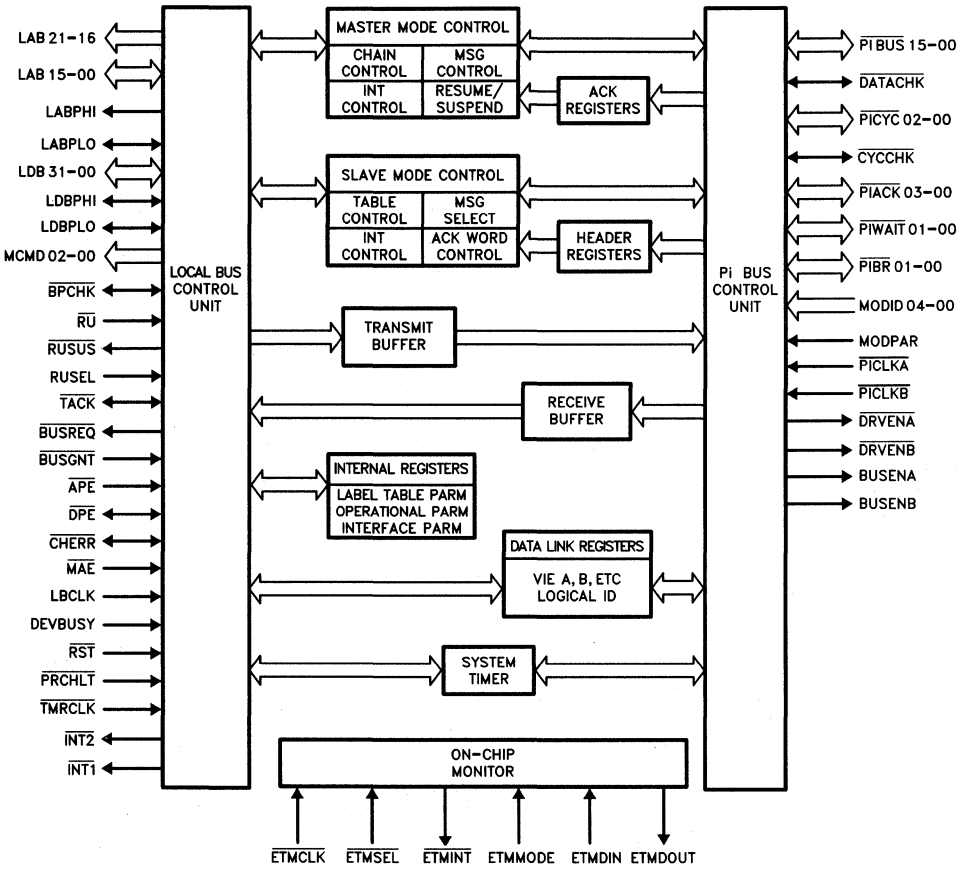
PRELIMINARY

M82916
Pi BUS INTERFACE UNIT

Military

- **32-Bit Host Interface**
- **Intelligent Host Interface Supports:**
  - **— Processor Systems**
  - **— I/O Processors**
  - **— Bulk Memory**
  - **— Discrete I/O Subsystems**
- **Extensive Interrupt Handling**
- **Military Temperature Range: 55°C to +125°C (T_C)**
- **16-Bit Pi Bus Implementation**
- **Master and Slave, or Slave Only Operation**
- **Error Detecting Operation**
- **Supports Dual Redundant Pi Bus Operation**
- **164 Pin Ceramic Quad Flat Pack (CQFP) Package**
- **High Performance CHMOS IV Process**

This data sheet is supplemented by a *M82916 User's Manual,* Intel literature number 271160-001. The *M82916 User's Manual* provides detailed information regarding hardware and software board design information.



271139-1

Figure 1. Block Diagram

Order Number: 271139-002

GENERAL DESCRIPTION

The M82916 Pi Bus Interface Unit is a highly integrated CMOS IV VLSI device that maximizes the performance of a Pi Bus based multi-processor system. The Pi Bus Interface Unit (PBIU) is designed to implement the interface between the Pi Bus and an electronics module (see Figure 2). It handles all bus protocol and DMA functions necessary to off-load Pi Bus communication from the host CPU. This allows maximum bus performance and subsequently increases system throughput.

The host system may be intelligent such as a processing module or an I/O module containing an I/O processor, or it may be a bulk memory or discrete I/O subsystem. The M82916 interfaces to the Pi Bus through a set of Pi Bus transceivers.

The M82916 implements a 16-bit (Type 16), Error Detecting (Class ED) Pi Bus Protocol, and can function as a Master and Slave (MS) or as a Slave Only (SO). These features are defined by the Society of Automotive Engineers (SAE), and the Joint Integrated Avionics Working Group (JIAWG) Pi Bus specifications. The M82916 will function on a Pi Bus with a clock rate up to 12.5 MHz and can be operated asynchronously to the host interface.

LOCAL BUS OPERATION

The M82916 communicates directly with Local Bus elements using either Direct Memory Access (DMA) or Register Update (RU) operations. DMA Read, Write and Read-Modify-Write sequences are used while processing Pi Bus messages, updating tables, and storing interrupt reports. Register Update sequences are used when the host processor is reading or writing the internal registers of the M82916. All Local Bus operations are pipelined with address before data and therefore require the use of external latches between the M82916 and the local memory.

The following sections describe bus activity on the Local Bus. Each form of bus cycles is described in detail. The interface signals associated with the bus cycle are shown. The relationship between the signals themselves and between the signals and the Local Bus clock are indicated. For exact timing relationships refer to the electrical specification section at the end of the data sheet. All signals output from the PBIU are generated on the rising edge of LBCLK. The Local Address Bus (LAB) and Memory Command Bus (MCMD) are driven active from the $\overline{\text{BUSGNT}}$ going low.

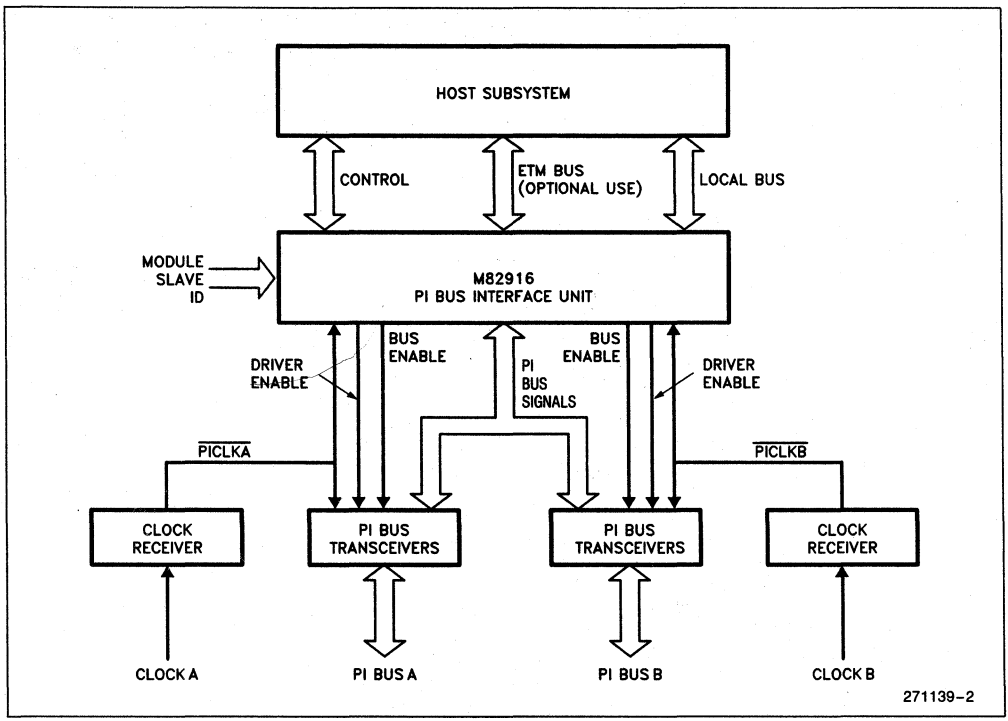


Figure 2. System Overview

Abbreviations found in the figures are defined as follows:

SOURCE — Identifies the source of the associated signal/bus activity in the illustrated LBCLK period.

MI — Memory Interface Logic

PBIU — Pi Bus Interface Unit

RD — Read Command

WR — Write Command

NOP — Null Command

A# — Memory Address

D# — Memory Data

RU — Register Update

Several signal representations are used in the bus cycle figures to follow. Three logic levels are used: Logical one (high), Logical zero (low), and float (middle). Busses are shown active with simultaneous high and low levels with a bus value indicated between the levels. Busses shown active with no value indicated are in a don't care input state, or an unknown output condition. Individual input signals may be drawn similar to a bus with both a high and low level. This indicates a don't care time period for that input.

Transitions on outputs are illustrated with some delay, reflective of a real device. Input signals are illustrated at their minimum requirements. When an input signal is shown making a transition to a don't care state, the transition is optional.

Bus Acquisition

The M82916 uses the $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSGNT}}$ signals to arbitrate for control of the Local Bus. $\overline{\text{BUSREQ}}$ is driven low by the M82916 when it requires control of the Lcoal Bus to perform a memory access. Local bus control is given to the M82916 when $\overline{\text{BUSGNT}}$ is driven low by either the host processor or some bus arbitration logic.

ACQUISITION WITH NO WAIT STATES

Figure 3 depicts a Bus Acquisition sequence for a two double-word (dword) memory read with no wait states. The bus cycle continues uninterrupted during a Bus Acquisition with no wait states. $\overline{\text{BUSGNT}}$ is driven low by the host's arbitration logic immediately following assertion of $\overline{\text{BUSREQ}}$ by the M82916. The address bus and Memory Command (MCMD) lines become active when $\overline{\text{BUSGNT}}$ goes low. The address is valid immediately, while the Memory Command lines are not valid until after the next rising edge of LBCLK.

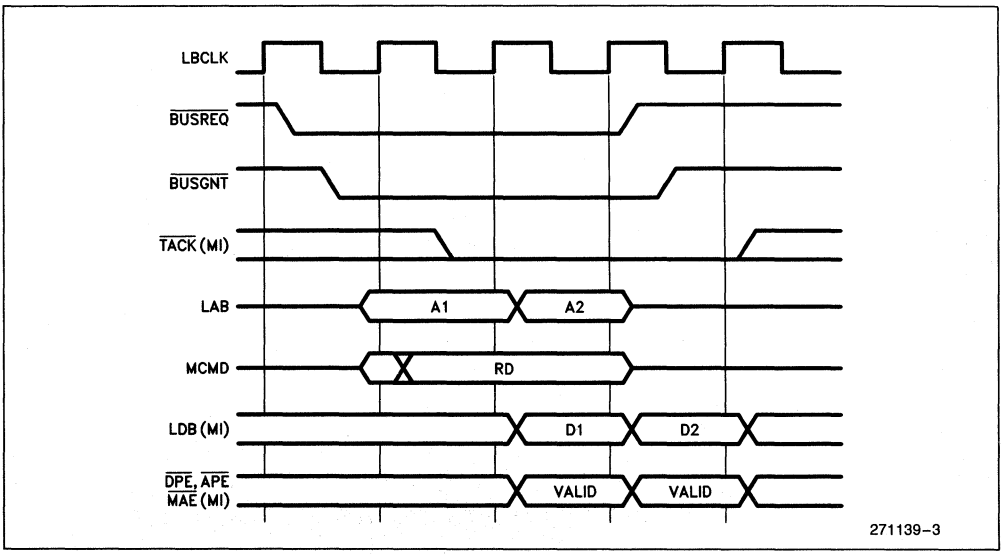


Figure 3. Local Bus Acquisition with No Waits (Two dword Read)

$\overline{BUSREQ}$ is released by the PBIU at the end of the last address cycle (see Figure 3). This is the same time the address and MCMD busses are placed in float. The M82916 uses a pipeline bus technique to provide the memory address one cycle before the data is required or provided. The bus cycle is completed when the last data word is acknowledged by $\overline{TACK}$ being low.

ACQUISITION WITH WAIT STATES

Figure 4 illustrates a Local Bus acquisition by the M82916 with wait states inserted. Two things can delay the M82916 from gaining control of the Local Bus. One is a delay of the $\overline{BUSGNT}$ signal to the PBIU. In Figure 4, $\overline{BUSGNT}$ is asserted one clock cycle after the request. The address bus and MCMD lines do not become active until $\overline{BUSGNT}$ is received. Additional wait states can be inserted by the arbiter or host processor by delaying $\overline{BUSGNT}$ even longer.

The second method wait states are added is by holding the $\overline{TACK}$ signal inactive (high). The LAB and MCMD busses are still driven active after grant is received. LAB and MCMD do not change until the clock edge after $\overline{TACK}$ is low. The second cycle is the wait state. The wait state occurs because the Transfer Acknowledge signal is detected inactive at the end of the first cycle. When $\overline{BUSGNT}$ goes low the LAB and MCMD busses become active. The LDB does not become active until after the $\overline{TACK}$ signal is sampled low.

ACQUISITION WITH RE-ARBITRATION

A third situation can occur during bus acquisition or during the middle of a bus cycle. Local bus control can be taken away from the M82916. Figure 5 illustrates this situation. The arbiter or host processor granted bus control to the PBIU and then reversed its decision by de-asserting $\overline{BUSGNT}$ on the next clock. The LAB and MCMD busses are floated when $\overline{BUSGNT}$ becomes inactive (high). The bus cycle begins again when $\overline{BUSGNT}$ becomes active.

