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	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/pro/item?MUrl=&PartUrl=mc68hc001ei16

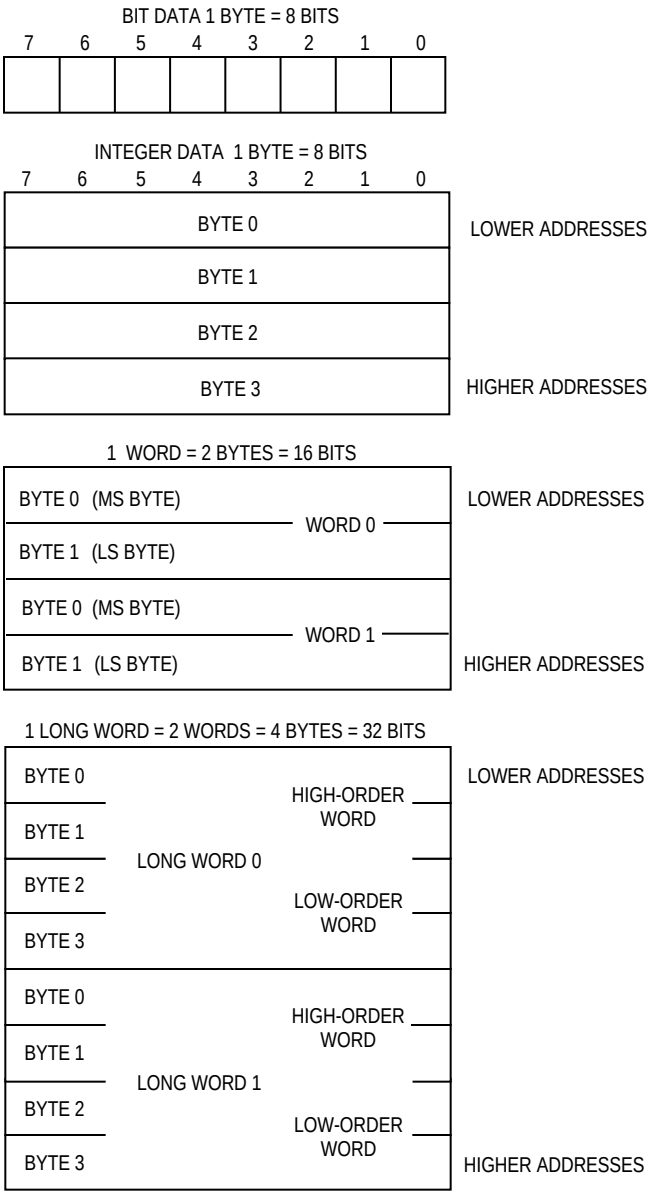


Figure 2-7. Memory Data Organization of the MC68008

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.

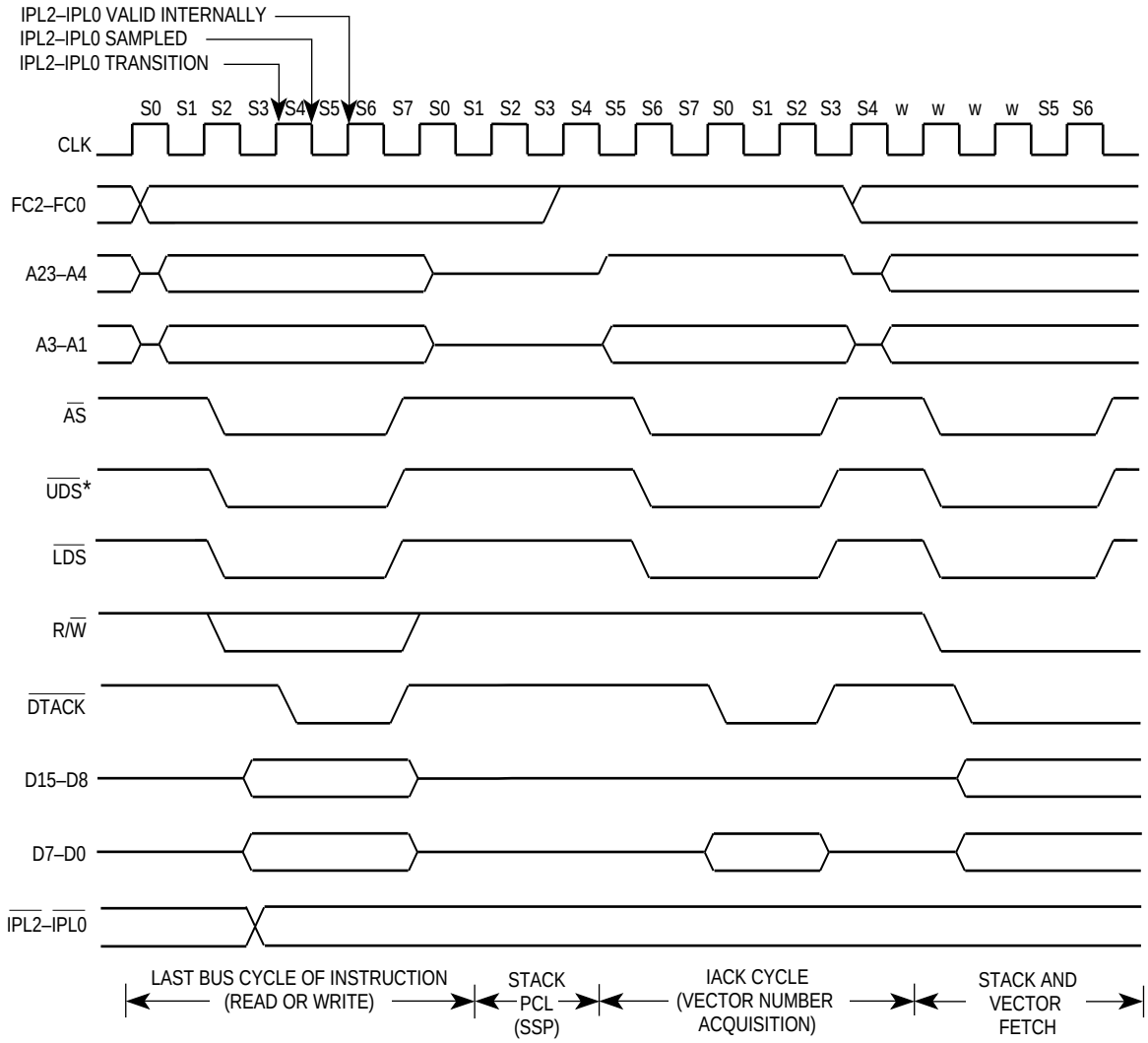
The **MC68008** does not supply a $\overline{\text{VMA}}$ signal. This signal can be produced by a transistor-to-transistor logic (TTL) circuit; an example is described in **Appendix B M6800 Peripheral Interface**.

