T-52-33-19

1. INTRODUCTION

TLCS-68000 microprocessors utilize state-of-the-art MOS technology to maximize performance and throughput. The TMP68440 dual-direct memory access controller (DDMA) is designed to complement the performance and architectural capabilities of TLCS-68000 Family microprocessors by moving blocks of data in a quick, efficient manner with minimum intervention from a processor. The DDMA performs memory-to-memory, peripheral-to-memory, and memory-to-peripheral data transfers by utilizing the following features:

- Two Independent DMA Channels with Programmable Priority
- Asynchronous TLCS-68000 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 TLCS-68000 Bus Options such as Halt, Bus Error, and Retry
- FIFO Locked Step Support with Device Transfer Complete Signal
- Can Operate on an 8-Bit Data Bus with the TMP68008
- Flexible Request Generation:

  Internal, Maximum Rate

  Internal, Limited Rate

  External, Cycle Steal

  External, Burst
- Programmable 8-Bit or 16-Bit Peripheral Device Types:

  Explicitly Addressed, TLCS-68000 Type

  Implicitly Addressed:

  Device with Request and Acknowledge

  Device with Request, Acknowledge, and Ready
- Non-Contiguous Block Transfer Operation (Continue Mode)
- Block Transfer Restart Operation (Reload Mode)
- Pin and Register Compatible Functional Subset of the TMP68450 DMAC

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 operations can occur concurrently with other operations that 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 DDMA will move blocks of data between the disk 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.

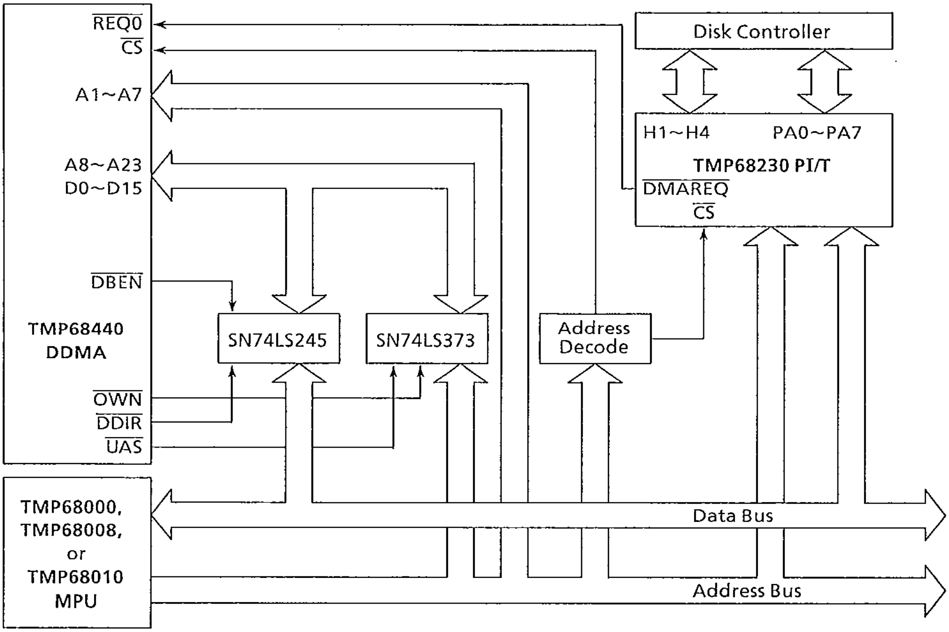


Figure 1.1 Typical System Congigurarion

A block of data consists of a sequence of byte or word operands starting at a specific address in memory with the length of the block determined by a transfer count as shown in Figure 1.2. A single channel operation may involve the transfer of several blocks of data between the memory and a device.

Any operation involving the DDMA 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 DDMA with control information, address pointers, and transfer counts and then starts the channel. During the transfer phase, the DDMA 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 DDMA indicates the status of the operation in a status register. During all phases of a data transfer operation, the DDMA will be in one of three operating modes:

IDLE This is the state that the DDMA 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 DDMA enters when it is chip selected by another bus master in the system (usually the main system processor) . In this mode, the DDMA 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 DDMA enters when it is acting as a bus master to perform an operand transfer.

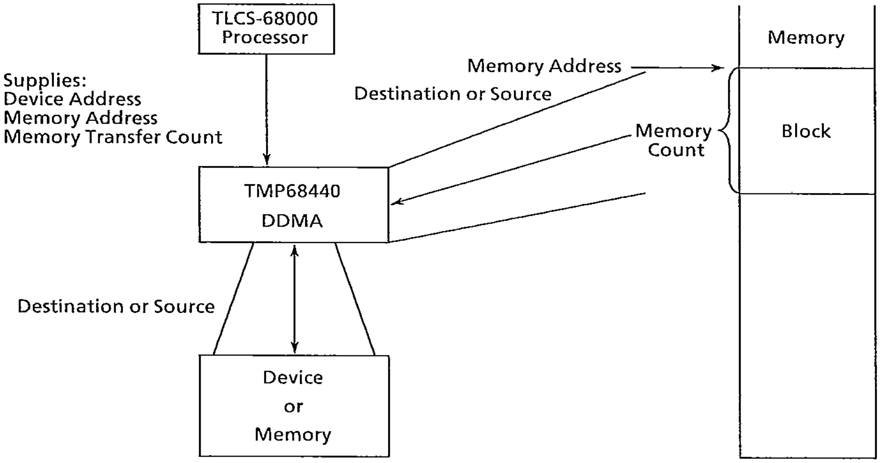


Figure 1.2 Data Block Format

1.2 TRANSFER MODES

The DDMA can perform implicit address or explicit address data transfers. 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 DDMA 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.

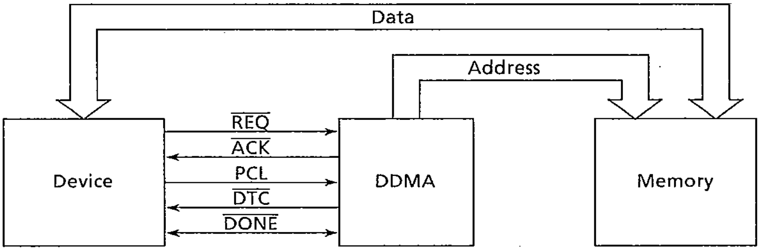


Figure 1.3 Implicitly Addressed Device Interface

Explicitly addressed devices require that a data register within the peripheral device be addressed. No signals other than the TLCS-68000 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 DDMA 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.

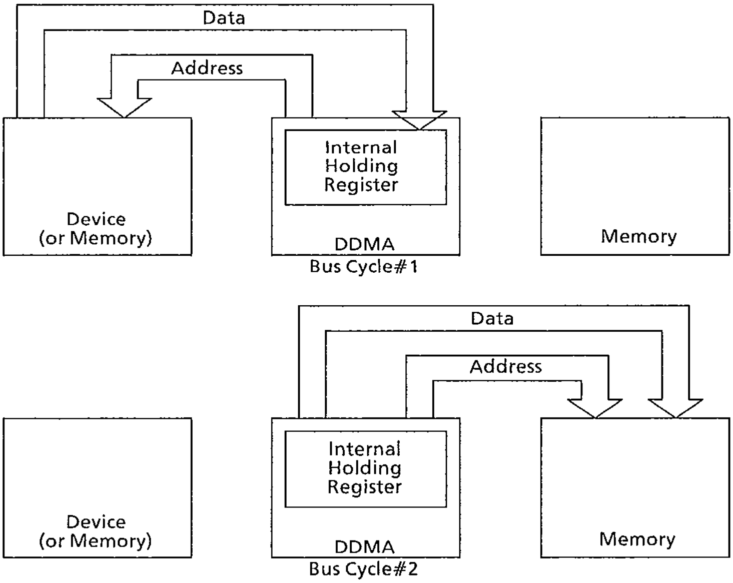


Figure 1.4 Dual-Address Transfer Sequence

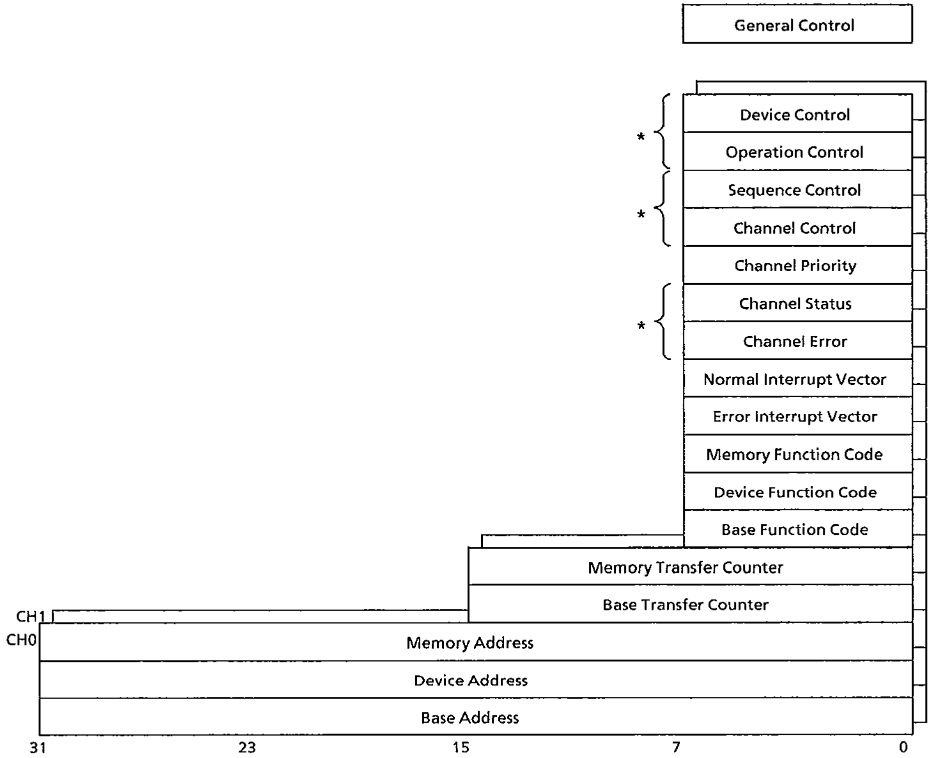
1.3 REQUEST MODES

Requests may be externally generated by a device or internally generated by the auto-request mechanism of the DDMA. 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.

1.4 REGISTERS

The DDMA contains 17 on-chip registers for each of the two 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 1.5.

The DDMA registers contain information about the data transfer 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 1.5 DDMA Programmer's Model

2. SIGNAL DESCRIPTION

This section contains a brief description of the DDMA 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 ture, independent of whether that level is represented by a high or low voltage. The term negateor negation is used to indicate that a signal is inactive or false.

2.1 SIGNAL ORGANIZATION

The pin assignments is show in Figure 2.1. And the groups shown in Figure 2.2. The function of each signal or group of signals is discussed in the following paragraphs.

2.1.1 Address/Data Bus (A8/D0 ~ 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 ~ A7)

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

2.1.3 Function Codes (FC0 ~ 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 spaces 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 source register for 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 (or A0 and data strobe when using an 8-bit bus) , and data transfer acknowledge. These signals are described in the following paragraphs.

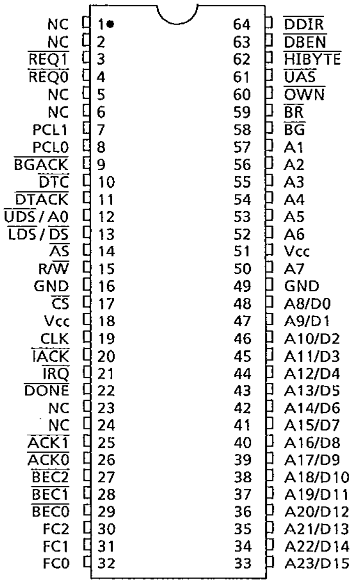


Figure 2.1 Pin Assignments

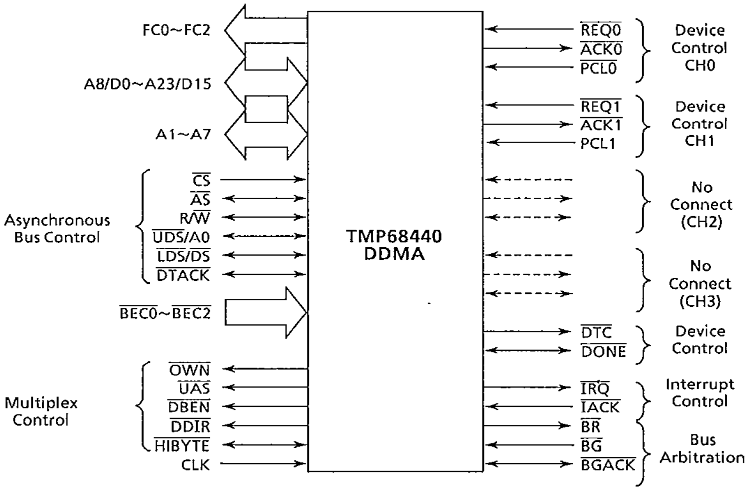


Figure 2.2 Signal Organization

2.1.4.1 Chip Select ($\overline{\text{CS}}$)

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

2.1.4.2 Address Strobe ($\overline{\text{AS}}$)

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 DDMA can take control of the bus (if the DDMA has requested and been granted use of the bus) .

2.1.4.3 Read/Write (R/$\overline{\text{W}}$)

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 DDMA to the data bus and a low level indicates a transfer from the data bus to the DDMA. 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 ($\overline{\text{UDS}}$/A0 AND $\overline{\text{LDS}}$/$\overline{\text{DS}}$)

These bidirectional lines are used for different purposes depending on whether the DDMA is operating on an 8-bit or a 16-bit bus (refer to "2.1.5.5 High Byte ($\overline{\text{HIBYTE}}$) ").

When using a 16-bit bus, these pins function as $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$. During any bus cycle, $\overline{\text{UDS}}$ is asserted if data is to be transferred over data lines D8~D15 and $\overline{\text{LDS}}$ is asserted if data is to be transferred over data lines D0~D7. $\overline{\text{UDS}}$/$\overline{\text{LDS}}$ are asserted by the DDMA when operating in the DMA mode and by another bus master when in the MPU mode.
When using an 8-bit bus, these pins function as A0 and $\overline{\text{DS}}$. A0 is an extension to the lower address lines to provide the address of a byte in the 16 megabyte address map and is valid when A1~A7 are valid. $\overline{\text{DS}}$ is used as a data strobe to enable external data buffers and to indicate that valid data is on the bus during a write cycle.

2.1.4.5 Data Transfer Acknowledge($\overline{\text{DTACK}}$)

This bidirectional line is used to signal that an asynchronous bus cycle may be terminated. In the MPU mode, this output indicates that the DDMA 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 DDMA to determine when to terminate a bus cycle. As long as $\overline{\text{DTACK}}$ remains negated, the DDMA 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 ($\overline{\text{BEC0}}$ ~ $\overline{\text{BEC2}}$)

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 2.3 shows the 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 to transfer data to or from a 16-bit system data bus during single address transfers. When the DDMA is used on an 8-bit system data bus with 8-bit devices, only the SN74LS245 buffer for D0~D7 is needed.

2.1.5.1 Own ($\overline{\text{OWN}}$)

This output indicates that the DDMA is controlling the bus. It is used as the enable signal to turn on the external address latch drivers and control signal buffers.

2.1.5.2 Upper Address Strobe ($\overline{\text{UAS}}$)

This 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 output is used as the enable signal to the external bidirectional data buffers.

2.1.5.4 Data Direction ($\overline{\text{DDIR}}$)

This output controls the direction of the external bidirectional data buffers. If $\overline{\text{DDIR}}$ is high, the data transfer is from the DDMA to the data bus. If $\overline{\text{DDIR}}$ is low, the data transfer is from the data bus to the DDMA.

2.1.5.5 High Byte ($\overline{\text{HIBYTE}}$)

This bidirectional signal determines the size of the bus used by the DDMA during a reset operation. If this signal is asserted (tied to ground) during reset, the data bus size is eight bits and $\overline{\text{HIBYTE}}$ will not be used as an output. If it is negated (pulled high by a resistor) during reset, the data bus size is assumed to be 16 bits and $\overline{\text{HIBYTE}}$ will be used as an output during single-address DMA transfers between an 8-bit device and a 16-bit memory. As an output, $\overline{\text{HIBYTE}}$ 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 signal address transfer between an 8-bit device and a 16-bit memory (refer to "4.3.2 Single Address Transfers") .

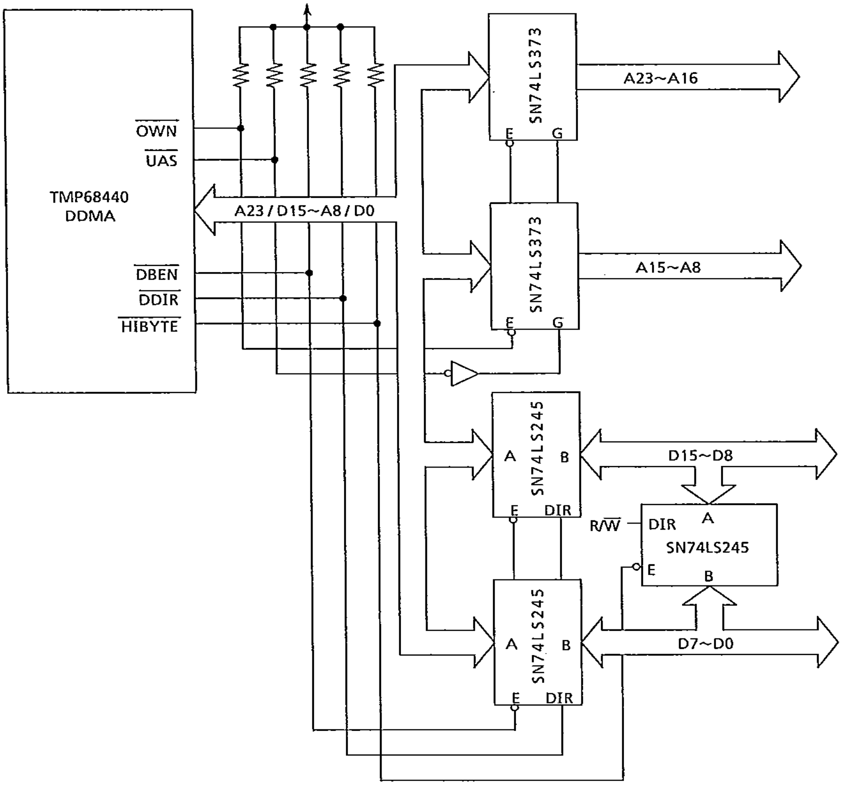


Figure 2.3 Demultiplex Logic

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 ($\overline{BR}$)

This output is asserted by the DDMA to request control of the bus.

2.1.6.2 Bus Grant ($\overline{BG}$)

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

2.1.6.3 Bus Grant Acknowledge ($\overline{\text{BGACK}}$)

This bidirectional signal is asserted by the DDMA to indicate that it is the current bus master. $\overline{\text{BGACK}}$ is monitored as an input to determine when the DDMA 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 DDMA 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 DDMA. In response to the assertion of $\overline{\text{IACK}}$, the DDMA will place a vector on D0~D7 that will be used by the MPU to fetch the address of the proper DDMA interrupt handler routine.

2.1.8 Device Control

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

2.1.8.1 Request ($\overline{\text{REQ0}}$, $\overline{\text{REQ1}}$)

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 ($\overline{\text{ACK0}}$, $\overline{\text{ACK1}}$)

These outputs are asserted by the DDMA 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, PCL1)

These inputs are multi-purpose signals that may be programmed to function as ready, abort, reload, status, or interrupt inputs.

2.1.8.4 Data Transfer Complete ($\overline{\text{DTC}}$)

This output is asserted by the DDMA during any DDMA 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 DDMA or a peripheral device during any DMA bus cycle to indicate that the data being transferred is the last item in a block. The DDMA will assert this signal during a bus cycle when the memory transfer count register is decremented to zero and the continue bit in the channel control register is not set.

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 DDMA. The clock input should not be gated off at any time and the clock signal must conform to minimum and maximum pulse width times.

2.1.10 No Connection (NC)

Six pins on the TMP68440 are not connected in order to maintain pin compatibility with the TMP68450 DMAC; these pins are in the positions of the device control signals for channels two and three ($\overline{\text{REQ2}}$, $\overline{\text{ACK2}}$, PCL2, $\overline{\text{REQ3}}$, $\overline{\text{ACK3}}$, and PCL3) on the DMAC. It is suggested that these pins be left unconnected to allow future expansion; however, if a DDMA is placed into a socket designed to also accommodate a DMAC, the four input signals will be ignored and the two output signals will be allowed to float.

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
A8 / D0~A23 / D15	In/Out	High	Three State
A1~A7	In/Out	High	Three State
FC0~FC2	Out	High	Three State
$\overline{CS}$	In	Low	
$\overline{AS}$	In/Out	Low	Three State*
R / $\overline{W}$	In/Out	High / Low	Three State*
$\overline{UDS}$ / A0	In/Out	Low / High	Three State*
$\overline{LDS}$ / $\overline{DS}$	In/Out	Low	Three State*
$\overline{DTACK}$	In/Out	Low	Open Drain*
$\overline{OWN}$	Out	Low	Open Drain*
$\overline{UAS}$	Out	Low	Three State*
$\overline{DBEN}$	Out	Low	Three State*
$\overline{DDIR}$	Out	High / Low	Three State*
$\overline{HIBYTE}$	In/Out	Low	Three State*
$\overline{BEC0}$~$\overline{BEC2}$	In	Low	
$\overline{BR}$	Out	Low	Open Drain
$\overline{BG}$	In	Low	
$\overline{BGACK}$	In/Out	Low	Open Drain*
$\overline{IRQ}$	Out	Low	Open Drain
$\overline{IACK}$	In	Low	
$\overline{REQ0}$, $\overline{REQ1}$	In	Low	
$\overline{ACK0}$, $\overline{ACK1}$	Out	Low	Three State*
PCL0, PCL1	In	Programmed	
$\overline{DTC}$	Out	Low	Three State*
$\overline{DONE}$	In/Out	Low	Open Drain
CLK	In		

*: These signals require a pullup resistor to maintain a high voltage when in the highimpedance or nagated state. However, when these signals go to the highimpedance or nagated state, they will first drive the pin high momentarily to reduce the signal rise time.

3. REGISTER DESCRIPTION

This section contains a decription of the DDMA internal registers and the control bit assignments within each register. Figure 3.1 shows the memory mapped locations of the registers for each channel on a 16-bit bus. Figure 3.2 (found on a foldout page at the back of this book) shows the register summary and may be used for a quick reference to the bit definitions within each register. The register locations defined as 'Null Register' may be read or written; however, a write access will not affect any DDMA channel operation and a read access will always return all ones. In the descriptions of each register, some bits are defined as 'not used', in which case writes to those bits will have no effect any they will always read as zeros.

The register memory map for the TMP68440 DDMA is identical to the register memory for map the TMP68450 DMAC, 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, and the channel 2 and 3 registers on the DMAC are treated as null registers on the DDMA. If any programmable options labled 'TMP68450 Reserved' or 'Undefined, Reserved' are programmed into a DDMA 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 both channels since they are indentical. When the operation of a register affects the two channels defferently or is common to both channels, that fact will be included in the register description.

All registers within the DDMA 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 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 DDMA to access memory while it is a bus master. Since the DDMA 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 DDMA will continue normal operation without any error indication.

Channel Base	
CH0	- 00
CH1	- 40
CH2*	- 80
CH3*	- C0

Register Offset	15** … 8	7 … 0	
00	CSR	CER	01
02	▨	▨	03
04	DCR	OCR	05
06	SCR	CCR	07
08	▨	▨	09
0A	MTCR		0B
0C	MSB MAR		0D
0E		LSB	0F
10	▨	▨	11
12	▨	▨	13
14	MSB DAR		15
16		LSB	17
18	▨	▨	19
1A	BTCR		1B
1C	MSB BAR		1D
1E		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

* All accesses to channel 2 and 3 registers will be treated as null accesses.

** When operated on an 8-Bit bus, all register data is transferred over D0~D7; the word and long word registers are then accessed as a contiguous set of bytes.

*** The GCR is located at FF only.

Figure 3.1 Register Memory Map

The memory address register (MAR) contents are used whenever the DDMA 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 DDMA 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) is used by the DDMA in the continue and reload modes of operation to hold the address to be transferred to the MAR in response to a continue operation or reload request.

