
**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 | 25MHz |
| Co-Processors/DSP | - |
| RAM Controllers | - |
| Graphics Acceleration | No |
| Display & Interface Controllers | - |
| Ethernet | - |
| SATA | - |
| USB | - |
| Voltage - I/O | 5V |
| 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=mc68040fe25v |

# TABLE OF CONTENTS (Continued)



**Section 8**
**Exception Processing**

# TABLE OF CONTENTS (Continued)

## 1.9 NOTATIONAL CONVENTIONS

Table 1-3 lists the notation conventions used throughout this manual unless otherwise specified.

**Table 1-3. Notational Conventions**

| | |
|---|---|
| **Single- And Double-Operand Operations** | |
| + | Arithmetic addition or postincrement indicator. |
| – | Arithmetic subtraction or predecrement indicator. |
| × | Arithmetic multiplication. |
| ÷ | Arithmetic division or conjunction symbol. |
| ~ | Invert; operand is logically complemented. |
| Λ | Logical AND |
| V | Logical OR |
| ⊕ | Logical exclusive OR |
| ø | Source operand is moved to destination operand. |
| łø | Two operands are exchanged. |
| <op> | Any double-operand operation. |
| <operand>tested | Operand is compared to zero and the condition codes are set appropriately. |
| sign-extended | All bits of the upper portion are made equal to the high-order bit of the lower portion. |
| **Other Operations** | |
| TRAP | Equivalent to Format ÷ Offset Word ø (SSP); SSP – 2 ø SSP; PC ø (SSP); SSP – 4 ø SSP; SR ø (SSP); SSP – 2 ø SSP; (Vector) ø PC |
| STOP | Enter the stopped state, waiting for interrupts. |
| <operand>$_{10}$ | The operand is BCD; operations are performed in decimal. |
| If <condition><br>then <operations><br>else <operations> | Test the condition. If true, the operations after “then” are performed. If the condition is false and the optional “else” clause is present, the operations after “else” are performed. If the condition is false and else is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example. |
| **Register Specification** | |
| An | Any Address Register n (example: A3 is address register 3) |
| Ax, Ay | Source and destination address registers, respectively. |
| BR | Base Register—An, PC, or suppressed. |
| Dc | Data register D7–D0, used during compare. |
| Dh, Dl | Data registers high- or low-order 32 bits of product. |
| Dn | Any Data Register n (example: D5 is data register 5) |
| Dr, Dq | Data register’s remainder or quotient of divide. |
| Du | Data register D7–D0, used during update. |
| Dx, Dy | Source and destination data registers, respectively. |
| MRn | Any Memory Register n. |
| Rn | Any Address or Data Register |
| Rx, Ry | Any source and destination registers, respectively. |
| Xn | Index Register—An, Dn, or suppressed. |

Descriptor Address

This 30-bit field, which contains the physical address of a page descriptor, is only used in indirect descriptors.

G—Global

When this bit is set, it indicates the entry is global. PFLUSH instruction variants that specify nonglobal entries do not invalidate global entries, even when all other selection criteria are satisfied. If these PFLUSH variants are not used, then system software can use this bit.

M—Modified

This bit identifies a modified page. The M68040 sets the M-bit in the corresponding page descriptor before a write operation to a page for which the M-bit is clear, except for write-protect or supervisor violations. The read portion of a read-modify-write access is considered a write for updating purposes. The M68040 never clears this bit.

PDT—Page Descriptor Type

This field identifies the descriptor as an invalid descriptor, a page descriptor for a resident page, or an indirect pointer to another page descriptor.

00 = Invalid
This code indicates that the descriptor is invalid. An invalid descriptor can represent a nonresident page or a logical address range that is out of bounds. All other bits in the descriptor are ignored. When an invalid descriptor is encountered, an ATC entry is created for the logical address with the resident bit in the MMUSR clear.

01 or 11 = Resident
These codes indicate that the page is resident.

10 = Indirect
This code indicates that the descriptor is an indirect descriptor. Bits 31–2 contain the physical address of the page descriptor. This encoding is invalid for a page descriptor pointed to by an indirect descriptor.

Physical Address

This 20-bit field contains the physical base address of a page in memory. The logical address supplies the low-order bits of the address required to index into the page. When the page size is 8-Kbyte, the least significant bit of this field is not used.

S—Supervisor Protected

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

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

Physical Address

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

R—Resident

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

S—Supervisor Protected

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

U0, U1—User Page Attributes

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

V—Valid

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

W—Write Protected

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

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

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

# SECTION 4
# INSTRUCTION AND DATA CACHES

**NOTE**

Ignore all references to the memory management unit (MMU) when reading for the MC68EC040 and MC68EC040V. The functionality of the MC68040 transparent translation registers has been changed in the MC68EC040 and MC68EC040V to the access control registers. Refer to **Appendix B MC68EC040** for details.

