MC68450

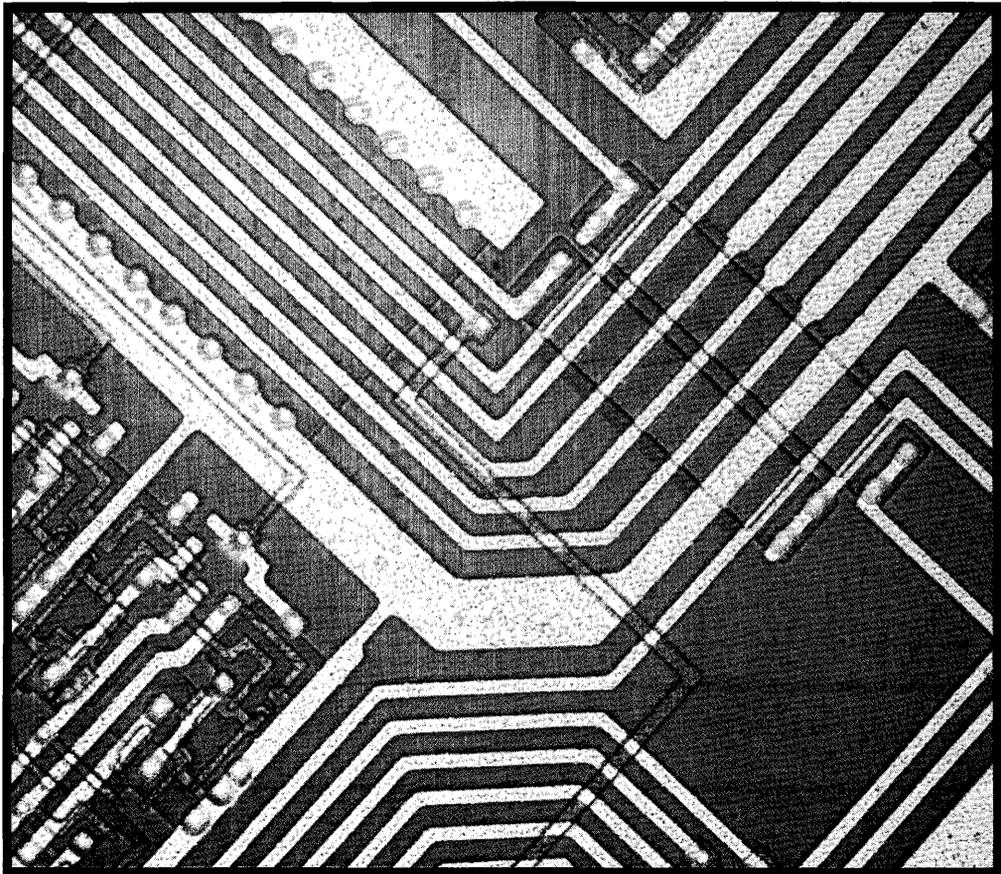*Advance Information*

Direct Memory Access Controller (DMAC)

This document contains information on a new product. Specifications and information herein are subject to change without notice.

MOTOROLA

ADI1216

TABLE OF CONTENTS

TABLE OF CONTENTS
(CONTINUED)

TABLE OF CONTENTS
(CONTINUED)

TABLE OF CONTENTS
(CONTINUED)

TABLE OF CONTENTS
(CONCLUDED)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(CONTINUED)

LIST OF TABLES

SECTION 1
INTRODUCTION

M68000 microprocessors utilize state-of-the-art MOS technology to maximize performance and throughput. The MC68450 direct memory access controller (DMAC) is designed to complement the performance and architectural capabilities of M68000 Family microprocessors by moving blocks of data in a quick, efficient manner with minimum intervention from a processor. The DMAC performs memory-to-memory, memory-to-device, and device-to-memory data transfers by utilizing the following features:

- Four Independent DMA Channels with Programmable Priority
- Asynchronous M68000 Bus Structure with a 24-Bit Address and a 16-Bit Data Bus
- Fast Transfer Rates: Up to 5 Megabytes per Second at 10 MHz, No Wait States
- Fully Supports all M68000 Bus Options such as Halt, Bus Error, and Retry
- FIFO Locked Step Support with Device Transfer Complete Signal
- Flexible Request Generation:
    Internal, Maximum Rate
    Internal, Limited Rate
    External, Cycle Steal (With or Without Hold)
    External, Burst
    Mixed Internal and External
- Programmable 8-Bit or 16-Bit Peripheral Device Types:
    Explicitly Addressed, M68000 Type
    Explicitly Addressed, M6800 Type
    Implicitly Addressed:
        Device with Request and Acknowledge
        Device with Request, Acknowledge, and Ready
- Pin and Register Compatible Functional Superset of the MC68440 DDMA

1.1 GENERAL OPERATION

The main purpose of a direct memory access (DMA) controller in any system is to transfer data at very high rates, usually much faster than a microprocessor under software control can handle. The term DMA is used to refer to the ability of a peripheral device to access memory in a system in the same manner as a microprocessor does. DMA operation can occur concurrently with other operations that the system processor needs to perform, thus greatly boosting overall system performance.

Figure 1-1 illustrates a typical system configuration using a DMA interface to a high speed disk storage device. In a system such as this, the DMAC will move blocks of data between the peripheral controllers and memory at rates approaching the limits of the memory bus, since the simple function of data movement is implemented in high speed MOS hardware. A block of data consists of a sequence of byte, word, or long-word operands starting at a specific address in memory with the length of the block determined by a transfer count. A single channel operation may involve the transfer of several blocks of data between the memory and a device.
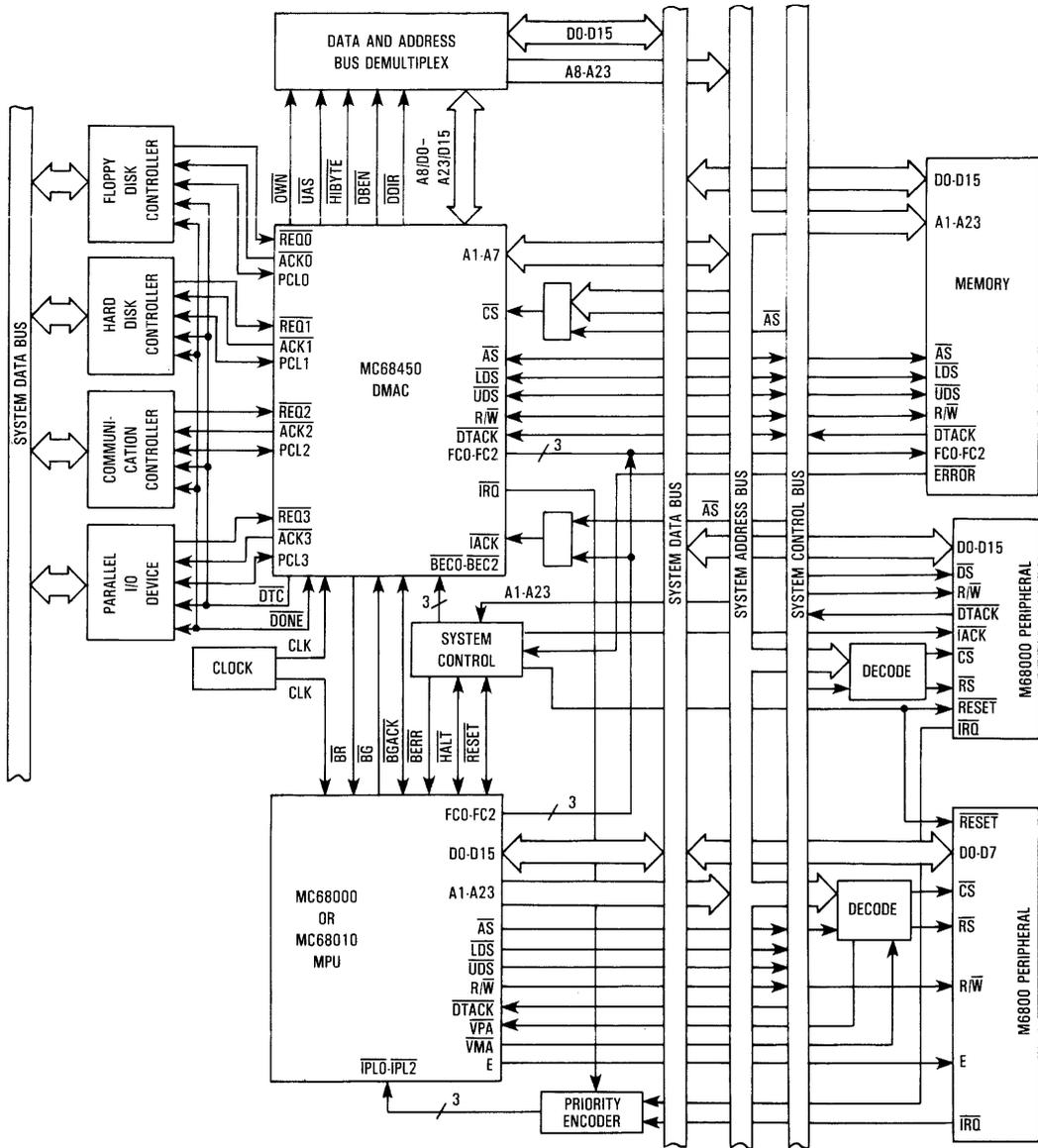
Figure 1-1. Typical M68000-Based System Configuration

Any operation involving the DMAC will follow the same basic steps: channel initialization by the system processor, data transfer, and block termination. In the initialization phase, the host processor loads the registers of the DMAC with control information, address pointers, and transfer counts and then starts the channel. During the transfer phase, the DMA accepts requests for operand transfers and provides addressing and bus control for the transfers. The termination phase occurs after the operation is complete, when the DMAC indicates the status of the operation in the status register. During all phases of a data transfer operation, the DMAC will be in one of three operating modes:

IDLE   This is the state that the DMAC assumes when it is reset by an external device and waiting for initialization by the system processor or an operand transfer request from a peripheral.

MPU   This is the state that the DMAC enters when it is chip selected by another bus master in the system (usually the main system processor). In this mode, the DMAC internal registers are written or read, to control channel operation or check the status of a block transfer.

DMA   This is the state that the DMAC enters when it is acting as a bus master to perform an operand transfer.

In addition to fully supporting the M68000 Family bus, the DMAC also supports transfers to or from M6800 Family peripheral devices. Figure 1-2 illustrates a typical system configuration utilizing an M6800 type peripheral devices.
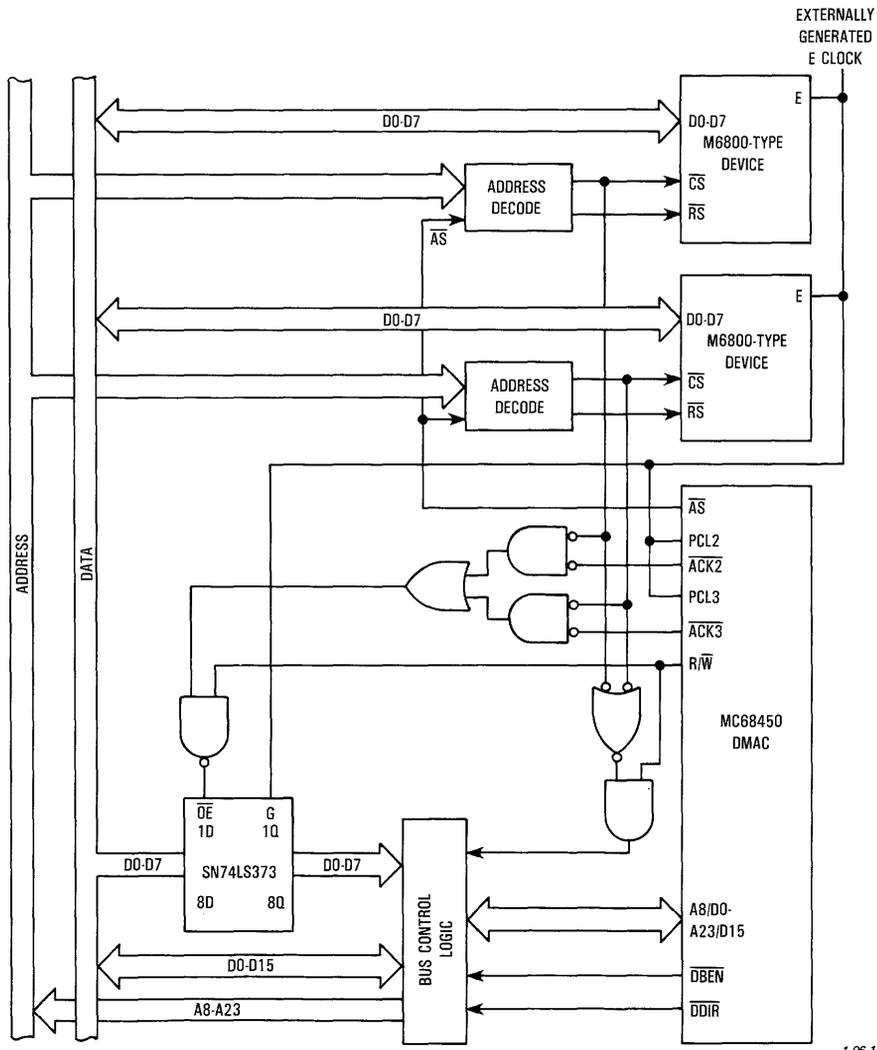
Figure 1-2. Typical System Configuration with M6800-Type Peripheral Devices

1.2 OPERAND TRANSFER MODES

The DMAC can perform implicit address or explicit address data transfers using any of the following protocols:
1) explicitly addressed, MC68000 compatible device,
2) explicitly addressed, MC6800 compatible device,
3) implicitly addressed, device with acknowledge, and
4) implicitly addressed, device with acknowledge and ready.

In the first two protocols, data is transferred from the source to an internal DMAC holding register, and then on the next bus cycle moved from the holding register to the destination. Protocols 3 and 4 require only one bus cycle for data transfer, since only one device needs to be addressed. With these protocols, communication is performed using a two-signal and three-signal handshake, respectively.

Implicitly addressed devices do not require the generation of a device data register address for a data transfer. Such a device is controlled by a five signal device control interface on the DMAC during implicit address transfers as shown in Figure 1-3. Since only memory is addressed during such a data transfer, this method is called the single-address method.

Explicitly addressed devices require that a data register within the peripheral device be addressed. No signals other than the M68000 asynchronous bus control signals are needed to interface with such a device, although any of the five device control signals may also be used. Because the address bus is used to access the peripheral, the data cannot be directly transferred to/from memory since memory also requires addressing. Therefore, data is transferred from the source to an internal holding register in the DMAC and then transferred to the destination during a second bus transfer as shown in Figure 1-4. Since both memory and the device are addressed during such a data transfer, this method is called the dual-address method.
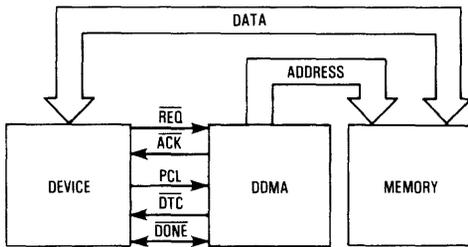
1-623

Figure 1-3. Implicitly Addressed Device Interface
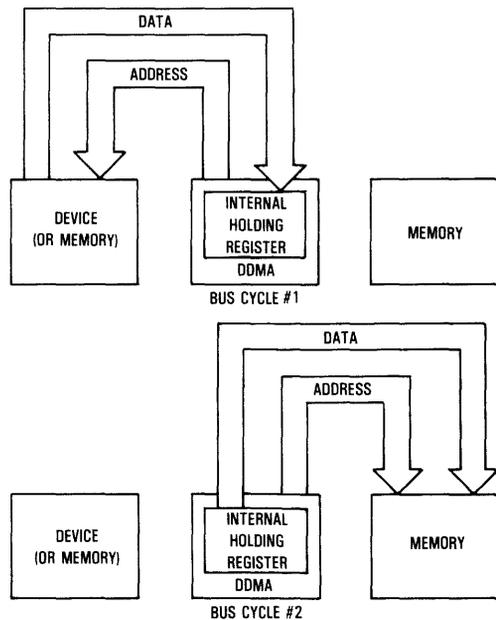
1-624

Figure 1-4. Dual-Address Transfer Sequence

1.3 CHANNEL OPERATING MODES

There are three types of channel operations: 1) single block transfers, 2) continued operation, and 3) chained operations. The first two modes utilize on-chip registers while the last mode uses an on-chip address register to point to address and count parameters stored in system memory.

When transferring single blocks of data, the memory address and device address registers are initialized by the user to specify the source and destination of the transfer. Also initialized is the memory transfer count register to count the number of operands transferred in a block. Repeated transfers are possible with the continue mode of operation, where the memory address and transfer count registers are automatically loaded from internal registers upon completion of a block transfer. See Figure 1-5.
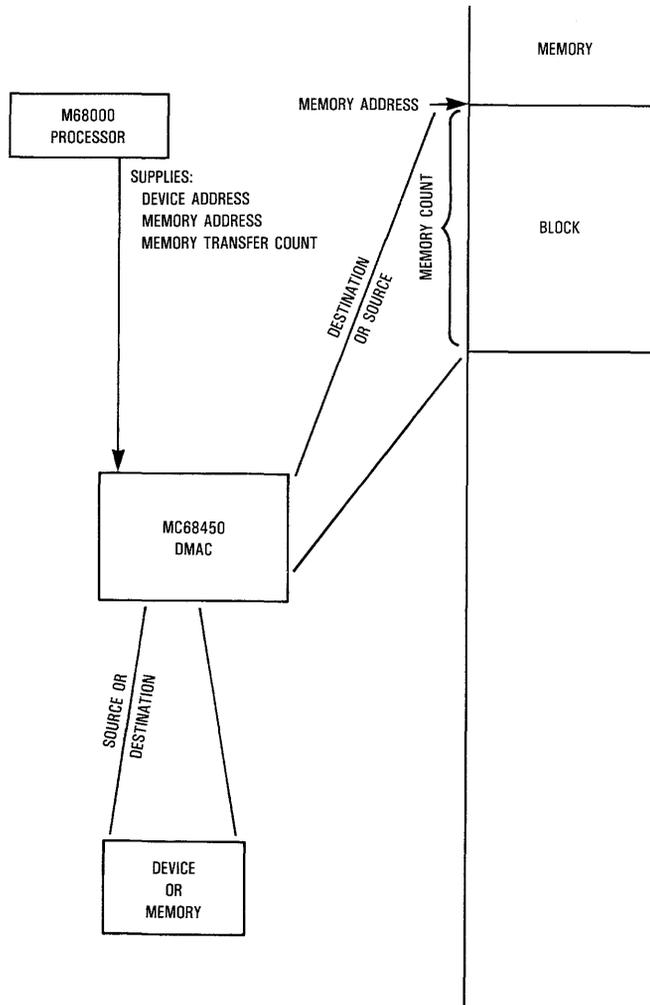
1-142-1

Figure 1-5. Single Block Transfer

The two chaining modes are array chaining and linked array chaining. The array chaining mode operates from a contiguous array in memory consisting of memory addresses and transfer counts. The base address register and base transfer count register are initialized to point to the beginning address of the array and the number of array entries, respectively. As each block transfer is completed, the next entry is fetched from the array, the base transfer count is decremented, and the base address is incremented to point to the next array entry. When the base transfer count reaches zero, the entry just fetched is the last block transfer defined in the array. See Figure 1-6.
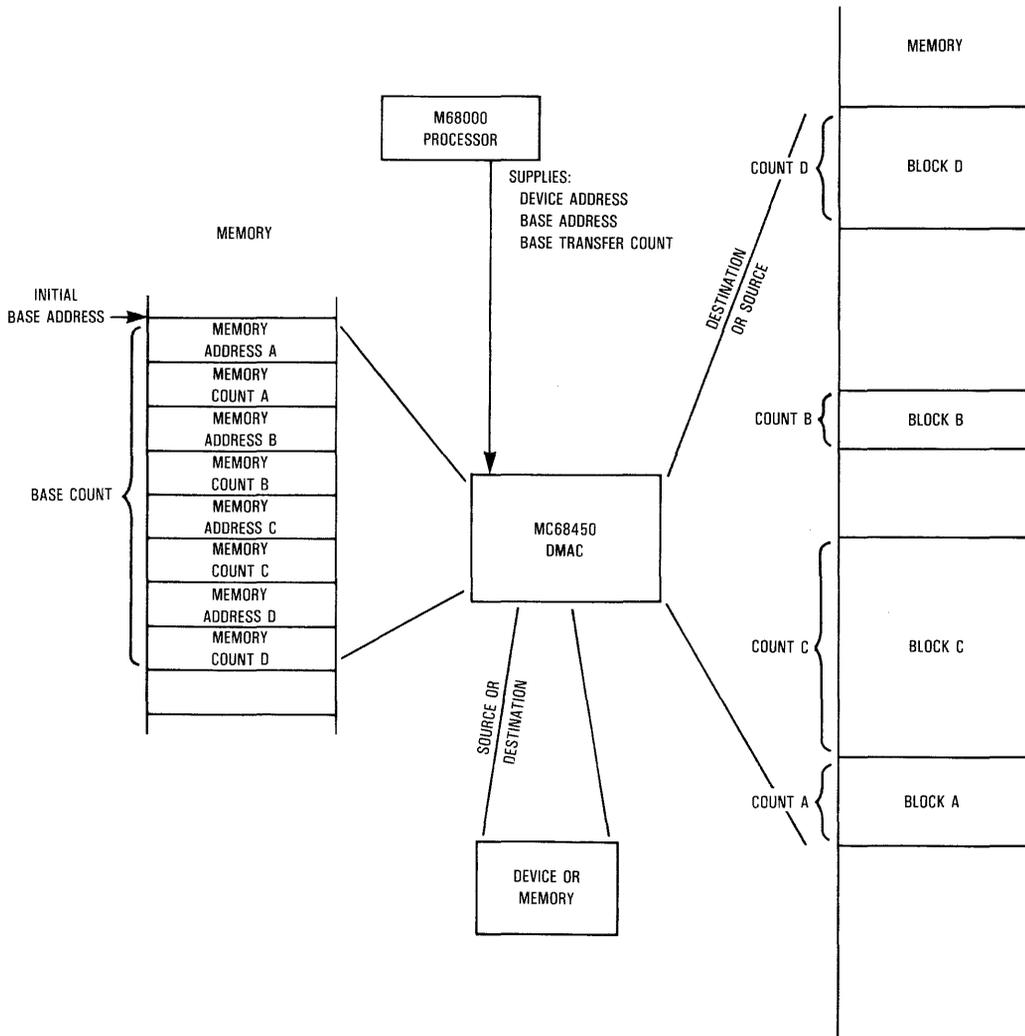
1-143-1

Figure 1-6. Array Chain Transfer

The linked array chaining mode is similar to the array chaining mode, except that each entry in the memory array also contains a link address which points to the next entry in the array. This allows a non-contiguous memory array. The last entry contains a link address set to zero. No base transfer count register is needed in this mode. The base address register is initialized to the address of the first entry in the array. The link address is used to update the base address register at the beginning of each block transfer. This chaining mode allows array entries to be easily moved or inserted without having to reorganize the array into sequential order. Also, the number of entries in the array need not be specified to the DMAC. See Figure 1-7.
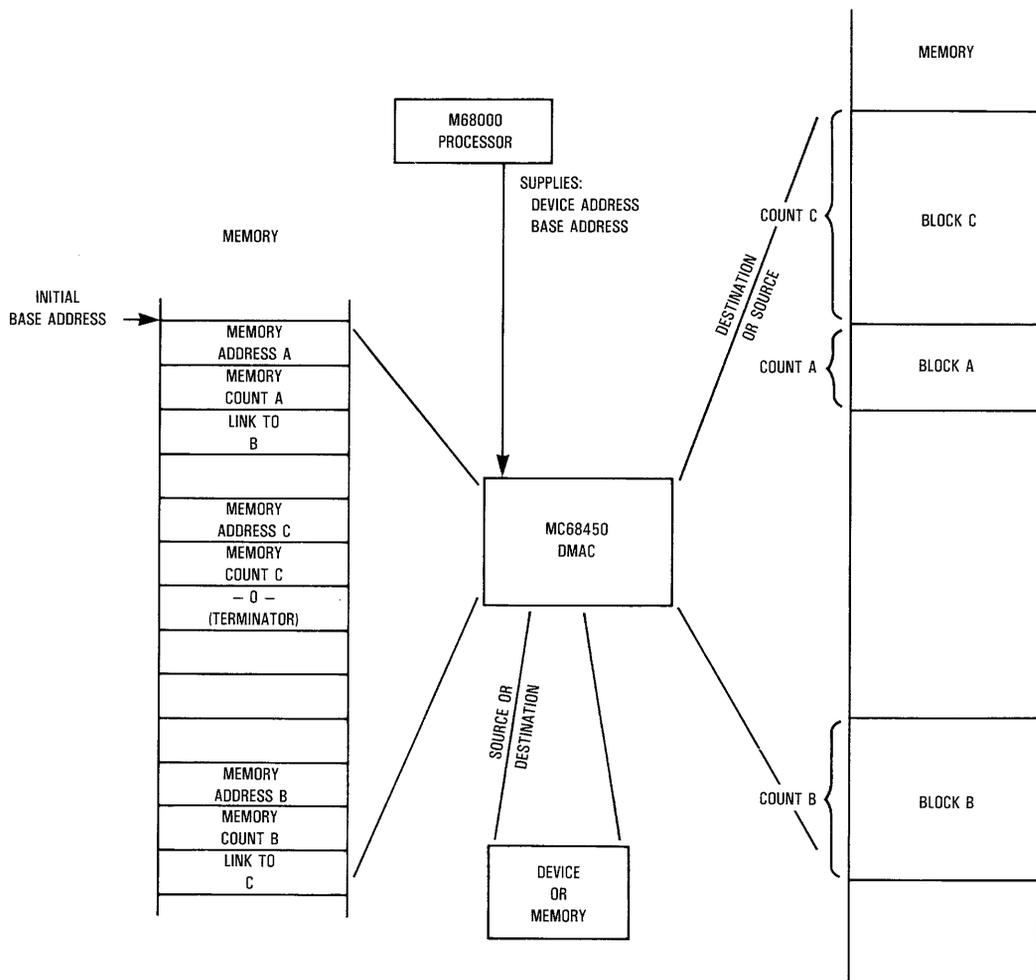
Figure 1-7. Linked Array Chain Transfer

1.4 INTERRUPT OPERATION

The DMAC will interrupt the MPU for a number of event occurrences such as the completion of a DMA operation, or at the request of a peripheral device using a PCL line. The user must write interrupt vectors into an on-chip vector register, for use in the M68000 vectored interrupt structure. Two vector registers are available for each channel.

1.5 CHANNEL PRIORITY

Each channel may be given a priority level of 0, 1, 2, or 3. If several channel requests occur at the same priority level, a round-robin mode is entered automatically.

1.6 REQUEST MODES

Requests may be externally generated by a device or internally generated by the auto-request mechanism of the DMAC. Auto-requests may be generated either at the maximum rate, where the channel always has a request pending, or at a limited rate determined by selecting a portion of the bus bandwidth to be available for DMA activity. External requests can be either burst requests or cycle steal requests that are generated by the request signal associated with each channel.

1.7 REGISTERS

The DMAC contains 17 registers for each of the four channels plus one general control register, all of which are under complete software control. The user programmer's model of the registers is shown in Figure 1-8.

The DMAC registers contain information about the data transfers such as the source and destination address and function codes, transfer count, operand size, device port size, channel priority, continuation address and transfer count, and the function of the peripheral control line. One register also provides status and error information on channel activity, peripheral inputs, and various events which may have occurred during a DMA transfer. A general control register selects the bus utilization factor to be used in limited rate auto-request DMA operations.

*Word Aligned Register Pairs
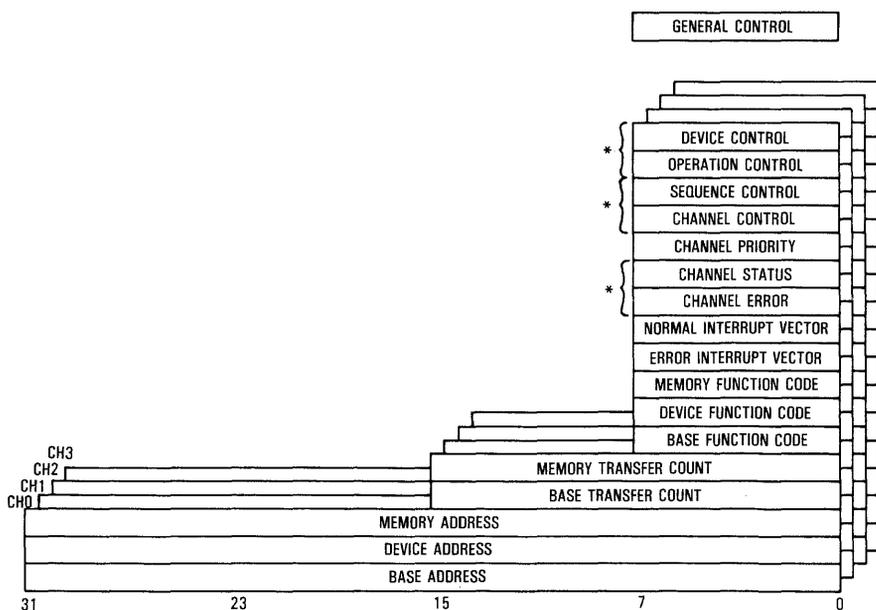
Figure 1-8. DMAC Programmer's Model

SECTION 2
SIGNAL DESCRIPTION

This section contains a brief description of the DMAC input and output signals. Included at the end of the functional description of the signals is a table describing the electrical characteristics of each pin (i.e., the type of driver used).

NOTE
The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true, independent of whether that level is represented by a high or low voltage. The term negate or negation is used to indicate that a signal is inactive or false.

2.1 SIGNAL ORGANIZATION

The input and output signals can be functionally organized into the groups shown in Figure 2-1. The function of each signal or group of signals is discussed in the following paragraphs.
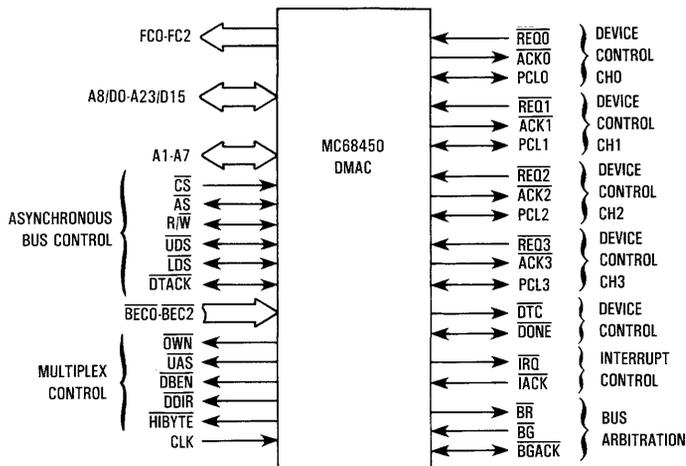
1-146-1

Figure 2-1. Signal Organization

2.1.1 Address/Data Bus (A8/D0 through A23/D15)

This 16-bit bus is time multiplexed to provide address outputs during the DMA mode of operation and is used as a bidirectional data bus to input data from an external device (during an MPU write or DMA read)

or to output data to an external device (during an MPU read or a DMA write). This is a three-state bus and is demultiplexed using external latches and buffers controlled by the multiplex control lines (refer to **2.1.5 Multiplex Control**).

2.1.2 Lower Address Bus (A1 through A7)

These bidirectional three-state lines are used to address the DMAC internal registers in the MPU mode and to provide the lower seven address outputs in the DMA mode.

2.1.3 Function Codes (FC0 through FC2)

These three-state output lines are used in the DMA mode to further qualify the value on the address bus to provide eight separate address space that may be defined by the user. The value placed on these lines is taken from one of the internal function code registers, depending on the register that provides the address used during a DMA bus cycle.

2.1.4 Asynchronous Bus Control

Asynchronous data transfers are handled using the following control signals: chip select, address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are described in the following paragraphs.

2.1.4.1 CHIP SELECT ($\overline{CS}$). This input signal is used to select the DMAC for an MPU bus cycle. When $\overline{CS}$ is asserted, the address on A1-A7 and the data strobes (or A0 when using an 8-bit bus) select the internal DMAC register that will be involved in the transfer. $\overline{CS}$ should be generated by qualifying an address decode signal with the address and data strobes.

2.1.4.2 ADDRESS STROBE ($\overline{AS}$). This bidirectional signal is used as an output in the DMA mode to indicate that a valid address is present on the address bus. In the MPU or IDLE modes, it is used as an input to determine when the DMAC can take control of the bus (if the DMAC has requested and been granted use of the bus).

2.1.4.3 READ/WRITE (R/$\overline{W}$). This bidirectional signal is used to indicate the direction of a data transfer during a bus cycle. In the MPU mode, a high level indicates that a transfer is from the DMAC to the data bus and a low level indicates a transfer from the data bus to the DMAC. In the DMA mode, a high level indicates a transfer from the addressed memory or device to the data bus, and a low level indicates a transfer from the data bus to the addressed memory or device.

2.1.4.4 UPPER AND LOWER DATA STROBE ($\overline{UDS}$ AND $\overline{LDS}$). These bidirectional lines indicate when data is valid on the bus and what portions of the bus should be involved in a transfer.

During any bus cycle, $\overline{UDS}$ is asserted if data is to be transferred over data lines D8-D15 and $\overline{LDS}$ is asserted if data is to be transferred over data lines D0-D7. $\overline{UDS}/\overline{LDS}$ are asserted by the DMAC when operating in the DMA mode and by another bus master when in the MPU mode.

2.1.4.5 DATA TRANSFER ACKNOWLEDGE ($\overline{DTACK}$). This bidirectional line is used to signal that an asynchronous bus cycle may be terminated. In the MPU mode, this output indicates that the DMAC has accepted data from the MPU or placed data on the bus for the MPU. In the DMA mode, this input is monitored by the DMAC to determine when to terminate a bus cycle. As long as $\overline{DTACK}$ remains negated,

the DMAC will insert wait cycles into a bus cycle and when $\overline{\text{DTACK}}$ is asserted, the bus cycle will be terminated (except when PCL is used as a ready signal, in which case both signals must be asserted before the cycle is terminated).

2.1.4.6 BUS EXCEPTION CONTROL ($\overline{\text{BEC0}}$ THROUGH $\overline{\text{BEC2}}$). These input lines provide an encoded signal that indicates an abnormal bus condition such as a bus error or reset. Refer to **4.4.2 Bus Exception Control Functions** for a detailed description of the function of these pins.

2.1.5 Multiplex Control

These signals are used to control external multiplex/demultiplex devices to separate the address and data information on the A8/D0-A23/D15 lines and to transfer data between the upper and lower halves of the data bus during certain DMA bus cycles.

Figure 2-2 shows the five external devices needed to demultiplex the address/data pins and the interconnection of the multiplex control signals. The SN74LS245 that may connect the upper and lower halves of the data bus is needed only if an 8-bit device is used during single address transfers.
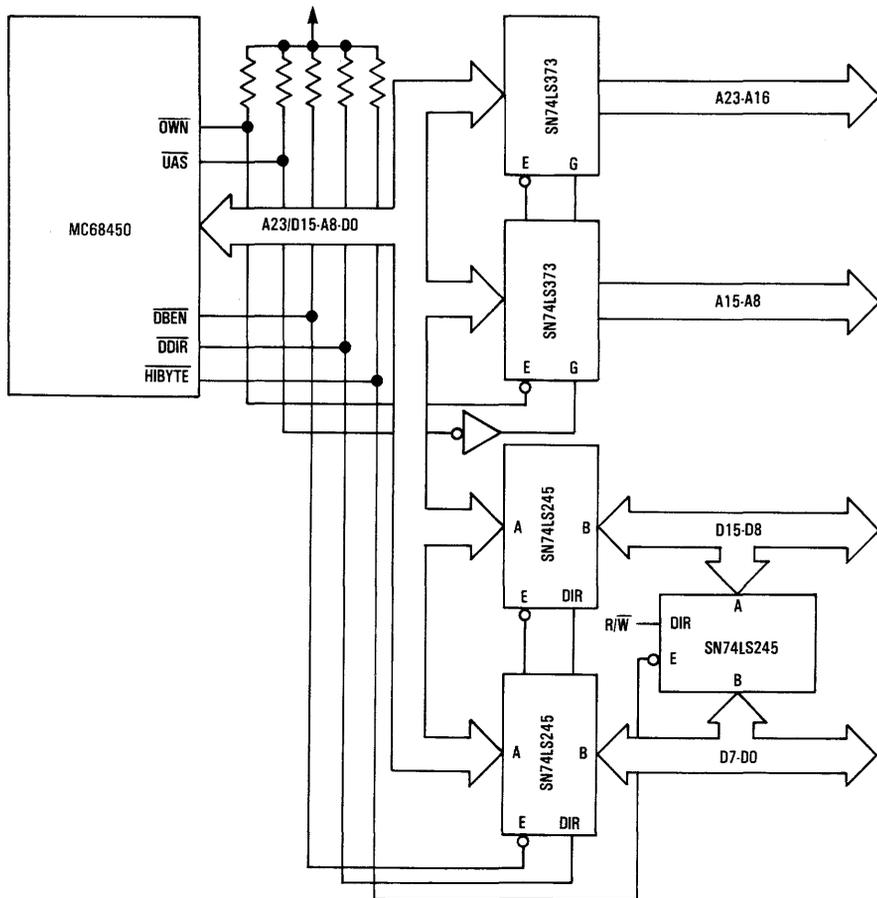
1-147-1

Figure 2-2. Demultiplex Logic

2.1.5.1 OWN ($\overline{\text{OWN}}$). This three-state output indicates that the DMAC is controlling the bus. It is used as the enable signal to turn on the external address drivers and control signal buffers.

