EK-DMB32-TD-001

DMB32

Technical Description

digital™

EK-DMB32-TD-001

DMB32

Technical Description

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, May 1986

Using Digital's networked computer systems, this book was produced electronically by the Media, Publishing and Design Services department in Reading, England.

Printed in U.S.A.

CONTENTS

APPENDIX A IC DESCRIPTIONS

APPENDIX B GATE ARRAYS

APPENDIX C GLOSSARY

FIGURES

TABLES

PREFACE

This document gives a functional and technical description of the DMB32 asynchronous/synchronous multiplexer. It contains information for field service support, and for anyone who needs to know, in detail, how the DMB32 works.

The manual is organized into three chapters plus appendixes.

Chapter 1	– General Description
Chapter 2	– Functional Description
Chapter 3	– Technical Description
Appendix A	– IC Descriptions
Appendix B	– Gate Arrays
Appendix C	– Glossary of Terms

The following is a list of related titles.

Document	**Number**
DMB32 User Guide	EK-DMB32-UG
DMB32 Field Maintenance Print Set	MP-01797-01

CHAPTER 1
GENERAL DESCRIPTION

1.1 SCOPE

Chapter 1 provides a general description of the main functions of the DMB32 module. This is followed by a hardware overview section which provides more information on the functions of the on-board devices. Two supporting block diagrams are included as an aid to understanding the descriptions.

1.2 GENERAL OVERVIEW

The DMB32 is an intelligent communications adapter for the VAXBI. The primary function of the DMB32 is to transfer data between its communications ports and other VAXBI nodes, by the methods described later in this chapter. The DMB32 communications ports consist of:

- One synchronous port handling data at up to 64 Kbits per second.
- Eight asynchronous ports each handling data asynchronously at up to 38.4 Kbits per second.
- One parallel printer port handling up to 1200 lines per minute.

The VAXBI host needs only to make a DMA buffer available, and to program the CSRs, as described in the *DMB32 Users Guide.* All other tasks are carried out by the firmware-driven on-board microprocessor which performs and manages all configuration and data transfer functions and instructs the relevant logic to do whatever tasks are required. The microprocessor will initiate and manage DMA transfers and will ensure that the selected communications interface protocols are observed.

Figure 1-1 shows a simplified block diagram of the DMB32 in which the three main functional areas are identified as follows:

1. Communications Interface
2. Control Logic
3. VAXBI Interface

Data flow is also indicated on Figure 1-1.

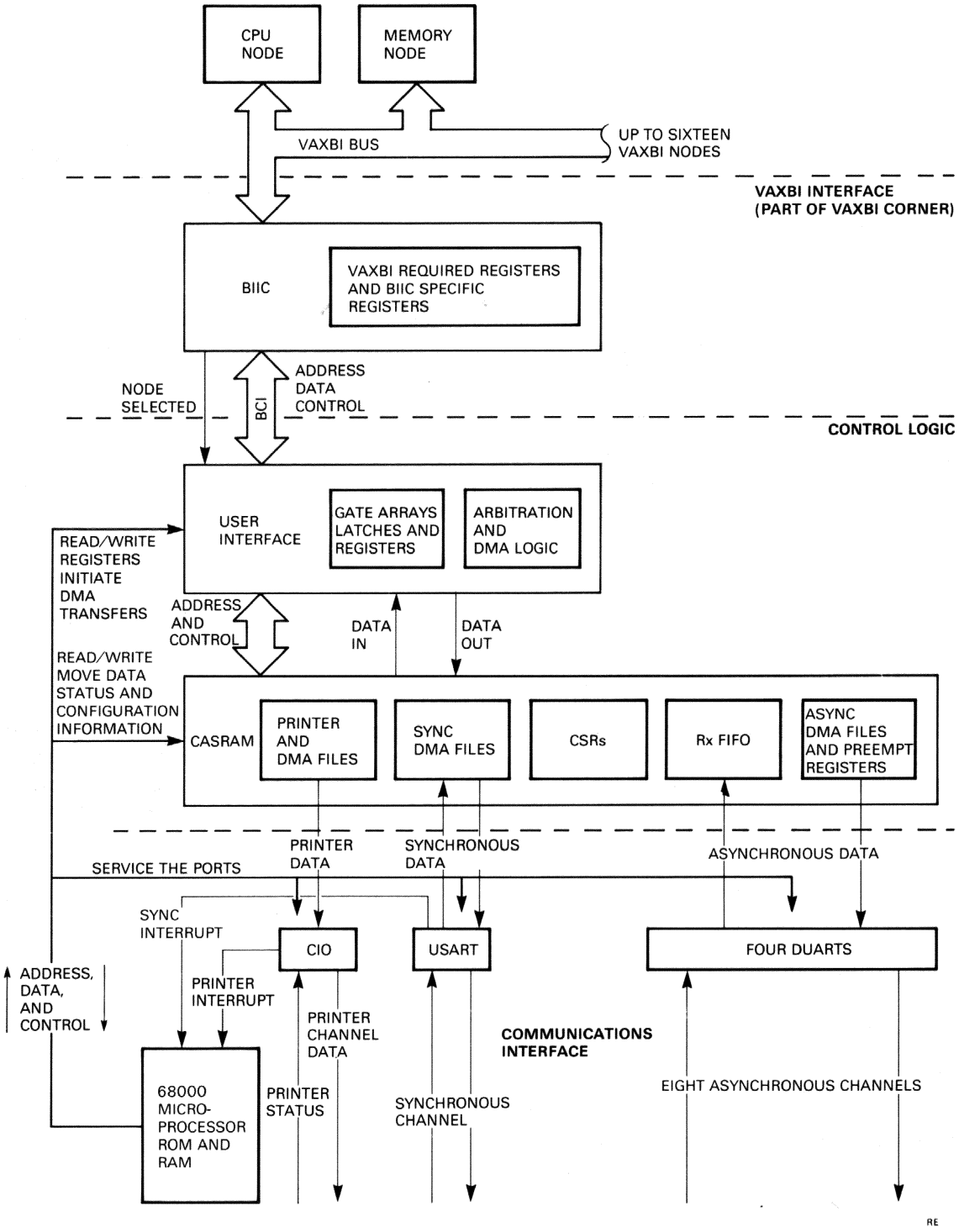


Figure 1-1 DMB32 Functional Block Diagram

1.2.1 Communications Interface

This consists of the asynchronous, synchronous, and printer ports under the control of a 68000 microprocessor. The microprocessor has firmware ROM and dedicated scratch RAM. It controls the VAXBI Interface by a combination of interrupts and polling.

1.2.2 Control Logic

For explanatory purposes, this can be subdivided into two areas of operation:

1. Common Address Store RAM (CASRAM)

2. User Interface

1.2.2.1 Common Address Store RAM (CASRAM) – CASRAM is accessed by the VAXBI host, or by the on-board microprocessor. It contains data buffer areas and most of the CSRs used by the DMB32.

1.2.2.2 User Interface – The interface between the BIIC and the user functions will be called the User Interface in this manual, to conform with the VAXBI generic documentation. Gate arrays in this block contain most of the DMB32 logic, and one of the arrays arbitrates between host and microprocessor access to CASRAM.

The User Interface can respond as slave to commands passed on by the BIIC, or as bus master it can request VAXBI transfers. VAXBI transfer requests are made to the BIIC, which arbitrates for the bus and then controls the transfer.

1.2.3 VAXBI Interface

Handling of VAXBI protocol, parity, and the recognition of commands and addresses are all performed by a VAXBI interface chip (BIIC) on the VAXBI Interface. Valid commands addressed to this node are passed to the User Interface which can then respond via the BIIC, as bus slave. The BIIC contains registers whose functions are described in Section 1.2.4.

1.2.4 Registers

The device registers (CSRs) are mapped into CASRAM; however, some are in the BIIC and some are duplicated in the gate arrays. Device registers can be considered as three groups.

1. VAXBI Required registers

2. BIIC Specific registers

3. User registers

1.2.4.1 VAXBI Required Registers – These are provided on every device (node) that is connected on the VAXBI. These registers hold status and control information which defines how the DMB32 will respond to commands on the VAXBI.

1.2.4.2 BIIC Specific Registers – These registers are not required by the VAXBI specification. They support specific functions implemented by this BIIC. Features selected by the BIIC Specific Registers include the commands to be implemented by this node.

1.2.4.3 User Registers – These are the registers through which the device application is controlled. They can be further subdivided into four groups as follows:

- General Registers – Used to control parameters and functions common to the communications interface.
- Asynchronous Registers – Used to control and monitor the asynchronous ports.
- Synchronous Registers – Used to control and monitor the synchronous port.
- Printer Register – Used to control and monitor the printer port.

1.2.5 Types of Data Transfer

The function of the DMB32 is to move data between the VAXBI and the communications interface; this can be achieved in three ways:

1. DMA Transfer
2. Programmed (RX FIFO) Asynchronous Reception
3. Programmed (Preempt) Asynchronous Transmission

1.2.5.1 DMA Transfer – Synchronous RX and TX data are transferred between host memory buffers and CASRAM by DMA. The microprocessor monitors the buffers, and initiates further DMA transfers as they are needed. Transfers between CASRAM and the synchronous port are initiated by interrupt from the synchronous port to the microprocessor.

Printer data is also transferred across the VAXBI by DMA. Transfers of data from CASRAM to the printer port are performed by the microprocessor. Printer status is polled by the microprocessor, which also makes status information available in the CSRs.

Asynchronous TX data can be transferred by DMA or by programmed transfer. DMA transfers to CASRAM are controlled in the same way as synchronous data transfers. The microprocessor monitors the asynchronous ports to check if they are ready to accept data.

1.2.5.2 Programmed (RX FIFO) Asynchronous Reception – The microprocessor transfers asynchronous RX data to a 512-character RX FIFO in CASRAM. The host reads the data, from the RX FIFO, together with error status information. The RX FIFO is also used to send diagnostic reports to the host and to send modem change information.

1.2.5.3 Programmed (Preempt) Asynchronous Transmission – Asynchronous TX data may be transferred to CASRAM by DMA, or it can be written to a single-character preempt register (one exists in each asynchronous channel). From there the transfer is completed by the microprocessor. A character written to the preempt register of a channel which is transmitting a DMA buffer, will be transferred to the asynchronous port before the remaining DMA data.

1.2.6 Interrupts to the Host

The DMB32 can be programmed to interrupt a VAXBI host under the following conditions:

- When a channel is ready for another preempt character.
- When transfer of a DMA buffer from system memory to an I/O channel has:
  - been aborted
  - been terminated due to a memory read error
  - been successfully completed
- When a received character has been placed in a previously empty RX FIFO. (The DMB32 can be programmed to delay this interrupt so that several characters can be placed in the FIFO before the interrupt is raised).
- When modem status information has been placed in the RX FIFO. This action overrides any programmed interrupt delay.

1.3 HARDWARE OVERVIEW

The major DMB32 hardware blocks and the buses that interconnect them are shown in Figure 1-2. In this figure, the three areas of logic are identified for comparison with Figure 1-1. A brief overview of the hardware is provided in the subsequent sections.

1.3.1 VAXBI Interface (BIIC)

The BIIC is the major functional element of the VAXBI interface. The BIIC provides a standard VAXBI family interface between the VAXBI bus and the user.

1.3.2 User Interface

The functions of the devices within the user interface are defined as follows:

1.3.2.1 Data Paths – The Data Paths is a data bus multiplexer. Latches in this block isolate the BCI (VAXBI Chip Interface) and internal data buses. Latching the data allows the DMB32 to meet VAXBI timing requirements.

1.3.2.2 Busmux – The Busmux is a bus multiplexer and isolator. This group of latches and drivers multiplexes the microprocessor data and address buses (68000 bus) onto the internal data bus.

1.3.2.3 CASRAM – CASRAM is a 2 K-longword RAM that contains device registers used by both the VAXBI host and the DMB32 microprocessor to configure and control the DMB32 module. All data to and from the communications interface is transferred via registers and data buffer areas in CASRAM.

1.3.2.4 Address Translation Gate Array (ATGA) – The ATGA multiplexes primary (VAXBI) and secondary (microprocessor) addresses for the CASRAM. It also performs address translation functions such as indexing and FIFO addressing, and 68000 versus VAX-11 addressing convention.

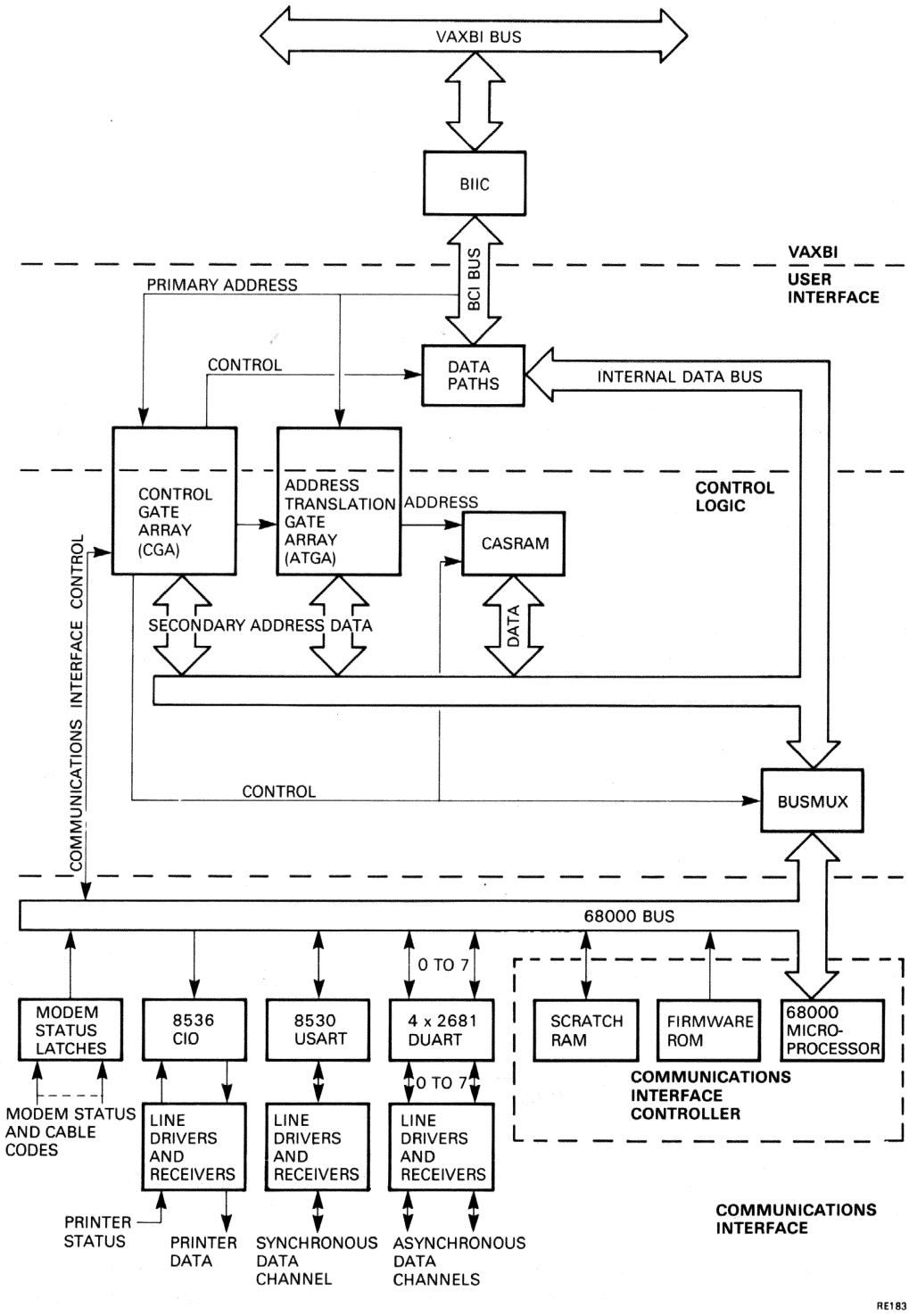


Figure 1-2 Hardware Overview Diagram

1.3.2.5 Control Gate Array (CGA) – The bulk of the DMB32 logic is in the CGA. By controlling the ATGA, the CGA arbitrates between the primary and secondary address ports, allowing only one port to address the CASRAM at any time. The CGA controls the CASRAM, Data Paths and Busmux, as necessary to transfer the data.

Some of the control and decoding logic for the microprocessor and the communications interface is integrated in the CGA.

1.3.3 Communications Interface

The functions of the devices within the communications interface are described in the following sections.

1.3.3.1 Microprocessor – A68000 microprocessor with dedicated ROM and RAM forms the Communications Interface Controller. The controller controls and configures the communications interface in response to information in device registers in the CASRAM. (Like the DMB32 itself, the ICs that form the I/O ports are controlled and configured by access to internal registers).

1.3.3.2 Asynchronous Ports – Eight asynchronous ports are provided by four 2681 Dual Universal Asynchronous Receiver Transmitter (DUART) ICs. These ICs perform the serial/parallel conversion of data, generate parity, and provide modem status and control signals.

1.3.3.3 Synchronous Port – The synchronous port uses an 8530 Universal Synchronous Asynchronous Receiver Transmitter (USART) IC. This IC performs serial/parallel conversion of data, generates and checks CRC, and provides the modem status and control signals required for the synchronous port.

1.3.3.4 Printer Port (Parallel Port) – The port is formed by an 8536 Counter/timer and parallel I/O chip (CIO). This IC has three ports, one of which is used for the printer. Other ports support the printer function, monitor some of the asynchronous port modem signals, and supply a clock to the synchronous port. Counter/timers in the CIO are used to provide delays, and to alert the microprocessor to tasks that must be performed at regular intervals.

1.3.3.5 Line Drivers and Receivers – Line drivers and receivers provide circuitry to drive data on a line, and to reconstitute received data. In the case of serial data communications lines, line drivers and receivers convert between TTL levels on the option and EIA levels on the lines. The printer channel is at TTL level only.

1.4 DATA ROUTING

During DMA transfers, or when the host writes or reads a register, data between the VAXBI and CASRAM is routed as follows:

WRITE	BIIC – Data Paths – CASRAM
READ	CASRAM – Data Paths – BIIC

Data to or from the I/O ports is routed as follows:

WRITE	BIIC – Data Paths – CASRAM – microprocessor – I/O Port
READ	I/O Port – microprocessor – CASRAM – Data Paths – BIIC

For each type of transfer, the CGA controls the ATGA, CASRAM, Data Paths, and Busmux, as necessary.

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 SCOPE

Chapter 2 provides operational information on the facilities provided by the DMB32. The Overview section contains details of DMA Operation, VAX-11 Address Translation, Interrupts, and Synchronous Operation; this is followed by a section on compliance with the Electrical Interface Standards. Sections are also included on Modem Control, DMB32 Protocol Support, HDLC/SDLC Support, DDCMP Support, IBM Bisync Support, and General Byte Support. Finally, the chapter provides information on the Asynchronous Multiplexer, and the Printer Interface.

2.2 OVERVIEW

The DMB32 module, containing an on-board 68000 microprocessor and firmware, implements a communications controller as described in the previous chapter. The main functions of the DMB32 are to provide the following communications facilities:

1. Synchronous Controller handling the following protocols:
   - HDLC/SDLC
   - IBM BISYNC
   - DDCMP
   - GEN BYTE
2. Asynchronous Multiplexer for eight lines
3. Line Printer Interface

The microprocessor is the main controller of the DMB32 but some state oriented control is implemented by the CGA controlling the CASRAM in particular. All VAXBI bus protocol and control is handled by the BIIC.

2.2.1 DMA Operation

DMA data transmissions are implemented for synchronous transmission and reception, asynchronous transmission, and line printer output. DMA operations support both virtual and physical mapping of the host memory.

In order to use the maximum bandwidth of the VAXBI bus, the DMB32 attempts to use octaword transfers wherever possible, and achieves this by the use of internal DMA files kept in CASRAM. A DMA file is used to accept data from the VAXBI on transmit, and to supply data to the VAXBI on receive. The DMA file holds up to an octaword of data and the relevant physical address within the host memory buffer.