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.

The interrupt acknowledge cycle places the level of the interrupt being acknowledged on address bits A3–A1 and drives all other address lines high. The interrupt acknowledge cycle reads a vector number when the interrupting device places a vector number on the data bus and asserts $\overline{\text{DTACK}}$ to acknowledge the cycle.

The timing diagram for an interrupt acknowledge cycle is shown in Figure 5-11. Alternately, the interrupt acknowledge cycle can be autovectored. The interrupt acknowledge cycle is the same, except the interrupting device asserts $\overline{\text{VPA}}$ instead of $\overline{\text{DTACK}}$. For an autovectored interrupt, the vector number used is $18 plus the interrupt level. This is generated internally by the microprocessor when $\overline{\text{VPA}}$ (or $\overline{\text{AVEC}}$) is asserted on an interrupt acknowledge cycle. $\overline{\text{DTACK}}$ and $\overline{\text{VPA}}$ ($\overline{\text{AVEC}}$) should never be simultaneously asserted.



*Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D8 through D15 at this time.

Figure 5-11. Interrupt Acknowledge Cycle Timing Diagram

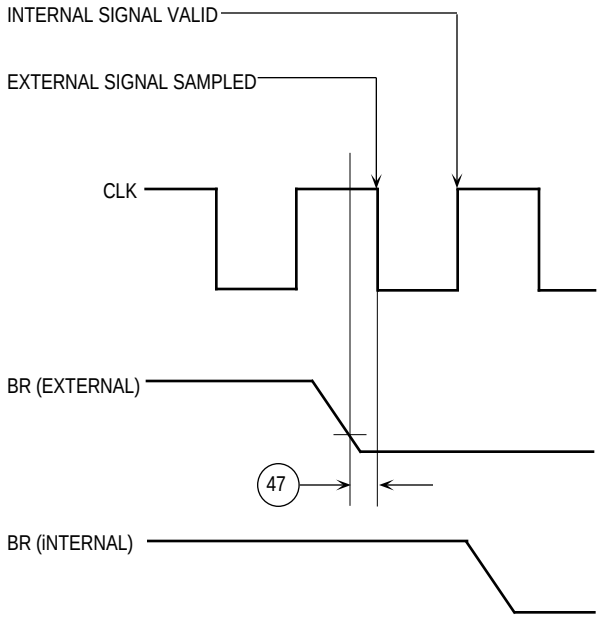


Figure 5-17. External Asynchronous Signal Synchronization

Bus arbitration control is implemented with a finite-state machine. State diagram (a) in Figure 5-18 applies to all processors using 3-wire bus arbitration and state diagram (b) applies to processors using 2-wire bus arbitration, in which $\overline{\text{BGACK}}$ is permanently negated internally or externally. The same finite-state machine is used, but it is effectively a two-state machine because $\overline{\text{BGACK}}$ is always negated.

In Figure 5-18, input signals R and A are the internally synchronized versions of $\overline{\text{BR}}$ and $\overline{\text{BGACK}}$. The $\overline{\text{BG}}$ output is shown as G, and the internal three-state control signal is shown as T. If T is true, the address, data, and control buses are placed in the high-impedance state when $\overline{\text{AS}}$ is negated. All signals are shown in positive logic (active high), regardless of their true active voltage level. State changes (valid outputs) occur on the next rising edge of the clock after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in Figure 5-19. The bus arbitration timing while the bus is inactive (e.g., the processor is performing internal operations for a multiply instruction) is shown in Figure 5-20.

When a bus request is made after the MPU has begun a bus cycle and before $\overline{\text{AS}}$ has been asserted (S0), the special sequence shown in Figure 5-21 applies. Instead of being asserted on the next rising edge of clock, $\overline{\text{BG}}$ is delayed until the second rising edge following its internal assertion.

