Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

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

Applications of Embedded - Microprocessors

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

Details	
Product Status	Obsolete
Core Processor	EC000
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	10MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.21x24.21)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc000ei10r

LIST OF TABLES

LIST OF TABLES (Concluded)

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.

Address Bus (A23–A0)

This 24-bit, unidirectional, three-state bus is capable of addressing 16 Mbytes of data. This bus provides the address for bus operation during all cycles except interrupt acknowledge cycles and breakpoint cycles. During interrupt acknowledge cycles, address lines A1, A2, and A3 provide the level number of the interrupt being acknowledged, and address lines A23–A4 and A0 are driven to logic high. In 16-Bit mode, A0 is always driven high.

MC68008 Address Bus

The unidirectional, three-state buses in the two versions of the **MC68008** differ from each other and from the other processor bus only in the number of address lines and the addressing range. The 20-bit address (A19–A0) of the 48-pin version provides a 1-Mbyte address space; the 52-pin version supports a 22-bit address (A21–A0), extending the address space to 4 Mbytes. During an interrupt acknowledge cycle, the interrupt level number is placed on lines A1, A2, and A3. Lines A0 and A4 through the most significant address line are driven to logic high.

3.2 DATA BUS (D15–D0; MC68008: D7–D0)

This bidirectional, three-state bus is the general-purpose data path. It is 16 bits wide in the all the processors except the **MC68008** which is 8 bits wide. The bus can transfer and accept data of either word or byte length. During an interrupt acknowledge cycle, the external device supplies the vector number on data lines D7–D0. The MC68EC000 and MC68HC001 use D7–D0 in 8-bit mode, and D15–D8 are undefined.

3.3 ASYNCHRONOUS BUS CONTROL

Asynchronous data transfers are controlled by the following signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are described in the following paragraphs.

Address Strobe ($\overline{AS}$).

This three-state signal indicates that the information on the address bus is a valid address.

Read/Write (R/$\overline{W}$).

This three-state signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal relates to the data strobe signals described in the following paragraphs.

Upper And Lower Data Strobes ($\overline{UDS}$, $\overline{LDS}$).

These three-state signals and R/$\overline{W}$ control the flow of data on the data bus. Table 3-1 lists the combinations of these signals and the corresponding data on the bus. When the R/$\overline{W}$ line is high, the processor reads from the data bus. When the R/$\overline{W}$ line is low, the processor drives the data bus. In 8-bit mode, $\overline{UDS}$ is always forced high and the $\overline{LDS}$ signal is used.

3.6 SYSTEM CONTROL

The system control inputs are used to reset the processor, to halt the processor, and to signal a bus error to the processor. The outputs reset the external devices in the system and signal a processor error halt to those devices. The three system control signals are described in the following paragraphs.

Bus Error ($\overline{BERR}$)

This input signal indicates a problem in the current bus cycle. The problem may be the following:

1. No response from a device.
2. No interrupt vector number returned.
3. An illegal access request rejected by a memory management unit.
4. Some other application-dependent error.

Either the processor retries the bus cycle or performs exception processing, as determined by interaction between the bus error signal and the halt signal.

Reset ($\overline{RESET}$)

The external assertion of this bidirectional signal along with the assertion of $\overline{HALT}$ starts a system initialization sequence by resetting the processor. The processor assertion of $\overline{RESET}$ (from executing a RESET instruction) resets all external devices of a system without affecting the internal state of the processor. To reset both the processor and the external devices, the $\overline{RESET}$ and $\overline{HALT}$ input signals must be asserted at the same time.

Halt ($\overline{HALT}$)

An input to this bidirectional signal causes the processor to stop bus activity at the completion of the current bus cycle. This operation places all control signals in the inactive state and places all three-state lines in the high-impedance state (refer to Table 3-4).

When the processor has stopped executing instructions (in the case of a double bus fault condition, for example), the $\overline{HALT}$ line is driven by the processor to indicate the condition to external devices.

