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	68040
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	33MHz
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	184-BCQFP
Supplier Device Package	184-CQFP (31.3x31.3)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68ec040fe33a

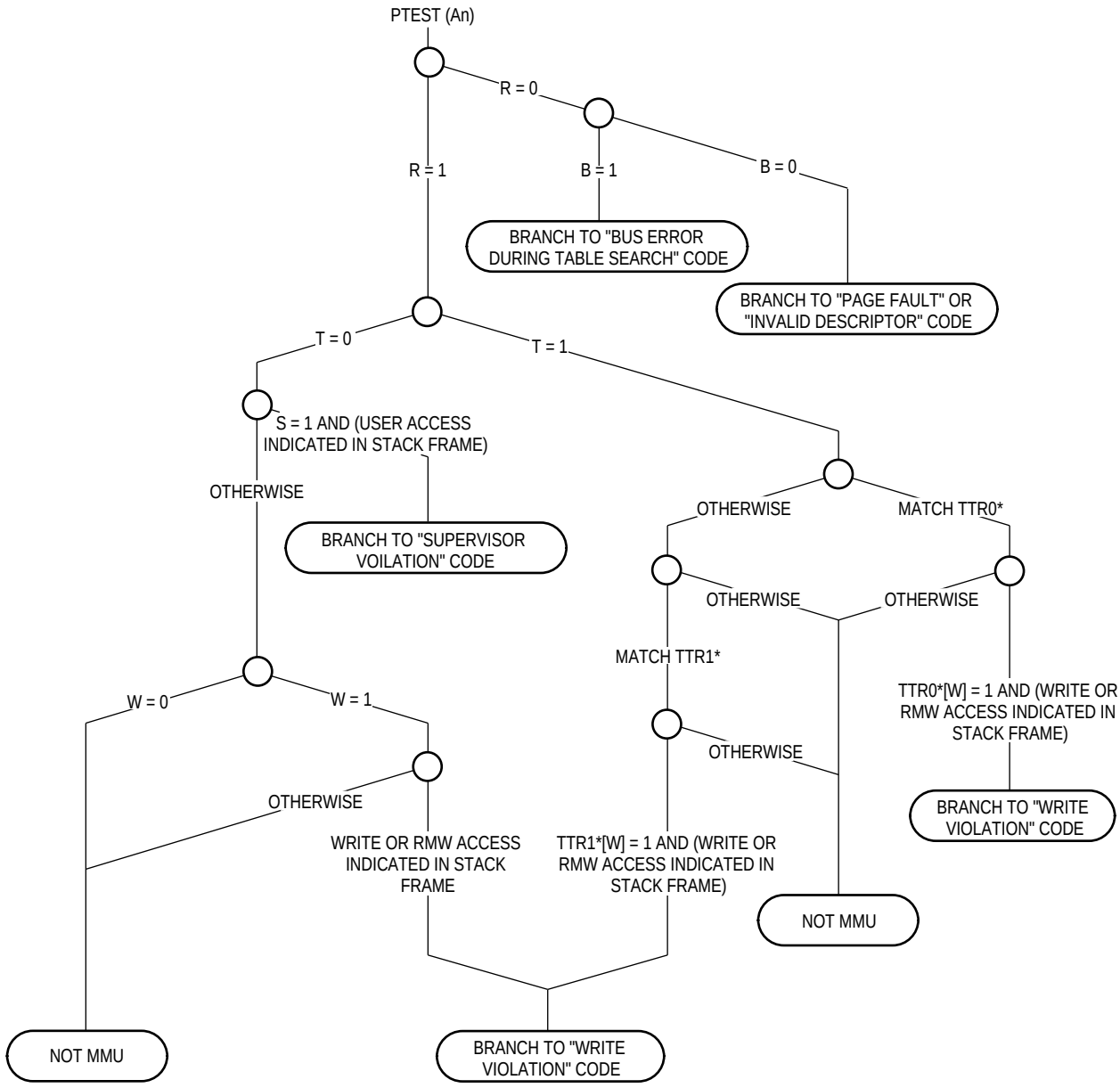
TABLE OF CONTENTS (Continued)

Table 1-3. Notational Conventions (Concluded)

Register Codes	
*	General Case.
C	Carry Bit in CCR
cc	Condition Codes from CCR
FC	Function Code
N	Negative Bit in CCR
U	Undefined, Reserved for Motorola Use.
V	Overflow Bit in CCR
X	Extend Bit in CCR
Z	Zero Bit in CCR
—	Not Affected or Applicable.
Stack Pointers	
ISP	Supervisor/Interrupt Stack Pointer
MSP	Supervisor/Master Stack Pointer
SP	Active Stack Pointer
SSP	Supervisor (Master or Interrupt) Stack Pointer
USP	User Stack Pointer
Miscellaneous	
<ea>	Effective Address
<label>	Assemble Program Label
<list>	List of registers, for example D3–D0.
LB	Lower Bound
m	Bit m of an Operand
m–n	Bits m through n of Operand
UB	Upper Bound

1.10 INSTRUCTION SET OVERVIEW

The instruction set is tailored to support high-level languages and is optimized for those instructions most commonly executed. The floating-point instructions for the M68040 are a commonly used subset of the MC68881/MC68882 instruction set with new arithmetic instructions to explicitly select single- or double-precision rounding. The remaining unimplemented instructions are less frequently used and are efficiently emulated in the M68040FPSP, maintaining compatibility with the MC68881/MC68882 floating-point coprocessors. The M68040 instruction set includes MOVE16, a new user instruction that allows high-speed transfers of 16-byte blocks between external devices such as memory to memory or coprocessor to memory. Table 1-4 provides an alphabetized listing of the M68040 instruction set's opcode, operation, and syntax. Refer to Table 1-3 for notations used in Table 1-4. The left operand in the syntax is always the source operand, and the right operand is the destination operand. Refer to M68000PM/AD, *M68000 Family Programmer's Reference Manual,* for details on instructions used by the M68040.



