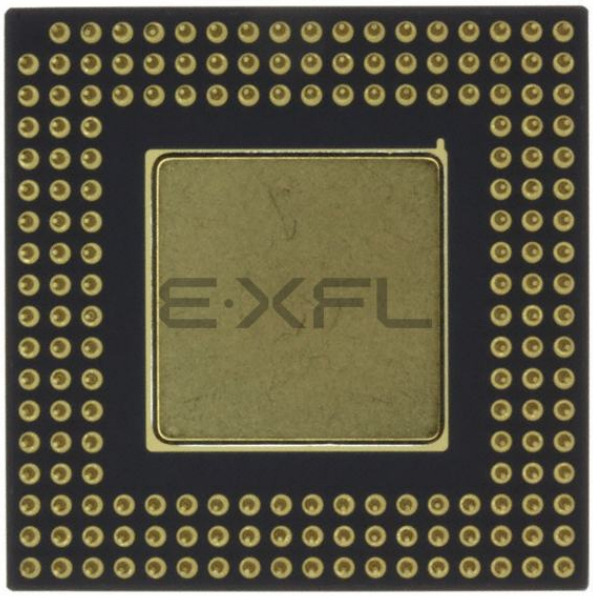


Welcome to E-XFL.COM

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	25MHz
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	179-BPGA
Supplier Device Package	179-PGA (47.24x47.24)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68lc040rc25a

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

3.2.6 Address Translation Protection

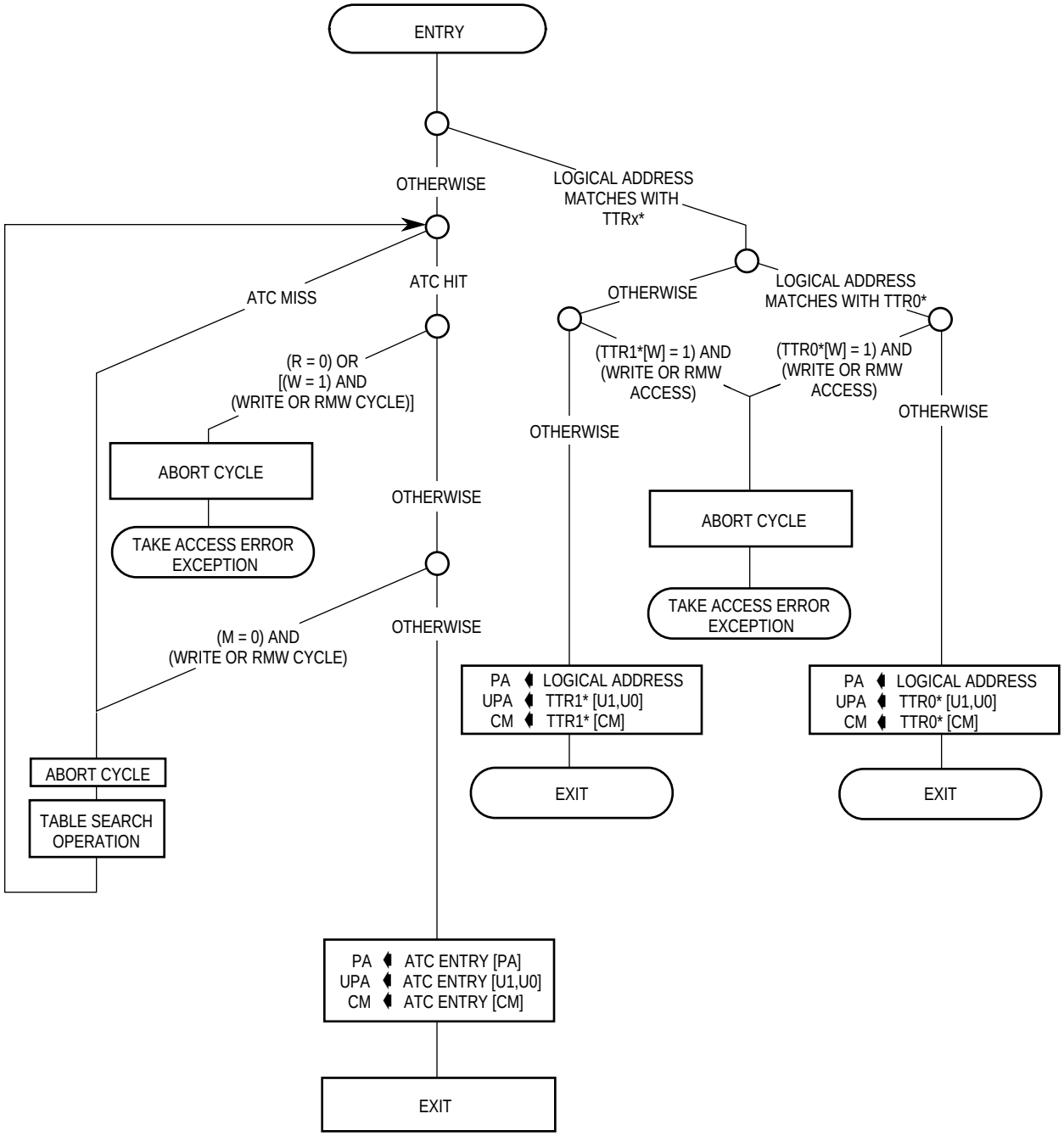
The M68040 MMUs provide separate translation tables for supervisor and user address spaces. The translation tables contain both mapping and protection information. Each table and page descriptor includes a write-protect (W) bit that can be set to provide write protection at any level. Page descriptors also contain a supervisor-only (S) bit that can limit access to programs operating at the supervisor privilege level.

