REVISIONS			
APPROVED	DATE	CHANGE NOTICE	REV
J.A.	5/9/85	PCN PDC-004	P1
J.A.	11/12/85	PCN PDC-102	P2

****
* *
* PRELIMINARY *
* *
****

ICM-32xx MiniBus Specification

WRITTEN BY: Jim Anderson

NATIONAL SEMICONDUCTOR CORPORATION
15201 GREENBRIER PARKWAY
BEAVERTON, OREGON 97006

1. Introduction

MiniBus is a high performance, synchronous 16 bit bus intended to provide the board interconnect required for systems and/or subsystems which do not require the bandwidth (and associated cost and power) provided in the modern 32 bit busses which are currently under development.

MiniBus is appropriate for use as a backplane interconnect for systems comprised of multiple 8 and/or 16 bit processors, or as an I/O bus for larger systems. When operated at a reduced clock rate MiniBus may be extended to moderate lengths through the use of a 64-conductor DIN cable.

MiniBus does not compete with any of the 32-bit busses in varied stages of market introduction (BI, Multibus II, VME, Futurebus, NUbus, etc.) which are aimed at satisfying the bandwidth requirements of multiple 32-bit processors. It is intended provide a low-cost, low-power, low real estate bus with similar multi-processor system characteristics on a smaller scale (i.e. 16-bits of data, 8 masters).

Although a 32 bit bus is required to support 32-bit processors with common memory resources, the width is at best awkward for 8 or 16 bit units. Virtually all the LSI peripheral chips to support LAN's, Disks, terminals, floppies, tapes, GPIB, and other system requirements are designed in 8 or 16 bit widths, and are cheaper and more convenient to implement on a sixteen bit bus where bandwidth permits. Many of the new 32-bit busses recognize this and attempt to provide some mechanism of attachment in less than the full width.

The reader should also refer to the MBIC Specification (426600020-000) for a complete description of the MiniBus Interface Chip.

1.1. Features

The general characteristics of MiniBus are summarized below:

- Complete bus interface in a single CMOS device
- Multiplexed Address/Data

  Data Width 16 bits

  Local Address space 12 MB

  Bus Address space 4 MB (22 bits)

  Physical Addressing of all major components
- Synchronous Operation (125 ns Clock)
- Full handshaking, parity generated/checked
- Asynchronous I/O cycles for ease of implementation of low performance peripheral units

  256 Eight Bit Ports

  16384 Sixteen Bit Ports
- Bus bandwidth:

	Read	Write
Single Access	4.0 MB/sec	5.3 MB/sec
Burst (16 bytes)	11.6 MB/sec	12.8 MB/sec

- Maximum of 8 bus masters

  Distributed Arbitration in single bus clock (125ns)

  Fixed or Fair priority

  Interlocked operations supported

- Interrupts

  Four (4) addressable interrupts/master

  Eight (8) bus interrupt lines

- Support for dual ported memory operations

  Quasi dual-ported without external circuitry

  Full dual ported with external circuitry

1.2. MiniBus Interface (Master)

MiniBus is specifically designed to provide a simplistic LSI interface through the MiniBus Interface Chip (MBIC) which minimizes the cost of interface of an intelligent unit to a modern, sophisticated bus. The MBIC provides all necessary logic associated with the bus, and will connect directly to the NS32016 CPU. An intelligent unit may be connected to the MiniBus with exactly four (4) components, CPU, Timing Control Unit (TCU), oscillator, and MBIC as shown in Figure 1.2.

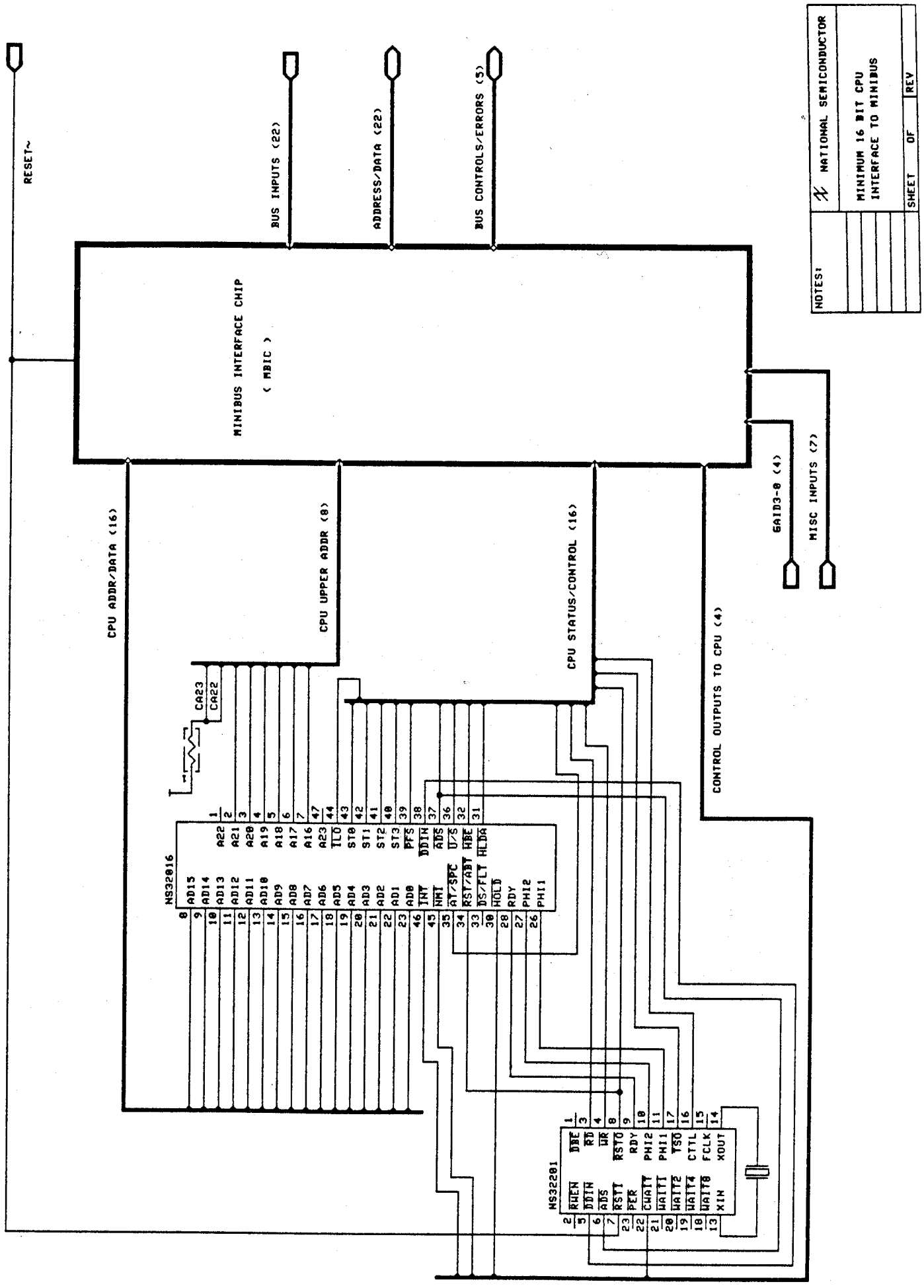
The MBIC provides all synchronization between the local CPU clock and the MiniBus clock, arbitration, parity on the transfers, interrupt circuitry, etc. to allow the circuit shown to operate as a MiniBus master and slave (for MBIC registers only). Although the circuit shown is not recommended (i.e. local memory would significantly improve performance), it is presented here to emphasize the ease with which MiniBus masters may be constructed.

