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	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=mc68040fe33v

SECTION 1
INTRODUCTION

The MC68040, MC68040V, MC68LC040, MC68EC040, and MC68EC040V (collectively called M68040) are Motorola's third generation of M68000-compatible, high-performance, 32-bit microprocessors. All five devices are virtual memory microprocessors employing multiple concurrent execution units and a highly integrated architecture that provides very high performance in a monolithic HCMOS device. They integrate an MC68030-compatible integer unit (IU) and two independent caches. The MC68040, MC68040V, and MC68LC040 contain dual, independent, demand-paged memory management units (MMUs) for instruction and data stream accesses and independent, 4-Kbyte instruction and data caches. The MC68040 contains an MC68881/MC68882-compatible floating-point unit (FPU). The use of multiple independent execution pipelines, multiple internal buses, and a full internal Harvard architecture, including separate physical caches for both instruction and data accesses, achieves a high degree of instruction execution parallelism on all three processors. The on-chip bus snoop logic, which directly supports cache coherency in multimaster applications, enhances cache functionality.

The M68040 family is user object-code compatible with previous M68000 family members and is specifically optimized to reduce the execution time of compiler-generated code. All five processors implement Motorola's latest HCMOS technology, providing an ideal balance between speed, power, and physical device size.

1.1 DIFFERENCES

Because the functionality of individual M68040 family members are similar, this manual is organized so that the reader will take the following differences into account while reading the rest of this manual. Unless otherwise noted, all references to M68040, with the exception of the differences outlined below, will apply to the MC68040, MC68040V, MC68LC040, MC68EC040, and MC68EC040V. The following paragraphs describe the differences of MC68040V, MC68LC040, MC68EC040, and the MC68EC040V from the MC68040.

1.1.1 MC68040V and MC68LC040

The MC68040V and MC68LC040 are derivatives of the MC68040. They implement the same IU and MMU as the MC68040, but have no FPU. The MC68LC040 is pin compatible with the MC68040. The MC68040V is not pin compatible with the MC68040 and contains some additional features. The following differences exist between the MC68040V, MC68LC040, and MC68040:

has control of the I/O pins. The 1149.1A interface is transparent to system operation except for drive control selection during execution of this instruction.

When the system logic has control of the signal I/O directions and levels, the drive control latches are loaded from the IPL2–IPL0 pins at the negation of the RSTI signal. After RSTI has been negated, and the 128-clock internal reset cycle has expired (see **Section 7 Bus Operation**), the DRVCTL.S instruction is executed. Each drive control latch is modified during the update-DR state. Any subsequent RSTI signal negation while in a system configuration (i.e., system logic has control of the signal I/O directions and levels) can cause the drive control latches to be overwritten with new IPL signal values. The system bus can be suspended in a wait state while this function is being performed.

6.2.8 BYPASS

The BYPASS instruction selects the single-bit bypass register, creating a single-bit shift-register path from TDI to the bypass register to TDO. The instruction enhances test efficiency when a component other than the M68040 becomes the device under test. When the bypass register is initially selected, the instruction shift register stage is set to a logic zero on the rising edge of TCK following entry into the capture-DR state. Therefore, the first bit to be shifted out after selecting the bypass register is always a logic zero. Figure 6-2 illustrates the bypass register.

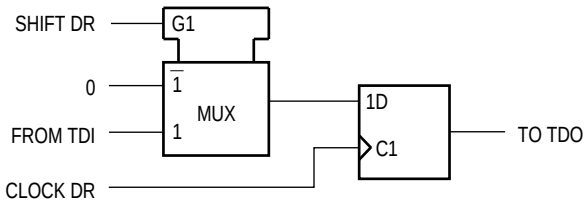


Figure 6-2. Bypass Register

6.3 BOUNDARY SCAN REGISTER

The 184-bit boundary scan register uses the TAP controller to scan user-defined values into the output buffers, capture values presented to input pins, and control the direction of bidirectional pins. The instruction shift register cell nearest TDO (i.e., first to be shifted out) is defined as bit zero. The last bit to be shifted out is bit 183. This register includes cells for all device signal pins and clock pins along with associated control signals.

The M68040 boundary scan register consists of three cell structure types, O.Latch, I.Pin, and IO.Ctl, that are associated with a boundary scan register bit. All boundary scan output cells capture the logic level of the device output latch during the capture-DR state. Figures 6-3 through 6-5 illustrate these three cell types. Figure 6-6 illustrates the general arrangement of these cells.

Table 6-2. Boundary Scan Bit Definitions[1]