The protection mechanisms can be used individually or in any combination to protect:

- Supervisor address space from accesses by user programs.
- User address space from accesses by other user programs.
- Supervisor and user program spaces from write accesses (implicitly supported by designating all memory pages used for program storage as write protected).
- One or more pages of memory from write accesses.

3.2.6.1 SUPERVISOR AND USER TRANSLATION TABLES. One way of protecting supervisor and user address spaces from unauthorized accesses is to use separate supervisor and user translation tables. Separate trees protect supervisor programs and data from accesses by user programs and user programs and data from access by supervisor programs. Access is granted to the supervisor programs that can accesses any area of memory with MOVES. The translation table pointed to by the SRP is selected for all other supervisor mode accesses. This translation table can be common to all tasks. Figure 3-17 illustrates separate translation tables for supervisor accesses and for two user tasks that share the common supervisor space. Each user task has an translation table with unique mappings for the logical addresses in its user address space.

3.2.6.2 SUPERVISOR ONLY. A second mechanism protects supervisor programs and data without requiring segmenting of the logical address space into supervisor and user address spaces. Page descriptors contain S-bits to protect areas of memory from access by user programs. When a table search for a user access encounters an S-bit set in a page descriptor, the table search ends, and an ATC descriptor corresponding to the logical address is created with the S-bit set. A subsequent retry of the user access results in an access error exception being taken. The S-bit can be used to protect one or more pages from user program access. Supervisor and user mode accesses can share descriptors by using indirect descriptors or by sharing tables. The entire user and supervisor address spaces can be mapped together by loading the same root pointer address into both the SRP and URP registers.



* Refers to either instruction or data transparent translation register.

Figure 3-22. Address Translation Flowchart

MC68030 and MC68851 cause F-line unimplemented instruction exceptions if executed in either supervisor or user mode by the M68040.

3.7.4 Register Programming Considerations

If the entries in the ATCs are no longer valid when a reset operation occurs (as is normally expected), an explicit flush operation must be specified by the system software. The assertion of RSTI disables translation by clearing the E-bits of the TCR, DTTRx, and ITTRx, but it does not flush the ATCs. Reading or writing any of the MMU registers (URP, SRP, TCR, MMUSR, DTTR0, DTTR1, ITTR0, ITTR1) does not flush the ATCs. Since a write to these registers can cause some or all the address translations to change, the write should be followed by a PFLUSH operation to flush the ATCs if necessary.

The status bits in the MMUSR indicate conditions to which the operating system should respond. In a typical access error exception handler, the flowchart illustrated in Figure 3-23 can be used to determine the cause of an MMU fault. The PTEST instruction sets the bits in the MMUSR appropriately, and the program can branch to the appropriate code segment for the condition.

Table 4-4. Data-Cache Line State Transitions

Cache Operation	Current State					
	Invalid Cases		Valid Cases		Dirty 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.	D1	Buffer dirty cache line; read new line from memory; supply data to CPU and update cache; write buffered dirty data to memory; go to valid state.
CPU Read Hit	I2	Not Possible	V2	Supply data to CPU; remain in current state.	D2	Supply data to CPU; remain in current state.
CPU Write Miss (Copyback)	I3	Read line from memory into cache; write data to cache; set Dn bits of modified long words; go to dirty state.	V3	Read line from memory into cache (replacing old line); write data to cache and set Dn bits; go to dirty state.	D3	Buffer dirty cache line; read new line from memory; write data to cache and set Dn bits; write buffered dirty data to memory; remain in current state.
CPU Write Miss (Write-through)	I4	Write data to memory; remain in current state.	V4	Write data to memory; remain in current state.	D4	Write data to memory; remain in current state (see note).
CPU Write Hit (Copyback)	I5	Not Possible	V5	Write data into cache; set Dn bits of modified long words; go to dirty state.	D5	Write data in cache; set Dn bits of modified long words; remain in current state.
CPU Write Hit (Write-through)	I6	Not Possible	V6	Write data to cache; write data to memory; remain in current state.	D6	Write data into cache (no change to Dn bits); write data to memory; remain in current state (see note).
Cache Invalidate (CINV)	I7	No action; remain in current state.	V7	No action; go to invalid state.	D7	No action (dirty data lost); go to invalid state.
Cache Push (CPUSH)	I8	No action; remain in current state.	V8	No action; go to invalid state.	D8	Write dirty data to memory; go to invalid state.
Alternate Master Read Hit (Snoop Control = 01 — Leave Dirty)	I9	Not Possible	V9	No action; remain in current state.	D9	Inhibit memory and source data; remain in current state.

NOTE: Dirty state transitions D4 and D6 are the result of a system programming error and should be avoided even though they are technically valid.

5.3.2 Transfer Modifier (TM2–TM0)

These three-state outputs provide supplemental information for each transfer type. Table 5-3 lists the encoding for normal and MOVE16 transfers, and Table 5-4 lists the encoding for alternate access transfers. For interrupt acknowledge transfers, the TMx signals carry the interrupt level being acknowledged; for breakpoint acknowledge transfers and LPSTOP broadcast cycles on the MC68040V and MC68EC040V, the TMx signals are low. When the M68040 is not the bus master, the TMx signals are set to a high-impedance state.

Table 5-3. Normal and MOVE16 Access Transfer Modifier Encoding

TM2	TM1	TM0	Transfer Modifier
0	0	0	Data Cache Push Access
0	0	1	User Data Access*
0	1	0	User Code Access
0	1	1	MMU Table Search Data Access
1	0	0	MMU Table Search Code Access
1	0	1	Supervisor Data Access*
1	1	0	Supervisor Code Access
1	1	1	Reserved

* MOVE16 accesses use only these encodings.