Mode (MODE) (MC68HC001/68EC000)

The MODE input selects between the 8-bit and 16-bit operating modes. If this input is grounded at reset, the processor will come out of reset in the 8-bit mode. If this input is tied high or floating at reset, the processor will come out of reset in the 16-bit mode. This input should be changed only at reset and must be stable two clocks after RESET is negated. Changing this input during normal operation may produce unpredictable results.

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.

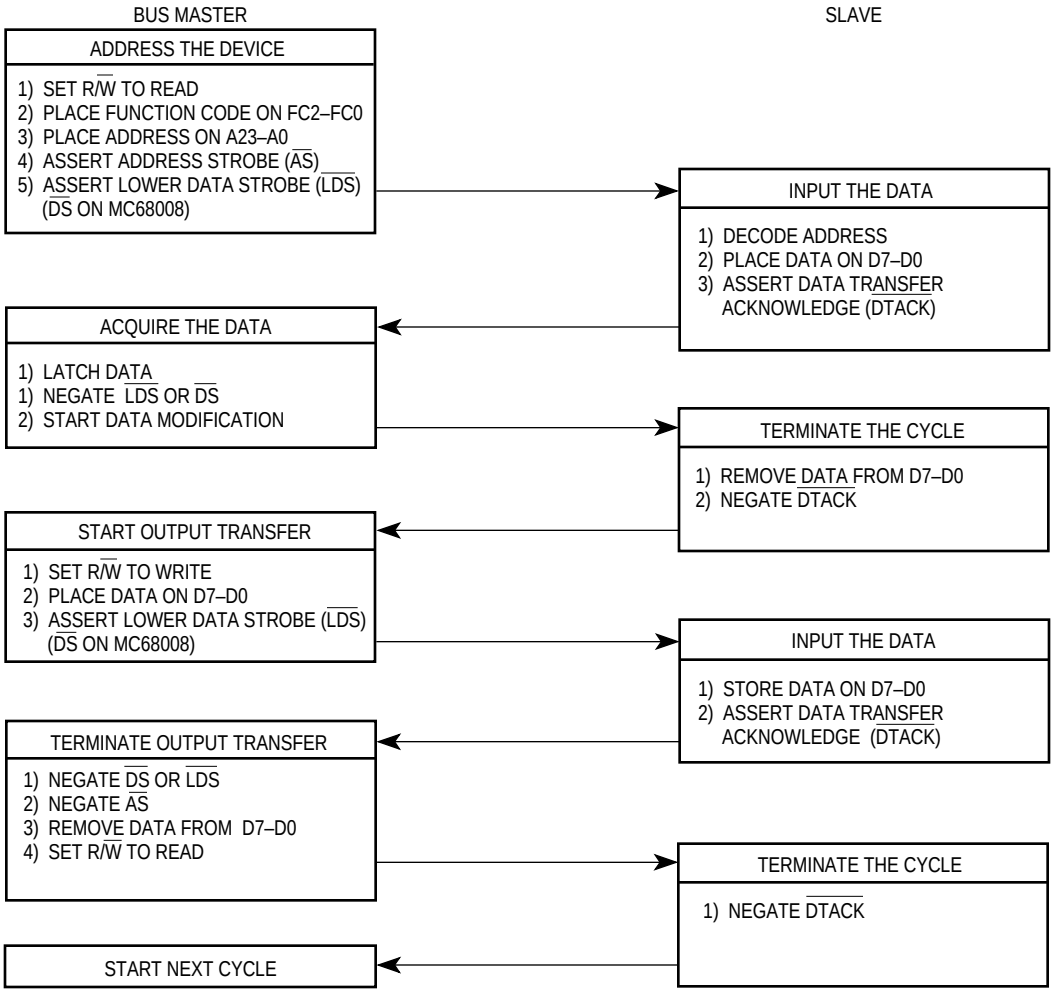


Figure 4-5. Read-Modify-Write Cycle Flowchart

bus request signal. When no acknowledge is received before the bus request signal is negated, the processor continues the use of the bus.

