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	68060
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	75MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	3.3V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	206-BPGA
Supplier Device Package	206-PGA (47.25x47.25)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68lc060rc75

Section 3
Integer Unit

Section 4
Memory Management Unit

Section 7
Bus Operation

2.11.4 Test Data In (TDI)

This input signal provides a serial data input to the TAP. TDI should be tied to V_{CC} if it is not used and emulation mode is not to be used.

2.11.5 Test Data Out (TDO)

This three-state output signal provides a serial data output from the TAP. The TDO output can be placed in a high-impedance mode to allow parallel connection to board-level test data paths.

2.11.6 Test Reset ($\overline{\text{TRST}}$)

This input signal provides an asynchronous reset of the TAP controller. $\overline{\text{TRST}}$ should be grounded if 1149.1 operation is not to be used.

2.12 THERMAL SENSING PINS (THERM1, THERM0)

THERM1 and THERM0 are connected to an internal thermal resistor and provide information about the average temperature of the die. The resistance across these two pins is proportional to the average temperature of the die. The temperature coefficient of the resistor is approximately 1.2 Ω/°C with a nominal resistance of 400Ω at 25°C.

2.13 POWER SUPPLY CONNECTIONS

The MC68060 requires connection to a V_{CC} power supply, positive with respect to ground. The V_{CC} and ground connections are grouped to supply adequate current to the various sections of the processor. **Section 13 Ordering Information and Mechanical Data** describes the groupings of the V_{CC} and ground connections.

2.14 SIGNAL SUMMARY

Table 2-8 provides a summary of the electrical characteristics of the MC68060 signals.

4.1 MEMORY MANAGEMENT PROGRAMMING MODEL

The memory management programming model is part of the supervisor programming model for the MC68060. The seven registers that control and provide status information for address translation in the MC68060 are: the user root pointer register (URP), the supervisor root pointer register (SRP), the translation control register (TCR), and four independent transparent translation registers (ITTR0, ITTR1, DTTR0, and DTTR1). Only programs that execute in the supervisor mode can directly access these registers. Figure 4-2 illustrates the memory management programming model.

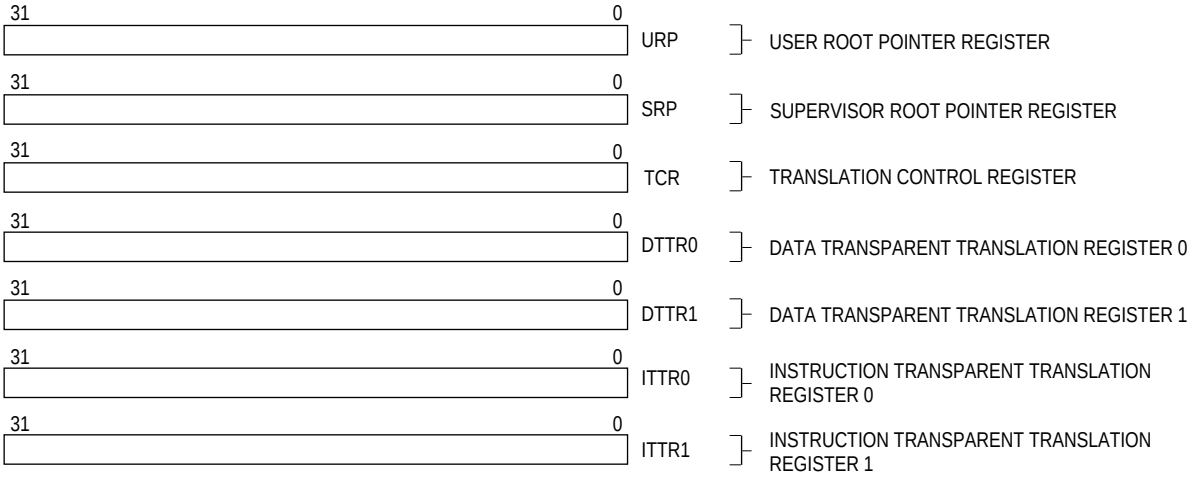


Figure 4-2. Memory Management Programming Model

4.1.1 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. Writing a root pointer register does not affect the contents of the ATCs. A PFLUSH instruction should be executed to flush the ATCs before loading a new root pointer value, if necessary. Figure 4-3 illustrates the format of the 32-bit URP and SRP registers. Bits 8–0 of an address loaded into the URP or the SRP must be zero. Transfers of data to and from these 32-bit registers are long-word transfers.

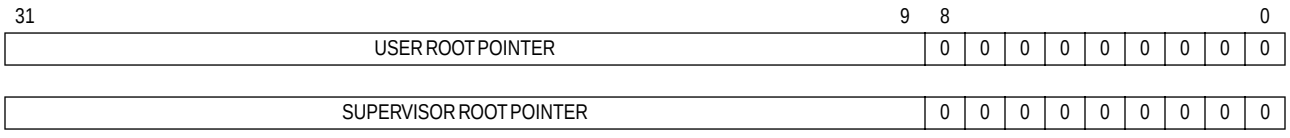


Figure 4-3. URP and SRP Register Formats

4.1.2 Translation Control Register

The 32-bit TCR contains control bits which select translation properties. The operating system must flush the ATCs before enabling address translation since the TCR accesses and reset do not flush the ATCs. All unimplemented bits of this register are read as zeros and must always be written as zeros. The MC68060 always uses long-word transfers to access this 32-bit register. All bits are cleared by reset. Figure 4-4 illustrates the TCR.

31															16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	E	P	NAD	NAI	FOTC	FITC	DCO		DUO		DWO		DCI		DUI	0

Figure 4-4. Translation Control Register Format

Bits 31–16—Reserved by Motorola. Always read as zero.