The M68040 contains two independent, 4-Kbyte, on-chip caches located in the physical address space. Accessing instruction words and data simultaneously through separate caches increases instruction throughput. The M68040 caches improve system performance by providing cached data to the on-chip execution unit with very low latency. Systems with an alternate bus master receive increased bus availability.

Figure 4-1 illustrates the instruction and data caches contained in the instruction and data memory units. The appropriate memory unit independently services instruction prefetch and data requests from the integer unit (IU). The memory units translate the logical address in parallel with indexing into the cache. If the translated address matches one of the cache entries, the access hits in the cache. For a read operation, the memory unit supplies the data to the IU, and for a write operation, the memory unit updates the cache. If the access does not match one of the cache entries (misses in the cache) or a write access must be written through to memory, the memory unit sends an external bus request to the bus controller. The bus controller then reads or writes the required data.

Cache coherency in the M68040 is optimized for multimaster applications in which the M68040 is the caching master sharing memory with one or more noncaching masters (such as DMA controllers). The M68040 implements a bus snooper that maintains cache coherency by monitoring an alternate bus master's access and performing cache maintenance operations as requested by the alternate bus master. Matching cache entries can be invalidated during the alternate bus master's access to memory, or memory can be inhibited to allow the M68040 to respond to the access as a slave. For an external write operation, the processor can intervene in the access and update its internal caches (sink data). For an external read operation, the processor supplies cached data to the alternate bus muster (source data). This prevents the M68040 caches from accumulating old or invalid copies of data (stale data). Alternate bus masters are allowed access to locally modified data within the caches that is no longer consistent with external memory (dirty data). Allowing memory pages to be specified as write-through instead of copyback also supports cache coherency. When a processor writes to write-through pages, external

