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-BCPGA
Supplier Device Package	68-PGA (26.92x26.92)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc000rc16

1.4 MC68HC000

The primary benefit of the MC68HC000 is reduced power consumption. The device dissipates an order of magnitude less power than the HMOS MC68000.

The MC68HC000 is an implementation of the M68000 16/-32 bit microprocessor architecture. The MC68HC000 has a 16-bit data bus implementation of the MC68000 and is upward code-compatible with the MC68010 virtual extensions and the MC68020 32-bit implementation of the architecture.

1.5 MC68HC001

The MC68HC001 provides a functional extension to the MC68HC000 HCMOS 16-/32-bit microprocessor with the addition of statically selectable 8- or 16-bit data bus operation. The MC68HC001 is object-code compatible with the MC68HC000, and code written for the MC68HC001 can be migrated without modification to any member of the M68000 Family.

1.6 MC68EC000

The MC68EC000 is an economical high-performance embedded controller designed to suit the needs of the cost-sensitive embedded controller market. The HCMOS MC68EC000 has an internal 32-bit architecture that is supported by a statically selectable 8- or 16-bit data bus. This architecture provides a fast and efficient processing device that can satisfy the requirements of sophisticated applications based on high-level languages.

The MC68EC000 is object-code compatible with the MC68000, and code written for the MC68EC000 can be migrated without modification to any member of the M68000 Family.

The MC68EC000 brings the performance level of the M68000 Family to cost levels previously associated with 8-bit microprocessors. The MC68EC000 benefits from the rich M68000 instruction set and its related high code density with low memory bandwidth requirements.

Notation for operands:

PC — Program counter
SR — Status register
V — Overflow condition code
Immediate Data — Immediate data from the instruction
Source — Source contents
Destination — Destination contents
Vector — Location of exception vector
+inf — Positive infinity
–inf — Negative infinity
<fmt> — Operand data format: byte (B), word (W), long (L), single (S), double (D), extended (X), or packed (P).
FPm — One of eight floating-point data registers (always specifies the source register)
FPn — One of eight floating-point data registers (always specifies the destination register)

Notation for subfields and qualifiers:

<bit> of <operand> — Selects a single bit of the operand
<ea>{offset:width} — Selects a bit field
(<operand>) — The contents of the referenced location
<operand>10 — The operand is binary-coded decimal, operations are performed in decimal
(<address register>) — The register indirect operator
–(<address register>) — Indicates that the operand register points to the memory
(<address register>)+ — Location of the instruction operand—the optional mode qualifiers are –, +, (d), and (d, ix)
#xxx or #<data> — Immediate data that follows the instruction word(s)

Notations for operations that have two operands, written <operand> <op> <operand>, where <op> is one of the following:

→ — The source operand is moved to the destination operand
↔ — The two operands are exchanged
+ — The operands are added
– — The destination operand is subtracted from the source operand
× — The operands are multiplied
÷ — The source operand is divided by the destination operand
< — Relational test, true if source operand is less than destination operand
> — Relational test, true if source operand is greater than destination operand
V — Logical OR
⊕ — Logical exclusive OR
Λ — Logical AND

SECTION 3
SIGNAL DESCRIPTION

This section contains descriptions of the input and output signals. The input and output signals can be functionally organized into the groups shown in Figure 3-1 (for the MC68000, the MC68HC000 and the MC68010), Figure 3-2 (for the MC68HC001), Figure 3-3 (for the MC68EC000), Figure 3-4 (for the MC68008, 48-pin version), and Figure 3-5 (for the MC68008, 52-pin version). The following paragraphs provide brief descriptions of the signals and references (where applicable) to other paragraphs that contain more information about the signals.

NOTE

The terms **assertion** and **negation** are used extensively in this manual to avoid confusion when describing a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true, independently of whether that level is represented by a high or low voltage. The term negate or negation is used to indicate that a signal is inactive or false.

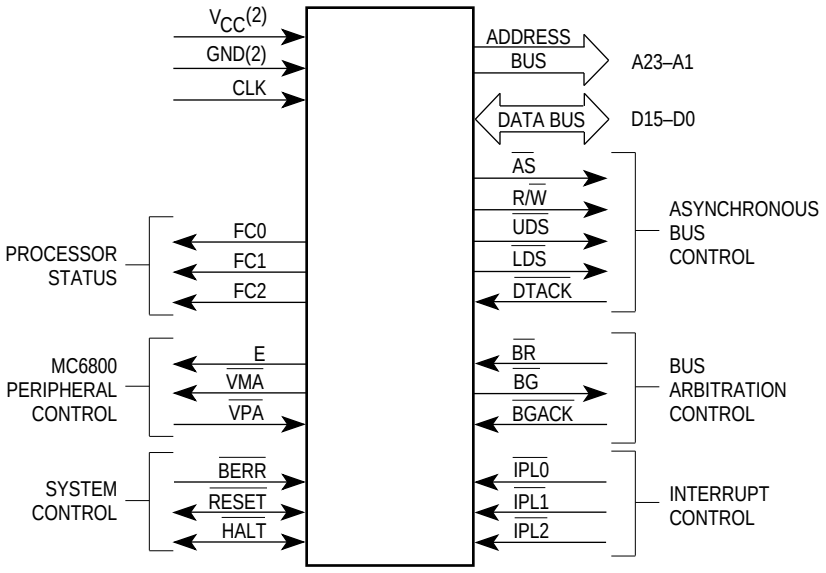


Figure 3-1. Input and Output Signals (MC68000, MC68HC000 and MC68010)

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.

