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	M68000
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
Co-Processors/DSP	-
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	114-BPGA
Supplier Device Package	114-PGA (34.55x34.55)
Purchase URL	https://www.e-xfl.com/product-detail/rochester-electronics/mc68020crc25e

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Concluded)

1.3 DATA TYPES AND ADDRESSING MODES OVERVIEW

For detailed information on the data types and addressing modes supported by the MC68020/EC020, refer to M68000PM/AD, *M68000 Family Programmer's Reference Manual*.

The MC68020/EC020 supports seven basic data types:

1. Bits
2. Bit Fields (Fields of consecutive bits, 1–32 bits long)
3. BCD Digits (Packed: 2 digits/byte, Unpacked: 1 digit/byte)
4. Byte Integers (8 bits)
5. Word Integers (16 bits)
6. Long-Word Integers (32 bits)
7. Quad-Word Integers (64 bits)

In addition, the MC68020/EC020 instruction set supports operations on other data types such as memory addresses. The coprocessor mechanism allows direct support of floating-point operations with the MC68881 and MC68882 floating-point coprocessors as well as specialized user-defined data types and functions.

The 18 addressing modes listed in Table 1-1 include nine basic types:

1. Register Direct
2. Register Indirect
3. Register Indirect with Index
4. Memory Indirect
5. PC Indirect with Displacement
6. PC Indirect with Index
7. PC Memory Indirect
8. Absolute
9. Immediate

The register indirect addressing modes have postincrement, predecrement, displacement, and index capabilities. The PC modes have index and offset capabilities. Both modes are extended to provide indirect reference through memory. In addition to these addressing modes, many instructions implicitly specify the use of the CCR, stack pointer, and/or PC.

3.8 BUS ARBITRATION CONTROL SIGNALS

The following signals are the bus arbitration control signals used to determine which device in a system is the bus master. Note that BGACK is implemented in the MC68020 and not implemented in the MC68EC020.

Bus Request (BR)

This input signal indicates that an external device needs to become the bus master. BR is typically a “wire-ORed” input (but does not need to be constructed from open-collector devices). Refer to **Section 5 Bus Operation** for more information on MC68020 bus arbitration. Refer to **Section 5 Bus Operation** and **Appendix A Interfacing an MC68EC020 to a DMA Device That Supports a Three-Wire Bus Arbitration Protocol** for more information on MC68EC020 bus arbitration.

Bus Grant (BG)

This output signal indicates that the MC68020/EC020 will release ownership of the bus when the current processor bus cycle completes. Refer to **Section 5 Bus Operation** for more information on MC68020 bus arbitration. Refer to **Section 5 Bus Operation** and **Appendix A Interfacing an MC68EC020 to a DMA Device That Supports a Three-Wire Bus Arbitration Protocol** for more information on MC68EC020 bus arbitration.

Bus Grant Acknowledge (BGACK, MC68020 only)

This input signal indicates that an external device has become the bus master. Refer to **Section 5 Bus Operation** for more information on MC68020 bus arbitration. Refer to **Section 5 Bus Operation** and **Appendix A Interfacing an MC68EC020 to a DMA Device That Supports a Three-Wire Bus Arbitration Protocol** for more information on MC68EC020 bus arbitration.

BGACK is not implemented in the MC68EC020.

3.9 BUS EXCEPTION CONTROL SIGNALS

The following signals are the bus exception control signals for the MC68020/EC020.

Reset (RESET)

This bidirectional open-drain signal is used to initiate a system reset. An external reset signal resets the MC68020/EC020 as well as all external devices. A reset signal from the processor (asserted as part of the RESET instruction) resets external devices only; the internal state of the processor is not altered. Refer to **Section 5 Bus Operation** for a description of reset bus operation and **Section 6 Exception Processing** for information about the reset exception.

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

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.

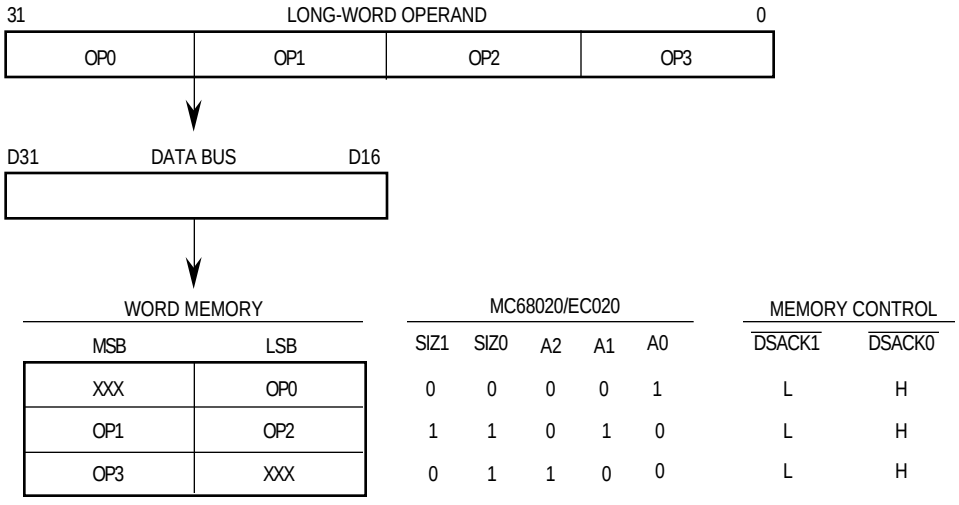


Figure 5-9. Misaligned Long-Word Operand Write to Word Port Example

State 4

MC68020/EC020—At the end of state 4 (S4), the processor latches the incoming data.

