# 80960SA/SB Bus 12

# CHAPTER 12
# 80960SA/SB BUS

The 16-bit multiplexed bus connects the 80960SA/SB processor to memory and I/O and forms the backbone of any 80960SA/SB processor-based system. This high bandwidth bus provides burst-transfer capability allowing up to 8 successive 16-bit data word transfers at a maximum rate of one word every clock cycle. In addition to the bus signals, the 80960SA/SB processor uses other signals to communicate to other bus masters. This chapter, which describes these signals and the associated operations, follows the outline shown below:

- Bus states and their relationship to each other
- Bus signal groups, which consist of address/data and control
- Bus read, write, and burst transactions
- Bus timing analysis and timing circuit generation
- Related bus operations such as arbitration, interrupt, and reset operations

## OVERVIEW OF THE 80960SA/SB BUS

The bus forms the data communication path between the various components in a basic 80960SA/SB hardware system. The 80960SA/SB processor utilizes the bus to fetch instructions, to manipulate information from both memory and I/O devices, and to respond to interrupts. To perform these functions, the 80960SA/SB processor provides a burst mode, which transfers up to eight data words at a maximum rate of one 16-bit word per clock cycle. The 80960SA/SB bus includes the following features:

- 32-bit addressing
- 16-bit multiplexed address/data path
- Two byte enables and an eight-word burst capability that allows transfers from 1 to 16 bytes in length
- Support for address latches and data buffers
- Basic bus states

## BASIC BUS STATES

The bus has five basic bus states: idle ($T_i$), address ($T_a$), data ($T_d$), recovery ($T_r$), and wait ($T_w$). During system operation, the 80960SA/SB processor continuously enters and exits different bus states (see Figure 12-1). The state diagram assumes that only one bus master resides on the bus.

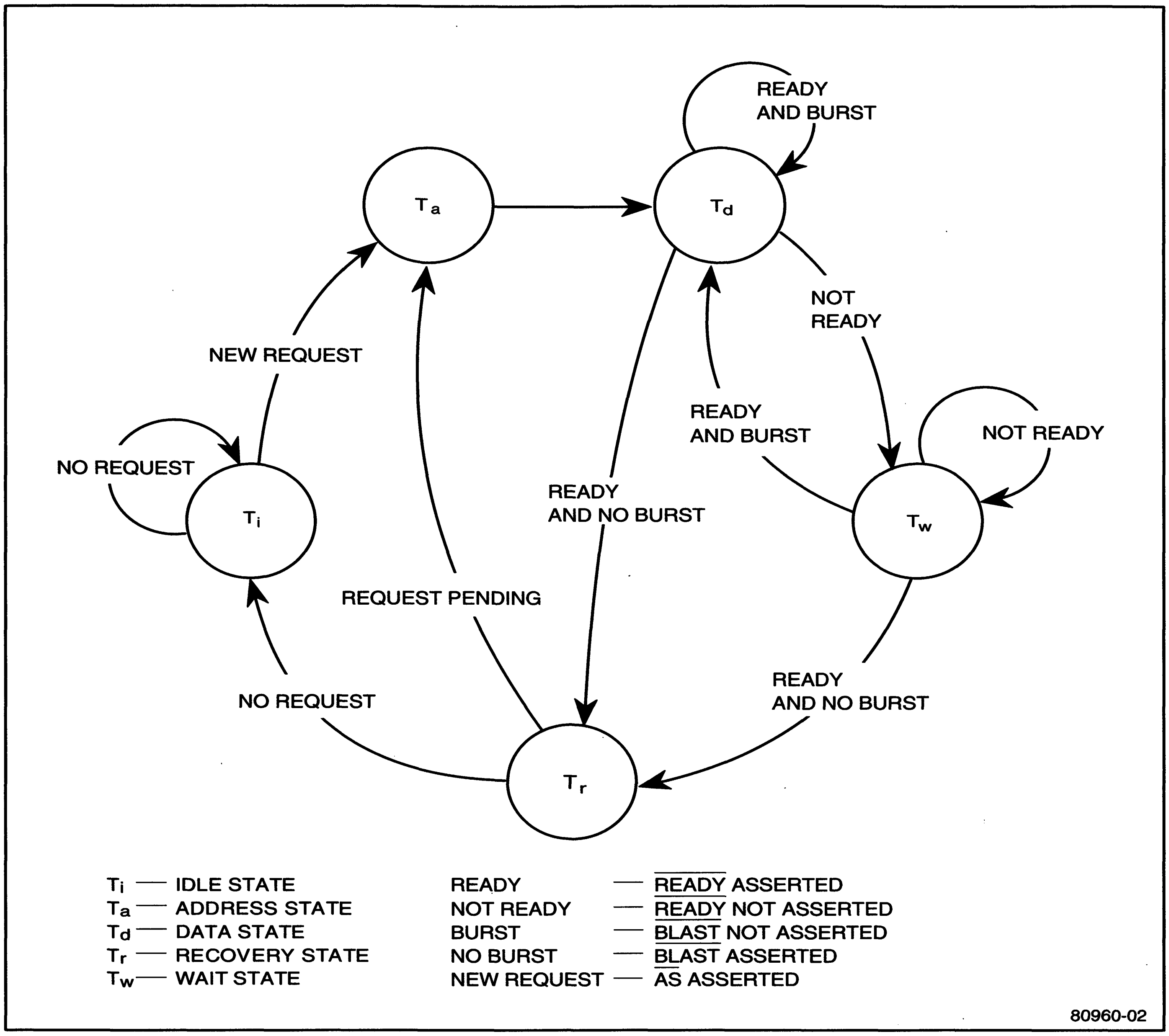


**Figure 12-1: Basic Bus States**

The bus occupies the idle ($T_i$) state when no address/data transfers are in progress. When the bus receives a new request, it enters the $T_a$ state to transmit the address.

Following a $T_a$ state, the bus enters a $T_d$ state to transmit or receive data on the address/data lines if data is ready. The assertion of the READY signal at the input of the processor indicates the ready state. If the data is not ready, the bus enters a $T_w$ state and remains in this state until data is ready. $T_w$ states may repeat as many times as necessary to allow sufficient time for the memory or I/O device to respond.

After a data word transfer in a non-burst transaction, the bus exits the $T_d$ or $T_w$ state and enters the recovery ($T_r$) state. In the case of a burst transaction, the local bus exits the $T_d$ or $T_w$ state and re-enters the $T_d$ state to transfer the next data word. Once all data words transfer in a burst transaction (up to eight), the bus enters the $T_r$ state to allow devices on the bus to recover.

When the recovery state completes, the bus enters the $T_i$ state if no new request is pending. If a request is pending, the bus enters the $T_a$ state to transmit the new address.

## BUS SIGNAL GROUPS

Signals on the bus (see Figure 12-2), consist of two basic groups: address/data and control. This section provides a description of both groups.

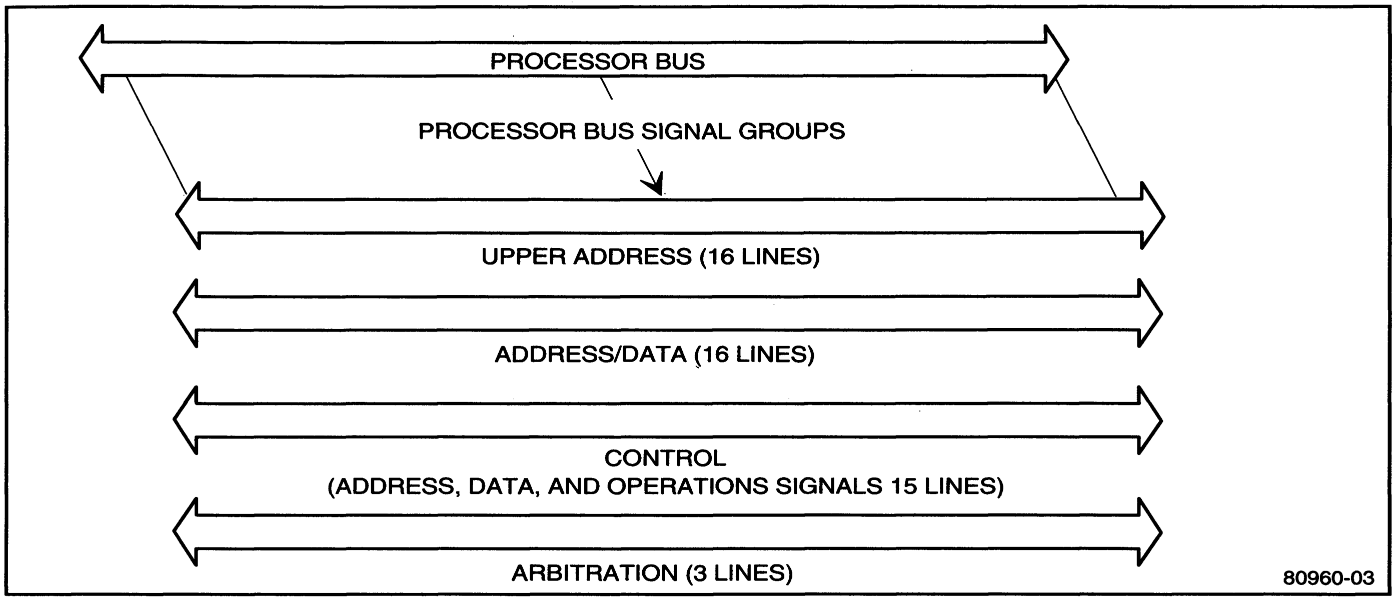


**Figure 12-2: Bus Signal Groups**

The 80960SA/SB, by default, drives all of the bus signals except during bus arbitration (HOLD, HLDA), in ONCE mode, and during reset. During idle states, the control signals are deasserted. The address lines remain valid until changed on a subsequent bus cycle.