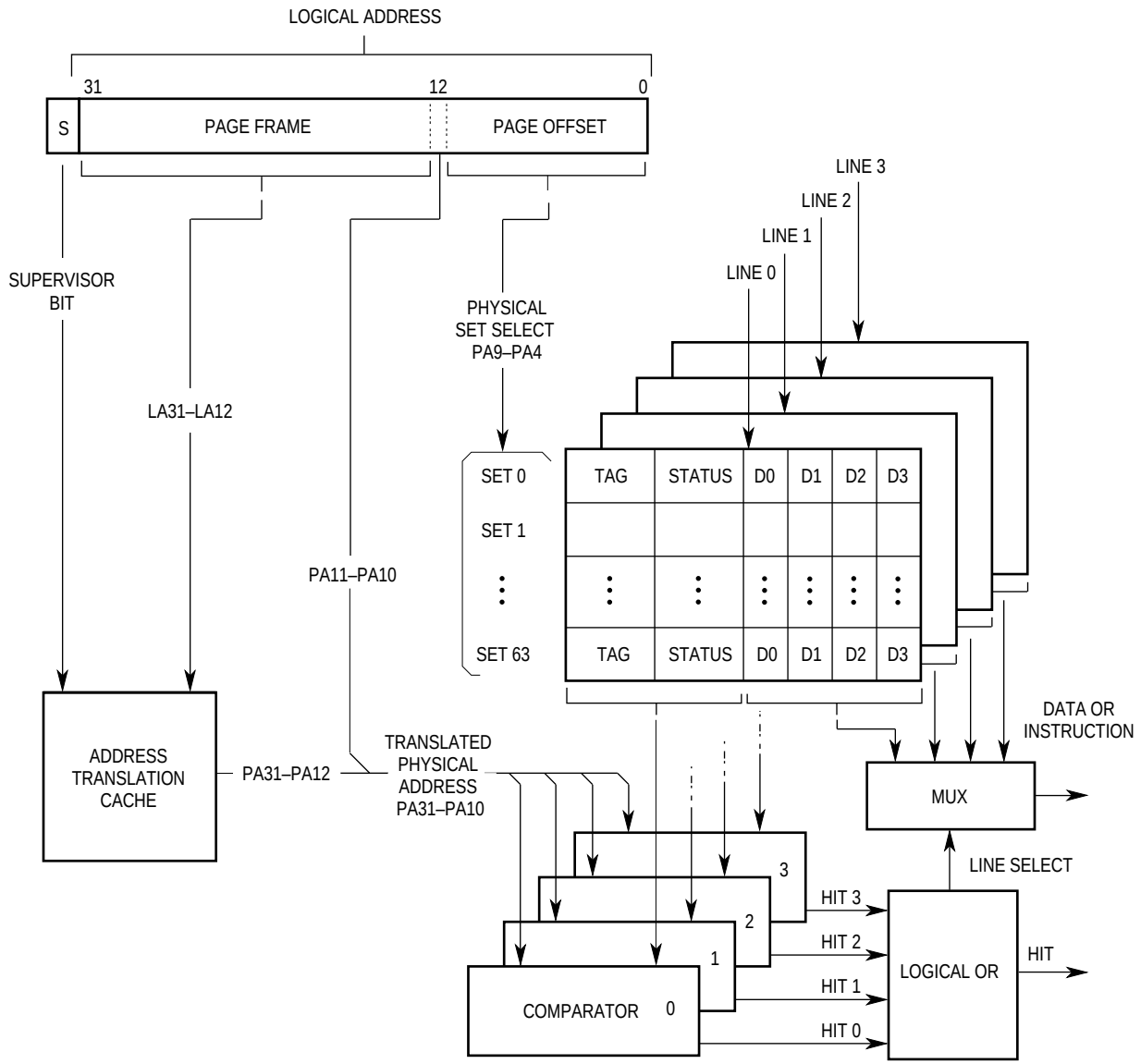



**Figure 4-3. Caching Operation**

Both caches contain circuitry to automatically determine which cache line in a set to use for a new line. The cache controller locates the first invalid line and uses it; if no invalid lines exist, then a pseudo-random replacement algorithm is used to select a valid line, replacing it with the new line. Each cache contains a 2-bit counter, which is incremented for each access to the cache. The instruction cache counter is incremented for each half-line accessed in the instruction cache. The data cache counter is incremented for each half-line accessed during reads, for each full line accessed during writes in copyback mode, and for each bus transfer resulting from a write in write-through mode. When a miss occurs and all four lines in the set are valid, the line pointed to by the current counter value is replaced, after which the counter is incremented.

**MC68040 Floating-Point Emulation (MC68040FPSP)** for descriptions of emulator use of this signal.

### 5.7.2 Reset In (RSTI)

This input signal causes the M68040 to enter reset exception processing. The RSTI signal is an asynchronous input that is internally synchronized to the next rising edge of the BCLK signal. All three-state signals are set to the high-impedance state, and all outputs, except MI, are negated when RSTI is recognized. The assertion of RSTI does not affect the test pins. Refer to **Section 7 Bus Operation** for a description of reset operation and to **Section 8 Exception Processing** for information about the reset exception.

### 5.7.3 Reset Out (RSTO)

The M68040 asserts this output during execution of the RESET instruction to initialize external devices. Refer to **Section 7 Bus Operation** for a description of reset out bus operation.

## 5.8 INTERRUPT CONTROL SIGNALS

The following signals control the interrupt functions.

### 5.8.1 Interrupt Priority Level (IPL2–IPL0)

These input signals provide an indication of an interrupt condition and the encoding of the interrupt level from a peripheral or external prioritizing circuitry. IPL2 is the most significant bit of the level number. For example, since the IPL$\overline{\text{x}}$ signals are active low, IPL2–IPL0 = $5 corresponds to an interrupt request at interrupt priority level 2.

During a processor reset, the levels on the IPL$\overline{\text{x}}$ lines are latched and used to select the output driver characteristics for three signal groups listed in Table 5-5. Refer to **Section 8 Exception Processing** for information on interrupts and to **Section 11 MC68040 Electrical and Thermal Characteristics** for information on driver characteristics. Refer to **Appendix A MC68LC040** and **Appendix B MC68EC040** for how these signals are different on power-up.

**Table 5-5. Output Driver Control Groups**

| Signal | Output Buffers Controlled |
|---|---|
| IPL2 | Data-Bus: D31–D0 |
| IPL1 | Address Bus and Transfer Attributes:<br>A31–A0, CIOUT, LOCK, LOCKE , R/W, SIZ1–SIZ0,<br>TLN1–TLN0, TM2–TM0, TT1–TT0, UPA1–UPA0 |
| IPL0 | Miscellaneous Control Signals:<br>BB, BR , IPEND, MI, PST3–PST0, RSTO , TA, TDO, TIP, TS |

NOTE: High input level = small buffers enabled; low input level = large buffers enabled.

## 5.14 SIGNAL SUMMARY

Table 5-7 provides a summary of the electrical characteristics of the signals discussed in this section.