E—Enable

This bit enables and disables paged address translation.

0 = Disable
1 = Enable

A reset operation clears this bit. When translation is disabled, logical addresses are used as physical addresses. The MMU instruction, PFLUSH, can be executed successfully despite the state of the E-bit. If translation is disabled and an access does not match a transparent translation register (TTR), the default attributes for the access on the TTR is defined by the DCO, DUO, DCI, DWO, DUI (default TTR) bits in TCR.

P—Page Size

This bit selects the memory page size.

0 = 4 Kbytes
1 = 8 Kbytes

NAD—No Allocate Mode (Data ATC)

This bit freezes the data ATC in the current state, by enforcing a no-allocate policy for all accesses. Accesses can still hit, misses will cause a table search. A write access which finds a corresponding valid read will update the M-bit and the entry remains valid.

0 = Disabled
1 = Enable

NAI—No Allocate Mode (Instruction ATC)

This bit freezes the instruction ATC in the current state, by enforcing a no-allocate policy for all accesses. Accesses can still hit, misses will cause a table search.

0 = Disabled
1 = Enable

FOTC—1/2-Cache Mode (Data ATC)

0 = The data ATC operates with 64 entries.
1 = The data ATC operates with 32 entries.

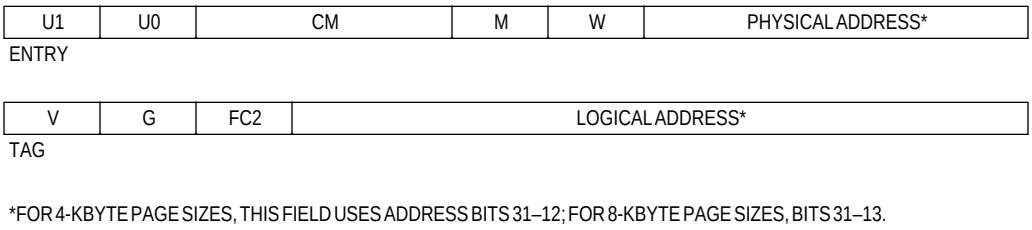


Figure 4-20. ATC Entry and Tag Fields

CM—Cache Mode

This field selects the cache mode and accesses serialization as follows:

00 = Cachable, Writethrough
01 = Cachable, Copyback
10 = Noncachable, Precise
11 = Noncachable, Imprecise

Section 5 Caches provides detailed information on caching modes.

FC2—Function Code Bit 2 (Supervisor/User)

This bit contains the function code corresponding to the logical address in this entry. FC2 is set for supervisor mode accesses and cleared for user mode accesses.

G—Global

When set, this bit indicates the entry is global. Global entries are not invalidated by the PFLUSH instruction variants that specify nonglobal entries, even when all other selection criteria are satisfied.

Logical Address

This 16-bit field contains the most significant logical address bits for this entry. All 16 bits of this field are used in the comparison of this entry to an incoming logical address when the page size is 4 Kbytes. For 8-Kbytes pages, the least significant bit of this field is ignored.

M—Modified

The modified bit is set when a valid write access to the logical address corresponding to the entry occurs. If the M-bit is clear and a write access to this logical address is attempted, the MC68060 suspends the access, initiates a table search to set the M-bit in the page descriptor, and writes over the old ATC entry with the current page descriptor information. The MMU then allows the original write access to be performed. This 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.

If the paged MMU is disabled (the E-bit in the TCR register is clear) and the TTRs are disabled or do not match, then the status and protection attributes are defined by the default translation bits (DCO, DUO, DWO, DCI, and DUI) in the TCR.

4.5 ADDRESS TRANSLATION SUMMARY

If the paged MMU is enabled (the E-bit in the TCR is set), the instruction and data MMUs process translations by first comparing the logical address and privilege mode with the parameters of the TTRs if they are enabled. If there is a match, the MMU uses the logical address as a physical address for the access. If there is no match, the MMU compares the logical address and privilege mode with the tag portions of the entries in the ATC and uses the corresponding physical address for the access when a match occurs. When neither a TTR nor a valid ATC entry matches, the MMU initiates a table search operation to obtain the corresponding physical address from the translation table. When a table search is required, the processor suspends instruction execution activity and, at the end of a successful table search, stores the address mapping in the appropriate ATC and retries the access. The MMU creates a valid ATC entry for the logical address. If the table search encounters an invalid descriptor, or a write-protect for a write, or is a user access and encounters a supervisor-only flag, then the access error exception is taken whenever the access is needed (immediately for operands and deferred for instruction fetches).

If a write or locked read-modify-write access results in an ATC hit but the page is write protected, the access is aborted, and an access error exception is taken. If the page is not write protected and the modified bit of the ATC entry is clear, a table search proceeds to set the modified bit in both the page descriptor in memory and in the ATC; the access is retried. The ATC provides the address translation for the access if the modified bit of the ATC entry is set for a write or locked read-modify-write access to an unprotected page and if none of the TTRs (instruction or data, as appropriate) match.

Figure 4-21 illustrates a general flowchart for address translation. The top branch of the flowchart applies to transparent translation. The bottom three branches apply to ATC translation.

4.6 $\overline{\text{RSTI}}$ AND $\overline{\text{MDIS}}$ EFFECT ON THE MMU

The following paragraph describes how the MMU is affected by the $\overline{\text{RSTI}}$ and $\overline{\text{MDIS}}$ pins.

