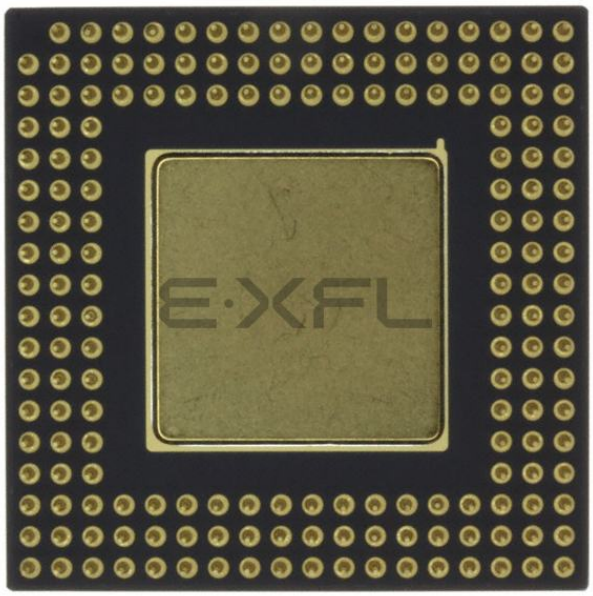


Welcome to E-XFL.COM

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

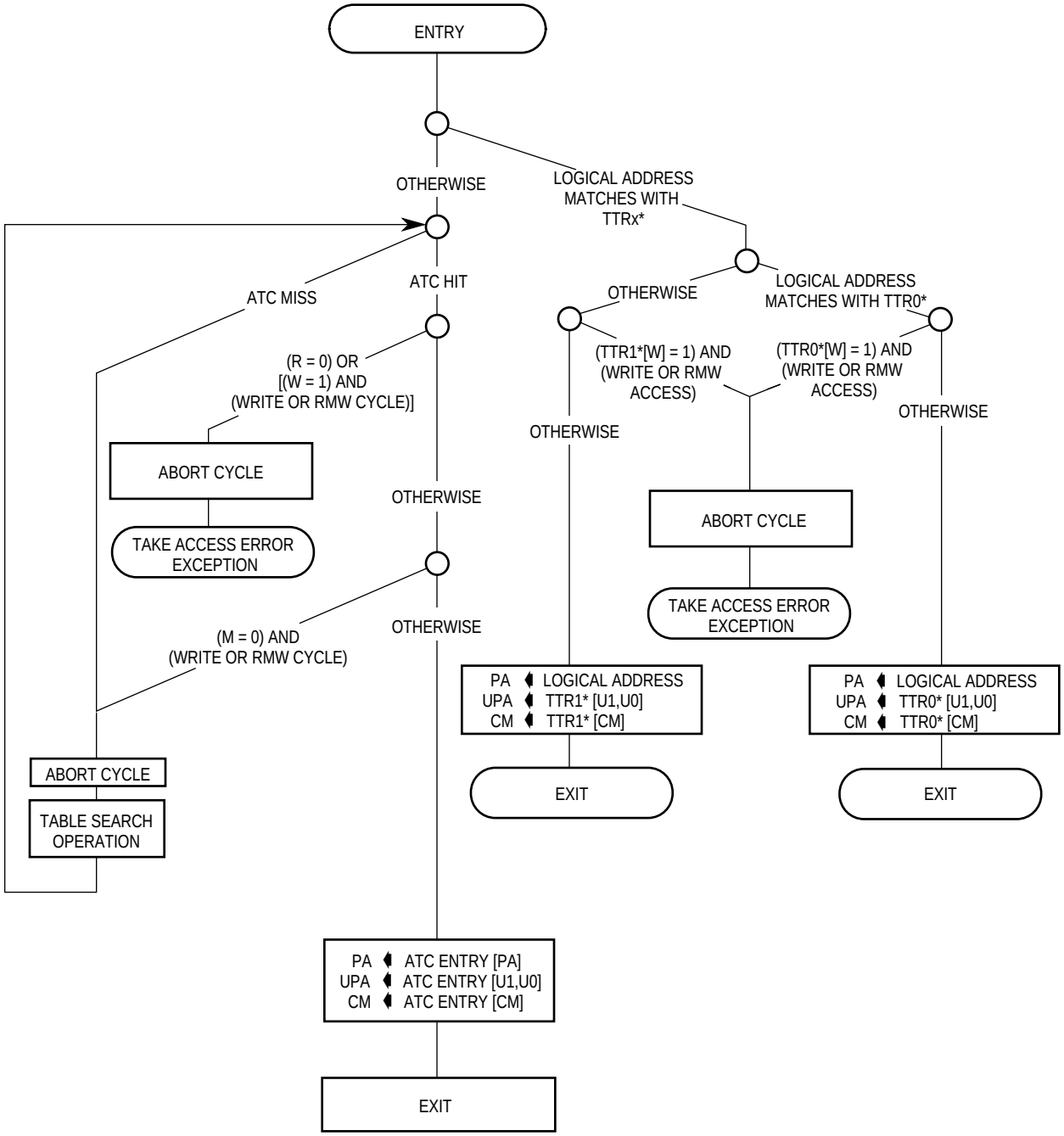
Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Obsolete
Core Processor	68040
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	20MHz
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-BEPGA
Supplier Device Package	179-PGA (47.24x47.24)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68lc040rc20a

TABLE OF CONTENTS (Continued)

Section 6
IEEE 1149.1 Test Access Port (JTAG)

LIST OF TABLES (Continued)



* Refers to either instruction or data transparent translation register.