## Address/Data

The address/data signal group consists of 35 lines. Sixteen of these signals multiplex within the processor to provide the lower address and data bits.

| | |
|---|---|
| A(16:31) | **Address Bus** carries the upper 16 bits of the 32-bit address to memory. It is valid throughout the burst cycle; no latch is required. |
| AD(0:15) | **Address/Data$_{15}$** through **Address/Data$_0$** represent the address signals on the bus during the $T_a$ state, and data signals during $T_d$ state. $AD_0$ represents the least-significant bit, and $AD_{15}$ is the most-significant address bit. $AD_{15}$ through $AD_0$ contain a physical byte address. The address/data signals float to a high-impedance state when not active. |
| A(1:3) | **Address Bus** carries the burst addresses of the 32-bit address to memory. These three bits are incremented during a burst access indicating the next byte address of the burst access. Note that A(1:3) are duplicated with AD(1:3) during the address cycle. |

## Control

The control signal group consists of 12 signals that permit the transfer of data. These signals control data buffers, address latches, and other standard interface logic. During idle cycles, control signals are deasserted.

| | |
|---|---|
| ALE | The **Address Latch Enable** is an active high signal that latches the address from the 80960SA/SB processor. The processor asserts ALE during the $T_a$ state and de-asserts ALE before the beginning of the $T_d$ state. ALE floats to a high-impedance level during bus arbitration when the processor is not the bus master (i.e., the hold state). |
| $\overline{AS}$ | **Address Status** indicates an address state. $\overline{AS}$ is asserted every $T_a$ state and de-asserted during the following $T_d$ state. $\overline{AS}$ is driven HIGH during reset. |
| DT/$\overline{R}$ | **Data Transmit/Receive** indicates the direction of data flow to or from the 80960SA/SB processor. For a read operation or an interrupt acknowledge, DT/$\overline{R}$ is low during the $T_a$, $T_d$, and $T_w$ states to indicate data flow into the 80960SA/SB processor. For a write operation, DT/$\overline{R}$ is high during the $T_a$, $T_d$, and $T_w$ states to indicate that data flows from the 80960SA/SB processor. DT/$\overline{R}$ never changes states when the processor asserts $\overline{DEN}$. DT/$\overline{R}$ is driven high on reset. |
| $\overline{DEN}$ | The **Data Enable** signal (active low) enables data transceivers. The processor asserts $\overline{DEN}$ during all $T_d$ and $T_w$ states. |

| | |
|---|---|
| $W/\overline{R}$ | The **Write/Read** signal instructs a memory or I/O device to write or read data on the bus. The 80960SA/SB processor asserts $W/\overline{R}$ during a $T_a$ state. The signal remains valid during subsequent $T_d$ and $T_w$ states. |
| $\overline{BE(0:1)}$ | The **Byte Enable** outputs of the 80960SA/SB processor specify which bytes on the 16-bit data bus will transfer during the transaction. When $\overline{BE_1}$ is asserted, only AD(1:7) and $D_0$ are valid. When both active-low signals are asserted, the entire 16-bit data bus is valid. $\overline{BE_1}$ and $\overline{BE_0}$ are deasserted except during read and write cycles. |

The byte enable signals are valid from the processor appropriately for each data cycle. Figure 12-3 shows a timing diagram for these signals.

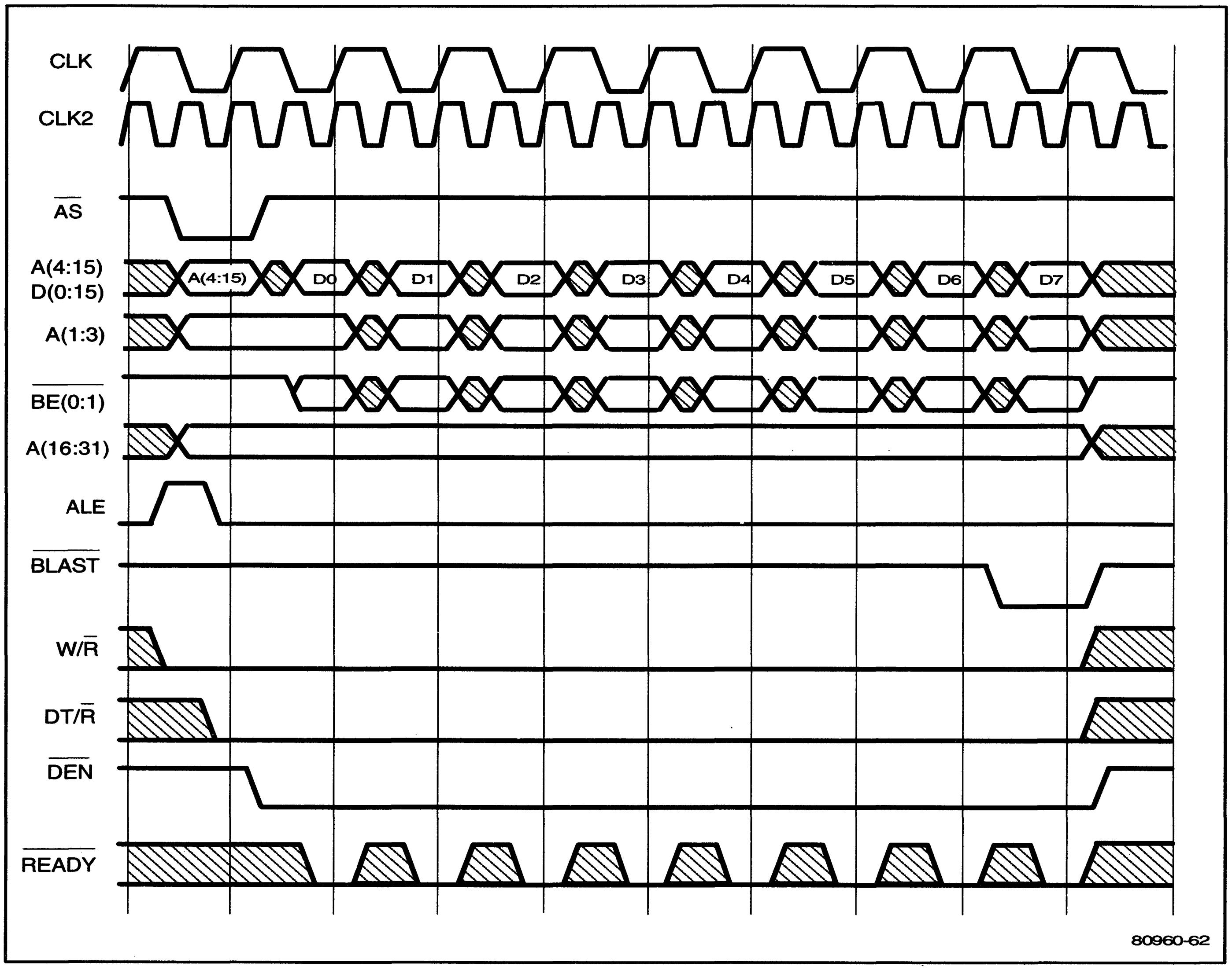


Figure 12-3: Byte Enable Timing Diagram

| | |
|---|---|
| $\overline{\text{READY}}$ | The $\overline{\text{READY}}$ signal indicates that the processor can sample (read) or remove (write) data on the bus. If the processor does not see READY asserted following the $T_a$ state or after a $T_d$ state, it generates a $T_w$ state internally. $\overline{\text{READY}}$ is an active-low input signal to the 80960SA/SB processor. |
| $\overline{\text{LOCK}}$ | **Bus $\overline{\text{Lock}}$** prevents other bus masters from gaining control of the bus during a bus operation. The processor activates this signal while processing specific operations and instructions.<br><br>For a read that is designated as an RMW-read, $\overline{\text{LOCK}}$ is examined. if asserted, the processor waits until it is not asserted; if not asserted, the processor asserts $\overline{\text{LOCK}}$ during the $T_a$ cycle and leaves it asserted.<br><br>A write that is designated as an RMW-write deasserts $\overline{\text{LOCK}}$ in the $T_a$ cycle. During the time $\overline{\text{LOCK}}$ is asserted, a bus agent can perform a normal read/write but no RMW operations. LOCK is also held asserted during an interrupt-acknowledge transaction.<br><br>The 80960SA/SB processor also asserts the $\overline{\text{LOCK}}$ signal during the interrupt acknowledge sequence. LOCK is an input and an open-drain output signal from the 80960SA/SB processor. LOCK must be pulled up if unused. |
| $\overline{\text{BLAST}}/\overline{\text{FAIL}}$ | This is a dual-function pin.<br><br>**Burst Last** indicates the last cycle ($T_d$) of any burst or single-cycle access. It is asserted low during the last $T_d$ cycle.<br><br>**Initialization Failure** indicates that the processor has failed to initialize correctly. The failure state is indicated by a combination of $\overline{\text{BLAST}}$ asserted and both $\overline{\text{BE}}$ signals not asserted. This condition occurs after $\overline{\text{RESET}}$ is de-asserted and before the first bus transaction begins.<br><br>$\overline{\text{FAIL}}$ is asserted while the processor performs a self test. If the self test completes successfully, then FAIL is de-asserted. Next, the processor performs a zero checksum on the first eight words of memory. If it fails, $\overline{\text{FAIL}}$ is asserted for a second time and remains asserted; if it passes, system initialization continues and FAIL remains de-asserted. |
| $\overline{\text{RESET}}$ | $\overline{\textbf{RESET}}$ clears the internal logic of the processor and causes it to re-initialize.<br><br>During $\overline{\text{RESET}}$ assertion, the input pins are ignored (except for $\overline{\text{INT}}_0$, $\text{INT}_1$, $\overline{\text{INT}}_3$, $\overline{\text{LOCK}}$), the tri-state output pins are placed in a HIGH impedance state (except for DT/R, $\overline{\text{DEN}}$, and $\overline{\text{AS}}$), and other output pins are placed in their non-asserted state. |

$\overline{\text{RESET}}$ must be asserted for at least 41 CLK2 cycles for a predictable reset. To synchronize the system clock (CLK in Figure 12-9), for a synchronous reset, the LOW to HIGH transition of $\overline{\text{RESET}}$ must occur after the rising edge of both CLK2 and the external bus clock, and before the next rising edge of CLK2.

The interrupt pins indicate the initialization sequence executed. Typical initialization requires driving only $\overline{\text{INT}_0}$ and $\overline{\text{INT}_3}$ to a HIGH state. The reset conditions follow:

| $\overline{\text{INT}_0}$ | $\text{INT}_1$ | $\overline{\text{INT}_3}$ | $\overline{\text{LOCK}}$ | Action taken |
|---|---|---|---|---|
| 1 | x | 1 | 1 | Run self-test (core initialization) |
| 0 | 0 | 1 | 1 | Disable self-test |
| 0 | 1 | x | x | Reserved |
| x | x | 0 | x | Reserved |
| x | x | x | 0 | ONCE mode |

Table 12-1 summarizes the bus signals.