2.1.5.2 UPPER ADDRESS STROBE ($\overline{\text{UAS}}$). This three-state output is used as the gate signal to the transparent latches that capture the value of A8-A23 on the multiplexed address/data bus.

2.1.5.3 DATA BUFFER ENABLE ($\overline{\text{DBEN}}$). This three-state output is used as the enable signal to the external bidirectional data buffers.

2.1.5.4 DATA DIRECTION ($\overline{\text{DDIR}}$). This three-state output controls the direction of the external bidirectional data buffers. If $\overline{\text{DDIR}}$ is high, the data transfer is from the DMAC to the data bus. If $\overline{\text{DDIR}}$ is low, the data transfer is from the data bus to the DMAC.

2.1.5.5 HIGH BYTE ($\overline{\text{HIBYTE}}$). This three-state output indicates that data will be present on data lines D8-D15 that must be transferred to data lines D0-D7, or vice versa, through an external buffer during a single address transfer between an 8-bit device and memory (refer to **4.3.2 Single Address Transfers**).

2.1.6 Bus Arbitration Control

These three signals form a bus arbitration circuit used to determine which device in a system will be the current bus master.

2.1.6.1 BUS REQUEST ($\overline{\text{BR}}$). This output is asserted by the DMAC to request control of the bus.

2.1.6.2 BUS GRANT ($\overline{\text{BG}}$). This input is asserted by an external bus arbiter to inform the DMAC that it may assume bus mastership as soon as the current bus cycle is completed. The DMAC will not take control of the bus until $\overline{\text{CS}}$, $\overline{\text{IACK}}$, $\overline{\text{AS}}$, and $\overline{\text{BGACK}}$ are all negated.

2.1.6.3 BUS GRANT ACKNOWLEDGE ($\overline{\text{BGACK}}$). This bidirectional signal is asserted by the DMAC to indicate that it is the current bus master. $\overline{\text{BGACK}}$ is monitored as an input to determine when the DMAC can become bus master and if a bus master other than the system MPU is a master during limited rate auto-request operation.

2.1.7 Interrupt Control

These two signals form an interrupt request/acknowledge handshake circuit with an MPU.

2.1.7.1 INTERRUPT REQUEST ($\overline{\text{IRQ}}$). This output is asserted by the DMAC to request service from the MPU.

2.1.7.2 INTERRUPT ACKNOWLEDGE ($\overline{\text{IACK}}$). This input is asserted by the MPU to acknowledge that it has received an interrupt from the DMAC. In response to the assertion of $\overline{\text{IACK}}$, the DMAC will place a vector on D0-D7 that will be used by the MPU to fetch the address of the proper DMAC interrupt handler routine.

2.1.8 Device Control

These eight lines perform the interface between the DMAC and four peripheral devices. Four sets of three lines are dedicated to a single DMAC channel and its associated peripheral; the remaining two lines are global signals shared by all channels.

2.1.8.1 REQUEST ($\overline{\text{REQ0}}$ THROUGH $\overline{\text{REQ3}}$). These inputs are asserted by a peripheral device to request an operand transfer between that peripheral device and memory. In the cycle steal request generation mode, these inputs are edge sensitive; in burst mode, they are level sensitive.

2.1.8.2 ACKNOWLEDGE ($\overline{\text{ACK0}}$ THROUGH $\overline{\text{ACK3}}$). These outputs are asserted by the DMAC to signal to a peripheral that an operand is being transferred in response to a previous transfer request.

2.1.8.3 PERIPHERAL CONTROL LINE (PCL0 THROUGH PCL3). These bidirectional lines are multi-purpose signals that may be programmed to function as ready, abort, reload, status, interrupt, or enable clock inputs or as start pulse outputs.

2.1.8.4 DATA TRANSFER COMPLETE ($\overline{\text{DTC}}$). This output is asserted by the DMAC during any DMAC bus cycle to indicate that data has been successfully transferred (i.e., the bus cycle was not terminated abnormally).

2.1.8.5 DONE ($\overline{\text{DONE}}$). This bidirectional signal is asserted by the DMAC or a peripheral device during DMA bus cycle to indicate that the data being transferred is the last item in a block. The DMAC will assert this signal during a bus cycle when the memory transfer count register is decremented to zero.

2.1.9 Clock (CLK)

The clock input is a TTL-compatible signal that is internally buffered for development of the internal clocks needed by the DMAC. The clock input should not be gated off at any time and the clock signal must conform to minimum and maximum pulse width times.

2.2 SIGNAL SUMMARY

Table 2-1 is a summary of all the signals discussed in the previous paragraphs.

Table 2-1. Signal Summary

Signal Name	Direction	Active State	Driver Type	Synchronizing Clock Edge
A8/D0-A23/D15	In/Out	High	Three State	Falling**
A1-A7	In/Out	High	Three State	Falling**
FC0-FC2	Out	High	Three State	
$\overline{CS}$	In	Low		Falling
$\overline{AS}$	In/Out	Low	Three State*	Falling
R/$\overline{W}$	In/Out	High/Low	Three State*	Falling
$\overline{UDS}$	In/Out	Low/High	Three State*	Falling
$\overline{LDS}$	In/Out	Low	Three State*	Falling
$\overline{DTACK}$	In/Out	Low	Open Drain*	Rising
$\overline{OWN}$	Out	Low	Open Drain*	
$\overline{UAS}$	Out	Low	Three State*	
$\overline{DBEN}$	Out	Low	Three State*	
$\overline{DDIR}$	Out	High/Low	Three State*	
$\overline{HIBYTE}$	Out	Low	Three State*	
$\overline{BEC0}$-$\overline{BEC2}$	In	Low		Rising
$\overline{BR}$	Out	Low	Open Drain	
$\overline{BG}$	In	Low		Rising
$\overline{BGACK}$	In/Out	Low	Open Drain*	
$\overline{IRQ}$	Out	Low	Open Drain	
$\overline{IACK}$	In	Low		Falling
$\overline{REQ0}$-$\overline{REQ3}$	In	Low		Rising
$\overline{ACK0}$-$\overline{ACK3}$	Out	Low	Always Driven	
PCL0-PCL3	In/Out	Programmed	Three State	Rising
$\overline{DTC}$	Out	Low	Three State*	
$\overline{DONE}$	In/Out	Low	Open Drain	Rising
CLK	In			

*These signals require a pullup resistor to maintain a high voltage when in the high-impedance or negated state. When these signals go to the high-impedance or negated state, they will first drive the pin high momentarily to reduce the signal rise time.

**These signals are latched on a clock edge, but are not synchronized (i.e., the latched value is used immediately, rather than delayed by one clock).

1-148

SECTION 3
REGISTER DESCRIPTION

This section contains a description of the DMAC internal registers and the control bit assignments within each register. Figure 3-1 shows the memory mapped locations of the registers for each channel. Figure 3-2 (found on a foldout page at the end of this document) shows the register summary and may be used for a quick reference to the bit definitions within each register.

The register memory map for the MC68450 DMAC is identical to the register memory map for the MC68440 Dual Channel DMA Controller (DDMA), including the individual bit assignments within the registers. However, not all functional options available on the DMAC are available on the DDMA and vice versa. If any programmable options labeled "MC68440 Reserved" or "Undefined, Reserved" are programmed into a DMAC channel, a configuration error will occur when the MPU attempts to start that channel.

The description given in this section is for a single channel, but applies to all channels since they are identical. When the operation of a register affects the four channels differently or is common to all channels, that fact will be included in the register description.

All registers within the DMAC are always accessible as bytes or words by the MPU (assuming that the MPU can gain control of the DMA bus); however, some registers may not or should not be modified while a channel is actively transferring data. If a register may not be modified during operation and an attempt is made to write to it, an operation timing error will be signaled and the channel operation aborted. Refer to **5.3.1 Programming Sequence** for the proper order to use when writing to channel registers in order to start an operation.

3.1 ADDRESS REGISTERS (MAR, DAR, AND BAR)

These three 32-bit registers contain addresses that are used by the DMAC to access memory while it is a bus master. Since the DMAC only has 24 address lines available externally, the upper eight bits of these registers are truncated; however, the upper byte is supported for reads and writes for compatibility with future expanded address capabilities. The upper eight bits will also be incremented if necessary; for example, an address register will count from $00FFFFFF to $01000000. If such an increment occurs, the external address bus will "roll over" to $000000 and the DMAC will continue normal operation without any error indication.

The memory address register (MAR) contents are used whenever the DMAC is running a bus cycle to fetch or store an operand part in memory. Likewise, the device address register (DAR) contents are used when the DMAC is running a bus cycle to transfer an operand part to or from a peripheral device during a dual address transfer (refer to **4.3 DMA MODE OPERATION**). The base address register (BAR) contents are used when the DMAC is performing table accesses in the chained modes of operation. The BAR is used by the DMAC in the continue mode of operation to hold the address to be transferred to the MAR in response to a continue operation.

CHANNEL BASE   REGISTER OFFSET

CH0 —00
CH1 —40
CH2 —80
CH3 —C0

Offset (left)	15 ... 8	7 ... 0	Offset (right)
00	CSR	CER	01
02	//// Null ////	//// Null ////	03
04	DCR	OCR	05
06	SCR	CCR	07
08	//// Null ////	//// Null ////	09
0A	MTCR		0B
0C	MSB		0D
0E	— MAR — LSB		0F
10	//// Null ////	//// Null ////	11
12	//// Null ////	//// Null ////	13
14	MSB		15
16	— DAR — LSB		17
18	//// Null ////	//// Null ////	19
1A	BTCR		1B
1C	MSB		1D
1E	— BAR — LSB		1F
20			21
22			23
24		NIVR	25
26		EIVR	27
28		MFCR	29
2A			2B
2C		CPR	2D
2E			2F
30		DFCR	31
32			33
34			35
36			37
38		BFCR	39
3A			3B
3C			3D
3E		GCR*	3F

//// Null Bit Position

*The GCR is located at FF only.

Figure 3-1. Register Memory Map   *1-149-1*

3.2 FUNCTION CODE REGISTERS (MFCR, DFCR, AND BFCR)

These 4-bit registers contain function code values that are used by the DMAC in conjunction with the three address registers during a DMA bus cycle. The address space value on the function code lines may be used by an external memory management unit (MMU) or other memory protection device to translate the DMAC logical addresses into the proper physical addresses. The value programmed into the memory function code register (MFCR), the device function code register (DFCR), or the base function code register

(BFCR) is placed on pins FC2-FC0 during a bus cycle to further qualify the address bus value, which will be the contents of the MAR, DAR, or BAR respectively. The BFCR is used by the DMAC in the continue mode of operation to hold the function code to be transferred to the MFCR in response to a continue operation request. Note that these are 4-bit registers, with bit 3 readable and writeable for future compatibility; however, it is not available as an external signal. Bits 4-7 of the function code registers will always read as zeros and are not affected by writes.

3.3 TRANSFER COUNT REGISTERS (MTCR AND BTCR)

These 16-bit registers are programmed with the operand count for a block transfer. The memory transfer count register (MTCR) is decremented each time the DMAC successfully transfers an operand between memory and a device until it is zero, at which time the channel will terminate the operation. The base transfer count register (BTCR) is used in the sequential array chained mode to count the number of block transfers to perform during a channel operation. The BTCR is used in the continue mode of operation to hold the operand count to be transferred to the MTCR in response to a continue operation request.

3.4 INTERRUPT VECTOR REGISTERS (NIVR AND EIVR)

These two 8-bit registers are used by the DMAC during interrupt acknowledge bus cycles. When the $\overline{\text{IACK}}$ signal is asserted to the DMAC, the contents of the normal interrupt vector register (NIVR) or the error interrupt vector register (EIVR) for the highest priority channel with an interrupt pending will be placed on A8/D0-A15/D7. If the ERR bit in the corresponding channel status register is clear when $\overline{\text{IACK}}$ is asserted, the contents of the NIVR will be used; otherwise, the contents of the EIVR will be used. Both of these registers are set to the uninitialized vector number ($0F) by a reset operation.

3.5 CHANNEL PRIORITY REGISTER (CPR)

This register determines the priority level of a channel. When simultaneous operand transfer requests are pending in two or more channels, the highest priority channel will be serviced first. When two or more channels are at the same priority level, a round-robin service order is used to resolve simultaneous requests. The same priority scheme is used to resolve simultaneous pending interrupts during an interrupt acknowledge cycle; however, two independent circuits within the DMAC perform the priority arbitration for operand transfer requests and interrupt requests.

Channel Priority Register (CPR)

7	6	5	4	3	2	1	0
0	0	0	0	0	0	CP	

0   Not Used

CP   Channel Priority
            00   Priority 0 (Highest)
            01   Priority 1
            10   Priority 2
            11   Priority 3 (Lowest)

3.6 CONTROL REGISTERS

These four registers control the operation of the DMA channels by selecting various options for operand size, device characteristics, and address sequencing.

3.6.1 Device Control Register (DCR)

This register defines the characteristics of the device interface, including the device transfer request type, bus interface, port size, and the function of the PCL line.

Device Control Register (DCR)

7	6	5	4	3	2	1	0
SRM		DTYP		DPS	PCL		
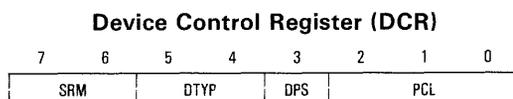
XRM   External Request Mode
- 00   Burst Transfer Mode
- 01   (Undefined, Reserved)
- 10   Cycle Steal Without Hold
- 11   Cycle Steal With Hold

DTYP   Device Type
- 00   M68000 Compatible, Explicitly Addressed
- 01   M6800 Compatible, Explicitly Addressed
- 10   Device with ACK, Implicitly Addressed
- 11   Device with ACK and RDY, Implicitly Addressed

DPS   Device Port Size
- 0   8-Bit Port
- 1   16-Bit Port

PCL   Peripheral Control Line Function
- 000   Status Input (Level Read in CSR)
- 001   Status Input with Interrupt
- 010   Start Pulse Output
- 011   Abort Input
- 100   (MC68440 Reserved)
- 101   ⎫
- 110   ⎬ (Undefined, Reserved)
- 111   ⎭

3.6.1.1 EXTERNAL REQUEST MODE. This field determines the method used by the device to request operand transfers. In the burst transfer mode, the $\overline{\text{REQ}}$ pin is a level sensitive input that generates transfer requests as long as it remains asserted. In the cycle steal modes, the $\overline{\text{REQ}}$ pin is an edge sensitive input; an asserted edge must occur to request each operand transfer. When the cycle steal with hold mode is selected, the DMAC will retain ownership of the bus for a programmable time period after an operand transfer occurs in order to wait for the next operand transfer request, thus reducing bus arbitration overload and response latency. For more detailed information, refer to **5.2.3.3 TRANSFER REQUEST GENERATION**.

3.6.1.2 DEVICE TYPE. This field selects one of four available device transfer protocols. For M68000 type devices, the DMAC will use a dual address transfer protocol by running M68000 type bus cycles to transfer data to or from the device registers and a second bus cycle to complete the operand transfer from or to memory (note that this device type also allows memory-to-memory block transfers). If the device is of the M6800 type, dual address transfers are also used; but the DMAC uses synchronous M6800 type bus cycles when accessing the device. In the remaining two device protocols, the DMAC asserts the acknowledge signal to implicitly address the device during a single address transfer while it is explicitly addressing a memory location. For a device with acknowledge only, the DMAC will assume that the device will place data onto or accept data from the bus in order to meet the timing requirements of a minimum length bus cycle and will terminate all bus cycles when the memory asserts $\overline{DTACK}$. For a device with acknowledge and ready, the PCL line is used as a ready signal that is similar to the $\overline{DTACK}$ signal. During a write to memory, the DMAC will not assert the data strobes to memory until the device has asserted ready. During a read from memory, the DMAC will not terminate the bus cycle until the device has asserted ready and memory has asserted $\overline{DTACK}$. Note that when the DTYP field is programmed for device with acknowledge and ready, the PCL function field in the DCR and the PCT and PCS bits in the CSR will be ignored.

3.6.1.3 DEVICE PORT SIZE. This field indicates how wide the device port is, either 8 or 16 bits. Refer to Table 5-1 for legal port and operand size combinations.

3.6.1.4 PERIPHERAL CONTROL LINE. If the DTYP field is not programmed for device with acknowledge and ready or explicit M6800, this field will define the function of the PCL line. The level on the PCL line can always be read from the PCS bit in the channel status register and can also be used to generate an interrupt when a high-to-low transition occurs on the pin. If programmed as an abort input, it is a high-to-low edge-sensitive signal that will cause the DMAC to abort the channel. The PCL line may also be programmed as a start pulse output, in which case it is asserted for eight clock cycles when a channel operation is started.

3.6.2 Operation Control Register (OCR)

This register defines the type of operation that will be performed by a channel. The parameters programmed in this register are the direction of the transfer, the operand size to be used, whether the operation is chained, and the type of request generation to be used.

Operation Control Register (OCR)

7	6	5	4	3	2	1	0
DIR	0	SIZE		CHAIN		REQG	

DIR   Transfer Direction
        0   From Memory to Device
        1   From Device to Memory

0   Not Used

SIZE   Operand Size
        00   Byte
        01   Word
        10   Long Word
        11   Byte, No Packing

CHAIN   Chain Operation
                  00   Chaining Disabled
                  01   (Undefined, Reserved)
                  10   Array Chaining Enabled
                  11   Linked Array Chaining Enabled

REQG   Request Generation
                  00   Internal Request at Limited Rate
                  01   Internal Request at Maximum Rate
                  10   External Request
                  11   Auto-Start, External Request

3.6.2.1 TRANSFER DIRECTION. The field controls the direction of the block transfer. A zero indicates that the transfer is from memory to the device, a one indicates that the transfer is from the device to memory.

3.6.2.2 OPERAND SIZE. This field defines the size of the data item to be transferred in response to each operand transfer request. The DMAC supports byte, word, and long word operand sizes; refer to Table 5-1 for legal port, memory, and operand size combinations.

3.6.2.3 CHAINING OPERATION. This field is used to program a channel for a chained operation. When chaining is enabled, the DMAC starts an operation by reading a data block descriptor, consisting of a memory address and transfer count, from a table pointed to by the BFCR and BAR. Array chaining uses a linear, sequential table of descriptors and the BTCR is used to count the number of descriptors remaining in the table. Linked array chaining uses a linked list table structure, with each descriptor containing a pointer to the next descriptor.

3.6.2.4 REQUEST GENERATION METHOD. This field defines the method used by the DMAC to detect operand transfer requests. Two types of internal request generation methods are available, requests may be externally generated, or the first request of a channel operation can be internally generated with
- subsequent requests generated externally. If one of the internal request generation methods is programmed, the XRM field in the DCR is ignored. Refer to **3.8 GENERAL CONTROL REGISTER (GCR)** for more details on programming the time intervals for the internal request at a limited rate method of request generation.

3.6.3 Sequence Control Register (SCR)

This register controls the manner in which the memory and device address registers are modified during transfer operations. Each address register may be programmed to remain unchanged during an operand transfer, or to be incremented or decremented after each operand is successfully transferred. The value by which a register is incremented or decremented after each DMA bus cycle depends on the operand size and the device port size; for more details on address sequencing, refer to **5.2.3.2 ADDRESS SEQUENCING**.

Sequence Control Register

7	6	5	4	3	2	1	0
0	0	0	0	MAC		DAC	
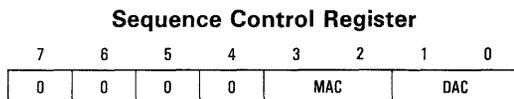
0   Not Used

MAC Memory Address Count
      00   MAR Does Not Count
      01   MAR Counts Up
      10   MAR Counts Down
      11   (Undefined, Reserved)

DAC Device Address Count
      00   DAR Does Not Count
      01   DAR Counts Up
      10   DAR Counts Down
      11   (Undefined, Reserved)

3.6.4 Channel Control Register (CCR)

This register is used to control the operation of the channel. By writing to this register, a channel operation may be started, set to be continued, halted, and aborted. Also, the channel can be enabled to issue interrupts when an operation is completed.

The STR and CNT bits are used to start the channel and to program it for continued operation. These bits can only be set by writing to them and are cleared internally by the DMAC. The STR bit will be cleared by the DMAC one clock after being set and the CNT bit is cleared when the MTCR is decremented to zero or by an error termination of the channel operation. The CNT bit must not be set at the beginning of a write cycle to the CCR to set the STR bit or an operation timing error will occur (the STR and CNT bits may be set by the same write cycle, however). Also, once the STR bit is set, none of the other control register may be modified or an operation timing error will occur. The CCR must be written with a byte operation when the STR bit is set, or an operation timing error will occur (i.e., the SCR may not be written simultaneously with the CCR when the STR bit is set).

The HLT bit may be used to suspend the channel operation at any time by writing a one to it and the operation may then be continued by clearing HLT. If it is desired to stop the channel and not continue at a later time, the SAB bit may be set at any time to abort a channel operation.

The INT bit controls the generation of interrupts by the DMAC. If the INT bit is set and the COC, BTC, or PCT (with PCL programmed as a status with interrupt input) bits are set in the CSR, then the $\overline{IRQ}$ signal will be asserted. If the PCL line is not programmed as an interrupt input, the PCT bit will be set by a high-to-low transition, but will not generate an interrupt.

If a one is written to the STR bit position, the one will not be stored, but the CSR will reflect the status of the channel after the start operation is executed; the ACT bit in the CSR will be set if no error occurs. The STR bit always reads as zero, and writing a zero to this position will have no effect on the channel operation.

Channel Control Register

7	6	5	4	3	2	1	0
STR	CNT	HLT	SAB	INT	0	0	0

STR Start Operation
      0   No Effect
      1   Start Channel

CNT   Continue Operation
      0   No Continuation is Pending
      1   Continue Operation at End of Block

HLT   Halt Operation
      0   Normal Channel Operation
      1   Halt Channel Operation

SAB   Software Abort
      0   Normal Channel Operation
      1   Abort Channel Operation

INT   Interrupt Enable
      0   Channel Will Not Generate Interrupts
      1   Channel May Generate Interrupts

0   Not Used

3.7 STATUS REGISTER

These two registers reflect the status of a channel operation. If an error is indicated by the channel status register (CSR), a more detailed description of the first error that occurred during an operation will be encoded in the lower five bits of the channel error register (CER).

3.7.1 Channel Status Register (CSR)

This register reflects the status of a channel operation. The status information supplied indicates if the channel is active or complete, whether an error has occurred, and whether or not the PCL signal has been asserted during the operation.

The ACT bit will be set when the channel is started by writing to the STR bit in the channel control register and will be cleared when the channel operation terminates. The COC, BTC, and NDT bits indicate the manner by which an operation terminated. COC is set when the channel is terminated, either successfully or for an error. BTC is set when the MTCR is decremented to zero and the CNT bit is set in the channel control register to indicate that a continued operation is complete and the DMAC is now transferring the next block. The NDT bit is set when the device terminates an operation by asserting the $\overline{\text{DONE}}$ signal. The ERR bit will be set any time that an error has occurred in the channel and indicates that the non-zero value in the channel error register reflects the type of the first error that has occurred since the channel was last started.

The PCT and PCS bits are associated with the function of the PCL signal. The PCT bit will be set anytime that a high-to-low transition occurs on the PCL line, regardless of its programmed function. The PCS bit reflects the instantaneous level of the PCL line; if this bit is set, the PCL line is at the high-voltage level, and if this bit is clear, the PCL line is at the low-voltage level.

Once they have been set by the DMAC, the COC, BTC, NDT, ERR, and PCT bits must be cleared either by a reset or by writing to the CSR. In order to clear a status bit, a one (1) is written to the bit position corresponding to that status bit. If a zero (0) is written to a status bit position, that bit will be unaffected; also, the ACT and PCS status bits are unaffected by all write operations to the CSR. In order to start channel operation, the COC, NDT, and ERR bits must be cleared; in order to perform a continued operation,

the BTC bit must be cleared before the CNT bit is set in the channel control register. Note that when the ERR bit is cleared, either by reset or by writing to it, the channel error register will also be cleared.

Channel Status Register

7	6	5	4	3	2	1	0
COC	BTC	NDT	ERR	ACT	0	PCT	PCS

COC   Channel Operation Complete
      0   Channel Operation Not Complete
      1   Channel Operation Has Completed

BTC   Block Transfer Complete
      0   Block Transfer Not Complete
      1   Continued Block Transfer Has Completed

NDT   Normal Device Termination
      0   Operation Not Terminated by Device
      1   Device Terminated Operation with DONE

ERR   Error
      0   No Error
      1   Error Has Occurred

ACT   Channel Active
      0   Channel Not Active
      1   Channel Is Active

0   Not Used

PCT   PCL Transition
      0   No PCL Transition Has Occurred
      1   High-to-Low PCL Transition Occurred

PCS   PCL State
      0   PCL Line is Low
      1   PCL Line is High

3.7.2 Channel Error Register (CER)

This register indicates the cause of the first error that has occurred since the ERR bit in the channel status register was last cleared. When the ERR bit in the CSR is set, bits 0-4 indicate the type of error that occurred. Refer to **5.6.3 Error Termination** for more information on the cause of the errors reported by the CER.

Channel Error Register

7	6	5	4	3	2	1	0
0	0	0	ERROR CODE				

0   Not Used

Error	00000	No Error
Code	00001	Configuration Error
	00010	Operation Timing Error
	00011	(Undefined, Reserved)
	001rr	Address Error
	010rr	Bus Error
	011rr	Count Error
	10000	External Abort
	10001	Software Abort
	10010	
	•	
	•	(Undefined, Reserved)
	•	
	11111	

rr   Register Code
         00   (MC68440 Reserved)
         01   MAR or MTCR
         10   DAR
         11   BAR or BTCR

3.8 GENERAL CONTROL REGISTER (GCR)

This is a global register that may affect the operation of all channels. If a channel is programmed for internal limited-rate request generation, the GCR is used to determine the time constants for the limited-rate timing. Refer to **5.2.3.2 Internal, Limited Rate** for more details on limited rate auto-request operation. If a channel is programmed for cycle steal with hold request generation, the DMAC will retain ownership of the bus until the end of the next full sample period (as defined in 5.2.3.3.2) after an operand transfer is completed. Refer to **5.4.4 External Request Recognition** for more details on cycle steal with hold operation.

General Control Register

7	6	5	4	3	2	1	0
0	0	0	0	BT		BT	

0   Not Used

BT   Burst Transfer Time
         00   16 Clocks
         01   32 Clocks
         10   64 Clocks
         11   128 Clocks

BR   Bandwidth Available to DMAC
         00   50.00%
         01   25.00%
         10   12.50%
         11   6.25%

3.9 REGISTER SUMMARY

A summary of the bit definitions for each of the channel registers along with the address of each register within the DMAC register memory map is provided in Figure 3-2.

3.10 RESET OPERATION RESULTS

When the DMAC is reset, either during a system power-up sequence or to re-initialize the DMAC, many of the registers will be affected and will be set to known values. Table 3-1 shows the hexadecimal value that will be placed in each register by a reset operation.

Table 3-1. Reset Operation Results

Register	Value	Comments
MARc	XXXXXXXX	Not Affected
DARc	XXXXXXXX	
BARc	XXXXXXXX	
MFCRc	X	
DFCRc	X	
BFCRc	X	
MTCRc	XXXX	
BRCRc	XXXX	
NIVRc	OF	Uninitialized Vector
EIVRc	OF	Uninitialized Vector
CPRc	00	
DCRc	00	
OCRc	00	
SCRc	00	
CCRc	00	Channel Not Active,
		Interrupts Disabled
CSRc	00 or 01	(Depending on the Level of PCLc)
CERc	00	No Errors
GCR	00	

X—Indicates an unknown value or the previous value of the register.
c—is the channel number (i.e., 0, 1, 2, or 3)

1-151

SECTION 4
BUS OPERATION

This section describes the bus operation of the DMAC in all operating modes. All bus operations are described in relation to the DMAC CLK signal, with all signal activity taking place during a certain period of the CLK input. The terminology and conventions used are identical to the bus description for the MC68000 16-bit microprocessor. Functional timing diagrams are included to assist in the definition of signal timings; however, these diagrams are not intended as parametric timing definitions. For detailed timing relationships between various signals, refer to **SECTION 6 ELECTRICAL SPECIFICATIONS**.

The term "synchronization delay" will be used repeatedly when discussing bus operation. This delay is the time period required for the DMAC to sample an external asynchronous input signal, determine whether it is high or low, and synchronize the input to the DMAC internal clocks. Figure 4-1 shows the relationship between the CLK signal, an external input, and its associated internal signal that is typical for all of the asynchronous inputs. Note that some input signals are synchronized on the rising edge of the CLK signal as shown, while others are synchronized on the falling edge of the CLK signal (in which case the EXT and INT signals below would be shifted to the left or right by one-half clock cycle). Refer to Table 2-1 for the synchronizing edge for each input signal.

1-846
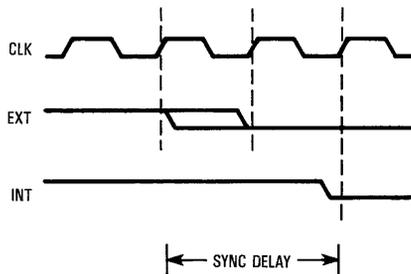
**Figure 4-1. Relationship Between
External and Internal Signals**

4.1 RESET OPERATION

The DMAC can be reset to the IDLE mode by an external device at any time by asserting the reset encoding on $\overline{BEC0}$-$\overline{BEC2}$. Two types of reset operations may be performed by external hardware, a power-on reset and a system reset after power has been stable for a long period of time. During power-on reset, the reset signal must be asserted to the DMAC for at least 100 milliseconds after V_{CC} has reached its nominal operating level, as shown in Figure 4-2. After power has been stable for a long period of time, reset must be asserted for at least ten clock cycles to reset the DMAC.
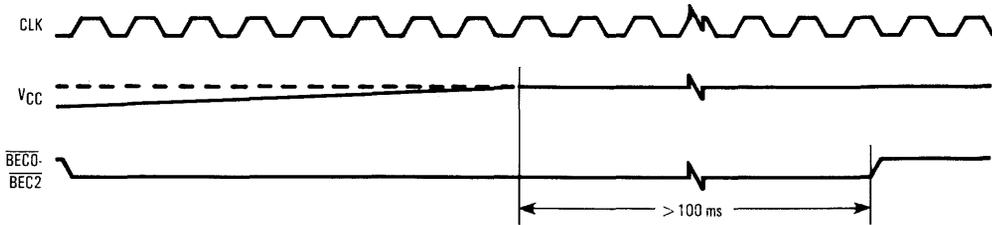
Figure 4-2. Reset Operation Timing Diagram

4.2 MPU MODE OPERATION

In the MPU mode, the DMAC acts as a peripheral slave to another bus master. When the $\overline{CS}$ and $\overline{UDS}$ or $\overline{LDS}$ signals are asserted, the DMAC will enter the MPU mode from the idle mode (if the DMAC is in the DMA mode and $\overline{CS}$ is asserted, an error will be signaled). During MPU mode operations, the DMAC will accept data from or place data on the data bus according to the level of the $R/\overline{W}$ pin. The data transferred will either be placed into or come from the internal register(s) that is selected by the encoding on A1-A7, $\overline{UDS}$, and $\overline{LDS}$. Once the data transfer has been completed, the DMAC will assert $\overline{DTACK}$ to terminate the bus cycle and return to the IDLE mode.

During MPU mode operations, the DMAC will synchronize input signals to the CLK input and all outputs will occur in relation to the CLK signal; however, the CLK does not need to be synchronous with the bus master's clock. Thus, the DMAC will appear to be completely asynchronous to the bus master unless they both use the same clock signal. In the functional diagrams showing MPU operations, the bus master is assumed to be an MC68000 or MC68010 with a clock signal identical to the DMAC CLK signal.

4.2.1 MPU Read Cycles

During MPU read cycles, the DMAC will place data on the data bus and assert $\overline{DTACK}$ to indicate to the bus master that the data from the register(s) selected is available to it. Figure 4-3 shows the functional timing for a word read cycle. The timing for even and odd byte MPU reads is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte.
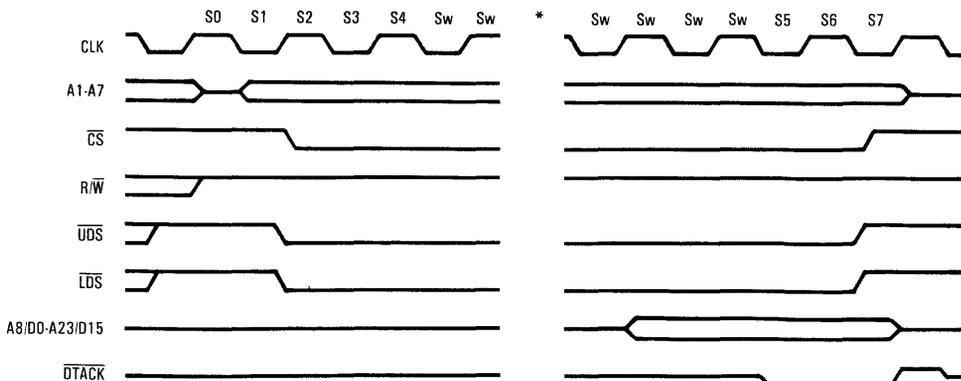
*11 Additional Wait Cycles

Figure 4-3. MPU Read Cycle Timing Diagram

The DMAC starts an MPU read operation when $\overline{CS}$ is asserted and synchronized internally with the R/$\overline{W}$ line high. The DMAC responds to the $\overline{CS}$ by decoding A1-A7, $\overline{UDS}$, and $\overline{LDS}$ and placing the data from the selected register(s) on the appropriate address/data pins, driving $\overline{DDIR}$ high, asserting $\overline{DBEN}$ and asserting $\overline{DTACK}$. The DMAC will then wait until $\overline{CS}$ is negated and three state the address/data pins, $\overline{DDIR}$, $\overline{DBEN}$, and $\overline{DTACK}$.

4.2.2 MPU Write Cycles

During MPU write cycles, the DMAC accepts data from the data bus and places it into the register(s) selected by the bus master. Figure 4-4 shows the functional timing for a word write cycle. The timing for even and odd byte MPU writes is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte.

*Eight Additional Wait Cycles
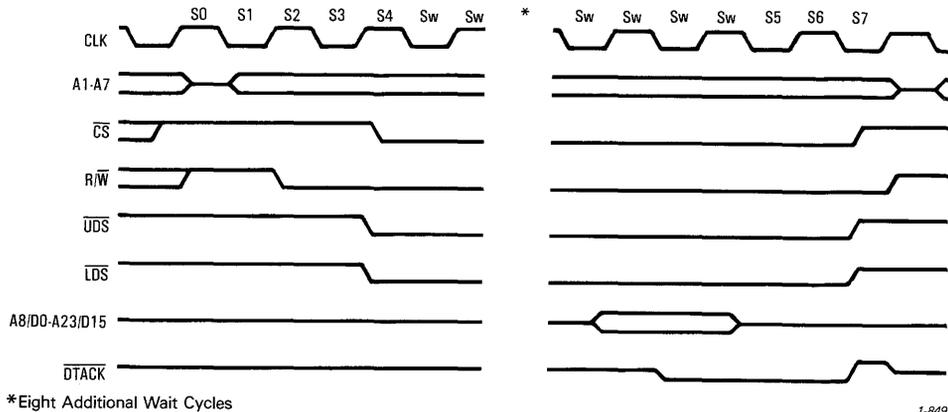
Figure 4-4. MPU Write Cycle Timing Diagram

The DMAC starts an MPU write operation when $\overline{CS}$ is asserted and synchronized internally with the R/$\overline{W}$ line low. Note that input data should be valid at the data buffers when $\overline{CS}$ is asserted; thus, the $\overline{CS}$ equation should include the data strobe(s). The DMAC responds to the $\overline{CS}$ by decoding A1-A7, $\overline{UDS}$, and $\overline{LDS}$, driving $\overline{DDIR}$ low and asserting $\overline{DBEN}$. Data is then taken from the appropriate address/data pins and placed into the selected register(s) and $\overline{DTACK}$ is asserted. The bus master should hold the data valid until $\overline{CS}$ is negated, at which time the DMAC will three state $\overline{DDIR}$, $\overline{DBEN}$, and $\overline{DTACK}$.

4.2.3 Interrupt Acknowledge Cycles

A special case of the MPU mode of operation is the interrupt acknowledge cycle during which the system processor is responding to an interrupt request from the DMAC. The timing of an interrupt acknowledge cycle is identical to an odd byte read cycle via $\overline{CS}$, except that it is started by the assertion of the $\overline{IACK}$ signal rather than $\overline{CS}$. $\overline{IACK}$ and $\overline{CS}$ are mutually exclusive signals and must not be asserted at the same time. Also, $\overline{IACK}$ should not be asserted while the DMAC is acting as a bus master, or an address error will occur for the channel that is being serviced when the $\overline{IACK}$ signal is asserted. Figure 4-5 shows the functional timing for an interrupt acknowledge cycle.