The descriptions of the eight states of a write cycle are as follows:

STATE 0 The write cycle starts in S0. The processor places valid function codes on FC2–FC0 and drives R/$\overline{W}$ high (if a preceding write cycle has left R/$\overline{W}$ low).

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

STATE 2 On the rising edge of S2, the processor asserts $\overline{AS}$ and drives R/$\overline{W}$ low.

STATE 3 During S3, the data bus is driven out of the high-impedance state as the data to be written is placed on the bus.

STATE 4 At the rising edge of S4, the processor asserts $\overline{LDS}$, or $\overline{DS}$. The processor waits for a cycle termination signal ($\overline{DTACK}$ or $\overline{BERR}$) or $\overline{VPA}$, an M6800 peripheral signal. When $\overline{VPA}$ is asserted during S4, the cycle becomes a peripheral cycle (refer to **Appendix B M6800 Peripheral Interface**). If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{DTACK}$ or $\overline{BERR}$ is asserted.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, no bus signals are altered.

STATE 7 On the falling edge of the clock entering S7, the processor negates $\overline{AS}$, $\overline{LDS}$, and $\overline{DS}$. 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.

4.1.3 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. Figure 4-5 and 4-6 illustrate the read-modify-write cycle operation.

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.

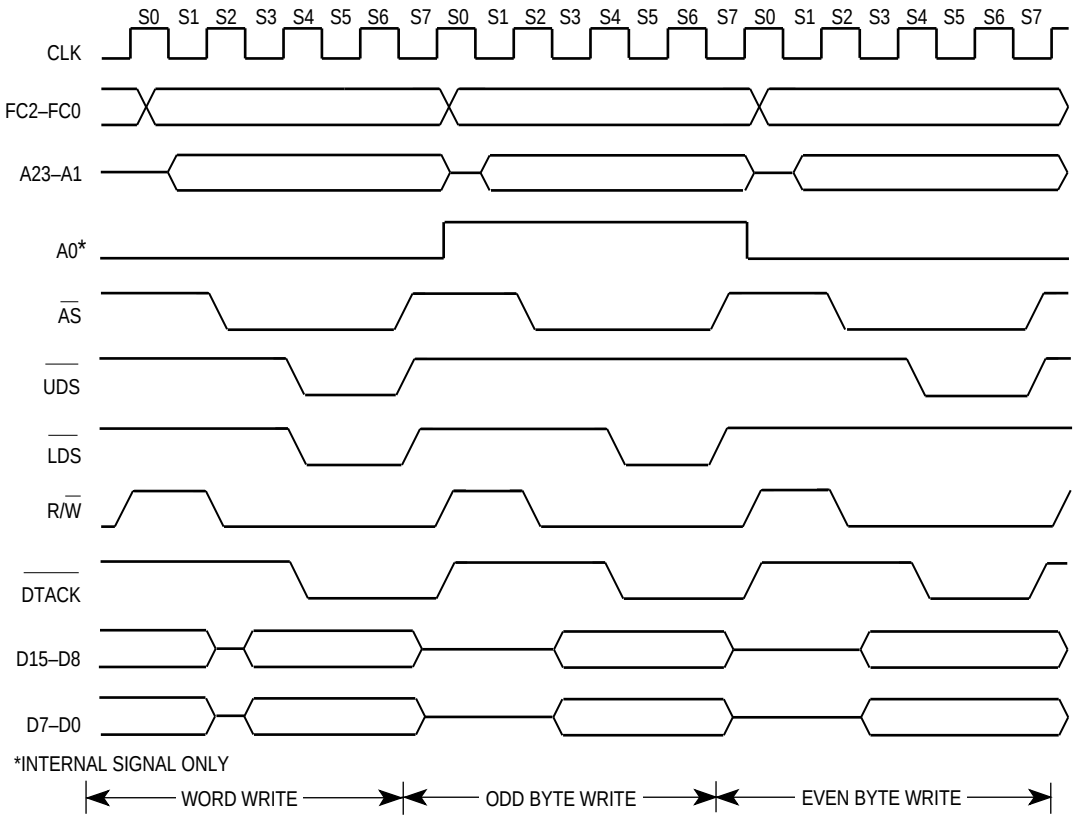


Figure 5-7. Word and Byte Write-Cycle Timing Diagram

The descriptions of the eight states of a write cycle are as follows:

STATE 0 The write cycle starts in S0. The processor places valid function codes on FC2–FC0 and drives R/$\overline{W}$ high (if a preceding write cycle has left R/$\overline{W}$ low).