**Table 12-1: Summary of Bus Signals**

| Signal Group | Signal Symbol | Signal Function | Active State | Type of Output and Direction |
|---|---|---|---|---|
| Upper Address | Address A(16:31) | 16-bit address | $T_a$ | 3-state (O) |
| Lower Address/Data | Address/Data AD(1:15), D0 | 16-bit address/data | $T_d$-$T_w$ $T_a$ | 3-state (I/O) |
| Lower Address 3-bit | Address A(1:3) | Lower 3 bits during burst | $T_a$-$T_d$ $T_w$ | 3-state (O) |
| Control | $\overline{\text{BLAST/FAIL}}$ | $\overline{\text{BLAST}}$: Initiates last data cycle; $\overline{\text{FAIL}}$: processor failed to initialize | $T_d$ | 3-state (O) |
| Control | ALE | Enables Address Latch | $T_a$ | 3-state (O) |
| Control | $\overline{\text{AS}}$ | Identifies an address state | $T_a$ | 3-state (O) |
| Control | DT/$\overline{\text{R}}$ | Controls direction of data flow | $T_a$,$T_d$, $T_w$ | 3-state(O) |
| Control | $\overline{\text{DEN}}$ | Enables data transceiver/latch | $T_d$,$T_w$ | 3-state(O) |
| Control | W/$\overline{\text{R}}$ | Read/Write Command | $T_a$,$T_d$, $T_w$ | 3-state(O) |
| Control | $\overline{\text{BE}}$(0:1) | Specifies which data bytes to transfer | $T_d$ | 3-state(O) |
| Control | $\overline{\text{READY}}$ | Indicates data is ready to transfer | Note 1 | N/A(I) |
| Control | $\overline{\text{LOCK}}$ | Locks bus | any | Open-drain(O) |
| Control | $\overline{\text{RESET}}$ | Clears processor's internal logic | $T_d$ | N/A (I) |

Note 1: sampled during $T_d$ and $T_w$

The 80960SA/SB processor uses additional pins to control the execution of instructions and to interface to other bus masters. These pins include the interrupt and error signals. Each of these signal groups are explained in following sections.

2. During the $T_d$ state, several actions occur.
   - The 80960SA/SB processor asserts $\overline{DEN}$ to enable data transceivers.
   - It asserts $\overline{BE_1\text{-}BE_0}$ to specify which bytes the processor uses when reading the data word.
   - External timing asserts $\overline{READY}$ logic allowing the processor to receive data from the storage devices. If $\overline{READY}$ is not asserted, the bus enters a $T_w$ state on the following CLK. The $T_w$ state repeats until the system asserts $\overline{READY}$.
3. The $T_r$ state follows the data state. This allows the system components enough time (one processor clock (CLK) cycle) to remove their outputs from the bus before the 80960SA/SB processor generates the next address on the address/data lines. During the $T_r$ state, W/R, DT/R, and DEN are inactive.

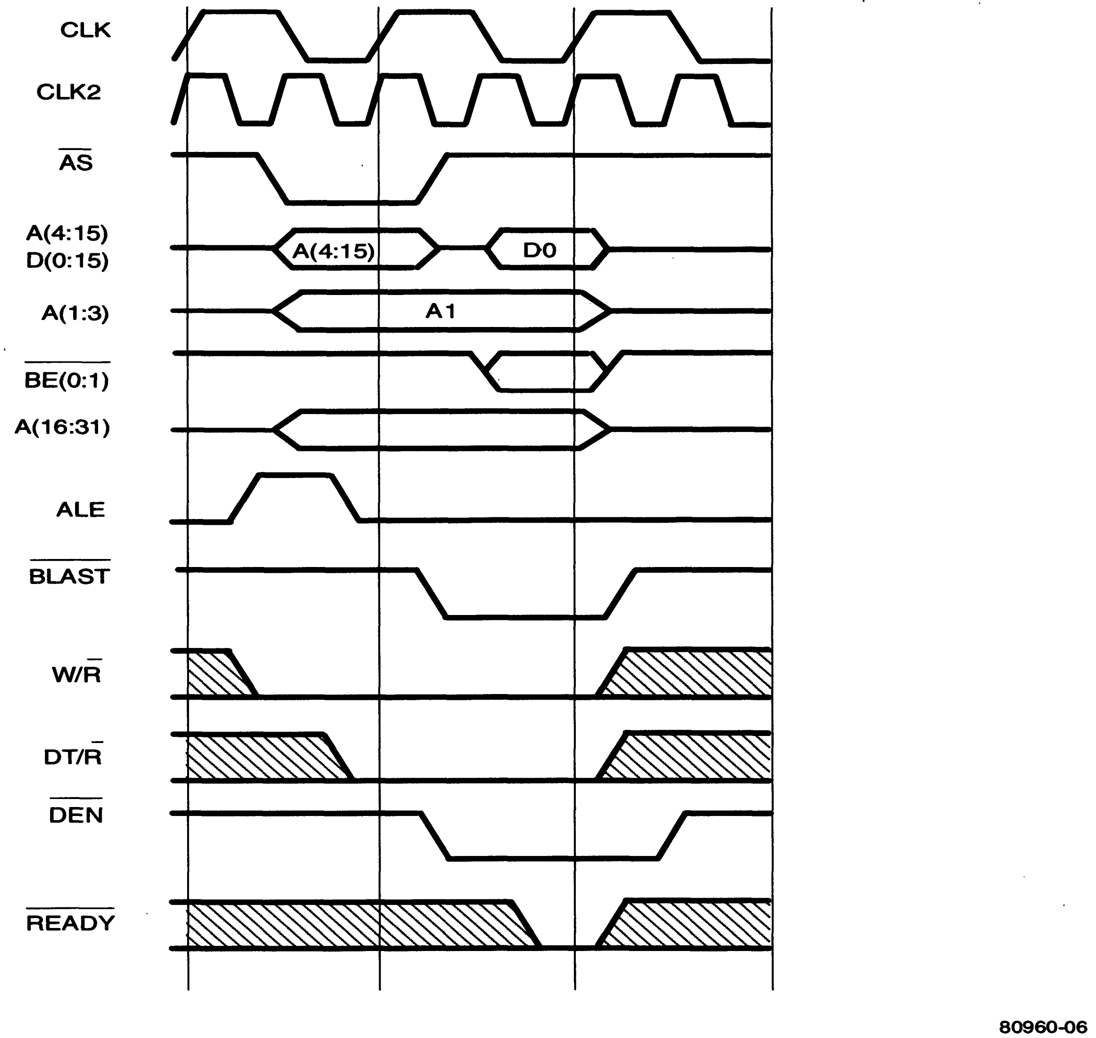


**Figure 12-5: 80960SA/SB Processor Read Transaction**

## BUS TRANSACTIONS

The 80960SA/SB processor uses the bus signals to perform transactions to transfer data to (or from) the CPU from (or to) another component. During a transaction, the 80960SA/SB processor can transfer up to eight contiguous 16-bit words of data for a single memory request to enhance system throughput.

### Clock Signal

The 80960SA/SB hardware system typically uses two clock signals (CLK2 and CLK) to synchronize the transitions between bus states. CLK2 provides the clock input to the 80960SA/SB. This clock is also the system clock. CLK is not used by the processor but is generated by external logic synchronously to the processor clock input (CLK2). CLK defines the state transition boundaries at one-half the frequency of CLK2, provides a convenient indicator of bus boundaries, and can drive peripheral devices. Figure 12-4 illustrates the relationship between the system CLK2 and CLK.

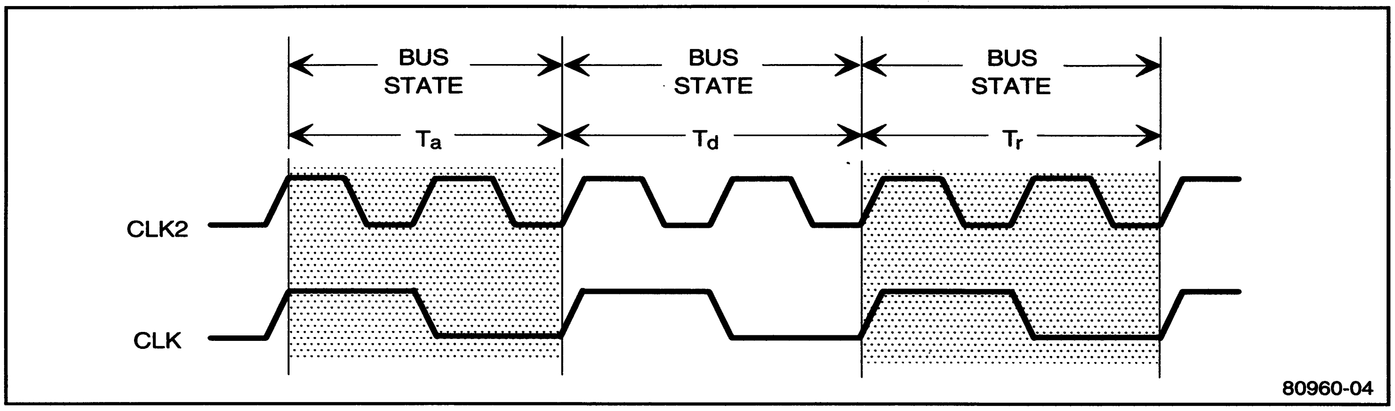


**Figure 12-4: Clock Relationships**

### Basic Read

A basic transaction reads or writes one data word. Figure 12-5 shows a typical timing diagram for a read transaction. See the 80960SA/SB processor data sheet for exact timing relationships. The following sequence of events explains the flow of the timing diagram. For simplicity, the figure shows no wait states.

1. The 80960SA/SB processor generates several signals during the $T_a$ state.
   - It transmits the address on the address and address/data lines.
   - It pulses ALE to latch the address.
   - It asserts $\overline{AS}$.
   - It brings $W/\overline{R}$ low to denote a read operation.
   - It brings $DT/\overline{R}$ low to define the input direction of the data transceivers.

## Write Transaction

Figure 12-6 shows a typical diagram for a write transaction using one wait state. The following sequence of events explains the flow of the timing diagram.

1. Similar to the read transaction, the 80960SA/SB processor generates several signals during the $T_a$ state.
   - It transmits the address on the address and address/data lines.
   - It pulses ALE to latch the address.
   - It asserts $\overline{AS}$.
   - It brings W/$\overline{R}$ high to denote a write operation.
   - It brings DT/$\overline{R}$ high to define the direction input for the data transceivers.
2. During the $T_d$ state, several actions occur.
   - The 80960SA/SB places the data on the address/data lines.
   - It asserts $\overline{BE}$(0:1) to specify which bytes the processor is writing in the word.
   - The 80960SA/SB processor asserts $\overline{DEN}$ to enable data transceivers.
   - In this example, external timing does *not* assert $\overline{READY}$, causing a wait state. Consequently, the processor holds data on the AD lines.
3. During the $T_w$ state, external timing asserts $\overline{READY}$ and completes the write to the storage device. Note that W/$\overline{R}$, DT/$\overline{R}$, and $\overline{DEN}$ remain constant until the bus state after the assertion of the signal $\overline{READY}$.
4. The $T_r$ state follows the wait state. During the $T_r$ state, W/$\overline{R}$, DT/$\overline{R}$, and $\overline{DEN}$ are inactive.

