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	16MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.21x24.21)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68hc000cei16

TABLE OF CONTENTS (Continued)

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.

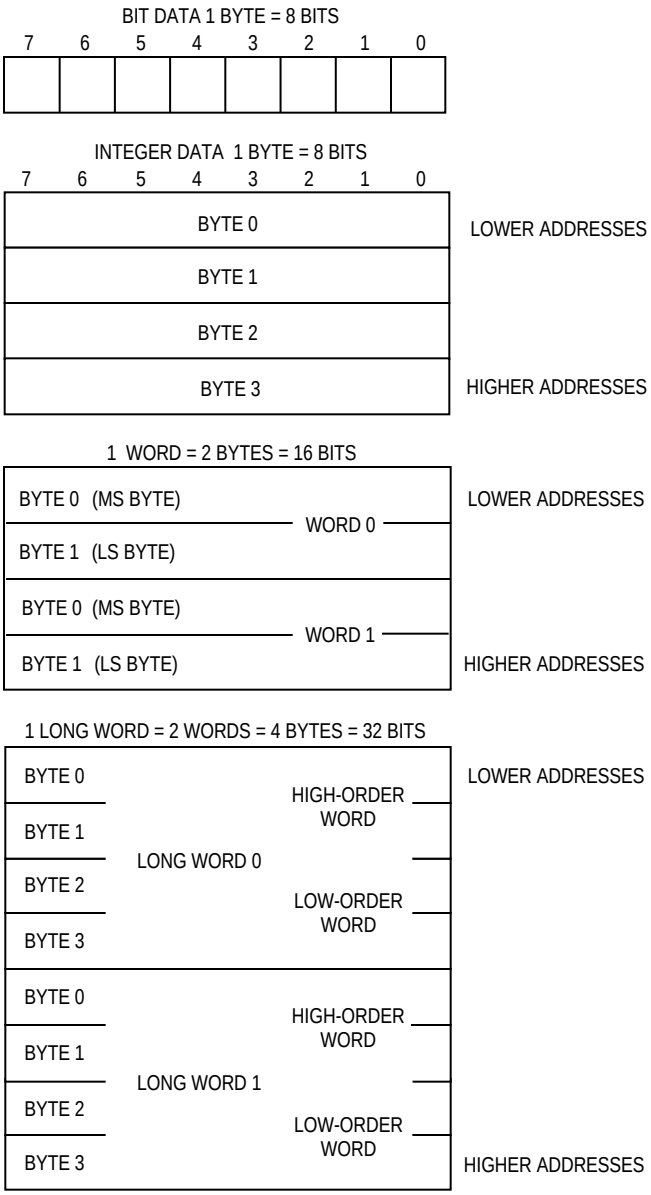


Figure 2-7. Memory Data Organization of the MC68008

2.5 INSTRUCTION SET SUMMARY

Table 2-2 provides an alphabetized listing of the M68000 instruction set listed by opcode, operation, and syntax. In the syntax descriptions, the left operand is the source operand, and the right operand is the destination operand. The following list contains the notations used in Table 2-2.

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.