5.2.2 Receiving The Bus Grant

The processor asserts $\overline{BG}$ as soon as possible. Normally, this process immediately follows internal synchronization, except when the processor has made an internal decision to execute the next bus cycle but has not yet asserted $\overline{AS}$ for that cycle. In this case, $\overline{BG}$ is delayed until $\overline{AS}$ is asserted to indicate to external devices that a bus cycle is in progress.

$\overline{BG}$ can be routed through a daisy-chained network or through a specific priority-encoded network. Any method of external arbitration that observes the protocol can be used.

5.2.3 Acknowledgment Of Mastership (3-Wire Bus Arbitration Only)

Upon receiving $\overline{BG}$, the requesting device waits until $\overline{AS}$, $\overline{DTACK}$, and $\overline{BGACK}$ are negated before asserting $\overline{BGACK}$. The negation of $\overline{AS}$ indicates that the previous bus master has completed its cycle. (No device is allowed to assume bus mastership while $\overline{AS}$ is asserted.) The negation of $\overline{BGACK}$ indicates that the previous master has released the bus. The negation of $\overline{DTACK}$ indicates that the previous slave has terminated the connection to the previous master. (In some applications, $\overline{DTACK}$ might not be included in this function; general-purpose devices would be connected using $\overline{AS}$ only.) When $\overline{BGACK}$ is asserted, the asserting device is bus master until it negates $\overline{BGACK}$. $\overline{BGACK}$ should not be negated until after the bus cycle(s) is complete. A device relinquishes control of the bus by negating $\overline{BGACK}$.

The bus request from the granted device should be negated after $\overline{BGACK}$ is asserted. If another bus request is pending, $\overline{BG}$ is reasserted within a few clocks, as described in **5.3 Bus Arbitration Control**. The processor does not perform any external bus cycles before reasserting $\overline{BG}$.

5.3 BUS ARBITRATION CONTROL

All asynchronous bus arbitration signals to the processor are synchronized before being used internally. As shown in Figure 5-17, synchronization requires a maximum of one cycle of the system clock, assuming that the asynchronous input setup time (#47, defined in **Section 10 Electrical Characteristic**) has been met. The input asynchronous signal is sampled on the falling edge of the clock and is valid internally after the next falling edge.

