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	66MHz
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	304-LBGA Exposed Pad
Supplier Device Package	304-TBGA (31x31)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68ec060zu66

Appendix C
MC68060 Software Package

Appendix D
MC68060 Instructions

for one full BCLK cycle and then three-stated one BCLK cycle after the address bus is idled. If LOCKE was already negated in the BCLK cycle in which the MC68060 relinquishes the bus, it will be three-stated in the same BCLK cycle the address bus is idled.

LOCKE is provided to help make the MC68060 bus compatible with the MC68040-style bus protocol; however, for new designs, external bus arbitration logic can be simplified with the use of BGR instead of LOCKE.

Do not use LOCKE. The LOCKE protocol breaks the integrity of the locked read-modify-write sequence if it is possible to retry the last write of a read-modify-write operation. The reason is that when LOCKE is asserted, a bus arbiter can grant the bus to an alternate master when the current bus cycle is finished (before the retry is attempted). The bus is arbitrated away, the last write's retry is deferred until the bus is returned to the processor. In the meantime, the alternate master can access the same location where the write should have taken place. Hence, the integrity of the locked read-modify-write sequence is compromised in this situation.

2.3.9 Cache Inhibit Out (CIOUT)

When asserted, this three-state output indicates that the MC68060 will not cache the current bus information in its internal caches. Refer to **Section 4 Memory Management Unit** for more information on CIOUT function. When the MC68060 is not the bus master, the CIOUT signal is placed in a high-impedance state.

2.3.10 Byte Select Lines (BS3–BS0)

These three-state outputs indicate which bytes within a long-word transfer are being selected and which bytes of the data bus will be used for the transfer. BS0 refers to D31–D24, BS1 refers to D23–D16, BS2 refers to D15–D8, and BS3 refers to D7–D0. These signals are generated to provide byte data select signals which are decoded from the SIZx, A1, and A0 signals as shown in Table 2-6. These signals are placed in a high-impedance state when the MC68060 is not the bus master.

Table 2-6. Data Bus Byte Select Signals

Transfer Size	SIZ1	SIZ0	A1	A0	BS0	BS1	BS2	BS3
					D31–D24	D23–D16	D15–D8	D7–D0
Byte	0	1	0	0	0	1	1	1
Byte	0	1	0	1	1	0	1	1
Byte	0	1	1	0	1	1	0	1
Byte	0	1	1	1	1	1	1	0
Word	1	0	0	0	0	0	1	1
Word	1	0	1	0	1	1	0	0
Long Word	0	0	x	x	0	0	0	0
Line	1	1	x	x	0	0	0	0

2.4 MASTER TRANSFER CONTROL SIGNALS

The following signals provide control functions for bus cycles when the MC68060 is the bus master. Refer to **Section 7 Bus Operation** for detailed information about the relationship of the bus cycle control signals to bus operation.

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

data cache is disabled for the second half of the operand. Internal accesses always bypass the instruction and data caches while $\overline{\text{CDIS}}$ is recognized, and the contents of the caches are unchanged. Disabling the caches with $\overline{\text{CDIS}}$ does not affect snoop operations. $\overline{\text{CDIS}}$ is intended primarily for use by in-circuit emulators to allow swapping between the tags and emulator memories.

The privileged CINV and CPUSH instructions support cache management, by selectively pushing and/or invalidating an individual cache line, a full page, or an entire cache, for either or both instruction and data caches. CINV allows selective invalidation of cache entries. The CPUSH instruction will either push and invalidate all matching lines, or push and leave the line valid, depending on the state of the DPI bit of the CACR register. (Note that only CPUSH instructions which specify the data cache are affected by the DPI bit. Since the instruction cache cannot have dirty data, a CPUSH specifying the instruction cache is interpreted as a CINV instruction.) Because of the size of the caches, pushing pages or an entire cache may incur a significant time penalty. Therefore, the CPUSH instruction may be interrupted to avoid large interrupt latencies. The state of the $\overline{\text{CDIS}}$ signal or the cache enable or no-allocate bits in the CACR does not affect the operation of CINV and CPUSH.

5.4 CACHING MODES

Every cache access has an associated caching mode from the MMU that determines how the cache handles the access. An access can be cachable in either the writethrough or copyback modes, or it can be cache inhibited in precise or imprecise modes. The CM field (from the transparent translation register (TTR) or MMU translation table page descriptor) corresponding to the logical address of the access normally specifies, on a page-by-page basis, one of these caching modes. When the cache is enabled and memory management is disabled, the default caching mode is writethrough.

The MMU provides the cache mode user page attributes (UPAx) and write protection for each access. This information may come from a TTR which matches or from the MMU translation tables via the ATC. If both the TTR and the ATC match the access, the TTR provides the information. If the paging MMU is disabled (TCR bit clear) and neither TTR matches, then the cache mode, UPAx, and write protection will be that which is specified in the default bits of the TCR. After reset, the defaults are writethrough cache mode, UPAx bits are zero, and all addresses may be written.

The TTRs and MMUs allow the defaults to be overridden. In addition, some instructions and integer unit operations perform data accesses that have an implicit caching mode associated with them. The following paragraphs discuss the different caching accesses and their related cache modes.

5.4.1 Cachable Accesses

If the CM field of a page descriptor, TTR, or default field of the TCR indicates writethrough or copyback, then the access is cachable. A read access to a writethrough or copyback page is read from the cache if matching data is found. Otherwise, the data is read from memory and used to update the cache. Since instruction cache accesses are always reads, the selection of writethrough or copyback modes do not affect them. The following paragraphs describe the writethrough and copyback modes in detail.

