Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

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

Applications of Embedded - Microprocessors

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

Details	
Product Status	Obsolete
Core Processor	EC000
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	16MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.21x24.21)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc000ei16

TABLE OF CONTENTS (Continued)

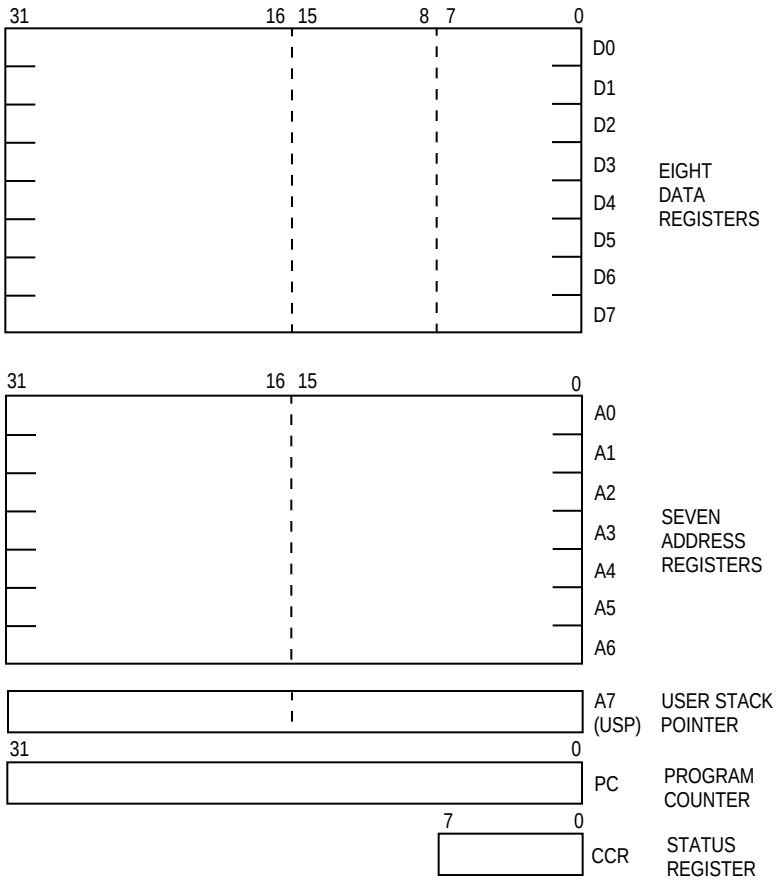


Figure 2-1. User Programmer's Model (MC68000/MC68HC000/MC68008/MC68010)

2.1.2 Supervisor Programmer's Model

The supervisor programmer's model consists of supplementary registers used in the supervisor mode. The M68000 MPUs contain identical supervisor mode register resources, which are shown in Figure 2-2, including the status register (high-order byte) and the supervisor stack pointer (SSP/A7').

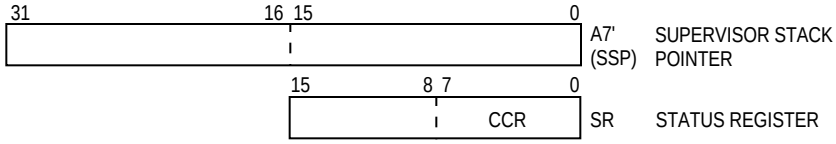


Figure 2-2. Supervisor Programmer's Model Supplement

The supervisor programmer's model supplement of the MC68010 is shown in Figure 2-3. In addition to the supervisor stack pointer and status register, it includes the vector base register (VRB) and the alternate function code registers (AFC).The VBR is used to determine the location of the exception vector table in memory to support multiple vector

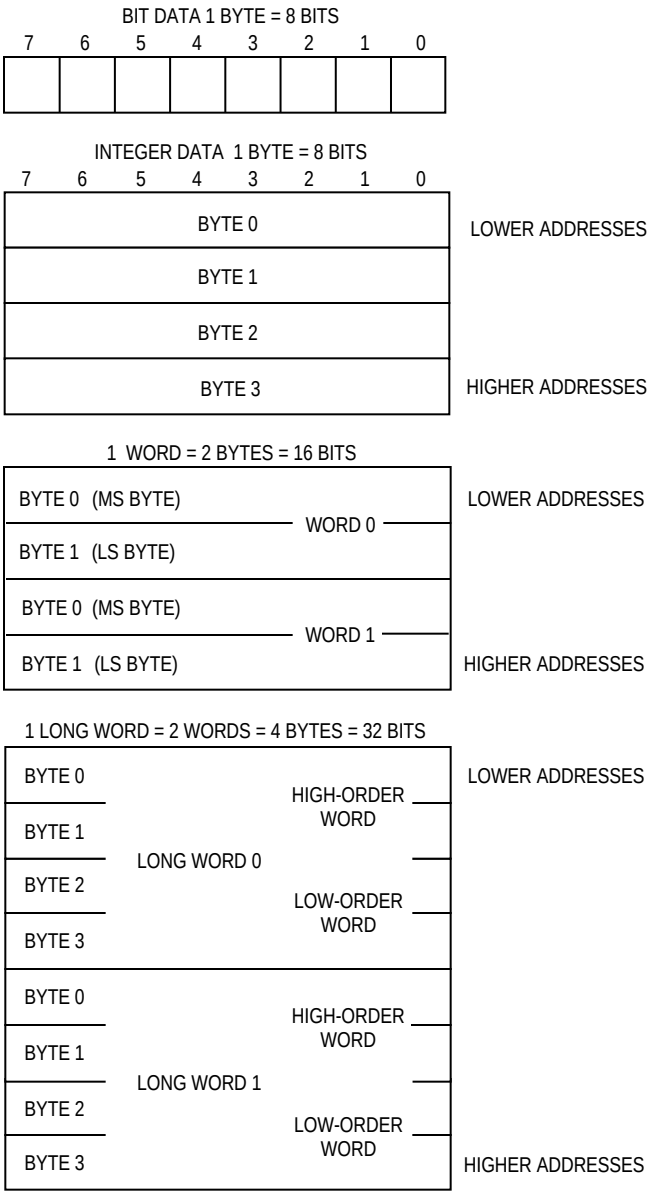


Figure 2-7. Memory Data Organization of the MC68008

2.5 INSTRUCTION SET SUMMARY

Table 2-2 provides an alphabetized listing of the M68000 instruction set listed by opcode, operation, and syntax. In the syntax descriptions, the left operand is the source operand, and the right operand is the destination operand. The following list contains the notations used in Table 2-2.