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

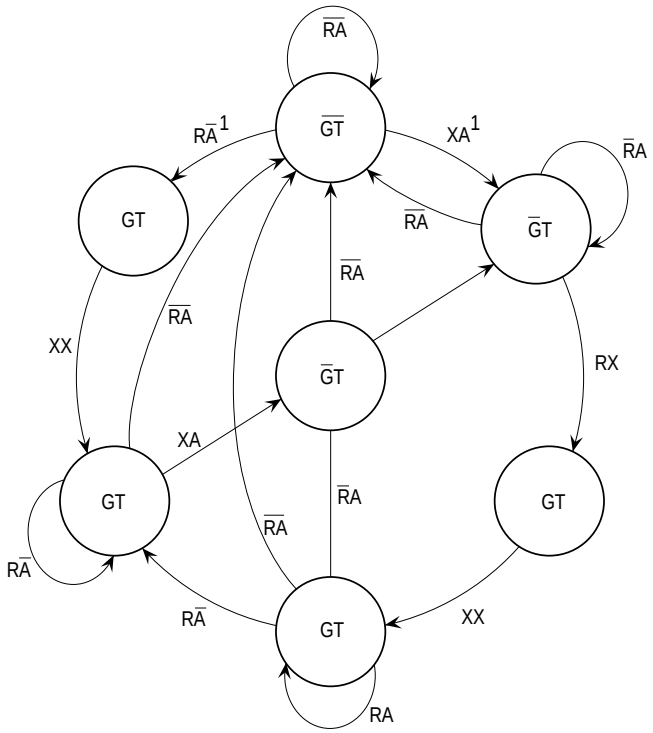
STATE 2 On the rising edge of S2, the processor asserts $\overline{AS}$ and drives R/$\overline{W}$ low.

STATE 3 During S3, the data bus is driven out of the high-impedance state as the data to be written is placed on the bus.

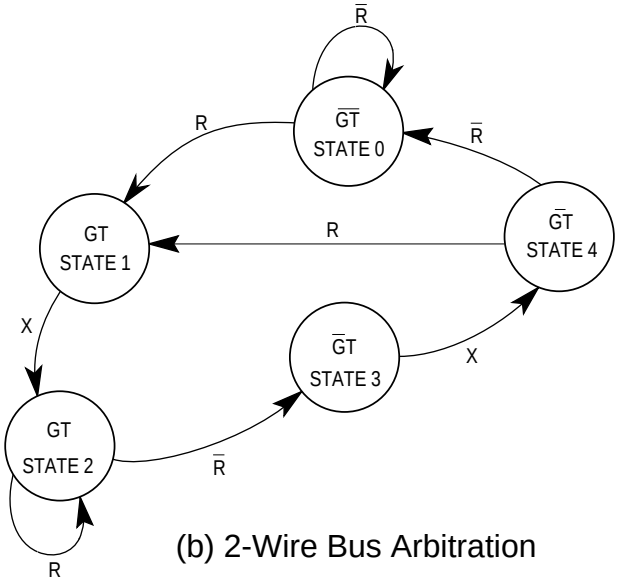
STATE 4 At the rising edge of S4, the processor asserts $\overline{UDS}$, or $\overline{LDS}$. The processor waits for a cycle termination signal ($\overline{DTACK}$ or $\overline{BERR}$) or $\overline{VPA}$, an M6800 peripheral signal. When $\overline{VPA}$ is asserted during S4, the cycle becomes a peripheral cycle (refer to **Appendix B M6800 Peripheral Interface**. If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{DTACK}$ or $\overline{BERR}$ is asserted.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, no bus signals are altered.



(a) 3-Wire Bus Arbitration



(b) 2-Wire Bus Arbitration

R = Bus Request Internal
A = Bus Grant Acknowledge Internal
G = Bus Grant
T = Three-state Control to Bus Control Logic
X = Don't Care

Notes:
1. State machine will not change if the bus is S0 or S1. Refer to **BUS ARBITRATION CONTROL.** 5.2.3.
2. The address bus will be placed in the high-impedance state if T is asserted and $\overline{AS}$ is negated.

Figure 5-18. Bus Arbitration Unit State Diagrams

Figures 5-19, 5-20, and 5-21 applies to all processors using 3-wire bus arbitration. Figures 5-22, 5-23, and 5-24 applies to all processors using 2-wire bus arbitration.