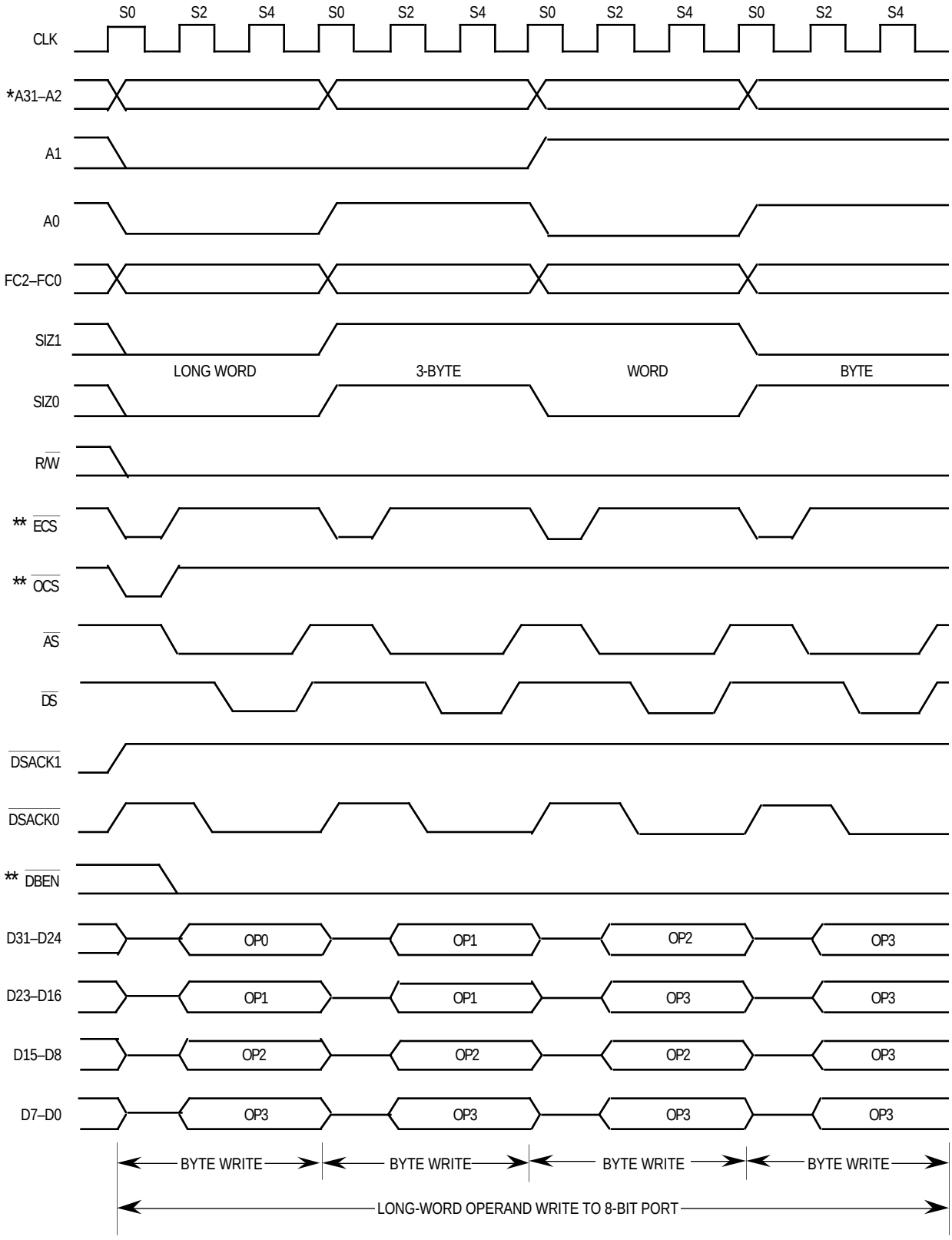
State 5

MC68020—The processor negates $\overline{AS}$, $\overline{DS}$, and $\overline{DBEN}$ during state 5 (S5). It holds the address valid during S5 to provide address hold time for memory systems. R/$\overline{W}$, SIZ1–SIZ0, and FC2–FC0 also remain valid throughout S5.

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 its 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 bus cycle.

