Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

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

Applications of Embedded - Microprocessors

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

Details	
Product Status	Obsolete
Core Processor	EC000
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	8MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.21x24.21)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc000ei8

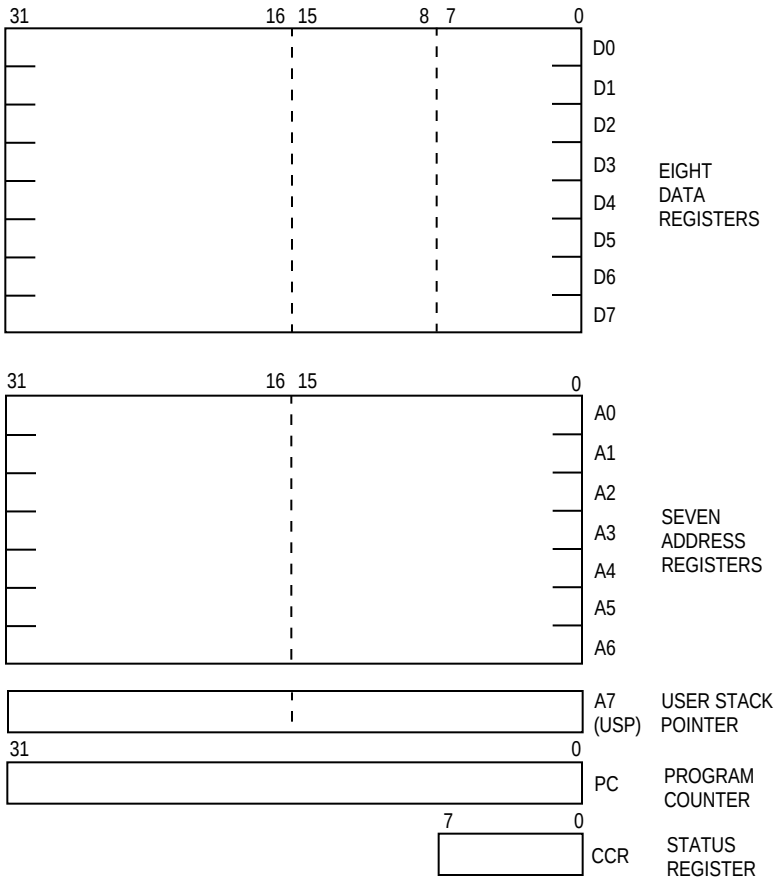


Figure 2-1. User Programmer's Model (MC68000/MC68HC000/MC68008/MC68010)

2.1.2 Supervisor Programmer's Model

The supervisor programmer's model consists of supplementary registers used in the supervisor mode. The M68000 MPUs contain identical supervisor mode register resources, which are shown in Figure 2-2, including the status register (high-order byte) and the supervisor stack pointer (SSP/A7').

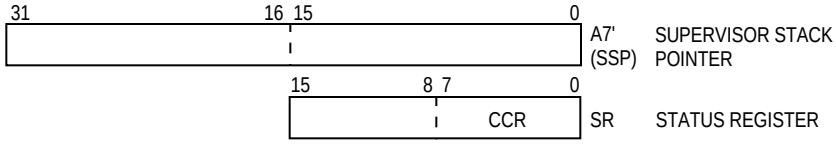


Figure 2-2. Supervisor Programmer's Model Supplement

The supervisor programmer's model supplement of the MC68010 is shown in Figure 2-3. In addition to the supervisor stack pointer and status register, it includes the vector base register (VRB) and the alternate function code registers (AFC).The VBR is used to determine the location of the exception vector table in memory to support multiple vector

A bus cycle consists of eight states. The various signals are asserted during specific states of a read cycle, as follows:

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

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

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

STATE 3 During state 3 (S3), no bus signals are altered.

STATE 4 During state 4 (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 state 5 (S5), no bus signals are altered.

STATE 6 During state 6 (S6), data from the device is driven onto the data bus.

STATE 7 On the falling edge of the clock entering state 7 (S7), the processor latches data from the addressed device and negates $\overline{AS}$ and $\overline{LDS}$, or $\overline{DS}$. At the rising edge of S7, the processor places the address bus in the high-impedance state. The device negates $\overline{DTACK}$ or $\overline{BERR}$ at this time.