The user BSUN exception handler must execute an FSAVE as its first floating-point instruction. FSAVE allows other floating-point instructions to execute without reporting the BSUN exception again, although none of the state frame values are useful in the execution of the user BSUN exception handler. The BSUN exception is unique in that the exception is taken before the conditional predicate is evaluated. If the user BSUN exception handler does not set the PC to the instruction following the one that caused BSUN exception when returning, the exception is executed again. Therefore, it is the responsibility of the user BSUN exception handler to prevent the conditional instruction from taking the BSUN exception again. There are four ways to prevent taking the exception again:

1. Incrementing the stored PC in the stack bypasses the conditional instruction. This technique applies to situations where a fall-through is desired. Note that accurate calculation of the PC increment requires detailed knowledge of the size of the conditional instruction being bypassed.
2. Clearing the NAN bit prevents the exception from being taken again. However, this alone cannot deterministically control the result's indication (true or false) that would be returned when the conditional instruction re-executes.
3. Disabling the BSUN bit also prevents the exception from being taken again. Like the second method, this method cannot control the result indication (true or false) that would be returned when the conditional instruction re-executes.
4. Examining the conditional predicate and setting the FPCC NAN bit accordingly prevents the exception from being taken again. This technique gives the most control since it is possible to predetermine the direction of program flow. Bit 7 of the F-line operation word indicates where the conditional predicate is located. If bit 7 is set, the conditional predicate is the lower six bits of the F-line operation word. Otherwise, the conditional predicate is the lower six bits of the instruction word, which immediately follows the F-line operation word. Using the conditional predicate and the table for IEEE nonaware test in **6.4.2 Conditional Testing**, the condition codes can be set to return a known result indication when the conditional instruction is re-executed.

Prior to exiting the user BSUN exception handler, the user exception handler discards the floating-point state frame before executing the RTE to return to normal program flow.

6.6.2 Signaling Not-a-Number (SNAN)

An SNAN is used as an escape mechanism for a user-defined, non-IEEE data type. The processor never creates an SNAN as a result of an operation; a NAN created by an operand error exception is always a nonsignaling NAN. When an operand is an SNAN involved in an arithmetic instruction, the SNAN bit is set in the FPSR EXC byte. Since the FMOVEM, FMOVE FPCR, and FSAVE instructions do not modify the status bits, they cannot generate exceptions. Therefore, these instructions are useful for manipulating SNANs.

6.6.2.1 TRAP DISABLED RESULTS (FPCR SNAN BIT CLEARED). If the destination data format is S, D, X, or P, then the most significant bit of the fraction is set to one and the resulting nonsignaling NAN is transferred to the destination. No bits other than the SNAN bit of the NAN are modified, although the input NAN is truncated if necessary. If the destination data format is B, W, or L, then the 8, 16, or 32 most significant bits of the SNAN significand, with the SNAN bit set, are written to the destination.

FPCR exception enable byte is set and the corresponding INEX bit in the FPSR EXC byte is also set).

6.6.4.1 TRAP DISABLED RESULTS (FPCR OVFL BIT CLEARED). The values defined in Table 6-13 are stored in the destination based on the rounding mode defined in the FPCR MODE byte. The result is rounded according to the rounding precision defined in the FPCR MODE byte if the destination is a floating-point data register. If the destination is in memory or an integer data register, then the rounding precision in the FPCR MODE byte is ignored, and the given destination format defines the rounding precision. If the instruction has a forced rounding precision (e.g., FSADD, FDMUL), the instruction defines the rounding precision.

Table 6-13. 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.

6.6.4.2 TRAP ENABLED RESULTS (FPCR OVFL BIT SET). The result stored in the destination is the same as the result stored when the trap is disabled before control is passed to the user OVFL handler. For an FMOVE OUT instruction, the operand is stored in memory or integer data register, and then control is passed to the user OVFL handler as a post-instruction exception. If the destination is a floating-point data register, control is passed to the user OVFL handler as a pre-instruction exception when the next floating-point operation is encountered.

The user OVFL handler must execute an FSAVE instruction as the first floating-point instruction to prevent further exceptions from being taken. The address of the instruction that causes the overflow is available to the user OVFL handler in the FPIAR. By examining the instruction, the user OVFL handler can determine the arithmetic operation type and destination location. The exception operand is stored in the floating-point state frame (generated by the FSAVE). When an overflow occurs, the exception operand is defined differently for various destination types:

1. FMOVE OUT instruction (memory or integer data register destination)—the value in the exception operand is the intermediate result mantissa rounded to the destination precision, with a 15-bit exponent biased as a normal extended-precision number. In the case of a memory destination, the evaluated effective address of the operand is available in the integer stack frame format $3. This allows the user OVFL handler to overwrite the default result, if necessary, without recalculating the effective address.
2. Floating-point data register destination—the value in the exception operand is the intermediate result rounded to extended precision, with an exponent bias of $3FFF–$6000 rather than $3FFF. The additional bias of –$6000 is used so that it is possible to represent the larger exponent in a 15-bit format.

In addition to normal overflow, the exponential instructions (e^x, 10^x, 2^x, SINH, COSH, and FSCALE) may generate results that grossly overflow the 16-bit exponent of the internal

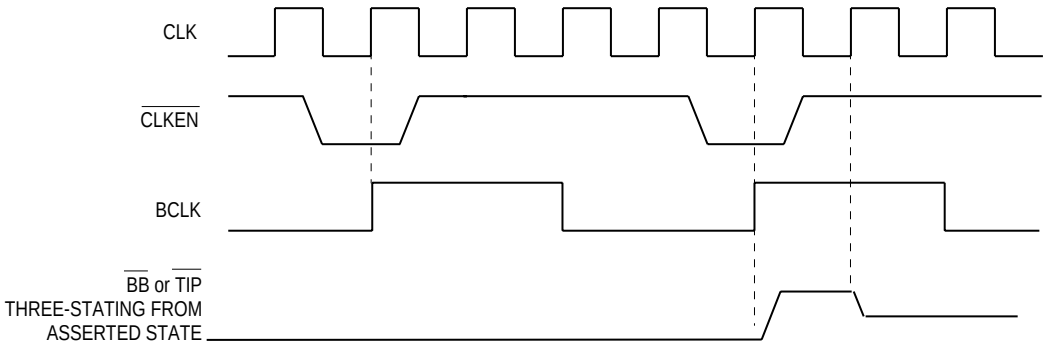


Figure 7-4. Quarter-Speed Clock

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

of BCLK represent the rising edges of CLK in which CLKEN is asserted. However, there are cases in which the BCLK concept does not apply.

