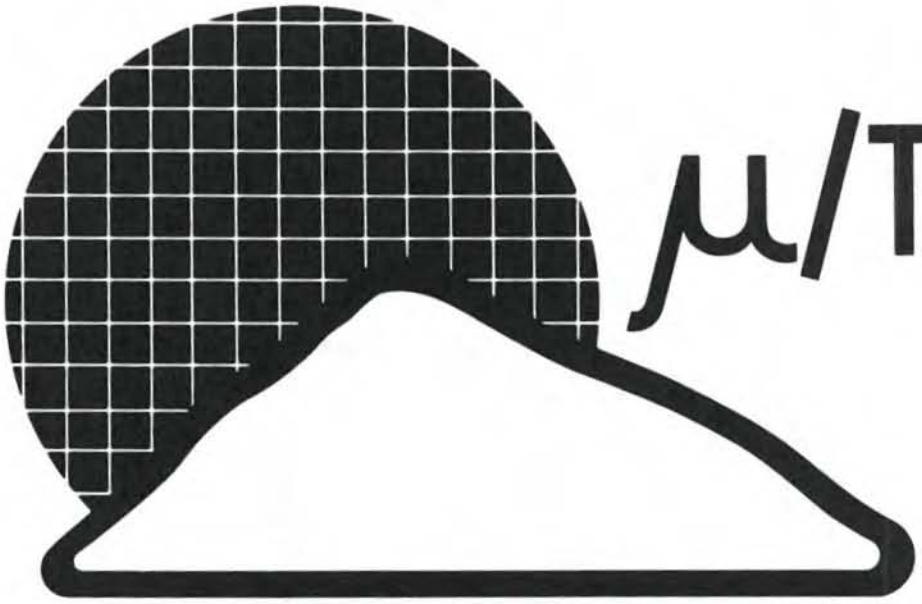
EK-DCT11-UG-002



μ/T-11

USER'S GUIDE

SECOND EDITION JANUARY 1982

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

CONTENTS

CHAPTER 3 PIN DESCRIPTION

CHAPTER 4 MODE SELECTION

FIGURES

CHAPTER 1 ARCHITECTURE

CHAPTER 2 BUS TRANSACTIONS

CHAPTER 3 PIN DESCRIPTION

CHAPTER 4 MODE SELECTION

CHAPTER 5 INTERFACING

CHAPTER 6 ADDRESSING MODES AND INSTRUCTION SET

APPENDIX A

APPENDIX B

TABLES

CHAPTER 1 ARCHITECTURE

CHAPTER 2 BUS TRANSACTIONS

PREFACE

This user's guide is designed for engineers familiar with PDP-11 architecture. Chapters one through six offer a tutorial on DCT11-AA architecture and operation (Chapter five includes some design examples). The appendix is exclusively reference material such as timing diagrams and instruction set tables.

The guide can be used by both hardware and software specialists. The hardware specialist should become familiar with Chapters one through five, whereas the software specialist should become familiar with Chapters one, four, and six.

One of the characteristics of the DCT11-AA is that it can be user-programmed to operate in a variety of modes, which affect both its functionality and timing. Chapter two (Bus Transactions) and Chapter three (Pin Description) are broken down by mode. This is done to allow the user to find, in one place, all the information relevant to the selected mode.

If the user does not know which mode to use for a given application it is suggested that Chapter four (Mode Selection) be read first.

Appendix B briefly lists the software differences and compatibilities of the DCT11-AA and other members of the PDP-11 family.

CHAPTER 1

ARCHITECTURE

1.1 INTRODUCTION

This chapter contains a description of the internal DCT11-AA architecture. The chapter is divided into five sections covering all aspects of the architecture. The sections are:

- Registers
- Arithmetic & logic unit (ALU)
- DCT11-AA hardware stack
- Interrupts
- DMA mechanism.

1.2 REGISTERS

With reference to Figure 1-1, DCT11-AA contains a number of internal registers which are used for various purposes. The registers are broken up into three groups:

- General
- Status
- Mode.

1.2.1 General Registers

The DCT11-AA microprocessor contains eight 16-bit general-purpose registers that can perform a variety of functions. These registers can serve as accumulators, index registers, autoincrement registers, autodecrement registers, or as stack pointers for temporary storage of data. Arithmetic operations can be from one general register to another, from one memory location or device register to another, or between memory locations or a device register and a general register. The eight 16-bit general registers (R0 through R7) are identified in Figure 1-2.

Registers R6 and R7, in the DCT11-AA, are dedicated. R6 serves as the Stack Pointer (SP) and contains the location (address) of the last entry in the stack. Register R7 serves as the processor Program Counter (PC) and contains the address of the next instruction to be executed. It is normally used for addressing purposes only and not as an accumulator.

1.2.2 Status Register

The Processor Status Word (PSW) contains information on the current processor status. This information includes the current processor priority, the condition codes describing the arithmetic or logic results of the last instruction, and an indicator for detecting the execution of an instruction to be trapped during program debugging. This indicator (the T-bit) cannot be directly set or cleared. The T-bit can only be set or cleared when entering or exiting an interrupt routine. The PSW format is shown in Figure 1-3 certain instructions allow programmed manipulation of condition code bits and loading and storing (moving) the processor status.

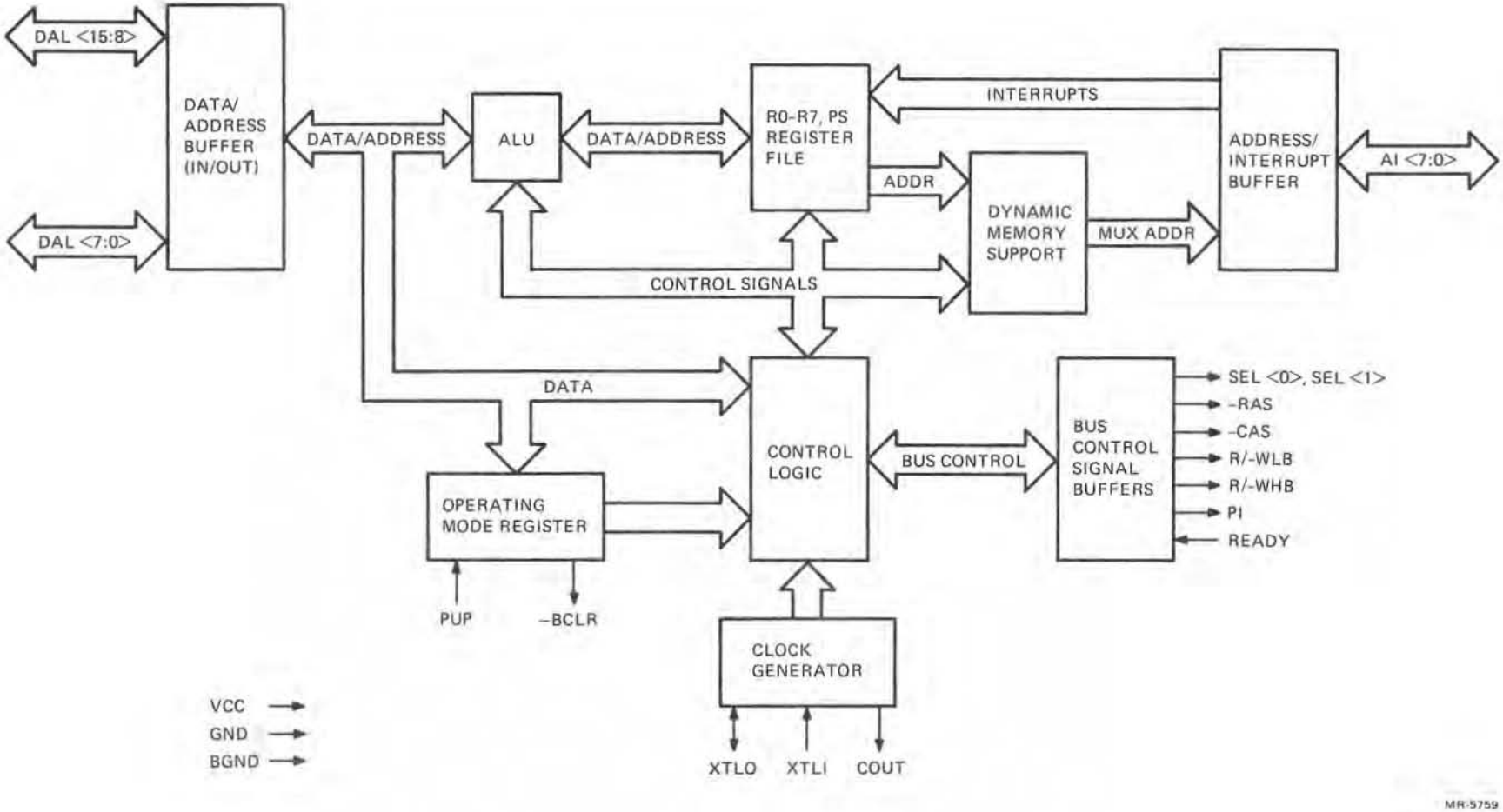


Figure 1-1 DCT11-AA Block Diagram

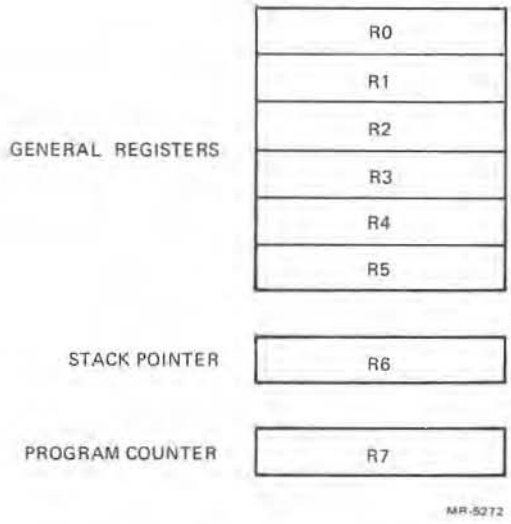


Figure 1-2 Registers

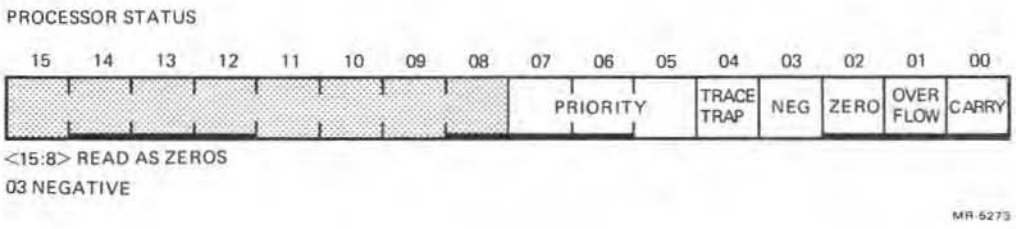


Figure 1-3 Processor Status Word

1.2.3 Mode Register

With reference to Figure 1-4, the DCT11-AA incorporates a user loadable mode register. The mode register is loaded at the time of power up. The user has the option of selecting any combination of the following modes:

- 16-bit or 8-bit data bus
- Dynamic or static memory support
- 64K or 4K/16K dynamic memory support
- Constant or processor clock
- Long or standard microcycle
- Normal or delayed read/write timing
- Tester or user operation
- 1 of 8 start/restart address pairs.

A complete discussion of the mode register is contained in Chapter 4.

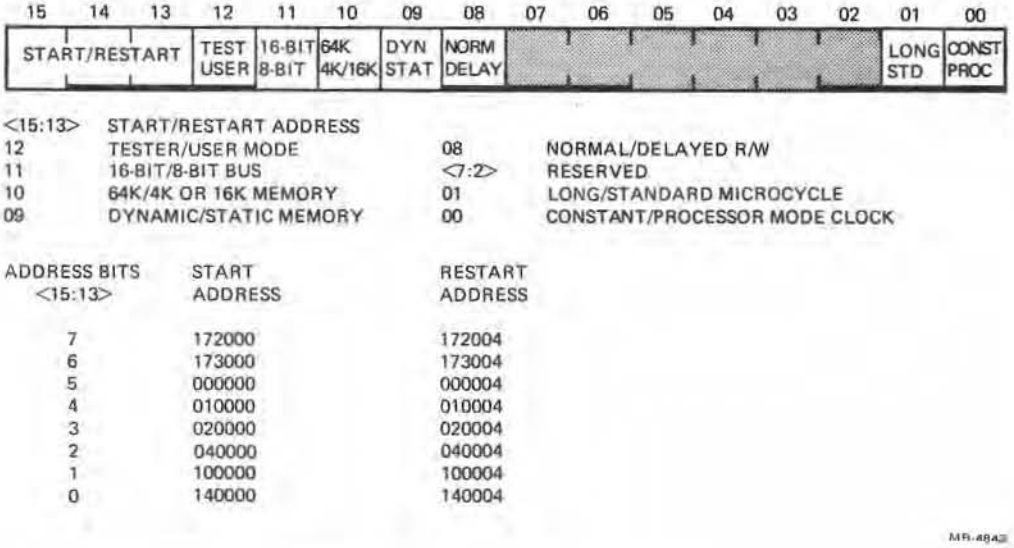


Figure 1-4 Mode Register

1.3 ARITHMETIC & LOGIC UNIT (ALU)

Arithmetic and logical instructions of the 16-bit CPU are executed in the ALU. The ALU internally communicates with registers and buffers in order to execute instructions.

1.4 DCT11-AA HARDWARE STACK

The hardware stack is part of the basic design architecture of the DCT11-AA. It is an area of memory set aside by the programmer or by the operating system for temporary storage and linkage. It is handled on a LIFO (last in/first out) basis, where items are retrieved in the reverse of the order they were stored. On the DCT11-AA the stack starts at the highest location reserved for it (376 octal at power up) and expands linearly downward to a lower address as items are added to the stack.

It is not necessary to keep track of the actual locations into which data is being stacked. This is done automatically through the use of the Stack Pointer (SP). Register six (R6) always contains the memory address where the last item is stored in the stack. Instructions associated with subroutine linkage and interrupt service automatically use register six as the hardware stack pointer. For this reason, R6 is frequently referred to as the system SP. The hardware stack is organized in full word units only.

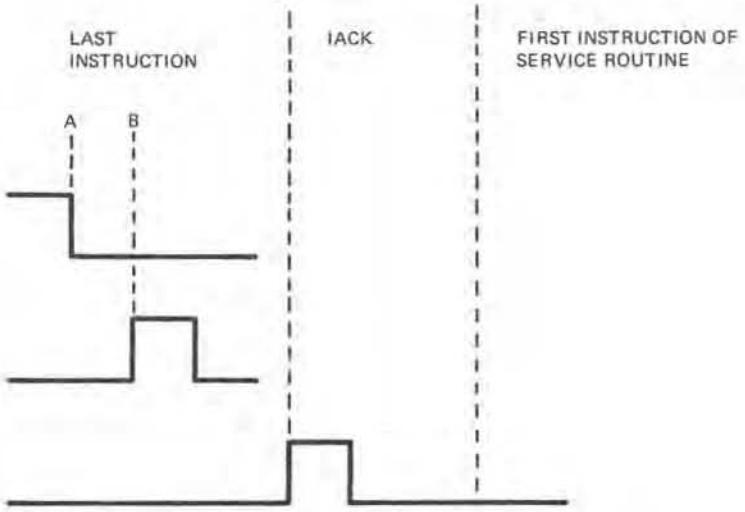
1.5 INTERRUPTS

Interrupts are requests, made by peripheral devices, which cause the processor to temporarily suspend its present program execution to service the requesting device. A device can interrupt the processor only when its priority is higher than the processor priority indicated by PSW<7:5>.

DCT11-AA supports a vectored interrupt structure (with optional internally generated vector addresses) with priority on four levels encoded on four lines. In addition, on separate pins it supports two non-maskable interrupts Power Fail (-PF) and -HALT.

1.5.1 Interrupt Mechanism

With reference to Figure 1-5, when the DCT11-AA receives an interrupt no action is taken until the end of the current instruction. Interrupts are only read during a read transaction or ASPI transaction. Before fetching the next instruction, the DCT11-AA arbitrates the interrupt priority. If the interrupt request has a higher priority than the processors, it initiates an Interrupt Acknowledge (Iack) transaction (Refer to paragraph 2.12.). Following the Iack transaction, the current PC and PSW are saved on the stack and the new PC and PSW are loaded from the vector address.



A. INTERRUPT REQUEST
B. INTERRUPT REQUEST LATCHED INTO DCT11-AA

MR-4997

Figure 1-5 Interrupt Request

1.5.2 Interrupt Posting

With the assertion of the signal Priority In (PI), interrupts are read into the DCT11-AA during any read transaction and Assert Priority In (ASPI) transaction.

Interrupts are read in only at the occurrence of PI.

1.5.3 Interrupt Request (IRQ)

With reference to Figures 1-5 and 1-6, during the assertion of PI the interrupt request is read by DCT11-AA. Refer to Table 1-1 for signal names. Interrupt requests are implemented from the following seven different signals;

Maskable Interrupts:

- -CP<3:0> (Coded Priority).

Non-maskable Interrupt:

- -PF (Power Fail)
- -HALT (Halt).

Control (Internal or External) Vector:

- -VEC (Vector).

DCT11-AA detects an interrupt request if, during the assertion of PI, at least one of the following signals is asserted low:

- -CP<3> (AI<1>)
- -CP<2> (AI<2>)
- -CP<1> (AI<3>)
- -CP<0> (AI<4>)
- -PF (AI<6>)
- -HALT (AI<7>).

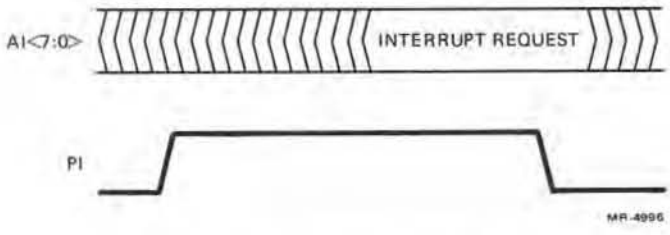


Figure 1-6 Interrupt Timing

Table 1-1 Interrupt Signals

Interrupt Signals	Pin Name	Pin Number
-CP<3>	AI<1>	33
-CP<2>	AI<2>	34
-CP<1>	AI<3>	35
-CP<0>	AI<4>	36
-VEC	AI<5>	37
-PF	AI<6>	38
-HALT	AI<7>	39

1.5.4 Vectors

Every interrupt except -HALT is associated with an interrupt vector. The interrupt vector consists of two words: the next PC and next PSW. The PC is the address of the routine to service an interrupt device. The PSW has new information to load into the processor status register. After the Iack transaction, the current PC and PSW are saved on the stack and the new PC and PSW are loaded from the vector address.

Up to 64 vectors may reside in the first 256 memory locations (octal 374 is the highest vector location). The vector address is provided by the interrupting device (external vector address) or by a fixed table stored in the DCT11-AA (internal vector address).

> NOTE
> The Power Fail (-PF) interrupt uses interrupt vector address 24 and is not acknowledged with an Iack transaction (refer to paragraph 2.12). -HALT interrupt is not associated with a vector, it pushes the PC and PSW onto the stack and immediately goes to the restart address with PSW 340. -HALT is not acknowledged.

1.5.4.1 Internal Vector Address -- With reference to Table 1-2, if -VEC (AI<5>) is not asserted (high) during the assertion of PI the DCT11-AA gets the vector address from an internal fixed table by decoding the inputs -HALT, -PF, and -CP<3:0>.

Table 1-2 Interrupt Decode

-CP<3> (AI<1>)	-CP<2> (AI<2>)	-CP<1> (AI<3>)	-CP<0> (AI<4>)	Priority Level	Vector Address	
X	X	X	X	8		-HALT *
X	X	X	X	8	24	-PF
L	L	L	L	7	140	
L	L	L	H	7	144	
L	L	H	L	7	150	
L	L	H	H	7	154	
L	H	L	L	6	100	
L	H	L	H	6	104	
L	H	H	L	6	110	
L	H	H	H	6	114	
H	L	L	L	5	120	
H	L	L	H	5	124	
H	L	H	L	5	130	
H	L	H	H	5	134	
H	H	L	L	4	60	
H	H	L	H	4	64	
H	H	H	L	4	70	
H	H	H	H	NO ACTION		

* PC is loaded with the restart address. PSW = 340.

1.5.4.2 External Vector Address -- During the assertion of PI (-PF or -HALT not asserted), if -VEC (AI<5>) is asserted (low) DCT11-AA obtains the vector from the external device during an Iack transaction. Asserting READY causes the DCT11-AA to wait for the vector.

1.5.5 Priority

With reference to Table 1-2, each interrupt is assigned a priority level. DCT11-AA divides interrupts into two groups:

- Maskable
- Non-maskable.

1.5.5.1 Maskable Interrupts -- Interrupts on -CP<3:0> are maskable. With reference to Table 1-2, the interrupts are serviced according to their priority level.

NOTE

As in any multilevel priority structure, the PSW of the service routine must contain a priority level as high or higher than that of the interrupt request. Otherwise, the interrupt request continues to cause Iack transactions (Refer to paragraph 2.12.) until the stack is full.

1.5.5.2 Non-maskable Interrupts -- -HALT is the highest priority and interrupts the processor whatever the processor's status.

NOTE

The -HALT interrupt or execution of the -HALT instruction results in an interrupt, not in stopping the processor.

1.6 DIRECT MEMORY ACCESS (DMA) MECHANISM

During a DMA transaction the only lines that are three-stated are DAL<15:0>. Low current pull-ups are placed on:

- AI<7:0>
- R/-WHB
- R/-WLB

The processor maintains control of -RAS, -CAS, and PI.

With reference to Figure 1-7, a device requests control of the DMA bus (DAL<15:0>, AI<7:0>, R/-WHB, and R/-WLB) by asserting Direct Memory Request (DMR (AI<0>)) during the assertion of PI. DMR is read during any assertion of PI, unlike interrupts which are read

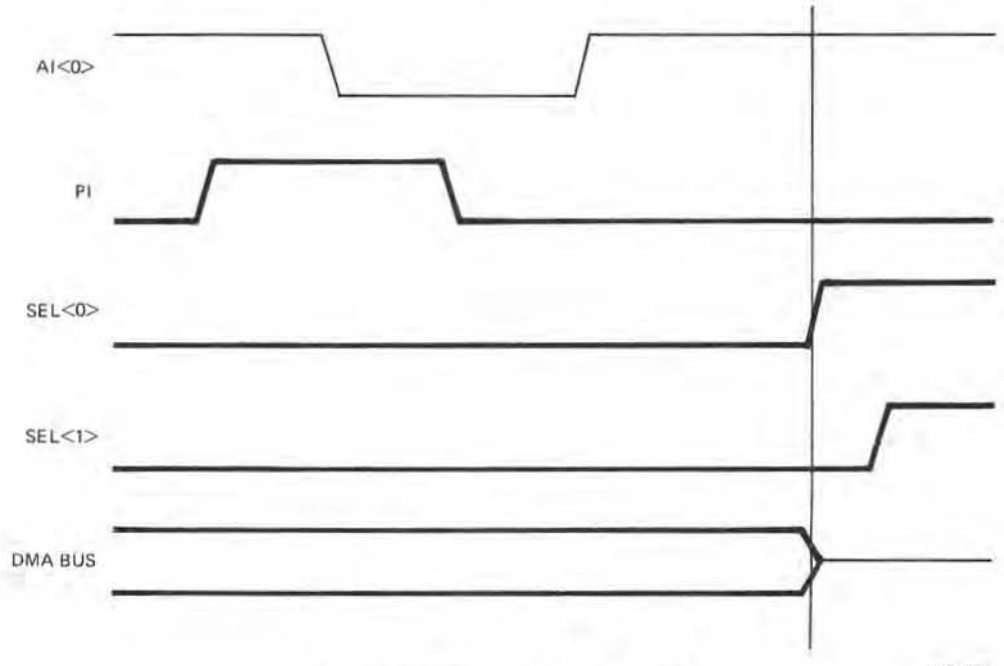


Figure 1-7 DMA Timing

only during a read transaction or an ASPI transaction. The processor waits for the end of the current transaction (read, write, DMG, or ASPI) and then releases the DMA bus. The requesting device is signaled, by the processor, by asserting the two signals:

- SEL<0> (high)
- SEL<1> (high).

SEL<0> and SEL<1> indicate a Direct Memory Grant (DMG).

The requesting device, having received DMG, performs the DMA by controlling the DMA bus. The processor continues to output PI in order to allow the negation of DMR. The device holds control of the DMA bus until DMR is negated during PI. Multiple DMA devices can be implemented using a daisy chain structure as shown in Figure 1-8.

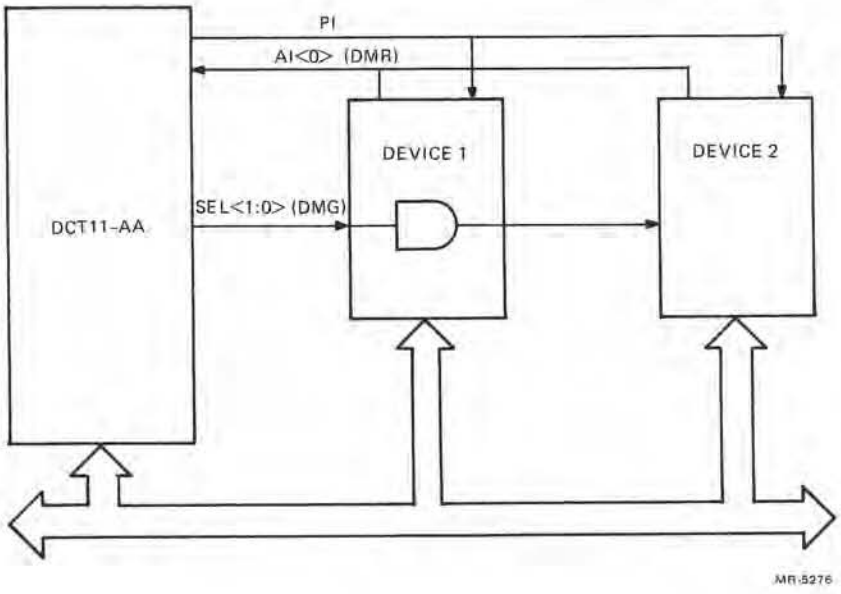


Figure 1-8 DMA Block Diagram

CHAPTER 2
BUS TRANSACTIONS

2.1 INTRODUCTION

A basic discussion of each bus transaction is contained within this chapter. Paragraphs 2.3 through 2.10 pertain to the read and write transactions. The details of the read and write transactions change considerably in each of the following modes:

- 8-bit static
- 8-bit dynamic
- 16-bit static
- 16-bit dynamic.

Therefore, a seperate discussion for each read and write transaction is presented. All other transactions are described as they apply to the DCT11-AA bus.

2.2 THE BUS TRANSACTION

With reference to Figure 2-1, each PDP-11 instruction is constructed of a number of transactions.

2.2.1 Transaction

A transaction is defined as the activity taking place on the DCT11-AA bus in order to perform a function such as:

- Read
- Write
- Refresh
- Iack (Interrupt Acknowledge)
- DMA (Direct Memory Access)
- ASPI (Assert Priority In)
- NOP (No Operation).

2.2.2 Microcycle

Each transaction is made up of either one or two microcycles. A microcycle is defined as the activity required for one micro-instruction to be executed. The microcycle performs the functions necessary for transferring information to and from the DCT11-AA bus, internally moving data, and calculating values.

2.2.3 Clock Phase

The basic building block of the DCT11-AA timing is the clock phase. Each microcycle is normally constructed of three clock phases 01, 02, and 0W. During an ASPI transaction, Iack transaction, DMA transaction, or when operating in long microcycle mode it is necessary to add a fourth phase, phase D (0D), between 02 and 0W. All clock phases have the same duration between assertions.

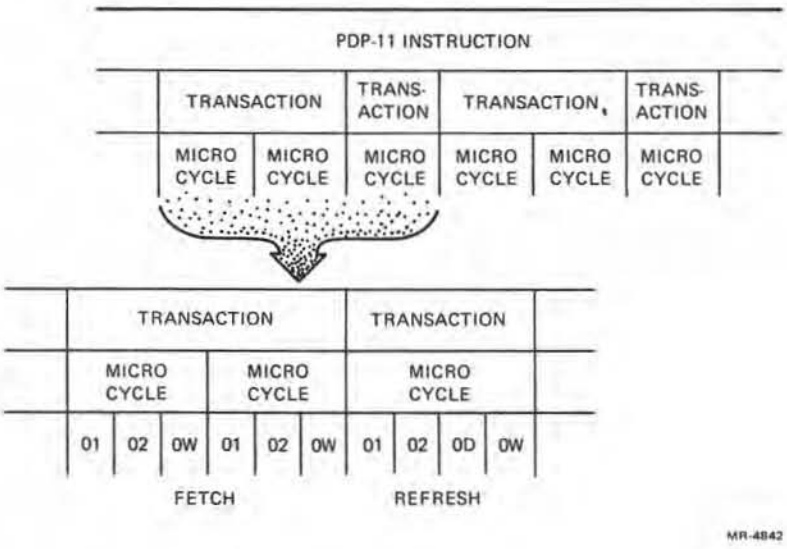


Figure 2-1 Transaction Breakdown

2.3 16-BIT STATIC READ TRANSACTION

A Read Transaction consists of three distinct processes:

- Output of address
- Input of data
- Input of interrupt and DMA request (refer to paragraphs 1.5 and 2.14).

Detailed timing of a 16-bit static read transaction is found in Figure A-2 of Appendix A.

NOTE
All references to input or output refer to the processor.

2.3.1 Output of Address

With reference to Figure 2-3, the address is output on the Data Address Lines (DAL) 15 through 0 (<15:0>). The condition of DAL<0> indicates the address of a word, high byte, or low byte. Data address lines are time multiplexed and used for both address and data.

2.3.1.1 Address Control -- Refer to Figures 2-2 and 2-3. Address Strobe, which is used to latch the address into the memory system or register, is accomplished by means of Row Address Strobe (-RAS). The address is latched upon the assertion (leading edge) of -RAS.

2.3.2 Input of Data

With reference to Figure 2-3, the input data should be valid on DAL<15:0> during the period of time that Priority In (PI) is asserted.

2.3.2.1 Data Control -- The data strobe, which the processor uses to latch the input data, is accomplished by means of Column Address Strobe (-CAS). The data is latched upon the negation (trailing edge) of -CAS. Read/write control is accomplished through the use of two signals:

- Read/-Write High Byte (R/-WHB)
- Read/-Write Low Byte (R/-WLB).

Both of these signals remain high during a read transaction.

2.3.3 Instruction Fetch

An instruction fetch is indicated by two signals:

- SEL<0> High
- SEL<1> Low.

Refer to Figure A-2 in Appendix A.

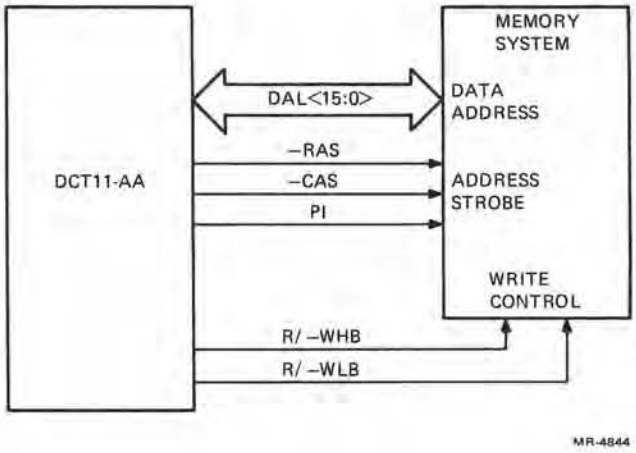


Figure 2-2 16-Bit Static Read Block Diagram

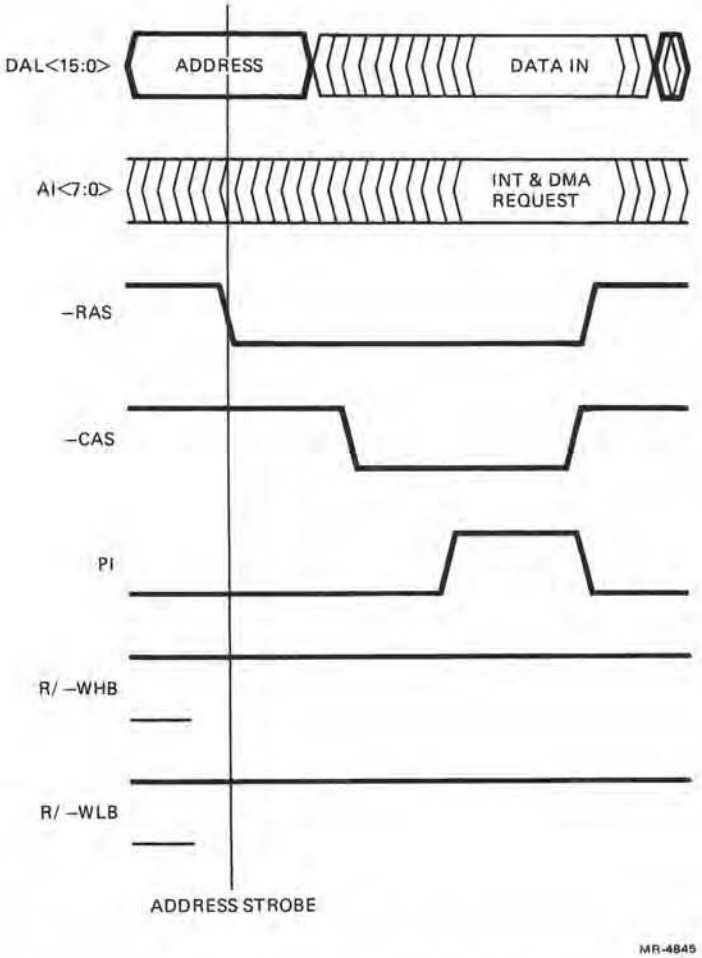


Figure 2-3 16-Bit Static Read Timing

2.4 16-BIT STATIC WRITE TRANSACTION

A write transaction is comprised of three distinct processes:

- Output of address
- Output of data
- Input of DMA request (refer to paragraph 2.14).

Detailed timing of a 16-bit static write transaction is found in Figure A-3 of Appendix A.

NOTE

All references to input or output refer to the processor.

NOTE

Other than writing the stack during an interrupt or trap, a write transaction is always preceded by a read transaction and the two are indivisible.

2.4.1 Output of Address

With reference to Figures 2-4 and 2-5, the address is output on DAL<15:0>. The condition of DAL<0> indicates the addressing of a word, high byte, or low byte. Refer to Table 2-1. DAL<15:0> are time multiplexed and used for both address and data.

2.4.1.1 Address Control -- Address strobe, which is used to latch the address into the memory system or register, is accomplished by means of -RAS. The address is latched upon the assertion (leading edge) of -RAS.

2.4.2 Output of Data

With reference to Figure 2-5, the data is output on DAL<15:0> before the assertion (leading edge) of PI.

2.4.2.1 Data Control -- The signal used to latch the data into the memory system or register and the edge required is found in Table 2-2.

Write control is accomplished through the use of two signals:

- R/-WHB
- R/-WLB.

Table 2-1 indicates the conditions necessary to address and write a memory.

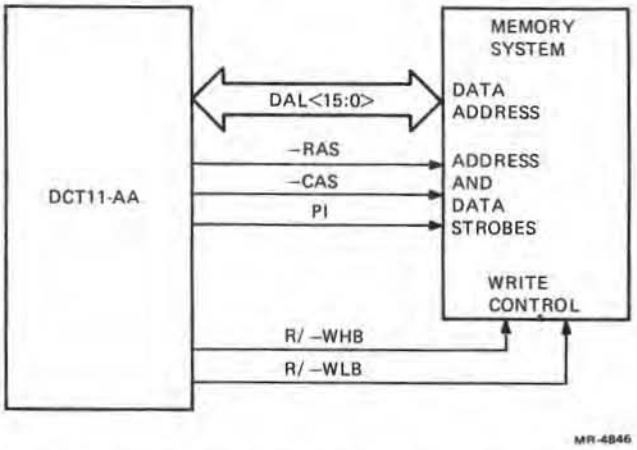


Figure 2-4 16-Bit Static Write Block Diagram

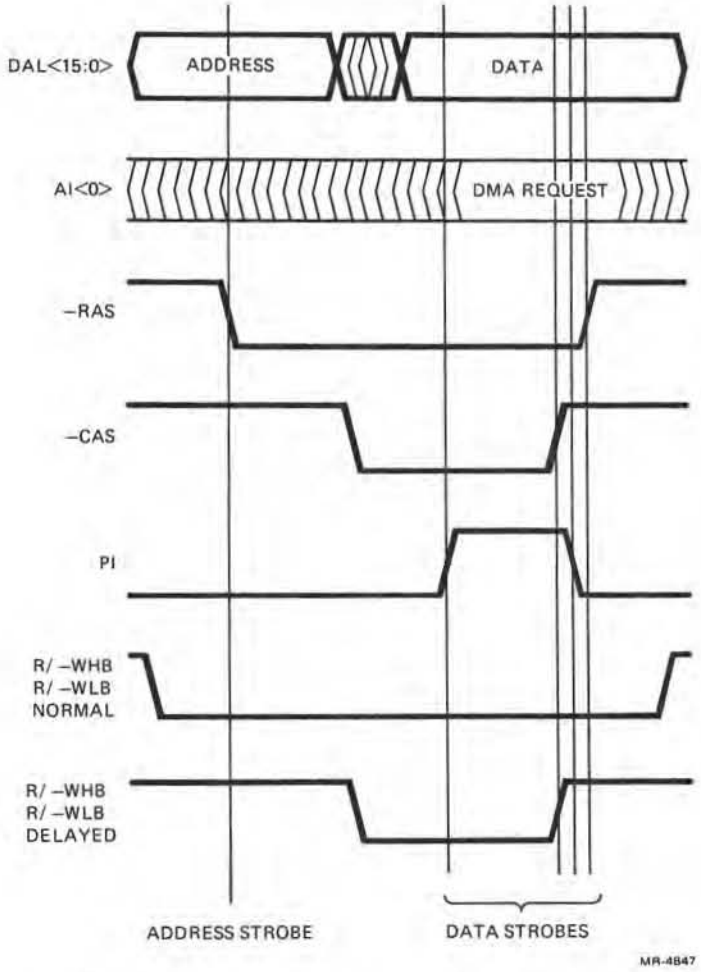


Figure 2-5 16-Bit Static Write Timing

Table 2-1 Write Conditions

Addressed Memory		Address	R/-WHB R/-WLB
WORD	EVEN (DAL<0>=0)	0	0
LOW BYTE	EVEN (DAL<0>=0)	1	0
HIGH BYTE	ODD (DAL<0>=1)	0	1

Table 2-2 Data Strobe

Signal	Edge
-RAS	Negation (trailing)
-CAS	Negation (trailing)
PI	Assertion (leading)
PI	Negation (trailing)

2.5 16-BIT DYNAMIC READ TRANSACTION

A read transaction consists of three distinct processes:

- Output of address
- Input of data
- Input of interrupt and DMA request (refer to paragraphs 1.5 and 2.14).

Detailed timing of a 16-bit dynamic read transaction is found in Figure A-4 of Appendix A.

NOTE
All references to input or output refer to the processor.

2.5.1 Output of Address

Both static and dynamic addresses are output concurrently while in dynamic mode.

2.5.1.1 Dynamic Address -- With reference to Figures 2-6 and 2-7, the address is output on the Address Interrupt (AI) lines 7 through 0 (<7:0>). The AI lines output the row address first and second the column address. Table 2-3 indicates the address bits required in 4/16K mode and 64K mode.

NOTE
The AI lines are not in order. Refer to Table 2-4.

2.5.1.2 Static Address -- The addressing of a static ROM, RAM, or register in a system supporting dynamic devices is accomplished by outputs concurrent with the AI<7:0>. The concurrent address is output on DAL<15:0>.

2.5.1.3 Address Control -- Table 2-5 indicates the signals and edge required to latch each portion of the address into the memory system or register.

2.5.2 Input of Data

With reference to Figure 2-7, the input data should be valid on DAL<15:0> during the period of time that PI is asserted.

Table 2-3 Dynamic Addressing Scheme

Mode	Memory Chip	Address	AI Used
4/16K	4K X 1	A1--A12	<6:1>
4/16K	16K X 1	A1--A14	<7:1>
64K	64K X 1	A1--A15	<7:0>

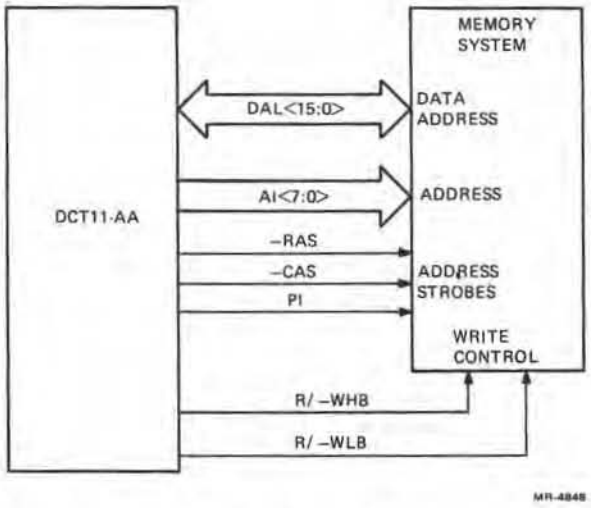


Figure 2-6 16-Bit Dynamic Read Block Diagram

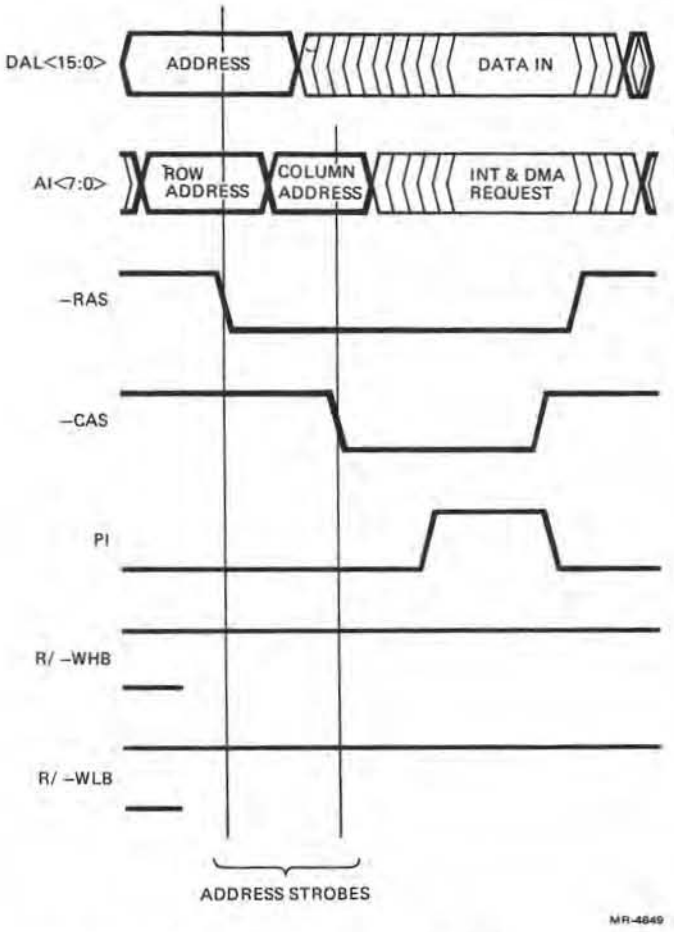


Figure 2-7 16-Bit Dynamic Read Timing

2.5.2.1 Data Control -- The data strobe, which the processor uses to latch the input data, is accomplished by means of -CAS. The data is latched upon the negation (trailing edge) of -CAS. Write control is accomplished through the use of two signals:

- R/-WHB
- R/-WLB.

Both of these signals remain high during a read transaction.

2.5.3 Instruction Fetch

An instruction fetch is indicated by different signals depending on the mode. Refer to Tables A-4, A-7, and Figure A-4 in Appendix A.

2.5.3.1 4K/16K Mode -- In 4K/16K 16-bit dynamic mode AI<0> is asserted at the leading edge of -RAS to indicate a fetch operation. AI<0> is 3-stated before the leading edge PI.

Fetch is indicated by: AI<0> high

NOTE
During refresh, the AI lines have the refresh counter address on them.

2.5.3.2 64K Mode -- 64K and static modes use SEL<0> high and SEL<1> low to indicate a fetch condition. When SEL<0> signifies a fetch, it is asserted only during the read cycle.

Fetch is indicated by: SEL<0> high
SEL<1> low

Table 2-4 AI Addressing

AI	ADDRESS			
	4K/16K		64K	
	-RAS	-CAS	-RAS	-CAS
<0>	FET	A14	A15	A14
<1>	A1	A2	A1	A2
<2>	A3	A4	A3	A4
<3>	A5	A6	A5	A6
<4>	A7	A8	A7	A8
<5>	A9	A10	A9	A10
<6>	A11	A12	A11	A12
<7>	A13	A14	A13	A14

Table 2-5 Address Strobes

Address	Signal	Edge	Device	R/-WHB	R/-WLB
ROW	-RAS	Assertion (leading)	Dynamic	1	1
COLUMN	-CAS	Assertion (leading)	Dynamic	1	1
DAL	-RAS	Assertion (leading)	Dynamic or Static	1	1

2.6 16-BIT DYNAMIC WRITE TRANSACTION

A write transaction consists of three distinct processes:

- Output of address
- Output of data
- Input of DMA request (refer to paragraph 2.14).

Detailed timing of a 16-bit dynamic write transaction is found in Figure A-5 of Appendix A.

NOTE
All references to input or output refer to the processor.

NOTE
Other than writing the stack during an interrupt or trap, a write transaction is always preceded by a read transaction and the two are indivisible.

2.6.1 Output of Address

Both static and dynamic addresses are output concurrently while in dynamic mode.

2.6.1.1 Dynamic Address -- With reference to Figures 2-8 and 2-9, the address is output on AI<7:0>. The AI lines output the row address first and second the column address. Table 2-6 indicates the address bits required by memories in 4/16K mode and 64K mode.

NOTE
The AI lines are not in order. Refer to Table 2-7.

2.6.1.2 Static Address -- The addressing of a static ROM, RAM, or register in a system supporting dynamic devices is accomplished by outputs concurrent with the AI<7:0>. The concurrent address is output on DAL<15:0>.

2.6.1.3 Address Control -- Table 2-8 indicates the signal and edge required to latch each portion of the address into the memory system or register.

2.6.2 Output of Data

With reference to Figure 2-9, the data is output on DAL<15:0>.

2.6.2.1 Data Control -- The signals used to latch the data into the memory system or register and the edge required is found in Table 2-9. Write control is accomplished through the use of two signals:

- R/-WHB
- R/-WLB.

Table 2-10 indicates the conditions necessary to address and write a memory system or register. The assertion of R/-WHB and R/-WLB is found in Table 2-11.

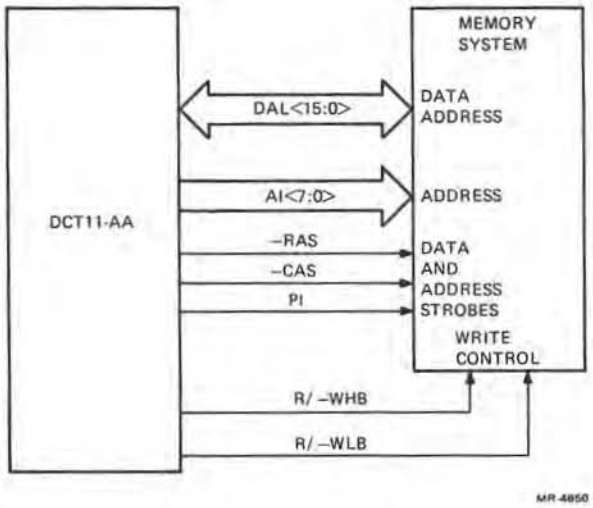


Figure 2-8 16-Bit Dynamic Write Block Diagram

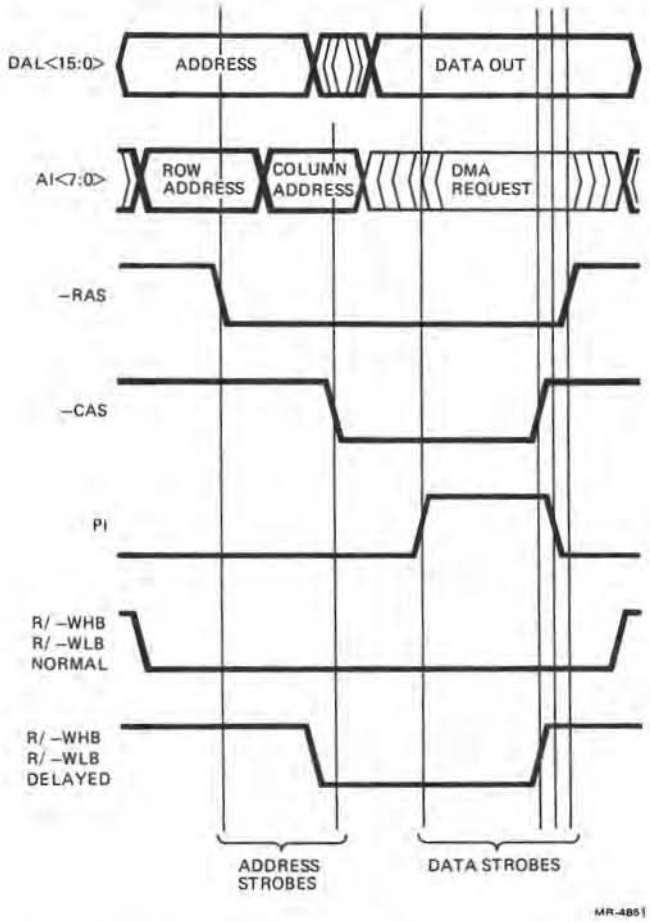


Figure 2-9 16-Bit Dynamic Write Timing

Table 2-6 Dynamic Addressing Scheme

Mode	Memory Chip	Address	AI Used
4/16K	4K X 1	A1--A12	<6:1>
4/16K	16K X 1	A1--A14	<7:1>
64K	64K X 1	A1--A15	<7:0>

Table 2-7 AI Addressing

AI	ADDRESS			
	4K/16K		64K	
	-RAS	-CAS	-RAS	-CAS
<0>	FET	A14	A15	A14
<1>	A1	A2	A1	A2
<2>	A3	A4	A3	A4
<3>	A5	A6	A5	A6
<4>	A7	A8	A7	A8
<5>	A9	A10	A9	A10
<6>	A11	A12	A11	A12
<7>	A13	A14	A13	A14

Table 2-8 Address Strobes

Address	Signal	Edge	Device
ROW	-RAS	Assertion (leading)	Dynamic
COLUMN	-CAS	Assertion (leading)	Dynamic
DAL	-RAS	Assertion (leading)	Dynamic or Static

Table 2-9 Data Strobes

Signal	Edge
-RAS	Negation (trailing)
-CAS	Negation (trailing)
PI	Assertion (leading)
PI	Negation (trailing)

Table 2-10 Write Conditions

Addressed Memory	Address	R/-WHB	R/-WLB
WORD	EVEN (DAL<0>=0)	0	0
LOW BYTE	EVEN (DAL<0>=0)	1	0
HIGH BYTE	ODD (DAL<0>=1)	0	1

Table 2-11 Write Control Timing

Signal	Mode	Parameter
R/-WHB	NORMAL	Write control before -CAS assertion
R/-WLB	NORMAL	Write control before -CAS assertion
R/-WHB	DELAYED	Write control at or after -CAS assertion
R/-WLB	DELAYED	Write control at or after -CAS assertion

2.7 8-BIT STATIC READ TRANSACTION

A read transaction consists of three distinct processes:

- Output of address
- Input of data
- Input of interrupt and DMA request (refer to paragraphs 1.5 and 2.14).

Detailed timing of a 8-bit static read transaction is found in Figure A-6 of Appendix A.

When a word-read or a word-write is being executed, the transaction is repeated twice and the two transactions are indivisible. For example, the MOV (move word) instruction first does a read transaction and addresses the low byte data. The address is then incremented by one and the second read transaction addresses the high byte data. In the case of the MOVB (move byte) instruction, the transaction occurs only once.

NOTE

All references to input or output refer to the processor.

2.7.1 Output of Address

With reference to Figures 2-10 and 2-11, the high byte address is output on the Static Address Lines (SAL) 15 through 8 (<15:8>). The low byte of the address is output on DAL<7:0>. Data address lines are time multiplexed and used for both address and data.

2.7.1.1 Address Control -- Address strobe which is used to latch the address into the memory system or register, is accomplished by means of -RAS. The address is latched upon the assertion (leading edge) of -RAS.

2.7.2 Input of Data

With reference to Figure 2-11, the input data should be valid on DAL<7:0> during the period of time that PI is asserted.

2.7.2.1 Data Control -- The data strobe, which the processor uses to latch the input data, is accomplished by means of -CAS. The data is latched upon the negation (trailing edge) of -CAS. Read control is accomplished through the use of the signal -Read (R/-WHB). The assertion of -Read is found in Table 2-12.

2.7.3 Instruction Fetch

An instruction fetch is indicated by two signals:

- SEL<0> High
- SEL<1> Low.

Refer to Figure A-6 in Appendix A.

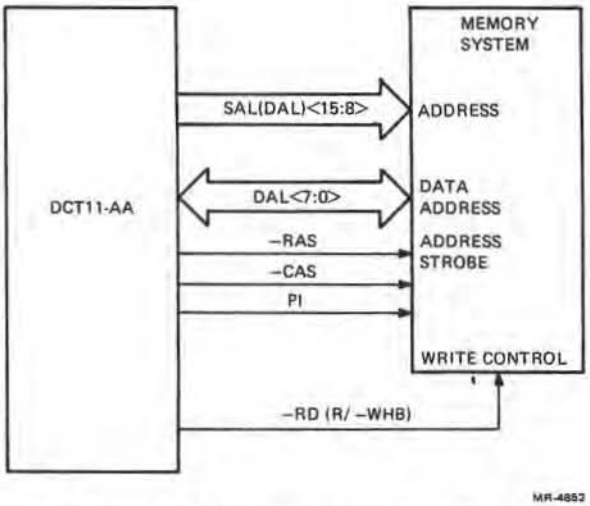


Figure 2-10 8-Bit Static Read Block Diagram

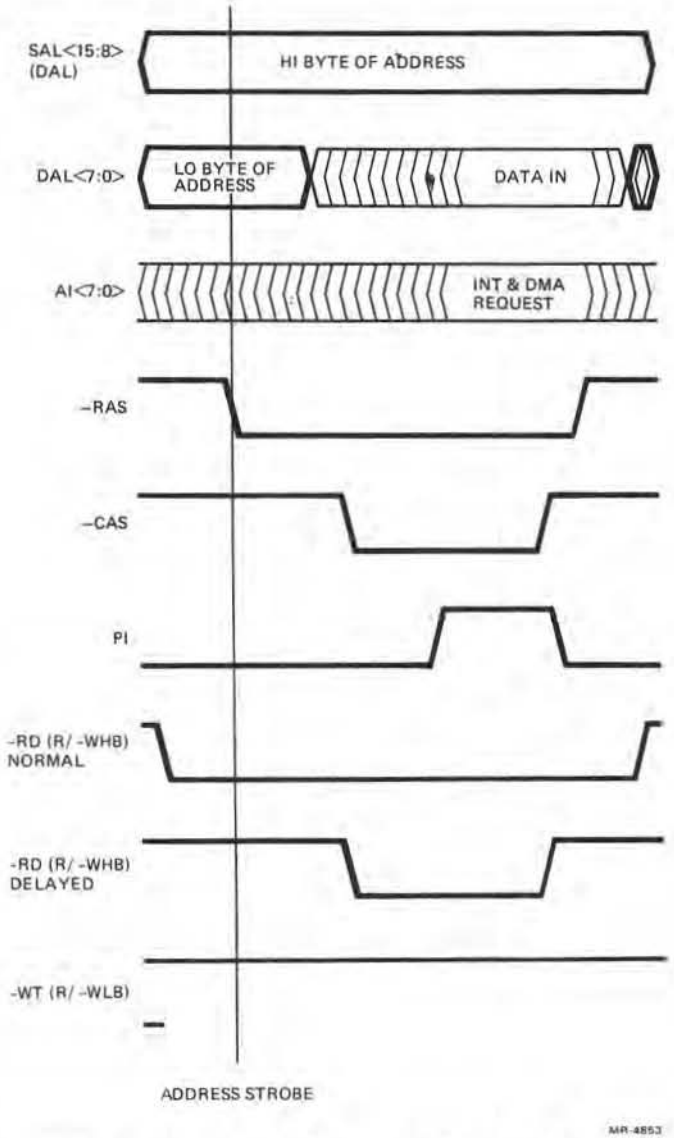


Figure 2-11 8-Bit Static Read Timing

Table 2-12 Read Control Timing

Signal	Mode	Parameter
-RD (R/-WHB)	NORMAL	Read control before -CAS assertion
-RD (R/-WHB)	DELAYED	Read control at or after -CAS assertion

Table 2-13 Data Strobes

Signal	Edge
-RAS	Negation (trailing)
-CAS	Negation (trailing)
PI	Assertion (leading)
PI	Negation (trailing)

2.8 8-BIT STATIC WRITE TRANSACTION

A write transaction consists of three distinct processes:

- Output of address
- Output of data
- Input of DMA request (refer to paragraph 2.14).

Detailed timing of a 8-bit static write transaction is found in Figure A-7 of Appendix A.

When a word-read or a word-write is being executed, the transaction is repeated twice and the two transactions are indivisible. For example, the MOV (move word) instruction first does a read transaction and addresses the low byte data. The address is then incremented by one and the second read transaction addresses the high byte data. In the case of the MOVB (move byte) instruction, the transaction occurs only once.

NOTE
All references to input or output refer to the processor.

NOTE
Other than writing the stack during an interrupt or trap, a write transaction is always preceded by a read transaction and the two are indivisible.

2.8.1 Output of Address

With reference to Figures 2-12 and 2-13, the high byte address is output on the Static Address Line (SAL) 15 through 8 (<15:8>). The low byte of the address is output on DAL<7:0>. Data address lines are time multiplexed and used for both address and data.

2.8.1.1 Address Control -- Address strobe, which is used to latch the address into the memory system or register, is accomplished by means of -RAS. The address is latched upon the assertion (leading edge) of -RAS.

2.8.2 Output of Data

With reference to Figure 2-13, the data is output on DAL<7:0> before the assertion (leading edge) of PI.

2.8.2.1 Data Control -- The signals used to latch the data into the memory system or register and the edge required is found in Table 2-13.

Write Control is accomplished through the use of the signal -Write (R/-WLB). The assertion of -Write is found in Table 2-14.

Table 2-14 Write Control Timing

Signal	Mode	Parameter
-WT (R/-WLB)	NORMAL	Write control before -CAS assertion
-WT (R/-WLB)	DELAYED	Write control at or after -CAS assertion

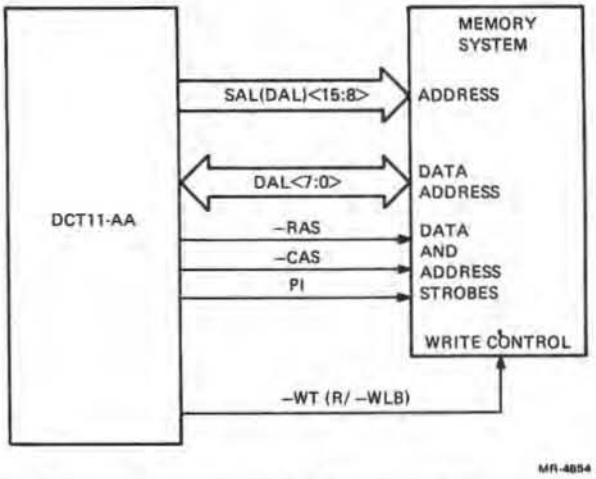


Figure 2-12 8-Bit Static Write Block Diagram

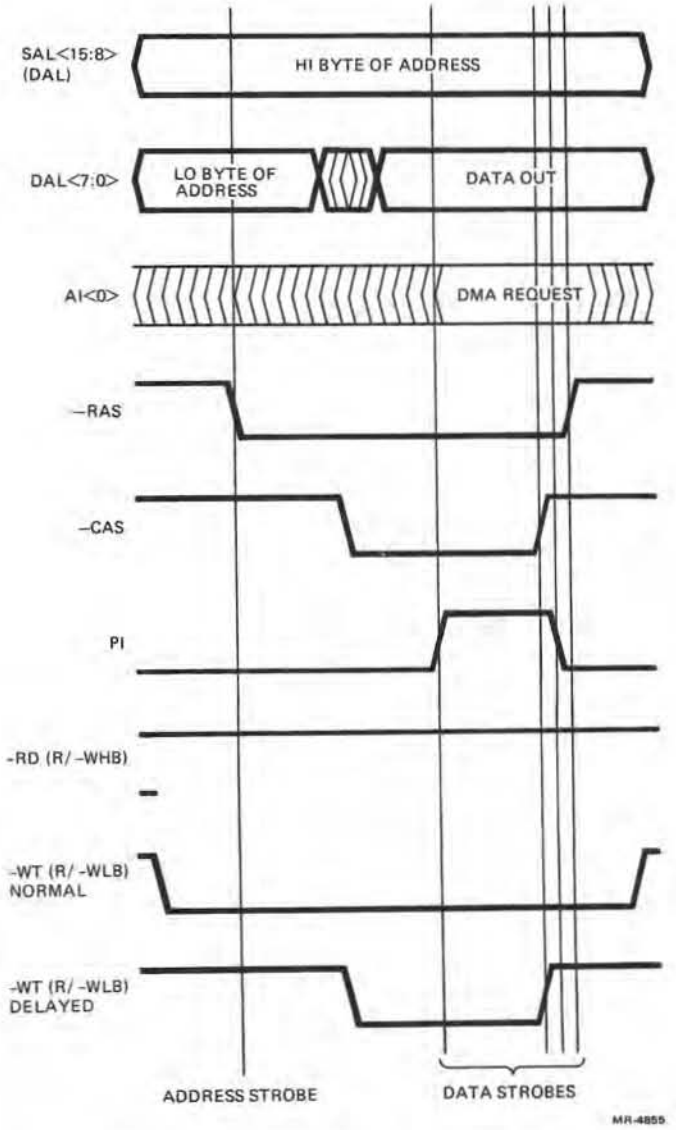


Figure 2-13 8-Bit Static Write Timing

2.9 8-BIT DYNAMIC READ TRANSACTION

A read transaction consists of three distinct processes:

- Output of address
- Input of data
- Input of interrupt and DMA request (refer to paragraphs 1.5 and 2.14).

Detailed timing of a 8-bit dynamic read transaction is found in Figure A-8 of Appendix A.

When a word-read or a word-write is being executed, the transaction is repeated twice and the two transactions are indivisible. For example, the MOV (move word) instruction first does a read transaction and addresses the low byte data. The address is then incremented by one and the second read transaction addresses the high byte data. In the case of the MOVB (move byte) instruction, the transaction occurs only once.

NOTE
All references to input or output refer to the processor.

2.9.1 Output of Address

Both static and dynamic addresses are output concurrently while in dynamic mode.

2.9.1.1 Dynamic Address -- With reference to Figures 2-14 and 2-15, the address is output on AI<7:0>. The AI lines output the row address first and second the column address. Table 2-15 indicates the address bits required in 4/16K mode and 64K mode.

NOTE
The AI lines are not in order. Refer to Table 2-16.

2.9.1.2 Static Address -- Addressing of a static ROM, RAM, or register in a system supporting dynamic devices is accomplished by outputs concurrent with the AI<7:0>. The high byte of the address is output on the Static Address Lines (SAL) 15 through 8 (<15:8>). The low byte of the address is output on DAL<7:0>.

2.9.1.3 Address Control -- Table 2-17 indicates the signal and edge required to latch each portion of the address into the memory system or register.

2.9.2 Input of Data

With reference to Figure 2-15, the input data should be valid on DAL<7:0> during the period of time that PI is asserted.

2.9.2.1 Data Control -- The data strobe, which the processor uses to latch the input data, is accomplished by means of -CAS. The data is latched upon the negation of (trailing edge) of -CAS. Read control is accomplished through the use of one signal -Read (R/-WHB). The timing of -Read is found in Table 2-18.

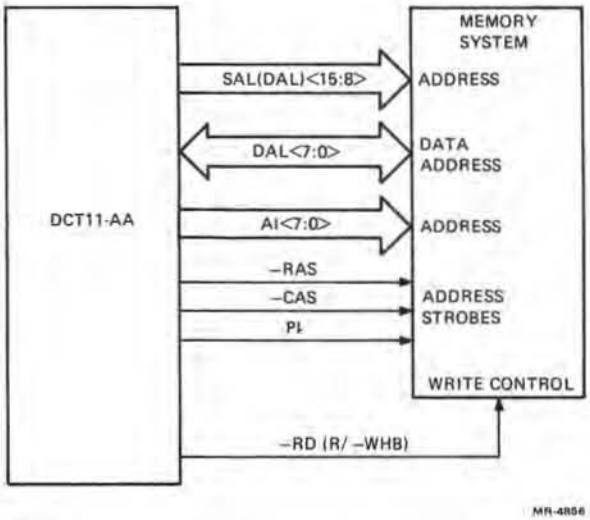


Figure 2-14 8-Bit Dynamic Read Block Diagram

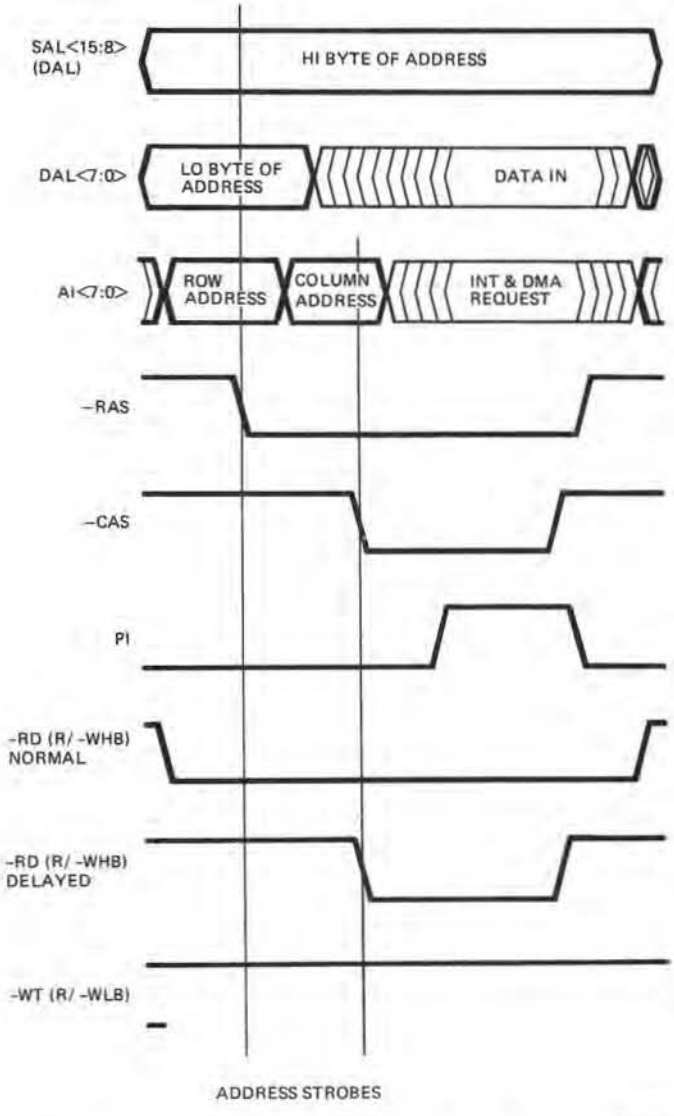


Figure 2-15 8-Bit Dynamic Read Timing

2.9.3 Instruction Fetch

An instruction fetch is indicated by different signals depending on the mode. Refer to Figure A-8 in Appendix A.

2.9.3.1 4K/16K Mode -- In 4K/16K 8-bit dynamic mode AI<0> is asserted at the leading edge of -RAS to indicate a fetch operation. AI<0> is 3-stated before the leading edge PI.

Fetch is indicated by: AI<0> high

NOTE
During refresh, the AI lines have the refresh counter address on them.

2.9.3.2 64K Mode -- 64K and static modes use SEL<0> high and SEL<1> low to indicate a fetch condition. When SEL<0> signifies a fetch, it is asserted only during the low byte read cycle.

Fetch is indicated by: SEL<0> high
SEL<1> low

Table 2-15 Dynamic Addressing Scheme

Mode	Memory Chip	Address	AI Used
4/16K	4K X 1	A0--A11	<6:1>
4/16K	16K X 1	A0--A13	<7:1>
64K	64K X 1	A0--A15	<7:0>

Table 2-16 AI Addressing

AI	ADDRESS			
	4K/16K		64K	
	-RAS	-CAS	-RAS	-CAS
<0>	FET	A14	A15	A14
<1>	A1	A2	A1	A2
<2>	A3	A4	A3	A4
<3>	A5	A6	A5	A6
<4>	A7	A8	A7	A8
<5>	A9	A10	A9	A10
<6>	A11	A0	A11	A0
<7>	A13	A12	A13	A12

Table 2-17 Address Strobes

Address	Signal	Edge	Device
ROW	-RAS	Assertion (leading)	Dynamic
COLUMN	-CAS	Assertion (leading)	Dynamic
SAL	-RAS	Assertion (leading)	Dynamic or Static
DAL	-RAS	Assertion (leading)	Dynamic or Static

Table 2-18 Read Control Timing

Signal	Mode	Parameter
-RD (R/-WHB)	NORMAL	Read control before -CAS assertion
-RD (R/-WHB)	DELAYED	Read control at or after -CAS assertion

2.10 8-BIT DYNAMIC WRITE TRANSACTION

A write transaction consists of three distinct processes:

- Output of addresses
- Output of data
- Input of DMA request (refer to paragraph 2.14).

Detailed timing of a 8-bit dynamic read transaction is found in Figure A-9 of Appendix A.

When a word-read or a word-write is being executed, the transaction is repeated twice and the two transactions are indivisible. For example, the MOV (move word) instruction first does a read transaction and addresses the low byte data. The address is then incremented by one and the second read transaction addresses the high byte data. In the case of the MOVB (move byte) instruction, the transaction occurs only once.

NOTE
All references to input or output refer to the processor.

NOTE
Other than writing the stack during an interrupt or trap, a write transaction is always preceded by a read transaction and the two are indivisible.

2.10.1 Output of Address

Both static and dynamic addresses are output concurrently while in dynamic mode.

2.10.1.1 Dynamic Address -- With reference to Figures 2-16 and 2-17, the address is output on AI<7:0>. The AI lines output the row address first and second the column address. Table 2-19 indicates the address bits required in 4/16K mode and 64K mode.

NOTE
The AI lines are not in order. Refer to Table 2-20.

2.10.1.2 Static Address -- Addressing of a static ROM, RAM, or register in a system which is supporting dynamic devices is accomplished by outputs concurrent with AI<7:0>. The high byte of the address is output on SAL<15:8>. The low byte of the address is output on DAL<7:0>.

2.10.1.3 Address Control -- Table 2-21 indicates the signal and edge required to latch each portion of the address into the memory system or register.

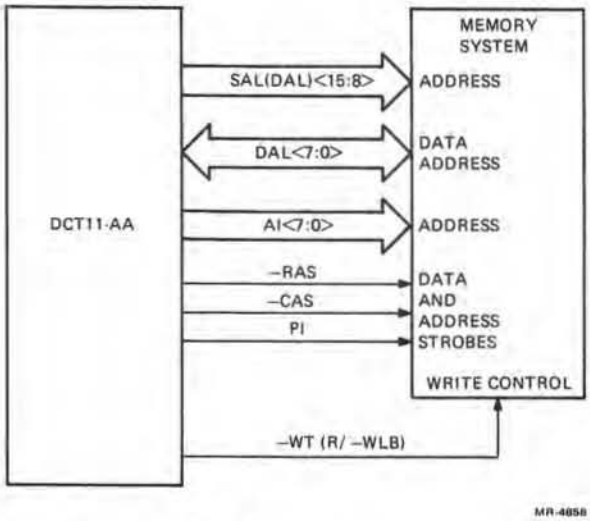


Figure 2-16 8-Bit Dynamic Write Block Diagram

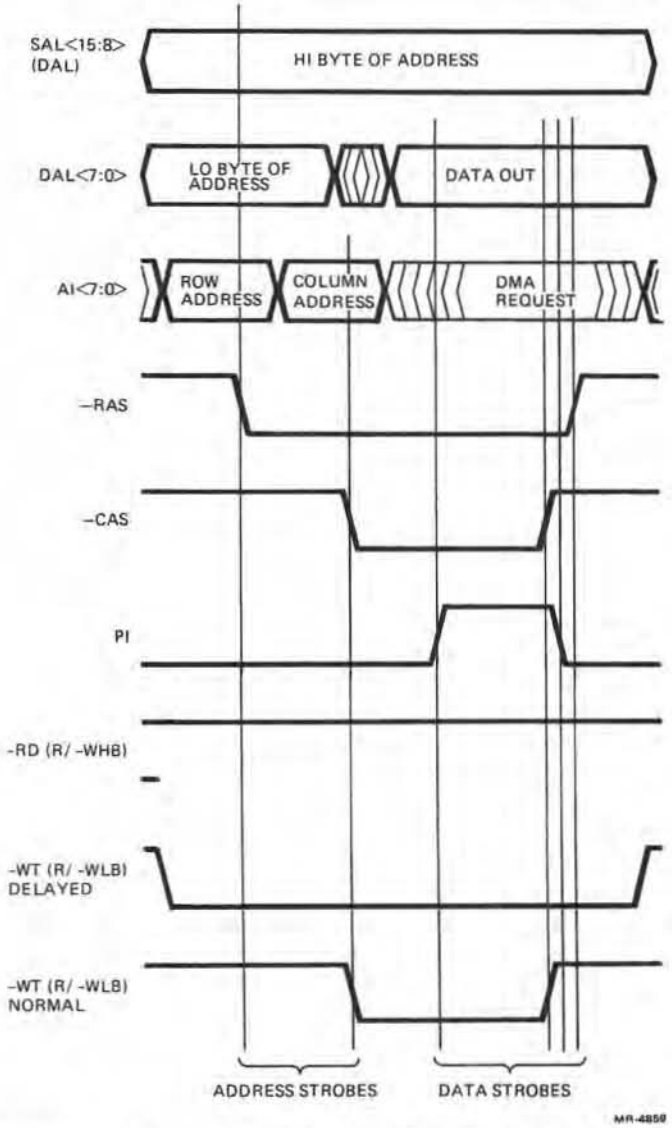


Figure 2-17 8-Bit Dynamic Write Timing

2.10.2 Output of Data

With reference to Figure 2-17, the data is output on DAL<7:0>.

2.10.2.1 Data Control -- The signals used to latch the data into a memory system or register and the edge required is found in Table 2-22.

Write control is accomplished through the use of one signal -Write (R/-WLB). The timing of -Write is found in Table 2-23.

Table 2-19 Dynamic Addressing Scheme

Mode	Memory Chip	Address	AI Used
4/16K	4K X 1	A0--A11	<6:1>
4/16K	16K X 1	A0--A13	<7:1>
64K	64K X 1	A0--A15	<7:0>

Table 2-20 AI Addressing

AI	ADDRESS			
	4K/16K		64K	
	-RAS	-CAS	-RAS	-CAS
<0>	FET	A14	A15	A14
<1>	A1	A2	A1	A2
<2>	A3	A4	A3	A4
<3>	A5	A6	A5	A6
<4>	A7	A8	A7	A8
<5>	A9	A10	A9	A10
<6>	A11	A0	A11	A0
<7>	A13	A12	A13	A12

Table 2-21 Address Strobes

Address	Signal	Edge	Device
ROW	-RAS	Assertion (leading)	Dynamic
COLUMN	-CAS	Assertion (leading)	Dynamic
SAL	-RAS	Assertion (leading)	Dynamic or Static
DAL	-RAS	Assertion (leading)	Dynamic or Static

Table 2-22 Data Strobes

Signal	Edge
-RAS	Negation (trailing)
-CAS	Negation (trailing)
PI	Assertion (leading)
PI	Negation (trailing)

Table 2-23 Write Control Timing

Signal	Mode	Parameter
-WT (R/-WLB)	NORMAL	Write control before -CAS assertion
-WT (R/-WLB)	DELAYED	Write control at or after -CAS assertion

2.11 REFRESH TRANSACTION

A refresh transaction consists of three distinct processes:

- Output of refresh address
- Address control
- Output of SEL<0> and SEL<1> (in 4K/16K mode only).

Detailed timing of a refresh transaction is found in Figure A-10 of Appendix A.

NOTE
All references to input or output refer to the processor.

2.11.1 Output of Refresh Address

With reference to Figures 2-18 and 2-19, the refresh address is output on AI<7:0>. Refresh occurs at different times;

- After an instruction fetch:
  8-bit mode - every instruction
  16-bit mode - after every other instruction.

- After addressing modes 5, 6, and 7:
  INDEX
  INDEX DEFERRED
  AUTO DECREMENT DEFERRED.

- During the following instructions:
  HALT
  TRAP
  BPT
  IOT.

- During all interrupts and traps.

2.11.2 Address Control

Address strobe, which is used to latch the address into the memory, is accomplished by means of -RAS. The address is latched upon the assertion (leading edge) of -RAS.

2.11.3 Output of SEL<0> and SEL<1>

With reference to Figure 2-19, if mode register bit 10 is not set (MR<10> = 1 4K/16K mode), during the refresh transaction:

- SEL<0> High
- SEL<1> Low.

If MR<10> is set (MR<10> = 0 64K mode), during the refresh transaction:

- SEL<0> Low
- SEL<1> Low.

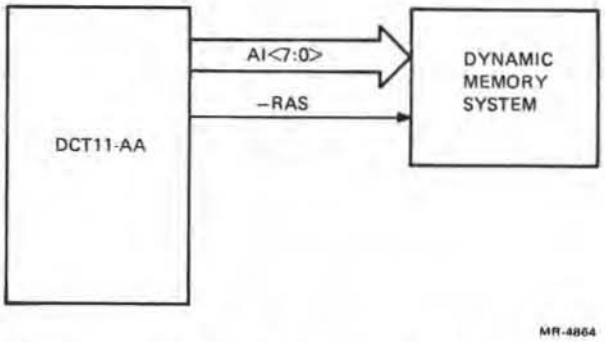


Figure 2-18 Refresh Transaction Block Diagram

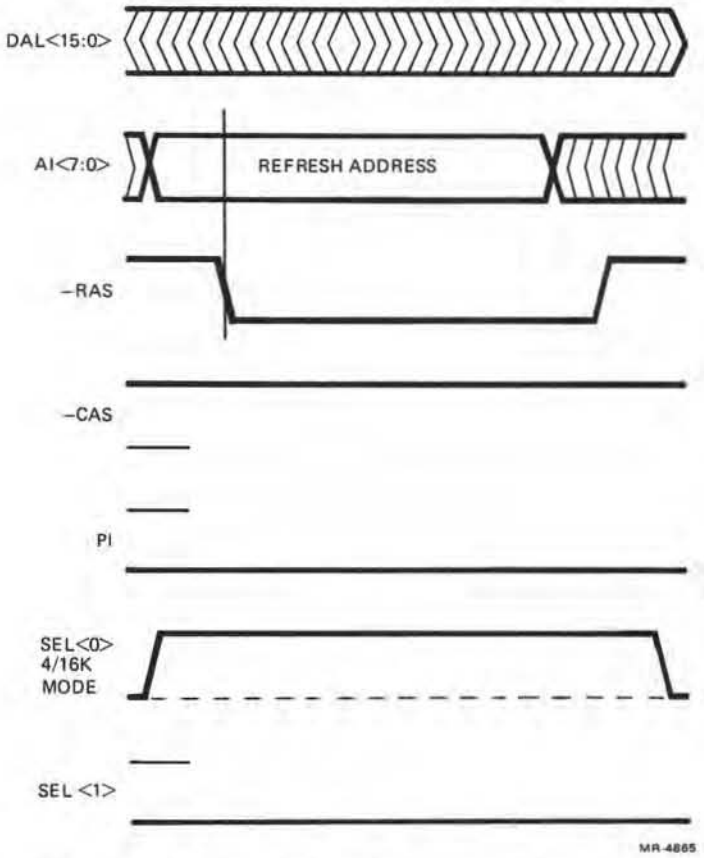


Figure 2-19 Refresh Transaction Timing

2.13 BUSNOP (No Operation) TRANSACTION

A busnop transaction is a specific processor state in which no processes occur at the outputs. The following is a list of the states found at the outputs:

- DAL<15:0> Previously latched data
- AI<7:0> Three-state (static mode) invalid output (dynamic mode)
- -RAS High
- -CAS High
- PI Low
- R/-WHB High
- R/-WLB High
- SEL<0> Low
- SEL<1> Low.

Detailed timing of a busnop transaction is found in Figure A-12 of Appendix A.

Examples of when a busnop transaction occurs are:

- Instruction decode cycle
- During internal processor computations.

2.14 DMA (Direct Memory Access) TRANSACTION

A DMA transaction consists of three processes:

- Three-state of DAL<15:0>, and internal pullups on AI<7:0>, R/-WHB, R/-WLB
- Output of -RAS, -CAS, and PI
- Output of DMG.

Detailed timing of a DMA transaction is found in Figure A-13 of Appendix A.

NOTE

All references to input or output refer to the processor.

Upon receiving a DMA request on AI<0> the processor at the end of the current transaction, initiates a DMA transaction. The DCT11-AA provides -RAS, -CAS, PI, and COUT signals. The external circuitry has the responsiblity for controlling the R/-WHB and R/-WLB lines, providing the address, and providing or accepting data.

During DMA transfers, system circuity goes through the following sequence:

- A DMA request (DMR) to the DCT11-AA is made by driving AI<0> low during PI
- The request is latched into the DCT11-AA during PI and shortly thereafter a DMA grant is issued.
- The processor relinquishes control of the bus to the device requesting the DMA.

If the bus is required for a longer period of time the requesting device must insure that AI<0> is low at the negation (trailing edge) of each PI.

2.14.1 Three-state of DAL<15:0>

With reference to Figure 2-22, the processor three-states DAL<15:0>. This is required to free the bus for the requesting device. AI<7:0>, R/-WHB, and R/-WLB have internal pull-ups.

2.14.2 Output of -RAS, -CAS, and PI

-RAS and -CAS are generated, during the DMA transaction, for use by the dynamic memory system as timing strobes. Refer to Figure 2-22. The output of PI is continued for the purpose of strobing the input of another DMA request on AI<0>. The DMA request is latched into the processor upon the negation (trailing edge) of PI.

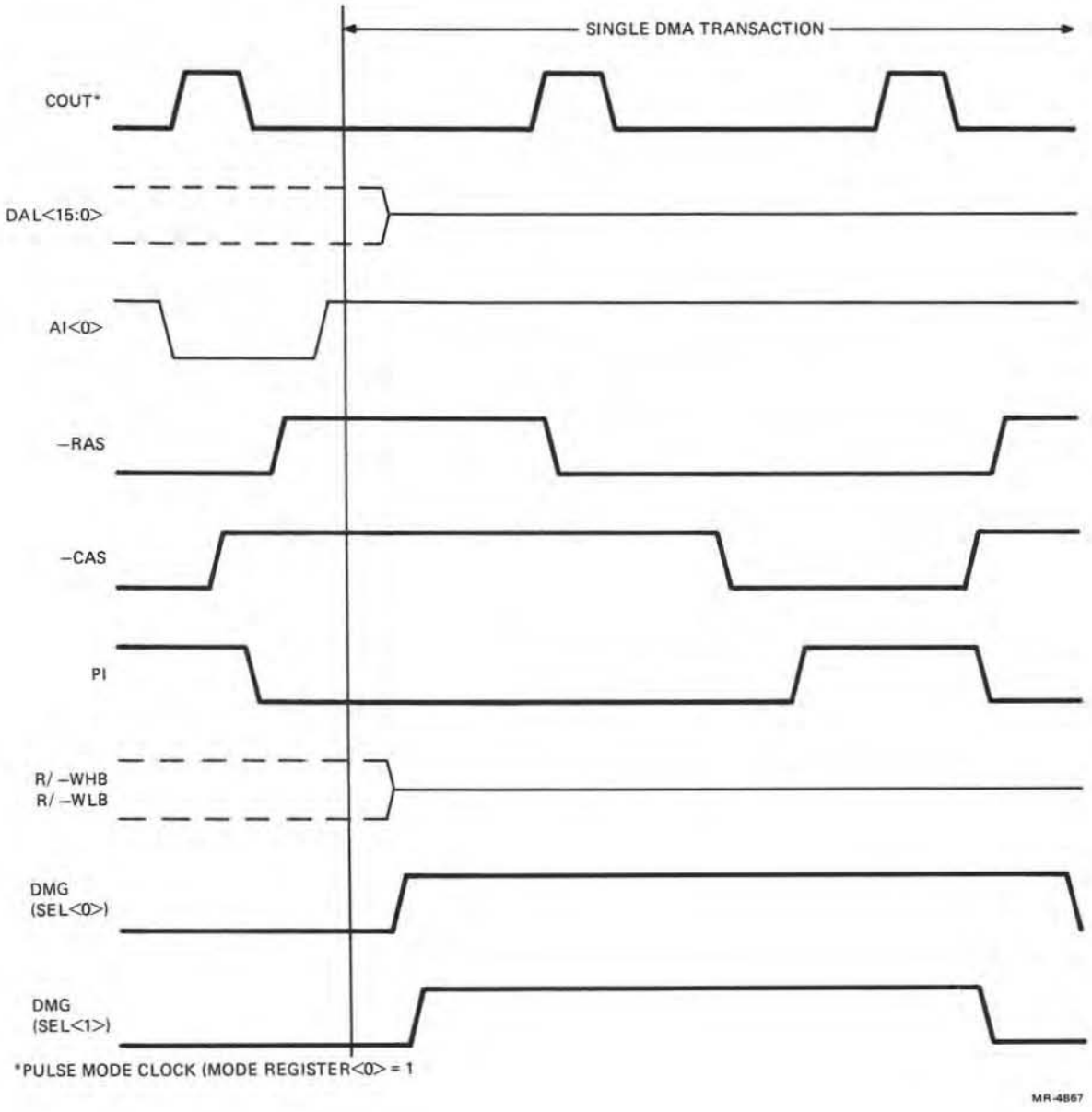


Figure 2-22 DMA Timing

2.14.3 Output of Direct Memory Grant (DMG)

With reference to Figure 2-22, when the grant is issued the DCT11-AA takes the following actions:

- SEL<0> and SEL<1> are asserted (high) informing the system that the grant has been issued and both signals are valid at the assertion (leading edge) of -RAS.
- -RAS, -CAS, PI, and COUT are driven with the timings specified in the DMA transaction timing diagram (Appendix A, Figure A-14)
- DAL's are three-stated
- AI<7:0>, R/-WHB, and R/-WLB have internal pull-ups.

When the grant is issued external circuitry must drive the R/-WHB and R/-WLB lines, and initially drive the DAL's with the address. In dynamic memory systems the address must be multiplexed on AI<7:0> so that the memory chips are provided with row and column addresses at the appropriate times. Later in the transaction the data transfer on the DAL's takes place in a direction controlled by the state of the R/-WHB and R/-WLB lines.

2.14.4 READY Input

If the READY input is activated (refer to paragraph 3.4.6) the DMA transaction is extended by one microcycle (depending on the pulsing of READY, more microcycles may be added).

2.15 ASPI (Assert Priority In) TRANSACTION

An ASPI transaction consists of two processes:

- Input of interrupt and DMA request
- -CAS without -RAS.

Detailed timing of an ASPI transaction is found in Figure A-14 of Appendix A.

NOTE
All reference to input or output refer to the processor.

With reference to Figures 2-23 and 2-24, the processor reads AI<7:0>. If any line is asserted the processor acts on the interrupt (depending on the priority); if not, no action takes place. For information concerning the interrupt structure, refer to paragraph 1.5. The ASPI transaction generates a -CAS without generating a -RAS. ASPI transactions occur only during a RESET instruction, HALT instruction/interrupt, WAIT instruction, or during the power up sequence.

2.15.1 Input Control

The interrupt strobe, which the processor uses to latch the interrupt and DMA request data, is accomplished by means of PI. The interrupt is latched by the processor upon the negation (trailing edge) of PI.

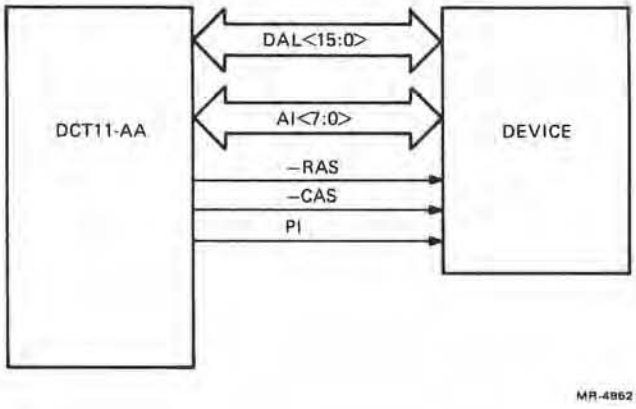


Figure 2-23 ASPI Transaction Block Diagram

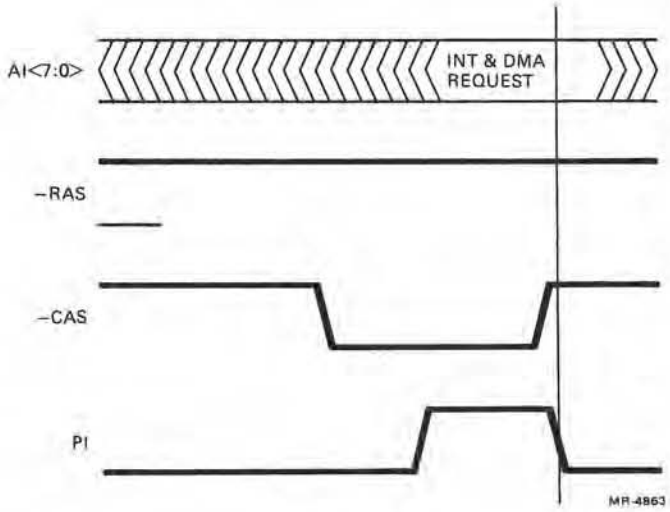


Figure 2-24 ASPI Transaction Timing

CHAPTER 3
PIN DESCRIPTION

3.1 INTRODUCTION

Chapter 3 describes the functions performed by each individual DCT11-AA pin. The pins and thus the chapter are divided into five groups:

- Data/Address Lines (DAL<15:0>)
- Address/Interrupt (AI<7:0>)
- Control lines (SEL<1:0>, R/-WHB, R/-WLB, -RAS, -CAS, PI, Ready)
- Miscellaneous signals (-BCLR, PUP, COUT, XTL1, XTL0)
- Power pins (BGND, GND, V_{cc}).

With reference to Figure 3-1, and Tables 3-1 through 3-5, several DCT11-AA pins perform different functions depending on the mode. Therefore, signal names vary from the pin names. The mode dependent pins are:

- DAL<15:0>
- AI<7:0>
- Select (SEL<1:0>)
- Read/-Write High Byte (R/-WHB)
- Read/-Write Low Byte (R/-WLB)
- Clock Output (COUT).

Each pin function is described under the pin name. If the pin is mode dependent a description of each mode is found under the pin name.

3.2 DATA ADDRESS LINES (DAL<15:0>)

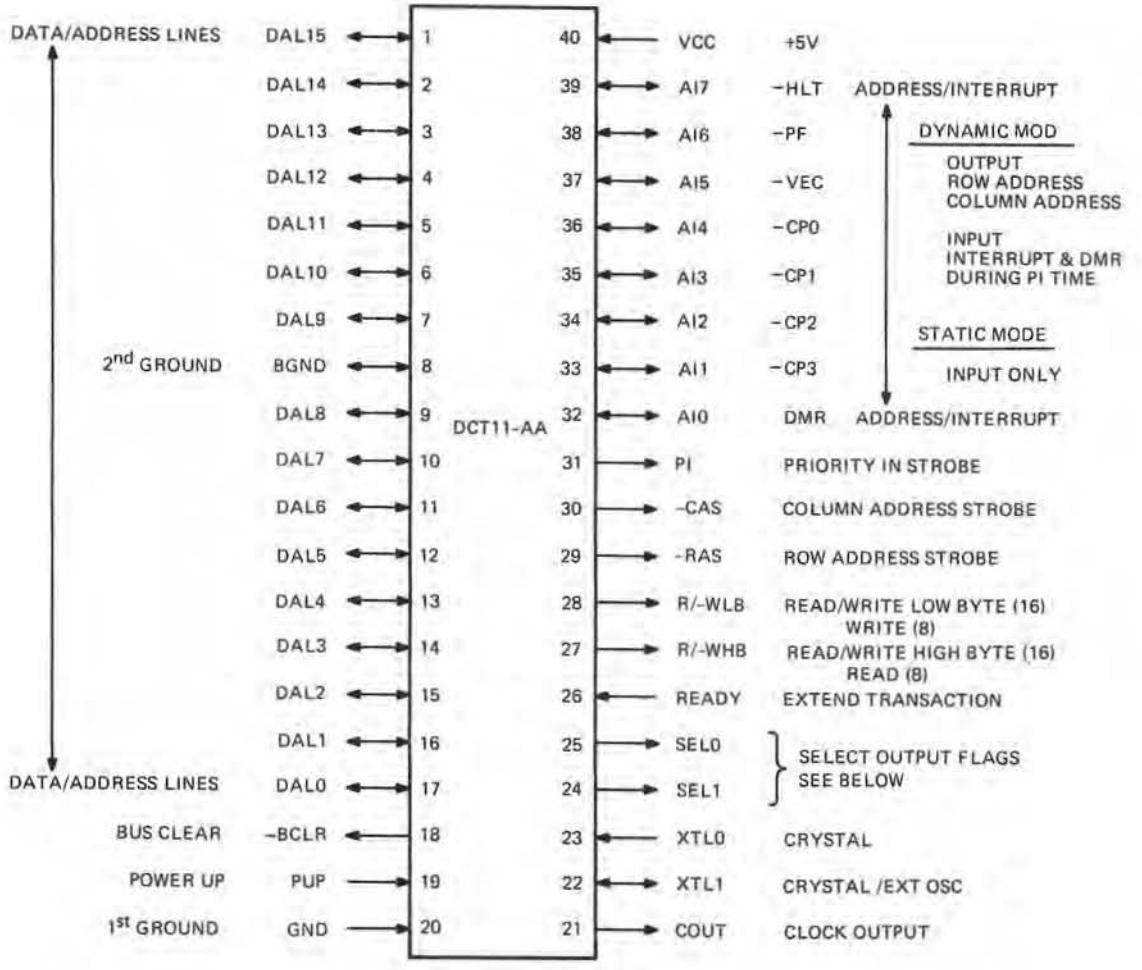
DAL<15:0> functions are dependent upon the selection of 8-bit or 16-bit mode. During read/write transactions (refer to paragraph 2.2.1) the DAL's are time multiplexed in two ways. In 16-bit mode, they multiplex the address then the data. In 8-bit mode, in addition to the address/data multiplexing there is a low byte/high byte multiplexing.

3.2.1 16-Bit Mode - DAL<15:0>

DAL<15:0> are used in six cases;

During a read/write transaction:
DAL<15:0> are time multiplexed and used for the address and the data. Read/write transactions are defined in paragraphs 2.3 - 2.10.

During an Iack transaction:
The information present on AI<5:1> at the time of the interrupt request is output on DAL<12:8>. Refer to Table 3-1. Paragraph 2.12 defines the Iack (Interrupt Acknowledge) transaction.



SELECT OUTPUT FLAGS

SEL<1>	SEL<0>	FUNCTION
L	L	READ/WRITE
L	H	REFRESH/FETCH
H	L	IACK
H	H	DMG

Figure 3-1 DCT11-AA Pin Layout

During a DMA transaction:

DAL<15:0> are three-stated. The DMA (direct memory access) transaction is defined in paragraph 2.14.

During a Busnop and Refresh Transaction:

DAL<15:0> contain previously latched data.

During an ASPI Transaction:

DAL<15:0> are three-stated.

During the power up sequence or a RESET instruction:

The mode register bits are read in from DAL<15:8, 1:0>. Low current internal pullups are enabled on these lines when -BCLR is asserted, this avoids the need to drive the bits which are to be high.

3.2.2 8-Bit Mode - DAL<15:8>

The signal name for DAL<15:8> in 8-bit mode is Static Address Lines (SAL<15:8>). They are used in six cases;

During a read/write transaction:

SAL<15:8> contains the high byte of the address throughout the transaction. In 8-bit mode two transactions (one data byte per transaction) are required for a word read or write. Read/write transactions are defined in paragraphs 2.3 - 2.10.

During an Iack transaction:

The information present on AI<5:1> at the time of the interrupt request is output on DAL<12:8>. Refer to Table 3-1. Paragraph 2.12 defines the Iack (Interrupt Acknowledge) transaction.

During a DMA transaction:

DAL<15:8> are three-stated. The DMA (direct memory access) transaction is defined in paragraph 2.14.

During a Busnop and Refresh Transaction:

DAL<15:0> contain previously latched data.

During an ASPI Transaction:

DAL<15:0> are three-stated.

During the power up sequence or a RESET instruction:

The mode register bits are read in from DAL<15:8>. Low current internal pullups are enabled on these lines when -BCLR is asserted, this avoids the need to drive the bits which are to be high.

3.2.3 8-Bit Mode - DAL<7:0>

DAL<7:0> are used in six cases;

During a read/write transaction:

DAL<7:0> are time multiplexed and used for the low byte of address and data. In 8-bit mode the data is either the low byte or the high byte. Refer to Figure 3-1. Read/write transactions are defined in paragraphs 2.3 - 2.10.

During an Iack transaction:

DAL<7:2> are used for the input of an external vector address (if -VEC was asserted during the interrupt request). DAL<1:0> are irrelevant, because DCT11-AA replaces them with a 0 after reading them in. This is due to the fact that vectors use two words: PC and PSW. Paragraph 2.12 defines the Iack (Interrupt Acknowledge) transaction.

During a DMA transaction:

DAL<7:0> are three-stated. The DMA (direct memory access) transaction is defined in paragraph 2.14.

During a Busnop and Refresh Transaction:

DAL<15:0> contain previously latched data.

During an ASPI Transaction:

DAL<15:0> are three-stated.

During the power up sequence or a RESET instruction:

The Mode Register Bits are read in from DAL<1:0>. Low current internal pullups are enabled on these lines when -BCLR is asserted, this avoids the need to drive the bits which are to be high.