Table 5-4. Alternate Access Transfer Modifier Encoding

TM2	TM1	TM0	Transfer Modifier
0	0	0	Logical Function Code 0
0	0	1	Reserved
0	1	0	Reserved
0	1	1	Logical Function Code 3
1	0	0	Logical Function Code 4
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Logical Function Code 7

5.3.3 Transfer Line Number (TLN1, TLN0)

These three-state outputs indicate which line in the set of four data cache lines is being accessed for normal push and line data read accesses. TLNx signals are undefined for all other accesses to instruction space and are placed in a high-impedance state when the processor relinquishes the bus.

Table 5-6. Processor Status Encoding

Hex	PST3	PST2	PST1	PST0	Internal Status
0	0	0	0	0	User, Start/Continue Current Instruction
1	0	0	0	1	User, End Current Instruction
2	0	0	1	0	User, Branch Not Taken/End Current Instruction
3	0	0	1	1	User, Branch Taken/End Current Instruction
4	0	1	0	0	User, Table Search
5	0	1	0	1	Halted State (Double Bus Fault)
6	0	1	1	0	Low-Power Stop Mode (Supervisor Instruction)*
7	0	1	1	1	Reserved
8	1	0	0	0	Supervisor, Start/Continue Current Instruction
9	1	0	0	1	Supervisor, End Current Instruction
A	1	0	1	0	Supervisor, Branch Not Taken/End Current Instruction
B	1	0	1	1	Supervisor, Branch Taken/End Current Instruction
C	1	1	0	0	Supervisor, Table Search
D	1	1	0	1	Stopped State (Supervisor Instruction)
E	1	1	1	0	RTE Executing
F	1	1	1	1	Exception Stacking

NOTE: *MC68040V and MC68EC040V only.

When a ‘branch taken/end current instruction’ is indicated, it means that a change of instruction flow is pending. Along with the following instructions, an exception stacking (encoding F) sequence is ended with the ‘supervisor, branch taken/end current instruction’ encoding as though it were a virtual JMP instruction. This includes all the possible exceptions listed in the processor’s vector table. Instructions that cause a ‘branch taken/end current instruction’ encoding when they are executed are as follows:

ANDI to SR	DBcc (Taken)	MOVE to SR	RTD
Bcc (Taken)	FBcc (Taken)	MOVE USP	RTE
BRA	FDBcc (Always)	MOVEC	RTR
BSR	FMOVEM Rc,MRn	MOVES	RTS
CAS	FMOVEM FPm,MRn	NOP	STOP
CAS2	FSAVE	ORI to SR	TAS
CINV	JMP	PFLUSH	
CPUSH	JSR	PTEST	

The Bcc (not taken) and DBcc (not taken) are the only instructions that cause a ‘branch not taken/end current instruction’ encoding. Note that the FBcc (not taken) is not included in this category. The FBcc (not taken) instruction ends with an ‘end current instruction’ encoding. All other instructions and conditions end with the ‘end current instruction’ encoding. For instance, if the processor is running back-to-back single clock instructions, the encoding ‘end current instruction’ remains asserted for as many clock cycles as instructions.