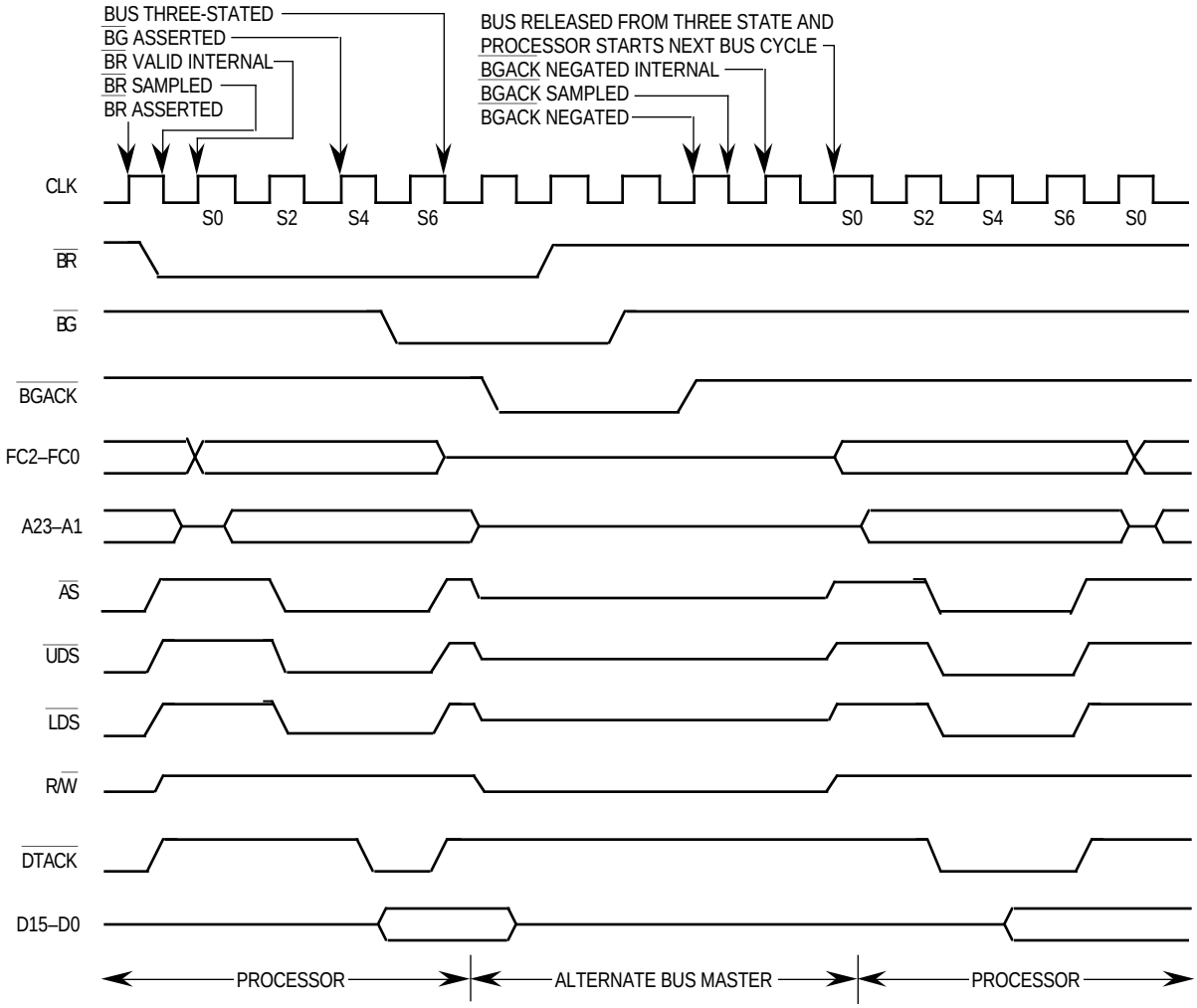


Figure 5-21. 3-Wire Bus Arbitration Timing Diagram—Special Case

The MC68010 also differs from the other microprocessors described in this manual regarding bus errors. The MC68010 can detect a late bus error signal asserted within one clock cycle after the assertion of data transfer acknowledge. When receiving a bus error signal, the processor can either initiate a bus error exception sequence or try running the cycle again.

5.4.1 Bus Error Operation

In all the microprocessors described in this manual, a bus error is recognized when $\overline{\text{DTACK}}$ and $\overline{\text{HALT}}$ are negated and $\overline{\text{BERR}}$ is asserted. In the MC68010, a late bus error is also recognized when $\overline{\text{HALT}}$ is negated, and $\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$ are asserted within one clock cycle.

When the bus error condition is recognized, the current bus cycle is terminated in S9 for a read cycle, a write cycle, or the read portion of a read-modify-write cycle. For the write portion of a read-modify-write cycle, the current bus cycle is terminated in S21. As long as $\overline{\text{BERR}}$ remains asserted, the data and address buses are in the high-impedance state. Figure 5-25 shows the timing for the normal bus error, and Figure 5-26 shows the timing for the MC68010 late bus error.

