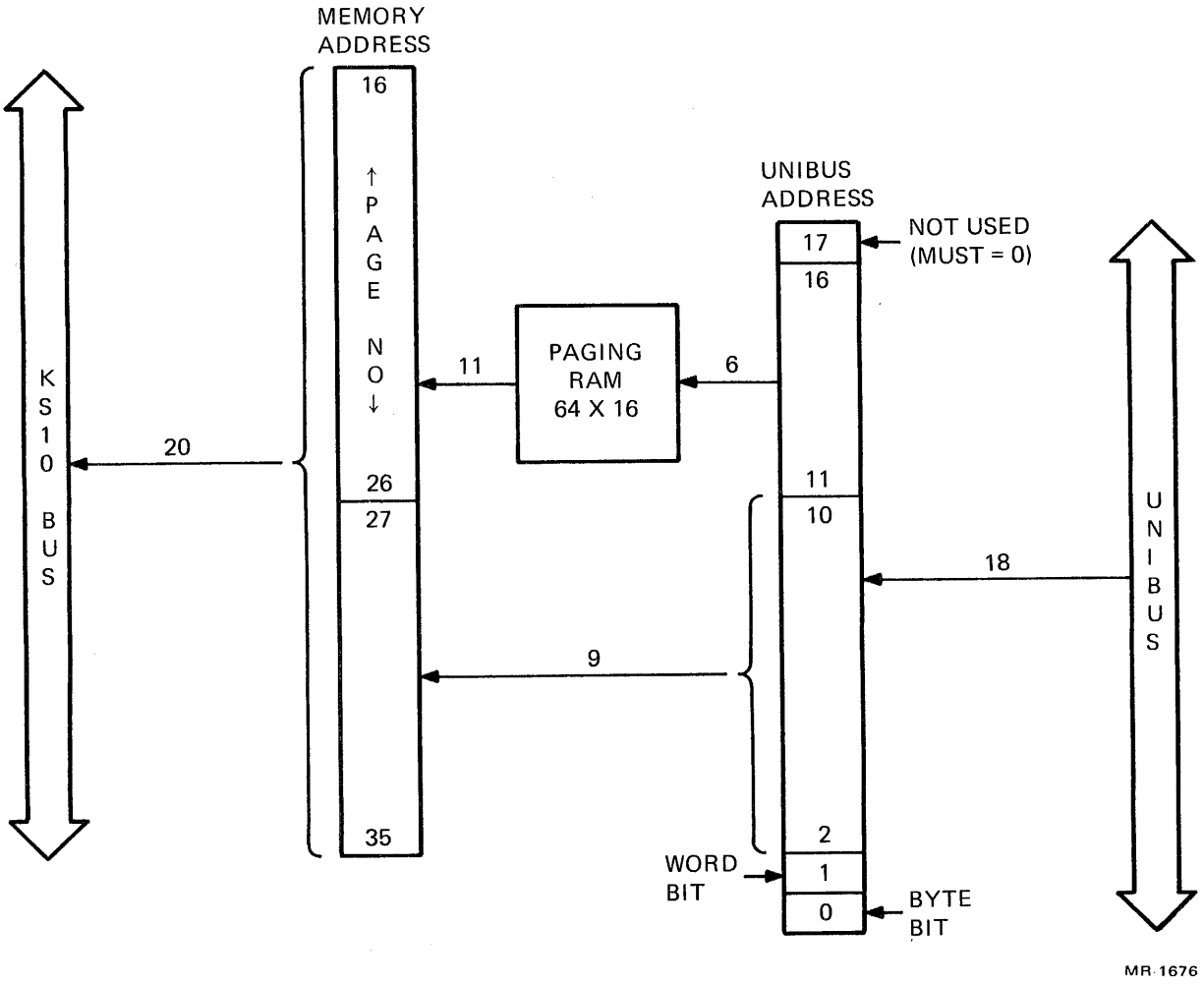


Figure 5-49 Unibus to Memory Address Translation

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.

5.9.2.2 Status Register – UBA status register bit format and definitions are given in Figure 5-50. 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 any of the following conditions.

   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.

   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 (for example, 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.

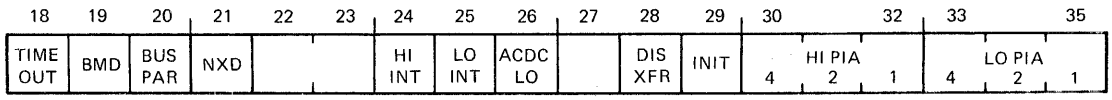
3. BUS PARITY ERROR (bit 20) – Indicates that the UBA detected a KS10 (backplane) bus parity error when it either received or transmitted bus information. Unless disabled by a console command, a BUS PARITY error causes the console module to stop the CPU clock. The BUS PARITY ERROR flag is cleared by 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 power supply), or assertion of KS10 bus AC LOW (condition sensed by H7130 power supply).

UBA STATUS REGISTER

RH (763100)



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 5-50 UBA Status

5.9.2.3 Maintenance Register – The maintenance register contains a single write-only control bit as shown in Figure 5-51. 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 5.9.7), 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.

5.9.3 Logical Organization

The UBA consists of the following major logic elements, shown in Figure 5-52.

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 tranceiver/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.

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 (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.

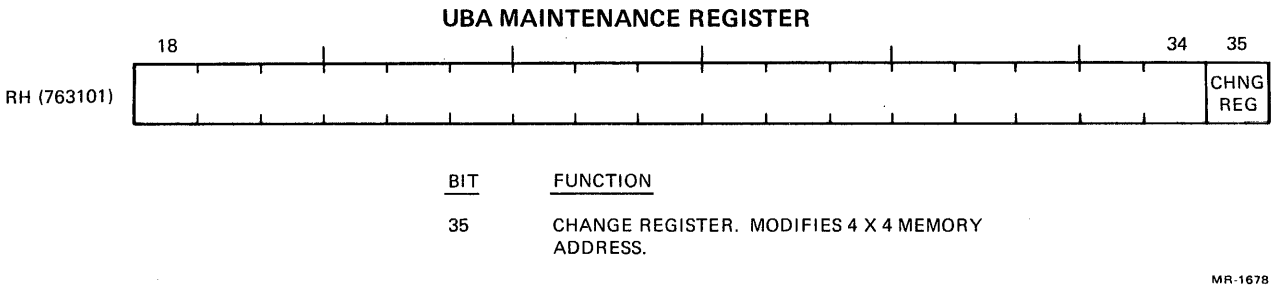


Figure 5-51 Maintenance Register

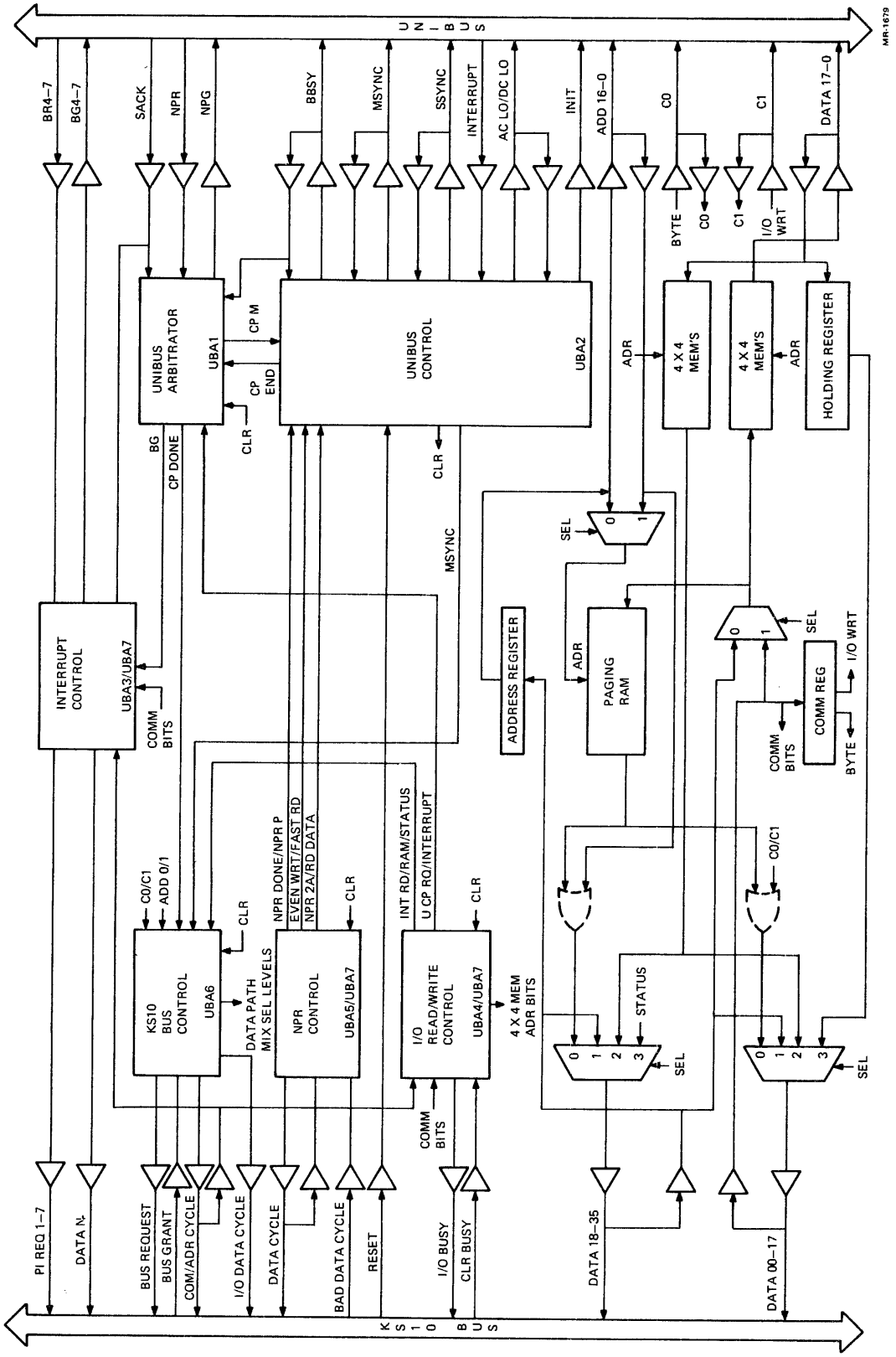


Figure 5-52 UBA, Detailed Block Diagram

5.9.4 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 5-53 and 5-54. 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 if 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.)

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 (for example, 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 reenable 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 reenable 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)

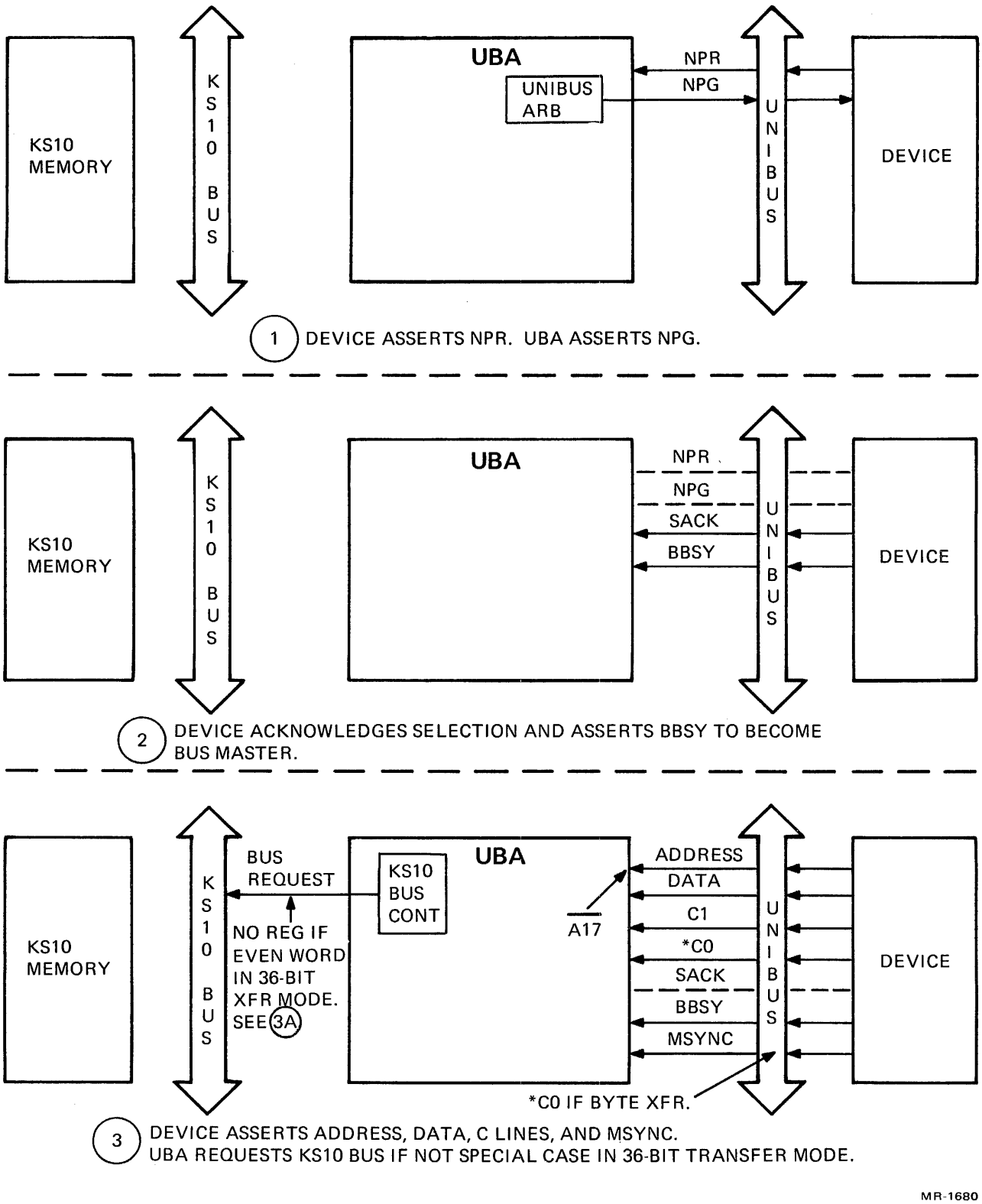


Figure 5-53 NPR Write, Bus Dialogue (Sheet 1 of 3)

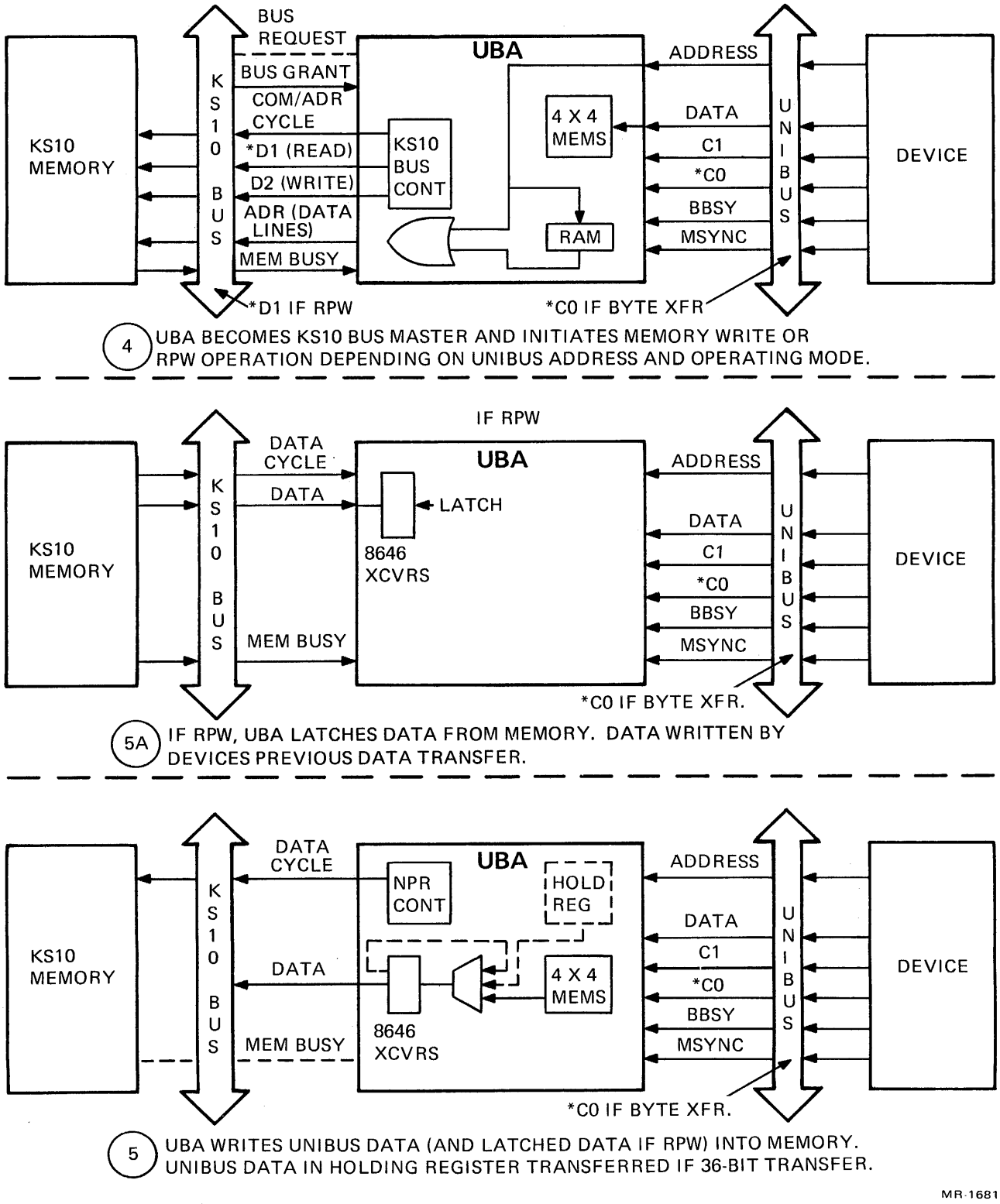


Figure 5-53 NPR Write, Bus Dialogue (Sheet 2 of 3)