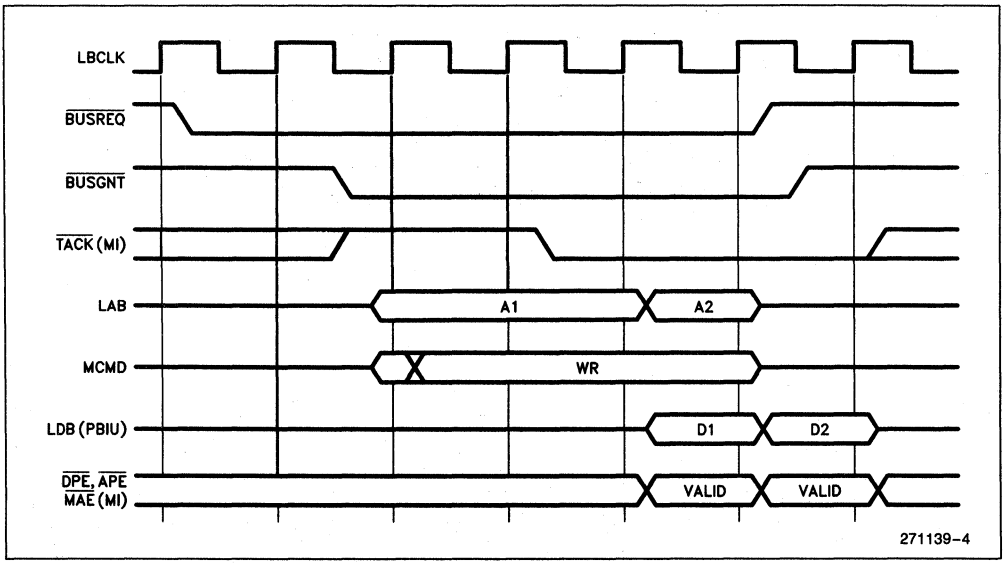


Figure 4. Local Bus Acquisition with Waits (Two dword Write)

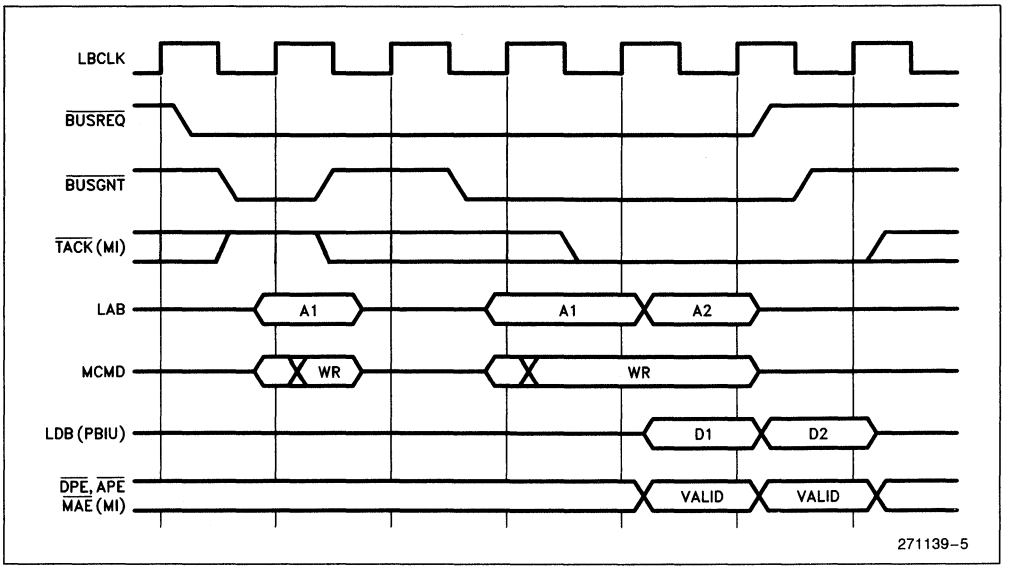


Figure 5. Local Bus Acquisition with Re-Arbitration (Two dword Write)

DMA Operations

The M82916 has a Direct Memory Access interface consisting of a 32-bit Local Data Bus (LDB) and a 22-bit Local Address Bus (LAB). Several control lines (BUSGNT, BUSREQ, BPCHK, MCMD, TACK, APE, DPE, and MAE) are also utilized during DMA operations. The Memory Command Bus determines the type of DMA operation that will be performed. Figure 6 identifies each possible operation.

MCMD Bus 02	MCMD Bus 01	MCMD Bus 00	Bus Cycle Type
0	0	0	dword DMA Write
0	0	1	Low word DMA Write
1	0	0	dword DMA Read
1	1	1	NOP

Figure 6. Memory Command Bus Decode

Error lines are utilized to provide fault tolerance in the host system. If the system detects an error during DMA operations, the appropriate error signal must be active during the corresponding data cycle. The M82916 samples the error lines during data cycles only. Active error signals indicate address and data errors during the cycle in which they are sampled.

READ OPERATIONS

The DMA interface performs 32-bit wide single, double, triple, and quadruple double-word read operations (also called two, four, six and eight single word read operations). A double-word (dword) read (32 bits) will always be performed, even if the M82916 requires only a single word (16 bits) from memory. In the case of a single word read, the M82916 will correctly choose the required 16-bit word from the 32 bits of data.

The various types of DMA operations performed by the M82916's DMA interface are illustrated in the following figures. Unless otherwise specified, zero wait state operation is shown.

Single Dword Read

A single dword read operation is illustrated in Figure 7. The address cycle is one clock cycle after BUSGNT is received. The clock cycle following the address cycle is the data cycle. Data will be latched on the rising edge of the LBCLK (assuming TACK is low).

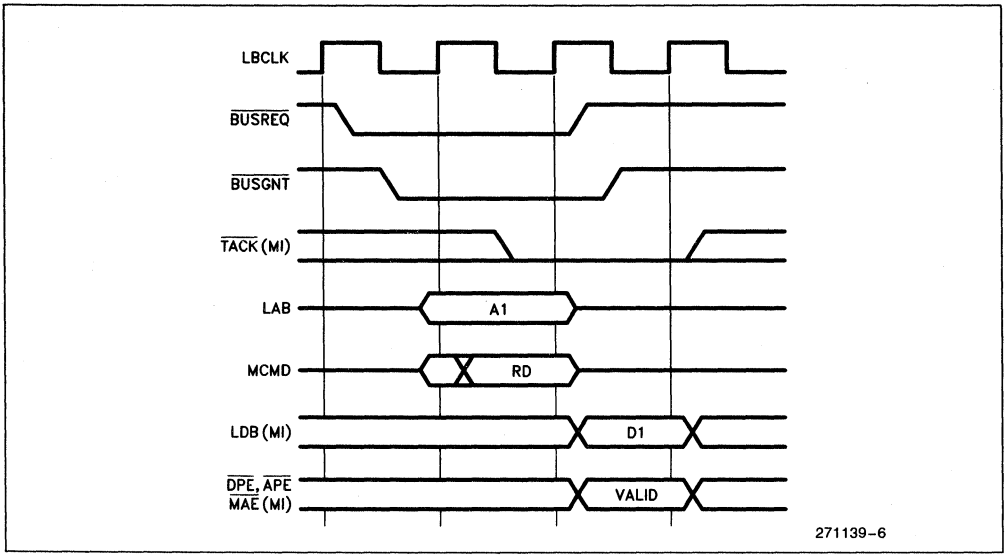


Figure 7. One dword DMA Read (No Wait States)

Two Dword Read

A two dword DMA read operation is depicted in Figure 8. As with a single dword access, the clock after BUSGNT becomes active is the first address cycle. The cycle following the first address cycle is the first data and the second address cycle. This allows the first address to be latched and remain active to the memory while the second address is driven out by the PBIU. The second data cycle is the last sequence of this operation.

Three Dword Read

Figure 9 illustrates the bus activity for a zero wait state three dword memory read. The BUSREQ signal is de-asserted by the M82916 at the same time the address and command busses become inactive. However, the bus cycle is not complete until after the last data cycle.

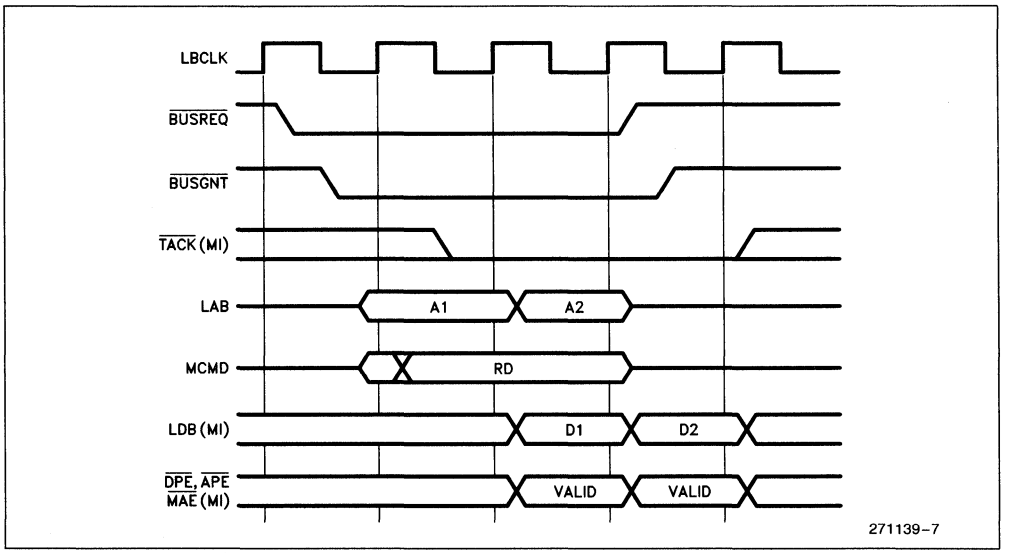


Figure 8. Two dword DMA Read Operation (with No Wait States)

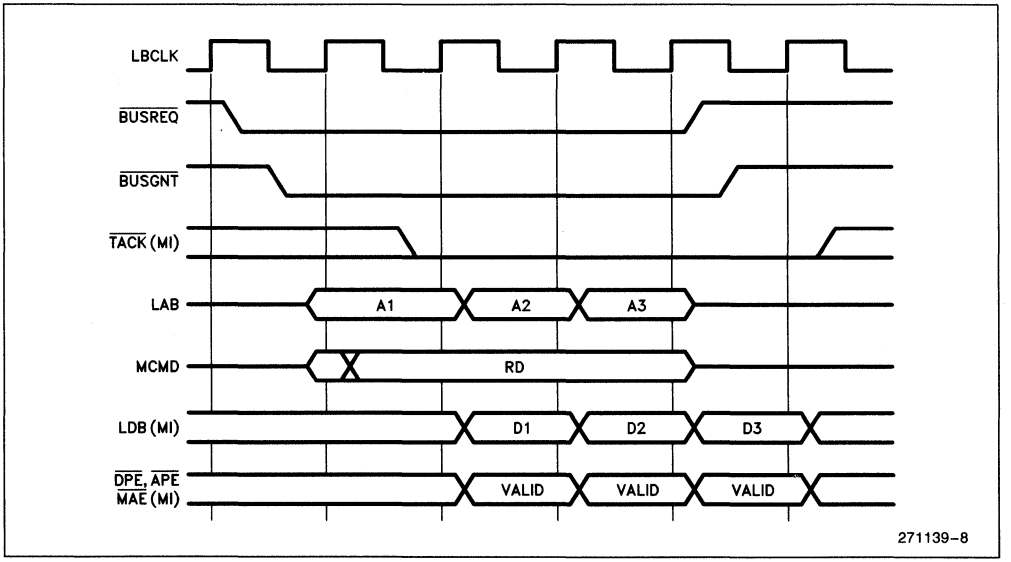


Figure 9. Three dword DMA Read Operation (with No Wait States)

Four Dword Read

The maximum DMA transfer by the M82916 is four dwords. This bus cycle is shown in Figure 10. Four word DMA operations are the same as two and three dword transfers with the exception of the fourth bus cycle.

Read Operations with Wait States

Wait states can be inserted during any cycle of a read operation to accommodate slower memory and peripheral devices. Figure 11 illustrates how $\overline{TACK}$ is used to insert wait states throughout the bus operation. The address and memory command of the current bus cycle remain valid as long as $\overline{TACK}$ is held high by the memory interface. The corresponding data to the address cycle in wait state need not be valid until the clock edge after $\overline{TACK}$ becomes active (low) again. More than one wait state can be inserted by holding $\overline{TACK}$ high longer.

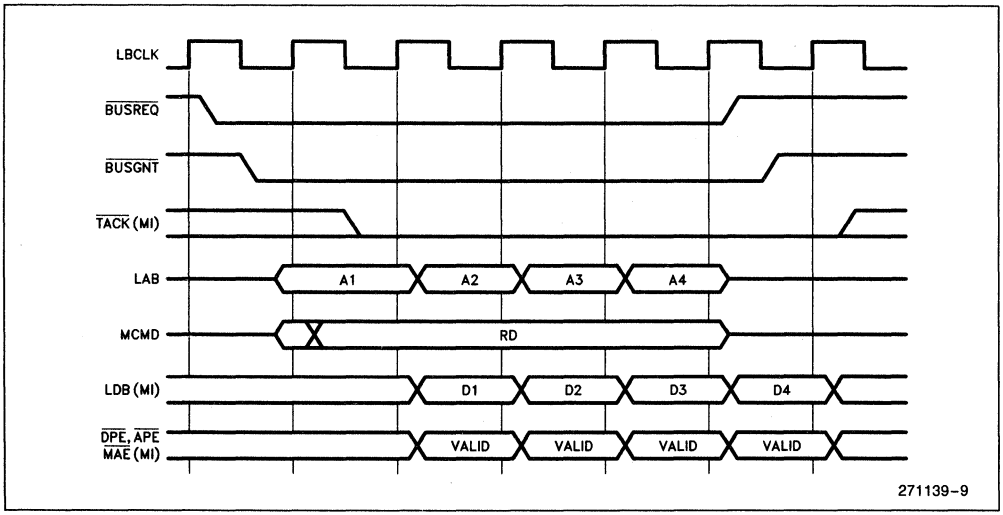


Figure 10. Four dword DMA Read Operation (with No Wait States)

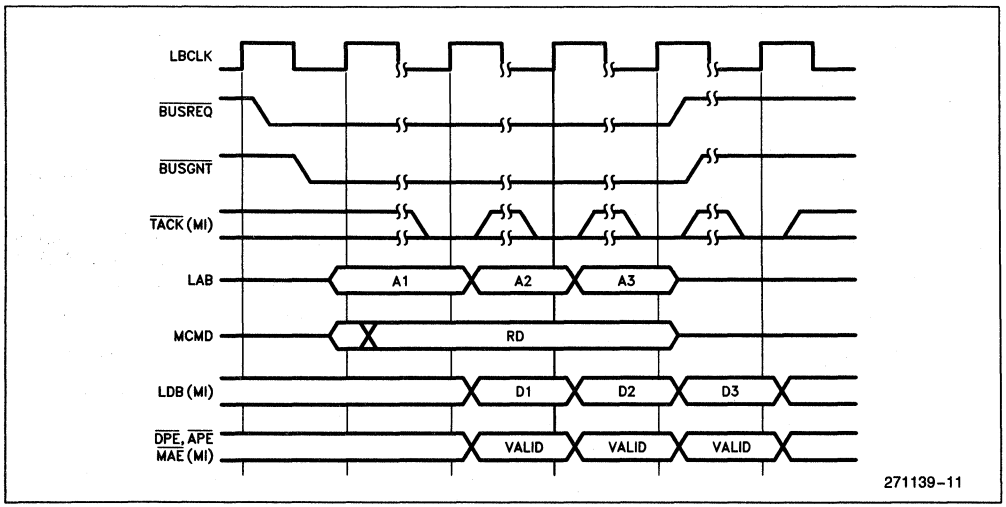


Figure 11. Three dword DMA Read Operation (with Optional Wait States)

WRITE OPERATIONS

Write sequences can be one, two, three, four, five, six, seven, or eight single words in length. A write operation with an odd number of words will begin with a single word transfer, but all remaining words will be written in dword transfers. This type of transfer will begin with a single word write of the low word (LDB 15–00). A write operation may also end with a single word (odd word count). In this situation the last valid word will be written to the high word (LDB 32–16). Indeterminable data will be written to the low word.

Write operations for the M82916 follow the same sequence as the read operations described in the Read Operations section. The only differences between read and write operations are that the data bus is driven by the M82916 and the MCMD lines reflect a write operation. Wait states can be inserted into write operations using the same methods previously shown for read operations (see Figure 11). The data, address and memory command for write operations remain valid during the cycles in which wait states are inserted.

Block Protect Check Operations

The Block Protect Check signal is used in systems with a protected memory structure. In this type of system, blocks of memory are defined as valid or invalid for devices such as the M82916. Each block is defined on a 4K address boundary. One protect bit for each block is maintained by the memory interface to define which block is available to the M82916. The memory interface uses protect bits to verify the address of the M82916 write operation. If the address is not valid, the memory interface should drive the $\overline{MAE}$ signal active (low).

A Block Protect Check sequence takes place on the first write after reset, on a write to an address in a new block of memory, if the memory interface did an update of its protect bits or when a read-modify-write operation is performed. The memory interface would indicate an update by driving $\overline{BPCHK}$ low for one clock cycle. The memory interface is not required to implement a memory block protection scheme. However, a Block Protect Check sequence will be executed by the M82916 during the situations identified above.