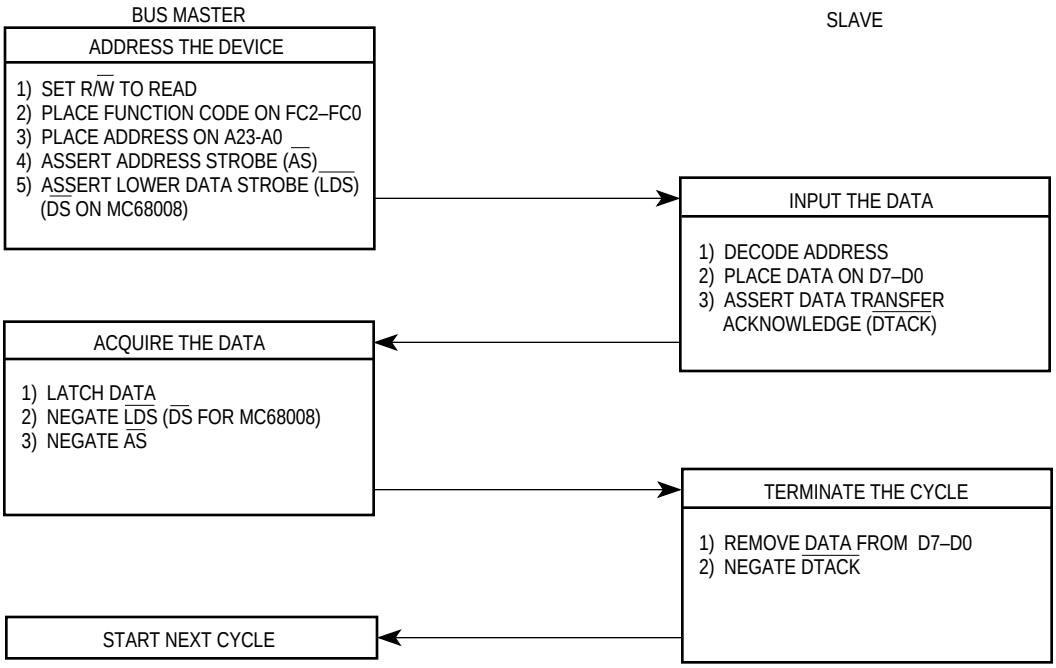


Figure 4-1. Byte Read-Cycle Flowchart

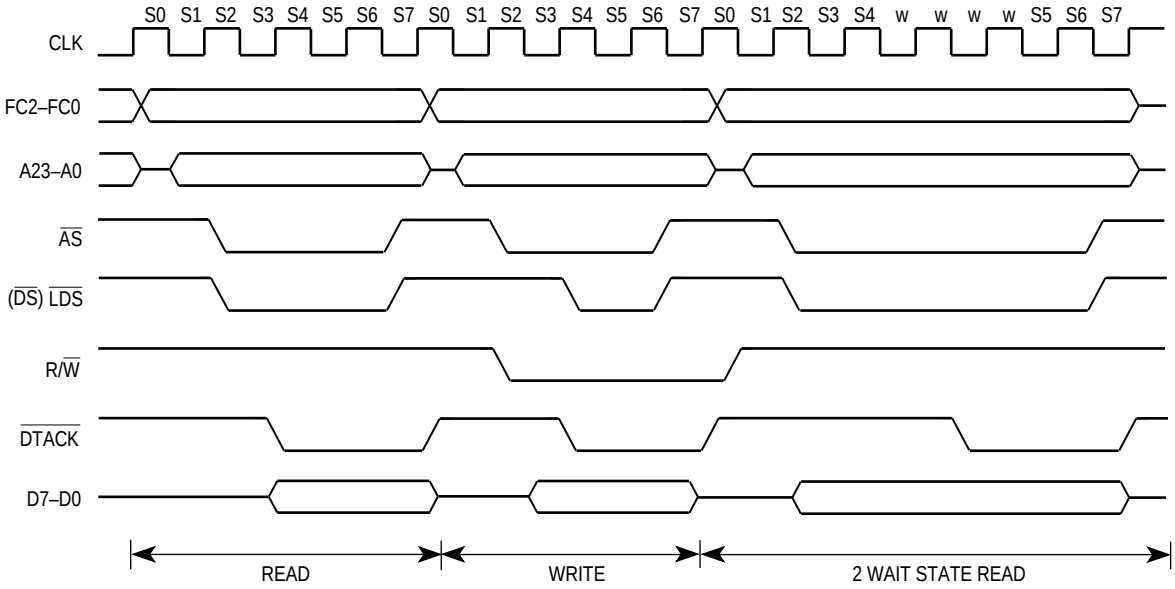


Figure 4-2. Read and Write-Cycle Timing Diagram

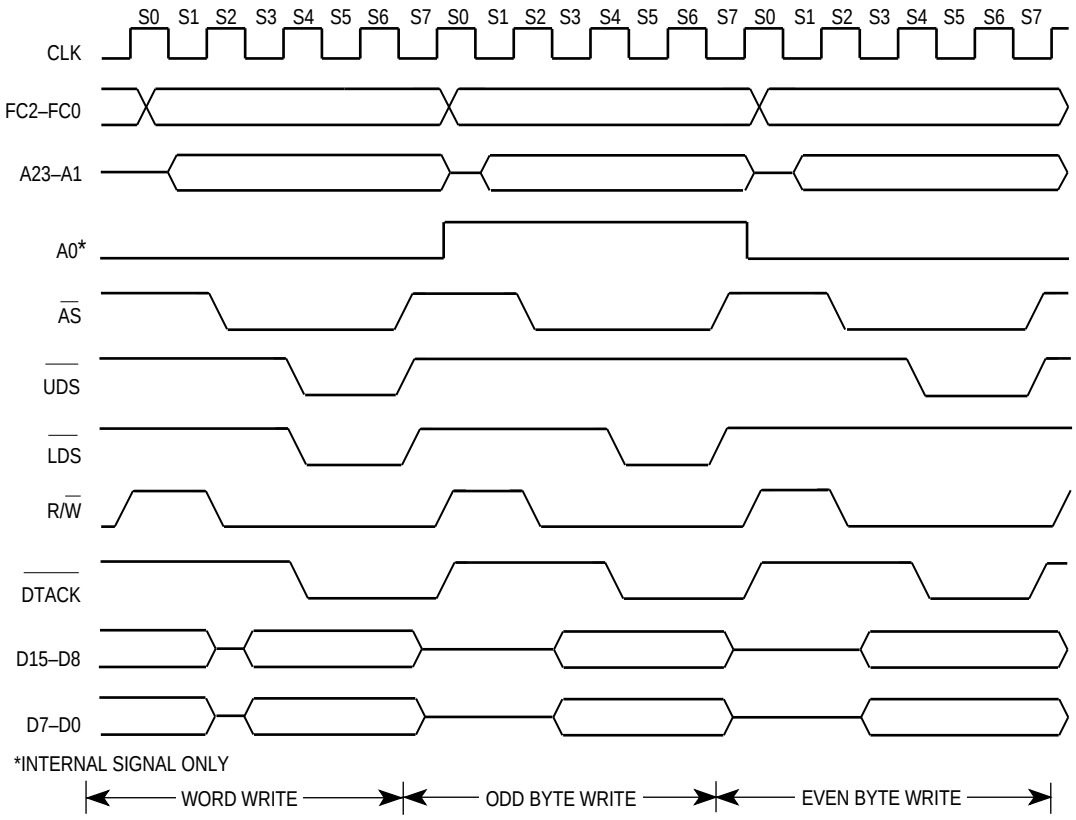


Figure 5-7. Word and Byte Write-Cycle Timing Diagram

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.