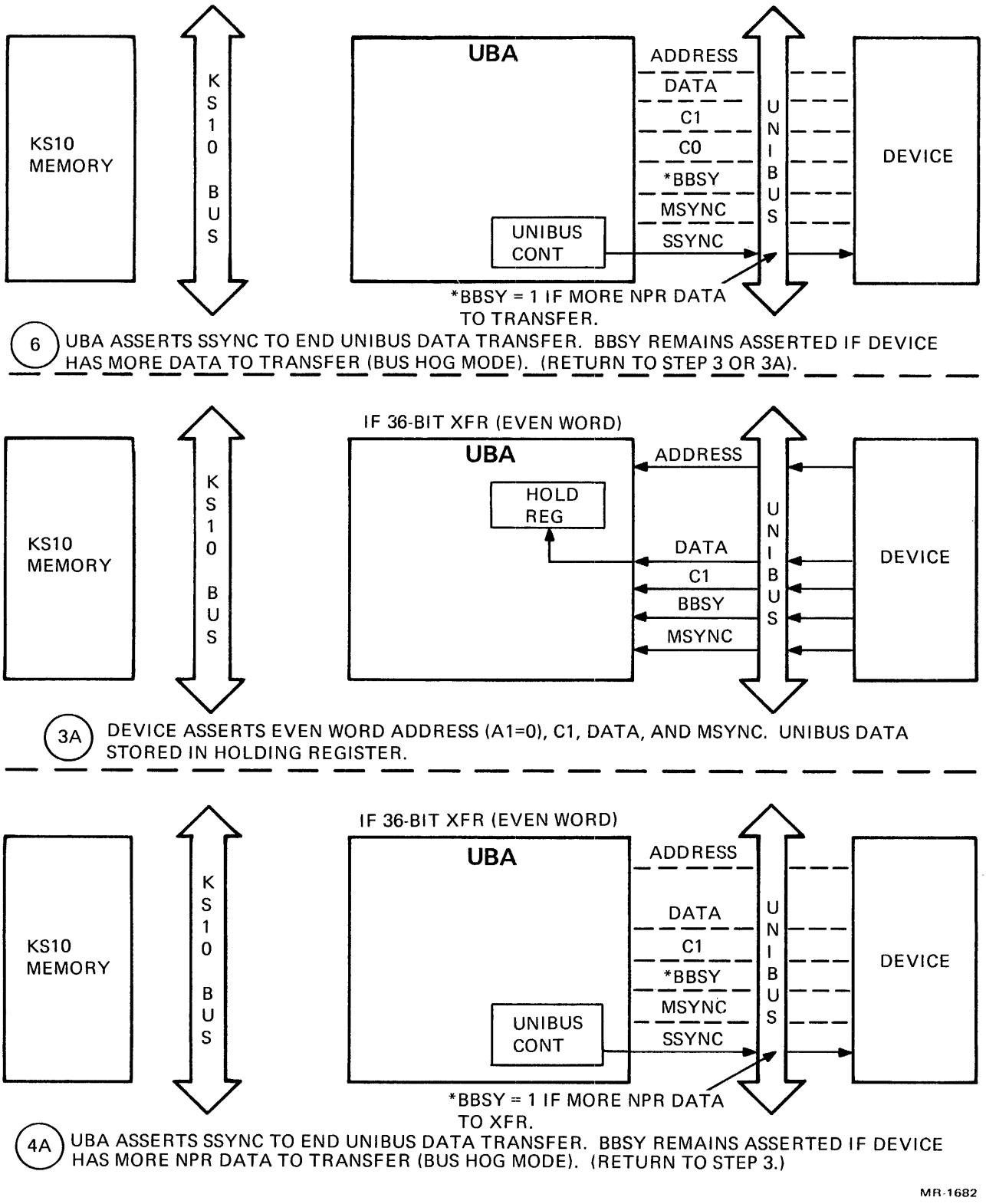


Figure 5-53 NPR Write, Bus Dialogue (Sheet 3 of 3)

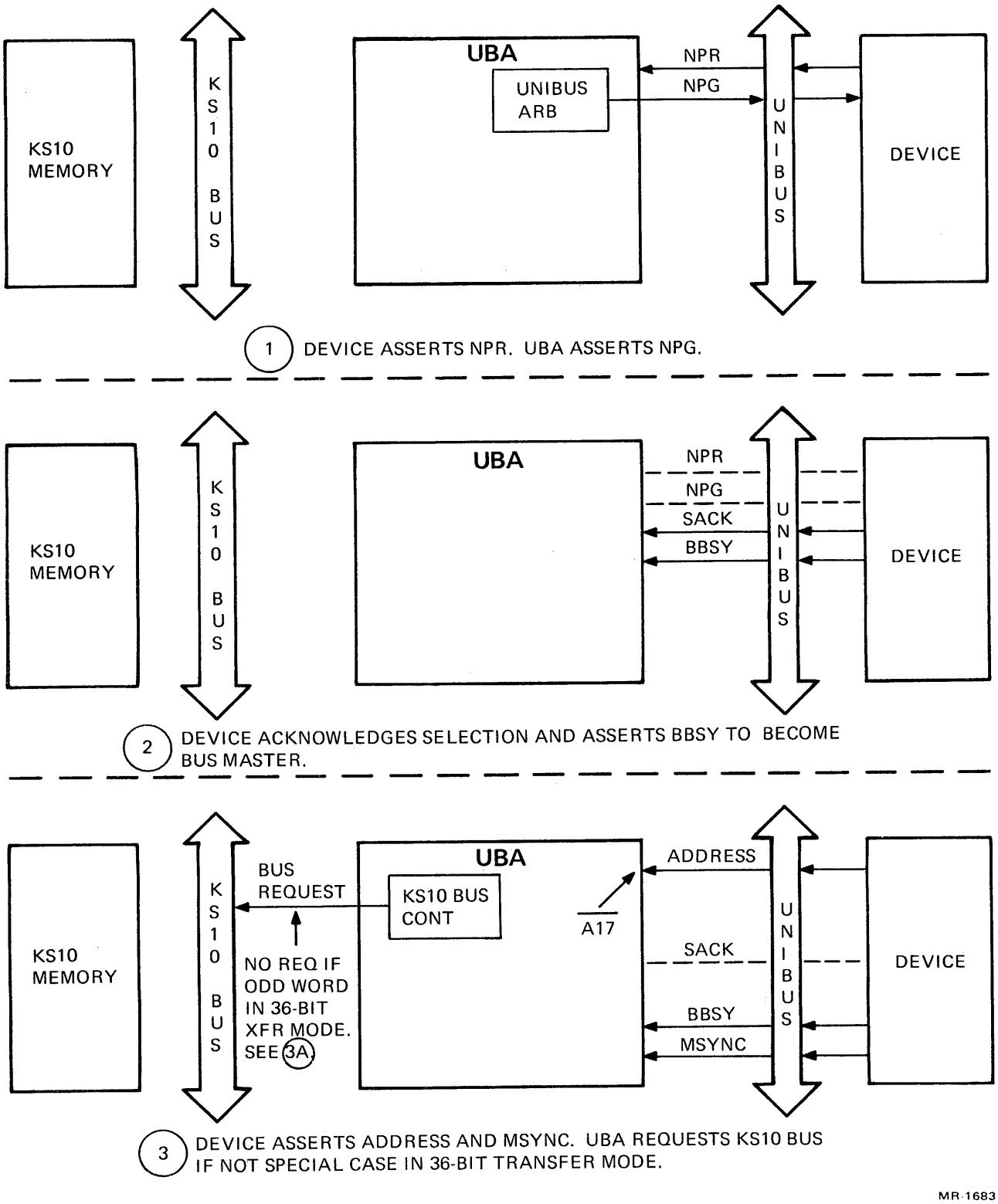


Figure 5-54 NPR Read, Bus Dialogue (Sheet 1 of 3)

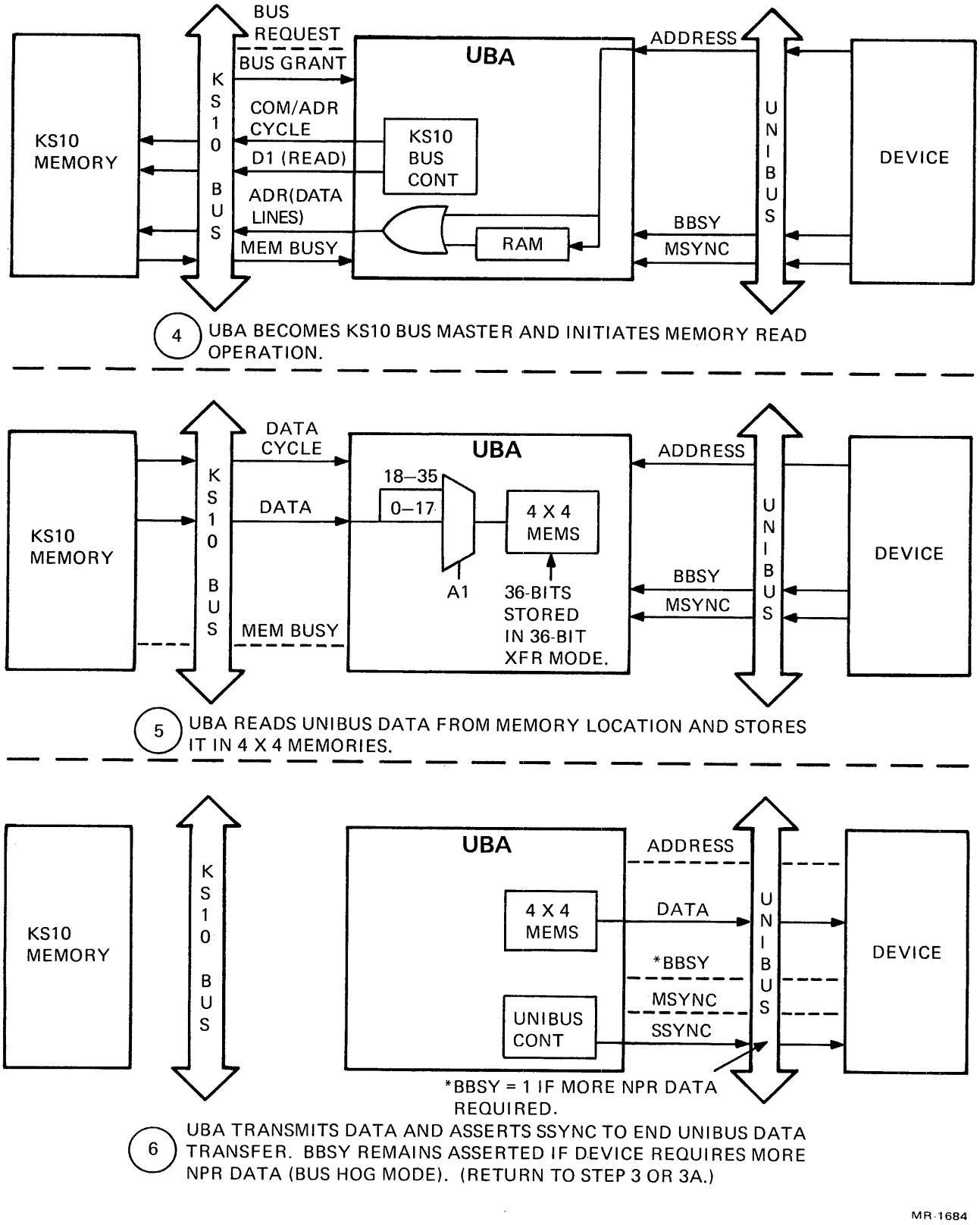


Figure 5-54 NPR Read, Bus Dialogue (Sheet 2 of 3)

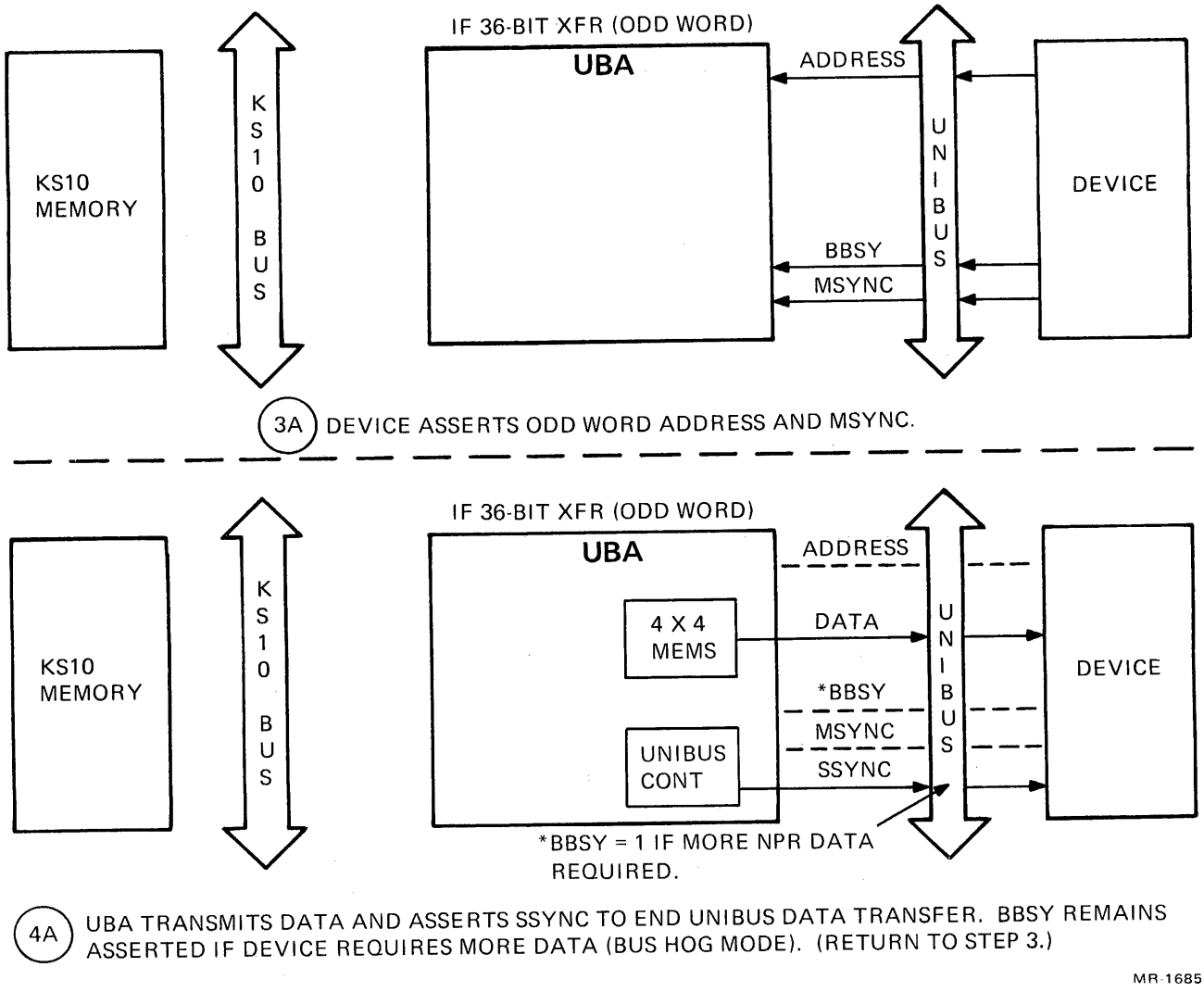


Figure 5-54 NPR Read, Bus Dialogue (Sheet 3 of 3)

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 tranceiver 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 (by a DATO or DATOB initiated by a 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 Lines		**Operation**
01	**02**	
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 on the Unibus operation being performed, the Unibus address, and any special operating modes set in the UBA. (Refer to Figures 5-43 and 5-44.) 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, (that is, 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 5-11.

   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. It first reads an odd word previously stored in memory, and then writes 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). Which half is stored depends on the state of select level UBA6 NPR at the input data mixers to the 4 × 4 memories. 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 UBA's 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 5-12.) 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 P 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.

Table 5-12 Data Path Mixer Selection for NPR Transfers

Select Level Inputs				Select Levels				
				USEL	UBSEL	LSEL	LBSEL	
A1	A0	FST D (0–35) >MOS	FORCE RPW	2 1	2 1	2 1	2 1	Function (See Below)
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	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

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 time-out 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 tranferred immediately (that is, 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.

5.9.5 I/O Data Transfer Operation

Figures 5-55 and 5-56 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 in the Field Maintenance Print Set. Note that the steps in the following description correspond to the steps in Figures 5-55 and 5-56.

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 UBA's 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 is 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.

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 is 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 (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).

