Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Obsolete
Core Processor	EC000
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	8MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	68-BCPGA
Supplier Device Package	68-PGA (26.92x26.92)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68hc000rc8

1.1 MC68000

The MC68000 is the first implementation of the M68000 16/-32 bit microprocessor architecture. The MC68000 has a 16-bit data bus and 24-bit address bus while the full architecture provides for 32-bit address and data buses. It is completely code-compatible with the MC68008 8-bit data bus implementation of the M68000 and is upward code compatible with the MC68010 virtual extensions and the MC68020 32-bit implementation of the architecture. Any user-mode programs using the MC68000 instruction set will run unchanged on the MC68008, MC68010, MC68020, MC68030, and MC68040. This is possible because the user programming model is identical for all processors and the instruction sets are proper subsets of the complete architecture.

1.2 MC68008

The MC68008 is a member of the M68000 family of advanced microprocessors. This device allows the design of cost-effective systems using 8-bit data buses while providing the benefits of a 32-bit microprocessor architecture. The performance of the MC68008 is greater than any 8-bit microprocessor and superior to several 16-bit microprocessors.

The MC68008 is available as a 48-pin dual-in-line package (plastic or ceramic) and 52-pin plastic leaded chip carrier. The additional four pins of the 52-pin package allow for additional signals: A20, A21, $\overline{\text{BGACK}}$, and $\overline{\text{IPL2}}$. The 48-pin version supports a 20-bit address that provides a 1-Mbyte address space; the 52-pin version supports a 22-bit address that extends the address space to 4 Mbytes. The 48-pin MC68008 contains a simple two-wire arbitration circuit; the 52-pin MC68008 contains a full three-wire MC68000 bus arbitration control. Both versions are designed to work with daisy-chained networks, priority encoded networks, or a combination of these techniques.

A system implementation based on an 8-bit data bus reduces system cost in comparison to 16-bit systems due to a more effective use of components and byte-wide memories and peripherals. In addition, the nonmultiplexed address and data buses eliminate the need for external demultiplexers, further simplifying the system.

The large nonsegmented linear address space of the MC68008 allows large modular programs to be developed and executed efficiently. A large linear address space allows program segment sizes to be determined by the application rather than forcing the designer to adopt an arbitrary segment size without regard to the application's individual requirements.

1.3 MC68010

The MC68010 utilizes VLSI technology and is a fully implemented 16-bit microprocessor with 32-bit registers, a rich basic instruction set, and versatile addressing modes. The vector base register (VBR) allows the vector table to be dynamically relocated

1.4 MC68HC000

The primary benefit of the MC68HC000 is reduced power consumption. The device dissipates an order of magnitude less power than the HMOS MC68000.

The MC68HC000 is an implementation of the M68000 16/-32 bit microprocessor architecture. The MC68HC000 has a 16-bit data bus implementation of the MC68000 and is upward code-compatible with the MC68010 virtual extensions and the MC68020 32-bit implementation of the architecture.

1.5 MC68HC001

The MC68HC001 provides a functional extension to the MC68HC000 HCMOS 16-/32-bit microprocessor with the addition of statically selectable 8- or 16-bit data bus operation. The MC68HC001 is object-code compatible with the MC68HC000, and code written for the MC68HC001 can be migrated without modification to any member of the M68000 Family.

1.6 MC68EC000

The MC68EC000 is an economical high-performance embedded controller designed to suit the needs of the cost-sensitive embedded controller market. The HCMOS MC68EC000 has an internal 32-bit architecture that is supported by a statically selectable 8- or 16-bit data bus. This architecture provides a fast and efficient processing device that can satisfy the requirements of sophisticated applications based on high-level languages.

The MC68EC000 is object-code compatible with the MC68000, and code written for the MC68EC000 can be migrated without modification to any member of the M68000 Family.

The MC68EC000 brings the performance level of the M68000 Family to cost levels previously associated with 8-bit microprocessors. The MC68EC000 benefits from the rich M68000 instruction set and its related high code density with low memory bandwidth requirements.

Table 3-1. Data Strobe Control of Data Bus

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8–D15	D0–D7
High	High	—	No Valid Data	No Valid Data
Low	Low	High	Valid Data Bits 15–8	Valid Data Bits 7–0
High	Low	High	No Valid Data	Valid Data Bits 7–0
Low	High	High	Valid Data Bus 15–8	No Valid Data
Low	Low	Low	Valid Data Bits 15–8	Valid Data Bits 7–0
High	Low	Low	Valid Data Bits 7–0*	Valid Data Bits 7–0
Low	High	Low	Valid Data Bits 15–8	Valid Data Bits 15–8*

*These conditions are a result of current implementation and may not appear on future devices.

Data Strobe ($\overline{DS}$) (MC68008)

This three-state signal and R/$\overline{W}$ control the flow of data on the data bus of the **MC68008**. Table 3-2 lists the combinations of these signals and the corresponding data on the bus. When the R/$\overline{W}$ line is high, the processor reads from the data bus. When the R/$\overline{W}$ line is low, the processor drives the data bus.

Table 3-2. Data Strobe Control of Data Bus (MC68008)

$\overline{DS}$	R/$\overline{W}$	D0–D7
1	—	No Valid Data
0	1	Valid Data Bits 7–0 (Read Cycle)
0	0	Valid Data Bits 7–0 (Write Cycle)

Data Transfer Acknowledge ($\overline{DTACK}$).

This input signal indicates the completion of the data transfer. When the processor recognizes $\overline{DTACK}$ during a read cycle, data is latched, and the bus cycle is terminated. When $\overline{DTACK}$ is recognized during a write cycle, the bus cycle is terminated.

3.4 BUS ARBITRATION CONTROL

The bus request, bus grant, and bus grant acknowledge signals form a bus arbitration circuit to determine which device becomes the bus master device. In the 48-pin version of the MC68008 and MC68EC000, no pin is available for the bus grant acknowledge signal; this microprocessor uses a two-wire bus arbitration scheme. All M68000 processors can use two-wire bus arbitration.

3.7 M6800 PERIPHERAL CONTROL

These control signals are used to interface the asynchronous M68000 processors with the synchronous M6800 peripheral devices. These signals are described in the following paragraphs.

Enable (E)

This signal is the standard enable signal common to all M6800 Family peripheral devices. A single period of clock E consists of 10 MC68000 clock periods (six clocks low, four clocks high). This signal is generated by an internal ring counter that may come up in any state. (At power-on, it is impossible to guarantee phase relationship of E to CLK.) The E signal is a free-running clock that runs regardless of the state of the MPU bus.

