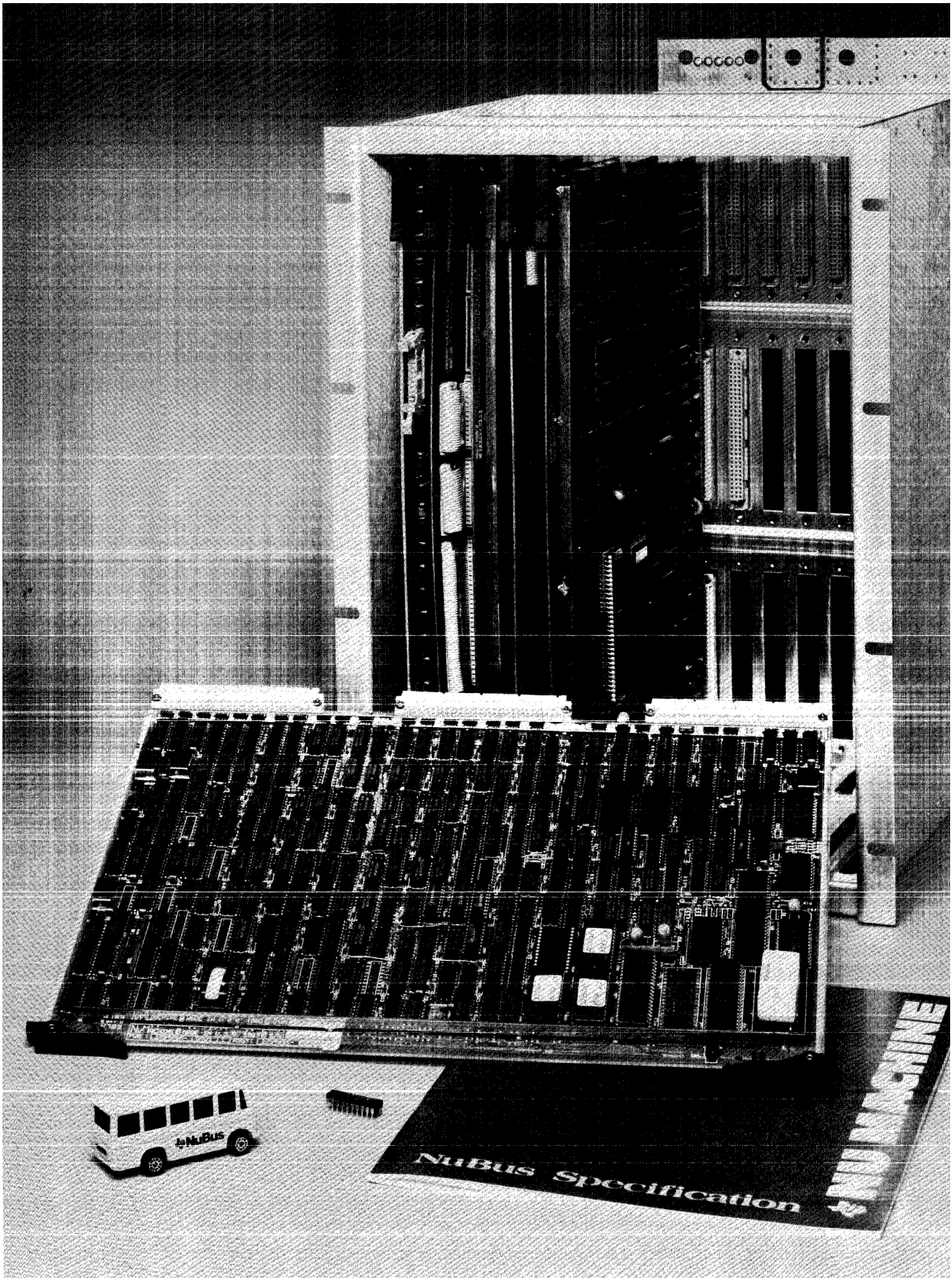
NU MACHINE

NuBus Specification


NuBus
NuBus Specification

NuBUS SPECIFICATION

Texas Instruments, Irvine, California 92714

Information furnished in this document is believed to be accurate and reliable. However, no responsibility is assumed by Texas Instruments for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Texas Instruments. Texas Instruments reserves the right to change product specifications at any time.

Contents

List of Figures

List of Tables

1 OBJECTIVES

The scope of this document is to define the NuBus interconnect system. Included are mechanical, electrical, pin assignment, and bus protocol specifications. Although some explanatory material is included, this document is primarily for those who intend to evaluate or design products that will be compatible with the NuBus.

The NuBus meets the following

OBJECTIVES

with these SUPPORTING FEATURES:

System architecture independent

Optimized for 32-bit transfers, but 8-bit and 16-bit nonjustified transfers supported.

Not based on a particular microprocessor's control structure.

Power up/down sequencing is system specified rather than dictated by bus specification.

High bandwidth

10 MHz basic cycle time provides 37.5 Mbyte bandwidth in block transfer mode.

Simplicity of protocol

Reads and writes are the only operations.

I/O and interrupts are memory mapped.

The single large physical address space allows all addressable units to be uniformly accessed.

Small pincount

Multiplexed data and address.

Excluding power and ground only 49 total signals.

Ease of system configuration

ID lines (geographical addressing) enable system to be free of "DIP" switches and jumpers.

Distributed, parallel arbitration eliminates daisy chains.

2 BASIC NuBUS STRUCTURE

2.1 Overview of NuBus Elements

The NuBus is a synchronous bus; all transitions and signal samplings are synchronized to a central system clock. However, it has many of the features of an asynchronous bus; transactions may be a variable number of clock periods long. This provides the adaptability of an asynchronous bus with the design simplicity of a synchronous bus.

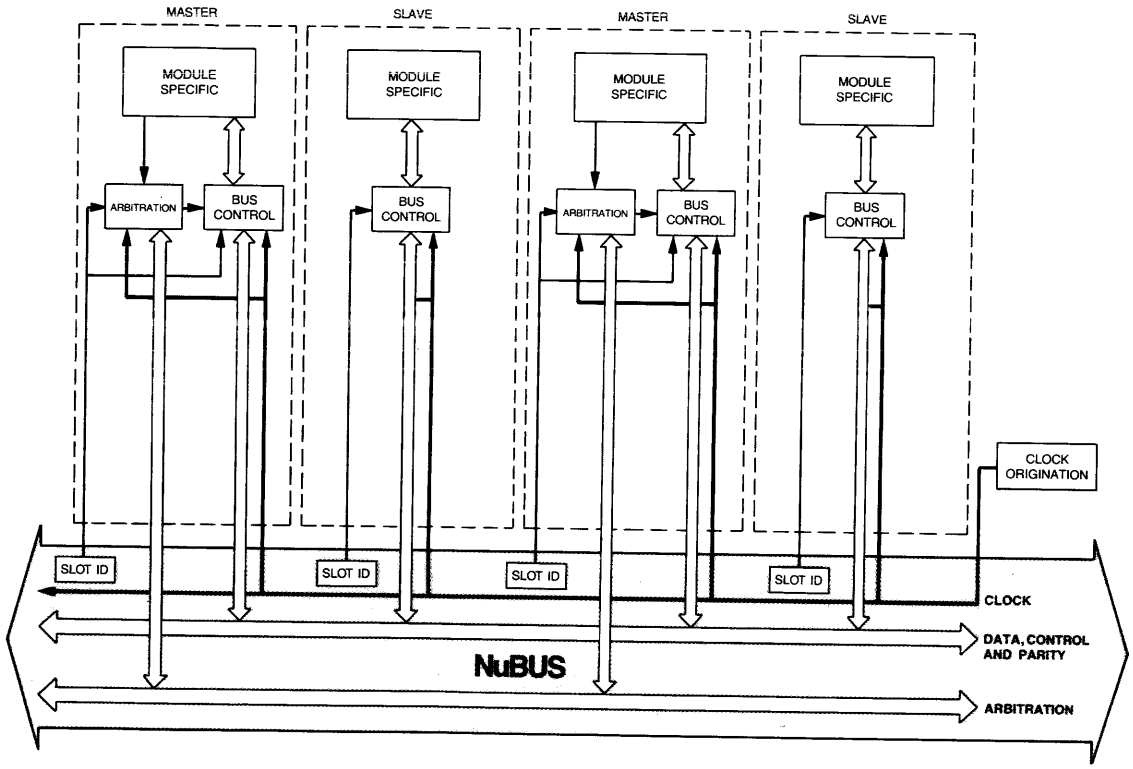


Figure 2–1. Simplified NuBus Diagram.

Figure 2–1 shows the major elements of a typical NuBus system. The NuBus supports multiprocessing and other sophisticated system architectures with a few simple mechanisms. This keeps interfaces to the NuBus "clean."

Only read and write operations in a single address space are supported; I/O and interrupts are accomplished within these mechanisms. The modules attached to the NuBus are peers; no card or slot position is the default master or "special slot." (The exception is that only one card drives the system clock line.) Each slot has an ID code hard wired into the

backplane. This allows boards to differentiate themselves without jumpers or "DIP" switches.

The simplified NuBus system diagramed in Figure 2 – 1 is made complete with the addition of only eight more lines. The ID, clock, address/data and ARB lines shown are supplemented with system reset, parity, and data transfer control lines.

2.2 NuBus Signal Classifications

NuBus signals can be grouped into five classes based on the functions which they perform. There are also power and ground lines. Table 2 – 1 shows the NuBus signal classifications.

Table 2–1. List of NuBus Signals.

CLASSIFICATION	SIGNAL	# OF PINS
Utility	RESET/	1
	CLK/	1
Control	START/	1
	ACK/	1
	TM0/	1
	TM1/	1
Address/Data	AD<31..0>/	32
Arbitration	ARB<3..0>/	4
	RQST/	1
Parity	SP/	1
	SPV/	1
Slot ID	ID<3..0>/	4
	Total Signals	**49**
Power/Ground	+5	11
	−5	8
	+12	2
	−12	2
	GND/	23
Reserved	RSVD/	1
	Total Pin Count	**96**

2.3 Basic NuBus Timing

The NuBus system clock has a 100 nsec period with a 75 nsec unasserted/25 nsec asserted duty cycle. Figure 2–2 shows the basic timing for most NuBus signals. The low to high transition of clock is used to assert and deassert signals on the bus. Signals are sampled on the high to low transition of the clock. The asymmetric duty cycle of the clock provides 75 nsec for propagation and setup time. With 25 nsecs between the sample and assertion edges, bus skew problems are avoided.

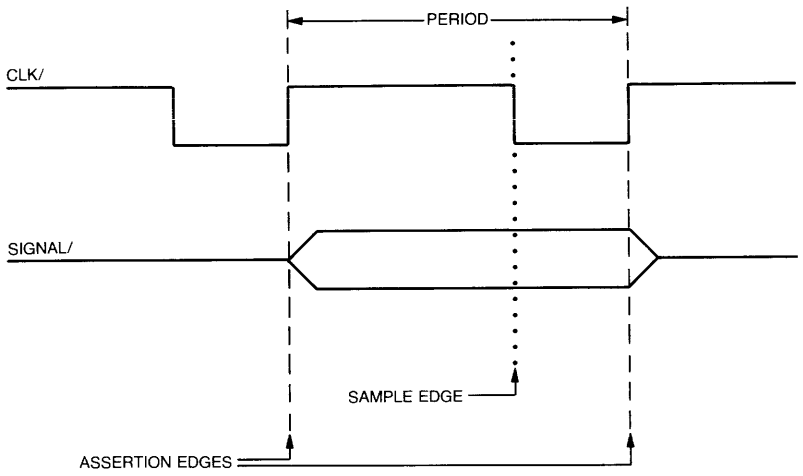


Figure 2–2. Basic NuBus Signal Timing.

2.4 Basic Definitions

Terms used throughout the NuBus specification are defined below. The relationships between some of these terms are illustrated in Figures 2 – 2 and 2 – 3. All NuBus signals are active when low and indicated by a "SIGNAL/" notation.

ACK CYCLE — Last period of a transaction (one clock period long) during which ACK/ is asserted. See Figure 2 – 3.

ADDRESS CYCLE — First period of a transaction (one clock period long) during which START/ is asserted. This term is synonymous with START cycle.

ASSERTED — This term is synonymous with low and true. Note: Since all NuBus signals are active low (indicated by a slash following the signal name), the terms low, true, and asserted are synonymous as are high, false, released and unasserted. The terms true and false have been avoided in this document.

ASSERTION EDGE — The rising edge (low to high) of the central system clock (CLK/).

CYCLE — A phase of a NuBus transaction. Address cycles are one clock period long and convey address and command information. Data cycles are also one period long and convey data and acknowledgement information.

DATA CYCLE	Any period in which data is known to be valid and acknowledged. It includes ACK cycles as well as intermediate data cycles within a block transfer. See Figure 2–3.
HIGH	This term is synonymous with inactive, deasserted, and released.
INACTIVE	This term is synonymous with high, deasserted, and released.
LOW	This term is synonymous with active and asserted.
PERIOD	The 100 nsec period of CLK/ (shown in Figure 2–2) consists of a 75 nsec unasserted state and a 25 nsec asserted state.
RELEASED	This term is synonymous with high, inactive, and deasserted.
SAMPLE EDGE	Falling edge (high to low) of the central system clock.
SLOT ID	The hex number (0 to 15) corresponding to each card slot. Each slot ID is established by the backplane and communicated to the card through the IDx/ lines.
SLOT SPACE	The upper one sixteenth of the total address space. These addresses are in the form F(ID)XXXXXX where F, (ID) and X are hex digits of 4 bits each. This address space is geographically divided among the NuBus cards according to slot ID numbers.
START CYCLE	This term is synonymous with Address cycle. See Figure 2–3.
TENURE	Bus tenure is a time period of unbroken bus ownership by a single master. A master may lock the bus and during one tenure, perform several transactions. The concept of bus locking is further explained in Section 5, Arbitration Operations.

TRANSACTION — A transaction is a complete NuBus operation such as read, write, block read, or block write. It is made up of one address cycle and one or more data cycles. See Figure 2–3.

UNASSERTED — This term is synonymous with high, false and released.

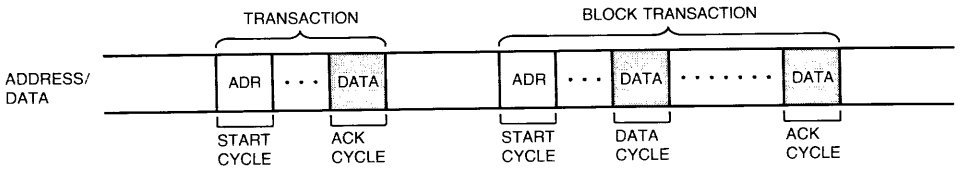


Figure 2–3. Cycle and Transaction Relationships.

3 UTILITY SIGNALS

This section identifies the signal lines which serve utility-type functions for the NuBus.

3.1 Clock Signal

Clock (CLK/), driven from a single source, synchronizes bus arbitration and data transfers between system modules. CLK/ has an asymmetric duty-cycle of 75% and a constant nominal frequency of 10 MHz. In general, signals are changed at the rising edge of CLK/, while they are sampled at the falling edge.

3.2 Reset Signal

Reset (RESET/) is an *open-collector* line which has two functions, depending on its duration. When asserted for a single clock period, RESET/ causes a NuBus interface initialization for all cards (Bus Reset). When active for more than one clock period, RESET/ returns all cards to the initial power-up state (System Reset). If RESET/ is asserted for more than one period, it must be asserted for one millisecond or longer.

3.3 Card Slot Identification Signals

Identification Signals 0 to 3 (ID0/ - ID3/) are binary coded to specify the physical location of each module. The highest numbered slot (15) has the four signals wired low. The lowest numbered slot (0) has all ID signals open. The distributed arbitration logic uses the ID numbers to uniquely identify cards for arbitration contests (explained in Section 5).

ID signals are also used to allocate a small portion of the total address space to each card. The upper 1/16th (256 megabytes) of the entire 4 gigabyte NuBus address space is called "slot space." As shown in Figure 3–1 , this area is further divided into sixteen, 16 megabyte pieces which are mapped to the 16 possible NuBus card slots (or ID codes). Addresses of the form F(ID)XXXXXX belong to a board's slot space. This fixed address allocation, based solely on a board's slot location, enables the design of systems which are free of jumpers and switches. The remaining 15/16ths of the 32-bit physical address space is uncommitted and allocated as required. Any allocation of that space is board and system dependent, but should be programmable via registers in slot space.