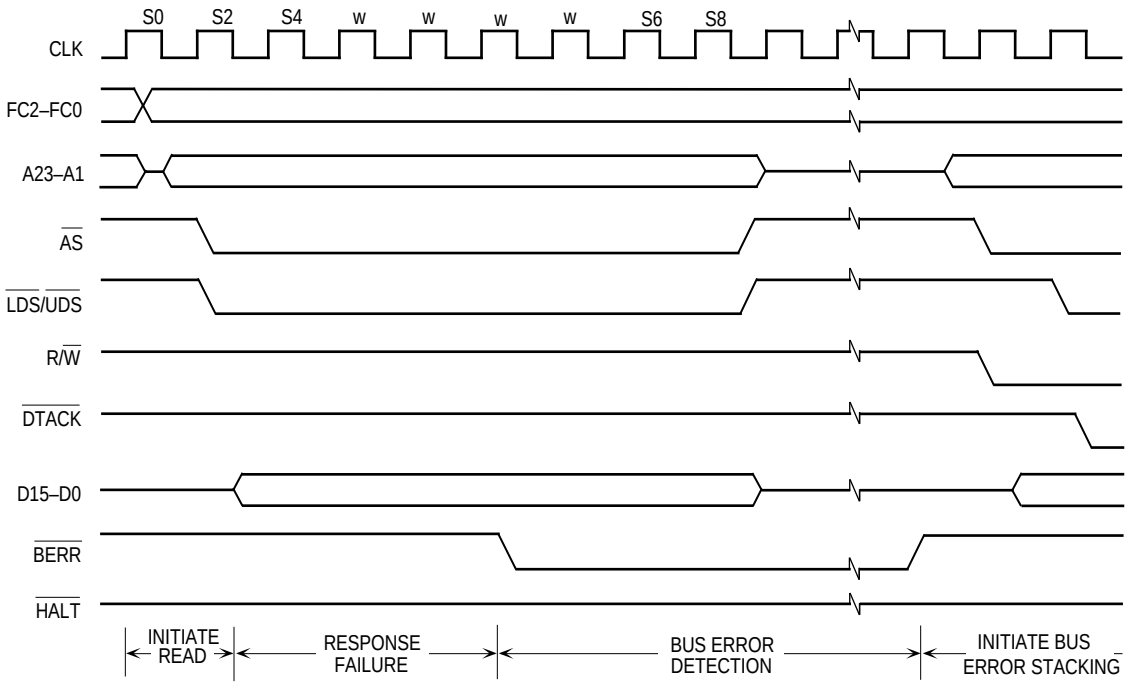


Figure 5-25. Bus Error Timing Diagram

A double bus fault occurs during a reset operation when a bus error occurs while the processor is reading the vector table (before the first instruction is executed). The reset operation is described in the following paragraph.

5.5 RESET OPERATION

$\overline{\text{RESET}}$ is asserted externally for the initial processor reset. Subsequently, the signal can be asserted either externally or internally (executing a RESET instruction). For proper external reset operation, $\overline{\text{HALT}}$ must also be asserted.

When $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are driven by an external device, the entire system, including the processor, is reset. Resetting the processor initializes the internal state. The processor reads the reset vector table entry (address $00000) and loads the contents into the supervisor stack pointer (SSP). Next, the processor loads the contents of address $00004 (vector table entry 1) into the program counter. Then the processor initializes the interrupt level in the status register to a value of seven. In the MC68010, the processor also clears the vector base register to $00000. No other register is affected by the reset sequence. Figure 5-30 shows the timing of the reset operation.

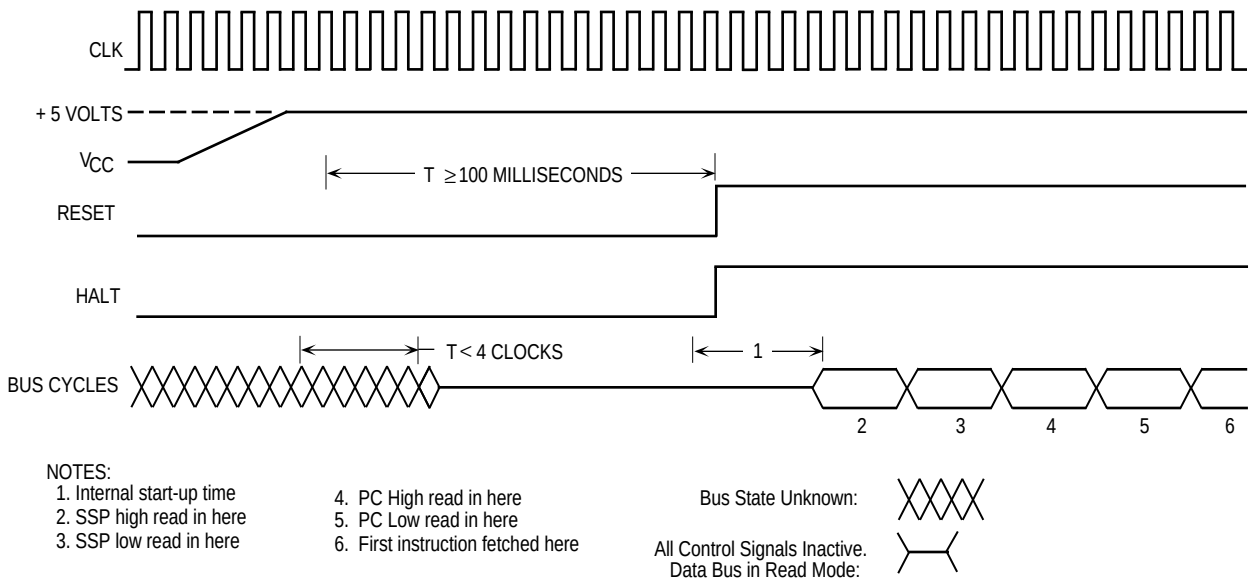


Figure 5-30. Reset Operation Timing Diagram

The RESET instruction causes the processor to assert $\overline{\text{RESET}}$ for 124 clock periods to reset the external devices of the system. The internal state of the processor is not affected. Neither the status register nor any of the internal registers is affected by an internal reset operation. All external devices in the system should be reset at the completion of the RESET instruction.

For the initial reset, $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ must be asserted for at least 100 ms. For a subsequent external reset, asserting these signals for 10 clock cycles or longer resets the processor. However, an external reset signal that is asserted while the processor is

Table 5-1. $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ Assertion Results

Case No.	Control Signal	Asserted on Rising Edge of State		MC68000/MC68HC000/001 EC000/MC68008 Results	MC68010 Results
		N	N+2		
1	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	S NA X	Normal cycle terminate and continue.	Normal cycle terminate and continue.
2	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA A/S	S NA S	Normal cycle terminate and halt. Continue when $\overline{\text{HALT}}$ negated.	Normal cycle terminate and halt. Continue when $\overline{\text{HALT}}$ negated.
3	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	X A NA	X S NA	Terminate and take bus error trap.	Terminate and take bus error trap.
4	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	S A NA	Normal cycle terminate and continue.	Terminate and take bus error trap.
5	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	X A A/S	X S S	Terminate and retry when $\overline{\text{HALT}}$ removed.	Terminate and retry when $\overline{\text{HALT}}$ removed.
6	$\overline{\text{DTACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	S A A	Normal cycle terminate and continue.	Terminate and retry when $\overline{\text{HALT}}$ removed.