Figure 12 illustrates the Block Protect Check operation. The $\overline{BPCHK}$ signal is driven low when the MCMD is in the NOP state. The valid state of MCMD bus for a Block Protect Check cycle is the NOP state. The data bus is floated by the M82916 on the following cycle. The memory interface can use the data bus during this cycle to retrieve protect bits. At the end of a Block Protect cycle, the M82916 drives the $\overline{BPCHK}$ line high for one clock. Wait states can be added to lengthen the bus cycle at any time during a Block Protect Check sequence.

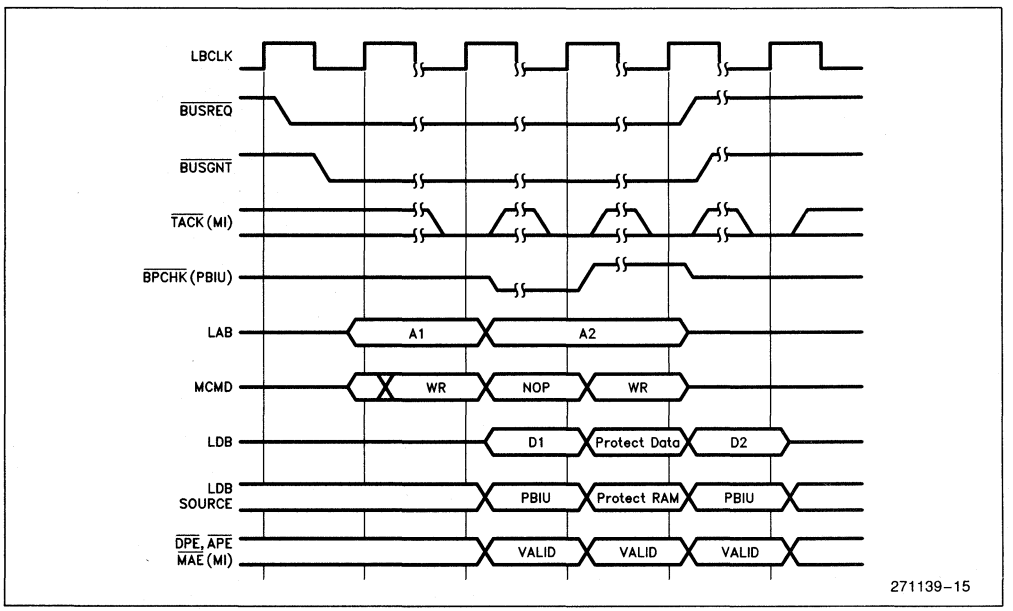


Figure 12. Block Protect Check (Optional Wait States Shown)

Read-Modify-Write Operations

The M82916 can execute a read-modify-write (RMW) sequence to update a single dword in memory. During a read-modify-write with no wait states, a read sequence is performed followed two LBCLK cycles later by a write sequence. Figure 13 illustrates this operation. The clocks between the read and write are spent bringing the data in, altering the data, then preparing the data to be sent back out on the Local Bus. This time is also used to perform a Block Protect Check sequence without any delay to the bus operation.

Read-modify-write sequences can also be performed with wait states. Figure 14 illustrates how a "free" wait state (with no additional clocks) can be inserted directly after the read of a RMW sequence. The $\overline{TACK}$ cycle postpones the $\overline{BPCHK}$ cycle until the third clock. This operation takes the same number of clocks to execute as a RMW sequence with no wait states.

Wait states can be added that lengthen a read-modify-write cycle. Figure 15 shows a wait state added to the read cycle. Wait states can also be added during other clock cycles within the RMW cycle.

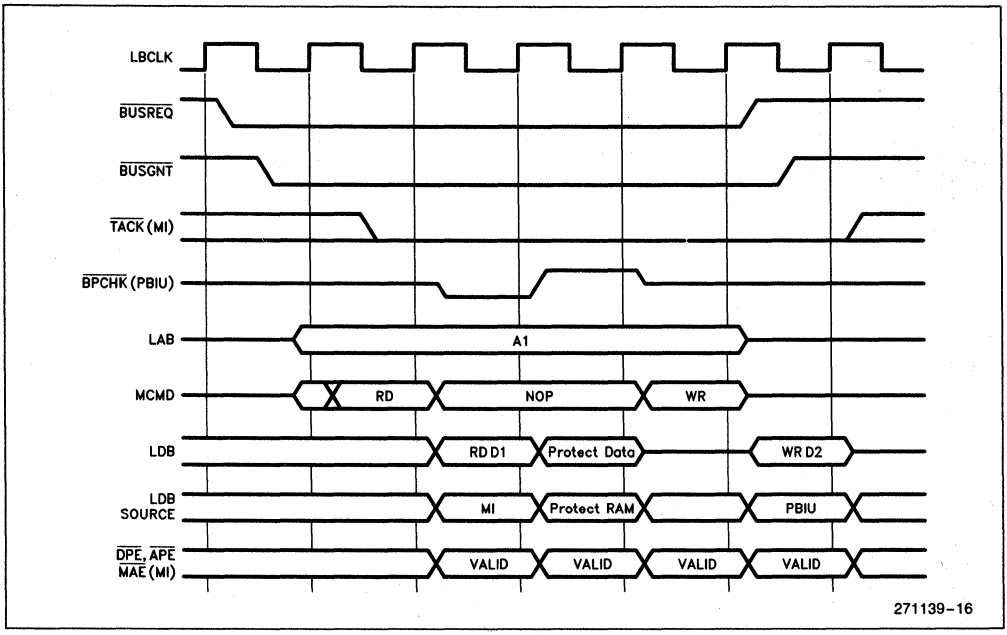


Figure 13. Read-Modify-Write (No Wait States)

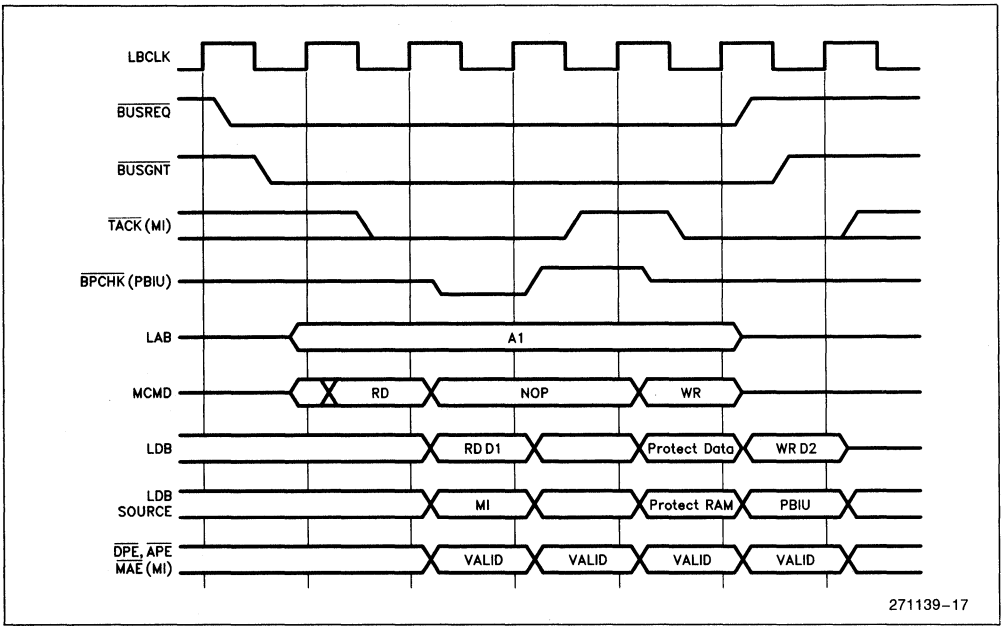


Figure 14. Read-Modify-Write (with "Free" Wait State)

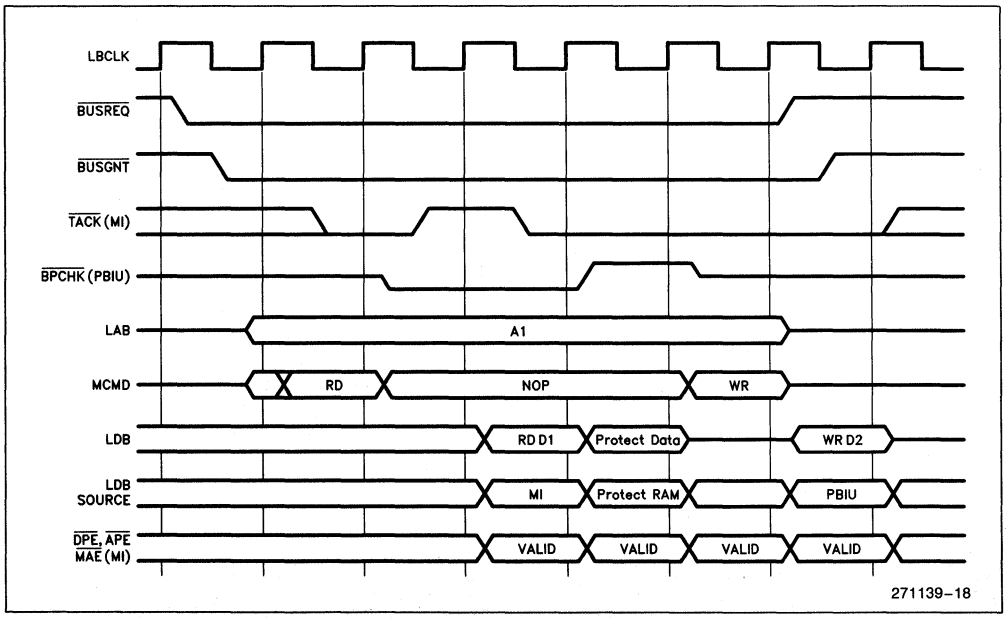


Figure 15. Read-Modify-Write (with Wait State for Read Data)

Register Update Operations

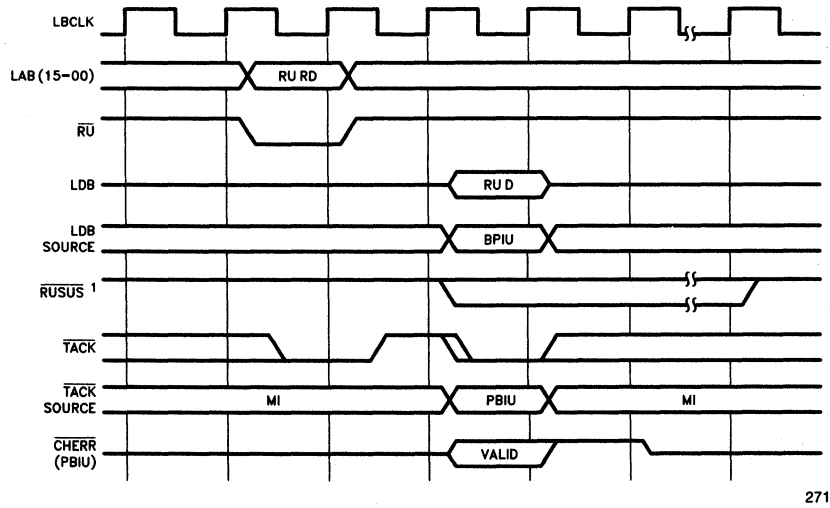
The M82916 uses internal registers to store status and control information pertaining to its operation. These internal registers can be written or read by a host processor using Register Update (RU) operations. M82916 internal registers are 16 and 32 bits wide and are divided into two groups: Local Bus registers, and Data Link Layer registers.

The Local Bus register group consists of registers that operate using the LBCLK. Access of these registers by the host processor is similar to a memory access. The M82916 Data Link Layer (DLL) registers are accessible to the host processor as well as to agents operating on the Pi Bus. DLL registers operate using the Pi Bus clock. RU operations by the host processor addressing these registers require additional time to synchronize the data to the different clock.

REGISTER READ OPERATION

Figure 16 illustrates a Register Update read operation. The host CPU or memory interface drives the $\overline{RU}$ input low, selecting the M82916 internal registers and initiating the RU sequence. The host CPU is also responsible for simultaneously driving $\overline{TACK}$ low and driving the register address on to the address bus. $\overline{TACK}$ and $\overline{RU}$ must be driven high after the second clock edge.

The M82916 will respond to the RU command on the rising edge of the LBCLK following de-assertion of $\overline{TACK}$ (inactive). During this clock cycle the M82916 will respond with data on the LDB and an active $\overline{TACK}$ (low). Channel Error ($\overline{CHERR}$) must be driven during this cycle if an error occurs. The M82916 releases all signals, except $\overline{CHERR}$, on the rising edge of the following clock. $\overline{CHERR}$ is driven high for one clock before being floated.



NOTE:
The assertion of $\overline{RUSUS}$ depends upon the addressed register. $\overline{RUSUS}$ will go active (low) for accesses of Data Link Layer registers, and remain inactive (high) for accesses to Local Bus registers.

Figure 16. RU Read Operation

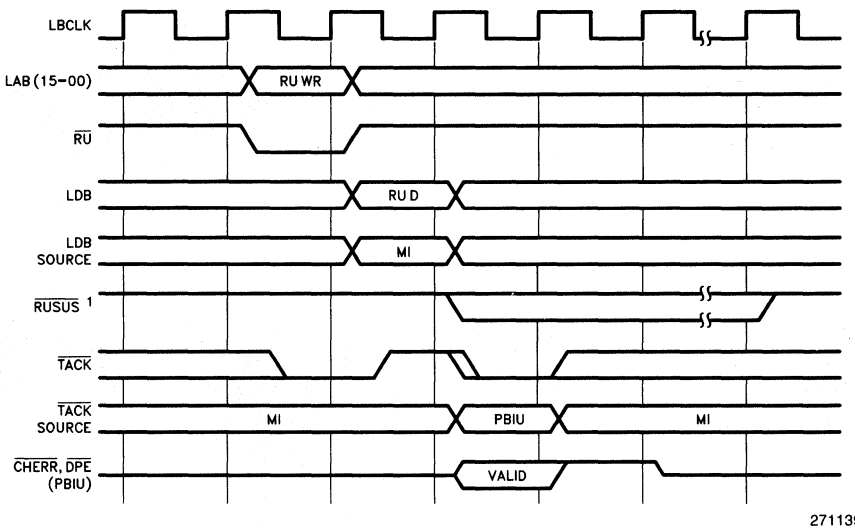
REGISTER WRITE OPERATION

An example of a Register Update write operation is shown in Figure 17. The host CPU initiates the operation by driving the $\overline{RU}$ input signal, $\overline{TACK}$, and the register address on to the address bus. It must also drive $\overline{TACK}$ and $\overline{RU}$ high and write data to the LDB after the second clock edge. Channel Error ($\overline{CHERR}$) and $\overline{DPE}$ will be driven on the cycle following the data cycle if an error occurs. The M82916 drives $\overline{TACK}$ low during this clock cycle. The M82916 releases all signals, except $\overline{CHERR}$ and $\overline{DPE}$, on the rising edge of the following clock. $\overline{CHERR}$ and $\overline{DPE}$ are driven high for one clock before being floated.