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

These 8-bit registers contain function code values that are used by the DDMA 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 DDMA logical addresses into the proper physical addresses. The value programmed into the memory function code register (MFCR) or the device function code register (DFCR) 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 or DAR respectively. The base function code register (BFCR) is used by the DDMA in the continue and reload modes of operation to hold the function code to be transferred to the MFCR in response to a continue operation or reload request. Note that these are 4-bit registers, with bit 3 readable and writable 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 DDMA 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 continue and reload modes of operation to hold the operand count to be transferred to the MTCR in response to a continue operation or reload request.

3.4 INTERRUPT VECTOR REGISTERS (NIVR AND EIVR)

These two 8-bit registers are used by the DDMA during interrupt acknowledge bus cycles. When the $\overline{\text{IACK}}$ signal is asserted to the DDMA, 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 each channel, the highest priority channel will be serviced first. When both channels are at the same priority level, an alternating 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 DDMA 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 (TMP68450 Reserved)
- 11 Priority 3 (Lowest) (TMP68450 Reserved)

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
XRM		DTYP		DPS	PCL		

XRM External Request Mode
- 00 Burst Transfer Mode
- 01 (Undefined, Reserved)
- 10 Cycle Steal Without Hold
- 11 (TMP68450 Reserved)

DTYP Device Type
- 00 TLCS-68000 Compatible, Explicitly Addressed
- 01 (TMP68450 Reserved)
- 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	(TMP68450 Reserved)
011	Abort Input
100	Reload Input
101	(Undefined, Reserved)
110	(Undefined, Reserved)
111	(Undefined, Reserved)

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 mode, the $\overline{\text{REQ}}$ pin is an edge sensitive input; an asserted edge must occur to request each operand transfer. For more detailed information, refer to "5.2.3.3 Transfer Request Generation".

3.6.1.2 Device type

This field selects one of three available device transfer protcols. For TLCS-68000 type devices, the DDMA will use a dual address transfer protocol by running TLCS-68000 type bus cycles to transfer data to or from the device registers and a second bus complete the operand transfer from or to memory (note that this device type also allows memory-to-memory block transfers) .

In the remaining two device protocols, the DDMA 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 DDMA 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 memory asserts $\overline{\text{DTACK}}$. For a device with acknowledge and ready, the PCL line is used as a ready signal that is similar to the $\overline{\text{DTACK}}$ signal. During a write to memory, the DDMA will not assert the data strobes to memory until the device has asserted ready. During a read from memory, the DDMA will not terminate the bus cycle until the device has asserted ready and memory has asseted $\overline{\text{DTACK}}$. 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 data port is, either 8 or 16 bits. Refer to Table 5.1 for legal port, memory, and operand size combinations.

3.6.1.4 Peripheral Control Line

If the DTYP field is not programmed for device with acknowdledge and ready, this field will define the function of the PLC line. The level on the PCL line can always be read from the PCL 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 or reload input, it is a high-to-low edge-sensitive signal that will cause the DDMA to abort the channel operation or reload the MAR, MFCR, and MTCR from the BAR, BFCR, and BTCR respectively.

3.6.2 Operation Control Register (OCR)

This register defines the type of operatoion 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 (TMP68450 Reserved)

11 (Undefined, Reserved)

CHAIN Chain Operation

00 Chaining Disabled

01 (Undefined, Reserved)

10 (TMP68450 Reserved)

11 (TMP68450 Reserved)

REQG Request Generation

00 Internal Request at Limited Rate

01 Internal Request at Maximum Rate
10 External Request
11 (TMP68450 Reserved)

3.6.2.1 Transfer Direction

This field controls the direction of the block transfer. A zero indicates that the transfer is from memory to the device, a one indicates that 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 DDMA supports byte and word operand sizes; refer to Table 5.2 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. No chaining functions are available on the DDMA; however, these bits are writable to maintain compatibility with the TMP68450 DMAC (although a confuguration error will occur if a non-zero value is used) .

3.6.2.4 Request Generation Method

This field defines the method used by the DDMA to detect operand transfer requests. Two types of internal request generation methods are available, or a request may be externally generated. 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 incremented during transfer operations. Each address register may be programmed to remain unchanged during an operand transfer or to be incremented after each operand is successfully transferred. The value by which a register is incremented after each DMA bus cycle depends on the operand size and device port size; for more details on address sequencing, refer to "5.2.3.2 Address Sequencing".

Sequence Control Register (SCR)

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	(TMP68450 Reserved)
	11	(Undefined, Reserved)
DAC	Device Address Count	
	00	DAC Does Not Count
	01	DAC Counts Up
	10	(TMP68450 Reserved)
	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 DDMA. The STR bit will be cleared by the DDMA 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 begining 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 registers may be modified or an operation timing error will occur. The SCR may be written simultaneously with the CCR with the STR bit set to complete the initialization of the channel and start it with one bus cycle.

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 DDMA. 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 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 wirtten 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 (CCR)

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 Contine 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 (Undefined, Reserved)

3.7 STATUS REGISTERS

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 reflrects the status of a channel operation. The status information supplied indicates if the channel is active or complete, whether an error has occured, 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 acontinued operation is complete and the DDMA 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 RLD, PCT, and PCS bits are associated with the function of the PCL signal. RLD will be set if the PCL line is programmed as a reload input and a reload operation was requested and has been successfully complated. The PCT bit will be set anytime that a high-to-low trasition 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 DDMA, the COC, BTC, NDT, ERR, RLD, 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 writing to it, the channel error register will also be cleared.

Channel Status Register (CSR)

7	6	5	4	3	2	1	0
COC	BTC	NDT	ERR	ACT	RLD	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 Termination by Device
- 1 Device Terminated Operation with $\overline{\text{DONE}}$

ERR Error
- 0 No Error
- 1 Error Has Occurred

ACT Channel Active
- 0 Channel Not Active
- 1 Channel Is Active

RLD Reload Occurred
0 Reload Has Not Occurred
1 Reload Operation Occurred

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 (CER)

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 · · · 11111	(Undefined, Reserved)

rr Register Code

00	MAR and DAR
01	MAR or MTCR
10	DAR
11	BAR or BTCR0

3.8 GENERAL CONTROL REGISTER (GCR)

This is a global register that may affect the operation of both 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 autorequest operation.

General Control Register (GCR)

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

0 Not Used

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

BR Bandwidth Available to DDMA
- 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 DDMA register memory map is provided in Figure 3.2 found on a foldout page at the back of this book

3.10 RESET OPERATION RESULTS

When the DDMA is reset, either during a system power-up sequence or to re-initialize the DDMA, 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 (1/2)

Register	Value	Comments
MARc	XXXXXXXX	
DARc	XXXXXXXX	
BARc	XXXXXXXX	
MFCRc	X	
DFCRc	X	

Table 3.1 Reset Operation Results (2/2)

Register	Value	Comments
BFCRc	X	
MTCRc	XXXX	
BTCRc	XXXX	
NIVRc	0F	Uninitialized Vector
EIVRc	0F	Uninitialized Vector
CPRc	00	
DCRc	00	
OCRc	00	
SCRc	00	
CCRc	00	Channel Not Active Interrput Disabled
CSRc	00 or 01	(Depending on the Level of PCLc)
CERc	00	No Errors
GCRc	00	

X–indicates an unknown value or the previous value of the register.

C–is the channel number (i.e., 0 or 1).

4. BUS OPERATION

This section describes the bus signal operation of the DDMA in all operating modes. All bus operations are described in relation to the DDMA 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 TMP68000 16-bit microprocessor. Functional timing diagrams are included to assist in the definition of signal timings; however, these diagrams are not indended 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 DDMA to sample an external asynchronous input signal, determine whether it is high or low, and synchronize the input to the DDMA 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.

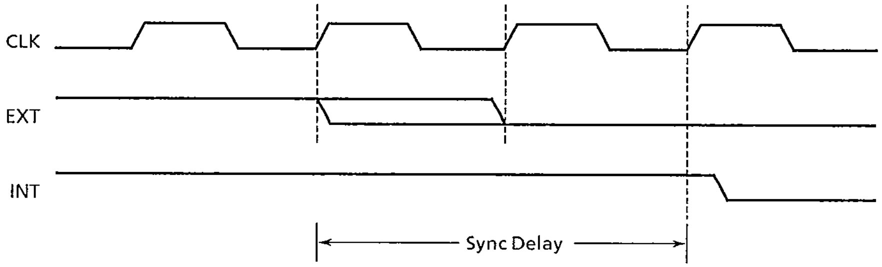


Figure 4.1 Relationship Between External and Internal Signals

4.1 RESET OPERATION

The DDMA can be reset to the IDLE mode by an external device at any time by asserting the reset encoding on $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$. 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 DDMA for at least 100 milliseconds after VCC 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 10 clock cycles to reset the DDMA.

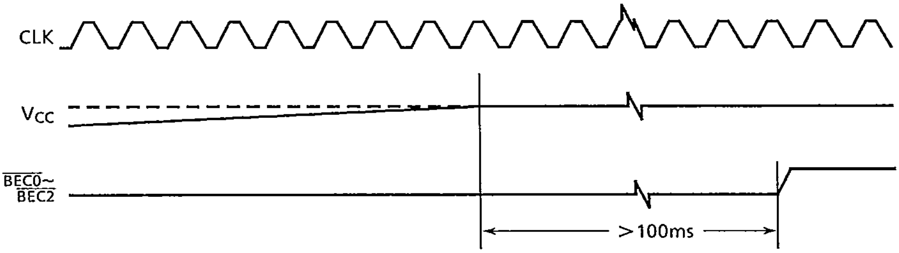


Figure 4.2 Reset Operation Timing Diagram

The level the $\overline{\text{HIBYTE}}$ signal during a reset operation is used to indicate to the DDMA what size the external data bus is. If $\overline{\text{HIBYTE}}$ is high when reset is negated, the bus is assumed to be 16 bits wide; if it is low when reset is negated, the bus is assumed to be eight bits wide and $\overline{\text{HIBYTE}}$ will never be used as an output. This convention allows operation without any extra hardware since $\overline{\text{HIBYTE}}$ will be pulled high by a resistor or tied to VCC when a 16-bit bus is used and should be tied to GND when an 8-bit bus is used (since an external buffer is not needed to transfer data from D8~D15 to D0~D7) .

4.2 MPU MODE OPERATION

In the MPU mode, the DDMA acts as a peripheral slave to another bus master. When the $\overline{\text{CS}}$ signal is asserted, the DDMA will enter the MPU mode from the idle mode (if the DDMA is in the DMA mode and $\overline{\text{CS}}$ is asserted, an error will be signaled) . During MPU mode operations, the DDMA will accept data from or place data on the data bus according to the level on the R/$\overline{\text{W}}$ 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, $\overline{\text{UDS}}$/A0, and $\overline{\text{LDS/DS}}$. Once the data transfer has been completed, the DDMA will assert $\overline{\text{DTACK}}$ to terminate the bus cycle and return to the IDLE mode.

During MPU mode operations, the DDMA 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 DDMA 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 TMP68000, TMP68010, or TMP68008 with a clock signal identical to the DDMA CLK signal. The state numbers (S0, S1, ect.) refer to the numbering convention for those processor.

4.2.1 MPU Read Cycles

During MPU read cycles, the DDMA will place data on the data bus and assert $\overline{\text{DTACK}}$ 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 on a 16-bit bus.

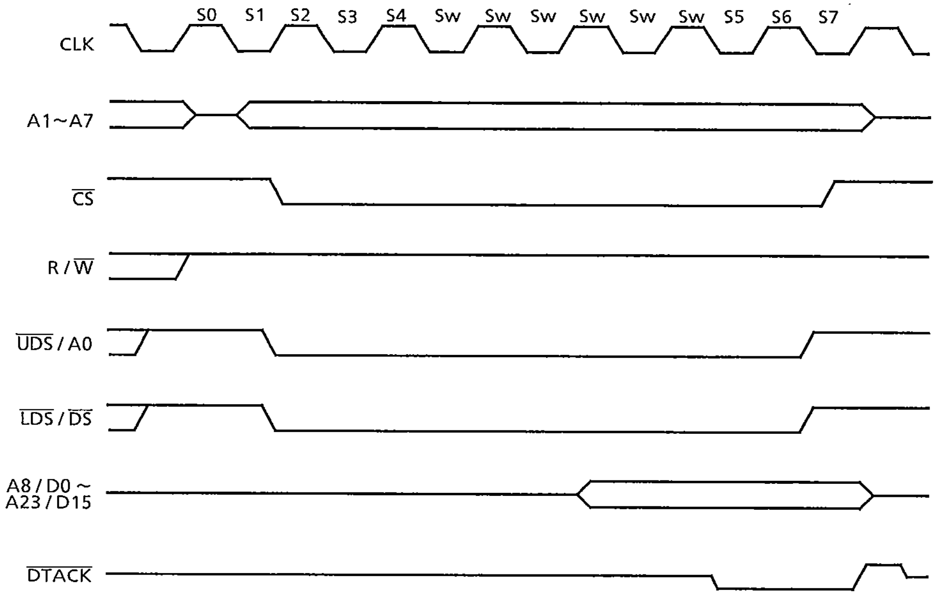


Figure 4.3 MPU Read Cycle Timing Diagram

The timing for even and odd byte MPU reads on a 16-bit bus or any MPU read on an 8-bit bus is identical, with the encoding of $\overline{UDS}$/A0 and $\overline{LDS}$/$\overline{DS}$ selecting the proper byte. A16/D8~A23/D15 are undefined during the data phase of an MPU read cycle on an 8-bit bus.

The DDMA starts an MPU read operation when $\overline{CS}$ is asserted and synchronized internally with the R/$\overline{W}$ line high. The DDMA responds to the $\overline{CS}$ by decoding A1~A7, $\overline{UDS}$/A0, and $\overline{LDS}$/$\overline{DS}$ 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 DDMA 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 DDMA 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 on a 16-bit bus. The timing for even and odd byte MPU writes on a 16-bit bus or any MPU write on an 8-bit bus is identical, with the encoding of $\overline{UDS}$/A0 and $\overline{LDS}$/$\overline{DS}$ selecting the proper byte. A16/D8~A23/D15 will be ignored during the data phase of an MPU write cycle on 8-bit bus.

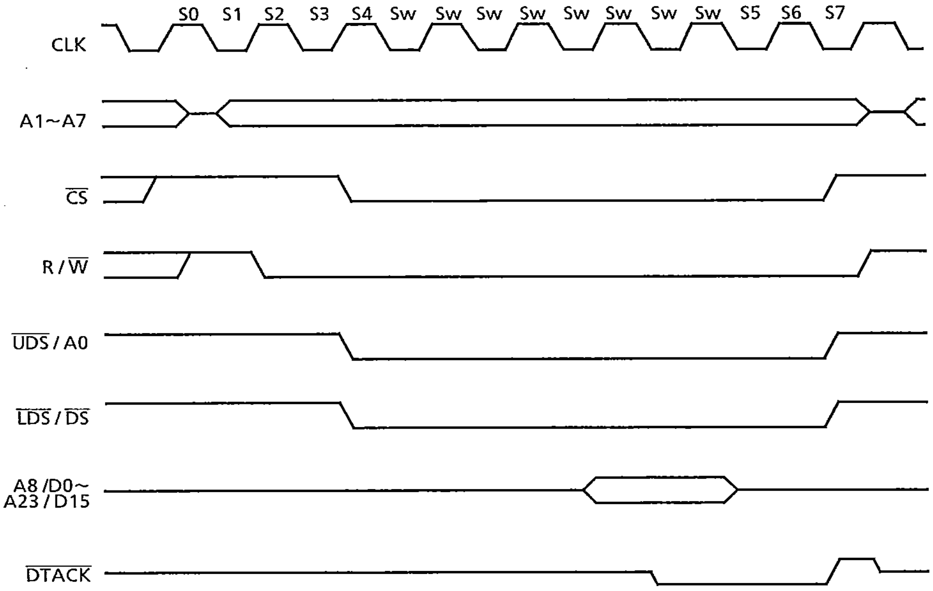


Figure 4.4 MPU Write Cycle Timing Diagram

The DDMA 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 DDMA responds to the $\overline{CS}$ by decording A1~A7, $\overline{UDS}$/A0, and $\overline{LDS}$/$\overline{DS}$, 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 DDMA 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 DDMA. The timing of an interrupt cycle is identical to an odd byte 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 DDMA 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.

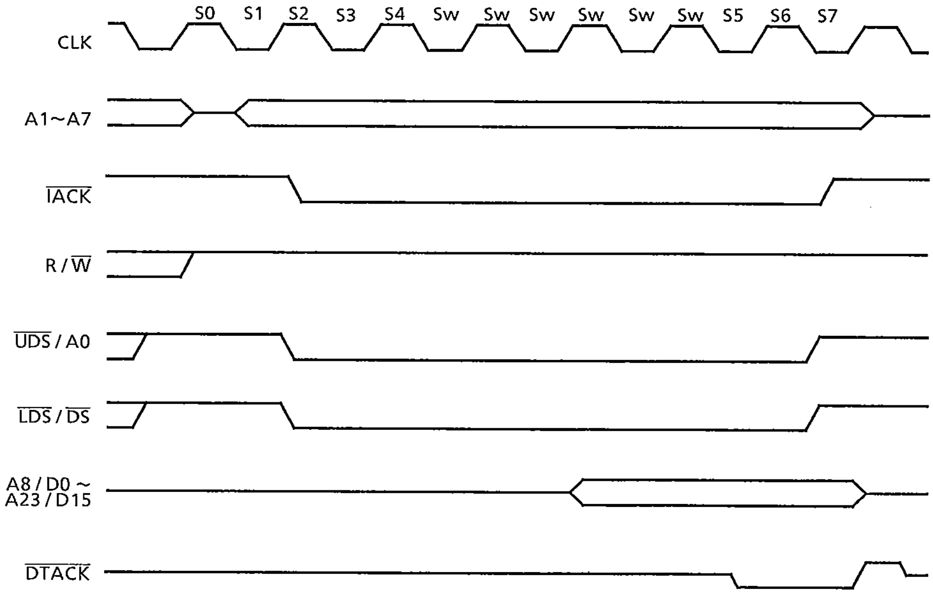


Figure 4.5 Interrupt Acknowledge Cycle Timing Diagram

The DDMA starts an interrupt acknowledge operation when $\overline{\text{IACK}}$ is asserted and synchronized internally. The DDMA responds to the $\overline{\text{IACK}}$ by placing a vector number on A8/D0~A15/D7, driving $\overline{\text{DDIR}}$ high and asserting $\overline{\text{DBEN}}$ and $\overline{\text{DTACK}}$. The A1~A7, $\overline{\text{UDS}}$/A0 and $\overline{\text{LDS}}$/$\overline{\text{DS}}$ 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 $\overline{\text{IACK}}$ signal is negated, at which time the DDMA will three state the address/data bus and negate $\overline{\text{DDIR}}$, $\overline{\text{DBEN}}$, and $\overline{\text{DTACK}}$.

4.3 DMA MODE OPERATION

In the DMA mode of operation, the DDMA is acting a bus master and performs data transfer operations between a device and memory in response to operand transfer requests. There are two classes of DMA bus cycles that may be run by the DDMA: TLCS-68000 cycles that transfer data between a device or memory and the DDMA internal holding register, and cycles that transfer data directly between a device and memory without going through the DDMA. The first class of cycle is referred to as dual address cycle, 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 cycle, the DDMA may execute a read or a write operation.

The following paragraphs describe the four DMA cycle types according to the signal transitions 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 TLCS-68000 processor. The DDMA 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 DDMA internal holding register or written from that register to the data bus. All cycle termination options of the TLCS-68000 bus are supported, such as halt, cycle retry, or bus error.

Note that several combinations of memory, 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 with a 16-bit memory bus, the device may be placed on the most significant (even address) or least significant (odd address) half of the data bus. When a 16-bit device is used with an 8-bit memory bus, the $\overline{\text{UDS}}$/A0 and $\overline{\text{LDS}}/\overline{\text{DS}}$ lines will operate as A0 and $\overline{\text{DS}}$ even when the DDMA is accessing the 16-bit device. In this case, the address used will always be even (A0 will be zero) and the device will always be accessed as a word, even though only one data strobe is generated.

4.3.1.1 Dual Address Read

During this type of a DMA cycle, the DDMA will read data from a device or memory into an internal holding register. Figure 4.6 shows the functional timing for a word read on a 16-bit bus. The timing for an even or odd byte read on a 16-bit bus or any read on an 8-bit bus is identical, with the encoding of $\overline{\text{UDS}}$/A0 and $\overline{\text{LDS}}/\overline{\text{DS}}$ selecting the proper byte. The signal protocol is as follows.

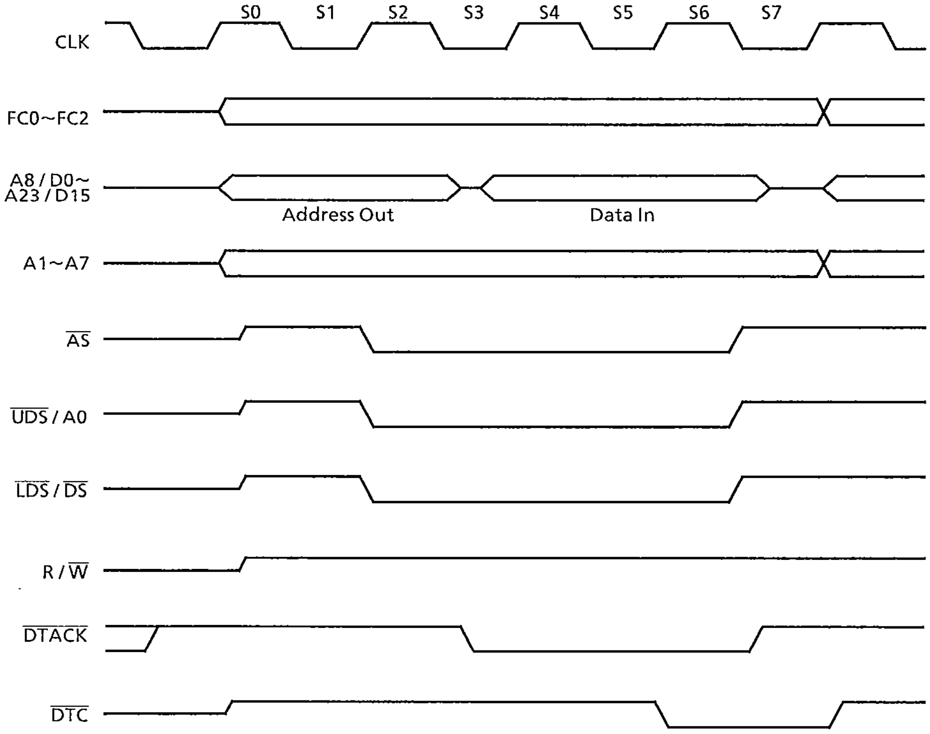


Figure 4.6 Dual Address Read Cycle Timing Diagram

S0 The DDMA drives the FC0~FC2, A1~A7, and A8/D0~A23/D15 pins with data from the function code and address registers (either the MFCR and MAR or DFCR and DAR) of the current channel, $\overline{UAS}$ is asserted to enable the external address latches and R/$\overline{W}$ is driven high. Also, if an 8-bit bus is being used, $\overline{UDS}$/A0 will be driven as an address signal.

S1 No signals change and the upper address information is allowed 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. $\overline{UDS}$/A0 (on a 16-bit bus) and/or $\overline{LDS}$/$\overline{DS}$ will also assert to indicate that data should be placed on the bus and $\overline{DDIR}$ will be driven low to turn the external data buffers/demultiplexers around (to drive the address/data pins when enabled) . If the current bus cycle is to a device, $\overline{ACK}$ will also be asserted to the device and $\overline{DONE}$ will be asserted if this is the last transfer to the device for current channel operation.

S3 The address/data pins are three stated in perparation to receive input data and $\overline{\text{DBEN}}$ is asserted to enable the data buffer/demultiplexer. The DDMA 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 DDMA 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 DDMA 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 DDMA is also internally synchronizing the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$ signals.

S5 Data is sampled during this period, with the data bus value latched at the end of S5. 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 DDMA address/data pins within the setup time to the rising edge of S5. For byte read operations on a 16-bit bus, the data lines for byte not being read will be ignored. The DDMA also continues to synchronize the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$ pins.

S6 $\overline{\text{DTC}}$ 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 the bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S7 $\overline{\text{AS}}$ is negated to indicate tha the cycle has terminated. $\overline{\text{UDS}}$/A0 and /or $\overline{\text{LDS/DS}}$ and $\overline{\text{DBEN}}$ will also negate to disable external data buffers/multiplexers. The memory or peripheral device that was addressed during the cycle may then negate $\overline{\text{DTACK}}$ and/or $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$.

