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	68020
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
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	132-BCQFP
Supplier Device Package	132-CQFP (24x24)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68020fe25e

TABLE OF CONTENTS (Continued)

Table 1-1. Addressing Modes

Addressing Modes	Syntax
Register Direct	
Data	Dn
Address	An
Register Indirect	
Address	(An)
Address with Postincrement	(An)+
Address with Predecrement	–(An)
Address with Displacement	(d_{16}, An)
Address Register Indirect with Index	
8-Bit Displacement	(d_8, An, Xn)
Base Displacement	(bd, An, Xn)
Memory Indirect	
Postindexed	([bd, An], Xn, od)
Preindexed	([bd, An, Xn], od)
PC Indirect with Displacement	(d_{16}, PC)
PC Indirect with Index	
8-Bit Displacement	(d_8, PC, Xn)
Base Displacement	(bd, PC, Xn)
PC Indirect	
Postindexed	([bd, PC], Xn, od)
Preindexed	([bd, PC, Xn], od)
Absolute Data Addressing	
Short	(xxx).W
Long	(xxx).L
Immediate	#<data>

NOTE:

Dn = Data Register, D7–D0

An = Address Register, A7–A0

d_8, d_{16} = A twos complement or sign-extended displacement added as part of the effective address calculation; size is 8 (d_8) or 16 (d_{16}) bits; when omitted, assemblers use a value of zero.

Xn = Address or data register used as an index register; form is Xn.SIZE∗SCALE, where SIZE is .W or .L (indicates index register size) and SCALE is 1, 2, 4, or 8 (index register is multiplied by SCALE); use of SIZE and/or SCALE is optional.

bd = A twos-complement base displacement; when present, size can be 16 or 32 bits.

od = Outer displacement added as part of effective address calculation after any memory indirection; use is optional with a size of 16 or 32 bits.

PC = Program Counter

<data> = Immediate value of 8, 16, or 32 bits

( ) = Effective Address

[ ] = Use as indirect access to long-word address.

2.1 PRIVILEGE LEVELS

The processor operates at one of two privilege levels: the user level or the supervisor level. The supervisor level has higher privileges than the user level. Not all processor or coprocessor instructions are permitted to execute at the lower privileged user level, but all are available at the supervisor level. This arrangement allows a separation of supervisor and user so the supervisor can protect system resources from uncontrolled access. The S-bit in the SR is used to select either the user or supervisor privilege level and either the USP or an SSP for stack operations. The processor identifies a bus access (supervisor or user mode) via the function codes so that differentiation between supervisor level and user level can be maintained.

In many systems, the majority of programs execute at the user level. User programs can access only their own code and data areas and can be restricted from accessing other information. The operating system typically executes at the supervisor privilege level. It has access to all resources, performs the overhead tasks for the user-level programs, and coordinates user-level program activities.

2.1.1 Supervisor Privilege Level

The supervisor level is the higher privilege level. The privilege level is determined by the S-bit of the SR; if the S-bit is set, the supervisor privilege level applies, and all instructions are executable. The bus cycles for instructions executed at the supervisor level are normally classified as supervisor references, and the values of the FC2–FC0 signals refer to supervisor address spaces.

In a multitasking operating system, it is more efficient to have a supervisor stack space associated with each user task and a separate stack space for interrupt-associated tasks. The MC68020/EC020 provides two supervisor stacks, master and interrupt; the M bit of the SR selects which of the two is active. When the M-bit is set, references to the SSP implicitly or to address register seven (A7) explicitly, access the MSP. The operating system sets the MSP for each task to point to a task-related area of supervisor data space. This arrangement separates task-related supervisor activity from asynchronous, I/O-related supervisor tasks that may be only coincidental to the currently executing task. The MSP can separately maintain task control information for each currently executing user task, and the software updates the MSP when a task switch is performed, providing an efficient means for transferring task-related stack items. The other supervisor stack pointer, the ISP, can be used for interrupt control information and workspace area as interrupt handling routines require.

When the M-bit is clear, the MC68020/EC020 is in the interrupt mode of the supervisor privilege level, and operation is the same as supervisor mode in the MC68000, MC68HC001, MC68008, and MC68010. (The processor is in this mode after a reset operation.) All SSP references access the ISP in this mode.

on top of the stack was generated by an interrupt, trap, or instruction exception, the RTE instruction restores the SR and PC to the values saved on the supervisor stack. The processor then continues execution at the restored PC address and at the privilege level determined by the S-bit of the restored SR. If the frame on top of the stack was generated by a bus fault (bus error or address error exception), the RTE instruction restores the entire saved processor state from the stack.

2.2 ADDRESS SPACE TYPES

The processor specifies a target address space for every bus cycle with the FC2–FC0 signals according to the type of access required. In addition to distinguishing between supervisor/user and program/data, the processor can identify special processor cycles, such as the interrupt acknowledge cycle, and the memory management unit can control accesses and translate addresses appropriately. Table 2-1 lists the types of accesses defined for the MC68020/EC020 and the corresponding values of the FC2–FC0 signals.

Table 2-1. Address Space Encodings

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

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

The memory locations of user program and data accesses are not predefined; neither are the locations of supervisor data space. During reset, the first two long words beginning at memory location zero in the supervisor program space are used for processor initialization. No other memory locations are explicitly defined by the MC68020/EC020.

A function code of $7 selects the CPU address space. This is a special address space that does not contain instructions or operands but is reserved for special processor functions. The processor uses accesses in this space to communicate with external devices for special purposes. For example, all M68000 processors use the CPU space for interrupt acknowledge cycles. The MC68020/EC020 also generate CPU space accesses for breakpoint acknowledge and coprocessor operations.

Supervisor programs can use the MOVES instruction to access all address spaces, including the user spaces and the CPU address space. Although the MOVES instruction can be used to generate CPU space cycles, this may interfere with proper system operation. Thus, the use of MOVES to access the CPU space should be done with caution.

SECTION 4
ON-CHIP CACHE MEMORY

The MC68020/EC020 incorporates an on-chip cache memory as a means of improving performance. The cache is implemented as a CPU instruction cache and is used to store the instruction stream prefetch accesses from the main memory.

An increase in instruction throughput results when instruction words required by a program are available in the on-chip cache and the time required to access them on the external bus is eliminated. In systems with more than one bus master (e.g., a processor and a DMA device), reduced external bus activity increases overall performance by increasing the availability of the bus for use by external devices without degrading the performance of the MC68020/EC020.

4.1 ON-CHIP CACHE ORGANIZATION AND OPERATION

The MC68020/EC020 on-chip instruction cache is a direct-mapped cache of 64 long-word entries. Each cache entry consists of a tag field (A31–A8 and FC2), one valid bit, and 32 bits (two words) of instruction data. Figure 4-1 shows a block diagram of the on-chip cache organization.