During the progress of a DMA, the DMB32 does not update the count and address registers, so it is not possible for the host to track the progress of a DMA via these registers. An exception to this is when the GEN BYTE protocol is being used in receive messages; for these messages, only the count register is maintained. The count is a DMA count and because of the nature of the DMA files, at any given time it may be up to 16 bytes out of date.

2.2.2 VAX-11 Address Translation

The DMB32 is able to carry out VAX-11 address translation for use when accessing DMA buffers. It can use one of four modes of address translation as described in the subsequent sections.

2.2.2.1 Physical Address of Buffer – The DMB32 performs no address translation, but is given the physical address of the buffer and the length. The buffer must be contiguous in physical memory. This mode makes no assumptions about the virtual address structure used by the host processor.

2.2.2.2 Physical Address of Page Table – The DMB32 is given the physical address of a page table. Each entry in the page table is 4 bytes long. The DMB32 is given the offset of the first byte of the buffer which is in the page described by the page table entry. (This number must be less than 512). Each page table entry contains bits 9 to 29 of the physical address of the page that is to be accessed. The remaining high order bits in the page table entry are ignored. This mode allows buffers to be split into several pages.

2.2.2.3 System Virtual Address of Buffer – This mode of address translation uses the VAX Address Translation Structure, as described in the VAX Architecture reference manual.

The DMB32 is given the system virtual address of the buffer. It then accesses page table entries in the system (and possibly global) page tables to perform the address translation. A page table entry has the following format:

- Bits<20:0> – PFN. These bits contain the high order 21 bits of the physical address to be used to access the page (except when the PTE indicates that a global page table entry is to be used).
- Bits<21:0> – Global Page Table Index. When the PTE indicates a global page is to be accessed, these bits indicate the index to be used to access the global page table entry.

NOTE

Only one of the above formats will apply at any given time, as determined by bits 31, 26, and 22.

- Bit<22> – This bit, together with bits 26 and 31, indicates the format of the PTE. The following combinations are recognized:

31	26	22		
1	x	x	–	valid page
0	0	0	–	valid page
0	0	1	–	use PFN to index global page table (not valid in an entry in the global page table).
0	1	x	–	not valid

- Bits<25:23> – Not used
- Bit<26> – See bit 22
- Bits<30:27> – Protection code. This field is checked to ensure that kernel mode has access to the page. The field must not contain 0 or 1 when the page is to be read by the DMB32, and must not contain 0, 1, 3, 7, 11 or 15 if the page is to be written by the DMB32. Only one protection code field is used for any page to be accessed. This comes from the first PTE that describes the page. The codes in global page table entries and in system page table entries, that are accessed in order to find another page table entry, are ignored.
- Bit<31> – See bit 22

When the PTE indicates that a global page table entry is to be found, the DMB32 calculates the system virtual address of the global page table entry by multiplying the index (bits<21:0>) by four and adding the result to the address of the base of the global page table. The DMB32 reads the PTE which results in another system page table translation. This second system PTE must not refer to another global page table, nor must the entry be found in the global page table. At most, one global page table entry is fetched for translating any page address.

2.2.2.4 System Virtual Address of Process Page Table – The DMB32 is given the system virtual address of a process page table entry which describes the page containing the first byte of the buffer. The longwords that follow describe the succeeding pages of the buffer. The format of the page table entries is the same as in the previous mode. This method allows the DMB32 to directly access process buffers. The use of the system virtual address allows for the handling of non-contiguous PTEs describing a process buffer.

Address translations using the system page table are necessary to access the entries in a process page table. If the entry in this table refers to a global page then further PTEs will need to be accessed from the system page table and the global page tables as described for the previous mode.

2.2.3 Interrupts

There are four interrupt vectors used by the DMB32, these are defined as follows:

- BIIC error interrupts
- Asynchronous Interrupts (Transmit and Receive)
- Printer Interrupts
- Synchronous Interrupts

These interrupts are all at the lowest interrupt priority on the VAXBI bus. The DMB32 interrupts with a vector made up in the following way:

- Bits<1:0> – 0
- Bits<5:2> – The Node ID (Node Identifier)
- Bits<7:6> – The value for the type of interrupt, identified as follows:

  0 – BIIC error interrupts
  1 – Synchronous Interrupts
  2 – Asynchronous Interrupts
  3 – Printer Interrupts
- Bit<8> – 1
- Bits<31:9> – 0

The use of these interrupts is controlled by four interrupt enable bits (two for asynchronous, one for synchronous, and one for printer). The error interrupt is controlled by a separate mechanism in the BIIC.

If the appropriate enable bit is set when an interrupting condition occurs, then the DMB32 will interrupt immediately. If the enable bit is not set then the interrupt will be delayed until the interrupt enable bit is set. This interrupt may occur even if the interrupt condition has been removed; for example, if the host empties a FIFO and then sets the interrupt enable bit, an interrupt may be generated with the FIFO empty.

If the enable bit is cleared after an interrupt request is made by the DMB32, then the interrupt may still take place; for example, if the interrupt enable bit is cleared while the host is running at an IPL which would prevent the interrupt taking place. The interrupt may still have taken place when the IPL was lowered, even if the interrupt condition had been removed and the interrupt enable bit cleared.

If the interrupt enable bit is cleared after an interrupt is requested, but without processing the interrupt condition, then there may not be a second interrupt if the enable bit is set later. This applies even if the condition which caused the original interrupt remains; for example, if a FIFO is not emptied but the enable bit is cleared, then when the enable bit is later set again, there may not be a further interrupt until the FIFO is emptied and new entries made. This means that the host driver is not able to set the interrupt bit, and therefore get an interrupt to continue processing the interrupt data, unless it is sure that the interrupt condition was removed when the enable bit was cleared.

2.2.4 Synchronous Operation

In order to facilitate synchronous communications the host system must first allocate a host memory buffer. The address and size of this buffer must be then passed to the DMB32 via the appropriate registers and the appropriate DMA START bit must be set.

Due to the time-critical nature of synchronous communications and use of generally higher transfer speeds, the synchronous port may be allocated two buffers for transmit and two for receive. This allows the host system to service one buffer while the DMB32 is transferring data to or from the other. For the same reasons, the DMB32 allocates two internal DMA files to each synchronous buffer within the host, thus allowing similar double buffering to take place between the microprocessor and the CGA. To permit the microprocessor to order these and other tasks, the synchronous DMA files are assigned to different work queues representing the DMA file state. The following queues are used:

2.2.4.1 Synchronous Transmit Full DMA File Queue – This queue is used to order the transfer of data from TX DMA files, having been filled with data transferred from the host memory transmit buffers, to the synchronous communications port.

2.2.4.2 Synchronous Transmit Done DMA File Queue – This queue is used to order the refilling of TX DMA Files from the host memory transmit buffers, once their contents have been transferred to the synchronous communications port.

2.2.4.3 Synchronous Receive Empty DMA File Queue – This queue is used to order the presenting of RX DMA files with valid translated physical addresses, to the synchronous receive process. These files are then filled with receive data from the synchronous line.

2.2.4.4 Synchronous Receive Full DMA File Queue – This queue is used to order the unloading of the RX DMA buffers to the host memory buffers once they have been filled by the synchronous receive process.

2.2.4.5 Synchronous Buffer Translate Queue – This queue is used to submit requests for host buffer address translation. It is shared by the synchronous transmit and the synchronous receive processes.

2.2.4.6 Further Information – The internal DMA files are pointed to indirectly, via Synchronous File Status Blocks (SFSBs) which track the contents of the DMA file and the phase of the communications protocol in use. The SFSBs also point to another status block called the Synchronous Buffer Status Block (SBSB) which describes and tracks the host memory buffer.

In the case of Synchronous RX DATA the synchronous receive process requests a DMA file from the Synchronous Receive Empty DMA file queue. This file is empty of data but contains the valid physical address of the next octaword to be filled in the host memory buffer. The microprocessor fills the file with 16 bytes of data received from the synchronous line. In the case of start or end of buffer, where less than an octaword transmission is required, the size of transfer is tracked via the SFSB. This information is used for the byte mask during the VAXBI data cycle. When filled, the DMA file is queued to the Synchronous Receive Full DMA file queue. The CGA gate array controls the actual DMA of the data in the DMA file, across the VAXBI bus to host memory. The microprocessor first has to set up the following registers:

1. DMA File Address Register – This register is found in the ATGA and contains the address of the DMA File in CASRAM. The address is incremented as each longword is transferred.
2. VAXBI Command Register – This register is found in the CGA. Writing to the register causes the corresponding command to be initiated on the VAXBI bus.
3. VAXBI Mask Register – This register is found in the CGA, and is used by the microprocessor to indicate which bytes to transfer, when less than a longword transfer is required.

The CGA indicates the status of the transfer to the microprocessor via its VAXBI Status register. Having completed the DMA the CGA indicates this to the microprocessor. The empty DMA file, after being loaded with a valid host memory physical address, is passed back to the Synchronous Receive Empty DMA file queue ready to accept more synchronous receive data.

The synchronous transmit procedure is similar to that of the receive procedure. This time the address of an empty DMA file, with a valid host memory transmit buffer physical address written into it, is loaded into the DMA file Address Register. Where necessary, the VAXBI mask bits are loaded into the VAXBI Mask Register and the appropriate command is loaded into the VAXBI Command Register. The CGA controls the DMA of data from host memory into the DMA file. When this transfer is complete the microprocessor places the full DMA file in the Synchronous Transmit Full DMA file queue. The synchronous transmit process can now service this queue, transferring data from the DMA file to the synchronous line. TX DMA files, when empty, are now passed to the Synchronous Transmit Done DMA file queue for re-use. The use of two DMA files per host buffer will allow the CGA to be filling or emptying one file while the microprocessor services the other.

2.3 SYNCHRONOUS INTERFACE

Electrically and mechanically, the T1012/H3033 synchronous port is:

- Compliant with RS-232-C, RS-422-A/RS-449, V.11, X.27 and V.35
- Compatible with RS-423-A/RS-449, V.28/V.24, V.10 and X.26

The maximum sensible speed that can be used on the sync port depends on the protocol and electrical interface standard selected. Table 2-1 lists the maximum sensible speeds for all the supported protocols. The maximum total throughput for the sync port is 16000 char/s.

Table 2-1 Maximum Sensible Speeds (Sync Port)

	Electrical Interface Standard				
Protocol	**RS-232**	**RS-423**	**RS-422**	**V.35**	
DDCMP	19200	19200	19200	19200	**Data rate (Bits/s)**
HDLC	19200	64000	64000	48000	
BISYNC	9600	9600	9600	9600	
GEN BYTE	9600	9600	9600	9600	

2.4 MODEM CONTROL (SYNC PORT)

Modem status changes are reported by an interrupt. Since there is no indication of which bits have changed, the host is expected to keep a record of the last modem status it saw. This will enable the host to identify which bits have changed. There are two special modem control bits which show whether or not the receive and transmit clocks are working. If no transition is detected on any clock line from the modem for 1 second, the appropriate modem control bit is cleared. When one of these bits is cleared, any transfer in the corresponding direction will terminate with an error indication. Other modem state changes can also affect transfers; for example, loss of DCD or DSR will abort a received message, and loss of DSR or CTS will abort a transmission. If the host drops DTR or RTS during a transmission, the DMB32 will not drop the corresponding signal to the modem, until the message and trailing pads have been transmitted.

2.4.1 Modem Control Signals

Table 2-2 lists the interchange circuits which are supported for the synchronous port on the DMB32.

The state of the modem status lines is sampled at 10 ms intervals and compared to the last reported state. If there has been a change of state the line status register is updated. Ring Indicator signal (Circuit 125) is monitored over a period of 30 ms, and a change recorded only if the state of the signal has been constant over this period.

Table 2-2 Synchronous Interchange Circuits

EIA RS-449 Signal	Name	Pin	EIA RS-232-C Signal	Name	Pin	CCITT V.24 Signal	Name	Pin
–	Shield	1	AA	Protective Ground	1	–		–
SG	Signal Ground	19	AB	Signal Ground	7	102	Signal Ground	7
SC	Send Common	37	–		–	–		–
RC	Receive Common	20	–		–	–		–
IS	Terminal In Service	36	–		–	–		–
IC	Incoming Call	15	CE	Ring Indicator	22	125	Calling Indicator	22
TR	Terminal Ready (+)	12	CD	Data Terminal Ready	20	108/2	Data Terminal Ready	20
TR	Terminal Ready (–)	30	–		–	–		–
DM	Data Mode (+)	1	CC	Data Set Ready	6	107	Data Set Ready	6
DM	Data Mode (–)	29	–		–	–		–
SD	Send Data (+)	4	BA	Transmitted Data	2	103	Transmitted Data	2
SD	Send Data (–)	22	–		–	–		–
RD	Received Data (+)	6	BB	Received Data	3	104	Received Data	3
RD	Received Data(–)	6	–		–	–		–
TT	Terminal Timing (+)	17	DA	Transmitter Signal Element Timing (DTR Source)	24	113	Transmitter Signal Element Timing (DTR Source)	24
TT	Terminal Timing (–)	35	–		–	–		–
ST	Send Timing (+)	5	DB	Transmitter Signal Element Timing (DCE Source)	15	114	Transmitter Signal Element Timing (DCE Source)	15
ST	Terminal Timing (–)	23	–		–	–		–
RT	Receive Timing (+)	8	DD	Receiver Signal Element Timing	17	115	Receiver Signal Element Timing	17
RT	Receive Timing (–)	26	–		–	–		–
RS	Request To Send (+)	7	CA	Request To Send	4	105	Request To Send	4
RS	Request To Send (–)	25	–		–	–		–
CS	Clear To Send (+)	9	CB	Clear To Send	5	106	Clear To Send	5
CS	Clear To Send (–)	27	–		–	–		–
RR	Receiver Ready (+)	13	CF	Received Line Signal Detector	8	109	Data Channel Received Line Signal Detector	8
RR	Receiver Ready (–)	81	–		–	–		–
SQ	Signal Quality	33	CG	Signal Quality Detector	21	110	Data Signal Quality Detector	21
NS	New Signal	34	–		–	–		–
SR	Signaling Rate Selector	16	CH	Data Signal Rate Selector (DTE Source)	23	111	Data Signaling Rate Selector (DCE Source)	23
SI	Signaling Rate Indicator	2	CI	Data Signal Rate Selector (DCE Source)	23	–		–
LL	Local Loopback	10	–		–	141	Local Loopback	18
RL	Remote Loopback	14	–		–	140	Remote Loopback	21
TM	Test Mode	18	–		–	142	Test Indicator	25
SS	Select Standby	32	–		–	–		–
SB	Standby Indicator	36	–		–	–		–

2.5 PROTOCOL SUPPORT (SYNC PORT)

The synchronous controller will support message framing and CRC generation and checking of the following protocols:

- HDLC/SDLC
- DDCMP
- IBM BISYNC

Error recovery, message sequencing, and retransmission are not performed by the DMB32. Software assistance is required to implement the full protocols.

Other byte oriented protocols can be supported by the GEN BYTE protocol. This allows the DMB32 to receive and transmit messages in other protocols, but software is required for all intelligent interpretation of their contents.

The synchronous line is capable of running at 64 Kbit/s (full duplex) for HDLC and SDLC protocols only. For DDCMP it will run at 19.2 Kbit/s. Other protocols are capable of running at 9600 bit/s.

2.6 HDLC/SDLC SUPPORT (SYNC PORT)

The DMB32 supports HDLC/SDLC protocol message framing. No error recovery or message sequencing is carried out without software help.

2.6.1 Message Definition

An HDLC/SDLC message is a frame consisting of a sequence of bits preceded by a flag character (01111110) and followed by either a flag character or a 7-bit abort sequence (1111111). The bits between these delimiters are the data followed by the block check characters.

To prevent bit patterns in the data stream from being wrongly interpreted as flag characters, the data is transparently encoded by inserting an extra '0' bit following each '11111' sequence found in the data to be transmitted.

On reception, these inserted '0' bits are removed. This bit stuffing and stripping is carried out by the DMB32. Only messages which are a multiple of the character size long are legal. If the final flag is not on a character boundary then an error is indicated.

The block check is normally found in the final bits of the frame; however, if the frame is terminated by an abort sequence then there is no block check sequence in the frame. After the transmission of the closing flag of a frame, the DMB32 can either send further flag sequences or a continuous mark, depending on the state of the IDLE.SYNC bit.

At least one flag will be transmitted before a message, this flag can be the same flag that closes the previous message. Messages must have at least two data characters. If a shorter message is received it will be discarded.

2.6.2 Block Check

The block check is normally CRC-CCITT, but the following block checks can also be specified:

- CRC-CCITT preset to 1s
- No block check – no block check is transmitted, but one can be included at the end of the buffer if an alternative block check is desired.

2.6.3 Character Size
Only the eight bit character size can be specified for HDLC/SDLC.

2.6.4 Address Bytes
If the device is set up as a secondary station then the first one or two bytes of a message are inspected to determine if they match the station address, in ADDRESS1 and ADDRESS2. If they do not match then the received message is ignored and the receiver waits for the next message.

The address match procedure is not the same for SDLC and for HDLC. SDLC only uses one-byte addresses; the first byte of the message is compared with the first byte of the address. The all stations address (11111111) is also recognized as an address match.

HDLC can use one- or two-byte addresses. The low bit of the first address byte indicates whether one- or two-byte addresses are used. If this bit is a '0', two-byte addresses are used. If it is a '1', then one-byte addresses are used. The appropriate bytes of the message are compared with the address bytes provided by the host. The all stations address (11111111) is also recognized as an address match.

The ADCCP protocol can also be implemented using HDLC mode. ADCCP allows more than two bytes of address. In this case the DMB32 checks the first two bytes, further bytes are checked by the host.

2.6.5 Aborting a Transmission
If the host aborts a transmission after the first byte has been passed to the USART, and before the final flag has been transmitted, then the DMB32 will transmit an 8-bit abort sequence (11111111) to abort transmission. A transmission may also be aborted if some error is discovered while transmitting a message; for example, memory error or transmit underrun condition.

2.7 DDCMP SUPPORT (SYNC PORT)
The DMB32 supports the framing of DDCMP messages, valid start of message recognition, and CRC generation and checking. Software is required to support error recovery, message sequencing, and retransmission.

2.7.1 Message Definition
The DMB32's interpretation and actions with regard to the structure of a DDCMP message are described as follows:

2.7.1.1 SYNSEQ – A number of SYNC (synchronization) characters (96_{16}) precede a DDCMP message. The DMB32 will accept abutting DDCMP messages that have no synchronization characters separating them, provided that no error has been detected and the QSYNC bit in the previous message is clear. The DMB32 will normally attempt to send abutting messages.

If a message that has the QSYNC bit set is sent, the DMB32 will send at least four synchronization characters following the message. If no message is available for transmission abutting a message with QSYNC clear (or the value in LPR1 is not zero), then at least 8 synchronization characters will be sent between messages. If any transmit error is detected; for example, memory error, or transmit underrun, then at least eight synchronization characters will be sent before the next message. The DMB32 will always send at least the number of synchronization characters specified in LPR1.

DDCMP requires that eight synchronization characters are transmitted before the following messages:

- Maintenance messages (starting with DLE)
- Messages with the SELECT bit set
- Messages with a change in the ADDR field
- Control messages other than ACK
- Any message after starting up
- Any message after idling mark
- The first message after receiving a NAK
- Messages with a change in the ADDR field

The DMB32 does not automatically send these synchronization sequences for the first five classes listed. The host is required to set the number of synchronization characters in LPR1 before initiating transmission. The DMB32 will force an eight byte sequence in the last two classes.

The DMB32 will not send PAD characters after a message if the next message abuts the first or is separated from it by exactly four synchronization characters (due to QSYNC being set).

2.7.2 Start of Message (SOM)

A single character indicates the start of a DDCMP message. This character must be one of the following valid DDCMP start of message characters:

- SOH (81_{16}) - Data Message
- ENQ (05_{16}) - Control Message
- DLE (90_{16}) - Maintenance Message (MOP)

The start of message character is transferred to memory on a received message.

If the first character of the message is not one of the valid starting characters then the DMB32 will re-synchronize, looking for a new SYNC character.