Figure 3-22. Address Translation Flowchart

TAG	V	LW3	LW2	LW1	LW0

(a) Instruction Cache Line

TAG	V	LW3	D3	LW2	D2	LW1	D1	LW0	D0

TAG — 22-Bit Physical Address Tag
V — Line VALID Bit
LW — Long Word n (32-Bit) Data Entry
Dn — DIRTY Bit for Long Word n

(b) Data Cache Line

Figure 4-2. Cache Line Formats

The cache stores an entire line, providing validity on a line-by-line basis. Only burst mode accesses that successfully read four long words can be cached. Memory devices unable to support bursting can respond to a cache line read or write access by asserting the transfer burst inhibit (TBI) signal, forcing the processor to complete the access as a sequence of three long-word accesses. The cache recognizes burst accesses as if the access were never inhibited, detecting no difference.

A cache line is always in one of three states: invalid, valid, or dirty. For invalid lines, the V-bit is clear, causing the cache line to be ignored during lookups. Valid lines have their V-bit set and D-bits cleared, indicating all four long words in the line contain valid data consistent with memory. Dirty cache lines have the V-bit and one or more D-bits set, indicating that the line has valid long-word entries that have not been written to memory (long words whose D-bit is set). A cache line changes from valid to invalid if the execution of the CINV or CPUSH instruction explicitly invalidates the cache line; if a snooped write access hits the cache line and the line is not dirty; or if the SCx signals for a snooped read access invalidates the line. Both caches should be explicitly cleared after a hardware reset of the processor since reset does not invalidate the cache lines.

Figure 4-3 illustrates the general flow of a caching operation. The corresponding memory unit translates the logical address of each access to a physical address allowing the IU to access the data in the cache. To minimize latency of the requested data, the lower untranslated bits of the logical address map directly to the physical address bits and are used to access a set of cache lines in parallel with the translation. Physical address bits 9–4 are used to index into the cache and select one of the 64 sets of four cache lines. The four tags from the selected cache set are compared with the translated physical address bits 31–12 and bits 11 and 10 of the untranslated page offset. If any one of the four tags matches and the tag status is either valid or dirty, then the cache has a hit. During read accesses, a half-line (two long words) is accessed at a time, requiring two cache accesses for reads that are greater than a half-line or two long words. Write accesses within a cache line require a single cache access. If a misaligned access crosses two pages, then the partial access to the first page always happens twice, even if the pages are serialized. Consequently, if the accesses span page boundaries, misaligned accesses to peripherals are not possible unless the peripheral can tolerate double reads or writes.

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.

Table 5-1. Signal Index (Continued)

Signal Name	Mnemonic	Function
Processor Clock	PCLK[4]	Clock input used for internal logic timing. The PCLK frequency is exactly 2 × the BCLK frequency.
Test Clock	TCK	Clock signal for the IEEE P1149.1 Test Access Port (TAP).
Test Mode Select	TMS	Selects the principle operations of the test-support circuitry.
Test Data Input	TDI	Serial data input for the TAP.
Test Data Output	TDO	Serial data output for the TAP.
Test Reset	TRST[4]	Provides an asynchronous reset of the TAP controller.
Power Supply	V_{CC}	Power supply.
Ground	GND	Ground connection.

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

7.4.1 Byte, Word, and Long-Word Read Transfers

During a read transfer, the processor receives data from a memory or peripheral device. Since the data read for a byte, word, or long-word access is not placed in either of the internal caches by definition, the processor ignores the level on the transfer cache inhibit ($\overline{TCI}$) signal when latching the data. The bus controller performs byte, word, and long-word read transfers for the following cases:

- Accesses to a disabled cache.
- Accesses to a memory page that is specified noncachable.
- Accesses that are implicitly noncachable (read-modify-write accesses and accesses to an alternate logical address space via the MOVES instruction).
- Accesses that do not allocate in the data cache on a read miss (table searches, exception vector fetches, and exception stack deallocation for an RTE instruction).
- The first transfer of a line read is terminated with transfer burst inhibit ($\overline{TBI}$), forcing completion of the line access using three additional long-word read transfers.

Figure 7-8 is a flowchart for byte, word, and long-word read transfers. Bus operations are similar for each case and vary only with the size indicated and the portion of the data bus used for the transfer. Figure 7-9 is a functional timing diagram for byte, word, and long-word read transfers.