Externally, the MC68EC020 does not use the upper eight bits of the address (A31–A24), and addresses $FF000000 and $00000000 from the MC68EC020 appear the same. However, the MC68EC020 does use A31–A24 internally in the instruction cache address tag, and addresses $FF000000 and $00000000 appear different in the MC68EC020 instruction cache. The MC68020, MC68030/EC030, and MC68040/EC040 use all 32 bits of the address externally. To maintain object-code upgrade compatibility when designing with the MC68EC020, the upper eight bits should be considered part of the address when assigning address spaces in hardware.

When enabled, the MC68020/EC020 instruction cache is used to store instruction prefetches (instruction words and extension words) as they are requested by the CPU. Instruction prefetches are normally requested from sequential memory addresses except when a change of program flow occurs (e.g., a branch taken) or when an instruction is executed that can modify the SR. In these cases, the instruction pipe is automatically flushed and refilled.

SECTION 5
BUS OPERATION

This section provides a functional description of the bus, the signals that control it, and the bus cycles provided for data transfer operations. It also describes the error and halt conditions, bus arbitration, and reset operation. Operation of the bus is the same whether the processor or an external device is the bus master; the names and descriptions of bus cycles are from the point of view of the bus master. For exact timing specifications, refer to **Section 10 Electrical Characteristics**.

The MC68020/EC020 architecture supports byte, word, and long-word operands, allowing access to 8-, 16-, and 32-bit data ports through the use of asynchronous cycles controlled by the $\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$ input signals.

The MC68020/EC020 allows byte, word, and long-word operands to be located in memory on any byte boundary. For a misaligned transfer, more than one bus cycle may be required to complete the transfer, regardless of port size. For a port less than 32 bits wide, multiple bus cycles may be required for an operand transfer due to either misalignment or a port width smaller than the operand size. Instruction words and their associated extension words must be aligned on word boundaries. The user should be aware that misalignment of word or long-word operands can cause the MC68020/EC020 to perform multiple bus cycles for the operand transfer; therefore, processor performance is optimized if word and long-word memory operands are aligned on word or long-word boundaries, respectively.

5.1 BUS TRANSFER SIGNALS

The bus transfers information between the MC68020/EC020 and an external memory, coprocessor, or peripheral device. External devices can accept or provide 8 bits, 16 bits, or 32 bits in parallel and must follow the handshake protocol described in this section. The maximum number of bits accepted or provided during a bus transfer is defined as the port width. The MC68020/EC020 contains an address bus that specifies the address for the transfer and a data bus that transfers the data. Control signals indicate the beginning of the cycle, the address space and size of the transfer, and the type of cycle. The selected device then controls the length of the cycle with the signal(s) used to terminate the cycle. Strobe signals, one for the address bus and another for the data bus, indicate the validity of the address and provide timing information for the data.

The bus operates in an asynchronous mode for any port width. The bus and control input signals are internally synchronized to the MC68020/EC020 clock, introducing a delay. This delay is the time period required for the MC68020/EC020 to sample an input signal, synchronize the input to the internal clocks of the processor, and determine whether the

Figure 5-7 shows a word write to an 8-bit bus port. Like the preceding example, this example requires two bus cycles. Each bus cycle transfers a single byte. SIZ1 and SIZ0 for the first cycle specify two bytes; for the second cycle, one byte. Figure 5-8 shows the associated bus transfer signal timing.

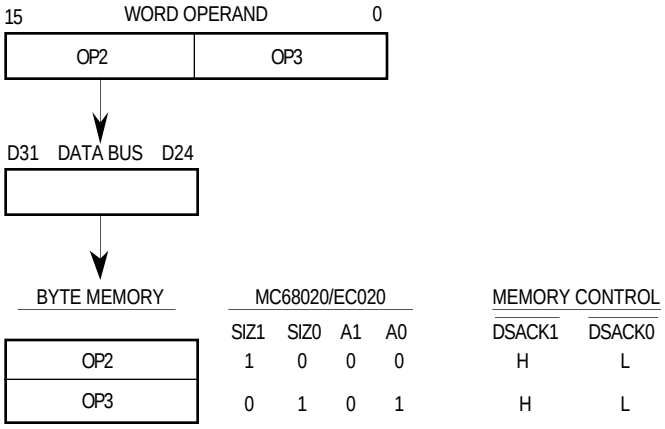


Figure 5-7. Word Operand Write to Byte Port Example

5.2.2 Misaligned Operands

Since operands may reside at any byte boundary, they may be misaligned. A byte operand is properly aligned at any address; a word operand is misaligned at an odd address; a long word is misaligned at an address that is not evenly divisible by four. The MC68000, MC68008, and MC68010 implementations allow long-word transfers on odd-word boundaries but force exceptions if word or long-word operand transfers are attempted at odd-byte addresses. Although the MC68020/EC020 does not enforce any alignment restrictions for data operands (including PC relative data addresses), some performance degradation occurs when additional bus cycles are required for long-word or word operands that are misaligned. For maximum performance, data items should be aligned on their natural boundaries. All instruction words and extension words must reside on word boundaries. Attempting to prefetch an instruction word at an odd address causes an address error exception.

Figure 5-9 shows the transfer (write) of a long-word operand to an odd address in word-organized memory, which requires three bus cycles. For the first cycle, SIZ1 and SIZ0 specify a long-word transfer, and A2–A0 = 001. Since the port width is 16 bits, only the first byte of the long word is transferred. The slave device latches the byte and acknowledges the data transfer, indicating that the port is 16 bits wide. When the processor starts the second cycle, SIZ1 and SIZ0 specify that three bytes remain to be transferred with A2–A0 = 010. The next two bytes are transferred during this cycle. The processor then initiates the third cycle, with SIZ1 and SIZ0 indicating one byte remaining to be transferred with A2–A0 = 100. The port latches the final byte, and the operation is complete. Figure 5-10 shows the associated bus transfer signal timing. Figure 5-11 shows the equivalent operation for a data read cycle.

