M82289
BUS ARBITER
FOR M80286 PROCESSOR FAMILY

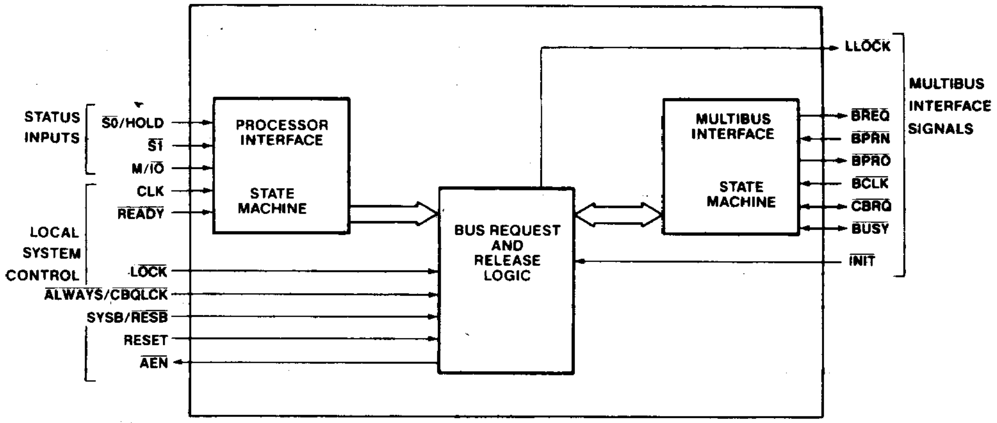
Military

- **Supports Multi-Master System Bus Arbitration Protocol**
- **Synchronizes M80286 Processor with Multi-Master Bus**
- **Compatible with Intel Bus Standard Multibus® (IEEE 796 Standard)**
- **Three Modes of Bus Release Operation for Flexible System Configuration**
- **Supports Parallel, Serial, and Rotating Priority Resolving Schemes**
- **Military Temperature Range: −55°C to +125°C (T_C)**
- **Available in a 20-pin Cerdip Package**

The Intel M82289 Bus Arbiter is a 5-volt, 20-pin HMOS* III component for use in multiple bus master M80286 systems. The M82289 provides a compact solution to system bus arbitration for the M80286 CPU.

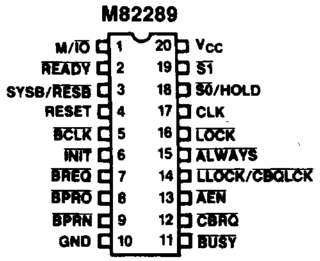
The complete IEEE 796 Standard bus arbitration protocol is supported. Three modes of bus release operation support a number of bus usage models.

*HMOS is a patented process of Intel Corporation.



271032-1

Figure 1. M82289 Block Diagram



271032-2

Figure 2. M82289 Pin Diagram

December 1989
Order Number: 271032-004

Table 1. M82289 Pin Description

Symbol	Pin(s)	Type	Name and Function
CLK	17	I	**SYSTEM CLOCK** accepts the CLK signal from the M82C284 Clock Generator chip as the timing reference for the bus arbiter and processor interface signals.
$\overline{S0}$/HOLD	18	I	**STATUS INPUT $\overline{S0}$ or HOLD** is either the $\overline{S0}$ status signal from M80286 or the HOLD signal from some other bus master. The function of this input is established during the processor reset of the M82289 Bus Arbiter. The M80286 $\overline{S0}$ pin meets the setup and hold time requirements of this pin. The $\overline{S0}$ pin function is selected by forcing this input high during the falling edge of processor reset. If the M82289 is used to support an M80286 processor, the $\overline{S0}$ output of the processor will be high during reset. In supporting the M80286 processor, the M82289 decodes the $\overline{S0}$ pin together with the other status input pins, $\overline{S1}$ and M/$\overline{IO}$, to determine the beginning of a processor bus cycle and initiate bus request and surrender actions. The HOLD function of the $\overline{S0}$/HOLD pin is selected by holding this input low during the falling edge of processor reset. When supporting a bus master other than M80286, the M82289 monitors the HOLD signal to initiate bus request and surrender actions.
$\overline{S1}$, M/$\overline{IO}$	19, 1	I	**STATUS INPUTS** are the status input signal pins from the M80286 processor. The arbiter decodes these inputs together with $\overline{S0}$/HOLD input to initiate bus request and surrender actions. A bus cycle is started when either $\overline{S1}$ or $\overline{S0}$ is sampled LOW at the falling edge of CLK. The M80286 $\overline{S1}$ and M/$\overline{IO}$ pins meet the setup and hold time requirements of these pins. (See M80286 Bus Cycle Status Encoding below.) When supporting the HOLD output of another bus master, the $\overline{S1}$ and M/$\overline{IO}$ pins must be held HIGH during T_S, the Status Cycle, for proper operation.
SYSB/$\overline{RESB}$	3	I	**SYSTEM BUS/$\overline{RESIDENT BUS}$** is an input signal which determines when the multi-master system bus is required for the current bus cycle. The signal can originate from address mapping circuitry such as a decoder or PROM attached to the processor address and status pins. The arbiter will request or retain control of the multi-master system bus when the SYSB/$\overline{RESB}$ pin is sampled HIGH at the end of the T_S bus state. During an interrupt acknowledge cycle, this input is sampled on every falling edge of CLK starting at the end of the T_S state until either SYSB/$\overline{RESB}$ is sampled HIGH or the bus cycle is terminated by the $\overline{READY}$ signal. Setup and hold times for this pin must be met for proper operation.

M80286 Bus Cycle Status Encoding

M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$/HOLD	Type of Bus Cycle
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	None; Bus Idle
1	0	0	Halt or Shutdown
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	None; Bus Idle

Table 1. M82289 Pin Description (Continued)

Symbol	Pin(s)	Type	Name and Function
$\overline{\text{READY}}$	2	I	**READY** is an active-LOW signal which indicates the end of the bus cycle. The M80286 halt or shutdown cycle does not require $\overline{\text{READY}}$ to terminate the bus cycle. Setup and hold times for this pin must be met for proper operation.
$\overline{\text{LOCK}}$	16	I	**$\overline{\text{LOCK}}$** is a processor-generated signal which when asserted (LOW) prevents the arbiter from surrendering the multi-master system bus to any other bus arbiter, regardless of its priority. $\overline{\text{LOCK}}$ is sampled by the arbiter at the end of the T_S (status) bus state. Setup and hold times for this pin must be met for proper operation.
$\overline{\text{ALWAYS}}$/ $\overline{\text{CBQLCK}}$	15	I	**$\overline{\text{ALWAYS RELEASE}}$ or $\overline{\text{COMMON BUS REQUEST LOCK}}$** can be programmed at processor reset to be either the ALWAYS RELEASE ($\overline{\text{ALWAYS}}$) strapping option or the COMMON BUS REQUEST LOCK ($\overline{\text{CBQLCK}}$) control input. Setup and hold times for this pin must be met for proper programming. When this pin is LOW during the falling edge of processor reset ($\overline{\text{ALWAYS}}$ option) the arbiter is programmed to surrender the multi-master system bus after each bus transfer cycle. The M82289 will remain in the ALWAYS RELEASE mode until it is reprogrammed during the next processor reset. The bus arbiter is programmed to support the COMMON BUS REQUEST LOCK function by forcing this input pin HIGH during the falling edge of the processor reset. $\overline{\text{CBQLCK}}$ itself is an active-LOW signal which when active prevents the arbiter from surrendering the multi-master system bus to a common bus request through the $\overline{\text{CBRQ}}$ input pin.
RESET	4	I	**PROCESSOR RESET** is an active-HIGH input synchronous to the system clock (CLK). RESET is the processor initialization of the arbiter to release the multi-master bus and clear any pending request.
$\overline{\text{INIT}}$	6	I	**$\overline{\text{INITIALIZE}}$** is an active-LOW Multibus signal used to reset all arbiters on the Multibus system. It will cause the release of the multi-master bus, but will not clear the pending bus master request so that the arbiter can again request the multi-master bus. No arbiters have the use of the multi-master bus immediately after initialization. $\overline{\text{INIT}}$ is an asynchronous signal to CLK.
$\overline{\text{BCLK}}$	5	I	**$\overline{\text{BUS CLOCK}}$** is the multi-master system bus clock to which the multi-master bus interface signals are synchronized. $\overline{\text{BCLK}}$ can be asynchronous to CLK.
$\overline{\text{BREQ}}$	7	O	**$\overline{\text{BUS REQUEST}}$** is an active-LOW output signal used in the parallel and rotating priority resolving schemes. The arbiter activates $\overline{\text{BREQ}}$ to request the use of the multi-master system bus. The arbiter holds $\overline{\text{BREQ}}$ active as long as it is requesting or has possession of the multi-master system bus.
$\overline{\text{CBRQ}}$	12	I/O (open-drain)	**$\overline{\text{COMMON BUS REQUEST}}$** is a Multibus signal that indicates when an arbiter is requesting the Multibus. This pin is an open-drain input/output requiring an external pullup resistor. As an input $\overline{\text{CBRQ}}$ indicates that another arbiter is requesting the multi-master system bus. The input function of this pin is enabled by the $\overline{\text{CBQLCK}}$ signal. Setup and hold times for this pin must be met for proper operation. As an output $\overline{\text{CBRQ}}$ is asserted to indicate that this arbiter is requesting the Multibus. The arbiter pulls $\overline{\text{CBRQ}}$ low when it issues a $\overline{\text{BREQ}}$. The arbiter releases $\overline{\text{CBRQ}}$ when it obtains the Multibus.