* Refers to either instruction or data transparent translation register.

Figure 3-23. MMU Status Interpretation

4.2 CACHE MANAGEMENT

Using the MOVEC instruction, the caches are individually enabled to access the 32-bit cache control register (CACR) illustrated in Figure 4-4. The CACR contains two enable bits that allow the instruction and data caches to be independently enabled or disabled. Setting one of these bits enables the associated cache without affecting the state of any lines within the cache. A hardware reset clears the CACR, disabling both caches; however, reset does not affect the tags, state information, and data within the caches. The CINV instruction must clear the caches before enabling them. It is not recommended that page descriptors be cached. Specifically, the M68040 does not support the caching of page descriptors in copyback mode with the bit pattern U = 0, M = 1, and R = 1 in a page descriptor. The M68040 table search algorithm will never leave this bit pattern for a page descriptor.

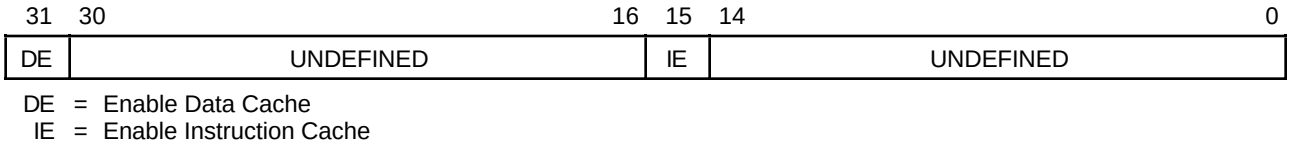


Figure 4-4. Cache Control Register

System hardware can assert the cache disable (CDIS) signal to dynamically disable both caches, regardless of the state of the enable bits in the CACR. The caches are disabled immediately after the current access completes. If CDIS is asserted during the access for the first half of a misaligned operand spanning two cache lines, the data cache is disabled for the second half of the operand. Accesses by the execution units bypass the caches while they are disabled and do not affect their contents (with the exception of CINV and CPUSH instructions). Disabling the caches with CDIS does not affect snoop operations. CDIS is intended primarily for use by in-circuit emulators to allow swapping between the tags and emulator memories.

Even if the instruction cache is disabled, the M68040 can cache instructions because of an internal cache line register. This happens for instruction loops that are completely resident within the first six bytes of a half-line. Thus, the cache line holding register can operate as a small cache. If a loop fits anywhere within the first three words of a half-line, then it becomes cached.

The CINV and CPUSH instructions support cache management in the supervisor mode. CINV allows selective invalidation of cache entries. CPUSH performs two operations: 1) any selected data cache lines containing dirty data are pushed to memory; 2) all selected cache lines are invalidated. This operation can be used to update a page in memory before swapping it out with snooping disabled or to push dirty data when changing a page caching mode to write-through. Because of the size of the caches, pushing pages or an entire cache incurs a significant time penalty. However, these instructions are interruptable to avoid large interrupt latencies. The state of the CDIS signal or the cache enable bits in the CACR does not affect the operation of CINV and CPUSH. Both instructions allow operation on a single cache line, all cache lines in a specific page, or an

5.12 TEST SIGNALS

The M68040 includes dedicated user-accessible test logic that is fully compatible with the IEEE 1149.1 *Standard Test Access Port and Boundary Scan Architecture*. Problems associated with testing high-density circuit boards have led to the development of this standard under the IEEE Test Technology Committee and Joint Test Action Group (JTAG) sponsorship. The M68040 implementation supports circuit board test strategies based on this standard. However, the JTAG interface is not intended to provide an in-circuit test to verify M68040 operations; therefore, it is impossible to test M68040 operations using this interface. **Section 6 IEEE 1149.1 Test Access Port (JTAG)** describes the M68040 implementation of the IEEE 1149.1 and is intended to be used with the supporting IEEE document.

5.12.1 Test Clock (TCK)

This input signal is used as a dedicated clock for the test logic. Since clocking of the test logic is independent of the normal operation of the MC68040, several other components on a board can share a common test clock with the processor even though each component may operate from a different system clock. The design of the test logic allows the test clock to run at low frequencies, or to be gated off entirely as required for test purposes.

5.12.2 Test Mode Select (TMS)

This input signal is decoded by the TAP controller and distinguishes the principle operationas of the test support circuitry.

5.12.3 Test Data In (TDI)

This input signal provides a serial data input to the TAP.

5.12.4 Test Data Out (TDO)

This three-state output signal provides a serial data output from the TAP. The TDO output can be placed in a high-impedance mode to allow parallel connection of board-level test data paths.

5.12.5 Test Reset ($\overline{TRST}$)—Not on MC68040V and MC68EC040V

This input signal provides an asynchronous reset of the TAP controller.

5.13 POWER SUPPLY CONNECTIONS

The M68040 requires connection to a V_{CC} power supply, positive with respect to ground. The V_{CC} and ground connections are grouped to supply adequate current to the various sections of the processor. **Section 12 Ordering Information and Mechanical Data** describes the groupings of V_{CC} and ground connections.

