Freescale Semiconductor - MC68EC040FE20A Datasheet

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	184-BCQFP
Supplier Device Package	184-CQFP (31.3x31.3)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68ec040fe20a

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

1.7 DATA FORMAT SUMMARY

The M68040 supports the basic data formats of the M68000 family. Some data formats apply only to the IU, some only to the FPU, and some to both. In addition, the instruction set supports operations on other data formats such as memory addresses.

The operand data formats supported by the IU are the standard twos-complement data formats defined in the M68000 family architecture plus a new data format (16-byte block) for the MOVE16 instruction. Registers, memory, or instructions themselves can contain IU operands. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation.

Whenever an integer is used in a floating-point operation, the FPU automatically converts it to an extended-precision floating-point number before using the integer. The FPU implements single- and double-precision floating-point data formats as defined by the IEEE 754 standard. The FPU does not directly support packed decimal real format. However, by trapping as an unimplemented data format instead of as an illegal instruction, software emulation supports the packed decimal format. Additionally, each data format has a special encoding that represents one of five data types: normalized numbers, denormalized numbers, zeros, infinities, and not-a-numbers (NANs). Table 1-1 lists the data formats for both the IU and the FPU. Refer to M68000PM/AD, *M68000 Family Programmer's Reference Manual,* for details on data format organization in registers and memory.

Table 1-1. M68040 Data Formats

Operand Data Format	Size	Supported In	Notes
Bit	1 Bit	IU	—
Bit Field	1–32 Bits	IU	Field of Consecutive Bits
Binary-Coded Decimal (BCD)	8 Bits	IU	Packed: 2 Digits/Byte; Unpacked: 1 Digit/Byte
Byte Integer	8 Bits	IU, FPU	—
Word Integer	16 Bits	IU, FPU	—
Long-Word Integer	32 Bits	IU, FPU	—
Quad-Word Integer	64 Bits	IU	Any Two Data Registers
16-Byte	128 Bits	IU	Memory Only, Aligned to 16-Byte Boundary
Single-Precision Real	32 Bits	FPU	1-Bit Sign, 8-Bit Exponent, 23-Bit Fraction
Double-Precision Real	64 Bits	FPU	1-Bit Sign, 11-Bit Exponent, 52-Bit Fraction
Extended-Precision Real	80 Bits	FPU	1-Bit Sign, 15-Bit Exponent, 64-Bit Mantissa

1.8 ADDRESSING CAPABILITIES SUMMARY

The M68040 supports the basic addressing modes of the M68000 family. The register indirect addressing modes support postincrement, predecrement, offset, and indexing, which are particularly useful for handling data structures common to sophisticated

SECTION 2 INTEGER UNIT

This section describes the organization of the M68040 integer unit (IU) and presents a brief description of the associated registers. Refer to **Section 3 Memory Management Unit (Except MC68EC040 and MC68EC040V)** for details concerning the memory management unit (MMU) programming model, and to **Section 9 Floating-Point Unit (MC68040 Only)** for details concerning the floating-point unit (FPU) programming model.

2.1 INTEGER UNIT PIPELINE

The IU carries out logical and arithmetic operations using six separate subunits. Each unit is dedicated to a different stage of the IU pipeline, handling a total of six separate instructions simultaneously. Pipelining is a technique that overlaps the processing of different parts of several instructions. Pipelining simulates an assembly line with the IU containing a number of instructions in different phases of processing. The IU pipeline consists of six stages:

1. Instruction Fetch—Fetching an instruction from memory.
2. Decode—Converting an instruction into micro-instructions.
3. <ea> Calculate—If the instruction calls for data from memory, the location of the data, its memory address is calculated.
4. <ea> Fetch—Data is fetched from memory.
5. Execute—The data is manipulated during execution.
6. Write-Back—The result of the computation is written back to on-chip caches or external memory.

The pipeline contains special shadow registers that can begin processing future instructions for conditional branches while the main pipeline is processing current instructions. The <ea> calculate stage eliminates pipeline blockage for instructions with postincrement, postdecrement, or immediate add and load to address register for updates that occur in the <ea> calculate stage. The write-back stage can write data over the system bus to store a result in external memory or directly to on-chip caches. These write-backs to memory can be deferred until the most opportune moment because of the M68040 bus interface. Figure 2-1 illustrates the IU pipeline.

logical address bits. If the translation is resident, the MMU provides the physical address to the cache controller, which determines if the instruction or data being accessed is cached. The cache controller uses the lower address bits to index into memory. An external bus cycle is performed only when explicitly requested by the cache controller. When the translation is not in the ATC, the MMU searches the translation tables in memory for the translation information. Microcode and dedicated logic perform the address calculations and bus cycles required for this search.

