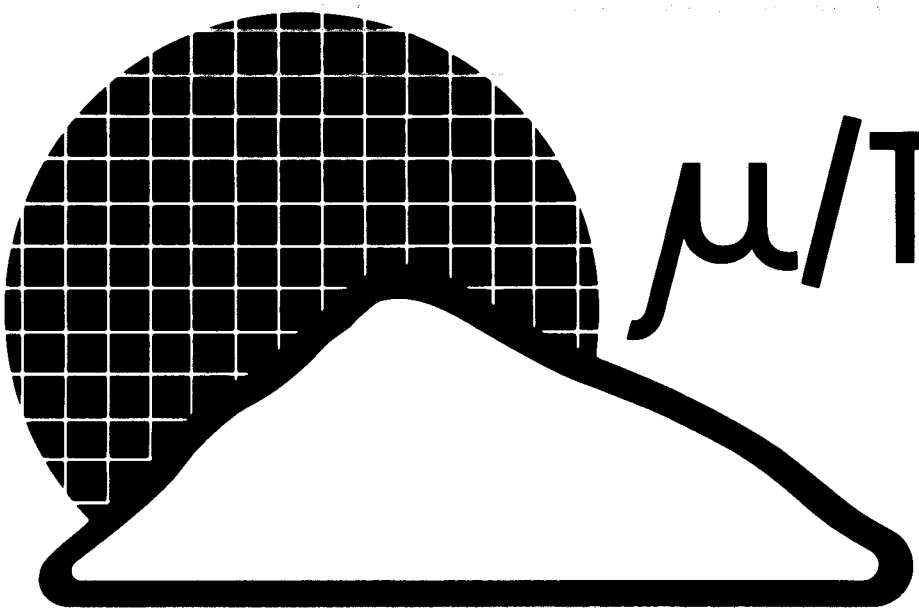
EK-DCT11-UG-003



μ/T-11

USER'S GUIDE

1st Edition, October 1980
2nd Edition, January 1982
3rd Edition, June 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.

The manuscript for this book was created on a DIGITAL Word Processing System and, via a translation program, was automatically typeset on DIGITAL's DECset-8000 Typesetting System. Book production was done by Educational Services Development and Publishing in Marlboro, MA.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

FIGURES

FIGURES (Cont)

FIGURES (Cont)

FIGURES (Cont)

TABLES

TABLES (Cont)

TABLES (Cont)

PREFACE

This user's guide is designed for engineers familiar with PDP-11 architecture. Chapters 1 through 6 offer a tutorial on DCT11-AA architecture and operation. (Chapter 5 includes some design examples.) Appendix A contains reference material (instruction set tables and timing diagrams). Appendix B briefly describes the software differences and compatibilities among the DCT11-AA and other members of the PDP-11 family.

This guide can be used by both hardware and software specialists. The hardware specialist should especially become familiar with Chapters 1 through 5, whereas the software specialist should become familiar with Chapters 1, 4, and 6.

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 2 (Bus Transactions) and Chapter 3 (Pin Descriptions) are arranged by mode. This allows the user to find, in one place, all the information relevant to a selected mode. A user not knowing which mode to use for a given application should first read Chapter 4 (Mode Selection).

CHAPTER 1
ARCHITECTURE

1.1 INTRODUCTION

This chapter describes the internal architecture of the DCT11-AA microprocessor. The chapter is divided into five sections covering all aspects of the architecture:

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

1.2 REGISTERS

The DCT11-AA contains a number of internal registers used for various purposes (refer to Figure 1-1). The registers are divided into three groups:

- General-Purpose
- Status
- Mode

1.2.1 General-Purpose 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 stack pointers for temporary storage of data. Arithmetic operations can be performed between one general-purpose register and another, one memory location or device register and another, between memory locations, or between a device register and a general register. The eight 16-bit general-purpose registers (R0–R7) are identified in Figure 1-2.

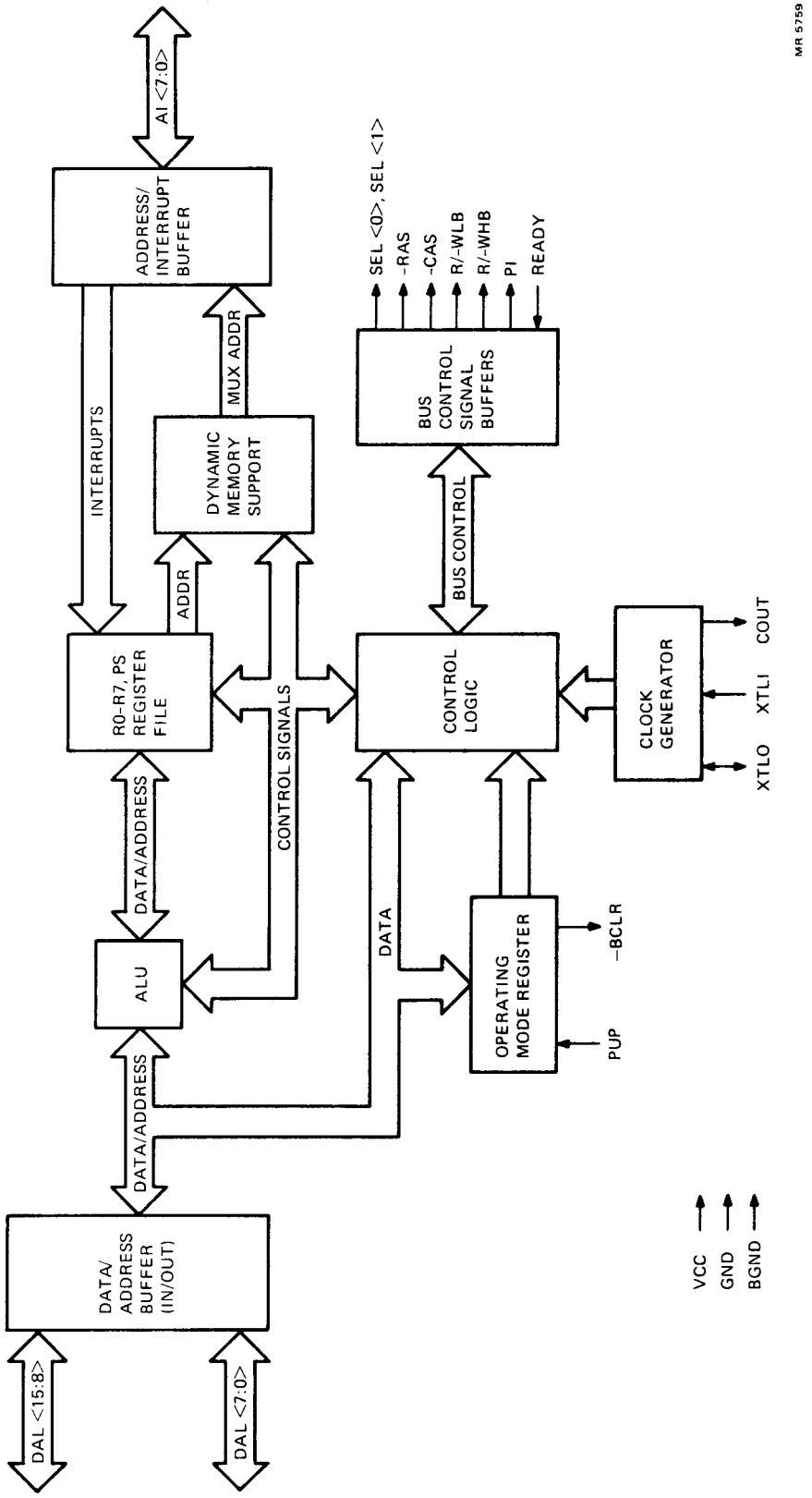


Figure 1-1 DCT11-AA, Block Diagram

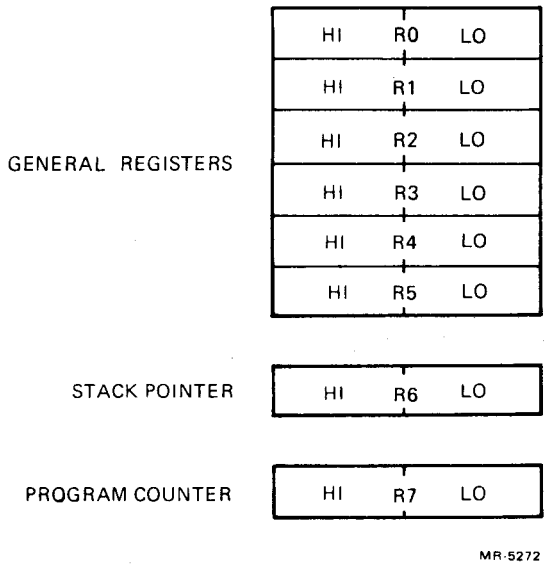


Figure 1-2 General-Purpose Registers

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. The PC 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.

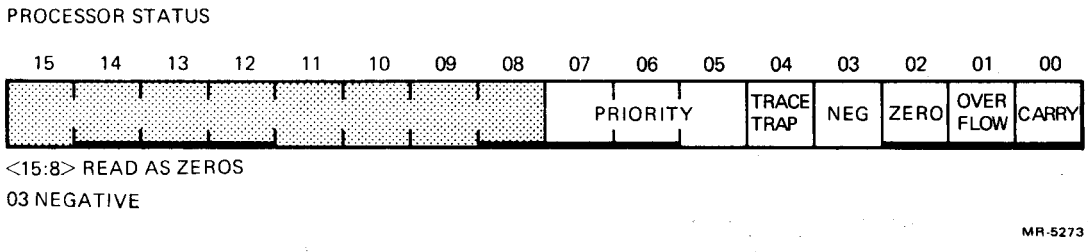


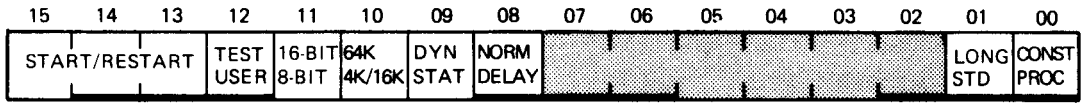
Figure 1-3 Processor Status Word

1.2.3 Mode Register

The DCT11-AA incorporates a user-loadable mode register (refer to Figure 1-4). The mode register is loaded at power-up or when a reset instruction is issued. Access to the mode register is not possible at any other time. 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
- One of eight start/restart address pairs

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



<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 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 in which they were stored. On the DCT11-AA the stack starts at the highest location reserved for it (376_8 at power-up) and expands linearly downward to a lower address as items are added to the stack. There is no stack overflow warning.

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 of the last item 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) that 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>.

The 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 nonmaskable interrupts, power fail (−PF) and −HALT.

1.5.1 Interrupt Mechanism

When the DCT11-AA receives an interrupt, no action is taken until the end of the current instruction (refer to Figure 1-5). Interrupts are only read during a read transaction or assert priority in (ASPI) transaction. Before fetching the next instruction, the DCT11-AA arbitrates the interrupt priority. If the interrupt request has a higher priority than the processor's, 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.

1.5.2 Interrupt Posting

With the assertion of the priority in (PI) signal, interrupts are read into the DCT11-AA during any read transaction and ASPI transaction. Interrupts are read in only at the occurrence of PI.

1.5.3 Interrupt Request (IRQ)

During the assertion of PI the interrupt request is read by the DCT11-AA (refer to Figures 1-5 and 1-6). Refer to Table 1-1 for signal names. Interrupt requests are implemented from the following seven different signals.

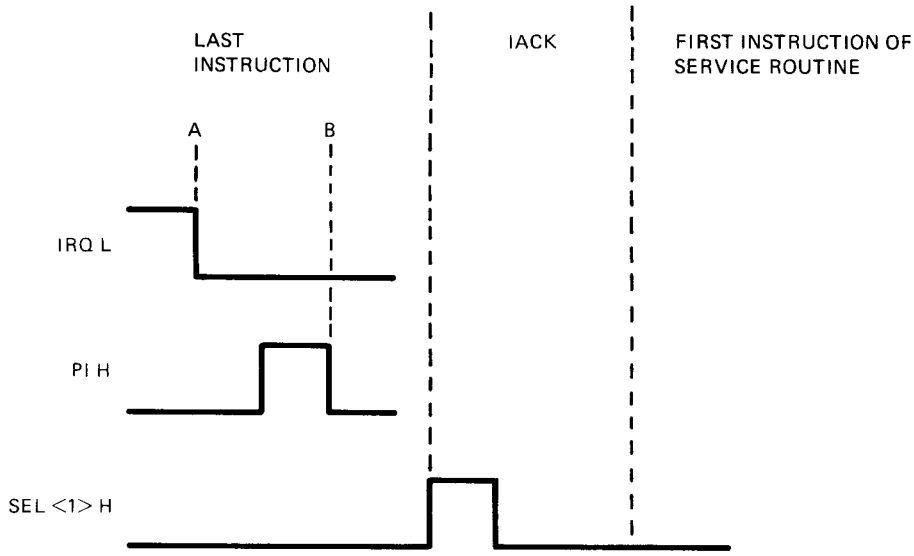


Figure 1-5 Interrupt Request

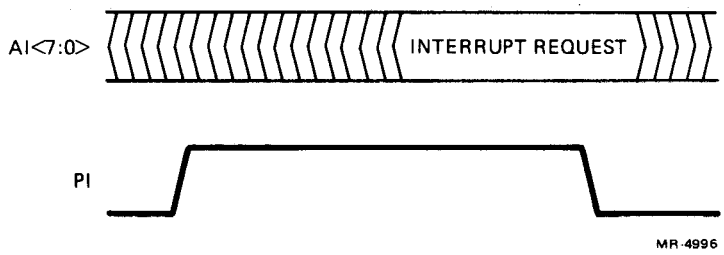


Figure 1-6 Interrupt Timing

Maskable interrupts:

- −CP<3:0> (coded priority)

Nonmaskable interrupt:

- −PF (power fail)
- −HALT (halt)

Control (internal or external) vector:

- −VEC (vector)

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

The 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>)

1.5.4 Vectors

Every interrupt except −HALT is associated with an interrupt vector. An 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 (374_8 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.) The −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_8). −HALT is not acknowledged.

1.5.4.1 Internal Vector Address - 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>. Refer to Table 1-2.

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
−HALT*	X	X	X	X	8	-
−PF	X	X	X	X	8	24
	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 - If during the assertion of PI (−PF or −HALT not asserted) −VEC (AI<5>) is asserted (low), the 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

Each interrupt is assigned a priority level (refer to Table 1-2). The DCT11-AA divides interrupts into two groups:

- Maskable
- Nonmaskable

1.5.5.1 Maskable Interrupts - Interrupts on −CP<3:0> are maskable. The interrupts are serviced according to their priority level (refer to Table 1-2).

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 until the stack is full. (Refer to Paragraph 2.12.)

1.5.5.2 Nonmaskable Interrupts - The −HALT interrupt has the highest priority; it interrupts the processor whatever the processor's status.

NOTE

The −HALT interrupt or execution of the −HALT instruction results in an interrupt, not in a stopping of 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.

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 (refer to Figure 1-7). DMR is read during any assertion of PI, unlike interrupts that are read only during a read or 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) when it asserts 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.

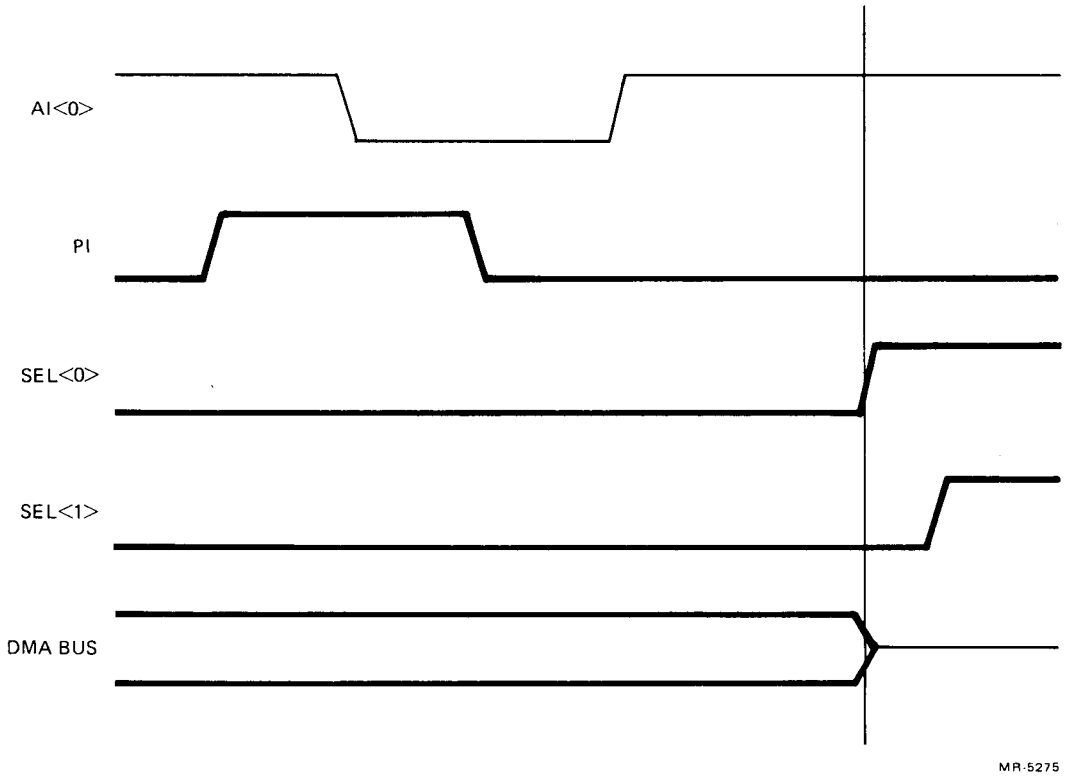


Figure 1-7 DMA Timing

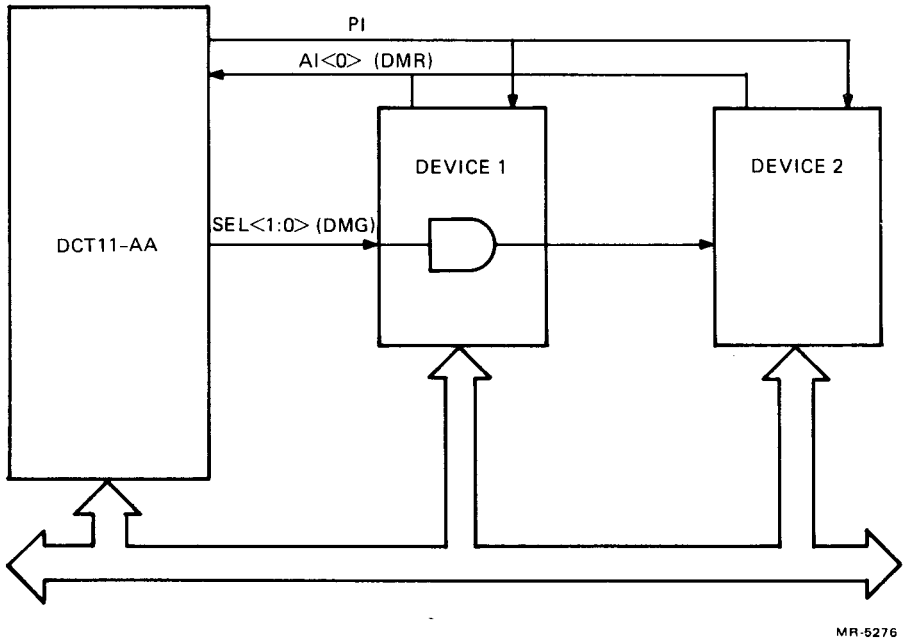


Figure 1-8 DMA, Block Diagram

CHAPTER 2
BUS TRANSACTIONS

2.1 INTRODUCTION

This chapter provides a basic discussion of each bus transaction. 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 separate discussion of each read and write transaction is presented. All other transactions are described as they apply to the DCT11-AA bus.

2.2 BUS TRANSACTION

Refer to Figure 2-1. Each PDP-11 instruction is composed of a number of transactions.

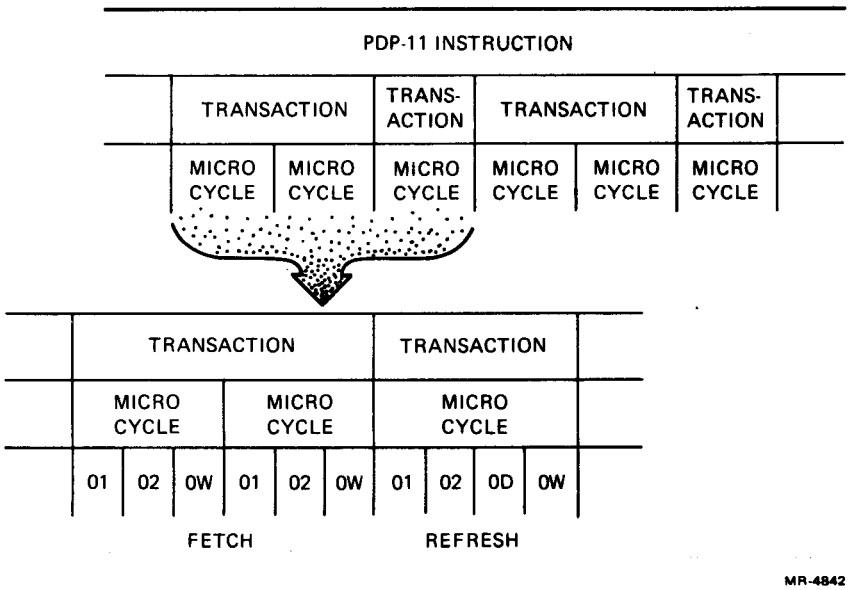


Figure 2-1 Parts of a Transaction

2.2.1 Transaction

A transaction is defined as an activity that takes 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 microinstruction to be executed. The microcycle performs the functions necessary to transfer information to and from the DCT11-AA bus, move data internally, and calculate 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: $\phi 1$, $\phi 2$, and ϕW. 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 (ϕD), between $\phi 2$ and ϕW. All clock phases have the same duration between assertions.

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 in Appendix A.

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

2.3.1 Output of Address

Refer to Figures 2-2 and 2-3. The address is output on the data address lines (DALs) 15–0 (<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 are used for both address and data.

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

The input data should be valid on DAL<15:0> during the period that priority in (PI) is asserted (refer to Figure 2-3).

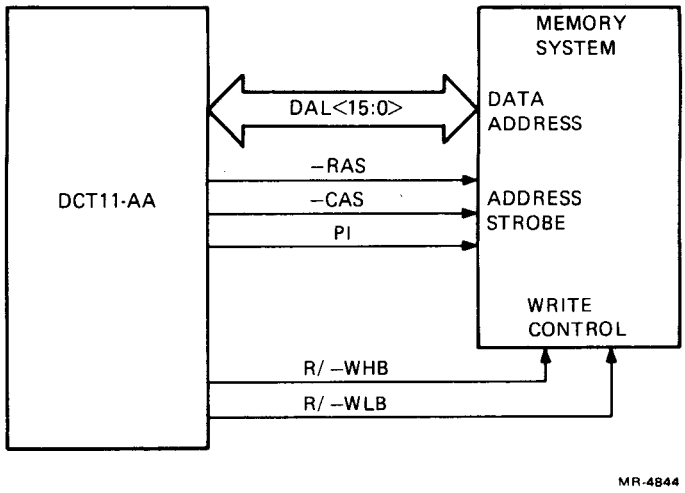


Figure 2-2 16-Bit Static Read, Block Diagram

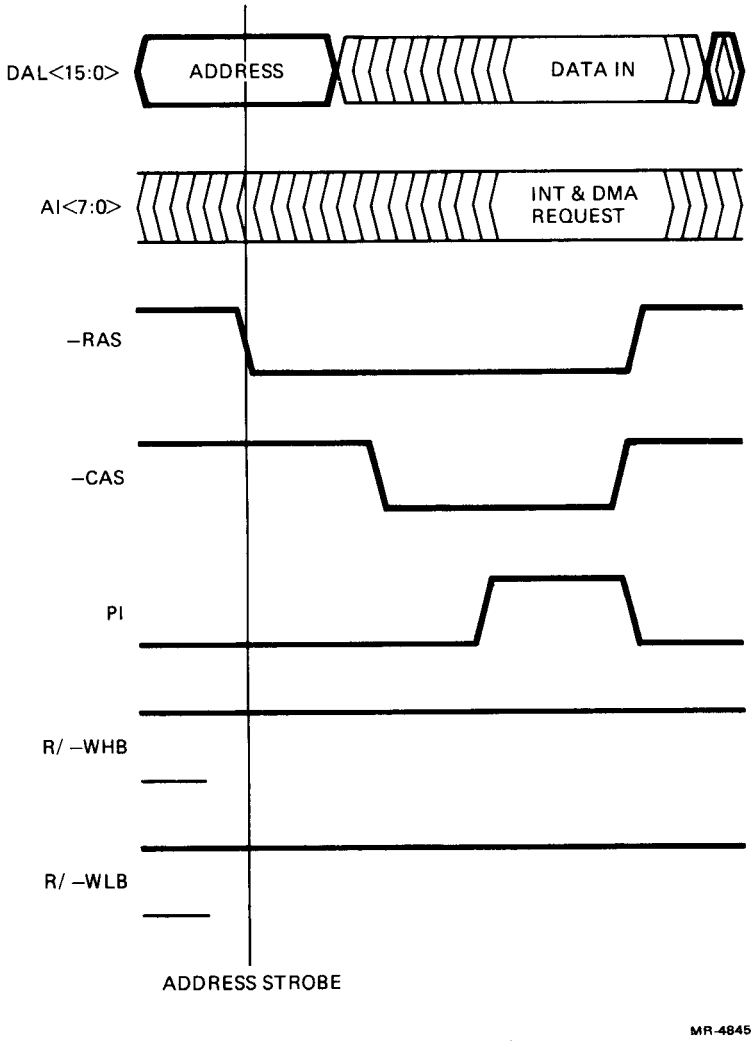


Figure 2-3 16-Bit Static Read Timing

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 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.

2.4 16-BIT STATIC WRITE TRANSACTION

A write transaction is composed 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 are to the processor.

A write transaction is always preceded by a read transaction (the two are indivisible) except when writing the stack during an interrupt or trap.

2.4.1 Output of Address

Refer 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.

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

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

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.

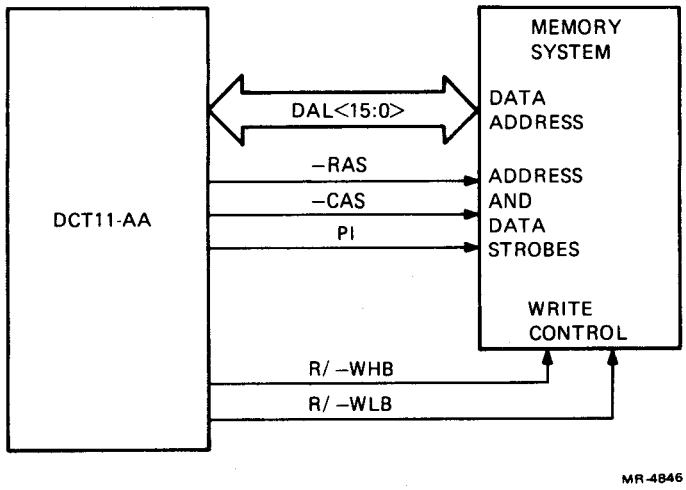


Figure 2-4 16-Bit Static Write, Block Diagram

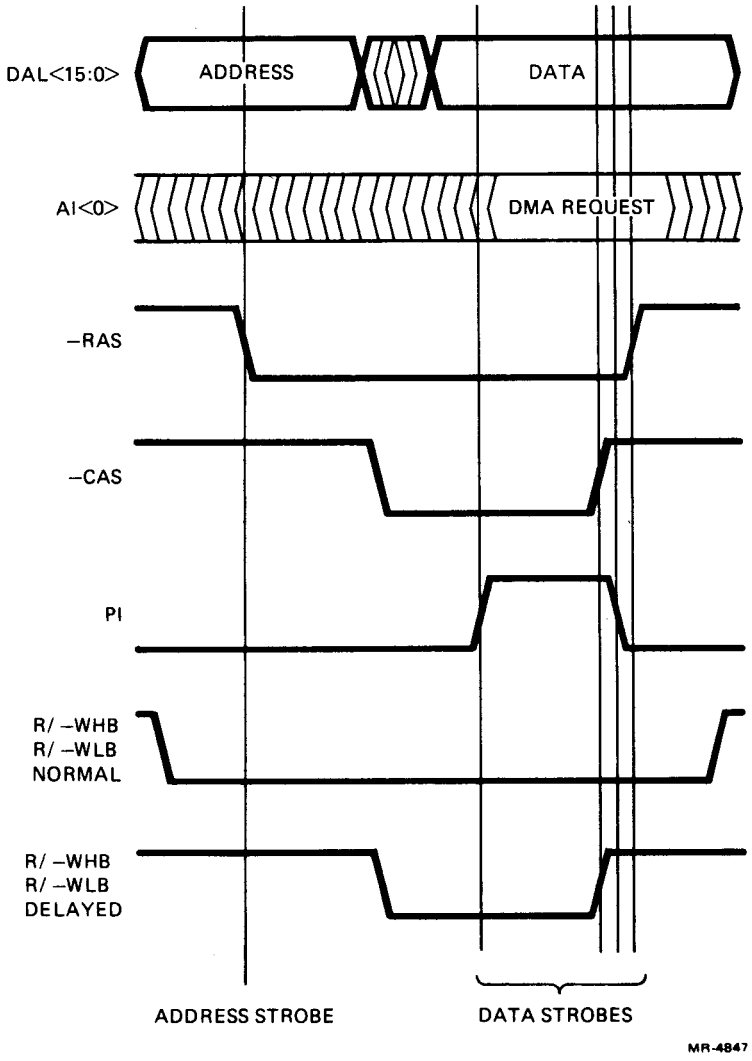


Figure 2-5 16-Bit Static Write Timing

Table 2-1 16-Bit Static 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 16-Bit Static Write Data Strobes

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 in Appendix A.

NOTE
All references to input or output are 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 – Refer to Figures 2-6 and 2-7. The address is output on the address interrupt (AI) lines 7–0 (<7:0>). The AI lines output the row address first and the column address second. Table 2-3 lists the address bits required in 4K/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 edges required to latch each portion of the address into the memory system or register.

2.5.2 Input of Data

Refer to Figure 2-7. The input data should be valid on DAL<15:0> during the period of time that PI is asserted. The negation of −CAS strobes the data into the DCT11-AA.

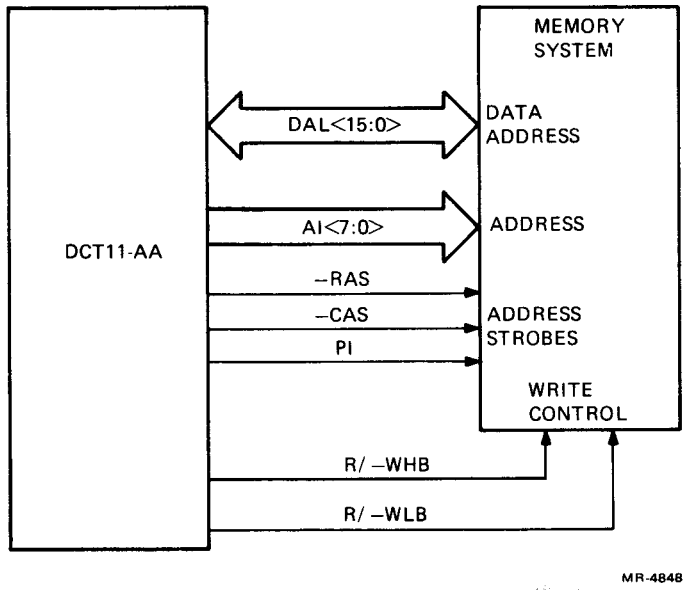


Figure 2-6 16-Bit Dynamic Read, Block Diagram

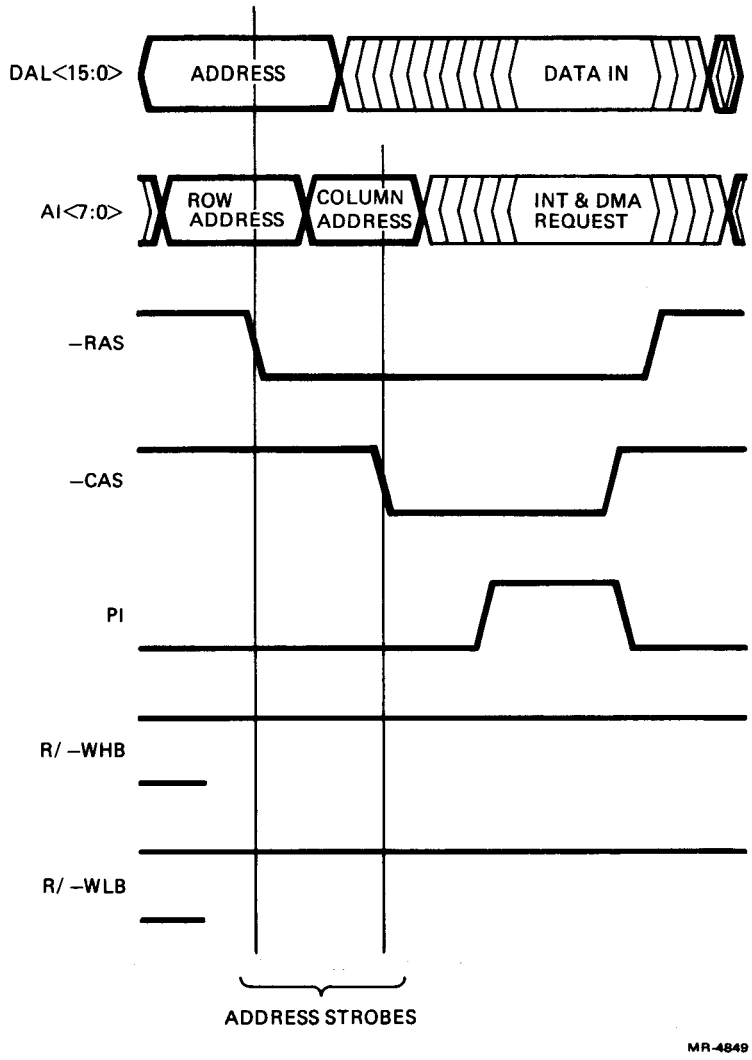


Figure 2-7 16-Bit Dynamic Read Timing

Table 2-3 16-Bit Dynamic Read Addressing Scheme

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

Table 2-4 16-Bit Dynamic Read AI Addressing

	Address			
AI	4K/16K −RAS	−CAS	64K −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	A12

Table 2-5 16-Bit Dynamic Read 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

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 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 three-stated before the leading edge of PI. Fetch is indicated by AI<0> high during −RAS.

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

2.5.3.2 64K Mode – Static modes and 64K 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 and SEL<1> low.

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 are to the processor.

A write transaction is always preceded by a read transaction (the two are indivisible) except when writing the stack during an interrupt or trap.

2.6.1 Output of Address

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

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

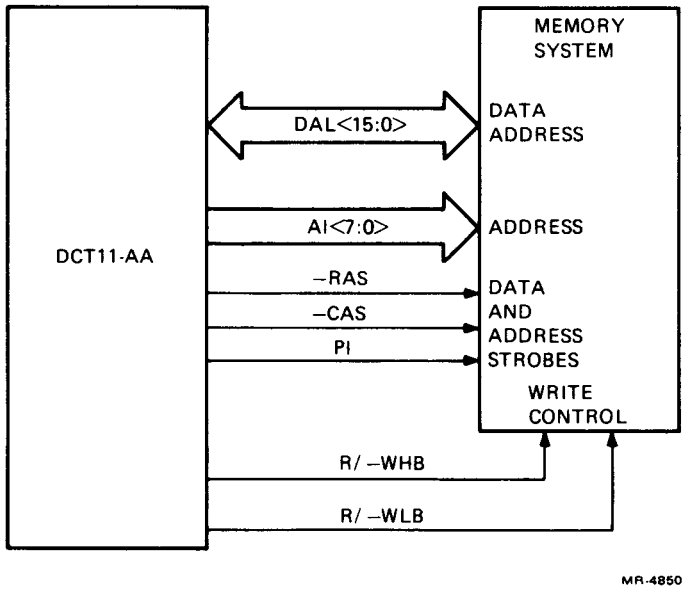


Figure 2-8 16-Bit Dynamic Write, Block Diagram

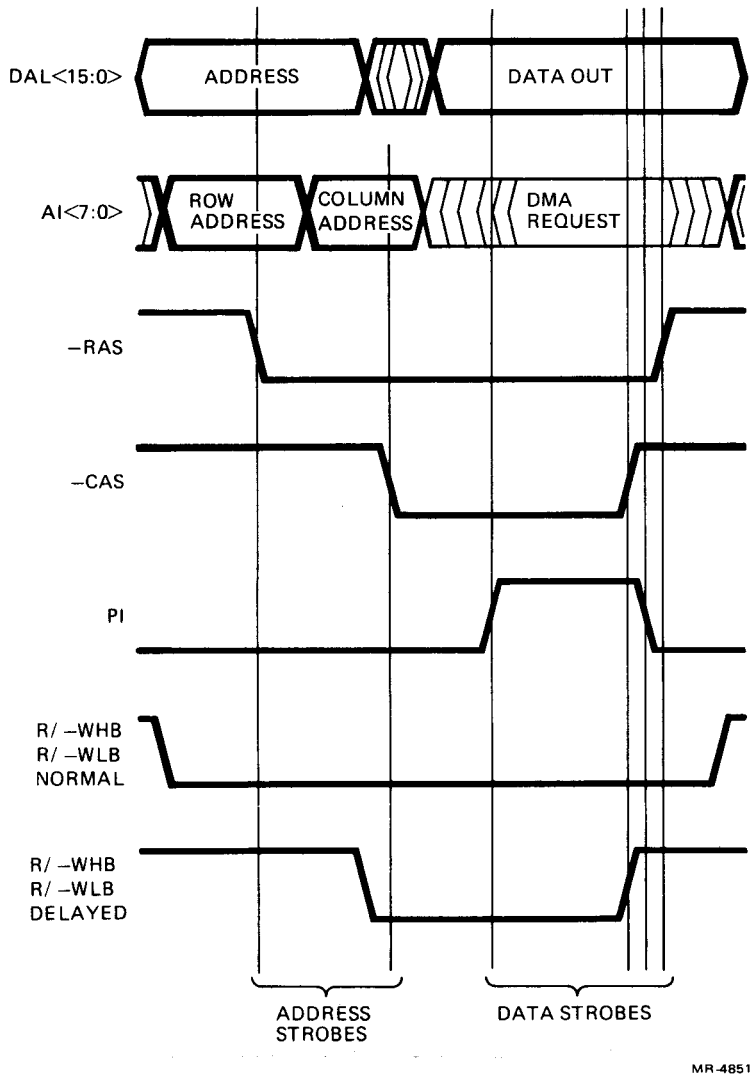


Figure 2-9 16-Bit Dynamic Write Timing

Table 2-6 16-Bit Dynamic Write Addressing Scheme

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

*Address lines necessary to address all bits in each chip.

Table 2-7 16-Bit Dynamic Write AI Addressing

	Address			
AI	4K/16K −RAS	−CAS	64K −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	A12

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

2.6.2 Output of Data

Refer to Figure 2-9. The data is output on DAL<15:0>.

Data Control – The signals used to latch the data into the memory system or register and the edges required are 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 timing of R/−WHB and R/−WLB is found in Table 2-11.

Table 2-8 16-Bit Dynamic Write 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 16-Bit Dynamic Write Data Strobes

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

Table 2-10 16-Bit Dynamic 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 16-Bit Dynamic 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 an 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 are to the processor.

2.7.1 Output of Address

Refer to Figures 2-10 and 2-11. The high byte address is output on the static address lines (SALs) 15–8 (<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.

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

Refer to Figure 2-11. The input data should be valid on DAL<7:0> during the period PI is asserted.

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 timing of −Read is found in Table 2-12.

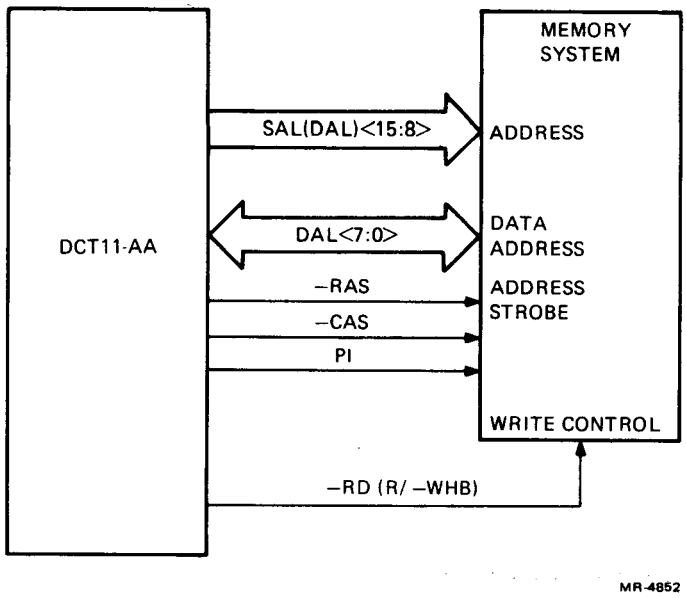


Figure 2-10 8-Bit Static Read, Block Diagram

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.

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 an 8-bit static write transaction is found in Figure A-7 in 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 are to the processor.

A write transaction is always preceded by a read transaction (the two are indivisible) except when writing the stack during an interrupt or trap.

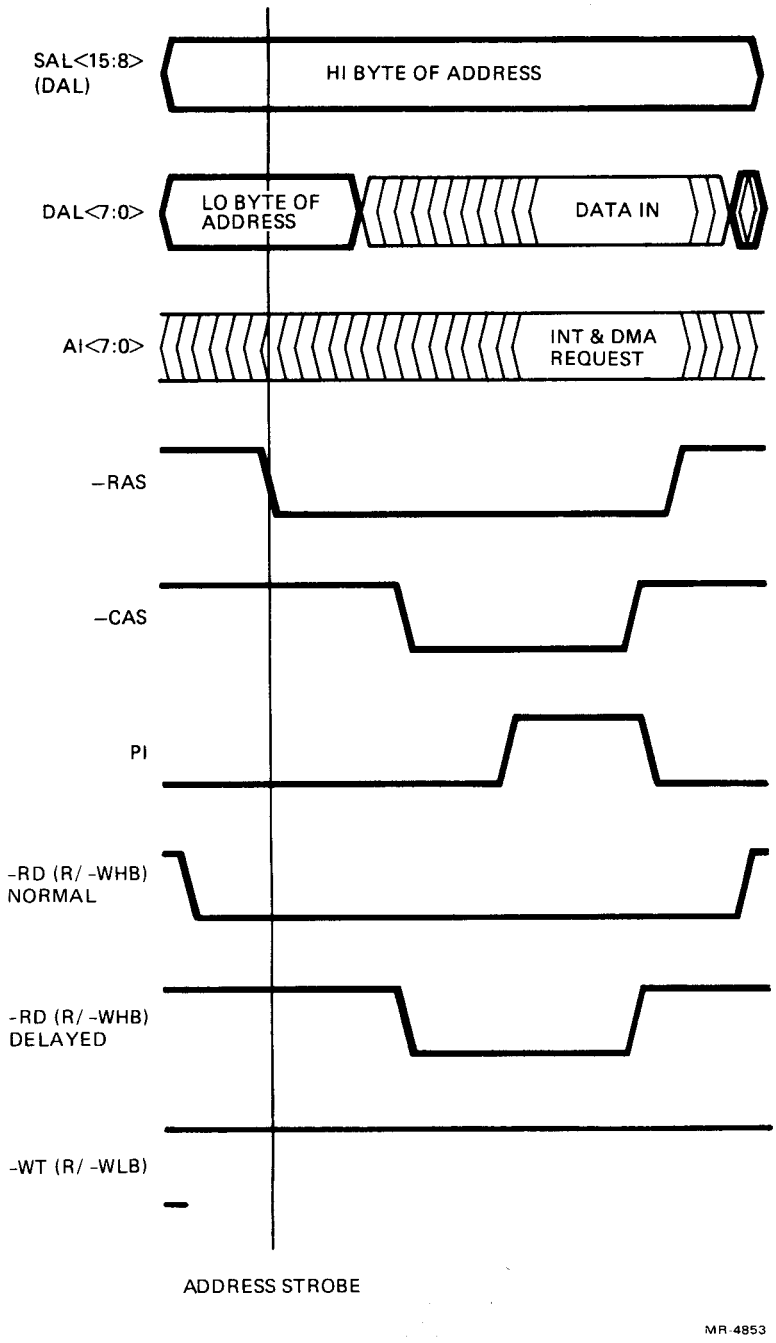


Figure 2-11 8-Bit Static Read Timing

Table 2-12 8-Bit Static 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

2.8.1 Output of Address

Refer to Figures 2-12 and 2-13. The high byte address is output on the static address lines (SALs) 15–8 (<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.

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

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

Data Control – The signals used to latch the data into the memory system or register and the edges required are found in Table 2-13. Write control is accomplished through the use of the signal –Write (R/–WLB). The timing of –Write is found in Table 2-14.

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 are to the processor.