4.6.1 Effect of $\overline{\text{RSTI}}$ on the MMUs

When the MC68060 is reset by the assertion of the reset input signal, the E-bits of the TCR and TTRs are cleared, disabling address translation. This reset causes logical addresses to be passed through as physical addresses, allowing an operating system to set up the translation tables and MMU registers as required. After the translation tables and registers are initialized, the E-bit of the TCR can be set, enabling paged address translation. While address translation is disabled, the default TTR is used. The default TTR attribute bits are cleared upon reset, so that immediately after assertion of $\overline{\text{RSTI}}$ the attributes will specify write-through cachable mode, no write protection, user page attribute bits cleared, and 1/2-cache mode disabled.

A reset of the processor does not invalidate any entries in the ATCs page size. A PFLUSH instruction must be executed to flush all existing valid entries from the ATCs after a reset

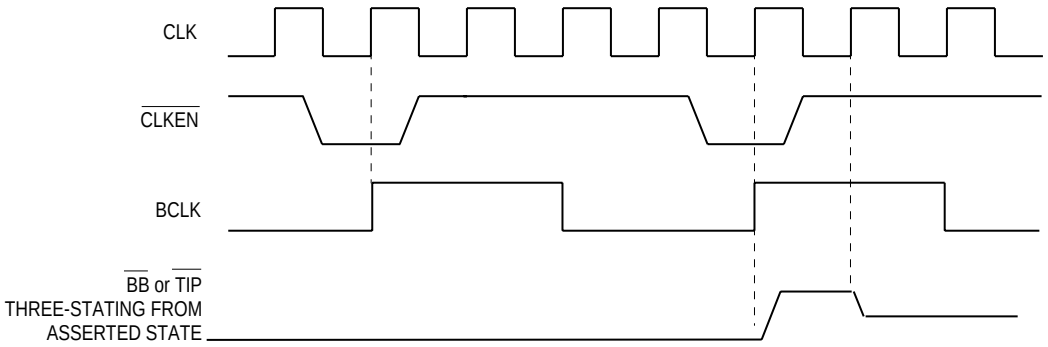


Figure 7-4. Quarter-Speed Clock

7.2 FULL-, HALF-, AND QUARTER-SPEED BUS OPERATION AND BCLK

To simplify the description of full-, half-, and quarter-speed bus operation, the term “bus clock” or “BCLK” is introduced to describe the effective frequency of bus operation. The bus clock is analogous to the MC68040 clock input called BCLK. The MC68040 BCLK defines when input signals are sampled and when output signals begin to transition. Once the relationship of CLK, $\overline{\text{CLKEN}}$, and the virtual BCLK is established, it is possible to describe the MC68060 bus more easily, relative to BCLK.

$\overline{\text{CLKEN}}$ allows the bus to synchronize to BCLK which is running at half or quarter speed of the processor clock (CLK). On rising CLK edges in which $\overline{\text{CLKEN}}$ is asserted, inputs to the processor are recognized and outputs of the processor may begin to assert, negate, or three-state. On rising CLK edges in which $\overline{\text{CLKEN}}$ is negated, no inputs are recognized and no outputs begin to change (except $\overline{\text{BB}}$ and $\overline{\text{TIP}}$). Figure 7-1 illustrates the general relationship between CLK, $\overline{\text{CLKEN}}$, and most input and output signals.

For brevity, the term “full-speed bus” is introduced to refer to systems in which BCLK is running at the same frequency as CLK. The term “half-speed bus” refers to systems in which BCLK is running at half the frequency of CLK. For those familiar with the MC68040, the half-speed bus is analogous to the MC68040 implementation. The term “quarter-speed bus” refers to systems in which BCLK is running at one quarter the frequency of CLK. The MC68060 clocking mechanism is designed so that systems designed today can be upgraded with higher-frequency MC68060s, without forcing the rest of the system to operate at the same higher processor frequency. This flexibility also allows the MC68060 to be used in existing MC68040 system designs.

A full-speed bus design is achieved by continuously asserting $\overline{\text{CLKEN}}$ as shown in Figure 7-2. A half speed bus is achieved by asserting $\overline{\text{CLKEN}}$ about every other rising edge of CLK. Figure 7-3 shows a timing diagram of the relationship between CLK, $\overline{\text{CLKEN}}$, and BCLK for half-speed bus operation. A quarter-speed bus is achieved by asserting $\overline{\text{CLKEN}}$ once about every four rising edges of CLK. Figure 7-4 shows a timing diagram of the relationship between CLK, $\overline{\text{CLKEN}}$, and BCLK for quarter-speed bus operation.

Note that once BCLK has been established, inputs and outputs appear to be synchronized to this virtual BCLK. To simplify the description of MC68060 bus operation, the rising edges

BUSCR is used to control the LOCK and LOCKE outputs. Refer to **7.4 Bus Control Register** for the format of the BUSCR. Emulation of these instructions is done as part of the MC68060 software package (M68060SP). Refer to **Appendix C MC68060 Software Package** for more information.

7.7.7 Using CLA to Increment A3 and A2

The MC68060 provides the capability to cycle long-word address bits A3, A2 based on the CLA signal, which should assist in supporting high-speed DRAM systems. CLA may also be used to support bursting for slaves which do not burst.

The processor begins sampling CLA immediately following the BCLK rising edge that causes TS to assert. The initial address of the line transfer is that of the first requested or needed long word and the attributes are those of the line transfer. After each BCLK rising edge when CLA is asserted, the long-word address (A3, A2) increments in circular wrap-around fashion. If CLA is negated, A3, A2 does not change, but remains fixed, as on the MC68040 processor. Since CLA is not an acknowledge termination signal, it is not affected by the acknowledge termination ignore state capability, if that mode is enabled. Also note that the A3, A2 increments in a circular wrap around fashion for as many times as CLA is asserted about a rising BCLK edge.