Table 1. **M82289 Pin Description** (Continued)

Symbol	Pin(s)	Type	Name and Function
BPRN	9	I	**BUS PRIORITY IN** is an active-LOW input indicating that this arbiter has the highest priority of any arbiter requesting the system bus. BPRN HIGH signals the arbiter that a higher priority arbiter is requesting or has possession of the system bus. Setup and hold times for this pin must be met for proper operation.
BPRO	8	O	**BUS PRIORITY OUT** is an active-LOW output signal used in the serial priority resolving scheme. BPRO is connected to BPRN of the next lower priority to grant or revoke priority from that arbiter.
BUSY	11	I/O (open-drain)	**BUSY** is a Multibus signal which is asserted when the system bus is in use. BUSY is an open drain input/output requiring an external pullup resistor. As an input BUSY asserted indicates when the Multibus is in use. Setup and hold times must be met for proper operation. As an output BUSY is asserted to signal when this arbiter has taken control of the Multibus.
AEN	13	O	**ADDRESS ENABLE** is the output of the arbiter which goes directly to the processor's address latches, the M82C288 Bus Controller and the M82C284 Clock Generator. AEN asserted causes the bus controller and address latches to enable their output drivers. AEN also drives the clock generator ARDYEN input to enable its asynchronous ready input (ARDY). AEN can also be used as an active-LOW Hold Acknowledge to a bus master other than M80286. It signals to the bus master that control of the system bus has been relinquished when AEN is inactive (HIGH). Note that AEN goes active relative to BCLK and goes inactive relative to CLK.
LLOCK	14	O	**LEVEL LOCK** is an active-LOW output signal decoded from processor LOCK signal. LLOCK can be used as Multibus LOCK when buffered with a tri-state buffer enabled by the AEN signal. LLOCK will be cleared by RESET but not by INIT.
V_{CC}	20	I	+5 volts supply voltage.
GND	10	I	Ground.

FUNCTIONAL DESCRIPTION

The M82289 Bus Arbiter in conjunction with the M82C288 Bus Controller and the M82C284 Clock Generator interfaces the M80286 processor or some other bus master to a multi-master system bus. The arbiter multiplexes a processor onto a multi-master system bus. It avoids contention with other bus masters.

The M82289 has two separate state machines which communicate through bus request and release logic. The processor interface state machine is synchronous with the local system clock (CLK) and the multi-master system bus interface state machine is synchronous with the bus clock (BCLK).

The M82289 performs all signalling to request, obtain, and release the system bus. External logic is used to determine which bus cycles require the system bus and to resolve priorities of simultaneous requests for control of the system bus.

M82289 with M80286

In a M80286 system using an M82289 Bus Arbiter, the M80286 processor is unaware of the arbiter's existence and issues commands as though it had exclusive use of the multi-master system bus such as Multibus. If the processor cycle requires Multibus access, the arbiter requests control of the Multibus. Until the request is granted the M82289 keeps AEN disabled to prevent the M82C288 Bus Controller and the address latches from accessing the Multibus. AEN inactive also disasserts the asynchronous ready enable (ARDYEN) input of the M82C284 clock chip so that the system bus will appear as "NOT READY" to the M80286 processor.

Once the M82289 Bus Arbiter has acquired the bus, it will assert $\overline{AEN}$ allowing the M82C288 Bus Controller and the address latches to access the system bus and assert the $\overline{ARDYEN}$ input of the M82C284 Clock chip.

Typically, once the data transfer command has been issued by the M82C288 and the data transfer has taken place, a transfer acknowledge ($\overline{XACK}$) signal is returned to the processor on the multi-master system bus to indicate "Ready" from the accessed slave device. The processor remains in a series of "Wait States" (Repeated T_C states) until the addressed device responds with $\overline{XACK}$ asserted signal to the M82C284 $\overline{ARDY}$ input and the M82C284 asserts $\overline{READY}$ to the processor. The processor then completes its bus cycle.

M82289 with Other Bus Masters

When supporting other bus masters, the $\overline{S0}$/HOLD and $\overline{READY}$ pins of the bus arbiter can be connected to the 'HOLD' pin of that master. The inverted $\overline{AEN}$ signal from the M82289 can be used as the hold acknowledge (HLDA) input for the other bus master.

The bus master sends a HOLD signal to the bus arbiter when it needs the system bus for a memory access. If the arbiter currently controls the system bus, $\overline{AEN}$ will be active. Otherwise, $\overline{AEN}$ will be inactive and the arbiter will request control of the system bus. The bus master will have to wait until the M82289 has asserted $\overline{AEN}$ (LOW), before it starts its bus cycle.

When the bus master no longer requires the Multibus it will have to inactivate the HOLD signal. The arbiter interprets the Multibus access as a single bus cycle which is terminated by HOLD going inactive (LOW). Thus the arbiter will not release the Multibus to any other bus master during a bus access cycle.

Processor Cycle Definition

Any M80286 system which gains access to the Multibus through the M82289 Bus Arbiter uses an internal clock which is one half the frequency of the system clock (CLK) (see Figure 3). Knowledge of the phase of the local bus master internal clock is required for proper M82289 control of the M80286 interface to Multibus. The local bus master informs the bus arbiter of its internal clock phase when it asserts the status signals. The M80286 $\overline{S0}$ and $\overline{S1}$ status signals are always first asserted in phase 1 of the local bus master's internal clock.

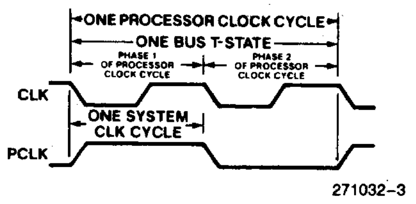

ONE PROCESSOR CLOCK CYCLE
ONE BUS T-STATE
PHASE 1 OF PROCESSOR CLOCK CYCLE
PHASE 2 OF PROCESSOR CLOCK CYCLE
CLK
ONE SYSTEM CLK CYCLE
PCLK
271032-3

Figure 3. CLK Relationship to Internal Processor Phase and Bus T-States

Bus State Definition

The M82289 Bus Arbiter has three processor bus states (see Figure 4): Idle (T_I), Status (T_S), Command (T_C). Each bus state is two CLK cycles long. Bus state phases correspond to the internal CPU processor clock phases.

Bus Cycle Definition

The $\overline{S1}$ and $\overline{S0}$ status inputs are sampled by the M82289 on the falling edge of CLK and signal the start of a bus cycle by going active (LOW). The T_S bus state is defined to be the two CLK cycles during which either $\overline{S1}$ or $\overline{S0}$ is active (see Figure 5). When either $\overline{S1}$ or $\overline{S0}$ is sampled LOW, the next CLK cycle is considered the second phase of the associated processor clock cycle.

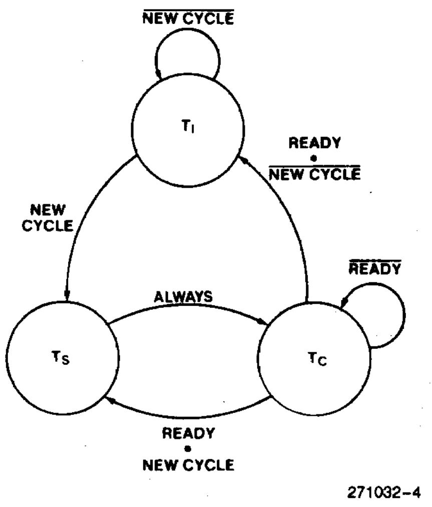


Figure 4. M82289 Processor Bus States

The arbiter enters the T_C bus state after the T_S state. The shortest bus cycle may have one T_S state and one T_C state. Longer bus cycles are formed by repeating T_C states. A repeated T_C bus state is called a wait state.

The $\overline{READY}$ input determines whether the current T_C bus state is to be repeated. The $\overline{READY}$ input has the same timing and effect for all bus cycles. $\overline{READY}$ is sampled at the end of each T_C bus state to see if it is active. If sampled HIGH, the T_C bus state is repeated. This is called inserting a wait state.

When $\overline{READY}$ is sampled LOW, the current bus cycle is terminated. Note that the bus arbiter may enter the T_S bus state directly from T_C if the status lines are sampled active (LOW) at the next falling edge of CLK (see Figure 5). If neither of the status lines are sampled active at that time the M82289 will enter the T_I bus state. The T_I bus state will be repeated until the status inputs are sampled active.

Arbitration Between Bus Masters

The Multibus protocol allows multiple processing elements to compete with each other to access common system resources. Since the local M80286 processor does not have exclusive use of the system bus, if the Multibus is "BUSY" the M80286 processor will have to wait before it can access the system bus.

The M82289 Bus Arbiter provides an integrated solution for controlling access to a multi-master system bus. The bus arbiter allows both higher and lower priority bus masters to acquire the system bus depending on which release mode is used. In general, higher priority masters obtain the bus immediately after any lower priority master completes its present transfer cycle. Lower priority bus masters obtain the bus when a higher priority master is not accessing the system bus or the proper surrender conditions exist. The M82289 handles this arbitration in a manner completely transparent to the bus master (e.g. M80286 processor).

At the end of each transfer, the arbiter may retain or release the system bus. This decision is controlled by the processor state, bus arbitration inputs and arbiter strapping options. (See Releasing the Multibus, ahead.)

Priority Resolving Techniques

Some means of resolving priority between bus masters requesting the multi-master bus simultaneously must be provided. The M82289 Bus Arbiter supports parallel, serial, and rotating system bus priority resolving techniques. All of these techniques are based on the concept that at a given time, one bus master will have priority above all the others.

An individual arbiter is the highest priority arbiter requesting the Multibus when its $\overline{BPRN}$ input is asserted (LOW). The highest priority requesting arbiter cannot immediately seize the system bus. It must wait until the present bus transaction is completed. Upon completing its current transaction the present bus owner surrenders the bus by releasing $\overline{BUSY}$.

$\overline{BUSY}$ is an active-LOW 'Wired-OR' Multibus signal which goes to every bus arbiter on the system bus. When $\overline{BUSY}$ goes inactive, the arbiter which has requested the system bus, *and* presently has bus priority ($\overline{BPRN}$ LOW), seizes the bus by pulling $\overline{BUSY}$ LOW (see waveform in Figure 6).