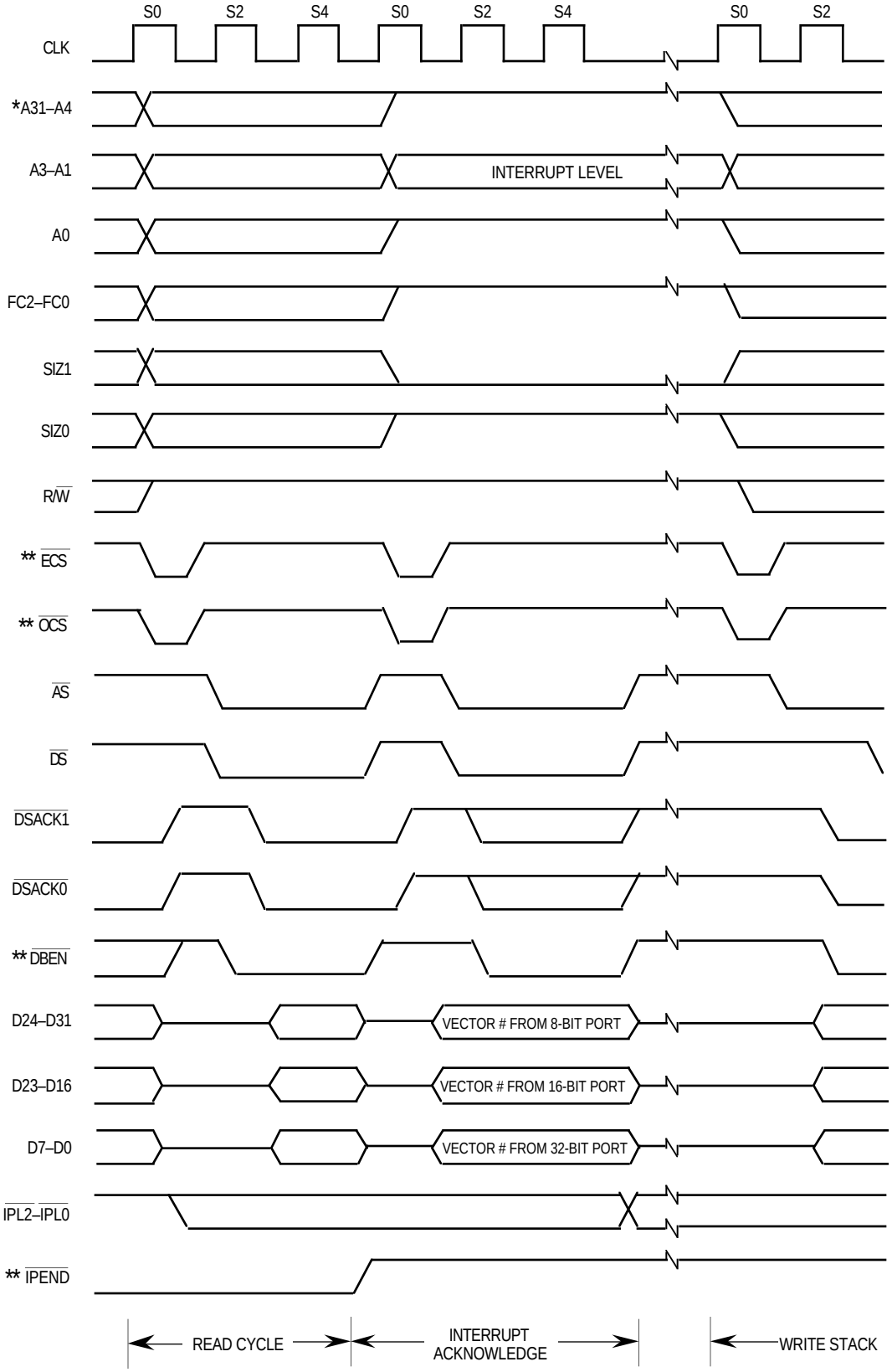
MC68EC020—The processor negates $\overline{AS}$ and $\overline{DS}$ during state S5. It holds the address valid during S5 to provide address hold time for memory systems. R/$\overline{W}$, SIZ1, SIZ0, and FC2–FC0 also remain valid throughout S5.

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 its 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 bus cycle.



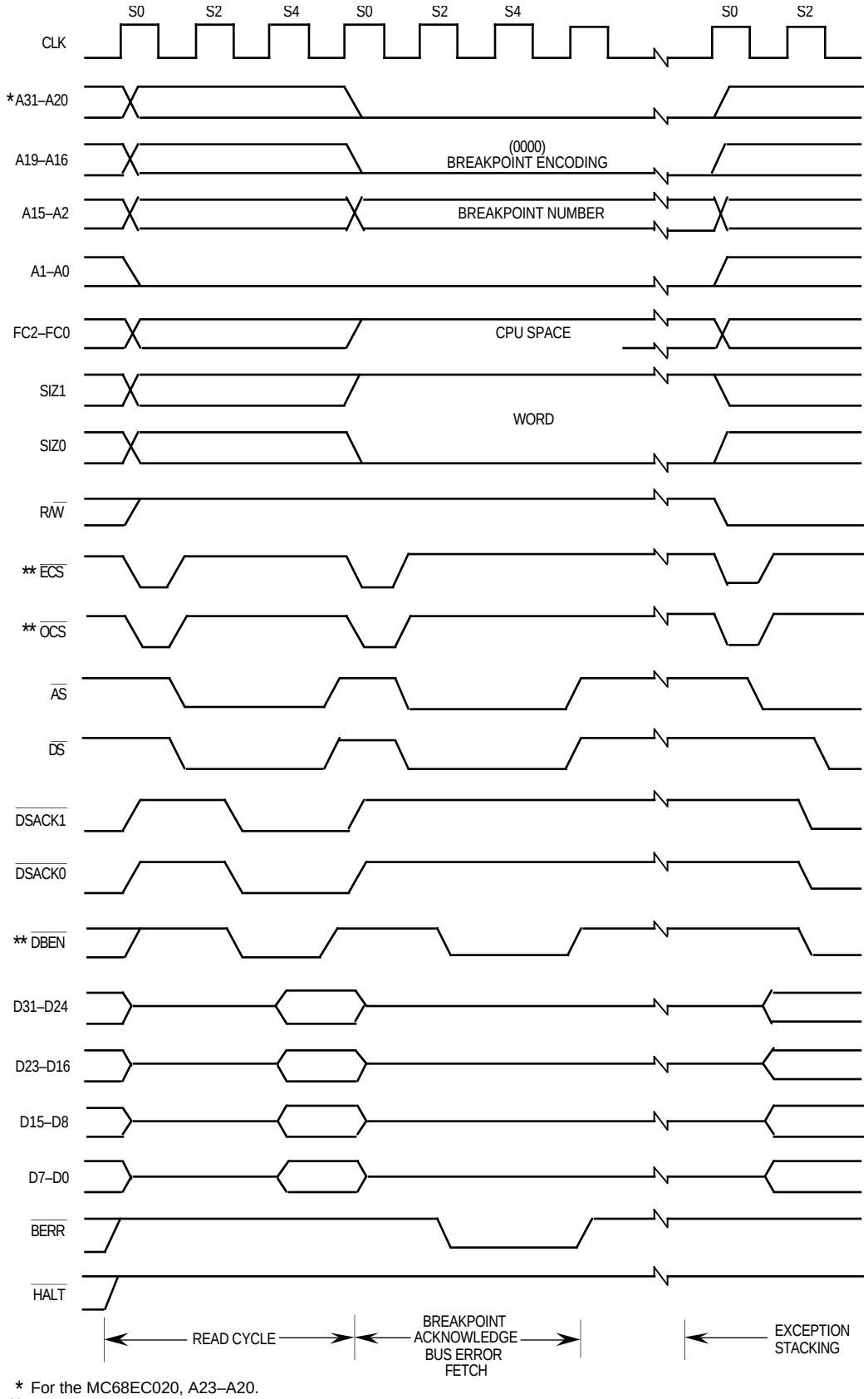
* For the MC68EC020, A23–A2.
This signal does not apply to the MC68EC020.