2.7.2.1 COUNT - This is a two byte field which indicates the number of characters in the data field of the message. Only the bottom 14 bits of this word are used as the count; the high order two bits are the control flags QSYNC and SELECT. The two bytes are transferred to memory on received messages. In a control message there is no data field and therefore no COUNT is required. In this case the count field contains control information.

2.7.2.2 QSYNC - This flag is bit six of the second byte of the COUNT field; it indicates that the next message will not abut this message. On transmission, the DMB32 uses the contents of the bit to check if it must insert SYNC characters between messages. On received messages the DMB32 uses the bit to check if it must re-synchronize at the end of the message.

2.7.2.3 SELECT - This flag is used in DDCMP to control line turn around in half duplex working. The DMB32 does NOT use this bit.

2.7.2.4 CONTROL – This field contains two bytes of control information on control messages. On received messages, the DMB32 will just transfer these bytes to memory.

2.7.2.5 ADDR – This field is one byte in length and is used in DDCMP multi-point operation, however the DMB32 does not support DDCMP multi-point operation.

2.7.2.6 CRC1 – This field is two bytes in length containing CRC-16 information for the message header: SOM, COUNT, CONTROL, and ADDR. This field is not transferred to memory on receive messages and is generated by the DMB32 on transmit messages. On received messages if this CRC is incorrect, reception is terminated and re-synchronization takes place.

2.7.2.7 DATA – This field contains COUNT bytes of data that are transferred to memory on receive messages. This field is not present in a control message that is started by an ENQ character (05_{16}).

2.7.2.8 CRC2 – This field contains two bytes of CRC-16 data for the DATA field that is not transferred to memory on receive messages. The DATA field is not present in control messages.

2.7.2.9 PAD – The PAD field consists of a number of DEL bytes (FF_{16}) which are transmitted following the final CRC. The DMB32 accepts messages that do not have these bytes. The DMB32 always transmits two PAD characters after a message. This field is not transferred to memory on received messages. If the character following the last byte of the last CRC in a message is not a DEL, DLE, SOH, ENQ or SYN then the last CRC is regarded as being invalid and re-synchronization takes place.

2.7.3 Character Size
The only character size allowed is 8.

2.7.4 Aborting Transmission
DDCMP does not allow a transmitter to abort transmission. If it is necessary to abort a transmission, for example, if a memory error occurs, then the DMB32 will transmit CAN characters (18_{16}) until three characters after the following CRC would have been sent. This is likely to cause a CRC error to be detected by the other receiver. Following this it will send at least eight further synchronizing characters before the next message, to allow the other receiver to re-synchronize.

2.8 IBM BISYNC SUPPORT (SYNC PORT)

2.8.1 Message Definition
The message defined here is the unit of transmission that is contained in one buffer, and does not correspond with the definition of a message used in the description of IBM BISYNC.

A message starts with the first non-synchronizing character, and continues until one of the following sequences is recognized:

- ENQ
- ACK0
- ACK1
- NAK
- WACK

- RVI
- TTD
- EOT
- ETB + block check
- ETX + block check

The DMB32 assumes that the data in a transmit buffer is a complete message, and thus does not check that the message ends with one of the above sequences.

When a message is received, the DMB32 automatically starts searching for a new synchronization sequence.

2.8.1.1 Transparent Mode – It is not possible to transmit certain control characters in a message. If binary data is being transmitted, a transparent mode must be used to enable these characters to be sent.

Transparent mode is indicated by the sequence DLE, STX in the message. Once this sequence has been seen, the characters that follow are in transparent mode, until a block check sequence is received.

In transparent mode a DLE has to be transmitted before any of the following characters:

- ETB
- ETX
- SYN
- ENQ
- DLE – to allow DLE to be part of the message
- ITB

Control characters received without the preceding DLE are assumed to be part of the text of the message. The DMB32 will transfer these DLE characters to the buffer that the host provides, on reception, and expects them to be in the transmit buffer. DLE characters are included in the block check.

Following a DLE, ITB, block-check sequence, non-transparent mode is entered. Normally a DLE, STX will be received which will return the DMB32 to transparent mode.

2.8.1.2 Block Check – Either VRC/LRC or CRC-16 are normally used with IBM BISYNC. VRC/LRC is normally used for ASCII, and CRC-16 for EBCDIC. The DMB32 will support either block check sequence for ASCII. If VRC is specified then 7-bit characters must also be specified. 8-bit characters must be specified with CRC-16.

The DMB32 will commence calculating the block check for a message, after it sees the first STX or SOH character. This character is not included in the block check. Other characters not included in the block check, are those preceding the STX or SOH.

The block check is inserted in transmitted messages and checked in received messages after the following characters; the character is included in the block check.

- ETX
- ETB
- ITB

ITB characters are used to separate records within one message. Each record has its own block check. If any of these checks fail in a received message, a block check error is indicated to the host.

On transmission, the VRC/LRC block check character is followed by two PAD characters, and if a PAD character is received it will be discarded. On reception, the block check characters are transferred to memory. On transmission, the host is expected to insert dummy characters in the buffer, and these are replaced by the calculated block check, unless the block check mode specifies that a block check is not required.

2.8.1.3 Inserted SYNC Characters – SYNC characters can be embedded in messages. These characters are not included in the block check and are ignored when they are received. The DMB32 will insert a SYNC sequence whenever the transmission has been in progress without a SYNC for more than 1 second. Note that SYNC sequences in transparent messages are formed by the combination DLE SYN, and for normal messages by SYN SYN.

At least three SYNC characters are transmitted before each message to ensure that the receiver is synchronized. This value is taken from NUMBER.SYNC in LPR1. This is normally set to three for IBM BISYNC by the host.

2.8.1.4 PAD Characters – A PAD character is transmitted after every message. This character is FF_{16}, and it is sent to ensure that data is passed to the modem before it turns the line round (BISYNC normally operates in half duplex mode). The DMB32 will ignore a PAD character following a message.

2.8.1.5 Character Sets – IBM BISYNC can work in either ASCII or EBCDIC, selected by a bit in the line parameter register (6-bit transcode is not supported by the DMB32). The control characters used by BISYNC depend on which character set is selected.

2.8.1.6 ASCII Character Set – The following control character sequences (in hexadecimal) are used in ASCII mode:

- SYN – 16_{16}
- ENQ – 05_{16}
- STX – 02_{16}
- SOH – 01_{16}
- ETB – 17_{16}
- EOT – 04_{16}
- ETX – 03_{16}

- NAK - 15_{16}
- ACK - 06_{16}
- ITB - $1F_{16}$
- ACK0 - DLE, 30_{16}
- ACK1 - DLE, 31_{16}
- WACK - DLE, $3B_{16}$
- DLE - 10_{16}
- RVI - DLE, $3C_{16}$
- TTD - STX, ENQ
- BEL - 07_{16}

The character size must be seven with VRC checking, or eight without VRC checking.

2.8.1.7 EBCDIC Character Set - The following character sequences (in hexadecimal) are used in EBCDIC mode:

- SYN - 32_{16}
- ENQ - $2D_{16}$
- STX - 02_{16}
- SOH - 01_{16}
- EOB/ETB - 26_{16}
- EOT - 37_{16}
- ETX - 03_{16}
- NAK - $3D_{16}$
- ACK - $2E_{16}$
- ITB - $1F_{16}$
- ACK0 - DLE, 70_{16}
- ACK1 - DLE, 61_{16}
- WACK - DLE, $6B_{16}$

- DLE - 10_{16}
- RVI - DLE, $7C_{16}$
- TTD - STX, ENQ
- BEL - $2F_{16}$

The character size must be eight.

2.8.1.8 Aborting a Transmission – If the host aborts a transmission after the first byte has been passed to the USART, and before the final character has been transmitted, then the DMB32 will transmit an ENQ sequence (DLE ENQ in transparent mode) to abort the transmission. A transmission may also be aborted if some error is discovered while transmitting a message; for example, memory error or transmit underrun condition.

2.9 GENERAL BYTE SUPPORT (SYNC PORT)

2.9.1 General Description

The GEN BYTE protocol is provided as an aid in using protocols that are not directly supported by the DMB32. It allows the host to receive and send arbitrary synchronous characters.

2.9.1.1 Received Data – Once synchronization is established, using the synchronization character specified by the host, all received characters are transferred to memory. Embedded synchronization characters can be transferred or ignored, as desired by the host. Block check is calculated on all characters transferred to memory. A transfer terminates at the end of the buffer, when the optional match character is detected or when some error is detected; for example, modem status error, or memory error. The host is expected to do the analysis of message structure from the characters received in the buffer, the DMB32 does little more than transfer them into memory.

The byte count field in RXBUFCT(1/2) is updated after every DMA. This enables the host to determine how many characters have been received while a message is being received, thus allowing the host to process messages before the host buffer is filled up. This is useful because there are many protocols for which the DMB32 is not able to determine the end of message itself.

Because of the use of internal DMA files by the DMB32, a DMA transfer will normally take place when sufficient characters have been received to cross an octaword memory boundary. This means that the DMA byte count will often be up to 16 bytes behind the actual received characters. Due to internal buffering the DMA byte count can be up to 322 bytes out of date.

2.9.1.2 Character Size – If VRC is not specified then a character size of 6 or 8 is selected. If VRC is specified then character size of 7 is selected.

NOTE
If VRC/LRC is specified as the block check then a parity bit will be added to all characters transmitted.

2.9.1.3 Block Check – The following block checks are supported:

- No block check – 6, 7, 8 bits per character only
- CRC-16 – 8 bits per character only
- VRC even – 7 bits per character only
- VRC odd – 7 bits per character only
- VRC/LRC even – 7 bits per character only
- VRC/LRC odd – 7 bits per character only

2.9.1.4 Aborting a Message – If a transmission is aborted then either mark or SYNC characters will be sent, according to IDLE.SYNC.

2.10 ASYNCHRONOUS INTERFACE

The DMB32 implements an 8-line asynchronous multiplexer, with split speed and full modem control capabilities. The following formats are supported in half duplex or full duplex modes:

- 1 start bit
- 5, 6, 7 or 8 Data bits
- Odd parity, even parity, or no parity
- 1, 1.5 or 2 stop bits

Electrically and mechanically, the T1012/H3033 asynchronous ports are:

- Compliant with RS-232-C
- Compatible with V.28/V.24

The T1012 module is also compliant with RS-423-A, V.10 and X.26, but the H3033 is not, due to pin limitations on the 25-way D-type connector.

Half duplex operation is only supported on systems using coded link control, because the DMB32 does not support the secondary transmit and receive signals which are often used for link control. The asynchronous ports are implemented using DUARTs and are associated in the following manner:

- Ports 0 and 1 use the same DUART
- Ports 2 and 3 use the same DUART
- Ports 4 and 5 use the same DUART
- Ports 6 and 7 use the same DUART

2.10.1 Speeds

Split speed operation, using different transmit and receive speeds, is supported on the DMB32 asynchronous ports. The supported speeds are given in Table 2-3.

Table 2-3 Supported Speeds (Async Port)

Speed (bit/s)	Groups
50	A
75	B
110	A and B
134.5	A and B
150	B
300	A and B
600	A and B
1200	A and B
1800	B
2000	B
2400	A and B
4800	A and B
7200	A
9600	A and B
19200	B
38400	A

2.10.2 Speed Selection

The transmit and receive speeds selected for both ports sharing a DUART must be contained in the same group (A or B). If the transmitter and receiver of a port are configured in conflicting groups, then the group of the receiver will be used. If two ports sharing a DUART are configured in conflicting groups, then the group of the most recently configured channel will be used.

Certain speed combinations are not valid. Any speed combination that requires a speed from the group 50, 7200 and 38400 and also a speed from the group 75, 150, 1800, 2000, 19200 on the same DUART is not valid. If the speed group of a port is changed by configuring a partner port in a conflicting group, then the new speed of that port is redefined as shown in Table 2-4.

Table 2-4 Redefinition of Conflicting Speeds

Previously Selected speed (bit/s)	New Speed (bit/s)
50	75
75	50
150	200
1800	7200
2000	1050
7200	1800
19200	38400
38400	19200

2.10.3 Performance

Each asynchronous channel is capable of full-duplex operation at data rates of up to 38400 bits/s. However, the DMB32 cannot support eight channels operating at 38400 bits/s at the same time. The total maximum throughput of the DMB32 is around 21000 characters per second (eight data bits with start and one stop bit), including all characters transmitted and received on the synchronous line and the printer port.

If congestion occurs, the DMB32 will give priority to the synchronous line, followed by the reception of async characters.

The individual maximum throughput for each async port is 1000 char/s.

2.11 DATA TRANSFERS (ASYNCHRONOUS PORT)

2.11.1 Receive Data

After characters have been assembled into a parallel form, as defined by the port characteristics, they are merged with the port number and any error status bits, and put into a 512-character FIFO. This can be read by any processor on the VAXBI bus.

If at any time both the received character FIFO and the DUART's internal FIFO become full, then new received characters are discarded until the congestion is relieved. If this happens, the DMB32 purges the DUARTs internal FIFO and then inserts a null character with an overrun error in the received character FIFO. When this is completed, the DMB32 resumes normal operation on that line.

The FIFO is also used for passing modem status information and diagnostic messages to the host.

2.11.2 Transmit Data

Transmit data is handled by the DMB32 either by DMA buffers or by loading single characters direct into the DMB32 by the host.

When using DMA buffers, the host passes a descriptor to the DMB32 for the start of a buffer which is to be transmitted. The DMB32 transmits the buffer and, if enabled, interrupts the host when the last character has been sent. The DMB32 translates addresses using page tables as previously described. DMA output can be terminated prematurely by use of the transmit DMA abort bit in the line control register. Asynchronous transmit DMA, uses internal DMA files in a similar manner to the synchronous port, but because the latency of the asynchronous operation is not as critical, DMA file queues are not used.

In the single-character mode, one character at a time is transferred to the DMB32. This character will be transmitted as soon as possible, having priority over those characters in DMA buffers. This is called a PREEMPT transfer.

2.11.3 Interrupts

The host can be interrupted on any of the following conditions, assuming the appropriate interrupt enable bit is set :

- The single character buffer has become empty.
- A DMA buffer previously passed to the DMB32 has completed (the last character has been transmitted).
- A DMA buffer previously passed to the DMB32 has been aborted by the host and the abort is now complete.

- A DMA buffer previously passed to the DMB32 has been terminated due to a memory read error (nonexistent memory, memory parity error, or invalid PTE).

- The received character FIFO has at least one character in it after a transition from the empty state.

The DMB32 has two interrupt control bits associated with the asynchronous lines; one enables the host to be interrupted (TX INTERRUPT ENABLE) on the first five conditions in the above list, and the second (RX INTERRUPT ENABLE) allows interrupts on the sixth condition . The receive interrupt can be delayed to allow more efficient processing by the host.

The received character FIFO data assembled in the receive FIFO includes modem status change information, and diagnostic data, in addition to actual received characters.

2.11.4 Modem Control

Table 2-5 lists the interchange circuits which are supported for each of the asynchronous channels on the DMB32.

Table 2-5 Asynchronous Interchange Circuits

EIA RS-449 Signal	Name	Pin	EIA RS-232-C Signal	Name	Pin	CCITT V.24 Signal	Name	Pin
–	Shield	1	AA	Protective Ground	1	–		–
SG	Signal Ground	19	AB	Signal Ground	7	102	Signal Ground	7
SC	Send Common	37	–		–	–		–
RC	Receive Common	20	–		–	–		–
IS	Terminal In Service	36	–		–	–		–
IC	Incoming Call	15	CE	Ring Indicator	22	125	Calling Indicator	22
TR	Terminal Ready (+)	12	CD	Data Terminal Ready	20	108/2	Data Terminal Ready	20
TR	Terminal Ready (–)	30	–		–	–		–
DM	Data Mode (+)	1	CC	Data Set Ready	6	107	Data Set Ready	6
DM	Data Mode (–)	29	–		–	–		–
SD	Send Data (+)	4	BA	Transmitted Data	2	103	Transmitted Data	2
SD	Send Data (–)	22	–		–	–		–
RD	Received Data(+)	6	BB	Received Data	3	104	Received Data	3
RD	Received Data(–)	6	–		–	–		–
TT	Terminal Timing (+)	17	DA	Transmitter Signal Element Timing (DTR Source)	24	113	Transmitter Signal Element Timing (DTR Source)	24
TT	Terminal Timing (–)	35	–		–	–		–
ST	Send Timing (+)	5	DB	Transmitter Signal Element Timing (DCE Source)	15	114	Transmitter Signal Element Timing (DCE Source)	15
ST	Terminal Timing (–)	23	–		–	–		–
RT	Receive Timing (+)	8	DD	Receiver Signal Element Timing	17	115	Receiver Signal Element Timing	17
RT	Receive Timing (–)	26	–		–	–		–

Table 2-5 Asynchronous Interchange Circuits (continued)

EIA RS-449 Signal Name		Pin	EIA RS-232-C Signal Name		Pin	CCITT V.24 Signal Name		Pin
RS	Request To Send (+)	7	CA	Request To Send	4	105	Request To Send	4
RS	Request To Send (–)	25	–		–	–		–
CS	Clear To Send (+)	9	CB	Clear To Send	5	106	Clear To Send	5
CS	Clear To Send (–)	27	–		–	–		–
RR	Receiver Ready (+)	13	CF	Received Line Signal Detector	8	109	Data Channel Received Line Signal Detector	8
RR	Receiver Ready (–)	81	–		–	–		–
SQ	Signal Quality	33	CG	Signal Quality Detector	21	110	Data Signal Quality Detector	21
NS	New Signal	34	–		–	–		–
SR	Signaling Rate Selector	16	CH	Data Signal Rate Selector (DTE Source)	23	111	Data Signaling Rate Selector (DCE Source)	23
SI	Signaling Rate Indicator	2	CI	Data Signal Rate Selector (DCE Source)	23	–		–
LL	Local Loopback	10	–		–	141	Local Loopback	18
RL	Remote Loopback	14	–		–	140	Remote Loopback	21
TM	Test Mode	18	–		–	142	Test Indicator	25
SS	Select Standby	32	–		–	–		–
SB	Standby Indicator	36	–		–	–		–

2.11.5 Protective Ground

EIA Circuit AA (Protective Ground) is also supported. There is no CCITT equivalent of this circuit. A wire link (zero ohm resistor) on the distribution panel PCB (H3033) can be removed to disconnect the protective ground.

The state of the modem status lines is sampled at 10 ms intervals and compared with the last reported state. If there has been a change of state, the line status register is updated. If the port is configured for modem control operation (LINK.TYPE = 1) the new line status information is also loaded into the received character FIFO. The Ring Indicator signal (Circuit 125) is monitored over a period of 30 ms, and a change is recorded only if no change has been detected in the signal over this period.

2.11.6 Auto XON/XOFF Operation

The DMB32 can be programmed to automatically respond to, and issue, XON and XOFF codes. This facility is individually selectable on a per-channel and responding or issuing basis

2.11.7 Received XON/XOFF Characters

When enabled, the DMB32 will stop transmitting data on any channel for which an XOFF has been received; transmission will resume when an XON character is received. The maximum number of characters, excluding XON/XOFF, characters which will be transmitted following the receipt of an XOFF, is three characters. This provides the following features:

- Reduced overheads on the host
- Quicker actioning of a received XON/XOFF than if host intervention is required, thus reducing the chance of lost characters
- Host can override XON/XOFF state at any time

Irrespective of whether or not this mode is enabled, XON and XOFF control characters are always passed to the host. This enables the host to monitor the state of the line and keep a check for potentially lost XON characters.

2.11.8 Transmitted XON/XOFF Characters

If it becomes congested when it is enabled, the DMB32 will transmit an XOFF character; 'congested' means that the received character FIFO is over three-quarters full. To avoid unnecessary line traffic, the XOFF will only be sent out to channels on which characters are received after the congestion has occurred. When the congestion is relieved (when the FIFO becomes less than half full), then the DMB32 will transmit an XON on any channel that has had an XOFF transmitted. If characters are received on a line which has had an XOFF sent then every second character received will result in an XOFF being transmitted.

The host can also request the DMB32 to transmit XON/XOFF characters for any line. If the host has requested XOFF then this will override the automatic sending of XON when the FIFO becomes less than half full.

The characters used for XOFF and XON are programmable for each line.