The DMAC starts an interrupt acknowledge operation when $\overline{IACK}$ is asserted and synchronized internally. The DMAC responds to the $\overline{IACK}$ by placing a vector number on A8/D0-A15/D7, driving $\overline{DDIR}$ high and asserting $\overline{DBEN}$ and $\overline{DTACK}$. The A1-A7, $\overline{UDS}$, and $\overline{LDS}$ signals are all ignored during an interrupt acknowledge cycle. The vector number placed on the bus will be taken from either the NIVR or EIVR of the highest priority channel with an interrupt pending. The vector number will remain valid until the $\overline{IACK}$ signal is negated, at which time the DMAC will three state the address/data bus and negate $\overline{DDIR}$, $\overline{DBEN}$, and $\overline{DTACK}$.

*11 Additional Wait Cycles

1-850
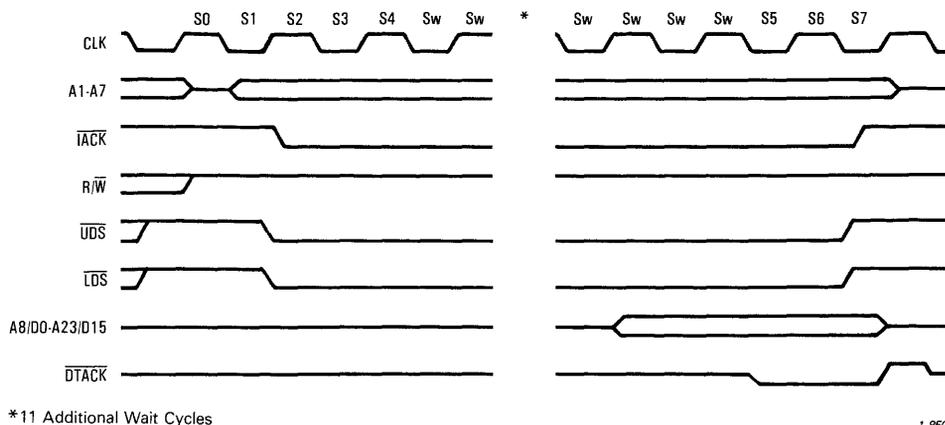
Figure 4-5. Interrupt Acknowledge Cycle Timing Diagram

4.3 DMA MODE OPERATION

In the DMA mode of operation, the DMAC is acting as a bus master and performs data transfer operations between a device and memory in response to operand transfer requests. There are three classes of DMA bus cycles that may be run by the DMAC; M68000 cycles that transfer data between a device or memory and the DMAC internal holding register, M6800 cycles that transfer data between an M6800 Family device and the DMAC internal holding register, and cycles that transfer data directly between a device and memory without going through the DMAC. The first two classes of cycles are referred to as dual address cycles, since any operand transfer takes place in two bus cycles, one where a device is addressed and one where memory is addressed. The second class of cycle is referred to as a single address transfer since the operand transfer takes place in one bus cycle where only the memory is explicitly addressed. Within each class of DMA bus cycles, the DMAC may execute a read or a write operation. The following paragraphs describe the six DMA cycle types according to the signal transitions during each clock cycle of a bus operation.

4.3.1 Dual Address Transfers

Dual address transfer signal timing closely resembles the signal timing of an M68000 processor. The DMAC starts a bus cycle by presenting function code, address, direction, and data size information to an external device and then waits for the addressed device to present or accept data and terminate the bus cycle. Since the bus structure used is asynchronous, the memory or device access time can be dynamically changed according to the needs of the current bus cycle. The data that is transferred during a dual address operation is either read from the data bus into the DMAC internal holding FIFO or written from that FIFO to the data bus. All cycle termination options of the M68000 bus are supported, such as halt, cycle retry, or bus error. Additionally, the M6800 synchronous bus protocol is supported for device accesses.

Note that several combinations of device and operand sizes may be used with dual address transfers, including combinations where the memory and device width are different. When an 8-bit device is used, the device may be placed on the most significant (even address) or least significant (odd address) half of the data bus.

4.3.1.1 DUAL ADDRESS READ, M68000 TYPE.
During this type of a DMA cycle, the DMAC will read data from a device or memory into an internal holding FIFO, the MAR, MTCR, or BAR. Figure 4-6 shows the functional timing for a word read. The timing for an even or odd byte read is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte. The signal protocol is as follows.
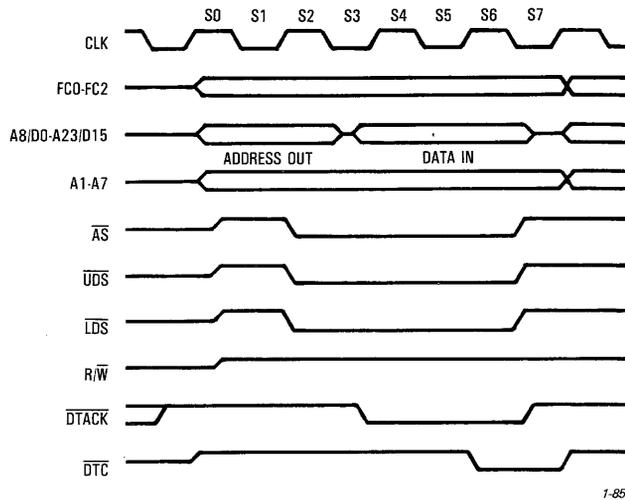
Figure 4-6. M68000 Type Dual Address Read Cycle Timing Diagram

S0 The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the function code and address registers (the MFCR and MAR, DFCR and DAR, or BFCR and BAR) of the current channel, $\overline{\text{UAS}}$ is asserted to enable the external address latches and R/$\overline{\text{W}}$ is driven high.

S1 No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{\text{UAS}}$ is negated to latch the upper address information and $\overline{\text{AS}}$ is asserted to signal that the address is valid and decoding may begin. $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ will also assert to indicate that data should be placed on the bus and $\overline{\text{DDIR}}$ will be driven low to prepare the external data buffers/demultiplexers to drive the address/data pins when enabled. If the current bus cycle is to a device, $\overline{\text{ACK}}$ will also be asserted to the device and $\overline{\text{DONE}}$ will be asserted if this is the last transfer to the device for the current data block.

S3 The address/data pins are three stated in preparation to receive input data and $\overline{\text{DBEN}}$ is asserted to enable the data buffer/demultiplexer. The DMAC also starts sampling the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins to determine when the cycle should be terminated. If $\overline{\text{DTACK}}$ is asserted by the end of S3, then the DMAC may continue into S4 immediately. If $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S3, two wait cycles will be inserted between S3 and S4. If $\overline{\text{DTACK}}$ is negated and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are coded to normal at the end of S3, then the DMAC will insert wait cycles after S3 until a termination signal is asserted, and then proceed to S4.

S4 No signals change during S4. The memory or device access is taking place at this time and data is allowed to propagate through external buffers, etc. The DMAC is also internally synchronizing the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ signals.

S5 No signals change during S5. The memory or device access continues and the DMAC completes synchronization of the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins.

S6   Data is sampled during this period, with the data bus value latched at the end of S6. Thus, valid data must be present at the external data buffers/demultiplexers such that it may propagate through those devices and be valid at the DMAC address/data pins within the setup time to the falling edge of S6. For byte read operations, the data lines for the byte not being read will be ignored. $\overline{DTC}$ is asserted to indicate that the bus cycle has been successfully terminated. $\overline{DTC}$ will not be asserted if a $\overline{BEC0}$-$\overline{BEC2}$ coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S7 $\overline{AS}$ is negated to indicate that the cycle has terminated. $\overline{UDS}$ and/or $\overline{LDS}$ and $\overline{DBEN}$ will also negate to disable external data buffers/demultiplexers. The memory or peripheral device that was addressed during the cycle may then negate $\overline{DTACK}$ and/or $\overline{BEC0}$-$\overline{BEC2}$.

S7 + 1   This may be S0 of the following cycle, if the DMAC is ready to immediately execute another bus cycle. The FC0-FC2 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DMAC is not ready to start a bus cycle, these lines will be three stated. $\overline{DDIR}$ is negated to turn the data buffers/demultiplexers away from the DMAC. $\overline{ACK}$, $\overline{DTC}$, and $\overline{DONE}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed.

4.3.1.2 DUAL ADDRESS WRITE, M68000 TYPE. During this type of a DMA cycle, the DMAC will write data to a device or memory from an internal holding register. Figure 4-7 shows the functional timing for a word write on a 16-bit bus. The timing for an even or odd byte write is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte. The signal protocol is as follows.

S0   The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the function code and address registers (either the MFCR and MAR or DFCR and DAR) of the current channel and $\overline{UAS}$ is asserted to enable the external address latches. $\overline{DDIR}$ will be driven high to prepare the external data buffers/demultiplexers to drive the external data bus when enabled.

S1   No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and decoding may begin. R/$\overline{W}$ is also driven low to indicate that this cycle is a write.

S3   The address/data pins change from address to data and $\overline{DBEN}$ is asserted to allow data to propagate through the external data buffers/demultiplexers. For byte write operations, the value of the data lines for the byte not being written is undefined. If the cycle is accessing a device, $\overline{ACK}$ will be asserted at this time and $\overline{DONE}$ will be asserted if this is the last transfer to the device for the current block operation. The DMAC also begins to sample the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins to determine when the cycle should be terminated. If $\overline{DTACK}$ is asserted by the end of S3, then the DMAC will 'skip' S6 and S7 and proceed directly from S5 to S8. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S3, one wait cycle will be inserted between S5 and S6. If $\overline{DTACK}$ is negated and $\overline{BEC0}$-$\overline{BEC2}$ are coded to normal at the end of S3, then the DMAC will execute S6 and S7.
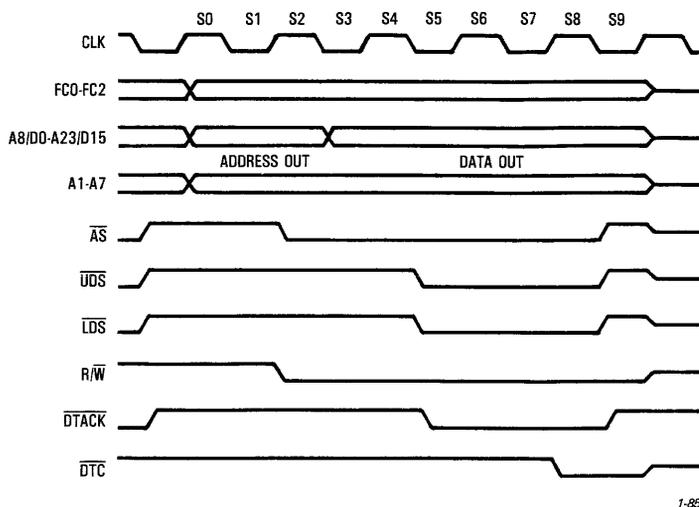
Figure 4-7. M68000 Type Dual Address Write Cycle Timing Diagram

S4 No signals change during S4. Data from the DMAC is allowed to propagate to the device or memory through external buffers, etc.

S5 $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ is asserted at this time to indicate that valid data is present on the bus and may be latched. The DMAC continues to sample the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins to determine when the cycle should be terminated. If $\overline{\text{DTACK}}$ is asserted by the end of S5, then the DMAC will continue into S6 immediately. If $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S5, two wait cycles will be inserted between S5 and S6. If $\overline{\text{DTACK}}$ is negated and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are coded to normal at the end of S5, then the DMAC will insert wait cycles after S5 until a termination signal is asserted and then proceed to S6.

S6 No signals change during S6. The DMAC is internally synchronizing the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins.

S7 No signals change during S7. The DMAC continues to synchronize the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins.

S8 $\overline{\text{DTC}}$ is asserted to indicate that the bus cycle has been successfully terminated. $\overline{\text{DTC}}$ will not be asserted if a $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S9 $\overline{\text{AS}}$ is negated to indicate that the cycle has terminated. $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ are negated to disable the memory or device data buffers. The memory or device that was addressed during the cycle may then negate $\overline{\text{DTACK}}$ and/or $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$.

S9 + 1 This may be S0 of the following cycle, if the DMAC is ready to execute another bus cycle immediately. The FC0-FC1 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DMAC is not ready to start a bus cycle, these lines will be three stated. $\overline{\text{DBEN}}$ is negated and $\overline{\text{DDIR}}$ and R/$\overline{\text{W}}$ are driven high to prepare the data bus for the next cycle. $\overline{\text{ACK}}$, $\overline{\text{DTC}}$, and $\overline{\text{DONE}}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed.

4.3.1.3 DUAL ADDRESS READ, M6800 TYPE DEVICE. During this type of a DMA cycle, the DMAC will read data from a device into an internal holding register, using an M6800 type synchronous bus cycle. Figure 4-8 shows the functional timing for a read from an M6800 type device. Note that $\overline{\text{LDS}}$ is asserted and $\overline{\text{UDS}}$ is negated during the cycle shown, since M6800 peripherals are eight bits wide and will normally be placed on the lower half of the data bus. The DMAC also will allow an M6800 type device to be accessed on the upper half of the data bus, and will allow word accesses using the M6800 synchronous bus interface. The signal protocol is as follows.

S0-1   The DMAC samples the E clock input (the PCL line) at the end of the state prior to the start of the M6800 bus cycle, and stores the level for later use. Throughout the remainder of this cycle, the E clock will be sampled on each rising edge of the clock and the level that is latched will be compared with the level latched on the previous rising edge of the clock. If the two sampled levels are different, then the DMAC will take action appropriate for the type of transition (high-to-low or low-to-high) that has been detected on the E input.

S0   The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the DFCR and DAR of the current channel, $\overline{\text{UAS}}$ is asserted to enable the external address latches and R/$\overline{\text{W}}$ is driven high.

S1   No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{\text{UAS}}$ is negated to latch the upper address information and $\overline{\text{AS}}$ is asserted to signal that the address is valid and the bus is in use. $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ are asserted to indicate that data should be placed on the bus and $\overline{\text{DDIR}}$ is driven low to prepare the external data buffers/demultiplexers to drive the address/data pins when enabled. $\overline{\text{DONE}}$ will also be asserted if this is the last operand transfer for the block operation. If a high-to-low transition on E was detected during S1, then $\overline{\text{ACK}}$ will be asserted as a valid memory address (VMA) strobe to indicate the start of an M6800 peripheral access; otherwise, the DMAC will insert wait cycles after S3 until a high-to-low transition of E is detected before asserting $\overline{\text{ACK}}$.

S3   The address/data pins are three stated in preparation to receive input data and $\overline{\text{DBEN}}$ is asserted to enable the data buffer/demultiplexer.

Sw   After the end of S3 and before the start of S4, the DMAC will insert wait cycles (of two wait states each) until $\overline{\text{ACK}}$ is asserted (following a high-to-low transition on E) and one low-to-high followed by one high-to-low transition occurs on the E clock. No output signals change during this synchronization period, and the M6800 device access is performed during the E clock high period.

S4   No signals change during S4. The DMAC is internally synchronizing the second high-to-low transition of the E clock.

S5   No signals change during S5. The DMAC continues to synchronize the second high-to-low transition of the E clock.

S6   Data is sampled during this period, with the data bus value latched at the end of S6. For byte read operations, the data lines for the byte not being read will be ignored. $\overline{\text{DTC}}$ is asserted to indicate that the bus cycle has been successfully terminated.
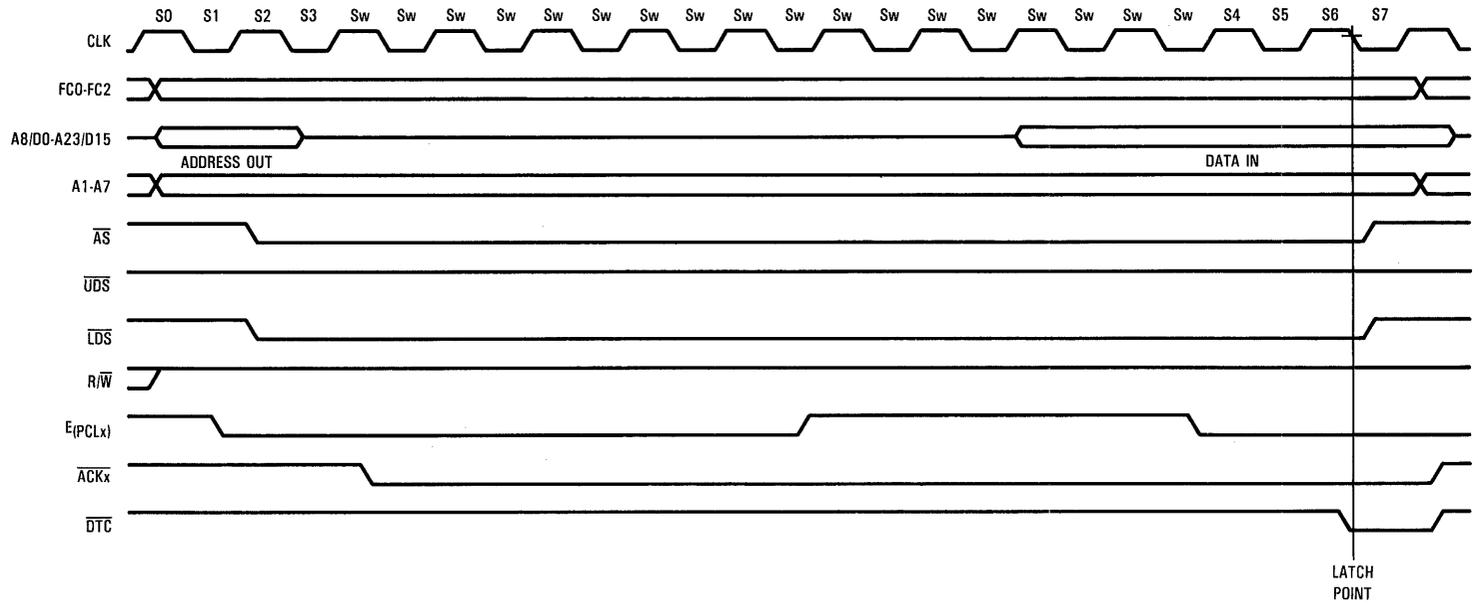
Figure 4-8. M6800 Type Read Cycle Timing Diagram

S7 $\overline{AS}$ is negated to indicate that the cycle has terminated. $\overline{UDS}$ and/or $\overline{LDS}$ and $\overline{DBEN}$ will also negate to disable external data buffers/demultiplexers.

S7 + 1   This may be S0 of the following cycle, if the DMAC is ready to execute another bus cycle immediately. The FC0-FC2 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DMAC is not ready to start a bus cycle, these lines will be three stated. $\overline{DDIR}$ is negated to turn the data buffers/demultiplexers away from the DMAC. $\overline{ACK}$ (VMA), $\overline{DTC}$ and $\overline{DONE}$ (if it was asserted) are all negated to indicate to the peripheral that the cycle has completed.

NOTE

As indicated in Figure 4-8, the DMAC does not latch data until after the E clock is low at the end of the cycle. Since M6800 peripherals remove data from the bus when E goes low, a latch must be provided to hold data valid at the DMAC until $\overline{AS}$ goes high.

4.3.1.4 DUAL ADDRESS WRITE, M6800 TYPE DEVICE. During this type of a DMA cycle, the DMAC will write data to a device from an internal holding register, using an M6800 type synchronous bus cycle. Figure 4-9 shows the functional timing for a write to an M6800 type device. Note that $\overline{LDS}$ is asserted and $\overline{UDS}$ is negated during the cycle shown, since M6800 peripherals are eight bits wide and will normally be placed on the lower half of the data bus. The DMAC also will allow an M6800 type device to be accessed on the upper half of the data bus, and will allow word accesses using the M6800 synchronous bus interface. The signal protocol is as follows.

S0-1   The DMAC samples the E clock input (the PCL line) at the end of the state prior to the start of the M6800 bus cycle, and stores the level for later use. Throughout the remainder of this cycle, the E clock will be sampled on each rising edge of the clock and the level that is latched will be compared with the level latched on the previous rising edge of the clock. If the two sampled levels are different, then the DMAC will take action appropriate for the type of transition (high-to-low or low-to-high) that has been detected on the E input.

S0   The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the DFCR and DAR of the current channel and $\overline{UAS}$ is asserted to enable the external address latches. $\overline{DDIR}$ will be driven high to prepare the external data buffers/demultiplexers to drive the external data bus when enabled.

S1   No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and the bus is in use. R/$\overline{W}$ is also driven low to indicate that this cycle is a write.

S3   The address/data pins change from address to data and $\overline{DBEN}$ is asserted to allow data to propagate through the external data buffers/demultiplexers. For byte write operations, the value of the data lines for the byte not being written is undefined. $\overline{DONE}$ will be asserted if this is the last transfer to the device for the current block operation.

S4   No signals change during S4. Data from the DMAC is propagated to the device or memory through external buffers, etc. If a high-to-low transition on E was detected during S1 or
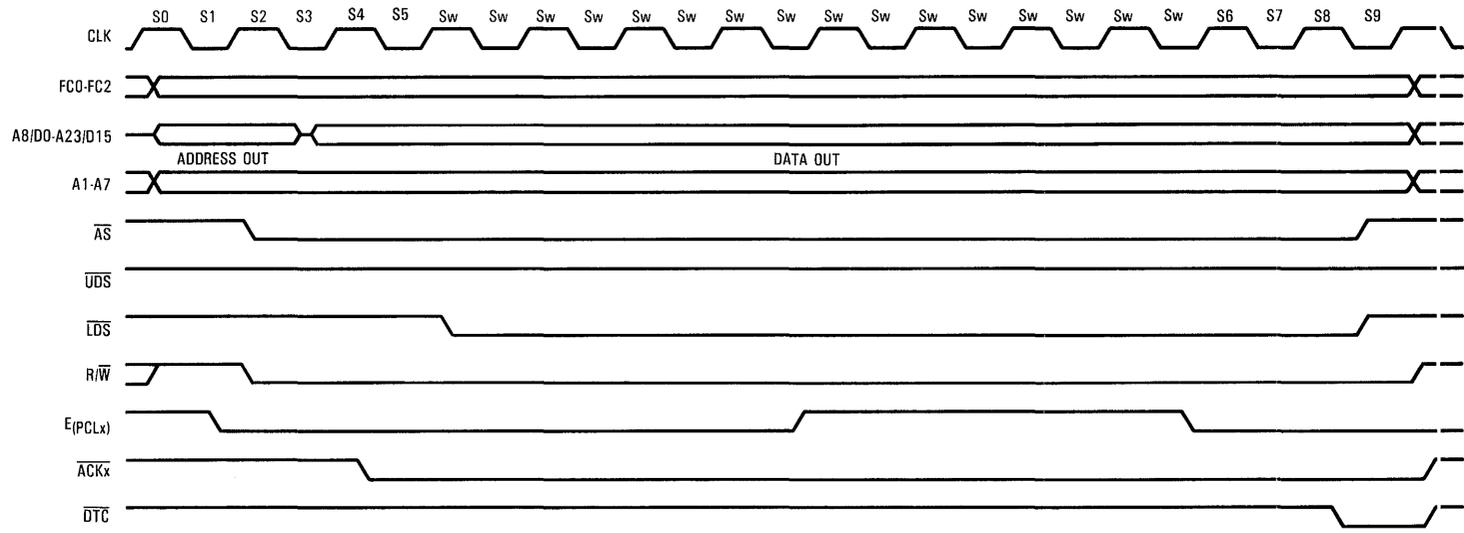
Figure 4-9. M6800 Type Write Cycle Timing Diagram

S3, then $\overline{\text{ACK}}$ will be asserted as a valid memory address (VMA) strobe to indicate the start of an M6800 peripheral access; otherwise, the DMAC will insert wait cycles after S5 until a high-to-low transition of E is detected before asserting $\overline{\text{ACK}}$.

S5 $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ is asserted to indicate that valid data is present on the bus.

Sw   After the end of S5 and before the start of S6, the DMAC will insert wait cycles (of two wait states each) until $\overline{\text{ACK}}$ is asserted (following a high-to-low transition on E) and one low-to-high followed by one high-to-low transition occurs on the E clock. No output signals change during this synchronization period; and the M6800 device access is performed during the E clock high period, with data latched by the device on the falling edge of E.

S6   No signals change during S6. The DMAC is internally synchronizing the second high-to-low transition of the E clock.

S7   No signals change during S7. The DMAC continues to synchronize the second high-to-low transition of the E clock.

S8 $\overline{\text{DTC}}$ is asserted to indicate that the bus cycle has been successfully terminated.

S9 $\overline{\text{AS}}$, $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ are negated to indicate that the cycle has terminated.

S9 + 1   This may be S0 of the following cycle, if the DMAC is ready to execute another bus cycle immediately. The FC0-FC2 and A1-A7 lines will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DMAC is not ready to start a bus cycle, these lines will be three stated. $\overline{\text{DBEN}}$ is negated, and $\overline{\text{DDIR}}$ and R/$\overline{\text{W}}$ are driven high to prepare the data bus for the next cycle. $\overline{\text{ACK}}$ (VMA), $\overline{\text{DTC}}$, and $\overline{\text{DONE}}$ (if it was asserted) are all negated to indicate to the peripheral that the cycle has completed.

4.3.2 Single Address Transfers

Single address transfer signal timing is similar to the timing for dual address transfers, except data flows directly between a device and memory without passing through the DMAC. There are two types of single address transfer protocols that are used to support different device types. A device with acknowledge uses a two signal handshake ($\overline{\text{REQ}}$ and $\overline{\text{ACK}}$) to perform transfers. A device with acknowledge and ready uses a three signal handshake ($\overline{\text{REQ}}$, $\overline{\text{ACK}}$, and PCL) to perform transfers.

When the DMAC is executing a single address transfer bus cycle, it will drive the address bus and the control signals for both the address and data bus, but will not put data onto, or read data from, the data bus. The device control signals $\overline{\text{ACK}}$, PCL (for a device with ready), and $\overline{\text{DTC}}$ are used to synchronize the device to the memory so that the data can be transferred directly between them. In this way, the combination of the DMAC and the device appears to execute bus cycles in the same manner as an MPU as far as the memory is concerned.

4.3.2.1 SINGLE ADDRESS READ. During this type of DMA cycle, the DMAC will control the transfer of data from memory to a device. Figure 4-10 shows the functional timing for an even byte read to an 8-bit device. The timing for a word or odd byte read to an 8-bit or 16-bit device is identical, with the encoding of $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ selecting the proper byte(s) and $\overline{\text{HIBYTE}}$ and R/$\overline{\text{W}}$ controlling the flow of data from D8-D15 to D0-D7 if necessary (which will only occur for the case shown in Figure 4-10).
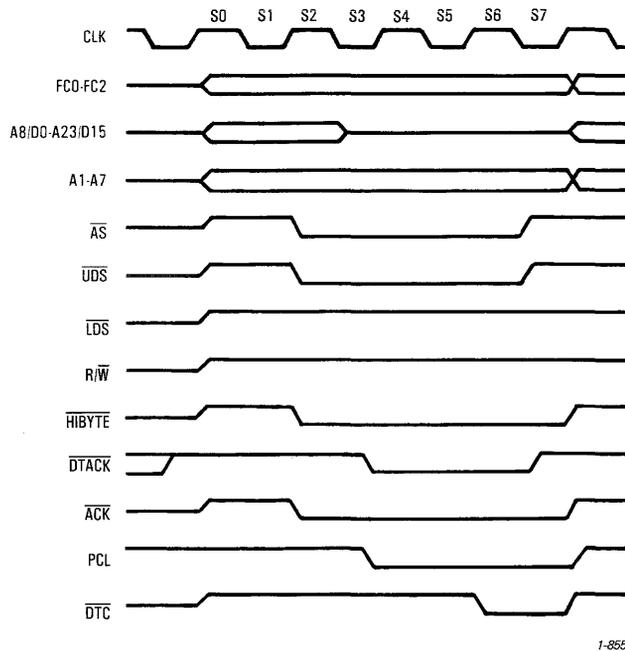
Figure 4-10. Single Address Read Cycle Timing Diagram

S0   The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the MFCR and MAR of the current channel, $\overline{\text{UAS}}$ is asserted to enable the external address latches, and R/$\overline{\text{W}}$ is driven high. $\overline{\text{DBEN}}$ will be driven high and remain high through the end of the cycle so that the data buffers/multiplexers will not interfere with the data transfer.

S1   No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{\text{UAS}}$ is negated to latch the upper address information and $\overline{\text{AS}}$ is asserted to signal that the address is valid and decoding may begin. $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ will assert to indicate that the memory should place data on the bus. $\overline{\text{ACK}}$ is also asserted to the device to indicate that the cycle is beginning, and $\overline{\text{DONE}}$ will be asserted if this is the last transfer for the current block operation. If the device port size is eight bits and the address is even, $\overline{\text{HIBYTE}}$ will be asserted to cause the memory data from D8-D15 to be driven onto D0-D7 through an external buffer (in this case, $\overline{\text{UDS}}$ will be asserted and $\overline{\text{LDS}}$ will be negated).

S3   The address/data pins are three stated and the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins will be sampled to determine when the cycle should be terminated. If $\overline{\text{DTACK}}$ is asserted by the end of S3, then the DMAC may continue into S4 immediately. If $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S3, two wait cycles will be inserted between S3 and S4. If $\overline{\text{DTACK}}$ is negated and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are coded to normal at the end of S3, then the DMAC will insert wait cycles after S3 until a termination signal is asserted and then proceed to S4.

If the device protocol for the selected device is acknowledge with ready, the PCL input will also be sampled as a ready signal from the device. In order for the DMAC to terminate the bus cycle, the memory must present a termination signal and the device must drive PCL low. In this way, the device can monitor the $\overline{DTACK}$ signal from the memory to determine when data is available, latch it, and terminate the bus cycle when access time requirements have been met. If either the memory or device has not asserted its termination signal by the end of S3, wait cycles will be inserted until both termination conditions are met and then the DMAC will proceed to S4.

S4   No signals change during S4. The memory may be placing data on the bus at this time. The DMAC is internally synchronizing the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins in both protocols and the PCL pin for the device with ready protocol.

S5   No signals change during S5, but data should be valid at the device port by the end of S5. The DMAC continues to synchronize the $\overline{DTACK}$, $\overline{BEC0}$-$\overline{BEC2}$, and PCL signals.

S6 $\overline{DTC}$ is asserted to the device at this time to indicate that the cycle is being terminated normally and that valid data may be latched from the bus. $\overline{DTC}$ will not be asserted if a $\overline{BEC0}$-$\overline{BEC2}$ coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S7 $\overline{AS}$ is negated to indicate to the memory that the cycle has terminated. $\overline{UDS}$ and/or $\overline{LDS}$ will also negate at this time and the memory may respond by negating $\overline{DTACK}$ and/or $\overline{BEC0}$-$\overline{BEC2}$.

S7 + 1   This may be S0 of the next bus cycle, if the DMAC is ready to execute another bus cycle immediately. The FC0-FC2 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory location. If the DMAC is not ready to start a bus cycle, these lines will be three stated. $\overline{ACK}$, $\overline{DTC}$, $\overline{DONE}$, and $\overline{HIBYTE}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed, and the device may negate the ready (PCL) input if the acknowledge with ready protocol was used.

4.3.2.2 SINGLE ADDRESS WRITE. During this type of a DMA cycle, the DMAC will control the transfer of data from a device to memory. Figure 4-11 shows the functional timing for an even byte write from an 8-bit device. The timing for a word or odd byte write from an 8-bit or 16-bit device is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte(s), and $\overline{HIBYTE}$ and R/$\overline{W}$ controlling the flow of data from D0-D7 to D8-D15 if necessary (which will only occur for the case shown in Figure 4-11.

S0   The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the MFCR and MAR of the current channel and $\overline{UAS}$ is asserted to enable the external address latches. $\overline{DBEN}$ will be driven high and remain high through the end of the cycle so that the data buffers/multiplexers will not interfere with the data transfer.

S1   No signals change and the upper address information is allowed to propagate through the external address latches. If the device with acknowledge and ready protocol is used, PCL will first be sampled on the rising edge at the end of S1; thus, if PCL is low by the end of S1, SD0 and SD1 will not occur and the timing will be the same as for the device with acknowledge-only protocol.
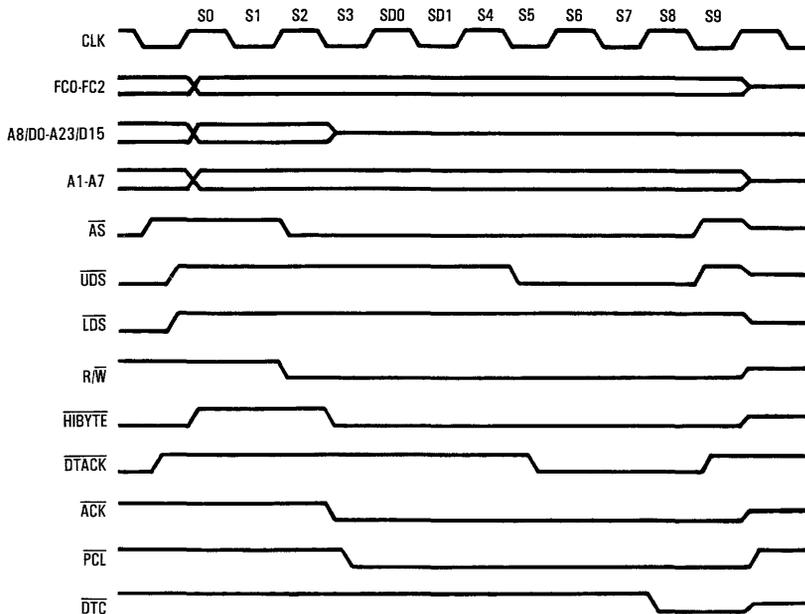
Figure 4-11. Single Address Write Cycle Timing Diagram

S2 $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and decoding may begin. $R/\overline{W}$ is asserted to indicate that data will flow to the memory from the device.

S3   The address/data pins are three stated and $\overline{ACK}$ is asserted to indicate that the device should place data on the bus. If the device port size is eight bits and the address is even, $\overline{HIBYTE}$ will be asserted to cause the device data on D0-D7 to be driven onto D8-D15 through an external buffer (in this case, $\overline{UDS}$ will be asserted and $\overline{LDS}$ will be negated). If this is the last transfer in a block operation, the $\overline{DONE}$ signal will be asserted by the DMAC.

If the protocol being used is a device with acknowledge only, then S4 will immediately follow S3. If protocol being used is device with acknowledge and ready, then the PCL pin will be sampled to determine when the DMAC should assert the data strobe(s). If PCL is low by the end of S3, then the DMAC will insert one delay cycle (SD0, SD1) and continue into S4. If PCL is high at the end of S3, then the DMAC will insert additional delay cycles (two delay states named SDw) after S3 until a ready signal is asserted and then proceed to SD0.

The DMAC also begins to sample the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins to determine when the cycle should be terminated. If $\overline{DTACK}$ is asserted by the end of S3, then the DMAC will 'skip' S6 and S7 and proceed directly from S5 to S8. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S3, one wait cycle will be inserted between S5 and S6. If $\overline{DTACK}$ is negated and $\overline{BEC0}$-$\overline{BEC2}$ are coded to normal at the end of S3, then the DMAC may execute S6 and S7 (see SD1 below).

SD0   This state is not executed for the device with acknowledge-only protocol. No signals change during SD0. The device is preparing to drive data onto the bus at this time and the DMAC is internally synchronizing PCL.

SD1   This state is not executed for the device with acknowledge-only protocol. No signals change during SD1, but the device continues to prepare to drive data onto the bus. The DMAC continues to synchronize the PCL signal. If $\overline{\text{DTACK}}$ is asserted by the end of SD1, then the DMAC will 'skip' S6 and S7 and proceed directly from S5 to S8. If $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are encoded with the bus error, retry, or relinquish and retry code by the end of SD1, one wait cycle will be inserted between S5 and S6. If $\overline{\text{DTACK}}$ is negated and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are coded to normal at the end of S3, then the DMAC will execute S6 and S7.

S4   No signals change during S4. Data from the device should be valid on the bus at the end of S4.

S5 $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ is asserted at this time to indicate that valid data is present on the bus and may be latched by the memory. If $\overline{\text{DTACK}}$ is asserted by the end of S5, then the DMAC may continue into S6 immediately. If $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S5, one wait cycle will be inserted between S5 and S6. If $\overline{\text{DTACK}}$ is negated and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ are coded to normal at the end of S5, then the DMAC will insert wait cycles after S5 until a termination signal is asserted and then proceed to S6.

S6   No signals change during S6. The DMAC is internally synchronizing the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins.

S7   No signals change during S7. The DMAC continues to synchronize the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins.

S8 $\overline{\text{DTC}}$ is asserted to indicate to the device that the bus cycle has been successfully terminated. $\overline{\text{DTC}}$ will not be asserted if a $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S9 $\overline{\text{AS}}$ is negated to indicate to the memory that the cycle has terminated. $\overline{\text{UDS}}$ and/or $\overline{\text{LDS}}$ is negated to disable the device to memory data buffers. The memory that was addressed during the cycle may then negate $\overline{\text{DTACK}}$ and/or $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$.

S9+1   This may be S0 of the following cycle, if the DMAC is ready to execute another bus cycle immediately. The FC0-FC2 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory location. If the DMAC is not ready to start a bus cycle, these lines will be three stated. R/$\overline{\text{W}}$ and $\overline{\text{HIBYTE}}$ are driven high to prepare the data bus for the next cycle. $\overline{\text{ACK}}$, $\overline{\text{DTC}}$, and $\overline{\text{DONE}}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed. The peripheral may respond by negating the ready signal at this time (for the device with acknowledge and ready protocol).