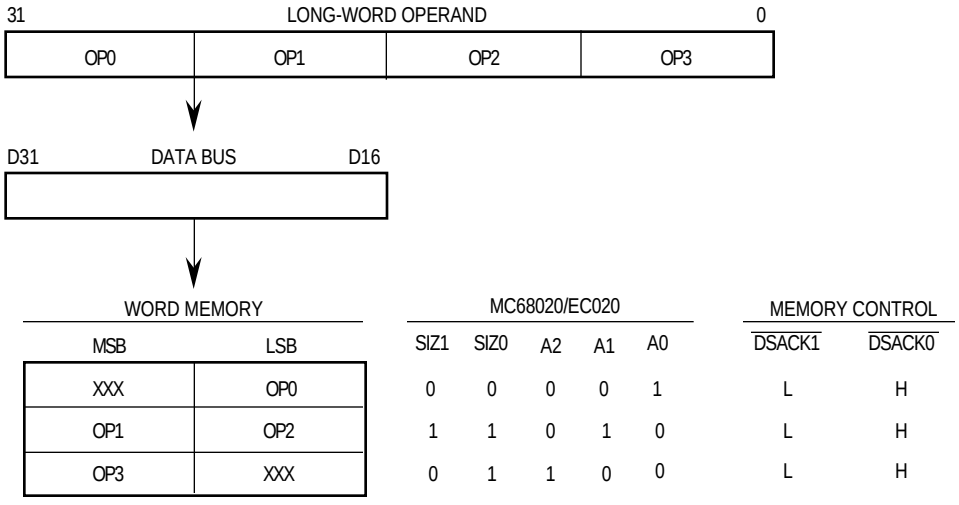


Figure 5-9. Misaligned Long-Word Operand Write to Word Port Example

Table 5-7. Data Bus Byte Enable Signals for Byte, Word, and Long-Word Ports

Transfer Size	SIZ1	SIZ0	A1	A0	Data Bus Active Sections Byte (B), Word (W) , Long-Word (L) Ports D31–D24	D23–D16	D15–D8	D7–D0
Byte	0	1	0	0	B W L	—	—	—
	0	1	0	1	B	W L	—	—
	0	1	1	0	B W	—	L	—
	0	1	1	1	B	W	—	L
Word	1	0	0	0	B W L	W L	—	—
	1	0	0	1	B	W L	L	—
	1	0	1	0	B W	W	L	L
	1	0	1	1	B	W	—	L
3 Bytes	1	1	0	0	B W L	W L	L	—
	1	1	0	1	B	W L	L	L
	1	1	1	0	B W	W	L	L
	1	1	1	1	B	W	—	L
Long Word	0	0	0	0	B W L	W L	L	L
	0	0	0	1	B	W L	L	L
	0	0	1	0	B W	W	L	L
	0	0	1	1	B	W	—	L

Figure 5-18 shows a logic diagram of one method for generating byte enable signals for 16- and 32-bit ports from the SIZ1, SIZ0, A1, and A0 encodings and the R/$\overline{W}$ signal.

5.2.5 Cache Interactions

The organization and requirements of the on-chip instruction cache affect the interpretation of $\overline{DSACK1}$ and $\overline{DSACK0}$. Since the MC68020/EC020 attempts to load all instructions into the on-chip cache, the bus may operate differently when caching is enabled. Specifically, on read cycles that terminate normally, the A1, A0, SIZ1, and SIZ0 signals do not apply.

The cache can also affect the assertion of $\overline{AS}$ and the operation of a read cycle. The search of the cache by the processor begins when the sequencer requires an instruction. At this time, the bus controller may also initiate an external bus cycle in case the requested item is not resident in the instruction cache. If an internal cache hit occurs, the external cycle aborts, and $\overline{AS}$ is not asserted.

For the MC68020, if the bus is not occupied with another read or write cycle, the bus controller asserts the $\overline{ECS}$ signal (and the $\overline{OCS}$ signal, if appropriate). It is possible to have $\overline{ECS}$ asserted on multiple consecutive clock cycles. Note that there is a minimum time specified from the negation of $\overline{ECS}$ to the next assertion of $\overline{ECS}$ (refer to **Section 10 Electrical Characteristics**). Instruction prefetches can occur every other clock so that if, after an aborted cycle due to an instruction cache hit, the bus controller asserts $\overline{ECS}$ on the next clock, this second cycle is for a data fetch. Note that, if the bus controller is executing other cycles, these aborted cycles due to cache hits may not be seen externally.

State 5

MC68020—The processor negates $\overline{AS}$, $\overline{DS}$, and $\overline{DBEN}$ during S5. If more than one read cycle is required to read in the operand(s), S0–S5 are repeated for each read cycle. When the read cycle(s) are complete, the processor holds the address, R/$\overline{W}$, and FC2–FC0 valid in preparation for the write portion of the cycle.

The external device keeps its data and $\overline{DSACK1}$/$\overline{DSACK0}$ signals asserted until it detects the negation of $\overline{AS}$ or $\overline{DS}$ (whichever it detects first). The device must remove the data and negate $\overline{DSACK1}$/$\overline{DSACK0}$ within approximately one clock period after sensing the negation of $\overline{AS}$ or $\overline{DS}$. $\overline{DSACK1}$/$\overline{DSACK0}$ signals that remain asserted beyond this limit may be prematurely detected for the next portion of the operation.

MC68EC020—The processor negates $\overline{AS}$, $\overline{DS}$, and $\overline{DBEN}$ during S5. If more than one read cycle is required to read in the operand(s), S0–S5 are repeated for each read cycle. When the read cycle(s) is complete, the processor holds the address, R/$\overline{W}$, and FC2–FC0 valid in preparation for the write portion of the cycle.

The external device keeps its data and $\overline{DSACK1}$/$\overline{DSACK0}$ signals asserted until it detects the negation of $\overline{AS}$ or $\overline{DS}$ (whichever it detects first). The device must remove the data and negate $\overline{DSACK1}$/$\overline{DSACK0}$ within approximately one clock period after sensing the negation of $\overline{AS}$ or $\overline{DS}$. $\overline{DSACK1}$/$\overline{DSACK0}$ signals that remain asserted beyond this limit may be prematurely detected for the next portion of the operation.

Idle States

MC68020/EC020—The processor does not assert any new control signals during the idle states, but it may internally begin the modify portion of the cycle at this time. S6–S11 are omitted if no write cycle is required. If a write cycle is required, the R/$\overline{W}$ signal remains in the read mode until S6 to prevent bus conflicts with the preceding read portion of the cycle; the data bus is not driven until S8.

State 6

MC68020—The processor asserts $\overline{ECS}$ and $\overline{OCS}$ in S6 to indicate that another external cycle is beginning. The processor drives R/$\overline{W}$ low for a write cycle. Depending on the write operation to be performed, the address lines may change during S6.

MC68EC020—During S6, the processor drives R/$\overline{W}$ low for a write cycle. Depending on the write operation to be performed, the address lines may change during S6.

State 7

MC68020—During S7, the processor asserts $\overline{AS}$, indicating that the address on the address bus is valid. The processor also asserts $\overline{DBEN}$, which can be used to enable data buffers. In addition, $\overline{ECS}$ (and $\overline{OCS}$, if asserted) is negated during S7.

MC68EC020—During S7, the processor asserts $\overline{AS}$, indicating that the address on the address bus is valid.

State 8

MC68020/EC020—During S8, the processor places the data to be written onto the data bus.

5.4.2 Breakpoint Acknowledge Cycle

The breakpoint acknowledge cycle is generated by the execution of a BKPT instruction. The breakpoint acknowledge cycle allows the external hardware to provide an instruction word directly into the instruction pipeline as the program executes. This cycle accesses the CPU space with a type field of zero and provides the breakpoint number specified by the instruction on address lines A4–A2. If the external hardware terminates the cycle with $\overline{DSACK1}/\overline{DSACK0}$, the data on the bus (an instruction word) is inserted into the instruction pipe, replacing the breakpoint opcode, and is executed after the breakpoint acknowledge cycle completes. The BKPT instruction requires a word to be transferred so that if the first bus cycle accesses an 8-bit port, a second cycle is required. If the external logic terminates the breakpoint acknowledge cycle with $\overline{BERR}$ (i.e., no instruction word available), the processor takes an illegal instruction exception. Figure 5-35 is a flowchart of the breakpoint acknowledge cycle. Figure 5-36 shows the timing for a breakpoint acknowledge cycle that returns an instruction word. Figure 5-37 shows the timing for a breakpoint acknowledge cycle that signals an exception.

