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	10MHz
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-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.21x24.21)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc000ei10

SECTION 1
OVERVIEW

This manual includes hardware details and programming information for the MC68000, the MC68HC000, the MC68HC001, the MC68008, the MC68010, and the MC68EC000. For ease of reading, the name M68000 MPUs will be used when referring to all processors. Refer to M68000PM/AD, *M68000 Programmer's Reference Manual*, for detailed information on the MC68000 instruction set.

The six microprocessors are very similar. They all contain the following features

- 16 32-Bit Data and Address Registers
- 16-Mbyte Direct Addressing Range
- Program Counter
- 6 Powerful Instruction Types
- Operations on Five Main Data Types
- Memory-Mapped Input/Output (I/O)
- 14 Addressing Modes

The following processors contain additional features:

- MC68010
  - —Virtual Memory/Machine Support
  - —High-Performance Looping Instructions
- MC68HC001/MC68EC000
  - —Statically Selectable 8- or 16-Bit Data Bus
- MC68HC000/MC68EC000/MC68HC001
  - —Low-Power

All the processors are basically the same with the exception of the MC68008. The MC68008 differs from the others in that the data bus size is eight bits, and the address range is smaller. The MC68010 has a few additional instructions and instructions that operate differently than the corresponding instructions of the other devices.

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

shifted by, rotated by — The source operand is shifted or rotated by the number of positions specified by the second operand

Notation for single-operand operations:

~<operand> — The operand is logically complemented

<operand>sign-extended — The operand is sign-extended, all bits of the upper portion are made equal to the high-order bit of the lower portion

<operand>tested — The operand is compared to zero and the condition codes are set appropriately

Notation for other operations:

TRAP — Equivalent to Format/Offset Word → (SSP); SSP–2 → SSP; PC → (SSP); SSP–4 → SSP; SR → (SSP); SSP–2 → SSP; (vector) → PC

STOP — Enter the stopped state, waiting for interrupts

If <condition> then <operations> else <operations> — The condition is tested. If true, the operations after "then" are performed. If the condition is false and the optional "else" clause is present, the operations after "else" are performed. If the condition is false and else is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.

The breakpoint acknowledge cycle is performed by the MC68010 to provide an indication to hardware that a software breakpoint is being executed when the processor executes a breakpoint (BKPT) instruction. The processor neither accepts nor sends data during this cycle, which is otherwise similar to a read cycle. The cycle is terminated by either $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, or as an M6800 peripheral cycle when $\overline{\text{VPA}}$ is asserted, and the processor continues illegal instruction exception processing. Figure 5-12 illustrates the timing diagram for the breakpoint acknowledge cycle.

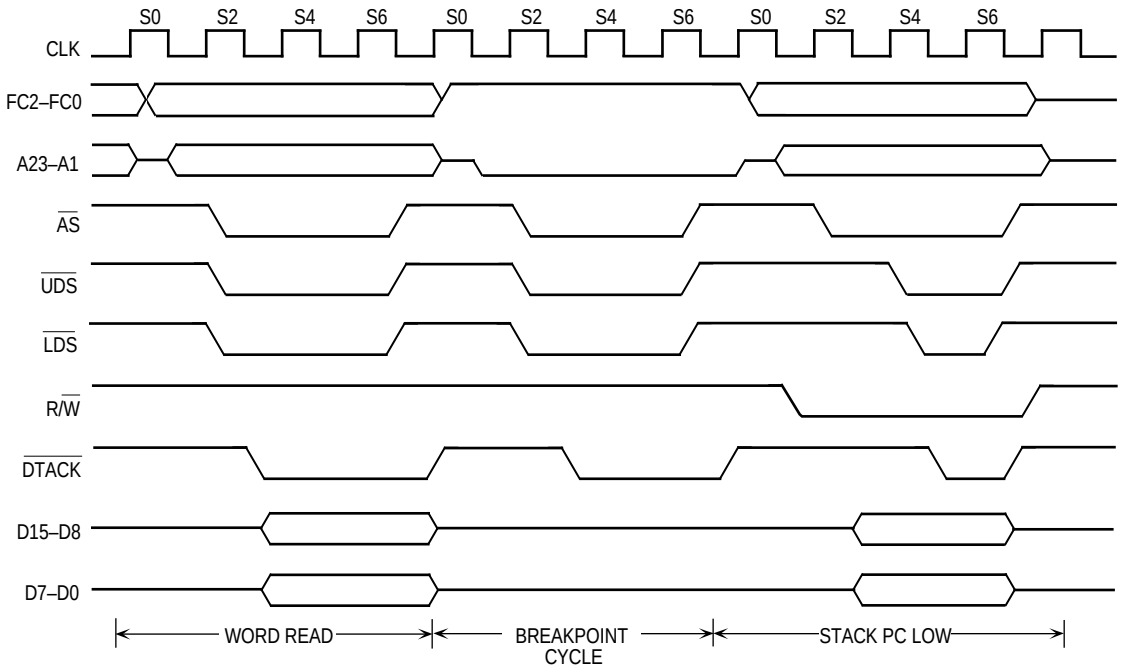


Figure 5-12. Breakpoint Acknowledge Cycle Timing Diagram

5.2 BUS ARBITRATION

Bus arbitration is a technique used by bus master devices to request, to be granted, and to acknowledge bus mastership. Bus arbitration consists of the following:

1. Asserting a bus mastership request
2. Receiving a grant indicating that the bus is available at the end of the current cycle
3. Acknowledging that mastership has been assumed

There are two ways to arbitrate the bus, 3-wire and 2-wire bus arbitration. The MC68000, MC68HC000, MC68EC000, MC68HC001, MC68008, and MC68010 can do 2-wire bus arbitration. The MC68000, MC68HC000, MC68HC001, and MC68010 can do 3-wire bus arbitration. Figures 5-13 and 5-15 show 3-wire bus arbitration and Figures 5-14 and 5-16 show 2-wire bus arbitration. Bus arbitration on all microprocessors, except the 48-pin MC68008 and MC68EC000, $\overline{\text{BGACK}}$ must be pulled high for 2-wire bus arbitration.