Figure 7-24 shows how CLA may be used for a high-speed DRAM design. In this figure, the DRAM design requires a means of cycling A3, A2 before TA is asserted to the processor. CLA provides a method of avoiding a delay which would otherwise be incurred with the use of an external medium-scale integration (MSI) counter. W0 to W3 represent A3, A2 incrementing. C0 to C3 represent the column address sequencing caused by the change of A3, A2. The timing diagram represents a 5:3:3:3 design, which is feasible with a full-speed 50-MHz clock and 65-ns page-mode DRAMs.

7.8 ACKNOWLEDGE CYCLES

Bus transfers with transfer type signals TT1 and TT0 = $3 are classified as acknowledge bus cycles. The following paragraphs describe interrupt acknowledge, breakpoint acknowledge, and LPSTOP broadcast bus cycles that use this encoding.

7.8.1 Interrupt Acknowledge Cycles

When a peripheral device requires the services of the MC68060 or is ready to send information that the processor requires, it can signal the processor to take an interrupt exception. The interrupt exception transfers control to a routine that responds appropriately. The peripheral device uses the interrupt priority level signals (IPLx) to signal an interrupt condition to the processor and to specify the priority level for the condition. Refer to **Section 8 Exception Processing** for a discussion on the IPLx levels and IPEND.

The status register (SR) of the MC68060 contains an interrupt priority mask (I2–I0 bits). The value in the interrupt mask is the highest priority level that the processor ignores. When an interrupt request has a priority higher than the value in the mask, the processor makes the request a pending interrupt. IPLx must maintain the interrupt request level until the MC68060 acknowledges the interrupt to guarantee that the interrupt is recognized. The MC68060 continuously samples IPLx on consecutive rising edges of CLK to synchronize

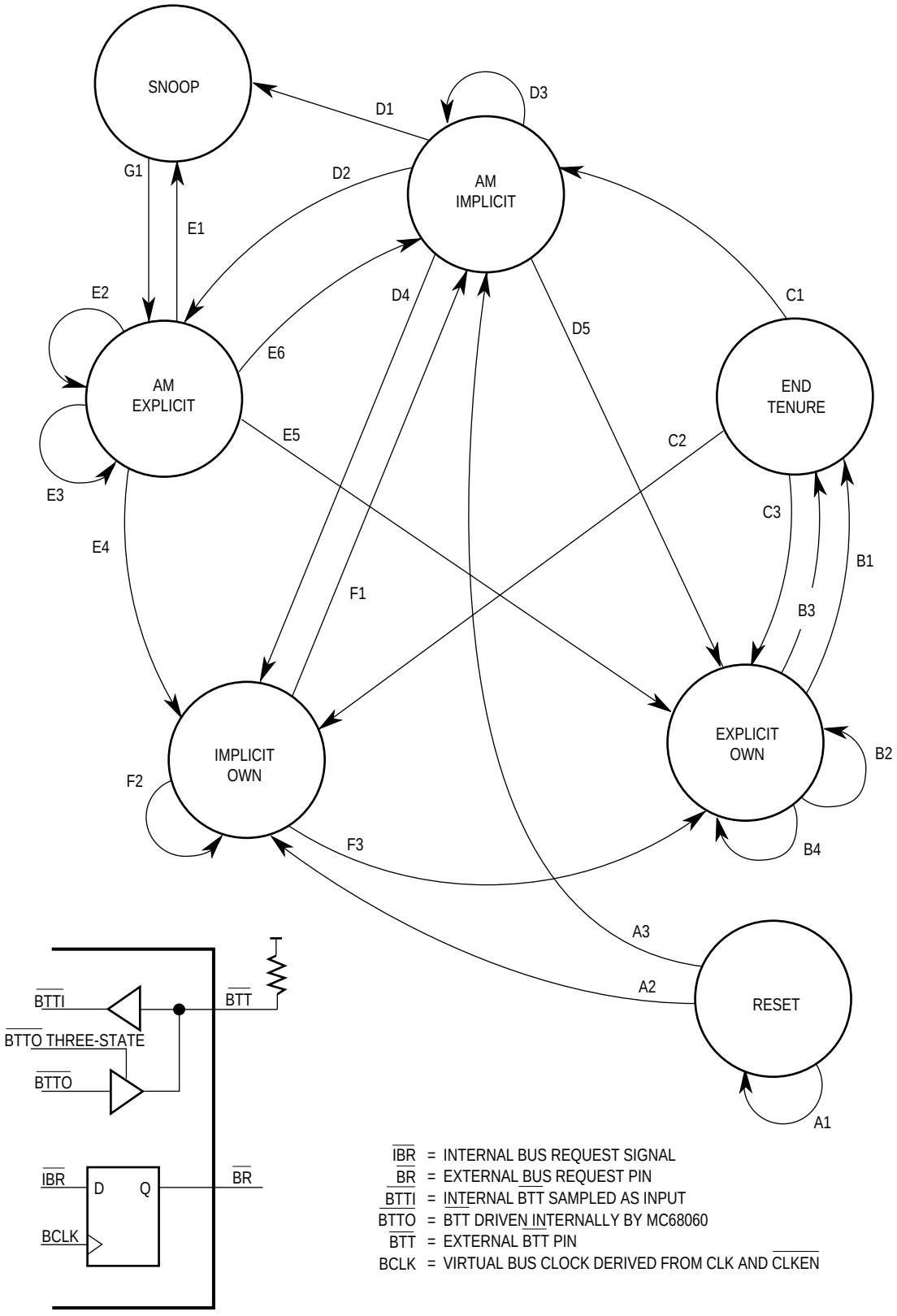


Figure 7-42. MC68060-Arbitration Protocol State Diagram

The ignore state settings can be used to make the system design of the acknowledge termination logic simpler than in existing MC68040 systems that required these signals to be valid (either asserted or negated) about every rising BCLK edge. Thus, using the acknowledge termination ignore state capability allows the use of slower ASICs and PALs to be used for generating the acknowledge termination signals without the requirement that these signals be at a valid logic level about every rising BCLK edge.