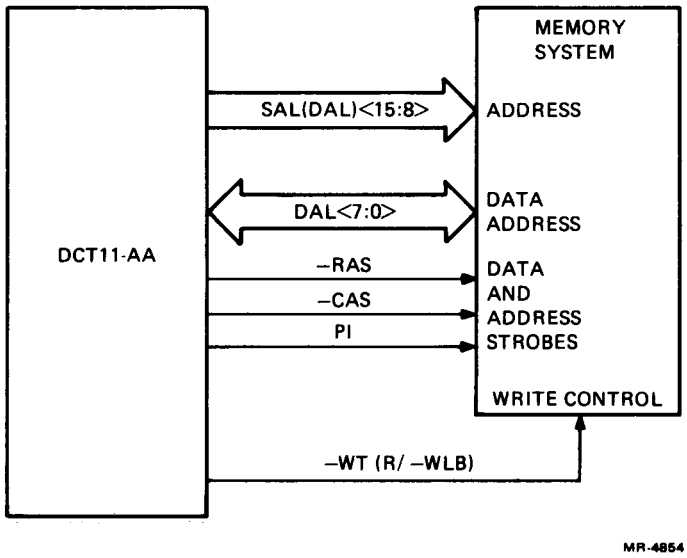


Figure 2-12 8-Bit Static Write, Block Diagram

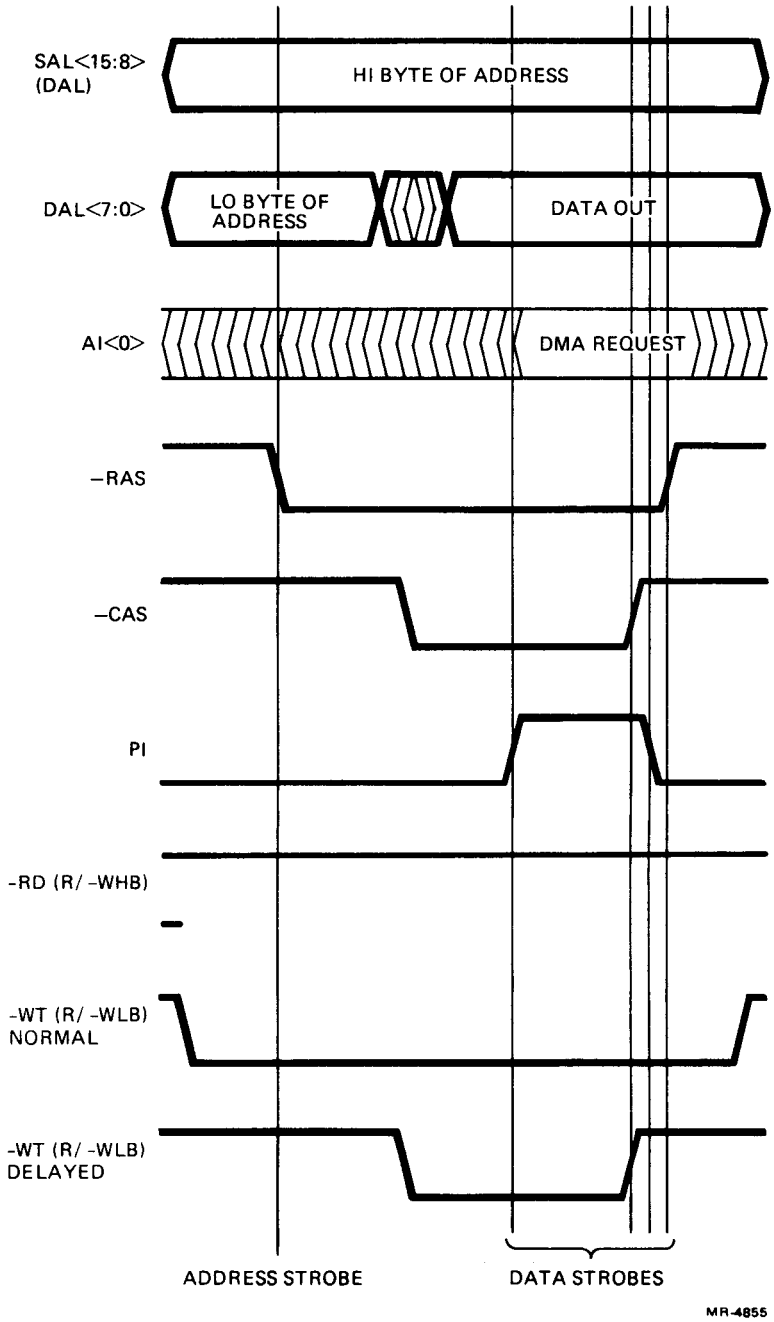


Figure 2-13 8-Bit Static Write Timing

Table 2-13 8-Bit Static Read Data Strobes

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

Table 2-14 8-Bit Static 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

2.9.1 Output of Address

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

2.9.1.1 Dynamic Address – Refer to Figures 2-14 and 2-15. The address is output on AI<7:0>. The AI lines output the row address first and the column address second. Table 2-15 lists the address bits required in 4K/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 (SALs) 15–8 (<15:8>). The low byte of the address is output on DAL<7:0>.

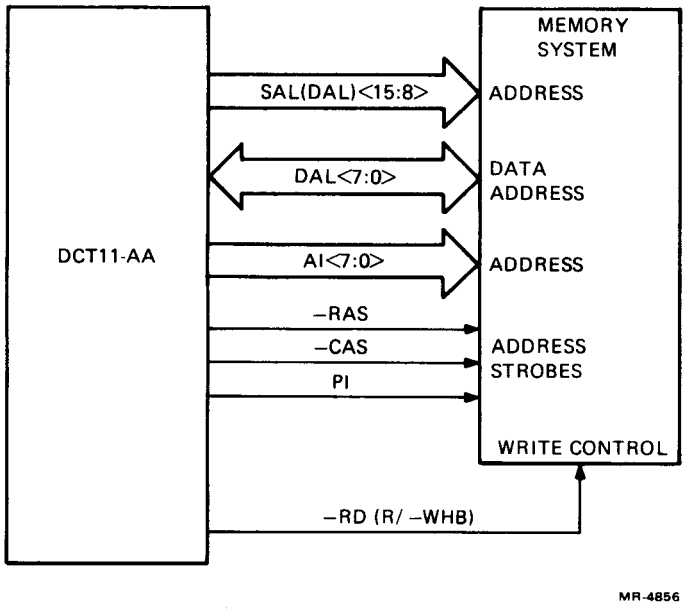


Figure 2-14 8-Bit Dynamic Read, Block Diagram

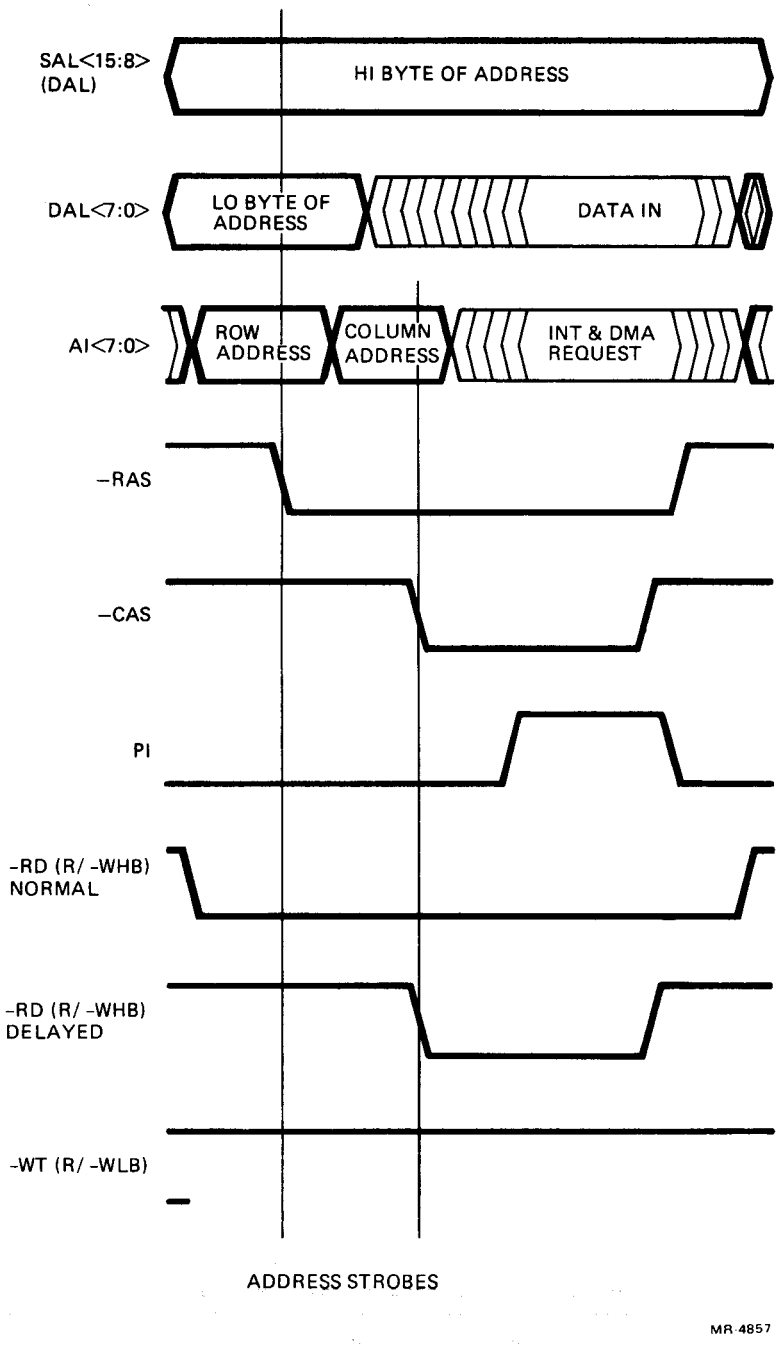


Figure 2-15 8-Bit Dynamic Read Timing

Table 2-15 8-Bit Dynamic Read Addressing Scheme

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

Table 2-16 8-Bit Dynamic AI Addressing

	Address			
AI	4K/16K −RAS	−CAS	64K −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

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

2.9.2 Input of Data

Refer to Figure 2-15. The input data should be valid on DAL<7:0> during the period PI is asserted.

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 one signal −Read (R/−WHB). The timing of −Read is found in Table 2-18.

Table 2-17 8-Bit Dynamic Read 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 8-Bit Dynamic 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

2.9.3 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 three-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 – Static modes and 64K 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 and SEL<1> low.

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 an 8-bit dynamic write transaction is found in Figure A-9 in 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 are to the processor.

A write transaction is always preceded by a read transaction (the two are indivisible) except when writing the stack during an interrupt or trap.

2.10.1 Output of Address

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

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

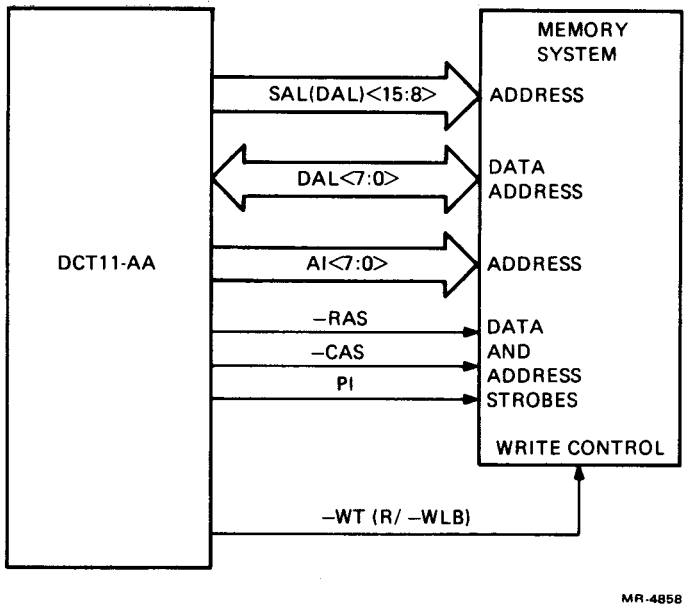


Figure 2-16 8-Bit Dynamic Write, Block Diagram

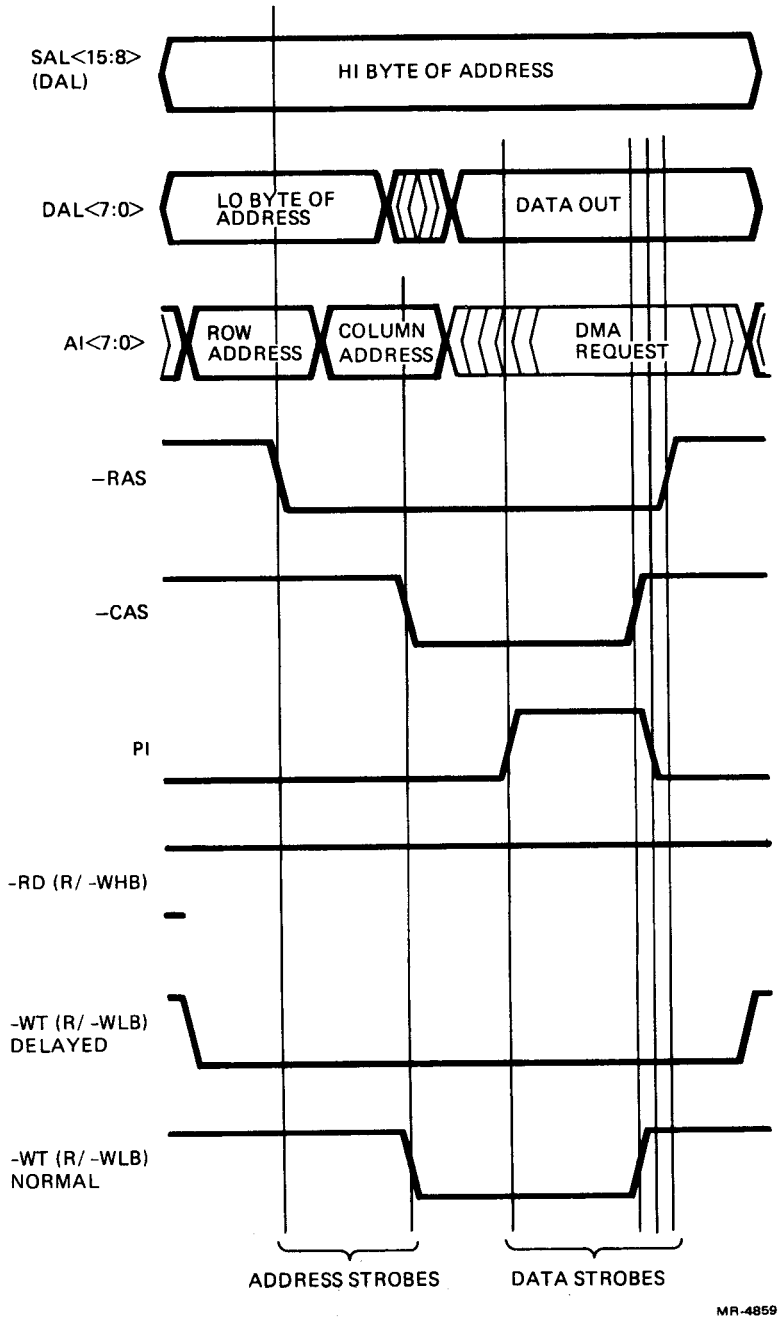


Figure 2-17 8-Bit Dynamic Write Timing

Table 2-19 8-Bit Dynamic Write Addressing Scheme

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

Table 2-20 8-Bit Dynamic Write AI Addressing

	Address			
AI	4K/16K −RAS	−CAS	64K −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

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

2.10.2 Output of Data

Refer to Figure 2-17. The data is output on DAL<7:0>.

Data Control – The signals used to latch the data into a memory system or register and the edge required are 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 2-21 8-Bit Dynamic Write 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 8-Bit Dynamic Write Data Strobes

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

Table 2-23 8-Bit Dynamic 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 in Appendix A.

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

2.11.1 Output of Refresh Address

Refer 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
  Autodecrement-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.

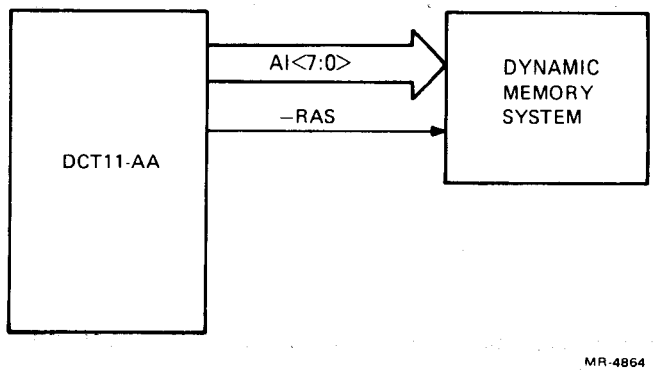


Figure 2-18 Refresh Transaction, Block Diagram

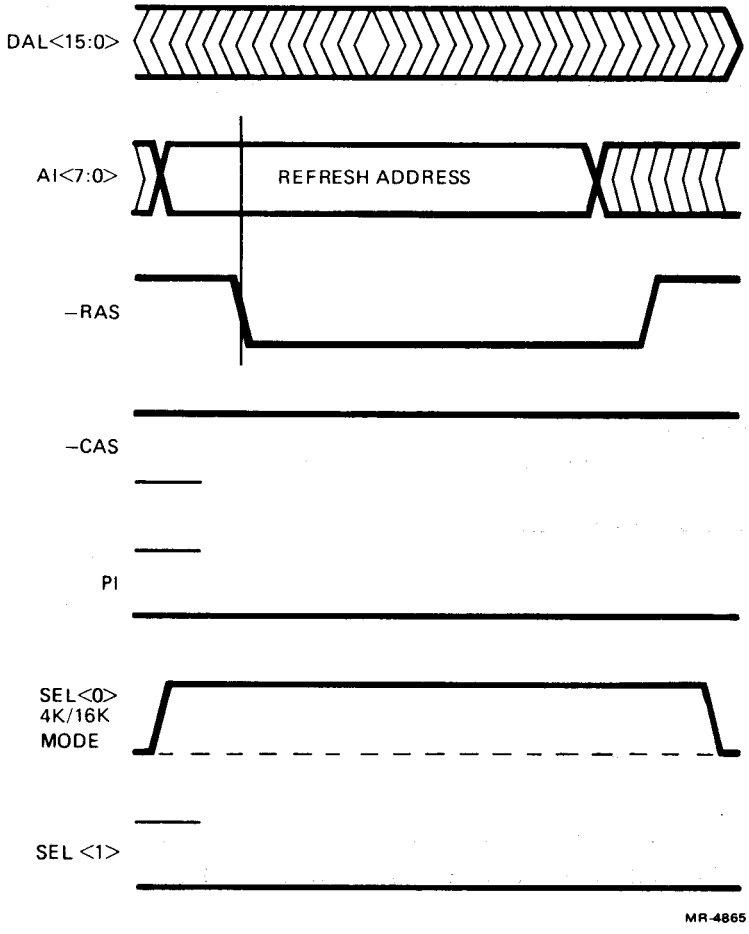


Figure 2-19 Refresh Transaction Timing

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

Refer 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

SEL<1:0> are low for other transactions (refer to Table A-10 in Appendix A).

2.12 IACK (INTERRUPT ACKNOWLEDGE) TRANSACTION

An IACK transaction, which clears the interrupt request, consists of two distinct processes:

- Output of interrupt acknowledge data
- Input of vector address [if −VEC (AI<5>) was asserted]

Detailed timing of an IACK transaction is found in Figure A-11 in Appendix A.

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

2.12.1 Output of Interrupt Acknowledge Data

Refer to Figures 2-20 and 2-21. The processor first outputs the interrupt acknowledge data on DAL<12:8> with the same polarity as the received data. The acknowledge data consists of the coded priority of the interrupting device. This coded priority was first received on AI<5:1> at the time of the interrupt request. Refer to Table 2-24. The strobe, which is provided for the interrupting device to use, is −RAS. The interrupt acknowledge is valid upon the assertion (leading edge) of −RAS.

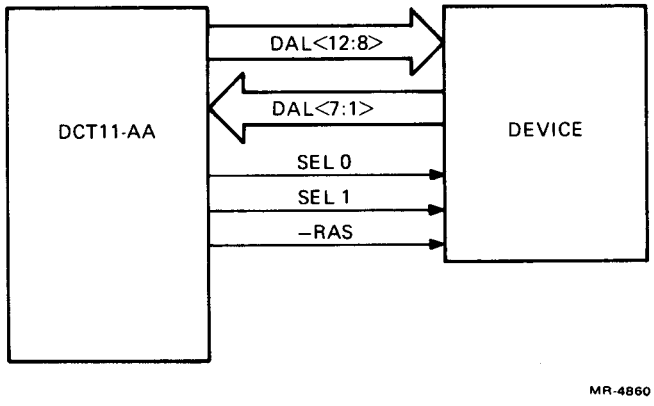


Figure 2-20 IACK Transaction, Block Diagram

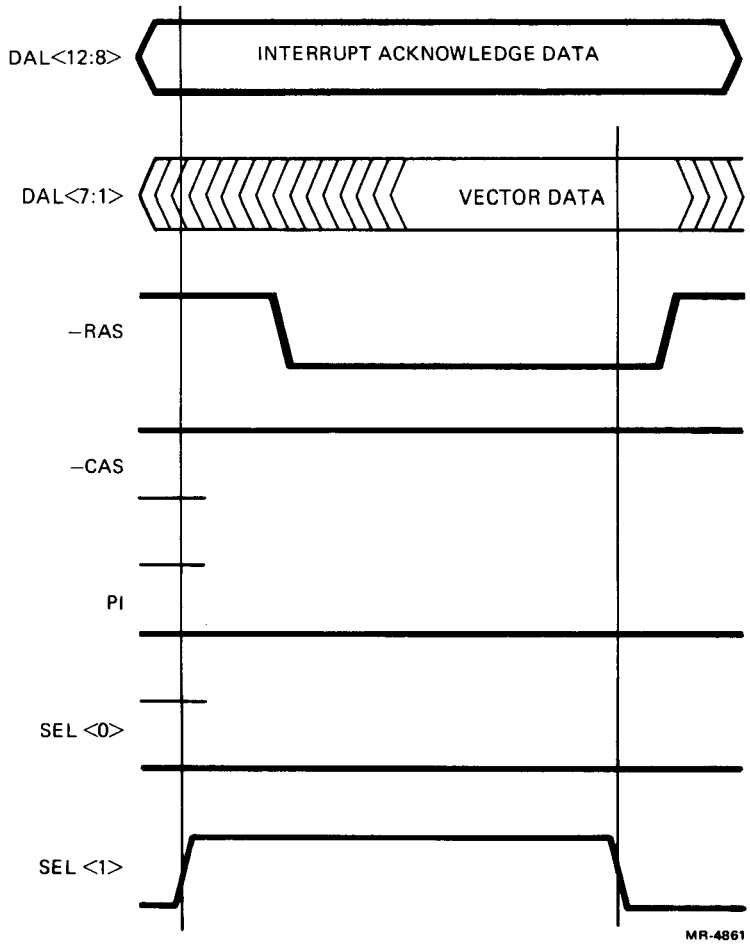


Figure 2-21 IACK Transaction Timing

Table 2-24 Interrupt Acknowledge Data

Interrupt Request	Acknowledge
–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>

2.12.2 Input of Vector Address

If vector (–VEC) AI<5> was asserted at the time of the interrupt request, the input of an external vector address should be driven by the user on DAL<7:2>. If –VEC was not asserted at the time of the interrupt request, 1 of the 15 vector addresses internal to the processor is used.

Refer to Figure 2-21. Select (SEL) output flag 1 (<1>) is used by the processor to input the vector. The vector address is latched upon the negation (trailing edge) of SEL<1>. If the READY input is asserted, the latching of the vector address into the DCT11-AA is delayed by one microcycle. (Depending on the pulsing of READY, more microcycles may be added.)

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

A busnop transaction occurs, for example, during an instruction decode cycle and internal processor computations. Detailed timing of a busnop transaction is found in Figure A-12 in Appendix A.

2.14 DMA (DIRECT MEMORY ACCESS) TRANSACTION

A DMA transaction consists of three processes:

- Three-state of DAL<15:0> and internal pull-ups 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 in Appendix A.

NOTE

All references to input or output are 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 is responsible 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.

1. A DMA request (DMR) to the DCT11-AA is made by driving AI<0> low during PI.

2. The request is latched into the DCT11-AA during PI and shortly thereafter a DMA grant is issued.

3. 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>

Refer 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.

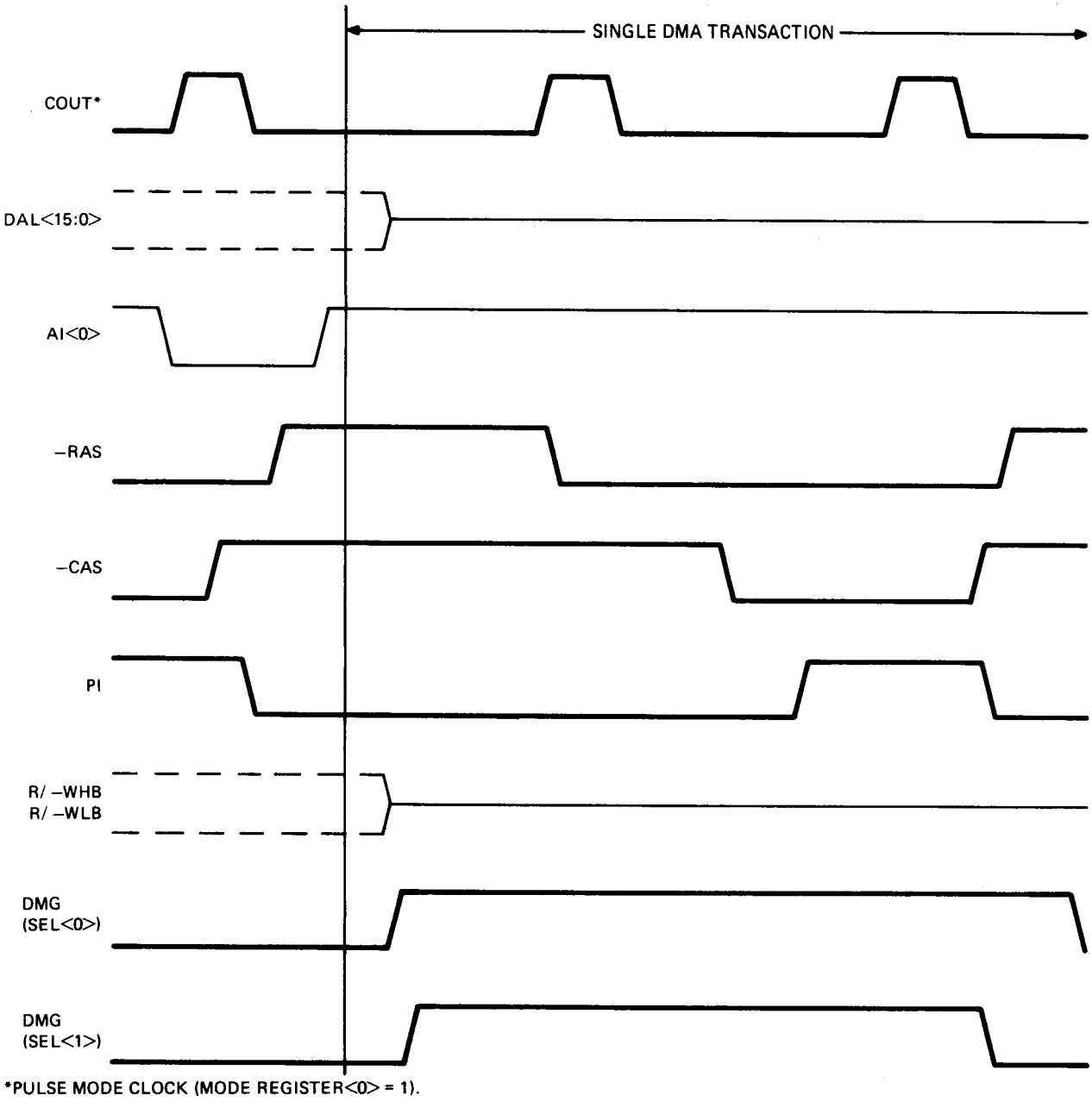


Figure 2-22 DMA Timing

2.14.2 Output of −RAS, −CAS, and PI

The −RAS and −CAS signals 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.

2.14.3 Output of Direct Memory Grant (DMG)

Refer 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 (refer to Figure A-14 in Appendix A).
- The DALs are three-stated.
- AI<7:0>, R/−WHB, and R/−WLB are implemented by internal pull-ups.

When the grant is issued, external circuitry must drive the R/−WHB and R/−WLB lines and initially drive the DALs 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 DALs 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 in Appendix A.

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

Refer 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.

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.

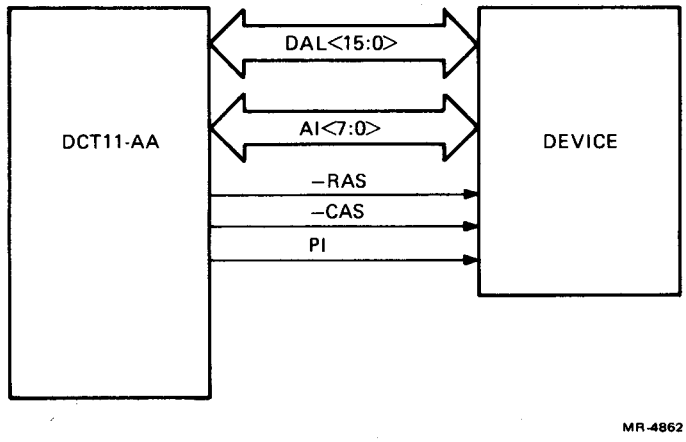


Figure 2-23 ASPI Transaction, Block Diagram

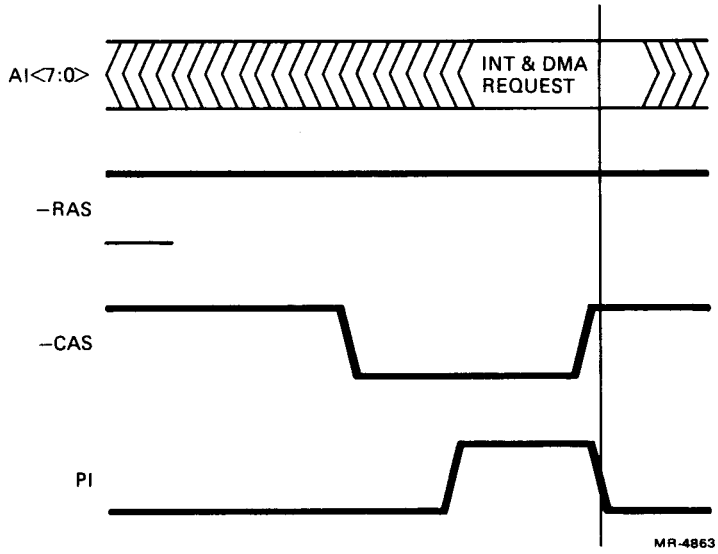


Figure 2-24 ASPI Transaction Timing

CHAPTER 3
PIN DESCRIPTIONS

3.1 INTRODUCTION

This chapter describes the functions performed by each 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})

Refer 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 pin names. The mode-dependent pins are

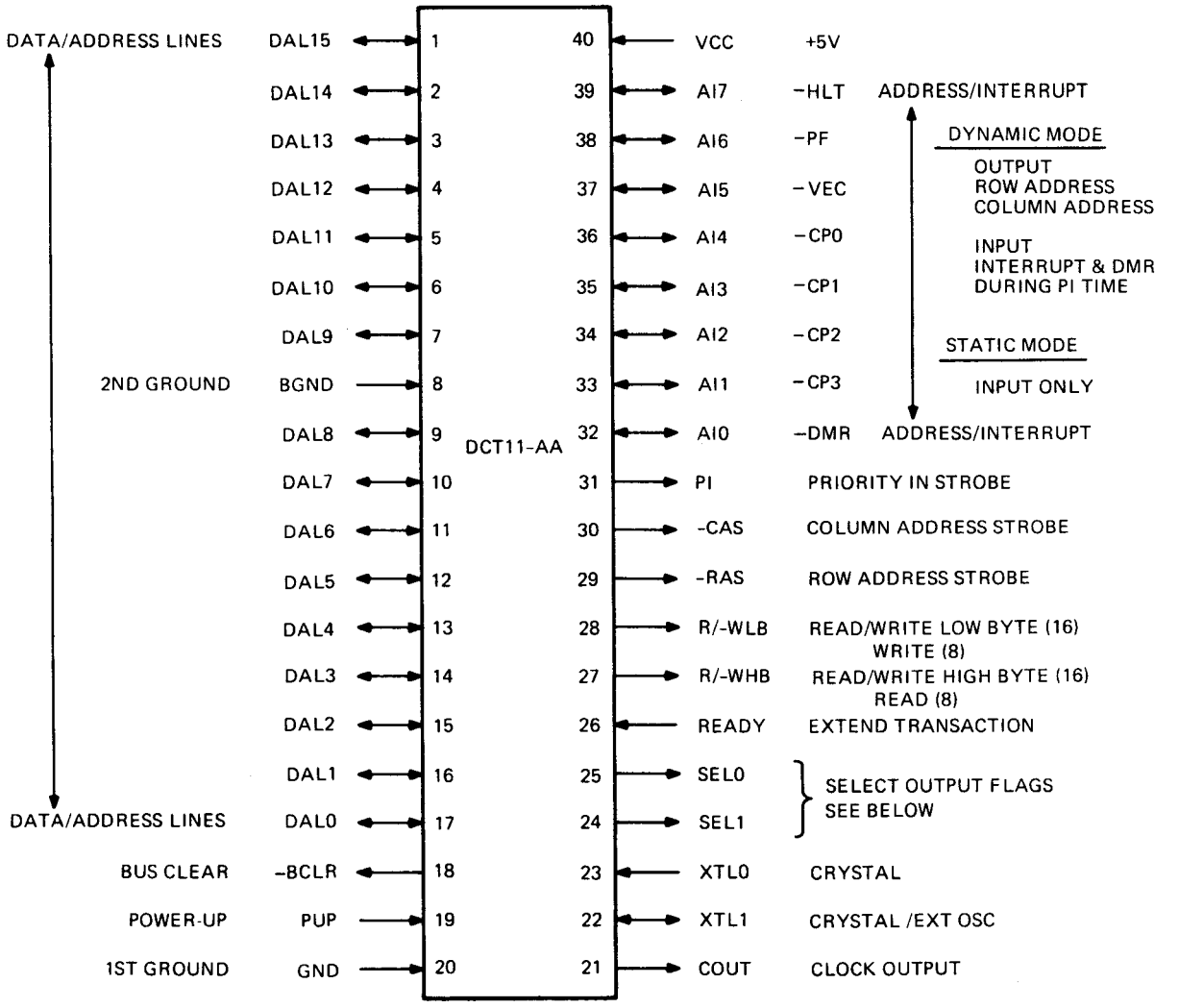
- DAL<15:0>
- AI<7:0>
- Select (SEL<1:0>)
- Read/−Write High Byte (R/−WHB)
- Read/−Write Low Byte (R/−WLB)
- Clock Output (COUT)

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.

Table 3-1 Mapping of AI onto DAL during an 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. For example, if AI<1> is high at interrupt request time, DAL<8> is high at IACK time.



SELECT OUTPUT FLAGS

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

MR-5271

Figure 3-1 DCT11-AA Pin Layout

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

		Signal Names							
Pin(s)	Pin Name	Static	4K/16K Dynamic			64K Dynamic			
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*	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†	IACK + DMG	IACK + DMG			IACK + DMG			
25	SEL0†	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

*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.

†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

		Signal Names							
Pin(s)	Pin Name	Static	4K/16K Dynamic			64K Dynamic			
Data Address Lines									
1–7,9 10–17	DAL<15:8> DAL<7:0>	SAL<15:8> DAL<7:0>	SAL<15:8> DAL<7:0>			SAL<15:8> DAL<7:0>			
Address Interrupt Lines									
			−RAS	−CAS	PI	−RAS	−CAS	PI	
32	AI<0>	−DMR	FET*	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†	IACK + DMG	IACK + DMG			IACK + DMG			
25	SEL0†	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

*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.

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

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

DAL<15:0> functions depend upon the selection of 8-bit or 16-bit mode. During read/write transactions (refer to Paragraph 2.2.1) the DALs 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 low byte/high byte multiplexing.

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)

3.2.1 16-Bit Mode – DAL<15:0>

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

1. 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 through 2.10.

2. 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.

3. During a DMA transaction:

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

4. During a busnop and refresh transaction:

   DAL<15:0> contain previously latched data.

5. During an ASPI transaction:

   DAL<15:0> are three-stated.

6. 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 pull-ups are enabled on these lines when –BCLR is asserted. This avoids the need to drive the bits that 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>), which are used in six cases.

1. 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 through 2.10.

2. 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.

3. During a DMA transaction:

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

4. During a busnop and refresh transaction:

   DAL<15:0> contain previously latched data.

5. During an ASPI transaction:

   DAL<15:0> are three-stated.

6. During the power-up sequence or a reset instruction:

   The mode register bits are read in from DAL<15:8>. Low-current internal pull-ups are enabled on these lines when –BCLR is asserted. This avoids the need to drive the bits that are to be high.

3.2.3 8-Bit Mode – DAL<7:0>

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

1. 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 through 2.10.

2. 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 the 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.

3. During a DMA transaction:

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

4. During a busnop and refresh transaction:

   DAL<15:0> contain previously latched data.

5. During an ASPI transaction:

   DAL<15:0> are three-stated.

6. During the power-up sequence or a reset instruction:

   The mode register bits are read in from DAL<1:0>. Low-current internal pull-ups are enabled on these lines when −BCLR is asserted. This avoids the need to drive the bits that are to be high.

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, dynamic 4K/16K, or dynamic 64K. 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)

While in static mode the address interrupt lines are used as inputs for interrupts and/or DMA requests during all transactions. 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 AIs 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 Modes)

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

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

* – Internal low-current passive pull-ups.
N/A – Not applicable.

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 the 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> are asserted. −VEC signals the processor to ignore the internal vector address indicated by −CP<3:0> and instead uses 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 setup 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_8. The −HALT input is pseudo-edge sensitive. It must be read as a negation before another assertion is recognized.

3.4 CONTROL LINES

The control lines are composed of signals 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).

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 as a PDP-11 instruction fetch; in 8-bit mode, asserted only in the low-byte transaction of a fetch.
2 – Asserted in dynamic 4K/16K mode.
3 – Three-stated.

The −RAS, −CAS, and PI signals 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 shows the leading and trailing edges. The leading edge is the edge that changes the signal from the unasserted state to the asserted state.

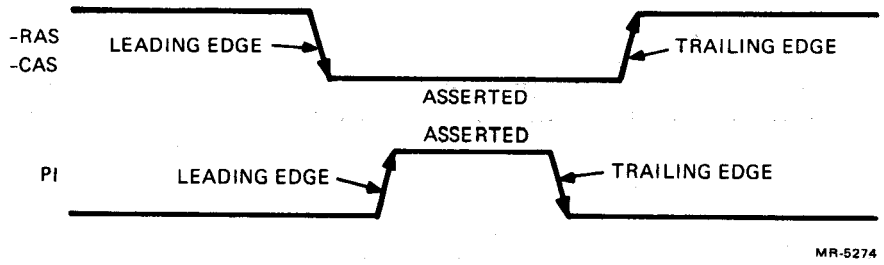


Figure 3-2 Leading and Trailing Edge

3.4.1 −RAS (Row Address Strobe)

The −RAS signal 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 present on the DALs (for memories not using the −RAS/−CAS multiplexing) and the row address present on the AIs (for the dynamic memories that 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 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)

The −CAS signal 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 occurrence of PI, and therefore, may be used to latch interrupt and DMA requests before they are strobed 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 DALs 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 used in read/write transactions.

During ASPI transactions the assertion (leading edge) of −CAS may be used to latch interrupt and DMA requests before they are strobed 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 are 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 data output strobes.

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. The −RD signal 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 with 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.

Refer to Figure 3-3. A single assertion of READY causes a single microcycle slip. An additional cycle slip requires the READY signal to be pulsed again. The assertion of READY has no 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.

™Intel is a trademark of the Intel Corporation.

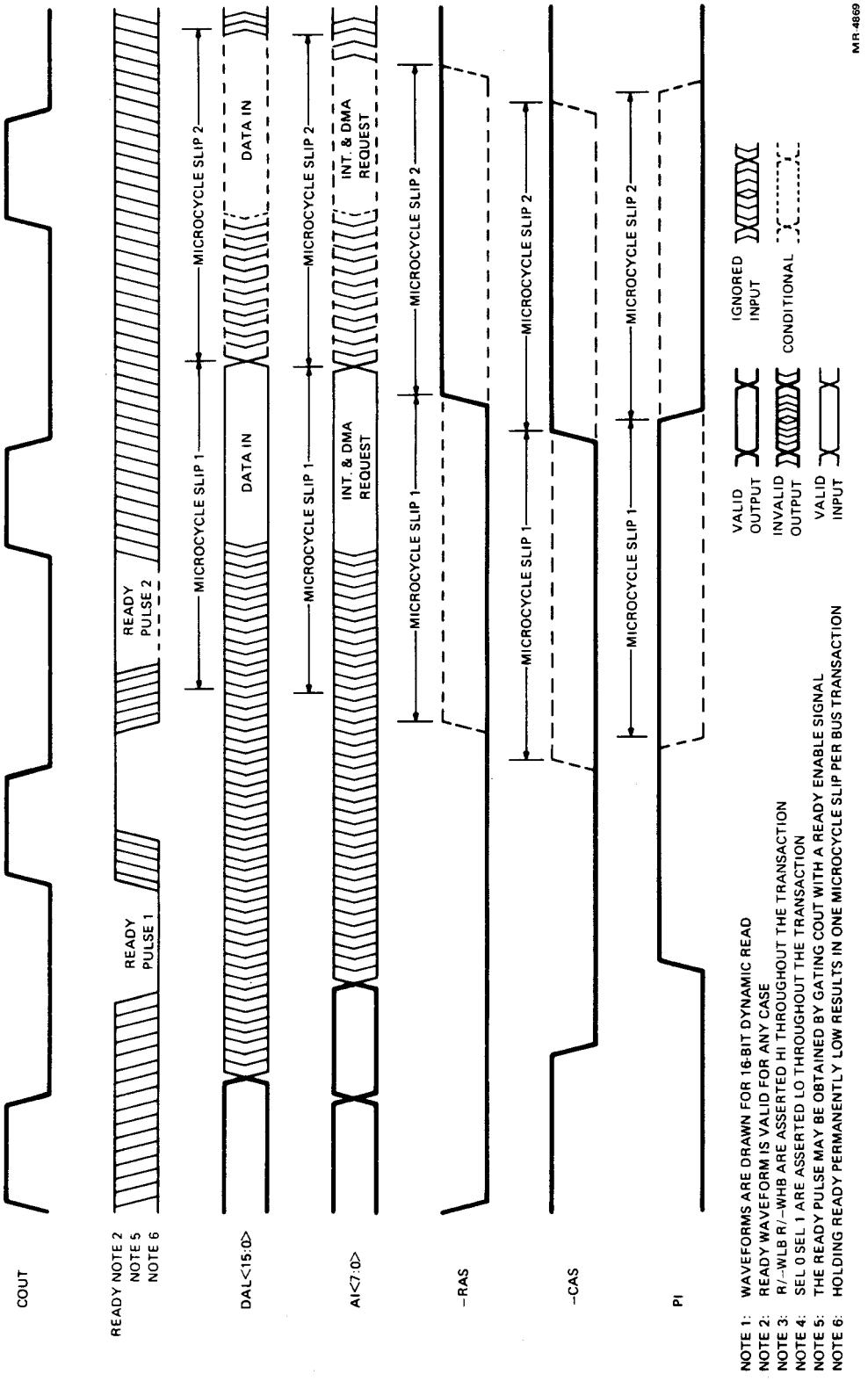


Figure 3-3 READY Timing

The READY signal extends the following transactions.

- Read
- Write
- IACK
- DMA

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

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. The −BCLR signal asserted (low) enables the mode register pull-ups on DAL<15:8,1:0>. The −BCLR pin must be connected to ground through a 1K Ω, 1% resistor. The signal's characterisitics are given in Table A-2 in Appendix A.

3.5.2 PUP (Power-Up)

PUP is a Schmitt-triggered input having 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 DALs 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 series of events that initializes the DCT11-AA. The power-up sequence occurs in two cases.

1. When V_{CC} is applied:
   - PUP changes state (low to high).
   - The −BCLR output is asserted.
   - PUP changes state (high to low).
   - The mode register is loaded.
   - The −BCLR output is cleared.
   - 20 refresh transactions (8-bit dynamic) and 10 refresh transactions (16-bit dynamic) or 20 busnop transactions (8-bit static) and 10 busnop transactions (16-bit dynamic) occur.
   - The stack pointer is loaded to 376_8, the program counter is loaded to the start address, and the processor status word is loaded to 340_8.
   - An ASPI transaction occurs.

2. When a reset instruction is executed:

   - The −BCLR output is asserted.
   - The mode register is loaded.
   - The −BCLR output is cleared.
   - An ASPI transaction occurs.

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

3.5.2.1 Power-Up (PUP) Input – Refer 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} = 0.8 V). When the processor detects a change from high back to low, the mode register load begins.

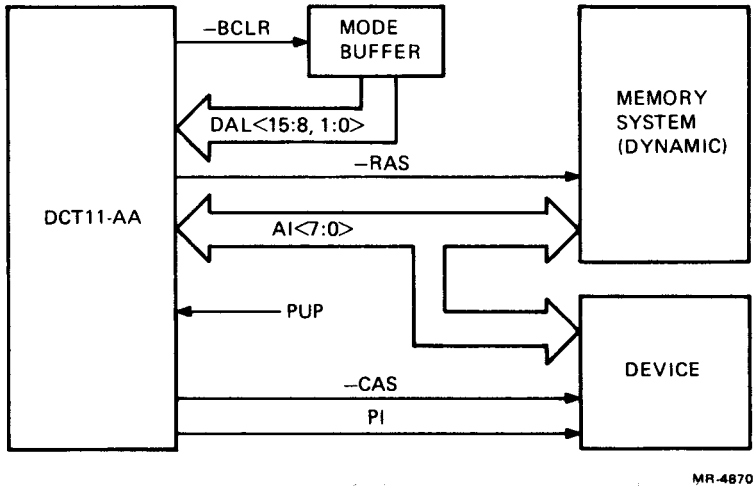


Figure 3-4 Power-Up Sequence, Block Diagram

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. The −BCLR signal is also asserted as a result of a program reset instruction. The −BCLR signal is a strobe used by the user to enable pull-downs on data address lines (DALs) <15:8>,<1:0> at mode register read time. The mode register is loaded through DAL<15:0>. However, DAL<7:2> are reserved. The −BCLR signal 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 Transaction – Depending on the condition of the mode register the processor generates either refresh or busnop transactions. Refer to Table 3-8 for the conditions and the number of transactions generated.

3.5.2.5 Loading the SP, PC, and PSW – After the completion of the refresh or busnop transactions the processor loads the stack pointer (SP) with 376_8. The program counter (PC) is loaded with the start address and, finally, the processor status word (PSW) is loaded with 340_8.

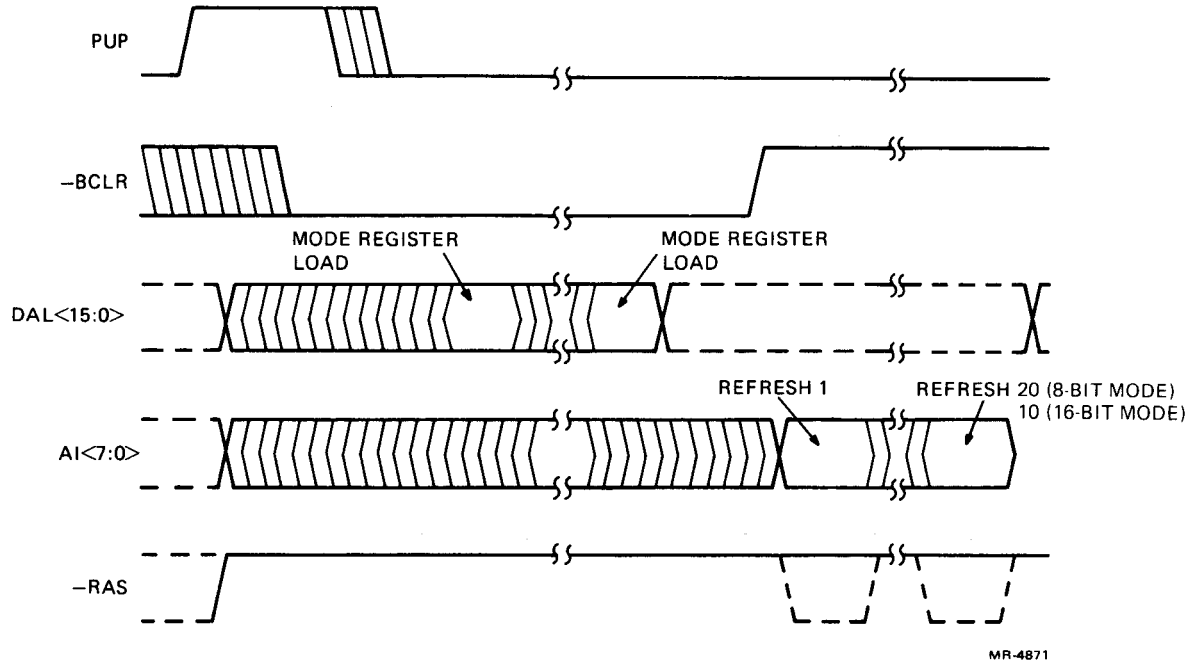


Figure 3-5 Power-Up Sequence Timing

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	

3.5.2.6 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 that 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). Refer 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 in Appendix A.