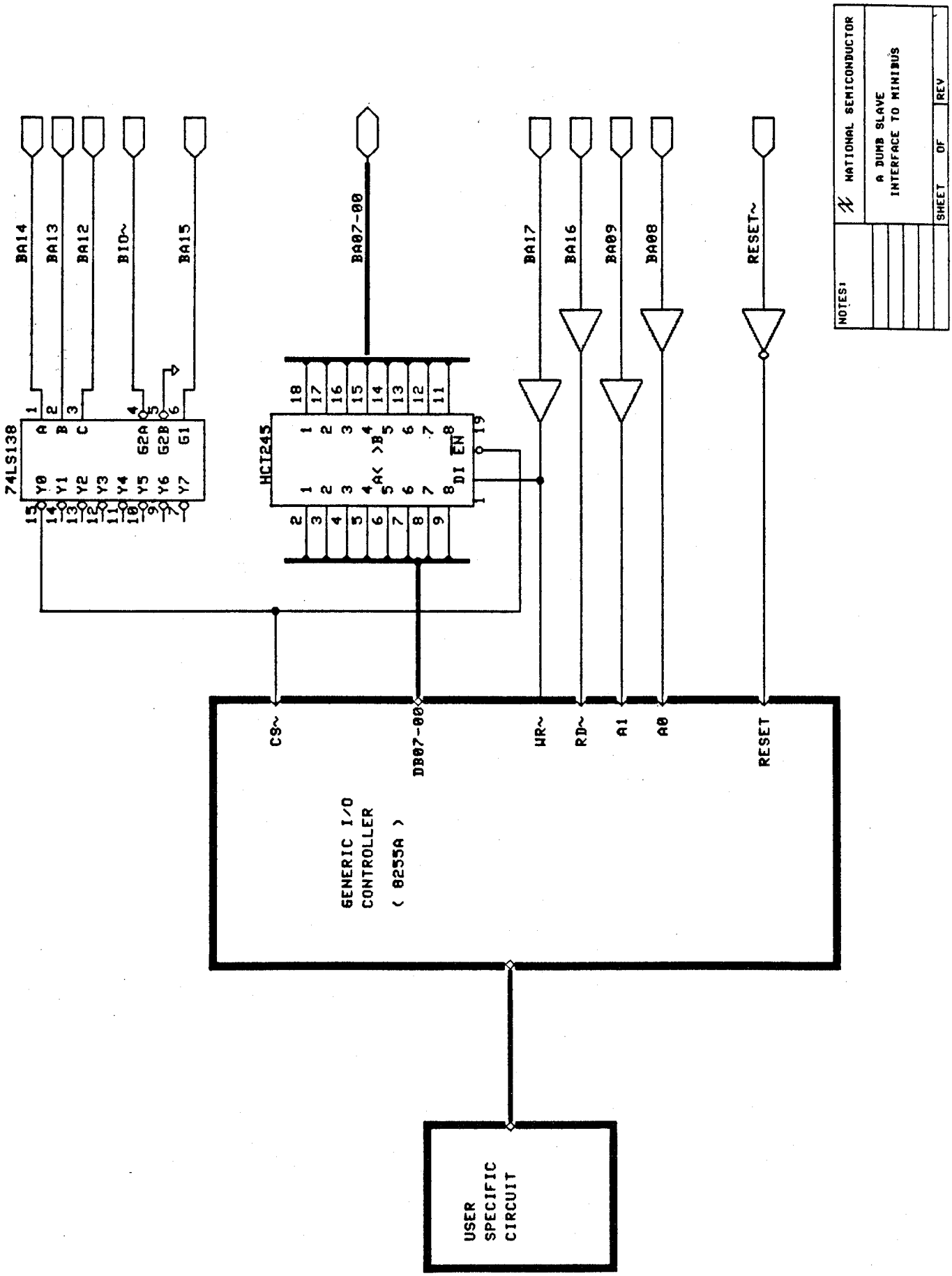
1.3. MiniBus Interface (Slave)

A unique MiniBus feature is the capability to "weld" together a series of bus clocks and simulate a simple asynchronous cycle for direct control of "dumb" I/O units. A diagram showing the necessary circuitry to connect a standard 8-bit peripheral chip to the MiniBus is shown in Figure 1.3.

This type of slave interface is obviously not suitable for high-performance devices, since it must be directly controlled by a CPU. It is presented to demonstrate the capability of MiniBus to maintain the cost of interface at a level which is appropriate to the need, rather than requiring interface circuitry whose cost dominates the cost of the unit.

Compare this interface with what would be required to attach the same device to Multibus II or VME.


RESET~
BUS INPUTS (22)
ADDRESS/DATA (22)
BUS CONTROLS/ERRORS (5)
MINIBUS INTERFACE CHIP
(MBIC)
CPU ADDR/DATA (16)
CPU UPPER ADDR (8)
CPU STATUS/CONTROL (16)
GAID3-0 (4)
MISC INPUTS (7)
CONTROL OUTPUTS TO CPU (4)
NOTES:
NATIONAL SEMICONDUCTOR
MINIMUM 16 BIT CPU
INTERFACE TO MINIBUS
SHEET OF REV
NS32016
NS32201


NATIONAL SEMICONDUCTOR
A DUMB SLAVE
INTERFACE TO MINIBUS
NOTES:
SHEET OF REV
BA14
BA13
BA12
BIO~
BA15
BA07-00
BA17
BA16
BA09
BA08
RESET~
74LS138
HCT245
CS~
DB07-00
WR~
RD~
A1
A0
RESET
GENERIC I/O
CONTROLLER
(8255A)
USER
SPECIFIC
CIRCUIT

2. MiniBus Signals

2.1. Pin Assignments

MiniBus requires fifty-one (51) signal lines and is implemented on a 96-pin DIN connector. A forty-six (46) pin subset of MiniBus is available on the A and C connectors, which makes it possible to extend the MiniBus with a 64-pin DIN cable and operate it at a reduced clock rate. The MiniBus signal mnemonics and their pin assignments are shown in Table 2.1 below.

2.2. Signal Descriptions

The MiniBus signals are summarized in Table 2.2. Each line is described briefly in the following subsections.

MiniBus Pin Assignments (J1)

c01 - gnd	b01 - +12v	a01 - gnd
c02 - BUSPAR	b02 - +12v	a02 - BUSERR~
c03 - AD20	b03 - +12v	a03 - IO~
c04 - AD19	b04 - rsvd(+15v)	a04 - AD21
c05 - BCOD0~	b05 - gnd	a05 - BCOD1~
c06 - gnd	b06 - gnd	a06 - gnd
c07 - AD16	b07 - +5v	a07 - AD18
c08 - AD17	b08 - +5v	a08 - AD02
c09 - AD03	b09 - +5v	a09 - AD00
c10 - AD01	b10 - gnd	a10 - AD04
c11 - gnd	b11 - gnd	a11 - gnd
c12 - AD05	b12 - +5v	a12 - AD06
c13 - AD07	b13 - +5v	a13 - RESET~
c14 - BREQ0~	b14 - +5v	a14 - BREQ1~
c15 - BREQ2~	b15 - HOLD~	a15 - BREQ3~
c16 - BREQ4~	b16 - reserved	a16 - BREQ5~
c17 - gnd	b17 - reserved	a17 - BREQ7~
c18 - BCLK	b18 - gnd	a18 - gnd
c19 - gnd	b19 - gnd	a19 - gnd
c20 - MBICCLK	b20 - gnd	a20 - gnd
c21 - gnd	b21 - gnd	a21 - gnd
c22 - BCLK/2	b22 - INT07	a22 - gnd
c23 - BREQ6~	b23 - INT06	a23 - PFAIL~
c24 - RFRSHTIM	b24 - INT05	a24 - PWAIT~
c25 - INT03	b25 - INT04	a25 - INT02
c26 - INT01	b26 - +5v	a26 - INT00
c27 - gnd	b27 - +5v	a27 - gnd
c28 - AD12	b28 - gnd	a28 - AD14
c29 - AD09	b29 - rsvd(-15v)	a29 - AD15
c30 - AD11	b30 - -12v	a30 - AD08
c31 - AD13	b31 - -12v	a31 - AD10
c32 - gnd	b32 - -12v	a32 - gnd