7.14.2 Acknowledge Termination Protocol

The MC68060 provides system designers a choice of using either the MC68040 acknowledge termination protocol or the native-MC68060 acknowledge termination protocol. The native-MC68060 acknowledge termination protocol is chosen if $\overline{\text{IPL1}}$ is asserted during reset.

The MC68040 acknowledge termination protocol is provided for MC68040 compatibility. In this protocol, a retry is indicated by having both $\overline{\text{TA}}$ and $\overline{\text{TEA}}$ asserted simultaneously. In this mode, the $\overline{\text{TRA}}$ signal must be pulled up at all times. Refer to Table 7-4 and Table 7-5 for details on acknowledge termination signal encoding.

The native-MC68060 acknowledge termination protocol is provided to aid in high-frequency designs. The signal $\overline{\text{TRA}}$ is used to indicate a retry operation, as opposed to using a combination of $\overline{\text{TA}}$ and $\overline{\text{TEA}}$ to indicate a retry. Refer to Table 7-4 and Table 7-5 for details on the native-MC68060 acknowledge termination signal encoding.

7.14.3 Extra Data Write Hold Time Mode

In this mode, the MC68060 holds the contents of the data bus valid during a write bus cycle for an extra BCLK period after a valid $\overline{\text{TA}}$ is sampled. This mode is enabled if $\overline{\text{IPL2}}$ is asserted during reset. When this mode is enabled, a zero wait state burst bus cycle is not possible and systems must be designed to insert wait states on burst accesses. Figure 7-52 shows an example of a line transfer cycle with this mode enabled. Read cycles are unaffected by this mode.

SECTION 8
EXCEPTION PROCESSING

Exception processing is the activity performed by the processor in preparing to execute a special routine for any condition that causes an exception. Exception processing does not include execution of the routine itself. This section describes the processing for each type of integer unit exception, exception priorities, the return from an exception, and bus fault recovery. This section also describes the formats of the exception stack frames. For details on floating-point exceptions refer to **Section 6 Floating-Point Unit**.

8.1 EXCEPTION PROCESSING OVERVIEW

Exception processing is the transition from the normal processing of a program to the processing required for any special internal or external condition that preempts normal processing. External conditions that cause exceptions are interrupts from external devices, bus errors, and resets. Internal conditions that cause exceptions are instructions, address errors, and tracing. For example, the TRAP, TRAPcc, CHK, RTE, DIV, and FDIV instructions can generate exceptions as part of their normal execution. In addition, illegal instructions, unimplemented integer instructions, unimplemented effective addresses, unimplemented floating-point instructions and data types, and privilege violations cause exceptions. Exception processing uses an exception vector table and an exception stack frame. The following paragraphs describe the vector table and a generalized exception stack frame.

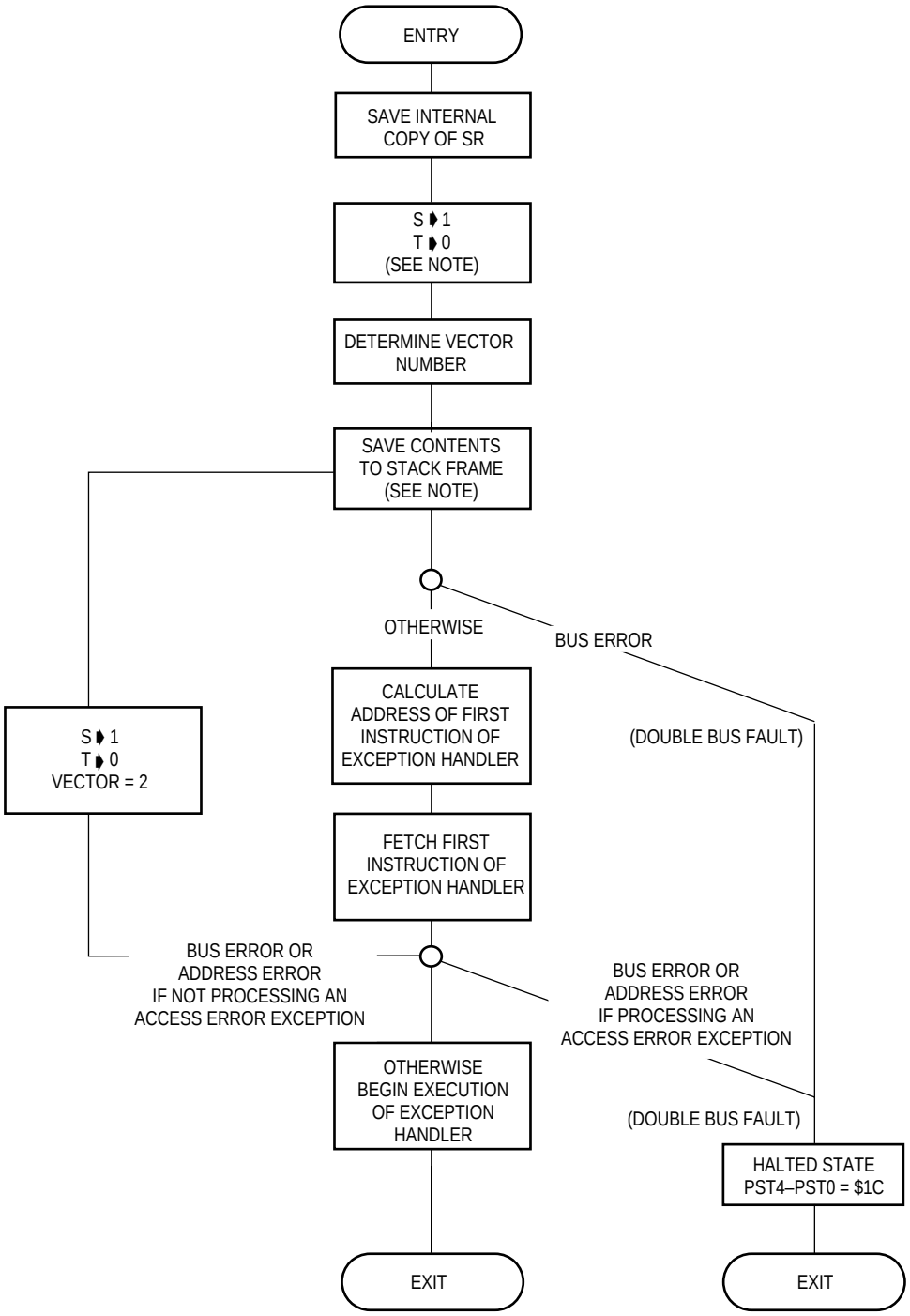
The MC68060 uses a restart exception processing model. Exceptions are recognized at the execution stage of the operand execution pipeline (OEP) and force later instructions that have not yet reached that stage to be aborted.