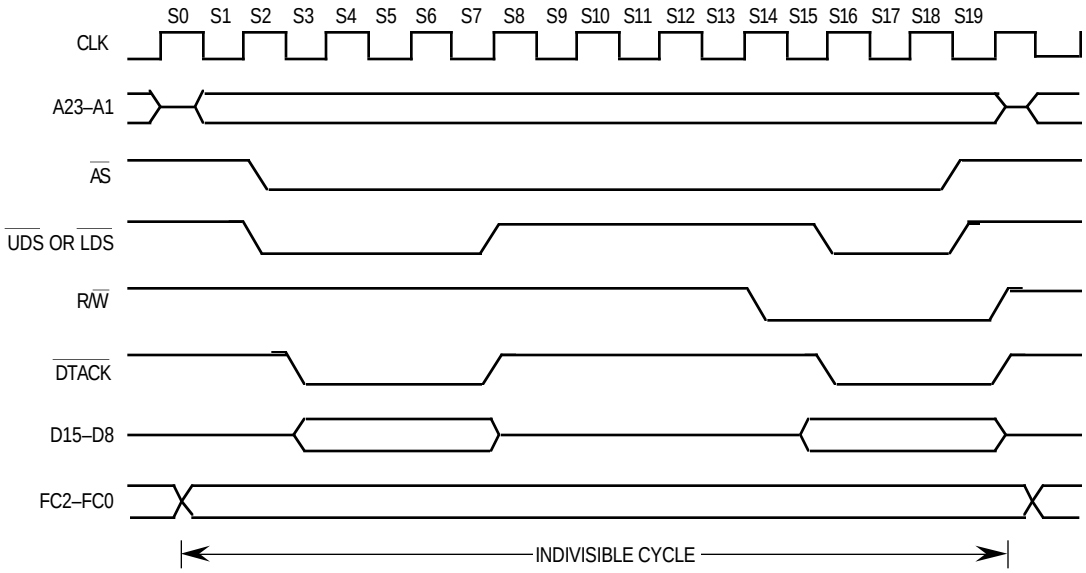


Figure 5-9. Read-Modify-Write Cycle Timing Diagram

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.

4. For an MC68010, return DTACK before data verification. If data is invalid, assert BERR on the next clock cycle (case 4).

Table 5-6. BERR and HALT Negation Results

Conditions of Termination in Table 4-4	Control Signal	Negated on Rising Edge of State			Results—Next Cycle
		N		N+2	
Bus Error	BERR HALT	• •	or or	• •	Takes bus error trap.
Rerun	BERR HALT	• •	or	•	Illegal sequence; usually traps to vector number 0.
Rerun	BERR HALT	•		•	Reruns the bus cycle.
Normal	BERR HALT	• •	or	•	May lengthen next cycle.
Normal	BERR HALT	•	or	• none	If next cycle is started, it will be terminated as a bus error.

• = Signal is negated in this bus state.

5.7 ASYNCHRONOUS OPERATION

To achieve clock frequency independence at a system level, the bus can be operated in an asynchronous manner. Asynchronous bus operation uses the bus handshake signals to control the transfer of data. The handshake signals are AS, UDS, LDS, DS (MC68008 only), DTACK, BERR, HALT, AVEC (MC68EC000 only), and VPA (only for M6800 peripheral cycles). AS indicates the start of the bus cycle, and UDS, LDS, and DS signal valid data for a write cycle. After placing the requested data on the data bus (read cycle) or latching the data (write cycle), the slave device (memory or peripheral) asserts DTACK to terminate the bus cycle. If no device responds or if the access is invalid, external control logic asserts BERR, or BERR and HALT, to abort or retry the cycle. Figure 5-31 shows the use of the bus handshake signals in a fully asynchronous read cycle. Figure 5-32 shows a fully asynchronous write cycle.

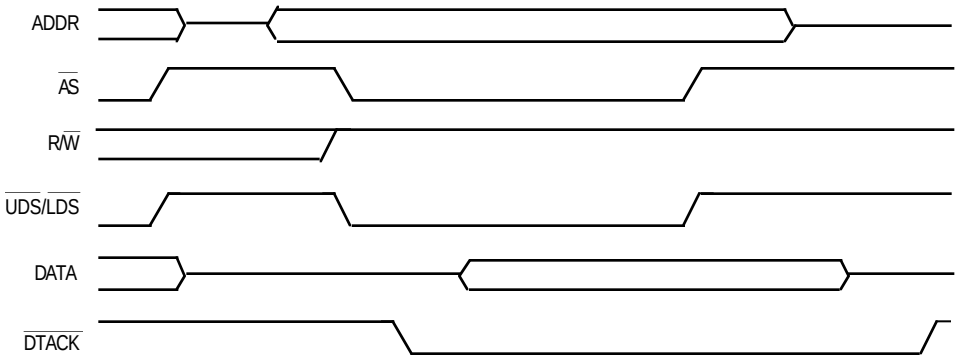
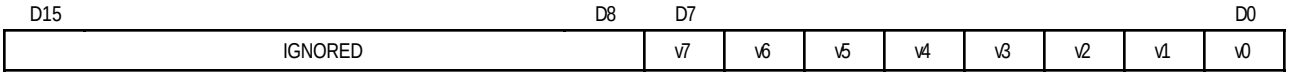