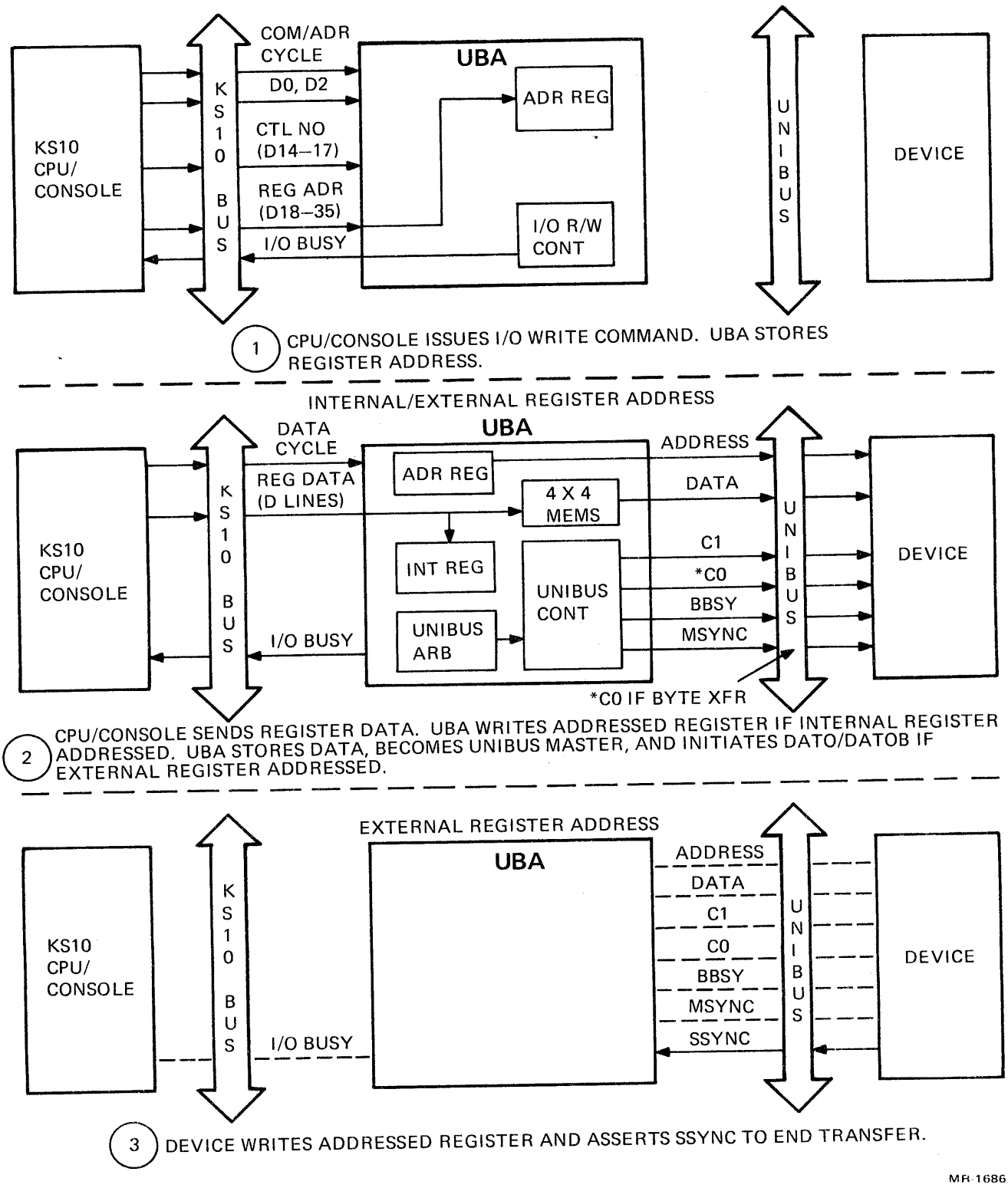


Figure 5-55 I/O Write, Bus Dialogue

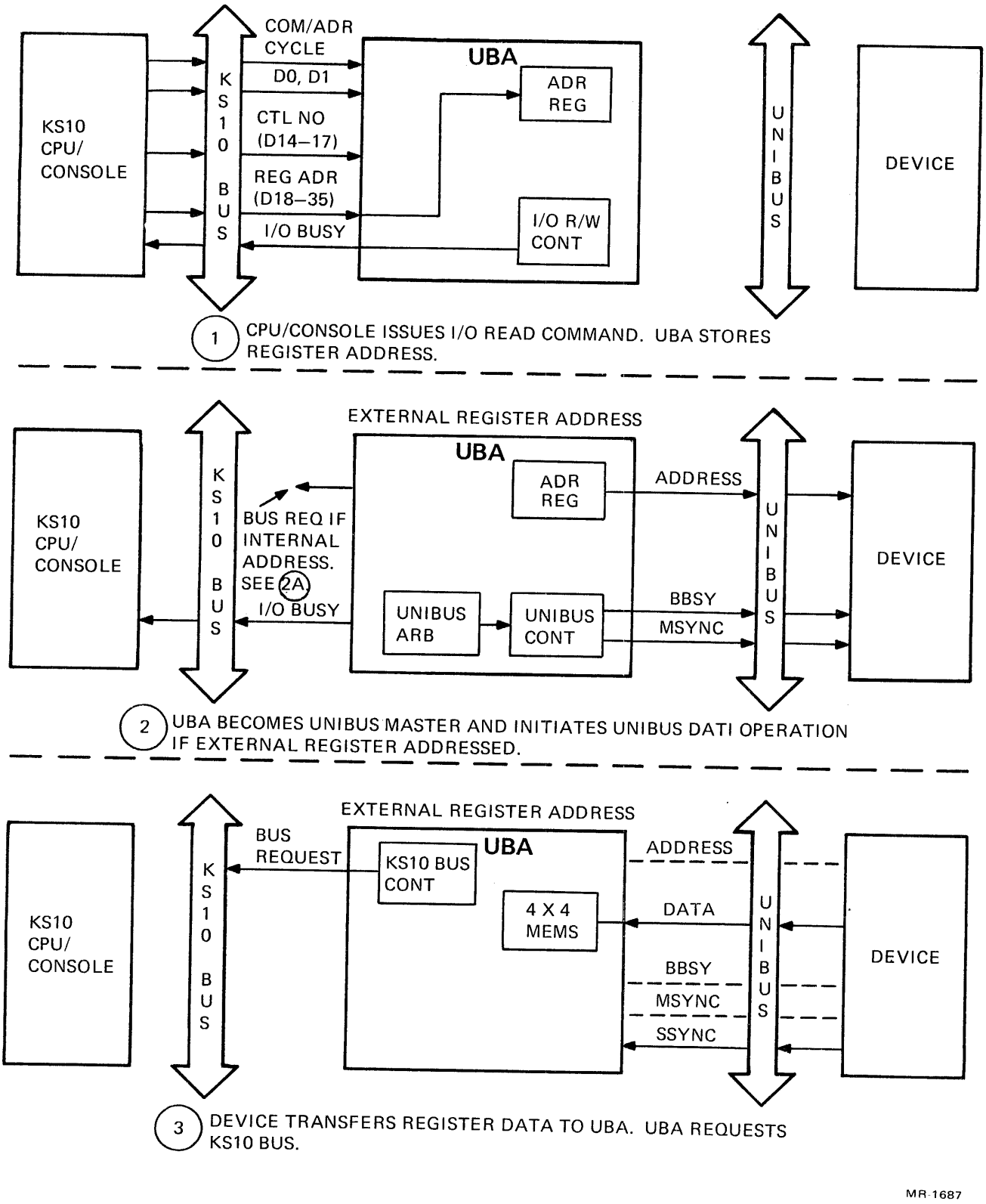


Figure 5-56 I/O Read, Bus Dialogue (Sheet 1 of 2)

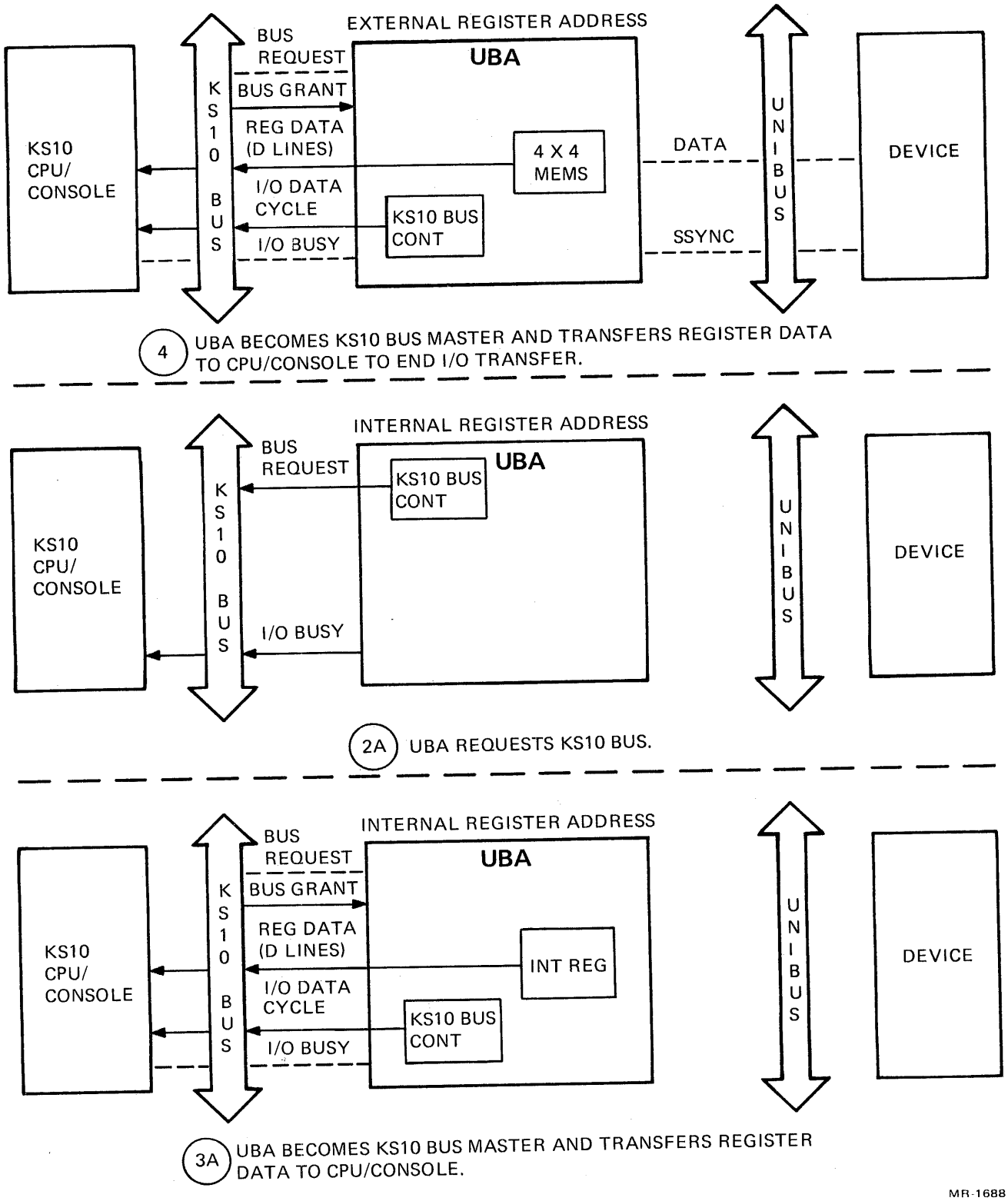


Figure 5-56 I/O Read, Bus Dialogue (Sheet 2 of 2)

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 UBA's 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 reenabling 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 tranferred 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.

5.9.6 PI Operation

Figure 5-57 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 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 (the UBAs) that interrupt 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 UBA's 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 numbers, 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.

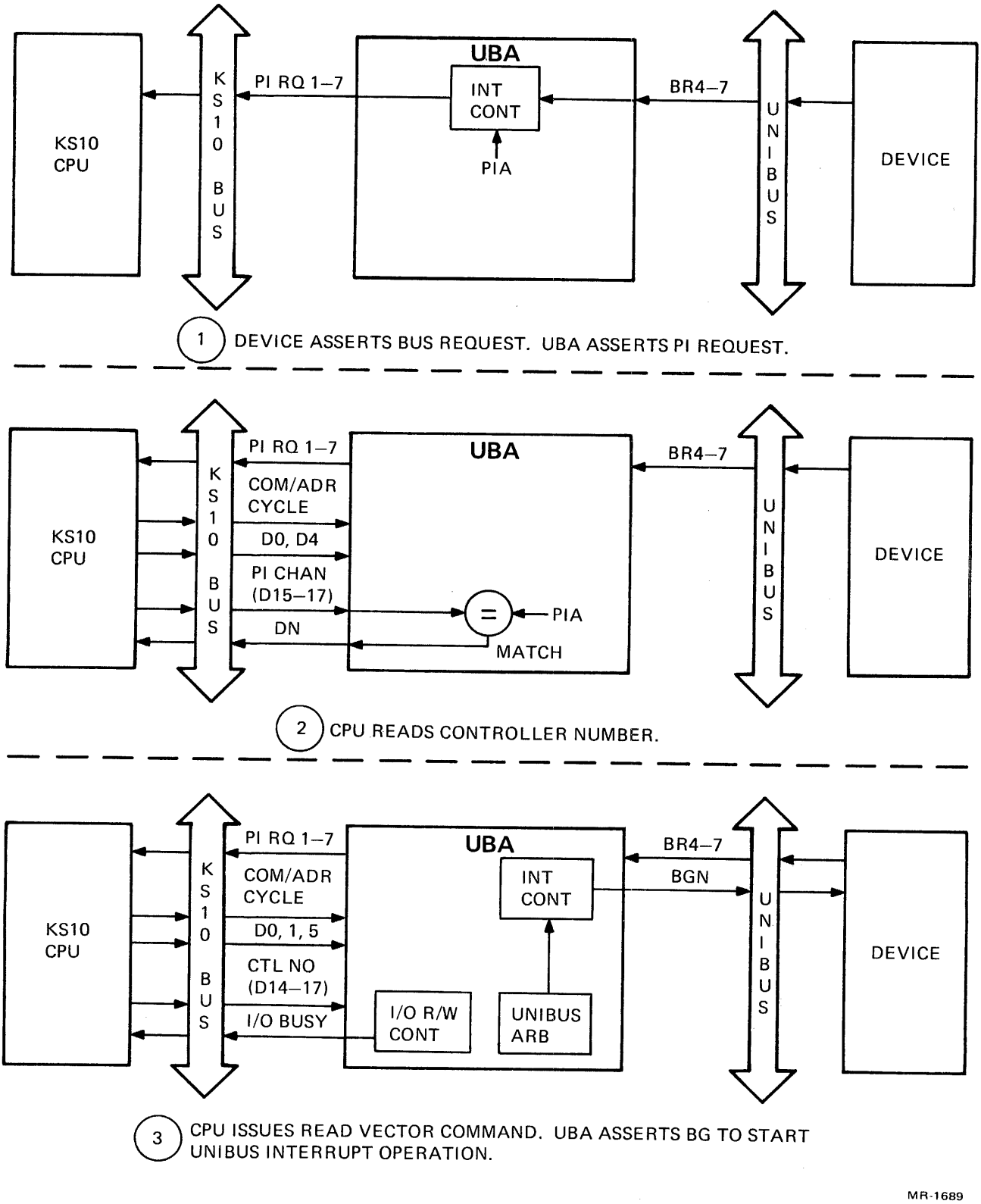


Figure 5-57 PI Operation, Bus Dialogue (Sheet 1 of 2)

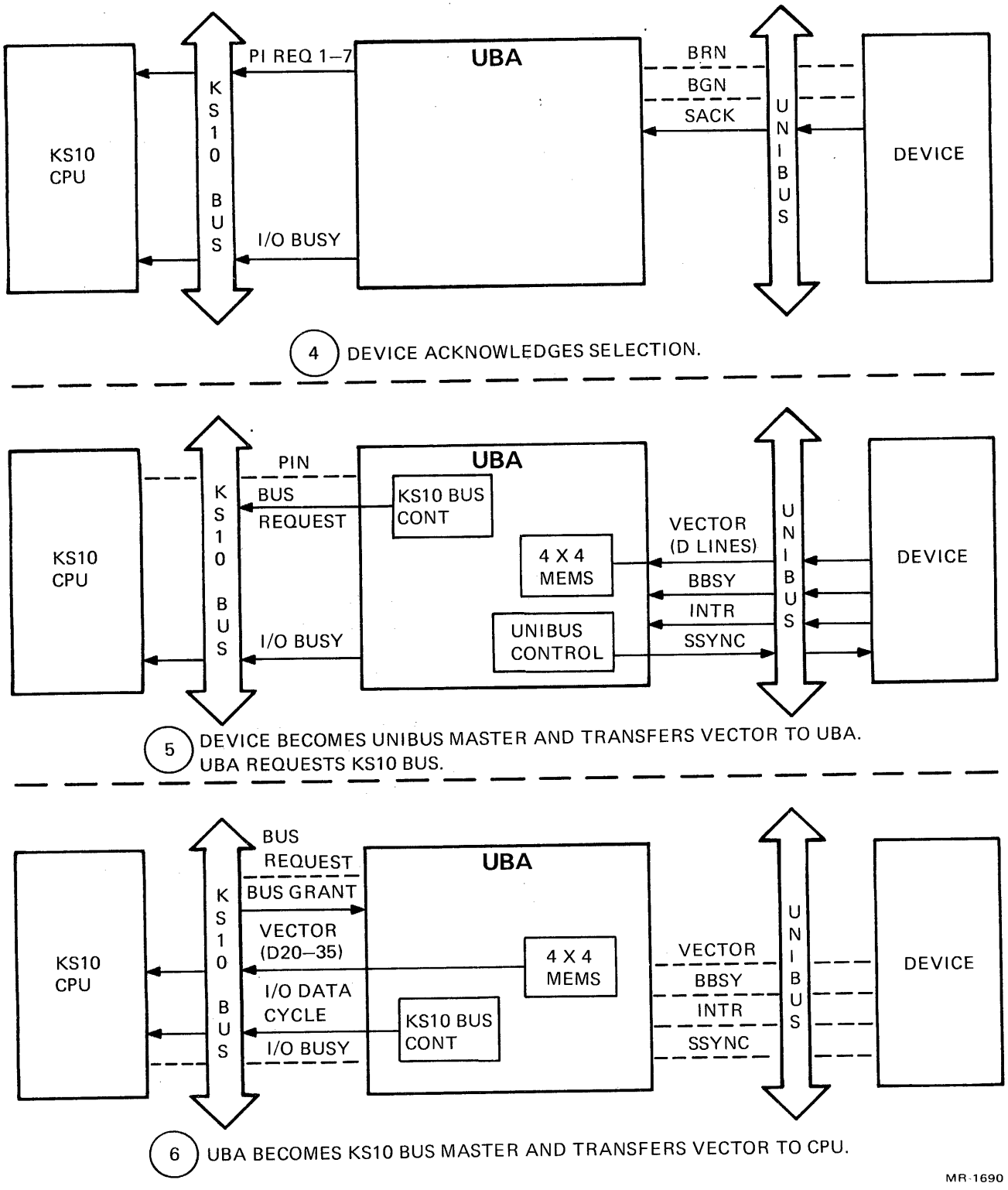


Figure 5-57 PI Operation, Bus Dialogue (Sheet 2 of 2)

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 asseted 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 (that is, if 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 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 interupt 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 reenable 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.