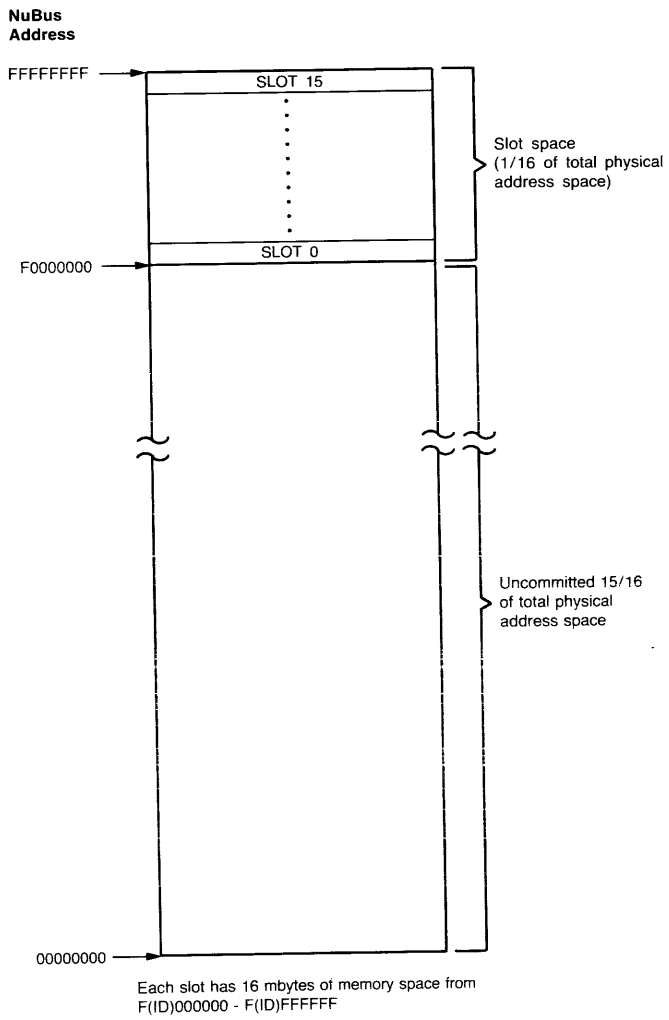


Figure 3–1. NuBus Address Space.

4 DATA TRANSFER OPERATIONS

4.1 Bus Data Transfer Signals

The bus data transfer signals, including Control, Address/Data, and Bus Parity, are all three-state.

4.1.1 Control Signals

This section describes the primary functions of the four NuBus control signals.

Transfer Start (START/) is driven for only one clock period by the current bus master at the beginning of a transaction. START/ indicates to the slaves that the address/data signals are carrying a valid address.

Transfer Acknowledge (ACK/) is driven for only one clock period by the addressed slave device and indicates the completion of the transaction.

Transfer Mode 0 and 1 (TM0/, TM1/) are driven by the current bus master during START/ cycles to indicate the type of bus operation being initiated. They are also driven by bus slaves during ACK/ cycles to denote type of acknowledgement. TMO/ and TM1/ encoding is given in Table 4 – 1.

4.1.2 Address/Data Signals

Address/Data 0 to 31 (AD0/ - AD31/) signals are multiplexed to carry a 32-bit byte address at the beginning of each transaction and up to 32 bits of data later in the transaction. Note that the AD0/ and AD1/ signals carry address information during the START cycle of a byte transaction and control information during the START cycle of a nonbyte transaction. AD0/ and AD1/ encoding is shown in Table 4 – 1.

4.1.3 Bus Parity Signals

System Parity (SP/) transmits parity information between NuBus cards which implement NuBus parity checking.

System Parity Valid (SPV/) indicates that the SP/ bit is being used. Cards which do not generate bus parity will never drive SPV/ active and cards which do not check parity will ignore SP/ and SPV/.

4.2 Data Transfer Specifications

The NuBus supports reads and writes of several different data sizes. Although optimized for transactions of words and blocks of words, the NuBus also supports byte and half word transactions. Table 4 – 1 shows the command encoding for data transfer operations, while Figure 4 – 1 shows the relationship between bytes, half words, and words.

Table 4–1. Transfer Mode Summary.

TM1/	TM0/	AD1/	AD0/	TYPE OF CYCLE
L	L	L	L	WRITE BYTE 3
L	L	L	H	WRITE BYTE 2
L	L	H	L	WRITE BYTE 1
L	L	H	H	WRITE BYTE 0
L	H	L	L	WRITE HALFWORD 1
L	H	L	H	WRITE, BLOCK
L	H	H	L	WRITE HALFWORD 0
L	H	H	H	WRITE WORD
H	L	L	L	READ BYTE 3
H	L	L	H	READ BYTE 2
H	L	H	L	READ BYTE 1
H	L	H	H	READ BYTE 0
H	H	L	L	READ HALFWORD 1
H	H	L	H	READ, BLOCK
H	H	H	L	READ HALFWORD 0
H	H	H	H	READ WORD

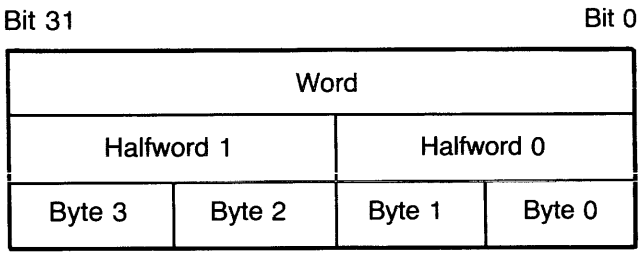


Figure 4–1. Layout of Words, Halfwords, and Bytes.

All NuBus data transfers are unjustified as shown in Figure 4–2. A byte of data is conveyed on the same signal lines regardless of the transfer mode used to access it. Therefore, bytes with address 0 modulo 4 are always carried by AD0/ through AD7/, bytes 1 modulo 4 by AD8/ through AD15/, bytes 2 modulo 4 by AD16/ through AD23/, bytes 3 modulo 4 by AD24/ through AD31/. This unjustified data path approach allows straightforward connection of 8-bit, 16-bit, and 32-bit devices.

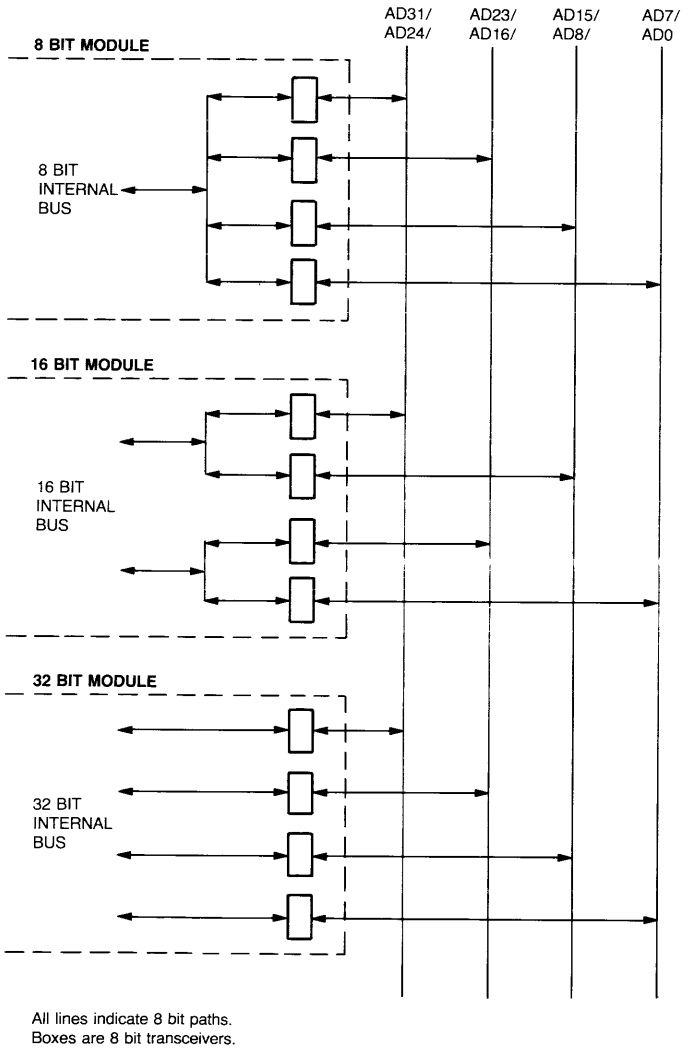


Figure 4–2. Data Paths for 8, 16, and 32 Bit Devices.

4.2.1 Single Data Cycle Transactions

The simplest transactions on the NuBus convey one data item and consist of a START cycle and a subsequent ACK cycle. These transactions are either reads or writes of either bytes, halfwords or words.

All transactions are initiated by a bus master which drives START/ active while driving the TMx/, AD0/ and AD1/ signals to define the cycle type. The remaining ADx/ signals are also driven to convey the address. The transaction is completed when the responding slave drives ACK/ active while driving status information on the TMx/ signals. For write transactions, the master must switch the ADx/ signals from address to data information in the

second clock period and hold that data until acknowledged. In read cycles, the slave drives the data simultaneously with the acknowledge in the last period.

4.2.1.1 Read Transactions Figure 4–3 shows the timing for read bus transactions other than block transfers. Read operations with data widths of 8, 16, and 32 bits are selected by the transfer mode signals (TMx/) and the two low order address signals (AD1/ and AD0/) as shown in Table 4–1. The slave must put the requested data item on either 8, 16, or all 32 of the AD0/ to AD31/ signals. Any bits other than the requested data may be driven either high or low by the slave.

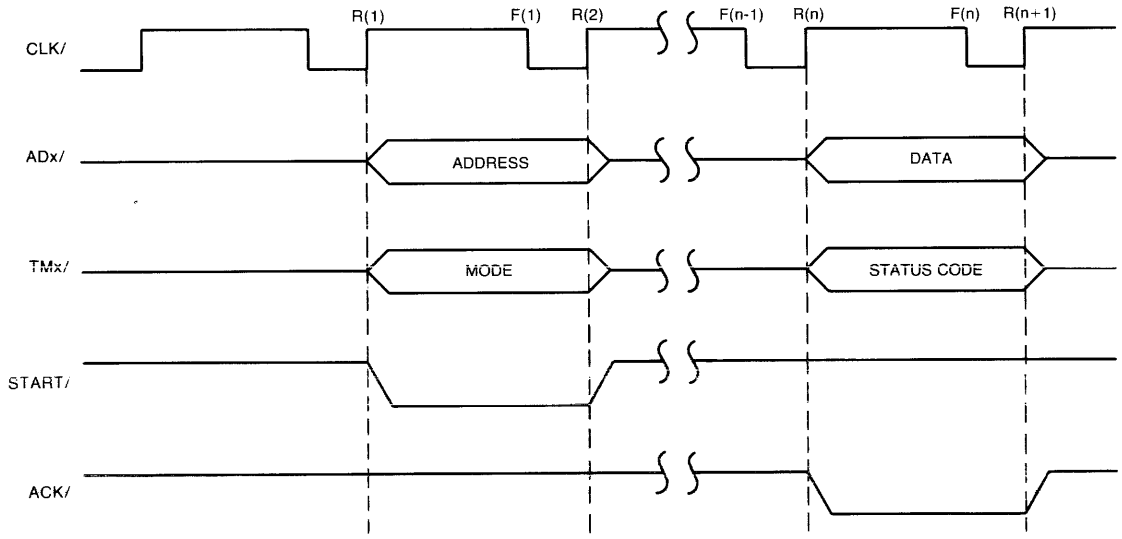


Figure 4–3. NuBus Read Operation.

Once the bus master has acquired the bus, a read bus transaction involves the following steps:

R(1) The bus master asserts START/ and the appropriate ADx/ and TMx/ lines to initiate the transfer.

F(1) The bus slaves sample the ADx/ and TMx/ lines.

R(2) The bus master releases the ADx/, TMx/, and START/ lines and waits for an ACK/.

R(n) The bus slave places the requested data onto the ADx/ lines, asserts ACK/, and places the appropriate status code on TM0/ and TM1/.

F(n) The bus master samples the ADx/ and TMx/ lines to receive the data and note any error condition.

R(n + 1) The bus slave releases the ADx/ and ACK/ lines and TMx/ lines. This may be the R(1) of the next transaction.

F = Falling edge of CLK/

R = Rising edge of CLK/

2 ≤ n ≤ System defined timeout period

4.2.1.2 Write Transactions Figure 4 – 4 shows the timing for write operations other than block transfers. Write operations with data widths of 8, 16, and 32 bits are selected by the transfer mode signals (TMx/) and the two low order address signals.

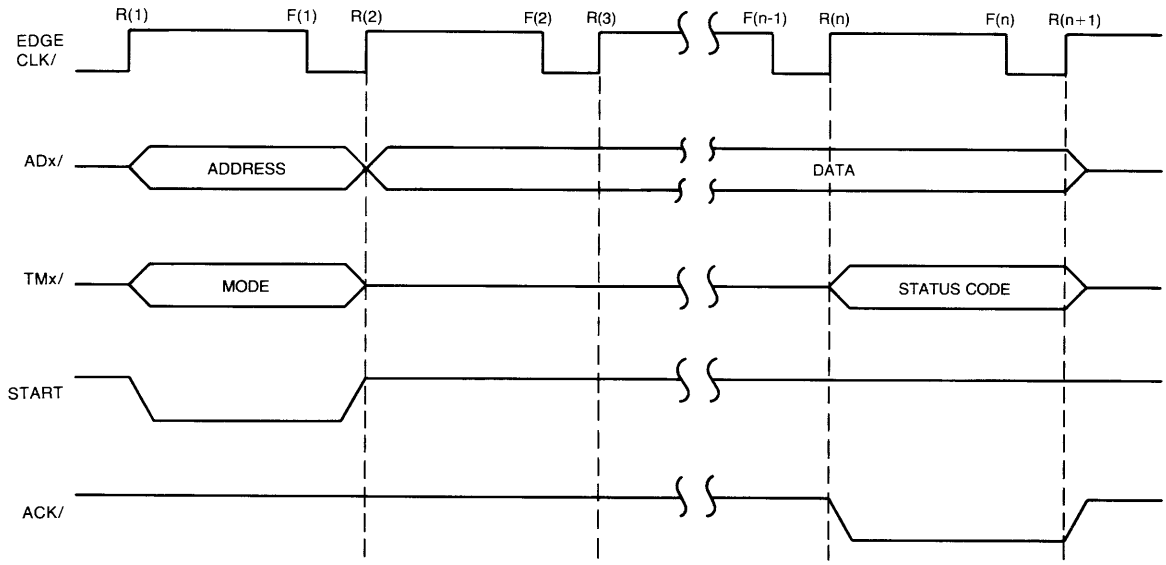


Figure 4–4. NuBus Write Operation.

The bus master has the responsibility for aligning data onto the appropriate ADx/ lines for halfword and byte writes. For example, a write byte 3 requires that the data be placed on AD24/ through AD31/; all other ADx/ lines are not defined and can each be driven to either a high or low state.

Once the bus master has acquired the bus, a write bus transaction involves the following steps:

R(1) The bus master asserts START/ and the appropriate ADx/ and TMx/ lines to initiate the transfer.

F(1) The bus slave samples the ADx/ and TMx/ lines.

F(2) The bus master places the data to be written onto the ADx/ lines, releases the START/ and TMx/ lines, and waits for an ACK/.

R(n) The bus slave asserts ACK/ and places the appropriate status code on TM0/ and TM1 when the data is accepted.

F(2)-F(n) The bus slave samples the ADx/ lines to capture the data. The data may be sampled before or during the assertion of ACK/.

R(n + 1) The bus master releases the ADx/ lines while the Bus Slave releases the ACK/ and TMx/ lines.

F = Falling edge of CLK/

R = Rising edge of CLK/

$2 \leq n \leq$ System defined timeout period

4.2.1.3 Idle Cycle Operation An idle cycle is defined as the assertion of START/ and ACK/ in the same period. In this case, the ADx/ and TMx/ lines are ignored by all bus modules and *no* data may be transferred.

Idle cycles are used solely to reinitiate bus arbitration. (Refer to Section 5 for a detailed explanation of bus arbitration.) If the bus is requested and acquired, but is not used (or wanted), an idle cycle must be inserted by the bus master to initiate a new contest among any contenders that are asserting RQST/. If RQST/ is not asserted, an idle cycle is not needed (but may be inserted). The arbitration period may now be timed from the assertion of RQST/.

4.2.1.4 Acknowledgement During ACK cycles the addressed slave drives the TMx/ lines while it drives ACK/. The TMx/ lines provide status information to the current bus master as shown in Table 4 – 2.

Table 4–2. Status Code Summary.

TM1/	TM0/	TYPE OF ACKNOWLEDGE
L	L	BUS TRANSFER COMPLETE
L	H	ERROR
H	L	BUS TIMEOUT ERROR
H	H	TRY AGAIN LATER

Bus Transfer Complete The bus transfer complete response indicates the normal valid completion of a bus transaction.

Error During a read or write operation, certain error conditions may occur. Examples of errors are: an uncorrectable error during an ECC memory read or a bus parity error. The transaction terminates in a normal manner and the bus master has responsibility for handling the error condition.

Bus Timeout If an unimplemented address location is summoned, the attempted transaction is acknowledged with a bus timeout response. This timeout response indicates that the system defined timeout period has elapsed while the bus is "busy" (between START/ and ACK/) and no transfer acknowledge has occurred.

Bus timeout also occurs if a contender requests the bus and does not generate a START cycle. In this case, the bus timeout logic generates an IDLE cycle to reinitiate bus arbitration.

Bus timeout responses are generated by the system timeout logic; therefore, timeout logic is not required on each card of the system. The bus timeout period is typically programmable and should be substantially longer than the response time of the slowest slave in the system.