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

5.5 RESET OPERATION

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

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

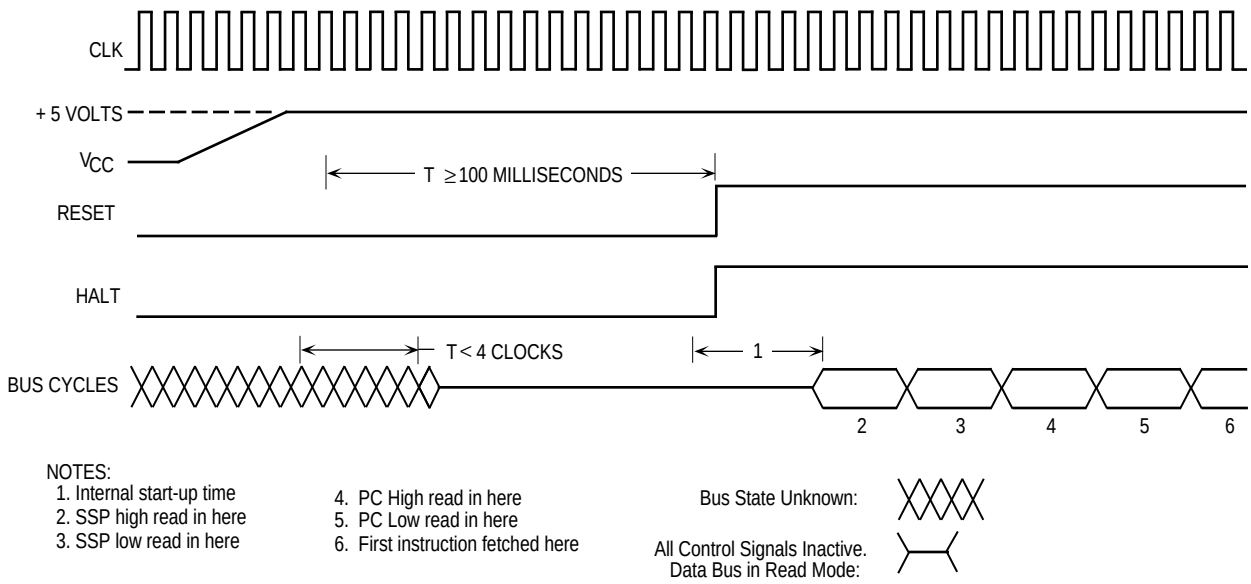


Figure 5-30. Reset Operation Timing Diagram

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

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

On the rising edge of the clock, at the end of S7 (which may be the start of S0 for the next bus cycle), the processor places the address bus in the high-impedance state. During a write cycle, the processor also places the data bus in the high-impedance state and drives R/$\overline{W}$ high. External logic circuitry should respond to the negation of the $\overline{AS}$ and $\overline{UDS}$, $\overline{LDS}$, and/or $\overline{DS}$ by negating $\overline{DTACK}$ and/or $\overline{BERR}$. Parameter #28 is the hold time for $\overline{DTACK}$, and parameter #30 is the hold time for $\overline{BERR}$.

Figure 5-35 shows a synchronous read cycle and the important timing parameters that apply. The timing for a synchronous read cycle, including relevant timing parameters, is shown in Figure 5-36.

