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	Active
Core Processor	EC000
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	12MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	68-BCPGA
Supplier Device Package	68-PGA (26.92x26.92)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68hc000rc12

In addition, operations on other data types, such as memory addresses, status word data, etc., are provided in the instruction set.

The 14 flexible addressing modes, shown in Table 2-1, include six basic types:

1. Register Direct
2. Register Indirect
3. Absolute
4. Immediate
5. Program Counter Relative
6. Implied

The register indirect addressing modes provide postincrementing, predecrementing, offsetting, and indexing capabilities. The program counter relative mode also supports indexing and offsetting. For detail information on addressing modes refer to M68000PM/AD, *M68000 Programmer Reference Manual*.

3.11 SIGNAL SUMMARY

Table 3-4 summarizes the signals discussed in the preceding paragraphs.

Table 3-4. Signal Summary

Signal Name	Mnemonic	Input/Output	Active State	Hi-Z	
				On $\overline{\text{HALT}}$	On Bus Relinquish
Address Bus	A0–A23	Output	High	Yes	Yes
Data Bus	D0–D15	Input/Output	High	Yes	Yes
Address Strobe	$\overline{\text{AS}}$	Output	Low	No	Yes
Read/Write	R/$\overline{\text{W}}$	Output	Read-High Write-Low	No	Yes
Data Strobe	$\overline{\text{DS}}$	Output	Low	No	Yes
Upper and Lower Data Strobes	$\overline{\text{UDS}}$, $\overline{\text{LDS}}$	Output	Low	No	Yes
Data Transfer Acknowledge	$\overline{\text{DTACK}}$	Input	Low	No	No
Bus Request	$\overline{\text{BR}}$	Input	Low	No	No
Bus Grant	$\overline{\text{BG}}$	Output	Low	No	No
Bus Grant Acknowledge	$\overline{\text{BGACK}}$	Input	Low	No	No
Interrupt Priority Level	$\overline{\text{IPL}}$0, $\overline{\text{IPL}}$1, $\overline{\text{IPL}}$2	Input	Low	No	No
Bus Error	$\overline{\text{BERR}}$	Input	Low	No	No
Mode	MODE	Input	High	—	—
Reset	$\overline{\text{RESET}}$	Input/Output	Low	No*	No*
Halt	$\overline{\text{HALT}}$	Input/Output	Low	No*	No*
Enable	E	Output	High	No	No
Valid Memory Address	$\overline{\text{VMA}}$	Output	Low	No	Yes
Valid Peripheral Address	$\overline{\text{VPA}}$	Input	Low	No	No
Function Code Output	FC0, FC1, FC2	Output	High	No	Yes
Clock	CLK	Input	High	No	No
Power Input	V_{CC}	Input	—	—	—
Ground	GND	Input	—	—	—

*Open drain.

SECTION 5
16-BIT BUS OPERATION

The following paragraphs describe control signal and bus operation for 16-bit bus operations during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation. The 16-bit bus operation devices are the MC68000, MC68HC000, MC68010, and the MC68HC001 and MC68EC000 in 16-bit mode. The MC68HC001 and MC68EC000 select 16-bit mode by pulling mode high or leave it floating during reset.

5.1 DATA TRANSFER OPERATIONS

Transfer of data between devices involves the following signals:

1. Address bus A1 through highest numbered address line
2. Data bus D0 through D15
3. Control signals

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cases, the bus master must deskew all signals it issues at both the start and end of a bus cycle. In addition, the bus master must deskew the acknowledge and data signals from the slave device.

The following paragraphs describe the read, write, read-modify-write, and CPU space cycles. The indivisible read-modify-write cycle implements interlocked multiprocessor communications. A CPU space cycle is a special processor cycle.

5.1.1 Read Cycle

During a read cycle, the processor receives either one or two bytes of data from the memory or from a peripheral device. If the instruction specifies a word or long-word operation, the MC68000, MC68HC000, MC68HC001, MC68EC000, or MC68010 processor reads both upper and lower bytes simultaneously by asserting both upper and lower data strobes. When the instruction specifies byte operation, the processor uses the internal A0 bit to determine which byte to read and issues the appropriate data strobe. When A0 equals zero, the upper data strobe is issued; when A0 equals one, the lower data strobe is issued. When the data is received, the processor internally positions the byte appropriately.

The word read-cycle flowchart is shown in Figure 5-1 and the byte read-cycle flowchart is shown in Figure 5-2. The read and write cycle timing is shown in Figure 5-3 and the word and byte read-cycle timing diagram is shown in Figure 5-4.