Table 2-2. Instruction Set Summary (Sheet 3 of 4)

Opcode	Operation	Syntax
MOVE USP	If supervisor state then USP → An or An → USP else TRAP	MOVE USP,An MOVE An,USP
MOVEC	If supervisor state then Rc → Rn or Rn → Rc else TRAP	MOVEC Rc,Rn MOVEC Rn,Rc
MOVEM	Registers → Destination Source → Registers	MOVEM register list,<ea> MOVEM <ea>,register list
MOVEP	Source → Destination	MOVEP Dx,(d,Ay) MOVEP (d,Ay),Dx
MOVEQ	Immediate Data → Destination	MOVEQ # <data>,Dn
MOVES	If supervisor state then Rn → Destination [DFC] or Source [SFC] → Rn else TRAP	MOVES Rn,<ea> MOVES <ea>,Rn
MULS	Source × Destination → Destination	MULS.W <ea>,Dn 16 x 16 → 32
MULU	Source × Destination → Destination	MULU.W <ea>,Dn 16 x 16 → 32
NBCD	0 – (Destination$_{10}$) – X → Destination	NBCD <ea>
NEG	0 – (Destination) → Destination	NEG <ea>
NEGX	0 – (Destination) – X → Destination	NEGX <ea>
NOP	None	NOP
NOT	~Destination → Destination	NOT <ea>
OR	Source V Destination → Destination	OR <ea>,Dn OR Dn,<ea>
ORI	Immediate Data V Destination → Destination	ORI # <data>,<ea>
ORI to CCR	Source V CCR → CCR	ORI # <data>,CCR
ORI to SR	If supervisor state then Source V SR → SR else TRAP	ORI # <data>,SR
PEA	Sp – 4 → SP; <ea> → (SP)	PEA <ea>
RESET	If supervisor state then Assert $\overline{\text{RESET}}$ Line else TRAP	RESET
ROL, ROR	Destination Rotated by <count> → Destination	ROd[1] Rx,Dy ROd[1] # <data>,Dy ROd[1] <ea>
ROXL, ROXR	Destination Rotated with X by <count> → Destination	ROXd[1] Dx,Dy ROXd[1] # <data>,Dy ROXd[1] <ea>
RTD	(SP) → PC; SP + 4 + d → SP	RTD #<displacement>