S7+1 This may be S0 of the following cycle, if the DDMA is ready to immediately execute another bus cycle (which will always be the case for normal termination, but not for exceptional terminations). The FC0~FC2, A1~A7, and $\overline{\text{UDS}}$/A0 (when used as an address line) pins will all be changed to their next states. If the DDMA is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DDMA is not ready to start a bus cycle, these lines will be three stated. $\overline{\text{DDIR}}$ is negated to turn the data buffers/demultiplexers away from the DDMA. $\overline{\text{ACK}}$, $\overline{\text{DTC}}$, and $\overline{\text{DONE}}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed.

4.3.1.2 Dual Address Write

During this type of a DMA cycle, the DDMA 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 on a 16-bit bus or any write on an 8-bit bus is identical, with the encoding of $\overline{\text{UDS}}$/A0 and $\overline{\text{LDS}}$/$\overline{\text{DS}}$ selecting the proper byte. The signal protocol is as follows.

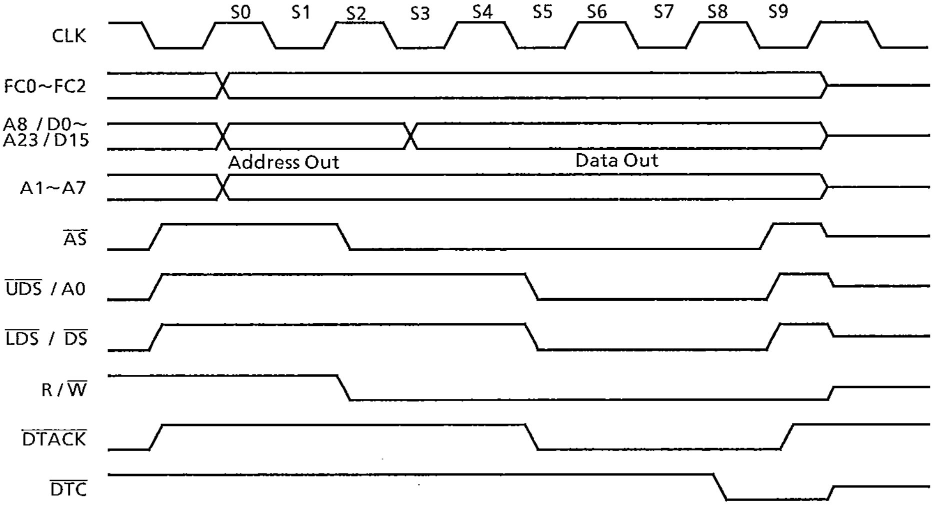


Figure 4.7 Dual Address Write Cycle Timing

S0 The DDMA drives the FC0~FC2, A1~A7, and A8/D0~A23/D15 pins with data from the function code and address registers (either the MFCR and MAR or DFCR and DAR) of the current channel and $\overline{\text{UAS}}$ is asserted to enable the external address latches. Also, if an 8-bit bus is being used,$\overline{\text{UDS}}$/A0 will be driven as an address signal. $\overline{\text{DDIR}}$ will be driven high to indicate that the direction of the data flow will be from the DDMA to the external bus.

S1 No signals change and the upper address information is allowed 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. R/$\overline{\text{W}}$ is also driven low to indicate that this cycle is a write.

S3 The address/data pins change from address to data and $\overline{\text{DBEN}}$ is asserted to allow data to propagate through the external data buffers/demultiplexers. For byte write operations on a 16-bit bus, the value of the data lines for the byte not being written is undefined. If the cycle is accessing a device, $\overline{\text{ACK}}$ will be asserted at this time and $\overline{\text{DONE}}$ will be asserted if this is the last transfer to device for the current channel operation. The DDMA also begins 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 S3, then the DDMA will "skip" S6 and S7 and proceed directly from S5 to S8. If $\overline{\text{BEC0}}\sim\overline{\text{BEC2}}$ are encoded with the bus error, retry, or reliquish and retry code by the end of S3, one wait cycle will be inserted between S5 and S6. If $\overline{\text{DTACK}}$ is negated and $\overline{\text{BEC0}}\sim\overline{\text{BEC2}}$ are coded to normal at the end of S3, then the DDMA will execute S6 and S7.

S4 No signals change during S4. Data from the DDMA is propagated to the device or memory through external buffers, ect.

S5 $\overline{\text{UDS}}$/A0 and/or $\overline{\text{LDS}}/\overline{\text{DS}}$ is asserted at this time to indicate that valid data is present on the bus and may be latched. The DDMA continues to sample the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}\sim\overline{\text{BEC2}}$ pins to determine when the cycle should be terminated. If $\overline{\text{DTACK}}$ is asserted by the end of S5, then the DDMA will cuntinue into S6 immediately. If $\overline{\text{BEC0}}\sim\overline{\text{BEC2}}$ 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 $\overline{\text{BEC0}}\sim\overline{\text{EBC2}}$ are coded to normal at the end of S5, then the DDMA will insert wait cycles after S5 until a termination signal is asserted and then proceed to S6.

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

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

S8 $\overline{\text{DTC}}$ is asserted to indicate that the bus cycle has been successfully terminated. $\overline{\text{DTC}}$ will be asserted if a $\overline{\text{BEC0}}\sim\overline{\text{BEC2}}$ coding of bus error, retry, reliquish and retry, or reset was used to terminate the cycle.

S9 $\overline{\text{AS}}$ is negated to indicate that the cycle has terminated. $\overline{\text{UDS}}$/A0 and/or $\overline{\text{LDS}}/\overline{\text{DS}}$ 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 $\overline{\text{BEC0}}\sim\overline{\text{BEC2}}$.

S9+1 This may be S0 of the following cycle, if te DDMA is ready to execute another bus cycle immediately. The FC0~FC1, A1~A7, and $\overline{\text{UDA}}$/A0 (when used as an address line) pins will all be changed to their next states. If the DDMA is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DDMA 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.

4.3.2 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 DDMA. 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. Differeces between the two protocol operations will be noted in the signal descriptions below.

When the DDMA 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 $\overline{\text{ACK}}$, $\overline{\text{PCL}}$ (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 DDMA and the device functions in the same as an MPU as far as the memory is concerned.

4.3.2.1 Single Address Read

During this type of a DMA cycle, the DDMA will control the transfer of data from memory to a device. Figure 4.8 shows the functional timing for an even byte read on a 16-bit bus to an 8-bit device.

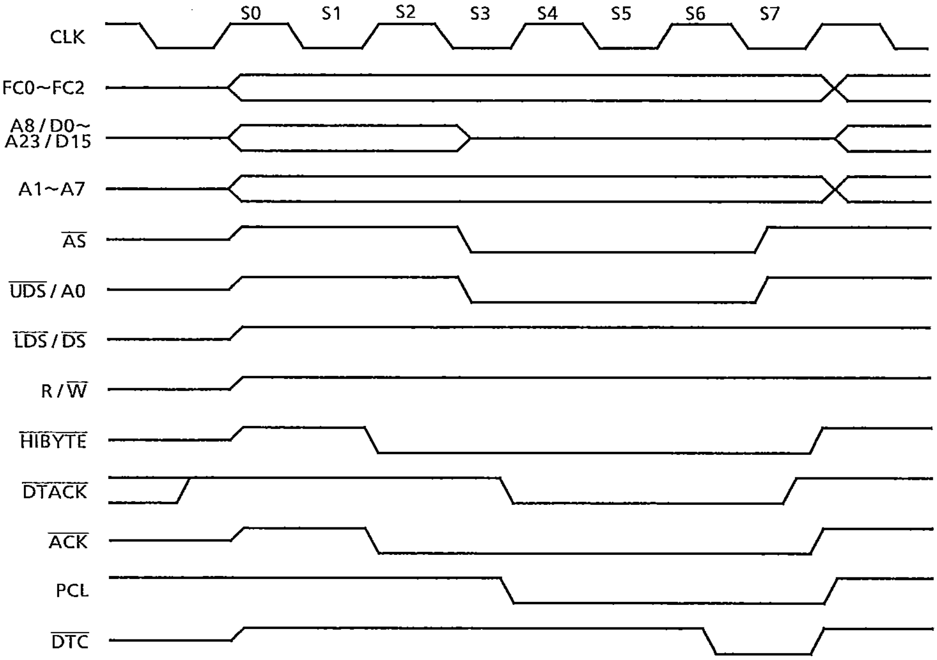


Figure 4.8 Single Address Read Cycle Timing Diagram

The timing for a word or odd byte read on a 16-bit bus to an 8-bit or 16-bit device or any read on an 8-bit bus to an 8-bit device is identical, with the encoding of $\overline{\text{UDS}}$/A0 and $\overline{\text{LDS}}$/$\overline{\text{DS}}$ 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 4.8) .

S0 The DDMA drives the FC0~FC2, A1~A7, and A8/D0~A23/D15 pins with data from the memory function code and memory address registers 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. Also, if an 8-bit bus is being used, $\overline{\text{UDS}}$/A0 will be driven as an address signal.

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. $\overline{\text{UDS}}$/A0 (on a 16-bit bus) and/or $\overline{\text{LDS}}$/$\overline{\text{DS}}$ 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 begining and $\overline{\text{DONE}}$ will be asserted if this is the last transfer for the current channel operation. If the device port size is eight bits, the memory size is 16 bits, and the address is even, $\overline{\mathrm{HIBYTE}}$ will be asserted to cause the memory data from D8~D15 to be driven onto D0~D7 through an external buffer (in this case, $\overline{\mathrm{UDS}}$/A0 will be asserted and $\overline{\mathrm{LDS/DS}}$ will be negated) .

S3 The address/data pins are three stated and the $\overline{\mathrm{DTACK}}$ and $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ pins will be sampled to determine when the cycle should be terminated. If $\overline{\mathrm{DTACK}}$ is asserted by the end of S3, then the DDMA may continue into S4 immediately. If $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ 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 amd S4. If $\overline{\mathrm{DTACK}}$ is negated and $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ are coded to normal at the end of S3, then the DDMA will insert wait cycles after S3 until a termination signal is asserted and then proceed to S4.

If the decice 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 DDMA to terminate the bus cycle, the memory must present a termination signal and the device must drive PCL low before S4 will start. In this way, the device can monitor the $\overline{\mathrm{DTACK}}$ 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 DDMA will proceed to S4.

S4 No signals change during S4. The memory may be placing data on the bus at this time. The DDMA is internally synchronizing the $\overline{\mathrm{DTACK}}$ and $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ 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 DDMA continues to synchronize the $\overline{\mathrm{DTACK}}$, $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$, and PCL signals.

S6 DTC is asserted to the device at this time to indicate that the cycle is being terminated normally and valid data may be latched from the bus. $\overline{\mathrm{DTC}}$ will be asserted if a $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S7 $\overline{\mathrm{AS}}$ is negated to indicate to the memory that the cycle has terminated. $\overline{\mathrm{UDS}}$/A0 and/or $\overline{\mathrm{LDS/DS}}$ will also negate at this and the memory may respond by negating $\overline{\mathrm{DTACK}}$ and/or $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$.

S7+1 This may be S0 of the next bus cycle, if the DDMA is ready to execute another bus cycle immediately. The FC0~FC2, A1~A7, and $\overline{\mathrm{UDS}}$/A0 (when used as an address lines) pins will all be changed to their next states. If the DDMA is ready to start the next bus cycle, the next state will be the address of the next memory location. If the DDMA is not ready to start a bus cycle, these lines will be three stated. $\overline{\text{ACK}}$, $\overline{\text{DTC}}$, $\overline{\text{DONE}}$, and $\overline{\text{HIBYTE}}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has complete 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 DDMA will control the transfer of data from a device to memory. Figure 4.9 shows the functional timing for an even byte write from an 8-bit device to a 16-bit memory. The timing for a word or odd byte write from an 8-bit or 16-bit device to a 16-bit memory or any write from an 8-bit device to 8-bit memory is identical, with the encoding of $\overline{\text{UDS}}$/A0 and $\overline{\text{LDS}}/\overline{\text{DS}}$ selecting the proper byte(s) , and $\overline{\text{HIBYTE}}$ and R/$\overline{\text{W}}$ controlling the flow of data from D0~D7 to D8~D15 if necessary (which will only occur for the case shown in Figure 4.9) .

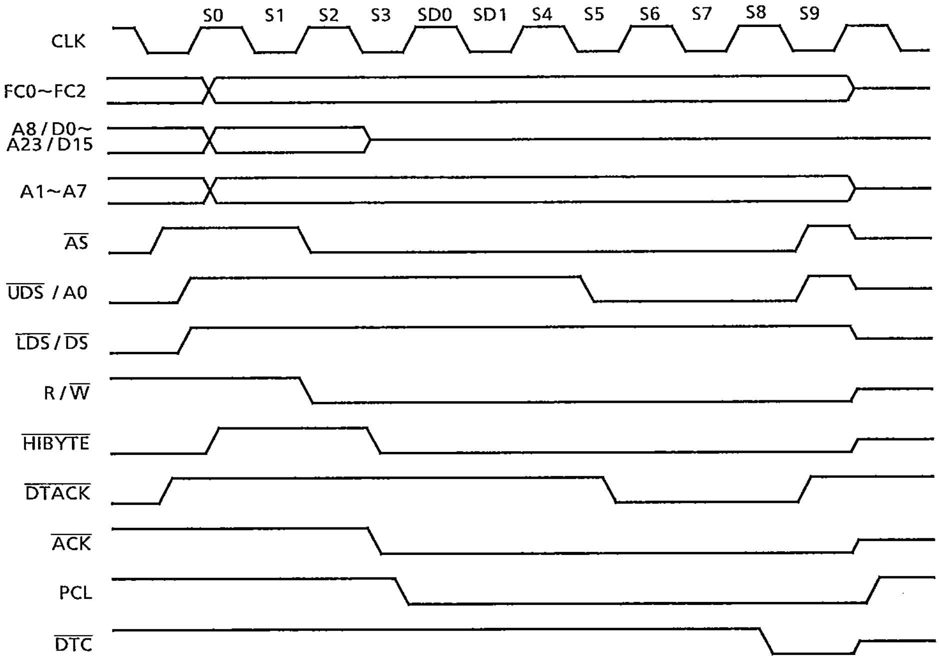

S0
S1
S2
S3
SD0
SD1
S4
S5
S6
S7
S8
S9
CLK
FC0~FC2
A8/D0~ A23/D15
A1~A7
$\overline{\text{AS}}$
$\overline{\text{UDS}}$ / A0
$\overline{\text{LDS}}$ / $\overline{\text{DS}}$
R / $\overline{\text{W}}$
$\overline{\text{HIBYTE}}$
$\overline{\text{DTACK}}$
$\overline{\text{ACK}}$
PCL
$\overline{\text{DTC}}$

Figure 4.9 Single Address Write Cycle Timing Diagram

S0 The DDMA drives the FC0~FC2, A1~A7, and A8/D0~A23/D15 pins with data from the memory function code and memory address registers of the current channel and $\overline{\mathrm{UAS}}$ is asserted to enable the external address latches. $\overline{\mathrm{DBEN}}$ 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. Also, if an 8-bit bus is being used, $\overline{\mathrm{UDS}}$/A0 will be driven as an address signal.

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, 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.

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

S3 The address/data pins are three stated and $\overline{\mathrm{ACK}}$ is asserted to indicate that the device should place data on the bus. If the device port size is eight bits, memory size is 16 bits, and the address is even, $\overline{\mathrm{HIBYTE}}$ will be asserted to cause the device data on D0~D7 to be driven onto D8~D15 through an external buffer (in this case, $\overline{\mathrm{UDS}}$/A0 will be asserted and $\overline{\mathrm{LDS}}$/$\overline{\mathrm{DS}}$ will be negated) . If this is the last transfer in a channel operation, the $\overline{\mathrm{DONE}}$ signal will be asserted by the DDMA.

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

The DDMA also begins to sample the $\overline{\mathrm{DTACK}}$ and $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ pins to determine when the cycle should be terminated. If $\overline{\mathrm{DTACK}}$ is asserted by the end of S3, then the DDMA will “skip” S6 and S7 and proceed directly from S5 to S8. If $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ are encoded with the bus error, retry, relinquish and retry code by the end of S3, one wait cycle will be inserted between S5 and S6. If $\overline{\mathrm{DTACK}}$ is negated and $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ are coded to normal at the end of S3, then the DDMA 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 DDMA 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 perpare to drive data onto the bus. The DDMA continues to synchronize the PCL signal. If $\overline{\text{DTACK}}$ is asserted by the end of SD1, then the DDMA 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 DDMA 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 $\overline{\text{UDS}}$/A0 and/or $\overline{\text{LDS/DS}}$ 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 DDMA 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 DDMA will insert wait cycles after S5 until a termination signal is asseted and then proceed to S6.

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

S7 No signals change during S7. The DDMA is 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. $\overline{\text{UDS}}$/A0 and/or $\overline{\text{LDS/DS}}$ 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 DDMAis ready to execute another bus cycle immediately. The FC0~FC2, A1~A7, and $\overline{\text{UDS}}$/A0 (when used as an address line) will all be changed to their next states. If the DDMA is ready to start the next bus cycle, the next state will be the address of the next memory location. If the DDMA 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 acknowlege and read 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 te 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 set-up and hold times are met for the memory with respect to the data strobes.

4.3.2.3 High-Byte Operation

During single-address transfers between an 8-bit device and a 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 $\overline{\text{UDS}}$/A0 will be asserted and $\overline{\text{LDS}}/\overline{\text{DS}}$ 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 TLCS-68000 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 a 16-bit memory: even and odd read, and even and odd write cycles. Figure 4.10 shows the data flow for the four cases.

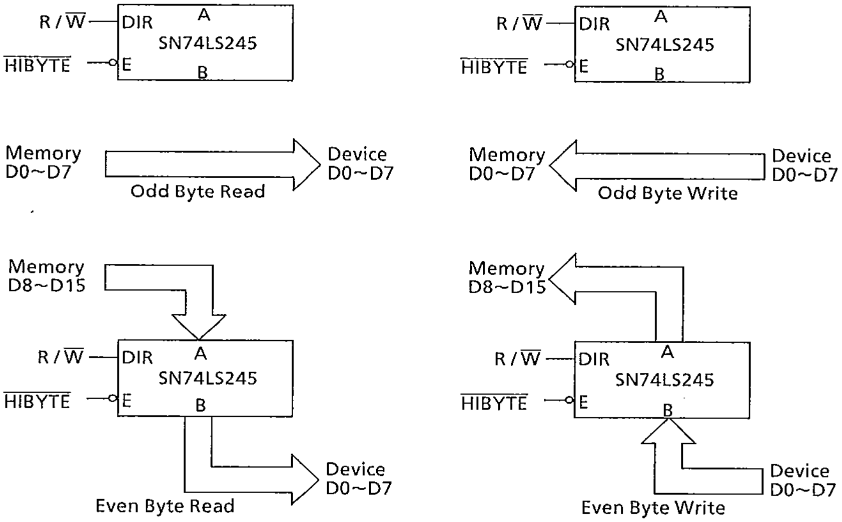


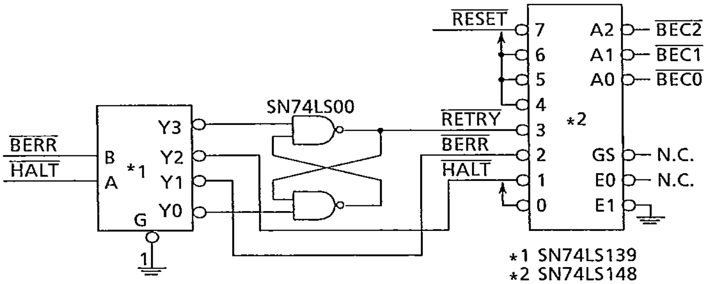
Figure 4.10 Implicitly Address 8-Bit Device with 16-Bit Memory Data Flow

4.4 BUS EXCEPTION CONTROL

In order to fully support the TLCS-68000 bus architecture, the DDMA uses three encoded inputs, $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$, to detect abnormal bus cycle termination conditions. These three lines function in a similar manner to the $\overline{\text{RESET}}$, $\overline{\text{HALT}}$, and $\overline{\text{BERR}}$ signals on an TLCS-68000 processor, but have different functionality than the processor counterparts. Table 4.1 shows the definition for each encoding of the $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$ pins and Figure 4.11 illustrates how the three inputs might be generated from signals normally present in an TLCS-68000 system.

Table 4.1 BEC Encoding Definitions

BEC2	BEC1	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	Relinguish and Retry
L	H	L	(Undefined Reserved)
L	L	H	(Undefined Reserved)
L	L	L	Reset



*1 SN74LS139
*2 SN74LS148

Figure 4.11 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 4.12. 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 that level is accepted as valid and the DDMA may take action on it. If the two encodings are different, the DDMA 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 deboucing 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.

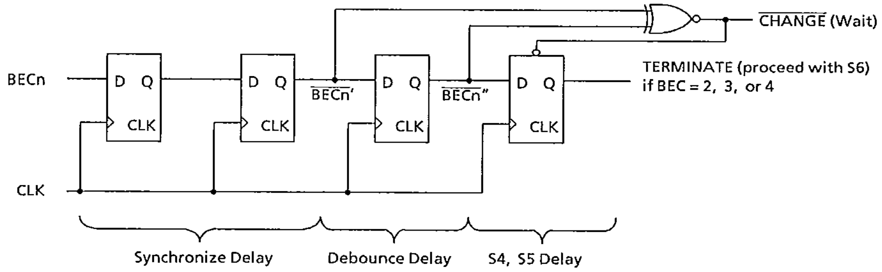


Figure 4.12 $\overline{\text{BEC}}$ Synchronizer Function Diagram

As an example of the debouncer operation, consider the timing diagram in Figure 4.13. When the SN74LS148 output change from halt to bus error, they may transition through retry at the time when the DDMA is latching the $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$ 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 DDMA 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 $\overline{\text{DTACK}}$ signal. Also, note that if $\overline{\text{DTACK}}$ and bus error, retry, or relinquish and retry are asserted at the same time and the asynchronous input setup time is met for the $\overline{\text{DTACK}}$ and $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$ signals, then the bus cycle will be two clock cycles longer than if it had been terminated with $\overline{\text{DTACK}}$ alone and $\overline{\text{DTACK}}$ will be ignored. This behavior is different from what an TLCS-68000 processor will exhibit since the DDMA must supporess the assertion of DTC and prevent the internal address counters from incrementing if necessary.

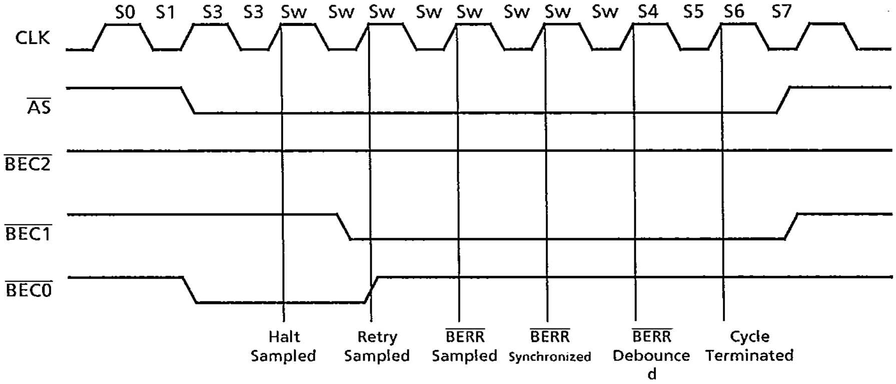


Figure 4.13 BEC Debouncer Timing Diagram

4.4.2 Bus Exception Control Functions

The three bus exception control signals allow eitht different bus termination conditions to be signaled to the DDMA as shown in Table 4.1. The function of each encoding is described below. In describing the encoding of the $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$ pins, $\overline{\text{BEC0}}$ is the least significant bit and $\overline{\text{BEC2}}$ 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 $\overline{\text{BEC}}$ inputs,the DDMA will enter the idle mode or DMA waiting modes as soon as the current bus cycle is terminated and remain in that state until the $\overline{\text{BEC}}$ encoding returns to normal. Another system bus master may access the DDMA registers while it is in the idle mode, if it can gain mastership of the DDMA bus. In this manner, the DDMA can be halted by external hardware to enable the system processor to modify the DDMA registers and alter or monitor channel operations if necessary.

4.4.2.1 Normal Termination

If a zero (0) is encoded on the BEC pins, the DDMA will operate in the normal mode and $\overline{\text{DTACK}}$ is uesd to terminate bus cycles.