Note that although the basic timing for a single address write using the device with acknowledge and ready protocol is six clock cycles, if the ready and $\overline{\text{DTACK}}$ signals are asserted for one-half clock before $\overline{\text{ACK}}$ and the data strobes are asserted respectively, then the timing will be reduced to four clock cycles. There are also two timing combinations that give a five clock cycle transfer: one for an 'early' ready and

one for an 'early' $\overline{\text{DTACK}}$. In all cases, the system designer must be certain that data setup and hold times are met for the memory with respect to the data strobes.

If a channel is programmed for the device with acknowledge and ready protocol and one of the chaining modes, then the ready signal (PCL) for that channel must be asserted during accesses to the chain control table. This can be accomplished with a circuit such as the one shown in Figure 4-12, which asserts ready (PCLx) whenever $\overline{\text{ACK}}$ for this channel is negated and $\overline{\text{DTACK}}$ is asserted. Even though this circuit causes ready (PCLc) to be asserted during DMAC bus cycles executed for other channels, this will not cause improper behavior, since the PCLc input is ignored when other channels are active.
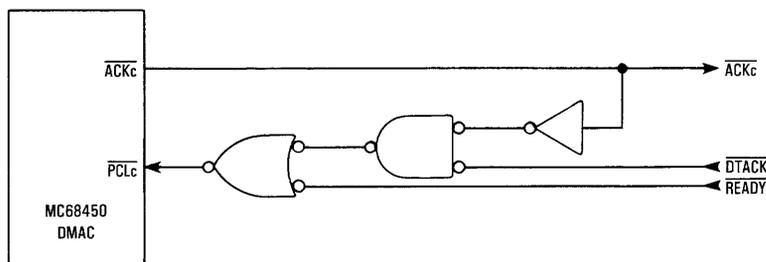
1-857

Figure 4-12. $\overline{\text{READY}}$ Circuit for Device with Acknowledge and Ready

Also note that when using the device with acknowledge and ready protocol, the assertion of $\overline{\text{DTACK}}$ should not preceed the assertion of the data strobe(s). This is due to the fact that if $\overline{\text{DTACK}}$ is asserted before ready (PCLc) is asserted, then the DMAC will terminate the cycle in the normal fashion, but the data strobe(s) will not be asserted. Thus, if the falling or rising edge of the data strobe(s) is used by memory to latch data, the data transfer will not execute properly.

4.3.2.3 HIGH-BYTE OPERATION. During single-address transfers between an 8-bit device and 16-bit memory, $\overline{\text{HIBYTE}}$ is used to control a bidirectional buffer as shown in Figure 2-2. $\overline{\text{HIBYTE}}$ is asserted when the memory address is even, in which case $\overline{\text{UDS}}$ will be asserted and $\overline{\text{LDS}}$ will be negated. R/$\overline{\text{W}}$ is used to determine the direction of the data transfer and the data flow through the external buffer. This method of operation allows the 8-bit device to be placed on D0-D7 so that it may properly return an interrupt vector number to an M68000 system processor, while preventing buffer contention on the data lines.

There are four possible cases for data flow during a single-address transfer between an 8-bit device and memory; even and odd read, and even and odd write cycles. Figure 4-13 shows the data flow for the four cases.
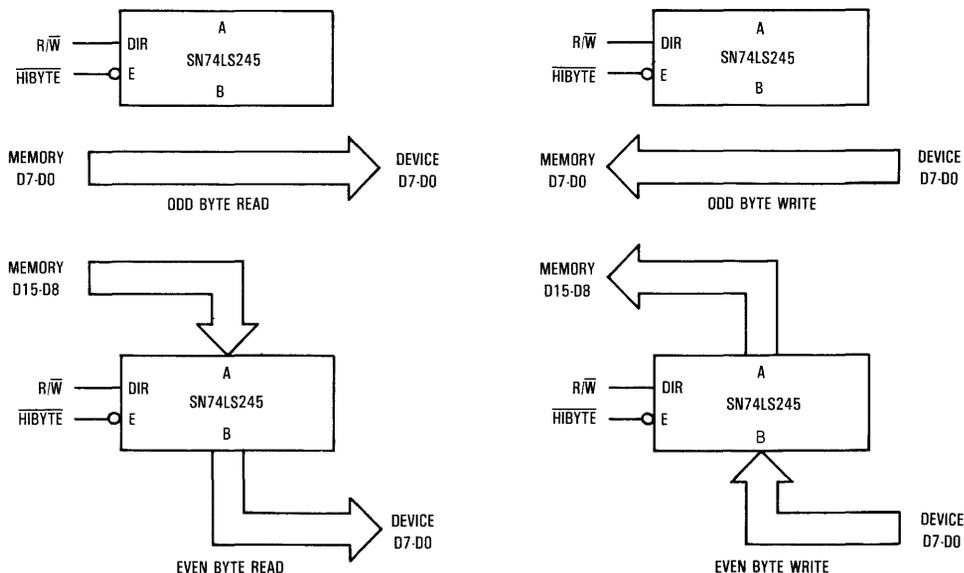
Figure 4-13. Implicitly Addressed 8-Bit Device with 16-Bit Memory Data Flow

4.4 BUS EXCEPTION CONTROL

In order to fully support the M68000 bus architecture, the DMAC uses three encoded inputs, $\overline{BEC0}$-$\overline{BEC2}$, to detect abnormal bus cycle termination conditions. These three lines function in a similar manner to the $\overline{RESET}$, $\overline{HALT}$, and $\overline{BERR}$ signals on an M68000 processor, but have different functionality than the processor counterparts. Table 4-1 shows the definition for each encoding of the $\overline{BEC0}$-$\overline{BEC2}$ pins and Figure 4-14 illustrates how the three inputs might be generated from signals normally present in an M68000 system.

Table 4-1. $\overline{BEC}$ Encoding Definitions

$\overline{BEC2}$	$\overline{BEC1}$	$\overline{BEC0}$	Definition
H	H	H	No Exception
H	H	L	Halt (and Release Bus)
H	L	H	Bus Error
H	L	L	Retry
L	H	H	Relinquish and Retry
L	H	L	(Undefined, Reserved)
L	L	H	(Undefined, Reserved)
L	L	L	Reset

1-859
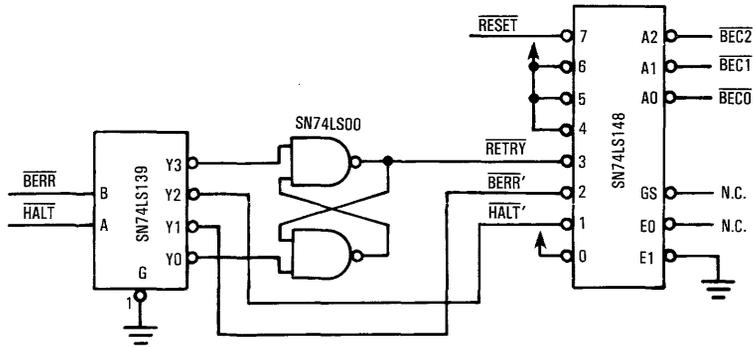
Figure 4-14. Example $\overline{\text{BEC}}$ Signal Generation Circuit

4.4.1 $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ Synchronization

The bus exception control inputs are synchronized in the same manner as all asynchronous inputs, but an additional debounce delay is included in order to assure valid operation in a system. The $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ pins are latched on each rising edge of CLK and are internally synchronized on the next rising edge of CLK. This synchronizer can be represented as a shift register as shown in Figure 4-15. On the $\overline{\text{BECn}}$ pins, an additional shift register stage is added to perform the debounce function. The internally synchronized signals $\overline{\text{BECn}}'$ and $\overline{\text{BECn}}''$ are continuously compared and if they are the same, then the level is accepted as valid and the DMAC may take action on it. If the two encodings are different, the DMAC will wait one more clock cycle and perform the comparison again. In this way, an SN74LS148 may be used to generate the $\overline{\text{BEC0}}$-$\overline{\text{BEC2}}$ inputs and transitional encodings will not cause erroneous operation. This debouncing also means that an encoding must be stable for at least two clock cycles to guarantee that it is latched at the same level on two successive rising clock edges.
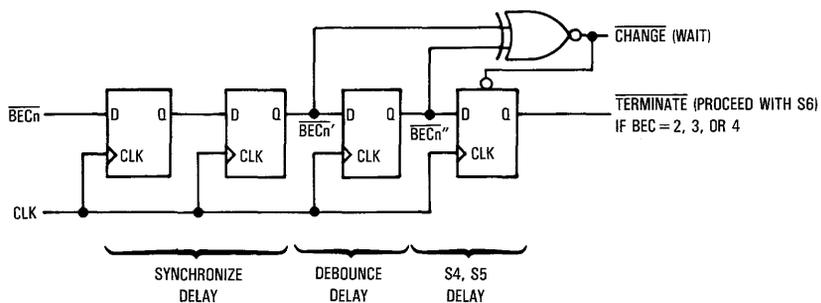
Figure 4-15. $\overline{\text{BEC}}$ Synchronizer Functional Diagram

As an example of the debouncer operation, consider the timing diagram in Figure 4-16. When the SN74LS148 outputs change from halt to bus error, they may transition through retry at the time when the DMAC is latching the $\overline{BEC0}$-$\overline{BEC2}$ pins. The internal debounce circuit will ignore the retry code and instead terminate the cycle with a bus error. Note that the delay from the first valid bus error, retry, or relinquish and retry encoding to when the bus cycle is ended is one synchronization delay (one to two clocks), plus one debounce delay (one clock), plus one bus controller delay (one clock) needed for the DMAC to switch from a normal termination to a bus exception type termination. This results in a bus cycle that is two clock cycles longer than if it had been terminated by a $\overline{DTACK}$ signal. Also, note that if $\overline{DTACK}$ and bus error, retry, or relinquish and retry are asserted at the same time and the asynchronous input setup time is met for the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ signals, then the bus cycle will be two clock cycles longer than if it had been terminated with $\overline{DTACK}$ alone and $\overline{DTACK}$ will be ignored. This behavior is different from what an M68000 processor will exhibit since the DMAC must suppress the assertion of $\overline{DTC}$ and prevent the internal address counters from incrementing if necessary.
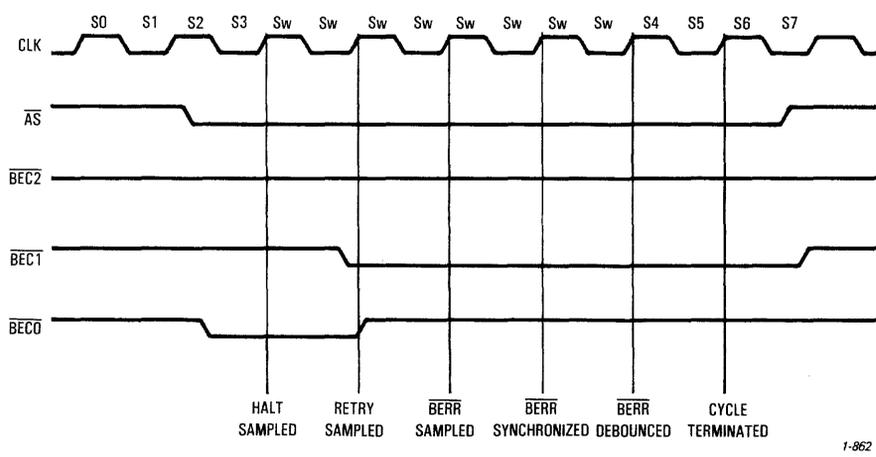
Figure 4-16. $\overline{BEC}$ Debouncer Timing Diagram

4.4.2 Bus Exception Control Functions

The three bus exception control signals allow eight different bus termination conditions to be signaled to the DMAC as shown in Table 4-1. The function of each encoding is described below. In describing the encoding of the $\overline{BEC0}$-$\overline{BEC2}$ pins, $\overline{BEC0}$ is the least significant bit and $\overline{BEC2}$ is the most significant bit of a binary coded digit from zero (0) to seven (7) with the encoding HHH equal to zero.

Whenever a non-zero value is encoded on the $\overline{BEC}$ inputs, the DMAC will enter the idle mode or DMA waiting mode as soon the current bus cycle is terminated and remain in that state until the $\overline{BEC}$ encoding returns to normal. Another system bus master may access the DMAC registers while it is in the idle mode, if it can gain mastership of the DMAC bus. In this manner, the DMAC can be halted by external hardware to enable the system processor to modify the DMAC registers and alter or monitor channel operations if necessary.

4.4.2.1 NORMAL TERMINATION. If a zero (0) is encoded on the $\overline{BEC}$ pins, the DMAC will operate in the normal mode and $\overline{DTACK}$ is used to terminate bus cycles.

4.4.2.2 HALT. If a one (1) is encoded on the $\overline{BEC}$ pins, the DMAC will be halted when the current bus cycle is terminated by the assertion of $\overline{DTACK}$. In order to preempt the bus cycle following the current cycle, the halt encoding must be valid one synchronization delay plus one debounce delay prior to when S0 of the next cycle would normally start. When the DMAC halts in response to the halt input, it will release ownership of the bus and enter the idle state until the $\overline{BEC}$ coding is returned to normal, at which time it will arbitrate for the bus and continue DMA operations if necessary. Figure 4-17 shows the timing of the halt operation. Note that if the halt encoding is asserted on the $\overline{BEC}$ pins at the same time that $\overline{DTACK}$ is asserted, the cycle will terminate two clock cycles later than if $\overline{DTACK}$ alone were asserted due to the debounce delay on the $\overline{BEC}$ inputs. Also, when the $\overline{BEC}$ pins are returned to normal coding, the DMAC will wait three clock cycles before asserting $\overline{BR}$ to re-arbitrate for the bus.

4.4.2.3 BUS ERROR. If a two (2) is encoded on the $\overline{BEC}$ pins, the DMAC will abort the current bus cycle and associated channel operation, and log an error in the channel status register and channel error register. When the $\overline{BEC}$ encoding is returned to normal, the DMAC will execute an internal error recovery sequence and then may relinquish ownership of the bus or start a bus cycle for the other channel if it has an operand transfer request pending. The error recovery sequence requires 25 clock cycles if the aborted cycle is a single address transfer or a dual address read, and 29 clock cycles if the aborted bus cycle is a dual address write. Figure 4-18 shows the timing of a bus error operation.

If a bus cycle is aborted with a bus error, $\overline{DTC}$ will not be asserted so that a peripheral device will not increment any operand counters. The DMAC will not increment or decrement the MAR, MTCR, or DAR.

4.4.2.4 RETRY. If a three (3) is encoded on the $\overline{BEC}$ pins, the DMAC will terminate the current bus cycle, enter the DMA waiting mode, and $\overline{BGACK}$ will remain asserted so that the DMAC retains ownership of the bus. When the $\overline{BEC}$ encoding is returned to normal, the DMAC will wait for three clocks and re-run the same bus cycle, using the same function code and address values. Figure 4-19 shows the functional timing of a retry operation.

If a bus cycle is terminated with a retry, $\overline{DTC}$ will not be asserted so that a peripheral device will not increment any operand counters. The DMAC will not increment or decrement the MAR, MTCR, or DAR and the internal data holding register will not be affected.

4.4.2.5 RELINQUISH AND RETRY. If a four (4) is encoded on the $\overline{BEC}$ pins, the DMAC will terminate the current bus cycle and relinquish ownership of the bus. When the $\overline{BEC}$ encoding is returned to normal, the DMAC will wait for three clocks, re-arbitrate for the bus, and re-run the same bus cycle using the same function code and address values. If an operand transfer request is internally asserted for a higher priority channel than the channel being retried before the DMAC regains control of the bus, then the retry and any remaining bus cycles required to complete the current operand transfer will be executed before the higher priority request will be serviced. Figure 4-20 shows the functional timing of a relinquish and retry operation.

If a bus cycle is terminated with a relinquish and retry, DTC will not be asserted so that a peripheral device will not increment any operand counters. The DMAC will not increment or decrement the MAR, MTCR, or DAR and the internal data holding register will not be affected.

CLK   S0   S1   S2   S3   Sw   Sw   S4   S5   S6   S7   S0   S1   S2   S3   S4

$\overline{BR}$

$\overline{BG}$

$\overline{BGACK}$

FC0-FC2

A8/D0-A23/D15

ADDRESS OUT   DATA IN   ADDRESS OUT

A1-A7

$\overline{AS}$

$\overline{UDS}/A0$

$\overline{LDS}/\overline{DS}$

$R/\overline{W}$

$\overline{DTACK}$

$\overline{HLT}$ *

$\overline{DTC}$

NORMAL CYCLE TERMINATION WITH HALT   RE-ARBITRATION DELAY FIVE CLOCKS MINIMUM**   NEXT BUS CYCLE
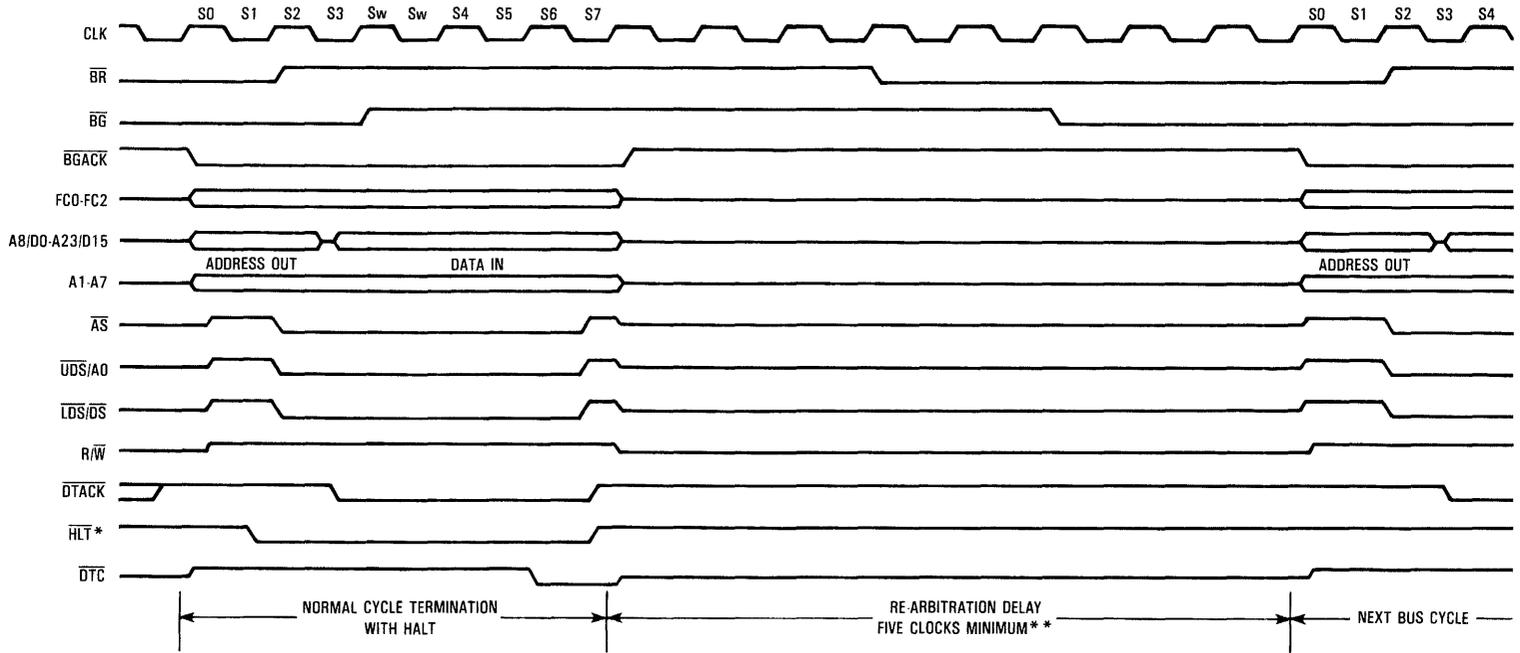
*$\overline{BEC2}$-$\overline{BEC0}$ encoded to signal either normal (high) or halt (low).

**This minimum delay will occur if $\overline{BG}$ is asserted prior to the clock edge on which $\overline{BR}$ is asserted.

1-863

Figure 4-17. Halt Operation Timing Diagram

CLK SO S1 S2 S3 Sw Sw Sw Sw S4 S5 S6 S7   SO S1 S2 S3

FCO-FC2

A8/DO-A23/D15

ADDRESS OUT   DATA IN

A1-A7

$\overline{AS}$

$\overline{UDS}$/A0

$\overline{LDS}$/$\overline{DS}$

R/$\overline{W}$

$\overline{DTACK}$

$\overline{BERR}$*

$\overline{DTC}$

ABORTED BUS CYCLE

25 OR 29 CLOCK DELAY

SERVICE OTHER CHANNEL (OR RELEASE BUS)
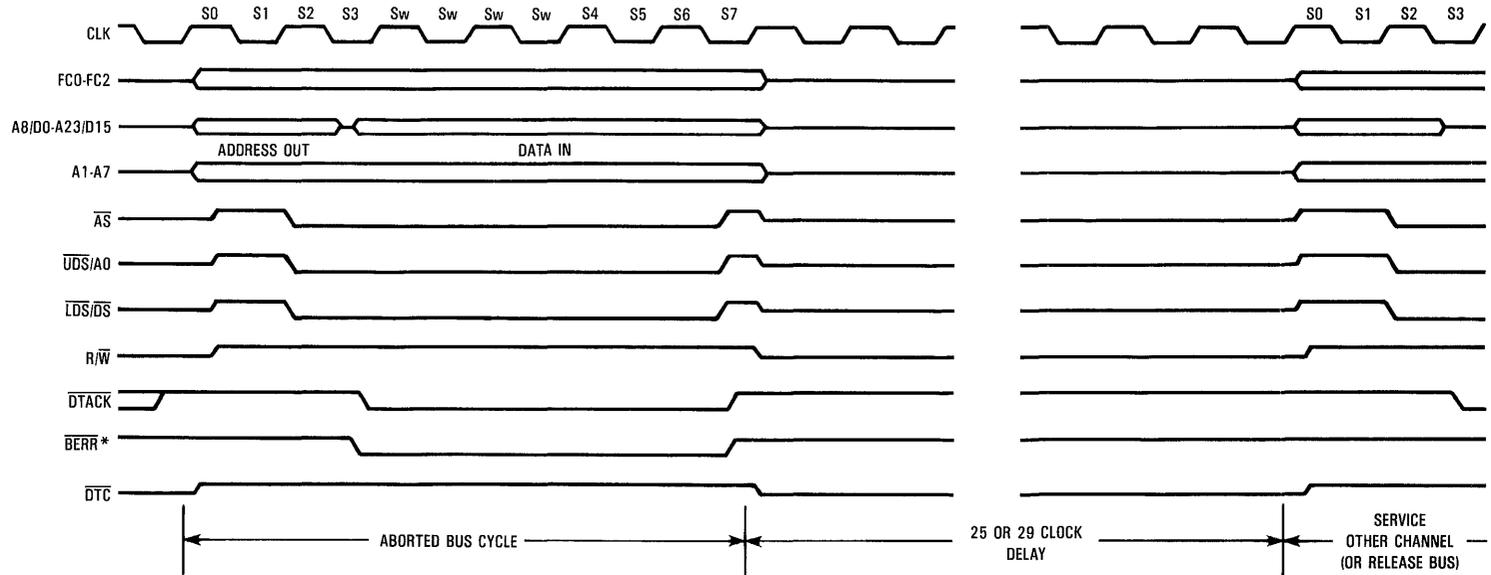
*$\overline{BEC2}$-$\overline{BEC0}$ encoded to signal either normal (high) or bus error (low).

1-864

Figure 4-18. Bus Error Operation
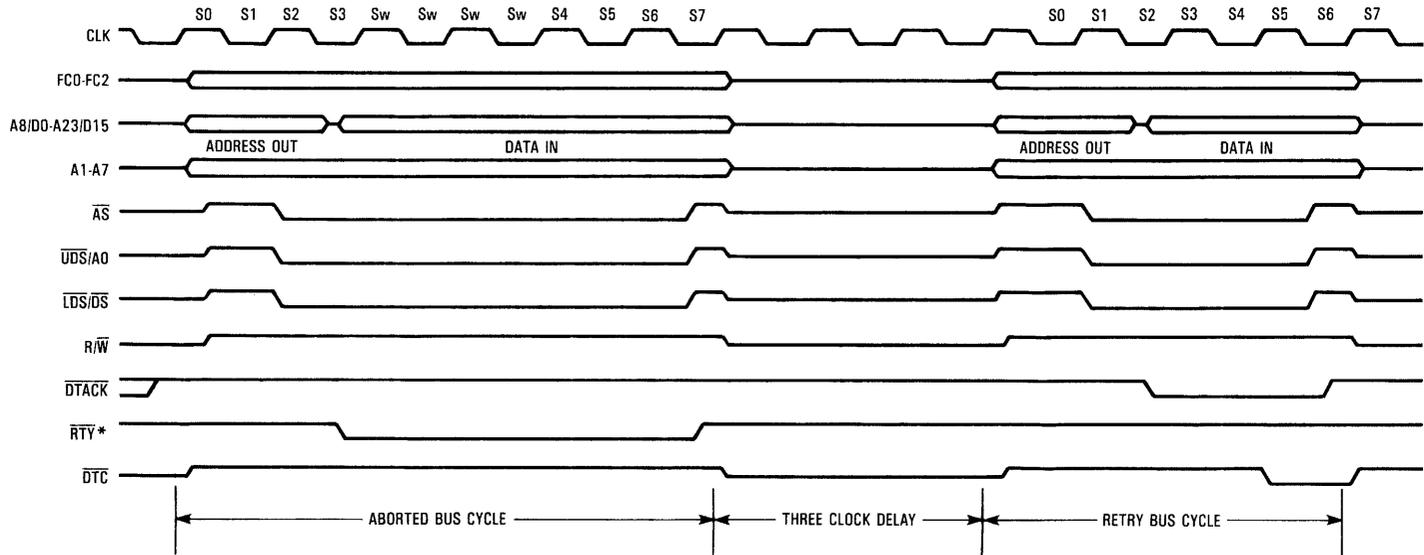
*$\overline{BEC2}$-$\overline{BEC0}$ encoded to signal either normal (high) or retry (low).

1-865

Figure 4-19. Retry Operation Timing Diagram
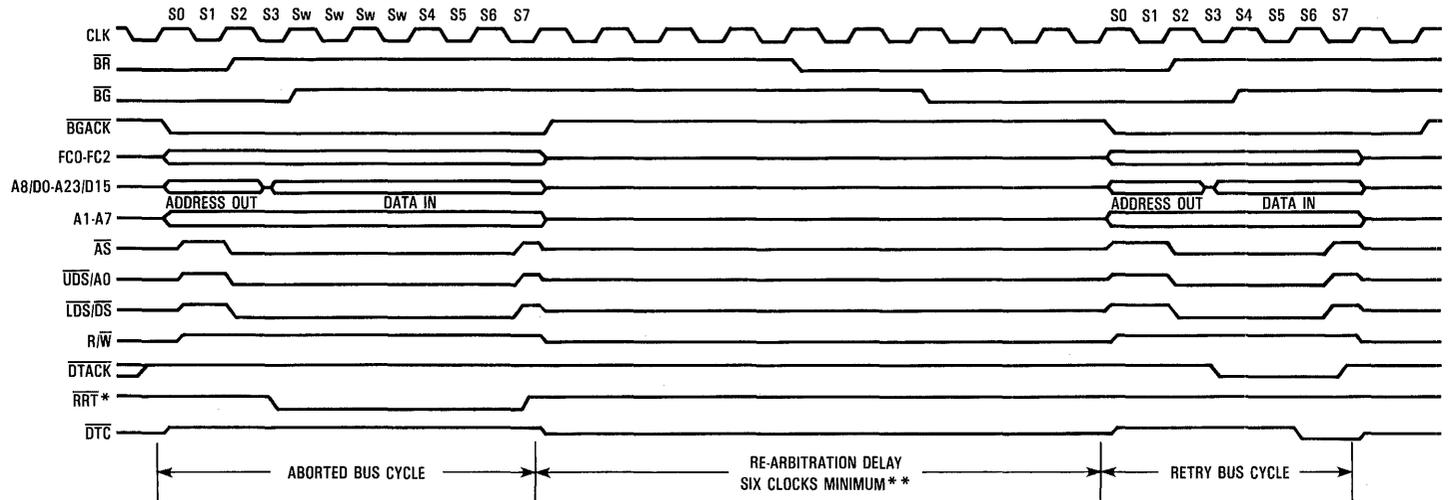
*$\overline{BEC2}$-$\overline{BEC0}$ encoded to signal either normal (high) or relinquish and retry (low).

**This minimum delay will occur if $\overline{BG}$ is asserted prior to the clock edge on which $\overline{BR}$ is asserted.

1-866

Figure 4-20. Relinquish and Retry Operation Timing Diagram

4.4.2.6 UNDEFINED $\overline{\text{BEC}}$ CODES. If a five (5) or six (6) is encoded on the $\overline{\text{BEC}}$ pins, the DMAC will synchronize and debounce the inputs as usual, but no action will be taken. One affect that these encodings will have is to lengthen a bus cycle until the $\overline{\text{BEC}}$ pins are returned to the normal encoding, regardless of the level on the $\overline{\text{DTACK}}$ pin. Furthermore, the DMAC will not continue or start to service any further operand transfer requests until one synchronization delay plus one debounce delay after the $\overline{\text{BEC}}$ pins are returned to the normal coding. Figure 4-21 shows the functional timing of an undefined $\overline{\text{BEC}}$ operation.
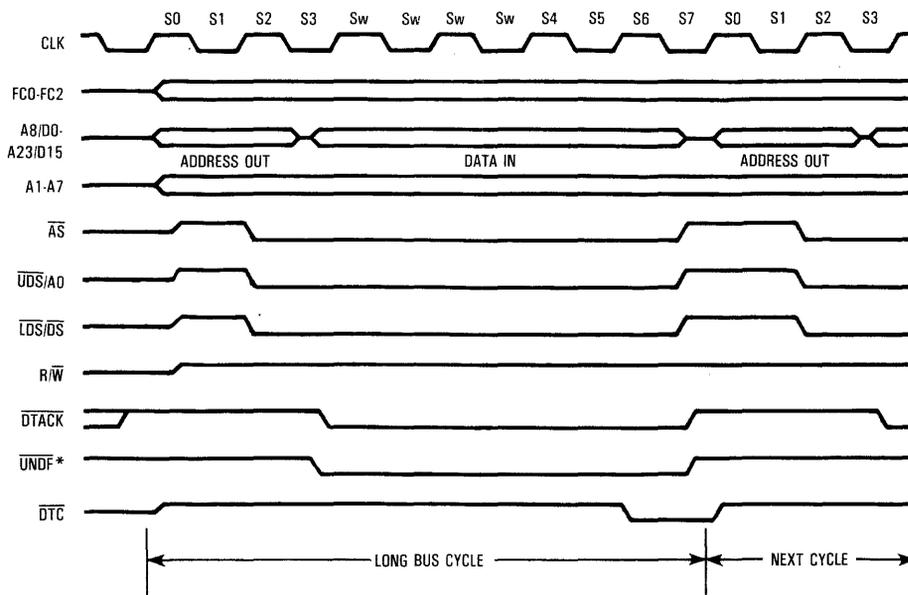
*$\overline{\text{BEC2}}$-$\overline{\text{BEC0}}$ encoded to signal either normal (high) or an undefined code (low).

1-867

Figure 4-21. Undefined $\overline{\text{BEC}}$ Operation Timing Diagram

4.4.2.7 RESET. If a seven (7) is encoded on the $\overline{\text{BEC}}$ pins, the DMAC will execute an internal reset sequence and enter the idle mode after one synchronization delay plus one debounce delay. Refer to **3.10 RESET OPERATION RESULTS** for a description of the effect that a reset operation has on the internal registers.

4.4.3 Bus Exception Control Summary

Figure 4-22 shows the bus exception control flow diagram for the DMAC in the idle and DMA modes of operation.

Figure 4-22. Bus Exception Control Flow Diagram
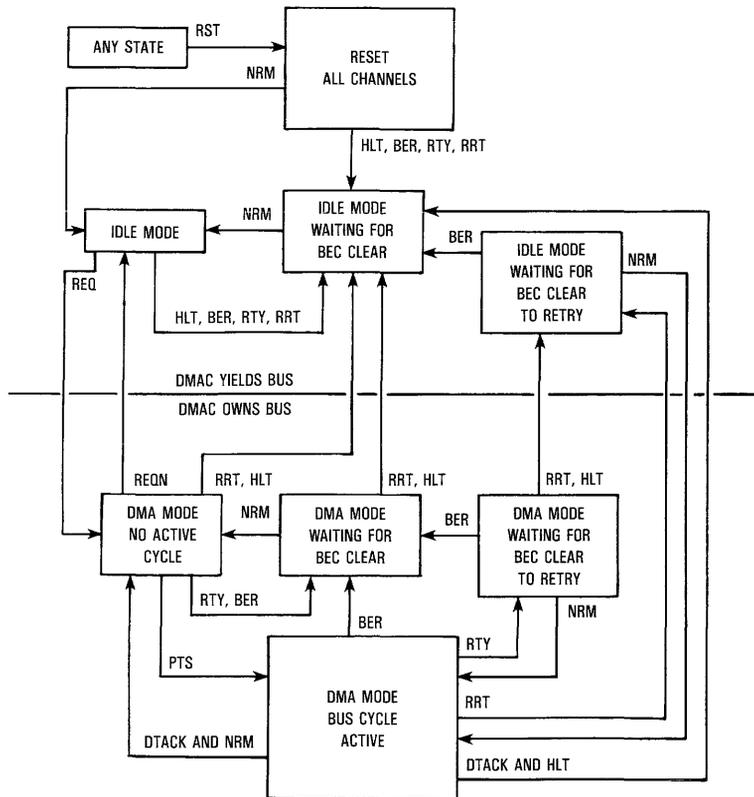
BEC Conditions:
NRM — Normal (No Exception)
HLT — Halt
BER — Bus Error
RTY — Retry
RRT — Relinquish and Retry
RST — Reset

Other Signals:
REQ — Channel Request
PTS — Permission to Start
DTACK — Data Transfer Acknowledge

4.5 BUS ARBITRATION

Once the system processor has initialized and started a DMAC channel and an operand transfer request has been made pending, either by an external device or the internal request generator, the DMAC will use the M68000 bus arbitration protocol to request bus mastership before entering the DMA mode of operation. The following paragraphs describe the arbitration protocol used by the DMAC and several special cases of operation relative to bus arbitration. Figure 4-23 shows a functional timing diagram of a typical bus arbitration sequence.

4.5.1 Requesting and Receiving the Bus

When an active channel has an operand transfer request pending, the DMAC will request bus mastership by asserting the $\overline{BR}$ signal. An external bus arbiter (either a separate unit such as the MC68452 Bus

Arbitration Module or the arbiter built into an M68000 processor) will then assert $\overline{BG}$ to indicate that bus mastership will belong to the DMAC as soon as the current bus master has released the bus. The DMAC then monitors the $\overline{AS}$ and $\overline{BGACK}$ signals to determine when it may assume mastership of the bus; these two signals must be negated to indicate that the previous bus cycle is complete and the previous bus master has released the bus. When this condition is met, the DMAC will assert $\overline{OWN}$ and then assert $\overline{BGACK}$ one-half clock cycle later to indicate that it has taken control of the bus. One clock after $\overline{BGACK}$ is asserted, $\overline{BR}$ will be negated to allow the external arbiter to begin arbitration for the next bus master. Once all operand transfer requests have been serviced, the DMAC will release control of the bus by negating $\overline{BGACK}$ and will then negate $\overline{OWN}$ one-half clock cycle later. Note that $\overline{OWN}$ may be used to control a bidirectional buffer for $\overline{BGACK}$ since it is asserted before and negated after $\overline{BGACK}$ is asserted and negated, respectively.

Note that if the DMAC is requesting the bus and $\overline{CS}$ or $\overline{IACK}$ is asserted, $\overline{BR}$ will be negated without waiting for the assertion of $\overline{BG}$. After the MPU cycle is completed, $\overline{BR}$ will be reasserted to continue the DMA operation.

4.5.2 Bus Overhead Time

In asynchronous bus systems such as those defined for the M68000 Family, a certain amount of time is used to synchronize incoming signals and is thus 'wasted', since no data transfer activity can take place during those periods. In many applications, the synchronization overhead time required to switch bus masters and channels within the DMAC must be known to predict system behavior. The following paragraphs describe three types of overhead periods: those associated with the DMAC taking control of the bus, those in between successive operand transfers, and those that occur when the DMAC is releasing control of the bus to another bus master. In all of these discussions, the system is assumed to consist of a single M68000 processor and a single DMAC using the same clock so that all arbitration delays are controlled by those two devices and can be predicted, as shown in Figure 4-23.

4.5.2.1 FRONT-END OVERHEAD. This is the delay that will occur from the time that the MPU terminates a bus cycle by negating $\overline{AS}$ (in S7 of the MPU cycle) to when the DMAC starts a bus cycle by placing function code and address information on the bus (in S0 of the DMA cycle). It is assumed that $\overline{BG}$ is asserted and $\overline{BGACK}$ is negated prior to the negation of $\overline{AS}$ by the MPU so that no additional synchronization delays are introduced by those signals.