Figure 5-31. Fully Asynchronous Read Cycle



Where:
v7 is the MSB of the vector number
v0 is the LSB of the vector number

Figure 6-2. Peripheral Vector Number Format

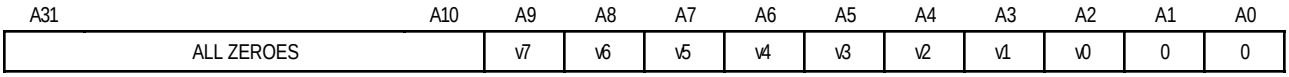


Figure 6-3. Address Translated from 8-Bit Vector Number (MC68000, MC68HC000, MC68HC001, MC68EC000, and MC68008)

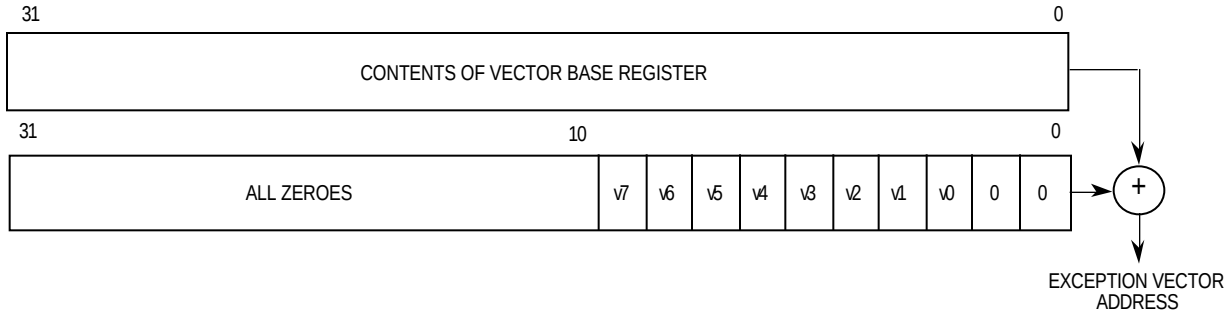


Figure 6-4. Exception Vector Address Calculation (MC68010)

The actual address on the address bus is truncated to the number of address bits available on the bus of the particular implementation of the M68000 architecture. In all processors except the MC68008, this is 24 address bits. (A0 is implicitly encoded in the data strobes.) In the MC68008, the address is 20 or 22 bits in length. The memory map for exception vectors is shown in Table 6-2.

The vector table, Table 6-2, is 512 words long (1024 bytes), starting at address 0 (decimal) and proceeding through address 1023 (decimal). The vector table provides 255 unique vectors, some of which are reserved for trap and other system function vectors. Of the 255, 192 are reserved for user interrupt vectors. However, the first 64 entries are not protected, so user interrupt vectors may overlap at the discretion of the systems designer.

6.2.2 Kinds of Exceptions

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts, the bus error, and reset. The interrupts are

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.

Table 8-5. Immediate Instruction Execution Times

Instruction	Size	op #, Dn	op #, An	op #, M
ADDI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**16**(3/0)	—	**20**(3/2)+
ADDQ	Byte, Word	**4**(1/0)	**4**(1/0)*	**8**(1/1)+
	Long	**8**(1/0)	**8**(1/0)	**12**(1/2)+
ANDI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**14**(3/0)	—	**20**(3/2)+
CMPI	Byte, Word	**8**(2/0)	—	**8**(2/0)+
	Long	**14**(3/0)	—	**12**(3/0)+
EORI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**16**(3/0)	—	**20**(3/2)+
MOVEQ	Long	**4**(1/0)	—	—
ORI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**16**(3/0)	—	**20**(3/2)+
SUBI	Byte, Word	**8**(2/0)	—	**12**(2/1)+
	Long	**16**(3/0)	—	**20**(3/2)+
SUBQ	Byte, Word	**4**(1/0)	**8**(1/0)*	**8**(1/1)+
	Long	**8**(1/0)	**8**(1/0)	**12**(1/2)+

8.5 SINGLE OPERAND INSTRUCTION EXECUTION TIMES

Table 8-6 lists 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 (+).