Instructions that cannot be interrupted, such as those that generate locked bus transfers or access noncachable precise pages, are allowed to complete before exception processing begins, unless an access error prevents this instruction from completing.

Exception processing occurs in four functional steps. However, all individual bus cycles associated with exception processing (vector acquisition, stacking, etc.) are not guaranteed to occur in the order in which they are described in this section. Figure 8-1 illustrates a general flowchart for the steps taken by the processor during exception processing.

During the first step, the processor makes an internal copy of the status register (SR). Then the processor changes to the supervisor mode by setting the S-bit and inhibits tracing of the exception handler by clearing the T-bit in the SR. For the reset and interrupt exceptions, the processor also updates the interrupt priority mask in the SR.

During the second step, the processor determines the vector number for the exception. For interrupts, the processor performs an interrupt acknowledge bus cycle to obtain the vector



NOTE: THESE BLOCKS VARY FOR RESET AND INTERRUPT EXCEPTIONS.

Figure 8-1. General Exception Processing Flowchart

number. For all other exceptions, internal logic provides the vector number. This vector number is used in the last step to calculate the address of the exception vector. Throughout this section, vector numbers are given in decimal notation.

8.4.5 Recovering from an Access Error

The access error exception handler can identify the cause of the fault by examining the FSLW. Unlike earlier processors, the MC68060 provides all the information needed to identify the fault by examining the FSLW. Note that this section does not discuss the use of the SEE (software emulation error) bit nor does it provide the procedure needed to support the M68060SP misaligned CAS and CAS2 emulation code. Refer to **Appendix C MC68060 Software Package** for details of how access error recovery is affected by the M68060SP.

The first step to recovering from an access error is for the exception handler to determine whether or not a branch prediction error has occurred. See **8.4.7 Branch Prediction Error** for details on how a branch prediction error occurs. If the BPE bit in the FSLW is set, flush the branch cache and continue with normal access error handling. If no other faults are indicated, then execute an RTE and continue normal operations.

The second step for the handler is to determine whether or not the access error is recoverable. In general, bus errors ($\overline{\text{TEA}}$ $\overline{\text{Asserted}}$) on write cycles must be avoided. Refer to **8.4.6 Bus Errors and Pending Memory Writes** for further details of bus errors and pending memory writes. In summary, check for any of the following nonrecoverable write cases:

- PBE = 1 (push buffer bus error)
- SBE = 1 (store buffer bus error)
- RW = 11, IO = 0, MA=1 (bus error on misaligned read-modify-write)
- RW = 01, for a MOVE <ea>, <ea> in which the destination operand writes over the source operand.

For these nonrecoverable write cases, the write reference has been lost and it is up to the system software designer to determine the next course of action. Probably the most prudent course of action is to discontinue the user program and enter a known supervisor state.

The third step is to handle the transparent translation access error cases. This is indicated by TTR=1. All of these cases are recoverable as long as step two from above has been taken out. At this point, the access error may be caused by the following errors, which are mutually exclusive.

- SP = 1 (supervisor protection violation detected by one of the four TTRs)
- WP = 1 (write protection violation detected by one of the four TTRs)
- RE = 1 (bus error on read)
- WE = 1 (bus error on write)

For the SP = 1 or WP = 1 cases, it is possible to modify the transparent translation descriptor to allow the access to occur once the instruction is restarted.

For the RE = 1 or WE = 1 cases, unless the cause of the bus error is removed, when the instruction is restarted, the access error handler is re-entered, possibly resulting in an infinite loop.

Table 10-22. Status Register (SR) Instruction Execution Times

Instruction	Execution Time
ANDI to SR	12(0/0)
EORI to SR	12(0/0)
MOVE from SR	1(0/1)[1]
MOVE to SR	12(1/0)[1]
ORI to SR	5(0/0)

[1] For these instructions, add the effective address calculation time.