2.12 PRINTER INTERFACE

The DMB32 supports the LP32 generic printer specification, this includes: LN01, LN01-B, LN01-S, LP25, LP26, LP27, LXY12, LXY22 printers.

2.12.1 Operation

To start printing, the host sets up the DMA address, offset and count registers, and sets the DMA start bit (PR.DMA.START).

The printer interface section of the DMB32 uses DMA files in a similar fashion to the asynchronous multiplexer. The DMB32 will clear the DMA start bit and interrupt the host when the print operation is finished. The type of VAX-11 address translation used for the DMA buffers is specified by the PR.DMA.PTE and PR.DMA.PHYS bits.

The host can abort the print operation at any time by setting the abort bit (PR.DMA.ABORT). When it detects that the abort bit is set, the DMB32 stops transferring characters to the printer, clears the DMA start bit, and interrupts the host.

After a print operation, the host can examine the number of bytes transferred to the printer, and if formatting is enabled, the number of lines of paper used.

The host can determine the printer status at any time by examining the printer CSR (PCSR). This contains three bits which are returned from the printer; these are defined as follows:

- PR.OFFLINE – When set, this bit indicates that the printer is offline.
- PR.CONNECT.VERIFY – When set, this bit indicates that a printer is connected to the distribution panel via the cable.
- PR.DAVFU.READY – This bit is set when the printer is ready for a DAVFU command sequence.

2.12.2 Formatting

The host can specify optional formatting operations to be performed by the DMB32. The formatting control information must be set up before the DMA is started, and must not be modified until the DMA has finished. The formatting functions are:

- Prefix Characters – Prefix characters can be sent to the line printer before the data in the DMA buffer is sent. The number of characters sent is controlled by PREFIX.COUNT. The character to be sent is controlled by PREFIX.CHAR. This feature is used to insert a form feed or new line before sending the data to be printed.
- Suffix Characters – Suffix characters can be sent to the line printer after the data in the DMA buffer. The number of characters sent is controlled by SUFFIX.COUNT. The character to be sent is controlled by SUFFIX.CHAR. This feature can be used to insert a form feed or new line, after the data to be printed.
- Automatic Carriage Return Insertion – If this mode is selected, carriage return characters will be inserted in the data stream, before any form feed or line feed character that is not already preceded by a carriage return.
- Automatic Form Feed to Line Feed Conversion – If this bit is set, any form feed in the data stream will be converted to the appropriate number of line feeds in order to reach the next top of form. The number of lines on a page must be specified in PR.PAGE.
- Non-Printing Character Deletion
- Conversion to Upper Case
- Automatic New Line Insertion (line wrapping)
- Tracking of Carriage Position – The DMB32 tracks the carriage position and uses this information to determine how many line feeds to convert a form feed to, and when it is necessary to insert a new line automatically.

CHAPTER 3
TECHNICAL DESCRIPTION

3.1 SCOPE

This description is based on the DMB32 Overview Diagram given in Figure 3-1. This diagram can be used as a reference throughout the Chapter. Operation of the individual blocks, and the use of buses, is described in the subsequent sections of the chapter.

3.2 MEMORY MAPS

The VAXBI has 30 address lines and can therefore address a space of one gigabyte (1024 Mbytes). Of this, locations 0000 0000_{16} to 1FFF $FFFF_{16}$ are assigned as memory space, and locations 2000 0000_{16} to 3FFF $FFFF_{16}$ are assigned as I/O space. Up to 16 VAXBI nodes (processors, memory, or adapters) can be serviced by one VAXBI bus, and up to 16 VAXBI buses can be mapped into I/O space.

Within I/O space, each VAXBI node is allocated an 8 Kbyte block of addresses for device registers. This is called its node space. The base address of the block for a given node is defined by the node identity (node ID) plug that is installed on the VAXBI backplane for that node, and is independent of the position of the node module in the VAXBI backplane.

In addition, each node is allocated a 256 Kbyte block of addresses in I/O space called its window space. This can be used to map non-VAXBI buses (for example, UNIBUS) to the VAXBI space via an adapter. In the DMB32, when it is in maintenance mode (MAINT<5> or <4> set), the DMB32 register space may be accessed through the window space.

Figure 3-3 shows the memory map of the DMB32, in which areas allocated to VAXBI required registers, BIIC specific registers, and DMB32 registers are identified. The term 'Base' refers to the first address within the node space.

The VAXBI required registers MUST be provided by each node. They hold control and status information relevant to the VAXBI interface. The following data are available:

- Device Type
- Revision Level
- Type of BIIC
- Module and BIIC self-test control and status bits
- Enable and control and status bits for error interrupts
- Type of arbitration used by this node
- Node ID

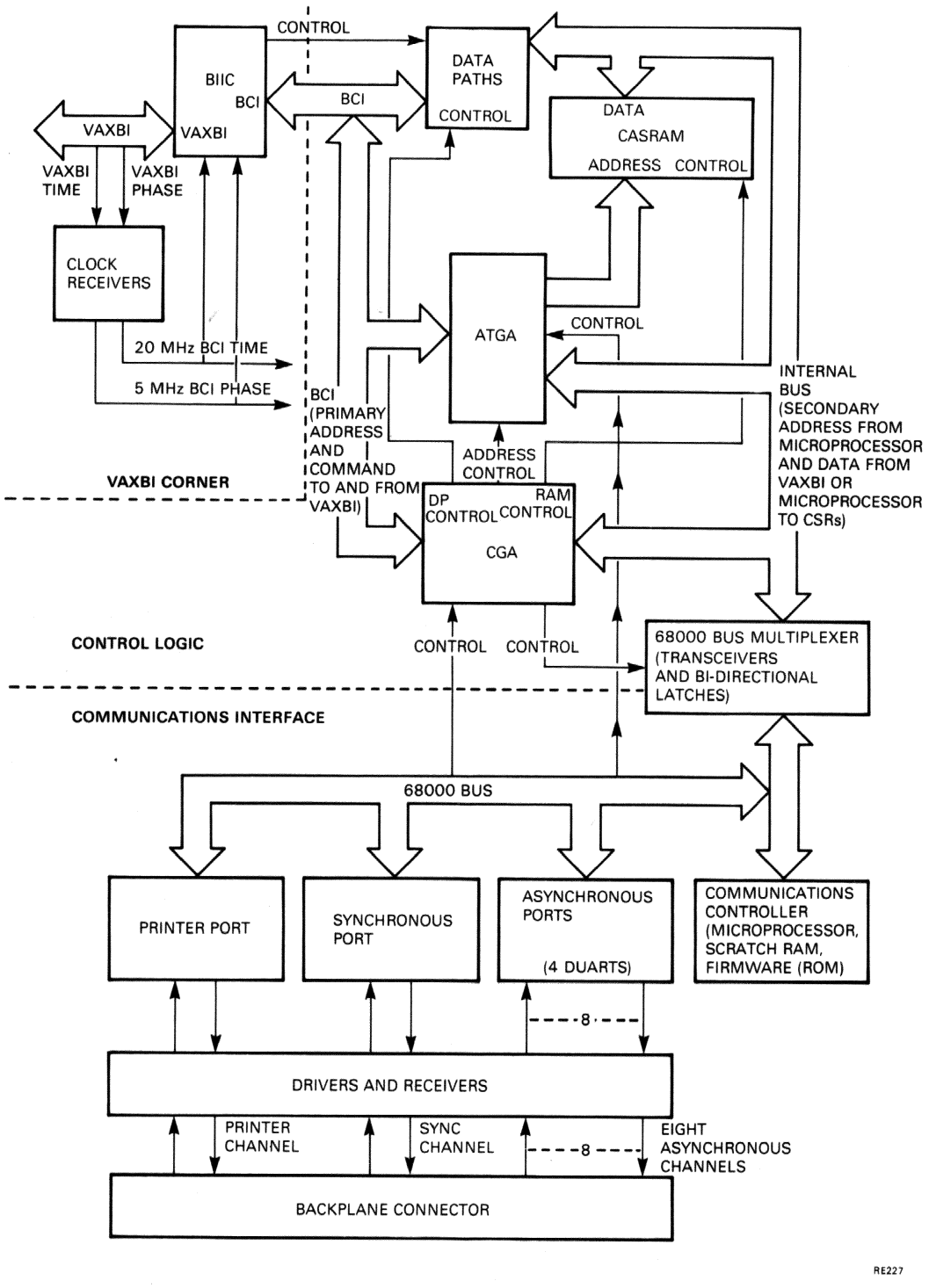


Figure 3-1 DMB32 Overview Diagram

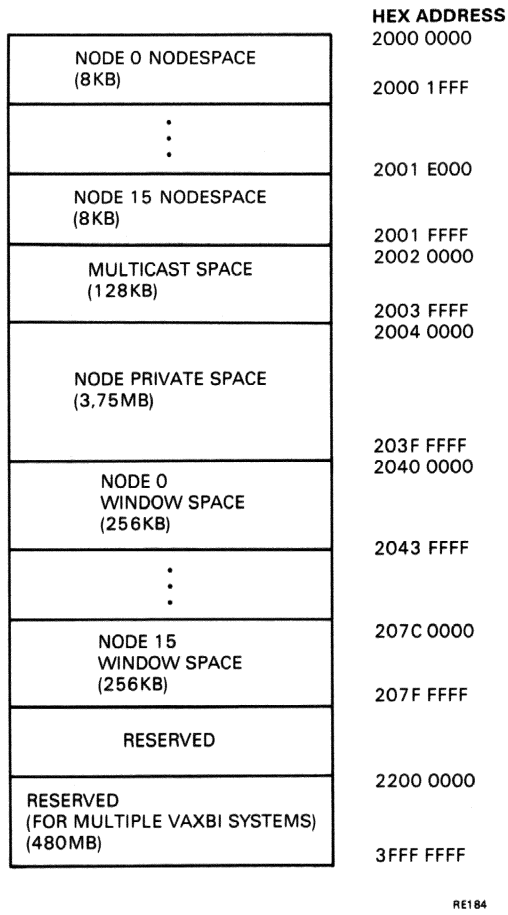


Figure 3-2 VAXBI I/O Address Space

- Error status bit for transfers on the VAXBI/BCI interface
- ID of nodes to which this node can send an INTR command and from which it can accept an IDENT command.

BIIC specific registers are not required by the VAXBI specification. They support specific functions implemented by this BIIC. The following data are available:

- The range of addresses, outside device register space, to which the device will respond as slave
- User interrupt control and status bits
- Enable/disable bits for the various VAXBI commands
- Flags to control use of general purpose registers.

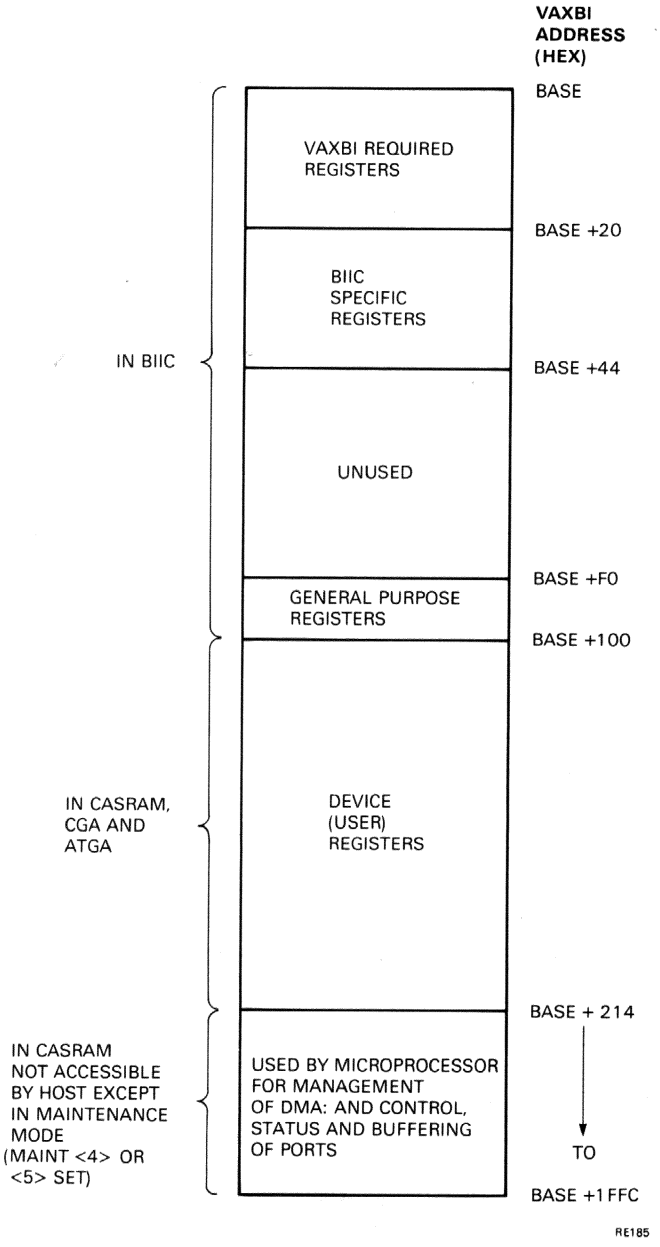


Figure 3-3 DMB32 Memory Map

The user registers are the CSRs via which the device is controlled. Most of the registers support the communications functions of the option, but some are used to support maintenance functions and the translation of virtual addresses. In broad terms, the functions of the user registers are:

- Transfer of data to/from the communications channels
- Configuration, control and monitoring of the communications functions
- Reporting of configuration status
- Holding system page table information (held by DMB32 to access system memory for address translation purposes)
- Support of maintenance functions and diagnostics.

3.3 DMB32 COMMANDS

Commands on the VAXBI demand an appropriate response from all connected nodes. At power-up or initialization, each node programs its BIIC to give the correct response for its node.

A node that is not addressed by a command, or that is unable to implement it, will respond with NO ACK.

An addressed node that can implement the command will respond with ACK, and will execute the command. If the node can implement the command but is not ready, STALL or RETRY responses will be issued until an ACK is generated.

The DMB32 does not implement the full set of VAXBI commands.

3.3.1 DMB32 Slave Response/Interpretation

The DMB32 supports the following VAXBI commands:

- Read
- Interlock read
- Read with cache intent
- Write
- Write with cache intent
- Unlock write – mask with cache intent
- Write mask with cache intent
- Ident
- Stop.

For more information see Section 3.17.1.

3.3.2 DMB32 Master Commands

In order to perform DMA transactions across the VAXBI, or to interrupt the host CPU, the DMB32 becomes bus master. As bus master it can issue only the commands listed below.

- Read
- Write
- Write Mask
- Interrupt (INTR).

All DMA transfers of communications data are octaword.

3.4 VAXBI CORNER

The specification of the electrical and physical parameters of options on the VAXBI bus is very critical because of the high bandwidth (13.3 Mbyte/s) of the VAXBI bus. Variations of impedance, signal delay, and other factors, can affect the proper operation of the bus. For this reason, all VAXBI adapters use a standard design and layout of the VAXBI bus interface. This common section of the module is called the VAXBI Corner.

3.4.1 Backplane Interconnect Interface Chip (BIIC)

The BIIC forms the major part of the standard interface for all VAXBI adapters and performs the following functions:

- Provides all VAXBI bus drivers and receivers (except for clocks and some async signals)
- Handles all aspects of VAXBI arbitration protocol
- Performs the address recognition function
- Handles VAXBI bus protocol and error-checking
- Handles all VAXBI transfers to/from the DMB32
- Can be programmed to provide parity for data and addresses from the DMB32
- Implements commands which are addressed to the VAXBI required registers
- Passes address, data, and command information addressed to the DMB32 registers
- Performs a BIIC self-test at power-up or reset, and disables its VAXBI drivers if the test fails.

Although the BIIC can implement all VAXBI command-types, the DMB32 only uses a subset of these. At initialization, the DMB32 firmware programs the BIIC for only those commands which are implemented by the DMB32. ACKs and certain other responses to VAXBI commands, can be made autonomously by the BIIC.

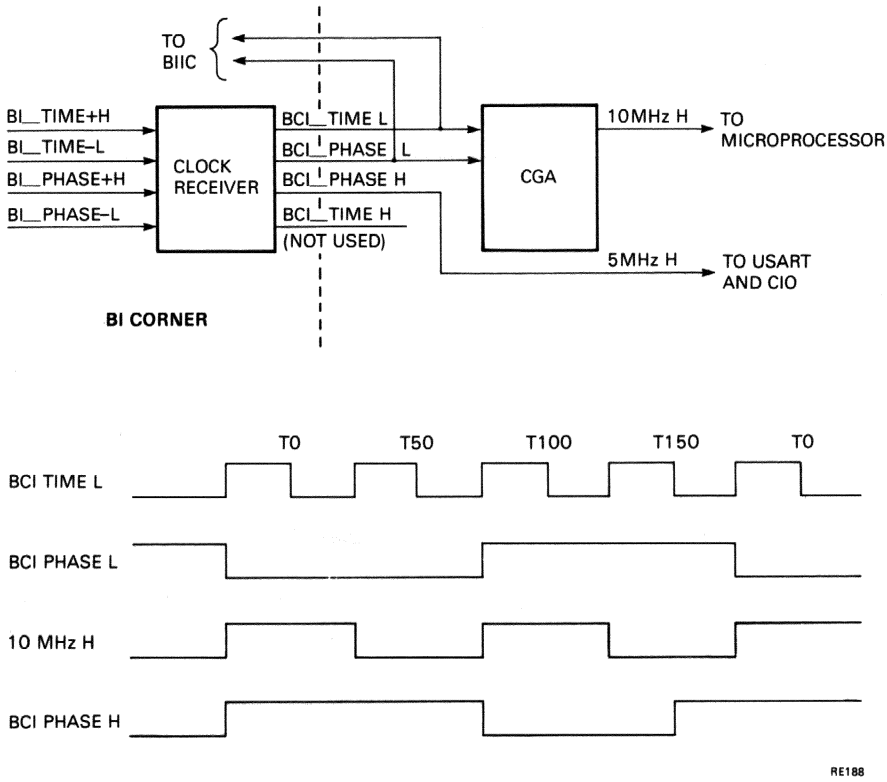


Figure 3-4 DMB32 Clock Circuits

3.4.2 Clock Circuits

The VAXBI bus carries differential square-wave clock signals at 20 MHz (BI TIME) and 5 MHz (BI PHASE). These are the basic timing elements for the VAXBI system and they connect directly to the clock receiver which produces BCI_TIME and BCI_PHASE signals at TTL levels. Figure 3-4 shows how the DMB32 clocks are derived.

The VAXBI corner layout provides for clock drivers and receivers to be mounted. Only one node of a VAXBI system is permitted to drive clock signals on the bus; clock drivers of all other nodes are either disabled or not mounted.

Drivers are not mounted in the DMB32, therefore the receivers are driven by clock signals from the VAXBI. These clock signals synchronize the DMB32 logic to operations on the VAXBI.

BCI_TIME L and BCI_PHASE L provide clocks for the BIIC and the CGA. BCI_PHASE H provides a clock for the 8530 USART and the 8536 CIO ICs.

A 10 MHz clock, derived by the CGA from BCI_TIME L and BCI_PHASE L, provides a clock for the microprocessor.

3.5 COMMON ADDRESS STORE RAM (CASRAM)

CASRAM is a 2K longword memory shared by the VAXBI host and the 68000 microprocessor. Access to CASRAM is arbitrated by the CGA, and addresses are translated by the ATGA.

CASRAM is used for DMB32 device registers and data buffering. It is mapped into the VAXBI system I/O space, its exact address is determined by the node ID plug on the backplane slot into which the specific DMB32 is installed.

On any VAXBI bus, up to 16 blocks of 2K longwords (8 Kbytes) can be defined as node space. Therefore, up to 16 VAXBI nodes can exist on a single bus.

Figure 3-5, the CASRAM block diagram, shows that the ATGA supplies a common 11-bit address to each 2 Kbyte RAM chip. Chip Select (CS) signals for each byte, and common Write Enable (WE) and Output Enable (OE) signals are supplied by the CGA. Data is written to and read from the RAM via the data bus, D<31:00> H.

For word or byte accesses, only the relevant CAS_CS L signals are asserted.