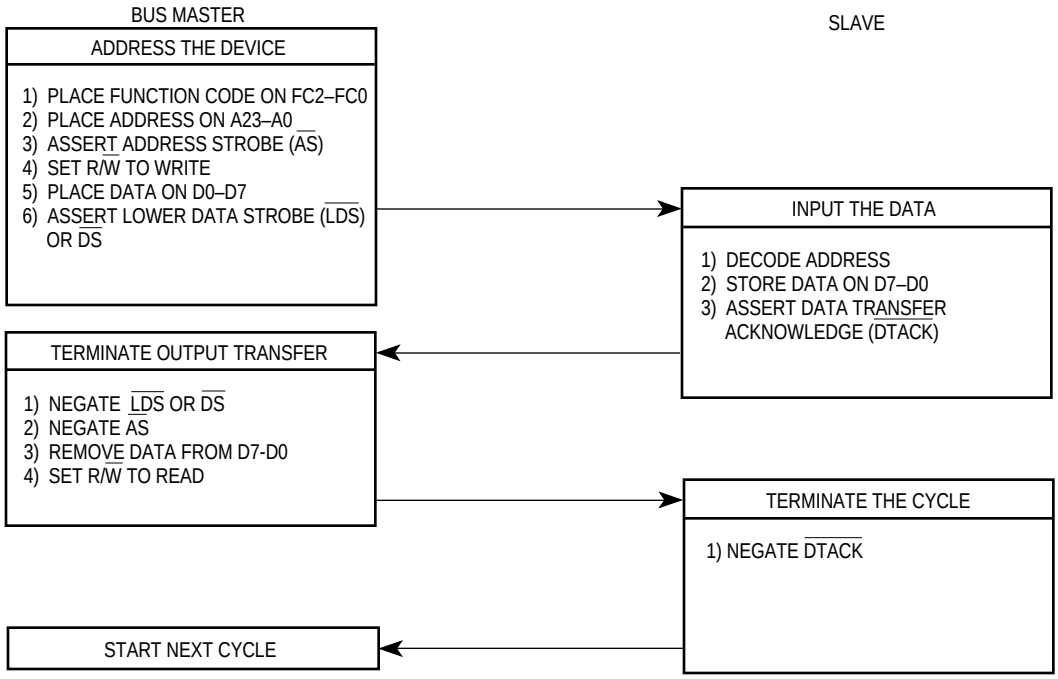


Figure 4-3. Byte Write-Cycle Flowchart

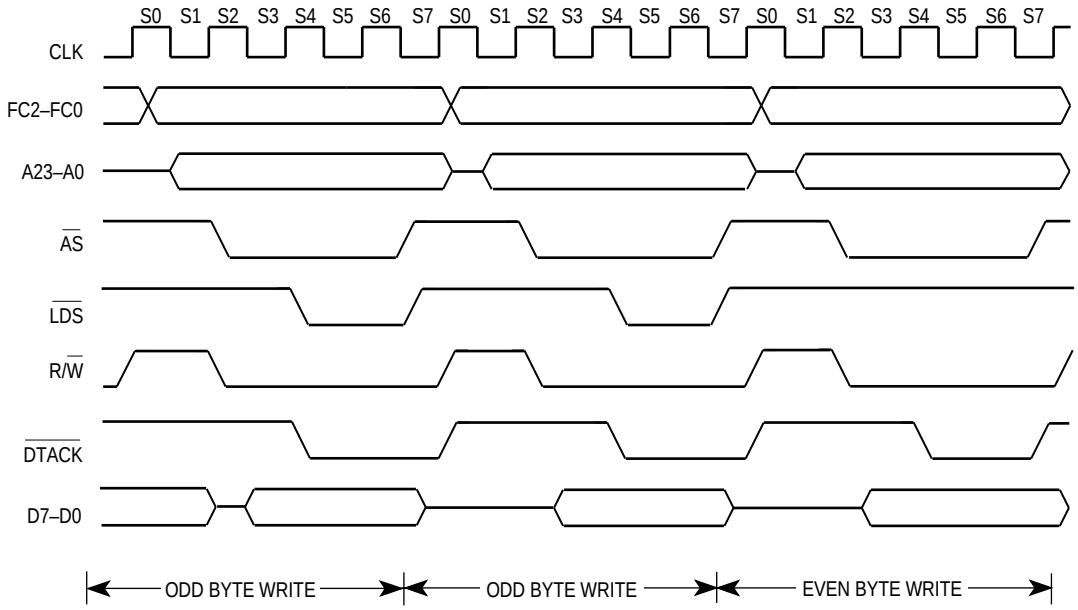


Figure 4-4. Write-Cycle Timing Diagram

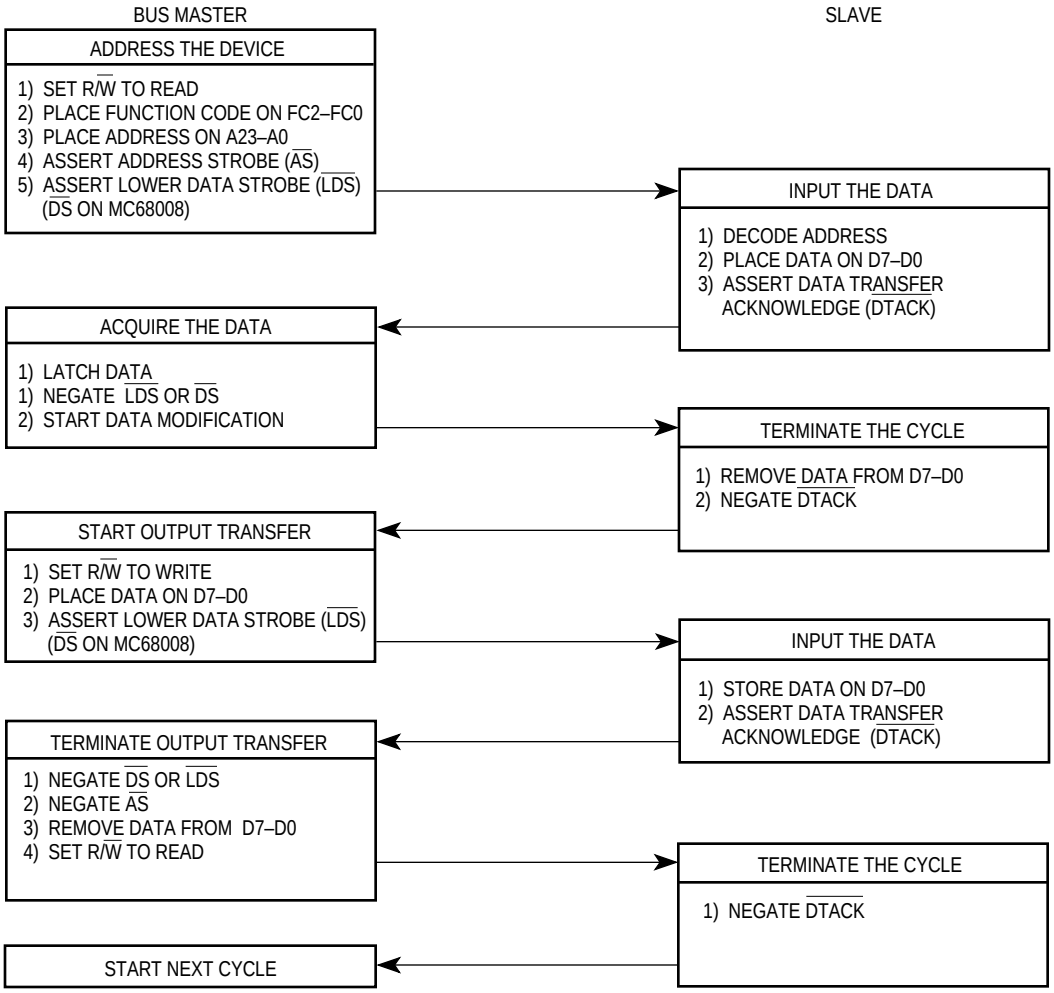


Figure 4-5. Read-Modify-Write Cycle Flowchart

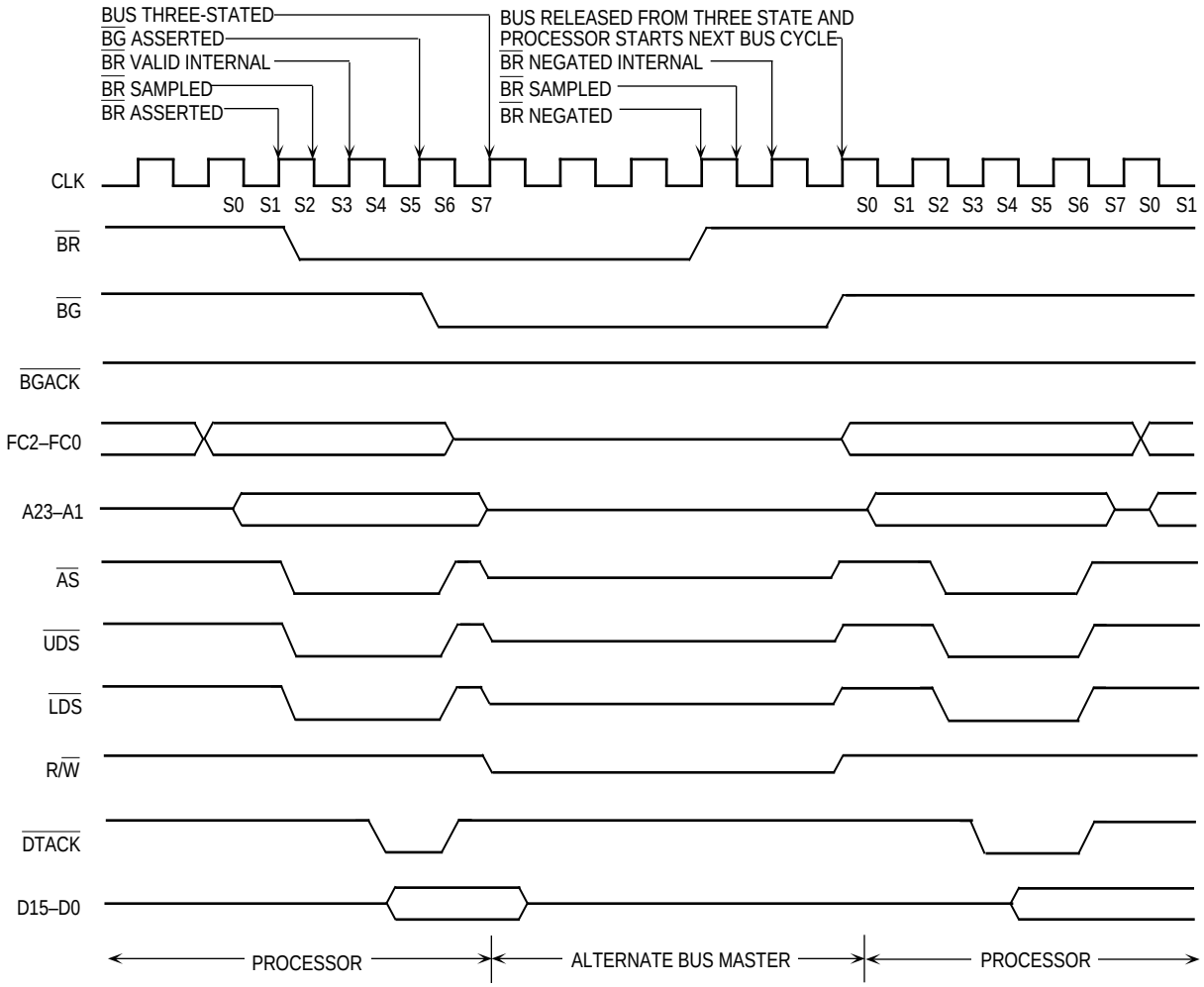


Figure 5-22. 2-Wire Bus Arbitration Timing Diagram—Processor Active

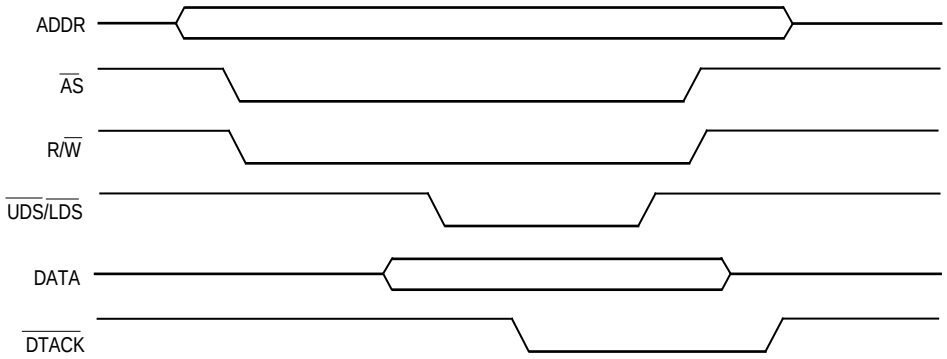


Figure 5-32. Fully Asynchronous Write Cycle

In the asynchronous mode, the accessed device operates independently of the frequency and phase of the system clock. For example, the MC68681 dual universal asynchronous receiver/transmitter (DUART) does not require any clock-related information from the bus master during a bus transfer. Asynchronous devices are designed to operate correctly with processors at any clock frequency when relevant timing requirements are observed.

A device can use a clock at the same frequency as the system clock (e.g., 8, 10, or 12.5, 16, and 20MHz), but without a defined phase relationship to the system clock. This mode of operation is pseudo-asynchronous; it increases performance by observing timing parameters related to the system clock frequency without being completely synchronous with that clock. A memory array designed to operate with a particular frequency processor but not driven by the processor clock is a common example of a pseudo-asynchronous device.

The designer of a fully asynchronous system can make no assumptions about address setup time, which could be used to improve performance. With the system clock frequency known, the slave device can be designed to decode the address bus before recognizing an address strobe. Parameter #11 (refer to **Section 10 Electrical Characteristics**) specifies the minimum time before address strobe during which the address is valid.

In a pseudo-asynchronous system, timing specifications allow $\overline{\text{DTACK}}$ to be asserted for a read cycle before the data from a slave device is valid. The length of time that $\overline{\text{DTACK}}$ may precede data is specified as parameter #31. This parameter must be met to ensure the validity of the data latched into the processor. No maximum time is specified from the assertion of $\overline{\text{AS}}$ to the assertion of $\overline{\text{DTACK}}$. During this unlimited time, the processor inserts wait cycles in one-clock-period increments until $\overline{\text{DTACK}}$ is recognized. Figure 5-33 shows the important timing parameters for a pseudo-asynchronous read cycle.