DATA LINK LAYER REGISTER READ OPERATION

Data Link Layer registers operate using the Pi Bus clock rather than the LBCLK. A register access addressing DLL registers requires a synchronization period for data to be passed from DLL registers to Non-DLL registers. The Register Update Suspend ($\overline{RUSUS}$) signal is used to indicate the synchronization period of register data between the two clocks. The $\overline{RUSUS}$ signal is only used when DLL registers are addressed.

Figures 18 shows an RU operation involving a Local Bus register. $\overline{RUSUS}$ remains inactive during accesses to this register group (read or write). The host CPU can follow a Local Bus register RU operation directly with another RU operation. No waiting period is necessary.



NOTE:
The assertion of $\overline{RUSUS}$ depends upon the addressed register. $\overline{RUSUS}$ will go active (low) for accesses of Data Link Layer registers, and remain inactive (high) for accesses to Local Bus registers.

Figure 17. RU Write Operation

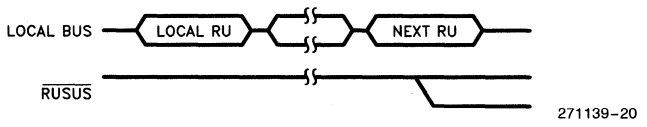


Figure 18. Register Update of Local Bus Register (Read or Write)

Figure 19 illustrates a DLL Register Update read operation. The M82916 drives the $\overline{\text{RUSUS}}$ signal during the RU read operation. The data output by the M82916 during this initial read cycle is meaningless. $\overline{\text{RUSUS}}$ was driven low by the M82916 to inform the host CPU that the register data requested is being synchronized to the LBCLK and is not yet available. After the $\overline{\text{RUSUS}}$ signal goes high the host CPU must perform a second RU operation to read the DLL register data that now resides in a Local Bus register. The DLL data register is designated for this task.

After the Register Update read of the holding register the next RU operation can take place. If an RU operation of a DLL register is followed by any RU operation other than the reading of the holding register, an error will be generated. If an RU operation is executed while the $\overline{\text{RUSUS}}$ line is low an error will be generated. The $\overline{\text{CHERR}}$ signal is used to indicate these errors. The LDB is available to the host to execute memory operations to other devices while $\overline{\text{RUSUS}}$ is low.

DATA LINK LAYER REGISTER WRITE OPERATION

An RU write to a DLL register is illustrated in Figure 20. For this operation the $\overline{\text{RUSUS}}$ line is driven low during the RU operation. The $\overline{\text{RUSUS}}$ signal remains active during the internal synchronization process. After the data has been synchronized and written into the DLL register, the $\overline{\text{RUSUS}}$ line is de-asserted. This lets the CPU know the operation has completed and the next RU operation can be executed. No RU operation can take place while $\overline{\text{RUSUS}}$ is low. The LDB is available to the host to execute memory operations to other devices while $\overline{\text{RUSUS}}$ is low.

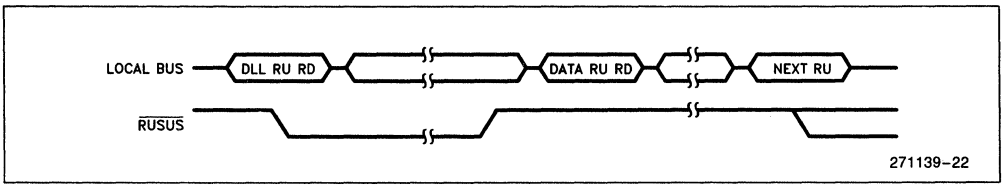


Figure 19. Register Update Read of a Data Link Layer Register

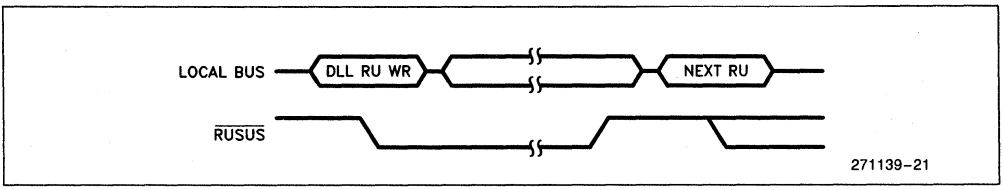


Figure 20. Register Update Write of a Data Link Layer Register

Pi BUS OPERATION

The Pi Bus protocol is implemented by the Pi Bus control unit of the M82916. The M82916 interfaces with the Pi Bus backplane through this functional block. This interface provides dual transceiver control signals for a redundant Pi Bus implementation. All Pi Bus protocol message handling, error handling, and error generation is performed by the M82916. The intelligent control unit performs all the protocol tasks autonomous from the host CPU.

Pi Bus Master Mode Description

The M82916 operates in the Master mode when it acquires Pi Bus mastership and begins to initiate messages. The Master mode is typically used when the M82916 is the Pi Bus interface on an intelligent module such as an M80960 processor module. The M82916 operates out of a command chain located in local memory.

Each instruction in the chain defines a single message for the Pi Bus. The master retrieves the chain instruction, interprets the contents, and executes the specified operation within the instruction. The contents of the chain instruction specify features such as data buffer address, message transfer word count, store an interrupt report, inhibit message suspension, NOP instruction, jump instruction, and the header words of the Pi Bus message.

The PBIU uses several Local Bus memory resources when operating as a Pi Bus Master. These resources are:

- Master Interrupt Queue
- Normal Command Cell
- Priority Command Cell
- Message Chains
- Message Data Buffers

The Master mode local memory resources are shown in Figure 21. The interrupt queue and command cells are located at the base address specified in the Base Address register. The command cells contain the address of their respective message chains, with each chain instruction (where required) containing the address of the data buffer to be used for the message.

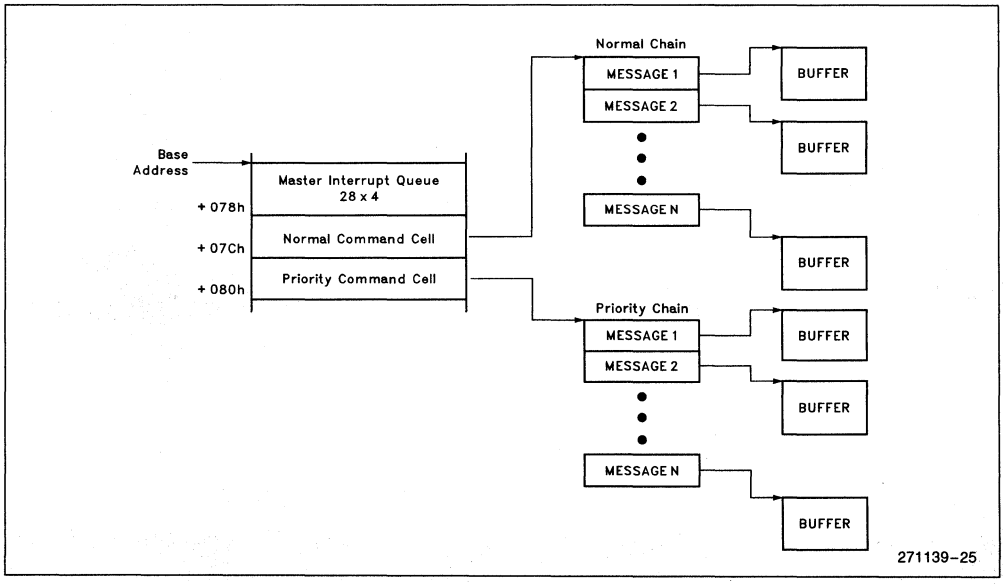


Figure 21. Master Mode Local Memory Resources

Message chain instructions are used by the M82916 to initiate messages over the Pi Bus. The chain instructions contain the header words to be used for the message. The types of messages that the M82916 can execute are:

- Parameter Write, Single Slave
- Parameter Write, Multiple Slaves
- Block Message Write, Short Header, Single Slave
- Block Message Read, Short Header, Single Slave
- Block Message Write, Short Header, Multiple Slaves
- Tenure Pass
- Bus Interface Write, Single Slave
- Bus Interface Read, Single Slave
- Bus Interface Write, Multiple Slaves

The host CPU executes a Register Update write operation to the M82916 Initiate Chain bit of the chain register. This starts the execution of a chain. After receiving the Initiate Chain bit, the PBIU uses the first two words of the Command Cell to form the address of the chain. The M82916 fetches chain instructions and executes the messages sequentially, continuing down the message chain in the same manner a processor executes code. The specified data buffers are accessed and interrupt reports are added to the master mode interrupt queue as required. This operation is illustrated in Figure 22.

The master stops Pi Bus operation after the last message in the chain has been executed. A Level 2 interrupt is generated at the end of the chain.

Two chain instructions are executed by the M82916; message chain instructions and jump chain instructions. A jump chain instruction addresses a new message chain, or a new chain instruction within the same chain. Jump chain instructions allow chains to be noncontiguous in memory.

To initiate a Pi Bus message, the M82916 must obtain Pi Bus mastership. When the first message instruction is fetched the PBIU acquires Pi Bus mastership through a Vie sequence. Tenure starts when the mastership is acquired. The message specified by the chain instruction is completed before the next chain instruction is fetched.

Pi Bus Slave Mode Description

The PBIU can act as a Slave independently of its Master mode operation. The PBIU, operating as a Slave, responds to the Pi Bus message types listed above in the master mode section. The M82916 requires local memory resources for Slave mode operation. These memory requirements are shown in Figure 23. The memory blocks are used to handle different message types and are discussed in the respective message type sections that follow.

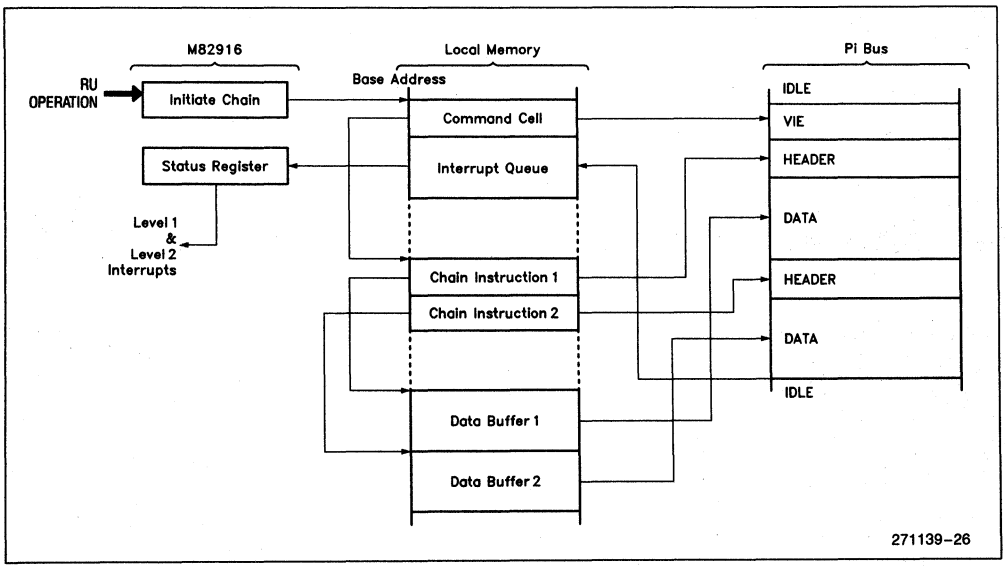


Figure 22. General Master Mode Operation

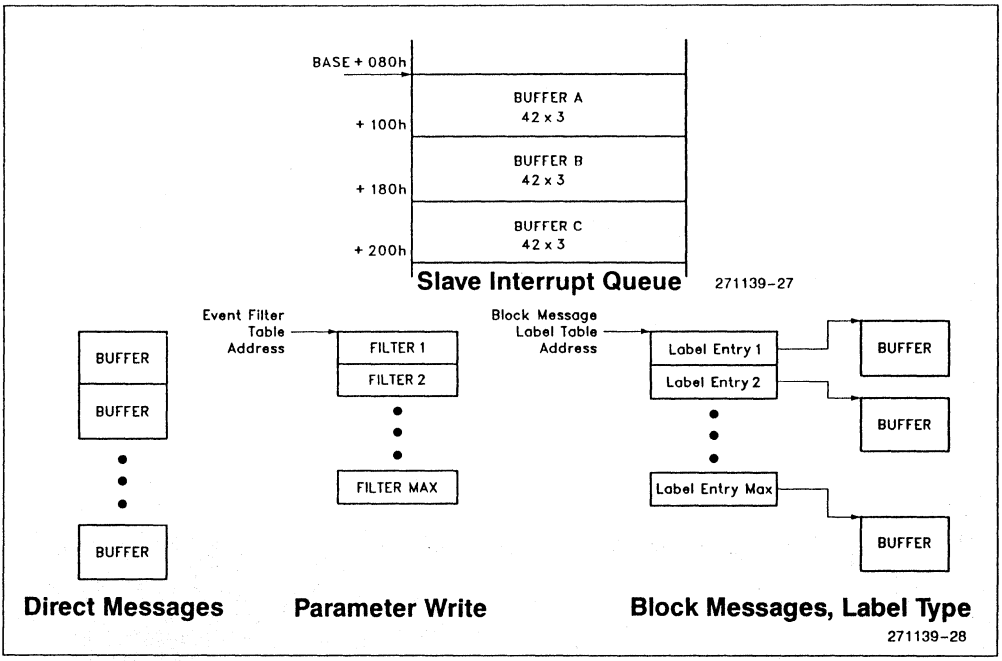


Figure 23. Slave Mode Memory Map

The Slave ID field of the message must match the PBIU module ID, must address an enabled logical slave ID or must be the broadcast address. The Slave identifies the message as pertinent when one of the addressing requirements has been met. The PBIU decodes the Message Type (MT) field of the header to determine if the message type is acceptable. The Access Type (AT) field is then decoded and compared with both the enabled and implemented access types to determine if the PBIU will respond to the message.

The Slave provides information regarding message operation to the host processor through interrupt reports. A three-buffer interrupt report queue is used to store these reports. Successful messages, unsuccessful messages, Pi Bus errors, and Local Bus errors are examples of the conditions for which an interrupt report would be generated.

The M82916 becomes Busy to Pi Bus messages when the slave interrupt queue becomes full or a Local Bus error is encountered during an access of local memory. Host action is required to clear the interrupt and hence remove the Busy condition. The PBIU also becomes busy following an Abort while it is storing the interrupt report (if required). Once the interrupt report is stored, the M82916 becomes not Busy.

Pi BUS MESSAGES

Message types 0, 2 and 6 are reserved codes and do not have a defined Pi Bus sequence. The slave will consider them to be command errors in the Interrupt Report and Multiple Slave Acknowledge Word. The Slave will not respond to a reserved message.

BLOCK MESSAGES

Pi Bus modules use block messages as the work horse of the bus. Block messages are used to transfer blocks of data between two or more modules. Transfers up to 64K words long can be performed with a block message. Block messages are both read and write data transfers.

The M82916 implements two types of block messages; direct and label type. These two block messages are identified by an access type in the message header word. Both are used to transfer blocks of data across the Pi Bus, but implement a different storage method within the Slave.

In direct type block messages the Pi Bus master specifies the data buffer address in the header words. This is the physical address of a data buffer in the Slave module. The sequence of a direct block message read transfer is shown in Figure 24. The header words are received by the slave and acknowledged by the addressed slave. The slave then uses the address in the header words to retrieve or store data transferred in the message. Any interrupt report is stored after the data acknowledge cycle. The memory resource needed for direct block messages is simply the data buffer.