SECTION 7
BUS OPERATION

The M68040 bus interface supports synchronous data transfers between the processor and other devices in the system. This section provides a functional description of the bus, the signals that control the bus, and the bus cycles provided for data transfer operations. Operation of the bus is defined for transfers initiated by the processor as a bus master and for transfers initiated by an alternate bus master, which the processor snoops as a slave device. Descriptions of the error and halt conditions, bus arbitration, and the reset operation are also included. For timing specifications, refer to **Section 11 MC68040 Electrical and Thermal Characteristics.**

NOTE

For the MC68040V, MC68LC040, and MC68EC040 ignore all references to floating-point. For the MC68EC040 and MC68EC040V ignore all references to the memory management unit (MMU). Special modes of operation do not apply to these devices. Refer to **Appendix A MC68LC040** and **Appendix B MC68EC040** for details.

7.1 BUS CHARACTERISTICS

The M68040 uses the address bus (A31–A0) to specify the address for a data transfer and the data bus (D31–D0) to transfer the data. Control signals indicate the beginning and type of a bus cycle as well as the address space and size of the transfer. The selected device then controls the length of the cycle by terminating it using the control signals.

The M68040 uses two clocks to generate timing: a processor clock (PCLK) and a bus clock (BCLK). The PCLK signal is twice the frequency of the BCLK signal and is internally phase-locked to BCLK. PCLK is also distributed throughout the device to generate additional timing for additional edges for internal logic blocks and has no bearing on bus timing. The use of dual clock inputs allows the bus interface to operate at half the speed of the internal logic of the processor, requiring less stringent memory interface requirements. Since the rising edge of BCLK is used as the reference point for the phase-locked loop (PLL), all timing specifications are referenced to this edge.

Figure 7-1 illustrates the general relationship between the two clock signals and most input and output signals. The rising edge of the internally phase-locked PCLK is aligned with the rising edge of BCLK, and the two PCLK cycles corresponding to each BCLK cycle are divided into four states, T1–T4. Most outputs change during state T4, whether transitioning between a driven and high-impedance state or switching between assert and