Bit	Cell Type	Pin/Cell Name	Pin Type	Output Ctrl Cell
0	O.Latch	RST0	Output[2]	(Note 3)
1	O.Latch	IPEND	Output[2]	(Note 3)
2	O.Latch	CIOUT	TS-Output[2]	io.0
3	O.Latch	UPA0	TS-Output[2]	io.0
4	O.Latch	UPA1	TS-Output[2]	io.0
5	O.Latch	TT0	I/O[2]	io.0
6	I.Pin	TT0	I/O	io.0
7	O.Latch	TT1	I/O[2]	io.0
8	I.Pin	TT1	I/O	io.0
9	O.Latch	A10	I/O[2]	io.ab
10	I.Pin	A10	I/O	io.ab
11	O.Latch	A11	I/O[2]	io.ab
12	I.Pin	A11	I/O	io.ab
13	O.Latch	A12	I/O[2]	io.ab
14	I.Pin	A12	I/O	io.ab
15	O.Latch	A13	I/O[2]	io.ab
16	I.Pin	A13	I/O	io.ab
17	O.Latch	A14	I/O[2]	io.ab
18	I.Pin	A14	I/O	io.ab
19	O.Latch	A15	I/O[2]	io.ab
20	I.Pin	A15	I/O	io.ab
21	O.Latch	A16	I/O[2]	io.ab
22	I.Pin	A16	I/O	io.ab
23	O.Latch	A17	I/O[2]	io.ab
24	I.Pin	A17	I/O	io.ab
25	O.Latch	A18	I/O[2]	io.ab
26	I.Pin	A18	I/O	io.ab
27	O.Latch	A19	I/O[2]	io.ab
28	I.Pin	A19	I/O	io.ab
29	O.Latch	A20	I/O[2]	io.ab
30	I.Pin	A20	I/O	io.ab
31	O.Latch	A21	I/O[2]	io.ab
32	I.Pin	A21	I/O	io.ab
33	O.Latch	A22	I/O[2]	io.ab
34	I.Pin	A22	I/O	io.ab
35	O.Latch	A23	I/O[2]	io.ab
36	I.Pin	A23	I/O	io.ab
37	O.Latch	A24	I/O[2]	io.ab
38	I.Pin	A24	I/O	io.ab
39	O.Latch	A25	I/O[2]	io.ab
40	I.Pin	A25	I/O	io.ab
41	O.Latch	A26	I/O[2]	io.ab
42	I.Pin	A26	I/O	io.ab
43	O.Latch	A27	I/O[2]	io.ab
44	I.Pin	A27	I/O	io.ab
45	O.Latch	A28	I/O[2]	io.ab
46	I.Pin	A28	I/O	io.ab
47	O.Latch	A29	I/O[2]	io.ab
48	I.Pin	A29	I/O	io.ab
49	O.Latch	A30	I/O[2]	io.ab
50	I.Pin	A30	I/O	io.ab
51	O.Latch	A31	I/O[2]	io.ab
52	I.Pin	A31	I/O	io.ab
53	O.Latch	D0	I/O[2]	io.db
54	O.Latch	D1	I/O[2]	io.db
55	O.Latch	D2	I/O[2]	io.db
56	O.Latch	D3	I/O[2]	io.db
57	O.Latch	D4	I/O[2]	io.db
58	O.Latch	D5	I/O[2]	io.db
59	O.Latch	D6	I/O[2]	io.db
60	O.Latch	D7	I/O[2]	io.db
61	O.Latch	D8	I/O[2]	io.db
62	O.Latch	D9	I/O[2]	io.db
63	O.Latch	D10	I/O[2]	io.db
64	O.Latch	D11	I/O[2]	io.db
65	O.Latch	D12	I/O[2]	io.db
66	O.Latch	D13	I/O[2]	io.db
67	O.Latch	D14	I/O[2]	io.db
68	O.Latch	D15	I/O[2]	io.db
69	O.Latch	D16	I/O[2]	io.db
70	O.Latch	D17	I/O[2]	io.db
71	O.Latch	D18	I/O[2]	io.db
72	O.Latch	D19	I/O[2]	io.db
73	O.Latch	D20	I/O[2]	io.db

restarted, and a proper reentry into any of the four instructions is again required before the system clocks can be stopped.

Control over the output enable signals using the boundary scan register and the EXTEST and HIGHZ instructions requires a compatible circuit-board test environment to avoid destructive configurations. The user is responsible for avoiding situations in which the M68040 output drivers are enabled into actively driven networks.

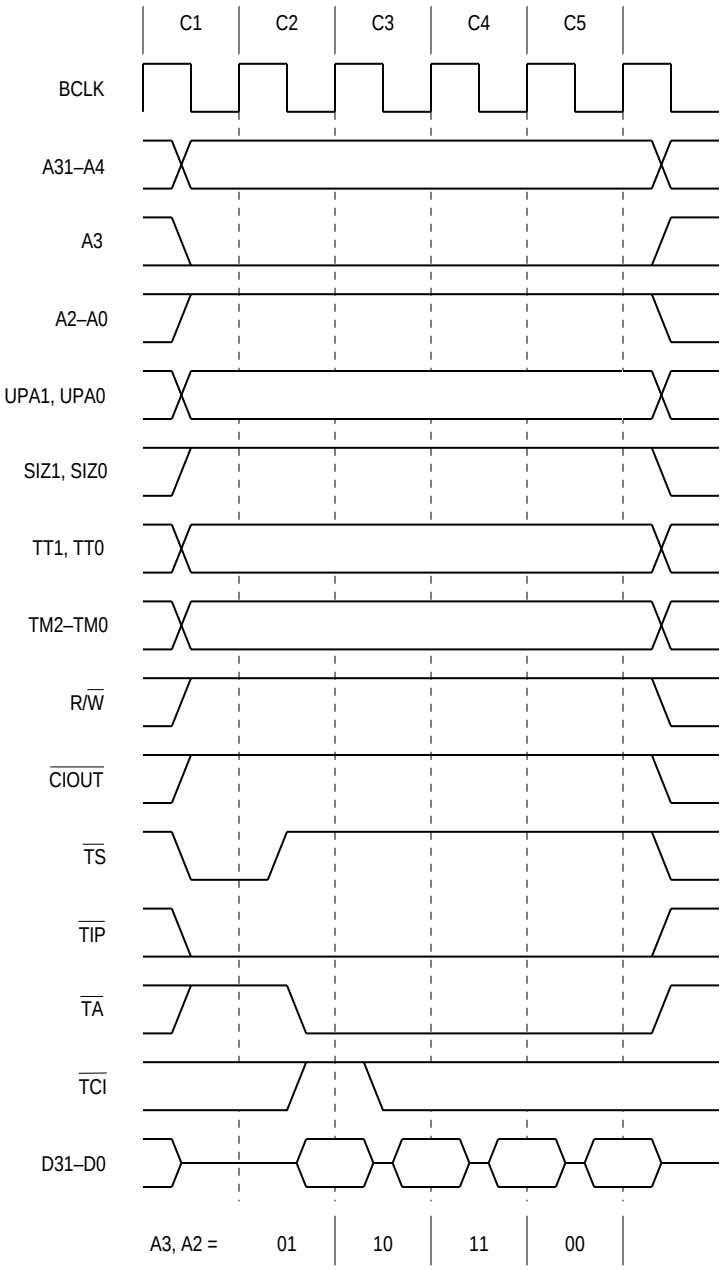
The TRST signal provides the ability for an asynchronous reset of the test logic and requires no internal clocking to force the TAP controller into the test-logic-reset state. This signal should be asserted during system power-up to initialize the 1149.1A test interface and avoid the potential for board-level bus conflicts. Essentially the TRST signal provides the ability to prevent possible board-level bus contention during power-up due to the test logic having control of the pins. The device has no internal power-up reset circuit. The TRST signal should be treated similar to the RSTI signal for board design considerations concerning power-up conditions.