The second type of block message transfer is a label type. The sequence of this type of message is illustrated in Figure 25. This message type is much more flexible than the direct message. The header word contains a label instead of a physical address. The label is used to select a block from the Label Table in Local memory. The Slave uses the Label Table Entry to define the handling of the block message.

The Lable Table Entry contains four fields; flags, count, address A, and address B. The flags define options available for the label. The count field specifies the number of words to be transferred in the block message. The two address fields define the location of two data buffers. These buffers can be used in a double buffering scheme through the control bits in the flag field.

The label type block message continues just as the direct type. The data is retrieved or stored in the data buffer available. Any interrupt report is stored in the slave interrupt queue after the data acknowledge cycle. Some updates to the label table entry may take place at the end of the message depending on the options selected in the flags field.

BUS INTERFACE MESSAGES

Bus interface messages are used to access the Data Link Layer registers of modules across the Pi Bus. A Pi Bus master can read or write the Data Link Layer registers of slaves on the Pi Bus. The Data Link Layer registers of the M82916 can be write protected against bus interface messages.

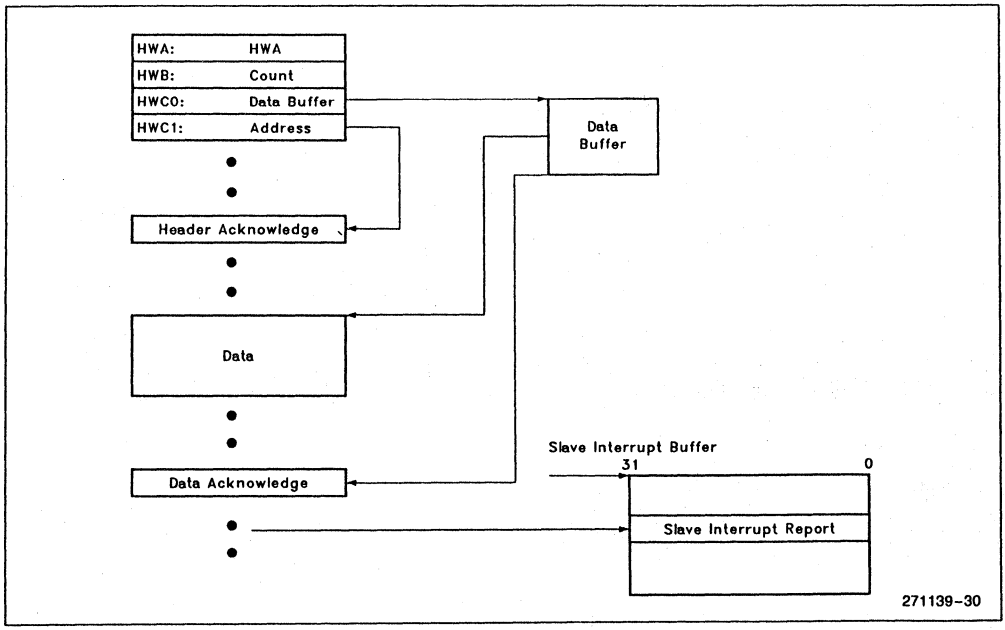


Figure 24. Block Message Read Sequence for a Direct Message

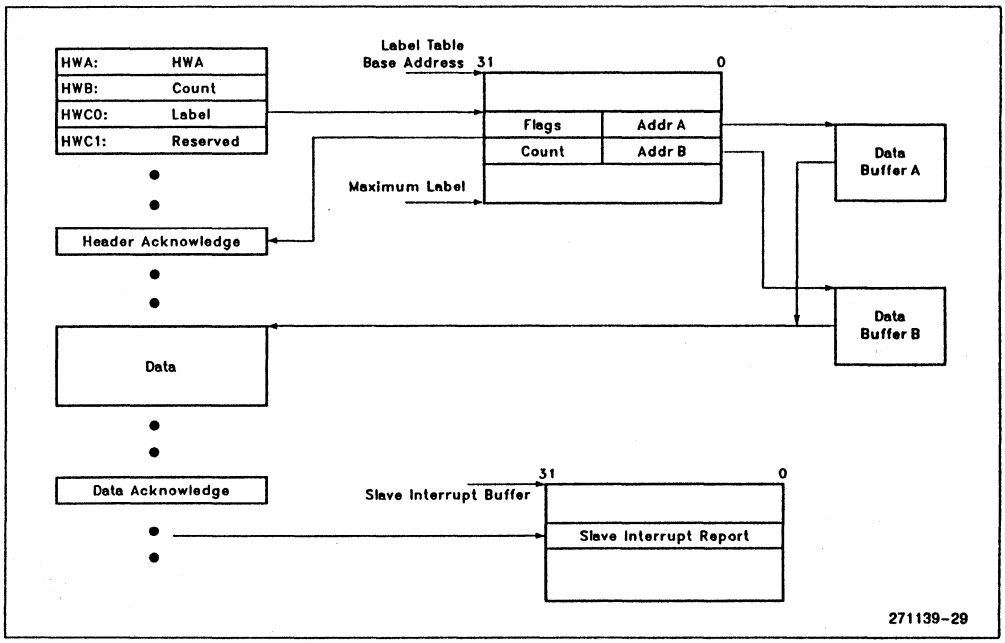


Figure 25. Block Message Read Sequence for a Label Message

The bus interface message sequence is similar to a block message. The header for bus interface messages is different than a block message header. For bus interface messages, only a two word header is used. The header contains the register address and the number of words in the message. The word count field can be specified so that sequential registers are accessed by a single bus interface message. The register address specifies the first register address of the message. The register contents are transmitted during the data cycles of the Pi Bus message.

PARAMETER WRITE MESSAGES

Parameter write messages are used to transfer small amounts of information between the master and one or more slaves. The information is part of the header word with only the header cycles and header acknowledge cycles constituting the entire Pi Bus message. The message does not require data cycles nor data acknowledge cycles like block messages. Two types of parameter write messages are implemented by the M82916. The first being a direct type and the second being an event filter type. Each message type is defined below.

The direct parameter write message is used to transfer three words of data quickly across the Pi Bus. The data is the last three words of the header. The first word of the header specifies the slave address to receive the message. As shown in Figure 26, the slave stores all four header words in an interrupt report. The interrupt is stored after the header acknowledge cycle.

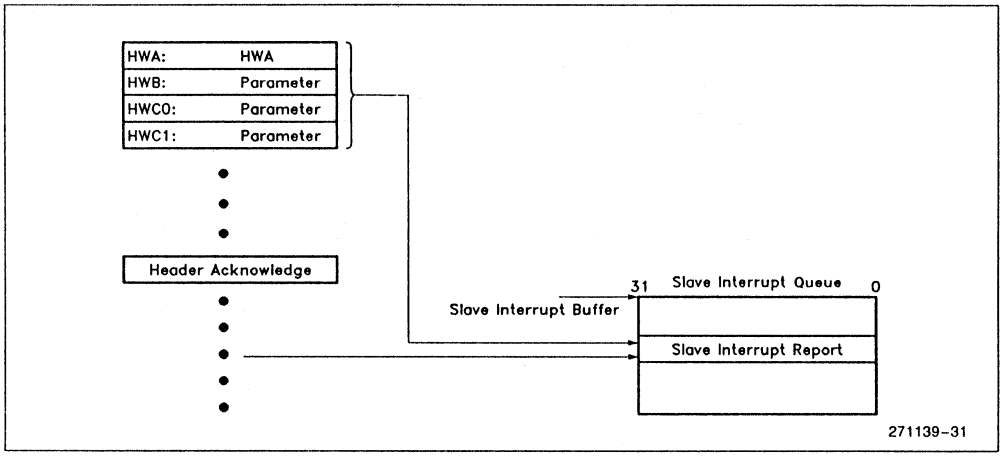


Figure 26. Parameter Write Sequence for a Direct Message

Event Filter parameter writes can be used to relay status information between modules. A label is defined in the header words which select an entry in the Event Filter Table (see Figure 27). The Event Filter Table is located in the local memory of the Slave M82916.

Each Event Filter block consists of an Event Flag and an Event Mask. The Event Flag word in the message header is OR'ed with the Event Flag of the filter block. The resulting word is then compared to the Event Mask word. If there is a match, an interrupt report is generated. If the resulting word does not match the Event Mask, the Event Flag is updated with the OR'ed word, and no interrupt report is generated. Event Filter parameter write messages are suitable for multiple task operations, where a module does not want to be interrupted until all modules working on tasks have reported back. Each module could be assigned a bit with the mask set to cause an interrupt only when all assigned bits have been set.

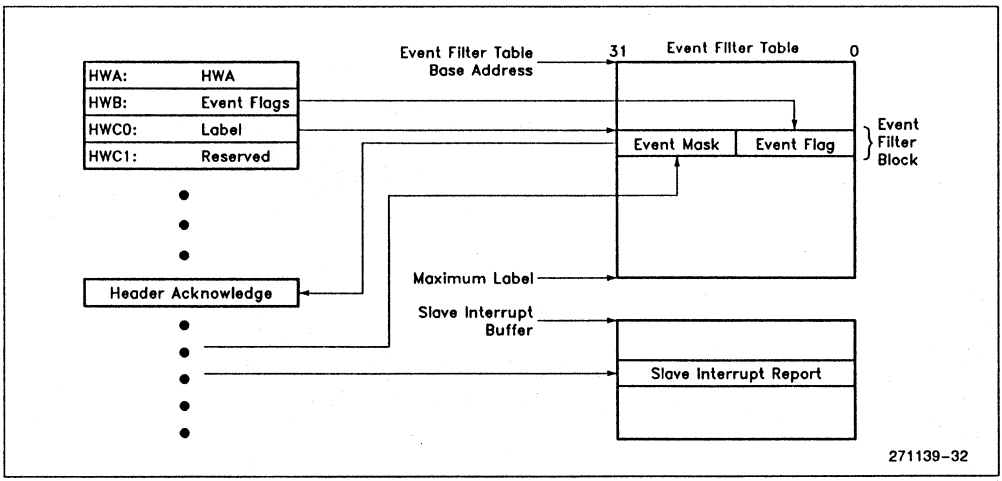


Figure 27. Parameter Write Sequence for Event Filter Message

ELEMENT TEST AND MAINTENANCE BUS (ETM BUS)

An "optional use" ETM Bus is provided as a standardized test interface for VLSI elements on a module. Each element utilizes a serial path to pass test and maintenance information. Both control and data are transmitted serially in a Master/Slave protocol. The M82916 interfaces with the ETM Bus through the On-Chip Monitor (OCM).

Facilities are provided to the ETM Bus Master for use in testing the M82916 and diagnosing system failures. The M82916 ETM Bus can be connected in either a star or ring configuration using the six standard ETM Bus signals described in the ETM Bus Signal Descriptions, Table 1C.

ETM Command Protocol

ETM Mode (ETMMODE) of logical 0 specifies the instruction mode. The ETM Select signal ($\overline{\text{ETMSEL}}$) is asserted low by the ETM Bus Master to indicate that the PBIU is to begin receiving data on the Data In line (ETMDIN) and send data out on the Data Out line (ETMDOUT). The received data is placed in the Command Register of the OCM, while the transmitted data is sent from the Response Register of the OCM. The release of $\overline{\text{ETMSEL}}$ causes the execution of the command received from the ETM Bus. All data transfers consist of 16 bits of data followed by an odd parity bit as shown in Figure 28. It should be noted that the LSB is always the first bit to be shifted in or out on the ETM data lines.

Data is latched on the falling edge of the ETM Clock. The OCM will begin receiving and transmitting data one clock cycle after the Select line is asserted low. The Select line must be released after the 16th data bit is transferred but before the parity bit is latched at the falling edge of the clock. The timing of the data transfer is shown in Figure 29.

ETM-OCM Commands

The OCM can process commands to update the status, echo commands to the ETM master, write and read PBIU registers, reset the chip, read the

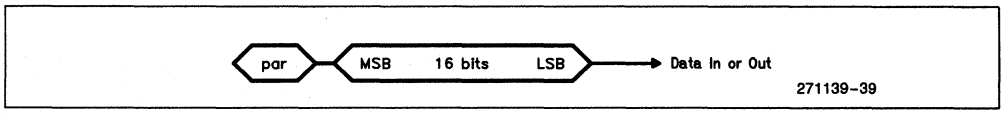


Figure 28. Data Transfer Format

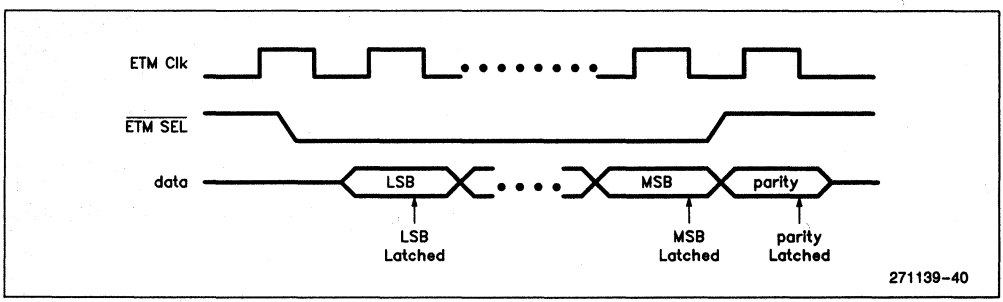


Figure 29. Data Transfer Timing

status, float all drivers, enable the Random Pattern Generator (RPG) and the Multiple Input Signature Register (MISR) and start, stop, and step PBIU clocks.

The ETM Write Register command can write to the Interrupt Enable Register and the PBIU Control Register (bits 0 through 7). The ETM Read Register Command can read from the PBIU Control Register, Local Bus Error Registers, and the Pi Bus Error Registers.

Chip Test Facilities

The PBIU has four chip test facilities available to the ETM Bus: signature analysis, boundary scan, scan string scan, and clock control. During signature analysis, the registers, counters, and other memory elements of the PBIU are configured into multiple shift registers or scan strings. Once an initial pattern of zeros is scanned into all scan strings, and the MISR and a seed into the RPG, the RPG will fill the scan strings with a random pattern. The PBIU is clocked once to pass the data through the combinational logic of the chip. The scan strings are then scanned again to collect the signature in the MISR and to fill them with a new random pattern. This is repeated until the proper test coverage is reached. The signature in the MISR is then read by the ETM bus and compared to the correct value.

The second chip test facility is boundary scan. All of the storage elements, registers, etc., which directly control or monitor the chip input are configured into two boundary scan strings, one input string and one output string. This permits loading of the chip boundaries with known patterns, clocking them to other chips and scanning the other chip boundary scan strings and comparing the results.

The third chip test facility is scan string scanning. Each of the 23 scan strings of the PBIU can be individually scanned by the ETM bus. This permits access to all internal storage elements (registers, counters, etc.) for reading and writing.

The fourth chip test facility is clock control. The OCM provides the capability to stop, start, and step the PBIU clocks by use of the Event Command.