3.1 MEMORY MANAGEMENT PROGRAMMING MODEL

The memory management programming model is part of the supervisor programming model for the M68040. The eight registers that control and provide status information for address translation in the M68040 are: the user root pointer register (URP), the supervisor root pointer register (SRP), the translation control register (TCR), four independent transparent translation registers (ITT0, ITT1, DTT0, and DTT1), and the MMU status register (MMUSR). Only programs that execute in the supervisor mode can directly access these registers. Figure 3-2 illustrates the memory management programming model.

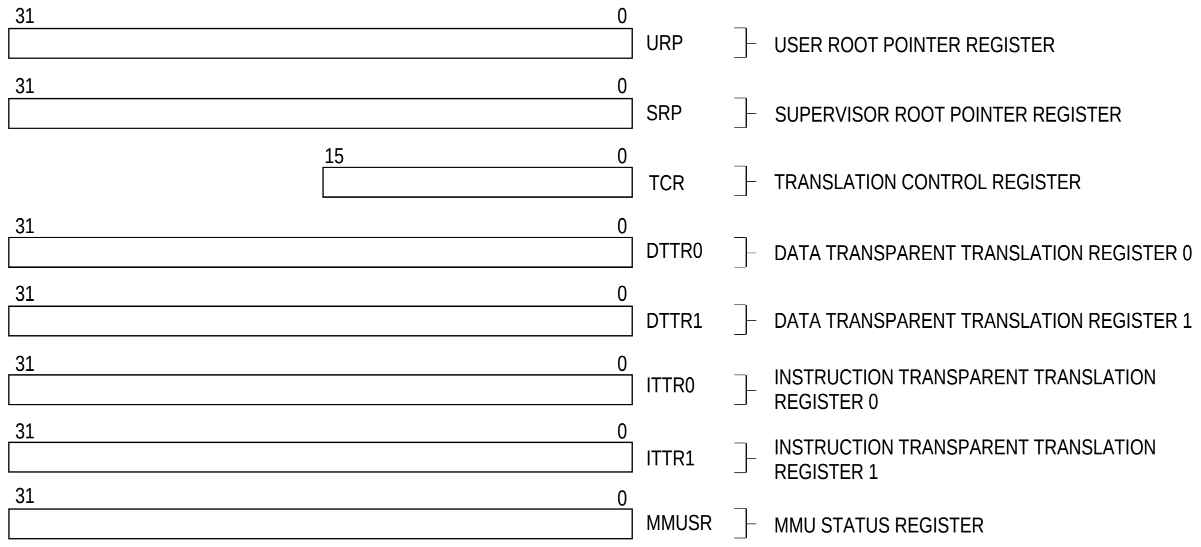

31
0
URP
USER ROOT POINTER REGISTER
31
0
SRP
SUPERVISOR ROOT POINTER REGISTER
15
0
TCR
TRANSLATION CONTROL REGISTER
31
0
DTTR0
DATA TRANSPARENT TRANSLATION REGISTER 0
31
0
DTTR1
DATA TRANSPARENT TRANSLATION REGISTER 1
31
0
ITTR0
INSTRUCTION TRANSPARENT TRANSLATION REGISTER 0
31
0
ITTR1
INSTRUCTION TRANSPARENT TRANSLATION REGISTER 1
31
0
MMUSR
MMU STATUS REGISTER

Figure 3-2. Memory Management Programming Model

3.1.1 User and Supervisor Root Pointer Registers

The SRP and URP registers each contain the physical address of the translation table's root, which the MMU uses for supervisor and user accesses, respectively. The URP points to the translation table for the current user task. When a new task begins execution, the operating system typically writes a new root pointer to the URP. A new translation table address implies that the contents of the ATCs may no longer be valid. A PFLUSH instruction should be executed to flush the ATCs before loading a new root pointer value, if necessary. Figure 3-3 illustrates the format of the 32-bit URP and SRP registers. Bits 8–

the access. During alternate bus master accesses, the M68040 samples TEA to detect completion of each bus transfer.

5.4.5 Transfer Cache Inhibit (TCI)

This input signal inhibits read data from being loaded into the M68040 instruction or data caches. TCI is ignored during all writes and after the first data transfer for both burst line reads and burst-inhibited line reads. TCI is also ignored during all alternate bus master transfers.

5.4.6 Transfer Burst Inhibit (TBI)

This input signal indicates to the processor that the accessed device cannot support burst mode accesses and that the requested line transfer should be divided into individual long-word transfers. Asserting TBI with TA terminates the first data transfer of a line access, which causes the processor to terminate the burst and access the remaining data for the line as three successive long-word transfers. During alternate bus master accesses, the M68040 samples the TBI to detect completion of each bus transfer.