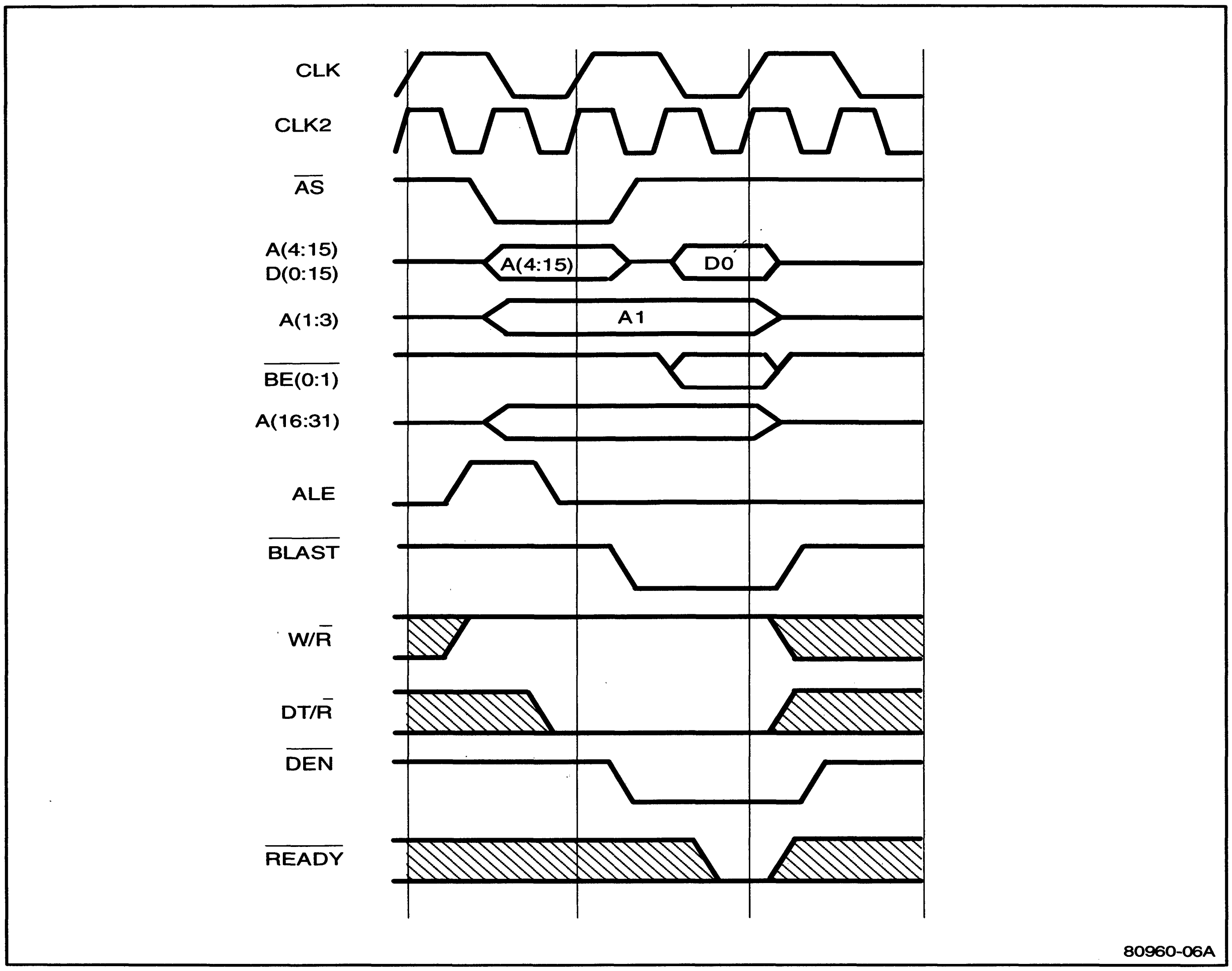


**Figure 12-6: 80960SA/SB Processor Write Transaction**

## Burst Transactions

The 80960SA/SB processor supports burst transactions that read or write up to 8, 16-bit words (16 contiguous bytes) at a maximum rate of one word every bus cycle. The byte enable signals are valid for each word. This allows byte write operations to a word.

A burst read or write transaction is similar to a single-word read or write operation. It differs primarily in the number of data words that the processor transfers; a basic transaction always transfers one word, while a burst transaction transfers up to eight data words. For a burst transaction, the byte enable signals operate during the $T_d$ state. Figure 12-7 shows the timing for a four-word burst read transaction without wait states. $D_0$ and $D_1$ are the two bytes that comprise the first word. $D_2$ and $D_3$ make up the second word, and so on.

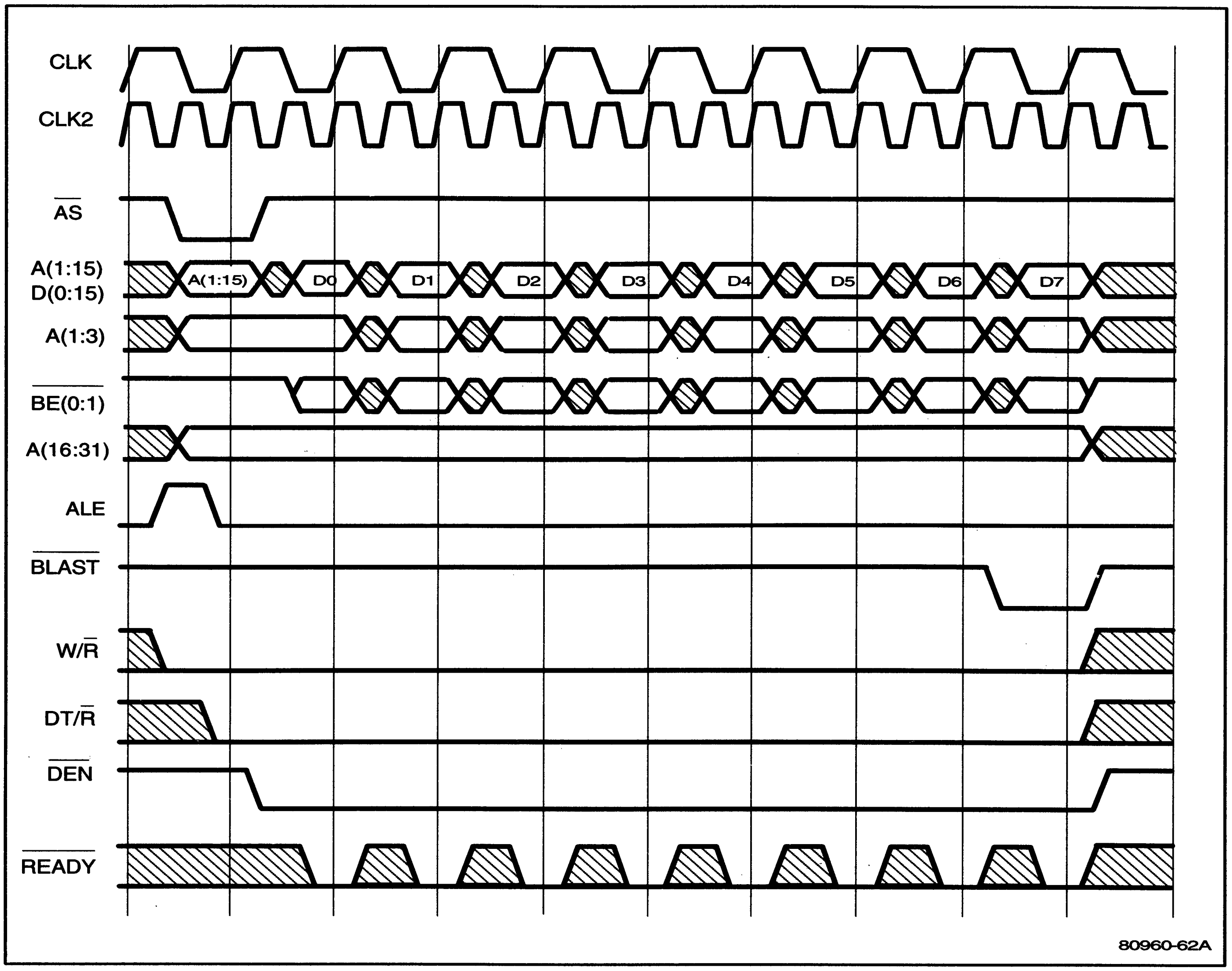


**Figure 12-7: Processor 8, 16-bit Words Burst Read Transaction**

Figure 12-8 shows the timing for a 2-word burst write transaction with a wait state occurring during transfer of the first word.