The amount of idle time between a bus cycle executed by another bus master and the start of a DMA cycle is dependent on the point at which the DMAC receives a request in relation to the bus cycle being executed by the other bus master (the MPU in this case). The minimum time for the recognition of a request from a device to when the DMAC will start a bus cycle is 12 clock cycles. A portion of these 12 clock cycles will normally overlap with the execution of a bus cycle by another bus master, as shown in Figure 4-23. This figure shows the best case overlap, which results in a front-end overhead of five clocks. Figure 4-24 shows the worst case overlap, which results in an overhead of eight clocks. The difference in the two cases is due to the fact that for the best case, $\overline{BR}$ is recognized, by the MPU, as asserted during S6 of a bus cycle, which is too late to preempt the next MPU cycle; thus the next bus cycle overlaps the DMAC bus arbitration sequence. For the worst case, the $\overline{BR}$ assertion is recognized by the MPU during S4 of a bus cycle, which is early enough to preempt the next MPU bus cycle, and only one and one-half clocks of the current MPU bus cycle overlap with the bus arbitration.
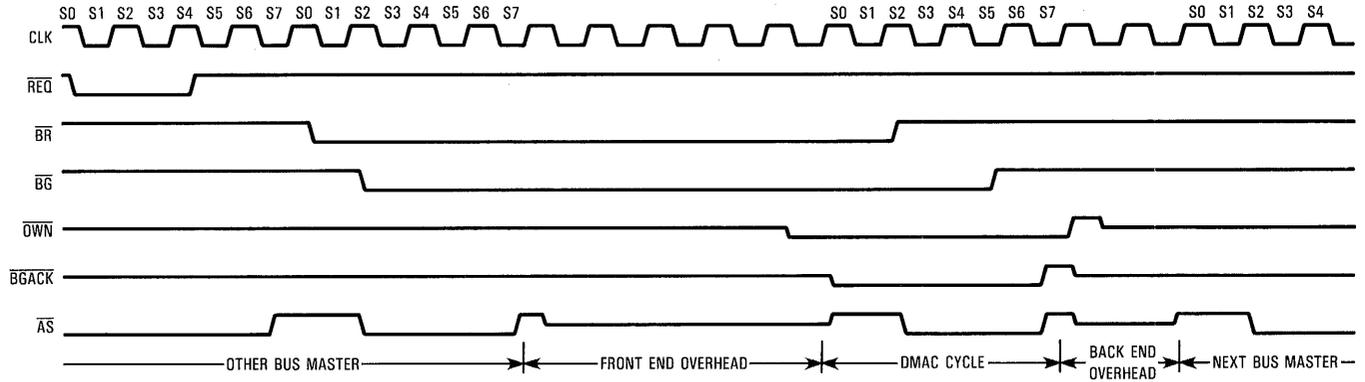
Figure 4-23. Bus Arbitration Timing Diagram (Best Case)
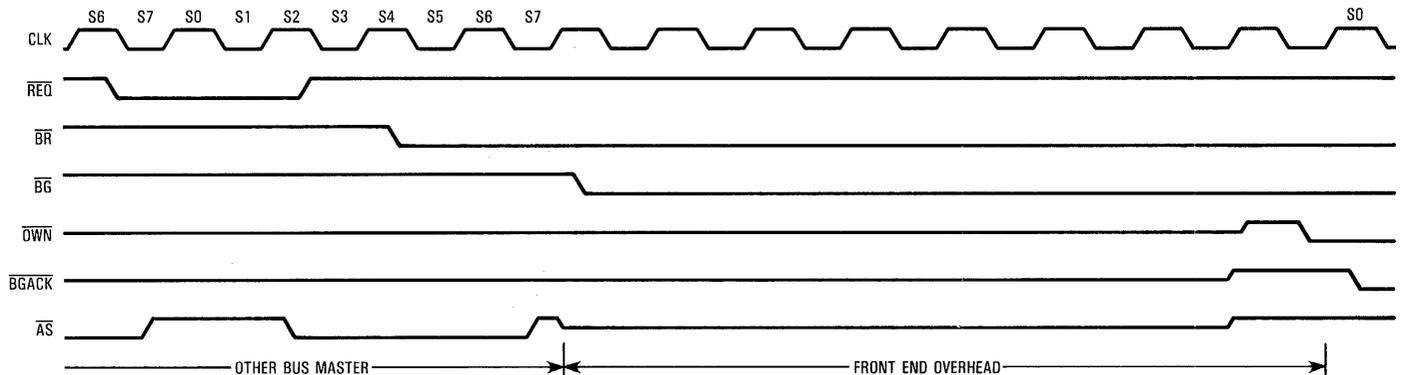
Figure 4-24. Front End Overhead (Worst Case)

4.5.2.2 BACK-END OVERHEAD. This overhead time is the delay between when the DMAC has completed all pending operand transfers and releases the bus, plus the synchronization of $\overline{BGACK}$ negated by the MPU. As the DMAC completes the last pending bus cycle in S7 for a DMA read or S9 for a DMA write, it may negate $\overline{BGACK}$ at the same time as $\overline{AS}$. The MPU will then wait one synchronization delay plus one-half clock cycle before S0 of the next MPU cycle. Thus, the best case back-end overhead time is two clock cycles, and the worst case is determined by the mode of operation as described below.

4.5.2.3 INTER-CYCLE OVERHEAD. This is the overhead delay that will occur between successive DMA bus cycles for any of the channels, or at the end of an operand transfer, while the DMAC maintains ownership of the bus.

Figure 4-25 shows the timing, in clock cycles, for various operations that the DMAC may perform while it owns the bus (i.e., when $\overline{BGACK}$ is asserted). The operations described include block transfer initiation and termination times for chained operations, all types of operand transfers, and channel shut down sequences. Each operation described by one box in this figure is independent of the number of channels that are active, and there is no overlap of execution between channels (i.e., idle termination time for one channel will not be concurrent with the idle initiation time for another channel). To use this table, select all of the individual operations that will be executed during one period of bus ownership (referred to as a burst period) and add the total number of clocks for each operation to get the total burst period time in clocks. Also, the total number of read and write cycles and the total number of overhead clocks can be calculated. In addition, the appropriate front-end and back-end overhead must also be included in the calculation.

In addition to the overhead times that may be introduced during the operations described in Figure 4-25, the timing of a cycle steal request assertion can also affect bus overhead times. Figure 4-26 indicates the number of idle clocks that will occur between the current operation (which is for a channel using cycle steal requests) and the start of the next operation (for any channel), depending on when a new request is asserted by a device. The intervals shown in this figure are in relation to the CLK signal, such that the start or end of an interval is slightly before a rising clock edge. With this assumption, an asserted edge of $\overline{REQ}c$ is guaranteed to meet the asynchronous input setup time to a rising edge of CLK within that interval and the behavior of the DMAC can be predicted. If $\overline{REQ}$ is asserted in a given interval shown in Figure 4-26, the number of idle clock cycles labeled in that interval will occur before the next operation. If $\overline{REQ}$ is asserted during an interval labeled "Ignore Requests", meets the two clock minimum pulse width, and is negated before the end of that interval, then the channel will not recognize the request. However, if $\overline{REQ}$ is asserted during such an interval and remains asserted for two clock cycles into the next interval, then a new request will be recognized.

**THE FOLLOWING NOTES APPLY TO
FIGURE 4-25 (SHEETS 1 THROUGH 7)
WHICH APPEARS ON THE PAGES 4-31 THROUGH 4-37**

NOTES
1. Each tic mark on a line represents the end of a two-clock period.
2. All read cycles are assumed to take four clocks and all write cycles are assumed to take five clocks. In a system that does not meet these assumptions, the number of additional clock cycles for each bus cycle is added to the total to obtain the actual operation timing.
3. Except where noted, the request generation method does not affect operation timing.
4. The only operation that will follow operations marked with an asterisk (*) will be the appropriate termination operation.
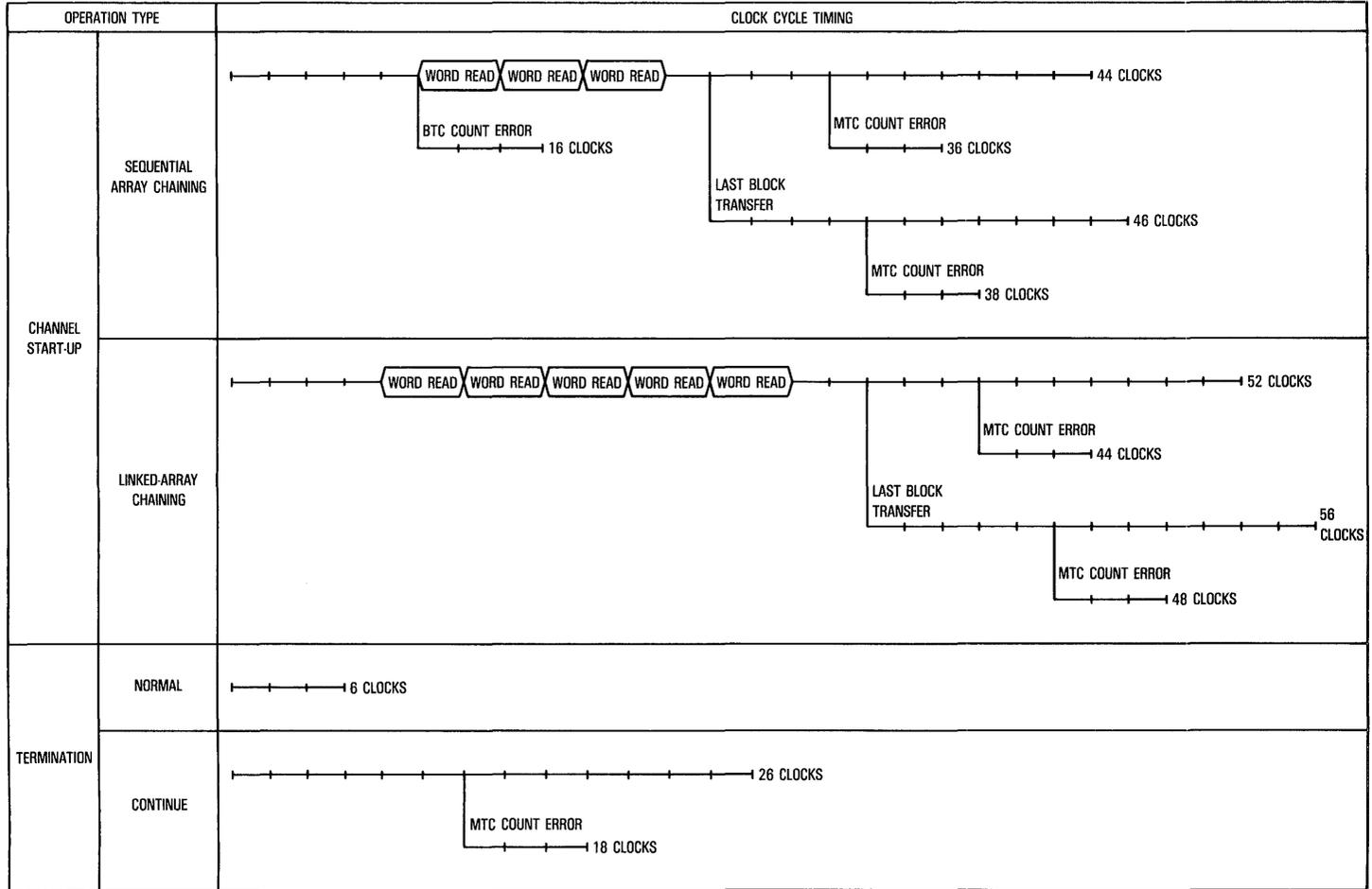
Figure 4-25. DMA Operation Timing Table (Sheet 1 of 7)

1-871

OPERATION TYPE		CLOCK CYCLE TIMING
TERMINATION (CONTINUED)	SEQUENTIAL ARRAY CHAINING	WORD READ ⟩ WORD READ ⟩ WORD READ — 36 CLOCKS MTC COUNT ERROR — 28 CLOCKS LAST BLOCK TRANSFER — 38 CLOCKS MTC COUNT ERROR — 30 CLOCKS
	LINKED-ARRAY CHAINING	WORD READ ⟩ WORD READ ⟩ WORD READ ⟩ WORD READ ⟩ WORD READ — 46 CLOCKS MTC COUNT ERROR — 38 CLOCKS LAST BLOCK TRANSFER — 50 CLOCKS MTC COUNT ERROR — 42 CLOCKS
SINGLE ADDRESS TRANSFERS	M → D	W/B READ ⟩ 4 CLOCKS
	D → M	W/B WRITE ⟩ 5 CLOCKS

Figure 4-25. DMA Operation Timing Table (Sheet 2 of 7)

1-871a

OPERATION TYPE		CLOCK CYCLE TIMING
DUAL ADDRESS TRANSFERS OPERAND SIZE IS BYTE DEVICE SIZE IS 8-BITS	M → D WITH BYTE PACKING MEMORY ADDRESS REGISTER COUNTS AND IS EVEN	**FIFO EMPTY, MTC ≥ 2** WORD READ / BYTE WRITE — 17 CLOCKS **MTC = 1** BYTE READ / BYTE WRITE — 13 CLOCKS* **ONE BYTE IN FIFO** **MTC ≥ 2** BYTE WRITE — 13 CLOCKS **MTC = 1** BYTE WRITE — 9 CLOCKS*
	M → D WITH BYTE PACKING MEMORY ADDRESS REGISTER DOES NOT COUNT OR IS ODD	**SOURCE ADDRESS IS ODD** BYTE READ / BYTE WRITE — 17 CLOCKS **SOURCE ADDRESS IS EVEN** BYTE READ / BYTE WRITE — 15 CLOCKS **MTC = 1** BYTE READ / BYTE WRITE — 11 CLOCKS*
	D → M WITH BYTE PACKING MEMORY ADDRESS REGISTER COUNTS AND IS EVEN	**MTC ≥ 3** BYTE READ / **TWO BYTES IN FIFO** WORD WRITE — 19 CLOCKS **ONE BYTE IN FIFO** — 14 CLOCKS **DONE INPUT ASSERTED** **TWO BYTES IN FIFO** WORD WRITE — 19 CLOCKS **ONE BYTE IN FIFO** BYTE WRITE — 19 CLOCKS **MTC ≤ 2** BYTE READ / BYTE WRITE — 15 CLOCKS **MTC = 1** BYTE READ / BYTE WRITE — 11 CLOCKS*

Figure 4-25. DMA Operation Timing Table (Sheet 3 of 7)

1-871b

OPERATION TYPE		CLOCK CYCLE TIMING
DUAL ADDRESS TRANSFERS OPERAND SIZE IS BYTE DEVICE SIZE IS 8-BITS (CONTINUED)	D → M WITH BYTE PACKING MEMORY ADDRESS REGISTER DOES NOT COUNT OR IS ODD	MTC ≥ 3 BYTE READ ╳ BYTE WRITE — MAR IS ODD — 19 CLOCKS MAR DOES NOT COUNT AND IS EVEN — 17 CLOCKS MTC ≤ 2 BYTE READ ╳ BYTE WRITE — 15 CLOCKS MTC = 1 BYTE READ ╳ BYTE WRITE ⟩ 11 CLOCKS*
DUAL ADDRESS TRANSFERS OPERAND SIZE IS BYTE DEVICE SIZE IS 16-BITS	M → D	MTC ≠ 0 WORD READ ╳ BYTE WRITE ╳ BYTE WRITE — 18 CLOCKS MTC → 0 WRITE ⟩ 14 CLOCKS
	D → M	BYTE READ ╳ BYTE READ ╳ WORD WRITE ⟩ 13 CLOCKS
DUAL ADDRESS TRANSFERS OPERAND SIZE IS WORD DEVICE SIZE IS 16-BITS	D → M OR M → D	MTC ≠ 0 WORD READ ╳ WORD WRITE ⟩ 9 CLOCKS MTC → 0 WRITE ⟩ 9 CLOCKS IDENTICAL OPERATION TIMING IS ACHIEVED FOR AN OPERAND SIZE OF "BYTE, NO PACKING", IN WHICH CASE BYTE READ AND WRITE CYCLES ARE USED.
DUAL ADDRESS TRANSFERS OPERAND SIZE IS LONG DEVICE SIZE IS 8-BITS	M → D	MTC ≠ 0 WORD READ ╳ BYTE WRITE ╳ BYTE WRITE ╳ WORD READ ╳ BYTE WRITE ╳ BYTE WRITE — 32 CLOCKS MTC → 0 WRITE ⟩ 28 CLOCKS
	D → M	BYTE READ ╳ BYTE READ ╳ WORD WRITE ╳ BYTE READ ╳ BYTE READ ╳ WORD WRITE ⟩ 26 CLOCKS

Figure 4-25. DMA Operation Timing Table (Sheet 4 of 7)

1-871c

OPERATION TYPE		CLOCK CYCLE TIMING

Figure 4-25. DMA Operation Timing Table (Sheet 5 of 7)

1-871d

OPERATION TYPE		CLOCK CYCLE TIMING

Figure 4-25. DMA Operation Timing Table (Sheet 6 of 7)

1-871e

OPERATION TYPE		CLOCK CYCLE TIMING
DUAL ADDRESS TRANSFERS WITH BYTE PACKING OPERAND SIZE IS BYTE DEVICE SIZE IS 8-BITS LIMITED OR MAXIMUM RATE AUTO REQUEST (CONTINUED)	D →M OR M→ D SOURCE ADDRESS REGISTER DOES NOT COUNT AND IS EVEN, OR IS ODD (CONTINUED)	

NOTES:
1. Each tic mark on a line represents the end of a two-clock period.
2. All read cycles are assumed to take four clocks and all write cycles are assumed to take five clocks. In a system that does not meet these assumptions, the number of additional clock cycles for each bus cycle is added to the total to obtain the actual operation timing.
3. Except where noted, the request generation method does not affect operation timing.
4. The only operation that will follow operations marked with an asterisk (*) will be the appropriate termination operation.

1-871f

Figure 4-25. DMA Operation Timing Table (Sheet 7 of 7)

Figure 4-26. Inter-Cycle Overhead Timing Diagram (Sheet 1 of 5)
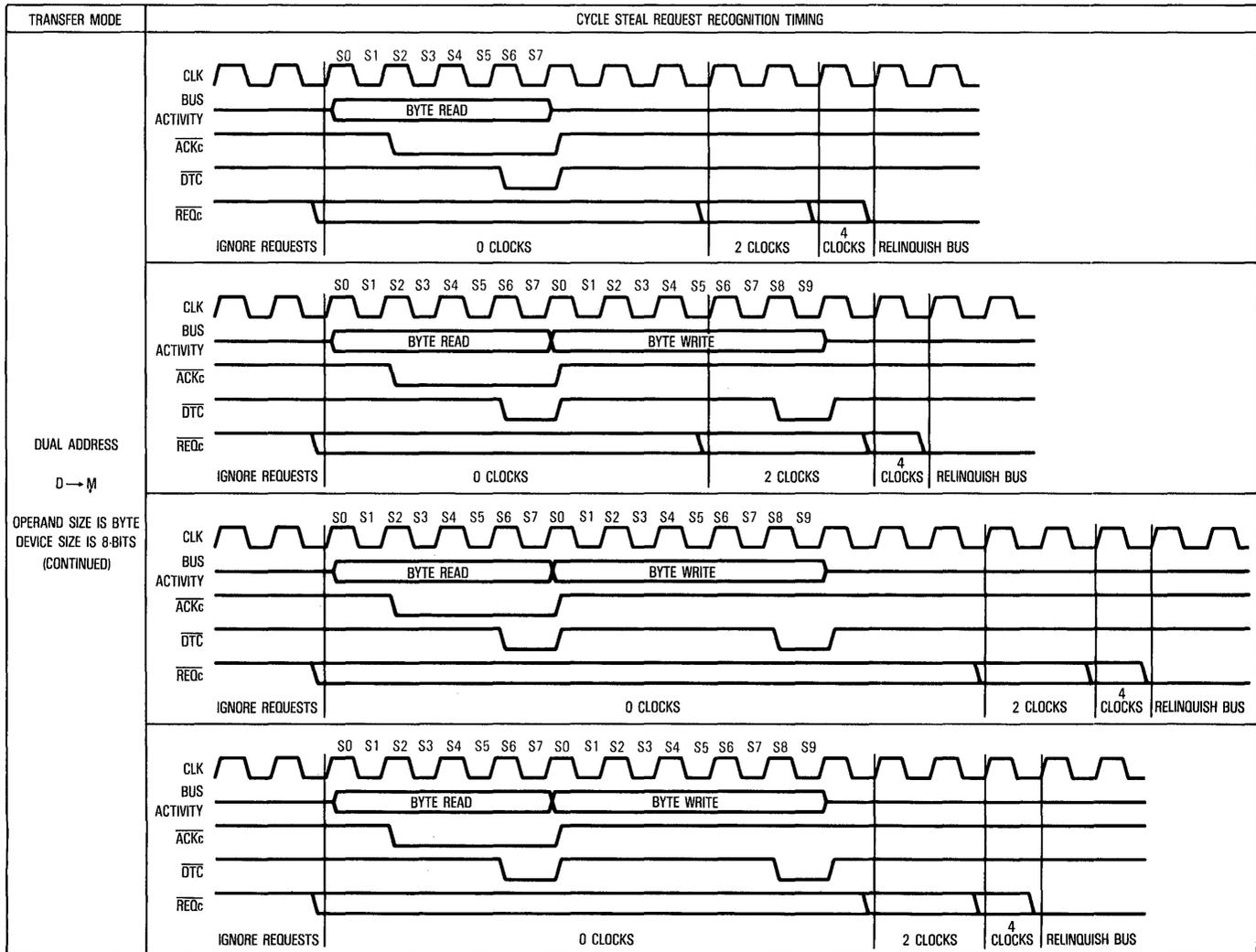
1-872

Figure 4-26. Inter-Cycle Overhead Timing Diagram (Sheet 2 of 5)

1-872a

Figure 4-26. Inter-Cycle Overhead Timing Diagram (Sheet 3 of 5)

1-872b

TRANSFER MODE	

DUAL ADDRESS

M → D

OPERAND SIZE IS LONG
DEVICE SIZE IS 8-BITS

CLK · BUS ACTIVITY · ACKc · DTC · REQc

WORD READ · BYTE WRITE · BYTE READ

IGNORE REQUESTS THIS PERIOD IS 23 CLOCKS LONG AND INCLUDES TWO READ CYCLES FROM MEMORY AND THREE WRITE CYCLES TO THE DEVICE.

0 CLOCKS · 2 CLOCKS · 4 CLOCKS · RELINQUISH BUS

DUAL ADDRESS

D → M

OPERAND SIZE IS LONG
DEVICE SIZE IS 8-BITS

BYTE READ · BYTE READ · WORD WRITE

IGNORE REQUESTS — SEE NOTE · 0 CLOCKS · 2 CLOCKS · 4 CLOCKS · RELINQUISH BUS

DUAL ADDRESS

M → D

OPERAND SIZE IS LONG
DEVICE SIZE IS 16-BITS

WORD READ · WORD WRITE · WORD READ · WORD WRITE

IGNORE REQUESTS · 2 CLOCKS · 4 CLOCKS · RELINQUISH BUS

NOTE: This period is 19 clocks long and includes three read cycles from the device and one write cycle to memory.

1-872c

Figure 4-26. Inter-Cycle Overhead Timing Diagram (Sheet 4 of 5)
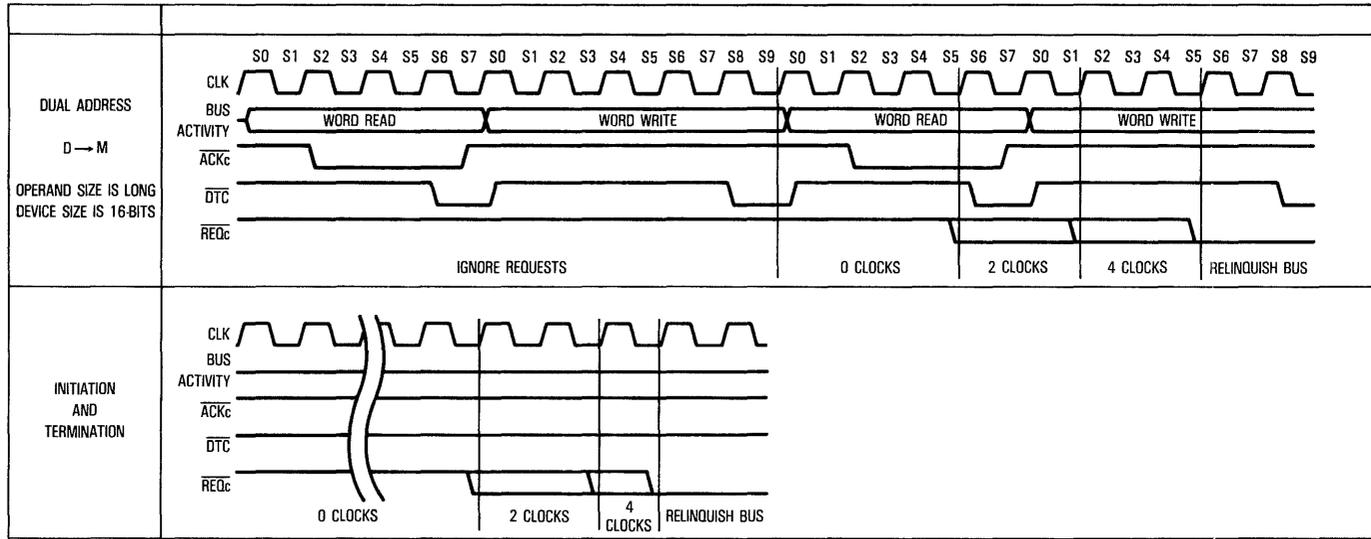
Figure 4-26. Inter-Cycle Overhead Timing Diagram (Sheet 5 of 5)

4.6 ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION

Because the M68000 Family of microprocessors uses an asynchronous bus protocol, precise timing considerations are not necessary in order for system components to communicate properly. By using such an asynchronous bus interface, peripherals in the system do not need to have any knowledge of the clock(s) used by bus masters in the system. However, even though the external system bus is asynchronous, any bus master that uses the bus is synchronous internally and must synchronize incoming handshake signals. If it is desired to insure exact timing relationships between a bus master and a peripheral during a bus cycle, it is necessary for the peripheral interface to use the same clock that the bus master uses. In this case, the points at which the bus master asserts outputs and recognizes the assertion of inputs will be precisely known. To a bus master, the type of peripheral interface used is not important, since it will always operate in the same manner; but the design requirements of a peripheral subsystem and the performance of the overall system can be affected by the type of interface that is used.

For a bus master such as the DMAC, there can be two distinct asynchronous or synchronous interfaces operating simultaneously. One is the M68000 Family bus interface to memory or an explicitly addressed peripheral. The second is the device control interface to an implicitly addressed peripheral using the device with acknowledge and ready protocol. An implicitly addressed peripheral using the device with acknowledge-only protocol must always operate in synchronization with the DMAC.

4.6.1 Asynchronous Operation

To achieve clock signal independence at a system level, the DMAC can be used in an asynchronous manner. This entails using only the bus handshake lines ($\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, and $\overline{BEC0}$-$\overline{BEC2}$) and, as needed, the device control lines ($\overline{REQc}$, $\overline{ACKc}$, $\overline{PCLc}$, $\overline{DTC}$, and $\overline{DONE}$). Using this method, $\overline{AS}$ (and $\overline{ACKc}$ during device accesses) signals the start of a bus cycle and the data strobes (possibly delayed until ready is asserted for single address writes) are used as a condition for valid data on a write cycle. The slave device(s) then responds by placing the requested data on the data bus or latching data from the data bus and asserting $\overline{DTACK}$ (and possibly read for single address reads). The DMAC then acknowledges the successful termination of the bus cycle by asserting $\overline{DTC}$ for one clock cycle and negating $\overline{AS}$, $\overline{ACKc}$, and the data strobes. If no slave responds or an access is invalid, external control logic uses the bus exception control signals to abort or re-run the bus cycle.

The $\overline{DTACK}$ signal is allowed to be asserted before the data from a slave device is valid on a read cycle (either single or dual address). The length of time that $\overline{DTACK}$ may precede data during a dual address read is given as parameter #8 and it must be met in any asynchronous system to insure that valid data is latched into the processor. This delay is allowed due to the manner in which the DMAC synchronizes the $\overline{DTACK}$ signal before asserting $\overline{DTC}$ and latching data. The length of time that $\overline{DTACK}$ may precede data during a single address read is dependent on the system design and manner in which the peripheral device latches the read data. In a like manner, the ready signal (PCLc) can be asserted during a single address read before the peripheral will latch data.

The ready signal ($\overline{PCLc}$) is allowed to be asserted before the data from a peripheral device is valid on a single address write cycle. The length of time that ready may precede data is dependent on the system design such that data will be valid when the memory system latches it. In order to operate in the same manner as an M68000 Family processor, data must be valid prior to the assertion of the data strobe(s).

If a bus exception control condition other than normal is used to terminate a bus cycle, the $\overline{BEC0}$-$\overline{BEC2}$ levels must be valid for specification #96 prior to the assertion of $\overline{DTACK}$ in order for the normal termination to be preempted and the exception action taken. Since a peripheral system will normally generate a single signal that is used by the DMAC interface logic to generate the $\overline{BEC0}$-$\overline{BEC2}$ signals, the delay from the

assertion of the signal by the peripheral to the assertion of the proper $\overline{\text{BEC}}$ encoding must be accounted for in order to insure that the setup time to the assertion of $\overline{\text{DTACK}}$ is met.

4.6.2 Synchronous Operation

If it is desired to precisely determine the behavior of a system, the relationship between a bus master's clock signal and asynchronous input signals to that bus master must be known. Thus, the system clock may be used in addition to the bus master control signals to generate pseudo synchronous bus slave handshake signals. In order to allow the system designer to predict when the DMAC will recognize the assertion or negation of input signals, the asynchronous input setup time is given as parameter #6. If an asynchronous input signal makes a transition from one level to the other and is stable for the setup time prior to a rising edge of the clock, that level is guaranteed to be recognized and is valid internally on the next rising edge of the clock. However, the converse is not true — if the input signal changes states and is not stable for the full setup time, the new level is not guaranteed not to be recognized. This means that if a signal is asserted (or negated) inside the setup time window prior to the rising edge of the clock, it may be recognized as negated or asserted by the DMAC, depending on several factors such as individual device characteristics, clock waveform and frequency, ambient temperature, supply voltage, and system age.

In order to allow the DMAC to execute minimum length bus cycles while giving the longest possible memory or peripheral access times, $\overline{\text{DTACK}}$ and data should be valid in precise relation to the DMAC clock. If $\overline{\text{DTACK}}$ is asserted prior to the setup time from the rising edge of S3 for a dual address read cycle, the DMAC will latch the value of the data bus on the falling edge of S6. Thus, if the $\overline{\text{DTACK}}$ and data setup times are both guaranteed to be met by the system, then specification #91 may be exceeded.

For single address read cycles, a peripheral device should not latch data before $\overline{\text{DTC}}$ is asserted or after the data strobes are negated and thus, might use the falling edge of S6 as a latch trigger. Additionally, if the peripheral asserts ready prior to the setup time for the rising edge of S3 and $\overline{\text{DTACK}}$ is also asserted by that time, then the falling edge of S6 will occur one and one-half clock cycles later. If it is not known whether or not $\overline{\text{DTACK}}$ will meet the setup time to the end of S3, then the peripheral should wait until $\overline{\text{DTACK}}$ has been asserted for at least one clock before asserting ready prior to the setup time for the next rising edge of the clock (which will be a wait cycle) so that the rising edge of S6 can be correctly predicted.

During single address write cycles from a device with acknowledge and ready, the device may assert ready ($\overline{\text{PCLc}}$) before valid data is on the bus. If ready is asserted prior to the setup time to the rising edge of the clock (either S3 or SDw), then the DMAC will assert the data strobe(s) one and one-half clocks later.

If an implicitly addressed peripheral is used with the device with acknowledge-only protocol, it may operate in a pseudo asynchronous manner; however, the timing constraints placed on the device will be closely related to the DMAC clock. During a read cycle, the device must be prepared to accept input data when $\overline{\text{DTC}}$ is asserted and cannot require a data hold time past the negation of the data strobe(s) one-half clock later. During a write cycle, the device must be able to place data onto the bus when the DMAC asserts $\overline{\text{ACKc}}$, with some setup time prior to the data strobe(s) assertion one clock later.

If a $\overline{\text{BEC}}$ encoding other than normal is used to terminate a bus cycle in which $\overline{\text{DTACK}}$ is also asserted, then the $\overline{\text{BEC}}$ encoding must be recognized as valid on the same clock edge that $\overline{\text{DTACK}}$ is first recognized as asserted. Thus, if $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}\text{-}\overline{\text{BEC2}}$ are all valid prior to the setup time from the same rising edge of the clock (S3 for a minimum length cycle) the normal termination is guaranteed to be preempted, and the exceptional action is taken.

SECTION 5
OPERATIONAL DESCRIPTION

This section describes the programmable channel functions available on the DMAC, the manner in which data transfer operations are executed, and behavior under exceptional conditions. Also included are descriptions of the general sequence of a channel operation, the organization of operands in memory, and the proper programming sequence used to start channel operations.

Throughout this section, references will be made to the DMAC internal registers and bits within those registers. It is suggested that the register summary foldout (Figure 3-2) at the end of this book be kept extended for reference while reading the operational description for the first time to quickly gain familiarity with the register functions.

5.1 GENERAL OPERATING SEQUENCE

Any channel operation will follow the same basic sequence of steps outlined below. The following sub-sections explain each of these steps in more detail, including how the channel registers should be programmed to select and control various operating modes.

1. Initialization of channel registers by the system processor.

2. Channel Start Up. This includes setting the STR bit in the CCR and recognizing the first operand transfer request (either internally or externally generated). For the chained modes of operation, the first array entry will be fetched to initialize the MAR and MTCR (and to update the BAR for the linked array chaining mode).

3. Transfer Data Block. After a channel is started, it will transfer one operand in response to each request until an entire data block has been transferred (for exceptions to this sequence, refer to **5.4.1 Bus Cycle Sequence**).

4. Chained Operation. If a channel is programmed for one of the chained operating modes, the channel operation may not terminate at the end of a block transfer. In this mode of operation, when a data block transfer is completed, the channel will read an entry in a command table to reprogram the MAR and MTCR for another block transfer; and will optionally read a new value for the BAR that points to the next chain element. Either the value of the BTCR or value of the BAR is used to determine when a channel operation should be terminated.

5. Continue Operation. In unchained operations, if a channel is initialized for the continue mode, a channel may deviate from a strictly sequential block transfer. In the continue mode, when the MTCR decrements to zero, the MFCR, MAR, and MTCR will be loaded from the BFCR, BAR, and BTCR before the next operand transfer request is honored and the channel will continue operation (returning to step 3).

6. Channel Termination. A channel operation can be terminated for several reasons: a peripheral device asserts the $\overline{\text{DONE}}$ signal during an operand transfer, the MTCR is decremented to zero, a bus cycle is terminated with a bus error, the system processor sets the SAB bit in the CCR, or an external device asserts an abort input. There are also several illegal operations that may terminate a channel operation.

5.2 CHANNEL INITIALIZATION

In order to start a block transfer operation, the system processor must initialize channel registers with information describing the data block, device type, request generation method, and other special control options. It may be necessary to write into as many as 17 registers to initialize a channel; however, after a channel has been initialized once, only three or four registers may need to be written to start subsequent transfer operations.

5.2.1 Memory and Operand Description

Most DMA transfer operations involve the movement of data into or out of a random-access memory array (although the DMAC also supports device-to-device transfer operations) and thus, the DMAC must be programmed with the location and size of the data block to be moved. Three registers are used to hold this information; the MAR and MFCR, which provide a 24-bit address and a 3-bit function code to locate data anywhere in one of eight 16 megabyte linear address spaces, and the MTCR which is a 16-bit transfer counter that defines a block size of up to 65,535 operands (bytes, words, or long words). The MTCR is decremented by the DMAC after each operand transfer until it reaches zero, and the MAR may be programmed to be decremented or incremented after each operand transfer.

Although the DMAC does not place any format requirements on the data that is transferred during a channel operation (other than the format of the command tables used during chained operations), it does use the M68000 conventions for operand alignment and byte significance. As shown in Figure 5-1, bytes are individually addressable with the high-order byte having an even address the same as the word. The low-order byte has an odd address that is one count higher than the word address. Word and long-word

1 BYTE = 8 BITS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOWER ADDRESSES	MSB		BYTE 0				LSB			BYTE 1					
		BYTE 2								BYTE 3					

1 WORD = 16 BITS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB							WORD 0								LSB
HIGHER ADDRESSES						WORD 1									

1 LONG WORD = 32 BITS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB															
HIGHER ADDRESSES					LONG WORD 0										LSB
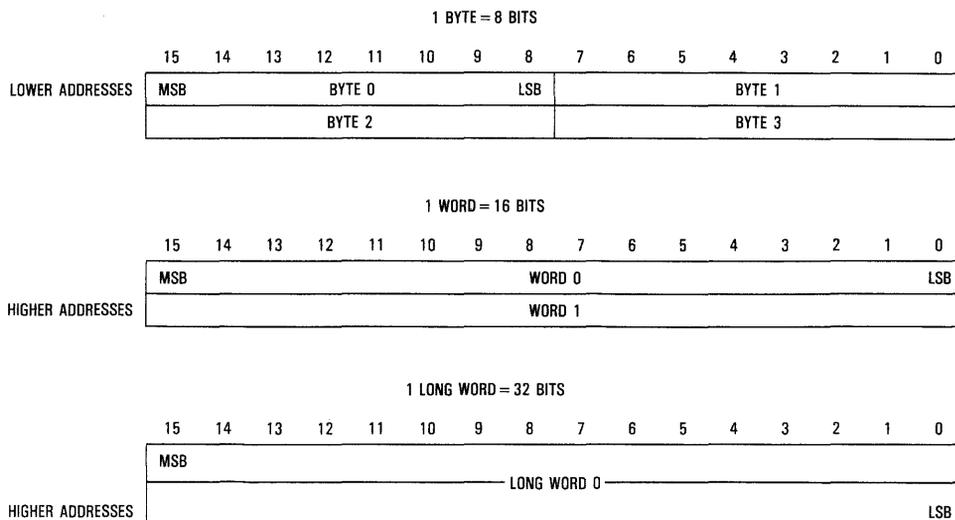
Figure 5-1. Memory Data Formats

1-873

data is accessed only on word (even byte) boundaries, with the lowest address byte being most significant. The size field in the OCR is used to select the operand size for a block transfer.

