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	-
Number of Cores/Bus Width	-
Speed	-
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	-
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	-
Operating Temperature	-
Security Features	-
Package / Case	-
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/rochester-electronics/mc68040rc25a

Table 3-1. Updating U-Bit and M-Bit for Page Descriptors

Previous Status		WP Bit	Access Type	Page Descriptor	New Status	
U-Bit	M-Bit			Update Operation	U-Bit	M-Bit
0	0	X	Read	Locked RMW Access to Set U	1	0
0	1			Locked RMW Access to Set U	1	1
1	0			None	1	0
1	1			None	1	1
0	0	0	Write	Write to Set U and M	1	1
0	1			Locked RMW Access to Set U	1	1
1	0			Write to Set M	1	1
1	1			None	1	1
0	0	1		Locked RMW Access to Set U	1	0
0	1			Locked RMW Access to Set U	1	1
1	0			None	1	0
1	1			None	1	1

NOTE: WP indicates the accumulated write-protect status.

An alternate address space access is a special case that is immediately used as a physical address without translation. Because the M68040 implements a merged instruction and data space, the integer unit translates MOVES accesses to instruction address spaces (SFC/DFC = $6 or $2) into data references (SFC/DFC = $5 or $1). The data memory unit handles these translated accesses as normal data accesses. If the access fails due to an ATC fault or a physical bus error, the resulting access error stack frame contains the converted function code in the TM field for the faulted access. Invalidation of the instruction cache line containing the referenced location to maintain cache coherency must precede MOVES accesses that write the instruction address space. The SFC and DFC values and results are listed in Table 3-2.

Table 3-2. SFC and DFC Values

SFC/DFC Value	Results	
	TT	TM
000	10	000
001	00	001
010	00	001
011	10	011
100	10	100
101	00	101
110	00	101
111	10	111

procedure ensures that the first write operation to a page sets the M-bit in both the ATC and the page descriptor in the translation tables, even when a previous read operation to the page had created an entry for that page in the ATC with the M-bit clear.

Physical Address

The upper bits of the translated physical address are contained in this field.

R—Resident

This bit is set if the table search successfully completes without encountering either a nonresident page or a transfer error acknowledge during the search.

S—Supervisor Protected

This bit identifies a pointer table or a page as a supervisor-only table or page. Only programs operating in the supervisor privilege mode are allowed to access the portion of the logical address space mapped by this descriptor when the S-bit is set. If the bit is clear, both supervisor and user accesses are allowed.

U0, U1—User Page Attributes

These user-defined bits are not interpreted by the M68040. U0 and U1 are echoed to the UPA0 and UPA1 signals, respectively, if an external bus transfer results from the access.

V—Valid

When set, this bit indicates the validity of the entry. This bit is set when the M68040 loads an entry. A flush operation by a PFLUSH or PFLUSHA instruction that selects this entry clears the bit.

W—Write Protected

This write-protect bit is set when a W-bit is set in any of the descriptors encountered during the table search for this entry. Setting a W-bit in a table descriptor write protects all pages accessed with that descriptor. When the W-bit is set, a write access or a read-modify-write access to the logical address corresponding to this entry causes an access error exception to be taken immediately.

For each access to a memory unit, the MMU uses the four bits of the logical address located just above the page offset (LA16–LA13 for 8K pages, LA15–LA12 for 4K pages) to index into the ATC. The tags are compared with the remaining upper bits of the logical address and FC2. If one of the tags matches and is valid, then the multiplexer choses the corresponding entry to produce the physical address and status information. The ATC outputs the corresponding physical address to the cache controller, which accesses the data within the cache and/or requests an external bus cycle. Each ATC entry contains a logical address, a physical address, and status bits.

When the ATC does not contain the translation for a logical address, a miss occurs. The MMU aborts the current access and searches the translation tables in memory for the correct translation. If the table search completes without any errors, the MMU stores the

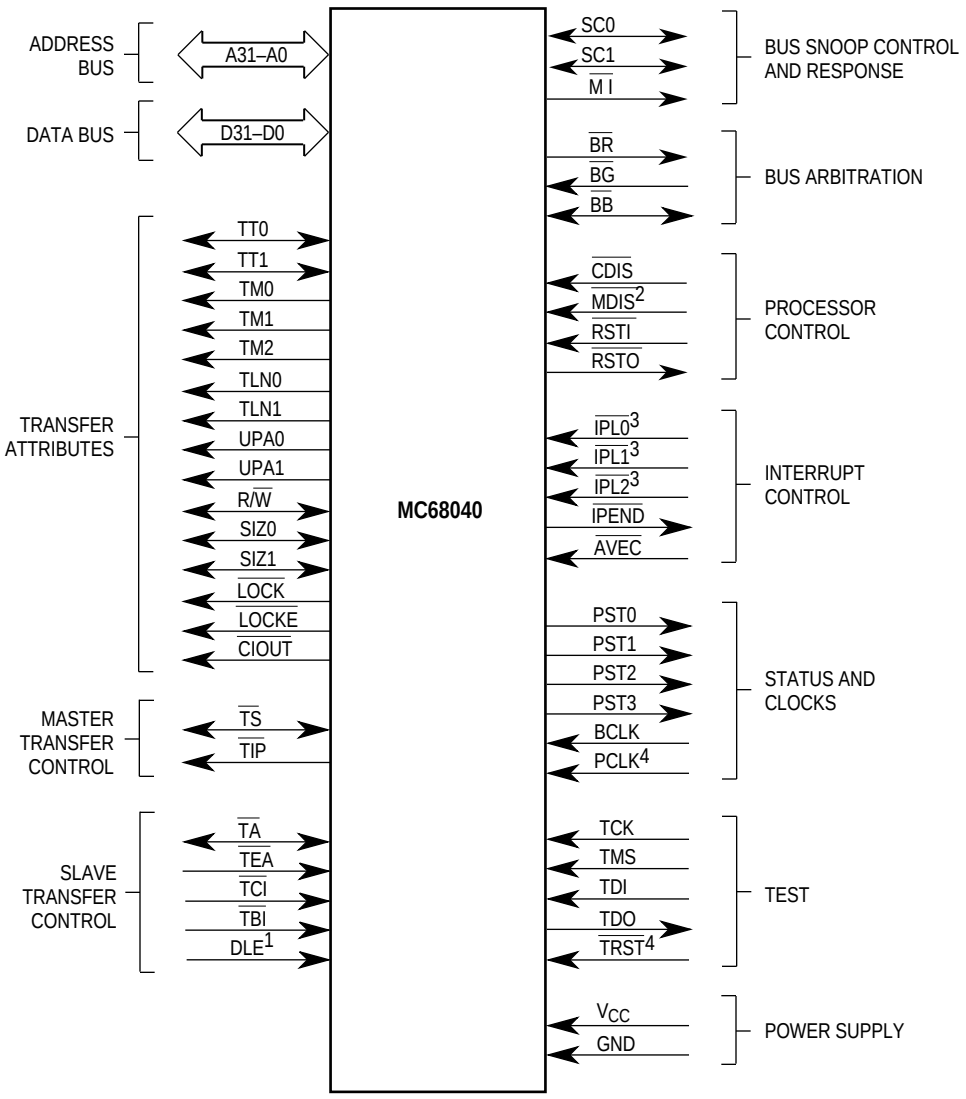
Table 4-3. Instruction-Cache Line State Transitions