Try Again Later An addressed slave may be unable to respond to a master's transfer request at the time of the request. Try again later indicates that the transfer cannot be accomplished at this time. The master should rearbitrate for the bus later. This is not an error indication.

4.2.2 Block Transfers

Block transfers are transactions which consist of a START cycle, multiple data cycles from/to sequential address locations, and an ACK cycle. The number of data cycles is controlled by the master and communicated during the START cycle. Allowed lengths of block transfers are 2, 4, 8, and 16 words. (Only word transfers are supported in block mode.)

The AD0/, AD1/ and TMx/ encoding for block transfers is shown in Table 4–1. The block's starting address must correspond to it's size and is encoded by the AD2/ through AD5/ lines as shown in Table 4–3.

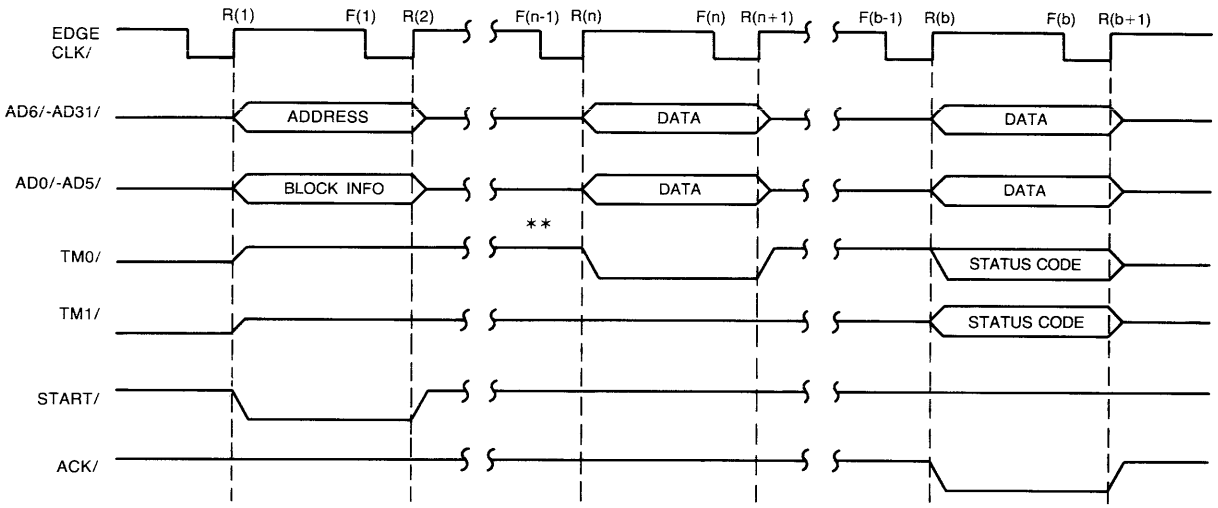
Table 4–3. Block Size and Starting Address Summary.

AD5/	AD4/	AD3/	AD2/	BLOCK SIZE (WORDS)	BLOCK STARTING ADDRESS
X	X	X	H	2	(AD31——AD3) 000
X	X	H	L	4	(AD31——AD4) 0000
X	H	L	L	8	(AD31—AD5) 00000
H	L	L	L	16	(AD31–AD6) 000000

During block transfers, each data cycle is acknowledged by the responding slave. The

intermediate acknowledges are data cycles where TM0/ is active and TM1/ and ACK/ are both inactive. For intermediate acknowledgements, TMO/ has the same significance and timing as the ACK/ signal for nonblock transfers. The final word transfer acknowledge is a standard ACK cycle. Status codes are indicated as shown in Table 4–2.

4.2.2.1 Block Read Figure 4–5 shows the timing for a NuBus block read operation. See Table 4–1 for AD0/, AD1/, TM0/ and TM1/ encoding which initiates block reads. The AD2/ through AD5/ lines determine the size and starting address of the transaction as shown in Table 4–3. The responding slave drives data onto the bus and the initiating bus master accepts the data on each intermediate or final acknowledge.



** The addressed slave is responsible for driving TM0/ to the desired state between R(n) and R(b+1)

Figure 4–5. NuBus Block Read Operation.

Once the bus master has acquired the bus, a block read consists of the following steps:

R(1)	The bus master asserts START/ and the appropriate ADx/ and TMx/ lines to initiate the transfer.
F(1)	The bus slaves sample the ADx/ and TMx/ lines.
R(2)	The bus master releases the ADx/, TMx/, and START/ lines and waits for an intermediate acknowledge (TM0/ active).
R(n)	The bus slave places the first word of requested data on the ADx/ lines and asserts TM0/.
F(n)	The bus master samples the ADx/ lines and TM0/ to capture data. TM0/ is active and data is next data item.
R(n + 1)	If the next consecutive word of data is not ready to be put on the bus, the slave drives TM0/ inactive until the word is ready.
NOTE:	The previous 3 steps are repeated for ascending addresses until B-1 words have been transferred, where B is the block size.

R(b) The bus slave places the final word of requested data onto the ADx/ lines, asserts ACK/ and places the appropriate status code on TM0/ and TM1/.

F(b) The bus master samples the ADx/ and TMx/ lines to receive the data and note any error conditions.

R(b + 1) The bus slave releases the ADx/, ACK/, and TMx lines.

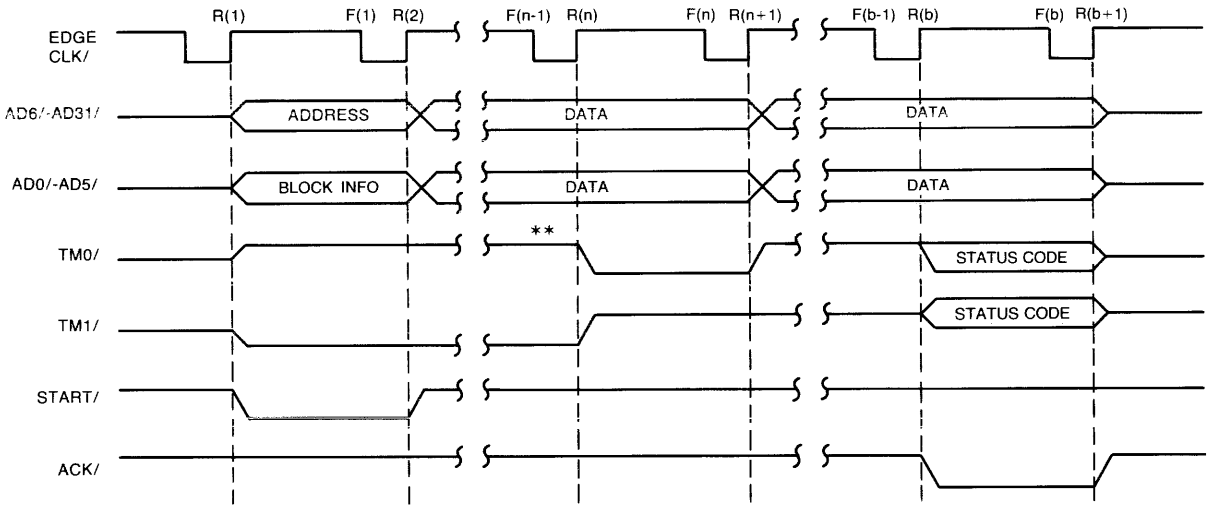
F = Falling edge of CLK/

R = Rising edge of CLK/

$2 \leq n \leq$ System defined timeout period

$2 \leq b \leq B$ * System defined timeout period

4.2.2.2 Block Write Figure 4 – 6 is a timing diagram for a NuBus block write operation. Block writes are similar to block reads except the bus master drives the data bus while the slave accepts data. The format for describing block size and starting address is the same as for block reads.



** The addressed slave is responsible for driving TM0/ to the desired state between R(n) and R(b+1)

Figure 4–6. NuBus Block Write Operation.

Once the bus master has acquired the bus, a block write consists of the following steps.

R(1) The bus master asserts START/ and the appropriate ADx/ and TMx/ lines to initiate the transfer.

F(1) The bus slave samples the ADx/ and TMx/ lines.

R(2)	The bus master places the data to be written onto the ADx/ lines, releases the START/ and TMx/ lines, and waits for an intermediate acknowledge (TMO/ active).
R(n)	The bus slave asserts TM0/ when the first word of data is accepted.
F(n-)	The bus slave samples the ADx/ lines to capture the data being written. The n- notation implies the data may be sampled before or during the assertion of TM0/.
R(n + 1)	The bus master places the next consecutive word of data on the bus.
NOTE:	The previous 3 steps are repeated for ascending addresses until B-1 words have been transferred, where B is the block size.
R(b)	The bus slave asserts ACK/ and places the appropriate status code on TM0/ and TM1/ when the final word of data is accepted
F(b-)	The bus slave samples the ADx/ lines to capture the data. The b- notation implies the data may be sampled before or during the assertion of ACK/.
R(b + 1)	The bus master releases the ADx/ lines while the bus slave releases the ACK/ and TMx/ lines.

F = Falling edge of CLK/

R = Rising edge of CLK/

$2 \leq n \leq$ System defined timeout period

$2 \leq b \leq B$ * System defined timeout period

4.2.2.3 Block Transfer Errors Although the length of a block transfer is dictated by the master during the START cycle, a block transfer may be cut short by an error acknowledgement from the slave at any time. The standard status codes shown in Table 4 – 2 are used.

4.2.3 Interrupt Operations

Interrupts on the NuBus are implemented as write transactions. Interrupt operations require no unique signals or protocols. Any module on the NuBus can interrupt a processor module by performing a write operation into an area of memory that is monitored by that processor. Any address range on the processor module can be defined as its interrupt space. This allows interrupts to be posted to individual processors and allows interrupt priority to be software specified by memory mapping the priority level.

4.3 Bus Parity

Parity protects the integrity of all address and data transfers. Parity generation is optional on a cycle by cycle basis. Only if the module driving the ADx/ lines generates parity *and* the module capturing the address or data information checks it, is parity useful.

The two signals SP/ and SPV/ are used to communicate parity and parity valid, respectively. When SPV/ is asserted (with the same timing as the Address/Data lines), parity is generated. SP/ carries even parity, meaning that if an even number of lines within AD0/ to AD3/ are asserted, SP/ is unasserted; otherwise it is asserted. Parity checking is optional. Modules which cannot check parity normally ignore the SPV/ and SP/ signals. Although byte

and half word transfers are supported, parity is always calculated over the complete 32 bits of the ADx/ lines, simplifying interface design.

If parity errors are detected:

During Address cycle

Slave ignores address. Transaction will probably timeout.

During Data cycle on read

Master ignores the data it received.

During Data cycle on write

Slave acknowledges with the error status code.

4.4 Compliance Categories

Devices may be designed which conform to the NuBus specification, but do not support all NuBus features. Masters and slaves do not need to support all transfer types. Any combination of 8-, 16-, and 32-bit single data transfers or block transfers as either master or slave is allowable.

5 ARBITRATION OPERATIONS

5.1 Arbitration Signals

5.1.1 Arbitrate Signals

Arbitrate 0 to 3 (ARB0/ - ARB3/) are *open-collector* binary coded lines driven by contenders for the bus. They are used by the distributed arbitration logic to determine bus mastership.

5.1.2 Request Signal

Bus Request (RQST/) is an *open-collector* line driven low by contenders for the bus.

5.2 Arbitration Specification

The NuBus fair arbitration mechanism differs from strict priority arbitration in that it prevents "starvation" of cards and distributes bus bandwidth evenly.

During arbitration, one or more modules contend for control of the NuBus. Modules which desire ownership of the NuBus must first assert the RQST/ line. RQST/ may only be asserted while it is in an unasserted state. All modules which assert RQST/ place their ID codes on the ARBx/ lines and contend for the bus. The arbitration logic distributed among the modules determines which of the modules has ownership of the NuBus. After two clock periods, the contest mechanism settles. The contender with the highest ID code has its code on the ARBx/ lines, has won bus ownership, and may initiate a transaction.

Presuming the winner does *not* desire to lock the bus, the winning module removes its RQST/ at the same time that START/ is asserted and removes its ARBx/ signals after the START cycle of its first transaction. The release of START/ initiates another contest between any modules who originally requested the bus in the same clock period, but who have not yet won. These modules will be granted ownership in turn from highest ID number to lowest ID number. The rule that RQST/ must be unasserted before a module may assert it keeps other modules from participating in contests until all the original requestors have been served.

Figure 5 – 1 shows a situation in which modules with codes #1 and #2 request the bus at the same clock period. Module #2 wins the first arbitration contest, and then removes its request after its START cycle. Module #15 desires the bus as well but may not request it because the RQST/ line is already asserted. Contesting against no one, module #1 wins the next contest and gains bus ownership. When module #1 releases RQST/, module #15 requests, arbitrates and wins. Note that module #1 owns the bus only after it both wins a contest *and* the transaction in progress ends.

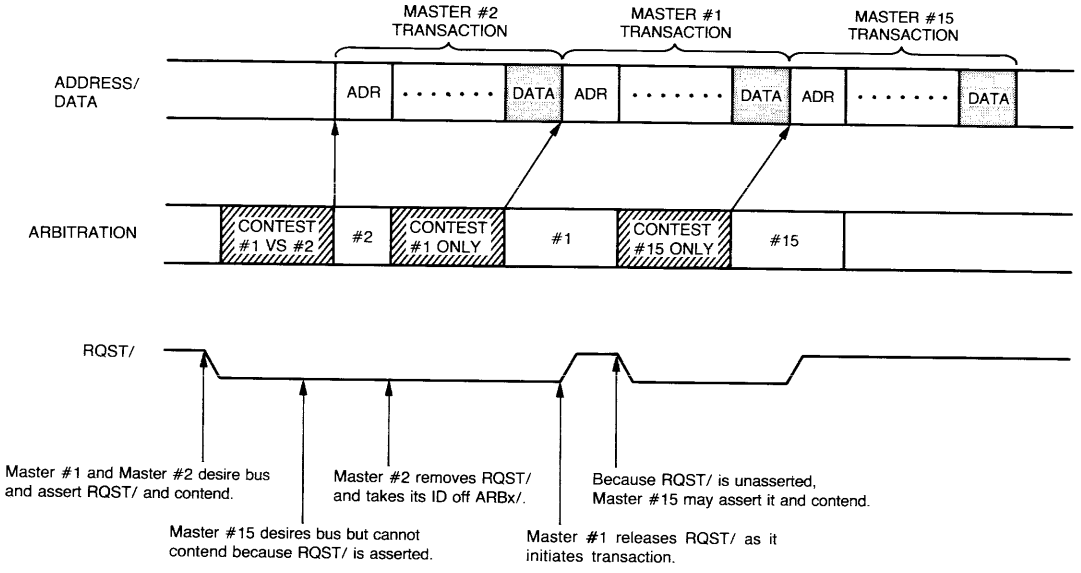


Figure 5–1. Sample Arbitration Contest.

5.2.1 Arbitration Logic Mechanism

When a bus contest occurs, each module drives the ARBx/ lines with its unique ID code and then deasserts ARBx/ lines if it detects higher ID codes than its own on the ARBx/ lines. One possible implementation of this arbitration logic is diagrammed in Figure 5 – 2.

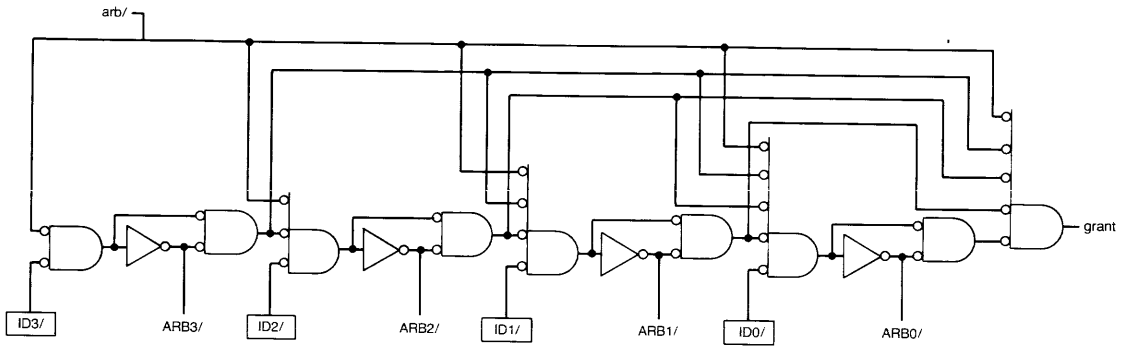


Figure 5–2. Typical Bus Arbitration Logic.

Note that the ARBx/ lines are bussed common to all cards while the IDx/ lines present a unique binary code to each card slot. The signals "arb" and "grant" are module signals. "Arb" indicates whether the module is contending for the bus and "grant" indicates whether the ARBx/ lines currently match this module's IDx/ lines. The following logic equations describe how the arbitration logic on any given module works:

ARB3 = ID3 * arb

ARB2 = ID2 * arb * (ID3 + ARB3 /)

ARB1 = ID1 * arb * (ID3 + ARB3 /) * (ID2 + ARB2 /)

ARB0 = ID0 * arb * (ID3 + ARB3 /) * (ID2 + ARB2 /) * (ID1 + ARB1 /)

* is LOGICAL AND
+ is LOGICAL OR

According to these equations, after a short delay (arbitration period) the ARBx/ lines will equal the ID code of the highest priority contender.