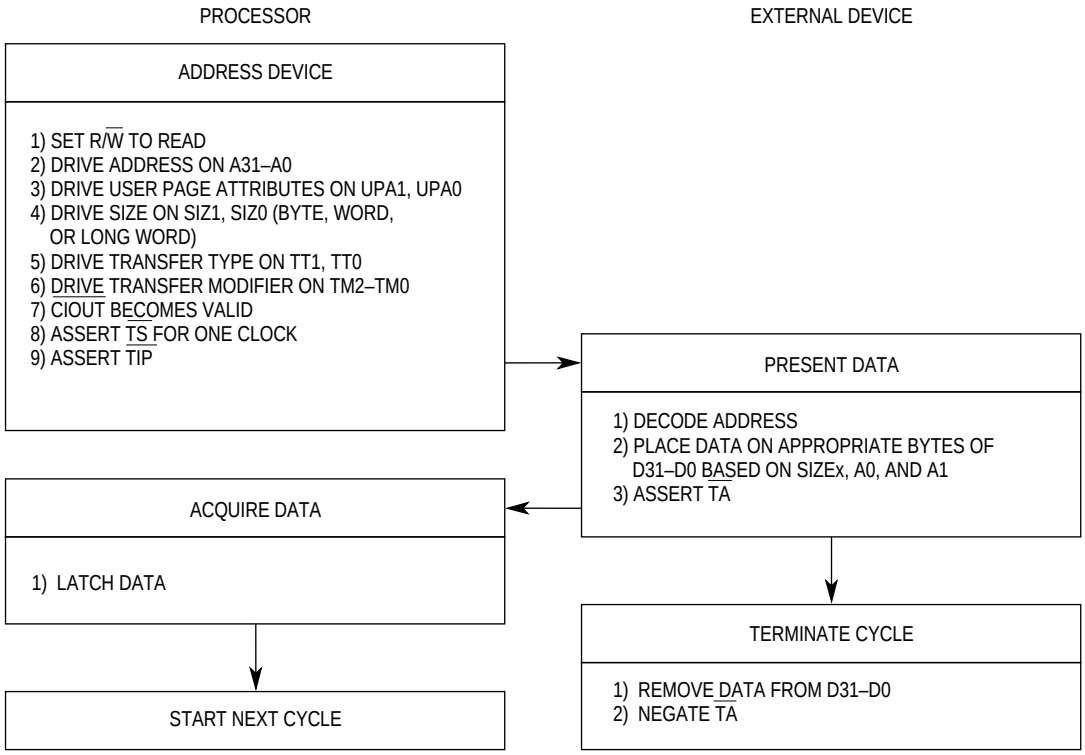


Figure 7-8. Byte, Word, and Long-Word Read Transfer Flowchart

Table 8-1. Exception Vector Assignments

Vector Number(s)	Vector Offset (Hex)	Assignment
0 1 2 3	000 004 008 00C	Reset Initial Interrupt Stack Pointer Reset Initial Program Counter Access Fault Address Error
4 5 6 7	010 014 018 01C	Illegal Instruction Integer Divide by Zero CHK, CHK2 Instruction FTRAPcc, TRAPcc, TRAPV Instructions
8 9 10 11	020 024 028 02C	Privilege Violation Trace Line 1010 Emulator (Unimplemented A-Line Opcode) Line 1111 Emulator (Unimplemented F-Line Opcode)
12 13 14 15	030 034 038 03C	(Unassigned, Reserved) Defined for MC68020 and MC68030, not used by M68040 Format Error Uninitialized Interrupt
16–23	040–05C	(Unassigned, Reserved)
24 25 26 27	060 064 068 06C	Spurious Interrupt Level 1 Interrupt Autovector Level 2 Interrupt Autovector Level 3 Interrupt Autovector
28 29 30 31	070 074 078 07C	Level 4 Interrupt Autovector Level 5 Interrupt Autovector Level 6 Interrupt Autovector Level 7 Interrupt Autovector
32–47	080–0BC	TRAP #0–15 Instruction Vectors
48–55	0C0–0DC	Floating-Point Exception Vectors (see Note)
56 57 58	0E0 0E4 0E8	Defined for MC68030 and MC68851, not used by M68040 Defined for MC68851, not used by M68040 Defined for MC68851, not used by M68040
59–63	0EC–0FC	(Unassigned, Reserved)
64–255	100–3FC	User Defined Vectors (192)

NOTE: Refer to **Section 9 Floating-Point Unit (MC68040 Only)**.

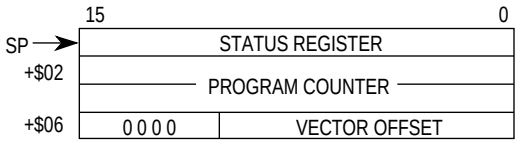
8.2 INTEGER UNIT EXCEPTIONS

The following paragraphs describe the external interrupt exceptions and the different types of exceptions generated internally by the M68040 integer unit. The following exceptions are discussed:

- Access Fault
- Address Error
- Instruction Trap
- Illegal and Unimplemented Instructions
- Privilege Violation

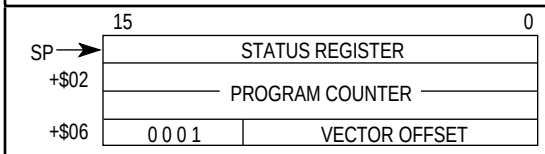
8.4.1 Four-Word Stack Frame (Format $0)

If a four-word stack frame is on the active stack and an RTE instruction is encountered, the processor updates the SR and PC with the data read from the stack, increments the stack pointer by eight, and resumes normal instruction execution.