Cache Operation	Current State			
	Invalid Cases		Valid Cases	
CPU Read Miss	I1	Read line from memory; supply data to CPU and update cache; go to valid state.	V1	Read line from memory; supply data to CPU and update cache (replacing old line); remain in current state.
CPU Read Hit	I2	Not Possible	V2	Supply data to CPU; remain in current state.
Cache Invalidate or Push (CINV or CPUSH)	I3	No action; remain in current state.	V3	No action; go to invalid state.
Alternate Master Read Hit (Snoop Control = 01 — Leave Dirty)	I4	Not possible; not snooped.	V4	Not possible; not snooped.
Alternate Master Read Hit (Snoop Control = 10 — Invalidate)	I5	Not Possible	V5	No action; go to invalid state.
Alternate Master Write Hit (Snoop Control = 01 — Leave Dirty or Snoop Control = 10 — Invalidate)	I6	Not Possible	V6	No action; go to invalid state.

4.7.2 Data Cache

The IU uses the data cache to store operand data as it generates the data. The data cache supports a line-based protocol allowing individual cache lines to be in one of three states: invalid, valid, or dirty. To maintain coherency with memory, the data cache supports both write-through and copyback modes, specified by the CM field for the page.

Read misses and write misses to copyback pages cause the cache controller to read a new cache line from memory into the cache. If available, an invalid line in the selected set is updated with the tag and data from memory. The line state then changes from invalid to valid by setting the V-bit for the line. If all lines in the set are already valid or dirty, the pseudo-random replacement algorithm is used to select one of the four lines and replace the tag and data contents of the line with the new line information. Before replacement, dirty lines are temporarily buffered and later copied back to memory after the new line has been read from memory. If a snoop access occurs before the buffered line is written to memory, the snoop controller snoops the buffer and the caches. Figure 4-6 illustrates the three possible states for a data cache line, with the possible transitions caused by either the processor or snooped accesses. Transitions are labeled with a capital letter, indicating the previous state, followed by a number indicating the specific case listed in Table 4-4.



NOTES:
1. This signal is only available on the MC68040.
2. This signal is not available on the MC68EC040 and MC68EC040V.
3. These signals are different on power-up for the MC68LC040 and MC68EC040.
4. These signals are not available on the MC68040V and MC68EC040V.

Figure 5-1. Functional Signal Groups

5.1 ADDRESS BUS (A31–A0)

These three-state bidirectional signals provide the address of the first item of a bus transfer (except for acknowledge transfers) when the M68040 is the bus master. When an alternate bus master is controlling the bus, the processor examines (snoops) these signals to determine whether the processor should intervene in the access to maintain cache coherency.

The level on CDIS can select a multiplexed bus mode during processor reset, which allows the address bus and data bus to be physically tied together for multiplexed bus

The DRVCTL.T instruction is intended to be used in test applications in conjunction with the EXTEST and SHUTDOWN instructions and not for system applications. It therefore differs from DRVCTL.S in that this instruction invokes the keep-alive clock, asserts the internal reset, and the test logic, not the system logic, has control of the I/O pins.

When the system logic has control of signal pin I/O directions and levels, the drive control latch is loaded from the IPL2–IPL0 pins during the negation of RSTI. DRVCTL.T overwrites this value with boundary scan data in the update-DR state. The selected output driver state remains unchanged if only the DRVCTL.T, EXTEST, or SHUTDOWN instructions are invoked. If an instruction other than one of these three is executed, the system logic protocol regains control of the output driver state and overwrites the value that the DRVCTL.T instruction previously defined.

Note that the output drive control state does not change while the 1149.1A instruction is one of the three instructions DRVCTL.T, EXTEST, or SHUTDOWN. If DRVCTL.T changes the output driver state and then the test-logic-reset state is entered, the instruction shift register is reset to BYPASS, and the system logic can change the output driver state.

6.2.5 SHUTDOWN

This instruction provides an opcode for automatic test pattern generation (ATPG) programs to cope with the clocking protocol required to stop the system clocks. This instruction asserts internal system reset, activates an internal keep alive clock, and selects the bypass register. Internal decoding of the instruction selects the bypass register, and the test logic, not the system logic, has control of the I/O ports. Note that initializing the boundary scan data register and then selecting the SHUTDOWN instruction provides a clamping function. The test logic controls the I/O state, and the bypass register is selected.

6.2.6 PRIVATE

Motorola reserves this instruction for manufacturing use. The instruction does not change pin I/O as defined for system operation.

6.2.7 DRVCTL.S

The DRVCTL.S instruction controls the output driver selection on a pin-by-pin basis. This instruction allows data in the boundary scan register to select the output driver during the update-DR state when the system logic has control of the signal I/O directions and levels. A logic zero selects the large buffer or driver; a logic one selects the small buffer or driver (see Table 6-1).

The DRVCTL.S instruction is intended to be used in system applications and not in test applications. In system applications, the system logic has control of the signal pin I/O directions and levels; whereas, in test applications, the 1149.1A test logic has control of it. It therefore differs from DRVCTL.T in that this instruction does not invoke the internal keep alive clock, it does not assert the internal reset, and the system logic, not the test logic,