Table 2.1

MiniBus Pins

Mnemonic	Description
AD21 - AD00	Address/Data/Control
BCOD0~, BCOD1~	Bus Code
BUSERR~	Bus error
BUSPAR	Bus Parity
BREQ0~ - BREQ7~	Bus Requests
MBICCLK	MBIC Bus Clock
BCLK,BCLK_2	System Bus Clocks
INT00 - INT07	Interrupt Requests
IO~	I/O Transfer
PFAIL~	Power Fail
PWAIT~	Peripheral Wait
RESET~	Reset
RFRSHTIM~	Refresh Interval

Table 2.2

MiniBus Signals

Address/Data/Control (AD21 - AD00)

These 21 lines provide bus address space of 4 MB, 16-bit wide data paths, status fields, and miscellaneous transfer controls.

Bus Code (BCOD0~, BCOD1~)

The Bus Code lines combine with IO~ to determine the current state of the bus.

I/O Transfer (IO~)

The I/O Transfer signal is used to access peripheral devices using I/O space addresses and redefines the interpretation of the remaining data segment lines.

Peripheral Wait (PWAIT~)

The Peripheral Wait signal can be issued by any "dumb" peripheral unit which cannot satisfy the master's access requirements in the specified amount of time. This signal causes the master to wait for the duration of the asserted signal, delaying the end of the master's read or write signal until the unit is ready to respond. When the unit is ready to proceed with the ongoing transfer, it removes the PWAIT~ signal. The master then concludes the I/O cycle in the normal manner.

Bus Parity (BUSPAR)

The BUSPAR line is utilized to add parity to the information present on the

bus during synchronous transactions. The scheme utilizes ODD parity. During the address cycle it covers bits AD21 through AD00 while during the data cycle it spans bits AD15 through AD00. The information carried on this line should be interpreted only during the synchronous address and data cycles.

It is driven by the master during the address cycle, the data cycle for a write, any asynchronous I/O cycle and while the bus is idle. It is driven by the slave during the data cycle for of a read.

Bus Error (BUSERR~)

The BUSERR~ line, when asserted (low) by the previous slave, indicates that one of the following conditions was detected during the previous synchronous transaction:

- An address parity error was detected by the slave after checking parity on the address previously identified as being its own.
- A data parity error was detected by the slave during the data portion of a write operation.

The BUSERR~ line is driven by the slave the cycle after a transfer is complete, that is, the cycle after ceasing being slave. By default it is driven by the master to its non-asserted (high) condition a cycle after becoming bus master until the cycle after dropping bus mastership except when the slave owns control of the line.

MBIC Bus Clock (MBICCLK)

The MBIC Bus Clock is synchronized to and has the same frequency as BCLK. It is used only by the MiniBus MBICs to derive their bus timing. All other system elements should utilize BCLK.

Bus Requests (BREQ0~ - BREQ7~)

Each of the eight Bus Request lines is driven by one and only one potential bus master, and is used for bus arbitration and bus conflict resolution.

Interrupt Requests (INT00 - INT07)

Eight separate Interrupt Request lines allow peripheral units to interrupt any of the MiniBus CPUs.

Any MiniBus unit capable of driving interrupts can have the appropriate signals strapped to one or more of these lines.

Power Fail (PFAIL~)

The Power Fail signal indicates that a Power Failure is imminent. Usually this signal is the highest priority interrupt into any system CPU, activating an interrupt service routine to prepare the system for power failure by placing valuable data in non-volatile storage.

Reset (RESET~)

The Reset signal places all system units into a known initial state.

System Bus Clocks (BCLK_2, BCLK)

The System Bus Clocks control the system timing and are generated by one and only one unit in the system.

Refresh Interval (RFRSHTIM~)

The RFRSHTIM~ signal is activated for a single bus clock period approximately every 15 usec (i.e. a period suitable for refresh of dynamic RAM devices).

3. MiniBus Address Space

Twenty-two address lines exist on the MiniBus backplane, allowing a bus addressing range of four (4) million bytes. The bulk of this address space is presumed to be memory, but the upper 128 KB is divided into 4 distinct address spaces for system purposes.

00 0000 -> 3D FFFF Memory Space

3E 0000 -> 3E FFFF Eight bit I/O space (256 ports)

This 64 KB of the bus address space will not appear on the bus, but are instead translated into asynchronous I/O cycles. AD15-08 will contain the port address, while AD07-00 will be configured as an eight bit data path in the direction appropriate for the current I/O operation. Since address and data are present on the bus in parallel during eight bit I/O cycles, only 256 distinct ports are available.

3F 0000 -> 3F 7FFF Sixteen bit I/O space (16384 ports)

This 32 KB of the bus address space will not appear on the bus but are instead translated to asynchronous extended I/O cycles. AD14-A00 may be used to select the appropriate I/O element for the sixteen bit transfer.

3F 8000 -> 3F BFFF Physical Address Space

This 16 KB of the bus address space is reserved for physical addressing of the units which implement the full MiniBus interface. Bits AD13-AD12 are ignored,

providing a physical space of 256 bytes for each interface. Transfers within this address space have identical bus characteristics as memory transfers, but the reference is to control registers within the interface element itself.

Each MiniBus interface has a unique identity which must be strapped on the card itself. Four bits of identity allow for eight masters (0-7) and seven non-masters (8-E). THE MASTER's IDENTITY MUST CORRESPOND TO THE BREQ[i]~ SIGNAL BY WHICH THAT MASTER OBTAINS THE BUS.

AD11-08 are used during access to physical address space to determine the board identity of the desired interface. An "F" in AD11-07 may not appear on the bus, and is instead used within a processor board as a fixed local interface address alias for the convenience of software.

AD07-00 are used during access to physical address space to select the individual register/element within the interface logic.

Each MiniBus interface is required to implement the following elements within it's physical address space:

AD07-00
---

00 - Interrupt Register
04 - Control and Status Register 0

These registers must be capable of being written as two individual bytes to allow access from 8-bit processors.

Details of these implementations will be provided in later sections.

3F C000 -> 3F FFFF Non-existent Address Space

The final 16 KB will never appear as a MiniBus address. Normal CPU implementation on MiniBus would be expected to map the 4 MB of MiniBus address space into the upper 4 MB of a sixteen megabyte (or larger) processor addressing range, reserving the lower portion of that range for the locally implemented memory. Many of the current micro-processors use the upper regions of the memory space for specialized cycles (i.e. INT/ACK, etc.) and this region is left uninterpreted to avoid potential conflicts in that area.

4. Bus Overview

MiniBus provides to the user a very comprehensive bus with an extensive set of features found to date only in very complex bus architectures, while retaining the ease of interface found in much simpler and less capable ones.

The operation of MiniBus can logically be divided into several signal groups which contribute to the overall functionality of the bus:

Data Segment -

Information is transferred on MiniBus through the use of 27 signal lines which provide all necessary controls, address, and data and error checking/reporting.

Arbitration Segment -

The arbitration segment of MiniBus implements a distributed arbitration on 8 backplane wires. The scheme provides for system units to arbitrate in a "fairness" or in a straight priority manner, dependent on the implementation, with both schemes capable of coexisting in the same system. The bus has a "default master" at all times.

Interrupt Segment -

MiniBus provides for 8 conventional hardware interrupt lines on the backplane which may be serviced by any processor in the system.