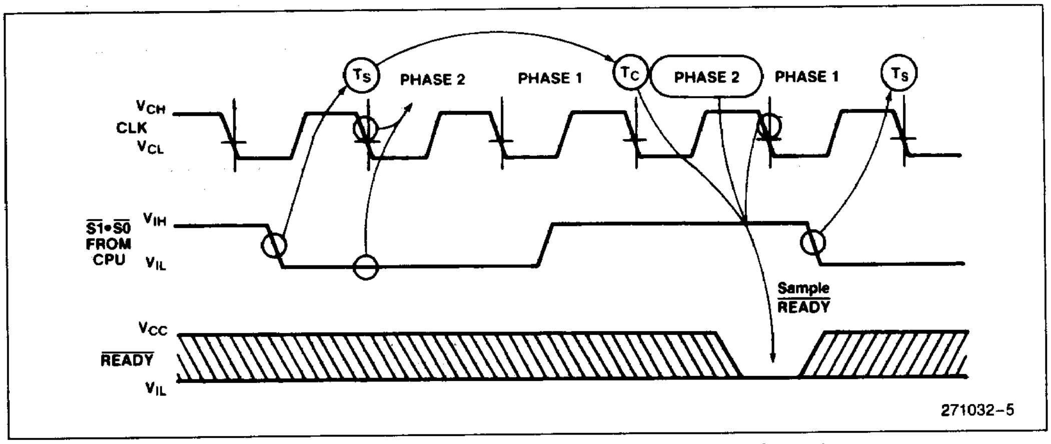


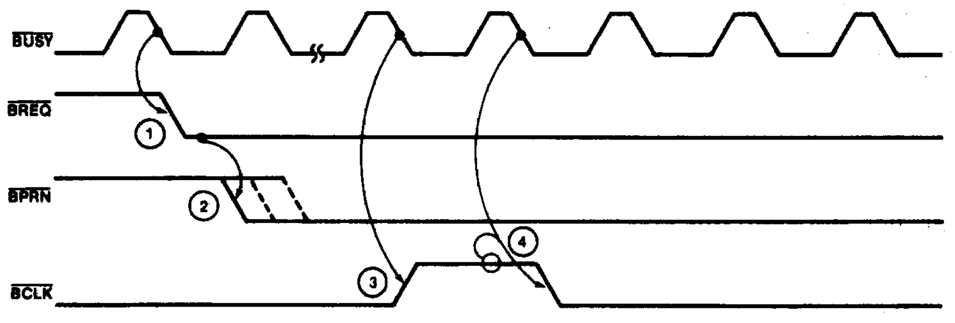
Figure 5. M80286 Bus Cycle Definition (Without Wait States)

The generation of a multi-master bus request ($\overline{BREQ}$) is controlled by the type of bus cycle and the SYSB/$\overline{RESB}$ input. Whenever the processor signals the status for memory read, memory write, I/O read, I/O write or interrupt acknowledge cycle, and SYSB/$\overline{RESB}$ is HIGH at the end of T_S, a bus request is generated.

When the status inputs indicate an interrupt acknowledge bus cycle, the arbiter allows external logic to decide (through the SYSB/$\overline{RESB}$ input) whether the interrupt acknowledge cycle should use the Multibus.

Figure 7 shows how SYSB/$\overline{RESB}$ is repeatedly sampled until it is sampled HIGH or the bus cycle is terminated. If the bus cycle is completed ($\overline{READY}$ is sampled LOW) before SYSB/$\overline{RESB}$ is sampled HIGH, the arbiter will not request the Multibus.

The M82289 bus Arbiter does not generate a separate $\overline{BREQ}$ for each bus cycle. Instead the M82289 generates $\overline{BREQ}$ when it requests the bus and holds $\overline{BREQ}$ active during the time that it has possession of the bus. Note that all multi-master system bus requests (via $\overline{BREQ}$) are synchronized to the system bus clock ($\overline{BCLK}$).



NOTES:
1. Higher priority bus arbiter requests the multi-master system bus.
2. Attains priority. (Does not yet own bus.)
3. Lower priority bus arbiter releases $\overline{BUSY}$.
4. Higher priority bus arbiter then acquires the bus and pulls $\overline{BUSY}$ low.

Figure 6. Bus Exchange Timing For The Multibus

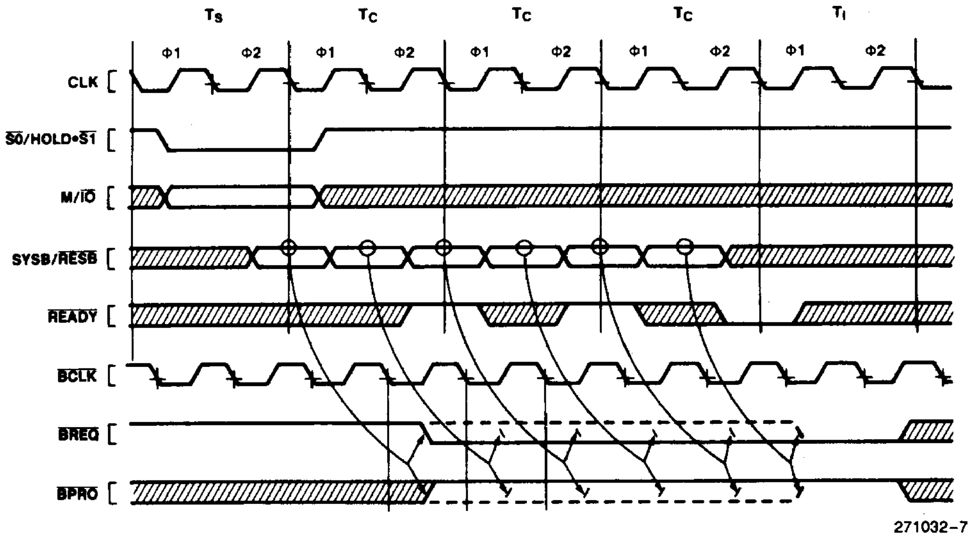


Figure 7. Bus Request Timing During an Interrupt Acknowledge Cycle

Parallel Priority Resolving Technique

The parallel priority resolving technique requires a separate bus request line ($\overline{\text{BREQ}}$) for each arbiter on the multi-master system bus (see Figure 8). Each $\overline{\text{BREQ}}$ line enters a priority encoder which generates the binary address of the highest priority $\overline{\text{BREQ}}$ line currently active. The binary address is decoded to select the $\overline{\text{BPRN}}$ line corresponding to the highest priority arbiter requesting the bus. In a parallel scheme, the $\overline{\text{BPRO}}$ output is not used.

The arbiter receiving priority ($\overline{\text{BPRN}}$ LOW) then allows its associated bus master onto the multi-master system bus as soon as the bus becomes available (i.e., the bus is no longer busy). Any number of bus masters may be accommodated in this way, limited only by the complexity of the external priority resolving circuitry. Such circuitry must resolve the priority within one $\overline{\text{BCLK}}$ period.

Serial Priority Resolving Technique

The serial priority resolving technique eliminates the need for the priority circuitry of the parallel technique by daisy-chaining the bus arbiters together, that is, connecting the higher priority arbiter's $\overline{\text{BPRO}}$ output to the $\overline{\text{BPRN}}$ of the next lower priority arbiter (see Figure 9). The highest priority bus arbiter would have its $\overline{\text{BPRN}}$ tied LOW in this configuration, signifying to the arbiter that it always has the highest priority when requesting the system bus. In a serial scheme, the $\overline{\text{BREQ}}$ output is not used.

Since arbitration must be resolved within one $\overline{\text{BCLK}}$ period the number of arbiters connected together in the serial priority is limited by arbiter $\overline{\text{BPRN}}$ to $\overline{\text{BPRO}}$ propagation delay (18 ns). For a 10 MHz Multibus $\overline{\text{BCLK}}$, five M82289 Bus Arbiters may be connected together in serial configuration.

Maximum number of chained-priority devices =

$$\frac{\overline{\text{BCLK}}\text{ period}}{\overline{\text{BPRN}}\text{ to }\overline{\text{BPRO}}\text{ delay}}$$

When using the serial priority resolving scheme, a higher priority arbiter (for example, arbiter 2, Figure 9) passes priority to the next lower priority arbiter