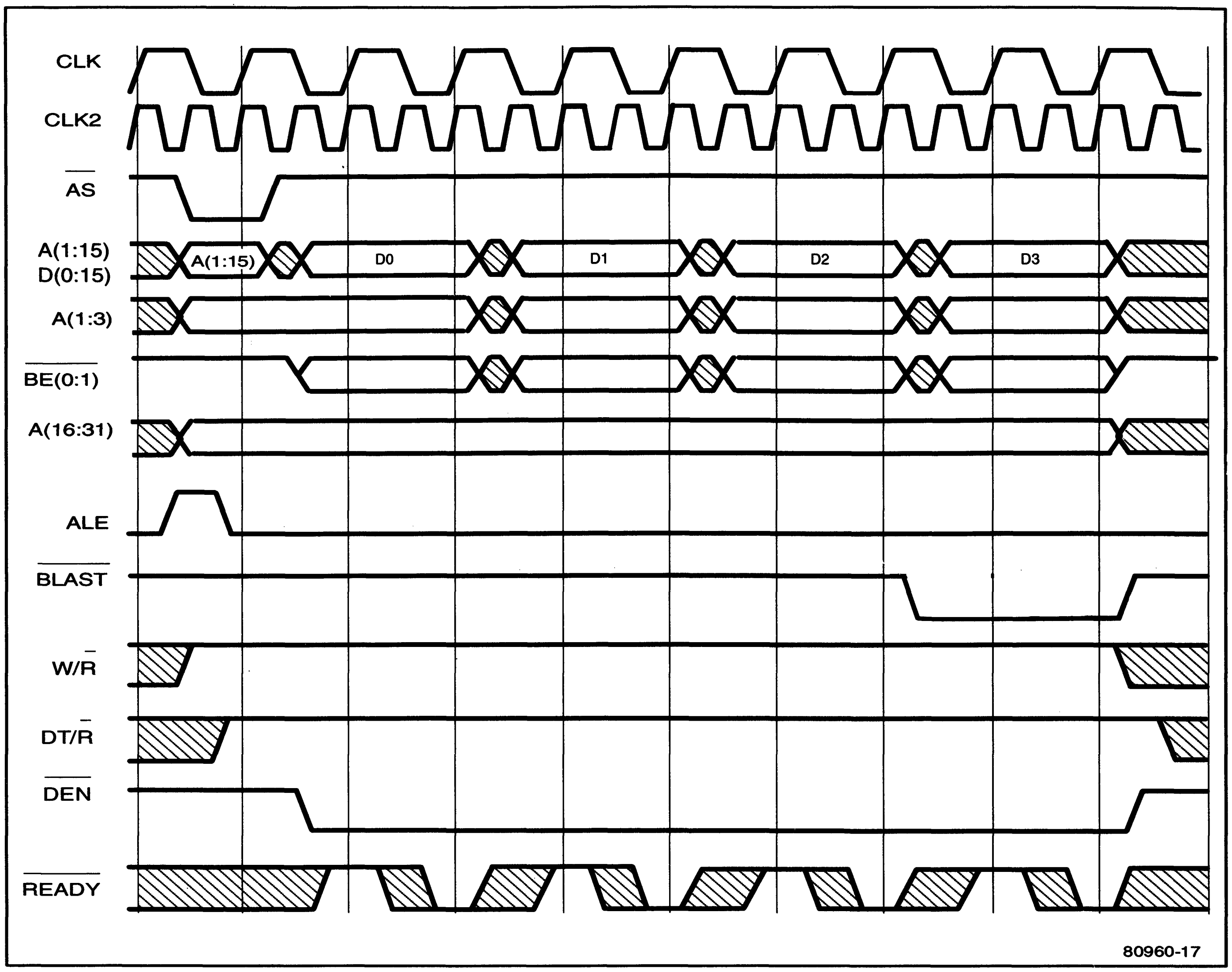


**Figure 12-8: Processor 2-word Burst Write Transaction With Wait States**

## TIMING GENERATION

In an 80960SA/SB processor-based system, external logic must provide circuitry that generates timing signals for the clock and reset inputs. To generate these signals, the logic should minimize skew and produce quick rise and fall times. This section describes a typical circuit that synthesizes the clock signal. The "Reset and Initialization" section of this chapter discusses the RESET timing generation.

## 80960SA/SB Clock Requirements

In order to design a clock generator, you must first examine the clock input specifications. The following four parameters specify the clock pulse.

1. The clock fall time ($t_f$)
2. The clock low time ($t_l$)
3. The clock rise time ($t_r$)
4. The clock period ($t_{cyc}$)

The time required to go from 90% of the difference between the high and low voltage levels (to 10% of the difference) or from low to high is defined as the clock fall (rise) time. The clock low time specifies the time required for the clock to remain within 10% of the low voltage level. Similarly, the clock high time specifies the required time for the clock pulse to remain within 10% of the high voltage level. The clock period is the sum of $t_f + t_l + t_r + t_h$.

The clock generator must have fast enough rise and fall times to comply with the requirements for high and low time and the overall clock period. For example, consider a clock pulse with a 50% duty cycle at 32 MHz. The clock period specifies a minimum of 31 ns, a low time at a minimum of 9 ns, and high time at a minimum of 9 ns. This implies that the sum of the rise and fall time must not be greater than 13 ns. Thus, the clock design should have rise and fall times not greater than 6.5 ns each. Besides specifying a maximum clock rate, the 80960SA/SB processor requires a minimum CLK2 rate of 4 MHz to maintain the state of the internal dynamic cells. Due to this minimum frequency requirement, you can not disable the clock to single-step the 80960SA/SB processor.

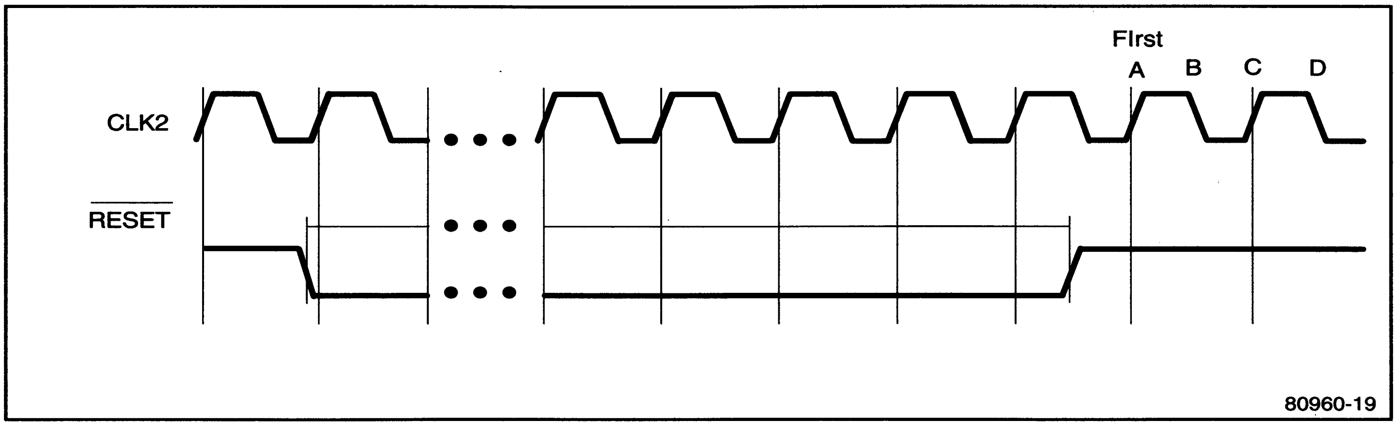


Figure 12-9: CLK2 Edges

## Clock Generation

The 80960SA clock input is divided by two to generate the processor's internal clock frequency, e. g. a 10 MHz 80960SA uses a 20 MHz clock oscillator. This clock oscillator (CLK2) is also typically used to clock other parts of a system such as bus control logic. Because of this some means must exist to determine the phase of the 80960SA's internal clock with respect to CLK2 (each processor clock period overlaps two CLK2 rising edges -- which of these corresponds to the beginning of the processor's cycle?). There are two means of doing this in 80960SA system. One is to syncronize RESET with the falling edge of a 1x frequency clock (CLK1) derived from CLK2. When synchronized in this way, the next rising edge of CLK2 after RESET is deasserted is guaranteed to be the beginning of a processor cycle. This will then correspond to all rising edges of CLK1. If an asynchronous RESET is desired then another method can be used. The 80960SA's $\overline{\text{AS}}$ output signal driver is asserted after a rising edge of CLK2 corresponding to the beginning of a processor cycle. Furthermore the driver is designed to be fast enough that $\overline{\text{AS}}$ is valid well before the next rising edge of CLK2. Thus, at the start of each bus cycle external logic can use $\overline{\text{AS}}$ to determine the processor's clock phase.

Knowing the processor's clock phase is important during bus cycles because most signals are asserted and deasserted on cycle boundaries and data is sampled (read) or removed (write) on cycle boundaries.

## Open-Drain Pull-ups

The $\overline{\text{LOCK}}$ output uses a high-current open-drain driver allowing the signal to be wire-ORed in systems with more than one bus master. The data sheet specifies the maximum fall time of this signal as 13 ns for a 16 Mhz part. The rise time, however, is a function of both the maximum float delay specification of 5 to 20 ns and the pull-up resistor values. If you want the signal rise time to equal the fall time, then the pull-up resistor must be able to charge the load capacitance within 10 ns after the driver turns off. The equation for the voltage on a capacitor charging from voltage $V_f$ through a resistor as a function of time is:

1. $V(t) = V_f(1 - e^{-t/RC})$

   solving for R:

2. $R = -t/(C(\ln(V_f - V)/V_f))$

   For an open-drain driver connected to a node at 0 V, assuming a worst case power supply voltage of 4.5V, the $V_f$ available to charge the load capacitance is 4.5V. Assuming a 50 pF load capacitance and assuming V equals 2.0 V (TTL high level) then:

3. $R = -10\ \text{ns}/(50\text{pF}(\ln(4.5\text{V} - 2.0\text{V})/4.5\text{V})) = 340\ \text{Ohms}$

If we recalculate the above formulas assuming that the node is at 0.45V (data sheet voltage specification at 25mA) then $V_f$ becomes (4.5V - 0.45V) and V becomes (2.0V - 0.45V) giving an R of 415 Ohms. Since the driver *sinks* only 13 mA with a 415 Ohm pull-up and 5.5V of supply voltage the VOL will be less than 0.45V. Thus, the lower 340 Ohm value represents the worst case. If we assume a 100 pF load capacitance and wish to maintain the 30 ns (float delay + 10 ns) rise time then the design requires a pull-up resistor of approximately 340/2 or 170 Ohms.

## ARBITRATION

When multiple bus masters exist, an arbitration process exchanges control of the bus. The process assumes two bus masters: a default bus master that controls the bus, and another that requests control of the bus when it performs an operation (e.g., a DMA controller). More than two bus masters may exist on the bus, but this requires external arbitration logic.

The 80960SA/SB processor controls the bus, and a master I/O peripheral (such as a DMA controller) requests control of the bus. The 80960SA/SB processor and the I/O peripheral device exchange control of the bus with two signals: the hold request (HOLD) and hold acknowledge (HLDA) signals.

HOLD is an input signal of the 80960SA/SB processor, which indicates that the master I/O peripheral requests control of the bus. When the peripheral asserts HOLD, the 80960SA/SB processor surrenders control of the bus after it completes the current bus transaction. The 80960SA/SB processor then acknowledges transfer of bus control to the requesting bus master when it asserts HLDA.

### State Diagram

Figure 12-10 shows the state diagram for a bus with two bus masters: an 80960SA/SB processor, and an I/O peripheral device. This state diagram includes a hold state ($T_h$) in addition to the five basic states described in the *Basic Bus States* section of this chapter. The 80960SA/SB processor enters the $T_h$ state when it surrenders control of the bus. It can enter the $T_h$ state from the $T_i$, $T_r$, $T_d$, or $T_w$ state. When the 80960SA/SB processor regains control of the bus, it enters the $T_a$ state if a new request is pending, or it enters the $T_i$ state if no new request is pending.

## Arbitration Timing

Figure 12-11 illustrates the arbitration timing diagram. The initial *T* state represents the last cycle of a transaction in which the processor asserts a READY signal or a $T_i$ state. The 80960SA/SB processor receives a request to relinquish control of the bus when the device asserts HOLD. After the 80960SA/SB processor completes the current transaction, it responds to this request when it floats the tri-state output signals and de-asserts the open-drain output signal. The HLDA output signal, however, remains active as the 80960SA/SB processor enters a $T_h$ state. During the $T_h$ state, the CPU ignores all input signals except HOLD and RESET. When the device de-asserts the HOLD input, the 80960SA/SB processor exits the $T_h$ state, de-asserts HLDA, and enters a $T_a$ state (for pending request) or it enters the $T_i$ state (if no request is pending). Refer to Table 12-1 for signal states during this operation.