NOTE

During an active bus cycle, $\overline{VPA}$ and $\overline{BERR}$ are sampled on every falling edge of the clock beginning with S4, and data is latched on the falling edge of S6 during a read cycle. The bus cycle terminates in S7, except when $\overline{BERR}$ is asserted in the absence of $\overline{DTACK}$. In that case, the bus cycle terminates one clock cycle later in S9.

4.1.2 Write Cycle

During a write cycle, the processor sends bytes of data to the memory or peripheral device. Figures 4-3 and 4-4 illustrate the write-cycle operation

The 8-bit operation performs two write cycles for a word write operation, issuing the data strobe signal during each cycle. The address bus includes the A0 bit to select the desired byte.

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.

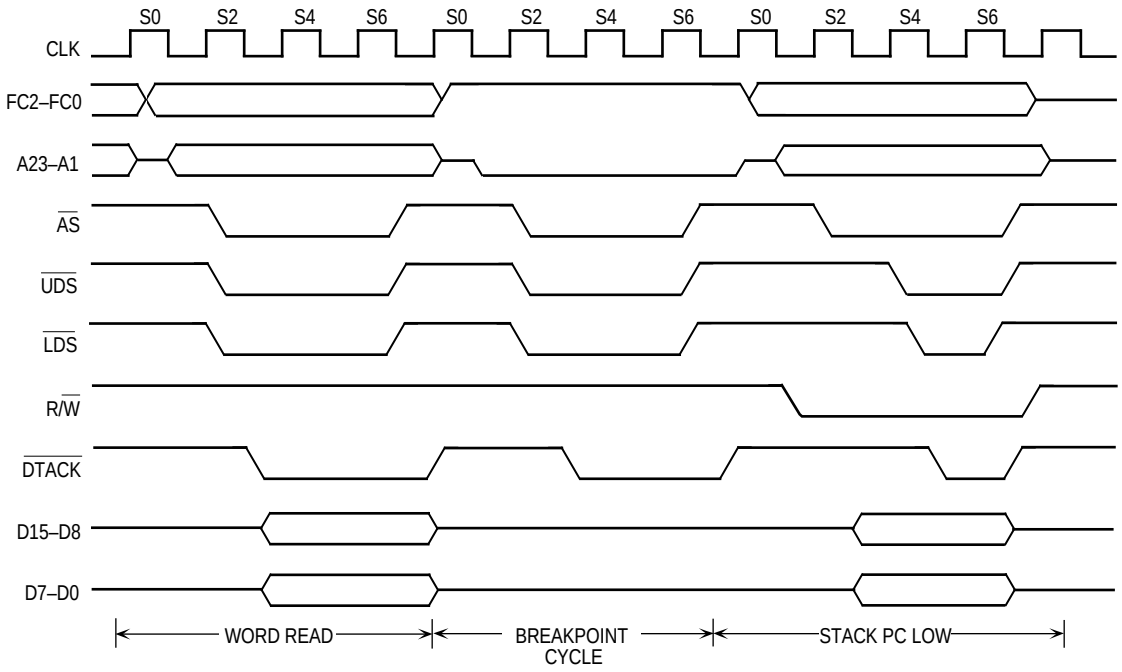


Figure 5-12. Breakpoint Acknowledge Cycle Timing Diagram

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.