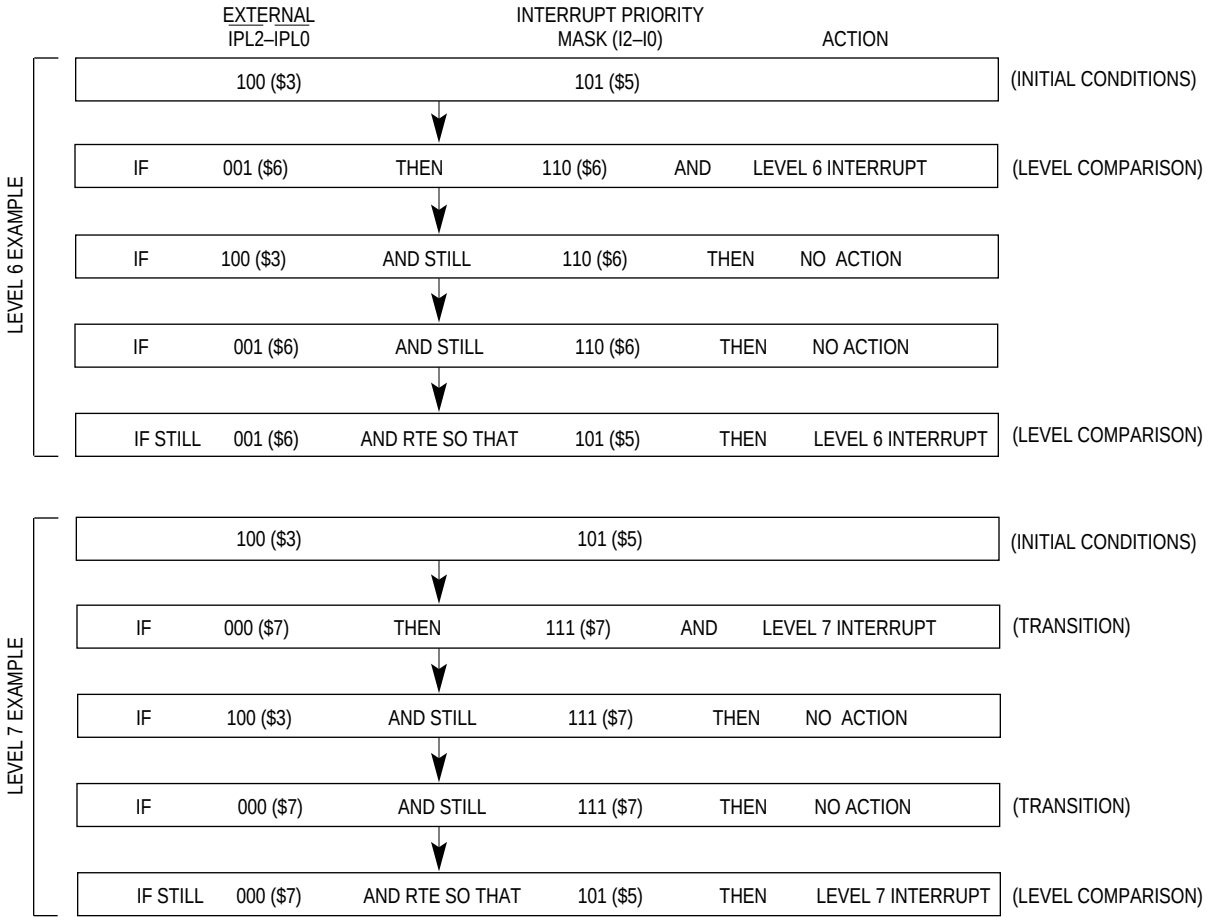


Figure 8-3. Interrupt Recognition Examples

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.

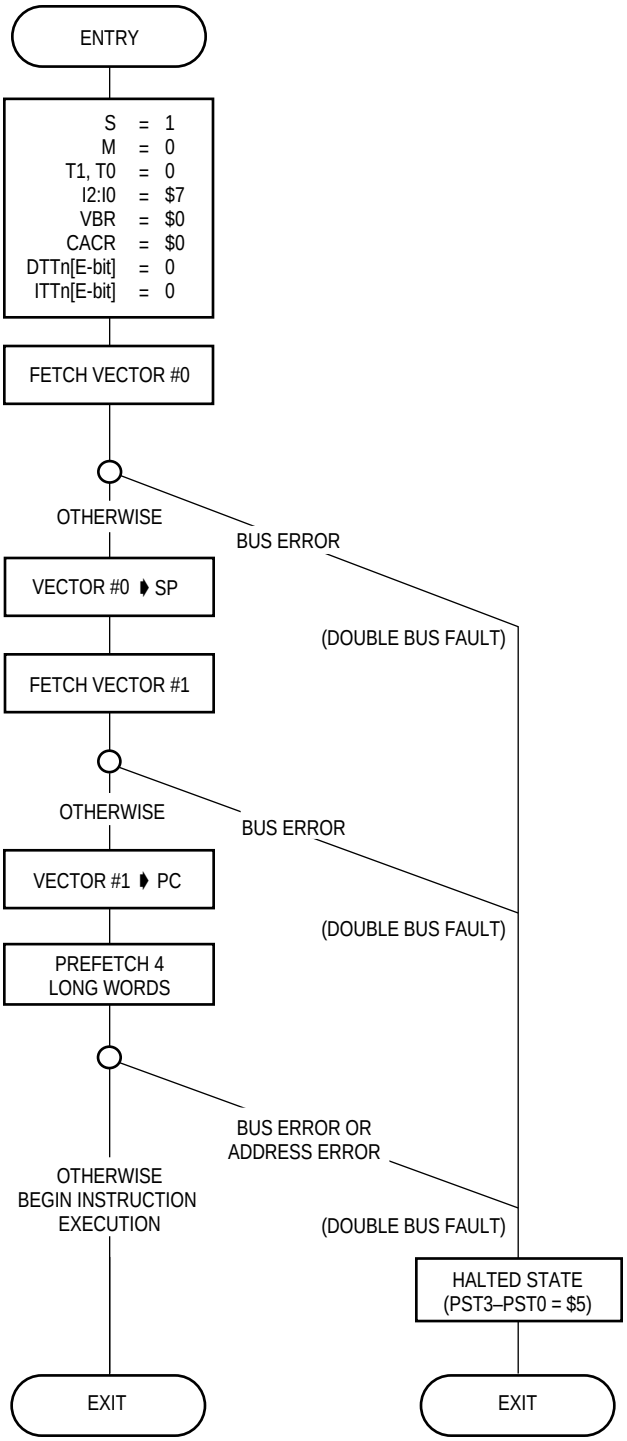


Figure 8-5. Reset Exception Processing Flowchart

After the initial instruction is prefetched, program execution begins at the address in the PC. The reset exception does not flush the ATCs or invalidate entries in the instruction or data caches; it does not save the value of either the PC or the SR. If an access fault or address error occurs during the exception processing sequence for a reset, a double bus fault is generated. The processor halts, and the processor status (PST3–PST0) signals indicate $5. Execution of the reset instruction does not cause a reset exception, or affect

exception instead of executing the exception handler for the original exception condition. For example, if simultaneous interrupt and trap exceptions are pending, the exception processing for the trap exception occurs first, followed immediately by exception processing for the interrupt. When the processor resumes normal instruction execution, it is in the interrupt handler, which returns to the trap exception handler.