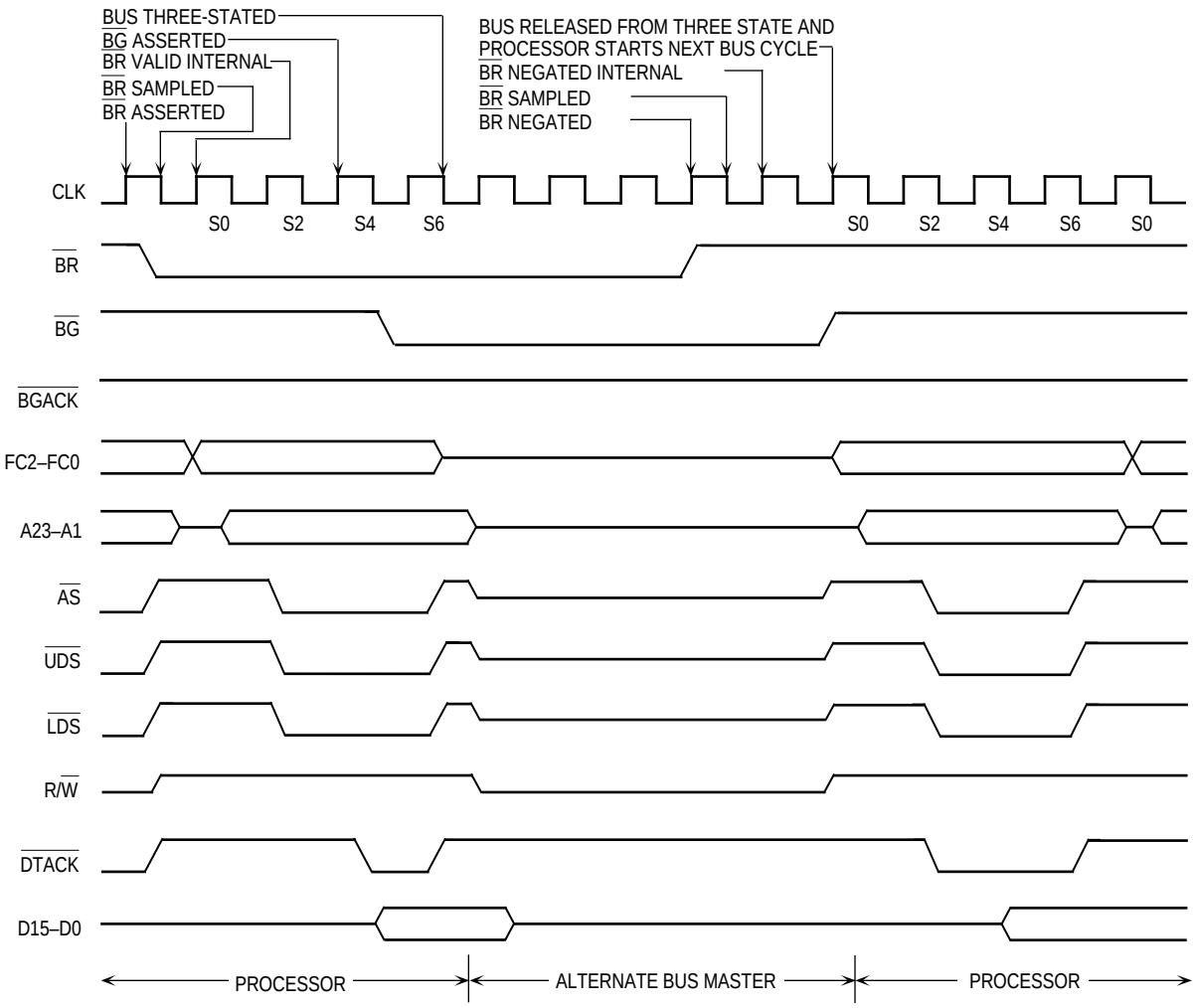


Figure 5-24. 2-Wire Bus Arbitration Timing Diagram—Special Case

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.

After the execution of the instruction is complete and before the start of the next instruction, exception processing for a trace begins. A copy is made of the status register. The transition to supervisor mode is made, and the T bit of the status register is turned off, disabling further tracing. The vector number is generated to reference the trace exception vector, and the current program counter and the copy of the status register are saved on the supervisor stack. On the MC68010, the format/offset word is also saved on the supervisor stack. The saved value of the program counter is the address of the next instruction. Instruction execution commences at the address contained in the trace exception vector.

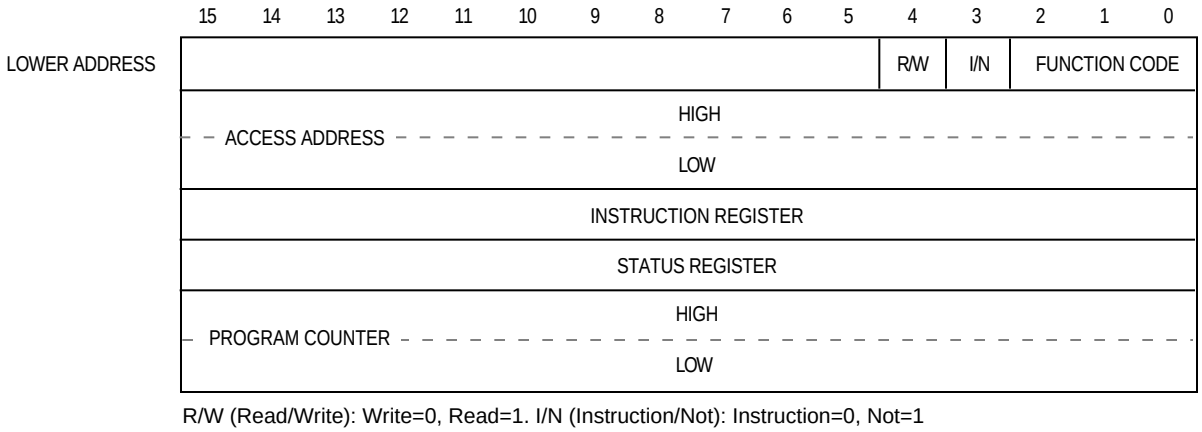
6.3.9 Bus Error

A bus error exception occurs when the external logic requests that a bus error be processed by an exception. The current bus cycle is aborted. The current processor activity, whether instruction or exception processing, is terminated, and the processor immediately begins exception processing. The bus error facility is identical on the all processors; however, the stack frame produced on the MC68010 contains more information. The larger stack frame supports instruction continuation, which supports virtual memory on the MC68010 processor.