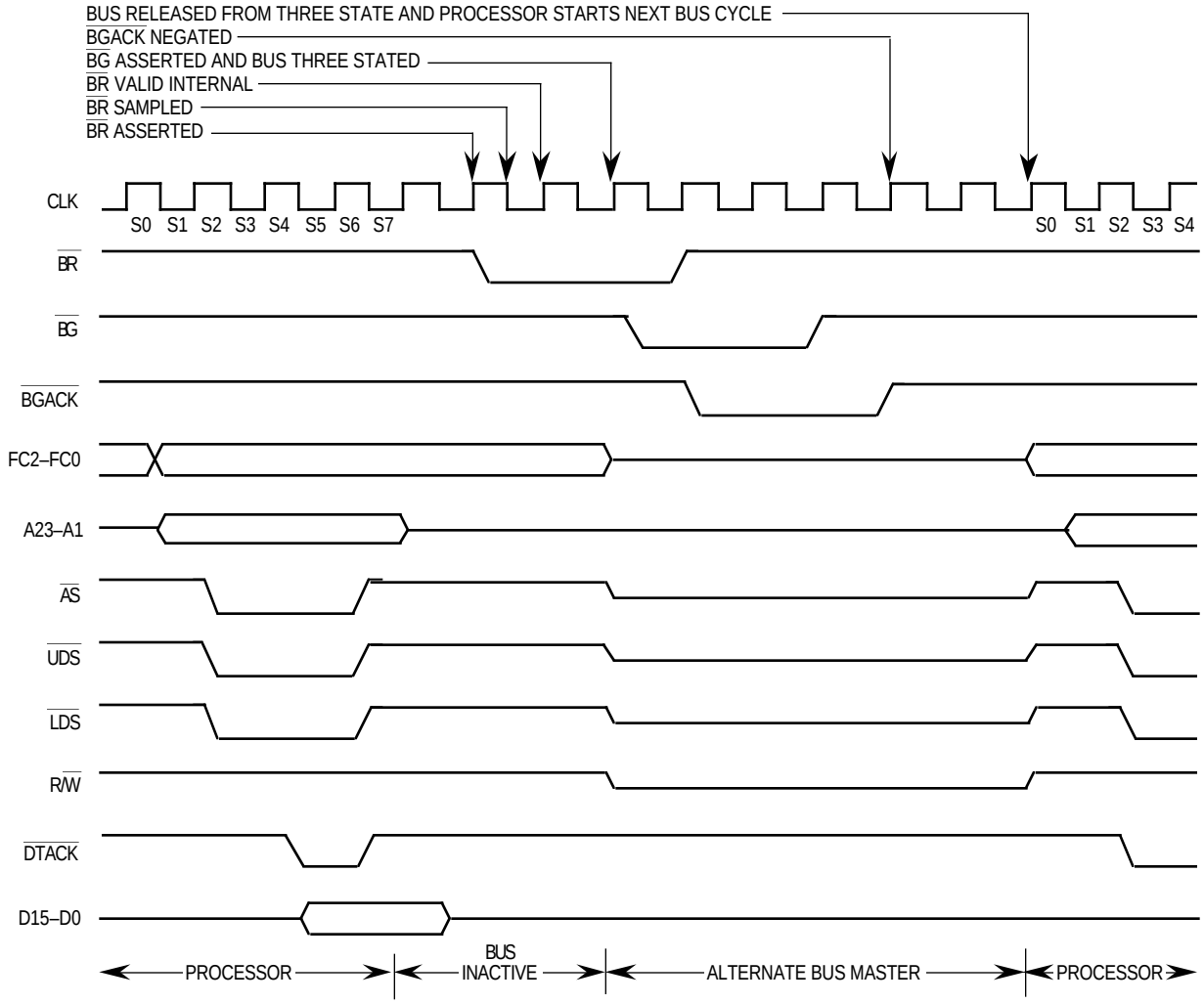


Figure 5-20. 3-Wire Bus Arbitration Timing Diagram—Bus Inactive

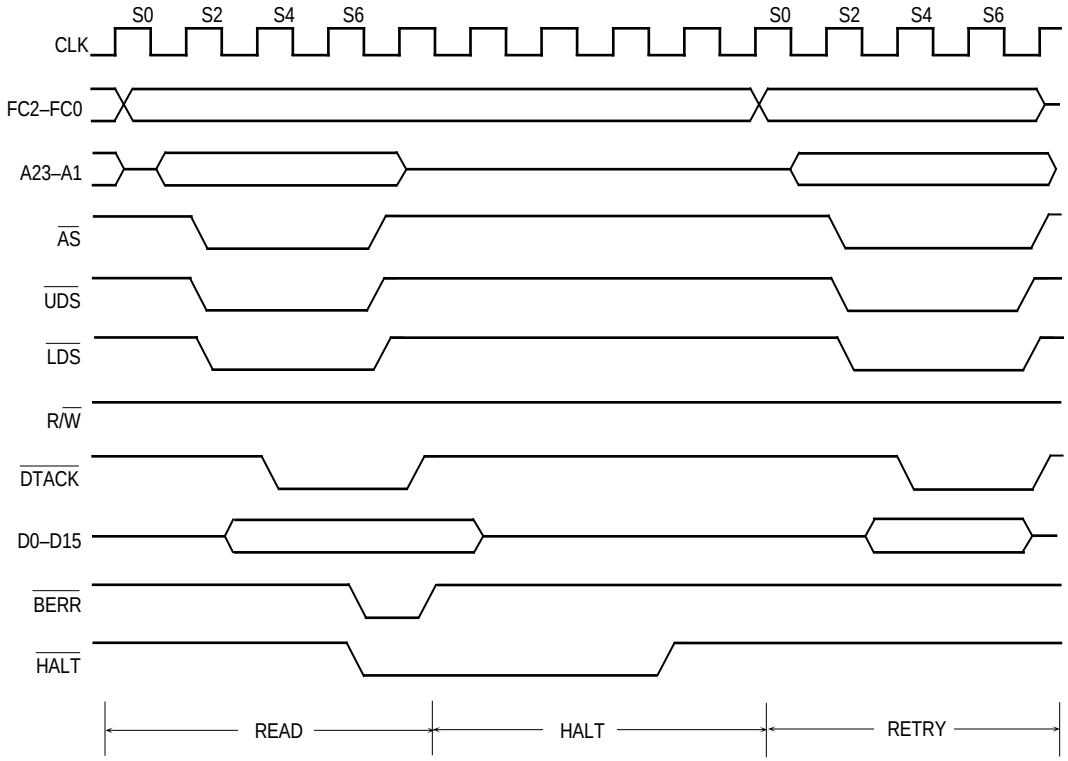


Figure 5-29. Halt Operation Timing Diagram

While the processor is halted, the address bus and the data bus signals are placed in the high-impedance state. Bus arbitration is performed as usual. Should a bus error occur while $\overline{\text{HALT}}$ is asserted, the processor performs the retry operation previously described.

The single-step mode is derived from correctly timed transitions of $\overline{\text{HALT}}$. $\overline{\text{HALT}}$ is negated to allow the processor to begin a bus cycle, then asserted to enter the halt mode when the cycle completes. The single-step mode proceeds through a program one bus cycle at a time for debugging purposes. The halt operation and the hardware trace capability allow tracing of either bus cycles or instructions one at a time. These capabilities and a software debugging package provide total debugging flexibility.

5.4.4 Double Bus Fault

When a bus error exception occurs, the processor begins exception processing by stacking information on the supervisor stack. If another bus error occurs during exception processing (i.e., before execution of another instruction begins) the processor halts and asserts $\overline{\text{HALT}}$. This is called a double bus fault. Only an external reset operation can restart a processor halted due to a double bus fault.

A retry operation does not initiate exception processing; a bus error during a retry operation does not cause a double bus fault. The processor can continue to retry a bus cycle indefinitely if external hardware requests.

6.2 EXCEPTION PROCESSING