Table 10-23. MOVES Execution Times

MOVES Function	Destination							
	Size	(An)	(An)+	–(An)	(d16,An)	(d8,An,Xi*SF)	(bd,An,Xi*SF)[1]	(xxx).WL
Source<SFC> -> Rn	Byte, Word	1(1/0)	1(1/0)	1(1/0)	1(1/0)	2(1/0)	3(1/0)	2(1/0)
“	Long	1(1/0)	1(1/0)	1(1/0)	1(1/0)	2(1/0)	3(1/0)	2(1/0)
Rn -> Dest <DFC>	Byte, Word	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	3(0/1)	2(0/1)
“	Long	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	3(0/1)	2(0/1)

[1] Add 2(1/0) cycles to the (bd,An,Xi*SF) time for a memory indirect address.

Table 10-24. Miscellaneous Instruction Execution Times

Instruction	Size	Register	Memory	Reg -> Dest	Source -> Reg
ANDI to CCR	Byte	1(0/0)	—	—	—
CHK	Word	2(0/0)	2(1/0) [1]	—	—
“	Long	2(0/0)	2(1/0) [1]	—	—
CINVA	—	—	<=17(0/0)	—	—
CINVL	—	—	<=18(0/0)	—	—
CINVP	—	—	<=274(0/0)	—	—
CPUSHA	—	—	<=5394(0/512)[2]	—	—
CPUSHL	—	—	<=26(0/1)[2]	—	—
CPUSHP	—	—	<=2838(0/256)[2]	—	—
EORI to CCR	Byte	1(0/0)	—	—	—
EXG	Long	1(0/0)	—	—	—
EXT	Word	1(0/0)	—	—	—
“	Long	1(0/0)	—	—	—
EXTB	Long	1(0/0)	—	—	—
LINK	Word	2(0/1)	—	—	—
“	Long	2(0/1)	—	—	—
LPSTOP	Word	15(0/1)	—	—	—
MOVE from CCR	Word	1(0/0)	1(0/1)[1]	—	—
MOVE to CCR	Word	1(0/0)	1(1/0)[1]	—	—
MOVE from USP	Long	1(0/0)	—	—	—
MOVE to USP	Long	2(0/0)	—	—	—
MOVEC (SFC,DFC, USP,VBR,PCR)	Long	—	—	12(0/0)	11(0/0)
MOVEC (CACR,TC, TTR,BUSCR,URP,SRP)	Long	—	—	15(0/0)	14(0/0)
NOP	—	9(0/0)	—	—	—
ORI to CCR	Byte	1(0/0)	—	—	—
PACK	—	2(0/0)	2(1/1)	—	—

Table 10-25. Floating-Point Instruction Execution Times (Continued)

Instruction	Effective Address, <ea>									
	FPn	Dn	(An)	(An)+	–(An)	(d16,An) (d16,PC)	(d8,An,Xi∗SF) (d8,PC,Xi∗SF)	(bd,An,XI∗SF) (bd,PC,XI∗SF)	(xxx).WL	#<imm>
FSGLDIV	37(0/0)	39(0/0)	37(1/0)	37(1/0)	37(1/0)	37(1/0)	38(1/0)	39(1/0)	38(1/0)	38(0/0)
FSGLMUL	3(0/0)	5(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	5(1/0)	4(1/0)	4(0/0)
FMOVEM ,FPx *	—	—	1+3n (3n/0)	1+3n (3n/0)	—	1+3n(3n/0)	2+3n(3n/0)	3+3n(3n/0)	2+3n(3n/0)	—
FMOVEM FPy, *	—	—	1+3n (0/3n)	—	1+3n (0/3n)	1+3n(0/3n)	2+3n(0/3n)	3+3n(0/3n)	2+3n(0/3n)	—
FMUL	3(0/0)	5(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	5(1/0)	4(1/0)	4(0/0)
FDMUL	3(0/0)	5(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	5(1/0)	4(1/0)	4(0/0)
FSMUL	3(0/0)	5(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	5(1/0)	4(1/0)	4(0/0)
FNEG	1(0/0)	3(0/0)	1(1/0)	1(1/0)	1(1/0)	1(1/0)	2(1/0)	3(1/0)	2(1/0)	2(0/0)
FDNEG	1(0/0)	3(0/0)	1(1/0)	1(1/0)	1(1/0)	1(1/0)	2(1/0)	3(1/0)	2(1/0)	2(0/0)
FSNEG	1(0/0)	3(0/0)	1(1/0)	1(1/0)	1(1/0)	1(1/0)	2(1/0)	3(1/0)	2(1/0)	2(0/0)
FSUB	3(0/0)	5(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	5(1/0)	4(1/0)	3(0/0)
FDSUB	3(0/0)	5(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	5(1/0)	4(1/0)	3(0/0)
FSSUB	3(0/0)	5(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	5(1/0)	4(1/0)	3(0/0)
FTST	1(0/0)	3(0/0)	1(1/0)	1(1/0)	1(1/0)	1(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)
FSQRT	68(0/0)	70(0/0)	68(1/0)	68(1/0)	68(1/0)	68(1/0)	69(1/0)	70(1/0)	69(1/0)	69(0/0)
FSSQRT	68(0/0)	70(0/0)	68(1/0)	68(1/0)	68(1/0)	68(1/0)	69(1/0)	70(1/0)	69(1/0)	69(0/0)
FDSQRT	68(0/0)	70(0/0)	68(1/0)	68(1/0)	68(1/0)	68(1/0)	69(1/0)	70(1/0)	69(1/0)	69(0/0)
FSAVE	—	—	3(0/3)	—	—	—	—	—	—	—
FRE-STORE	—	—	6(3/0)	—	—	—	—	—	—	—
FMOVEM ,FPxR	—	—	7(n/0)	—	—	—	—	—	—	—
FMOVEM FPxR,	—	—	5(0/n)	—	—	—	—	—	—	—

NOTES:

“n” is the number of registers being moved.