Table 3-1 Mapping of AI onto DAL in Iack Transaction *

Interrupt Request Time		Iack Transaction
-CP<3>	AI<1>	DAL<8>
-CP<2>	AI<2>	DAL<9>
-CP<1>	AI<3>	DAL<10>
-CP<0>	AI<4>	DAL<11>
-VEC	AI<5>	DAL<12>
	AI<0> not mapped	
	AI<6> not mapped	
	AI<7> not mapped	
		DAL<7:0> don't care
		DAL<15:13> don't care

* - The logic level is maintained in the AI to DAL mapping. As an example, if AI<1> is high at interrupt request time, then DAL<8> is high at Iack time.

Table 3-2 Signal and Pin Utilization 16-Bit Mode

Pin	Pin Name	Signal Names		
		Static	4K/16K Dynamic	64K Dynamic
Data Address Lines				
1-7&9	DAL<15:8>	DAL<15:8>	DAL<15:8>	DAL<15:8>
10-17	DAL<7:0>	DAL<7:0>	DAL<7:0>	DAL<7:0>
Address Interrupt Lines				
			-RAS -CAS PI	-RAS -CAS PI
32	AI<0>	-DMR	FET[1] A14 -DMR	A15 A14 -DMR
33	AI<1>	-CP<3>	A1 A2 -CP<3>	A1 A2 -CP<3>
34	AI<2>	-CP<2>	A3 A4 -CP<2>	A3 A4 -CP<2>
35	AI<3>	-CP<1>	A5 A6 -CP<1>	A5 A6 -CP<1>
36	AI<4>	-CP<0>	A7 A8 -CP<0>	A7 A8 -CP<0>
37	AI<5>	-VEC	A9 A10 -VEC	A9 A10 -VEC
38	AI<6>	-PF	A11 A12 -PF	A11 A12 -PF
39	AI<7>	-HALT	A13 A14 -HALT	A13 A14 -HALT
Control Signals				
24	SEL1[2]	Iack+DMG	Iack+DMG	Iack+DMG
25	SEL0[2]	FET+DMG	REF+DMG	FET+DMG
26	READY	READY	READY	READY
27	R/-WHB	R/-WHB	R/-WHB	R/-WHB
28	R/-WLB	R/-WLB	R/-WLB	R/-WLB
29	-RAS	-RAS	-RAS	-RAS
30	-CAS	-CAS	-CAS	-CAS
31	PI	PI	PI	PI
Miscellaneous Signals				
18	-BCLR	-BCLR	-BCLR	-BCLR
19	PUP	PUP	PUP	PUP
21	COUT	COUT	COUT	COUT
22	XTL1	XTL1	XTL1	XTL1
23	XTL0	XTL0	XTL0	XTL0
Power Pins				
8	BGND	BGND	BGND	BGND
20	GND	GND	GND	GND
40	V_{cc}	V_{cc}	V_{cc}	V_{cc}

NOTES

1 During -RAS, AI<0> is used to indicate a fetch operation in progress. During refresh, AI<0> is the output of the refresh counter at -RAS time.

2 SEL<1> and SEL<0> are encoded refer to Tables 3-4 and 3-5.

Table 3-3 Signal and Pin Utilization 8-Bit Mode

Pin	Pin Name	Signal Names							
		Static	4K/16K Dynamic			64K Dynamic			
Data Address Lines									
1-7&9	DAL<15:8>	SAL<15:8>	SAL<15:8>			SAL<15:8>			
10-17	DAL<7:0>	DAL<7:0>	DAL<7:0>			DAL<7:0>			
Address Interrupt Lines									
			-RAS	-CAS	PI	-RAS	-CAS	PI	
32	AI<0>	-DMR	FET[1]	A14	-DMR	A15	A14	-DMR	
33	AI<1>	-CP<3>	A1	A2	-CP<3>	A1	A2	-CP<3>	
34	AI<2>	-CP<2>	A3	A4	-CP<2>	A3	A4	-CP<2>	
35	AI<3>	-CP<1>	A5	A6	-CP<1>	A5	A6	-CP<1>	
36	AI<4>	-CP<0>	A7	A8	-CP<0>	A7	A8	-CP<0>	
37	AI<5>	-VEC	A9	A10	-VEC	A9	A10	-VEC	
38	AI<6>	-PF	A11	A0	-PF	A11	A0	-PF	
39	AI<7>	-HALT	A13	A12	-HALT	A13	A12	-HALT	
Control Signals									
24	SEL1[2]	Iack+DMG	Iack+DMG			Iack+DMG			
25	SEL0[2]	FET+DMG	REF+DMG			FET+DMG			
26	READY	READY	READY			READY			
27	R/-WHB	-RD	-RD			-RD			
28	R/-WLB	-WT	-WT			-WT			
29	-RAS	-RAS	-RAS			-RAS			
30	-CAS	-CAS	-CAS			-CAS			
31	PI	PI	PI			PI			
Miscellaneous Signals									
18	-BCLR	-BCLR	-BCLR			-BCLR			
19	PUP	PUP	PUP			PUP			
21	COUT	COUT	COUT			COUT			
22	XTL1	XTL1	XTL1			XTL1			
23	XTL0	XTL0	XTL0			XTL0			
Power Pins									
8	BGND	BGND	BGND			BGND			
20	GND	GND	GND			GND			
40	V_{cc}	V_{cc}	V_{cc}			V_{cc}			

NOTES

1 During -RAS, AI<0> is used to indicate a fetch operation in progress. During refresh, AI<0> is the output of the refresh counter at -RAS time.

2 SEL<1> and SEL<0> are encoded refer to Tables 3-4 and 3-5.

Table 3-4 SEL<1:0> Functions in Static Mode or Dynamic 64K Mode

SEL<1>	SEL<0>	Function
L	L	read, write, ASPI, or busnop
L	H	fetch (PDP-11 instruction fetch)
H	L	Iack (interrupt acknowledge)
H	H	DMG (direct memory grant)

Table 3-5 SEL<1:0> Functions in Dynamic 4K/16K Mode

SEL<1>	SEL<0>	Function
L	L	read, write, ASPI, or busnop
L	H	refresh
H	L	Iack (interrupt acknowledge)
H	H	DMG (direct memory grant)

3.3 ADDRESS INTERRUPT (AI<7:0>)

During read, write, refresh, DMA, and ASPI transactions the AI lines (AI<7:0>) perform various functions. The function of AI<7:0> depends upon the selection of one of the following modes, static or dynamic mode and 4K/16K or 64K mode. Three functions are time multiplexed on AI<7:0>:

- Output of row address
- Output of column address
- Input of interrupts and/or DMA requests.

During Busnop and Iack transactions, AI<7:0> act as inputs in static modes and contain previously latched data in dynamic modes. The AI lines are described in three parts:

- At -RAS and -CAS time (static mode)
- At -RAS and -CAS time (dynamic mode)
- At PI time (static or dynamic mode).

3.3.1 AI<7:0> At -RAS and -CAS Time (Static Mode)

The address interrupt lines are used as inputs for interrupts and/or DMA requests during all transactions while in static mode. AI<7:0> are implemented by internal active low current pull ups.

3.3.2 AI<7:0> At -RAS and -CAS Time (Dynamic Mode)

During read/write transactions the address interrupt lines are used as outputs at -RAS and -CAS time only. The AI's are time multiplexed in two ways:

- Prior to the assertion (leading edge) of Row Address Strobe (-RAS) the AI lines output the row address for a dynamic RAM. At the occurrence of -RAS, the data on the AI lines is valid.

- Prior to the assertion (leading edge) of Column Address Strobe (-CAS) the AI lines output the column address for a dynamic RAM. At the occurrence of -CAS, the data on the AI lines is valid.

During refresh transactions AI<7:0> are used to output the row address at -RAS time.

During DMA and ASPI transactions AI<7:0> have internal low current pull-ups and are used as inputs.

NOTE

The dynamic address on AI<7:0> available at -RAS and -CAS time is duplicated on DAL<15:0> at -RAS time.

3.3.3 AI<7:0> At Priority In (PI) Time (Dynamic and Static Mode)

During read/write, DMA, and ASPI transactions, at PI time, AI<7:0> are used as inputs. These lines are implemented by internal low current pull-ups. The AI lines input interrupt and DMA requests at the negation (trailing edge) of PI. Refer to Table 3-6.

NOTE

DCT11-AA does not react to interrupt requests posted during write and DMA transactions.

Interrupt and DMA requests are implemented by the following signals:

-DMR (Direct Memory Request) AI<0>. When the processor reads a DMA request asserted, it (upon termination of almost any current bus transaction) frees the bus for the DMA device. Refer to paragraph 2.14 for a definition of a DMA transaction.

-CP<3:0> (Coded Priority) AI<1:4>. Logic internal to the processor decodes these inputs as an interrupt request on one of four maskable levels. Refer to paragraph 1.5 for the definition of the DCT11-AA interrupt structure.

-VEC (Vector) AI<5>. The signal has meaning only if one or more of -CP<3:0> is asserted. -VEC signals the processor to ignore the internal vector address indicated by -CP<3:0> and use instead the vector address to be provided by the user. The priority of the -CP lines is not ignored. The user provided vector address is read during the Iack Transaction.

-PF (Power Fail) AI<6>. -PF has the highest priority on level seven. If -PF and a level seven request from CP<3:0> are both present at PI time, the DCT11-AA services the -PF first by stacking the PC and PS and jumping to vector address (24). The input circuit requires no data set up time. Internal logic samples the -PF and then pauses for up to one instruction before recognizing a request. The -PF input is pseudo edge sensitive. It must be read as a negation before another assertion is recognized.

-HALT (Halt) AI<7>. -HALT is an unmaskable interrupt. It always causes a jump, after stacking the PS and PC, to the restart address with PS = 340. The -HALT input is pseudo edge sensitive. It must be read as a negation before another assertion is recognized.

Table 3-6 AI Functions

Transaction	@ -RAS (L.E.) Output	@ -CAS (L.E.) Output	@ PI (T.E.) Input
Read (static)	*	*	interrupt/DMR
Write (static)	*	*	DMR
Read (dynamic)	row address	column address	interrupt/DMR
Write (dynamic)	row address	column address	DMR
Refresh	row address	N/A	N/A
DMA	*	*	DMR
ASPI	N/A	*	interrupt/DMR

N/A - not applicable
* - internal low current passive pullups

3.4 CONTROL LINES

The control lines are comprised of signals which the DCT11-AA uses to control the normal operation of the system. The lines are:

- -RAS
- -CAS
- PI
- R/-WHB
- R/-WLB
- SEL<1>
- SEL<0>
- READY.

Table 3-7 indicates the transactions in which each of these signals is used. During all transactions not mentioned in the following description, the control lines remain in their unasserted state (except READY which is an input).

-RAS, -CAS, and PI are control strobes and act on a logic transition. R/-WHB, R/-WLB, SEL<1>, SEL<0>, and READY are static control lines and act on a logic level. Figure 3-2 provides an illustration of leading and trailing edges. The leading edge is the edge that changes the signal from the unasserted state to the asserted state.

Table 3-7 Control Signal Usage

Transaction	-RAS	-CAS	PI	R/-WHB	R/-WLB	SEL<0>	SEL<1>	READY
Read/Write	X	X	X	X	X	1		*
Refresh	X					2		
Iack	X						X	*
DMG	X	X	X	3	3	X	X	*
ASPI		X	X					

X - asserted
* - causes one or more microcycle slips
1 - Asserted in static mode and dynamic 64K mode when read is a PDP-11 instruction fetch. In 8-bit mode it is asserted only in the low byte transaction of a fetch.
2 - asserted in dynamic 4K/16K mode
3 - three-stated

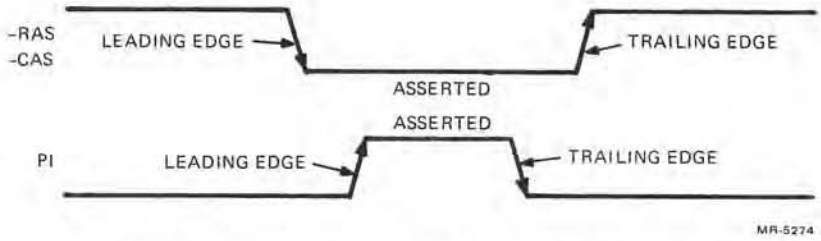


Figure 3-2 Leading and Trailing Edge

3.4.1 -RAS (Row Address Strobe)

-RAS is the system address strobe. Table 3-7 indicates the transactions in which -RAS is asserted. During read/write transactions the assertion (leading edge) of -RAS is used to strobe the address which is present on the DAL's (for memories not using the -RAS/-CAS multiplexing) and the row address which is present on the AI's (for the dynamic memories which use it). During a write transaction the negation (trailing edge) of -RAS may be used as the data output strobe.

During a refresh transaction (dynamic mode only) the assertion (leading edge) of -RAS is used to strobe the row address which is present on the AI lines.

During an Iack transaction the assertion (leading edge) of -RAS strobes the Iack information, which is present on DAL<12:8>, to the system. The negation (trailing edge) of -RAS strobes the vector address (user supplied) into the DCT11-AA.

During a DMA transaction, -RAS provides the DMA device with the same function and timing as used in read/write transactions.

3.4.2 -CAS (Column Address Strobe)

-CAS is an address and chip select strobe. Table 3-7 indicates the transactions in which -CAS is asserted. During read/write transactions -CAS provides various functions:

- The assertion (leading edge) of -CAS provides an early warning of the impending occurence of PI and therefore may be used to latch interrupt and DMA requests before strobing them onto the AI lines.
- In dynamic read/write transactions the assertion (leading edge) of -CAS strobes the column address present on the AI lines.
- In read transactions the negation (trailing edge) of -CAS is used to strobe the data (user supplied) from the DAL's into the DCT11-AA.
- In write transactions the negation (trailing edge) of -CAS may be used as the data output strobe.

During a DMA transaction, the assertion (leading edge) of -CAS provides the DMA device with the same function and timing as used in read/write transactions.

During ASPI transactions, the assertion (leading edge) of -CAS may be used to latch interrupt and DMA requests before strobing them onto the AI lines.

3.4.3 PI (Priority In)

PI is the system interrupt request strobe. PI is used in read, write, DMA, and ASPI transactions. Refer to Tables 3-6 and 3-7. The function and timing of PI is the same in all four transactions.

Whenever PI is asserted the AI lines are used as inputs. These lines are implemented by internal low current pull-ups. Therefore the assertion (leading edge) of PI can be used to strobe the signals -HALT, PF, -VEC, -CP<3:0>, and DMR onto the AI lines (Refer to paragraph 3.3.).

During write transactions both the assertion (leading edge) and the negation (trailing edge) of PI can be used as a data output strobe.

During write transactions PI can be used to gate the write enable signals (R/-WHB and R/-WLB) for memories and peripherals requiring write enable after the assertion of -CAS.

3.4.4 R/-WHB and R/-WLB

The signal names for pin 27 (R/-WHB) and pin 28 (R/-WLB) change according to the selection of 8-bit or 16-bit data bus mode.

3.4.4.1 R/-WHB and R/-WLB 16-Bit Mode -- The write enable signals Read/-Write High Byte (R/-WHB) and Read/-Write Low Byte (R/-WLB) are used exclusively in read/write transactions. R/-WHB and R/-WLB are asserted (low) when the transaction is a write to a high byte or a low byte.

Normal or delayed mode affects the timing of R/-WHB and R/-WLB. In normal mode, the read/write timing is compatible with that of the Motorola 6800 bus peripherals. In delayed mode, the timing is compatible with that of the Intel 8080 bus peripherals.

During a DMA transaction both pins are internal low current pull-ups.

3.4.4.2 R/-WHB (-RD) and R/-WLB (-WT) 8-Bit Mode -- The mutually exclusive signals -RD (read enable) and -WT (write enable) are used only in read/write transactions. -RD is asserted low during a read transaction and -WT is asserted low during a write transaction.

Normal or delayed mode affects the timing of -RD and -WT. In normal mode, the read/write timing is compatible with that of the Motorola 6800 bus peripherals. In delayed mode, the timing is compatible with that of the Intel 8080 bus peripherals.

During a DMA transaction both pins are internal low current pull-ups.

3.4.5 SEL<1> and SEL<0>

Select 1 (SEL<1>) and Select 0 (SEL<0>) are encoded lines and indicate which transaction is being performed. Refer to Tables 3-4 and 3-5.

3.4.6 READY

Through the use of the READY signal, I/O devices or memory of any speed may be synchronized to the DCT11-AA. The READY signal is not generated by the DCT11-AA but by some peripheral device. The signal is input to the DCT11-AA via the READY input. The signal is used to place the DCT11-AA into an idle state while the peripheral device finishes its operation.

With reference to Figure 3-3, a single assertion of READY causes a single microcycle slip. Any additional cycle slips require the READY signal to be pulsed again. The assertion of READY does not have any effect unless -RAS is also asserted. The microcycle slip starts after the assertion of -RAS, -CAS, and PI leading edges. A single microcycle slip occurs during every bus transaction if the READY input is connected to ground.

The READY signal extends the following transactions:

- Read
- Write
- Iack
- DMA.

Detailed timing of READY is found in Figure A-15 of Appendix A.

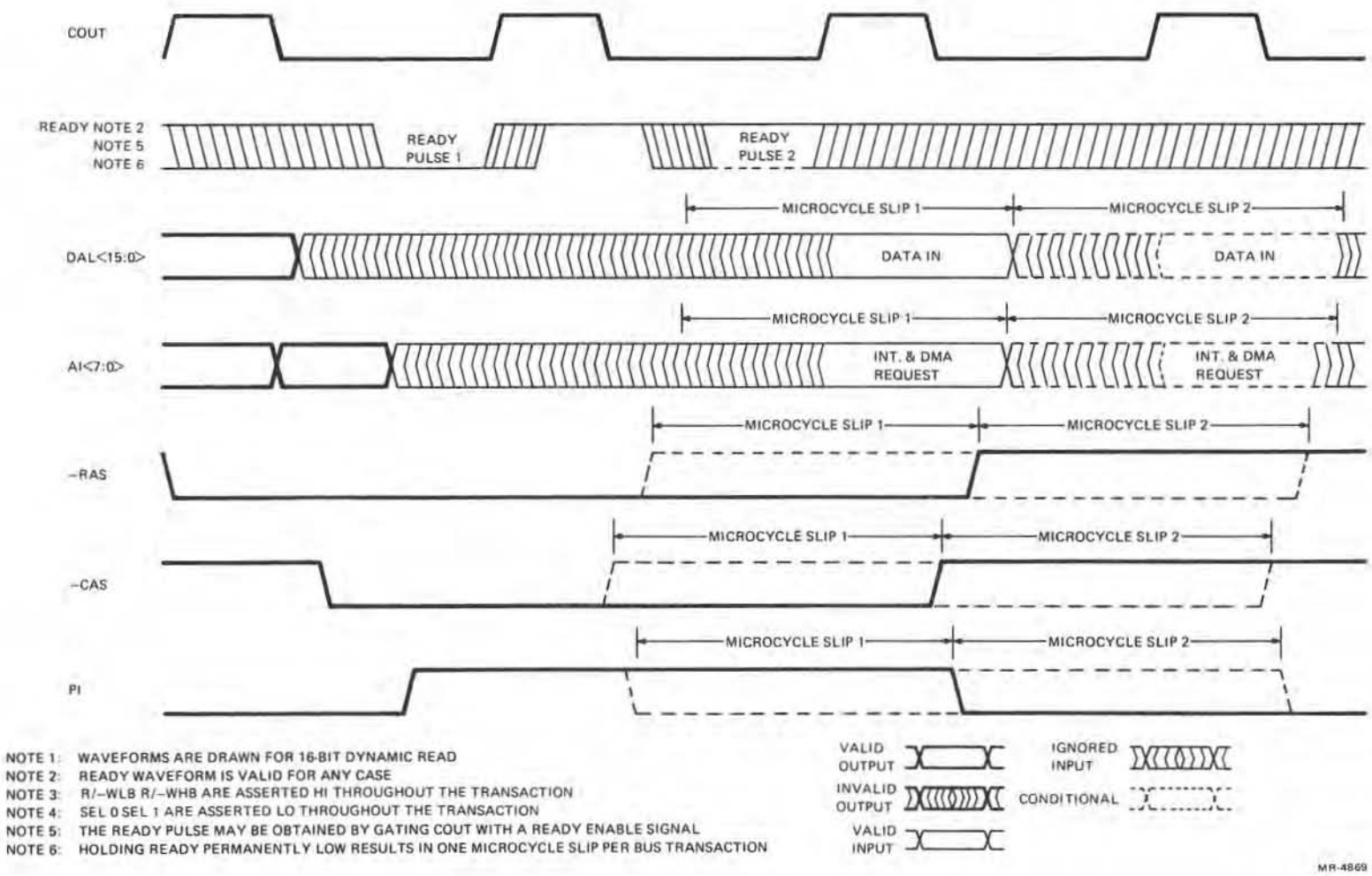


Figure 3-3 READY Timing

3.5 MISCELLANEOUS SIGNALS

This group of signals includes the following:

- -BCLR
- PUP
- COUT
- XTL1
- XTL0.

3.5.1 -BCLR (Bus Clear)

The signal -BCLR on pin 18 is asserted low by the processor during the power up sequence and during the execution of a PDP-11 RESET instruction. -BCLR asserted (low) enables the mode register pullups on DAL<15:8, 1:0>. The -BCLR pin must be connected to ground through a 1Kohm, 1% resistor. Its fan out is given in Table A-2 (DC characteristics) of Appendix A.

3.5.2 PUP (Power Up)

PUP is a schmitt-triggered input and has a low current internal pull down that is always enabled. When PUP is forced high, the schmitt-trigger senses the transition. When the processor detects a change from high back to low the power up sequence begins.

If PUP is asserted high during a DCT11-AA operation, the current transaction is terminated and all internal registers go to an undefined state. The DAL's and AI lines output undefined data and the control and miscellaneous signals are in an unasserted state. As soon as PUP is asserted low the power up sequence begins.

The power up sequence is a succession of events which initializes the DCT11-AA. The events occur in two cases;

1. When V_{CC} is applied:
   - PUP changes state (low to high)
   - Assertion of -BCLR output
   - PUP changes state (high to low)
   - Mode register load
   - Clearing of -BCLR output
   - 20 refresh transactions (8-bit Dynamic), 10 refresh transactions (16-bit Dynamic) or 20 busnop transactions (8-bit Static), 10 busnop transactions (16-bit Dynamic)
   - Load the stack pointer to 376, the program counter to start address, and the processor status word to 340
   - ASPI transaction.

2. When a RESET instruction is executed:
   - Assertion of -BCLR output
   - Mode register load
   - Clearing of -BCLR output
   - ASPI transaction.

Detailed timing of power up is found in Figure A-16 of Appendix A.

3.5.2.1 Power Up (PUP) Input -- With reference to Figures 3-4 and 3-5, the processor detects a transition from low to high on the PUP input. The transition is sensed by an internal Schmitt trigger which provides a clean, fast, edge when the input reaches a predetermined level (TTL V_{il} = .8v). When the processor detects a change from high back to low the mode register load begins.

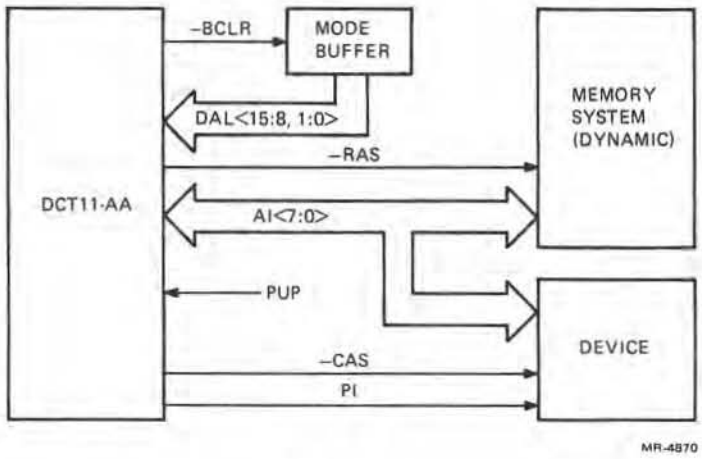


Figure 3-4 Power Up Sequence Block Diagram

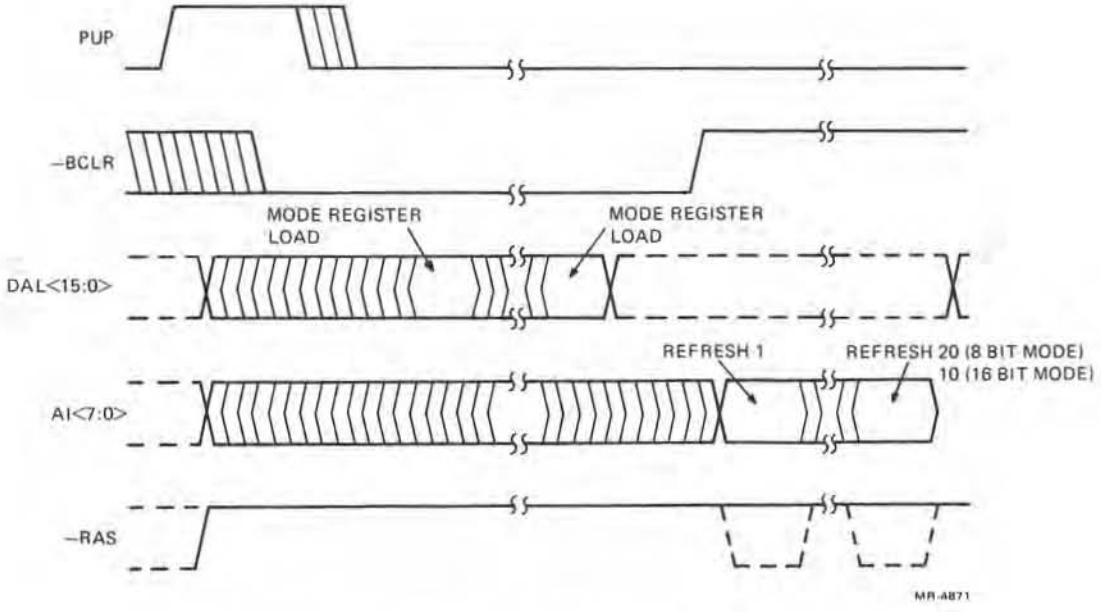


Figure 3-5 Power Up Sequence Timing

3.5.2.2 **Bus Clear (-BCLR)** -- As a result of PUP being high, the processor is forced to an initial condition with undefined register states. It is at this time (PUP high) that -BCLR is asserted. -BCLR is also asserted as a result of a program RESET instruction. -BCLR is a strobe which is used by the user to enable pulldowns on Data Address Lines (DAL) 15 through 8 and 1 through 0 (<15:8>, <1:0>) at mode register read time. The mode register is loaded through DAL<15:0>. However, DAL<7:2> are reserved. -BCLR may also be used to initialize the rest of the system.

3.5.2.3 **Mode Register Load** -- The mode register input begins after -BCLR is asserted and PUP is low. The load process continues until the microcode returns -BCLR to a high.

3.5.2.4 **Refresh or Busnop Transactions** -- Depending on the condition of the mode register the processor generates either refresh or busnop transactions. Refer to Table 3-8 for the conditons and the number transactions generated.

Table 3-8 Refresh and Busnop

MODE	BUSNOP	REFRESH
8 bit/dynamic		20
16 bit/dynamic		10
8 bit/static	20	
16 bit/static	10	

3.5.2.5 **Load the SP, PC, and PSW** -- After the completion of the refresh or busnop transactions the processor loads the Stack Pointer (SP) with 376 (octal). The Program Counter (PC) is loaded with the start address and, finally, the processor Status Word (PSW) is loaded with 340 (octal).

3.5.2.6 **ASPI Transaction** -- The last process in the power up sequence is an ASPI transaction to check for interrupts and DMA. At the completion of the ASPI transaction, normal operation begins. Refer to paragraph 2.15 for details on the ASPI transaction.

3.5.3 COUT (Clock Output)

COUT outputs a TTL level clock which is a function of mode register bit 0 (MR<0>). MR<0> determines if the output is to be processor mode clock (MR<0> = 1) or constant clock (MR<0> = 0). With reference to Figure 3-6, in constant clock mode the output is at a frequency half that of the operating frequency (the frequency of XTL0 and XTL1). In processor clock mode a clock pulse is asserted once every microcycle (every three or four oscillator periods).

Detailed timing of COUT is found in Figure A-17 of Appendix A.

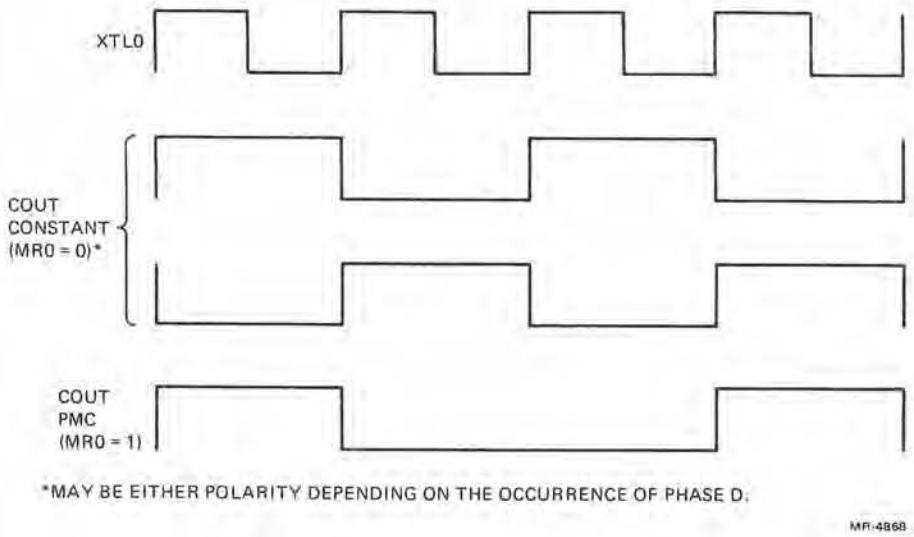


Figure 3-6 COUT

3.5.4 XTL1 and XTL0 (Crystal Inputs)

These two pins (22 and 23) are the external crystal connections to the internal clock generator. If an external TTL clock is used, it must be applied to XTL1 (pin 22) and XTL0 (pin 23) must be grounded.

Detailed timing of XTAL is found in Figure A-17 of the Appendix A.

3.6 POWER PINS

The following are pins associated with the power source to the DCT11-AA:

- BGND
- GND
- V_{cc}.

3.6.1 GND and BGND

BGND and GND should be connected together. They provide the reference ground for all lines of the DCT11-AA.

3.6.2 V_{cc}

Pin 40 is the +5 volt supply for the DCT11-AA. This voltage must be maintained to within +/-5% of 5 volts.

CHAPTER 4

MODE SELECTION

4.1 INTRODUCTION

Most of the DCT11-AA features are programmable through the use of an internal 16-bit Mode Register (MR). The DCT11-AA must be programmed during the power up sequence and may be reprogrammed when the PDP-11 RESET instruction is executed.

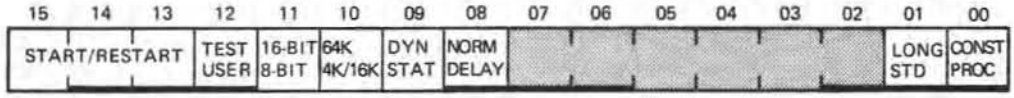
This chapter is divided into four sections:

- Description of modes related to function
- Description of modes related to timing
- Mode register bit setting
- Mode register selection guidelines.

4.2 MODES RELATED TO FUNCTION

With reference to Figure 4-1 and Table 4-1, the modes related to function effect the functionality of the processor. These modes are:

- 16-bit or 8-bit data bus MR<11>
- Dynamic or static memory MR<9>
- 64K or 4K/16K memory chip size MR<10>
- Tester or user MR<12>
- Start/restart address MR<15:13>.



<15:13>	START/RESTART ADDRESS		
12	TESTER/USER MODE	08	NORMAL/DELAYED R/W
11	16-BIT/8-BIT BUS	<7:2>	RESERVED
10	64K/4K OR 16K MEMORY	01	LONG/STANDARD MICROCYCLE
09	DYNAMIC/STATIC MEMORY	00	CONSTANT/PROCESSOR MODE CLOCK

ADDRESS BITS <15:13>	START ADDRESS	RESTART ADDRESS
7	172000	172004
6	173000	173004
5	000000	000004
4	010000	010004
3	020000	020004
2	040000	040004
1	100000	100004
0	140000	140004

MR-4843

Figure 4-1 Mode Register

Table 4-1 Mode Register Bit Settings

Mode Register Bit	State	Mode
0	1	Processor clock
	0	Constant clock
1	1	Standard microcycle
	0	Long microcycle
8	1	Delayed read/write
	0	Normal read/write
9	1	Static memory
	0	Dynamic memory
10	1	4K/16K memory
	0	64K memory
11	1	8 bit bus
	0	16 bit bus
12	1	User
	0	Tester

4.2.1 16-Bit or 8-Bit Mode MR<11>
Mode register bit 11 determines if the processor operates the data bus in 8-bit mode or 16-bit mode. The selection of either 8-bit or 16-bit data bus effects the DAL<15:0>, R/-WHB, R/WLB, and AI<7:6> lines, during read/write transactions. It also determines the number of transactions needed to read or write a word.

4.2.1.1 16-Bit Mode -- If mode register bit 11 is asserted low (MR<11> = 0) 16-bit data bus mode is selected and the following occurs in a read or write transaction (refer to Figures 2-2 through 2-9);

Data address lines:
DAL<15:0> - output of 16-bit address before the assertion (leading edge) of -RAS
DAL<15:0> - input or output of 16-bit data at read/write time.

Read/write control:
Each byte of a PDP-11 16-bit word is assigned a seperate write control signal (R/-WHB and R/-WLB).

4.2.1.2 8-Bit Mode -- If mode register bit 11 is not asserted (MR<11> = 1) 8-bit data bus mode is selected. Two transactions are required to perform a word read or word write. The following occurs during a word read or word write operation (refer to Figures 2-10 through 2-17);

Data address lines:
DAL<15:0> - output of 16-bit address before the assertion (leading edge) of -RAS
DAL<15:8> - the signal names for these pins are Static Address Lines (SAL<15:8>) and they hold the high byte address throughout the two transactions
DAL<7:0> - contains the low byte of address during read/write time of the first transaction and the data during the read/write time of the second transaction.

Read/write control:

A seperate read/write control signal is provided for a read and for a write. The read/write control signals are Read (-RD, pin name R/-WHB) and Write (-WT, pin name R/-WLB). They are mutually exclusive.

4.2.2 Dynamic or Static Mode MR<9>

Mode register bit nine determines if the processor supports dynamic or static memories. This mode affects the operation of the AI lines and SEL<1:0> during read/write transactions, and the occurence of the refresh transaction (which adds time to the instruction execution time).

4.2.2.1 Dynamic Mode -- If mode register bit nine is asserted low (MR<9> = 0), dynamic mode is selected and dynamic memories are directly supported. Besides outputting the address on the DAL<15:0> before the assertion of -RAS, DCT11-AA also outputs row and column addresses on the AI<7:0>. The row address is output before the assertion (leading edge) of -RAS which strobes it into the memory chips. The column address is output before the assertion (leading edge) of -CAS which strobes it into the memory chips. In addition, automatic refresh is provided by means of the refresh transaction (refer to paragraph 2.11).

4.2.2.2 Static Mode -- If mode register bit nine is not asserted (MR<9> = 1) static mode is selected. The memory is addressed using DAL<15:0> at -RAS time and no refresh is provided. AI<7:0> are used only for inputting interrupt and/or DMA information.

4.2.3 64K or 4K/16K Mode MR<10>

Mode register bit 10 applies to dynamic mode only (in static mode it does not have any effect) and is used for selecting the dynamic memory chip type. In 64K mode (MR<10> = 0), memory chips such as 64K X 1-bit are supported.

In 4K/16K mode (MR<10> = 1), memory chips such as 4K X 1-bit or 16K X 1-bit are supported. Refer to Table 4-2.

Table 4-2 DCT11-AA Modes

Class	Bit	Mode Name	Function
Modes related to function.	MR<9>	static or dynamic	dynamic RAM support
	MR<10>	4K/16K or 64K	RAM chip type
	MR<11>	8-bit or 16-bit bus	data bus width
	MR<12>	tester or user	tester or user
	MR<15:13>	start/restart	start/restart address
Modes related to timing.	MR<0>	processor clock or constant clock	COUT timing
	MR<1>	long or standard microcycle	microcode length
	MR<8>	normal or delayed read/write	read/write timing

4.2.4 Tester or User Mode MR<12>

Tester mode is for Digital Equipment Corporation use only.

If mode register bit 12 is high (MR<12> = 1) then user mode is selected.

4.2.5 Start and Restart Address MR<15:13>

Mode register bits 15 through 13 are used to specify one of eight start/restart addresses. The start address is internally loaded into the Program Counter (PC) during the power-up sequence. For details on the power-up sequence refer to paragraph 3.5.2. The restart address is loaded into the PC when a HALT interrupt is received or during the execution of a PDP-11 HALT instruction. Figure 4-1 indicates the available start/restart addresses.

4.3 MODES RELATED TO TIMING

The modes related to timing affect the timing of the processor but not its funtionality. These modes are:

- Constant or processor clock MR<0>
- Long or standard microcycle MR<1>
- Normal or delayed read/write MR<8>.

4.3.1 Constant or Processor Clock MR<0>

If mode register bit zero is asserted low (MR<0> = 0) constant clock mode is selected. The output of COUT (pin 21) is a continuous clock waveform at a frequency half that of the operating frequency (the frequency at XTL0 and XTL1).

If mode register bit zero is high (MR<0> = 1) processor clock mode is selected. In processor clock mode, COUT outputs a clock pulse once every microcycle at phase W. This will occur every three or four clock phases depending on the presence of phase D.

4.3.2 Long or Standard Microcycle MR<1>

Mode register bit one allows for the selection of a long or standard microcycle. If the bit is low (MR<1> = 0) long microcycle mode is selected. Long microcycle mode is used in conjunction with memory or peripherial chips which require a long access time. When long microcycle mode is selected, all microcycles are made up of four operating frequency periods (they all contain 0D).

If mode register bit one is high (MR<1> = 1) a standard microcycle takes place. A standard microcycle is three or four operating frequency periods long depending on the type of transaction.

4.3.3 Normal or Delayed Read/Write MR<8>

If mode register bit eight is low (MR<8> = 0) the DCT11-AA is in the normal read/write mode. In normal read/write mode, the read/write control lines (R/-WHB and R/-WLB) become valid before the assertion (leading edge) of -RAS and remain valid after its negation (trailing edge).

If mode register bit eight is not asserted (MR<8> = 1) the DCT11-AA is in the delayed read/write mode and the read/write control signals have the same timing as -CAS.

4.4 MODE REGISTER BIT SETTING

During the power up sequence or when the RESET instruction is executed. At this time the DCT11-AA asserts (low) the Bus Clear (-BCLR) signal which may be used to enable external drivers. The external drivers assert specific bits on the DAL's to load the desired mode in the mode register. The data on the DAL's must be stable through the duration of the -BCLR pulse.

NOTE

The assertion of -BCLR enables active internal pull-ups on DAL<15:8, 1:0>. Only those mode register bits that must be driven low need be asserted.

4.5 MODE REGISTER SELECTION GUIDELINES

The general guidelines given in this section start from the assumption that a DCT11-AA user tries to achieve one or more of the following goals:

- minimize cost
- maximize speed
- minimize size (chip count)
- minimize development time.

The suggested user modes are listed in order of their influence upon the desired goal.

4.5.1 Minimize Cost

In order to minimize the cost of a system the implementation of the following modes is suggested:

- 8-bit
- dynamic
- long microcycle.

4.5.1.1 8-Bit Mode -- This mode allows the use of 8-bit wide device registers, data bus, and memories. In this mode the minimum memory (typically n X 1 organization) uses eight chips.

4.5.1.2 Dynamic Mode -- Although dynamic RAMs require refresh logic (provided by the DCT11-AA) they provide greater memory capacity at less cost.

4.5.1.3 Long Microcycle Mode -- Long microcycle mode allows for the use of slower (less expensive) chips.

4.5.2 Maximize Speed

In order to maximize the speed of a system the implementation of the following modes is suggested:

- 16-bit
- static
- standard microcycle.

4.5.2.1 16-Bit Mode -- Every word read or word write operation is performed in a single transaction verses two in 8-bit mode. 16-bit mode is typically 50% to 70% faster than 8-bit mode.

4.5.2.2 Static Mode -- In static mode no refresh transactions occur. Without refresh transactions a 10% time savings for computational code is possible.

4.5.2.3 Standard Microcycle -- A minor savings in time is possible through the use of this mode because of the use of faster chips.

4.5.3 Minimize Size (Chip Count)

In order to minimize the size (chip count) of a system the implementation of the following modes is suggested:

- 8-bit
- static

4.5.3.1 8-Bit Mode -- This mode allows the use of 8-bit wide device registers, data bus, and memories. In this mode the minimum memory (typically n X 1 organization) uses eight chips.

4.5.3.2 Static Mode -- Static mode can take advantage of n X 4 and n X 8 static RAMs inorder to minimize chip count.

4.5.4 Minimize Development Time

In order to minimize the development time of a system the implementation of the following modes is suggested:

- 16-bit
- static

4.5.4.1 16-Bit Mode -- A 16-bit system is simpler to develop than an 8-bit system because in 16-bit mode a single transaction performs a word read and a word write. A 16-bit system is easier to debug.

4.5.4.2 Static Mode -- A static mode system is simpler to develop because no refresh transactions are needed. Also, in a static system the AI lines are inputs at all times.

CHAPTER 5

INTERFACING

5.1 INTRODUCTION

This chapter contains information that is useful for interfacing the DCT11-AA to most systems. The chapter does not provide answers to all the possible questions, but offers a few examples and solutions that will enable the reader to get started. Interfacing information is presented for the following areas:

- Power up
- Loading the Mode register
- System clock
- Address latch and decode
- Memory subsystems
- Interrupts
- DMA
- Working with peripheral chips

NOTE

This chapter assumes that the reader is familiar with the material presented in the previous chapters.

5.2 POWER UP

With reference to Figure 5-1, a simple circuit can be constructed from a single ceramic capacitor C. The capacitor must satisfy the following conditions:

- $C \geq .04$ uf
- C (uf) $\geq .05\ t_r$ (ms).

t_r is the rise time of V_{cc}.

NOTE

The DCT11-AA powers up in an undefined state (regardless of the state of PUP) until V_{cc} is stable at V_{cc} minimum.

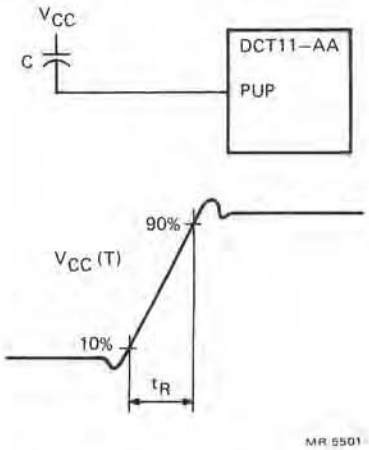


Figure 5-1 Power Up Circuit

5.3 LOADING THE MODE REGISTER

Figure 5-2 indicates how to program the mode register. On power up or when executing a RESET instruction the -BCLR pin is asserted low, this enables the desired bits onto the Data Address Lines (DAL). While -BCLR is asserted the DAL's map one for one onto the internal mode register. When -BCLR is negated the mode register is write-protected and the LS244 (buffer) shows a three-state load onto the DAL's. Unasserted bits may be left floating, since they are pulled-up internally by the DCT11-AA when -BCLR is low. -BCLR is buffered to provide enough drive for the system initialization. All devices in the system (except the buffer containing data for the mode register) should 3-state their outputs connected to DAL<15:0> at -BCLR time. This is done to preclude the mode register from being loaded with questionable data.

NOTE

The pull-down resistor on -BCLR must be 1K ohm @ 1% to guarantee timing specifications. -BCLR can sink up to 3.2 ma and source 80 ua (can drive 2 TTL loads in addition to the 1K ohm load).

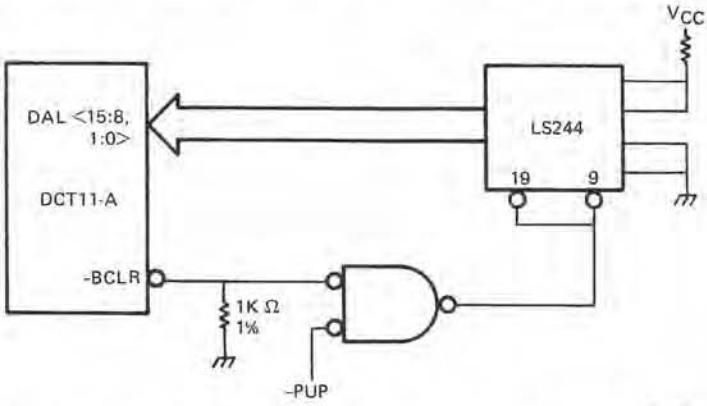


Figure 5-2 Mode Register Loading

5.4 CLOCK

The DCT11-AA clock is generated by an internal clock circuit. This circuit uses as an input one of two sources:

- A crystal
- A TTL oscillator.

5.4.1 Crystal Based Clock

The DCT11-AA oscillator circuit is a quasi-linear wideband amplifier. With reference to Figure 5-3, three components and proper layout are required to use a crystal with the DCT11-AA. The three components are:

- A crystal, with loss resistance (R_s) at various resonancies
- An input capacitor (C_a) connected to XTL0 (pin 23)
- An output capacitor (C_b) connected to XTL1 (pin 22).

A fourth component (caused by stray effects of crystal and layout) is a strong input-output capacitance (C_d) between XTL0 and XTL1. Other stray components, such as high frequency inductance of the connections, have only minor effects at frequencies ($\leq$7.5Mhz) when connection paths are less than two inches.

The capacitors C_a and C_b are needed to adjust the load to the crystal. The inputs XTL0 and XTL1 load the crystal to more than 30pf (which is the nominal load for most crystals) thus pulling it off frequency.

The recommended circuit values for the crystal oscillator in Figure 5-3 are;

Crystal:
- Cut at fundamental (At)
- Load 30pf
- R_s <200ohms ÷ fMHz (fundamental i.e. 27ohms @ 7.5MHz)
- R_s >4000ohms ÷ fMHz (spurious)
- R_s >400ohms ÷ fMHz (harmonic)
- C_o <4pf.

Capacitors:
- Type mica
- Nominal value (+/- 10%)
  - C_a (XTL0) 500pf ÷ fMHz (example 67pf @ 7.5MHz)
  - C_b (XTL1) 200pf ÷ fMHz (example 27pf @ 7.5MHz).

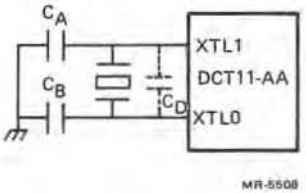


Figure 5-3 Crystal Clock

These specifications guarantee against third harmonic and spurious startups. If such guarantees are not necessary and only a steady oscillation is required then most crystals can be used.

Detailed timing of XTAL is found in Figure A-17 of the Appendix.

5.4.2 TTL Oscillator Based Clock

With reference to Figure 5-4 and 5-5, a TTL signal may be used to drive XTL1 (pin 22) while XTL0 (pin 23) is grounded. The XTL1 TTL signal must satisfy the following criteria:

- Period T > 133ns
- Rise time t_R < 80ns
- Fall time t_F < 80ns
- Low time t_L > 60ns
- High time t_H > 60ns

With reference to Figure 5-6, the XTL1 signal may be gated to stop operation of the DCT11-AA as long as the signal at XTL1 pin meets or exceeds the above criteria. XTL1 may be left high or low indefinitely.

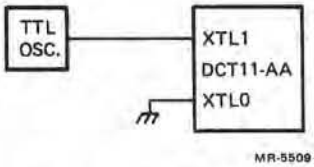


Figure 5-4 TTL Oscillator Clock

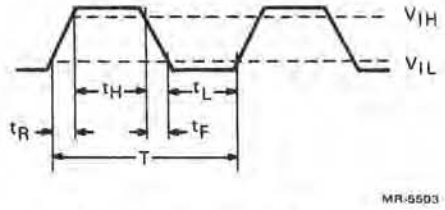


Figure 5-5 TTL Oscillator Waveform

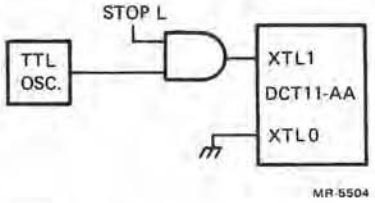


Figure 5-6 Gating XTL1

5.5 ADDRESS LATCH AND DECODE

With reference to Figure 5-7, in 16-bit mode the 16 bits of address can be conveniently latched by a transparent latch (such as an LS373) enabled by Row Address Strobe (-RAS) leading edge.

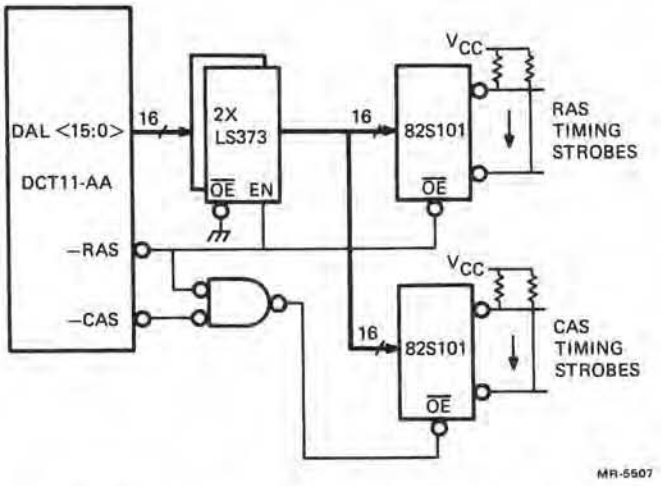


Figure 5-7 16-Bit Address Latch and Decode

With reference to Figure 5-8, in 8-bit mode only the low byte of the address needs to be latched since the high byte remains stable on the Static Address Lines (SAL<15:8>) throughout the whole read or write transaction.

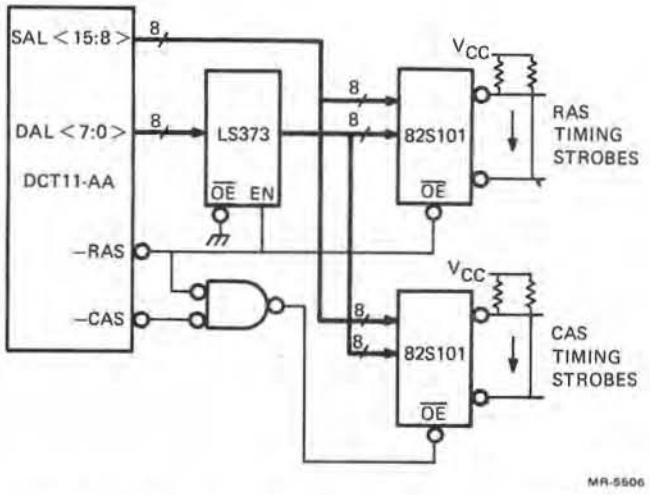


Figure 5-8 8-Bit Address Latch and Decode

Address decoding can be done in a number of traditional ways. In Figures 5-7 and 5-8 PLA's are used to provide direct strobing of several registers. Because the circuit uses a transparent latch, the PLA inputs are stable before -RAS therefore some of the strobes have -RAS timing, whereas others have Column Address Strobe (-CAS) timing. -CAS should be ANDed with -RAS to prevent the enabling of strobes during an ASPI transaction.

5.6 MEMORY SUBSYSTEMS

Two examples of memory subsystems will be described, one using 16-bit mode and the other using 8-bit mode.

5.6.1 16-Bit Mode Memory System

An example of a 16-bit mode dynamic memory system is given in Figure 5-10. The memory map is shown in Figure 5-9. The address is latched in the LS373's (latch) by RAS. The address is then decoded in the LS138 into 8 sectors in the upper half of the memory. The sector 160000 to 167776 maps into the ROM. This is implemented by selecting the ROM (-CE) with the output Y6 and selecting -OE with -CAS (refer to Figure 5-9). The fast variation ROM (2716-1) must be used if running at more than 6.9MHz in order to meet the DCT11-AA specification of t_{RRD} (405ns at 7.5MHz).

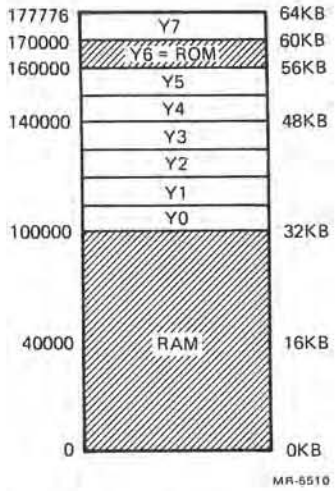


Figure 5-9 16-Bit System Memory Map

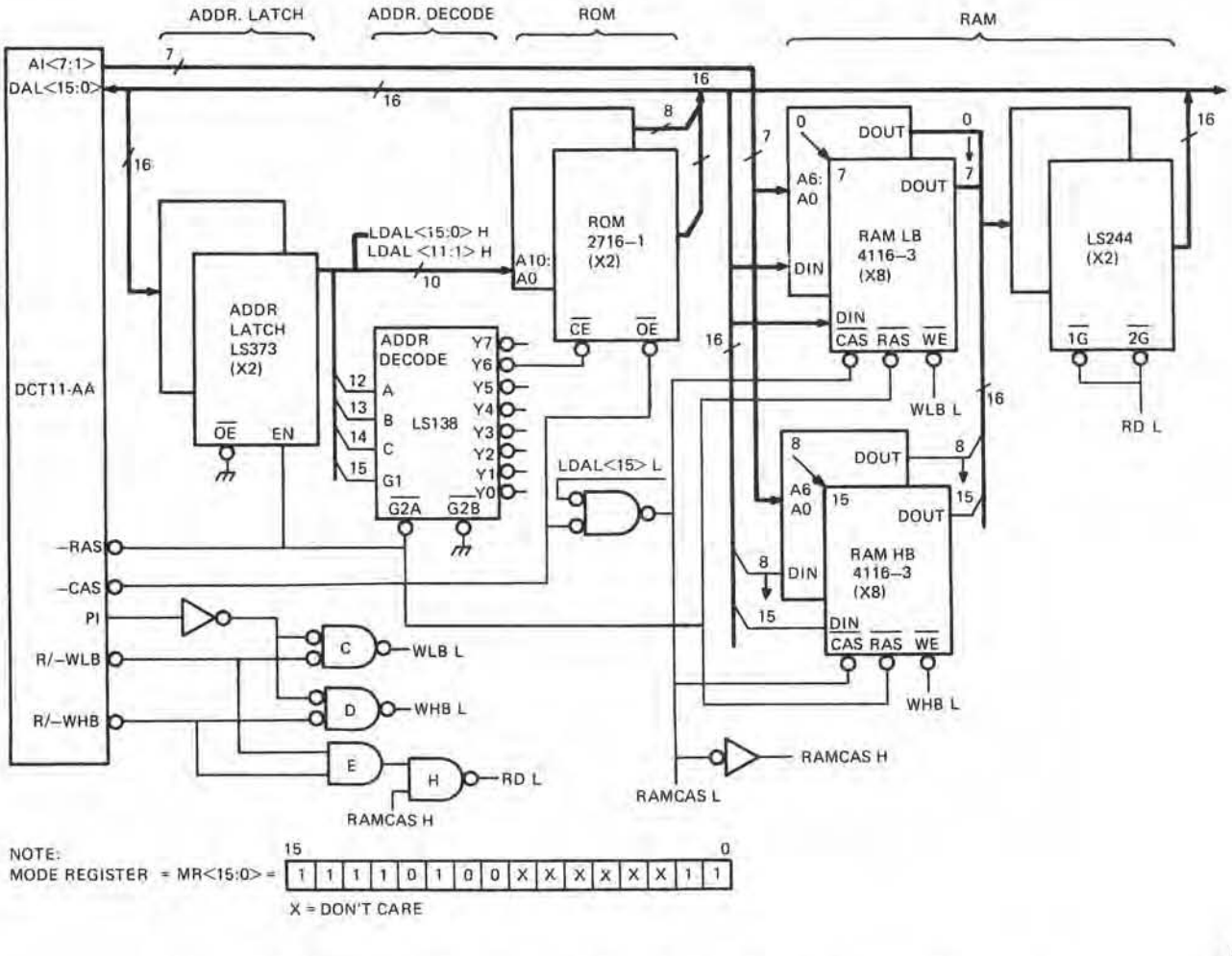


Figure 5-10 16-Bit ROM (4K) & Dynamic RAM (32K) Subsystem

The RAM is made of high byte and low byte sections which have every thing in common except the -WE strobe. The whole RAM is chip selected by controlling -CAS and sending -RAS to all chips at all times. This approach saves power because it leaves the chips in a standby state when not selected. Using -CAS as chip select has the advantage of refreshing a row at every occurrence of -RAS.

Although -CAS drives 160pf (10pf per RAM chip), it does not require buffering. The DCT11-AA output timings are specified for a purely capacative load of 80pf. For loading other than 80pf the following correction factors should be used:

- 80pf < CL < 200pf +0.3ns/pf
- 25pf < CL < 80pf -0.3ns/pf

This results in a delay of -CAS of 80 X .3 = 24ns with respect to the nominal timing specifications. Such a delay still meets the RAM chip specifications for -RAS and -CAS hold time (55ns min for 4116-3). Refer to Figure 5-11.

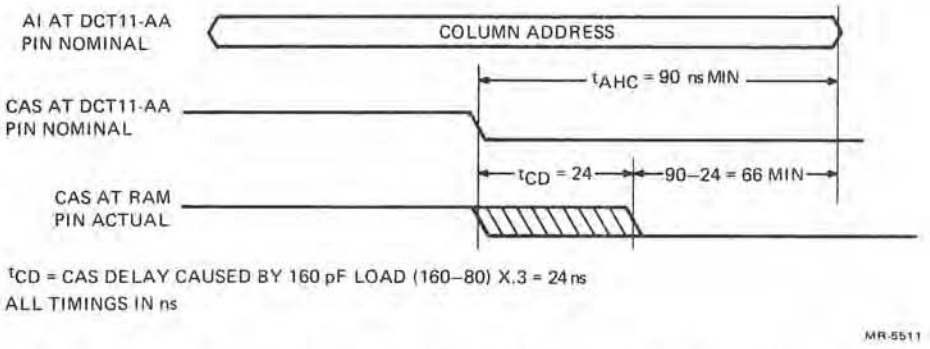


Figure 5-11 Column Address Setup and Hold Time Calculations

The DAL's drive the DIN inputs of the RAM's directly. The DOUT lines cannot be connected directly to the DAL bus, but they must be buffered. This is required because the DCT11-AA does not drive the Data Out on the DAL's soon enough (before -CAS leading edge) to perform an early write on the RAM's. Thus the system only performs a read modify write which would result in contention on the DAL's. The contention would occur between the Data Out driven by the DCT11-AA and the DOUT driven by the RAM chips.

The buffer is enabled only when the DCT11-AA is performing a read from RAM (signal -RD L). With reference to Tables 5-1 and 5-2, read transactions are identified by the signal -RD. "(RAS), (CAS), (R/WLB), and (R/WHB)" are implemented by gates E, G, and H in Figure 5-10.

Table 5-1 Control Signals for Each Transaction

TRANSACTION	-RAS	-CAS	PI	R/-WHB	R/-WLB	SEL0	SEL1
Read	*	*	*	X			
Fetch	*	*	*	X		1	
Write	*	*	*	-	*		
Refresh	*					2	
Iack	*						*
DMA	*	*	*	3S	3S	*	*
ASPI		*	*				
Busnop							

* Signal asserted during the transaction
1 Static modes and dynamic 64K
2 Dynamic modes 4K/16K
X Signal asserted during 8-bit mode only
\- Signal asserted during 16-bit mode only
3S Three-state

Table 5-2 Data Bus for Each Transaction

TRANSACTION	DAL LOW BYTE	DAL HIGH BYTE	AI
Read		X	
Fetch		X	
Write	*	X	
Refresh	*	*	*
Iack		*	1
DMA	3S	3S	3S
ASPI			
Busnop	*	*	1

X Lines driven after address portion of transaction 8-bit mode
* Lines driven after address portion of transaction
1 Dynamic modes only
3S Three-state

5.6.2 8-Bit Mode Memory System

With reference to Figure 5-12, since the data bus is 8 bits wide and the memory organization is different, an 8-bit minimum system can be implemented using only half as many memory chips as a 16-bit minimum system. The memory map implemented is shown in Figure 5-13 and the circuit schematic in Figure 5-14. The signals WT L and RD L are easily generated in this configuration.

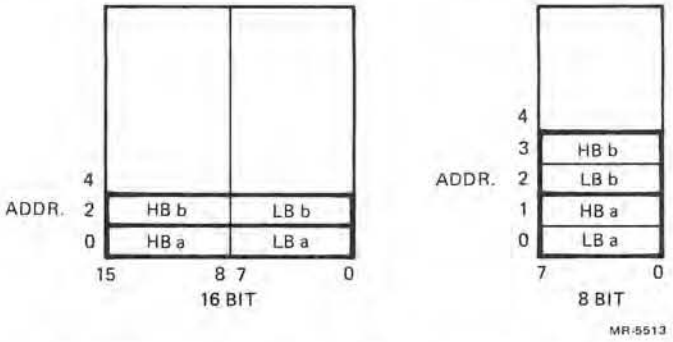


Figure 5-12 16-Bit - 8-Bit Memory Organization

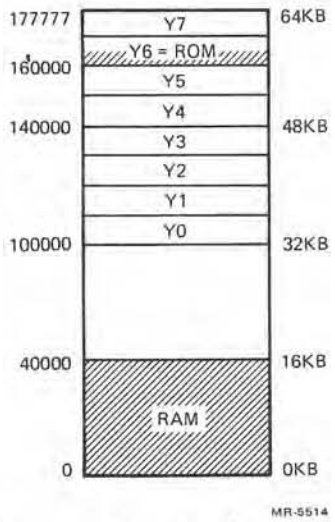


Figure 5-13 8-Bit System Memory Map

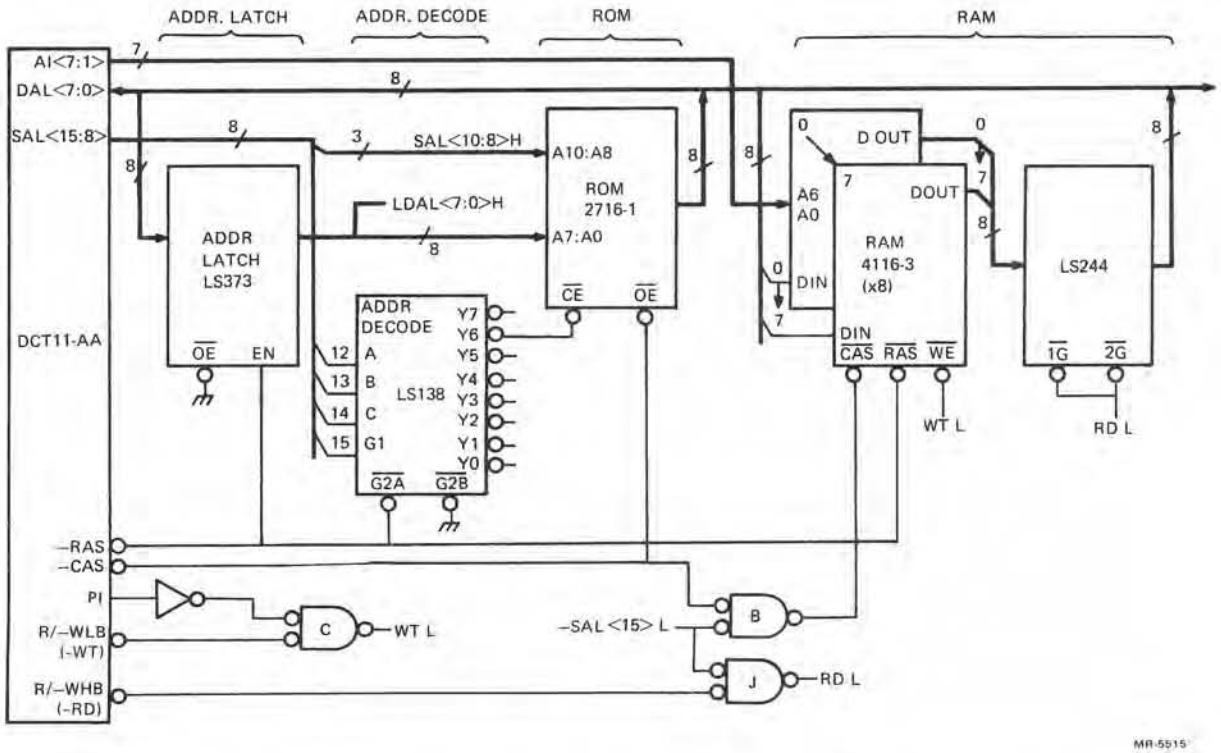


Figure 5-14 8-Bit ROM (2K) & Dynamic RAM (16K) Subsystem

An alternative design for the RAM implements a common I/O via an early write. This is done by latching the column address and delaying -CAS until the data out is available on the DCT11-AA pins (Refer to Figure 5-15.). Figure 5-16 is the timing diagram for an early write.

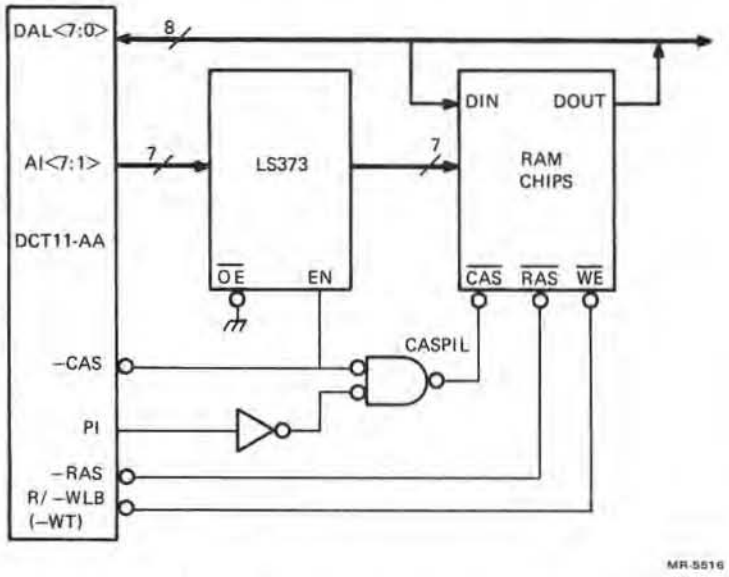


Figure 5-15 8-Bit Mode Common I/O RAM

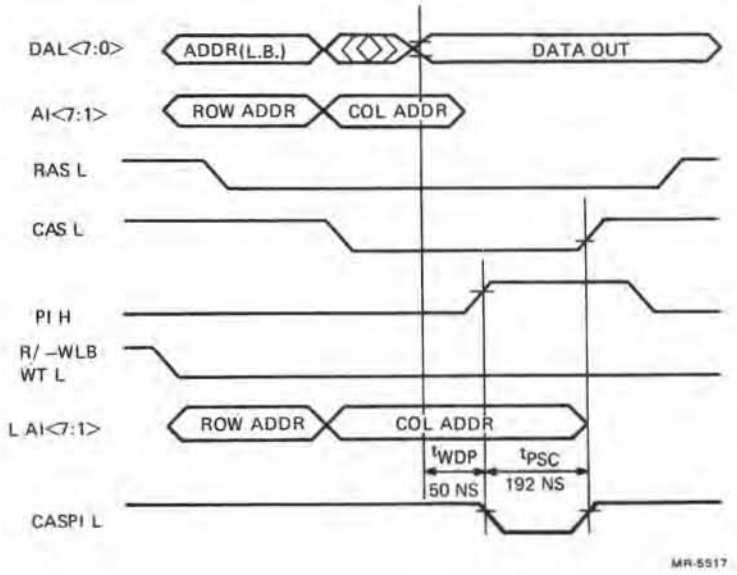


Figure 5-16 Early Write Timing Diagram

5.7 INTERRUPTS

The examples of interrupts will cover the following areas:

- Posting interrupts
- Decoding Iack information
- External vectors
- Using a priority encoder chip
- Direct CP (coded priority) encoding

5.7.1 Posting Interrupts

With reference to Figure 5-17, to avoid propagation of metastable states, it is necessary to drive stable signals as interrupt requests. A latch can be used for this purpose. The delay between -CAS and PI (t_{csp} = 105ns @ 7.5MHz) is long enough to settle any metastable states on the output of the flip flop.

5.7.2 Decoding Iack Information

Figure 5-18 is an example of how to decode Iack information for 15 CP devices. An LS138 can be used instead of a LS154 when fewer interrupts are needed. The LS138 can also be used if care is taken to pick CP codes such that one of the CP lines is always low for all CP codes (3 line encoding).

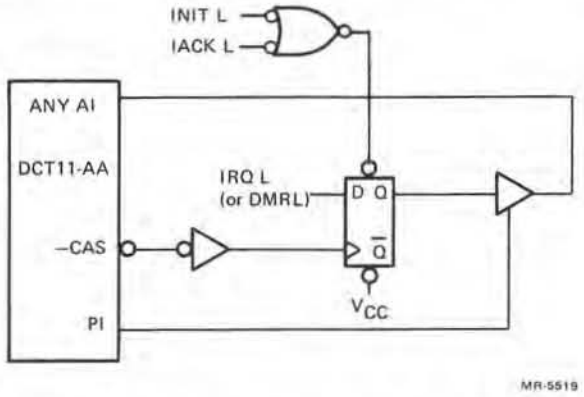


Figure 5-17 General Interrupt & DMA Request Circuit

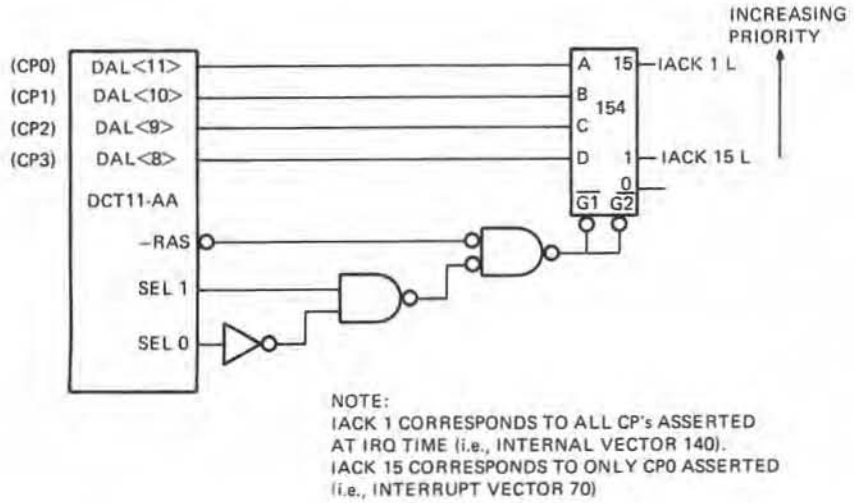
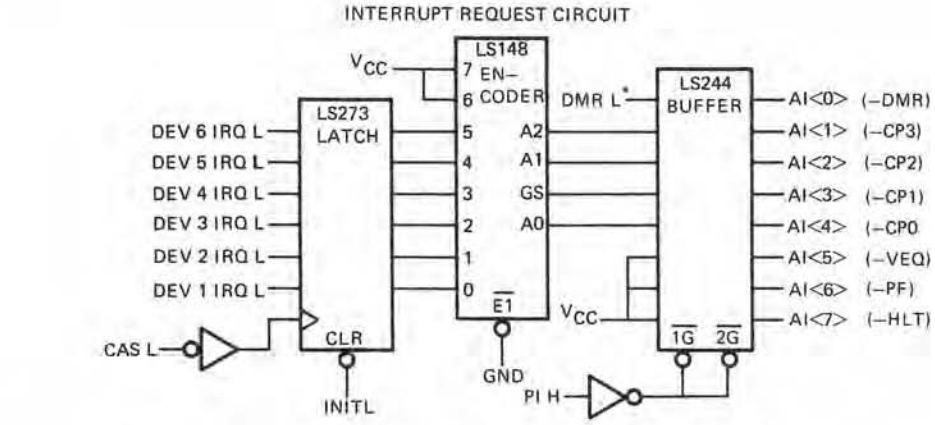
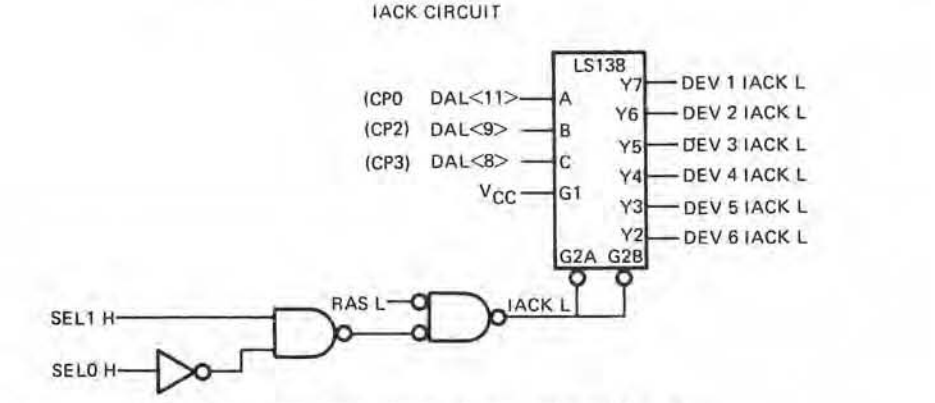


Figure 5-18 Decoding Iack Information for 16 CP Devices

Figure 5-19 is an example of a complete interrupt system (interrupt request and interrupt acknowledge) for six CP devices.



LS138	A		B	C							Y2	Y3	Y4	Y5	Y6	Y7		
LS148	A0	GS	A1	A2	0	1	2	3	4	5								
DURING IACK	DAL11	DAL10	DAL9	DAL8														
DURING IRQ	–CP0	–CP1	–CP2	–CP3													INTERNAL VECTORS	
	AI4	AI3	AI2	AI1														
	L	L	H	L	H	H	H	H	H	L	L	H	H	H	H	H	DEV6	100
	L	L	H	L	H	H	H	H	L	H	H	L	H	H	H	H	DEV5	104
	H	H	H	H	H	H	H	L	H	H	H	H	L	H	H	H	DEV4	120
	H	L	L	H	H	H	L	H	H	H	H	H	H	L	H	H	DEV3	124
	L	L	H	H	H	L	H	H	H	H	H	H	H	H	L	H	DEV2	60
	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	L	DEV1	64



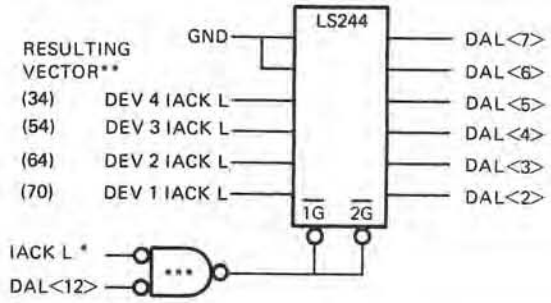
*SIGNAL IS GENERATED BY THE USER AND MUST BE STABLE AT PI TIME.

MR-5520

Figure 5-19 Interrupt System

5.7.3 External Vectors

If -VEC (AI<5>) is asserted (low), during the interrupt request, DCT11-AA obtains the vector on DAL<7:2> from the device during the Iack transaction. Figure 5-20 gives an example of such a circuit.



*SIGNAL IS THE SAME AS IN FIGURE 5.18

**CAN BE HARDWIRED IF A SINGLE EXTERNAL VECTOR INTERRUPT IS USED

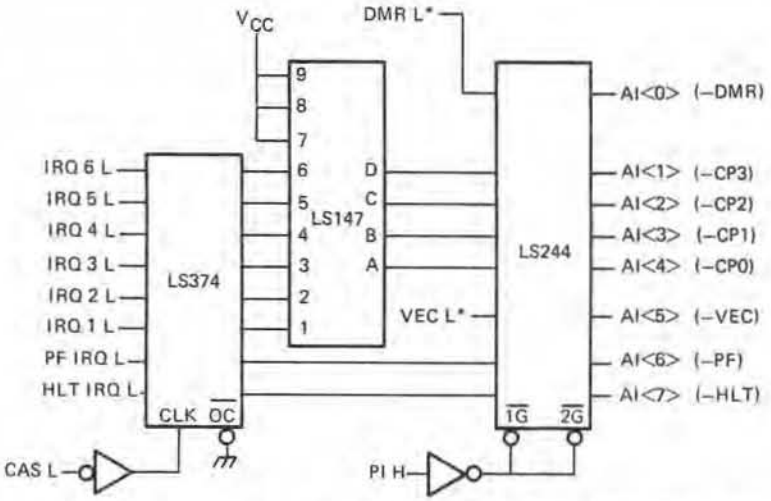
***USED ONLY WITH BOTH EXTERNAL AND INTERNAL VECTOR
(IF EXTERNAL ONLY, USE IACK L)

MR-5521

Figure 5-20 Driving External Vector During Iack

5.7.4 Using a Priority Encoder Chip

With reference to Figure 5-21, up to 15 CP devices may be used with this scheme if two LS148's are chained together. The chaining of two LS148's (encoder) produces a 16 to 4 priority encoder (instead of using LS147).



*SIGNALS ARE GENERATED BY THE USER AND MUST BE STABLE AT PI TIME (L.E.).

–CP<3> (AI<1>)	–CP<2> (AI<2>)	–CP<1> (AI<3>)	–CP<0> (AI<4>)	PRIORITY LEVEL	INTERNAL VECTOR ADDRESS	DEVICES IMPLEMENTED
X	X	X	X	8	–	HLT IRQ
X	X	X	X	8	24	PF IRQ
L	L	L	L	7	140	
L	L	L	H	7	144	
L	L	H	L	7	150	
L	L	H	H	7	154	
L	H	L	L	6	100	
L	H	L	H	6	104	
L	H	H	L	6	110	
L	H	H	H	6	114	
H	L	L	L	5	120	
H	L	L	H	5	124	IRQ6
H	L	H	L	5	130	IRQ5
H	L	H	H	5	134	IRQ4
H	H	L	L	4	60	IRQ3
H	H	L	H	4	64	IRQ2
H	H	H	L	4	70	IRQ1
H	H	H	H	NO ACTION		

MR-5522

Figure 5-21 Interrupt Request Circuit (Priority Encoder)

In order to handle more than 15 CP interrupts, each of the 15 prioritized lines can be made up of a daisy chain of several devices. All devices on the same daisy chain have the same CP code, but they identify from each other by different external vectors during the Iack transaction.

5.7.5 Direct CP Encoding

Direct CP encoding (Figure 5-22) can only be accomplished when there are 4 or less CP devices (one device per CP line). The highest priority device D3 will connect directly to CP<3> and the lowest to CP<0>. If internal vectors are used, locations 140 thru 154 and 100 thru 114 must be loaded with the vector address relative to D3. Locations 120 thru 134 must be loaded with the vector address relative to D2. Locations 60 and 64 must be loaded with vector address relative to D1 and 70 to D0. AI<7:5,0> do not need to be driven high because the DCT11-AA has internal pull-ups on those lines at PI time.

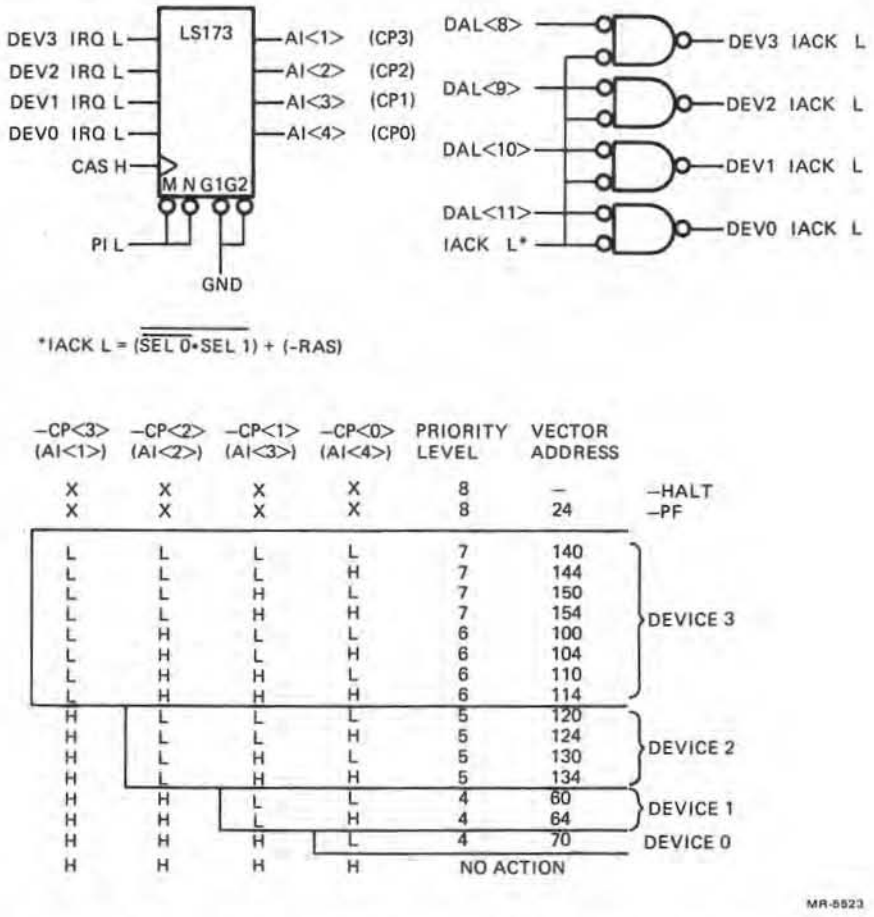


Figure 5-22 Direct CP Encoding Interrupt System

5.8 DMA

During DMA, the DCT11-AA provides -RAS, -CAS, PI, and COUT signals. The external circuitry has the responsiblity for controlling the R/-WHB and R/-WLB lines, providing the address, and supplying or accepting data. The DCT11-AA can easily be connected to single channel or multiple channel DMA circuitry.

During DMA transfers, system circuitry goes through the following sequence:

- A DMA request (DMR) to the DCT11-AA is made by driving AI<0> low during PI
- The request is latched into the DCT11-AA during PI and shortly thereafter a DMA grant is issued.

The maximum time from the request origination until the grant is issued, is a function of the DCT11-AA mode. This time is specified in Table 5-3. When the grant is issued the DCT11-AA takes the following actions:

- SEL0 and SEL1 go to a high thus informing the system that the grant has been issued
- -RAS, -CAS, PI, and COUT are driven with the timings specified in the DMA transaction timing diagram (Appendix Figure A-13)
- DAL's are three-stated
- AI<7:0>, R/-WHB, and R/-WLB have low current pull-ups.

Table 5-3 DMR Maximum Latencies

16-Bit Mode

Static			Dynamic		
ucycle	+ Phase	+ uS	ucycle	+ Phase	+ uS
8	1	.19	8	2	.19
(@ 7.5Mhz = 3.523uS)			(@ 7.5Mhz = 3.657uS)		

8-Bit Mode

Static			Dynamic		
ucycle	+ Phase	+ uS	ucycle	+ Phase	+ uS
10	1	.19	10	2	.19
(@ 7.5Mhz = 4.323uS)			(@ 7.5Mhz = 4.457uS)		

When the grant is issued, external circuitry must drive the R/-WHB and R/-WLB lines, and initially drive the DAL's with the address. In dynamic memory systems the address must be multiplexed on AI<7:0> so that the memory chips are provided with row and column addresses at the appropriate times. Later in the transaction the data transfer on the DAL's takes place in a direction controlled by the state of the R/-WHB and R/-WLB lines. The DCT11-AA continues issuing grants for DMA transactions until DMR L is no longer asserted low on AI<0>, during PI. When this happens the present sequence of DMA transactions finishes and the DCT11-AA resumes normal operation.

5.8.1 Single Channel DMA Controller (16-Bit Mode)

With reference to Figure 5-23, this section describes a single channel DMA controller for use with dynamic or static memory systems.

5.8.1.1 Address Latches (Single Channel DMA Controller) -- Address latches can be shared with the rest of the system. In a static memory system, if address latches are not necessary for the rest of the system the four chips (E3 thru E6) may be eliminated. In a dynamic memory system, if address latches are not necessary for the rest of the system the two chips (E3 and E4) may be replaced by one latch that will save the appropriate AI bits for the -CAS strobe.

5.8.1.2 Pulse Mode Clock (Single Channel DMA Controller) -- With reference to Figure 5-23, pulse mode clock is used in this circuit. If this is not possible a delay line or RC combination can be used for the generation of an edge between -RAS assertion (low) and -CAS assertion (low). This edge switches the AI multiplexer from row address to column address in dynamic memory systems. If a static memory system is being used this switching is not needed. Pulse mode clock also produces a convenient edge in the middle of PI that is useful when writing to peripherals.

5.8.1.3 Address Decode Structures -- A PLA or any other address decode structure provides the following register selects:

S-A L	Select address counter in DC006's (DEC DMA chip)
S-C L	Select word counter in DC006's
DMACR L	DMACR (DMA Control Register) is an optional register which may specify DMA direction and make DMA requests under software control.

5.8.1.4 Operation Sequence (Single Channel DMA Controller) -- The DCT11-AA software loads the DC006's with the two's complement of the word count and then with the bus address. Following the loading, the peripheral is signaled that the DMA set up is complete.

The peripheral device makes a DMA request by asserting the upper DMA REQ H which in turn causes the assertion of DMR L. The DCT11-AA issues a DMG and drives -RAS, -CAS, PI, and COUT. The peripheral drives R/-WHB and R/-WLB and negates the signal upper DMA REQ H. The signal ROWAD H is already asserted high when the DMG cycle starts. ROWAD H and DMG H send the output of AND gate E8A high which asserts the read input to the DC006's. The DC006's drive the address onto the DAL's where it is input by the address latches which in turn drive the AI multiplexer inputs. During this period the AI multiplexer is pointing to the row address inputs.

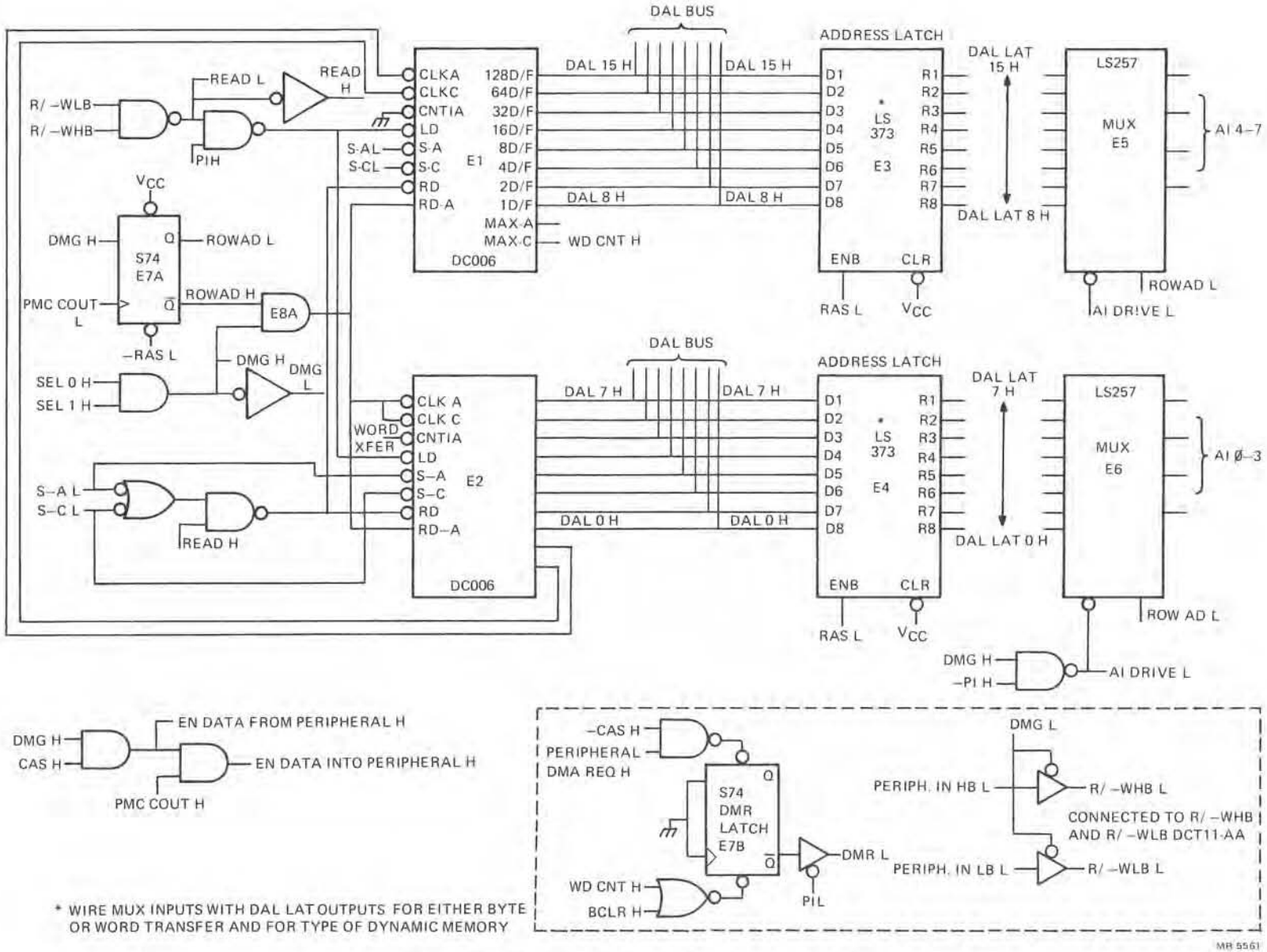


Figure 5-23 Single Channel DMA

Between -RAS assertion (low) and -CAS assertion (low) the trailing edge of PMC COUT clocks latch E7A driving ROWAD H to a low state. When ROWAD L goes high the AI multiplexer inputs switch to the column address and the output of AND gate E8A goes low which effects the DC006's in two ways:

- The DC006's inputs go three-state so they stop driving the address onto the DAL's.
- The word count and bus address registers of the low byte DC006 are incremented. The word count increments by one if the CNT1A input to E2 (WORD XFER) is low and by two if it is high.

If the count in E2 reaches a maximum, the next count edge will also increment E1. When the word count overflows, WDCNT H is asserted, which sets the DMR latch E7B and no further DMA requests are made.

5.8.2 Software DMA Requests

A small modification to the hardware permits software DMA requests and software specification of the transfer direction. Substitution of the schematic in Figure 5-24 for the enclosed section of Figure 5-23 results in the necessary hardware modification.

The transfer direction is specified by writing to bit 0 of the DMACR (DMA control register). Writing a "1" to bit 0 of the DMACR specifies DMA transfers from memory to the peripheral. Writing a "0" to bit 0 of the DMACR specifies DMA transfers from the peripheral to memory. Writing a "1" to bit 1 of the DMACR makes an immediate DMA request. The request will be latched into the DCT11-AA during the same transaction that wrote the DMACR.

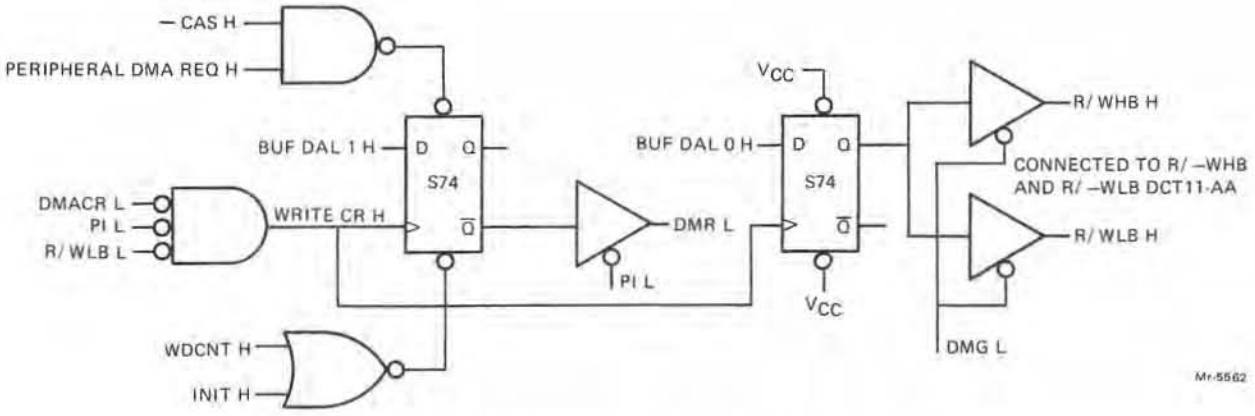


Figure 5-24 Software DMR Control

5.9 WORKING WITH PERIPHERAL CHIPS

This discussion does not encompass all the peripheral chips that will work with the DCT11-AA, but does cover a few selected ones. Almost all peripheral chips will work with the DCT11-AA. The following chips will be discussed:

- 8155 RAM, three ports, and timer
- 2651 PUSART (programmable serial line unit)
- DC003 Interrupt Logic.

5.9.1 8155 RAM and Three Ports and Timer

With reference to Figure 5-25, this example uses 8-bit mode, delayed read/write mode. If normal read/write is desired, it is necessary to gate the read/write lines with -CAS. Chip enable and IO/M control is accomplished by static addresses which remain valid throughout the transaction.

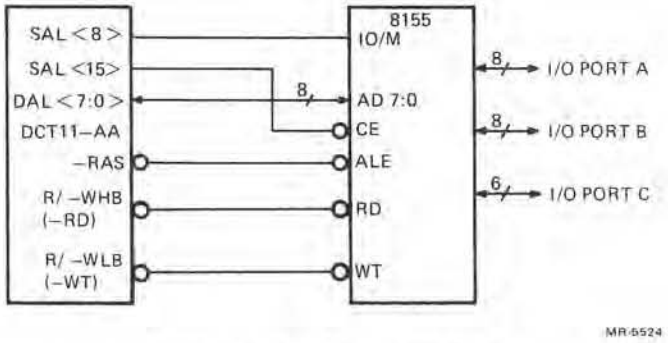


Figure 5-25 8155

5.9.2 2651 PUSART

With reference to Figure 5-26, there are two facts which must be understood when interfacing the 2651 PUSART to the DCT11-AA. Compatibility depends on these two facts:

- Every DCT11-AA write is preceded by a read from the same location (except stack operations, traps, interrupts, and subroutines).
- The 2651's receive data buffer and transmit data buffer have the same address inside the chip. The buffers are selected by the R/W input.

The involuntary read from the receive buffer clears the receive ready pin, and may result in resetting the receiver interrupt. To avoid this it is necessary to assign separate addresses to the receive and transmit buffers and disable read transactions from the transmit buffer.

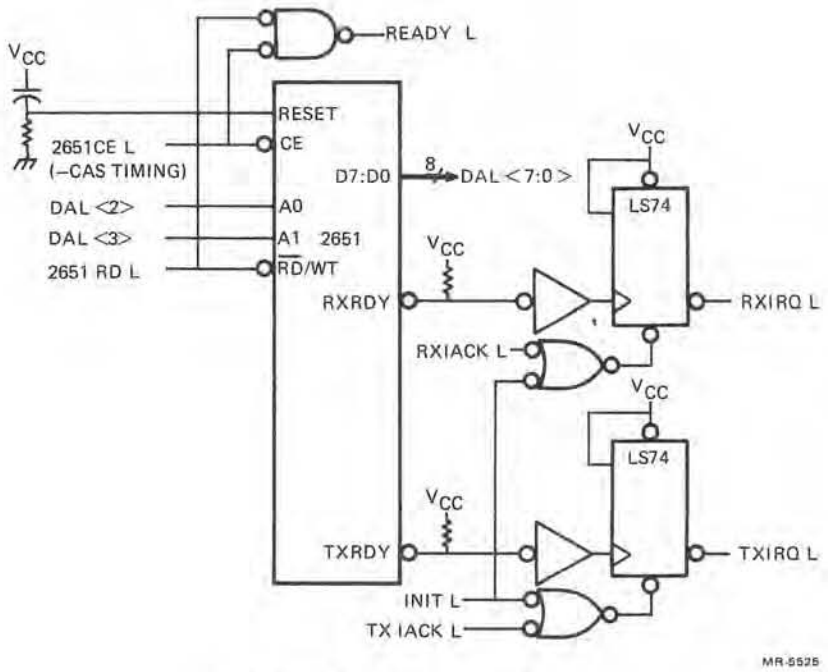


Figure 5-26 2651

For the same reason, the 2651 mode registers must be accessed by a proper sequence of reads and writes. An example in 16-bit mode:

- To write mode register 1: disable transmit and receive (control register <- 5), read mode register, write mode register.
- To write mode register 2: disable transmit and receive (control register <- 5), write mode register.

In 8-bit mode, the same operation takes place but byte instructions must be used. If word instructions are used, then the same 2651 register is accessed twice, thus incrementing the mode register pointer.

The 2651's access time is 250ns, a READY slip should be used when the DCT11-AA is running at frequencies greater than 7MHz.

If the DCT11-AA is running at a frequency greater than 6MHz, D0:D7 require buffering to the DAL's. The buffering is because the 2651 turn off time is 150ns maximum with a 100pf load. t_{CDE} for the DCT11-AA at 6MHz is 148ns maximum.

5.9.3 DC003 Interrupt Logic

With reference to Figures 5-27 and 5-28, the interrupt chip is an 18-pin dual-in-line-package device that provides the circuits to handle interrupts. The chip can be used in any externally vectored interrupt scheme and the system does not have to be daisy-chained. The device is used in peripheral interfaces to provide two interrupt channels labeled "A" and "B" with the A section at a higher priority than the B section. DC003's may be daisy-chained at any priority level.