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 in Appendix A.

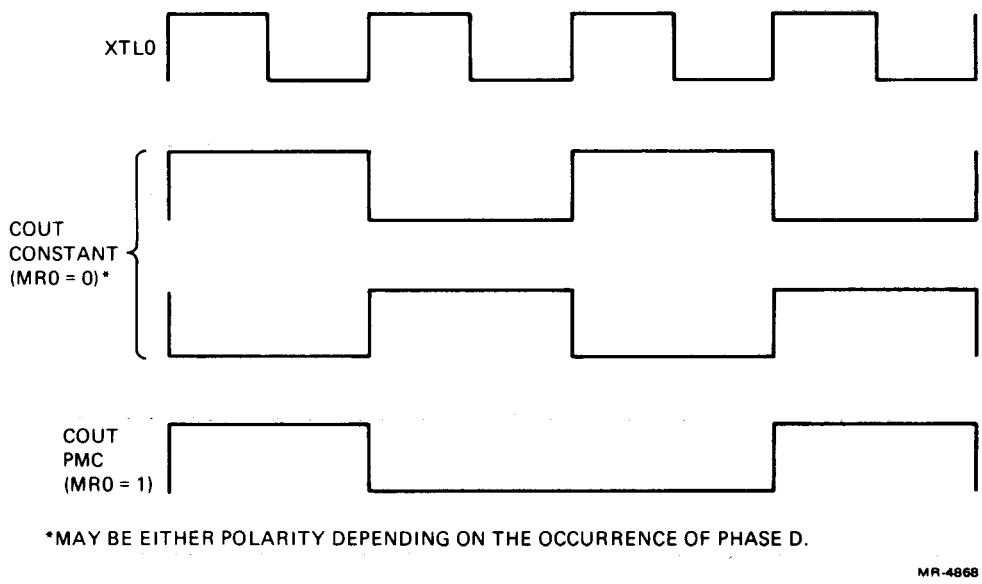


Figure 3-6 COUT Timing

3.6 POWER PINS

The following are pins associated with the power source of 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 V supply for the DCT11-AA. This voltage must be maintained to within ±5% of 5 V.

CHAPTER 4
MODE SELECTION

4.1 INTRODUCTION

Most 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.

The four sections of this chapter describe:

- Modes related to function
- Modes related to timing
- Mode register bit settings
- Mode register selection guidelines

4.2 MODES RELATED TO FUNCTION

Refer 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>

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).

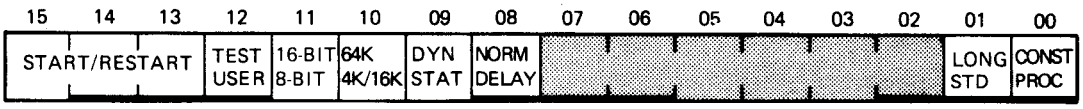
Data address lines:

DAL<15:0> – Output of the 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 separate write control signal (R/−WHB and R/−WLB).



<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

4.2.1.2 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 the 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>); they hold the high byte of the address throughout the two transactions.

DAL<7:0> - Contains the low byte of the address during the read/write time of the first transaction and the data during the read/write time of the second transaction.

Read/write control:

A separate 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). These signals are mutually exclusive.

4.2.2 Dynamic or Static Mode (MR<9>)

Mode register bit 9 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 occurrence of the refresh transaction (which adds time to the instruction execution time).

4.2.2.1 Dynamic Mode – If mode register bit 9 is asserted low (MR<9> = 0), dynamic mode is selected and dynamic memories are directly supported. Besides outputting the address on DAL<15:0> before the assertion of −RAS, the DCT11-AA also outputs row and column addresses on 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 9 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 has no effect) and is used for selecting the dynamic memory chip type. In 64K mode (MR<10> = 0), memory chips such as 64K × 1-bit are supported.

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

4.2.4 Tester or User Mode (MR<12>)

Tester mode is for Digital Equipment Corporation's use only. If mode register bit 12 is high (MR<12> = 1), user mode is selected.

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

4.2.5 Start and Restart Address (MR<15:13>)

Mode register bits 15–13 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 following modes related to timing affect the timing of the processor but not its functionality.

- 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 0 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 0 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 1 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 that 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 1 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 8 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 8 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

The mode register is set during the power-up sequence, or when the reset instruction is executed. At either of these times 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 DALs to load the desired mode in the mode register. The data on the DALs must be stable throughout 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 below presume the DCT11-AA user has one or more of the following goals in mind.

- Minimum cost
- Maximum speed
- Minimum size (chip count)
- Minimum development time

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

4.5.1 Minimum 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 × 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 Maximum 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 rather than in two (8-bit mode). The 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 saving for computational code is possible.

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

4.5.3 Mimimum 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 × 1 organization) uses eight chips.

4.5.3.2 Static Mode – Static mode can take advantage of n × 4 and n × 8 static RAMs in order to minimize chip count.

4.5.4 Minimum 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. Also, 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 possible questions, but offers a few examples and solutions that will enable the reader to get started. Interfacing information is presented on 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

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

- $C \geqslant 0.04\ \mu F$
- $C\ (\mu F) \geqslant 0.05\ t_R\ (ms)$

where 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.

5.3 LOADING THE MODE REGISTER

Figure 5-2 shows 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 (DALs). While −BCLR is asserted the DALs 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 DALs. Unasserted bits may be left floating since they are pulled up internally by the DCT11-AA when −BCLR is low. The −BCLR signal is buffered to provide enough drive for system initialization. All

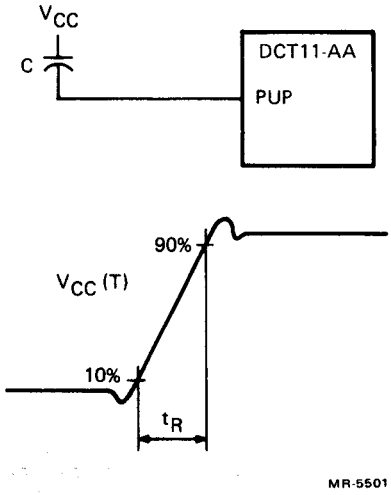


Figure 5-1 Power-Up Circuit

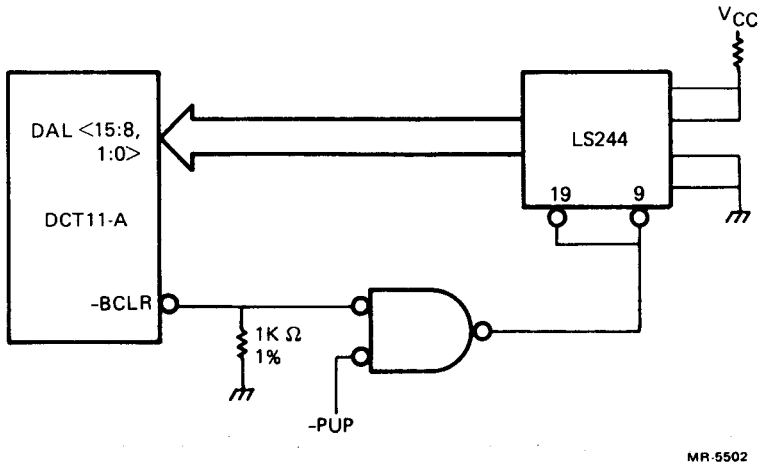


Figure 5-2 Mode Register Loading

devices in the system (except the buffer containing data for the mode register) should three-state their outputs connected to DAL<15:0> at −BCLR time. This is done to prevent the mode register from being loaded with questionable data.

NOTE

The pull-down resistor on −BCLR must be 1K Ω @ 1% to guarantee timing specifications. The −BCLR signal can sink up to 3.2 mA and source 80 μA (can drive two TTL loads in addition to the 1K Ω load).

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 wide-band amplifier. Refer 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)

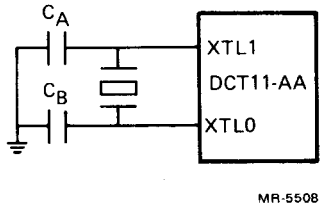


Figure 5-3 Crystal Oscillator Clock

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 ($\leqslant$ 7.5 MHz) 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 30 pF (which is the nominal load for most crystals), thus pulling it off frequency.

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

Crystal:

Cut at fundamental (At)
Load 30 pF
$R_S < 200\ \Omega \div$ fMHz (fundamental; i.e., 27 Ω @ 7.5 MHz)
$R_S > 4000\ \Omega \div$ fMHz (spurious)
$R_S > 400\ \Omega \div$ fMHz (harmonic)
$C_O < 4$ pF

Capacitors:

Type mica
Nominal value (± 10%)
C_A (XTL0) 500 pF ÷ fMHz (example: 67 pF @ 7.5 MHz)
C_B (XTL1) 200 pF ÷ fMHz (example: 27 pF @ 7.5 MHz)

These specifications guarantee against third harmonic and spurious start-ups. If such guarantees are not necessary and only a steady oscillation is required, most crystals can be used. Detailed timing of XTAL is found in Figure A-17 in Appendix A.

5.4.2 TTL Oscillator-Based Clock

Refer to Figures 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 > 133 ns
- Rise time t_R < 80 ns
- Fall time t_F < 80 ns
- Low time t_L > 60 ns
- High time t_H > 60 ns

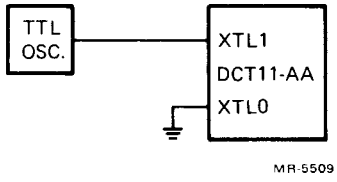


Figure 5-4 TTL Oscillator Clock

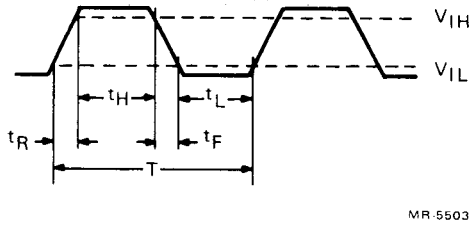


Figure 5-5 TTL Oscillator Waveform

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

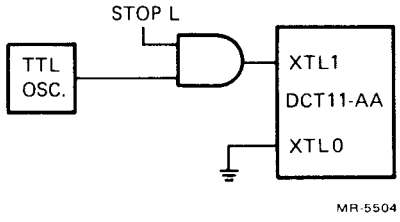


Figure 5-6 Gating XTL1

5.5 ADDRESS LATCH AND DECODE

Refer 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 the row address strobe (−RAS) leading edge.

Refer 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.

Address decoding can be done in a number of traditional ways. In Figures 5-7 and 5-8 PLAs 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. *The CAS signal should be ANDed with RAS to prevent the enabling of strobes during an ASPI transaction.*

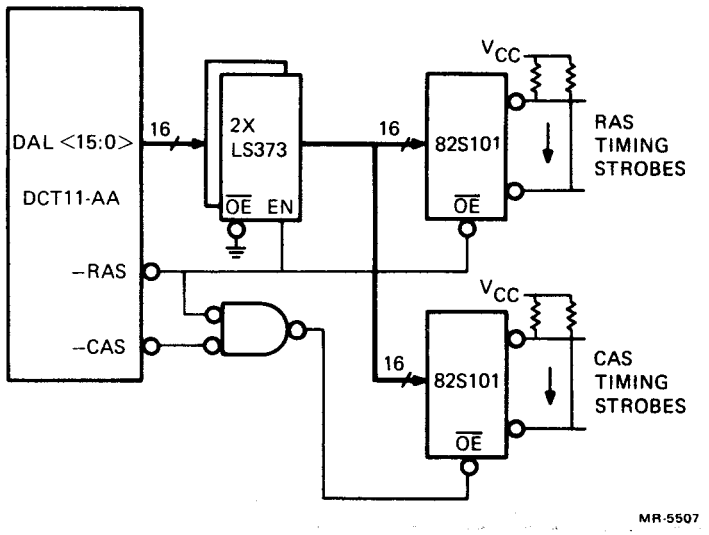


Figure 5-7 16-Bit Address Latch and Decode

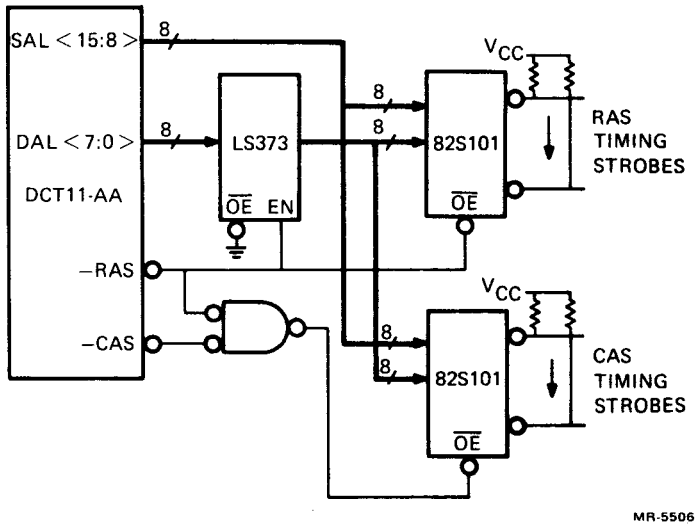


Figure 5-8 8-Bit Address Latch and Decode

5.6 MEMORY SUBSYSTEMS

Two examples of memory subsystems are described below, 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 shown in Figure 5-9. The memory map is shown in Figure 5-10. The address is latched in the LS373 chips by −RAS. The address is then decoded in the LS138 into eight sectors in the upper half of the memory. The sector 140000 to 147776

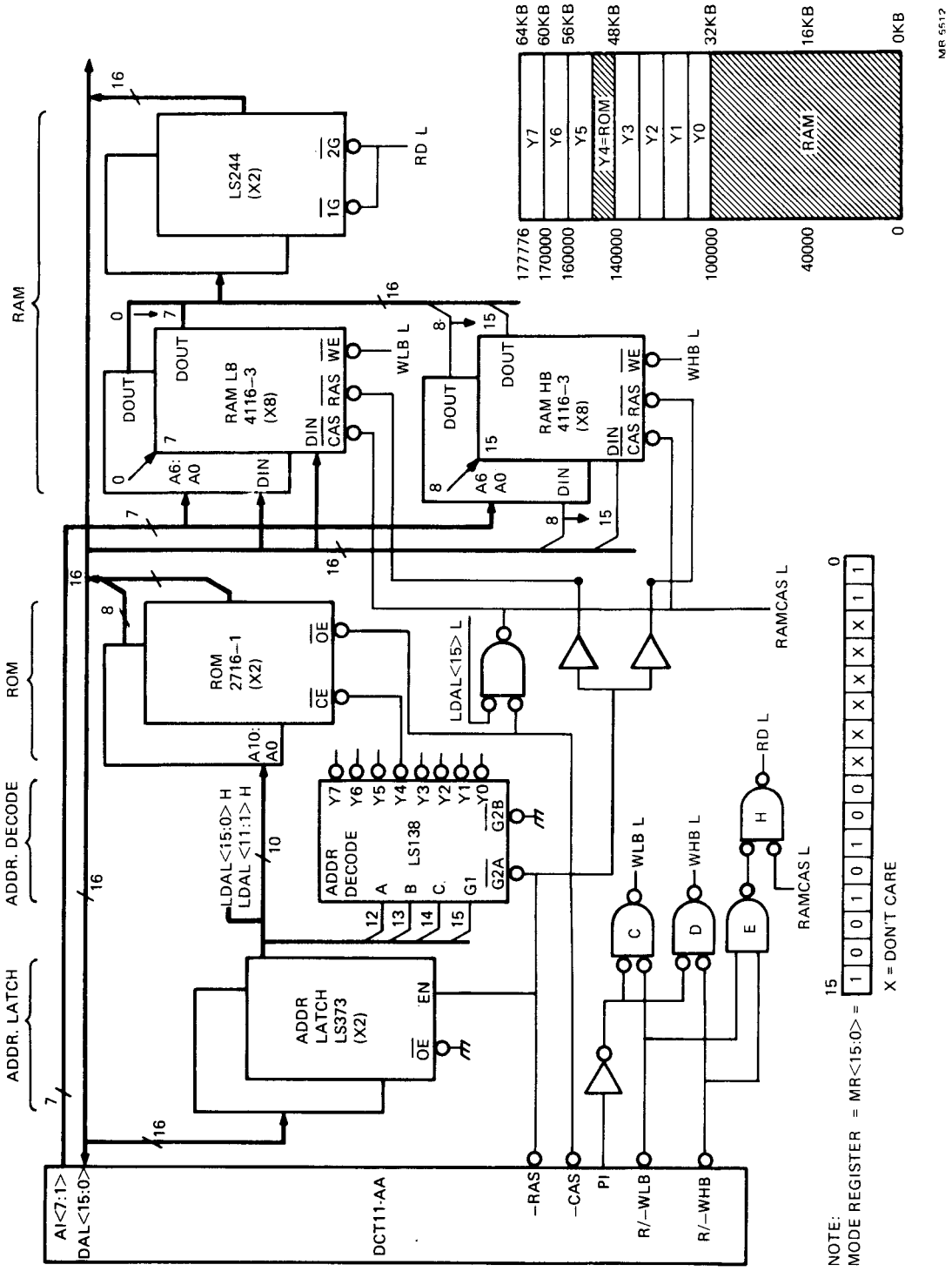


Figure 5-9 16-Bit ROM (4K) and Dynamic RAM (32K) Subsystem

maps into the ROM. This is implemented by selecting the ROM (−CE) with the output Y4 and selecting −OE with −CAS (refer to Figure 5-10). The fast variation ROM (2716-1) must be used, if running at more than 6.9 MHz, in order to meet the DCT11-AA specification of t_{RRD} (405 ns at 7.5 MHz).

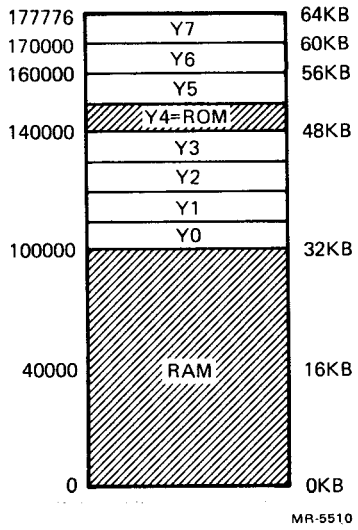


Figure 5-10 16-Bit System Memory Map

The RAM is made of high-byte and low-byte sections that have everything in common except the −WE strobe. The whole RAM is chip selected by controlling −CAS and sending −RAS to all chips at all times. Using −CAS as chip select has the advantage of refreshing a row at every occurrence of −RAS.

Although −CAS drives 160 pF (10 pF per RAM chip), it does not require buffering. The DCT11-AA output timings are specified for a purely capacitive load of 80 pF. For loading other than 80 pF use the following correction factors.

- 80 pF < CL < 200 pF +0.3 ns/pF
- 25 pF < CL < 80 pF −0.3 ns/pF

This results in a −CAS delay of 80 × 0.3 = 24 ns with respect to the nominal timing specifications. Such a delay still meets the RAM chip specifications for −RAS and −CAS hold time (55 ns minimum for 4116-3). Refer to Figure 5-11.

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

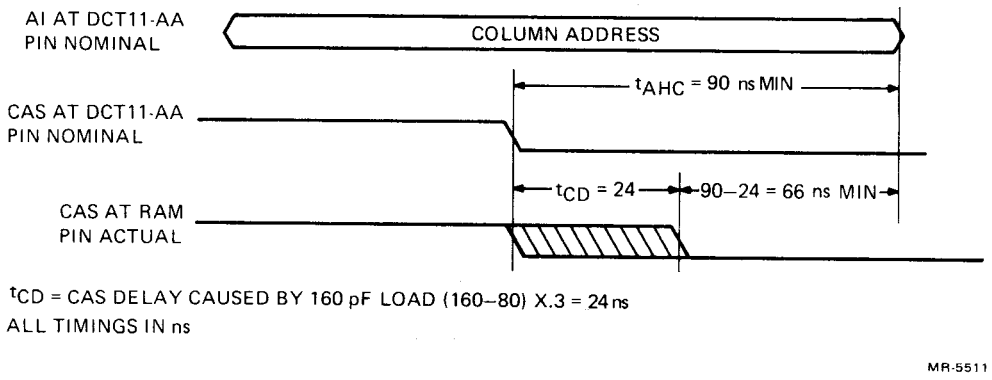


Figure 5-11 Column Address Setup and Hold-Time Calculations

The buffer is enabled only when the DCT11-AA is performing a read from RAM (signal RD L). Tables 5-1 and 5-2 list the control signals and the states of the data bus for each transaction, respectively.

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.

5.6.2 8-Bit Mode Memory System

Refer to Figure 5-12. Since the data bus is 8 bits wide and the memory organization is different from that of a 16-bit system, 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.

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 only).
* – Lines driven after address portion of transaction (8-bit and 16-bit modes).
1 – Dynamic modes only.
3S – Three-state.

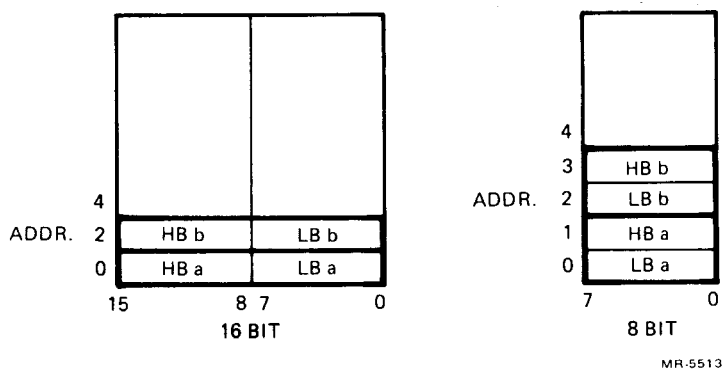


Figure 5-12 16-Bit/8-Bit Memory Organization

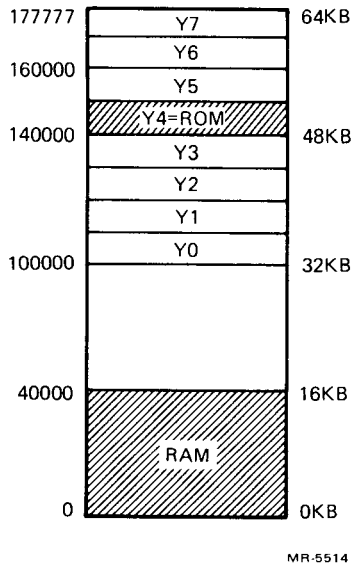


Figure 5-13 8-Bit System Memory Map

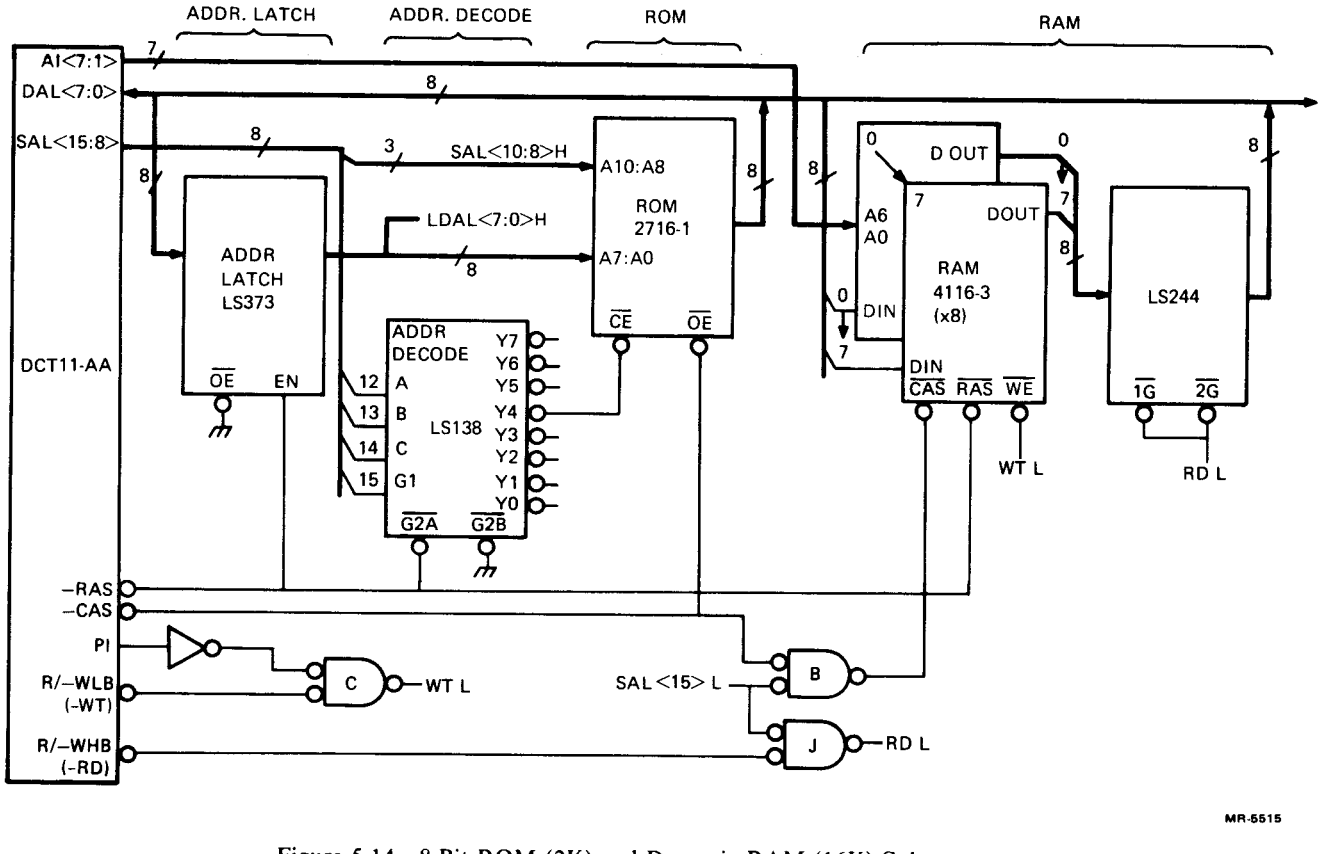


Figure 5-14 8-Bit ROM (2K) and Dynamic RAM (16K) Subsystem

5.7 INTERRUPTS

The examples of interrupts 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

Refer to Figure 5-15. 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} = 105 ns @ 7.5 MHz) is long enough to settle any metastable states on the output of the flip-flop.

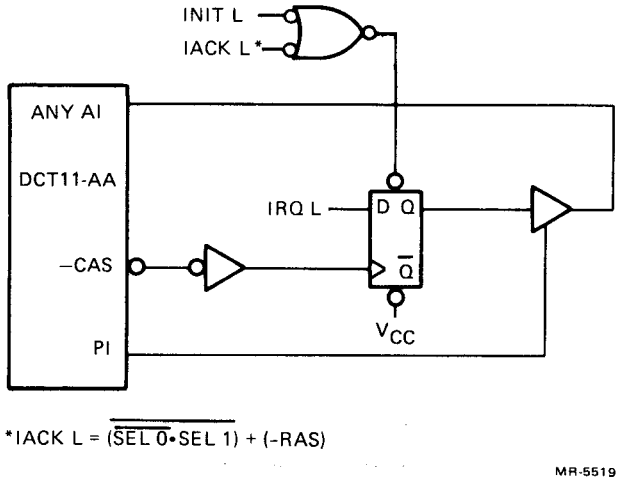


Figure 5-15 General Interrupt

5.7.2 Decoding IACK Information

Figure 5-16 shows an example of how to decode IACK information for 15 CP devices. An LS138 can be used instead of an 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 shows an example of a complete interrupt system (interrupt request and interrupt acknowledge) for six CP devices.

5.7.3 External Vectors

If −VEC (AI<5>) is asserted (low) during the interrupt request, the DCT11-AA obtains the vector on DAL<7:2> from the device during the IACK transaction. Figure 5-18 shows an example of such a circuit.

5.7.4 Using a Priority Encoder Chip

Refer to Figure 5-19. Six devices can generate an interrupt using the internal vectors of the DCT11-AA.

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 are distinguished from one another by different external vectors during the IACK transaction.

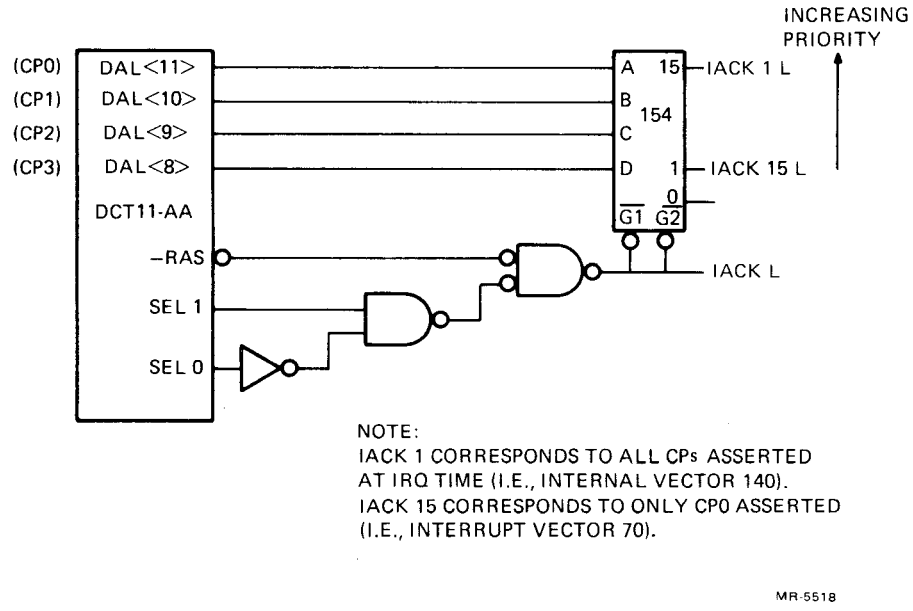
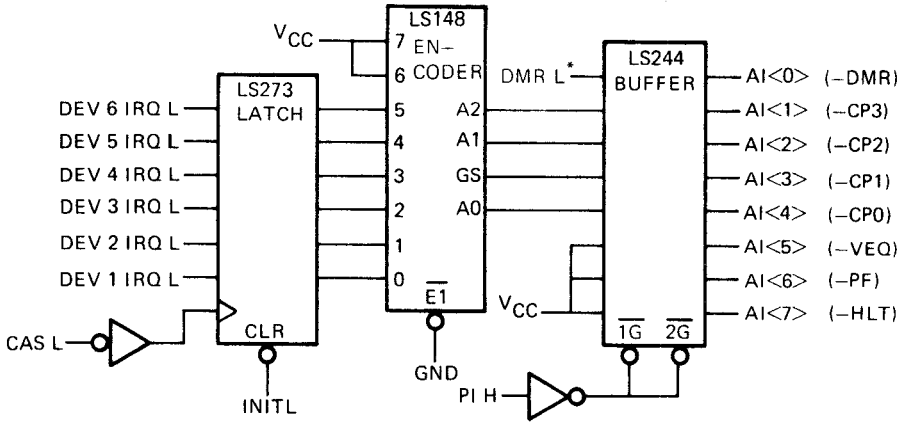


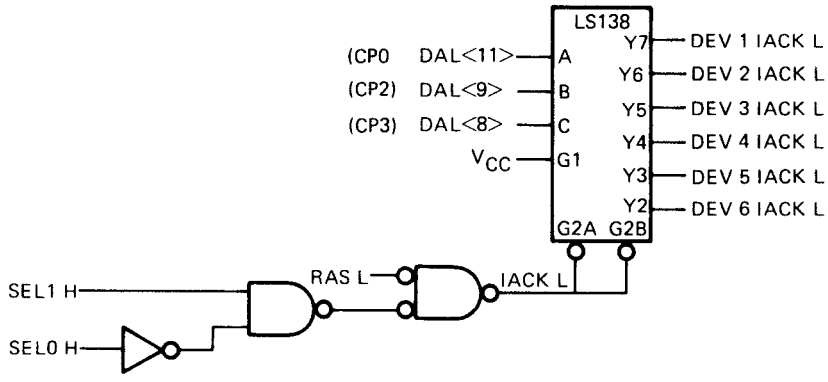
Figure 5-16 Decoding IACK Information for 16 CP Devices

INTERRUPT REQUEST CIRCUIT



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														
	AI4	AI3	AI2	AI1													INTERNAL VECTORS	
	L	L	H	L	H	H	H	H	H	L	L	H	H	H	H	H	DEV6	100
	H	L	H	L	H	H	H	H	L	H	H	L	H	H	H	H	DEV5	104
	L	L	L	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

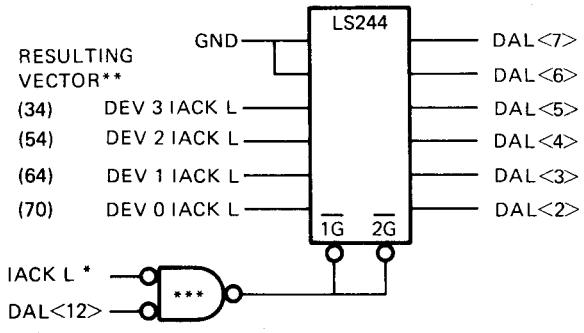
IACK CIRCUIT



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

MR 5520

Figure 5-17 Interrupt System



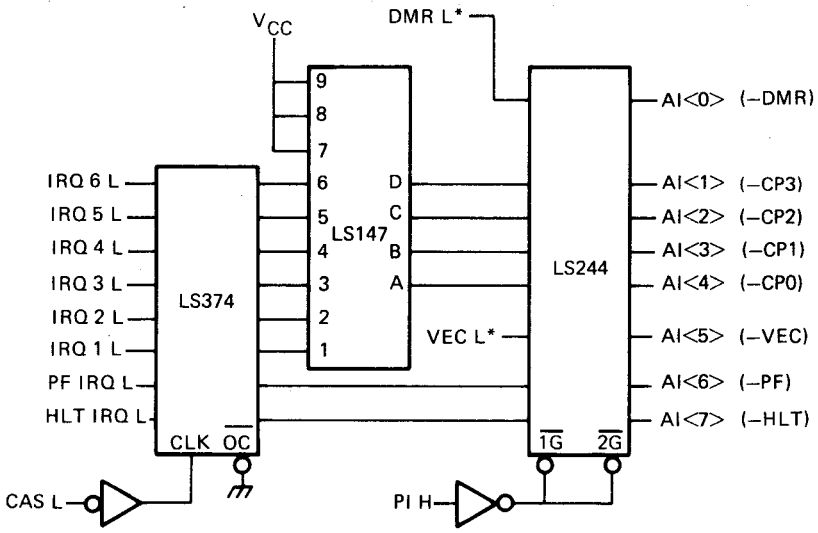
*IACK L = $\overline{(\overline{SEL\ 0} \cdot SEL\ 1)}$ + (-RAS)

**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 5-18 Driving an External Vector During IACK



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

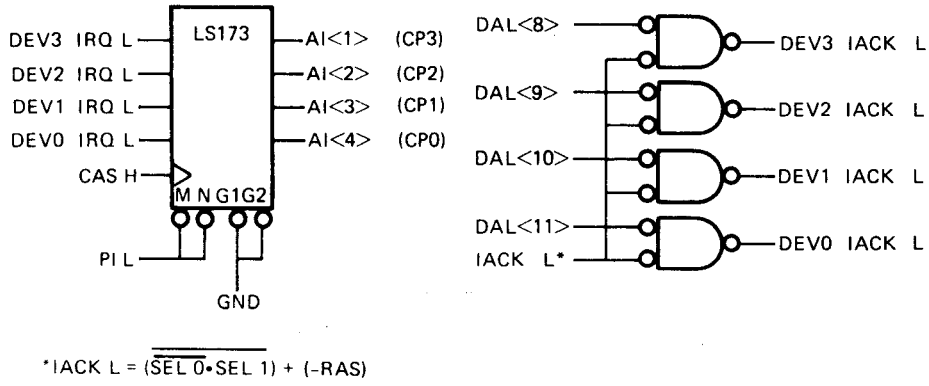
–CP<3> (AI<1>)	–CP<2> (AI<2>)	–CP<1> (AI<3>)	–CP<0> (AI<4>)	PRIORITY LEVEL	INTERNAL VECTOR ADDRESS	DEVICES IMPLEMENTED
X	X	X	X	8	–	HLT IRQ
X	X	X	X	8	24	PF IRQ
L	L	L	L	7	140	
L	L	L	H	7	144	
L	L	H	L	7	150	
L	L	H	H	7	154	
L	H	L	L	6	100	
L	H	L	H	6	104	
L	H	H	L	6	110	
L	H	H	H	6	114	
H	L	L	L	5	120	
H	L	L	H	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 5-19 Interrupt Request Circuit (Priority Encoder)

5.7.5 Direct CP Encoding

Direct CP encoding (refer to Figure 5-20) can be accomplished only when there are four 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 140–154 and 100–114 must be loaded with the vector address relative to D3. Locations 120–134 must be loaded with the vector address relative to D2. Locations 60 and 64 must be loaded with the 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. Refer again to Figure 5-20. A higher device IACK will clear a lower device interrupt request before the request is serviced.



*IACK L = $\overline{(\overline{SEL\ 0} \cdot SEL\ 1)}$ + (-RAS)

–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	DEVICE 3
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	DEVICE 2
H	L	L	H	5	124	
H	L	H	L	5	130	
H	L	H	H	5	134	
H	H	L	L	4	60	DEVICE 1
H	H	L	H	4	64	
H	H	H	L	4	70	DEVICE 0
H	H	H	H	NO ACTION		

MR-5523

Figure 5-20 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.

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

1. A DMA request (DMR) to the DCT11-AA is made by driving AI<0> low during PI.

2. The request is latched into the DCT11-AA during PI and shortly thereafter a DMA grant is issued.

The maximum time from the request's origination to the grant's issuance is a function of the DCT11-AA mode. This time is specified in Table A-22 in Appendix A. When the grant is issued the DCT11-AA takes the following actions.

- SEL0 and SEL1 become 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 A, Figure A-13).
- The DALs are three-stated.
- AI<7:0>, R/−WHB, and R/−WLB have low-current pull-ups.

When the grant is issued, external circuitry must drive the R/−WHB and R/−WLB lines, and initially drive the DALs 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 DALs 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 current sequence of DMA transactions finishes and the DCT11-AA resumes normal operation.

5.8.1 Single-Channel DMA Controller (16-Bit Mode)

This section describes a single-channel DMA controller for use with dynamic or static memory systems. Refer to Figure 5-21.

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–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) – Refer to Figure 5-21. The 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. In a static memory system 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 the DC006s (DEC DMA chip)
S-C L	Select word counter in the DC006s
DMACR L	DMACR (DMA control register) is an optional register that 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 DC006s with the 2's complement of the word count and then with the bus address. After the loading the peripheral is signaled that the DMA setup is complete.

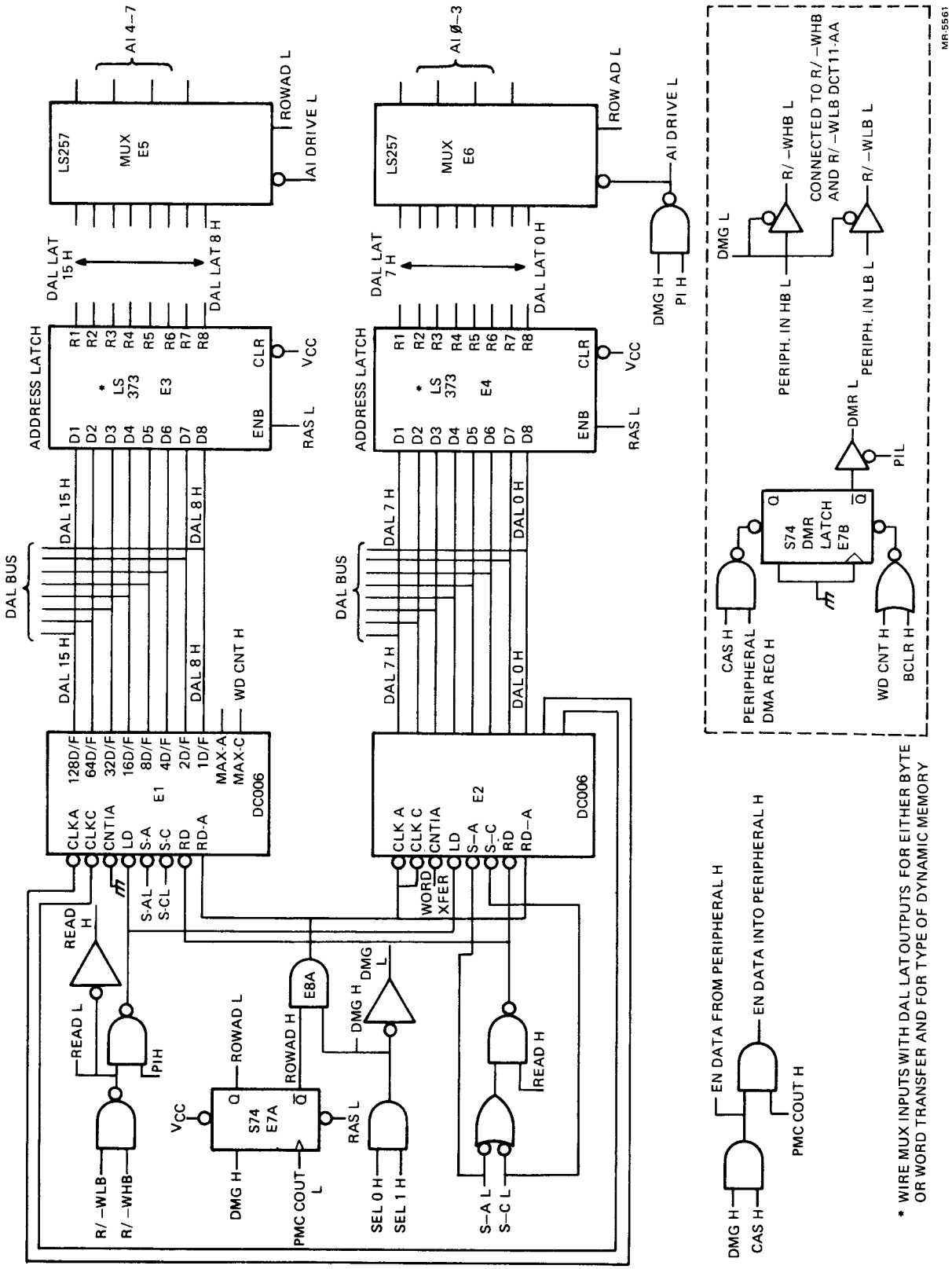


Figure 5-21 Single-Channel DMA

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 DC006s. The DC006s drive the address onto the DALs, where it is input by the address latches. These in turn drive the AI multiplexer inputs. During this period the AI multiplexer is pointing to the row address inputs.

Between −RAS assertion (low) and −CAS assertion (low) the trailing edge of the PMC COUT signal 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 affects the DC006s in two ways:

- The DC006s' inputs become three-stated, so they stop driving the address onto the DALs.
- 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 increments 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; 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 5-22 for the enclosed section of Figure 5-21 results in the necessary hardware modification.

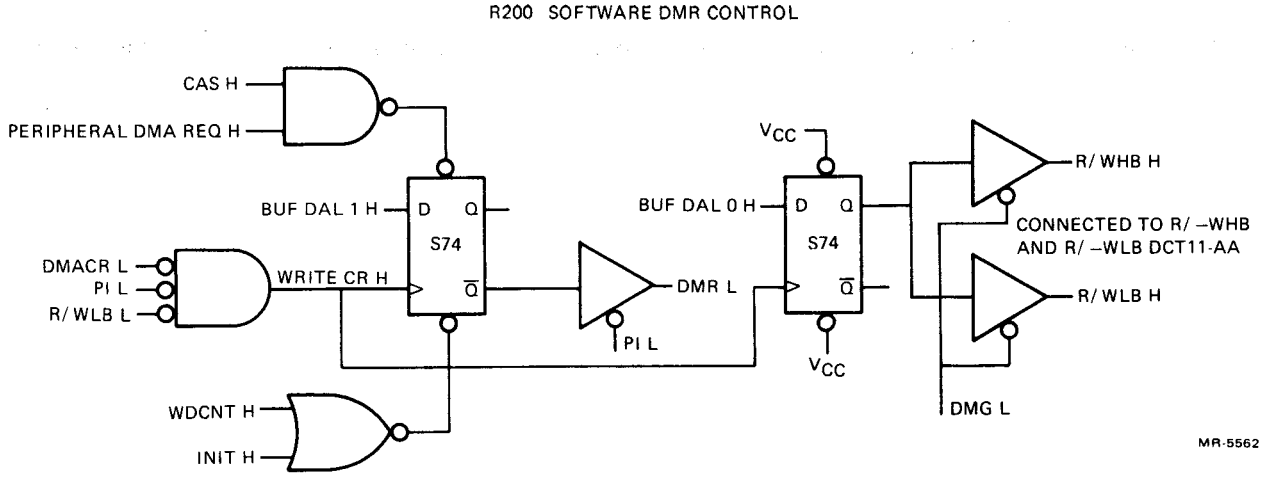


Figure 5-22 Software DMR Control

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.

5.9 WORKING WITH PERIPHERAL CHIPS

Though almost all peripheral chips will work with the DCT11-AA, this section discusses only these three selected ones:

- 8155 RAM, three ports, and timer
- 2651 PUSART
- DC003 interrupt logic

5.9.1 8155 – RAM, Three Ports, and Timer

Refer to Figure 5-23. 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.