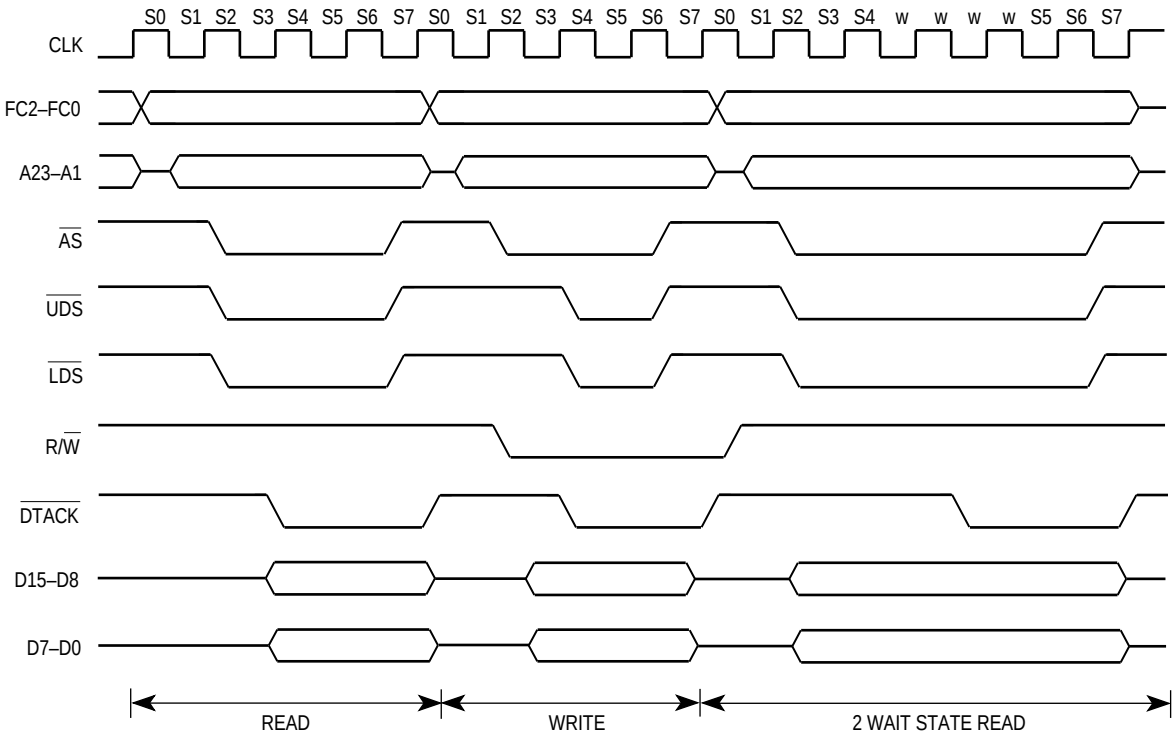


Figure 5-3. Read and Write-Cycle Timing Diagram

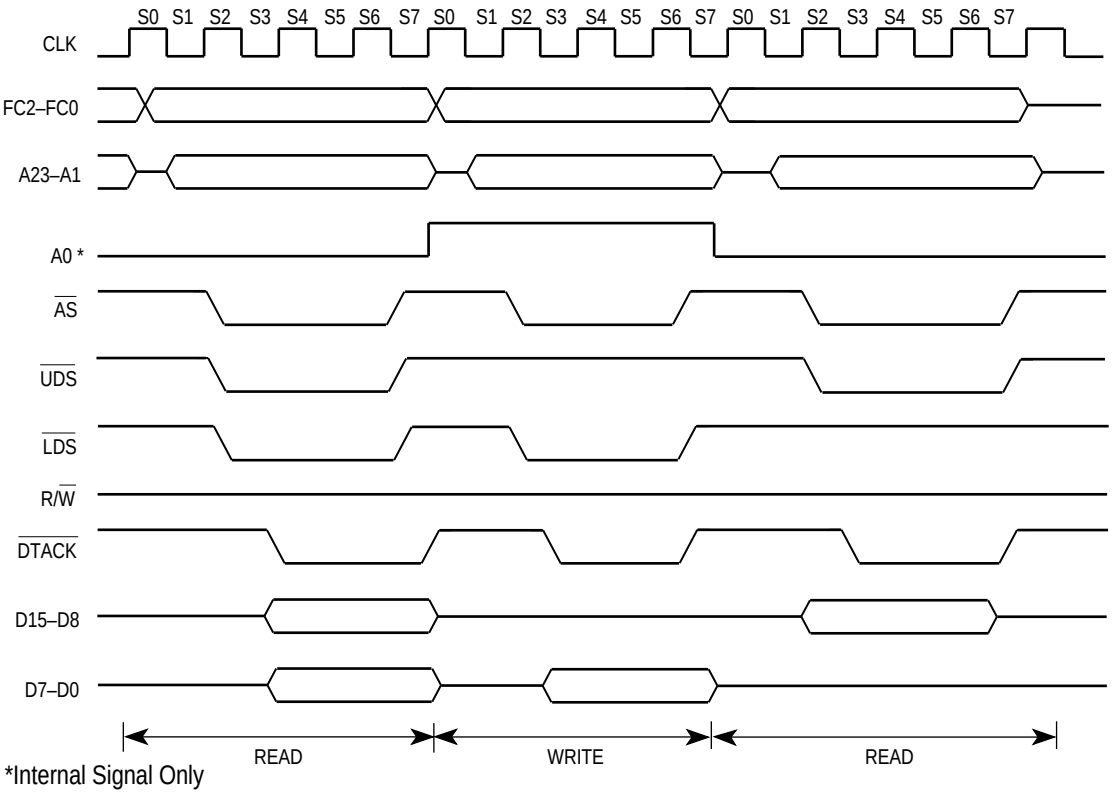


Figure 5-4. Word and Byte Read-Cycle Timing Diagram

STATE 7 On the falling edge of the clock entering S7, the processor negates $\overline{AS}$, $\overline{UDS}$, or $\overline{LDS}$. As the clock rises at the end of S7, the processor places the address and data buses in the high-impedance state, and drives R/$\overline{W}$ high. The device negates $\overline{DTACK}$ or $\overline{BERR}$ at this time.