Figure 5-27. Long-Word Operand Write—8-Bit Port



* For the MC68EC020, A23–A4.
** This signal does not apply to the MC68EC020.

Figure 5-33. Interrupt Acknowledge Cycle Timing



* For the MC68EC020, A23–A20.
** This signal does not apply to the MC68EC020.

Figure 5-38. Bus Error without DSACK1/DSACK0

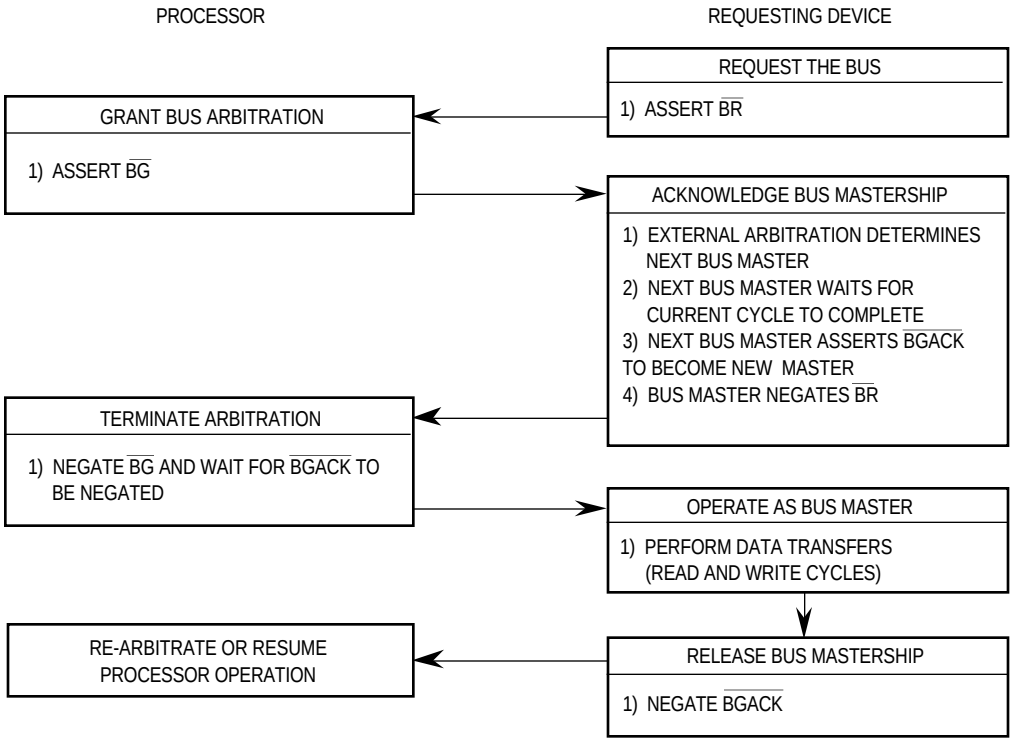


Figure 5-42. MC68020 Bus Arbitration Flowchart for Single Request

The timing diagram (see Figure 5-43) shows that $\overline{BR}$ is negated at the time that $\overline{BGACK}$ is asserted. This type of operation applies to a system consisting of the processor and one device capable of bus mastership. In a system having a number of devices capable of bus mastership, the $\overline{BR}$ line from each device can be wire-ORed to the processor. In such a system, more than one bus request can be asserted simultaneously.

The timing diagram in Figure 5-43 shows that $\overline{BG}$ is negated a few clock cycles after the transition of $\overline{BGACK}$. However, if bus requests are still pending after the negation of $\overline{BG}$, the processor asserts another $\overline{BG}$ within a few clock cycles after it was negated. This additional assertion of $\overline{BG}$ allows external arbitration circuitry to select the next bus master before the current bus master has finished with the bus. The following paragraphs provide additional information about the three steps in the arbitration process.

Bus arbitration requests are recognized during normal processing, $\overline{RESET}$ assertion, $\overline{HALT}$ assertion, and when the processor has halted due to a double bus fault.

Table 6-1. Exception Vector Assignments

Vector Number	Vector Offset		Assignment
	Hex	Space	
0	000	SP	Reset Initial Interrupt Stack Pointer
1	004	SP	Reset Initial Program Counter
2	008	SD	Bus Error
3	00C	SD	Address Error
4	010	SD	Illegal Instruction
5	014	SD	Zero Divide
6	018	SD	CHK, CHK2 Instruction
7	01C	SD	cpTRAPcc, TRAPcc, TRAPV Instructions
8	020	SD	Privilege Violation
9	024	SD	Trace
10	028	SD	Line 1010 Emulator
11	02C	SD	Line 1111 Emulator
12	030	SD	(Unassigned, Reserved)
13	034	SD	Coprocessor Protocol Violation
14	038	SD	Format Error
15	03C	SD	Uninitialized Interrupt
16–23	040 05C	SD SD	Unassigned, Reserved
24	060	SD	Spurious Interrupt
25	064	SD	Level 1 Interrupt Autovector
26	068	SD	Level 2 Interrupt Autovector
27	06C	SD	Level 3 Interrupt Autovector
28	070	SD	Level 4 Interrupt Autovector
29	074	SD	Level 5 Interrupt Autovector
30	078	SD	Level 6 Interrupt Autovector
31	07C	SD	Level 7 Interrupt Autovector
32–47	080 0BC	SD SD	TRAP #0–15 Instruction Vectors
48	0C0	SD	FPCP Branch or Set on Unordered Condition
49	0C4	SD	FPCP Inexact Result
50	0C8	SD	FPCP Divide by Zero
51	0CC	SD	FPCP Underflow
52	0D0	SD	FPCP Operand Error
53	0D4	SD	FPCP Overflow
54	0D8	SD	FPCP Signaling NAN
55	0DC	SD	Unassigned, Reserved
56	0E0	SD	PMMU Configuration
57	0E4	SD	PMMU Illegal Operation
58	0E8	SD	PMMU Access Level Violation
59–63	0EC 0FC	SD SD	Unassigned, Reserved
64–255	100 3FC	SD SD	User-Defined Vectors (192)