5.2.2 Device Description

Four types and two sizes of peripheral devices are supported by the DMAC. The types supported are explicitly addressed M68000 bus compatible, explicitly addressed M6800 bus compatible, implicitly addressed with acknowledge only, and implicitly addressed with acknowledge and ready. The type and size of a device is programmed in the DCR, and the address and address space of an explicitly addressed device is programmed into the DAR and DFCR.

A device port may be 8 or 16 bits wide. If a 16 bit wide, explicitly addressed device is to support byte operand sizes, the data port must be accessible as two bytes as well as a word. For all other legal combinations of port and operand sizes, the device port will always be accessed full width (i.e., an 8-bit port accessed as a byte and a 16-bit port accessed as a word). If a memory-to-memory transfer is to be executed by the DMAC, the device is programmed as M68000 compatible, with a 16-bit port size.

The functional definition of the PCL input is also part of the device description. If the transfer protocol selected is implicitly addressed with acknowledge and ready, the PCL input is used as an active low ready signal. If the device type selected is M6800, the PCL input is used as the E clock for the synchronous bus cycles. Otherwise, the DCR may be programmed to select one of four other functions for the PCL line; a status, interrupt, or abort input or a start pulse output. As a status input, the level of the PCL line can be read at anytime through the PCS bit in the CSR, regardless of any other function associated with it. Also, in all modes, a high-to-low transition on the input will set the PCL transition bit (PCT) in the CSR and that bit can be read at any time. If the status with interrupt function is selected, the PCT bit is set and interrupts are enabled, the DMAC will assert an interrupt request to the system processor. If the abort function is selected, the channel operation will be terminated when the PCT bit is set and the CSR will reflect the aborted operation. As a start pulse output, the PCL line will be driven low for eight clocks when the channel is started, and will remain high otherwise.

The DMAC supports peripheral device sizes of 8 or 16 bits using either single or dual address transfers of byte, word, or long-word operands. When all combinations of the above are considered, 12 different transfer operations are possible; however, four of those operations are not legal. Table 5-1 shows a matrix of the possible device, operand size, and transfer type combinations, and whether or not each is legal.

Table 5-1. Device and Operand Size Combinations

	Dual Address			Single Address		
	Byte Operand	Word Operand	Long Operand	Byte Operand	Word Operand	Long Operand
8-Bit Device					X	X
16-Bit Device	*			X		X

X Denotes Illegal Combinations
*Legal Only with Internal Maximum-Rate or Limited-Rate Auto-Request Generation

1-873

5.2.3 Operation Description

A wide variety of block transfer operations are supported by the DMAC by choosing from the available device sizes, bus transfer protocols, request generation methods, and special operating modes. The following paragraphs discuss the channel operation functions that can be programmed to facilitate various block transfers.

5.2.3.1 TRANSFER DIRECTION. The DIR bit in the OCR is used to indicate the direction of the block transfer, either from memory to the device or vice versa. During single address transfers, the R/$\overline{W}$ signal will be driven high if the direction is from memory to device and driven low if the direction is device to memory. This is identical to a normal read or write cycle for the memory, but exactly opposite for the peripheral, since a high level indicates a write to the peripheral and a low level indicates a read from the peripheral.

During dual-address transfers, the DIR bit determines whether the MAR or DAR is used as the source pointer or destination pointer. For memory-to-device transfers, the value in the MAR (and MFCR) will be used during read cycles and the value in the DAR (and DFCR) will be used during write cycles. The opposite relationship is true if the DIR bit indicates a device-to-memory transfer.

5.2.3.2 ADDRESS SEQUENCING. The manner in which the MAR and DAR are incremented or decremented during a transfer operation depends on the programming of the SCR, operand size, device port size, and transfer protocol used. Both the MAR and DAR can be individually programmed to be incremented or decremented after a successful operand transfer or to remain unchanged.

When using the single-address transfer protocol, the operand size must always be equal to the device port size. Thus, all operands can be transferred in one bus cycle and only the MAR is incremented or decremented (the DAR is not used). The amount by which the MAR is incremented or decremented will be one or two for byte or word operands respectively (if the MAC field in the SCR is programmed for increment or decrement). If no increment has been programmed for the MAR, all bus cycles will use the same address.

When using the dual-address transfer protocol, the DMAC will always run at least two, and up to six bus cycles to transfer each operand. When the operand size is larger than either the memory width or device port size, the DMAC will run multiple bus cycles to transfer operand parts until the entire operand has been transferred. If the source (memory or device) is smaller than the operand size, two or four read cycles will be executed to fetch each operand; and, if the destination (device or memory) is smaller than the operand size, two or four writes cycles will be executed to store each operand.

One exception to the above rules is when the combination of a 16-bit device and byte operand size is used with internal request generation. In this case, the DMAC minimizes bus utilization by transferring two operands with each bus cycle if possible. In other words, most bus cycles during such a transfer will be word read or write cycles to transfer two operands (bytes) at a time.

If the SCR is programmed to increment or decrement for either the MAR or DAR, the manner in which the address register(s) is incremented or decremented will depend on whether the access is to memory or the device, and the operand/port size combination. Table 5-2 shows how the MAR and DAR will be incremented for all combinations of operand and device sizes in the dual-address mode. It should be noted that when an 8-bit device is used, the DAR will be incremented or decremented by two, four, or eight for byte, word, or long-word operands respectively, and consecutive odd or even addresses will be accessed. This is consistent with the M68000 MOVEP instruction, so that an 8-bit device may be placed on one-half of a 16-bit bus and successive bytes within the device will be accessed. If no increment is selected for the DAR, and word or long-word operands are to be transferred to or from an 8-bit device, successive bytes will be accessed in the same manner as just described, but the value of the DAR will not be changed; thus each operand transfer will start at the same address.

Table 5-2. Address Register Sequencing Rules

	16-Bit Memory			
	8-Bit Device (Memory / Device)		16-Bit Device (Memory / Device)	
Byte Operand	±1	±2	±1*	±1*
Word Operand	±2	±4	±2	±2
Long Word Operand	±4	±8	±4	±4
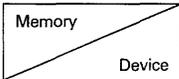
(Legend: upper diagonal = Memory, lower diagonal = Device — Address Register Change)

*If an internal request generation method is used, these values will be doubled and word bus cycles will be used when possible to minimize bus utilization.

1-875

If an exceptional condition occurs during an operand transfer that preempts normal termination, no address registers will be incremented. Also, any data that was previously read from the source (memory or device) into the internal holding register will be lost when a bus error occurs. Thus, if it is desired to continue a channel operation after a write cycle is aborted due to a bus error (for example, after a page fault in a virtual memory system), the system processor may simply restart the channel and the aborted operand transfer will be attempted again. The MTCR will not be changed until all bus cycle required to transfer an operand have completed normally, so it will not need to be modified to continue the operation.

5.2.3.3 TRANSFER REQUEST GENERATION. In order for a peripheral to control the transfer of operands to or from memory in an orderly manner, it uses an input signal to the DMAC as a service request. If a memory-to-memory transfer or a peripheral with no request signal is used, the DMAC can internally generate operand transfer requests. If an internal request generation method and dual-address transfer protocol is used, the DMAC will not assert $\overline{ACK}$ when performing device accesses. Thus, a channel can be used both for transfers between a device (either implicitly or explicitly addressed) and memory using external request generation, and for memory-to-memory transfers using internal request generation without conflicting with the operation of the device.

The request generation method used for a channel is programmed in the OCR. If an external request method is used, the type is further defined by the DCR. If the internal limited rate method is used, the GCR is used to determine the timing constants for bus utilization calculations. If the auto-sort external request generation method is used, the first request is generated internally when the channel is started; subsequent requests are generated using the selected external request method.

5.2.3.3.1 Internal, Maximum Rate. The simplest method of internal request generation is to always have a transfer request pending for a channel until the transfer count is exhausted. If this method is used, as soon as a channel is started the DMAC will arbitrate for the bus and begin to transfer data when it becomes bus master. If no exceptional conditions occur, all operands in the data block will be transferred in one burst, so that the DMAC will use 100% of the available bus bandwidth.

5.2.3.3.2 Internal, Limited Rate. In order to guarantee that the DMAC will not use all of the available system bus bandwidth during a transfer, internal requests can be generated according to the amount of bus bandwidth allocated to the DMAC. This method of request generation is called limited-rate auto-request (LRAR) and uses three constants programmed via the GCR to monitor bus activity and allow the DMAC to use a portion of the bus bandwidth.

One constant is a sample period measured in clock cycles. During each sample period, the DMAC monitors the $\overline{\text{BGACK}}$ signal as an input and counts the number of clocks during which it is asserted. After a sample period has ended, the number of clocks during which $\overline{\text{BGACK}}$ was asserted (by any bus master, including the DMAC) is compared to a second programmable constant called the burst time. If the previous sample count is less than the burst time, then the DMAC will generate internal requests for any channel programmed for LRAR during the burst time for the current sample period. In this manner, if the DMAC or other bus master asserts $\overline{\text{BGACK}}$ for more than the burst time during any sample period, no limited rate requests will be generated during the next sample period so that the system processor will be able to use the bus (this assumes that the system processor does not assert $\overline{\text{BGACK}}$ when it owns the bus). The third programmable constant, which determines the amount of the bus bandwidth available to DMA devices, is the ratio of the sample period length to the burst time length.

Of the three constants used in the LRAR equation, two are programmed directly into the GCR and the third is calculated from the other two. The fraction of the bus bandwidth available to the DMAC is equal to $2^{-(BR+1)}$, the burst time is equal to $2^{(BT+BR+5)}$. Figure 5-2 depicts graphically how the three constants are used to determine bus utilization by the DMAC. Table 5-3 lists all combinations of BT and BR and the resulting MPU, burst and sample period lengths for each.

$2^{-(BR+1)}$ is fraction of bandwidth available to bus masters other than MPU.
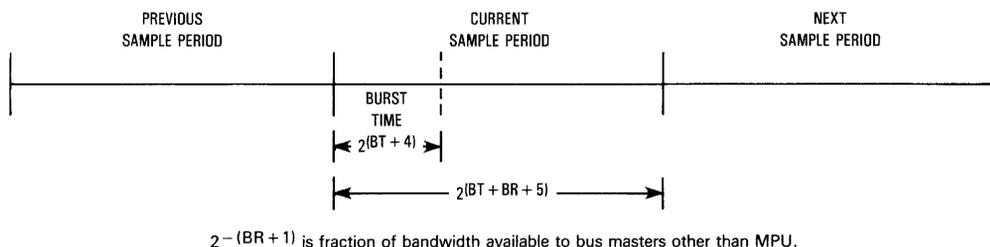
1-875

Figure 5-2. Limited-Rate Auto-Request Operation Timing Parameter Relationships

**Table 5-3. Limited-Rate Auto-Request Timing
Parameter Value Combinations**

BR	BT	Burst Time*	MPU Period*	DMA Ratio	Sample Period*
00	00	16	16	50.00%	32
00	01	32	32	50.00%	64
00	10	64	64	50.00%	128
00	11	128	128	50.00%	256
01	00	16	48	25.00%	64
01	01	32	96	25.00%	128
01	10	64	192	25.00%	256
01	11	128	384	25.00%	512
10	00	16	112	12.50%	128
10	01	32	224	12.50%	256
10	10	64	448	12.50%	512
10	11	128	896	12.50%	1024
11	00	16	240	6.25%	256
11	01	32	480	6.25%	512
11	10	64	960	6.25%	1024
11	11	128	1920	6.25%	2048

*Measured in cycles of the DDMA clock input.

1-876

5.2.3.3.3 External, Burst Mode. For peripheral devices that require very high data transfer rates, the burst request mode allows the DMAC to use all of the bus bandwidth under control of the device. In the burst mode, the $\overline{\text{REQ}}$ input to the DMAC is level sensitive and sampled at certain points to determine when a valid request is asserted by the peripheral. The device requests service by asserting $\overline{\text{REQ}}$ and leaving it asserted. In response, the DMAC arbitrates for the system bus and begins to perform an operand transfer. During each operand transfer, the DMAC will assert $\overline{\text{ACK}}$ to indicate to the device that a request is being serviced. If $\overline{\text{REQ}}$ is asserted when the DMAC asserts $\overline{\text{ACK}}$ and remains asserted until the rising edge of the clock on which DTC is asserted, then a valid request for another operand transfer is recognized and the DMAC will service that request immediately (if a higher priority channel does not have a pending request). If $\overline{\text{REQ}}$ is negated before the rising edge of the clock one clock cycle before DTC is asserted, a new request will not be recognized and other channels may be serviced, or the DMAC will release ownership of the bus and return to the idle state.

5.2.3.3.4 External, Cycle Steal. For devices that generate a pulsed signal for each operand to be transferred, the cycle steal request generation mode uses the $\overline{\text{REQ}}$ pin as a falling edge sensitive input. The $\overline{\text{REQ}}$ pulse generated by the device must be asserted during two consecutive falling edges of the DMAC clock to be recognized as valid; thus, if the peripheral generates it asynchronously, it must be at least two clock periods long. The DMAC will respond to cycle steal requests in the same manner as all other requests; however, if subsequent $\overline{\text{REQ}}$ pulses are generated before $\overline{\text{ACK}}$ is asserted in response to each request, they will be ignored. If $\overline{\text{REQ}}$ is asserted after the DMAC asserts $\overline{\text{ACK}}$ for the previous request and before DTC is asserted, then the new request will be serviced before bus ownership is released. If a new request has not been generated for any channel by the time DTC is asserted, the bus may be released to the next master.

5.2.3.3.5 External, Cycle Steal with Hold. In order to reduce the latency time required for the DMAC to respond to a cycle steal request, the DMAC can retain bus ownership for a short time after an operand transfer in anticipation of a new request within that time period. This eliminates bus arbitration delays, at the expense of wasting some bus bandwidth due to idle "hold" periods.

This request generation method is identical to external, cycle steal (refer to **5.2.3.3.4 External, Cycle Steal**) with the exception that the DMAC will retain bus mastership for one full sample period (as defined by the values programmed into the GCR, refer to **5.2.3.3.2 Internal, Limited Rate**) and then relinquish the bus if a new request has not been received. The bus hold time is one full sample period. Thus, the DMAC will wait until the end of two sample periods before giving up the bus. If a request is recognized on a higher priority channel during the hold time, that channel will be serviced and then the DMAC will relinquish the bus if a new request is not present for the cycle steal with hold channel. If a request is received during the hold time, a bus cycle to service the request will start within six clocks.

Note that the DMAC will hold the bus for one full sample period after the last operand transfer of a channel operation, even though no further requests can be serviced.

5.2.3.4 CHANNEL PRIORITY. Each channel has a priority level, which is determined by the contents of the channel priority register (CPR). The priority of a channel is a number from zero to three, with zero being the highest priority level. When multiple requests are pending on the DMAC, the channel with the highest priority receives first service. The priority of a channel is independent of the device protocol or the request mechanism for that channel. If there are several requesting channels at the highest priority level, a round-robin arbitration scheme is used, such that the least recently serviced channel will be the next one to be serviced. This priority scheme is also used when multiple channels have pending interrupts and the MPU executes an interrupt acknowledge.

5.2.3.5 INTERRUPT OPERATION. In order for the system processor to determine the status of a channel, it may read the CSR at any time (assuming that it can gain control of the DMAC bus) or the DMAC can be programmed to generate an interrupt to inform the processor of certain events. Four registers take part in the interrupt generation and response structure of the DMAC: the CCR, CSR, NIVR, and EIVR.

The interrupt bit in the CCR is used to enable a channel to generate interrupts via the $\overline{\text{IRQ}}$ signal. If a channel is enabled to generate interrupts and the COC, BTC, or PCT (with PCL programmed as a status with interrupt input) bit is set in the CSR, then the $\overline{\text{IRQ}}$ signal will be asserted. As long as a valid interrupt condition is present for any channel, the $\overline{\text{IRQ}}$ signal will be asserted, and it is the responsibility of the system processor to clear the appropriate bit(s) to negate the interrupt (refer to **3.7.1 Channel Status Register**).

In response to the assertion of $\overline{\text{IRQ}}$, an M68000 Family processor will execute an interrupt acknowledge bus cycle to read a vector number that is used to locate the interrupt handler routine. External hardware may be used to return a vector number to the processor during the interrupt acknowledge cycle; however, the DMAC efficiently supports the M68000 interrupt structure by supporting two separate vector numbers for each channel and using internal priority and status information to select the proper value to be returned.

In order to cause the DMAC to return one of its vectors to the processor, the $\overline{\text{IACK}}$ input is asserted during the interrupt acknowledge cycle. If the DMAC is asserting $\overline{\text{IRQ}}$ when $\overline{\text{IACK}}$ is asserted, it will return a vector number from the highest priority channel that has an interrupt condition present. The vector number that is returned depends further on the error (ERR) bit in the CSR of the highest priority channel. If ERR is clear, then the value in the NIVR will be returned, otherwise the value of the EIVR will be returned. If $\overline{\text{IRQ}}$ is negated when $\overline{\text{IACK}}$ is asserted, the $\overline{\text{IACK}}$ will be ignored.

5.3 CHANNEL START-UP

Once a channel has been initialized with all parameters required for a transfer operation, it is started by writing a one with a byte write to the STR bit in the CCR. This write cycle will not leave a one in the STR bit position, but the ACT bit in the CSR will be set to indicate that the channel has been started. After the channel has been started, any register that describes the current operation may not be modified or an operation timing error will occur. The registers that may not be modified are the DCR, OCR, SCR, MTCR, MAR, MFCR, DAR, and DFCR. All of the other channel registers may be modified if desired during a channel operation; thus, status bits can be cleared in the CSR, the operation can be controlled by setting bits in the CCR, and the channel priority level and interrupt vector numbers may be modified through the CPR, NIVR, and EIVR. The GCR can always be modified so that DMA bus utilization may be dynamically controlled.

In order to assure that status information from previous transfer operations has been read by the system processor before starting a new operation, certain status bits in the CSR must be clear when the STR bit is set. The status bits that must be clear are the COC, BTC, NDT, and ERR bits. Thus, part of the channel initialization and start up procedure is to clear the CSR before the STR bit is set. If any of the above mentioned bits are set when an attempt is made to set the STR bit, an operation timing error will occur.

If the channel operation being started is the first block of a transfer using the continue mode, the CNT bit in the CCR may be set by the same write cycle that sets the STR bit; however, CNT must not be set prior to setting STR. Also, the BTC bit in the CSR must be clear before the CNT bit is set, for the first or subsequent blocks in the transfer operation.

Once the STR bit has been set, the channel will become active and accept operand transfer requests. When the first valid request is recognized for the channel, it will enter the DMA mode of operation by

arbitrating for the bus. It should be noted that the $\overline{REQ}$ input is ignored until the channel is started, so that the channel will not recognize transfer requests from the device until it is made active (the PCL line can be programmed as a start pulse output that will signal an external device when a channel is ready to accept requests). The PCL input, however, is not ignored before the channel is started. If the PCL line is programmed as a status input and a high-to-low transition occurs on it at any time, the PCT bit in the CSR will be set. Thus, an interrupt may be generated even when the channel is not active.

In order to begin a data transfer operation, the following general sequence for programming the DMAC should be followed.

1. Write a one to the SAB bit and a zero to the INT bit in the CCR. This will cause any previous operation to be terminated and prevent the channel from generating an interrupt request. If the previous state of a channel is known, this operation is not necessary.

2. Load the channel registers with parameters describing the memory, device, and operation to be performed. If interrupts are to be generated by the channel operation, the NIVR and EIVR should be loaded with the appropriate vector numbers and the corresponding locations in the exception vector table should be loaded with the start address of the interrupt handler routines. If the continue or chaining modes of operation are to be used, the BFCR, BAR, and BTCR must also be loaded.

3. Clear any previous status information by writing a $F6 (or $FF) to the CSR.

4. Start the channel by writing a one to the STR bit of the CCR. The CNT and INT bits may also be set by the same write cycle, if desired, in order to start a continued operation and enable interrupts. Also note that the SAB and HLT bits should be written as zero in order to avoid an immediate abort or halt operation.

5. Write to the peripheral device control registers as necessary to enable the device to generate requests and begin the transfer operation. This operation may be performed before the DMAC channel is started if internal request generation is used, depending on the requirements of the device.

5.4 DATA TRANSFERS

When the DMAC is performing a data transfer operation, several factors affect the order in which bus cycles will be run and when each cycle starts in relationship to other operations. The following paragraphs describe the bus cycle sequence for each of the transfer protocols and the effects of simultaneous channel operations, bus exceptions, and special operations on that sequence.

5.4.1 Bus Cycle Sequence

The order in which the DMAC executes bus cycles during a transfer operation depends primarily on the transfer protocol used and the device and operand size combination. The following descriptions of the sequence used by each protocol assumes that only one channel is active during the operation.

5.4.1.1 SINGLE ADDRESS TRANSFERS. The sequence used for this protocol is very simple. When a request is recognized internally, the DMAC will arbitrate for the bus if necessary and execute one bus cycle in response to each request. Since the operand size must match the device port size for single address transfers and one bus cycle is run for each request, the number of normally terminated bus cycles

executed during a transfer operation will always be equal to the value programmed into the MTCR (unless the transfer is terminated early by the peripheral device). The sequencing of the address bus will follow the programming of the MAC field in the SCR. If programmed for no increment, all of the bus cycles will access the same, initial address. If programmed to count up or down, each successive bus cycle will access the address one operand size beyond or before the last cycle in a linear fashion.

5.4.1.2 DUAL ADDRESS TRANSFERS. Each operand transfer in the dual address mode will require at least two and as many as six bus cycles in response to each operand transfer request. Table 5-4 shows the number of bus cycles required to transfer each operand for the eight memory, device, and operand combinations.

Table 5-4. Bus Cycles per Operand Transfer-Dual Address

	16-Bit Memory	
	8-Bit Device	16-Bit Device
Byte Operand	1 or 2**	2*
Word Operand	3	2
Long-Word Operand	6	4

*Packing will be performed such that two operands will be transferred per cycle.
**Depends on the state of the internal FIFO holding register prior to the request.   1-877

For the combinations that require two of four bus cycles per operand, an equal number of cycles will access memory and the device; for the three and six bus cycle cases, two cycles will access whichever port is smaller and one cycle will access the larger port. The bus cycle sequence will always follow the pattern of: read a byte or word using as many bus cycles as are required and then write that operand part, again using as many bus cycles as are required; this process is then repeated for long-word operands.

The address register and transfer count register sequencing during a dual address transfer will follow the rules discussed previously for increment/decrement amounts except for the case discussed in the next paragraph. An address register will be incremented or decremented only after the entire operand has been successfully transferred. The MTCR will be decremented by one when the last bus cycle of the operand transfer has been completed successfully.

For most combinations of memory device, and operand size, the number of bus cycles run for each operand transfer count will be two, three, or four, according to Table 5-4, regardless of the operand request generation method used or transfer count value. One combination that is not consistent with this rule is when an operand size of byte is used to transfer data between memory and a 16-bit device using internal request generation. In this case, the DMAC will perform the transfer in order to best utilize the available bus bandwidth by performing word read and write cycles when possible.

Once the DMAC has started a dual-address operand transfer, it must complete that transfer before releasing ownership of the bus or servicing requests for another channel of equal or higher priority, unless one of the bus cycles during the transfer is terminated with a relinquish and retry condition. When any bus cycle is terminated with a halt, the DMAC will halt operation and release ownership of the bus immediately following the termination of that bus cycle. When the $\overline{BEC}$ encoding is returned to normal and bus ownership is returned to the DMAC, any operand transfer previously suspended must then be completed, regardless of pending requests for the other channels. The same rule applies to cycles terminated with retry or relinquish and retry; the same cycle that was terminated with the retry will be run again before the other channels can be serviced. If a bus cycle is terminated with a bus error, the channel operation will be terminated and any portion of the operand that was a read but not yet written to the destination will be lost.

5.4.1.3 TRANSFER COUNT SYNCHRONIZATION. An important consideration in any system where two devices (the DMAC and a peripheral) are transferring a fixed number of data items is that both devices count the same number of operand transfers under all conditions. The DMAC facilitates this count synchronization with the $\overline{DTC}$ signal in two ways. First, $\overline{DTC}$ always indicates to a device that a bus cycle to transfer an operand or part of an operand has been successfully completed so that the device may update its transfer pointer (for an internal FIFO for example). Second, if the device counts the number of assertions of $\overline{DTC}$ and knows how many device bus cycles are required to transfer each operand, it can properly decrement its transfer counter in step with the DMAC MTCR. A device should never use $\overline{ACK}$ by itself to clock pointer and counter registers if it is desired to support exceptional bus conditions. This is due to the fact that $\overline{ACK}$ will be asserted at the beginning of a bus cycle to the device, but if a bus cycle is terminated with a bus error, retry, or relinquish and retry, $\overline{DTC}$ will not be asserted.

5.4.2 Effects of Channel Priority

When all the channels of the DMAC are active, there exists the possibility that operand transfers will be interleaved due to the priority scheme used to resolve simultaneous pending requests. The effects of channel priority on the operand transfer sequence during simultaneous channel operations is independent of the request generation method or transfer protocol used. In all cases, the following rules apply to the operand transfer order used.

1. Once a channel has started the transfer of an operand, the entire operand must be successfully transferred or the channel operation aborted before the other channels may be serviced, regardless of priority relationships. The halt, retry, and relinquish and retry exceptional bus conditions will not affect the order in which operands are transferred, only the order of bus cycles during an operand transfer or the delay between successive bus cycles. For example, if a lower priority channel is being serviced and a bus cycle is terminated with a relinquish and retry, when the DMAC regains bus ownership, the lower priority operand transfer will be completed even if a higher priority request is pending. This is to preserve the contents of the internal data holding register.

2. If two channels are at different priority levels, the lower priority channel will not be serviced as long as higher priority channels have a request being serviced or pending.

3. If two or more channels are at the same priority level and have pending requests, the last channel serviced will be the last channel serviced again. For example, if two channels use internal maximum rate request generation and are active simultaneously, successive operand transfers will service alternate channels.

4. When the memory transfer counter (MTC) is exhausted, the channel operation will continue if the channel is set to a chaining mode and the chain is not exhausted. The reading of the next descriptor from the array is treated as an operand transfer, and must be completed before any higher priority requests can be serviced.

5.4.3 Exceptional Bus Conditions

In any computer system, there always exists the possibility that an error will occur during a bus cycle due to a hardware failure, random noise, or because an improper access is being attempted. When an asynchronous bus structure, such as the one supported by the M68000 Family, is used in a system, it is very simple to make provisions to allow bus masters to detect and respond to errors during a bus cycle. The DMAC uses a three bit, binary encoded input bus ($\overline{BEC0}$-$\overline{BEC2}$) to allow external hardware to indicate when an abnormal condition has occurred. The following paragraphs describe the exceptional conditions

that can be detected by the DMAC and their affects on a data transfer operation. Refer to **4.4.2 Bus Exception Control Functions** for a detailed description of the operation of the $\overline{\text{BEC}}$ pins and the encodings used to indicate various conditions.

5.4.3.1 RESET. In the event of a catastrophic system failure, including recovery from a power-down state, the DMAC may be returned to an uninitialized state by asserting the reset encoding on the $\overline{\text{BEC}}$ pins. Regardless of any current operation being performed, the DMAC will immediately abort all channel operations and return to the idle state. In the event that a bus cycle is in progress when reset is detected, it will be terminated in an orderly fashion, the control and address/data pins will be three-stated and bus ownership released.

5.4.3.2 HALT. Channel transfer operations may be temporarily suspended at any time by asserting halt to the DMAC. In response, any DMA bus cycle that is in progress will be completed (when $\overline{\text{DTACK}}$ and/ or ready is asserted) and bus ownership will be released. No further bus cycles will be started as long as halt remains asserted. The system processor may access the DMAC internal registers to determine channel status or alter operations. When halt is negated, if the DMAC was in the middle of an operand transfer when halted or if there is a transfer request pending, it will arbitrate for the bus and continue normal operation.

5.4.3.3 BUS ERROR. When a fatal error occurs during a bus cycle, bus error is used to abort the cycle and terminate a channel operation. When bus error is asserted during a bus cycle, it will terminate in an orderly fashion, but the channel that was being serviced by that bus cycle will have the current operation terminated with an error signaled in the CSR. When the bus cycle is terminated, the DMAC will retain ownership of the bus as long as bus error remains asserted and will not start any new bus cycles to service the other channel. When bus error is negated, a bus cycle will be started to service the other channel, if a transfer request is pending; otherwise, bus ownership will be released.

5.4.3.4 RETRY OPERATIONS. If a correctable error occurs during a bus cycle, it is often desirable to cause the bus cycle to be terminated and then re-execute at a later time. The DMAC supports two such operations that perform the same function, but in different ways: retry, and relinquish and retry. When either retry signal is asserted during any DMA bus cycle, that cycle is terminated in an orderly fashion, but no data transfer operation takes place. When the retry signal is later negated, the same bus cycle that was terminated with the retry will be executed again, using the same address, data, and control information. If the other channel has a higher priority request pending, it will not be serviced before the retry has been executed and the entire operand for the current channel has been transferred.

5.4.3.4.1 Retry. When the retry encoding is asserted during a bus cycle, the DMAC will terminate the cycle and suspend any further bus cycles until retry is negated, while retaining ownership of the bus. When retry is negated, the bus cycle will be executed again and normal operations will be continued.

5.4.3.4.2 Relinquish and Retry. When the relinquish and retry encoding is asserted during a bus cycle, a DMAC will terminate the cycle, release ownership of the bus, and suspend any further operations until relinquish and retry is negated. During this time, the system processor may access the DMAC internal registers to check channel status or modify the operation. When relinquish and retry is negated, the DMAC will then arbitrate for the bus, re-execute the previous bus cycle, and continue normal operations.

5.4.4 External Request Recognition

When an external generation method is used to inform the DMAC that a peripheral device needs service, only one request can be pending for each channel at any time. In order to guarantee that every peripheral request is recognized by the DMAC, the protocol described previously must be followed; i.e., the peripheral must wait until $\overline{ACK}$ has been asserted once for each successive transfer request before issuing a new request. Furthermore, the assertion of $\overline{DTC}$ must be used by the peripheral to determine if a bus cycle terminated normally in order to properly handle a bus error, retry, or relinquish and retry termination. Thus, $\overline{ACK}$ indicates that the DMAC is starting to service a request, while $\overline{DTC}$ indicates that the request has been successfully serviced. If a request is pending for a channel at the end of a bus cycle terminated with retry or relinquish and retry, further requests will not be recognized until the retry has been successfully completed and the DMAC has begun to service the pending request.

When single address transfers are used, this protocol is very easy to follow since $\overline{ACK}$ will only be asserted once for each operand transfer. However, during dual address transfers, $\overline{ACK}$ may be asserted up to four times for each operand transfer and thus the DMAC will only recognize requests at certain points during such a transfer.

5.4.5 Software Control

During a data transfer operation, the system processor may interrogate the CSR (and other registers if desired) to monitor a channel operation. It is assumed that the system processor may gain ownership of the DMAC bus in order to perform the MPU accesses to the DMAC. If it is determined that an operation should be suspended or stopped, the system processor may also write to the CCR to cause such action to be taken by the DMAC. The following paragraphs describe the software control functions available and the DMAC response to them.

5.4.5.1 SOFTWARE HALT. The system processor may temporarily suspend channel operation at any time by setting the HLT bit in the CCR. In response, the channel will remain active, but no further operand transfer requests will be serviced for that channel as long as the HLT bit remains set. If no requests are pending when the HLT bit is set, the channel will recognize a transition or assertion of $\overline{REQ}$ as a valid request, but only one request will be made pending while the channel is halted. When the processor clears the HLT bit, the channel will then resume normal operation and may service a pending request.

The software halt functions similarly to the hardware halt operation in that a channel may be halted at any point during a block transfer, but the system processor will not normally have access to the DMAC registers in the middle of an operand transfer operation. One exception to this is the case when a dual address transfer read cycle (or the first of two write cycles) is terminated with a hardware halt or relinquish and retry; the system processor will have access to the CCR to set the HLT bit before the next bus cycle. In this case, when the $\overline{BEC}$ pins return to normal, the DMAC will rearbitrate for the bus and complete the dual address operand transfer before the software halt will be honored. It should also be noted that pending transfer requests for the other channels can not be serviced until the system processor has returned bus ownership to the DMAC and the operand transfer for the halted channel is complete.

5.4.5.2 SOFTWARE ABORT. The system processor may terminate a channel operation at any time by setting the SAB bit in the CCR. When this bit is set, the channel operation will be terminated immediately, the ACT bit will be cleared, and the COC and ERR bits will be set in the CSR. A software abort will be flagged in the CER to indicate that the abort has been honored.

The software abort operation may be executed at any point during a channel operation, but the system processor will normally not have access to the DMAC registers in the middle of an operand transfer operation. One exception to this is the case when a dual address transfer read cycle (or the first of two write cycles) is terminated with a hardware halt or relinquish and retry, then the system processor will have access to the CCR to set the SAB bit before the next bus cycle. In this case, the DMAC will remain idle until the BEC pins return to normal, but the remainder of the operand transfer will not be executed and the data that was read from the source will be lost.

5.4.6 Hardware Abort

In order to allow external hardware to abort a channel operation at any time, the PCL signal may be programmed as an abort input. This input operates in a manner similar to the software abort just described by setting the PCT bit in the CSR when a high-to-low transition occurs on PCL. The DMAC responds to PCT being set with PCL programmed as an abort input by terminating the channel operation, including any bus cycle that is being executed. If a bus cycle is in progress when the abort signal is asserted, an orderly termination of the bus cycle will occur in the same manner as if a bus error exception code were used to terminate the cycle.

The PCL signal may be asserted at any time to set the PCT bit. It should be noted that if PCL is programmed as an abort input and the PCT bit is set when a channel is started, it will be aborted before the first operand is transferred. This is true whether the PCT bit was set during a previous operation and was never cleared, or by a transition of PCL before the channel was started.

NOTE
The contents of the MAR, DAR, and MTCR may not be valid after an external abort.

5.5 BASE REGISTER OPERATIONS

Three special operating modes are available to reduce the system processor overhead required for certain operations, and to allow non-contiguous data blocks to be transferred to or from a device or a contiguous data block; all in one channel operation. These modes are the continue and chaining modes of operation and all three utilize the BFCR, BAR, and BTCR as temporary holding registers to be transferred to the MFCR, MAR, and MTCR at certain points during a channel operation, or as pointers into an array of data block descriptors. All of these modes of operation are independent of the transfer protocol or request generation method used and do not affect the DFCR or DAR.

5.5.1 Continue Operation

To utilize the continue mode of operation, the system processor loads the BFCR, BAR, and BTCR with the address and size of the next block to be transferred after the completion of the current block (described by the MFCR, MAR, and MTCR) and sets the CNT bit in the CCR. The only limitation on when this operation may be performed is that the CNT bit may not be set before a channel is active (it may be set at the same time that the channel is made active) and the BTC bit in the CSR may not be set prior to an attempt to set CNT. When the channel decrements the MTCR to zero and the CNT bit is set, it will not terminate the operation, but instead remain active (although the $\overline{\text{DONE}}$ signal is asserted to signal the end of each block). The BFCR, BAR, and BTCR will then be transferred to the MFCR, MAR, and MTCR, the CNT bit will be cleared and the BTC bit will be set. The channel will then resume normal operation by responding to operand transfer requests to transfer the new block of data. If the channel completes the transfer of the new block of data and the CNT bit is not set, the channel operation will be terminated in the normal fashion.