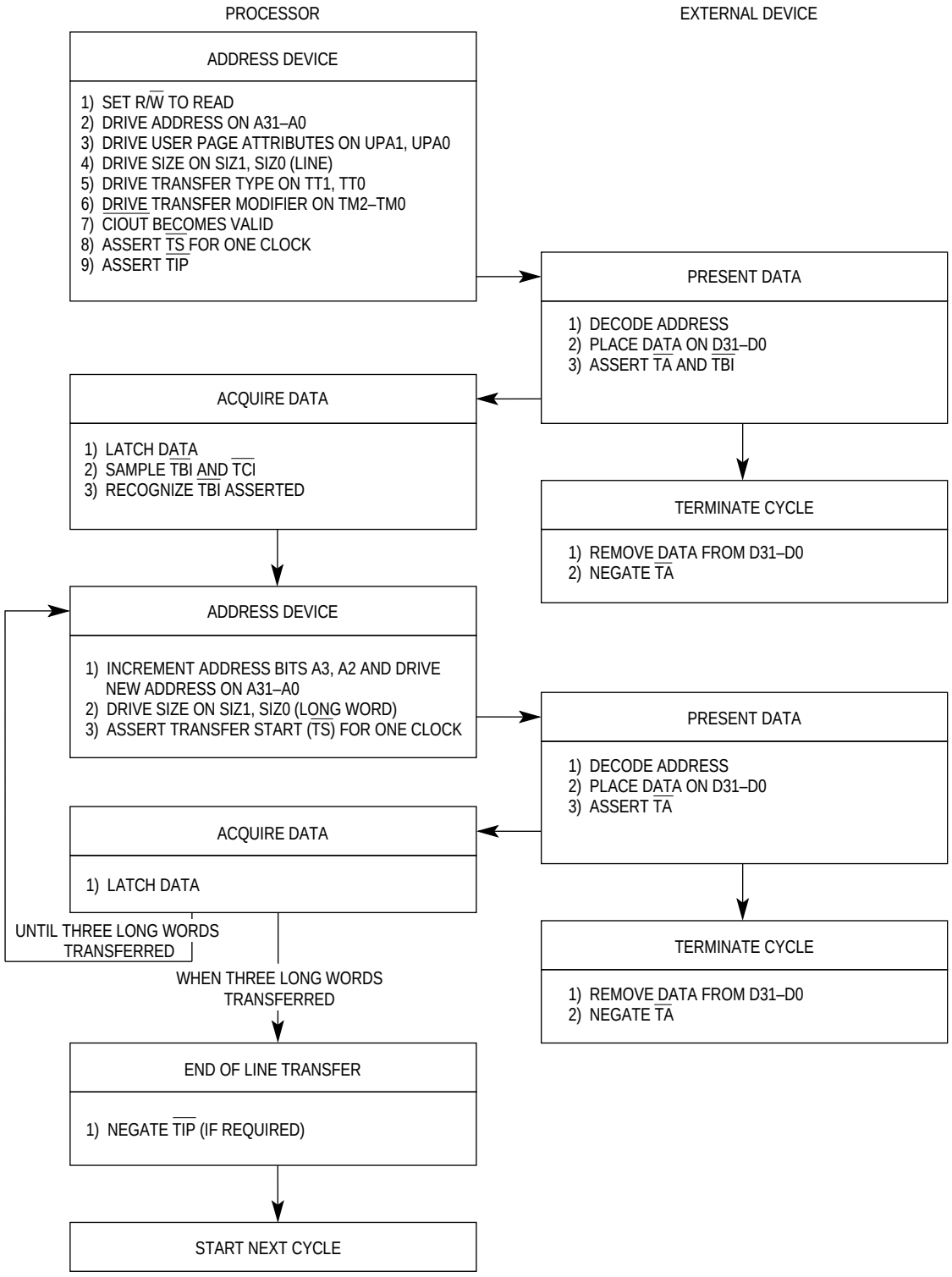


Figure 7-12. Burst-Inhibited Line Read Transfer Flowchart

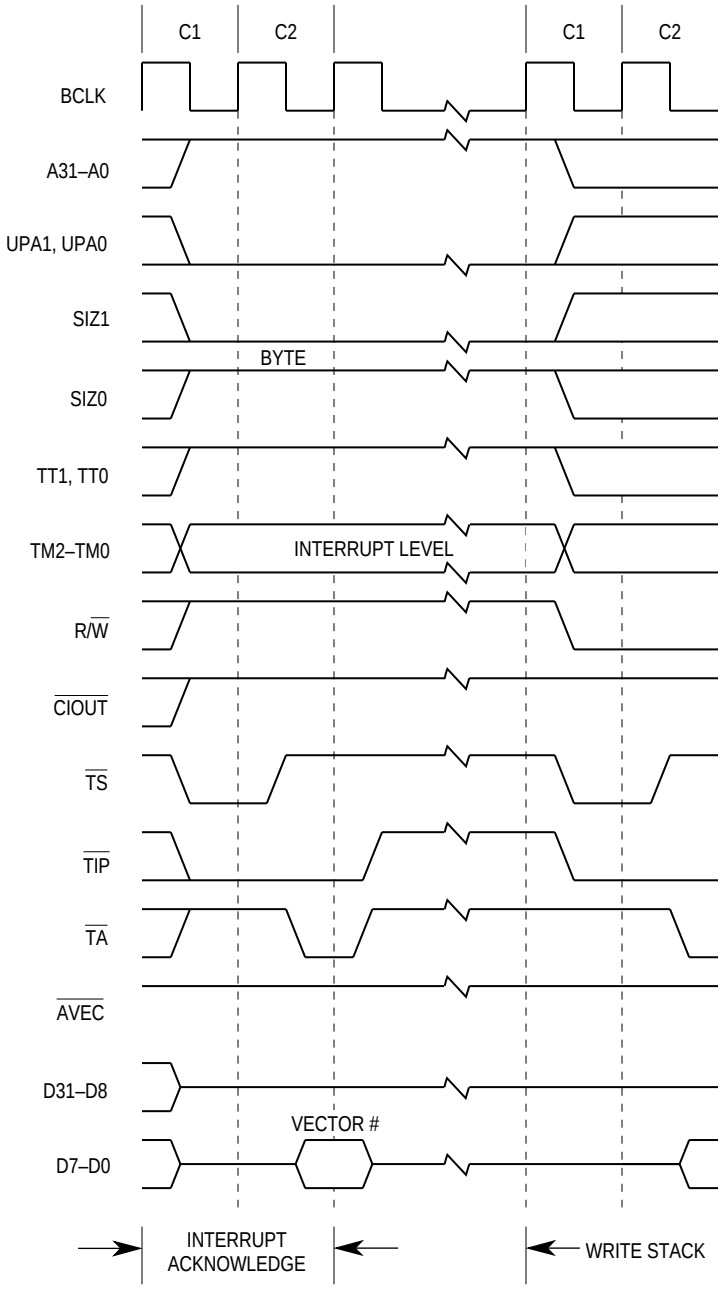


Figure 7-22. Interrupt Acknowledge Bus Cycle Timing

7.5.1.2 AUTOVECTOR INTERRUPT ACKNOWLEDGE BUS CYCLE. When the interrupting device cannot supply a vector number, it requests an automatically generated vector (autovector). Instead of placing a vector number on the data bus and asserting $\overline{TA}$, the device asserts the autovector ($\overline{AVEC}$) signal with $\overline{TA}$ to terminate the cycle. $\overline{AVEC}$ is only sampled with $\overline{TA}$ asserted. $\overline{AVEC}$ can be grounded if all interrupt requests are autovectored.

The vector number supplied in an autovector operation is derived from the interrupt priority level of the current interrupt. When the $\overline{AVEC}$ signal is asserted with $\overline{TA}$ during an interrupt acknowledge bus cycle, the M68040 ignores the state of the data bus and internally

To properly control termination of a bus cycle for a bus error or retry condition, $\overline{TA}$ and $\overline{TEA}$ must be asserted and negated for the same rising edge of BCLK. Table 7-5 lists the control signal combinations and the resulting bus cycle terminations. Bus error and retry terminations during burst cycles operate as described in **7.4.2 Line Read Transfers** and **7.4.4 Line Write Transfers**.

Table 7-5. $\overline{TA}$ and $\overline{TEA}$ Assertion Results

Case No.	$\overline{TA}$	$\overline{TEA}$	Result
1	High	Low	Bus Error—Terminate and Take Bus Error Exception, Possibly Deferred
2	Low	Low	Retry Operation—Terminate and Retry
3	Low	High	Normal Cycle Terminate and Continue
4	High	High	Insert Wait States

7.6.1 Bus Errors

The system hardware can use the $\overline{TEA}$ signal to abort the current bus cycle when a fault is detected. A bus error is recognized during a bus cycle when $\overline{TA}$ is negated and $\overline{TEA}$ is asserted. When the processor recognizes a bus error condition for an access, the access is terminated immediately. A line access that has $\overline{TEA}$ asserted for one of the four long-word transfers aborts without completing the remaining transfers, regardless of whether the line transfer uses a burst or burst-inhibited access.