- Exception processing for access error exceptions creates a format $7 stack frame that contains status information that can indicate a pending trace, floating-point post-instruction, or unimplemented floating-point instruction exception. The RTE instruction used to return from the access error exception handler checks the status bits for one of these pending exceptions. If one is indicated, the RTE changes the access error stack frame to match the pending exception and fetches the vector for the exception. Instruction execution then resumes in the new exception handler.
- If an access error, trace, and one of the two (mutually exclusive) floating-point exceptions occur simultaneously, the pending floating-point exception is indicated in the access error stack and the trace exception flag is undefined. The exception handler for the floating-point exception must check the trace bits on the stack and call the trace handler directly (after adjusting the stack frame to match the format for the trace exception).
- If a trace exception is pending at the same time an exception priority level 3 or floating-point post-instruction exception is pending, the trace exception is not reported, and the exception handler for the other exception condition must check for the trace condition.

8.4 RETURN FROM EXCEPTIONS

After the processor has completed executing the exception handlers for all pending exceptions, the processor resumes normal instruction execution at the address in the processor's vector table for the last exception processed. Once the exception handler has completed execution, if possible the processor must return the system context as it was prior to the exception using the RTE instruction. (If the internal data of the exception stack frames are manipulated, M68040 may enter into an undefined state; this applies specifically to the SSW on the access error stack frame.)

When the processor executes an RTE instruction, it examines the stack frame on top of the active supervisor stack to determine if it is a valid frame and what type of context restoration it requires. If during restoration, a stack frame has an odd address PC and an SR that indicates user trace mode enabled, then an address error is taken. The SR stacked for the address error has the SR S-bit set. For previous members of the M68000 family the S-bit is clear. When the M68040 writes or reads a stack frame, it uses long-word operand transfers wherever possible. Using a long-word-aligned stack pointer greatly enhances exception processing performance. The processor does not necessarily read or write the stack frame data in sequential order. The system software should not depend on a particular exception generating a particular stack frame. For compatibility with future devices, the software should be able to handle any format of stack frame for any type of exception. The following paragraphs discuss in detail each stack frame format.

Table 9-16. State Frame Field Information (Continued)

FSAVE State Frame Field	Contents
OVFL (FMOVE to Memory)	
CMDREG1B	FMOVE instruction command word
FPTEMP	Intermediate result with mantissa rounded to correct precision.
STAG	Source Operand Tag = Normalized
E1	Always 1
T	Always 1
UNFL (FMOVE to Register, FABS, and FNEG)	
CMDREG1B	Exception Instruction Command Word
FPTEMP	Unrounded, Extended-Precision Intermediate Result
STAG	Source Operand Tag = Normalized
E1	Always 1
T	Always 0
UNFL (FADD, FSUB, FMUL, FDIV, and FSQRT)	
CMDREG3B	Encoded Exception Instruction Command Word
WBTEMP	WBTS, WBTE, and WBTM = intermediate result sign, biased 15-bit exponent, and 64-bit mantissa prior to rounding.
WBTE15	Bit 15 of the intermediate result's 16-bit exponent = 1 for underflow.
WBTM1, WBTM0, SBIT	Guard, round, and sticky of intermediate result’s 67-bit mantissa.
E3	Always 1
T	Either 1 or 0
UNFL (FMOVE to Memory)	
CMDREG1B	FMOVE Instruction Command Word
FPTEMP	Intermediate result with mantissa prior to rounding.
STAG	Source Operand Tag = Normalized
E1	Always 1
T	Always 1
DZ	
CMDREG1B	M68040FPSP divide by zero can generate.
ETEMP	Source operand is converted to extended precision.
STAG	Source Operand Tag
FPTEMP	Destination operand is converted to extended precision.
E1	Always 1
T	Always 0

10.6 INTEGER UNIT INSTRUCTION TIMINGS (Continued)

Addressing Mode	BFFFO[a,b]		BFINS[a,c]		BFTST[a]	
	<ea> Calculate	Execute	<ea> Calculate	Execute	<ea> Calculate	Execute
Dn	3/4[d]	6/7[d]	2/3[d]	5/6[d]	1/2[d]	3/4[d]
An	—	—	—	—	—	—
(An)	9	2_L + 9	9	2_L + 7	9	2_L + 7
(An)+	—	—	—	—	—	—
–(An)	—	—	—	—	—	—
(d_{16},An)	9	2_L + 9	9	2_L + 7	9	2_L + 7
(d_{16},PC)	10	3_L + 9	—	—	10	3_L + 7
(xxx).W, (xxx).L	9	2_L + 9	9	2_L + 7	9	2_L + 7
#<xxx>	—	—	—	—	—	—
(d_8,An,Xn)	10	12	10	10	10	10
(d_8,PC,Xn)	11	1_L + 12	—	—	11	1_L + 10
(BR,Xn)	13	1_L + 14	13	1_L + 12	13	1_L + 12
(bd,BR,Xn)	14	1_L + 15	14	1_L + 13	14	1_L + 13
([bd,BR,Xn])	16	1_L + 17	16	1_L + 15	16	1_L + 15
([bd,BR,Xn],od)	17	1_L + 18	17	1_L + 16	17	1_L + 16
([bd,BR],Xn)	17	3_L + 16	17	3_L + 14	17	3_L + 14
([bd,BR],Xn,od)	18	3_L + 17	18	3_L + 15	18	3_L + 15

NOTES:
a. This instruction interlocks the <ea> calculate and execute stages.
b. If the bit field spans a long-word boundary, add two clocks to the execute time. Two memory addresses are accessed in this case.
c. If the bit field spans a long-word boundary, add seven clocks to both the <ea> calculate and execute times. Two memory addresses are accessed in this case.
d. If the bit field spans a long-word boundary, add ten and nine clocks to both the <ea> calculate and execute times, respectively. Two memory addresses are accessed in this case.

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.