The BCLK concept does not apply to the IPLx and RSTI input signals. These inputs are sampled every CLK edge. The processor status (PSTx), RSTO, and IPEND outputs do not follow the BCLK concept, either, since these outputs can change on any CLK rising edge, regardless of CLKEN. The BB and TIP signals generally follow the BCLK concept except when these signals are already driven asserted by the processor and then three-stated. This occurs when the bus is arbitrated away from the processor immediately after an active bus cycle. These outputs are actively negated for one CLK period before three-stating. Figure 7-2, Figure 7-3, and Figure 7-4 illustrate the behavior of BB and TIP in the case mentioned. The BB signal is not recommended for use in full-speed bus designs since bus contention is possible when tied to alternate masters' BB pins.

Other implementations of CLKEN are not supported.

7.3 ACKNOWLEDGE TERMINATION IGNORE STATE CAPABILITY

The MC68060 provides the capability to ignore termination acknowledgments to assist in system designs. Independent ignore state counters for read and write, primary (initial) transfer, and secondary (burst) transfer are used during bus cycles to determine which BCLK rising edges transfer acknowledge termination signals should be ignored or sampled.

This special mode is selected during a reset operation. Please refer to **7.14 Special Modes of Operation** for details on how to enable this mode.

7.4 BUS CONTROL REGISTER

The bus control register (BUSCR) is accessed via the MOVEC instruction. Its main purpose is to provide a way to control the external LOCK and LOCKE signals in software. This feature is essential in emulating the CAS2 instruction and in providing a means to control bus arbitration activity during critical code segments. Figure 7-5 shows the BUSCR format.

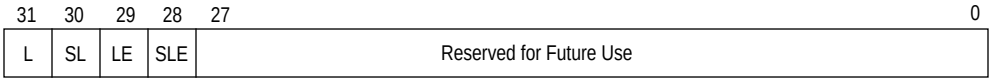


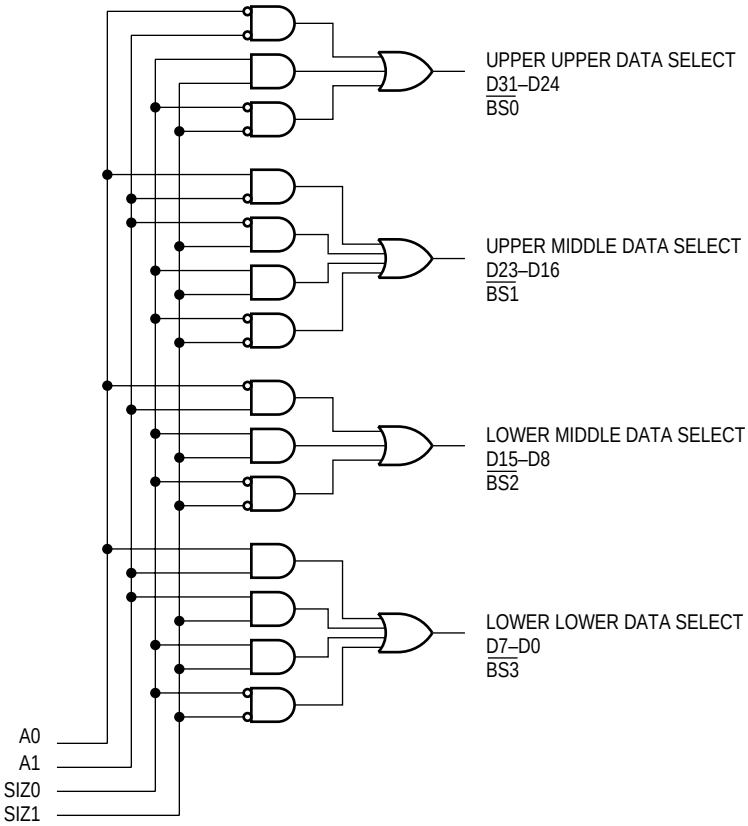
Figure 7-5. Bus Control Register Format

L—Lock Bit

0 = Negate external LOCK signal.
1 = Assert external LOCK signal.

SL—Shadow Copy, Lock Bit

0 = LOCK negated sequence at time of exception.
1 = LOCK asserted at time of exception.




```
PAL16L8
U1
MC68060 Byte Data Select Generation.
A0 A1 SIZ0 SIZ1 NC NC NC NC NC GND NC UUD UMD LMD LLD
NC NC NC NC VCC

/UUD =   /A0 * /A1          ; directly addressed, any size
           + /SIZ1 * /SIZ0  ; enable every byte for long-word size
           + SIZ1 * SIZ0    ; enable every byte for line size
/UMD =   A0 * /A1           ; directly addressed, any size
           + /A1 * /SIZ1    ; word aligned, size is word or line
           + SIZ1 * SIZ0    ; enable every byte for long-word size
           + /SIZ1 * /SIZ0  ; enable every byte for line size
/LMD =   /A0 * /A1          ; directly addressed, any size
           + /SIZ1 * /SIZ0  ; enable every byte for long-word size
           + SIZ1 * SIZ0    ; enable every byte for line size
/LLD =     A0 * /A1         ; directly addressed, any size
           + /A1 * /SIZ1    ; word aligned, word or line size
           + SIZ1 * SIZ0    ; enable every byte for long-word size
           + /SIZ1 * /SIZ0  ; enable every byte for line size
```

Figure 7-8. Byte Select Signal Generation and PAL Equation

asserted. The registered data and the value of $\overline{TCI}$ are then passed to the appropriate memory unit.

If $\overline{TBI}$ was negated with the assertion of $\overline{TA}$, the processor continues the cycle with C3. Otherwise, if $\overline{TBI}$ was asserted, the line transfer is burst inhibited, and the processor reads the remaining three long words using long-word read bus cycles. The processor increments A3 and A2 for each read, and the new address is placed on the address bus for each bus cycle. Refer to **7.7.1 Byte, Word, and Long-Word Read Transfer Cycles** for information on long-word reads. If no wait states are generated, a burst-inhibited line read completes in eight clocks instead of the five required for a burst read.

Clock 3 (C3)

The processor holds the address and transfer attribute signals constant during C3 if $\overline{CLA}$ is negated. The selected device must either increment A3 and A2 to reference the next long word to transfer, place the data on the data bus, and assert $\overline{TA}$, or alteratively assert the $\overline{CLA}$ input to request the processor to increment A3 and A2. Refer to **7.7.7 Using CLA to Increment A3 and A2** for details on $\overline{CLA}$ operation.