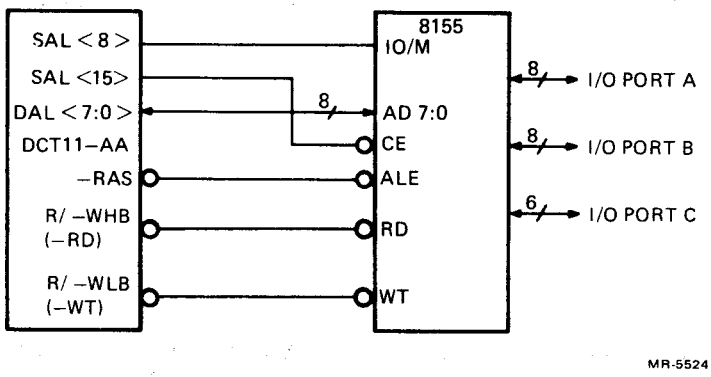


Figure 5-23 8155 RAM

5.9.2 2651 – PUSART

Refer to Figure 5-24. Two facts 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 in stack operations, traps, interrupts, and subroutines).
- The 2651 PUSART's receive data buffer and transmit data buffer have the same address inside the chip. The buffers are selected by the R/W input.

An involuntary read from the receive buffer clears the receive ready pin, and may result in the resetting of 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.

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

- To write mode register 1: disable transmit and receive, read the mode register, and write the mode register.
- To write mode register 2: disable transmit and receive, and write the mode register.

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

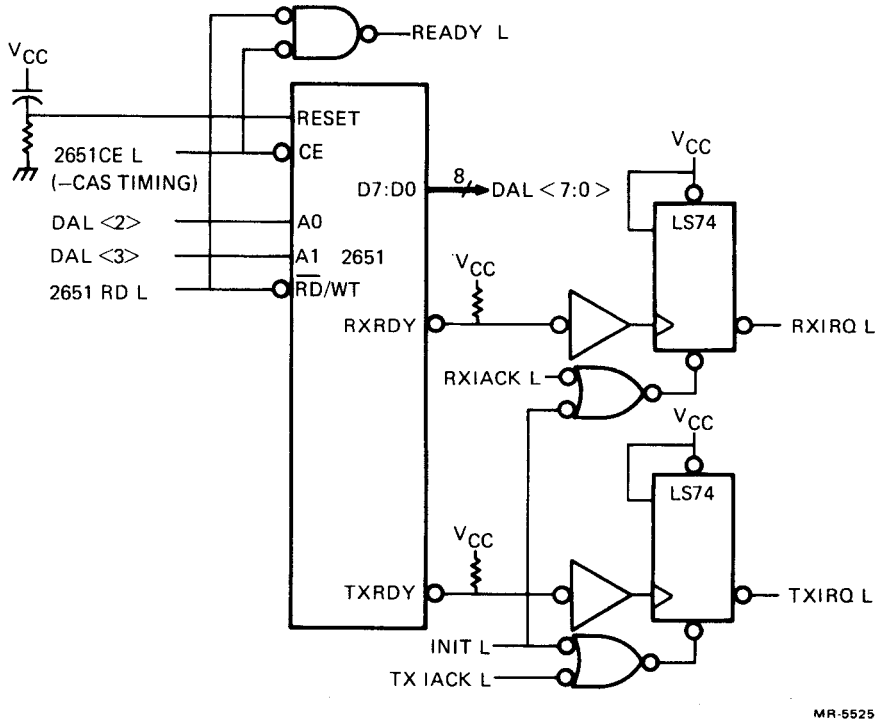


Figure 5-24 2651 PUSART

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

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

5.9.3 DC003 – Interrupt Logic

Refer to Figures 5-25 and 5-26. The interrupt chip is an 18-pin, dual in-line package device that provides the circuits for handling 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. DC003s may be daisy-chained at any priority level.

Daisy-chaining multiple DC003s may cause an error condition. If the requesting device receives the signal IAK L too late, it will not be able to assert its vector during the IACK transaction.

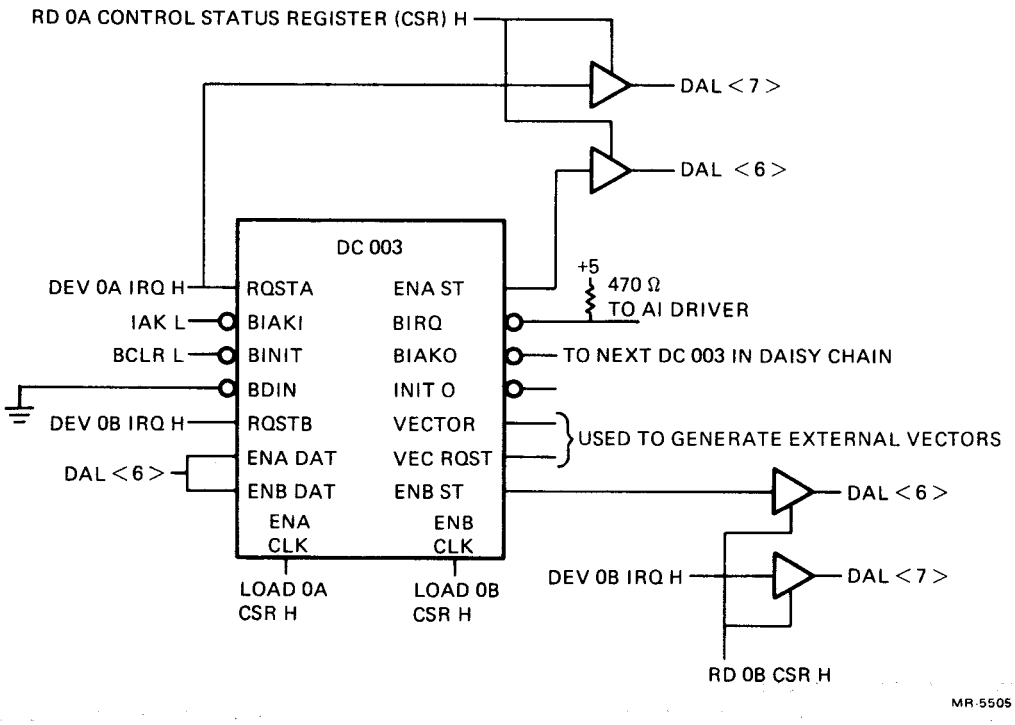
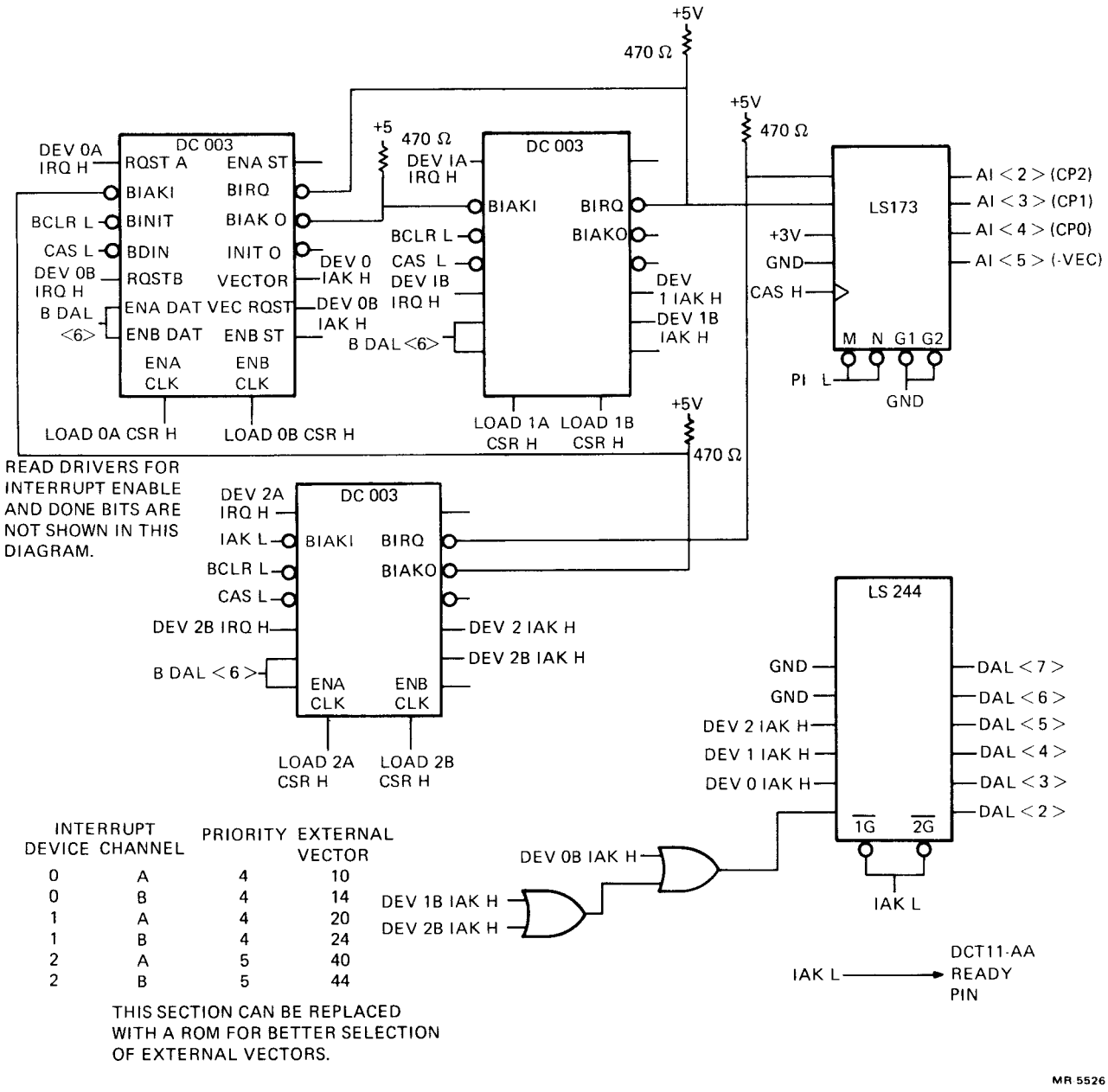


Figure 5-25 DC003 Interrupt Logic



INTERRUPT DEVICE	CHANNEL	PRIORITY	EXTERNAL VECTOR
0	A	4	10
0	B	4	14
1	A	4	20
1	B	4	24
2	A	5	40
2	B	5	44

Figure 5-26 DC003 at Different Priority Levels

CHAPTER 6
ADDRESSING MODES AND INSTRUCTION SET

6.1 INTRODUCTION

This chapter is divided into six major sections:

- **Single-Operand Addressing** – One part of the instruction word specifies the registers; the other part provides information for locating the operand.
- **Double-Operand Addressing** – One 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 different 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** – General-purpose registers 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 is to be used when locating the destination operand.
- An addressing mode (for specifying how the selected register(s) is/are to be used).

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

A general-purpose register may be used with an instruction in any of the following ways.

- As an accumulator – The data to be manipulated resides in the register.
- As a pointer – The contents of the register is the address of an operand, rather than the operand itself.
- As a pointer that automatically steps 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 an index register – 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–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, the 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. These 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, the stack pointer 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 shown in Figure 6-1.

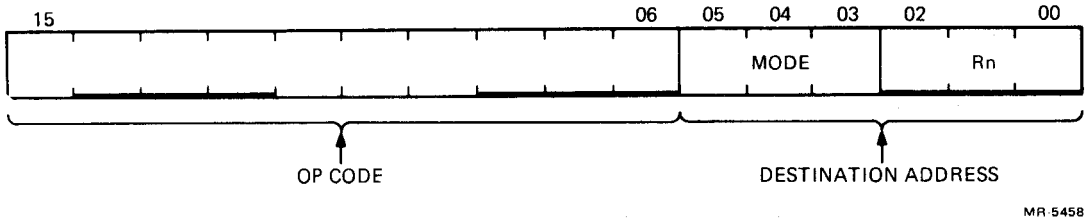


Figure 6-1 Single-Operand Addressing

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

Bits 5–0 form a 6-bit field called the destination address field. The destination address field consists of two subfields:

- Bits 0–2 specify which of the 8 general-purpose registers is to be referenced by this instruction word.
- Bits 3–5 specify how the selected register will be used (in address mode). Bit 3 is set to indicate deferred (indirect) addressing.

6.2.2 Double-Operand Addressing

Operations that 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 shown in Figure 6-2.

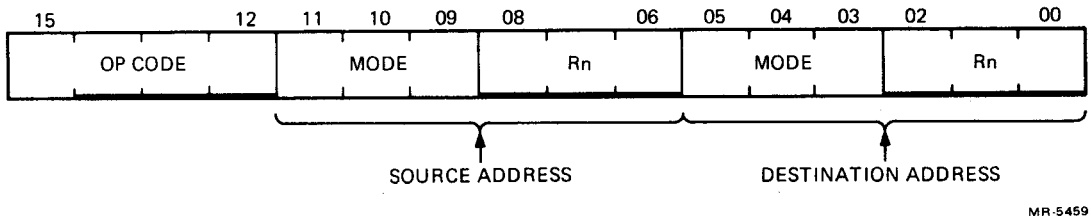


Figure 6-2 Double-Operand Addressing

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 the rest of the chapter use the following sample DCT11-AA instructions. (A complete listing of the DCT11-AA instructions appears 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 the destination.)	0052DD
INCB	Increment byte. (Add one to the contents of the destination byte.)	1052DD
COM	Complement. (Replace the contents of the destination by its logical complement; each 0 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 the source operand to the destination operand and store the result at the destination address.)	06SSDD

DD = destination field (six bits)
SS = source field (six bits)
() = contents of

6.2.3 Direct Addressing

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

Direct Modes (Figures 6-3 to 6-6)

Mode	Name	Assembler Syntax	Function
0	Register	Rn	Register contains operand.

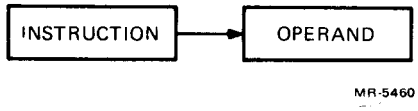


Figure 6-3 Mode 0 Register

Mode	Name	Assembler Syntax	Function
2	Autoincrement	(Rn)+	Register is used as a pointer to sequential data and then incremented.

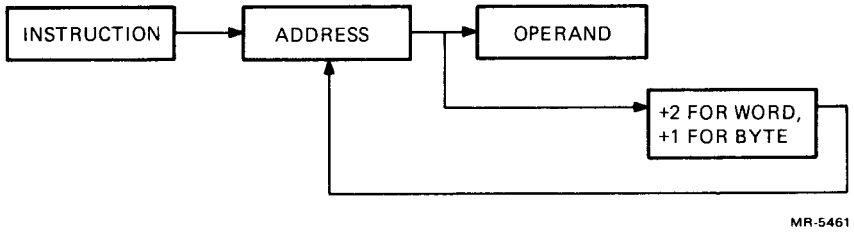


Figure 6-4 Mode 2 Autoincrement

Mode	Name	Assembler Syntax	Function
4	Autodecrement	−(Rn)	Register is decremented and then used as a pointer.

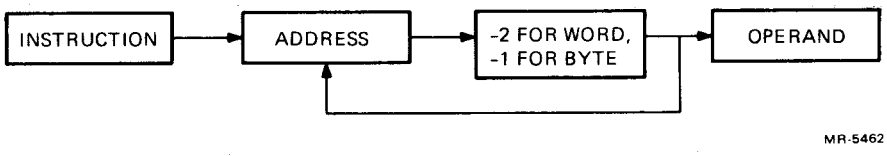


Figure 6-5 Mode 4 Autodecrement

Mode	Name	Assembler Syntax	Function
6	Index	X(Rn)	Value X is added to (Rn) to produce address of operand. Neither X nor (Rn) is modified.

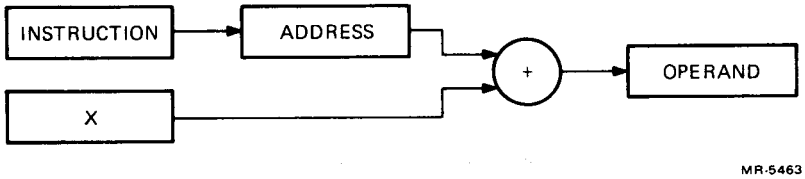


Figure 6-6 Mode 6 Index

6.2.3.1 Register Mode – With register mode any of the general registers may be used as simple accumulators, with the operand 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.

OPR Rn

Register Mode Examples (Figures 6-7 to 6-9)

1.	Symbolic	Octal Code	Instruction Name
	INC R3	005203	Increment

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

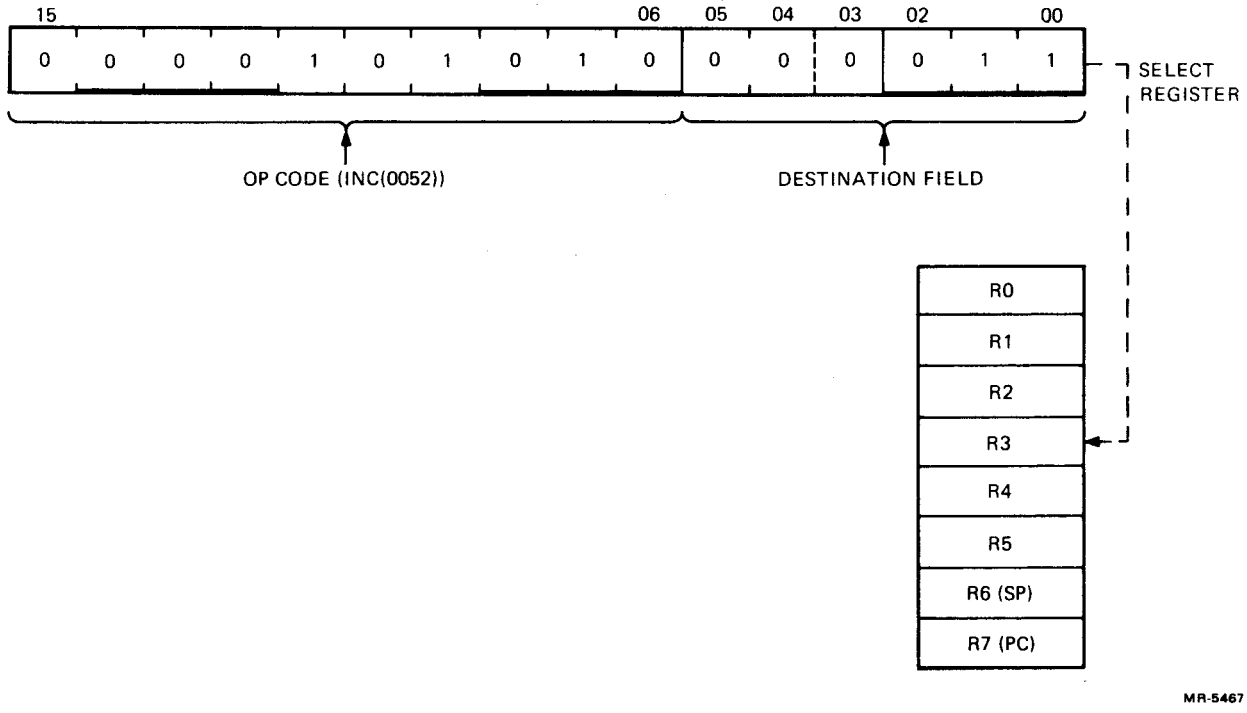


Figure 6-7 INC R3 Increment

2.	Symbolic	Octal Code	Instruction Name
	ADD R2, R4	060204	Add

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

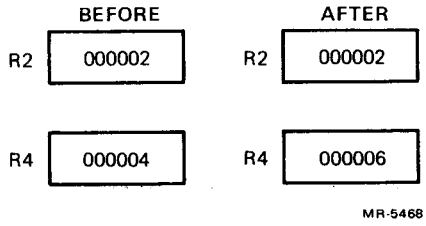


Figure 6-8 ADD R2, R4 Add

3.	Symbolic	Octal Code	Instruction Name
	COMB R4	105104	Complement byte

Operation: 1's complement bits 0–7 (byte) in R4. (When general registers are used, byte instructions operate only on bits 0–7; i.e., byte 0 of the register.)

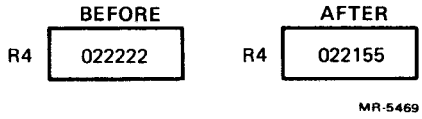


Figure 6-9 COMB R4 Complement Byte

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 (Figures 6-10 to 6-12)

1.	Symbolic	Octal Code	Instruction Name
	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.

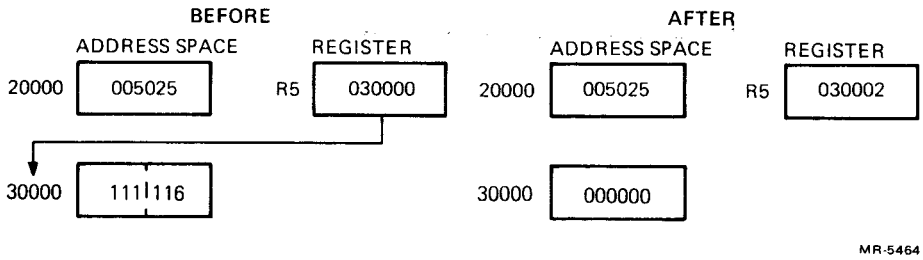


Figure 6-10 CLR (R5)+ Clear

2.	Symbolic	Octal Code	Instruction Name
	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.

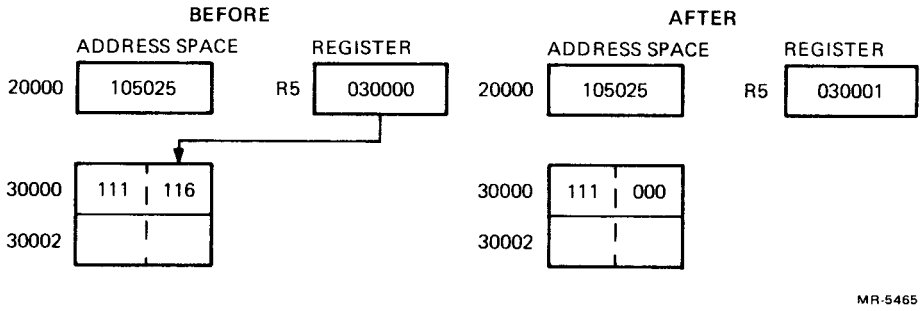


Figure 6-11 CLRB (R5)+ Clear Byte

3.	Symbolic	Octal Code	Instruction Name
	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.

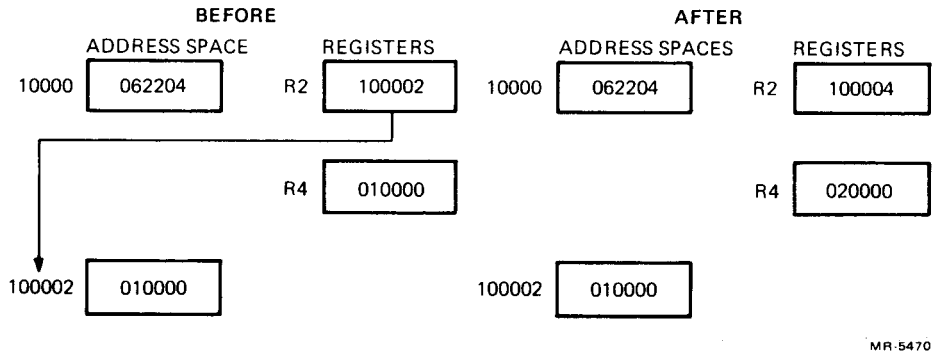


Figure 6-12 ADD (R2)+ R4 Add

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 (Figures 6-13 to 6-15)

1.	Symbolic	Octal Code	Instruction Name
	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.

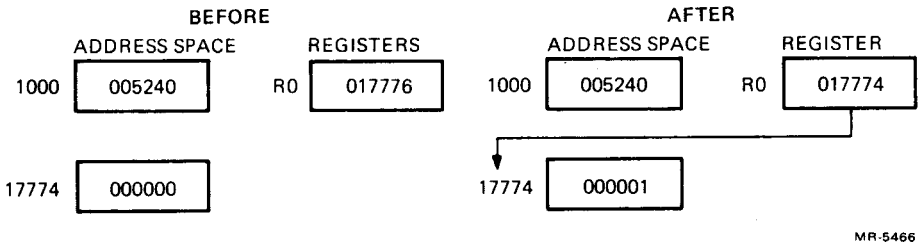


Figure 6-13 INC –(R0) Increment

2.	Symbolic	Octal Code	Instruction Name
	INCB –(R0)	105240	Increment byte

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

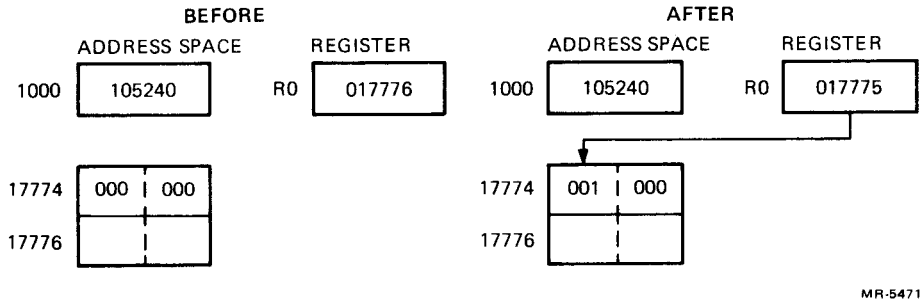


Figure 6-14 INCB –(R0) Increment Byte

3.	Symbolic	Octal Code	Instruction Name
	ADD −(R3),R0	064300	Add

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

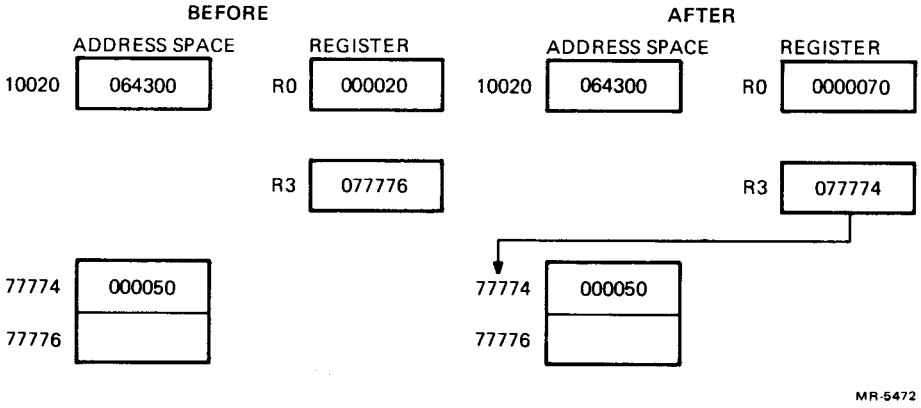


Figure 6-15 ADD −(R3), R0 Add

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 located in the memory location following the instruction word and Rn is the selected general-purpose register.

OPR X(Rn)

Index Mode Examples (Figures 6-16 to 6-18)

1.	Symbolic	Octal Code	Instruction Name
	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.

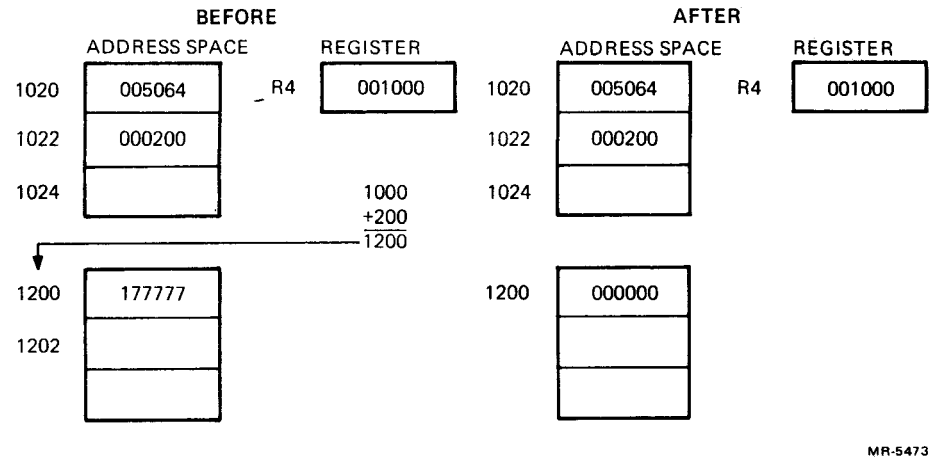


Figure 6-16 CLR 200 (R4) Clear

2.	Symbolic	Octal Code	Instruction Name
	COMB 200(R1)	105161 000200	Complement byte

Operation: The contents of a location, which are determined by adding 200 to the contents of R1, are 1's complemented (i.e., logically complemented).

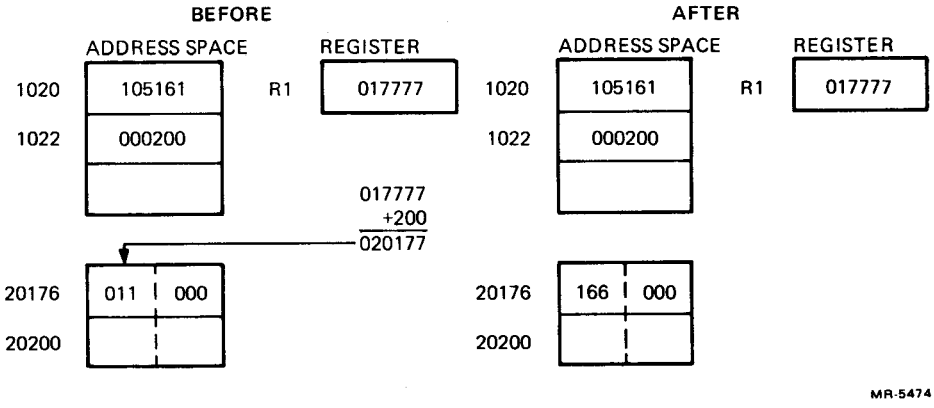


Figure 6-17 COMB 200 (R1) Complement Byte

3.	Symbolic	Octal Code	Instruction Name
	ADD 30(R2),20(R5)	066265 000030 000020	Add

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

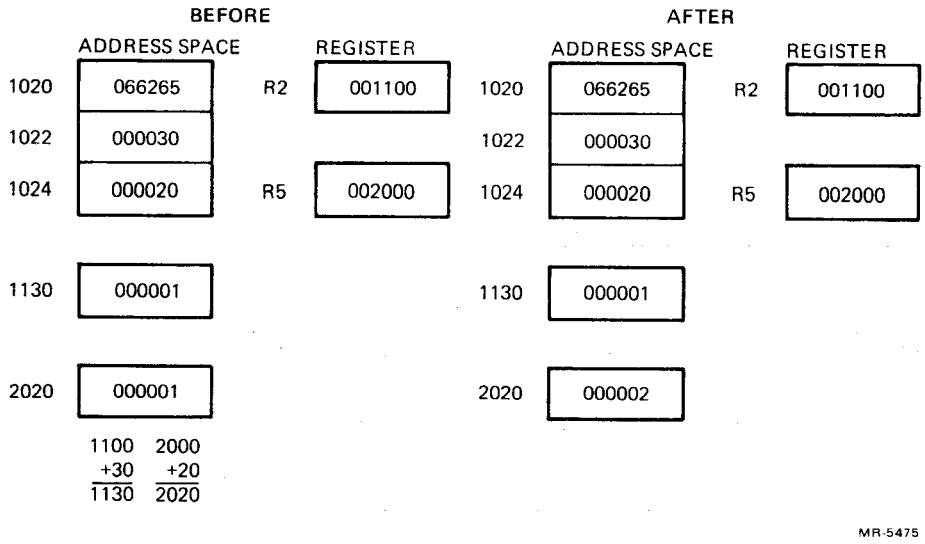


Figure 6-18 ADD 30 (R2), 20 (R5) Add

6.2.4 Deferred (Indirect) Addressing

The four basic modes may also be used with deferred addressing. Whereas in register mode the operand is the contents of the selected register, in 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. The assembler syntax for indicating deferred addressing is @ [or () when this is not ambiguous]. The following summarizes the deferred versions of the basic modes.

Deferred Modes (Figures 6-19 to 6-22)

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

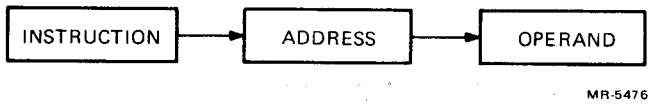


Figure 6-19 Mode 1 Register-Deferred

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

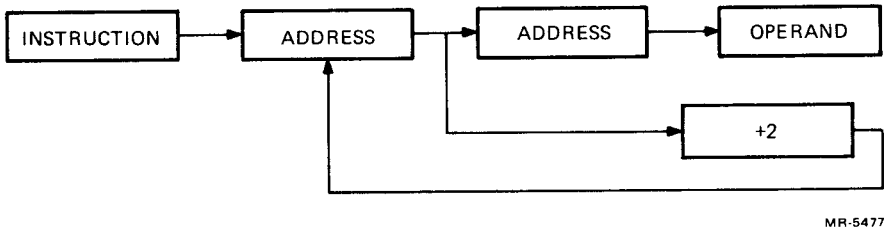


Figure 6-20 Mode 3 Autoincrement-Deferred

Mode	Name	Assembler Syntax	Function
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.

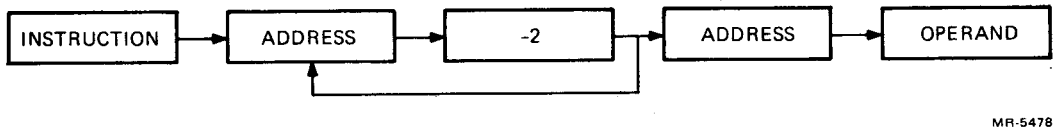


Figure 6-21 Mode 5 Autodecrement-Deferred

Mode	Name	Assembler Syntax	Function
7	Index-deferred	@X(Rn)	Value X (stored in a word following the instruction) and (Rn) are added; the sum is used as a pointer to a word containing the address of the operand. Neither X nor (Rn) is modified.

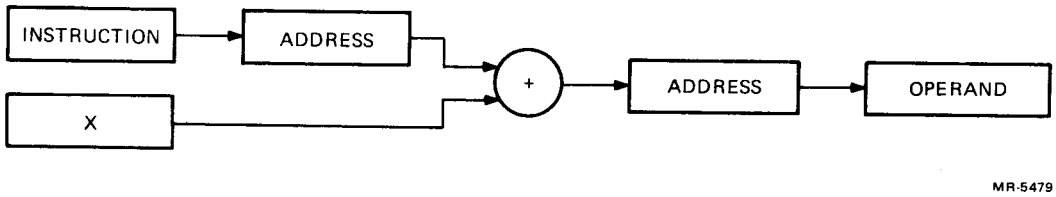


Figure 6-22 Mode 7 Index-Deferred

The following examples illustrate the deferred modes.

Register-Deferred Mode Example (Figure 6-23)

Symbolic	Octal Code	Instruction Name
CLR @R5	005015	Clear

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

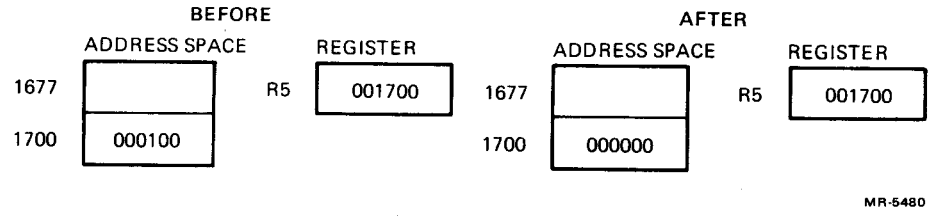


Figure 6-23 CLR @ R5 Clear

Autoincrement-Deferred Mode Example (Mode 3) (Figure 6-24)

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. The operand is increased by one; the contents of R2 are incremented by two.

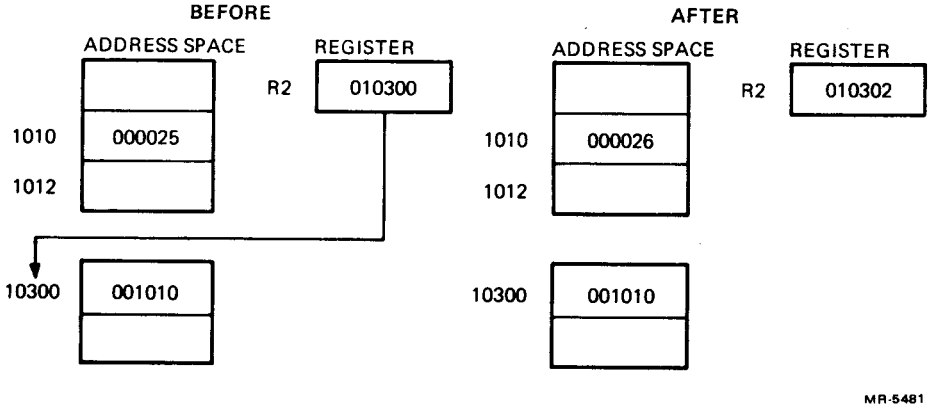


Figure 6-24 INC @ (R2) + Increment

Autodecrement-Deferred Mode Example (Mode 5) (Figure 6-25)

Symbolic	Octal Code
COM @–(R0)	005150

Operation: The contents of R0 are decremented by two and then used as the address of the address of the operand. The operand is 1's complemented (i.e., logically complemented).

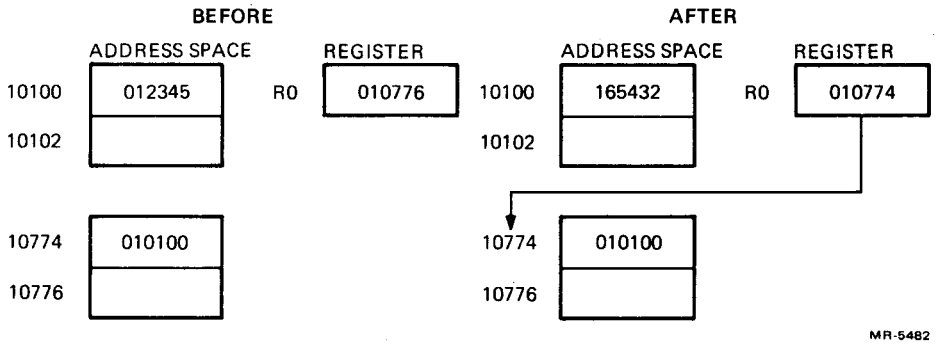


Figure 6-25 COM @ (R0) Complement

Index-Deferred Mode Example (Mode 7) (Figure 6-26)

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

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

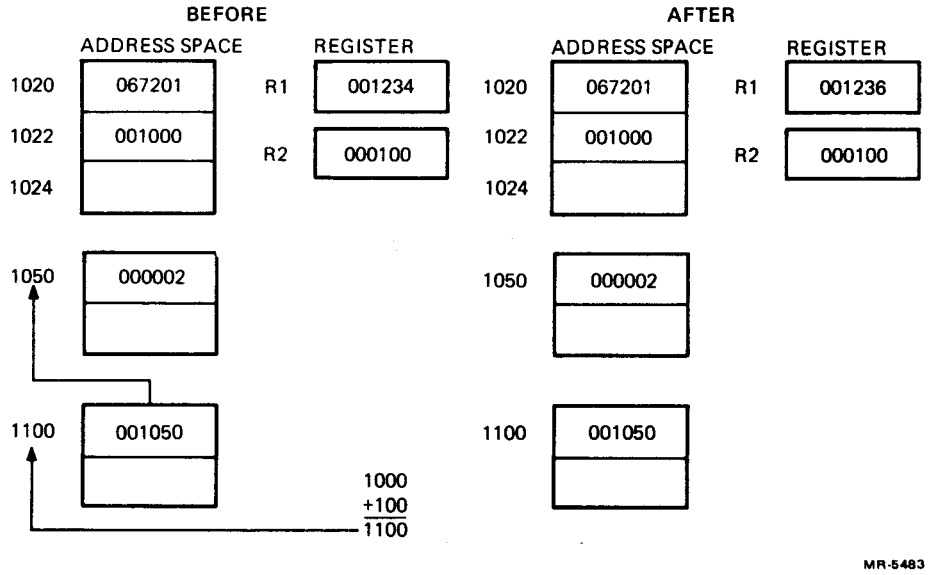


Figure 6-26 ADD @ 1000 (R2), R1 Add

6.2.5 Use of the PC as a General-Purpose Register

Although register 7 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, with four of these modes 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. The modes are summarized below.

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

When a standard program is available for different users, it is often helpful to be able to load it into different areas of memory and run it in those areas. The 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 in use to 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 (Figure 6-27)

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 it is incremented by two to point to the next instruction.

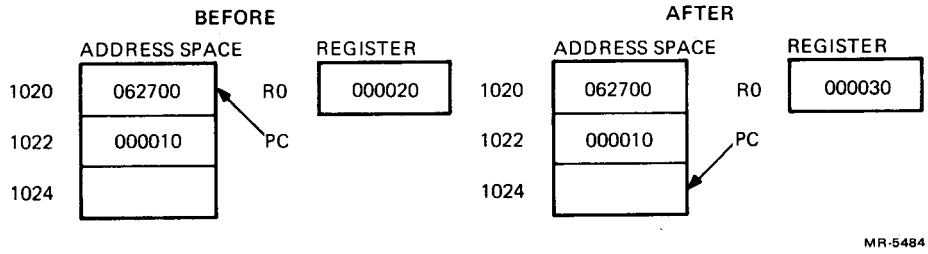


Figure 6-27 ADD # 10, R0 Add

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 (Figures 6-28 and 6-29)

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

Operation: Clear the contents of location 1100.

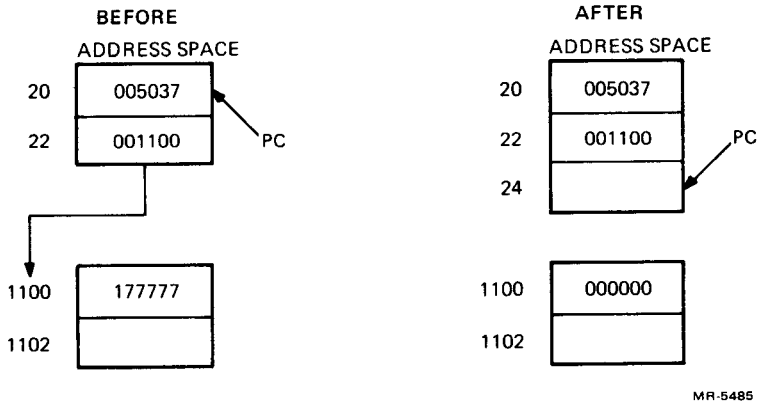


Figure 6-28 CLR @ # 1100 Clear

2.	Symbolic	Octal Code	Instruction Name
	ADD @#2000,R3	063703 002000	Add

Operation: Add contents of location 2000 to R3.

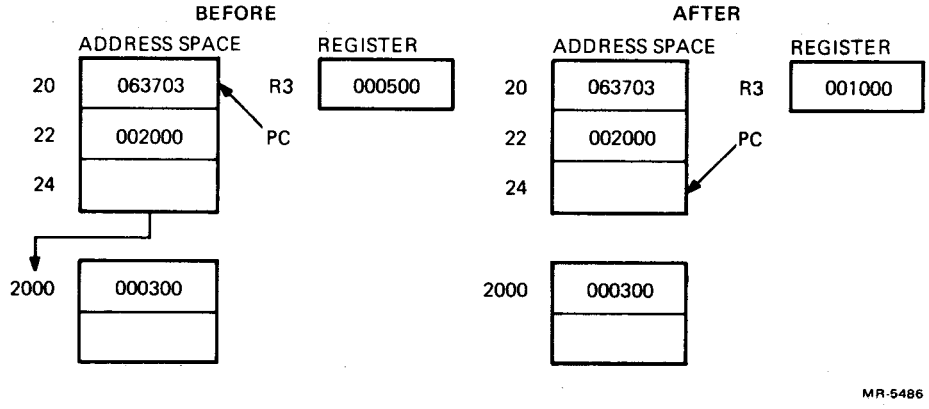


Figure 6-29 ADD @ # 2000 Add

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) (X is the location of A relative to the instruction)

Relative Addressing Example (Figure 6-30)

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.

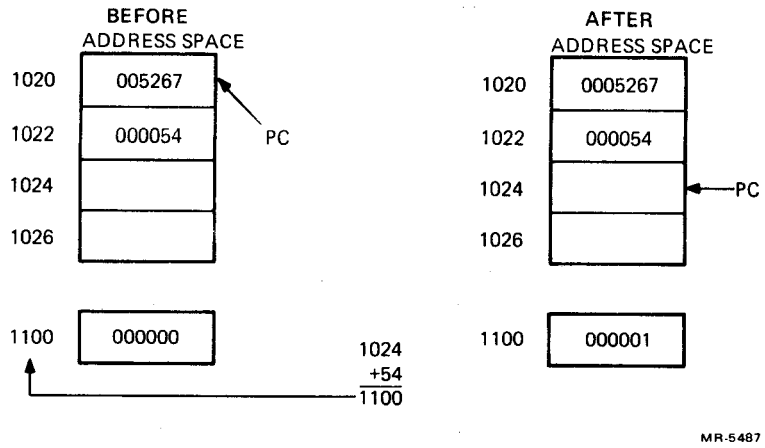


Figure 6-30 INC A Increment

6.2.5.4 Relative-Deferred Addressing [OPR @A or OPR @X(PC)] – This mode (mode 7) is similar to 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) (X is the location containing the address of A, relative to the instruction)

Relative-Deferred Mode Example (Figure 6-31)

Symbolic	Octal Code	Instruction Name
CLR @A	005077 000020	Clear

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

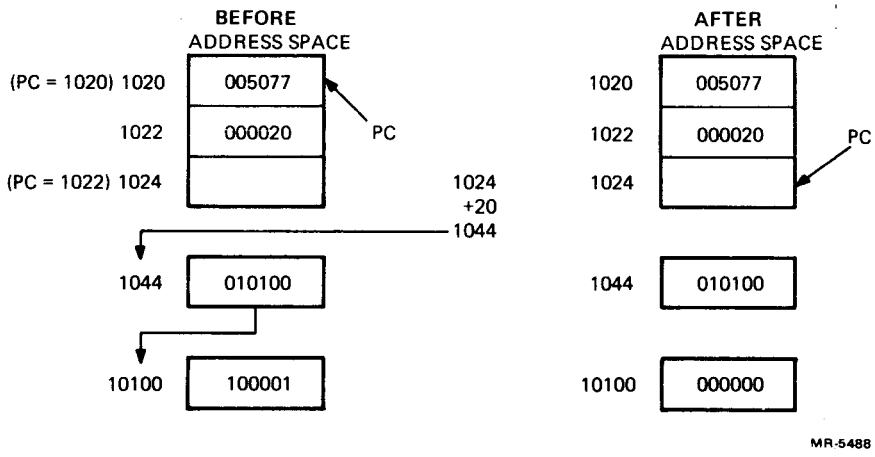


Figure 6-31 CLR @ A Clear

6.2.6 Use of the Stack Pointer as a General-Purpose Register

The processor stack pointer (SP, register 6) is in most cases the general register used for the stack operations related to program nesting. Autodecrement with register 6 "pushes" data onto the stack and autoincrement with register 6 "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 this chapter describes the DCT11-AA's instruction set. Each instruction's explanation includes the instruction's mnemonic, octal code, binary code, a diagram showing the format of the instruction, a symbolic notation describing its execution and effect on the condition codes, a description, special comments, and examples.

Each instruction's explanation is headed by its mnemonic. When the word instruction has a byte equivalent, the byte mnemonic also appears.