5.1.3 Read-Modify-Write Cycle.

The read-modify-write cycle performs a read operation, modifies the data in the arithmetic logic unit, and writes the data back to the same address. The address strobe ($\overline{AS}$) remains asserted throughout the entire cycle, making the cycle indivisible. The test and set (TAS) instruction uses this cycle to provide a signaling capability without deadlock between processors in a multiprocessing environment. The TAS instruction (the only instruction that uses the read-modify-write cycle) only operates on bytes. Thus, all read-modify-write cycles are byte operations. The read-modify-write flowchart shown in Figure 5-8 and the timing diagram in Figure 5-9, applies to the MC68000, the MC68HC000, the MC68HC001 (in 16-bit mode), the MC68EC000 (in 16-bit mode), and the MC68010.

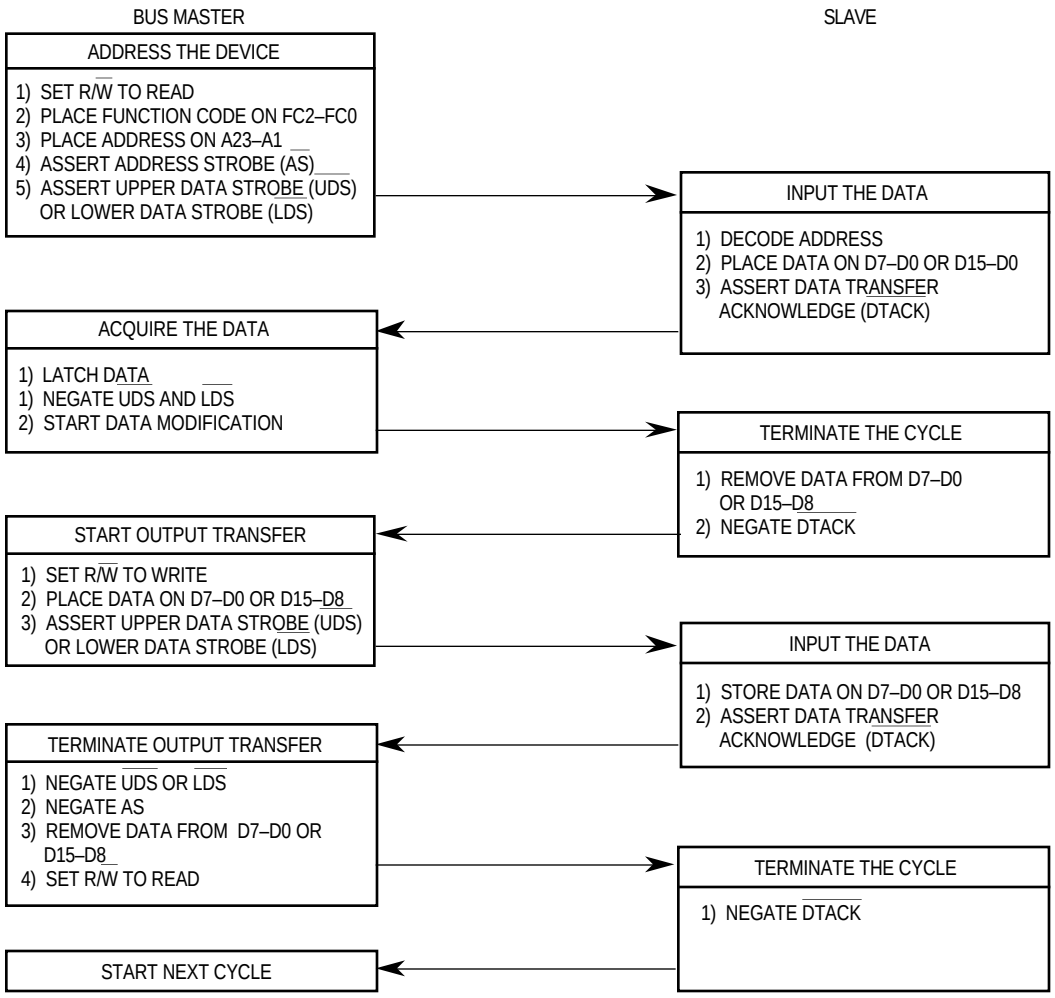


Figure 5-8. Read-Modify-Write Cycle Flowchart

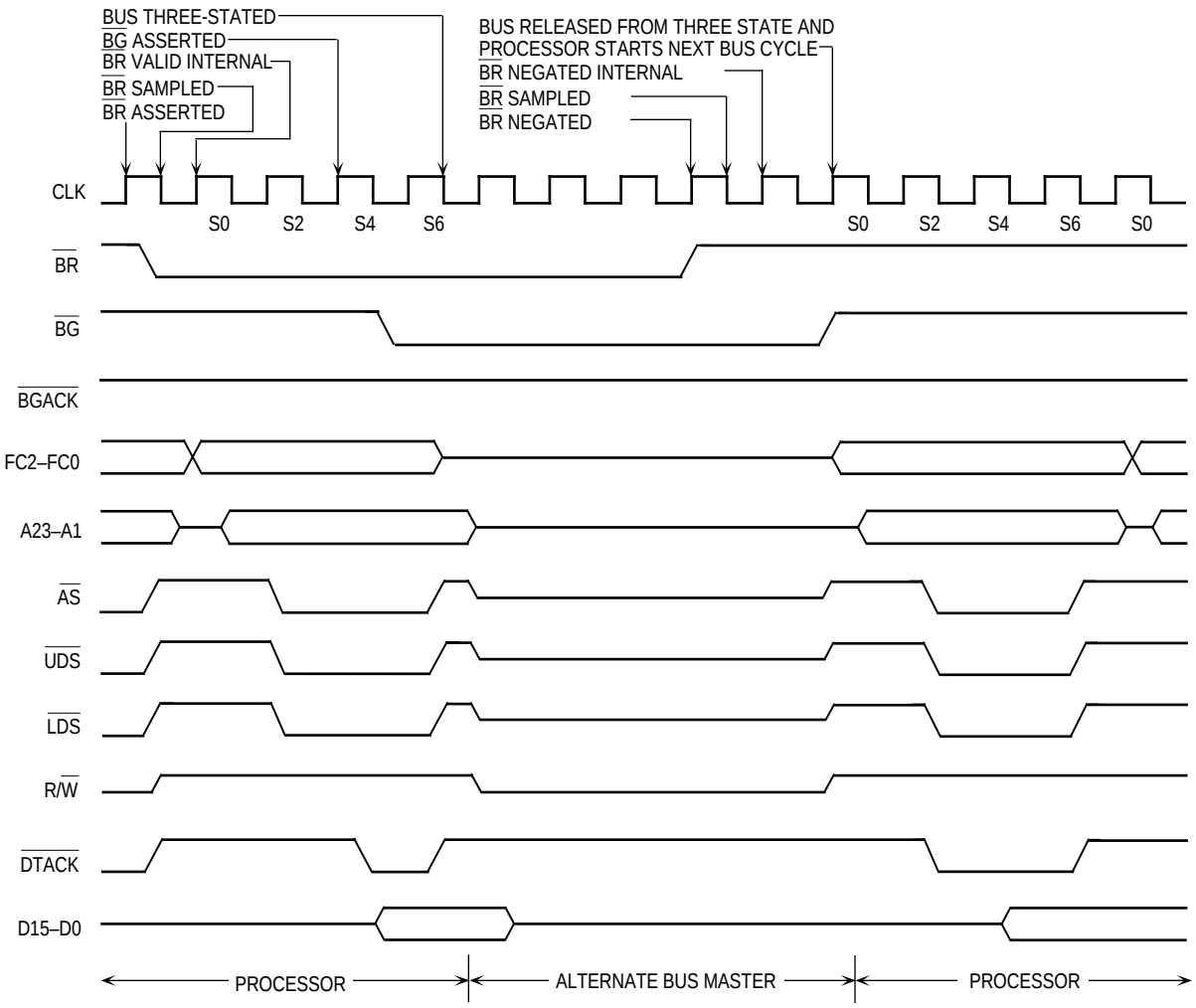


Figure 5-24. 2-Wire Bus Arbitration Timing Diagram—Special Case

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.

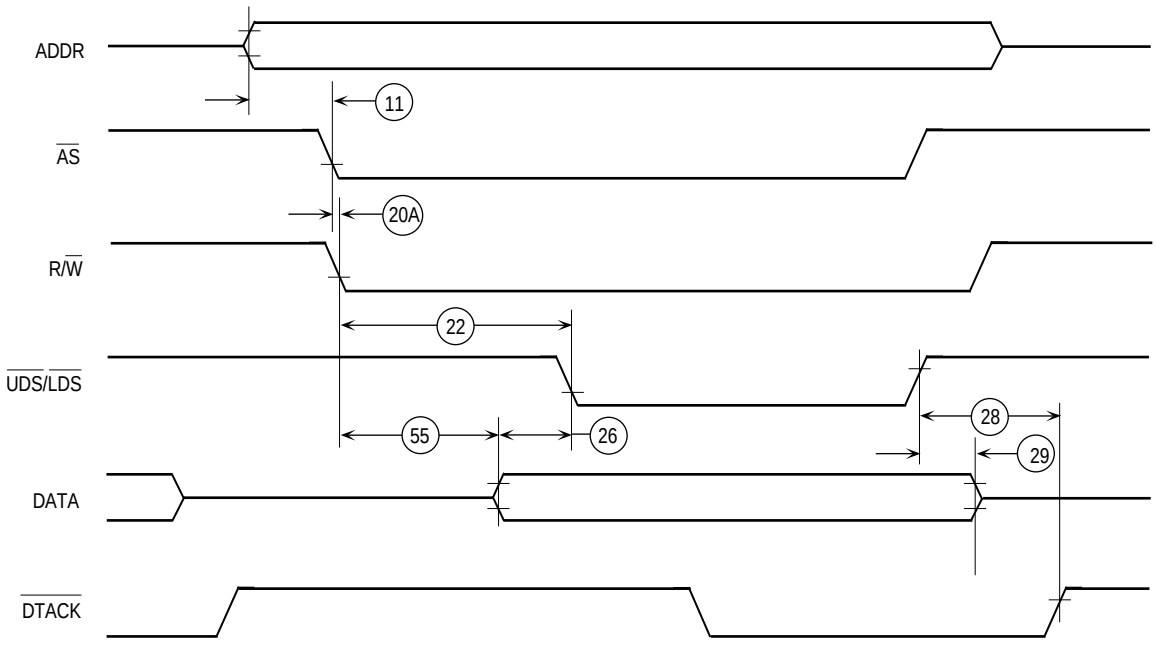


Figure 5-34. Pseudo-Asynchronous Write Cycle

In the MC68010, the BERR signal can be delayed after the assertion of DTACK. Specification #48 is the maximum time between assertion of DTACK and assertion of BERR. If this maximum delay is exceeded, operation of the processor may be erratic.