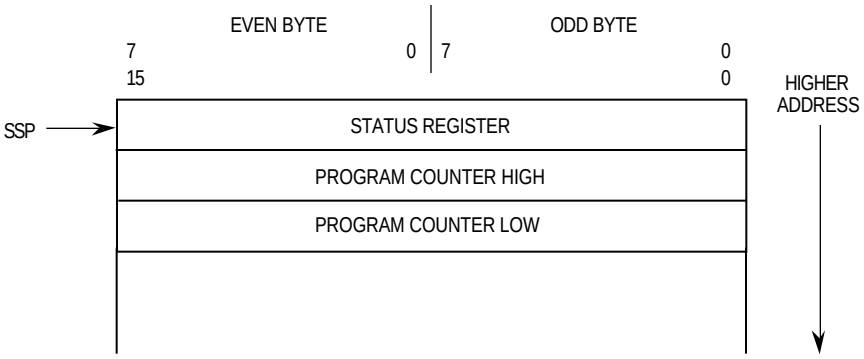


Figure 6-5. Group 1 and 2 Exception Stack Frame (MC68000, MC68HC000, MC68HC001, MC68EC000, and MC68008)

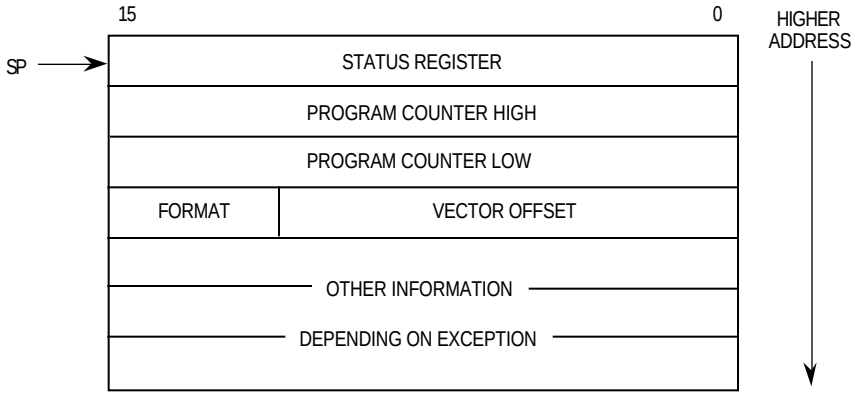


Figure 6-6. MC68010 Stack Frame

SECTION 8
16-BIT INSTRUCTION EXECUTION TIMES

This section contains listings of the instruction execution times in terms of external clock (CLK) periods for the MC68000, MC68HC000, MC68HC001, and the MC68EC000 in 16-bit mode. In this data, it is assumed that both memory read and write cycles consist of four clock periods. A longer memory cycle causes the generation of wait states that must be added to the total instruction times.

The number of bus read and write cycles for each instruction is also included with the timing data. This data is shown as

n(r/w)

where:

n is the total number of clock periods

r is the number of read cycles

w is the number of write cycles

For example, a timing number shown as 18(3/1) means that the total number of clock periods is 18. Of the 18 clock periods, 12 are used for the three read cycles (four periods per cycle). Four additional clock periods are used for the single write cycle, for a total of 16 clock periods. The bus is idle for two clock periods during which the processor completes the internal operations required for the instruction.

NOTE

The total number of clock periods (n) includes instruction fetch and all applicable operand fetches and stores.

8.1 OPERAND EFFECTIVE ADDRESS CALCULATION TIMES

Table 8-1 lists the numbers of clock periods required to compute the effective addresses for instructions. The total includes fetching any extension words, computing the address, and fetching the memory operand. The total number of clock periods, the number of read cycles, and the number of write cycles (zero for all effective address calculations) are shown in the previously described format.