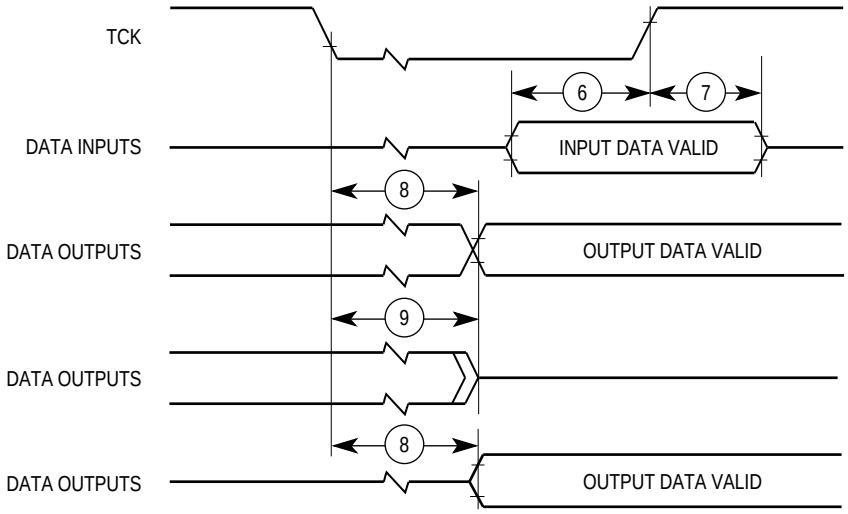


Figure 6-10. Boundary Scan Timing Diagram

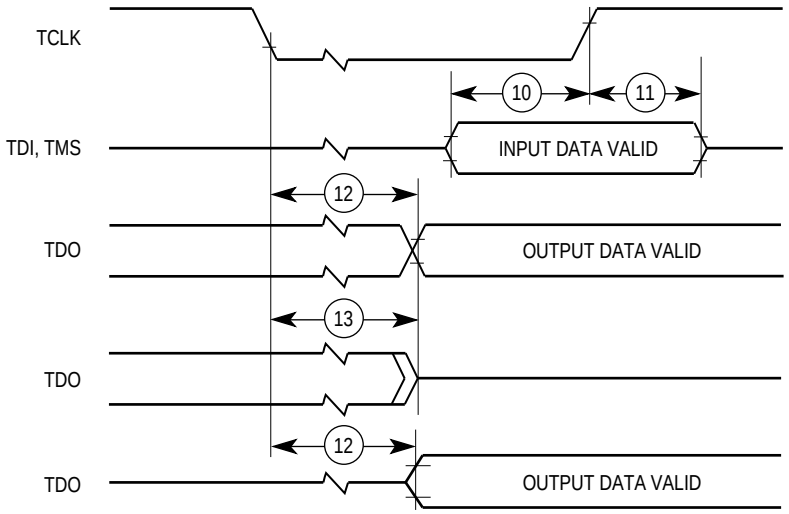
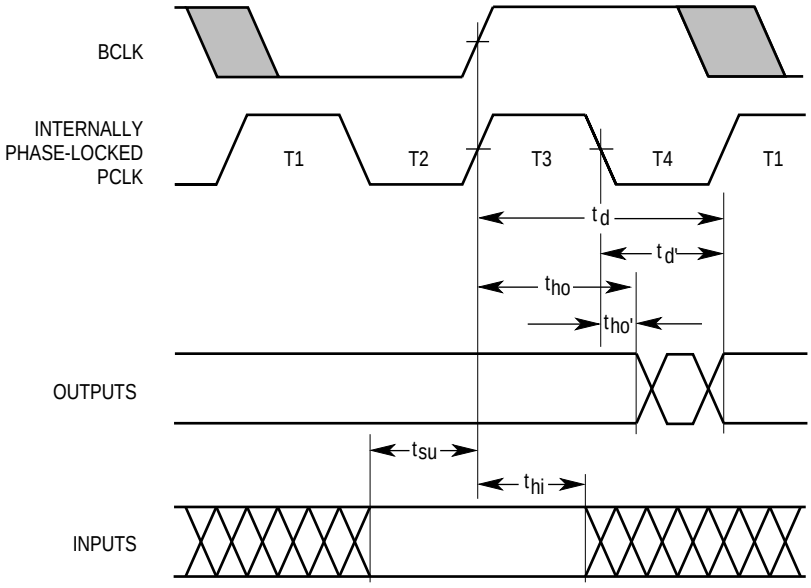


Figure 6-11. Test Access Port Timing Diagram

negate logic levels. The exceptions to this rule are the $\overline{TIP}$, $\overline{TA}$, and $\overline{BB}$ signals that transition between logic levels during T4 but transition from a driven state to a high-impedance state during T1. The input setup time (t_{su}), input hold time (t_{hi}), output hold time (t_{ho}), and delay time (t_d) illustrated in Figure 7-1 are described in the AC electrical timing specifications in **Section 11 MC68040 Electrical and Thermal Characteristics**.



NOTES:
1. t_d = Propagation delay of signal relative to BLK rising edge.
2. $t_{d'}$ = Propagation delay of signal relative to PCLK falling edge; $t_{d'} = t_d - 1/2$ PCLK except for $\overline{TIP}$, $\overline{TA}$, $\overline{BB}$ when used as outputs.
3. t_{ho} = Output hold time relative to BCLK rising edge.
4. $t_{ho'}$ = Output hold time relative to BCLK rising edge; $t_{ho'} = t_h - 1/2$ PCLK.
5. t_{su} = Required input setup time relative to BCLK rising edge.
6. t_{hi} = Required input hold time relative to BCLK rising edge.

Figure 7-1. Signal Relationships to Clocks

Inputs to the M68040 (other than the $\overline{IPL2}$–$\overline{IPL0}$ and $\overline{RSTI}$ signals) are synchronously sampled and must be stable during the sample window defined by t_{su}, t_{hi}, and t_{ho} (see Figure 7-1) to guarantee proper operation. The asynchronous $\overline{IPLx}$ and $\overline{RSTI}$ signals are also sampled on the rising edge of BCLK, but are internally synchronized to resolve the input to a valid level before using it. Since the timing specifications for the M68040 are referenced to the rising edge of BCLK, they are valid only for the specified operating frequency and must be scaled for lower operating frequencies.