Timing of CASRAM control signals is dependent upon the type of transaction being performed. However, two examples of signal relationships (for CASRAM Read and Write transactions) are given in Figure 3-5.

3.6 ADDRESS TRANSLATION GATE ARRAY (ATGA)

The ATGA is an address multiplexer which also performs some address translation functions.

CASRAM ports that are multiplexed by the ATGA are:

- The primary access port – this is the VAXBI access port
- The secondary access port – this is the 68000 microprocessor access port.

The primary access port is input only. It carries addresses and command information received from the VAXBI.

The secondary access port accesses CASRAM via a multiplexed address and data bus. Addresses, which come from the microprocessor are input only. Data, from the microprocessor or the VAXBI host, is both input and output.

The ATGA contains on-board registers which are used to support its address translation function. These registers are a duplication of similar registers in CASRAM. There is very little control logic in the ATGA; strobes which select address or data, and enable signals for the primary or secondary access ports, are provided by the Control Gate Array (CGA).

3.6.1 Address Translation

Addresses from the VAXBI host or from the microprocessor are translated before they are used to address the CASRAM. Translation includes the indexing of addresses, selection of FIFO counters in the ATGA, and the inversion of microprocessor addresses.

3.6.1.1 Indexing – Each async port has some indexed registers and these are accessed by a field in ACSR; for example, each port's PREEMPT register can be accessed at the same node space address after the port number is written to ACSR<7:0>.

It is the ATGA's indexing function that combines the index in ACSR with the PREEMPT address to access the correct CASRAM location.

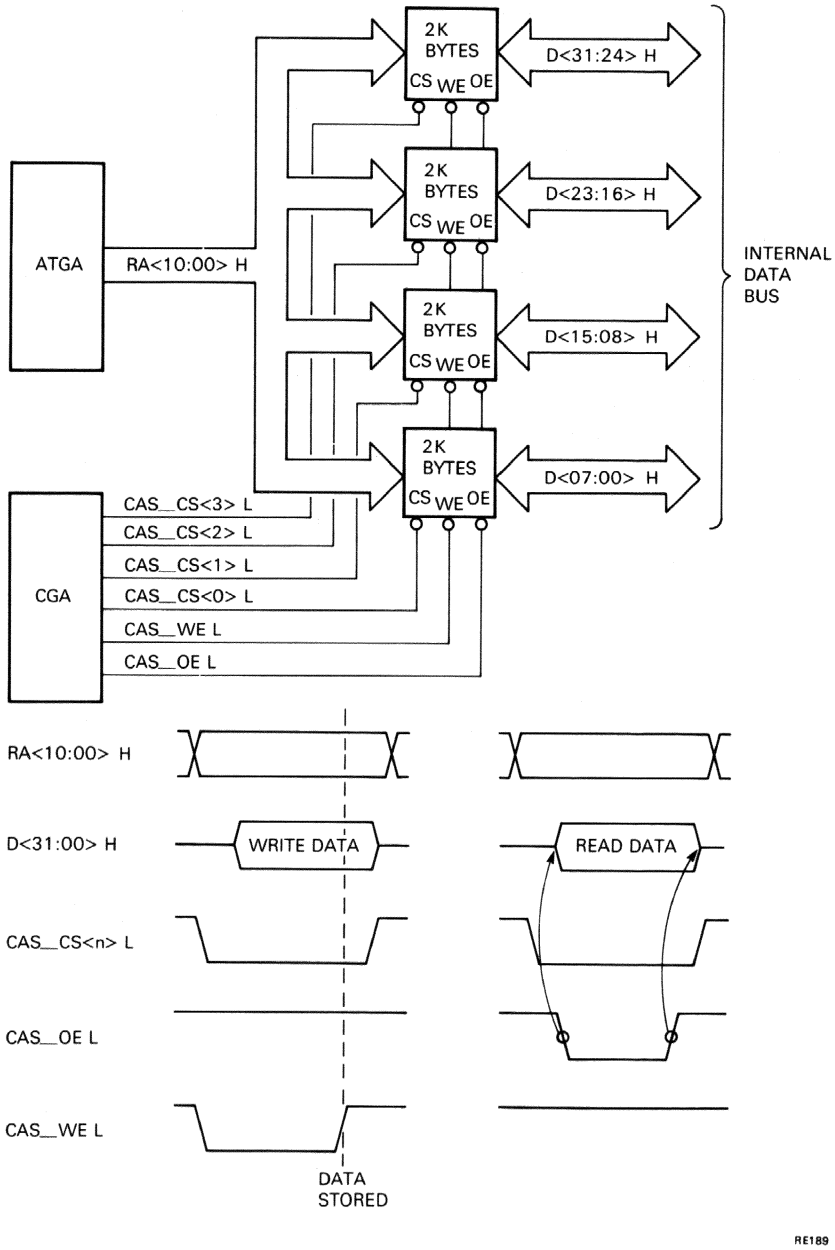


Figure 3-5 CASRAM Block Diagram

3.6.1.2 FIFO Controllers – The ATGA controls three FIFOs in CASRAM. These are:

- RX Data FIFO (512 longword entries)
- TX Completion FIFO (16 longword entries)
- Sync Line Completion FIFO (16 longword entries)

Each FIFO is accessed via a single location. When a FIFO address is recognized by the ATGA, an address from the appropriate FIFO counter maintained in the ATGA, is output to CASRAM.

1. The RX Data FIFO – (RBUF) is used for async port RX data and status, async modem status, and diagnostic reports.

2. The TX Completion FIFO – (TBUF) is used to queue reports of async channels which have completed, aborted or terminated transmission of a DMA buffer, or which have just emptied their preempt register.

3. The Synchronous Line Completion FIFO – (SBUF) is used to report that a DMA transfer on the sync channel has been completed with or without error, or that a modem status change has been detected.

3.6.1.3 Inversion of Microprocessor Addresses – CASRAM is a common resource for both the VAXBI host and the microprocessor. However, the significance of bytes within words, and words within longwords, is different for the 68000 microprocessor and VAX-11. To overcome this difference, addresses supplied by the microprocessor are inverted in the ATGA. This correction of the data structure is invisible to the user. Address inversion is described in Appendix B.

The ATGA also controls octaword DMA files in CASRAM (these are not accessible to the user). They exist for each sync, async, and printer channel.

3.6.2 ATGA Registers

By writing to specific registers in ATGA, the microprocessor can clear individual FIFOs or reset all ATGA registers to their initial power-on condition. Other ATGA registers hold information used by the ATGA and the microprocessor to provide CASRAM addresses, to manage the FIFOs, and for diagnostic use.

Information held by the registers includes:

- Channel number – Copies of the channel number from the async CSR and the sync CSR.
- Base addresses – Base addresses of FIFOs and the CASRAM areas allocated to sync and async registers. These values, are loaded at initialization.
- FIFO status – Status bits used for control of the FIFOs, generation of async RX data interrupts, generation of auto-flow characters and so on.
- DMA file – Points to the beginning of a DMA file address in CASRAM.

The ATGA, its registers and control signals, are described in more detail in Appendix B.

3.7 CONTROL GATE ARRAY (CGA)

The major functions of this array are:

- To control the BCI
- To control the ATGA and CASRAM

The BCI consists of a slave port and a master port. The slave port responds to valid VAXBI commands addressed to the node identified by the node identification plug on the BI backplane slot. Node and command recognition are performed by the BIIC.

By asserting a request on the BCI, the CGA can initiate a transaction across the VAXBI. Before causing the request to be asserted, the microprocessor must set up the appropriate DMA file in CASRAM and load the DMA File Address register in the ATGA. For a WRITE transaction, the DMA file must be loaded with the data to be transferred. The BIIC uses the VAXBI address from the DMA file, to address system memory.

The ATGA is presented with address and command information at its primary and secondary access ports. The CGA monitors both ports, and provides signals to enable the appropriate port and to control the transfer of information into and out of the ATGA. The primary access port has priority over the secondary access port.

To reduce the component count, much of the microprocessor memory/peripheral decoding logic is included in the CGA.

3.8 DATA PATHS

To comply with BCI timing requirements, a number of latches are used to transfer data between the BIIC and CASRAM. Before it is written to CASRAM, data from the BIIC is latched to make time for the BIIC to check parity, so that bad data is not written. Data is fetched from CASRAM and is held in the latches until the VAXBI needs it, thus avoiding the need to stall the VAXBI.

The above latching requirements are met by the data paths logic, which consists of latches controlled by the CGA and the BIIC. By use of these latches, a longword can be latched and enabled from the BCI to the internal bus, or from the internal bus to the BCI. (The CASRAM is on the internal bus D<31:00>). Figure 3-6 shows the bus arrangement.

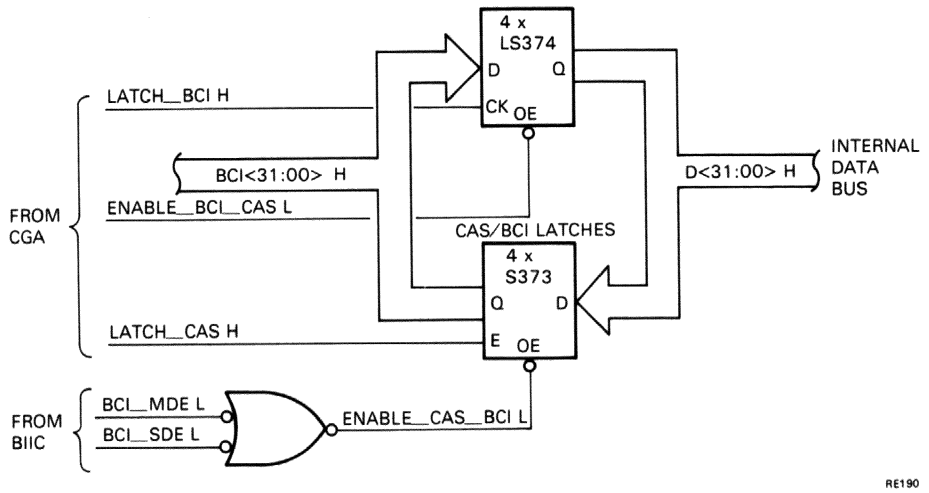


Figure 3-6 Data Paths Logic

The BCI/CAS latches are configured for input (master READ or slave WRITE) and the CAS/BCI latches are configured for output (master WRITE or slave READ).

In Figure 3-6 LATCH_BCI H, LATCH_CAS H and ENABLE_BCI_CAS L are supplied by the CGA. BCI Master and Slave Data Enable signals (BCI_MDE L and BCI_SDE L) come from the BIIC. ORing MDE and SDE, allows the BIIC to read the data as required by the VAXBI.

During any VAXBI command cycle, even those not addressed to this node, the command address is latched into the data paths. However, the latched address is not used except when the DMB32 is in maintenance mode (MAINT<4> or <5> set).

3.9 68000 BUS MULTIPLEXER

CASRAM is organized as longword memory but the 68000 microprocessor is a 16-bit (word) device. To allow access to and from the high and low words of CASRAM, the microprocessor data bus is switched by four LS245 transceivers onto both words of the internal data bus. The switched microprocessor data is time multiplexed with microprocessor addresses, using two LS646 latches. These latches are bi-directional to allow the VAXBI to access the microprocessor local bus (in maintenance modes only).

Figure 3-7 shows the relationship between the 68000 microprocessor bus and the internal bus, and identifies signals by which the CGA controls the Busmux.

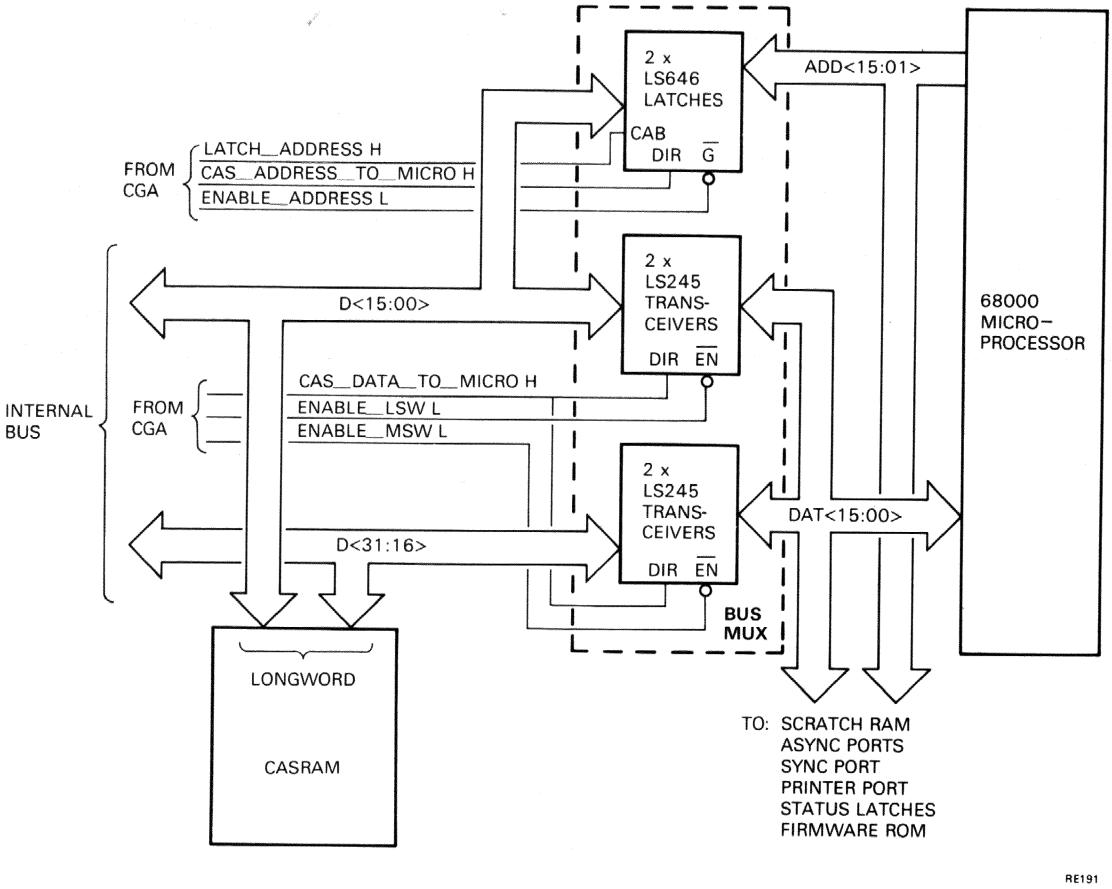


Figure 3-7 Multiplexing the 68000 Microprocessor Bus

The LS245s are transceivers. When ENABLE_LSW L or ENABLE_MSW L is asserted, the appropriate transceivers are enabled in the direction selected by CAS_ADDRESS_TO_MICRO H (Asserted = to microprocessor, de-asserted = from microprocessor). When transceivers are not enabled they are tri-stated.

The LS646s are transceivers with integral latches configured to achieve the functions defined in Table 3-1.

Table 3-1 Address Latch Truth Table

G	Dir	CAB	Function
H	x	H or L	Buses isolated
H	x	⌐¬	Latch address from D<15:00> H
L	H	H or L	Enable latched address onto ADD<15:01> H
L	L	x	Enable ADD<15:01> H, from the microprocessor onto D<15:01> H (Transparent state)

x = don't care

3.10 68000 MICROPROCESSOR

The 68000 microprocessor is a 16-bit device with an address range of 16 Mbytes (8 Mwords). In the DMB32, the top eight microprocessor address bits are not connected, therefore the address range is 64 Kbytes (32 Kwords). The microprocessor uses an 16 Kword ROM and a 2 Kword scratch RAM. The microprocessor manages DMB32 functions such as:

- Self test and initialization at power-up and reset
- Programming, configuring and monitoring the communications and printer ports in accordance with information written to the CSRs
- Managing data buffers and DMA queues in CASRAM, and transferring data between CASRAM and the communications and printer ports
- Controlling and monitoring modem and printer control and status signals
- Initializing DMA sequences to transfer sync port RX data to system memory, from DMA files in CASRAM
- Translating VAX-11 virtual addresses to physical addresses
- Sending flow control characters as required.

Microprocessor functions are performed in a priority sequence defined by the firmware.

3.11 FIRMWARE FLOW

The functions of the DMB32 are defined by firmware routines. Routines are performed according to a fixed priority. The microprocessor checks a list of tasks and takes the one at the top of the list; tasks at the top of the list have the highest priority.

To keep track of tasks, and the channels on which they have been performed, the microprocessor maintains flags and copies of status registers in its scratch RAM.

The structure of the priority loop is shown by Figure 3-8. Note that there is no routine to transfer characters between the USART and the DMA files in CASRAM. This is done under microprocessor interrupt control.