5.2.2 General Arbitration Timing

The details of arbitration timing are covered in Paragraph 6.2. Arbitration events generally occur on assertion edges and sample edges, synchronous to the system clock, with the same timing as the basic Address/Data, control and utility signals. For example, RQST/ may be asserted on a particular assertion edge only if it is seen to be unasserted on the previous sample edge. However, the ARBx/ lines differ from all other NuBus signals in that their assertion timing is specified from the sample edge of the bus clock.

Contests last two clock periods by definition. On the second sample edge after a contest starts, all contenders sample their internal grant signal. The highest priority contender will find its grant signal asserted. The winner may now take control of the bus and assert START/ on the next assertion edge (25 nsec after the contest's second sample edge) if the bus isn't in use. If the bus *is* in use, the new winner asserts START/ on the assertion edge immediately after the next sample edge where the current transaction's ACK/ is asserted. The new winner continues to assert it's ID code on the ARBx/ lines throughout the START/ cycle of it's first transaction. This facilitates bus lock detection and bus diagnostics.

5.2.3 Bus Locking

Although modules generally use the bus for short periods, sometimes a module must lock the bus. An example of this is an indivisible test-and-set operation performed in a multiprocessor environment.

Bus locking is accomplished with no added mechanism. To lock the bus, a master simply continues to request and contend. Since it has the highest ID code of those modules present, it wins subsequent contests. Figure 5–3 shows an example in which module #4 locks the bus for two transactions.

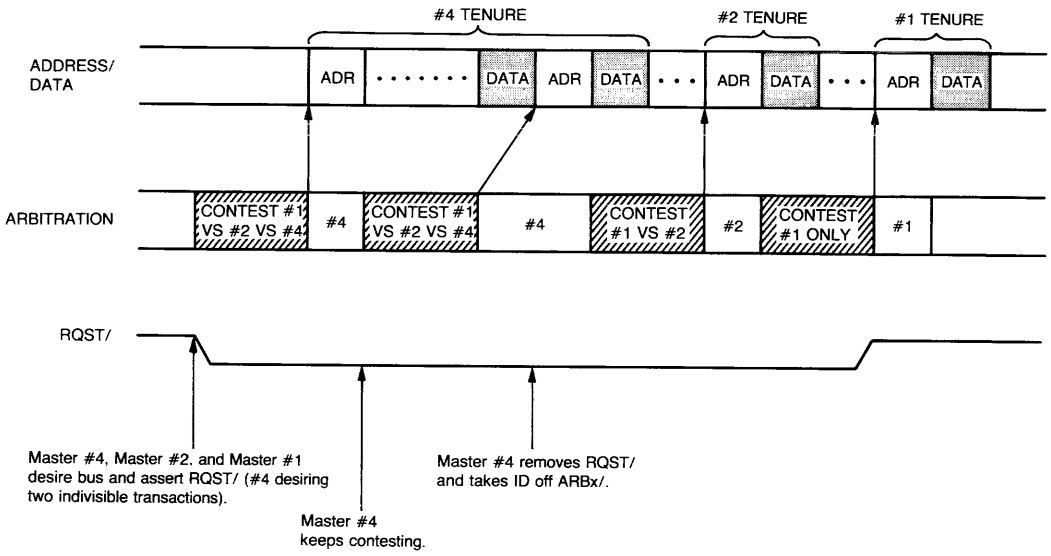


Figure 5–3. Sample Bus Lock.

5.2.4 Bus Parking

In NuBus systems with few modules, a bus master which has released RQST/, "parks" on the bus and may use it at any time until another module asserts RQST/. When RQST/ is finally asserted, the "parked" bus master finishes its current transaction and relinquishes the bus to the new winner. Bus parking reduces the average time period to acquire the bus in systems with a small number of contenders.

6 ELECTRICAL AND TIMING SPECIFICATIONS

6.1 Electrical Requirements

6.1.1 Logical and Electrical State Relationships

All NuBus signals are active when low. The relationship between logical states and electrical signal levels for all NuBus lines is shown in Table 6 – 1.

Table 6–1. Logical State Definitions.

LOGICAL STATE	ELECTRICAL SIGNAL LEVEL (ACTIVE LOW)
H (Unasserted)	> 2.0V at the receiver
L (Asserted)	< 0.8V at the receiver

6.1.2 DC and AC Specifications for Signals

This paragraph provides the drive requirements, the load allowance, and the required termination for each of the NuBus lines. These lines can be divided into four basic types: Clock, Address/Data(ADx/, SP/, SPV/), Control (START/, ACK/, TMx/), and Open Collector (RESET/, RQST/, ARBx/). Table 6 – 2 lists the specifications for these line (signal) types. Note that the 16 pF AC loading for each of the lines includes the etch and device capacitances only, the capacitive load of the connector need not be included. An additional 2 pf have been allowed for the connectors mating the backplane to the card.

Table 6–2. Bus Drivers, Receivers, and Terminations.

SIGNAL TYPE	DC DRIVE		AC LOADING	DC LOADING		TERMINATION	
	I_{OL} (min)	I_{OH} (min)	C_L (max)	I_{IL} (max)	I_{IH} (max)	Pullup/ Pulldown	Location
Clock	60mA	−40mA @2.0V	16pF	−1.4mA	0.1mA	120/180Ω	1 end (away from driver)
Address/Data	48mA	−25mA (I_{OSC})	16pF	−0.9mA	0.1mA	270/470Ω	both ends
Control	48mA	−25mA (I_{OSC})	16pF	−0.9mA	0.1mA	270/470Ω	both ends
Open Collector	60mA	n.a.	16pF	−0.625mA	0.1mA	180/470Ω	both ends

I_{OL} — Low Output Drive Current available at 0.5 volts.
I_{OH} — High Output Drive Current available at specified voltage
I_{OSC} — Short Circuit Output Current
I_{IL} — D.C. Low Level Input Current
I_{IH} — D.C. High Level Input Current

NOTE

Negative currents indicate flow out of a node and positive currents indicate flow into a node.

6.1.3 Line (Signal) Characteristics

To meet the NuBus timing requirements with the drivers and terminations described in Paragraph 6.1.2, the characteristic impedance (Z_0') and the propagation delay (T_{pd}) of a backplane loaded with cards must be controlled. Both of these parameters depend on backplane geometries (length, card spacing, layer separation, etc.) as well as the sort of dielectric used in the backplane. Only the critical parameters (Z_0' and T_{pd}) are specified by this document allowing variations in geometry and dielectric.

Z_0' is the loaded impedance of the NuBus backplane. T_{pd} is the bus propagation delay. Their specifications are:

$$Z_0' \geq 25 \text{ ohms}$$

$$T_{pd} \leq 8.5 \text{ nsec}$$

6.1.4 Backplane Physics

To assist designing backplanes, the following background information is offered.

Impedance:

$$Z_0' \geq 25 \text{ ohms}$$

Where $Z_0' = \dfrac{Z_0}{\sqrt{1 + \dfrac{C_d}{C_0}}}$

Z_0 is the characteristic impedance of the line without cards, connectors or feed-throughs present. C_0 is the corresponding capacitance in pf/in. C_d is the distributed capacitance and equals 18 pf/card divided by the card spacing (sp).

For example, if the unloaded characteristic impedance was 80 ohms and the unloaded capacitance was 2.5 pf/in, then, the minimum card spacing would be determined as:

$$25 = \frac{80}{\sqrt{1 + \dfrac{C_d}{2.5}}} \longrightarrow \frac{C_d}{2.5} = 9.24 \longrightarrow C_d = 23.1$$

$C_d = 23.1 = 18/\text{sp}$
minimum sp = 0.78 inches

Thus, 0.8 inch spacing is suitable for this backplane.

Delay:

$$T_{pd} \leq 8.5 \text{ nsec}$$

Where $T_{pd} = L \times D_0 \times \sqrt{1 + \dfrac{C_d}{C_0}}$

L is the length of the backplane in inches, D_0 is the unloaded propagation delay in nsec/in and C_d and C_0 are as above. If D_0 was 0.2 nsec/in then

$$L < \frac{8.5}{0.2 \times 3.2} \Longrightarrow L < 13.2 \text{ inches}$$

Thus, the backplane can be as long as 13.2 inches.

6.1.5 Power Supply Specifications

Four voltages are specified on the NuBus. These voltages, +5, −5, +12 and −12 volts, are listed in Table 6−3 with their specifications.

Table 6–3. Power Supply Specifications.

SOURCE LABEL	NOMINAL VALUE (VOLTS)	TOLERANCE FROM NOMINAL	COMBINED LINE AND LOAD REGULATION	MAX RIPPLE (PK-PK)
+5	5V	+/− 3%	0.3%	50MV
−5	−5V	+/− 3%	0.3%	50MV
+12	12V	+/− 3%	0.3%	75MV
−12	−12V	+/− 3%	0.3%	75MV

6.2 Timing Requirements

6.2.1 Utility and Data Transfer Timing

Figure 6 – 1 shows the clock, control, and address/data timing relationships during data transfers.

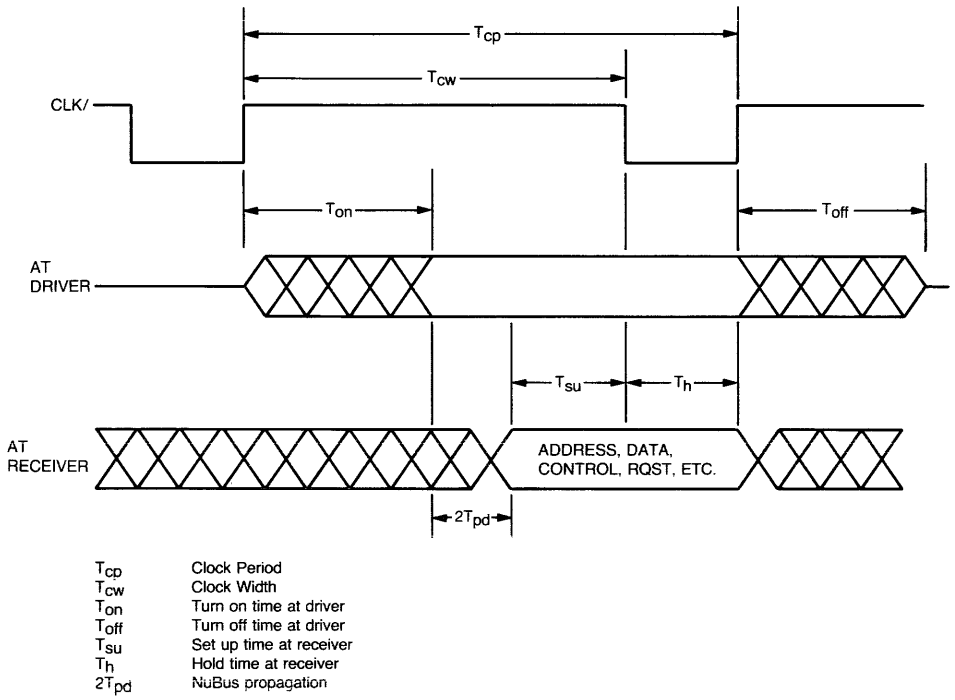


Figure 6–1. Data Transfer Timing.

Control and Address/Data signals are changed on the rising edge of CLK/ and sampled on the falling edge of CLK/. This gives protection from bus skew. Table 6–4 lists the bus timing specification for all these signals.

Table 6–4. Bus Timing Specification Summary.

PARAMETER	DESCRIPTION	MINIMUM	MAXIMUM	UNITS	FIGURE
T_{cp}	Clock Period	99.99	100.01	ns	3.2
T_{cw}	Clock Width	73	77	ns	3.2
T_{on}	Turn on Time	0	35	ns	3.2
T_{off}	Turn off Time	0	35	ns	3.2
$2T_{pd}$	NuBus Delay		17	ns	3.2
T_{su}	Set up Time	21		ns	3.2
T_h	Hold Time	25		ns	3.2

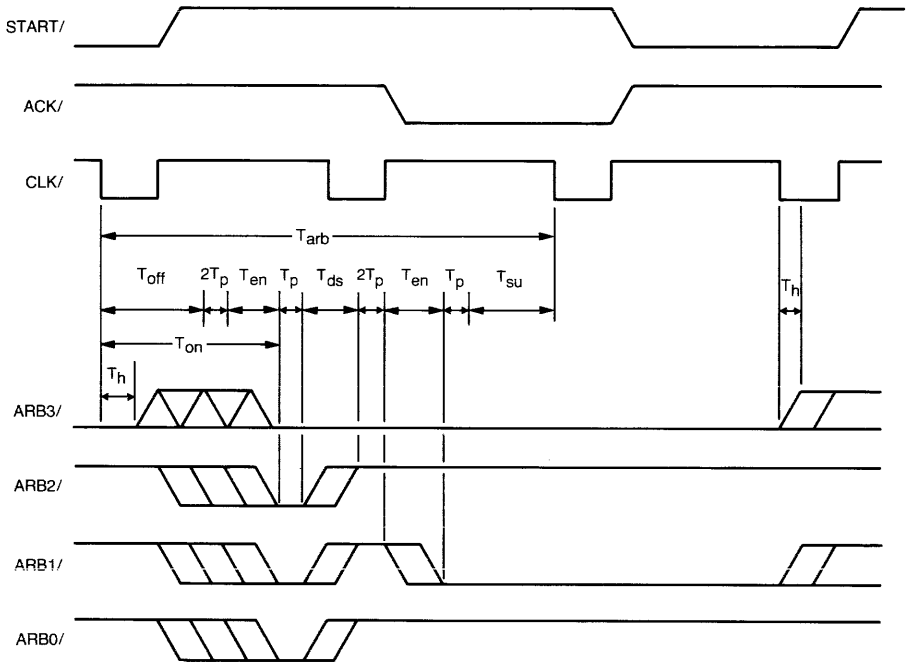
NOTE

Setup, Hold, and other times are defined at the board-to-NuBus connector. All board-internal delays must be taken into account while providing for the above specified times.

6.2.2 Arbitration Timing

Refer to Paragraph 5.2 for a description of NuBus arbitration. The settling time for the ARBx/ signals is not the same as the timing of the data transfer signals. Arbitration begins on the falling edge of CLK/ before the assertion of RQST/ or, if RQST is already active on the falling edge of CLK/, during START/. The contenders assert their respective slot ID's on the ARBx/ lines. The bus contest must settle within two cycles of CLK/.

Figure 6–2 details the ARBx/ timing for an arbitration between module #A (1010) and module #5 (0101) following a START/ initiated by module #8. In the general case, contenders must wait for the preceding bus master to release the ARBx/ lines before the succeeding bus arbitration can take place. Thus, the assertion time (Ton) for ARBx/ signals is the turn off time of the preceding master (Toff), plus the bus propagation delay (2Tp), plus the time taken to react to the change in logic levels (Ten). At the end of this time (Ton), both devices temporarily assert their slot ID's on the ARBx/ lines resulting in an "F" (1111) pattern; this causes module #A to release ARB1/ and module #5 to release ARB2/ and ARB0/. After ARB2/ reaches a high state, module #A reasserts ARB1/. On the second falling edge of CLK/ following the assertion of RQST/ or the negation of START/, module #A will win the arbitration contest and will be the next bus master.



T_{arb}-	Arbitration time
T_{on}-	Turn on time of initial assertion
$2T_p$-	NuBus delay (1 reflection)
T_p-	NuBus delay
T_{ds}-	Disable time from detection of higher priority request
T_{en}-	Reassertion after detection of false higher priority request
T_{su}-	Set up time of ARBx/ lines at receiver
T_h-	Hold time at receiver
T_{off}-	Turn off time of ARBx/ lines

Figure 6–2. Detail Arbitration Timing.

Table 6–5 lists the timing specifications for the ARBx/ lines. The numbers in Table 6–5 correspond to the parameters illustrated in Figure 6–2.

Table 6-5. Bus Arbitration Timing Specification Summary.

PARAMETER	DESCRIPTION	MINIMUM	MAXIMUM	UNITS	FIGURE
T_{arb}	Arbitration Time		200	ns	3.3
T_{on}	ARB Turn on Time	10	83	ns	3.3
T_{ds}	Arbitration Disable Time		26	ns	3.3
T_{en}	Arbitration Enable Time		26	ns	3.3
T_{su}	Arbitration Set up Time	31		ns	3.3
T_h	Hold Time	10		ns	3.3
T_{off}	Turn off Time	10	40	ns	3.3
$2T_{pd}$	NuBus Delay		17	ns	3.3
T_{pd}	NuBus Delay		8.5	ns	3.3

7 MECHANICAL SPECIFICATIONS

Mechanically, the NuBus derives from IEC specifications for connectors (IEC 603), boards (IEC 297 SC 480), and mechanical structures called "Eurocards". This section details the particular version of the IEC family used by the NuBus and any variations to the IEC specifications.

7.1 Boards

NuBus boards are a subset of the general IEC recommendations. For consistency, they are defined to be a single size without any options or variations. Similarly, the mounting provisions are very specific so that any NuBus board will fit into any NuBus chassis.