For all FPU operations, if the external operand format is byte, word, or long, add three cycles to the execution time.

*For all FPU operations except FMOVEM, if the external operand format is extended precision, add two cycles to the execution time.

Add 2(1/0) cycles to the (bd,An,Xi*SF) time for a memory indirect address.

Add 1(0/0) cycle if the <ea> specifies a double precision immediate operand.

gal instruction exception is taken. This instruction must be removed from existing MC68040 software since it is not emulated in the M68060SP.

The MC68060 compensates for the lack of these instructions by providing extensive information in the FSLW in the access error stack frame. In addition, a new instruction, PLPA, is added to translate a logical to physical address by initiating a table search. This instruction may be used to provide most of the function of the PTEST instruction. As with the PTEST instruction, PLPA loads the valid page descriptor into the ATC when the table search it initiates executes successfully.

If it is absolutely necessary to emulate the PTEST and the MOVEC of the MMUSR, Motorola provides assembly source code for these instructions in the bulletin board (see **C.5.4 AESOP Electronic Bulletin Board** for bulletin board details). The source code is provided as-is and is only a rough approximation of these instructions and may need customizing. No documentation is provided other than what is available in the source code.

11.1.2.3 CONTEXT SWITCH INTERRUPT HANDLERS. Context switch interrupt handlers that use the same virtual address to map into multiple physical address locations must flush the branch cache via the MOVEC to CACR instruction. The reason for this is that the branch cache is a logical cache and not a physical cache. For systems that transparently translate logical addresses to physical addresses, the branch cache need not be flushed.

In multiprocessor systems, care must be taken so that saved contexts generated by an MC68040-based node not be restored into an MC68060-based node, or vice-versa. The floating-point frames are different between an MC68040 and MC68060; incorrect swapping of contexts may cause format errors to be incurred.

If the context switch interrupt handler uses a nonmaskable interrupt (level 7), CAS (misaligned operands) or CAS2 instruction emulation may result in data corruption. There is no good workaround except by either avoid using the level 7 interrupt for context switching, or by using external hardware to block the interrupt lines from reporting an interrupt whenever $\overline{\text{LOCK}}$ is asserted.

11.1.2.4 TRACE HANDLERS. The MC68060 does not implement “trace on change of flow”. Debug software that rely on this feature must take this into consideration. When a change of flow trace encoding is encountered, the processor does not trace.

11.1.2.5 I/O DEVICE DRIVER SOFTWARE. The MC68060, like the MC68040, has a restart model, and device drivers that have been written for the MC68040 probably do not need any modification in device driver software when porting to the MC68060; however there are a few issues to consider.

The cache mode (CM) encoding on the TTRs and the page descriptors is different between the MC68060 and MC68040. The MC68060 executes reads and writes in strict program order, and therefore, whenever the CM bits indicate either a noncachable precise or noncachable imprecise, the accesses are serialized. Areas that are marked cache-inhibited serialized for I/O devices should not be affected adversely by the cache mode change. Otherwise, the TTR format and the page descriptor formats have not changed for the MC68060.

M68060FPSP SNAN and OPERR exception handlers, _fpsp_snan and _fpsp_operr, will be provided for SNAN and OPERR enabled exceptions for the following reasons:

- For opclass two pre-instruction exceptions using a single or double source format with an infinity, denorm, NAN, or zero source operand, the processor does not create the correct extended-precision value for the FSAVE frame. The MC68060FPSP handlers convert the value in the FSAVE frame to extended-precision format before passing control to the user enabled SNAN or OPERR exception handlers (_real_{snan,operr}). No parameters are passed to the user enabled SNAN or OPERR exception handlers from the M68060FPSP package since the package provides the illusion that it never existed.
- For opclass three post-instruction exceptions, the processor does not store the default result to the destination memory or integer data register before taking the enabled exception. The MC68881/882 stored the default result in this scenario. Therefore, to maintain compatibility, the M68060FPSP SNAN and OPERR exception handlers calculate and store the default result before passing control to the user enabled SNAN and OPERR exception handlers (_real_{snan,operr}). No parameters are passed to the user SNAN or OPERR exception handlers since the M68060FPSP provides the illusion that it never existed.

A simple pseudo-code diagram for the SNAN and OPERR handlers is provided in the code sequence shown in Figure C-8.

```
_fpsp_{snan,operr}() {
 if ((opclass == 0) || (opclass == 2)) {
       /*
        * if src operand is a sgl or dbl
        * zero,NAN,denorm, or infinity,
        * fix operand in FSAVE frame.
        */
       fix_FSAVE_op();

       bra.l _real_{snan,operr}();
 }
 else {/* opclass 3 */
       /*
        * save default result to memory
        * or integer register file.
        */
       save_default_result();

       bra.l _real_{snan,operr}();
```

Figure C-8. SNAN/OPERR Exception Handler Pseudo-Code

C.3.2.3.3 Inexact Exception. Opclass zero and two exception instructions taking the inexact exception cause pre-instruction exceptions, and opclass three instructions cause post-instruction inexact exceptions. The processor takes exception vector number forty-nine for the inexact exception. The FSAVE frame for the exception is valid and contains the source operand converted to extended precision.

The inexact exception is a maskable exception on the MC68060 for the trap-disabled case. The floating-point hardware produces the correct result when the inexact exception enable