Table 9-2. Move Byte and Word Instruction Execution Times

Source	Destination								
	Dn	An	(An)	(An)+	–(An)	(d_{16}, An)	(d_8, An, Xn)*	(xxx).W	(xxx).L
Dn	**4**(1/0)	**4**(1/0)	**8**(1/1)	**8**(1/1)	**8**(1/1)	**12**(2/1)	**14**(2/1)	**12**(2/1)	**16**(3/1)
An	**4**(1/0)	**4**(1/0)	**8**(1/1)	**8**(1/1)	**8**(1/1)	**12**(2/1)	**14**(2/1)	**12**(2/1)	**16**(3/1)
(An)	**8**(2/0)	**8**(2/0)	**12**(2/1)	**12**(2/1)	**12**(2/1)	**16**(3/1)	**18**(3/1)	**16**(3/1)	**20**(4/1)
(An)+	**8**(2/0)	**8**(2/0)	**12**(2/1)	**12**(2/1)	**12**(2/1)	**16**(3/1)	**18**(3/1)	**16**(3/1)	**20**(4/1)
–(An)	**10**(2/0)	**10**(2/0)	**14**(2/1)	**14**(2/1)	**14**(2/1)	**18**(3/1)	**20**(3/1)	**18**(3/1)	**22**(4/1)
(d_{16}, An)	**12**(3/0)	**12**(3/0)	**16**(3/1)	**16**(3/1)	**16**(3/1)	**20**(4/1)	**22**(4/1)	**20**(4/1)	**24**(5/1)
(d_8, An, Xn)*	**14**(3/0)	**14**(3/0)	**18**(3/1)	**18**(3/1)	**18**(3/1)	**22**(4/1)	**24**(4/1)	**22**(4/1)	**26**(5/1)
(xxx).W	**12**(3/0)	**12**(3/0)	**16**(3/1)	**16**(3/1)	**16**(3/1)	**20**(4/1)	**22**(4/1)	**20**(4/1)	**24**(5/1)
(xxx).L	**16**(4/0)	**16**(4/0)	**20**(4/1)	**20**(4/1)	**20**(4/1)	**24**(5/1)	**26**(5/1)	**24**(5/1)	**28**(6/1)
(d_{16}, PC)	**12**(3/0)	**12**(3/0)	**16**(3/1)	**16**(3/1)	**16**(3/1)	**20**(4/1)	**22**(4/1)	**20**(4/1)	**24**(5/1)
(d_8, PC, Xn)*	**14**(3/0)	**14**(3/0)	**18**(3/1)	**18**(3/1)	**18**(3/1)	**22**(4/1)	**24**(4/1)	**22**(4/1)	**26**(5/1)
#<data>	**8**(2/0)	**8**(2/0)	**12**(2/1)	**12**(2/1)	**12**(2/1)	**16**(3/1)	**18**(3/1)	**16**(3/1)	**20**(4/1)

*The size of the index register (Xn) does not affect execution time.

Table 9-3. Move Byte and Word Instruction Loop Mode Execution Times

Source	Loop Continued			Loop Terminated					
	Valid Count, cc False			Valid count, cc True			Expired Count		
	Destination								
	(An)	(An)+	–(An)	(An)	(An)+	–(An)	(An)	(An)+	–(An)
Dn	**10**(0/1)	**10**(0/1)	—	**18**(2/1)	**18**(2/1)	—	**16**(2/1)	**16**(2/1)	—
An*	**10**(0/1)	**10**(0/1)	—	**18**(2/1)	**18**(2/1)	—	**16**(2/1)	**16**(2/1)	—
(An)	**14**(1/1)	**14**(1/1)	**16**(1/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)	**18**(3/1)	**18**(3/1)	**20**(3/1)
(An)+	**14**(1/1)	**14**(1/1)	**16**(1/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)	**18**(3/1)	**18**(3/1)	**20**(3/1)
–(An)	**16**(1/1)	**16**(1/1)	**18**(1/1)	**22**(3/1)	**22**(3/1)	**24**(3/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)

*Word only.

Table 9-11. Single Operand Instruction Loop Mode Execution Times

		Loop Continued			Loop Terminated					
		Valid Count, cc False			Valid Count, cc True			Expired Count		
Instruction	Size	(An)	(An)+	–(An)	(An)	(An)+	–(An)	(An)	(An)+	–(An)
CLR	Byte, Word	**10**(0/1)	**10**(0/1)	**12**(0/1)	**18**(2/1)	**18**(2/1)	**20**(2/0)	**16**(2/1)	**16**(2/1)	**18**(2/1)
	Long	**14**(0/2)	**14**(0/2)	**16**(0/2)	**22**(2/2)	**22**(2/2)	**24**(2/2)	**20**(2/2)	**20**(2/2)	**22**(2/2)
NBCD	Byte	**18**(1/1)	**18**(1/1)	**20**(1/1)	**24**(3/1)	**24**(3/1)	**26**(3/1)	**22**(3/1)	**22**(3/1)	**24**(3/1)
NEG	Byte, Word	**16**(1/1)	**16**(1/1)	**18**(2/2)	**22**(3/1)	**22**(3/1)	**24**(3/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)
	Long	**24**(2/2)	**24**(2/2)	**26**(2/2)	**30**(4/2)	**30**(4/2)	**32**(4/2)	**28**(4/2)	**28**(4/2)	**30**(4/2)
NEGX	Byte, Word	**16**(1/1)	**16**(1/1)	**18**(2/2)	**22**(3/1)	**22**(3/1)	**24**(3/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)
	Long	**24**(2/2)	**24**(2/2)	**26**(2/2)	**30**(4/2)	**30**(4/2)	**32**(4/2)	**28**(4/2)	**28**(4/2)	**30**(4/2)
NOT	Byte, Word	**16**(1/1)	**16**(1/1)	**18**(2/2)	**22**(3/1)	**22**(3/1)	**24**(3/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)
	Long	**24**(2/2)	**24**(2/2)	**26**(2/2)	**30**(4/2)	**30**(4/2)	**32**(4/2)	**28**(4/2)	**28**(4/2)	**30**(4/2)
TST	Byte, Word	**12**(1/0)	**12**(1/0)	**14**(1/0)	**18**(3/0)	**18**(3/0)	**20**(3/0)	**16**(3/0)	**16**(3/0)	**18**(3/0)
	Long	**18**(2/0)	**18**(2/0)	**20**(2/0)	**24**(4/0)	**24**(4/0)	**26**(4/0)	**20**(4/0)	**20**(4/0)	**22**(4/0)

9.6 SHIFT/ROTATE INSTRUCTION EXECUTION TIMES

Tables 9-12 and 9-13 list the timing data for the shift and rotate 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 (+).