As in the description of C2, using acknowledge termination ignore state capability, the processor ignores any termination signal, such as $\overline{TA}$, until a user-programmable number of BCLK edges has expired. And, as in the description in C2, $\overline{SAS}$ indicates the first BCLK rising edge in which acknowledge termination signals become significant. If this mode is disabled, $\overline{SAS}$ stays asserted in C3 to indicate that the processor will sample $\overline{TA}$ immediately. Refer to **7.14.1 Acknowledge Termination Ignore State Capability** for details on this mode.

Assuming that the acknowledge termination ignore state capability is disabled, the processor samples the level of $\overline{TA}$ and registers the current value on the data bus at the end of C3. If $\overline{TA}$ is asserted, the transfer terminates and the second long word of data is passed to the appropriate memory unit. If $\overline{TA}$ is not recognized asserted at the end of C3, the processor ignores the latched data and inserts wait states instead of terminating the transfer. The processor continues to sample $\overline{TA}$ on successive rising edges of BCLK until it is recognized asserted. The registered data is then passed to the appropriate memory unit.

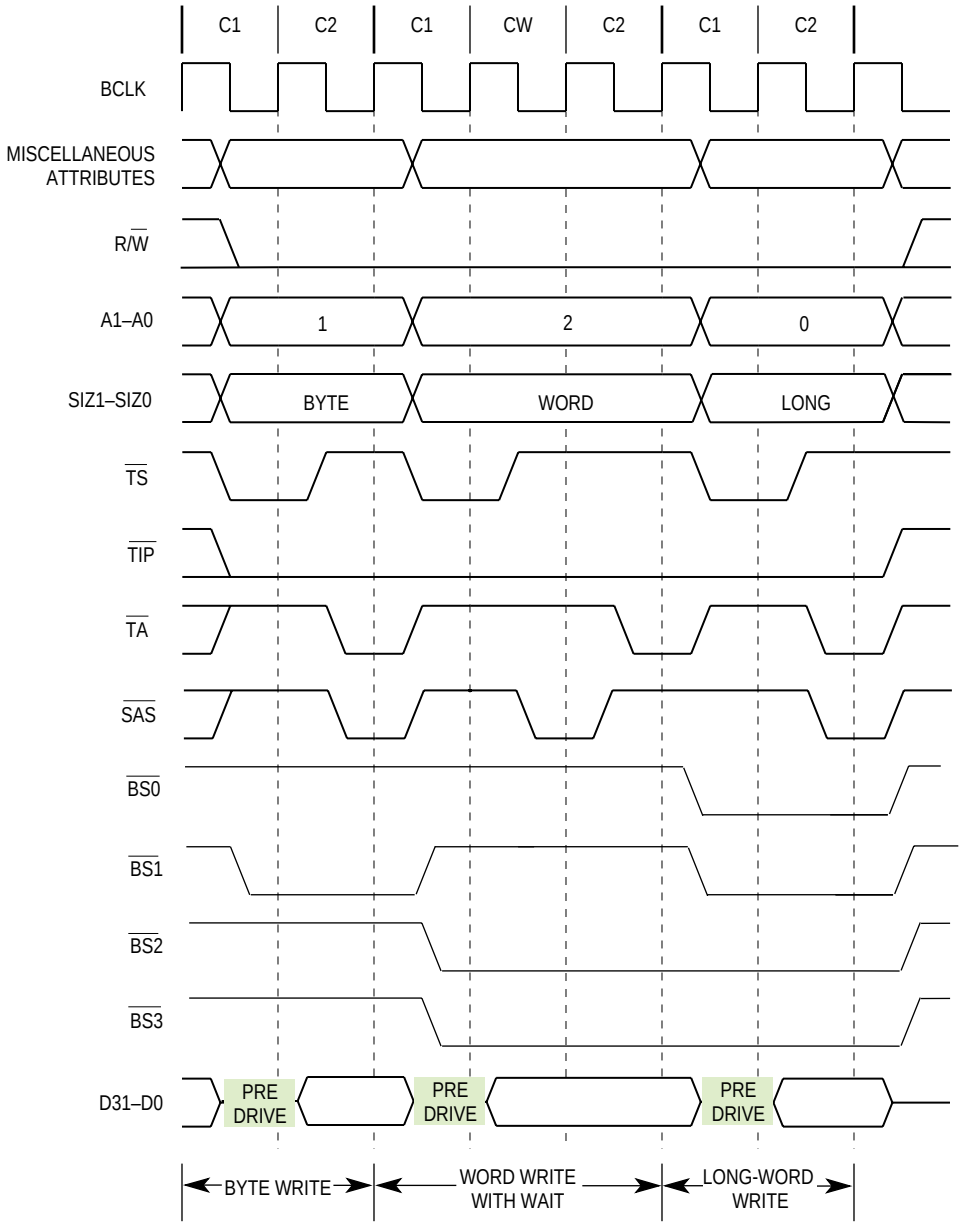
Clock 4 (C4)

This clock is identical to C3 except that once $\overline{TA}$ is recognized asserted, the registered value corresponds to the third long word of data for the burst.

Clock 5 (C5)

This clock is identical to C3 except that once $\overline{TA}$ is recognized, the registered value corresponds to the fourth long word of data for the burst. After the processor recognizes the last $\overline{TA}$ assertion and terminates the line read bus cycle, $\overline{TIP}$ remains asserted if the processor is ready to begin another bus cycle. Otherwise, the processor negates $\overline{TIP}$ during the next clock.

Figure 7-16 and Figure 7-17 illustrate a flowchart and functional timing diagram for a burst-inhibited line read bus cycle.



NOTE: It is assumed that the acknowledge termination ignore state capability is disabled.