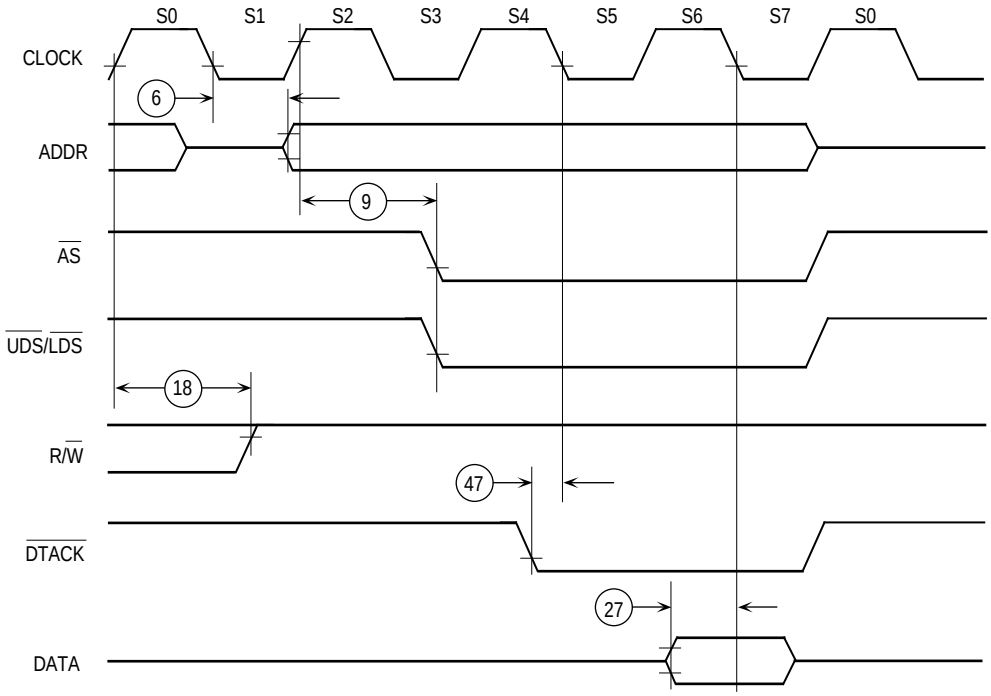


Figure 5-35. Synchronous Read Cycle

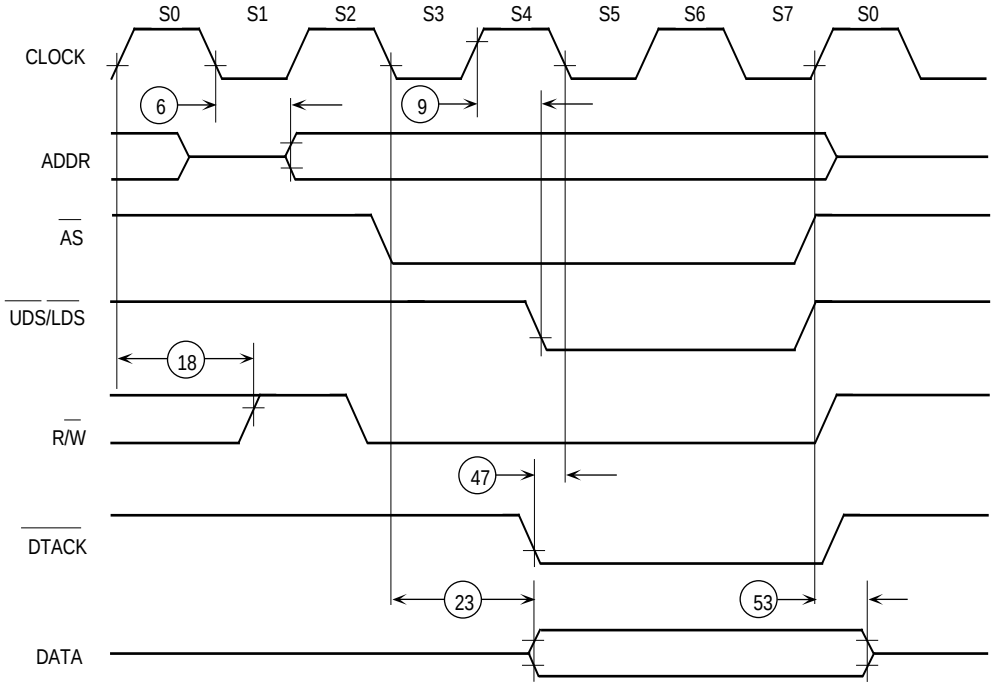


Figure 5-36. Synchronous Write Cycle

A key consideration when designing in a synchronous environment is the timing for the assertion of $\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$ by an external device. To properly use external inputs, the processor must synchronize these signals to the internal clock. The processor must sample the external signal, which has no defined phase relationship to the CPU clock, which may be changing at sampling time, and must determine whether to consider the signal high or low during the succeeding clock period. Successful synchronization requires that the internal machine receives a valid logic level (not a metastable signal), whether the input is high, low, or in transition. Metastable signals propagating through synchronous machines can produce unpredictable operation.

Figure 5-37 is a conceptual representation of the input synchronizers used by the M68000 Family processors. The input latches allow the input to propagate through to the output when E is high. When low, E latches the input. The three latches require one cycle of CLK to synchronize an external signal. The high-gain characteristics of the devices comprising the latches quickly resolve a marginal signal into a valid state.

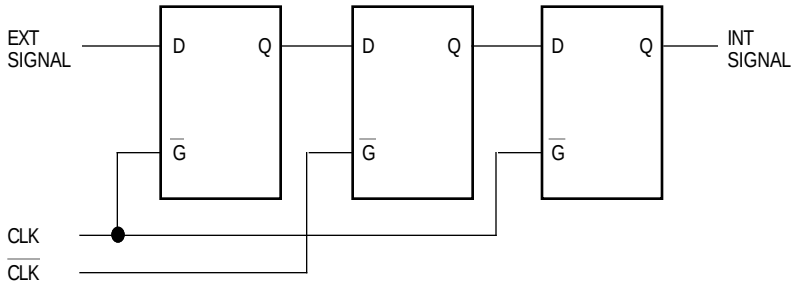


Figure 5-37. Input Synchronizers

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

7.7 BIT MANIPULATION INSTRUCTION EXECUTION TIMES

Table 7-9 lists the timing data for the bit manipulation instructions. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format. The number of clock periods, the number of read cycles, and the number of write cycles, respectively, must be added to those of the effective address calculation where indicated by a plus sign (+).

Table 7-9. Bit Manipulation Instruction Execution Times

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

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

7.8 CONDITIONAL INSTRUCTION EXECUTION TIMES

Table 7-10 lists the timing data for the conditional instructions. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format. The number of clock periods, the number of read cycles, and the number of write cycles, respectively, must be added to those of the effective address calculation where indicated by a plus sign (+).