Table 9-12. Shift/Rotate Instruction Execution Times

Instruction	Size	Register	Memory*
ASR, ASL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
LSR, LSL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
ROR, ROL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
ROXR, ROXL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—

+Add effective address calculation time.
n is the shift or rotate count.
* Word only.

10.3 POWER CONSIDERATIONS

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

$$T_J = T_A + (P_D \bullet {}^{\theta}J_A) \quad (1)$$

where:

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

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

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

$$P_D = K \div (T_J + 273\ °C) \quad (2)$$

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

$$K = P_D \bullet (T_A + 273°C) + {}^{\theta}JA \bullet P_D{}^2 \quad (3)$$

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

The curve shown in Figure 10-1 gives the graphic solution to the above equations for the specified power dissipation of 1.5 W over the ambient temperature range of -55 °C to 125 °C using a maximum ${}^{\theta}J_A$ of 45 °C/W. Ambient temperature is that of the still air surrounding the device. Lower values of ${}^{\theta}J_A$ cause the curve to shift downward slightly; for instance, for ${}^{\theta}J_A$ of 40 °/W, the curve is just below 1.4 W at 25 °C.

The total thermal resistance of a package (${}^{\theta}J_A$) can be separated into two components, ${}^{\theta}J_C$ and ${}^{\theta}C_A$, representing the barrier to heat flow from the semiconductor junction to the package (case) surface (${}^{\theta}J_C$) and from the case to the outside ambient air (${}^{\theta}C_A$). These terms are related by the equation:

$${}^{\theta}J_A = {}^{\theta}J_C + {}^{\theta}C_A \quad (4)$$

${}^{\theta}J_C$ is device related and cannot be influenced by the user. However, ${}^{\theta}C_A$ is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling, and thermal convection. Thus, good thermal management on the part of the user can significantly reduce ${}^{\theta}C_A$ so that ${}^{\theta}J_A$ approximately equals ;${}^{\theta}J_C$. Substitution of ${}^{\theta}J_C$ for ${}^{\theta}J_A$ in equation 1 results in a lower semiconductor junction temperature.

Table 10-1. Power Dissipation and Junction Temperature vs Temperature ($\theta_{JC}=\theta_{JA}$)

Package	T_A Range	θ_{JC} (°C/W)	P_D (W) @ T_A Min.	T_J (°C) @ T_A Min.	P_D (W) @ T_A Max.	T_J (°C) @ T_A Max.
L/LC	0°C to 70°C	15	1.5	23	1.2	88
	-40°C to 85°C	15	1.7	-14	1.2	103
	0°C to 85°C	15	1.5	23	1.2	103
P	0°C to 70°C	15	1.5	23	1.2	88
R/RC	0°C to 70°C	15	1.5	23	1.2	88
	-40°C to 85°C	15	1.7	-14	1.2	103
	0°C to 85°C	15	1.5	23	1.2	103
FN	0°C to 70°C	25	1.5	38	1.2	101

NOTE: Table does not include values for the MC68000 12F.
Does not apply to the MC68HC000, MC68HC001, and MC68EC000.

Table 10-2. Power Dissipation and Junction Temperature vs Temperature ($\theta_{JC}\neq\theta_{JC}$)

Package	T_A Range	θ_{JA} (°C/W)	P_D (W) @ T_A Min.	T_J (°C) @ T_A Min.	P_D (W) @ T_A Max.	T_J (°C) @ T_A Max.
L/LC	0°C to 70°C	30	1.5	23	1.2	88
	-40°C to 85°C	30	1.7	-14	1.2	103
	0°C to 85°C	30	1.5	23	1.2	103
P	0°C to 70°C	30	1.5	23	1.2	88
R/RC	0°C to 70°C	33	1.5	23	1.2	88
	-40°C to 85°C	33	1.7	-14	1.2	103
	0°C to 85°C	33	1.5	23	1.2	103
FN	0°C to 70°C	40	1.5	38	1.2	101

NOTE: Table does not include values for the MC68000 12F.
Does not apply to the MC68HC000, MC68HC001, and MC68EC000.

Values for thermal resistance presented in this manual, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843 “Thermal Resistance Measurement Method for MC68XXX Microcomponent Devices”’ and are provided for design purposes only. Thermal measurements are complex and dependent on procedure and setup. User-derived values for thermal resistance may differ.

10.4 CMOS CONSIDERATIONS

The MC68HC000, MC68HC001, and MC68EC000, with it significantly lower power consumption, has other considerations. The CMOS cell is basically composed of two complementary transistors (a P channel and an N channel), and only one transistor is turned on while the cell is in the steady state. The active P-channel transistor sources current when the output is a logic high and presents a high impedance when the output is logic low. Thus, the overall result is extremely low power consumption because no power

is lost through the active P-channel transistor. Also, since only one transistor is turned on during the steady state, power consumption is determined by leakage currents.

Because the basic CMOS cell is composed of two complementary transistors, a virtual semiconductor controlled rectifier (SCR) may be formed when an input exceeds the supply voltage. The SCR that is formed by this high input causes the device to become latched in a mode that may result in excessive current drain and eventual destruction of the device. Although the MC68HC000 and MC68EC000 is implemented with input protection diodes, care should be exercised to ensure that the maximum input voltage specification is not exceeded. Some systems may require that the CMOS circuitry be isolated from voltage transients; other may require additional circuitry.