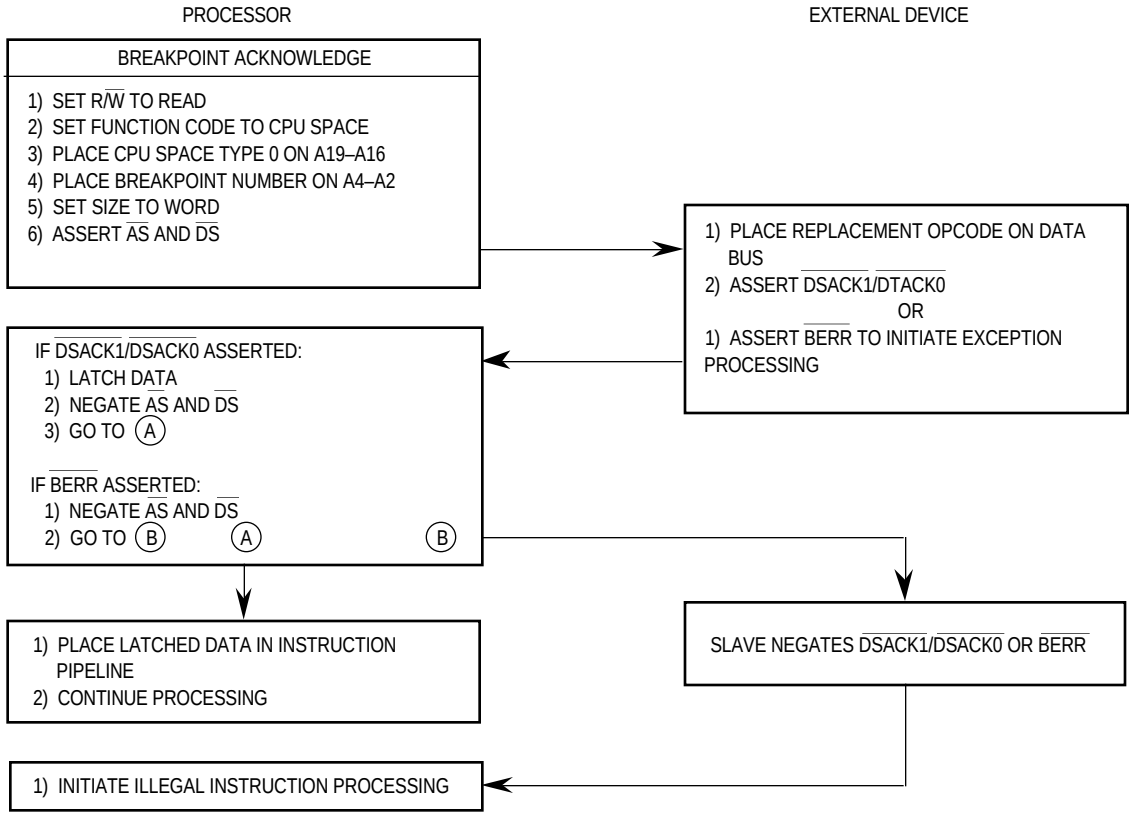
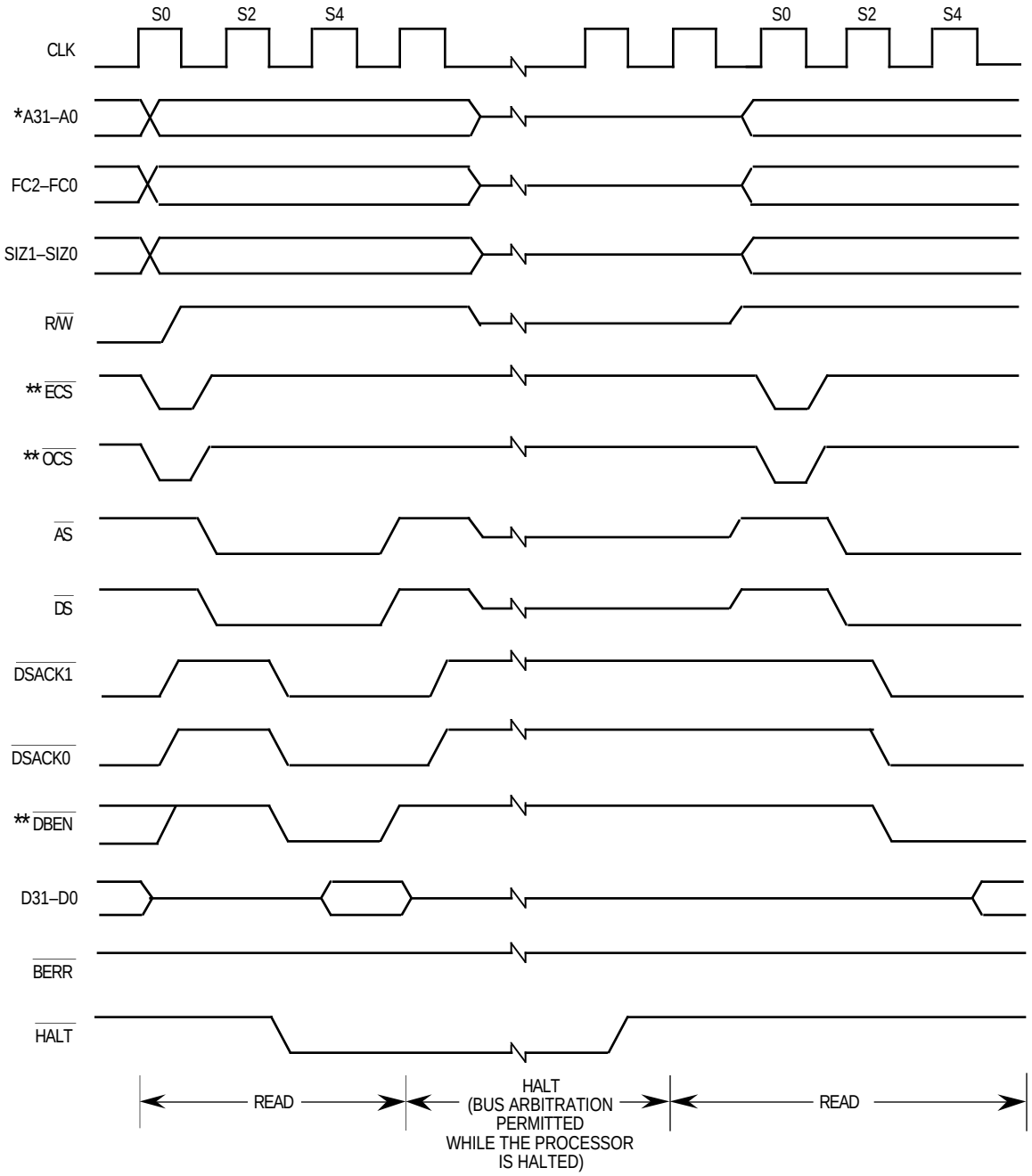


Figure 5-35. Breakpoint Acknowledge Cycle Flowchart



* For the MC68EC020, A23–A0.
** This signal does not apply to the MC68EC020.