Table 7-10. Conditional Instruction Execution Times

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

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

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 9-11. Single Operand 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)
CLR	Byte, Word	**10**(0/1)	**10**(0/1)	**12**(0/1)	**18**(2/1)	**18**(2/1)	**20**(2/0)	**16**(2/1)	**16**(2/1)	**18**(2/1)
	Long	**14**(0/2)	**14**(0/2)	**16**(0/2)	**22**(2/2)	**22**(2/2)	**24**(2/2)	**20**(2/2)	**20**(2/2)	**22**(2/2)
NBCD	Byte	**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)
NEG	Byte, Word	**16**(1/1)	**16**(1/1)	**18**(2/2)	**22**(3/1)	**22**(3/1)	**24**(3/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)
	Long	**24**(2/2)	**24**(2/2)	**26**(2/2)	**30**(4/2)	**30**(4/2)	**32**(4/2)	**28**(4/2)	**28**(4/2)	**30**(4/2)
NEGX	Byte, Word	**16**(1/1)	**16**(1/1)	**18**(2/2)	**22**(3/1)	**22**(3/1)	**24**(3/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)
	Long	**24**(2/2)	**24**(2/2)	**26**(2/2)	**30**(4/2)	**30**(4/2)	**32**(4/2)	**28**(4/2)	**28**(4/2)	**30**(4/2)
NOT	Byte, Word	**16**(1/1)	**16**(1/1)	**18**(2/2)	**22**(3/1)	**22**(3/1)	**24**(3/1)	**20**(3/1)	**20**(3/1)	**22**(3/1)
	Long	**24**(2/2)	**24**(2/2)	**26**(2/2)	**30**(4/2)	**30**(4/2)	**32**(4/2)	**28**(4/2)	**28**(4/2)	**30**(4/2)
TST	Byte, Word	**12**(1/0)	**12**(1/0)	**14**(1/0)	**18**(3/0)	**18**(3/0)	**20**(3/0)	**16**(3/0)	**16**(3/0)	**18**(3/0)
	Long	**18**(2/0)	**18**(2/0)	**20**(2/0)	**24**(4/0)	**24**(4/0)	**26**(4/0)	**20**(4/0)	**20**(4/0)	**22**(4/0)

9.6 SHIFT/ROTATE INSTRUCTION EXECUTION TIMES

Tables 9-12 and 9-13 list the timing data for the shift and rotate instructions. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format. The number of clock periods, the number of read cycles, and the number of write cycles, respectively, must be added to those of the effective address calculation where indicated by a plus sign (+).

Table 9-12. Shift/Rotate Instruction Execution Times

Instruction	Size	Register	Memory*
ASR, ASL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
LSR, LSL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
ROR, ROL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
ROXR, ROXL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—

+Add effective address calculation time.
n is the shift or rotate count.
* Word only.

9.12 EXCEPTION PROCESSING EXECUTION TIMES

Table 9-19 lists the timing data for exception processing. The numbers of clock periods include the times for all stacking, the vector fetch, and the fetch of the first instruction of the handler routine. 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 9-19. Exception Processing Execution Times

Exception	
Address Error	**126**(4/26)
Breakpoint Instruction*	**45**(5/4)
Bus Error	**126**(4/26)
CHK Instruction**	**44**(5/4)+
Divide By Zero	**42**(5/4)+
Illegal Instruction	**38**(5/4)
Interrupt*	**46**(5/4)
MOVEC, Illegal Control Register**	**46**(5/4)
Privilege Violation	**38**(5/4)
Reset***	**40**(6/0)
RTE, Illegal Format	**50**(7/4)
RTE, Illegal Revision	**70**(12/4)
Trace	**38**(4/4)
TRAP Instruction	**38**(4/4)
TRAPV Instruction	**38**(5/4)