Stack Frames	Exception Types	Stacked PC Points To
 FOUR-WORD STACK FRAME–FORMAT $0	• Interrupt • Format Error • TRAP #N • Illegal Instruction • A-Line Instruction • F-Line Instruction • Privilege Violation • Floating-Point Pre-Instruction	• Next Instruction • RTE or RESTORE Instruction • Next Instruction • Illegal Instruction • A-Line Instruction • F-Line Instruction • First Word of Instruction Causing Privilege Violation • Floating-Point Pre-Instruction Exception

8.4.2 Four-Word Throwaway Stack Frame (Format $1)

If a four-word throwaway stack frame is on the active stack and an RTE instruction is encountered, the processor increments the active stack pointer by eight, updates the SR with the value read from the stack, and then begins RTE processing again, as illustrated in Figure 8-6. The processor reads a new format word from the stack frame on top of the active stack (which may or may not be the same stack used for the previous operation) and performs the proper operations corresponding to that format. In most cases, the throwaway frame is on the interrupt stack, and when the SR value is read from the stack, the S-bit and M-bit are set. In that case, there is a normal four-word frame on the master stack. However, the second frame can be any format (even another throwaway frame) and can reside on any of the three system stacks.

Stack Frames	Exception Types	Stacked PC Points To
 THROWAWAY FOUR-WORD STACK FRAME–FORMAT $1	• Created on interrupt stack during interrupt exception processing when transition from master state to interrupt state occurs.	• Next Instruction: same as on master stack.

SECTION 9
FLOATING-POINT UNIT (MC68040 ONLY)

NOTE

This section does not apply to the MC68040V, MC68LC040, MC68EC040, or MC68EC040V. Refer to **Appendix A MC68LC040** and **Appendix B MC68EC040** for details.

Floating-point math refers to numeric calculations with a variable decimal point location. It is distinguished from integer math, which deals only with whole numbers and fixed decimal point locations. Historically, general-purpose microprocessors have had to depend on add-on coprocessors and accelerators such as the MC68881/MC68882 for fast floating-point capabilities. The MC68040 features a built-in floating-point unit (FPU). Consolidating this important function on chip speeds up the overall processing and eliminates some interfacing overhead required for external accelerators. The MC68040 FPU operates in parallel with the integer unit (IU). The FPU does the numeric calculation while the IU moves on to other tasks. Like the IU, the FPU has its own three-stage pipeline overlapping operations such as integer to floating-point conversion, instruction execution, and write-back. When used with the M68040FPSP, the MC68040 FPU is fully compliant with IEEE floating-point standards.

9.1 FLOATING-POINT UNIT PIPELINE

Integer data from memory (memory to register) requires a pass through the FPU pipeline, converting the data to the extended-precision format for the FPU to use. The result of this conversion is presented to the conversion stage of the FPU pipeline where the desired operation begins, starting a second pass through the pipeline. The IU is then released to execute other instructions once the data has been transferred to the FPU.

Floating-point data to memory (register to memory) requires a complete pass through the FPU pipeline, converting the data from the extended-precision format to an integer data format. Register-to-memory instructions are normally handled entirely by the conversion stage of the pipeline where the data move to memory operation completes. The IU is not released until it has received the converted data (during the last conversion unit cycle).

Like the IU, the FPU has been optimized for the most frequently used instructions and data types to provide the highest possible performance. To boost performance further, the FMOVE instruction concurrently executes with arithmetic calculations and executes completely transparent to the user. Instructions can execute nonsequentially as long as there are no register dependencies. Refer to **Section 10 Instruction Timings** for details on floating-point timings.

particularly careful of the lack of trichotomy in the floating-point branches since it is common for compilers to invert the sense of conditions.

When using the IEEE nonaware tests, the user receives a BSUN exception whenever a branch is attempted and the NAN condition code bit is set, unless the branch is an FBEQ or an FBNE. If the BSUN exception is enabled in the FPCR, the exception causes another exception. Therefore, the IEEE nonaware program is interrupted if an unexpected condition occurs. Compilers and programmers who are knowledgeable of the IEEE 754 standard should use the IEEE aware tests in programs that contain ordered and unordered conditions. Since the ordered or unordered attribute is explicitly included in the conditional test, the BSUN bit is not set in the FPSR EXC byte when the unordered condition occurs. Table 9-8 summarizes the conditional mnemonics, definitions, equations, predicates, and whether the BSUN bit is set in the FPSR EXC byte for the 32 floating-point conditional tests. The equation column lists the combination of FPCC bits for each test in the form of an equation.