Valid Peripheral Address ($\overline{\text{VPA}}$)

This input signal indicates that the device or memory area addressed is an M6800 Family device or a memory area assigned to M6800 Family devices and that data transfer should be synchronized with the E signal. This input also indicates that the processor should use automatic vectoring for an interrupt. Refer to **Appendix B M6800 Peripheral Interface**.

Valid Memory Address ($\overline{\text{VMA}}$)

This output signal indicates to M6800 peripheral devices that the address on the address bus is valid and that the processor is synchronized to the E signal. This signal only responds to a $\overline{\text{VPA}}$ input that identifies an M6800 Family device.

The **MC68008** does not supply a $\overline{\text{VMA}}$ signal. This signal can be produced by a transistor-to-transistor logic (TTL) circuit; an example is described in **Appendix B M6800 Peripheral Interface**.

3.8 PROCESSOR FUNCTION CODES (FC0, FC1, FC2)

These function code outputs indicate the mode (user or supervisor) and the address space type currently being accessed, as shown in Table 3-3. The function code outputs are valid whenever $\overline{\text{AS}}$ is active.

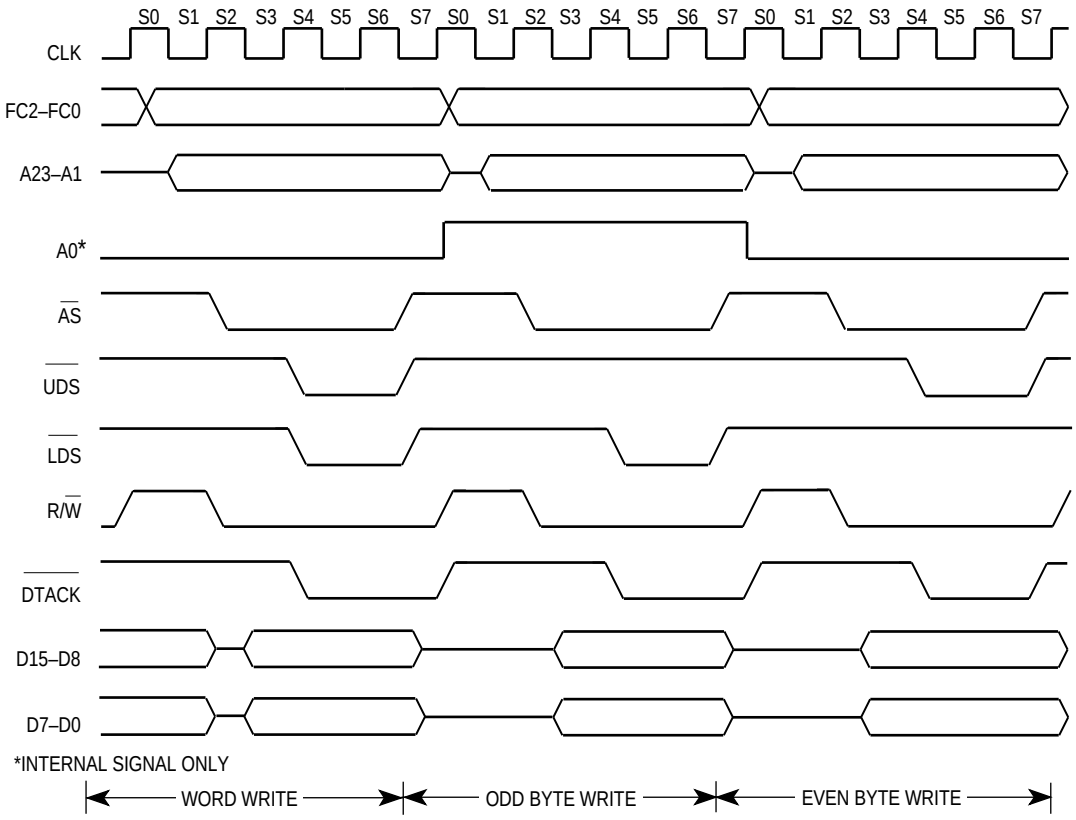


Figure 5-7. Word and Byte Write-Cycle Timing Diagram

The descriptions of the eight states of a write cycle are as follows:

STATE 0 The write cycle starts in S0. The processor places valid function codes on FC2–FC0 and drives R/$\overline{W}$ high (if a preceding write cycle has left R/$\overline{W}$ low).

STATE 1 Entering S1, the processor drives a valid address on the address bus.

STATE 2 On the rising edge of S2, the processor asserts $\overline{AS}$ and drives R/$\overline{W}$ low.

STATE 3 During S3, the data bus is driven out of the high-impedance state as the data to be written is placed on the bus.

STATE 4 At the rising edge of S4, the processor asserts $\overline{UDS}$, or $\overline{LDS}$. The processor waits for a cycle termination signal ($\overline{DTACK}$ or $\overline{BERR}$) or $\overline{VPA}$, an M6800 peripheral signal. When $\overline{VPA}$ is asserted during S4, the cycle becomes a peripheral cycle (refer to **Appendix B M6800 Peripheral Interface**. If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{DTACK}$ or $\overline{BERR}$ is asserted.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, no bus signals are altered.

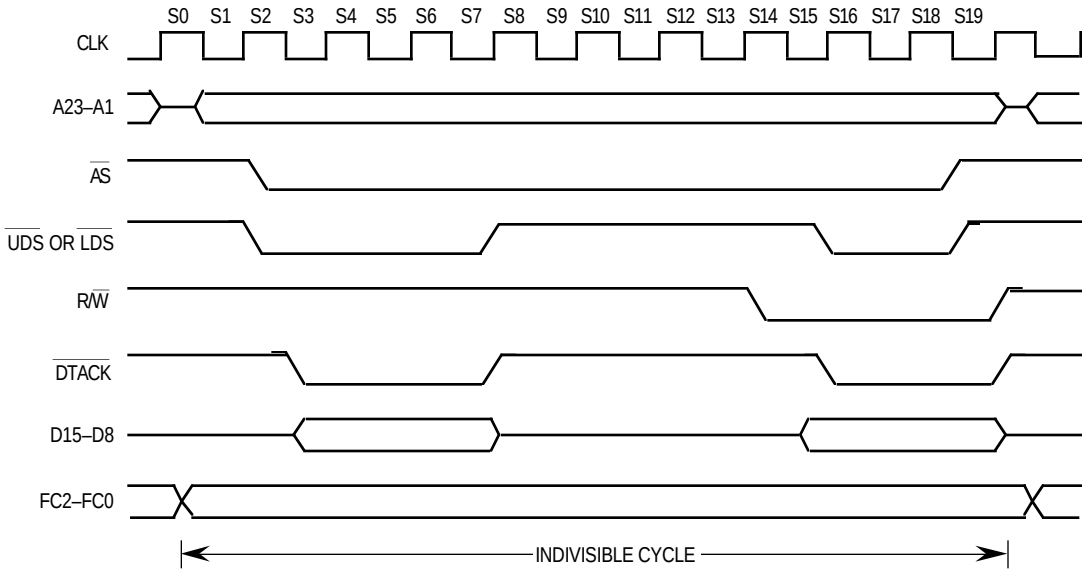


Figure 5-9. Read-Modify-Write Cycle Timing Diagram

The descriptions of the read-modify-write cycle states are as follows:

STATE 0 The read cycle starts in S0. The processor places valid function codes on FC2–FC0 and drives R/$\overline{W}$ high to identify a read cycle.

STATE 1 Entering S1, the processor drives a valid address on the address bus.

STATE 2 On the rising edge of S2, the processor asserts $\overline{AS}$ and $\overline{UDS}$, or $\overline{LDS}$.

STATE 3 During S3, no bus signals are altered.

STATE 4 During S4, the processor waits for a cycle termination signal ($\overline{DTACK}$ or $\overline{BERR}$) or $\overline{VPA}$, an M6800 peripheral signal. When $\overline{VPA}$ is asserted during S4, the cycle becomes a peripheral cycle (refer to **Appendix B M6800 Peripheral Interface**). If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{DTACK}$ or $\overline{BERR}$ is asserted.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, data from the device are driven onto the data bus.

STATE 7 On the falling edge of the clock entering S7, the processor accepts data from the device and negates $\overline{UDS}$, and $\overline{LDS}$. The device negates $\overline{DTACK}$ or $\overline{BERR}$ at this time.

STATES 8–11

The bus signals are unaltered during S8–S11, during which the arithmetic logic unit makes appropriate modifications to the data.

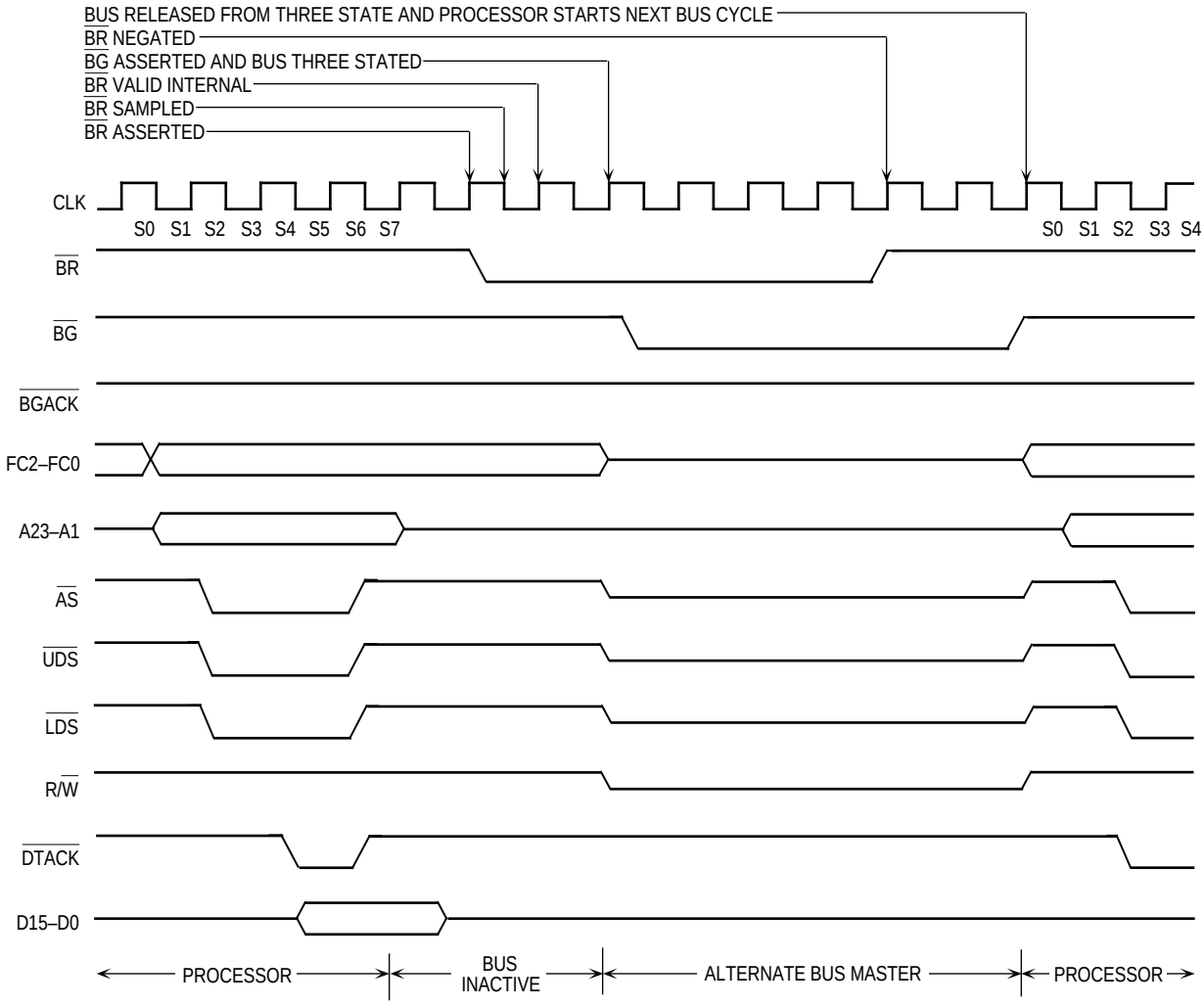


Figure 5-23. 2-Wire Bus Arbitration Timing Diagram—Bus Inactive

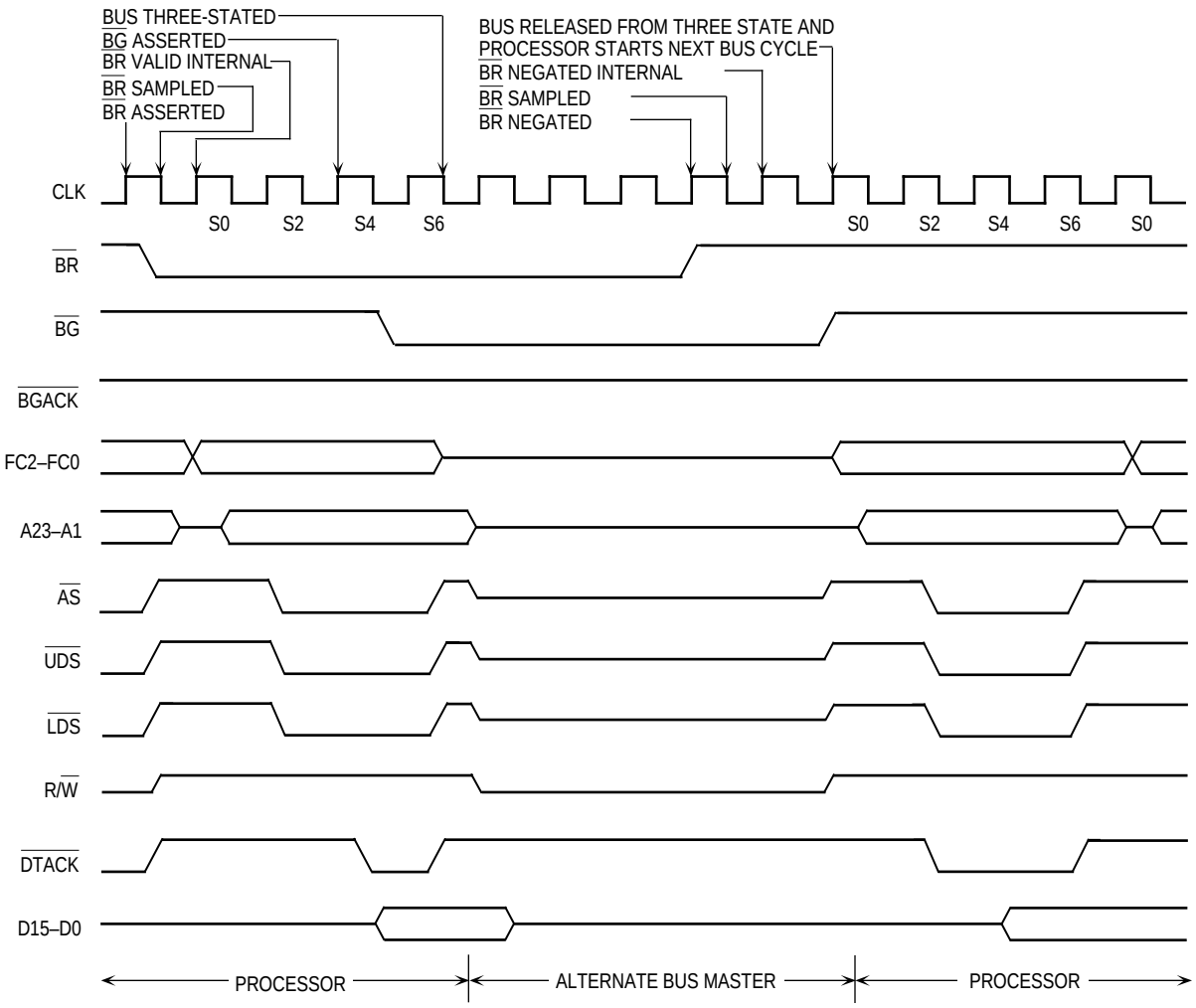


Figure 5-24. 2-Wire Bus Arbitration Timing Diagram—Special Case

5.4. BUS ERROR AND HALT OPERATION

In a bus architecture that requires a handshake from an external device, such as the asynchronous bus used in the M68000 Family, the handshake may not always occur. A bus error input is provided to terminate a bus cycle in error when the expected signal is not asserted. Different systems and different devices within the same system require different maximum-response times. External circuitry can be provided to assert the bus error signal after the appropriate delay following the assertion of address strobe.

In a virtual memory system, the bus error signal can be used to indicate either a page fault or a bus timeout. An external memory management unit asserts bus error when the page that contains the required data is not resident in memory. The processor suspends execution of the current instruction while the page is loaded into memory. The MC68010 pushes enough information on the stack to be able to resume execution of the instruction following return from the bus error exception handler.

The privilege mode is a mechanism for providing security in a computer system. Programs should access only their own code and data areas and should be restricted from accessing information that they do not need and must not modify. The operating system executes in the supervisor mode, allowing it to access all resources required to perform the overhead tasks for the user mode programs. Most programs execute in user mode, in which the accesses are controlled and the effects on other parts of the system are limited.