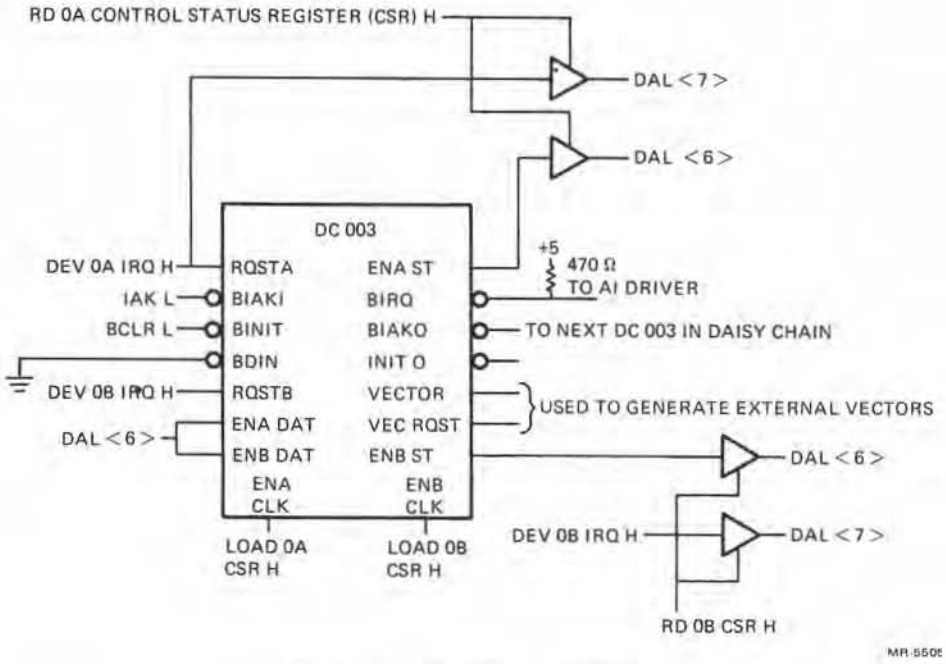


Figure 5-27 DC003

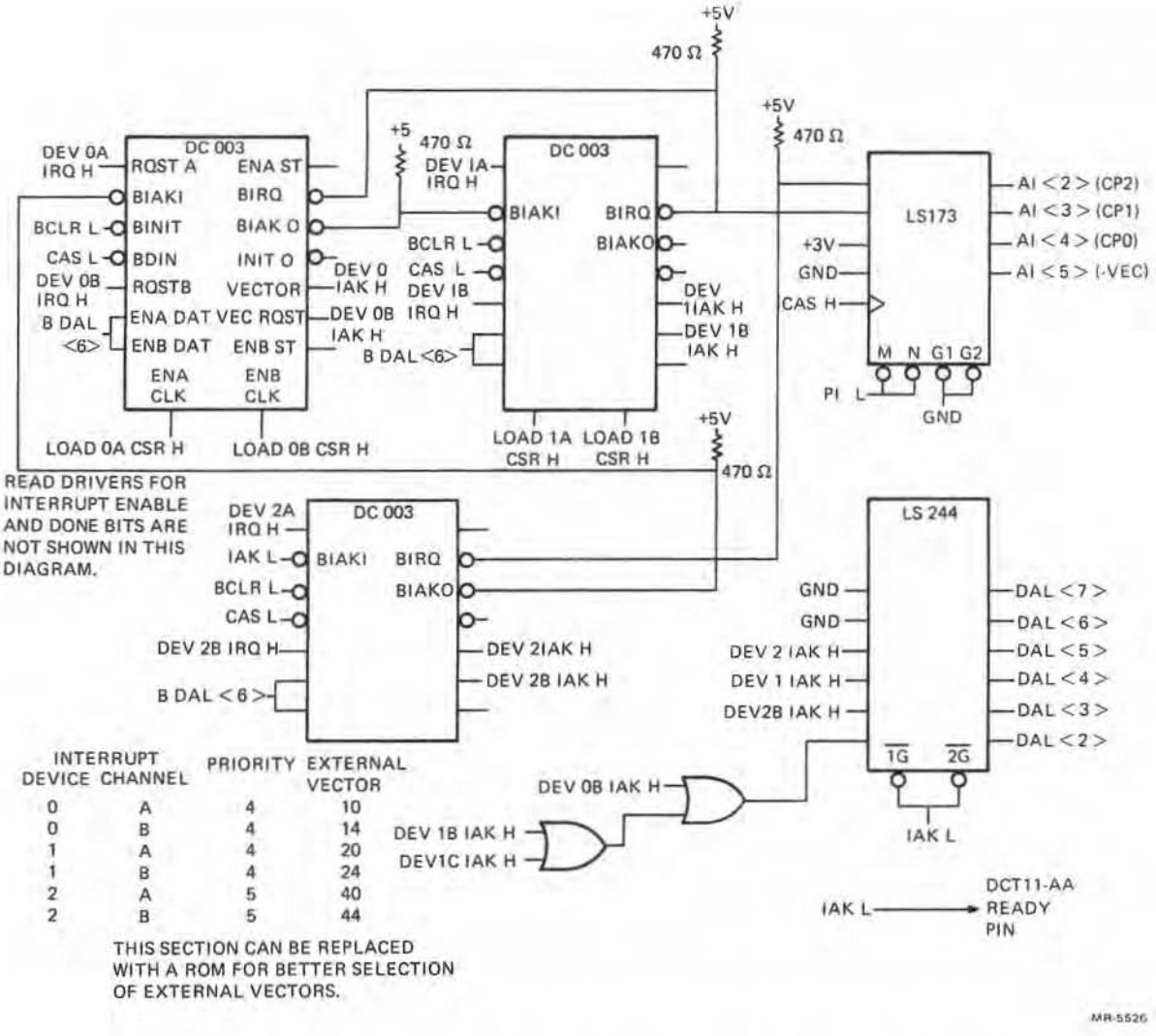


Figure 5-28 DC003 at Different Priority Levels

CHAPTER 6

ADDRESSING MODES AND INSTRUCTION SET

6.1 INTRODUCTION

This chapter is divided into six major categories:

- Single Operand Addressing -- One part of the instruction word specifies the registers; the remaining part provides information for locating the operand.
- Double Operand Addressing -- Part of the instruction word specifies the registers; the remaining parts provide information for locating two operands.
- Direct Addressing -- The operand is the content of the selected register.
- Deferred (Indirect) Addressing -- The contents of the selected register is the address of the operand.
- Use of the PC as a General Purpose Register -- The PC is unique from other general-purpose registers in one important respect. Whenever the processor retrieves an instruction, it automatically advances the PC by 2. By combining this automatic advancement of the PC with four of the basic addressing modes, we produce the four special PC modes -- immediate, absolute, relative, and relative deferred.
- Use of the Stack Pointer as a General Purpose Register -- Can be used for stack operations.

6.2 ADDRESSING MODES

Data stored in memory must be accessed and manipulated. Data handling is specified by a DCT11-AA instruction (MOV, ADD, etc.), which usually indicates:

- The function (operation code).
- A general-purpose register is to be used when locating the source operand and/or a general-purpose register to be used when locating the destination operand.
- An addressing mode (to specify how the selected register(s) is/are to be used).

A large portion of the data handled by a computer is usually structured (in character strings, arrays, lists, etc.). DCT11-AA addressing modes provide for efficient and flexible handling of structured data.

The general registers may be used with an instruction in any of the following ways.

- As accumulators. The data to be manipulated resides within the register.
- As pointers. The contents of the register is the address of the operand, rather than the operand itself.
- As pointers which automatically step through memory locations. Automatically stepping forward through consecutive locations is known as autoincrement addressing; automatically stepping backwards is known as autodecrement addressing. These modes are particularly useful for processing tabular or array data.
- As index registers. In this instance, the contents of the register and the word following the instruction are summed to produce the address of the operand. This allows easy access to variable entries in a list.

An important DCT11-AA feature, which should be considered with the addressing modes, is the register arrangement.

- Six general-purpose registers (R0--R5)
- A hardware Stack Pointer (SP) register (R6)
- A Program Counter (PC) register (R7)

Registers R0 through R5 are not dedicated to any specific function; their use is determined by the instruction that is decoded.

- They can be used for operand storage. For example, contents of two registers can be added and stored in another register.
- They can contain the address of an operand or serve as pointers to the address of an operand.
- They can be used for the autoincrement or autodecrement features.
- They can be used as index registers for convenient data and program access.

The DCT11-AA also has instruction addressing mode combinations that facilitate temporary data storage structures. This can be used for convenient handling of data that must be accessed frequently. This is known as stack manipulation. The register that keeps track of stack manipulation is known as the stack pointer (SP). Any register can be used as a "stack pointer" under program control; however, certain instructions associated with subroutine linkage and interrupt service automatically use register R6 as a "hardware stack pointer". For this reason, R6 is frequently referred to as the "SP".

- The stack pointer (SP) keeps track of the latest entry on the stack.
- The stack pointer moves down as items are added to the stack and moves up as items are removed. Therefore, it always points to the top of the stack.
- The hardware stack is used during trap or interrupt handling to store information allowing the processor to return to the main program.

Register R7 is used by the processor as its program counter (PC). It is recommended that R7 not be used as a stack pointer or accumulator. Whenever an instruction is fetched from memory, the program counter is automatically incremented by two to point to the next instruction word.

6.2.1 Single Operand Addressing

The instruction format for all single operand instructions (such as clear, increment, test) is:

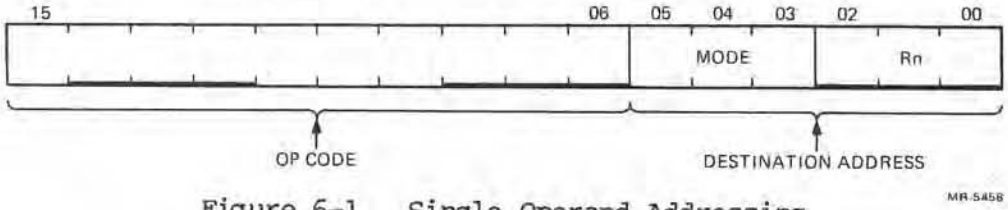


Figure 6-1 Single Operand Addressing

Bits 15 through 6 specify the operation code that defines the type of instruction to be executed.

Bits five through zero form a six-bit field called the destination address field. This consists of two subfields.

- Bits zero through two specify which of the eight general-purpose registers is to be referenced by this instruction word.
- Bits three through five specify how the selected register will be used (address mode). Bit three is set to indicate deferred (indirect) addressing.

6.2.2 Double Operand Addressing

Operations which imply two operands (such as add, subtract, move, and compare) are handled by instructions that specify two addresses. The first operand is called the source operand; the second is called the destination operand. Bit assignments in the source and destination address fields may specify different modes and different registers. The instruction format for the double operand instruction is:

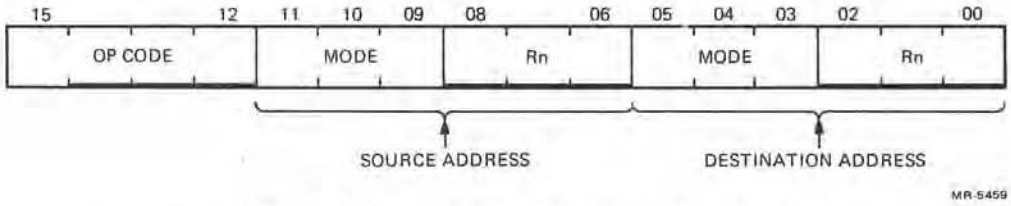


Figure 6-2 Double Operand Addressing

The source address field is used to select the source operand, the first operand. The destination is used similarly, and locates the second operand and the result. For example, the instruction ADD A, B adds the contents (source operand) of location A to the contents (destination operand) of location B. After execution B will contain the result of the addition and the contents of A will be unchanged.

Examples in this paragraph and chapter use the following sample DCT11-AA instructions. A complete listing of the DCT11-AA instructions is located in Paragraph 6.3.

Mnemonic	Description	Octal Code
CLR	Clear (zero the specified destination)	0050DD
CLRB	Clear byte (zero the byte in the specified destination)	1050DD
INC	Increment (add one to contents of destination)	0052DD
INCB	Increment byte (add one to the contents of destination byte)	1052DD
COM	Complement (replace the contents of the destination by its logical complement; each zero bit is set and each one bit is cleared)	0051DD
COMB	Complement byte (replace the contents of the destination byte by its logical complement; each 0 bit is set and each 1 bit is cleared).	1051DD
ADD	Add (add source operand to destination operand and store the result at destination address)	06SSDD

N = number of a general register

DD = destination field (six bits)

SS = source field (six bits)

( ) = contents of

6.2.3 Direct Addressing

The following table summarizes the four basic modes used with direct addressing.

DIRECT MODES

Mode	Name	Assembler Syntax	Function
0	Register	Rn	Register contains operand

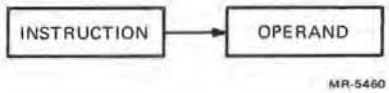


Figure 6-3 Mode 0 Register

2 Autoincrement (Rn)+ Register is used as a pointer to sequential data then incremented

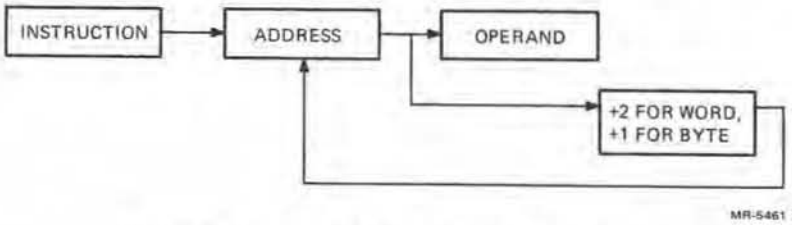


Figure 6-4 Mode 2 Autoincrement

4 Autodecrement -(Rn) Register is decremented and then used as a pointer.

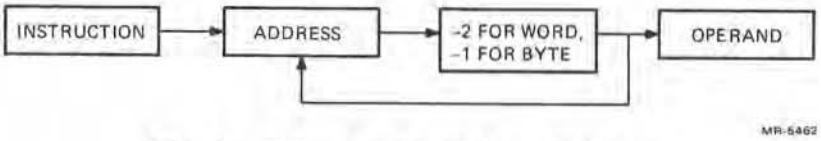


Figure 6-5 Mode 4 Autodecrement

6 Index X(Rn) Value X is added to (Rn) to produce address of operand. Neither X nor (Rn) is modified.

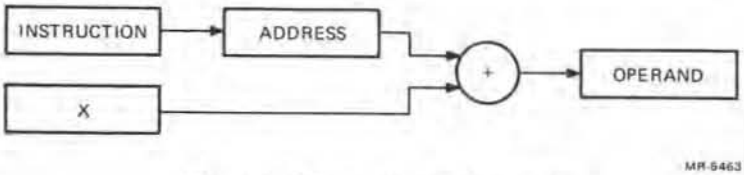


Figure 6-6 Mode 6 Index

6.2.3.1 Register Mode -- With register mode any of the general registers may be used as simple accumulators and the operand is contained in the selected register. Since they are hardware registers, within the processor, the general registers operate at high speeds and provide speed advantages when used for operating on frequently-accessed variables. The assembler interprets and assembles instructions of the form OPR Rn as register mode operations. Rn represents a general register name or number and OPR is used to represent a general instruction mnemonic. Assembler syntax requires that a general register be defined as follows.

```
R0 = %0 (% sign indicates register definition)
R1 = %1
R2 = %2, etc.
```

Registers are typically referred to by name as R0, R1, R2, R3, R4, R5, R6, and R7. However, R6 and R7 are also referred to as SP and PC, respectively.

Register Mode Examples
(all numbers in octal)

	Symbolic	Octal Code	Instruction Name
1.	INC R3	005203	Increment

Operation: Add one to the contents of general purpose register R3.

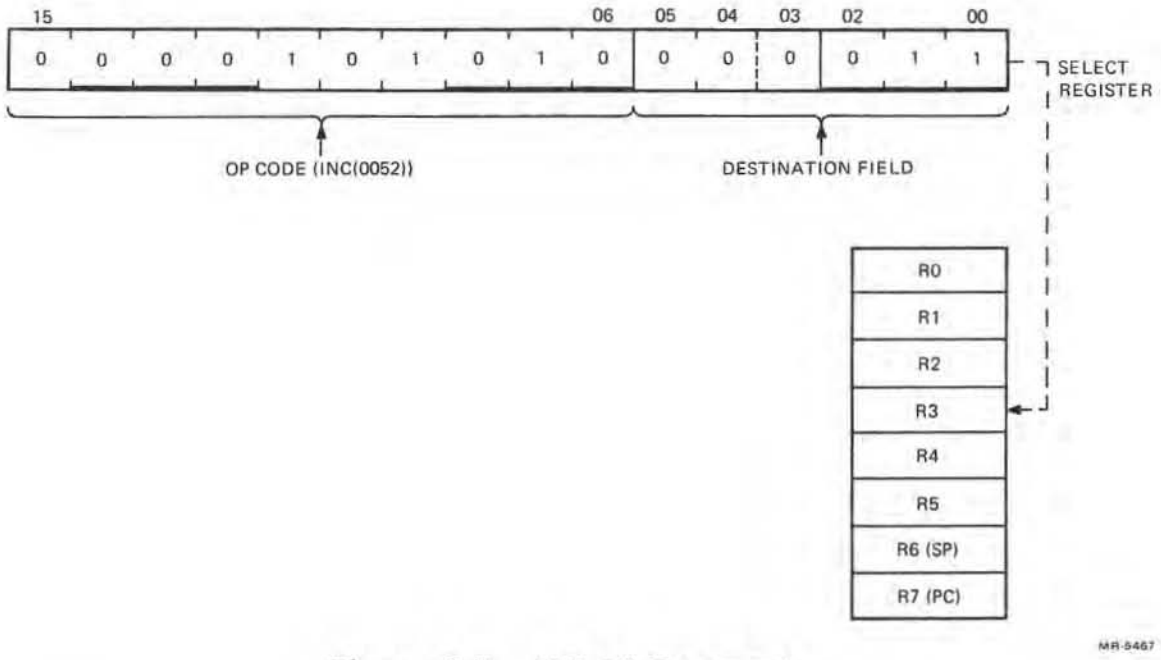


Figure 6-7 INC R3 Increment

2.	ADD R2, R4	060204	Add

Operation: Add the contents of R2 to the contents of R4.

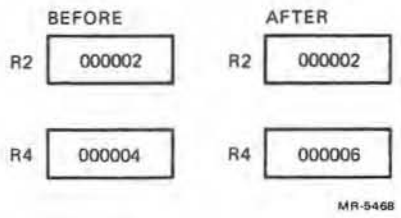


Figure 6-8 ADD R2, R4 Add

3. COMB R4 105104 Complement Byte

Operation: One's complement bits 0--7 (byte) in R4. (When general registers are used, byte instructions only operate on bits 0--7; i.e., byte 0 of the register.)

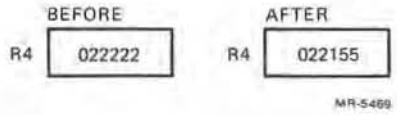


Figure 6-9 COMB R4 Complement Byte

6.2.3.2 Autoincrement Mode (OPR (Rn)+) -- This mode (mode 2) provides for automatic stepping of a pointer through sequential elements of a table of operands. It assumes the contents of the selected general purpose register to be the address of the operand. Contents of registers are stepped (by one for bytes, by two for words, always by two for R6 and R7) to address the next sequential location. The autoincrement mode is especially useful for array processing and stack processing. It will access an element of a table and then step the pointer to address the next operand in the table. Although most useful for table handling, this mode is completely general and may be used for a variety of purposes.

OPR (Rn)+

Autoincrement Mode Examples

Symbolic	Octal Code	Instruction Name
1. CLR (R5)+	005025	Clear

Operation: Use contents of R5 as the address of the operand. Clear selected operand and then increment the contents of R5 by two.

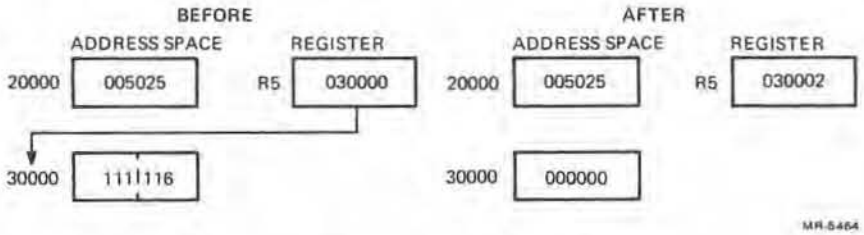


Figure 6-10 CLR (R5)+ Clear

2. CLRB (R5)+ 105025 Clear Byte

Operation: Use contents of R5 as the address of the operand. Clear selected byte operand and then increment the contents of R5 by one.

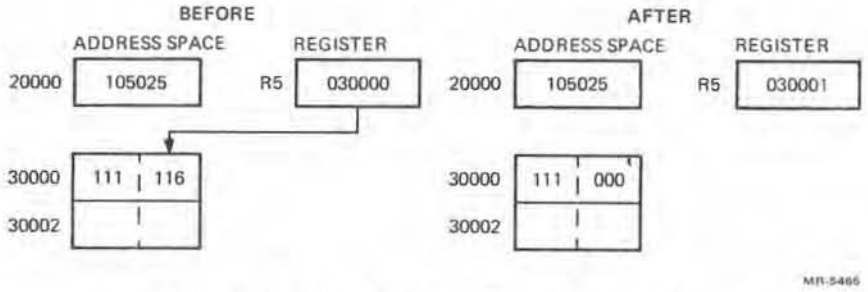


Figure 6-11 CLRB (R5)+ Clear Byte

3. ADD(R2)+,R4 062204 Add

Operation: The contents of R2 are used as the address of the operand which is added to the contents of R4. R2 is then incremented by two.

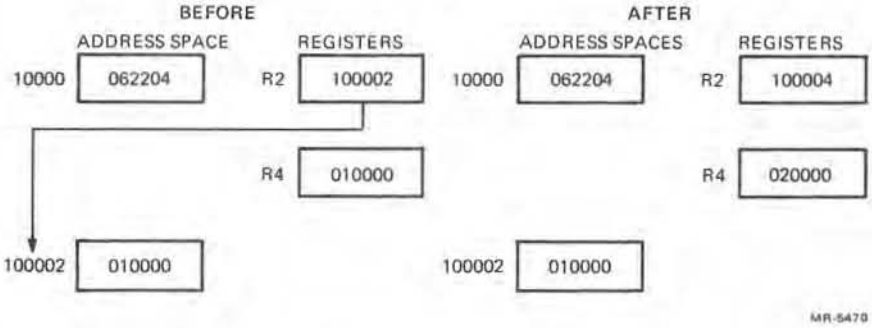


Figure 6-12 ADD (R2) + R4 Add

6.2.3.3 Autodecrement Mode (OPR-(Rn)) -- This mode (mode 4) is useful for processing data in a list in reverse direction. The contents of the selected general purpose register are decremented (by two for word instructions, by one for byte instructions) and then used as the address of the operand. The choice of postincrement, predecrement features for the DCT11-AA were not arbitrary decisions, but were intended to facilitate hardware/software stack operations.

OPR-(Rn)

Autodecrement Mode Examples

	Symbolic	Octal Code	Instruction Name
1.	INC-(R0)	005240	Increment

Operation: The contents of R0 are decremented by two and used as the address of the operand. The operand is incremented by one.

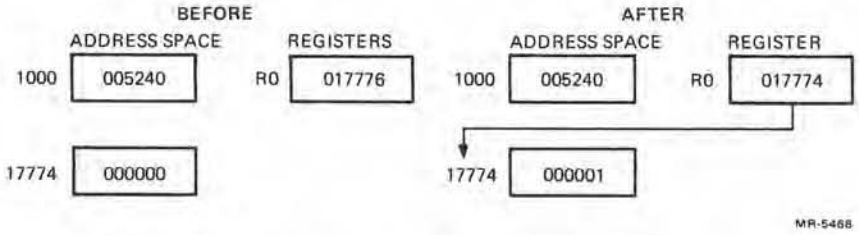


Figure 6-13 INC -(R0) Increment

2. INCB -(R0) 105240 Increment Byte

Operation: The contents of R0 are decremented by one then used as the address of the operand. The operand byte is increased by one.

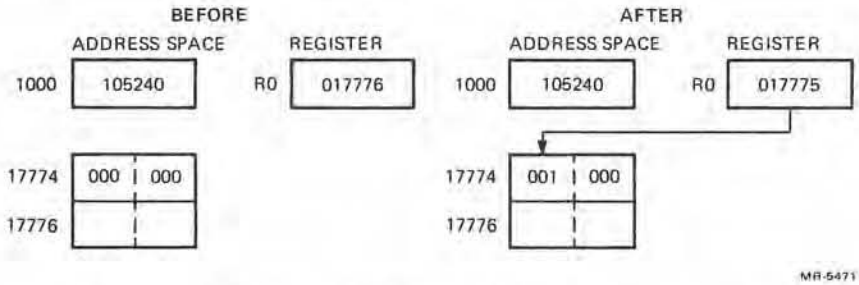


Figure 6-14 INCB -(R0) Increment Byte

3. ADD -(R3),R0 064300 Add

Operation: The contents of R3 are decremented by two then used as a pointer to an operand (source) which is added to the contents of R0 (destination operand).

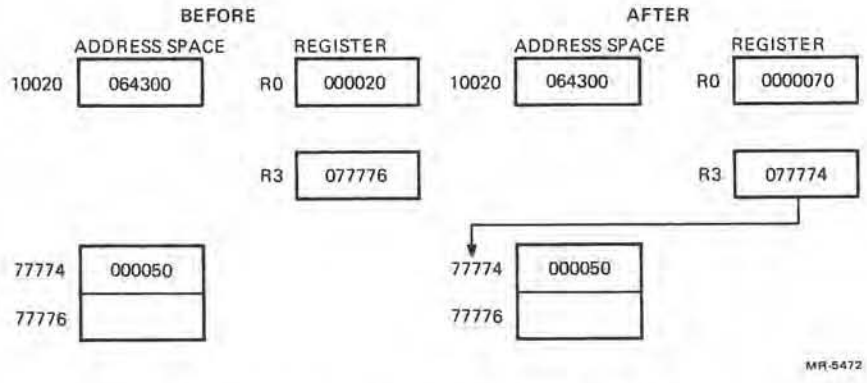


Figure 6-15 ADD -(R3), R0 Add

6.2.3.4 Index Mode (OPR X(Rn)) -- In this mode (mode 6) the contents of the selected general purpose register, and an index word following the instruction word, are summed to form the address of the operand. The contents of the selected register may be used as a base for calculating a series of addresses, thus allowing random access to elements of data structures. The selected register can then be modified by program to access data in the table. Index addressing instructions are of the form OPR X(Rn) where X is the indexed word and is located in the memory location following the instruction word and Rn is the selected general purpose register.

OPR X(Rn)

Index Mode Examples

	Symbolic	Octal Code	Instruction Name
1.	CLR 200(R4)	005064 000200	Clear

Operation: The address of the operand is determined by adding 200 to the contents of R4. The operand location is then cleared.

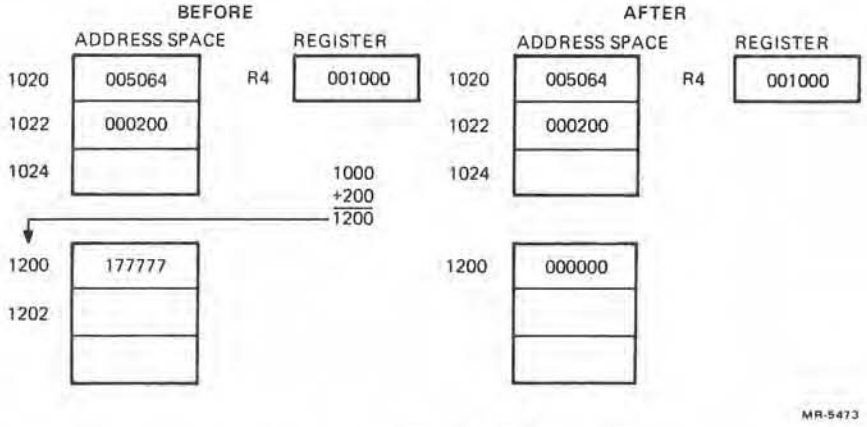


Figure 6-16 CLR 200 (R4) Clear

2. COMB 200(R1) 105161 Complement Byte
000200

Operation: The contents of a location which is determined by adding 200 to the contents of R1 are one's complemented (i.e., logically complemented).

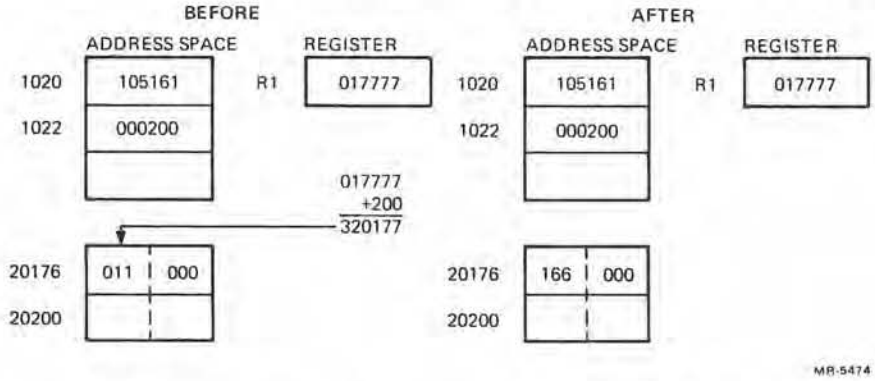


Figure 6-17 COMB 200 (R1) Complement Byte

3. ADD 30(R2),20(R5) 066265 Add
000030
000020

Operation: The contents of a location which is determined by adding 30 to the contents of R2 are added to the contents of a location which is determined by adding 20 to the contents of R5. The result is stored at the destination address, i.e., 20(R5).

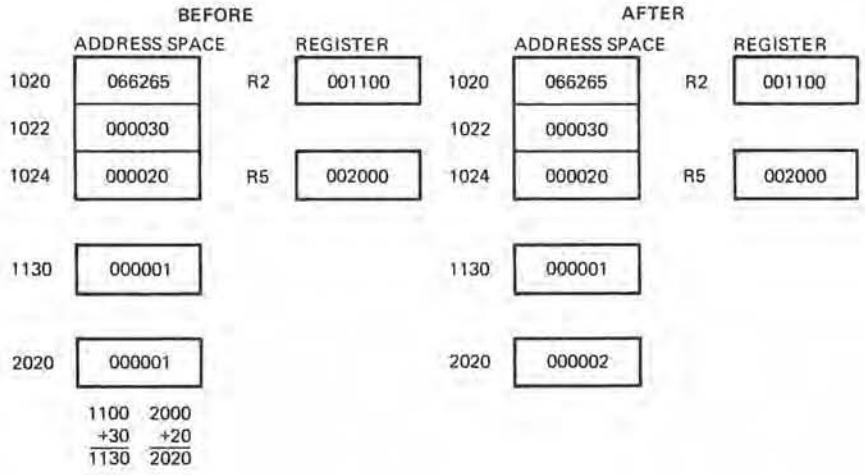


Figure 6-18 ADD 30 (R2), 20 (R5) Add

6.2.4 Deferred (Indirect) Addressing

The four basic modes may also be used with deferred addressing. Whereas in the register mode the operand is the contents of the selected register, in the register deferred mode the contents of the selected register is the address of the operand.

In the three other deferred modes, the contents of the register select the address of the operand rather than the operand itself. These modes are therefore used when a table consists of addresses rather than operands. Assembler syntax for indicating deferred addressing is "@" (or "()" when this is not ambiguous). The following table summarizes the deferred versions of the basic modes.

Mode	Name	Assembler Syntax	Function
1	Register Deferred	@Rn or (Rn)	Register contains the address of the operand.

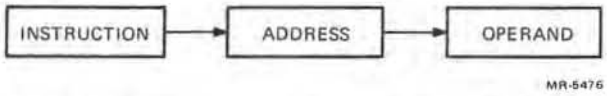


Figure 6-19 Mode 1 Register Deferred

3	Autoincrement Deferred	@(Rn) +	Register is first used as a pointer to a word containing the address of the operand, then incremented (always by two; even for byte instructions).

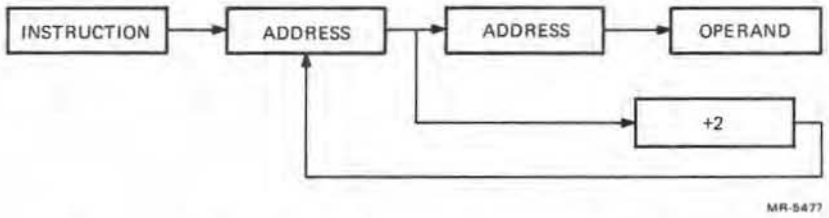


Figure 6-20 Mode 3 Autoincrement Deferred

5	Autodecrement Deferred	@-(Rn)	Register is decremented (always by two; even for byte instructions) and then used as a pointer to a word containing the address of the operand.

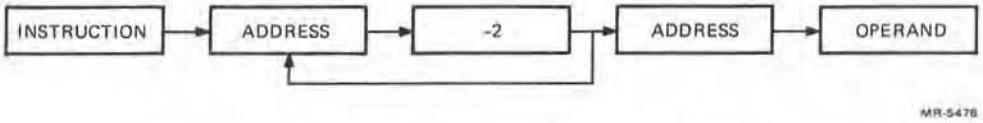


Figure 6-21 Mode 5 Autodecrement Deferred

7	Index Deferred	@X(Rn)	Value X (stored in a word following the instruction) and (Rn) are added and the sum is used as a pointer to a word containing the address of the operand. Neither X nor (Rn) is modified.

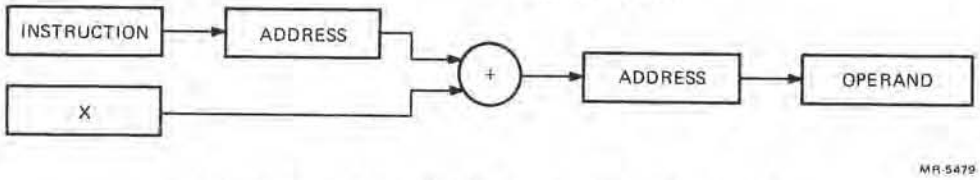


Figure 6-22 Mode 7 Index Deferred

The following examples illustrate the deferred modes.

Register Deferred Mode Example

Symbolic	Octal Code	Instruction Name
CLR @R5	005015	Clear

Operation: The contents of location specified in R5 are cleared.

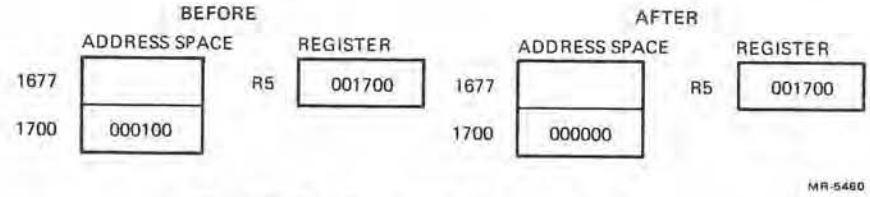


Figure 6-23 CLR @ R5 Clear

Autoincrement Deferred Mode Example (Mode 3)

Symbolic	Octal Code	Instruction Name
INC@(R2)+	005232	Increment

Operation: The contents of R2 are used as the address of the address of the operand.
Operand is increased by one. Contents of R2 are incremented by two.

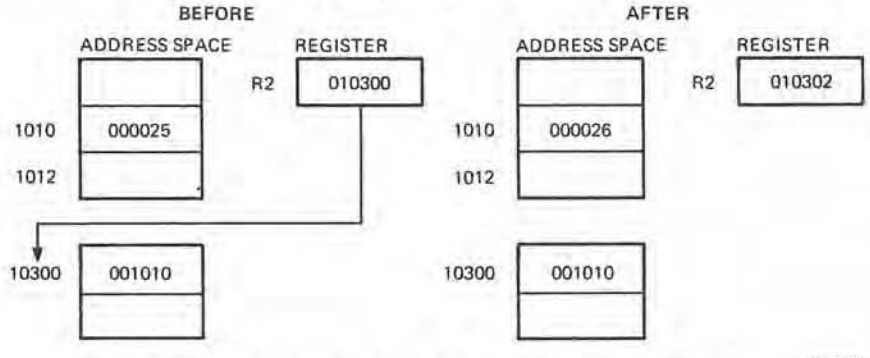


Figure 6-24 INC @ (R2) + Increment

Autodecrement Deferred Mode Example (Mode 5)

Symbolic	Octal Code	Complement
COM @*(R0)	005150	

Operation: The contents of R0 are decremented by two and then used as the address of the address of the operand. Operand is one's complemented (i.e., logically complemented).

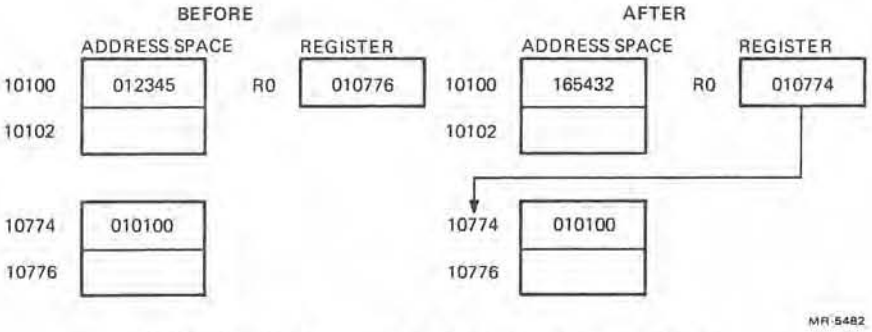


Figure 6-25 COM @ (R0) Complement

Index Deferred Mode Example (Mode 7)

Symbolic	Octal Code	Instruction Name
ADD @ 1000(R2),R1	067201 001000	ADD

Operation: 1000 and contents of R2 are summed to produce the address of the address of the source operand the contents of which are added to contents of R1; the result is stored in R1.

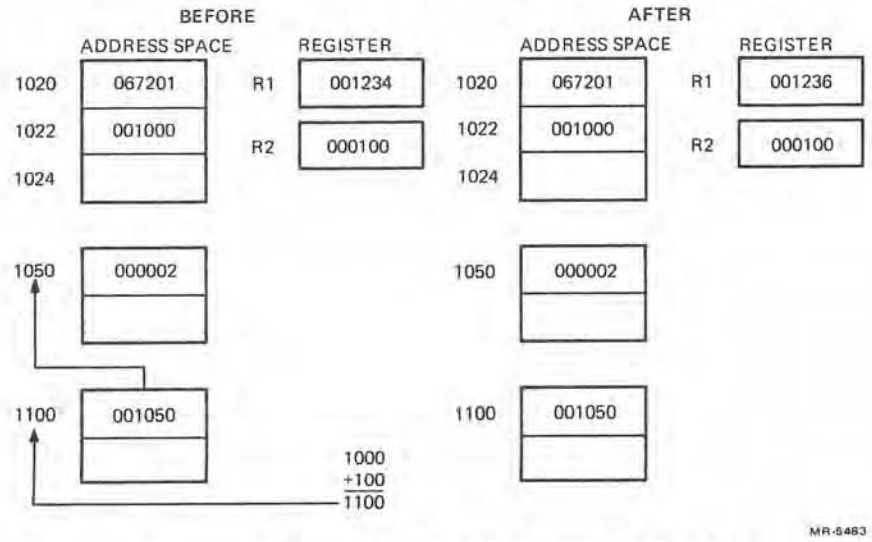


Figure 6-26 ADD @ 1000 (R2), R1 Add

6.2.5 Use of the PC as a General Register

Although register seven is a general purpose register, it doubles in function as the program counter for the DCT11-AA. Whenever the processor uses the program counter to acquire a word from memory, the program counter is automatically incremented by two to contain the address of the next word of the instruction being executed or the address of the next instruction to be executed. (When the program uses the PC to locate byte data, the PC is still incremented by two.)

The PC responds to all the standard DCT11-AA addressing modes. However, there are four of these modes with which the PC can provide advantages for handling position independent code and unstructured data. When utilizing the PC these modes are termed immediate, absolute (or immediate deferred), relative and relative deferred, and are summarized below.

Mode	Name	Assembler Syntax	Function
2	Immediate	#n	Operand follows instruction.
3	Absolute	@#A	Absolute Address of operand follows instruction.
6	Relative	A	Relative Address (index value) follows the instruction.
7	Relative Deferred	@A	Index value (stored in the word following the instruction) is the relative address for the address of the operand.

When a standard program is available for different users, it often is helpful to be able to load it into different areas of memory and run it there. DCT11-AA can accomplish the relocation of a program very efficiently through the use of position independent code (PIC) which is written by using the PC addressing modes. If an instruction and its operands are moved in such a way that the relative distance between them is not altered, the same offset relative to the PC can be used in all positions in memory. Thus, PIC usually references locations relative to the current location.

The PC also greatly facilitates the handling of unstructured data. This is particularly true of the immediate and relative modes.

6.2.5.1 Immediate Mode (OPR #n,DD) -- Immediate mode (mode 2) is equivalent to using the autoincrement mode with the PC. It provides time improvements for accessing constant operands by including the constant in the memory location immediately following the instruction word.

OPR #n,DD

Immediate Mode Example

Symbolic	Octal Code	Instruction Name
ADD #10,R0	062700 000010	Add

Operation: The value 10 is located in the second word of the instruction and is added to the contents of R0. Just before this instruction is fetched and executed, the PC points to the first word of the instruction. The processor fetches the first word and increments the PC by two. The source operand mode is 27 (autoincrement the PC). Thus, the PC is used as a pointer to fetch the operand (the second word of the instruction) before being incremented by two to point to the next instruction.

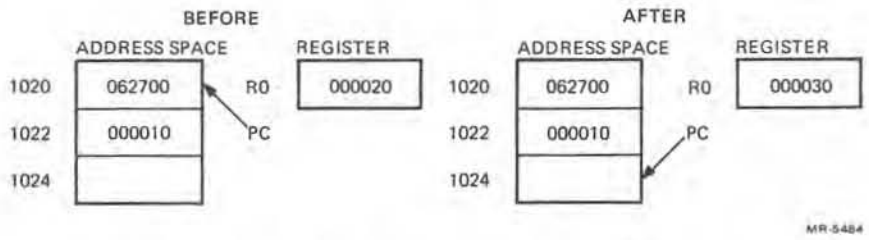


Figure 6-27 ADD # 10, R0 Add

6.2.5.2 Absolute Addressing (OPR @#A) -- This mode (mode 3) is the equivalent of immediate deferred or autoincrement deferred using the PC. The contents of the location following the instruction are taken as the address of the operand. Immediate data is interpreted as an absolute address (i.e., an address that remains constant no matter where in memory the assembled instruction is executed).

OPR @#A

Absolute Mode Examples

	Symbolic	Octal Code	Instruction Name
1.	CLR @#1100	005037 001100	Clear

Operation: Clear the contents of location 1100.

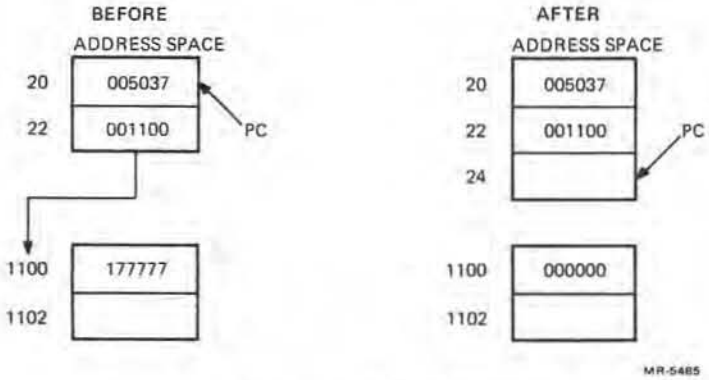


Figure 6-28 CLR @ # 1100 Clear

2.	ADD @#2000,R3	063703 002000	

Operation: Add contents of location 2000 to R3.

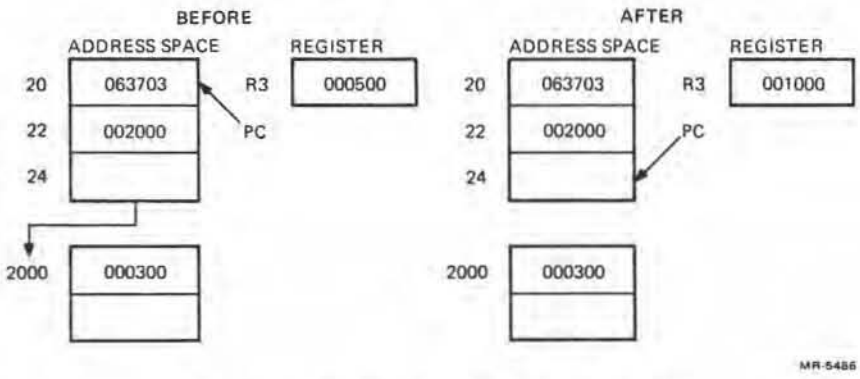


Figure 6-29 ADD @ # 2000 Add

6.2.5.3 Relative Addressing (OPR A or OPR X(PC)) -- This mode (mode 6) is assembled as index mode using R7. The base of the address calculation, which is stored in the second or third word of the instruction, is not the address of the operand, but the number which, when added to the (PC), becomes the address of the operand. This mode is useful for writing position independent code since the location referenced is always fixed relative to the PC. When instructions are to be relocated, the operand is moved by the same amount.

OPR A or OPR X(PC)

where X is the location of A relative to the instruction

Relative Addressing Example

Symbolic	Octal Code	Instruction Name
INC A	005267 000054	Increment

Operation: To increment location A, contents of memory location immediately following instruction word are added to (PC) to produce address A. Contents of A are increased by one.

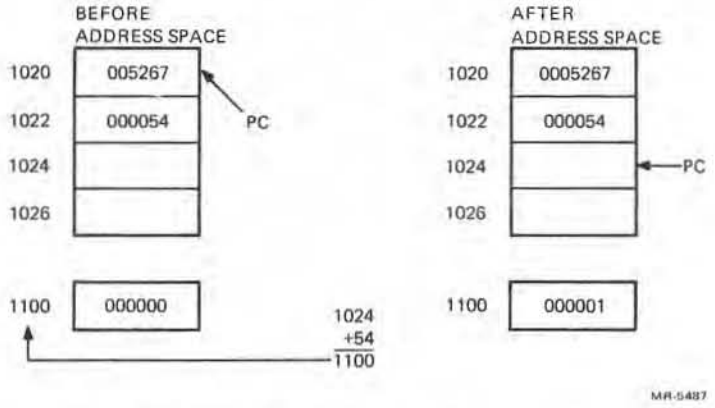


Figure 6-30 INC A Increment

6.2.5.4 Relative Deferred Addressing (OPR @A or OPR @X(PC)) -- This mode (mode 7) is similar to the relative mode, except that the second word of the instruction, when added to the PC, contains the address of the address of the operand, rather than the address of the operand.

OPR @A or OPR @X(PC)

where x is location containing address of A, relative to the instruction

Relative Deferred Mode Example

Symbolic	Octal Code	Instruction Name
CLR @A	005077 000020	Clear

Operation: Add second word of instruction to updated PD to produce address of address of operand. Clear operand.

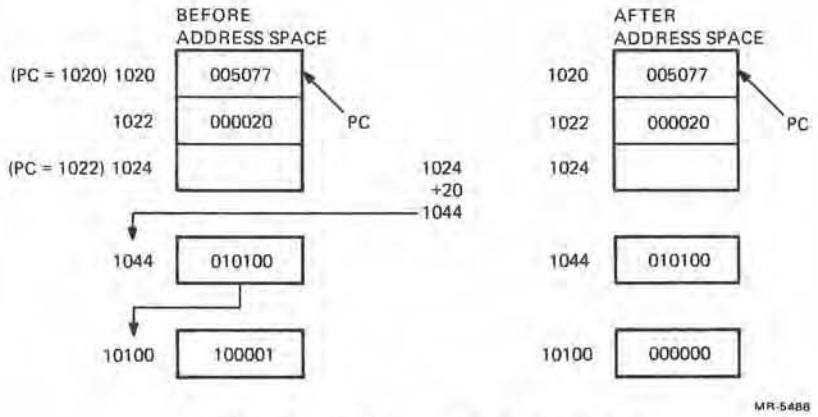


Figure 6-31 CLR @ A Clear

6.2.6 Use of Stack Pointer as General Register
The processor stack pointer (SP, Register six) is in most cases the general register used for the stack operations related to program nesting. Autodecrement with Register six "pushes" data on to the stack and autoincrement with Register six "pops" data off the stack. Since the SP is used by the processor for interrupt handling, it has a special attribute: autoincrements and autodecrements are always done in steps of two. Byte operations using the SP in this way leave odd addresses unmodified.

6.3 INSTRUCTION SET

The rest of Chapter 6 describes the DCT11-AA instructions. Each instruction includes the mnemonic, octal code, binary code, a diagram showing the format of the instruction, a symbolic notation describing its execution and the effect on the condition codes, a description, special comments, and examples.

MNEMONIC: This is indicated before each description. When the word instruction has a byte equivalent, the byte mnemonic is also shown.

INSTRUCTION FORMAT: A diagram accompanying each instruction shows the octal op code, the binary op code, and bit assignments. [Note that in byte instructions the most significant bit (bit 15) is always a one].

SYMBOLS:

() = contents of

SS or src = source address

DD or dst = destination address

loc = location

← = becomes

↑ = "is popped from stack"

↓ = "is pushed onto stack"

∧ = boolean AND

∨ = boolean OR

⊻ = exclusive OR

~ = boolean not

REG or R = register

B = Byte

■ = 0 for word
1 for byte

, = concatenated

6.3.1 Instruction Formats

The following formats include all instructions used in the DCT11-AA. Refer to individual instructions for more detailed information.

1. Single Operand Group (CLR, CLRB, COM, COMB, INC, INCB, DEC, DECB, NEG, NEGB, ADC, ADCB, SBC, SBCB, TST, TSTB, ROR, RORB, ROL, ROLB, ASR, ASRB, ASL, ASLB, JMP, SWAB, MFPS, MTPS, SXT, XOR)

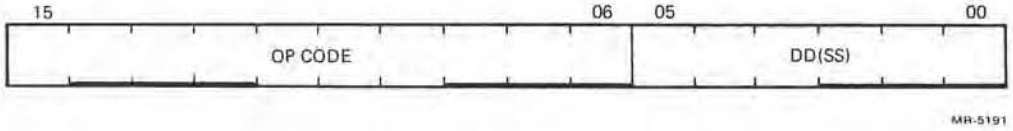


Figure 6-32 Single Operand Group

2. Double Operand Group (BIT, BITB, BIC, BICB, BIS, BISB, ADD, SUB, MOV, MOVB, CMP, CMPB)

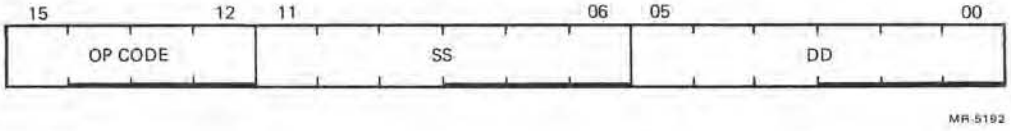


Figure 6-33 Double Operand Group

3. Program Control Group
   a. Branch (all branch instructions)

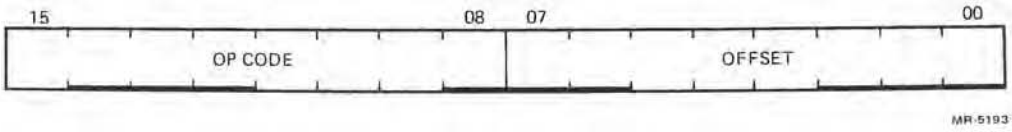


Figure 6-34 Program Control Group Branch

   b. Jump To Subroutine (JSR)

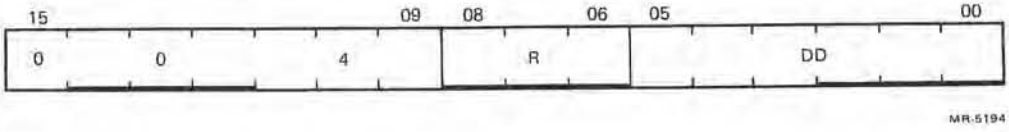


Figure 6-35 Program Control Group JSR

   c. Subroutine Return (RTS)

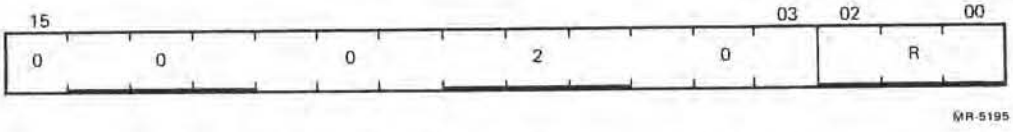


Figure 6-36 Program Control Group RST

d. Traps (break point, IOT, EMT, TRAP, BPT)

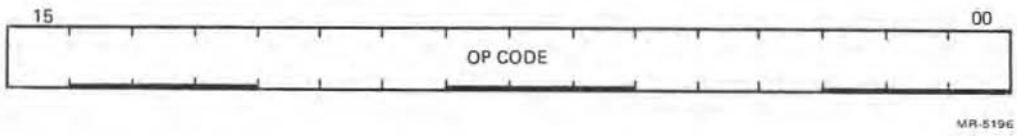


Figure 6-37 Program Control Group Traps

e. Subtract I and branch (if = 0)(SOB)

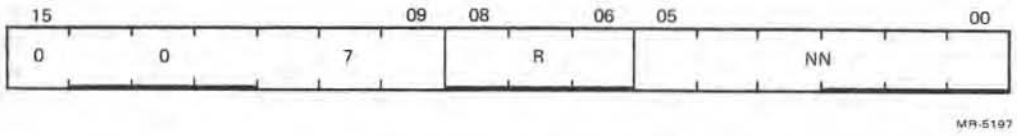


Figure 6-38 Program Control Group Subtract

4. Operate Group (HALT, WAIT, RTI, RESET, RTT, NOP, MFPT)

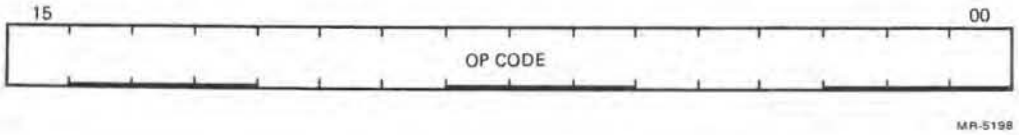


Figure 6-39 Operate Group

5. Condition Code Operators (all condition code instructions)

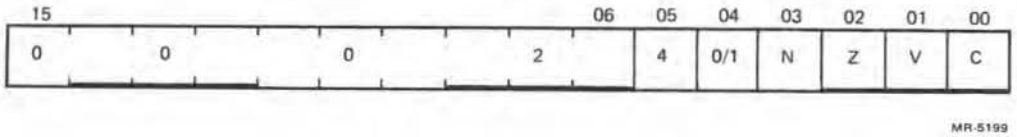


Figure 6-40 Condition Group

6.3.1.1 Byte Instructions -- The DCT11-AA includes a full complement of instructions that manipulate byte operands. Since all DCT11-AA addressing is byte-oriented, byte manipulation addressing is straightforward. Byte instructions with autoincrement or autodecrement direct addressing cause the specified register to be modified by one to point to the next byte of data. Byte operations in register mode access the low-order byte of the specified register. These provisions enable the

DCT11-AA to perform as either a word or byte processor. The numbering scheme for word and byte addresses in memory is:

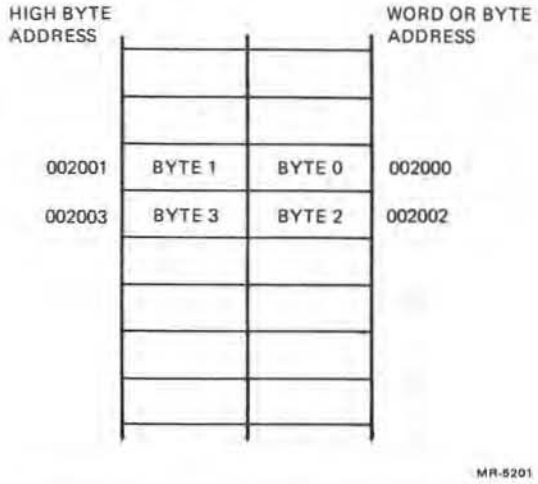


Figure 6-41 Byte Instructions

The most significant bit (Bit 15) of the instruction word is set to indicate a byte instruction.

Example:

Symbolic	Octal	
CLR	0050DD	Clear Word
CLRB	1050DD	Clear Byte

6.3.2 LIST OF INSTRUCTIONS

The DCT11-AA instruction set is shown as follows:

SINGLE OPERAND

Mnemonic	Instruction	Op Code
General		
CLR(B)	clear dst	■050DD
COM(B)	complement dst	■051DD
INC(B)	increment dst	■052DD
DEC(B)	decrement dst	■053DD
NEG(B)	negate dst	■054DD
TST(B)	test dst	■057DD
Shift & Rotate		
ASR(B)	arithmetic shift right	■062DD
ASL(B)	arithmetic shift left	■063DD
ROR(B)	rotate right	■060DD
ROL(B)	rotate left	■061DD
SWAB	swap bytes	0003DD
Multiple Precision		
ADC(B)	add carry	■055DD
SBC(B)	subtract carry	■056DD
SXT	sign extend	0067DD

PS Word Operators

MFPS	move byte from PS	1067DD
MTPS	move byte to PS	1064SS

DOUBLE OPERAND

General

MOV(B)	move source to destination	■1SSDD
CMP(B)	compare src to dst	■2SSDD
ADD	add src to dst	06SSDD
SUB	subtract src from dst	16SSDD

Logical

BIT(B)	bit test	■3SSDD
BIC(B)	bit clear	■4SSDD
BIS(B)	bit set	■5SSDD
XOR	exclusive or	074RDD

PROGRAM CONTROL

Mnemonic	Instruction	Op Code or Base Code
Branch		
BR	branch (unconditional)	000400
BNE	branch if not equal (to zero)	001000
BEQ	branch if equal (to zero)	001400
BPL	branch if plus	100000
BMI	branch if minus	100400
BVC	branch if overflow is clear	102000
BVS	branch if overflow is set	102400
BCC	branch if carry is clear	103000
BCS	branch if carry is set	103400
Signed Conditional Branch		
BGE	branch is greater than or equal (to zero)	002000
BLT	branch if less than (zero)	002400
BGT	branch if greater than (zero)	003000
BLE	branch if less than or equal (to zero)	003400
Unsigned Conditional Branch		
BHI	branch if higher	101000
BLOS	branch if lower or same	101400
BHIS	branch if higher or same	103000
BLO	branch if lower	103400

Jump & Subroutine		
JMP	jump	0001DD
JSR	jump to subroutine	004RDD
RTS	return from subroutine	00020R
SOB	subtract one and branch (if ≠ 0)	077R00
Trap & Interrupt		
EMT	emulator trap	104000--104377
TRAP	trap	104400--104777
BPT	breakpoint trap	000003
IOT	input/output trap	000004
RTI	return from interrupt	000002
RTT	return from interrupt	000006
MISCELLANEOUS		
HALT	halt	000000
WAIT	wait for interrupt	000001
RESET	reset external bus	000005
MFPT	move processor type	000007
RESERVED INSTRUCTIONS		
		00021R
		00022
CONDITION CODE OPERATORS		
CLC	clear C	000241
CLV	clear V	000242
CLZ	clear Z	000244
CLN	clear N	000250
CCC	clear all CC bits	000257
SEC	set C	000261
SEV	set V	000262
SEZ	set Z	000264
SEN	set N	000270
SCC	set all CC bits	000277
NOP	no operation	000240

6.3.3 Single Operand Instructions

NOTE

In all DCT11-AA instructions a write operation (which in 8-bit mode consists of two adjacent and indivisible write transactions) to a memory location or register is always preceded by a read operation from the same location. The exception is when writing PC and PSW to the stack in two cases:

1. The execution of the microcode preceding an interrupt or trap service routine.

2. Interrupt and trap instructions:

HLT
TRAP
BPT
IOT

6.3.3.1 General --

CLR
CLRB

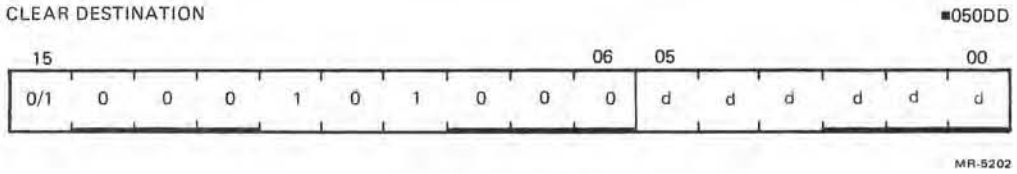


Figure 6-42 CLR

Operation: (dst)←0

Condition Codes:
N: cleared
Z: set
V: cleared
C: cleared

Description: Word: Contents of specified destination are replaced with zeros.
Byte: Same

Example:

```
CLR R1

        Before                  After
(R1) = 177777          (R1) = 000000

        NZVC                    NZVC
        1111                    0100
```

COM
COMB

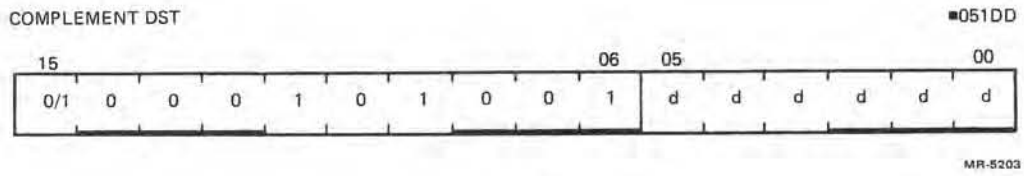


Figure 6-43 COM

Operation: (dst)←~(dst)

Condition Codes:
N: set if most significant bit of result is set; cleared otherwise
Z: set if result is 0; cleared otherwise
V: cleared
C: set

Description: Replaces the contents of the destination address by their logical complement (each bit equal to 0 is set and each bit equal to one is cleared)
Byte: Same

Example:

```
COM R0

        Before                    After
(R0) = 013333          (R0) = 164444

        NZVC                      NZVC
        0110                      1001
```

INC
INCB

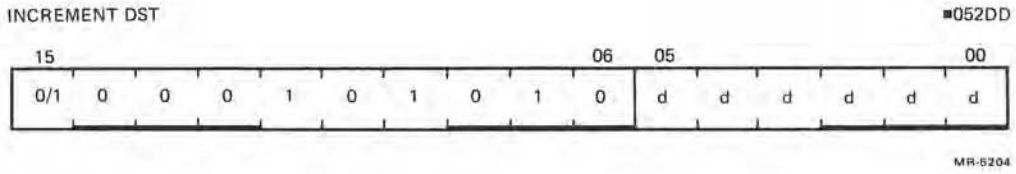


Figure 6-44 INC

Operation: (dst)←(dst) + 1

Condition Codes:
N: set if result is <0; cleared otherwise
Z: set if result is 0; cleared otherwise
V: set if (dst) held 077777; cleared otherwise
C: not affected

Description: Word: Add one to contents of destination
Byte: Same

Example:

```
INC R2

        Before                    After
(R2) = 000333          (R2) = 000334

        NZVC                      NZVC
        0000                      0000
```

DEC
DECB

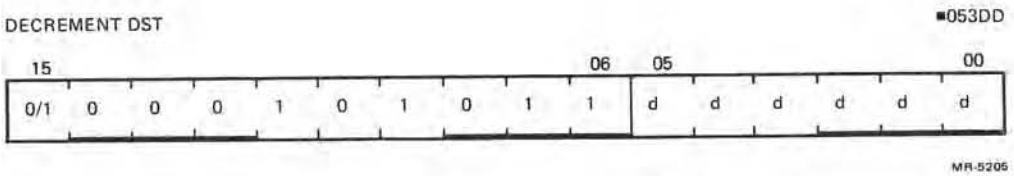


Figure 6-45 DEC

Operation: (dst)<(dst) -1

Condition Codes:
N: set if result is <0, cleared otherwise
Z: set is result is 0; cleared otherwise
V: set if (dst) was 100000; cleared otherwise
C: not affected

Description: Word: Subtract one from the contents of the destination
Byte: Same

Example:

DEC R5

Before	After
(R5) = 000001	(R5) = 000000
NZVC	NZVC
1000	0100

NEG
NEGB

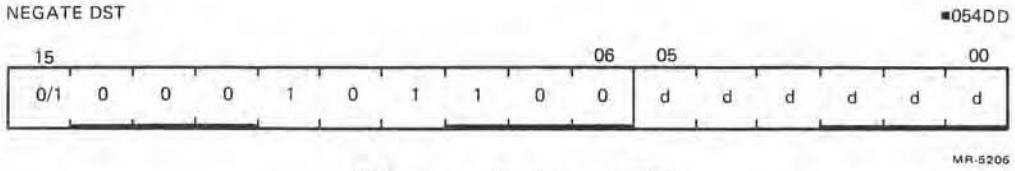


Figure 6-46 NEG

Operation: (dst)← -(dst)

Condition Codes:
N: set if the result is <0; cleared otherwise
Z: set if result is 0; cleared otherwise
V: set if the result is 100000; cleared otherwise
C: cleared if the result is 0; set otherwise

Description: Word: Replaces the contents of the destination address by its two's complement. Note that 100000 is replaced by itself (in two's complement notation the most negative number has no positive counterpart).
Byte: Same

Example:

NEG R0

Before	After
(R0) = 000010	(R0) = 177770
NZVC	NZVC
0000	1001

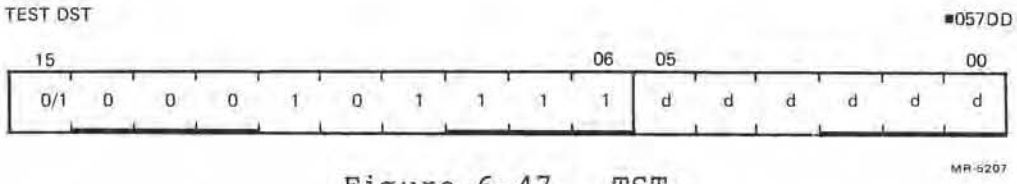


Figure 6-47 TST

Operation: (dst)←(dst)

Condition Codes:
N: set if the result is <0; cleared otherwise
Z: set if result is 0; cleared otherwise
V: cleared
C: cleared

Description: Word: Sets the condition codes N and Z according to the contents of the destination address, contents of dst remains unmodified
Byte: Same

Example:

TST R1

Before	After
(R1) = 012340	(R1) = 012340
NZVC	NZVC
0011	0000

6.3.3.2 **Shifts & Rotates** -- Scaling data by factors of two is accomplished by the shift instructions:

ASR -- Arithmetic shift right

ASL -- Arithmetic shift left

The sign bit (bit 15) of the operand is reproduced in shifts to the right. The low order bit is filled with zero in shifts to the left. Bits shifted out of the C bit, as shown in the following examples, are lost.

The rotate instructions operate on the destination word and the C bit as though they formed a 17-bit "circular buffer." These instructions facilitate sequential bit testing and detailed bit manipulation.

ASR
ASRB

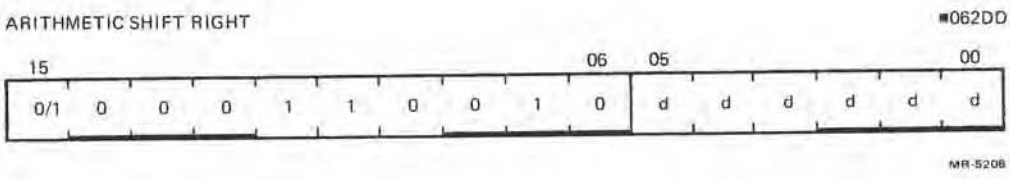


Figure 6-48 ASR

Operation: (dst)←(dst) shifted one place to the right

Condition Codes:
- N: set if the high-order bit of the result is set (result < 0); cleared otherwise
- Z: set if the result = 0; cleared otherwise
- V: loaded from the Exclusive OR of the N-bit and C-bit (as set by the completion of the shift operation)
- C: loaded from low-order bit of the destination

Description: Word: Shifts all bits of the destination right one place. Bit 15 is reproduced. The C-bit is loaded from bit zero of the destination. ASR performs signed division of the destination by two.
Byte: Same

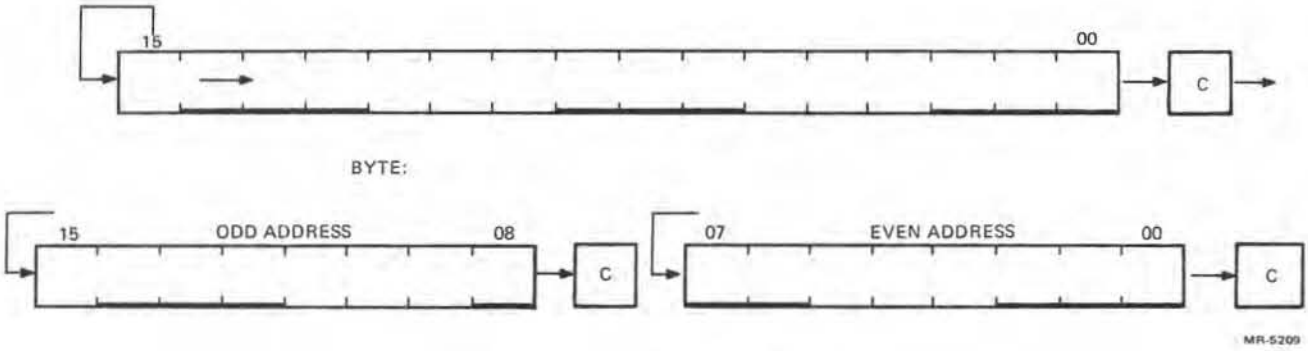


Figure 6-49 ASR Description

ASL
ASLB

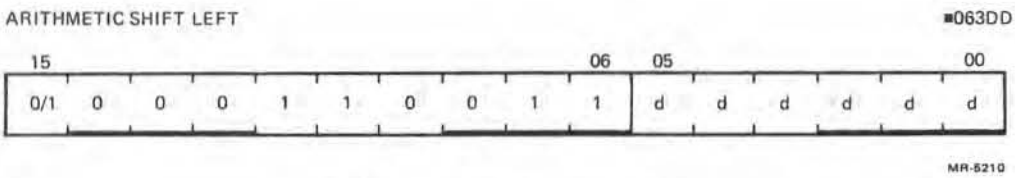


Figure 6-50 ASL

Operation: (dst)←(dst) shifted one place to the left

Condition Codes:
- N: set if high-order bit of the result is set (result < 0); cleared otherwise
- Z: set if the result = 0; cleared otherwise
- V: loaded with the exclusive OR of the N-bit and C-bit (as set by the completion of the shift operation)
- C: loaded with the high-order bit of the destination

Description: Word: Shifts all bits of the destination left one place. Bit zero is loaded with an zero. The C-bit of the status word is loaded from the most significant bit of the destination. ASL performs a signed multiplication of the destination by two with overflow indication.
Byte: Same

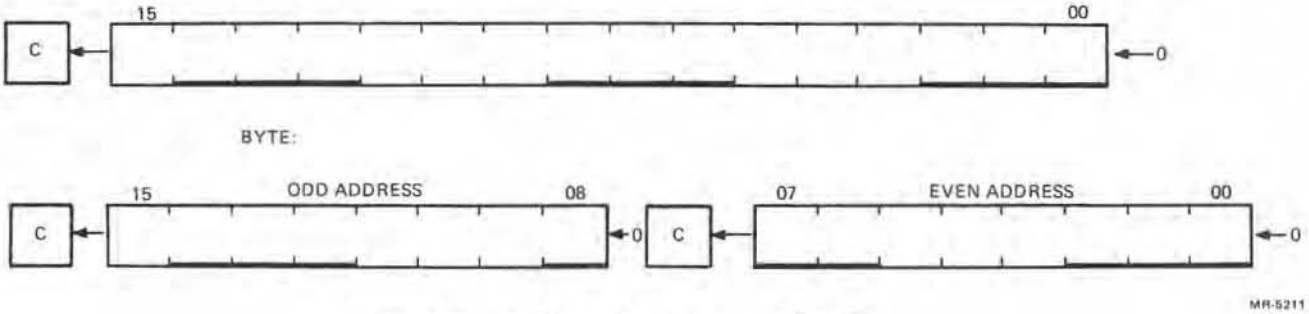


Figure 6-51 ASL Description

ROR
RORB

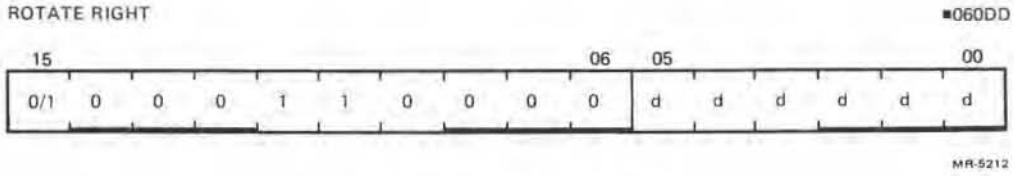


Figure 6-52 ROR

Operation: (dst)←(dst) rotate right one place

Condition Codes:
N: set if the high-order bit of the result is set (result <0); cleared othewise
Z: set if all bits of result = 0; cleared otherwise
V: loaded with the Exclusive OR of the N-bit and C-bit (as set by the completion of the rotate operation)
C: loaded with the low-order bit of the destination

Description: Word: Rotates all bits of the destination right one place. Bit 0 is loaded into the C-bit and the previous contents of the C-bit are loaded into bit 15 of the destination.
Byte: Same

Example:

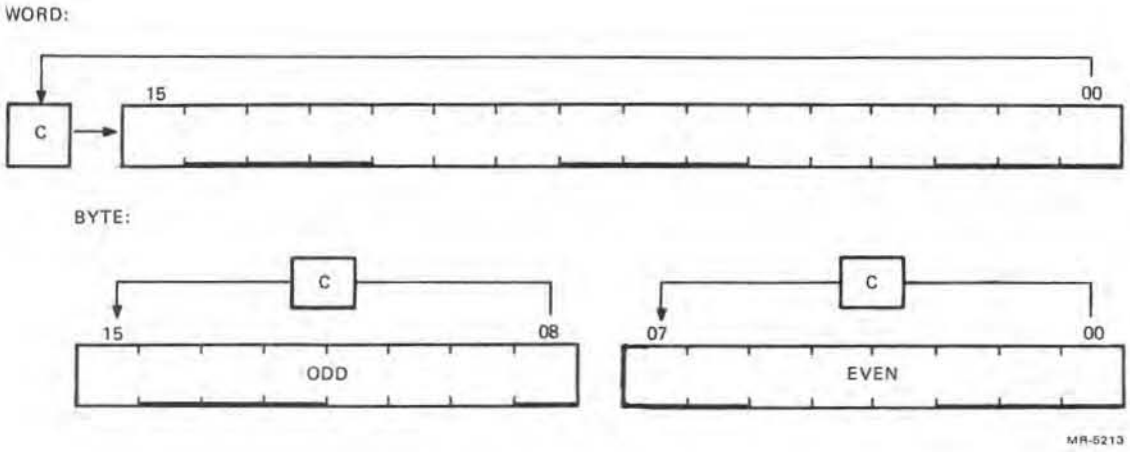


Figure 6-53 ROR Description

ROL
ROLB

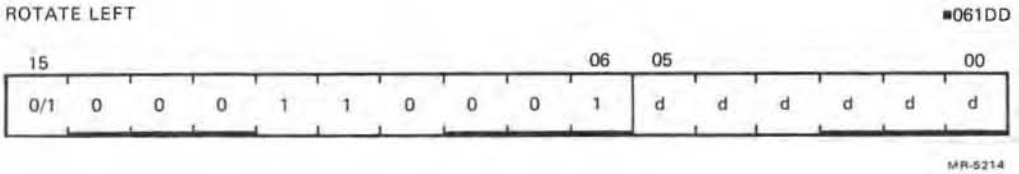


Figure 6-54 ROL

Operation: (dst)←(dst) rotate left one place

Condition Codes:
- N: set if the high-order bit of the result word is set (result < 0); cleared otherwise
- Z: set if all bits of the result word = 0; cleared otherwise
- V: loaded with the Exclusive OR of the N-bit and C-bit (as set by the completion of the rotate operation)
- C: loaded with the high-order bit of the destination

Description: Word: Rotate all bits of the destination left one place. Bit 15 is loaded into the C-bit of the status word and the previous contents of the C-bit are loaded into Bit 0 of the destination.
Byte: Same

Example:

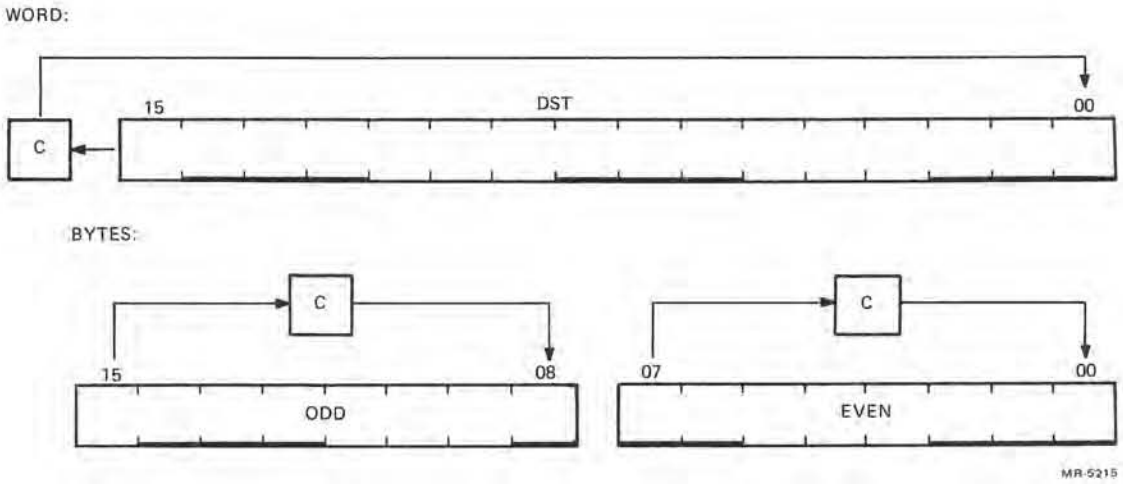


Figure 6-55 ROL Description

SWAB

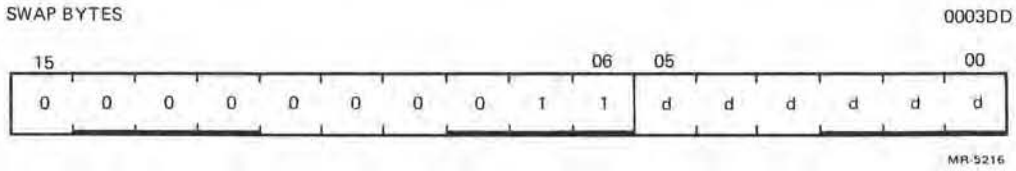


Figure 6-56 SWAB

Operation: Byte 1/Byte 0 ←Byte 0/Byte 1

Condition Codes:
N: set if high-order bit of low-order byte (bit 7) of result is set; cleared otherwise
Z: set if low-order byte of result = 0; cleared otherwise
V: cleared
C: cleared

Description: Exchanges high-order byte and low-order byte of the destination word (destination must be a word address).

Example:

```
                    SWAB R1

          Before                    After
(R1) = 077777             (R1) = 177577

          NZVC                      NZVC
          1111                      0000
```

6.3.3.3 Multiple Precision -- It is sometimes necessary to do arithmetic on operands considered as multiple words or bytes. The DCT11-AA makes special provision for such operations with the instructions ADC (Add Carry) and SBC (Subtract Carry) and their byte equivalents.

For example, two 16-bit words may be combined into a 32-bit double precision word and added or subtracted as shown below.

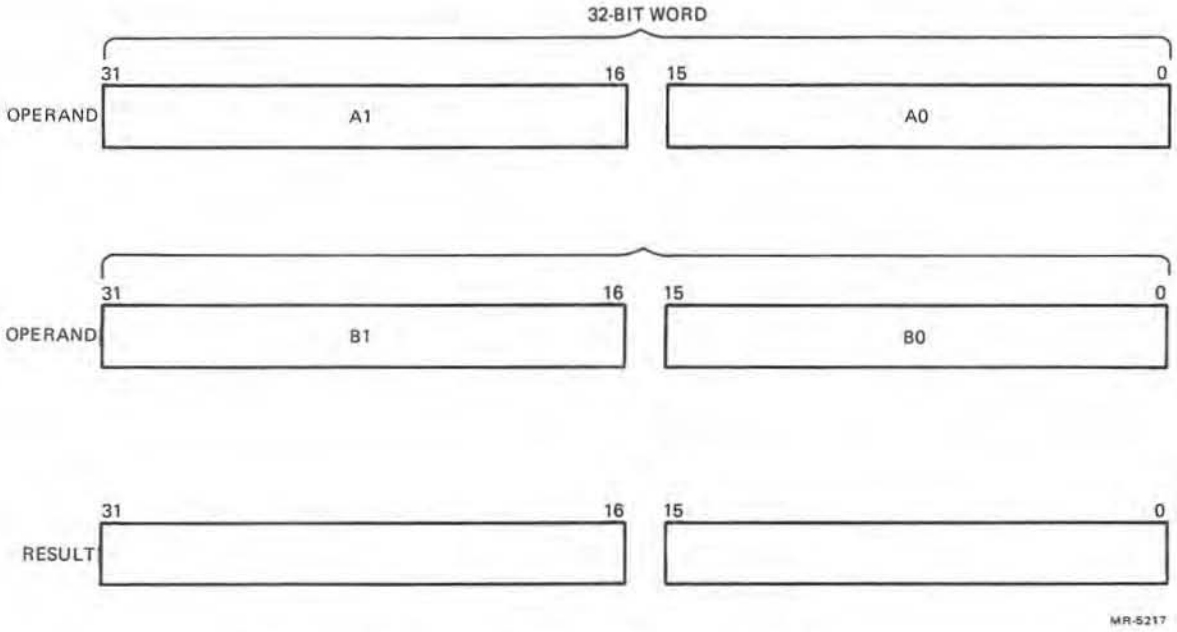


Figure 6-57 Multiple Precision

The addition of -1 and -1 could be performed as follows:

-1 = 37777777777

(R1) = 177777 (R2) = 177777 (R3) = 177777 (R4) = 177777

```
ADD    R1,R2
ADC    R3
ADD    R4,R3
```

1. After (R1) and (R2) are added, 1 is loaded into the C bit
2. ADC instruction adds C bit to (R3); (R3) = 0
3. (R3) and (R4) are added
4. Result is 37777777776 or -2

ADC
ADCB

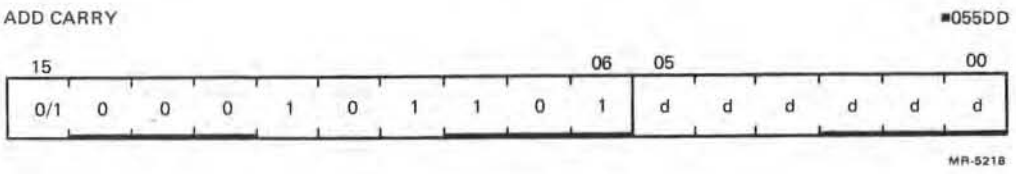


Figure 6-58 ADC

Operation: (dst)←(dst)+(C bit)

Condition Codes:
N: set if result < 0: cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if (dst) was 077777 and (C) was 1; cleared otherwise
C: set if (dst) was 177777 and (c) was 1; cleared otherwise

Description:
Adds the contents of the C-bit into the destination. This permits the carry from the addition of the low-order words to be carried into the high-order result.
Byte: Same

Example:
Double precision addition may be done with the following instruction sequence:

```
ADD     A0,B0           ;add low-order parts
ADC     B1              ;add carry into high-order
ADD     A1,B1           ;add high-order parts
```

SBC
SBCB

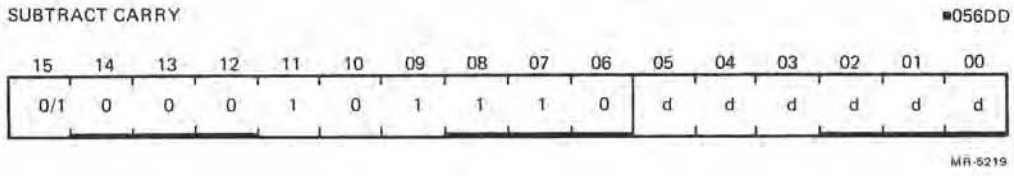


Figure 6-59 SBC

Operation:
(dst)←(dst)-(C)

Condition Codes:
N: set if result <0; cleared otherwise
Z: set if result 0; cleared otherwise
V: set if (dst) was 100000; cleared otherwise
C: set if (dst) was 0 and C was 1; cleared otherwise

Description:
Word: Subtracts the contents of the C-bit from the destination. This permits the carry from the subtraction of two low-order words to be subtracted from the high order part of the result.
Byte: Same

Example:
Double precision subtraction is done by:

```
SUB     A0,B0
SBC     B1
SUB     A1,B1
```

SXT

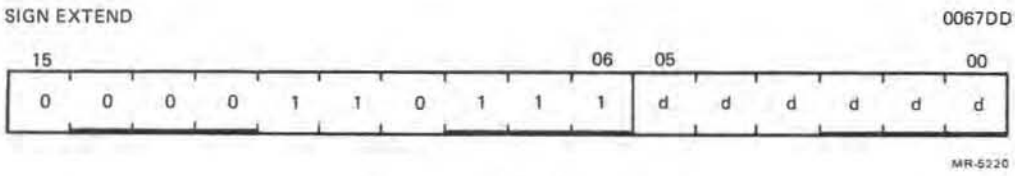


Figure 6-60 SXT

Operation: (dst)←0 if N-bit is clear
(dst)←1 N-bit is set

Condition Codes:
N: unaffected
Z: set if N-bit clear
V: cleared
C: unaffected

Description: If the condition code bit N is set then a -1 is placed in the destination operand; if N bit is clear, then a zero is placed in the destination operand. This instruction is particularly useful in multiple precision arithmetic because it permits the sign to be extended through multiple words.

Example:

SXT A

Before	After
(A) = 012345	(A) = 177777
NZVC	NZVC
1000	1000

6.3.3.4 PS Word Operators --

MFPS

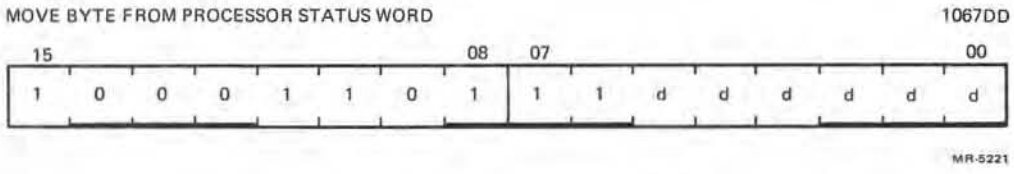


Figure 6-61 MFPS

Operation: (dst)←PS
dst lower 8 bits

Condition Code Bits:
N = set if PS bit 7 = 1; cleared otherwise
Z = set if PS <0:7> = 0; cleared otherwise
V = cleared
C = not affected

Description: The 8 bit contents of the PS are moved to the effective destination. If destination is mode 0, PS bit 7 is sign extended through upper byte of the register. The destination operand address is treated as a byte address.

Example:

```
MFPS R0

          Before                    After
R0 [0]                    R0 [000014]
PS [000014]               PS [000000]
```

MTPS

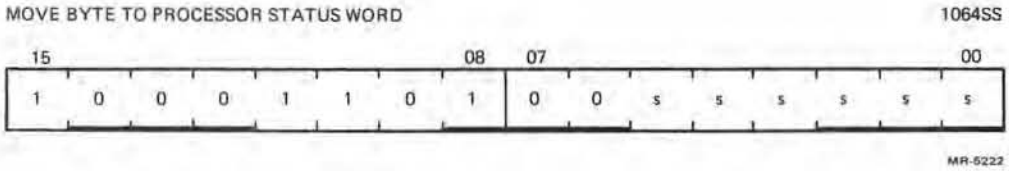


Figure 6-62 MTPS

Operation: PS←(src)

Condition Codes: Set according to effective SRC operand bits 0--3

Description: The eight bits of the effective operand replaces the current contents of the PS. The source operand address is treated as a byte address. Note: the T bit (PS bit four) cannot be set with this instruction. The SRC operand remains unchanged. This instruction can be used to change the priority bits (PS bits 7--5) in the PS.

Example:

```
MTPS R1

          Before                    After
(R1) = 000777             (R1) = 000777
(PS) = XXX000             (PS) = XXX357

          NZVC                      NZVC
          0000                      1111
```

6.3.4 Double Operand Instructions

Double operand instructions save instructions (and time) since they eliminate the need for "load" and "save" sequences such as those used in accumulator-oriented machines.

6.3.4.1 General --

MOV
MOVB

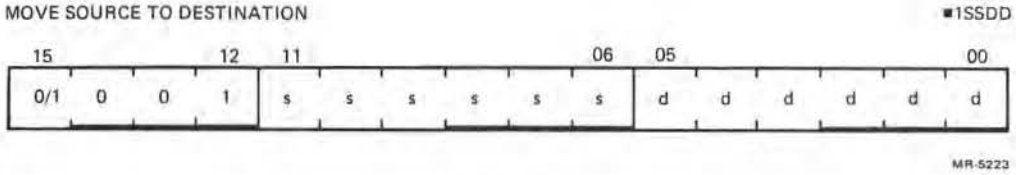


Figure 6-63 MOV

Operation: (dst)←(src)

Condition Codes:
N: set if (src) <0; cleared otherwise
Z: set if (src) = 0; cleared otherwise
V: cleared
C: not affected

Description: Word: Moves the source operand to the destination location. The previous contents of the destination are lost. The contents of the source address are not affected.
Byte: Same as MOV. The MOVB to a register (unique among byte instructions) extends the most significant bit of the low order byte (sign extension). Otherwise MOVB operates on bytes exactly as MOV operates on words.

Example:

```
MOV XXX,R1          ;loads Register one with the contents of memory location; XXX represents a programmer-defined mnemonic used to represent a memory location.

MOV #20,R0          ;loads the number 20 into Register 0; "#" indicates that the value 20 is the operand.

MOV @#20,-(R6)      ;pushes the operand contained in location 20 onto the stack.

MOV (R6)+,@#177566  ;pops the operand off a stack and moves it into memory location 177566 (terminal print buffer).

MOV R1,R3           ;performs an inter register transfer.
```

```
MOVB @#177562,@#177566  ;moves a character from
                        terminal keyboard buffer
                        to terminal printer
                        buffer.
```

CMP
CMPB

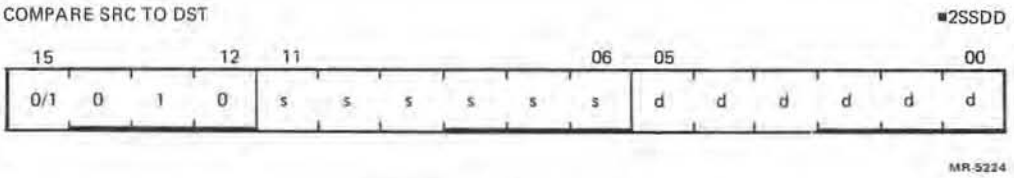


Figure 6-64 CMP

Operation: (src)-(dst)

Condition Codes:
N: set if result <0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if there was arithmetic overflow; that is, operands were of opposite signs and the sign of the destination was the same as the sign of the result; cleared otherwise.
C: cleared if there was a carry from the most significant bit of the result; set otherwise.

Description: Compares the source and destination operands and sets the condition codes, which may then be used for arithmetic and logical conditional branches. Both operands are unaffected. The only action is to set the condition codes. The compare is customarily followed by a conditional branch instruction. Note: Unlike the subtract instruction the order of operation is (src)-(dst), not (dst)-(src).

ADD

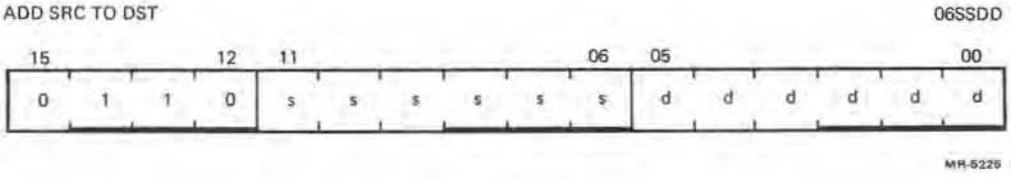


Figure 6-65 ADD

Operation: (dst)←(src)+(dst)

Condition Codes:
N: set if result <0; cleared otherwise.
Z: set if result = 0; cleared othrwise.
V: set if there was arithmetic overflow as a result of the operation; that is both operands were of the same sign and the result was of the opposite sign; cleared otherwise.
C: set if there was a carry from the most significant bit of the result; cleared otherwise.

Description: Adds the source operand to the destination operand and stores the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. Two's complement addition is performed. Note: There is no equivalent byte mode.

Examples:

Add to register:	ADD 20,R0
Add to memory:	ADD R1,XXX
Add register to register:	ADD R1,R2
Add memory to memory:	ADD @#17750,XXX

XXX is a programmer-defined mnemonic for a memory location.

SUB

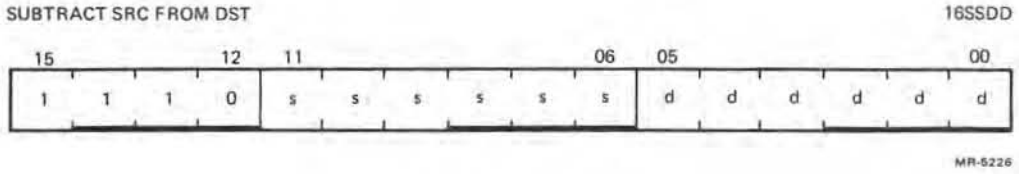


Figure 6-66 SUB

Operation: (dst)←(dst)-(src)

Condition Codes:
N: set if result <0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if there was arithmetic overflow as a result of the operation, that is if operands were of opposite signs and the sign of the source was the same as the sign of the result; cleared otherwise.
C: cleared if there was a carry from the most significant bit of the result; set otherwise.

Description: Subtracts the source operand from the destination operand and leaves the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. In double-precision arithmetic the C-bit, when set, indicates a "borrow". Note: There is no equivalent byte mode.

Example: SUB R1,R2

Before	After
(R1) = 011111	(R1) = 011111
(R2) = 012345	(R2) = 001234
NZVC	NZVC
1111	0000

6.3.4.2 Logical -- These instructions have the same format as the double operand arithmetic group. They permit operations on data at the bit level.

BIT
BITB

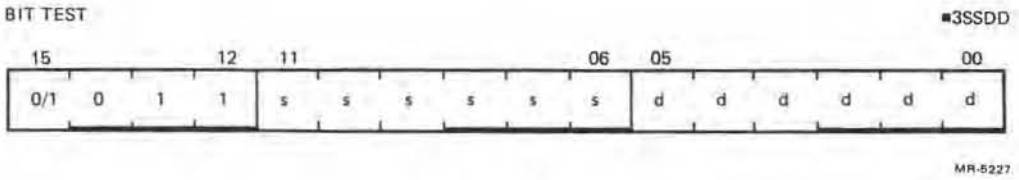


Figure 6-67 BIT

Operation: (src) ∧ (dst)

Condition Codes:
N: set if high-order bit of result set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not affected

Description: Performs logical "and" comparison of the source and destination operands and modifies condition codes accordingly. Neither the source nor destination is affected. The BIT instruction may be used to test whether any of the corresponding bits that are set in the destination are also set in the source or whether all corresponding bits set in the destination are clear in the source.

Example: BIT #30,R3 test bits three and four of R3 to see if both are off

R3 = 0 000 000 000 011 000

Before	After
NZVC	NZVC
1111	0001

BIC
BICB

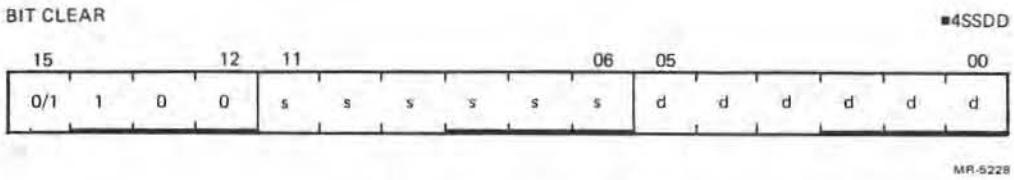


Figure 6-68 BIC

Operation: (dst)←~(src) ∧ (dst)

Condition Codes:
N: set if high order bit of result set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not affected

Description: Clears each bit in the destination that corresponds to a set bit in the source. The original contents of the destination are lost. The contents of the source are unaffected.

Example:

```
                    BIC R3,R4

          Before                        After
(R3) = 001234                 (R3) = 001234
(R4) = 001111                 (R4) = 000101

          NZVC                          NZVC
          1111                          0001

Before:        (R3) = 0 000 001 010 011 100
               (R4) = 0 000 001 001 001 001

After:         (R4) = 0 000 000 001 000 001
```

BIS
BISB

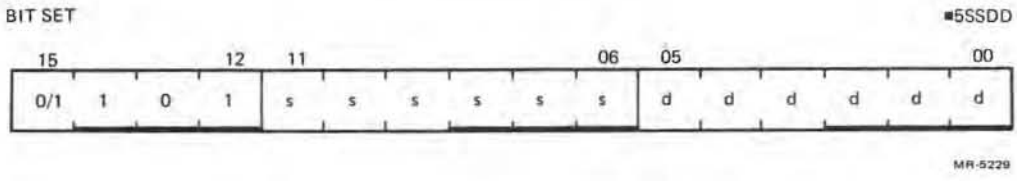


Figure 6-69 BIS

Operation: (dst)←(src) V (dst)

Condition Codes:
N: set if high-order bit of result set, cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not affected

Description: Performs "Inclusive OR" operation between the source and destination operands and leaves the result at the destination address; that is, corresponding bits set in the source are set in the destination. The contents of the destination are lost.

Example:

```
                        BIS R0,R1

          Before                          After
(R0) = 001234                   (R0) = 001234
(R1) = 001111                   (R1) = 001335

          NZVC                            NZVC
          0000                            0000

Before:        (R0) = 0 000 001 010 011 100
               (R1) = 0 000 001 001 001 001

After:         (R1) = 0 000 001 011 011 101
```

XOR

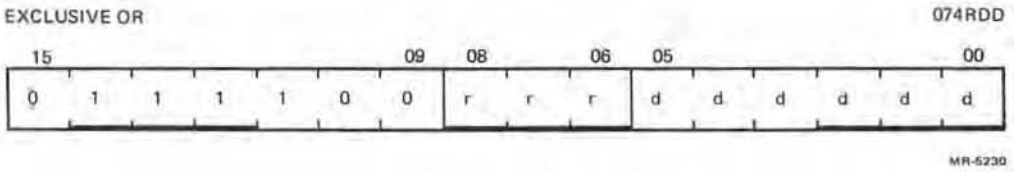


Figure 6-70 XOR

Operation: (dst)←(reg)⊻(dst)

Condition Codes:
N: set if the result <0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: unaffected

Description: The exclusive OR of the register and destination operand is stored in the destination address. Contents of register are unaffected. Assembler format is: XOR R,D.

Example:

```
                        XOR R0,R2

          Before                          After
(R0) = 001234                   (R0) = 001234
(R2) = 001111                   (R2) = 000325

          NZVC                            NZVC
          1111                            0001

Before:        (R0) = 0 000 001 010 011 100
               (R2) = 0 000 001 001 001 001

After:         (R2) = 0 000 000 011 010 101
```

6.3.5 Program Control Instructions

6.3.5.1 Branches -- These instructions cause a branch to a location defined by the sum of the offset (multiplied by two) and the current contents of the Program Counter if:

1. the branch instruction is unconditional.