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

FSAVE State Frame Field	Contents
INEX (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
INEX (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	Either 1 or 0, generally useless for INEX exceptions.
WBTM1, WBTM0, SBIT	Guard, round, and sticky of intermediate result's 67-bit mantissa.
E3	Always 1
T	Either 1 or 0
INEX (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

NOTE: If the M68040FPSP unimplemented exception handler is used, the above state frame information applies. The CMDREG1B or CMDREG3B fields of the state frame are modified as appropriate to encode the unimplemented instruction opcode. It is the user exception handler's responsibility to use the E3 and E1 field encodings to recognize which state frame information applies. When E3 = 1 and E1 = 1, E3 takes priority and the state frame information for E3 = 1 must be used.

10.5 MISCELLANEOUS INTEGER UNIT INSTRUCTION TIMINGS (Continued)

Instruction	Condition	<ea> Calculate	Execute
ORI #<xxx>,CCR	—	1	4
ORI #<xxx>,SR[a]	—	9	1_L + 8
PACK	Dx,Dy,#<xxx> –(Ay),–(Ax),#<xxx>	1 3	3 2_L + 3
PFLUSH[b]	—	11	1_L + 10
PFLUSHA[b]	—	11	1_L + 10
PFLUSHAN[b]	—	27	1_L + 26
PFLUSHN (An)[b]	—	11	1_L + 10
PTESTR, PTESTW[e]	—	25	11_L + 14
RESET[a]	—	521	521
RTD[c]	—	6	1_L + 5
RTE[a]	Stack Format $0 Stack Format $1 Stack Format $2 Stack Format $3 Stack Format $4 Stack Format $7	2 4 2 3 2 4	13 23 14 20 15 23
RTR[c]	—	7	1_L + 6
RTS[c]	—	5	5
SBCD	Dy,Dx –(Ay),–(Ax)	1 3	3 1_L + 3
SUBX	Dy,Dx –(Ay),–(Ax)	1 3	1 1_L + 2
SWAP	—	1	2
TRAP#[a]	—	16	16
TRAPcc[f]	Taken Not Taken	19 5	19 5
TRAPV[f]	Taken Not Taken	19 5	19 5
UNLK	—	2	1_L + 1
UNPK	Dx,Dy,# –(Ay),–(Ax),#	1 3	4 2_L + 4

NOTES:

a. Times listed are minimum. This instruction interlocks the <ea> calculate and execute stages and synchronizes some portions of the processor before execution.
b. Times listed are typical. This instruction interlocks the <ea> calculate and execute stages and synchronizes some portions of the processor before execution.
c. This instruction interlocks the <ea> calculate and execute stages.
d. Successive in-line MOVE16 instructions each add eight clocks to the <ea> calculate and execute times.
e. Typical measurement for three-level table search with no descriptor writes, no entries cached, and four-clock memory access times.
f. This instruction interlocks the <ea> calculate and execute stages. For the exception taken, this instruction also synchronizes some portions of the processor before execution; times listed are minimum in this case.

10.6 INTEGER UNIT INSTRUCTION TIMINGS (Continued)

Addressing Mode	BCHG, BCLR, BSET[a]		BFCHG, BFCLR, BFSET[b,c]		BFEXTS, BFEXTU[b,d]	
	<ea> Calculate	Execute	<ea> Calculate	Execute	<ea> Calculate	Execute
Dn	1	3/4	3/4[e]	6/7[e]	1/2[e]	4/5[e]
An	—	—	—	—	—	—
(An)	1	3/4	9	2_L + 8	9	2_L + 7
(An)+	1	3/4	—	—	—	—
–(An)	1	3/4	—	—	—	—
(d_{16},An)	2/1	1_L + 3/4	9	2_L + 8	9	2_L + 7
(d_{16},PC)	—	—	—	—	10	3_L + 7
(xxx).W, (xxx).L	2/1	1_L + 3/4	9	2_L + 8	9	2_L + 7
#<xxx>	—	—	—	—	—	—
(d_8,An,Xn)	3	5/6	10	11	10	10
(d_8,PC,Xn)	—	—	—	—	11	1_L + 10
(BR,Xn)	7	1_L + 8/1_L + 9	13	1_L + 13	13	1_L + 12
(bd,BR,Xn)	8	1_L + 9/1_L + 10	14	1_L + 14	14	1_L + 13
([bd,BR,Xn])	10	1_L + 11/1_L + 12	16	1_L + 16	16	1_L + 15
([bd,BR,Xn],od)	11	1_L + 12/1_L + 13	17	1_L + 17	17	1_L + 16
([bd,BR],Xn)	11	3_L + 10/3_L + 11	17	3_L + 15	17	3_L + 14
([bd,BR],Xn,od)	12	3_L + 11/3_L + 12	18	3_L + 16	18	3_L + 15

NOTES:
a. Bit instruction <ea> calculate and execute times T1/T2 apply to #<xxx>/Dn bit numbers.
b. This instruction interlocks the <ea> calculate and execute stages.
c. If the bit field spans a long-word boundary, add ten and nine clocks to the <ea> calculate and execute times, respectively. Two memory addresses are accessed in this case.
d. If the bit field spans a long-word boundary, add two clocks to the execute time. Two memory addresses are accessed in this case.
e. Immediate count specified for both width and offset and width and/or offset specified in register, respectively.