If the interrupt bit in the CCR is set when the channel completes a block transfer with the CNT bit set, the system processor will receive an interrupt when the BTC bit is set to indicate that the DMAC has completed the previous block transfer and is starting on the next (or the processor may poll the DMAC). At this time, the interrupt handler routine executed by the processor can load the BFCR, BAR, and BTCR with information describing the next data block if necessary, clear the BTC bit, and set the CNT bit to repeat the operation described above as many times as desired. Figure 5-3 shows a flowchart depicting the use of the continue mode.
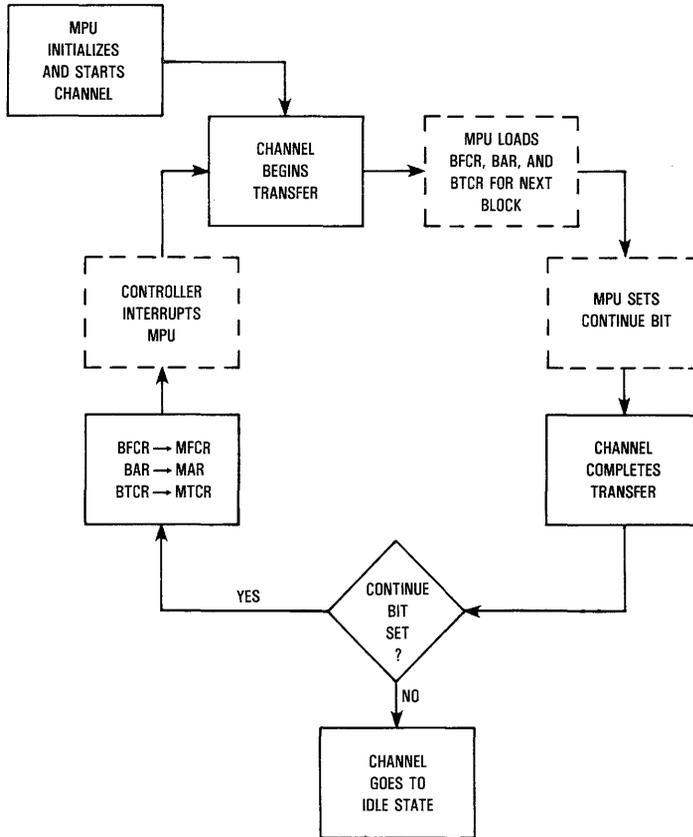
1-878

Figure 5-3. Continue Mode Operation Flowchart

5.5.2 Sequential Array Chaining

This type of a chaining operation uses an array of data block descriptors, organized sequentially in memory, to define the data to be transferred during a single channel operation. The data block descriptor array is defined by the BFCR, BAR, and BTCR, and each data block descriptor defines the location and size of a data block. Figure 5-4 shows the format of the data block descriptor array and the data blocks described by that array.

MEMORY

START OF →
ARRAY

| MAR A |
| MTCR A |
| MAR B |
| MTCR B |
| MAR C |
| MTCR C |

ARRAY →

MEMORY →
ADDRESS C

BLOCK C

TRANSFER
COUNT C

MC68000
MPU

MC68450
DMAC

MAR	*
DAR	PERIPHERAL DEVICE ADDRESS
BAR	START OF ARRAY
MTC	*
BTC	SIZE OF ARRAY

*Loaded from the array

MEMORY →
ADDRESS A

BLOCK A

TRANSFER
COUNT A

MEMORY →
ADDRESS B

BLOCK B

TRANSFER
COUNT B

PERIPHERAL DEVICE
ADDRESS →

PERIPHERAL DEVICE
OR
MEMORY

1-879

Figure 5-4. Sequential Array Chaining Mode Example

To utilize this mode of operation, the BFCR and BAR are loaded with the starting address of the data block descriptor array and the BTCR is loaded with the number of descriptors in the array. The MFCR must be loaded with the address space of all of the data blocks, since that information is not included in a data block descriptor. The channel is then started, and the DMAC will read the first descriptor into the MAR and MTCR, increment the BAR by six, decrement the BTCR by one, and begin to transfer the first data block by servicing transfer requests. When the MTCR is decremented to zero, the DMAC will assert $\overline{\text{DONE}}$ to signal the completion of a block transfer and read the next data block descriptor from the array. This process continues until the DMAC completes a block transfer and the BTCR value is zero, at which time the channel operation is terminated.

NOTE

The data block descriptor array must start at an even address or an address error will occur when the channel is started.

5.5.3 Linked Array Chaining

This mode of operation is very similar to sequential array chaining, except for the organization of the data block descriptor array. In this case, each descriptor not only describes a data block, but also contains a pointer to the next descriptor. This organization allows for easier editing of the descriptor array in order to modify the order of transfer for several data blocks. Figure 5-5 illustrates the organization of a descriptor array for this mode of operation.

To utilize this mode of operation, the BFCR and BAR are loaded with the address of the first descriptor in the array, the MFCR is loaded with the address space for all of the data blocks, and the channel is started. The DMAC then reads the first descriptor block into the MAR and MTCR, and loads the link address from the descriptor into the BAR. The data block is then transferred in the normal manner, and when the MTCR is decremented to zero, the DMAC asserts $\overline{\text{DONE}}$ to indicate the end of a data block transfer, and checks the value in the BAR. If the value is not the NULL pointer (all zeros), then it is used as the address of the next descriptor and the whole sequence is repeated. The channel operation will continue until a block transfer completes and the BAR contains a value of zero.

NOTE

Each of the data block descriptors must start at an even address or an address error will occur.

A comparison of both chaining modes is shown in Table 5-5.

Table 5-5. Chaining Mode Address/Count Information

Chaining Mode	Base Address Register	Base Transfer Counter	Completed When
Sequential Array Chaining	Address of the Array	Number of Data Blocks to be Transferred	BTCR = 0
Linked Array Chaining	Address of the Next Lined Array Descriptor	(Unused)	BAR = 0

1-881

Figure 5-5. Linked Array Chaining Mode Example

NOTE

The chaining modes of operation can be used with any transfer mode or request generation method. However, if the implicitly addressed, with ACK and RDY protocol is used, then the RDY signal must be asserted during the DMA read cycles from the descriptor array. The circuit shown below can be used to provide the correct RDY signal timing for this operating mode. With this circuit, notice that RDY is asserted during every cycle that is not acknowledging the chained channel; however, this does not cause incorrect operation since RDY is ignored when the channel is not active.

1-857

Figure 5-6. $\overline{READY}$ Circuit for Device With Acknowledge and Ready

5.5.4 BTCR Value Restrictions

In the continue or sequential array chaining modes of operation, the BTCR should not be loaded with a value of zero by the MPU or a count error will occur. If a continue or chaining operation occurs and the MTCR is loaded with zeros by the operation, a count error will occur.

5.6 CHANNEL TERMINATION

Once a channel operation has been completed normally or terminated due to an error, the DMAC will reflect the final status of the operation in the CSR and optionally interrupt the system processor. The following paragraphs describe the occurrences that cause a channel operation to be terminated and the status information affected by each.

5.6.1 Transfer Count Exhausted

When the DMAC begins an operand transfer, if the current value of the MTCR is one and an external request generation method is being used, then the $\overline{DONE}$ signal will be asserted during the last bus cycle to the device to indicate that the channel operation will be terminated when the current operand transfer is successfully completed. When the operand transfer is completed and the MTCR is decremented to zero, the channel operation will be terminated, the ACT bit will be cleared, and the COC bit will be set. The MAR and/or DAR will also be incremented in the normal fashion.

5.6.2 Device Termination

A transfer operation may be terminated before the MTCR is decremented to zero by the device if desired. If an external request generation method is used and the $\overline{DONE}$ signal is asserted when $\overline{DTC}$ is asserted during a device access, then the channel operation will be terminated. Also, $\overline{DONE}$ must be sampled as asserted on two consecutive rising clock edges; so if it is generated asynchronously, it must be asserted for two DMAC clock periods.

If the dual address transfer protocol is used, the $\overline{\text{DONE}}$ input will be recognized during any bus cycle of an operand transfer. If $\overline{\text{DONE}}$ is asserted during the first read or write cycle of a word or long-word operand from or to an 8-bit device, the channel operation will be terminated after the entire operand transfer is completed. This means that several more bus cycles may be executed, after $\overline{\text{DONE}}$ is recognized, to write the byte to the memory or a device, such that a full size operand transfer will be executed.

Once the operand transfer is completed, the channel will be terminated, the ACT bit will be cleared, and the COC and NDT bits will be set. The MTCR will also be decremented and the MAR and/or DAR will be incremented in the normal fashion. If $\overline{\text{DONE}}$ is asserted by the device at the same time that the DMAC is asserting it, the channel will be terminated, but the device termination will not be recognized and the NDT bit will not be set.

5.6.3 Error Termination

There are two basic types of errors that can cause a channel operation to be terminated. Internal errors are those generated by the DMAC either during start up by the system processor or while it is bus master. External errors are those generated by external hardware in response to abnormal conditions. When a channel operation is terminated due to any of these errors, the COC and ERR bits in the CSR will be set and the ERROR CODE field in the CER will indicate what error caused the termination. Table 5-6 summarizes all of the error conditions that can cause channel termination and the source of each error.

Table 5-6. Channel Error Condition Summary

Error Type	Cause
Configuration Error	Any undefined reserved bit pattern, MC68440 reserved option, or illegal device/operand size combination is programmed into a channel and an attempt is made to set the STR bit.
Operation Timing Error	An attempt is made to set STR with ACT, COC, BTC, NDT, or ERR set. An attempt is made to set CNT without setting STR simultaneously or if ACT is not set. An attempt is made to set CNT with BTC and ACT set. An attempt is made to write to the DCR, OCR, SCR, MAR, MFCR, MTCR, DAR, or DFCR with STR or ACT set.

Address Error	$\overline{\text{CS}}$ or $\overline{\text{IACK}}$ is asserted when the DDMA is acting as bus master.
	A word bus cycle is attempted to an odd address. This may occur during the following operations:
	    Any word or long-word operand transfer.
	    An access to the descriptor array during chained operations.

Bus Error	A DMAC bus cycle is terminated with a bus error exception code.
Count Error	The MTCR contains $0000 and an attempt is made to start the channel while not in a chaining mode. $0000 is loaded into the MTCR by a continue or chain operation. The BTCR contains $0000 and an attempt is made to start the channel in the sequential array chaining mode.
External Abort	The PCL line is programmed as an abort input and the PCT bit is set with the ACT bit set.
Software Abort	The SAB bit is set by the MPU.

1-882

5.6.3.1 INTERNAL ERRORS. There are five types of internal errors that can occur: configuration errors, operation timing errors, address errors, count errors, and software abort. Software abort has already been discussed (refer to **5.4.5.2 SOFTWARE ABORT**), and the following paragraphs describe each of the other four.

5.6.3.1.1 Configuration Error. When a channel is configured incorrectly and the STR bit is set by the system processor, a configuration error will occur and the channel will not be started.

5.6.3.1.2 Operation Timing Error. When a write access is attempted to certain registers or bits within registers while a channel is active or when certain status bits are set, an operation timing error will occur and the channel operation will be aborted if the channel is active.

5.6.3.1.3 Address Error. When the DMAC is acting as the bus master and $\overline{CS}$ or $\overline{IACK}$ is asserted, or a word or long-word operation is attempted to an odd address, an address error will be signaled and the channel operation will be aborted. If $\overline{CS}$ or $\overline{IACK}$ is asserted during a DMA access, no internal registers will be modified by the access and the operation of the other channel will not be affected. If a word access to an odd address is attempted, the address error will be detected before the DMAC executes an operand transfer and no DMAC internal registers will be affected (other than the CSR).

5.6.3.1.4 Count Error. When a channel is started or a continue operation is performed and the MTCR value is zero, a count error will be signaled and the channel operation terminated. If the MTCR is zero when the STR bit is set, the count error will be detected before the channel is started and no transfer requests will be recognized. If the BTCR is zero when a block transfer is terminated with the CNT bit set, a count error is detected, and the channel operation is terminated whether or not a transfer request is pending. In the sequential array chaining mode of operation, if the BTCR contains zero when the channel is started, a count error will be detected and the operation terminated before the array is accessed. If the MTCR is loaded with zero by a chaining operation, the count error will cause the channel operation to be terminated before the BAR and BTCR are updated.

5.6.3.2 EXTERNAL ERRORS. There are two types of error conditions that are generated by external hardware: bus error and hardware abort. The behavior of the DMAC when either of these conditions is signaled has been previously discussed (refer to **5.4.3.3 BUS ERROR** and **5.4.6 HARDWARE ABORT**).

SECTION 6
ELECTRICAL SPECIFICATIONS

This section contains electrical specifications and associated timing information for the MC68450 direct memory access controller.

6.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to $+7.0$	V
Input Voltage	V_{in}	-0.3 to $+7.0$	V
Operating Temperature Range	T_A	0 to 70	°C
Storage Temperature	T_{stg}	-55 to 150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

6.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Symbol	Value	Unit
Thermal Resistance	θ_{JA}		θ_{JC}		°C/W
Ceramic (L/LC)		30		15*	
Plastic (P)		30		15*	
Pin Grid Array (R/RC)		30		15	

*Estimated
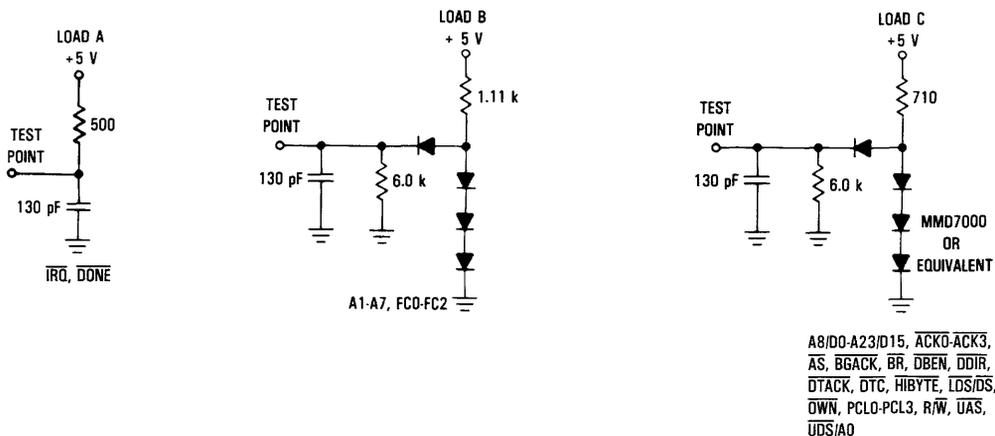
Figure 6-1. Test Loads

6.3 POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \tag{1}$$

where:

T_A = Ambient Temperature, °C
θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W
P_D = $P_{INT} + P_{I/O}$
P_{INT} = $I_{CC} \times V_{CC}$, Watts — Chip Internal Power
$P_{I/O}$ = Power Dissipation on Input and Output Pins, Watts — User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \tag{2}$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273°C) + \theta_{JA} \cdot P_D^2 \tag{3}$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

The total thermal resistance of a package (θ_{JA}) can be separated into two components, θ_{JC} and θ_{CA}, representing the barrier to heat flow from the semiconductor junction to the package (case) surface (θ_{JC}) and from the case to the outside ambient (θ_{CA}). These terms are related by the equation:

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \tag{4}$$

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling and thermal convection. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC}. Substitution of θ_{JC} for θ_{JA} in equation (1) will result in a lower semiconductor junction temperature.

Values for thermal resistance presented in this document, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843, "Thermal Resistance Measurement Method for MC68XX Microcomponent Devices," and are provided for design purposes only. Thermal measurements are complex and dependent on procedure and setup. User derived values for thermal resistance may differ.

1-1184

6.4 DC ELECTRICAL CHARACTERISTICS
($V_{CC} = 5 \text{ V} \pm 5\%$, GND = 0 V, $T_A = 0$ to 70°C, unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	2.0	V_{CC}	V
Input Low Voltage	V_{IL}	GND − 0.3	0.8	V
Input Leakage Current $\overline{CS}$, $\overline{IACK}$, $\overline{BG}$, CLK, $\overline{BEC0}$-$\overline{BEC2}$, $\overline{REQ0}$-$\overline{REQ3}$	I_{in}	—	10	μA
Three-State (Off-State) Input Current A1-A7, D0/A8-D15/A23, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{UAS}$, $\overline{DTACK}$, $\overline{BGACK}$, $\overline{OWN}$, DTC, HIBYTE, $\overline{DDIR}$, $\overline{DBEN}$, FC0-FC2, PCL0-PCL3	I_{TSI}	—	20	μA
Input Capacitance ($V_{in} = 0$ V, $T_A = 25$°C, f = 1 MHz)	C_{in}	—	15	pF
Open-Drain (Off-State) Input Current $\overline{IRQ}$, $\overline{DONE}$	I_{DD}	—	20	μA
Output High Voltage $I_{OH} = -400$ μA A1-A7, D0/A8-D15/A23, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{UAS}$, $\overline{DTACK}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{OWN}$, DTC, HIBYTE, $\overline{DDIR}$, $\overline{DBEN}$, $\overline{ACK0}$-$\overline{ACK3}$, PCL0-PCL3, FC0-FC2	V_{OH}	2.4	—	V
Output Low Voltage $I_{OL} = 3.2$ mA A1-A7, FC0-FC2 $I_{OL} = 5.3$ mA D0/A8-D15/A23, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{DTACK}$, $\overline{BR}$, $\overline{OWN}$, DTC, HIBYTE, $\overline{DDIR}$, $\overline{DBEN}$, $\overline{ACK0}$-$\overline{ACK3}$, $\overline{UAS}$, PCL0-PCL3, $\overline{BGACK}$ $I_{OL} = 8.9$ mA $\overline{IRQ}$, $\overline{DONE}$	V_{OL}	—	0.5	V
Power Dissipation at 25°C (Frequency = 8 MHz)	P_D	—	2.0	W

6.5 AC ELECTRICAL SPECIFICATIONS—CLOCK TIMING (See Figure 6-2)

Parameter	Symbol	8 MHz		10 MHz		Unit
		Min	Max	Min	Max	
Frequency of Operation	f	2	8	2	10	MHz
Clock Period	t_{cyc}	125	500	100	500	ns
Clock Width Low	t_{CL}	55	250	45	250	ns
Clock Width High	t_{CH}	55	250	45	250	ns
Clock Fall Time	t_{Cf}	—	10	—	10	ns
Clock Rise Time	t_{Cr}	—	10	—	10	ns
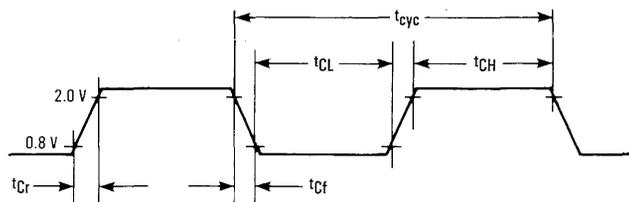
NOTE:
Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 volt and 2.0 volts.

1-634

Figure 6-2. Clock Input Timing Diagram

6.6 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES

(V_{CC} = 15 V ± 5%, GND = 0 V, T_A = 0°C to 70°C, unless otherwise noted)
(See Figures 6-4 through 6-9)

No.	Parameter	Symbol	8 MHz		10 MHz		Unit
			Min	Max	Min	Max	
1	Clock Period	t_{cyc}	125	500	100	500	ns
2	Clock Width Low	t_{CL}	55	250	45	250	ns
3	Clock Width High	t_{CH}	55	257	45	250	ns
4	Clock Fall Time	t_{Cf}	—	10	—	10	ns
5	Clock Rise Time	t_{Cr}	—	10	—	10	ns
6	Asynchronous Input Setup Time	t_{ASI}	20	—	15	—	ns
7	Data In to $\overline{DBEN}$ Low	t_{DIDBL}	0	—	0	—	ns
8	$\overline{DTACK}$ Low to Data In Invalid	t_{DTLDI}	0	—	0	—	ns
9	Address In to $\overline{AS}$ In Low	t_{AIASL}	0	—	0	—	ns
10	$\overline{AS}$ In High to Address In Invalid	t_{SIHAIV}	0	—	0	—	ns
11	Clock High to $\overline{DDIR}$ Low	t_{CHDRL}	—	70	—	60	ns
12	Clock High to $\overline{DDIR}$ High	t_{CHDRH}	—	70	—	60	ns
13	$\overline{DS}$ In High to $\overline{DDIR}$ High Impedance	t_{DSHDRZ}	—	120	—	110	ns
14	Clock Low to $\overline{DBEN}$ Low	t_{CLDBL}	—	70	—	60	ns
15	Clock Low to $\overline{DBEN}$ High	t_{CLDBH}	—	70	—	60	ns
16	$\overline{DS}$ In High to $\overline{DBEN}$ High Impedance	t_{DSHDBZ}	—	120	—	110	ns
17	Clock High to Data Out Valid (MPU Read)	t_{CHDVM}	—	180	—	160	ns
18	$\overline{DS}$ In High to Data Out Invalid	t_{DSHDZn}	0	—	0	—	ns
19	$\overline{DS}$ In High to Data High Impedance	t_{DSHDZ}	—	120	—	110	ns
20	Clock Low to $\overline{DTACK}$ Low	t_{CLDTL}	—	70	—	60	ns
21	$\overline{DS}$ In High to $\overline{DTACK}$ High	t_{DSHDTH}	—	110	—	110	ns
22	$\overline{DTACK}$ Width High	t_{DTH}	10	—	10	—	ns
23	$\overline{DS}$ In High to $\overline{DTACK}$ High Impedance	t_{DSHDTZ}	—	180	—	160	ns
24	$\overline{DTACK}$ Low to $\overline{DS}$ In High	t_{DTLDSH}	0	—	0	—	ns
25	$\overline{REQ}$ Width Low	t_{REQL}	2.0	—	2.0	—	Clk. Per.
26	$\overline{REQ}$ Low to $\overline{BR}$ Low	t_{RELBRL}	250	—	200	—	ns
27	Clock High to $\overline{BR}$ Low	t_{CHBRL}	—	70	—	60	ns
28	Clock High to $\overline{BR}$ High	t_{CHBRH}	—	70	—	60	ns
29	$\overline{BG}$ Low to $\overline{BGACK}$ Low	t_{BGLBL}	4.5	—	4.5	—	Clk. Per.
31	MPU Cycle End ($\overline{AS}$ In High) to $\overline{BGACK}$ Low	t_{ASHBL}	4.5	5.5	4.5	5.5	Clk. Per.
32	$\overline{REQ}$ Low to $\overline{BGACK}$ Low	t_{REQLBL}	12	—	12	—	Clk. Per.
33	Clock High to $\overline{BGACK}$ Low	t_{CHBL}	—	70	—	60	ns
34	Clock High to $\overline{BGACK}$ High	t_{CHBH}	—	70	—	60	ns
35	Clock Low to $\overline{BGACK}$ High Impedance	t_{CLBZ}	—	80	—	70	ns
36	Clock High to FC Valid	t_{CHFCV}	—	100	—	90	ns
37	Clock High to Address Valid	t_{CHAV}	—	120	—	110	ns
38	Clock High to Address/FC/Data High Impedance	t_{CHAZx}	—	100	—	100	ns
39	Clock Higrs to Address/FC/Data Invalid	t_{CHAZn}	0	—	0	—	ns
40	Clock Low to Address High Impedance (Read)	t_{CLAZ}	—	100	—	90	ns
41	Clock High to $\overline{UAS}$ Low	t_{CHUL}	—	70	—	60	ns
42	Clock High to $\overline{UAS}$ High	t_{CHUH}	—	70	—	60	ns
43	Clock Low to $\overline{UAS}$ High Impedance	t_{CLUZ}	—	80	—	70	ns
44	$\overline{UAS}$ High to Address Invalidance	t_{UHAI}	30	—	20	—	ns
45	Clock High to $\overline{AS}$, $\overline{DS}$ Low	t_{CHSL}	—	60	—	55	ns
46	Clock Low to $\overline{DS}$ Low (Write)	t_{CLDSL}	—	60	—	55	ns

6.6 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (Continued)

No.	Parameter	Symbol	8 MHz		10 MHz		Unit
			Min	Max	Min	Max	
47	Clock Low to $\overline{AS}$, $\overline{DS}$ High	t_{CLSH}	—	70	—	60	ns
48	Clock Low to $\overline{AS}$, $\overline{DS}$ High Impedance	t_{CLSZ}	—	80	—	70	ns
49	$\overline{AS}$ Width Low	t_{ASL}	255	—	195	—	ns
50	$\overline{DS}$ Width Low	t_{DSL}	255	—	190	—	ns
51	$\overline{AS}$, $\overline{DS}$ Width High	t_{SH}	150	—	105	—	ns
52	Address/FC Valid to $\overline{AS}$, $\overline{DS}$ Low (Read)	t_{AVSL}	30	—	20	—	ns
53	$\overline{AS}$, $\overline{DS}$ High to Address/FC/Data Invalid	t_{SHAZ}	30	—	20	—	ns
54	Clock High to R/$\overline{W}$ Low	t_{CHRL}	—	70	—	60	ns
55	Clock High to R/$\overline{W}$ High	t_{CHRH}	—	70	—	60	ns
56	Clock Low to R/$\overline{W}$ High Impedance	t_{CLRZ}	—	80	—	70	ns
57	Address/FC Valid to R/$\overline{W}$ Low	t_{AVRL}	20	—	10	—	ns
58	R/$\overline{W}$ Low to $\overline{DS}$ Low (Write)	t_{RLSL}	120	—	90	—	ns
59	$\overline{DS}$ High to R/$\overline{W}$ High	t_{SHRH}	40	—	20	—	ns
60	Clock Low to $\overline{OWN}$ Low	t_{CLOL}	—	70	—	60	ns
61	Clock Low to $\overline{OWN}$ High	t_{CLOH}	—	70	—	60	ns
62	Clock High to $\overline{OWN}$ High Impedance	t_{CHOZ}	—	80	—	70	ns
63	$\overline{OWN}$ Low to $\overline{BGACK}$ Low	t_{OLBL}	30	—	20	—	ns
64	$\overline{BGACK}$ High to $\overline{OWN}$ High	t_{BHOH}	30	—	20	—	ns
65	$\overline{OWN}$ Low to $\overline{UAS}$ Low	t_{OLUL}	30	—	20	—	ns
66	Clock High to $\overline{ACK}$ Low (Read)	t_{CHACL}	—	70	—	60	ns
67	Clock Low to $\overline{ACK}$ Low (Write)	t_{CLACL}	—	70	—	60	ns
68	Clock High to $\overline{ACK}$ High	t_{CHACH}	—	70	—	60	ns
69	$\overline{ACK}$ Low to $\overline{DS}$ Low	t_{ACLDSL}	100	—	80	—	ns
70	$\overline{DS}$ High to $\overline{ACK}$ High	t_{DSHACH}	30	—	20	—	ns
71	Clock High to $\overline{HIBYTE}$ Low	t_{CHHIL}	—	70	—	60	ns
72	Clock Low to $\overline{HIBYTE}$ Low	t_{CLHIL}	—	70	—	60	ns
73	Clock High to $\overline{HIBYTE}$ High	t_{CHHIH}	—	70	—	60	ns
74	Clock Low to $\overline{HIBYTE}$ High Impedance	t_{CLHIZ}	—	80	—	70	ns
75	Clock High to $\overline{DTC}$ Low	t_{CHDTL}	—	70	—	60	ns
76	Clock High to $\overline{DTC}$ High	t_{CHDTH}	—	70	—	60	ns
77	Clock Low to $\overline{DTC}$ High Impedance	t_{CLDTZ}	—	80	—	70	ns
78	$\overline{DTC}$ Width Low	t_{DTCL}	105	—	80	—	ns
79	$\overline{DTC}$ Low to $\overline{DS}$ High	t_{DTLDH}	30	—	20	—	ns
80	Clock High to $\overline{DONE}$ Low	t_{CHDOL}	—	70	—	60	ns
81	Clock Low to $\overline{DONE}$ Low	t_{CLDOL}	—	70	—	60	ns
82	Clock High to $\overline{DONE}$ High	t_{CHDOH}	—	130	—	120	ns
83	Clock Low to $\overline{DDIR}$ High Impedance	t_{CLDRZ}	—	80	—	70	ns
84	Clock Low to $\overline{DBEN}$ High Impedance	t_{CLDBZ}	—	80	—	70	ns
85	$\overline{DDIR}$ Low to $\overline{DBEN}$ Low	t_{DRLDBL}	30	—	20	—	ns
86	$\overline{DBEN}$ High to $\overline{DDIR}$ High	t_{DBHDRH}	30	—	20	—	ns
87	$\overline{DBEN}$ Low to Address/Data High Impedance	t_{DBLAZ}	—	17	—	17	ns
88	Clock Low to PCL Low (1/8 Clock)	t_{CLPL}	—	70	—	60	ns
89	Clock Low to PCL High (1/8 Clock)	t_{CLPH}	—	70	—	60	ns
90	PCL Width Low (1/8 Clock)	t_{PCLL}	4.0	—	4.0	—	Clk. Per.
91	$\overline{DTACK}$ Low to Data In (Setup Time)	t_{DALDI}	—	150	—	115	ns
92	$\overline{DS}$ High to Data Invalid (Hold Time)	t_{SHDI}	0	—	0	—	ns

6.6 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (Continued)

No.	Parameter	Symbol	8 MHz		10 MHz		Unit
			Min	Max	Min	Max	
93	$\overline{DS}$ High to $\overline{DTACK}$ High	t_{SHDAH}	0	120	0	90	ns
94	Data Out Valid to $\overline{DS}$ Low	t_{DOSL}	0	—	0	—	ns
95	Data In to Clock Low (Setup Time)	t_{DICL}	15	—	15	—	ns
96	$\overline{BEC}$ Low to $\overline{DTACK}$ Low	t_{BECDAL}	50	—	50	—	ns
97	$\overline{BEC}$ Width Low	t_{BECL}	2.0	—	2.0	—	Clk. Per.
98	Clock High to $\overline{IRQ}$ Low	t_{CHIRL}	—	70	—	60	ns
99	Clock High to $\overline{IRQ}$ High Impedance	t_{CHIRH}	—	130	—	120	ns
100	PCL (as $\overline{READY}$) In to $\overline{DTC}$ Low (Read)	t_{RALDTL}	145	—	120	—	ns
101	PCL (as $\overline{READY}$) In to $\overline{DS}$ Low (Write)	t_{RALDSL}	205	—	170	—	ns
102	$\overline{DS}$ High to PCL (as $\overline{READY}$) High	t_{DSHRAH}	0	120	0	90	ns
103	$\overline{DONE}$ In Low to $\overline{DTACK}$ Low	t_{DOLDAL}	50	—	50	—	ns
104	$\overline{DS}$ High to $\overline{DONE}$ In High	t_{DSHDOH}	0	120	0	90	ns

Timing Diagrams (Figures 6-4 through 6-9) are located on foldout pages at the end of this document.

DEVICE CONTROL REGISTER

04 | 7 XRM 6 | 5 DTYP 4 | 3 DPS | 2 PCL 1 0

000 STATUS
001 INTERRUPT
010 START PULSE
011 ABORT
100 (MC68440)
101 (UNDEFINED)
110 (UNDEFINED)
111 (UNDEFINED)

0 8-BIT
1 16-BIT

00 EXPLICIT M68000
01 EXPLICIT M6800
10 IMPLICIT W/ACK
11 IMPLICIT W/ACK AND RDY

00 BURST
01 (UNDEFINED)
10 CYCLE STEAL WITHOUT HOLD
11 CYCLE STEAL WITH HOLD

OPERATION CONTROL REGISTER

05 | 7 DIR | 6 0 | 5 SIZE 4 | 3 CHAIN 2 | 1 REQG 0

00 INTERNAL LIMITED RATE
01 INTERNAL MAXIMUM RATE
10 EXTERNAL
11 AUTO START, EXTERNAL

00 DISABLED
01 (UNDEFINED)
10 ARRAY CHAINING
11 LINKED ARRAY CHAINING

00 BYTE
01 WORD
10 LONG WORD
11 BYTE, NO PACKING

0 MEMORY TO DEVICE
1 DEVICE TO MEMORY

SEQUENCE CONTROL REGISTER

06 | 7 0 | 6 0 | 5 0 | 4 0 | 3 MAC 2 | 1 DAC 0

00 NO COUNT
01 COUNT UP
10 COUNT DOWN
11 (UNDEFINED)

00 NO COUNT
01 COUNT UP
10 COUNT DOWN
11 (UNDEFINED)

CHANNEL CONTROL REGISTER

07 | 7 STR | 6 CNT | 5 HLT | 4 SAB | 3 INT | 2 0 | 1 0 | 0 0

INTERRUPT ENABLE
SOFTWARE ABORT
SOFTWARE HALT
CONTINUE OPERATION
START CHANNEL

CHANNEL STATUS REGISTER

00 | 7 COC | 6 BTC | 5 NDT | 4 ERR | 3 ACT | 2 0 | 1 PCT | 0 PCS

PCL LEVEL
PCL TRANSITION
CHANNEL ACTIVE
ERROR
NORMAL DEVICE TERMINATION
BLOCK TRANSFER COMPLETE
CHANNEL OPERATION COMPLETE

CHANNEL ERROR REGISTER

01 | 7 0 | 6 | 5 0 | 4 ERROR 3 2 1 0

00000 NO ERROR
00001 CONFIGURATION ERROR
00010 OPERATION TIMING ERROR
00011 (UNDEFINED)
001rr ADDRESS ERROR
010rr BUS ERROR
011rr COUNT ERROR
10000 EXTERNAL ABORT
10001 SOFTWARE ABORT
10010 }
• } (UNDEFINED)
11111 }
— rr
00 (MC68440)
01 MAR/MTCR
10 DAR
11 BAR/BTCR

CHANNEL PRIORITY REGISTER

2D | 7 0 | 6 0 | 5 0 | 4 0 | 3 0 | 2 0 | 1 CP 0

00 PRIORITY 0 (HIGHEST)
01 PRIORITY 1
10 PRIORITY 2
11 PRIORITY 3 (LOWEST)

GENERAL CONTROL REGISTER

FF | 7 0 | 6 0 | 5 0 | 4 0 | 3 BT 2 | 1 BR 0

00 50.00%
01 25.00%
10 12.50%
11 6.25%

00 16 CLOCKS
01 32 CLOCKS
10 64 CLOCKS
11 128 CLOCKS

NORMAL INTERRUPT VECTOR REGISTER

25 | 7 0

ERROR INTERRUPT VECTOR REGISTER

27 | 7 0

MEMORY TRANSFER COUNT REGISTER

0A | 15 0

BASE TRANSFER COUNT REGISTER

1A | 15 0

DEVICE FUNCTION CODE REGISTER*

31 | 3 ... 0

DEVICE ADDRESS REGISTER

14 | 31 0

MEMORY FUNCTION CODE REGISTER

29 | 3 ... 0

MEMORY ADDRESS REGISTER

0C | 31 0

BASE FUNCTION CODE REGISTER

39 | 3 ... 0

BASE ADDRESS REGISTER

1C | 31 0

CHANNEL BASE ADDRESSES: CH0-00 CH2-80
CH1-40 CH3-C0

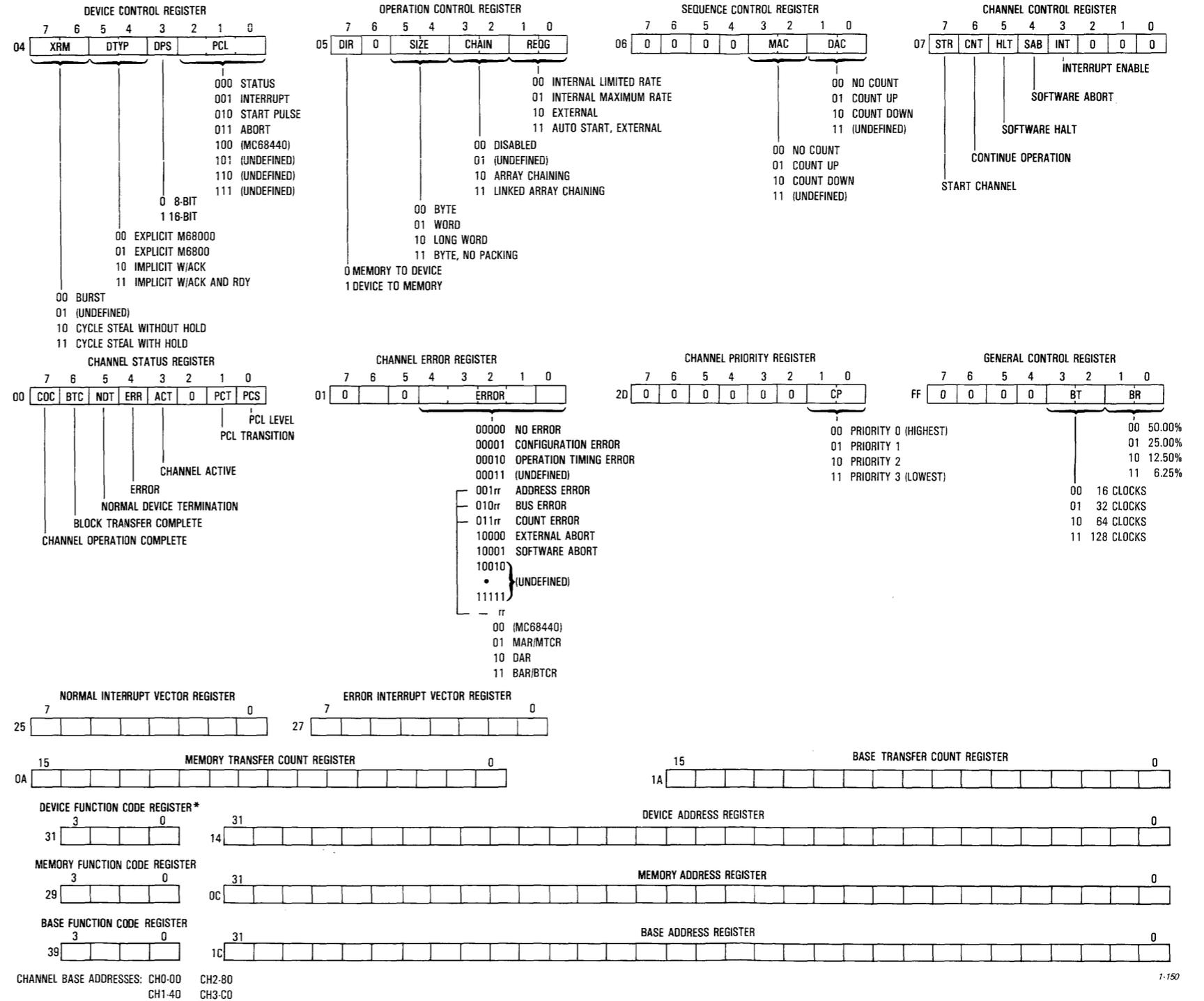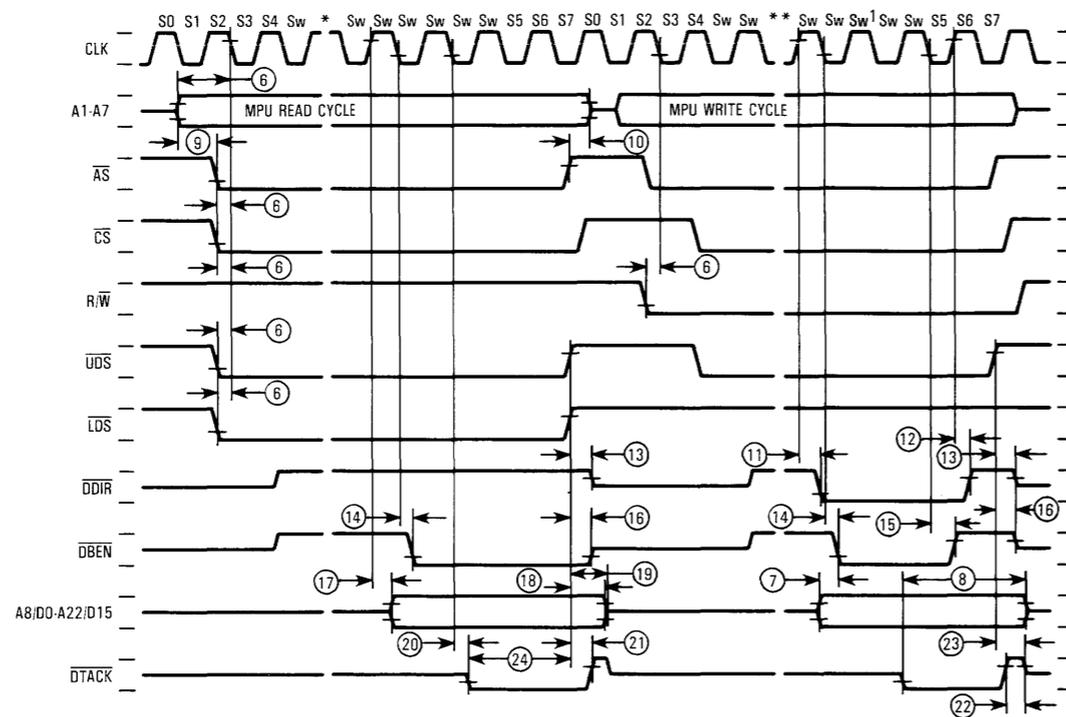*Bits 7-4 read as zeros and are ignored on writes.