The processing of an exception occurs in four steps, with variations for different exception causes:

1. Make a temporary copy of the status register and set the status register for exception processing.
2. Obtain the exception vector.
3. Save the current processor context.
4. Obtain a new context and resume instruction processing.

6.2.1 Exception Vectors

An exception vector is a memory location from which the processor fetches the address of a routine to handle an exception. Each exception type requires a handler routine and a unique vector. All exception vectors are two words in length (see Figure 6-1), except for the reset vector, which is four words long. All exception vectors reside in the supervisor data space, except for the reset vector, which is in the supervisor program space. A vector number is an 8-bit number that is multiplied by four to obtain the offset of an exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. For interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (see Figure 6-2) to the processor on data bus lines D7–D0.

The processor forms the vector offset by left-shifting the vector number two bit positions and zero-filling the upper-order bits to obtain a 32-bit long-word vector offset. In the MC68000, the MC68HC000, MC68HC001, MC68EC000, and the MC68008, this offset is used as the absolute address to obtain the exception vector itself, which is shown in Figure 6-3.

NOTE

In the MC68010, the vector offset is added to the 32-bit vector base register (VBR) to obtain the 32-bit absolute address of the exception vector (see Figure 6-4). Since the VBR is set to zero upon reset, the MC68010 functions identically to the MC68000, MC68HC000, MC68HC001, MC68EC000, and MC68008 until the VBR is changed via the move control register MOVEC instruction.