SP—Supervisor Program Space

SD—Supervisor Data Space

return a response primitive to request services necessary to evaluate the condition. If the coprocessor returns the false condition indicator, the main processor executes the next instruction in the instruction stream. If the coprocessor returns the true condition indicator, the main processor adds the displacement to the MC68020/EC020 scanPC (refer to **7.4.1 ScanPC**) to determine the address of the next instruction for the main processor to execute. The scanPC must be pointing to the location of the first word of the displacement in the instruction stream when the address is calculated. The displacement is a twos-complement integer that can be either a 16-bit word or a 32-bit long word. The main processor sign-extends the 16-bit displacement to a long-word value for the destination address calculation.

7.2.2.2 SET ON COPROCESSOR CONDITION INSTRUCTION. The set on coprocessor condition instruction sets or resets a flag (a data alterable byte) according to a condition evaluated by the coprocessor. The operation of this instruction type is similar to the operation of the Scc instruction in the M68000 family instruction set. Although the Scc instruction and the cpScc instruction do not explicitly cause a change of program flow, they are often used to set flags that control program flow.

7.2.2.2.1 Format. Figure 7-11 shows the format of the set on coprocessor condition instruction, denoted by the cpScc mnemonic.

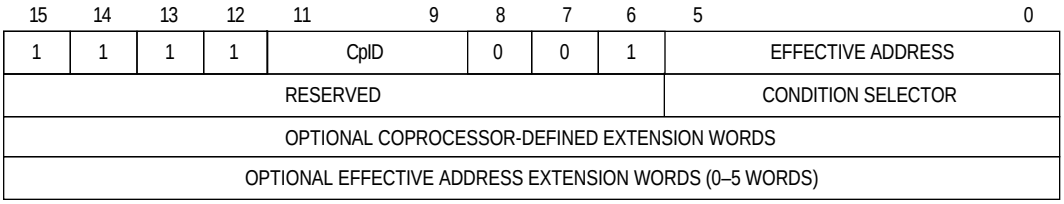


Figure 7-11. Set on Coprocessor Condition Instruction Format (cpScc)

The first word of the cpScc instruction, the F-line operation word, contains the CpID field in bits 11–9 and 001 in bits 8–6 to identify the cpScc instruction. Bits 5–0 of the F-line operation word are used to encode an M68000 family effective addressing mode (refer to M68000PM/AD, *M68000 Family Programmer's Reference Manual*).

The second word of the cpScc instruction format contains the coprocessor condition selector field in bits 5–0. Bits 15–6 of this word are reserved by Motorola and should be zero to ensure compatibility with future M68000 products. This word is written to the condition CIR to initiate execution of the cpScc instruction.

If the coprocessor requires additional information to evaluate the condition, the instruction can include extension words to provide this information. The number of these extension words, which follow the word containing the coprocessor condition selector field, is determined by the coprocessor design.

The PC value saved in this stack frame is the operation word address of the coprocessor instruction during which the primitive is received. The scanPC field contains the value of the MC68020/EC020 scanPC when the primitive is received. If the current instruction does not evaluate an effective address prior to the exception request primitive, the value of the effective address field in the stack frame is undefined.

The coprocessor uses this primitive to request exception processing for an exception during the instruction dialog with the main processor. If the exception handler does not modify the stack frame, the MC68020/EC020 returns from the exception handler and reads the response CIR. Thus, the main processor attempts to continue executing the suspended instruction by reading the response CIR and processing the primitive it receives.

7.4.20 Take Postinstruction Exception Primitive

The take postinstruction exception primitive initiates exception processing using a coprocessor-supplied exception vector number and the postinstruction exception stack frame format. This primitive applies to general and conditional category instructions. Figure 7-44 shows the format of the take postinstruction exception primitive.

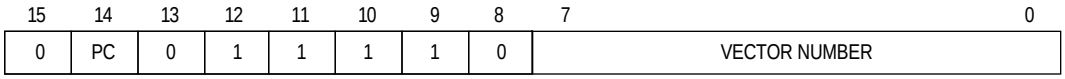


Figure 7-44. Take Postinstruction Exception Primitive Format

The take postinstruction exception primitive uses the PC bit as described in **7.4.2 Coprocessor Response Primitive General Format**. The vector number field contains the exception vector number used by the main processor to initiate exception processing.

When the main processor receives this primitive, it acknowledges the coprocessor exception request by writing an exception acknowledge mask to the control CIR (refer to **7.3.2 Control CIR**). The MC68020/EC020 then performs exception processing as described in **Section 6 Exception Processing**. The vector number for the exception is taken from the vector number field of the primitive, and the MC68020/EC020 uses the six-word stack frame format shown in Figure 7-45.