Table 8-1. Effective Address Calculation Times

Addressing Mode		Byte, Word	Long
	Register		
Dn	Data Register Direct	**0**(0/0)	**0**(0/0)
An	Address Register Direct	**0**(0/0)	**0**(0/0)
	Memory		
(An)	Address Register Indirect	**4**(1/0)	**8**(2/0)
(An)+	Address Register Indirect with Postincrement	**4**(1/0)	**8**(2/0)
–(An)	Address Register Indirect with Predecrement	**6**(1/0)	**10**(2/0)
(d_{16}, An)	Address Register Indirect with Displacement	**8**(2/0)	**12**(3/0)
(d_8, An, Xn)*	Address Register Indirect with Index	**10**(2/0)	**14**(3/0)
(xxx).W	Absolute Short	**8**(2/0)	**12**(3/0)
(xxx).L	Absolute Long	**12**(3/0)	**16**(4/0)
(d_8, PC)	Program Counter Indirect with Displacement	**8**(2/0)	**12**(3/0)
(d_{16}, PC, Xn)*	Program Counter Indirect with Index	**10**(2/0)	**14**(3/0)
#<data>	Immediate	**4**(1/0)	**8**(2/0)

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

8.2 MOVE INSTRUCTION EXECUTION TIMES

Tables 8-2 and 8-3 list the numbers of clock periods for the move instructions. The totals include instruction fetch, operand reads, and operand writes. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format.

Table 8-2. Move Byte and Word Instruction Execution Times

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

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

8.9 JMP, JSR, LEA, PEA, AND MOVEM INSTRUCTION EXECUTION TIMES

Table 8-10 lists the timing data for the jump (JMP), jump to subroutine (JSR), load effective address (LEA), push effective address (PEA), and move multiple registers (MOVEM) instructions. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format.

Table 8-10. JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times

Instruction	Size	(An)	(An)+	–(An)	(d_{16},An)	(d_8,An,Xn)+	(xxx).W	(xxx).L	(d_{16} PC)	(d_8, PC, Xn)*
JMP	—	**8**(2/0)	—	—	**10** (2/0)	**14** (3/0)	**10** (2/0)	**12** (3/0)	**10** (2/0)	**14** (3/0)
JSR	—	**16** (2/2)	—	—	**18** (2/2)	**22** (2/2)	**18** (2/2)	**20** (3/2)	**18** (2/2)	**22** (2/2)
LEA	—	4(1/0)	—	—	**8**(2/0)	**12** (2/0)	**8**(2/0)	**12** (3/0)	**8**(2/0)	**12** (2/0)
PEA	—	**12** (1/2)	—	—	**16** (2/2)	**20** (2/2)	**16** (2/2)	**20** (3/2)	**16** (2/2)	**20** (2/2)
MOVEM M → R	Word	**12+4n** (3+n/0)	**12+4n** (3+n/0)	—	**16+4n** (4+n/0)	**18+4n** (4+n/0)	**16+4n** (4+n/0)	**20+4n** (5+n/0)	**16+4n** (4n/0)	**18+4n** (4+n/0)
	Long	**12+8n** (3+2n/0)	**12+8n** (3+n/0)	—	**16+8n** (4+2n/0)	**18+8n** (4+2n/0)	**16+8n** (4+2n/0)	**20+8n** (5+2n/0)	**16+8n** (4+2n/0)	**18+8n** (4+2n/0)
MOVEM R → M	Word	**8+4n** (2/n)	—	**8+4n** (2/n)	**12+4n** (3/n)	**14+4n** (3/n)	**12+4n** (3/n)	**16+4n** (4/n)	— —	— —
	Long	**8+8n** (2/2n)	— —	**8+8n** (2/2n)	**12+8n** (3/2n)	**14+8n** (3/2n)	**12+8n** (3/2n)	**16+8n** (4/2n)	— —	— —

n is the number of registers to move.
*The size of the index register (Xn) does not affect the instruction's execution time.

8.10 MULTIPRECISION INSTRUCTION EXECUTION TIMES

Table 8-11 lists the timing data for multiprecision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next 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 following notation applies in Table 8-11:

Dn — Data register operand
M — Memory 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)