**Table 5-7. Signal Summary**

| Signal Name | Mnemonic | Type | Active | Three-State |
|---|---|---|---|---|
| Address Bus | A31–A0 | Input/Output | High | Yes |
| Autovector | AVEC | Input | Low | — |
| Bus Busy | BB | Input/Output | Low | Yes |
| Bus Clock | BCLK | Input | — | — |
| Bus Grant | BG | Input | Low | — |
| Bus Request | BR | Output | Low | No |
| Cache Disable | CDIS | Input | Low | — |
| Cache Inhibit Out | CIOUT | Output | Low | Yes |
| Data Bus | D31–D0 | Input/Output | High | Yes |
| Data Latch Enable[1] | DLE | Input | High | — |
| Ground | GND | Ground | — | — |
| Interrupt Pending | IPEND | Output | Low | No |
| Interrupt Priority Level[2] | IPL2–IPL0 | Input | Low | — |
| Bus Lock | LOCK | Output | Low | Yes |
| Bus Lock End | LOCKE | Output | Low | Yes |
| Memory Inhibit | MI | Output | Low | No |
| MMU Disable[3] | MDIS | Input | Low | — |
| Processor Clock | PCLK | Input | — | — |
| Processor Status | PST3–PST0 | Output | High | No |
| Read/Write | R/W | Input/Output | High/Low | Yes |
| Reset In | RSTI | Input | Low | — |
| Reset Out | RSTO | Output | Low | No |
| Snoop Control | SC1, SC0 | Input | High | — |
| Transfer Acknowledge | TA | Input/Output | Low | Yes |
| Transfer Burst Inhibit | TBI | Input | Low | — |
| Transfer Cache Inhibit | TCI | Input | Low | — |
| Transfer Error Acknowledge | TEA | Input | Low | — |
| Transfer in Progress | TIP | Output | Low | Yes |
| Transfer Line Number | TLN1, TLN0 | Output | High | Yes |
| Transfer Modifier | TM2–TM0 | Output | High | Yes |
| Transfer Size | SIZ1, SIZ0 | Input/Output | High | Yes |

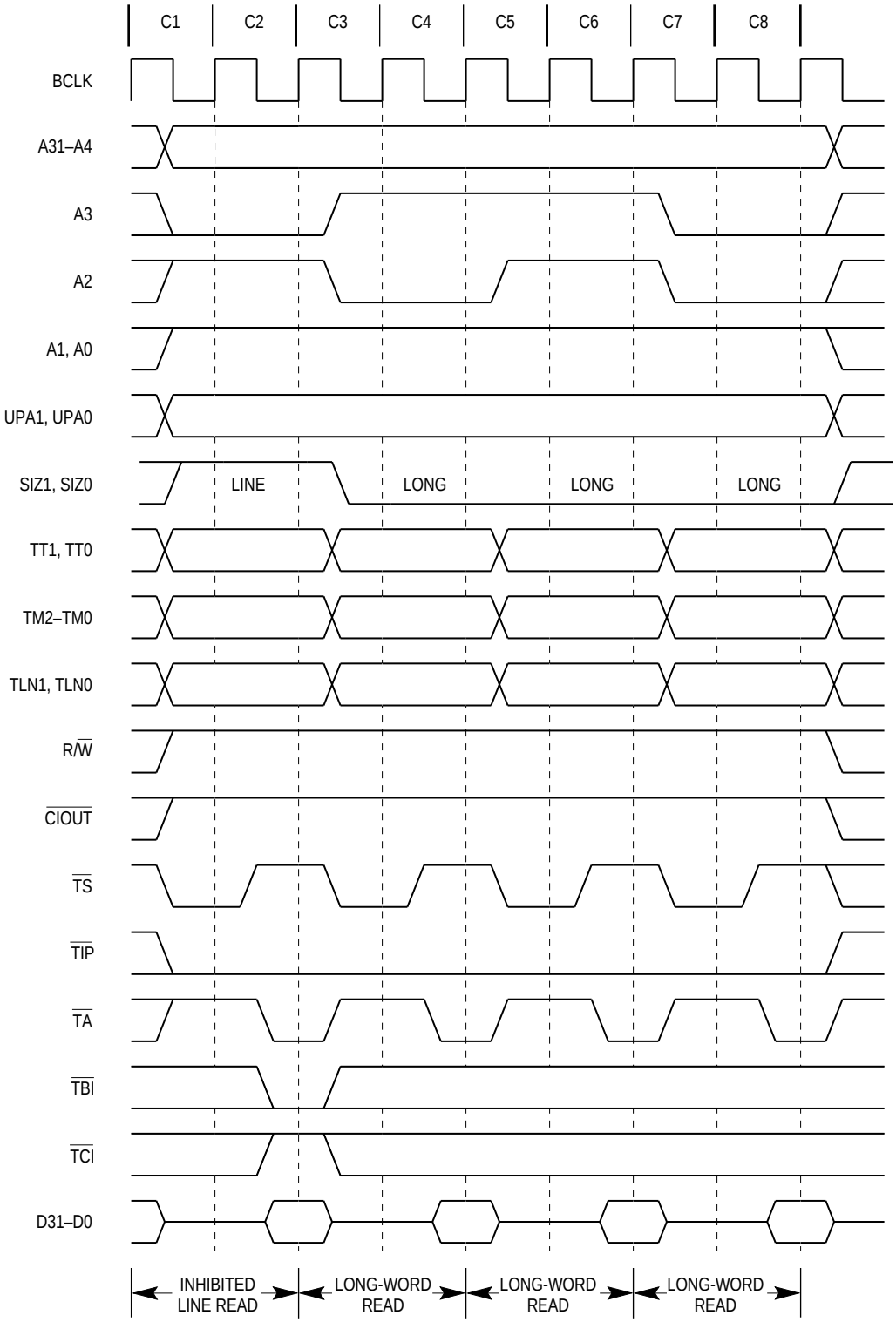



**Figure 7-13. Burst-Inhibited Line Read Transfer Timing**

For processor resets after the initial power-on reset, $\overline{RSTI}$ should be asserted for at least 10 clock periods. Figure 7-45 illustrates timings associated with a reset when the processor is executing bus cycles. Note that $\overline{BB}$ and $\overline{TIP}$ (and $\overline{TA}$ if driven during a snooped access) are negated before transitioning to a three-state level.