The MC68HC000 and MC68EC000, implemented in CMOS, is applicable to designs to which the following considerations are relevant:

1. The MC68HC000 and MC68EC000 completely satisfies the input/output drive requirements of CMOS logic devices.
2. The HCMOS MC68HC000 and MC68EC000 provides an order of magnitude reduction in power dissipation when compared to the HMOS MC68000. However, the MC68HC000 does not offer a "power-down" mode.

10.5 AC ELECTRICAL SPECIFICATION DEFINITIONS

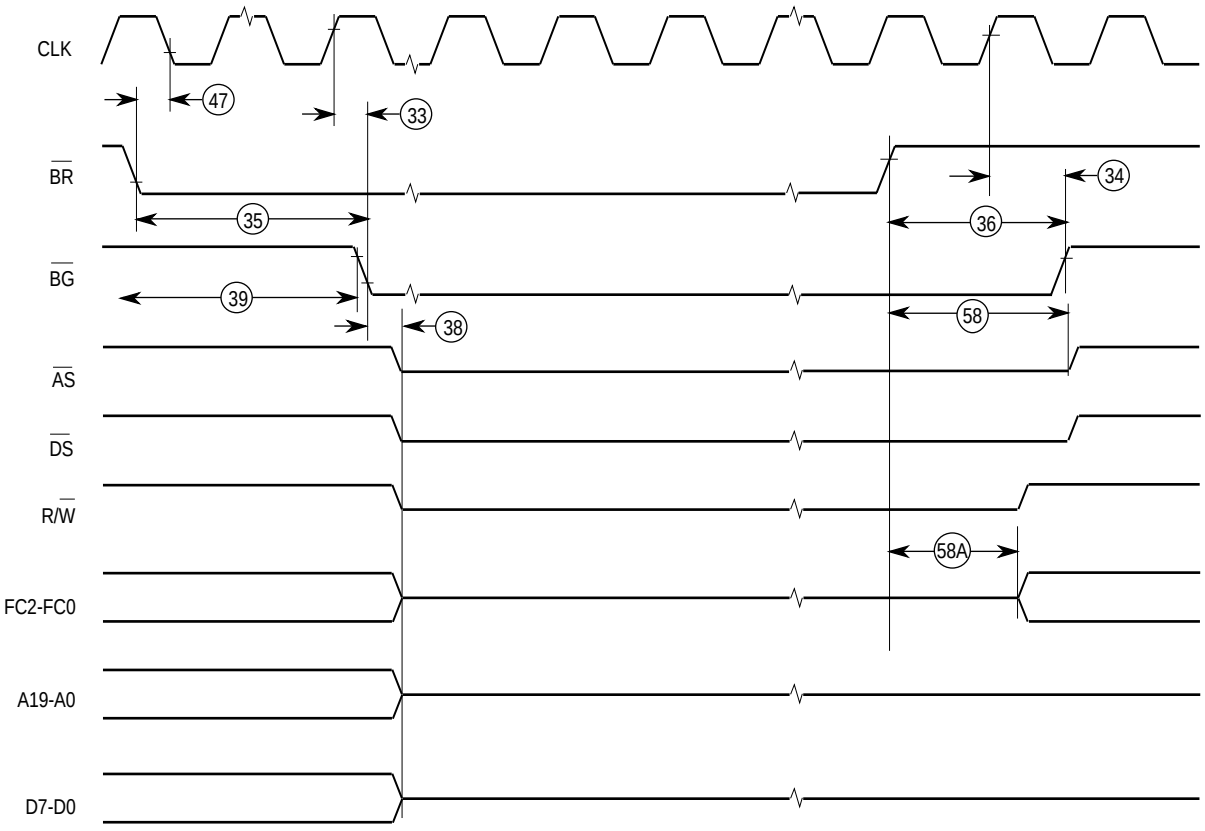
The AC specifications presented consist of output delays, input setup and hold times, and signal skew times. All signals are specified relative to an appropriate edge of the clock and possibly to one or more other signals.

The measurement of the AC specifications is defined by the waveforms shown in Figure 10-2. To test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in the figure. Outputs are specified with minimum and/or maximum limits, as appropriate, and are measured as shown. Inputs are specified with minimum setup and hold times, and are measured as shown. Finally, the measurement for signal-to-signal specifications are shown.

NOTE

The testing levels used to verify conformance to the AC specifications does not affect the guaranteed DC operation of the device as specified in the DC electrical characteristics.