6.1.1 Supervisor Mode

The supervisor mode has the higher level of privilege. The mode of the processor is determined by the S bit of the status register; if the S bit is set, the processor is in the supervisor mode. All instructions can be executed in the supervisor mode. The bus cycles generated by instructions executed in the supervisor mode are classified as supervisor references. While the processor is in the supervisor mode, those instructions that use either the system stack pointer implicitly or address register seven explicitly access the SSP.

6.1.2 User Mode

The user mode has the lower level of privilege. If the S bit of the status register is clear, the processor is executing instructions in the user mode.

Most instructions execute identically in either mode. However, some instructions having important system effects are designated privileged. For example, user programs are not permitted to execute the STOP instruction or the RESET instruction. To ensure that a user program cannot enter the supervisor mode except in a controlled manner, the instructions that modify the entire status register are privileged. To aid in debugging systems software, the move to user stack pointer (MOVE to USP) and move from user stack pointer (MOVE from USP) instructions are privileged.

NOTE

To implement virtual machine concepts in the MC68010, the move from status register (MOVE from SR), move to/from control register (MOVEC), and move alternate address space (MOVES) instructions are also privileged.

The bus cycles generated by an instruction executed in user mode are classified as user references. Classifying a bus cycle as a user reference allows an external memory management device to translate the addresses of and control access to protected portions of the address space. While the processor is in the user mode, those instructions that use either the system stack pointer implicitly or address register seven explicitly access the USP.

6.1.3 Privilege Mode Changes

Once the processor is in the user mode and executing instructions, only exception processing can change the privilege mode. During exception processing, the current state of the S bit of the status register is saved, and the S bit is set, putting the processor in the

Table 6-3. Exception Grouping and Priority

Group	Exception	Processing
0	Reset Address Error Bus Error	Exception Processing Begins within Two Clock Cycles
1	Trace Interrupt Illegal Privilege	Exception Processing Begins before the Next Instruction
2	TRAP, TRAPV, CHK Zero Divide	Exception Processing Is Started by Normal Instruction Execution

6.2.4 Exception Stack Frames

Exception processing saves the most volatile portion of the current processor context on the top of the supervisor stack. This context is organized in a format called the exception stack frame. Although this information varies with the particular processor and type of exception, it always includes the status register and program counter of the processor when the exception occurred.