LEGEND:
- N — The number of the current even bus state (e.g., S4, S6, etc.)
- A — Signal asserted in this bus state
- NA — Signal not asserted in this bus state
- X — Don't care
- S — Signal asserted in preceding bus state and remains asserted in this state

NOTE: All operations are subject to relevant setup and hold times.

The negation of $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ under several conditions is shown in Table 5-6. ($\overline{\text{DTACK}}$ is assumed to be negated normally in all cases; for reliable operation, both $\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$ should be negated when address strobe is negated).

EXAMPLE A:

A system uses a watchdog timer to terminate accesses to unused address space. The timer asserts $\overline{\text{BERR}}$ after timeout (case 3).

EXAMPLE B:

A system uses error detection on random-access memory (RAM) contents. The system designer may:

1. Delay $\overline{\text{DTACK}}$ until the data is verified. If data is invalid, return $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ simultaneously to retry the error cycle (case 5).
2. Delay $\overline{\text{DTACK}}$ until the data is verified. If data is invalid, return $\overline{\text{BERR}}$ at the same time as $\overline{\text{DTACK}}$ (case 3).
3. For an MC68010, return $\overline{\text{DTACK}}$ before data verification. If data is invalid, assert $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ to retry the error cycle (case 6).

Table 6-4. MC68010 Format Codes

Format Code	Stacked Information
0000	Short Format (4 Words)
1000	Long Format (29 Words)
All Others	Unassigned, Reserved

6.2.5 Exception Processing Sequence

In the first step of exception processing, an internal copy is made of the status register. After the copy is made, the S bit of the status register is set, putting the processor into the supervisor mode. Also, the T bit is cleared, which allows the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated appropriately.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor bus cycle classified as an interrupt acknowledge cycle. For all other exceptions, internal logic provides the vector number. This vector number is then used to calculate the address of the exception vector.

The third step, except for the reset exception, is to save the current processor status. (The reset exception does not save the context and skips this step.) The current program counter value and the saved copy of the status register are stacked using the SSP. The stacked program counter value usually points to the next unexecuted instruction. However, for bus error and address error, the value stacked for the program counter is unpredictable and may be incremented from the address of the instruction that caused the error. Group 1 and 2 exceptions use a short format exception stack frame (format = 0000 on the MC68010). Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector. The processor then resumes instruction execution. The instruction at the address in the exception vector is fetched, and normal instruction decoding and execution is started.

6.3 PROCESSING OF SPECIFIC EXCEPTIONS

The exceptions are classified according to their sources, and each type is processed differently. The following paragraphs describe in detail the types of exceptions and the processing of each type.

6.3.1 Reset

The reset exception corresponds to the highest exception level. The processing of the reset exception is performed for system initiation and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, and the trace state is forced off. The

interrupt priority mask is set at level 7. In the MC68010, the VBR is forced to zero. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the SSP, neither the program counter nor the status register is saved. The address in the first two words of the reset exception vector is fetched as the initial SSP, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The initial program counter should point to the power-up/restart code.

The RESET instruction does not cause a reset exception; it asserts the $\overline{\text{RESET}}$ signal to reset external devices, which allows the software to reset the system to a known state and continue processing with the next instruction.

6.3.2 Interrupts

Seven levels of interrupt priorities are provided, numbered from 1–7. All seven levels are available except for the 48-pin version for the MC68008.

NOTE

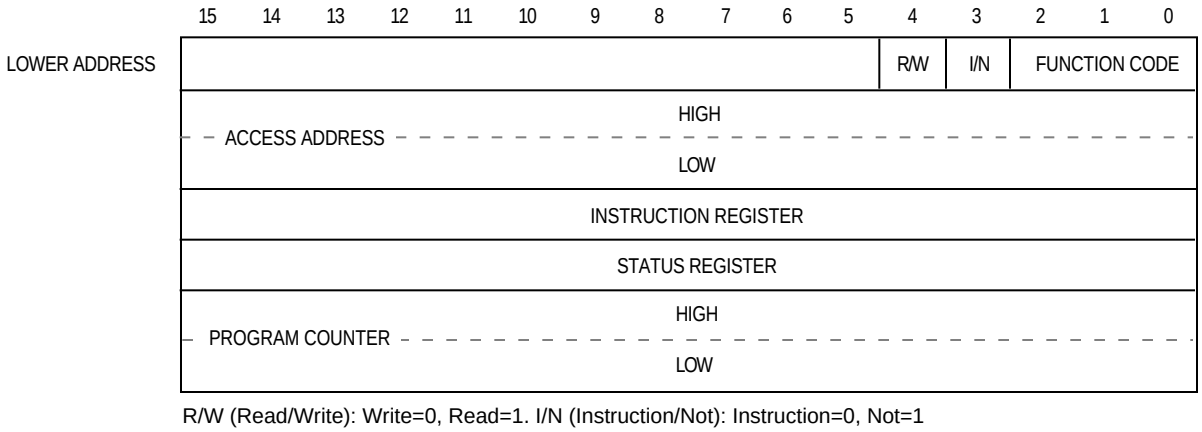
The MC68008 48-pin version supports only three interrupt levels: 2, 5, and 7. Level 7 has the highest priority.

Devices can be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. The status register contains a 3-bit mask indicating the current interrupt priority, and interrupts are inhibited for all priority levels less than or equal to the current priority.

An interrupt request is made to the processor by encoding the interrupt request levels 1–7 on the three interrupt request lines; all lines negated indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but the requests are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction, and the interrupt exception processing is postponed until the priority of the pending interrupt becomes greater than the current processor priority.

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. A copy of the status register is saved; the privilege mode is set to supervisor mode; tracing is suppressed; and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device by executing an interrupt acknowledge cycle, which displays the level number of the interrupt being acknowledged on the address bus. If external logic requests an automatic vector, the processor internally generates a vector number corresponding to the interrupt level number. If external logic indicates a bus error, the interrupt is considered spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the format/offset word (MC68010 only), program counter, and status

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.