Negation of the TRST signal requires certain precautions to achieve a predictable TAP controller state. The TMS signal is sampled on the rising edge of TCK and sequences the TAP controller. If TMS is low and TRST is negated simultaneously with the rising edge of TCK, the resultant TAP controller state is unpredictable but will be either test-logic-reset or run-test/idle. To avoid this uncertainty, either 1) the negation of TRST can be synchronized with the falling edge of TCK or 2) TMS can remain high until after TRST negation. Alternatively, holding TMS low for two or more TCK periods following TRST negation ensures that the TAP controller is in the run-test/idle state.

6.5 DISABLING THE IEEE STANDARD 1149.1A OPERATION

There are two considerations for non-IEEE standard 1149.1A operation. First, TCK does not include an internal pullup resistor and should not be left unconnected to preclude mid-level inputs. The second consideration is to ensure that the IEEE standard 1149.1A test logic remains transparent to the system logic by providing the ability to force the test-logic-reset state.

Figure 6-7 illustrates disabling the IEEE standard 1149.1A operation through connecting TRST directly or through a resistor to ground or a suitable logic network. Connecting TRST to RSTI while TCK is held either high or low meets the two considerations. If a pulse asserts TRST , the TAP controller is forced into the test-logic-reset state and can remain in this state as long as a rising edge on the TCK signal does not occur when TMS is low.

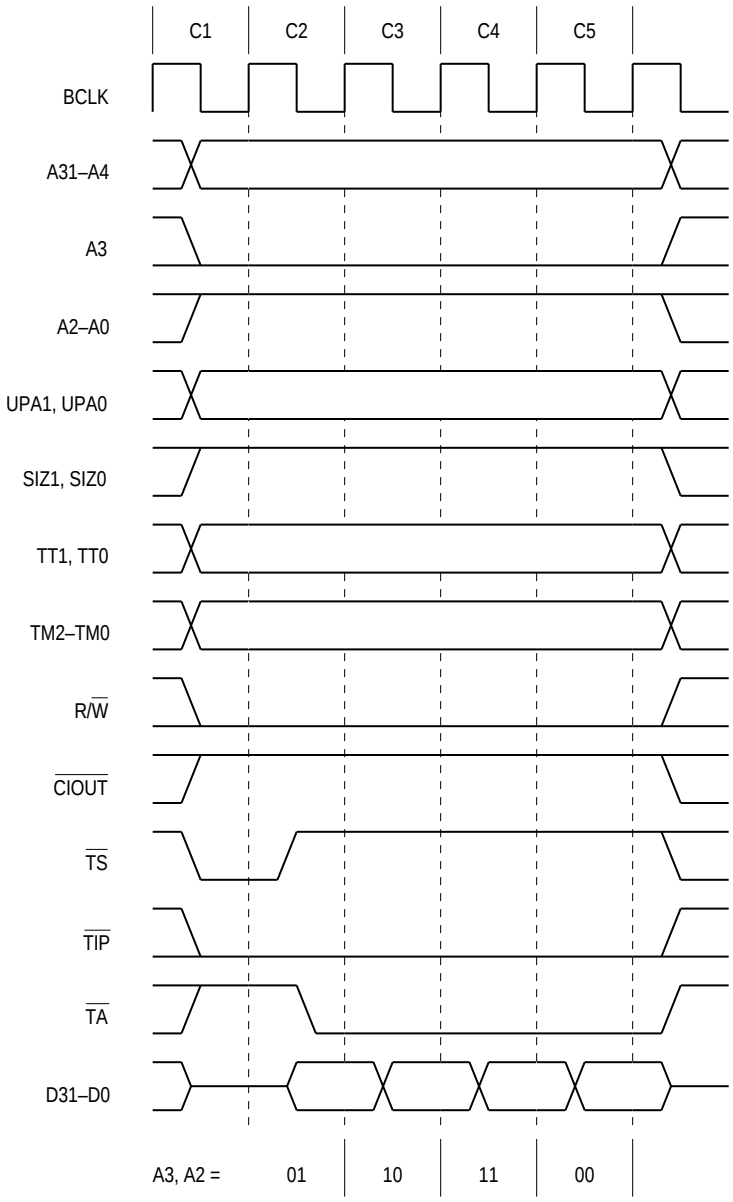


NOTE: The selected device increments the value of A3 and A2.

Figure 7-11. Line Read Transfer Timing

Clock 1 (C1)

The line read 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 that are translated by the corresponding memory unit, the UPAx signals are driven with the values from the matching U1 and U0 bits. The TTx and TMx signals identify the specific access type. The R/$\overline{W}$ signal is driven high for a read cycle, and the size signals (SIZx) indicate line size. $\overline{CIOUT}$ is asserted for a MOVE16 operand read if the access is identified as noncachable. Refer to **Section 3 Memory Management Unit**



NOTE: The selected device increments the value of A3 and A2.