The amount and type of information saved on the stack are determined by the processor type and exception type. Exceptions are grouped by type according to priority of the exception.

Of the group 0 exceptions, the reset exception does not stack any information. The information stacked by a bus error or address error exception in the MC68000, MC68HC000, MC68HC001, MC68EC000, or MC68008 is described in **6.3.9.1 Bus Error** and shown in Figure 6-7.

The MC68000, MC68HC000, MC68HC001, MC68EC000, and MC68008 group 1 and 2 exception stack frame is shown in Figure 6-5. Only the program counter and status register are saved. The program counter points to the next instruction to be executed after exception processing.

The MC68010 exception stack frame is shown in Figure 5-6. The number of words actually stacked depends on the exception type. Group 0 exceptions (except reset) stack 29 words and group 1 and 2 exceptions stack four words. To support generic exception handlers, the processor also places the vector offset in the exception stack frame. The format code field allows the return from exception (RTE) instruction to identify what information is on the stack so that it can be properly restored. Table 6-4 lists the MC68010 format codes. Although some formats are specific to a particular M68000 Family processor, the format 0000 is always legal and indicates that just the first four words of the frame are present.

register on the supervisor stack. The offset value in the format/offset word on the MC68010 is the vector number multiplied by four. The format is all zeros. The saved value of the program counter is the address of the instruction that would have been executed had the interrupt not been taken. The appropriate interrupt vector is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine. Priority level 7 is a special case. Level 7 interrupts cannot be inhibited by the interrupt priority mask, thus providing a "nonmaskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level 7. A level 7 interrupt may still be caused by the level comparison if the request level is a 7 and the processor priority is set to a lower level by an instruction.

6.3.3 Uninitialized Interrupt

An interrupting device provides an M68000 interrupt vector number and asserts data transfer acknowledge ($\overline{\text{DTACK}}$), or asserts valid peripheral address ($\overline{\text{VPA}}$), or auto vector ($\overline{\text{AVEC}}$), or bus error ($\overline{\text{BERR}}$) during an interrupt acknowledge cycle by the MC68000. If the vector register has not been initialized, the responding M68000 Family peripheral provides vector number 15, the uninitialized interrupt vector. This response conforms to a uniform way to recover from a programming error.

6.3.4 Spurious Interrupt

During the interrupt acknowledge cycle, if no device responds by asserting $\overline{\text{DTACK}}$ or $\overline{\text{AVEC}}$, $\overline{\text{VPA}}$, $\overline{\text{BERR}}$ should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by forming a short format exception stack and fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

6.3.5 Instruction Traps

Traps are exceptions caused by instructions; they occur when a processor recognizes an abnormal condition during instruction execution or when an instruction is executed that normally traps during execution.

Exception processing for traps is straightforward. The status register is copied; the supervisor mode is entered; and tracing is turned off. The vector number is internally generated; for the TRAP instruction, part of the vector number comes from the instruction itself. The format/offset word (MC68010 only), the program counter, and the copy of the status register are saved on the supervisor stack. The offset value in the format/offset word on the MC68010 is the vector number multiplied by four. The saved value of the program counter is the address of the instruction following the instruction that generated the trap. Finally, instruction execution commences at the address in the exception vector.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a run-time error, which may be an arithmetic overflow or a subscript out of bounds.

6.3.7 Privilege Violations

To provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user mode causes an exception. The privileged instructions are as follows:

AND Immediate to SR	MOVE USP
EOR Immediate to SR	OR Immediate to SR
MOVE to SR (68010 only)	RESET
MOVE from SR (68010 only)	RTE
MOVEC (68010 only)	STOP
MOVES (68010 only)	

Exception processing for privilege violations is nearly identical to that for illegal instructions. After the instruction is fetched and decoded and the processor determines that a privilege violation is being attempted, the processor starts exception processing. The status register is copied; the supervisor mode is entered; and tracing is turned off. The vector number is generated to reference the privilege violation vector, and the current program counter and the copy of the status register are saved on the supervisor stack. If the processor is an MC68010, the format/offset word is also saved. The saved value of the program counter is the address of the first word of the instruction causing the privilege violation. Finally, instruction execution commences at the address in the privilege violation exception vector.

6.3.8 Tracing

To aid in program development, the M68000 Family includes a facility to allow tracing following each instruction. When tracing is enabled, an exception is forced after each instruction is executed. Thus, a debugging program can monitor the execution of the program under test.

The trace facility is controlled by the T bit in the supervisor portion of the status register. If the T bit is cleared (off), tracing is disabled and instruction execution proceeds from instruction to instruction as normal. If the T bit is set (on) at the beginning of the execution of an instruction, a trace exception is generated after the instruction is completed. If the instruction is not executed because an interrupt is taken or because the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. During the execution of the instruction, if an exception is forced by that instruction, the exception processing for the instruction exception occurs before that of the trace exception.

As an extreme illustration of these rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First, the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.

shown in Figure 6-9. If the bus cycle is a read, the data at the fault address should be written to the images of the data input buffer, instruction input buffer, or both according to the data fetch (DF) and instruction fetch (IF) bits.* In addition, for read-modify-write cycles, the status register image must be properly set to reflect the read data if the fault occurred during the read portion of the cycle and the write operation (i.e., setting the most significant bit of the memory location) must also be performed. These operations are required because the entire read-modify-write cycle is assumed to have been completed by software. Once the cycle has been completed by software, the rerun (RR) bit in the special status word is set to indicate to the processor that it should not rerun the cycle when the RTE instruction is executed. If the RR bit is set when an RTE instruction executes, the MC68010 reads all the information from the stack, as usual.

15	14	13	12	11	10	9	8	7 — 3	2 — 0
RR	*	IF	DF	RM	HB	BY	RW	*	FC2–FC0

RR — Rerun flag; 0=processor rerun (default), 1=software rerun
IF — Instruction fetch to the instruction input buffer
DF — Data fetch to the data input buffer
RM — Read-modify-write cycle
HB — High-byte transfer from the data output buffer or to the data input buffer
BY — Byte-transfer flag; HB selects the high or low byte of the transfer register. If BY is clear, the transfer is word.
RW — Read/write flag; 0=write, 1=read
FC — The function code used during the faulted access
* — These bits are reserved for future use by Motorola and will be zero when written by the MC68010.