7.1.1 Board Size

NuBus boards correspond to triple height IEC boards, 9U high where U = 44.45mm (1.75 in), the standard IEC unit of height. Cards are triple IEC depth, 280mm (11.024 in). Tolerances are +0.0/−0.3mm (+0.000/−0.012 in). The dimensions and layout of a board are shown in Figure 7−1.

7.1.2 Connectors

Three 603-2-IEC-C096-M connectors are mounted in the standard IEC positions as shown in Figure 7−1. The connectors are referred to as P1, P2 and P3 with P1 at the top and P3 at the bottom. The NuBus is defined on P1. P2 and P3 must be 603-2-IEC-C096-M connectors. Only ground pins and +5 pins are recommended. Other pins are user definable. Refer to Paragraph 7.3, Pin Assignments.

7.1.3 Component Height

The board top component height must be less than 13.97mm (0.55 in). No component or lead should extend beyond the board bottom more than 2.54mm (0.10 in).

7.1.4 Board Thickness

In the area of the card guides, that is, within 2.5mm (0.098 in) of the top and bottom edge, NuBus cards should be 1.6mm ± 0.2mm (0.063 in ±0.0008 in) thick.

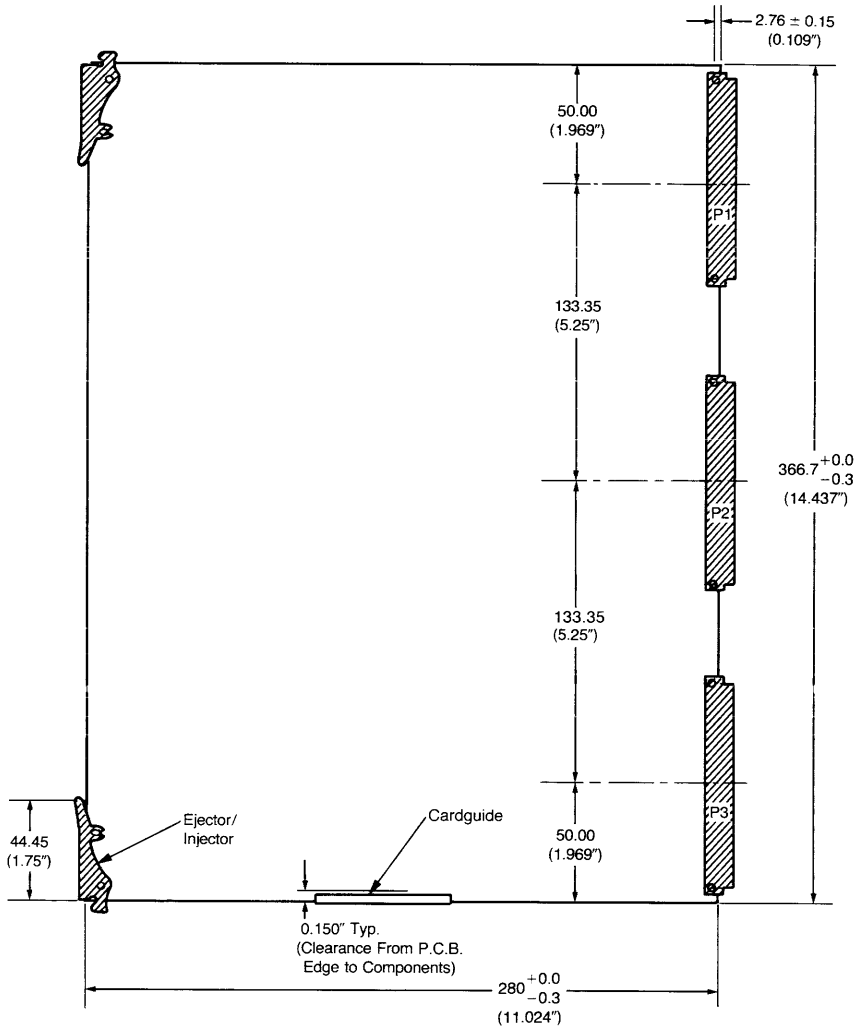


Figure 7–1. NuBus Board.

7.1.5 Working Area

Components may be placed on any part of the board except within 2.5mm (0.098 in) of the top and bottom edge and where they would interfere with card ejectors, stiffeners or connectors.

7.1.6 Board Warpage and Stiffness

Provisions should be made on the board with stiffeners or other means so that no part of the board deviates by more than 1.27mm (0.05 in) from the ideal plane.

7.1.7 Board Ejector/Injector

Unlike the IEC specifications, NuBus boards have one specific form of ejector/injector. These devices are mounted on the front edge of the board at the top and bottom as shown in Figure 7–1 with pins pressed into the board. The device itself should have the dimensions shown in Figure 7–2.

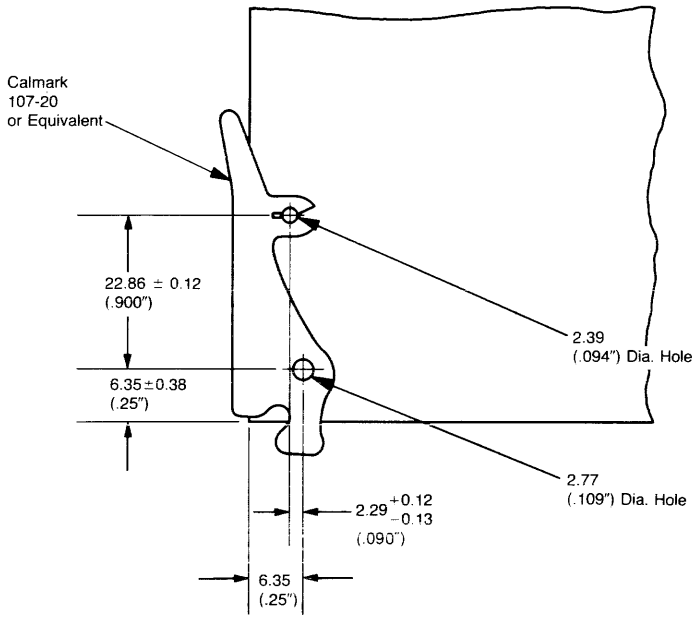


Figure 7–2. NuBus Board Ejector/Injector.

7.1.8 Malfunction Indicator LED

Boards should have a malfunction indicator light consisting of a red LED. When this light is on, it should indicate that the board is defective in some way and should receive attention from service personnel. The light may turn on temporarily after power up while the board is being checked for correct operation.

7.2 Card Cages

The NuBus places few constraints on the card cage. Basically, the cage should seat the boards and not violate the electrical requirements of the backplane. Obviously, provisions should be made for suitable mechanical stability, adequate cooling, etc.

7.2.1 Backplanes

The cardcage must support a backplane with 603-2-IEC-C096-F connectors mounted in the P1 position for the male board connectors to mate with.

7.2.2 Intercard Spacing

The card cage should space boards at least 20.32mm (0.80 in) apart.

7.2.3 Card Guides

Card guides should support the NuBus board thickness of 1.6mm ±0.2mm (0.063 in ±0.0008 in). They should not be more than 2.5mm (0.098 in) high. Card guides should not extend far enough forward on the board to interfere with the ejector/injectors.

7.2.4 Card Cage Lip

The required dimensions of the card cage lip are shown in Figure 7–3. The card cage lip should be 289.3 mm (11.4 in) from the plane of the backplane where the P1 connectors reside.

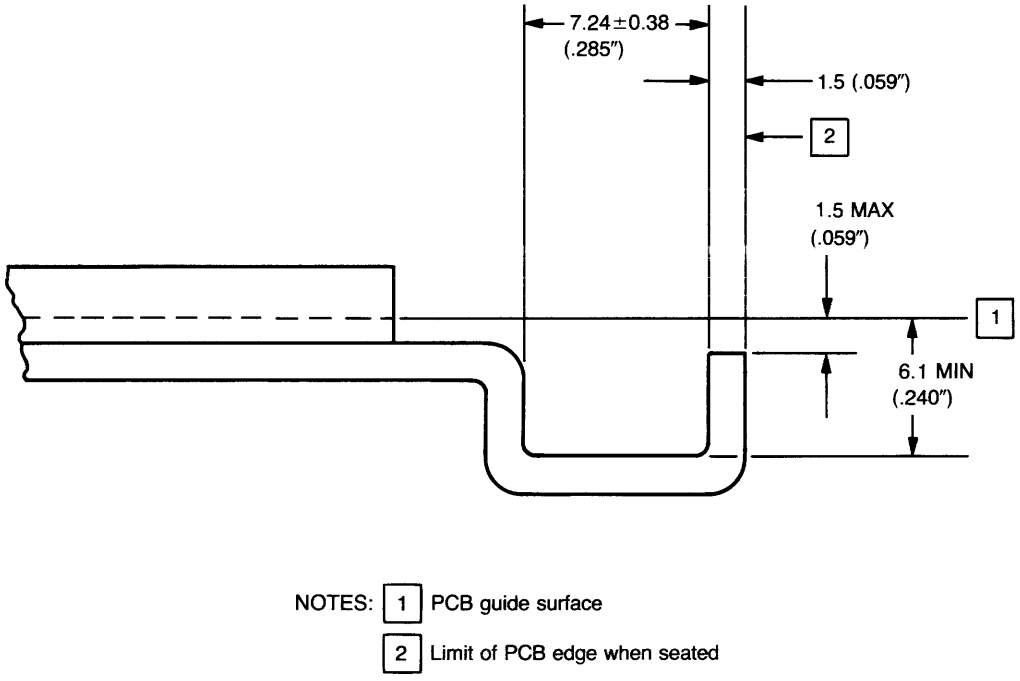


Figure 7–3. Required Card Cage Lip Dimensions.

7.3 Pin Assignments

The pin assignments for the P1 connector are listed in Table 7–1 and the recommended pin assignments for the P2 and P3 connectors are listed in Table 7–2.

Table 7–1. Signal Definitions and Pin Assignments (P1)

PIN/ROW	A	B	C
1	−12	−12	RESET/
2	GND	GND	GND
3	SPV/	GND	+5
4	SP/	+5	+5
5	TM1/	+5	TM0/
6	AD1/	+5	AD0/
7	AD3/	+5	AD2/
8	AD5/	−5	AD4/
9	AD7/	−5	AD6/
10	AD9/	−5	AD8/
11	AD11/	−5	AD10/
12	AD13/	GND	AD12/
13	AD15/	GND	AD14/
14	AD17/	GND	AD16/
15	AD19/	GND	AD18/
16	AD21/	GND	AD20/
17	AD23/	GND	AD22/
18	AD25/	GND	AD24/
19	AD27/	GND	AD26/
20	AD29/	GND	AD28/
21	AD31/	GND	AD30/
22	GND	GND	GND
23	GND	GND	RSVD/
24	ARB1/	−5	ARB0/
25	ARB3/	−5	ARB2/
26	ID1/	−5	ID0/
27	ID3/	−5	ID2/
28	ACK/	+5	START/
29	+5	+5	+5
30	RQST/	GND	+5
31	GND	GND	GND
32	+12	+12	CLK/

NOTE — Connector is 603-2-IEC-C096-F type. Row b of connector is optional on boards, allowing 64 pin subset for boards with minimum power requirements.

Table 7–2. Recommended +5V and GND Pinout on P2 and P3 Connectors

PIN/ROW	A	B	C
1	–	–	–
2	–	GND*	–
3	–	GND*	–
4	–	–	–
5	–	+5	–
6	–	+5	–
7	–	+5	–
8	–	–	–
9	–	–	–
10	–	–	–
11	–	–	–
12	–	GND*	–
13	–	–	–
14	–	–	–
15	–	–	–
16	–	GND*	–
17	–	–	–
18	–	–	–
19	–	GND*	–
20	–	–	–
21	–	–	–
22	–	–	–
23	–	GND*	–
24	–	–	–
25	–	–	–
26	–	–	–
27	–	–	–
28	–	+5	–
29	–	–	–
30	–	GND*	–
31	–	GND*	–
32	–	–	–

*These pins should be grounded for EMI purposes even if not needed for power requirements of the board.

NEWS RELEASE H-703

TEXAS INSTRUMENTS
INCORPORATED

Press Contacts:
Sue Metzler (512) 250-7302
Reba Cardenas (713) 895-3133
DO NOT PUBLISH THESE NUMBERS

TI TO LICENSE NUBUS TECHNOLOGY

AUSTIN, TEXAS (March 16, 1984) -- Under an agreement with Massachusetts Institute of Technology (MIT), the Data Systems Group of Texas Instruments is now licencing its 32-bit NuBus™ technology to commercial and non-commercial users. Designed at MIT, NuBus is a 10MHZ synchronous bus that provides significant performance advantages over other commercially available 32-bit buses.

The advantages of TI's NuBus include a high 37.5 megabyte per second bandwidth and a truly processor independent, memory-mapped architecture that is optimized for 32-bit data transfer and multiprocessing applications. According to George White, TI's Nu Machine™ development manager, "The beauty of NuBus is the simplicity and flexibility of its design. It provides features required by high-performance systems such as directed interrupts, high-bandwidth, and a fair bus arbitration policy."

The NuBus specification has been defined and in use by customers for more than a year. Components for NuBus configurations such as wire-wrap boards are now available from third-parties, and complete systems based on the NuBus such as TI's Nu Machine began shipping in 1983. TI is developing integrated circuits to facilitate design of boards interfacing to NuBus.

-more-

The three-high Eurocard format utilized with NuBus adds to its flexibility. The card format uses three 96-pin DIN connectors per card. NuBus requires only one of these connectors leaving the user the flexibility of two uncommitted connectors for I/O or subsidiary buses. For example, TI's recently announced Nu Machine takes advantage of this flexibility by providing a bus converter between NuBus and a Multibus™ subsystem.

A major design advantage of NuBus architecture is its processor independence which makes it well-suited for multiprocessor systems and applications requiring special-purpose processors. According to Steve Ward, associate professor of computer science and engineering at MIT, "NuBus is designed to anticipate the communication needs of a wide-range of high-performance systems that will be emerging over the next decade."

The NuBus specification is now being considered as a standard by the IEEE P896 working group. This group, also known as the "futurebus committee", is working on standards for advanced 32-bit buses. A one-time $2000 license fee entitles the purchaser to a non-exclusive, royalty-free, commercial license to use NuBus technology; non-commercial licenses are available at a $200 fee. NuBus specifications are available from TI and additional information is available from MIT.

-30-

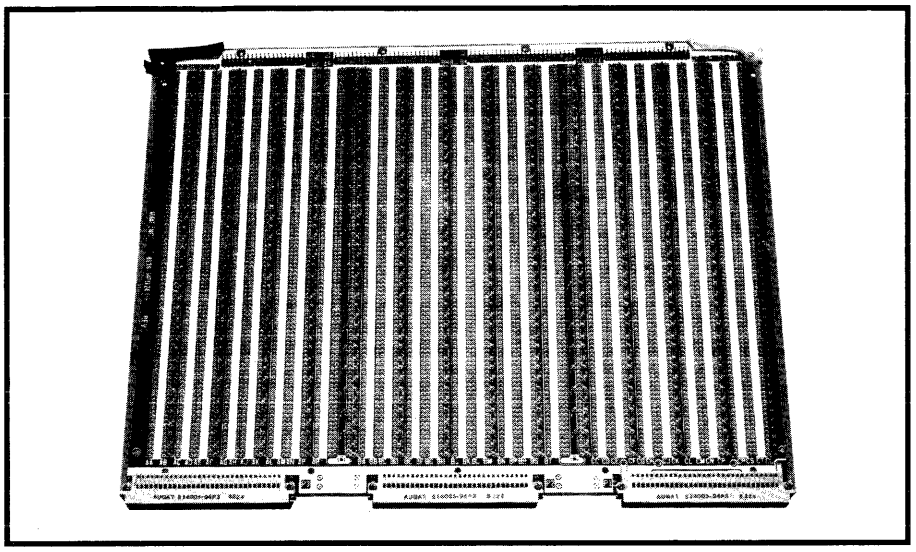
For more information, contact:

George White
Nu Machine Development Manager

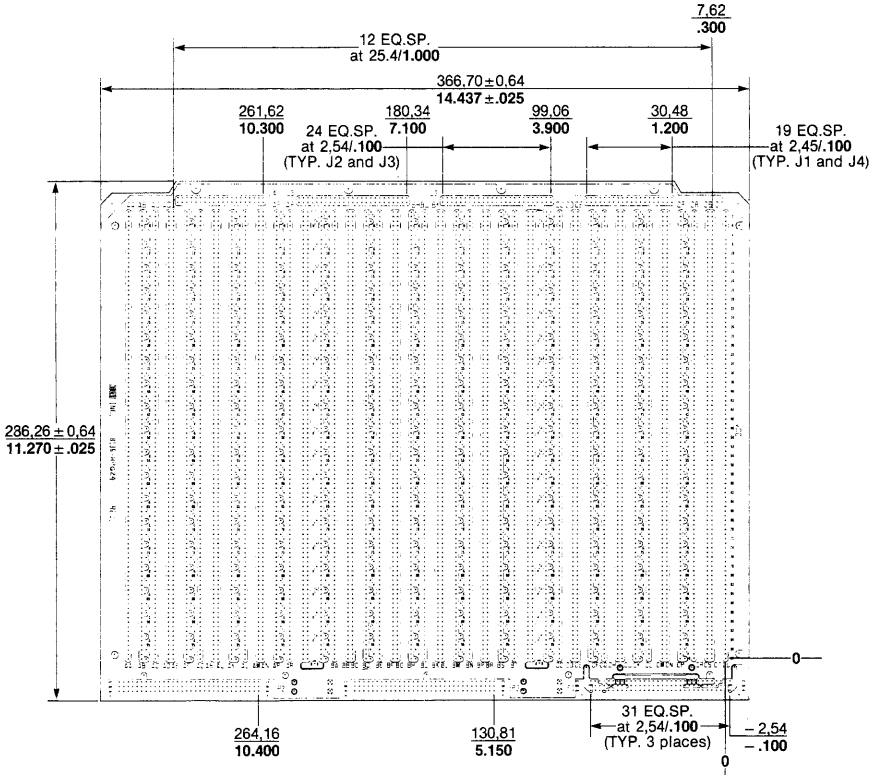
Texas Instruments:

Data Systems Group
17881 Cartwright Road
Irvine, California 92714
714 660-8205

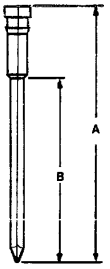
TEXAS INSTRUMENTS NuBus



8136-HPG24 SHOWN



PART NUMBER	CONTACT PLATING	SLEEVE PLATING	DIMENSIONS
8136-HPG24	GOLD	GOLD	A .703
8136-HPG24-T/G	GOLD	TIN	B .510
8136-HPG24-2	GOLD	GOLD	A .563
8136-HPG24-2T/G	GOLD	TIN	B .370



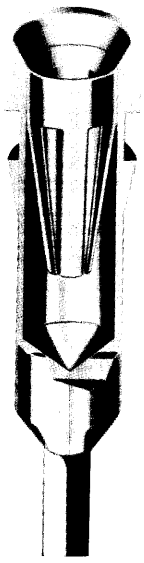
- **CAPACITY: 297 16-PIN DEVICES**
- **HI-PAK™ (High Packing Density)** Universal pattern allows devices with between-row spacing of .3", .4", .6", and .9" to be placed any where on the panel
- Pins on connectors P1 are committed per NuBus specifications.
- Pins on connectors P2 and P3 are uncommitted
- Front of panel (opposite) has feed thruPins for accepting flat cable. J1 and J4 have 40 Pins each, J3 and J4 have 50 Pins each.
- VCC and GND pins will accept radial lead capacitors with lead spacing of .3" or .2"
- Green solder mask with white nomenclature for easy readability
- Solder plate alongside columns allows installation of solder clips when required
- Two additional uncommitted voltage planes are provided between rows AT- AV and CA-CB on the compoment side.
- Injector / Ejector keys and stiffener allow for easy insertion and removal on the panel.

MATERIALS

Inner Contact: Machined beryllium copper per QQ-C-530, gold- over nickel-plated

Outer Sleeve: Machined brass per QQ-B-626, gold-over nickel-plated or tin-plate per MIL-T-10727

Board: Glass epoxy, 2 oz. copper with solder mask



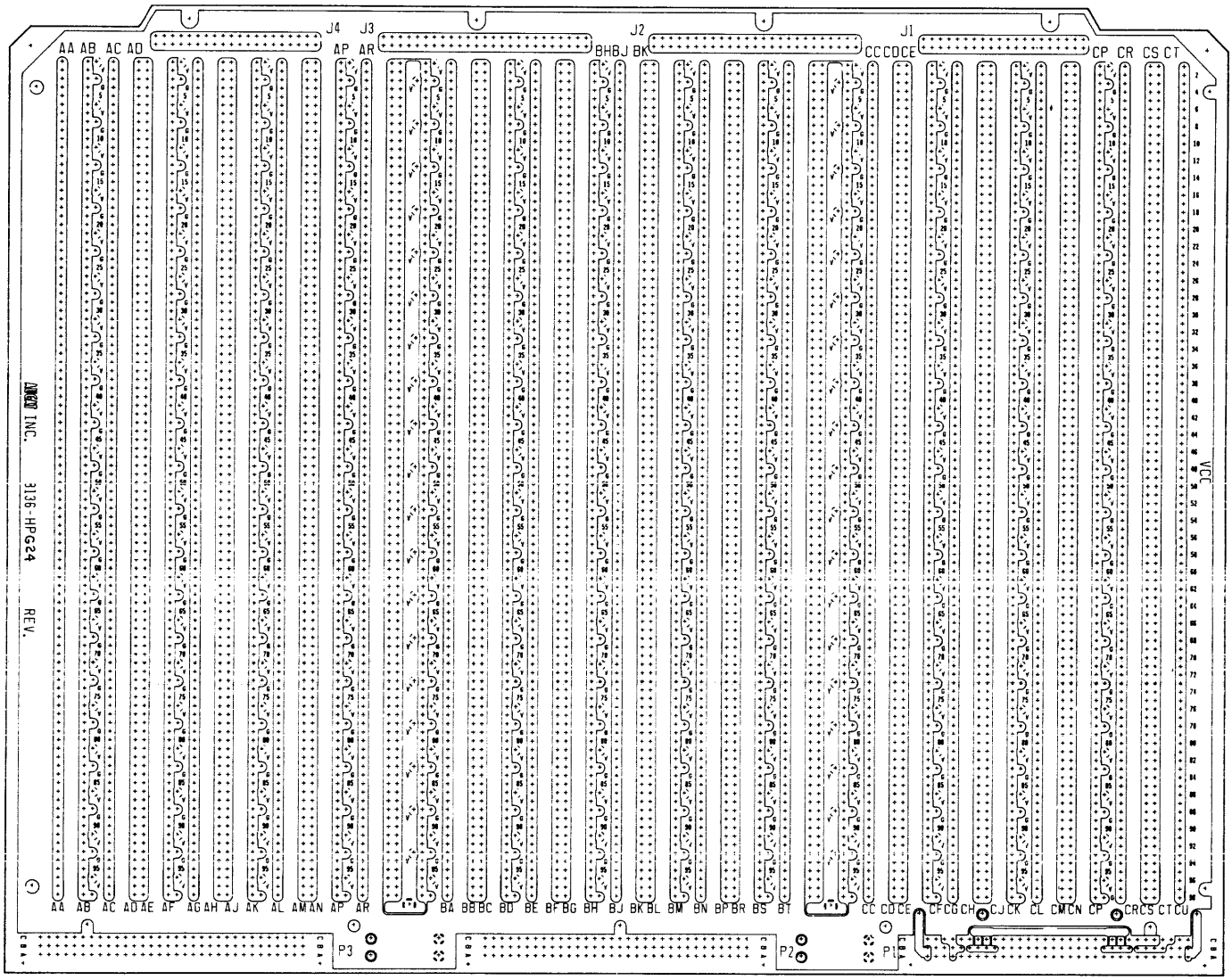
Augat's precision machined, four-fingered, beryllium copper internal contact makes Augat wire-wrap panels ideal for permanent packaging applications in hazardous environments, as well as prototyping applications

AUGAT® AUGAT INTERCONNECTION SYSTEMS DIVISION
40 Perry Avenue, P.O. Box 1037, Attleboro, Massachusetts 02703 (617) 222-2202 TWX 710-931-0644

S-2881

LAYOUT SHEET
for
TI NuBus
Augat Part Number 8136-HPG24

(not to scale)



COMPONENT SIDE

S-2881

THE WELL-ENGINEERED WORKSTATION

NU MACHINE

Unusual power. Self-diagnostic capability. Ease of use, reliability, and quiet operation. Ability to incorporate future microprocessor developments without architectural change. All innovatively engineered into a high-performance system which eliminates configuration and maintenance problems associated with previous computers.

Processor-Independent Bus

The Nu Machine is built around the truly microprocessor-independent, M.I.T.-conceived NuBus. With Nu Machine, multiple super-mini class CPU boards may be accommodated without changes to the computer's architecture.

An Actual Super-Mini Class CPU

The Nu Machine CPU board has a Motorola 68010 as its main computational unit but is surrounded with logic to create a super-mini class architecture. It has a 4K byte data cache of 45 nsec-access-time, static RAM. It also has a two-level page table in main memory with a "translate-look aside" buffer which is implemented by a hardware state machine on the CPU board. With Nu Machine, the microprocessor-based CPU is differentiated by its powerful MSI-implemented cache and memory management logic.

Autonomous System Diagnostic Unit

The Nu Machine System Diagnostic Unit (SDU) is an 8088-based single board computer. In order to test "robustness," the SDU can, under software control, margin the +5 volt power supply and the system clock. With Nu Machine, the autonomous SDU can test each element of the computer independent of all other functioning or nonfunctioning elements.

Simple diagnostics for other cards are in ROMs on those cards. More extensive diagnostics may also be loaded from disk, from the ¼-inch tape drive which interfaces directly to the SDU, or down-loaded over either of two serial ports.

Two Configurations Engineered to Meet the User's Needs

Two Nu Machine configurations bring advanced computer technology to the office and computer room:

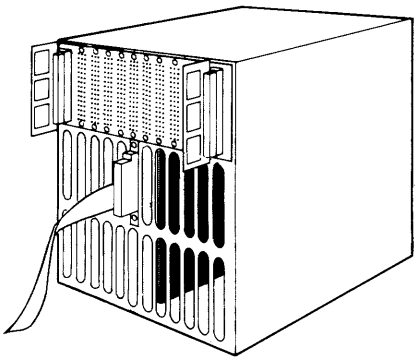
The Office Module with a 12-slot card cage supports 8-inch peripherals.

The Rack Module with a 21-slot card cage supports peripherals mounted in a standard 19-inch RETMA enclosure.

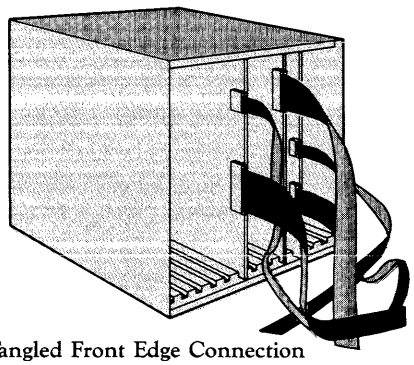
Chassis Features

No matter which Nu Machine configuration the user chooses, these important design features will stand out:

- I/O connection through the NuBus motherboard backplane eliminates card interference and cable tangle on the front of the cardcage.

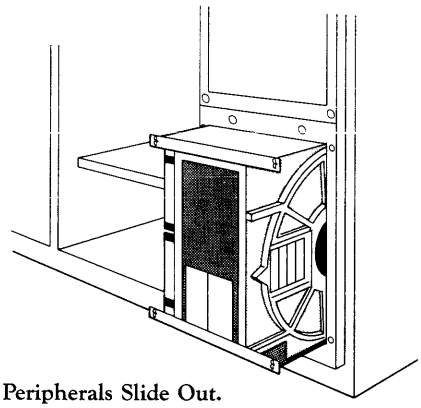


Nu Machine Cable Connection through Backplane Eliminates . . .



the Tangled Front Edge Connection

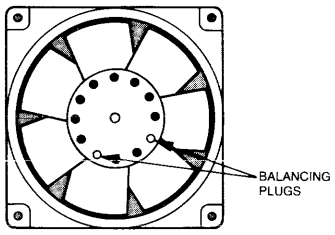
- Power supply and peripherals are mounted to plates which slide out for easy field maintenance and upgrading capabilities.



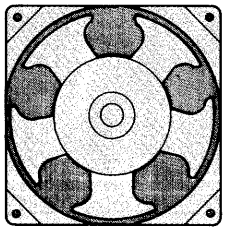
Peripherals Slide Out.

- Reliable DIN connectors are used for NuBus and I/O interconnection.
- With NuBus-supported autoconfiguration, "geographical" addressing by slot numbers is hardwired in the backplane. ROMs on all cards identify board type, serial number, and revision level. CMOS memory on the SDU gives setup data such as baud rates. These features eliminate the need for "DIP" switches and jumpers, simplifying maintenance.

- The quiet, aluminum fans are individually balanced for truer spin which extends lifespan.



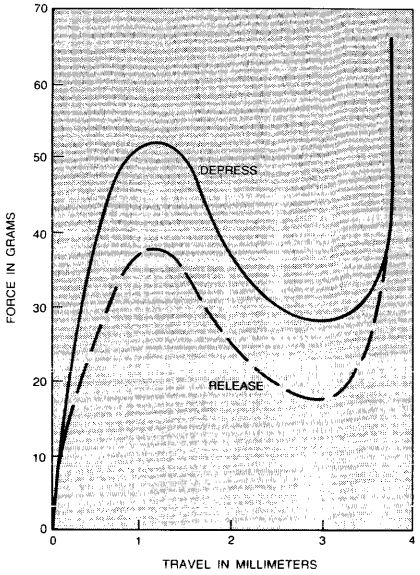
Quiet Nu Machine Fan Outlasts . . .



the Ordinary Fan.

Video Display Subsystem Features

- The quiet, low profile keyboard has infinite height adjustment from 5 to 15 degrees slope. For comfort in positioning, the keyboard attaches to the terminal with a coiled cord. The "tactile feel" keyboard, based on a force displacement curve, has been scientifically arranged for fast, easy operation.



Keyboard Force Displacement Curve

- The bit-mapped graphics display is 800 by 1024 pixels on a 15-inch monochrome screen refreshed at 60 Hz.

TEXAS INSTRUMENTS
17881 Cartwright Road
Irvine, California 92714
(714) 660-8205

JANUARY 12, 1984
AFTER DIVESTITURE: THE SPLINTERING OF BELL LABS/101
Network protocols and VLSI design: why they can't be separated/157
Tightly coupled microprocessors speed transaction processing/164
SIX DOLLARS A Mc GRAW-HILL PUBLICATION
Electronics

Windowing Unix-based system's 32-bit bus accepts future processors, Multibus cards

Designed to stay young over the long haul, the Nu Machine is processor-independent, thanks to the design of its bus. The latest in 16- or 32-bit processors should find a comfortable home in this box, within the generous limits of its 10-MHz (synchronous) 32-bit NuBus.

Developed at the Massachusetts Institute of Technology, the bus can move 37.5 megabytes/s, assuming 16 data words transferred for each bus cycle used to transfer an address word (addresses and data words travel the same data highway). The system's designers are attempting to bootstrap users into the 32-bit world by linking the NuBus intimately with the 16-bit Multibus. A user might start with a set of Multibus-based peripherals and an unaided Multibus processor and move to one or more 32-bit processors as his software becomes ready.

The Nu Machine's standard configurations, as offered by Texas Instruments, are built around a NuBus central-processing-unit card, however. Based on a 68010 processor, it has a 4-K-byte cache memory; a high-resolution display processor is on another NuBus card. Two versions differ only in cabinet size, disk-storage capacity, and the number of NuBus and Multibus slots.

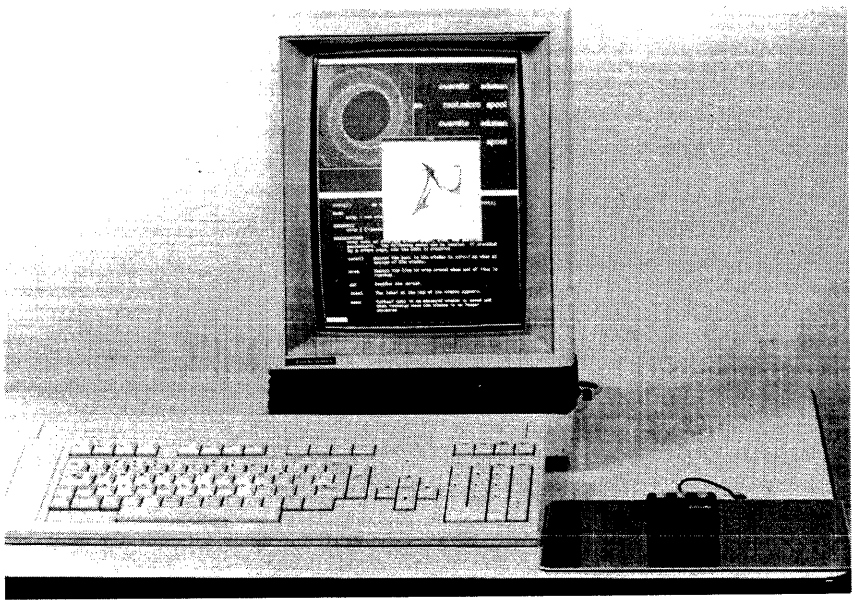
Overlap. In both versions, the Multibus and NuBus overlap physically: three slots can accommodate either Multibus or NuBus cards; one of the three, however, is occupied by the card that handles communications between the two buses, including input/output- and memory-space mapping. The NuBus lines are confined to a single 96-pin connector at one end of a long edge of the 11-by-14½-in. (triple-height Eurocard form-factor) boards, leaving room on the back panel for Multibus connections along the same edge.

The 12-slot version, a 26-in.-high unit designed to operate in a quiet office environment, has six NuBus slots, three Multibus slots, and three hybrid slots that can connect to both buses. The 21-slot cage (in a taller 19-in. rack-mounting enclosure) holds up to 16 NuBus cards, the design maximum, and up to eight Multibus cards—but not at the same time because, again, three slots are hybrid. In the NuBus slots, I/O lines can be connected to the rear edge of the card, which is convenient and makes available a quieter I/O-line ground connection.