5.5 SNOOP CONTROL SIGNALS

The following signals control the operation of the M68040 on-chip snoop logic. **Section 4 Instruction and Data Caches** provides information about the relationship of the snoop control signals to the caches, and **Section 7 Bus Operation** discusses the relationship of these signals to bus operation.

5.5.1 Snoop Control (SC1, SC0)

These input signals specify the snoop operation to be performed by the M68040 for an alternate bus master transfer. If the M68040 is allowed to snoop an alternate bus master read transfer, it can intervene in the access to supply data from its data cache when the memory copy is stale, ensuring that the alternate bus master receives valid data. Writes by an alternate bus master can also be snooped to either update the M68040 internal data cache with the new data or invalidate the matching cache lines, ensuring that subsequent M68040 reads access valid data. These signals are ignored when the processor is the bus master.

5.5.2 Memory Inhibit (MI)

This output signal prevents an alternate bus master from accessing possibly stale data in memory while the M68040 is unable to respond. MI is asserted during reset preventing external memory from responding. When the SCx signals indicate an access should be snooped, the M68040 keeps MI asserted until it determines if intervention in the access is required. If no intervention is required, MI is negated and memory is allowed to respond to complete the access. Otherwise, MI remains asserted and the M68040 completes the transfer as a slave. It updates its caches on a write or supplies data to the alternate bus master on a read. MI is negated when the M68040 is the bus master. During a snoop

Table 5-7 Signal Summary (Continued)

Signal Name	Mnemonic	Type	Active	Three-State
Transfer Start	TS	Input/Output	Low	Yes
Transfer Type	TT1, TT0	Input/Output	High	Yes
Test Clock	TCK	Input	—	—
Test Data Input	TDI	Input	High	—
Test Data Output	TDO	Output	High	Yes
Test Mode Select	TMS	Input	High	—
Test Reset	TRST	Input	Low	—
User-Programmable Attributes	UPA1, UPA0	Output	High	Yes
Power Supply	V_{CC}	Power	—	—

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

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

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

6.2.8 BYPASS

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

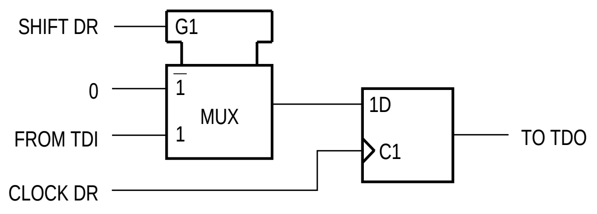


Figure 6-2. Bypass Register

6.3 BOUNDARY SCAN REGISTER

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

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

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

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

the data bus and asserting $\overline{\text{TA}}$. A line transfer performed in this manner with a single address is referred to as a line burst transfer.

The M68040 also supports burst-inhibited line transfers for memory devices that are unable to support bursting. For this type of bus cycle, the selected device supplies the first long word pointed to by the processor address and asserts transfer burst inhibit ($\overline{\text{TBI}}$) with $\overline{\text{TA}}$ for the first transfer of the line access. The processor responds by terminating the line burst transfer and accessing the remainder of the line, using three long-word read bus cycles. Although the selected device can then treat the line transfer as four, independent, long-word bus cycles, the bus controller still handles the four transfers as a single line transfer and does not allow other unrelated processor accesses or bus arbitration to intervene between the transfers. $\overline{\text{TBI}}$ is ignored after the first long-word transfer.

Line reads to support cache line filling can be cache inhibited by asserting transfer cache inhibit ($\overline{\text{TCI}}$) with $\overline{\text{TA}}$ for the first long-word transfer of the line. The assertion of $\overline{\text{TCI}}$ does not affect completion of the line transfer, but the bus controller latches and passes it to the memory controller for use. $\overline{\text{TCI}}$ is ignored after the first long-word transfer of a line burst transfer and during the three long-word bus cycles for a burst-inhibited line transfer.

The address placed on the address bus by the processor for line transfers does not necessarily point to the most significant byte of each long word because for a line read, A1 and A0 are copied from the original operand address supplied to the memory unit by the integer unit. These two bits are also unchanged for the three long-word bus cycles for a burst-inhibited line transfer. The selected device should ignore A1 and A0 for long-word and line read transfers.