6.3.9.1 BUS ERROR. Exception processing for a bus error follows the usual sequence of steps. The status register is copied, the supervisor mode is entered, and tracing is turned off. The vector number is generated to refer to the bus error vector. Since the processor is fetching the instruction or an operand when the error occurs, the context of the processor is more detailed. To save more of this context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are saved. The value saved for the program counter is advanced 2–10 bytes beyond the address of the first word of the instruction that made the reference causing the bus error. If the bus error occurred during the fetch of the next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. In addition to the usual information, the processor saves its internal copy of the first word of the instruction being processed and the address being accessed by the aborted bus cycle. Specific information about the access is also saved: type of access (read or write), processor activity (processing an instruction), and function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a group 2 exception; the processor is not processing an instruction if it is processing a group 0 or a group 1 exception. Figure 6-7 illustrates how this information is organized on the supervisor stack. If a bus error occurs during the last step of exception processing, while either reading the exception vector or fetching the instruction, the value of the program counter is the address of the exception vector. Although this information is not generally sufficient to effect full recovery from the bus error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address in the vector. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution.

If a bus error occurs during the exception processing for a bus error, an address error, or a reset, the processor halts and all processing ceases. This halt simplifies the detection of a catastrophic system failure, since the processor removes itself from the system to

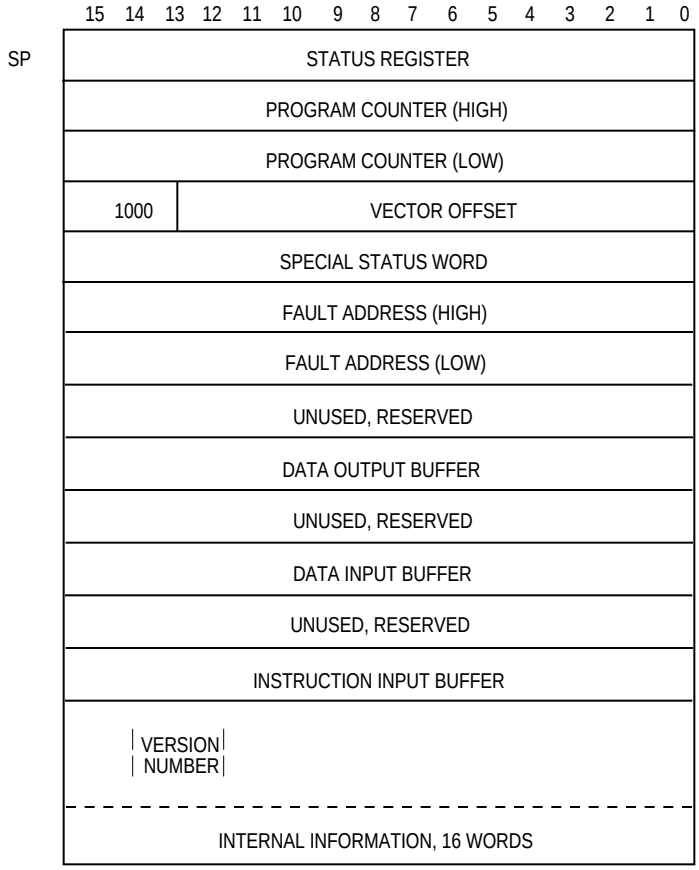
protect memory contents from erroneous accesses. Only an external reset operation can restart a halted processor.



R/W (Read/Write): Write=0, Read=1. I/N (Instruction/Not): Instruction=0, Not=1

Figure 6-7. Supervisor Stack Order for Bus or Address Error Exception

6.3.9.2 BUS ERROR (MC68010). Exception processing for a bus error follows a slightly different sequence than the sequence for group 1 and 2 exceptions. In addition to the four steps executed during exception processing for all other exceptions, 22 words of additional information are placed on the stack. This additional information describes the internal state of the processor at the time of the bus error and is reloaded by the RTE instruction to continue the instruction that caused the error. Figure 6-8 shows the order of the stacked information.



NOTE: The stack pointer is decremented by 29 words, although only 26 words of information are actually written to memory. The three additional words are reserved for future use by Motorola.

Figure 6-8. Exception Stack Order (Bus and Address Error)

The value of the saved program counter does not necessarily point to the instruction that was executing when the bus error occurred, but may be advanced by as many as five words. This incrementing is caused by the prefetch mechanism on the MC68010 that always fetches a new instruction word as each previously fetched instruction word is used. However, enough information is placed on the stack for the bus error exception handler to determine why the bus fault occurred. This additional information includes the address being accessed, the function codes for the access, whether it was a read or a write access, and the internal register included in the transfer. The fault address can be used by an operating system to determine what virtual memory location is needed so that the requested data can be brought into physical memory. The RTE instruction is used to reload the internal state of the processor at the time of the fault. The faulted bus cycle is then rerun, and the suspended instruction is completed. If the faulted bus cycle is a read-modify-write, the entire cycle is rerun, whether the fault occurred during the read or the write operation.

An alternate method of handling a bus error is to complete the faulted access in software. Using this method requires the special status word, the instruction input buffer, the data input buffer, and the data output buffer image. The format of the special status word is

SECTION 7
8-BIT INSTRUCTION EXECUTION TIMES

This section contains listings of the instruction execution times in terms of external clock (CLK) periods for the MC68008 and MC68HC001/MC68EC000 in 8-bit mode. In this data, it is assumed that both memory read and write cycles consist of four clock periods. A longer memory cycle causes the generation of wait states that must be added to the total instruction times.

The number of bus read and write cycles for each instruction is also included with the timing data. This data is shown as

n(r/w)

where:

n is the total number of clock periods
r is the number of read cycles
w is the number of write cycles

For example, a timing number shown as 18(3/1) means that 18 clock periods are required to execute the instruction. Of the 18 clock periods, 12 are used for the three read cycles (four periods per cycle). Four additional clock periods are used for the single write cycle, for a total of 16 clock periods. The bus is idle for two clock periods during which the processor completes the internal operations required for the instruction.

NOTE

The total number of clock periods (n) includes instruction fetch and all applicable operand fetches and stores.

7.1 OPERAND EFFECTIVE ADDRESS CALCULATION TIMES

Table 7-1 lists the numbers of clock periods required to compute the effective addresses for instructions. The totals include fetching any extension words, computing the address, and fetching the memory operand. The total number of clock periods, the number of read cycles, and the number of write cycles (zero for all effective address calculations) are shown in the previously described format.