Figure 7-17. Line Write Transfer Timing

Clock 1 (C1)

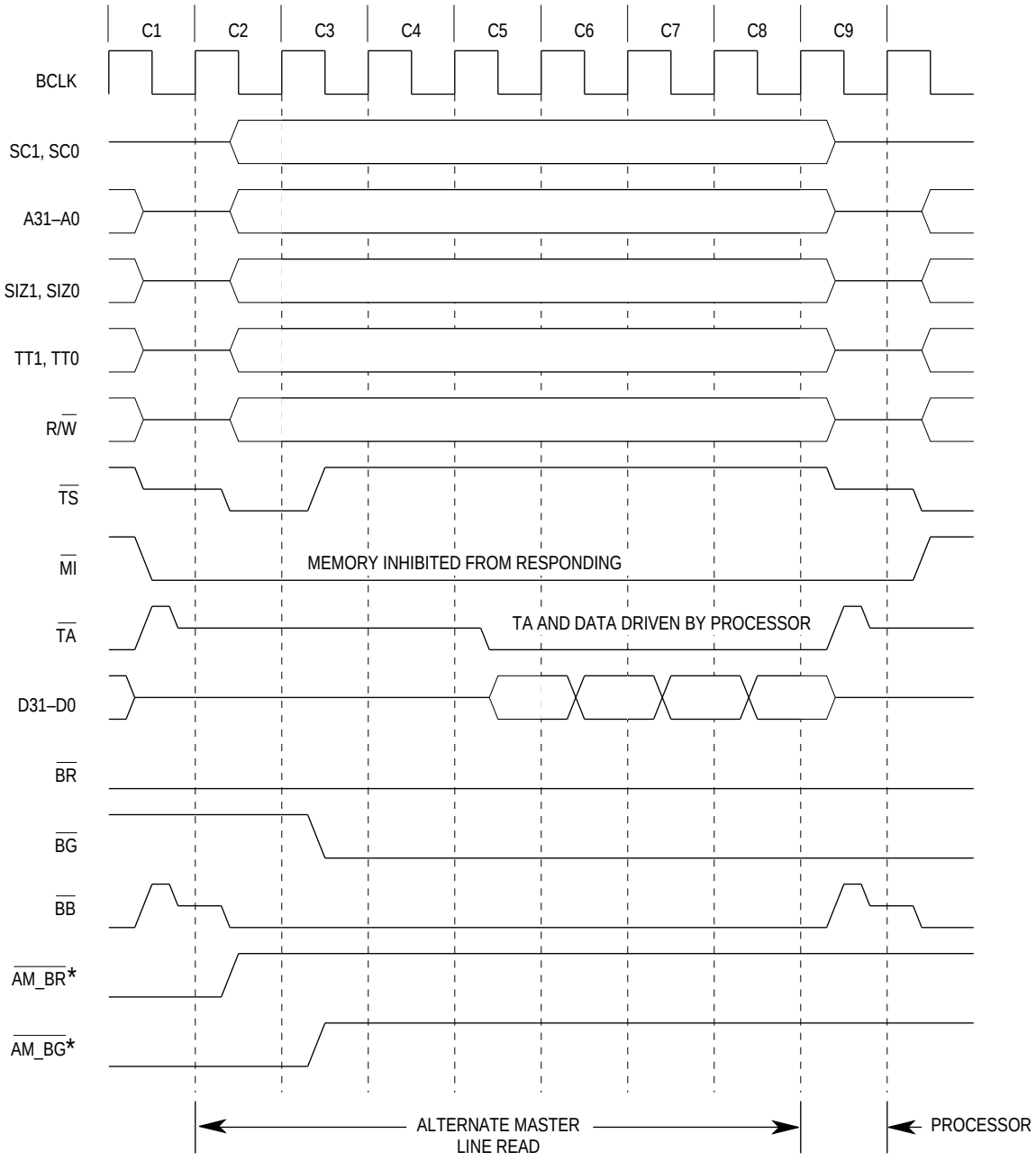
The line write cycle starts in C1. During the first half of C1, the processor places valid values on the address bus and transfer attributes. For user and supervisor mode accesses that are translated by the corresponding memory unit, UPAx signals are driven with the values from the matching U1 and U0 bits. The TTx and TMx signals identify the specific access type. The R/$\overline{W}$ signal is driven low for a write cycle, and SIZ1 and SIZ0 indicate line size. $\overline{CIOUT}$ is asserted for a MOVE16 operand read if the access is identified as noncachable. Refer to **Section 3 Memory Management Unit (Except MC68EC040 and MC68EC040V)** for information on the M68040 and

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**.



*AM indicates the alternate bus master.

Figure 7-42. Snooped Line Read, Memory Inhibited

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.

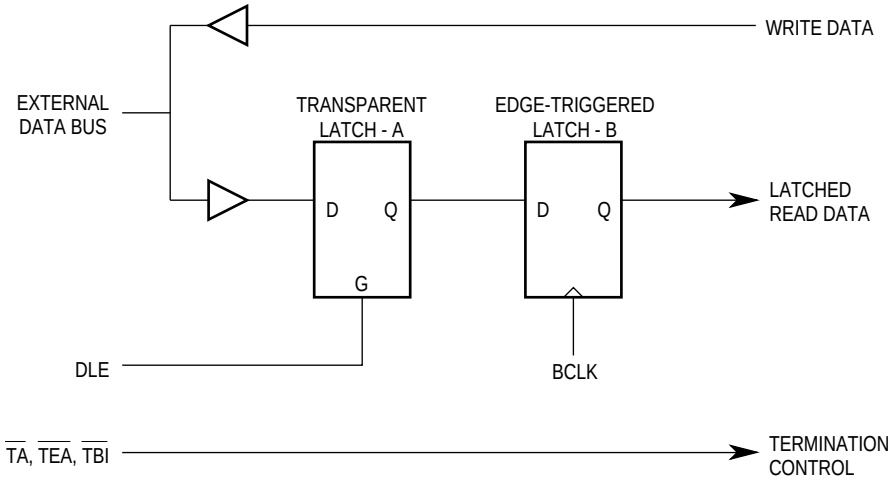


Figure 7-47. DLE Mode Block Diagram

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.

that caused the trap is also saved. Instruction execution resumes at the address in the exception vector after the required instruction is prefetched.