The address of an instruction fetch will always be aligned to a half-line boundary ($XXXXXXX0 or $XXXXXXX8); therefore, compilers should attempt to locate branch targets on half-line boundaries to minimize branch stalls. For example, if the target of a branch is a two-word instruction located at $1000000C, the following burst sequence will occur upon a cache miss: $10000008, $1000000C, $10000000, then $10000004. The internal pipeline of the M68040 stalls until the second access of the burst (the address of the instruction to be executed) has completed. Figures 7-10 and 7-11 illustrate a flowchart and functional timing diagram for a line read bus transfer.

MC68LC040 memory units and **Appendix B MC68EC040** for information on the MC68EC040 memory unit.

The processor asserts $\overline{TS}$ during C1 to indicate the beginning of a bus cycle. If not already asserted from a previous bus cycle, the $\overline{TIP}$ signal is also asserted at this time to indicate that a bus cycle is active.

Clock 2 (C2)

During the first half of the first clock after C1, the processor negates $\overline{TS}$ and drives the data bus with the data to be written. The selected device uses R/$\overline{W}$, SIZ1, and SIZ0 to latch the data on the data bus. Concurrently, the selected device asserts $\overline{TA}$ and either negates or asserts $\overline{TBI}$ to indicate it can or cannot support a burst transfer. At the end of the first clock after C1, the processor samples the level of $\overline{TA}$ and $\overline{TBI}$. If $\overline{TA}$ is asserted, the transfer terminates. If $\overline{TA}$ is not recognized asserted, the processor inserts wait states instead of terminating the transfer. The processor continues to sample $\overline{TA}$ and $\overline{TBI}$ on successive rising edges of BCLK until $\overline{TA}$ is recognized asserted.

If $\overline{TBI}$ was negated with $\overline{TA}$, the processor continues the cycle with C3. Otherwise, if $\overline{TBI}$ was asserted, the line transfer is burst inhibited, and the processor writes the remaining three long words using long-word write bus cycles. Only in this case does the processor increment A3 and A2 for each write, and the new address is placed on the address bus for each bus cycle. Refer to **7.4.3 Byte, Word, and Long-Word Write Transfers** for information on long-word writes. If no waits states are generated, a burst-inhibited line write completes in eight clocks instead of the five required for a burst write.

Clock 3 (C3)

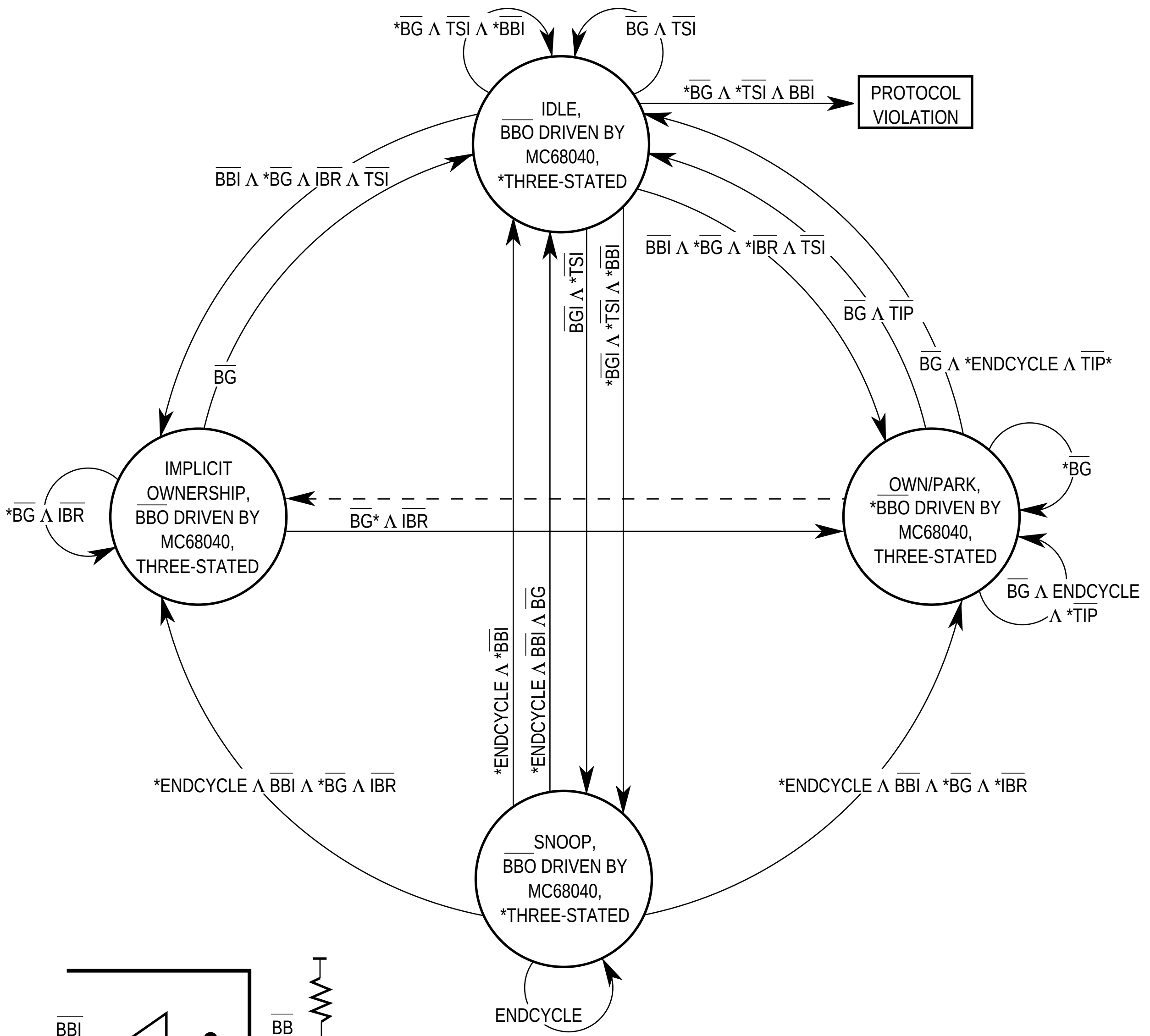
The processor drives the second long word of data on the data bus and holds the address and transfer attribute signals constant during C3. The selected device increments A3 and A2 to reference the next long word, latches this data from the data bus, and asserts $\overline{TA}$. At the end of C3, the processor samples the level of $\overline{TA}$; if $\overline{TA}$ is asserted, the transfer terminates. If $\overline{TA}$ is not recognized asserted at the end of C3, the processor inserts wait states instead of terminating the transfer. The processor continues to sample $\overline{TA}$ on successive rising edges of BCLK until $\overline{TA}$ is recognized asserted.