5.9.7 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_8) 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 5.9.5, and as shown in step 1 of Figures 5-55 and 5-56. The I/O address is decoded as an internal address (that is, 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 5.9.5 and as shown in step 2 of Figures 5-55 and 5-56. 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 5.9.4, and as shown in steps 3–6 of Figures 5-53 and 5-54. 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 peformed 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 5.9.5 (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 5.9.5 (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.

5.10 KS10 POWER SYSTEM

A simplified block diagram of the KS10 power distribution system is shown in Figure 5-58. 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
KS10-AB	207–253 Vac	50 Hz	4.95 A	12.5 A	6 cycles	1.14

The major power system components are as follows.

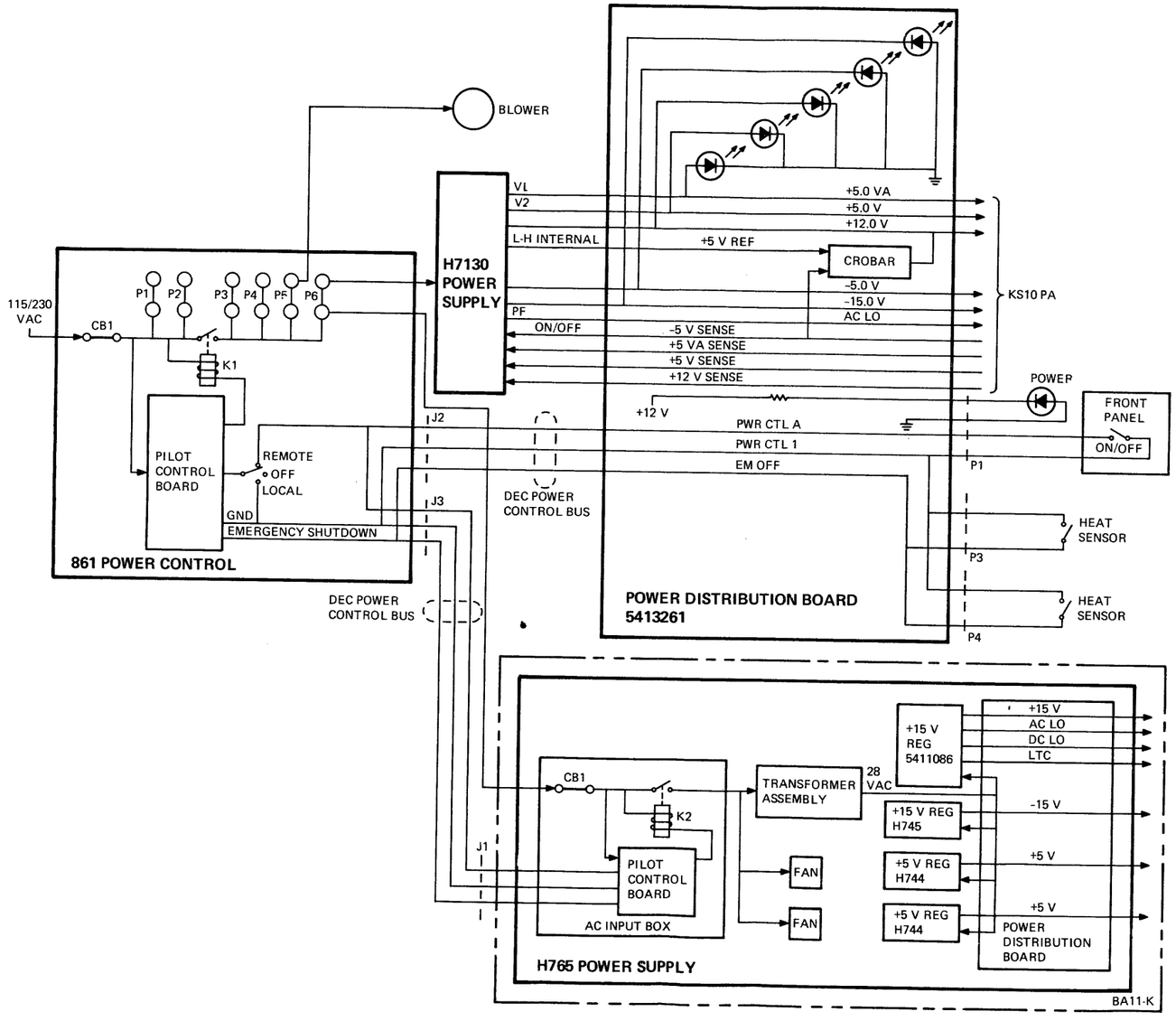
1. 861-C (60 Hz) or 861-B (50 Hz) power controller.
2. H7130 power supply and 5413261 power distribution module.
3. H765A (60 Hz) or H765B (50 Hz) power supply.

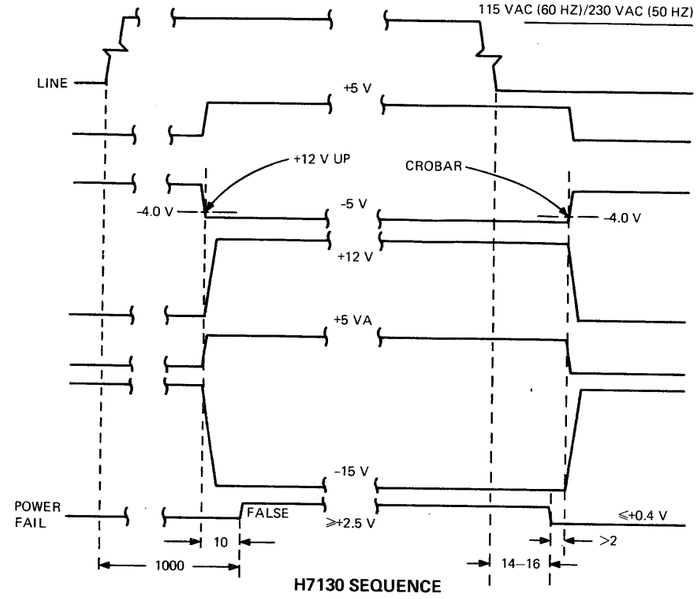
The H7130 power supply is used to power the KS10-PA card cage assembly. The H765A power supply is used to power the BA11-KE drawer (115 Vac version); the H765B power supply is used to power the BA11-KF drawer (230 Vac version).

The location of the major power system components are given in Figure 1-7 and on the KS10 Unit Assembly drawing (D-UA-KS10-0-0).

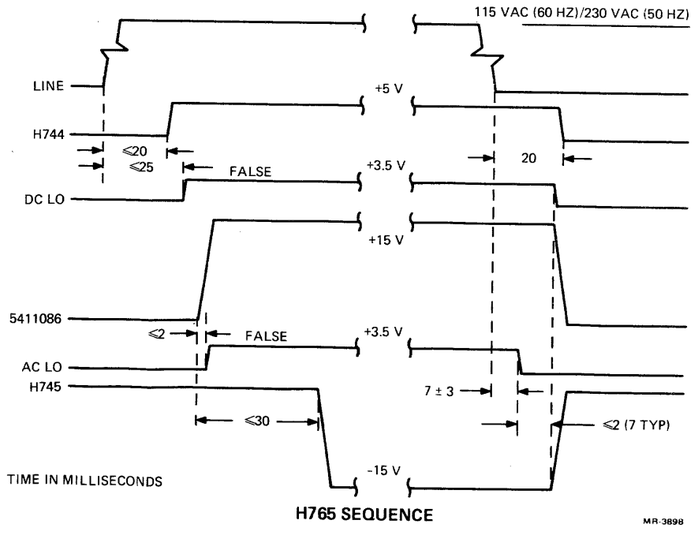
5.10.1 861 Power Controller

The 861 controls and distributes power in the KS10 cabinet. It contains a line cord circuit breaker, a contactor with associated control circuitry, line filters, two unswitched duplex outlets, and four switched duplex outlets. Two of the switched outlets are used to provide switched input power to the H7130 and H765 power supplies and to the blower for the KS10-PA card cage assembly. When the 861's REMOTE/LOCAL switch is in LOCAL, power is switched on and off at the back of the cabinet by means of the line cord circuit breaker. In REMOTE, power is controlled by the KS10 front panel POWER switch via the DEC power control bus. Also (via the DEC power control bus) power is shut down if overheating is detected by the heat sensors mounted over the KS10-PA assembly. Input power for the 861 is as follows.





H7130 SEQUENCE



H765 SEQUENCE

MR-3898

Figure 5-58 KS10 Power System

Power Controller	Voltage	Current (Max.)	Phase
861-B	180–264 Vac	16 A	1
861-C	90–132 Vac	24 A	1

NOTE

Loads external to the KS10 cabinet are NOT to be plugged into the 861 power control.

5.10.2 H7130 Power Supply and 5413261 Power Distribution Module

The H7130 is a multiple output off line switching type power supply with remote sensing capability. The dc outputs (that is, +5 V, +5 VA, +12 V, – 5 V, and –15 V) connect to the KS10-PA backplane via the 5413261 power distribution module. (The remote sense lines for the H7130 that monitor voltages directly on the backplane are also routed through the power distribution module.) Power supply features include overcurrent, overvoltage, power-fail, and thermal shutdown protection, plus power sequencing of +12 V with respect to the –5 V output. Electrical specifications for the H7130 are as follows.

Power Supply	Line Voltage	Freq.	Current (Max.)
H7130C	115 Vac ± 10%	60 Hz	3.75 A
H7130D	230 Vac ± 10%	50 Hz	1.87 A

NOTE

The voltage adjustment procedure for the H7130 showing measuring points on the KS10-PA backplane is given in the *KS10-Based DECSYSTEM-2020 Installation Manual* **(EK-0KS10-IN).**

Power sequencing for the H7130 is shown in Figure 5-58. The +5 V and –5 V sequence on first. When the –5 V sense line (connected to the H7130 ON/OFF terminal) reaches –4 V, the other three voltages (+5 VA, +12 V, and –15 V) sequence on.

Besides supplying the normal operating voltages, the H7130 supplies +5 V REF to power a 12 V crobar circuit on the power distribution module. This circuit crobars (short circuits) +12 V to ground to protect MOS memory in the event –5 V fails. The circuit activates when the –5 V sense line is at a value of –4 V.

In addition to its power distribution and crobar functions, the 5413261 power distribution module has light-emitting diodes (LEDS) which indicate when the H7130 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.

NOTE

Physical locations of the LEDs on the 5413261 are shown in the *KS10-Based DECSYSTEM-2020 Installation Guide* **(EK-0KS10-IN).**

Another function of the 5413261 power distribution module is to connect the front panel power switch and the heat sensors to the DEC power control bus so that the 861 may turn power on and off as discussed in Paragraph 5.10.1. The 5413261 also routes +12 V through a resistor to light the front panel POWER indicator. In addition, it interconnects the other front panel indicators and switches to the KS10-PA backpanel.

5.10.3 H765 Power Supply (BA11-K)

The H765 power supply consists of five standard DIGITAL regulators (two H744s, one H745, one H754 which is not used, and one 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 KS10).

The 7009811 power control box contains a line cord circuit breaker, power relay, and relay control circuitry. Like the 861 power control, the 7009811 connects to the DEC power bus. Two versions of the power control box are used: the 7009811-1 for 115 Vac operation, and the 7009811-2 for 230 Vac operation.

The power distribution board on the H765 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.

Power Supply	Line Voltage	Freq.	Current (Max.)
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 5-58.

APPENDIX A
KS10 DIFFERENCES (KS10 VS. KL10)

A.1 INTRODUCTION

The major differences in operation and programming between the KS10 and KL10 are as follows.

- Paging – Both TOPS-10 and TOPS-20 paging are supported on the KS10.

- Machine Modes – Because they are not supported by TOPS-20, the submodes associated with User mode (Public and Concealed) and with Exec mode (Supervisor and Kernal) are not implemented in the KS10.

- 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.

- Interrupt Handling – KS10 priority interrupt operation is the same as the KL10-B (KL10-PA processor), with the following exceptions.

  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.

  2. The KS10 implements only the standard interrupt function (40 + 2n) and the dispatch (vector) interrupt function. Other interrupt functions (increment, byte transfer, etc.) are not incorporated.

  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).

- KS10 Instruction Set – The KS10 has the same instruction set as the KL10-B (Model PA Processor-section 0 addressing only), with the following exceptions.

  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 instruction op codes are in the range 700–777$_8$. Op code assignments are shown in Table A-1.

Table A-1 I/O Instruction Op Codes (Octal)

Op Code	Op Code Last Digit (X)							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
70X	APR0	APR1	APR2	–	UMOVE	UMOVEM		–
71X	TIOE	TION	RDIO	WRIO	BSIO	BCIO	–	–
72X	TIOEB	TIONB	RDIOB	WRIOB	BSIOB	BCIOB	–	–
73X	–	–	–	–	–	–	–	–
74X	–	–	–	–	–	–	–	–
75X	–	–	–	–	–	–	–	–
76X	–	–	–	–	–	–	–	–
77X	–	–	–	–	–	–	–	–

A.2 INTERNAL (APR) I/O INSTRUCTIONS

The internal I/O instructions (APR0-2; op codes 700–702$_8$) use the AC field as an extension of the op code as indicated in Table A-2. For example, the RDEBR instruction (an APR instruction with op code = 701$_8$) is specified by an AC value of 24$_8$. 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 A-2 AC Field Assignments (Octal) for APR I/O Instructions

AC	Op Code		
	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_8)** - 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 A-1.

- **WRAPR (70020_8)** - 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 A-2.

- **RDAPR (70024_8)** - The RDAPR instruction stores APR status in E. It corresponds to the CONI APR instruction used in the KL10. Bit format is shown in Figure A-3.

- **WRPI (70060_8)** - 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 A-4.

- **RDPI (70064_8)** - The RDPI instruction is identical to the KL10 CONI PI instruction except that bits 18–20 read no status (write even parity is not implemented). Bit format is shown in Figure A- 5.

- **RDUBR (70104_8)** - 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 A-6.

- **CLRPT (70110_8)** - 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 A-7.

- **WRUBR (70114_8)** – 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 A-8.

- **WREBR (70120_8)** – 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 A-9.

- **RDEBR (70124_8)** – The RDEBR instruction reads the EBR into the right half of E. It is comparable to the KL10 CONI PAG instruction. Bit format is shown in Figure A-10.

- **RDSPB (70200_8)** – The RDSPB instruction reads the shared pointer table (SPT) base register and stores the value in E. Bit format is shown in Figure A-11.

- **RDCSB (70204_8)** – The RDCSB instruction reads the core status table (CST) base register and stores the value in E. Bit format is shown in Figure A-12.

- **RDPUR (70210_8)** – The RDPUR instruction reads the process use register (PUR) and stores the value in E. Bit format is shown in Figure A-13.

- **RDCSTM (70214_8)** – The RDCSTM instruction reads the CST mask register and stores the value in E. Bit format is shown in Figure A-14.

- **RDTIME (70220_8)** – 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 A-15.

- **RDINT (70224_8)** – The RDINT instruction reads the current value of the interval timer period register and stores the value in E. Bit format is shown in Figure A-16.

- **RDHSB (70230_8)** – The RDHSB instruction stores the value of the halt status block address at E. Bit format is shown in Figure A-17.

- **WRSPB (70240_8)** – The WRSPB instruction loads the SPT base register from E. Bit format is shown in Figure A-18.

- **WRCSB (70244_8)** – The WRCSB instruction loads the CST base register from E. Bit format is shown in Figure A-19.