Figure 7-19. Long-Word Write Bus Cycle Timing

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

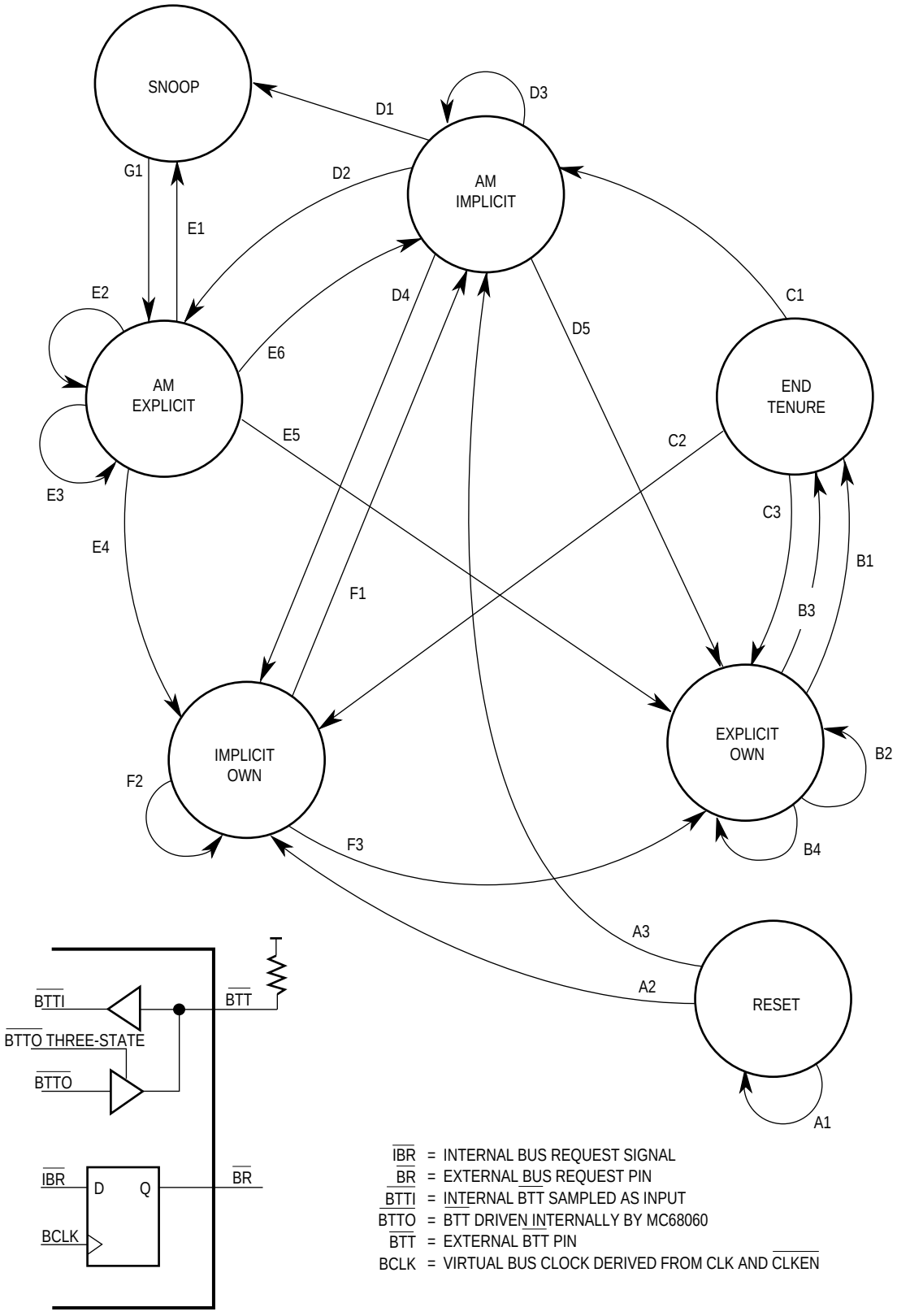
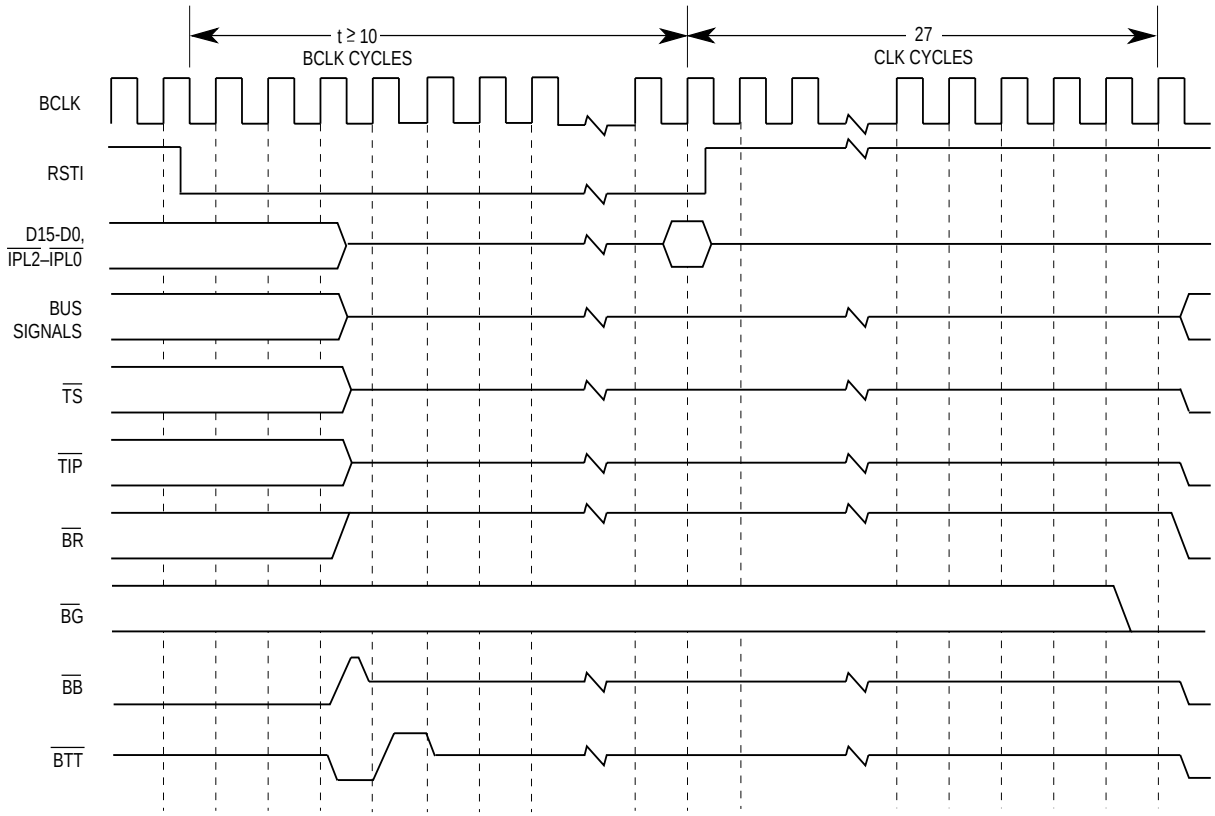


Figure 7-42. MC68060-Arbitration Protocol State Diagram

For processor resets after the initial power-on reset, RSTI should be asserted for at least ten BCLK periods. Figure 7-49 illustrates timings associated with a reset when the processor is executing bus cycles. BB and TIP are negated before transitioning to a three-state level.