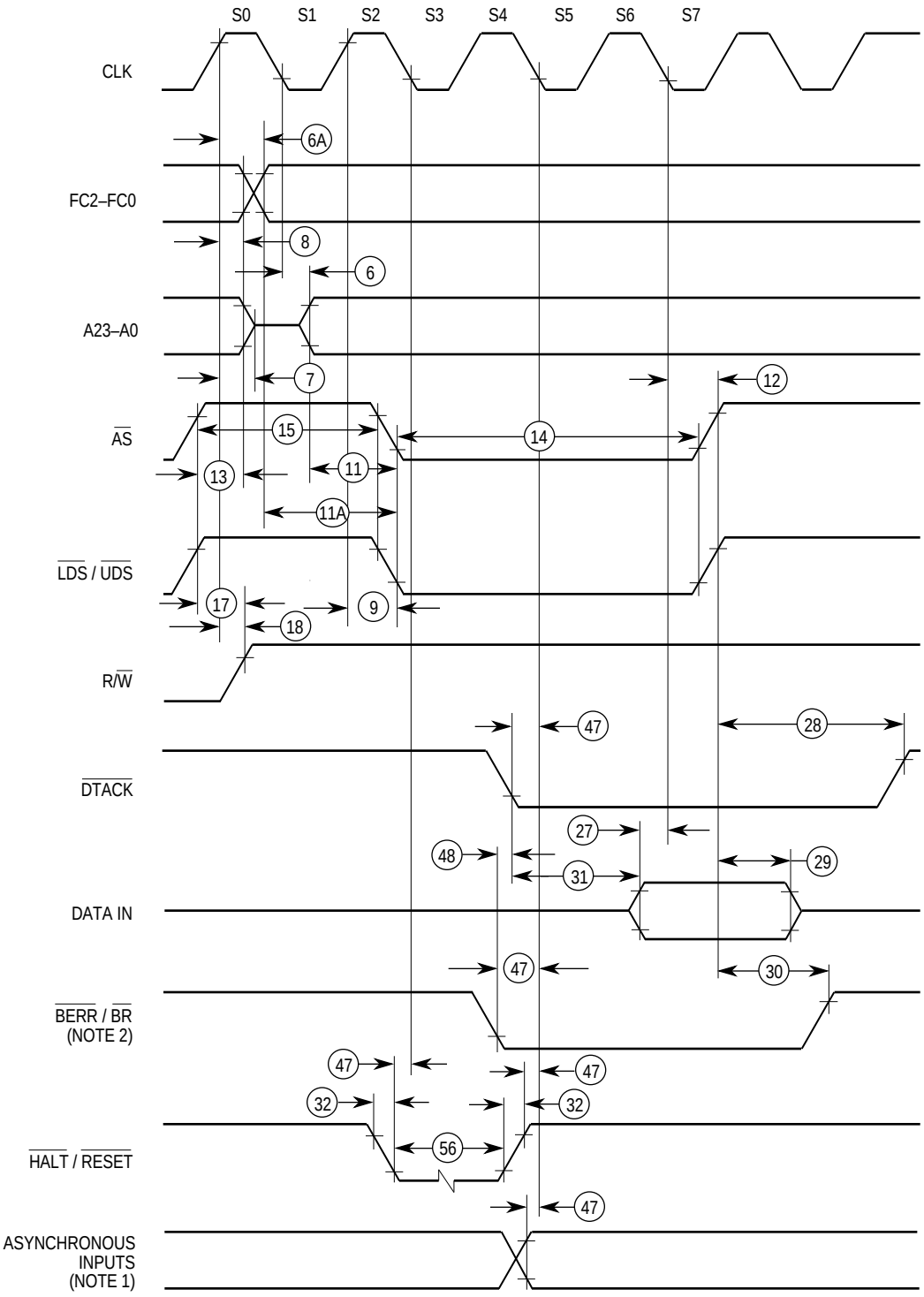
9.6 SHIFT/ROTATE INSTRUCTION EXECUTION TIMES

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

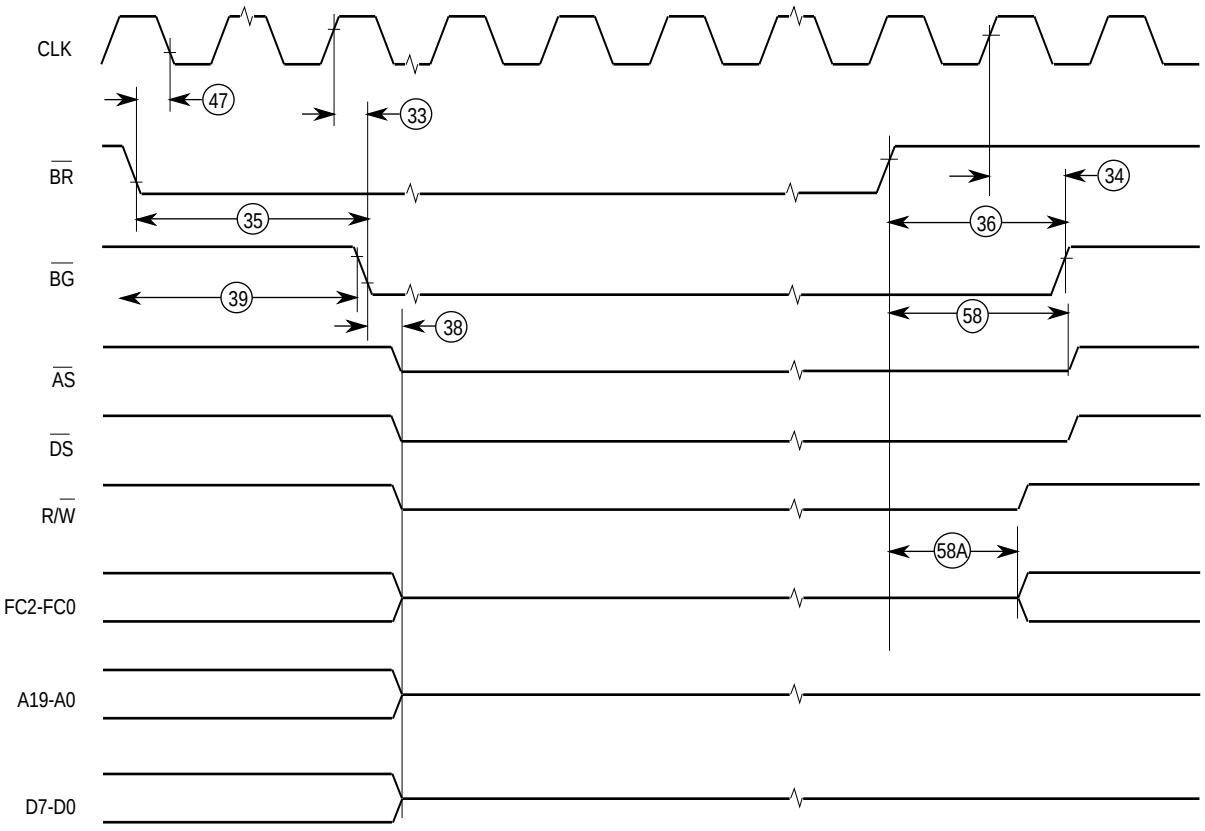
Instruction	Size	Register	Memory*
ASR, ASL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
LSR, LSL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
ROR, ROL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)	—
ROXR, ROXL	Byte, Word	**6+2n**(1/0)	**8**(1/1)+
	Long	**8+2n**(1/0)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

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



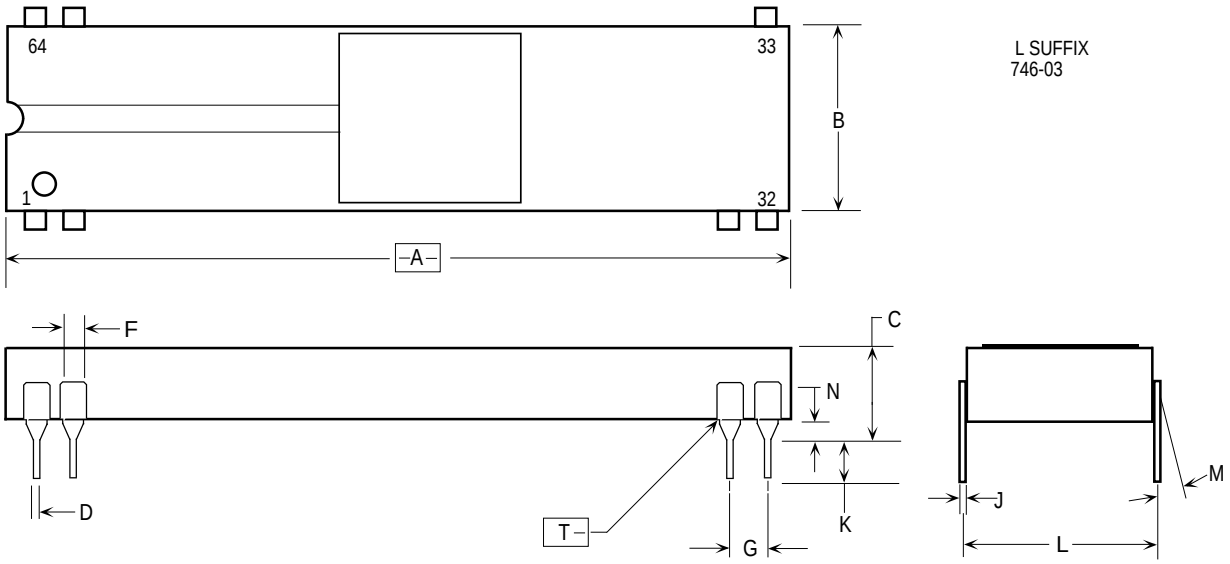
NOTES:
1. Setup time for the asynchronous inputs IPL2–IPL0 and AVEC (#47) guarantees their recognition at the next falling edge of the clock.
2. BR need fall at this time only to insure being recognized at the end of the bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8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

Figure 10-12. MC68EC000 Read Cycle Timing Diagram



NOTES: Waveform measurements for all inputs and outputs are specified at: logic high 2.0 V, logic low = 0.8 V.