Figure 6-9. Special Status Word Format

6.3.10 Address Error

An address error exception occurs when the processor attempts to access a word or long-word operand or an instruction at an odd address. An address error is similar to an internally generated bus error. The bus cycle is aborted, and the processor ceases current processing and begins exception processing. The exception processing sequence is the same as that for a bus error, including the information to be stacked, except that the vector number refers to the address error vector. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted.

On the MC68010, the address error exception stacks the same information stacked by a bus error exception. Therefore, the RTE instruction can be used to continue execution of the suspended instruction. However, if the RR flag is not set, the fault address is used when the cycle is retried, and another address error exception occurs. Therefore, the user must be certain that the proper corrections have been made to the stack image and user registers before attempting to continue the instruction. With proper software handling, the address error exception handler could emulate word or long-word accesses to odd addresses if desired.

*If the faulted access was a byte operation, the data should be moved from or to the least significant byte of the data output or input buffer images, unless the high-byte transfer (HB) bit is set. This condition occurs if a MOVEP instruction caused the fault during transfer of bits 8–15 of a word or long word or bits 24–31 of a long word.

In Table 7-5, the following notation applies:

An — Address register operand
Dn — Data register operand
ea — An operand specified by an effective address
M — Memory effective address operand

Table 7-5. Standard Instruction Execution Times

Instruction	Size	op<ea>, An	op<ea>, Dn	op Dn, <M>
ADD/ADDA	Byte Word Long	— **12**(2/0)+ **10**(2/0)+**	**8**(2/0)+ **8**(2/0)+ **10**(2/0)+**	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+
AND	Byte Word Long	— — —	**8**(2/0)+ **8**(2/0)+ **10**(2/0)+**	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+
CMP/CMPA	Byte Word Long	— **10**(2/0)+ **10**(2/0)+	**8**(2/0)+ **8**(2/0)+ **10**(2/0)+	— — —
DIVS DIVU	— —	— —	**162**(2/0)+* **144**(2/0)+*	— —
EOR	Byte, Word, Long	— — —	**8**(2/0)+*** **8**(2/0)+*** **12**(2/0)+***	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+
MULS MULU	— —	— —	**74**(2/0)+* **74**(2/0)+*	— —
OR	Byte, Word Long	— — —	**8**(2/0)+ **8**(2/0)+ **10**(2/0)+**	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+
SUB	Byte, Word Long	 **12**(2/0)+ **10**(2/0)+**	**8**(2/0)+ **8**(2/0)+ **10**(2/0)+**	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+

\+ Add effective address calculation time.
* Indicates maximum base value added to word effective address time
** The base time of 10 clock periods is increased to 12 if the effective address mode is register direct or immediate (effective address time should also be added).
*** Only available effective address mode is data register direct.
DIVS, DIVU — The divide algorithm used by the MC68008 provides less than 10% difference between the best- and worst-case timings.
MULS, MULU — The multiply algorithm requires 42+2n clocks where n is defined as:
MULS: n = tag the <ea> with a zero as the MSB; n is the resultant number of 10 or 01 patterns in the 17-bit source; i.e., worst case happens when the source is $5555.
MULU: n = the number of ones in the <ea>

7.4 IMMEDIATE INSTRUCTION EXECUTION TIMES

The numbers of clock periods shown in Table 7-6 include the times to fetch immediate operands, perform the operations, store the results, and read the next operation. The total number of clock periods, the number of read cycles, and the number of write cycles are

Table 8-4. Standard Instruction Execution Times

Instruction	Size	op<ea>, An†	op<ea>, Dn	op Dn, <M>
ADD/ADDA	Byte, Word	**8**(1/0)+	**4**(1/0)+	**8**(1/1)+
	Long	**6**(1/0)+**	**6**(1/0)+**	**12**(1/2)+
AND	Byte, Word	—	**4**(1/0)+	**8**(1/1)+
	Long	—	**6**(1/0)+**	**12**(1/2)+
CMP/CMPA	Byte, Word	**6**(1/0)+	**4**(1/0)+	—
	Long	**6**(1/0)+	**6**(1/0)+	—
DIVS	—	—	**158**(1/0)+*	—
DIVU	—	—	**140**(1/0)+*	—
EOR	Byte, Word	—	**4**(1/0)***	**8**(1/1)+
	Long	—	**8**(1/0)***	**12**(1/2)+
MULS	—	—	**70**(1/0)+*	—
MULU	—	—	**70**(1/0)+*	—
OR	Byte, Word	—	**4**(1/0)+	**8**(1/1)+
	Long	—	**6**(1/0)+**	**12**(1/2)+
SUB	Byte, Word	**8**(1/0)+	**4**(1/0)+	**8**(1/1)+
	Long	**6**(1/0)+**	**6**(1/0)+**	**12**(1/2)+

\+ Add effective address calculation time.
† Word or long only
* Indicates maximum basic value added to word effective address time
** The base time of six clock periods is increased to eight if the effective address mode is register direct or immediate (effective address time should also be added).
*** Only available effective address mode is data register direct.
DIVS, DIVU — The divide algorithm used by the MC68000 provides less than 10% difference between the best- and worst-case timings.
MULS, MULU — The multiply algorithm requires 38+2n clocks where n is defined as:
MULU: n = the number of ones in the <ea>
MULS: n=concatenate the <ea> with a zero as the LSB; n is the resultant number of 10 or 01 patterns in the 17-bit source; i.e., worst case happens when the source is $5555.

8.4 IMMEDIATE INSTRUCTION EXECUTION TIMES

The numbers of clock periods shown in Table 8-5 include the times to fetch immediate operands, perform the operations, store the results, and read the next operation. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format. The number of clock periods, the number of read cycles, and the number of write cycles, respectively, must be added to those of the effective address calculation where indicated by a plus sign (+).

In Table 8-5, the following notation applies:

\# — Immediate operand
Dn — Data register operand
An — Address register operand
M — Memory operand

9.4 IMMEDIATE INSTRUCTION EXECUTION TIMES

The numbers of clock periods shown in Table 9-8 include the times to fetch immediate operands, perform the operations, store the results, and read the next operation. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format. The number of clock periods, the number of read cycles, and the number of write cycles, respectively, must be added to those of the effective address calculation where indicated by a plus sign (+).