The diagram that accompanies each instruction shows the octal op code, 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

$\leftarrow$ = becomes

$\uparrow$ = "is popped from stack"

$\downarrow$ = "is pushed onto stack"

$\wedge$ = boolean AND

$\vee$ = boolean OR

$\forall$ = exclusive OR

$\sim$ = 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: (Figure 6-32) 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

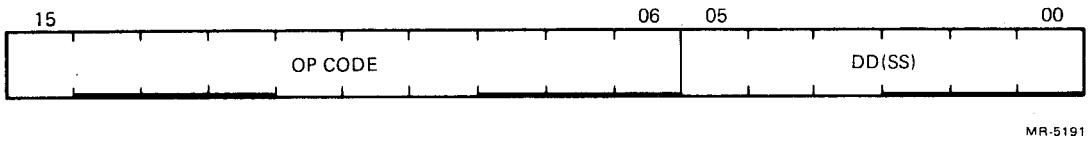


Figure 6-32 Single-Operand Group

2. Double-Operand Group: (Figure 6-33) BIT, BITB, BIC, BICB, BIS, BISB, ADD, SUB, MOV, MOVB, CMP, CMPB

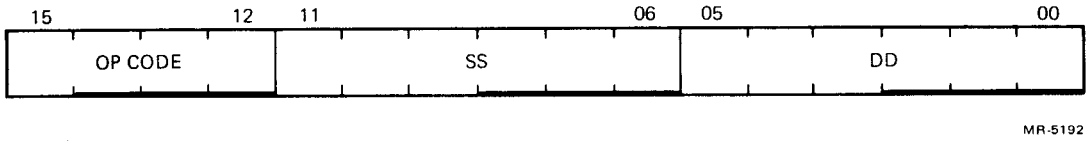


Figure 6-33 Double-Operand Group

3. Program Control Group:

   a. Branch (all branch instructions) (Figure 6-34)

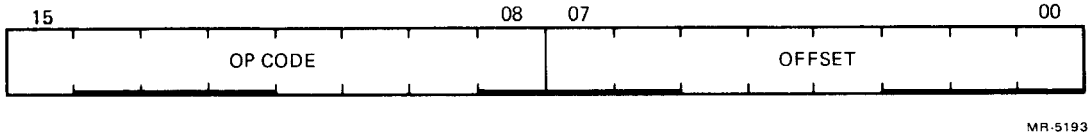


Figure 6-34 Program Control Group Branch

   b. Jump to Subroutine (JSR) (Figure 6-35)

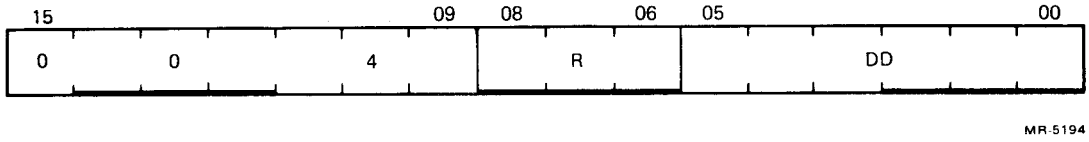


Figure 6-35 Program Control Group JSR

c. Subroutine Return (RTS) (Figure 6-36)

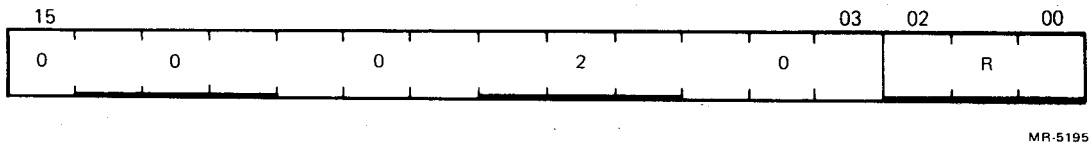


Figure 6-36 Program Control Group RTS

d. Traps (breakpoint, IOT, EMT, TRAP, BPT) (Figure 6-37)

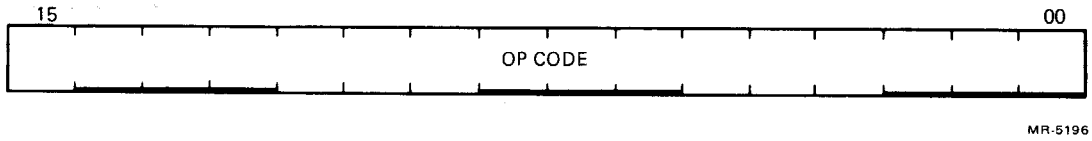


Figure 6-37 Program Control Group Traps

e. Subtract 1 and Branch (if = 0) (SOB) (Figure 6-38)

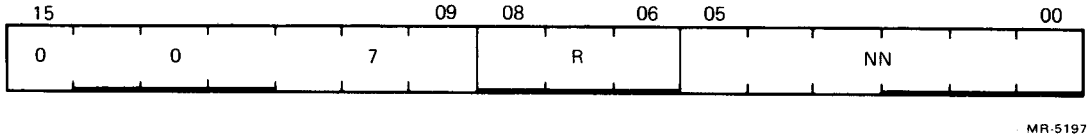


Figure 6-38 Program Control Group Subtract

4. Operate Group: HALT, WAIT, RTI, RESET, RTT, NOP, MFPT (Figure 6-39)

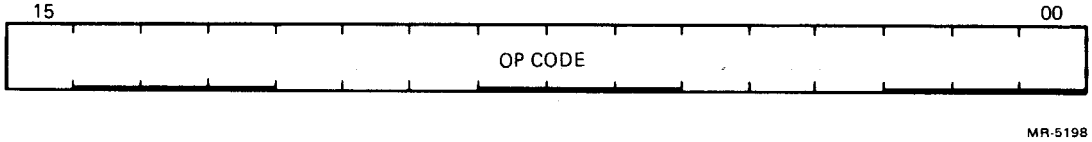


Figure 6-39 Operate Group

5. Condition Code Operators (all condition code instructions) (Figure 6-40)

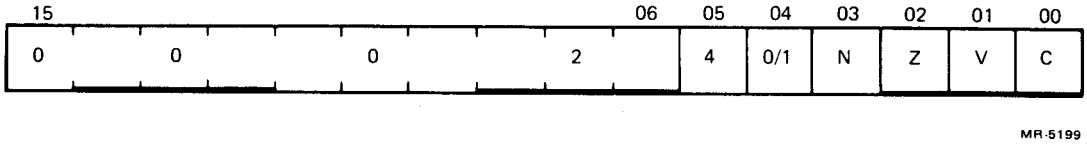


Figure 6-40 Condition Group

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 shown in Figure 6-41.

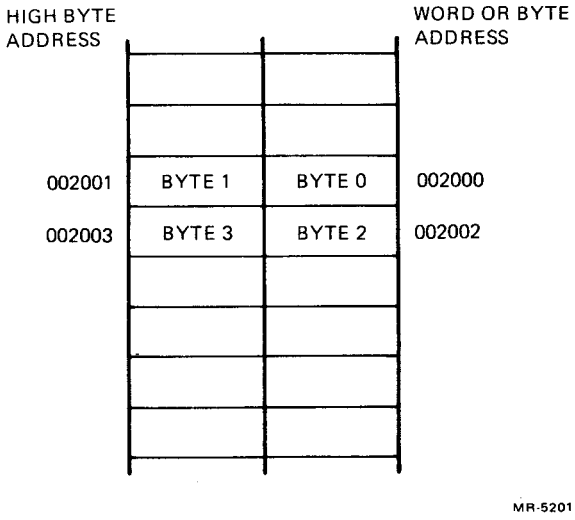


Figure 6-41 Byte Instructions

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

Example:

Symbolic	Octal Code	Instruction Name
CLR	0050DD	Clear word
CLRB	1050DD	Clear byte

6.3.2 List of Instructions

The following is a list of the DCT11-AA instruction set.

SINGLE-OPERAND

General

Mnemonic	Instruction	Op Code
CLR(B)	Clear destination	■050DD
COM(B)	Complement destination	■051DD
INC(B)	Increment destination	■052DD
DEC(B)	Decrement destination	■053DD
NEG(B)	Negate destination	■054DD
TST(B)	Test destination	■057DD

Shift and Rotate

Mnemonic	Instruction	Op Code
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

Mnemonic	Instruction	Op Code
ADC(B)	Add carry	■055DD
SBC(B)	Subtract carry	■056DD
SXT	Sign extend	0067DD

PS Word Operators

Mnemonic	Instruction	Op Code
MFPS	Move byte from PS	1067DD
MTPS	Move byte to PS	1064SS

DOUBLE-OPERAND

General

Mnemonic	Instruction	Op Code
MOV(B)	Move source to destination	■1SSDD
CMP(B)	Compare source to destination	■2SSDD
ADD	Add source to destination	06SSDD
SUB	Subtract source from destination	16SSDD

Logical

Mnemonic	Instruction	Op Code
BIT(B)	Bit test	■3SSDD
BIC(B)	Bit clear	■4SSDD
BIS(B)	Bit set	■5SSDD
XOR	Exclusive OR	074RDD

PROGRAM CONTROL

Branch

Mnemonic	Instruction	Op Code or Base Code
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

Mnemonic	Instruction	Op Code or Base Code
BGE	Branch if 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

Mnemonic	Instruction	Op Code or Base Code
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 and Subroutine

Mnemonic	Instruction	Op Code or Base Code
JMP	Jump	0001DD
JSR	Jump to subroutine	004RDD
RTS	Return from subroutine	00020R
SOB	Subtract one and branch (if $\neq 0$)	077R00

Trap and Interrupt

Mnemonic	Instruction	Op Code or Base Code
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

Mnemonic	Instruction	Op Code or Base Code
HALT	Halt	000000
WAIT	Wait for interrupt	000001
RESET	Reset external bus	000005
MFPT	Move processor type	000007

RESERVED INSTRUCTIONS

00021R
00022R

CONDITION CODE OPERATORS

Mnemonic	Instruction	Op Code or Base Code
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

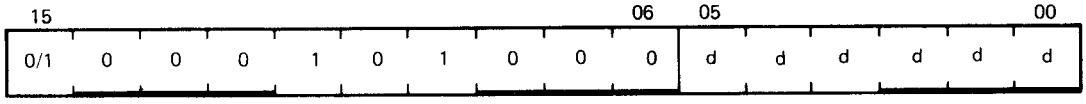
In all DCT11-AA instructions a write operation (which in 8-bit mode consists of two adjacent and indivisible write transactions) to a memory location or register is always preceded by a read operation from the same location. The exceptions are when writing the PC and PSW to the stack in two cases:

1. **During the execution of the microcode preceding an interrupt or trap service routine.**
2. **In interrupt and trap instructions (HLT, TRAP, BPT, IOT).**

6.3.3.1 General

CLR
CLRB

CLEAR DESTINATION ■050DD



MR-5202

Operation: (dst) ← 0

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

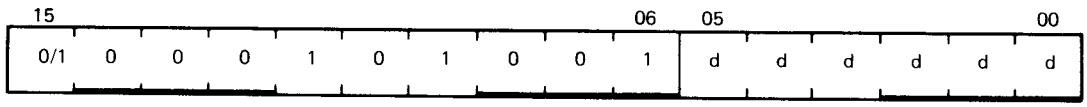
Description: *Word:* The contents of the specified destination are replaced with 0s.
Byte: Same.

Example: CLR R1

Before	After
(R1) = 177777	(R1) = 000000
NZVC	NZVC
1111	0100

COM
COMB

COMPLEMENT DST ■051DD



MR-5203

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

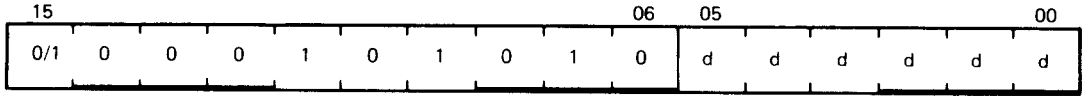
Description: *Word:* Replaces the contents of the destination address by their logical complement. (Each bit equal to 0 is set and each bit equal to 1 is cleared.)
Byte: Same.

Example: COM R0

Before	After
(R0) = 013333	(R0) = 164444
NZVC 0110	NZVC 1001

INC
INCB

INCREMENT DST ■052DD



MR-5204

Operation: (dst) ← (dst) + 1

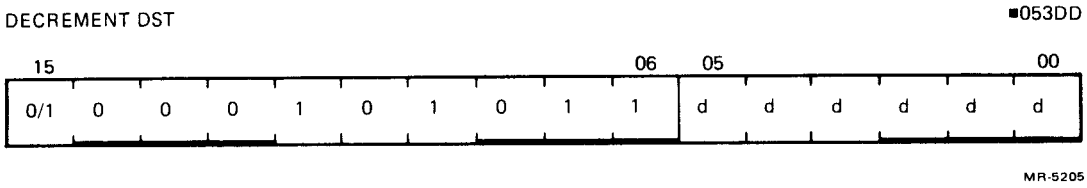
Condition Codes:
N: set if result is < 0; cleared otherwise
Z: set if result is 0; cleared otherwise
V: set if (dst) held 077777; cleared otherwise
C: not affected

Description: *Word:* Add 1 to the contents of the destination.
Byte: Same.

Example: INC R2

Before	After
(R2) = 000333	(R2) = 000334
NZVC 0000	NZVC 0000

DEC
DECB



Operation: (dst) ← (dst) − 1

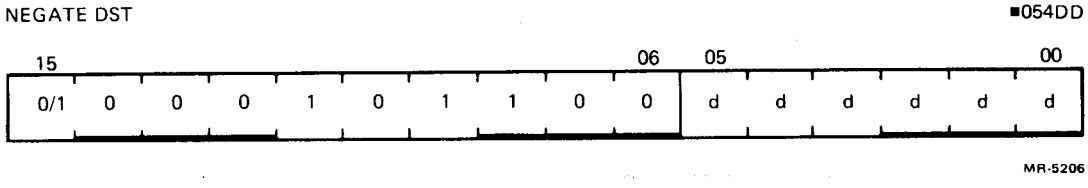
Condition Codes:
N: set if result is < 0; cleared otherwise
Z: set if result is 0; cleared otherwise
V: set if (dst) was 100000; cleared otherwise
C: not affected

Description: *Word:* Subtract 1 from the contents of the destination.
Byte: Same.

Example: DEC R5

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

NEG
NEGB



Operation: (dst) ← − (dst)

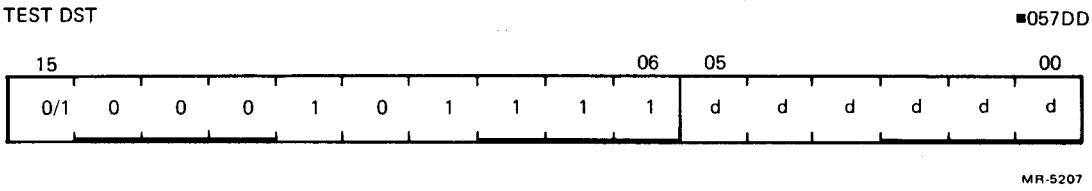
Condition Codes:
N: set if result is < 0; cleared otherwise
Z: set if result is 0; cleared otherwise
V: set if result is 100000; cleared otherwise
C: cleared if result is 0; set otherwise

Description: *Word:* Replaces the contents of the destination address by its 2's complement. Note that 100000 is replaced by itself. (In 2's complement notation the most negative number has no positive counterpart.)
Byte: Same.

Example: NEG R0

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

TST
TSTB



Operation: (dst) ← (dst)

Condition Codes:
N: set if 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; the contents of dst remain unmodified.
Byte: Same.

Example: TST R1

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

6.3.3.2 Shifts and Rotates – Scaling data by factors of two is accomplished by the shift instructions:

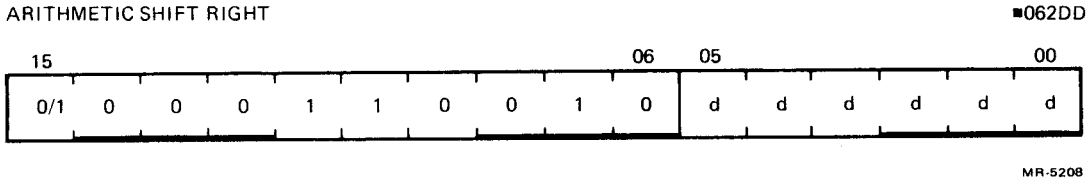
ASR – Arithmetic shift right

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 0s in shifts to the left. Bits shifted out of the C bit, as shown in the following instructions, 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

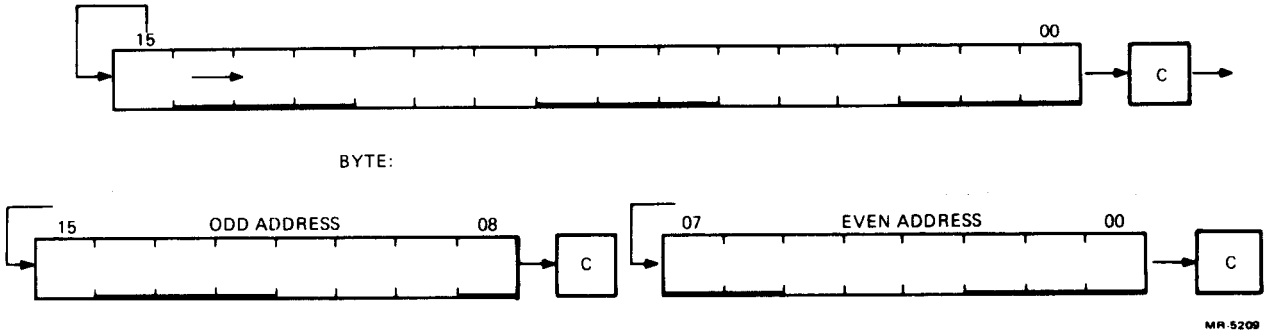


Operation: (dst) ← (dst) shifted one place to the right

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

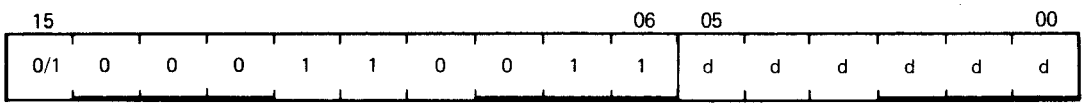
Description: *Word:* Shifts all bits of the destination right one place. Bit 15 is reproduced. The C bit is loaded from bit 0 of the destination. ASR performs signed division of the destination by 2.
Byte: Same.

Example:



ASL
ASLB

ARITHMETIC SHIFT LEFT ■063DD



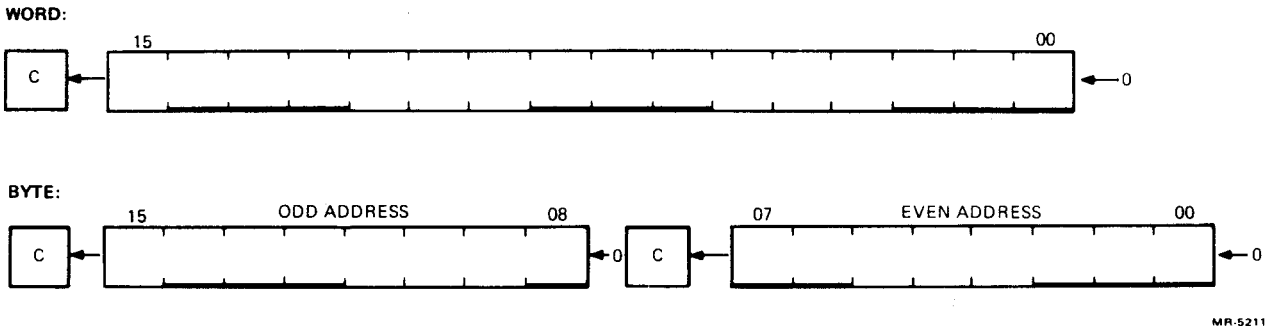
MR-5210

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

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

Description: *Word:* Shifts all bits of the destination left one place. Bit 0 is loaded with a 0. 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 2 with overflow indication.
Byte: Same.

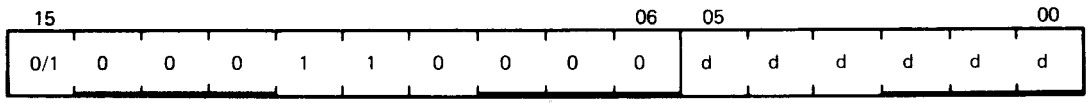
Example:



MR-5211

ROR
RORB

ROTATE RIGHT ■060DD



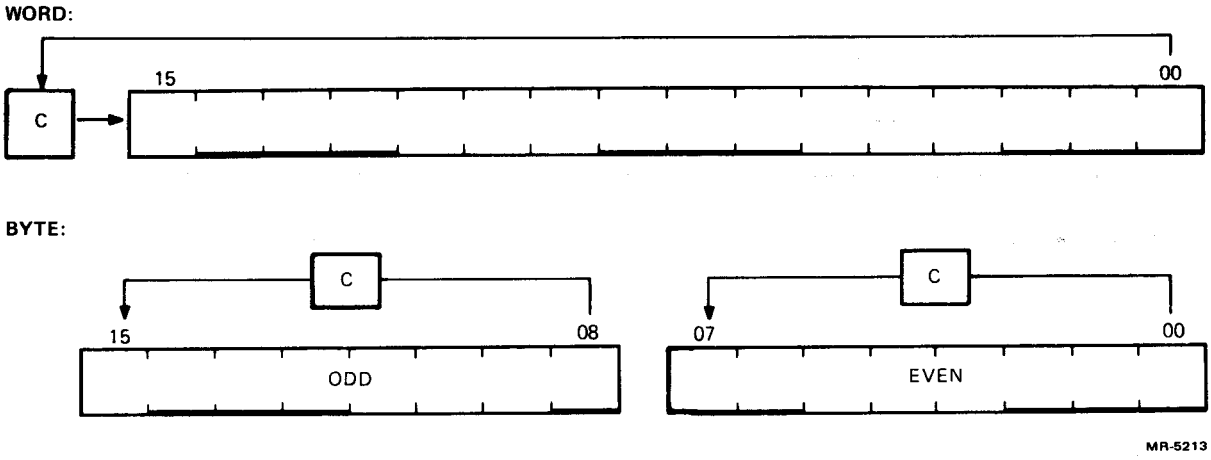
MR-5212

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

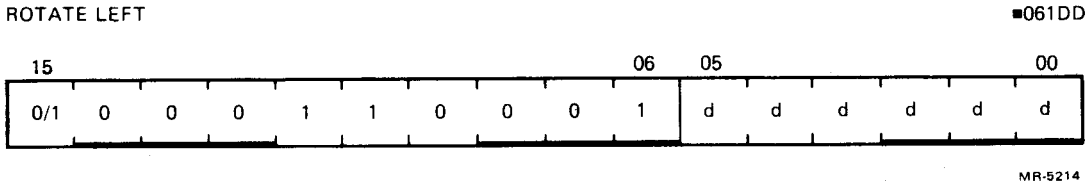
Condition Codes:
N: set if high-order bit of result is set (result < 0); cleared otherwise
Z: set if all bits of result = 0; cleared otherwise
V: loaded with exclusive OR of N bit and C bit (as set by the completion of the rotate operation)
C: loaded with low-order bit of destination

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:



ROL
ROLB

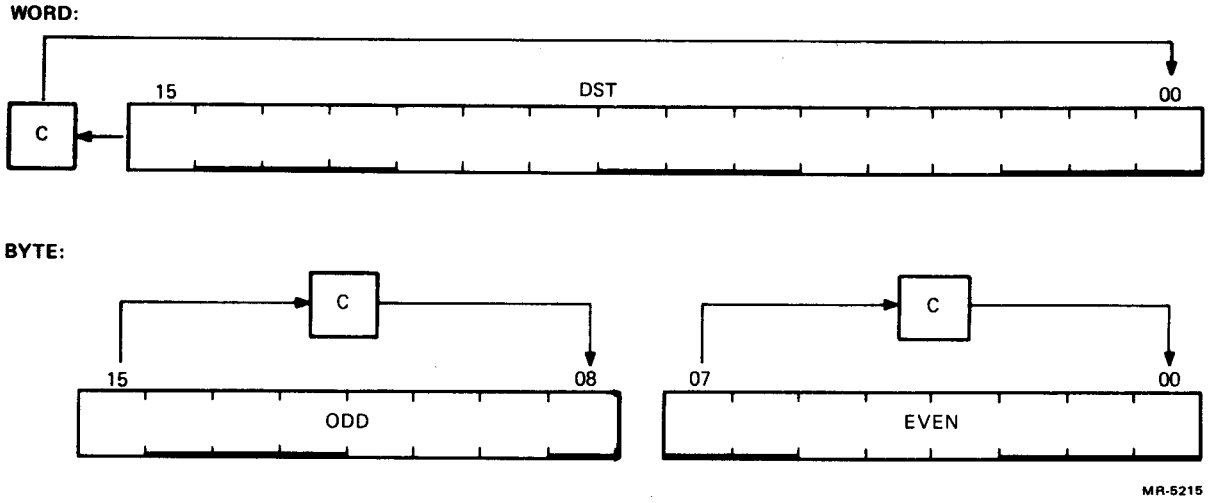


Operation: (dst) ← (dst) rotate left one place

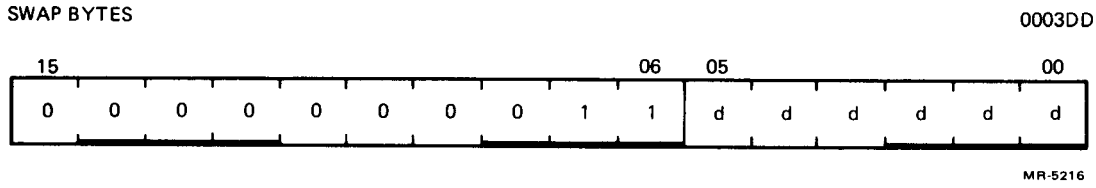
Condition Codes:
N: set if high-order bit of result word is set (result < 0); cleared otherwise
Z: set if all bits of result word = 0; cleared otherwise
V: loaded with exclusive OR of the N bit and C bit (as set by the completion of the rotate operation)
C: loaded with high-order bit of destination

Description: *Word:* Rotates 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:



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

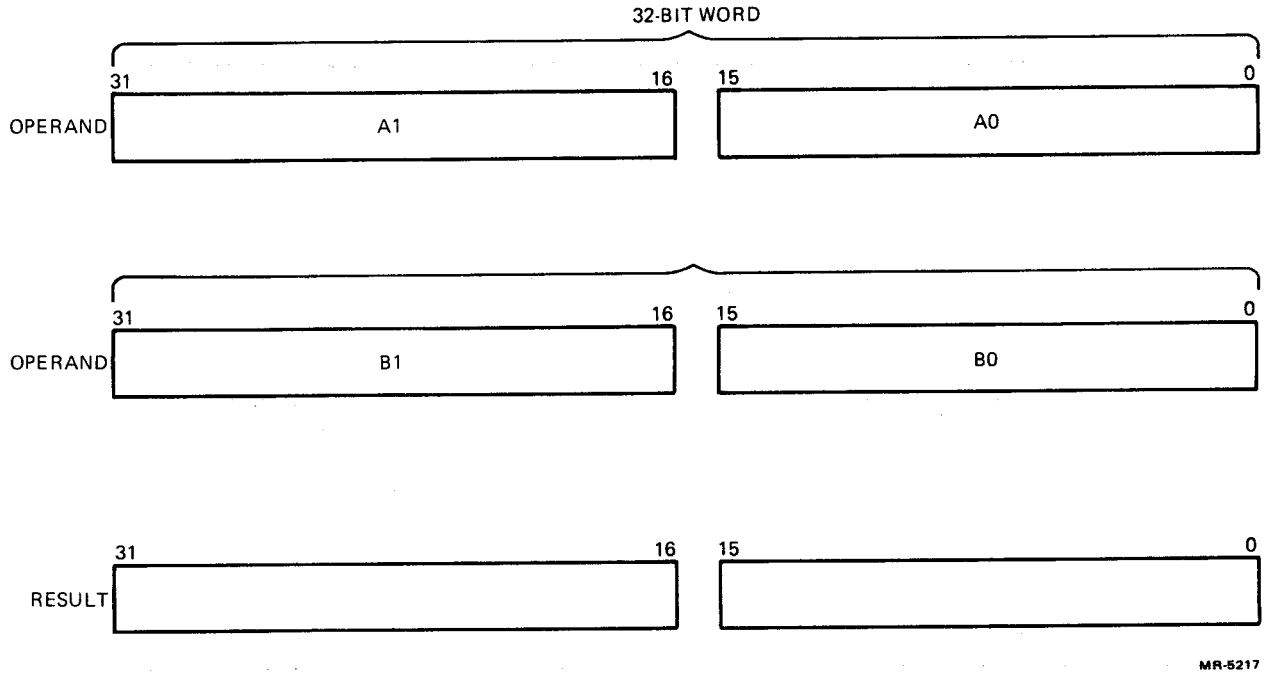
Description: Exchanges high-order byte and low-order byte of the destination word. (The destination must be a word address.)

Example: SWAB R1

Before	After
(R1) = 077777	(R1) = 177577
NZVC 1111	NZVC 0000

6.3.3.3 Multiple-Precision – It is sometimes necessary to do arithmetic operations 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.



Example:

The addition of −1 and −1 could be performed as follows.

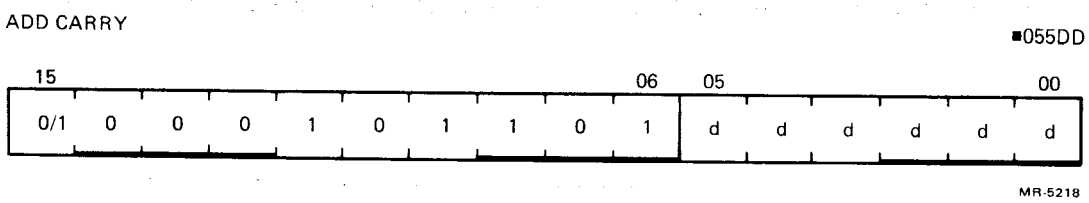
−1 = 37777777777

(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. The ADC instruction adds the C bit to (R3); (R3) = 0.
3. The (R3) and (R4) are added.
4. The result is 37777777776, or −2.

ADC
ADCB



Operation: (dst) ← (dst) + (C bit)

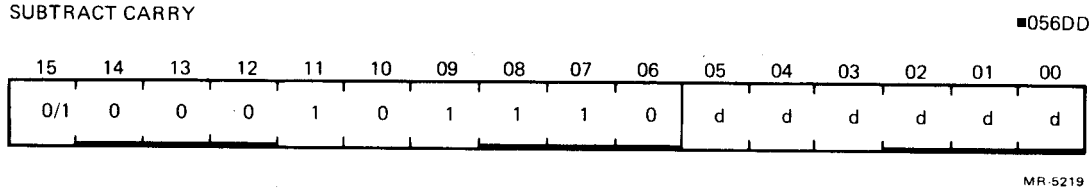
Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if (dst) was 077777 and (C) was 1; cleared otherwise
C: set if (dst) was 177777 and (C) was 1; cleared otherwise

Description: *Word:* Adds the contents of the C bit to the destination. This permits the carry from the addition of the low-order words to be carried to 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



Operation: (dst) ← (dst) − (C)

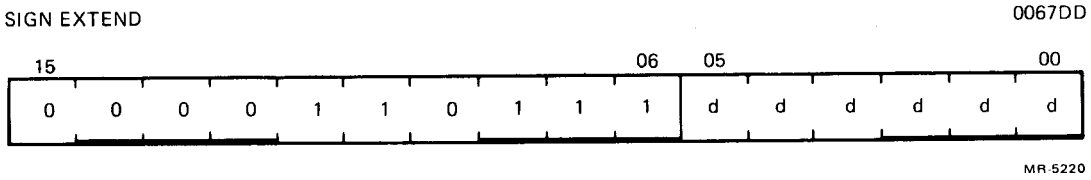
Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if (dst) was 100000; cleared otherwise
C: set if (dst) was 0 and C was 1; cleared otherwise

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



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

Condition Codes: N: not affected
Z: set if N bit is clear
V: cleared
C: not affected

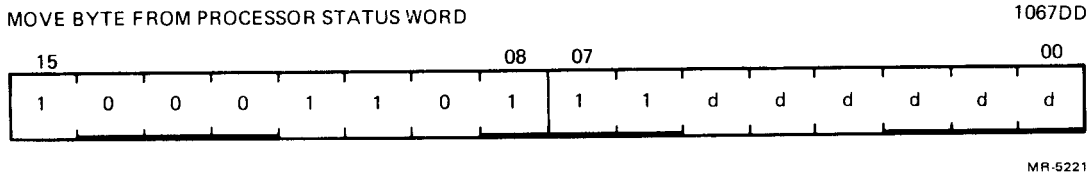
Description: If the condition code bit N is set, a −1 is placed in the destination operand; if the N bit is clear, a 0 is placed in the destination operand. This instruction is particularly useful in multiple-precision arithmetic because it permits the sign to be extended through multiple words.

Example: SXT A

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

6.3.3.4 PS Word Operators

MFPS



Operation: (dst) ← PS
dst lower 8 bits

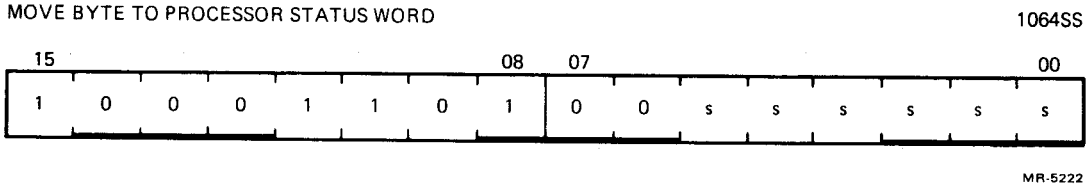
Condition Codes: N: set if PS <7> = 1; cleared otherwise
Z: set if PS <7:0> = 0; cleared otherwise
V: cleared
C: not affected

Description: The 8-bit contents of the PS are moved to the effective destination. If the destination is mode 0, PS bit 7 is sign-extended through the 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



Operation: PS ← (src)

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

Description: The eight bits of the effective operand replace the current contents of the PS. The source operand address is treated as a byte address. Note: The T bit (PS bit 4) 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

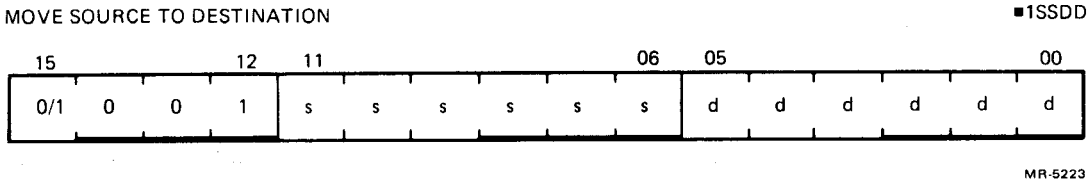
Before	After
(R1) = 000777	(R1) = 000777
(PS) = XXX000	(PS) = XXX357
NZVC	NZVC
0000	1111

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



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. 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.

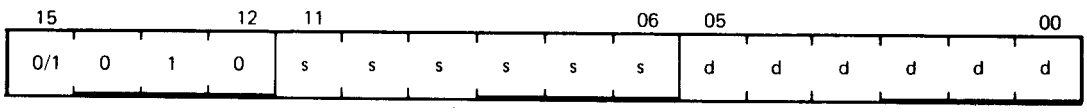
Example:

MOV XXX,R1	;loads register 1 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 the terminal keyboard buffer to the terminal printer buffer

CMP
CMPB

COMPARE SRC TO DST

■2SSDD



MR-5224

Operation: (src) — (dst)

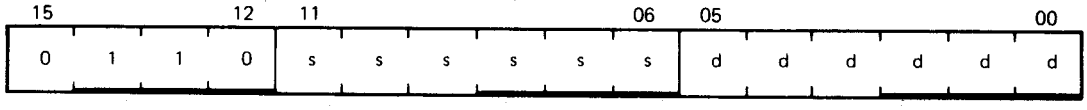
Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if there was arithmetic overflow; that is, operands were of opposite signs and the sign of the destination was the same as the sign of the result; cleared otherwise
C: cleared if there was a carry from the result's most significant bit; 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 not affected. 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

ADD SRC TO DST

06SSDD



MR-5225

Operation: (dst) ← (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 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 result's most significant bit; 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.

Example: Add to register: ADD 20,R0

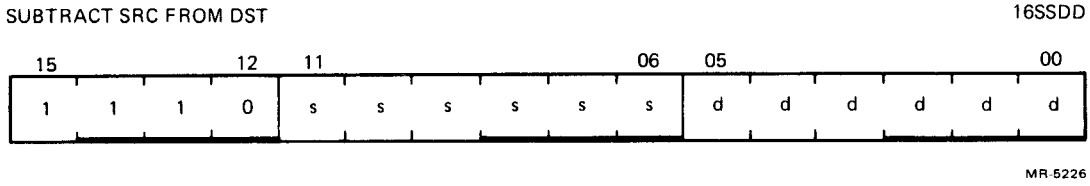
Add to memory: ADD R1,XXX

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



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 result's most significant bit; set otherwise

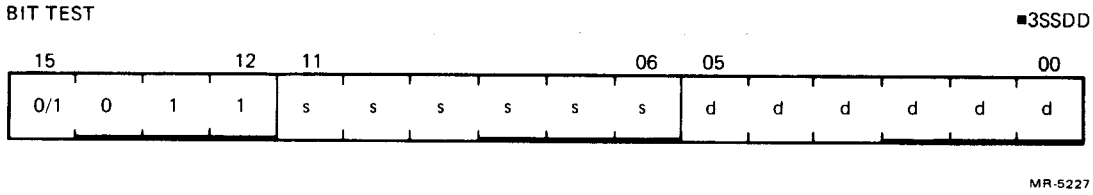
Description: Subtracts the source operand from the destination operand and leaves the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. In double-precision arithmetic the C bit, when set, indicates a "borrow." Note: There 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 those in the double-operand arithmetic group. They permit operations on data at the bit level.

BIT
BITB



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

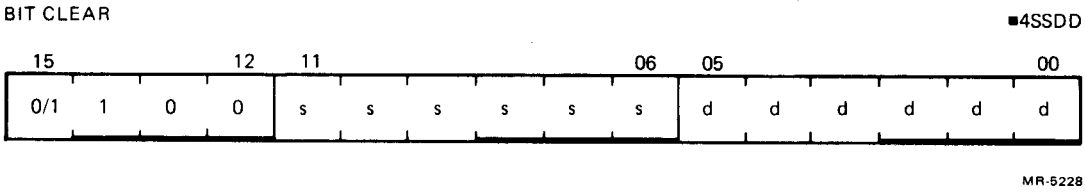
Description: Performs logical AND comparison of the source and destination operands and modifies condition codes accordingly. Neither the source nor the destination is affected. The BIT instruction may be used to test whether any of the corresponding bits set in the destination are also set in the source,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



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 not affected.

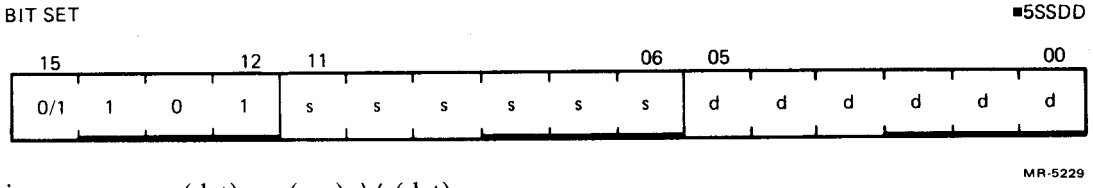
Example: BIC R3,R4

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



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: Performs an 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.

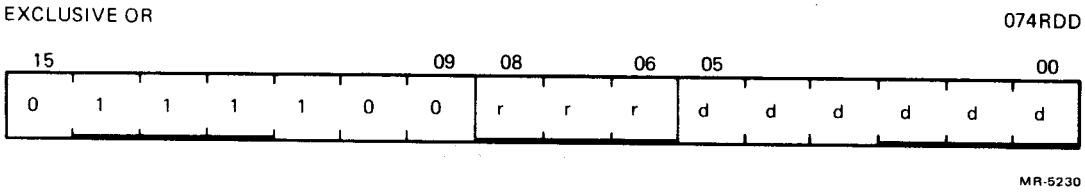
Example: BIS R0,R1

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



Operation: (dst) ← (reg) ∀ (dst)

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

Description: The exclusive OR of the register and destination operand is stored in the destination address. The contents of the register are not affected. The 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 2) 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 2 and sign-extended to express words before it is added to the PC. Bit 7 is the sign of the offset. If it is set, the offset is negative and the branch is done in the backward direction. If it is not set, the offset is positive and the branch is done in the forward direction.

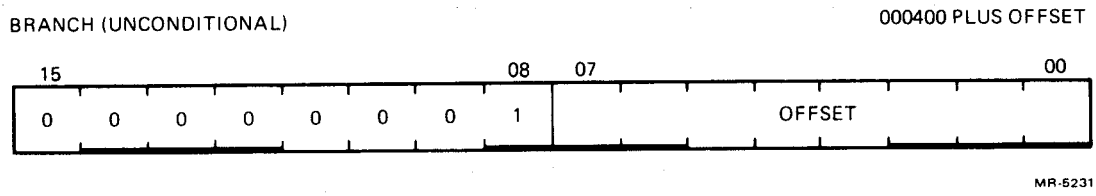
The 8-bit offset allows branching in the backward direction by 200_8 words (400_8 bytes) from the current PC, and in the forward direction by 177_8 words (376_8 bytes) from the current PC.

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

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 if 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 NOPs.

BR



Operation: PC ← PC + (2 × offset)

Condition Codes: Not affected

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

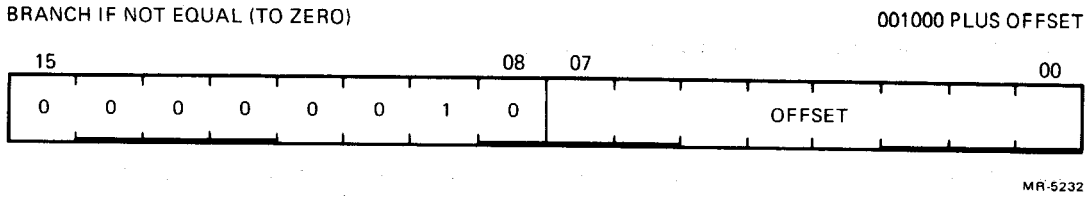
New PC address = updated PC + (2 × offset)

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

BNE



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

Condition Codes: Not affected

Description: Tests the state of the Z bit and causes a branch if the Z bit is clear. BNE is the complementary operation of BEQ. It is used to test: (1) inequality following a CMP, (2) that some bits set in the destination were also in the source following a BIT operation, and (3) generally, that the result of the previous operation was not 0.