Table 7-1. Effective Address Calculation Times

Addressing Mode		Byte	Word	Long
	Register			
Dn	Data Register Direct	**0**(0/0)	**0**(0/0)	**0**(0/0)
An	Address Register Direct	**0**(0/0)	**0**(0/0)	**0**(0/0)
	Memory			
(An)	Address Register Indirect	**4**(1/0)	**8**(2/0)	**16**(4/0)
(An)+	Address Register Indirect with Postincrement	**4**(1/0)	**8**(2/0)	**16**(4/0)
–(An)	Address Register Indirect with Predecrement	**6**(1/0)	**10**(2/0)	**18**(4/0)
(d_{16}, An)	Address Register Indirect with Displacement	**12**(3/0)	**16**(4/0)	**24**(6/0)
(d_8, An, Xn)*	Address Register Indirect with Index	**14**(3/0)	**18**(4/0)	**26**(6/0)
(xxx).W	Absolute Short	**12**(3/0)	**16**(4/0)	**24**(6/0)
(xxx).L	Absolute Long	**20**(5/0)	**24**(6/0)	**32**(8/0)
(d_{16}, PC)	Program Counter Indirect with Displacement	**12**(3/0)	**16**(3/0)	**24**(6/0)
(d_8, PC, Xn)*	Program Counter Indirect with Index	**14**(3/0)	**18**(4/0)	**26**(6/0)
#<data>	Immediate	**8**(2/0)	**8**(2/0)	**16**(4/0)

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

7.2 MOVE INSTRUCTION EXECUTION TIMES

Tables 7-2, 7-3, and 7-4 list the numbers of clock periods for the move instructions. The totals include instruction fetch, operand reads, and operand writes. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format.

Table 7-2. Move Byte Instruction Execution Times

	Destination								
Source	**Dn**	**An**	**(An)**	**(An)+**	**–(An)**	**(d_{16}, An)**	**(d_8, An, Xn)***	**(xxx).W**	**(xxx).L**
Dn	**8**(2/0)	**8**(2/0)	**12**(2/1)	**12**(2/1)	**12**(2/1)	**20**(4/1)	**22**(4/1)	**20**(4/1)	**28**(6/1)
An	**8**(2/0)	**8**(2/0)	**12**(2/1)	**12**(2/1)	**12**(2/1)	**20**(4/1)	**22**(4/1)	**20**(4/1)	**28**(6/1)
(An)	**12**(3/0)	**12**(3/0)	**16**(3/1)	**16**(3/1)	**16**(3/1)	**24**(5/1)	**26**(5/1)	**24**(5/1)	**32**(7/1)
(An)+	**12**(3/0)	**12**(3/0)	**16**(3/1)	**16**(3/1)	**16**(3/1)	**24**(5/1)	**26**(5/1)	**24**(5/1)	**32**(7/1)
–(An)	**14**(3/0)	**14**(3/0)	**18**(3/1)	**18**(3/1)	**18**(3/1)	**26**(5/1)	**28**(5/1)	**26**(5/1)	**34**(7/1)
(d_{16}, An)	**20**(5/0)	**20**(5/0)	**24**(5/1)	**24**(5/1)	**24**(5/1)	**32**(7/1)	**34**(7/1)	**32**(7/1)	**40**(9/1)
(d_8, An, Xn)*	**22**(5/0)	**22**(5/0)	**26**(5/1)	**26**(5/1)	**26**(5/1)	**34**(7/1)	**36**(7/1)	**34**(7/1)	**42**(9/1)
(xxx).W	**20**(5/0)	**20**(5/0)	**24**(5/1)	**24**(5/1)	**24**(5/1)	**32**(7/1)	**34**(7/1)	**32**(7/1)	**40**(9/1)
(xxx).L	**28**(7/0)	**28**(7/0)	**32**(7/1)	**32**(7/1)	**32**(7/1)	**40**(9/1)	**42**(9/1)	**40**(9/1)	**48**(11/1)
(d_{16}, PC)	**20**(5/0)	**20**(5/0)	**24**(5/1)	**24**(5/1)	**24**(5/1)	**32**(7/1)	**34**(7/1)	**32**(7/1)	**40**(9/1)
(d_8, PC, Xn)*	**22**(5/0)	**22**(5/0)	**26**(5/1)	**26**(5/1)	**26**(5/1)	**34**(7/1)	**36**(7/1)	**34**(7/1)	**42**(9/1)
#<data>	**16**(4/0)	**16**(4/0)	**20**(4/1)	**20**(4/1)	**20**(4/1)	**28**(6/1)	**30**(6/1)	**28**(6/1)	**36**(8/1)

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

7.7 BIT MANIPULATION INSTRUCTION EXECUTION TIMES

Table 7-9 lists the timing data for the bit manipulation 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 7-9. Bit Manipulation Instruction Execution Times

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte Long	— **12**(2/0)*	**12**(2/1)+ —	— **20**(4/0)*	**20**(4/1)+ —
BCLR	Byte Long	— **14**(2/0)*	**12**(2/1)+ —	— **22**(4/0)*	**20**(4/1)+ —
BSET	Byte Long	— **12**(2/0)*	**12**(2/1)+ —	— **20**(4/0)*	**20**(4/1)+ —
BTST	Byte Long	— **10**(2/0)	**8**(2/0)+	— **18**(4/0)	**16**(4/0)+ —

+Add effective address calculation time.
* Indicates maximum value; data addressing mode only.

7.8 CONDITIONAL INSTRUCTION EXECUTION TIMES

Table 7-10 lists the timing data for the conditional 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 7-10. Conditional Instruction Execution Times

Instruction	Displacement	Trap or Branch Taken	Trap or Branch Not Taken
Bcc	Byte Word	**18**(4/0) **18**(4/0)	**12**(2/0) **20**(4/0)
BRA	Byte Word	**18**(4/0) **18**(4/0)	— —
BSR	Byte Word	**34**(4/4) **34**(4/4)	— —
DBcc	CC True CC False	— **18**(4/0)	**20**(4/0) **26**(6/0)
CHK	—	**68**(8/6)+*	**14**(2/0)
TRAP	—	**62**(8/6)	**—**
TRAPV	—	**66**(10/6)	**8**(2/0)

+Add effective address calculation time for word operand.
* Indicates maximum base value.