2. it is conditional and the conditions are met after testing the condition codes (NZVC).

The offset is the number of words from the current contents of the PC forward or backward. Note that the current contents of the PC point to the word following the branch instruction.

Although the offset expresses a byte address the PC is expressed in words. The offset is automatically multiplied by two and sign extended to express words before it is added to the PC. Bit seven is the sign of the offset. If it is set, the offset is negative and the branch is done in the backward direction. Similarly if it is not set, the offset is positive and the branch is done in the forward direction.

The 8-bit offset allows branching in the backward direction by 200_8 words (400 bytes) from the current PC, and in the forward direction by 177_8 words (376 bytes) from the current PC.

The DCT11-AA assembler handles address arithmetic for the user and computes and assembles the proper offset field for branch instructions in the form:

Bxx loc

Where "Bxx" is the branch instruction and "loc" is the address to which the branch is to be made. The assembler gives an error indication in the instruction it the permissible branch range is exceeded. Branch instructions have no effect on condition codes. Conditional branch instructions where the branch condition is not met, are treated as NO OPs.

BR

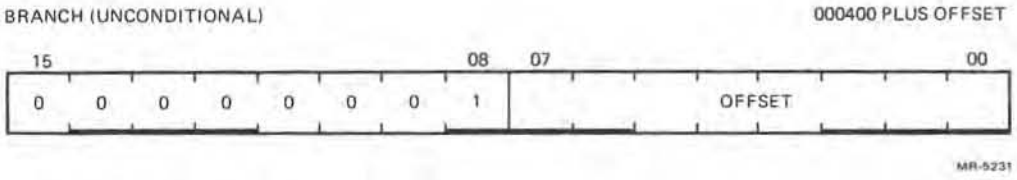


Figure 6-71 BR

Operation: PC←PC+ (2 X offset)

Condition Codes: Unaffected

Description: Provides a way of transferring program control within a range of -128_{10} to $+127_{10}$ words with a one word instruction.

New PC address = updated PC + (2 X offset)

Updated PC = address of branch instruction +2

Example: With the Branch instruction at location 500, the following offsets apply.

New PC Address	Offset Code	Offset (decimal)
474	375	-3
476	376	-2
500	377	-1
502	000	0
504	001	+1
506	002	+2

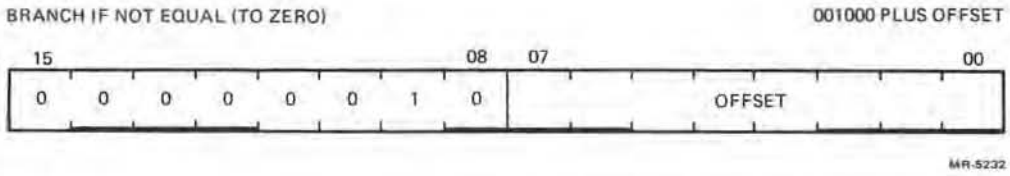


Figure 6-72 BNE

Operation: PC←PC+ (2 X offset) if Z = 0

Condition Codes: Unaffected

Description: Tests the state of the Z-bit and causes a branch if the Z-bit is clear. BNE is the complementary operation to BEQ. It is used to test inequality following a CMP, to test that some bits set in the destination were also in the source, following a BIT operation, and generally, to test that the result of the previous operation was not zero.

Example: Branch to C if A ≠ B

```
CMP A,B          ;compare A and B
BNE C            ;branch if they are not equal
```

Branch to C if A + B ≠ 0

```
ADD A,B          ;add A to B
BNE C            ;Branch if the result is not
                  equal to 0
```

BEQ

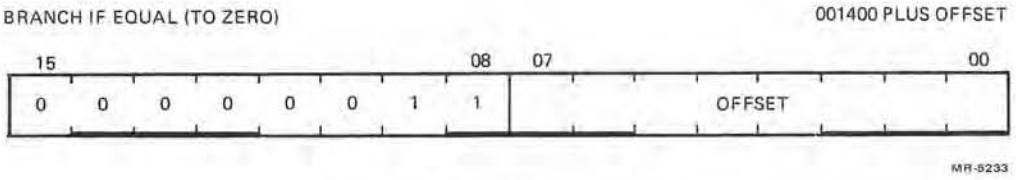


Figure 6-73 BEQ

Operation: PC←PC+ (2 X offset) if Z = 1

Condition Codes: Unaffected

Description: Tests the state of the Z-bit and causes a branch if Z is set. As an example, it is used to test equality following a CMP operation, to test that no bits set in the destination were also set in the source following a BIT operation, and generally, to test that the result of the previous operation was zero.

Example: Branch to C if A = B (A - B = 0)

```
CMP A,B          ;compare A and B
BEQ C            ;branch if they are equal
```

Branch to C if A + B = 0

```
ADD  A,B         ;add A to B
BEQ  C           ;branch if the result = 0
```

BPL

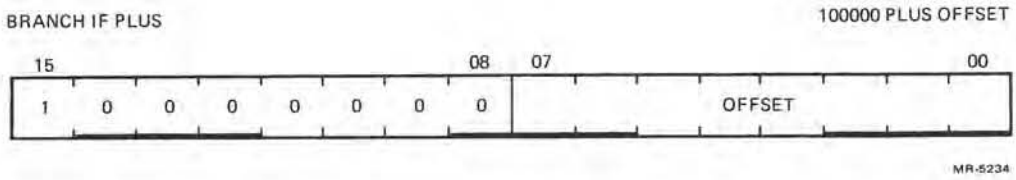


Figure 6-74 BPL

Operation: PC←PC+ (2 X offset) if N = 0

Condition Codes: Unaffected

Description: Tests the state of the N-bit and causes a branch if N is clear, (positive result). BPL is the complementary operation of BMI.

BMI

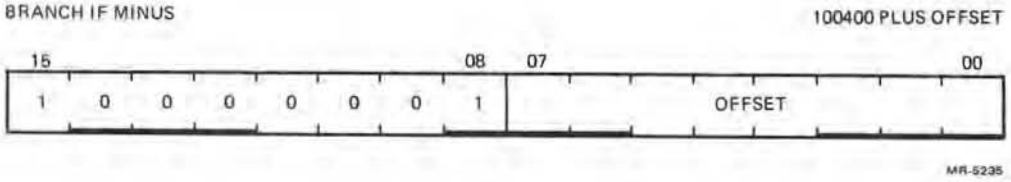


Figure 6-75 BMI

Operation: PC←PC+ (2 X offset) if N = 1

Condition Codes: Unaffected

Description: Tests the state of the N-bit and causes a branch if N is set. It is used to test the sign (most significant bit) of the result of the previous operation), branching if negative. BMI is the complementary function of BPL.

BVC

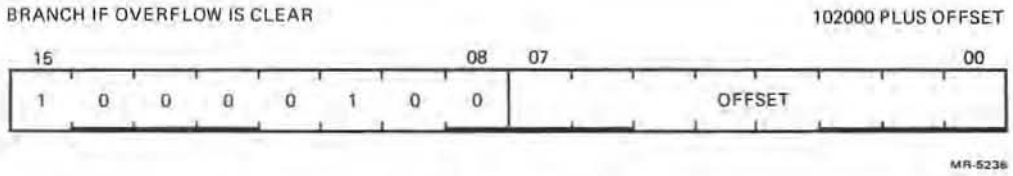


Figure 6-76 BVC

Operation: PC←PC+ (2 X offset) if V = 0

Condition Codes: Unaffected

Description: Tests the state of the V-bit and causes a branch if the V bit is clear. BVC is complementary operation to BVS.

BVS

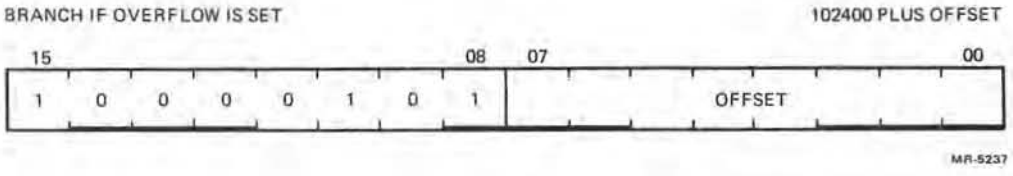


Figure 6-77 BVS

Operation: PC←PC+ (2 X offset) if V = 1

Condition Codes: Unaffected

Description: Tests the state of V-bit (overflow) and causes a branch if the V bit is set. BVS is used to detect arithmetic overflow in the previous operation.

BCC

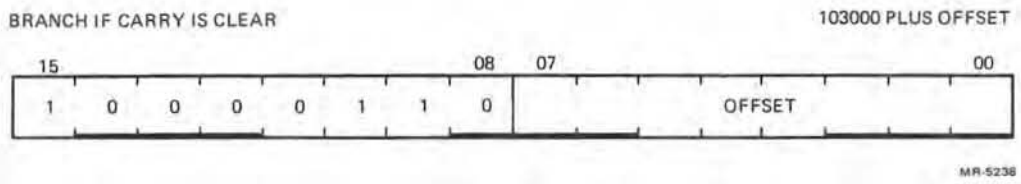


Figure 6-78 BCC

Operation: PC←PC+ (2 X offset) if C = 0

Condition Codes: Unaffected

Description: Tests the state of the C-bit and causes a branch if C is clear. BCC is the complementary operation to BCS.

BCS

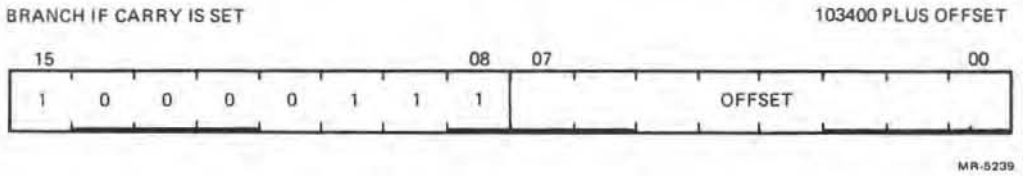


Figure 6-79 BCS

Operation: PC←PC+ (2 X offset) if C = 1

Condition Codes: Unaffected

Description: Tests the state of the C-bit and causes a branch if C is set. It is used to test for a carry in the result of a previous operation.

6.3.5.2 Signed Conditional Branches -- Particular combinations of the condition code bits are tested with the signed conditional branches. These instructions are used to test the results of instructions in which the operands were considered as signed (two's complement) values.

Note that the sense of signed comparisons differs from that of unsigned comparisons in that in signed 16-bit, two's complement arithmetic the sequence of values is as follows.

largest	077777
	077776
positive	.
	.
	.
	000001
	000000
	177777
	177776
	.
negative	.
	.
	100001
smallest	100000

whereas in unsigned 16-bit arithmetic the sequence is considered to be

highest	177777
	.
	.
	.
	.
	.
	.
	000002
	000001
lowest	000000

BGE

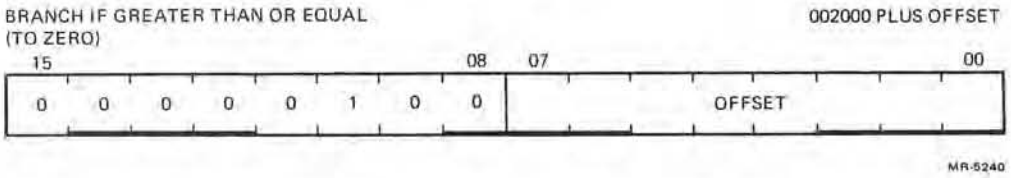


Figure 6-80 BGE

Operation: PC←PC+ (2 X offset) if N⊻V = 0

Condition Codes: Unaffected

Description: Causes a branch if N and V are either both clear or both set. BGE is the complementary operation to BLT. Thus BGE will always cause a branch when it follows an operation that caused addition of two positive numbers. BGE will also cause a branch on a zero result.

BLT

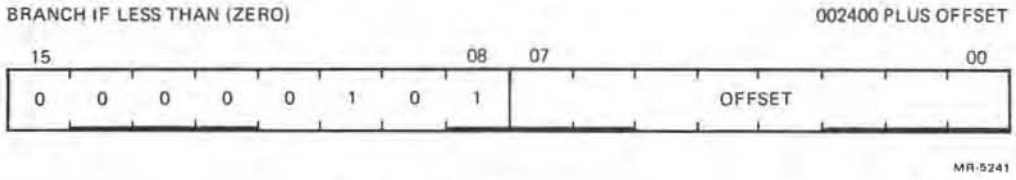


Figure 6-81 BLT

Operation: PC←PC+ (2 X offset) if N⊻V = 1

Condition Codes: Unaffected

Description: Causes a branch if the "Exclusive Or" of the N and V bits are one. Thus BLT will always branch following an operation that added two negative numbers, even if overflow occurred. In particular, BLT will always cause a branch if it follows a CMP instruction operating on a negative source and a positive destination (even it overflow occurred). Further, BLT will never cause a branch when it follows a CMP instruction operating on a positive source and negative destination. BLT will not cause a branch if the result of the previous operation was zero (without overflow).

BGT

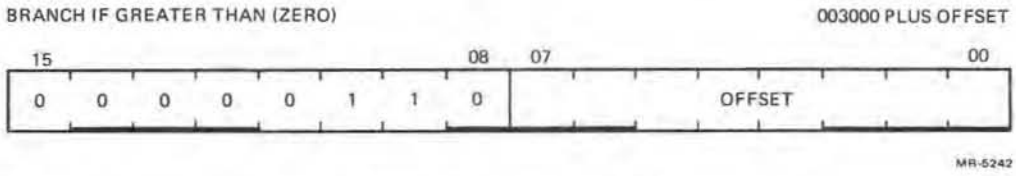


Figure 6-82 BGT

Operation: PC←PC+ (2 X offset) if Z V (N⊻V) = 0

Condition Codes: Unaffected

Description: Operation of BGT is similar to BGE, except BGT will not cause a branch on a zero result.

BLE

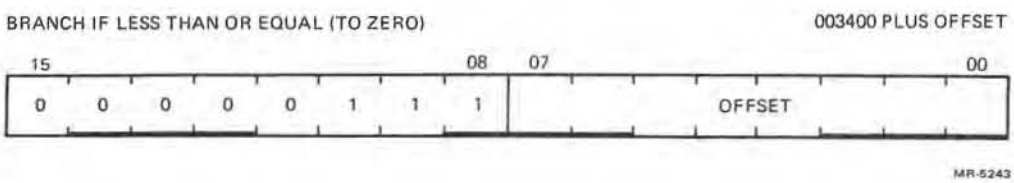


Figure 6-83 BLE

Operation: PC←PC+ (2 X offset) if Z V (N⊻V) = 1

Condition Codes: Unaffected

Description: Operation is similar to BLT but in addition will cause a branch if the result of the previous operation was zero.

6.3.5.3 Unsigned Conditional Branches -- The unsigned conditional Branches provide a means for testing the result of comparison operations in which the operands are considered as unsigned values.

BHI

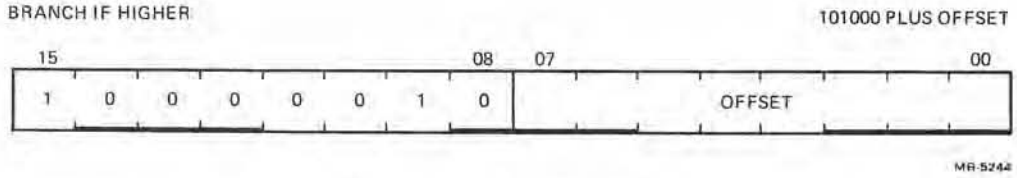


Figure 6-84 BHI

Operation: PC←PC+ (2 X offset) if C = 0 and Z = 0

Condition Codes: Unaffected

Description: Causes a branch if the previous operation caused neither a carry nor a zero result. This will happen in comparison (CMP) operations as long as the source has a higher unsigned value than the destination.

BLOS

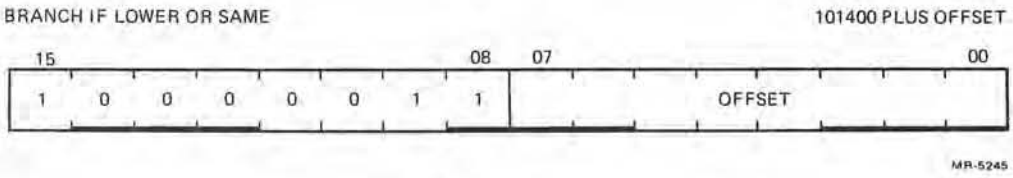


Figure 6-85 BLOS

Operation: PC←PC+ (2 X offset) if C V Z = 1

Condition Codes: Unaffected

Description: Causes a branch if the previous operation caused either a carry or a zero result. BLOS is the complementary operation to BHI. The branch will occur in comparison operations as long as the source is equal to, or has a lower unsigned value than the destination.

BHIS

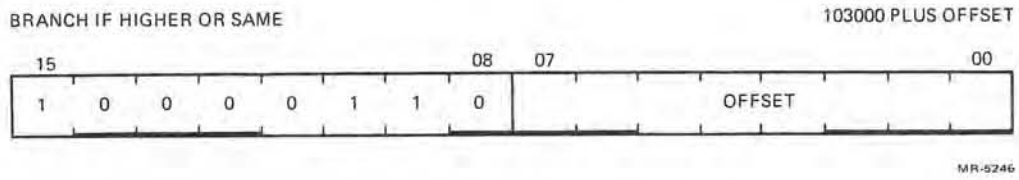


Figure 6-86 BHIS

Operation: PC←PC+ (2 X offset) if C = 0

Condition Codes: Unaffected

Description: BHIS is the same instruction as BCC. This mnemonic included only for convenience.

BLO

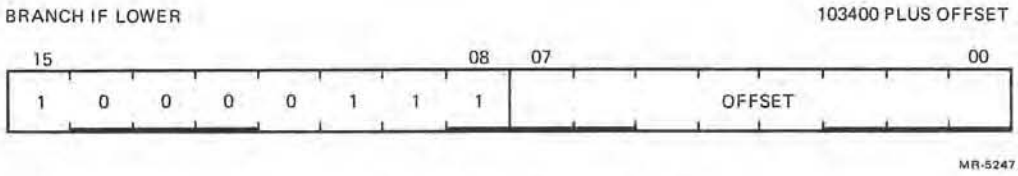


Figure 6-87 BLO

Operation: PC←PC+ (2 X offset) if C = 1

Condition Codes: Unaffected

Description: BLO is same instruction as BCS. This mnemonic is included only for convenience.

6.3.5.4 Jump & Subroutine Instructions -- The subroutine call in the DCT11-AA provides for automatic nesting of subroutines, reentrancy, and multiple entry points. Subroutines may call other subroutines (or indeed themselves) to any level of nesting without making special provision for storage of return addresses at each level of subroutine call. The subroutine calling mechanism does not modify any fixed location in memory, thus providing for reentrancy. This allows one copy of a subroutine to be shared among several interrupting processes.

JMP

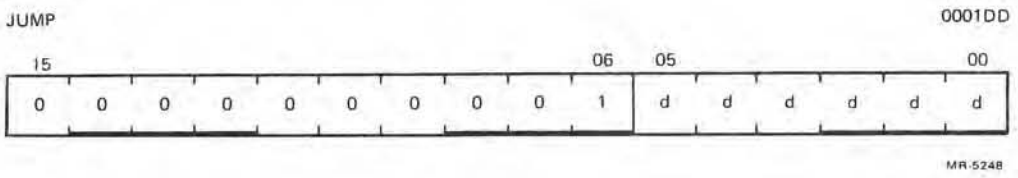


Figure 6-88 JMP

Operation: PC←(dst)

Condition Codes: Unaffected

Description: JMP provides more flexible program branching than provided with the branch instructions. Control may be transferred to any location in memory (no range limitation) and can be accomplished with the full flexibility of the addressing modes, with the exception of register mode 0. Execution of a jump with Mode 0 will cause an "illegal instruction" condition, and will cause the CPU to trap to vector address four. (Program control cannot be transferred to a register.) Register deferred mode is legal and will cause program control to be transferred to the address held in the specified register. Note that instructions are word data and must therefore be fetched from an even-numbered address.

Deferred index mode JMP instructions permit transfer of control to the address contained in a selectable element of a table of dispatch vectors.

Example:

```
First:  JMP FIRST        ;Transfers to First
        .....
        .....
        JMP @LIST        ;Transfers to location pointed
                          to at LIST
        .....
List:   FIRST            ;pointer to FIRST

        JMP @(SP)+       ;Transfer to location pointed
                          to by the top of the stack, and
                          remove the pointer from the
                          stack.
```

JSR

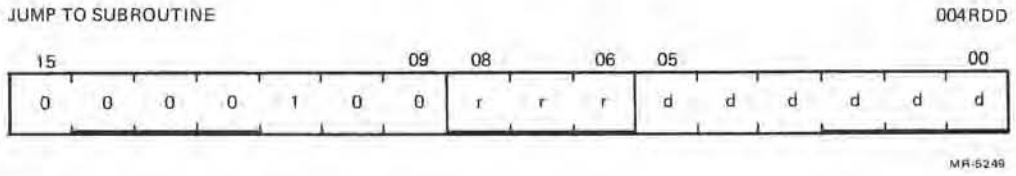


Figure 6-89 JSR

Operation: (tmp)←(dst) (tmp is an internal processor register)

↓(SP)←reg (Push reg contents onto processor stack)

reg←PC (PC holds location following JSR; this address now put in reg)

Operation: PC←(dst) (PC now points to subroutine destination)

Description: In execution of the JSR, the old contents of the specified register (the "LINKAGE POINTER") are automatically pushed onto the processor stack and new linkage information placed in the register. Thus subroutines nested within subroutines to any depth may all be called with the same linkage register. There is no need either to plan the maximum depth at which any particular subroutine will be called or to include instructions in each routine to save and restore the linkage pointer. Further, since all linkages are saved in a reentrant manner on the processor stack execution of a subroutine may be interrupted, the same subroutine reentered and executed by an interrupt service routine. Execution of the initial subroutine can then be resumed when other requests are satisfied. This process (called nesting) can proceed to any level.

A subroutine called with a JSR reg,dst instruction can access the arguments following the call with either autoincrement addressing, (reg) +, (if arguments are accessed sequentially) or by indexed addressing, X(reg), (if accessed in random order). These addressing modes may also be deferred, @(reg)+ and @X(reg) if the parameters are operand addresses rather than the operands themselves.

JSR PC, dst is a special case of the DCT11-AA subroutine call suitable for subroutine calls that transmit parameters through the general registers. The SP and the PC are the only registers that may be modified by this call.

Another special case of the JSR instruction is JSR PC, @(SP) + which exchanges the top element of the processor stack and the contents of the program counter. Use of this instruction allows two routines to swap program control and resume operation when recalled where they left off. Such routines are called "co-routines."

Return from a subroutine is done by the RTS instruction. RTS reg loads the contents of reg into the PC and pops the top element of the processor stack into the specified register.

Example:

```
                                        R5          R6      R7
SBCALL:         JSR R5,SBR              #1          n       SBCALL
SBCALL+4:       ARG 1
                ARG 2
                .
                .
SBCALL+2+2M:    ARG M
CONT:           Next Instruction        #1          n       CONT
                .
                .
SBR:            MOV (R5)+,dst 1         SBCALL+4    n-2     SBR
                MOV (R5)+,dst 2
                .
                MOV (R5)+,dst M         SBCALL+2+2M
                Other Instructions      CONT
EXIT:           RTS R5                  CONT        n-2     EXIT
```

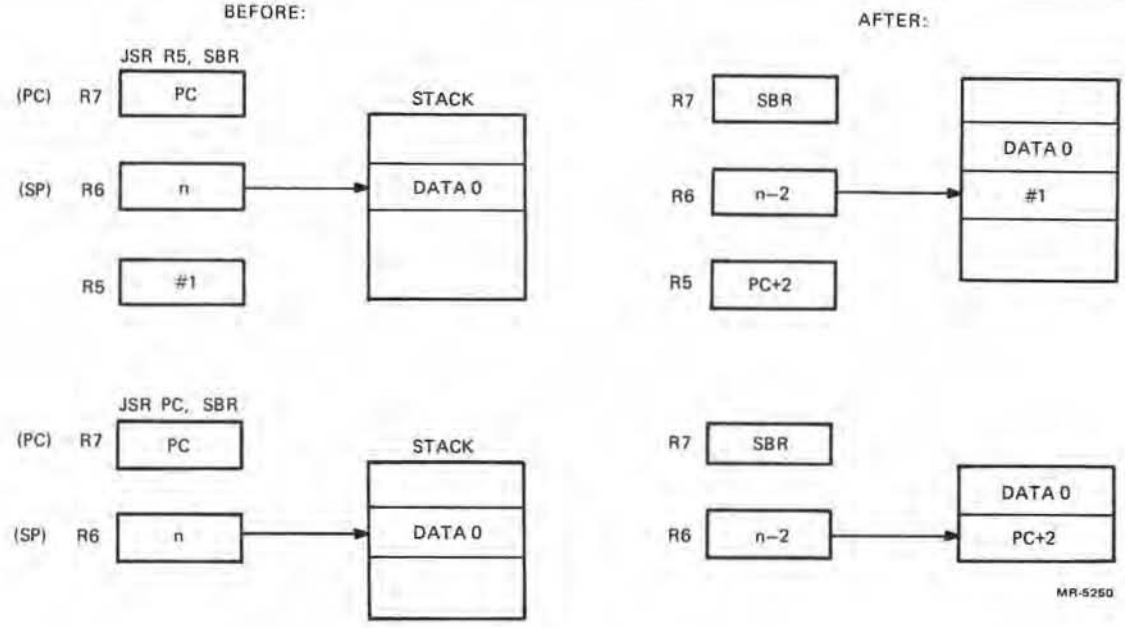


Figure 6-90 JSR Example

RTS

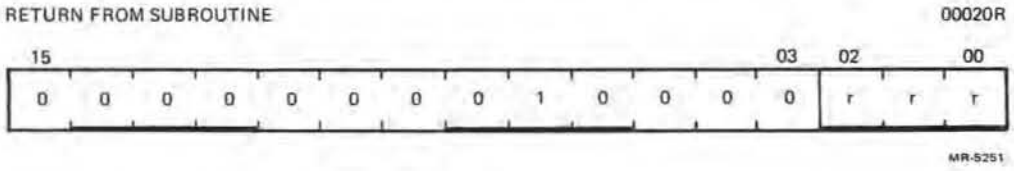


Figure 6-91 RTS

Operation: PC ← (reg)
(reg) ← (SP) ↑

Description: Loads contents of register into PC and pops the top element of the processor stack into the specified register.
Return from a non-reentrant subroutine is typically made through the same register that was used in its call. Thus, a subroutine called with a JSR PC, dst exists with a RTS PC and a subroutine called with a JSR R5, dst, may pick up parameters with addressing modes (R5) +, X(R5), or @X(R5) and finally exists, with an RTS R5.

Example: RTS R5

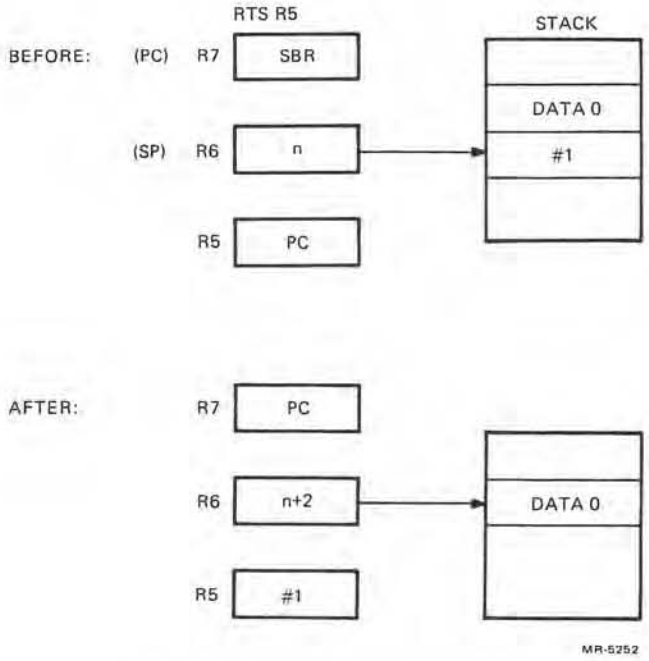


Figure 6-92 RTS Example

SOB

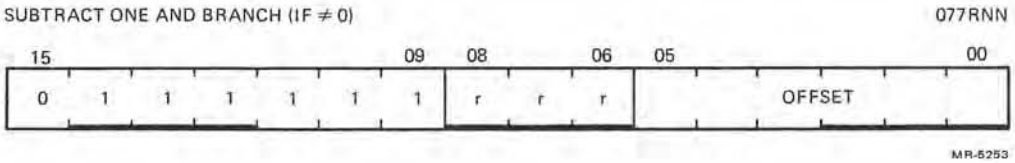


Figure 6-93 SOB

Operation: (R) ← (R) -1; it this result ≠ 0 then PC ← PC -(2 X offset) if (R) = 0; PC ← PC

Condition Codes: Unaffected

Description: The register is decremented. If it is not equal to zero, twice the offset is subtracted from the PC (now pointing to the following word). The offset is interpreted as a sixbit positive number. This instruction provides a fast, efficient method of loop control. Assembler syntax is:

SOB R,A

where A is the address to which transfer is to be made if the decremented R is not equal to 0. Note that the SOB instruction cannot be used to transfer control in the forward direction.

6.3.5.5 **Traps** -- Trap instructions provide for calls to emulators, I/O monitors, debugging packages, and user-defined interpreters. A trap is effectively an interrupt generated by software. When a trap occurs the contents of the current Program Counter (PC) and processor Status Word (PS) are pushed onto the processor stack and replaced by the contents of a two-word trap vector containing a new PC and new PS. The return sequence from a trap involves executing an RTI or RTT instruction which restores the old PC and old PS by popping them from the stack. Trap instruction vectors are located at permanently assigned fixed addresses.

EMT

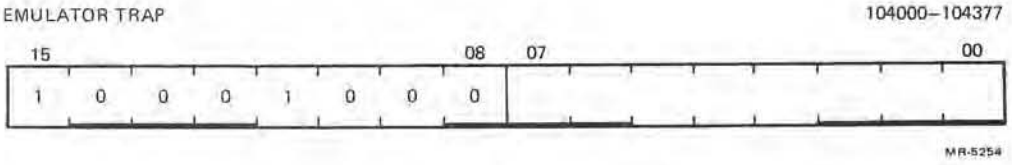


Figure 6-94 EMT

Operation:
↓(SP)←PS
↓(SP)←PC
PC←(30)
PS←(32)

Condition Codes:
N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description: All operation codes from 104000 to 104377 are EMT instructions and may be used to transmit information to the emulating routine (e.g., function to be performed). The trap vector for EMT is at address 30. The new PC is taken from the word at address 30; the new processor status (PS) is taken from the word at address 32.

Caution: EMT is used frequently by DEC system software and is therefore not recommended for general use.

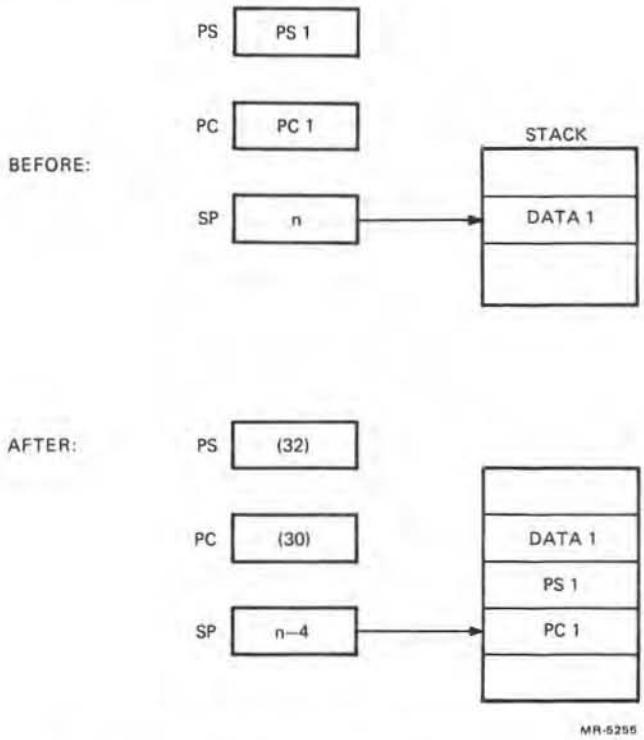


Figure 6-95 EMT Example

TRAP

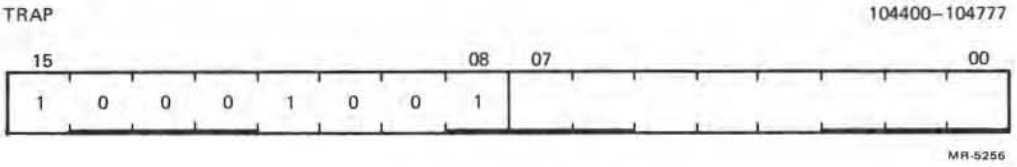


Figure 6-96 TRAP

Operation:

ψ(SP)←PS
ψ(SP)←PC
PC←(34)
PS←(36)

Condition Codes:

N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description:

Operation codes from 104400 to 104777 are TRAP instructions. TRAPs and EMTs are identical in operation, except that the trap vector for TRAP is at address 34.

Note: Since DEC software makes frequent use of EMT, the TRAP instruction is recommended for general use.

BPT

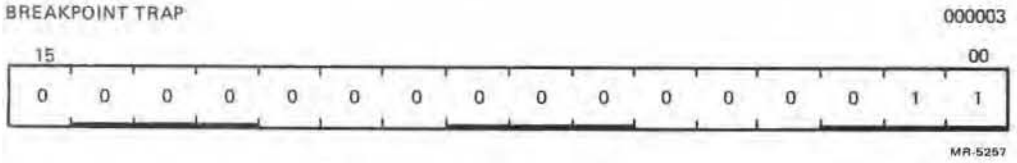


Figure 6-97 BPT

Operation:

$\downarrow$(SP)$\leftarrow$PS
$\downarrow$(SP)$\leftarrow$PC
PC$\leftarrow$(14)
PS$\leftarrow$(16)

Condition Codes:

N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description:

Performs a trap sequence with a trap vector address of 14. Used to call debugging aids. The user is cautioned against employing code 000003 in programs run under these debugging aids.

(No information is transmitted in the low byte.)

IOT

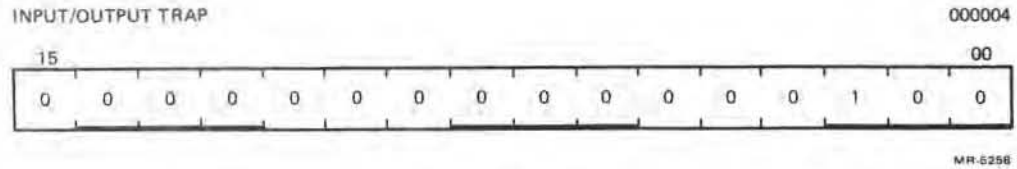


Figure 6-98 IOT

Operation:

$\downarrow$(SP)$\leftarrow$PS
$\downarrow$(SP)$\leftarrow$PC
PC$\leftarrow$(20)
PS$\leftarrow$(22)

Condition Codes:

N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description:

Performs a trap sequence with a trap vector address of 20.

(No information is transmitted in the low byte.)

RTI

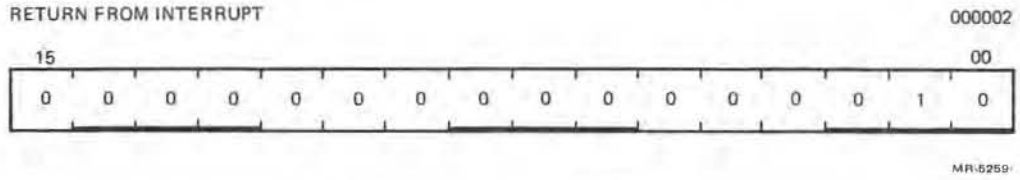


Figure 6-99 RTI

Operation: PC←(SP)↑
PS←(SP)↑

Condition Codes: N: loaded from processor stack
Z: loaded from processor stack
V: loaded from processor stack
C: loaded from processor stack

Description: Used to exit from an interrupt or TRAP service routine. The PC and PS are restored (popped) from the processor stack. If a trace trap is pending, the first instruction after RTI will not be executed prior to the next T traps.

RTT

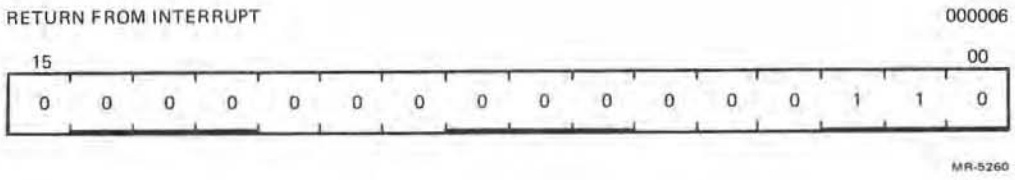


Figure 6-100 RTT

Operation: PC←(SP)↑
PS←(SP)↑

Condition Codes: N: loaded from processor stack
Z: loaded from processor stack
V: loaded from processor stack
C: loaded from processor stack

Description: Operation is the same as RTI except that it inhibits a trace trap while RTI permits trace trap. If new PS has T bit set, trap will occur after execution of first instruction after RTT.

6.3.5.6 Reserved Instruction Traps -- These are caused by attempts to execute instruction codes reserved for future processor expansion (reserved instructions) or instructions with illegal addressing modes (illegal instructions). Order codes not corresponding to any of the instructions described are considered to be reserved instructions. JMP and JSR with register mode destinations are illegal instructions, and trap to vector address four. Reserved instructions trap to vector address.

6.3.5.7 Halt Interrupt -- This is caused by the -HALT line (AI<7>). The -HALT interrupt saves the PC and PSW and goes to the restart address with PS = 340.

6.3.5.8 Trace Trap -- Trace Trap is enabled by bit four of the PS and causes processor traps at the end of instruction execution. The instruction that is executed after the instruction that set the T-bit will proceed to completion and then trap through the trap vector at address 14. Note that the trace trap is a system debugging aid and is transparent to the general programmer.

6.3.5.9 Power Failure Interrupt -- Occurs when -PF line (AI<6>) is asserted. Vector for power failure is location 24 and 26. Trap will occur if an RTI instruction is executed in power fail service routine.

6.3.5.10 CP<3:0> Interrupts -- Refer to paragraph 1.5.3.

NOTE

Bit four of the PS can only be set indirectly by executing a RTI or RTT instruction with the desired PS on the stack.

6.3.5.11 Special Cases T-bit -- The following are special cases of the T-bit.

NOTE

The traced instruction is the instruction after the one that set the T-bit.

1. An instruction that cleared the T-bit -- Upon fetching the traced instruction, an internal flag, the trace flag, was set. The trap will still occur at the end of execution of this instruction. The status word on the stack, however, will have a clear T-bit.

2. An instruction that set the T-bit -- Since the T-bit was already set, setting it again has no effect. The trap will occur.

3. An instruction that caused an Instruction Trap -- The instruction trap is performed and the entire routine for the service trap is executed. If the service routine exists with an RTI or in any other way restores the stacked status word, the T-bit is set again, the instruction following the traced instruction is executed and, unless it is one of the special cases noted previously, a trace trap occurs.

4. Interrupt Trap Priorities -- In case of multiple processor trap and interrupt conditions, occurring simultaneously, the following order of priorities is observed (from high to low):

Halt Line
Trace Trap
Power Fail Trap
CP<3:0> Interrupt Request
Instruction Traps

6.3.6 Miscellaneous Instructions

HALT

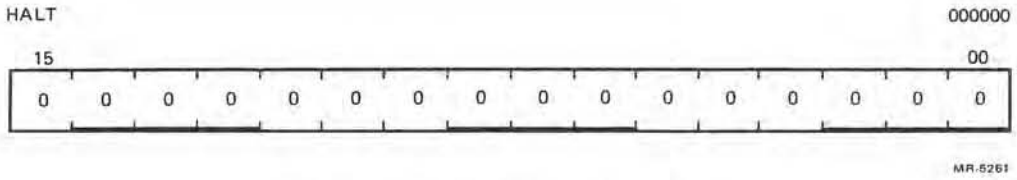


Figure 6-101 HALT

Operation:

↓(SP)←PS
↓(SP)←PC
PC←restart address
PS←340

Condition Codes: Unaffected

Description: The processor goes to the restart address after placing the current PC and PS on the stack. PS is initialized to 340.

WAIT

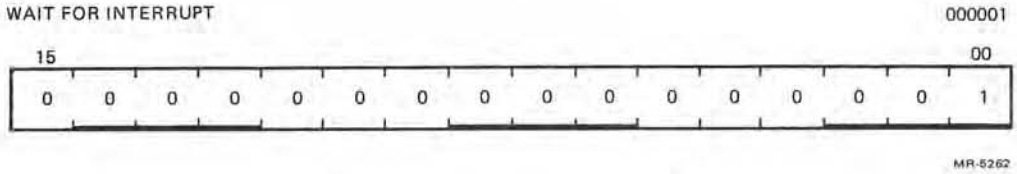


Figure 6-102 WAIT

Condition Codes: Unaffected

Description: In WAIT, as in all instructions, the PC points to the next instruction following the WAIT instruction. Thus when an interrupt causes the PC and PS to be pushed onto the processor stack, the address of the next instruction following the WAIT is saved. The exit from the interrupt routine (i.e., execution of an RTI instruction) will cause resumption of the interrupted process at the instruction following the WAIT.

RESET

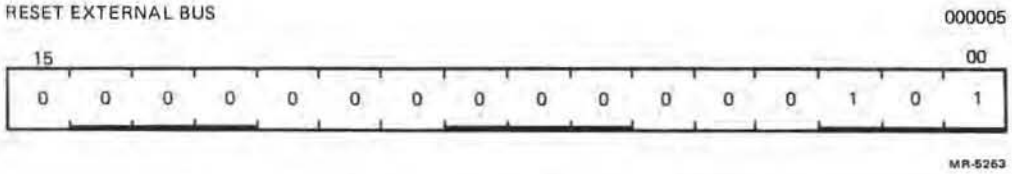


Figure 6-103 RESET

Condition Codes: Unaffected

Description: The -BCLR line is asserted and the mode register is loaded. -BCLR is negated and an ASPI transaction takes place. PC, PS, and R0:R5 are not affected.

MFPT

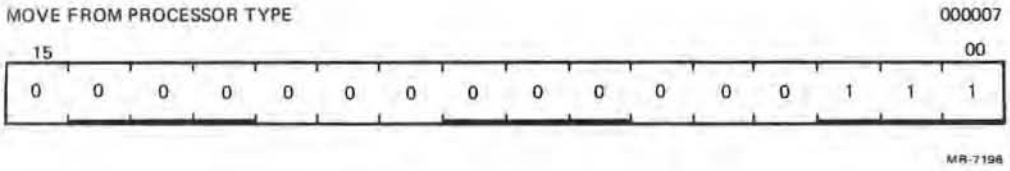


Figure 6-104 MFPT

Operation: R0←4

Condition Codes: Unaffected

Description: The number four is placed in R0 indicating to the system software that the processor type is DCT11-AA.

6.3.7 Condition Code Operators

CLN SEN
CLZ SEZ
CLV SEV
CLC SEC
CCC SCC

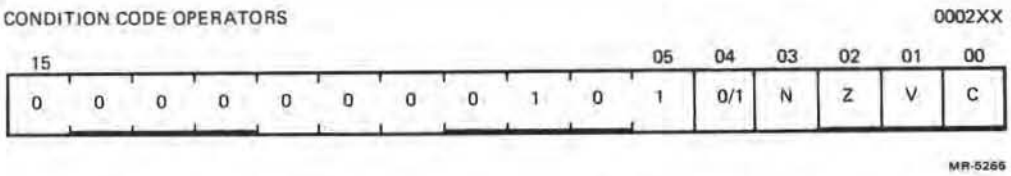


Figure 6-105 Condition Code Operators

Description: Set and clear conditon code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (Bits 0--3) are modified according to the sense of bit four, the set/clear bit of the operator; i.e., set the bit specified by bit zero, one, two, or three, if bit four is a one. Clear corresponding bits if bit 4 = 0.

Mnemonic	Operation	OP Code
CLC	Clear C	000241
CLV	Clear V	000242
CLZ	Clear Z	000244
CLN	Clear N	000250
SEC	Set C	000261
SEV	Set V	000262
SEZ	Set Z	000264
SEN	Set N	000270
SCC	Set all CCs	000277
CCC	Clear all CCs	000257
	Clear V and C	000243
NOP	No Operation	000240

Combinations of the above set or clear operations may be ORed together to form combined instructions.

APPENDIX A

Table A-1 Interrupt Decode

-CP<3> (AI<1>)	-CP<2> (AI<2>)	-CP<1> (AI<3>)	-CP<0> (AI<4>)	Priority Level	Vector Address	
X	X	X	X	8	-	-HALT*
X	X	X	X	8	24	-PF
L	L	L	L	7	140	
L	L	L	H	7	144	
L	L	H	L	7	150	
L	L	H	H	7	154	
L	H	L	L	6	100	
L	H	L	H	6	104	
L	H	H	L	6	110	
L	H	H	H	6	114	
H	L	L	L	5	120	
H	L	L	H	5	124	
H	L	H	L	5	130	
H	L	H	H	5	134	
H	H	L	L	4	60	
H	H	L	H	4	64	
H	H	H	L	4	70	
H	H	H	H	NO ACTION		

* PC is loaded with the restart address. PSW = 340.

Table A-2 D.C. Characteristics

Absolute Maximum Ratings

Pin Voltages	-0.5 to +7 V
Storage temperature range	-55 to +125° C (-67 to 257° F)
Max power dissipation	1.0 watt (TJ = 0o C) (32° F)
Chip ambient temperature operating range	0 to 70° C (32 to 158° F)

NOTE

Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Static Characteristics

(TA = 0 to 70° C (32 to 158° F), VCC = 5.0 V $\pm$ 5%, VSS = 0 V)

Table A-2 D.C. Characteristics (Cont)

Symbol	Parameter/Pins	Min	Max	Units	Comments and Conditions
IIL	(Low Input) Three-state leakage current on DAL<15:0>		-50	uA	VIN = 0.4 V
IIL	(High Input) Three-state leakage current on DAL<15:0>		+10	uA	VIN = VCC max
IIH	(Min) Input current for internal pullups on AI<7:0>, DAL<15:7,2:0>, READY	-0.1		mA	VIN = 2.4 V
IIH	(Max) Input Current for internal pullups on AI<7:07>, READY, DAL<15:7,2:0>		-0.1	mA	VIN = 0.4 V
ICC	Power Supply Current on VCC		190	mA	TCYC = 400 ns,
IXLIH	Input High Current on XTL1		+700	uA	2.4<VIN<VCC, XTL0 grounded
IXLIL	Input Low Current on XTL1		-6.4	mA	-0.5<VIN<+0.8V, XTL0 grounded
VIH	Input High Voltage on READY,DAL<15:0>,AI<7:0>	2	VCC	V	
VIL	Input Low Voltage on READY,DAL<15:0> AI<7:0>	-0.5*	+0.8	V	
VOH	Output High Voltage for DAL<15:0>,COUT, PI,SEL1,SEL0	2.4		V	IOH = 700 uA
VOHA	Output High Voltage for AI<7:0>	2.6		V	IOH = -700 uA
VOHB	Output High Voltage for BCLR	2.2		V	IOH = -700 uA terminated with 1K resistor to VSS
VOHC	Output High Voltage for -RAS, -CAS, R/-WLB and R/-WHB	2.8		V	IOH = -700 uA
VOL	Output Low Voltage for DAL<15:0>, AI<7:0>, COUT, PI, SEL1, SEL0, -BCLR, -RAS, -CAS, R/-WHB R/-WLB	0.0	0.4	V	IOL = 3.2 mA
VILPUP	Input Low Level for PUP	-0.5*	+0.8	V	
VIHPUP	Input High Level for PUP	1.6	VCC	V	
VHY	Hysteresis, PUP	0.6		V	
CIN	Input Capacitance for READY,DAL<15:0>, AI<7:0>		10	pF	
COUT	Output Capacitance for three-state load calculation on DAL<15:0>, AI<7:0>, COUT, PI, SEL1, SEL0, BCLR, RAS, CAS, R/-WHB, R/-WLB		20	pF	

* -0.5 V on input pins allows for ringing on unterminated lines.

Table A-3 Sequences of Transactions

Transaction:	R - Read W - Write Ref - Refresh (replaced by N in static modes)	I - Iack D - DMA A - ASPI N - Busnop	Note: R-W means read modify write (- indicates indivisible)

Instruction	16 Bit	8 Bit	Sequence of Transaction
CLR R0	X		R Ref N
		X	R-R Ref N
CLR (R0) or MOV R0, (R1) or MOV R0, (R1)+	X		R Ref R-W
		X	R-R Ref R-R-W-W
MOV R0, -(R1)	X		R Ref N R-W
		X	R-R Ref N R-R-W-W
MOV R0, @X(R1)	X		R Ref R N R-W Ref[1]
		X	R-R Ref R-R n R-R-W-W Ref[1]
MTSP R0	X		R Ref N N N N N
		X	R-R Ref N N N N N
JMP (R0)	X		R Ref N N
		X	R-R Ref N N
JSR R0, (R1)	X		R Ref N N N N N
		X	R-R Ref N N W-W N N
WAIT	X		R [Ref N A][2]
		X	R-R [Ref N A][2]
HLT	X		R Ref N N W Ref N W N N A N
		X	R-R Ref N N W-W Ref N W-W N N A N
EMT	X		R Ref N N W Ref N W R R N
		X	R-R Ref N N W-W Ref N W-W R-R R N
RESET	X		R Ref N N N N [N N N][3] N N N A N
		X	R-R Ref N N N N [N N N][3] N N N A N
Interrupt Sequence	X		...R-W[4] | I N N[5] N W Ref N W R R N| R[6]...
		X	...R-R-W-W[5] |[6] I N N[5] N W-W Ref N W-W R-R R N| R-R[6]...
DMA Sequence	X		...R-W[7] D R...
		X	...R-R-W-W[7] D R-R...

Table A-3 Sequence of Transactions Cont'd

1 Missing transaction in static mode.
2 Sequence repeated until interrupt request.
3 Sequence repeated nine times (-BCLR is low during this time.)
4 Last transactions of instruction in which interrupt is posted.
5 Transaction missing if internal vector is used.
6 Fetch of first instrution of interrupt service routine.
7 R-W (R-R-W-W) are indivisible.

Table A-4 Signal and Pin Utilization 16-Bit Mode

Pin	Pin Name	Signal Names							
		Static	4K/16K			64K			
Data Address Lines									
1-7&9 10-17	DAL<15:8> DAL<7:0>	DAL<15:8> DAL<7:0>	DAL<15:8> DAL<7:0>			DAL<15:8> DAL<7:0>			
Address Interrupt Lines									
			-RAS	-CAS	PI	-RAS	-CAS	PI	
32	AI<0>	-DMR	FET[1]	A14	-DMR	A15	A14	-DMR	
33	AI<1>	-CP<3>	A1	A2	-CP<3>	A1	A2	-CP<3>	
34	AI<2>	-CP<2>	A3	A4	-CP<2>	A3	A4	-CP<2>	
35	AI<3>	-CP<1>	A5	A6	-CP<1>	A5	A6	-CP<1>	
36	AI<4>	-CP<0>	A7	A8	-CP<0>	A7	A8	-CP<0>	
37	AI<5>	-VEC	A9	A10	-VEC	A9	A10	-VEC	
38	AI<6>	-PF	A11	A12	-PF	A11	A12	-PF	
39	AI<7>	-HALT	A13	A14	-HALT	A13	A12	-HALT	
Control Signals									
24	SEL1[2]	Iack+DMG	Iack+DMG			Iack+DMG			
25	SEL0[2]	FET+DMG	REF+DMG			FET+DMG			
26	READY	READY	READY			READY			
27	R/-WHB	R/-WHB	R/-WHB			R/-WHB			
28	R/-WLB	R/-WLB	R/-WLB			R/-WLB			
29	-RAS	-RAS	-RAS			-RAS			
30	-CAS	-CAS	-CAS			-CAS			
31	PI	PI	PI			PI			
Miscellaneous Signals									
18	-BCLR	-BCLR	-BCLR			-BCLR			
19	PUP	PUP	PUP			PUP			
21	COUT	COUT	COUT			COUT			
22	XTL1	XTL1	XTL1			XTL1			
23	XTL0	XTL0	XTL0			XTL0			
Power Pins									
8	BGND	BGND	BGND			BGND			
20	GND	GND	GND			GND			
40	V_{cc}	V_{cc}	V_{cc}			V_{cc}			

Table A-4 Signal and Pin Utilization 16-Bit Mode Cont'd

NOTES

1 During -RAS, AI<0> is used to indicate a fetch operation in progress. During refresh, AI<0> is the output of the refresh counter at -RAS time.

2 SEL<1> and SEL<0> are encoded refer to Tables 3-4 and 3-5.

Table A-5 Signal and Pin Utilization 8-Bit Mode

Pin	Pin Name	Signal Names							
		Static	4K/16K			64K			
Data Address Lines									
1-7&9	DAL<15:8>	SAL<15:8>	SAL<15:8>			SAL<15:8>			
10-17	DAL<7:0>	DAL<7:0>	DAL<7:0>			DAL<7:0>			
Address Interrupt Lines									
			-RAS	-CAS	PI	-RAS	-CAS	PI	
32	AI<0>	-DMR	FET[1]	A14	-DMR	A15	A14	-DMR	
33	AI<1>	-CP<3>	A1	A2	-CP<3>	A1	A2	-CP<3>	
34	AI<2>	-CP<2>	A3	A4	-CP<2>	A3	A4	-CP<2>	
35	AI<3>	-CP<1>	A5	A6	-CP<1>	A5	A6	-CP<1>	
36	AI<4>	-CP<0>	A7	A8	-CP<0>	A7	A8	-CP<0>	
37	AI<5>	-VEC	A9	A10	-VEC	A9	A10	-VEC	
38	AI<6>	-PF	A11	A0	-PF	A11	A0	-PF	
39	AI<7>	-HALT	A13	A12	-HALT	A13	A12	-HALT	
Control Signals									
24	SEL1[2]	Iack+DMG	Iack+DMG			Iack+DMG			
25	SEL0[2]	FET+DMG	REF+DMG			FET+DMG			
26	READY	READY	READY			READY			
27	R/-WHB	-RD	-RD			-RD			
28	R/-WLB	-WT	-WT			-WT			
29	-RAS	-RAS	-RAS			-RAS			
30	-CAS	-CAS	-CAS			-CAS			
31	PI	PI	PI			PI			
Miscellaneous Signals									
18	-BCLR	-BCLR	-BCLR			-BCLR			
19	PUP	PUP	PUP			PUP			
21	COUT	COUT	COUT			COUT			
22	XTL1	XTL1	XTL1			XTL1			
23	XTL0	XTL0	XTL0			XTL0			
Power Pins									
8	BGND	BGND	BGND			BGND			
20	GND	GND	GND			GND			
40	V_{cc}	V_{cc}	V_{cc}			V_{cc}			

Table A-5 Signal and Pin Utilization 16-Bit Mode Cont'd

NOTES

1 During -RAS, AI<0> is used to indicate a fetch operation in progress. During refresh, AI<0> is the output of the refresh counter at -RAS time.

2 SEL<1> and SEL<0> are encoded refer to Tables 3-4 and 3-5.

Table A-6 Dynamic Addressing Scheme

Mode	Memory Chip	Address*	AI Used
4/16K	4K X 1	A1--A12	<6:1>
4/16K	16K X 1	A1--A14	<7:1>
64K	64K X 1	A1--A15	<7:0>

* Address lines necessary to address all bits in each chip

Table A-7 SEL<1:0> Functions in Static Mode or Dynamic 64K Mode

SEL<1>	SEL<0>	Function
L	L	read, write, ASPI, or busnop
L	H	fetch (PDP-11 instruction fetch)
H	L	Iack (interrupt acknowledge)
H	H	DMG (direct memory grant)

Table A-8 SEL<1:0> Functions in Dynamic 4K/16K Mode

SEL<1>	SEL<0>	Function
L	L	read, write, ASPI, or busnop
L	H	refresh
H	L	Iack (interrupt acknowledge)
H	H	DMG (direct memory grant)

Table A-9 AI Functions

Transaction	@ -RAS (L.E.) Output	@ -CAS (L.E.) Output	@ PI (T.E.) Input
Read (static)	*	*	interrupt/DMR
Write (static)	*	*	DMR
Read (dynamic)	row address	column address	interrupt/DMR
Write (dynamic)	row address	column address	DMR
Refresh	row address	N/A	N/A
DMA	*	*	DMR
ASPI	N/A	*	interrupt/DMR

N/A - not applicable
* - internal low current passive pullups

Table A-10 Control Signals for Each Transaction

TRANSACTION	-RAS	-CAS	PI	R/-WHB	R/-WLB	SEL0	SEL1
Read	*	*	*	X			
Fetch	*	*	*	X		1	
Write	*	*	*	-	*		
Refresh	*					2	
Iack	*						*
DMA	*	*	*	3S	3S	*	*
ASPI		*	*				
Busnop							

*	Signal asserted during the transaction
1	Static modes and dynamic 64K
2	Dynamic modes 4K/16K
X	Signal asserted during 8-bit mode only
-	Signal asserted during 16-bit mode only
3S	Three-state

Table A-11 Data Bus for Each Transaction

TRANSACTION	DAL LOW BYTE	DAL HIGH BYTE	AI
Read		X	
Fetch		X	
Write	*	X	
Refresh	*	*	*
Iack		*	1
DMA	3S	3S	3S
ASPI			
Busnop	*	*	1

X Lines driven after address portion of transaction 8-bit mode
* Lines driven after address portion of transaction
1 Dynamic modes only
3S Three-state

Table A-12 Summary of DCT11-AA Instructions

SINGLE OPERAND

Mmemonic	Op Code	Instruction	dst Result	N	Z	V	C
General							
CLR(B)	050DD	clear	0	0	1	0	0
COM(B)	051DD	complement (1's)	d	*	*	0	1
INC(B)	052DD	increment	d + 1	*	*	*	-
DEC(B)	053DD	decrement	d - 1	*	*	*	-
NEG(B)	054DD	negate (2's compl)	-d	*	*	*	*
TST(B)	057DD	test	d	*	*	0	0
Rotate & Shift							
ROR(B)	060DD	rotate right	→ C, d	*	*	*	*
ROL(B)	061DD	rotate left	C, d ←	*	*	*	*
ASR(B)	062DD	arith shift right	d/2	*	*	*	*
ASL(B)	063DD	arith shift left	2d	*	*	*	*
SWAB	0003DD	swap bytes		*	*	0	0
Multiple Precision							
ADC(B)	055DD	add carry	d + c	*	*	*	*
SBC(B)	056DD	subtract carry	d - c	*	*	*	*
SXT	0067DD	sign extend	0 or -1	-	*	0	-
Processor Status (PS) Operators							
MFPS	1067DD	move byte from PS	d ← PS	*	*	0	-
MTPS	1064SS	move byte to PS	PS ← s	*	*	*	*

DOUBLE OPERAND

General							
MOV(B)	1SSDD	move	d ← s	*	*	0	-
CMP(B)	2SSDD	compare	s - d	*	*	*	*
ADD	06SSDD	add	d ← s + d	*	*	*	*
SUB	16SSDD	subtract	d ← d - s	*	*	*	*
Logical							
BIT(B)	3SSDD	bit test (AND)	s d	*	*	0	-
BIC(B)	4SSDD	bit clear	d ← (s) d	*	*	0	-
BIS(B)	5SSDD	bit set (OR)	d ← s v d	*	*	0	-
XOR	074RDD	exclusive (OR)	d ← r v d	*	*	0	-

BRANCH

Mnemonic	Base Code	Instruction	Branch Condition
Branches			
BR	000400	branch (unconditional)	(always)
BNE	001000	br if not equal (to 0)	$\neq 0$ Z = 0
BEQ	001400	br if equal (to 0)	= 0 Z = 1
BPL	100000	branch if plus	+ N = 0
BMI	100400	branch if minus	− N = 1
BVC	102000	br if overflow is clear	V = 0
BVS	102400	br if overflow is set	V = 1
BCC	103000	br if carry is clear	C = 0
BCS	103400	br if carry is set	C = 1
Signed Conditional Branches			
BGE	002000	br if greater or equal	≥ 0 N v V = 0
BLT	002400	br if less than (0)	$\overline{<} 0$ N v V = 1
BGT	003000	br if greater than (0)	> 0 Z v (N v V) = 0
BLE	003400	br if less or equal	$\leq$ Z v (N v V) = 1
Unsigned Conditional Branches			
BHI	101000	branch if higher	> C v Z = 0
BLOS	101400	branch if lower or same	$\leq$ C v Z = 1
BHIS	103000	branch if higher or same	$\geq$ C = 0
BLO	103400	branch if lower	$\overline{\geq}$ C = 1

JUMP & SUBROUTINE

Mnemonic	Op Code	Instruction	Notes
JMP	0001DD	jump	PC ← dst
JSR	004RDD	jump to subroutine	use same R
RTS	00020R	return from subroutine	use same R
SOB	077RNN	subtract 1 & br (if $\neq 0$)	R − 1, then if R $\neq$ 0: PC ← Updated PC − (2 X NN)
TRAP AND INTERRUPT			
EMT	104000	emulator trap	PC at 30, PS at 32
	to 104377	(not for general use)	
TRAP	104400	trap	PC at 34, PS at 36
	to 104777		
BPT	000003	breakpoint trap	PC at 14, PS at 16
IOT	000004	input/output trap	PC at 20, PS at 22
RTI	000002	return from interrupt	
RTT	000006	return from interrupt	inhibit T bit trap

MISCELLANEOUS

Mnemonic	Op Code	Instruction
HALT	000000	halt
WAIT	000001	wait for interrupt
RESET	000005	reset external bus
MFPT	000007	move from processor type
NOP	000240	(no operation)

CONDITION CODE OPERATORS

Mnemonic	Op Code	Instruction	N	Z	V	C
CLC	000241	clear C	-	-	-	0
CLV	000242	Clear V	-	-	0	-
CLZ	000244	clear Z	-	0	-	-
CLN	000250	clear N	0	-	-	-
CCC	000257	clear all cc bits	0	0	0	0
SEC	000261	set C	-	-	-	1
SEV	000262	set V	-	-	1	-
SEZ	000264	set Z	-	1	-	-
SEN	000270	set N	1	-	-	-
SCC	000277	set all cc bits	1	1	1	1

Table A-13 Numerical OP Code List

Op Code	Mnemonic	Op Code	Mnemonic	Op Code	Mnemonic
00 00 00	HALT	00 53 DD	DEC	10 40 00	
00 00 01	WAIT	00 54 DD	NEG	thru	EMT
00 00 02	RTI	00 55 DD	ADC	10 43 77	
00 00 03	BPT	00 56 DD	SBC		
00 00 04	IOT	00 57 DD	TST	10 44 00	
00 00 05	RESET			thru	TRAP
00 00 06	RTT			10 47 77	
00 00 07	MFPT	00 60 DD	ROR		
00 00 77	(unused)	00 61 DD	ROL	10 50 DD	CLRB
		00 62 DD	ASR	10 51 DD	COMB
00 01 DD	JMP	00 63 DD		10 52 DD	INCB
00 02 OR	RST	00 67 DD	SXT	10 53 DD	DECB
				10 54 DD	NEGB
00 02 10		00 70 00		10 55 DD	ADCB
thru	reserved	thru	(unused)	10 56 DD	SBCB
00 02 27		00 77 77		10 57 DD	TSTB
00 02 40	NOP	01 SS DD	MOV	10 60 DD	RORB
		02 SS DD	CMP	10 61 DD	ROLB
00 02 41	cond.	03 SS DD	BIT	10 62 DD	ASRB
thru	codes	04 SS DD	BIC	10 63 DD	ASLB
00 02 77		05 SS DD	BIS	10 64 SS	MTPS
		06 SS DD	ADD	10 67 DD	MFPS
00 03 DD	SWAB				
		07 50 40		11 SS DD	MOVB
00 04 XXX	BR	thru	(unused)	12 SS DD	CMPB
00 10 XXX	BNE	07 67 77		13 SS DD	BITB
00 14 XXX	BEQ			14 SS DD	BICB
00 20 XXX	BGE	07 7R NN	SOB	15 SS DD	BISB
00 24 XXX	BLT			16 SS DD	SUB
00 30 XXX	BGT	10 00 XXX	BPL		
00 34 XXX	BLE	10 04 XXX	BMI	17 00 00	
		10 10 XXX	BHI	thru	reserved
00 4R DD	JSR	10 14 XXX	BLOS	17 77 77	
		10 20 XXX	BVC		
00 50 DD	CLR	10 24 XXX	BVS		
00 51 DD	COM	10 30 XXX	BCC, BHIS		
00 52 DD	INC	10 34 XXX	BCS, BLO		

Table A-14 Reserved Trap and Interrupt Vectors

000	Default vector = 0 for interrupting device failing to put vector out on DAL's
004	If mode 0 is the destination address in a JMP or JSR instruction a trap will occur to vector location 4
010	Illegal and reserved instruction
014	BPT instruction and T bit
020	IOT instruction
024	Power fail
030	EMT instruction
034	TRAP instruction

Table A-15 7-Bit ASCII Code

Octal	Char	Octal	Char	Octal	Char	Octal	Char
000	NUL	040	SP	100	@	140	
001	SOH	041	!	101	A	141	a
002	STX	042	"	102	B	142	b
003	ETX	043	#	103	C	143	c
004	EOT	044	$	104	D	144	d
005	ENQ	045	%	105	E	145	e
006	ACK	046	&	106	F	146	f
007	BEL	047	'	107	G	147	g
010	BS	050	(	110	H	150	h
011	HT	051	)	111	I	151	i
012	LF	052	*	112	J	152	j
013	VT	053	+	113	K	153	k
014	FF	054	,	114	L	154	l
015	CR	055	-	115	M	155	m
016	SO	056	.	116	N	156	n
017	SI	057	/	117	O	157	o
020	DLE	060	0	120	P	160	p
021	DC1	061	1	121	Q	161	q
022	DC2	062	2	122	R	162	r
023	DC3	063	3	123	S	163	s
024	DC4	064	4	124	T	164	t
025	NAK	065	5	125	U	165	u
026	SYN	066	6	126	V	166	v
027	ETB	067	7	127	W	167	w
030	CAN	070	8	130	X	170	x
031	EM	071	9	131	Y	171	y
032	SUB	072	:	132	Z	172	z
033	ESC	073	;	133	[	173	{
034	FS	074	<	134	\	174	|
035	GS	075	=	135	]	175	}
036	RS	076	>	136	^	176	~
037	US	077	?	137	_	177	DEL

Table A-16 Octal, Hex, Decimal Memory Addresses

Octal	K bytes	Hex	Decimal	Octal of High Byte 8-Bit Mode
200 000	64	10 000	65 536	N/A
177 000		F E00	65 024	376
176 000	63	F C00	64 512	374
175 000		F A00	64 000	372
174 000	62	F 800	63 488	370
173 000		F 600	62 976	366
172 000	61	F 400	62 464	364
171 000		F 200	61 952	362
170 000	60	F 000	61 440	360
167 000		E E00	60 928	356
166 000	59	E C00	60 416	354
165 000		E A00	59 904	352
164 000	58	E 800	59 392	350
163 000		E 600	58 880	346
162 000	57	E 400	58 368	344
161 000		E 200	57 856	342
160 000	56	E 000	57 344	340
150 000	52	D 000	53 248	320
140 000	48	C 000	49 152	300
130 000	44	B 000	45 056	260
120 000	40	A 000	40 960	240
110 000	36	9 000	36 864	220
100 000	32	8 000	32 768	200
70 000	28	7 000	28 672	160
60 000	24	6 000	24 576	140
50 000	20	5 000	20 480	120
40 000	16	4 000	16 384	100
30 000	12	3 000	12 288	60
20 000	8	2 000	8 192	40
10 000	4	1 000	4 096	20
7 000		E00	3 584	16
6 000	3	C00	3 072	14
5 000		A00	2 560	12
4 000	2	800	2 048	10
3 000		600	1 536	6
2 000	1	400	1 024	4
1 000		200	512	2
0		0	0	0

DCT11-AA Instruction Execution Times at Maximum Operating Frequency

The following pages contain charts that give execution times for all instructions executable by the DCT11-AA. The charts are organized to help calculate program execution times. To do such a calculation, first choose the system configuration and then outline that column in the charts. Use only those execution times listed in the outlined column. The possible system configurations are:

- 16-bit mode--REFRESH on
- 16-bit mode--REFRESH off
- 8-bit mode--REFRESH on
- 8-bit mode--REFRESH off.

It is possible for an instruction to have varying execution times when REFRESH is on. 8-bit mode REFRESH is done every instruction cycle and the REFRESH cycle adds a small increment of time to the machine cycle. Addressing mode 5, 6, or 7, I/O and trap (2 occurances) also add time. This is why MIN and MAX execution times are given in "REFRESH on" configurations (in 16 bit mode REFRESH is done every other). Program execution times can be calculated for "REFRESH on" configurations by summing the average of the MIN and Max execution times.

The following notes apply to all instruction execution charts:

- All times are in microseconds
- Add 0.4 us for every -READY pulse during an I/O transaction.
- Operating frequency is 7.5 MHz. To calculate instruction execution times (IET) at different operating frequencies use the following formula:

- IET(fOP) = (7.5 MHz/fOP) * IET(7.5)

  Where:

  IET(fOP) = Instruction Execution Time for the new frequency, fOP.

  fOP = The operating frequency at which the instruction execution times are needed.

  IET(7.5) = Instruction Execution Times with an operating frequency of 7.5 MHz. These times are listed in the following charts.
- NA => Not Applicable

NOTE

The times calculated are those using revision 5.18 of the microcode.

Table A-17 XOR & Single Operand Instructions

Instructions	Dest Mode	16 BIT MODE REFRESH ON Min	16 BIT MODE REFRESH ON Max	16 BIT MODE REFR OFF	8 BIT MODE REFRESH ON Word Instr	8 BIT MODE REFRESH ON Byte Instr	8 BIT MODE REFRESH OFF Word Instr	8 BIT MODE REFRESH OFF Byte Instr
CLR(B),COM(B),	0	1.6	1.73	1.6	2.53	2.53	2.4	2.4
INC(B),DEC(B),	1	2.8	2.93	2.8	5.33	3.73	5.2	3.6
NEG(B),ROR(B),	2	2.8	2.93	2.8	5.33	3.73	5.2	3.6
ROL(B),ASR(B),	3	3.6	3.73	3.6	6.93	5.33	6.8	5.2
ASL(B),SWAB,	4	3.2	3.33	3.2	5.73	4.13	5.6	4.0
ADC(B),SBC(B),	5	4.13	4.26	4.0	7.46	5.86	7.2	5.6
SXT,MFPS,	6	4.13	4.26	4.0	7.46	5.86	7.2	5.6
XOR	7	4.93	5.06	4.8	9.06	7.46	8.8	7.2
	0	1.6	1.73	1.6	2.53	2.53	2.4	2.4
	1	2.4	2.53	2.4	4.13	3.33	4.0	3.2
	2	2.4	2.53	2.4	4.13	3.33	4.0	3.2
TST(B)	3	3.2	3.33	3.2	5.73	4.93	5.6	4.8
	4	2.8	2.93	2.8	5.33	3.73	5.2	3.6
	5	3.73	3.86	3.6	6.26	5.46	6.0	5.2
	6	3.73	3.86	3.6	6.26	5.46	6.0	5.2
	7	4.53	4.66	4.4	7.86	7.06	7.6	6.8
	0	3.2	3.33	3.2	4.13	4.13	4.0	4.0
	1	4.0	4.13	4.0	4.93	4.93	4.8	4.8
	2	4.0	4.13	4.0	4.93	4.93	4.8	4.8
MTPS	3	4.8	4.93	4.8	6.53	6.53	6.4	6.4
	4	4.4	4.53	4.4	5.33	5.33	5.2	5.2
	5	5.33	5.46	5.2	7.06	7.06	6.8	6.8
	6	5.33	5.46	5.2	7.06	7.06	6.8	6.8
	7	6.13	6.26	6.0	8.66	8.66	8.4	8.4

* XOR and Single Operand Instruction execution times include instruction fetch, instruction decode, operand fetch, instruction operation and result output (except in Mode 0 and the TST(B) instruction where there is no output).

Table A-18 Double Operand Instructions

NOTE
Double Operand Execution Time = Source Mode Time + Destination Mode Time

Source Mode Time *

16 BIT MODE

Instructions	Src Mode	REFRESH ON				REFR OFF
		Dst Mode 0-4		Dst Mode 5-7		
		Min	Max	Min	Max	
MOV(B),CMP(B), ADD,SUB BIT(B),BIC(B), BIS(B)	0	1.2	1.33	1.33	1.33	1.2
	1	2.0	2.13	2.13	2.13	2.0
	2	2.0	2.13	2.13	2.13	2.0
	3	2.8	2.93	2.93	2.93	2.8
	4	2.4	2.53	2.53	2.53	2.4
	5	3.33	3.33	3.33	3.46	3.2
	6	3.33	3.33	3.33	3.46	3.2
	7	4.13	4.13	4.13	4.26	4.0

8 BIT MODE

Instructions	Src Mode	REFRESH ON		REFRESH OFF	
		Word Instr	Byte Instr	Word Instr	Byte Instr
MOV(B),CMP(B), ADD,SUB BIT(B),BIC(B), BIS(B)	0	2.13	2.13	2.0	2.0
	1	3.73	2.93	3.6	2.8
	2	3.73	2.93	3.6	2.8
	3	5.33	4.53	5.2	4.4
	4	4.13	3.33	4.0	3.2
	5	5.86	5.06	5.6	4.8
	6	5.86	5.06	5.6	4.8
	7	7.46	6.66	7.2	6.4

* Source Mode times include instruction fetch, instruction decode and source operand fetch.

Table A-18 Double Operand Instructions cont'd

Destination Mode Time *

Instruction	Dest Mode	16 BIT MODE			8 BIT MODE			
		REFRESH ON			REFRESH ON		REFRESH OFF	
		Min	Max	REFR OFF	Word Instr	Byte Instr	Word Instr	Byte Instr
MOV(B),ADD, SUB,BIC(B) BIS(B)	0	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	1	1.6	1.6	1.6	2.4	1.6	2.4	1.6
	2	1.6	1.6	1.6	2.4	1.6	2.4	1.6
	3	2.4	2.4	2.4	4.0	3.2	4.0	3.2
	4	2.0	2.0	2.0	2.8	2.0	2.8	2.0
	5	2.8	2.8	2.8	4.53	3.73	4.4	3.6
	6	2.8	2.8	2.8	4.53	3.73	4.4	3.6
	7	3.6	3.6	3.6	6.13	5.33	6.0	5.2
CMP(B),BIT(B)	0	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	1	1.2	1.2	1.2	2.0	1.2	2.0	1.2
	2	1.2	1.2	1.2	2.0	1.2	2.0	1.2
	3	2.0	2.0	2.0	3.6	2.8	3.6	2.8
	4	1.6	1.6	1.6	2.4	1.6	2.4	1.6
	5	2.4	2.4	2.4	4.13	3.33	4.0	3.2
	6	2.4	2.4	2.4	4.13	3.33	4.0	3.2
	7	3.2	3.2	3.2	5.73	4.93	5.6	4.8

* Destination Mode times include destination operand fetch, instruction operation and result output (except in Destination Mode 0 and the CMP(B) and BIT(B) instructions where there is no output).

Table A-19 Jump & Subroutine Instructions

Instruction	Dest Mode	16 BIT MODE			8 BIT MODE			
		REFRESH ON			REFRESH ON		REFRESH OFF	
		Min	Max	REFR OFF	Word Instr	Byte Instr	Word Instr	Byte Instr
JMP	1	2.0	2.13	2.0	2.93	NA	2.8	NA
	2	2.4	2.53	2.4	3.33	NA	3.2	NA
	3	2.4	2.53	2.4	4.13	NA	4.0	NA
	4	2.4	2.53	2.4	3.33	NA	3.2	NA
	5	2.93	2.93	2.8	4.53	NA	4.4	NA
	6	2.93	2.93	2.8	4.53	NA	4.4	NA
	7	3.73	3.73	3.6	6.13	NA	6.0	NA
JSR	1	3.6	3.73	3.6	5.33	NA	5.2	NA
	2	4.0	4.13	4.0	5.73	NA	5.6	NA
	3	4.0	4.13	4.0	6.53	NA	6.4	NA
	4	4.0	4.13	4.0	5.73	NA	5.6	NA
	5	4.53	4.53	4.4	6.93	NA	6.8	NA
	6	4.53	4.53	4.4	6.93	NA	6.8	NA
	7	5.33	5.33	5.2	8.53	NA	8.4	NA
RTS	NA	2.8	2.93	2.8	4.53	NA	4.4	NA
SOB	NA	2.4	2.53	2.4	3.33	NA	3.2	NA

Notes

1. JMP/JSR Destination Mode 0 is an illegal instruction that traps to vector location 10.

2. JMP execution times include instruction fetch, instruction decode, operand fetch and loading of the PC.

3. JSR execution times include instruction fetch, instruction decode, operand fetch, pushing the linkage register onto the stack and loading the PC.

4. RTS execution times include instruction fetch, instruction decode, loading the PC and popping the stack and loading the linkage register.

5. SOB execution times include instruction fetch, instruction decode, decrementing the count register, testing for zero and branching if necessary (NOTE: whether or not a branch is taken does not affect the execution time).

Table A-20 Branch, Trap & Interrupt Instructions

Instruction	Dest Mode	16 BIT MODE			8 BIT MODE			
		REFRESH ON			REFRESH ON		REFRESH OFF	
		Min	Max	REFR OFF	Word Instr	Byte Instr	Word Instr	Byte Instr
BR,BNE,BEQ, BPL,BMI,BVC, BVS,BCC,BCS, BGE,BLT,BGT, BLE,BHI,BLOS, BHIS,BLO	NA	1.6	1.73	1.6	2.53	NA	2.4	NA
EMT,TRAP, BPT,IOT	NA	6.53	6.66	6.4	9.73	NA	9.6	NA
RTI	NA	3.2	3.33	3.2	4.93	NA	4.8	NA
RTT	NA	4.4	4.53	4.4	7.13	NA	7.0	NA

Notes

1. Branch instructions execution times include instruction fetch, instruction decoding, doubling the offset, testing the conditions and adding the offset to the PC if the conditions are met (NOTE: Whether or not a branch is taken does not affect the execution times).

2. Trap instructions execution times include instruction fetch, instruction decode, pushing the PS and PC onto the stack, loading the PC with the contents of the vector location and loading the PS with the contents of the vector location plus two.

3. Return from interrupt instructions execution times include instruction fetch, instruction decode and popping the PC and PS from the stack.

Table A-21 Miscellaneous & Condition Code Instructions

Instruction	Dest Mode	16 BIT MODE			8 BIT MODE			
		REFRESH ON		REFR OFF	REFRESH ON		REFRESH OFF	
		Min	Max		Word Instr	Byte Instr	Word Instr	Byte Instr
HALT	NA	5.73	5.86	5.6	8.4	NA	8.0	NA
WAIT	NA	1.6	1.73	1.6	2.43	NA	2.4	NA
RESET	NA	14.6	14.73	14.6	16.53	NA	16.4	NA
NOP	NA	2.4	2.53	2.4	3.33	NA	3.2	NA
CLC,CLV,CLZ, CLN,CCC,SEC, SEV,SEZ,SEN, SCC	NA	2.4	2.53	2.4	3.33	NA	3.2	NA
MFPT	NA	2.0	2.13	2.0	2.93	NA	2.8	NA

Notes

1. HALT execution times include instruction fetch, instruction decode, writing the PC & PS into stack then loading the PS with 340 and loading the PC with the RESTART address.

2. WAIT execution times include instruction fetch, instruction decode, pulsing PI to sample the interrupt lines and doing a REFRESH cycle if REFRESH is on. (NOTE: If no interrupt lines were asserted during the PI pulse, the WAIT instruction will cycle in a 1.2 us loop pulsing PI (if REFRESH is on the loop will be 1.33 us max). The looping will continue until an interrupt line is asserted and sensed by the DCT11-AA.)

3. RESET execution times include instruction fetch, instruction decode, the assertion of -BCLR and the writing of DAL<15:0> into the MODE register.

4. NOP execution times include instruction fetch, instruction decode and idle time.

5. Condition Code instructions execution times include instruction fetch, instruction decode and the setting or resetting of the appropriate status flags in the PS.

Table A-22 Maximum Latencies

Active Inputs	Dest Mode	16 BIT MODE		8 BIT MODE	
		Dynamic	Static	Dynamic	Static
-CP<3:0>, -PF (Internal Vector)	NA	15.47	15.20	22.13	21.60
-VEC, -CP<3:0> (External Vector)	NA	15.87	15.60	22.53	22.00
DMR	NA	3.66	3.52	4.46	4.32
WAIT Instruction					
Internal Vector	NA	7.87	7.73	10.53	10.13
External Vector	NA	8.27	8.13	10.93	10.53
DMR	NA	1.66	1.66	1.79	1.66

Notes

1. These timings are given in microseconds and assume clock frequency of 7.5 MHz.

2. Interrupt latency is measured from the time the INTERRUPT REQUEST is asserted either on the AI lines (in static modes), or the input of the AI line driver (in dynamic modes), until the time the DCT11-AA is ready to fetch the first instruction in the interrupts' service routine. In this time the DCT11-AA takes three actions:

   a) Keeps going until a PI latches the request. This could happen in the instruction following the request.

   b) Finishes the instruction that latched the request.

   c) Executes the IACK microcode, which involves priority arbitration, issuing IACK, generating the interrupt vector (or in the case of -VEC being asserted, reading in the external vector), pushing PSW and PC onto the stack and loading PC and PSW from vector and vector +2.

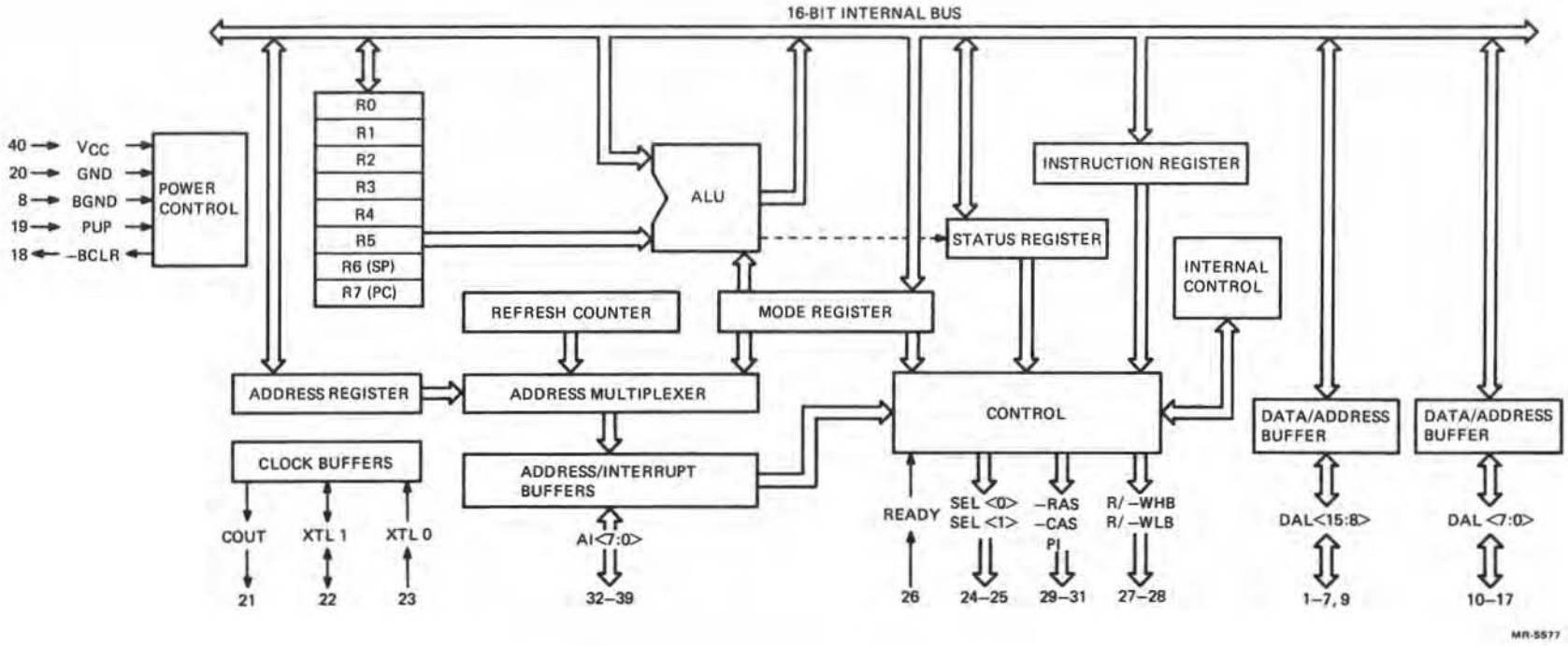
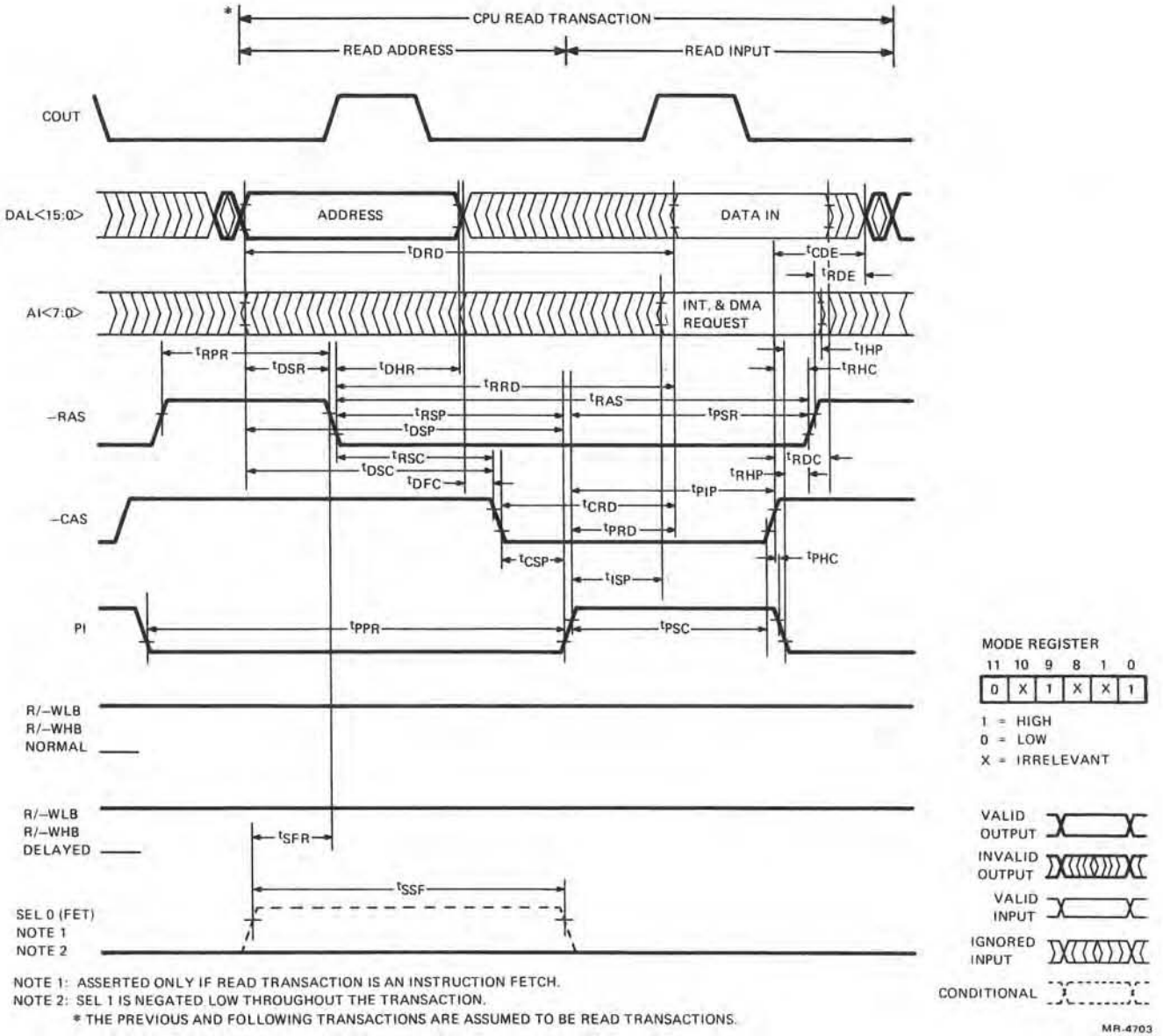


Figure A-1 DCT11-AA Block Diagram



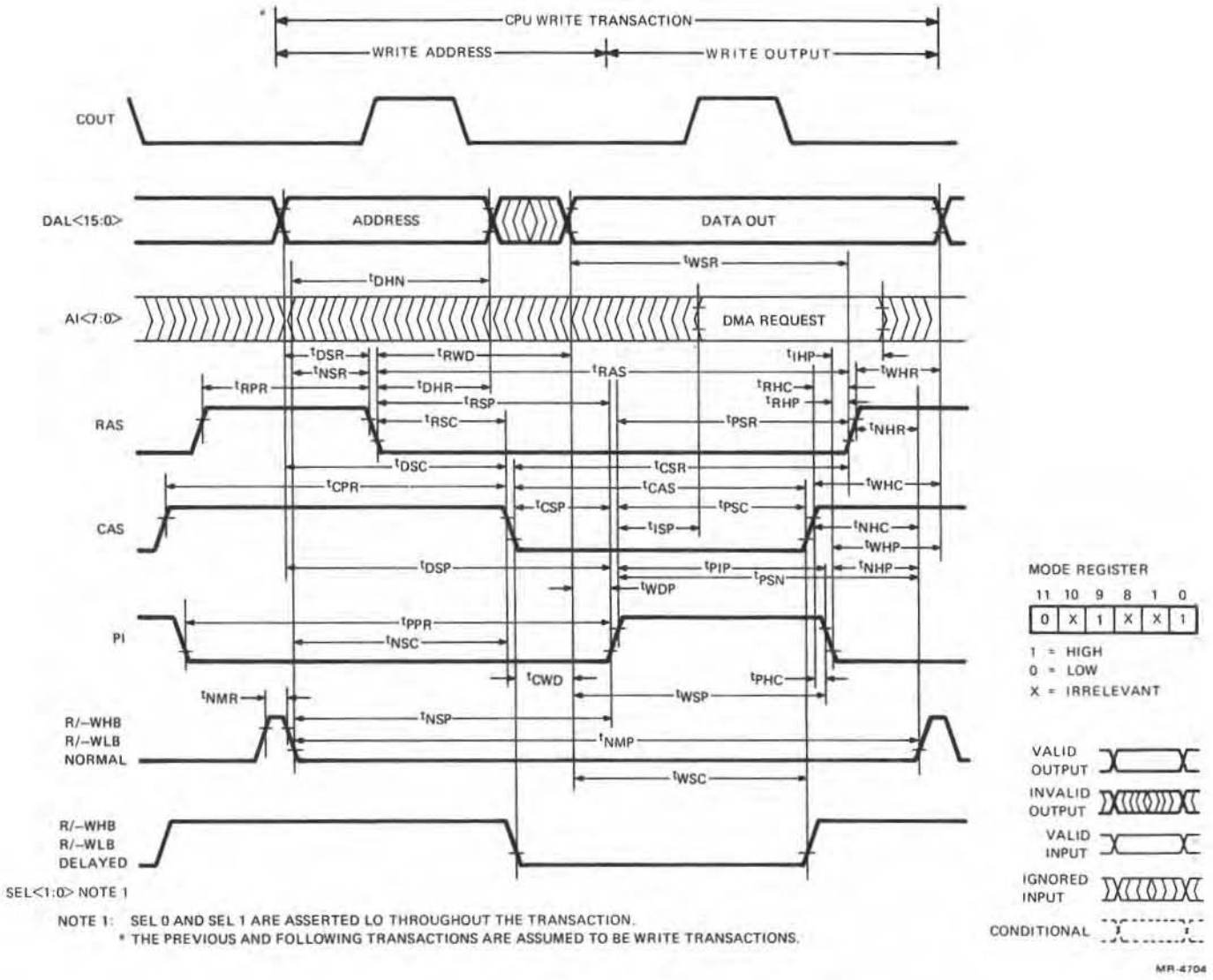
NOTE 1: ASSERTED ONLY IF READ TRANSACTION IS AN INSTRUCTION FETCH.
NOTE 2: SEL 1 IS NEGATED LOW THROUGHOUT THE TRANSACTION.
* THE PREVIOUS AND FOLLOWING TRANSACTIONS ARE ASSUMED TO BE READ TRANSACTIONS.

Figure A-2 16-Bit Static Read

	SYMBOL	PARAMETER	FUNCTION OF t_{CYC}	MIN/MAX
	t_{CDE}	–CAS (T.E.) to next DAL<15:0> Address Enable	(T –18) = 115 ns	min
1	t_{CRD}	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) to Read Data Valid	(3T –180) = 220 ns	max
	t_{CSP}	–CAS (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
	t_{CYC}	XTL1, XTL0 Operating Period	T = 133 ns	min
	t_{DFC}	DAL<15:0> Address Float to –CAS (L.E.)	0 ns	min
	t_{DHR}	DAL<15:0> Address Hold Time from –RAS (L.E.)	(T –12) = 121 ns	min
1	t_{DRD}	DAL<15:0> Address Set Up Time to Read Data Valid	(5T –157) = 510 ns	max
	t_{DSC}	DAL<15:0> Address Set Up Time to –CAS (L.E.)	(2T –22) = 245 ns	min
	t_{DSP}	DAL<15:0> Address Set Up Time to PI (L.E.)	(3T –20) = 380 ns	min
	t_{DSR}	DAL<15:0> Address Set Up Time to –RAS (L.E.)	(T –48) = 85 ns	min
	t_{IHP}	Input on AI<7:0> Hold Time from PI (T.E.)	0 ns	min
1	t_{ISP}	PI (L.E.) to Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	t_{PHC}	PI Hold Time from –CAS (T.E.)	10 ns	min
1	t_{PIP}	PI Pulse Width	(2T –47) = 220 ns	min
	t_{PPR}	PI Precharge Time	(4T –33) = 500 ns	min
1	t_{PRD}	PI (L.E.) to Read Data Valid	(2T –176) = 91 ns	max
	t_{PSC}	PI (L.E.) Set Up Time to –CAS (T.E.)	(2T –75) = 192 ns	min
1	t_{PSR}	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T –14) = 281 ns	min
1	t_{RAS}	–RAS Pulse Width	(4T –35) = 568 ns	min
	t_{RDC}	Read Data Hold Time from –CAS (T.E.)	0 ns	min
	t_{RDE}	–RAS (T.E.) to next DAL<15:0> Address Enable	(T –118) = 15 ns	min
	t_{RHC}	–RAS (T.E.) Hold Time from –CAS (T.E.)	50 ns	min
	t_{RHP}	–RAS (T.E.) Hold Time from PI (T.E.)	10 ns	min
	t_{RPR}	–RAS Precharge Time	(2T –120) = 147 ns	min
1	t_{RRD}	–RAS (L.E.) to Read Data Valid	(4T –128) = 405 ns	max
	t_{RSC}	–RAS (L.E.) Set Up Time to –CAS (L.E.)	(T + 10) = 143 ns	min
	t_{RSP}	–RAS (L.E.) Set Up Time to PI (L.E.)	(2T + 10) = 277 ns	min
	t_{SFR}	DMA on SEL<0> (L.E.) Set Up Time to –RAS (L.E.)	(2T –23) = 243 ns	min
	t_{SSF}	Fetch SEL<0> Pulse Width	(3T –38) = 362 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle

MR-5581



NOTE 1: SEL 0 AND SEL 1 ARE ASSERTED LO THROUGHOUT THE TRANSACTION.
* THE PREVIOUS AND FOLLOWING TRANSACTIONS ARE ASSUMED TO BE WRITE TRANSACTIONS.