PIN DESCRIPTIONS

Table 1A. Local Bus Signal Descriptions

Pin Name	I/O	Function
LAB (21–16) LAB (15–00)	O I/O	**LOCAL ADDRESS BUS:** The 22 local address bus lines output physical memory addresses when the M82916 has control of the bus. Address lines 21–16 are output only. Addresses 15–0 are bi-directional and are used as inputs when the M82916 is being accessed by the host CPU. These lines are floated when the M82916 does not have control of the Local Bus. LAB21 is the Most Significant Bit (MSB), and LAB00 is the Least Significant Bit (LSB).
LABPHI LABPLO	O I/O	**ADDRESS PARITY:** The address bus is parity protected and uses odd parity. The LABPHI line applies to address lines 21–16 and the MCMD (02–00) lines. LABPLO applies to address lines 15–00. These lines are floated when the M82916 does not have control of the Local Bus.
LDB (31–00)	I/O	**LOCAL DATA BUS:** The local data bus supports 32-bit data transfers with an odd parity bit associated with each 16-bit word. Data bus lines 15–00 are the low word and are used during 16-bit transfers. Bits 31–16 constitute the high word. All 32 LDB lines are floated when the PBIU is not the Local Bus master. LDB31 is the MSB, and LDB00 is the LSB.
LDBPHI LDBPLO	I/O I/O	**DATA BUS PARITY:** Odd parity is checked or generated during read or write operations respectively. LDBPHI is used for LDB lines 31–16, and LDBPLO is used for LDB lines 15–00. These lines are floated when the M82916 does not have control of the Local Bus.
MCMD (02–00)	O	**MEMORY COMMAND BUS:** The MCMD lines are used to determine type of memory accesses. A memory access can be a read or a write and it can be 16 or 32 bits wide. MCMD02 is the MSB, and MCMD00 is the LSB. The MCMD lines are included with LAB lines 21–16 for the parity line LABPHI. These lines are floated when the M82916 does not have control of the Local Bus.
$\overline{\text{BPCHK}}$	I/O	**BLOCK PROTECT CHECK:** Indicates a Block Protect Check sequence has been initiated by the Local Bus Master. The M82916 will perform a Block Protect Check sequence under the following conditions: — First write operation following a reset. — First write operation following notification to the M82916 that the protect bits have changed. — Before crossing a 4K memory address boundary. — During a Read-Modify-Write Sequence.
$\overline{\text{RU}}$	I	**REGISTER UPDATE OPERATION:** A low on this input pin indicates that a Register Update cycle is beginning. This pin in used as a chip-select input to execute register accesses.
$\overline{\text{RUSUS}}$	I/O	**REGISTER UPDATE SUSPEND:** The M82916 asserts a low on this pin during accesses of the Data Link Layer registers. While $\overline{\text{RUSUS}}$ is low, the register data is being synchronized between the LBCLK and Pi Bus clock. $\overline{\text{RUSUS}}$ going high indicates the Data Link Layer register data has been written or is available to be read.
$\overline{\text{RUSEL}}$	I	**REGISTER UPDATE ADDRESS SELECT:** The address space used for Register Updates is selected with the $\overline{\text{RUSEL}}$ bit. With $\overline{\text{RUSEL}}$ pin tied high (Vcc), the register addresses are contained in the address space of 3100–31FF and B100–B1FF. With the $\overline{\text{RUSEL}}$ pin tied low (GND), the register addresses are contained in the address space of 3000–30FF and B000–B0FF. This bit allows two PBIUs to be resident on the same bus and still be addressed individually.

Table 1A. Local Bus Signal Descriptions (Continued)

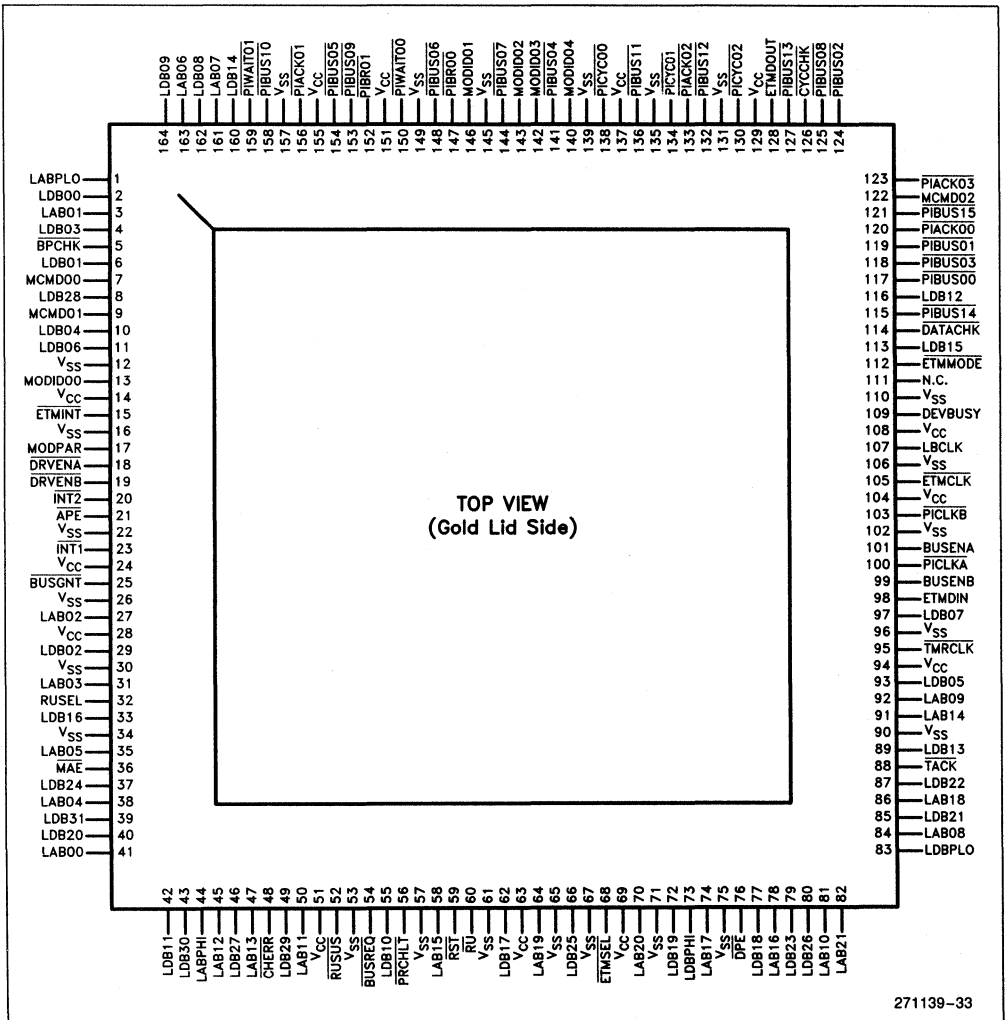
Pin Name	I/O	Function
$\overline{TACK}$	I/O	**TRANSFER ACKNOWLEDGE:** $\overline{TACK}$ is asserted by the addressed device and indicates a valid data transfer on the Local Bus. $\overline{TACK}$ must be low along with $\overline{BUSGNT}$ to give control of the Local Bus to the M82916 after Bus Request has gone low. This pin is an input and floated by the M82916 except during one clock cycle of a Register Update sequence.
$\overline{BUSREQ}$	O	**LOCAL BUS REQUEST:** A low is asserted by the M82916 when an access of the Local Bus is requested. The M82916 releases control of the bus by de-asserting this signal.
$\overline{BUSGNT}$	I	**LOCAL BUS GRANT:** A low on this input and the $\overline{TACK}$ input indicates access of the Local Bus has been granted to the M82916. This input must remain low until $\overline{BUSREQ}$ is released.
$\overline{APE}$	I	**ADDRESS PARITY ERROR:** $\overline{APE}$ is an active low input indicating an error on the Local Address Bus has occurred during memory operations for which the MCMD field is not a NOP.
$\overline{DPE}$	I/O	**DATA PARITY ERROR:** A low input on this pin indicates that a parity error has been detected during a memory operation. A low output on this pin indicates that a parity error existed during a register update operation. This signal is otherwise floated by the M82916.
$\overline{CHERR}$	O	**I/O CHANNEL ERROR:** Signal which is used to indicate an illegal Register Update or reserved Register Update operation. The M82916 will drive this output low when an invalid register address is detected during an RU operation.
$\overline{MAE}$	I	**MEMORY ACCESS ERROR:** Active low input indicating an access to unimplemented memory.
LBCLK	I	**LOCAL BUS CLOCK:** Free running input clock used for all Local Bus timing. Inputs are latched in on the rising edge of the clock. Outputs are clocked out by the rising edge of this clock.
$\overline{INT1}$	O	**INTERRUPT 1:** A level 1 interrupt is indicated to the host CPU when this pin is driven to a low level. The M82916 will halt Pi Bus operations when a level 1 interrupt occurs. When in the interrupt pulse mode, a low pulse on the $\overline{INT1}$ pin indicates that the current Pi Bus message ended with an error. This occurs during Slave Mode only.
$\overline{INT2}$	O	**INTERRUPT 2:** A level 2 interrupt is indicated to the host CPU when this pin is driven to a low level. Pi Bus operation will continue when the M82916 generates a level 2 interrupt. When in the interrupt pulse mode, a low pulse on the $\overline{INT2}$ pin indicates that the current Pi Bus message ended successfully. This occurs during Slave Mode only.
$\overline{TMRCLK}$	I	**TIMER CLOCK:** An input for the internal 32-bit System Timer. The System Timer is incremented by a high to low transition of the $\overline{TMRCLK}$ signal. $\overline{TMRCLK}$ is expected to be 1 MHz.
$\overline{PRCHLT}$	I	**PROCESSOR HALT:** Active low input indicates that the host processor has been halted. It is used to halt the incrementing of the System Timer.
$\overline{RST}$	I	**SYSTEM RESET:** A low on this pin for 10 LBCLK cycles will reset the M82916.
DEVBUSY	I	**DEVICE BUSY:** An active high signal which is asserted by the host to indicate that it is busy and cannot respond to accesses by the M82916. DEVBUSY is valid only when the PBIU Control register specifies "Pulsed Interrupt" mode.

Table 1B. Pi Bus Bus Signal Descriptions

Pin Name	I/O	Function
$\overline{\text{PIBUS}}$ (15–00)	I/O	**Pi BUS DATA:** Local data interface between the M82916 and the Pi Bus transceivers. The M82916 drives the bus during the first half of the bus cycle and reads the bus during the second half of the bus cycle.
$\overline{\text{DATACHK}}$	I/O	**Pi BUS DATA CHECK:** $\overline{\text{DATACHK}}$ is the parity line used to generate even parity on the Pi Bus data lines.
$\overline{\text{PICYC}}$ (02–00)	I/O	**Pi BUS CYCLE TYPE:** Asserted by the Pi Bus master to indicate the current Pi Bus cycle type.
$\overline{\text{CYCCHK}}$	I/O	**Pi BUS CYCLE CHECK:** $\overline{\text{CYCCHK}}$ is the parity line used to generate even parity on the Pi Bus cycle type bus.
$\overline{\text{PIACK}}$ (03–00)	I/O	**Pi BUS ACKNOWLEDGE:** Asserted by Pi Bus slaves to indicate synchronization or uncorrectable detected errors.
$\overline{\text{PIWAIT}}$ (01, 00)	I/O	**Pi BUS WAIT:** Asserted by the current Pi Bus master or slaves to obtain extra non-transfer bus cycles.
$\overline{\text{PIBR}}$ (01, 00)	I/O	**Pi BUS REQUEST:** Request is asserted by any Pi Bus interface unit if its Vie Priority register is higher than that of the current Pi Bus master.
MODID (04-00)	I	**Pi BUS MODULE ID:** An input bus that provides the M82916 with a 5-bit Module Slave ID.
MODPAR	I	**Pi BUS MODULE ID PARITY:** MODPAR is the parity bit used to generate odd parity on the Module ID bus.
$\overline{\text{PICLK}}$ (A, B)	I	**Pi BUS CLOCKS A AND B:** Input signals providing the Pi Bus clock to the M82916.
$\overline{\text{DRVEN}}$ (A, B)	O	**Pi BUS DRIVER ENABLES A AND B:** Active low output that allows the Pi Bus transceivers to drive the Pi Bus.
BUSEN (A, B)	O	**Pi BUS ENABLES A AND B:** Active high output that enables the transceivers to drive the local Pi Bus.

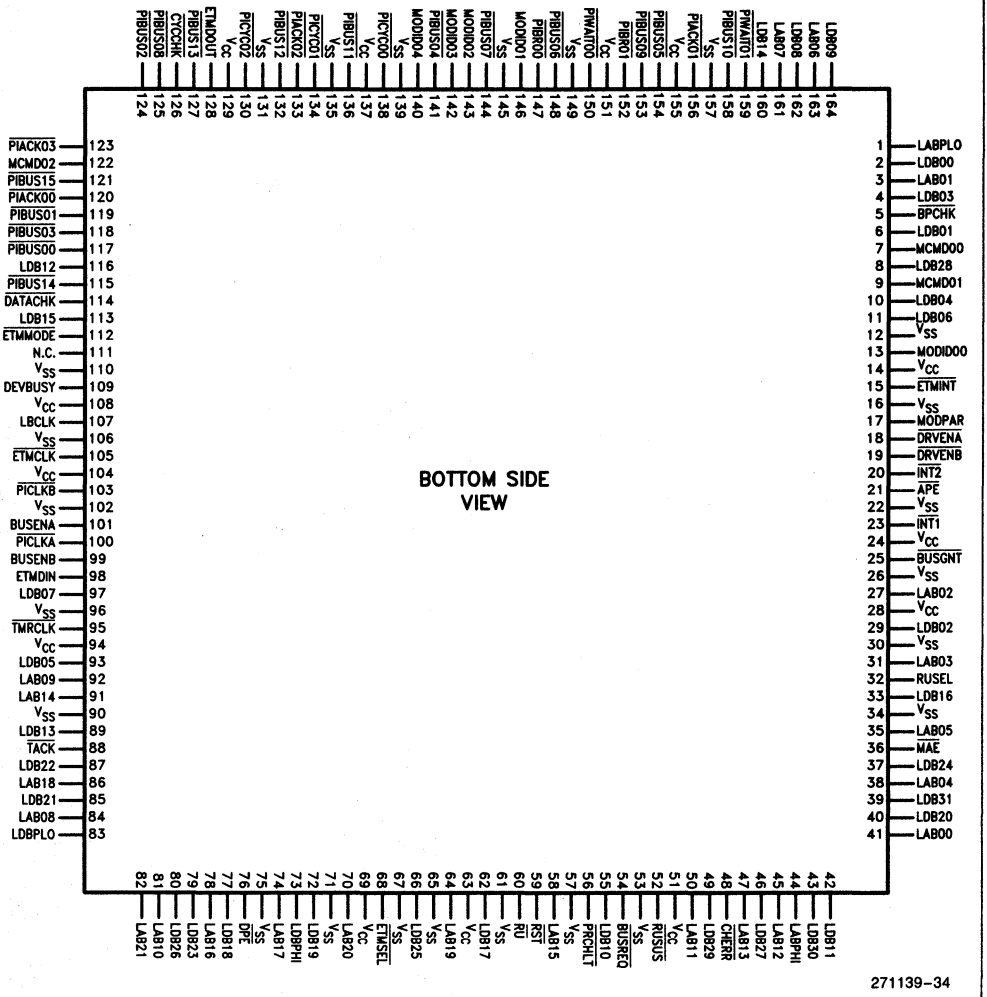
Table 1C. ETM Bus Signal Descriptions

Pin Name	I/O	Function
$\overline{\text{ETMCLK}}$	I	**ETM CLOCK:** Free running input clock (6.25 MHz). All ETM control signals and data are latched on the falling edge of the $\overline{\text{ETMCLK}}$.
$\overline{\text{ETMSEL}}$	I	**ETM SELECT:** A low on this input pin indicates a data transfer on the ETMDIN and ETMOUT lines. When the $\overline{\text{ETMSEL}}$ signal is released, any command received from the ETM Bus will be executed by the M82916.
$\overline{\text{ETMINT}}$	O	**ETM INTERRUPT:** An active low signal asserted by the M82916 to indicate that an error condition has been recorded in the ETM Status Register.
ETMMODE	I	**ETM MODE:** A low on this input indicates the ETM bus is in the command mode. A high indicates the ETM Bus is in the scan string mode.
ETMDIN	I	**ETM DATA IN:** Input from the ETM Bus for data received by the M82916.
ETMDOUT	O	**ETM DATA OUT:** Output to the ETM Bus for data transmitted by the M82916.