Figure 5-41. Halt Operation Timing

State changes occur on the next rising edge of the clock after the internal signal is recognized as valid. The $\overline{BG}$ signal transitions on the falling edge of the clock after a state is reached during which G changes. The bus control signals (controlled by T) are driven by the processor immediately following a state change when bus mastership is returned to the MC68EC020.

State 0, at the top center of the diagram, in which both G and T are negated, is the state of the bus arbiter while the processor is bus master. Request R keeps the arbiter in state 0 as long as it is negated. When a request R is received, both grant G and signal T are asserted (in state 1 at the top left). The next clock causes a change to state 2, at the lower left, in which G and T are held. The bus arbiter remains in that state until request R is negated. Then the arbiter changes to the center state, state 3, and negates grant G. The next clock takes the arbiter to state 4, at the upper right, in which grant G remains negated and signal T remains asserted. The arbiter returns to the original state, state 0, and negates signal T. This sequence of states follows the normal sequence of signals for relinquishing the bus to an external bus master. Other states apply to other possible sequences of R.

The MC68EC020 does not allow arbitration of the external bus during the read-modify-write sequence. For the duration of this sequence, the MC68EC020 ignores the $\overline{BR}$ input. If mastership of the MC68EC020 bus is required during a read-modify-write operation, $\overline{BERR}$ must be used to abort the read-modify-write sequence. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in Figure 5-49.

The transfer operation word primitive uses the CA and PC bits as described in **7.4.2 Coprocessor Response Primitive General Format**. If this primitive is issued with CA = 0 during a conditional category instruction, the main processor initiates protocol violation exception processing.

When the main processor reads this primitive from the response CIR, it transfers the F-line operation word of the currently executing coprocessor instruction to the operation word CIR. The value of the scanPC is not affected by this primitive.

7.4.7 Transfer from Instruction Stream Primitive

The transfer from instruction stream primitive initiates transfers of operands from the instruction stream to the coprocessor. This primitive applies to general and conditional category instructions. Figure 7-27 shows the format of the transfer from instruction stream primitive.

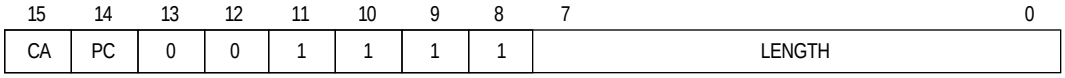


Figure 7-27. Transfer from Instruction Stream Primitive Format

The transfer from instruction stream primitive uses the CA and PC bits as described in **7.4.2 Coprocessor Response Primitive General Format**. If this primitive is issued with CA = 0 during a conditional category instruction, the main processor initiates protocol violation exception processing.

The length field of this primitive specifies the length, in bytes, of the operand to be transferred from the instruction stream to the coprocessor. The length must be an even number of bytes. If an odd length is specified, the main processor initiates protocol violation exception processing (refer to **7.5.2.1 Protocol Violations**).

This primitive transfers coprocessor-defined extension words to the coprocessor. When the main processor reads this primitive from the response CIR, it copies the number of bytes indicated by the length field from the instruction stream to the operand CIR. The first word or long word transferred is at the location pointed to by the scanPC when the primitive is read by the main processor. The scanPC is incremented after each word or long word is transferred. When execution of the primitive has completed, the scanPC has been incremented by the total number of bytes transferred and points to the word following the last word transferred. The main processor transfers the operands from the instruction stream, using a sequence of long-word writes, to the operand CIR. If the length field is not an even multiple of four bytes, the last two bytes from the instruction stream are transferred using a word write to the operand CIR.

8.1.5 Instruction Stream Timing Examples

A programming example allows a more detailed examination of these effects. The effect of instruction execution overlap on instruction timing is illustrated by the following example instruction stream.

Instruction	
#1) MOVE.L	D4,(A1)+
#2) ADD.L	D4,D5
#3) MOVE.L	(A1), –(A2)
#4) ADD.L	D5,D6

Example 1

For the first example, the assumptions are:

1. The data bus is 32 bits,
2. The first instruction is prefetched from an odd-word address,
3. Memory access occurs with no wait states, and
4. The instruction cache is disabled.

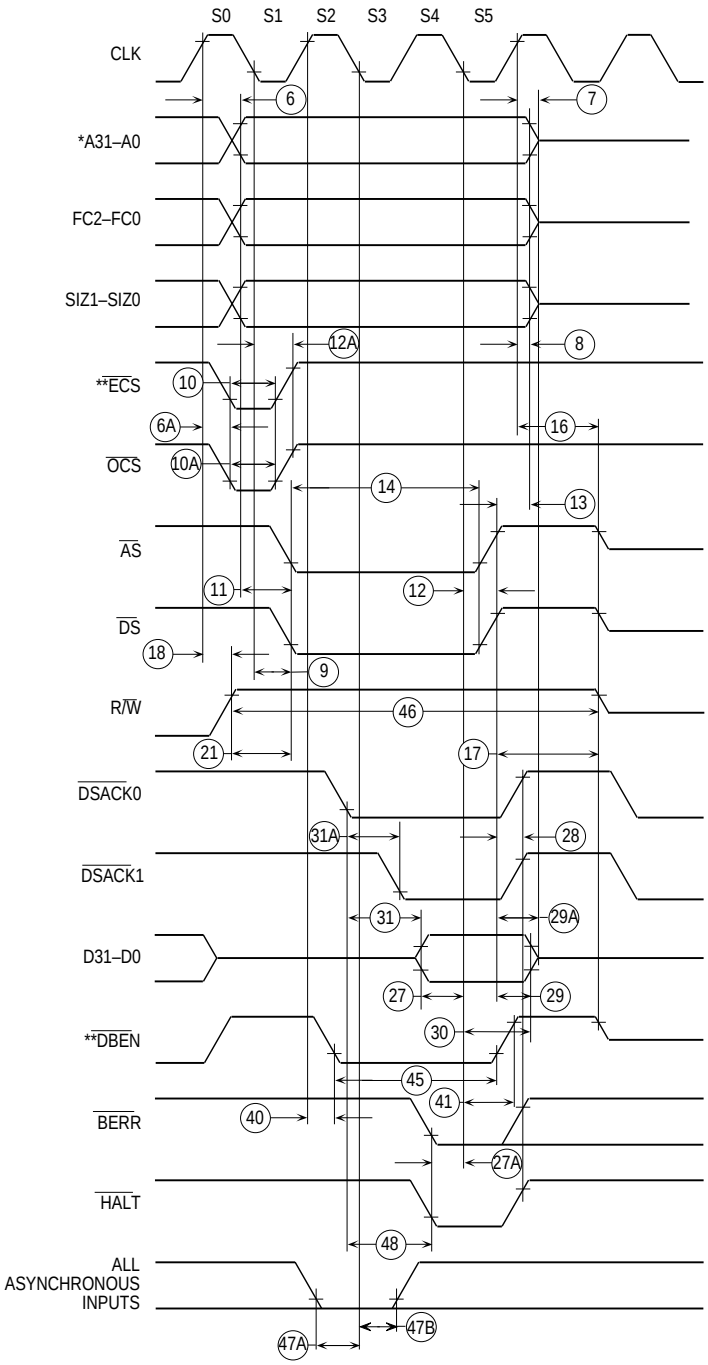
For example 1, the instruction stream is positioned in 32-bit memory as follows:

Address			
	n	•••	MOVE #1
	n + 4	ADD #2	MOVE #3
	n + 8	ADD #4	•••

Figure 8-3 shows processor activity on the first example instruction stream. It shows the activity of the external bus, the bus controller, the sequencer, and the attributed instruction execution time.

CACHE CASE (Continued)

Source Address Mode	Destination							
	(d$_8$,An,Xn)	**(d$_{16}$,An,Xn)**	**(B)**	**(d$_{16}$,B)**	**(d$_{32}$,B)**	**([B],I)**	**([B],I,d$_{16}$)**	**([B],I,d$_{32}$)**
Rn	**7**(0/0/1)	**9**(0/0/1)	**8**(0/0/1)	**10**(0/0/1)	**14**(0/0/1)	**12**(1/0/1)	**14**(1/0/1)	**15**(1/0/1)
#<data>.B,W	**7**(0/0/1)	**9**(0/0/1)	**8**(0/0/1)	**10**(0/0/1)	**14**(0/0/1)	**12**(1/0/1)	**14**(1/0/1)	**15**(1/0/1)
#<data>.L	**9**(0/0/1)	**11**(0/0/1)	**10**(0/0/1)	**12**(0/0/1)	**16**(0/0/1)	**14**(1/0/1)	**16**(1/0/1)	**17**(1/0/1)
(An)	**9**(1/0/1)	**11**(1/0/1)	**10**(1/0/1)	**12**(1/0/1)	**16**(1/0/1)	**14**(2/0/1)	**16**(2/0/1)	**17**(2/0/1)
(An)+	**9**(1/0/1)	**11**(1/0/1)	**10**(1/0/1)	**12**(1/0/1)	**16**(1/0/1)	**14**(2/0/1)	**16**(2/0/1)	**17**(2/0/1)
–(An)	**10**(1/0/1)	**12**(1/0/1)	**11**(1/0/1)	**13**(1/0/1)	**17**(1/0/1)	**15**(2/0/1)	**17**(2/0/1)	**18**(2/0/1)
(d$_{16}$,An) or (d$_{16}$,PC)	**10**(1/0/1)	**12**(2/0/1)	**11**(1/0/1)	**13**(1/0/1)	**17**(1/0/1)	**15**(2/0/1)	**17**(2/0/1)	**18**(2/0/1)
(xxx).W	**9**(1/0/1)	**11**(1/0/1)	**10**(1/0/1)	**12**(1/0/1)	**16**(1/0/1)	**14**(2/0/1)	**16**(2/0/1)	**17**(2/0/1)
(xxx).L	**9**(1/0/1)	**11**(1/0/1)	**10**(1/0/1)	**12**(1/0/1)	**16**(1/0/1)	**14**(2/0/1)	**16**(2/0/1)	**17**(2/0/1)
(d$_8$,An,Xn) or (d$_8$,PC,Xn)	**12**(1/0/1)	**14**(1/0/1)	**13**(1/0/1)	**15**(1/0/1)	**19**(1/0/1)	**17**(2/0/1)	**19**(2/0/1)	**20**(2/0/1)
(d$_{16}$,An,Xn) or (d$_{16}$,PC,Xn)	**12**(1/0/1)	**14**(1/0/1)	**13**(1/0/1)	**15**(1/0/1)	**19**(1/0/1)	**17**(2/0/1)	**19**(2/0/1)	**20**(2/0/1)
(B)	**12**(1/0/1)	**14**(1/0/1)	**13**(1/0/1)	**15**(1/0/1)	**19**(1/0/1)	**17**(2/0/1)	**19**(2/0/1)	**20**(2/0/1)
(d$_{16}$,B)	**14**(1/0/1)	**16**(1/0/1)	**15**(1/0/1)	**17**(1/0/1)	**21**(1/0/1)	**19**(2/0/1)	**21**(2/0/1)	**22**(2/0/1)
(d$_{32}$,B)	**18**(1/0/1)	**20**(1/0/1)	**19**(1/0/1)	**21**(1/0/1)	**25**(1/0/1)	**23**(2/0/1)	**25**(2/0/1)	**26**(2/0/1)
([B],I)	**17**(2/0/1)	**19**(2/0/1)	**18**(2/0/1)	**20**(2/0/1)	**24**(2/0/1)	**22**(3/0/1)	**24**(3/0/1)	**25**(3/0/1)
([B],I,d$_{16}$)	**19**(2/0/1)	**21**(2/0/1)	**20**(2/0/1)	**22**(2/0/1)	**26**(2/0/1)	**24**(3/0/1)	**26**(3/0/1)	**27**(3/0/1)
([B],I,d$_{32}$)	**19**(2/0/1)	**21**(2/0/1)	**20**(2/0/1)	**22**(2/0/1)	**26**(2/0/1)	**24**(3/0/1)	**26**(3/0/1)	**27**(3/0/1)
([d$_{16}$,B],I)	**19**(2/0/1)	**21**(2/0/1)	**20**(2/0/1)	**22**(2/0/1)	**26**(2/0/1)	**24**(3/0/1)	**26**(3/0/1)	**27**(3/0/1)
([d$_{16}$,B],I,d$_{16}$)	**21**(2/0/1)	**23**(2/0/1)	**22**(2/0/1)	**24**(2/0/1)	**28**(2/0/1)	**26**(3/0/1)	**28**(3/0/1)	**29**(3/0/1)
([d$_{16}$,B],I,d$_{32}$)	**21**(2/0/1)	**23**(2/0/1)	**22**(2/0/1)	**24**(2/0/1)	**28**(2/0/1)	**26**(3/0/1)	**28**(3/0/1)	**29**(3/0/1)
([d$_{32}$,B],I)	**23**(2/0/1)	**25**(2/0/1)	**24**(2/0/1)	**26**(2/0/1)	**30**(2/0/1)	**28**(3/0/1)	**30**(3/0/1)	**31**(3/0/1)
([d$_{32}$,B],I,d$_{16}$)	**25**(2/0/1)	**27**(2/0/1)	**26**(2/0/1)	**28**(2/0/1)	**32**(2/0/1)	**30**(3/0/1)	**32**(3/0/1)	**33**(3/0/1)
([d$_{32}$,B],I,d$_{32}$)	**25**(2/0/1)	**27**(2/0/1)	**26**(2/0/1)	**28**(2/0/1)	**32**(2/0/1)	**30**(3/0/1)	**32**(3/0/1)	**33**(3/0/1)