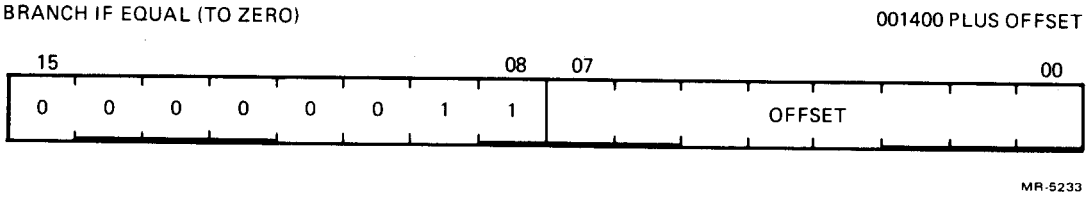
Example: Branch to C if A ≠ B

```
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



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

Condition Codes: Not affected

Description: Tests the state of the Z bit and causes a branch if Z is set. It is used to test: (1) equality following a CMP operation, (2) that no bits set in the destination were also set in the source following a BIT operation, and (3) generally, that the result of the previous operation was 0.

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

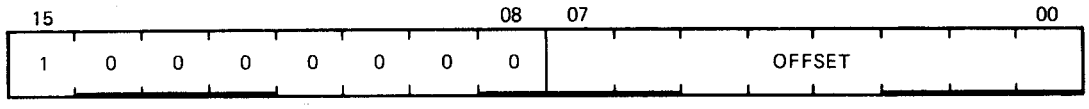
```
CMP A,B         ;compare A and B
BEQ C           ;branch if they are equal
```

Branch to C if A + B = 0

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

BPL

BRANCH IF PLUS — 100000 PLUS OFFSET



MR-5234

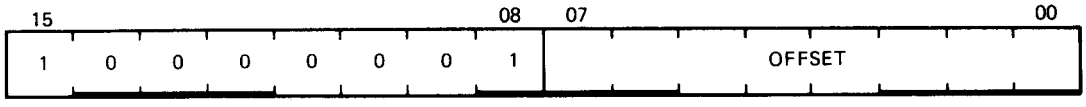
Operation: PC ← PC + (2 × offset) if N = 0

Condition Codes: Not affected

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

BRANCH IF MINUS — 100400 PLUS OFFSET



MR-5235

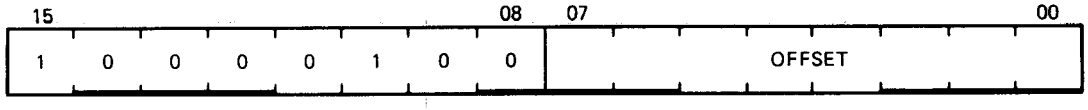
Operation: PC ← PC + (2 × offset) if N = 1

Condition Codes: Not affected

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

BRANCH IF OVERFLOW IS CLEAR — 102000 PLUS OFFSET



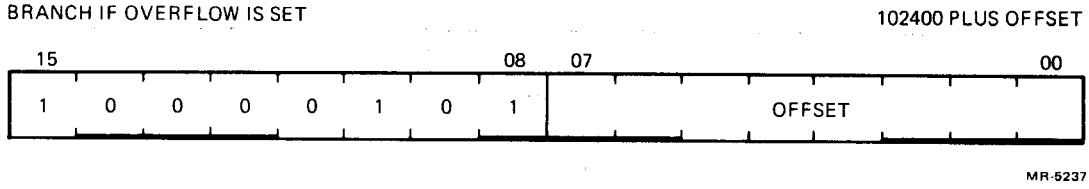
MR-5236

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

Condition Codes: Not affected

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

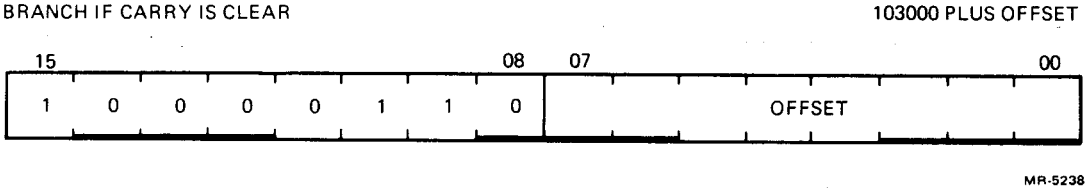


Operation: PC ← PC + (2 × offset) if V = 1

Condition Codes: Not affected

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

BCC

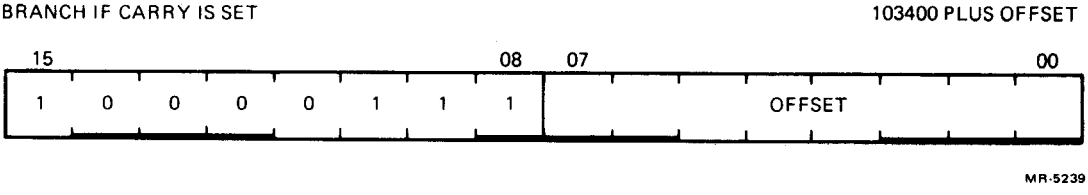


Operation: PC ← PC + (2 × offset) if C = 0

Condition Codes: Not affected

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

BCS



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

Condition Codes: Not affected

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 (2's complement) values.

Note that the sense of signed comparisons differs from that of unsigned comparisons in that in signed, 16-bit, 2's complement arithmetic the sequence of values is as follows.

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

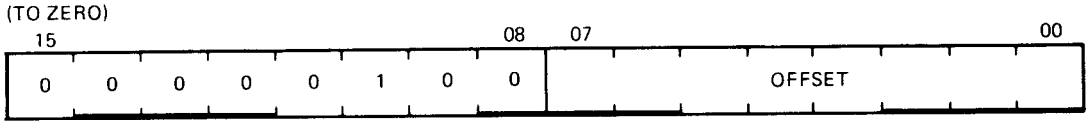
Whereas, in unsigned, 16-bit arithmetic, the sequence is considered to be:

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

BGE

BRANCH IF GREATER THAN OR EQUAL (TO ZERO)

002000 PLUS OFFSET



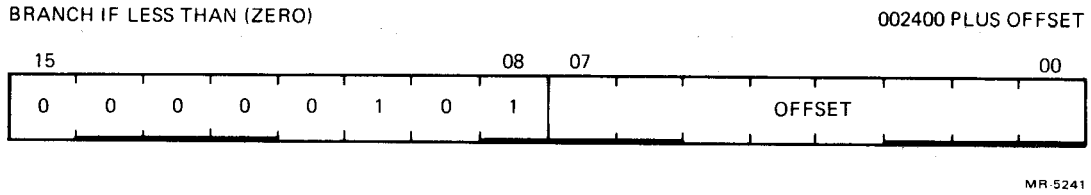
MR-5240

Operation: PC ← PC + (2 × offset) if N ∀ V = 0

Condition Codes: Not affected

Description: Causes a branch if N and V are either both clear or both set. BGE is the complementary operation of 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 0 result.

BLT

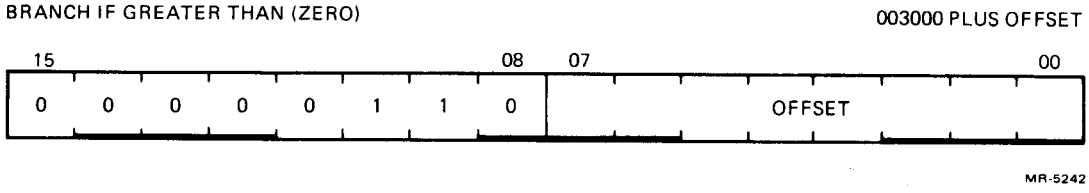


Operation: PC ← PC + (2 × offset) if N ∀ V = 1

Condition Codes: Not affected

Description: Causes a branch if the exclusive OR of the N and V bits is 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 if 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 0 (without overflow).

BGT



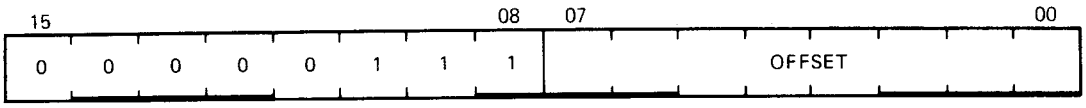
Operation: PC ← PC + (2 × offset) if Z ∨ (N ∀ V) = 0

Condition Codes: Not affected

Description: Operation of BGT is similar to BGE, except that BGT will not cause a branch on a 0 result.

BLE

BRANCH IF LESS THAN OR EQUAL (TO ZERO) 003400 PLUS OFFSET



MR-5243

Operation: PC ← PC + (2 × offset) if Z ∨ (N ∀ V) = 1

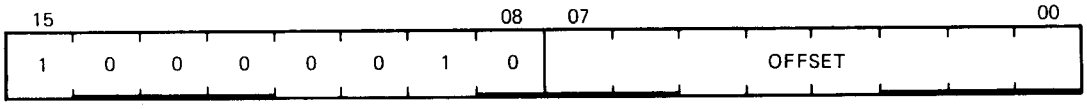
Condition Codes: Not affected

Description: Operation is similar to BLT, but in addition will cause a branch if the result of the previous operation was 0.

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

BRANCH IF HIGHER 101000 PLUS OFFSET



MR-5244

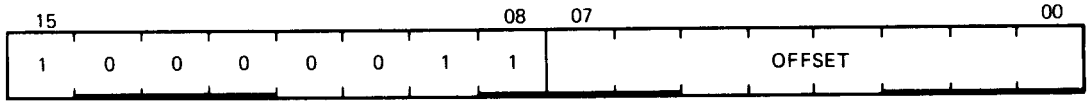
Operation: PC ← PC + (2 × offset) if C = 0 and Z = 0

Condition Codes: Not affected

Description: Causes a branch if the previous operation caused neither a carry nor a 0 result. This will happen in comparison (CMP) operations as long as the source has a higher unsigned value than the destination.

BLOS

BRANCH IF LOWER OR SAME 101400 PLUS OFFSET



MR-5245

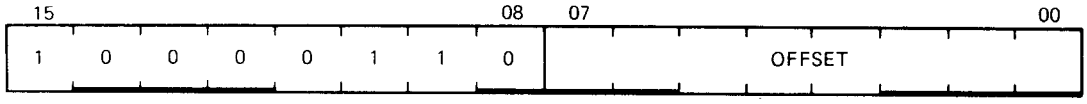
Operation: PC ← PC + (2 × offset) if C ∨ Z = 1

Condition Codes: Not affected

Description: Causes a branch if the previous operation caused either a carry or a 0 result. BLOS is the complementary operation of 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

BRANCH IF HIGHER OR SAME 103000 PLUS OFFSET



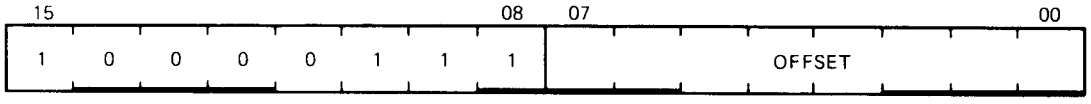
Operation: PC ← PC + (2 × offset) if C = 0

Condition Codes: Not affected

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

BLO

BRANCH IF LOWER 103400 PLUS OFFSET



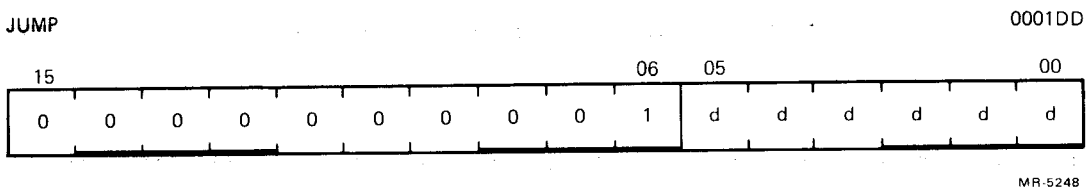
Operation: PC ← PC + (2 × offset) if C = 1

Condition Codes: Not affected

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

6.3.5.4 Jump and 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, and thus provides for reentrancy. This allows one copy of a subroutine to be shared among several interrupting processes.

JMP



Operation: PC ← (dst)

Condition Codes: Not affected

Description: JMP provides more flexible program branching than the branch instructions do. 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.

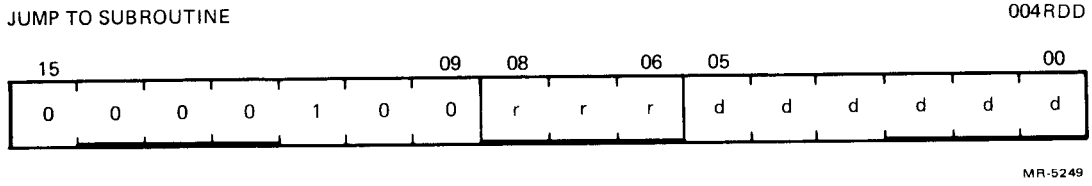
Deferred-index mode JMP instructions permit transfer of control to the address contained in a selectable element of a table of dispatch vectors.

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



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)

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 is 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 may be 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.

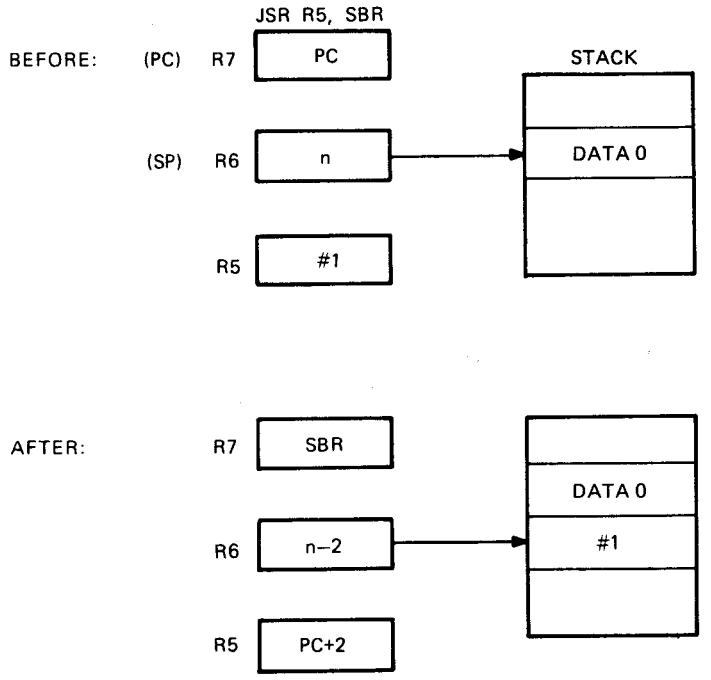
Another special case of the JSR instruction is JSR PC,@(SP) +, which exchanges the top element of the processor stack with the contents of the program counter. This instruction allows two routines to swap program control and resume operation from where they left off when they are recalled. Such routines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

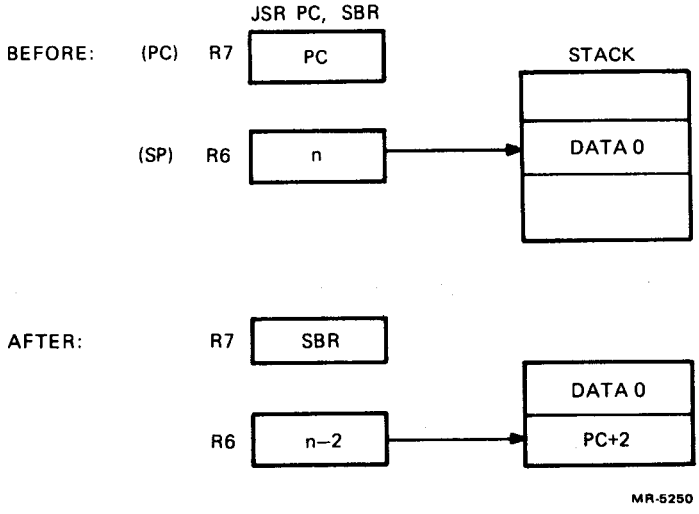
Return from a subroutine is done by the RTS instruction. RTS reg loads the contents of reg into the PC and pops the top element of the processor stack into the specified register.

Example:

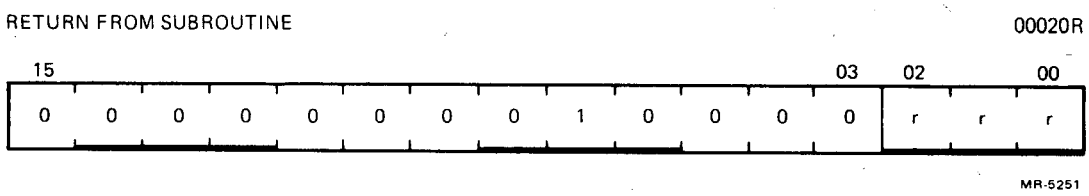
```
                                        R5          R6      R7
SBCALL:         JSR R5,SBR              #1          n       SBCALL
SBCALL+4:       ARG 1
                ARG 2
                .
                .
SBCALL+2+2M:    ARG M
CONT:           Next Instruction        #1          n       CONT
                .
                .
```

SBR:	MOV (R5)+,dst 1 MOV (R5)+,dst 2	SBCALL+4	n−2	SBR
	MOV (R5)+,dst M	SBCALL+2+2M		
	Other Instructions	CONT		
EXIT:	RTS R5	CONT	n−2	EXIT





RTS

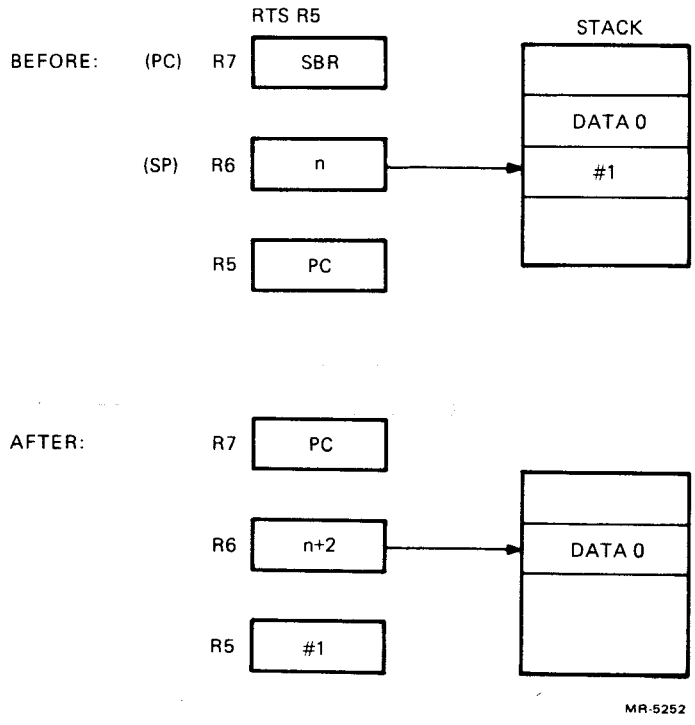


Operation: PC ← (reg)
(reg) ← (SP) ↑

Description: Loads the contents of the register into PC and pops the top element of the processor stack into the specified register.

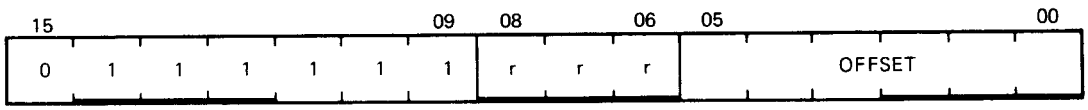
Return from a nonreentrant subroutine is typically made through the same register that was used in its call. Thus, a subroutine called with a JSR PC, dst exits with a RTS PC and a subroutine called with a JSR R5, dst, may pick up parameters with addressing modes (R5) +, X(R5), or @X(R5) and finally exits, with an RTS R5.

Example: RTS R5



SOB

SUBTRACT ONE AND BRANCH (IF ≠ 0) 077RNN



MR-5253

Operation: (R) ← (R) − 1; if this result ≠ 0, then PC ← PC − (2 × offset); if (R) = 0 then PC ← PC

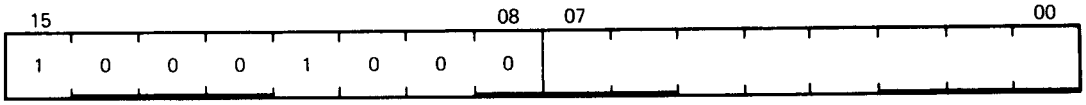
Condition Codes: Not affected

Description: The register is decremented. If the contents does not equal 0, twice the offset is subtracted from the PC (now pointing to the following word). The offset is interpreted as a 6-bit positive number. This instruction provides a fast, efficient method of loop control. The 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: 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 2-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

EMULATOR TRAP 104000–104377



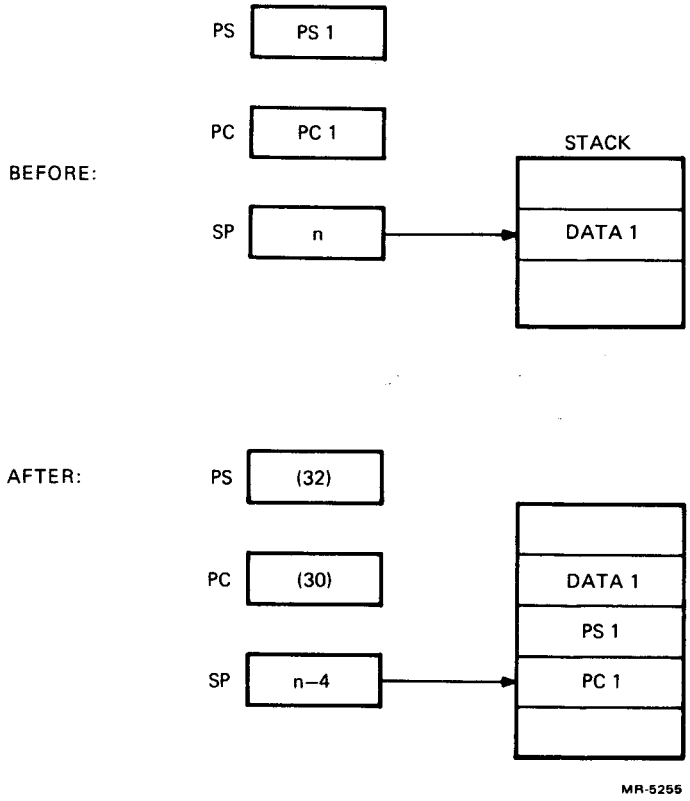
MR-5254

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

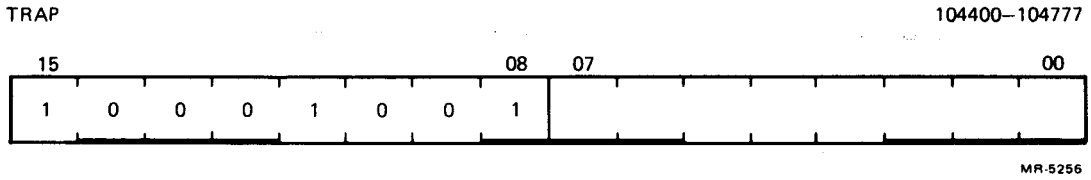
Condition Codes:
N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

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 DIGITAL system software and is therefore not recommended for general use.



TRAP



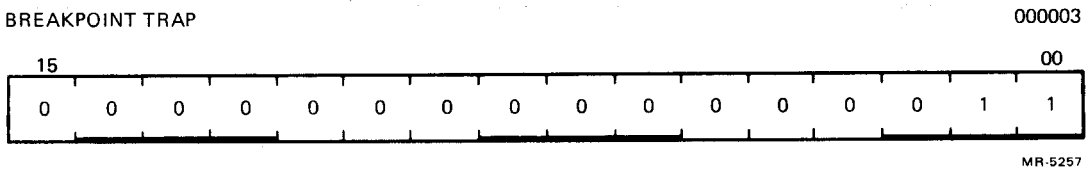
Operation:
↓ (SP) ← PS
↓ (SP) ← PC
PC ← (34)
PS ← (36)

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 DIGITAL software makes frequent use of EMT, the TRAP instruction is recommended for general use.

BPT

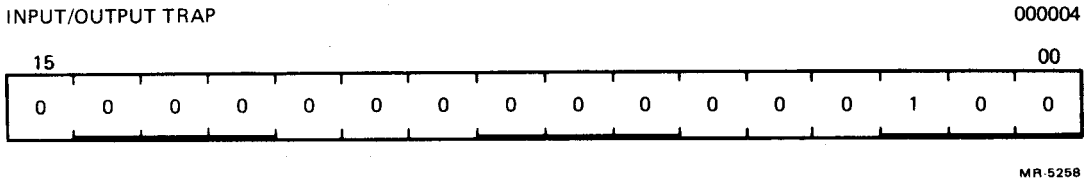


Operation:
↓ (SP) ← PS
↓ (SP) ← PC
PC ← (14)
PS ← (16)

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

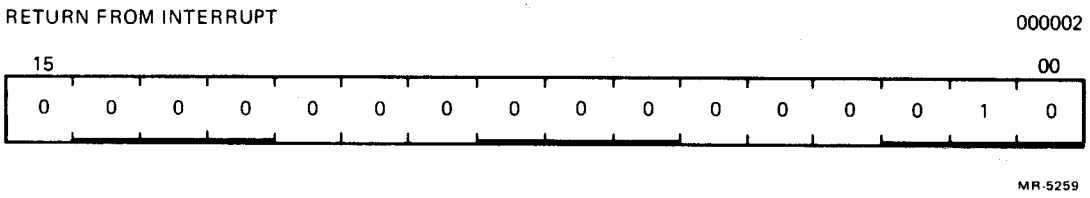


Operation:
↓ (SP) ← PS
↓ (SP) ← PC
PC ← (20)
PS ← (22)

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

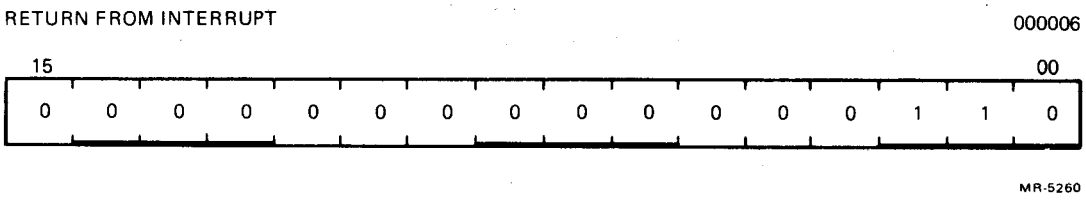


Operation: PC ← (SP) ↑
PS ← (SP) ↑

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 trap.

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 whereas RTI permits trace trap. If the new PS has the T bit set, a trap will occur after execution of the 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; they trap to vector address 4. Reserved instructions trap to the vector addresses as listed in Table A-14 in Appendix A.

6.3.5.7 Halt Interrupt – This is caused by the −HALT line (AI<7>). The −HALT interrupt saves the PC and PS and goes to the restart address with PS = 340_8.

6.3.5.8 Trace Trap – Trace trap is enabled by bit 4 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.

NOTE
Bit 4 of the PS can only be set indirectly by executing a RTI or RTT instruction with the desired PS on the stack.

6.3.5.9 Power Failure Interrupt – Occurs when the −PF line (AI<6>) is asserted. The vector for power failure is in locations 24 and 26. A trap will occur if an RTI instruction is executed in a power-fail service routine.

6.3.5.10 CP<3:0> Interrupts – Refer to Paragraph 1.5.3.

6.3.5.11 Special Cases of the 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 this instruction's execution. 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.

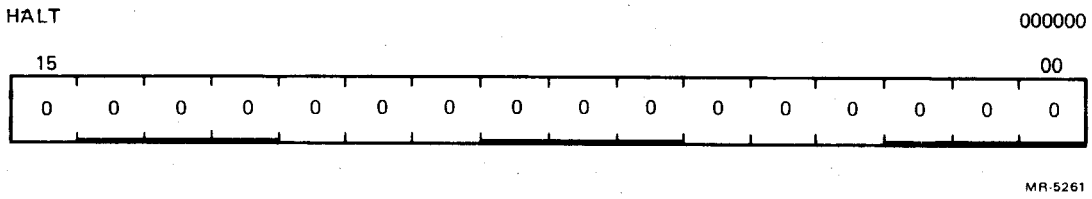
3. An instruction that caused an instruction trap – The instruction trap is performed and the entire routine for the service trap is executed. If the service routine exits with an RTI, or in any other way restores the stacked status word, the T bit is set again, the instruction following the traced instruction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

4. Interrupt trap priorities – In the 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

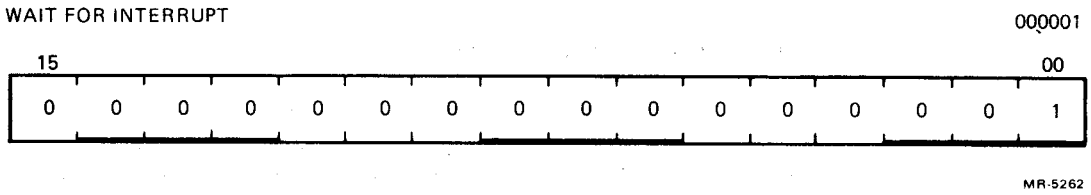


Operation:
↓ (SP) ← PS
↓ (SP) ← PC
PC ← restart address
PS ← 340

Condition Codes: Not affected

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

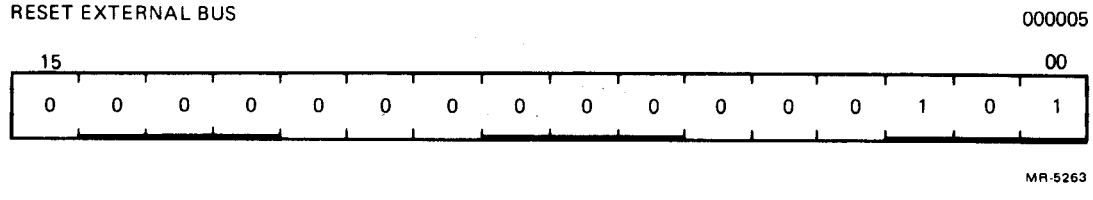
WAIT



Condition Codes: Not affected

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.

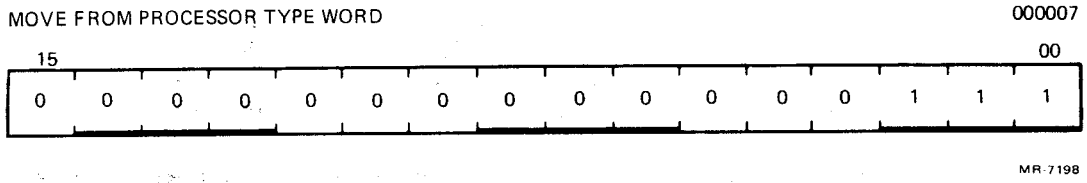
RESET



Condition Codes: Not affected

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

MFPT



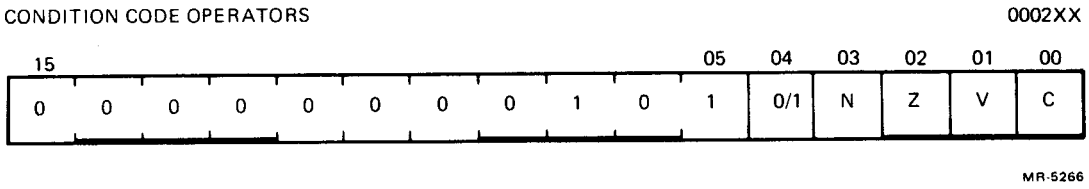
Operation: R0 ← 4

Condition Codes: Not affected

Description: The number 4 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



Description: Set and clear condition 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 4, the set/clear bit of the operator; i.e., set the bit specified by bit 0, 1, 2, or 3, if bit 4 = 1. 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
TABLES AND TIMING DIAGRAMS

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
−HALT*	X	X	X	X	8	–
−PF	X	X	X	X	8	24
	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 DC Characteristics

Absolute Maximum Ratings	
Pin voltages	−0.5 V to +7 V
Storage temperature range	−55° C to +125° C (−67° F to 257° F)
Maximum power dissipation	1.1 W
Chip ambient temperature operating range	0° C to 70° C (32° F 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 the device's reliability.

Static Characteristics

TA = 0° C to 70° C (32° F to 158° F), V_{CC} = 5.0 V ± 5%, V_{SS} = 0 V

Symbol	Parameter/Pins	Min.	Max.	Units	Comments and Conditions
I_{IL}	(Low input) Three-state leakage current on DAL<15:0>		−50	μA	V_{IN} = 0.4 V
I_{IL}	(High input) Three-state leakage current on DAL<15:0>		+10	μA	V_{IN} = V_{CC} max.
I_{IH}	(Min.) Input current for internal pull-ups on AI<7:0>, READY, DAL<15:7,2:0>	−0.1		mA	V_{IN} = 2.4 V
I_{IH}	(Max.) Input current for internal pull-ups on AI<7:0>, READY, DAL<15:7,2:0>		−0.1	mA	V_{IN} = 0.4 V
I_{CC}	Power supply current on V_{CC}		190	mA	T_{CYC} = 400 ns
I_{XLIH}	Input high current on XTL1		+700	μA	2.4 < V_{IN} < V_{CC}, XTL0 grounded
I_{XLIL}	Input low current on XTL1		−6.4	mA	−0.5 < V_{IN} < +0.8 V, XTL0 grounded
V_{IH}	Input high voltage on READY, DAL<15:0>, AI<7:0>	2	V_{CC}	V	
V_{IL}	Input low voltage on READY, DAL<15:0>, AI<7:0>	−0.5*	+0.8	V	
V_{OH}	Output high voltage for DAL<15:0>, COUT, PI, SEL1, SEL0	2.4		V	I_{OH} = 700 μA
V_{OHA}	Output high voltage for AI<7:0>	2.6		V	I_{OH} = −700 μA
V_{OHB}	Output high voltage for BCLR	2.2		V	I_{OH} = −700 μA terminated with 1K resistor to V_{SS}
V_{OHC}	Output high voltage for −RAS, −CAS, R/−WLB, R/−WHB	2.8		V	I_{OH} = −700 μA

Table A-2 DC Characteristics (Cont)

Symbol	Parameter/Pins	Min.	Max.	Units	Comments and Conditions
V_{OL}	Output low voltage for DAL<15:0>, AI<7:0>, COUT, PI, SEL1, SEL0, −BCLR, −RAS, −CAS, R/−WLB, R/−WHB	0.0	0.4	V	I_{OL} = 3.2 mA
V_{ILPUP}	Input low level for PUP	−0.5*	+0.8	V	
V_{IHPUP}	Input high level for PUP	1.6	V_{CC}	V	
V_{HY}	Hysteresis, PUP	0.6		V	
C_{IN}	Input capacitance for READY, DAL<15:0>, AI<7:0>		10	pF	
C_{OUT}	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

	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]	
MTPS 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][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... ...R–R–W–W[7] D R–R...	

[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 instruction of interrupt service routine.
[7] R–W (R–R–W–W) are indivisible.

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

		Signal Names							
Pin(s)	Pin Name	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*	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†	IACK + DMG	IACK + DMG			IACK + DMG			
25	SEL0†	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

*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.

†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

		Signal Names							
Pin(s)	Pin Name	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*	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†	IACK + DMG	IACK + DMG			IACK + DMG			
25	SEL0†	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

*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.

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

Table A-6 16-Bit Dynamic Write Addressing Scheme

Mode	Memory Chip	Address*	AI Used
4K/16K	4K × 1	A1–A12	<6:1>
4K/16K	16K × 1	A1–A14	<7:1>
64K	64K × 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

* – Internal low-current passive pull-ups.
N/A – Not applicable.

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 only).
* – Lines driven after address portion of transaction (8-bit and 16-bit modes).
1 – Dynamic modes only.
3S – Three-state.

Table A-12 Summary of DCT11-AA Instructions

Mnemonic	Op Code	Instruction	dst Result	N Z V C
SINGLE OPERAND				
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 complement)	–d	* * * *
TST(B)	■057DD	Test	d	* * 0 0
Rotate and Shift				
ROR(B)	■060DD	Rotate right	→ C, d	* * * *
ROL(B)	■061DD	Rotate left	C, d ←	* * * *
ASR(B)	■062DD	Arithmetic shift right	d/2	* * * *
ASL(B)	■063DD	Arithmetic 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 ∨ d	* * 0 –
XOR	074RDD	Exclusive (OR)	d ← r ∀ d	* * 0 –
BRANCH				
Mnemonic	**Base Code**	**Instruction**		**Branch Condition**
Branches				
BR	000400	Branch (unconditional)	(always)	
BNE	001000	Branch if not equal (to 0)	≠ 0	Z = 0
BEQ	001400	Branch if equal (to 0)	= 0	Z = 1
BPL	100000	Branch if plus	+	N = 0
BMI	100400	Branch if minus	–	N = 1
BVC	102000	Branch if overflow is clear		V = 0
BVS	102400	Branch if overflow is set		V = 1

Table A-12 Summary of DCT11-AA Instructions(Cont)

BCC	103000	Branch if carry is clear		C = 0
BCS	103400	Branch if carry is set		C = 1
Signed Conditional Branches				
BGE	002000	Branch if greater or equal	≥ 0	N ∀ V = 0
BLT	002400	Branch if less than (0)	< 0	N ∀ V = 1
BGT	003000	Branch if greater than (0)	> 0	Z ∨ (N ∀ V) = 0
BLE	003400	Branch if less or equal		Z ∨ (N ∀ V) = 1
Unsigned Conditional Branches			$\leq$	
BHI	101000	Branch if higher	$>$	C ∨ Z = 0
BLOS	101400	Branch if lower or same	$\leq$	C ∨ Z = 1
BHIS	103000	Branch if higher or same	$\geq$	C = 0
BLO	103400	Branch if lower	$<$	C = 1

JUMP and 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 and branch (if ≠ 0)	R − 1, then if R ≠ 0: PC ← Updated PC − (2 × NN)

TRAP and INTERRUPT

EMT	104000 to 104377	Emulator trap (not for general use)	PC at 30, PS at 32
TRAP	104400 to 104777	Trap	PC at 34, PS at 36
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 34 XXX	BCS, BLO
00 00 01	WAIT	00 54 DD	NEG	10 40 00	EMT
00 00 02	RTI	00 55 DD	ADC	through	
00 00 03	BPT	00 56 DD	SBC	10 43 77	
00 00 04	IOT	00 57 DD	TST	10 44 00	TRAP
00 00 05	RESET	00 60 DD	ROR	through	
00 00 06	RTT	00 61 DD	ROL	10 47 77	
00 00 07	MFPT	00 62 DD	ASR	10 50 DD	CLRB
00 00 77	Unused	00 63 DD		10 51 DD	COMB
00 01 DD	JMP	00 67 DD	SXT	10 52 DD	INCB
00 02 0R	RST	00 70 00	Unused	10 53 DD	DECB
00 02 10	Reserved	through		10 54 DD	NEGB
through		00 77 77		10 55 DD	ADCB
00 02 27		01 SS DD	MOV	10 56 DD	SBCB
00 02 40	NOP	02 SS DD	CMP	10 57 DD	TSTB
00 02 41	Condition	03 SS DD	BIT	10 60 DD	RORB
through	codes	04 SS DD	BIC	10 61 DD	ROLB
00 02 77		05 SS DD	BIS	10 62 DD	ASRB
00 03 DD	SWAB	06 SS DD	ADD	10 63 DD	ASLB
00 04 XXX	BR	07 50 40	Unused	10 64 SS	MTPS
00 10 XXX	BNE	through		10 67 DD	MFPS
00 14 XXX	BEQ	07 67 77		11 SS DD	MOVB
00 20 XXX	BGE	07 7R NN	SOB	12 SS DD	CMPB
00 24 XXX	BLT	10 00 XXX	BPL	13 SS DD	BITB
00 30 XXX	BGT	10 04 XXX	BMI	14 SS DD	BICB
00 34 XXX	BLE	10 10 XXX	BHI	15 SS DD	BISB
00 4R DD	JSR	10 14 XXX	BLOS	16 SS DD	SUB
00 50 DD	CLR	10 20 XXX	BVC	17 00 00	Reserved
00 51 DD	COM	10 24 XXX	BVS	through	
00 52 DD	INC	10 30 XXX	BCC, BHIS	17 77 77	