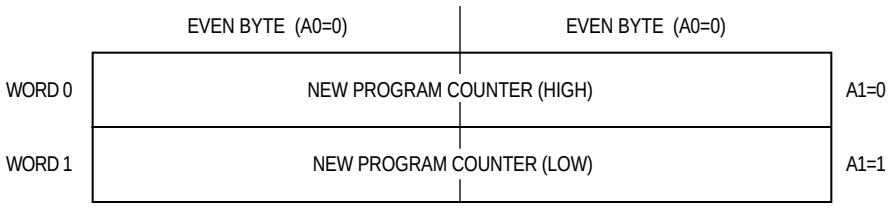


Figure 6-1. Exception Vector Format

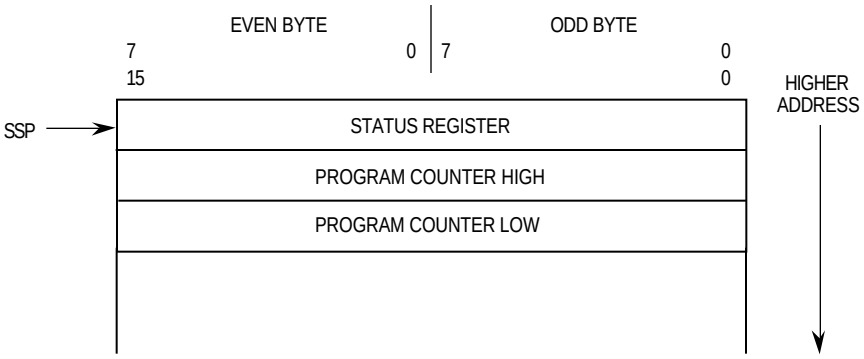


Figure 6-5. Group 1 and 2 Exception Stack Frame (MC68000, MC68HC000, MC68HC001, MC68EC000, and MC68008)

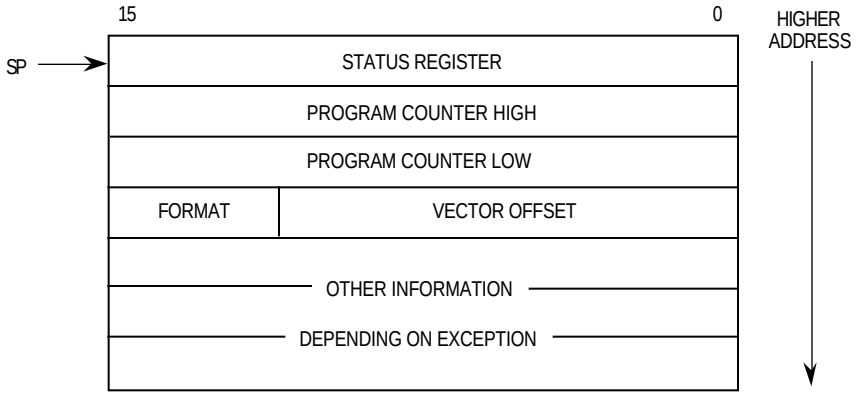


Figure 6-6. MC68010 Stack Frame

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.

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 9-13. Shift/Rotate Instruction Loop Mode Execution Times

		Loop Continued			Loop Terminated					
		Valid Count cc False			Valid Count cc True			Expired Count		
Instruction	Size	(An)	(An)+	–(An)	(An)	(An)+	–(An)	(An)	(An)+	–(An)
ASR, ASL	Word	**18**(1/1)	**18**(1/1)	**20**(1/1)	**24**(3/1)	**24**(3/1)	**26**(3/1)	**22**(3/1)	**22**(3/1)	**24**(3/1)
LSR, LSL	Word	**18**(1/1)	**18**(1/1)	**20**(1/1)	**24**(3/1)	**24**(3/1)	**26**(3/1)	**22**(3/1)	**22**(3/1)	**24**(3/1)
ROR, ROL	Word	**18**(1/1)	**18**(1/1)	**20**(1/1)	**24**(3/1)	**24**(3/1)	**26**(3/1)	**22**(3/1)	**22**(3/1)	**24**(3/1)
ROXR, ROXL	Word	**18**(1/1)	**18**(1/1)	**20**(1/1)	**24**(3/1)	**24**(3/1)	**26**(3/1)	**22**(3/1)	**22**(3/1)	**24**(3/1)

9.7 BIT MANIPULATION INSTRUCTION EXECUTION TIMES

Table 9-14 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 9-14. Bit Manipulation Instruction Execution Times

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

9.8 CONDITIONAL INSTRUCTION EXECUTION TIMES

Table 9-15 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.

Num	Characteristic	8 MHz*		10 MHz*		12.5 MHz*		16.67 MHz 12F		16 MHz		20 MHz**		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
47[5]	Asynchronous Input Setup Time	10	—	10	—	10	—	10	—	5	—	5	—	ns
48[2,3]	$\overline{BERR}$ Asserted to $\overline{DTACK}$ Asserted	20	—	20	—	20	—	10	—	10	—	10	—	ns
48[2,3,5]	$\overline{DTACK}$ Asserted to $\overline{BERR}$ Asserted (MC68010 Only)	—	80	—	55	—	35	—	—	—	—	—	—	ns
49[9]	$\overline{AS}$, $\overline{DS}$, Negated to E Low	-70	70	-55	55	-45	45	-35	35	-35	35	–30	30	ns
50	E Width High	450	—	350	—	280	—	220	—	220	—	190	—	ns
51	E Width Low	700	—	550	—	440	—	340	—	340	—	290	—	ns
53	Data-Out Hold from Clock High	0	—	0	—	0	—	0	—	0	—	0	—	ns
54	E Low to Data-Out Invalid	30	—	20	—	15	—	10	—	10	—	5	—	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	30	—	20	—	10	—	0	—	0	—	0	—	ns
56[4]	$\overline{HALT}$ ($\overline{RESET}$ Pulse Width	10	—	10	—	10	—	10	—	10	—	10	—	clks
57	$\overline{BGACK}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
57A	$\overline{BGACK}$ Negated to FC, $\overline{VMA}$ Driven	1	—	1	—	1	—	1	—	1	—	1	—	clks
58[7]	$\overline{BR}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
58A[7]	$\overline{BR}$ Negated to FC, $\overline{AS}$ Driven	1	—	1	—	1	—	1	—	1	—	1	—	clks