When $\overline{TEA}$ is asserted to terminate a bus cycle, the M68040 can enter access error exception processing immediately following the bus cycle, or it can defer processing the exception. The instruction prefetch mechanism requests instruction words from the instruction memory unit before it is ready to execute them. If a bus error occurs on an instruction fetch, the processor does not take the exception until it attempts to use the instruction. Should an intervening instruction cause a branch or should a task switch occur, the access error exception for the unused access does not occur. Similarly, if a bus error is detected on the second, third, or fourth long-word transfer for a line read access, an access error exception is taken only if the execution unit is specifically requesting that long word. Otherwise, the line is not placed in the cache, and the processor repeats the line access when another access references the line. If a misaligned operand spans two long words in a line, a bus error on either the first or second transfer for the line causes exception processing to begin immediately. A bus error termination for any write accesses or for read accesses that reference data specifically requested by the execution unit causes the processor to begin exception processing immediately. Refer to **Section 8 Exception Processing** for details of access error exception processing.

When a bus error terminates an access, the contents of the corresponding cache can be affected in different ways, depending on the type of access. For a cache line read to replace a valid instruction or data cache line, the cache line being filled is invalidated before the bus cycle begins and remains invalid if the replacement line access is terminated with a bus error. If a dirty data cache line is being replaced and a bus error occurs during the replacement line read, the dirty line is restored from an internal push

7.6.2 Retry Operation

When an external device asserts both the $\overline{TA}$ and $\overline{TEA}$ signals during a bus cycle, the processor enters the retry sequence. The processor terminates the bus cycle and immediately retries the cycle using the same access information (address and transfer attributes). However, if the bus cycle was a cache push operation, the bus is arbitrated away from the M68040 before the retry operation, and a snoop during the arbitration invalidates the cache push, then the processor does not use the same access information. Figure 7-28 illustrates a functional timing diagram for a retry of a read bus transfer.

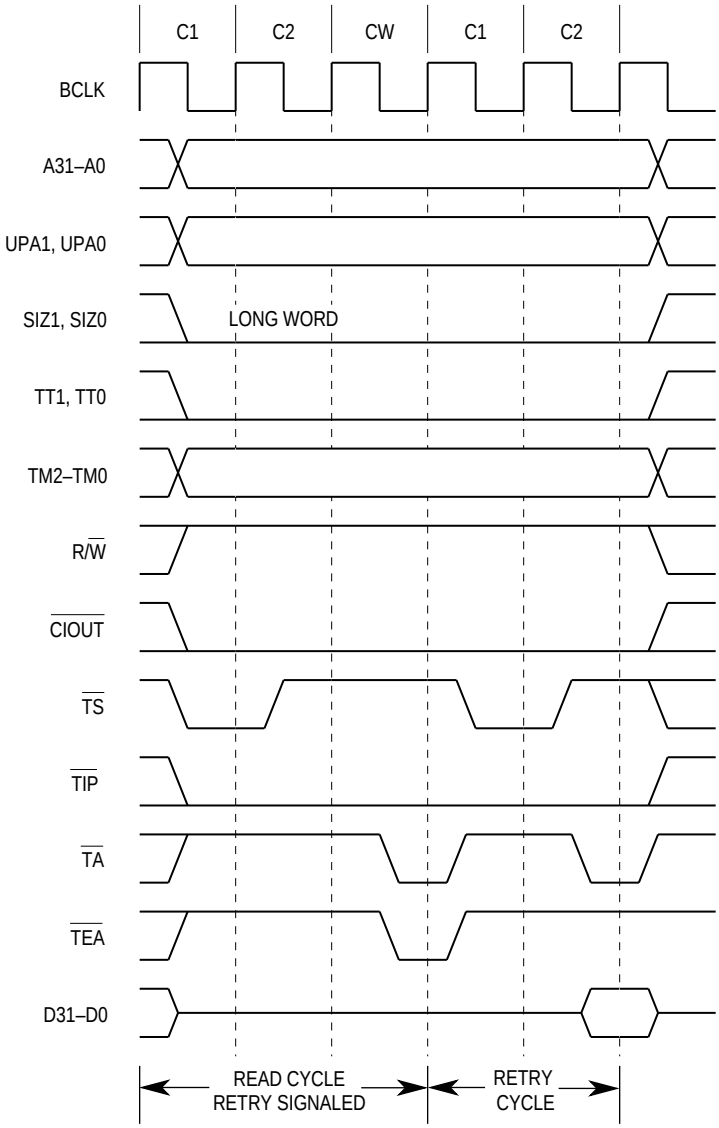


Figure 7-28. Retry Read Transfer Timing

The processor retries any read or write cycles of a read-modify-write transfer separately; $\overline{LOCK}$ remains asserted during the entire retry sequence. If the last bus cycle of a locked access is retried, $\overline{LOCKE}$ remains asserted through the retry of the write cycle.

transfer. Edge-triggered latch B is clocked by the rising edge of BCLK and latches the data from latch A for use by internal logic.