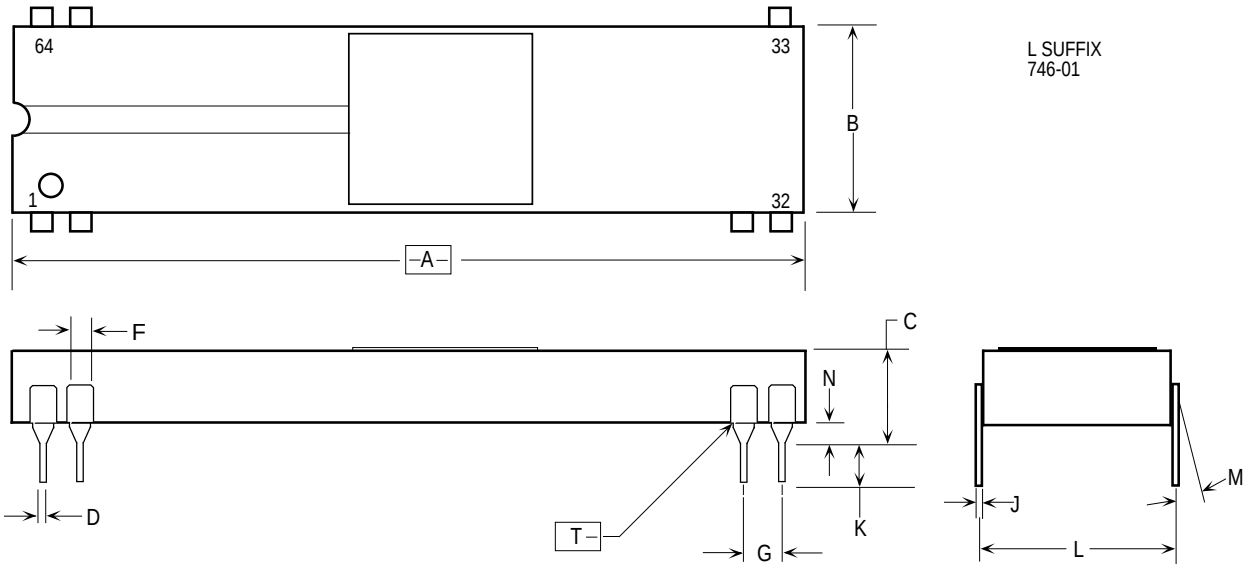
Figure 10-14. MC68EC000 Bus Arbitration Timing Diagram



NOTES:
1. DIMENSION -A- IS DATUM.
2. POSTIONAL TOLERANCE FOR LEADS:
   ⌖ Ø 0.25 (0.010) Ⓜ T A Ⓜ
3. -T- IS SEATING PLANE
4. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5m, 1982.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	60.36	61.56	2.376	2.424
B	14.64	15.34	0.576	0.604
C	3.05	4.32	0.120	0.160
D	3.81	0.533	0.015	0.021
F	.762	1.397	0.030	0.055
G	2.54 BSC		0.100 BSC	
J	0.204	0.330	0.008	0.013
K	2.54	4.19	0.100	0.165
L	15.24 BSC		0.600 BSC	
M	0°	10°	0°	10°
N	1.016	1.524	0.040	0.060

Figure 11-7. Case 740-03—L Suffix



NOTES:
1. DIMENSION -A- IS DATUM.
2. POSTIONAL TOLERANCE FOR LEADS:
   ⌖ ∅ 0.25 (0.010) Ⓜ T A Ⓜ
3. -T- IS SEATING PLANE
4. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	80.52	82.04	3.170	3.230
B	22.25	22.96	0.876	0.904
C	3.05	4.32	0.120	0.160
D	0.38	0.53	0.015	0.021
F	.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	22.61	23.11	0.890	0.910
M	0°	10°	0°	10°
N	1.02	1.52	0.040	0.060