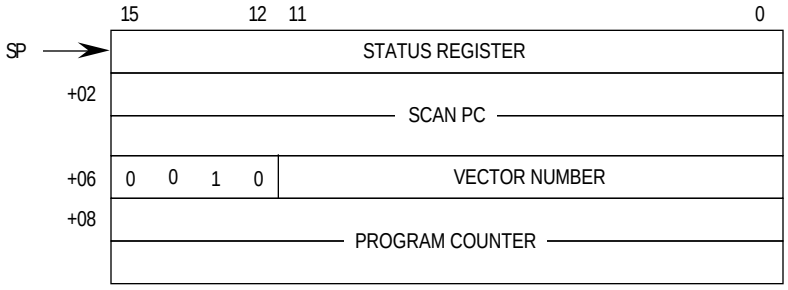


Figure 7-45. MC68020/EC020 Postinstruction Stack Frame

WORST 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	**9**(0/1/1)	**12**(0/2/1)	**10**(0/1/1)	**14**(0/2/1)	**19**(0/2/1)	**14**(1/1/1)	**17**(1/2/1)	**20**(1/2/1)
#<data>.B,W	**9**(0/1/1)	**12**(0/2/1)	**10**(0/1/1)	**14**(0/2/1)	**19**(0/2/1)	**14**(1/1/1)	**17**(1/2/1)	**20**(1/2/1)
#<data>.L	**11**(0/1/1)	**14**(0/2/1)	**12**(0/1/1)	**16**(0/2/1)	**21**(0/2/1)	**16**(1/1/1)	**19**(1/2/1)	**22**(1/2/1)
(An)	**11**(1/1/1)	**14**(1/2/1)	**12**(1/1/1)	**16**(1/2/1)	**21**(1/2/1)	**12**(2/1/1)	**19**(2/2/1)	**22**(2/2/1)
(An)+	**11**(1/1/1)	**14**(1/2/1)	**12**(1/1/1)	**16**(1/2/1)	**21**(1/2/1)	**12**(2/1/1)	**19**(2/2/1)	**22**(2/2/1)
–(An)	**12**(1/1/1)	**15**(1/2/1)	**13**(1/1/1)	**17**(1/2/1)	**22**(1/2/1)	**13**(2/1/1)	**20**(2/2/1)	**23**(2/2/1)
(d_{16},An) or (d_{16},PC)	**13**(1/2/1)	**16**(1/3/1)	**14**(1/2/1)	**18**(1/3/1)	**23**(1/3/1)	**14**(2/2/1)	**21**(2/3/1)	**24**(2/3/1)
(xxx).W	**12**(1/2/1)	**15**(1/3/1)	**13**(1/2/1)	**17**(1/3/1)	**22**(1/3/1)	**13**(2/2/1)	**20**(2/3/1)	**23**(2/3/1)
(xxx).L	**14**(1/2/1)	**17**(1/3/1)	**15**(1/2/1)	**19**(1/3/1)	**24**(1/3/1)	**15**(2/2/1)	**22**(2/3/1)	**25**(2/3/1)
(d_8,An,Xn) or (d_8,PC,Xn)	**15**(1/2/1)	**18**(1/3/1)	**16**(1/2/1)	**20**(1/3/1)	**25**(1/3/1)	**16**(2/2/1)	**23**(2/3/1)	**26**(2/3/1)
(d_{16},An,Xn) or (d_{16},PC,Xn)	**16**(1/2/1)	**19**(1/3/1)	**17**(1/2/1)	**21**(1/3/1)	**26**(1/3/1)	**17**(2/2/1)	**24**(2/3/1)	**27**(2/3/1)
(B)	**16**(1/2/1)	**19**(1/3/1)	**17**(1/2/1)	**21**(1/3/1)	**26**(1/3/1)	**17**(2/2/1)	**24**(2/3/1)	**27**(2/3/1)
(d_{16},B)	**19**(1/2/1)	**22**(1/3/1)	**20**(1/2/1)	**24**(1/3/1)	**29**(1/3/1)	**20**(2/2/1)	**27**(2/3/1)	**30**(2/3/1)
(d_{32},B)	**23**(1/3/1)	**26**(1/4/1)	**24**(1/3/1)	**28**(1/4/1)	**33**(1/4/1)	**24**(2/3/1)	**31**(2/4/1)	**34**(2/4/1)
([B],I)	**20**(2/2/1)	**23**(2/3/1)	**21**(2/2/1)	**25**(2/3/1)	**30**(2/3/1)	**21**(3/2/1)	**28**(3/3/1)	**31**(3/3/1)
([B],I,d_{16})	**23**(2/2/1)	**26**(2/3/1)	**24**(2/2/1)	**28**(2/3/1)	**33**(2/3/1)	**24**(3/2/1)	**31**(3/3/1)	**34**(3/3/1)
([B],I,d_{32})	**24**(2/3/1)	**27**(2/4/1)	**25**(2/3/1)	**29**(2/4/1)	**34**(2/4/1)	**25**(3/3/1)	**32**(3/4/1)	**35**(3/4/1)
([d_{16},B],I)	**23**(2/2/1)	**26**(2/3/1)	**24**(2/2/1)	**28**(2/3/1)	**33**(2/3/1)	**24**(3/2/1)	**31**(3/3/1)	**34**(3/3/1)
([d_{16},B],I,d_{16})	**26**(2/3/1)	**29**(2/4/1)	**27**(2/3/1)	**31**(2/4/1)	**36**(2/4/1)	**27**(3/3/1)	**34**(3/4/1)	**37**(3/4/1)
([d_{16},B],I,d_{32})	**27**(2/3/1)	**30**(2/4/1)	**28**(2/3/1)	**32**(2/4/1)	**37**(2/4/1)	**28**(3/3/1)	**35**(3/4/1)	**38**(3/4/1)
([d_{32},B],I)	**27**(2/3/1)	**30**(2/4/1)	**28**(2/3/1)	**32**(2/4/1)	**37**(2/4/1)	**28**(3/3/1)	**35**(3/4/1)	**38**(3/4/1)
([d_{32},B],I,d_{16})	**29**(2/3/1)	**32**(2/4/1)	**30**(2/3/1)	**34**(2/4/1)	**39**(2/4/1)	**30**(3/3/1)	**37**(3/4/1)	**40**(3/4/1)
([d_{32},B],I,d_{32})	**31**(2/4/1)	**34**(2/5/1)	**32**(2/4/1)	**36**(2/5/1)	**41**(2/5/1)	**32**(3/4/1)	**39**(3/5/1)	**42**(3/5/1)

8.2.16 Control Instructions

The control instructions table indicates the number of clock periods needed for the processor to perform the specified operation. Footnotes specify when it is necessary to add an entry from another table to calculate the total effective execution time for the given instruction. The total number of clock cycles is outside the parentheses; the number of read, prefetch, and write cycles is given inside the parentheses as (r/p/w). These cycles are included in the total clock cycle number.