5.8 SYNCHRONOUS OPERATION

In some systems, external devices use the system clock to generate DTACK and other asynchronous input signals. This synchronous operation provides a closely coupled design with maximum performance, appropriate for frequently accessed parts of the system. For example, memory can operate in the synchronous mode, but peripheral devices operate asynchronously. For a synchronous device, the designer uses explicit timing information shown in **Section 10 Electrical Characteristics**. These specifications define the state of all bus signals relative to a specific state of the processor clock.

The standard M68000 bus cycle consists of four clock periods (eight bus cycle states) and, optionally, an integral number of clock cycles inserted as wait states. Wait states are inserted as required to allow sufficient response time for the external device. The following state-by-state description of the bus cycle differs from those descriptions in **5.1.1 READ CYCLE** and **5.1.2 WRITE CYCLE** by including information about the important timing parameters that apply in the bus cycle states.

STATE 0 The bus cycle starts in S0, during which the clock is high. At the rising edge of S0, the function code for the access is driven externally. Parameter #6A defines the delay from this rising edge until the function codes are valid. Also, the R/W signal is driven high; parameter #18 defines the delay from the same rising edge to the transition of R/W. The minimum value for parameter #18 applies to a read cycle preceded by a write cycle; this value

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

6.1.1 Supervisor Mode

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

6.1.2 User Mode

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

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

NOTE

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

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

6.1.3 Privilege Mode Changes

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

supervisor mode. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege mode.

NOTE

The transition from supervisor to user mode can be accomplished by any of four instructions: return from exception (RTE) (MC68010 only), move to status register (MOVE to SR), AND immediate to status register (ANDI to SR), and exclusive OR immediate to status register (EORI to SR). The RTE instruction in the MC68010 fetches the new status register and program counter from the supervisor stack and loads each into its respective register. Next, it begins the instruction fetch at the new program counter address in the privilege mode determined by the S bit of the new contents of the status register.

The MOVE to SR, ANDI to SR, and EORI to SR instructions fetch all operands in the supervisor mode, perform the appropriate update to the status register, and then fetch the next instruction at the next sequential program counter address in the privilege mode determined by the new S bit.

6.1.4 Reference Classification

When the processor makes a reference, it classifies the reference according to the encoding of the three function code output lines. This classification allows external translation of addresses, control of access, and differentiation of special processor states, such as CPU space (used by interrupt acknowledge cycles). Table 6-1 lists the classification of references.

Table 6-1. Reference Classification

Function Code Output			
FC2	FC1	FC0	Address Space
0	0	0	(Undefined, Reserved)*
0	0	1	User Data
0	1	0	User Program
0	1	1	(Undefined, Reserved)*
1	0	0	(Undefined, Reserved)*
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	CPU Space

*Address space 3 is reserved for user definition, while 0 and 4 are reserved for future use by Motorola.

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.

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)

9.1 OPERAND EFFECTIVE ADDRESS CALCULATION TIMES

Table 9-1 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 9-1. Effective Address Calculation Times

Addressing Mode		Byte, Word		Long	
		Fetch	No Fetch	Fetch	No Fetch
	Register				
Dn	Data Register Direct	**0**(0/0)	—	**0**(0/0)	—
An	Address Register Direct	**0**(0/0)	—	**0**(0/0)	—
	Memory				
(An)	Address Register Indirect	**4**(1/0)	**2**(0/0)	**8**(2/0)	**2**(0/0)
(An)+	Address Register Indirect with Postincrement	**4**(1/0)	**4**(0/0)	**8**(2/0)	**4**(0/0)
–(An)	Address Register Indirect with Predecrement	**6**(1/0)	**4**(0/0)	**10**(2/0)	**4**(0/0)
(d_{16}, An)	Address Register Indirect with Displacement	**8**(2/0)	**4**(0/0)	**12**(3/0)	**4**(1/0)
(d_8, An, Xn)*	Address Register Indirect with Index	**10**(2/0)	**8**(1/0)	**14**(3/0)	**8**(1/0)
(xxx).W	Absolute Short	**8**(2/0)	**4**(1/0)	**12**(3/0)	**4**(1/0)
(xxx).L	Absolute Long	**12**(3/0)	**8**(2/0)	**16**(4/0)	**8**(2/0)
(d_{16}, PC)	Program Counter Indirect with Displacement	**8**(2/0)	—	**12**(3/0)	—
(d_8, PC, Xn)*	Program Counter Indirect with Index	**10**(2/0)	—	**14**(3/0)	—
#<data>	Immediate	**4**(1/0)	—	**8**(2/0)	—

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

9.2 MOVE INSTRUCTION EXECUTION TIMES

Tables 9-2, 9-3, 9-4, and 9-5 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 9-2. Move Byte and Word Instruction Execution Times

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

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

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

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

*Word only.

SECTION 10
ELECTRICAL AND THERMAL CHARACTERISTICS

This section provides information on the maximum rating and thermal characteristics for the MC68000, MC68HC000, MC68HC001, MC68EC000, MC68008, and MC68010.