Table A-14 Reserved Trap and Interrupt Vectors

Vector	Description
000	Default vector = 0 for interrupting device failing to put vector out on DALs.
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

Tables A-17 to A-22 list the execution times for all instructions executable by the DCT11-AA. The tables are organized so as to help you calculate program execution times. To do such computations, you must first choose a system configuration and then find the columns in the tables that apply to it. Only those execution times listed may be used. 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. In 8-bit mode REFRESH is done every instruction cycle; in 16-bit mode it is done every other cycle. The refresh cycle adds a small increment of time to the machine cycle. Addressing modes 5, 6, and 7, I/O, and trap (two occurrences) also add time. Therefore, minimum and maximum execution times are given in REFRESH ON configurations. The program execution time is computed for REFRESH ON configurations by totaling the average execution times of the instructions used.

The following notes apply to Tables A-17 through A-22.

- All times are in microseconds.
- Add 0.4 μs for every $-$READY pulse that occurs during an I/O transaction.
- Operating frequency is 7.5 MHz. Use the following formula to compute instruction execution times (IETs) for different operating frequencies.

  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 tables.

- NA = Not applicable.

NOTE

The times calculated are those using revision 5.18 of the microcode.

Table A-17 XOR and Single-Operand Instructions

		16-Bit Mode			8-Bit Mode			
REFRESH		ON	ON	OFF	ON	ON	OFF	OFF
Instructions	Dest. Mode	Min.	Max.		Word Instr.	Byte Instr.	Word Instr.	Byte Instr.
CLR(B), COM(B),	0	1.60	1.73	1.6	2.53	2.53	2.4	2.4
INC(B), DEC(B),	1	2.80	2.93	2.8	5.33	3.73	5.2	3.6
NEG(B), ROR(B),	2	2.80	2.93	2.8	5.33	3.73	5.2	3.6
ROL(B), ASR(B),	3	3.60	3.73	3.6	6.93	5.33	6.8	5.2
ASL(B), SWAB,	4	3.20	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
TST (B)	0	1.60	1.73	1.6	2.53	2.53	2.4	2.4
	1	2.40	2.53	2.4	4.13	3.33	4.0	3.2
	2	2.40	2.53	2.4	4.13	3.33	4.0	3.2
	3	3.20	3.33	3.2	5.73	4.93	5.6	4.8
	4	2.80	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
MTPS	0	3.20	3.33	3.2	4.13	4.13	4.0	4.0
	1	4.00	4.13	4.0	4.93	4.93	4.8	4.8
	2	4.00	4.13	4.0	4.93	4.93	4.8	4.8
	3	4.80	4.93	4.8	6.53	6.53	6.4	6.4
	4	4.40	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

NOTE:

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					8-Bit Mode			
REFRESH		ON	ON	ON	ON	OFF	ON	ON	OFF	OFF
		Dst. Mode (0–4)		Dst. Mode (5–7)						
Instructions	Src. Mode	Min.	Max.	Min.	Max.		Word Instr.	Byte Instr.	Word Instr.	Byte Instr.
MOV(B), CMP(B),	0	1.20	1.33	1.33	1.33	1.2	2.13	2.13	2.0	2.0
ADD, SUB, BIT(B),	1	2.00	2.13	2.13	2.13	2.0	3.73	2.93	3.6	2.8
BIC(B), BIS(B)	2	2.00	2.13	2.13	2.13	2.0	3.73	2.93	3.6	2.8
BIT(B), BIC(B),	3	2.80	2.93	2.93	2.93	2.8	5.33	4.53	5.2	4.4
BIS(B)	4	2.40	2.53	2.53	2.53	2.4	4.13	3.33	4.0	3.2
	5	3.33	3.33	3.33	3.46	3.2	5.86	5.06	5.6	4.8
	6	3.33	3.33	3.33	3.46	3.2	5.86	5.06	5.6	4.8
	7	4.13	4.13	4.13	4.26	4.0	7.46	6.66	7.2	6.4

*Source mode times include instruction fetch, instruction decode, and source operand fetch.

Destination Mode Time†

		16-Bit Mode			8-Bit Mode			
REFRESH		ON	ON	OFF	ON	ON	OFF	OFF
Instructions	Dest. Mode	Min.	Max.		Word Instr.	Byte Instr.	Word Instr.	Byte Instr.
MOV(B), ADD,	0	0.4	0.4	0.4	0.40	0.40	0.4	0.4
SUB, BIC(B)	1	1.6	1.6	1.6	2.40	1.60	2.4	1.6
BIS(B)	2	1.6	1.6	1.6	2.40	1.60	2.4	1.6
	3	2.4	2.4	2.4	4.00	3.20	4.0	3.2
	4	2.0	2.0	2.0	2.80	2.00	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	0.40	0.40	0.4	0.4
	1	1.2	1.2	1.2	2.00	1.20	2.0	1.2
	2	1.2	1.2	1.2	2.00	1.20	2.0	1.2
	3	2.0	2.0	2.0	3.60	2.80	3.6	2.8
	4	1.6	1.6	1.6	2.40	1.60	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 are no outputs).

Table A-19 Jump and Subroutine Instructions

		16-Bit Mode			8-Bit Mode			
REFRESH		ON	ON	OFF	ON	ON	OFF	OFF
Instructions	Dest. Mode	Min.	Max.		Word Instr.	Byte Instr.	Word Instr.	Byte Instr.
JMP	1	2.00	2.13	2.0	2.93	NA	2.8	NA
	2	2.40	2.53	2.4	3.33	NA	3.2	NA
	3	2.40	2.53	2.4	4.13	NA	4.0	NA
	4	2.40	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	3.60	3.73	3.6	5.33	NA	5.2	NA
	2	4.00	4.13	4.0	5.73	NA	5.6	NA
	3	4.00	4.13	4.0	6.53	NA	6.4	NA
	4	4.00	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	2.80	2.93	2.8	4.53	NA	4.4	NA
SOB	NA	2.40	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 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, 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, and Interrupt Instructions

		16-Bit Mode			8-Bit Mode			
REFRESH		ON	ON	OFF	ON	ON	OFF	OFF
Instructions	Dest. Mode	Min.	Max.		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	1.60	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	3.20	3.33	3.2	4.93	NA	4.8	NA
RTT	NA	4.40	4.53	4.4	7.13	NA	7.0	NA

NOTES:

1. Branch instruction 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 instruction 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 instruction execution times include instruction fetch, instruction decode, and popping the PC and PS from the stack.

Table A-21 Miscellaneous and Condition Code Instructions

		16-Bit Mode			8-Bit Mode			
REFRESH		ON	ON	OFF	ON	ON	OFF	OFF
Instructions	Dest. Mode	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	1.60	1.73	1.6	2.43	NA	2.4	NA
RESET	NA	14.60	14.73	14.6	16.53	NA	16.4	NA
NOP	NA	2.40	2.53	2.4	3.33	NA	3.2	NA
CLC, CLV, CLZ, CLN, CCC, SEC, SEV, SEZ, SEN, SCC	NA	2.40	2.53	2.4	3.33	NA	3.2	NA
MFPT	NA	2.00	2.13	2.0	2.93	NA	2.8	NA

NOTES:

1. HALT execution times include instruction fetch, instruction decode, writing the PC and PS onto 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 μs loop pulsing PI. (If REFRESH is on the loop will be 1.33 μs maximum). 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 instruction 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

		16-Bit Mode		8-Bit Mode	
Active Inputs	**Dest. 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:

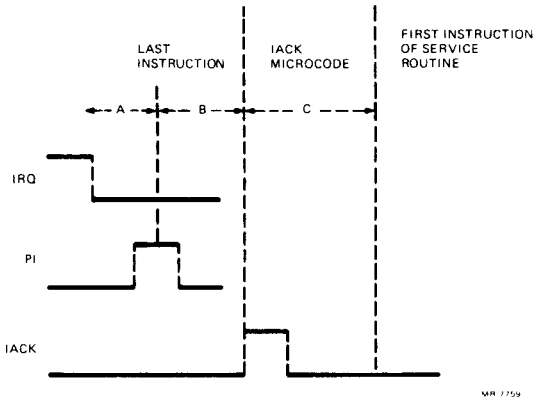
These timings are given in microseconds and assume a clock frequency of 7.5 MHz.

1. Interrupt latency is measured from the time the interrupt request is asserted either on the AI lines (in static modes) or on the input of the AI line driver (in dynamic modes) to the time the DCT11-AA is ready to fetch the first instruction in the interrupt's service routine. During this time the DCT11-AA:

   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.



   Note that the time to synchronize the IRQ and perform any external priority arbitration is not included in the interrupt latency.

2. DMG latency is calculated from the time DMR is valid on the input of the AI line driver to the time the DCT11-AA asserts DMG.

3. WAIT instruction latencies are the maximum encountered in the instruction's execution state. These times do not include the instruction fetch or the instruction decode.

4. Times refer to IRQ occurring during a JSR (mode 2 or 4) EMT sequence, which is the worst case.
5. Times refer to DMR occurring during a MTPS (mode 0) instruction, which is the worst case.
6. Timings assume the DCT11-AA is not in long bus cycle (mode register bit 1) and there are no ready slips.

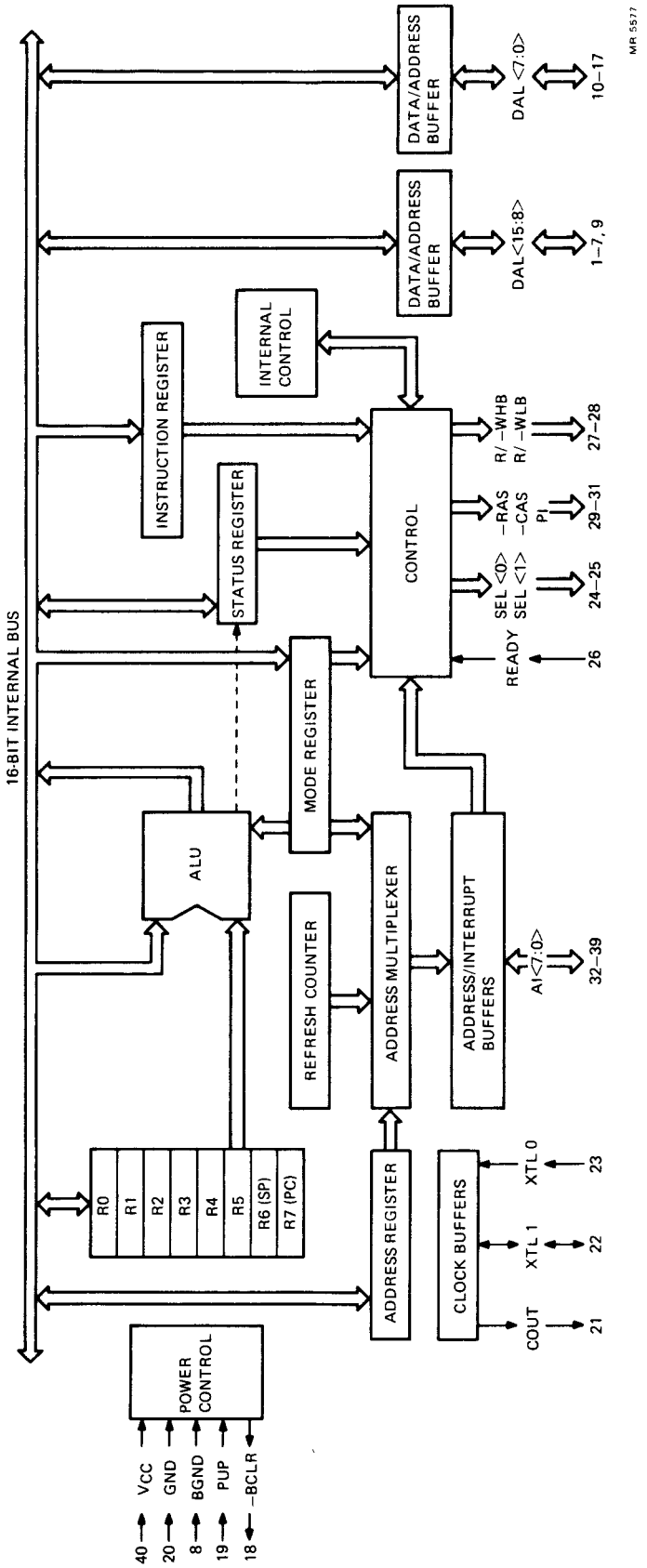
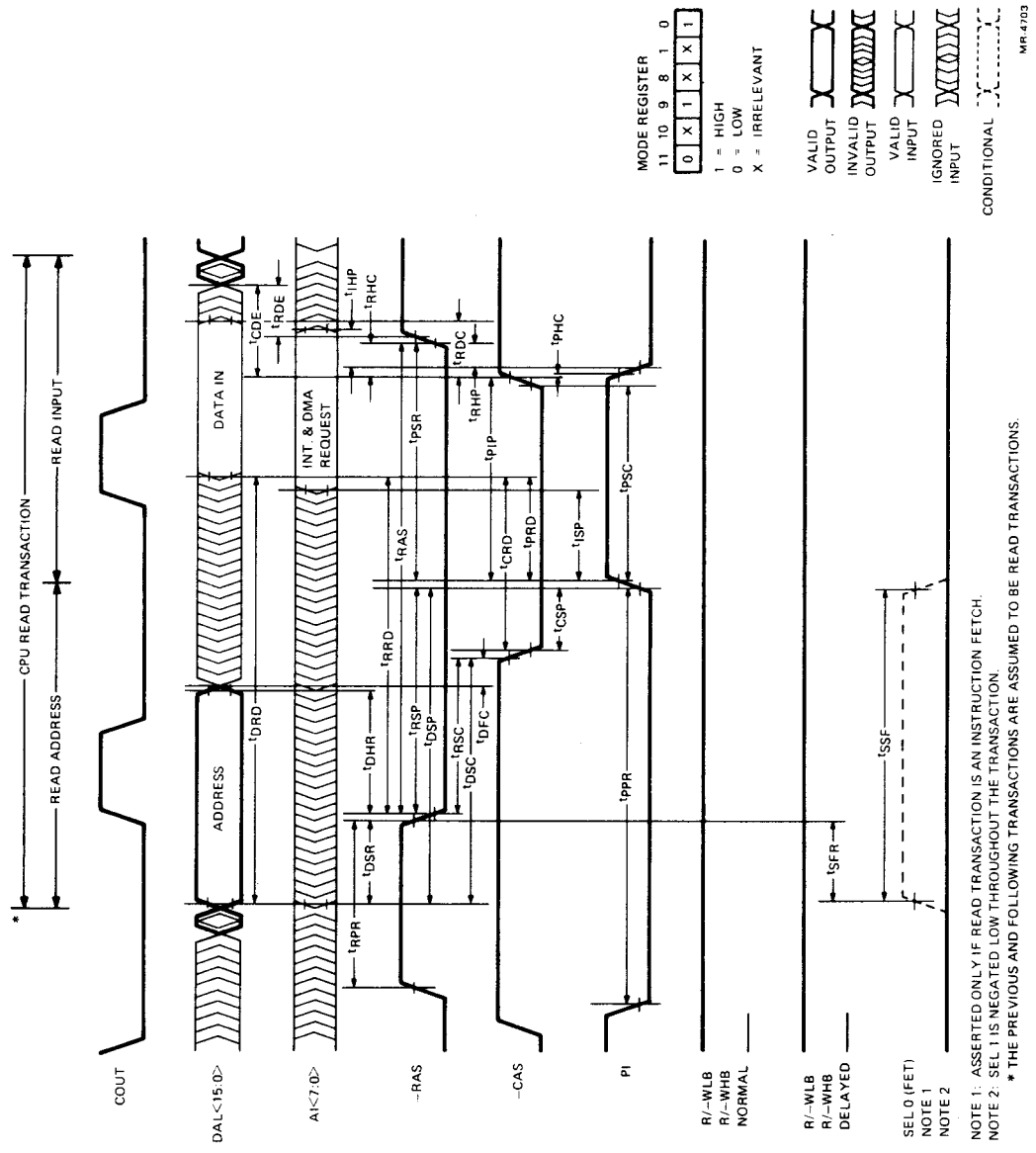


Figure A-1 DCT11-AA, Block Diagram



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

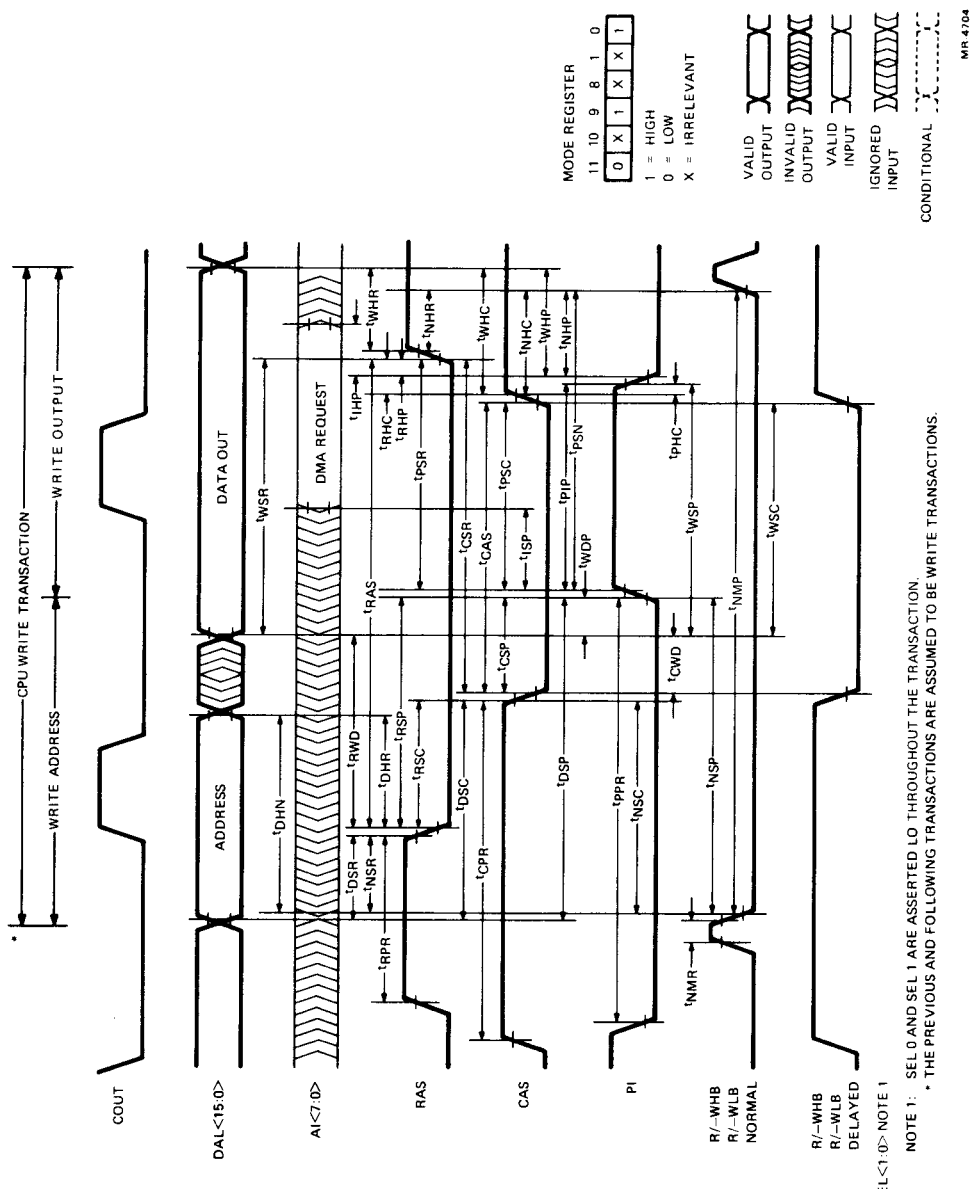


Figure A-3 16-Bit Static Write

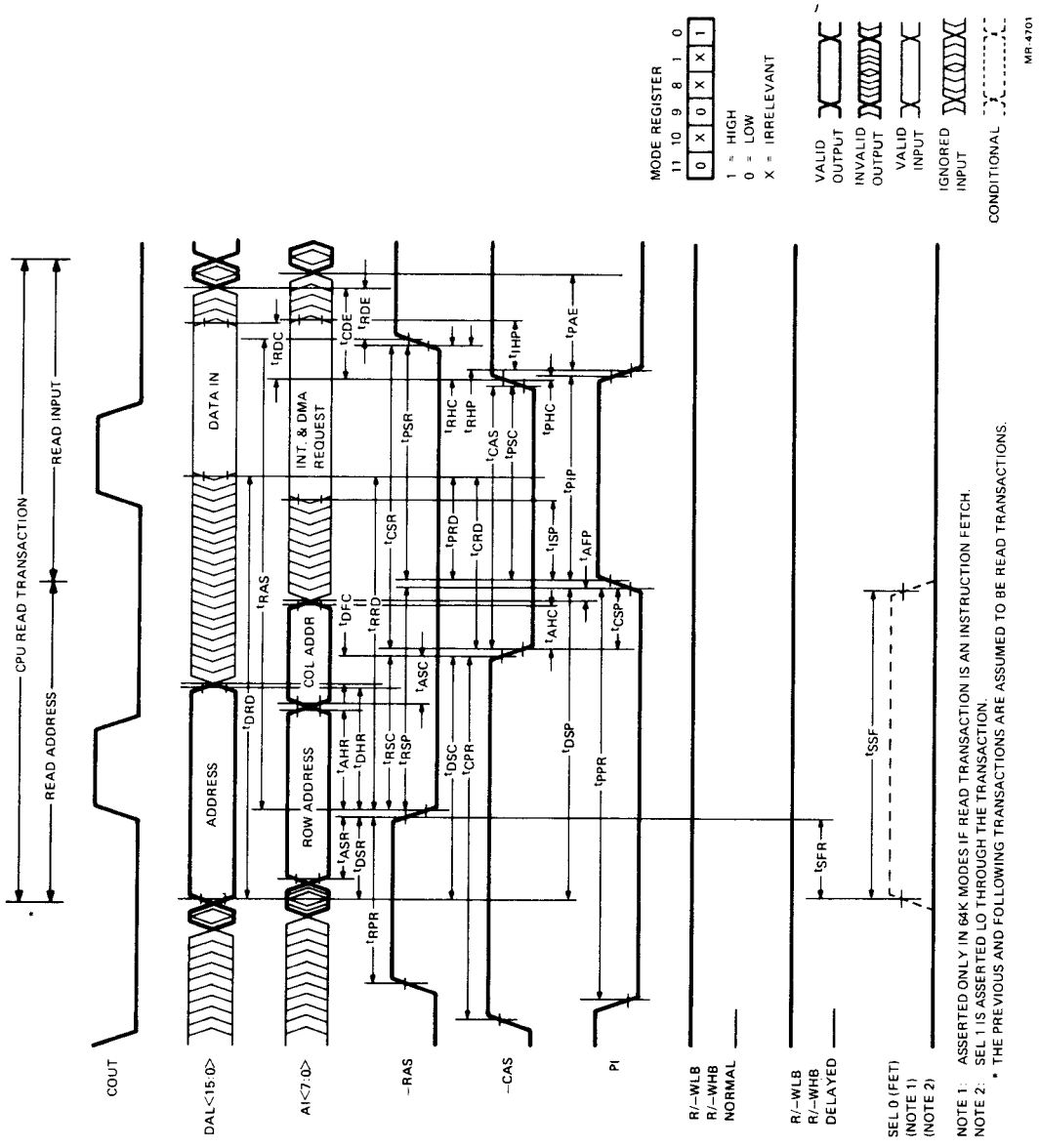
	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tCAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^tCPR	–CAS Precharge Time	(3T –5) = 395 ns	min
	^tCSP	–CAS (L.E.) or Delayed Mode R/$\overline{W}$ (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
1	^tCSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^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
	^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

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
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
	^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-5582

PRELIMINARY



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 ^{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 –68) = 65 ns)	min
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}CPR	–CAS Precharge Time	(3T –5) = 395 ns	min
1	^{t}CRD	–CAS (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
1	^{t}CSR	–CAS (L.E.) Set Up Time to –RAS (T.E.)	(3T –40) = 360 ns	min
	^{t}CYC	XTLI, XTLO 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}PAE	PI (T.E.) to next AI<7:0> Address Enable	(T –40) = 93 ns	min
	^{t}PHC	PI Hold Time from –CAS (T.E.)	10 ns	min
1	^{t}PIP	PI Pulse Width	(2T –37) = 230 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.)	(2T –75) = 192 ns	min
1	^{t}PSR	PI (L.E.) Set Up Time to –RAS (T.E.)	(2T + 24) = 291 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	Fetch SEL<0> (L.E.) Set Up Time to –RAS (L.E.)	(T –23) = 110 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-5579

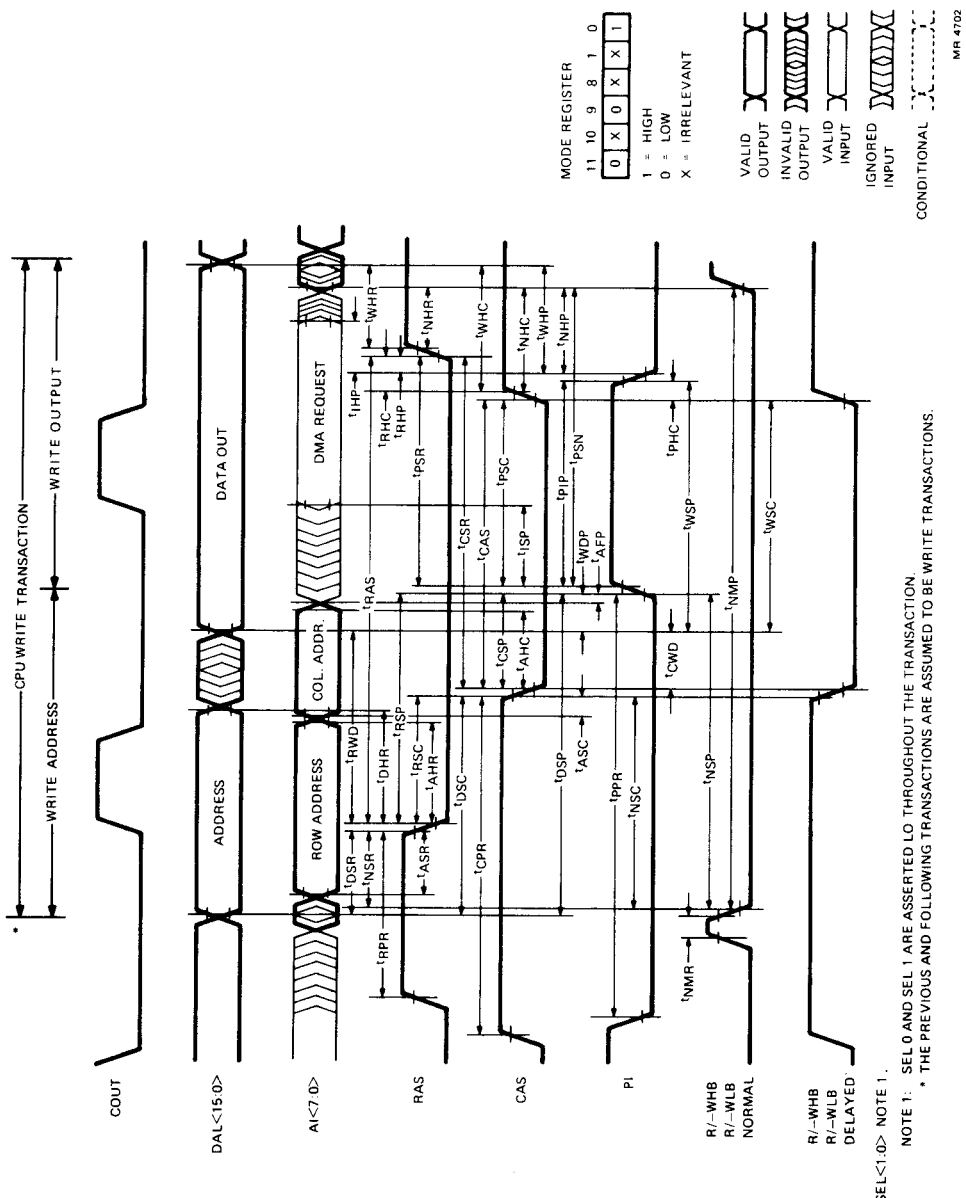


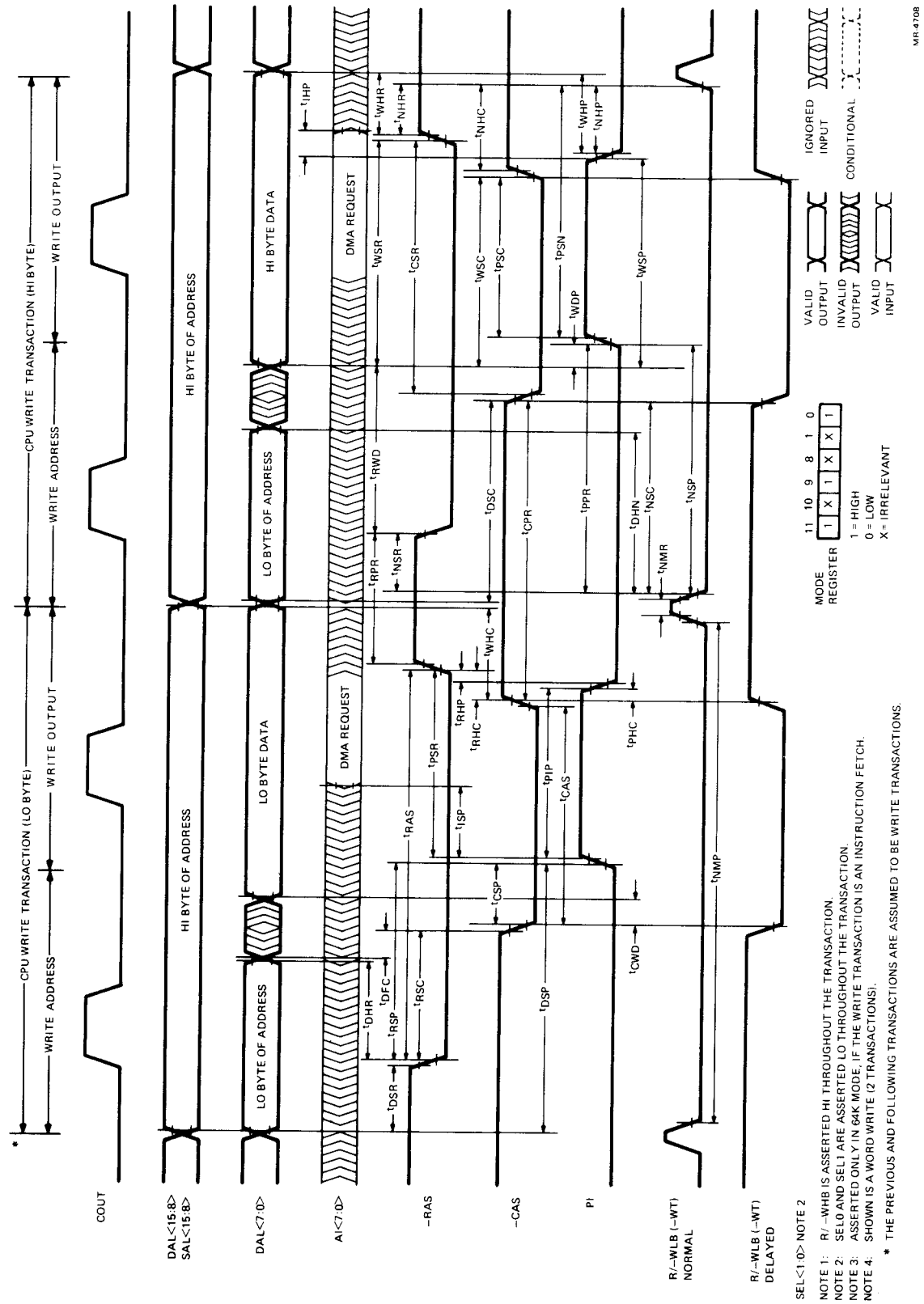
Figure A-5 16-Bit Dynamic Write

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tCAS	–CAS Pulse Width	(3T –90) = 310 ns	min
1	^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
	^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
	^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
	^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
1	^tNRD	Normal Mode R/$\overline{W}$ Set Up Time to Read Data Valid	(5T –148) = 519 ns	max
	^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
	^tPHC	PI Hold Time from –CAS (T.E.) or Delayed Mode R/$\overline{W}$ (T.E.)	10 ns	min

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
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
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
	^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
	^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.

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 ^tCYC	MIN/MAX
	^tCAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^tCPR	–CAS Precharge Time	(3T –5) = 395 ns	min
	^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
	^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
	^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
	^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
	^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
	^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
2	^tWSC	Write Data Set Up Time to –CAS (T.E.)	(3T –150) = 250 ns	min
2	^tWSP	Write Data Set Up Time to PI (T.E.)	(3T –110) = 290 ns	min
2	^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.

2: Add 4T for each READY pulse.

MR-5586

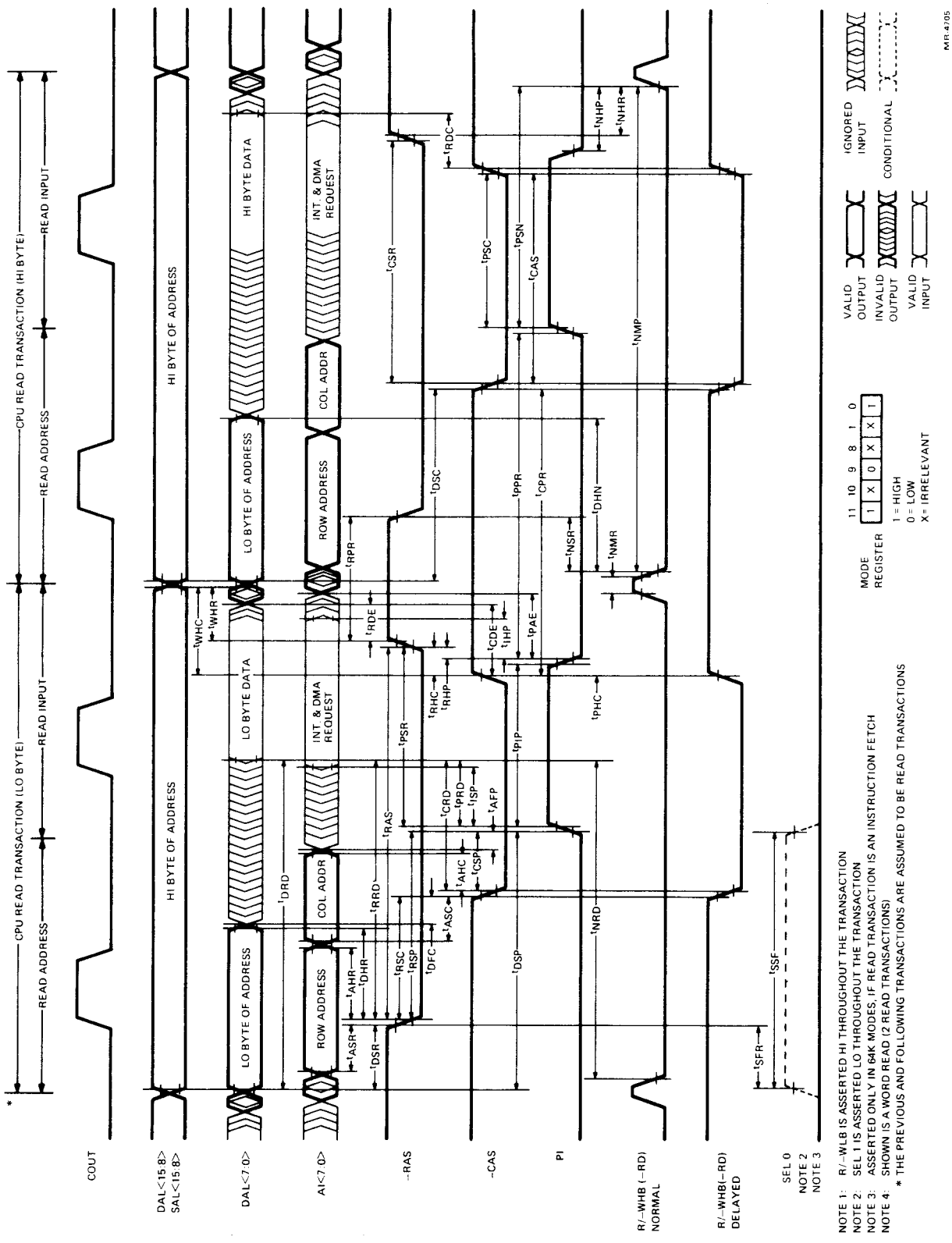


Figure A-8 8-Bit Dynamic Read

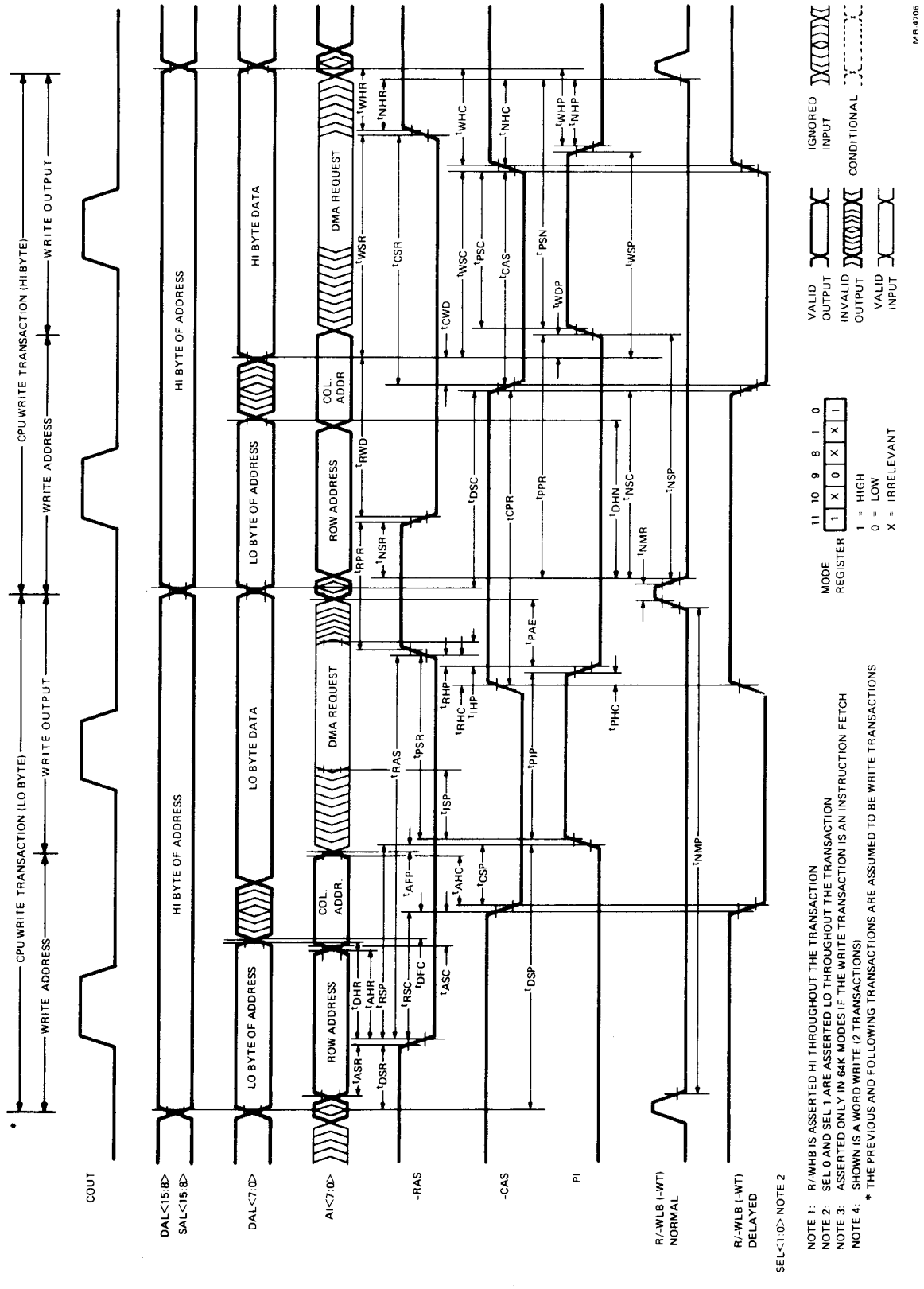
	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tAFP	Column Address on AI<7:0> Float to PI (L.E.)	0 ns	min
	^tAHC	Column Address on AI<7:0> Hold Time from −CAS (L.E.)	(T −43) = 90 ns	min
	^tAHR	Row Address on AI<7:0> Hold Time from −RAS (L.E.)	(T −60) = 73 ns	min
	^tASC	Column Address on AI<7:0> Set Up 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

PRELIMINARY



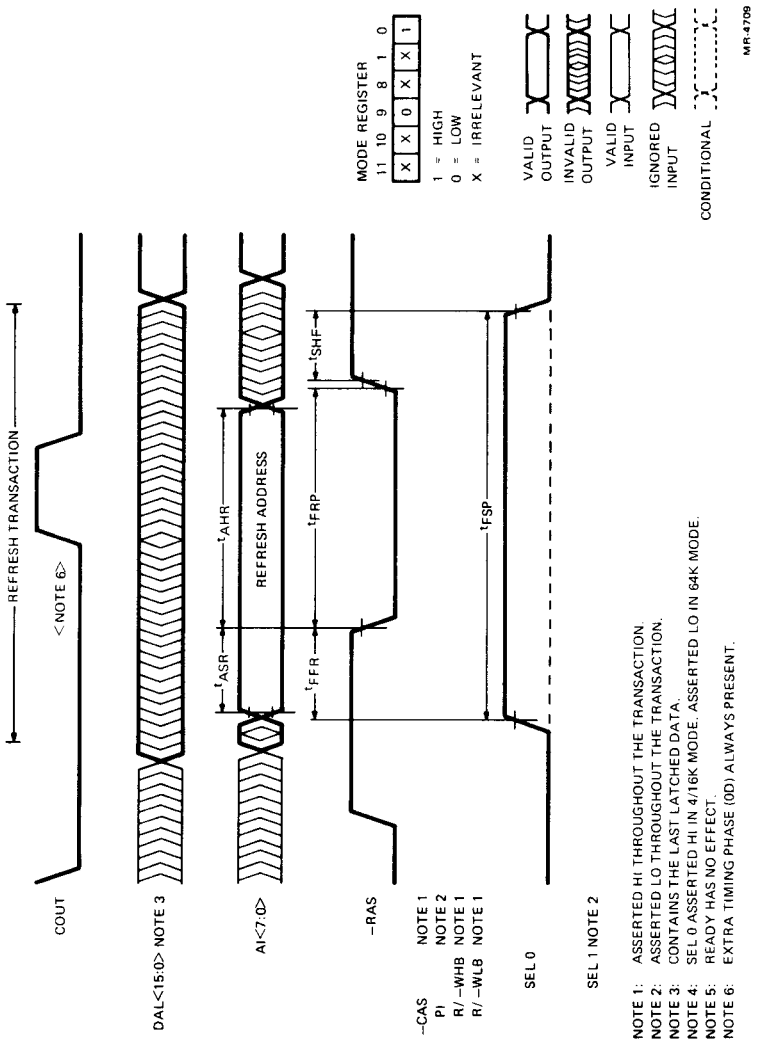
NOTE 1: R/-WHB IS ASSERTED HI THROUGHOUT THE TRANSACTION
NOTE 2: SEL 0 AND SEL 1 ARE ASSERTED LO THROUGHOUT THE TRANSACTION
NOTE 3: ASSERTED ONLY IN 64K MODES IF THE WRITE TRANSACTION IS AN INSTRUCTION FETCH
NOTE 4: SHOWN IS A WORD WRITE (2 TRANSACTIONS)
* THE PREVIOUS AND FOLLOWING TRANSACTIONS ARE ASSUMED TO BE WRITE TRANSACTIONS

Figure A-9 8-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
	^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
	^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
	^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
	^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
	^tPAE	PI (T.E.) to Next AI <7:0> Address Enable	(T –40) = 93 ns	min
	^tPHC	PI Hold Time from –CRS (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
	^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 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
1	^tWSC	Write Data Set Up Time to –CAS (T.E.)	(3T –150) = 250 ns	min
1	^tWSP	Write Data Set Up Time to PI (T.E.)	(3T –110) = 290 ns	min
1	^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-5584

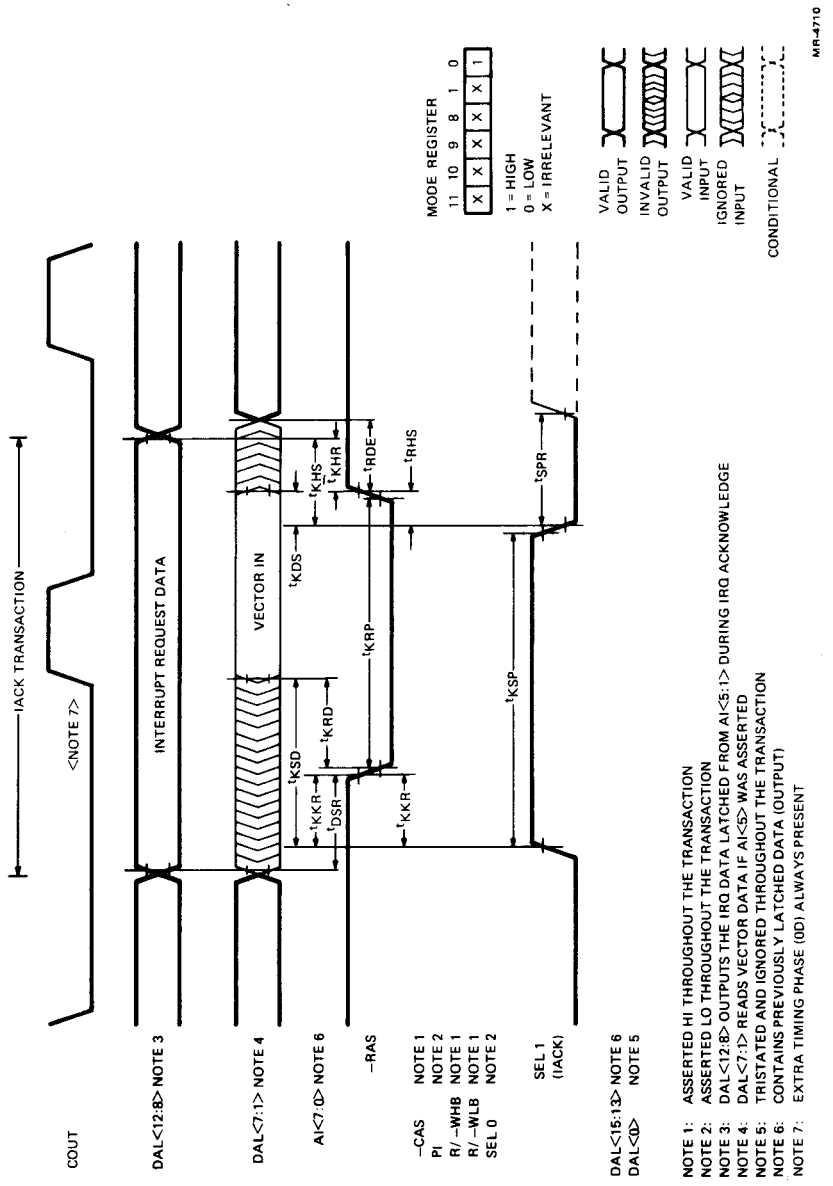


NOTE 1: ASSERTED HI THROUGHOUT THE TRANSACTION.
NOTE 2: ASSERTED LO THROUGHOUT THE TRANSACTION.
NOTE 3: CONTAINS THE LAST LATCHED DATA.
NOTE 4: SEL 0 ASSERTED HI IN 4/16K MODE. ASSERTED LO IN 64K MODE.
NOTE 5: READY HAS NO EFFECT.
NOTE 6: EXTRA TIMING PHASE (0D) ALWAYS PRESENT.

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



NOTE 1: ASSERTED HI THROUGHOUT THE TRANSACTION
NOTE 2: ASSERTED LO THROUGHOUT THE TRANSACTION
NOTE 3: DAL<12:8> OUTPUTS THE IRQ DATA LATCHED FROM AI<5:1> DURING IRQ ACKNOWLEDGE
NOTE 4: DAL<7:1> READS VECTOR DATA IF AI<5> WAS ASSERTED
NOTE 5: TRISTATED AND IGNORED THROUGHOUT THE TRANSACTION
NOTE 6: CONTAINS PREVIOUSLY LATCHED DATA (OUTPUT)
NOTE 7: EXTRA TIMING PHASE (0D) ALWAYS PRESENT

Figure A-11 IACK Transaction

	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tCYC	XTL1, XTL0 Operating Period	T = 133 ns	min
	^tDSR	IACK Data Set Up Time	(T −48) = 85 ns	min
	^tKDS	Vector Data on DAL<7:2> Hold Time from IACK Select on SEL<1> (T.E.)	0 ns	min
1	^tKHR	IACK info on DAL<15:8> Hold Time from −RAS (T.E.)	(T −118) = 15 ns	min
	^tKHS	IACK info on DAL<15:8> Hold Time from IACK Select on SEL<1> (T.E.)	(T −50) = 83 ns	min
	^tKKR	IACK Select on SEL<1> (L.E.) Set Up Time to −RAS (L.E.)	(T−63) = 70 ns	min
1	^tKRD	IACK −RAS (L.E.) to Vector Data on DAL<7:0> Valid	(2T −148) = 119 ns	max
1	^tKRP	IACK −RAS Pulse Width	(2T + 35) = 302 ns	min
1	^tKSD	IACK Select on SEL<1> (L.E.) to Vector Data on DAL<7:2> Valid	(3T −155) = 245 ns	max
1	^tKSP	IACK Select on SEL<1> Pulse Width	(3T −66) = 334 ns	min
	^tRDE	−RAS (T.E.) to next DAL<15:0> Address Enable	(T −118) = 15 ns	min
	^tRHS	−RAS (T.E.) Hold Time from IACK Select on SEL<1> (T.E.)	45 ns	min
	^tSPR	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

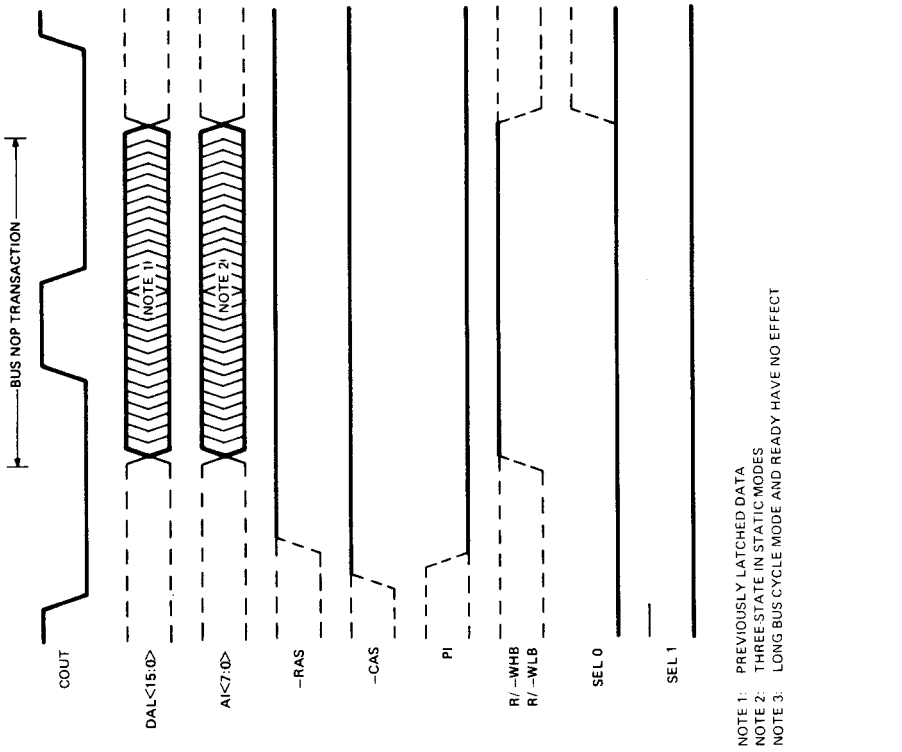
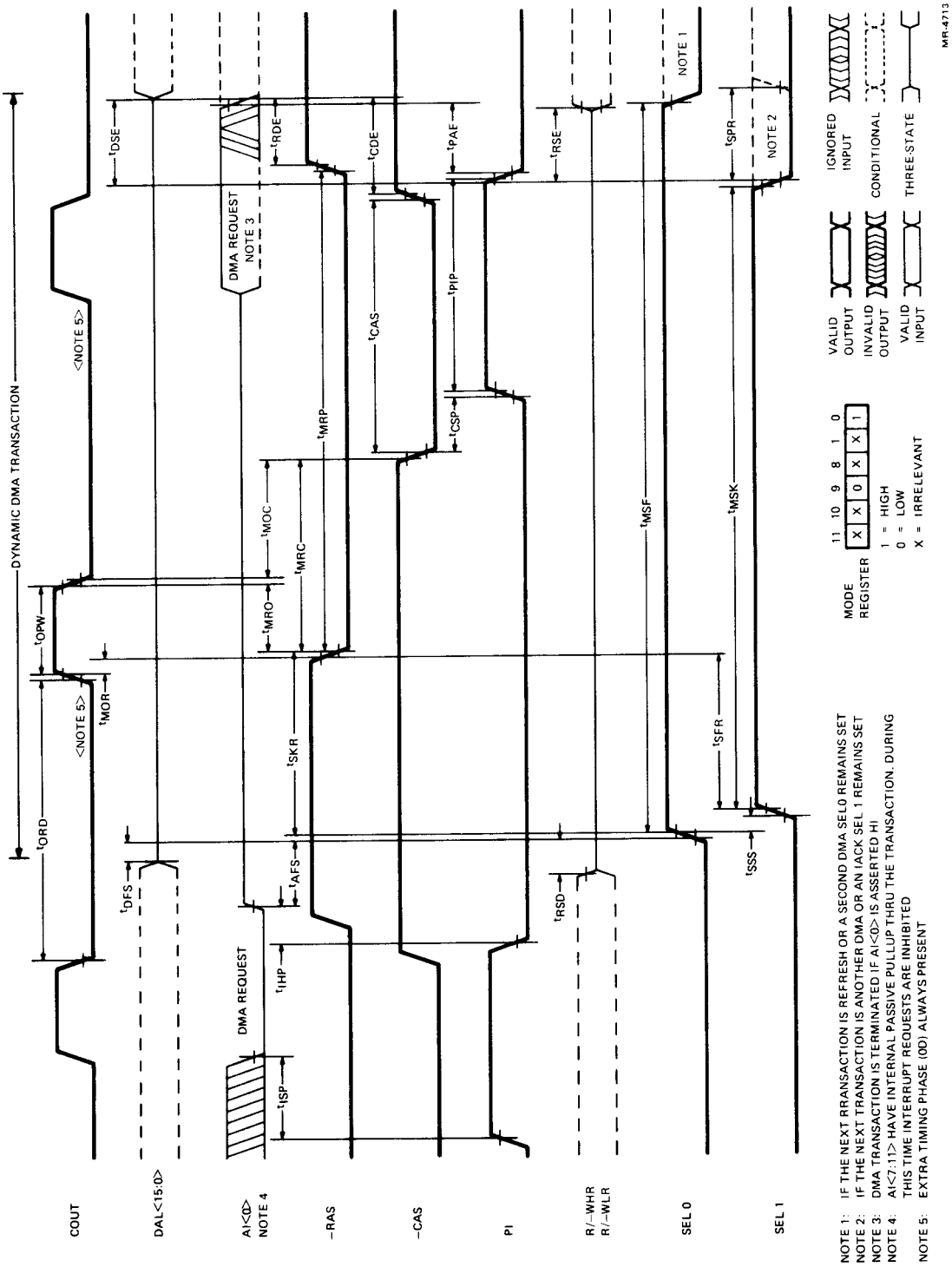


Figure A-12 Busnop Transaction

(A Busnop transaction is a specific state; therefore, no timings are provided.)



NOTE 1: IF THE NEXT RRANSACTION IS REFRESH OR A SECOND DMA SEL0 REMAINS SET
NOTE 2: IF THE NEXT TRANSACTION IS ANOTHER DMA OR AN IACK SEL 1 REMAINS SET
NOTE 3: DMA TRANSACTION IS TERMINATED IF AI<0> IS ASSERTED HI
NOTE 4: AI<7:1> HAVE INTERNAL PASSIVE PULLUP THRU THE TRANSACTION. DURING THIS TIME INTERRUPT REQUESTS ARE INHIBITED
NOTE 5: EXTRA TIMING PHASE (0D) ALWAYS PRESENT

Figure A-13 DMA Transaction

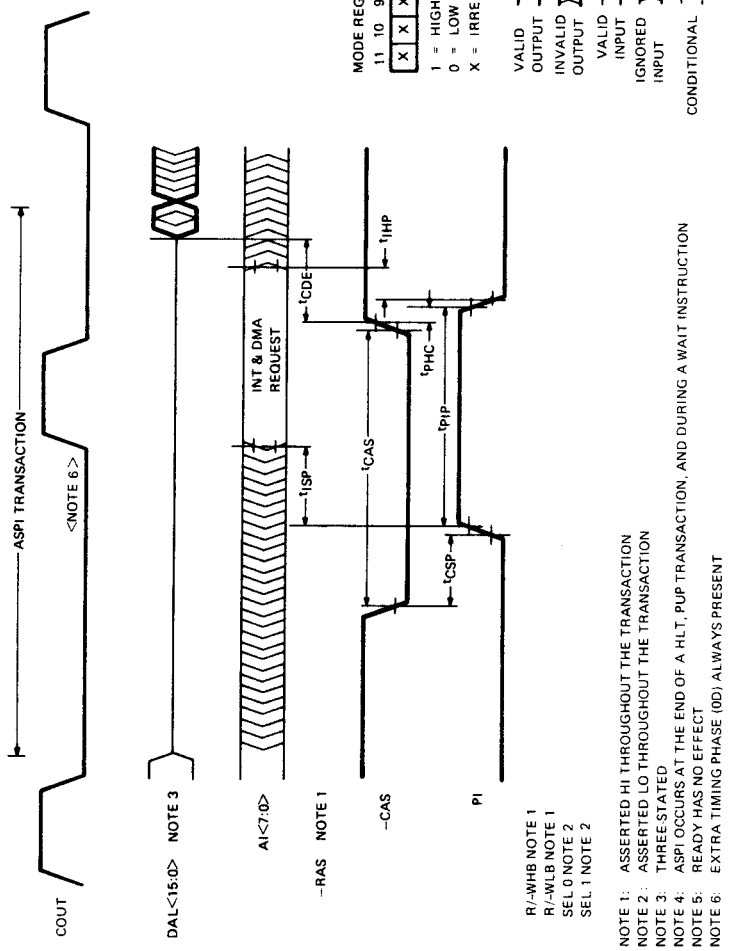
	SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
	^tAFS	AI<7:0> Float to DMA Select on SEL<0>	0 ns	min
1	^tCAS	–CAS Pulse Width	(3T –90) = 310 ns	min
	^tCDE	–CAS (T.E.) to next DAL<15:0> Address Enable	(T –18) = 115 ns	min
	^tCSP	–CAS (L.E.) Set Up Time to PI (L.E.)	(T –28) = 105 ns	min
	^tCYC	XTL1, XTL0 Operating 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
	^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
	^tMRP	DMA Select –RAS Pulse Width	(5T + 35) = 702 ns	min
	^tMSF	DMA Select on SEL<0> Pulse Width	(8T –38) = 1029 ns	min
2	^tMSK	DMA Select on SEL<1> Pulse Width	(7T –68) = 865 ns	min
2,3	^tOPW	Pulse Mode COUT Pulse Width	(T –33) = 100 ns	min
2,3	^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
	^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
1	^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
	^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
	^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



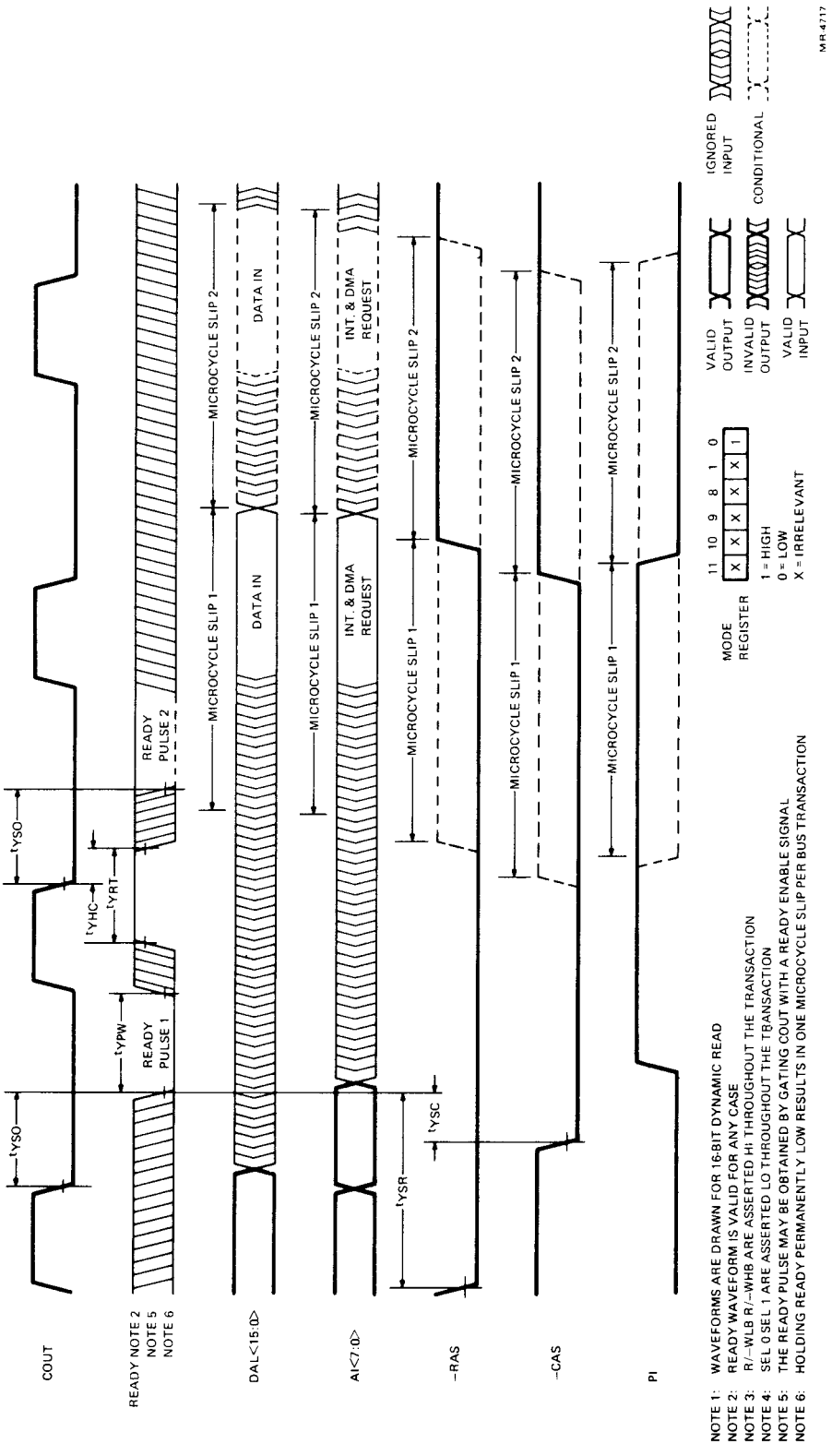
NOTE 1: ASSERTED HI THROUGHOUT THE TRANSACTION
NOTE 2: ASSERTED LO THROUGHOUT THE TRANSACTION
NOTE 3: THREE STATED
NOTE 4: ASPI OCCURS AT THE END OF A HLT, PUP TRANSACTION, AND DURING A WAIT INSTRUCTION
NOTE 5: READY HAS NO EFFECT
NOTE 6: EXTRA TIMING PHASE (0D) ALWAYS PRESENT

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



NOTE 1: WAVEFORMS ARE DRAWN FOR 16-BIT DYNAMIC READ
NOTE 2: READY WAVEFORM IS VALID FOR ANY CASE
NOTE 3: R/–WLB R/–WHB ARE ASSERTED HI THROUGHOUT THE TRANSACTION
NOTE 4: SEL 0 SEL 1 ARE ASSERTED LO THROUGHOUT THE TRANSACTION
NOTE 5: THE READY PULSE MAY BE OBTAINED BY GATING COUT WITH A READY ENABLE SIGNAL
NOTE 6: HOLDING READY PERMANENTLY LOW RESULTS IN ONE MICROCYCLE SLIP PER BUS TRANSACTION

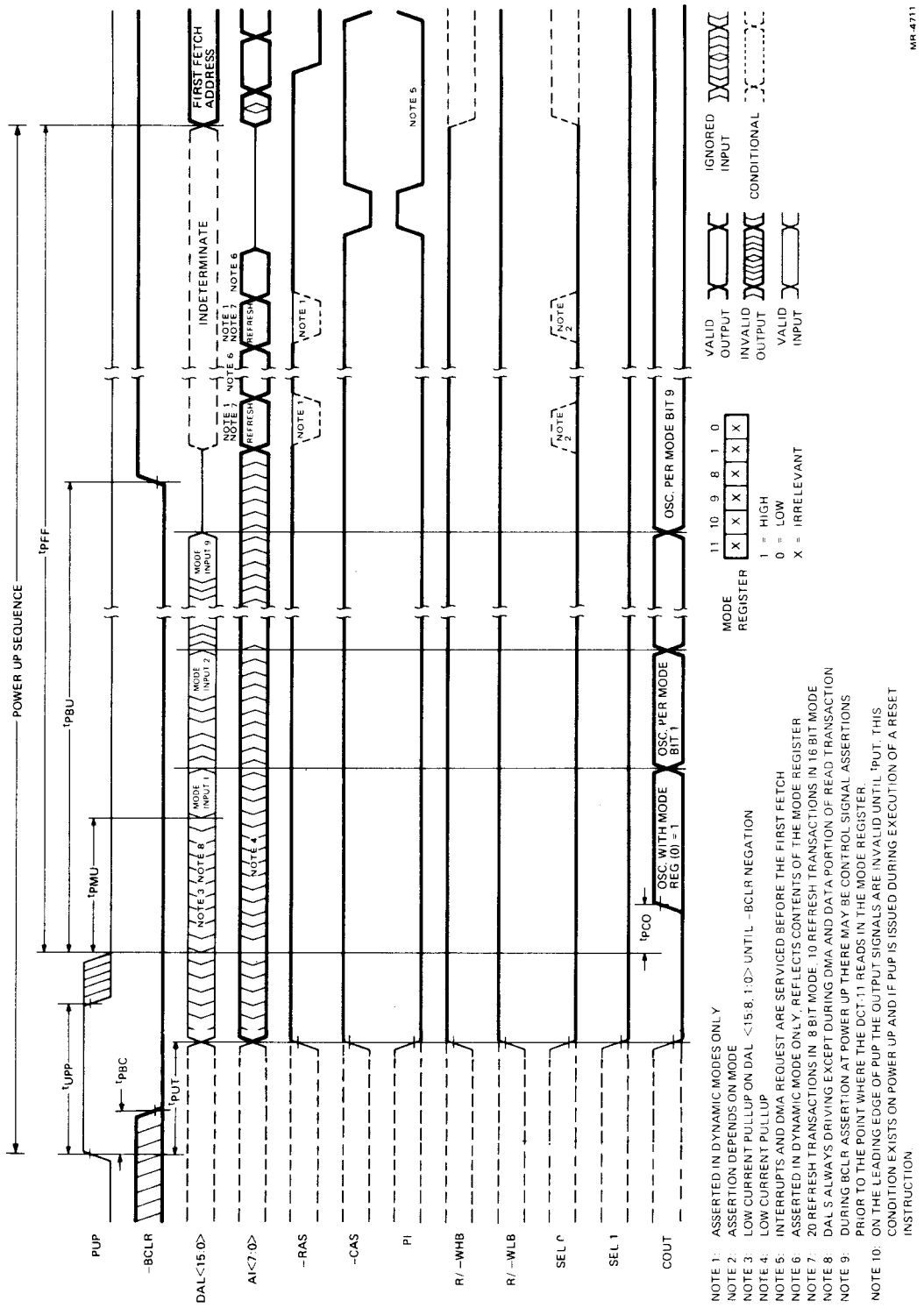
Figure A-15 Ready

	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

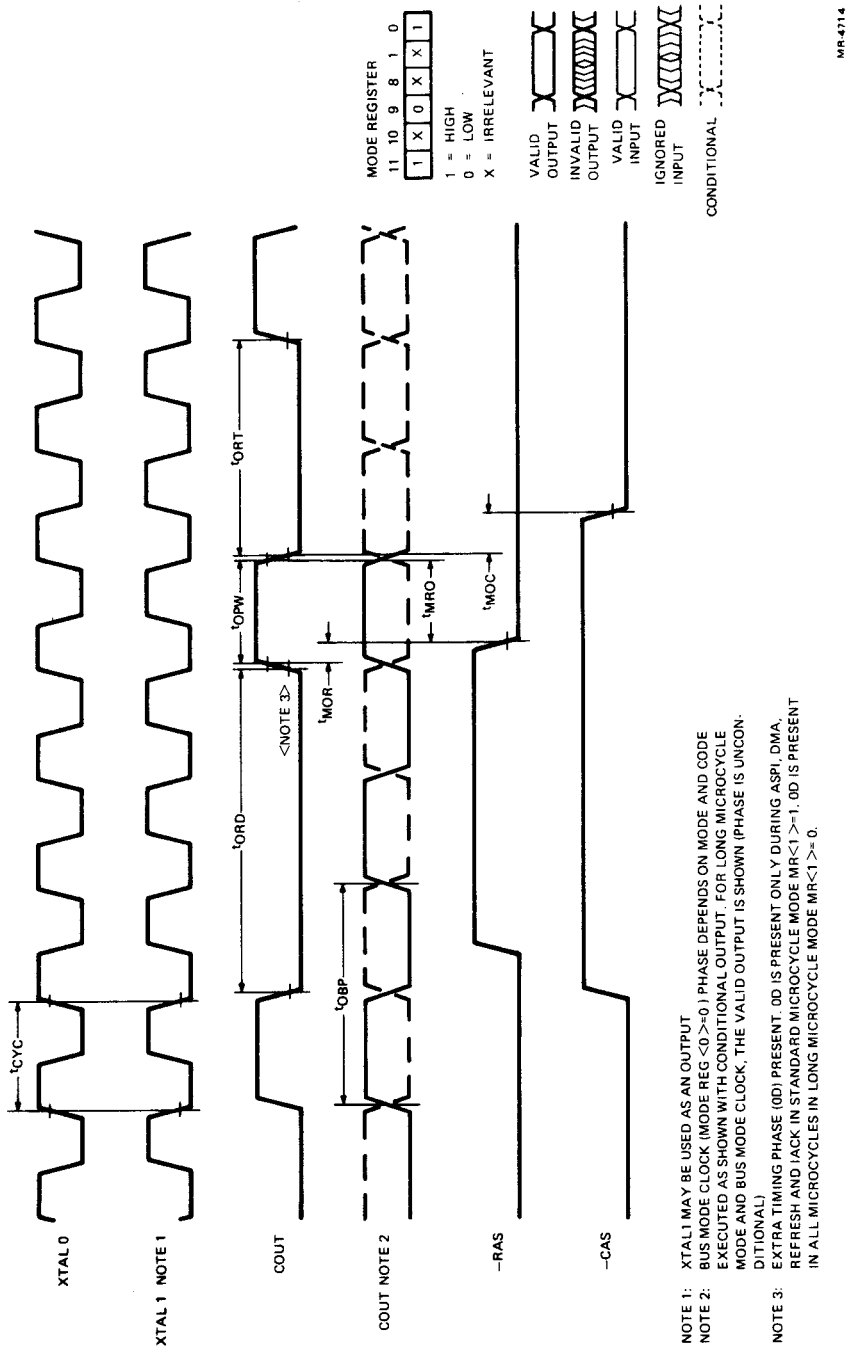


NOTE 1: ASSERTED IN DYNAMIC MODES ONLY
NOTE 2: ASSERTION DEPENDS ON MODE
NOTE 3: LOW CURRENT PULLUP ON DAL <15:8, 1:0> UNTIL -BCLR NEGATION
NOTE 4: LOW CURRENT PULLUP
NOTE 5: INTERRUPTS AND DMA REQUEST ARE SERVICED BEFORE THE FIRST FETCH
NOTE 6: ASSERTED IN DYNAMIC MODE ONLY, REFLECTS CONTENTS OF THE MODE REGISTER
NOTE 7: 20 REFRESH TRANSACTIONS IN 8 BIT MODE 10 REFRESH TRANSACTIONS IN 16 BIT MODE
NOTE 8: DAL S ALWAYS DRIVING EXCEPT DURING DMA AND DATA PORTION OF READ TRANSACTION
NOTE 9: DURING BCLR ASSERTION AT POWER UP THERE MAY BE CONTROL SIGNAL ASSERTIONS PRIOR TO THE POINT WHERE THE DCT-11 READS IN THE MODE REGISTER.
NOTE 10: ON THE LEADING EDGE OF PUP THE OUTPUT SIGNALS ARE INVALID UNTIL TPUT. THIS CONDITION EXISTS ON POWER UP AND IF PUP IS ISSUED DURING EXECUTION OF A RESET INSTRUCTION.

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

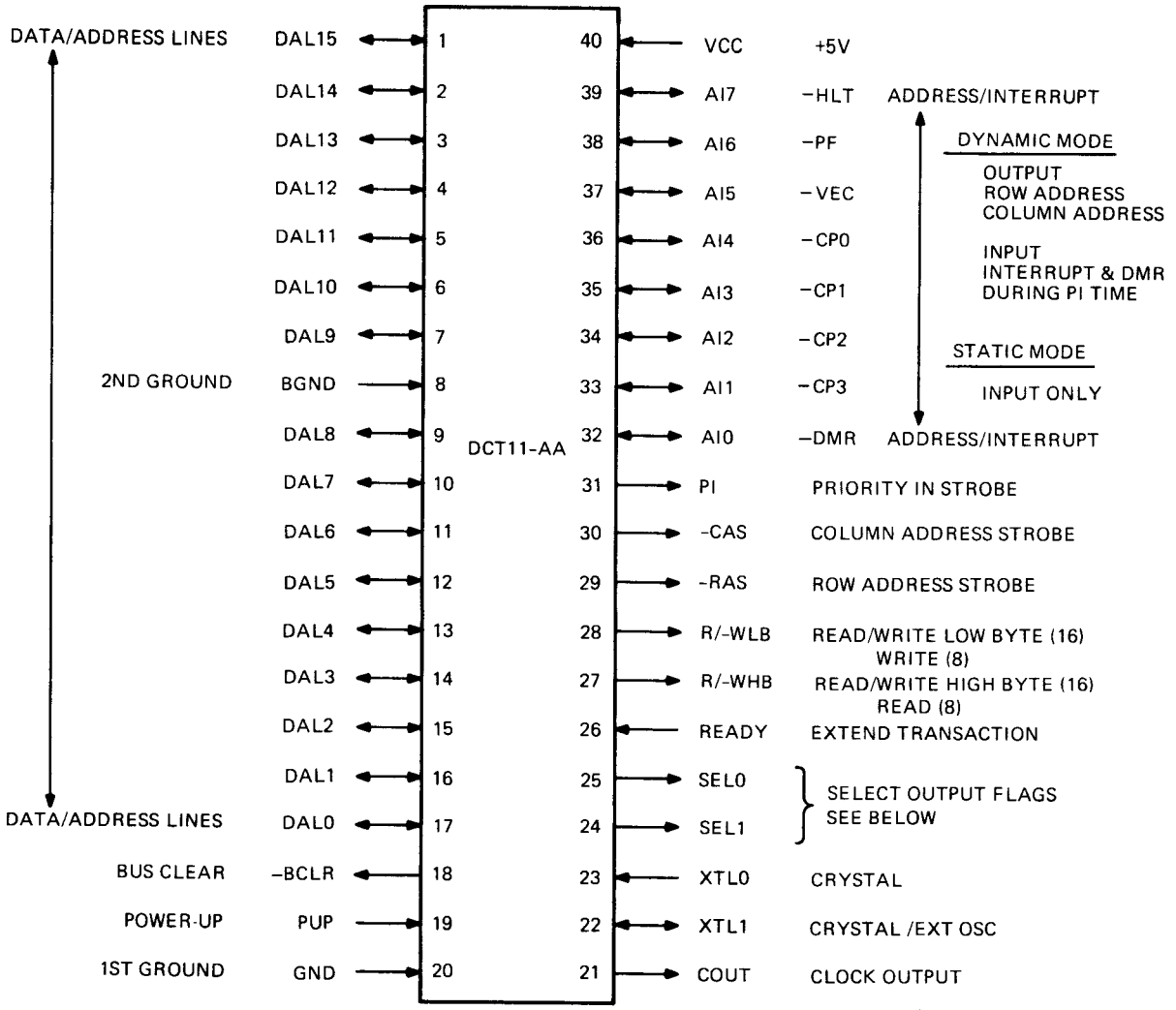


NOTE 1: XTAL1 MAY BE USED AS AN OUTPUT
NOTE 2: BUS MODE CLOCK (MODE REG <0>=0) PHASE DEPENDS ON MODE AND CODE EXECUTED AS SHOWN WITH CONDITIONAL OUTPUT. FOR LONG MICROCYCLE MODE AND BUS MODE CLOCK, THE VALID OUTPUT IS SHOWN (PHASE IS UNCONDITIONAL)
NOTE 3: EXTRA TIMING PHASE (0D) PRESENT ONLY DURING ASPI, DMA, REFRESH AND JACK IN STANDARD MICROCYCLE MODE MR<1>=1. 0D IS PRESENT IN ALL MICROCYCLES IN LONG MICROCYCLE MODE MR<1>= 0.

Figure A-17 XTAL and COUT

SYMBOL	PARAMETER	FUNCTION OF ^tCYC	MIN/MAX
^tCYC	XTL1, XTL0 Operating Period	T = 133 ns	min
^tMOC	Pulse Mode COUT (T.E.) Set Up Time to –CAS (L.E.)	(T+10) = 143 ns	min
^tMOR	Read/Write or DMA Pulse Mode COUT (L.E.) Set Up Time to –RAS (L.E.)	10 ns	min
^tMRO	Read/Write or DMA Pulse Mode COUT (T.E.) Hold Time to –RAS (L.E.)	(T –51) = 82 ns	min
^tOPW	Pulse Mode COUT Pulse Width	(T –33) = 100 ns	min
^tORD	Pulse Mode COUT Recovery Time when Phase D is Present	(3T –37) = 363 ns	min
^tORT	Pulse Mode COUT Recovery Time	(2T –37) = 230 ns	min

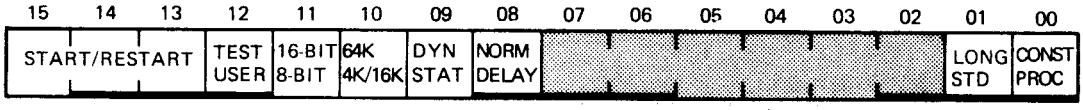
MR-5591