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

** This frequency applies only to MC68HC000 and MC68HC001.

NOTES:

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

10.14 MC68EC000 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (VCC=5.0 VDC ± 5;PC; GND = 0 VDC; $T_A = T_L$ TO T_H; (See Figures 10-12 and 10-13)

Num	Characteristic	8 MHz		10 MHz		12.5 MHz		16.67 MHz		20 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
6	Clock Low to Address Valid	—	35	—	35	—	35	—	30	—	25	ns
6A	Clock High to FC Valid	—	35	—	35	—	35	—	30	0	25	ns
7	Clock High to Address, Data Bus High Impedance (Maximum)	—	55	—	55	—	55	—	50	—	42	ns
8	Clock High to Address, FC Invalid (Minimum)	0	—	0	—	0	—	0	—	0	—	ns
9[1]	Clock High to $\overline{AS}$, $\overline{DS}$ Asserted	3	35	3	35	3	35	3	30	3	25	ns
11[2]	Address Valid to $\overline{AS}$, $\overline{DS}$ Asserted (Read)/$\overline{AS}$ Asserted (Write)	30	—	20	—	15	—	15	—	10	—	ns
11A[2]	FC Valid to $\overline{AS}$, $\overline{DS}$ Asserted (Read)/ $\overline{AS}$ Asserted (Write)	45	—	45	—	45	—	45	—	40	—	ns
12[1]	Clock Low to $\overline{AS}$, $\overline{DS}$ Negated	3	35	3	35	3	35	3	30	3	25	ns
13[2]	$\overline{AS}$, $\overline{DS}$ Negated to Address, FC Invalid	15	—	15	—	15	—	15	—	10	—	ns
14[2]	$\overline{AS}$ (and $\overline{DS}$ Read) Width Asserted	270	—	195	—	160	—	120	—	100	—	ns
14A[2]	$\overline{DS}$ Width Asserted (Write)	140	—	95	—	80	—	60	—	50	—	ns
15[2]	$\overline{AS}$, $\overline{DS}$ Width Negated	150	—	105	—	65	—	60	—	50	—	ns
16	Clock High to Control Bus High Impedance	—	55	—	55	—	55	—	50	—	42	ns
17[2]	$\overline{AS}$, $\overline{DS}$ Negated to R/$\overline{W}$ Invalid	15	—	15	—	15	—	15	—	10	—	ns
18[1]	Clock High to R/$\overline{W}$ High (Read)	0	35	0	35	0	35	0	30	0	25	ns
20[1]	Clock High to R/$\overline{W}$ Low (Write)	0	35	0	35	0	35	0	30	0	25	ns
20A[2,6]	$\overline{AS}$ Asserted to R/$\overline{W}$ Low (Write)	—	10	—	10	—	10	—	10	—	10	ns
21[2]	Address Valid to R/$\overline{W}$ Low (Write)	0	—	0	—	0	—	0	—	0	—	ns
21A[2]	FC Valid to R/$\overline{W}$ Low (Write)	60	—	50	—	30	—	30	—	25	—	ns
22[2]	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write)	80	—	50	—	30	—	30	—	25	—	ns
23	Clock Low to Data-Out Valid (Write)	—	35	—	35	—	35	—	30	—	25	ns
25[2]	$\overline{AS}$, $\overline{DS}$ Negated to Data-Out Invalid (Write)	40	—	30	—	20	—	15	—	10	—	ns
26[2]	Data-Out Valid to $\overline{DS}$ Asserted (Write)	40	—	30	—	20	—	15	—	10	—	ns
27[5]	Data-In Valid to Clock Low (Setup Time on Read)	5	—	5	—	5	—	5	—	5	—	ns
28[2]	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DTACK}$ Negated (Asynchronous Hold)	0	110	0	110	0	110	0	110	0	95	ns
28A	Clock High to $\overline{DTACK}$ Negated	0	110	0	110	0	110	0	110	0	95	ns