10.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	–0.3 to 7.0	V
Input Voltage	V_{in}	–0.3 to 7.0	V
Maximum Operating Temperature Range Commerical Extended "C" Grade Commerical Extended "I" Grade	T_A	T_L to T_H 0 to 70 –40 to 85 0 to 85	°C
Storage Temperature	T_{stg}	–55 to 150	°C

This device contains protective circuitry against damage due to high static voltages or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

10.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Symbol	Value	Rating
Thermal Resistance	θ_{JA}		θ_{JC}		°C/W
Ceramic, Type L/LC		30		15*	
Ceramic, Type R/RC		33		15	
Plastic, Type P		30		15*	
Plastic, Type FN		45*		25*	

*Estimated

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

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

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

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

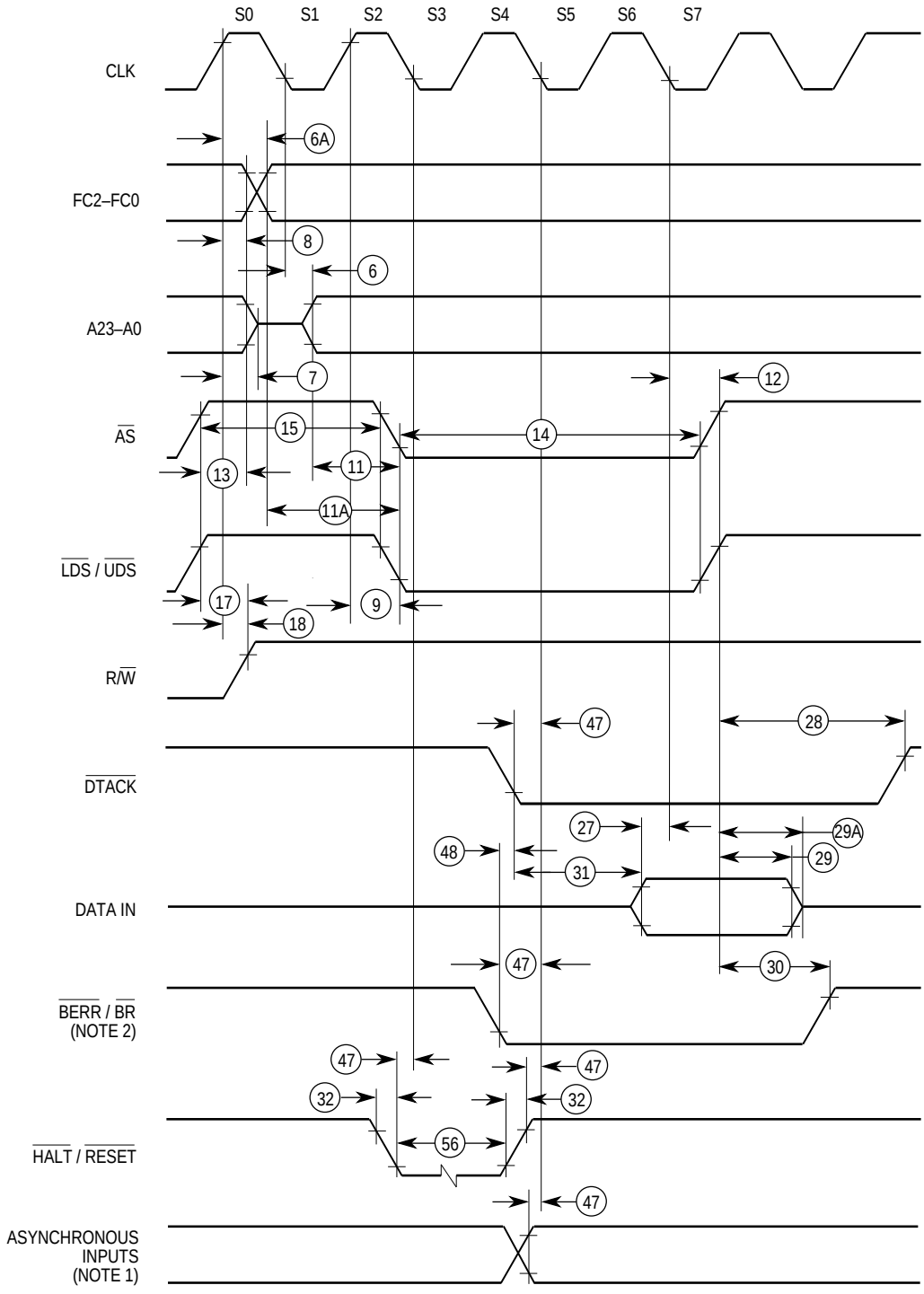
10.5 AC ELECTRICAL SPECIFICATION DEFINITIONS

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

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

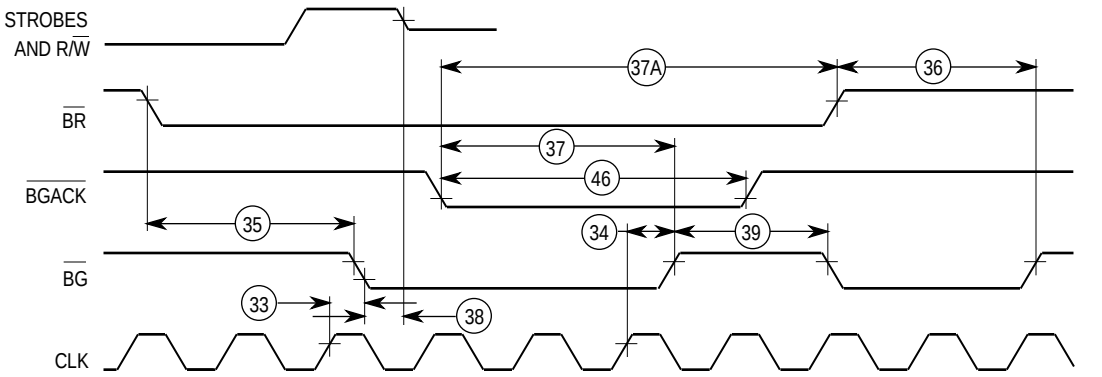
NOTE

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



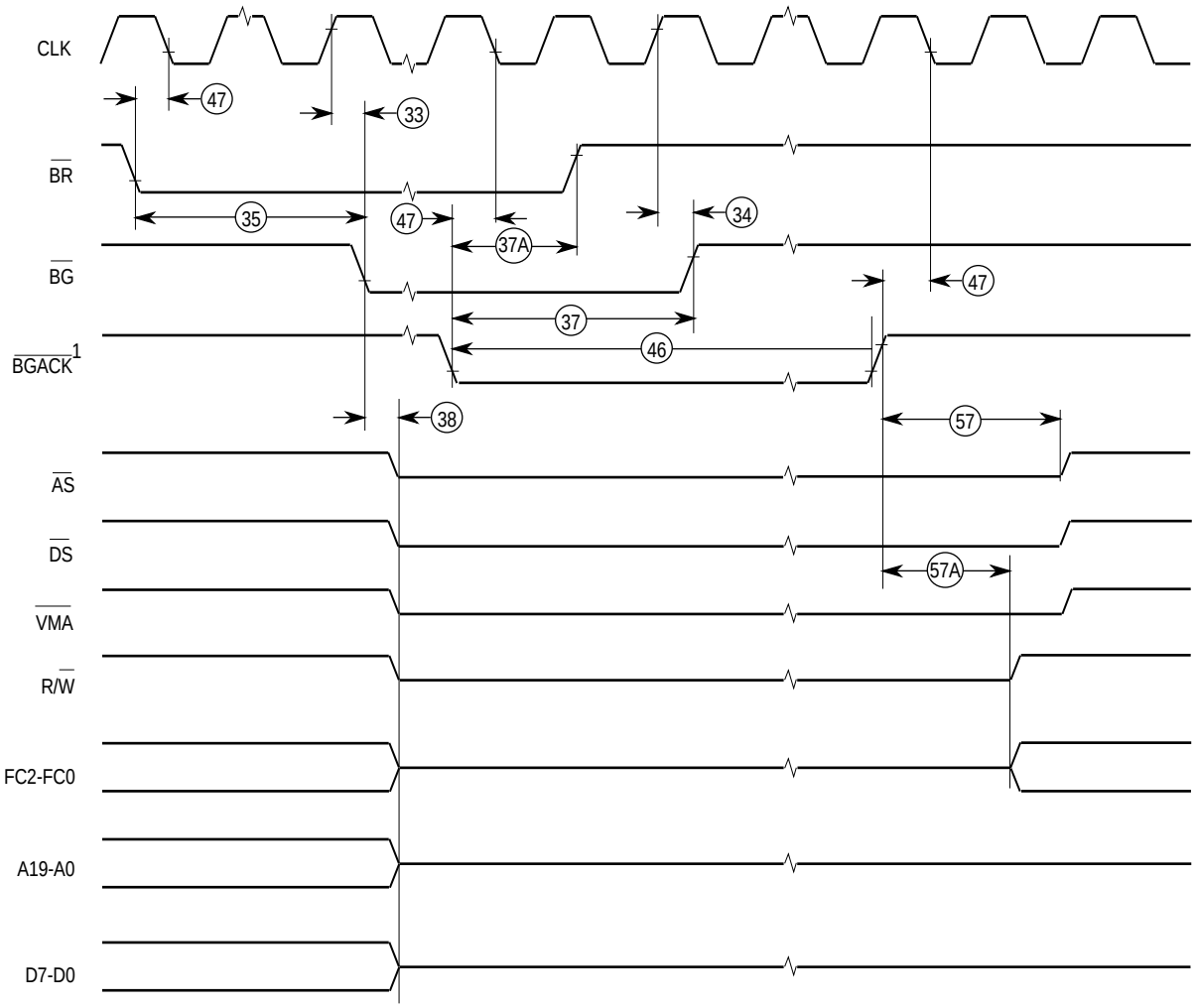
NOTES:
1. Setup time for the asynchronous inputs IPL2–IPL0 and AVEC (#47) guarantees their recognition at the next falling edge of the clock.
2. BR need fall at this time only to insure being recognized at the end of the bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall is linear between 0.8 V and 2.0 V.

Figure 10-4. Read Cycle Timing Diagram
(Applies To All Processors Except The MC68EC000)



NOTE: Setup time to the clock (#47) for the asynchronous inputs BERR, BGACK, BR, DTACK, IPL2-IPL0, and VPA guarantees their recognition at the next falling edge of the clock.