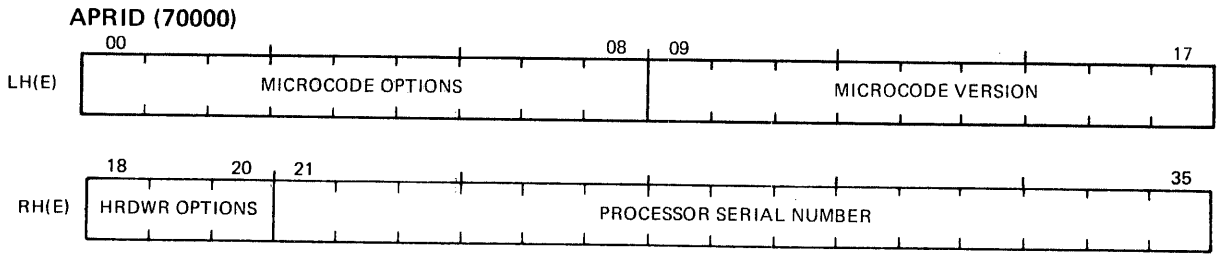
- **WRPUR (70250_8)** – The WRPUR instruction loads the PUR from E. (Bit format is shown in Figure A-20.) The PUR contains the AGER in the left-most bits. These bits are cleared by ANDing the CST entry with the CST mask; then the entire PUR is ORed with the CST entry.

- **WRCSTM (70254_8)** – 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 A-21.

- **WRTIME (70260_8)** – The WRTIME instruction loads the double-word at E and E + 1 into the time base (Bit format is shown in Figure A-22.) The time base up-counts at 4.096 MHz.

- **WRINT (70264_8)** – 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 A-23.

- **WRHSB (70270_8)** – The WRHSB instruction loads the signed word at E as the address of the halt status block. 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_8. Bit format for the WRHSB instruction is shown in Figure A-24.

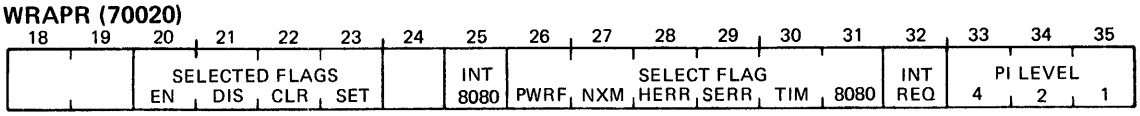


BIT(S)	FUNCTION
0-8	RESERVED FOR MICROCODE VERSION
9-17	MICROCODE VERSION NUMBER
18-20	HARDWARE OPTIONS (BITS CURRENTLY = 0)
21-35	PROCESSOR SERIAL NUMBER

MR-0229

Figure A-1 APRID Instruction

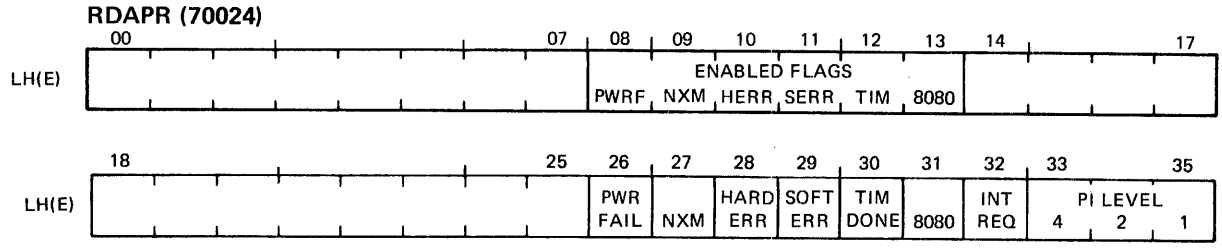
WRAPR (70020)



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 A-2 WRAPR Instruction



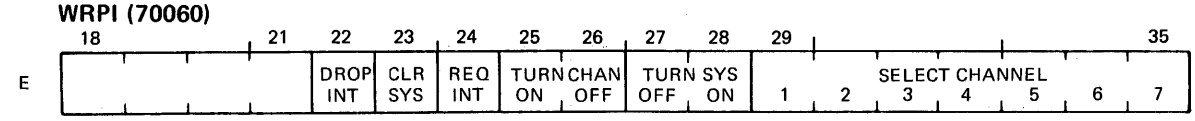
BIT(S)	FUNCTION
08	POWER FAIL ENABLED
09	NON-EXISTENT MEMORY ERROR ENABLED
10	HARD MEMORY ERROR INTERRUPT ENABLED
11	SOFT MEMORY ERROR INTERRUPT ENABLED
12	INTERVAL TIMER ENABLED
13	8080 CONSOLE INTERRUPT ENABLED
26	POWER FAIL ERROR
* 27	NON-EXISTENT MEMORY ERROR
* 28	HARD MEMORY ERROR (CANNOT BE CORRECTED 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 A-3 RDAPR Instruction

WRPI (70060)

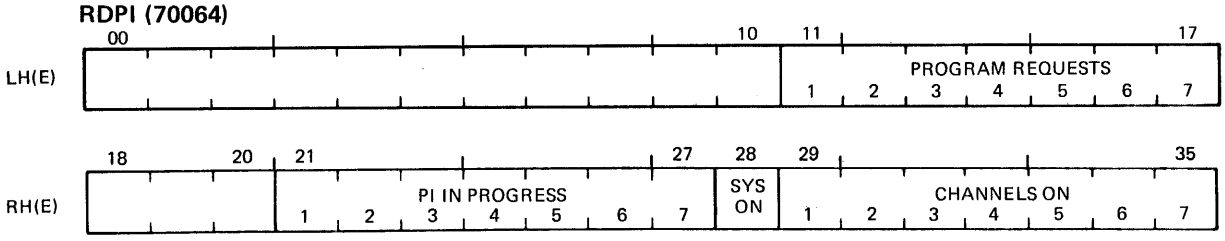


BIT(S)	FUNCTION
22	DROP PROGRAM REQUESTS ON SELECTED CHANNELS
23	CLEAR PI SYSTEM
24	INITIATE INTERRUPTS ON THE SELECTED CHANNELS
25	TURN ON THE SELECTED CHANNELS
26	TURN OFF THE SELECTED CHANNELS
27	DEACTIVATE THE PI SYSTEM
28	ACTIVATE THE PI SYSTEM
29-35	SELECT CHANNELS FOR BITS 22, 24, 25, AND 26

MR-0232

Figure A-4 WRPI Instruction

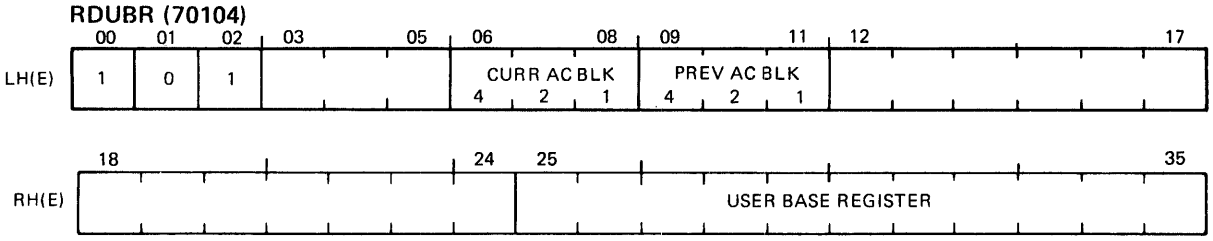
RDPI (70064)