NOTE: For the processor to reset begin bus cycles after reset, BG must be asserted, TS must be negated or pulled up. BTT must be asserted (or BTT transition from asserted to negated) eventually to indicate an end to the alternate master's tenure.

Figure 7-49. Normal Reset Timing

Resetting the processor causes any bus cycle in progress to terminate as if TEA had been asserted. In addition, the processor initializes registers appropriately for a reset exception. **Section 8 Exception Processing** describes reset exception processing. When a RESET bus operation instruction is executed, the processor drives the reset out (RSTO) signal for 512 CLK cycles. In this case, the processor can be used to reset external devices in a system, and the internal registers of the processor are unaffected. The external devices connected to the RSTO signal are reset at the completion of the RESET instruction. An RSTI signal that is asserted to the processor during execution of a RESET instruction immediately resets the processor and causes the RSTO signal to negate. RSTO can be logically ANDed with the external signal driving RSTI to derive a system reset signal that is asserted for both an external processor reset and execution of a RESET instruction.

7.14 SPECIAL MODES OF OPERATION

The MC68060 supports the following three operation modes, which are selectively enabled during processor reset and remain in effect until the next processor reset. Refer to **7.13 Reset Operation** for reset timing information. Table 7-10 summarizes the three special modes and associates them with the appropriate $\overline{\text{IPLx}}$ signal.

Table 7-10. Special Mode vs. $\overline{\text{IPLx}}$ Signals

Signal	Value During Reset Time	Action
$\overline{\text{IPL2}}$	Asserted	Extra Data Write Hold Mode Enabled
	Negated	Extra Data Write Hold Mode Disabled
$\overline{\text{IPL1}}$	Asserted	Native-MC68060 Acknowledge Termination Protocol
	Negated	MC68040 Acknowledge Termination Protocol
$\overline{\text{IPL0}}$	Asserted	Acknowledge Termination Ignore State Capability Enabled
	Negated	Acknowledge Termination Ignore State Capability Disabled

7.14.1 Acknowledge Termination Ignore State Capability

The MC68060 provides acknowledge termination ignore state capability to make high-frequency system design easier. This feature defines BCLK edges during which the acknowledge termination signals ($\overline{\text{TA}}$, $\overline{\text{TEA}}$, and $\overline{\text{TRA}}$) are ignored. This feature is enabled if $\overline{\text{IPL0}}$ is asserted during reset.

During reset, 16 bits of information (from D15–D0) are registered into the MC68060. These 16 bits define four values of four bits each. Two of the four values are used for read bus cycles; the other two values are used for write bus cycles. For the read bus cycle, the first value is the primary ignore state count value. The primary ignore state count value is used during the first long-word transfer of a line transfer cycle, or the only data transfer for byte, word, or long-word bus cycles. The second value is the secondary ignore state count value. The secondary ignore state count value is used during the next three long words for line transfer cycles, after the first long word has been transferred. Similarly, the two values of the write bus cycle are defined as a primary ignore state count value and a secondary ignore state count value, respectively. Figure 7-50 shows the assignment of the four data nibbles at reset.

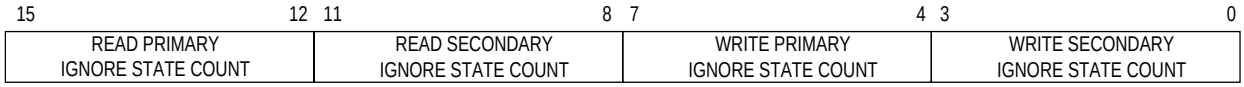


Figure 7-50. Data Bus Usage During Reset

At the beginning of a bus cycle, the appropriate primary ignore state count value is loaded into an internal counter. The counter decrements every BCLK rising edge. As long as the counter has a non-zero count value, the MC68060 ignores the acknowledge termination signals. Once the counter reaches zero, the MC68060 asserts $\overline{\text{SAS}}$ for one BCLK period and begins to sample the acknowledge termination signals and acts accordingly. For byte, word, or long-word transfers, the bus cycle ends when a valid termination is detected. For line transfer cycles after the first long-word transfer, the secondary ignore state count value is

- Instruction Trap
- Illegal and Unimplemented Instruction Exceptions
- Privilege Violation
- Trace
- Format Error
- Breakpoint Instruction
- Interrupt
- Reset

8.2.1 Access Error Exception

An access error exception occurs when a bus cycle is terminated with $\overline{\text{TEA}}$ ($\overline{\text{TA}}$ must be negated if in MC68040 acknowledge termination mode) asserted externally or an internal access error.

An external access error (bus error) occurs when external logic aborts a bus cycle and asserts the $\overline{\text{TEA}}$ input signal ($\overline{\text{TA}}$ must be negated if in MC68040 acknowledge termination mode).

A bus error on an operand write access always results in an access error exception, causing the processor to begin exception processing. However, the time of reporting this bus error is a function of the instruction type and/or memory mapping of the destination pages. For writes that are precise (this includes certain atomic instructions like TAS and CAS and references to pages marked noncachable precise), the occurrence of a bus error causes the pipeline to be aborted immediately and initiates exception processing. For writes that are imprecise (stored in push or store buffers or reference to pages marked noncachable imprecise), the actual bus cycle is decoupled from the instruction which generated the access. For these types of bus errors, the exception is taken, but the state of the processor may be advanced from the actual instruction which generated the write.