11.3 DC ELECTRICAL SPECIFICATIONS (V_{CC} = 5.0 VDC ±5 %)

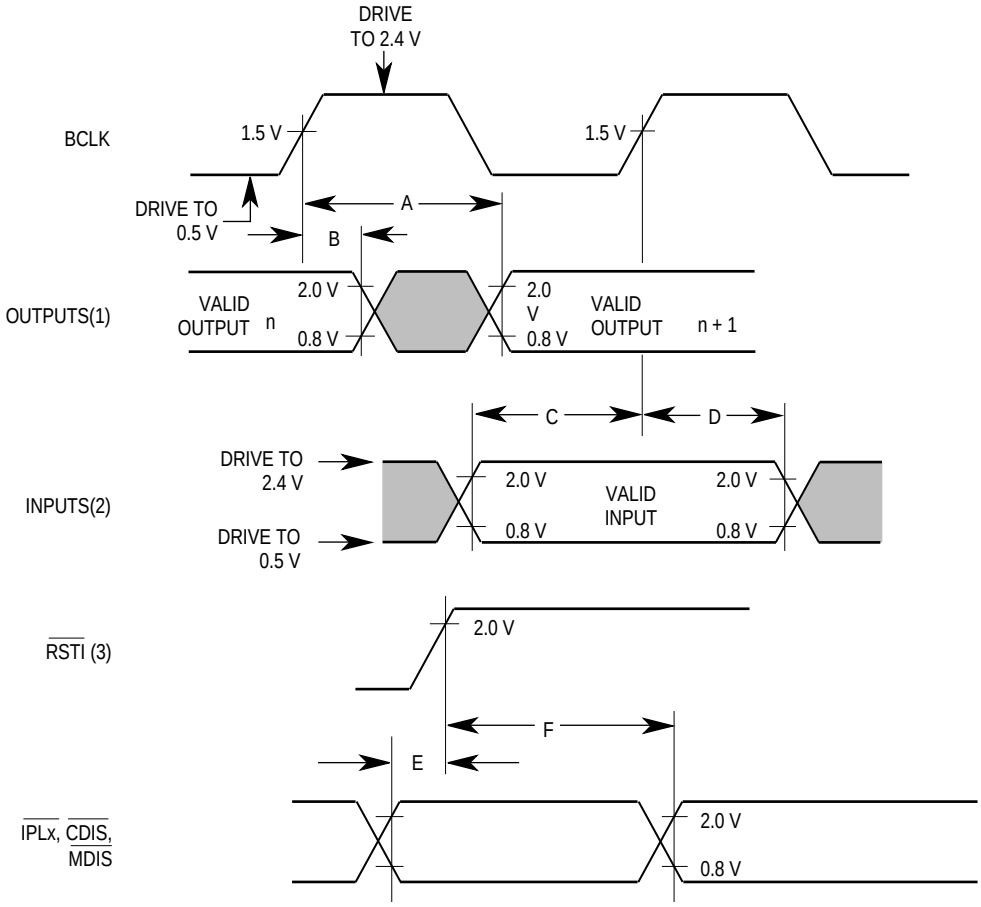
Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	2	V_{CC}	V
Input Low Voltage	V_{IL}	GND	0.8	V
Undershoot	—	—	0.8	V
Input Leakage Current @ 0.5/2.4 V $\overline{AVEC}$, BCLK, $\overline{BG}$, $\overline{CDIS}$, $\overline{MDIS}$, $\overline{IPLx}$, PCLK, $\overline{RSTI}$, SCx, $\overline{TBI}$, TLNx , $\overline{TCI}$, TCK, $\overline{TEA}$	I_{in}	20	20	µA
Hi-Z (Off-State) Leakage Current @ 0.5/2.4 V An, $\overline{BB}$, $\overline{CIOUT}$, Dn, $\overline{LOCK}$, $\overline{LOCKE}$, R/$\overline{W}$, SIZx, $\overline{TA}$, TDO, $\overline{TIP}$, TMx, TLNx, $\overline{TS}$, TTx, UPAx	I_{TSI}	20	20	µA
Signal Low Input Current, V_{IL} = 0.8 V TMS, TDI, $\overline{TRST}$	I_{IL}	–1.1	–0.18	mA
Signal High Input Current, V_{IH} = 2.0 V TMS, TDI, $\overline{TRST}$	I_{IH}	–0.94	–0.16	mA
Output High Voltage, I_{OH} = 5 mA (Small Buffer Mode)	V_{OH}	2.4	—	V
Output Low Voltage, I_{OL} = 5 mA (Small Buffer Mode)	V_{OL}	—	0.5	V
Output High Voltage, I_{OH} = 55 mA (Large Buffer Mode)	V_{OH}	2.4	—	V
Output Low Voltage, I_{OL} = 55 mA (Large Buffer Mode)	V_{OL}	—	0.5	V
Capacitance*, V_{in} = 0 V, f = 1 MHz	C_{in}	—	25	pF

*Capacitance is periodically sampled rather than 100% tested.

11.4 POWER DISSIPATION

Buffer Mode	25 MHz	33 MHz	40 MHz
Worst Case (V_{CC} = 5.25 V , T_A = 0°C)			
Small Unterminated, I_{OL} = I_{OH} = 5 mA	4.9 W	6.2 W	7.2 W
Large Unterminated, I_{OL} = I_{OH} = 5 mA	5.1 W	6.6 W	7.7 W
Large Terminated, 50 Ω, 2.5 V, I_{OL} = I_{OH} = 55 mA	6.5 W	8.0 W	9.1 W
Typical Values (V_{CC} = 5 V, T_J = 90°C)*			
Small	3.0 W	4.1 W	4.5 W
Large Unterminated	3.3 W	4.4 W	4.8 W
Large Terminated, 50 Ω, 2.5 V	4.7 W	5.8 W	6.2 W

*This information is for system reliability purposes.