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

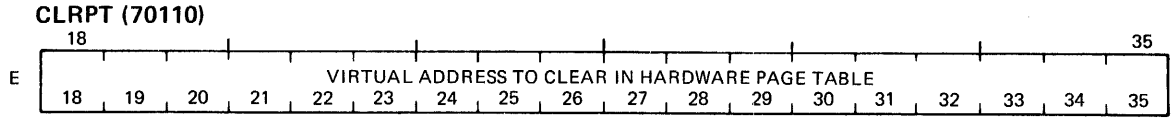
Figure A-5 RDPI Instruction



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

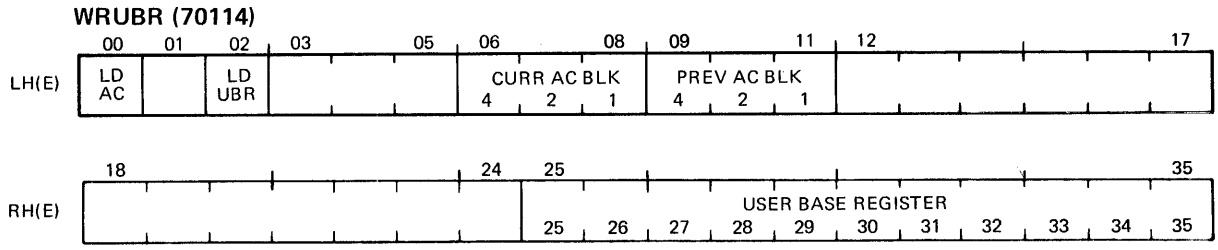
Figure A-6 RDUBR Instruction



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

MR-0235

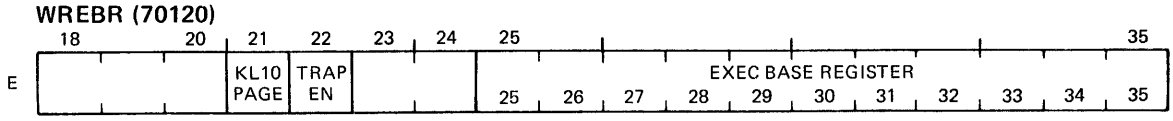
Figure A-7 CLRPT Instruction



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

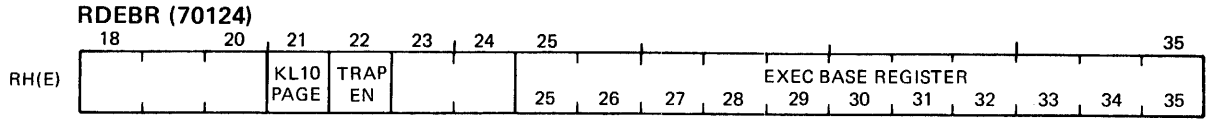
Figure A-8 WRUBR Instruction



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

MR-0237

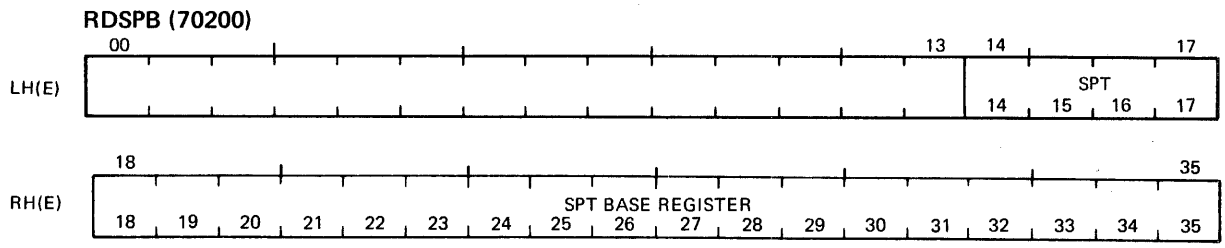
Figure A-9 WREBR Instruction



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

MR-0238

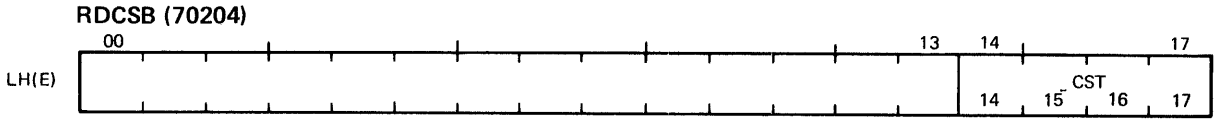
Figure A-10 RDEBR Instruction



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

MR-0239

Figure A-11 RDSPB Instruction



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

MR-0240

Figure A-12 RDCSB Instruction

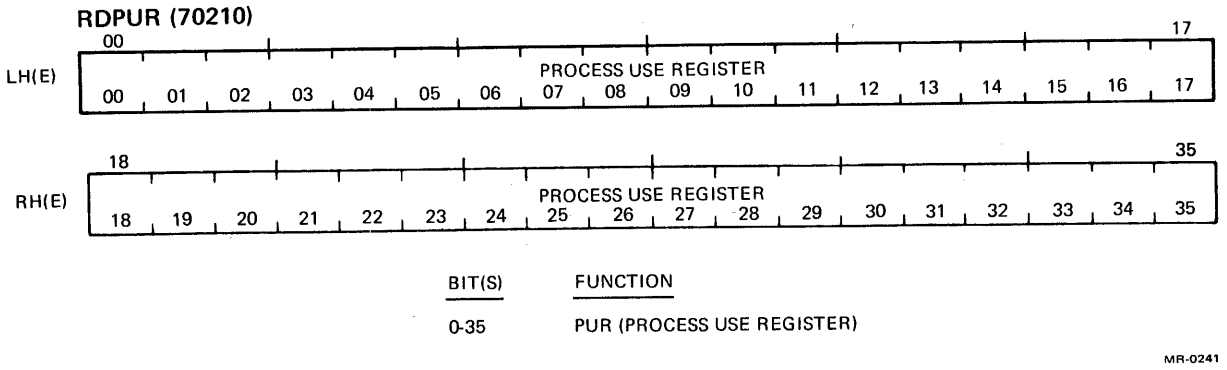


Figure A-13 RDPUR Instruction

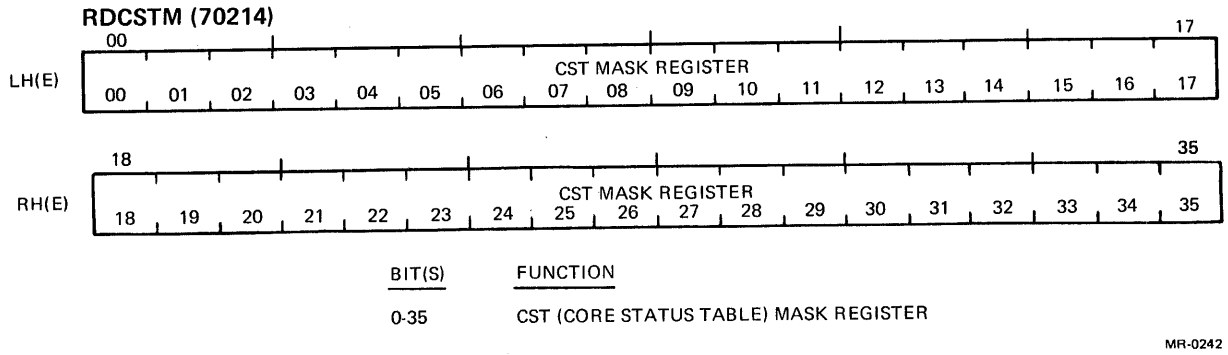


Figure A-14 RDCSTM Instruction

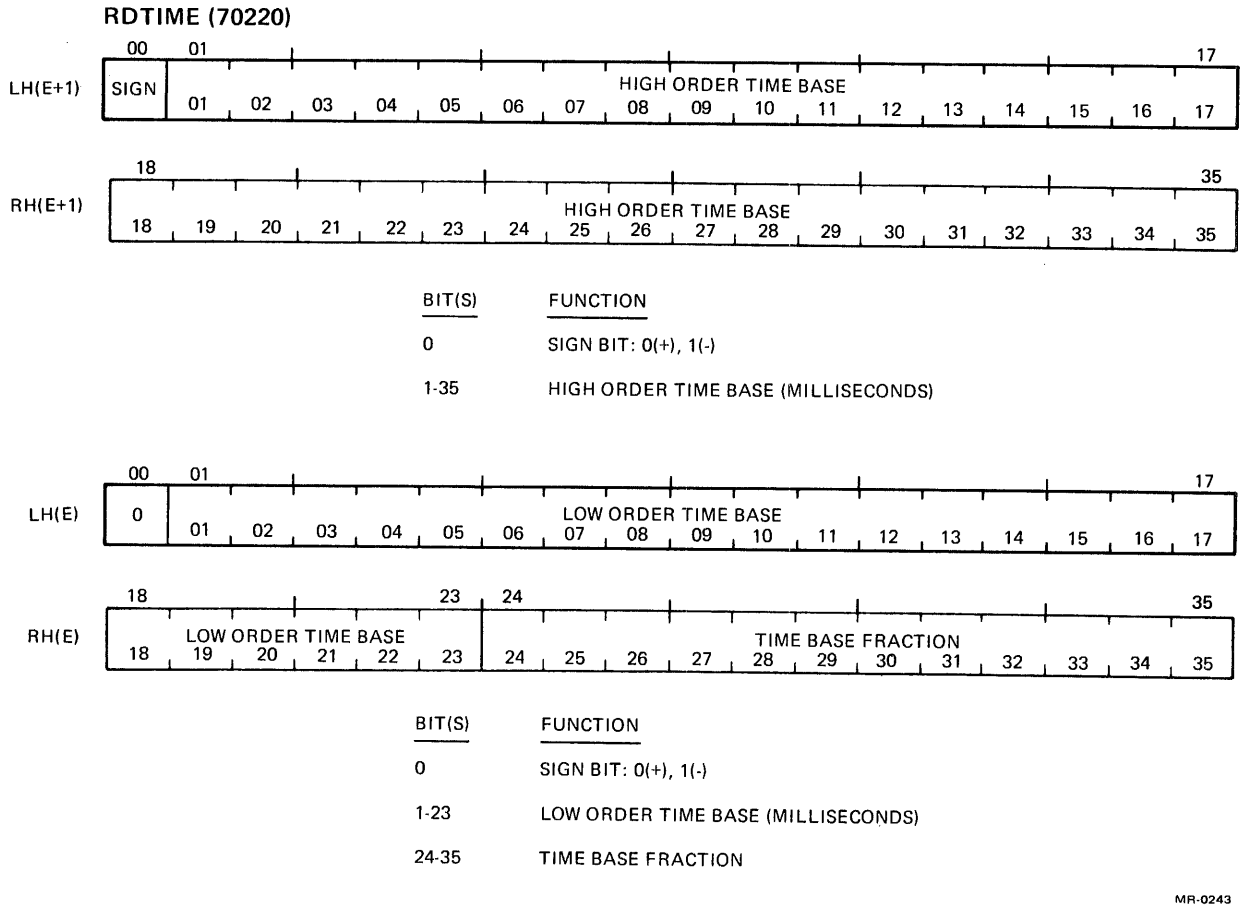


Figure A-15 RDTIME Instruction

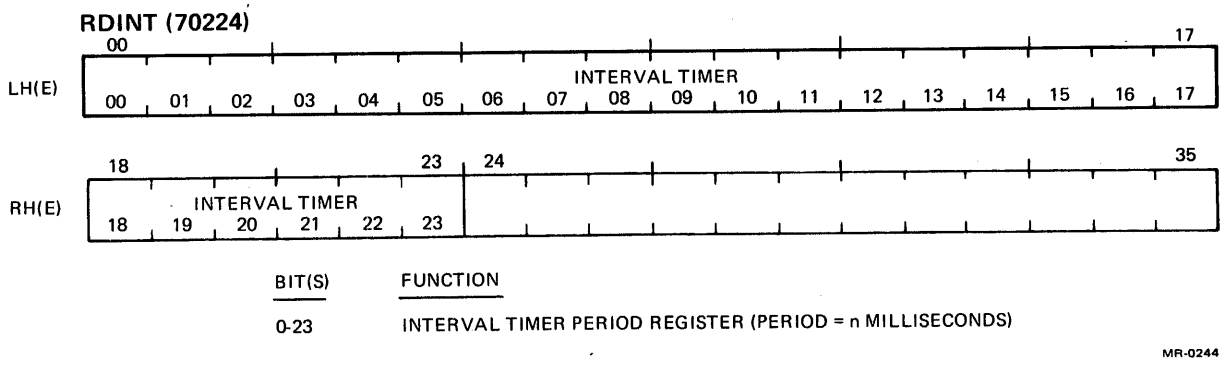


Figure A-16 RDINT Instruction

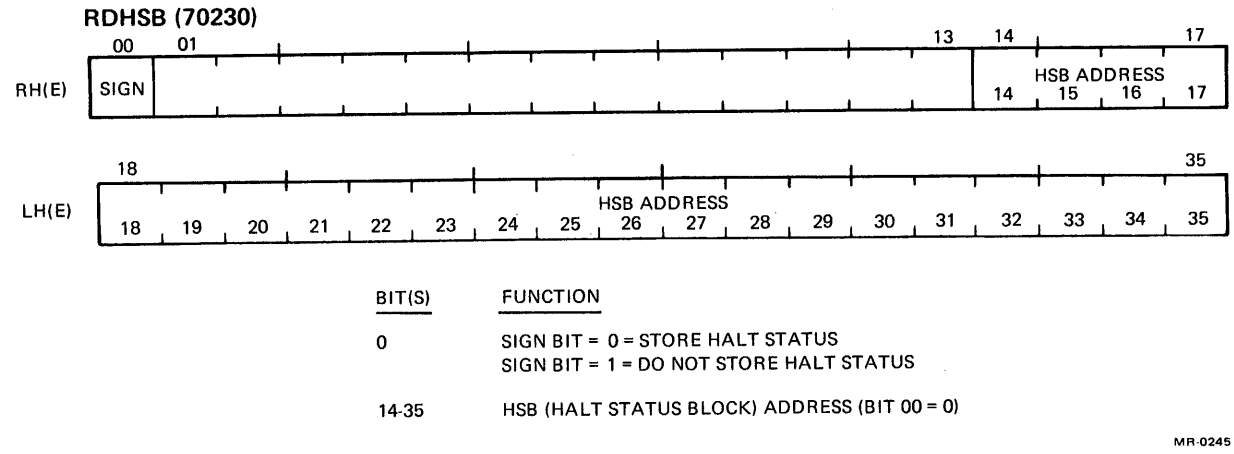


Figure A-17 RDHSB Instruction

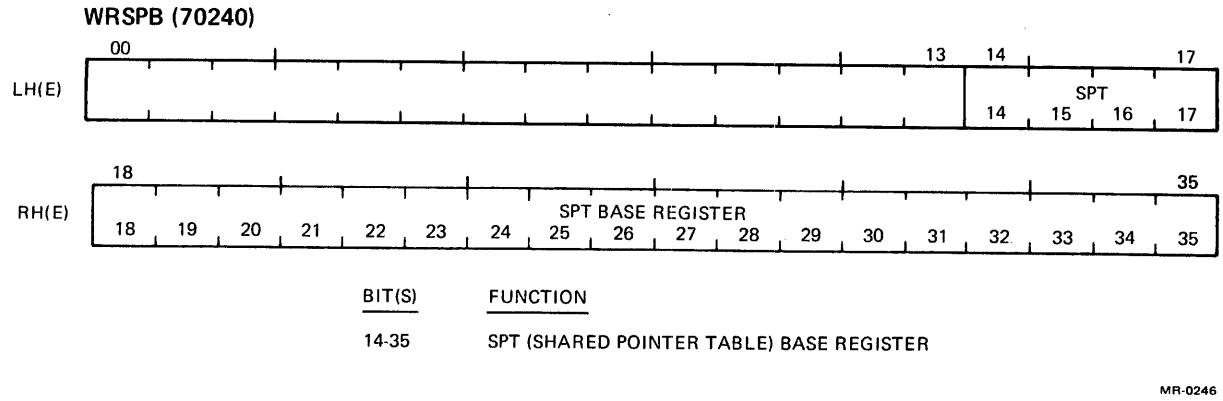


Figure A-18 WRSPB Instruction

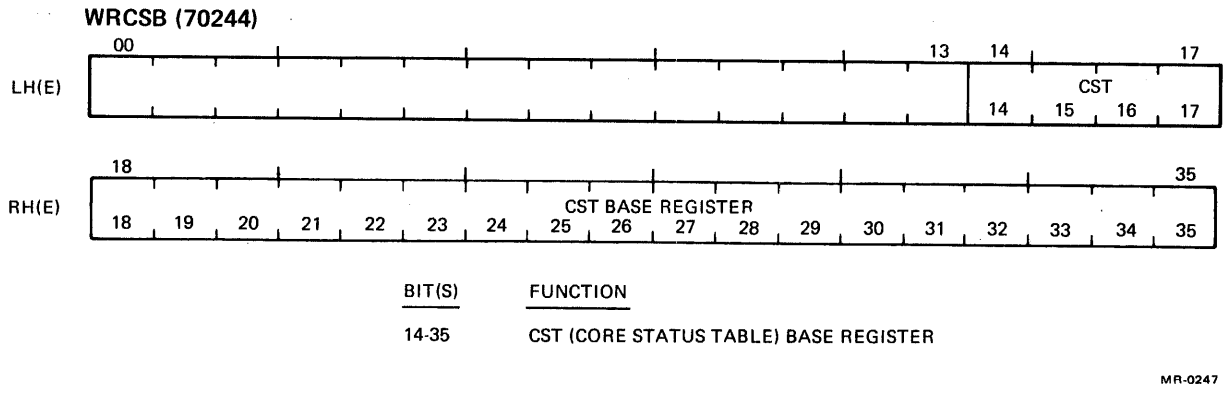


Figure A-19 WRCSB Instruction

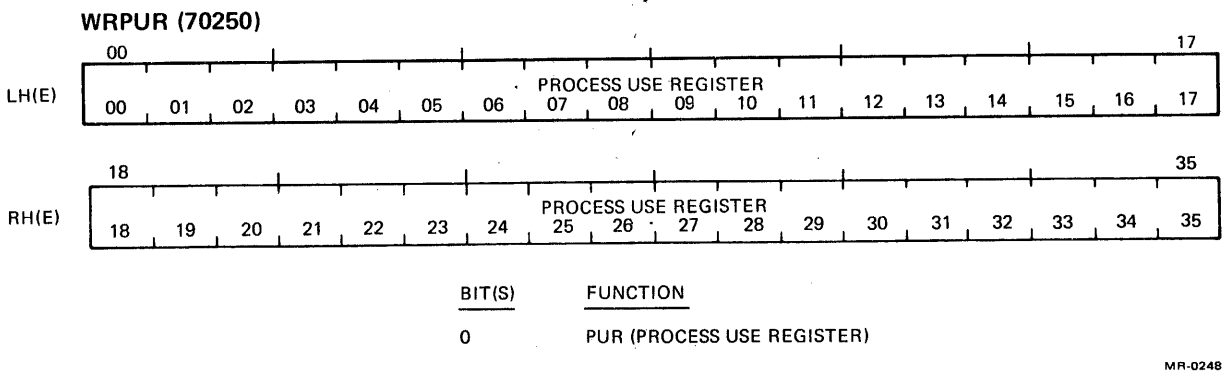


Figure A-20 WRPUR Instruction

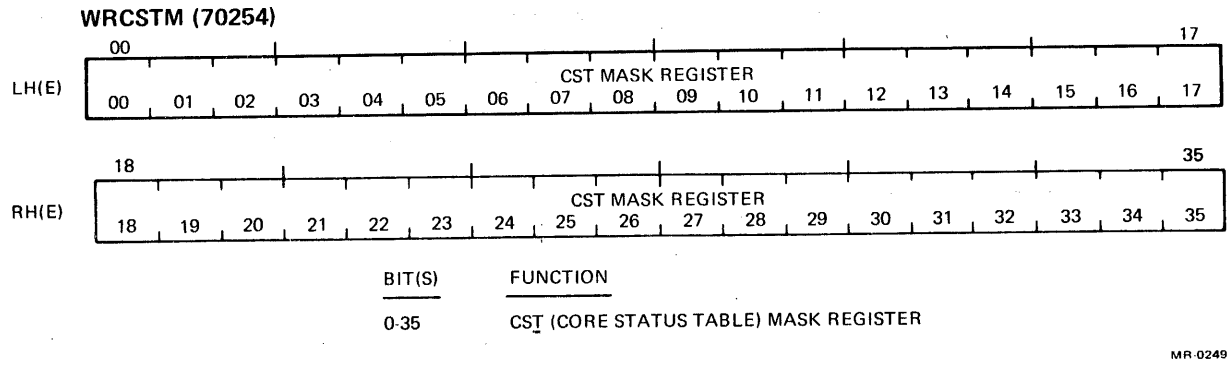


Figure A-21 WRCSTM Instruction

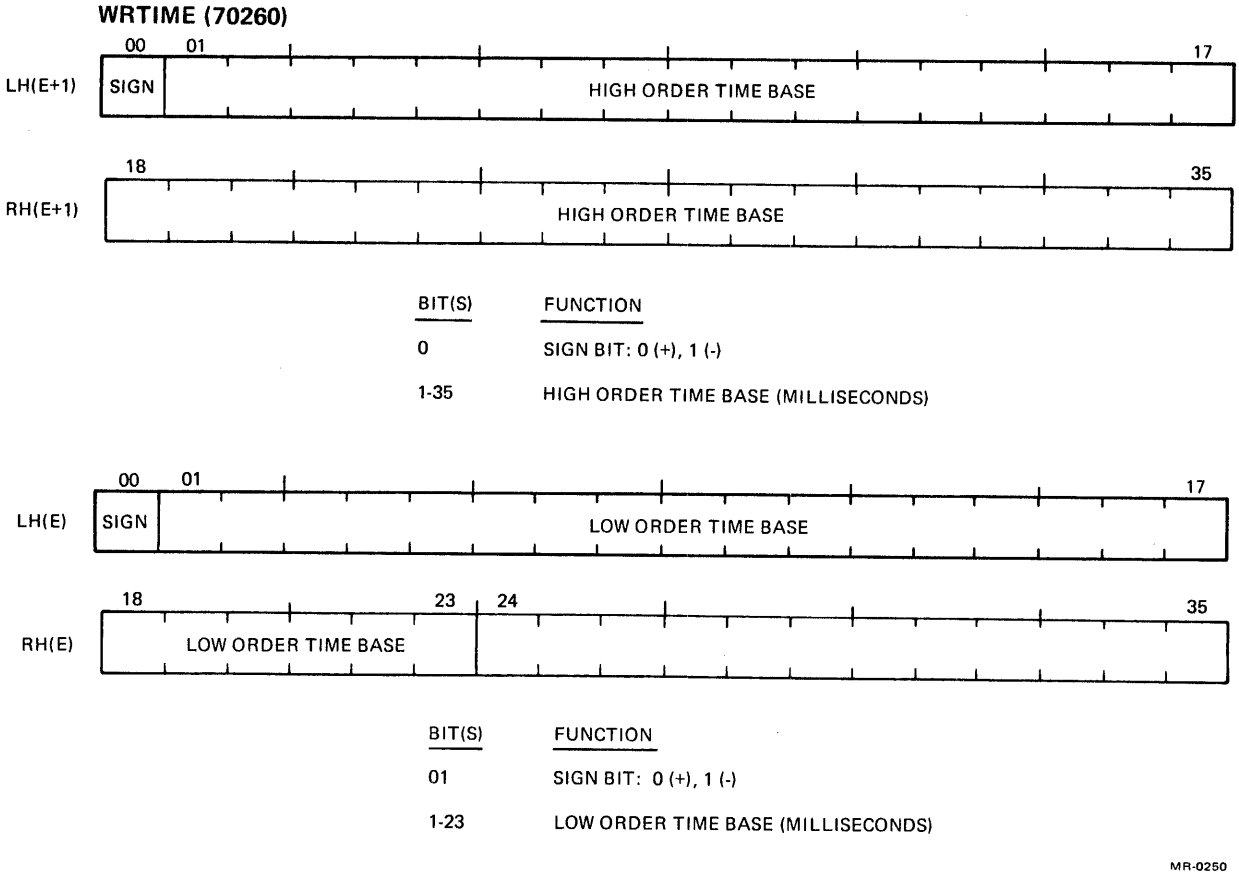


Figure A-22 WRTIME Instruction

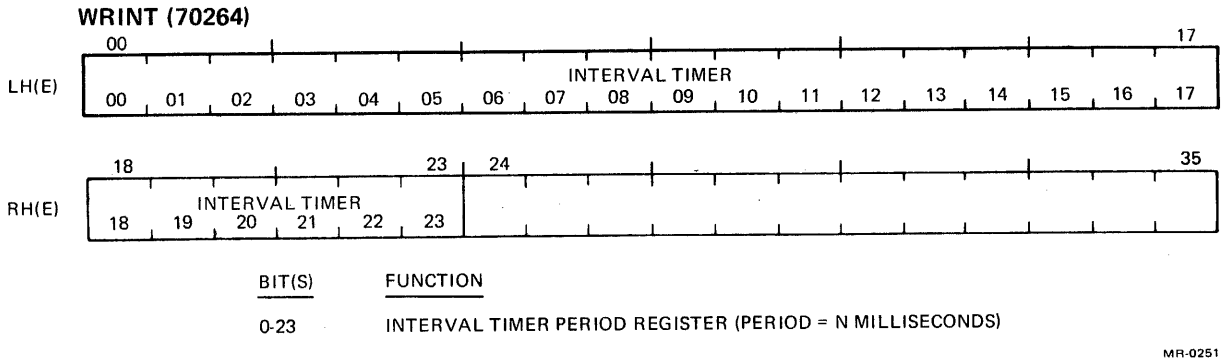


Figure A-23 WRINT Instruction

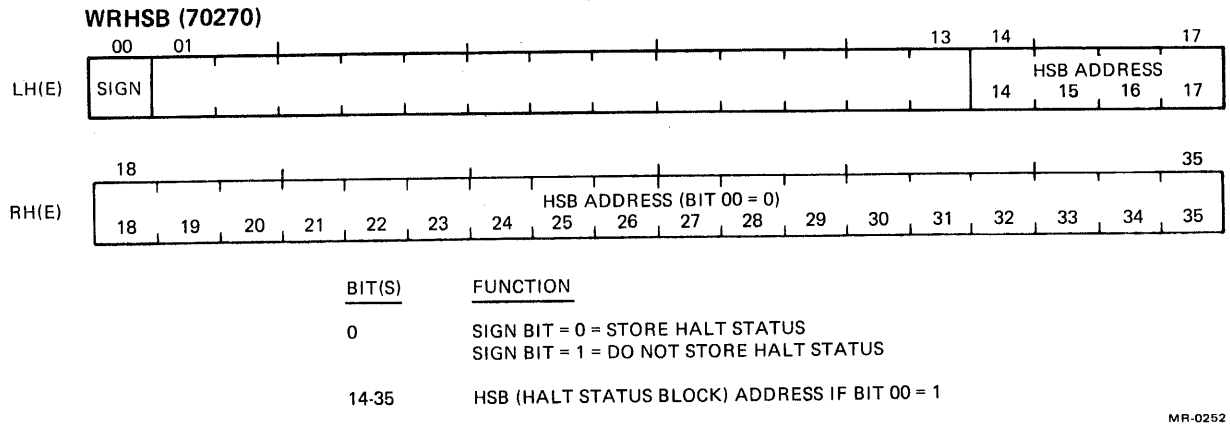


Figure A-24 WRHSB Instruction

A.3 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.