1-150

Figure 3-2. Register Summary

Figure 3-2. Register Summary

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.
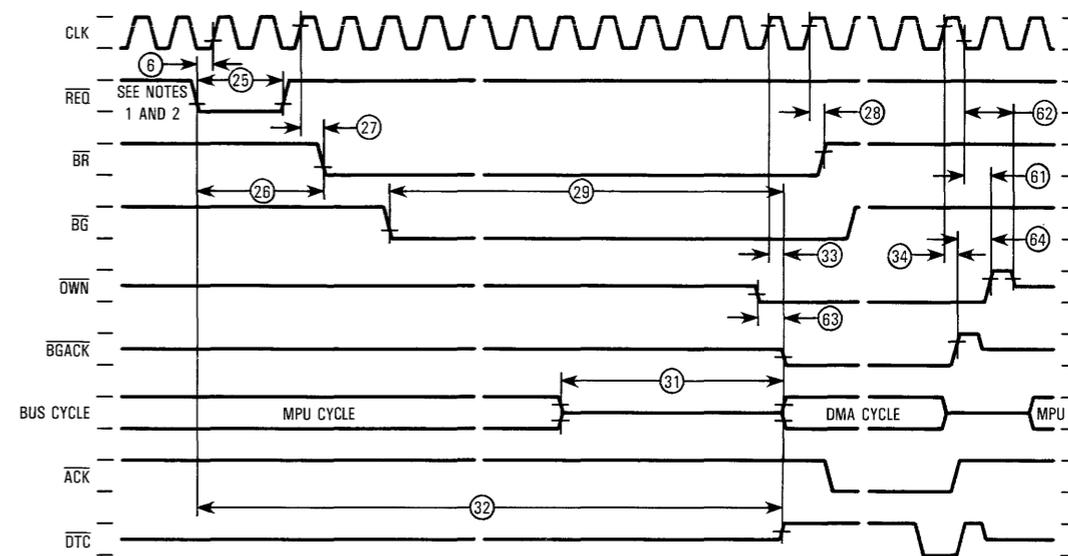
These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

1-885

*10 Additional Wait Cycles
**8 Additional Wait Cycles

NOTES:
1. Data is latched at the end of this clock.
2. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-3. MPU Read/Write Timing Diagram

1-886

NOTES:
1. $\overline{REQ}$ is sampled on the rising edge of CLK in cycle steal and burst modes.
2. $\overline{BR}$ will not be asserted while any $\overline{BEC}$ exception condition exists, or when $\overline{CS}$ or $\overline{IACK}$ is asserted.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.
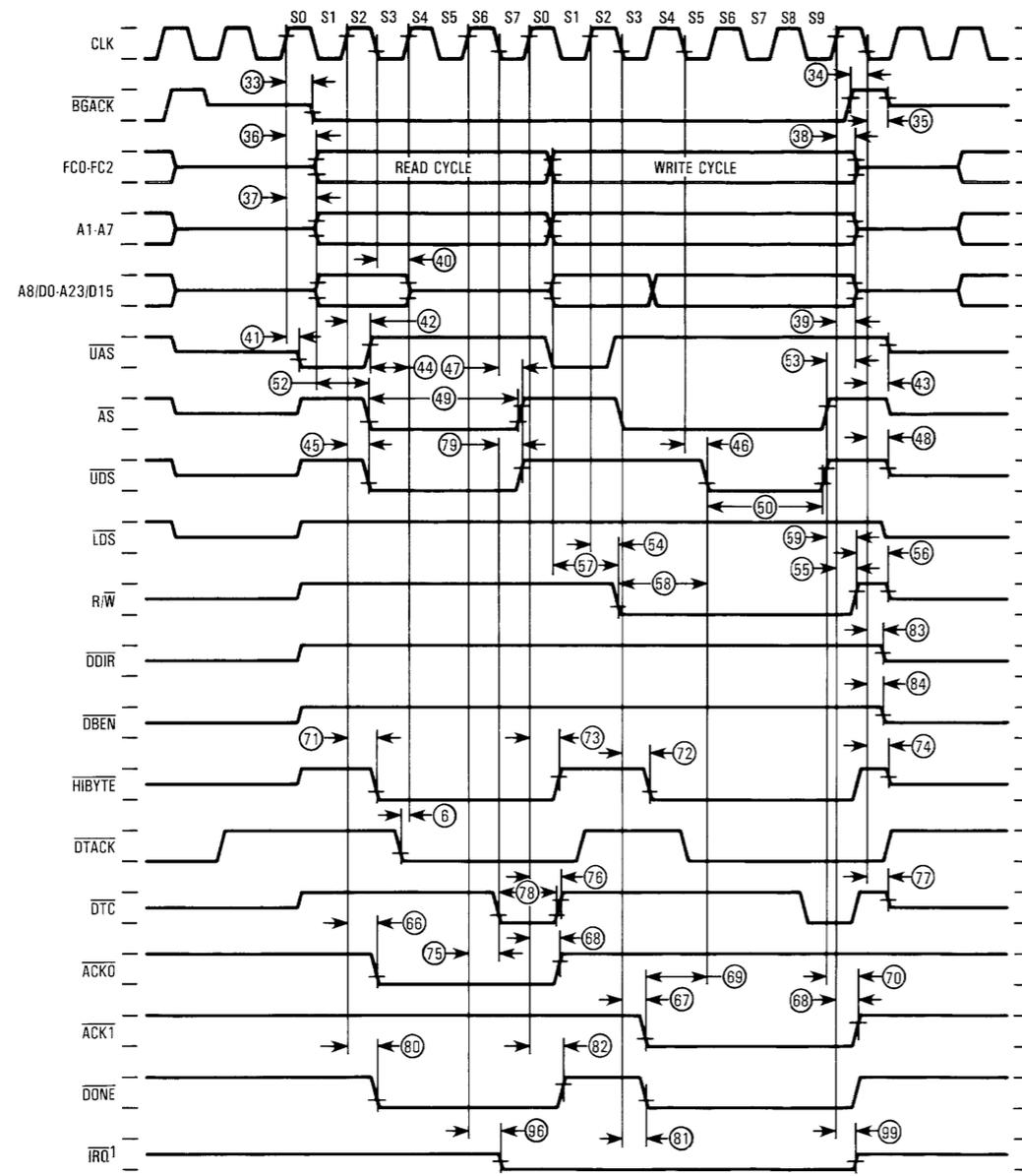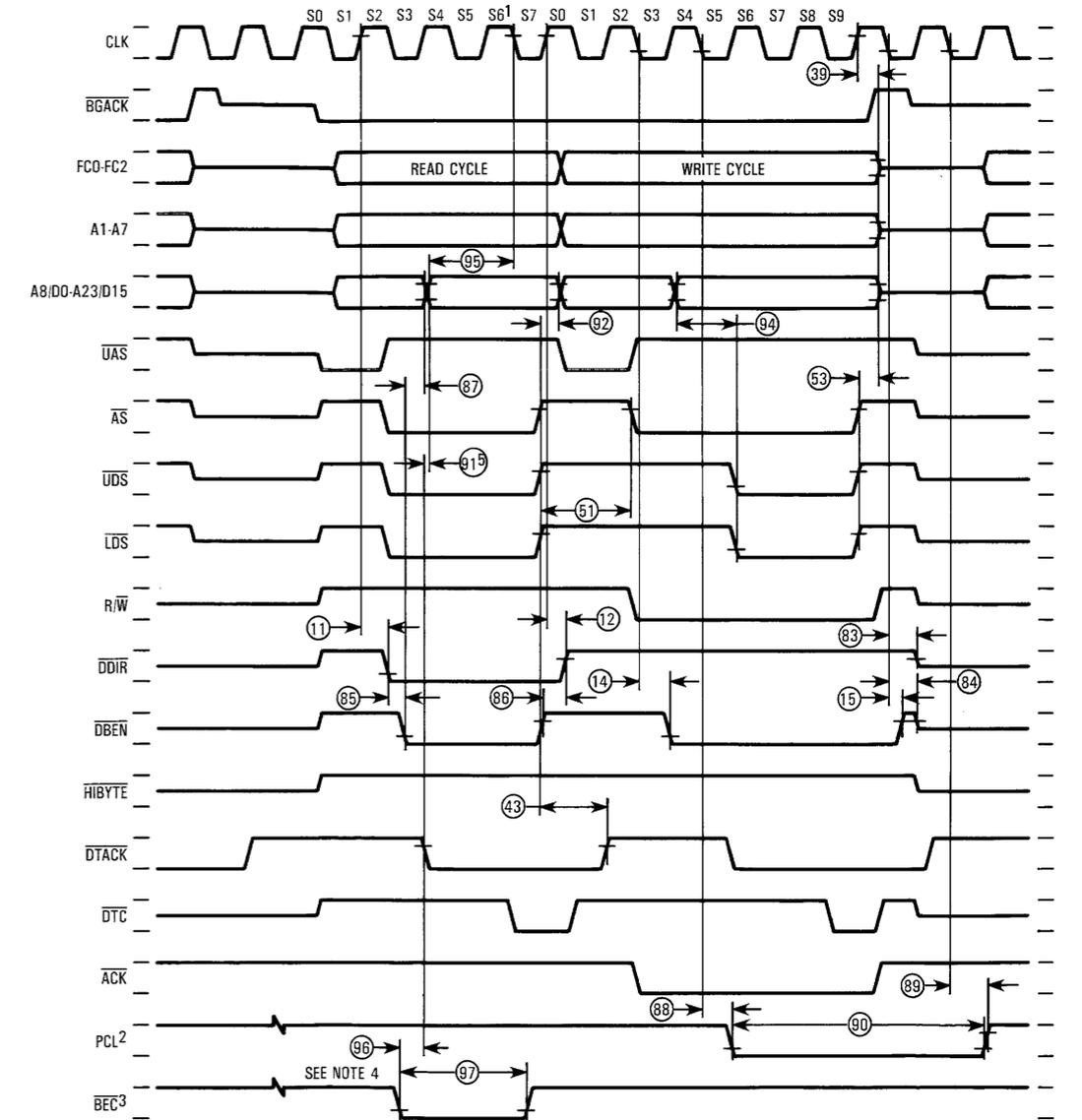
Figure 6-4. Bus Arbitration Timing Diagram

Figure 6-3. MPU Read/Write Timing Diagram

Figure 6-4. Bus Arbitration Timing Diagram

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

Figure 6-5. DMA Read/Write Timing Diagram (Single Cycle)

NOTES:
1. This timing is not directly related to the DMA read/write (single cycle) sequence.
2. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-6. DMA Read/Write Timing Diagram (Dual Cycle)

NOTES:
1. Data is latched at the end of clock S6.
2. Timing is not directly related to the DMA read/write (dual cycle) sequence and is only applicable when the start pulse mode is selected.
3. Timing is applicable when a bus exception occurs.
4. If specification number 6 is satisfied for both $\overline{\text{DTACK}}$ and $\overline{\text{BEC}}$, specification number 96 may be 0 nanoseconds.
5. If the propagation delay of the external bidirectional buffer is less than 17 nanoseconds, a conflict may occur between the address output of the DMAC and the system data bus. In this case, the output of $\overline{\text{DBEN}}$ must be delayed externally.
6. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.
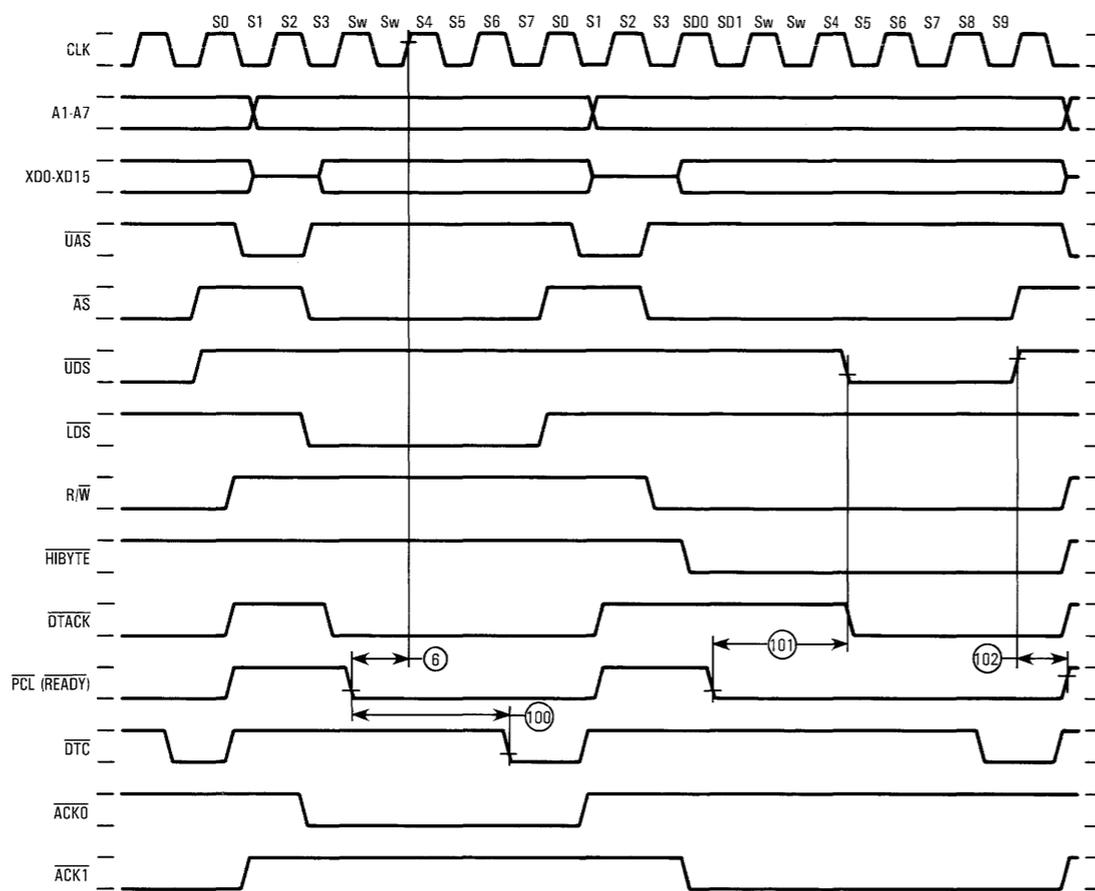
Figure 6-5. DMA Read/Write Timing Diagram (Single Cycle)

Figure 6-6. DMA Read/Write Timing Diagram (Dual Cycle)

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.


1-889

NOTE:
1. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-7. DMA Read/Write Timing Diagram (Single Cycle with PCL as READY)

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.
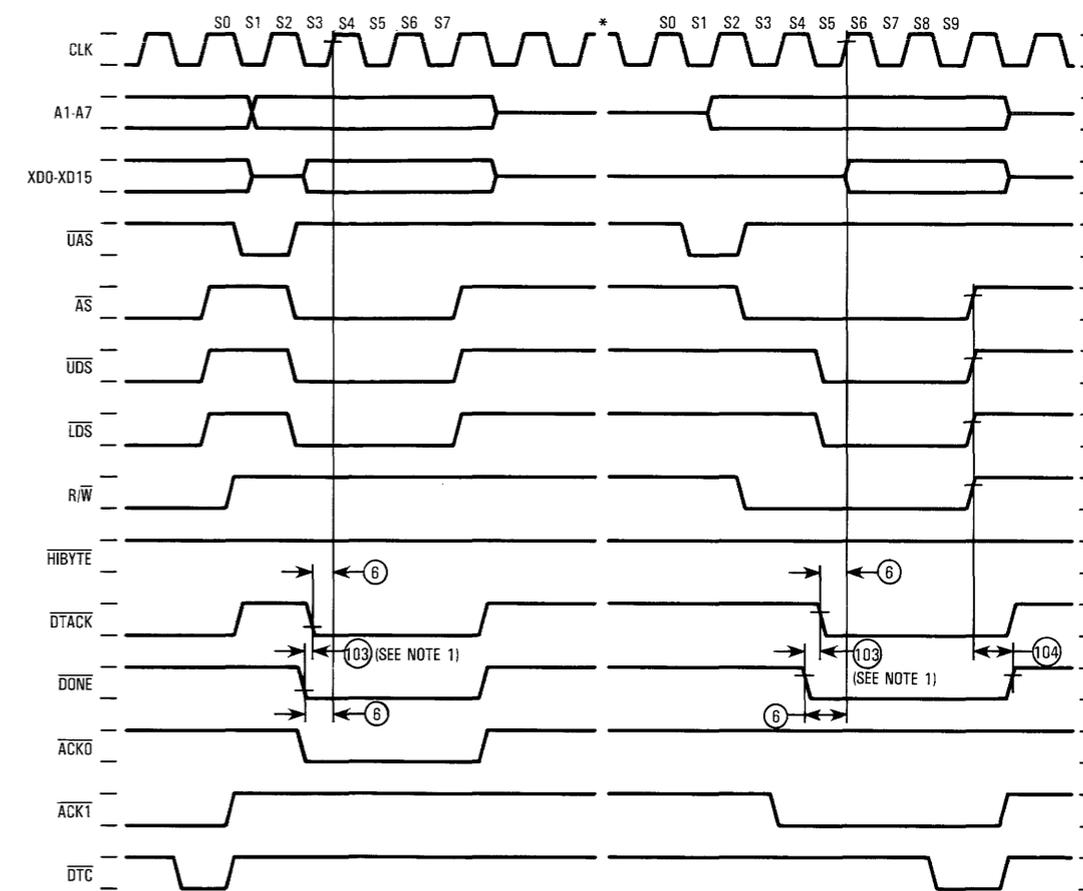
*3 to 9 Additional Clocks

1-890

NOTES:
1. If specification number 6 is satisfied for both DTACK and DONE, specification number 103 may be 0 nanoseconds.
2. The setup time for asynchronous inputs BG, BGACK, CS, IACK, AS, UDS, LDS, and R/W guarantees their recognition at the next falling edge of the clock. The setup time for BEC0-BEC2, REQ0-REQ3, PCL0-PCL3, DTACK, and DONE guarantees their recognition at the next rising edge of the clock.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-8. DONE Input Timing Diagram

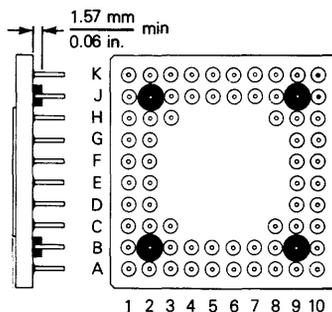Figure 6-8. $\overline{\text{DONE}}$ Input Timing Diagram

Figure 6-7. DMA Read/Write Timing Diagram (Single Cycle with PCL as $\overline{\text{READY}}$)

SECTION 7
ORDERING INFORMATION

The following information should be used as a guide when ordering the MC68450.

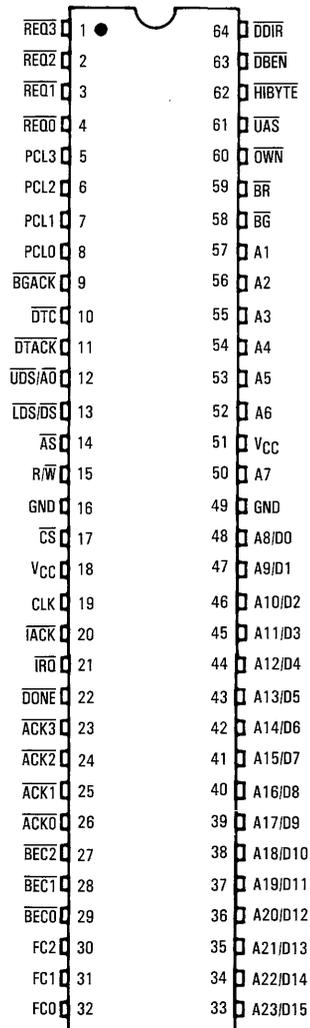| Package Type	Frequency (MHz)	Temperature	Order Number
Ceramic	8.0	0°C to 70°C	MC68450L8
L Suffix	10.0	0°C to 70°C	MC68450L10
	12.5	0°C to 70°C	MC68450L12
Plastic	8.0	0°C to 70°C	MC68450P8
P Suffix	10.0	0°C to 70°C	MC68450P10
	12.5	0°C to 70°C	MC68450P12
Pin Grid Array	8.0	0°C to 70°C	MC68450R8
(with Standoffs)	10.0	0°C to 70°C	MC68450R10
R Suffix	12.5	0°C to 70°C	MC68450R12
(See Figure 7-1)			
Pin Grid Array	8.0	0°C to 70°C	MC68450RC8
with Gold Lead	10.0	0°C to 70°C	MC68450RC10
Finish (without	12.5	0°C to 70°C	MC68450RC12
Standoffs)			
RC Suffix			

Figure 7-1. 68-Pin Grid Array
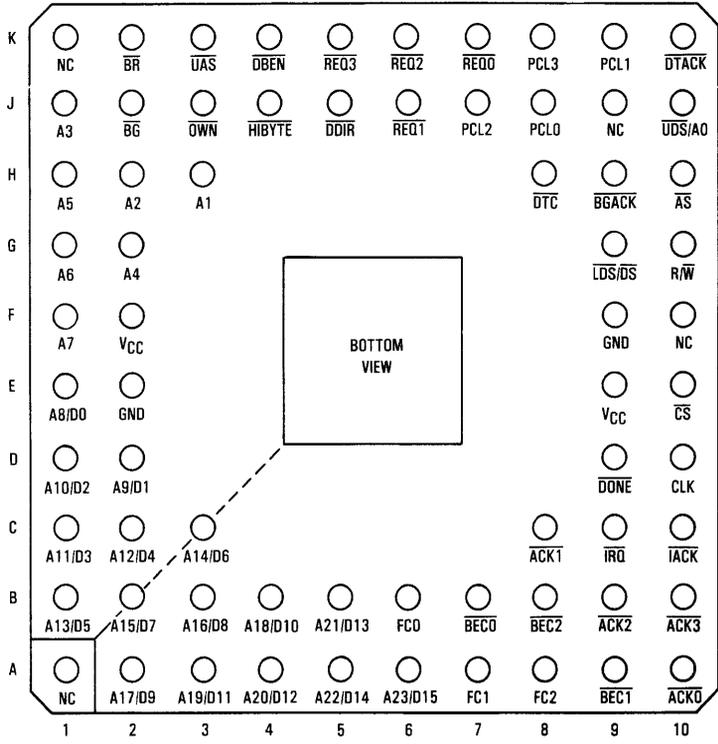**(Pins Soldered Dipped with Plastic Standoff)**

SECTION 8
MECHANICAL DATA

This section contains the pin assignments and package dimensions for the MC68450.

8.1 PIN ASSIGNMENTS

8.1.1 68-Pin Dual-In-Line Package

$\overline{REQ3}$	1 ●	64	$\overline{DDIR}$
$\overline{REQ2}$	2	63	$\overline{DBEN}$
$\overline{REQ1}$	3	62	$\overline{HIBYTE}$
$\overline{REQ0}$	4	61	$\overline{UAS}$
PCL3	5	60	$\overline{OWN}$
PCL2	6	59	$\overline{BR}$
PCL1	7	58	$\overline{BG}$
PCL0	8	57	A1
$\overline{BGACK}$	9	56	A2
$\overline{DTC}$	10	55	A3
$\overline{DTACK}$	11	54	A4
$\overline{UDS/A0}$	12	53	A5
$\overline{LDS/DS}$	13	52	A6
$\overline{AS}$	14	51	V_{CC}
$R/\overline{W}$	15	50	A7
GND	16	49	GND
$\overline{CS}$	17	48	A8/D0
V_{CC}	18	47	A9/D1
CLK	19	46	A10/D2
$\overline{IACK}$	20	45	A11/D3
$\overline{IRQ}$	21	44	A12/D4
$\overline{DONE}$	22	43	A13/D5
$\overline{ACK3}$	23	42	A14/D6
$\overline{ACK2}$	24	41	A15/D7
$\overline{ACK1}$	25	40	A16/D8
$\overline{ACK0}$	26	39	A17/D9
$\overline{BEC2}$	27	38	A18/D10
$\overline{BEC1}$	28	37	A19/D11
$\overline{BEC0}$	29	36	A20/D12
FC2	30	35	A21/D13
FC1	31	34	A22/D14
FC0	32	33	A23/D15

8.1.2 68-Terminal Pin-Grid Array

8.2 PACKAGE DIMENSIONS

L SUFFIX
CERAMIC PACKAGE
CASE 746-01
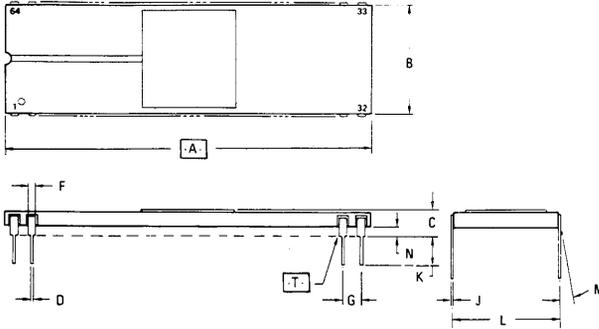
NOTES:
1. DIMENSION ⌐A⌐IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:

   | ⊕ | 0.25 (0.010)Ⓜ T | A Ⓜ |

3. ⌐T⌐ IS SEATING PLANE.
4. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	80.52	82.04	3.170	3.230
B	22.25	22.96	0.876	0.904
C	3.05	4.32	0.120	0.170
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	22.61	23.11.	0.890	0.910
M	–	10⁰	–	10⁰
N	1.02	1.52	0.040	0.060

P SUFFIX
PLASTIC PACKAGE
CASE 754-01
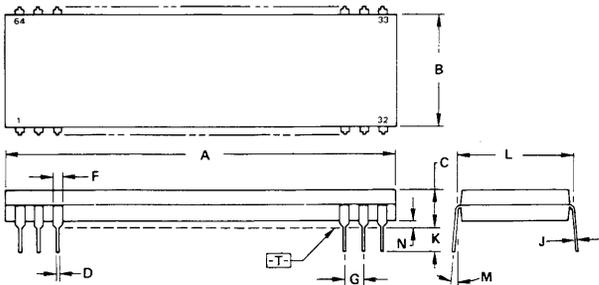
NOTES:
1. DIMENSIONS A AND B ARE DATUMS.
2. ⌐T⌐ IS SEATING PLANE.
3. POSITIONAL TOLERANCE FOR LEADS (DIMENSION D):

   | ⊕ | Ø 0.25 (0.010) Ⓜ | T A Ⓜ B Ⓜ |

4. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSION B DOES NOT INCLUDE MOLD FLASH.
6. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	81.16	81.91	3.195	3.225
B	20.17	20.57	0.790	0.810
C	4.83	5.84	0.190	0.230
D	0.33	0.53	0.013	0.021
F	1.27	1.77	0.050	0.070
G	2.54 BSC		0.100 BSC	
J	0.20	0.38	0.008	0.015
K	3.05	3.55	0.120	0.140
L	22.86 BSC		0.900 BSC	
M	0⁰	15⁰	0⁰	15⁰
N	0.51	1.01	0.020	0.040

RC SUFFIX
PIN GRID ARRAY
CASE 765A-01

R SUFFIX
PIN GRID ARRAY
WITH STANDOFF
(Dimensions essentially those of
Case 765A-01. See Figure 7-1 for
standoff detail.)

NOTES:
1. POSITIONAL TOLERANCE FOR
   LEADS (68 PLACES):

   ⊕ | ⌀ 0.13 (0.005) Ⓜ

2. DIMENSIONING AND
   TOLERANCING PER ANSI
   Y14.5, 1973.
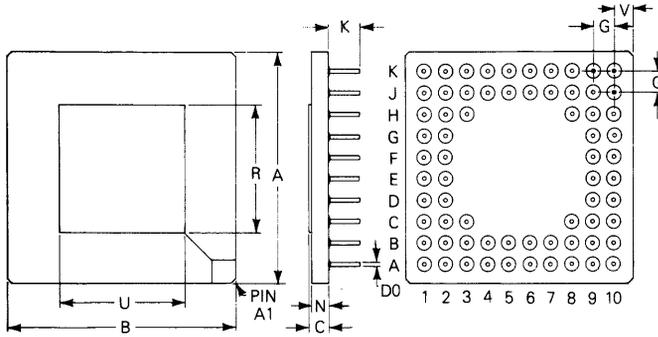
DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	26.41	27.43	1.040	1.080
B	26.41	27.43	1.040	1.080
C	2.10	2.59	0.083	0.102
D	0.51	0.60	0.020	0.024
G	2.54 BSC		0.100 BSC	
K	3.56	4.14	0.140	0.163
N	1.83	2.23	0.072	0.088
R	15.54	15.95	0.612	0.628
U	15.54	15.95	0.612	0.628
V	1.79	2.28	0.070	0.090

Addendum to
MC68450
Direct Memory Access Controller

The following information applies to ADI1216, *MC68450 Direct Memory Access Controller (DMAC)* Advance Information Data Sheet for the **A84G Mask only**:

1. When not using the function code lines, these lines must be kept high by using pullup resistors.

2. When the following two conditions occur simultaneously, incorrect data is supplied by the DMAC during the write cycle immediately following the negation of the relinquish and retry (R&R) exception:
   a. R&R is asserted at the write cycle in the dual address mode.
   b. MPU access to the DMAC's internal register is performed after the DMAC relinquishes the bus due to the R&R exception.

   Suggested countermeasures are as follows:
   a. Assert R&R exception only during the read cycle when using dual addressing mode.
   b. If the R&R exception occurs during the write cycle of the dual addressing mode, avoid accesses to the DMAC's registers.
   c. Use the HALT exception instead of R&R exception to access the DMAC's internal registers.

3. If the MPU asserts $\overline{CS}$ of the DMAC when the DMAC has its $\overline{BR}$ asserted, then the DMAC negates its $\overline{BR}$. (This timing does not cause any problems when used directly with an M68000 MPU. This is a consideration for designs external bus arbiters.)

(M) *MOTOROLA* ■

4. When the MPU sets the START bit of the DMAC and a different channel is already active in the limited rate auto-request mode (LRAR), then the following bus arbitration timing may occur:
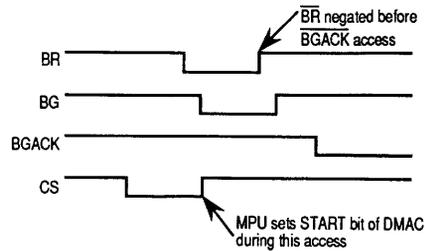

This timing diagram shows that the $\overline{BR}$ is negated too early and may cause a bus contention between the MPU and DMAC. The problem occurs in the following conditions:
  a. Two or more channel are in operation.
  b. One of the channels is in the LRAR.
  c. The priority level of the LRAR channel is higher than or equal to the other channels.
  d. The LRAR channel is initiated before the other channels.

Suggested countermeasures are as follows:
  a. Set the priority of the LRAR channel to the lowest level. If two or more channels are set to LRAR, a lower priority channel should be initiated first.
  b. External latches should be provided for as shown in the following diagram:
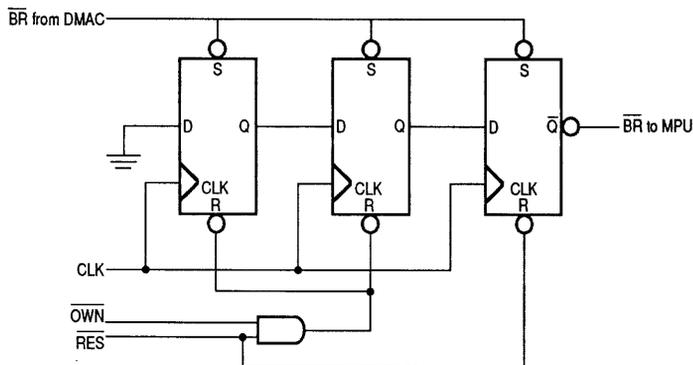
5. No DMAC register must be accessed sooner than seven DMAC clock cycles after setting the STR bit of the CCR.

6. When $\overline{\text{HALT}}$ ($\overline{\text{BEC}}$ input) is asserted to halt an active channel operation and then the STR bit is set to start another channel, neither channel asserts the $\overline{\text{BR}}$, and the entire DMAC operation stops. However, these modes operate correctly under the following constraints:
    a. Single addressing mode, except when $\overline{\text{HALT}}$ ($\overline{\text{BEC}}$ input) is asserted during the last bus cycle of any channel.
    b. Dual addressing mode, 16-bit operand size, 16-bit port size, except when $\overline{\text{HALT}}$ ($\overline{\text{BEC}}$ input) is asserted during the last bus cycle of any channel.

   Suggested countermeasures are as follows:
    a. When new channels are started by setting the STR bit by asserting $\overline{\text{HALT}}$, these conditions should be met:
       Assert the $\overline{\text{REQ}}$ input to the channel that was halted.
       Set the priority level of the newly started channel lower than the halted channel.
    b. The following procedure is suggested when setting the STR bit of other channels by halting the DMAC:
       Assert $\overline{\text{HALT}}$ ($\overline{\text{BEC}}$ input).
       Set the $\overline{\text{HLT}}$ bit of the CCR of all active channels.
       Negate $\overline{\text{HALT}}$ ($\overline{\text{BEC}}$ input).
       Set the STR bit to start the new channel.
       Reset the $\overline{\text{HLT}}$ of the other channels.
    c. Use single addressing mode whenever possible and do not set the STR bit of other channels when the $\overline{\text{HALT}}$ is performed during the last DMA bus cycle of a channel.

7. An incorrect vector may be returned under these conditions:
    a. One channel asserts $\overline{\text{IRQ}}$.
    b. A different channel is being serviced with an active bus cycle.
    c. The MPU runs an $\overline{\text{IACK}}$ cycle (in parallel with the DMA cycle) and causes $\overline{\text{HALT}}$ ($\overline{\text{BEC}}$ input) to be asserted to the DMAC.
   The vector returned is that of the channel running the bus cycles rather than the channel actually requesting the interrupt.

8. When using LRAR, the GCR must be set so that the LRAR burst time is at least (12.5 clocks) + (one MPU read cycle). LRAR burst time refers to the length of time the internal auto-request signal is asserted.

9. When using LRAR, assert the $\overline{\text{BEC}}$ lines for bus exceptions only when the DMAC has the bus ownership (i.e., $\overline{\text{BGACK}}$ bit of DMAC is asserted).

10. An operation timing error might occur on an attempt to program the SCR and CCR using a single 16-bit bus cycle instead of two, separate, 8-bit bus cycles.

Ⓜ **MOTOROLA**

A18566-2 PRINTED IN USA 7/89 IMPERIAL LITHO C66104 12,000 MPU