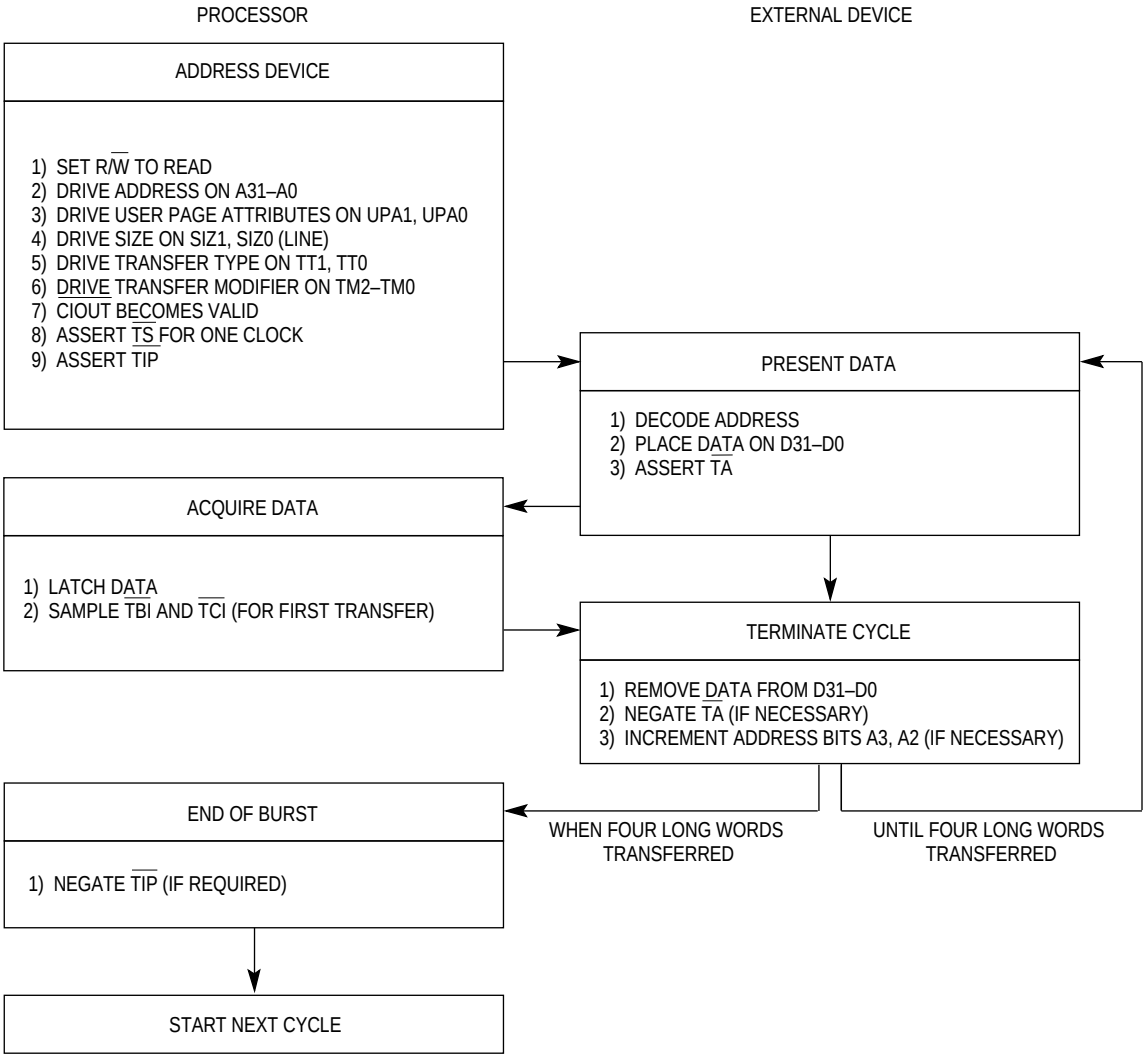


Figure 7-10. Line Read Transfer Flowchart

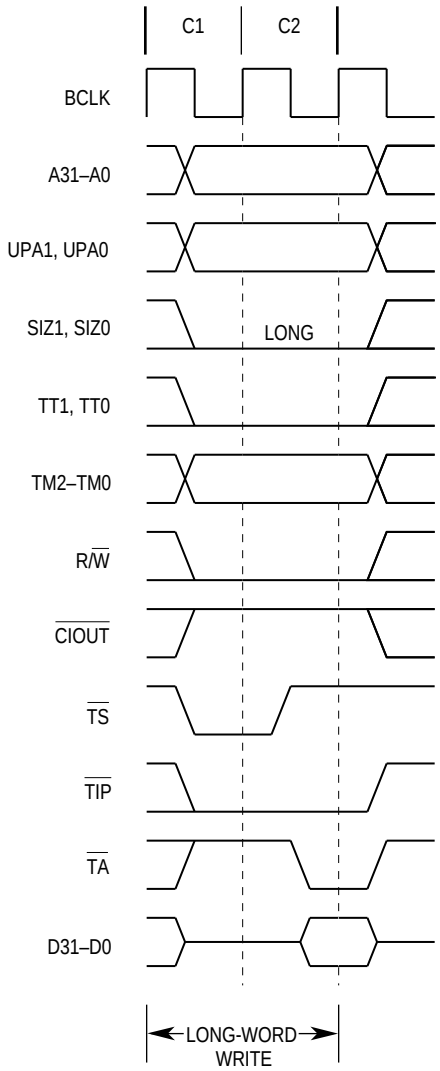


Figure 7-15. Long-Word Write Transfer Timing

Clock 1 (C1)

The write cycle starts in C1. During the first half of C1, the processor places valid values on the address bus and transfer attributes. For user and supervisor mode accesses, which the corresponding memory unit translates, the UPAx signals are driven with the values from the U1 and U0 bits for the area. The TTx and TMx signals identify the specific access type. The R/$\overline{W}$ signal is driven low for a write cycle. $\overline{CIOUT}$ is asserted if the access is identified as noncachable or if the access references an alternate address space. Refer to **Section 3 Memory Management Unit (Except MC68EC040 and MC68EC040V)** for information on the M68040 and MC68LC040 memory units and **Appendix B MC68EC040** for information on the MC68EC040 memory unit.

The processor asserts $\overline{TS}$ during C1 to indicate the beginning of a bus cycle. If not already asserted from a previous bus cycle, the $\overline{TIP}$ signal is also asserted at this time to indicate that a bus cycle is active.

Clock 2 (C2)

During the first half of the clock after C1, the processor negates $\overline{TS}$ and drives the appropriate bytes of the data bus with the data to be written. All other bytes are driven with undefined values. The selected device uses R/$\overline{W}$, SIZ1, SIZ0, A1, A0, and $\overline{CIOUT}$ to latch only the required information on the data bus. With the exception of R/$\overline{W}$ and $\overline{CIOUT}$, these signals also select any or all of the bytes (D31–D24, D23–D16, D15–D8, and D7–D0). If the first clock after C1 is not a wait state, then the selected peripheral device asserts the $\overline{TA}$ signal.

At the end of the first clock cycle after C1, the processor samples the level of $\overline{TA}$, terminating the bus cycle if $\overline{TA}$ is asserted. If $\overline{TA}$ is not recognized asserted at the end of the clock cycle, the processor ignores the data and inserts a wait state instead of terminating the transfer. The processor continues to sample $\overline{TA}$ on successive rising edges of BCLK until $\overline{TA}$ is recognized asserted. The data bus then three-states and the bus cycle ends.

When the processor recognizes $\overline{TA}$ at the clock edge and terminates the bus cycle, $\overline{TIP}$ remains asserted if the processor is ready to begin another bus cycle. Otherwise, the processor negates $\overline{TIP}$ during the first half of the next clock. The processor also three-states the data bus during the first half of the next clock following termination of the write transfer.

7.4.4 Line Write Transfers

The processor uses line write bus cycles to access a 16-byte operand for a MOVE16 instruction and to support cache line pushes. Both burst and burst-inhibited transfers are supported. Figures 7-16 and 7-17 illustrate a flowchart and functional timing diagram for a line write bus cycle.