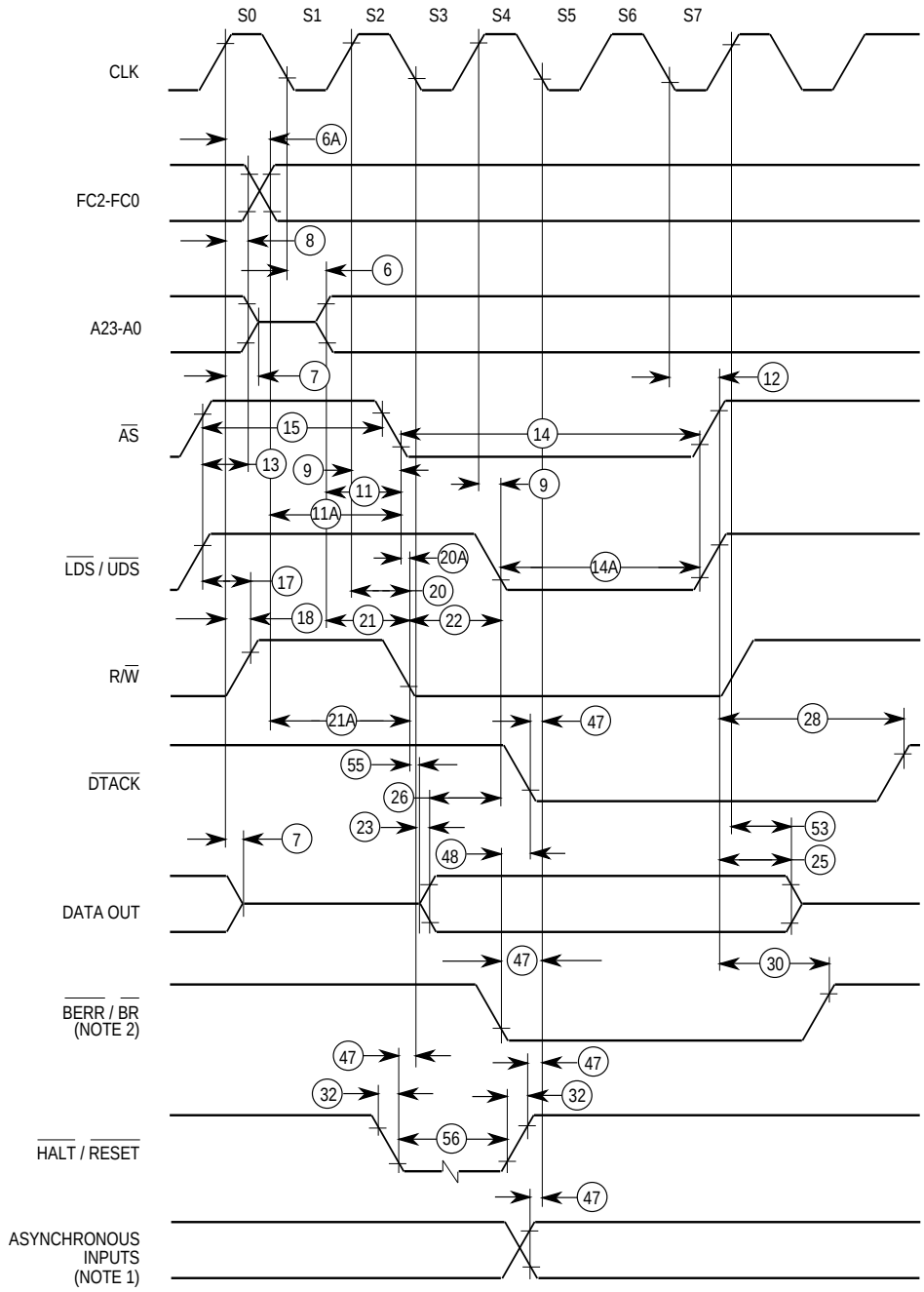
\+ Add effective address calculation time.
* The interrupt acknowledge and breakpoint cycles are assumed to take four clock periods.
** Indicates maximum value.
*** Indicates the time from when $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are first sampled as negated to when instruction execution starts.