- **TIOE and TIOEB (710_8 and 720_8)** – 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_8 and 721_8)** – 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_8 and 722_8)** – 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_8 and 723_8)** – 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_8 and 724_8)** – 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_8 and 725_8)** – The BCIO (or BCIOB) instruction is similar to the BSIO (or BSIOB) instruction except that the word (or byte) read from the 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.

A.4 PXCT EXTENSIONS

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

- **UMOVE (704_8)** – The UMOVE (move from previous context) instruction performs the same functions as:

  ```
  PXCT 4, [MOVE AC,E]
  ```

- **UMOVEM (705_8)** – The UMOVEM (move to previous context) instruction performs the same function as:

  ```
  PXCT 4, [MOVEM AC,E]
  ```

APPENDIX B
KS10 UNIBUS

B.1 INTRODUCTION

I/O devices connect to the KS10 system via a Unibus and a unibus adapter (UBA) as shown in Figure B-1. (The UBA, which interfaces the Unibus to the KS10 bus, is described in Paragraph 5.9.) 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 B-2. Unibus signals are summarized in Table B-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.)**

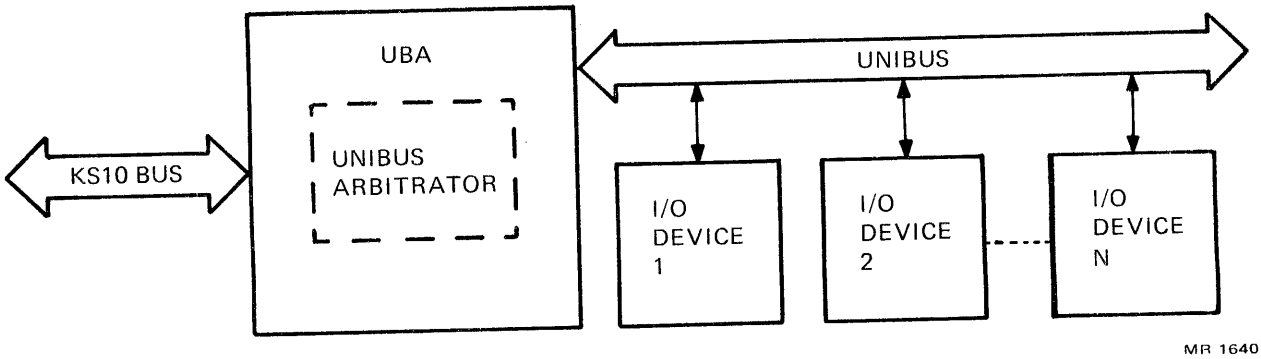


Figure B-1 KS10 Unibus Connection

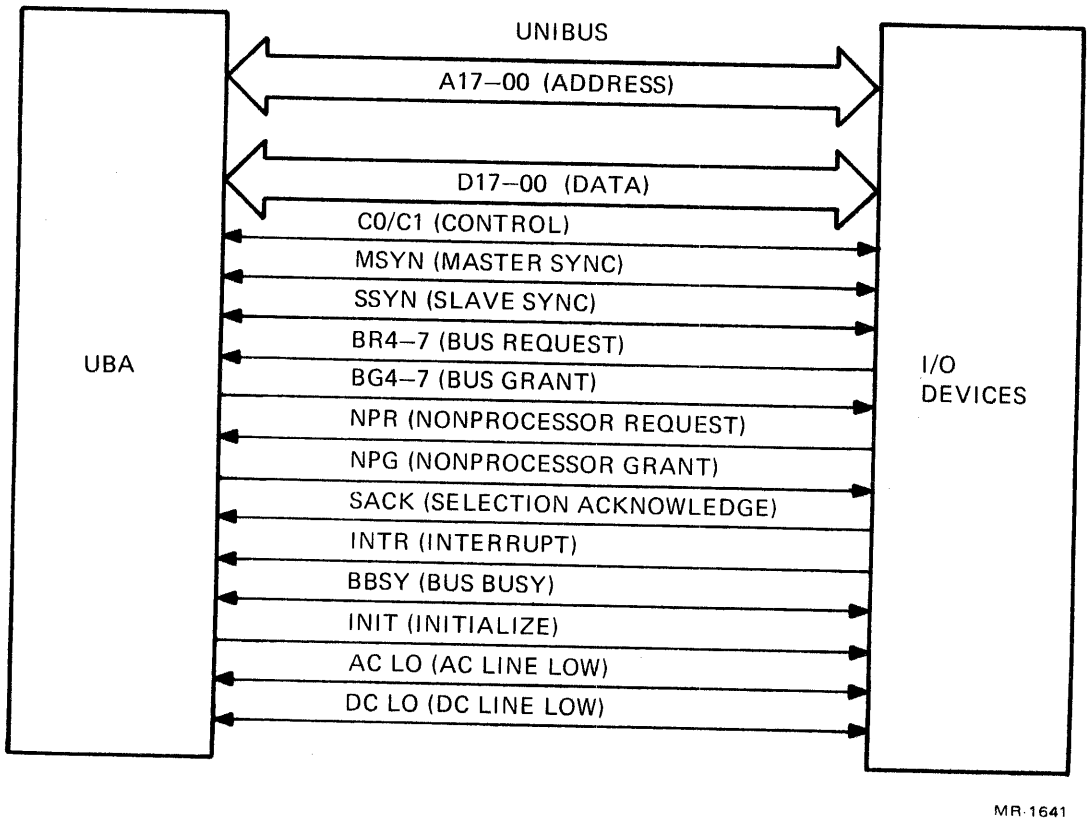


Figure B-2 Unibus Interface

Table B-1 Unibus Signal Summary

Signal(s)	Description
Address lines (A17–00)	Selects slave device register (UBA master) or memory address (device master).
Data lines (D17–00)	Transfers register data (UBA master) or NPR data (device master) between master and slave.
Control lines (C0/C1)	Specifies 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 MSYN.

Table B-1 Unibus Signal Summary (Cont)

Signal(s)	Description
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 request (NPR)	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 (AC LO)	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).

B.2 PRIORITY ARBITRATOR

The priority arbitrator for a Unibus connected to the KS10 system is located on the associated UBA. As shown in Figure B-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

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 B-4). If and when the Unibus is inactive (BBSY 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 B-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.

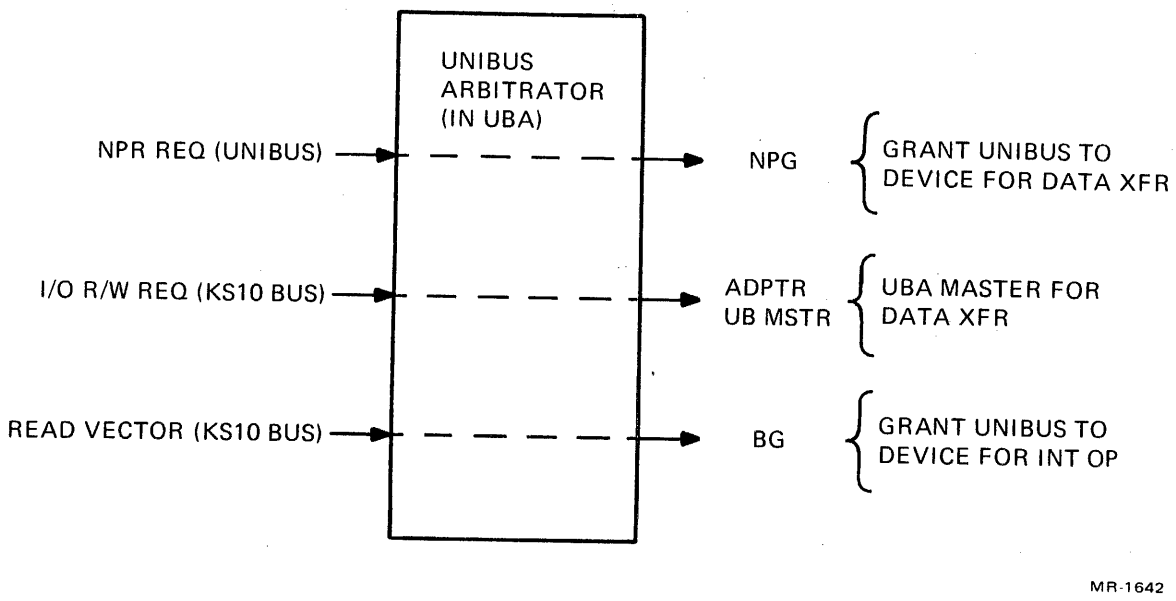


Figure B-3 Arbitrator Inputs/Outputs

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 (that is, 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.

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 B-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.