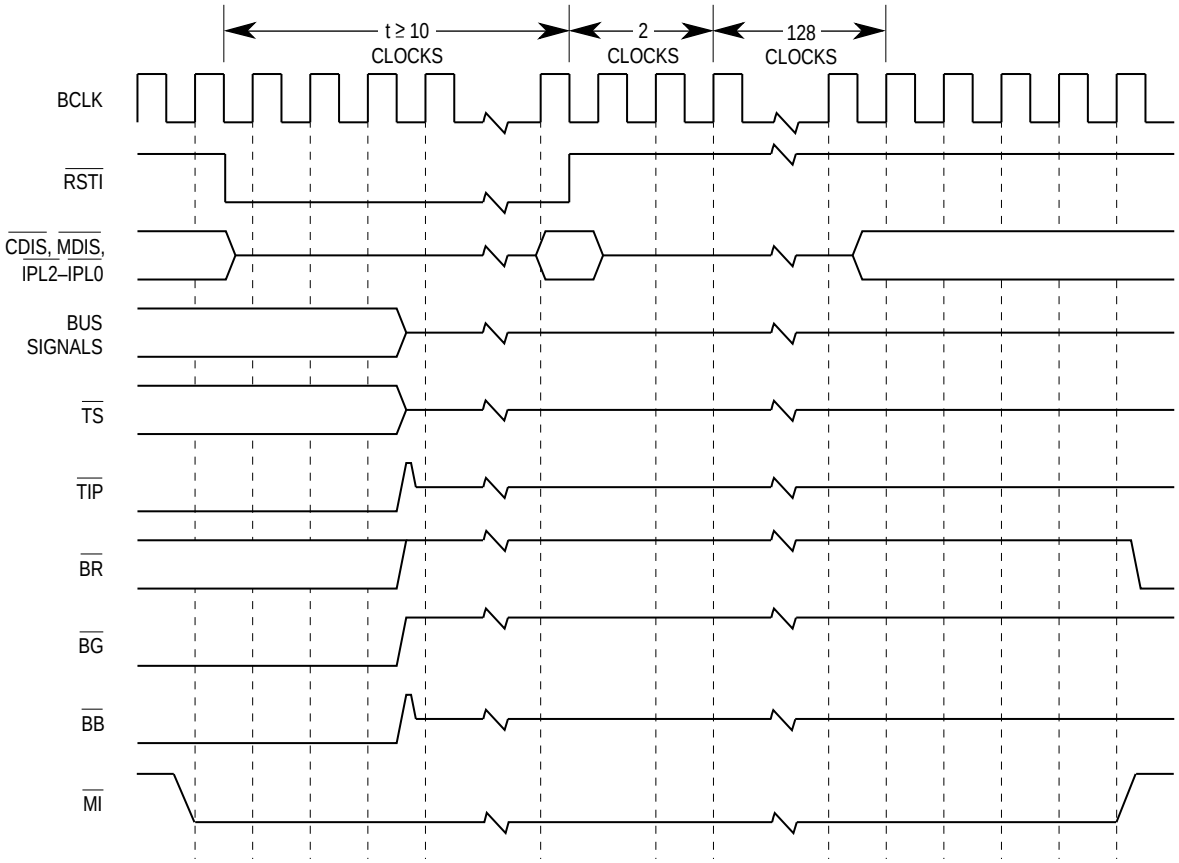



**Figure 7-45. Normal Reset Timing**

Resetting the processor causes any bus cycle in progress to terminate as if $\overline{TA}$ or $\overline{TEA}$ had been asserted. In addition, the processor initializes registers appropriately for a reset exception. **Section 8 Exception Processing** describes exception processing. When a RESET instruction is executed, the processor drives the reset out ($\overline{RSTO}$) 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 the $\overline{RSTO}$ signal 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 the $\overline{RSTO}$ signal to negate. $\overline{RSTO}$ can be logically ANDed with the external signal driving $\overline{RSTI}$ to derive a system reset signal that is asserted for both an external processor reset and execution of a RESET instruction.

they can be used to read the FPIAR in an exception handler without changing the previous value. A reset or a restore operation of the null state clears the FPIAR.