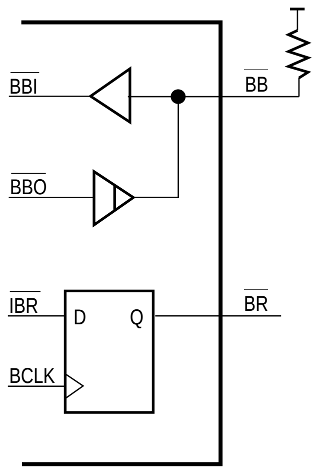
Clock 4 (C4)

This clock is identical to C3 except that the value driven on the data bus corresponds to the third long word of data for the burst.

Clock 5 (C5)

This clock is identical to C3 except that the value driven on the data bus corresponds to the fourth long word of data for the burst. After the processor recognizes the last $\overline{TA}$ assertion and terminates the line write bus cycle, $\overline{TIP}$ remains asserted if the processor is ready to begin another bus cycle. Otherwise, the processor negates $\overline{TIP}$ during the first half of the next clock. The processor also three-states the data bus during the first half of the next clock following termination of the write cycle.





$\overline{\text{IBR}}$	=	Internal bus request signal (see schematic below).
$\overline{\text{BBI}}$	=	Bus busy driven by alternate bus master.
$\overline{\text{TSI}}$	=	Transfer start as an input, sampled by the MC68040.
ENDCYCLE	=	Whatever terminates a bus transaction whether it is normal, bus error, or retried. Note that false burst cycles are treated as a line transaction. False locked transactions are treated the same as any other bus cycle.
– – – – – –	=	The 040 may or may not transition if an active bus cycle is terminated with a bus error, and BG is asserted.
*	**=**	**Indicates the signal is asserted for that device.**

Figure 7-30. M68040 Internal Interpretation State Diagram and External Bus Arbiter Circuit

give the bus to processor 2. The arbiter remains in state B until $\overline{\text{BB}}$ is negated, signifying the end of the bus cycle.

Once state C is reached, depending on whether or not processor 2 asserts $\overline{\text{BR2}}$ and then negates $\overline{\text{BR2}}$ because of a disregard request condition, processor 1 may or may not actively begin a bus cycle. If no other bus requests are pending by the time state C is reached, processor 2 is in the implicit ownership state. If processor 1 asserts $\overline{\text{BR1}}$, then it is possible for state C to persist for only one clock. In this case, processor 2 does not have a chance to run any active bus cycles.