8.9 JMP, JSR, LEA, PEA, AND MOVEM INSTRUCTION EXECUTION TIMES

Table 8-10 lists the timing data for the jump (JMP), jump to subroutine (JSR), load effective address (LEA), push effective address (PEA), and move multiple registers (MOVEM) 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.

Table 8-10. JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times

Instruction	Size	(An)	(An)+	–(An)	(d_{16},An)	(d_8,An,Xn)+	(xxx).W	(xxx).L	(d_{16} PC)	(d_8, PC, Xn)*
JMP	—	**8**(2/0)	—	—	**10** (2/0)	**14** (3/0)	**10** (2/0)	**12** (3/0)	**10** (2/0)	**14** (3/0)
JSR	—	**16** (2/2)	—	—	**18** (2/2)	**22** (2/2)	**18** (2/2)	**20** (3/2)	**18** (2/2)	**22** (2/2)
LEA	—	4(1/0)	—	—	**8**(2/0)	**12** (2/0)	**8**(2/0)	**12** (3/0)	**8**(2/0)	**12** (2/0)
PEA	—	**12** (1/2)	—	—	**16** (2/2)	**20** (2/2)	**16** (2/2)	**20** (3/2)	**16** (2/2)	**20** (2/2)
MOVEM M → R	Word	**12+4n** (3+n/0)	**12+4n** (3+n/0)	—	**16+4n** (4+n/0)	**18+4n** (4+n/0)	**16+4n** (4+n/0)	**20+4n** (5+n/0)	**16+4n** (4n/0)	**18+4n** (4+n/0)
	Long	**12+8n** (3+2n/0)	**12+8n** (3+n/0)	—	**16+8n** (4+2n/0)	**18+8n** (4+2n/0)	**16+8n** (4+2n/0)	**20+8n** (5+2n/0)	**16+8n** (4+2n/0)	**18+8n** (4+2n/0)
MOVEM R → M	Word	**8+4n** (2/n)	—	**8+4n** (2/n)	**12+4n** (3/n)	**14+4n** (3/n)	**12+4n** (3/n)	**16+4n** (4/n)	— —	— —
	Long	**8+8n** (2/2n)	— —	**8+8n** (2/2n)	**12+8n** (3/2n)	**14+8n** (3/2n)	**12+8n** (3/2n)	**16+8n** (4/2n)	— —	— —

n is the number of registers to move.
*The size of the index register (Xn) does not affect the instruction's execution time.

8.10 MULTIPRECISION INSTRUCTION EXECUTION TIMES

Table 8-11 lists the timing data for multiprecision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next 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 following notation applies in Table 8-11:

Dn — Data register operand
M — Memory operand

Table 9-4. Move Long 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)	**12**(1/2)	**12**(1/2)	**14**(1/2)	**16**(2/2)	**18**(2/2)	**16**(2/2)	**20**(3/2)
An	**4**(1/0)	**4**(1/0)	**12**(1/2)	**12**(1/2)	**14**(1/2)	**16**(2/2)	**18**(2/2)	**16**(2/2)	**20**(3/2)
(An)	**12**(3/0)	**12**(3/0)	**20**(3/2)	**20**(3/2)	**20**(3/2)	**24**(4/2)	**26**(4/2)	**24**(4/2)	**28**(5/2)
(An)+	**12**(3/0)	**12**(3/0)	**20**(3/2)	**20**(3/2)	**20**(3/2)	**24**(4/2)	**26**(4/2)	**24**(4/2)	**28**(5/2)
–(An)	**14**(3/0)	**14**(3/0)	**22**(3/2)	**22**(3/2)	**22**(3/2)	**26**(4/2)	**28**(4/2)	**26**(4/2)	**30**(5/2)
(d_{16}, An)	**16**(4/0)	**16**(4/0)	**24**(4/2)	**24**(4/2)	**24**(4/2)	**28**(5/2)	**30**(5/2)	**28**(5/2)	**32**(6/2)
(d_8, An, Xn)*	**18**(4/0)	**18**(4/0)	**26**(4/2)	**26**(4/2)	**26**(4/2)	**30**(5/2)	**32**(5/2)	**30**(5/2)	**34**(6/2)
(xxx).W	**16**(4/0)	**16**(4/0)	**24**(4/2)	**24**(4/2)	**24**(4/2)	**28**(5/2)	**30**(5/2)	**28**(5/2)	**32**(6/2)
(xxx).L	**20**(5/0)	**20**(5/0)	**28**(5/2)	**28**(5/2)	**28**(5/2)	**32**(6/2)	**34**(6/2)	**32**(6/2)	**36**(7/2)
(d_{16}, PC)	**16**(4/0)	**16**(4/0)	**24**(4/2)	**24**(4/2)	**24**(4/2)	**28**(5/2)	**30**(5/2)	**28**(5/2)	**32**(5/2)
(d_8, PC, Xn)*	**18**(4/0)	**18**(4/0)	**26**(4/2)	**26**(4/2)	**26**(4/2)	**30**(5/2)	**32**(5/2)	**30**(5/2)	**34**(6/2)
#<data>	**12**(3/0)	**12**(3/0)	**20**(3/2)	**20**(3/2)	**20**(3/2)	**24**(4/2)	**26**(4/2)	**24**(4/2)	**28**(5/2)

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

Table 9-5. Move Long 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	**14**(0/2)	**14**(0/2)	—	**20**(2/2)	**20**(2/2)	—	**18**(2/2)	**18**(2/2)	—
An	**14**(0/2)	**14**(0/2)	—	**20**(2/2)	**20**(2/2)	—	**18**(2/2)	**18**(2/2)	—
(An)	**22**(2/2)	**22**(2/2)	**24**(2/2)	**28**(4/2)	**28**(4/2)	**30**(4/2)	**24**(4/2)	**24**(4/2)	**26**(4/2)
(An)+	**22**(2/2)	**22**(2/2)	**24**(2/2)	**28**(4/2)	**28**(4/2)	**30**(4/2)	**24**(4/2)	**24**(4/2)	**26**(4/2)
–(An)	**24**(2/2)	**24**(2/2)	**26**(2/2)	**30**(4/2)	**30**(4/2)	**32**(4/2)	**26**(4/2)	**26**(4/2)	**28**(4/2)