## INTERRUPTS

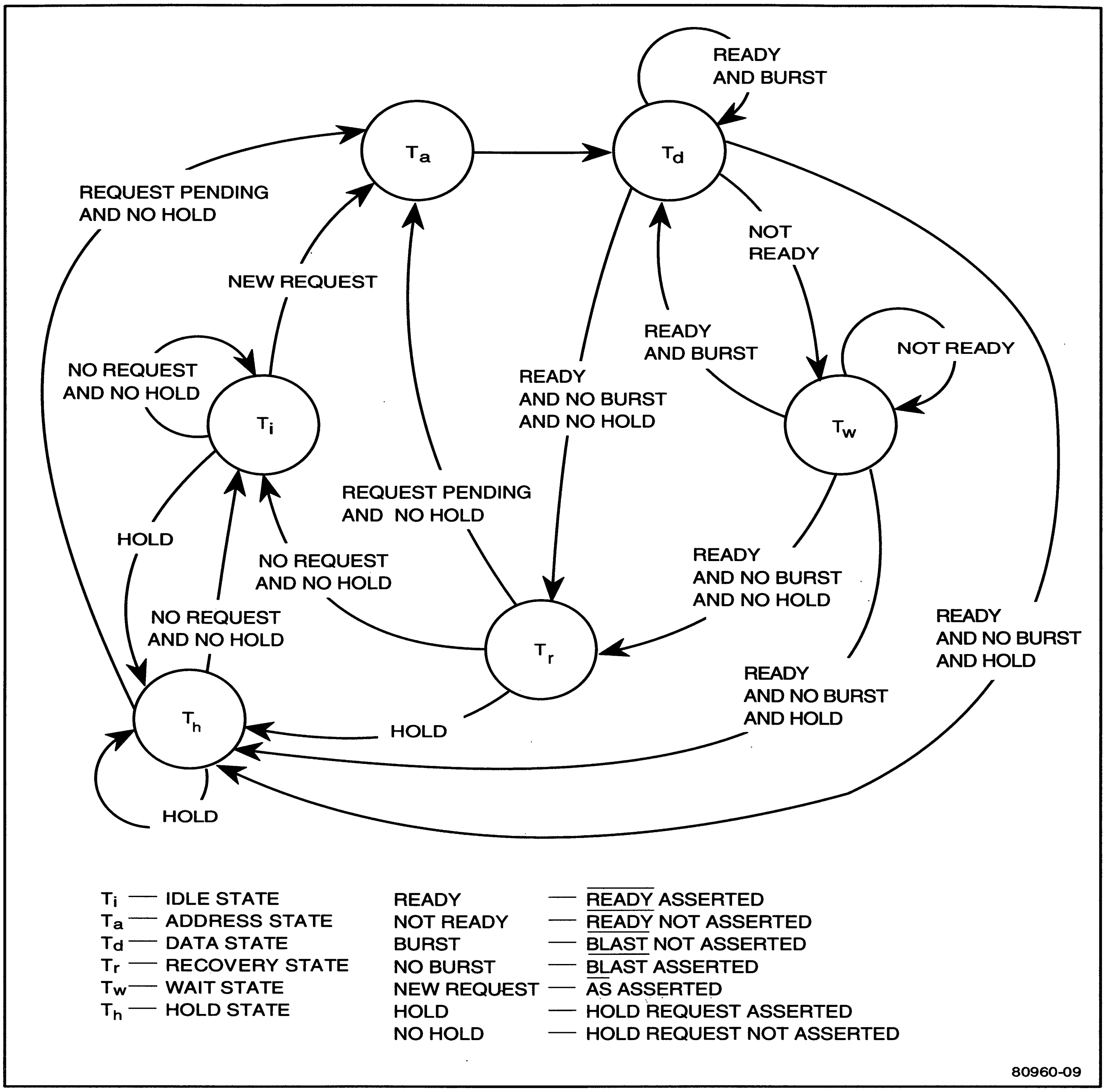
The 80960SA/SB processor responds to external events occurring at arbitrary times through an interrupt signal. Various sources, which include hardware components and special software instructions, generate an interrupt signal that can suspend execution of the 80960SA/SB processor's current instruction stream. This section discusses hardware-generated interrupts. For complete information on software-generated interrupts, see Chapter 5, "Interrupts."

The 80960SA/SB architecture provides a flexible interrupt structure. The processor can be interrupted using any of the two methods shown below:

1. Receipt of a signal on any or all of the four direct interrupt input signals ($\overline{\text{INT}_0}$, $\text{INT}_1$, $\text{INT}_2$, and $\overline{\text{INT}_3}$).
2. Receipt of a signal on the interrupt request (INTR) line to obtain an external interrupt vector.

The setting in the on-chip Interrupt Control Register selects one of the methods. Interrupt signals can occur during any bus state regardless of the selected method.

This section provides details on the multiplexed interrupt pins, the two interrupt methods, the Interrupt Control Register, synchronization, and interrupt latency.


READY AND BURST
Ta
Td
REQUEST PENDING AND NO HOLD
NEW REQUEST
NOT READY
READY AND BURST
NOT READY
NO REQUEST AND NO HOLD
Ti
READY AND NO BURST AND NO HOLD
Tw
REQUEST PENDING AND NO HOLD
HOLD
NO REQUEST AND NO HOLD
READY AND NO BURST AND NO HOLD
NO REQUEST AND NO HOLD
Tr
READY AND NO BURST AND HOLD
READY AND NO BURST AND HOLD
Th
HOLD
HOLD
80960-09

| | | | |
|---|---|---|---|
| $T_i$ | — IDLE STATE | READY | — $\overline{\text{READY}}$ ASSERTED |
| $T_a$ | — ADDRESS STATE | NOT READY | — $\overline{\text{READY}}$ NOT ASSERTED |
| $T_d$ | — DATA STATE | BURST | — $\overline{\text{BLAST}}$ NOT ASSERTED |
| $T_r$ | — RECOVERY STATE | NO BURST | — $\overline{\text{BLAST}}$ ASSERTED |
| $T_w$ | — WAIT STATE | NEW REQUEST | — $\overline{\text{AS}}$ ASSERTED |
| $T_h$ | — HOLD STATE | HOLD | — HOLD REQUEST ASSERTED |
| | | NO HOLD | — HOLD REQUEST NOT ASSERTED |

**Figure 12-10: Bus States with Arbitration**

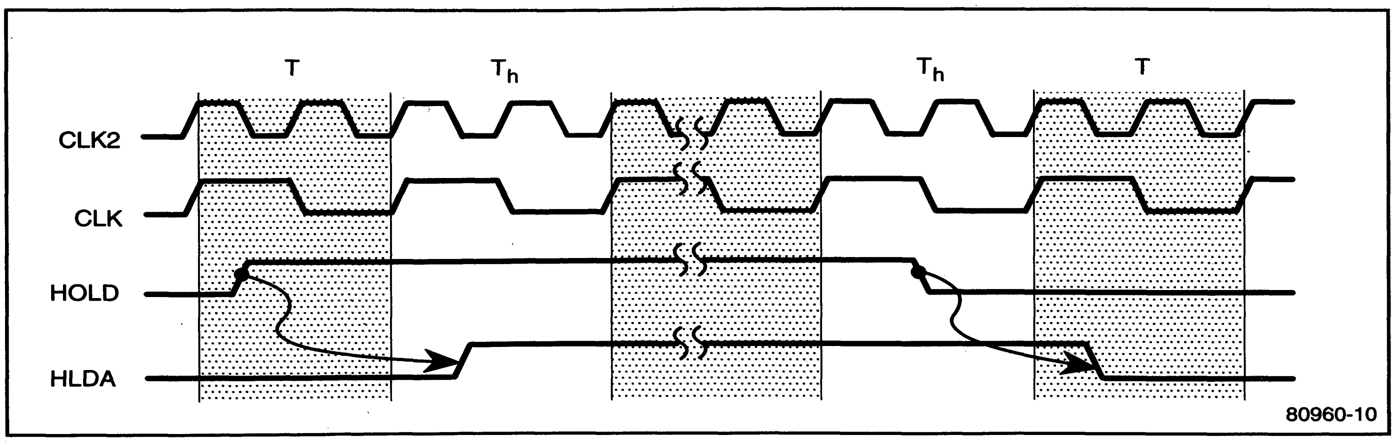


**Figure 12-11: Arbitration Timing Diagram for a Bus Master**

## Interrupt Signals

The interrupt signals multiplex on four pins of the 80960SA/SB processor: $\overline{INT_0}$, $INT_1$, $INT_2$/INTR, and $\overline{INT_3}$/INTA. The on-chip Interrupt Control Register determines how the processor uses these pins (see "Interrupt Control Register" section of this chapter).

| | |
|---|---|
| $\overline{INT_0}$ | **Interrupt 0** signals an interrupt. The bus interrupt control register determines in which way the signal should be interpreted. To signal an interrupt request in a synchronous system, this pin (as well as the other interrupt pins) must be enabled by being de-asserted for at least one bus cycle and then asserted for at least one additional bus cycle; in an asynchronous system, the pin must remain de-asserted for at least two bus cycles and then be asserted for at least two more bus cycles. $\overline{INT_0}$ is sampled during RESET to determine if the self-test sequence is to be executed. |
| $INT_1$ | **Interrupt 1** signals a direct interrupt, like $\overline{INT_0}$. $INT_1$ is sampled during RESET to determine if the self-test sequence is to be executed. |
| $INT_2$/INTR | **Interrupt 2/Interrupt Request**: The bus control register determines how this pin is interpreted. If $INT_2$, it has the same interpretation as the $\overline{INT_0}$ and $INT_1$ pins. If INTR, it is used to receive an interrupt request from an external 8259A compatible interrupt controller. |

$\overline{\text{INT}_3}/\overline{\text{INTA}}$

**Interrupt 3/Interrupt Acknowledge**: The bus control register determines how this pin is interpreted. If $\overline{\text{INT}_3}$, it has the same interpretation as the $\text{INT}_0$ and $\text{INT}_1$ pins. If INTA, it is used as an output to control interrupt-acknowledge bus transactions. The $\overline{\text{INTA}}$ output is latched on-chip and remains valid during $T_d$ cycles. $\text{INT}_3$ must be pulled to a HIGH state during RESET.

## Using an External Interrupt Controller

Using the INTR and $\overline{\text{INTA}}$ signals, the 80960SA/SB processor can perform an interrupt acknowledge sequence to communicate with an external interrupt controller. Figure 12-12 shows a timing example of an interrupt acknowledge sequence using the 8259A Programmable Interrupt Controller.

The 8259A asserts and holds INTR until the 80960SA/SB processor activates the $\overline{\text{INTA}}$ signal for the first time. When the 80960SA/SB processor receives an interrupt request, the CPU completes the current transaction (at some interruptible point), and asserts $\overline{\text{INTA}}$. $\overline{\text{INTA}}$ remains valid through the $T_a$, $T_d$, and $T_w$ states of the interrupt acknowledge cycle. The first assertion of $\overline{\text{INTA}}$ triggers the 8259A to resolve priority among its interrupt requests. External logic must assert $\overline{\text{READY}}$ and generate wait states if required (typically 4).