NOTE:
Package dimensions can be found in Chapter 2, Military Packaging Outlines and Dimensions, of the *Intel Military Handbook*.

Figure 30. Pin Diagram (Top View)



271139-34

NOTE:
Package dimensions can be found in Chapter 2, Military Packaging Outlines and Dimensions, of the *Intel Military Handbook*.

Figure 31. Pin Diagram (Bottom Side View)

Table 2. M82916 PBIU CQFP Pinout—In Pin Order

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	LABPLO	42	LDB11	83	LDBPLO	124	PIBUS02
2	LDB00	43	LDB30	84	LAB08	125	PIBUS08
3	LAB01	44	LABPHI	85	LDB21	126	CYCCHK
4	LDB03	45	LAB12	86	LAB18	127	PIBUS13
5	BPCHK	46	LDB27	87	LDB22	128	ETMDOUT
6	LDB01	47	LAB13	88	TACK	129	V_{CC}
7	MCMD00	48	CHERR	89	LDB13	130	PICYC02
8	LDB28	49	LDB29	90	V_{SS}	131	V_{SS}
9	MCMD01	50	LAB11	91	LAB14	132	PIBUS12
10	LDB04	51	V_{CC}	92	LAB09	133	PIACK02
11	LDB06	52	RUSUS	93	LDB05	134	PICYC01
12	V_{SS}	53	V_{SS}	94	V_{CC}	135	V_{SS}
13	MODID00	54	BUSREQ	95	TMRCLK	136	PIBUS11
14	V_{CC}	55	LDB10	96	V_{SS}	137	V_{CC}
15	ETMINT	56	PRCHLT	97	LDB07	138	PICYC00
16	V_{SS}	57	V_{SS}	98	ETMDIN	139	V_{SS}
17	MODPAR	58	LAB15	99	BUSENB	140	MODID04
18	DRVENA	59	RST	100	PICLKA	141	PIBUS04
19	DRVENB	60	RU	101	BUSENA	142	MODID03
20	INT2	61	V_{SS}	102	V_{SS}	143	MODID02
21	APE	62	LDB17	103	PICLKB	144	PIBUS07
22	V_{SS}	63	V_{CC}	104	V_{CC}	145	V_{SS}
23	INT1	64	LAB19	105	ETMCLK	146	MODID01
24	V_{CC}	65	V_{SS}	106	V_{SS}	147	PIBR00
25	BUSGNT	66	LDB25	107	LBCLK	148	PIBUS06
26	V_{SS}	67	V_{SS}	108	V_{CC}	149	V_{SS}
27	LAB02	68	ETMSEL	109	DEVBUSY	150	PIWAIT00
28	V_{CC}	69	V_{CC}	110	V_{SS}	151	V_{CC}
29	LDB02	70	LAB20	111	N.C.	152	PIBR01
30	V_{SS}	71	V_{SS}	112	ETMMODE	153	PIBUS09
31	LAB03	72	LDB19	113	LDB15	154	PIBUS05
32	RUSEL	73	LDBPHI	114	DATACHK	155	V_{CC}
33	LDB16	74	LAB17	115	PIBUS14	156	PIACK01
34	V_{SS}	75	V_{SS}	116	LDB12	157	V_{SS}
35	LAB05	76	DPE	117	PIBUS00	158	PIBUS10
36	MAE	77	LDB18	118	PIBUS03	159	PIWAIT01
37	LDB24	78	LAB16	119	PIBUS01	160	LDB14
38	LAB04	79	LDB23	120	PIACK00	161	LAB07
39	LDB31	80	LDB26	121	PIBUS15	162	LDB08
40	LDB20	81	LAB10	122	MCMD02	163	LAB06
41	LAB00	82	LAB21	123	PIACK03	164	LDB09

NOTE:
1. N.C.—No Connect. Pin must be left floating.

Table 3. M82916 PBIU CQFP Pinout—In Signal Order

Signal	Pin
LAB21	82
LAB20	70
LAB19	64
LAB18	86
LAB17	74
LAB16	78
LAB15	58
LAB14	91
LAB13	47
LAB12	45
LAB11	50
LAB10	81
LAB09	92
LAB08	84
LAB07	161
LAB06	163
LAB05	35
LAB04	38
LAB03	31
LAB02	27
LAB01	3
LAB00	41
LABPHI	44
LABPLO	1
MCMD02	122
MCMD01	9
MCMD00	7
$\overline{\text{BPCHK}}$	5
$\overline{\text{RU}}$	60
$\overline{\text{RUSUS}}$	52
RUSEL	32
$\overline{\text{TACK}}$	88
$\overline{\text{BUSREQ}}$	54
$\overline{\text{BUSGNT}}$	25
$\overline{\text{APE}}$	21
$\overline{\text{DPE}}$	76
$\overline{\text{MAE}}$	36
$\overline{\text{CHERR}}$	48
LBCLK	107

Signal	Pin
LDB31	39
LDB30	43
LDB29	49
LDB28	8
LDB27	46
LDB26	80
LDB25	66
LDB24	37
LDB23	79
LDB22	87
LDB21	85
LDB20	40
LDB19	72
LDB18	77
LDB17	62
LDB16	33
LDB15	113
LDB14	160
LDB13	89
LDB12	116
LDB11	42
LDB10	55
LDB09	164
LDB08	162
LDB07	97
LDB06	11
LDB05	93
LDB04	10
LDB03	4
LDB02	29
LDB01	6
LDB00	2
LDBPHI	73
LDBPLO	83
$\overline{\text{ETMCLK}}$	105
$\overline{\text{ETMSEL}}$	68
$\overline{\text{ETMINT}}$	15
ETMMODE	112
ETMDIN	98
ETMDOUT	128
N.C.	111

Signal	Pin
$\overline{\text{PIBUS15}}$	121
$\overline{\text{PIBUS14}}$	115
$\overline{\text{PIBUS13}}$	127
$\overline{\text{PIBUS12}}$	132
$\overline{\text{PIBUS11}}$	136
$\overline{\text{PIBUS10}}$	158
$\overline{\text{PIBUS09}}$	153
$\overline{\text{PIBUS08}}$	125
$\overline{\text{PIBUS07}}$	144
$\overline{\text{PIBUS06}}$	148
$\overline{\text{PIBUS05}}$	154
$\overline{\text{PIBUS04}}$	141
$\overline{\text{PIBUS03}}$	118
$\overline{\text{PIBUS02}}$	124
$\overline{\text{PIBUS01}}$	119
$\overline{\text{PIBUS00}}$	117
$\overline{\text{DATACHK}}$	114
$\overline{\text{PICYC02}}$	130
$\overline{\text{PICYC01}}$	134
$\overline{\text{PICYC00}}$	138
$\overline{\text{CYCCHK}}$	126
$\overline{\text{PIACK03}}$	123
$\overline{\text{PIACK02}}$	133
$\overline{\text{PIACK01}}$	156
$\overline{\text{PIACK00}}$	120
$\overline{\text{PIBR01}}$	152
$\overline{\text{PIBR00}}$	147
$\overline{\text{PIWAIT01}}$	159
$\overline{\text{PIWAIT00}}$	150
MODID04	140
MODID03	142
MODID02	143
MODID01	146
MODID00	13
MODPAR	17
$\overline{\text{PICLKA}}$	100
$\overline{\text{PICLKB}}$	103
$\overline{\text{DRVENA}}$	18
$\overline{\text{DRVENB}}$	19
BUSENA	101
BUSENB	99

Signal	Pin
$\overline{\text{INT1}}$	23
$\overline{\text{INT2}}$	20
$\overline{\text{TMRCLK}}$	95
$\overline{\text{PRCHLT}}$	56
$\overline{\text{RST}}$	59
DEVBUSY	109
V_{CC}	14
V_{CC}	24
V_{CC}	28
V_{CC}	51
V_{CC}	63
V_{CC}	69
V_{CC}	94
V_{CC}	104
V_{CC}	108
V_{CC}	129
V_{CC}	137
V_{CC}	151
V_{CC}	155
V_{SS}	12
V_{SS}	16
V_{SS}	22
V_{SS}	26
V_{SS}	30
V_{SS}	34
V_{SS}	53
V_{SS}	57
V_{SS}	61
V_{SS}	65
V_{SS}	67
V_{SS}	71
V_{SS}	75
V_{SS}	90
V_{SS}	96
V_{SS}	102
V_{SS}	106
V_{SS}	110
V_{SS}	131
V_{SS}	135
V_{SS}	139
V_{SS}	145
V_{SS}	149
V_{SS}	157

NOTE:
1. N.C.—No Connect. Pin must be left floating.

ABSOLUTE MAXIMUM RATINGS*

Case Temperature under Bias . . . −55°C to +125°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin to V_{SS} . . . . . 0V to +6.0V

NOTICE: This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice. Verify with your local Intel sales office that you have the latest data sheet before finalizing a design.

**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.*

ADVANCED INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
T_C	Case Temperature (Instant On)	−55	+125	°C
V_{CC}	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		0.80	V	Note 1
V_{IL1}	Input Low Voltage		0.60	V	Note 1
V_{IH}	Input High Voltage	2.0		V	Note 1
V_{IH1}	Input High Voltage	2.2		V	Note 1
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 3.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OL} = 100 μA
I_{IL}	Input Leakage Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V
R_{PULLUP}	Input Pull-Up Resistor	5.7K	25K	Ω	V_{IN} = V_{IH} and V_{IL} Note 2
I_{CC}	Power Supply Current		225	mA	12.5 MHz

NOTES:
1. V_{IL1} and V_{IH1} apply to clock pins: $\overline{\text{PRCHLT}}$, $\overline{\text{RST}}$, $\overline{\text{TMRCLK}}$, $\overline{\text{PICLKA}}$, $\overline{\text{PICLKB}}$, $\overline{\text{ETMCLK}}$, LBCLK, DEVBUSY, MODID03, and MODID02. V_{IL} and V_{IH} apply to all other pins.
2. Only applicable for ETMMODE, $\overline{\text{ETMSEL}}$, and $\overline{\text{ETMDIN}}$.

A.C. SPECIFICATION TABLES (Over Specified Operating Conditions)

LBCLK INPUT TIMINGS

Symbol	Parameter	Min	Max	Units	Fig.	Comments
	LBCLK Frequency		12.5	MHz	32	
T_1	LBCLK Period	80		ns	32	
T_2	LBCLK High Time	30		ns	32	
T_3	LBCLK Low Time	30		ns	32	
T_4	LBCLK Fall Time		5	ns	32	
T_5	LBCLK Rise Time		5	ns	32	
	LBCLK to PICLK Frequency Ratio	0.25	1	—		

DMA OUTPUT TIMINGS

Symbol	Parameter	Min	Max	Units	Fig.	Comments
T_6	LBCLK Rise to $\overline{\text{BUSREQ}}$ Valid		26	ns	33	
T_7	$\overline{\text{BUSGNT}}$ Fall to LAB Valid		41	ns	33	
T_8	$\overline{\text{BUSGNT}}$ Rise to LAB Float		24	ns	33	
T_9	LBCLK Rise to LAB Valid		58	ns	33	
T_{10}	LBCLK Rise to LAB Float		46	ns	33	
T_{11}	$\overline{\text{BUSGNT}}$ Fall to MCMD Active		41	ns	33	
T_{12}	$\overline{\text{BUSGNT}}$ Rise to MCMD Float		24	ns	33	
T_{13}	LBCLK Rise to MCMD Valid		54	ns	33	
T_{14}	LBCLK Rise to MCMD Float		46	ns	33	
T_{15A}	LBCLK Rise to LDB Valid		47	ns	33	First Data Word
T_{15B}	LBCLK Rise to LDB Valid		38	ns	33	Next Data Word
T_{16}	LBCLK Rise to LDB Float		22	ns	33	

INPUT TIMINGS

Symbol	Parameter	Min	Max	Units	Fig.	Comments
T_{17}	$\overline{\text{BUSGNT}}$ Setup to LBCLK Rise	15		ns	33	
T_{18}	$\overline{\text{BUSGNT}}$ Hold after LBCLK Rise	12		ns	33	
T_{19}	LAB, LDB Setup to LBCLK Rise	12		ns	34	
T_{20}	LAB, LDB Hold after LBCLK Rise	13		ns	35	
T_{21}	Error Setup to LBCLK Rise	12		ns	33, 35	Note 3
T_{22}	Error Hold after LBCLK Rise	11		ns	33, 35	Note 3
T_{23}	$\overline{\text{RU}}$ Setup to LBCLK Rise	11		ns	34	
T_{24}	$\overline{\text{RU}}$ Hold after LBCLK Rise	7		ns	34	
T_{25}	$\overline{\text{RU}}$ Low Width		1	T_1	34	

NOTE:
3. Error signals include: $\overline{\text{APE}}$, $\overline{\text{MAE}}$, $\overline{\text{DPE}}$, and $\overline{\text{BPCHK}}$.