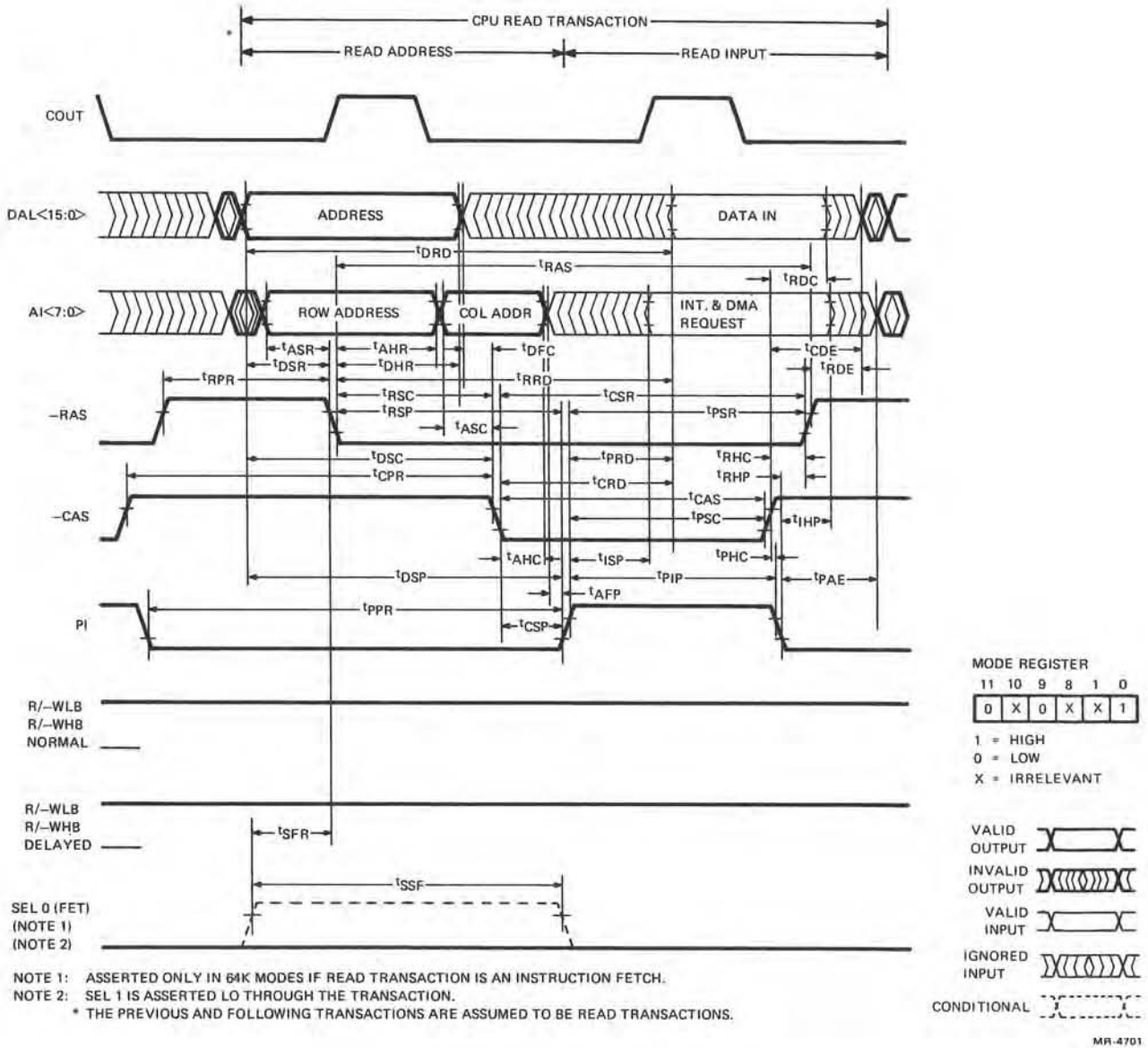
Figure A-3 16-Bit Static Write

	SYMBOL	PARAMETER	FUNCTION OF ^{t}CYC	MIN/MAX
	^{t}CAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^{t}CPR	–CAS Precharge Time	(3T –5) = 395 ns	min
	^{t}CSP	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
1	^{t}CSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^{t}CWD	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) to Write Data Valid	80 ns	max
	^{t}CYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^{t}DHN	DAL<15:0> Address Hold Time from Normal Mode R/$\overline{W}$ (L.E.)	(2T –20) = 247 ns	min
	^{t}DHR	DAL<15:0> Address Hold Time from –RAS (L.E.)	(T –12) = 121 ns	min
	^{t}DSC	DAL<15:0> Address Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(2T –22) = 245 ns	min
	^{t}DSP	DAL<15:0> Address Set Up Time to PI (L.E.)	(3T –20) = 380 ns	min
	^{t}DSR	DAL<15:0> Address Set Up Time to –RAS (L.E.)	(T –48) = 85 ns	min
	^{t}IHP	Input on AI<7:0> Hold Time from PI (T.E.)	0 ns	min
1	^{t}ISP	PI (L.E.) to Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	^{t}NHC	Normal Mode R/$\overline{W}$ Hold Time from –CAS (T.E.)	(T –32) = 101 ns	min
	^{t}NHP	Normal Mode R/$\overline{W}$ Hold Time from PI (T.E.)	(T –43) = 90 ns	min
	^{t}NHR	Normal Mode R/$\overline{W}$ Hold Time from RAS (T.E.)	(T –108) = 25 ns	min
1	^{t}NMP	Normal Mode R/$\overline{W}$ Pulse Width	(6T –66) = 734 ns	min
	^{t}NMR	Normal Mode R/$\overline{W}$ Recovery Time	0 ns	min
	^{t}NSC	Normal Mode R/$\overline{W}$ Set Up Time to –CAS (L.E.)	(2T –37) = 230 ns	min
	^{t}NSP	Normal Mode R/$\overline{W}$ Set Up Time to PI (L.E.)	(3T –45) = 355 ns	min
	^{t}NSR	Normal Mode R/$\overline{W}$ Set Up Time to –RAS (L.E.)	(T –78) = 55 ns	min
	^{t}PHC	PI Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	10 ns	min
1	^{t}PIP	PI Pulse Width	(2T –47) = 220 ns	min
	^{t}PPR	PI Precharge Time	(4T –33) = 500 ns	min
1	^{t}PSC	PI (L.E.) Set Up Time to –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(2T –75) = 192 ns	min
1	^{t}PSN	PI (L.E.) Set Up Time to Normal Mode R/$\overline{W}$ (T.E.)	(3T –90) = 310 ns	min
1	^{t}PSR	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T –14) = 281 ns	min

	SYMBOL	PARAMETER	FUNCTION OF ^{t}CYC	MIN/MAX
1	^{t}RAS	–RAS Pulse Width	(4T + 35) = 568 ns	min
	^{t}RHC	–RAS (T.E.) Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	50 ns	min
	^{t}RHP	–RAS (T.E.) Hold Time from PI (T.E.)	10 ns	min
	^{t}RPR	–RAS Precharge Time	(2T –120) = 147 ns	min
	^{t}RSC	–RAS (L.E.) Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(T + 10) = 143 ns	min
	^{t}RSP	–RAS (L.E.) Set Up Time to PI (L.E.)	(2T + 10) = 277 ns	min
	^{t}RWD	–RAS (L.E.) to Write Data Valid	(2T + 4) = 270 ns	max
	^{t}WDP	Write Data Set Up Time to PI (L.E.)	(T –83) = 50 ns	min
	^{t}WHC	Write Data or SAL<15:8> Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(T –28) = 105 ns	min
	^{t}WHP	Write Data Hold Time from PI (T.E.)	(T –88) = 45 ns	min
	^{t}WHR	Write Data or SAL<15:8> Hold Time from –RAS (T.E.)	(T –118) = 15 ns	min
	^{t}WSC	Write Data Set Up Time to –CAS (T.E.)	(3T –150) = 250 ns	min
	^{t}WSP	Write Data Set Up Time to PI (T.E.)	(3T –110) = 290 ns	min
	^{t}WSR	Write Data Set Up Time to –RAS (T.E.)	(3T –55) = 345 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

MR-5582



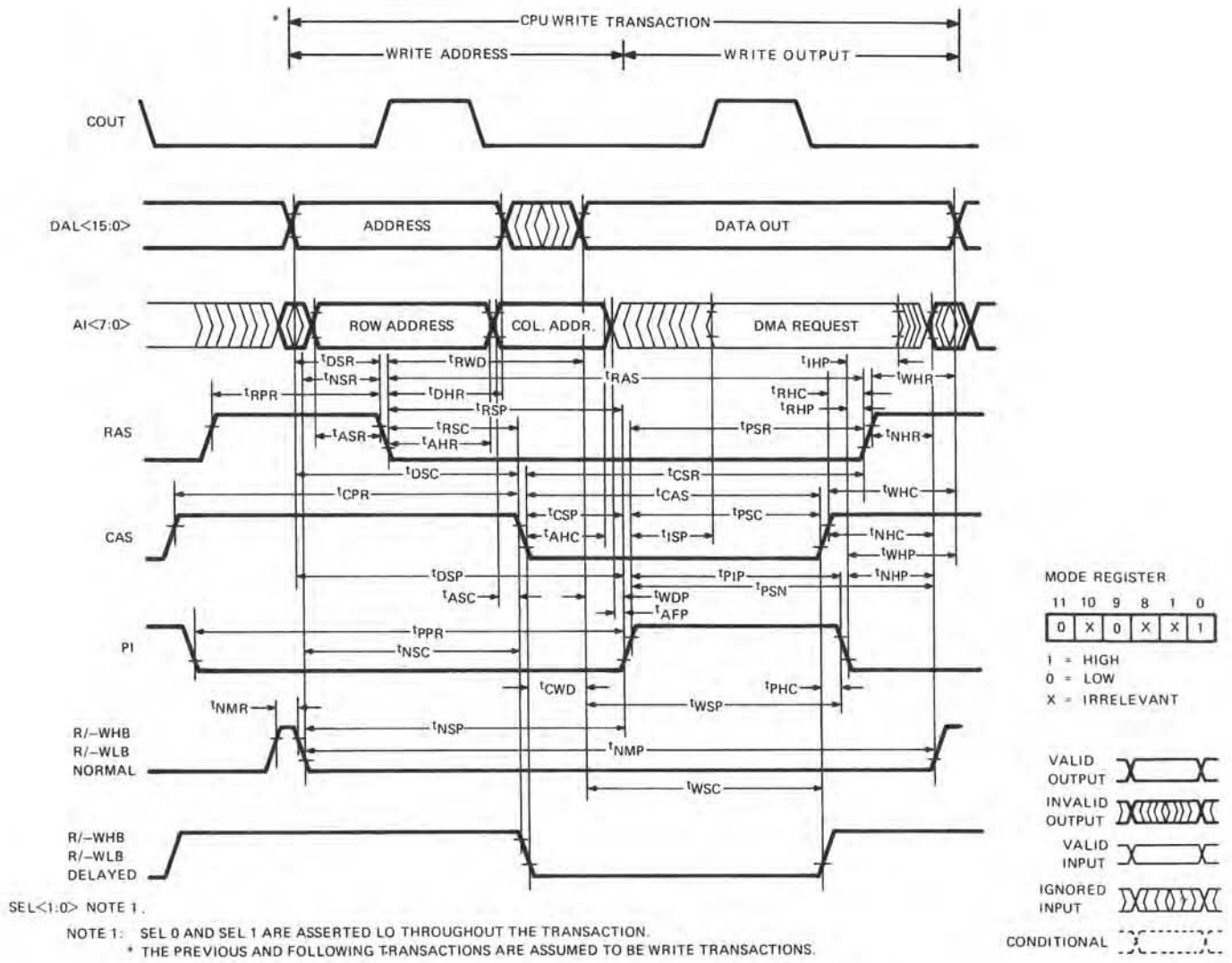
NOTE 1: ASSERTED ONLY IN 64K MODES IF READ TRANSACTION IS AN INSTRUCTION FETCH.
NOTE 2: SEL 1 IS ASSERTED LO THROUGH THE TRANSACTION.
* THE PREVIOUS AND FOLLOWING TRANSACTIONS ARE ASSUMED TO BE READ TRANSACTIONS.

Figure A-4 16-Bit Dynamic Read

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tAFP	Column Address on AI<7:0> Float to PI (L.E.)	0 ns	min
	^tAHC	Column Address on AI<7:0> Hold Time from –CAS (L.E.)	(T –43) = 90 ns	min
	^tAHR	Row Address on AI<7:0> Hold Time from –RAS (L.E.)	(T –60) = 73 ns	min
	^tASC	Column Address on AI<7:0> Set Up Time to –CAS (L.E.)	20 ns	min
	^tASR	Row Address on AI<7:0> Set Up Time to –RAS (L.E.)	(T –83) = 50 ns	min
1	^tCAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^tCDE	–CAS (T.E.) to next DAL<15:0> Address Enable	(T –18) = 115 ns	min
	^tCPR	–CAS Precharge Time	(3T –5) = 395 ns	min
1	^tCRD	–CAS (L.E.) to Read Data Valid	(3T –180) = 220 ns	max
	^tCSP	–CAS (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
1	^tCSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^tCYC	XTLI, XTL0 Operating Period	T = 133 ns	min
	^tDFC	DAL<15:0>Address Float to –CAS (L.E.)	0 ns	min
	^tDHR	DAL<15:0> Address Hold Time from –RAS (L.E.)	(T –12) = 121 ns	min
1	^tDRD	DAL<15:0> Address Set Up Time to Read Data Valid	(5T –157) = 510 ns	max
	^tDSC	DAL<15:0> Address Set Up Time to –CAS (L.E.)	(2T –22) = 245 ns	min
	^tDSP	DAL<15:0> Address Set Up Time to PI (L.E.)	(3T –20) = 380 ns	min
	^tDSR	DAL<15:0> Address Set Up Time to –RAS (L.E.)	(T –48) = 85 ns	min
	^tIHP	Input on AI<7:0> Hold Time from PI (T.E.)	0 ns	min
1	^tISP	PI (L.E.) to Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	^tPAE	PI (T.E.) to next AI<7:0> Address Enable	(T –40) = 93 ns	min
	^tPHC	PI Hold Time from –CAS (T.E.)	10 ns	min
1	^tPIP	PI Pulse Width	(2T –47) = 220 ns	min
	^tPPR	PI Precharge Time	(4T –33) = 500 ns	min
1	^tPRD	PI (L.E.) to Read Data Valid	(2T –176) = 91 ns	max
1	^tPSC	PI (L.E.) Set Up Time to –CAS (T.E.)	(2T –75) = 192 ns	min
1	^tPSR	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T –14) = 281 ns	min
1	^tRAS	–RAS Pulse Width	(4T + 35) = 568 ns	min
	^tRDC	Read Data Hold Time from –CAS (T.E.)	0 ns	min
	^tRDE	–RAS (T.E.) to Next DAL<15:0> Address Enable	(T –118) = 15 ns	min
	^tRHC	–RAS (T.E.) Hold Time from –CAS (T.E.)	50 ns	min
	^tRHP	–RAS (T.E.) Hold Time from PI (T.E.)	10 ns	min
	^tRPR	–RAS Precharge Time	(2T –120) = 147 ns	min
1	^tRRD	–RAS (L.E.) to Read Data Valid	(4T –128) = 405 ns	max
	^tRSC	–RAS (L.E.) Set Up Time to –CAS (L.E.)	(T + 10) = 143 ns	min
	^tRSP	–RAS (L.E.) Set Up Time to PI (L.E.)	(2T + 10) = 277 ns	min
	^tSFR	DMA SEL<0> (L.E.) Set Up Time to –RAS (L.E.)	(2T–23) = 243 ns	min
	^tSSF	Fetch SEL<0> Pulse Width	(3T –38) = 362 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

MR-5579



NOTE 1: SEL 0 AND SEL 1 ARE ASSERTED LO THROUGHOUT THE TRANSACTION.
* THE PREVIOUS AND FOLLOWING TRANSACTIONS ARE ASSUMED TO BE WRITE TRANSACTIONS.

Figure A-5 16-Bit Dynamic Write

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tAFP	Column Address on AI<7:0> Float to PI (L.E.)	0 ns	min
	^tAHC	Column Address on AI<7:0> Hold Time from –CAS (L.E.)	(T –43) = 90 ns	min
	^tAHR	Row Address on AI<7:0> Hold Time from –RAS (L.E.)	(T –60) = 73 ns	min
	^tASC	Column Address on AI<7:0> Set Up Time to –CAS (L.E.)	20 ns	min
	^tASR	Row Address on AI<7:0> Set Up Time to –RAS (L.E.)	(T –83) = 50 ns	min
1	^tCAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^tCPR	–CAS Precharge Time	(3T –5) = 395 ns	min
1	^tCSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^tCWD	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) to Write Data Valid	80 ns	max
	^tCYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^tDHN	DAL<15:0> Address Hold Time from Normal Mode R/$\overline{W}$ (L.E.)	(2T –20) = 247 ns	min
	^tDHR	DAL<15:0> Address Hold Time from –RAS (L.E.)	(T –12) = 121 ns	min
	^tDSC	DAL<15:0> Address Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(2T –22) = 245 ns	min
	^tDSP	DAL<15:0> Address Set Up Time to PI (L.E.)	(3T –20) = 380 ns	min
	^tDSR	DAL<15:0> Address Set Up Time to –RAS (L.E.)	(T –48) = 85 ns	min
	^tIHP	Input on AI<7:0> Hold Time from PI (T.E.)	0 ns	min
1	^tISP	PI (L.E.) to Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	^tNHC	Normal Mode R/$\overline{W}$ Hold Time from –CAS (T.E.)	(T –32) = 101 ns	min
	^tNHP	Normal Mode R/$\overline{W}$ Hold Time from PI (T.E.)	(T –43) = 90 ns	min
	^tNHR	Normal Mode R/$\overline{W}$ Hold Time from –RAS (T.E.)	(T –108) = 25 ns	min
1	^tNMP	Normal Mode R/$\overline{W}$ Pulse Width	(6T –66) = 734 ns	min
	^tNMR	Normal Mode R/$\overline{W}$ Recovery Time	0 ns	min
	^tNSC	Normal Mode R/$\overline{W}$ Set Up Time to –CAS (L.E.)	(2T –37) = 230 ns	min
	^tNSP	Normal Mode R/$\overline{W}$ Set Up Time to PI (L.E.)	(3T –45) = 355 ns	min
	^tNSR	Normal Mode R/$\overline{W}$ Set Up Time to –RAS (L.E.)	(T –78) = 55 ns	min

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tPAE	PI (T.E.) to next AI<7:0> Address Enable	(T –40) = 93 ns	min
	^tPHC	PI Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	10 ns	min
1	^tPIP	PI Pulse Width	(2T –47) = 220 ns	min
	^tPPR	PI Precharge Time	(4T –33) = 500 ns	min
1	^tPSC	PI (L.E.) Set Up Time to –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(2T –75) = 192 ns	min
1	^tPSN	PI (L.E.) Set Up Time to Normal Mode R/$\overline{W}$ (T.E.)	(3T –90) = 310 ns	min
1	^tPSR	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T –14) = 281 ns	min
1	^tRAS	–RAS Pulse Width	(4T + 35) = 568 ns	min
	^tRHC	–RAS (T.E.) Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	50 ns	min
	^tRHP	–RAS (T.E.) Hold Time from PI (T.E.)	10 ns	min
1	^tRPR	–RAS Precharge Time	(2T –120) = 147 ns	min
	^tRSC	–RAS (L.E.) Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(T + 10) = 143 ns	min
	^tRSP	–RAS (L.E.) Set Up Time to PI (L.E.)	(2T + 10) = 277 ns	min
	^tRWD	–RAS (L.E.) to Write Data Valid	(2T + 4) = 270 ns	max
	^tWDP	Write Data Set Up Time to PI (L.E.)	(T –83) = 50 ns	min
	^tWHC	Write Data or SAL<15:8> Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(T –28) = 105 ns	min
	^tWHP	Write Data Hold Time from PI (T.E.)	(T –88) = 45 ns	min
	^tWHR	Write Data or SAL<15:8> Hold Time from –RAS (T.E.)	(T –118) = 15 ns	min
	^tWSC	Write Data Set Up Time to –CAS (T.E.)	(3T –150) = 250 ns	min
	^tWSP	Write Data Set Up Time to PI (T.E.)	(3T –110) = 290 ns	min
	^tWSR	Write Data Set Up Time to –RAS (T.E.)	(3T –55) = 345 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

MR-5580

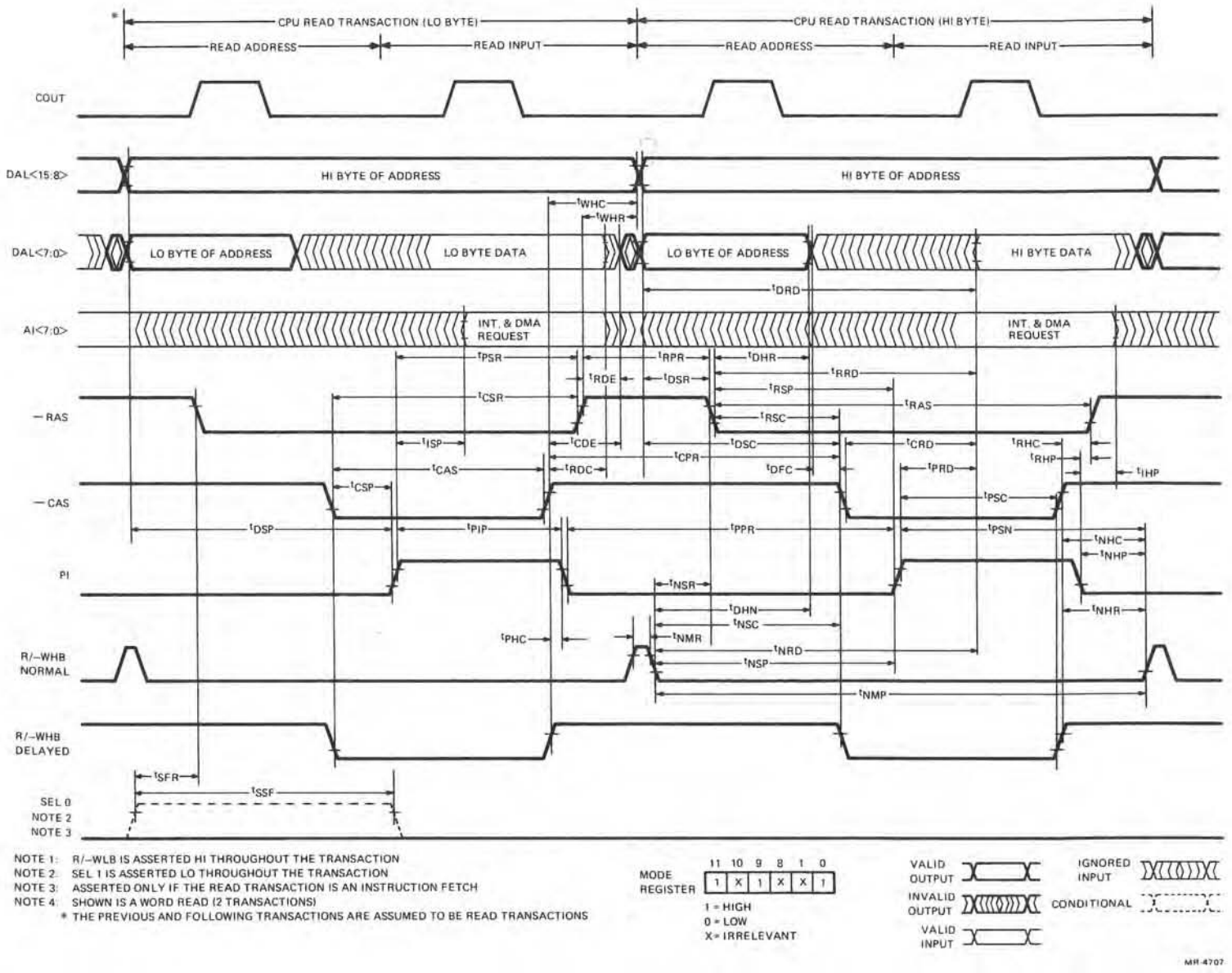


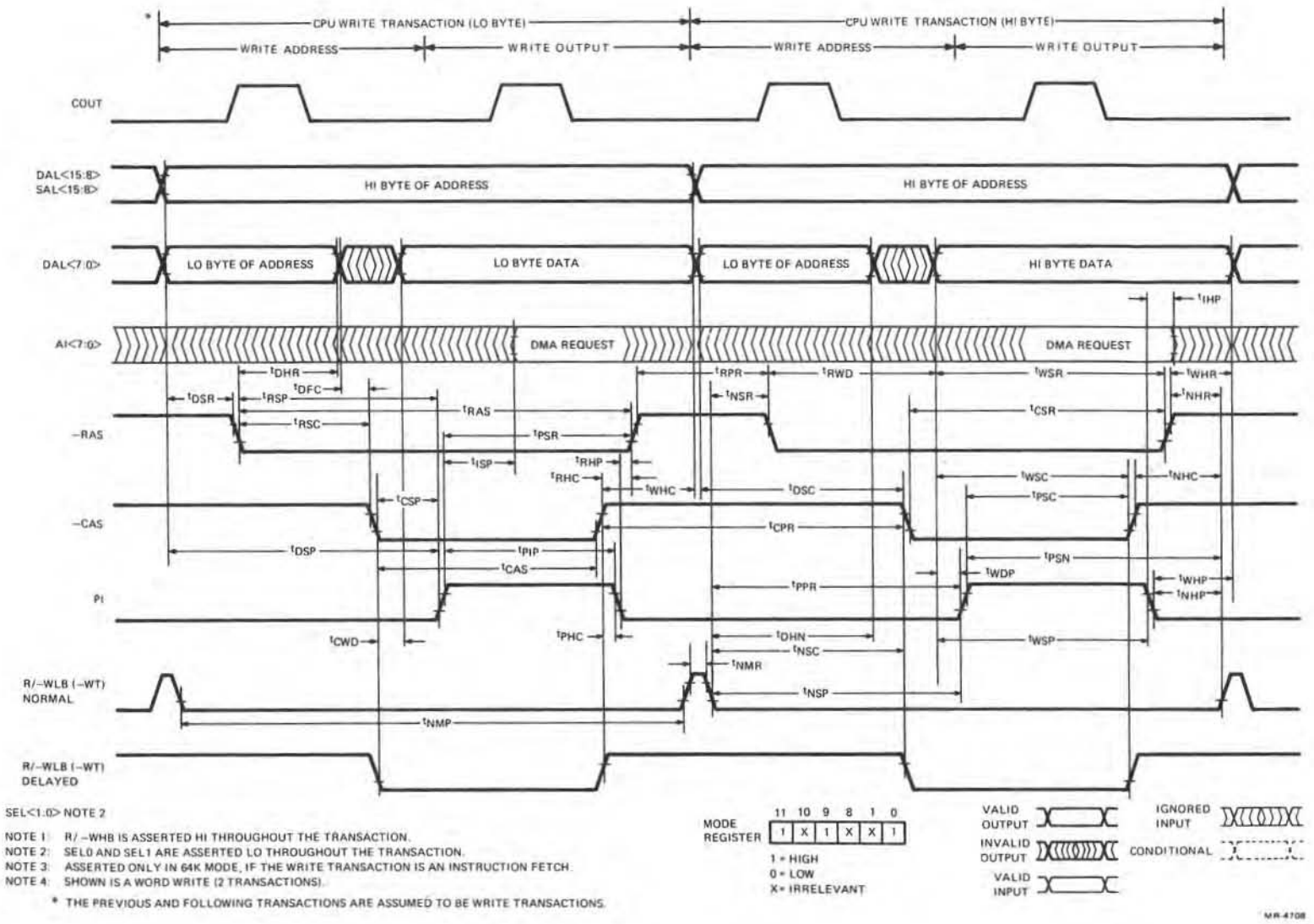
Figure A-6 8-Bit Static Read

	SYMBOL	PARAMETER	FUNCTION OF ^{t}CYC	MIN/MAX
	^{t}CAS	–CAS Pulse Width	(3T –90) = 310 ns	min
1	^{t}CDE	–CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.) to next DAL<15:0> Address Enable	(T –18) = 115 ns	min
	^{t}CPR	–CAS Precharge Time	(3T –5) = 395 ns	min
	^{t}CRD	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) to Read Data Valid	(3T –180) = 220 ns	max
	^{t}CSP	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
1	^{t}CSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^{t}CYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^{t}DFC	DAL<15:0> Address Float to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	0 ns	min
	^{t}DHN	DAL<15:0> Address Hold Time from Normal Mode R/$\overline{W}$ (L.E.)	(2T –20) = 247 ns	min
	^{t}DHR	DAL<15:0> Address Hold Time from –RAS (L.E.)	(T –12) = 121 ns	min
	^{t}DRD	DAL<15:0> Address Set Up Time to Read Data Valid	(5T –157) = 510 ns	max
	^{t}DSC	DAL<15:0> Address Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(2T –22) = 245 ns	min
	^{t}DSP	DAL<15:0> Address Set Up Time to PI (L.E.)	(3T –20) = 380 ns	min
	^{t}DSR	DAL<15:0> Address Set Up Time to –RAS (L.E.)	(T –48) = 85 ns	min
	^{t}IHP	Input on AI<7:0> Hold Time from PI (T.E.)	0 ns	min
1	^{t}ISP	PI (L.E.) to Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	^{t}NHC	Normal Mode R/$\overline{W}$ Hold Time from –CAS (T.E.)	(T –32) = 101 ns	min
	^{t}NHP	Normal Mode R/$\overline{W}$ Hold Time from PI (T.E.)	(T –43) = 90 ns	min
	^{t}NHR	Normal Mode R/$\overline{W}$ Hold Time from –RAS (T.E.)	(T –108) = 25 ns	min
1	^{t}NMP	Normal Mode R/$\overline{W}$ Pulse Width	(6T –66) = 734 ns	min
	^{t}NMR	Normal Mode R/$\overline{W}$ Recovery Time	0 ns	min
1	^{t}NRD	Normal Mode R/$\overline{W}$ Set Up Time to Read Data Valid	(5T –148) = 519 ns	max
	^{t}NSC	Normal Mode R/$\overline{W}$ Set Up Time to –CAS (L.E.)	(2T –37) = 230 ns	min
	^{t}NSP	Normal Mode R/$\overline{W}$ Set Up Time to PI (L.E.)	(3T –45) = 355 ns	min
	^{t}NSR	Normal Mode R/$\overline{W}$ Set Up Time to –RAS (L.E.)	(T –78) = 55 ns	min
	^{t}PHC	PI Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	10 ns	min

	SYMBOL	PARAMETER	FUNCTION OF ^{t}CYC	MIN/MAX
1	^{t}PIP	PI Pulse Width	(2T –47) = 220 ns	min
	^{t}PPR	PI Precharge Time	(4T –33) = 500 ns	min
1	^{t}PRD	PI (L.E.) to Read Data Valid	(2T –176) = 91 ns	max
1	^{t}PSC	PI (L.E.) Set Up Time to –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(2T –75) = 192 ns	min
1	^{t}PSN	PI (L.E.) Set Up Time to Normal Mode R/$\overline{W}$ (T.E.)	(3T –90) = 310 ns	min
1	^{t}PSR	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T –14) = 281 ns	min
1	^{t}RAS	–RAS Pulse Width	(4T + 35) = 568 ns	min
	^{t}RDC	Read Data Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	0 ns	min
	^{t}RDE	–RAS (T.E.) to next DAL<15:0> Address Enable	(T –118) = 15 ns	min
	^{t}RHC	–RAS (T.E.) Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	50 ns	min
	^{t}RHP	–RAS (T.E.) Hold Time from PI (T.E.)	10 ns	min
	^{t}RPR	–RAS Precharge Time	(2T –120) = 147 ns	min
1	^{t}RRD	–RAS (L.E.) to Read Data Valid	(4T –128) = 405 ns	max
	^{t}RSC	–RAS (L.E.) Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(T + 10) = 143 ns	min
	^{t}RSP	–RAS (L.E.) Set Up Time to PI (L.E.)	(2T + 10) = 277 ns	min
	^{t}SFR	DMA on SEL <0> (L.E.) Set Up Time to –RAS (L.E.)	(2T –23) = 243 ns	min
	^{t}SSF	SEL<0> Pulse Width	(3T –38) = 362 ns	min
	^{t}WHC	Write Data or SAL<15:8> Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(T –28) = 105 ns	min
	^{t}WHR	Write Data or SAL<15:8> Hold Time from –RAS (T.E.)	(T –118) = 15 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

MR-5585



NOTE 1: R/-WHB IS ASSERTED HI THROUGHOUT THE TRANSACTION.
NOTE 2: SEL0 AND SEL1 ARE ASSERTED LO THROUGHOUT THE TRANSACTION.
NOTE 3: ASSERTED ONLY IN 64K MODE, IF THE WRITE TRANSACTION IS AN INSTRUCTION FETCH.
NOTE 4: SHOWN IS A WORD WRITE (2 TRANSACTIONS).

* THE PREVIOUS AND FOLLOWING TRANSACTIONS ARE ASSUMED TO BE WRITE TRANSACTIONS.

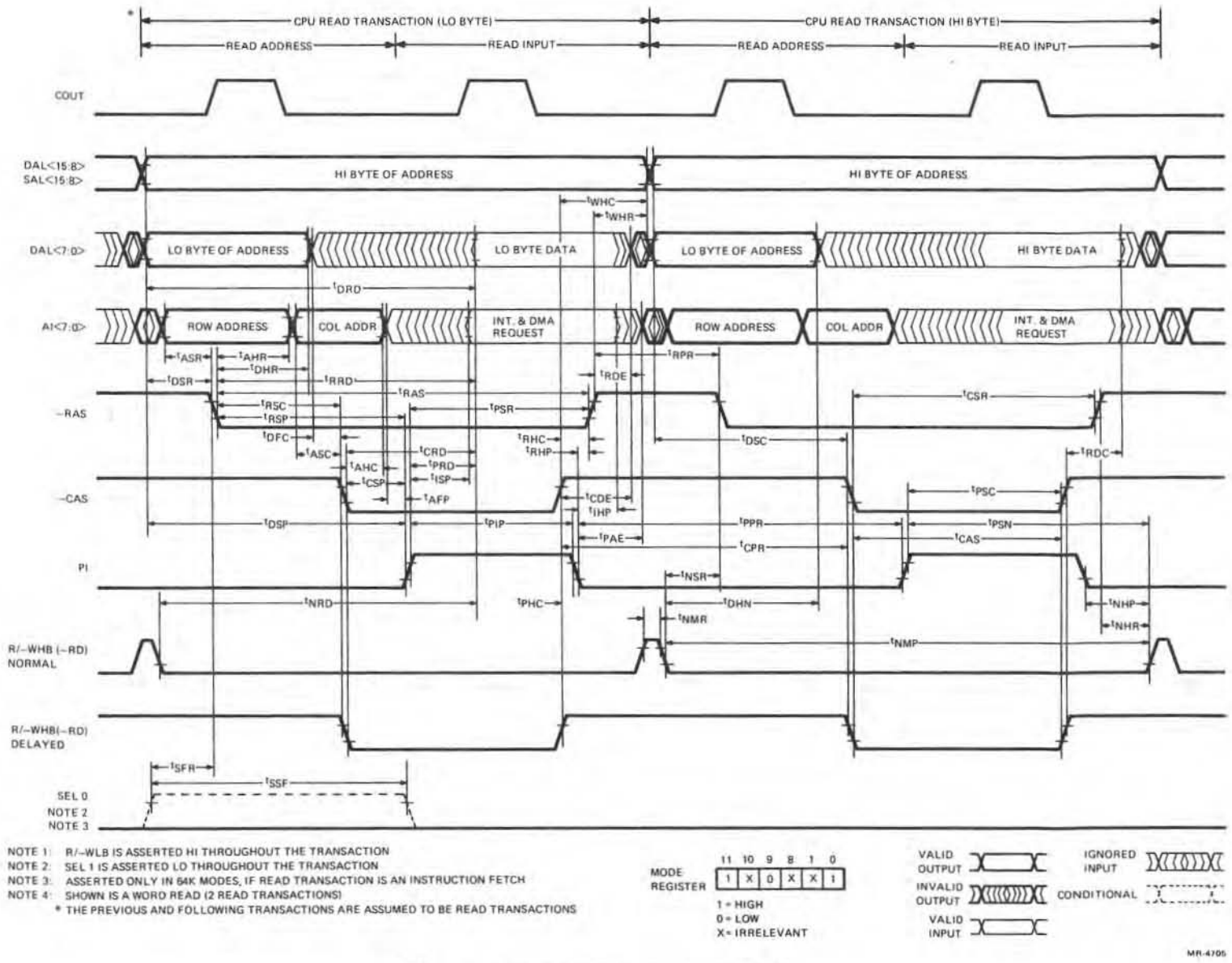
Figure A-7 8-Bit Static Write

	SYMBOL	PARAMETER	FUNCTION OF ^{t}CYC	MIN/MAX
	^{t}CAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^{t}CPR	–CAS Precharge Time	(3T –5) = 395 ns	min
	^{t}CSP	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
1	^{t}CSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^{t}CWD	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E. to Write Data Valid	80 ns	max
	^{t}CYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^{t}DFC	DAL<15:0> Address Float to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	0 ns	min
	^{t}DHN	DAL<15:0> Address Hold Time from Normal Mode R/$\overline{W}$ (L.E.)	(2T –20) = 247 ns	min
	^{t}DHR	DAL<15:0> Address Hold Time from –RAS (L.E.)	(T –12) = 121 ns	min
	^{t}DSC	DAL<15:0> Address Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(2T –22) = 245 ns	min
	^{t}DSP	DAL<15:0> Address Set Up Time to PI (L.E.)	(3T –20) = 380 ns	min
	^{t}DSR	DAL<15:0> Address Set Up Time to –RAS (L.E.)	(T –48) = 85 ns	min
	^{t}IHP	Input on AI<7:0> Hold Time from PI (T.E.)	0 ns	min
1	^{t}ISP	PI (L.E.) to Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	^{t}NHC	Normal Mode R/$\overline{W}$ Hold Time from –CAS (T.E.)	(T –32) = 101 ns	min
	^{t}NHP	Normal Mode R/$\overline{W}$ Hold Time from PI (T.E.)	(T –43) = 90 ns	min
	^{t}NHR	Normal Mode R/$\overline{W}$ Hold Time from –RAS (T.E.)	(T –108) = 25 ns	min
1	^{t}NMP	Normal Mode R/$\overline{W}$ Pulse Width	(6T –66) = 734 ns	min
	^{t}NMR	Normal Mode R/$\overline{W}$ Recovery Time	0 ns	min
	^{t}NSC	Normal Mode R/$\overline{W}$ Set Up Time to –CAS (L.E.)	(2T –37) = 230 ns	min
	^{t}NSP	Normal Mode R/$\overline{W}$ Set Up Time to PI (L.E.)	(3T –45) = 355 ns	min
	^{t}NSR	Normal Mode R/$\overline{W}$ Set Up Time to –RAS (L.E.)	(T –78) = 55 ns	min
	^{t}PHC	PI Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	10 ns	min
1	^{t}PIP	PI Pulse Width	(2T –47) = 220 ns	min
	^{t}PPR	PI Precharge Time	(4T –33) = 500 ns	min
1	^{t}PSC	PI (L.E.) Set Up Time to –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(2T –75) = 192 ns	min
1	^{t}PSN	PI (L.E.) Set Up Time to Normal Mode R/$\overline{W}$ (T.E.)	(3T –90) = 310 ns	min
1	^{t}PSR	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T –14) = 281 ns	min
1	^{t}RAS	–RAS Pulse Width	(4T + 35) = 568 ns	min
	^{t}RHC	–RAS (T.E.) Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	50 ns	min
	^{t}RHP	–RAS (T.E.) Hold Time from PI (T.E.)	10 ns	min
	^{t}RPR	–RAS Precharge Time	(2T –120) = 147 ns	min
	^{t}RSC	–RAS (L.E.) Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(T + 10) = 143 ns	min
	^{t}RSP	–RAS (L.E.) Set Up Time to PI (L.E.)	(2T + 10) = 277 ns	min
	^{t}RWD	–RAS (L.E.) to Write Data Valid	(2T + 4) = 270 ns	max
	^{t}WDP	Write Data Set Up Time to PI (L.E.)	(T –83) = 50 ns	min
	^{t}WHC	Write Data or SAL<15:8> Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(T –28) = 105 ns	min
	^{t}WHP	Write Data Hold Time from PI (T.E.)	(T –88) = 45 ns	min
	^{t}WHR	Write Data or SAL<15:8> Hold Time from –RAS (T.E.)	(T –118) = 15 ns	min
2	^{t}WSC	Write Data Set Up Time to –CAS (T.E.)	(3T –150) = 250 ns	min
2	^{t}WSP	Write Data Set Up Time to PI (T.E.)	(3T –110) = 290 ns	min
2	^{t}WSR	Write Data Set Up Time to –RAS (T.E.)	(3T –55) = 345 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

2: Add 4T for each READY pulse.

MR-5586



NOTE 1: R/~WLB IS ASSERTED HI THROUGHOUT THE TRANSACTION
NOTE 2: SEL 1 IS ASSERTED LO THROUGHOUT THE TRANSACTION
NOTE 3: ASSERTED ONLY IN 64K MODES, IF READ TRANSACTION IS AN INSTRUCTION FETCH
NOTE 4: SHOWN IS A WORD READ (2 READ TRANSACTIONS)
* THE PREVIOUS AND FOLLOWING TRANSACTIONS ARE ASSUMED TO BE READ TRANSACTIONS

MODE REGISTER

11	10	9	8	1	0
1	X	0	X	X	1

1 = HIGH
0 = LOW
X = IRRELEVANT

VALID OUTPUT
INVALID OUTPUT
VALID INPUT
IGNORED INPUT
CONDITIONAL

MR-4705

Figure A-8 8-Bit Dynamic Read

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tAFP	Column Address on AI<7:0> Float to PI (L.E.)	0 ns	min
	^tAHC	Column Address on AI<7:0> Hold Time from –CAS (L.E.)	(T –43) = 90 ns	min
	^tAHR	Row Address on AI<7:0> Hold Time from –RAS (L.E.)	(T –60) = 73 ns	min
	^tASC	Column Address on AI<7:0> Set Up Time to –CAS (L.E.)	20 ns	min
	^tASR	Row Address on AI<7:0> Set Up Time to –RAS (L.E.)	(T –83) = 50 ns	min
1	^tCAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^tCDE	–CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.) to next DAL<15:0> Address Enable	(T –18) = 115 ns	min
	^tCPR	–CAS Precharge Time	(3T –5) = 395 ns	min
1	^tCRD	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) to Read Data Valid	(3T –180) = 220 ns	max
	^tCSP	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
1	^tCSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^tCYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^tDFC	DAL<15:0> Address Float to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	0 ns	min
	^tDHN	DAL<15:0> Address Hold Time from Normal Mode R/$\overline{W}$ (L.E.)	(2T –20) = 247 ns	min
	^tDHR	DAL<15:0> Address Hold Time from –RAS (L.E.)	(T –12) = 121 ns	min
1	^tDRD	DAL<15:0> Address Set Up Time to Read Data Valid	(5T –157) = 510 ns	max
	^tDSC	DAL<15:0> Address Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(2T –22) = 245 ns	min
	^tDSP	DAL<15:0> Address Set Up Time to PI (L.E.)	(3T –20) = 380 ns	min
	^tDSR	DAL<15:0> Address Set Up Time to –RAS (L.E.)	(T –48) = 85 ns	min
	^tIHP	Input on AI <7:0>Hold Time from PI (T.E.)	0 ns	min
1	^tISP	PI (L.E.) to Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	^tNHP	Normal Mode R/$\overline{W}$ Hold Time from PI (T.E.)	(T –21) = 112 ns	min
	^tNHR	Normal Mode R/$\overline{W}$ Hold Time from –RAS (T.E.)	(T –108) = 25 ns	min
2	^tNMP	Normal Mode R/$\overline{W}$ Pulse Width	(6T –66) = 734 ns	min
	^tNMR	Normal Mode R/$\overline{W}$ Recovery Time	0 ns	min
2	^tNRD	Normal Mode R/W Set Up Time to Read Data Valid	(5T –148) = 519 ns	max

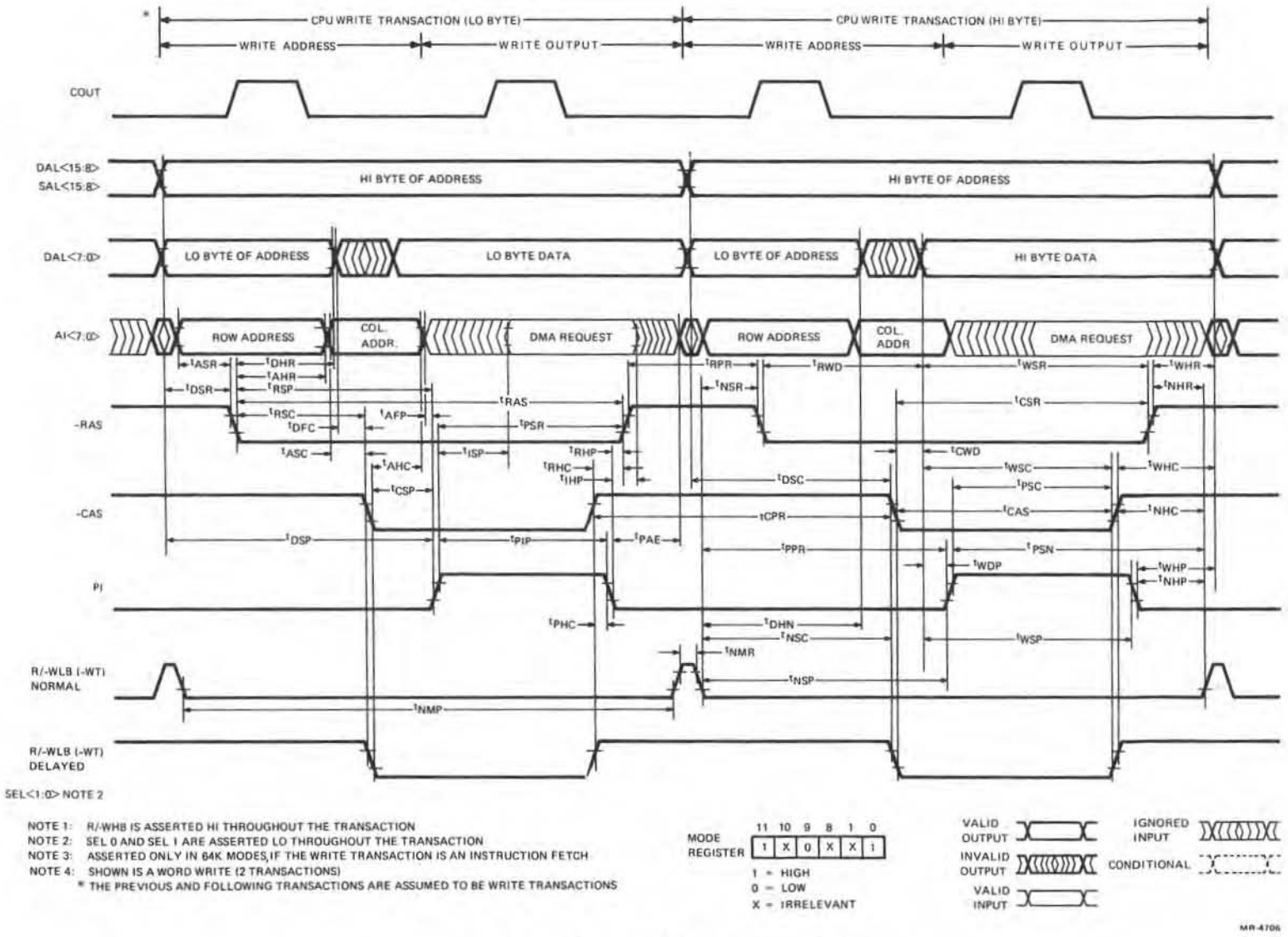
	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tNSR	Normal Mode R/$\overline{W}$ Set Up Time to –RAS (L.E.)	(T –78) = 55 ns	min
	^tPAE	PI (T.E.) to next AI<7:0> Address Enable	(T –40) = 93 ns	min
	^tPHC	PI Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	10 ns	min
1	^tPIP	PI Pulse Width	(2T –47) = 220 ns	min
	^tPPR	PI Precharge Time	(4T –33) = 500 ns	min
1	^tPRD	PI (L.E.) to Read Data Valid	(2T –176) = 91 ns	max
1	^tPSC	PI (L.E.) Set Up Time to –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(2T –75) = 192 ns	min
2	^tPSN	PI (L.E.) Set Up Time to Normal Mode R/$\overline{W}$ (T.E.)	(3T –90) = 310 ns	min
1	^tPSR	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T –14) = 281 ns	min
1	^tRAS	–RAS Pulse Width	(4T + 35) = 568 ns	min
	^tRDC	Read Data Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	0 ns	min
	^tRDE	–RAS (T.E.) to next DAL<15:0> Address Enable	(T –118) = 15 ns	min
	^tRHC	–RAS (T.E.) Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	50 ns	min
	^tRHP	–RAS (T.E.) Hold Time from PI (T.E.)	10 ns	min
	^tRPR	–RAS Precharge Time	(2T –120) = 147 ns	min
1	^tRRD	–RAS (L.E.) to Read Data Valid	(4T –128) = 405 ns	max
	^tRSC	–RAS (L.E.) Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(T + 10) = 143 ns	min
	^tRSP	–RAS (L.E.) Set Up Time to PI (L.E.)	(2T + 10) = 277 ns	min
	^tSFR	DMA on SEL<0> (L.E.) Set Up Time to –RAS (L.E.)	(2T –23) = 243 ns	min
3	^tSSF	SEL<0> Pulse Width	(3T –38) = 362 ns	min
	^tWHC	Write Data or SAL<15:8> Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(T –28) = 105 ns	min
	^tWHR	Write Data or SAL<15:8> Hold Time from –RAS (T.E.)	(T –118) = 15 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

2: Add 4T for each READY pulse.

3: Add 3T if multiple DMA cycles are granted.

MR-5583



NOTE 1: R/-WHB IS ASSERTED HI THROUGHOUT THE TRANSACTION
NOTE 2: SEL 0 AND SEL 1 ARE ASSERTED LO THROUGHOUT THE TRANSACTION
NOTE 3: ASSERTED ONLY IN 64K MODES, IF THE WRITE TRANSACTION IS AN INSTRUCTION FETCH
NOTE 4: SHOWN IS A WORD WRITE (2 TRANSACTIONS)
* THE PREVIOUS AND FOLLOWING TRANSACTIONS ARE ASSUMED TO BE WRITE TRANSACTIONS

MODE REGISTER

11	10	9	8	1	0
1	X	0	X	X	1

1 = HIGH
0 = LOW
X = IRRELEVANT

VALID OUTPUT
INVALID OUTPUT
VALID INPUT
IGNORED INPUT
CONDITIONAL

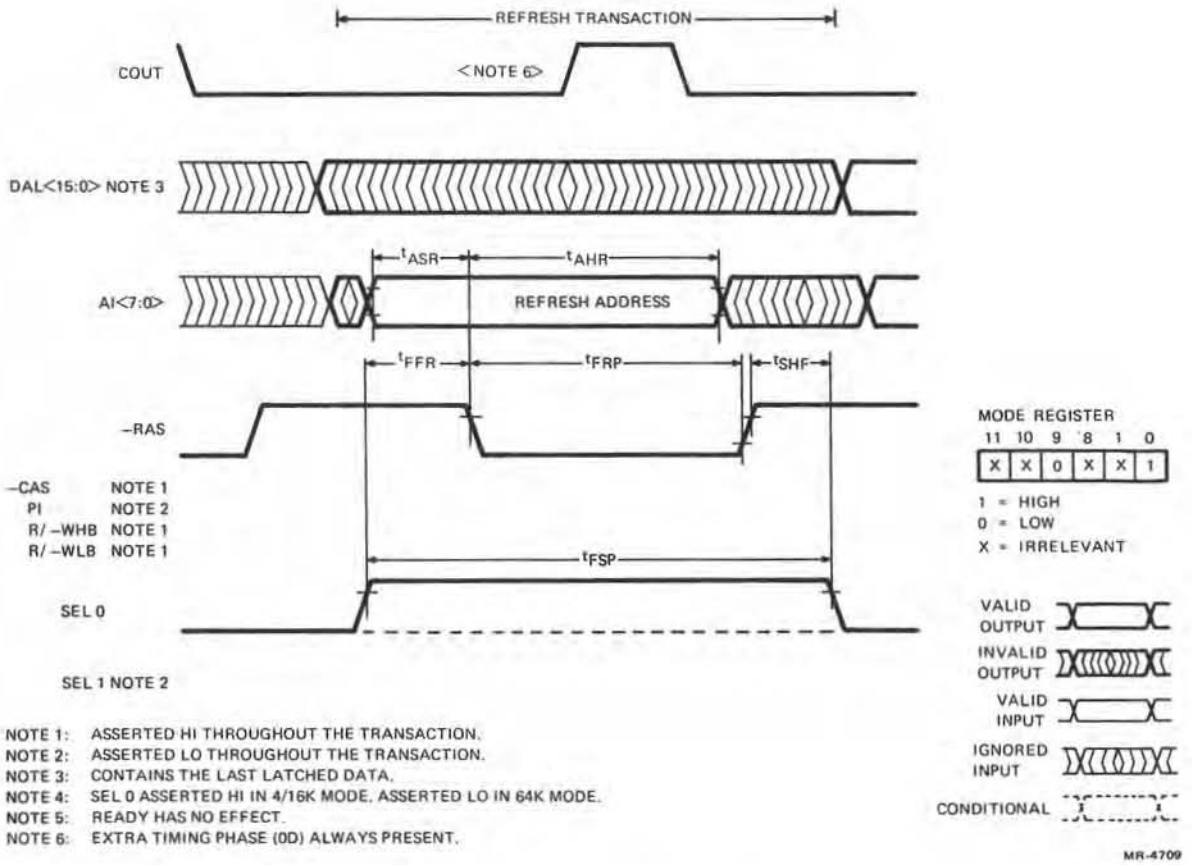
MR-4706

Figure A-9 8-Bit Dynamic Write

	SYMBOL	PARAMETER	FUNCTION OF ^{t}CYC	MIN/MAX
	^{t}AFP	Column Address on AI<7:0> Float to PI (L.E.)	0 ns	min
	^{t}AHC	Column Address on AI<7:0> Hold Time from –CAS (L.E.)	(T –43) = 90 ns	min
	^{t}AHR	Row Address on AI<7:0> Hold Time from –RAS (L.E.)	(T –60) = 73 ns	min
	^{t}ASC	Column Address on AI<7:0> Set Up Time to –CAS (L.E.)	20 ns	min
	^{t}ASR	Row Address on AI<7:0> Set Up Time to –RAS (L.E.)	(T –83) = 50 ns	min
1	^{t}CAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^{t}CPR	–CAS Precharge Time	(3T –5) = 395 ns	min
	^{t}CSP	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
1	^{t}CSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^{t}CWD	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) to Write Data Valid	80 ns	max
	^{t}CYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^{t}DFC	DAL<15:0>Address Float to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	0 ns	min
	^{t}DHN	DAL<15:0> Address Hold Time from Normal Mode R/$\overline{W}$ (L.E.)	(2T –20) = 247 ns	min
	^{t}DHR	DAL<15:0> Address Hold Time from –RAS (L.E.)	(T –12) = 121 ns	min
	^{t}DSC	DAL<15:0> Address Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(2T –22) = 245 ns	min
	^{t}DSP	DAL <15:0> Address Set Up Time to PI (L.E.)	(3T –20) = 380 ns	min
	^{t}DSR	DAL <15:0> Address Set Up Time to –RAS (L.E.)	(T –48) = 85 ns	min
	^{t}IHP	Input on AI <7:0> Hold Time from PI (T.E.)	0 ns	min
1	^{t}ISP	PI (L.E.) to Input on AI <7:0> Valid	(2T –167) = 100 ns	max
	^{t}NHC	Normal Mode R/$\overline{W}$ Hold Time from –CAS (T.E.)	(T –32) = 101 ns	min
	^{t}NHP	Normal Mode R/$\overline{W}$ Hold Time from PI (T.E.)	(T –43) = 90 ns	min
	^{t}NHR	Normal Mode R/$\overline{W}$ Hold Time from –RAS (T.E.)	(T –108) = 25 ns	min
1	^{t}NMP	Normal Mode R/$\overline{W}$ Pulse Width	(6T –66) = 734 ns	min
	^{t}NMR	Normal Mode R/$\overline{W}$ Recovery Time	0 ns	min
	^{t}NSC	Normal Mode R/$\overline{W}$ Set Up Time to –CAS (L.E.)	(2T –37) = 230 ns	min
	^{t}NSP	Normal Mode R/$\overline{W}$ Set Up Time to PI (L.E.)	(3T –45) = 355 ns	min
	^{t}NSR	Normal Mode R/$\overline{W}$ Set Up Time to –RAS (L.E.)	(T –78) = 55 ns	min
	^{t}PAE	PI (T.E.) to Next AI <7:0> Address Enable	(T –40) = 93 ns	min
	^{t}PHC	PI Hold Time from –CRS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	10 ns	min
1	^{t}PIP	PI Pulse Width	(2T –47) = 220 ns	min
	^{t}PPR	PI Precharge Time	(4T –33) = 500 ns	min
1	^{t}PSC	PI (L.E.) Set Up Time to –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(2T –75) = 192 ns	min
1	^{t}PSN	PI (L.E.) Set Up Time to Normal Mode R/W (T.E.)	(3T –90) = 310 ns	min
1	^{t}PSR	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T –14) = 281 ns	min
1	^{t}RAS	–RAS Pulse Width	(4T + 35) = 568 ns	min
	^{t}RHC	–RAS (T.E.) Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	50 ns	min
	^{t}RHP	–RAS (T.E.) Hold Time from PI (T.E.)	10 ns	min
	^{t}RPR	–RAS Precharge Time	(2T –120) = 147 ns	min
	^{t}RSC	–RAS (L.E.) Set Up Time to –CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.)	(T + 10) = 143 ns	min
	^{t}RSP	–RAS (L.E.) Set Up Time to PI (L.E.)	(2T + 10) = 277 ns	min
	^{t}RWD	–RAS (L.E.) to Write Data Valid	(2T + 4) = 270 ns	max
	^{t}WDP	Write Data Set Up Time to to PI (L.E.)	(T –83) = 50 ns	min
	^{t}WHC	Write Data or SAL <15:8> Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	(T –28) = 105 ns	min
	^{t}WHP	Write Data Hold Time from PI (T.E.)	(T –88) = 45 ns	min
	^{t}WHR	Write Data or SAL <15:8> Hold Time from –RAS (T.E.)	(T –118) = 15 ns	min
1	^{t}WSC	Write Data Set Up Time to –CAS (T.E.)	(3T –150) = 250 ns	min
1	^{t}WSP	Write Data Set Up Time to PI (T.E.)	(3T –110) = 290 ns	min
1	^{t}WSR	Write Data Set Up Time to –RAS (T.E.)	(3T –55) = 345 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

MR-5584



NOTE 1: ASSERTED HI THROUGHOUT THE TRANSACTION.
NOTE 2: ASSERTED LO THROUGHOUT THE TRANSACTION.
NOTE 3: CONTAINS THE LAST LATCHED DATA.
NOTE 4: SEL 0 ASSERTED HI IN 4/16K MODE, ASSERTED LO IN 64K MODE.
NOTE 5: READY HAS NO EFFECT.
NOTE 6: EXTRA TIMING PHASE (0D) ALWAYS PRESENT.

Figure A-10 Refresh

SYMBOL	PARAMETER	FUNCTION OF t_{CYC}	MIN/MAX
t_{ASR}	Refresh Address on AI<7:0> Set Up Time to −RAS (L.E.)	(T −83) = 50 ns	min
t_{AHR}	Refresh Address on AI<7:0> Hold Time from −RAS (L.E.)	(T−60) =73	min
t_{CYC}	XTL1, XTL0 Operating Period	T = 133 ns	min
t_{FRP}	Refresh −RAS Pulse Width	(2T + 35) = 302 ns	min
t_{FSP}	Refresh Select on SEL<0> Pulse Width	(4T −20) = 513 ns	min
t_{FFR}	Refresh Select on SEL<0> (L.E.) Set Up Time to −RAS (L.E.)	(T−23) = 110 ns	min
t_{SHF}	Refresh Select on SEL<0> (T.E.) Hold Time from −RAS (T.E.)	(T−123) =10	min

MR-5587

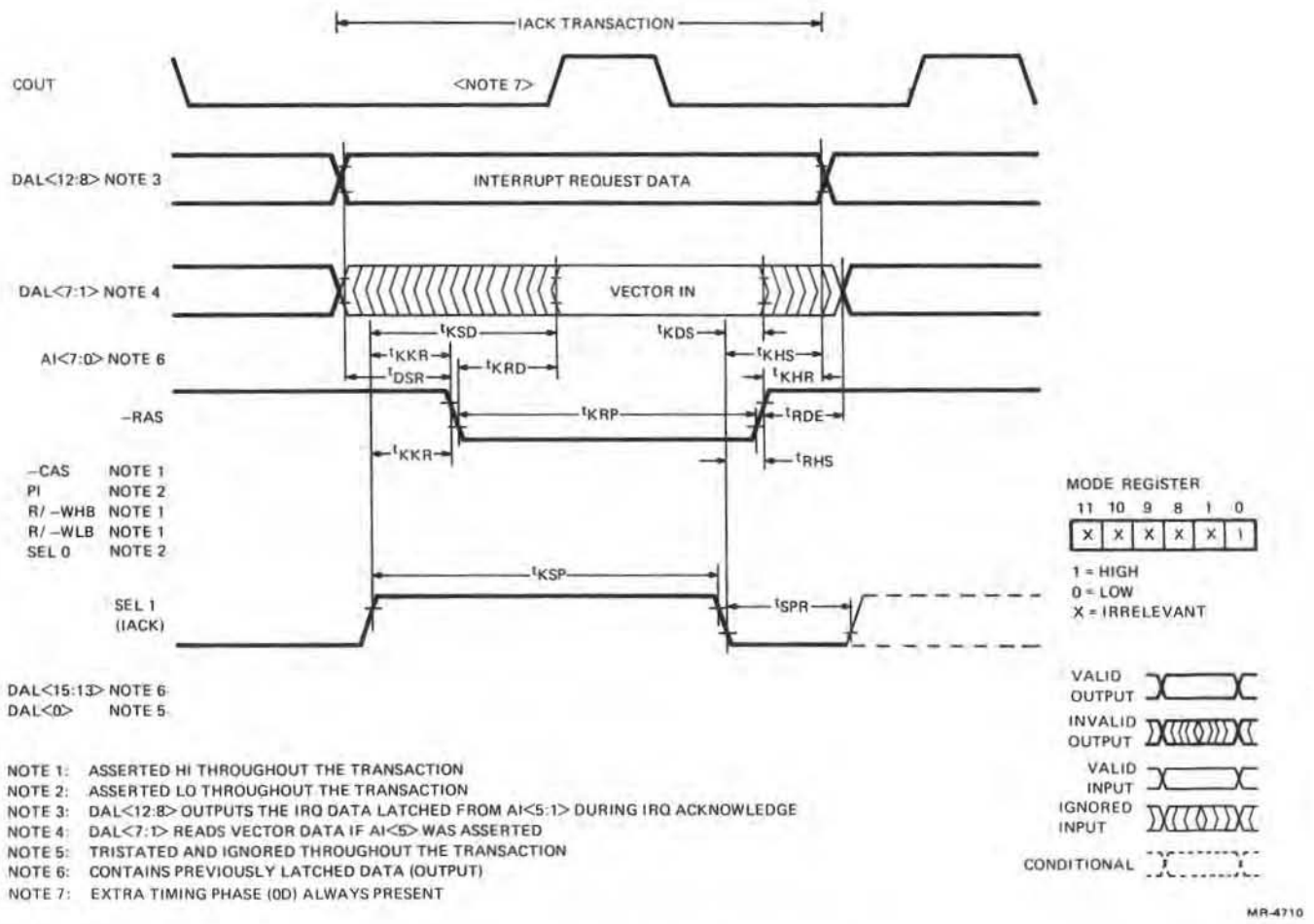


Figure A-11 Iack Transaction

	SYMBOL	PARAMETER	FUNCTION OF t_{CYC}	MIN/MAX
	t_{CYC}	XTL1, XTL0 Operating Period	T = 133 ns	min
	t_{DSR}	IACK Data Set Up Time	(T −48) = 85 ns	min
	t_{KDS}	Vector Data on DAL<7:2> Hold Time from IACK Select on SEL<1> (T.E.)	0 ns	min
1	t_{KHR}	IACK info on DAL<15:8> Hold Time from −RAS (T.E.)	(T −118) = 15 ns	min
	t_{KHS}	IACK info on DAL<15:8> Hold Time from IACK Select on SEL<1> (T.E.)	(T −50) = 83 ns	min
	t_{KKR}	IACK Select on SEL<1> (L.E.) Set Up Time to −RAS (L.E.)	(T−63) = 70 ns	min
1	t_{KRD}	IACK −RAS (L.E.) to Vector Data on DAL<7:0> Valid	(2T −148) = 119 ns	max
1	t_{KRP}	IACK −RAS Pulse Width	(2T + 35) = 302 ns	min
1	t_{KSD}	IACK Select on SEL<1> (L.E.) to Vector Data on DAL<7:2> Valid	(3T −155) = 245 ns	max
1	t_{KSP}	IACK Select on SEL<1> Pulse Width	(3T −66) = 334 ns	min
	t_{RDE}	−RAS (T.E.) to next DAL<15:0> Address Enable	(T −118) = 15 ns	min
	t_{RHS}	−RAS (T.E.) Hold Time from IACK Select on SEL<1> (T.E.)	45 ns	min
	t_{SPR}	IACK or DMA Select on SEL<1> Recovery Time	(T) = 133 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

MR-5588

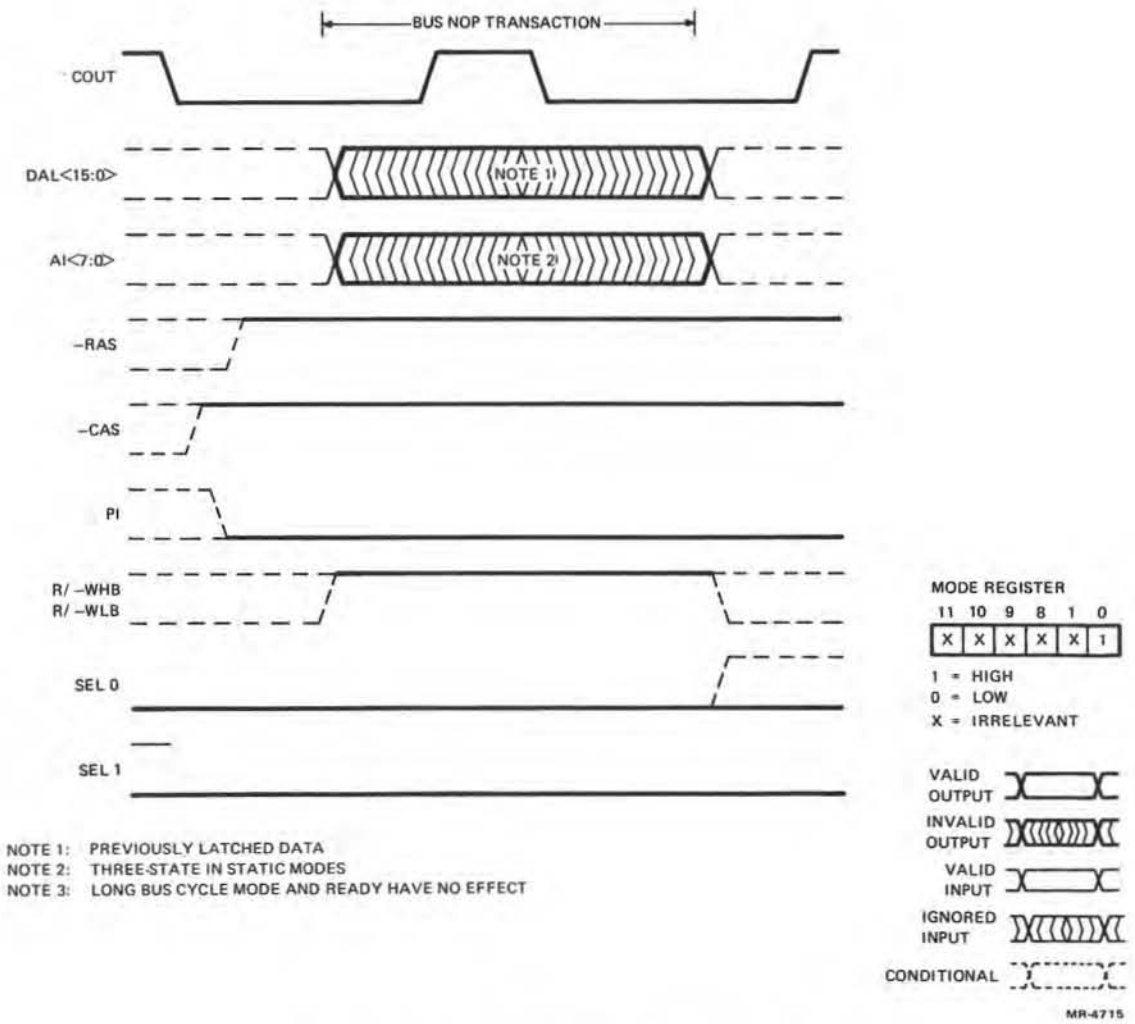


Figure A-12 Bus Nop Transaction

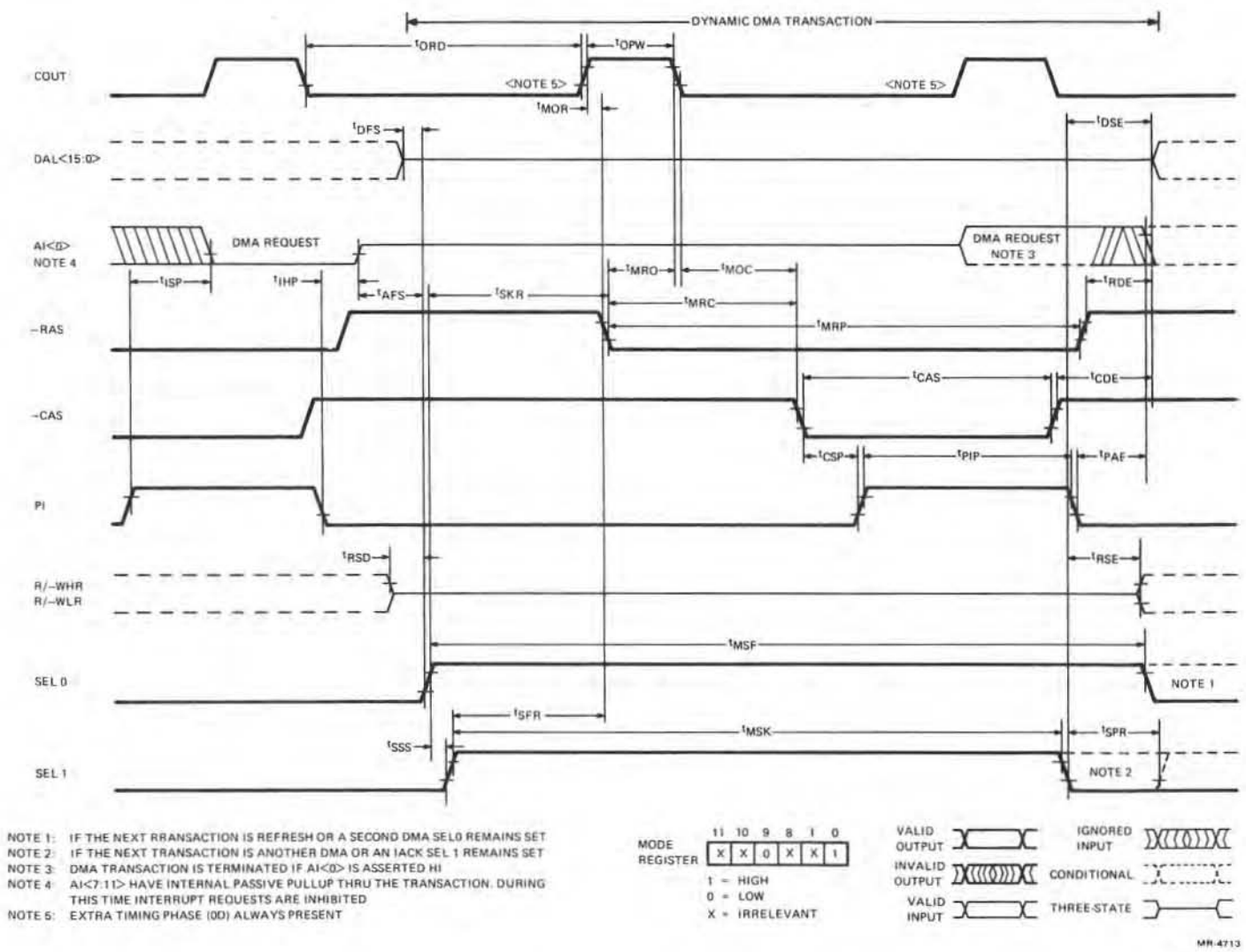


Figure A-13 DMA Transaction

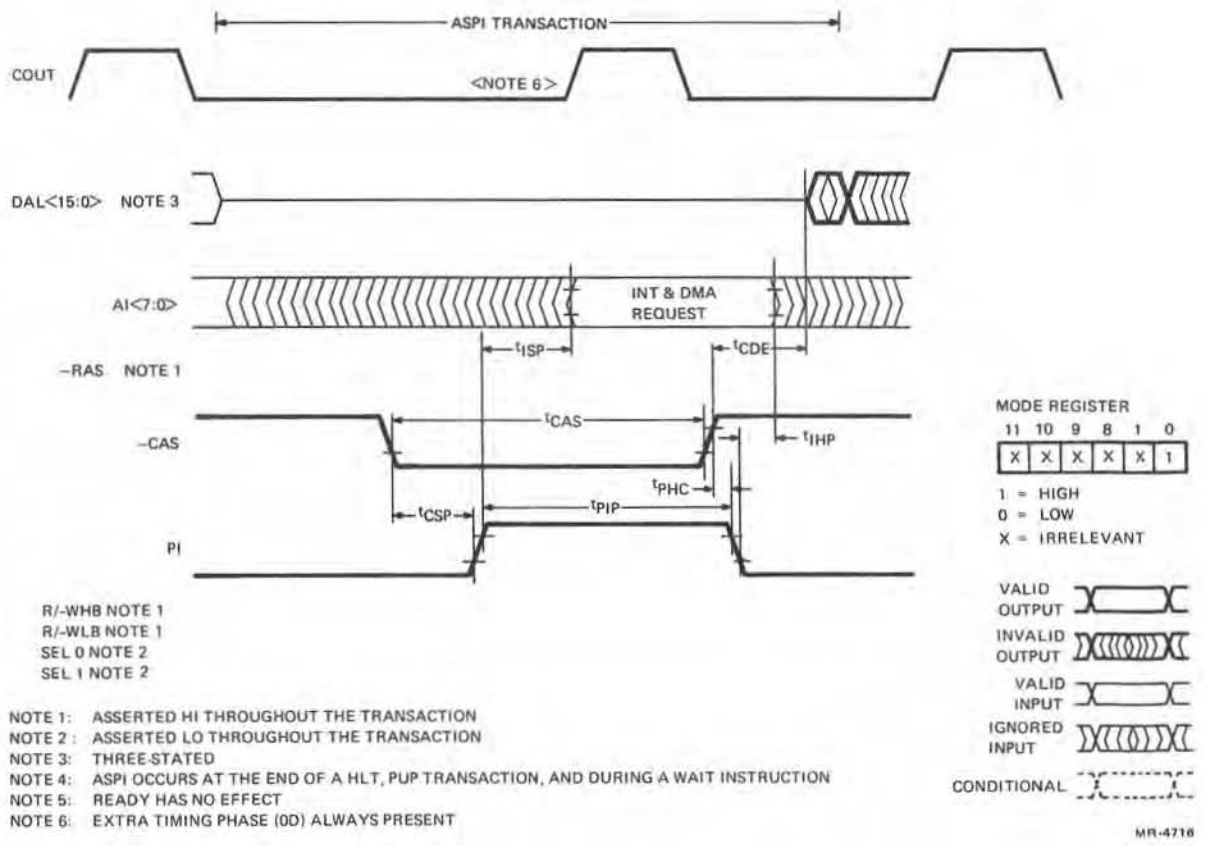
	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tAFS	AI<7:0> Float to DMA Select on SEL<0>	0 ns	min
1	^tCAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^tCDE	–CAS (T.E.) to next DAL<15:0> Address Enable	(T –18) = 115 ns	min
	^tCSP	–CAS (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
	^tCYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^tDFS	DAL<15:0> Float to DMA Select on SEL<0> (L.E.)	0 ns	min
	^tDSE	DAL<15:0> Enable from DMA Select on SEL<1> (T.E.)	(T –27) = 106 ns	min
	^tIHP	Input on AI<7:0> Hold Time from PI (T.E.)	0 ns	min
1	^tISP	PI (L.E.) to Interrupt or DMA Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	^tSFR	DMA Select on SEL<0> (L.E.) Set Up Time to –RAS (L.E.)	(2T –23) = 243 ns	min
	^tSKR	DMA Select on SEL<1> (L.E.) Set Up Time to–RAS (L.E.)	(2T –63) = 203 ns	min
	^tMOC	Pulse Mode COUT (T.E.) Set Up Time to –CAS (L.E.)	(T+10) = 143 ns	min
	^tMOR	Pulse Mode COUT (L.E.) Set Up Time to –RAS (L.E.)	0 ns	min
	^tMRC	DMA Select –RAS (L.E.) Set Up Time to –CAS (L.E.)	(2T + 10) = 277 ns	min
	^tMRO	DMA Pulse Mode COUT (T.E.) Hold Time from –RAS (L.E.)	(T –51) = 82 ns	min
2	^tMRP	DMA Select –RAS Pulse Width	(5T + 35) = 702 ns	min
2,3	^tMSF	DMA Select on SEL<0> Pulse Width	(8T –38) = 1029 ns	min
2,3	^tMSK	DMA Select on SEL<1> Pulse Width	(7T –68) = 865 ns	min
	^tOPW	Pulse Mode COUT Pulse Width	(T –33) = 100 ns	min
	^tORD	Pulse Mode COUT Recovery Time when 0D is present	(3T –37) = 363 ns	min
	^tPAE	PI (T.E.) to next AI<7:0> Address Enable	(T –40) = 93 ns	min
1	^tPIP	PI Pulse Width	(2T –47) = 220 ns	min
	^tRDE	–RAS (T.E.) to next DAL<15:0> Address Enable	(T –118) = 15 ns	min
	^tRSD	R/$\overline{W}$ Drivers disabled and Passive Pull Up Enabled Set Up Time to DMA Select on SEL<0> (L.E.)	0 ns	min
	^tRSE	R/$\overline{W}$ driver enable from DMA Select on SEL<1> (T.E.)	(T –27) = 106 ns	min
	^tSPR	IACK or DMA Select on SEL<1> Recovery Time	(T) = 133 ns	min
	^tSSS	SEL<0> (L.E.) Set Up Time to SEL<1> (L.E.)	0 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

2: Add 4T for each READY pulse.

3: Add 8T if multiple DMA cycles are granted

MR-5590



NOTE 1: ASSERTED HI THROUGHOUT THE TRANSACTION
NOTE 2: ASSERTED LO THROUGHOUT THE TRANSACTION
NOTE 3: THREE-STATED
NOTE 4: ASPI OCCURS AT THE END OF A HLT, PUP TRANSACTION, AND DURING A WAIT INSTRUCTION
NOTE 5: READY HAS NO EFFECT
NOTE 6: EXTRA TIMING PHASE (0D) ALWAYS PRESENT

Figure A-14 ASPI Transaction

	SYMBOL	PARAMETER	FUNCTION OF ^{t}CYC	MIN/MAX
1	^{t}CAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^{t}CDE	–CAS (T.E.) to next DAL<15:0> Address Enable	(T –18) = 115 ns	min
	^{t}CSP	–CAS (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
	^{t}CYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^{t}IHP	Input on AI<7:0> Hold Time from PI (T.E.)	0 ns	min
1	^{t}ISP	PI (L.E.) to Input on AI<7:0> Valid	(2T –167) = 100 ns	max
	^{t}PHC	PI Hold Time from –CAS (T.E.)	10 ns	min
1	^{t}PIP	PI Pulse Width	(2T –37) = 230 ns	min

1: Add T ns if in long bus cycle mode, then if RDY slips are initiated, add H*T ns, where:
T = 1/fop, H = number of RDY pulses times 3 if mode = normal, times 4 if mode = long bus cycle.

MR-5592

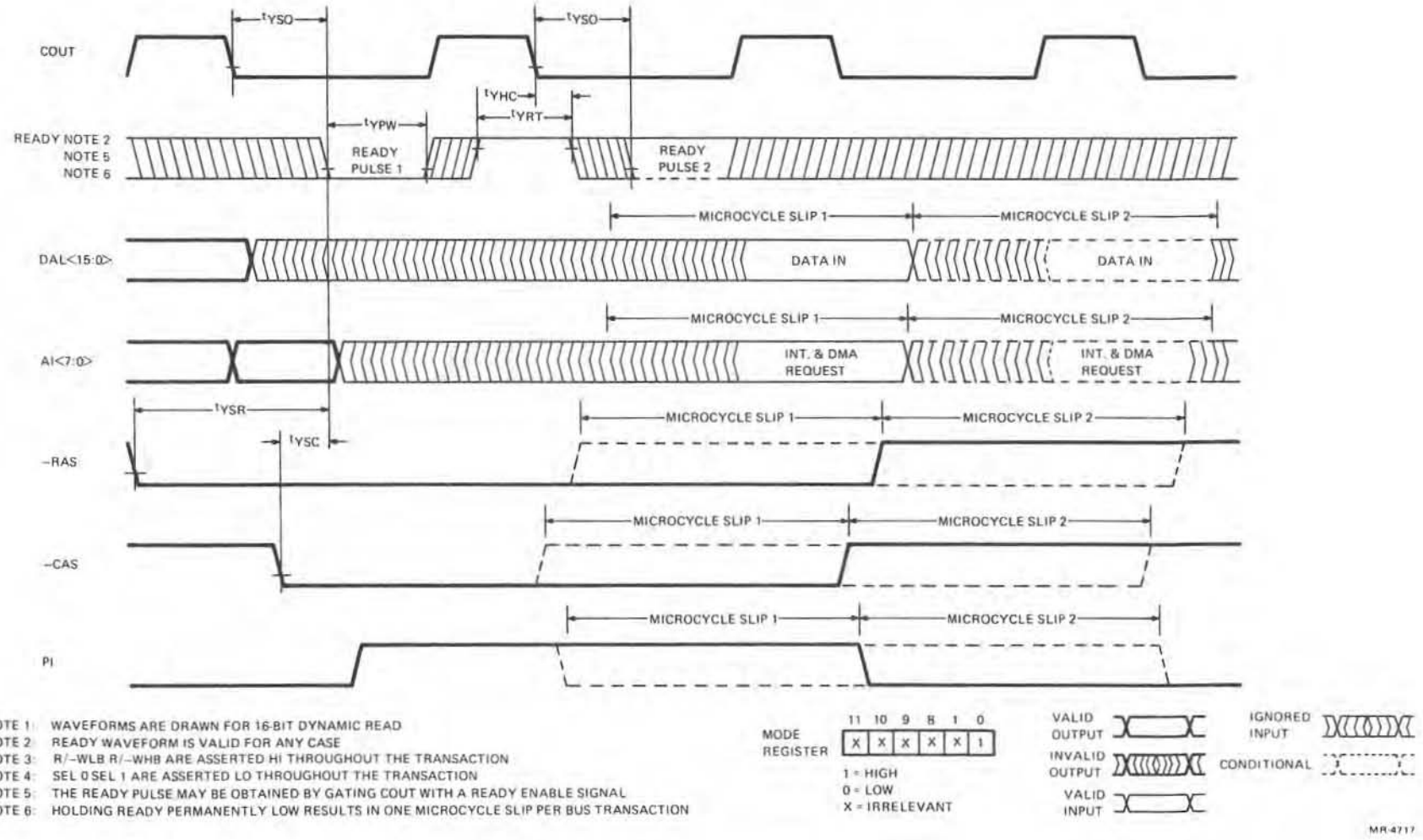


Figure A-15 Ready

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tCYC	XTL1, XTL0 Operating Period	T = 133 ns	max
1	^tYHC	Ready (L.E.) Hold Time from COUT (T.E.)	0 ns	min
2	^tYPW	Ready Unasserted Pulse Width	60 ns	min
1	^tYRT	Ready Recovery Time	60 ns	min
2	^tYSC	Ready (T.E.) Delay from –CAS (L.E.)	(2T –135) = 132 ns	max
2	^tYSO	Ready (L.E.) Delay from Pulsed Mode COUT (T.E.)	(2T –127) = 140 ns	max
2	^tYSR	Ready (T.E.) Delay from –RAS (L.E.)	(3T –100) = 300 ns	max

1: These timing parameters apply only to cases where multiple READY pulses are required; i.e., multiple microcycle slips.

2: READY is an edge-triggered input that is usually activated by asserting a low on its pin. However, READY is internally activated by the leading edge of –RAS if its pin has been asserted low before these edges.

MR-5593

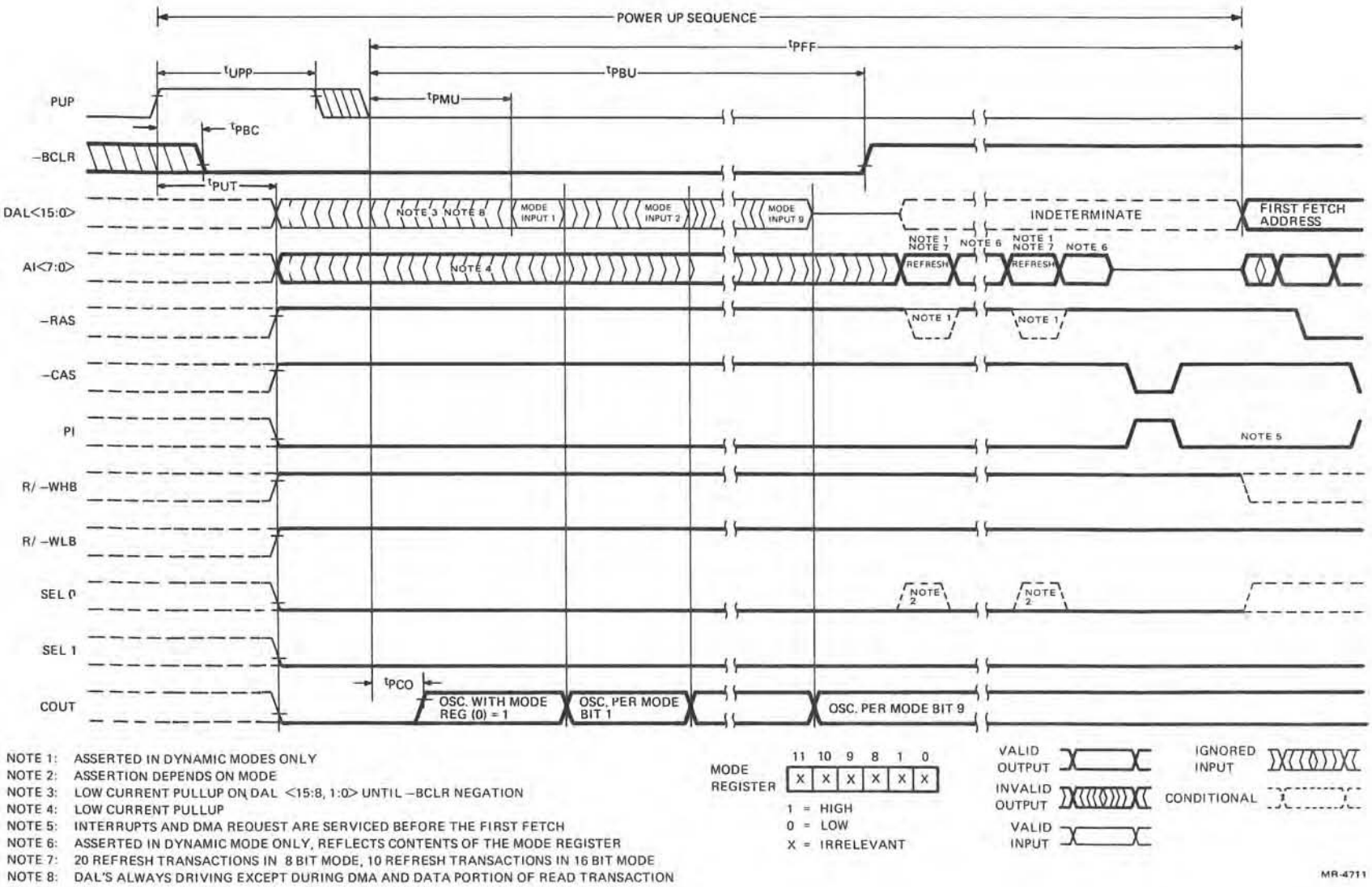
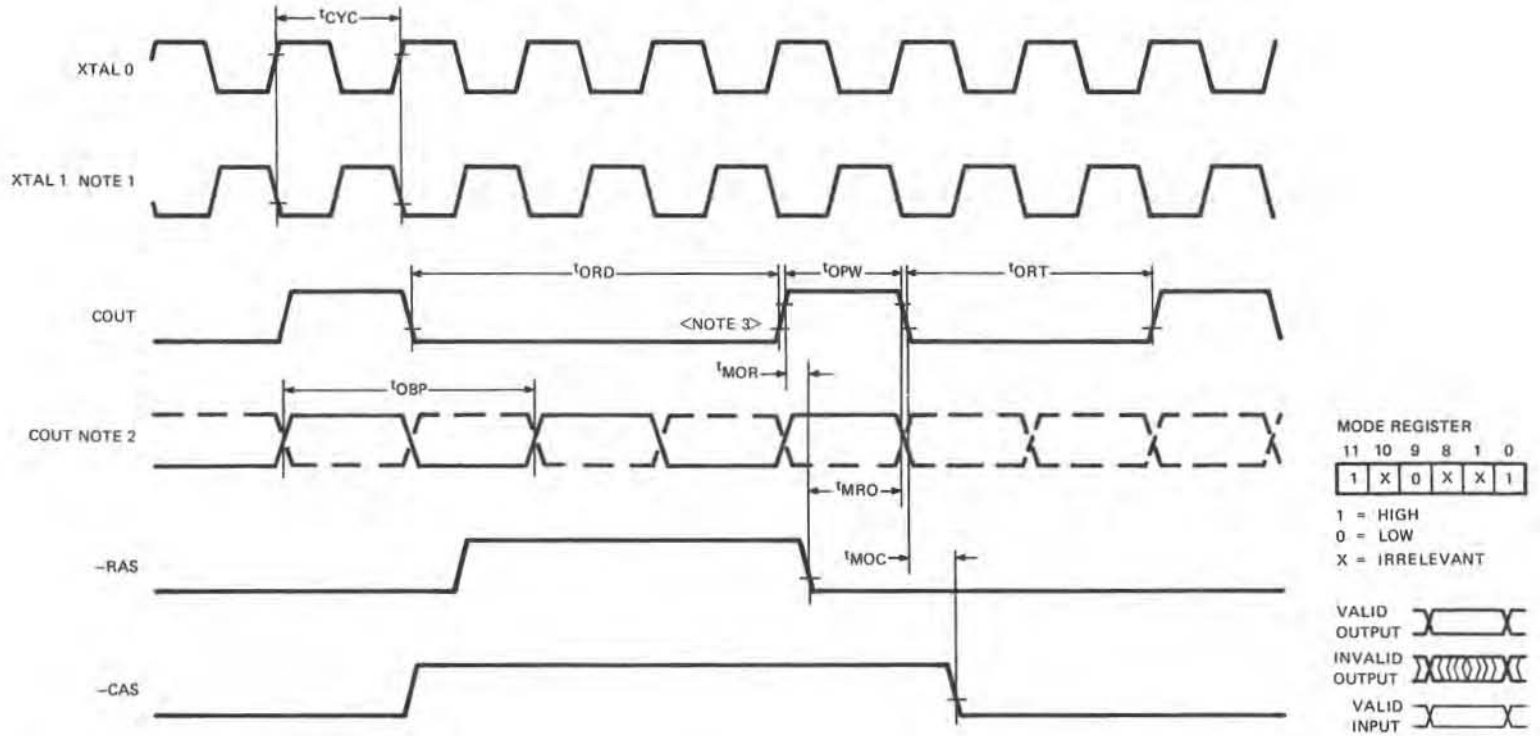


Figure A-16 Power Up

SYMBOL	PARAMETER	FUNCTION OF t_{CYC}	MIN/MAX
t_{CYC}	XTL1, XTL0 Operating Period	T = 133 ns	min
t_{PBC}	Power Up to –BCLR (L.E.) Set Up Time	100 ns	max
t_{PBU}	Power Up (T.E.) to –BCLR (T.E.)	99T = 13200 ns	min
		100T = 13333 ns	max
t_{PCO}	Power Up (T.E.) to COUT (L.E.)	(T + 60) 193 ns	max
t_{PFF}	Power Up (T.E.) to Beginning of First Instruction Fetch	295T = 39333 ns	min
		315T = 42000 ns	max
t_{PMU}	Power Up (T.E.) to Mode Bits on DAL<15:00> Valid	18T = 2400 ns	min
		19T = 2533 ns	max
t_{PUT}	Power Up (L.E.) to Output Pins Preset	250 ns	max
t_{UPP}	Power Up Time	100 μs	min

MR-5589



NOTE 1: XTAL1 MAY BE USED AS AN OUTPUT

NOTE 2: BUS MODE CLOCK (MODE REG <0>=0) PHASE DEPENDS ON MODE AND CODE EXECUTED AS SHOWN WITH CONDITIONAL OUTPUT. FOR LONG MICROCYCLE MODE AND BUS MODE CLOCK, THE VALID OUTPUT IS SHOWN (PHASE IS UNCONDITIONAL)

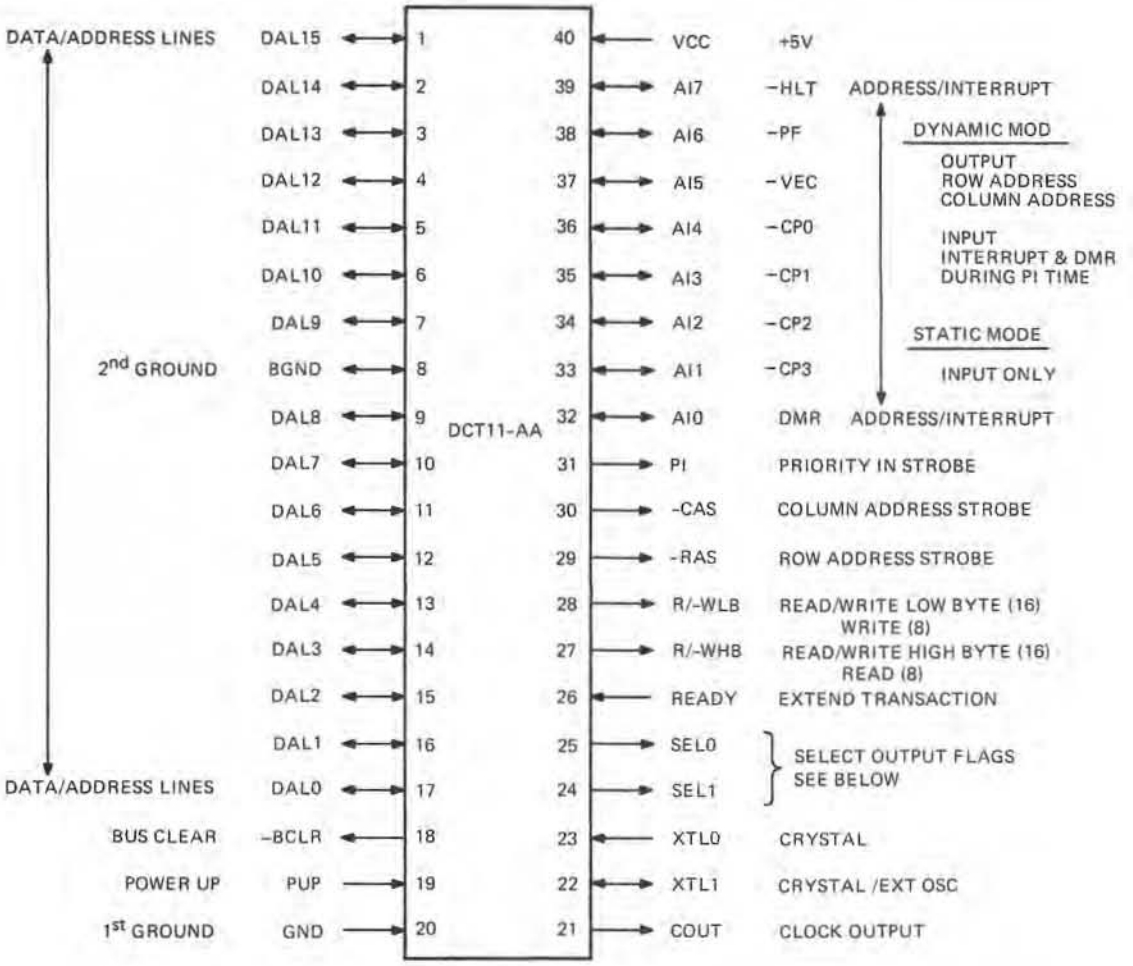
NOTE 3: EXTRA TIMING PHASE (0D) PRESENT. 0D IS PRESENT ONLY DURING ASPI, DMA, REFRESH AND IACK IN STANDARD MICROCYCLE MODE MR<1>=1. 0D IS PRESENT IN ALL MICROCYCLES IN LONG MICROCYCLE MODE MR<1>= 0.