To compensate for the timing of the 8259A, the 80960SA/SB processor inserts five $T_i$ states before asserting the INTA again to read the interrupt vector. Figure 12-12 shows READY asserted without a wait state during the first interrupt acknowledgement cycle and with one wait state during the second interrupt acknowledgement cycle. In practice, the 8259A would require about four wait states in both cycles. The address during the $T_a$ state for both interrupt acknowledge cycles is $\text{FFFFFFFC}_{16}$.

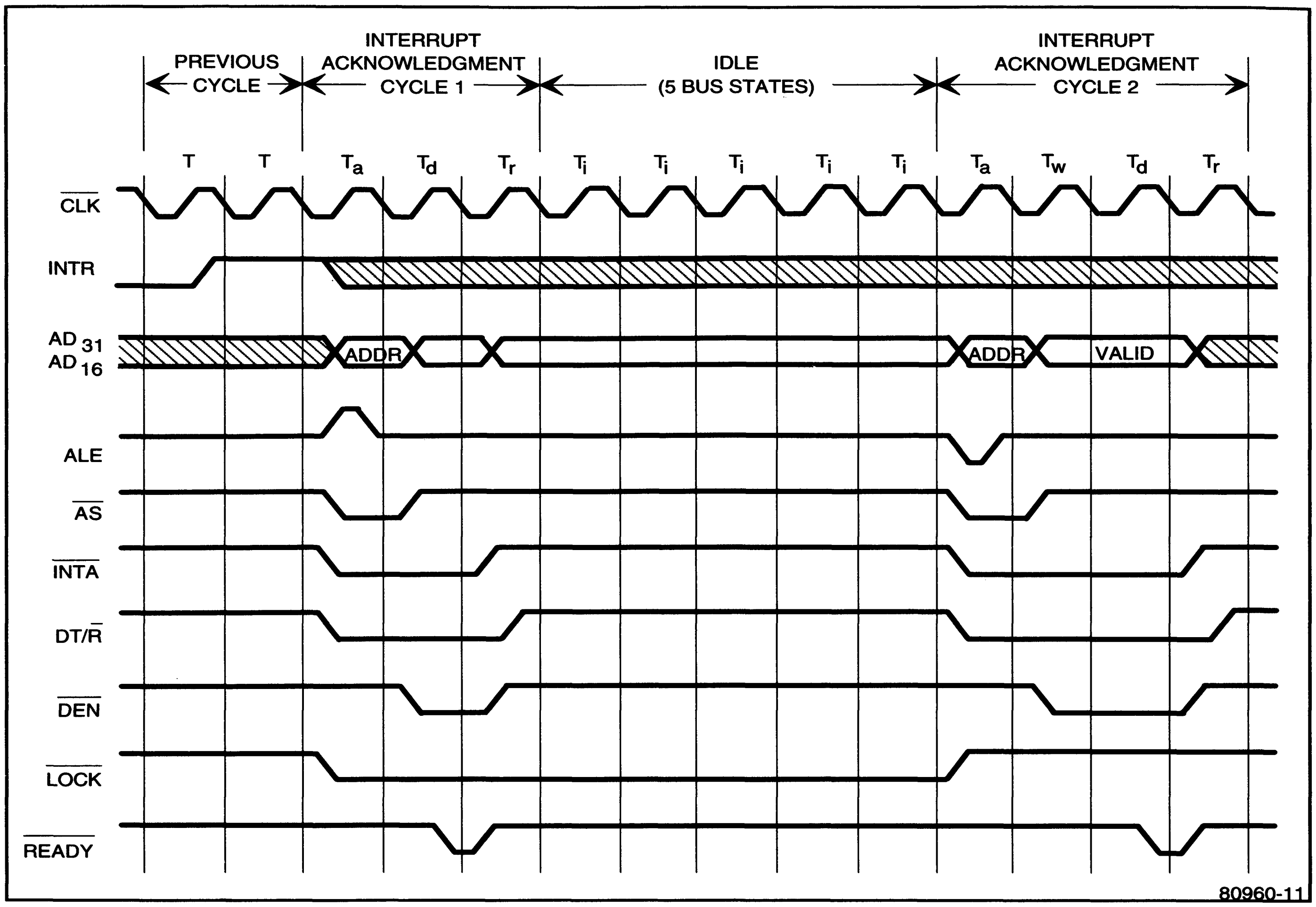


**Figure 12-12: Timing Diagram for Interrupt Acknowledge Transaction**

The 80960SA/SB processor services the interrupt according to its priority. If the interrupt has higher priority than the current activity, the 80960SA/SB processor services it immediately. Otherwise, after reading the interrupt vector, the 80960SA/SB processor posts the interrupt vector in the interrupt table. Typically, the 80960SA/SB processor responds within 4 microseconds for an interrupt with higher priority than the current process (assuming CLK2 at 32 Mhz). If the interrupt has lower priority than the current activity, the processor services the interrupt when its priority is higher than the priority of the subsequent activity of the 80960SA/SB processor.

## Synchronization

The $\overline{\mathrm{INT}_0}$, $\mathrm{INT}_1$, $\mathrm{INT}_2$/INTR, and $\overline{\mathrm{INT}_3}$ input signals are either synchronous or asynchronous to the system clock (CLK2). To properly preset the interrupt signals for synchronous operation, $\overline{\mathrm{INT}_0}$, $\mathrm{INT}_1$, $\mathrm{INT}_2$/INTR, and $\overline{\mathrm{INT}_3}$ must be de-asserted for at least one processor clock (CLK) cycle and asserted for at least one processor clock (CLK) cycle. These signals may be de-asserted and asserted individually.

If the interrupt signals are asynchronous to CLK2, the 80960SA/SB processor internally synchronizes them. For the CPU to recognize the asynchronous interrupt input signals, external logic must de-assert them for at least two processor clock (CLK) cycles, and then assert them for at least two processor clock (CLK) cycles. External logic may de-assert and assert the signals individually.

## Hardware Interrupt Acknowledge

The interrupt inputs may be individually asserted or de-asserted. The 80960SA/SB interrupt controller intelligently manages interrupts. There are two main stages that the 80960SA/SB enters before it executes the interrupt handler: hardware recognizes the interrupt and then a microcode interrupt routine executes. First, the interrupt pin is polled. Hardware stores this in a 4-bit register. The interrupt routine assigns one bit to each pin. This register captures subsequent interrupts once it recognizes one interrupt. The interrupt routine recognizes interrupts at instruction boundaries or interruptible points in long instructions (floating point) and then immediately disables them. However, it is important to note that disabling the interrupts does *not* disable the 4-bit register. Interrupts are posted in this register until microcode reaches a point where it checks the register again. When the interrupt routine reads the register again, it clears it. The highest priority bit in the 4-bit vector is cleared, which indicates that the interrupt vector associated with it will be used. Then this vector is written back to the register by an ORing function within the register, thus maintaining any new interrupts that may have been signalled.

Next, the 80960SA/SB processor recognizes an interrupt (since an event has been stored in the 4-bit register). At this point, a hardware mechanism in the interrupt controller calls the interrupt microcode routine, which executes all the instructions needed to get into or out of the interrupt handler. After the interrupt microcode routine completes, it calls the user-supplied interrupt handler and begins executing instructions. The user never needs to be concerned with housekeeping activities, since the processor handles it automatically.

## RESET AND INITIALIZATION

The system $\overline{\text{RESET}}$ signal provides an orderly way to start or restart the 80960SA/SB processor. When the 80960SA/SB processor detects the high-to-low transition of $\overline{\text{RESET}}$, it terminates all external activities and places the output pins in the high-impedance state or de-asserted condition. When the $\overline{\text{RESET}}$ signal goes high again, the 80960SA/SB processor begins the initialization process and later begins fetching instructions from a specific address.

### Reset Timing Requirements

To properly reset the 80960SA/SB processor to a known state, the processor must receive the high-to-low transition of $\overline{\text{RESET}}$ relative to any rising edge of CLK2 and remain high for at least 41 CLK2 cycles (see Figure 12-13). $\overline{\text{RESET}}$ must be de-asserted after the rising edge of CLK2, and prior to the next rising edge of CLK2. This establishes the next rising edge of CLK2 as edge A.

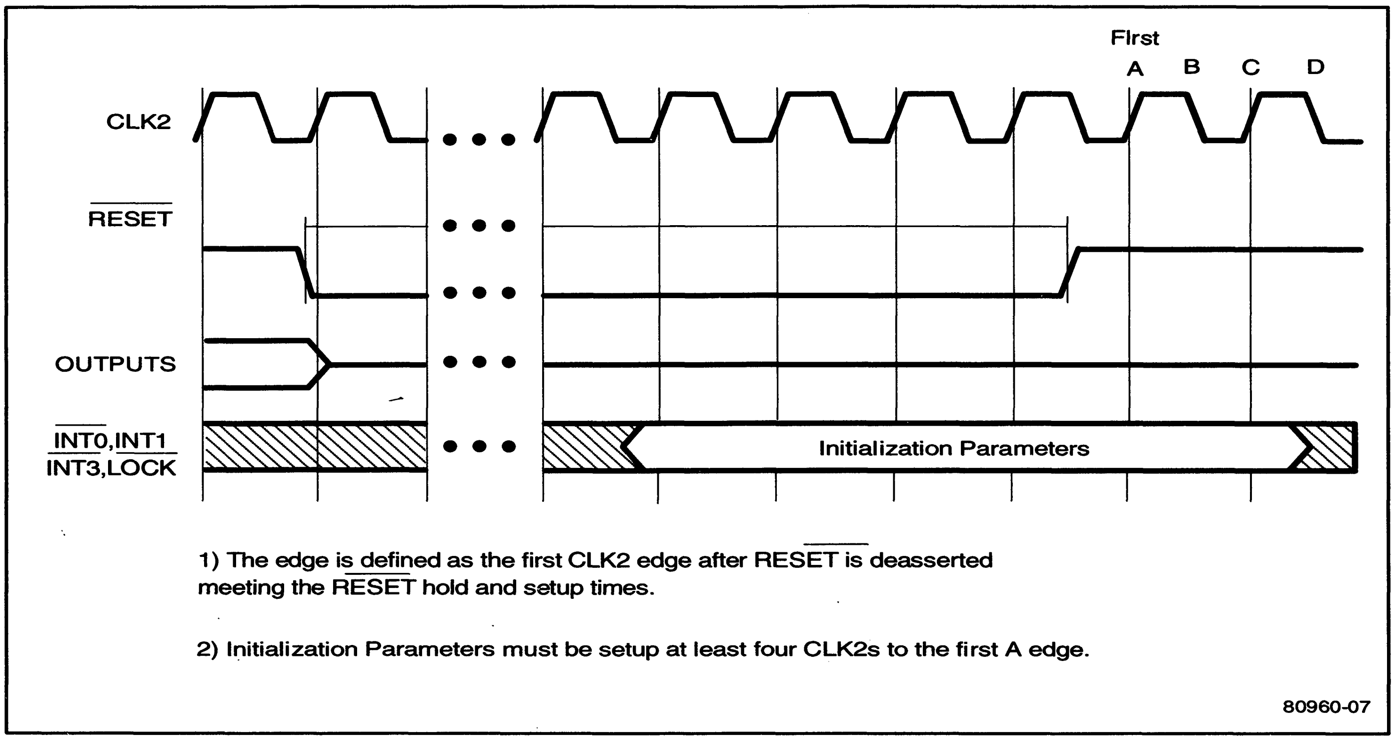

FIrst
A
B
C
D
CLK2
$\overline{\text{RESET}}$
OUTPUTS
$\overline{\text{INT0}}$,$\overline{\text{INT1}}$
INT3,$\overline{\text{LOCK}}$
Initialization Parameters
1) The edge is defined as the first CLK2 edge after $\overline{\text{RESET}}$ is deasserted meeting the $\overline{\text{RESET}}$ hold and setup times.
2) Initialization Parameters must be setup at least four CLK2s to the first A edge.
80960-07