Figure 10-7. Bus Arbitration Timing
(Applies To All Processors Except The MC68EC000)



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)

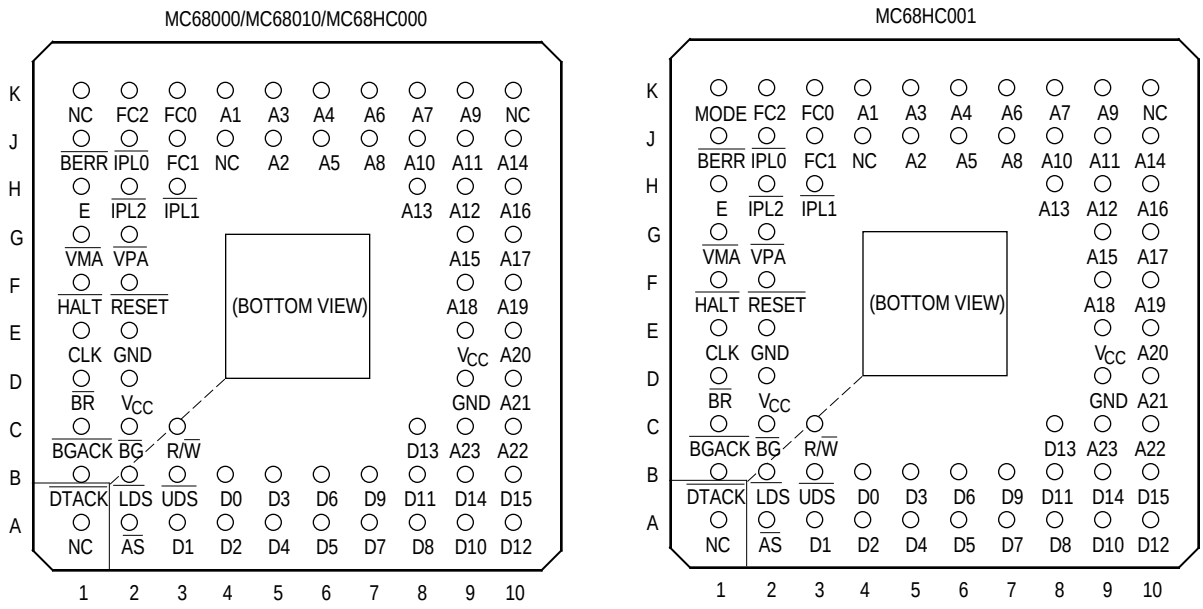


Figure 11-2. 68-Lead Pin Grid Array

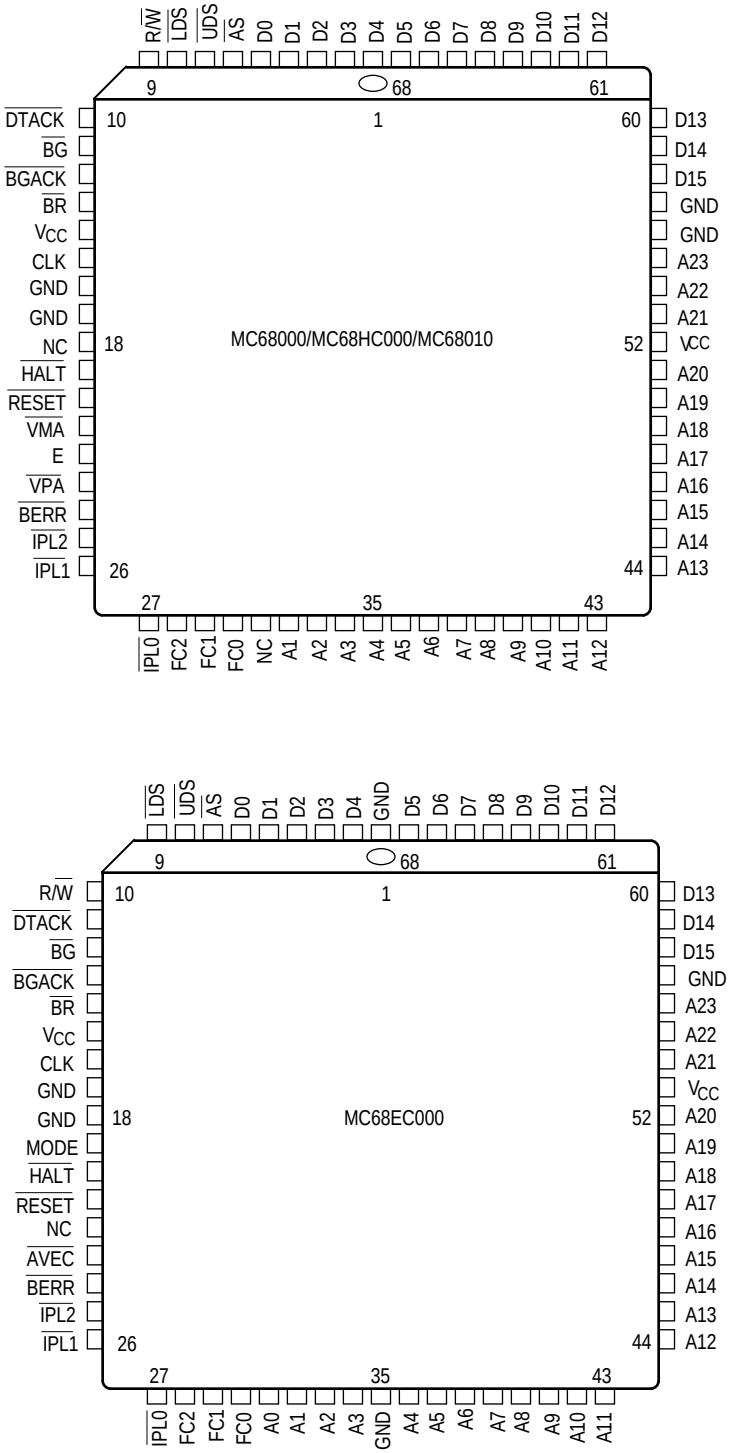


Figure 11-3. 68-Lead Quad Pack (1 of 2)