MR-4714

Figure A-17 XTAL and COUT

SYMBOL	PARAMETER	FUNCTION OF ^{t}CYC	MIN/MAX
^{t}CYC	XTL1, XTL0 Operating Period	T = 133 ns	min
^{t}MOC	Pulse Mode COUT (T.E.) Set Up Time to –CAS (L.E.)	10 ns	min
^{t}MOR	Read/Write or DMA Pulse Mode COUT (L.E.) Set Up Time to –RAS (L.E.)	10 ns	min
^{t}MRO	Read/Write or DMA Pulse Mode COUT (T.E.) Hold Time to –RAS (L.E.)	(T –51) = 82 ns	min
^{t}OPW	Pulse Mode COUT Pulse Width	(T –33) = 100 ns	min
^{t}ORD	Pulse Mode COUT Recovery Time when Phase D is Present	(3T –37) = 363 ns	min
^{t}ORT	Pulse Mode COUT Recovery Time	(2T –37) = 230 ns	min

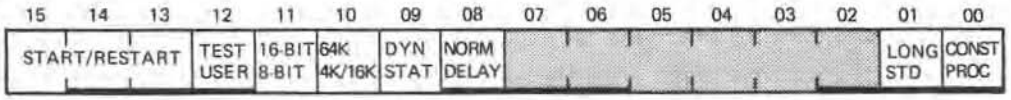
MR-5591



SELECT OUTPUT FLAGS

SEL<1>	SEL<0>	FUNCTION
L	L	READ/WRITE
L	H	REFRESH/FETCH
H	L	IACK
H	H	DMG

Figure A-18 DCT11-AA Pin Layout



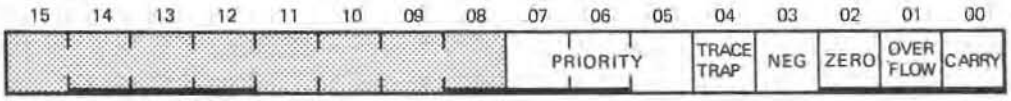
<15:13>	START/RESTART ADDRESS		
12	TESTER/USER MODE	08	NORMAL/DELAYED R/W
11	16-BIT/8-BIT BUS	<7:2>	RESERVED
10	64K/4K OR 16K MEMORY	01	LONG/STANDARD MICROCYCLE
09	DYNAMIC/STATIC MEMORY	00	CONSTANT/PROCESSOR MODE CLOCK

ADDRESS BITS <15:13>	START ADDRESS	RESTART ADDRESS
7	172000	172004
6	173000	173004
5	000000	000004
4	010000	010004
3	020000	020004
2	040000	040004
1	100000	100004
0	140000	140004

MR-4843

Figure A-19 Mode Register

PROCESSOR STATUS

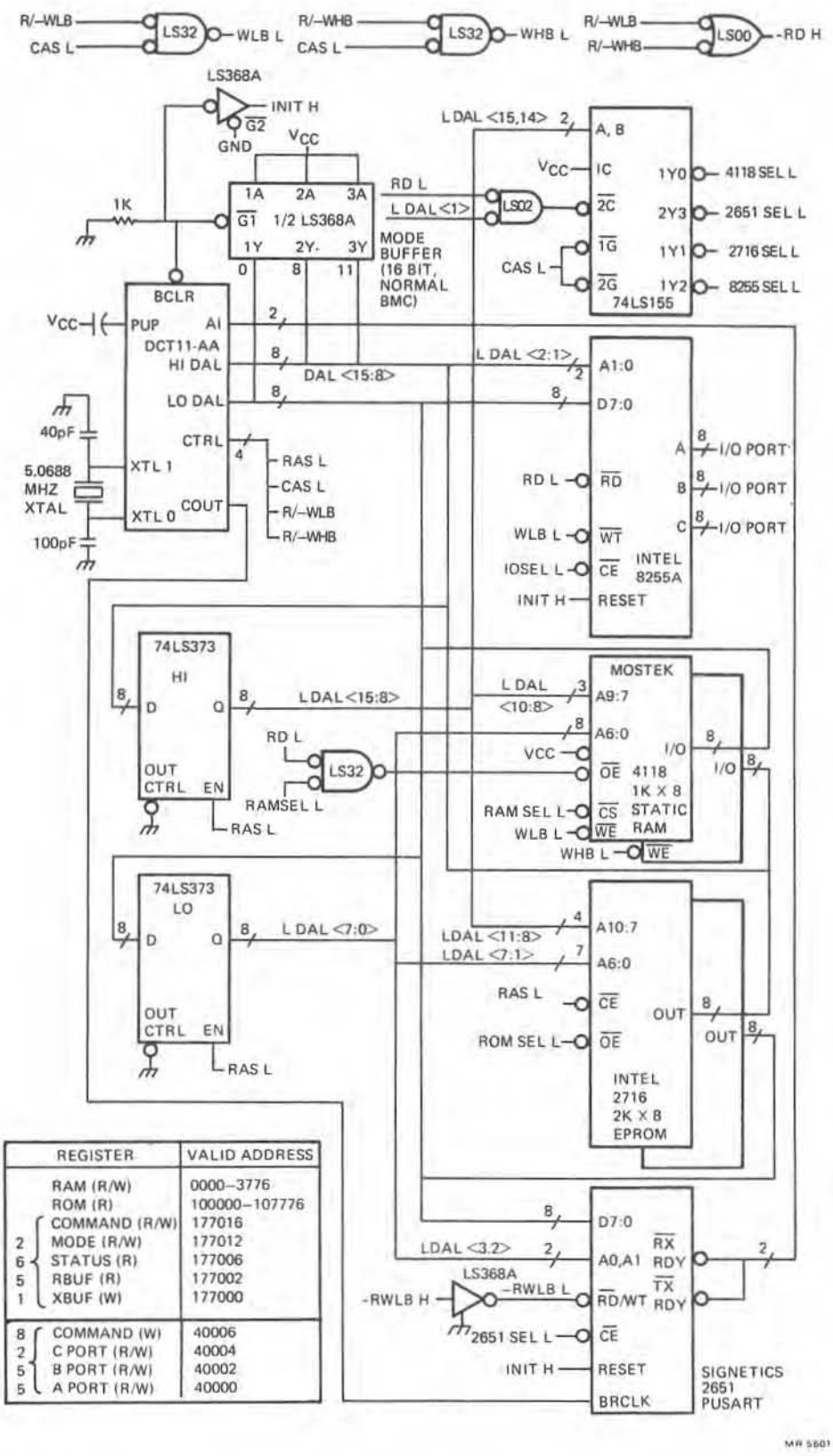


<15:8> READ AS ZEROS

03 NEGATIVE

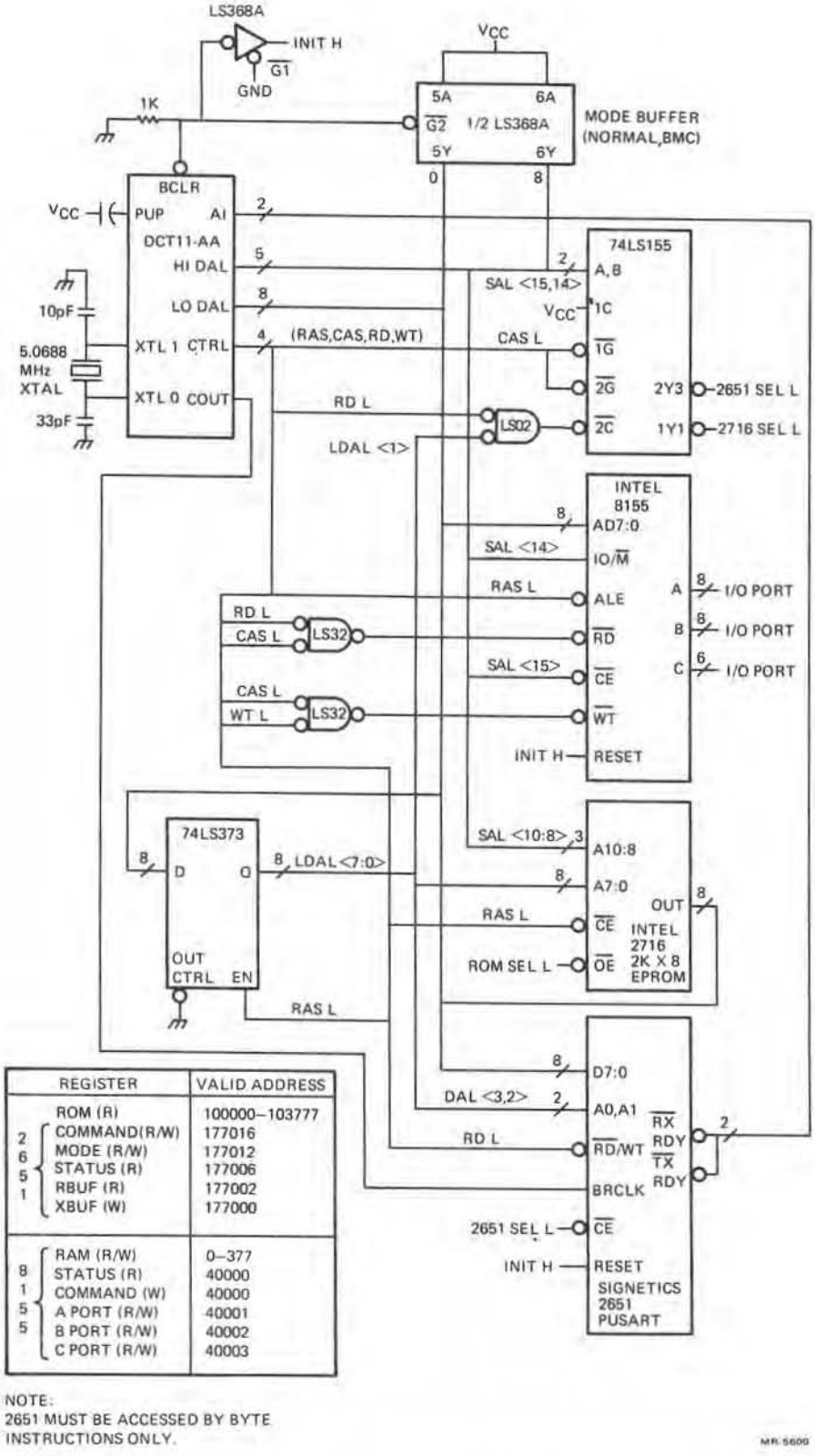
MR-5273

Figure A-20 Processor Status Word



REGISTER		VALID ADDRESS
	RAM (R/W)	0000–3776
	ROM (R)	100000–107776
2	COMMAND (R/W)	177016
2	MODE (R/W)	177012
6	STATUS (R)	177006
5	RBUF (R)	177002
1	XBUF (W)	177000
8	COMMAND (W)	40006
2	C PORT (R/W)	40004
5	B PORT (R/W)	40002
5	A PORT (R/W)	40000

Figure A-21 16-Bit Application



REGISTER		VALID ADDRESS
2	ROM (R)	100000–103777
6	COMMAND(R/W)	177016
5	MODE (R/W)	177012
1	STATUS (R)	177006
	RBUF (R)	177002
	XBUF (W)	177000
8	RAM (R/W)	0–377
1	STATUS (R)	40000
5	COMMAND (W)	40000
5	A PORT (R/W)	40001
	B PORT (R/W)	40002
	C PORT (R/W)	40003

NOTE:
2651 MUST BE ACCESSED BY BYTE INSTRUCTIONS ONLY.

MR-5600

Figure A-22 8-Bit Application

APPENDIX B

B.1 INTRODUCTION

This appendix is designed to make the user aware of variations between the DCT11-AA and other members of the PDP-11 family. These variations fall into the following major categories:

Addressing Modes
PDP-11 Instruction Set
DCT11-AA Instruction Execution Sequence on the Data Bus
Exceptions and Interrupts
Power Up.

The processors that are used in this appendix, for comparison to the DCT11-AA, are:

PDP-11/03
PDP-11/04
PDP-11/23
PDP-11/24
PDP-11/34A
PDP-11/40
PDP-11/44
PDP-11/45
PDP-11/70.

Table B-1 (which is found at the end of this appendix) indicates those functions which operate the same among the various PDP-11 processors.

B.2 ADDRESSING MODES

Most basic instructions operate the same from one PDP-11 processor to another. However, there are variations in the way the address is computed depending on the addressing mode being used. This section covers the variations in the addressing modes that are implemented by the DCT11-AA. An explanation of the symbols that are used in this section is found at the end of Chapter 6.

When executing a double operand instruction the same general purpose register may be used in both the source and destination fields of the instruction. When the same registers are used in the DCT11-AA, PDP-11/23, PDP-11/24 and PDP-11/40, the results vary from other PDP-11 processors.

B.2.1 Modes 2 and 4

If the addressing mode of the destination operand is autoincrement (mode 2) the contents of the register are incremented by 2 before being used as the source operand. If the addressing mode of the destination operand is autodecrement (mode 4) the contents of the register are decremented by 2 before being used as the source operand.

In the other processors covered in this appendix, the initial content of the source register is not modified and is used as the source operand.

The following is an example of an autoincrement (mode 2). Register 0 contains 1000 (octal).

```
MOV  R0, (R0)+
```

In the DCT11-AA, the quantity 1002 is moved to location 1000.

In the other processors, the quantity 1000 is moved to location 1000.

The following is an example of an autodecrement (mode 4). Register 0 contains 1000 (octal).

```
MOV  R0, -(R0)
```

In the DCT11-AA, the quantity 776 is moved to location 776.

In the other processors, the quantity 1000 is moved to location 776.

B.2.2 Modes 3 and 5

If the addressing mode of the destination operand is autoincrement deferred (mode 3) the contents of the register are incremented by 2 before being used as the source operand. If the addressing mode of the destination operand is autodecrement deferred (mode 5) the contents of the register are decremented by 2 before being used as the source operand.

In the other processors covered in this appendix, the initial content of the source register is not modified and is used as the source operand.

The following is an example of an autoincrement deferred (mode 3). Register 0 contains 1000 (octal) and location 1000 contains 2000 (octal).

```
MOV  R0, @(R0)+
```

In the DCT11-AA, the quantity 1002 is moved to location 2000.

In the other processors, the quantity 1000 is moved to location 2000.

The following is an example of an autodecrement deferred (mode 5). Register 0 contains 1000 (octal) and location 776 contains 2000 (octal).

MOV R0, @-(R0)	In the DCT11-AA, the quantity 776 is moved to location 2000. In the other processors, the quantity 1000 is moved to location 2000.

B.2.3 Using the PC Contents as the Source Operand

```
Op Code PC, X(R)
Op Code PC, @X(R)
Op Code PC, @A
Op Code PC, A
```

In the above operations, the resultant source operand is the value of the location of the op code plus 4. This is true for the DCT11-AA, PDP-11/23, PDP-11/24, and PDP-11/40. This varies from other PDP-11 processors covered in this appendix where the source operand is the value of the location of the op code plus 2.

In the following example the PC contains the value 1000 (octal). Location 1002 contains the offset value 2. R0 contains the value 2000 (octal).

MOV PC, 2(R0)	In the DCT11-AA, the value 1004 is moved to location 2002. In the other processors, the value 1002 is moved to location 2002.

The final source operand is the same (1004) for all of the addressing modes explained above.

NOTE

The use of the above forms of addressing should be avoided. The MACRO-11 assembler generates an error code (Z) which is printed in the listing. This occurs in each instruction when the addressing mode is found not to be compatible among all members of the PDP-11 family.

B.2.4 Jump (JMP) and Jump to Subroutine (JSR) Instructions

```
JMP  %R
JSR  reg, %R
```

When programming JMP and JSR instructions care must be taken in selecting the destination mode of the instruction. When mode 0 is selected an error condition is created and the DCT11-AA traps through location 4 of the trap vectors (refer to exceptions and interrupt servicing). This is true of all PDP-11 processors except the PDP-11/45.

The PDP-11/45 when executing this instruction causes a trap through memory location 10.

B.3 PDP-11 INSTRUCTION SET

The DCT11-AA implements the basic PDP-11 instruction set. The PDP-11 instruction set offers a wide choice of operations, such that a single instruction often does a task that would need many in other computers. PDP-11 instructions, allow byte and word addressing in both single and double operand formats. This saves memory space and simplifies the implementation of control and communications applications.

Instruction set variations fall into these catagories:

Instructions not common to all PDP-11s
Basic instruction execution
Instructions not executed
Effect of the T-bit (instruction trace trap).

B.3.1 Instructions not Common to all PDP-11s

As the number of PDP-11 processor types increased, instructions that were not included in the basic instruction set were added. The DCT11-AA includes the following instructions:

MFPT (move from processor type)
MFPS (move byte from processor status)
MTPS (move byte to processor status).

B.3.1.1 MFPT Instruction --

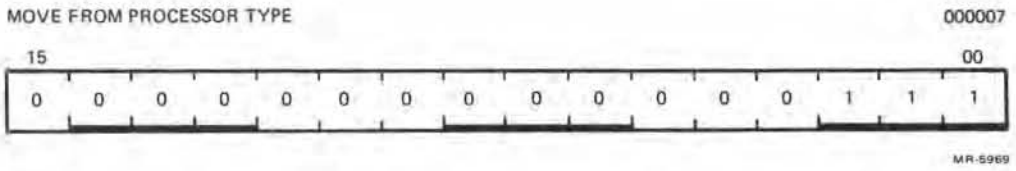


Figure B-1 MFPT Instruction

Operation: R0← processor type

Condition Codes: Unaffected

The DCT11-AA, PDP-11/23, PDP-11/24, and PDP-11/44 are the only processors that execute the MFPT instruction. The model code is placed in the low byte of register R0 indicating to the system software the processor type. Table B-2 shows the codes assigned to identify the processor in use.

Table B-2 Processor Codes

Model Code	Processor Type
4	DCT11-AA
3	PDP-11/23 or PDP-11/24
1	PDP-11/44.

NOTE

The PDP-11/23 and PDP-11/24 are controlled by the same processor and have the same model code.

B.3.1.2 MFPS Instruction --

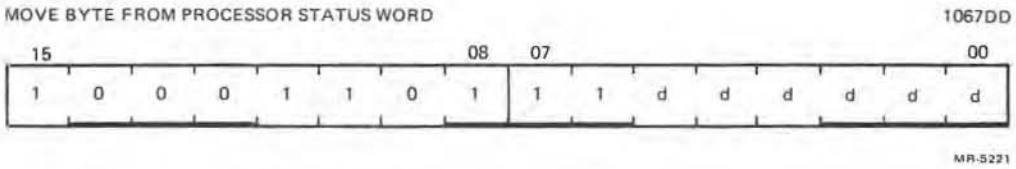


Figure B-2 MFPS Instruction

Operation: (dst)←PS
dst lower 8 bits

Condition Codes: N: set if PS bit 7 = 1; cleared otherwise
Z: set if PS <0:7> = 0; cleared otherwise
V: cleared
C: not affected

The low byte of the PS is used as the source operand. The destination operand is treated as a byte.

The DCT11-AA, PDP-11/03, PDP-11/23, PDP-11/24, and PDP-11/34 implement this instruction to save the processor status register (PS) without directly accessing the PS on the data/address bus.

NOTE

The DCT11-AA is not restricted from having memory or a device at the PS address 177776. In addition the DCT11-AA does not recognize that an error has occurred when addressing bus locations that do not contain memory (refer to the exceptions and interrupts section). To attempt to read or write data at address 177776 expecting the PS will cause unpredictable results.

B.3.1.3 MTPS Instruction --

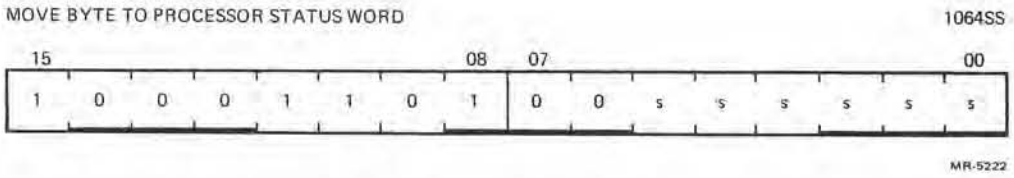


Figure B-3 MTPS Instruction

Operation: PS←(src)

Condition Codes:
N: set according to effective source operand
Z: set according to effective source operand
V: set according to effective source operand
C: set according to effective source operand

The source operand is treated as a byte and the destination operand is always the low byte of the PS. The source operand is not affected by the MTPS instruction.

NOTE
The T-bit (bit 4 of the PS) cannot be set with the MTPS instruction.

The DCT11-AA, PDP-11/03, PDP-11/23, PDP-11/24, and PDP-11/34 implement this instruction in order to load the low byte of the processor status register without directly addressing the PS on the data/address bus.

NOTE
When developing software for the DCT11-AA on PDP-11 systems that have memory management, the priority bits of the PS (bits <7:5>) may not be affected. Refer to the appropriate processor handbook.

B.3.2 Basic Instruction Execution

The DCT11-AA executes all basic PDP-11 instructions except MARK. Some instructions vary in execution from other PDP-11 processors. These instructions are covered in this section.

B.3.2.1 HALT Instruction --

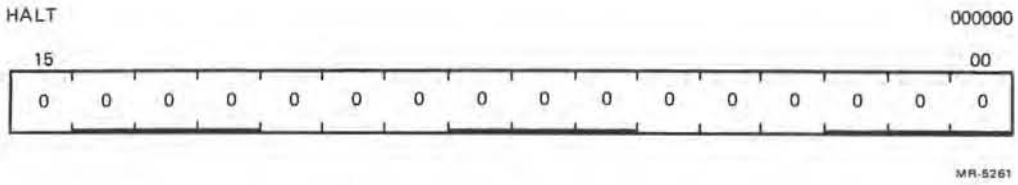


Figure B-4 HALT Instruction

Condition Codes: Unaffected

When other PDP-11 processors covered in this appendix execute the HALT instruction, it causes processor operation to cease. Control is given to the console if one is present or to a console microprogram within the processor. The DCT11-AA has no console or console microprogram. The DCT11-AA executes a HALT instruction like a trap. The DCT11-AA pushes the current PS and PC onto the stack. The PC is loaded with the value of the restart address (power up address + 4), and the PS is loaded with a value of 340 to inhibit interrupts. The power up and restart addresses are explained in the power up section of this appendix.

NOTE

When developing software for the DCT11-AA on PDP-11 systems that have memory management, the trap sequence when executing a HALT instruction is different. Refer to the appropriate processor handbook.

B.3.2.2 RESET Instruction --

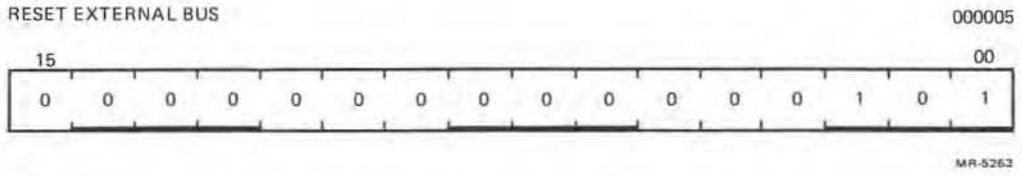


Figure B-5 RESET Instruction

Condition Codes: Unaffected

The DCT11-AA RESET instruction causes the assertion of the Bus Clear (-BCLR) signal. An Assert Priority In (ASPI) transaction takes place to input interrupt and DMA information. The condition codes and general purpose registers R0 - R5, SP, and PC are not affected. The -BCLR signal is asserted low for a minimum of 8.4 microseconds followed by a minimum 150 nanosecond pause. No processor operations are performed during this pause. The next programmed instruction is executed after the pause. Timing for the -BCLR signal is a function of the processor clock or crystal frequency.

If the power fail interrupt is asserted during the RESET instruction, it is not recognized until the instruction has completed the -BCLR sequence. This is also true of the PDP-11/03, PDP-11/23, and PDP-11/24.

When a power fail interrupt occurs during a RESET instruction in the PDP-11/04 and PDP-11/34 it is a fatal error and no power down sequence occurs. The PDP-11/44, PDP-11/45, and PDP-11/70 RESET instruction is aborted in the event of a power fail.

B.3.3 Instructions not Executed

The DCT11-AA does not execute the PDP-11 instructions and op codes listed in Table B-3. If an attempt is made to execute these instructions the processor traps through location 10.

Table B-3 PDP-11 Instructions not Executed

Op Code	Mnemonic	Op Code	Mnemonic
00 00 10	reserved	07 0R SS	MUL
		07 1R SS	DIV
00 00 77		07 2R SS	ASH
		07 3R SS	ASHC
00 02 10	reserved	07 50 0R	FADD
		07 50 1R	FSUB
00 02 27		07 50 2R	FMUL
		07 50 3R	FDIV
00 02 3N	SPL	07 50 40	unused
00 64 NN	MARK		
00 65 SS	MFPI	07 67 77	
00 66 DD	MTPI	10 65 SS	MFPD
		10 66 DD	MTPD
00 70 00	reserved	17 00 00	FPP Instructions
00 77 77		17 77 77	

B.3.4 Effect of the T-Bit (Instruction Trace Trap)

The processor status register contains information on the current status of the CPU. This information includes:

the current processor priority for interrupts

the condition codes describing the result of the last instruction

a bit that indicates a trap occurs after the execution of the current instruction.

The DCT11-AA does not allow the T-bit to be set directly. This is true of all processors covered by this appendix except the PDP-11/04. Only indirect references to the PS can cause the T-bit to be set. Such references occur when executing:

RTI (return from interrupt) instruction
RTT (return from trap) instruction
trap instructions
exceptions or interrupts.

If the RTI instruction causes the T-bit to be set, the T-bit trap is taken through location 14 **BEFORE** executing the next instruction. If the RTT instruction causes the T-bit to be set, the t-bit trap is taken **AFTER** executing the next instruction. The above is true for all processors covered in this appendix.

The DCT11-AA and all processors (except the PDP-11/45 and PDP-11/70) acknowledge the T-bit trap before acknowledging an interrupt that occurs during instruction execution. The PDP-11/45 and PDP-11/70 give the pending interrupt priority over the T-bit trap.

If a WAIT instruction is executed and the T-bit is set, the DCT11-AA sequences out of the WAIT. After the T-bit is serviced the instruction following the WAIT is executed. This is true of all processors except the PDP-11/03, PDP-11/45 and PDP-11/70. These processors return to the WAIT until an interrupt occurs.

B.4 DCT11-AA INSTRUCTION EXECUTION SEQUENCE ON THE DATA BUS

Each PDP-11 instruction executed by the DCT11-AA performs a number of transactions on the data/address bus. The number and type of transaction is determined by the instruction being executed. Every instruction that ends in a write transaction to a memory location is always preceded by a read transaction from the same location.

B.4.1 Examples Using the Move (MOV) Instruction

In all other processors covered in this appendix, the MOV instruction consists of the following bus transactions:

The processor fetches the opcode of the instruction
The processor then obtains the source operand
The destination operand is computed
The source operand is written into the destination address.

In the DCT11-AA, the MOV instruction operates similar to the other processors except for the last bus transaction. After the destination address has been computed, the DCT11-AA reads from the destination address before writing to that address. Clear (CLR) and sign extend (SXT) follow a similar bus sequence.

This bus sequence is important when connecting the DCT11-AA directly to interface devices. For example the Intel 8251A serial interface contains data input and output registers at the same bus address. When the data has been assembled in the input register, the signal (RxRDY) is generated to indicate the receiver is ready. The RxRDY signal is cleared when the processor reads the input register. During a write operation to the Intel 8251A data registers, the DCT11-AA first reads the input register then writes to the output register. This may result in the RxRDY signal being cleared. When RxRDY is cleared in this manner, data may be lost.

NOTE

When connecting interface devices to the DCT11-AA that do not have DEC standard bus addresses and status registers, it is important to know the device addresses and bit patterns in the status register.

B.5 EXCEPTIONS AND INTERRUPTS

The DCT11-AA has a flexible hardware and software interrupt structure. Hardware interrupts cause the DCT11-AA to temporarily suspend program operation to execute a service routine. Software interrupts call service routines required by the program. Software interrupts occur when executing trap instructions or when the trace bit is set in the processor status register. When the service routine is completed, program execution is resumed.

The DCT11-AA services calls and interrupts in the following order of priority:

HALT (non-maskable interrupt or instruction)
Power Fail (non-maskable interrupt)
Trace Trap (T-bit)
CP<3:0> priority 7 (interrupt)
CP<3:0> priority 6 (interrupt)
CP<3:0> priority 5 (interrupt)
CP<3:0> priority 4 (interrupt)
Trap instruction call.

The DCT11-AA supports a vectored interrupt structure with four priority levels. Interrupts are input on four CODED priority lines (CP<3:0>). The value encoded on these lines indicates an interrupt request is pending from one of fifteen devices on one of four priority levels. Interrupts are maskable in that the priority code of the interrupting device must exceed the value in the PS (bits <7:5>) or the interrupt is not acknowledged.

The DCT11-AA also has two non-maskable interrupt lines, HALT and Power Fail (PF). Assertion of either of these lines interrupts the processor regardless of the priority level in the PS. HALT and PF have individual input lines. The non-maskable interrupt HALT is not associated with an interrupt vector. When a HALT interrupt occurs, the current PS and PC are pushed on the stack, the PC is loaded with the restart address, and the PS is loaded with 340.
A device requests service by asserting one or more of theh CP lines (CP<3:0>). If the priority of the requesting device is higher than that of the processor, the interrupt is acknowledged and the device is serviced at the completion of the current instruction.

NOTE

If the T-bit is set in the PS the trace trap is taken before servicing the interrupt. The T-bit must not be set in the PS word of the T-bit trap vector. If this occurs continuous T-bit trapping will result.

The current state of the machine is saved so that program execution may continue after completion of the service routine. The contents of the program counter (address of the next instruction) and the PS are pushed onto the system stack. The new contents of the PS and PC are loaded from two consecutive memory locations called "vector locations". The first location contains the address of the service routine and the second contains the new PS value. All information in the vector locations must be loaded under program control.

NOTE

The device requesting an interrupt must remove the request when it receives an interrupt acknowledge (IACK) from the DCT11-AA. If the request is not removed and the PS word of the service vector does not contain a prority level as high or higher than that of the interrupt request, the request continues to be serviced until the stack is full. This causes a loss of program and data.

The vector address is provided, during an interrupt acknowledge transaction, by either a fixed table stored in the DCT11-AA (internal vector address) or by the interrupting device (external vector address). Table B-4 indicates the internal vectors assigned to interrupt priority codes.

Table B-4 Interrupt Priority Codes

PRIORITY LEVEL		VECTOR ADDRESS	
		New PC at	New PS at
non-maskable	HALT	restart address	340
non-maskable	PF	24	26
7		140	142
7		144	146
7		150	152
7		154	156
6		100	102
6		104	106
6		110	112
6		114	116
5		120	122
5		124	126
5		130	132
5		134	136
4		60	62
4		64	66
4		70	72

B.5.1 Bus Errors

The DCT11-AA does not support bus errors. Most PDP-11 processors indicate that an error has occurred and interrupt program execution when:

a word instruction executes with an odd address (odd address error)

a non-existent memory location is accessed (non existent memory (NXM) error)

the stack value approaches the vector location area (stack overflow error).

If a word instruction is executed and the source or destination address is odd, the least significant address bit is ignored and a word operation is performed at the even address.

If the DCT11-AA attempts to read or write a non-existent memory location, the transaction is completed and program execution continues. If the transaction is a read, undifined data is received. A write to a non-exsistent memory location outputs data onto the data address lines as if memory is present and the data is lost.

No warning is given by the DCT11-AA if the hardware stack pointer (SP) decrements below 377 (octal). This can cause unpredictable results when the contents of the vector addresses are changed.

NOTE

It is important to leave enough room for the stack area so the vector locations will not be destroyed.

B.5.2 Internal Register Access

None of the internal registers of the DCT11-AA are directly accessable as memory locations to the programmer. All transactions involving these registers are done internally by the DCT11-AA. The addresses assigned to these registers by other PDP-11 processors are within the 16-bit address space of the DCT11-AA. These addresses can be used as memory locations or as peripheral device registers.

NOTE

The PS, general purpose registers R0 - R5, SP, and PC are examples of registers that cannot be directly accessed as memory locations by the programmer.

B.6 POWER UP

The DCT11-AA is a flexible microprocessor which can be adapted to many different applications. The power up process is used to set one of eight different start/restart addresses. The instruction in the start address is always the first executed after power is applied to the DCT11-AA. During power up or when executing a RESET instruction, the DCT11-AA loads an internal register with a three bit code which represents one of the eight start/restart address pairs. Table B-5 shows the start/restart addresses.

Table B-5 Start/Restart Addresses

START ADDRESS (Used at Power Up)	RESTART ADDRESS (Used for HALT)
000000	000004
010000	010004
020000	020004
040000	040004
100000	100004
140000	140004
172000	172004
173000	173004

NOTE

The start address is only used at the time power is applied to the DCT11-AA. A RESET instruction loads the mode register. It does not cause the start address to be loaded into the PC.

When a HALT instruction is executed or a hardware halt interrupt is asserted, the value of the PS and PC is placed on the hardware stack. The DCT11-AA loads the PC with the restart address and sets the PS to 340.

SYMBOLS AND NOTATION

The following symbols are used in the explanation of the various modes:

%R	Mode 0 addressing. The contents of the register are to be used as the source operand.
(R)+	Mode 2 addressing. The register contents are to be used as the address of the destination operand and then the register contents are to be incremented by 2 (autoincrement).
-(R)	Mode 4 addressing. The register contents are to be decremented by 2 and then used as the address of the destination operand (autodecrement).
@(R)+	Mode 3 addressing. The contents of the register are to be used as an address of the address of the destination operand. The contents of R are incremented by 2 (autoincrement deferred).
@-(R)	Mode 5 addressing. The contents of the register are to be decremented by 2 and then used as an address of the address of the destination operand (autodecrement deferred).
PC	Program counter mode 0 addressing. The contents of the program counter are to be used as the source operand.
X(R)	Indexed addressing (register mode 6). The value of X is added to the contents of register R to form the address of the destination operand.
@X(R)	Indexed deferred addressing (register mode 7). In this mode the value of X is added to the contents of register R to form the address of the address of the destination operand.
A	Program counter relative addressing. Relative addressing uses the contents of the location following the op code as the address of the destination operand.
@A	Program counter relative defferred addressing. Relative defferred addressing uses the contents of the location following the op code as the address of the address of the destination operand.

Table B-1 Software Differences

	Activity	PDP-11 Family Machines								
		T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
1.	OPR %R,(R)+ or OPR %R,-(R) using the same register as both source and destination: contents of R are incremented (decremented) by 2 before being used as the source operand.	X	X					X	X	
	OPR %R,(R)+ or OPR %R,-(R) using the same register as both register and destination: initial contents of R are used as the source operand.			X	X	X	X			X
2.	OPR %R,@(R)+ or OPR %R,@-(R) using the same register as both source and destination: contents of R are incremented (decremented) by 2 before being used as the source operand.	X	X					X	X	
	OPR %R,@(R)+ or OPR %R,@-(R) using the same register as both source and destination: initial contents of R are used as the source operand.			X	X	X	X			X
3.	OPR PC,X(R); OPR PC, @X(R); OPR PC,@A; OPR PC,A: Location A will contain the PC of OPR +4.	X	X					X	X	
	OPR PC,X(R); OPR PC,@X(R); OPR PC,A; OPR PC,@A: Location A will contain the PC or OPR +2.			X	X	X	X			X
4.	JMP (R)+ or JSR reg,(R)+: Contents of R are incremented by 2, then used as the new PC address.						X	X		

Table B-1 Software Differences (Cont)

	Activity	PDP-11 Family Machines								
		T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
	JMP (R)+ or JSR reg,(R)+: Initial contents of R are used as the new PC.	X	X	X	X	X			X	X
5.	JMP %R OR JSR reg,%R traps to 4 (illegal instruction).	X	X	X	X	X	X	X	X	
	JMP %R or JSR reg,%R traps to 10 (illegal instruction).									X
6.	SWAB does not change V.							X		
	SWAB clears V.	X	X	X	X	X	X		X	X
7.	Register addresses (177700 - 177717) are valid program addresses when used by the CPU.						X			
	Register addressed (177700 - 177717) time out when used as a program address by the CPU. Can be addressed under console operation. Note: Addresses cannot be addressed under console for LSI-11 or 11-23.		X			X		X	X	X
	Register addresses (177700 - 177717) are handled as regular memory addresses (in the BSIO page). No internal regsiters are addressable from either the bus or the console.	X								
8.	BASIC INSTRUCTIONS noted in PDP-11 processor handbook.	X	X	X	X	X	X	X	X	X
	MFPT (move from processor type)	X								
	SOB, RTT, SXT instructions.	X	X	X	X	X			X	X

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
MARK instruction.		X	X	X	X			X	X
ASH, ASHC, DIV, MUL instructions.		X		X	X			X	X
XOR instruction.	X	X		X	X			X	X
The external option KE11-A provides MUL, DIV and SHIFT operations in the same data format.						X	X		
The KE11-E (Expansion Instruction Set) provides the instructions MUL, DIV, ASH, and ASHC. These new instructions are 11/45 compatible.								X	
The KE11-F adds unique stack ordered floating point instructions: FADD, FSUB, FMUL, FDIV.								X	
The KEV-11 adds EIS/FIS instructions.					X				
SPL instruction.									X
9. Power fail during RESET instruction is not recognized until after the instruction is finished (70 milliseconds). RESET instruction consists of 70 milliseconds pause with INIT occurring during first 20 milliseconds.							X	X	
Power fail immediately ends the RESET instruction and traps if an INIT is in progress. A minimum INIT of 1 microsecond occurs if instructions aborted.									X

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
Power fail acts the same as 11/45 (22 milliseconds with about 300 nanoseconds minimum). Power fail during RESET fetch is fatal with no power down sequence.			X	X		X			
RESET instruction consists of 10 microseconds of INIT followed by a 90 microsecond pause. Power fail not recognized until the instruction is complete.		X			X				
RESET instruction consists of a minimum 8.4 microseconds followed by a minimum 150 nanosecond pause. Power fail is not recognized until the instruction is complete.	X								
10. No RTT instruction.						X	X		
If RTT sets the T-bit, the T-bit trap occurs after the instruction following RTT.	X	X	X	X	X		X	X	
11. If RTI sets the T-bit, the T-bit trap is acknowledged after the instruction following RTI.						X	X		
If RTI sets the T-bit, the T-bit trap is acknowledged immediately following RTI.	X	X	X	X	X		X	X	

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
11A. When operating with the T-bit set (e.g. single stepping) no interrupt requests will be serviced. At the end of instruction execution, the T-bit has higher priority than interrupt requests. Once in the T-bit service routine, other interrupts are blocked to ensure no unexpected occurrences. When the RTT instruction is executed to leave the service routine, interrupts will not be serviced if the T-bit is set in the new PS popped off the stack. The user will, therefore, not see any interrupt requests he is expecting.	X								
12. If an interrupt occurs during an instruction that has the T-bit set, the T-bit trap is acknowledged before the interrupt.	X	X	X	X	X	X	X	X	
If an interrupt occurs during an instruction and the T-bit is set, the interrupt is acknowledged before the T-bit trap.									X
13. T-bit trap will sequence out of WAIT instruction.	X	X	X	X		X	X	X	
T-bit trap will not sequence out of WAIT instruction. Waits until an interrupt.					X				X
14. Explicit reference (direct access) to PS can load the T-bit. Console can also load the T-bit.			X			X	X		
Only implicit references (RTI, RTT, traps and interrupts) can load the T-bit. Console cannot load the T-bit.	X	X		X	X			X	X

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
15. Odd address/nonexistent references using the SP cause a HALT. This is a case of double bus error with the second error occurring in the trap servicing the first error. Odd address trap not in LSI-11 or 11-23.		X	X	X	X	X	X		
Odd address/nonexistent references using the SP cause a fatal trap. On bus error in trap service, new stack created at 0/2.								X	X
Odd address/nonexistent references using the SP do not trap.	X								
16. The first instruction in an interrupt routine will not be executed if another interrupt occurs at a higher priority level than assumed by the first interrupt.	X	X	X	X	X	X		X	X
The first instruction in an interrupt service is guaranteed to be executed.							X		
17. 8 General-purpose registers.	X	X	X	X	X	X	X	X	
16 General-purpose registers.									X
18. PSW address, 177776, not implemented. Must use new instructions, MTPS (Move to PS) and MFPS(Move from PS).	X				X				
PSW address implemented. MTPS and MFPS not implemented.			X			X	X	X	X

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
PSW address and MTPS and MFPS implemented.		X		X					
19. Only one interrupt level (BR4) exists.					X				
Four interrupt levels exist.		X	X	X		X	X	X	X
Four interrupt levels exist encoded in four lines.	X								
20. Stack overflow not implemented.	X				X				
Some sort of stack overflow implemented.		X	X	X		X	X	X	X
21. Odd address trap not implemented.	X	X			X				
Odd address trap implemented.			X	X		X	X	X	X
22. FMUL and FDIV instructions implicitly use R6 (one push and pop); hence R6 must be set up correctly.					X				
FMUL and FDIV instructions do not implicitly use R6.								X	
23. Due to their execution time, EIS instructions can abort because of a device interrupt.					X				
EIS instructions do not abort because of a device interrupt.		X						X	X
24. Due to their execution time, FIS instructions can abort because of a device interrupt.					X			X	
25. EIS instructions do a DATIP and DATO bus sequence when fetching source operand.					X				

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
EIS instructions do a DATI bus sequence when fetching source operand.		X						X	X
26. MOV instruction does just a DATO bus sequence for the last memory cycle.		X	X	X	X			X	X
MOV instruction does a DATIP and DATO bus sequence for the last memory cycle.			X			X	X		
MOV instruction does a READ (DATI) and a WRITE (DATO) bus sequence for the last memory cycle.	X								
27. If PC contains nonexistent memory address and a bus error occurs, PC will have been incremented.		X	X	X	X	X	X		X
If PC contains nonexistent memory address and a bus error occurs, PC will be unchanged.								X	
Does not support bus errors.	X								
28. If register contains nonexistent memory address in mode 2 and a bus error occurs, register will be incremented.		X			X	X	X	X	X
If register contains nonexistent memory address in mode 2 and a bus error occurs, register will be unchanged.			X	X					
Does not support bus errors.	X								
29. If register contains an odd value in mode 2 and a bus error occurs, register will be incremented.		X			X			X	X

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
If register contains an odd value in mode 2 and a bus error occurs, register will be unchanged.			X	X		X	X		
Does not support bus errors.	X								
30. Condition codes restored to original values after FIS interrupt abort (EIS doesn't abort on 35/40).								X	
Condition codes that are restored after EIS/FIS interrupt abort are indeterminate.					X				
31. Op codes 075040 through 075377 unconditionally trap to 10 as reserved Op codes.	X	X	X	X		X	X	X	X
If KEV-11 option is present, Op codes 075040 through 075377 perform a memory read using the register specified by the low order 3 bits as a pointer. If the register contents is a nonexistent address, a trap to 4 occurs. If the register contents is an existent address, a trap to 10 occurs if user microcode is not present. If no KEV-11 options is present, a trap to 10 occurs.					X				
32. Op codes 210 through 217 trap to 10 as reserved Op codes.	X	X	X	X		X	X	X	X
Op codes 210 through 217 are used as a maintenance instruction.					X				
33. Op codes 75040 through 75777 trap to 10 as reserved Op codes.	X	X	X	X		X	X	X	X

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
Only if KEV-11 option is present, Op codes 75040 through 75377 can be used as escapes to user microcode. Op codes 75400 through 75777 can also be used as escapes to user microcode and KEV-11 option need not be present. If no user microcode exists, a trap to 10 occurs.					X				
34. Op codes 170000 through 177777 trap to 10 as reserved instructions.	X		X			X	X	X	
Op codes 170000 through 177777 are implemented as floating point instructions.		X		X					X
Op codes 170000 through 177777 can be used as escapes to user microcode. If no user mocrocode exists, a trap to 10 occurs.					X				
35. CLR and SXT do just a DATO sequence for the last bus cycle.		X							
CLR and SXT do DATIP-DATO sequence for the last bus cycle.			X	X	X	X	X	X	X
CLR and SXT do a READ (DATI) and a WRITE (DATO) sequence for the last bus cycle.	X								
36. MEM. MGT. maintenance mode SR0 bit 8 is implemented.				X				X	X
MEM. MGT. maintenance mode SR0 bit 8 is not implemented.		X							

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
37. PS<15:12>, user mode, user stack pointer, and MTPX and MFPX instructions exist even when MEM. MGT. is not configured.		X							X
PS<15:12>, user mode, user stack pointer, and MTPX and MFPX instructions exist only when MEM. MGT. is configured.								X	
38. Current mode PS bit <15:14> set to 01 or 10 will cause a MEM. MGT. trap upon any memory reference.				X				X	X
Current mode PS bits <15:14> set to 01 or 10 will be treated as kernel mode (00) and not cause a MEM. MGT. trap.		X							
39. MTPS in user mode will cause a MEM. MGT. trap if PS addres 177776 is not mapped. If mapped PS<7:5> and <3:0> are affected.		X							
MTPS in user mode will only affect PS<3:0> regardless of whether PS address 177776 is mapped.				X					
40. MFPS in user mode will cause MEM. MGT. trap if PS address 177776 not mapped. If mapped, PS<7:0> are addressed.				X					
MRPS in user mode will access PS<7:0> regardless of whether PS address 177776 is mapped.		X							

Table B-1 Software Differences (Cont)

Activity	PDP-11 Family Machines								
	T11	11 23	04	34	LSI 11	05 10	15 20	35 40	45
41. A HALT instruction in user mode traps to 4									X
A HALT instruction in user mode traps to 10.		X		X				X	
42. A HALT instruction pushes PS & PSW to stack, load the PS with 340, and load the PC with power up address + 4 (restart address).	X								
43. Resident ODT microcode.		X			X				
44. All data outs (DATO) are preceeded by a data in (DATI).	X								
45. Instruction execution runs to completion regardless of bus errors.	X								
46. Vector address range limited to 4 to 374.	X								

HARDWARE DIFFERENCES - TRAPS
(TRANSPARENT TO SOFTWARE)

T11	11/23	11/04	11/34
Priority of internal processor traps, external interrupts, HALT and WAIT:	Priority of internal processor traps, external interrupts, HALT and WAIT:	Priority of internal processor traps, external interrupts, HALT and WAIT:	Priority of internal processor traps, external interrupts, HALT and WAIT:
TRAP Instructions	Memory Parity Errors	Bus Error Trap	Memory Parity Errors
HALT INTERRUPT	Memory Management Fault	TRAP Intructions	Memory Management Fault
TRACE Trap	Bus Error Traps	TRACE Trap	Bus Error Traps
External Vector Interrupt	TRAP Instructions	OVFL Trap	TRAP Instructions
Internal vector Interrupt	TRACE Trap	Power Fail Trap	TRACE Trap
Power Fail Trap	OVFL Trap	UNIBUS Bus Request	OVFL Trap
WAIT Loop	Power Fail Trap	Console HALT	Power Fail Trap
TEST Mode Request	Console Bus Request	WAIT Loop	Console Bus Request
	Q-BUS Bus Request		UNIBUS Bus Request
	WAIT Loop		WAIT Loop

HARDWARE DIFFERENCES - TRAP
(TRANSPARENT TO SOFTWARE)

LSI11	PDP11/05,10	PDP11/15,20	PDP11/35,40
Priority of internal processor traps, external interrupts, HALT and WAIT:	Priority of internal processor traps, external interrupts, HALT and WAIT:	Priority of internal processor traps, external interrupts, HALT and WAIT:	Priority of internal processor traps, external interrupts, HALT and WAIT:
Bus Error Trap	Bus Error Trap	Bus Error Trap	Memory Parity Errors

EK-ORA80-SV-001

RA80 Disk Drive Service Manual

digital

EK-ORA80-SV-001

RA80 Disk Drive Service Manual

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, March, 1982

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

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

- **Class A Computing Devices:**

Notice: This equipment generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference in which case the user at his own expense may be required to take measures to correct the interference.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DEC	DECnet	OMNIBUS
DECUS	DECsystem-10	OS/8
DIGITAL	DECSYSTEM-20	PDT
Digital Logo	DECwriter	RSTS
PDP	DIBOL	RSX
UNIBUS	EduSystem	VMS
VAX	IAS	VT
	MASSBUS	

5/82-15

CONTENTS

Page

FIGURES

Page

FIGURES (Cont)

Page

TABLES

CHAPTER 1
INTRODUCTION

1.1 SCOPE OF MANUAL

This service manual provides the information needed to implement the RA80 Disk Drive corrective maintenance procedures. Chapter 1 provides an overview for RA80 maintenance and reference documentation related to the RA80 Disk Drive. Chapter 2 describes the removal and replacement procedure for each RA80 field replaceable unit (FRU). Chapter 3 describes how to perform the field adjustments. Chapter 4 explains how to use the operator control panel and the internal maintenance controls. Chapter 4 also provides a list of the fault codes displayed on internal and external indicators. Chapter 5 contains the fault isolation procedures that Field Service engineers use to troubleshoot the disk drive. Appendix A provides a binary/hexadecimal conversion chart. Appendix B provides a decimal/hexadecimal conversion chart.

1.2 RA80 MAINTENANCE FEATURES

The RA80 Disk Drive is designed with emphasis placed on serviceability. The drive incorporates the following maintenance features.

- Quick access to all field replaceable parts
- Fault reporting by the operator control panel
- Built-in maintenance controls and indicators
- Drive-resident diagnostic and utility routines
- Read/write diagnostic tracks inside the disk guard band
- No head alignments
- No preventive maintenance procedures
- No special tools

1.3 RA80 MAINTENANCE PHILOSOPHY

The repair philosophy for the RA80 is, "intelligent module replacement." This philosophy is accomplished by use of the drive-resident diagnostics. The Field Service engineer uses these diagnostics to isolate fault conditions to the FRU level. In addition to the drive-resident diagnostics, host-level diagnostics are available to support and verify corrective maintenance decisions.

1.4 RA80 RELATED DOCUMENTATION

The RA80 Disk Drive related documentation is listed below.

The following documentation is available from Printing and Circulation Services, 444 Whitney St., Northboro, Massachussetts 01532.

- RA80 Disk Drive User Guide — EK-ORA80-UG

- RA80 Series Disk Drive Illustrated Parts Breakdown — EK-ORA80-IP

The following documentation is available from the Software Distribution Center, Order Administration/Processing, 20 Forbes Rd., Northboro, Massachusetts 01532.

- RA80 Field Maintenance Print Set — MP-01286

- UDA50 Maintenance Documentation Kit — QP904-GZ
  (This kit consists of a looseleaf binder, the UDA50 Maintenance Guide, and the current drive maintenance guides that operate on the UDA50.)

- RA80 Maintenance Guide — AA-M186A-TC
  (This is a small, looseleaf, plastic-wrapped package.)

- 7-1/2 X 5 inch binder — AV-L980A-TK
  (This small binder fits the maintenance guides.)

CHAPTER 2
REMOVAL AND REPLACEMENT PROCEDURES

2.1 INTRODUCTION
This chapter describes the RA80 parts removal and replacement procedures. The chapter begins with the power precautions that should be observed before replacing FRUs. The remainder of the chapter supplies detailed procedures for FRU replacement.

2.2 POWER PRECAUTIONS
Since hazardous voltages are present inside this equipment, servicing should be performed only by qualified service representatives. Bodily injury or equipment damage may result from improper servicing.

NOTE

Always remove power from the unit before replacing any internal part or cables.

2.3 POWER SUPPLY LOCATION AND CONTROLS
The power controls for the RA80 Disk Drive and power controller (H874) are shown in Figure 2-1.

2.4 REMOVING POWER FROM THE DISK DRIVE
Before replacing assemblies in the RA80 Disk Drive, the disk should be spun down and the ac line power removed. Use the instructions in the following two paragraphs.

2.4.1 Removing Power from the Drive Internal Assemblies
To remove power to everything but the power supply, switch off CB1 at the rear of the RA80 Disk Drive.

2.4.2 Removing Power from the H766C or D Power Supply
To remove power to the H766C or D power supply, unplug the ac cord from the receptacle on the power control unit at the bottom of the RA80 cabinet.

2.5 REMOVAL AND REPLACEMENT SEQUENCE
Figure 2-2 provides an RA80 part removal sequence.

To remove a part, locate it on the sequential flow diagram and follow the path to the top of the diagram. Begin by removing the topmost item on the path that the flow-line passes through. Continue down the flow-line until the desired part is reached. Paragraph numbers assist in the location of each removal procedure. Parts that can be directly removed are not shown on the diagram.

NOTE

Unless otherwise indicated, reverse the removal procedure to replace each FRU.

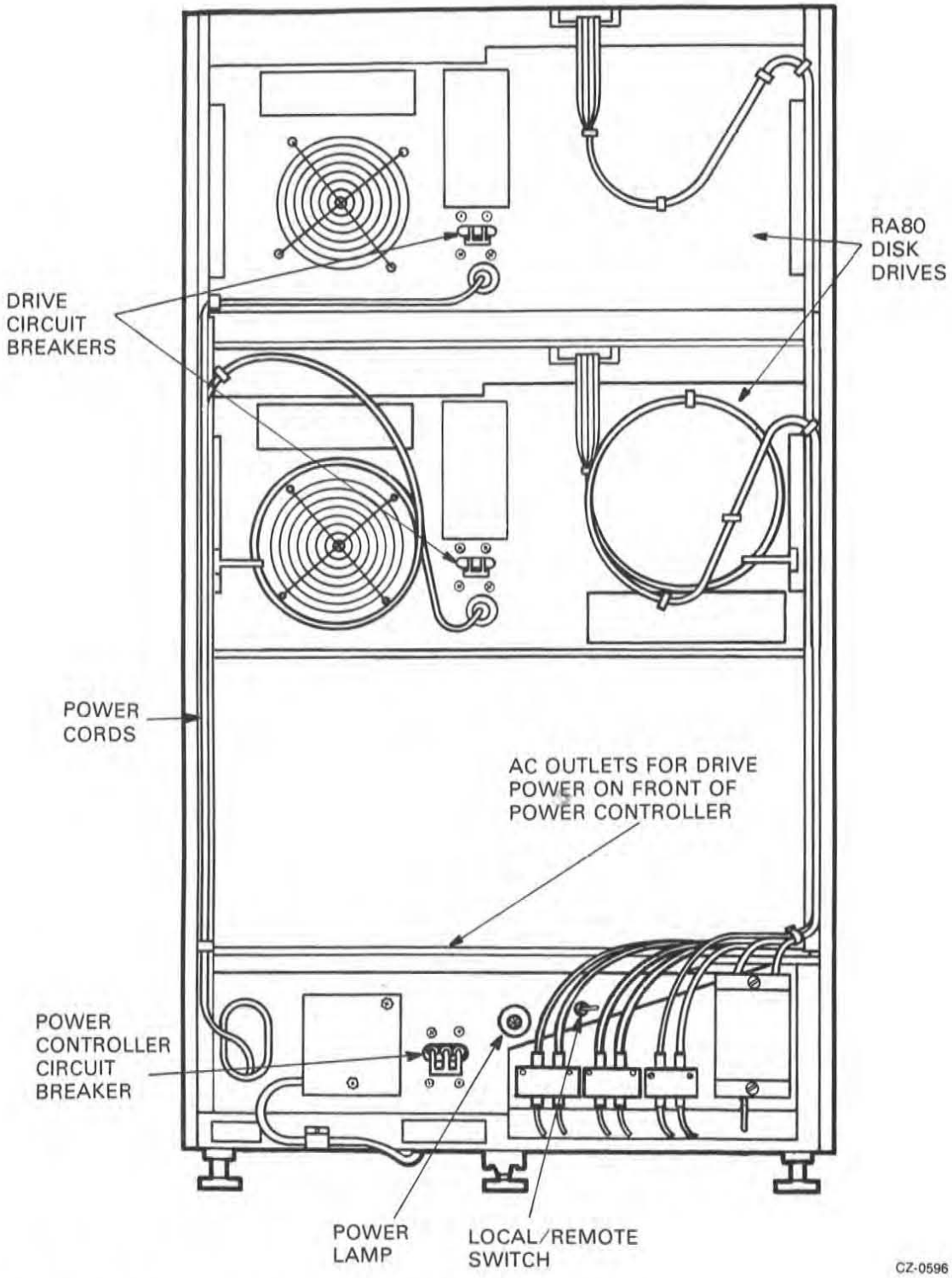


Figure 2-1 Location of Power Controls

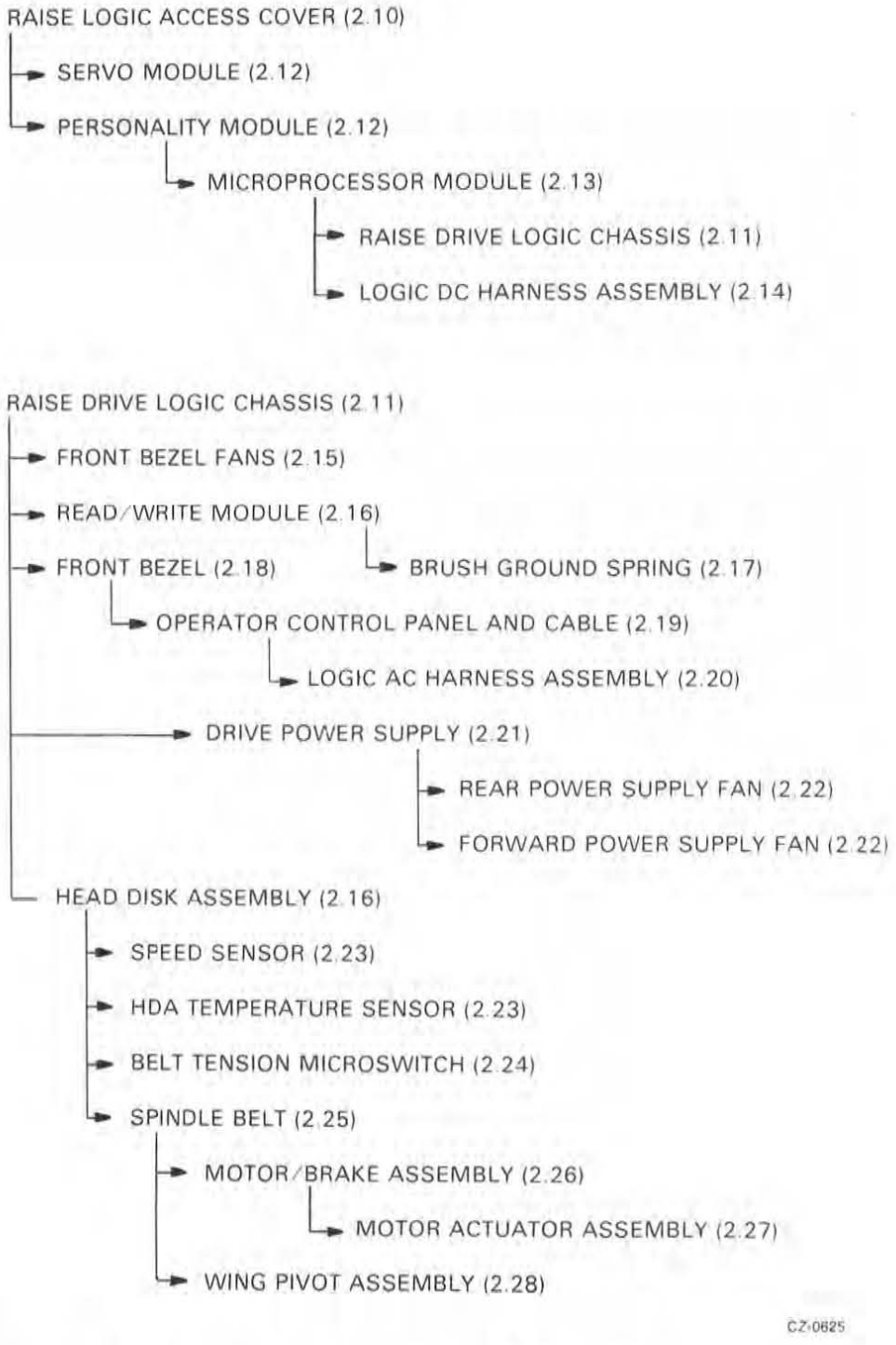


Figure 2-2 Sequential Part Removal and Replacement

2.6 REMOVING THE REAR CABINET DOOR AND END PANELS

Refer to Figures 2-3 and 2-4 while performing this procedure.

1. Unlock the rear door with a 5/32" hex wrench.
2. Disconnect the green/yellow striped ground wire using a Phillips screwdriver.
3. Unlatch the rear door and lift it off of the frame assembly.
4. The end panels are removed by first removing the two end panel locks. The end panel locks are removed by loosening the screws and then lifting the locks off the panels.
5. Lift the end panels up and away from the cabinet.
6. Unscrew the green/yellow ground wire from the cabinet and set the end panel aside.

2.7 REMOVING THE SWITCH CAPS AND LAMPS

Refer to Figure 2-5 while performing this procedure.

1. Remove the operator control panel switch caps by prying the recessed side of the cap with a screwdriver.

NOTE

A piece of paper or some other material should be placed between the screwdriver and the switch cap to avoid chipping the paint on the front bezel.

2. To remove one of the lamps, reach into the switch opening and pull on the metal slide. The lamp will pull forward with the slide.
3. Insert a lamp with the rear flat portion of the lamp in a horizontal position. Push the lamp into the lamp holder as far as it will go.
4. To replace the switch cap, push the cap into the switch opening as far as it will go, using only a small amount of pressure to snap the cap into place.

2.8 REMOVING THE AIR FILTER

Refer to Figure 2-6 while performing this procedure.

1. Locate the filter door latch on the front bezel.
2. Press up on the door latch and lower the door to a horizontal position.
3. Remove the foam air filter by pulling down on the top half of the filter first and then lifting the filter out of the drive.

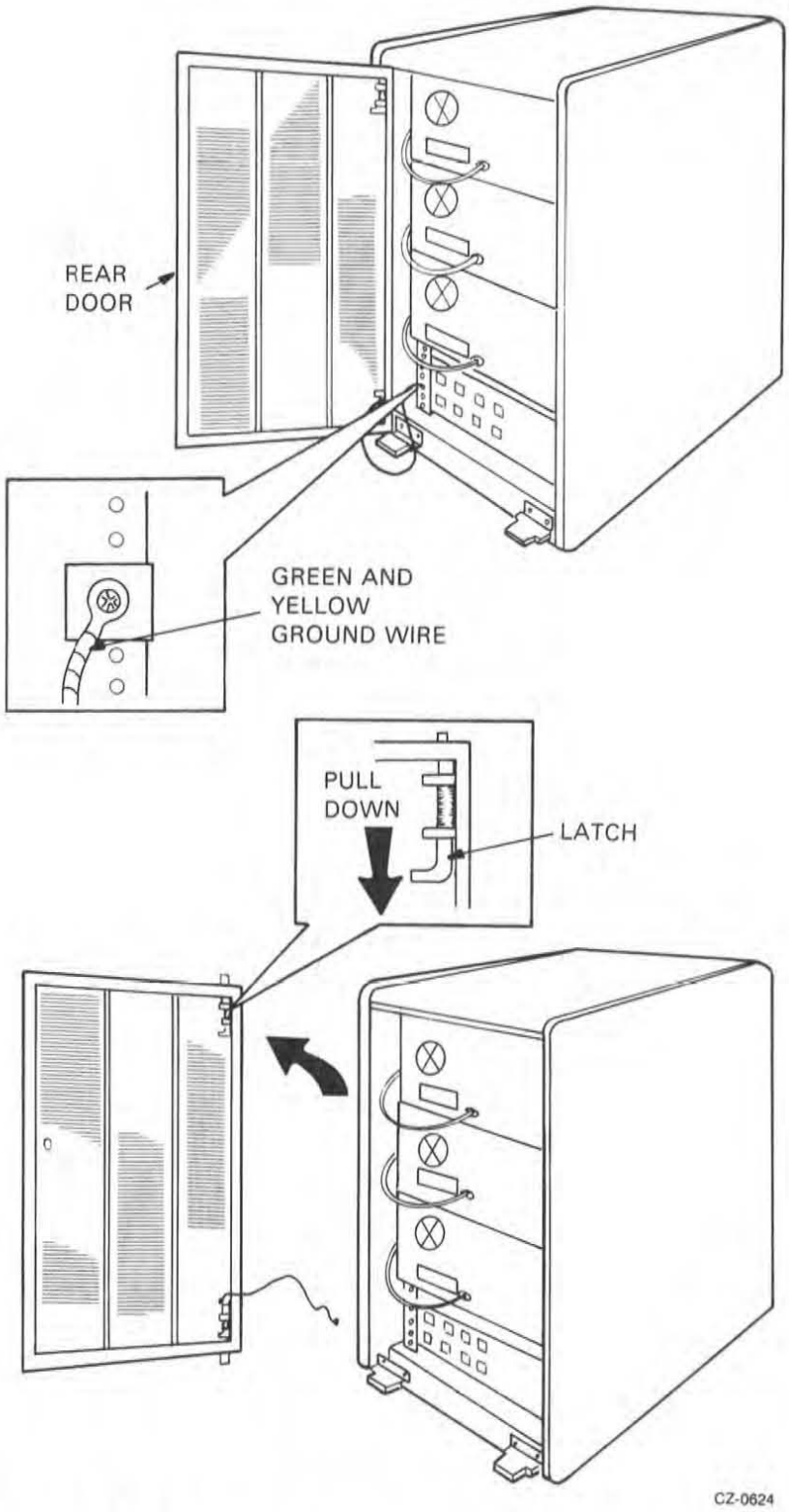


Figure 2-3 Rear Door Removal

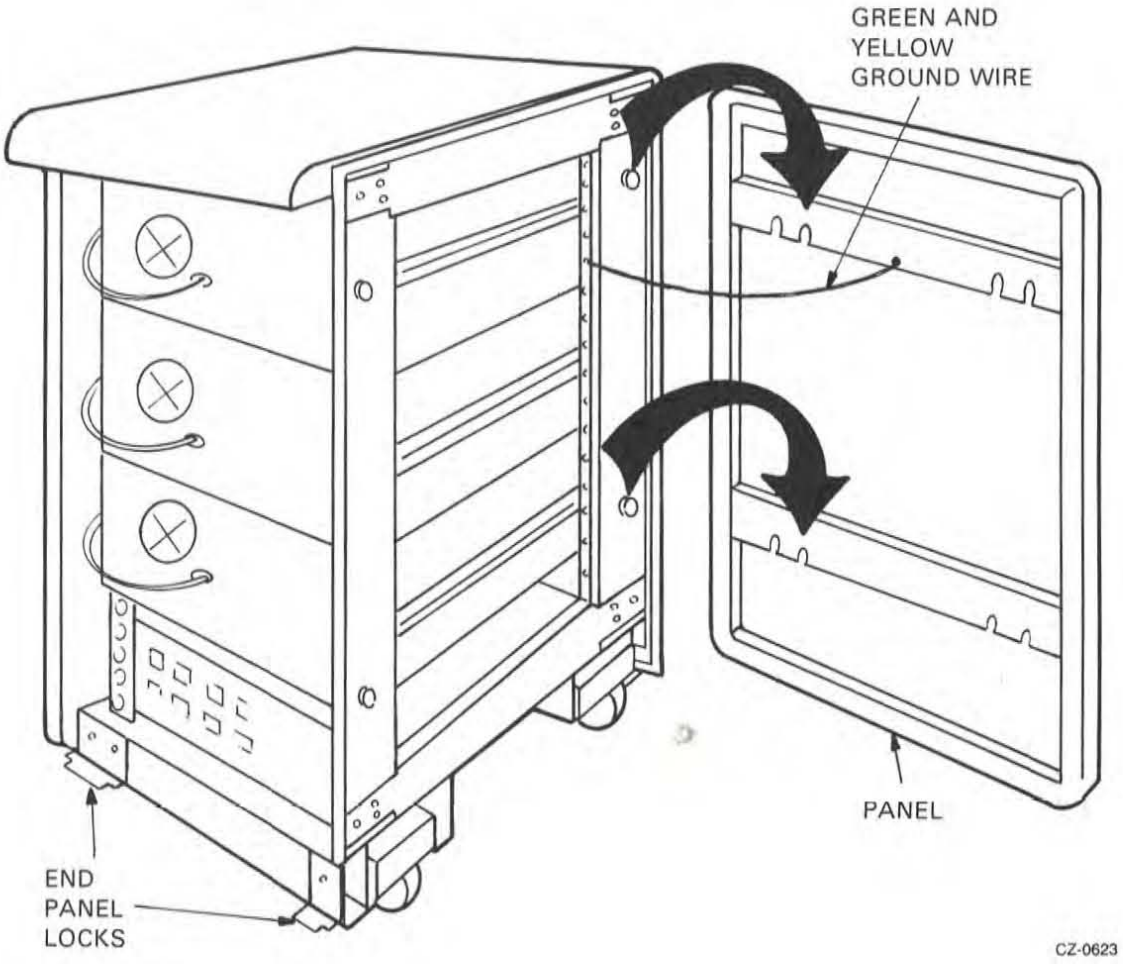


Figure 2-4 End Panel Removal

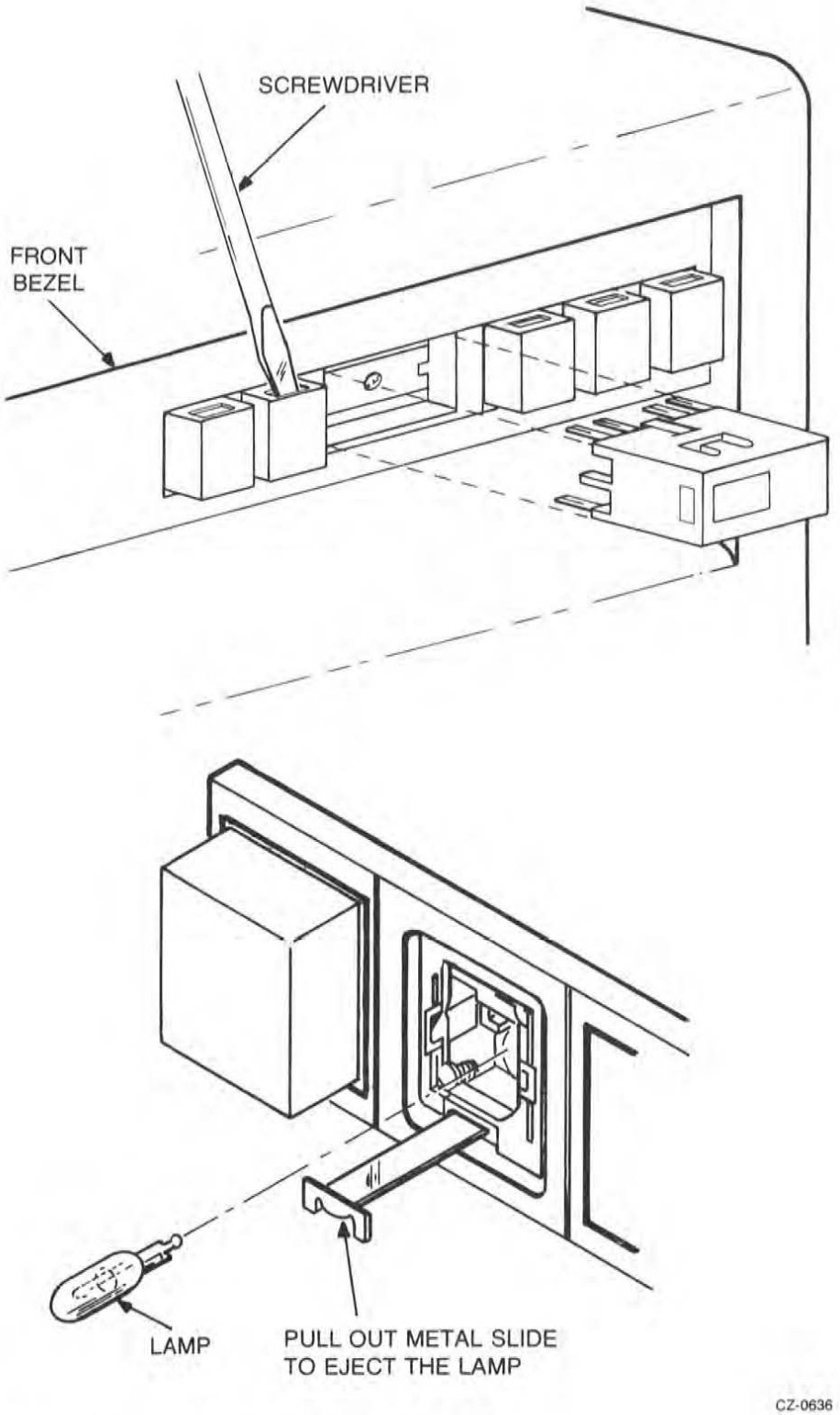


Figure 2-5 Switch Cap and Lamp Removal

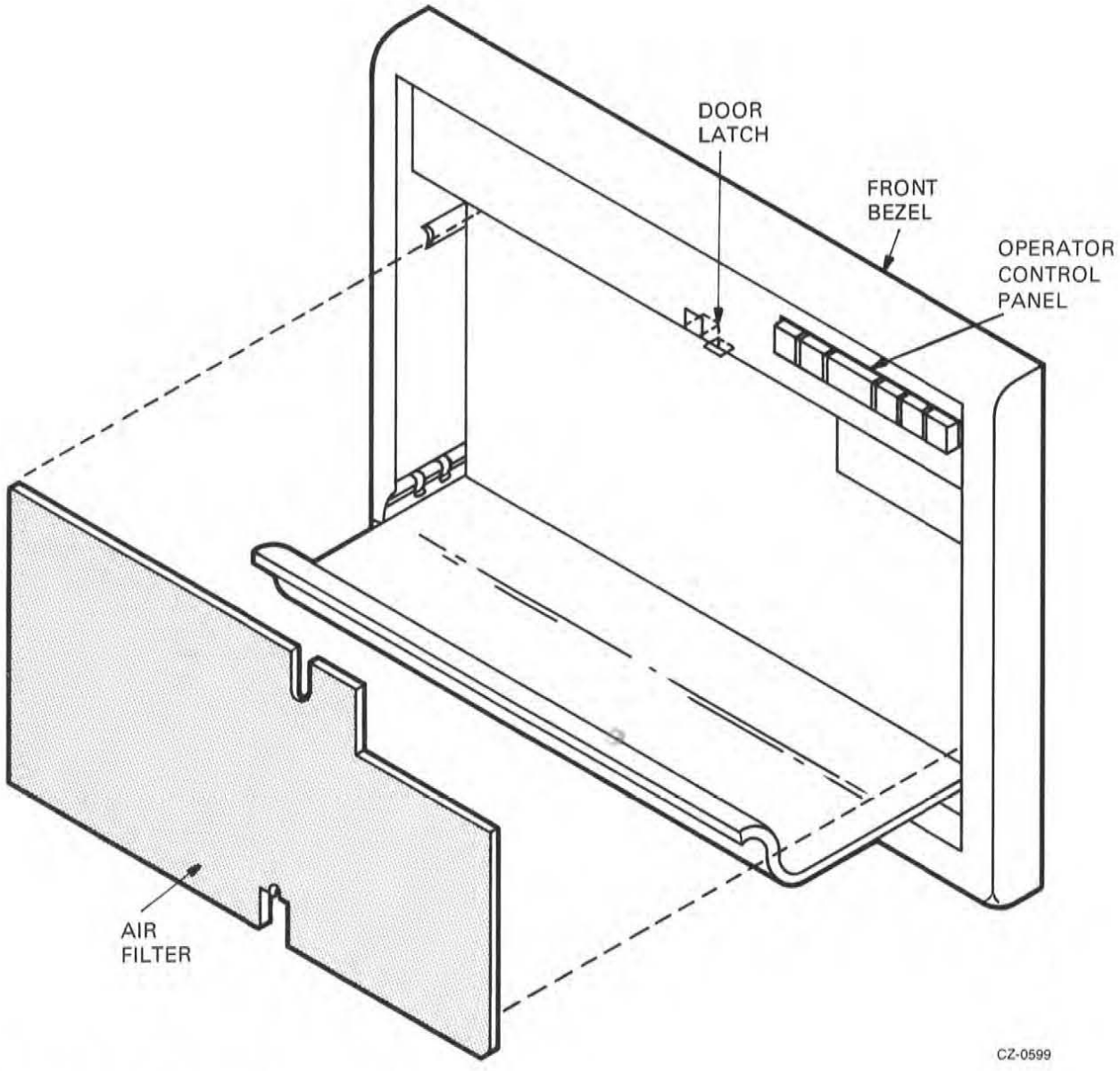


Figure 2-6 Air Filter Removal

2.9 EXTENDING AND RETRACTING THE DRIVE ON SLIDES

Replace internal parts of drives on slides by following the procedures in the next two paragraphs. Refer to Figures 2-7, 2-8, and 2-9 while performing these procedures.

2.9.1 Extending the Drive on Slides

1. Pull out the cabinet stabilizer foot. Refer to Figure 2-7.

CAUTION

Never slide a drive out of the cabinet without first extending the stabilizer foot.

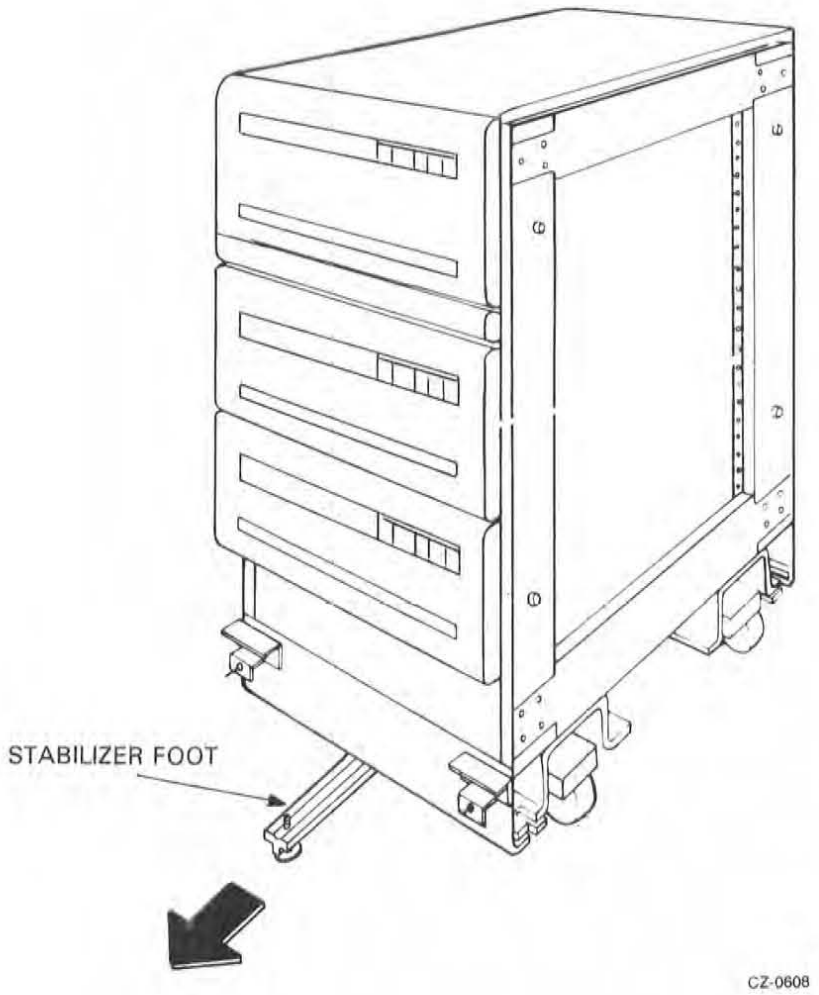


Figure 2-7 Extending the Stabilizer Foot

2. Open the rear door of the cabinet.

3. Remove the power supply screw that holds the rear of the disk drive to the electrostatic discharge bracket (ESD). Refer to Figure 2-8.

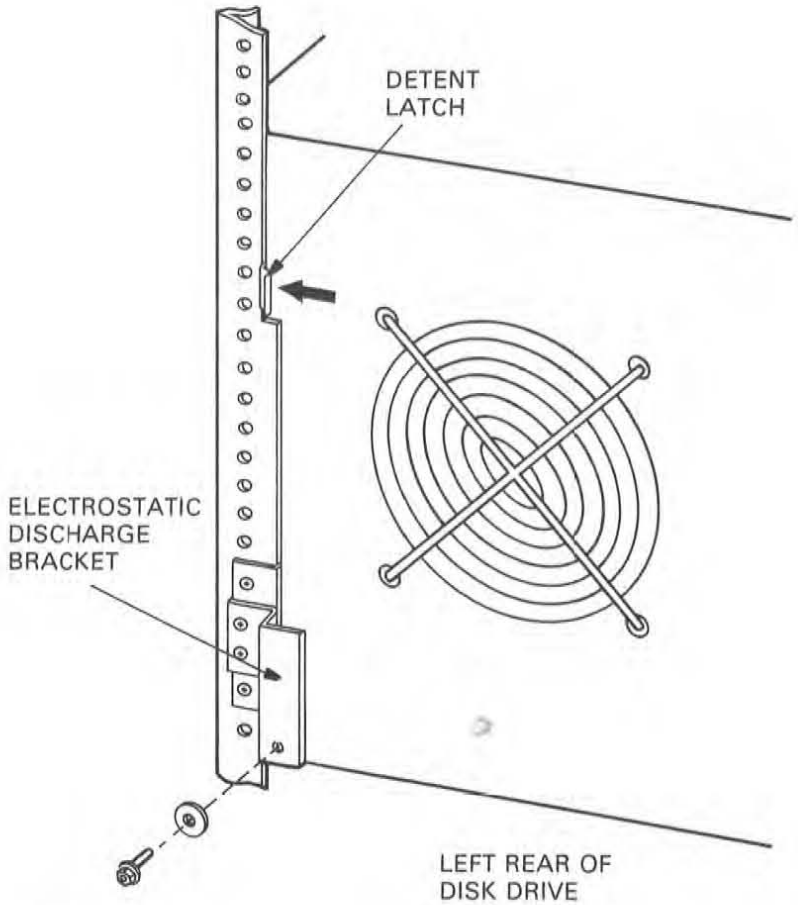


Figure 2-8 Preparing for Drive Extension

4. Push the detent latch to the left and push the drive forwards.

5. Go to the front of the drive and pull the drive out on its slides until it locks in place. Refer to Figure 2-9.

6. Push up on slide lock arm A to extend the drive all the way out on its slides.

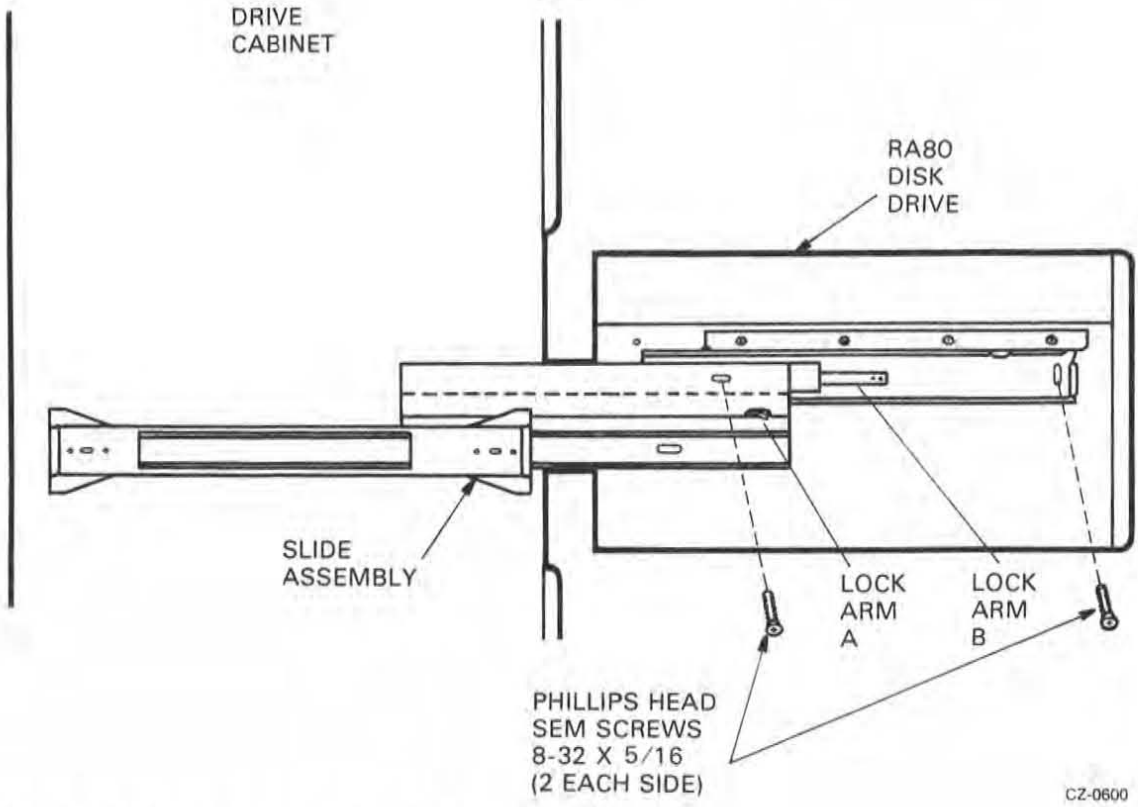


Figure 2-9 Extending the Chassis Slides

2.9.2 Sliding the Drive Back into the Cabinet

1. Push in on slide lock arm B and slide the drive into the cabinet.

2. Check that the detent latch at the rear of the cabinet has locked the drive in place.

3. Install the power supply screw through the electrostatic discharge bracket.

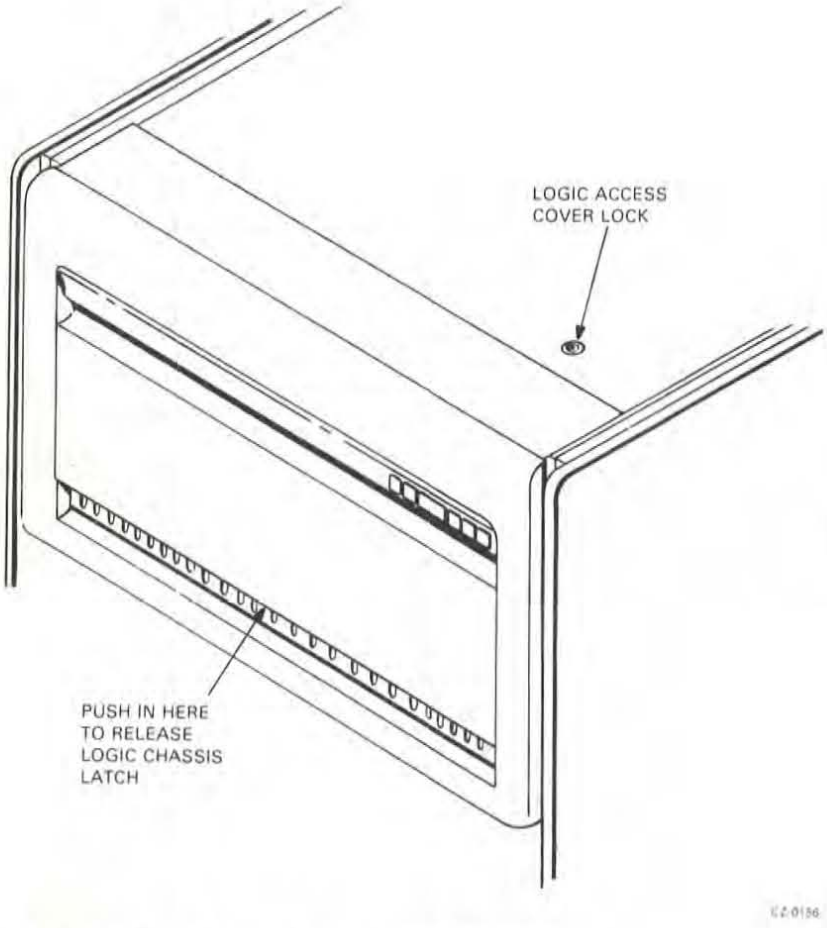
4. Close the rear door of the cabinet.

2.10 RAISING THE LOGIC ACCESS COVER

Refer to Figure 2-10 while performing this procedure.

NOTE

Foldout Figure 2-10 provides an overview of the main access compartments of the RA80 Disk Drive.



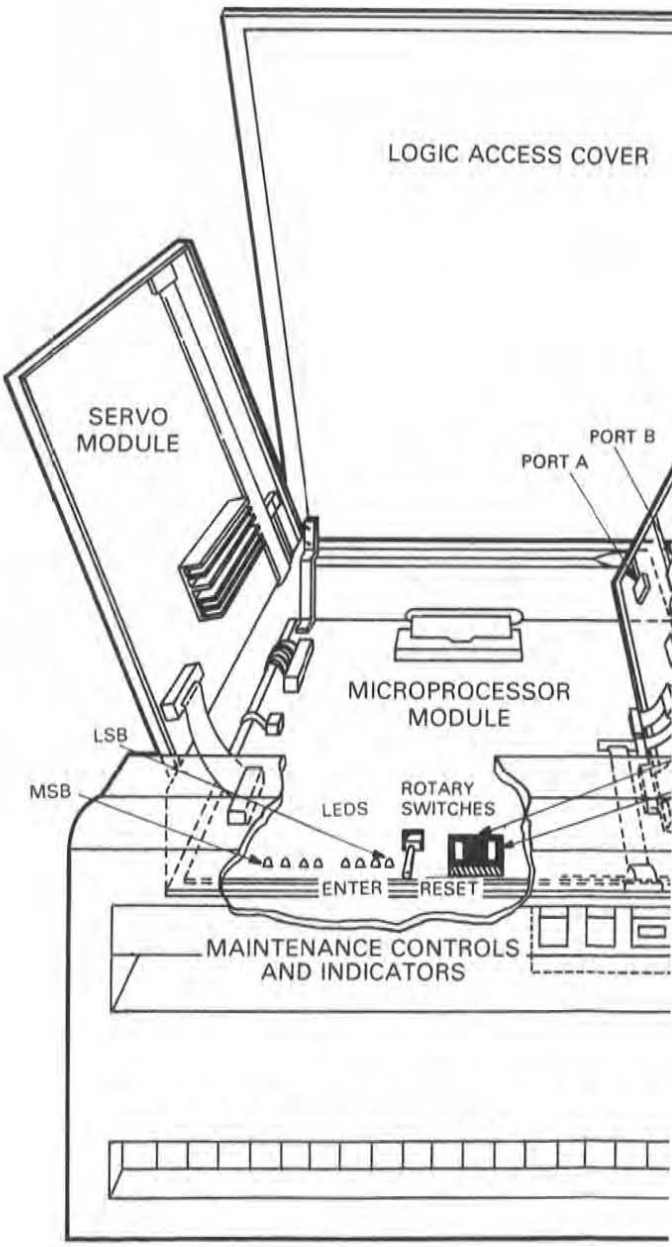
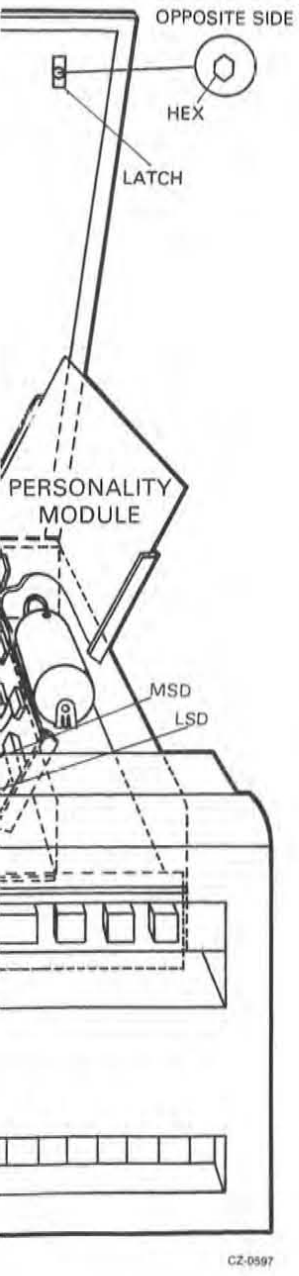
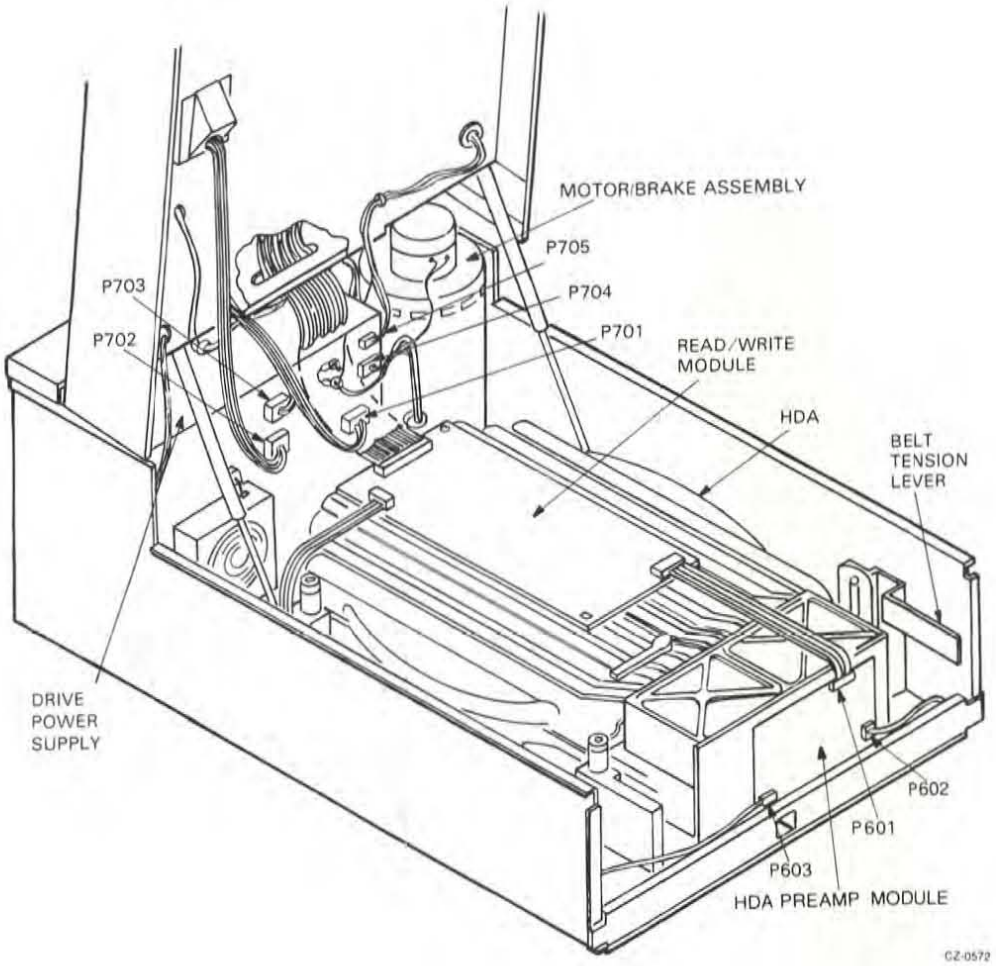


Figure 2-10 Access to the Inside of the Drive


OPPOSITE SIDE
HEX
LATCH
PERSONALITY MODULE
MSD
LSD
CZ-0597


MOTOR/BRAKE ASSEMBLY
P705
P704
P703
P702
P701
READ/WRITE MODULE
HDA
BELT TENSION LEVER
DRIVE POWER SUPPLY
P602
P601
P603
HDA PREAMP MODULE
CZ-0572

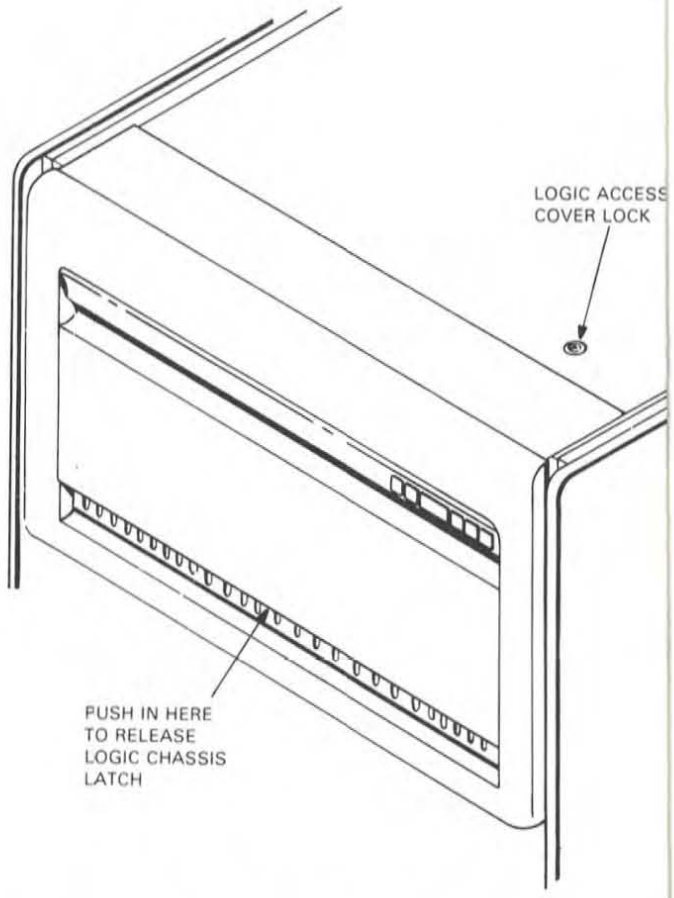


Figure 2-10 Access to the Inside of the Drive

1. Turn the logic access cover lock 90° counterclockwise while placing downward pressure on the cover.

2. The cover will pop up slightly once the cover latching mechanism is released.

3. Raise the logic access cover to expose the drive logic modules.

2.11 RAISING THE DRIVE LOGIC CHASSIS

Refer to Figure 2-10 while performing this procedure.

1. Push in the logic chassis release latch.

2. Lift the drive logic chassis to its fully raised position.

2.12 REMOVING THE SERVO AND PERSONALITY MODULES

Refer to Figure 2-11 while performing this procedure.

1. Raise the logic access cover. (Paragraph 2.10.)

2. Pivot the servo and personality modules up so that the microprocessor module is exposed.

3. ,Unplug all the module cables.

NOTE

Do not cut the cable tie wraps on the SI cables. Instead, remove the nylon nut and unplug the connector with the strain relief attached.

4. Lift the module(s) out of the logic chassis.

2.13 REMOVING THE MICROPROCESSOR MODULE

Refer to Figure 2-12 while performing this procedure.

1. Raise the logic access cover. (Paragraph 2.10.)

2. Remove the personality module. (Paragraph 2.12.)

3. Unplug all cables to the microprocessor module.

4. Pull up on the five push pins.

5. Lift the microprocessor module out of the logic chassis.

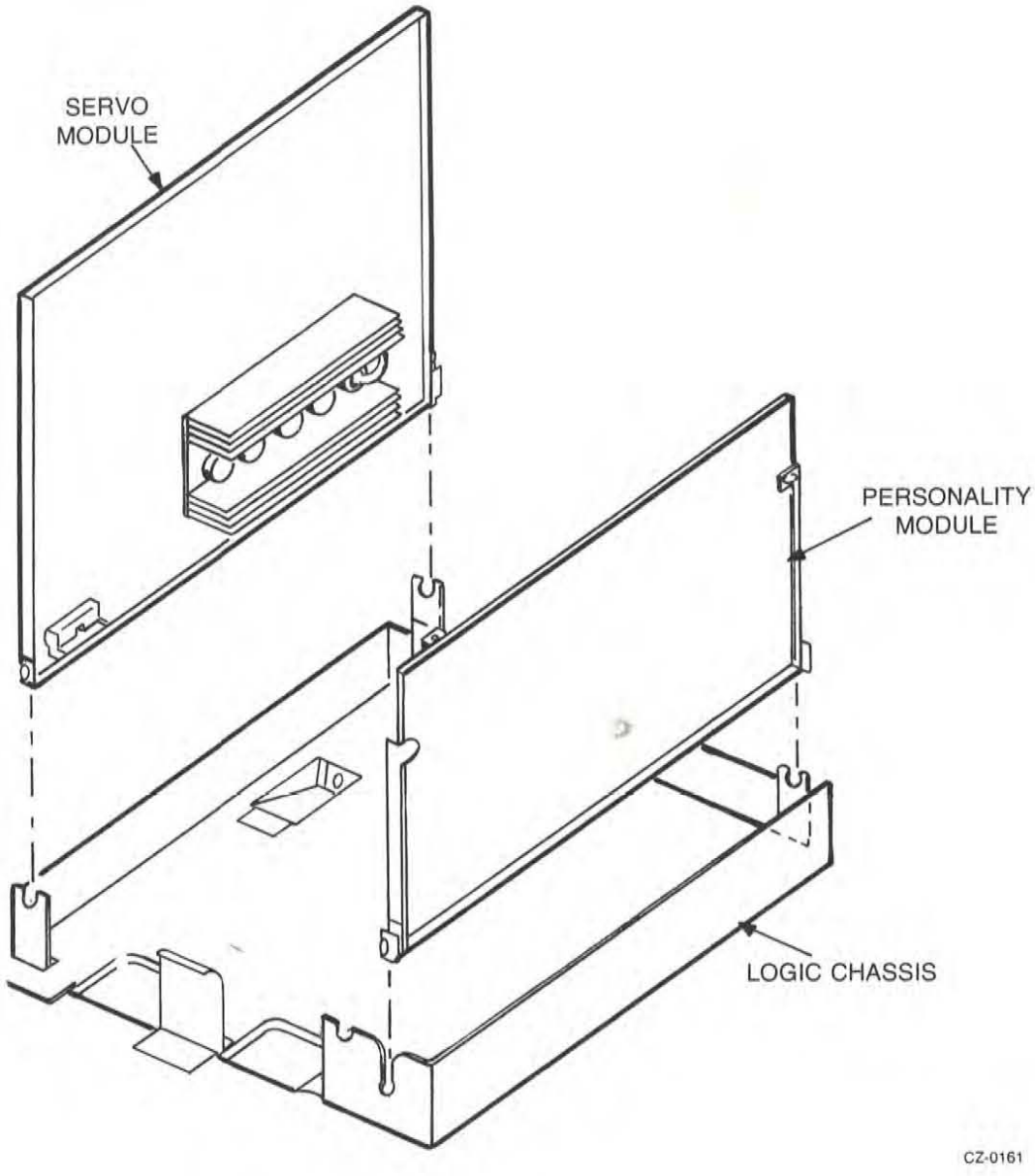


Figure 2-11 Servo and Personality Module Removal

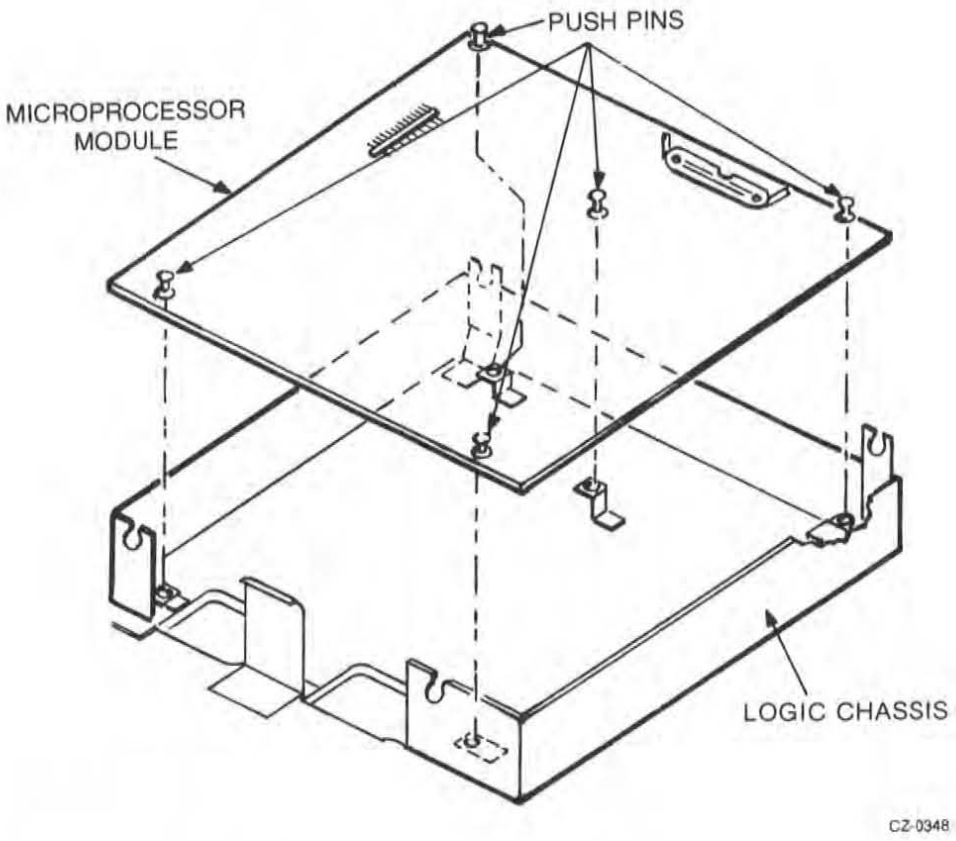


Figure 2-12 Microprocessor Module Removal

2.14 REMOVING THE LOGIC DC HARNESS ASSEMBLY

Refer to Figures 2-13 and 2-14 while performing this procedure.