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



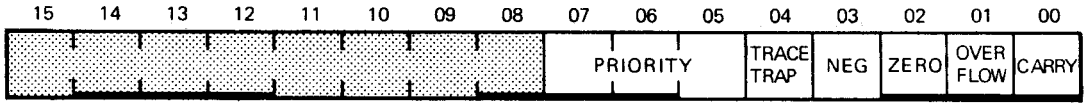
<15:13>	START/RESTART ADDRESS		
12	TESTER/USER MODE	08	NORMAL/DELAYED R/W
11	16-BIT/8-BIT BUS	<7:2>	RESERVED
10	64K/4K OR 16K MEMORY	01	LONG/STANDARD MICROCYCLE
09	DYNAMIC/STATIC MEMORY	00	CONSTANT/PROCESSOR MODE CLOCK

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

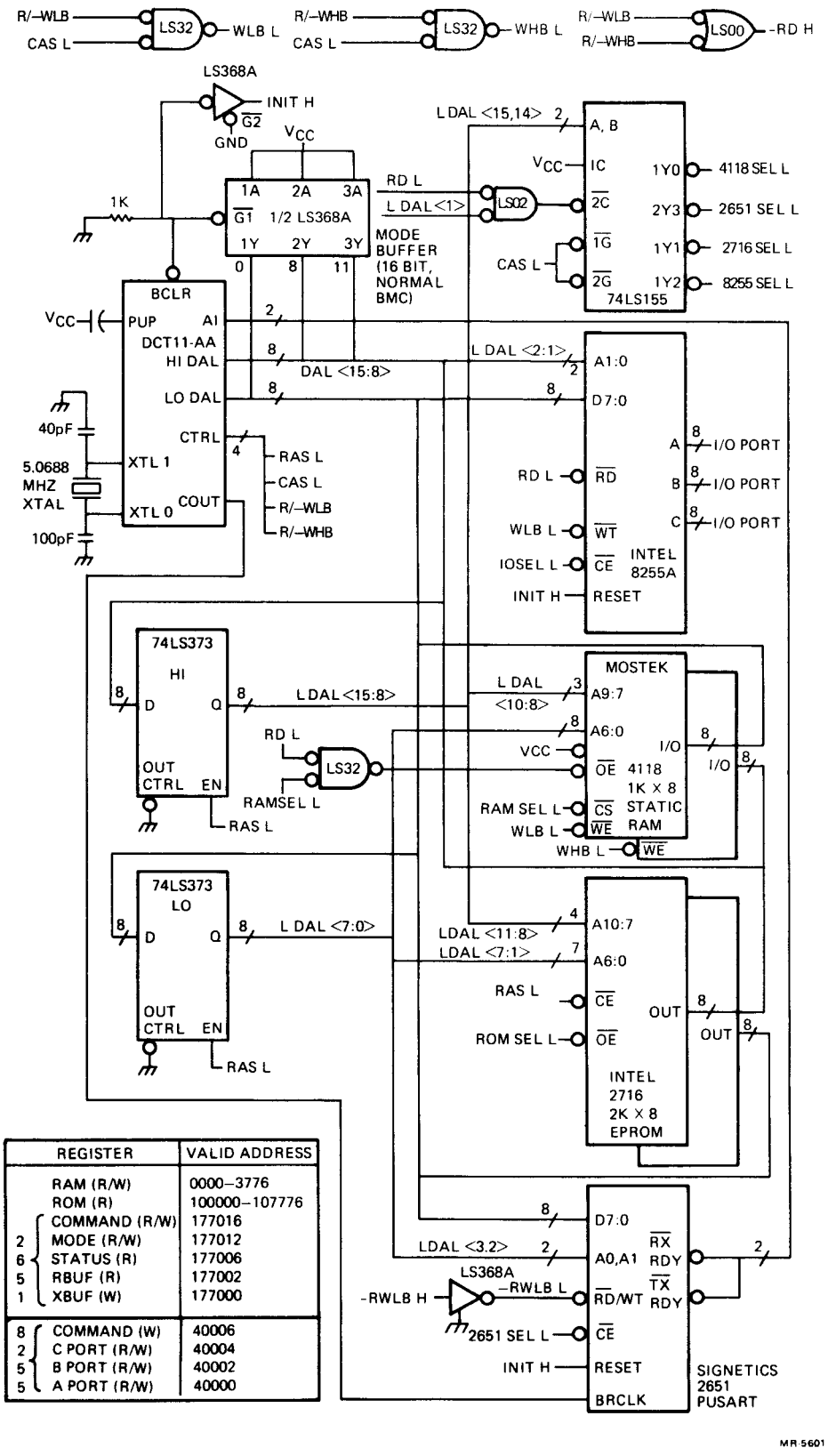
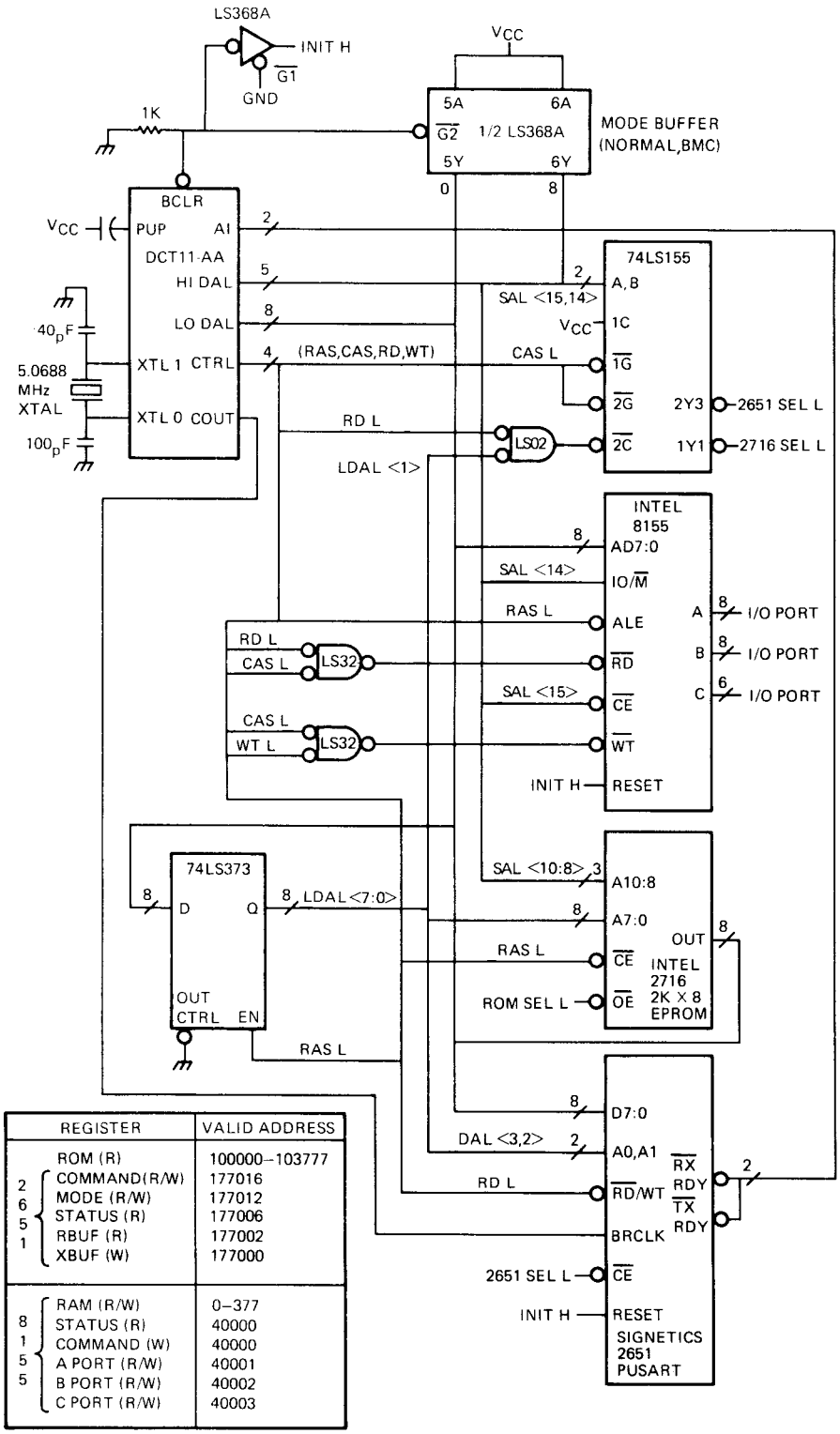


Figure A-21 16-Bit Application



REGISTER	VALID ADDRESS
ROM (R)	100000–103777
COMMAND(R/W)	177016
MODE (R/W)	177012
STATUS (R)	177006
RBUF (R)	177002
XBUF (W)	177000
RAM (R/W)	0–377
STATUS (R)	40000
COMMAND (W)	40000
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
SOFTWARE DIFFERENCES

B.1 INTRODUCTION

This appendix is meant to make the reader aware of the 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 compared with the DCT11-AA in this appendix are

PDP-11/03	PDP-11/24	PDP-11/44
PDP-11/04	PDP-11/34A	PDP-11/45
PDP-11/23	PDP-11/40	PDP-11/70

Table B-5 (found at the end of this appendix) describes the software differences and compatibilities among the DCT11-AA and other members of the PDP-11 family.

B.2 ADDRESSING MODES

Most basic instructions operate in the same way from one PDP-11 processor to another. However, there are variations in the way an 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 used in this section is found in Paragraph 6.3.

When executing a double-operand instruction, the same general-purpose register may be used for both the source and destination fields of the instruction. Note that 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_8.

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_8.

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_8 and location 1000 contains 2000_8.

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_8 and location 776 contains 2000_8.

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 operations above, the resulting 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_8. Location 1002 contains the offset value 2. R0 contains the value 2000_8.

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 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, take care 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 Paragraph B.5). This is true of all PDP-11 processors except the PDP-11/45.

The PDP-11/45 causes a trap through memory location 10 when executing this instruction.

B.3 PDP-11 INSTRUCTION SET

The DCT11-AA implements the basic PDP-11 instruction set. This instruction set offers a wide choice of operations, and often a single instruction will do 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 categories:

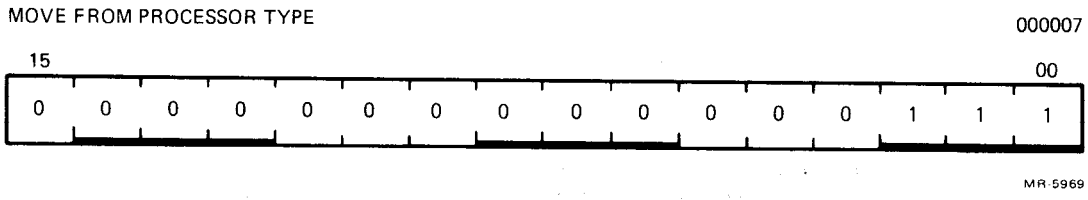
- Instructions not common to all PDP-11s
- Basic instruction execution
- Instructions not executed
- Effect of the T bit (instruction trace trap)

B.3.1 Instructions Not Common to All PDP-11s

As the number of PDP-11 processor types increased, instructions were added to the basic instruction set. 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



Operation: R0 ← processor type

Condition Codes: Not affected

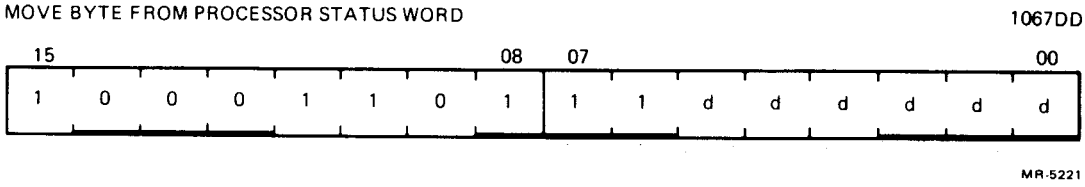
The DCT11-AA, PDP-11/23, PDP-11/24, and PDP-11/44 are the only processors that execute the MFPT instruction. The model code is placed in the low byte of register R0, indicating to the system software the processor type. Table B-1 shows the codes assigned to identify the processor in use.

NOTE
The PDP-11/23 and PDP-11/24 are controlled by the same processor and have the same model code.

Table B-1 Processor Codes

Model Code	Processor Type
4	DCT11-AA
3	PDP-11/23 or PDP-11/24
1	PDP-11/44

B.3.1.2 MFPS Instruction



Operation: (dst) ← PS

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

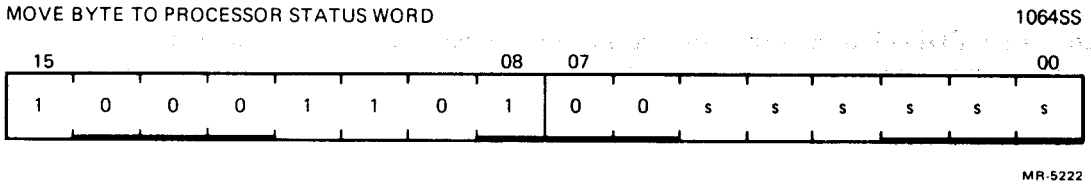
The low byte of the PS is used as the source operand. The destination operand is treated as a byte.

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 a nonexistent memory location. (Refer to Paragraph B.5.) Attempting to read or write data at address 177776 and expecting the PS will cause unpredictable results.

B.3.1.3 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.

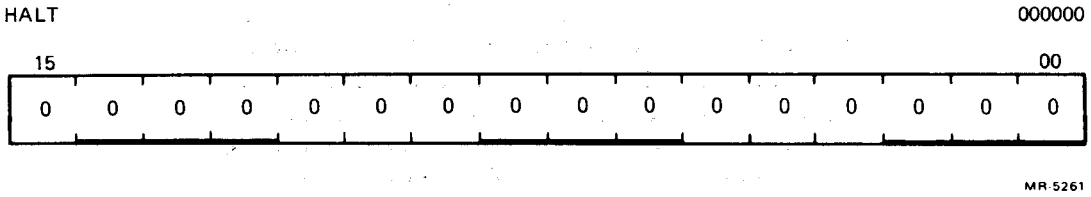
NOTE

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



Condition Codes: Not affected

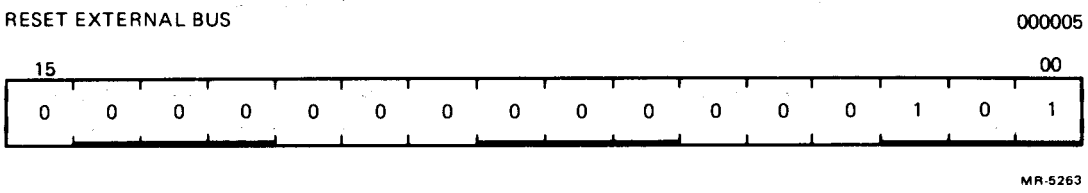
When the other PDP-11 processors covered in this appendix execute the halt instruction, their operations cease. Control goes to the console (if one is present) or to a console microprogram within the processor. The DCT11-AA has neither console nor console microprogram; it executes a halt instruction the way it would 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 Paragraph B.6.

NOTE

When developing software for the DCT11-AA on PDP-11 systems that have memory management, be aware that the trap sequence is different when executing a halt instruction. Refer to the appropriate processor handbook.

B.3.2.2 Reset Instruction



Condition Codes: Not affected

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 μs followed by a minimum 150 ns 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 with the PDP-11/03, PDP-11/23, and PDP-11/24.

A power-fail interrupt occurring during a reset instruction in the PDP-11/04 and PDP-11/34 is a fatal error, and no power-down sequence occurs. PDP-11/44, PDP-11/45, and PDP-11/70 reset instructions are 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-2. An attempt to execute these instructions causes the processor to trap through location 10.

Table B-2 PDP-11 Instructions Not Executed by the DCT11-AA

Op Code	Mnemonic	Op Code	Mnemonic
00 00 10	Reserved	07 04 SS	MUL
through		07 1R SS	DIV
00 00 77		07 2R SS	ASH
00 02 10	Reserved	07 3R SS	ASHC
through		07 50 0R	FADD
00 02 27		07 50 1R	FSUB
00 02 3N	SPL	07 50 2R	FMUL
00 64 NN	MARK	07 50 3R	FDIV
00 65 SS	MFPI	07 50 40	Unused
00 66 DD	MTPI	through	
00 70 00	Reserved	07 67 77	
through		10 65 SS	MFPD
00 77 77		10 66 DD	MTPD
		17 00 00	FPP Instructions
		through	
		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 will occur 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 in 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* the execution of the next instruction. If the RTT instruction causes the T bit to be set, the T bit trap is taken *after* the execution of 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 they acknowledge 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.

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 op code of the instruction.
- The processor then obtains the source operand.
- The destination operand is computed.
- The source operand is written into the destination address.

The MOV instruction operates similarly in the DCT11-AA and 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 it writes 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 and then writes to the output register. This may result in the RxRDY signal's being cleared. Data may be lost when RxRDY is cleared in this manner.

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 in order to execute a service routine. Software interrupts call service routines required by the program. They occur when executing trap instructions or when the trace bit is set in the processor status register. Program execution is resumed when the service routine is completed.

The DCT11-AA services calls and interrupts in the following order of priority.

1. HALT (nonmaskable interrupt or instruction)
2. Power-fail (nonmaskable interrupt)
3. Trace trap (T bit)
4. CP<3:0> priority 7 (interrupt)
5. CP<3:0> priority 6 (interrupt)
6. CP<3:0> priority 5 (interrupt)
7. CP<3:0> priority 4 (interrupt)
8. Trap instruction call

™Intel is a trademark of the Intel Corporation.

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 1 of 15 devices on 1 of 4 priority levels. Interrupts are maskable in that the priority code of the interrupting device must exceed the value in the PS (bits <7:5>); otherwise the interrupts are not acknowledged.

The DCT11-AA also has two nonmaskable 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 nonmaskable interrupt HALT is not associated with an interrupt vector. When a HALT interrupt occurs, the current PS and PC are pushed onto 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 the 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 the interrupt is serviced. The T bit must not be set in the PS word of the T bit trap vector. If it is, 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.

During an interrupt acknowledge transaction, the vector address is provided by either a fixed table stored in the DCT11-AA (internal vector address) or by the interrupting device (external vector address). Table B-3 lists the internal vectors assigned to interrupt priority codes.

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 nonexistent memory location is accessed (nonexistent memory (NXM) error).
- The stack value approaches the vector location area (stack overflow error).

Table B-3 Interrupt Priority Codes

	Vector Address	
Priority Level	**New PC at:**	**New PS at:**
Nonmaskable HALT	Restart address	340
Nonmaskable 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

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 nonexistent memory location, the transaction is completed and program execution continues. If the transaction is a read, undefined data is received. A write to a nonexistent 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_8. If it does, unpredictable results may occur 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 accessible to the programmer as memory locations. 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 by the programmer as memory locations.

B.6 POWER-UP

The DCT11-AA is a flexible microprocessor that 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 3-bit code that represents one of the eight start/restart addresses. Table B-4 lists the start/restart addresses.

Table B-4 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 used only at the time power is applied to the DCT11-AA. The 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 values of the PS and PC are 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 explanations of the various modes described in Table B-5.

%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 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 the 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 the 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) — Index-deferred addressing (register mode 7). 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-deferred addressing. Relative-deferred addressing uses the contents of the location following the op code as the address of the address of the destination operand.

Table B-5 Software Differences and Compatibilities

	Activity	DCT11	LSI-11	PDP-11/ 23	04	34	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 (or 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 (or 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		
	JMP (R)+ or JSR reg,(R)+: Initial contents of R are used as the new PC.	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			

Table B-5 Software Differences and Compatibilities (Cont)

	Activity	DCT11	LSI-11	PDP-11/ 23	04	34	05,10	15,20	35,40	45
	Register addresses 177700–177717 timeout 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 PDP-11/23.		X	X				X	X	X
	Register addresses 177700–177717 are handled as regular memory addresses (in the BSIO page). No internal registers 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
	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 PDP-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.	A power-fail during a RESET instruction is not recognized until after the instruction is finished (70 ms). A RESET instruction consists of a 70 ms pause with INIT occurring during the first 20 ms.							X	X	

Table B-5 Software Differences and Compatibilities (Cont)

	Activity	DCT11	LSI-11	PDP-11/ 23	04	34	05,10	15,20	35,40	45
	A power-fail immediately ends the RESET instruction and traps if an INIT is in progress. A minimum INIT of 1 μs occurs if instructions aborted.									X
	A power-fail acts the same as in the PDP-11/45 (22 ms with about 300 ns minimum). A power-fail during a RESET fetch is fatal with no power-down sequence.			X	X		X			
	The RESET instruction consists of 10 μs of INIT followed by a 90 μs pause. A power-fail is not recognized until the instruction is complete.		X	X						
	The RESET instruction consists of a minimum 8.4 μs followed by a minimum 150 ns pause. A 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	
	When operating with the T bit set (e.g., when 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	

Table B-5 Software Differences and Compatibilities (Cont)

	Activity	DCT11	LSI-11	PDP-11/ 23	04	34	05,10	15,20	35,40	45
	If an interrupt occurs during an instruction and the T bit is set, the interrupt is acknowledged before the T bit trap.									X
13.	A T bit trap will sequence out of a WAIT instruction.	X		X	X		X	X	X	
	A T bit trap will not sequence out of a WAIT instruction; waits until an interrupt.		X							X
14.	An explicit reference (direct access) to the 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
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 PDP-11/23.		X	X	X	X	X	X		
	Odd address/nonexistent references using the SP cause a fatal trap. Upon bus error in trap service, a new stack is 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.	Eight general-purpose registers.	X	X	X	X	X	X	X	X	
	Sixteen general-purpose registers.									X

Table B-5 Software Differences and Compatibilities (Cont)

Activity	DCT11	LSI-11	PDP-11/ 23	04	34	05,10	15,20	35,40	45
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
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 a source operand.		X							
EIS instructions do a DATI bus sequence when fetching a source operand.			X					X	X

Table B-5 Software Differences and Compatibilities (Cont)

Activity	DCT11	LSI-11	PDP-11/ 23	04	34	05,10	15,20	35,40	45
26. MOV instruction does only 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 the PC contains a nonexistent memory address and a bus error occurs, the PC will have been incremented.		X	X	X	X	X	X		X
If the PC contains a nonexistent memory address and a bus error occurs, the PC will be unchanged.								X	
Does not support bus errors.	X								
28. If a register contains a nonexistent memory address in mode 2 and a bus error occurs, the register will be incremented.		X	X			X	X	X	X
If a register contains nonexistent memory address in mode 2 and a bus error occurs, the register will be unchanged.				X	X				
Does not support bus errors.	X								
29. If a register contains an odd value in mode 2 and a bus error occurs, the register will be incremented.		X	X					X	X
If a register contains an odd value in mode 2 and a bus error occurs, the 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 does not abort on the PDP-11/35 and PDP-11/40.)								X	
Condition codes that are restored after EIS/FIS interrupt abort are indeterminate.		X							

Table B-5 Software Differences and Compatibilities (Cont)

Activity	DCT11	LSI-11	PDP-11/ 23	04	34	05,10	15,20	35,40	45
31. Op codes 075040–075377 unconditionally trap to 10 as reserved op codes.	X		X	X	X	X	X	X	X
If the KEV-11 option is present, op codes 075040–075377 perform a memory read using as a pointer the register specified by the low-order 3 bits. 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 option is present, a trap-to-10 occurs.		X							
32. Op codes 210–217 trap to 10 as reserved op codes.	X		X	X	X	X	X	X	X
Op codes 210–217 are used as a maintenance instruction.		X							
33. Op codes 75040–75777 trap to 10 as reserved op codes.	X		X	X	X	X	X	X	X
Op codes 75040–75377 can be used as escapes to user microcode only if the KEV-11 option is present. Op codes 75400–75777 can also be used as escapes to user microcode and the KEV-11 option need not be present. If no user microcode exists, a trap-to-10 occurs.		X							
34. Op codes 170000–177777 trap to 10 as reserved instructions.	X			X		X	X	X	
Op codes 170000–177777 are implemented as floating-point instructions.			X						X
Op codes 170000–177777 can be used as escapes to user microcode. If no user mocrocode exists, a trap-to-10 occurs.		X							
35. CLR and SXT do only a DATO sequence for the last bus cycle.			X						
CLR and SXT do a DATIP-DATO sequence for the last bus cycle.		X		X		X	X	X	X

Table B-5 Software Differences and Compatibilities (Cont)

Activity	DCT11	LSI-11	PDP-11/ 23	04	34	05,10	15,20	35,40	45
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						
37. PS<15:12>, user mode, user stack pointer, and the			X						X
MTPX and MFPX instructions exist even when MEM. MGT. is not configured.									
PS<15:12>, user mode, user stack pointer, and the 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 address 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 a 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-5 Software Differences and Compatibilities (Cont)

	Activity	DCT11	LSI-11	PDP-11/ 23	04	34	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 the PS and PSW to the stack, loads the PS with 340, and loads the PC with power-up address + 4 (restart address).	X								
43.	Resident ODT microcode.		X	X						
44.	All data outs (DATOs) are preceded 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								

Table B-6 Hardware Differences – Traps
(Transparent to Software)

DCT11	PDP-11/23	PDP-11/04	PDP-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 HALT interrupt TRACE trap External vector interrupt Internal vector interrupt Power-fail trap WAIT loop Test mode request	Memory parity errors Memory management Fault Bus error traps TRAP instructions TRACE trap OVFL trap Power-fail trap Console bus request QBus bus request WAIT loop	Bus error trap TRAP intructions TRACE trap OVFL trap Power-fail trap Unibus bus request Console HALT WAIT loop	Memory parity errors Memory management Fault Bus error traps TRAP instructions TRACE trap OVFL trap Power-fail trap Console bus request Unibus bus request WAIT loop
LSI-11	**PDP-11/05,10**	**PDP-11/15,20**	**PDP-11/35,40**
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

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

What features are most useful? _____

What faults or errors have you found in the manual? _____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532
Attention: Printing and Circulating Service (NR2/M15)
Customer Services Section

Order No. EK-DCT11-UG

MR