Table 7-7. Single Operand Instruction Execution Times

Instruction	Size	Register	Memory
CLR	Byte Word Long	**8**(2/0) **8**(2/0) **10**(2/0)	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+
NBCD	Byte	**10**(2/0)	**12**(2/1)+
NEG	Byte Word Long	**8**(2/0) **8**(2/0) **10**(2/0)	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+
NEGX	Byte Word Long	**8**(2/0) **8**(2/0) **10**(2/0)	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+
NOT	Byte Word Long	**8**(2/0) **8**(2/0) **10**(2/0)	**12**(2/1)+ **16**(2/2)+ **24**(2/4)+
Scc	Byte, False Byte, True	**8**(2/0) **10**(2/0)	**12**(2/1)+ **12**(2/1)+
TAS	Byte	**8**(2/0)	**14**(2/1)+
TST	Byte Word Long	**8**(2/0) **8**(2/0) **8**(2/0)	**8**(2/0)+ **8**(2/0)+ **8**(2/0)+

+Add effective address calculation time.

7.6 SHIFT/ROTATE INSTRUCTION EXECUTION TIMES

Table 7-8 lists 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 7-8. Shift/Rotate Instruction Execution Times

Instruction	Size	Register	Memory
ASR, ASL	Byte Word Long	**10+2n**(2/0) **10+2n**(2/0) **12+n2**(2/0)	— **16**(2/2)+ —
LSR, LSL	Byte Word Long	**10+2n**(2/0) **10+2n**(2/0) **12+n2**(2/0)	— **16**(2/2)+ —
ROR, ROL	Byte Word Long	**10+2n**(2/0) **10+2n**(2/0) **12+n2**(2/0)	— **16**(2/2)+ —
ROXR, ROXL	Byte Word Long	**10+2n**(2/0) **10+2n**(2/0) **12+n2**(2/0)	— **16**(2/2)+ —

+Add effective address calculation time for word operands.
n is the shift count.

8.7 BIT MANIPULATION INSTRUCTION EXECUTION TIMES

Table 8-8 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 8-8. Bit Manipulation Instruction Execution Times

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte	—	**8**(1/1)+	—	**12**(2/1)+
	Long	**8**(1/0)*	—	**12**(2/0)*	—
BCLR	Byte	—	**8**(1/1)+	—	**12**(2/1)+
	Long	**10**(1/0)*	—	**14**(2/0)*	—
BSET	Byte	—	**8**(1/1)+	—	**12**(2/1)+
	Long	**8**(1/0)*	—	**12**(2/0)*	—
BTST	Byte	—	**4**(1/0)+	—	**8**(2/0)+
	Long	**6**(1/0)	—	**10**(2/0)	—

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

8.8 CONDITIONAL INSTRUCTION EXECUTION TIMES

Table 8-9 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.

Table 8-9. Conditional Instruction Execution Times

Instruction	Displacement	Branch Taken	Branch Not Taken
Bcc	Byte	**10**(2/0)	**8**(1/0)
	Word	**10**(2/0)	**12**(2/0)
BRA	Byte	**10**(2/0)	—
	Word	**10**(2/0)	—
BSR	Byte	**18**(2/2)	—
	Word	**18**(2/2)	—
DBcc	cc true	—	**12**(2/0)
	cc false, Count Not Expired	**10**(2/0)	—
	cc false, Counter Expired	—	**14**(3/0)

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.

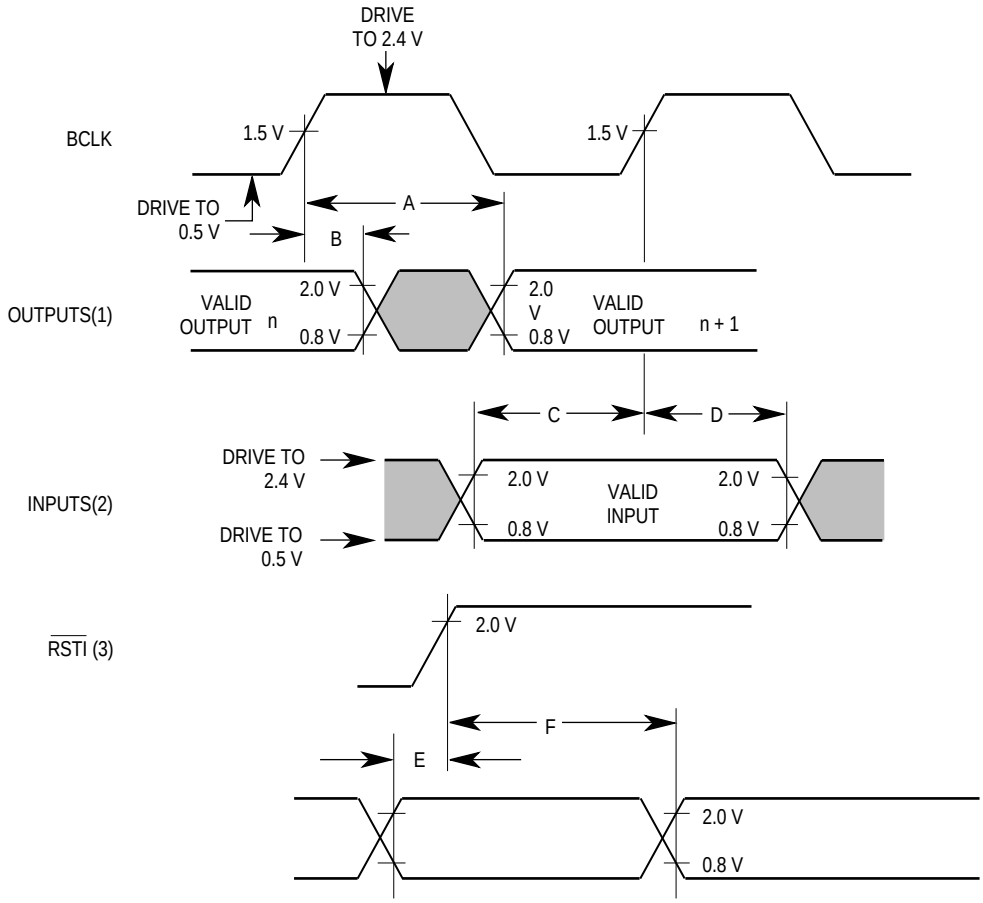
Table 9-16. JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times