Figure 11-9. Case 746-01—LC Suffix

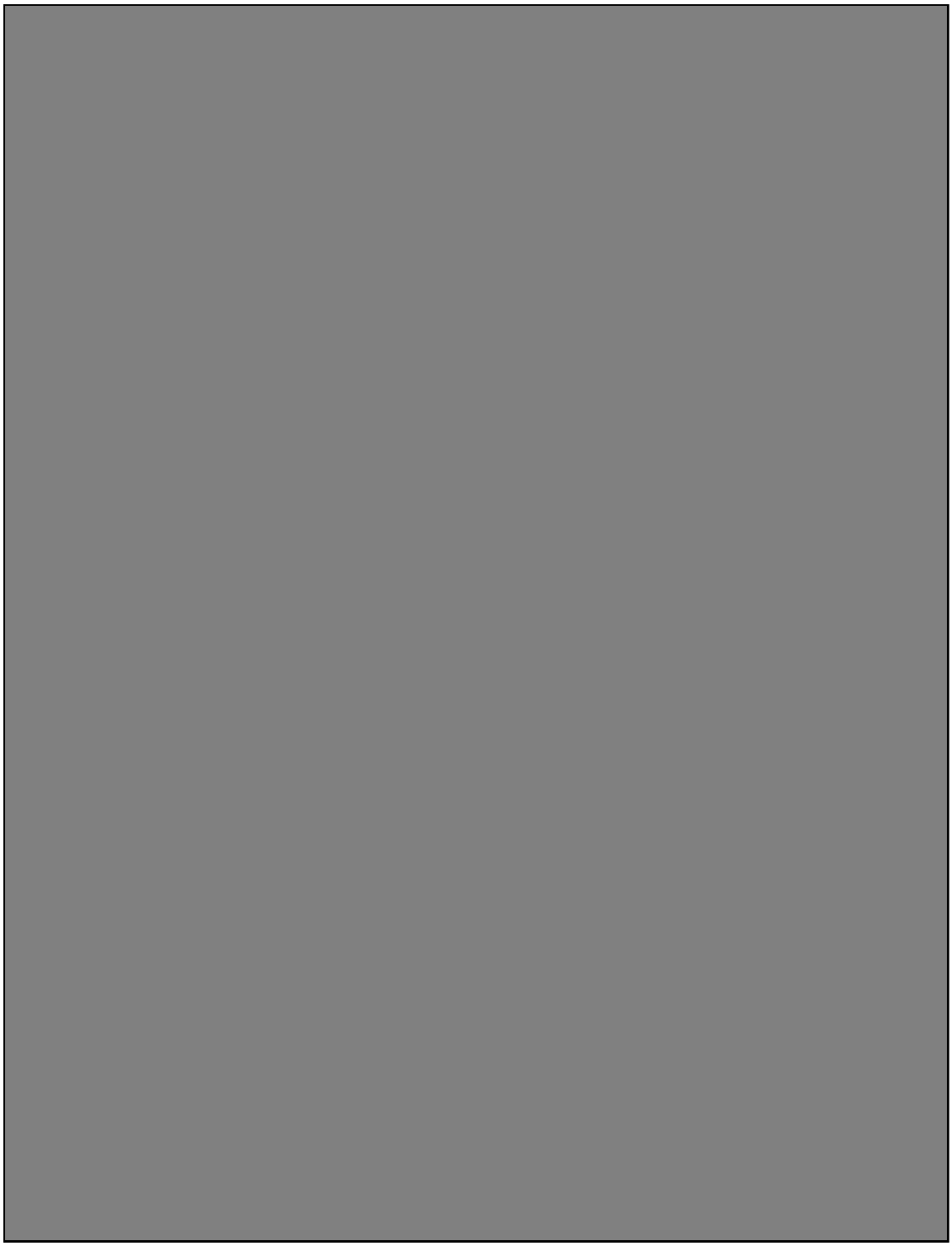


Figure 11-11. Case 765A-05—RC Suffix

processors. Enable has a 60/40 duty cycle; that is, it is low for six system clocks and high for four system clocks. This duty cycle allows $\overline{VPA}$ accesses on successive E pulses.

In the MC68000, MC68HC000, MC68HC001, and the MC68010, $\overline{VMA}$ is provided to indicate synchronization with E. The MC68008 does not provide a $\overline{VMA}$ signal; external circuitry similar to that shown in Figure B-2 using transistor-to-transistor (TTL) logic must be included in the system to provide $\overline{VMA}$. The $\overline{VMA}$ signal indicates to the M6800 devices that the address on the address bus is a valid device address and that the processor is synchronized to the enable clock. The VPA decode input is an active-high signal that is asserted when address strobe $\overline{AS}$ has been asserted and the address on the address bus is that of a peripheral device. The flip-flop on the left sets at the falling edge of E; the flip-flop on the right sets at the next fall of system clock, asserting $\overline{VMA}$. $\overline{VMA}$ remains asserted until the fall of system clock immediately following the negation of VPA decode. Figure B-3 shows the timing for the $\overline{VMA}$ signal provided by this circuitry.

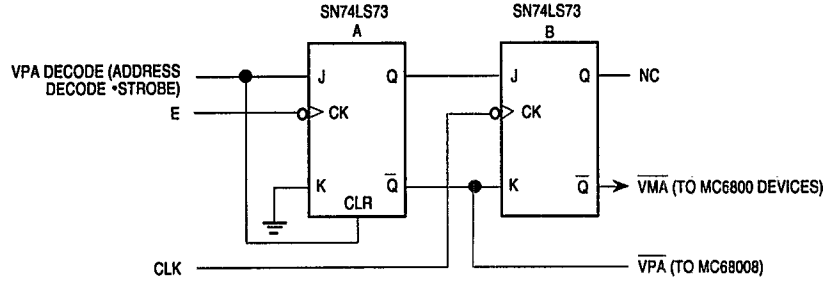


Figure B-2. Example External $\overline{VMA}$ Circuit

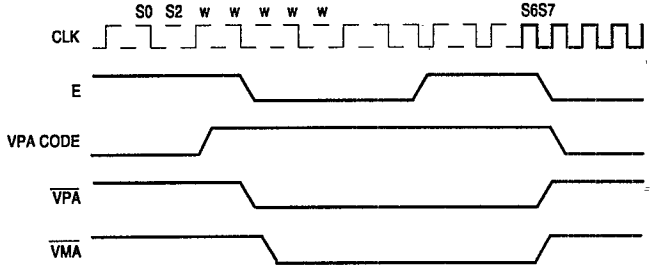


Figure B-3 External $\overline{VMA}$ Timing

B

M6800 cycle timing is shown in Figures B-4 and B-5. At state 0 (S0) in the cycle, the address bus is in the high-impedance state. A function code is asserted on the function code output lines. In state 1 (S1), the address is placed on the address bus. During state 2 (S2), the address strobe ($\overline{AS}$) is asserted to indicate that the address on the bus is valid. If the bus cycle is a read cycle, the upper and/or lower data strobe ($\overline{UDS}$, $\overline{LDS}$) (MC68000/MC68HC000/MC68HC001/MC68010) or data strobe ($\overline{DS}$) (MC68008) is also

interrupt service routine can be located anywhere within the supervisor program address space because the user assigns the vectors in the vector table.

Since $\overline{\text{VMA}}$ is asserted during an autovector operation, care should be taken to prevent an unintended access to the device. An unintended access could occur if the peripheral address were on the address bus during the autovector operation.

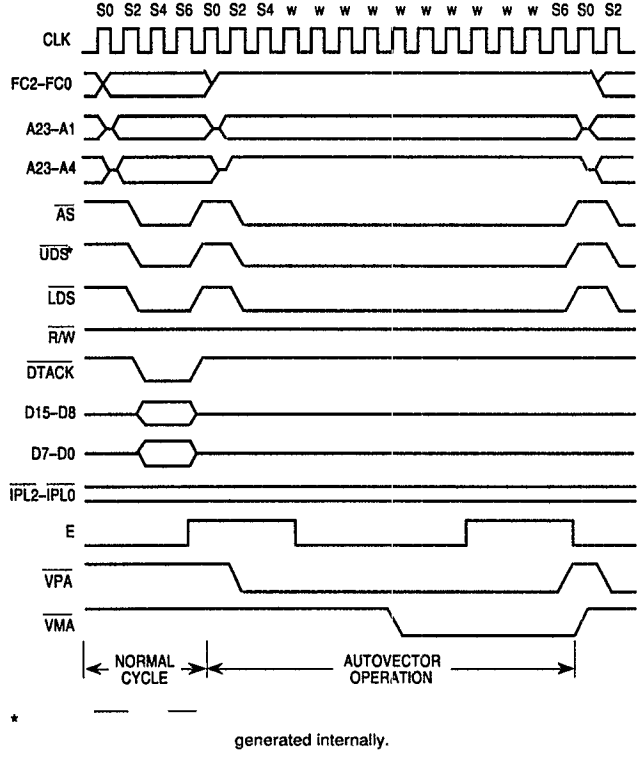


Figure B-6. Autovector Operation Timing Diagram