System Control Segment -

This section of the bus provides miscellaneous signals required for system operation, such as the bus clocks, reset signal, power fail indication, etc.

MiniBus operates in a somewhat unconventional manner since the drive capability of CMOS devices prohibits the use of other than very light termination resistors. MiniBus ALWAYS HAS A MASTER DRIVING ALL CRITICAL LINES, although the master may simply be driving the bus to its "idle" state. During the power-up/reset CPU number 7 becomes the initial bus master, and will retain the bus until the bus is requested by another master through the normal arbitration procedure. Rather than simply releasing the bus lines when the master has no immediate need for the bus, the MiniBus master "parks" on the bus until another master is requesting and a controlled transfer of bus ownership can be performed.

A dual purpose exists for this strategy -

1) Light termination does not provide the rise time necessary to recover the signals in a single bus clock interval, but a driver driving them to the 1 (one) state has can easily accomplish this.

2) By retaining the bus until it is specifically requested by another master, the master which has most recently used the bus suffers no arbitration penalty. This can be significant in systems with a predominant bus master (such as a single CPU), or where I/O units operate with bursts of activity.

5. Information Transfer Overview

Information Transfer on MiniBus requires twenty-seven (27) signal lines:

AD21-00	Address or Status/Data
BUSPAR	Parity of Address/Data
BUSERR~	Bus Error line
IO~,BCOD1~,BCOD0~	Bus State Lines

MiniBus provides for two distinct transfer disciplines:

Synchronous Transfers -

These transfers occur between units within the MiniBus community which implement the full bus discipline. Since MiniBus is a sophisticated bus, it is impractical to implement the full discipline without the use of a semi-custom LSI component such as a Gate Array or Standard Cell device. This interface device is referred to in the following text as MBIC (MiniBus Interface Chip).

Asynchronous Transfers -

To achieve desired simplicity of interface of non-intelligent peripherals MiniBus allows for attachment of units which are directly controlled from a processor through a simple asynchronous bus cycle and dedicated hardware interrupt lines. The asynchronous transfer is actually generated by the MBIC as a "welded" set of bus clocks, but appears to the peripheral as an asynchronous sequence.

To achieve this duality of cycles with appropriate handshaking between units, and to minimize the number of bus lines required so that the entire transfer discipline can be implemented in a single LSI package, MiniBus time-shares the upper six address bits using them for address information during a synchronous address transfer, re-defining them as handshake and control functions during synchronous data transfers and asynchronous operations.

The state of the bus in any given bus clock can be determined by the state of three lines (IO~, BCOD1~, BCOD0~) as follows:

IO~	BCOD1~	BCOD0~	
0	x	x	Asynchronous Eight Bit I/O Cycle

The MBIC will drive these lines to this state for seven (7) consecutive bus clocks while the upper address lines are manipulated to simulate a single asynchronous I/O cycle.

IO~	BCOD1~	BCOD0~	
1	0	0	Asynchronous Sixteen Bit I/O Cycle

The MBIC will drive these lines

to this state for nine (9) consecutive bus clocks while the upper address lines are manipulated to simulate a muxed address/data I/O cycle.

IO~ BCOD1~ BCOD0~
1 0 1 Synchronous Address Clock

The MBIC will drive these lines to this state for a single bus clock as the first clock of any synchronous transfer sequence. During this clock, AD21-00 will contain valid address information to select the desired synchronous slave. All synchronous slaves must be capable of recognizing an address transfer when it occurs, and latching the information from the bus for use in the transfer sequence.

IO~ BCOD1~ BCOD0~
1 1 0 Data/Idle Clock

The MBIC which is the current bus master will drive these lines to this state during idle cycles. While a synchronous transfer is in progress, the master MBIC will drive IO~ and BCOD1~, and the selected slave will drive BCOD0~.

When the bus is in this state, the master MBIC is driving its status on AD21-19, and the slave MBIC is driving its status on AD18-16. If no slave is selected (idle cycle) the master MBIC will also drive AD18-16 to the appropriate status.

The number of data clocks occurring in a synchronous transfer sequence is determined by the transfer type and response of the units involved.

IO~ BCOD1~ BCOD0~
1 1 1 No-response Clock

This bus state will only occur immediately following an address clock. During the address clock the master MBIC drives BCOD0~ to "1", and the light pullup on that line will maintain the "1" in the absence of a slave responding by driving BCOD0~ to "0" as it becomes selected and the bus enters the data state.

The current master will recognize the no-response state and the bus will revert to the

idle state in the next clock period as the master unit takes appropriate action to report the error.

A complete summary of the control lines involved in both synchronous and asynchronous transfers is given in Table 5.1

	IO~	BCOD 1~	0~	A21 <--	A20 mstr	A19 ->	A18 <--	A17 slv	A16 -->
Address Clock -	1	0	1	<------	Address	------>			
Data/Idle Clock -	1	1	0						
Idle Clock - (parked mstr drives)									
No arbitration	1	1	0	1	0	0	0	0	0
Arbitration	1	1	0	0	1	1	0	1	1
Data Clock - (master drives)									
Master Data Status -									
Master not ready	1	1		0	0	0			
Ready,odd byte,arb	1	1		h	0	1			
Ready,even byte,arb	1	1		h	1	0			
Ready,word,arb	1	1		h	1	1			
Ready,word,burst	1	1		1	0	0			
				(h) -> no arbitration					
Slave Response - (slave drives)									
wait			0				0	0	0
data next			0				0	0	1
data now			0				0	1	0
local parity err			0				0	1	1
(reserved)			0				1	0	0
(reserved)			0				1	0	1
data now-burst ok			0				1	1	0
retry			0				1	1	1
No-response Clock -	1	1	1	x	x	x	?	?	?
I/O cycle -	0	x	x						
Write				1	even	odd	x	wr~	1
Read				1	even	odd	x	1	rd~
Extended I/O	1	0	0						
Write				1	even	odd	ad~	wr~	1
Read				1	even	odd	ad~	1	rd~

Table 5.1 Transfer Control and Status fields

Master Status

The master MBIC can "stall" the transfer by transmitting status 000 if the data is not yet available. All other status codes imply the master has placed the appropriate data on the bus if a "write" is in progress, or is prepared

to receive the data at the end of this data clock if a "read" is in progress.

Burst status implies a 16-bit data element, and must be accepted by the slave by the return of a slave status code of 110 to allow continuation of the burst to the next data element. All MBIC's which implement burst mode accesses must be able to break the burst sequence if the burst is denied by the slave, and re-initiate the remaining transfer with a new address transmission. A slave denial of burst may occur because the slave cannot handle burst operations, the maximum burst count acceptable to the slave has been reached, or the next address would fall outside the range to which that slave responds.

Slave Status

The slave may also stall the master with status 000 until it is able to proceed with the transfer. The optional "data next" response is by the slave promises that the next clock will be the final clock of this data element transfer. MBIC's are not required to issue or act upon "data next", since its function is actually that of a special form of "wait".

The slave's willingness to complete the transfer will normally be signified by the status code "data now" or "data now - burst ok" which indicates the slave has placed the data on the bus if a "read" is in progress, or will accept the data at the end of this clock if a "write" is in progress.

In exceptional cases, the slave may respond with "local parity error" which means the slave has placed the data on the bus as with "data now" but an error was indicated by the error circuits of the local memory when the data was accessed.

Slaves which have local dual ported memory may terminate the transfer with "retry" status, indicating that their local resources are temporarily unavailable. The master must drop the bus at this response, and re-initiate the entire transfer at a later time. (See discussion in MBIC Specification 426600020-000).

NO MBIC MAY ISSUE A "RETRY" STATUS IN RESPONSE TO A REFERENCE TO THE PHYSICAL ADDRESS SPACE ASSOCIATED WITH THAT MBIC.