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,



NOTES:

1. This output timing is applicable to all parameters specified relative to the rising edge of the clock.
2. This input timing is applicable to all parameters specified relative to the rising edge of the clock.
3. This timing is applicable to all parameters specified relative to the negation of the RESET signal.

LEGEND:

A. Maximum output delay specification.
B. Minimum output hold time.
C. Minimum input setup time specification.
D. Minimum input hold time specification.
E. Mode select setup time to RESET negated.
F. Mode select hold time from RESET negated.

Figure 10-2. Drive Levels and Test Points for AC Specifications

10.10 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES

(V_{CC}=5.0 VDC±5%; GND=0 V; T_A=T_L to T_H; (see Figures 10-4 and 10-5) (Applies To All Processors Except The MC68EC000)

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	
6	Clock Low to Address Valid	—	62	—	50	—	50	—	50		30	—	25	ns
6A	Clock High to FC Valid	—	62	—	50	—	45	—	45	0	30	0	25	ns
7	Clock High to Address, Data Bus High Impedance (Maximum)	—	80	—	70	—	60	—	50		50	—	42	ns
8	Clock High to Address, FC Invalid (Minimum)	0	—	0	—	0	—	0	—	0	—	0	—	ns
9[1]	Clock High to $\overline{AS}$, $\overline{DS}$ Asserted	3	60	3	50	3	40	3	40	3	30	3	25	ns
11[2]	Address Valid to $\overline{AS}$, $\overline{DS}$ Asserted (Read)/AS Asserted (Write)	30	—	20	—	15	—	15	—	15	—	10	—	ns
11A[2]	FC Valid to $\overline{AS}$), $\overline{DS}$ Asserted (Read)/$\overline{AS}$) Asserted (Write)	90	—	70	—	60	—	30	—	45	—	40	—	ns
12[1]	Clock Low to $\overline{AS}$, $\overline{DS}$ Negated	—	62	—	50	—	40	—	40	3	30	3	25	ns
13[2]	$\overline{AS}$, $\overline{DS}$ Negated to Address, FC Invalid	40	—	30	—	20	—	10	—	15	—	10	—	ns
14[2]	$\overline{AS}$and $\overline{DS}$ Read) Width Asserted	270	—	195	—	160	—	120	—	120	—	100	—	ns
14A	$\overline{DS}$ Width Asserted (Write)	140		95		80		60		60	—	50	—	ns
15[2]	$\overline{AS}$, $\overline{DS}$ Width Negated	150	—	105	—	65	—	60	—	60	—	50	—	ns
16	Clock High to Control Bus High Impedance	—	80	—	70	—	60	—	50	—	50	—	42	ns
17[2]	$\overline{AS}$, $\overline{DS}$ Negated to R/$\overline{W}$ Invalid	40	—	30	—	20	—	10	—	15	—	10	—	ns
18[1]	Clock High to R/$\overline{W}$ High (Read)	0	55	0	45	0	40	0	40	0	30	0	25	ns
20[1]	Clock High to R/$\overline{W}$ Low (Write)	0	55	0	45	0	40	0	40	0	30	0	25	ns
20A[2,6]	$\overline{AS}$ Asserted to R/$\overline{W}$ Valid (Write)	—	10	—	10	—	10	—	10	—	10	—	10	ns
21[2]	Address Valid to R/$\overline{W}$ Low (Write)	20	—	0	—	0	—	0	—	0	—	0	—	ns
21A[2]	FC Valid to R/$\overline{W}$ Low (Write)	60	—	50	—	30	—	20	—	30	—	25	—	ns
22[2]	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write)	80	—	50	—	30	—	20	—	30	—	25	—	ns
23	Clock Low to Data-Out Valid (Write)	—	62	—	50	—	50	—	550	—	30	—	25	ns
25[2]	$\overline{AS}$, $\overline{DS}$) Negated to Data-Out Invalid (Write)	40[10]	—	30	—	20	—	15	—	15	—	10	—	ns

10.11 AC ELECTRICAL SPECIFICATIONS—MC68000 TO M6800 PERIPHERAL (V_{CC} = 5.0 Vdc ±5%; GND=0 Vdc; $T_A = T_L$ TO T_H; refer to figures 10-6)

(Applies To All Processors Except The MC68EC000)

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	
12[1]	Clock Low to $\overline{AS}$, $\overline{DS}$ Negated	—	62	—	50	—	40	—	40	3	30	3	25	ns
18[1]	Clock High to R/$\overline{W}$ High (Read)	0	55	0	45	0	40	0	40	0	30	0	25	ns
20[1]	Clock High to R/$\overline{W}$ Low (Write)	0	55	0	45	0	40	0	40	0	30	0	25	ns
23	Clock Low to Data-Out Valid (Write)	—	62	—	50	—	50	—	50	—	30	—	25	ns
27	Data-In Valid to Clock Low (Setup Time on Read)	10	—	10	—	10	—	7	—	5	—	5	—	ns
29	$\overline{AS}$, $\overline{DS}$ Negated to Data-In Invalid (Hold Time on Read)	0	—	0	—	0	—	0	—	0	—	0	—	ns
40	Clock Low to $\overline{VMA}$ Asserted	—	70	—	70	—	70	—	50	—	50	—	40	ns
41	Clock Low to E Transition	—	55	—	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
47	Asynchronous Input Setup Time	10	—	10	—	10	—	10	—	10	—	5	—	ns
49[2]	$\overline{AS}$, $\overline{DS}$, 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
54	E Low to Data-Out Invalid	30	—	20	—	15	—	10	—	10	—	5	—	ns

*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. 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. Specificaton #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.