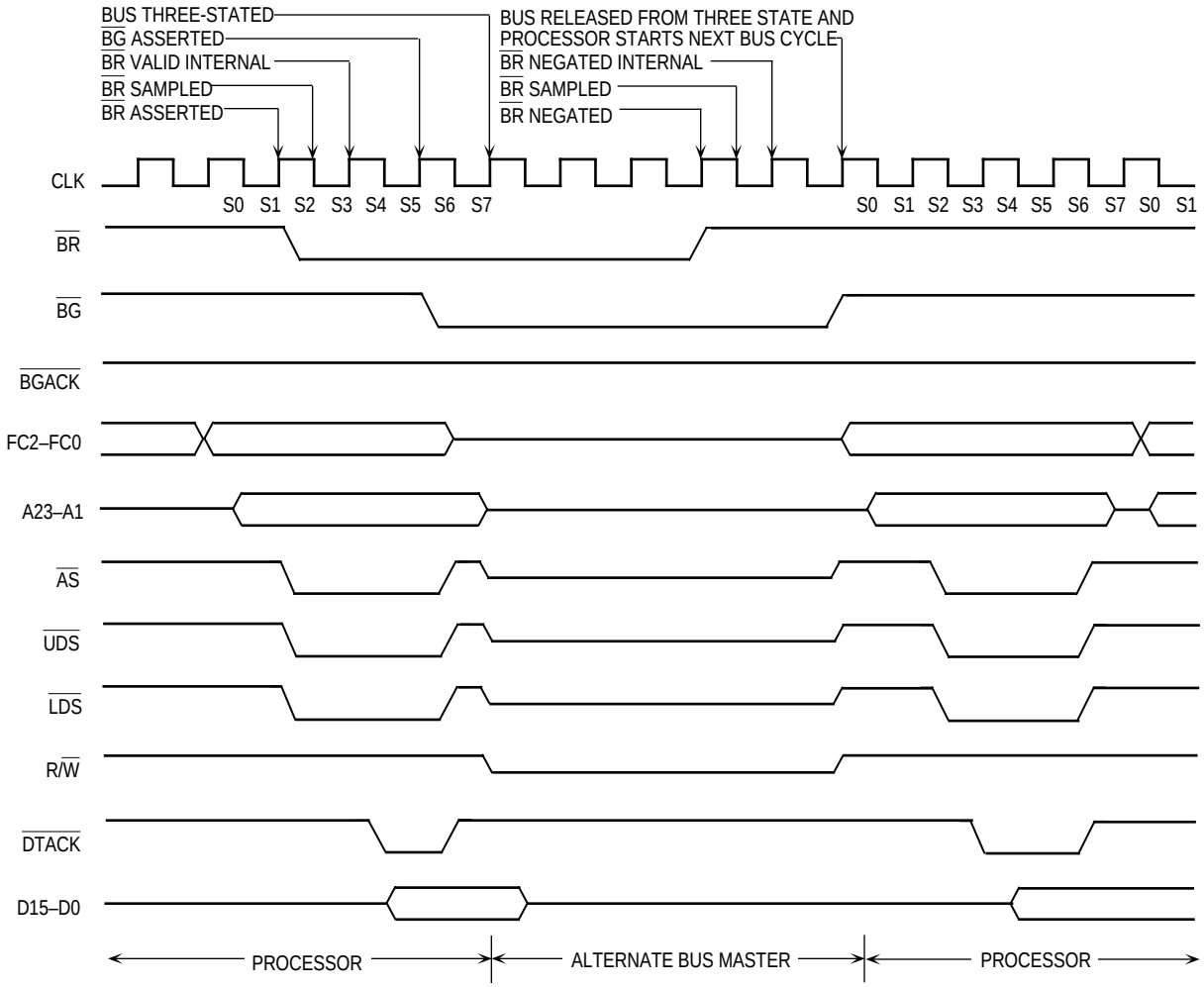


Figure 5-22. 2-Wire Bus Arbitration Timing Diagram—Processor Active

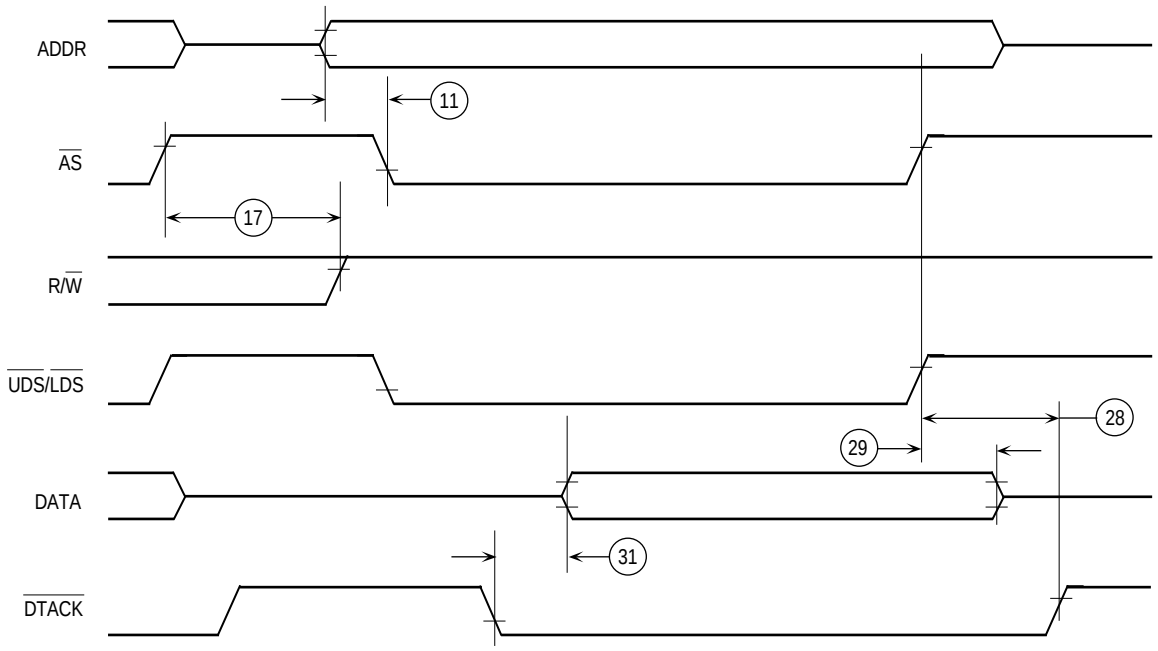


Figure 5-33. Pseudo-Asynchronous Read Cycle

During a write cycle, after the processor asserts $\overline{AS}$ but before driving the data bus, the processor drives R/$\overline{W}$ low. Parameter #55 specifies the minimum time between the transition of R/$\overline{W}$ and the driving of the data bus, which is effectively the maximum turnoff time for any device driving the data bus.

After the processor places valid data on the bus, it asserts the data strobe signal(s). A data setup time, similar to the address setup time previously discussed, can be used to improve performance. Parameter #29 is the minimum time a slave device can accept valid data before recognizing a data strobe. The slave device asserts $\overline{DTACK}$ after it accepts the data. Parameter #25 is the minimum time after negation of the strobes during which the valid data remains on the address bus. Parameter #28 is the maximum time between the negation of the strobes by the processor and the negation of $\overline{DTACK}$ by the slave device. If $\overline{DTACK}$ remains asserted past the time specified by parameter #28, the processor may recognize it as being asserted early in the next bus cycle and may terminate that cycle prematurely. Figure 5-34 shows the important timing specifications for a pseudo-asynchronous write cycle.

Table 6-3. Exception Grouping and Priority

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

6.2.4 Exception Stack Frames

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

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

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

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

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

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

6.4 RETURN FROM EXCEPTION (MC68010)

In addition to returning from any exception handler routine on the MC68010, the RTE instruction resumes the execution of a suspended instruction by returning to the normal processing state after restoring all of the temporary register and control information stored during a bus error. For the RTE instruction to execute properly, the stack must contain valid and accessible data. The