In Tables 9-8, the following notation applies:

— Immediate operand
Dn — Data register operand
An — Address register operand
M — Memory operand

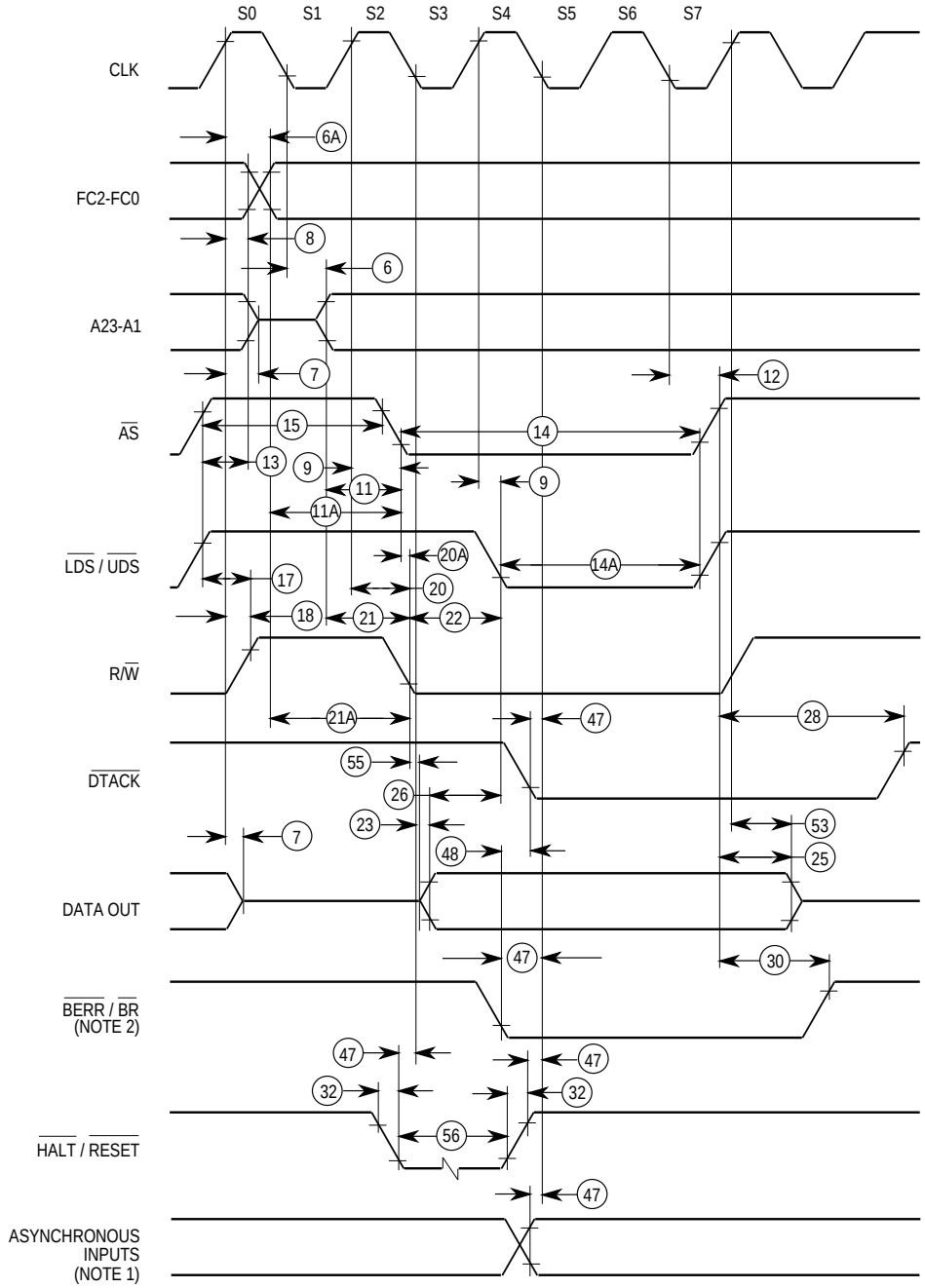
Table 9-8. Immediate Instruction Execution Times

Instruction	Size	op #, Dn	op #, An	op #, M
ADDI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**14**(3/0)	—	**20**(3/2)+
ADDQ	Byte, Word	**4**(1/0)	**4**(1/0)*	**8**(1/2)+
	Long	**8**(1/0)	**8**(1/1)	**12**(1/2)+
ANDI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**14**(3/0)	—	**20**(3/1)+
CMPI	Byte, Word	**8**(2/0)	—	**8**(2/0)+
	Long	**12**(3/0)	—	**12**(3/0)+
EORI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**14**(3/0)	—	**20**(3/2)+
MOVEQ	Long	**4**(1/0)	—	—
ORI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**14**(3/0)	—	**20**(3/2)+
SUBI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**14**(3/0)	—	**20**(3/2)+
SUBQ	Byte, Word	**4**(1/0)	**4**(1/0)*	**8**(1/1)+
	Long	**8**(1/0)	**8**(1/0)	**12**(1/2)+

+Add effective address calculation time.
*Word only.