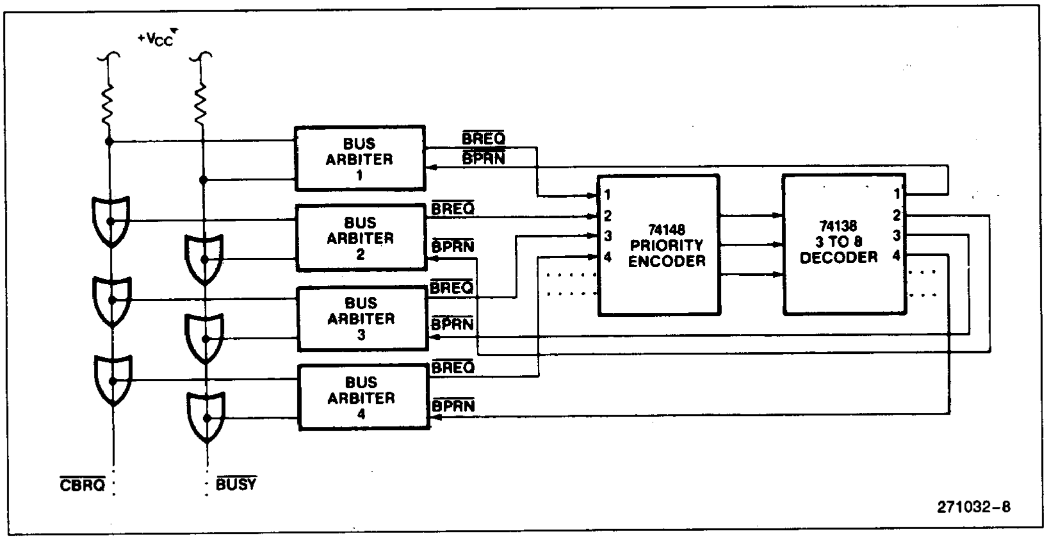


Figure 8. Parallel Priority Resolving Technique

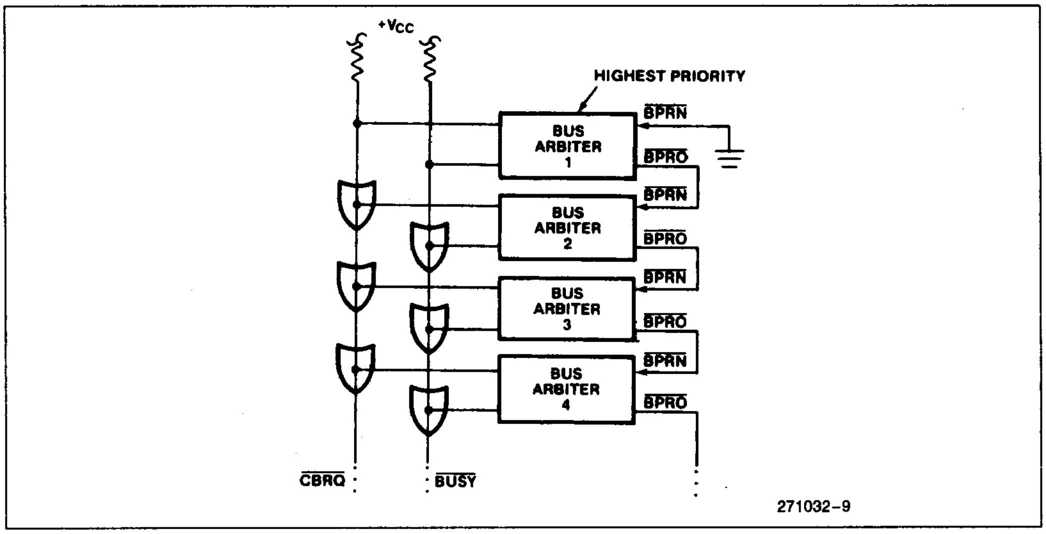
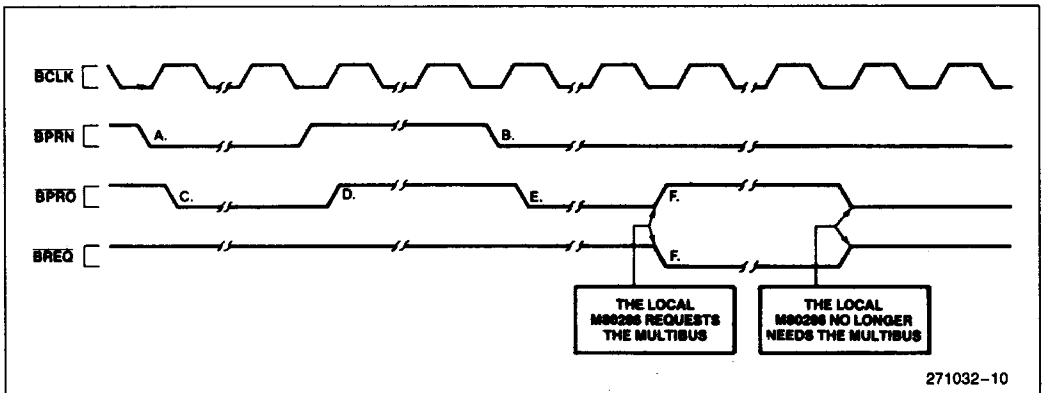


Figure 9. Connections for Serial Priority Resolving Technique



NOTE:
Events A through F described above.

Figure 10. Serial Priority Bus Behavior

(arbiter 3) by asserting its $\overline{\text{BPRO}}$ signal (LOW). This asserts $\overline{\text{BPRN}}$ of next arbiter (arbiter 3) as shown in Figure 10A and 10B. An arbiter's $\overline{\text{BPRO}}$ is asserted if the arbiter has priority ($\overline{\text{BPRN}}$ is asserted) but is not accessing or requesting the system bus (as indicated by $\overline{\text{BREQ}}$ inactive as shown in Figure 10C and 10E for arbiter 3). Whenever a higher priority arbiter (arbiter 3) issues a bus request its $\overline{\text{BPRO}}$ goes inactive causing the next lower priority arbiter (arbiter 4) to lose its bus priority (Figure 10F). Any arbiter (arbiter 3) will also bring its $\overline{\text{BPRO}}$ inactive if its $\overline{\text{BPRN}}$ goes inactive (from arbiter 2), thereby passing the loss of bus priority onto the lower priority arbiters (e.g. arbiter 4) as shown in Figure 10D.

Rotating Priority Resolving Technique

The rotating priority resolving technique is similar to the parallel priority resolving technique except that priority is dynamically re-assigned. The priority encoder is replaced by a more complex circuit which rotates priority between requesting arbiters, thus allowing each arbiter an equal chance to use the multi-master system bus over a given period of time.

Selecting the Appropriate Priority Resolving Technique

The choice of a priority resolving technique involves a tradeoff between external logic complexity and ease of Multibus access for the different bus masters in the system. The rotating priority resolving technique requires a substantial amount of external logic, but guarantees all the bus masters an equal opportunity to access the system bus. The serial priority resolving technique uses no external logic but has fixed bus master priority levels and can accommodate only a limited number of bus arbiters. The parallel priority resolving technique is in general a compromise between the other two techniques. (For example parallel priority configuration in Figure 8 allows up to eight arbiters to be present on the Multibus, with fixed priority levels, while not requiring a large amount of complex external logic to implement.)

Releasing the Multibus

Following a data transfer cycle on the Multibus, the M82289 Bus Arbiter can either retain control of the system bus or release the bus for use by some other bus master. The M82289 can operate in one of three modes, defining different conditions under which the arbiter relinquishes control of the multi-master system bus. These release modes are described in Table 2.

Table 2. M82289 Release Modes

Release Mode	Conditions Under Which the Bus Arbiter Releases the System Bus (Unless Cycles Are LOCKed)
Mode 1	The Bus Arbiter always releases the bus at the end of each transfer cycle.
Mode 2	The Bus Arbiter retains the bus until: • a higher-priority bus master requests the bus, driving $\overline{\text{BPRN}}$ HIGH. • a lower-priority bus master requests the bus by pulling $\overline{\text{CBRQ}}$ LOW.
Mode 3	The Bus Arbiter retains the bus until: • a higher-priority bus master requests the bus, driving $\overline{\text{BPRN}}$ HIGH. ($\overline{\text{CBRQ}}$ LOW ignored)

If the arbiter was programmed to operate in the Always Release mode (Mode 1) during the previous reset, it will surrender the Multibus after each complete transfer cycle. If the arbiter is not in the Always Release mode, it will not surrender the bus until the local M80286 processor enters a halt state, the arbiter is forced off of the bus by the loss of $\overline{\text{BPRN}}$ (Mode 2 or 3), or by a common bus request when the $\overline{\text{CBRQ}}$ input is enabled by the $\overline{\text{CBQLCK}}$ input (Mode 2).

$\overline{\text{CBRQ}}$ can save the bus exchange overhead in many cases. If $\overline{\text{CBRQ}}$ is high, it indicates to the bus master that no other master is requesting the bus and therefore the present bus master can retain the bus. Without $\overline{\text{CBRQ}}$, only $\overline{\text{BPRN}}$ indicates whether or not another master is requesting the bus and, that only if the other master is of higher priority. Between the master's bus transfer cycles, in order to allow lower priority masters to take the bus if they need it, the master must give up the bus. At the start of the master's next transfer cycle, the bus must be regained. If no other master has the bus, this can take approximately two $\overline{\text{BCLK}}$ periods. To avoid this overhead of unnecessarily giving up and regaining the bus when no other masters need it, $\overline{\text{CBRQ}}$ is extremely useful. Any master that wants but does not have the bus, must assert $\overline{\text{CBRQ}}$ (LOW). If $\overline{\text{CBRQ}}$ line is not asserted the bus does not have to be released, thereby eliminating the delay of regaining the bus at the start of the next cycle.

The $\overline{\text{LOCK}}$ input to the arbiter can be used to override any of the conditions shown in Table 2. While $\overline{\text{LOCK}}$ is asserted, the arbiter will not surrender control of the Multibus to any other requesting arbiter. Note that the arbiter will surrender the Multibus (synchronous to $\overline{\text{BCLK}}$) either in response to RESET or $\overline{\text{INIT}}$ signals independent of the current release mode or the state of the arbiter inputs.

The three bus release modes have the same operation when supporting either the M80286 processor or some other bus master.

Selecting the Appropriate Release Mode

The choice of which release mode to use may affect the bus utilization of the individual subsystems, and the system as a whole. Mode dependent performance variations are due to the bus acquisition/release overhead. The effect of these acquire and release times on system bus efficiency is illustrated in Figure 11.

An isolated transfer on the multi-master system bus is depicted in Figure 11A. Figure 11B shows utilization for the bus arbiter operating in Mode 1. The arbiter must request and release the system bus for each transfer cycle. Lower priority arbiters have easy access to the system bus, but overall bus efficiency is low. Bus utilization for a bus arbiter operating in Mode 2 or 3 is shown in Figure 11C. In this situation the arbiter acquires the bus once for a sequence of transfers. The arbiter retains the bus until forced off by another bus master's request as defined in Table 2.

The three release modes of the M82289 allow the designer to optimize the system use of the Multibus.