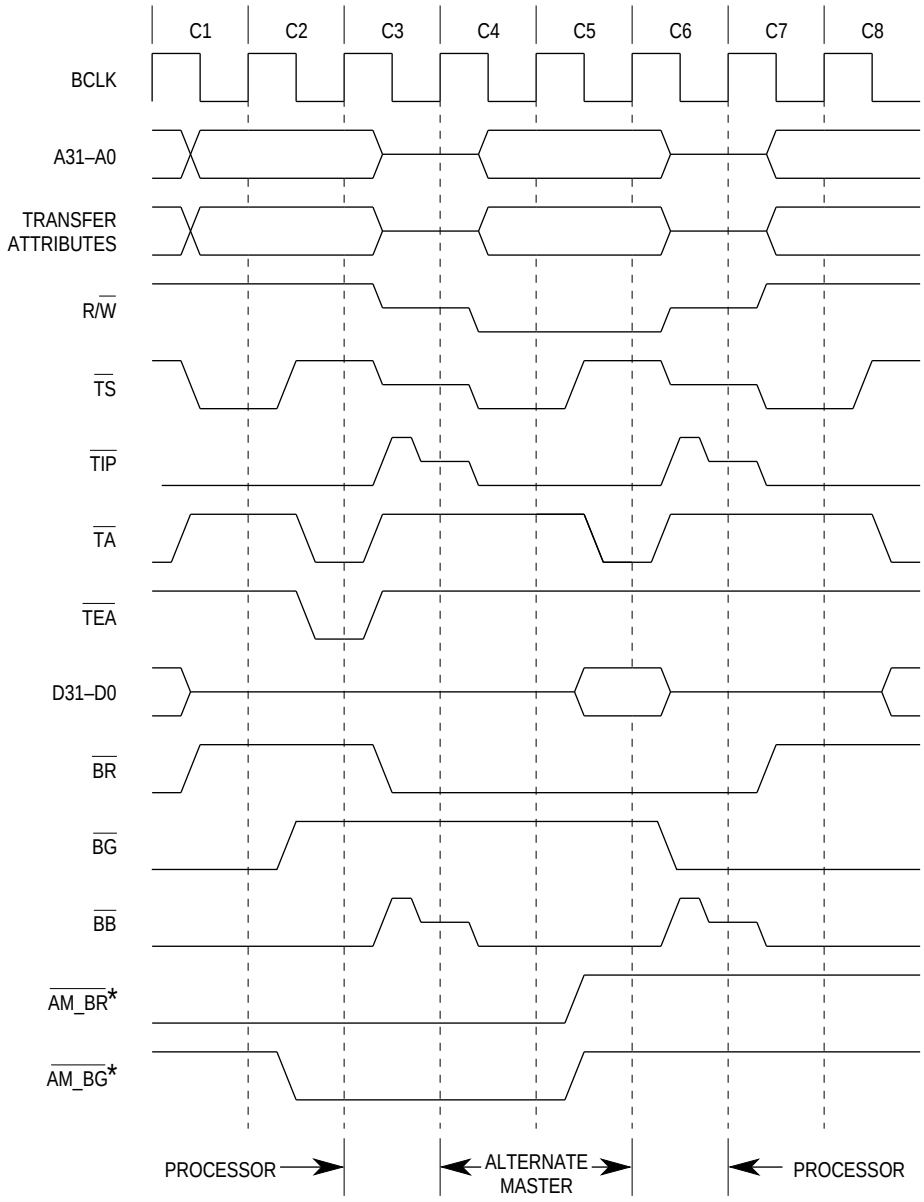
The M68040 takes an interrupt exception for a pending interrupt within one instruction boundary after processing any other pending exception with a higher priority. Thus, the M68040 executes at least one instruction in an interrupt exception handler before recognizing another interrupt request. The following paragraphs describe the various kinds of interrupt acknowledge bus cycles that can be executed as part of interrupt exception processing. Table 7-4 provides a summary of the possible interrupt acknowledge terminations and the exception processing results.

Table 7-4. Interrupt Acknowledge Termination Summary

TA	TEA	AVEC	Termination Condition
High	High	Don't Care	Insert Waits
High	Low	Don't Care	Take Spurious Interrupt Exception
Low	High	High	Latch Vector Number on D7–D0 and Take Interrupt Exception
Low	High	Low	Take Autovectored Interrupt Exception
Low	Low	Don't Care	Retry Interrupt Acknowledge Cycle

7.5.1.1 INTERRUPT ACKNOWLEDGE BUS CYCLE (TERMINATED NORMALLY). When the M68040 processes an interrupt exception, it performs an interrupt acknowledge bus cycle to obtain the vector number that contains the starting location of the interrupt exception handler. Some interrupting devices have programmable vector registers that contain the interrupt vectors for the exception handlers they use. Other interrupting conditions or devices cannot supply a vector number and use the autovector bus cycle described in **7.5.1.2 Autovector Interrupt Acknowledge Bus Cycle**.

Figure 7-33 illustrates a functional timing diagram for an arbitration of a relinquish and retry operation. Figure 7-34 is a functional timing diagram for implicit ownership of the bus. In Figure 7-33, the processor read access that begins in C1 is terminated at the end of C2 with a retry request and $\overline{BG}$ negated, forcing the processor to relinquish the bus and allow the alternate master to access the bus. Note that the processor reasserts $\overline{BR}$ during C3 since the original access is pending again. After alternate bus master ownership, the bus is granted to the processor to allow it to retry the access beginning in C7.



*AM indicates the alternate bus master.