4.4.2.2 Halt

If a one (1) is encoded on the $\overline{\text{BEC}}$ pins, the DDMA will be halted when the current bus cycle is termnated by the assertion of $\overline{\text{DTACK}}$. 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 DDMA halts in response to the halt input, it will release ownership of the bus and enter the idle state until the $\overline{\text{BEC}}$ coding is returned to normal, at which time it will arbitrate for the bus and continue DMA operations if necessary. Figure 4.14 shows the timing of the halt operation. Note that if the halt encoding is asserted on the $\overline{\text{BEC}}$ pins at the same time that $\overline{\text{DTACK}}$ is asserted, the cycle will terminate one clock cycle later than if $\overline{\text{DTACK}}$ alone were asserted due to the debounce delay on the $\overline{\text{BEC}}$ inputs. Also, when the $\overline{\text{BEC}}$ pins are returned to the normal conding, the DDMA will wait three clock cycles before asserting $\overline{\text{BR}}$ to rearbitrate for the bus.

4.4.2.3 Bus Error

If a two (2) is encoded on the $\overline{\text{BEC}}$ pins, the DDMA 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 $\overline{\text{BEC}}$ encoding is returned to normal, the DDMA will wait for five clock cycles 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. Figure 4.15 shows the timing of a bus error operation.

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

4.4.2.4 Retry

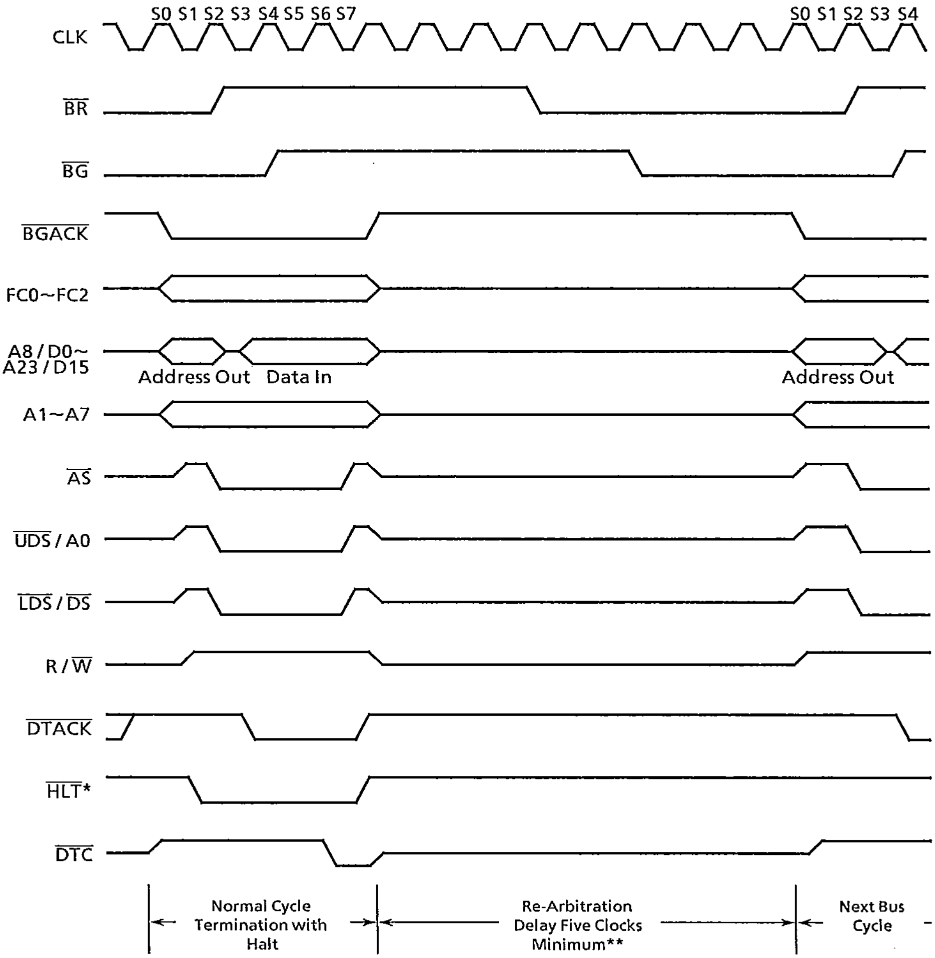
If a three (3) is encoded on the $\overline{\text{BEC}}$ pins,the DDMA will terminate the current bus cycle, enter the DDMA waiting mode, and $\overline{\text{BGACK}}$ will remain asserted so that the DDMA retains ownership of the bus. When the $\overline{\text{BEC}}$ encoding is returned to normal, the DDMA will wait for three clocks 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 retied before the normal $\overline{\text{BEC}}$ encoding is synchronized and debounced, the retry and any remaining bus cycles required by the current operand transfer will be executed before the higher priority request will be serviced. Figure 4-16 shows the functional timing of a retry operation.

If a bus cycle is terminated with a retry, $\overline{\text{DTC}}$ will not be asserted so that a preipheral device will not increment any operand counters. The DDMA 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 $\overline{\text{BEC}}$ pins, the DDMA will terminate the current bus cycle and relinquish ownership of the bus. When the $\overline{\text{BEC}}$ encoding is returned to normal, the DDMA will wait for four clocks, re-arbirate 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 DDMA regains control of the bus, then the retry and remaining bus cycles required to complete the current operand transfer will be executed before the higher priority request will be serviced. Figure 4.17 shows the functional timing of a relinquish and retry operation.

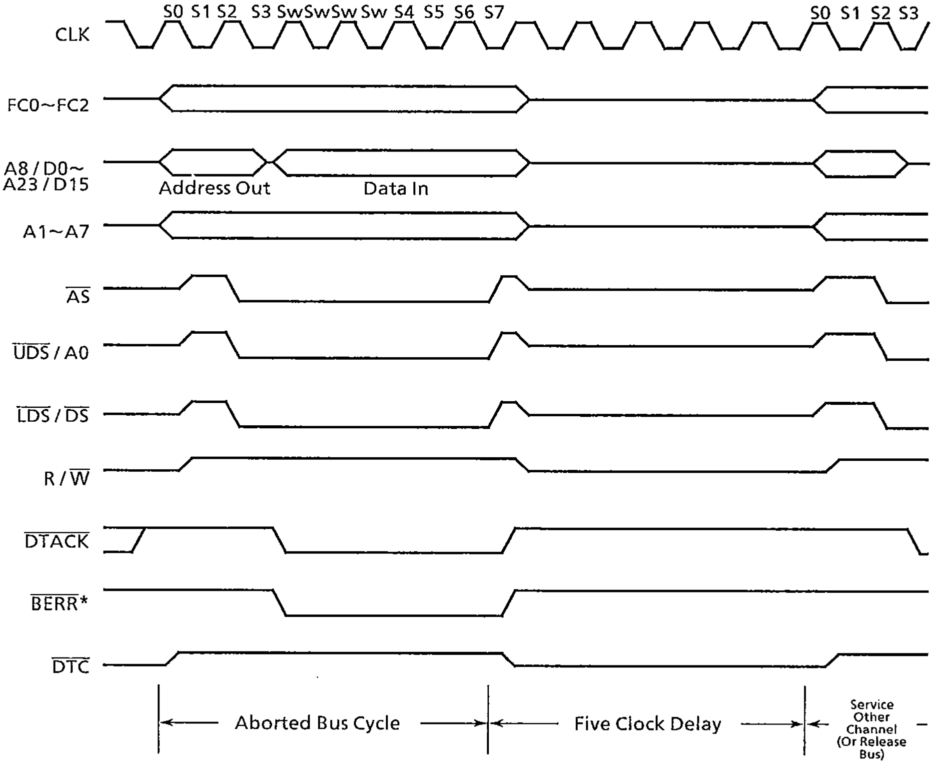
If a bus cycle terminated with a relinquish and retry, $\overline{DTC}$ will not be asserted so that a peripheral device will not increment any operand counters. The DDMA will not increment or decrement the MAR, MTCR, or DAR and the internal data holding register will not be affected.



* $\overline{BEC2}$~$\overline{BEC0}$ encoded to signal either normal (high) or halt (low).

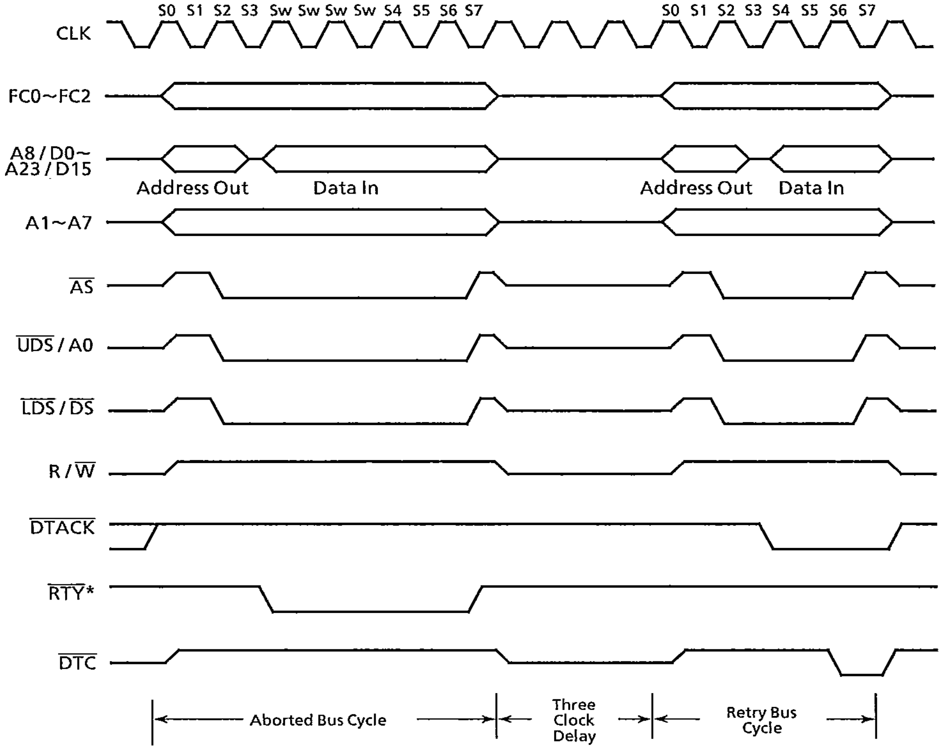
** This minimum delay will occur if $\overline{BG}$ is asserted prior to the clock edge on which $\overline{BR}$ is asserted.

Figure 4.14 Halt Operation Timing Diagram



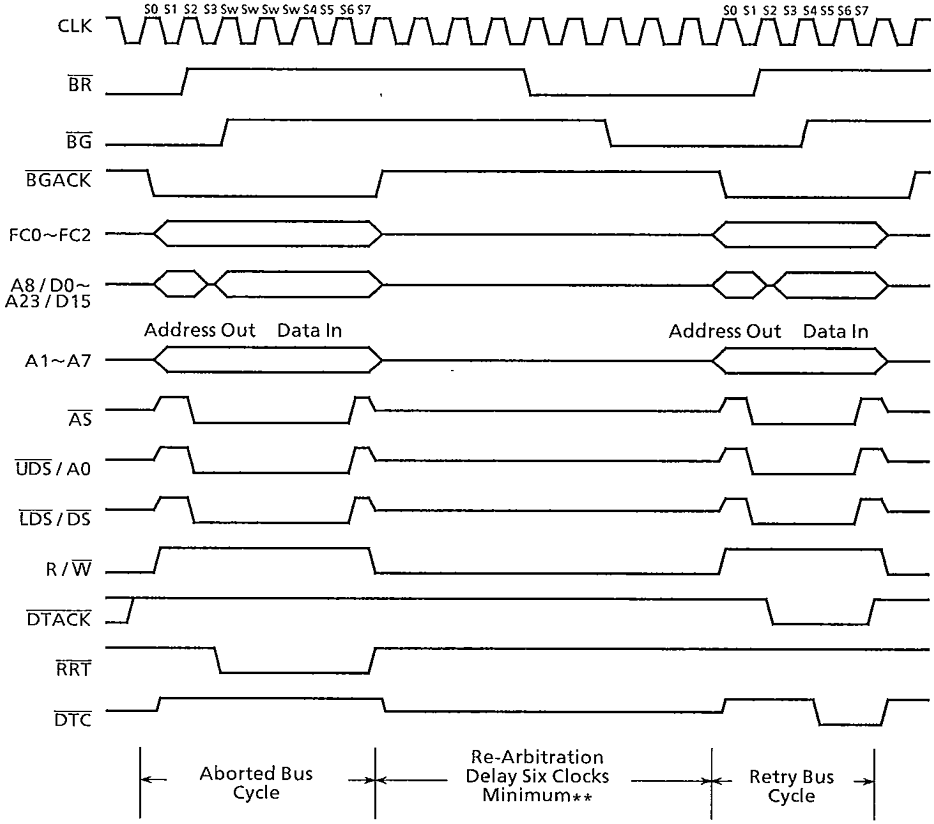
* $\overline{BEC2}$~$\overline{BEC0}$ encoded to signal either normal (high) or bus error (low).

Figure 4.15 Bus Error Operation Timing Diagram



* $\overline{\text{BEC2}}$~$\overline{\text{BEC0}}$ encoded to signal either normal (high) or retry (low).

Figure 4.16 Retry Operation Timing Diagram



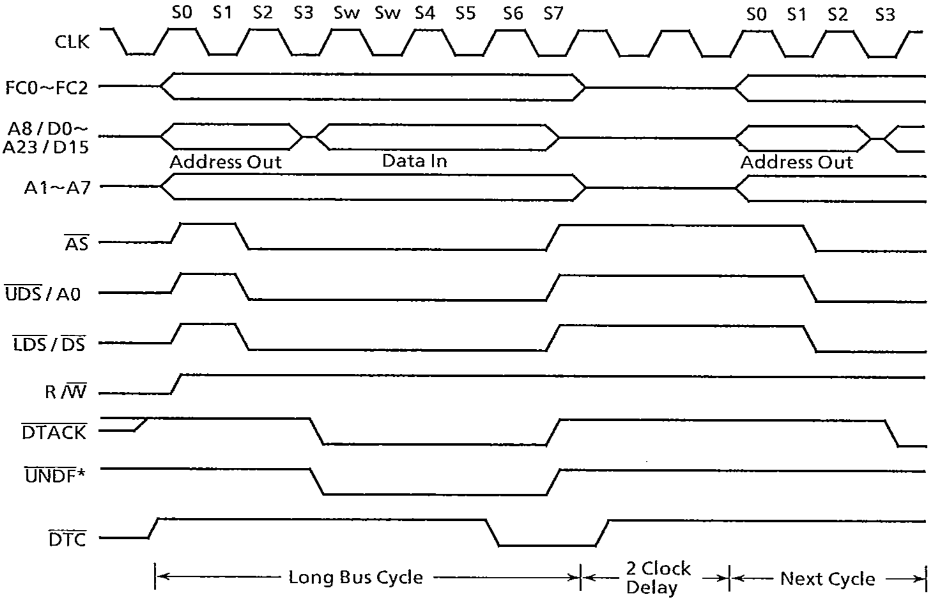
* $\overline{BEC2}$~$\overline{BEC0}$ encoded to signal either normal (high) or relinquish and retry (low).

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

Figure 4.17 Relinquish and Retry Operation Timing Diagram

4.4.2.6 Undefined BEC Codes

If a five (5) or six (6) is encoded on the $\overline{\text{BEC}}$ pins, the DDMA will synchronize and debounce the inputs as usual, but no action will be taken. The affect that one of these encodings will have is to lengthen a bus cycle by one debounce delay since the DDMA must wait until the internal $\overline{\text{BEC}}$ signals are stable for one clock before deciding to ignore a level five or six input. Furthermore, the DDMA will not start to service any futher operand transfer requests until one synchronization delay plus one debounce delay after the $\overline{\text{BEC}}$ pins are returned to the normal coding. However, it will complete the current operand transfer and may relinquish bus mastership if no futher operand requests are pending, even though the undefined encoding remains asserted. Figure 4.18 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).

Figure 4.18 Undefined BEC Operation Timing Diagram

4.4.2.7 Reset

If a seven (7) is encoded on the $\overline{\text{BEC}}$ pins, the DDMA 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.

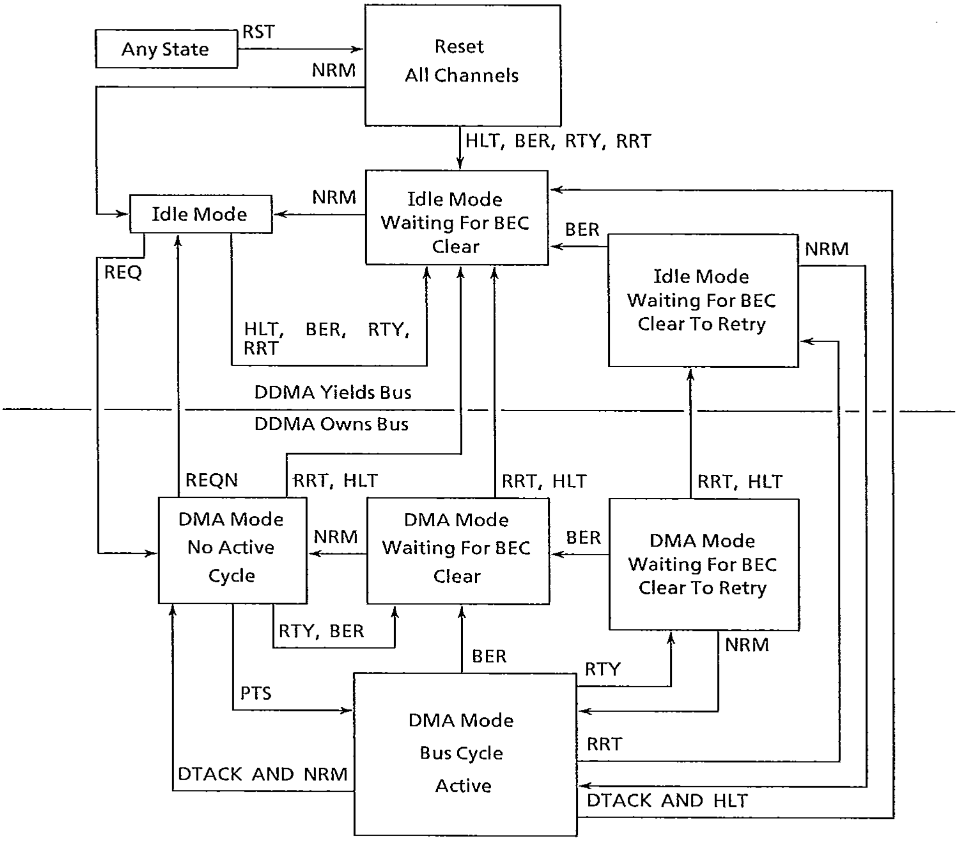
Note that when reset is debounced and asserted internally, the DDMA will terminate any active bus cycle in an orderly fashion (i.e., in S7 or S9) . Thus, if reset is asserted one clock cycle before a bus cycle is to begin, the reset will be internally synchronized in S2. The DDMA will still assert $\overline{\text{AS}}$ as usual, proceed through S3, S4, etc., and negate $\overline{\text{AS}}$ in S7 so that the $\overline{\text{AS}}$ low time will meet specification #62. If $\overline{\text{AS}}$ is used to generate a row address strobe ($\overline{\text{RAS}}$) to dynamic RAMs, this will help guarantee the minimum $\overline{\text{RAS}}$ low time so that data in the RAM array will not be destroyed.

4.4.3 Bus Exception Control Summary

Figure 4.19 shows the bus exception control flow diagram for the DDMA in the idle and DMA modes of operation.

4.5 BUS ARBITARTION

Once the system processor has initialized and started a DDMA channel and an operand transfer request has been made pending, either by an external device or the internal request generator, the DDMA will use the TLCS-68000 bus arbitration protocol to request bus mastership before entering the DMA mode of operation. The following paragraphs describe the arbitration protcol used by the DDMA and several special cases of operation relative to bus arbitration. Figure 4.20 shows a functional timing diagram of a typical bus arbitration sequence.



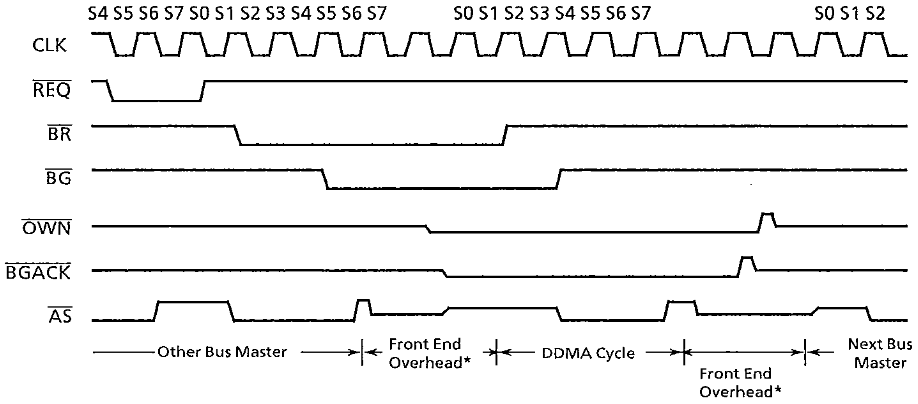
BEC Conditions :

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

Figure 4.19 Bus Exception Control Flow Diagram



* The front end overhead will be one clock cycle less than shown if only one channel is active.

** The back end overhead will be one clock cycle less than shown if a DMA cycle is terminated with halt or relinquish and retry.

Figure 4.20 Bus Arbitration Timing Diagram

4.5.1 Requesting and Receiving the Bus

When an active channel has an operand transfer request pending, the DDMA will request bus mastership by asserting the $\overline{\text{BR}}$ signal. An external bus arbiter (either a separate unit such as the TMP68452 Bus Arbitartion Module or the arbiter built into an TLCS-68000 processor) will then assert $\overline{\text{BG}}$ to indicate that bus mastership will belong to the DDMA as soon as the current bus master has released the bus. The DDMA then monitors the $\overline{\text{AS}}$ and $\overline{\text{BGACK}}$ 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 DDMA will assert $\overline{\text{OWN}}$ after a synchronization delay plus one-half clock cycle and then assert $\overline{\text{BGACK}}$ to indicate that it has taken control of the bus. One clock after $\overline{\text{BGACK}}$ is asserted, $\overline{\text{BR}}$ 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 DDMA will release control of the bus by negating $\overline{\text{BGACK}}$ and will then negate $\overline{\text{OWN}}$ one-half clock cycle later. Note that $\overline{\text{OWN}}$ may be used to control a bidirectional buffer for $\overline{\text{BGACK}}$ since it is asserted before and negated after $\overline{\text{BGACK}}$ is asserted and negated respectively.

4.5.2 Bus Overhead Time

In asynchronous bus systems such as those defined for the TLCS-68000 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 DDMA must be known to predict system behavior. The following paragraphs describe three types of overhead periods: those associated with the DDMA taking control of the bus, those in between successive operand transfers, and those that occur when the DDMA is releasing control of the bus to another bus master. In all of these discussions, the system is assumed to consist of a singe TLCS-68000 processor and a single DDMA using the same clock so that all arbitration delay are controlled by those two devices and can be predicted, as shown in Figure 4.20.

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 $\overline{\text{AS}}$ (in S7 of the MPU cycle) to when the DDMA starts a bus cycle by placing function code and address information on the bus (in S0 of the DMA cycle) . It is assumed that $\overline{\text{BG}}$ is asserted and $\overline{\text{BGACK}}$ is negated prior to the negation of $\overline{\text{AS}}$ by the MPU so that no additional synchronization delays are introduced by those signals. After one synchronization delay plus one-half clock cycle,the DDMA will assert $\overline{\text{OWN}}$ and one-half clock later assert $\overline{\text{BGACK}}$ to indicate that it is assuming bus mastership and begins to drive the function code and address lines.