A null bus cycle tenure is better than having the external bus arbiter wait for processor 2 to perform at least one bus cycle before returning bus ownership to processor 1, even though this appears to be a waste of bus arbitration overhead. Note that once processor 2 enters the disregard request condition, processor 2 reasserts $\overline{\text{BR}}$ anywhere from one clock to an undetermined number of clocks before running another bus cycle. Waiting for processor 2 to run a bus cycle can result in a temporary bus arbitration lockup.

This bus arbitration scheme is restricted if the system supports the relinquish and retry operation that can occur for the last write cycle of a locked transfer. In this case, $\overline{\text{LOCKE}}$ cannot be used. Assuming that $\overline{\text{LOCKE}}$ is always negated excludes the need for $\overline{\text{LOCKE}}$ in an arbitration similar to this example. The reason for this restriction is that the external bus arbiter gives up the bus to the other processor once $\overline{\text{LOCKE}}$ is asserted. If a relinquish and retry operation were to occur, then the next bus cycle would be from the other processor violating the integrity of the locked transfer.

7.8.2.2 DUAL M68040 PRIORITIZED ARBITRATION. This example is very similar to the dual M68040 fairness arbitration example, except that one processor is assigned higher priority over the other. Processor 2 can own the bus only if there are no processor 1 pending requests. It is important to note that when the processor asserts the $\overline{\text{LOCK}}$ signal, it also asserts $\overline{\text{BR1}}$. This implementation replaces $\overline{\text{LOCK}}$ with $\overline{\text{BR}}$ because $\overline{\text{BR}}$ is more demanding than using $\overline{\text{LOCK}}$. Only when processor 2 is in the middle of a locked operation does it have higher priority than processor 1. Similar to the M68040 fairness arbitration example, the restriction on using $\overline{\text{LOCKE}}$ applies to this example. Figure 7-36 illustrates the state diagram for dual M68040 prioritized arbitration.