Figure 7-33. Arbitration During Relinquish and Retry Timing

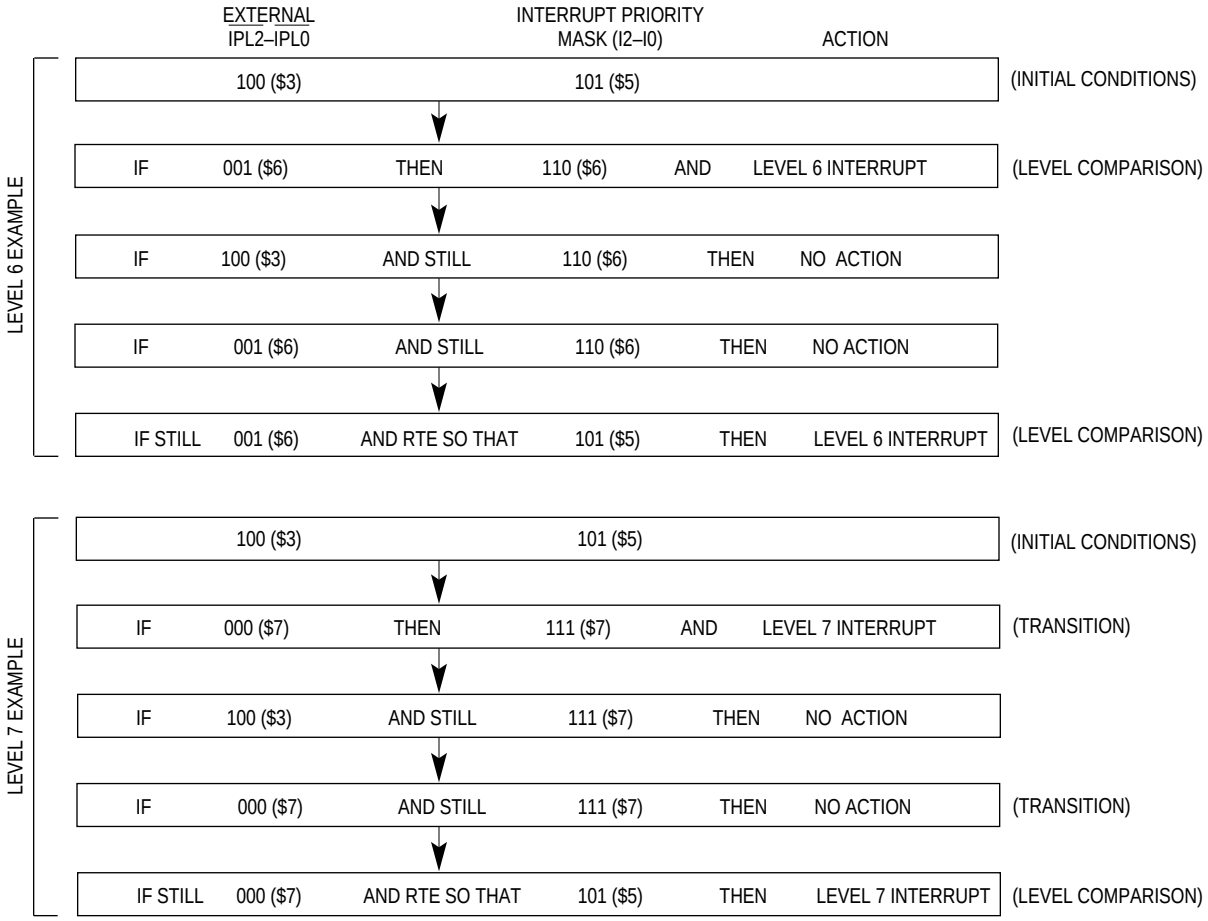


Figure 8-3. Interrupt Recognition Examples

Note that a mask value of 6 and a mask value of 7 both inhibit request levels of 1–6 from being recognized. In addition, neither masks a transition to an interrupt request level of 7. The only difference between mask values of 6 and 7 occurs when the interrupt request level is 7 and the mask value is 7. If the mask value is lowered to 6, a second level 7 interrupt is recognized.

External circuitry can chain or otherwise merge signals from devices at each level, allowing an unlimited number of devices to interrupt the processor. When several devices are connected to the same interrupt level, each device should hold its interrupt priority level constant until its corresponding interrupt acknowledge bus cycle ensures that all requests are processed. Refer to **Section 7 Bus Operation** for details on the interrupt acknowledge cycle.

zeros, infinities, or NANs separately from normal data types. The floating-point condition codes allow users to efficiently detect and handle these special values.

9.2.3.2 QUOTIENT BYTE. The quotient byte (see Figure 9-4) provides compatibility with the MC68881/MC68882 FPU. This byte contains the seven least significant bits of the unsigned quotient as well as the sign of the entire quotient.

The quotient bits can be used in argument reduction for transcendentals and other functions. For example, seven bits are more than enough to determine the quadrant of a circle in which an operand resides. The quotient field (bits 22–16) remains set until the user clears it.

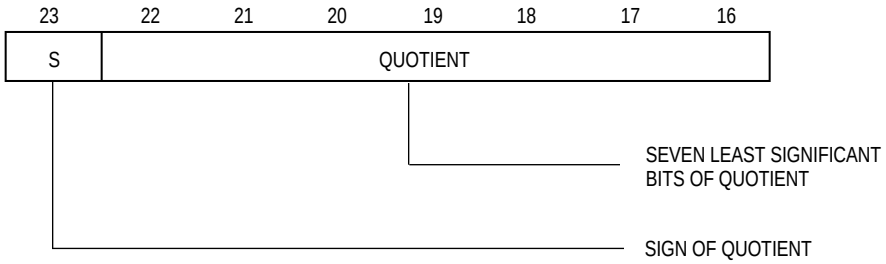


Figure 9-4. FPSR Quotient Byte

9.2.3.3 EXCEPTION STATUS BYTE. The EXC byte (see Figure 9-5) contains a bit for each floating-point exception that can occur during the most recent arithmetic instruction or move operation. The start of most operations clears this byte; however, operations that cannot generate floating-point exceptions do not clear this byte. An exception handler can use this byte to determine which floating-point exception(s) caused a trap.

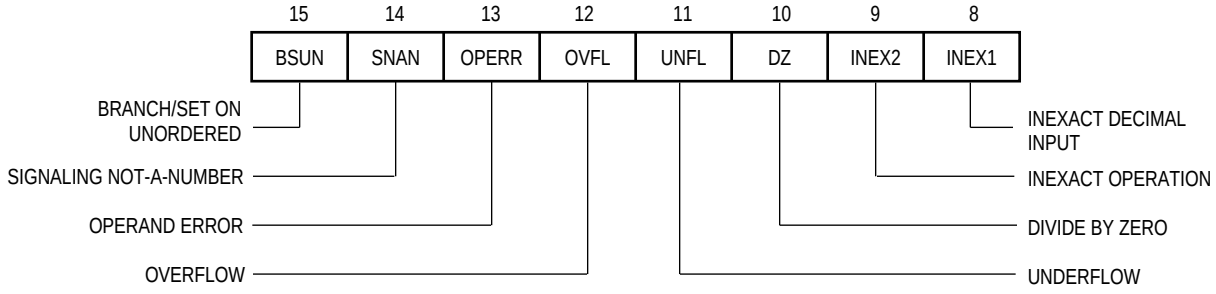


Figure 9-5. FPSR Exception Status Byte

9.2.3.4 ACCRUED EXCEPTION (AEXC) BYTE. The AEXC byte contains five exception bits (see Figure 9-6) that the IEEE 754 standard requires for exception disabled operations. These exceptions are logical combinations of the bits in the EXC byte. The AEXC byte contains the history of all floating-point exceptions that have occurred since the user last cleared the AEXC byte. In normal operations, only the user clears this byte by writing to the FPSR; however, a reset or a restore operation of the null state can also clear the AEXC byte.

9.8 FLOATING-POINT STATE FRAMES

All floating-point arithmetic exception handlers must have FSAVE as the first floating-point instruction; any other floating-point instruction causes another exception to be reported. Once the FSAVE instruction has executed, the exception handler should use only the FMOVEM instruction to read or write to the floating-point data registers since FMOVEM cannot generate further exceptions or change the FPCR.