6. Synchronous Transfers

MiniBus masters exchange information with slaves or other masters in a synchronous manner controlled by the central bus clock. (All MiniBus masters are capable of responding as slaves when addressed by the physical addressing mode provided on MiniBus.) The current bus master will place the information on the bus lines shortly after the rising edge of the bus clock, and the selected slave will sample that data at the next rising edge of the bus clock.

Responses from slaves which implement the full bus interface (i.e. other masters, memories, dma's etc) are returned to the master in a similarly synchronous manner. Provision is also made within the bus discipline to accommodate units which implement only a very primitive asynchronous interface to simplify the attachment of low performance I/O boards.

The synchronous nature of the bus transmission does not imply that transfers require fixed response. Handshaking signals between the master and slave may be used to satisfy

any reasonable response time (measured in integral numbers of the bus clock).

All non-I/O operations assume a full sixteen bit transfer on the bus, and all units designed to MiniBus assume that the transfer is aligned with the even numbered byte in the low portion (AD07-00) and the odd numbered byte in the high portion (AD15-08). Two byte enable control signals from the master status are available on the bus to distinguish between actual word operations and individual byte operations. All burst operations on the bus are defined to be word operations.

This first portion of this discussion will concentrate on the exchange mechanism between MiniBus units which have the full bus implementation.

The general format of a bus access is as follows:

The unit desiring to initiate a bus transfer (master) must acquire bus mastership if it is not already in control of the bus.

Clock 0

The master initiates the transfer with an address cycle by placing the appropriate information on the bus which indicates that the current cycle is an address transfer, the address involved, and the nature of the transfer (i.e. read or write).

All other units which have the potential of becoming slaves recognize that an address cycle is occurring and perform the appropriate address decode to recognize their address range. These units must be able to recognize their address at all times, latch the information required from the address cycle, and be prepared to reply to the master in the next bus clock. The actual reply may, however, simply acknowledge the selection and require the master to wait until the slave can begin to act on the request.

Clocks 1 thru N

The data section of the transfer begins in the clock immediately following the address cycle. During the entire data section of the transfer, the slave will drive the BCOD0~ line to inform the master that the connection has been established, and the slave status bits contain valid data. The master uses BCOD0~ to distinguish a "no-response" condition and take the appropriate action for an invalid address.

During the data section of the transfer, the upper address bits (AD21-16) are re-apportioned into two "data status" fields, one for the master, the other for the slave. The data field of the bus (AD15-00,parity) is driven by either the master or slave as appropriate for the read/write operation as requested in the

address cycle.

The master status field conveys to the slave when the master is ready, and whether it requires a word, high byte, low byte, or burst transfer, and further conveys to the entire MiniBus community whether the master is willing to give up the bus at the completion of the current transfer (i.e. allow arbitration)

The slave status field conveys to the master whether the slave is:

- ready to proceed
- capable of accepting bursts
- returning incorrect data
- unavailable due to local deadlock

(NOTE: All MiniBus Masters must be capable of immediately releasing the bus if the slave indicates it's unavailability (RETRY) and re-initiating the complete transfer when it can again gain bus mastership (See discussion under "Conflicts").

The effective result of any data clock on the bus is determined at the end of that clock by each of the connected units examining the received status in combination with the status that the unit was transmitting during that clock. If both units were transmitting a "ready" class of status, the current transaction is complete.

Clock N+1

During the bus clock following the actual data exchange, the slave unit will drive BERR~ if it detects a bus parity error on the data transfer of a write operation.

6.1. Normal Read Operation

Figure 6.1 shows the bus sequence which occurs during a normal read operation.

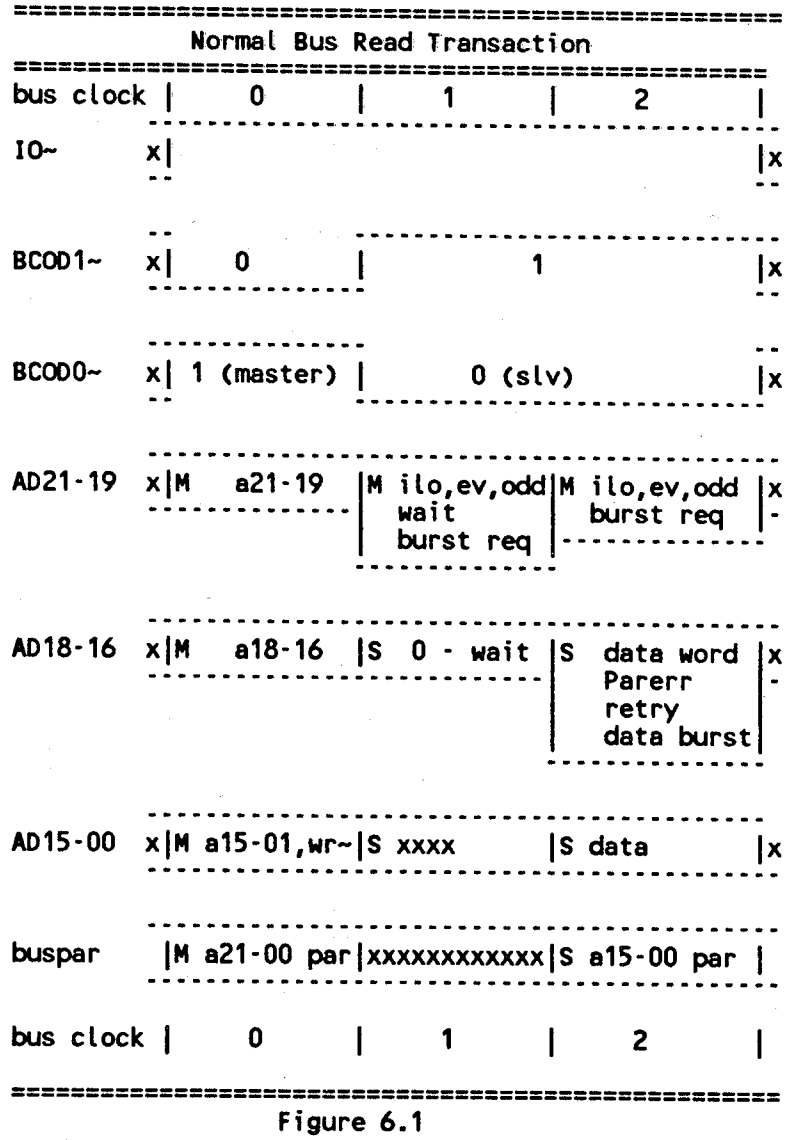


Figure 6.1

Normal MBIC Read Transaction

The address cycle is identified by the state of IO~, BCOD1~, BCOD0~ = 101 (clock 0). In clock 1, a slave which recognized its address during the address cycle would drive BCOD0~ to acknowledge selection. If no slave responds, the light pullup on this line will maintain it high and the master will recognize that the "no-response" condition is present.

(Note: It is acceptable for a slave unit to drive BCOD0~ to acknowledge the address and several clocks later to drive it back to high to reject the transfer. This situation may occur in units which must respond prior to checking the address transfer parity.)

The master will drive AD21-19 during clock 1 to present its status. "ilo" will inhibit

the arbitration procedure if the master requires successive bus cycles for an interlocked operation or burst transfer. "even" and "odd" are individual byte enables which would normally be ignored on a "read" transaction. Due to limitations in the early LSI components a master which is capable of requesting "burst" operations must also be capable of re-transmitting the next address if a burst denial is received from the slave.

The master will continue to drive these lines until he gets an appropriate "ready" from the slave to indicate the transfer has completed.

The slave will normally drive back a status of "wait" on AD18-16 during clock 1 on a read, since the requested data must be obtained before it can be placed on the bus. The slave may continue to drive a "wait" status during subsequent clocks as required to obtain the data.