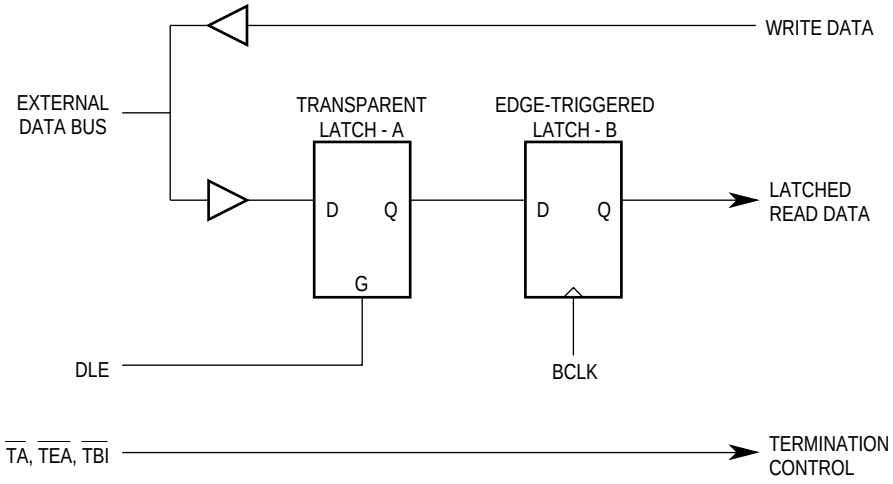


Figure 7-47. DLE Mode Block Diagram

Figure 7-48 illustrates the data read timing for both normal operation and DLE mode. During normal operation (i.e., DLE mode disabled), latch A is always transparent, and by the rising edge of BCLK, read data is latched. Data must meet setup and hold time specifications #15 and #16 in this case. When the DLE mode is enabled, the data can be latched by the rising edge of BCLK or the falling edge of DLE, depending on the timing for DLE.

level 6 interrupt, the SR mask is automatically updated with a value of 6 before entering the handler routine so that subsequent level 6 interrupts and lower level interrupts are masked. Provided no instruction that lowers the mask value is executed, the external request can be lowered to level 3 and then raised back to level 6 and a second level 6 interrupt is not processed. However, if the M68040 is handling a level 7 interrupt (SR mask set to level 7) and the external request is lowered to level 3 and than raised back to level 7, a second level 7 interrupt is processed. The second level 7 interrupt is processed because the level 7 interrupt is transition sensitive. A level comparison also generates a level 7 interrupt if the request level and mask level are at 7 and the priority mask is then set to a lower level (with the MOVE to SR or RTE instruction, for example). The level 6 interrupt request and mask level example in Figure 8-3 is the same as for all interrupt levels except 7.

Table 8-6 lists the possible combinations of write-backs and the proper way to handle them. The SSW_RW column indicates a read or write cycle; the SSW_PUSH column indicates whether the fault is for a push (TT = 00 and TM = 000). The WB1S, WB2S, and WB3S columns list the respective field's V-bit and indicate a MOVE16 transfer type (TT = 01). The easy cleanup data written column lists the stack's field to be written out to memory if the user is not concerned with retouching peripherals. The hard cleanup action column lists the action to be taken if the peripherals cannot be retouched by MOVE16 (if different from easy cleanup). Note that if a push access error is reported and the size is long word, all four long words, PD0–PD3, are still valid for the line. The exception handler can either write PD0–PD3 using the fault address with bits 3–0 cleared or write the PD corresponding to bits 3–2 of the address (e.g., address $0000000C corresponds to PD3). Note that a MOVE16 is never reported in the WB3S. The SIZE field of WB3S is never a line.

After the bus error exception handler completes all pending operations and executes an RTE to return, the RTE reads only the stack information from offset $0–$D in the access error stack frame. For a pending trace exception, unimplemented floating-point instruction exception, or floating-point post-instruction exception, the RTE adjusts the stack to match the pending exception and immediately begins exception processing, without requiring the exception to reoccur.

exception and the F-line illegal instruction share the same vector, the exception handler uses the stack frame format ($0 or $2) to distinguish between the two.

Table 9-10. Unimplemented Instructions

Monadic Operations	
FACOS	FINTRZ
FASIN	FLOG10
FATAN	FLOGN
FATANH	FLOGNP1
FCOS	FMOVECR
FCOSH	FSIN
FETOX	FSINCOS
FETOXM1	FSINH
FGETEXP	FTAN
FGETMAN	FTANH
FINT	FTENTOX
FTWOTOX	—
Dyadic Operations	
FMOD	FREM
FSCALE	—

When an unimplemented floating-point instruction is encountered, the processor waits for all previous floating-point instructions to complete execution. Pending exceptions are taken and handled prior to the execution of the unimplemented instruction.

Next, the instruction is partially decoded to allow fetching of the memory source operand, if required. When the operand fetch begins, all other read accesses for previous instructions are complete, and only the execution and write-back of results for previous integer instructions remains to be completed. If an access error (bus error) occurs in fetching the operand or in completing any other access before beginning the operand fetch, the unimplemented instruction is restarted after the processor returns from exception handling for the error. Refer to **Section 8 Exception Processing** for more information on access errors.

The fetched source operand is passed to the FPU, which converts the operand to extended precision and saves the intermediate result. If the operand is an unsupported data type (denormalized, unnormalized, or packed decimal real), the unimplemented floating-point exception takes precedence, and the floating-point instruction emulation routine must detect the unsupported data type.