## 9.3 FLOATING-POINT DATA FORMATS AND DATA TYPES

The M68000 floating-point model (MC68881, MC68882, MC68040) supports the following data formats: single precision, double precision, extended precision, and packed decimal. The M68000 floating-point model supports the following data types: normalized, zeros, infinities, denormalized numbers, and NANs. The MC68040 supports part of the M68000 floating-point model in hardware. Table 9-2 lists the data formats and data types supported by the MC68040. Tables 9-3 through 9-6 summarize the floating-point data formats and data types details. For further information on the data formats and data types, refer to the M68000UM/AD, *M68000 Family Programmer's Reference Manual*.

**Table 9-2. MC68040 FPU Data Formats and Data Types**

| Number Types | Data Formats | | | | | | |
|---|---|---|---|---|---|---|---|
| | Single-Precision Real | Double-Precision Real | Extended-Precision Real | Packed-Decimal Real | Byte Integer | Word Integer | Long-Word Integer |
| Normalized | * | * | * | † | * | * | * |
| Zero | * | * | * | † | * | * | * |
| Infinity | * | * | * | † | | | |
| NAN | * | * | * | † | | | |
| Denormalized | † | † | † | † | | | |
| Unnormalized | | | † | † | | | |

*Data Format/Type Supported by On-Chip MC68040 FPU Hardware
†Data Format/Type Supported by Software (MC68040FPSP)

## 9.6 FLOATING-POINT EXCEPTIONS

There are two classes of floating-point-related exceptions: nonarithmetic floating-point exceptions and arithmetic floating-point exceptions. The latter relates to the handling of arithmetic exceptions caused by floating-point activity, and the former includes unimplemented floating-point instructions and unsupported data types not related to the handling of arithmetic exceptions. Format error and FTRAPcc exceptions may seem to be floating-point related, but are considered IU exceptions (see **Section 8 Exception Processing**). The following sections detail floating-point exceptions and how the MC68040 and M68040FPSP handle them. Table 9-9 lists the vector numbers related to floating-point exceptions.

**Table 9-9. Floating-Point Exception Vectors**

| Vector Number | Vector Offset (Hex) | Assignment |
|---|---|---|
| 11 | 02C | Floating-Point Unimplemented Instruction (also used for F-line instruction) |
| 48 | 0C0 | Floating-Point Branch or Set on Unordered Condition |
| 49 | 0C4 | Floating-Point Inexact Result |
| 50 | 0C8 | Floating-Point Divide by Zero |
| 51 | 0CC | Floating-Point Underflow |
| 52 | 0D0 | Floating-Point Operand Error |
| 53 | 0D4 | Floating-Point Overflow |
| 54 | 0D8 | Floating-Point SNAN |
| 55 | 0DC | Floating-Point Unimplemented Data Type |

The following paragraphs detail nonarithmetic floating-point exceptions.

### 9.6.1 Unimplemented Floating-Point Instructions

F-line instructions are instruction word patterns with bits 15–12 that have an $F encoding, causing F-line exceptions. These instructions are termed unimplemented floating-point instructions and cause an unimplemented floating-point exception. The MC68040 recognizes some F-line instructions, such as the FMUL and CPUSH, which do not cause F-line exceptions. There are some F-line instructions that the MC68040 recognizes as valid MC68881/MC68882 floating-point instruction patterns, but as floating-point instructions that the processor cannot complete in hardware. Table 9-10 lists the floating-point instructions that are unimplemented and therefore cause an unimplemented instruction exception.

If the processor encounters an F-line instruction and the instruction patterns do not match either of the above two cases, the processor takes an F-line illegal exception. F-line illegal exceptions are discussed further in **Section 8 Exception Processing**. The processor generates an exception with vector number 11 and pushes a four-word stack frame format $0 on the system stack. An illegal instruction exception is also reported when a breakpoint acknowledge bus cycle is run and terminated with either a transfer acknowledge ($\overline{TA}$) or transfer error acknowledge ($\overline{TEA}$) signal. Since the unimplemented floating-point

**Table 9-12. Overflow Rounding Mode Values**

| Rounding Mode | Result |
|---|---|
| RN | Infinity, with the sign of the intermediate result. |
| RZ | Largest magnitude number, with the sign of the intermediate result. |
| RM | For positive overflow, largest positive number; for negative overflow, infinity. |
| RP | For positive overflow, infinity; for negative overflow, largest negative number. |

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

b. If the user OVFL exception handler is enabled, the M68040FPSP OVFL restores the FPU to its exceptional state, cleans up the stack to the conditions prior to execution, and continues instruction execution at the user OVFL exception handler. No parameters are passed to the user OVFL exception handler since the M68040FPSP OVFL exception handler provides the illusion that it never existed.