Configuring the M82289 Release Mode

The M82289 Bus Arbiter can be configured in any of its three bus release modes without additional hardware. The M82289 can also be configured to switch between Mode 2 and Mode 3 under software control

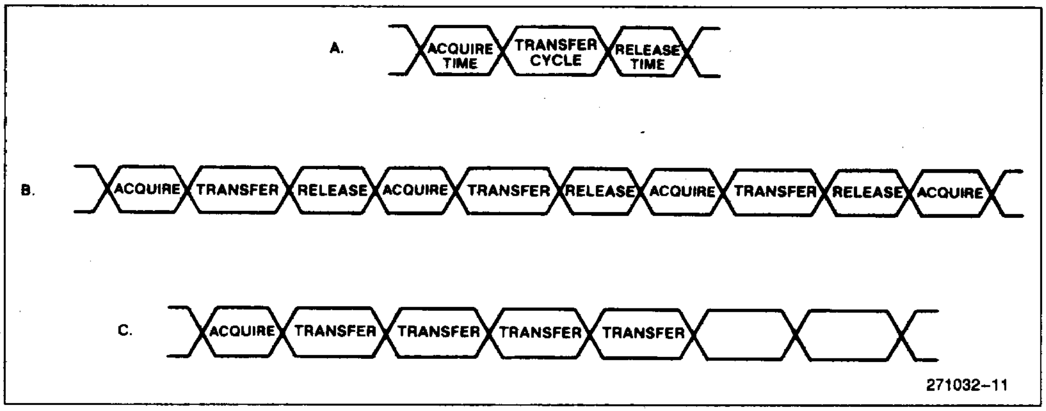


Figure 11. Effects of Bus Contention on Bus Efficiency

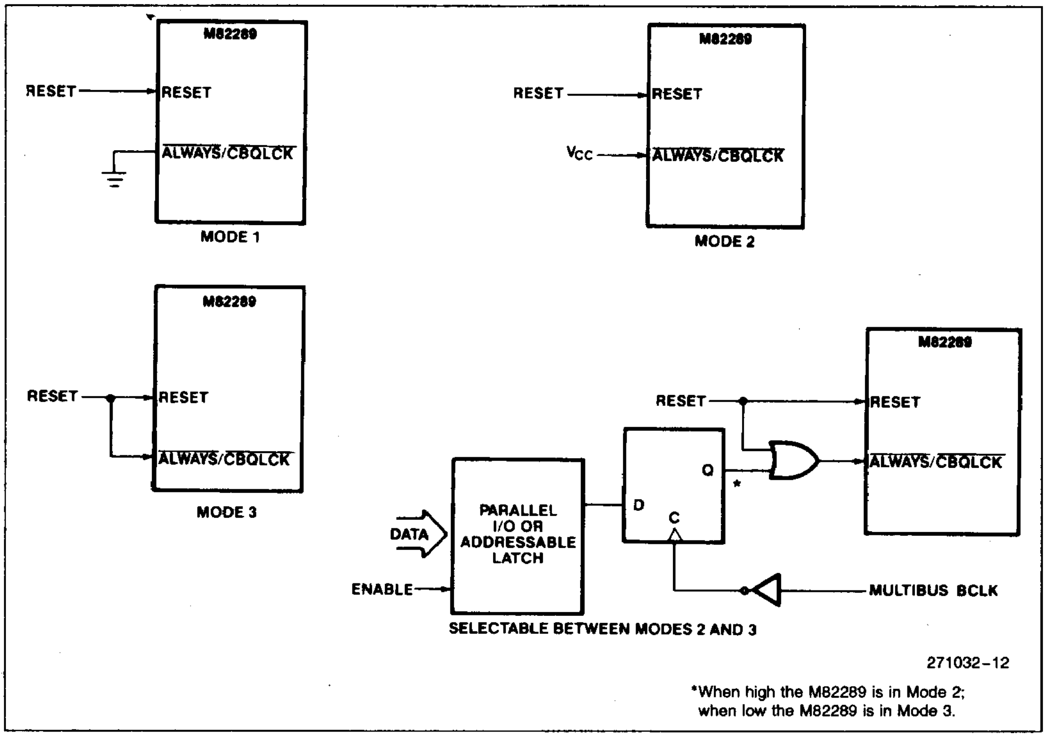


Figure 12. M82289 Release Mode Configurations

of the M80266 processor, requiring that a parallel port or addressable latch be used to drive the ALWAYS/$\overline{\text{CBQLCK}}$ INPUT pin of the M82289 (see Figure 12).

Asserting the $\overline{\text{LOCK}}$ Signal

Independent of the particular release mode of the M82289 Bus Arbiter, the M80286 processor can assert a $\overline{\text{LOCK}}$ signal synchronously to CLK to prevent the arbiter from releasing the Multibus. This software-controlled $\overline{\text{LOCK}}$ signal prevents the M82289 from surrendering the system bus to any other bus master, whether that bus master is of higher or lower priority. The $\overline{\text{LOCK}}$ signal is typically used for implementing software semaphores for shared resources or for critical processes that must run in real-time.

The M82289 $\overline{\text{LLOCK}}$ output is the Multibus signal asserted during all bus cycles which are locked together. The $\overline{\text{LLOCK}}$ is set or reset depending on processor $\overline{\text{LOCK}}$ at the end of the T_S cycle. The $\overline{\text{LLOCK}}$ will delay going inactive until the termination of the current transfer cycle.

The M82289 will continue to assert the $\overline{\text{LLOCK}}$ signal, retaining control of the Multibus, until the end of the first 'unLOCKed' M80286 bus cycle (M80286 disables its $\overline{\text{LOCK}}$ output on the last bus cycle indicating that no future locked cycles are needed). While the $\overline{\text{LOCK}}$ signal will force the arbiter presently in control to hold the system bus, it cannot force another arbiter to surrender the bus any earlier than it normally would.

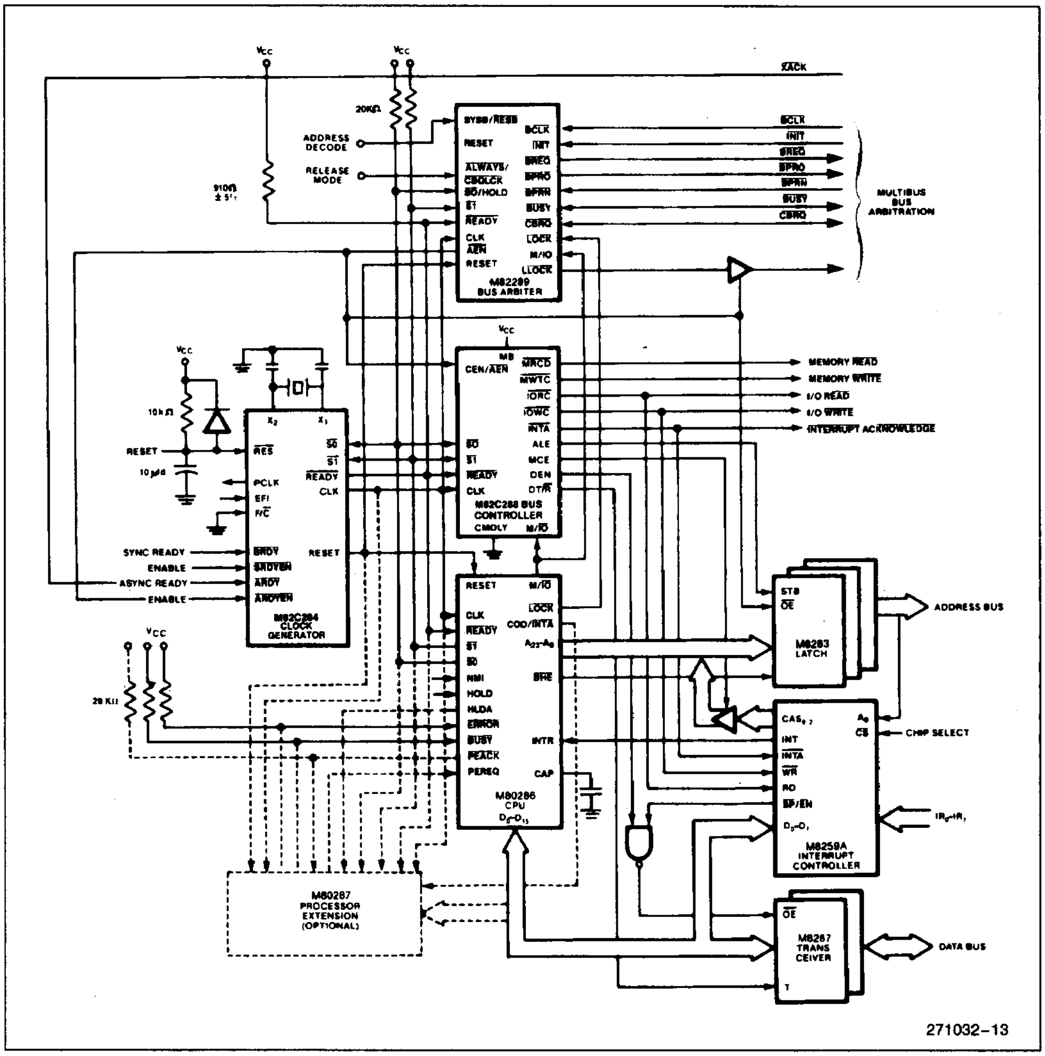
The $\overline{\text{LLOCK}}$ signal from the M82289 must be connected to a tri-state buffer in order to drive the Multibus $\overline{\text{LOCK}}$ signal. This tri-state buffer should be enabled by the $\overline{\text{AEN}}$ signal from the arbiter going active.

M82289 Reset and Initialization

The M82289 Bus Arbiter provides the RESET and $\overline{\text{INIT}}$ pins for initialization. RESET is a CLK synchronous signal from the M80286 processor and $\overline{\text{INIT}}$ is an asynchronous signal on the multi-master system bus. By having RESET pin high or $\overline{\text{INIT}}$ pin low, the $\overline{\text{BREQ}}$, $\overline{\text{BUSY}}$, and $\overline{\text{AEN}}$ output pins will all be cleared and become inactive. RESET will also clear the $\overline{\text{LLOCK}}$ signal. Unlike RESET, $\overline{\text{INIT}}$ will not clear any pending bus request; the bus request would be asserted after the $\overline{\text{INIT}}$ signal goes inactive.