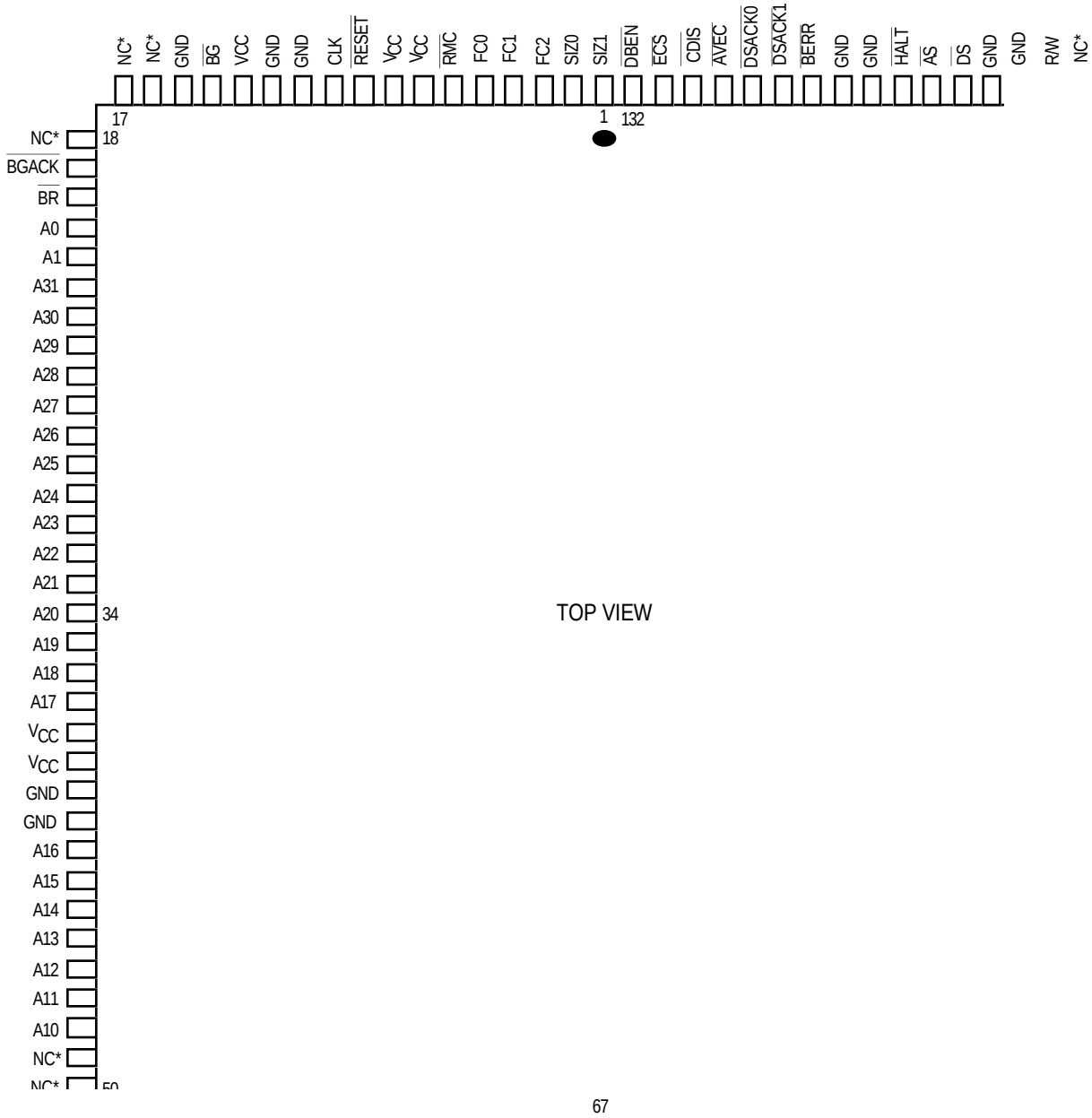
*For the MC68EC020, A23–A0.
**This signal does not apply to the MC68EC020.

NOTE: 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 will be linear between 0.8 V and 2.0 V.