RTE instruction checks for data validity in two ways. First, the format/offset word is checked for a valid stack format code. Second, if the format code indicates the long stack format, the validity of the long stack data is checked as it is loaded into the processor. In addition, the data is checked for accessibility when the processor starts reading the long data. Because of these checks, the RTE instruction executes as follows:

1. Determine the stack format. This step is the same for any stack format and consists of reading the status register, program counter, and format/offset word. If the format code indicates a short stack format, execution continues at the new program counter address. If the format code is not an MC68010-defined stack format code, exception processing starts for a format error.
2. Determine data validity. For a long-stack format, the MC68010 begins to read the remaining stack data, checking for validity of the data. The only word checked for validity is the first of the 16 internal information words (SP + 26) shown in Figure 5-8. This word contains a processor version number (in bits 10–13) and proprietary internal information that must match the version number of the MC68010 attempting to read the data. This validity check is used to ensure that the data is properly interpreted by the RTE instruction. If the version number is incorrect for this processor, the RTE instruction is aborted and exception processing begins for a format error exception. Since the stack pointer is not updated until the RTE instruction has successfully read all the stack data, a format error occurring at this point does not stack new data over the previous bus error stack information.
3. Determine data accessibility. If the long-stack data is valid, the MC68010 performs a read from the last word (SP + 56) of the long stack to determine data accessibility. If this read is terminated normally, the processor assumes that the remaining words on the stack frame are also accessible. If a bus error is signaled before or during this read, a bus error exception is taken. After this read, the processor must be able to load the remaining data without receiving a bus error; therefore, if a bus error occurs on any of the remaining stack reads, the error becomes a double bus fault, and the MC68010 enters the halted state.