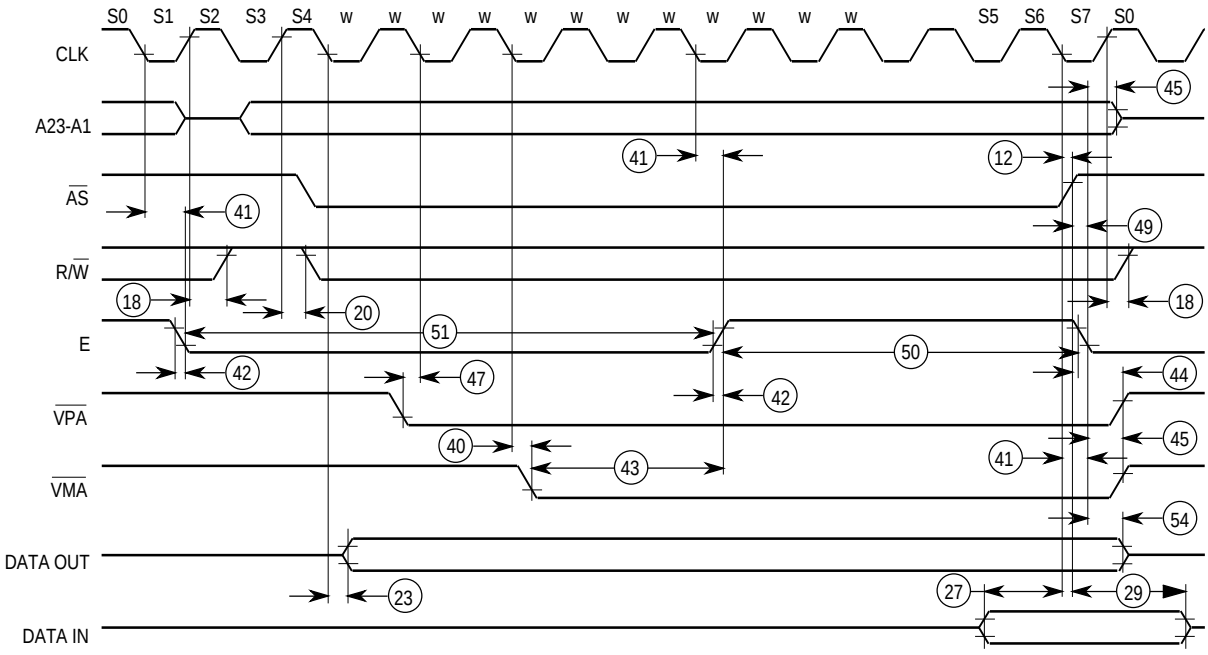
9.5 SINGLE OPERAND INSTRUCTION EXECUTION TIMES

Tables 9-9, 9-10, and 9-11 list the timing data for the single operand instructions. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format. The number of clock periods, the number of read cycles, and the number of write cycles, respectively, must be added to those of the effective address calculation where indicated by a plus sign (+).



NOTES:
1. Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall is linear between 0.8 V and 2.0 V.
2. Because of loading variations, R/W may be valid after AS even though both are initiated by the rising edge of S2 (specification #20A).

Figure 10-5. Write Cycle Timing Diagram
(Applies To All Processors Except The MC68EC000)



NOTE: This timing diagram is included for those who wish to design their own circuit to generate $\overline{VMA}$. It shows the best case possible attainable

Figure 10-6. MC68000 to M6800 Peripheral Timing Diagram (Best Case)

(Applies To All Processors Except The MC68EC000)

Table A-1. MC68010 Loop Mode Instructions

Opcodes	Applicable Addressing Modes
MOVE [BWL]	(Ay) to (Ax) (Ay) to (Ax)+ (Ay) to –(Ax) (Ay)+ to (Ax) (Ay)+ to –(Ax) –(Ay) to (Ax) –(Ay) to (Ax)+ –(Ay) to –(Ax) Ry to (Ax) Ry to (Ax)+
ADD [BWL] AND [BWL] CMP [BWL] OR [BWL] SUB [BWL]	(Ay) to Dx (Ay)+ to Dx –(Ay) to Dx
ADDA [WL] CMPA [WL] SUBA [WL]	(Ay) to Ax –(Ay) to Ax (Ay)+ to Ax
ADD [BWL] AND [BWL] EOR [BWL] OR [BWL] SUB [BWL]	Dx to (Ay) Dx to (Ay)+ Dx to –(Ay)
ABCD [B] ADDX [BWL] SBCD [B] SUBX [BWL]	–(Ay) to –(Ax)
CMP [BWL]	(Ay)+ to (Ax)+
CLR [BWL] NEG [BWL] NEGX [BWL} NOT [BWL] TST [BWL] NBCD [B]	(Ay) (Ay)+ –(Ay)
ASL [W] ASR [W] LSL [W] LSR [W] ROL [W] ROR [W] ROXL [W] ROXR	(Ay) by #1 (Ay)+ by #1 –(Ay) by #1

NOTE: [B, W, or L] indicate an operand size of byte, word, or long word.

After recognizing $\overline{VPA}$, the processor assures that enable (E) is low by waiting, if necessary, and subsequently asserts $\overline{VMA}$. $\overline{VMA}$ is then used as part of the chip-select equation of the peripheral to ensure correct timing for selection and deselection of the M6800 device. Once selected, the peripheral runs its cycle during the high portion of the E signal. Figure B-4 shows the best-case timing of an M6800 cycle, and Figure B-5 shows the worst-case timing. The cycle length is entirely dependent on the relationship of the assertion of $\overline{VPA}$ to the E clock.

When external circuitry asserts $\overline{VPA}$ as soon as possible following the assertion of $\overline{AS}$, the assertion of $\overline{VPA}$ is recognized on the falling edge of S4. In this case, no extra wait states are inserted (waiting for the assertion of $\overline{VPA}$). The only wait states inserted are those required to synchronize with the E clock. The synchronization delay is an integral number of system clock cycles within the following extremes:

1. Best Case—the assertion of $\overline{VPA}$ is recognized on the falling edge three clock cycles before E rises (or three clock cycles after E falls).
2. Worst Case—the assertion of $\overline{VPA}$ is recognized on the falling edge two clock cycles before E rises (or four clock cycles after E falls).

The processor latches the peripheral data in state 6 (S6) during a read cycle. For all cycles, the processor negates the address and data strobes one-half clock cycle later in state 7 (S7), and E goes low at this time. Another half clock later, the address bus is placed in the high-impedance state, and R/$\overline{W}$ is driven high. Logic in the peripheral must remove $\overline{VPA}$ within one clock after the negation of address strobe.

Data transfer acknowledge ($\overline{DTACK}$) must not be asserted while VPA is asserted. The state machine in the processor looks for $\overline{DTACK}$ to identify an asynchronous bus cycle and for $\overline{VPA}$ to identify a synchronous peripheral bus cycle. If both signals are asserted, the operation of the state machine is unpredictable.

To allow the processor to place its buses in the high-impedance state during DMA requests without inadvertently selecting the peripherals, $\overline{VMA}$ is active low for the M68000 Family of processors. The active-low $\overline{VMA}$ is in contrast to the active-high VMA signal of the M6800.

B.2 INTERRUPT INTERFACE OPERATION

During an interrupt acknowledge cycle while the processor is fetching the vector, $\overline{VPA}$ is asserted, and the processor (or external circuitry) asserts $\overline{VMA}$ and completes a normal M6800 read cycle as shown in Figure B-6. For the interrupt vector, the processor uses an internally generated vector number called an autovector. The autovector corresponds to the interrupt level being serviced. The seven autovectors are decimal vector numbers 25–31.

B

The autovector operation, which can be used with all peripherals, is similar to the normal interrupt acknowledge cycle. The autovector capability provides vectors for each of the six maskable interrupt levels and for the nonmaskable interrupt level. Whether the device supplies the vector number or the processor generates an autovector number, the