1. Raise the logic access cover. (Paragraph 2.10.)
2. Remove the servo, personality, and microprocessor modules. (Paragraphs 2.12 and 2.13.)
3. Remove the feedthrough bracket screws.
4. Remove the two screws holding the DC power harness.
5. Raise the drive logic chassis by releasing the chassis latch. (Paragraph 2.10.)
6. Unplug the P701, P702, and P703 connectors.
7. Remove the dc power harness.

2.15 REMOVING THE FRONT BEZEL FANS

Refer to Figure 2-15 while performing this procedure.

1. Raise the drive logic chassis by releasing the chassis latch. (Paragraph 2.11.)
2. Remove the two screws securing the fan that is to be removed.
3. Slide the fan out of the chassis and remove the quick-connect terminals.
4. Remove the four screws and retaining nuts that hold the fan on its bracket.

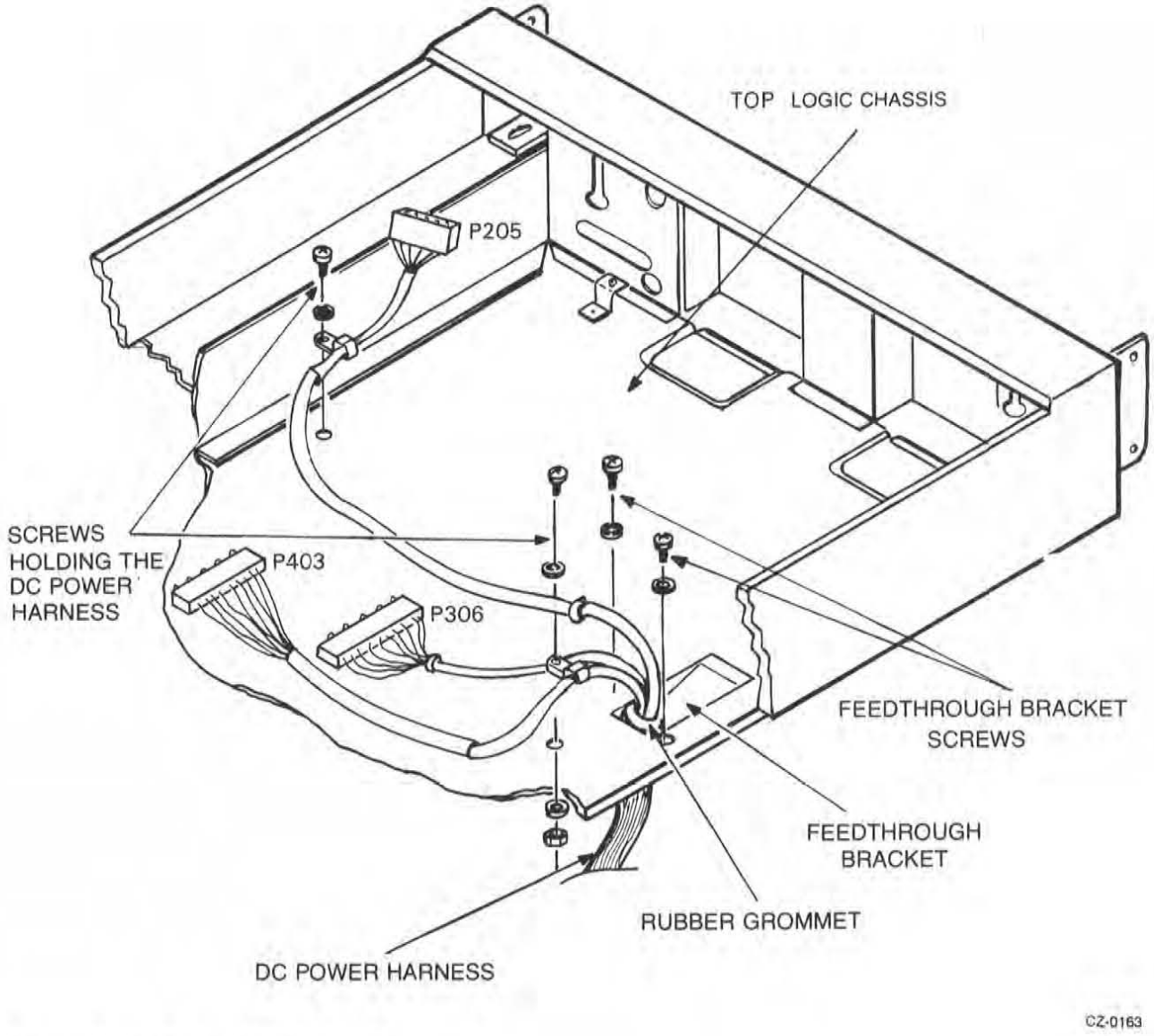


Figure 2-13 Logic DC Harness Removal

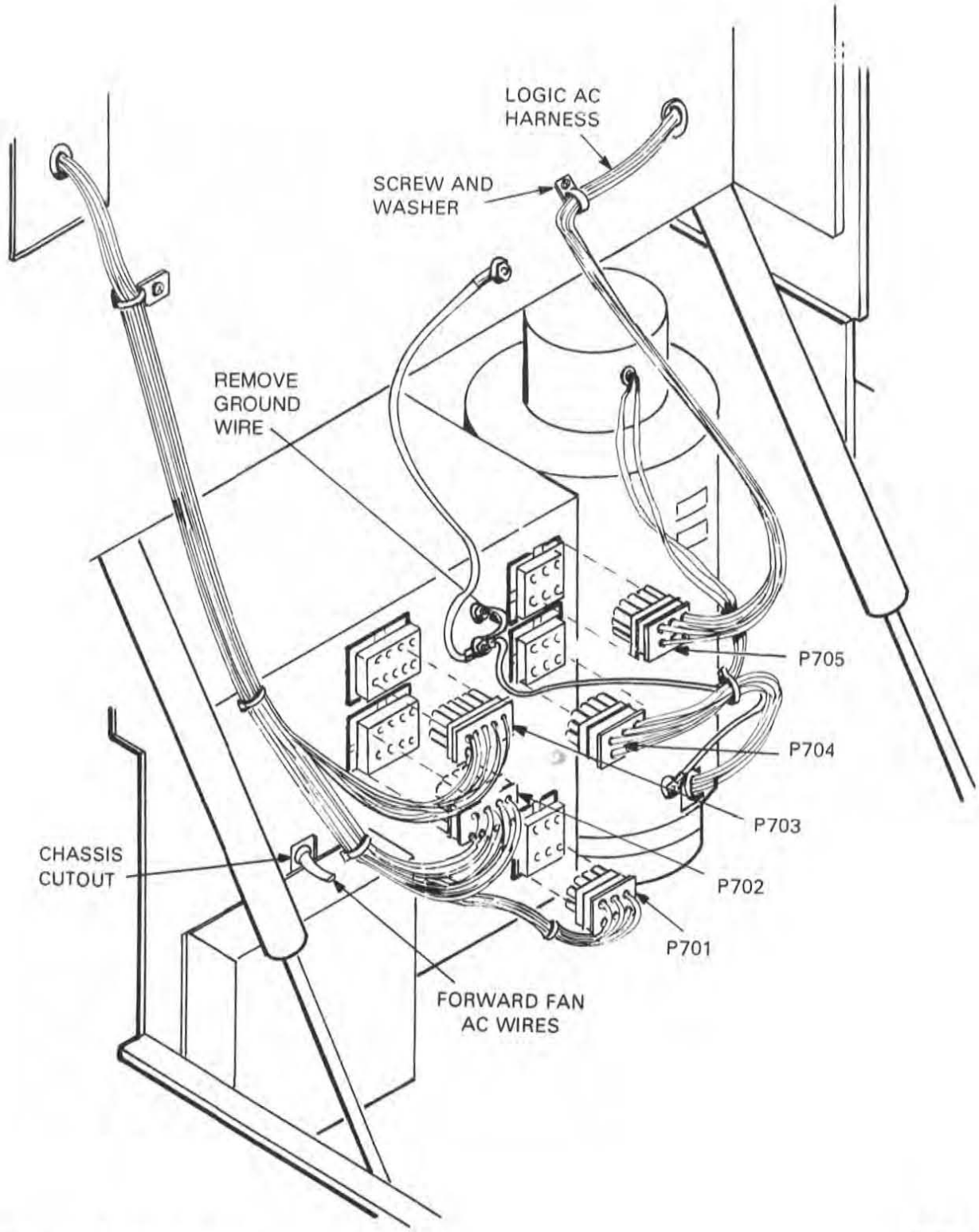


Figure 2-14 Drive Power Supply Connectors

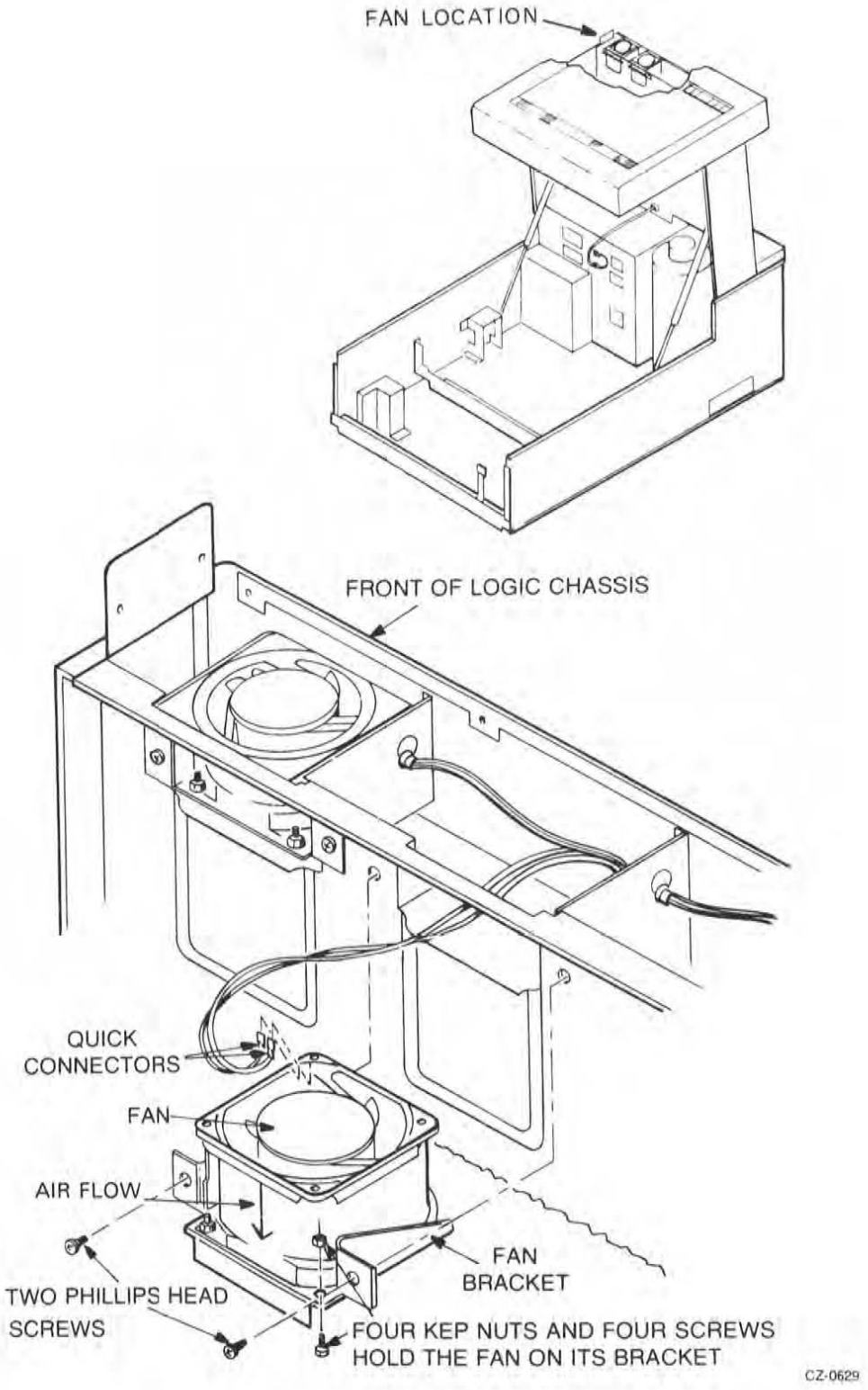


Figure 2-15 Front Bezel Fan Removal

2.16 REMOVING THE HDA AND THE READ/WRITE MODULE

Refer to Figures 2-16 and 2-17 while performing the following removal and replacement procedures.

2.16.1 Removing the HDA and the Read/Write Module

1. Raise the drive logic chassis by releasing the chassis latch. (Paragraph 2.11.)
2. Unplug connector P502 from the read/write module.
3. Unplug connectors P602 and P603 from the HDA preamp module.
4. Place the belt tension lever into the release position.
5. Remove the four HDA retaining nuts.
6. Place the positioner lock into the LOCK position.

CAUTION
If Step 6 is not performed, the HDA may be damaged.

7. Remove the HDA from the drive by placing your hands at diagonally opposite corners and then lifting the unit out of the drive.

CAUTION
Never place the HDA down on its pulley.

8. Place the HDA on a level surface in the vertical position only. Plastic bulkhead feet are provided for this purpose on the front bulkhead of the HDA.
9. Unplug connectors P501 and P503 from the read/write module, if the module is to be removed.
10. Remove the four screws holding the read/write module to the HDA.

2.16.2 Replacing the HDA and the Read/Write Module

1. Check that the drive belt is centered on the motor pulley. The other end of the belt should be even with the top of the nylon rollers on the wing pivot assembly.

CAUTION
If the drive belt is not aligned properly, damage to the HDA, drive belt, or nylon roller may result.

2. Lift the HDA by two diagonally opposite corners and lower it over the four mounting bolts.
3. Tighten the HDA in position with the four nuts and washers.
4. Reconnect P602 and P603 to the preamp module on the front of the HDA.
5. Install the read/write module on top of the HDA if it is removed or if this is a new HDA.

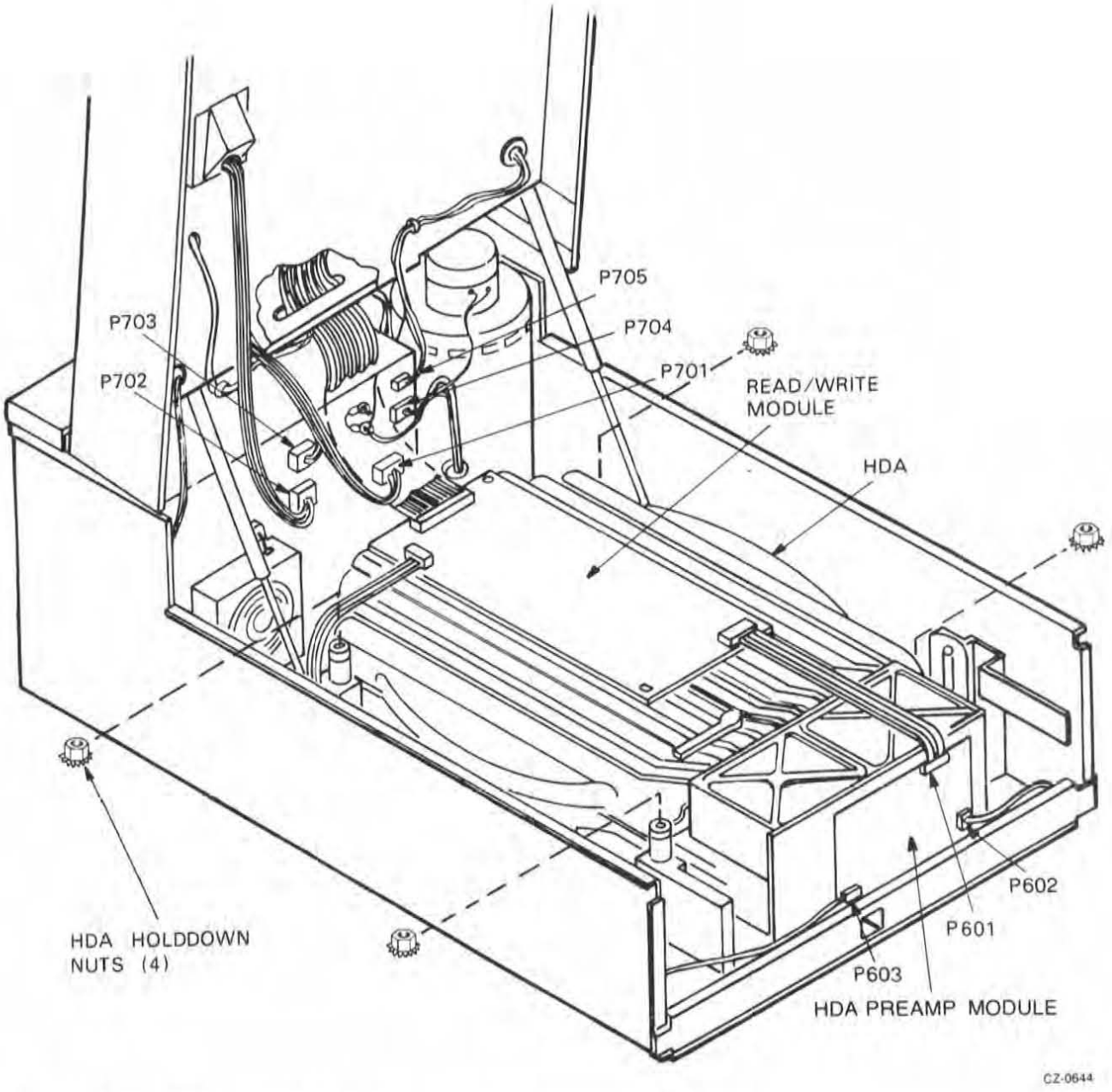


Figure 2-16 HDA and Read/Write Module Removal

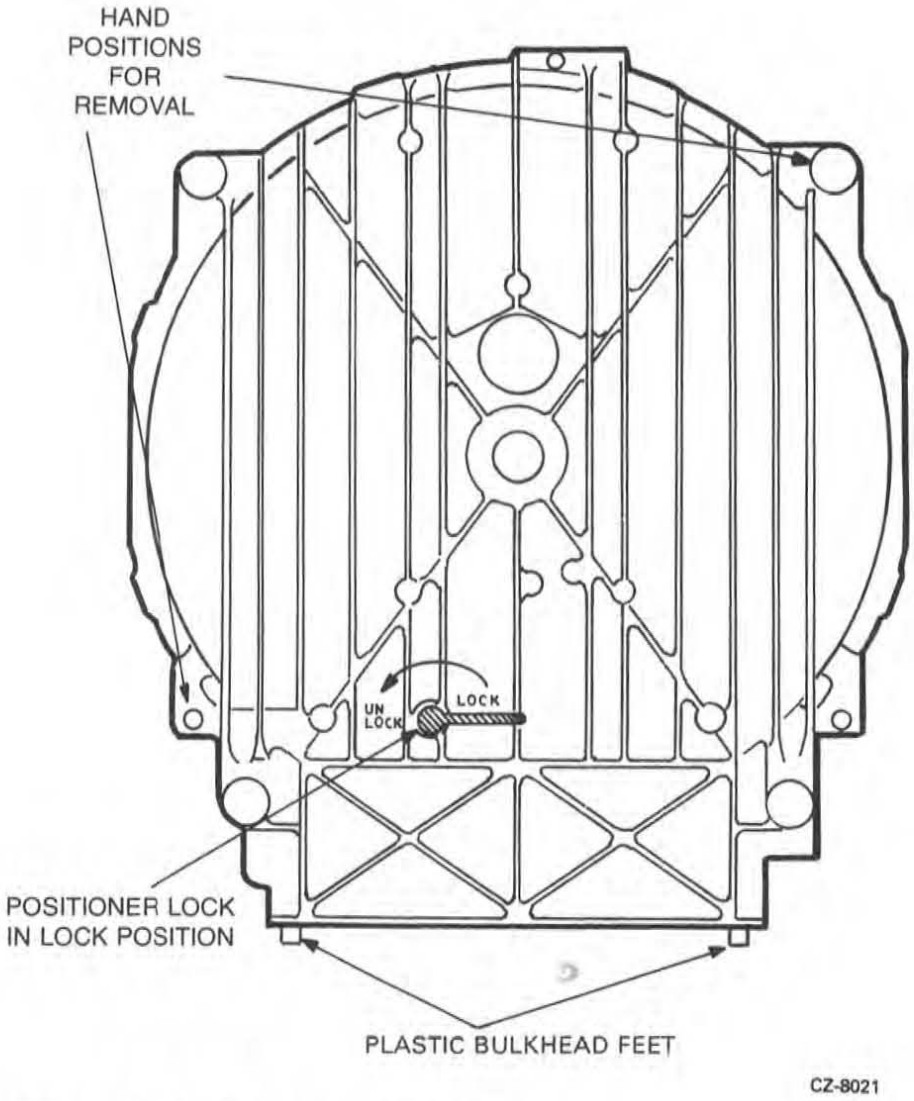


Figure 2-17 HDA Positioner Lock Lever

6. Reconnect P501, P502, and P503 to the read/write module.
7. Reconnect P601 to the HDA preamp module if it was removed.
8. Place the belt tension lever in the full forward (locked) position.
9. Turn the HDA positioner lock counterclockwise into the locked position.
10. Perform the servo adjustment described in Chapter 3 if a new HDA is installed.

CAUTION

The read/write heads rest on the HDA disk surfaces. Therefore, if the HDA is to be shipped or moved to another location, certain precautions must be taken to prevent damage to these heads. The HDA spindle pulley must be taped in place to prevent movement of the spindle.

2.17 REMOVING THE BRUSH GROUND SPRING

Refer to Figures 2-16 and 2-18 while performing this procedure.

1. Raise the drive logic chassis by releasing the chassis latch. (Paragraph 2.11.)
2. Remove the read/write module.
3. Remove the brush ground spring retaining ring using a retaining ring tool.
4. Remove the brush ground spring from the top of spindle.

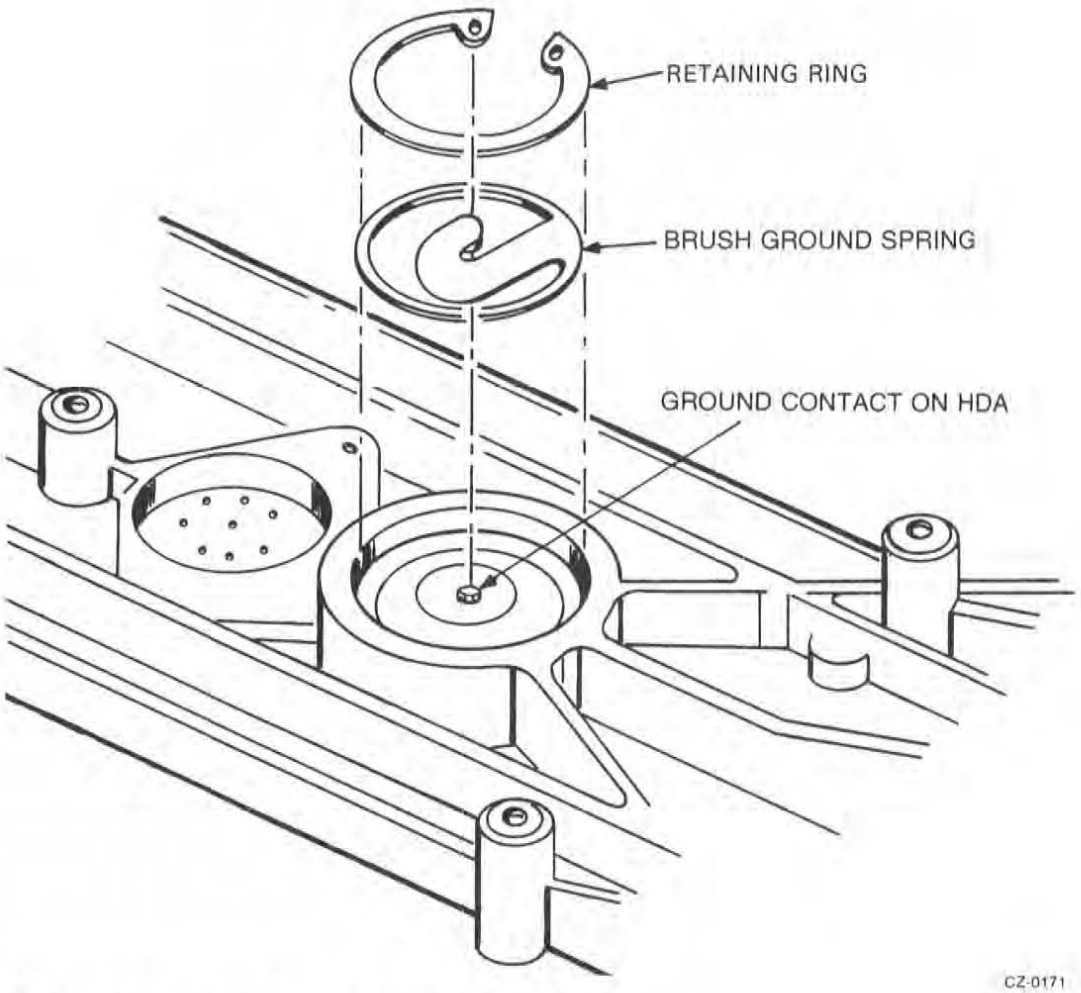


Figure 2-18 Brush Ground Spring Removal

2.18 REMOVING THE FRONT BEZEL

Refer to Figure 2-19 while performing this procedure.

1. Raise the drive logic chassis by releasing the chassis latch. (Paragraph 2.11.)
2. Remove the eight screws that secure the front bezel to the drive logic chassis.

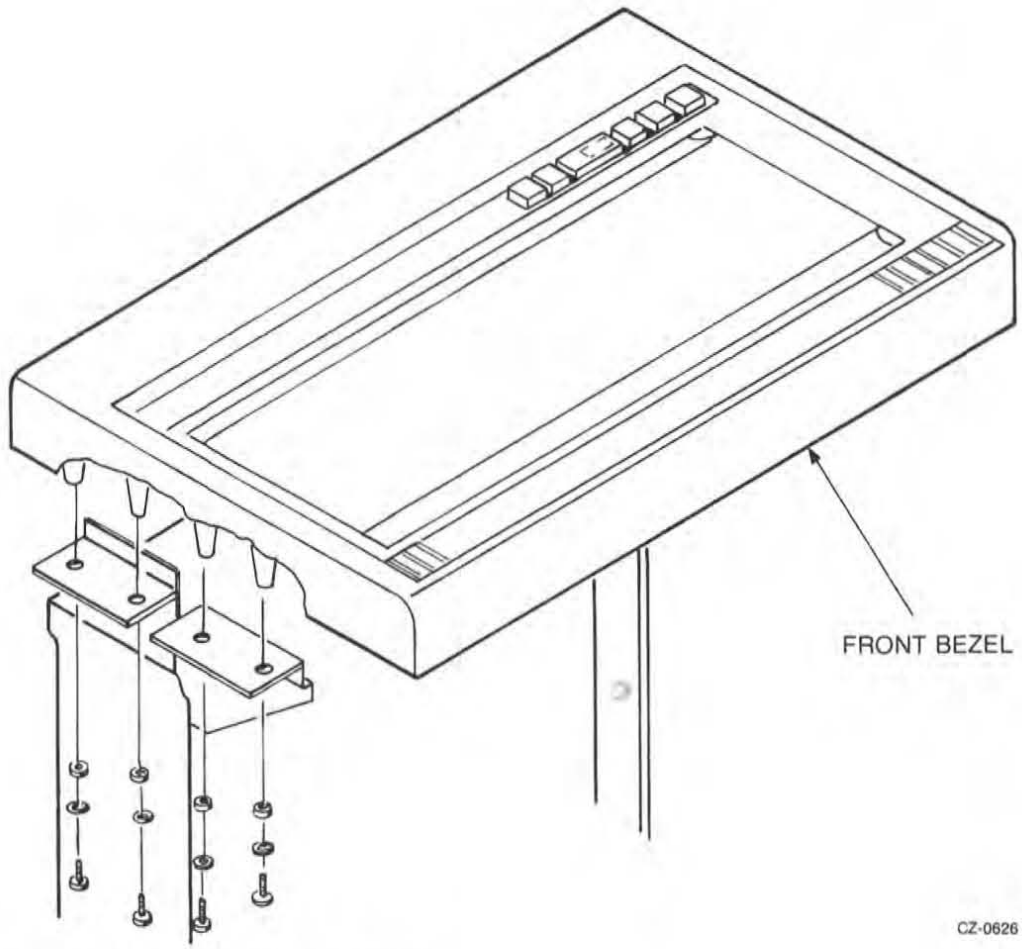


Figure 2-19 Front Bezel Removal

2.19 REMOVING THE OPERATOR CONTROL PANEL AND CABLE

Refer to Figures 2-20 and 2-21 while performing this procedure.

1. Raise the drive logic chassis by releasing the chassis latch. (Paragraph 2.11.)
2. Remove the front bezel. (Paragraph 2.18.)
3. Remove the two screws that secure the operator control panel bracket to the chassis. These screws are located under the logic chassis.
4. Unplug connector P101 from the operator control panel.

NOTE

When installing a new operator control panel, cut the shunts on the module that determine the drive serial number. Also, a DIP switch must be set up to determine the revision level of the drive to the software. (Refer to Figure 2-21.)

2.20 REMOVING THE LOGIC AC HARNESS ASSEMBLY

Refer to Figures 2-22 and 2-23 while performing this procedure.

1. Raise the drive logic chassis by releasing the chassis latch. (Paragraph 2.11.)
2. Remove the front bezel. (Paragraph 2.18.)
3. Remove the operator control panel. (Paragraph 2.19.)
4. Remove the front bezel fans. (Paragraph 2.15.)
5. Remove the screw and washer that hold the harness to the chassis. (Refer to Figure 2-22.)
6. Unplug connector P705 from the power supply to the harness.
7. Raise the logic access cover. (Paragraph 2.10.)
8. Locate the motor start capacitor to the right of the personality module. (Refer to Figure 2-23.)
9. Remove the wires from the motor start capacitor.
10. Cut the ac harness cable clamps.
11. Remove the cable clamp screws.
12. Pull the two grommets from their retaining holes.
13. Remove the ac harness from the chassis front.
14. Pull the ac harness down through the hole in the rear of the chassis.

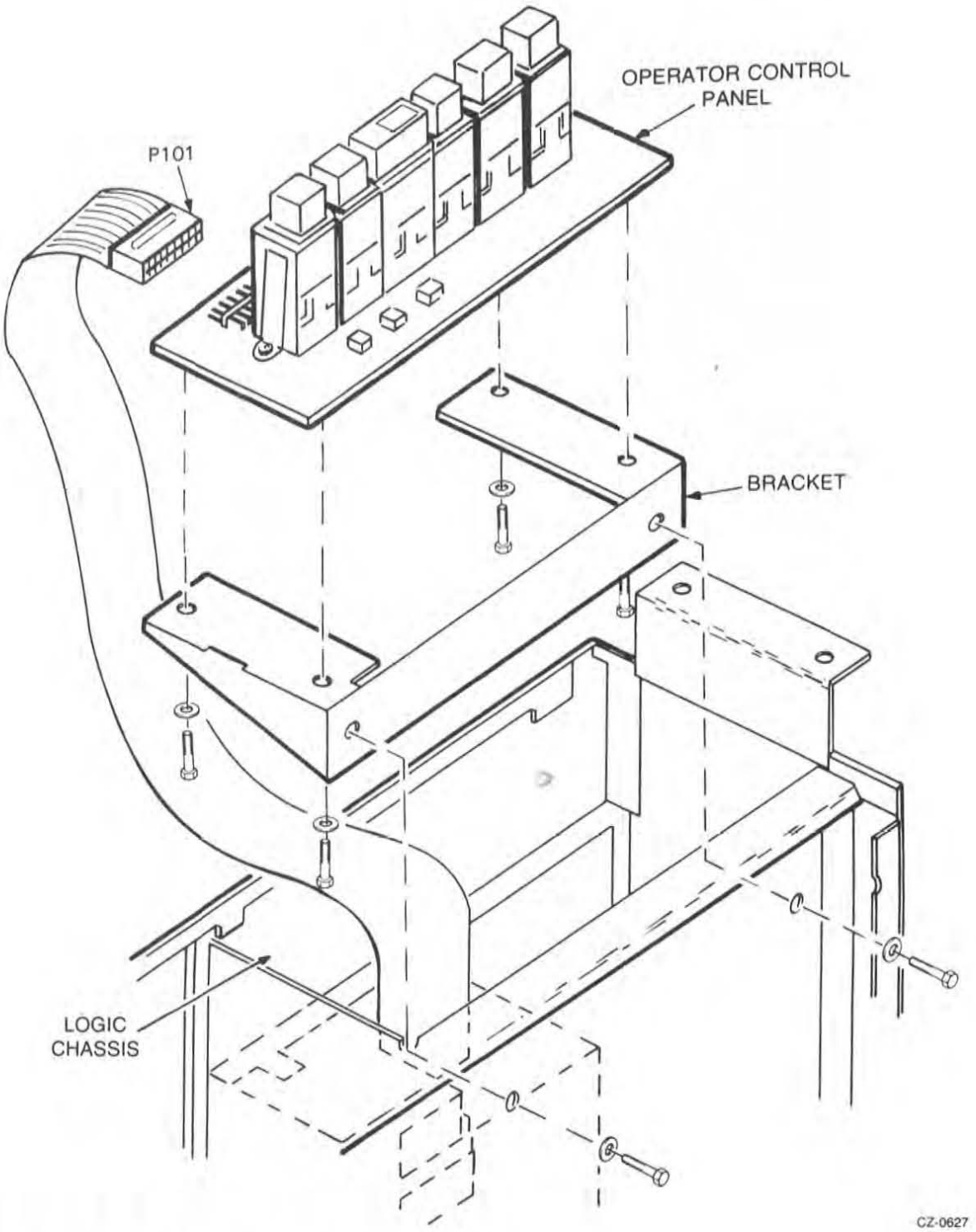
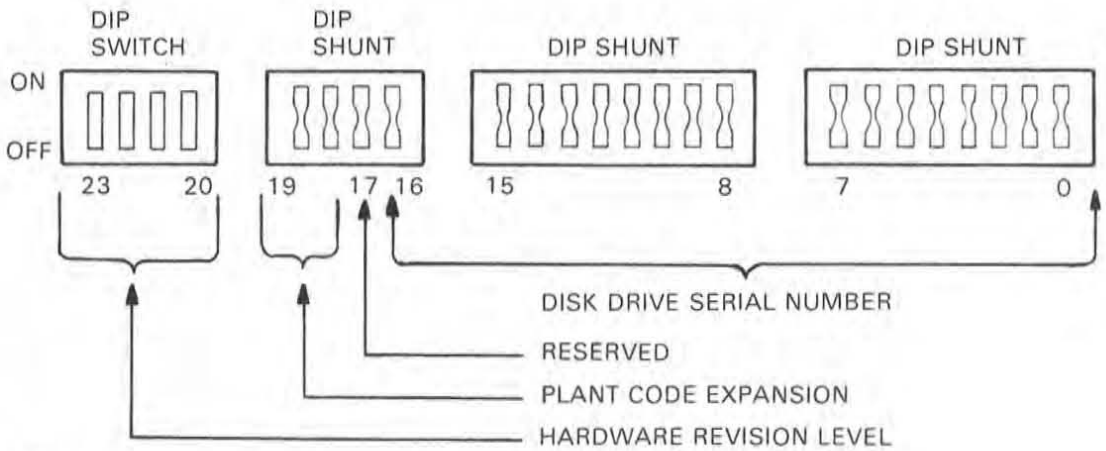


Figure 2-20 Operator Control Panel Removal



NOTES

1. WHEN REPLACING THE OPERATOR CONTROL PANEL, THE SHUNTS SHOULD BE SET ON THE NEW PANEL TO RESEMBLE THE OLD PANEL. BREAK EACH SHUNT BY PUSHING DOWN ON IT WITH A SMALL SCREWDRIVER. EACH SHUNT LEFT INTACT REPRESENTS A LOGICAL 0. A BROKEN SHUNT REPRESENTS A LOGICAL 1.

CZ-0641

Figure 2-21 Setting the Drive Serial Number

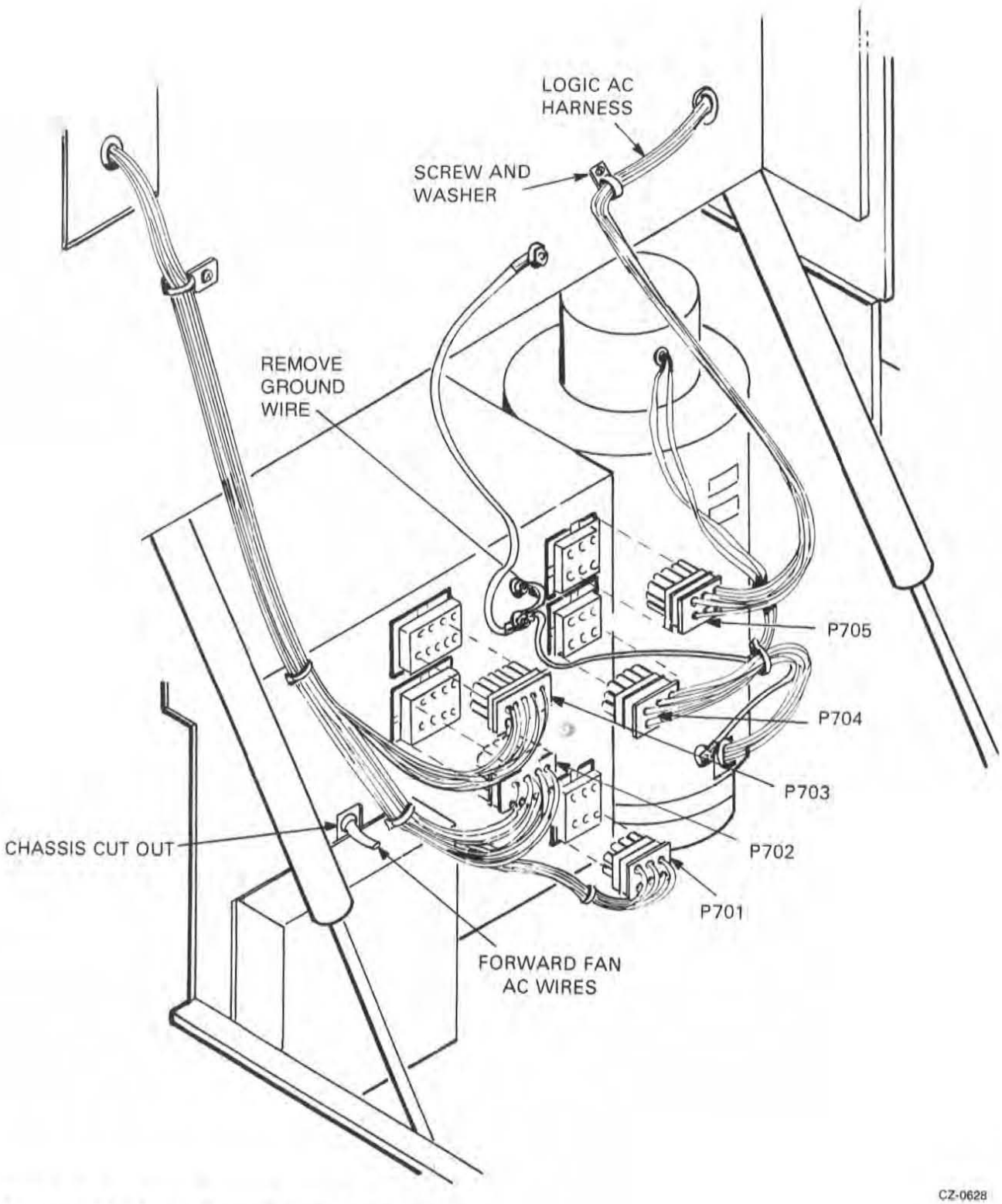


Figure 2-22 Drive Power Supply Connectors

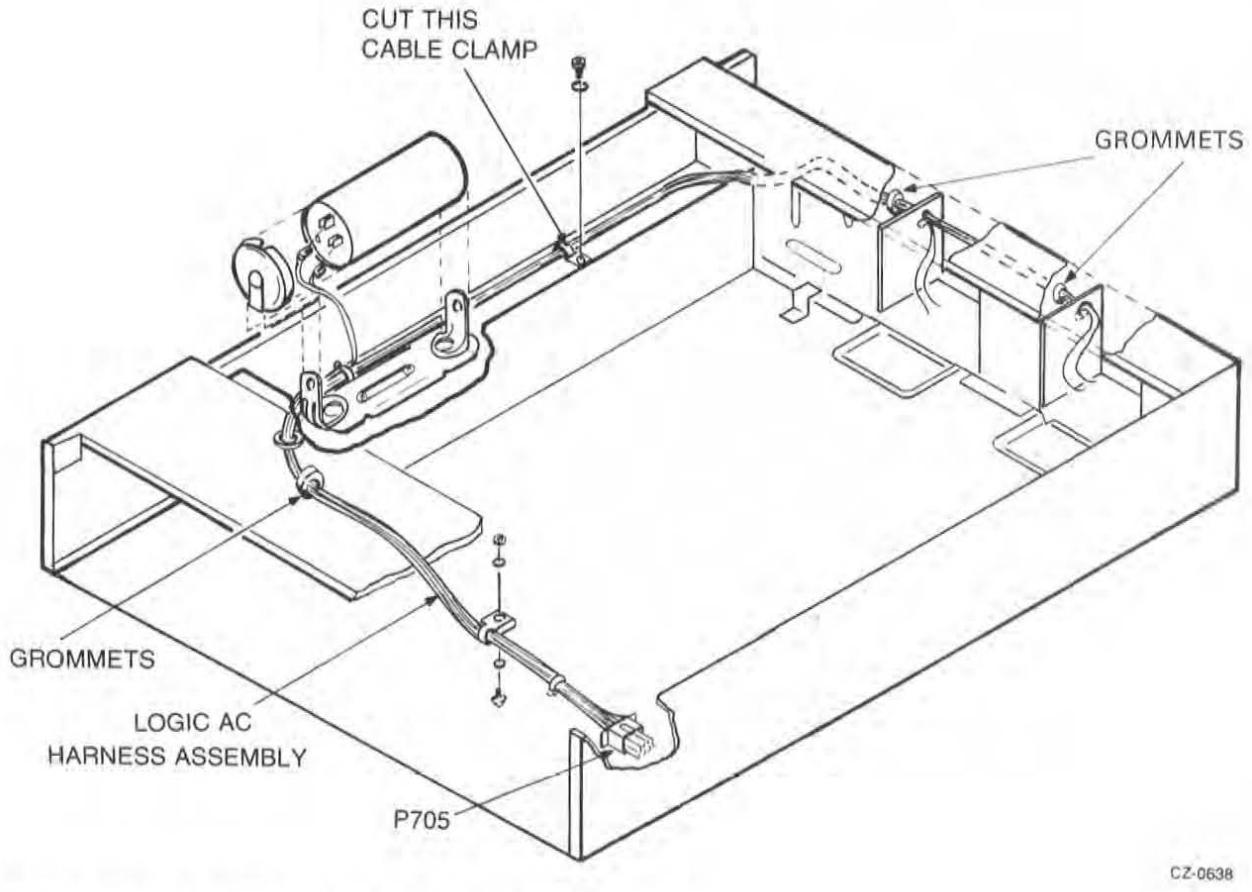


Figure 2-23 Logic AC Harness Assembly Removal

2.21 REMOVING THE DRIVE POWER SUPPLY

Refer to Figures 2-22 and 2-24 while performing this procedure.

1. Unplug the power supply ac line cord from the power controller at the base of the cabinet.
2. Raise the drive logic chassis. (Paragraph 2.11.)
3. Unplug connectors P701, P702, P703, P704, and P705 from the drive power supply.
4. Remove the ground wire from the top ground terminal located on the front of the power supply.
5. Free the ac power cord from any cable clamps or cable ties that hold it on the cabinet.
6. Remove the six 1/4 inch hex head screws from the rear of the power supply.
7. Pull the power supply out of the rear of the drive guiding the two fan wires through the chassis cutout.

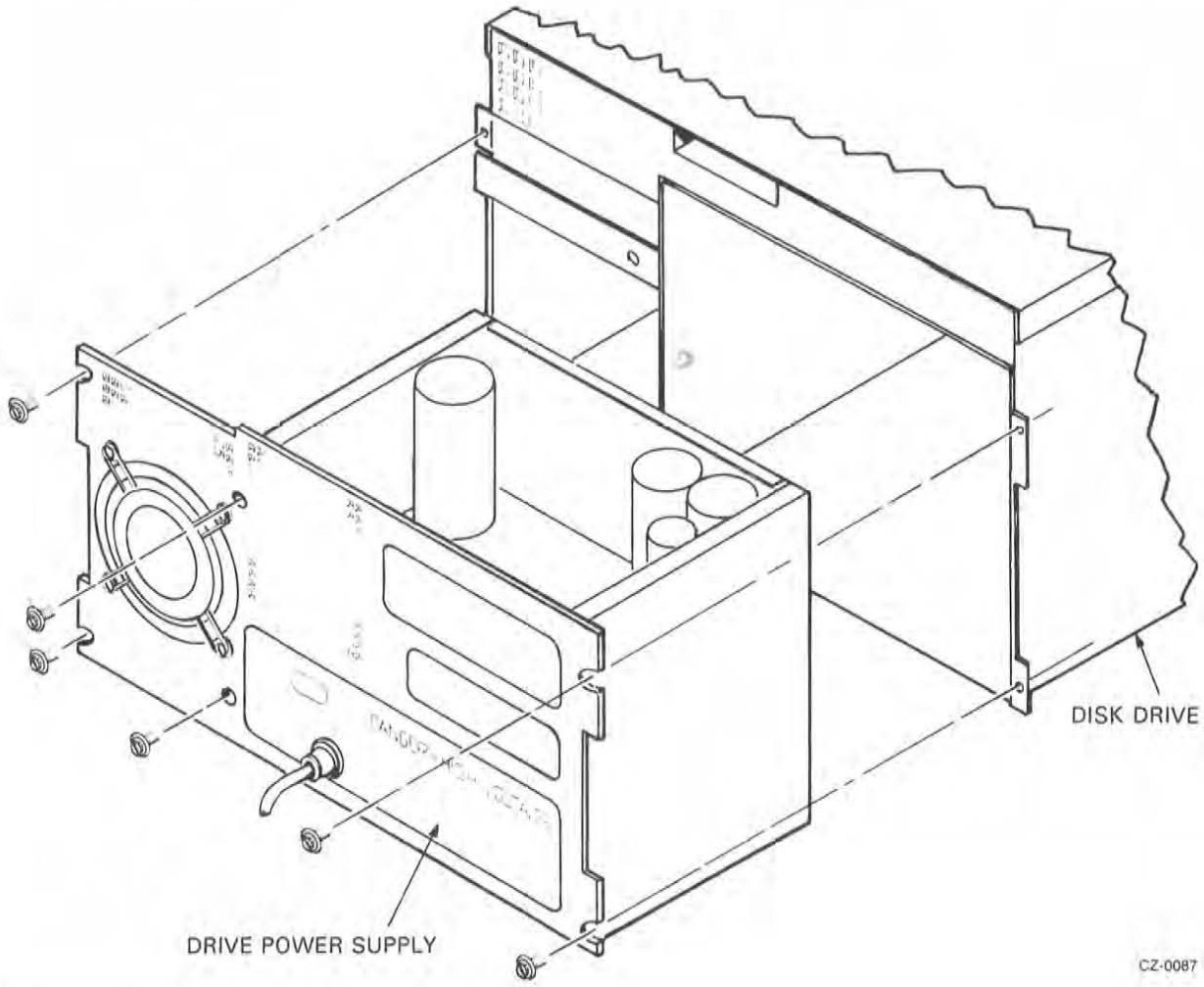


Figure 2-24 Drive Power Supply Removal

2.22 REMOVING THE POWER SUPPLY FANS

Refer to Figures 2-24 and 2-25 while performing this procedure.

1. Remove the drive power supply. (Paragraph 2.21.)
2. Unplug the black connector from the fan.
3. Remove the four screws that hold the fan in place.
4. Remove the fan guard if the rear fan is to be removed.
5. Remove the four Tinnerman nuts from the old fan and mount them on the new fan.

2.23 REMOVING THE HDA SPEED AND TEMPERATURE SENSORS

Refer to Figures 2-26 and 2-27 while performing this procedure.

1. Raise the drive logic chassis. (Paragraph 2.11.)

CAUTION

Place the HDA positioner lock in the locked position before removing the HDA.

2. Remove the HDA with the read/write module in place. (Paragraph 2.16.)
3. Unplug connector P501 from the read/write module.
4. Remove the quick-connect terminals from the temperature sensor.
5. Remove the temperature sensor by turning it clockwise.
6. Remove the two screws that hold the speed sensor on the HDA.
7. Remove the speed sensor assembly.

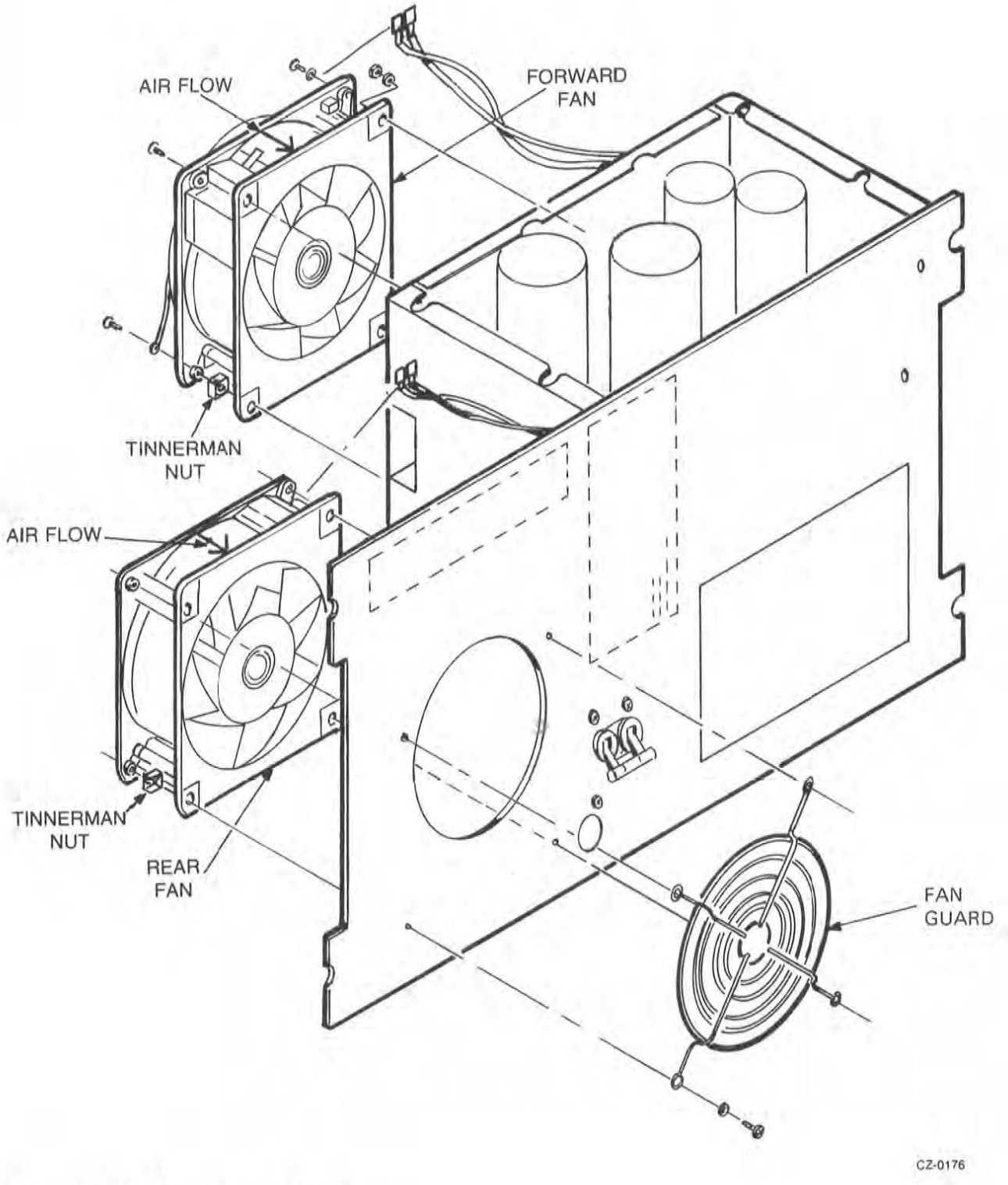


Figure 2-25 Power Supply Fan Removal

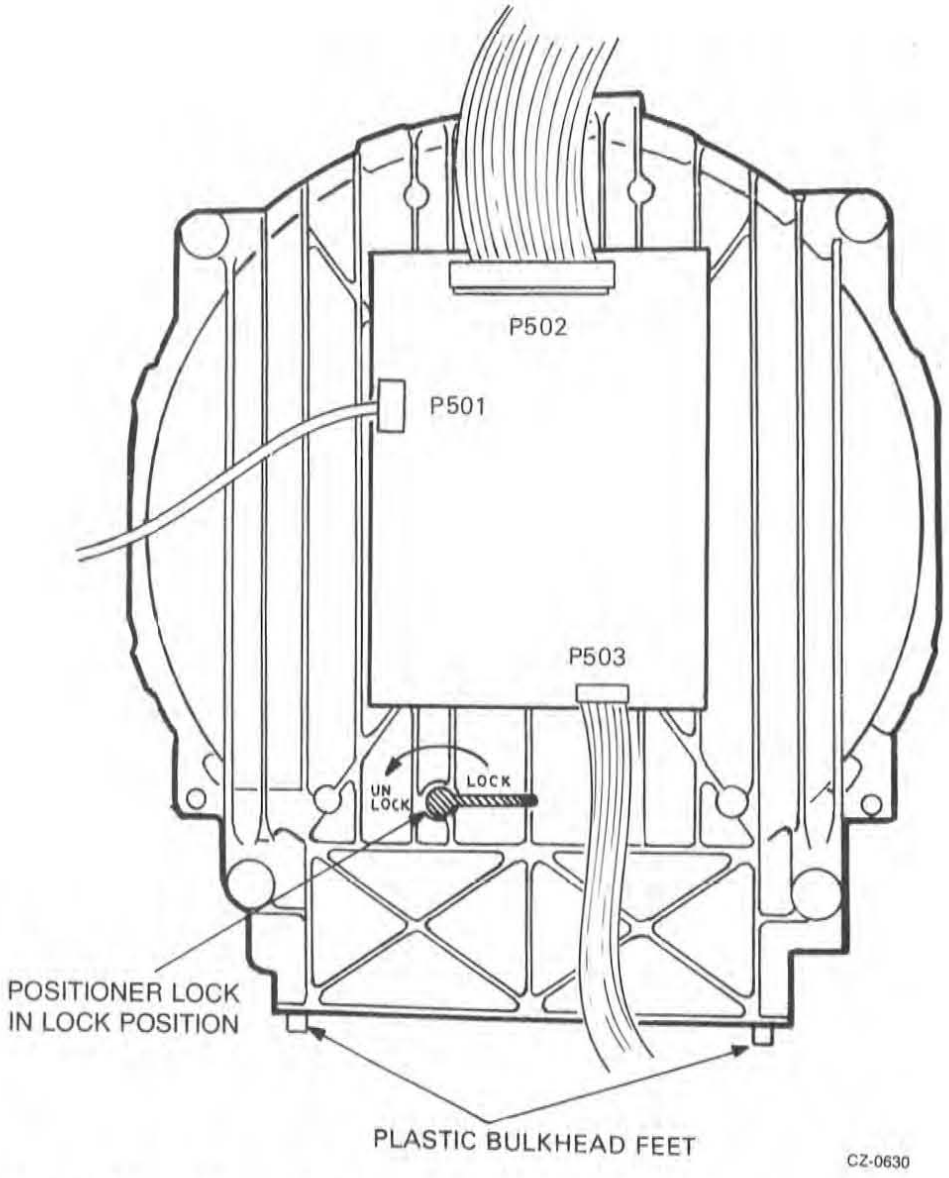


Figure 2-26 HDA and Read/Write Module (Top View)

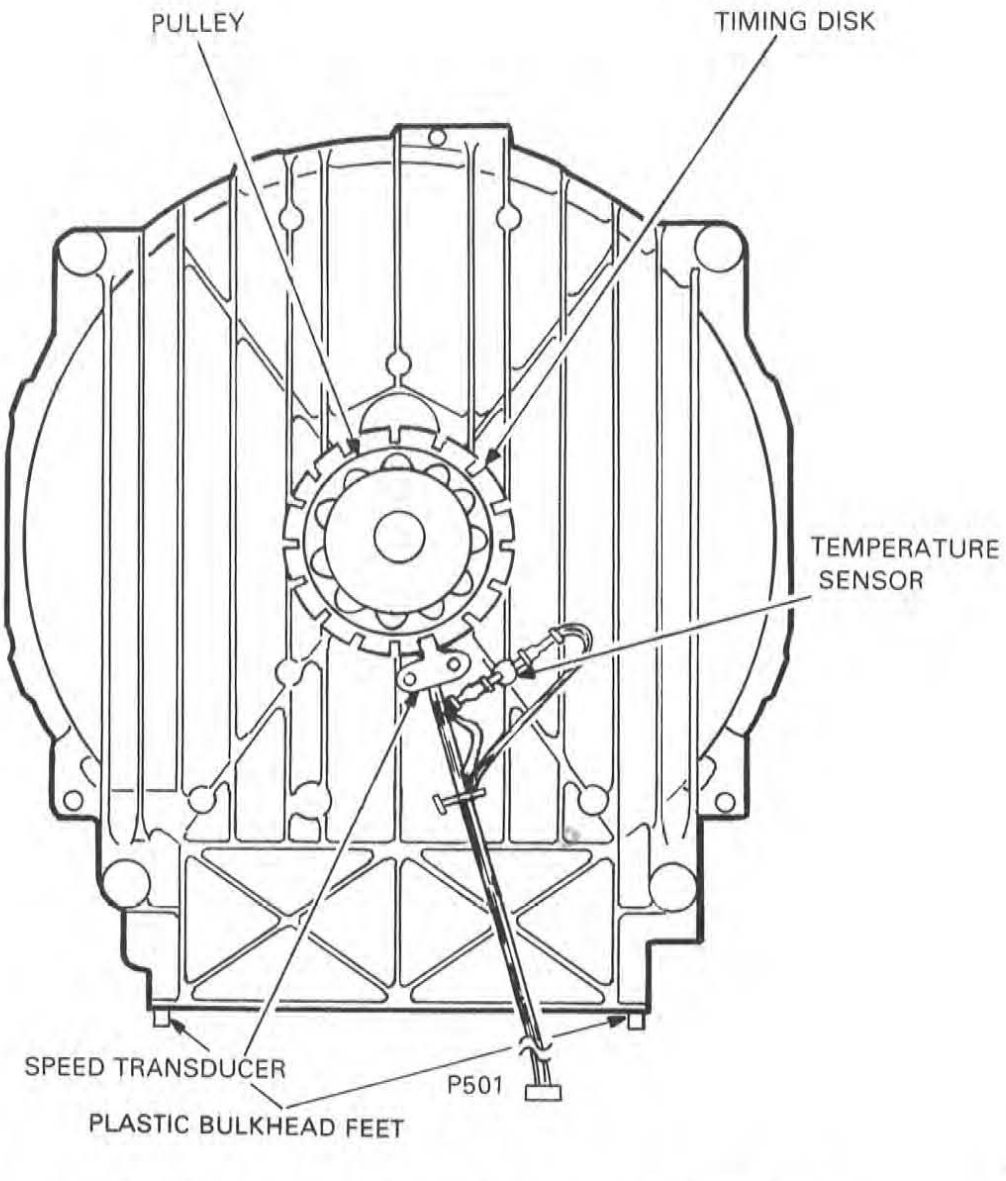


Figure 2-27 HDA Speed and Temperature Sensor (Bottom View)

2.24 REMOVING THE BELT TENSION MICROSWITCH

Refer to Figure 2-28 while performing this procedure.

1. Remove the HDA. (Paragraph 2.16.)
2. Remove the screw that holds the belt tension switch to the chassis side wall.
3. Remove the microswitch from its bracket.
4. Unplug the two quick connect terminals from the microswitch.

NOTE

Check that the microswitch leads are replaced exactly as removed. Figure 2-28 shows the correct placement.

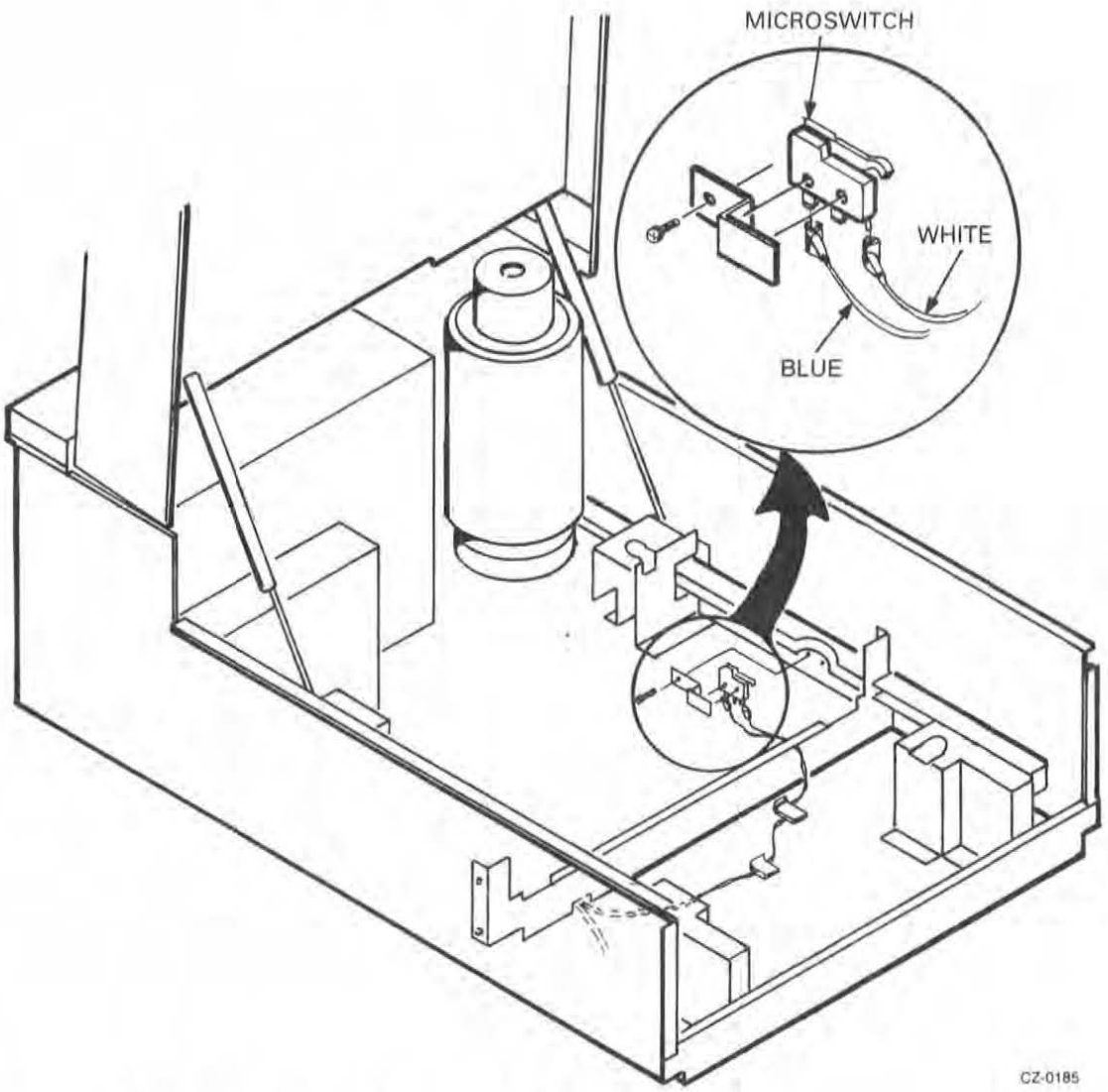


Figure 2-28 Belt Tension Microswitch Removal

2.25 REMOVING AND REPLACING THE SPINDLE BELT

Refer to Figure 2-29 while performing the following removal and replacement procedures.

2.25.1 Removing the Spindle Belt

1. Remove the HDA. (Paragraph 2.16.)
2. Lift the belt off the motor pulley and pull it forward.

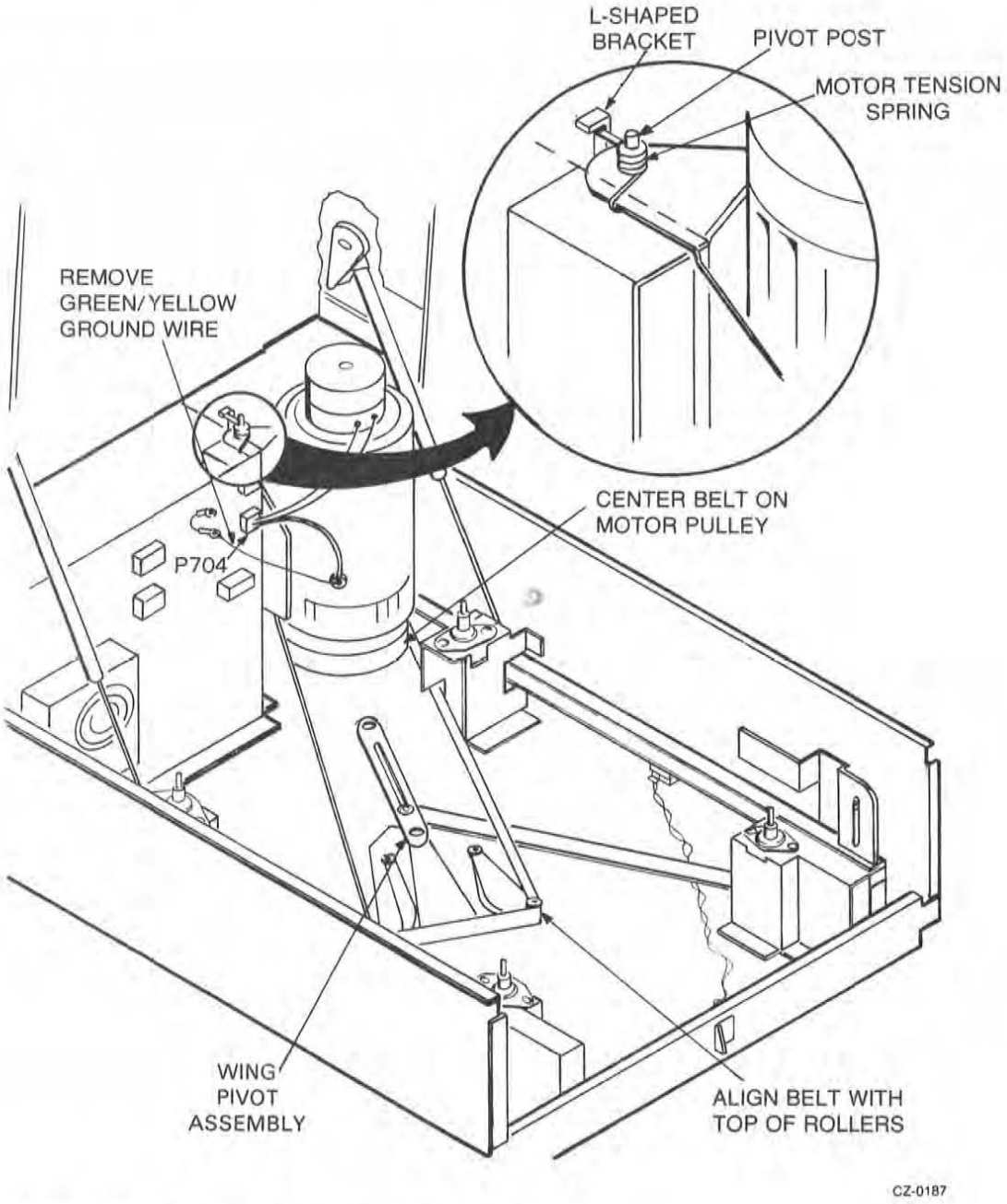


Figure 2-29 Belt and Motor/Brake Removal

2.25.2 Replacing the Spindle Belt

1. Slide the new belt under and around the motor pulley checking that the smooth side of the belt faces in toward the motor pulley.

2. Center the belt on the motor pulley.

3. Slide the other end of the belt over the two nylon roller bearings located on the wing pivot assembly.

NOTE
The belt should be positioned so that the top of the belt is flush with the top of the nylon rollers.

4. Replace the HDA.

2.26 REMOVING THE MOTOR/BRAKE ASSEMBLY

Refer to Figures 2-29 and 2-30 while performing this procedure.

1. Remove the HDA. (Paragraph 2.16.)

2. Remove the two hex nuts on the drive motor ground wire. The ground wire is green and yellow and grounds the motor to the power supply chassis.

3. Remove the drive motor ground wire from the grounding bolt.

4. Unplug connector P704 from the power supply chassis.

5. Slide the spindle motor drive belt off the wing pivot assembly and the motor pulley. (Paragraph 2.25.)

6. Remove the drive motor tension spring. A pair of long-nosed pliers will have to be used to maneuver the spring out from under the L-shaped bracket.

7. Lift the motor/brake assembly off the pivot posts.

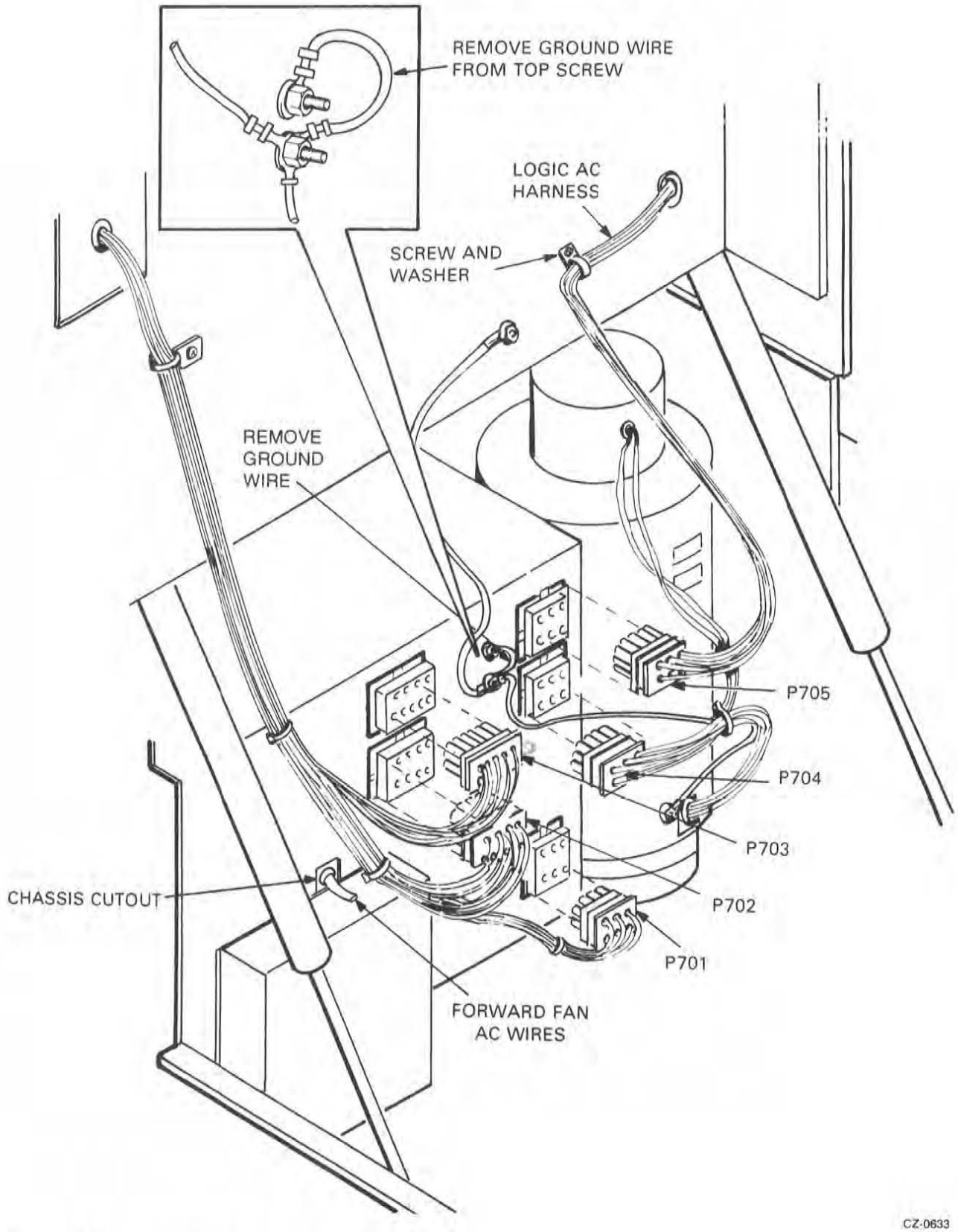


Figure 2-30 Removing the Ground Wire

2.27 REMOVING THE MOTOR ACTUATOR ASSEMBLY

Refer to Figures 2-31 and 2-32 while performing this procedure.

1. Remove the HDA. (Paragraph 2.16.)
2. Remove the motor/brake assembly. (Paragraph 2.26.)
3. Remove the four kepnuts and washers that hold the lower air baffle.
4. Remove the baffle.
5. Remove the three retaining rings and washers that hold the motor actuator assembly in place.
6. Remove the screw and beveled washer that secures the belt tension lever and lock spring to the chassis side wall.

NOTE

The hollow side of the beveled washer should face the locking spring upon reassembly.

7. Slide the motor actuator assembly off the chassis side wall studs.
8. Remove the motor actuator assembly through the front of the drive.

2.28 REMOVING THE WING PIVOT ASSEMBLY

Refer to Figure 2-33 while performing this procedure.

1. Remove the HDA. (Paragraph 2.16.)
2. Lift the belt off the wing pivot assembly.
3. Remove the two screws, three retaining rings, and washers that hold the wing pivot assembly in place.
4. Lift the wing pivot assembly off the locating studs and slide it under the lower air baffle.

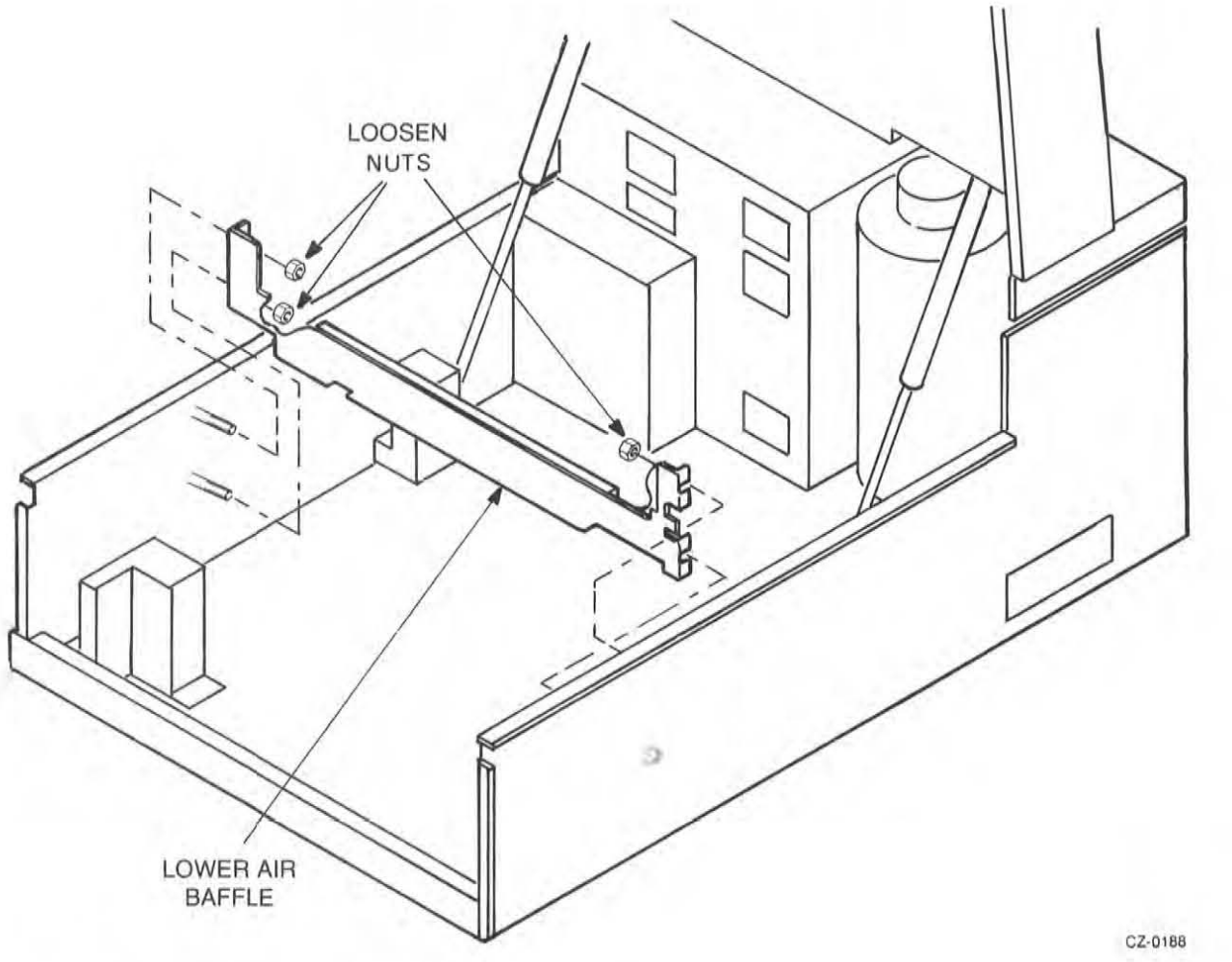


Figure 2-31 Lower Air Baffle Removal

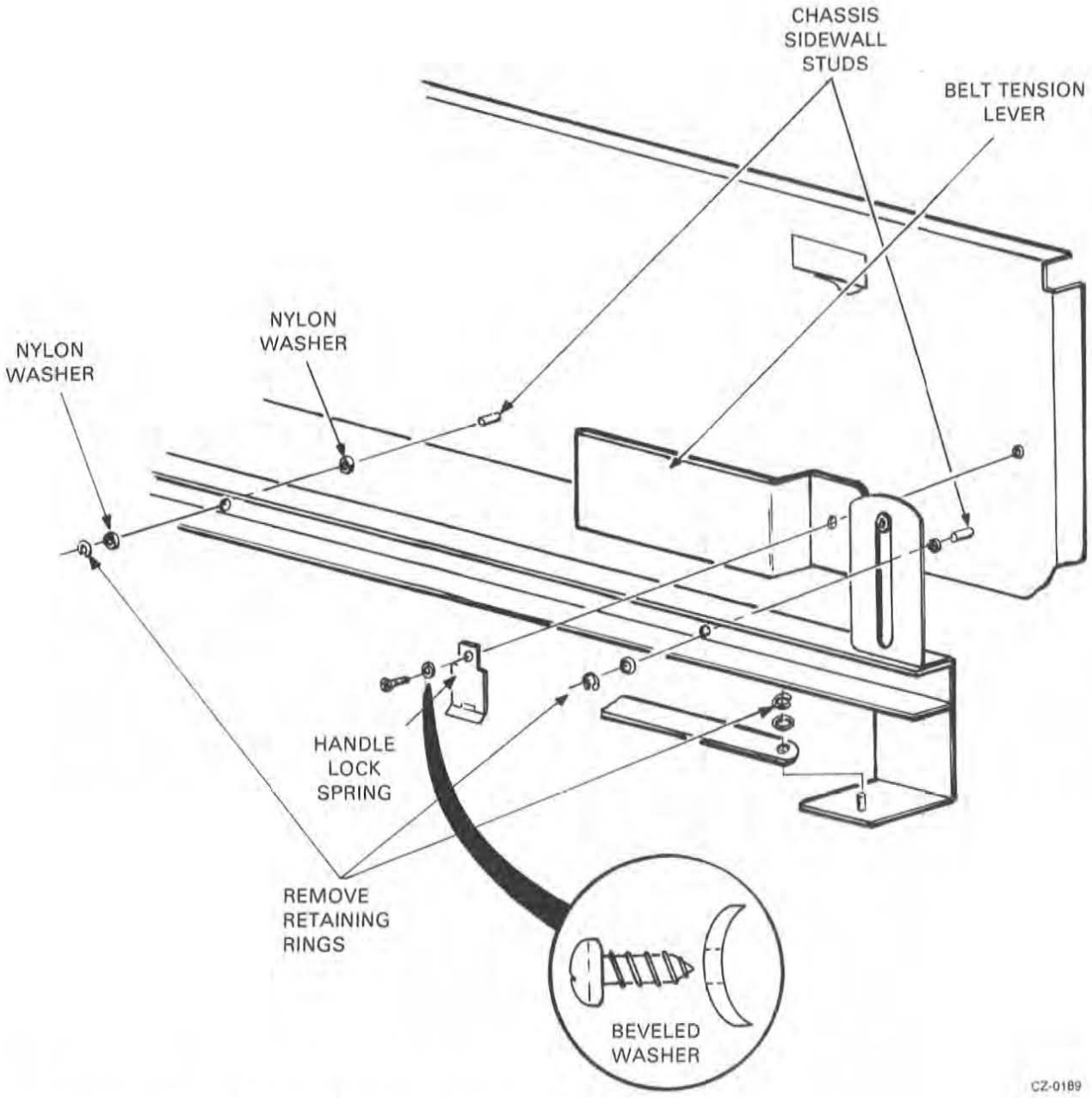


Figure 2-32 Motor Actuator Assembly Removal

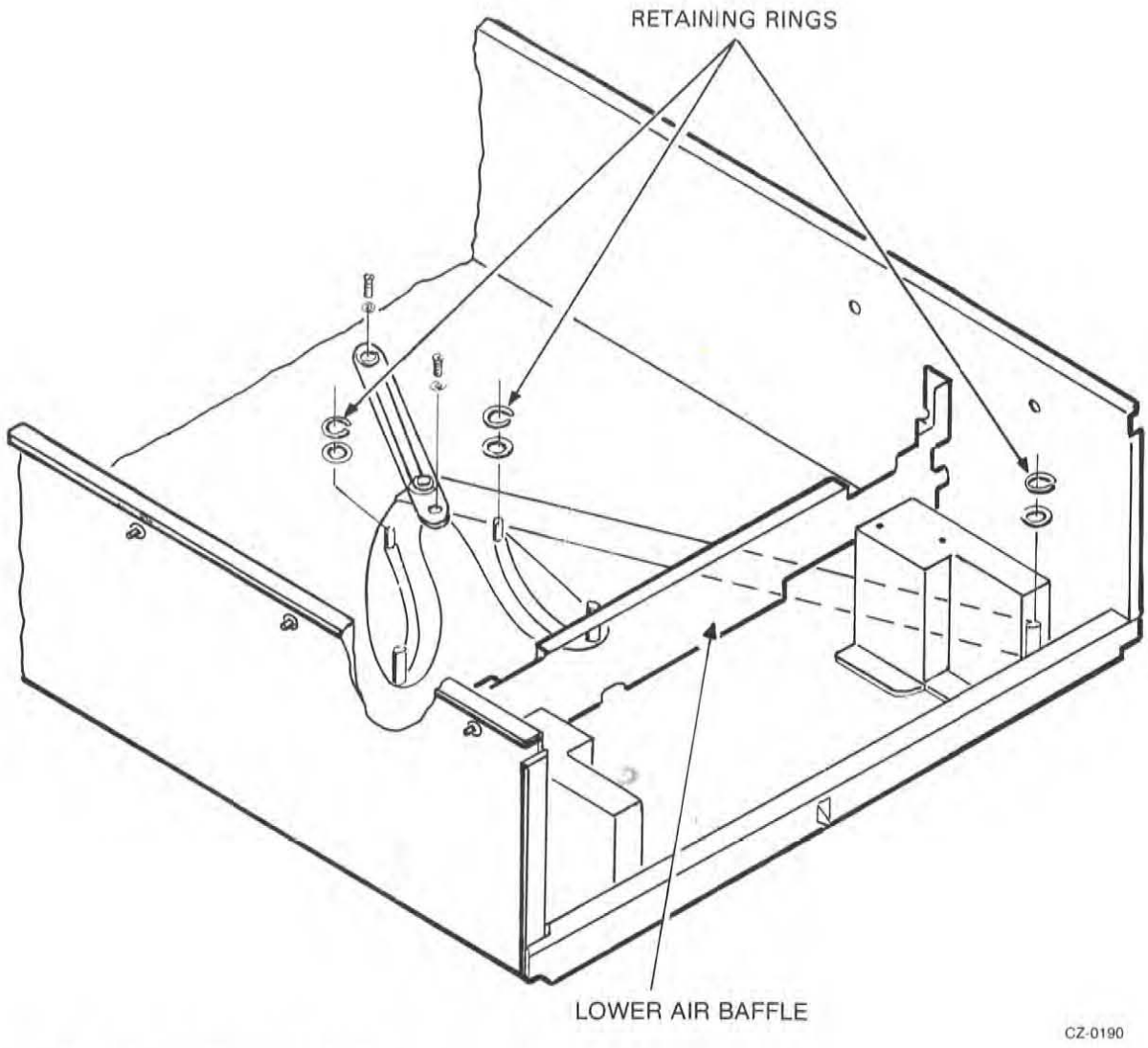


Figure 2-33 Wing Pivot Assembly Removal

CHAPTER 3
ADJUSTMENTS

3.1 INTRODUCTION

This chapter describes the adjustment procedures for the RA80 Disk Drive. The only adjustments that can be made are on the belt tension and on the servo.

3.2 BELT TENSION ADJUSTMENT

The belt tension should be checked whenever the motor, drive belt, or HDA is replaced. Since the belt may stretch slightly with use, belt tension should also be examined any time a drive corrective action call is made. To check or adjust the belt tension, use the following procedure with the HDA in place (refer to Figures 3-1 and 3-2).

1. Spin down the HDA by depressing the RUN/STOP switch on the operator control panel.

2. Raise the drive logic chassis and determine if the brass-colored slider is flush with the outside edge of the metal reference marker.

NOTE

If the brass-colored slider needs adjustment, turn off the ac power at the power controller.

3. Loosen the locking nut and adjust the belt tension screw until the brass-colored slider is flush with the reference marker shown. Turn the screw clockwise to move the slider forward.

4. Tighten the locking nut after the adjustment is made.

5. Restore the ac power to the drive and spin up the HDA by depressing the RUN/STOP switch.

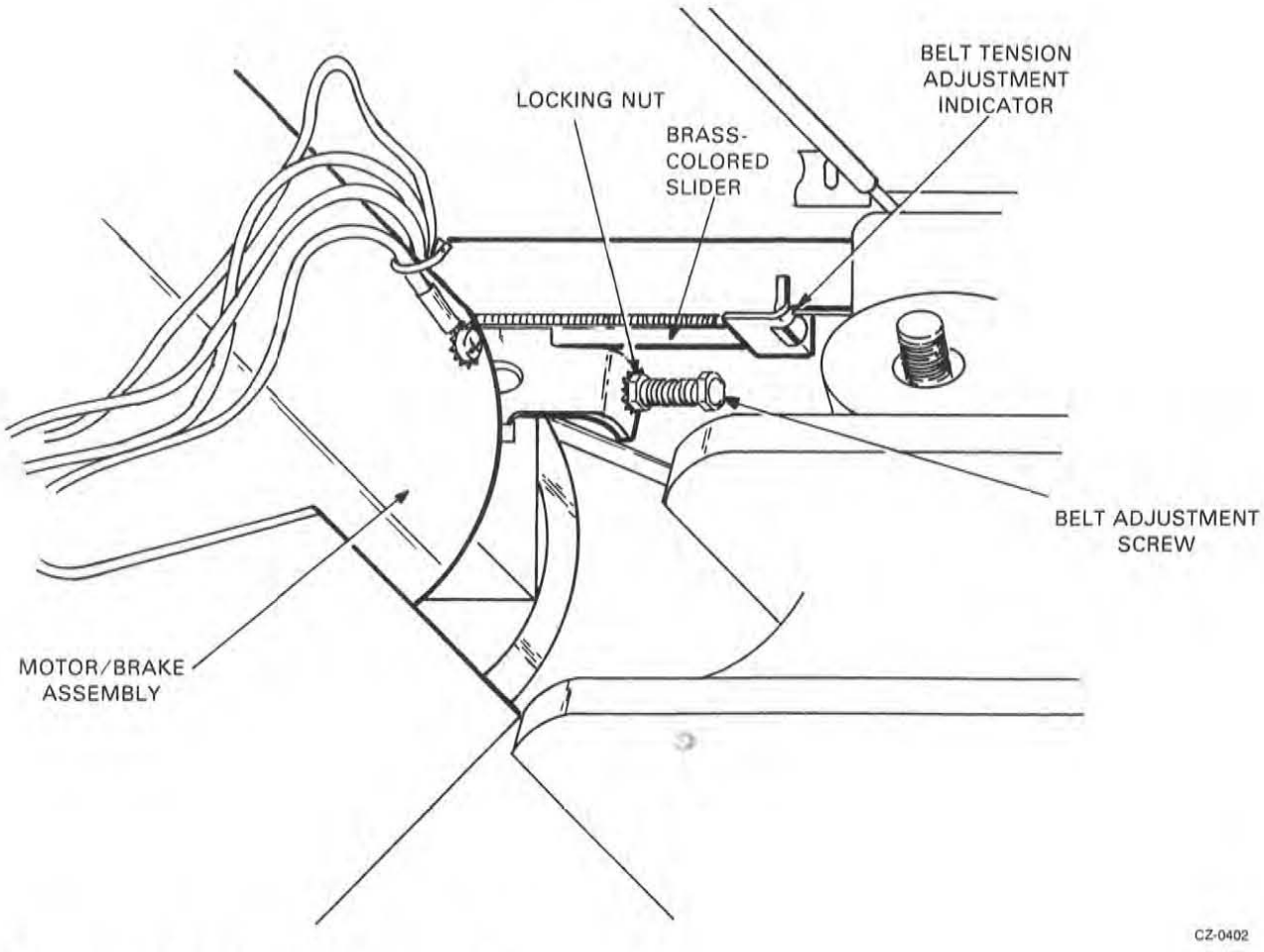


Figure 3-1 Belt Tension Adjustment Screw

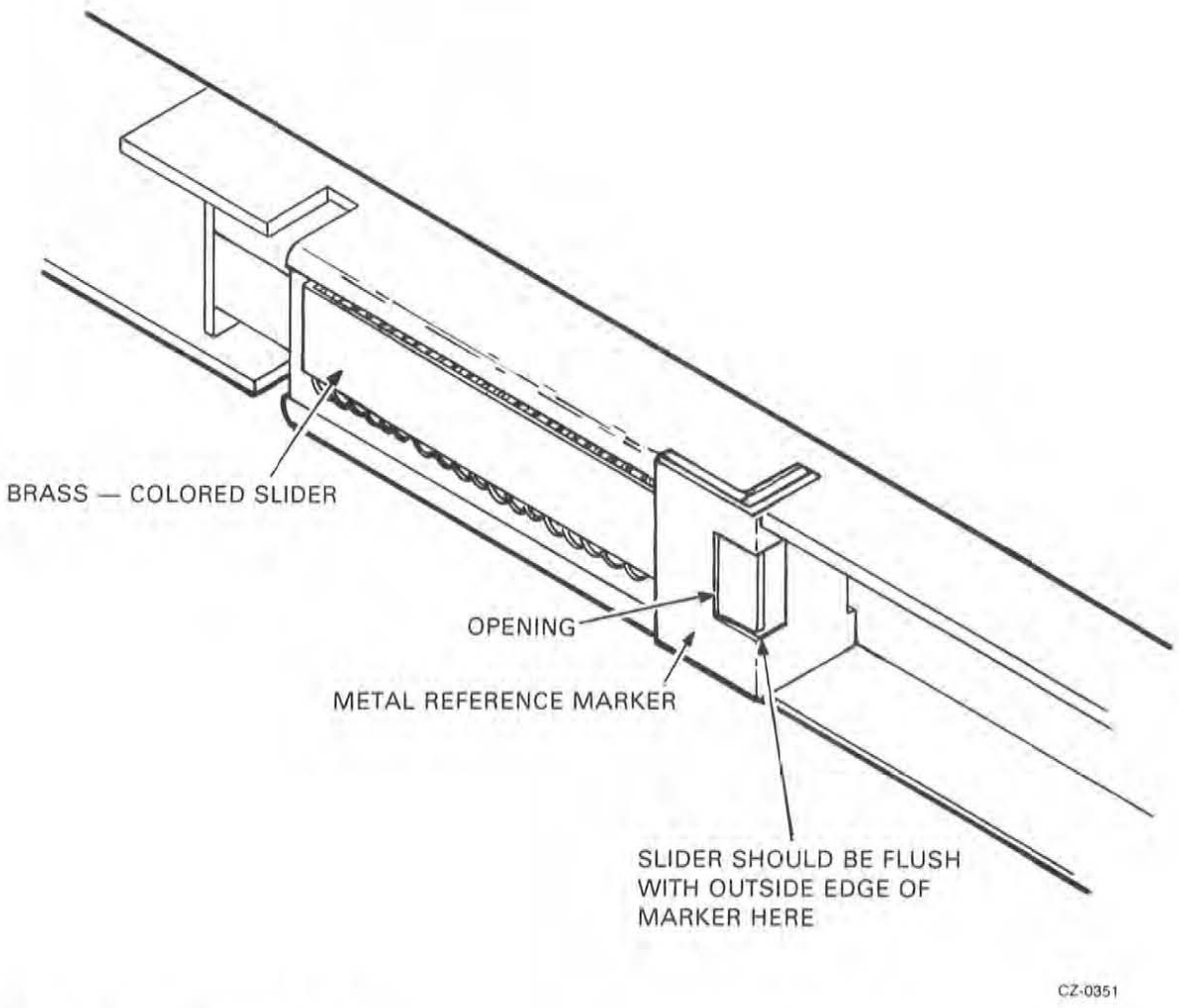


Figure 3-2 Reference Marker

3.3 SERVO ADJUSTMENTS

The servo adjustments must be performed after replacing a servo module or HDA. Refer to Figure 3-3 for the locations of the maintenance controls. Use the following procedures to make these adjustments.

1. Switch on the circuit breaker at the rear of the H766 power supply.
2. Enter the diagnostic mode using the following procedure.
   a. Turn the rotary switches to FF.
   b. Push the ENTER switch. The LEDs blink FF.
   c. Push the ENTER switch again and the LEDs display a steady state FF.
   d. Turn the rotary switches to 00.
   e. Push the ENTER switch. The LEDs display a steady state 00.
   f. Push the ENTER switch again and the LEDs display the blinking EC prompt.
3. Call up the static servo test using the following procedure.
   a. Turn the rotary switches to 27.
   b. Push the ENTER switch. The LEDs momentarily display 27 and then AA when the test completes successfully. If an error code occurs, replace the servo module and repeat the adjustments from Step 1. If the test ends with an AA, proceed to Step c.
   c. Push the ENTER switch again and the LEDs display the blinking EC prompt.
4. Spin up the disk using the following procedure.
   a. Turn the rotary switches to 1E.
   b. Push the ENTER switch. The LEDs momentarily display 1E and then a steady state E7.
   c. Push the RUN/STOP switch on the operator control panel. The drive spins up and the LEDs display an AA when the spin-up is complete.

      NOTE
      The READY indicator is not lit when the disks are spinning in the diagnostic mode.

   d. Push the ENTER switch and the LEDs display the blinking EC prompt.
5. Call up the servo velocity adjustment utility using the following procedure.
   a. Turn the rotary switches to 26.
   b. Push the ENTER switch. The LEDs momentarily display 26, then momentarily E7, then one or two LEDs remain lit. If the LEDs display an error code of 7C, 7D, or 7E, proceed to Step c; otherwise, proceed to Step d.