8.2.4 Illegal Instruction and Unimplemented Instruction Exceptions

An illegal instruction exception corresponds to vector number 4, and occurs when the processor attempts to execute an illegal instruction. An illegal instruction is an instruction that contains any bit pattern that does not correspond to the bit pattern of a valid M68040 instruction. An illegal instruction exception is also taken after a breakpoint acknowledge bus cycle is terminated, either by the assertion of the transfer acknowledge ($\overline{TA}$) or the transfer error acknowledge ($\overline{TEA}$) signal. An illegal instruction exception can also be a MOVEC instruction with an undefined register specification field in the first extension word.

Instruction word patterns with bits 15–12 equal to $A do not correspond to legal instructions for the M68040 and are treated as unimplemented instructions. $A word patterns are referred to as an unimplemented instruction with A-line opcodes. When the processor attempts to execute an unimplemented instruction with an A-line opcode, an exception is generated with vector number 10, permitting efficient emulation of unimplemented instructions. For instruction word patterns with bits 15–12 equal to $F refer to **Section 9 Floating-Point Unit (MC68040 Only)**.

Exception processing for illegal and unimplemented instructions is similar to that for instruction traps. When the processor has identified an illegal or unimplemented instruction, it initiates exception processing instead of attempting to execute the instruction. The processor copies the SR, enters the supervisor mode, and clears T1 and T0, disabling further tracing. The processor generates the vector number, either 4 or 10, according to the exception type. The illegal or unimplemented instruction vector offset, current PC, and copy of the SR are saved on the supervisor stack, with the saved value of the PC being the address of the illegal or unimplemented instruction. Instruction execution resumes at the address contained in the exception vector. It is the responsibility of the exception handling routine to adjust the stacked PC if the instruction is emulated in software or is to be skipped on return from the exception handler.

8.2.5 Privilege Violation Exception

To provide system security, some instructions are privileged. An attempt to execute one of the following privileged instructions while in the user mode causes a privilege violation exception:

ANDI to SR	FSAVE	MOVEC	PTEST
CINV	MOVE from SR	MOVES	RESET
CPUSH	MOVE to SR	ORI to SR	RTE
EORI to SR	MOVE USP	PFLUSH	STOP
FRESTORE			

Exception processing for privilege violations is similar to that for illegal instructions. When the processor identifies a privilege violation, it begins exception processing before

for the pending exception. The processor sets only one of the continuation bits when the access error stack frame is created. If the access error exception handler sets multiple bits, operation of the RTE instruction is undefined.

If the frame format field in the stack frame contains an illegal format code, a format exception occurs. If a format error or access fault exception occurs during the frame validation sequence of the RTE instruction, the processor creates a normal four-word or an access error stack frame below the frame that it was attempting to use. The illegal stack frame remains intact, so that the exception handler can examine or repair the illegal frame. In a multiprocessor system, the illegal frame can be left so that, when appropriate, another processor of a different type can use it.

The bus error exception handler can identify bus error exceptions due to instruction faults by examining the TM field in the SSW of the access error stack frame. For user and supervisor instruction faults, the TM field contains $2 and $6, respectively (see Figure 8-7). Since the processor allows all pending accesses to complete before reporting an instruction fault, the stack frame for an instruction fault will not contain any pending write-backs. The ATC bit of the SSW is used to distinguish between ATC faults and physical bus errors, and the FA field contains the logical address of the instruction prefetch. For ATC faults, the exception handler can execute a PTEST instruction (using the FA and TM fields from the SSW) to determine the specific cause of the address translation failure. After the handler corrects the cause of the fault, it executes an RTE instruction to restart execution of the instruction that contained the faulted prefetch.

For an address error fault, the processor saves a format $2 exception stack frame on the stack. This stack frame contains the PC pointing to the instruction that caused the address error as well as the actual address referenced by the instruction. Note that bit 0 of the referenced address is cleared on the stack frame. Address error faults must be repaired in software.

For a fault due to a data ATC fault or bus error, pending write-backs are also saved on the access error stack frame and must be completed by the exception handler. For the faulted access, the fault address in the FA field combined with the transfer attribute information from the SSW can be used to identify the cause of the fault. In identifying the fault, the system programmer should be aware that the data memory unit considers the read portion of read-modify-write transfers (for TAS, CAS, CAS2, and some translation table updates) a write. This prevents both read and write accesses from occurring unless all pages touched by the instruction or table update are write enabled.

All accesses other than instruction prefetches go through the data memory unit, and the M68040 treats the instruction and data address spaces as a single merged address space (the exception is the presence of separate transparent translation registers). The function codes for accesses such as PC relative operand addressing and MOVES transfers to function codes $2 and $6 (user and supervisor instruction spaces in the MC68000) are converted to data references to go through the data memory unit, and appear in the TM field of the access error stack frame as data references.

Table 8-6. Access Error Stack Frame Combinations

Main Case	SSW_RW	SSW_PUSH	WB1S		WB2S		WB3S	Easy Cleanup Data Written	Hard Cleanup Action
			1V	1M16	2V	2M16	3V		
All Read Access Errors	1[a]	No	0	X	0	X	0	None	(Note b)
	1[a]	No	0	X	0	X	1	WB3D	
	All other read cases are not possible.								
Cache Push Physical Bus Error[c]	0	Yes	0	X	0	X	0	PD3–0	(Note b)
	0	Yes	0	X	0	X	1	PD3–0, WB3D	
	0	Yes	0	X	1	0	0	PD3–0, WB2D	
	0	Yes	0	X	1	0	1	PD3–0, WB2D, WB3D	
	0	Yes	0	X	1	1	0	PD3–0, ~WB2D[d]	
Normal Write Physical bus Error	0	No	1	0	0	X	0	WB1D	(Note b)
	0	No	1	0	0	X	1	WB1D, WB3D	
	0	No	1	0	1	0	0	WB1D, WB2D	
	0	No	1	0	1	0	1	WB1D, WB2D, WB3D	
	0	No	1	0	1	1	0	WB1D, ~WB2D[d]	
MOVE16 Write Physical Bus Error	0	No	1	1	0	X	1	PD3–0, WB3D	(Note b)
	0	No	1	1	0	X	0	PD3–0	
	0	No	1	1	1	0	0	PD3–0, WB2D	
	0	No	1	1	1	0	1	PD3–0, WB2D, WB3D	
	0	No	1	1	1	1	0	PD3–0, ~WB2D[d]	
Write Page Fault	0	No	0	X	1	0	0	WB2D	Write PD3–0 and skip[e].
	0	No	0	X	1	0	1	WB2D, WB3D	
	0	No	0	X	1	1	0	~WB2D[d]	
Impossible Write Cases	0	Yes	1	X	X	X	X	(Note f)	—
	0	Don't Care	X	X	X	1	1	(Note g)	