When the slave has obtained the data and placed it on the bus (or determined an internal deadlock prohibits this) it will change the returning status to one of those shown in clock 2 in the diagram. Since both units are transmitting a "ready" status during clock 2, that clock is the last clock of the read transaction.

6.2. Normal Write Transaction

Figure 6.2 shows a normal MiniBus write transaction.

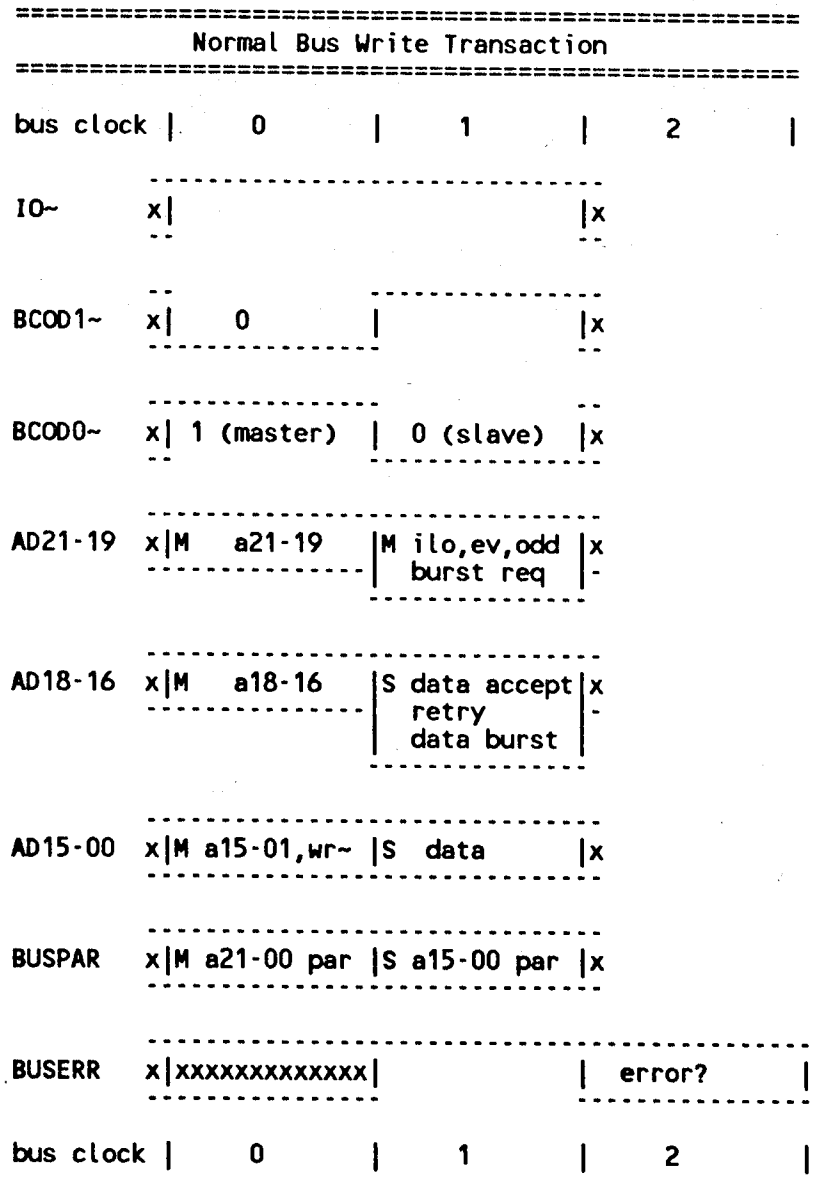


Figure 6.2

MiniBus Write Transfer

The address cycle of the write is identical to that of the read except for the state of the "wr~" signal transmitted in the least significant data position (AD00). This indication must be transmitted with the address to determine which of the units will drive the data lines on the bus in subsequent clocks.

In clock 1, the master would normally place the data on the bus along with the appropriate status as described above. During a write transfer, "even" and "odd" are necessary to distinguish a word, high byte, or low byte transfer.

The slave unit will drive BCOD0~ from clock 1 to the end of the transfer as before to acknowledge the selection process. The slave may also drive "wait" in the slave status field until it is able to proceed with the transfer. The master is required to maintain the data on the bus until it receives a slave response which terminates the transfer.

The BUSERR line will be driven by the slave during the clock following the transfer if a bus parity error was detected.

7. Dual Port Memory

At microprocessor clock rates of 8-10 MHz, the time available to access the memory precludes the use of the traditional single bus design without serious degradation of CPU performance (i.e. waits). Instruction processing is very "memory intensive" and in multiprocessor systems, as more than one CPU attempts to execute from a single memory system, access conflicts quickly become the dominant mode, negating the advantages of the independent processing units.

Each processor must be provided with some amount of local memory (implemented as either local RAM or Cache) which it can normally access without having to obtain the use of a common system resource (such as a system bus). If a significant percentage of the CPU's activity can be directed to this "private" memory resource, although both access degradation and conflicts will still be present when that CPU must access system resources (I/O or global memory), the lower frequency of system bus accesses reduces the impact on system performance to an acceptable level.

The implementation of "private" memory presents some major system problems:

1) There must exist at least one global memory board in the system accessible to all bus masters in order for information to be exchanged.

2) All DMA type I/O traffic must be transferred to this global memory, then transferred again to the "private" memory for which it is intended.

These problems are solved if the "private" memory is implemented as a "local" memory, where it is resident within the CPU complex, but can be accessed from external masters through a Dual Port mechanism. The local CPU then enjoys a de-facto preferred access to the local memory, but it is accessible to other masters as required.

To implement true dual porting, external circuitry must be added outside a microprocessor to allow forcing the device off the address and data lines to the local memory. Microprocessors will cooperate and Tri-State the lines (via the HOLD mechanism), but this will only be honored between CPU memory cycles. In order for one CPU to access the local memory of another, it must control its internal bus, the system bus, and the internal bus of the other CPU. If two CPUs attempt to reference each other simultaneously, a stalemate occurs since neither can give up its internal bus through the HOLD mechanism without completing the access. It is the resolution of this deadlock in an orderly fashion which requires external parts to force the CPU off the lines without its cooperation.

The MiniBus processor card interface will support true dual porting if the designer of the CPU card provides the additional circuitry, but will also implement a limited subset of true dual porting without external components which is sufficient for smaller systems. This is referred to as quasi dual ported memory, and we will examine the limitations of its applicability.

Assumptions:

1) All MiniBus bus masters will be able to back off from a system bus access and retry the transfer. This implies that no address, data or control information be discarded until the transaction is complete.

2) MiniBus processor card interfaces will attempt to gain access to their internal

bus through a HOLD signal and the corresponding HOLD/ACK when an incoming transfer is received.

3) MiniBus processor cards will detect the occurrence of a stalemate (i.e. outgoing request and incoming request at the same time) and will resolve it by the full dual-port mechanism if available, otherwise beginning a retry sequence on the bus.

4) The retry sequence on the bus will allow all but the true conflicts to be resolved (i.e. if the receiving CPU is trying to access system memory and not the other CPU's memory, it will resolve properly simply by one CPU relinquishing the bus to the other, allowing completion). In the event of a true deadlock, the retry sequence will identify that state, and abort the highest priority CPU actively engaged in the deadlock via a Non-Maskable Interrupt (NMI).

The system implications of this quasi dual porting are as follows:

1) Boards which can support true dual porting can coexist with quasi dual ported boards and will never cause nor suffer from the NMI to release a deadlock.

2) Boards which contain no system-accessible memory on board (DMA masters or CPUs which have declared their memory as "private" rather than "local") can never cause nor suffer from the NMI to release a deadlock.