The Nu Machine comes with an operating system derived directly from AT&T Bell Laboratories' Unix Version 7—enhanced to accommodate mouse-oriented windowing graphics. "The combination of high performance and flexible architecture makes the Nu Machine particularly attractive to sophisticated end users, systems integrators, and OEMs in the scientific and engineering marketplace," says Joe Watson, vice president of TI's Data Systems Group. Design of integrated circuits and printed-circuit boards, computer-aided engineering, mechanical design, simulation, and advanced software development are suggested as possible applications.

Chameleon. Indeed, the system's flexibility is underlined by the very different version offered by a different vendor. "One of our first Nu Machine customers is Lisp Machine Inc.," says George White, Nu Machine development manager. "They designed their own Lisp processor and chose the Nu Machine because of its high-resolution display coupled with the power and flexibility."

The result of a "happy marriage between academia and industry," as White puts it, the Nu Machine was brought to its present form by an Irvine, Calif., group that TI hired when it purchased a license to manufacture the Nu Machine from Western Digital Corp. [*Electronics*, Aug. 25, 1983, p. 41]. It boasts bit-mapped graphics of 800 by 1,024 picture elements on a 15-in. monochrome screen refreshed at 60 Hz; its 10-MHz 68010 processor comes with a 45-ns



cache of random-access memory on the same board. This configuration uses about one third of the bus's available bandwidth, according to White.

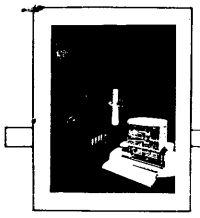
Smart boot. The system diagnostic unit, the card that houses the Multibus-to-NuBus–conversion hardware, is an independent 8088-based microsystem. It has 2-K bytes of battery-backed RAM, two RS-232-C interfaces, and an interface for a ¼-in. streaming cartridge tape drive. It bootstraps the system and performs a number of front-end, diagnostic, and other services.

The NuBus cards are addressed by slot position, rather than by jumpers or switches. The bus structure itself is built on a master/slave concept: for each transaction, a device takes control of the NuBus, becomes a master, and addresses another unit as its slave for that transaction. The slave device acts on the command that the master provides. A simple handshake protocol allows devices of different speeds to use the bus.

Multiple master modules may be connected for multiprocessing configurations, thanks to the NuBus's high bandwidth. Arbitration among the masters, says the firm, gives each processor an essentially equal share of the bus bandwidth. Any module can interrupt a processor by writing to an area of address space monitored by that processor. This technique eliminates interrupt lines and allows interrupts to be dynamically reassigned to different processors.

The office model, with 512-K bytes of memory, an 84-megabyte hard disk with a 20-ms average access time, ¼-in. cartridge-tape drive, display, keyboard, and mouse, is priced at $33,680 in quantity. A 21-slot rack-mountable configuration, similar save for a 474-megabyte disk with an 18-ms average access time, is priced at $50,370. The Nu Machine, which will be shown for the first time at the UniForum conference in Washington, D. C., Jan. 17–20, is available now.

Texas Instruments Inc., Data Systems Group, P. O. Box 402430, Dallas, Texas 75240 [Circle 339]



BUS STRUCTURE EASES MULTIPROCESSOR INTEGRATION

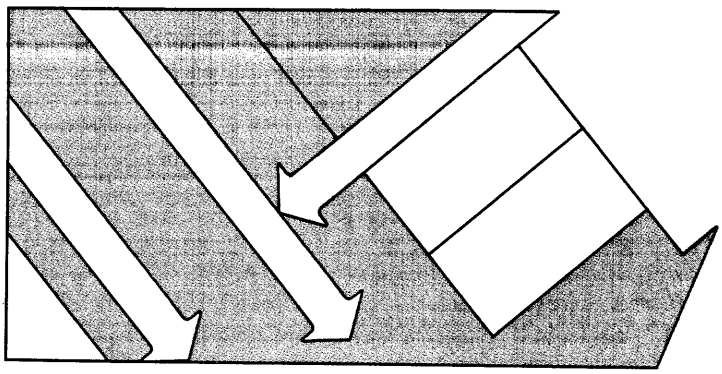
Use of a 32-bit NuBus in memory-mapped I/O and interrupt operations aids system configuration.

by George P. White

Where system integrators once needed to carefully account for processor cycles, the availability of low cost, high performance microprocessors is fostering a new approach to system design. Now, auxiliary processors handle cycle-hungry support functions such as graphics or numerical processing, thus freeing the CPU to direct its power directly to the user's application. Yet, because conventional architectures wrap system resources such as memory and I/O hardware tightly around a CPU core, traditional architectures still do not provide a sufficiently flexible framework for exploiting the cost and performance advantages of multiprocessor designs.

Originally developed at the Massachusetts Institute of Technology specifically for multiprocessor architectures, Texas Instruments' 32-bit NuBus provides system integrators with system architecture independence, high bandwidth, easy system configuration, a simple protocol, and small pin count. In fact, because the NuBus maintains a simple protocol for all bus operations, 49 signal lines are sufficient to handle all transactions for up to 16 different devices in the 4-Gbyte address space of the buses.

George P. White is manager of Nu Machine development at Texas Instruments, 17881 Cartwright Rd, Irvine, CA 92714. He holds a BS in electrical engineering from the Massachusetts Institute of Technology.



The NuBus accomplishes this feat by supporting only read/write transactions. Unlike conventional bus structures, which require separate signals for memory access, interrupts, and I/O operations, the NuBus includes all these operations under its umbrella of read/write transactions. I/O and interrupt operations are mapped into the address space and handled just as accesses to system memory.

Furthermore, by laying system resources within this 4-Gbyte memory address space, the NuBus decouples the logical function of system resources from the physical implementation of system hardware. Instead of altering hardware switches and jumpers, engineers can reconfigure systems simply by changing variables in the NuBus address space. Consequently, without disturbing the logical architecture, system integrators are free to modify a system's physical framework to achieve the required balance between system cost and performance.

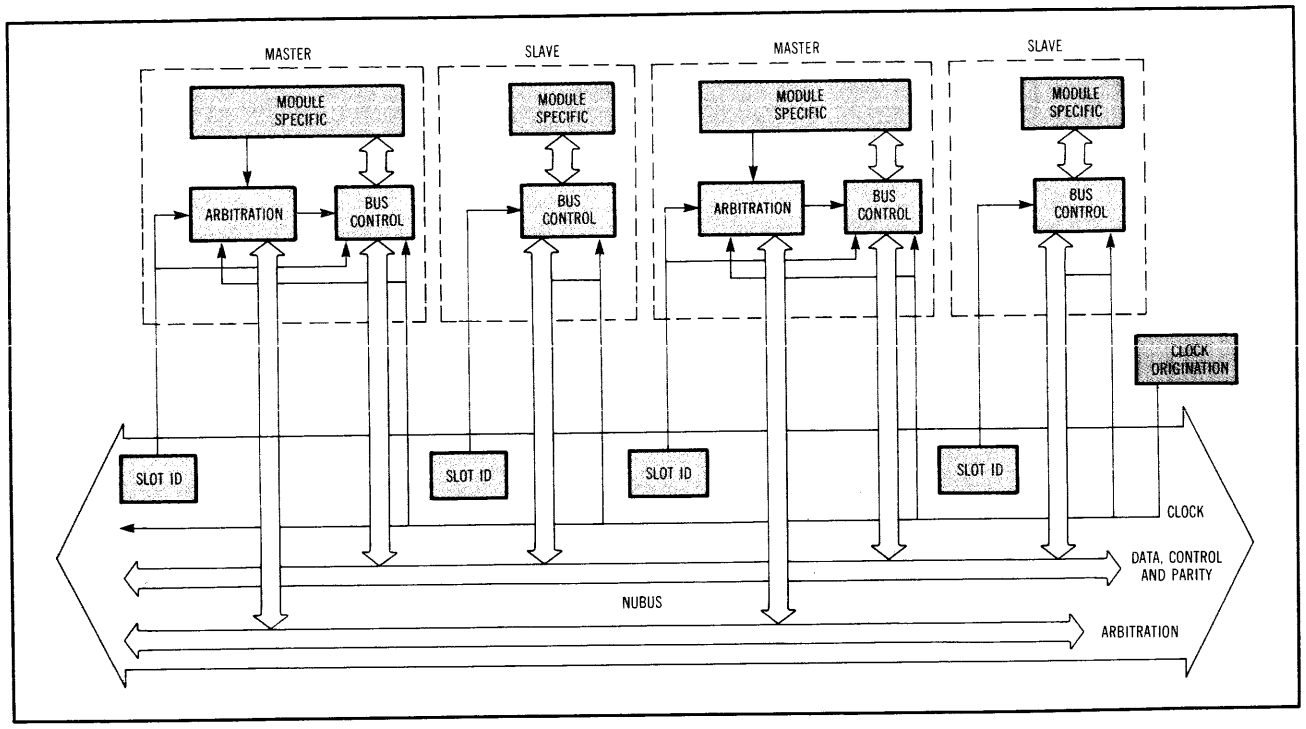


Fig 1 Identified by a unique value wired into the backplane, each module capable of participating in the arbitration mechanism is free to serve as either bus master or slave in the NuBus. A 10-MHz system clock synchronizes this 32-bit multiplexed bus to provide up to a 37.5-Mbyte/s transfer rate in block mode.

Accommodating I/O and interrupt operations within this framework provides good mapping between bus operations and the structures in memory, which can be manipulated by high level programming languages. By uniformly dealing with all resources in the address space, even a high level language such as Fortran can affect I/O and interrupt operations just by transferring a 32-bit number to some specified memory location.

TABLE 1

NuBus Bus Definitions

Classification	Signal	No. of Pins
Utility	$\overline{\text{RESET}}$	1
	$\overline{\text{CLK}}$	1
Control	$\overline{\text{START}}$	1
	$\overline{\text{ACK}}$	1
	TM0	1
	TM1	1
Addressdata	$\overline{\text{AD}}$<31..0>	32
Arbitration	$\overline{\text{ARB}}$<3..0>	4
	$\overline{\text{RQST}}$	1
Parity	$\overline{\text{SP}}$	1
	$\overline{\text{SPV}}$	1
Slot ID	$\overline{\text{ID}}$<3..0>	4
	Total signals	=49
Powerground	+5	11
	−5	8
	+12	2
	−12	2
	$\overline{\text{GND}}$	23
Reserved	$\overline{\text{RSVD}}$	1
	Total pin count	=96

Furthermore, uniform memory of the NuBus model aids device access to memory at a lower level. The small address space supported in earlier bus structures (eg, Multibus) could force programmers to deal differently with memory, depending on the source of the memory reference. For example, a CPU would access its onboard local memory with one set of addresses, while references to the same memory originating from other boards would need to incorporate an offset into that set of memory. With its single memory map, the NuBus permits any device to access any memory—even that local to a particular board—with a consistent set of addresses.

Simple structure

Driven by a central system clock, the NuBus is a synchronized bus that provides designers with a framework free from the specific control structure of a particular microprocessor. In fact, the NuBus specification imposes few constraints on system design—any module that can arbitrate for the NuBus can potentially serve as bus master (Fig 1).

No particular slot position is defined as the bus master. Instead, each slot has an identification hardwired into the backplane. Consequently, boards can be differentiated without the need for jumpers or switches. Besides the signals shown in Fig 1, the only other signals required by the NuBus are reset and power (Table 1). Although adopting a triple-height Eurocard form, the NuBus specifies use of only a single 96-pin connector and uses only 49 of those for signals—relegating the rest as extra power and ground lines.

To the system integrator, this low pin count benefits both present and future designs. Having fewer interconnection pins in current designs translates into more mechanically reliable systems. On the other hand, future systems can easily migrate to designs using VLSI bus interface chips, which demand low pin count for low cost designs.

With its 10-MHz cycle, the NuBus supports a 37.5-Mbyte/s block transfer rate across 32 multiplexed address and data lines. Although optimized for 32-bit transfers, the NuBus also supports unjustified 8-bit byte and 16-bit half-word transactions. In contrast, a justified bus like Multibus II always places 16-bit data in the least significant 16 lines, even if the address specifies data in a more significant position in a word.

A justified bus structure permits designers to attach 8- or 16-bit interfaces to a common set of lines, but at the cost of a more complex bus structure. Moreover, the NuBus's unjustified structure results in a simpler organization at the small cost of a few more transceivers.

Bus transactions

Each 100-ns NuBus cycle ensures that signals settle along the NuBus before receivers latch in the values (Fig 2). With the rising edge of the cycle, drivers assert (low) or deassert (high) signals on NuBus lines. After signals settle during the 75-ns (unasserted) portion of the bus cycle, receivers sample the signals on the falling edge of the clock signal. The 25-ns (asserted) portion of the signal helps avoid skew in bus signals.

All basic NuBus signals require a clock cycle. In addition, various NuBus signals combine to form higher level transactions, like those for read/write operations. In fact, the NuBus permits these transactions to be a variable number of clock cycles long. Consequently, although it is a synchronous bus controlled by a central system clock, the NuBus provides the adaptability of an asynchronous bus without losing the design simplicity of a synchronous bus.

All NuBus transactions involve a dialogue between a device requesting service (master) and a device providing service (slave). Unlike traditional systems where a bus slave is a device incapable of independent action, the NuBus master/slave concept simply describes the temporary relationship between two NuBus modules during a particular bus transaction. In fact, provided that it can arbitrate for the NuBus, any module can serve equally well as slave or master in the NuBus protocol.

In the NuBus, a transaction commences with a START signal from a bus master and terminates with an acknowledge (ACK) signal from a slave. In parallel with toggling the START signal, the master notifies the slave of the type of transaction by manipulating transfer mode (TM) and the lower 2 bits of the AD lines. Thus, with these four control signals, the master can initiate read/write transactions involving bytes, half-words, or words (Table 2).

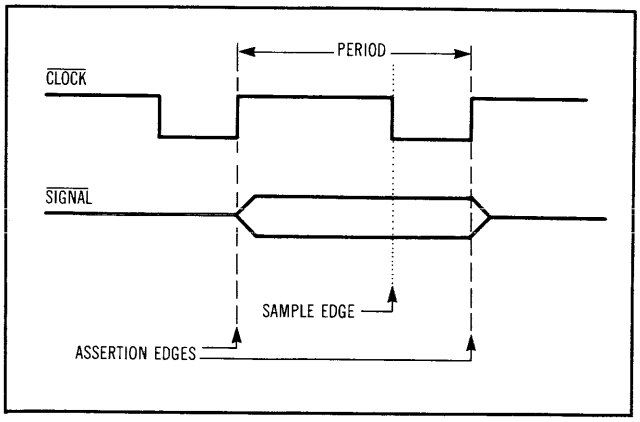


Fig 2 Each NuBus clock cycle lasts 100 ns. During the rising edge, devices place signals on the NuBus. After allowing 75 ns for these signals to settle, receivers latch in data during the falling edge of the clock signal. A 25-ns sample period helps avoid bus skew problems.

Transactions across the multiplexed NuBus include separate phases for address and data. For example, in the initial phase of a read transaction, a bus master such as a CPU sets the address lines, asserts the TM control signals to indicate the type of transaction, and toggles the START line (Fig 3). When it has prepared its response, the slave replies in the latter phase of the transaction by placing the data on the AD lines, indicting the status of its response on the TM lines, and asserting ACK.

On the other hand, in a write operation, the bus master first places the address on the AD lines and toggles the TM and START control lines. In the next cycle, the bus master places the data to be written on the AD lines and waits. After it has latched the data, the addressed slave acknowledges and indicates the status of the completed transaction by setting the TM lines.

TABLE 2
Read/Write Transactions

$\overline{TM1}$	$\overline{TM0}$	$\overline{AD1}$	$\overline{AD0}$	Type of Cycle
low	low	low	low	write byte 3
low	low	low	high	write byte 2
low	low	high	low	write byte 1
low	low	high	high	write byte 0
low	high	low	low	write half-word 1
low	high	low	high	write, block
low	high	high	low	write half-word 0
low	high	high	high	write word
high	low	low	low	read byte 3
high	low	low	high	read byte 2
high	low	high	low	read byte 1
high	low	high	high	read byte 0
high	high	low	low	read half-word 1
high	high	low	high	read, block
high	high	high	low	read half-word 0
high	high	high	high	read word

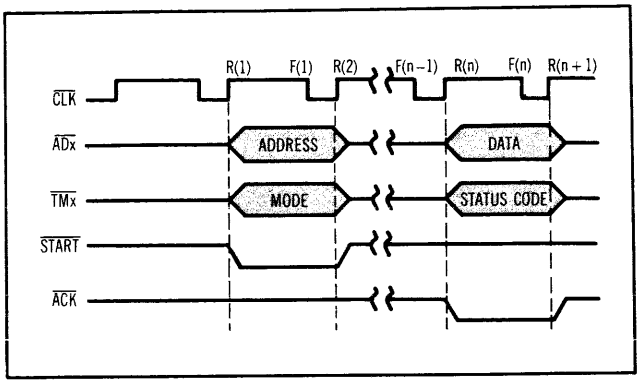


Fig 3 During a NuBus read transaction, a bus master places the address of the desired data on the bus, identifies the transfer mode (TM), and toggles the START line. When the addressed slave is ready to respond, it places the data on the address/data lines, indicates the status, and toggles the acknowledge (ACK) line.