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.
3. This timing is applicable to all parameters specified relative to the negation of the RSTI signal.

LEGEND:

A. Maximum output delay specification.
B. Minimum output hold time.
C. Minimum input setup time specification.
D. Minimum input hold time specification.
E. Mode select setup time to RSTI negated.
F. Mode select hold time from RSTI negated.

Figure 11-2. Drive Levels and Test Points for AC Specifications

REV2.3 (01/31/2000)

APPENDIX B
MC68EC040

NOTE

Rev. 2.3 contains timing informationg for 40 MHz operation. Refer to chang bars for these additions.

All references to MC68EC040 also apply to the MC68EC040V. Refer to **Appendix C MC68040V and MC68EC040V** for more information on the MC68EC040V.

The MC68EC040 is Motorola's third generation of M68000-compatible, high-performance, 32-bit microprocessors. The MC68EC040 is an embedded controller employing a highly integrated architecture to provide very high performance in a monolithic HCMOS device. The MC68EC040 integrates an MC68040-compatible integer unit, an access control unit (ACU), and independent 4-Kbyte instruction and data caches. A six-stage instruction pipeline, multiple internal buses, and a full internal Harvard architecture, including separate caches for both instruction and data accesses, provides a high degree of instruction execution parallelism. The inclusion of on-chip bus snooping logic, which directly supports cache coherency in multimaster applications, enhances cache functionality.

The MC68EC040 is user-object-code compatible with previous members of the M68000 family and is specifically optimized to reduce the execution time of compiler-generated code. The MC68EC040 is pin compatible with the MC68040 and MC68LC040. The MC68EC040 is implemented in Motorola's latest HCMOS technology, providing an ideal balance between speed, power, and physical device size. Figure B-1 provides a simplified block diagram of the MC68EC040.

The main features of the MC68EC040 include:

- MC68040-Compatible Integer Execution Unit
- 4-Kbyte Instruction Cache and 4-Kbyte Data Cache Accessible Simultaneously
- 32-Bit, Nonmultiplexed External Address and Data Buses with Synchronous Bursting Interface
- User-Object-Code Compatible with All M68000 Microprocessors
- Concurrent Integer Unit, ACU, and Bus Controller Operation Maximizes Throughput
- Low-Latency Bus Accesses for Reduced Cache-Miss Penalty
- Multimaster/Multiprocessor Support via Bus Snooping
- 4-Gbyte Direct Addressing Range

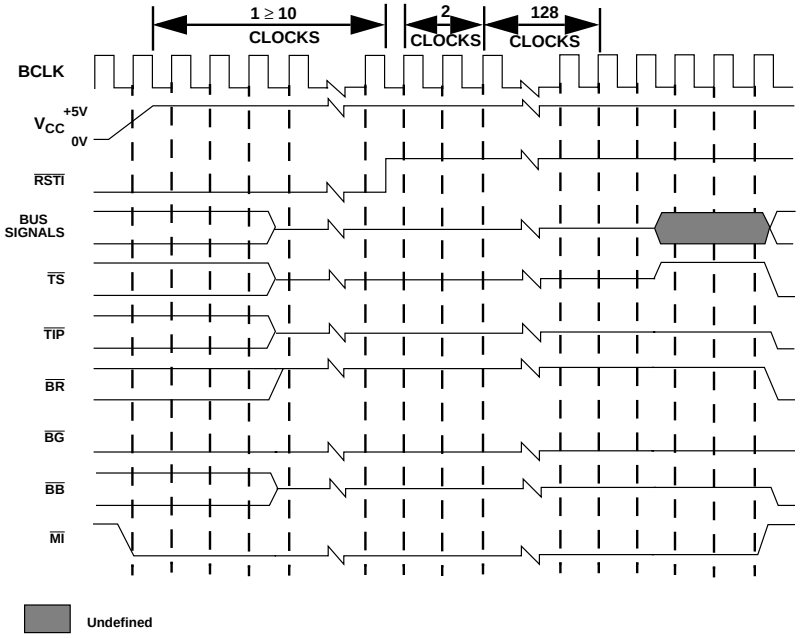


Figure B-5. MC68EC040 Initial Power-On Reset Timing

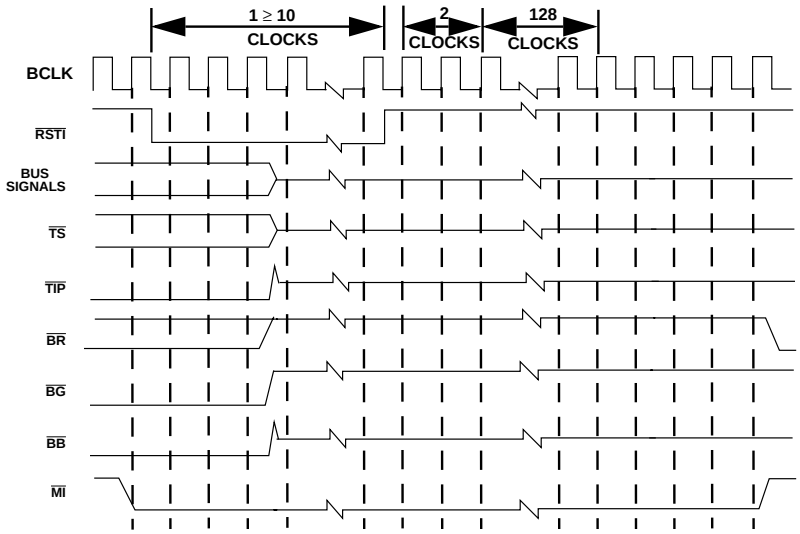


Figure B-6. MC68EC040 Normal Reset Timing

When a RESET instruction is executed, the processor drives the reset out ($\overline{\text{RSTO}}$) signal for 512 BCLK cycles. In this case, the processor resets the external devices of the system, and the internal registers of the processor are unaffected. The external devices connected to