3) The lowest priority CPU (among the set of CPUs with quasi dual ported "local" memory) can cause, but not suffer from the NMI.

In practice, this means that single CPU systems with multiple DMA units will work just as well as if fully dual ported. If a second CPU is added (and both are quasi dual ported) only one of the CPUs may be programmed to directly reference the other's local memory. DMA's may reference both at will. It is possible to develop software strategies to recover from the NMI, and thus allow both to reference each other, but none of those which come to mind are particularly "clean" and thus must be considered as limited to specific applications.

For a complete explanation of the RETRY procedure, please refer to MBIC specification (426600020-000).

8. Asynchronous Bus Cycles

MiniBus is a sophisticated, modern bus structure which of necessity requires a fairly complex discipline to accomplish the objectives of a high-performance multi-master system. Many potential members of the MiniBus community do not require the performance or complexity of the full bus capability.

Recognizing this need, MiniBus includes an I/O mode of operation which allows "dumb" slaves to participate in the bus family. This mode of operation essentially "welds" together a series of bus clocks and simulates a simple asynchronous handshake equivalent

to the "IN" and "OUT" instructions of popular 8-bit micro-processors. To attach a peripheral board which operates in this mode, one simply needs address decode and transceiver control circuitry.

8.1. 8-Bit I/O Sequence

Figure 8.1 shows the I/O sequence of MBIC.

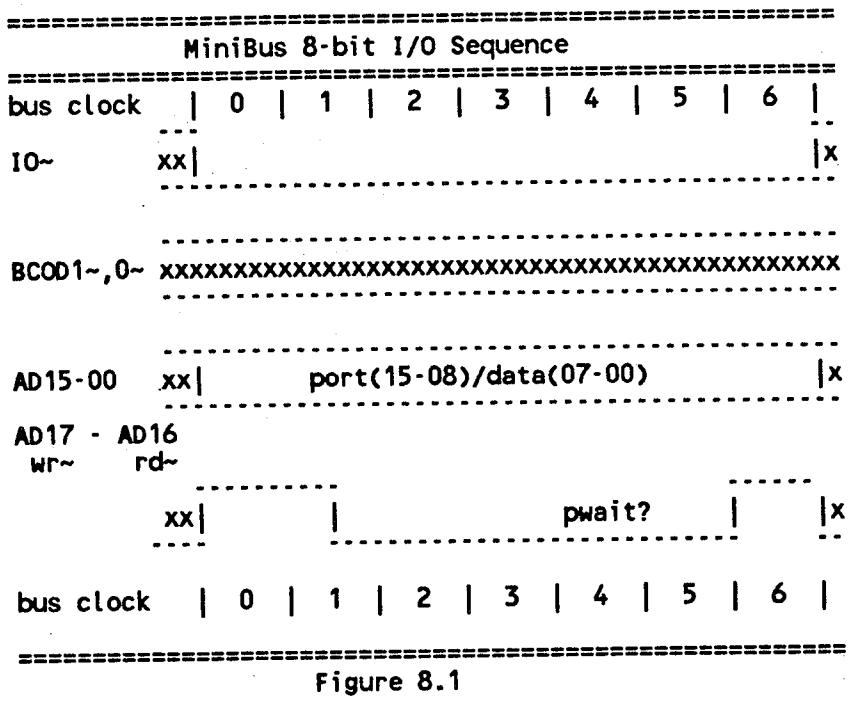


Figure 8.1

Eight Bit I/O Transaction

The master identifies this bus cycle by driving the IO~ line low and placing the port address on AD15-08, and drives the data on AD07-00 for a write operation. After allowing 1.5 bus clocks of setup time, the master activates AD17 if a write operation is required, or AD16 if a read operation is required. The slave board need only connect to AD15-00, IO~, and AD17 (interpreted as wr*) and AD16 (interpreted as rd~) to attach to MiniBus in this mode.

On a read transaction, the master will sample the data 4.5 bus clocks later, de-activate AD16, and maintain the address and IO~ line stable for another bus clock before the cycle is complete. On a write cycle, an identical window of control exists around the activation of AD17.

If the slave requires additional time to respond, he may attach to the "pwait" line and activate it as required to extend the rd or wr strobe time.

No bus arbitration is permitted during an I/O operation.

8.2. 16-Bit I/O Transfers

The standard I/O transfer mechanism described above is limited to byte transfers since it is derived from the 8-bit world. To provide a capability which allows 16-bit peripherals to attach in a similar manner, MiniBus also has an extended I/O cycle defined.

The extended I/O cycle first muxes the address out in a two clock period as shown below in Figure 8.2, and then proceeds thru the same timing sequence for the data portion as was performed for the normal IO.

The basic difference between the eight and sixteen bit I/O mechanisms is the use of AD18 (interpreted as ADS~) during the prefixed 2 clocks, and the configuration of the bus AD15-00 lines as 16-bit data during the data segment of the transfer. The IO~ line is not active during the extended I/O cycle to prevent 8-bit boards from mis-interpreting the transfer.

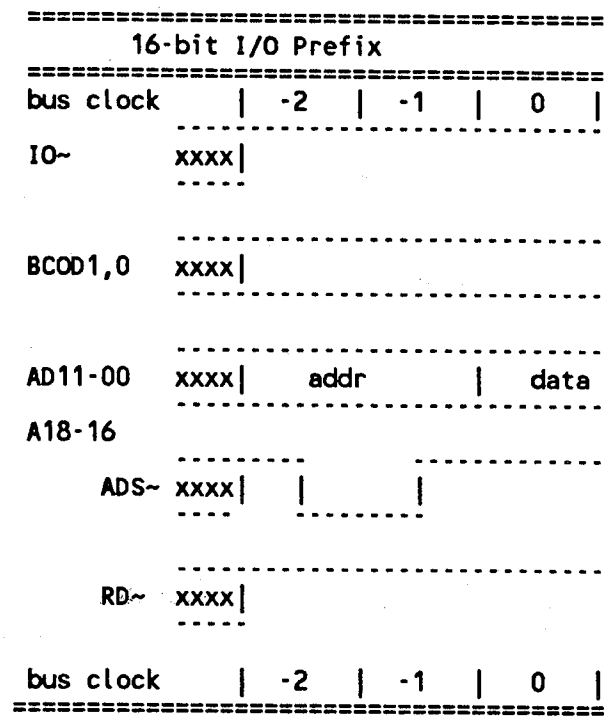


Figure 8.2

Sixteen-bit I/O Transfer

9. Arbitration

MiniBus allows for up to eight (8) system masters which can consist of any mixture of processor cards and intelligent peripheral units with DMA capability. The arbitration segment is implemented using only eight backplane wires (BREQ0~-BREQ7~). Each master is assigned a priority line on the backplane which corresponds to its board identification number. No centralized unit exists to control the arbitration, the logic is distributed among the potential bus masters.

MiniBus provides for the existence of a default master at all times. The current bus master does not release control of the bus until it is notified by another unit that the latter requires access to it. It then does so whenever it is appropriate. (A CPU would probably release the bus immediately, while a DMA unit which is emptying a FIFO may delay the release until the burst transfer is complete). In any case, to minimize the possibility of causing overruns, CONTROL OF THE BUS SHOULD NOT BE RETAINED FOR PERIODS OVER 10 MICROSECONDS while any requests are outstanding.

Not relinquishing the bus until another external bus request is detected allows the bus owner to avoid the overhead required to arbitrate for the bus before every bus transfer. This is particularly useful in the case of a system with only one CPU.

Two priority schemes are available which can coexist on the same system: Fixed and Fair Priority.

Fixed Priority -

Using this scheme, the system unit asserts its request whenever it requires access to the bus. Upon noticing other requests, it releases the bus at the appropriate time and can immediately request control of the bus again if it needs to. This scheme is usually implemented in I/O units with high speed bus access requirements.