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

Table 7-11 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 7-11. 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	—	**16** (4/0)	—	—	**18** (4/0)	**22** (4/0)	**18** (4/0)	**24** (6/0)	**18** (4/0)	**22** (4/0)
JSR	—	**32** (4/4)	—	—	**34** (4/4)	**38** (4/4)	**34** (4/4)	**40** (6/4)	**34** (4/4)	**32** (4/4)
LEA	—	**8**(2/0)	—	—	**16** (4/0)	**20** (4/0)	**16** (4/0)	**24** (6/0)	**16** (4/0)	**20** (4/0)
PEA	—	**24** (2/4)	—	—	**32** (4/4)	**36** (4/4)	**32** (4/4)	**40** (6/4)	**32** (4/4)	**36** (4/4)
MOVEM M → R	Word	**24+8n** (6+2n/0)	**24+8n** (6+2n/0)	—	**32+8n** (8+2n/0)	**34+8n** (8+2n/0)	**32+8n** (10+n/0)	**40+8n** (10+2n/0)	**32+8n** (8+2n/0)	**34+8n** (8+2n/0)
	Long	**24+16n** (6+4n/0)	**24+16n** (6+4n/0)	—	**32+16n** (8+4n/0)	**34+16n** (8+4n/0)	**32+16n** (8+4n/0)	**40+16n** (8+4n/0)	**32+16n** (8+4n/0)	**34+16n** (8+4n/0)
MOVEM R → M	Word	**16+8n** (4/2n)	— —	**16+8n** (4/2n)	**24+8n** (6/2n)	**26+8n** (6/2n)	**24+8n** (6/2n)	**32+8n** (8/2n)	— —	— —
	Long	**16+16n** (4/4n)	— —	**16+16n** (4/4n)	**24+16n** (6/4n)	**26+16n**	**24+16n** (8/4n)	**32+16n** (6/4n)	— —	— —

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

7.10 MULTIPRECISION INSTRUCTION EXECUTION TIMES

Table 7-12 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 7-12:

Dn — Data register operand

M — Memory operand

Table 8-3. 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)	**12**(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)	**12**(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, 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, 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.

8.3 STANDARD INSTRUCTION EXECUTION TIMES

The numbers of clock periods shown in Table 8-4 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 Table 8-4, the following notation applies:

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

Table 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 (+).

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 (+).

Table 10-1 summarizes maximum power dissipation and average junction temperature for the curve drawn in Figure 10-1, using the minimum and maximum values of ambient temperature for different packages and substituting $^{\theta}J_C$ for $^{\theta}J_A$ (assuming good thermal management). Table 10-2 provides the maximum power dissipation and average junction temperature assuming that no thermal management is applied (i.e., still air).

NOTE

Since the power dissipation curve shown in Figure 10-1 is negatively sloped, power dissipation declines as ambient temperature increases. Therefore, maximum power dissipation occurs at the lowest rated ambient temperature, but the highest average junction temperature occurs at the maximum ambient temperature where *power dissipation is lowest*.