NOTES:

a. The data memory unit stage is tied up until the bus controller passes the read back through the data memory unit and to the execution stage in the integer unit. Therefore, no pending write is possible in WB1 or WB2. WB3 could hold a pending write that was deferred due to operand read or was generated after the read.

b. If any kind of access error is reported and if a MOVE16 write is pending in the WB2 stage, then that MOVE16 read must hit in the cache so the MOVE16 can be safely restarted since it has not caused bus cycles that could retouch peripherals.

c. A cache push physical bus error is normally considered a fatal error. For these cases, the FA field is a physical address, not a logical address as in the other cases.

d. Indicates that the data should not be written even though the V-bit for it is set (WB2 corresponds to a MOVE16 write).

e. The exception handler must alter the stacked PC to point past the MOVE16 and predecrement and postincrement address registers.

f. 1V must be 0 for push exceptions.

g. The execution stage does not post a write until the MOVE16 is in the integer unit.

Table 9-5. Extended-Precision Real Format Summary (Continued)

NANs	
Sign	Don't Care
Explicit Integer Bit	Don't Care
Biased Exponent Format Maximum	32767 ($7FFF)
Mantissa	Nonzero
Representation of Mantissa Nonsignaling Signaling Nonzero Bit Pattern Created by User Mantissa When Created by FPCP	 x.1xxxx…xxxx x.0xxxx…xxxx x.xxxxx…xxxx 1.11111…1111
Approximate Ranges	
Maximum Positive Normalized	1.2×10^{4932}
Minimum Positive Normalized	1.7×10^{-4932}
Minimum Positive Denormalized	3.7×10^{-4951}

Table 9-6. Packed Decimal Real Format Summary

Data Type	SM	SE	Y	Y	3-Digit Exponent	1-Digit Integer	16-Digit Fraction
±Infinity	0/1	1	1	1	$FFF	$XXXX	$00…00
±NAN	0/1	1	1	1	$FFF	$XXXX	Nonzero
±SNAN	0/1	1	1	1	$FFF	$XXXX	Nonzero
+Zero	0	0/1	X	X	$000–$999	$XXX0	$00…00
–Zero	1	0/1	X	X	$000–$999	$XXX0	$00…00
+In-Range	0	0/1	X	X	$000–$999	$XXX0–$XXX9	$00…01–$99…99
–In-Range	1	0/1	X	X	$000–$999	$XXX0–$XXX9	$00…01–$99…99

9.4 COMPUTATIONAL ACCURACY

Whenever an attempt is made to represent a real number in a binary format of finite precision, there is a possibility that the number can not be represented exactly. This is commonly referred to as a round-off error. Furthermore, when two inexact numbers are used in a calculation, the error present in each number is reflected, and possibly aggravated, in the result. All FPU calculations use an intermediate result. When the MC68040 performs an operation, the calculation is carried out using extended-precision inputs, and the intermediate result is calculated as if to produce infinite precision. After the calculation is complete, the intermediate result is rounded to the selected precision and stored in the destination.

The FPCR encodings provide emulation for devices that only support single and double precision. The execution speed of all instructions is the same whether using single- or double-precision rounding. When using these two forced rounding precisions, the

10.7.2 Integer Unit Support Timings (Concluded)

Addressing Mode	FRESTORE <ea>*					
	Idle or Null		Short		Long	
	<ea> Calculate	Execute	<ea> Calculate	Execute	<ea> Calculate	Execute
FPn	—	—	—	—	—	—
Dn	—	—	—	—	—	—
An	—	—	—	—	—	—
(An)	13	1_L + 12	26	1_L + 25	40	1_L + 39
(An)+	13	1_L + 12	26	1_L + 25	40	1_L + 39
–(An)	—	—	—	—	—	—
(d_{16},An)	13	1_L + 12	26	1_L + 25	40	1_L + 39
(d_{16},PC)	—	—	—	—	—	—
(xxx).W, (xxx).L	14	1_L + 12	27	1_L + 25	41	1_L + 39
#<xxx>	—	—	—	—	—	—
(d_8,An,Xn)	14	14	27	27	41	41
(d_8,PC,Xn)	—	—	—	—	—	—
(An,Xn)	16	1_L + 14	29	1_L + 27	43	1_L + 41
(bd,An,Xn)	17	1_L + 15	30	1_L + 28	44	1_L + 42
([bd,An,Xn])	20	1_L + 19	33	1_L + 32	47	1_L + 46
([bd,An,Xn],od)	21	1_L + 19	34	1_L + 32	48	1_L + 46
([bd,An],Xn)	21	3_L + 18	34	3_L + 31	48	3_L + 45
([bd,An],Xn,od)	22	3_L + 19	35	3_L + 31	49	3_L + 45

*Timings are for an idle floating-point unit.