The FPU executes an FSAVE instruction to save the current floating-point internal state for context switches and floating-point exception handling. When an FSAVE is executed, the processor waits until the FPU either completes execution of all current instructions or is unable to perform further processing due to a pending exception that must be serviced. Any exceptions generated during this time are not reported and are saved in the resulting busy state frame.

Four state frames can be generated as a result of an FSAVE instruction: busy, null, idle, and unimplemented floating-point instruction. When an unimplemented floating-point exception occurs, the FSAVE generates a 26-word unimplemented instruction state frame. When an unsupported data type exception occurs, the FSAVE generates a 50-word busy state frame. All floating-point arithmetic exceptions causes the FSAVE to generate either the 26-word unimplemented instruction state frame or the 50-word busy state frame. For a hardware reset or an FRESTORE of a null state frame, the FSAVE instruction generates a null state frame. This null state frame is generated until the first nonconditional floating-point instruction is executed (conditionals include FNOP, FBcc, FDBcc, FScc, and FTRAPcc). Floating-point conditional instructions do not set an internal flag, which changes the state frame from null to idle. If these instructions are the only ones executed after a reset or an FRESTORE of a null state frame, then when FSAVE is executed, it stacks a null state frame instead of an idle state frame. Note that this function is different from that of the MC68881 and MC68882, and software must be aware of this difference if compatibility with the MC68881 and MC68882 is desired. Once a nonconditional floating-point instruction is executed, an FSAVE generates an idle state frame. The idle state frame is generated whenever the FPU has no exceptions pending. An idle state frame is saved if no exceptions are pending and at least one instruction has been executed since the last hardware reset or FRESTORE of a null state frame. A 26-word unimplemented floating-point instruction state frame is saved if the last instruction was an unimplemented floating-point instruction. Figure 9-10 illustrates each of these state frames, followed by definitions for each of the fields listed in alphabetical order.

NOTE

The notation [XX–XX] indicates the length of the field but does not indicate the field's actual location. [XX, XX–XX] indicates that one bit of the field is located separately or termed differently from the other bits. This notation is for convenience of explanation only. For example, WBTM [65–34] indicates that WBTM is 32 bits long and gives a reference to each bit in WBTM without giving its actual location in the state frame. For the actual locations refer to Figure 9-10.

10.6 INTEGER UNIT INSTRUCTION TIMINGS (Continued)

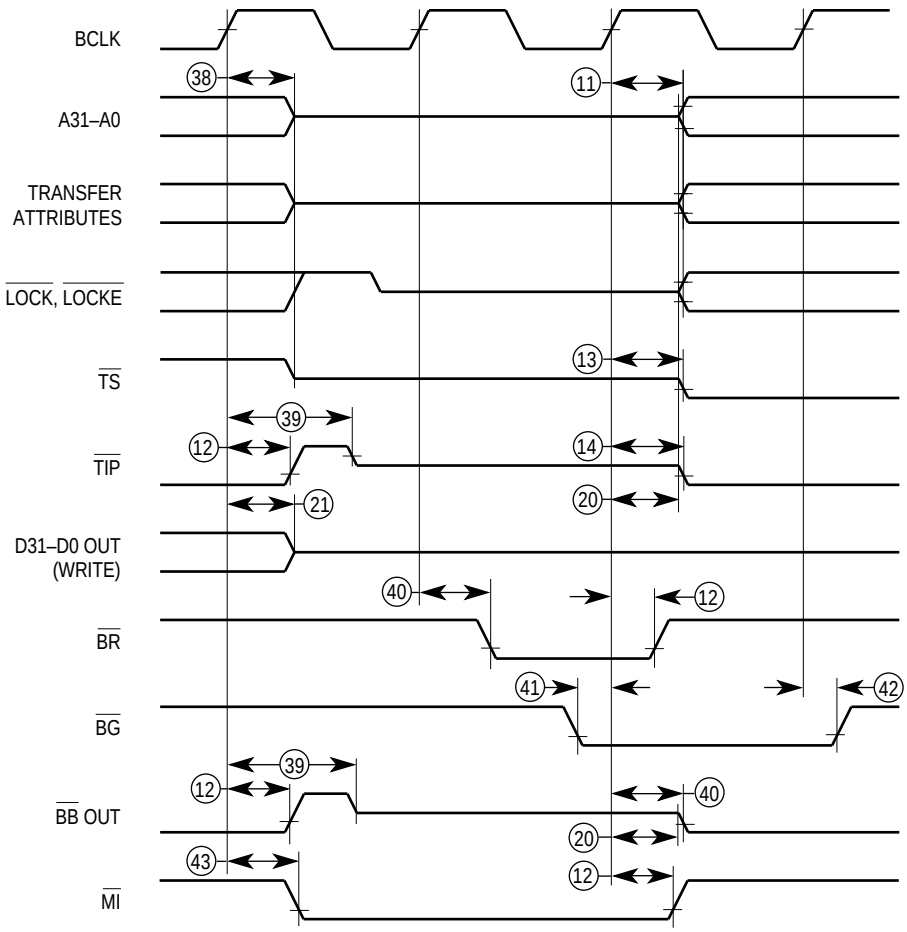
Addressing Mode	DIVS.W, DIVU.W*		DIVS.L, DIVU.L, DIVSL.L, DIVUL.L*		JMP	
	<ea> Calculate	Execute	<ea> Calculate	Execute	<ea> Calculate	Execute
Dn	8	27	9	44	—	—
An	—	—	—	—	—	—
(An)	8	27	9	44	3	2_L + 1
(An)+	8	27	9	44	—	—
–(An)	8	27	9	44	—	—
(d_{16},An)	8	27	11	2_L + 44	4	3_L + 1
(d_{16},PC)	11	3_L + 27	12	3_L + 44	6	5_L + 1
(xxx).W, (xxx).L	8	27	11	2_L + 44	3	2_L + 1
#<xxx>	8	27	10	1_L + 44	—	—
(d_8,An,Xn)	11	30	12	47	6	6
(d_8,PC,Xn)	12	1_L + 30	13	1_L + 47	7	1_L + 6
(BR,Xn)	13	1_L + 31	14	1_L + 48	8	1_L + 7
(bd,BR,Xn)	14	1_L + 32	15	1_L + 49	9	1_L + 8
([bd,BR,Xn])	17	1_L + 35	18	1_L + 52	12	1_L + 11
([bd,BR,Xn],od)	18	1_L + 36	19	1_L + 53	12	1_L + 11
([bd,BR],Xn)	18	3_L + 34	19	3_L + 51	13	3_L + 10
([bd,BR],Xn,od)	19	3_L + 35	20	3_L + 52	14	3_L + 11

*This instruction interlocks the <ea> calculate and execute stages. Execution time for a DIV/0 exception taken and exception processing is approximately 16 + <ea> calculate clocks. For example, DIV.W #0,Dn takes approximately 24 clocks in both the <ea> calculate and execute times to execute the divide instruction, perform exception stacking, fetch the exception vector, and prefetch the next instruction.