The user OVFL exception handler must execute an FSAVE as its first floating-point instruction. The destination contains the rounding mode values listed in Table 9-12, and the user OVFL exception handler can choose to modify these values. The E3 and E1 bits of the floating-point state frame are examined to determine which fields on the floating-point state frame are valid. E3 always takes precedence and must be serviced first. Table 9-16 lists the floating-point state frame fields for OVFL exceptions with E3 set or with E3 clear and E1 set. Note that it is possible for an FADD, FSUB, FMUL, and FDIV to report a post-instruction exception, although these instructions normally generate a pre-instruction exception. The following example illustrates the reason why a post-instruction exception is generated.

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

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

The FMOVE OUT instruction generates a post-instruction exception. For this case, the effective address field in the format $3 stack frame points to the destination memory location. If the destination is an integer data register, the FPIAR points to the F-line word

## 10.1 OVERVIEW

Refer to **Section 2 Integer Unit** for information on the integer unit pipeline. The <ea> fetch timing is not listed in the following tables because most instructions require one clock in the <ea> fetch stage for each memory access to obtain an operand. An instruction requires one clock to pass through the <ea> fetch stage even if no operand is fetched. Table 10-2 summarizes the number of memory fetches required to access an operand using each addressing mode for long-word aligned accesses. The user must perform his own calculations for <ea> fetch timing for misaligned accesses.

**Table 10-2. Number of Memory Accesses**

| Addressing Mode | Evaluate <ea> And Fetch Operand | Evaluate <ea> And Send To Execution Stage |
|---|---|---|
| Dn | 0 | 0 |
| An | 0 | 0 |
| (An) | 1 | 0 |
| (An)+ | 1 | 0 |
| –(An) | 1 | 0 |
| ($d_{16}$,An) | 1 | 0 |
| ($d_{16}$,PC) | 1 | 0 |
| (xxx).W, (xxx).L | 1 | 0 |
| #<xxx> | 0 | 0 |
| ($d_8$,An,Xn) | 1 | 0 |
| ($d_8$,PC,Xn) | 1 | 0 |
| (BR,Xn) | 1 | 0 |
| (bd,BR,Xn) | 1 | 0 |
| ([bd,BR,Xn]) | 2 | 1 |
| ([bd,BR,Xn],od) | 2 | 1 |
| ([bd,BR],Xn) | 2 | 1 |
| ([bd,BR],Xn,od) | 2 | 1 |

In the instruction timing tables, the <ea> calculate column lists the number of clocks required for the instruction to execute in the <ea> calculate stage of the integer unit pipeline. Dual effective address instructions such as ABCD –(Ay),–(Ax) require two calculations in the <ea> calculate stage and two memory fetches. Due to pipelining, the fetch of the first operand occurs in the same clock as the <ea> calculation for the second operand.

The execute column lists the number of clocks required for the instruction to execute in the execute stage of the integer unit pipeline. This number is presented as a lead time and a base time. The lead time is the number of clocks the instruction can stall when entering the execution stage without delaying the instruction execution. If the previous instruction is still executing in the execution stage when the current instruction is ready to move from the <ea> fetch stage, the current instruction stalls until the previous one completes. For

## 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><br>–(Ay),–(Ax),#<xxx> | 1<br>3 | 3<br>$2_L$ + 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<br>Stack Format $1<br>Stack Format $2<br>Stack Format $3<br>Stack Format $4<br>Stack Format $7 | 2<br>4<br>2<br>3<br>2<br>4 | 13<br>23<br>14<br>20<br>15<br>23 |
| RTR[c] | — | 7 | $1_L$ + 6 |
| RTS[c] | — | 5 | 5 |
| SBCD | Dy,Dx<br>–(Ay),–(Ax) | 1<br>3 | 3<br>$1_L$ + 3 |
| SUBX | Dy,Dx<br>–(Ay),–(Ax) | 1<br>3 | 1<br>$1_L$ + 2 |
| SWAP | — | 1 | 2 |
| TRAP#[a] | — | 16 | 16 |
| TRAPcc[f] | Taken<br>Not Taken | 19<br>5 | 19<br>5 |
| TRAPV[f] | Taken<br>Not Taken | 19<br>5 | 19<br>5 |
| UNLK | — | 2 | $1_L$ + 1 |
| UNPK | Dx,Dy,#<br>–(Ay),–(Ax),# | 1<br>3 | 4<br>$2_L$ + 4 |