Note that when the M82289 is initialized by the RESET input it does not wait until the end of the current bus cycle to reset. Any bus cycle in process when RESET goes active will be aborted by the arbiter. Although the $\overline{\text{INIT}}$ signal will also interrupt an active bus cycle, the arbiter can request the Multibus and complete the bus cycle when $\overline{\text{INIT}}$ goes inactive.

As mentioned in the Table 1 Pin Description and Figure 12, the functions of the $\overline{\text{S0}}$/HOLD pin and the release mode (ALWAYS/$\overline{\text{CBQLCK}}$ pin) are programmed at the falling edge of RESET.



Schematic 1. Typical M80286 Subsystem MULTIBUS® Interface

ABSOLUTE MAXIMUM RATINGS*

Storage Temperature .... −65° to +150°C
Case Temperature Under Bias... −55°C to +125°C
Voltage on Any Pin With
Respect to GND .... −0.5V to +7V
Power Dissipation .... 1 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

ELECTRICAL CHARACTERISTICS AND WAVEFORMS

Operating Conditions

Symbol	Description	Min	Max	Units
T_C	Case Temperature (Instant On)	−55	+125	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V

D.C. CHARACTERISTICS (Over Specified Operating Conditions)

Symbol	Parameter	Min	Max	Units	Comments
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} +0.5	V	
V_{ILC}	CLK Input Low Voltage	−0.5	0.6	V	
V_{IHC}	CLK Input High Voltage	3.8	V_{CC} +1.0	V	
V_{OL}	Output Low Voltage: $\overline{BUSY}$, $\overline{CBRQ}$ $\overline{BPRO}$, $\overline{BREQ}$, $\overline{AEN}$ $\overline{LLOCK}$		 0.45 0.45 0.45	 V V V	 I_{OL} = 32 mA I_{OL} = 16 mA I_{OL} = 5 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{LI}	Input Leakage Current		±10 ±1	μA mA	$0.45V \leq V_{IN} \leq V_{CC}$ $0V \leq V_{IN} < 0.45V$
I_{LO}	Output Leakage Current		±10	μA	$0.45V \leq V_{OUT} \leq V_{CC}$
I_{CC}	Power Supply Current		120	mA	
C_{CLK}	CLK, $\overline{BCLK}$ Input Capacitance		12	pF	F_C = 1 MHz
C_{IN}	Input Capacitance		10	pF	F_C = 1 MHz
C_O	Input/Output Capacitance		20	pF	F_C = 1 MHz

A.C. CHARACTERISTICS (Over Specified Operating Conditions)

AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

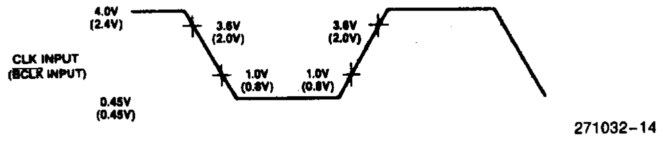
Sym	Parameter	8 MHz		Units	Comments	Shown In Figure
		Min	Max			
1	CLK Cycle Period	62	t5+ 50	ns		13
2	CLK Low Time	15	230	ns	at 1.0V	13
3	CLK High Time	20	235	ns	at 3.6V	13
4	CLK Rise/Fall Time		10	ns	1.0 to 3.6V	13
5	$\overline{\text{BCLK}}$ Cycle Time	100	∞	ns		13
6	$\overline{\text{BCLK}}$ High/Low Time	30		ns		13
7	$\overline{\text{S0}}$/HOLD, $\overline{\text{S1}}$, M/$\overline{\text{IO}}$ Setup	22		ns		13
8	$\overline{\text{S0}}$/HOLD, $\overline{\text{S1}}$, M/$\overline{\text{IO}}$ Hold	1		ns		13
9	$\overline{\text{READY}}$ Setup	38		ns		13
10	$\overline{\text{READY}}$ Hold Time	25		ns		13
11	$\overline{\text{LOCK}}$, SYSB/$\overline{\text{RESB}}$ Setup Time	20		ns		13, 18
12	$\overline{\text{LOCK}}$, SYSB/$\overline{\text{RESB}}$ Hold Time	1		ns		13, 18
13	RESET Setup Time	20		ns		19
14	RESET Hold Time	1		ns		19
15	RESET ACTIVE Pulse Width	16		CLKs		19
16	$\overline{\text{INIT}}$ Setup Time	45		ns	Note 9	20
17	$\overline{\text{INIT}}$ Hold Time	1		ns	Note 9	20
18	$\overline{\text{INIT}}$ Active Pulse Width	3(t1) +3(t14)		ns		20
19	$\overline{\text{BUSY}}$, $\overline{\text{BPRN}}$, $\overline{\text{CBRQ}}$, $\overline{\text{CBQLCK}}$/$\overline{\text{ALWAYS}}$ Setup to $\overline{\text{BCLK}}$ (or to RESET)	20		ns		13, 15, 21
20	$\overline{\text{BUSY}}$, $\overline{\text{BPRN}}$, $\overline{\text{CBRQ}}$, $\overline{\text{CBQLCK}}$/$\overline{\text{ALWAYS}}$ Hold to $\overline{\text{BCLK}}$ (or to RESET)	1		ns		13, 15, 21
21	$\overline{\text{BCLK}}$ to $\overline{\text{BREQ}}$ Delay		30	ns	Note 1	13, 14
22	$\overline{\text{BCLK}}$ to $\overline{\text{BPRO}}$ Delay		35	ns	Note 2	17
23	$\overline{\text{BPRN}}$ to $\overline{\text{BPRO}}$ Delay		25	ns	Note 2	17
24	$\overline{\text{BCLK}}$ to $\overline{\text{BUSY}}$ Active Delay		60	ns	Note 3	13
25	$\overline{\text{BCLK}}$ to $\overline{\text{BUSY}}$ Float Delay		35	ns	Note 4	13, 14
26	$\overline{\text{BCLK}}$ to $\overline{\text{CBRQ}}$ Active Delay		55	ns	Note 5	13
27	$\overline{\text{BCLK}}$ to $\overline{\text{CBRQ}}$ Float Delay		35	ns	Note 4	13, 20

A.C. CHARACTERISTICS (Continued)

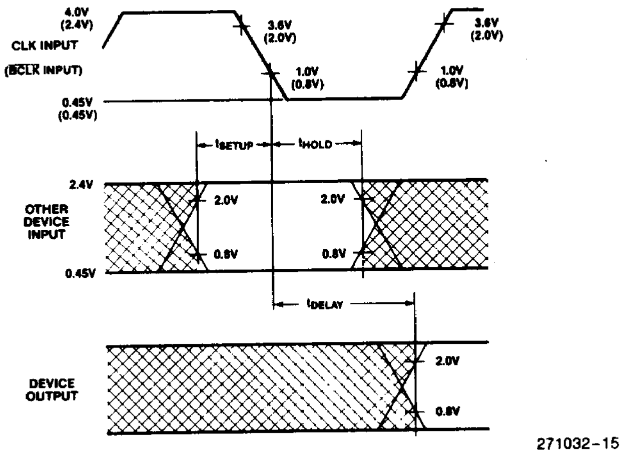
Sym	Parameter	8 MHz		Units	Comments	Shown in Figure
		Min	Max			
28	$\overline{BCLK}$ to $\overline{AEN}$ Active Delay		25	ns	Note 6	13
29	CLK to $\overline{AEN}$ Inactive Delay	3	25	ns	Note 6	13, 14
30	CLK to $\overline{LLOCK}$ Delay		20	ns	Note 7	18
31	RESET to $\overline{LLOCK}$ Delay		35	ns	Note 7	19
32	CLK to $\overline{BCLK}$ Setup Time	38		ns	Note 8	13, 16, 20

NOTES:
1. $\overline{BREQ}$ load: $C_L = 60$ pF.
2. $\overline{BPRO}$ load: $C_L = 60$ pF.
3. $\overline{BUSY}$ load: $C_L = 300$ pF.
4. Float condition occurs when output current is less that I_{LO} in magnitude.
5. $\overline{CBRQ}$ load: $C_L = 300$ pF.
6. $\overline{AEN}$ load: $C_L = 150$ pF.
7. $\overline{LLOCK}$ load: $C_L = 60$ pF.
8. In actual use, CLK and $\overline{BCLK}$ are usually asynchronous to each other. However, for component testing purposes, this specifictaion is required to assure signal recognition at specific CLK and $\overline{BCLK}$ edges.
9. $\overline{INIT}$ is asynchronous to CLK and to $\overline{BCLK}$. However for component testing purposes, this specification is required to assure signal recognition at specific CLK and $\overline{BCLK}$ edges.

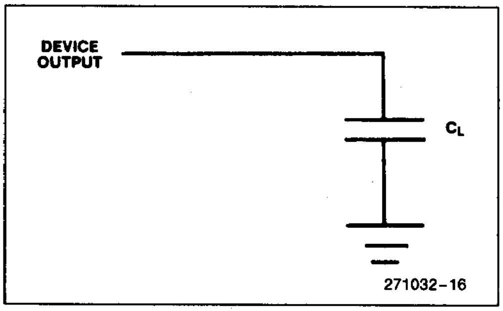
21



Note 10: AC Drive and Measurement Points—CLK Input ($\overline{BCLK}$ Input)



Note 11: AC Setup, Hold and Delay Time Measurement—General



Note 12: AC Test Loading on Outputs

Waveforms

The waveforms (Figures 13–21) show the timing relationship of the inputs and the outputs and do not show all possible transitions of all signals in all modes. Instead, all signal timing relationships are shown via the general cases. Special cases are shown when needed.

To find the timing specification for a signal transition in a particular mode, first look for a special case in the waveforms. If no special case applies, then use a timing specification for the same or related function in another mode.

The M82289 Bus Arbiter serves as an interface between the M80286 subsystem which operates synchronous to the CLK signal and MULTIBUS which operates synchronous to $\overline{\text{BCLK}}$ signal. CLK and $\overline{\text{BCLK}}$ generally operate asynchronously to each other and at different frequencies. Thus, the exact clock period in which an input synchronous to one clock will cause a response synchronous to the other clock depends on the relative phase and frequency of CLK and $\overline{\text{BCLK}}$ at the time the input is sensed.