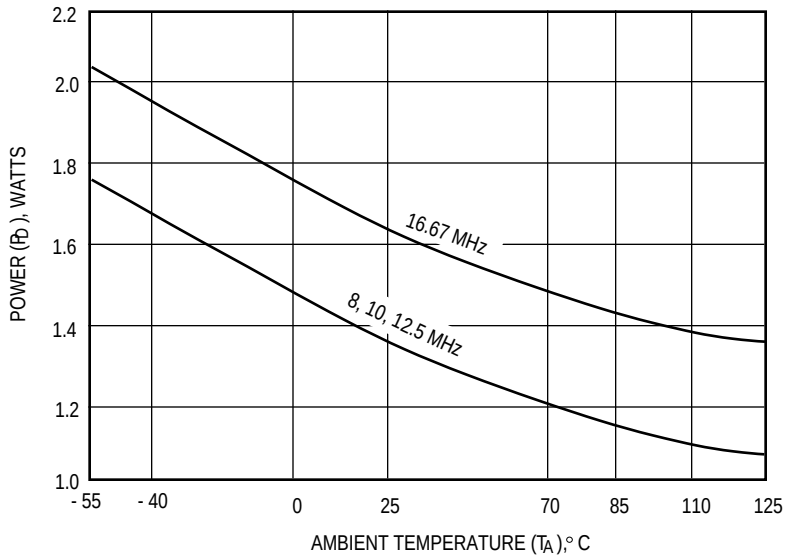


Figure 10-1. MC68000 Power Dissipation (P_D) vs Ambient Temperature (T_A) (Not Applicable to MC68HC000/68HC001/68EC000)

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.

10.12 AC ELECTRICAL SPECIFICATIONS — BUS ARBITRATION (V_{CC}=5.0 VDC±5%; GND=0 VDC, T_A=T_L TO T_H; See Figure s 10-7 – 10-11) (Applies To All Processors Except The MC68EC000)

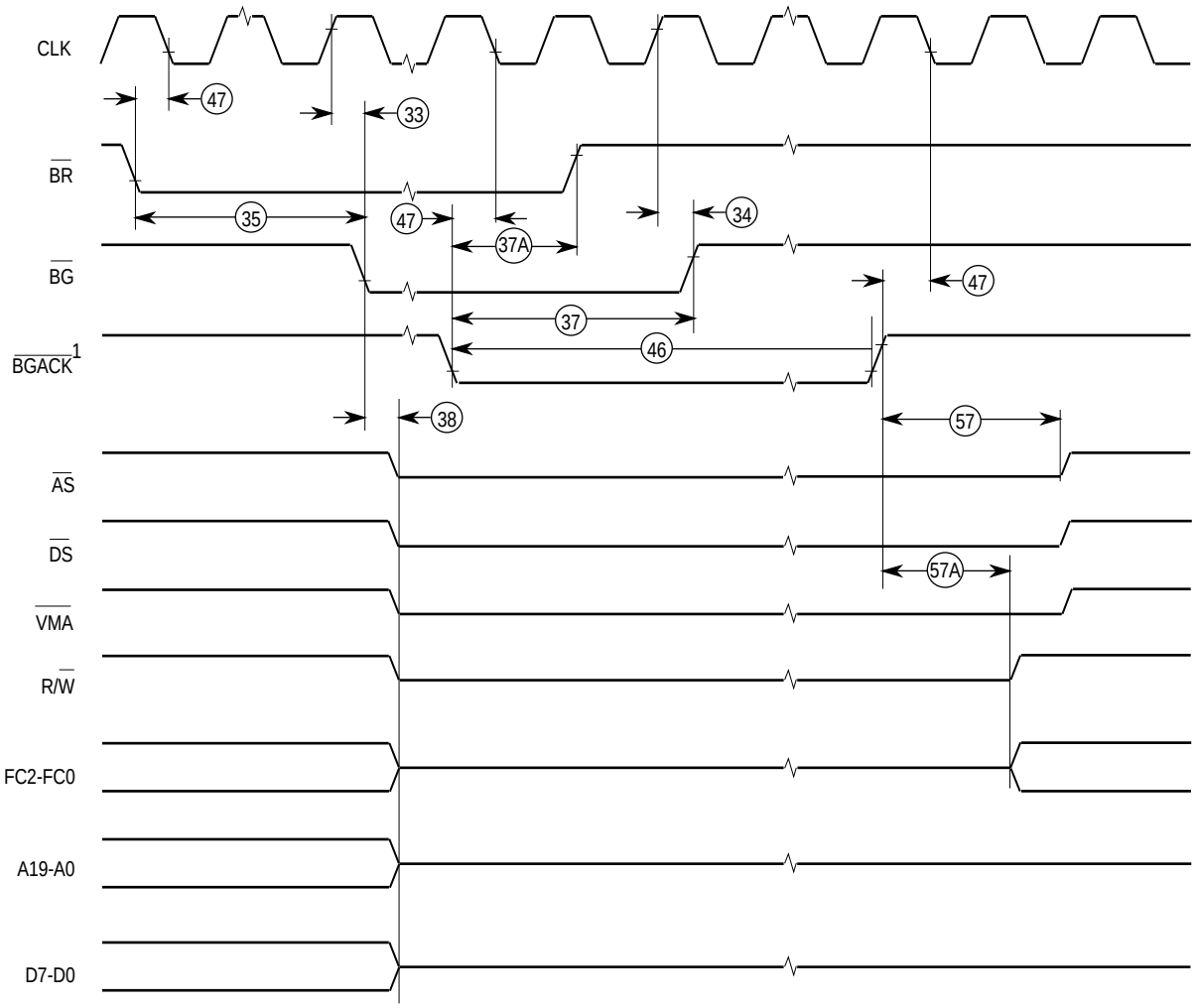
Num	Characteristic	8 MHz*		10 MHz*		12.5 MHz*		16.67 MHz 12F		16 MHz		20 MHz**		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
7	Clock High to Address, Data Bus High Impedance (Maximum)	—	80	—	70	—	60	—	50	—	50	—	42	ns
16	Clock High to Control Bus High Impedance	—	80	—	70	—	60	—	50	—	50	—	42	ns
33	Clock High to $\overline{BG}$ Asserted	—	62	—	50	—	40	0	40	0	30	0	25	ns
34	Clock High to $\overline{BG}$ Negated	—	62	—	50	—	40	0	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[1]	$\overline{BR}$ 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[2]	$\overline{BGACK}$ 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	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
46	$\overline{BGACK}$ Width Low	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	Clks
47	Asynchronous Input Setup Time	10	—	10	—	10	—	5	—	5	—	5	—	ns
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[1]	$\overline{BR}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	Clks
58A[1]	$\overline{BR}$ Negated to FC, $\overline{VMA}$ 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.