10.7.3 Timings in the Floating-Point Unit (Continued)

Instruction	Opclass	Size	Precision	Operands	Conversion	Execution	Normalization
FDIV	2	S,D	Any	— ,NAN	4	0	0
	2	X	Any	Norm,Norm	3(4)	37.5	2(3)
	2	X	Any	— ,Zero	5	0	0
	2	—	Any	— ,Inf	5	0	0
	2	X	Any	— ,NAN	5	0	0
FSQRT	0	—	Any	Norm	2(3)	103	2(3)
	0	—	Any	(Zero|Inf|NAN)	4	0	0
	2	S,D	Any	Norm	2(3)	103	2(3)
	2	S,D	Any	(Zero|Inf|NAN)	4	0	0
	2	X	Any	Norm	3(4)	103	2(3)
	2	X	Any	(Zero|Inf|NAN)	5	0	0
FMOVE,	0	—	X	(Norm|Zero|Inf)	2	0	0
FABS,	0	—	X	NAN	3	0	0
FNEG	0	—	S,D	Norm	5	0	0
	0	—	S,D	(Zero|Inf)	3	0	0
	0	—	S,D	NAN	4	0	0
	2	S	Any	(Norm|Zero|Inf)	3	0	0
	2	S	Any	NAN	4	0	0
	2	D	D,X	(Norm|Zero|Inf)	3	0	0
	2	D	D,X	NAN	4	0	0
	2	D	S	Norm	5	0	0
	2	D	S	(Zero|Inf)	4	0	0
	2	D	S	NAN	5	0	0
	2	X	X	(Norm|Zero|Inf)	4	0	0
	2	X	X	NAN	5	0	0
	2	X	S,D	Norm	6	0	0
	2	X	S,D	(Zero|Inf)	5	0	0
	2	X	S,D	NAN	6	0	0
	2	B,W	Any	(+Norm|Zero)	1.5(11)	4.5	2
	2	L	D,X	(+Norm|Zero)	1.5(11)	4.5	2
	2	L	S	(+Norm|Zero)	1.5(12.5)	4.5	2
	2	B,W	Any	—Norm	1.5(11.5)	5	2
	2	L	D,X	—Norm	1.5(11.5)	5	2
	2	L	S	—Norm	1.5(13)	5	2
FMOVE	3	S,D	Any	Any	3	0	0
	3	X	Any	Any	4	0	0
	3	B,W,L	Any	+(Norm|Zero)	3(9)	1.5	3.5
	3	B,W,L	Any	–(Norm|Zero)	3(10)	1.5	4.5

$\overline{RSTO}$ are reset at the completion of the RESET instruction. An $\overline{RSTI}$ signal that is asserted to the processor during execution of a RESET instruction immediately resets the processor and causes $\overline{RSTO}$ to negate. $\overline{RTSO}$ can be logically ANDed with the external signal driving $\overline{RTSI}$ to derive a system reset signal that is asserted for both an external processor reset and execution of a RESET instruction.

B.5 EXCEPTION PROCESSING

The MC68EC040 provides five different stack frames for exception processing and allows for a MC68040-specific stack frame. Refer to **Section 8 Exception Processing** for details on exception processing.

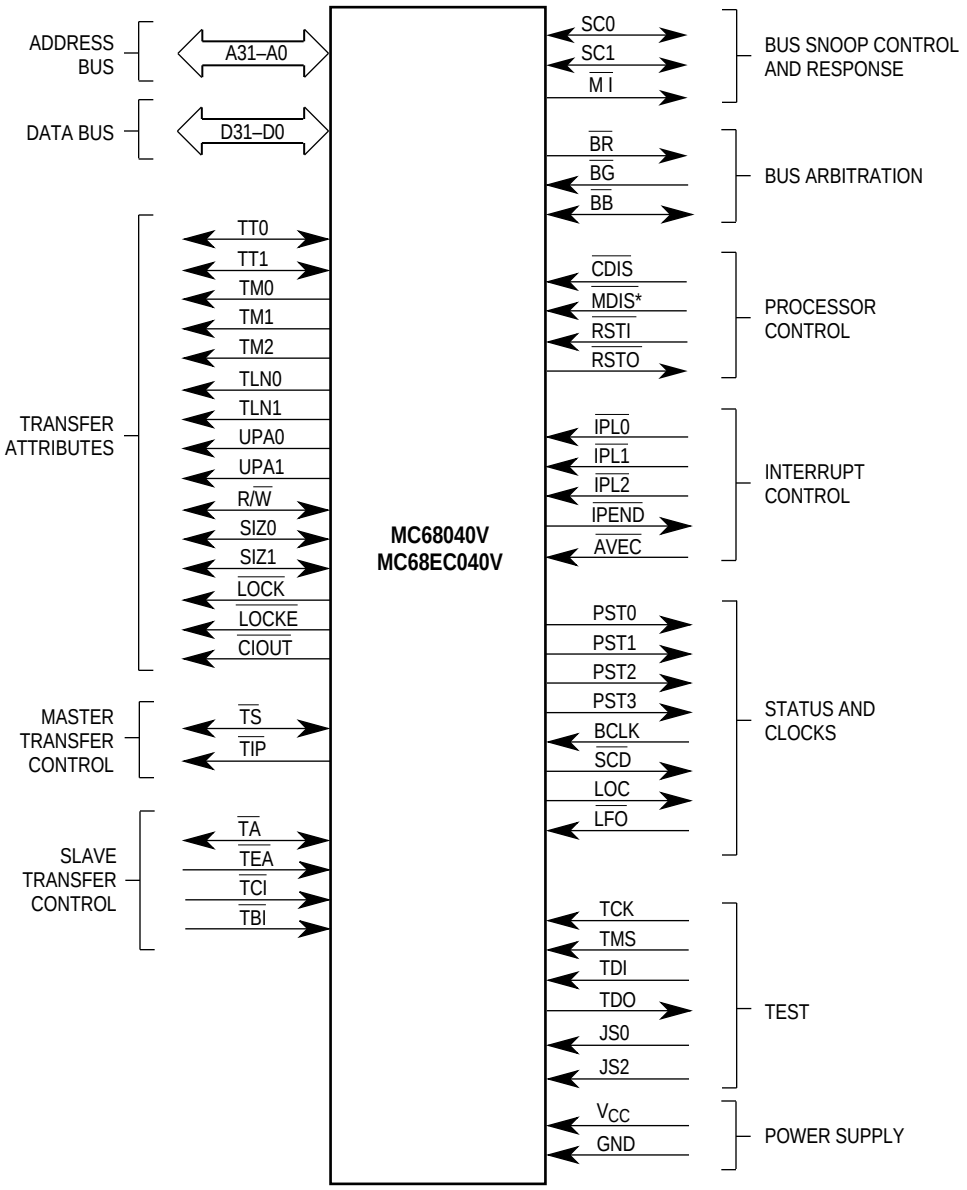
B.5.1 Unimplemented Floating-Point Instructions and Exceptions

All legal MC68040 and MC68881/MC68882 floating-point instructions are defined as unimplemented floating-point instructions on the MC68EC040. These instructions generate an eight-word stack frame (format $4) during exception processing before taking an F-line exception. These instructions trap as an F-line exception and can be emulated in software by the F-line exception handler to maintain user-object-code compatibility.