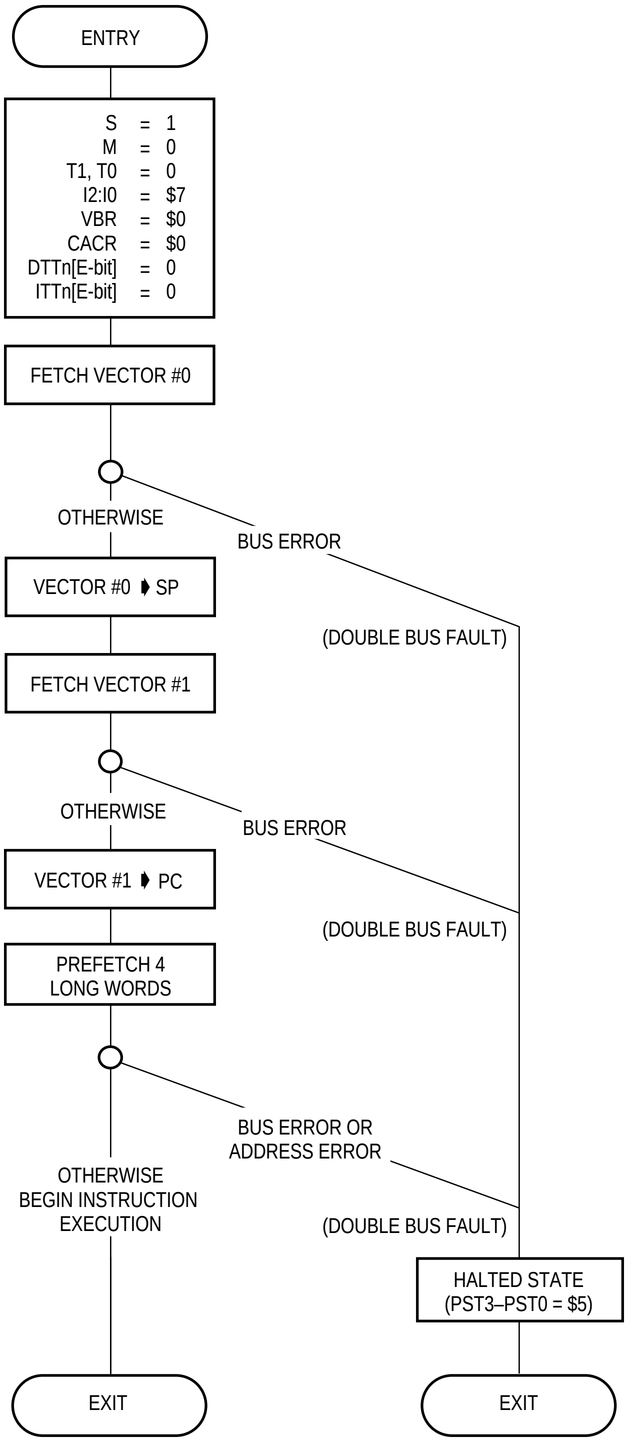


Figure 8-5. Reset Exception Processing Flowchart

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

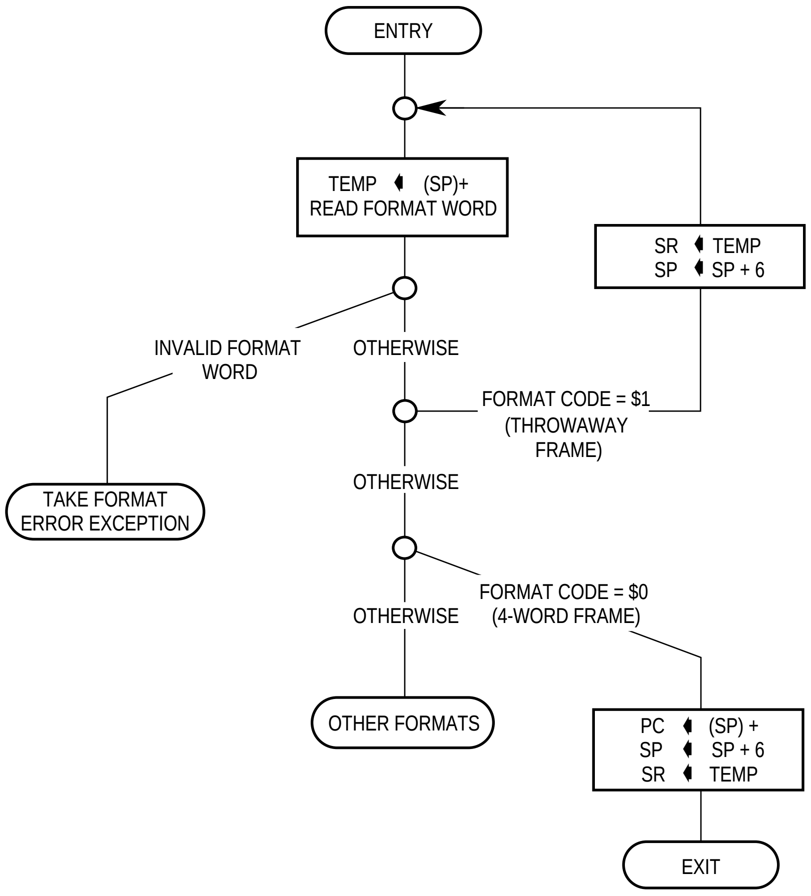
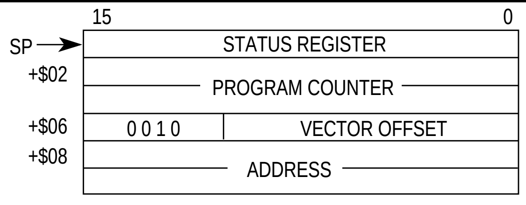


Figure 8-6. Flowchart of RTE Instruction for Throwaway Four-Word Frame

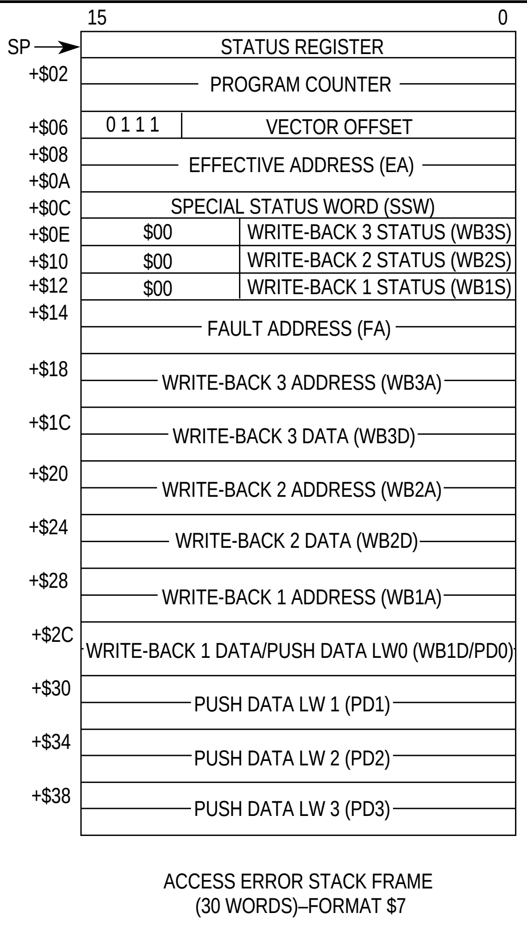
8.4.3 Six-Word Stack Frame (Format $2)

If a six-word throwaway stack frame is on the active stack and an RTE instruction is encountered, the processor restores the SR and PC values from the stack, increments the active supervisor stack pointer by $C, and resumes normal instruction execution.