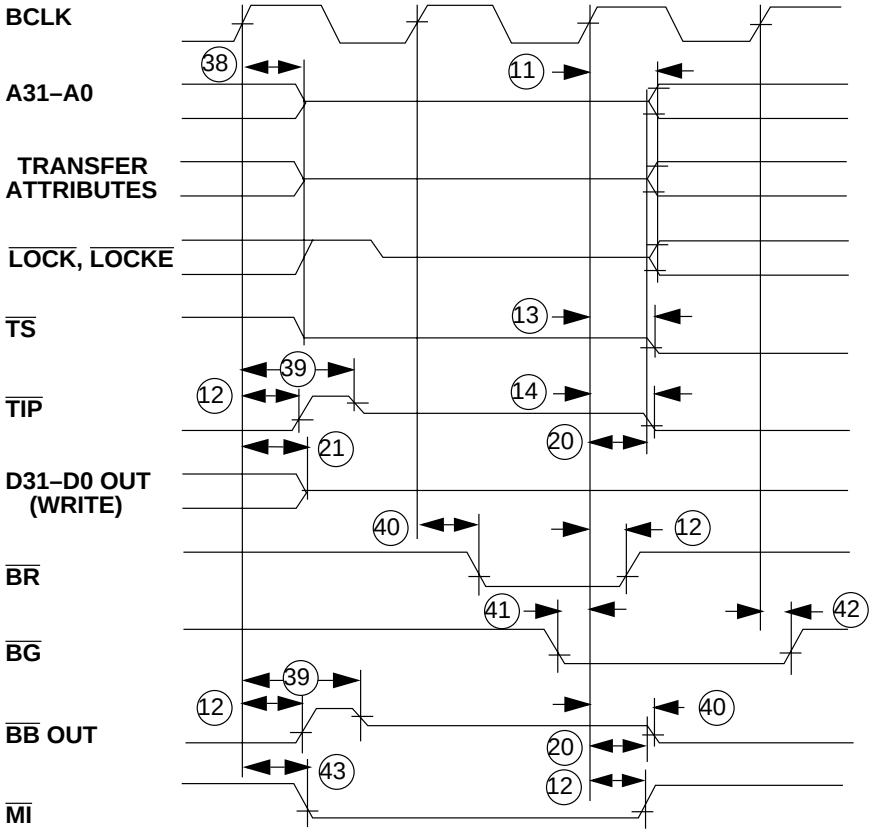
NOTES:

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 | BTST | | CAS[b] | | CHK[c,d] (<ea>, Dn) | |
|---|---|---|---|---|---|---|
| | <ea> Calculate | Execute | <ea> Calculate | Execute | <ea> Calculate | Execute |
| Dn | 1 | 1/2[a] | — | — | 8 | $1_L$ + 7 |
| An | — | — | — | — | — | — |
| (An) | 1 | 1/2 | 36 | $6_L$ + 31 | 9 | $2_L$ + 7 |
| (An)+ | 1 | 1/2 | 37 | $5_L$ + 31 | 9 | $2_L$ + 7 |
| –(An) | 1 | 1/2 | 37 | $5_L$ + 31 | 9 | $2_L$ + 7 |
| ($d_{16}$,An) | 2/1 | $1_L$ + 1/2 | 37 | $5_L$ + 31 | 9 | $2_L$ + 7 |
| ($d_{16}$,PC) | 3 | $2_L$ + 1/2 $_L$ + 2 | — | — | 10 | $3_L$ + 7 |
| (xxx).W, (xxx).L | 2/1 | $1_L$ + 1/2 | 36 | $5_L$ + 31 | 9 | $2_L$ + 7 |
| #<xxx> | — | — | — | — | 8 | $1_L$ + 7 |
| ($d_8$,An,Xn) | 3 | 3/4 | 36 | 36 | 10 | 10 |
| ($d_8$,PC,Xn) | 5 | $1_L$ + 4/1 $_L$ + 5 | — | — | 11 | $1_L$ + 10 |
| (BR,Xn) | 7/6 | $1_L$ + 6/1 $_L$ + 7 | 36 | $1_L$ + 35 | 12 | $1_L$ + 11 |
| (bd,BR,Xn) | 8/7 | $1_L$ + 7/1 $_L$ + 8 | 37 | $1_L$ + 36 | 13 | $1_L$ + 12 |
| ([bd,BR,Xn]) | 10/9 | $1_L$ + 9/1 $_L$ + 10 | 42 | 40 | 16 | $1_L$ + 15 |
| ([bd,BR,Xn],od) | 11/10 | $1_L$ + 10/1 $_L$ + 11 | 42 | $1_L$ + 41 | 17 | $1_L$ + 16 |
| ([bd,BR],Xn) | 11/10 | $3_L$ + 8/3 $_L$ + 9 | 42 | $3_L$ + 38 | 17 | $3_L$ + 14 |
| ([bd,BR],Xn,od) | 12/11 | $3_L$ + 9/3 $_L$ + 10 | 42 | $3_L$ + 39 | 18 | $3_L$ + 15 |

NOTES:
a. Bit instruction <ea> calculate and execute times T1/T2 apply to #<xxx>/Dn bit numbers.
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. Times listed are for Dn within bounds. This instruction interlocks the <ea> calculate and execute stages.




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

**Figure A-6. Bus Arbitration Timing**

## 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<br>$\overline{AVEC}$, 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<br>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<br>TMS, TDI, $\overline{TRST}$ | $I_{IL}$ | –1.1 | –0.18 | mA |
| Signal High Input Current, $V_{IH}$ = 2.0 V<br>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.

### 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. |