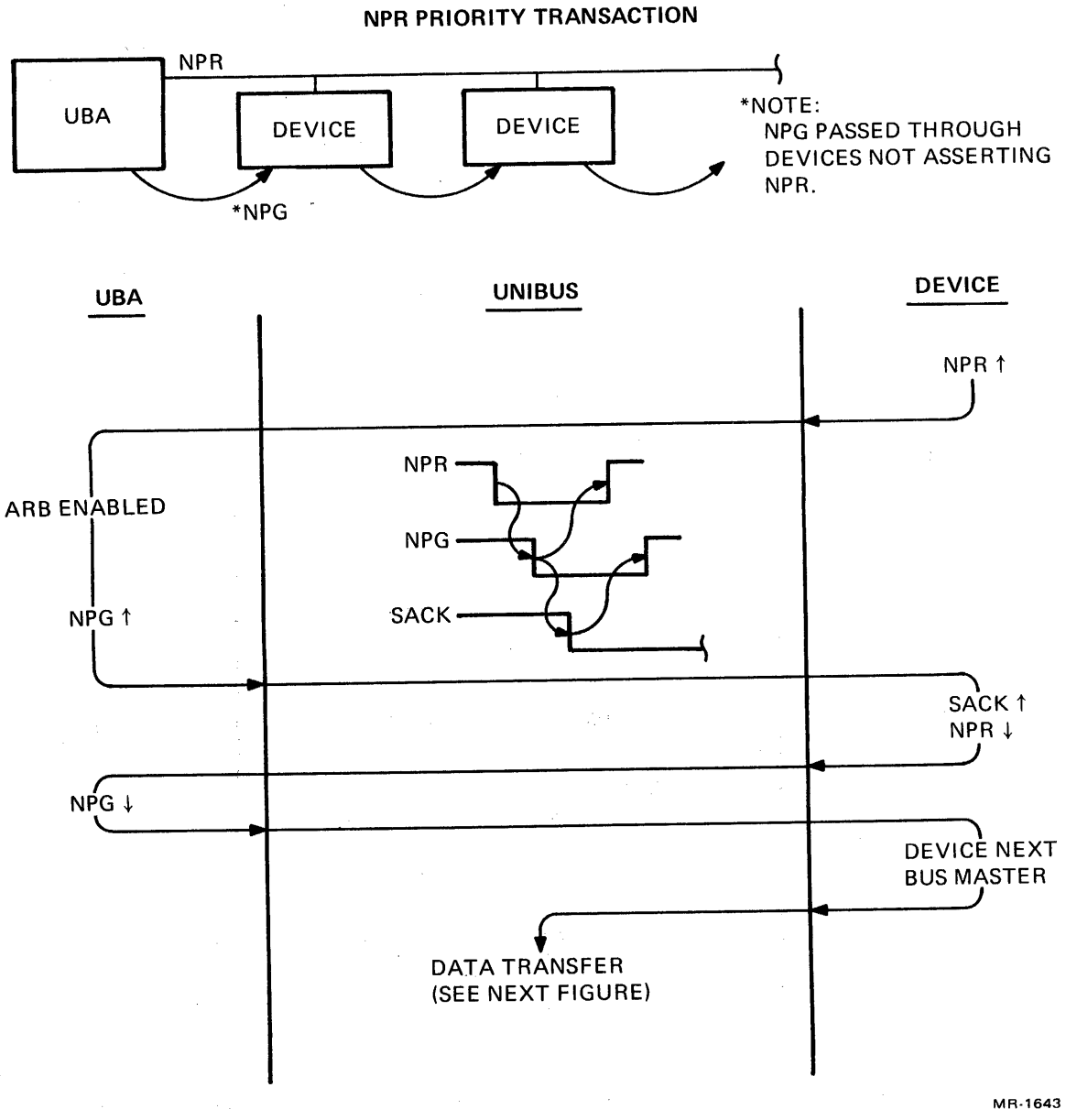


Figure B-4 NPR Priority Transaction

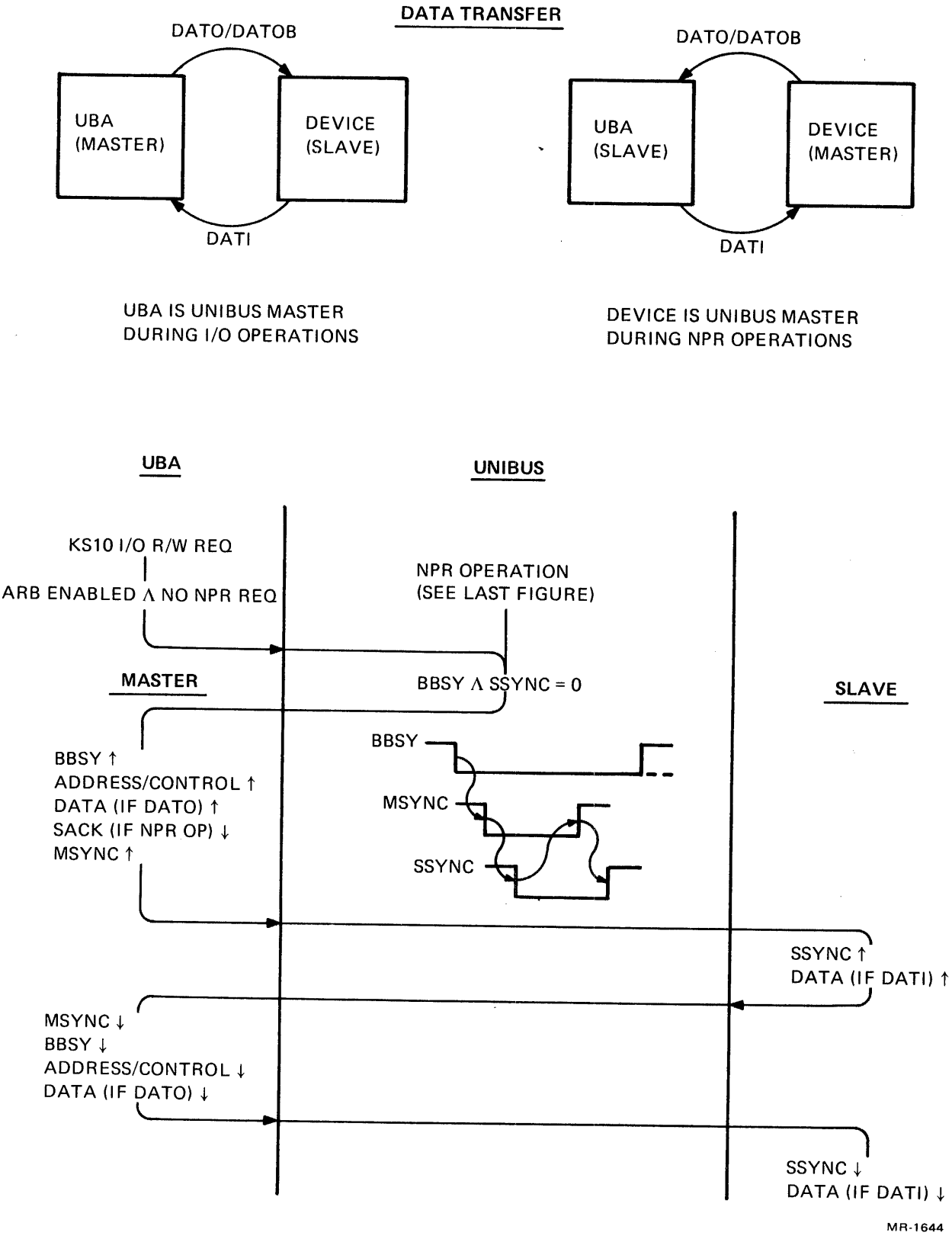


Figure B-5 Data Transfer Operation

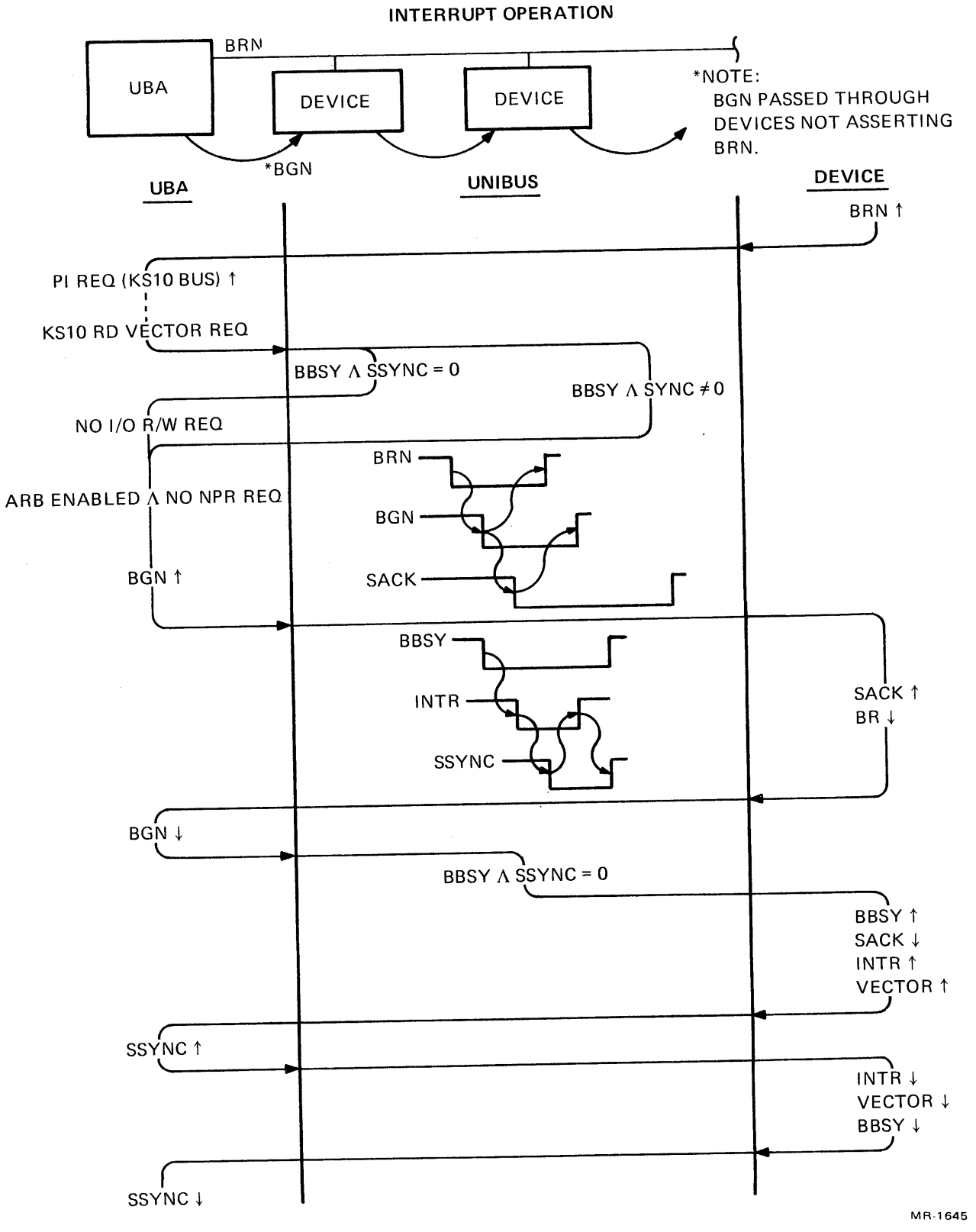


Figure B-6 Interrupt Operation

B.3 UNIBUS DRIVE INTERRUPT VECTORS AND BR LEVELS

Table B-2 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.

Table B-2 I/O (Unibus) Device Vectors and BR Levels

Device	UBA No.	PIA	Interrupt Vector (octal)	BR
RH11 #1 (RP06/RM03)	1	HI	254	6
RH11 #3 (TU45)	3	HI	224	6
LP20 #1	3	LO	754	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
CD11 #1	3	LO	230	4

B.4 UNIBUS (UBA AND DEVICE) REGISTER ADDRESSES

UBA and Unibus device register addresses for a fully configured system are listed in Table B-3. Note that the device register addresses given are base addresses only.

Table B-3 Unibus Register Addresses

Device	UBA	CTL No.	Register Address (octal)	Register
	UBA1	1	763000–77	Paging RAM
	UBA1	1	763100	Status Register
	UBA1	1	763101	Maintenance Register
RH11 #1 (RP06/RM03)	UBA1	1	776700*	Control & Status Register 1
	UBA3	3	763000–77	Paging RAM
	UBA3	3	763100	Status Register
	UBA3	3	763101	Maintenance Register
RH11 #3 (TU45)	UBA3	3	772440*	Control & Status Register 1

Table B-3 Unibus Register Addresses (Cont)

Device	UBA	CTL No.	Register Address (octal)	Register
LP20 #1	UBA3	3	775400*	Control & Status Register A
DZ11 #1	UBA3	3	760010*	Control & Status Register
DZ11 #2	UBA3	3	760020*	Control & Status Register
DZ11 #3	UBA3	3	760030*	Control & Status Register
DZ11 #4	UBA3	3	760040*	Control & Status Register
KMC11 #1	UBA3	3	760540	Maintenance Register
DUP11 #1	UBA3	3	760300*	Control & Status Register
DUP11 #2	UBA3	3	760310*	Control & Status Register
CD11 #1	UBA3	3	777160*	Control & Status Register

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

APPENDIX C
MICROCODE OPERATION

C.1 INTRODUCTION

The KS10 microcode is contained in the 96-bit × 2K CRAM located on the CPU's microprocessor modules. KS10 microcode operation is shown in Figure C-1. The microcode performs the following basic functions.

- Processor initialization
- Machine halt sequence
- KS10 system-level instruction execution
- NICOND dispatch
- Page fail handling

C.2 PROCESSOR INITIALIZATION

After loading the microcode during the system bootstrap operation, the console sets the CRAM address in the microprocessor to all 0s and starts the CPU clock to begin microcode execution. The first section of microcode executed (starting at CRAM address 0000_8) serves only to initialize the CPU; it is not executed again until the next time the console bootstraps the system. The initialization code clears (or sets to some initial value) the various flags and registers in the processor. It then forces the machine to exec mode and enters the halt loop.

C.3 HALT SEQUENCE

During the halt sequence, the microcode writes the halt status block (optional), the halt status word, and the PC in memory; it then enters the halt loop. The halt loop, which is the idle state for the CPU (KS10 program halted), is the repeated execution of CRAM address 0005_8. The microcode remains in the halt loop until a CONTINUE signal is asserted by the console.

When the console asserts CONTINUE, it may also assert an EXECUTE signal that determines from where the first instruction will be fetched after the microcode leaves the halt loop. If EXECUTE is asserted, the CPU fetches the first instruction from the console's instruction register. The fetch is over the KS10 bus by means of an I/O register read operation. If EXECUTE is not asserted, the instruction is fetched from memory. In this case, the fetch is a memory read operation over the KS10 bus to the memory address specified by the PC.

During system bootstrap, after the microcode has been started (to initialize the CPU) and has entered the halt loop, the console asserts both EXECUTE and CONTINUE to begin KS10 program execution. The console's instruction register is loaded previously (by the console itself) with a JRST (jump) to the first memory address in the monitor boot program. (The boot program is loaded in the KS10 memory beginning at address 1000_8 prior to starting the microcode.) Starting execution of the monitor boot program brings in the system monitor (under operator control) and completes the system bootstrap procedure.

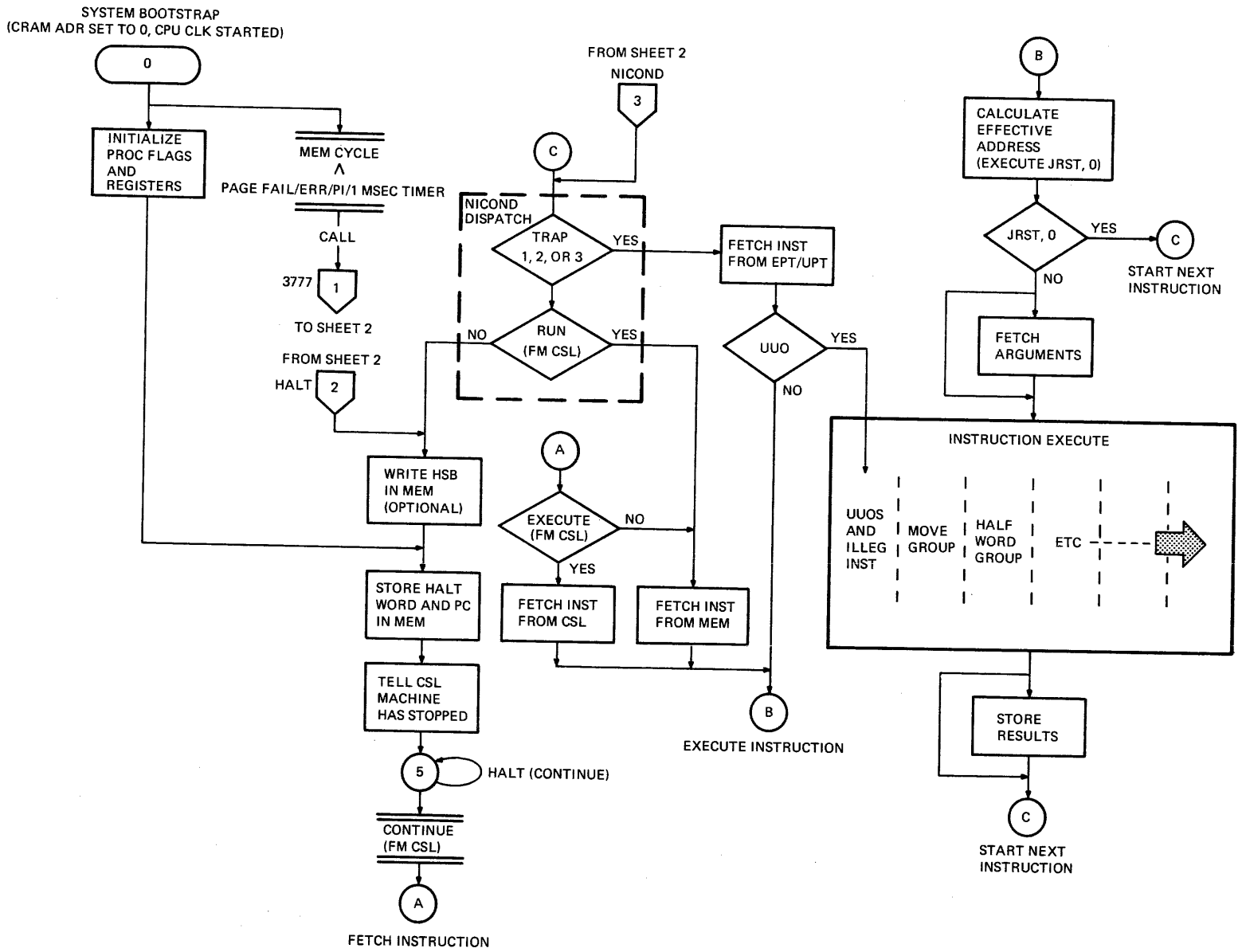


Figure C-1 Basic Microcode Operation (Sheet 1 of 2)

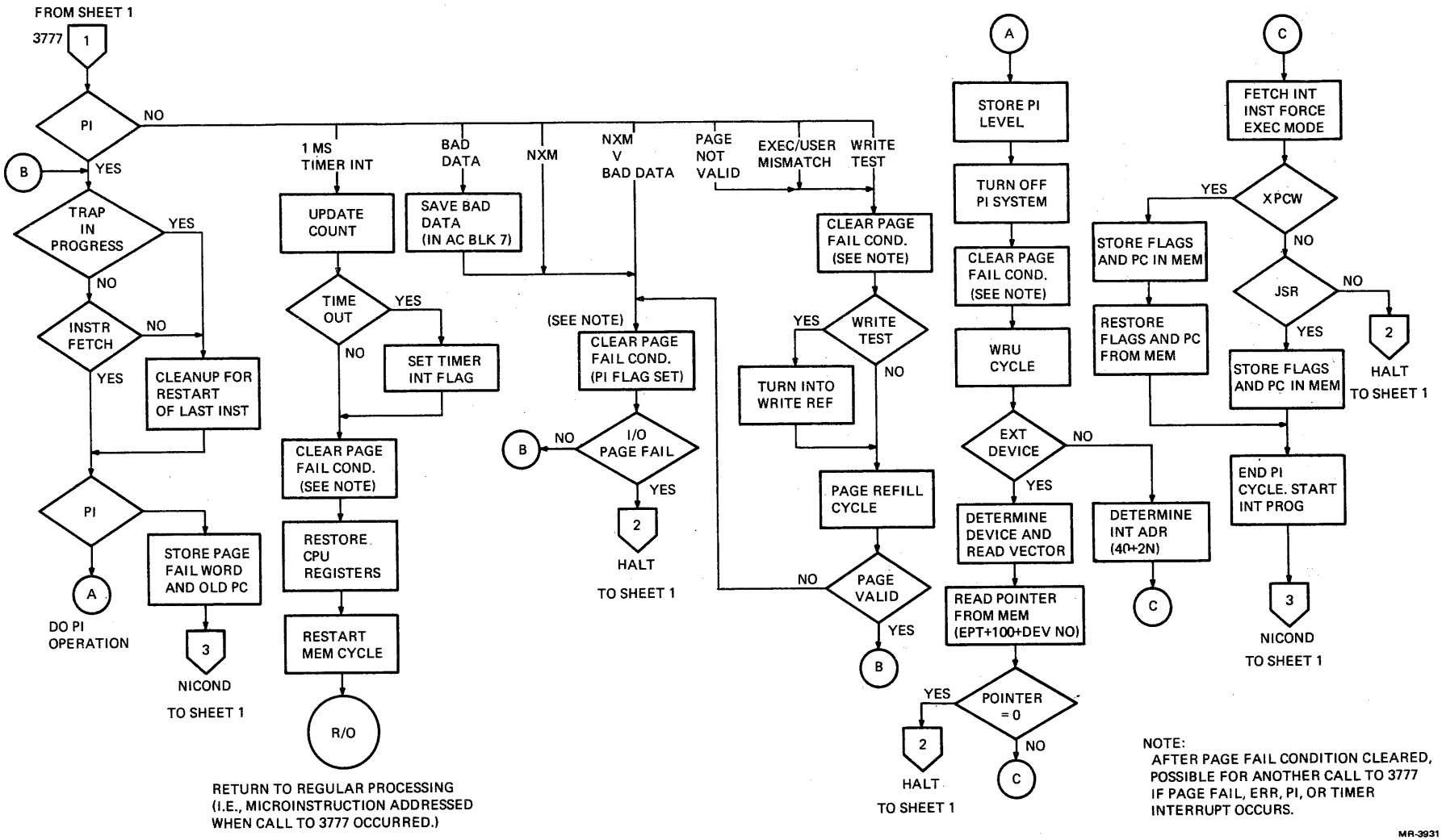


Figure C-1 Basic Microcode Operation (Sheet 2 of 2)

KS10 PROGRAM EXECUTION

Once KS10 program execution has been started, one KS10 system-level instruction after the other is fetched from memory and executed by the microcode. As shown in Figure C-1, microcode flow is generally the same for each instruction. After the instruction fetch, there is an effective address calculation followed by an argument fetch for certain instruction types. The instruction is then executed and the results stored. After the execution of each KS10 instruction, the microcode does a NICOND dispatch.

NICOND DISPATCH

The NICOND dispatch provides for either halting KS10 program execution, or for interrupting KS10 program execution in order to dispatch to trap handling microcode. A trap can occur for an arithmetic overflow (trap 1), a stack overflow (trap 2), or by program request (trap 3). A KS10 program halt is caused by the console-generated RUN signal being false. If RUN is equal to 0 at the end of the instruction, the microcode enters the halt sequence.

The RUN signal is normally asserted by the console when KS10 program execution is started; that is, it is asserted with CONTINUE during system bootstrap or by the start (ST) or continue (CO) console commands. EXECUTE is also asserted by the ST command, causing the CPU to fetch a console-generated JRST that specifies the starting address.

RUN is negated by the halt (HA) console command. Also, RUN is not asserted with CONTINUE for the single instruct (SI) or execute (EX) console commands. Therefore the microcode enters the halt loop immediately after a single instruction is executed. The EXECUTE signal is also asserted during the EX command, causing the CPU to fetch the single instruction to be executed from the console.

PAGE FAIL HANDLING

In addition to the possibility of a NICOND dispatch interrupting KS10 program execution at the end of each KS10 instruction, the microcode sequence during the actual execution of a KS10 instruction may be interrupted by a PAGE FAIL signal. PAGE FAIL, which is generated in the CPU, causes a subroutine call in the microcode to CRAM address 3777_8 at the beginning of the next CPU-generated memory cycle. CRAM address 3777_8, which is the last location in the CRAM, causes a jump to the microcode's page fail handler.

PAGE FAIL is not only generated for an actual page failure; it is also generated so that the page fail handler may service KS10 priority interrupts, hard memory errors detected by the CPU, nonexistent memory errors detected by the CPU, and a 1 ms count by the binary counter for the interval timer. For all conditions except the interval timer interrupt, the KS10 instruction being executed at the time of the PAGE FAIL is restarted following execution of the page fail handler routine. For the 1 ms interrupt, which causes the page fail handler to update the interval timer (contained in RAM file working locations), a return is made to the microcode being executed at the time of the trap to 3777_8.

Reader's Comments

KS10-BASED DECSYSTEM-2020
TECHNICAL MANUAL
EK-0KS10-TM-002

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 Circulating Service (NR2/M15)
Customer Services Section

Order No. EK-0KS10-TM

Fold Here

Do Not Tear – Fold Here and Staple

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MA.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services Development and Publishing
200 Forest Street (MR1-2/T17)
Marlboro, MA 01752