For operand read accesses generating non-line-sized references, a bus error causes the pipeline to be immediately aborted and initiates exception processing. This is also true if a bus error occurs on the first transfer of a line-sized transfer. For a bus error that occurs on the second, third, or fourth transfers of a line access, the line is not allocated in the cache and no exception is reported. If a subsequent instruction references another operand within the given line, another system bus cycle is generated and the bus error reported at that time (i.e., as the subsequent reference receives a bus error on its initial transfer) and the exception is then taken.

Bus errors that are signaled during instruction prefetches are deferred until the processor attempts to execute that instruction. At that time, the bus error is signaled and exception processing is initiated. If a bus error is encountered during an instruction prefetch cycle, but the corresponding instruction is never executed due to a change-of-flow in the instruction stream, the bus error is discarded.

Exception processing for illegal and unimplemented instructions is similar to that for instruction traps. When the processor has identified an illegal or unimplemented instruction, it initiates exception processing instead of attempting to execute the instruction. The processor copies the SR, enters the supervisor mode, and clears T-bit, disabling further tracing. The processor generates the vector number according to the exception type. The illegal or unimplemented instruction vector offset, current PC, and copy of the SR are saved on the supervisor stack. Instruction execution resumes at the address contained in the exception vector.

8.2.5 Privilege Violation Exception

To provide system security, certain instructions are privileged. An attempt to execute one of the following privileged instructions while in the user mode causes a privilege violation exception:

ANDI to SR	FSAVE	MOVEC	PLPA
CINV	MOVE from SR	MOVES	RESET
CPUSH	MOVE to SR	ORI to SR	RTE
EORI to SR	MOVE USP	PFLUSH	STOP
FRESTORE	LPSTOP		

Exception processing for privilege violations is similar to that for illegal instructions. When the processor identifies a privilege violation, it begins exception processing before executing the instruction. As illustrated in Figure 8-1, the processor copies the SR, enters the supervisor mode, and clears the T-bit. The processor generates vector number 8, saves the privilege violation vector offset, the current PC value, and the internal copy of the SR on the supervisor stack. The saved value of the PC is the logical address of the first word of the instruction that caused the privilege violation. Instruction execution resumes after the initial instruction is fetched from the address in the privilege violation exception vector.

8.2.6 Trace Exception

To aid in program development, the M68000 family includes an instruction-by-instruction tracing capability. In the trace mode, an instruction generates a trace exception after the instruction completes execution, allowing a debugging program to monitor execution of a program.

In general terms, a trace exception is an extension to the function of any traced instruction. The execution of a traced instruction is not complete until trace exception processing is complete. If an instruction does not complete due to an access error or address error exception, trace exception processing is deferred until after execution of the suspended instruction is resumed. If an interrupt is pending at the completion of an instruction, trace exception processing occurs before interrupt exception processing starts. If an instruction forces an exception as part of its normal execution, the forced exception processing occurs before the trace exception is processed.

The T-bit in the supervisor portion of the SR controls tracing. The state of the T-bit when an instruction begins execution determines whether the instruction generates a trace exception after the instruction completes.

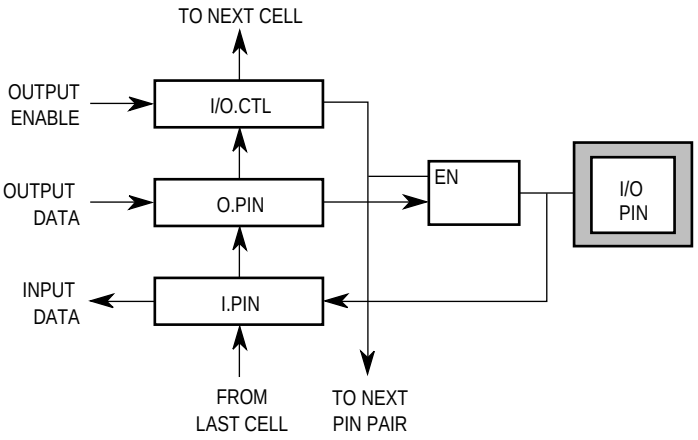
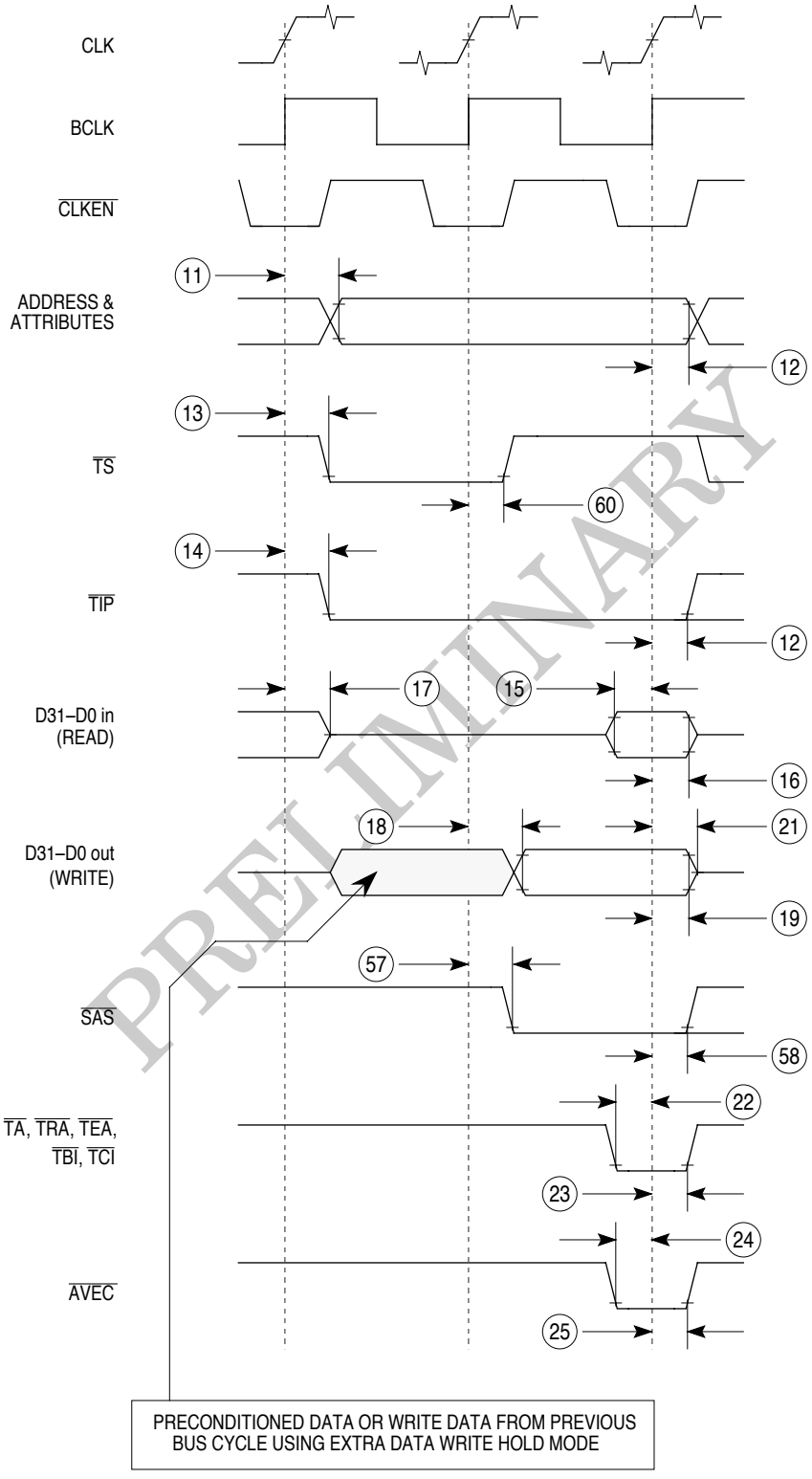