9.3 STANDARD INSTRUCTION EXECUTION TIMES

The numbers of clock periods shown in tables 9-6 and 9-7 indicate the times required to perform the operations, store the results, and read the next instruction. 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 9-6 and 9-7, the following notation applies:

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

Table 9-15. Conditional Instruction Execution Times

Instruction	Displacement	Branch Taken	Branch Not Taken
Bcc	Byte	**10**(2/0)	**6**(1/0)
	Word	**10**(2/0)	**10**(2/0)
BRA	Byte	**10**(2/0)	—
	Word	**10**(2/0)	—
BSR	Byte	**18**(2/2)	—
	Word	**18**(2/2)	—
DBcc	cc true	—	**10**(2/0)
	cc false	**10**(2/0)	**16**(3/0)

9.9 JMP, JSR, LEA, PEA, AND MOVEM INSTRUCTION EXECUTION TIMES

Table 9-16 lists the timing data for the jump (JMP), jump to subroutine (JSR), load effective address (LEA), push effective address (PEA), and move multiple registers (MOVEM) 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.

Table 9-16. JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times

Instruction	Size	(An)	(An)+	–(An)	(d_{16},An)	(d_8,An,Xn)+	(xxx) W	(xxx).L	(d_8 PC)	(d_{16}, PC, Xn)*
JMP	—	**8**(2/0)	—	—	**10** (2/0)	**14** (3/0)	**10** (2/0)	**12** (3/0)	**10** (2/0)	**14** (3/0)
JSR	—	**16** (2/2)	—	—	**18** (2/2)	**22** (2/2)	**18** (2/2)	**20** (3/2)	**18** (2/2)	**22** (2/2)
LEA	—	4(1/0)	—	—	**8**(2/0)	**12** (2/0)	**8**(2/0)	**12** (3/0)	**8**(2/0)	**12** (2/0)
PEA	—	**12** (1/2)	—	—	**16** (2/2)	**20** (2/2)	**16** (2/2)	**20** (3/2)	**16** (2/2)	**20** (2/2)
MOVEM M → R	Word	**12+4n** (3+n/0)	**12+4n** (3+n/0)	— —	**16+4n** (4+n/0)	**18+4n** (4+n/0)	**16+4n** (4+n/0)	**20+4n** (5+n/0)	**16+4n** (4+n/0)	**18+4n** (4+n/0)
	Long	**24+8n** (3+2n/0)	**12+8n** (3+2n/0)	— —	**16+8n** (4+2n/0)	**18+8n** (4+2n/0)	**16+8n** (4+2n/0)	**20+8n** (5+2n/0)	**16+8n** (4+2n/0)	**18+8n** (4+2n/0)
MOVEM R → M	Word	**8+4n** (2/n)	— —	**8+4n** (2/n)	**12+4n** (3/n)	**14+4n** (3/n)	**12+4n** (3/n)	**16+4n** (4/n)	— —	— —
	Long	**8+8n** (2/2n)	— —	**8+8n** (2/2n)	**12+8n** (3/2n)	**14+8n** (3/2n)	**12+8n** (3/2n)	**16+8n** (4/2n)	— —	— —
MOVES M → R	Byte/ Word	**18** (3/0)	**20** (3/0)	**20** (3/0)	**20** (4/0)	**24** (4/0)	**20** (4/0)	**24** (5/0)		
	Long	**22** (4/0)	**24** (4/0)	**24** (4/0)	**24** (5/0)	**28** (5/0)	**24** (5/0)	**28** (6/0)		
MOVES R → M	Byte/ Word	**18** (2/1)	**20** (2/1)	**20** (2/1)	**20** (3/1)	**24** (3/1)	**20** (3/1)	**24** (4/1)		
	Long	**22** (2/2)	**24** (2/2)	**24** (2/2)	**24** (3/2)	**28** (3/2)	**24** (3/2)	**28** (4/2)		

n is the number of registers to move.
*The size of the index register (Xn) does not affect the instruction's execution time.

9.10 MULTIPRECISION INSTRUCTION EXECUTION TIMES

Table 9-17 lists the timing data for multiprecision instructions. The numbers of clock periods include the times to fetch both operands, perform the operations, store the results,

and read the next 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 following notation applies in Table 9-17:

Dn — Data register operand
M — Memory operand

Table 9-17. Multiprecision Instruction Execution Times

		Nonlooped		Loop Mode		
				Continued	Terminated	
				Valid Count, cc False	Valid Count, cc True	Expired Count
Instruction	**Size**	**op Dn, Dn**	**op M, M***			
ADDX	Byte, Word	**4**(1/0)	**18**(3/1)	**22**(2/1)	**28**(4/1)	**26**(4/1)
	Long	**6**(1/0)	**30**(5/2)	**32**(4/2)	**38**(6/2)	**36**(6/2)
CMPM	Byte, Word	—	**12**(3/0)	**14**(2/0)	**20**(4/0)	**18**(4/0)
	Long	—	**20**(5/0)	**24**(4/0)	**30**(6/0)	**26**(6/0)
SUBX	Byte, Word	**4**(1/)	**18**(3/1)	**22**(2/1)	**28**(4/1)	**26**(4/1)
	Long	**6**(1/0)	**30**(5/2)	**32**(4/2)	**38**(6/2)	**36**(6/2)
ABCD	Byte	**6**(1/0)	**18**(3/1)	**24**(2/1)	**30**(4/1)	**28**(4/1)
SBCD	Byte	**6**(1/0)	**18**(3/1)	**24**(2/1)	**30**(4/1)	**28**(4/1)

*Source and destination ea are (An)+ for CMPM and –(An) for all others.

9.11 MISCELLANEOUS INSTRUCTION EXECUTION TIMES

Table 9-18 lists the timing data for miscellaneous 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 (+).