**Figure 12-13: RESET Signal Timing Relationship**

### Reset Timing Generation

Figure 12-14 illustrates a typical synchronization circuit comprised of two D-type flip-flops. This circuitry generates the RESET input signal to the 80960SA/SB processor.

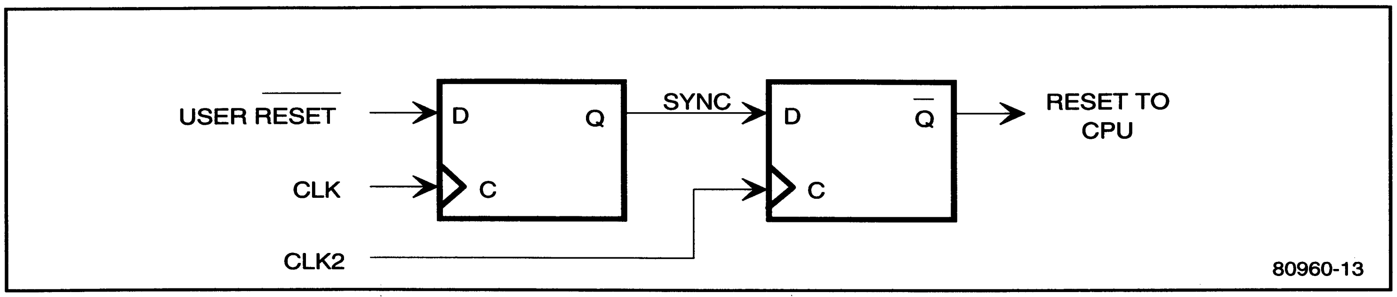


Figure 12-14: Asynchronous RESET Circuit

The timing diagram for these signals is shown in Figure 12-15. CLK or CLK2 can be used instead of CLK in Figure 12-15.

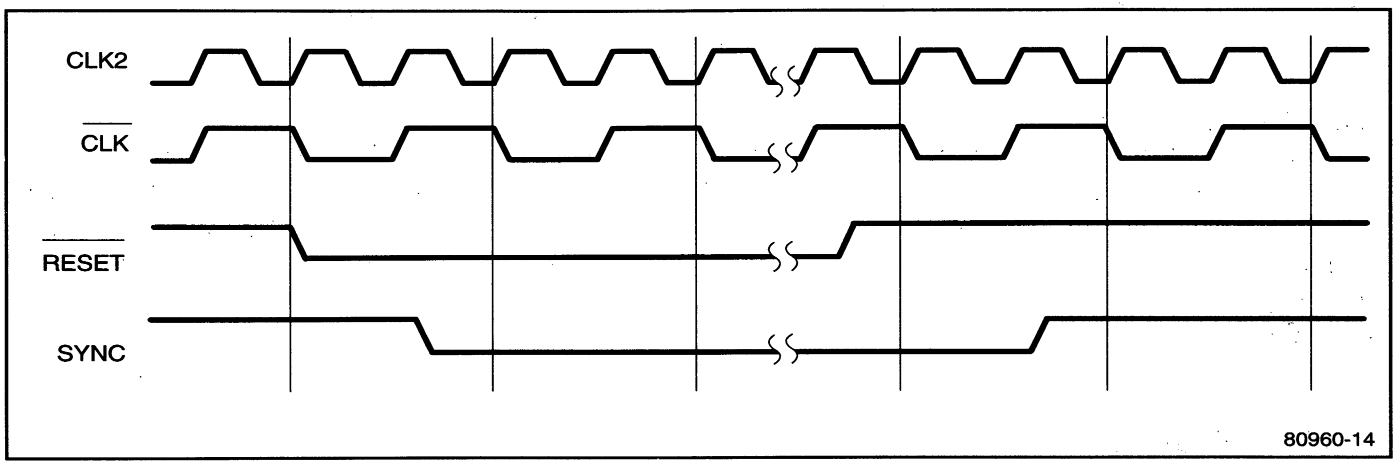


Figure 12-15: Timing Diagram for Reset Generation

## Initialization

Figure 3-4 illustrates the initialization sequence of events. When $\overline{\text{RESET}}$ is de-asserted, several actions occur: the processor samples the $\text{INT}_0$ and $\text{INT}_1$ pins, it asserts the BLAST/FAIL output signal (see next section for the pin description), and it performs the self-test (if required).

Table 12-2: $\overline{\text{RESET}}$ Initialization Conditions

| $\overline{\text{INT}_0}$ | $\text{INT}_1$ | $\overline{\text{INT}_3}$ | $\overline{\text{LOCK}}$ | Action taken |
|---|---|---|---|---|
| 1 | x | 1 | 1 | Run self-test (core initialization) |
| 0 | 0 | 1 | 1 | Disable self-test |
| 0 | 1 | x | x | Reserved |
| x | x | 0 | x | Reserved |
| x | x | x | 0 | ONCE mode |

Besides sampling $\overline{\text{INT}_0}$ and $\text{INT}_1$, the 80960SA/SB processor holds the $\overline{\text{BLAST/FAIL}}$ output signal asserted after $\overline{\text{RESET}}$ is de-asserted. The $\overline{\text{BLAST/FAIL}}$ signal remains asserted while the CPU performs the self-test. If the processor detects a failure during the self-test, $\overline{\text{BLAST/FAIL}}$ remains asserted and the CPU enters the stopped state. At this time, all outputs from the 80960SA/SB will be disabled (high-impedance or de-asserted). If the self-test completes successfully, the CPU de-asserts the $\overline{\text{BLAST/FAIL}}$ signal.

The 80960SA/SB processor proceeds with a checksum test of 16 words fetched from memory at physical address $00000000_{16}$. This ensures that the memory and bus operate correctly. If the checksum is incorrect, the $\overline{\text{BLAST/FAIL}}$ signal is reasserted and the 80960SA/SB processor enters the stopped state. After a successful checksum test, the 80960SA/SB processor uses some of the previously fetched words as addresses to initial data structures. See section in this chapter, "Initialization". Just prior to executing the first instruction, the 80960SA/SB processor clears any latched interrupt signals.

## ERROR SIGNALS

The 80960SA/SB processor provides an output signal ($\overline{\text{BLAST/FAIL}}$) and $\overline{\text{BE}_0}$-$\overline{\text{BE}_1}$ for notifications of an error within the processor.

| | |
|---|---|
| $\overline{\text{BLAST/FAIL}}$ | **Burst Last** indicates the last cycle ($T_d$) of an access. It is asserted low during the last $T_d$ cycle of any access.<br><br>**Initialization Failure** indicates that the processor has failed to initialize correctly. The failure state is indicated by a combination of $\overline{\text{BLAST}}$ asserted and both $\overline{\text{BE}}$ signals not asserted. This condition occurs after $\overline{\text{RESET}}$ is de-asserted and before the first bus transaction begins. $\overline{\text{FAIL}}$ is asserted while the processor performs a self-test. If the self-test completes successfully, then $\overline{\text{FAIL}}$ is de-asserted. Next, the processor performs a zero checksum on the first eight words of memory. If it fails, $\overline{\text{FAIL}}$ is asserted for a second time and remains asserted; if it passes, system initialization continues and $\overline{\text{FAIL}}$ remains de-asserted. |

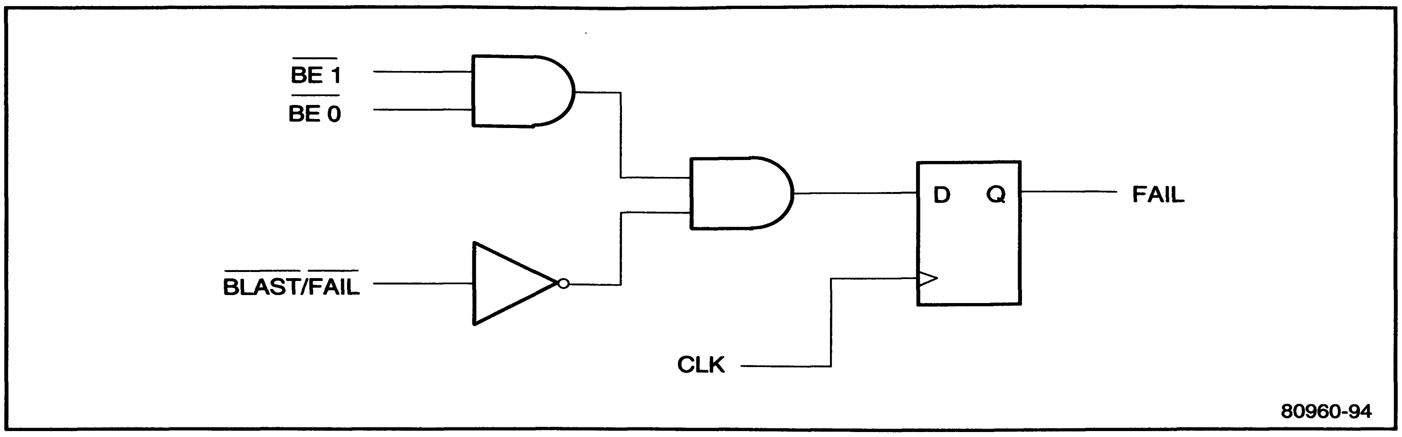


**Figure 12-16: Fail Detect Circuit**