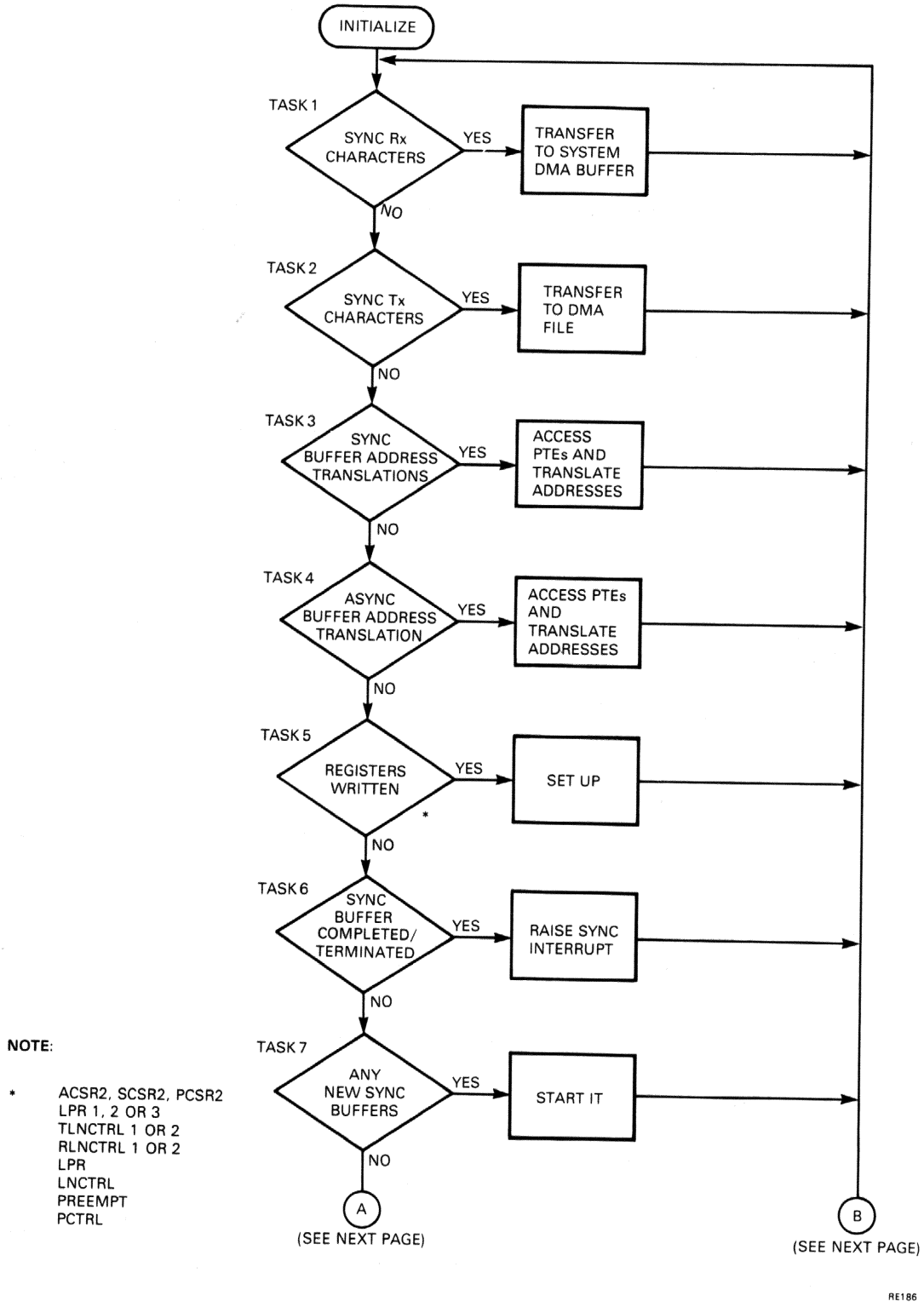


Figure 3-8 DMB32 Firmware Flow Diagram

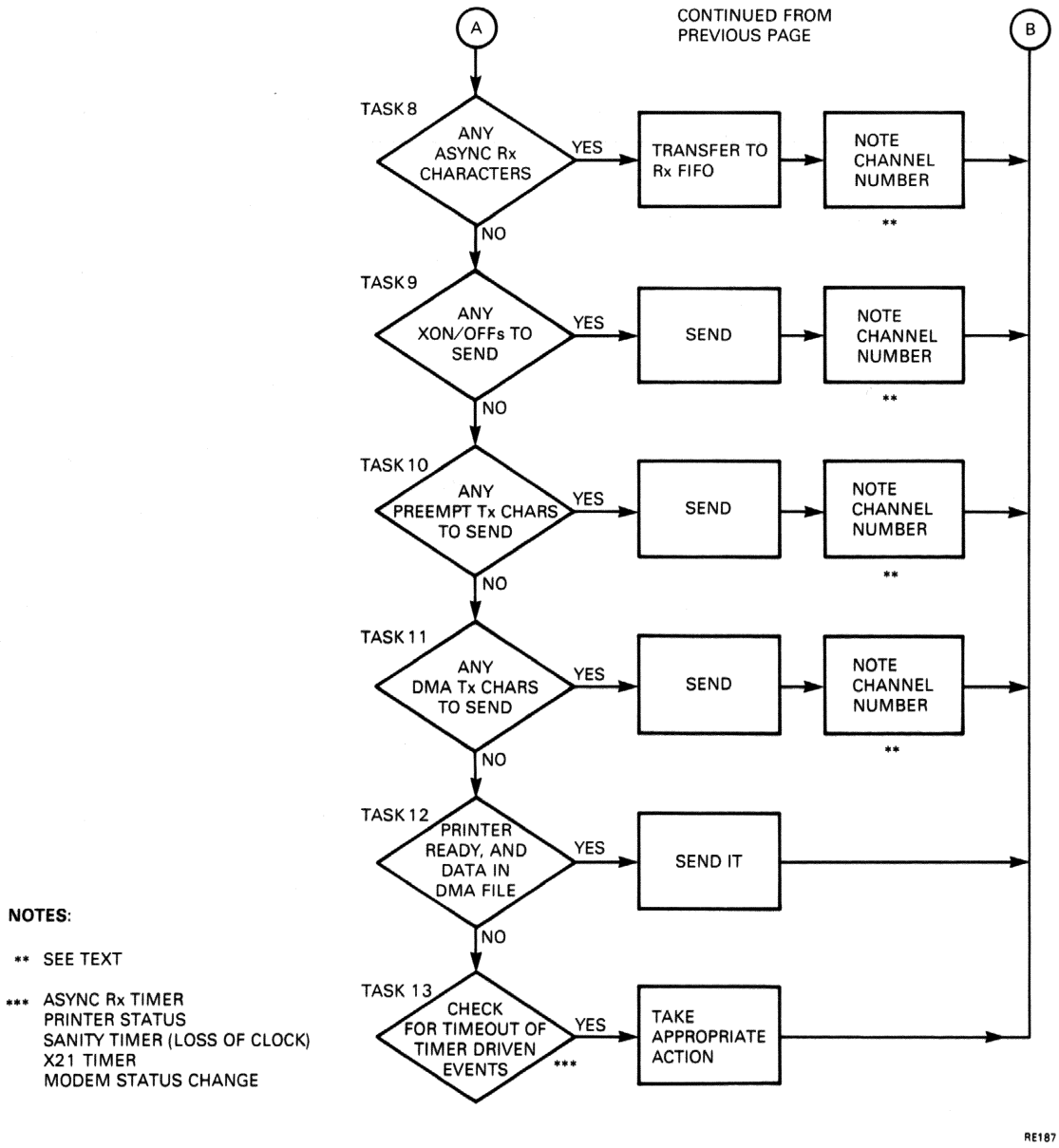


Figure 3-8 DMB32 Firmware Flow Diagram (continued)

The sequence of Figure 3-8 is as follows. Note that after any task is performed, the next task is task 1.

1. Is there a sync channel RX DMA file ready for transfer to system memory? If so, send it.
2. Is there a sync channel TX DMA file able to accept data from system memory? If so, get it.
3. Do any sync buffer addresses need translating? If so, access PTEs and do the translation.
4. Do any async buffer addresses need translating? If so, access PTEs and do the translation.

5. Have any of the listed registers* been accessed since the last check? If so, do any configuration and set up registers as necessary.

6. Has a sync DMA buffer completed/terminated? If so, and the sync interrupt is enabled, interrupt the host at the sync interrupt vector.

7. Is there a new sync DMA buffer to start? If so, start it.

8. Is there an async RX character ready in a DUART? If so, transfer it to the RX FIFO. Note the channel number**.

   If the FIFO was previously empty and the RXIE is set and no delay is programmed, interrupt the host at the async RX interrupt vector. If a delay is programmed, start the delay counter.

9. Is there an XON or XOFF to send? If so, send it. Note the channel number**.

10. Is there a preempt character to send? If so, send it. Note the channel number**. If TXIE is set, interrupt the host at the async TX interrupt vector.

11. Are there any async TX DMA characters to transfer to or from a DMA file? If so, and TX.ENA is set, do the DMA transaction and/or transfer one character to the DUART as required. Note the channel number**.

    If TXIE is set and transmission of the DMA buffer is complete, interrupt the host at the async TX interrupt vector.

12. Are there any printer characters to transfer to or from the DMA file? If so, and the printer is ready, do the DMA transaction and/or transfer one character to the CIO as required.

    If PR.I.E is set and transmission of the DMA buffer is complete, interrupt the host at the printer interrupt vector.

13. Have any timers timed-out?*** Take appropriate action for the relevant timer-driven function (For example if the RX timer has timed-out, raise the RX interrupt).

* When the host writes to any of the registers listed on the flowchart, bits are also set in a CSR Change register (CSR_CHGE) in the CGA. The firmware reads CSR_CHGE to check if registers have been written.

** Async channels are checked in a cyclic sequence, and a flag is set when an operation is performed on a channel. When the microprocessor returns to this step, the next channel will be checked first.

*** Modem and printer status lines are checked at intervals determined by timers in the 8536 IC. By polling the timers, the microprocessor can check the status as required. Timer intervals are:

10 ms – async port modem status
1 ms – sync port status

More information on the 68000 microprocessor is given in Appendix A.

3.12 ASYNCHRONOUS PORTS

Four type 2681 DUARTs (Dual UARTs) provide eight async serial data channels. The DUARTs are controlled and serviced by the microprocessor.

Each DUART has internal CSRs and data registers. By reading and writing these internal registers, the microprocessor transfers data, and configures and controls each async channel.

A common polled output tells the microprocessor when one of the DUARTs has assembled a receive character. Incoming data goes into the Rx FIFO.

Data and modem control lines of each DUART are connected via line drivers and receivers.

More details of the 2681 DUART are given in Appendix A.

3.13 SYNCHRONOUS PORT

The 8530 USART is a dual channel device. However, in the DMB32, some modem control and status lines are shared by the two channels, so only one channel can be selected at a time. In the DMB32, channel B is used for a CCITT V.35 standard interface and channel A is used for all the others. Both channels are connected to a 50-pin, miniature D-type male connector on the distribution panel. The female connectors on four different adapter cables supplied for RS-422, RS-423, V.24, and V.35, are wired to use only the appropriate pins.

The USART has internal CSRs and data registers. By reading and writing these registers the microprocessor transfers data, and programs the IC.

Functions which are programmable on the 8530 are:

- The method of signal encoding
- Type of protocol (see below)
- Flag and zero insertion and stripping
- CRC and parity generation and stripping

When it has assembled a received character, or is ready for a transmit character, the USART interrupts the microprocessor which then performs the transfer to or from the CASRAM.

Modem status lines are polled by the microprocessor.

Data and modem control/status lines are connected to the interchange circuits by line drivers and receivers.

Sync protocols supported by DMB32 are:

SDLC/HDLC	- bit oriented
IBM BISYNC	- byte oriented
DDCMP	- byte-count oriented
GEN BYTE	- protocol supported by the host

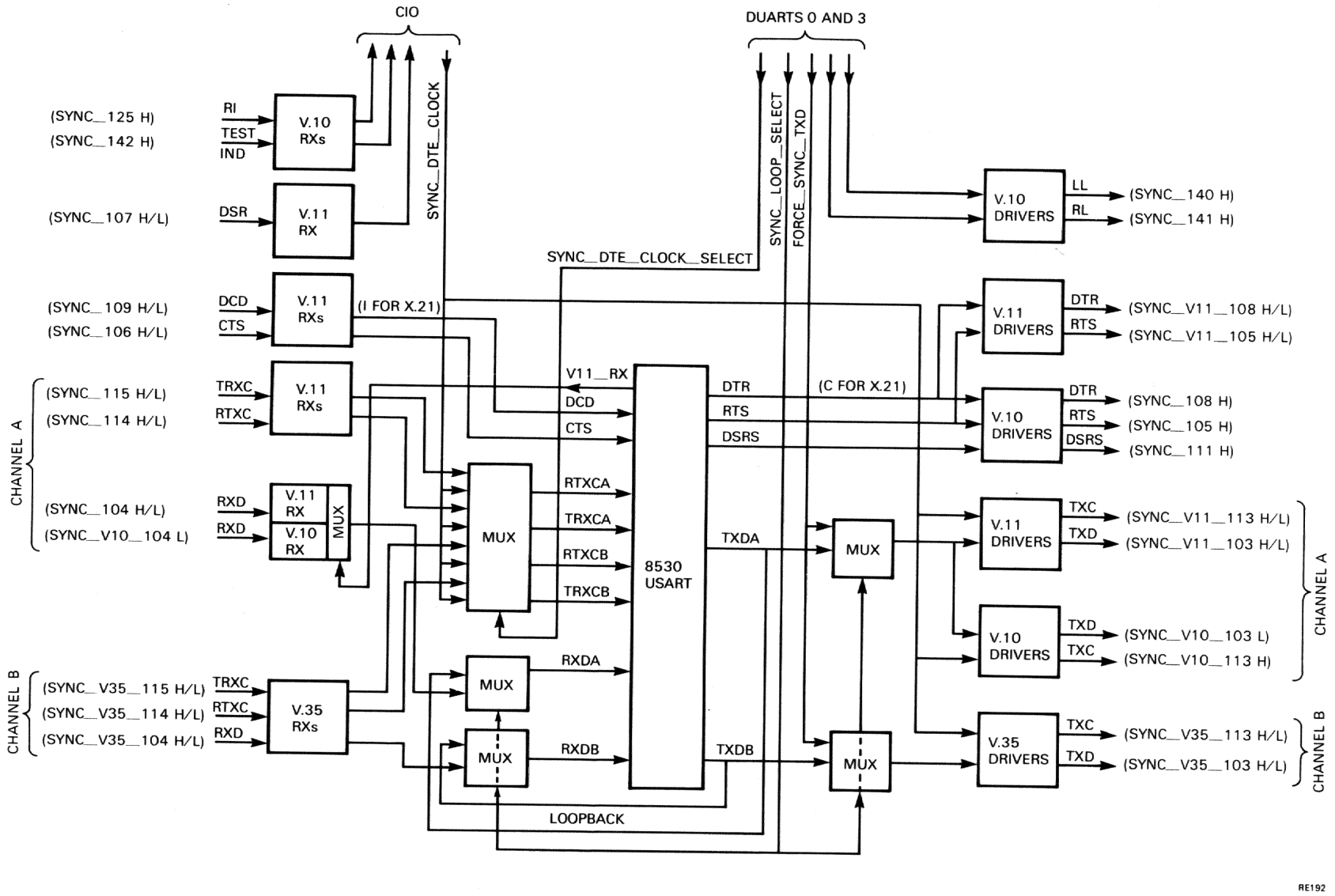


Figure 3-9 Control of Sync Channel Drivers and Receivers

Details of protocol support provided by the DMB32 are given in Chapter 2. More detail of the 8530 USART is provided in Appendix A.

All signals are made available or monitored at the sync interface connector, Except for the data and clock signals. A specific standard is selected by connecting the appropriate adapter cable to the interface connector.

Figure 3-9 shows several sync interface and modem control signals provided by the CIO and DUARTs 0 and 3. These are:

- SYNC_DTE_CLOCK L is an internally generated clock intended for direct DTE to DTE connections

- SYNC_DTE_CLOCK_SELECT H selects either the internally generated clock, or the clocks received from the modem

  Asserted = Internally generated clock to USART
  De-asserted = Modem clocks to USART

- FORCE_SYNC_TXD H provides a steady 1 or 0 condition on the TXD lines during internal loopback tests

  Asserted = Steady 1 condition
  De-asserted = Steady 0 condition

- SYNC_LOOP_SELECT H connects TXDA and TXDB to RXDA and RXDB for internal loopback tests, and outputs FORCE_SYNC_TXD on TXDA and TXDB

  Asserted = Loopback ON, and FORCE_SYNC_TXD H out
  De-asserted = Loopback OFF, and TXDA and TXDB out

- V11_RX H enables the V.11 (balanced) or V.10 (unbalanced) data receivers

  Asserted = V.11 receiver enabled
  Deasserted = V.10 receiver enabled

3.14 PRINTER PORT AND DISCRETE I/O

An 8536 IC and two LS374 latches form a common interface for the printer, for modem status signals, and for discrete I/O signals.

The 8536, which is covered in Appendix A, is a counter/timer and parallel I/O chip (CIO) that has three parallel ports and three counter/timer circuits. Figure 3-10 shows the use of the ports, of which A and B are byte-wide, and C is 4-bits wide. Note that some lines of port C are programmed as inputs and some as outputs. The sync port internal clock (SYNC_DTE_CLOCK L) is driven by the counter/timers.

CODE_SEL<7:0> is used to identify the type of sync cable that is connected.

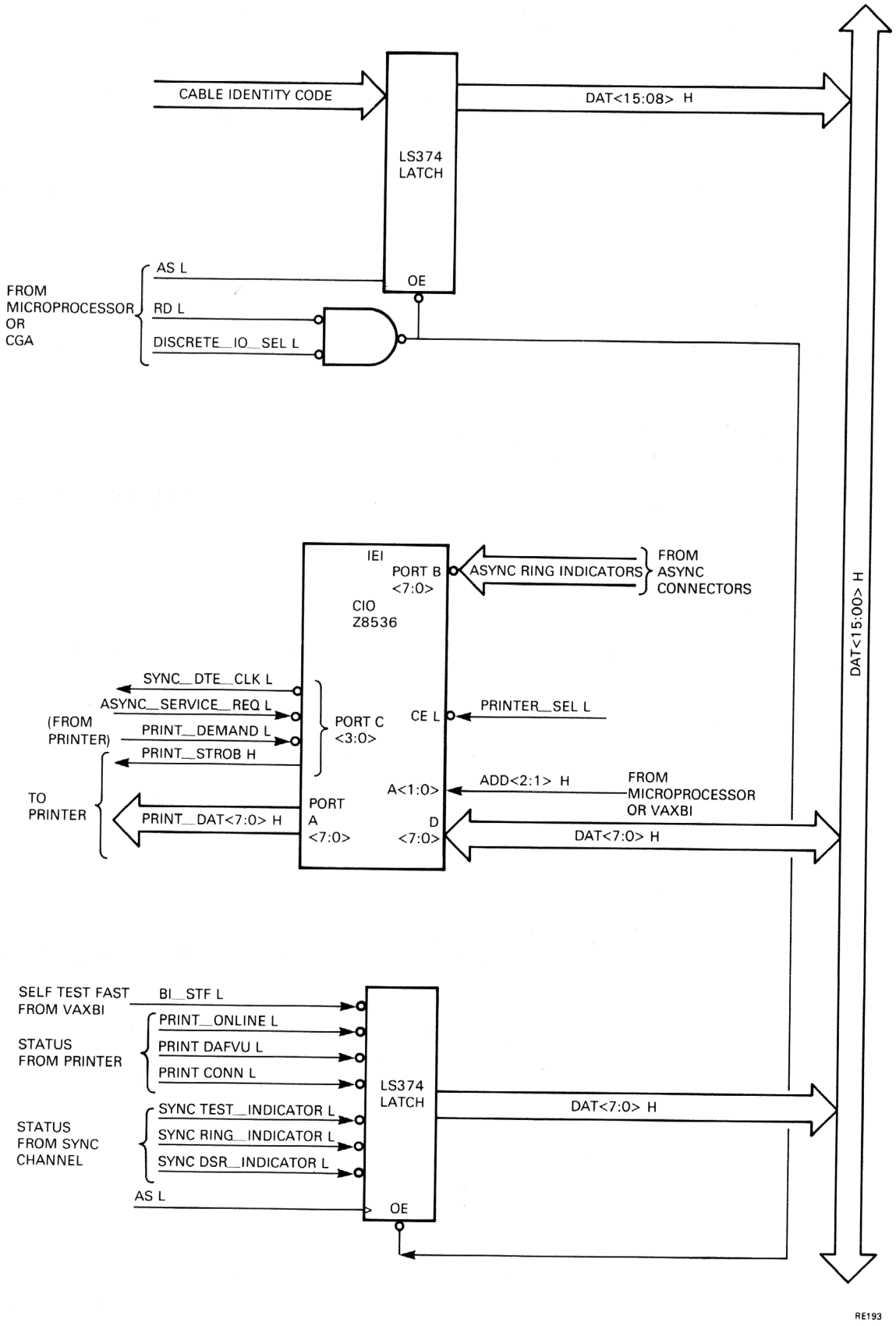


Figure 3-10 Printer Port and Discrete Logic

3.15 OUTPUT/TTL LEVEL CONVERSION

Line drivers and receivers convert between TTL levels at the sync and async port logic, to output levels on the communications lines.

Drivers and receivers (printer status) on the printer lines operate at TTL levels only, so there is no conversion.

3.16 COMMUNICATIONS INTERFACE

The communications interface provides an interface between the DMB32 and the sync, async, and printer channels. It consists of an interface controller and a number of peripheral I/O ICs. See Figure 3-11.

The USART and CIO are connected to DAT<7:0> to allow their interrupt capability to be used. This is because the microprocessor reads interrupts on the low data byte only.

3.16.1 Address and Signal Decoding

Much of the microprocessor address and control signal decoding logic has been built into the CGA. Figure 3-12 includes an equivalent circuit of control signal decoding.

Address decoding in the CGA is in two parts. ADD<15:13> selects the control signals to be generated by the CGA. ADD<15:12> selects how many wait states are to be inserted in the present microprocessor transaction cycle. This is done by delaying the assertion of DTACK L for a number of cycles.

Table 3-2 relates addresses to devices, control signals, and wait states. Note that a wait state is a full period of the 10 MHz clock.

Table 3-2 CGA Decoding of Microprocessor Addresses

Microprocessor Address (Hex)	Devices	Signals Generated	Default Wait States
XX0000 to XX7FFF	firmware ROM		0
XX8000 to XX9FFF	scratch RAM	RAM_SEL L	0
XXA000 to XXDFFF	CASRAM	*	1**
XXE000 to XXEFFF	DUARTs	IO_SEL L	1
XXF000 to XXFFFF	USART, CIO	IO_SEL L	2
XXF800 to XXFFFF	discrete IO	IO_SEL L	2

* Appropriate combination of CAS_OE L, CAS_WE L and CAS_CS<3:0> L for the type of CASRAM access.