One strict relation between CLK and $\overline{\text{BCLK}}$ must be maintained for proper MULTIBUS arbitration. If the CLK period is too long relative to $\overline{\text{BCLK}}$ period (t1 greater than t5 + 50 ns), another arbiter could gain control of the system bus before this arbiter has released $\overline{\text{AEN}}$ synchronous to its CLK. This situation arises since the release of $\overline{\text{AEN}}$ is synchronous to the next falling CLK edge after the processor cycle ends but the release of $\overline{\text{BREQ}}$ and $\overline{\text{BUSY}}$ is synchronous to the next falling $\overline{\text{BCLK}}$ edge after the processor cycle ends. In practice, any CLK frequency greater than 6.66 MHz (i.e., M80286 processor speeds greater than 3.33 MHz) will avoid conflict with a 10 MHz $\overline{\text{BCLK}}$. Therefore all M80286 speed selections are MULTIBUS compatible.

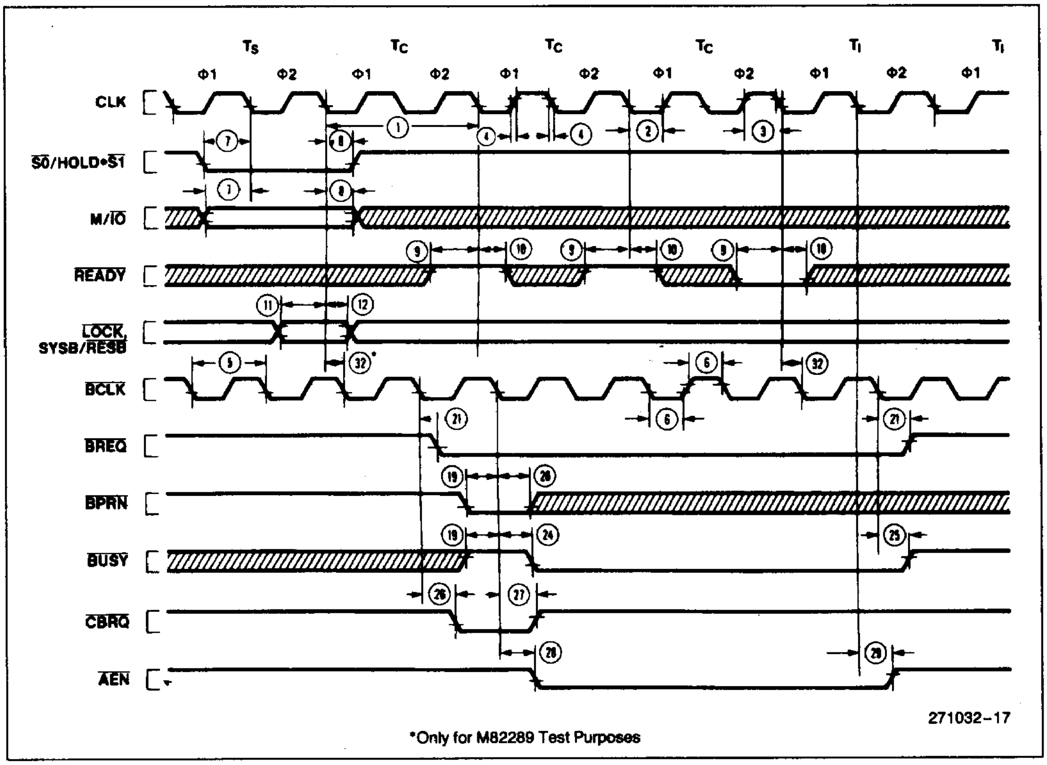


Figure 13. MULTIBUS® Acquisition and Always-Release Operation