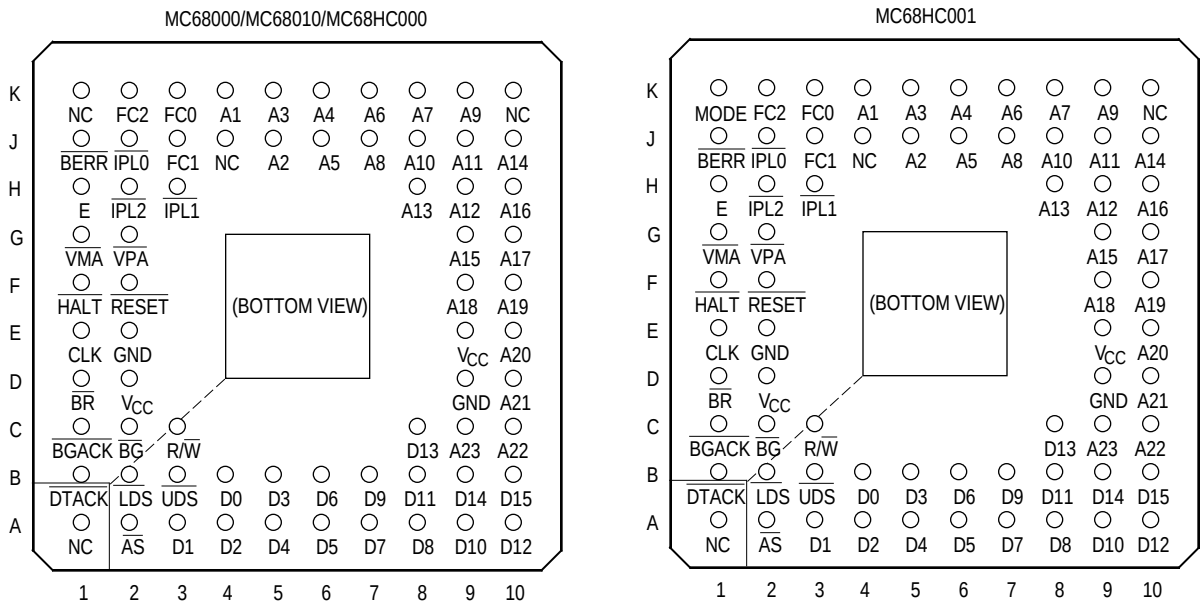


Figure 11-2. 68-Lead Pin Grid Array

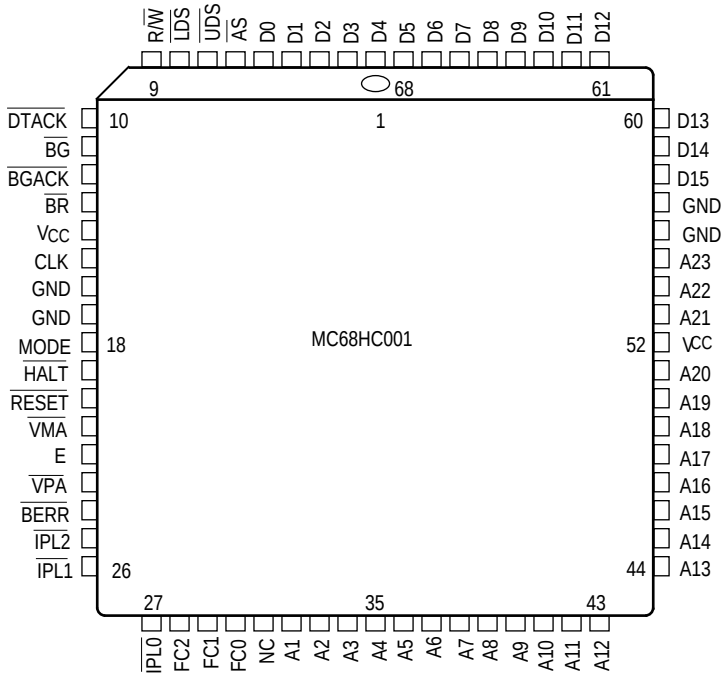


Figure 11-3. 68-Lead Quad Pack (2 of 2)

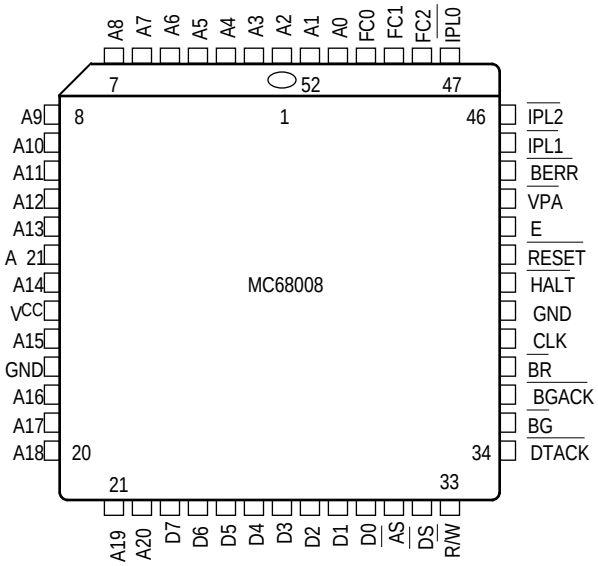


Figure 11-4. 52-Lead Quad Pack