When the DDMA asserts $\overline{\text{BGACK}}$ to assume control of the bus, it may also perform an internal channel arbitration to determine which channel will be serviced. This internal arbitartion is required to be certain that the highest priority channel is serviced first, even if a lower priority channel issued a request first and started the bus arbitartion sequence. If only one channel is active when the DDMA receives the bus, this arbitration is unnecessary and will not be perfomed. However, if both channels are active, then this arbitration will introduce one additional clock cycle of delay before the DDMA asserts $\overline{\text{AS}}$. If the lower priority channel issued a request first and started the bus arbitration sequence, but the higher priority channel also has an operand transfer request pending by the time $\overline{\text{BGACK}}$ is asserted, then the function code and address/data lines will first be driven with addresses from the lower priority channel for one clock cycle and then change to the addresss from the higher priority channel. If the higher priority channel had issued the original request, the one clock arbitration delay will still occur but the function code and address lines will remain stable for one extra clock before $\overline{\text{AS}}$ is asserted. Neither of these cases will cause an erroneous address decode since $\overline{\text{AS}}$ will not be asserted until the function code and address pins are stable.

Taking into consideration he above factors of internal arbitration and synchronization delays, the best case front-end overhead time is two clock cycles if $\overline{\text{AS}}$ is sampled as negated by the DDMA at the end of S7 in the MPU cycle and only one channel is active. The worst case front-end overhead is four clock cycles if $\overline{\mathrm{AS}}$ is not negated until after the rising edge of S7 in the MPU cycle and both channels are active.

4.5.2.2 Inter-Cycle Overhead

This is the overhead delay that will occur between successive DMA bus cycles for either or both of the channeles while the DDMA maintains ownership of bus. Such a delay will occur if a channel with priority equal to or higher than the priority of the channel being serviced has an operand transfer request pending at the completion of an operand transfer, or when a new cycle steal request is asserted for the current channel.

The inter-cycle overhead time is illustrated in Figure 4.21 where the three cases of zero, one, and two clock cycle are shown. If a channel switch is not necessary and the current channel has further cycle steal transfer requests asserted by E1 (where E1 is the rising clock edge one clock before the edge on which $\overline{\mathrm{DTC}}$ is asserted) , then the inter-cycle overhead time is zero. The second case of one clock overhead will occur when a channel switch request is asserted by E1 in the current bus cycle, or if a cycle steal request for the current channel is not asserted until E2 (where E2 is the risiing clock edge on which $\overline{\mathrm{DTC}}$ is asserted) . The worst case will occur if a channel switch request is not asserted until after E1, but before E2. If no requsets are asserted for either channel when the DDMA reaches E2, it will release ownership of the bus and a minimum of the back-end and front-end overhead times will occur before another operand transfer will be able to start (plus any bus cycle time used by another bus master before the DDMA regains control of the bus) . For the one and two clock overhead cases, the DDMA may begin to drive the function code and address pins with the values for the next operand of the current channel for one clock cycle before changing to the values for the higher priority channel. This will not cause in valid address decoding since AS will not be asserted until the function code and address pins are stable.

One other case of inter-cycle overhead time is related to the reload mode of operation and is very similar to the overhead time for channel switches. A more detailed description of the delays associated with this type of operation is given in "5.5 SPECIAL CHANNEL OPERATIONS".

4.5.2.3 Back-End Overhead

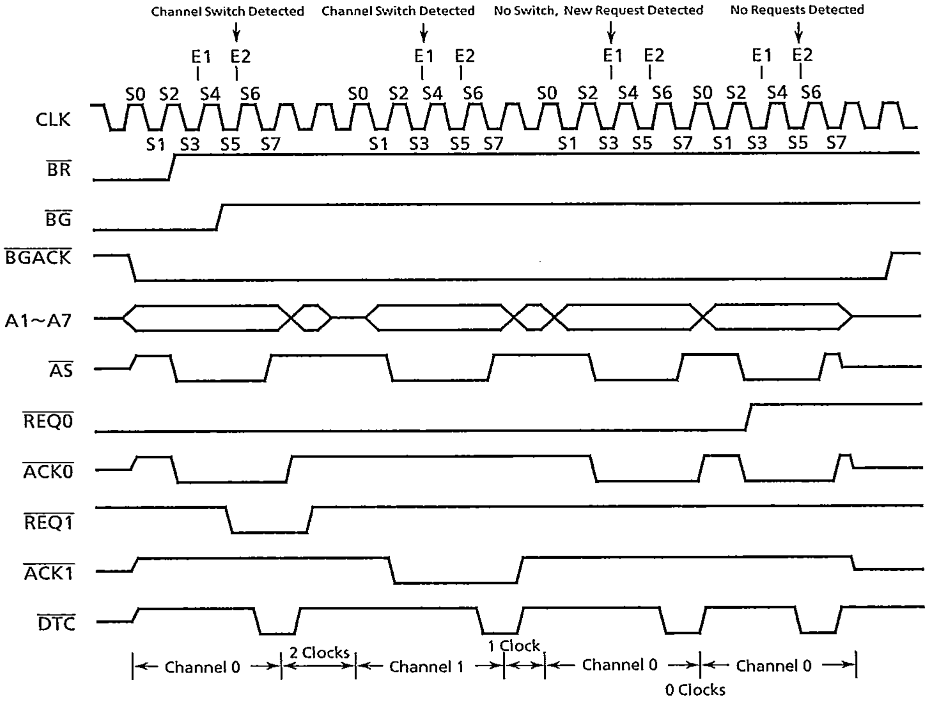
This overhead time is the delay between when the DDMA has completed all pending operand transfers and releases the bus, plus the synchronization of $\overline{\mathrm{BGACK}}$ negated by the MPU. When the DDMA has completed the last pending bus cycle in S7 for a DMA read or S9 for a DMA write, it will negated $\overline{\mathrm{BGACK}}$ after a one clock cycle delay. The MPU will then wait one synchronization delay plus one-half clock cycle before S0 of the next MPU cycle. Given these restraints, the best case back-end overhead time is three clock cycles and the worst case is four clock cycles.

4.5.2.4 Overhead Due to BEC Operation

When a $\overline{\text{BEC}}$ encoding of bus error, retry, or undefined is asserted during a bus cycle, the DDMA will introduce delays after it terminates the bus cycle while it performs internal operations before starting the next bus cycle or releasing ownership of the bus. If a bus cycle is terminated with a halt or relinquish and retry BEC encoding, then the back-end overhead time will also be affected.

If a bus cycle is terminated with a bus error, there will be a five clock minimum delay before the DDMA starts the next bus cycle or releases ownership of the bus. If a bus cycle is terminated with a retry, there will be a three clock minimum delay before the DDMA start the retry operation. If a bus cycle is terminated with an undefined code, there will be a two clock minimum delay before the DDMA continues operation. All of these minimum delay times assume that the $\overline{\text{BEC}}$ encoding is returned to normal as soon as $\overline{\text{AS}}$ is negated and that encoding is valid for the required setup time before the next rising edge of the clock (the end of S7 for read cycles or S9 for write cycles) .

If a bus cycle terminated with a relinquish and retry or a $\overline{\text{DTACK}}$/halt, the back-end overhead time will be one cycle less than for normal operation.



In this example, CH0 is using burst requests and is at priority 1, CH1 is using cycles steal requests and is at priority 0.

Figure 4.21 Inter-Cycle Overhead Timing Diagram

4.6 ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION

Because the TLCS-68000 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 aserts output 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 DDMA, there can be two distinct asynchronous or synchronous interfaces operating simultaneously. One is the TLCS-68000 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 acknowlege and ready protocol. An implicitly addressed peripheral using the device with acknowledge-only protocol must always operate in synchronization with the DDMA.

4.6.1 Asynchronous Operation

To achieve clock signal independence at a system level, the DDMA can be used in an asynchronous manner. This entails using only the bus handshake lines ($\overline{\text{AS}}$, $\overline{\text{UDS}}$/A0, $\overline{\text{LDS}}/\overline{\text{DS}}$, $\overline{\text{DTACK}}$, and $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$) and, as needed, the device control lines ($\overline{\text{REQ}}$c, $\overline{\text{ACK}}$c, PCLc, $\overline{\text{DTC}}$, and $\overline{\text{DONE}}$) . Using this method, $\overline{\text{AS}}$ (and $\overline{\text{ACK}}$c 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 $\overline{\text{DTACK}}$ (and possibly ready for single address reads) . The DDMA then acknowledges the successfull termination of the bus cycle by asserting $\overline{\text{DTC}}$ for one clock cycle and negating $\overline{\text{AS}}$, $\overline{\text{ACK}}$c, 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 $\overline{\text{DTACK}}$ signal is allowed to be asserted before the data from a slave device is valid on a read cycle (either signal or dual address) . The length of time that $\overline{\text{DTACK}}$ may precede data during a dual address read is given as parameter #73 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 DDMA synchronizes the $\overline{\text{DTACK}}$ signal before asserting $\overline{\text{DTC}}$ and latching data. The length of the time that $\overline{\text{DTACK}}$ 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, since the DDMA will not negate the data strobes and terminate the cycle for a minimum of specification #107.

The ready signal (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 TLCS-68000 Family processor, data must be valid prior to the assertion of the data strobe(s) , which will occur a minimum of specification #110 after the assertion of ready.

If a bus exception control condition other than normal is used to terminate a bus cycle, the $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$ levels must be valid for specification #75 prior to the assertion of $\overline{\text{DTACK}}$ in order for the normal termination to be preempted and the exceptional action taken. Since a peripheral system will normally generate a single signal that is used by the DDMA interface logic to generate the $\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$ signals, the delay from assertion of the signal by the peripheral to the assertion of the proper 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 singals. In order to allow the system designer to predict when the DDMA will recognize the assertion or negation of input signals, the asynchronous input setup time is given as parameter #8. 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 DDMA, depending on several factors such as individual device characteristics, clock wave form and frequency, ambient temperature, supply voltage, and system age.

In order to allow the DDMA 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 DDMA 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 DDMA will latch the value of the data bus on the next rising edge of the clock. Thus, if the $\overline{\text{DTACK}}$ and data setup time

(#8 and #57) are both guaranteed to be met by the system, then specification #73 may be exceeded.

For single address read cycle, a peripheral device should not latch data before $\overline{\mathrm{DTC}}$ 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 from the rising edge of S3 and $\overline{\mathrm{DTACK}}$ 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 $\overline{\mathrm{DTACK}}$ will meet the setup time to the end of S3, then the peripheral should wait until $\overline{\mathrm{DTACK}}$ has been asserted for at least one clock before asserting ready prior to the setup time from 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 (PCLc) before it will have valid data 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 DDMA will assert the data strobe(s) one and one-half clocks later. If the peripheral device will have valid data on the bus to give the proper setup time prior to the assertion of the data strobe(s) , then specification #110 may be exceeded.

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 DDMA clock. During a read cycle, the device must be prepared to accept input data when $\overline{\mathrm{DTC}}$ is asserted and can not 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 DDMA asserts $\overline{\mathrm{ACK}}$c, with some setup time prior to the data strobe(s) assertion one clock later.

If a BEC encoding other than normal is used to terminate a bus cycle in which $\overline{\mathrm{DTACK}}$ is also asserted, then the BEC encoding must be recognized as valid on the same clock edge that $\overline{\mathrm{DTACK}}$ is first recognized as asserted. Thus, if $\overline{\mathrm{DTACK}}$ and $\overline{\mathrm{BEC0}}$~$\overline{\mathrm{BEC2}}$ are all valid prior to the setup time from the same rising edge of the clock (S3 for a minimum length cycle) , then specification #75 may be violated, the normal termination is guaranteed to be preempted, and the exceptional action is taken.

5. OPERATIONAL DESCRIPTION

This section describes the programmable channel functions available on the DDMA, 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 DDMA 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 subsections 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) .

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. Continue or Reload Operation
   If a channel is initialized for the continue or reload operating 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. In the reload mode, the MFCR, MAR, and MTCR will be loaded from the BFCR, BAR, and BTCR in response to an input signal from the peripheral device.

5. Channel Termination
   A channel operation can be terminated for several resons: 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 sytem 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 cannel 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 DDMA also supports device-to-device transfer operations) and thus, the DDMA 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 locate data anywhere in one of eight 16 megabyte linear address space; and the MTCR which is a 16-bit transfer counter that defines a block size of up to 65, 535 operands (bytes or words) . The MTCR is decremented by the DDMA after each operand transfer until it reaches zero and the MAR may be programmed to be incremented after each operand transfer.

Although the DDMA does not place any format requirements on the data that is transferred during a channel operation, it does use the TLCS-68000 conventions for operand alignment and byte significance. As shown in Figure 5.1, bytes are individually addressable on a 16-bit bus 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 word address. Word data is accessed only on word (even byte) boundaries with the lower addressed byte being most significant. On an 8-bit data bus, the same data organization is used; however, word transfers will require two bus cycles to the memory for each operand. The size field in the OCR is used to select the operand size for a block transfer.

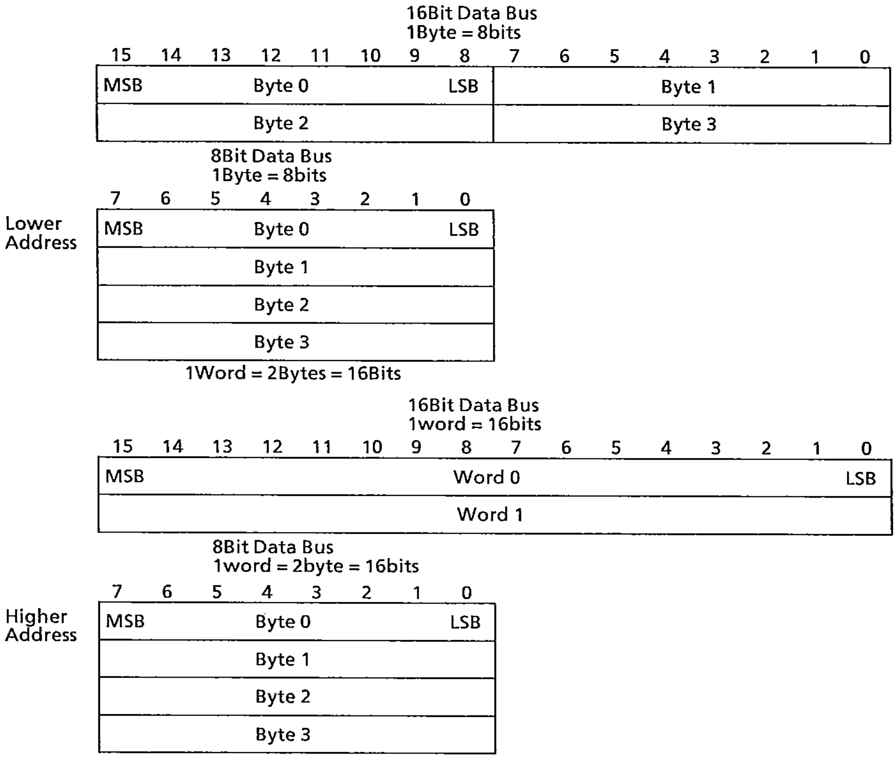


Figure 5.1 Memory Data Formats

5.2.2 Device Description

Three types and two sizes of peripheral devices are supported by the DDMA. The types supported are explicitly addressed TLCS-68000 bus compatible, implicitly addressed with acknowledge only, and implicitly addressed with acknowledge and ready. The type and size of a device is prorgammed 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 DDMA, the device is programmed as TLCS-68000 compatible with a size of the external data bus width.

Also, part of the device description is the functional definition of the PCL input. If the transfer protocol selected is implicitly addressed with acknowledge and ready, the PCL input is used as an active low ready signal. Otherwise, the DCR may be programmed to select one of four other functions for the PCL line: status, interrupt, abort, or reload input. 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 interrupt function is selected, the PCT bit is set and interrupts are enabled, the DDMA 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. When the reload function is selected and the PCT bit is set as the DDMA is ready to start an operand transfer, the bus cycle will be delayed while the BFCR, BAR, and BTCR are transferred to the MFCR, MAR, and MTCR (for more details refer to "5.5.2 Reload Operation") .

In addition to supporting either an 8- or 16-bit memory data bus, the DDMA also supports peripheral device sizes of 8- or 16-bits using either single or dual address transfers of byte or word operands. When all combinations of the above are considered, 16 different transfer operations are possible; however, six of those operations are not legal. Table 5.1 shows a matrix of the possible memory, device, and operand size and transfer type combinations, and whether each is legal or not.

Table 5.1 Memory, Device, and Operand Size Combinations

		Dual Address		Single Address	
		Byte Operand	Word Operand	Byte Operand	Word Operand
8-Bit Memory	8-Bit Device				×
	16-Bit Device	×		×	×
16-Bit Memory	8-Bit Device				×
	16-Bit Device			×	

× Denotes illegal Combinations

5.2.3 Operation Description

A wide variety of block transfer operations are supported by the DDMA by choosing from the available memory and device combinations, 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/$\overline{W}$ single will be driven high of 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 during a transfer operation depends on the programming of the SCR, operand size, memory/device width combination, and transfer protocol used. Both the MAR and DAR can be individually programmed to be incremented 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 and no lager than the memory width (i.e., word operands cannot be transferred from a 16-bit device to an 8-bit memory) . Thus, all operands can be transferred in one bus cycle and only the MAR is incremented (the DAR is not used) . The amount by which the MAR is incremented will be one or two for byte or word operands respectively (if the MAC field in the SCR is programmed for increment) . 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 DDMA will always run at least two and up to four bus cycles to transfer each operand. When the operand size is larger than either the memory width or device port size, the DDMA will run multiple bus cycles to transfer operand parts (bytes) until the entire operand (word) has been transferred. If the source (memory or device) is sammller than the operand size, two read cycles will be executed to fetch each operand; and, if the destination (device or memory) is smaller than the operand size, two write cycles will be executed to store each operand. The entire operand will always be fetched first and then stored, even if the operand size is word and both the memory and device are eight bits wide.

Two exceptions to the above rules are when the combinations of an 8- or 16-bit device, 16-bit memory and byte operand size are used with internal request generation. In this case, the DDMA 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 tarnsfer two operands (bytes) at a time. If an 8-bit device or an odd transfer count is used, two byte device accesses will be performed for each word memory access or the last operand transfer of the operation will use a byte read and a byte write cycle.

If the SCR is programmed to increment for either the MAR or DAR, the manner in which the addtress register(s) is incremented 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, memory, and device sizes in the dual address mode. It should be noted that when an 8-bit device is used with a 16-bit memory, the DAR will be incremented by two or four for byte or word operands respectively so that consecutive odd or even addresses will be accessed. This is consistant with the TLCS-68000 MOVEP instruction so that an 8-bit device may be placed on one-half a 16-bit bus and successive bytes within the device will be accessed. If no increment is selected for either the MAR or DAR and word operands are to be transferred to or from an 8-bit memory or device, two byte accesses will be made to the same address to read or write the word operand.

Table 5.2 Address Register Sequencing Rules

	8Bit-Memory		16Bit-Memory	
	8Bit Device	16Bit Device	8Bit Device	16Bit Device
Byte Operand	+ 1 + 1	– –	+ 1* + 2*	+ 1* + 1*
Word Operand	+ 2 + 2	+ 2 + 2	+ 2 + 4	+ 2 + 2

Memory Device	Address Register Increment

* 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.

If an exceptional condition occurs during a bus cycle that preempts normal termination, the address register associated with that bus cycle will not be incremented. However, if the second, third, or fouth bus cycle of a dual address operand transfer is terminated abnormally, the address register associated with the source (memory or device) may have been incremented if the entire operand was read successfully. Furthermore, any data that was 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 operatin 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 must return the source address register to the value before the start of the aborted operand transfer by decrementing it by one operand size. The MTCR will not be changed until all bus cycles 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 DDMA as a service request. If a memory-to-memory transfer, or a peripheral with no request signal is used, the DDMA can internally generate operand transfer requests. If an internal request generation method and dual address transfer protocol is used, the DDMA will not assert $\overline{\text{ACK}}$ 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 and if the internal limited rate method is used, the GCR is used to determine the timing constants for bus utilization calculations.

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 DDMA 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 DDMA will use 100% of the available bus bandwidth.

5.2.3.3.2 Internal, Limited Rate

In order to guarantee that the DDMA 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 DDMA. 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 DDMA to use a portion of the bus bandwidth.

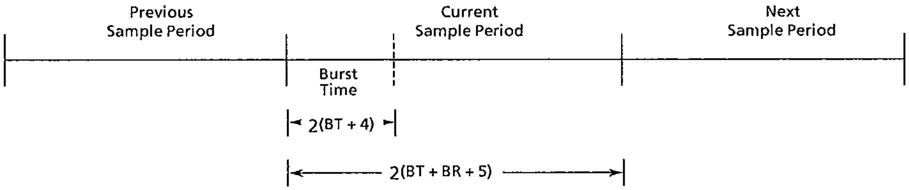
One constant is a sample period measured in clock cycles. During each sample period, the DDMA monitors the $\overline{\text{BGACK}}$ signal as an input and counts the number of clocks during which it is asserted. After a sample period has ended, the number of clocks during which $\overline{\text{BGACK}}$ was asserted (by any bus master, including the DDMA) is compared to a second programmable constant called the burst time. If the previous sample count is less than the burst time, then the DDMA will generate internal requests for any channel programmed for LRAR during the burst time for the current sample period. In this manner, if the DDMA or another 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 DDMA is equal to $2^{-(BR+1)}$, the burst time is equal to $2^{(BT+4)}$ and the sample period is equal to $2^{(BT+BR+5)}$. Figure 5.2 depicts graphically how the three constants are used to determine bus utilization by the DDMA. Table 5.3 lists all combinations of BT and BR and the resulting MPU, burst and sample period lengths for each.

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.



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

Figure 5.2 Limited Rate Auto Request Operation Timing Parameter Relationships

5.2.3.3.3 External, Burst Mode

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

If a request is recognized, but $\overline{\text{REQ}}$ is then negated before $\overline{\text{ACK}}$ is asserted, this is a violation of the burst-mode protocol and the DDMA may or may not start an operand transfer, depending on when the negated $\overline{\text{REQ}}$ is recognized internally. If the first bus cycle of an operand transfer has started when $\overline{\text{REQ}}$ is negated internally, that operand transfer will be completed. However, if it is negated internally before S0 of the first bus cycle in the transfer, the DDMA will drive the address bus for one clock cycle, but will not assert the address or data strobes. The DDMA will then check the other channel for a pending request and service it if necessary or release bus ownership if there is none. If $\overline{\text{REQ}}$ is negated before the DDMA has been granted ownership of the bus, $\overline{\text{BR}}$ will be negated without asserting $\overline{\text{BGACK}}$.

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 DDMA clock to be recognized as valid; thus, if the peripheral generates it asynchronously, it must be at least two clock periods long. The DDMA 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 DDMA asserts $\overline{\text{ACK}}$ for the previous request and before $\overline{\text{DTC}}$ is asserted, then the new request will be serviced before bus ownership is released. If a new request has not been generated for either channel by the time $\overline{\text{DTC}}$ is asserted, the bus will be released to the next master. (Refer to "4.5.2.2 Inter-cycle Overhead".)

5.2.3.4 Channel Priority

Whenever both channels are active at the same time, there exists the possibility of simultaneous pending requests. The DDMA uses a programmable channel priority scheme to resolve such occurrences. Each channel can be programmed, via the CPR, to operate at either priority 0 or 1, with 0 being the highest. When simultaneous requests occur, regardless of the request generation methods or transfer protocols used, the channel with the highest priority will be serviced first. If both channels are programmed for the same priority level, then an alternating priority is used where the last channel serviced will be the last to be serviced again (this is a round robin with only two participants) .

The effects of this priority scheme are very important in time critical applications. If one channel is programmed for burst request generation at priority 1 and the other is programmed for cycle steal request generation at priority 1 or 0, then request on the cycle steal channel will be allowed to interrupt a burst transfer on the other channel. Also, if the lower priority channel is using a dual address transfer protocol and a request for the higher priority channel is recognized as early as S0 of the first read cycle of the low priority transfer, that entire operand will be transferred (which may require four or more bus cycles) before the higher priority request can be serviced.

The channel priority is also used during interrupt acknowledge cycles to resolve simultaneous interrupt requests. When an interrupt acknowledge cycle is executed by the system processor, the highest priority channel with an inetrrupt pending will return the vector number and the lower priority channel will leave an interrupt request pending. If both channels are programmed for the same priority, the last channel to return a vector number will be the lowest priority, so that the other channel will return the vector number for the current acknowledge cycle. If neither channel has returned a vector number since the channels were started, the first channel to recognize an interrupt condition will be the highest priority.

5.2.3.5 Base Register Operations

Two special mode of operation are available on the DDMA that utilize the BFCR, BAR, and BTCR as holding registers whose values can be transferred to the MFCR, MAR, and MTCR under certain conditions. It should be noted that the continue and reload modes of operation are not mutually exclusive, but both modes use the BFCR, BAR, and BTCR for slightly different purposes. Thus, a clever system designer may use both modes simultaneously if care is taken to assure proper behavior under all circumstances.