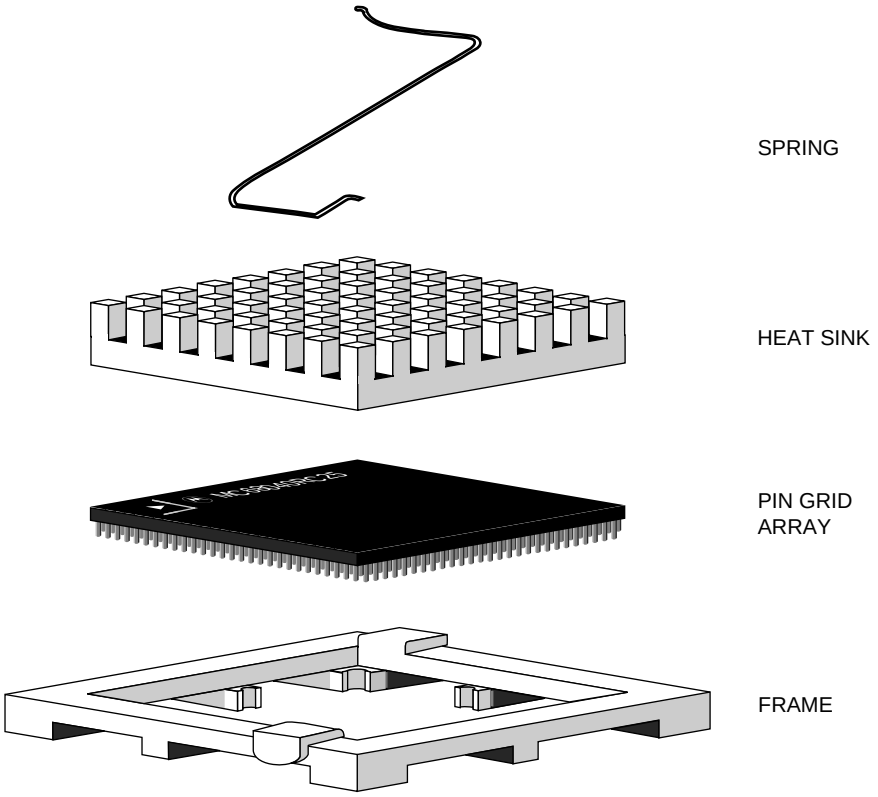
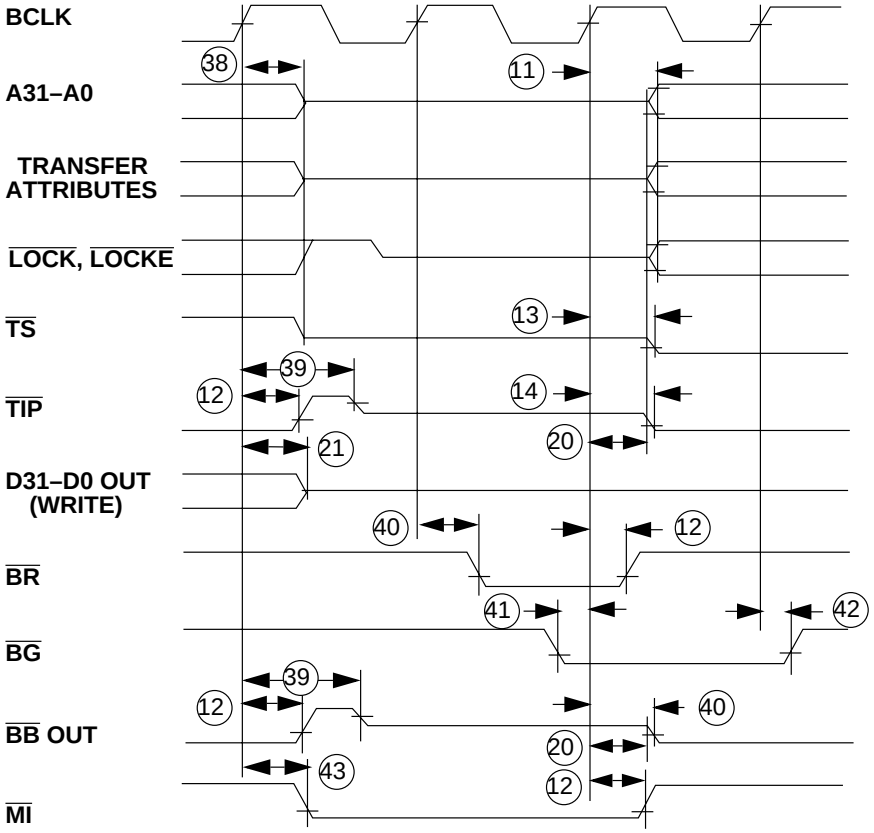


Figure 11-11. Heat Sink with Attachment

In the specification provided by Thermalloy, Inc., a chart illustrates the heat-sink temperature rise above ambient versus heat dissipated. This chart applies if no airflow is used with the heat-sink. Table 11-5 lists the calculations based on this chart.