	Instruction	Best Case	Cache Case	Worst Case
	ANDI to SR	**9**(0/0/0)	**12**(0/0/0)	**15**(0/2/0)
	EORI to SR	**9**(0/0/0)	**12**(0/0/0)	**15**(0/2/0)
	ORI to SR	**9**(0/0/0)	**12**(0/0/0)	**15**(0/2/0)
	ANDI to CCR	**9**(0/0/0)	**12**(0/0/0)	**15**(0/2/0)
	EORI to CCR	**9**(0/0/0)	**12**(0/0/0)	**15**(0/2/0)
	ORI to CCR	**9**(0/0/0)	**12**(0/0/0)	**15**(0/2/0)
	BSR	**5**(0/0/1)	**7**(0/0/1)	**13**(0/2/1)
**	CALLM (Type 0)	**28**(2/0/6)	**30**(2/0/6)	**36**(2/2/6)
**	CALLM (Type 1)—No Stack Copy	**48**(5/0/8)	**50**(5/0/8)	**56**(5/2/8)
**	CALLM (Type 1)—No Stack Copy	**55**(6/0/8)	**57**(6/0/8)	**64**(6/2/8)
**	CALLM (Type 1)—Stack Copy	**63 + 6n**(7 + n/0/8 + n)	**65 + 6n**(7 + n/0/8 + n)	**71 + 6n**(7 + n/2/8 + n)
‡	CAS (Successful Compare)	**15**(1/0/1)	**15**(1/0/1)	**16**(1/1/1)
‡	CAS (Unsuccessful Compare)	**12**(1/0/0)	**12**(1/0/0)	**13**(1/1/0)
	CAS2 (Successful Compare)	**23**(2/0/2)	**25**(2/0/2)	**28**(2/2/2)
	CAS2 (Unsuccessful Compare)	**19**(2/0/0)	**22**(2/0/0)	**25**(2/2/0)
*	CHK	**8**(0/0/0)	**8**(0/0/0)	**8**(0/1/0)
**	CHK2 EA,Rn	**16**(2/0/0)	**18**(2/0/0)	**18**(2/1/0)
%	JMP	**1**(0/0/0)	**4**(0/0/0)	**7**(0/2/0)
%	JSR	**3**(0/0/1)	**5**(0/0/1)	**11**(0/2/1)
†	LEA	**2**(0/0/0)	**2**(0/0/0)	**3**(0/1/0)
	LINK.W	**3**(0/0/1)	**5**(0/0/1)	**7**(0/1/1)
	LINK.L	**4**(0/0/1)	**6**(0/0/1)	**10**(0/2/1)
	NOP	**2**(0/0/0)	**2**(0/0/0)	**3**(0/1/0)
†	PEA	**3**(0/0/1)	**5**(0/0/1)	**6**(0/1/1)
	RTD	**9**(1/0/0)	**10**(1/0/0)	**12**(1/2/0)
	RTM (Type 0)	**18**(4/0/0)	**19**(4/0/0)	**22**(4/2/0)
	RTM (Type 1)	**31**(6/0/1)	**32**(6/0/1)	**35**(6/2/1)
	RTR	**13**(2/0/0)	**14**(2/0/0)	**15**(2/2/0)
	RTS	**9**(1/0/0)	**10**(1/0/0)	**12**(1/2/0)
	UNLK	**5**(1/0/0)	**6**(1/0/0)	**7**(1/1/0)