$\overline{RU}$ OUTPUT TIMINGS

Symbol	Parameter	Min	Max	Units	Fig.	Comments
T_{26}	LBCLK Rise to $\overline{RUSUS}$ Low		56	ns	34	
T_{27}	LBCLK Rise to $\overline{RUSUS}$ High		41	ns	34	
T_{28}	LBCLK Rise to $\overline{DPE}$ Valid		65	ns	34	
T_{29}	LBCLK Rise to $\overline{DPE}$ High		53	ns	34	
T_{30}	LBCLK Rise to $\overline{DPE}$ Float		44	ns	34	
T_{31}	LBCLK Rise to $\overline{CHERR}$ Valid		46	ns	34	
T_{32}	LBCLK Rise to $\overline{CHERR}$ High		31	ns	34	
T_{33}	LBCLK Rise to $\overline{CHERR}$ Float		43	ns	34	

OTHER TIMINGS

Symbol	Parameter	Min	Max	Units	Fig.	Comments
T_{34}	$\overline{TACK}$ Setup to LBCLK Rise	13		ns	33, 34	
T_{35}	$\overline{TACK}$ Hold after LBCLK Rise	10		ns	33, 34	
T_{36}	LBCLK Rise to $\overline{TACK}$ Low		47	ns	34	
T_{37}	LBCLK Rise to $\overline{TACK}$ Float		33	ns	34	
T_{38}	LBCLK Rise to $\overline{BPCHK}$		41	ns	35	

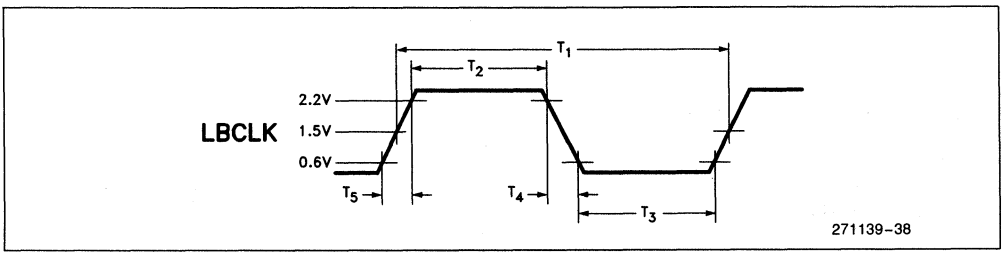


Figure 32. LBCLK Timings

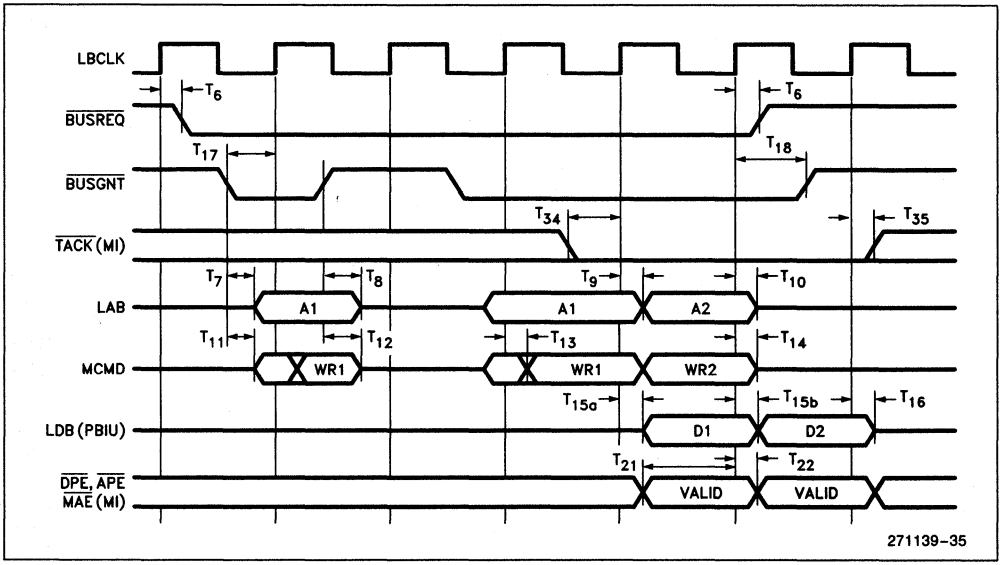


Figure 33. Local Bus Acquisition and DMA Write Timings

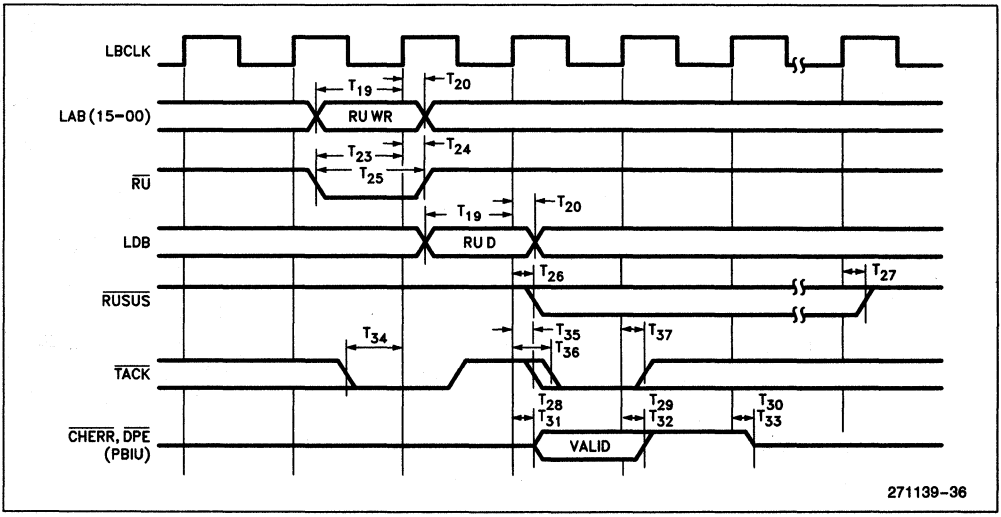


Figure 34. Register Update Timings

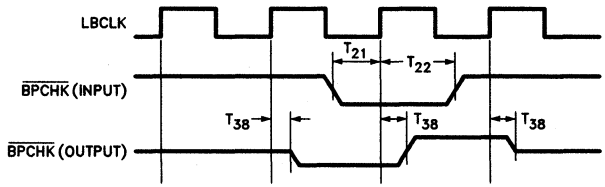
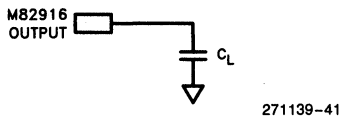
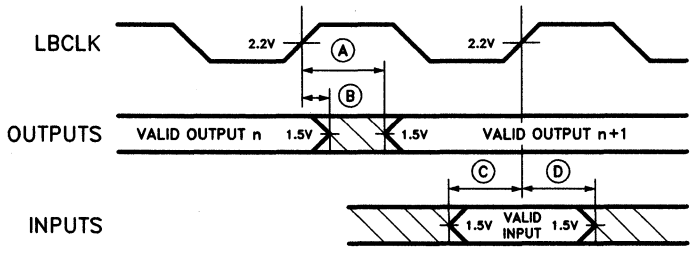


Figure 35. BPCHK Timings



C_L = 50 pF; all pins

Figure 36. Load Circuit



Legend:
A. Maximum output delay spec
B. Minimum output delay spec
C. Minimum input setup spec
D. Minimum input hold spec

Figure 37. Drive Levels and Measurement Points for A.C. Specifications

DOMESTIC SALES OFFICES

ALABAMA

†Intel Corp.
5015 Bradford Dr., #2
Huntsville 35805
Tel: (205) 830-4010
FAX: (205) 837-2640

ARIZONA

†Intel Corp.
410 North 44th Street
Suite 500
Phoenix 85008
Tel: (602) 231-0386
FAX: (602) 244-0446

Intel Corp.
7225 N. Mona Lisa Rd.
Suite 215
Tucson 85741
Tel: (602) 544-0227
FAX: (602) 544-0232

CALIFORNIA

†Intel Corp.
21515 Vanowen Street
Suite 116
Canoga Park 91303
Tel: (818) 704-8500
FAX: (818) 340-1144

†Intel Corp.
2250 E. Imperial Highway
Suite 218
El Segundo 90245
Tel: (213) 640-6040
FAX: (213) 640-7133

Intel Corp.
1 Sierra Gate Plaza
Suite 280C
Roseville 95678
Tel: (916) 782-8086
FAX: (916) 782-8153

†Intel Corp.
9665 Chesapeake Dr.
Suite 325
San Diego 92123
Tel: (619) 292-8086
FAX: (619) 292-0628

†Intel Corp.*
400 N. Tustin Avenue
Suite 450
Santa Ana 92705
Tel: (714) 835-9642
TWX: 910-595-1114
FAX: (714) 541-9157

†Intel Corp.*
San Tomas 4
2700 San Tomas Expressway
2nd Floor
Santa Clara 95051
Tel: (408) 986-8086
TWX: 910-338-0255
FAX: (408) 727-2620

COLORADO

Intel Corp.
4445 Northpark Drive
Suite 100
Colorado Springs 80907
Tel: (719) 594-6622
FAX: (303) 594-0720

†Intel Corp.*
650 S. Cherry St.
Suite 915
Denver 80222
Tel: (303) 321-8086
TWX: 910-931-2289
FAX: (303) 322-8670

CONNECTICUT

Intel Corp.
301 Lee Farm Corporate Park
83 Wooster Heights Rd.
Danbury 06810
Tel: (203) 748-3130
FAX: (203) 794-0339

FLORIDA

†Intel Corp.
800 Fairway Drive
Suite 160
Deerfield Beach 33441
Tel: (305) 421-0506
FAX: (305) 421-2444

†Intel Corp.
5850 T.G. Lee Blvd.
Suite 340
Orlando 32822
Tel: (407) 240-8000
FAX: (407) 240-8097

Intel Corp.
11300 4th Street North
Suite 170
St. Petersburg 33716
Tel: (813) 577-2413
FAX: (813) 578-1607

GEORGIA

†Intel Corp.
20 Technology Parkway
Suite 150
Norcross 30092
Tel: (404) 449-0541
FAX: (404) 605-9762

ILLINOIS

†Intel Corp.*
Woodfield Corp. Center III
300 N. Martingale Road
Suite 400
Schaumburg 60173
Tel: (708) 605-8031
FAX: (708) 706-9762

INDIANA

Intel Corp.
8910 Purdue Road
Suite 350
Indianapolis 46268
Tel: (317) 875-0623
FAX: (317) 875-8938

IOWA

Intel Corp.
1930 St. Andrews Drive N.E.
2nd Floor
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS

†Intel Corp.
10985 Cody St.
Suite 140
Overland Park 66210
Tel: (913) 345-2727
FAX: (913) 345-2076

MARYLAND

†Intel Corp.*
10010 Junction Dr.
Suite 200
Annapolis Junction 20701
Tel: (301) 206-2860
FAX: (301) 206-3677
(301) 206-3678

MASSACHUSETTS

†Intel Corp.*
Westford Corp. Center
3 Carlisle Road
2nd Floor
Westford 01886
Tel: (508) 692-0960
TWX: 710-343-6333
FAX: (508) 692-7867

MICHIGAN

†Intel Corp.
7071 Orchard Lake Road
Suite 100
West Bloomfield 48322
Tel: (313) 851-8096
FAX: (313) 851-8770

MINNESOTA

†Intel Corp.
3500 W. 80th St.
Suite 360
Bloomington 55431
Tel: (612) 835-6722
TWX: 910-576-2867
FAX: (612) 831-6497

MISSOURI

†Intel Corp.
4203 Earth City Expressway
Suite 131
Earth City 63045
Tel: (314) 291-1990
FAX: (314) 291-4341

NEW JERSEY

†Intel Corp.*
Parkway 109 Office Center
328 Newman Springs Road
Red Bank 07701
Tel: (201) 747-2233
FAX: (201) 747-0983

Intel Corp.
280 Corporate Center
75 Livingston Avenue
First Floor
Roseland 07068
Tel: (201) 740-0111
FAX: (201) 740-0626

NEW YORK

Intel Corp.*
850 Crosskeys Office Park
Fairport 14450
Tel: (716) 425-2750
TWX: 510-253-7391
FAX: (716) 223-2561

†Intel Corp.*
2950 Express Dr., South
Suite 130
Islandia 11722
Tel: (516) 231-3300
TWX: 510-227-6236
FAX: (516) 348-7939

†Intel Corp.
Westage Business Center
Bldg. 300, Route 9
Fishkill 12524
Tel: (914) 897-3860
FAX: (914) 897-3125

NORTH CAROLINA

†Intel Corp.
5800 Executive Center Dr.
Suite 105
Charlotte 28212
Tel: (704) 568-8966
FAX: (704) 535-2236

†Intel Corp.
5540 Centerview Dr.
Suite 215
Raleigh 27606
Tel: (919) 851-9537
FAX: (919) 851-8974

OHIO

†Intel Corp.*
3401 Park Center Drive
Suite 220
Dayton 45414
Tel: (513) 890-5350
TWX: 810-450-2528
FAX: (513) 890-8658

†Intel Corp.*
25700 Science Park Dr.
Suite 100
Beachwood 44122
Tel: (216) 464-2736
TWX: 810-427-9298
FAX: (804) 282-0673

OKLAHOMA

Intel Corp.
6801 N. Broadway
Suite 115
Oklahoma City 73162
Tel: (405) 848-8086
FAX: (405) 840-9819

OREGON

†Intel Corp.
15254 N.W. Greenbrier Parkway
Building B
Beaverton 97005
Tel: (503) 645-8051
TWX: 910-467-8741
FAX: (503) 645-8181

PENNSYLVANIA

†Intel Corp.*
925 Harvest Drive
Suite 200
Blue Bell 19422
Tel: (215) 641-1000
FAX: (215) 641-0785

†Intel Corp.*
400 Penn Center Blvd.
Suite 610
Pittsburgh 15235
Tel: (412) 823-4970
FAX: (412) 829-7578

PUERTO RICO

†Intel Corp.
South Industrial Park
P.O. Box 910
Las Piedras 00671
Tel: (809) 733-8616

TEXAS

Intel Corp.
8911 Capital of Texas Hwy.
Austin 78759
Tel: (512) 794-8086
FAX: (512) 338-9335

†Intel Corp.*
12000 Ford Road
Suite 400
Dallas 75248
Tel: (214) 241-8087
FAX: (214) 484-1180

†Intel Corp.*
7322 S.W. Freeway
Suite 1490
Houston 77074
Tel: (713) 988-8086
TWX: 910-881-2490
FAX: (713) 988-3660

UTAH

†Intel Corp.
428 East 6400 South
Suite 104
Murray 84107
Tel: (801) 263-8051
FAX: (801) 268-1457

VIRGINIA

†Intel Corp.
1504 Santa Rosa Road
Suite 108
Richmond 23288
Tel: (804) 282-5668
FAX: (216) 464-2270

WASHINGTON

†Intel Corp.
155 108th Avenue N.E.
Suite 386
Bellevue 98004
Tel: (206) 453-8086
TWX: 910-443-3002
FAX: (206) 451-9556

Intel Corp.
408 N. Mullan Road
Suite 102
Spokane 99206
Tel: (509) 928-8086
FAX: (509) 928-9467

WISCONSIN

Intel Corp.
330 S. Executive Dr.
Suite 102
Brookfield 53005
Tel: (414) 784-8087
FAX: (414) 796-2115

CANADA

BRITISH COLUMBIA

Intel Semiconductor of
Canada, Ltd.
4585 Canada Way
Suite 202
Burnaby V5G 4L6
Tel: (604) 298-0387
FAX: (604) 298-8234

ONTARIO

†Intel Semiconductor of
Canada, Ltd.
2650 Queensview Drive
Suite 250
Ottawa K2B 8H6
Tel: (613) 829-9714
FAX: (613) 820-5936

†Intel Semiconductor of
Canada, Ltd.
190 Attwell Drive
Suite 500
Rexdale M9W 6H8
Tel: (416) 675-2105
FAX: (416) 675-2438

QUEBEC

†Intel Semiconductor of
Canada, Ltd.
1 Rue Holiday
Suite 115
Tour East
Pt. Claire H9R 5N3
Tel: (514) 694-9130
FAX: 514-694-0064

†Sales and Service Office
*Field Application Location

UNITED STATES, Intel Corporation
3065 Bowers Ave., Santa Clara, CA 95051
Tel: (408) 765-8080

JAPAN, Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi, Ibaraki, 300-26
Tel: 0298-47-8511

FRANCE, Intel Corporation S.A.R.L.
1, Rue Edison, BP 303, 78054 Saint-Quentin-en-Yvelines Cedex
Tel: (33) (1) 30 57 70 00

UNITED KINGDOM, Intel Corporation (U.K.) Ltd.
Pipers Way, Swindon, Wiltshire, England SN3 1RJ
Tel: (44) (0793) 696000

WEST GERMANY, Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen
Tel: (49) 089/90992-0

HONG KONG, Intel Semiconductor Ltd.
10/F East Tower, Bond Center, Queensway, Central
Tel: (852) 844-4555

CANADA, Intel Semiconductor of Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8
Tel: (416) 675-2105

Printed in U.S.A./M346/0990/3K/HP DM

CG/081090