Num	Characteristic	8 MHz*		10 MHz*		12.5 MHz*		16.67 MHz 12F		16 MHz		20 MHz*		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
26[2]	Data-Out Valid to $\overline{DS}$ Asserted (Write)	40	—	30	—	20	—	15	—	15	—	10	—	ns
27[5]	Data-In Valid to Clock Low (Setup Time on Read)	10	—	10	—	10	—	7	—	5	—	5	—	ns
27A[5]	Late $\overline{BERR}$ Asserted to Clock Low (setup Time)	45	—	45	—	45	—	—	—	—	—	—	—	ns
28[2]	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DTACK}$ Negated (Asynchronous Hold)	0	240[1] [1]	0	190	0	150	0	110	0	110	0	95	ns
28A	$\overline{AS}$, $\overline{DS}$ Negated to Data-In High Impedance	—	187	—	150	—	120	—	110	—	110	—	95	ns
29	$\overline{AS}$, $\overline{DS}$ Negated to Data-In Invalid (Hold Time on Read)	0	—	0	—	0	—	0	—	0	—	0	—	ns
29A	$\overline{AS}$, $\overline{DS}$ Negated to Data-In High Impedance	—	187	—	150	—	120	—	90	—	90	—	75	ns
30	$\overline{AS}$, $\overline{DS}$) Negated to $\overline{BERR}$ Negated	0	—	0	—	0	—	0	—	0	—	0	—	ns
31[2,5]	$\overline{DTACK}$ Asserted to Data-In Valid (Setup Time)	—	90	—	65	—	50	—	40	—	50	—	42	ns
32	$\overline{HALT}$) and $\overline{RESET}$ Input Transition Time	0	200	0	200	0	200	0	150	—	150	0	150	ns
33	Clock High to $\overline{BG}$ Asserted	—	62	—	50	—	40	—	40	0	30	0	25	ns
34	Clock High to $\overline{BG}$ Negated	—	62	—	50	—	40	—	40	0	30	0	25	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
36[7]	$\overline{BR}$ Negated to$\overline{BG}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37A[8]	$\overline{BGACK}$ Asserted to $\overline{BR}$ Negated	20	1.5 Clks	20	1.5 Clks	20	1.5 Clks	10	1.5 Clks	10	1.5 Clks	10	1.5 Clks	ns
38	$\overline{BG}$ Asserted to Control, Address, Data Bus High Impedance ($\overline{AS}$ Negated)	—	80	—	70	—	60	—	50	—	50	—	42	ns
39	$\overline{BG}$ Width Negated	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
40	Clock Low to $\overline{VMA}$ Asserted	—	70	—	70	—	70	—	50	—	50	—	40	ns
41	Clock Low to E Transition	—	55[12]	—	45	—	35	—	35	—	35	—	30	ns
42	E Output Rise and Fall Time	—	15	—	15	—	15	—	15	—	15	—	12	ns
43	$\overline{VMA}$ Asserted to E High	200	—	150	—	90	—	80	—	80	—	60	—	ns
44	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{VPA}$ Negated	0	120	0	90	0	70	0	50	0	50	0	42	ns
45	E Low to Control, Address Bus Invalid (Address Hold Time)	30	—	10	—	10	—	10	—	10	—	10	—	ns
46	$\overline{BGACK}$ Width Low	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	ns



NOTES: Waveform measurements for all inputs and outputs are specified at: logic high 2.0 V, logic low = 0.8 V.

Figure 10-14. MC68EC000 Bus Arbitration Timing Diagram

asserted in S2. If the bus cycle is a write cycle, the read/write (R/$\overline{W}$) signal is driven low (for write) during S2. In state 3 (S3) of a write cycle, the write data is placed on the data bus, and in state 4 (S4), the data strobes are asserted to indicate valid data on the bus. Next, the processor inserts wait states until it recognizes the assertion of $\overline{VPA}$.

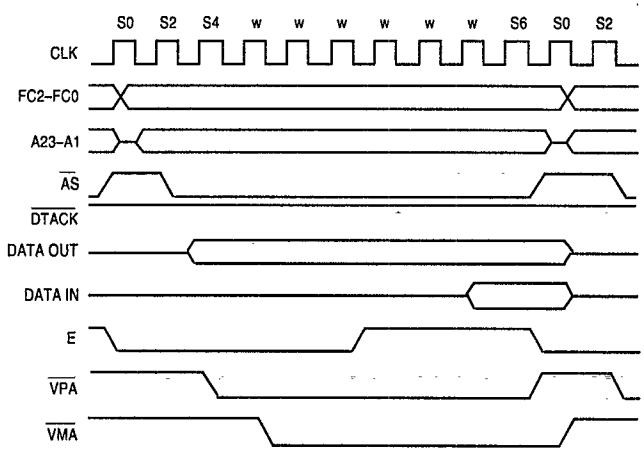


Figure B-4 M6800 Peripheral Timing—Best Case

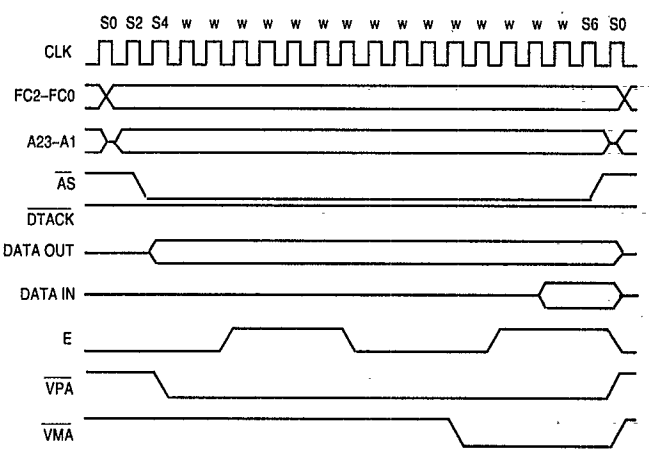


Figure B-5. M6800 Peripheral Timing—Worst Case

The $\overline{VPA}$ input indicates to the processor that the address on the bus is the address of an M6800 device (or an area reserved for M6800 devices) and that the bus should conform to the phase-two transfer characteristics of the M6800 bus. $\overline{VPA}$ is derived by decoding the address bus, conditioned by the address strobe ($\overline{AS}$).

B