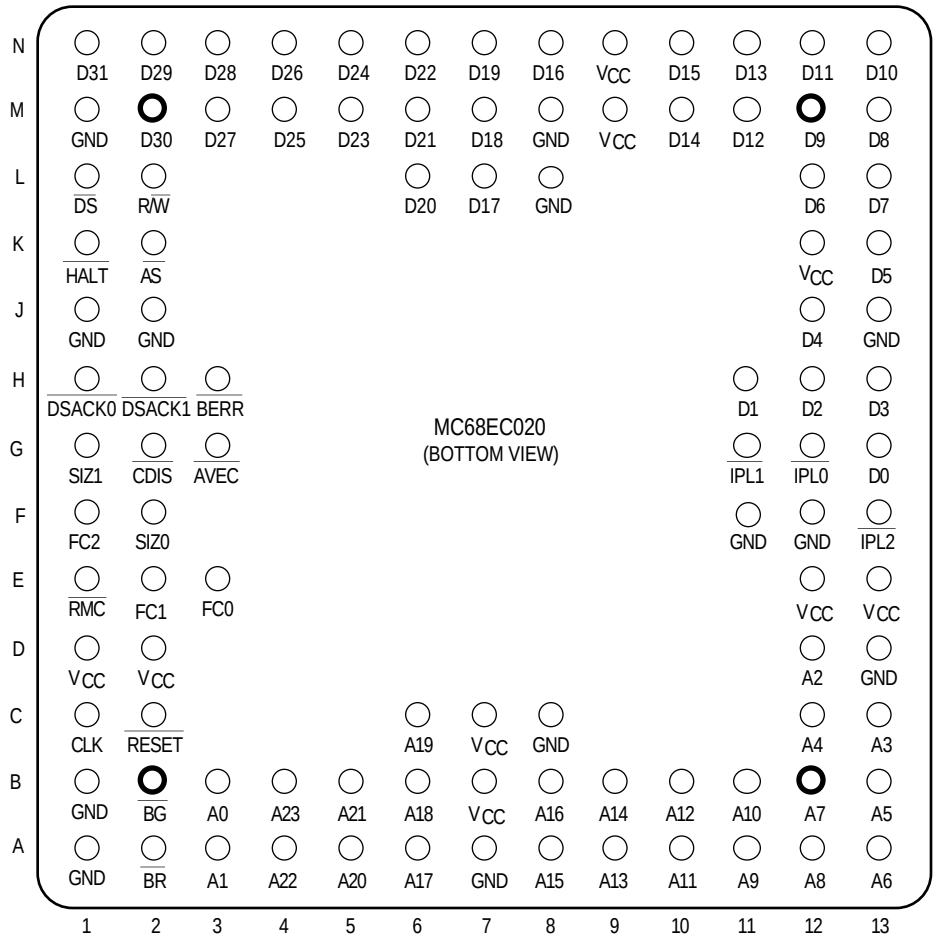
n—Number of Operand Transfers Required
%—Add Jump Effective Address Time
*Add Fetch Effective Address Time
**Add Fetch Immediate Address Time
†Add Calculate Effective Address Time
‡Add Calculate Immediate Address Time

```
PAL16L8
BYTE_SELECT
MC68020/EC020 BYTE DATA SELECT GENERATION FOR 32-BIT PORTS, MAPPED AND UNMAPPED.
MOTOROLA INC., AUSTIN, TEXAS
INPUTS:     A0      A1      SIZ0    SIZ1    RW      A18     A19     A20     A21     ~CPU
OUTPUTS:    ~UUDA       ~UMDA       ~LMDA       ~LLDA       ~UUDA       ~UMDB       ~LMDB       ~LLDB

!~UUDA      = RW                                    ;enable upper byte on read of 32-bit port
            +!A0 *!A1                               ;directly addressed, any size
!~UMDA      = RW                                    ;enable upper middle byte on read of 32-bit port
            +A0 *!A1                                ;directly addressed, any size
            +!A1 *!SIZ0                             ;even word aligned, size word or long word
            +!A1 *SIZ1                              ;even word aligned, size is word or three byte
!~LMDA      = RW                                    ;enable lower middle byte on read of 32-bit port    +!A0 *A1
            ;directly addressed, any size
            +!A1 *!SIZ0 *!SIZ1                      ;even word aligned, size is long word
            +!A1 *SIZ0 *SIZ1                        ;even word aligned, size is three byte
            +!A1 *A0 *!SIZ0                         ;even word aligned, size is word or long word
!~LLDA      = RW                                    ;enable lower byte on read of 32-bit port
            +A0 *A1                                 ;directly addressed, any size
            +A0 *SIZ0 *SIZ1                         ;odd byte alignment, three byte size
            +!SIZ0 *!SIZ1                           ;size is long word, any address
            +A1 *SIZ1                               ;odd word aligned, word or three byte size

!~UUDB      = RW *!~CPU * (addressb)                ;enable upper byte on read of 32-bit port
            +!A0 *!A1 *!~CPU * (addressb)           ;directly addressed, any size
!~UMDB      = RW *!~CPU * (addressb)                ;enable upper middle byte on read of 32-bit port
            + A0 *!A1 *!~CPU * (addressb)           ;directly addressed, any size
            +!A1 *!SIZ0 *!~CPU * (addressb)         ;even word aligned, size word or long word
            +!A1 *SIZ1 *!~CPU * (addressb)          ;even word aligned, size is word or three byte
!~LMDB      =RW *!~CPU * (addressb)                 ;enable lower middle byte on read of 32-bit port
            +!A0 * A1 *!~CPU * (addressb)           ;directly addressed, any size
            +!A1 *!SIZ0 *!SIZ1 *!~CPU * (addressb)  ;even word aligned, size is long word
            +!A1 * SIZ0 * SIZ1 *!~CPU * (addressb)  ;even word aligned, size is three byte
            +!A1 * A0 *!SIZ0 *!~CPU * (addressb)    ;even word aligned, size is word or long word
!~LLDB      =RW *!~CPU * (addressb)                 ;enable lower byte on read of 32-bit port
            +A0 * A1 *!~CPU * (addressb)            ;directly addressed, any size
            + A0 * SIZ0 * SIZ1 *!~CPU * (addressb)  ;odd byte alignment, three byte size
            +!SIZ0 *!SIZ1 *!~CPU * (addressb)       ;size is long word, any address
            +A1 * SIZ1 *!~CPU * (addressb)          ;odd word aligned, word or three byte size
```

DESCRIPTION: Byte select signals for writing. On reads, all byte selects are asserted if the respective memory block is addressed. The input signal CPU prevents byte select assertion during CPU space cycles and is derived from NANDing FC1–FC0 or FC2–FC0. The label (addressb) is a designer-selectable combination of address lines used to generate the proper address decode for the system's memory bank. With the address lines given here, the decode block size is 256 Kbytes to 2 Mbytes. A similar address might be included in the equations for UUDA, UMDA, etc. if the designer wishes them to be memory mapped also.

Figure 9-6. MC68020/EC020 Byte Select PAL Equations

11.2.7 MC68EC020 RP 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.

Group	V_{CC}	GND
Address Bus	B7, C7	A1, A7, C8, D13
Data Bus	K12, M9, N9	J13, L8, M1, M8
Logic	D1, D2, E12, E13	F11, F12, J1, J2
Clock	—	B1