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	
47[5]	Asynchronous Input Setup Time	10	—	10	—	10	—	10	—	5	—	5	—	ns
48[2,3]	$\overline{BERR}$ Asserted to $\overline{DTACK}$ Asserted	20	—	20	—	20	—	10	—	10	—	10	—	ns
48[2,3,5]	$\overline{DTACK}$ Asserted to $\overline{BERR}$ Asserted (MC68010 Only)	—	80	—	55	—	35	—	—	—	—	—	—	ns
49[9]	$\overline{AS}$, $\overline{DS}$, Negated to E Low	-70	70	-55	55	-45	45	-35	35	-35	35	–30	30	ns
50	E Width High	450	—	350	—	280	—	220	—	220	—	190	—	ns
51	E Width Low	700	—	550	—	440	—	340	—	340	—	290	—	ns
53	Data-Out Hold from Clock High	0	—	0	—	0	—	0	—	0	—	0	—	ns
54	E Low to Data-Out Invalid	30	—	20	—	15	—	10	—	10	—	5	—	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	30	—	20	—	10	—	0	—	0	—	0	—	ns
56[4]	$\overline{HALT}$ ($\overline{RESET}$ Pulse Width	10	—	10	—	10	—	10	—	10	—	10	—	clks
57	$\overline{BGACK}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
57A	$\overline{BGACK}$ Negated to FC, $\overline{VMA}$ Driven	1	—	1	—	1	—	1	—	1	—	1	—	clks
58[7]	$\overline{BR}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
58A[7]	$\overline{BR}$ Negated to FC, $\overline{AS}$ Driven	1	—	1	—	1	—	1	—	1	—	1	—	clks

*These specifications represent improvement over previously published specifications for the 8-, 10-, and 12.5-MHz MC68000 and are valid only for product bearing date codes of 8827 and later.

** This frequency applies only to MC68HC000 and MC68HC001.

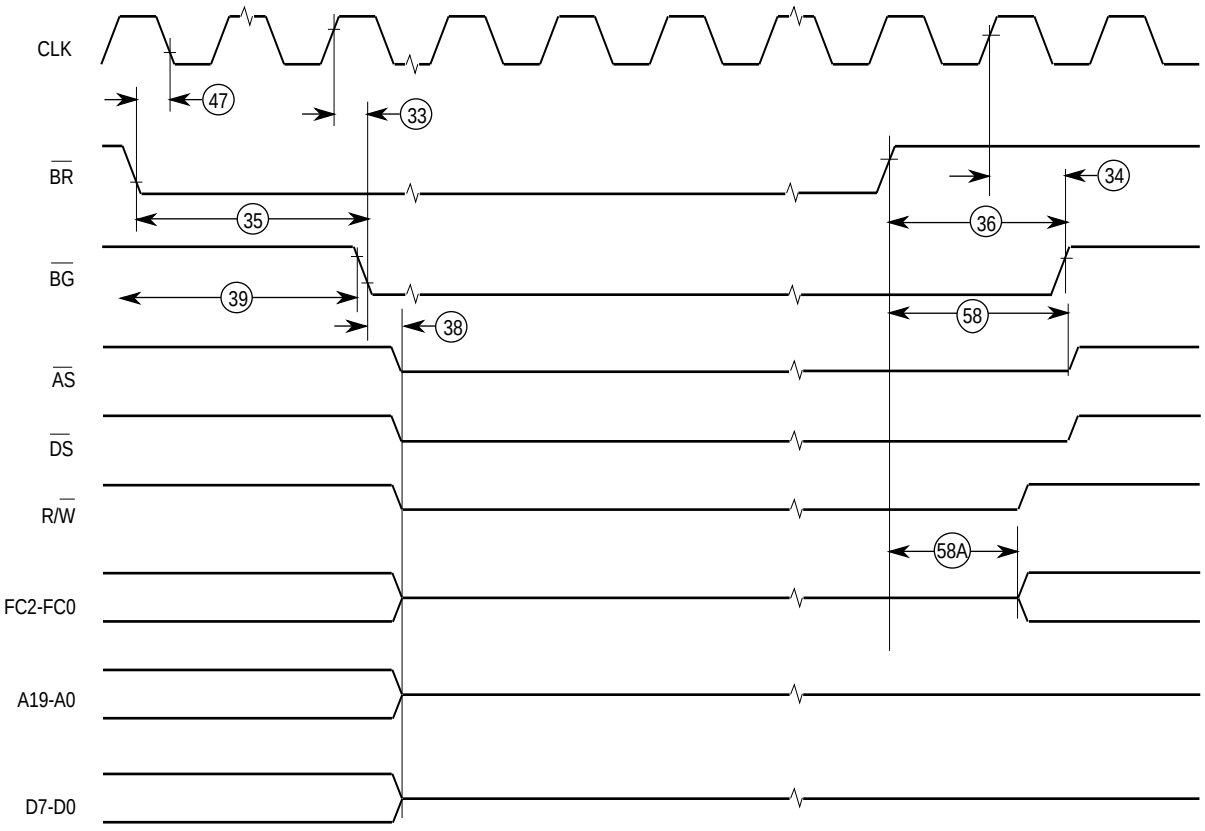
NOTES:

1. For a loading capacitance of less than or equal to 50 pF, subtract 5 ns from the value given in the maximum columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both $\overline{DTACK}$ and $\overline{BERR}$, #48 may be ignored. In the absence of $\overline{DTACK}$, $\overline{BERR}$ is an asynchronous input using the asynchronous input setup time (#47).
4. For power-up, the MC68000 must be held in the reset state for 100 ms to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the processor.
5. If the asynchronous input setup time (#47) requirement is satisfied for $\overline{DTACK}$, the $\overline{DTACK}$ asserted to data setup time (#31) requirement can be ignored. The data must only satisfy the data-in to clock low setup time (#27) for the following clock cycle.
6. When $\overline{AS}$ and R/$\overline{W}$ are equally loaded (±20;pc), subtract 5 ns from the values given in these columns.
7. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
8. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.
9. The falling edge of S6 triggers both the negation of the strobes ($\overline{AS}$ and $\overline{DS}$) and the falling edge of E. Either of these events can occur first, depending upon the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of E.
10. 245 ns for the MC68008.
11. 50 ns for the MC68008
12. 50 ns for the MC68008.