Stack Frames	Exception Types	Stacked PC Points To
 15 0 SP +$02 +$06 +$08 STATUS REGISTER PROGRAM COUNTER 0 0 1 0 VECTOR OFFSET ADDRESS SIX-WORD STACK FRAME–FORMAT $2	• CHK, CHK2, TRAPcc, FTRAPcc, TRAPV, Trace, or Zero Divide • Unimplemented Floating-Point Instruction • Address Error	• Next Instruction: address is the address of the instruction that caused the exception. • Next Instruction: address is the calculated <ea> for the floating-point instruction. • Instruction that caused the address error, address is the reference address – 1.

8.4.6 Access Error Stack Frame (Format $7)

A 30-word access error stack frame is created for data and instruction access faults other than instruction address errors. In addition to information about the current processor status and the faulted access, the stack frame also contains pending write-backs that the access error exception handler must complete. The following paragraphs describe in detail the format for this frame and how the processor uses it when returning from exception processing.

Stack Frames	Exception Types	Stacked PC Points To
	• Data or Instruction Access Fault (ATC Fault or Bus Error)	• Next Instruction

8.4.6.1 EFFECTIVE ADDRESS. The effective address contains address information when one of the continuation flags CM, CT, CU, or CP in the SSW is set.

8.4.6.2 SPECIAL STATUS WORD (SSW). The SSW information indicates whether an access to the instruction stream or the data stream (or both) caused the fault and contains status information for the faulted access. Figure 8-7 illustrates the SSW format.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CP	CU	CT	CM	MA	ATC	LK	RW	X	SIZE		TT		TM		

Figure 8-7. Special Status Word Format

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

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

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

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

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

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

10.5 MISCELLANEOUS INTEGER UNIT INSTRUCTION TIMINGS

Instruction	Condition	<ea> Calculate	Execute
ABCD	Dy,Dx –(Ay),–(Ax)	1 3	3 1_L + 3
ADDX	Dy,Dx –(Ay),–(Ax)	1 3	1 1_L + 2
ANDI #<xxx>,CCR	—	1	4
ANDI #<xxx>,SR[a]	—	9	1_L + 8
Bcc	Branch Taken Branch Not Taken	2 3	2 3
BRA	Branch Taken Branch Not Taken	2 3	2 3
BSR <offset>	—	2	1_L + 1
CAS2[b]	True False	56 51	6_L + 49 6_L + 44
CMPM	—	3	1_L + 2
DBcc[c]	False, Count > –1 False, Count = –1 True	3 4 4	3 4 4
EORI #<xxx>,CCR	—	1	4
EORI #<xxx>,SR[a]	—	9	1_L + 8
EXG	Dy,Dx Ay,Ax Dy,Ax	1 2 1	1 1_L + 1 1
EXT	Word Long Word	1 1	2 1
EXTB	Long Word	1	1
ILLEGAL[a]	A-Line Unimplemented F-Line Unimplemented	16 16	16 16
LINK	—	3	2_L + 1
MOVE USP	USP,An An,USP[a]	3 7	2_L + 1 1_L + 6
MOVE16[c,d]	(Ax)+,(Ay)+ xxx.L,(An) xxx.L,(An)+ (An),xxx.L (An)+,xxx.L	6 4 5 4 4	1_L+ 7 7 8 7 7
MOVEC[b]	Rn,Rc Rc,Rn	7 11	1_L + 6 1_L + 10
MOVEP[c]	MOVEP.W Dn,d16(An) MOVEP.L Dn,d16(An) MOVEP.W d16(An),Dn MOVEP.L d16(An),Dn	11 13 4 8	2_L + 9 2_L + 11 2_L + 5 2_L + 8
MOVEQ	—	1	1
NOP[a]	—	8	1_L + 7

10.7.3 Timings in the Floating-Point Unit (Continued)

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

C.2.4 LPSTOP Instruction Summary

Operation: If Supervisor State
    Immediate Data ➧ SR
    SR ➧ Broadcast Cycle
    STOP
  Else TRAP

Assembler Syntax: LPSTOP #<data>

Attributes: Privileged Word Sized

Condition Codes: Set according to the immediate operand.

Description: See **C.2 Low Power Stop Mode**.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
IMMEDIATE DATA															

Instruction Fields: Immediate field—Specifies the data to be loaded into the status register.