Figure 9-7. General Arrangement of Bidirectional Pin Cells

Table 9-3. Boundary Scan Bit Definitions

Bit	Cell Type	Pin/Cell Name	Pin Type
0	O.Pin	A31	I/O
1	I.Pin	A31	I/O
2	O.Pin	A30	I/O
3	I.Pin	A30	I/O
4	IO.Ctl	A31–A28 ena	—
5	O.Pin	A29	I/O
6	I.Pin	A29	I/O
7	O.Pin	A28	I/O
8	I.Pin	A28	I/O
9	O.Pin	A27	I/O
10	I.Pin	A27	I/O
11	O.Pin	A26	I/O
12	I.Pin	A26	I/O
13	IO.Ctl	A27–A24 ena	—
14	O.Pin	A25	I/O
15	I.Pin	A25	I/O
16	O.Pin	A24	I/O
17	I.Pin	A24	I/O
18	O.Pin	A23	I/O
19	I.Pin	A23	I/O
20	O.Pin	A22	I/O
21	I.Pin	A22	I/O
22	IO.Ctl	A23–A20 ena	—
23	O.Pin	A21	I/O
24	I.Pin	A21	I/O
25	O.Pin	A20	I/O
26	I.Pin	A20	I/O
27	O.Pin	A19	I/O
28	I.Pin	A19	I/O
29	O.Pin	A18	I/O
30	I.Pin	A18	I/O
31	IO.Ctl	A19–A16 ena	—



NOTE: Address and attributes refer to the following signals:
A31–A0, SIZ1, SIZ0, R/$\overline{W}$, TT1, TT0, TM2–TM0, TLN1, TLN0, UPA1, UPA0, $\overline{CIOUT}$, $\overline{BS3}$-$\overline{BS0}$

Figure 12-3. Read/Write Timing

Table D-3. Exception Vector Assignments for the M68000 Family

Vector Number(s)	Vector Offset (Hex)	Assignment
0	000	Reset Initial Interrupt Stack Pointer
1	004	Reset Initial Program Counter
2	008	Access Fault
3	00C	Address Error
4	010	Illegal Instruction
5	014	Integer Divide-by-Zero
6	018	CHK, CHK2 Instruction
7	01C	FTRAPcc, TRAPcc, TRAPV Instructions
8	020	Privilege Violation
9	024	Trace
10	028	Line 1010 Emulator (Unimplemented A-Line Opcode)
11	02C	Line 1111 Emulator (Unimplemented F-Line Opcode)
12	030	(Reserved)
13	034	Coprocessor Protocol Violation (Defined for MC68020 and MC68030)
14	038	Format Error
15	03C	Uninitialized Interrupt
16–23	040–05C	(Unassigned, Reserved)
24	060	Spurious Interrupt
25	064	Level 1 Interrupt Autovector
26	068	Level 2 Interrupt Autovector
27	06C	Level 3 Interrupt Autovector
28	070	Level 4 Interrupt Autovector
29	074	Level 5 Interrupt Autovector
30	078	Level 6 Interrupt Autovector
31	07C	Level 7 Interrupt Autovector
32–47	080–0BC	TRAP #0–15 Instruction Vectors
48	0C0	Floating-Point Branch or Set on Unordered Condition (Defined for MC68881, MC68882, MC68040, and MC68060)
49	0C4	Floating-Point Inexact Result (Defined for MC68881, MC68882, MC68040, and MC68060)
50	0C8	Floating-Point Divide-by-Zero (Defined for MC68881, MC68882, MC68040, and MC68060)
51	0CC	Floating-Point Underflow (Defined for MC68881, MC68882, MC68040, and MC68060)
52	0D0	Floating-Point Operand Error (Defined for MC68881, MC68882, MC68040, and MC68060)
53	0D4	Floating-Point Overflow (Defined for MC68881, MC68882, MC68040, and MC68060)
54	0D8	Floating-Point Signaling NAN (Defined for MC68881, MC68882, MC68040, and MC68060)
55	0DC	Floating-Point Unimplemented Data Type (Defined for MC68040 and MC68060)
56	0E0	MMU Configuration Error (Defined for MC68030 and MC68851)
57	0E4	MMU Illegal Operation Error (Defined for MC68851)
58	0E8	MMU Access Level Violation Error (Defined for MC68851)
59	0EC	(Unassigned, Reserved)
60	0F0	Unimplemented Effective Address (Defined for MC68060)
61	0F4	Unimplemented Integer Instruction (Defined for MC68060)
62–63	0F8–0FC	(Unassigned, Reserved)
64–255	100–3FC	User Defined Vectors (192)