Table 11-5. Thermal Parameters with Heat Sink and No Airflow

MHz	Thermal Mgmt. Technique / Airflow Velocity	Defined Parameters P_D	Defined Parameters T_J	Defined Parameters θ_{JC}	Heat-Sink Spec. $T_C–T_A$	Calculated T_C	Calculated T_A
MC68040							
25	0	6.3 W	110 °C	3 °C/W	64.4 °C	91.1 °C	26.7 °C
25	0	6.6 W	110 °C	3 °C/W	66.8 °C	90.2 °C	23.4 °C
25	0	8.6 W	110 °C	3 °C/W	82.8 °C	84.2 °C	1.4 °C
33	0	7.7 W	110 °C	3 °C/W	75.6 °C	86.9 °C	11.3 °C
33	0	8.0 W	110 °C	3 °C/W	78.0 °C	86.0 °C	8.0 °C
33	0	10.0 W	110 °C	3 °C/W	94.0 °C	80.0 °C	–14.0 °C
MC68LC040 and MC68EC040							
20	0	4.0 W	110 °C	3 °C/W	45.0 °C	98.0 °C	53.0 °C
25	0	5.0 W	110 °C	3 °C/W	54.0 °C	95.0 °C	41.0 °C
33	0	6.3 W	110 °C	3 °C/W	64.4 °C	91.1 °C	26.7 °C



NOTE: Transfer Attribute Signals = UPAx, SIZx, TTx, TMx, TLNx, R/$\overline{W}$, and $\overline{CIOUT}$

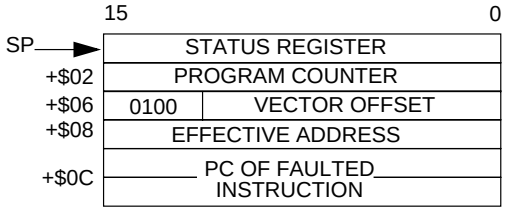
Figure A-6. Bus Arbitration Timing

direct addressing modes, this field is $0. The saved PC value is the logical address of the instruction that follows the unimplemented floating-point instruction. This value will be restored during RTE execution. The vector offset format number ($4) is used for this eight-word stack frame. Note that an MC68040 cannot correctly handle a stack format $4. The PC of the faulted instruction contains a long-word PC of the floating-point instruction that caused the trap to occur. The information is provided so that the instruction is available for software emulation of floating-point instructions. The processor generates exception vector number 11 for the unimplemented F-line instruction exception vector, fetches the address of the F-line exception handler from the exception vector table, and begins execution of the handler after prefetching instructions to fill the pipeline. Refer to **Section 8 Exception Processing** for details about exception processing.

B.5.2 MC68EC040 Stack Frames

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. The set of stack frames included for exception processing are four- and six-word stack frames, a four-word throwaway stack frame, an access error stack frame, and a new eight-word unimplemented floating-point stack frame. The stack frame that the MC68040 can generate and the MC68EC040 can process is the floating-point post-instruction stack frame. Refer to **Section 8 Exception Processing** for details about exception stack frames.

Eight-Word Stack Frame (Format $4)

Stack Frames	Exception Types	Stacked PC Points To
	• The MC68040 cannot generate or read this stack.	• Effective address field is the address of the faulted instruction operand.

When the MC68EC040 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 type of stack frame for any type of exception. The MC68EC040 does not generate the floating-point post-instruction stack frame. The MC68040 cannot accept the eight-word unimplemented stack frame. The MC68EC040 can handle all MC68040 stack frame formats.

B.6 SOFTWARE CONSIDERATIONS

The following MC68EC040 instructions are different from the MC68040: PTEST, PFLUSH, CPUSH, CINV, MOVEC, and all floating-point instructions.

APPENDIX C
MC68040V AND MC68EC040V

The MC68040V and MC68EC040V are Motorola's 3.3 volt, static versions of the MC68040 third-generation, M68000-compatible, high-performance, 32-bit microprocessor. They require a 3.3V power supply providing over 50 percent reduction in power consumption compared to a 5.0V device. The maximum power used at 3.3 volts is 1.5 watts at an operating frequency of 33 MHz. They also have a low-power stop mode. Once in this state, both devices remain quiescent, consuming less than 330 µW of power. The low-power usage of these microprocessors makes them an ideal choice for portable computing and power constrained applications.

The MC68040V programming model, data formats and types, instruction set, caches, and MMUs are the same as those described for the MC68LC040 in **Appendix A MC68LC040**. The MC68EC040V programming model, data formats and types, and instruction set are the same as those described for the MC68EC040 in **Appendix B MC68EC040**. However, both devices contain additional features:

- For the MC68040V, all differences that exist between the MC68LC040 and the MC68040, as described in **Appendix A MC68LC040**, also apply to the MC68040V. For the MC68EC040V, all differences that exist between the MC68EC040 and the MC68040, as described in **Appendix B MC68EC040**, also apply to the MC68EC040V.
- Both devices operate to 0 Hz and can accept 3.3V or 5V input.
- Both devices have a new processor status state, low-power stop mode, indicated when PST(3–0) = $6.
- There is no PCLK or $\overline{\text{TRST}}$ pin on either device.
- Both devices provide three new pins, system clock disable ($\overline{\text{SCD}}$), low frequency operation ($\overline{\text{LFO}}$), and loss of clock (LOC).