The processor begins exception processing for the unimplemented floating-point instruction by making an internal copy of the current SR. The processor then enters the supervisor mode and clears the trace bits (T1, T0). The processor creates a format $2 stack frame and saves the vector offset, PC, internal copy of the SR, and calculated

b. If the user OPERR exception handler is enabled and the destination floating-point data register is not modified, an OPERR exception is posted. The next floating-point instruction that is encountered takes a pre-instruction exception. The OPERR entry in the processor's vector table points to the M68040FPSP OPERR exception handler. Once the M68040FPSP OPERR exception handler recognizes the operand error as a maskable condition, it does not modify the destination or pass control to the user OPERR exception handler.

9.7.3.2 NONMASKABLE EXCEPTION CONDITIONS. If an FMOVE to byte, word, or long word has a source operand that is too large to be represented in the specified destination integer format (integer overflow, NAN, infinity) or if the source operand is equal to the largest negative integer representable in the specified destination integer format (erroneous MC68040 condition), the processor immediately takes a post-instruction exception. Instruction execution continues at the M68040FPSP OPERR exception handler.

If the M68040FPSP determines a nonmaskable erroneous MC68040 condition caused the exception, it stores the largest negative integer representable in the given destination integer format (-2^7 for byte, -2^{15} for word, and -2^{31} for long word). The M68040FPSP OPERR exception handler then returns the processor to normal processing. If an integer overflow or an FMOVE to byte, word, or long word with a source of infinity causes the exception, then the destination is written with the largest positive or negative integer that can be represented in the given format. If an FMOVE to byte of word or long word with a source of NAN causes the exception, then the most significant 8, 16, or 32 bits, respectively, are written to the destination. Next, the M68040FPSP OPERR exception handler checks to see if the user OPERR exception handler is enabled.

a. If the user OPERR exception handler is disabled, an exception-causing INEX1 or INEX2 condition exists, and the user INEX exception handler is enabled. The M68040FPSP OPERR exception handler restores the FPU to its exceptional state, cleans up the stack to the conditions prior to execution, and continues instruction execution at the user INEX exception handler. No parameters are passed to the user INEX exception handler since the M68040FPSP OPERR exception handler provides the illusion that it never existed. Otherwise, the M68040FPSP OPERR exception handler returns the processor to normal processing.

b. If the user OPERR exception handler is enabled and the destination is a floating-point data register, then the M68040FPSP exception handler does not modify the register. The M68040FPSP OPERR exception handler restores the FPU to its exceptional state, cleans up the stack to the conditions prior to execution, and continues instruction execution at the user OPERR exception handler. No parameters are passed to the user OPERR exception handler since the M68040FPSP OPERR exception handler provides the illusion that it never existed.

The user OPERR exception handler must execute an FSAVE as its first floating-point instruction. Table 9-16 lists the floating-point state frame fields for OPERR exceptions resulting from the execution of opclass 010 or 000 (register-to-register or memory-to-register) instructions and opclass 011 (register-to-memory) instructions defined for the use by the supervisor exception handler.

a. If the user UNFL exception handler is disabled, the M68040FPSP UNFL exception handler checks for an INEX1 or INEX2 exception condition with the user INEX exception handler enabled. If not, the processor returns to normal instruction flow. Otherwise, the M68040FPSP UNFL exception handler restores the FPU to its exceptional state, cleans up the stack to the conditions prior to execution, and continues instruction execution at the user INEX exception handler. No parameters are passed to the user INEX exception handler since the M68040FPSP UNFL exception handler provides the illusion that it never existed. Otherwise, the M68040FPSP UNFL exception handler returns the processor to normal processing.

b. If the user UNFL exception handler is enabled, the M68040FPSP UNFL exception handler restores the FPU to its exceptional state, cleans up the stack to the conditions prior to execution, and continues instruction execution at the user UNFL exception handler. Once the M68040FPSP UNFL exception handler recognizes the operand error as a maskable condition, it does not modify the destination or pass control to the user UNFL exception handler.

The user UNFL exception handler must execute an FSAVE as its first floating-point instruction. At this point, the destination contains the rounding mode values listed in Table 9-13, and the user UNFL exception handler can choose to modify these values. The E3 and E1 bits of the floating-point state frame need to be examined to determine which fields on the floating-point state frame are valid. E3 always takes precedence and must always be serviced first. Table 9-16 lists the floating-point state frame fields for OVFL exceptions with E3 set or with E3 clear and E1 set. It is possible for an FADD, FSUB, FMUL, and FDIV to report a post-instruction exception, although these instructions normally generate a pre-instruction exception. The following example illustrates why a post-instruction exception is generated.

```
FADD      FP2,FP0       ; this instruction generates an underflow exception
FMOVE     FP0, <ea>     ; this instruction is executing when underflow occurs
```

In this example, assume that the FMOVE instruction starts once the FADD instruction generates an underflow. Given the register dependency on FP0, the destination of the FADD instruction, FP0 needs to be resolved prior to the FMOVE instruction execution. For this example, there is no choice but to have the FADD instruction report a post-instruction exception immediately. Note that for this case, even though the T-bit of the floating-point state frame is set (post-instruction exception), it does not imply an FMOVE OUT instruction. Therefore, the effective address field in the format $3 stack frame is invalid.

The FMOVE OUT instruction generates a post-instruction exception. For this case, the effective address field in the format $3 stack frame points to the destination memory location. If the destination is an integer data register, the FPIAR points to the F-line word of the offending instruction, and the F-line word contains the integer data register number. If the M68040FPSP unimplemented instruction exception handler is used, there can be some other cases in which an underflow is reported. If an INEX2 or INEX1 exceptional condition exists and the user INEX exception handler is enabled, it is the responsibility of the user UNFL exception handler to look for this situation.