Just as the AD lines double for address and data, the TM lines indicate transaction type during the address phase and the status of the result during the data phase. Of the four possible result codes, two results—bus transfer complete and error—correspond to positive and negative ACK codes commonly found in bus architectures.

Similarly, the third result code—bus timeout errors—is a signal used to guard against transactions targeted for nonexistent memory locations. For example, a NuBus master might initiate a transaction into the area of memory occupied by a certain module and receive a bus timeout code (from a separate logic). This indicates that the required module has been functionally removed from the NuBus.

The fourth result code—try-again-later—is a unique signal that should find a major role in multiprocessor systems. Logically indicating a result falling between error and success, the try-again-later code indicates that the master should simply defer the transaction to a later time, rather than jump into extensive exception-handling routines. This signal can find extensive applications in situations where a device serves more than one master. In dual-ported memory, for example, when a master finds an access blocked because of contention with another device using the memory, the master can sit on the bus—preventing its use by other potential bus masters—or release the bus and try again later. The NuBus try-again-later signal provides a mechanism to implement the latter, more efficient method.

Besides potentially improving bus throughput in multiprocessor systems, the try-again-later signal fills a critical need in avoiding deadlock in these systems. For example, in the Nu Machine, a separate 8088-based module acts as a converter between the NuBus and the Multibus (see Panel, "The key role of the diagnostic unit"). But, the use of such a converter can easily result in deadlock if the converter tries to access the NuBus at the same time that a NuBus module tries to access the converter. When this happens, both the converter and the NuBus master may find themselves deadlocked waiting for access to the other. The converter, however, relieves the potential deadlock by simply transmitting a try-again-later signal to the NuBus master and completing its own operation.

In addition to its role in boosting bus efficiency and mediating deadlock possibilities, the try-again-later signal can be used as a prefetch signal to slow devices. Moreover, in a bus converter to some slow bus, (rather than holding up a high speed bus like the NuBus), a converter can transmit a try-again-later signal to free the high speed bus and simultaneously access the information from the slower bus. In this way, the converter has the desired data available immediately upon the original requester's return.

Block transfers

Besides simple read/write transactions, the NuBus also supports block-mode transfers. Although some bus protocols such as Multibus II permit block-mode transfers of any length, the NuBus supports transfers only of smaller blocks—2, 4, 8, and 16 words. More compact block transfers of this sort

The key role of the diagnostic unit

Designed specifically for multiprocessor architectures, the NuBus already stars as central performer in Texas Instruments' Nu Machine. Equipped with a 68000-based CPU, the Nu Machine is designed to be independent of any particular CPU. Although such a processor-independent architecture provides system integrators with a flexible framework for crafting their own systems, it also demands an alternate approach for ensuring basic system operation in the absence of any particular CPU.

Filling this maintenance role as well as illustrating the use of the NuBus in a simple multi-CPU design, is the Nu Machine's system diagnostic unit (SDU). This unit is a separate 8088-based module with onboard memory, serial I/O for console communications, and maintenance and diagnostic test stored in onboard ROM. When the system is powered up, the SDU tests the NuBus through a series of bus transfers, identifies all boards in the system, initiates self-test routines associated with each board in the system, and signals the operator.

In addition to this maintenance work, the SDU becomes another potential master on the NuBus, acting as a converter between the NuBus and Multibus, once the system's integrity is ensured. In normal operation, the NuBus and Multibus system can operate independently. The Multibus appears as a 1-Mbyte window within the SDU's address space.

When a NuBus master addresses a memory location falling in this Multibus window, hardware-mapping logic converts the NuBus access into a Multibus reference without intervention of the 8088 CPU at bus speeds. On the other side of the window, the converter monitors each Multibus cycle for references to addresses which are mapped to NuBus. When a conversion is required, the SDU uses a Multi-to-NuBus page map to construct references to the NuBus address space.

TABLE 3

Block Length and Destination Address

$\overline{AD5}$	$\overline{AD4}$	$\overline{AD3}$	$\overline{AD2}$	Block Size Words	Block Starting Address
don't care	don't care	don't care	high	2	(AD31 to AD3) 000
don't care	don't care	high	low	4	(AD31 to AD4) 0000
don't care	high	low	low	8	(AD31 to AD5) 00000
high	low	low	low	16	(AD31 to AD6) 000000

conform more closely to the fundamental concept that a bus is a data-transfer highway freely available to any potential bus master. Furthermore, transferring arbitrarily long blocks requires a mechanism to suspend, or preempt, the transfer. This results in a more complex structure.

In addition, unlike other protocols, the master warns the slave that it is going to transfer a block and supplies the transfer length at the beginning of the operation. Supplying this information at the beginning opens the possibility for higher performance response. For example, if it is supplied with the size of the transfer beforehand, a slave has the opportunity to speed the transfer by prefetching the requested data from its storage.

By setting address bits AD2 to AD5, the bus master indicates the length of the block in the first phase of the transaction, as well as the destination address of the block (Table 3). In subsequent phases of the transaction, the bus master transmits (in block write) or receives (in block read) successive words from memory until the requested amount of words has been transmitted or an error occurs.

As each word within a block transfer is read or written, the slave uses TM0 as an intermediate ACK. The slave sends the ACK only after the final word in the block transfer has been transmitted.

One for all

Unlike other bus architectures, read and write serves for all operations on the NuBus, including I/O and interrupts, because I/O and interrupts are mapped into the NuBus's 4-Gbyte address space. In fact, all devices occupy a reserved portion of the NuBus address space specified by each slot's ID (Fig 4). Thus, the device that occupies slot 0 may occupy addresses between F0000000 to F0FFFFFF.

Originally popularized by Digital Equipment Corp's PDP-11, memory-mapped I/O operations write to memory addresses instead of using special I/O instructions or wires. Instead of a memory cell, the specified location contains a universal asynchronous receiver/transmitter command register. As a result, high level languages gain the ability to deal with I/O directly. Furthermore, the same memory management schemes that translate memory references and isolate system memory from application programs, now apply to I/O. Consequently, application programs can safely draw on a subset of system resources directly without concern that users might intrude on sensitive areas.

In the NuBus, a similar situation applies to interrupts. To initiate an interrupt in conventional systems, a device asserts a special line that runs to the CPU. When the CPU acknowledges the interrupt, the device replies with some identifying code that the CPU uses to enter an interrupt software routine. These conventional systems need only deal with the problem of detecting the source of an interrupt.

On the other hand, a multi-CPU system not only needs to specify the source of an interrupt, but must also be able to post an interrupt to a specific device. Thus, the NuBus maps interrupts into its address space so that devices can direct interrupt requests to specific devices.

Memory-mapped interrupts become particularly important in systems that can support multiple bus masters. Where a conventional approach would require a separate wire for each potential bus master, the NuBus approach provides a more flexible, less hardware-dependent approach.

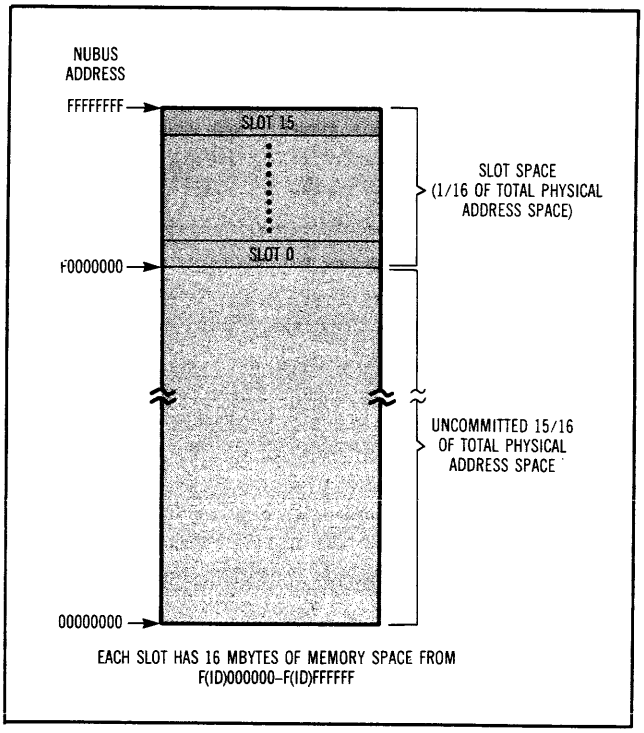


Fig 4 The NuBus associates each of its 16 slots with a reserved portion of its 4-Gbyte address space. Each slot spans an address range from F(ID) 000000 to F(ID) FFFFFF. Thus, slot 0 fills addresses from F0000000 to F0FFFFFF.

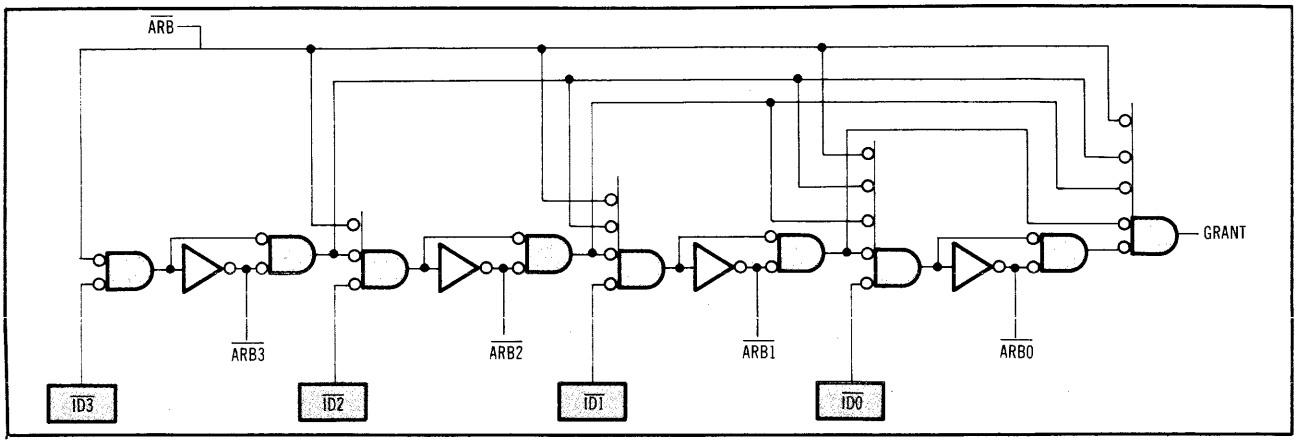


Fig 5 Simple combinatorial logic is sufficient to implement the NuBus arbitration (ARB) logic. The $\overline{ARB}x$ lines are common to all modules, while the $\overline{ID}x$ inputs are unique to each card. The $\overline{ARB}$ signal is asserted if the module is requesting the bus, while the grant signal indicates whether the $\overline{ARB}x$ lines match the slot's $\overline{ID}x$ lines.

Furthermore, as separate system resources evolve into more powerful devices, memory-mapped interrupts provide a graceful migration path. For example, devices such as disk controllers are gaining more independence from the CPU. After transferring a block of data between memory and disk, a smart controller on the NuBus can write its status word to a special location, interrupting the CPU for further action. The NuBus offers system integrators a simple mechanism to take advantage of such powerful techniques.

Using the NuBus's interrupt structures, programmers can easily implement advanced features like coroutines, in which a high level language issues an interrupt to another process. Similarly, software-configured interrupts like this aid design of systems requiring rapid response, such as in realtime environments. Here, a system can use two levels of interrupts to handle external events. When an external event causes an interrupt, the CPU can quickly perform some critical task (eg, removing a value from an input hardware register), and dispatch a lower priority interrupt to itself to handle any subsequent processing associated with the external event.

In the Nu Machine's 68000-based CPU board, a 256-word memory block serves as the interrupt area. Although the lower 32 words are not used because the 68000 uses only seven hardware interrupts, each priority level falls into a corresponding 32-word area. Special hardware logic on the CPU board monitors these words, and when it detects a write operation into any of these 32 words, the logic initiates the appropriate hardware interrupt procedure supported by the 68000. Software-interrupt routines can then use normal procedures to detect the source of the interrupt, including using the stored value as a vector to a particular device handler, or using the stored value as input to a particular service routine.

In addition to providing a consistent approach for handling I/O and interrupts, the memory-mapped approach adopted by the NuBus permits system integrators to rely on software to configure systems. In the Nu Machine, each hardware module uses a special set of memory locations within its reserved memory to specify configuration information. Thus, during a software configuration phase, an operating system can simply access these locations to obtain necessary system information. If a module is not present, the bus timeout that results during the NuBus transaction tells the operating system that the corresponding device is unavailable and should not be included in the system definition.

Critical arbitration

In any bus-oriented system with more than one potential master, the concept of arbitration is crucial. Multiprocessor systems such as the Nu Machine rely on bus arbitration mechanisms ensuring that each processor is allowed access to the bus regardless of its defined priority. The NuBus, optimized for multiprocessor architectures, provides a fair arbiration mechanism that guarantees access to any module requesting bus access.

The NuBus traces its arbitration lineage through a long history of bus protocols extending back to the S-100 bus. In the daisy chain arbitration technique, a signal propagates serially through all boards in the backplane. If it decides that it wants to assume control of the bus, a board simply does not propagate the signal.

Unfortunately, this approach does not provide a fair distribution of bus cycles. Boards closer to the daisy chain origin stand a better chance of acquiring the signal and of starving lower priority boards of bus access. Furthermore, this approach requires that all slots on the backplane be filled or bypassed with jumper cables. This is often at the cost of mechanical difficulties as jumpers fall off or users connect incorrect pins.

In the analogous software situation, scheduling algorithms can avoid starvation of lower priority tasks caught in a mix of higher priority processes. However, arbitration demands strict mechanisms integrated into the basic structure of the bus itself. The

mechanism cannot provide exceptions. During multiprocessor system design, an engineer cannot prejudice the architecture with the idea that a certain potential master should be allowed more bus cycles than others.

For example, when compared with a disk controller, an Ethernet communication module might be considered a lower priority device and assigned a correspondingly lower hardware priority. However, the data transmitted through this communication module may assume major importance in the system. Consequently, a system integrator must be certain to disassociate hardware priority on the bus from importance. Conventional daisy chained systems force designers to associate hardware priority with importance. The NuBus, however, maintains a strict approach to fairness in arbitration. No single module can be weighted to enjoy more bus activity at the expense of others.

To do this, the NuBus adopts a parallel scheme. Bus arbitration begins when one or more potential bus masters assert the bus request (RQST) line, and each attempts to place its unique ID on four open-collector lines on the bus—the arbitration (ARB) lines. Each slot is given a unique identification by a 4-bit ID code hardwired into the etch of the backplane.

In placing ID codes on the ARB bus, the boards use a strategy common to other bus architectures that ensures that only the highest ID stays on the bus. If a board sees a bit that is of a higher order than its own bits as it puts its code on the bus, the board lifts its signal. For example, if the board with ID4 attempts to place its ID on the ARB lines (assert ARB2) and finds that ARB3 is already asserted—indicating that a higher ID is already on the ARB lines—it simply removes its signal.

Implemented with simple combinational logic (Fig 5), this technique dictates that only the highest ID remains on the ARB lines at the end of the two bus cycle arbitration contests. Because this arbitration contest occurs over separate lines in parallel with read/write transactions, this mechanism does not cut into bus throughput. In fact, because typical NuBus transactions require at least two bus cycle, a new bus master will be ready to initiate its own transactions when the previous bus master has completed its turn on the NuBus.

Determining fair arbitration

Although this arbitration scheme does seem to involve a priority aspect, fairness counteracts it. If three boards request the NuBus simultaneously, the highest numbered board wins. However, fairness ensures that the lower numbered pair of boards will gain access before that particular higher numbered winner gains the bus again.

Fairness in the NuBus is a simple protocol—once the RQST line is asserted, no other boards are allowed to request the bus. For example, if three boards want the bus at the same time, they will all assert the RQST line at the same time. The highest numbered board will win out, but the other two boards will still be asserting the RQST line. After the last board of the three uses the bus, the RQST line will become deasserted. This then starts another arbitration contest.

In addition to serving as the medium for bus request and achieving arbitration fairness, this single RQST line also mediates bus locking in the NuBus. Bus locking is typically used for indivisible test and set instructions used to implement semaphores for interprocess communication. In bus locking, once a device wins the bus, it simply continues to assert RQST and maintain its ID on the ARB lines. During the next arbitration, the same board will always win because, by the definition of fairness, its ID will always be the highest numbered in the current arbitration round. The NuBus needs no extra wire, mechanism, or state to accommodate bus locking. This mechanism is enfolded within the larger concept of fairness.

Please rate the value of this article to you by circling the appropriate number in the "Editorial Score Box" on the Inquiry Card.

High 716 *Average 717* *Low 718*

TEXAS INSTRUMENTS
17881 Cartwright Road
Irvine, California 92714
(714) 660-8205

DNAB014
2242825-0001*B