FIGURE 10-3
MC68020UM

Figure 10-3. Read Cycle Timing Diagram

11.2.4 MC68020 FC and FE Suffix—Pin Assignment

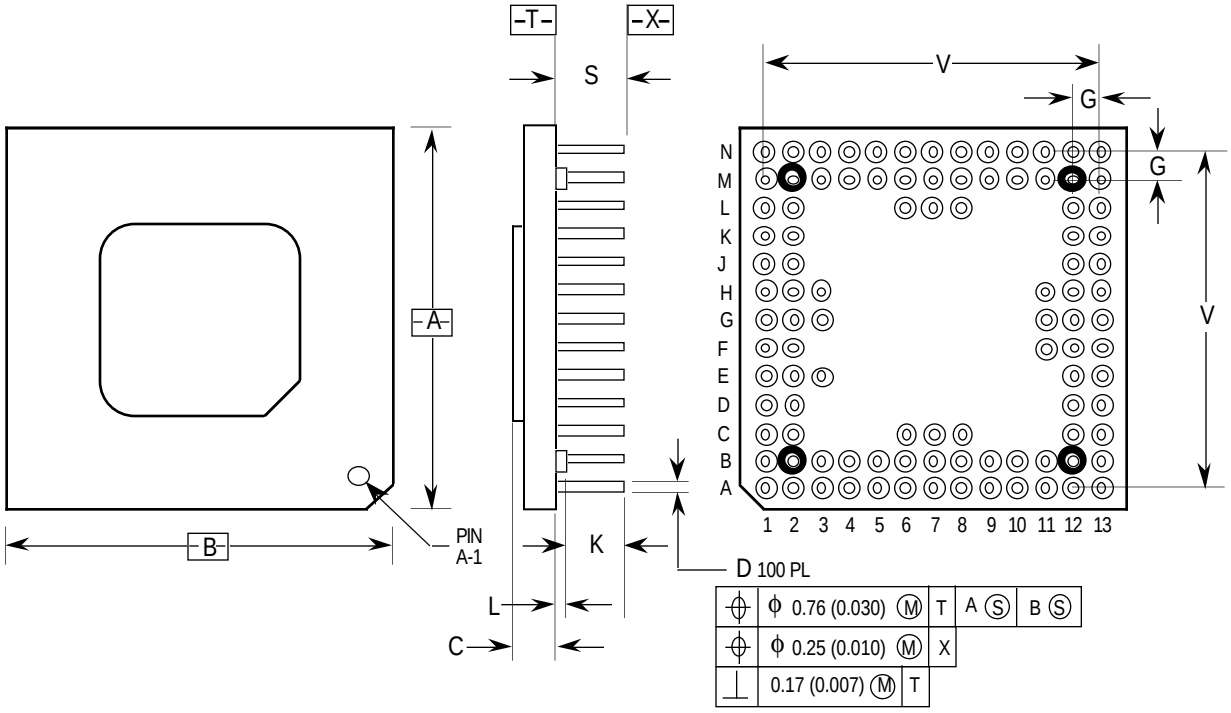


The V_{CC} and GND pins are separated into four groups to provide individual power supply connections for the address bus buffers, data bus buffers, and all other output buffers and internal logic. It is recommended that all pins be connected to power and ground as indicated. NC pins are reserved by Motorola for future use and should have no external connection.

Group	V_{CC}	GND
Address Bus	13, 38, 39	15, 40, 41, 62
Data Bus	79, 80, 96, 97	77, 78, 98, 99, 119, 120
Logic	7, 8, 65, 66	67, 68, 124, 125
Clock	—	11, 12

11.2.8 MC68EC020 RP Suffix—Package Dimensions

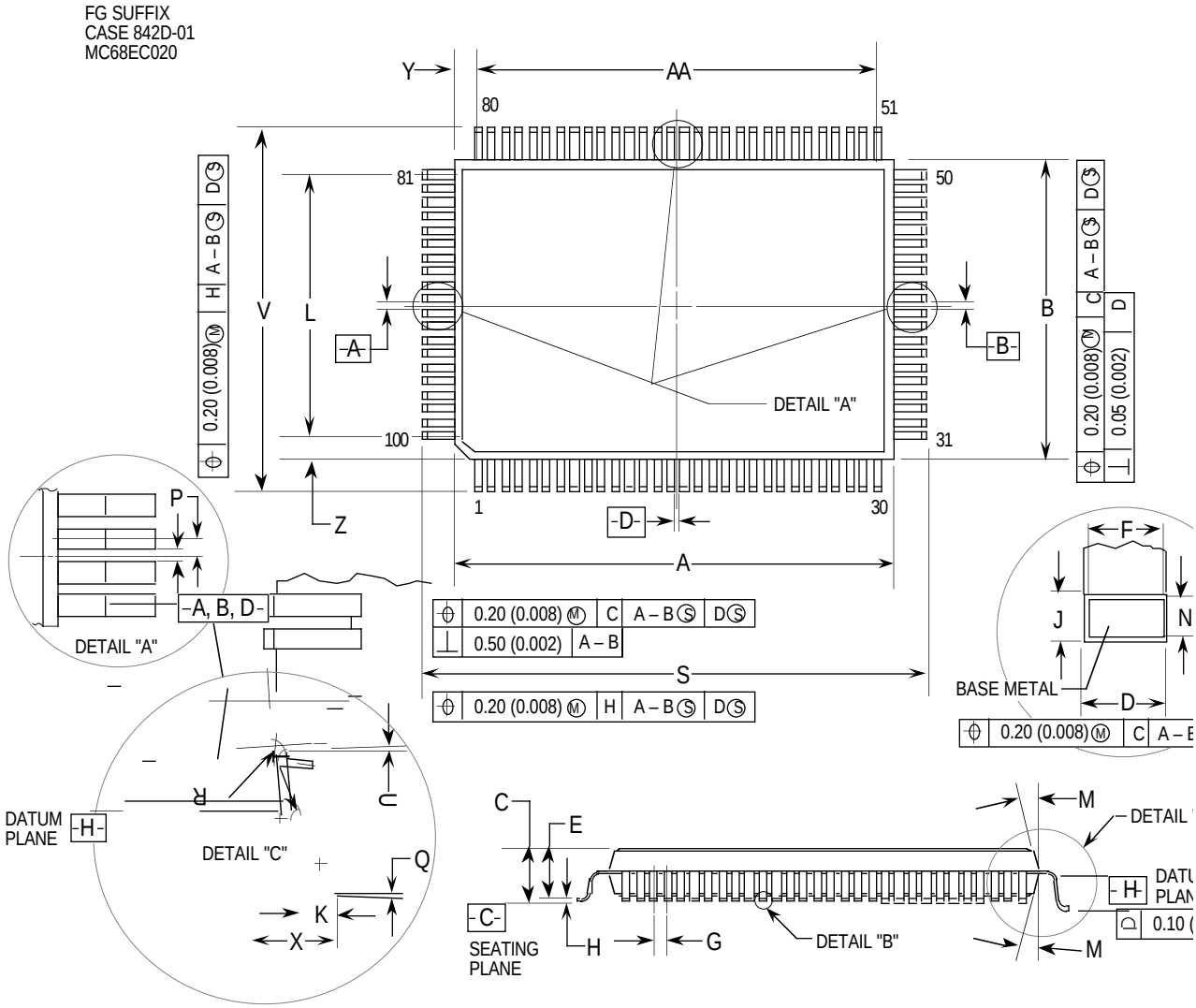
RP SUFFIX
CASE 789H-01
MC68EC020



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	34.04	35.05	1.340	1.380
B	34.04	35.05	1.340	1.380
C	2.92	3.18	0.115	0.135
D	0.44	0.55	0.017	0.022
G	2.54 BSC		0.100 BSC	
K	3.05	3.55	0.120	0.140
L	1.02	1.52	0.040	0.060
S	4.32	4.83	0.170	0.190
V	30.48 BSC		1.200 BSC	

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH
3. DIMENSION D INCLUDES LEAD FINISH.

11.2.10 MC68EC020 FG Suffix—Package Dimensions



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	19.90	20.10	0.783	0.791
B	13.90	14.10	0.547	0.555
C	—	3.30	—	0.130
D	0.22	0.38	0.009	0.015
E	2.55	3.05	0.100	0.120
F	0.22	0.33	0.009	0.013
G	0.65 BSC		0.026 BSC	
H	0.10	0.36	0.004	0.014
J	0.13	0.23	0.005	0.009
K	0.65	0.95	0.026	0.037
L	12.35 REF		0.486 REF	
M	5°	16°	5°	16°
N	0.13	0.17	0.005	0.007
P	0.325 BSC		0.013 BSC	
Q	0°	7°	0°	7°
R	0.25	0.35	0.010	0.014
S	23.65	24.15	0.931	0.951
T	0.13	—	0.005	—
U	0°	—	0°	—

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE **-H-** IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS **-A-**, **-B-**, AND **-D-** TO BE DETERMINED AT DATUM PLANE **-H-**.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE **-C-**.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOW PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLU MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE **-H-**.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWAE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) TOTAL IN EXCESS OF THE DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.