The user UNFL exception handler examines the E3 bit of the floating-point state frame to exit from this exception handler. If the E3 bit is set, it must be cleared prior to restoring the floating-point frame through the FRESTORE instruction. If the E3 bit is clear and the E1 bit

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.

Table C-1. Additional MC68040V and MC68EC040V Signals

Signal Name	Mnemonic	Function
Low Frequency Operation	$\overline{\text{LFO}}$	Used to enter the low frequency mode of operation.
Loss of Clock	LOC	Indicates loss of BCLK input, a reset is required
System Clock Disable	$\overline{\text{SCD}}$	Indicates normal operation is suspended and low-power stop mode is active, system logic may remove or change the frequency of the BCLK input.

C.1.1 Low Frequency Operation ($\overline{\text{LFO}}$)

When asserted, this input signal allows the frequency of BCLK to be changed instantaneously (0 to 16 MHz) providing minimum pulse width constraints are met (see **C.7 MC68040V and MC68EC040V Electrical Characteristics**. $\overline{\text{LFO}}$ is only recognized during low-power stop mode and reset.

C.1.2 Loss of Clock (LOC)

Whenever the internal clock circuitry detects either a phase lock error or a loss of BCLK, this output signal is driven high (only during normal mode of clocking operation). LOC is also three-stated during reset, low-power stop, or low frequency operation. There should be a pull-down resistor on the system board to ground.

C.1.3 System Clock Disable ($\overline{\text{SCD}}$)

When asserted this output signal indicates, when asserted, that the BCLK input can be disabled or changed in frequency. $\overline{\text{SCD}}$ is asserted upon termination of the LPSTOP broadcast cycle. BCLK must be stable when $\overline{\text{SCD}}$ is negated, in accordance with the specifications in **C.7 MC68040V and MC68EC040V Electrical Characteristics**.

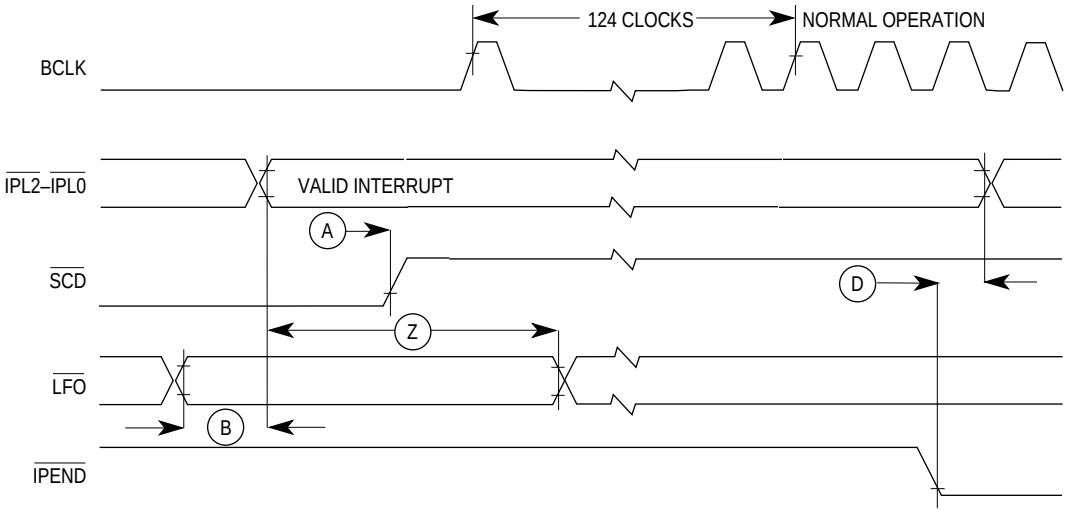


Figure C-20. Exiting LPSTOP with Interrupt

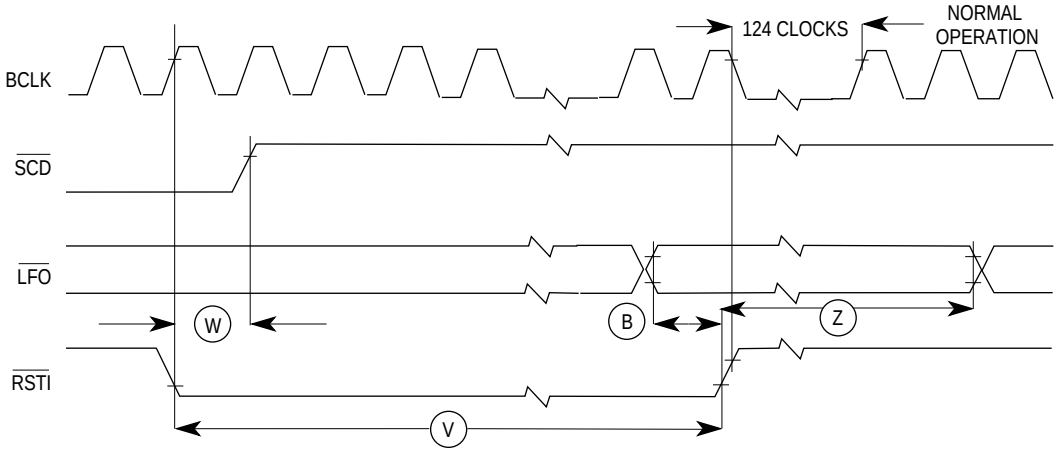


Figure C-21. Exiting of LPSTOP with RESET

INDEX

–A–

–B–