10.15 MC68EC000 AC ELECTRICAL SPECIFICATIONS—BUS ARBITRATION (VCC=5.0VDC ± 5%; GND=0 VDC; $T_A = T_L$ TO T_H; see Figure 10-14)

Num	Characteristic	8 MHz		10 MHz		12.5 MHz		16.67 MHz		20 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
7	Clock High to Address, Data Bus High Impedance (Maximum)	—	55	—	55	—	55	—	50	—	42	ns
16	Clock High to Control Bus High Impedance	—	55	—	55	—	55	—	50	—	42	ns
33	Clock High to $\overline{BG}$ Asserted	—	35	—	35	—	35	0	30	0	25	ns
34	Clock High to $\overline{BG}$ Negated	—	35	—	35	—	35	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	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	Clks
38	$\overline{BG}$ Asserted to Control, Address, Data Bus High Impedance ($\overline{AS}$ Negated)	—	55	—	55	—	55	—	50	—	42	ns
39	$\overline{BG}$ Width Negated	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	Clks
47	Asynchronous Input Setup Time	5	—	5	—	5	—	5	—	5	—	ns
58[1]	$\overline{BR}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	Clks
58A[1]	$\overline{BR}$ Negated to FC Driven	1	—	1	—	1	—	1	—	1	—	Clks

NOTES: 1. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.
2. $\overline{DS}$ is used in this specification to indicate $\overline{UDS}$ and $\overline{LDS}$.



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

1. Fetch the MOVE instruction.
2. Fetch the DBEQ instruction.
3. Read the operand at the address in A0.
4. Write the operand at the address in A1.
5. Fetch the displacement word of the DBEQ instruction.

Of these five bus cycles, only two move the data. However, the MC68010 has a two-word prefetch queue in addition to the one-word instruction decode register. The loop mode uses the prefetch queue and the instruction decode register to eliminate the instruction fetch cycles. The processor places the MOVE instruction in the instruction decode register and the two words of the DBEQ instruction in the prefetch queue. With no additional opcode fetches, the processor executes these two instructions as required to move the entire block or to move all nonzero words that precede a zero.

The MC68010 enters the loop mode automatically when the conditions for loop mode operation are met. Entering the loop mode is transparent to the programmer. The conditions are that the loop count and branch condition of the DBcc instruction must result in looping, the branch displacement must be minus four, and the branch must be to a one-word loop mode instruction preceding the DBcc instruction. The looped instruction and the first word of the DBcc instruction are each fetched twice when the loop is entered. When the processor fetches the looped instruction the second time and determines that the looped instruction is a loop mode instruction, the processor automatically enters the loop mode, and no more instruction fetches occur until the count is exhausted or the loop condition is true.

In addition to the normal termination conditions for the loop, several abnormal conditions cause the MC68010 to exit the loop mode. These abnormal conditions are as follows:

- Interrupts
- Trace Exceptions
- Reset Operations
- Bus Errors

Any pending interrupt is taken after each execution of the DBcc instruction, but not after each execution of the looped instruction. Taking an interrupt exception terminates the loop mode operation; loop mode operation can be restarted on return from the interrupt handler. While the T bit is set, a trace exception occurs at the end of both the looped instruction and the DBcc instruction, making loop mode unavailable while tracing is enabled. A reset operation aborts all processing, including loop mode processing. A bus error during loop mode operation is handled the same as during other processing; however, when the return from exception (RTE) instruction continues execution of the looped instruction, the three-word loop is not fetched again.

Table A-1 lists the loop mode instructions of the MC68010. Only one-word versions of these instructions can operate in the loop mode. One-word instructions use the three address register indirect modes: (An), (An)+, and –(An).

processors. Enable has a 60/40 duty cycle; that is, it is low for six system clocks and high for four system clocks. This duty cycle allows $\overline{VPA}$ accesses on successive E pulses.

In the MC68000, MC68HC000, MC68HC001, and the MC68010, $\overline{VMA}$ is provided to indicate synchronization with E. The MC68008 does not provide a $\overline{VMA}$ signal; external circuitry similar to that shown in Figure B-2 using transistor-to-transistor (TTL) logic must be included in the system to provide $\overline{VMA}$. The $\overline{VMA}$ signal indicates to the M6800 devices that the address on the address bus is a valid device address and that the processor is synchronized to the enable clock. The VPA decode input is an active-high signal that is asserted when address strobe $\overline{AS}$ has been asserted and the address on the address bus is that of a peripheral device. The flip-flop on the left sets at the falling edge of E; the flip-flop on the right sets at the next fall of system clock, asserting $\overline{VMA}$. $\overline{VMA}$ remains asserted until the fall of system clock immediately following the negation of VPA decode. Figure B-3 shows the timing for the $\overline{VMA}$ signal provided by this circuitry.

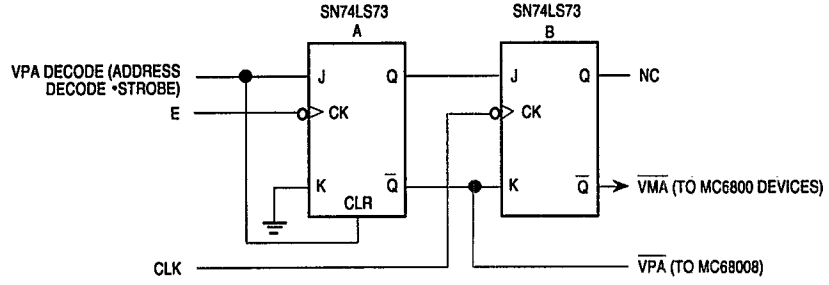


Figure B-2. Example External $\overline{VMA}$ Circuit

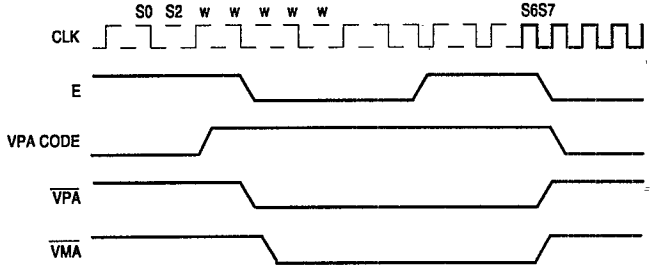


Figure B-3 External $\overline{VMA}$ Timing

B

M6800 cycle timing is shown in Figures B-4 and B-5. At state 0 (S0) in the cycle, the address bus is in the high-impedance state. A function code is asserted on the function code output lines. In state 1 (S1), the address is placed on the address bus. During state 2 (S2), the address strobe ($\overline{AS}$) is asserted to indicate that the address on the bus is valid. If the bus cycle is a read cycle, the upper and/or lower data strobe ($\overline{UDS}$, $\overline{LDS}$) (MC68000/MC68HC000/MC68HC001/MC68010) or data strobe ($\overline{DS}$) (MC68008) is also