** In addition to any wait states caused by CASRAM arbitration.

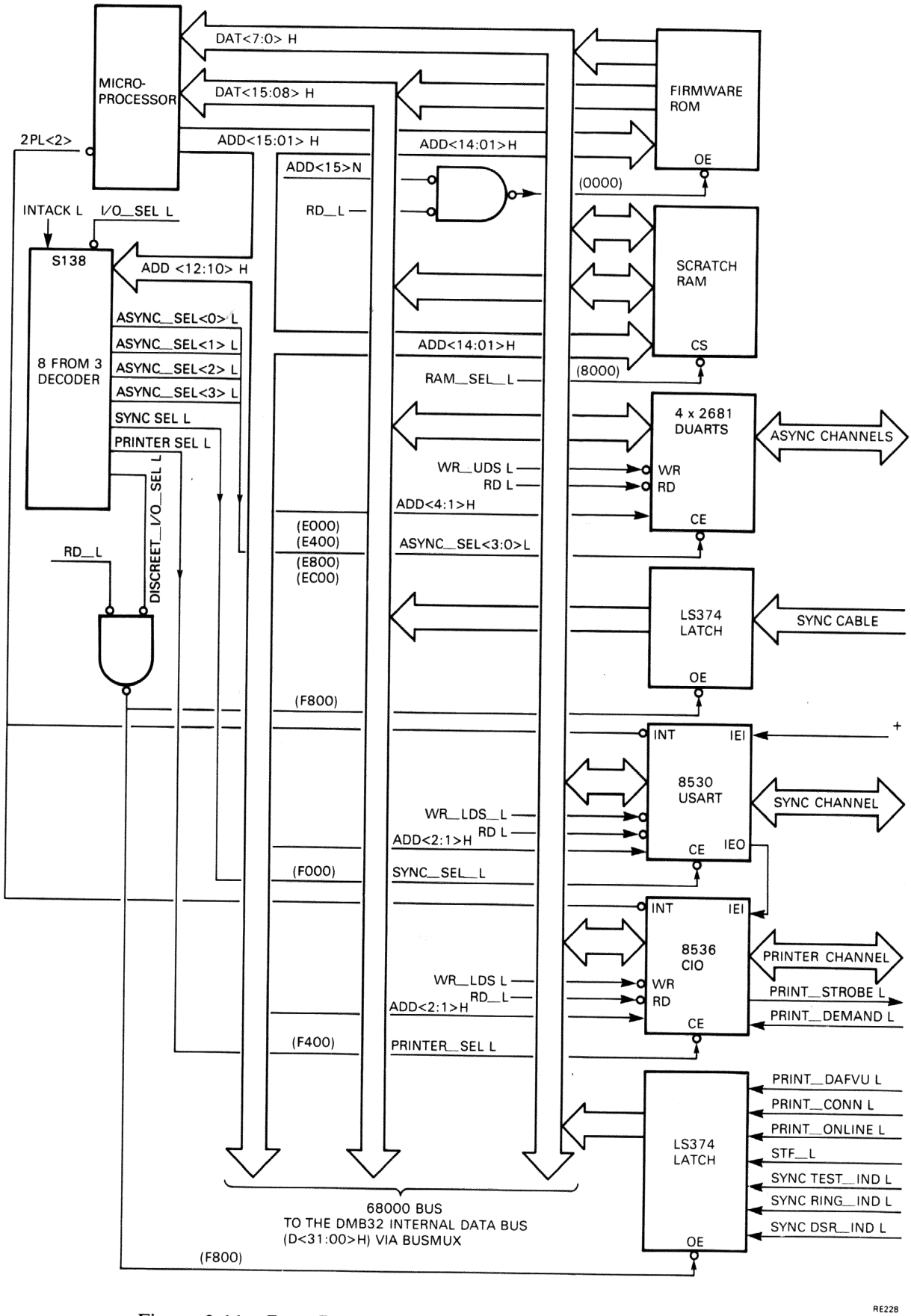


Figure 3-11 Data Bus Arrangement – Communications Interface

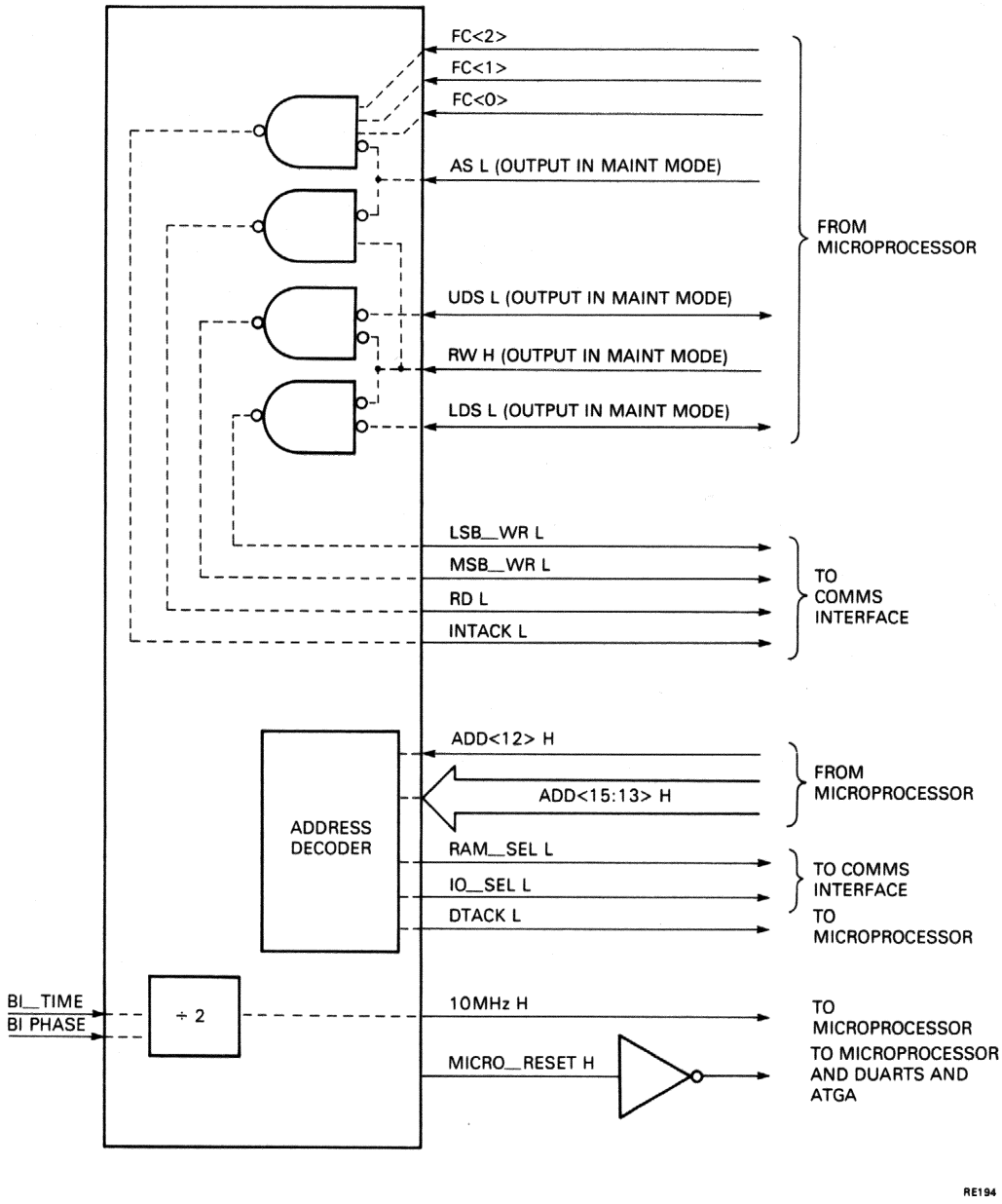


Figure 3-12 68000 Microprocessor Bus Address and Signal Decoding

3.17 BCI INTERFACE

Figure 3-13 shows the architecture of the DMB32 BCI interface which is formed by the data path logic and the BCI sides of CGA and the ATGA.

The BCI interface has two ports, one for master transactions and one for slave transactions. The ports are separate, making it possible for the master port to perform a data transaction to or from the slave port. This type of transfer (VAXBI loopback) is used during diagnostic procedures. Timing of data transfers across the interface is controlled by the BIIC.

In Figure 3-13 the interface connections and the function of each line are identified. Note that BCI_D<31:00> H, BCI_I<3:0> H, BCI_EV<4:0> L and BCI_P0 H are common to both ports and that the BIIC alone is responsible for enabling these lines, allowing the slave and master ports to share these signals.

A description of DMB32 BCI interface signals is provided in the CGA and ATGA material in Appendix B.

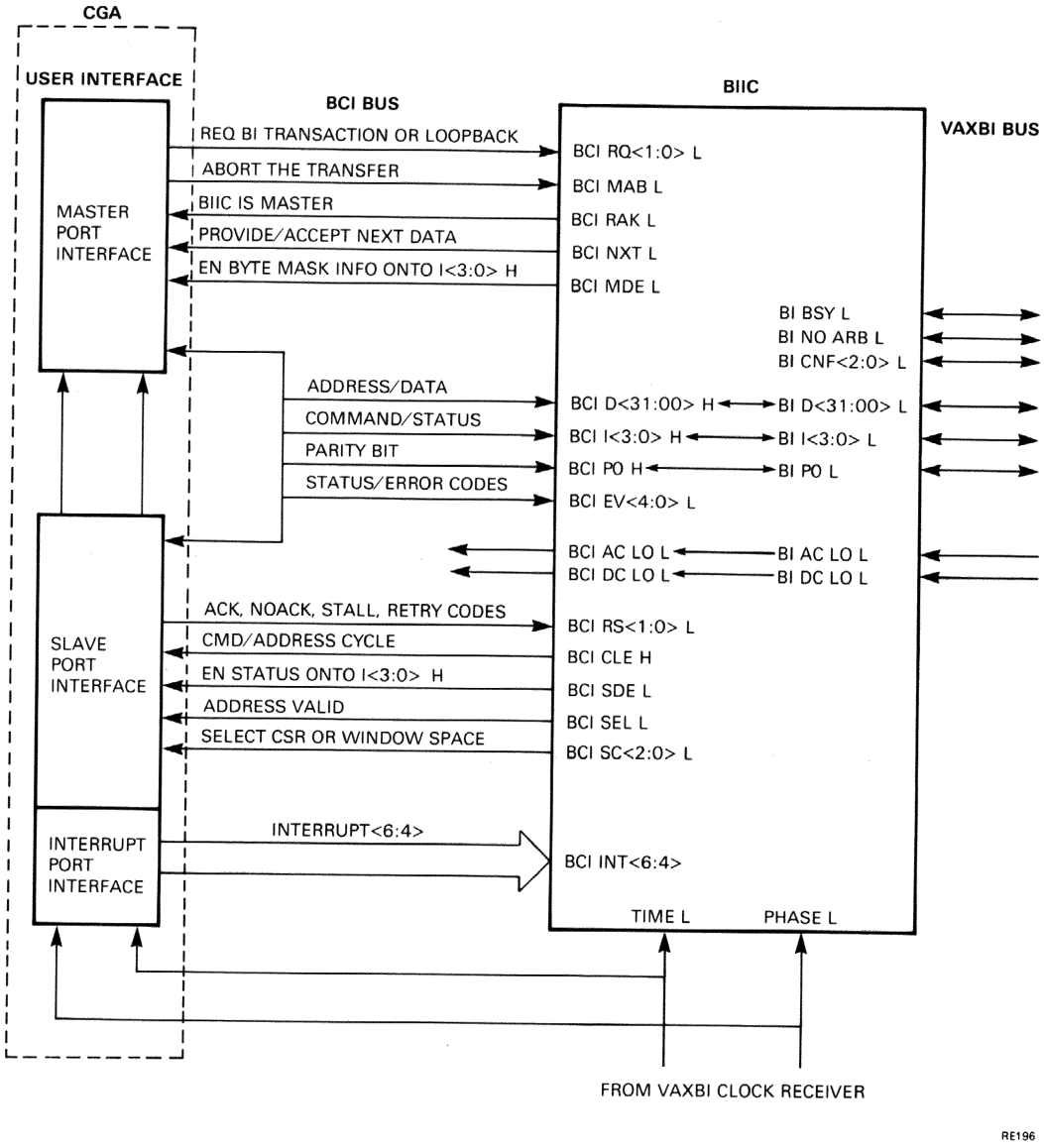


Figure 3-13 DMB32 BCI Interface

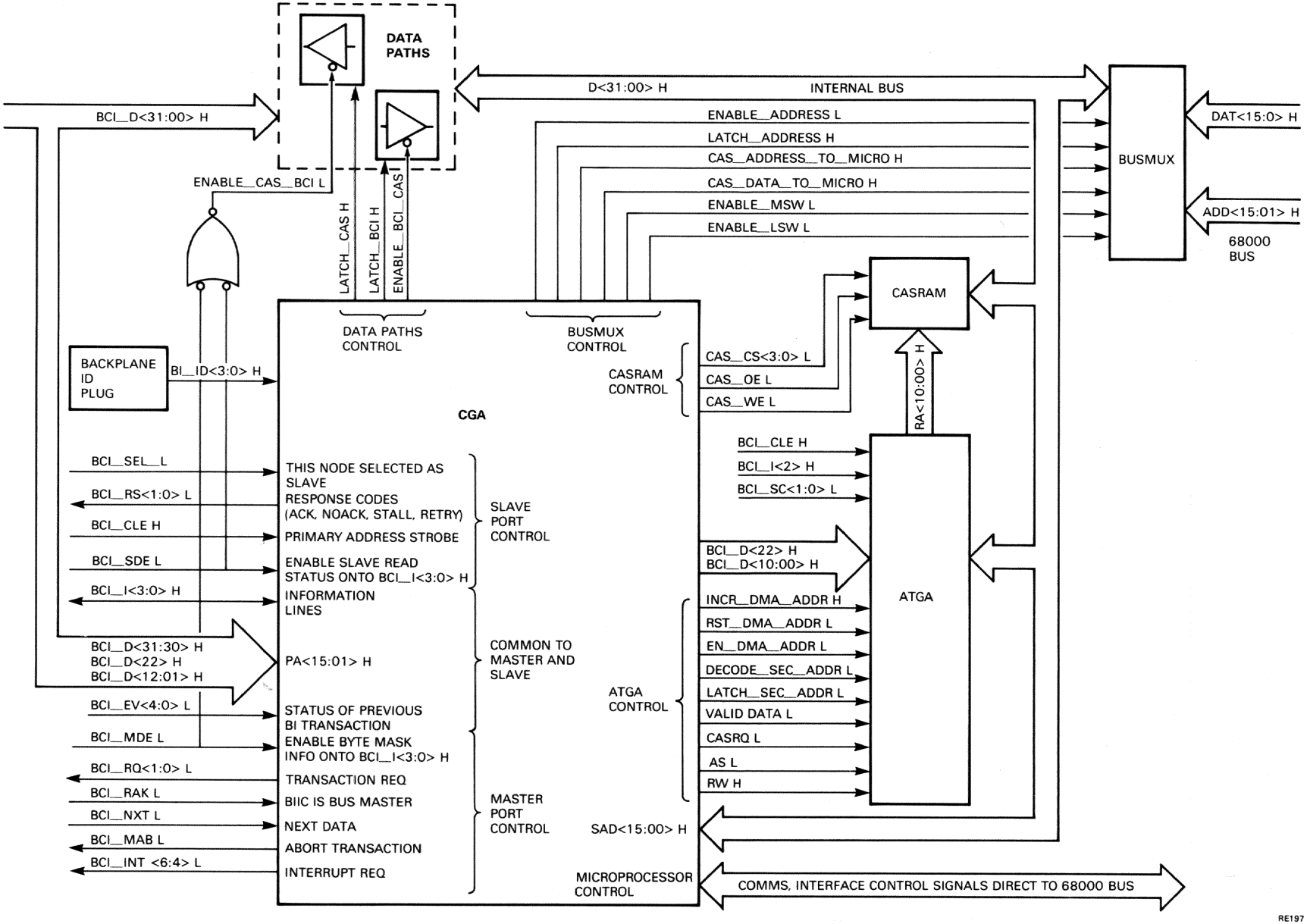


Figure 3-14 Control and BCI Interface Logic

In this section, the operation and interaction of the BIIC, the gate arrays, CASRAM, and the data paths are described. Timing diagrams are provided for each type of transfer. Some VAXBI signals are included in the diagrams for reference only and are not described.

Figure 3-14 is an overall block diagram of the logic which illustrates the overall control exerted by the CGA. It contains details of all relevant signal lines except those of the communications interface, which are covered in Section 3.15. Signal lines are grouped according to their function.

As in all VAXBI transfers, the number of longwords to be transferred is indicated by BI_D<31:30> during a VAXBI command address cycle.

In the DMB32, all DMA buffers larger than one octaword in length are transferred as a series of octaword transactions. If the buffer is not aligned on octaword boundaries some of the data will not be required. For a Write transaction, the unwanted data is masked out. For a Read transaction the complete octaword is transferred to the DMA file, but only the relevant data is transferred to the I/O ports. The unwanted data is overwritten when the DMA file is used again.

3.17.1 DMB32 Slave Transactions

The DMB32 supports Read and Write longword transactions from the VAXBI; byte and word write operations use masked longword transactions. The only other transactions supported by the DMB32 are the IDENT transaction following an interrupt request, and the STOP transaction.

The DMB32 looks at the BCI I<3:0>H command lines and in addition checks the BCI D<31:30>H lines during the command address cycle. If the function translates to a read or write, then the BCI D<31:30> lines are checked to see if the transfer is a longword; if it is, then the BCI SEL L line is checked to ensure this node is addressed. Any other transfers are invalid, and will be NO ACKed by the DMB32. Table 3-3 shows the action taken when any VAXBI command is received.

Table 3-3 DMB32 Response to VAXBI Commands

Command Code BCI I<3:0>	Command Mnemonic	BCI D <31:30>	Action Taken
0000	Reserved	x x	NO ACK
0001	READ	L H	Read Longword
0010	IRCI (Interlock read with cache intent)		Translates to a Read Function as above
0011	RCI (Read with cache intent)		Read performed as above
0100	WRITE	L H	Write Longword
0101	WCI (Write with cache intent)		Write performed as above
0110	UWMCI (Unlock write mask with cache intent)		Performs Write Mask (No Interlock)
0111	WMCI (Write mask with cache intent)		Performs write mask
1000	INTR (Interrupt)	x x	NO ACK

Table 3-3 DMB32 Response to VAXBI Commands (continued)

Command Code BCI I<3:0>	Command Mnemonic	BCI D <31:30>	Action Taken
1001	IDENT (Identify)	x x	Interrupt vector sent
1010	Reserved	x x	NO ACK
1011	Reserved	x x	NO ACK
1100	STOP	x x	ACK Microprocessor halted
1101	INVAL (Invalidate)	x x	NO ACK
1110	BDCST (Broadcast)	x x	NO ACK
1111	IPINTR	x x	NO ACK

x = don't care

A slave transaction is started when a VAXBI command/address cycle addressed to this node is recognized by the BIIC. A VAXBI command/address cycle is recognized from the preceding VAXBI cycle by the assertion of BI_NOARB L and the de-assertion of BI_BSY L during that cycle.

The current VAXBI bus master puts a command on BI_I<3:0> L, and an address on BI_D<31:00> L. When a BIIC recognizes a command/address cycle on the VAXBI, it asserts BCI_CLE H and transfers the command/address information from the VAXBI onto BCI_I<3:0> H and BCI_D<31:00> H.

The gate arrays use the de-asserting edge of BCI_CLE H to latch the command/address information from the BCI_I<3:0> H and BCI_D<31:0> H lines. At the same time, the CGA asserts LATCH_BCI H to latch the address into the data paths. Although the BCI address information is always latched it is only used during VAXBI maintenance procedures.

If the VAXBI address selects the slave node the BIIC asserts BCI_SEL L and a select code on the BCI_SC<2:0> L lines. When the CGA recognizes BCI_SEL L, it de-asserts DECODE_SEC_ADDR L to enable VAXBI access to CASRAM (by default, the microprocessor has access). This causes the address, translated as necessary by the ATGA, to be output on the CASRAM address bus RA<10:00> H. If no assertion of BCI_SEL L is detected by the CGA, the interface returns to an idle state.

The CGA verifies that the addressed location selects an implemented CSR before selecting the CASRAM. If the address selects a non-implemented CSR then the CGA will return a NOACK response on the BCI_RS<1:0> and the interface returns to an idle state.