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

Num	Characteristic	8 MHz		10 MHz		12.5 MHz		16.67 MHz		20 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
7	Clock High to Address, Data Bus High Impedance (Maximum)	—	55	—	55	—	55	—	50	—	42	ns
16	Clock High to Control Bus High Impedance	—	55	—	55	—	55	—	50	—	42	ns
33	Clock High to $\overline{BG}$ Asserted	—	35	—	35	—	35	0	30	0	25	ns
34	Clock High to $\overline{BG}$ Negated	—	35	—	35	—	35	0	30	0	25	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
36[7]	$\overline{BR}$ Negated to $\overline{BG}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
38	$\overline{BG}$ Asserted to Control, Address, Data Bus High Impedance ($\overline{AS}$ Negated)	—	55	—	55	—	55	—	50	—	42	ns
39	$\overline{BG}$ Width Negated	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	Clks
47	Asynchronous Input Setup Time	5	—	5	—	5	—	5	—	5	—	ns
58[1]	$\overline{BR}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	Clks
58A[1]	$\overline{BR}$ Negated to FC Driven	1	—	1	—	1	—	1	—	1	—	Clks

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

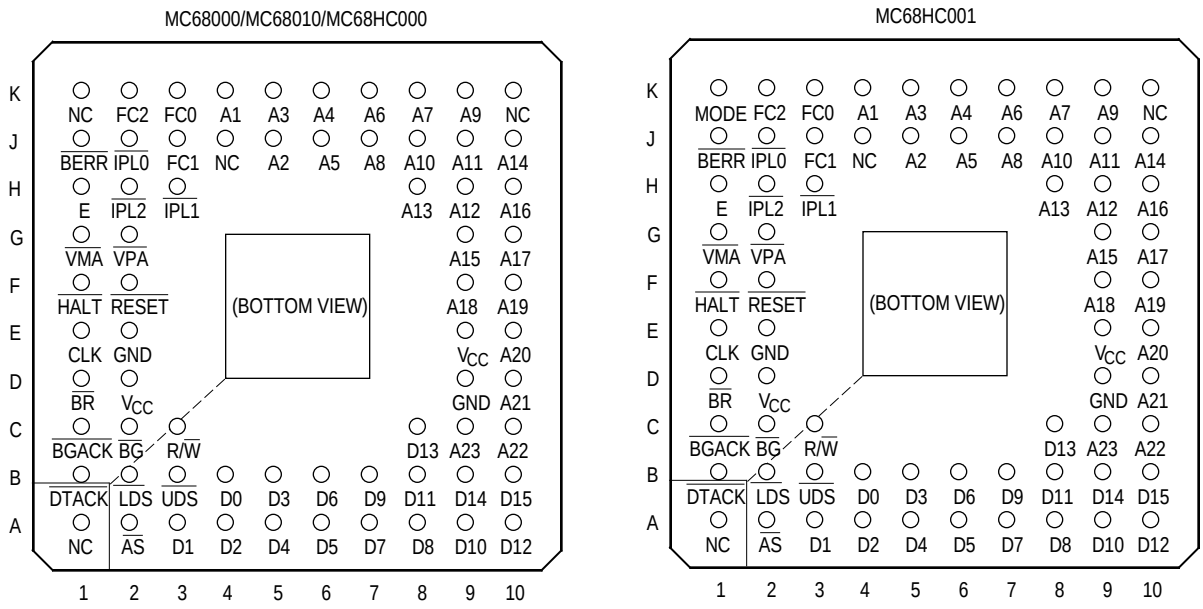


Figure 11-2. 68-Lead Pin Grid Array

Table A-1. MC68010 Loop Mode Instructions

Opcodes	Applicable Addressing Modes
MOVE [BWL]	(Ay) to (Ax) (Ay) to (Ax)+ (Ay) to –(Ax) (Ay)+ to (Ax) (Ay)+ to –(Ax) –(Ay) to (Ax) –(Ay) to (Ax)+ –(Ay) to –(Ax) Ry to (Ax) Ry to (Ax)+
ADD [BWL] AND [BWL] CMP [BWL] OR [BWL] SUB [BWL]	(Ay) to Dx (Ay)+ to Dx –(Ay) to Dx
ADDA [WL] CMPA [WL] SUBA [WL]	(Ay) to Ax –(Ay) to Ax (Ay)+ to Ax
ADD [BWL] AND [BWL] EOR [BWL] OR [BWL] SUB [BWL]	Dx to (Ay) Dx to (Ay)+ Dx to –(Ay)
ABCD [B] ADDX [BWL] SBCD [B] SUBX [BWL]	–(Ay) to –(Ax)
CMP [BWL]	(Ay)+ to (Ax)+
CLR [BWL] NEG [BWL] NEGX [BWL} NOT [BWL] TST [BWL] NBCD [B]	(Ay) (Ay)+ –(Ay)
ASL [W] ASR [W] LSL [W] LSR [W] ROL [W] ROR [W] ROXL [W] ROXR	(Ay) by #1 (Ay)+ by #1 –(Ay) by #1

NOTE: [B, W, or L] indicate an operand size of byte, word, or long word.