5.2.3.5.1 Continue Operation

This mode of operation is used to allow multiple non-contiguous blocks of data to be transferred to a device or a single contiguous memory area in a single channel operation. In order to use this facility, the system processor programs the MFCR, MAR, and MTCR with the values for the first data block and the BFCR, BAR, and BTCR with the values for the second data block and sets the CNT bit in the CCR when the channel is started. When the initial MTCR value is exhausted and a new request is recognized, the DDMA will transfer the BFCR, BAR, and BTCR values into the MFCR, MAR, and MTCR, clear the CNT bit, set the BTC bit in the CSR, and continue the channel transfer operation. If interrupts are enable, then the DDMA will request service from system processor (or the processor may poll the BTC bit) to indicate that the first block has been transferred. At that time, the processor may program the BFCR, BAR, and BTCR with the values for the next block, clear the BTC bit, and set the CNT bit. These steps are repeated until the DDMA decrements the MTCR to zero and the CNT bit is not set, at which time it will terminate channel operation. Figure 5.3 shows a flowchart depicting the use of the continue mode.

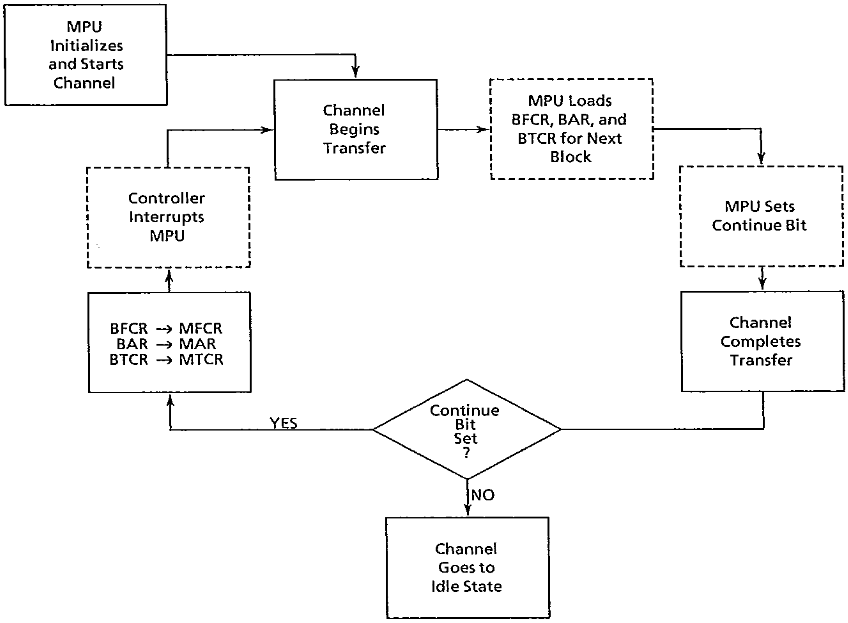

MPU Initializes and Starts Channel
Channel Begins Transfer
MPU Loads BFCR, BAR, and BTCR for Next Block
Controller Interrupts MPU
MPU Sets Continue Bit
BFCR → MFCR
BAR → MAR
BTCR → MTCR
Channel Completes Transfer
YES
Continue Bit Set ?
NO
Channel Goes to Idle State

Figure 5.3 Continue Mode Operation Flowchart

5.2.3.5.2 Reload Operation

This mode is used to allow a device to divert from sequential block transfers if necessary by asserting an input signal (PCL programmed as a reload input) . When the peripheral detects a condition requiring a change in the MFCR, MAR, and MTCR, it asserts the reload signal to the DDMA, causing the BFCR, BAR, and BTCR to be transferred to the MFCR, MAR, and MTCR before the next operand transfer starts. Figure 5.4 shows a flowchart depicting the use of the reload operation.

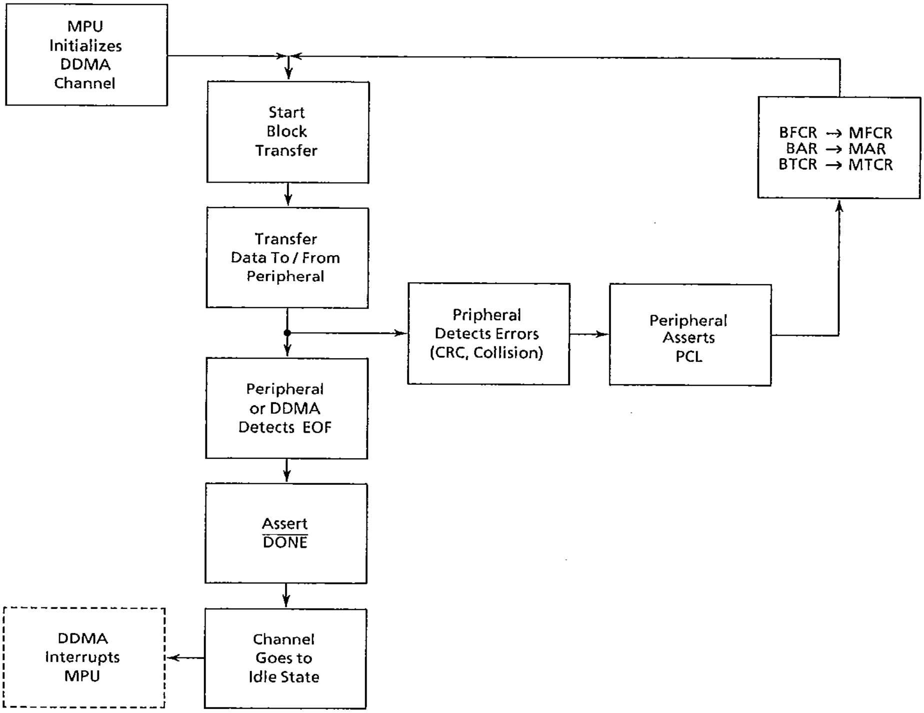

MPU Initializes DDMA Channel
Start Block Transfer
Transfer Data To / From Peripheral
Pripheral Detects Errors (CRC, Collision)
Peripheral Asserts PCL
BFCR → MFCR
BAR → MAR
BTCR → MTCR
Peripheral or DDMA Detects EOF
Assert DONE
Channel Goes to Idle State
DDMA Interrupts MPU

Figure 5.4 Reload Operation Flowchart

The values programmed into the BFCR, BAR, and BTCR are application dependent, but will often be equal to the initial values of the MFCR, MAR, and MTCR. If this is the case, when the device asserts reload during the transfer of a block, the channel will return to the initial state of the operation and begin to retransfer the same block. This is quite useful to allow communication or mass storage devices to retry a transmission, reception, read, or write operation without intervention from the system processor.

5.2.3.5.3 BTCR Value Restrictions

In the continue or reload modes of operation, the BTCR should not be loaded with value of zero. If a continue or reload operation occurs and the MTCR is loaded with zero by the operation, a count error will occur. If the following combination of parameters is programmed into a channel, the BTCR must not be loaded with a value of $0001: 16-bit memory, 8- or 16-bit device, 8-bit operand, maximum or limited-rate auto-request. If a continue or reload operation occurs and the MTCR is loaded with $0001 under these cricumstances, the DDMA will perform an incorrect operand transfer and may lock-up the bus by leaving $\overline{\text{BGACK}}$ asserted. Also, note that the channel operation will not be terminated and an error will not be flagged in the CSR.

5.2.3.6 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 DDMA bus) or the DDMA 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 DDMA: 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 enable 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 IRQ signal will be asserted. As long as a valid interrupt condition is present for either channel, the 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 TLCS-68000 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 DDMA efficiently supports the TLCS-68000 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 DDMA to return one of its vector to the processor, the $\overline{\text{IACK}}$ input is asserted during the interrupt acknowledge cycle. If the DDMA 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 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 pervious 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 $\overline{\text{REQ}}$ 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 earliest that a request will be recognized is on the first rising edge of the clock following the assertion of $\overline{\text{DTACK}}$ during an MPU write cycle that sets the STR bit. If internal maximum rate requests are used or an external request is recognized as early as possible, $\overline{\text{BR}}$ will be asserted two and one-half clock cycles after $\overline{\text{DTACK}}$ is asserted. The PCL input, however, is not ignored before the channel started. If the PCL line is programmed as a status or reload 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. If the PCT bit is still set when the channel is made active, a reload operation may occur.

5.3.1 Programming Sequence

In order to begin a data transfer operation, the following general sequence for programming the DDMA 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 operetion 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 ELVR should be loaded with the appropriate vector numbers and the corresponding locations in the exception 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 reload mode of operation is to be used, the BFCR, BAR, and BTCR should also be loaded so that the occurrence of either event can be handled when the channel is started.

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 being the transfer operation. This operation may be performed before the DDMA channel is started if internal request generation is used, depending on the requirements of the device.

5.4 DATA TRANSFERS

When the DDMA 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 sections describe the bus cycle sequence for each of the transfer protocols and effects of simultaneous channel operations, bus exceptions, and special operations on that sequence.

5.4.1 Bus Cycle Sequence

The order in which the DDMA executes bus cycles during a transfer operation depends primarily on the transfer protocol used and memory, 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 DDMA 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 tarnsfer 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, each successive bus cycle will access the address one operand size beyond 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 four 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

	8-Bit Memory		16-Bit Memory	
	8-Bit Device	16-Bit Device	8-Bit Device	16-Bit Device
Byte Operand	2	×	2	2*
Word Operand	4	3	3	2

* If internal request generation is used, tow operands will be transferred per cycle.

For the combinations that require two or four bus cycles per operand, and equal number of cycles will access memory and the device; for the three bus cycle cases, two cycles will access which ever port is smaller and one cycle will access the larger port. The bus cycle sequence will always follow the patten of: read the entire operand using as many bus cycles as are required and then write the entire operand, again using as many bus cycles as are required. Also, there will never be any 'dead time' between the bus cycles of a dual address operand transfer, so that S7 of the first read cycle will be followed immediately by S0 of the next read or write cycle, etc.

The address register and transfer count register sequencing during a dual address transfer will follow the rules discussed previously for increment amounts except for the case discussed in the next paragraph. The point a which an address register is incremented depends on whether it is used as the source or destination pointer. If used for the source pointer, it will be incremented after each operand part is successfully transferred to the internal holding register and if used for the destination pointer, it will be incremented after each operand part is successfully transferred from the internal holding register. Thus, the address registers will be incremented by one after each bus cycle required to transfer word operands to or from an 8-bit memory or device. 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 a 16-bit memory and a 16-bit device using internal request generation.

T-52-33-19

In this case, the DDMA will perform the transfer in the case manner as it would if the operand size were word; i.e.,the MAR and DAR must be even and word reads and write will be used to transfer two bytes per internal request. The MAR and DAR will be incremented by two (if programmed for increment) and the MTCR will be decremented by two after each transfer operation, except when the initial transfer count is odd, in which case the last transfer will be for a single byte.

Once the DDMA 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 DDMA will halt operation and release ownership of the bus immediately following the termination of that bus cycle. When the $\overline{\text{BEC}}$ encoding is returned to normal and bus ownership is returned to the DDMA, any operand transfer previously suspended must then be completed, regardless of pending requests for the other channel. 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 channel 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 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 DDMA 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 DDMA facilitates this count synchronization with the $\overline{\text{DTC}}$ signal in two ways. First, $\overline{\text{DTC}}$ 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 updata its transfer pointer (for an internal FIFO for example) . Second, if the device counts the number of assertions of $\overline{\text{DTC}}$ and knows how many device bus cycles are required to transfer each operand, it can properly decrement its transfer counter in step with the DDMA MTCR. A device should never use $\overline{\text{ACK}}$ by itself to clock pointer and counter registers if it is desired to support exceptional bus conditions. This is due to the fact that $\overline{\text{ACK}}$ will be asserted at the beginning of a bus cycle, but if a bus cycle is terminated with a bus error, retry, or relinquish and retry, $\overline{\text{DTC}}$ will not be asserted.

5.4.2 Effects of Channel Priority

When both channels of the DDMA are active, there exists the possibility that operand transfers for both channels 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 channel 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 the lower priority channel is being serviced and a bus cycle is terminated with a relinquish and retry, when the DDMA 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 the two channels are at different priority levels, the lower prioroty channel will not be serviced as long as the higher priority channel has a request being serviced or pending.

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

5.4.3 Exceptional Bus Condition

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 TLCS-68000 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 DDMA uses a three bit, binary encoded input bus ($\overline{\text{BEC0}}$~$\overline{\text{BEC2}}$) 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 DDMA 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 condirtions.

5.4.3.1 Reset

In the event of a catastrophic system failure, including recovery from a power-down state, the DDMA may be returned to an uninitialized idle state by asserting the reset encoding on the $\overline{\text{BEC}}$ pins. Regardless of any current operation being performed, the DDMA 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 detedcted, it will be terminated in an orderly fasion, 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 DDMA. In response, any DMA bus cycle that is 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 DDMA internal registers to determine channel status or alter operations. When halt is negated, if the DDMA 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 assered during a bus cycle, it will be terminated 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 DDMA 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.

Any DMA bus cycle may be terminated with a bus error and the response by the DDMA will be the same as described above. However, if the bus cycle aborted was during a dual address transfer, the state of the channel registers termination may be affected, according to which bus cycle of the transfer was aborted. If all read cycles of a dual address transfer were completed successfully and a write cycle to store the operand is aborted with a bus error, then the source address register (MAR or DAR) will have been incremented as needed but the destination address register (DAR or MAR) and the transfer count (MTCR) will not be modified.

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-executed at a later time. The DDMA 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 tarminated with the retry will be executed again, using the same address, data, and control information. If the other channel has a higher priority reqest pending, it will not be serviced before the retry has been executed and the entire oprand for the current channel has been transferred.

5.4.3.4.1 Retry

When the retry encoding is asserted during a bus cycle, the DDMA 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 DDMA 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 DDMA internal registers to check channel status or modify the operation. When relinquish and retry is negated, the DDMA 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 request generation method is used to inform the DDMA 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 DDMA, the protocol described previously must be followed; i.e., the peripheral must wait until $\overline{\text{ACK}}$ has been asserted once for each successive transfer request before issuing a new request. Futhermore, the assertion of $\overline{\text{DTC}}$ 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, $\overline{\text{ACK}}$ indicates that the DDMA is starting to service a request, while $\overline{\text{DTC}}$ 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 DDMA has begun to service the pending request.

When single address transfers are used, this protocol is very easy to follow since $\overline{\text{ACK}}$ will only be asserted once for each operand transfer. However, during address transfers, $\overline{\text{ACK}}$ may be asserted twice for each operand transfer and thus the DDMA will only recognize requests at certain points during such a transfer. The following paragraphs describe when new external requests are recognized during various types of dual address transfers.

5.4.4.1 Device-to-Memory Transfer

For this type of a transfer, $\overline{\text{ACK}}$ will be asserted during the first bus cycle of each operand transfer and thus it is similar to the protocol for single address transfers. A new request will be recognized during the first bus cycle according to the rules for the request generation method used. For cycle steal, an asserted transition on $\overline{\text{REQ}}$ after the assertion of $\overline{\text{ACK}}$ will be recognized. For burst mode, if $\overline{\text{REQ}}$ is asserted when $\overline{\text{DTC}}$ is asserted, then a new request will be recognized. If a new request is not recognized during the first bus cycle of an operand transfer, one may still be recognized during the second, third, or fourth bus cycles. However, if a new request is recognized during the first, second, or third bus cycle, further requests will be ignored during the remaining bus cycles of the operand transfer.

When a new request is not recognized during the first bus cycle of an operand transfer and the second bus cycle of the transfer is also to the device (as will occur when transferring word operands from an 8-bit device) , new requests will be recognized in the same manner as during the first bus cycle. If a new request is still not pending after the operand has been transferred to the internal holding register, requests may be recognized during the bus cycle(s) to tarnsfer the operand to memory; however, the assertion of $\overline{\text{ACK}}$ will not be available as a timing reference point. In order for a new operand transfer request to be serviced immediately after the current operand transfer, the only requirement is that the new request is recognized by E2 (refer to "4.5.2.2 Inter-Cycle Overhead" for the definition of the E2 point) of the last bus cycle in the current transfer, and that the other channel does not have a higher priority request pending.

5.4.4.2 Memory-to-Device Transfers

The manner in which the DDMA recognizes requests during this type of transfer is identcal to the device-to-memory case discussed above, except that new requests can be recognized before $\overline{\text{ACK}}$ is asserted in response to the pervious request. This is because new requests will be recognized during the bus cycle(s) to move an operand from memory to the internal holding register but $\overline{\text{ACK}}$ is not asserted until the first bus cycle to move the operand to the device. Thus, if the device asserts a request during the operand read cycle(s) of a transfer to service a previous request and then asserts another request when $\overline{\text{ACK}}$ is asserted, only one of those requests will be recognized. The important consideration in this case is that the peripheral device must count number of assertions of $\overline{\text{ACK}}$ and $\overline{\text{DTC}}$ in response to each assumed request recognition.

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 DDMA bus in order to perform the MPU accesses to the DDMA. 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 DDMA. The following paragaraphs describe the software control functions available and the DDMA 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 $\overline{\text{REQ}}$ 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 normally not have access to the DDMA 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 HLT bit before the next bus cycle. In this case, when the $\overline{\text{BEC}}$ pins return to normal, the DDMA will rearbitrate for the bus complete tha dual address operand transfer before the software halt will be honored. It should also be noted that pending transfer requests for the other channel can not be serviced unil system processor has returned bus ownership to the DDMA 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 DDMA 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 DDMA will remain idle until the $\overline{\text{BEC}}$ 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. Also, the source address register may have been incremented, following the same rules as described for bus error terminaton of a dual address transfer bus cycle (refer to “5.4.1.2 Dual Address Transfers”) .

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 DDMA 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 ture 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.

5.5 BASE REGISTER OPERATIONS

Two special operating modes are available to reduce the system processor overhead required for certain transfers. These two modes are the continue and reload modes of operation and both 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. Both of these modes of operation are independent of the transfer protocol or request generation method used and do not affect the DFCR or DAR.

Note that for the reload mode of operation, if the DDMA is ready to start a bus cycle when the BFCR/BAR/BTCR to MFCR/MAR/MTCR transfer is occurring, the function code and address buses will be driven with the old MFCR and MAR values for one clock cycle. The function code and address line will then change to new MFCR and MAR values and a normal bus cycle will start. This is similar to the behavior described in "4.5.2.2 Inter-Cycle Overhead" when a channel switch occurs.

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 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 or assert the $\overline{\text{DONE}}$ signal, but instead remain active. 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 DDMA has completed the previous block transfer and is starting on the next. 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.

5.5.2 Reload Operation

To utilize the reload mode of operation, the PCL input is programmed as a reload (RLD) signal and the BFCR, BAR, and BTCR are loaded with the start address and size of the block to be trandsferred when the RLD signal is asserted by the peripheral. The most common use for this mode of operation is to allow the peripheral to signal the DDMA to restart the transfer of a data block, in which case the BFCR, BAR, and BTCR are programmed with the initial values of the MFCR, MAR, and MTCR before the channel is stared.

When the RLD signal is asserted by the peripheral, no action is taken immediately, but the PCT bit in the CSR will be set and acted on later. Once the channel is active, the PCT bit is checked each time a transfer request is recognized. If PCT is clear, then the request is serviced in the normal manner. However, if the PCT bit is set, the contents of the BFCR, BAR, and BTCR will be transferred to the MFCR, MAR, and MTCR, the PCT bit will be cleared, and the RLD bit in the CSR will be set. The request will then be serviced in the normal fashion and the channel will continue the transfer operation. Since the RLD bit will not generate an interrupt when it is set, the system processor is not disturbed by the occurrence of the reload operation, but it may poll the CSR to determine thet it took place.

5.6 CHANNEL TERMINATION

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

5.6.1 Transfer Count Exhausted

When the DDMA begins an operand transfer, if the current value of the MTCR is one and an external request generation method is being used, then the $\overline{\text{DONE}}$ 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 tarnsfer 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 $\overline{\text{DONE}}$ signal is asserted when $\overline{\text{DTC}}$ is asserted during a device access, then the channel operation will be terminated. Also, $\overline{\text{DONE}}$ must be sampled as aserted on two consecutive rising clock edges; so if it is generated asynchronously, it must be asserted for two DDMA 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 operand from or an 8-bit memory or device, the channel operation will be terminated with only one byte being transferred. This means that one more bus cycle may be executed after $\overline{\text{DONE}}$ is recognized to write the byte to the memory or a device, but a full size operand transfer will not be executed. In this case, the MAR and/or DAR will be incremented by one, and the MTCR will be decremented by one, even though an entire operand was not transferred.

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 DDMA 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 DDMA 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 5.5 summarizes all of the error conditions that can cause channel termination and the source of each error.

5.6.3.1 Internal Errors

There are five types of internal errors that can occur: configuration errors, operation timing errors, address error, 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.

Table 5.5 Channel Error Condition Summary

Error Type	Cause
Configuration Error	Any undefined reserved bit pattern, TMP68450 reserved option, or illegal device/memory/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 ot if ACT is not set. An attempt si 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	$\overline{CS}$ or $\overline{IACK}$ is asserted when the DDMA is acting as bus master. A word bus cycle is attempted to an odd address. This may occur with the following configurations: Single address transfer, word operand size, and the MAR is odd. Dual address transfer, word operand size (or byte operand size with internal request generation) , and the MAR or DAR is odd.
Bus Error	A DDMA 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 or $0000 is loaded into the MTCR by a continue or reload operation.
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.

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 DDMA is acting as the bus master and $\overline{CS}$ or $\overline{IACK}$ is asserted, or a 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 DDMA executes an operand transfer and no DDMA internal registers will be affected (other than the CSR) . Note that since an address error is detected before an operation start, it is possible for the DDMA to flag an address error on both the MAR and the DAR with a special CER encoding.

5.6.3.1.4 Count Error

When a channel is started or a continue or reload 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, the count error is detected after the MFCR, MAR, and MTCR are loaded from the BFCR, BAR, and BTCR, and the channel operation is terminated whether or not a transfer request is pending. If an operand transfer request is recognized, the PCT bit is set (with the PCL line programmed as a reload input) , and the BTCR is zero, then a count error is detected after the MFCR, MAR, and MTCR are loaded from the BFCR, BAR, and BTCR.

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 DDMA when either of these conditions is signaled has been previously discussed (refer to “5.4.3.3 Bus Error” and “5.4.6 Hard-Ware Abort”) .

6. ELECTRICAL SPECIFICATIONS

This section contains electrical specifications and associated timing information for the TMP68440 dual-channel direct memory access controller.

6.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	Vcc	−0.3∼+7.0	V
Input Voltage	Vin	−0.3∼+7.0	V
Operating Temperature Range	Ta	0∼70	℃
Storage Temperature	Tstg	−55∼+150	℃

This device contains circuitry to protect the input 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-impendance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or Vcc) .