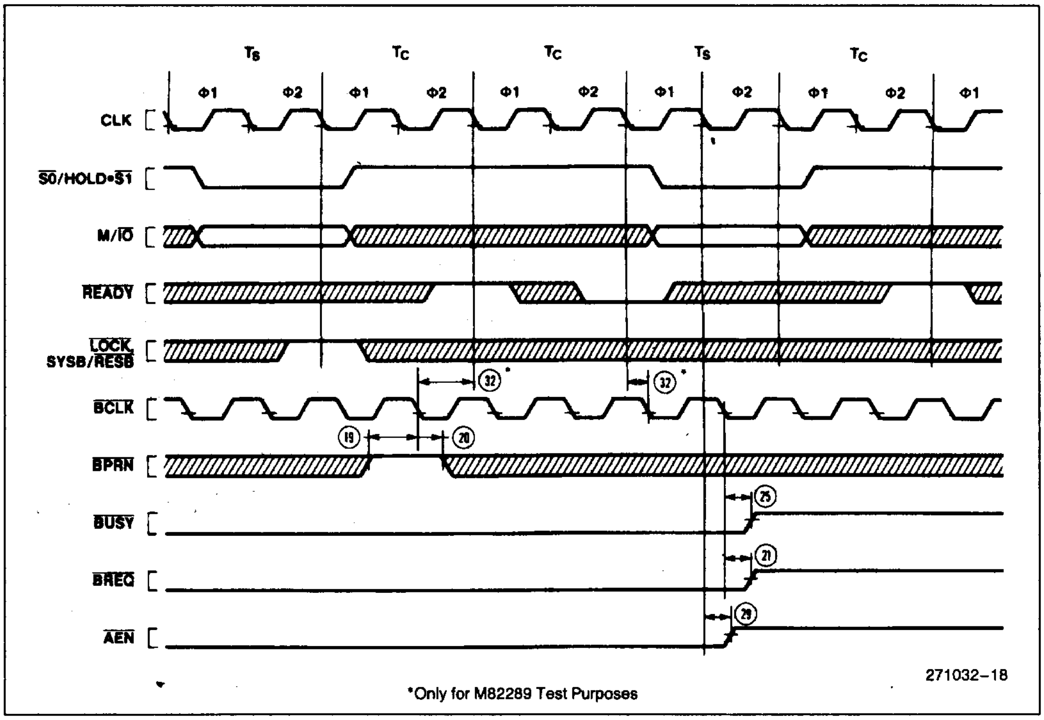


Figure 14. MULTIBUS® Release due to $\overline{\text{BPRN}}$ Inactive

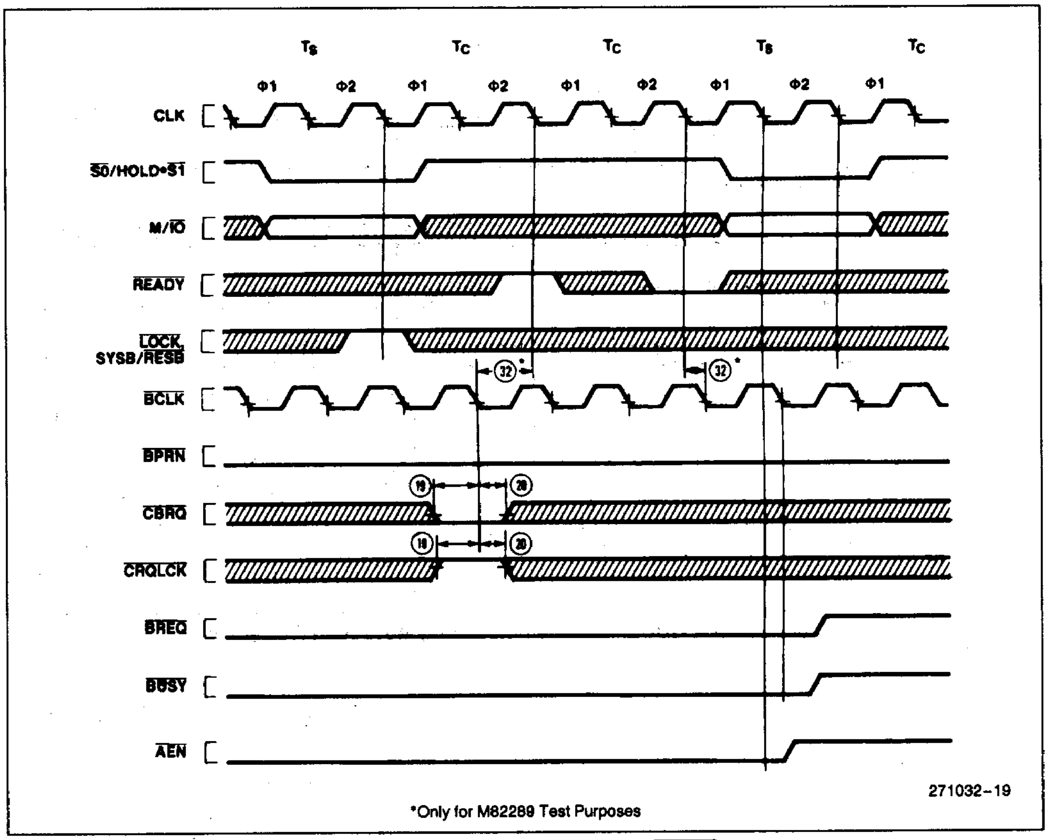


Figure 15. MULTIBUS® Release due to $\overline{\text{CBRQ}}$ Active

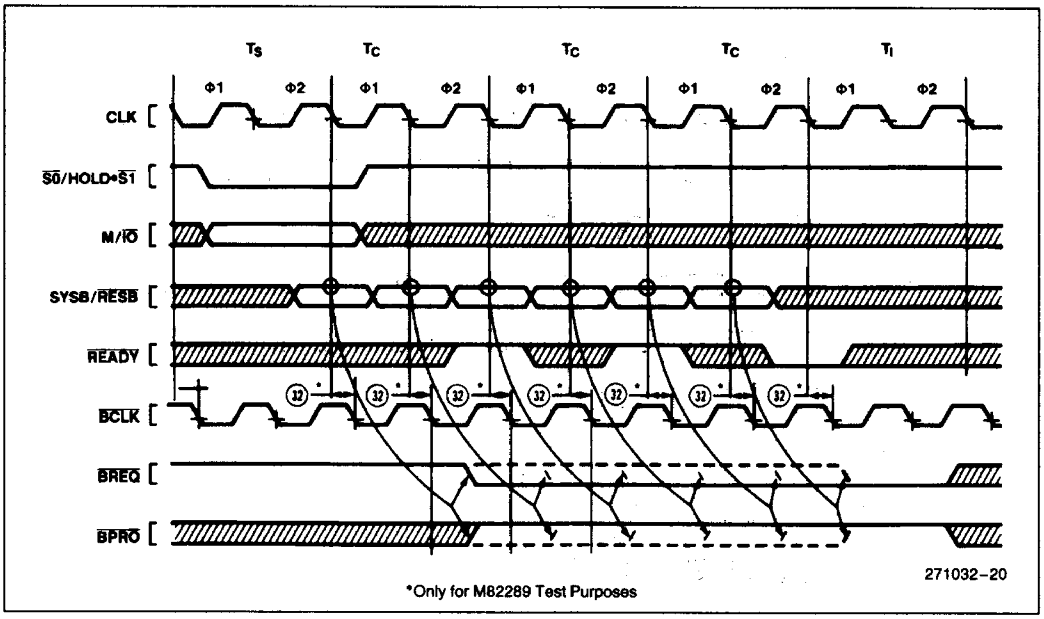


Figure 16. MULTIBUS® Acquisition During M80286 INTA Cycles

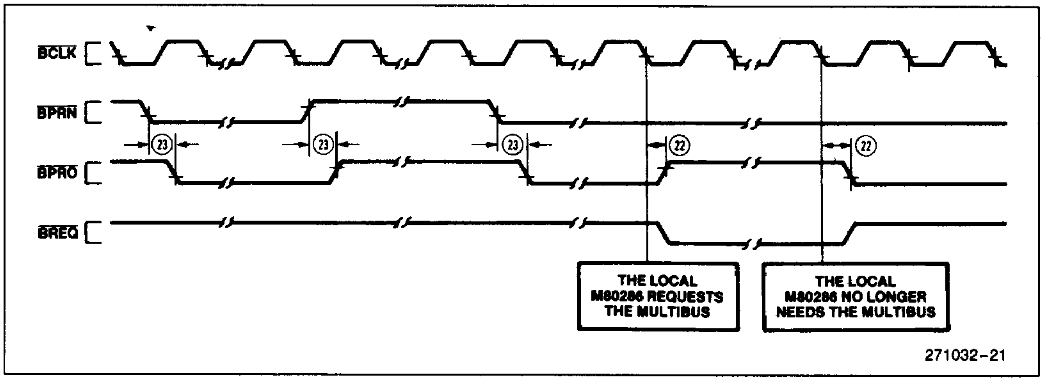

BCLK
BPRN
BPRO
BREQ
23
22
THE LOCAL M80286 REQUESTS THE MULTIBUS
THE LOCAL M80286 NO LONGER NEEDS THE MULTIBUS
271032-21

Figure 17. BPRN to BPRO Timing Relationship

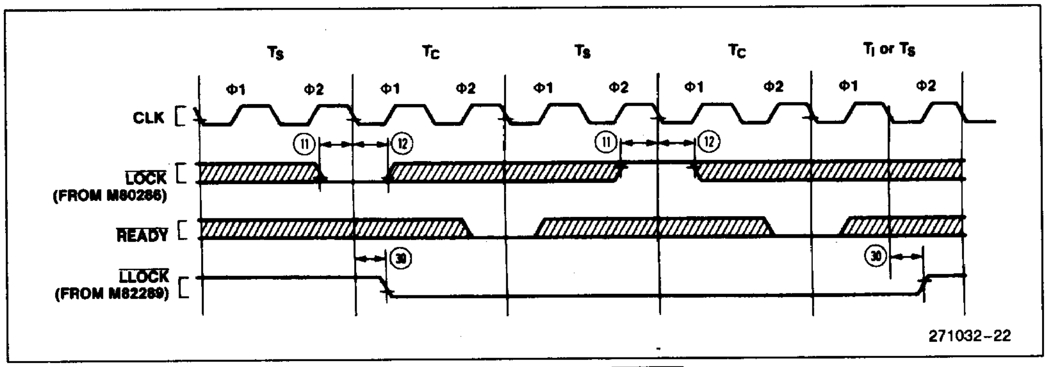
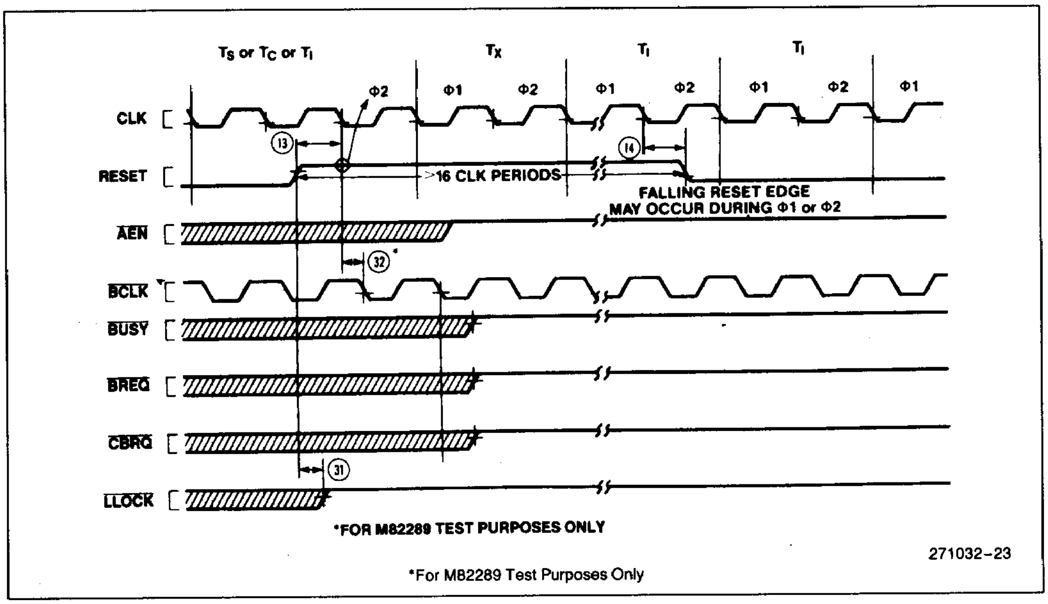


Figure 18. M80286 $\overline{\text{LOCK}}$ and M82289 $\overline{\text{LLOCK}}$ Relationship



*For M82289 Test Purposes Only

Figure 19. RESET Active Pulse

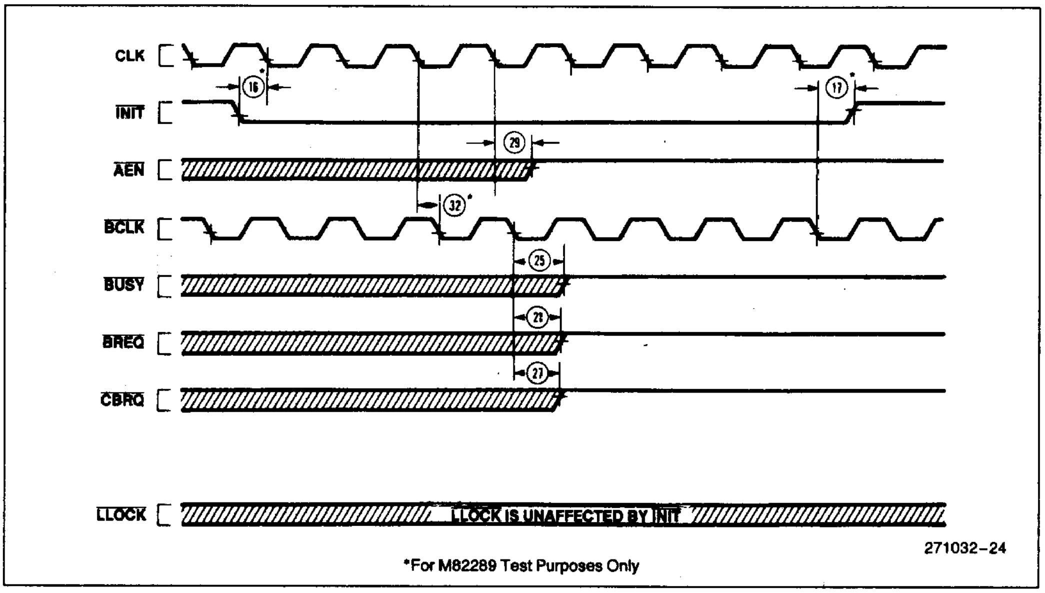


Figure 20. $\overline{\text{INIT}}$ Active Pulse

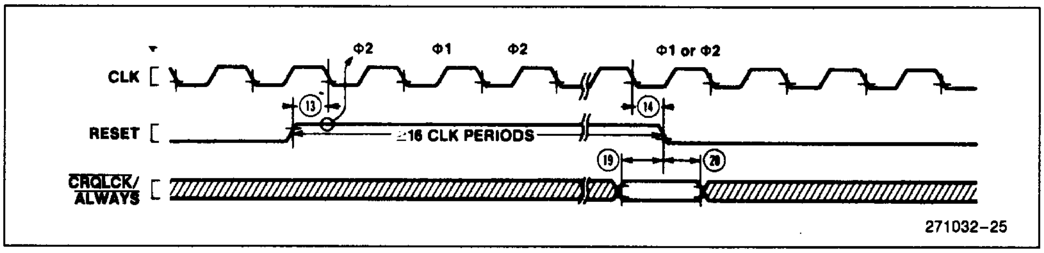


Figure 21. Programming the Always-Release/Common-Bus-Request-Release Option