10.6 INTEGER UNIT INSTRUCTION TIMINGS (Continued)

Addressing Mode	NEG, NEGX, NOT		PEA		ROL, ROR	
	<ea> Calculate	Execute	<ea> Calculate	Execute	<ea> Calculate	Execute
Dn	1	1	—	—	1	3/4*
An	—	—	—	—	—	—
(An)	1	1	2	1_L + 1	1	3
(An)+	1	1	—	—	1	3
–(An)	1	1	—	—	1	3
(d_{16},An)	1	1	2	1_L + 1	1	3
(d_{16},PC)	—	—	4	3_L + 1	—	—
(xxx).W, (xxx).L	1	1	2	1_L + 1	1	3
#<xxx>	—	—	—	—	—	—
(d_8,An,Xn)	3	3	4	1_L + 3	3	5
(d_8,PC,Xn)	—	—	6	2_L + 4	—	—
(BR,Xn)	6	1_L + 5	7	2_L + 5	6	1_L + 7
(bd,BR,Xn)	7	1_L + 6	8	2_L + 6	7	1_L + 8
([bd,BR,Xn])	9	1_L + 8	10	2_L + 8	9	1_L + 10
([bd,BR,Xn],od)	10	1_L + 9	11	2_L + 9	10	1_L + 11
([bd,BR],Xn)	10	3_L + 7	11	4_L + 7	10	3_L + 9
([bd,BR],Xn,od)	11	3_L + 8	12	4_L + 8	11	3_L + 10

*Immediate count specified for shift count/shift count specified in register, respectively.



NOTE: Transfer Attribute Signals = UPAx, SIZx, TTx, TMx, TLNx, R/W, CIOUT

Figure 11-4. Bus Arbitration Timing

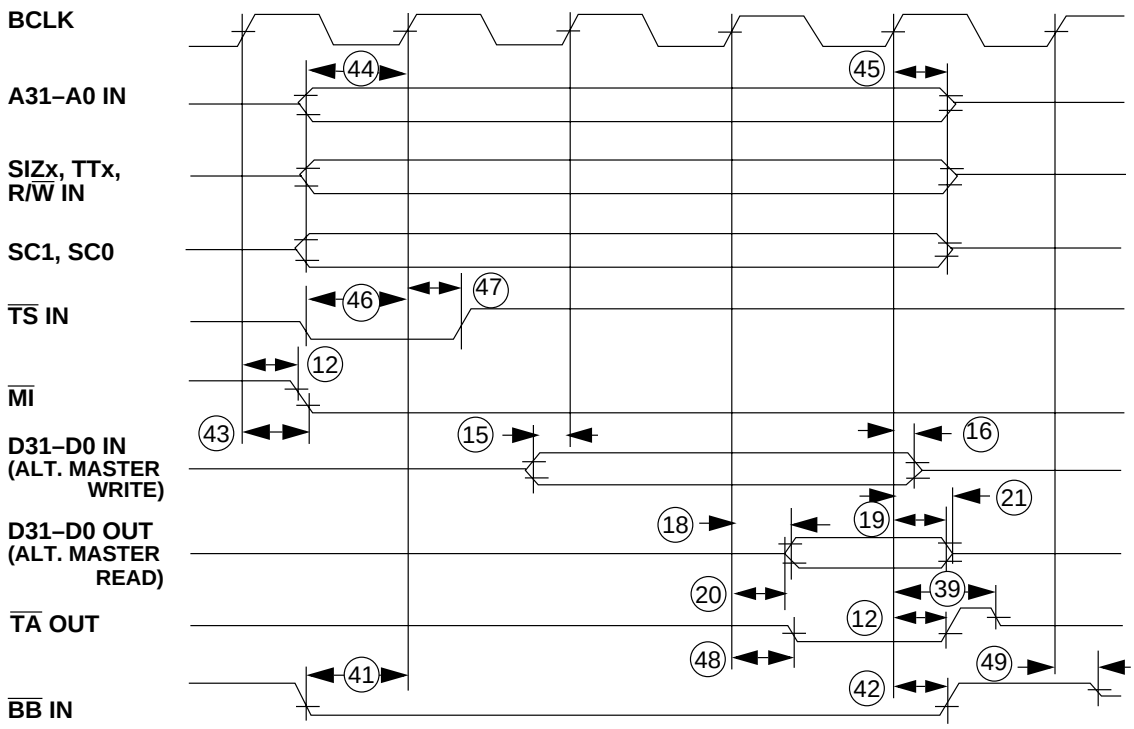
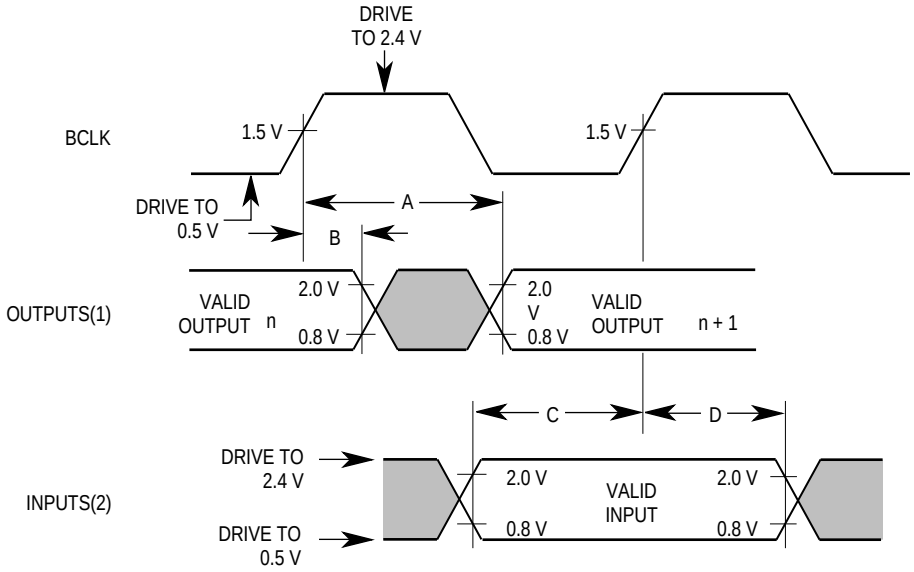


Figure B-10. Snoop Hit Timing



NOTES:

1. This output timing is applicable to all parameters specified relative to the rising edge of the clock.
2. This input timing is applicable to all parameters specified relative to the rising edge of the clock.

LEGEND:

A. Maximum output delay specification.
B. Minimum output hold time.
C. Minimum input setup time specification.
D. Minimum input hold time specification.

Figure C-11. Drive Levels and Test Points for AC Specifications