Fair Priority -

When implementing this scheme, upon noticing another unit's request for the bus, the master releases it as appropriate. The unit will then only reassert its request when ALL other outstanding bus requests (including those of lower priority units) have been satisfied. This results in a "round robin" priority scheme, usually implemented in processor cards, whereby system units alternate in the use of the bus.

The arbitration lines on MiniBus are used for different purposes during bus conflicts. All units not involved in the resolution process are not permitted to arbitrate for the bus until the recovery is complete. The conflicting units utilize the arbitration lines to inform each other of their identity and ultimately to arbitrate in case of an irresolvable deadlock. No conventional arbitration for bus ownership takes place during a bus conflict, control alternates between the players involved in a specified manner.

9.1. Arbitration Procedure

The eight signals on MiniBus pertaining to the arbitration segment are:

BREQ0~ - BREQ7~

where 0 has the highest priority and is normally assigned to a DMA unit and 7 has the lowest priority and is usually reserved for the system master CPU. SINCE MiniBus REQUIRES A MASTER AT ALL TIMES, EACH SYSTEM MUST CONTAIN A UNIT ON BREQ7~ WHICH IS THE POWER-UP MASTER.

The conventional arbitration sequence (i.e. no bus conflict) is to obtain control of the MiniBus for information transfer is as follows:

Upon detection of an internal request for MiniBus , if the unit is not currently the bus master or in a "fair request" state, it will assert its BREQ~ signal synchronously with BCLK.

The current bus master always has control of the arbitration procedure. Arbitration will occur in a bus clock only if:

1) The IO~,BCOD1~,BCOD0~ lines

indicate a data-idle cycle (110). These lines may all be driven by the master if it is "parked" on the bus or BCOD0~ may be driven by a slave if a transaction is currently in process.

2) The Master Status Field (AD21-19) indicates the master's willingness to end the transfer (i.e. status 001,010, or 011).

3) The Slave Status Field (AD18-16) indicates the slave's willingness to end the transfer (i.e. status 001,011,1xx).

If the master is "parked" on the bus, the master will drive both the master and slave status fields to values which allow/disallow arbitration as required.

When the above conditions are met, each unit with an outstanding bus request examines the BREQ~ lines to determine if a higher priority unit is also requesting. The highest priority unit will acquire the bus at the rising edge of the bus clock.

Upon obtaining the bus, the unit drops its BREQ for the duration of its mastership. The unit will then proceed to perform its transfers over the bus and will retain control of the bus allowing arbitration at appropriate points, until another unit requests and wins the bus.

10. Interrupts

MiniBus provides for two distinct interrupt mechanisms:

1) Hardware Interrupts

Eight (8) dedicated lines on the MiniBus are reserved for hardware interrupts, INT00 (highest priority) thru INT07 (lowest priority). These are intended to allow "dumb" I/O boards to request processor attention in the simplest manner possible.

A peripheral desiring to interrupt the processor simply drives the appropriate interrupt line high. No "Interrupt Acknowledge" procedure is defined as a part of this bus specification, it is presumed that the requesting peripheral will continue driving the INT line until the appropriate service has been received form the processor through the use of I/O transfers to/from the requesting device.

2) Intelligent Interrupts

For multiple processors to operate in a

fairly cohesive manner, it is necessary that they have the capability to interrupt each other to assign tasks, report progress, etc. This requirement cannot be served by dedicated lines, since the number of lines required is impractical for physical backplanes. For eight processors, for example, each processor would require an interrupt line to seven other processors for a total of fifty six lines.

MiniBus provides for this functionality by through the use of interrupt flip-flops within each master board which may be addressed via a MiniBus transfer sequence. By performing the appropriate "write" operation to the physical address space of the target processor, the desired interrupt flip-flop can be set from any master (including the target processor itself).

Once the flip-flop is set, the request is queued in the target unit, and the requestor can proceed to other activities. The interrupt flip-flop will be driven into the normal mask/priority structure of the target processor and actually receive interrupt service when the local conditions permit. The target processor will reset the interrupt flip-flop with another "write" operation to its own physical address space as a part of the interrupt handling code.

11. Other Bus Signals

There exists a set of signals that are not generated or driven by the MBIC : Those related to the power source and to the overall system timing, as well as the signals that are generated by non-intelligent I/O boards.

Table 11 lists the signals not directly driven by the MBIC, listing their source and bus pin assignment.

Mnemonic	Signal	Pin	Source
BCLK	Bus Clock	C-18	Master #7
BCLK_2	1/2 x Bus Clock	C-22	"
RFRSHTIM~	Refresh Interval	C-24	"
MBICCLK	MBIC Bus Clock	C-20	"
RESET~	System Reset	A-13	"
PWAIT~	Periph. Wait	A-24	Slave w/o MBIC
PFAIL~	Power Fail	A-23	Power Source
INT00-08	Interrupt Req.	various	Boards w/o MBIC

Table 11.

MiniBus non-MBIC Signals

12. Interfacing to MiniBus

The previous sections have provided a comprehensive description of all the features available for users of MiniBus. It is a very complex bus with a multitude of capabilities from a functional standpoint, but at the same time it is a bus to which it is extremely simple to interface. Its simplicity stems from the fact that users need not implement all the available features but can just add the ones necessary to perform the desired function.

From a designer's viewpoint, two distinct type of products can be made to interface to the bus: Non-intelligent ("Dumb" I/O) or Intelligent (processors, DMA) units. The following sections detail some of the interface considerations for these units.

12.1. Non-Intelligent Interfaces

A wide variety of peripheral cards on MiniBus do not require high speed support and are designed to control a device where the protocol is simple enough as to not require a processor on the I/O board. There is never any need for these units to become bus masters and are thus considered Non-Intelligent or slave units on MiniBus .

All of the bus transfer timing and control is performed by the master, the slave just need decode its I/O page address and route the information to/from the master as appropriate.

The minimum number of signals required to interface a non-intelligent I/O unit to MiniBus is 20, not including the DC power and ground, and are shown in Table 12.1

Mnemonic	Description
A00 - A07	Data Lines
A08 - A15	Address Lines
A16	Read
A17	Write
IO~	I/O Transfer
RESET~	System reset

Table 12.1

Minimum Configuration Slave Signals

At the designer's option, PWAIT~ and one or more of the INT signals may be required to provide the timing or functionality desired.

12.2. Intelligent Interfaces

Intelligent units on MiniBus are defined to be those cards which contain an on-board processor or those with high speed I/O handling requirements so as to require DMA capability. All these units can potentially become bus masters and must therefore incorporate all MiniBus features.

MiniBus incorporates all the complex bus functions as well as all the interface drivers and receivers into a single 124-pin Gate Array, occupying an area of only 1.3 x 1.3 inches. This Gate Array provides an internal interface which is suitable for direct interface to NSC 32C016 micro-processor and CMOS static rams. Additional CMOS LSI interface devices will be developed in the near future.

13. Interface Requirements

13.1. MiniBus Signal Timing

13.1.1. Synchronous Signals

MiniBus signals which are specified relative to MBICCLK use MBICCLK as measured at the input pin of the MBIC which is driving/receiving the signal being measured.

Signal	Output from MBICCLK Min	Max	Input Setup Time Max
A21-00	5ns	40ns	40ns
BPAR~	5ns	70ns	15ns
BERR~	5ns	50ns	20ns
BCOD1~	5ns	40ns	40ns
BCOD0~	5ns	40ns	40ns
IO~	5ns	40ns	40ns
BREQi~	5ns	40ns	40ns

Table 14.1.1

Synchronous Signal Timing

13.1.2. Asynchronous Signals

To be completed

13.2. Electrical Specifications

To be completed