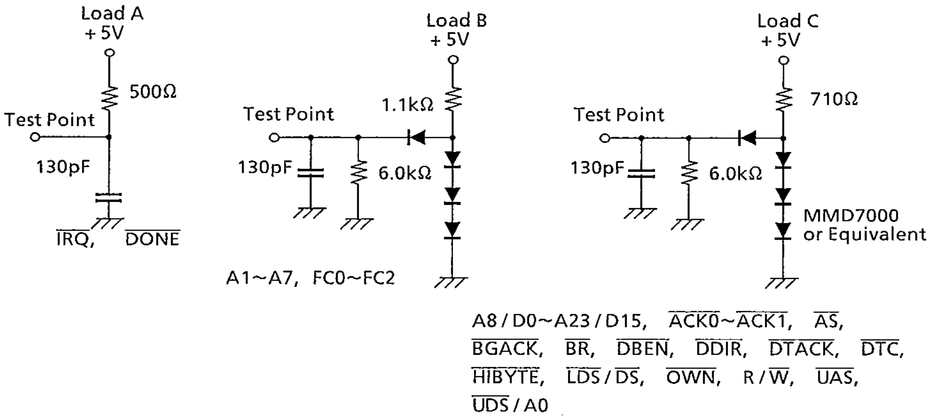


Figure 6.1 Test Loads

6.2 DC ELECTRICAL CHARACTERISTIC Vcc=4.75V to 5.25V, GND=0V, Ta=0℃ to 70℃ unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
Input High Voltage All Inputs	VIH	2.0	Vcc	V
Input Low Voltage All Inputs	VIL	Vss −0.3	0.8	V
Input Leakage Current @ 5.25 V All Inputs	Iin	—	10	μA
Input Capacitance (V_{in} = 0V, Ta = 25°C, Frequency = 1 MHz) All Inputs	Cin	—	13	pF
Three-State (Off-State) Input Current@2.4V/0.4V $\overline{AS}$, A1~A7, $\overline{BGACK}$, $\overline{DTACK}$, A8 / D0~A23 / D15, $\overline{HIBYTE}$, $\overline{LDS}$ / $\overline{DS}$, $\overline{UDS}$ / A0, R / $\overline{W}$	ITSI	—	20	μA
Open-Drain (Off-State) Input Current @ 5.25 V $\overline{IRQ}$, $\overline{DONE}$	IDD	—	20	μA
Output High Voltage (IOH = −400 μA Min) $\overline{AS}$, A1~A7, A8 / D0~A23 / D15, $\overline{ACK0}$, $\overline{ACK1}$, $\overline{BR}$, $\overline{BGACK}$, $\overline{DBEN}$, $\overline{DDIR}$, $\overline{DTACK}$, $\overline{OWN}$, $\overline{LDS}$ / $\overline{DS}$, $\overline{UDS}$ / A0, R/$\overline{W}$, $\overline{UAS}$, $\overline{DTC}$, FC0~FC2, $\overline{IRQ}$, $\overline{DONE}$, $\overline{HIBYTE}$	VOH	2.4	—	V
Output Low Voltage (IOL = 3.2mA Min) A1~A7, FC0~FC2 (IOL = 5.3mA Min) A8 / D0~A12 / D15, $\overline{ACK0}$, $\overline{ACK1}$, $\overline{AS}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DBEN}$, $\overline{DDIR}$, $\overline{DTACK}$, $\overline{DTC}$, $\overline{HIBYTE}$, $\overline{LDS}$ / $\overline{DS}$, $\overline{UDS}$ / A0, $\overline{OWN}$, R/$\overline{W}$, $\overline{UAS}$ (IOL = 8.9mA Min) $\overline{IRQ}$, $\overline{DONE}$	VOL	—	0.4	V
Power Dissipation at 0°c (Frequency = 8 MHz)	PD	—	1.5	W

6.3 AC ELECTRICAL SPECIFICATIONS - CLOCK TIMING (See Figure 6.2)

Characteristic	Symbol	8MHz		10MHz		Unit
		Min	Max	Min	Max	
Frequency of Operation	F	2.0	8.0	2.0	10.0	MHz
Cycle Period	tCYC	125	500	100	500	ns
Clock Width Low	tCL	55	250	45	250	ns
Clock Width High	tCH	55	250	45	250	ns
Clock Fall Time	tCf	—	10	—	10	ns
Clock Rise Time	tCr	—	10	—	10	ns

6.4 AC ELECTRICAL SPECIFICATIONS - READ AND WRITE CYCLES (Vcc=4.75V to 5.25V, GND=0V, Ta=0℃ to 70℃ unless otherwise noted) (See Figures 6.3 through 6.9)

(1/6)

Num.	Characteristic	Symbol	8MHz		10MHz		Unit
			Min	Max	Min	Max	
1	Clock Period	tCYC	125	500	100	500	ns
2	Clock Width Low	tCL	55	250	45	250	ns
3	Clock Width High	tCH	55	250	45	250	ns
4	Clock Fall Time	tCf	–	10	–	10	ns
5	Clock Rise Time	tCr	–	10	–	10	ns
6	MPU Address Valid to $\overline{CS}$ Low	tADVCSL	0	–	0	–	ns
7	MPU $\overline{AS}$ High to ddress Invalid	tASHADI	0	–	0	–	ns
8	Asynchronous Input Setup Time	tASI	20	–	20	–	ns
9	Data Strobe(s) Low to $\overline{CS}$ Low	tDSLCSL	0	–	0	–	ns
10[1]	$\overline{CS}$ Low to $\overline{DDIR}$ High (MPU Read)	tCSLDDHR	2Clks + 20	3Clks + 80	2Clks + 20	3Clks + 70	ns
11[1]	$\overline{CS}$ Low to $\overline{DBEN}$ Low (MPU Read)	tCSLENLR	2.5clks + 20	3.5clks + 80	2.5clks + 20	3.5clks + 70	ns
12[1]	$\overline{CS}$ Low to Data Out Valid (MPU Read)	tCSLDOV	2Clks + 20	3Clks + 100	2Clks + 20	3Clks + 85	ns
13[1]	$\overline{CS}$ Low to $\overline{DTACK}$ Low (MPU Read)	tCSLDTLR	3.5clks + 20	4.5clks + 80	3.5clks + 20	4.5clks + 70	ns
14	Clock High to Data Out Valid	tCHDOV	–	80	–	65	ns
15	$\overline{CS}$ High to $\overline{DDIR}$ High Impedance	tCSHDDZ	–	60	–	50	ns
16	$\overline{CS}$ High to $\overline{DBEN}$ High Impedance	tCSHENZ	–	60	–	50	ns
17	$\overline{CS}$ High to Data High Impedance	tCSHDZ	–	60	–	50	ns
18	Clock Low to $\overline{DTACK}$ Low	tCLDTL	–	60	–	50	ns
19	$\overline{DTACK}$ Low to $\overline{CS}$ High	tDTLCSH	0	–	0	–	ns
20	$\overline{CS}$ High to $\overline{DTACK}$ High	tCSHDTH	–	50	–	45	ns
21	$\overline{CS}$ High to $\overline{DTACK}$ High Impedance	tCSHDTZ	–	60	–	50	ns

(2/6)

Num.	Characteristic	Symbol	8MHz		10MHz		Unit
			Min	Max	Min	Max	
22	$\overline{CS}$ Width High	tCSWH	1	—	1	—	Clk. Per.
23	R / $\overline{W}$ Low to $\overline{CS}$ Low	tRWLCSL	0	—	0	—	ns
24[1]	$\overline{CS}$ Low to $\overline{DDIR}$ Low (MPU Write)	tCSLDDLW	1Clk + 20	2Clks + 80	1Clk + 20	2Clks + 70	ns
25[1]	$\overline{CS}$ Low to $\overline{DBEN}$ Low (MPU Write)	tCSLENLW	1.5Clks + 20	2.5Clks + 80	1.5Clks + 20	2.5Clks + 70	ns
26	$\overline{CS}$ Low to Data In Valid (MPU Write)	tCSLDIV	—	3	—	3	Clk. Per.
27[1]	$\overline{CS}$ Low to $\overline{DTACK}$ Low (MPU Write)	tCSLDTLW	2.5Clks + 20	3.5Clks + 80	2.5Clks + 20	3.5Clks + 70	ns
28[8]	$\overline{DDIR}$ Low to $\overline{DBEN}$ Low	tDDLENL	30	—	20	—	ns
29	$\overline{DBEN}$ Low to Data In Valid	tENLDIV	—	105	—	80	ns
30	Data In Valid to Clock High (Setup Time)	tDIVCH	15	—	15	—	ns
31[8]	$\overline{DTACK}$ LOW to $\overline{DDIR}$ High	tDTLDDH	125	—	100	—	ns
32[8]	$\overline{DTACK}$ LOW to $\overline{DBEN}$ High	tDTLENH	65	—	50	—	ns
33[8]	$\overline{DBEN}$ High to $\overline{DDIR}$ High (Read)	tENHDDH	30	—	20	—	ns
34	$\overline{CS}$ High to Data Not Valid	tCSHDNV	0	—	0	—	ns
35	$\overline{REQ}$ Width Low	tREQL	2	—	2	—	Clk. per.
36	$\overline{REQ}$ Low to $\overline{BR}$ Low	tREQLBRL	2Clks + 20	3Clks + 80	2Clks + 20	3Clks + 70	ns
37[2]	$\overline{REQ}$ Low to $\overline{BGACK}$ Low	tREQLBKL	4	—	4	—	Clk. per.
38	Clock High to $\overline{BR}$ Low	tCHBRL	—	60	—	50	ns
39[3]	$\overline{BR}$ Low to $\overline{BG}$ Low	tBRLBGL	− 1	—	− 1	—	Clk. per.
40[3]	$\overline{BR}$ Low to $\overline{AS}$ In High	tBRLASH	− 1	—	− 1	—	Clk. per.
41	Clock High to $\overline{BR}$ High Impedance	tCHBRZ	—	60	—	50	ns

(3/6)

Num.	Characteristic	Symbol	8MHz		10MHz		Unit
			Min	Max	Min	Max	
42	Clock Low to $\overline{OWN}$ Low	tCLOL	—	60	—	50	ns
43	Clock High to $\overline{BGACK}$ Low	tCHBKL	—	60	—	50	ns
44[8]	$\overline{BGACK}$ Low to $\overline{BR}$ High Impedance	tBKLBRZ	60	1Clk +60	50	1Clk +50	ns
45	$\overline{BGACK}$ Low to $\overline{BG}$ High	tBKLBGH	0	—	0	—	ns
46[2]	$\overline{AS}$, $\overline{CS}$ In High to $\overline{BGACK}$ Low	tASHBKL	2	—	2	—	Clk. per.
47[2]	Clock Low on which $\overline{OWN}$ Asserted to Clock High on which $\overline{AS}$ Asserted	tOLASL	—	1.5	—	1.5	Clk. per.
48	Clock High to $\overline{BGACK}$ High	tCHBKH	—	60	—	50	ns
49	Clock High to $\overline{BGACK}$ High Impedance	tCHBKZ	—	65	—	55	ns
50	Clock Low to $\overline{OWN}$ High	tCLOH	—	60	—	50	ns
51	Clock Low to $\overline{OWN}$ High Impedance	tCHOZ	—	65	—	50	ns
52[4, 8]	$\overline{BGACK}$ High to $\overline{OWN}$ High	tBKHOH	30	—	20	—	ns
53[8]	$\overline{DTC}$ High Impedance to $\overline{BGACK}$ High	tTCZBKH	—	1Clk +60	—	1Clk +50	ns
54	Clock High to Address / FC Valid	tCHAV	—	80	—	65	ns
55	Clock High to Control and Non-Muxed Bus Lines High Impedance	tCHNXZ	—	60	—	50	ns
56	CLK Low to Muxed Address Bus High Impedance	tCLMXAZ	—	60	—	50	ns
57	Data In Valid to Clock High (Setup Time)	tDIVCH	15	—	15	—	ns
58	Clock High to $\overline{UAS}$ Low	tCHUL	—	80	—	65	ns
59	Clock High to $\overline{UAS}$ High	tCHUH	—	60	—	50	ns

(4/6)

Num.	Characteristic	Symbol	8MHz		10MHz		Unit
			Min	Max	Min	Max	
8 60	$\overline{UAS}$ High to Address Invalid	tUHAI	20	—	20	—	ns
8 61	Address / FC Valid to $\overline{AS}$ / $\overline{DS}$ (Read), $\overline{AS}$ (Write) Low	tAVSL	60	—	50	—	ns
62	$\overline{AS}$, $\overline{DS}$ Width Low (Read)	tASLR	125	—	100	—	ns
63	Clock Low to $\overline{AS}$, $\overline{DS}$ High	tCLSH	—	60	—	50	ns
8 64	$\overline{AS}$, $\overline{DS}$ High to Address / FC / Data Invalid	tSHAI	40	—	20	—	ns
8 65	$\overline{AS}$ High to $\overline{UAS}$ Low	tASHUL	20	—	20	—	ns
5 66	Clock High to $\overline{AS}$ LOW	tCHASL	—	50	—	40	CLK. Per.
8 67	$\overline{AS}$ Low to $\overline{DBEN}$ Low	tASLENL	—	120	—	100	ns
68	Clock High to $\overline{DS}$ Low (Read)	tCHDSLR	—	60	—	50	ns
69	Clock High to $\overline{DDIR}$ Low	tCHDDL	—	60	—	50	ns
70	Clock High to $\overline{DDIR}$ High	tCHDDH	—	60	—	50	ns
71	Clock Low to $\overline{DBEN}$ Low	tCLENL	—	60	—	50	ns
72	Clock Low to $\overline{DBEN}$ High (Read)	tCLENH	—	60	—	50	ns
73	$\overline{DTACK}$ Low to Data In Valid	tDTLDIV	—	90	—	65	ns
74	$\overline{DS}$ High to $\overline{DTACK}$ High	tDSHDTH	0	120	0	100	ns
75	$\overline{BEC}$ Valid to $\overline{DTACK}$ Low	tBECVDTL	0	—	0	—	ns
76	$\overline{AS}$ High to $\overline{BEC}$ Negated	tASHBECN	—	0	—	0	ns
77	$\overline{BEC}$ Width Low	tBECL	2	—	2	—	Clk. Per.
78	Clock High to $\overline{ACK}$ Low (Read)	tCHAKLR	—	60	—	50	ns
79	Clock High to $\overline{ACK}$ High	tCHAKH	—	60	—	50	ns

(5/6)

Num.	Characteristic	Symbol	8MHz		10MHz		Unit
			Min	Max	Min	Max	
80	Clock High to $\overline{\text{DTC}}$ Low	tCHTCL	–	50	–	40	ns
81[4, 8]	$\overline{\text{DTC}}$ Low to $\overline{\text{DS}}$ High	tTCLDSH	30	–	20	–	ns
82	Clock High to $\overline{\text{DTC}}$ High	tCHTCH	–	60	–	50	ns
83	$\overline{\text{DONE}}$ Input Low to Clock on which $\overline{\text{DTC}}$ Asserted	tDNLTCL	20	–	20	–	ns
84	$\overline{\text{DTC}}$ Width Low	tDTCL	1	–	1	–	Clk. Per.
85	Clock High to $\overline{\text{DONE}}$ Low (Read)	tCHDNL	–	60	–	50	ns
86	Clock High to $\overline{\text{DONE}}$ High Impedance	tCHDNZ	–	60	–	50	ns
87	Clock High to $\overline{\text{IRQ}}$ Low	tCHIRL	–	60	–	50	ns
88	Clock High to $\overline{\text{IRQ}}$ High Impedance	tCHIRZ	–	60	–	50	ns
89	Data Out Valid to $\overline{\text{DS}}$ Low	tDOVDSL	80	–	60	–	ns
90	Clock High to Muxed Data Bus High Impedance	tCHMXDZ	–	60	–	50	ns
91[8]	$\overline{\text{UAS}}$ Low to $\overline{\text{AS}}$ Low	tULASL	60	–	50	–	ns
92	$\overline{\text{AS}}$ Width Low (Write)	tASLW	250	–	200	–	ns
93	Clock Low to $\overline{\text{DS}}$ Low (Write)	tCLDSLW	–	60	–	50	ns
94[8]	$\overline{\text{DBEN}}$ Low to $\overline{\text{DS}}$ Low (Write)	tENLDSL	60	–	50	–	ns
95	$\overline{\text{DS}}$ Width Low (Write)	tDSLW	70	–	55	–	ns
96[8]	Address / FC Valid to R / $\overline{\text{W}}$ Low	tAVRL	60	–	50	–	ns
97[8]	R/$\overline{\text{W}}$ Low to $\overline{\text{DS}}$ Low (Write)	tRLDSL	80	–	60	–	ns
98[8]	$\overline{\text{DS}}$ High to R/$\overline{\text{W}}$ High	tDSHRH	20	–	20	–	ns
99	Clock High to R / $\overline{\text{W}}$ High	tCHRH	–	60	–	50	ns

(6/6)

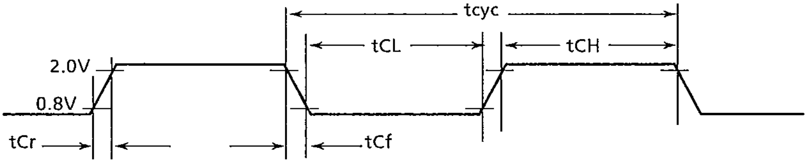
Num.	Characteristic	Symbol	8MHz		10MHz		Unit
			Min	Max	Min	Max	
100[5]	Clock High to R / $\overline{W}$ Low	tCHRL	—	60	—	50	ns
101	Clock High to $\overline{DBEN}$ High (Write)	tCHENH	—	60	—	50	ns
102	$\overline{DTACK}$ Width High	tDTWH	0	—	0	—	ns
103	Clock Low to $\overline{ACK}$ Low (Write)	tCLAKLW	—	60	—	60	ns
104	Clock Low to $\overline{DONE}$ Low (Write)	tCLDNL	—	60	—	50	ns
105	Clock High to $\overline{HIBYTE}$ Low (Read)	tCHHBLR	—	60	—	50	ns
106	Clock High to $\overline{HIBYTE}$ High	tCHHBH	—	60	—	50	ns
107	$\overline{DTACK}$ and PCL Low to $\overline{AS}$ High (Single Address Read)	tDTLASH	190	—	150	—	ns
108	Clock Low to $\overline{HIBYTE}$ Low (Write)	tCLHBLW	—	60	—	50	ns
109[8]	$\overline{ACK}$ Low to $\overline{DS}$ Low (Single Address Write)	tAKLDSL	80	—	60	—	ns
110	PCL Low to DS Low ($\overline{ACK}$ with Ready Write)	tPCLDSL	190	—	150	—	ns

Notes :

1. These specifications assume that the input setup time for $\overline{CS}$ is zero, which violates #8, but it is still recognized as asserted.
2. With both channels active these numbers increase by one clock.
3. $\overline{AS}$ and $\overline{BG}$ from the MPU are first sampled on the rising clock edge on which $\overline{BR}$ is asserted. Therefore, if $\overline{AS}$ is negated and $\overline{BG}$ is asserted for at least one asynchronous input setup time prior to that clock edge, then the minimum arbitration times will be achieved.
4. These minimum times assume that the two signals have equal resistive and capacitive loading (±20%) .
5. When $\overline{AS}$ and R/$\overline{W}$ are equally loaded (±10%) $\overline{AS}$ will be asserted no more than 20 ns before R/$\overline{W}$.
6. Minimum timing for single address write cycles occurs with $\overline{ACK}$ only or with $\overline{ACK}$ and PCL as READY when PCL is asserted for more than one synchronization delay before the clock edge on which $\overline{ACK}$ is asserted.
7. Specifications that include a number of clock periods refer to the actual input clock used and not the specified clock minimum or maximum values.
8. These specifications refer to the skew between two output signals that change following different edges of the clock; therefore, the actual value depends on the clock signal that is used. The minimum times are guaranteed for a minimum clock timing (high or low and period) .

Note : For clarity, specification numbers are shown only once in Figures 6.6 through 6.9. However, many specifivctions apply equally to all four diagrams. For example, specification numbers 54 and 55 are shown only in Figure 6.6, but apply to Figures 6.7 through 6.9 as well. As a guideline, Figure 6.6 includes all necessary specifications for a dual-address read cycle and Figure 6.8 includes additional specifications for a single-address read cycle; the same relationship is true for Figures 6.7 and 6.9. Thus, the specifications shown in Figure 6.7 through 6.9 can be considered to be additions to or substitutions for the specifications shown in Figure 6.6.

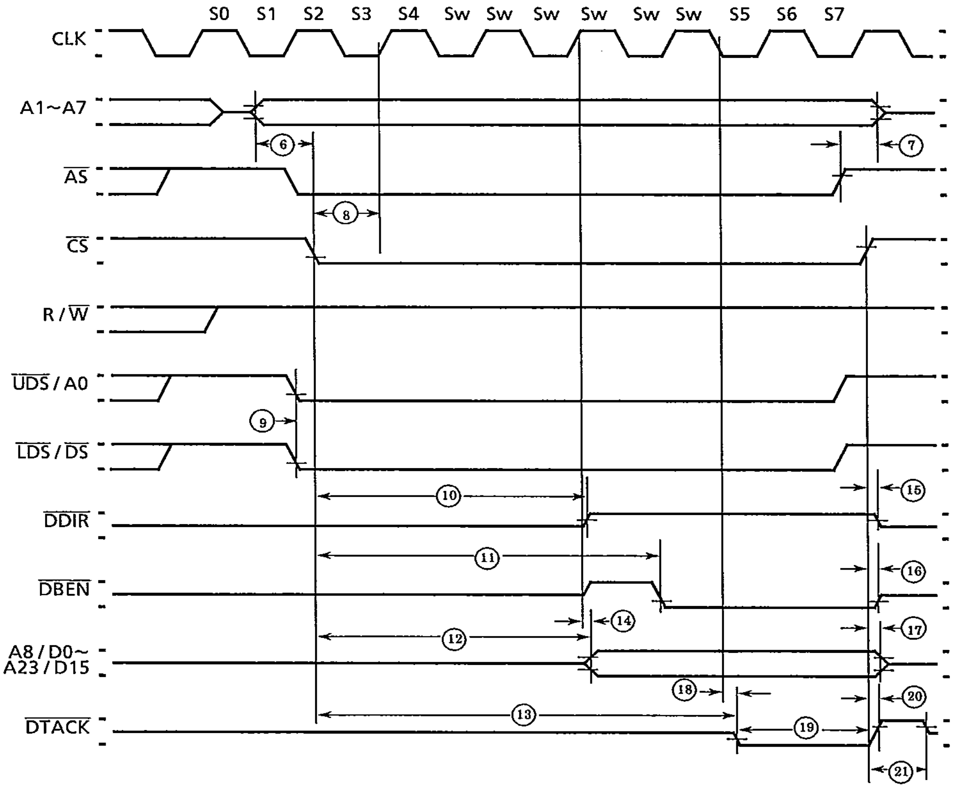
When referring to the timing specifications shown in Figure 6.3 through 6.9, it is helpful to remember that all output signals will change states only in response to a specific transition on the CLK input and that all input signals are latched and synchronized on rising edges of the CLK input.



Note : Timing measurements are referenced to and from a low voltage of 0.8 volts 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 volts and 2.0 volts.

Figure 6.2 Clock Input Timing Diagram

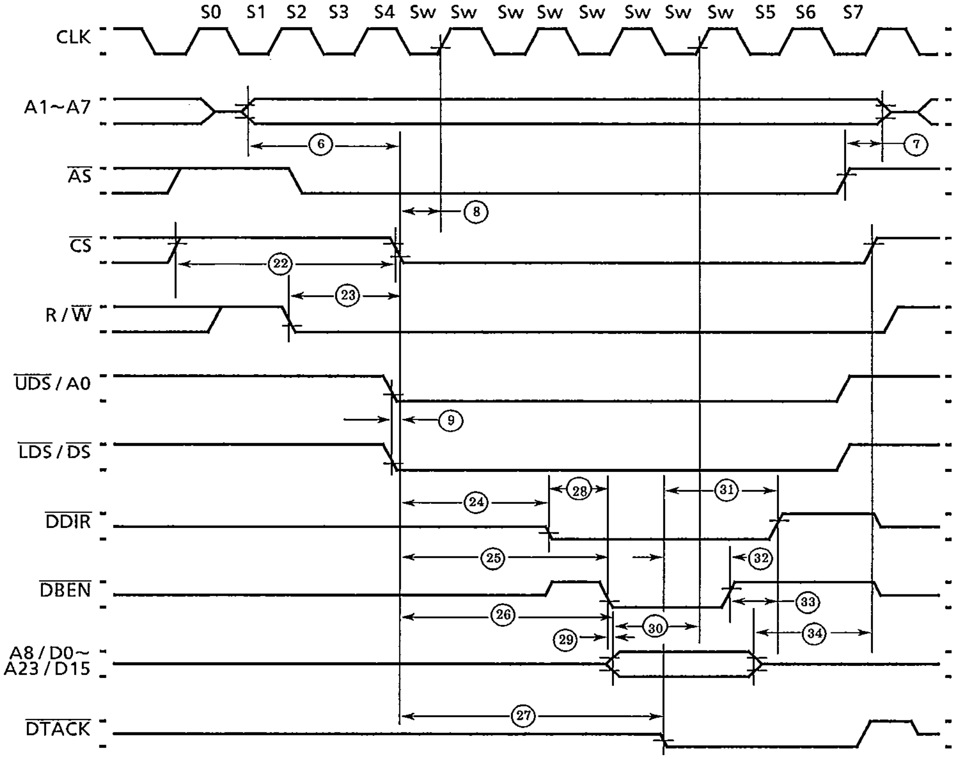
These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



Note : Timing measurements are referenced to and from a low voltage of 0.8 volts 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 volts and 2.0 volts.