C.1 ADDITIONAL SIGNALS

Table C-1 lists the additional signals and Figure C-1 illustrates the functional signal groups of the MC68040V and MC68EC040V.

C.3 CLOCKING DURING NORMAL OPERATION

During normal operation of the processor, the BCLK should be driven with a 50% (±5%) duty cycle (refer to **C.7 MC68040V and MC68EC040V Electrical Characteristics** for details). The frequency of BCLK can not be changed during normal operation. Altering the BCLK frequency during normal operation (the $\overline{\text{LFO}}$ signal is negated) will result in unspecified operation. In the event that the BCLK input is lost, a processor reset is required. Once the loss of BCLK is detected during normal operation, the processor asserts LOC, indicating a loss of clock. External logic can then reset the processor to resume normal operation.

C.4 RESET OPERATION

An external device asserts the $\overline{\text{RSTI}}$ to reset the processor. When power is applied to the system, external circuitry should assert $\overline{\text{RSTI}}$ for a minimum of 10 BCLK cycles after V_{CC} is within tolerance. Figure C-2 is a functional timing diagram of the power-on reset operation, illustrating the relationships among V_{CC}, $\overline{\text{RSTI}}$, and bus signals. The BCLK signal is required to be stable by the time $\overline{\text{RSTI}}$ is negated. The V_{IH} levels of any pin must not exceed V_{CC} + 2.5V. $\overline{\text{RSTI}}$ is internally synchronized for two BCLKs before being used and must meet the specified setup and hold times to BCLK (specifications #51 and #52 in **C.7 MC68040V and MC68EC040V Electrical Characteristics**) only if recognition by a specific BCLK rising edge is required. $\overline{\text{MI}}$ is asserted while the MC68040V is in reset.

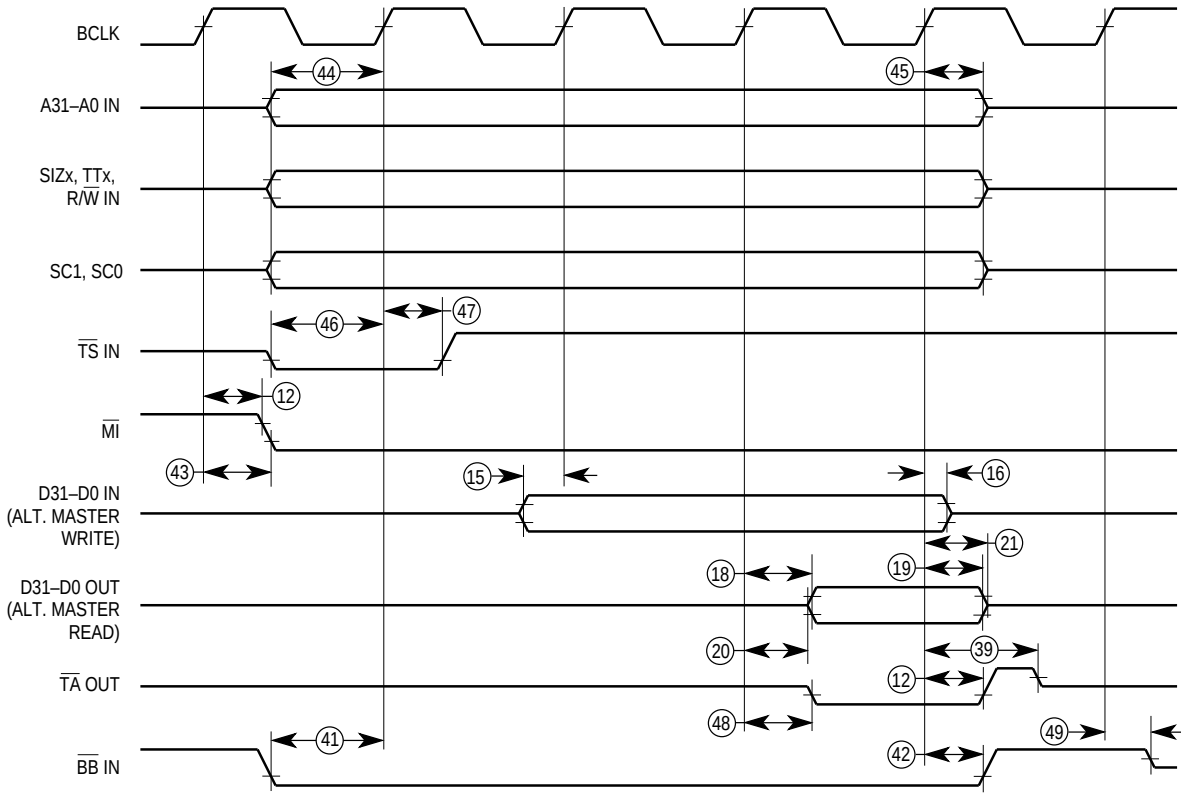


Figure C-15. Snoop Hit Timing