The MC68EC040 assists the emulation process by distinguishing unimplemented floating-point instructions from other unimplemented F-line instructions. To aid emulation, the effective address is calculated and saved in the format $4 stack frame. This simplifies and speeds up the emulation process by eliminating the need for the emulation routine to determine the effective address and by providing information required to emulate the instruction on the exception stack frame in the supervisor address space. However, the floating-point instruction can reside in user space; therefore, the floating-point unimplemented exception handler may need to access user instruction space. The following processing steps occur for an unimplemented floating-point instruction:

1. When an unimplemented floating-point instruction is encountered, the instruction is partially decoded, and the effective address is calculated, if required.
2. The processor waits for all previous integer instructions, write-backs, and associated exception processing to complete before beginning exception processing for the unimplemented floating-point instruction. Any access error that occurs in completing the write-backs causes an access error exception, and the resulting stack frame indicates a pending unimplemented floating-point instruction exception. The access error exception handler then completes the write-backs in software, and exception processing for the unimplemented floating-point instruction exception begins immediately after return from the access error handler.
3. 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 and T0). It creates a format $4 stack frame and saves the internal copy of the SR, PC, vector offset, calculated effective address, and PC value of the faulted instruction in the stack frame.

   The effective address field of the format $4 stack frame contains the calculated effective address of the operand for the faulted floating-point instruction using the addressing mode in which the effective address is calculated. For immediate and register



NOTE: *This signal is JS1 on the MC68EC040V.

Figure C-1. MC68040V and MC68EC040V Functional Signal Groups

C.2 LOW-POWER STOP MODE

The low-power stop mode is a reduced power mode of operation, that causes the MC68040V and MC68EC040V to remain quiescent until either a reset or non-masked interrupt occurs. This mode of operation has four phases of operation and is triggered by the low-power stop (LPSTOP) instruction:

1. Perform a LPSTOP broadcast cycle.
2. End integer unit (IU) instruction pipeline sequencing, which is similar to the STOP instruction sequence (IMM data ➧ SR), at termination of the LPSTOP broadcast cycle.

Stack Pointer

MC68000, MC68008, MC68010	USP, SSP
MC68020, MC68030, MC68040	USP, SSP (MSP, ISP)

Status Register Bits

MC68000, MC68008, MC68010	T, S, I0/I1/I2, X/N/Z/V/C
MC68020, MC68030, MC68040	T0, T1, S, M, I0/I1/I2, X/N/Z/V/C

Function Code/Address Space

MC68000, MC68008	FC2–FC0 = 7 Is Interrupt Acknowledge Only
MC68010, MC68020, MC68030, MC68040	FC2–FC0 = 7 Is CPU Space
MC68040	User, Supervisor, and Acknowledge

Indivisible Bus Cycles

MC68000, MC68008, MC68010	Use $\overline{AS}$ Signal
MC68020, MC68030	Use $\overline{RMC}$ Signal
MC68040	Use $\overline{LOCK}$ and $\overline{LOCKE}$ Signal

Stack Frames

MC68000, MC68008	Supports Original Set
MC68010	Supports Formats $0, $8
MC68020, MC68030	Supports Formats $0, $1, $2, $9, $A, $B
MC68040	Supports Formats $0, $1, $2, $3, $7
MC68EC040, MC68LC040	Supports Formats $0, $1, $2, $3, $4, $7

Addressing Modes

MC68020, MC68030, and MC68040 Extensions	Memory indirect addressing modes, scaled index, and larger displacements. Refer to specific data sheets for details.

APPENDIX E
FLOATING-POINT EMULATION (M68040FPSP)

The MC68040 is user-object-code compatible with the MC68030 and MC68881/MC68882. The MC68040 floating-point unit is optimized to directly execute the most commonly used subset of the extensive MC68881/MC68882 instruction set through hardware. Special traps and stack frames for the unimplemented instructions and data types provide support for the remaining instructions. These functions coupled with Motorola's floating-point software package (M68040FPSP) ensure complete user-object-code compatibility.

There are two versions of the M68040FPSP, one for applications compiled for the MC68881/MC68882 (kernel version) and the other for applications compiled for the MC68040 (library version). System integrators can install the kernel version as part of an MC68040-based operating system. The kernel version is used to execute preexisting user object code written for the MC68881/MC68882 as part of the operating system. User applications need not be recompiled or modified in any way once the kernel version is installed.

The MC68040 compiler writer and system integrator use the library version which provides less overhead than the kernel version. Overhead is reduced because the appropriate floating-point exception routine is called directly rather than taking an unimplemented instruction trap. The library is M68000 application binary interface (ABI) and IEEE exception-reporting compliant; it is not UNIX® exception-reporting compliant.

The M68040FPSP provides the following features:

- Arithmetic and Transcendental Instructions
- IEEE-Compliant Exception Handlers
- MC68040 Unimplemented Data Type and Data Format Handlers
- Can Reside in a 64-Kbyte ROM
- Code Is Reentrant

The M68040FPSP satisfies the IEEE *Standard 754 for Binary Floating-Point Arithmetic*. The average 25-MHz performance of the transcendental function subroutines is equivalent to that of the 33-MHz MC68881/MC68882. The error bound is equivalent to that of the MC68881/MC68882.

®UNIX is a registered trademark of AT&T Bell Laboratories.