** Applies only to the MC68HC000 and MC68HC001.

NOTES:

1. Setup time for the synchronous inputs $\overline{BGACK}$, $\overline{IPL0}$-$\overline{IPL2}$, and $\overline{VPA}$ guarantees their recognition at the next falling edge of the clock.
2. $\overline{BR}$ need fall at this time only in order to insure being recognized at the end of the bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be lienar between 0.8 volt and 2.0 volts.
4. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
5. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.



NOTES: Waveform measurements for all inputs and outputs are specified at: logic high 2.0 V, logic low = 0.8 V.
1. MC68008 52-Pin Version only.

Figure 10-8. Bus Arbitration Timing
(Applies To All Processors Except The MC68EC000)

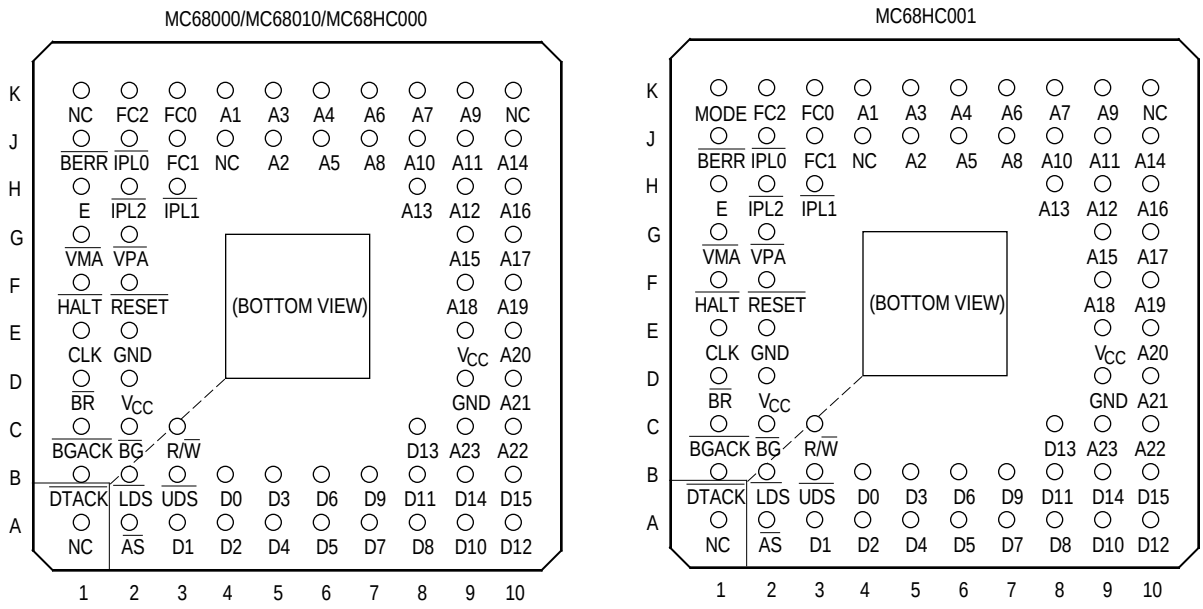


Figure 11-2. 68-Lead Pin Grid Array

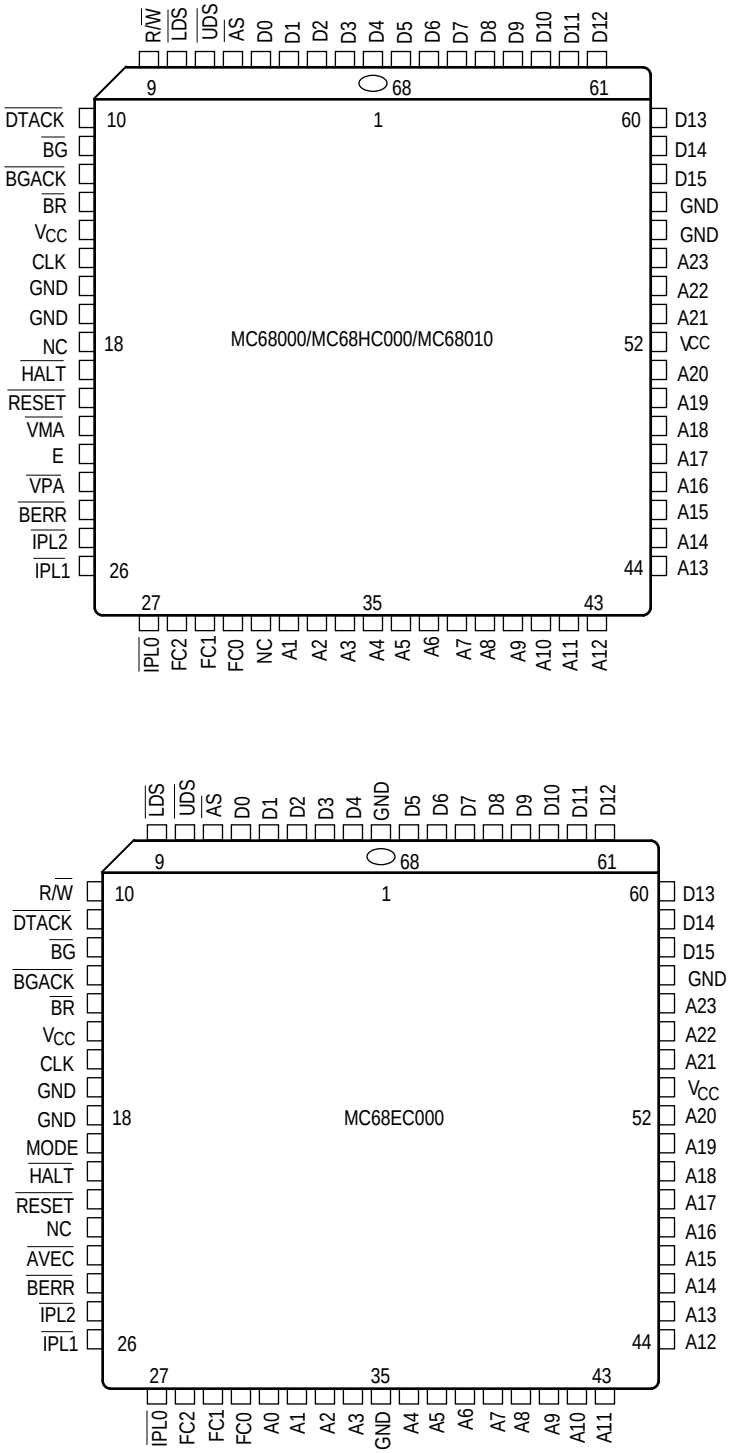


Figure 11-3. 68-Lead Quad Pack (1 of 2)

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

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

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

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

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

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

B.2 INTERRUPT INTERFACE OPERATION

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

B

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