B.7.3 DC Electrical Specifications (V_{CC} = 5.0 Vdc ±5 %)

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	2	V_{CC}	V
Input Low Voltage	V_{IL}	GND	0.8	V
Undershoot	—	—	0.8	V
Input Leakage Current @ 0.5–2.4 V $\overline{AVEC}$, BCLK, $\overline{BG}$, $\overline{CDIS}$, $\overline{IPLA}$, PCLK, $\overline{RSTI}$, SCx, $\overline{TBI}$, TLNx, $\overline{TCI}$, TCK, $\overline{TEA}$	I_{in}	20	20	mA
Hi-Z (Off-State) Leakage Current @ 0.5–2.4 V An, $\overline{BB}$, $\overline{CIOUT}$, Dn, $\overline{LOCK}$, $\overline{LOCKE}$, R/$\overline{W}$, SIZx, $\overline{TA}$, TDO, $\overline{TIP}$, TMx, TLNx, $\overline{TS}$, TTx, UPAx	I_{TSI}	20	20	mA
Signal Low Input Current, V_{IL} = 0.8 V TMS, TDI, $\overline{TRST}$	I_{IL}	–1.1	–0.18	mA
Signal High Input Current, V_{IH} = 2.0 V TMS, TDI, $\overline{TRST}$	I_{IH}	–0.94	–0.16	mA
Output High Voltage, I_{OH} = 5 mA	V_{OH}	2.4	—	V
Output Low Voltage, I_{OL} = 5 mA	V_{OL}	—	0.5	V
Capacitance*, V_{in} = 0 V, f = 1 MHz	C_{in}	—	25	pF

*Capacitance is periodically sampled rather than 100% tested.

B.7.4 Power Dissipation

Frequency	Watts
Maximum Values (V_{CC} = 5.25 V, T_A = 0°C)	
20 MHz	3.2
25 MHz	3.9
33 MHz	4.9
40 MHz	5.5
Typical Values (V_{CC} = 5 V, T_A = 25°C)*	
20 MHz	2.0
25 MHz	2.4
33 MHz	3.0
40 MHz	3.5

*This information is for system reliability purposes.

C.7.6 Output AC Timing Specifications (see Figures C-13* to C-21)

PRELIMIINARY								
		0–16.67 MHz		25 MHz		33 MHz		
Num	Characteristic	Min	Max	Min	Max	Min	Max	Unit
11	BCLK to Address, $\overline{\text{CIOUT}}$, $\overline{\text{LOCK}}$, $\overline{\text{LOCKE}}$, PSTx, R/$\overline{\text{W}}$, SIZx, TLNx, TMx, TTx, UPAx Valid	9	30	9	30	6.5	25	ns
12	BCLK to Output Invalid (Output Hold)	9	—	9	—	6.5	—	ns
13	BCLK to $\overline{\text{TS}}$ Valid	9	30	9	30	6.5	25	ns
14	BCLK to $\overline{\text{TIP}}$ Valid	9	30	9	30	6.5	25	ns
18	BCLK to Data-Out Valid	9	32	9	32	6.5	27	ns
19	BCLK to Data-Out Invalid (Output Hold)	9	—	9	—	6.5	—	ns
20	BCLK to Output Low Impedance	9	—	9	—	6.5	—	ns
21	BCLK to Data-Out High Impedance	9	20	9	20	6.5	17	ns
38	BCLK to Address, $\overline{\text{CIOUT}}$, $\overline{\text{LOCK}}$, $\overline{\text{LOCKE}}$, R/$\overline{\text{W}}$, SIZx, $\overline{\text{TS}}$, TLNx, TMx, TTx, UPAx High Impedance	9	18	9	18	6.5	15	ns
39	BCLK to $\overline{\text{BB}}$, $\overline{\text{TA}}$, $\overline{\text{TIP}}$ High Impedance	19	28	19	28	14	25	ns
40	BCLK to $\overline{\text{BR}}$, $\overline{\text{BB}}$ Valid	9	30	9	30	6.5	23	ns
43	BCLK to $\overline{\text{MI}}$ Valid	9	30	9	30	6.5	25	ns
48	BCLK to $\overline{\text{TA}}$ Valid	9	30	9	30	6.5	25	ns
50	BCLK to $\overline{\text{IPEND}}$, PSTx, $\overline{\text{RSTO}}$ Valid	9	30	9	30	6.5	25	ns
W	$\overline{\text{RSTI}}$ active to $\overline{\text{SCD}}$ inactive.	8	100	8	100	8	100	ns
A	$\overline{\text{IPLx}}$ to $\overline{\text{SCD}}$ invalid	8	100	8	100	8	100	ns

NOTE:

*Output timing is specified for a valid signal measured at the pin. Timing is specified driving an unterminated 30-Ω transmission line with a length characterized by a 2.5-ns one-way propagation delay. Buffer output impedance is typically 30 Ω; the buffer specifications include approximately 5 ns for the signal to propagate the length of the transmission line and back.