10.13 MC68EC000 DC ELECTRICAL SPECIFICATIONS (VCC=5.0 VDC ± 5;PC; GND=0 VDC; $T_A = T_L$ TO T_H)

Characteristic		Symbol	Min	Max	Unit
Input High Voltage		V_{IH}	2.0	V_{CC}	V
Input Low Voltage		V_{IL}	GND–0.3	0.8	V
Input Leakage Current @5.25 V	$\overline{BERR}$, $\overline{BR}$, $\overline{DTACK}$, CLK, $\overline{IPL2}$–$\overline{IPL0}$, $\overline{AVEC}$ MODE, $\overline{HALT}$, $\overline{RESET}$	I_{in}	— —	2.5 20	µA
Three-State (Off State) Input Current @2.4 V/0.4 V	AS, A23–A0, D15–D0, FC2–FC0, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$	I_{TSI}	—	20	µA
Output High Voltage (I_{OH}=–400 µA)	AS, A23–A0, $\overline{BG}$, D15–D0, FC2–FC0, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$	V_{OH}	VCC –0.75	—	V
Output Low Voltage		V_{OL}			V
(IOL = 1.6 mA)	$\overline{HALT}$		—	0.5	
(IOL = 3.2 mA)	A23–A0, $\overline{BG}$, FC2–FC0		—	0.5	
(IOL = 5.0 mA)	$\overline{RESET}$		—	0.5	
(IOL = 5.3 mA)	$\overline{AS}$, D15–D0, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$		—	0.5	
Current Dissipation*	f=8 MHz	I_D	—	25	mA
	f=10 MHz		—	30	
	f=12.5 MHz		—	35	
	f=16.67 MHz		—	50	
	f= 20 MHz		—	70	
Power Dissipation	f=8 MHz	P_D	—	0.13	W
	f=10 MHz		—	0.16	
	f=12.5 MHz		—	0.19	
	f=16.67 MHz		—	0.26	
	f=20 MHz		—	0.38	
Capacitance (Vin=0 V, TA=25°C, Frequency=1 MHz)**		C_{in}	—	20.0	pF
Load Capacitance	$\overline{HALT}$	C_L	—	70	pF
	All Others		—	130	

*Currents listed are with no loading.

**Capacitance is periodically sampled rather than 100% tested.



NOTES:
1. 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.
2. Because of loading variations, R/W may be valid after AS even though both are initiated by the rising edge of S2 (specification #20A).

Figure 10-13. MC68EC000 Write Cycle Timing Diagram

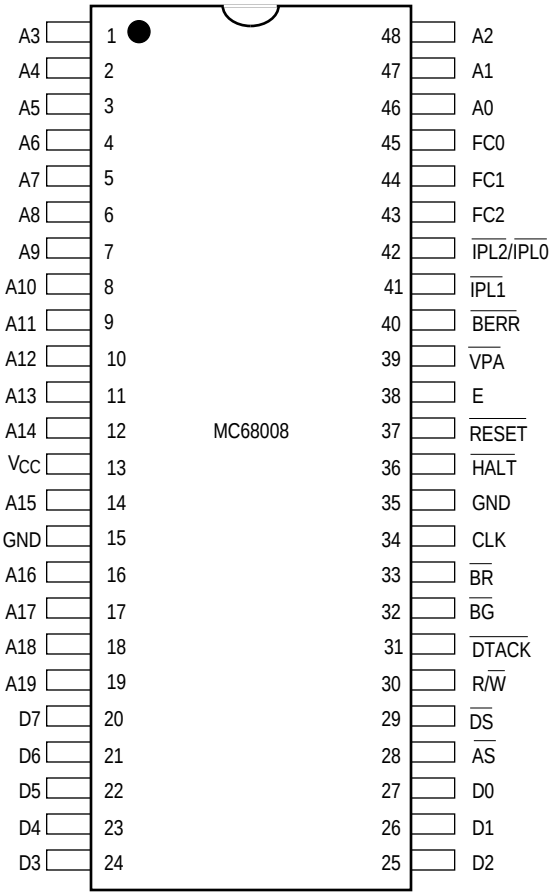


Figure 11-5. 48-Pin Dual In Line

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.