Figure 6.3 MPU Read Cycle Timing Diagram

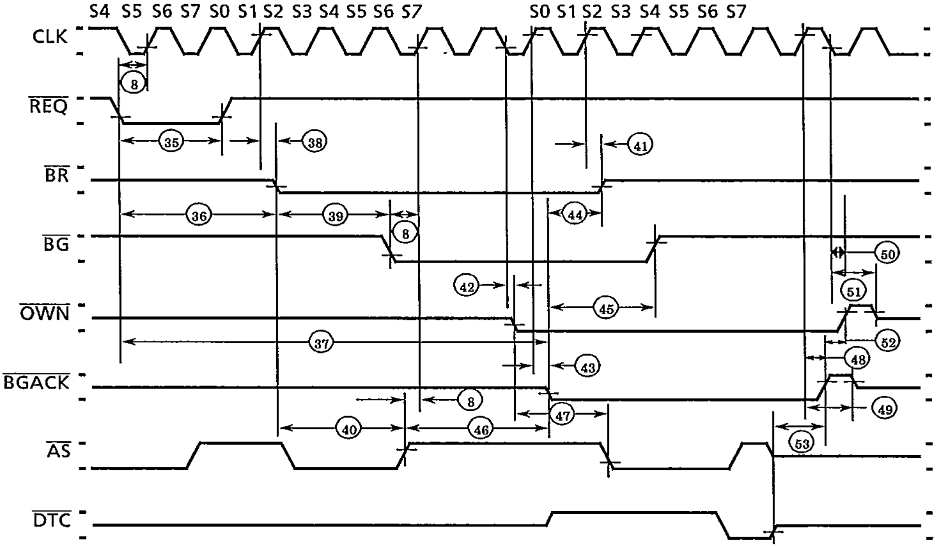
These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



Note: Timing measurements are referenced to and from a low voltage of 0.8 volts and 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 volts and 2.0 volts.

Figure 6.4 MPU Write Cycle Timing Diagram

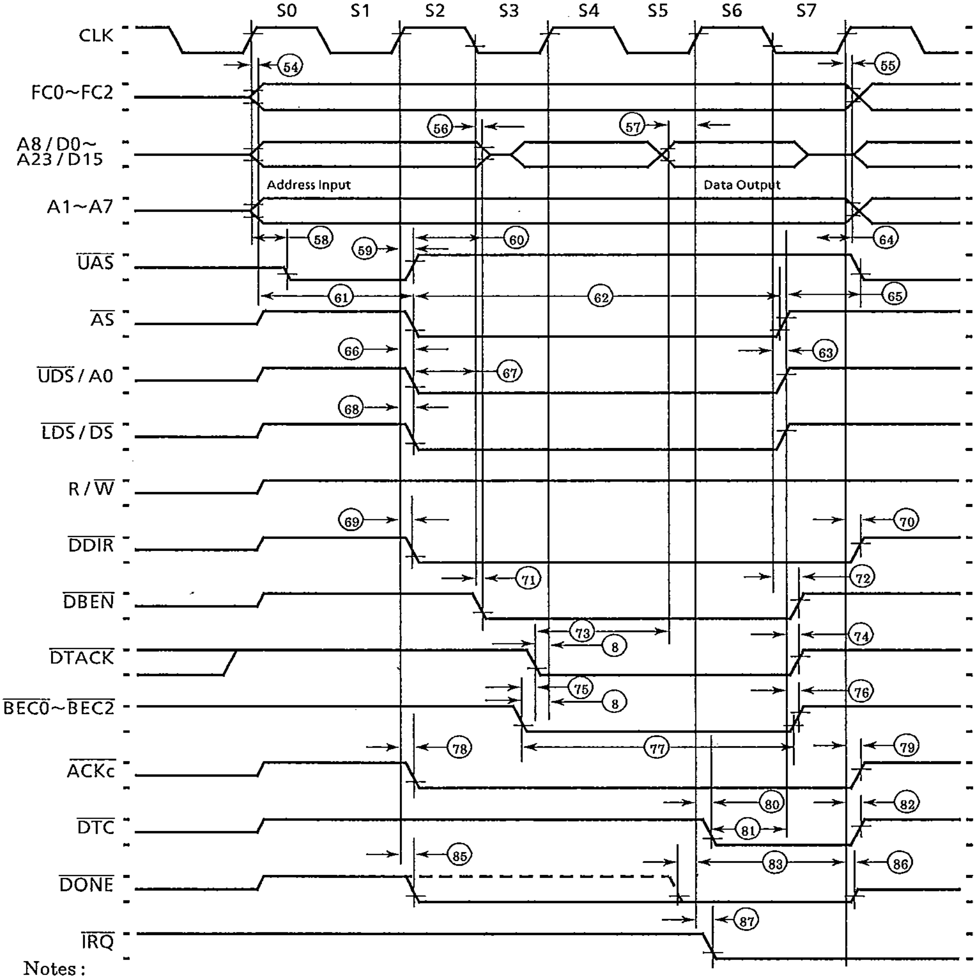
These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



Note : Timing measurements are referenced to and from a low voltage of 0.8 volts 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 volts and 2.0 volts.

Figure 6.5 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.

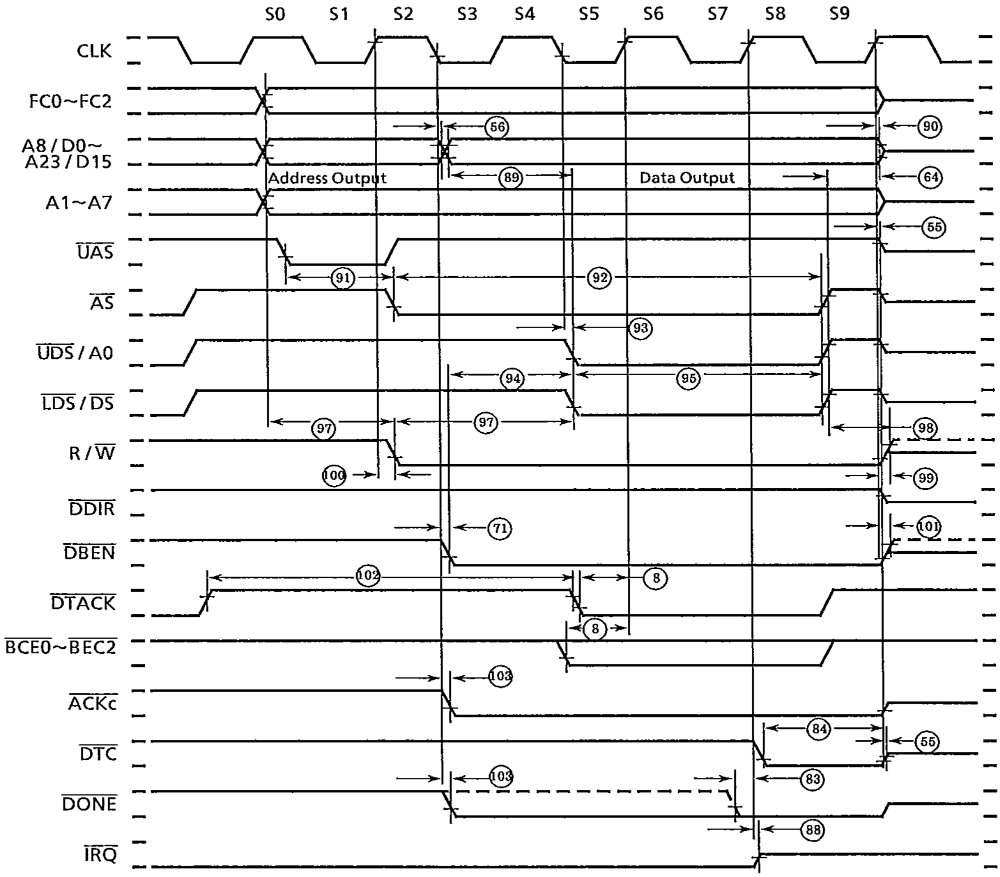


Notes :

1. The solid line illustrates $\overline{\text{DONE}}$ as an output, and the dotted line illustrates $\overline{\text{DONE}}$ as an input.
2. Timing measurements are referenced to and from a low voltage of 0.8 volts 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 volts and 2.0 volts.

Figure 6.6 Dual Address Read Cycle Timing Diargam

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.

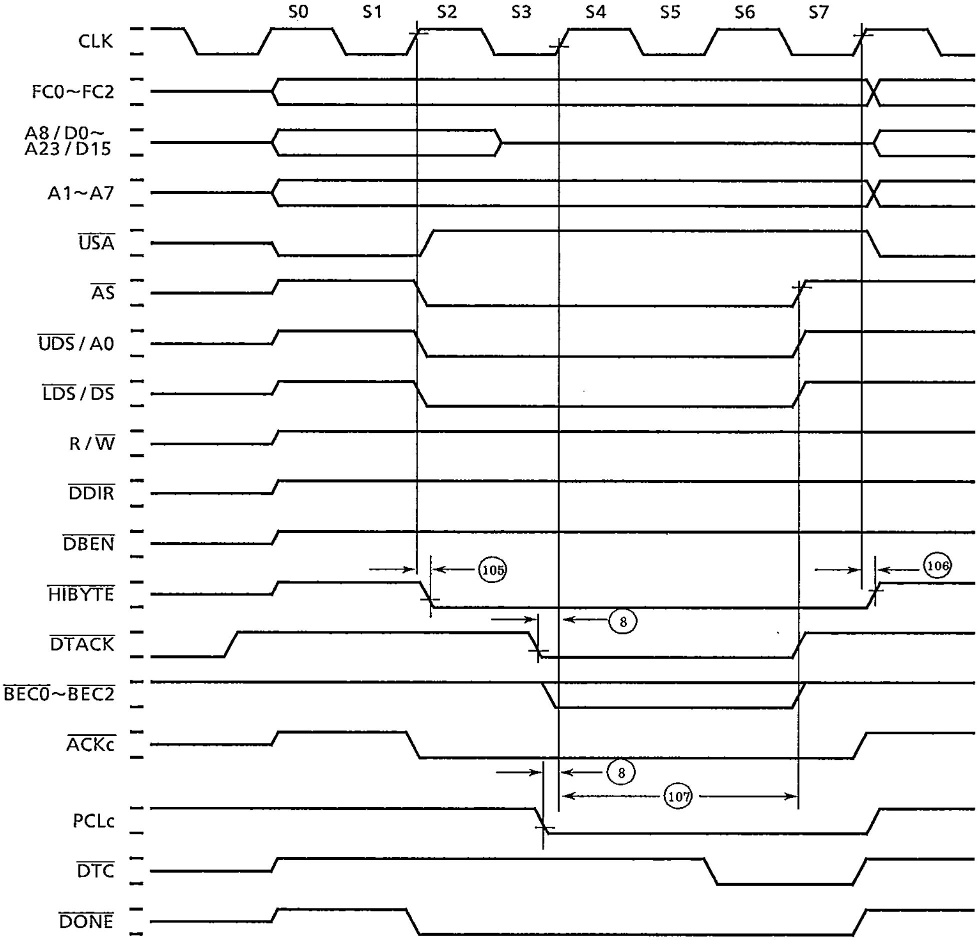


Notes:

1. Timing specifications 99 and 101 are only applicable when another DDMA bus cycle immediately follows this one.
2. Timing measurements are referenced to and from a low voltage of 0.8 volts 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 volts and 2.0 volts.

Figure 6.7 Dual Address Write Cycle Timing Diagram

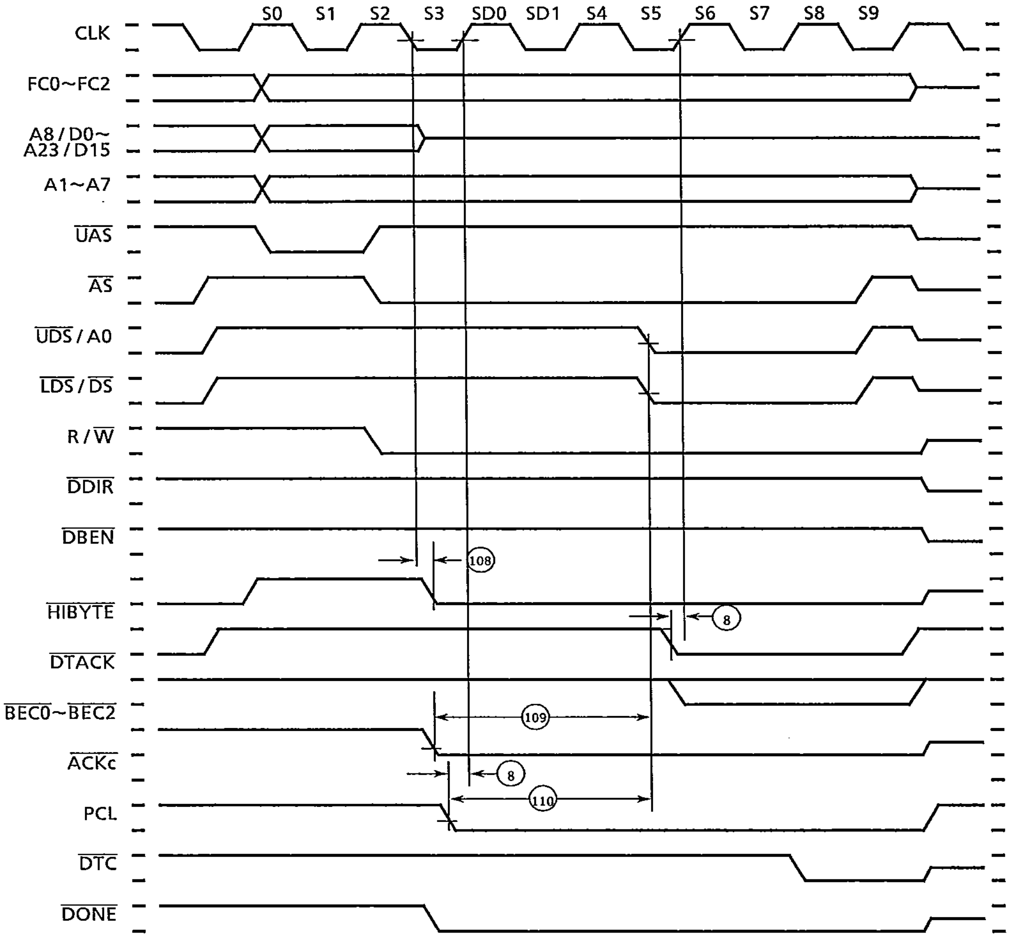
These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



Note : Timing measurements are referenced to and from a low voltage of 0.8 volts 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 volts and 2.0 volts.

Figure 6.8 Single Address Read Cycle 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 related diagrams for device operation.



Note : Timing measurements are referenced to and from a low voltage of 0.8 volts and 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 volts and 2.0 volts.

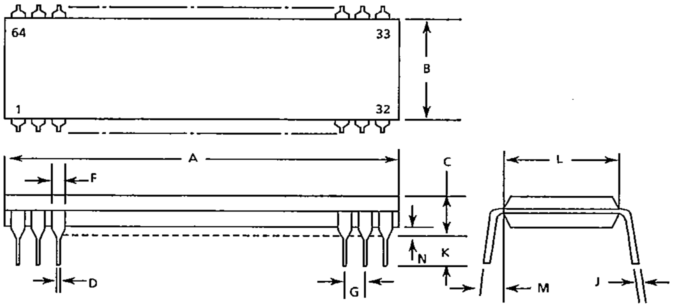
Figure 6.9 Single Address Write Cycle Timing Diagram

7. MECHANICAL DATA

The section contains package dimensions for the TMP68440.

7.1 PACKAGE DIMENSIONS

PLASTIC PACKAGE



MILLIMETERS

DIM	MIN	MAX
A	81.16	81.91
B	20.17	20.57
C	4.83	5.84
D	0.33	0.53
F	1.27	1.77
G	2.54BSC	
J	0.20	0.38
K	3.05	3.55
L	22.86BSC	
M	0°	15°
N	0.51	1.01

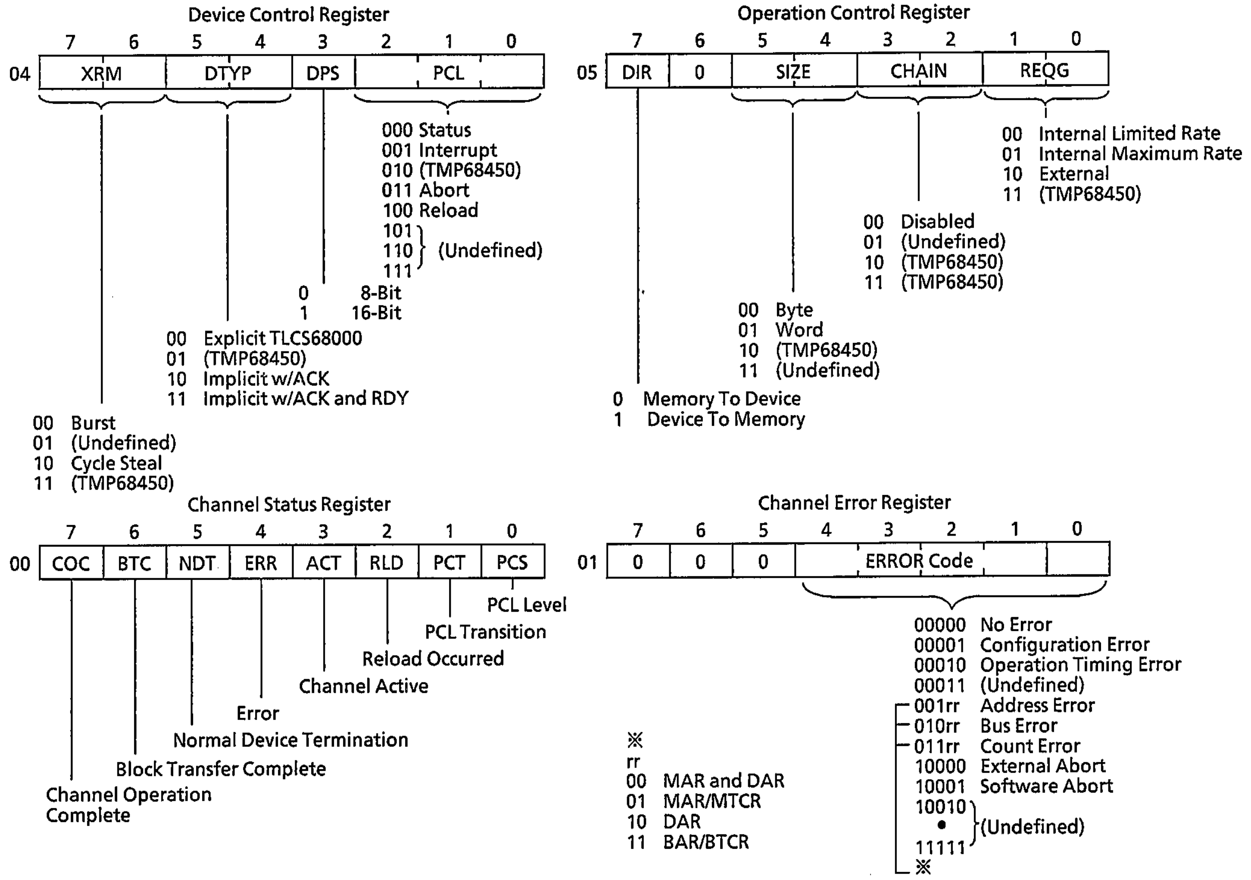


Figure 3.2 Register Summary(1/3)

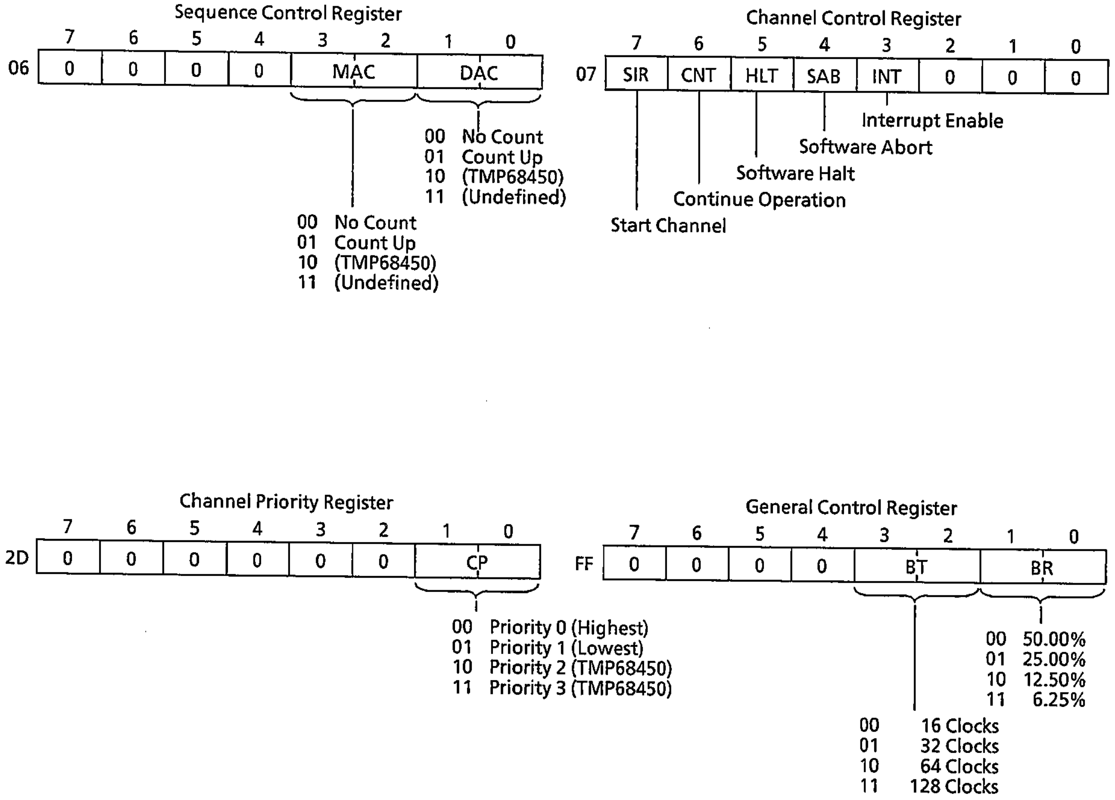
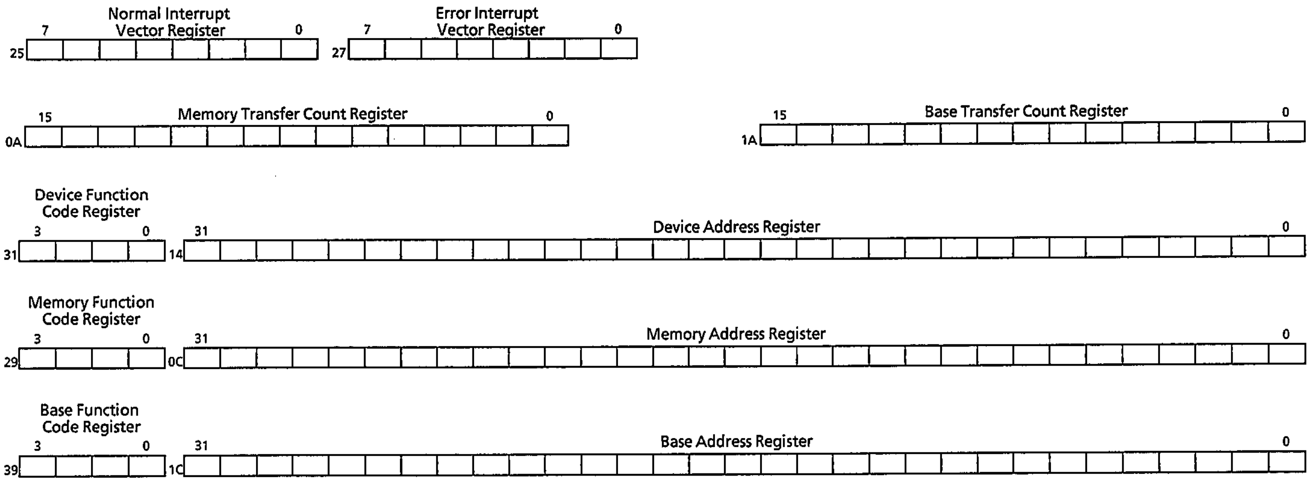


Figure 3.2 Register Summary (2/3)


Normal Interrupt Vector Register
7
0
25
Error Interrupt Vector Register
7
0
27
Memory Transfer Count Register
15
0
0A
Base Transfer Count Register
15
0
1A
Device Function Code Register
3
0
31
Device Address Register
31
0
14
Memory Function Code Register
3
0
29
Memory Address Register
31
0
0C
Base Function Code Register
3
0
39
Base Address Register
31
0
1C

Channel Base Addresses	:	CH0	00	CH2	80	(TMP68450)
		CH1	40	CH3	C0	(TMP68450)

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

Figure 3.2 Register Summary (3/3)