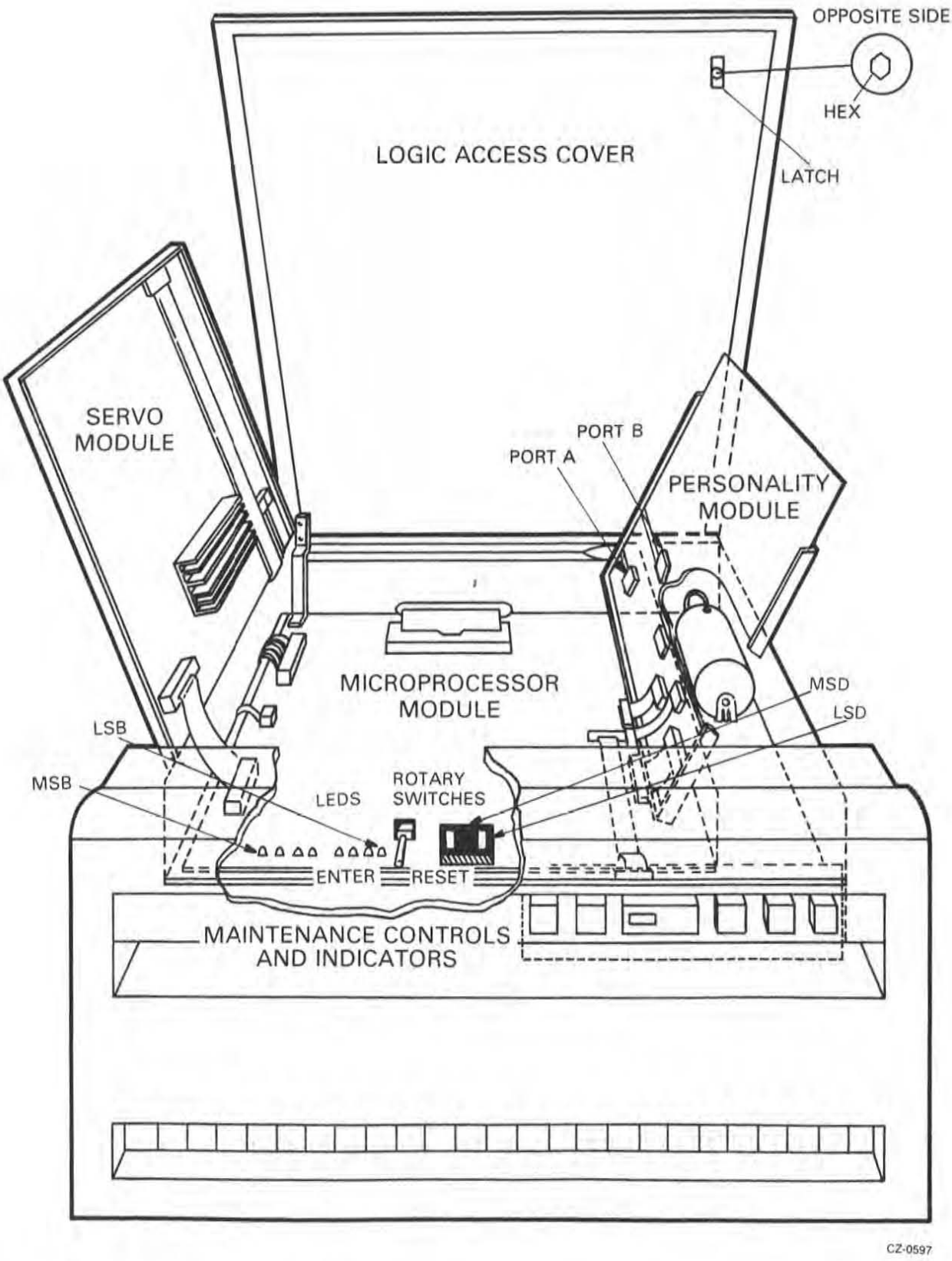


Figure 3-3 RA80 Maintenance Controls

c. Look up the applicable LED error code in the following procedure and do what it says.

- Error code 7C – when this error code is indicated by the LEDs, the positioner access time is too slow and requires that the R281 potentiometer be turned counterclockwise two revolutions. (Refer to Figure 3-4.) Push the ENTER switch to return to the blinking EC prompt and repeat Step 5.
- Error code 7D – this error code indicates that there are too many seek errors. The servo module or the HDA may be defective.
- Error code 7E – when this error code is indicated by the LEDs, the positioner access time is too fast and requires that the R281 potentiometer be turned clockwise two revolutions. Push the ENTER switch to return to the blinking EC prompt and repeat Step 5.

d. Read the LED display. If any LEDs other than the center two LEDs are on or flashing (refer to Figure 3-5), adjust the R281 potentiometer on the servo module (refer to Figure 3-4). The objective is to adjust the potentiometer until only the center two LEDs are flashing. Rotate R281 clockwise to move the LED pattern towards the ENTER switch, and counterclockwise to move the LED pattern away from the ENTER switch.

NOTE
Rotate the servo velocity adjustment potentiometer (R281) slowly since there is a time delay before the LED patterns react to the adjustment.

e. Run the servo velocity test for 20 minutes until the temperature stabilizes in the positioner motor.

f. Read the LED display. Readjust the R281 potentiometer until only the center two LEDs of the display are on or flashing.

g. Turn the rotary switches to DD.

h. Push the ENTER switch. The LEDs display AA or blink EC depending on how long the ENTER switch is pushed.

i. Push the ENTER switch again if the LEDs display an AA. The LEDs should then display the blinking EC prompt.

6. Call up the entire unit test using the following procedure.

a. Turn the rotary switches to 25.

b. Push the ENTER switch. The LEDs momentarily display 25 and then E7. Wait until all the tests are run and the LEDs display AA.

c. Push the ENTER switch and the LEDs display the blinking EC prompt.

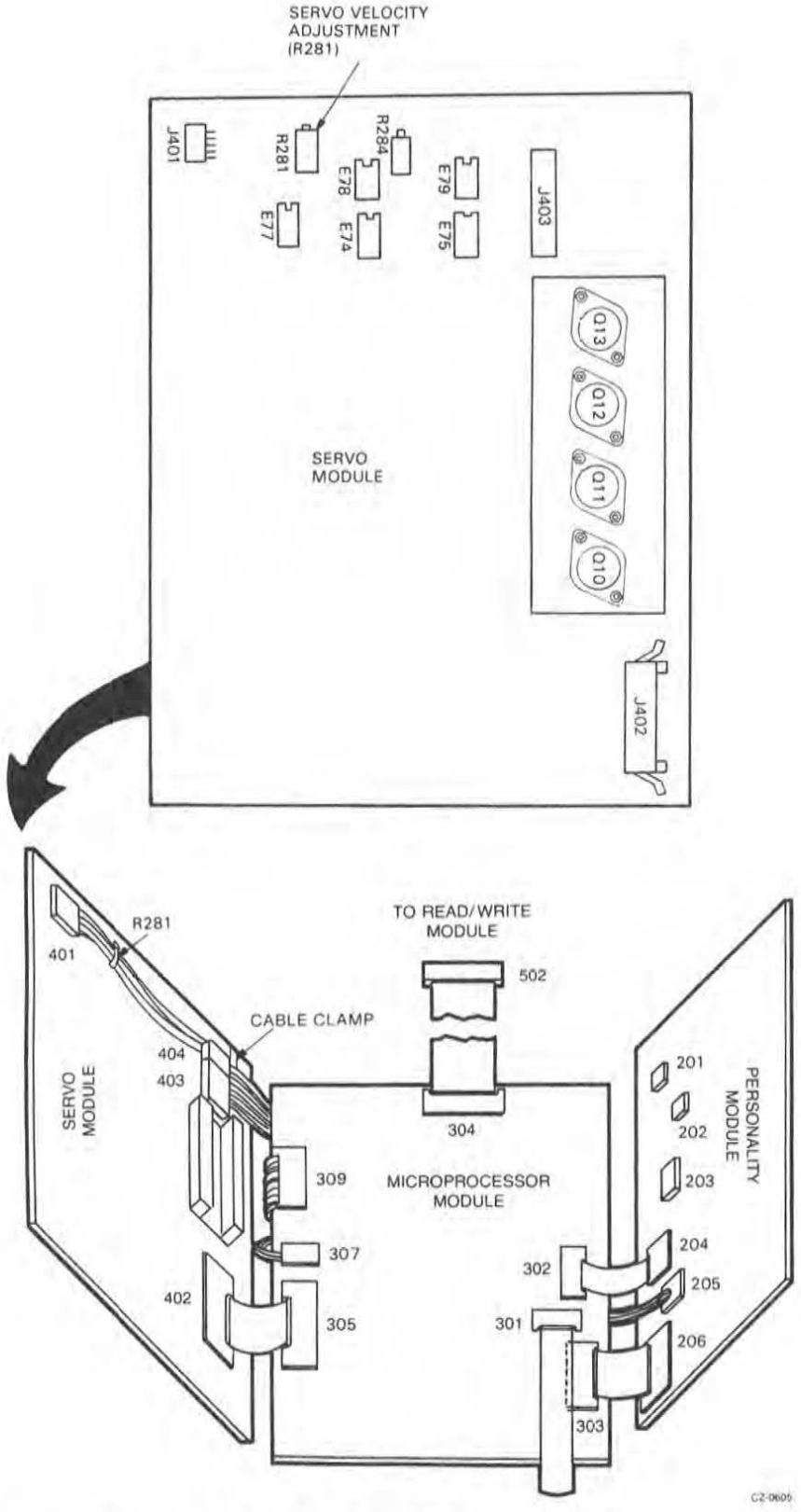


Figure 3-4 Servo Velocity Adjustment

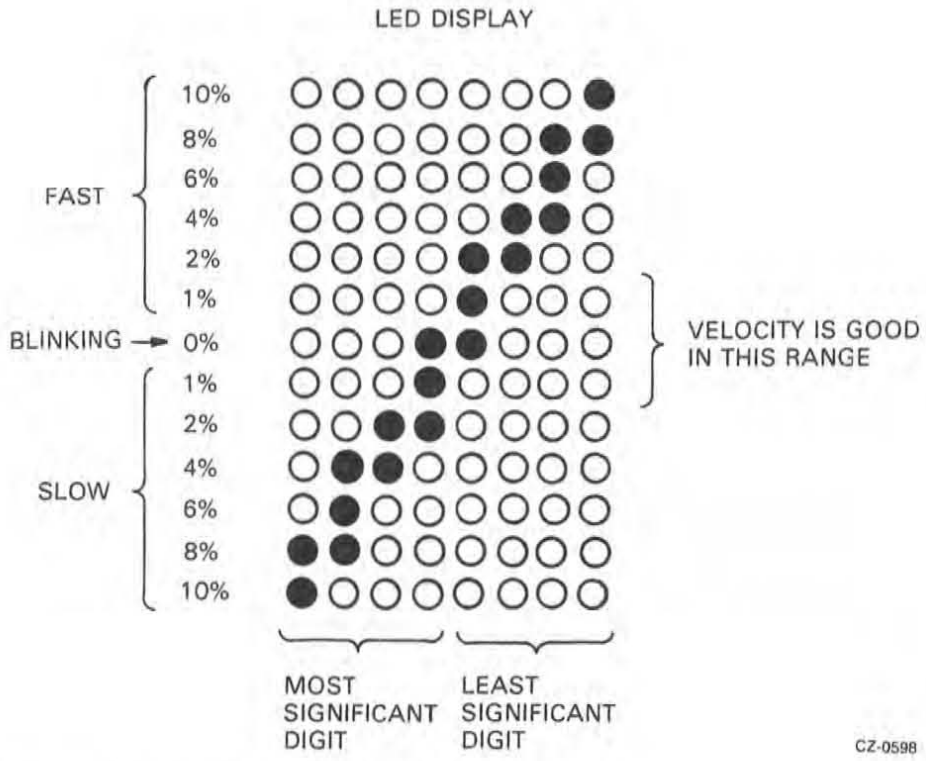


Figure 3-5 Servo Velocity LED Pattern

7. Return on-line using the following procedure.

   a. Turn the rotary switches to 1D.

   b. Push the ENTER switch. The LEDs momentarily display 1D and then E7.

   c. Turn the rotary switches to 00.

   d. Push the ENTER switch. The LEDs display 00 and the drive is on-line. (A test for determining if the drive is on-line is to push the WRITE PROTect switch. If the WRITE PROTect indicator lights, then the drive is back in the on-line mode.)

CHAPTER 4
DRIVE-RESIDENT DIAGNOSTICS

4.1 INTRODUCTION

This chapter describes the RA80 firmware diagnostic capabilities. It begins with a description of the functional firmware fault and error codes. It also shows the location of the maintenance controls and how to use them. Finally, a description of each utility routine and diagnostic test is presented.

4.2 FUNCTIONAL AND DIAGNOSTIC FIRMWARE

The functional and diagnostic firmware are two distinct software modules in the RA80 Disk Drive. The functional firmware controls the spin-up cycle, seek command, and recalibrate command. It also performs fault monitoring and interface handshaking operations. When the drive is operating under the control of the functional firmware, it is in the functional mode (on-line). When the drive is operating under the control of the diagnostic firmware, it is in the diagnostic mode. The diagnostic firmware controls the drive-resident tests and utilities. These tests and utilities are invoked after the drive is placed in the diagnostic mode.

4.3 FUNCTIONAL FIRMWARE FAULT CODES

Functional firmware general fault codes are reported through the operator control panel. A general fault code is obtained by entering the fault display mode when the FAULT indicator is on. To enter the fault display mode, push the FAULT switch. All of the indicators should remain on until the FAULT switch is released, providing a method of checking all the indicators. Upon releasing the FAULT switch, the indicators display the general fault code as illustrated in Figure 4-1. The LEDs on the microprocessor module further define the general faults with specific hexadecimal error codes.

To exit the fault display mode, push the FAULT switch. This action stores the LED error code in the area of the RAM allocated for previous faults. If the fault has been cleared, the firmware returns the operator control panel to its previous state. If the fault still exists, the FAULT indicator lights again.

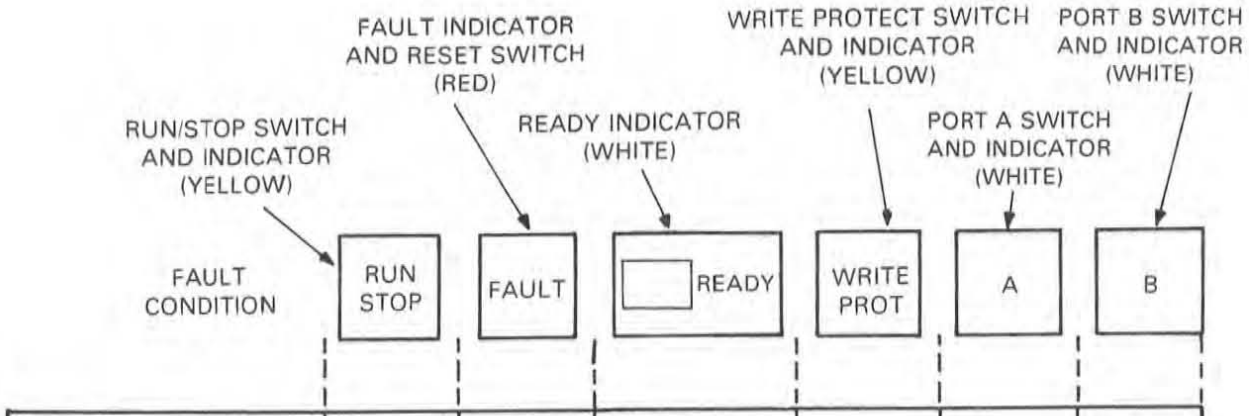
The read-/write-unsafe and invalid interface command fault conditions can be corrected by the CPU. In this case, the FAULT indicator goes off without operator intervention. If the fault cannot be corrected by the CPU, the FAULT indicator remains on and the fault must then be cleared through manual intervention.

4.4 GENERAL FAULT CODE DESCRIPTIONS

Eleven general fault codes are reported through the operator control panel indicators. These eleven general fault codes are described in the following paragraphs.

4.4.1 Spin-Up Fault

The spindle begins rotating when the RUN switch is pushed. During the spin-up cycle, the firmware monitors spindle acceleration and speed. If the spindle takes too long to attain its final speed, is rotating too slowly, or is not accelerating, the firmware stops the spindle and turns on the FAULT indicator. If the fault display mode is entered, a spin-up fault is indicated. The internal LED display indicates the specific error code.



SPIN-UP		ON				ON
R/W DIAGNOSTIC	*	ON			ON	
READ/WRITE UNSAFE	*	ON			ON	ON
SPINDLE MOTOR INTERLOCK	*	ON		ON	ON	
SPINDLE MOTOR SPEED	*	ON		ON	ON	ON
DRIVE DISABLED BY "DD" BIT	*	ON	ON			ON
HDA OR SERVO BOARD OVERTEMPERATURE	*	ON	ON		ON	
MICROCODE	*	ON	ON		ON	ON
SERVO DIAGNOSTIC	*	ON	ON	ON		ON
INITIAL RECAL	*	ON	ON	ON	ON	
MICROPROCESSOR HARDCORE TEST	ON	ON	ON	ON	ON	ON

*THE INDICATOR STATE WILL BE THE SAME AS IT WAS BEFORE THE FAULT SWITCH WAS PUSHED

CZ-0428

Figure 4-1 Operator Control Panel General Fault Indicators

4.4.2 Read/Write Diagnostic Fault

After the firmware has performed the initial recalibration, it executes the read/write diagnostics before lighting the drive READY indicator. The tests listed below are performed during the read/write diagnostics.

- Read-only test
- Write/read test

If the read/write diagnostics fail, the FAULT indicator lights and the spin-up cyle cannot be completed. (Refer to Figure 4-2.) Pushing the FAULT switch lights all the operator panel indicators, then displays the fault. If the FAULT switch is pushed again, the firmware runs the read/write diagnostics again. If the read/write diagnotics pass, the firmware lights the drive READY indicator.

4.4.3 Read-/Write-Unsafe Fault

A read-/write-unsafe fault is detected any time the microprocessor interrupt is enabled. Upon recognizing a read-/write-unsafe condition, the firmware sets drive fault in the status register on the personality module. The internal LED display indicates the specific error code.

4.4.4 Spindle Motor Interlock Fault

Before the firmware enters the spin-up routine, the belt tension microswitch is checked. If the belt tension lever is in the released position, the firmware senses this, lights the FAULT indicator, and does not spin up the disk. If the disks are already spinning and a failure occurs in the belt tension interlock circuit, the disks spin down and the FAULT indicator lights. The internal LED error code is 23.

4.4.5 Spindle Motor Speed Fault

If the spindle slows to below 3420 r/min, the firmware turns on the FAULT indicator and stops the spindle motor. The internal LED error code is 26.

4.4.6 Drive Disabled by DD Bit

The controller asserts the DD bit to remove the drive from service whenever it detects that operational thresholds are being exceeded. The DD bit can be reset by the controller using the RIP bit and a change mode command.

4.4.7 HDA or Servo Overtemperature

There are two temperature sensors in the RA80 Disk Drive; one on the bottom of the HDA and the other on the servo module heatsink. If the firmware detects an overtemperature condition, the FAULT indicator lights.

4.4.8 Microcode Fault

Near the end of each ROM is an instruction to jump to a microcode fault routine. This instruction is only executed if the firmware malfunctions. When this instruction is executed, it calls the microcode fault routine. This routine turns off the spindle motor, displays an 06 in the LEDs, and lights the FAULT indicator on the operator control panel.

4.4.9 Servo Diagnostic Fault

After the spindle is up to speed, the firmware performs static servo diagnostics before doing the initial recalibration of the heads. The tests listed below are run during the static servo diagnostics.

- Track counter test
- Servo position loop test
- Servo velocity loop test

If the servo diagnostics fail, the FAULT indicator lights and the spin-up cycle cannot be completed. (Refer to Figure 4-2.) Pushing the FAULT switch lights all the operator control panel indicators, then displays the fault. If the FAULT switch is pushed again, the firmware runs the servo diagnostics again.

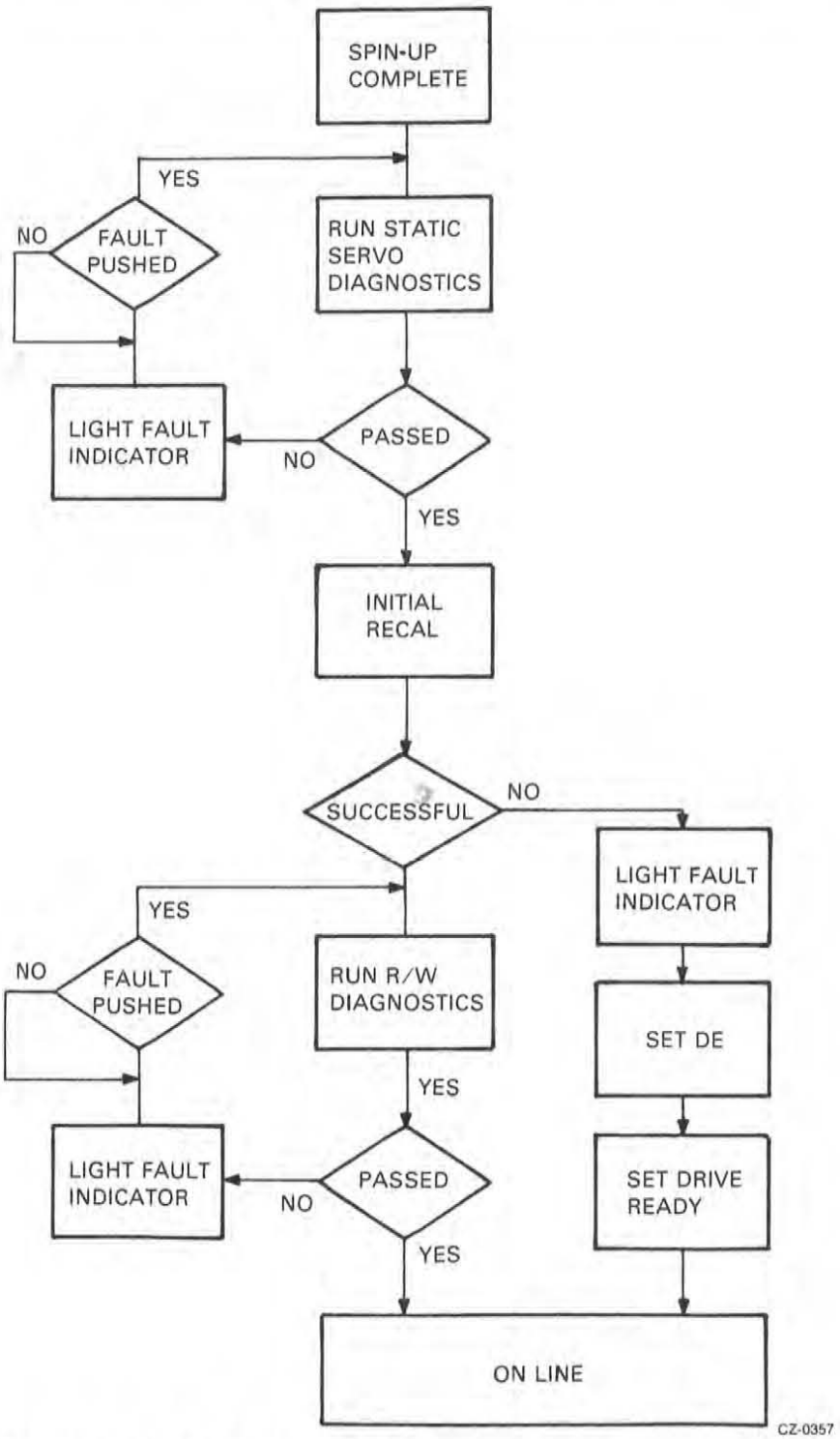


Figure 4-2 Spin-Up Diagnostic Flowchart

4.4.10 Initial Recalibration Fault

During the spin-up cycle, the firmware does the initial recalibration after running the static servo diagnostics (refer to Figure 4-2). If the initial recalibration is successful, the firmware runs the read/write diagnostics. If the initial recalibration is unsuccessful, the firmware lights the FAULT indicator, sets the SEEK INCOMPLETE signals and lights the drive READY indicator. The CPU then comes back with a recalibration command which clears the FAULT indicator and retries a recalibration. The CPU retries the recalibration until it is successful or until the number of retries for the CPU are executed.

4.4.11 Microprocessor Hardcore Test Fault

The microprocessor hardcore test is run when power is applied through CB1 on the back of the drive. POWER UP RESET starts the firmware at memory location 0000. The firmware then performs tests on the hardware listed below.

- Microprocessor
- ROMs
- RAMs
- Microprocessor bus
- Servo bus
- Personality bus
- Sector/byte counter

If the hardcore test fails, all the indicators on the operator control panel are on. More detailed information on the hardcore failure can be obtained by examining the internal LED display.

4.5 SERVO ERRORS

Servo errors do not light the FAULT indicator, but display an error code in the internal LEDs. The drive flags the CPU with a DE bit and waits for a drive clear command. Upon recognizing the drive clear command, the firmware stores the LED code in memory, clears the internal LED display, and then calls the recalibrate routine.

4.6 DIAGNOSTIC FIRMWARE CONTROLS AND INDICATORS

The RA80 internal maintenance controls and LEDs are located in the front left-hand corner of the microprocessor module. Refer to Figure 4-3. These maintenance controls and LEDs are divided into three functional areas, rotary switches, an ENTER/RESET switch, and an 8-bit LED display.

4.6.1 Rotary Switches

There are two hexadecimal rotary switches (S2 and S3) next to each other. The switch on the right is the least significant hexadecimal digit. The switch on the left is the most significant hexadecimal digit. The rotary position of each switch is shown by a hexadecimal digit on the top surface of the switch. The test-select codes and input parameters to run the utility routines and diagnostics are entered through the rotary switches.

4.6.2 ENTER/RESET Switch

The ENTER/RESET switch (S1) is a dual-throw, center-off-position switch. When the toggle switch is pushed to the right, it is in the RESET position, which is marked by an R on the circuit module. When the toggle switch is pushed to the left, it is in the ENTER position, which is marked by an E on the circuit module.

4.6.2.1 The ENTER Position - The ENTER position is periodically polled by the firmware to determine if a new action is required. The ENTER switch provides the Field Service engineer with a means of communicating with the firmware. For example, to initiate a diagnostic test, a test-select code is placed in the rotary switches and then the ENTER switch is pushed to start the test.

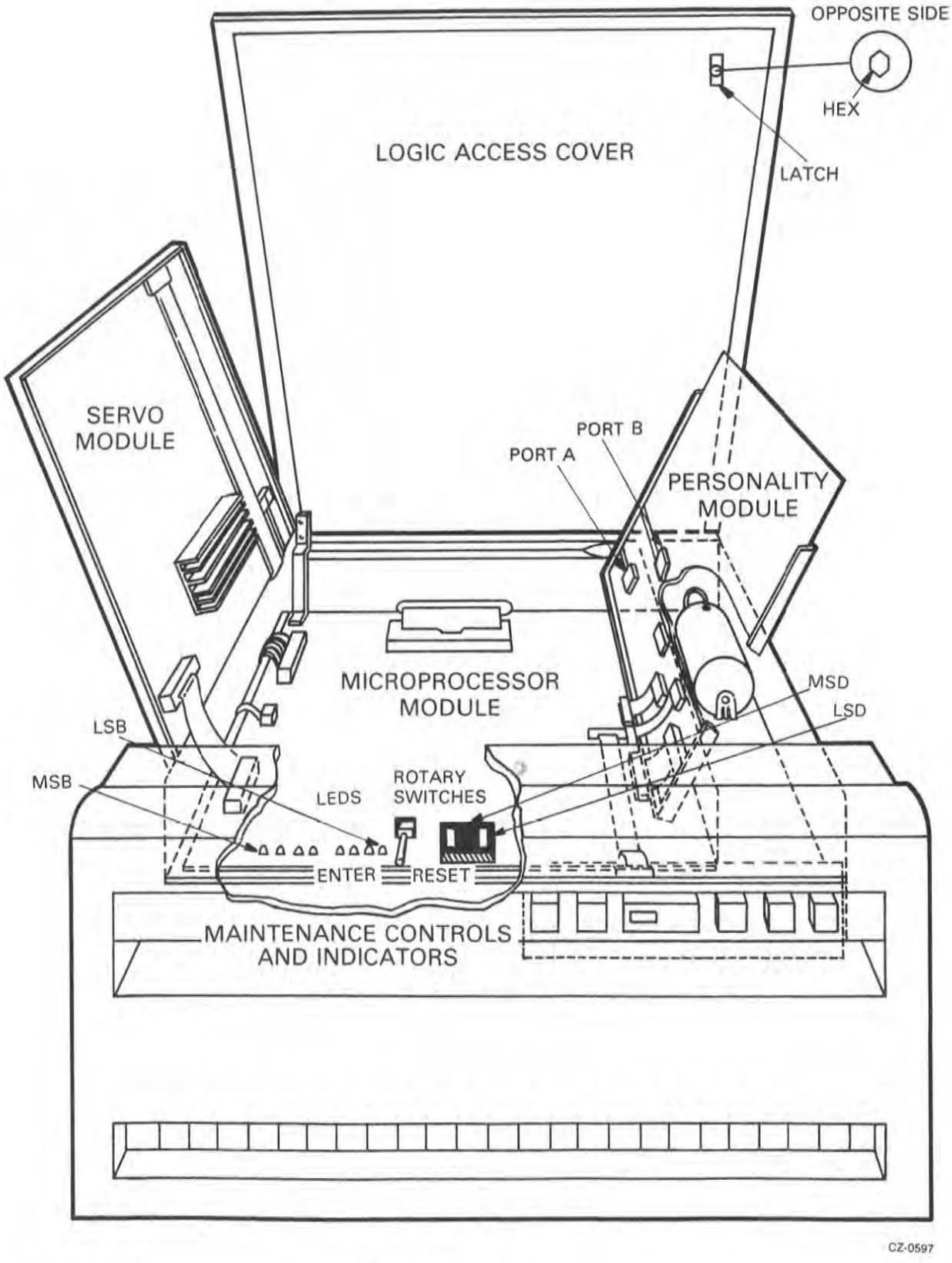


Figure 4-3 Microprocessor Module Maintenance Controls

4.6.2.2 The RESET Position - The RESET position is connected to interrupt line RST 5.5 on the microprocessor chip. RST 5.5 is a maskable interrupt that is disabled by the firmware during seek operations. Pushing the RESET switch while the drive is in functional mode forces the firmware to reinitialize the drive logic. If the drive halts on a hardcore fault during power-up, pushing the RESET switch places the drive in the functional mode. While in the diagnostic mode, pushing the RESET switch terminates the test.

4.6.3 LED Display

The LED display consists of a row of eight LEDs. The least significant LED is to the right. The codes displayed in the LEDs are read as two hexadecimal digits (four LEDs to each digit). Several kinds of codes appear in the LEDs. They display error codes, prompt codes, test complete codes and test-select codes for entry verification.

4.7 TEST-SELECT CODES

Each utility routine and diagnostic test has a unique test-select code. The test-select codes are listed in Table 4-1. To call a routine or test, enter the test-select code into the rotary switches and push the ENTER/RESET switch to the ENTER position. When the firmware detects that the switch has been pushed to the ENTER position, it reads the contents of the rotary switches. The LEDs momentarily display the test-select code to verify that the firmware has received it.

Table 4-1 Test-Select Codes

Test-Select Codes	Tests
01	Examine diagnostic extended status area utility
02	Examine previous error utility
03	Examine drive state utility
04	Examine operational counters utility
05	Memory-examine up utility
06	Memory-examine down utility
07	Three-module bus test
08	Microprocessor module bus test
09	Microprocessor and personality module bus test
0A	Microprocessor and servo module bus test
0B	Personality module microsequencer test
0C	Sector/byte counter test
0D	Control panel, serial number, and drive number test
0E	Head-select multiplexer test
0F	General purpose counter test
11	Track counter test
12	Read/write fault force test
13	Servo position loop test
14	Servo velocity loop test
15	Servo functional test

Table 4-1 Test-Select Codes (Cont.)

Test-Select Codes	Tests
16	Random seek test
17	Seek - seek test with input parameters
18	Seek - seek test with fixed parameters
19	Incremental seek test with input parameters
1A	Incremental seek test with fixed parameters
1B	Read-only test
1C	Write/read test
1D	Go on-line
1E	Spindle control utility
1F	Head-select and seek utility
20	Maintenance controls and indicators test
21	Read-only cylinder formatter utility
22	Logic tests
23	Servo tests
24	Read/write tests
25	Entire unit test
26	Servo velocity adjustment utility
27	Static servo test
28	Personality board test
CF	Loop mode set utility
FF	(Go off-line)

A test termination code of DD provides the user with a way of halting diagnostic drive tests. Most tests complete so quickly that they cannot be terminated in this way. However, seek-seek and incremental seek tests are of sufficient duration that they may be terminated prematurely in this manner. Also, tests with the loop mode set can be run for long periods of time. To stop all tests safely, use the DD test termination code.

If the drive is in the loop mode, it is still in the loop mode when the tests are halted with a DD code.

4.8 PROMPT AND STEADY STATE CODES

The firmware uses several blinking prompt codes and steady state codes to notify the user when it is waiting for information, is active, or has completed a test. These codes are listed in Table 4-2.

4.9 RA80 UTILITY AND DIAGNOSTIC PROCEDURES

The procedures for running the drive utilities and diagnostics are described in Table 4-3. If a fault occurs during these tests, check the error code table in Chapter 5.

Table 4-2 Prompt and Steady State Codes*

Code	Significance
Blinking EC (•••o••oo)	Indicates that firmware is waiting for a test-select code.
Blinking 01 (ooooooo•), 02 (ooooooo•o), 03 (ooooooo••), or 04 (oooooo•oo)	Indicates the number of the current input parameter required by the diagnostic utility or test.
Blinking EE (•••o•••o)	Indicates an invalid test-select code has been entered. Requires entry of a valid test-select code.
Steady State AA (•o•o•o•o)	Indicates test completion. Return to the EC prompt before running another test.
Steady State E7 (•••oo•••)	Indicates that the firmware is actively executing a test.
Steady State Error Codes	Indicates why a test or utility has failed. Refer to LED error code list in Chapter 5. Push the ENTER switch to recover from an error code.

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures*

Test-Select Code	Operating Procedures
01	Examine diagnostic extended status area utility Purpose: to examine the 16 bytes of status information of the diagnostic test just run. a. Set rotary switch to 01 and push ENTER switch. LEDs momentarily display 01 (ooooooo•). LEDs then display the first status byte. b. Push the ENTER switch 15 times to display each remaining status byte in the LEDs. • Byte 1 = test-select code • Bytes 2 through 7 = input parameters • Byte 8 = test result (AA or error code) • Bytes 9 through 16 = diagnostic-dependent test data c. Push ENTER switch. LEDs display steady AA (•o•o•o•o). d. Push ENTER switch again. LEDs display blinking EC (•••o••oo).
02	Examine previous error utility Purpose: to examine the 16 latest LED error codes stored in the drive. a. Set rotary switch to 02 and push ENTER switch. LEDs momentarily display 02 (ooooo•o). LEDs display the most recent error code. b. Push the ENTER switch until all 16 error codes are examined. c. Push the ENTER switch again after examining all 16 error codes. LEDs display steady AA (•o•o•o•o). d. Push the ENTER switch again. LEDs display blinking EC (•••o••oo).
03	Examine drive state utility Purpose: to examine the twenty software state words maintained by the functional firmware.

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	a. Set rotary switch to 03 and push ENTER switch. LEDs momentarily display 03 (oooooo••). LEDs display contents of first state word. b. Push ENTER switch 19 times until all state words are examined. c. Push ENTER switch. LEDs display steady AA (•o•o•o•o). d. Push ENTER switch. LED display blinks EC (•••o••oo).
04	Examine operational counters utility Purpose: to examine the contents of the seek/read error counter, the seek counter, and the read and no enable counter. a. Set rotary switch to 04 and push ENTER switch. LEDs momentarily display 04 (ooooo•oo). LEDs display the first byte of the following list. • Seek/recal error counter contents Byte 1 – overflow indicator Byte 2 – low byte Byte 3 – high byte • Seek counter contents Byte 4 – overflow indicator Byte 5 – low byte Byte 6 – middle byte Byte 7 – high byte • Read and no enable counter contents Byte 8 – overflow indicator Byte 9 – low byte Byte 10 – high byte b. Push the ENTER switch 9 or more times to examine the remaining 9 bytes. c. Push the ENTER switch again. LEDs display steady AA (•o•o•o•o). d. Push the ENTER switch again. LEDs display blinking EC (•••o••oo).

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	NOTE **These counters are reset during drive power-up or whenever the RESET switch is pushed while the drive is on-line in the functional mode. When a counter overflows, its indicator byte contains an FF.**
05	Memory-examine up utility Purpose: to examine the contents of any memory location by incrementing through the address locations. a. Set rotary switch to 05 and push ENTER switch. LEDs momentarily display 05 (oooooooo). LEDs blink 01 (oooooooo). b. Set rotary switch to low byte of memory address and push ENTER switch. LEDs momentarily display contents of rotary switch. LEDs blink 02 (oooooooo). c. Set rotary switch to high byte of memory address push ENTER switch. LEDs momentarily display contents of rotary switch. LEDs display the eight bits of data stored at that address. d. Push the ENTER switch each time you wish to increment the address. e. Exit this utility by setting the rotary switch to DD. Push the ENTER switch. LEDs display either a steady AA (oooooooo) or a blinking EC (oooooooo). f. Push the ENTER switch again to return to the EC prompt if a steady AA is displayed.
06	Memory-examine down utility Purpose: to examine the contents of any memory location by decrementing through the address locations. This utility operates similarly to the memory-examine up utility described above. The only difference is that the address pointer is decremented each time the ENTER switch is pushed, instead of being incremented. Use a test-select code of 06 to call this utility.
07	Three-module bus test Purpose: this test checks the microprocessor data bus, the servo bus and the personality bus. All cables must be connected to serve this test.

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	a. Set rotary switch to 07 and push ENTER switch. LEDs momentarily display 07 (oooooooo). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
08	Microprocessor module bus test
	Purpose: this test checks the microprocessor data bus with the servo and personality modules disconnected. This test assumes that the three-module bus test was run first and indicated a microprocessor bus error. If the error is no longer present, then the servo or personality module is the cause of the error.
	a. Set rotary switch to 08 and push ENTER switch. LEDs momentarily display 08 (oooo•ooo). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
09	Microprocessor and personality module bus test
	Purpose: this test checks the data bus between the personality and microprocessor modules. The test is run with the servo module disconnected. The microprocessor module bus test should be successfully completed before running this test.
	a. Set rotary switch to 09 and push ENTER switch. LEDs momentarily display 09 (ooooo•oo•). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
0A	Microprocessor and servo module bus test
	Purpose: this test checks the data bus between the servo and the microprocessor modules. This test is run with the personality module disconnected. The microprocessor module bus test should be successfully completed before running this test.
	a. Set rotary switch to 0A and push ENTER switch. LEDs momentarily display 0A (oooo•o•o). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
0B	Personality module microsequencer test

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	Purpose: this test checks out the ECHO bit path on the personality module and then calls the microprocessor and personality bus test. a. Set rotary switch to 0B and push ENTER switch. LEDs momentarily display 0B (oooo•o••). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
0C	Sector/byte counter test Purpose: to test the sector/ byte counter by means of diagnostic controls. a. Set rotary switch to 0C and push ENTER switch. LEDs momentarily display 0C (oooo••oo). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
0D	**Control panel, serial number and drive number test** Purpose: this test checks out the operator control panel indicators, the drive serial number and the drive unit plug number. a. Set the rotary switch to 0D and push ENTER switch. LEDs momentarily display 0D (oooo••o•). LEDs display control panel switch setting. b. Push front panel RUN switch. LEDs display 01 (ooooooo•). c. Push front panel WRITE PROTect switch. LEDs display steady 02 (oooooo•o). d. Push front panel port A switch. LEDs display steady 04 (ooooo•oo). e. Push front panel Port B switch. LEDs display steady 08 (oooo•ooo). **NOTE** **The front panel indicators should light as each switch is pushed.** f. FAULT indicator should light when FAULT switch is pushed. g. Remove the unit address plug. READY indicator should go out. h. Replace the unit address plug. READY indicator should be lit.

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	i. Push ENTER switch. LEDs display LSB of drive serial numbers.
	j. Push ENTER switch. LEDs display second byte of drive serial number.
	k. Push ENTER switch. LEDs display third byte of drive serial number.
	l. Push ENTER switch. LEDs display drive unit plug number.
	m. Push ENTER switch. LEDs display steady AA (•o•o•o•o).
	n. Push ENTER switch. LEDs blink EC (•••o••oo).
0E	Head-select multiplexer test
	Purpose: this test checks the head-select multiplexer on the microprocessor module.
	a. Set rotary switch to 0E and push ENTER switch. LEDs momentarily display 0E (oooo•••o). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
0F	General purpose counter test
	Purpose: this test checks the operation of the two RAM chip general purpose counters.
	a. Set rotary switch to 0F and push ENTER switch. LEDs momentarily display 0F (oooo••••). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
11	Track counter test
	Purpose: this test checks the operation of the track difference counter.
	a. Set rotary switch to 11 and push ENTER switch. LEDs momentarily display 11 (ooo•ooo•). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
12	Read/write fault force test

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	Purpose: this test checks the read/write safety sense circuits. The spindle must be spinning for this test. a. Set rotary switch to 12 and push ENTER switch. LEDs momentarily display 12 (ooo•oo•o). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
13	Servo position loop test Purpose: this test checks the servo position loop circuitry in a static mode. a. Set rotary switch to 13 and push ENTER switch. LEDs momentarily display 13 (ooo•oo••). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
14	Servo velocity loop test Purpose: this test checks the servo velocity loop with a sawtooth waveform that forces the acceleration thresholds. a. Set rotary switch to 14 and push ENTER switch. LEDs momentarily display 14 (ooo•o•oo). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
15	Servo functional test Purpose: this test checks the operation of the servo system by performing seeks and recalibrations. a. Set rotary switch to 15 and push ENTER switch. LEDs momentarily display 15 (ooo•o•o•). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
16	Random seek test Purpose: this test does 32 repetitions of a random sequence of 32 seeks. a. Set rotary switch to 16 and push ENTER switch. LEDs momentarily display 16 (ooo•o•••o). LEDs display steady E7 (•••oo•••). LEDs display steady AA (•o•o•o•o).

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
17	Seek-seek test with input parameters Purpose: this test does 32 repetitive seeks between cylinders that are entered at the beginning of the test. a. Set rotary switch to 17 and push ENTER switch. LEDs momentarily display 17 (ooo•o•••). LEDs blink 01 (ooooooo•). b. Set rotary switch to low byte of starting cylinder and push ENTER switch. LEDs blink 02 (ooooooo•). c. Set rotary switch to high byte of starting cylinder and push ENTER switch. LEDs blink 03 (ooooooo•). d. Set rotary switch to low byte of ending cylinder and push ENTER switch. LEDs blink 04 (ooooo•oo). e. Set rotary switch to high byte of ending cylinder and push ENTER switch. LEDs display steady E7 (•••oo•••). LEDs display steady AA (•o•o•o•o). f. Push ENTER switch. LEDs blink EC (•••o••oo).
18	Seek-seek test with fixed parameters Purpose: this test does 32 repetitive seeks between cylinders 0 and 560. a. Set rotary switch to 18 and push ENTER switch. LEDs momentarily display 18 (ooo••ooo). LEDs display steady E7 (•••oo•••). LEDs display steady AA (•o•o•o•o). b. PUSH ENTER SWITCH. LEDS BLINK EC (•••O••OO).
19	Incremental seek test with input parameters Purpose: this test does an incremental seek between cylinders entered at the beginning of the test. a. Set rotary switch to 19 and push ENTER switch. LEDs momentarily display 19 (ooo••oo•). b. Set rotary switch to low byte of starting cylinder and push ENTER switch. LEDs blink 02 (ooooooo•).

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	c. Set rotary switch to high byte of starting cylinder and push ENTER switch. LEDs blink 03 (oooooo••).
	d. Set rotary switch to low byte of ending cylinder and push ENTER switch. LEDs blink 04 (ooooo•oo).
	e. Set rotary switch to high byte of ending cylinder and push ENTER switch. LEDs display steady E7 (•••oo•••). LEDs display steady AA (•o•o•o•o).
	f. Push ENTER switch. LEDs blink EC (•••o••oo).
1A	Incremental seek with fixed parameters
	Purpose: this test does an incrementing seek between cylinder 0 and 560.
	a. Set rotary switch to 1A and push ENTER switch. LEDs momentarily display 1A (ooo••o•o). LEDs display steady E7 (•••oo•••). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
1B	Read-only test
	Purpose: this test reads with each read/write head using the prerecorded read-only cylinder.
	a. Set rotary switch to 1B and push ENTER switch. LEDs momentarily display 1B (ooo••o••). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
1C	Write/read test
	Purpose: this test does a write followed by a read with each write/read head on the two write/read cylinders in the guard band.
	a. Set rotary switch to 1C and push ENTER switch. LEDs momentarily display 1C (ooo•••oo). LEDs display steady AA (•o•o•o•o).
	b. Push ENTER switch. LEDs blink EC (•••o••oo).
1D	Go on-line
	Purpose: to return the drive to functional on-line mode.

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	a. Set rotary switch to 1D and push ENTER switch. LEDs momentarily display 1D (ooo●●●o●).
	b. Set rotary switch to 00 (oooooooo). LEDs display steady 00 (oooooooo). Drive is back on-line when READY light comes on.
1E	Spindle control utility
	Purpose: this utility allows the spindle to be spun up or down from the drive diagnostic mode.
	a. Set rotary switch to 1E and push ENTER switch. LEDs momentarily display 1E (ooo●●●●o). LEDs display steady E7 (●●●oo●●●).
	b. Push the front panel RUN switch. The spindle will spin up. LEDs display steady AA (●o●o●o●o).
	To spin down from the diagnostic mode, call the spindle control utility again with the 1E test-select code and release the RUN switch.
	c. Push ENTER switch. LEDs blink EC (●●●o●●oo).
1F	Head-select and seek utility
	Purpose: this utility enables the field engineer to seek to a specific head and cylinder address.
	a. Set rotary switch to 1F and push ENTER switch. LEDs momentarily display 1F (ooo●●●●●). LEDs blink 01 (ooooooo●).
	b. Set rotary switch to head number and push ENTER switch. LEDs blink 02 (oooooo●o).
	c. Set rotary switch to cylinder low byte and push ENTER switch. LEDs blink 03 (oooooo●●).
	d. Set rotary switch to cylinder high byte and push ENTER switch. LEDs display E7 (●●●oo●●●).
	e. Set rotary switch to DD and push ENTER switch to terminate this test. LEDs blink EC (●●●o●●oo).
20	Maintenance controls and indicators test

* ● = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	Purpose: this test checks the ability of the firmware to read the rotary switches and control LEDs. a. Set rotary switch to 20 and push ENTER switch. LEDs momentarily display 20 (oo•ooooo). LEDs blinks FF (••••••••). b. Set rotary switch to FF and push ENTER switch. LEDs display steady FF (••••••••). c. Set rotary switch to 00 and push ENTER switch. LEDs display steady 00 (oooooooo). d. Push ENTER switch. LEDs display steady AA (•o•o•o•o). e. Push ENTER switch. LEDs blink EC (•••o••oo).
21	Read-only cylinder formatter utility Purpose: this utility enables the field service engineer to reformat the prerecorded readonly cylinder in the guard band. As a safety precaution, pin 13 of E11 on the microprocessor or module must first be grounded to the center pin of the E/R switch. Figure 4-4 shows the location of E11. a. Set rotary switch to 21 and push ENTER switch. LEDs momentarily display 21 (oo•oooo•). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
22	Logic test Purpose: this test runs the logic tests called by the following test-select codes: 07 = Personality module microsequencer test 0C = Sector/byte counter test 0E = Head-select multiplexer test 0F = General purpose counter test a. Set rotary switch to 22 and push ENTER switch. LEDs momentarily display 22 (oo•ooo•o). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
23	Servo test Purpose: this test runs the servo tests called by the following test-select codes; 11 and 13-16. a. Set rotary switch to 23 and push ENTER switch. LEDs momentarily display 23 (oo•ooo••). LEDs display steady E7 (•••oo•••). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
24	Read/write test Purpose: this test runs the read-only and the write/head test. a. Set rotary switch to 24 and push ENTER switch. LEDs momentarily display 24 (oo•oo•oo). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
25	Entire unit test Purpose: this test runs the diagnostic programs called by the following test-select codes; 07, 0C, 0E, 0F, 11, 12, 13, 14, 15, 16, 1B, 1C. a. Set rotary switch to 25 and push ENTER switch. LEDs momentarily display 25. LEDs display steady E7 (•••oo•••). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
26	Servo velocity adjustment utility Purpose: this utility enables the Field Service engineer to adjust the servo velocity potentiometer. Before doing this adjustment, the random seek test should be run for 20 minutes to stabilize the temperature of the servo system. Adjust the servo for a 1% or better servo speed as shown in Figure 4-5. a. Set rotary switch to 26 and push ENTER switch. LEDs momentarily display 26 (oo•oo••o). LEDs display servo speed pattern. Adjust the servo speed (see chapter on adjustments). b. Set rotary switch to DD and push ENTER switch. If LEDs display AA (•o•o•o•o), push ENTER switch. LEDs blink EC (•••o••oo).
27	Static servo test

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	Purpose: this test runs the static servo tests called up by test-select codes 11, 13, and 14. a. Set rotary switch to 27 and push ENTER switch. LEDs momentarily display 27 (oo•oo•••). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
28	Personality board test Purpose: this test checks the operation of status registers and the communication path on the personality module. This test cannot be run while the disk is spinning, and the loop-back plugs must be inserted as shown in Figure 4-6 before the test is run. a. Set rotary switch to 28 and push ENTER switch. LEDs momentarily display 28 (oo•o•ooo). LEDs display steady AA (•o•o•o•o). b. Push ENTER switch. LEDs blink EC (•••o••oo).
CF	Loop mode set utility Purpose: this utility enables the Field Service engineer to set up the diagnostic firmware to do one of the following. • Loop forever on the test – 0F • Loop forever but halt on error – 4F • Halt on error or at end of test – FF Once a loop mode is selected, the firmware remains in this state until the utility is recalled and an alternate mode is selected. a. Set rotary switch to CF and push ENTER switch. LEDs momentarily display CF (••oo••••). LEDs blink 01 (ooooooo•). b. Set rotary switch to loop mode desired and push ENTER switch. LEDs display steady AA (•o•o•o•o). c. Push ENTER switch. LEDs blink EC.
FF	Go off-line Purpose: to place drive into diagnostic mode. a. Set rotary switch to FF and push ENTER switch. LEDs blink FF (••••••••).

* • = LED on, o = LED off

Table 4-3 RA80 Utility and Diagnostic Test Procedures (Cont)*

Test-Select Code	Operating Procedures
	b. Push ENTER switch. LEDs blink steady FF (••••••••).
	c. Set rotary switch to 00 and push ENTER switch. LEDs display steady 00 (oooooooo).
	d. Push ENTER switch. LEDs blink EC (•••o••oo).
	NOTE
	Drive is now in diagnostic mode.

* • = LED on, o = LED off

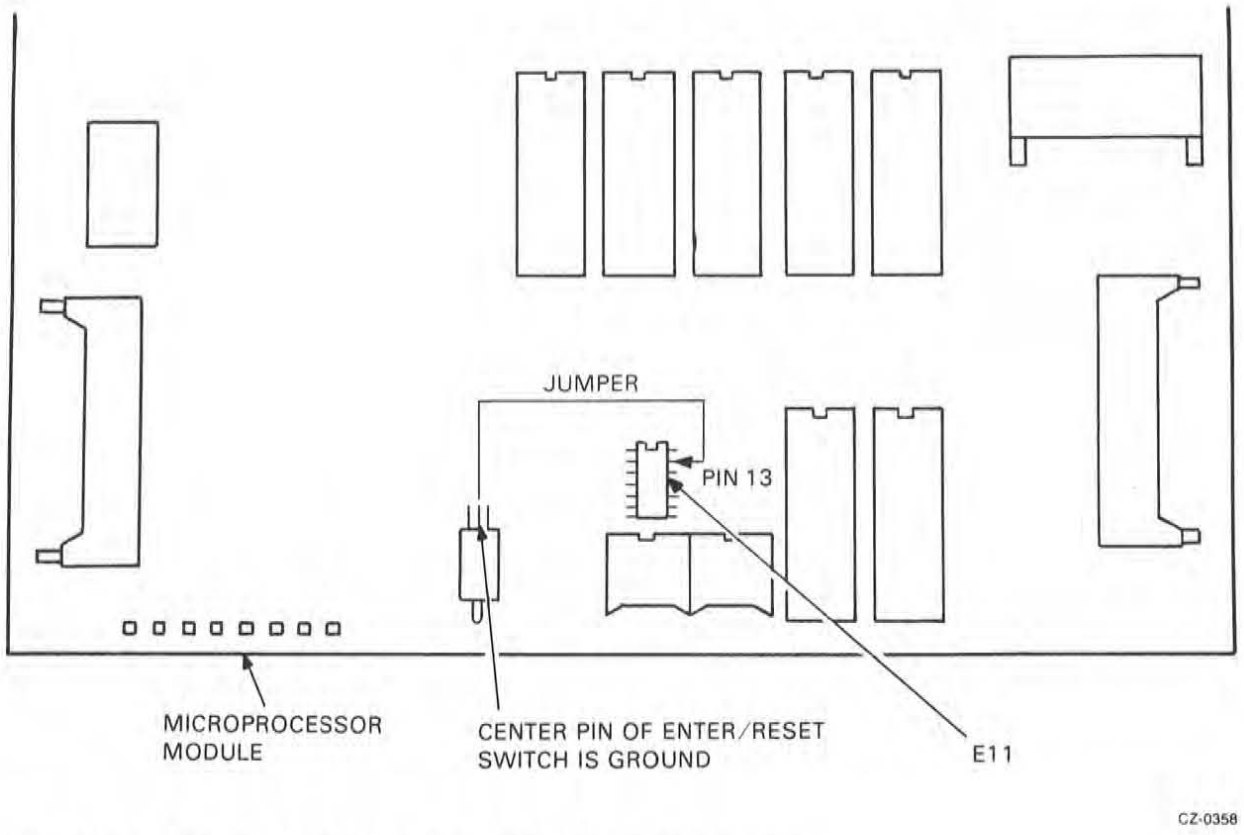


Figure 4-4 Ground Jumper for Read-Only Cylinder Formatter

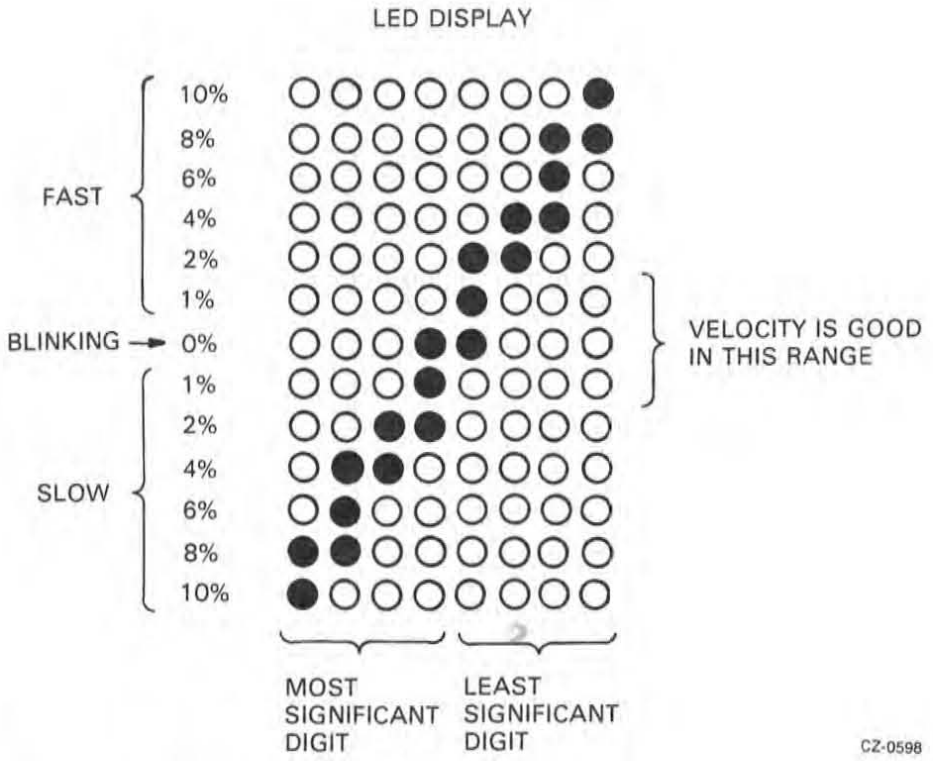
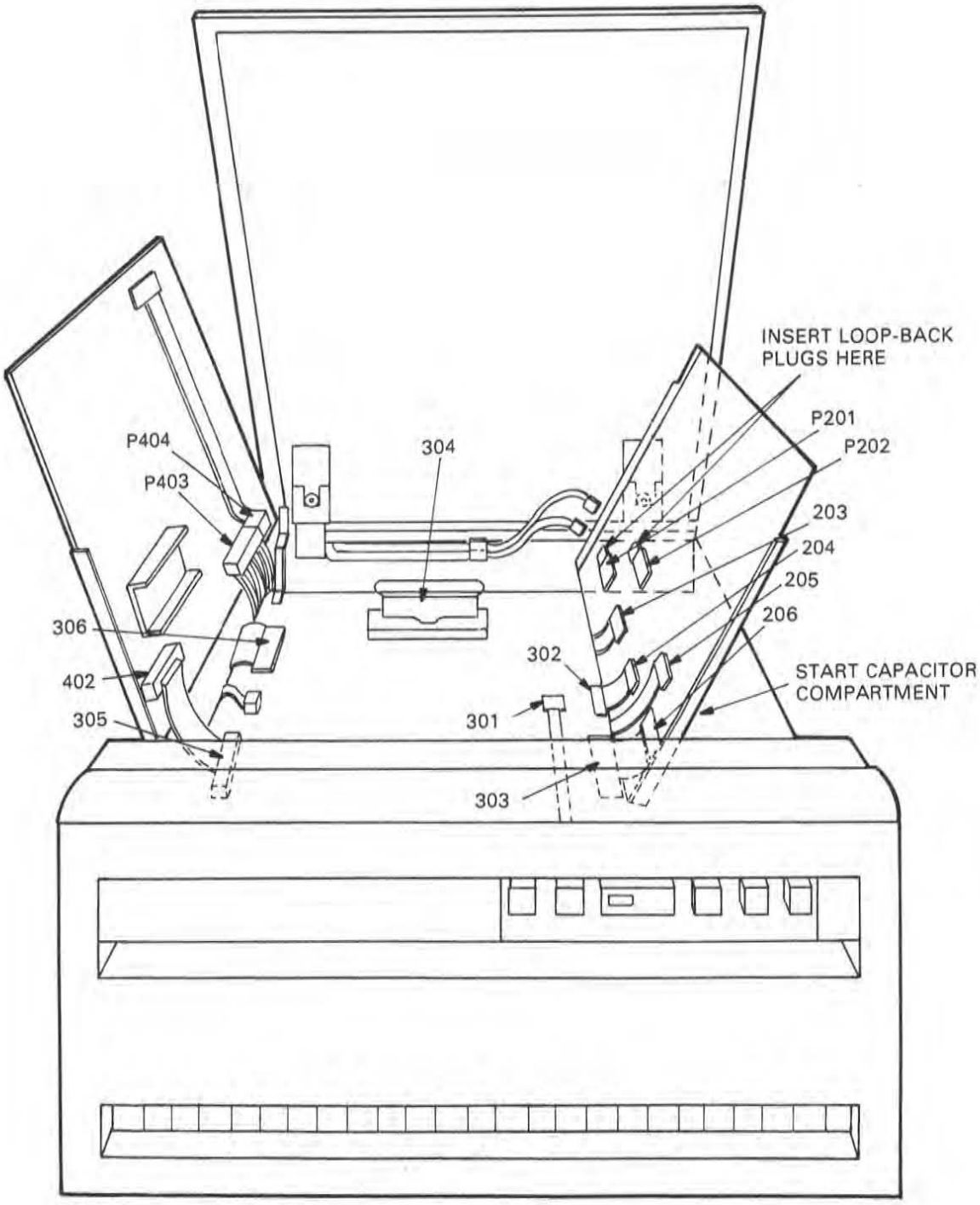


Figure 4-5 Velocity Adjustment LED Pattern



NOTE

1. LOOP-BACK PLUGS MUST BE INSERTED IN P201 AND P202 BEFORE RUNNING THE PERSONALITY BOARD TEST 28. THE LOOP-BACK PLUGS SHOULD BE IN THE MOTOR START CAPACITOR COMPARTMENT LOCATED TO THE RIGHT OF THE LOGIC CHASSIS.

CZ-0615

Figure 4-6 Personality Board Loop Back Plugs

CHAPTER 5
FAULT ISOLATION

5.1 INTRODUCTION

This chapter describes the RA80 fault isolation procedure. It begins with a description of the subsystem error messages. Next, the LED error code troubleshooting charts are presented. Additional troubleshooting tips are given at the end of the chapter.

5.2 HDA FORMATTING PROCEDURE

Replacement HDAs have RM80 formatting and must be reformatted for the RA80 Disk Drive. To reformat the maintenance cylinders, run the RA80 formatter utility (test-select code 21). Refer to Table 4-3. To reformat customer data areas, run the host-resident formatter program.

5.3 SUBSYSTEM ERROR MESSAGE INFORMATION

Subsystem error messages printed out in the error log or the subsystem diagnostics contain specific drive error information. The RA80 error messages show up in bytes 9-15 of the status error message. A sample printout of the status message is shown below.

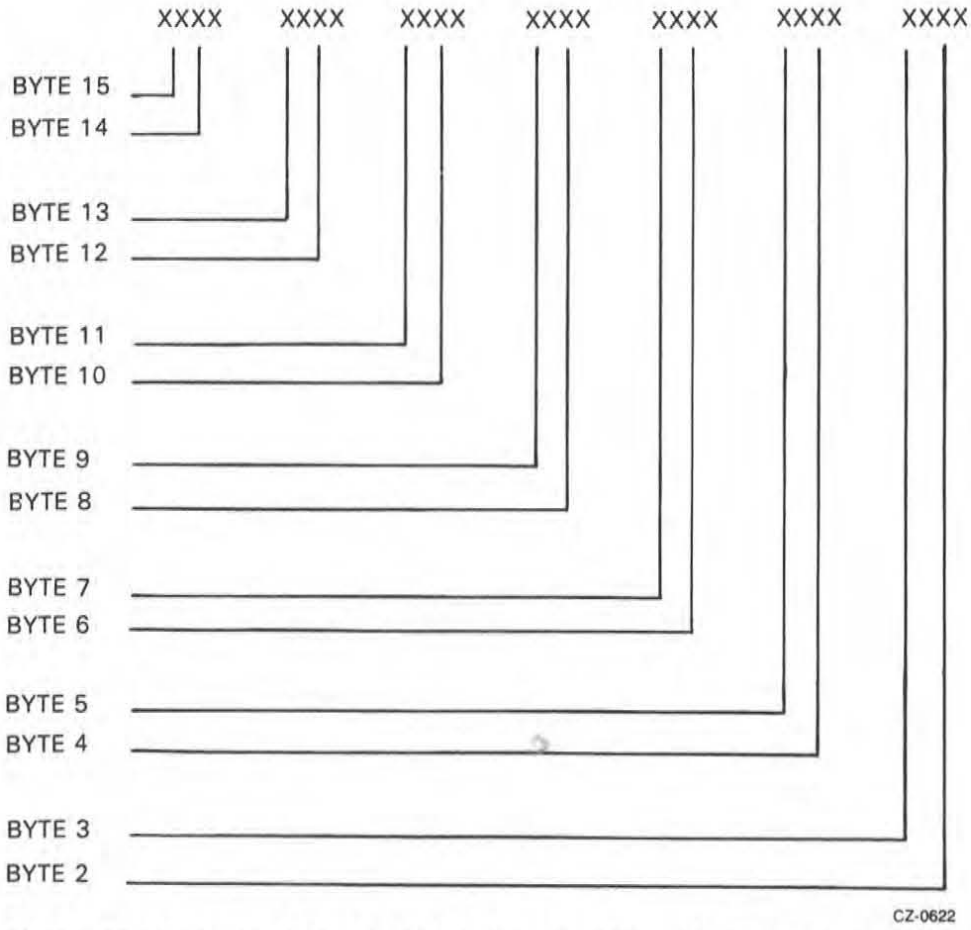
Error Message Sample Printout:

UDAHRD HRD ERR 00044 ON UNIT 00 TST 004 SUB 000 PC: 021044
DISK EXERCISER DM PC:5110 UDA AT 172150 DRIVE 032 RUNTIME 0:00:23
ENTIRE RCT AREA SEARCHED, COULD NOT FIND RBN TO REPLACE
LBN WITH HEADER COMPARE ERROR
SEARCHING FOR LBN: 900

UDAHRD SFT ERR 00006 ON UNIT 00 TST 004 SUB 000 PC: 021044
DISK EXERCISER DM PC:5324 UDA AT 172150 DRIVE 032 RUNTIME 0:00:37
TIMEOUT OF DRIVE DURING WRITE ATTEMPT
WRITE ATTEMPT RETRIES: 0
L/DBN NUMBER 5252
ACTUAL L/R/DBN 0
TRK 1 CRP 0 CYL 6
ORIGIN OF LAST SEEK WAS CYL 5 GROUP 1
REAL TIME DRIVE STATE 8001
STATUS: 0001 1100 0000 0A00 0000 0613 1020

Refer to Figure 5-1 to locate bytes 9-15 of the status message. Refer to Figure 5-2 to see contents of each status byte.

- Byte 9 of the status message contains the following examples.
  - Seek CMD = hex code 0A
  - Recal CMD = hex code 8E



Note: The content of these status bytes is shown in Figure 5-2.

Figure 5-1 Status Message Interpretation

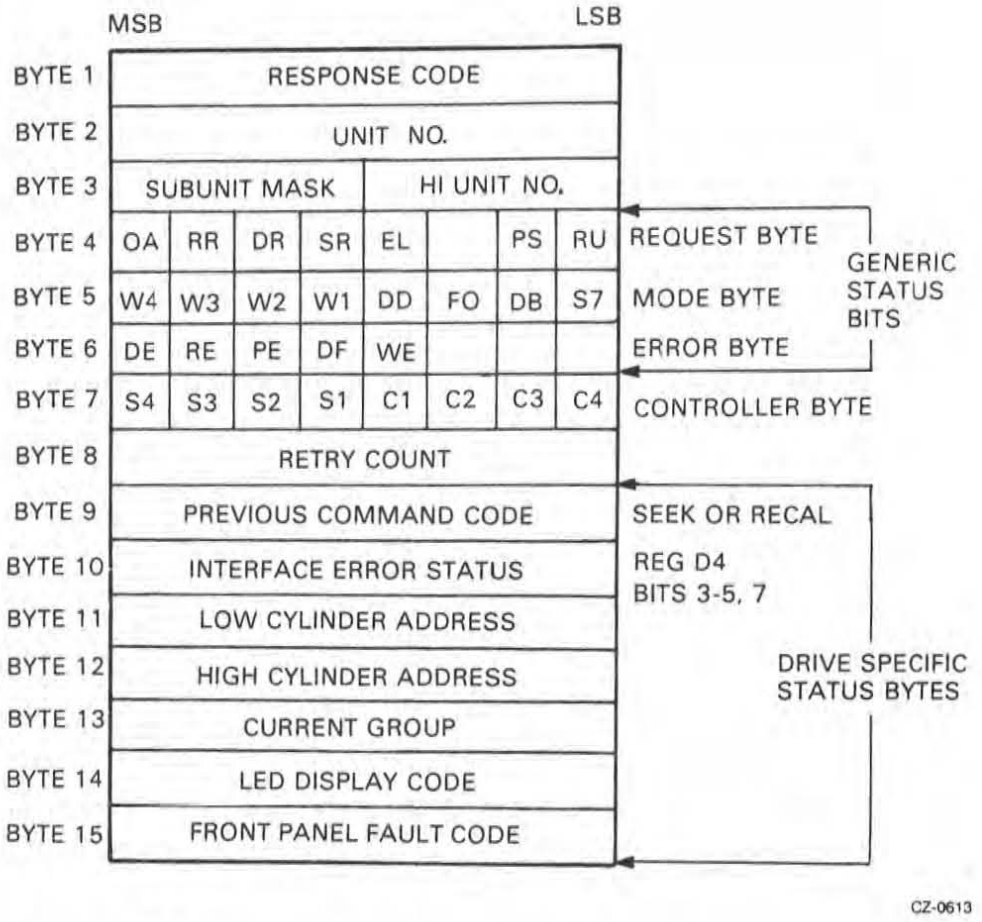


Figure 5-2 Response to Get Status Command

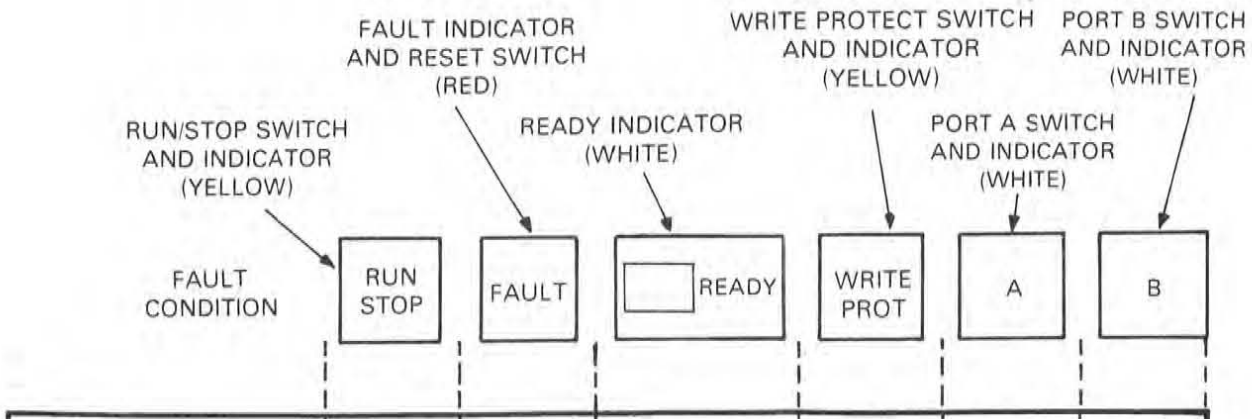
- Byte 10 contains bits 3-5 and 7 of the drive register D4.
  - Bit 3 (OVERRUN ERROR) – this bit is set when either a read or a write extends past the sector or index pulse following the sector on which the operation is taking place.
  - Bit 4 (PARITY ERROR FOUND) – this bit is set when a parity error is discovered on the real time controller status line during a real time command.
  - Bit 5 (CONTROL PULSE ERROR) – this bit is set during a real time command if two or more pulses of the same polarity are detected on the real time controller status line of the SI.
  - Bit 7 (DATA PULSE ERROR) – this bit is set during a real time command if two or more pulses of the same polarity are detected on the write command data line of the SI.
- Bytes 11 and 12 contain head location information (the last group cylinder seeked).
- Byte 13 contains present head location information. (A group is a logical point that is accessible from the present head location in less time than one disk rotation.)
- Byte 14 contains the hexadecimal LED error codes displayed in the RA80 Disk Drive. Refer to Paragraph 5.4.
- Byte 15 contains the hexadecimal code for the front panel fault indicators. (Refer to Figure 5-3.) The hexadecimal codes are listed in Table 5-1.

5.4 LED ERROR CODES TROUBLESHOOTING CHARTS

Table 5-2 provides a list of the LED error codes for the drive and the errors they represent. The numbers in the right-hand column correspond to reference numbers in Table 5-3 and denote the most likely causes of the error.

NOTE

The part numbers for the FRUs in Table 5-3 are accurate up to the time of this manual edition. When ordering FRUs, always refer to the latest RA80 Illustrated Parts Breakdown Manual for the most up-to-date part numbers.



SPIN-UP		ON				ON
R/W DIAGNOSTIC	*	ON			ON	
READ/WRITE UNSAFE	*	ON			ON	ON
SPINDLE MOTOR INTERLOCK	*	ON		ON	ON	
SPINDLE MOTOR SPEED	*	ON		ON	ON	ON
DRIVE DISABLED BY "DD" BIT	*	ON	ON			ON
HDA OR SERVO BOARD OVERTEMPERATURE	*	ON	ON		ON	
MICROCODE	*	ON	ON		ON	ON
SERVO DIAGNOSTIC	*	ON	ON	ON		ON
INITIAL RECAL	*	ON	ON	ON	ON	
MICROPROCESSOR HARDCORE TEST	ON	ON	ON	ON	ON	ON

*THE INDICATOR STATE WILL BE THE SAME AS IT WAS BEFORE THE FAULT SWITCH WAS PUSHED

CZ-0428

Figure 5-3 Fault Identification Codes

Table 5-1 Hex Representation of Front Panel Fault Codes

Front Panel Faults	Hex Codes
Spin-up	11
R/W diagnostic	12
Read-/write-unsafe	13
Spindle motor interlock	16
Spindle motor speed	17
Drive disabled by DD bit	19
HDA or servo board overtemp	1A
Microcode	1B
Servo diagnostic	1D
Initial recal	1E
Microprocessor hardcore test	3F

Table 5-2 LED Error Codes and Possible Cause(s)

Error Code	Error	Possible Cause(s)*
01	Spindle motor timeout	7, 16, 21, 20, 35, 3, 1, 25, 28
02	Spin-up too slow	5, 25, 7, 16, 21, 23
03	Spindle not accelerating	5, 25, 7, 16, 21, 23
04	Spin-up too long	5, 25, 7, 16, 21, 23
05	Cannot spin up because sip set grant not set	†
07	Level 2 message frame sequence	†
08	Level 2 message checksum	†
09	SI message framing	†
0A	Wrong level 2 op code parity	†
0B	Invalid level 2 op code	†
0C	Invalid level 2 command length	†
0D	Attempt to execute command with status byte non-zero	†
0E	Incorrect group select code	†

*See Table 5-3

†This error may not indicate a hardware problem. Try the test again. If failure persists, further troubleshooting may indicate a controller or drive personality module problem. This error should not occur while the drive is off-line.

‡Perform the servo adjustments before replacing the modules.

Table 5-2 LED Error Codes and Possible Cause(s) (Cont)

Error Code	Error Codes	Possible Cause(s) (Cont)
0F	Attempt to write enable drive with switch in the PROT position	†
11‡	Wrong peak entering detent	38, 21, 32, 23
12	Servo active PLO	38, 21, 32, 30, 23, 1
13	No fine track	38, 21, 32, 23, 1
14‡	Servo speed or direction	38, 21, 32, 23, 1
15‡	Seek/recal timeout	56, 38, 21, 32, 39, 23, 1
16‡	Guard band	38, 21, 32, 23, 1
17‡	Track counter underflow	38, 21, 32, 23, 1
1A	Invalid cylinder address	21, 31, 46, 47
1F	Sector overrun	†
20	Controller real time state parity	†
21	Control pulse	†
22	Data pulse	†
23	Spindle motor interlock	57, 21, 15, 39
24	Servo inactive PLO	38, 21, 32, 30, 23, 1
25‡	Servo error set	38, 21, 32, 23, 1
26	Spindle speed	5, 58
27	HDA overtemperature	5, 59, 60, 17, 20, 35, 21
28	Servo module overtemperature	6, 59, 38, 32, 21
29	Invalid error recovery level specified	†
2A	Invalid subunit specified	†
2B	Invalid region (test number) in diagnose command	†
2C	Seek/recal command given when spindle not spinning	†
2D	Invalid command timeout value given	†
2E	Controller flags prohibit spin-up	†
2F	Run command issued with RUN/STOP switch in RUN	†
30	Write current and no write-gate	20, 21, 35, 23
31	Read and write	21, 33, 46, 47
32	Read/write while faulted	21, 43, 20, 32, 35, 34, 23
33	Data separator/encoder	21, 20, 43, 35, 33
34	Write precompensation	21, 43, 31
35	Write and write-unsafe	20, 35, 34, 21, 23
36	Read/write head shorted	20, 35, 34, 21, 23

*See Table 5-3

†This error may not indicate a hardware problem. Try the test again. If failure persists, further troubleshooting may indicate a controller or drive personality module problem. This error should not occur while the drive is off-line.

‡Perform the servo adjustments before replacing the modules.

Table 5-2 LED Error Codes and Possible Cause(s) (Cont)

Error Codes	Error Codes	Possible Cause(s) (Cont)
37	Write gate and no write current	20, 35, 34, 21, 43, 33, 46, 47, 23
38	Read and multichip-select	20, 35, 34, 21, 23
39‡	Write and off-track	37, 21, 31, 29, 23
3A	Write and write-protected	21, 41, 35
40	Invalid read/write region	†
41	Response timed out	†
42	Seek command issued when drive not on-line	†
43	Real time command when read/write ready not set	†
44	Format command when format enable not set	†
45	Invalid head (track) address in real time command	†
46	Read/write safety interrupt with no cause bits set	†
47	TT bit incorrect in disconnect command	†
48	Invalid write memory offset or byte count	†
49	Invalid command during topology mode	†
4A	Drive disabled by DD bit	†
51	Sector/byte counter	21, 37, 31
53	Personality module microsequencer	43, 21, 30
54	Multiplexer head-select	21, 43, 20, 32, 34
57	RAM 1 general purpose counter	21
58	RAM 0 general purpose counter	21
60	Read/write head-select	21, 20, 32, 34, 33, 22
61	Data port preset	43, 21, 30
62	Read-only test overall read	20, 21, 32,34, 33, 22
63	Read-only test partial read	20, 33, 22, 34, 21, 32
64	Read/write test guard band	37, 21, 31, 22
65	Sector timeout	37, 21, 31, 43, 32
66	Read-only test read and no enable	53, 37, 22
67	Write test not executable	61
6A	Write/read test overall read	20, 21, 43, 32, 34, 33, 22
6B	Write/read test partial read	33, 20, 34, 42, 32
70	Read/write control-select	21, 43, 30

*See Table 5-3

†This error may not indicate a hardware problem. Try the test again. If failure persists, further troubleshooting may indicate a controller or drive personality module problem. This error should not occur while the drive is off-line.

‡Perform the servo adjustments before replacing the modules.

Table 5-2 LED Error Codes and Possible Cause(s) (Cont)

Error Code	Error Codes	Possible Cause(s) (Cont)
71	Utility head-select	21
72	Microprocessor and personality board communication error 1	77, 43, 21, 31
73	Microprocessor and personality board communication error 2	77, 43, 21, 31
74	Initial board status	77, 43, 21, 31
75	Control clock error detect circuit	77, 43, 21, 31
76	Data clock error detect circuit	77, 43, 21, 31
77	Port A select flop	77, 43, 21, 31
78	Port A data xmit/rcvrs	77, 43, 21, 31
79	Port A control xmit/rcvrs	77, 43, 21, 31
7A	Port B select flop	77, 43, 21, 31
7B	PCB test cannot be done while motor is spinning	78
7C	Too slow seek	62
7D	Bad seek count overflow	37, 42, 31, 22, 1
7E	Too fast seek	63
7F	Spindle not spinning	64
80	ROM set	21
81	Command available reset	77, 43, 21, 31
82	Frame code reset and response	77, 43, 21, 31
83	Init receivability from port B	77, 43, 21, 31
84	CLRINI not clearing pending init	77, 43, 21, 31
85	RAM 0	21
86	RAM 1	21
87	ROM 0 checksum	21
8A	Module interlock	65
8B	Discrete port-enable	21
8F	ROM 1 checksum	21
90	Port B data xmit/rcvr	77, 43, 21, 31
91	Port B control xmit/rcvr	77, 43, 21, 31
92	Port A wraparound reselect	77, 43, 21, 31
93	Response serializer	77, 43, 21, 31

*See Table 5-3

†This error may not indicate a hardware problem. Try the test again. If failure persists, further troubleshooting may indicate a controller or drive personality module problem. This error should not occur while the drive is off-line.

‡Perform the servo adjustments before replacing the modules.

Table 5 ED Error Codes and Possible Cause(s) (Cont)

Error Codes	r Codes	Possible Cause(s) (Cont)
94	Frame around receive	77, 43, 21, 31
95	Loop around decode	77, 43, 21, 31
96	Data byte receive	77, 43, 21, 31
97	ROM 2 checksum	21
9F	ROM 3 checksum	21
A0	Read and write safety	21, 43, 20, 31, 33, 35, 23
A3	Forced read and write	21, 43, 34, 31
A4	Forced write current and no write gate	21, 43, 31
A5	Forced write gate and no write current	21, 43, 31
A6	Forced separator/encoder	21
A7	ROM 4 checksum	21
A9‡	Servo caused read/write forced fault	38, 43, 20, 21, 32
AB‡	Outer guard band seek	38, 21, 32, 23, 1
AF	ROM 5 checksum	21
B0	Three-module microprocessor bus	21
B1	Three-module personality bus	43, 21, 31
B2	Three-module servo bus	38, 21, 32
B3	Microprocessor module bus	21
B4	Personality module bus	43, 21, 31
B5	Servo module bus	38, 21, 31
C2	Fine-track status	38, 21, 31
C3	Fine-track overrange	38, 21, 31
C4	Fine-track underrange	38, 21, 31
C5	Off-track status	38, 21, 31
C6	Off-track overrange	38, 21, 31
C7	Off-track underrange	38, 21, 31
CB	Acceleration status	38, 21, 32
CD	Track counter	38, 31
CE‡	Funct. check PLO-OK not false after disable	38, 21, 32, 23, 1
CF‡	Funct. check PLO-OK not true after enable	38, 21, 32, 23, 1

*See Table 5-3
†This error may not indicate a hardware problem. Try the test again. If failure persists, further troubleshooting may indicate a controller or drive personality module problem. This error should not occur while the drive is off-line.
‡Perform the servo adjustments before replacing the modules.

Table 5-2 LED Error Codes and Possible Cause(s) (Cont)

Error Codes	Error Codes	Possible Cause(s) (Cont)
D0‡	Recalibration	38, 21, 32, 23, 1
D1‡	Outer guard band status	38, 21, 32, 23, 1
D2‡	Inner guard band status	38, 21, 32, 23, 1
D3‡	Seek into outer guard band	38, 21, 32, 23, 1
D4‡	Outer guard band status not true	38, 21, 32, 23, 1
D5‡	Recal from outer guard band	38, 21, 32, 23, 1
D6‡	Two-track seek to track 560	38, 21, 32, 23, 1
D7‡	Seek into inner guard band	38, 21, 32, 23, 1
D8‡	Inner guard band status not true	38, 21, 32, 23, 1
D9‡	Recal from inner guard band	38, 21, 32, 23, 1
E0‡	Random seek	38, 21, 31, 22, 1
E1	Seek check	66, 67
EA	Cannot run test, drive-faulted	68
EE	Entry	69, 70
EF	F.E. entered invalid cylinder address	71
FE	Rotary switch	21

*See Table 5-3
†This error may not indicate a hardware problem. Try the test again. If failure persists, further troubleshooting may indicate a controller or drive personality module problem. This error should not occur while the drive is off-line.
‡Perform the servo adjustments before replacing the modules.

Table 5-3 FRU/Service Reference Table

Ref. No.	Description	Part No./ Reference No.
1	Power supply, 115 V, 60 Hz	H766-C
2	Power supply, 220 V, 50 Hz	H766-D
3	Motor start cap., 115 V, 60 Hz	10-16924-00
4	Motor start cap., 220 V, 50 Hz	10-17217-00
5	Fan and bracket assembly, 117 CFM, ball bearing	70-16745-00
6	Fan, 117 CFM, ball bearing (front bezel)	12-10719-03

*SV = RA80 Disk Drive Service Manual
†UG = RA80 Disk Drive User Guide

Table 5-3 FRU/Service Reference Table (Cont)

Ref. No.	Description	Part No./ Reference No.
7	Belt, 60 Hz	12-12635-03
8	Belt, 50 Hz	12-12635-04
9	Switch cap (RUN/STOP)	12-12714-00
10	Switch cap (WRITE PROTect)	12-12714-01
11	Switch cap (FAULT)	12-12714-02
12	Switch cap (A port)	12-12714-03
13	Switch cap (B port)	12-12714-04
14	Wedge lamp, 6.3 V	12-12716-00
15	Microswitch (IPSA) (belt tension)	12-14011-00
16	Optical switch (HDA speed sensor)	12-16817-00
17	HDA thermal switch	12-16870-00
18	Gas spring assembly	12-17072-00
19	Encoder switch cap	12-18199-00
20	Read/write module	54-13596-00
21	Microprocessor and ROM assembly	54-15284-01
22	Brush ground assembly	70-16215-00
23	Head disk assembly	70-16225-00
24	Wing pivot assembly	70-16230-00
25	Motor/brake assembly, 115 V, 60 Hz	70-16723-00
26	Motor/brake assembly, 220 V, 50 Hz	70-16723-01
27	Motor actuator assembly	70-16724-00
28	Logic dc power cable assembly	70-16732-00
29	Logic ac power harness assembly	70-16733-00
30	Servo preamp cable assembly	70-16735-00
31	Data cable, 40 cond. (personality)	70-16737-00
32	Data cable, 40 cond. (servo)	70-16737-01
33	Data cable, 20 cond. (personality)	70-16738-00
34	Data cable, 20 cond. (read/write)	70-16738-01
35	Read/write cable, 50 cond.	70-16739-00
36	Operator control panel cable assembly	70-16740-00
37	Shock mount assembly	70-16742-00
38	Servo module/stiffener	70-16976-00
39	Belt tension switch harness assembly	70-16980-00
40	Line cord assembly, 115 V, 60 Hz	70-18345-00
41	Line cord assembly, 220 V, 50 Hz	70-18345-01

*SV = RA80 Disk Drive Service Manual
UG = RA80 Disk Drive User Guide

Table 5-3 FRU/Service Reference Table (Cont)

Ref. No.	Description	Part No./ Reference No.
42	Control panel assembly	70-18324-00
43	Personality module/stiffener	70-18327-00
44	Cabinet I/O bulkhead assembly	70-18340-01
45	Sequence terminator	70-09490-00
46	External SI I/O cable, (variable lengths)	BC26V-XX
47	Internal SI I/O cable	70-18487-6B
48	Motor tension spring	74-22440-00
49	Foam air filter (front bezel)	74-22816-00
50	Microprocessor board (UDA50)	M-7161
51	SI interface (UDA50)	M-7162
52	Belt tension adjustment	*SV Chap 3
53	Servo adjustments	*SV Chap 3
54	Drive I/O cables not connected properly	†UG Chap 2
55	Drive sequence cable not connected properly or defective	†UG Chap 2
56	HDA positioner lock in wrong position	*SV Chap 2
57	Belt tension lever in wrong position	*SV Chap 2
58	Spindle motor overheated	*SV Chap 5
59	Ambient room temperature too high	Decrease temp.
60	Speed transducer cable not connected	*SV Chap 2
61	Read-only test failed	*SV Chap 5
62	Positioner motor going to slow	*SV Chap 4
63	Positioner motor going too fast	*SV Chap 4
64	Spindle not spinning	*SV Chap 5
65	Cables not connected properly	†UG Chap 2
66	Drive-unsafe condition	*SV Chap 4
67	Recalibration failure	*SV Chap 5
68	Test inhibited by drive error	*SV Chap 4
69	Starting and ending address same as for seek test	Enter correct address
70	Jumper for read-only cylinder formatter utility missing	*SV Chap 4

*SV = RA80 Disk Drive Service Manual
†UG = RA80 Disk Drive User Guide

Table 5-3 FRU/Service Reference Table (Cont)

Ref. No.	Description	Part No./ Reference No.
71	Cylinder address greater than 560 has been entered	Enter correct address
72	Wrong procedure used or wrong value entered	*SV Chap 4
73	Software problem	Check prog.
74	Cable problems	†UG Chap 2
75	Operator error	*SV Chap 4
76	Low ac power input	†UG Chap 1
77	Loop-back plugs need to be in	*SV Chap 4
78	Spin motor down	*SV Chap 4
	Drive sequence cable assembly (20 in)	70-18328-00
	Internal drive sequence cable (6 ft)	70-18457-06
	Drive sequence cable (8 ft)	70-18458-08
	External drive sequence cable (variable lengths)	70-18458-XX

*SV = RA80 Disk Drive Service Manual
†UG = RA80 Disk Drive User Guide

5.5 POWER SUPPLY FAULT ISOLATION

Power supply failure symptoms are listed in Table 5-4. Power supply voltages can be measured on the circuit modules at the connector pins indicated in Figure 5-4, and the tolerances for these measurements are given in Table 5-5.

5.6 TROUBLESHOOTING TIPS

The following text describes some general troubleshooting tips that may be useful when performing RA80 drive fault isolation.

Table 5-4 Power Supply Failure Indications

Voltage Checks	Action
+5 volt indicators	Check that the operator control panel indicators and the internal LED display flash on momentarily when power is applied to the drive. The flash indicates that +5 volts is present.
WRITE PROTect switch	Check that +5 volts is present at the operator control panel WRITE PROTect switch. The WRITE PROTect switch should light when pushed if +5 volts is present and the drive is on-line.
±12 volt LEDs	Check that the two 12 volt LEDs on the servo module (next to the heat-sink) are on. When lit, they indicate that both +12 and -12 volts are present.
FAULT indicator off, All other operator control panel indicators on	Check to see if only the FAULT indicator on the operator control panel is off. This condition occurs only when the DC LOW signal is asserted. Possible causes could be the power supply, microprocessor module, or servo module.

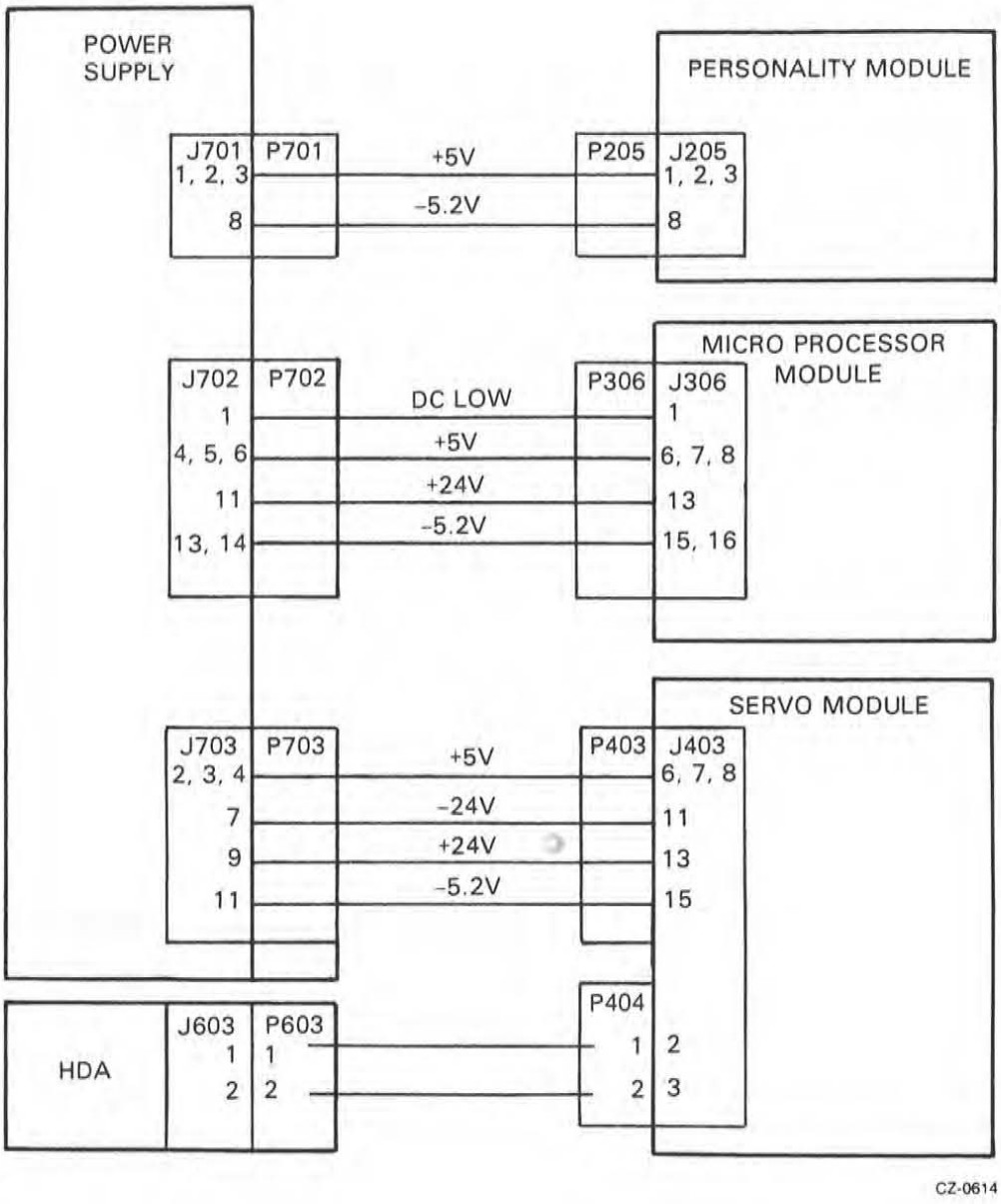


Figure 5-4 Voltage Test Points

Table 5-5 DC Voltage Tolerances

DC Voltages	Acceptable Tolerances	Test Points (See Figure 5-4)
+5	±0.2 volts	Pins 1, 2, or 3 of J205
-5.2	±0.2 volts	Pins 15 or 16 of J306
+24	+3, -1 volts	Pin 13 of J403
-24	+1, -3 volts	Pin 11 of J403
DC Low		Pin 1 of J306

5.6.1 Check Firmware Revision and ROM Set Numbers

The firmware revision and ROM set numbers are in the last few bytes of each ROM. The easiest way to examine these numbers is by using the memory-examine down utility, test-select code "06". Enter the address of the last byte of the ROM to be examined, then push the ENTER switch to examine each byte location starting with the last byte in the ROM. Refer to Table 5-6 for the last address of each ROM. Figure 5-5 shows what is contained in the last eight bytes of each ROM.

Table 5-6 Last Byte Address of Each ROM

ROM	Address of Last Byte
0	07FF
1	0FFF
2	17FF
3	1FFF
4	27FF
5	2FFF

REV HIGH BYTE	REV LOW BYTE	ROM SET HIGH BYTE	ROM SET LOW BYTE	ZEROS (UNUSED)	ONES COLUMN CHECK	ZEROS COLUMN CHECK	CHECK SUM
XXX8	XXX9	XXXA	XXXB	XXXC	XXXD	XXXE	XXXF

CZ-0366

Figure 5-5 Last Eight Bytes of a ROM

5.6.2 Testing the Write Protect Function

The write protect function in the RA80 Disk Drive may be tested while the drive is in the functional mode with the following procedure.

1. Spin down the disks by releasing the RUN/STOP switch.
2. Push in the WRITE PROTect switch.
3. Spin up the disk by pushing in the RUN/STOP switch. If the write protect function is working, the FAULT indicator lights. The internal LEDs on the microprocessor module will display an error code of 6A.
4. Push the FAULT switch to enter the fault display mode. The operator control panel displays a R/W diagnostic fault code. Both the FAULT and port A indicators should be on.
5. Push the FAULT switch again to clear this fault condition.
6. Release the WRITE PROTect switch.

5.6.3 Spindle Motor Thermal Timeouts

The RA80 Engineering Specification calls for a three-minute waiting period between successive start-up cycles of the spindle motor. This waiting period is required to prevent the spindle motor from overheating and setting an internal thermal switch in the motor. A spindle motor thermal timeout may be caused by one of the following.

- Frequent spindle motor start-up
- Loss of cooling due to a fan failure

If the spindle motor thermal switch is set, it results in the following symptoms.

- LED error codes of 01, 02, 03, 04, or 26
- Spin-up fault on the operator control panel

To recover from a spindle motor thermal timeout, check the fans first. If the fans are operating, let the motor cool off for 10 to 15 minutes while power is applied to the drive.

APPENDIX A
HEXADECIMAL TO BINARY CONVERSION

The LED display on the microprocessor module consists of eight LEDs in a row which are read as two hexadecimal digits. Use Table A-1 to convert the binary LED code into hexadecimal.

Table A-1 Hexadecimal/Binary Conversion Chart

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

APPENDIX B
DECIMAL TO HEXADECIMAL CONVERSION

Some of the seek diagnostic tests in the RA80 Disk Drive call for the user to enter the cylinder addresses in hexadecimal format. Since there are 560 (decimal) cylinders available, the hexadecimal code must be entered into the two rotary switches in two bytes consisting of two hexadecimal digits each. The first byte entered is always the low byte of the cylinder address. The second byte is the high byte of the cylinder address. Use Table B-1 to convert the cylinder addresses between hexadecimal and decimal.

Table B-1 Hexadecimal/Decimal Cylinder Conversion Chart

Decimal Value	Hexadecimal Value	
	High Byte	Low Byte
10	00	0A
20	00	14
30	00	1E
40	00	28
50	00	32
60	00	3C
70	00	46
80	00	50
90	00	5A
100	00	64
110	00	6E
120	00	78
130	00	82
140	00	8C
150	00	96
160	00	A0
170	00	AA
180	00	B4
190	00	BE
200	00	C8

Table B-1 Hexadecimal/Decimal Cylinder Conversion Chart (Cont)

Decimal Value	Hexadecimal Value	
	High Byte	Low Byte
210	00	D2
220	00	DC
230	00	E6
240	00	F0
250	00	FA
260	01	04
270	01	0E
280	01	18
290	01	22
300	01	2C
310	01	36
320	01	40
330	01	4A
340	01	54
350	01	5E
360	01	68
370	01	72
380	01	7C
390	01	86
400	01	90

Table B-1 Hexadecimal/Decimal Cylinder Conversion Chart (Cont)

Decimal Value	Hexadecimal Value	
	High Byte	Low Byte
410	01	9A
420	01	A4
430	01	AE
440	01	B8
450	01	C2
460	01	CC
470	01	D6
480	01	E0
490	01	EA
500	01	F4
510	01	FE
520	02	08
530	02	12
540	02	1C
550	02	26
560	02	30

Reader's Comments

RA80 Disk Drive
Service Manual
EK-ORA80-SV-001

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: Communications Services (NR2/M15)
Customer Services Section

Order No. EK-ORA80-SV-001

— — — — — — — — — — — — 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
301 Rockrimmon Boulevard South
Colorado Springs, Colorado 80919

Educational Services
Development and Publishing

Digital Equipment Corporation • Bedford, MA 01730

RA80 SERVICE

EK-ORA80-SV-001

RA80 Disk Drive Service Manual

digital