3.17.1.1 Slave Read Transaction – During the VAXBI arbitration cycle, the CGA issues the STALL code on BCI_RS<1:0> L (which the BIIC then drives on BI_CNF<2:0>L), while the CASRAM chips are enabled by CAS_CS<3:0> L and CAS_OE L, and the selected data is latched into the data paths by the CGA assertion of LATCH_CAS H.

VALID_DATA L is asserted with CAS_OE L and is used by the ATGA, or CGA,to enable a selected internal register onto the least significant word of the CASRAM data bus. If an internal gate array register is selected the CGA will not assert CAS_CS<1:0>, thus disabling the least significant word of CASRAM.

During the following cycle, the BIIC enables the data path latches onto BCI_D<31:00> H, and the CGA supplies a READ DATA STATUS code on BCI_I<3:0> H together with an ACK code on BCI_RS<1:0> L.

The BIIC enables the contents of the data paths latches onto BCI_D<31:00> H by the assertion of BCI_SDE L. Note that BCI_SDE L is asserted twice because the first data cycle is stalled, the data being transferred by the second assertion.

This information is output onto BI_D<31:00>L and BI_I<3:0>L. The ACK is transferred to the VAXBI on BI_CNF<2:0>L. Parity is controlled by the BIIC.

At the end of the data cycle, the CASRAM chips are deselected and the ATGA microprocessor port is enabled again.

Figure 3-15 gives timing details of a VAXBI master reading data from the slave DMB32.

3.17.1.2 Slave Write Transaction – During the VAXBI imbedded arbitration cycle, the CGA issues the ACK code on BCI_RS<1:0> L, and the CASRAM chips are enabled by CAS_CS<3:0> L.

After this imbedded arbitration cycle, the VAXBI master places data on BI_D<31:00> L, and for a write-mask transaction, a byte mask on BI_I<3:0> L. The selected slave BIIC passes this information to the BCI_D<31:0> H and BCI_I<3:0> H lines at the end of the VAXBI cycle.

The BCI data is latched into the data paths by LATCH_BCI H while the byte mask information is latched by the CGA. The data is enabled onto the CASRAM data bus by the assertion of ENABLE_BCI_CAS L, by the CGA.

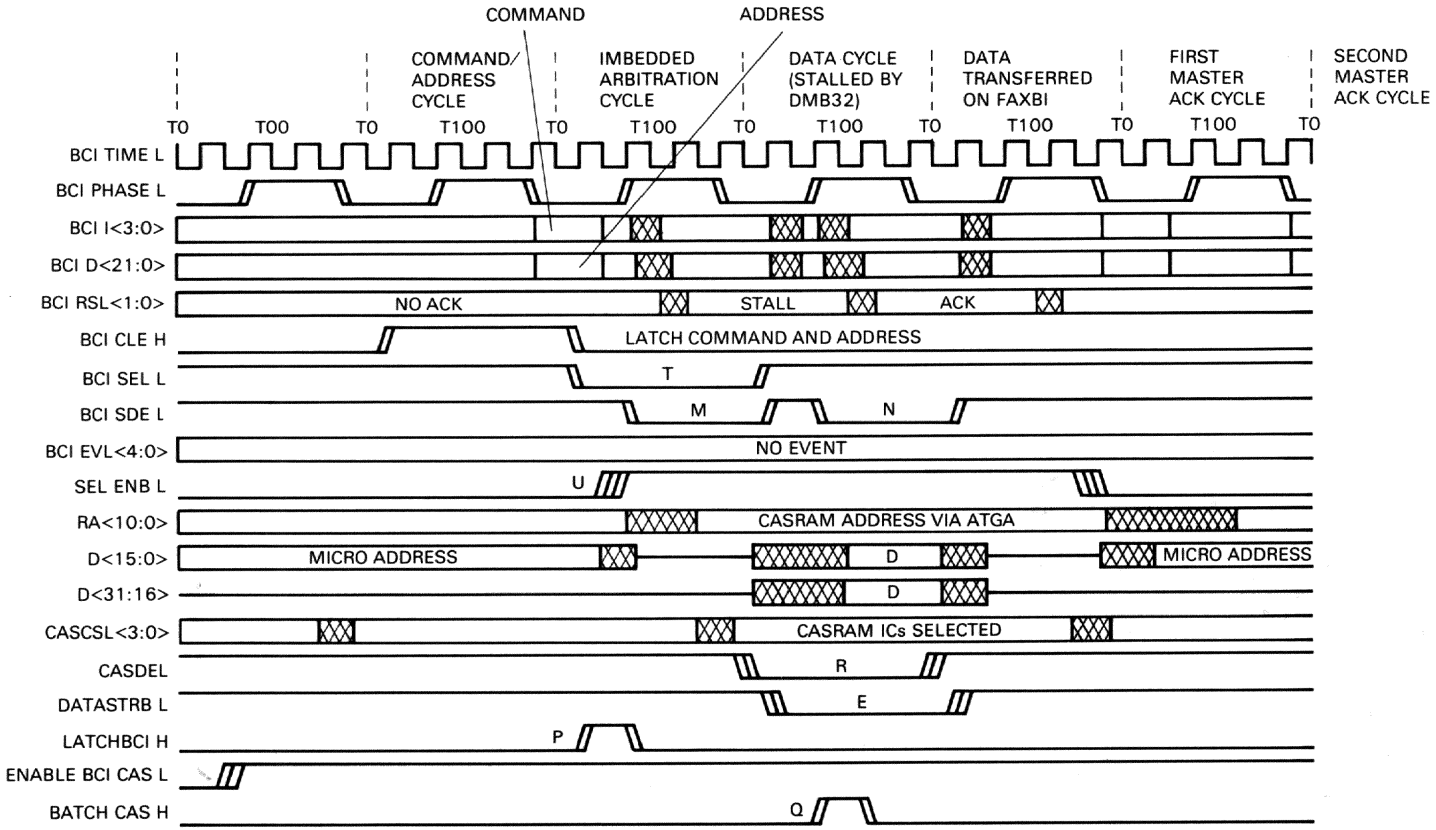
The BIIC checks the parity of the VAXBI bus during data cycles and if bad parity is detected, will assert a BPR event code on BCI_EV<4:0> L. This occurs during the VAXBI cycle following the detection of bad parity. If bad parity is not detected the CGA writes the data to the addressed CASRAM location by asserting CAS_WE L. Masked writes are implemented by using the byte mask to de-assert one or more of CAS_CS<3:0> L as required.

VALID_DATA L is asserted with CAS_WE L and is used by the ATGA, or CGA, to write to a selected internal register from the least significant word of the CASRAM data bus. If an internal gate array register is selected the CGA will still assert CS<1:0>, thus writing the least significant word into CASRAM. This permits the contents of some write only registers within the gate arrays to be read from CAS.

At the end of the write, the CASRAM chips are deselected and the ATGA microprocessor port is enabled again. The CGA completes the transaction with two ACK cycles, using BCI_RS<1:0> L, to indicate a successful transfer of data.

If bad parity is detected the CGA will not assert CAS_WE L and will return to the idle state issuing NOACKs on the BCI_RS<1:0> L lines.

Figure 3-16 gives timing details of a VAXBI master, writing data to the DMB32



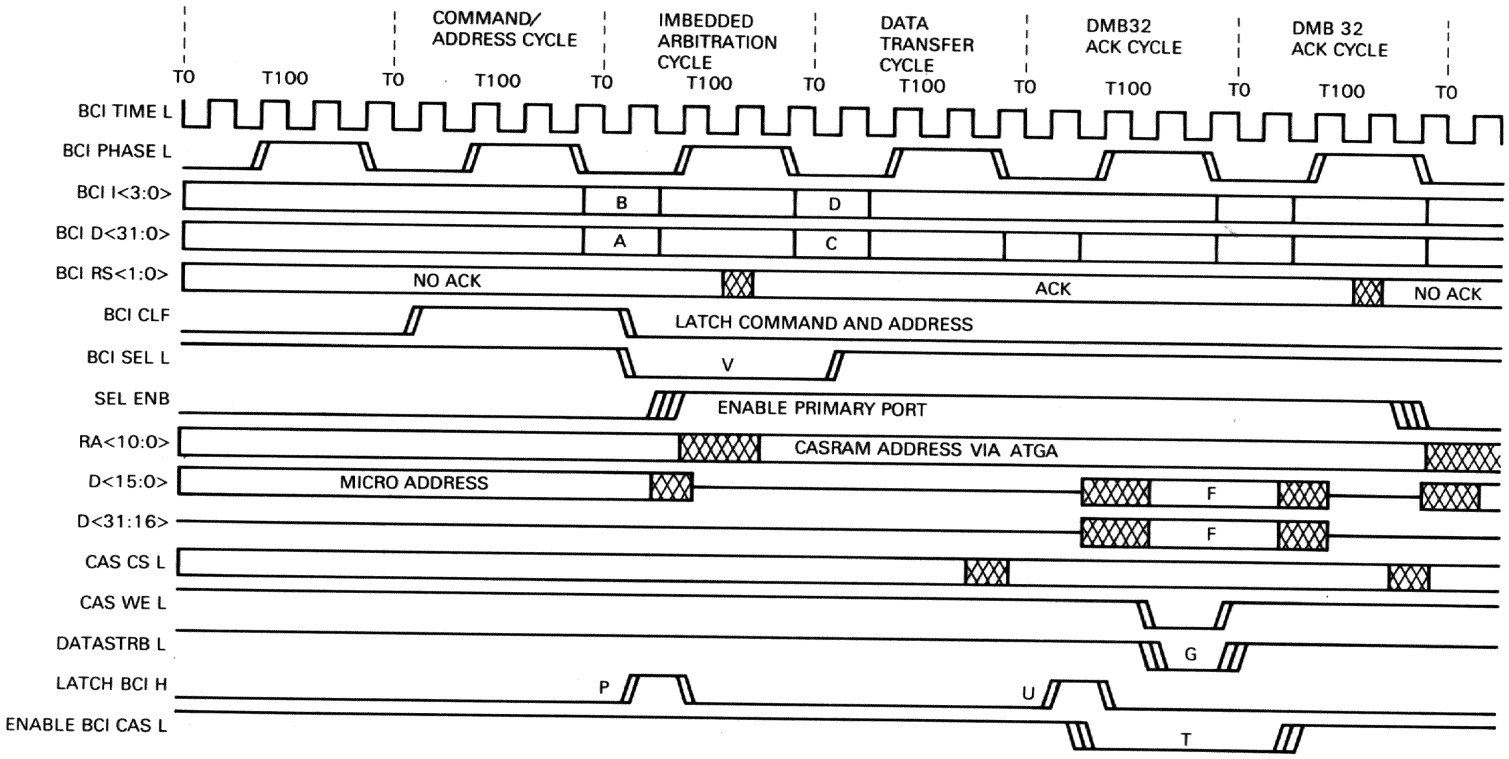
LEGEND

D = DATA OUTPUT
E = ENABLE DATA FROM GA REGISTER
F = DATA STATUS
T = THIS NODE SELECTED
U = SELECT PRIMARY (BI) PORT
X = UNDEFINED

M = ENABLE DATA PATHS ONTO BCI D<31:0>H
N = REPEAT BECAUSE PREVIOUS CYCLE STALLED
P = LATCH VAXBI ADDRESS INTO DATA PATHS (BI MAINT SEQUENCES)
Q = LATCH CASRAM DATA INTO DATA PATHS
R = CASRAM DATA TO INTERNAL BUS

RE198

Figure 3-15 DMB32 Slave Read Transaction



LEGEND

A = BI ADDRESS
B = BI COMMAND
C = VALID DATA
F = DATA TO CASRAM
G = WRITE TO INTERNAL ADDRESS ARRAY REGISTERS
D = DATA STATUS

P = LATCH VAXBI ADDRESS INTO DATA PATHS (BI MAINT SEQUENCES)
S = COMMAND ADDRESS CYCLE RECOGNISED
T = ENABLE DATA PATHS ONTO INTERNAL BUS
U = LATCH VAXBI DATA INTO DATA PATHS
V = THIS NODE SELECTED

RE199

Figure 3-16 DMB32 Slave Write Transaction

3.17.1.3 IDENT Transaction – IDENT transactions are treated by the DMB32 in a similar way to READ transactions, although the VAXBI transaction itself has an additional arbitration cycle. On detection of an IDENT command the BIIC will assert the BCI_SEL L line. Assuming that an interrupt is pending, the CGA will recognize the IDENT command and will assert the JAM0 L signal. This forces the ATGA to ignore the address from BCI_D<31:0>H and to output an address of '0' on the CASRAM address lines RA<10:0>H.

The BIIC on the DMB32 module is programmed to request an external vector, in response to an IDENT command. The CGA will supply the least significant word of the required vector, the most significant word being sourced from longword 0 of CASRAM. The DMB32 will stall the first vector cycle in response to a VAXBI IDENT command, returning an ACK response with a valid vector in the next VAXBI cycle. Figure 3-17 details the BCI timing during an IDENT transaction.

In the delay between requesting a processor interrupt and supplying the interrupt vector, it is possible that the interrupt may have been disabled after it was posted. If a single pending interrupt is cleared by the host, via the interrupt enable control bits, the DMB32 will return a vector with bits 7 and 6 of the interrupt vector clear. This is used as the BIIC error interrupt vector (see Section 2.2.3) and will result in an error logging routine being executed. The error logging routine must, therefore, check whether either of the error summary flags in VAXBICSR are set, and whether the force bit in EINTRCSR is set. If none of these bits are set, the error interrupt resulted from a sync, async, or printer interrupt that was cleared by the host before the IDENT cycle.

3.17.1.4 Maintenance Levels – Maintenance Levels are selected by bits 4 and 5 of the MAINT CSR as follows:

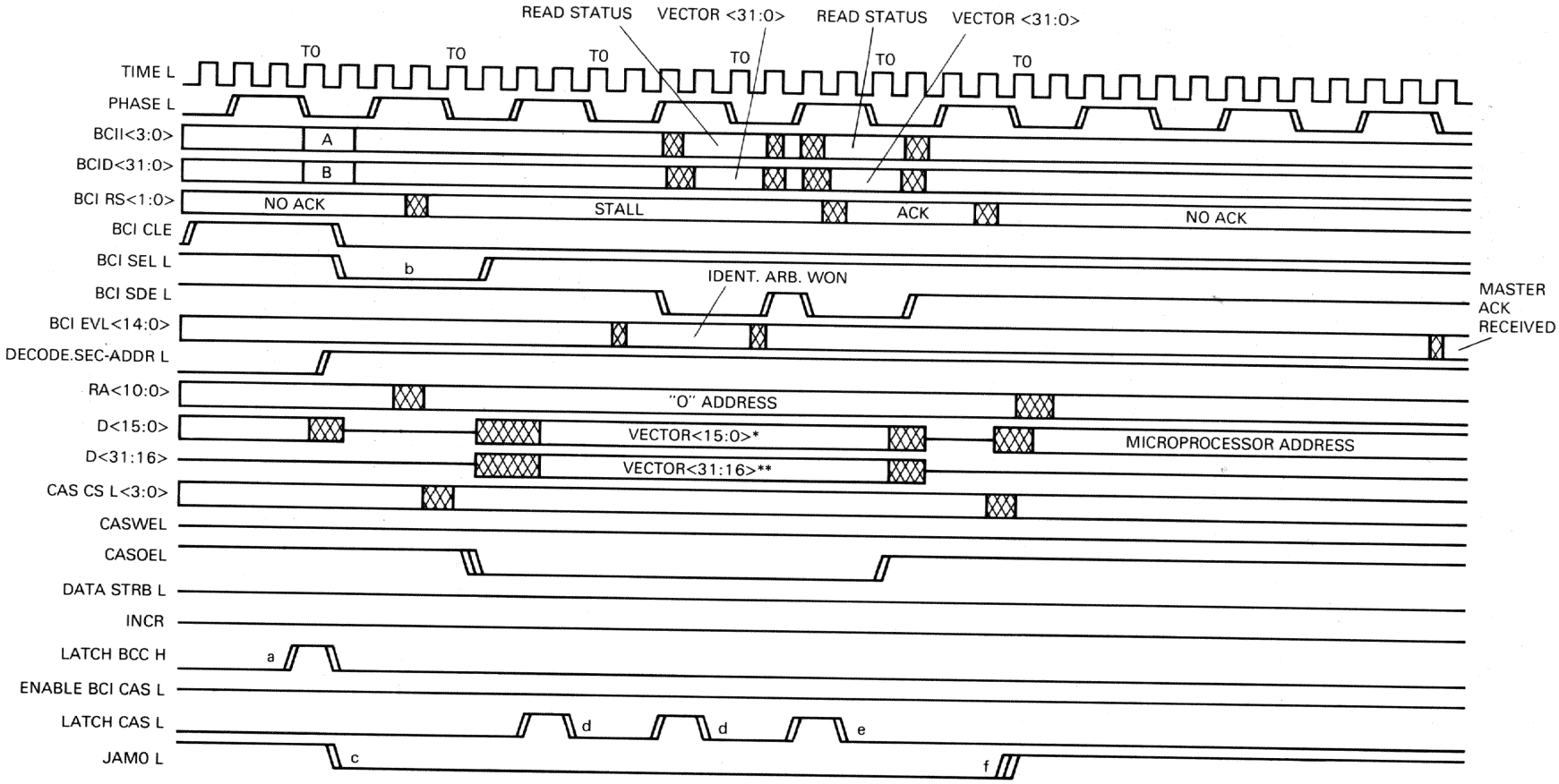
5	4	
0	0	Normal Operation
0	1	Maintenance Level 1
1	0	Maintenance Level 2
1	1	Maintenance Level 3

During normal operation, unimplemented CSRs, within DMB32 node space, and all of DMB32 VAXBI window space will receive a NO ACK, but in two of the three possible Maintenance Levels all 8 Kbytes of node space, and 256 Kbytes of window space will be accessible. Maintenance Level 1 transactions to window space cause the CGA to request control of the microprocessor bus for the duration of the transaction. This will result in extended, stalled transactions.

The contents of window space are defined by the mapping of devices on the microprocessor bus. This will include the CASRAM. Note, however, that although CASRAM will be selected at the microprocessor memory map addresses, the CASRAM itself is accessed by the addresses from the BCI rather than from the microprocessor. This results in CSRs being located at the VAXBI addresses, rather than the inverted microprocessor addresses within CASRAM. Gate array registers are not accessible via window space, thus permitting the host to perform pure RAM tests on CASRAM via window space while being able to access any of the gate array registers via node space.

NOTE

The microprocessor memory map on the DMB32 is 64 Kbytes deep, resulting in it being replicated 4 times within the 256 Kbytes of WINDOW space.



LEGEND

a = VAXBI ADDRESS LATCHED INTO DATA PATHS (MAINT. MODE ONLY)
b = THIS NODE SELECTED
c = INDENT COMMAND RECOGNISED
d = CGA ATTEMPTS TO LATCH DATA INTO DATA PATHS
e = CGA LATCHES VALID DATA INTO DATA PATHS
f = JAMO L DEASSERTED WITH CASCSL<3:0>
* = VECTOR<15:0> SOURCED FROM CGA "BI VECTOR" REGISTER
** = VECTOR<31:16> SOURCED FROM CASRAM LOCATION "0"
A = VAXBI COMMAND
B = VAXBI ADDRESS

RE1448

Figure 3-17 VAXBI IDENT Transaction

Due to differences in bus widths, the CASRAM area within the window space will be longword accessible, while the remainder of window space will only be word accessible.

The duration of VAXBI transfers to window space is controlled by the response time of these peripherals on the microprocessor bus and additionally, in Maintenance Level 1, by the bus grant response time of the microprocessor.

On gaining control of the microprocessor bus the CGA will drive all of the microprocessor control signals, except FC<2:0>, and will emulate a microprocessor access to the devices on this bus. Figure 3-18 shows a VAXBI write to microprocessor RAM and Figure 3-19 shows a VAXBI read from microprocessor RAM.

In Maintenance Level 2, transfers to DMB32 VAXBI window space are similar to those for Maintenance Level 1, with the exception that the CGA requests control of the microprocessor bus with the setting of Maintenance Level 2, and will maintain mastership until exit from this maintenance level.

In Maintenance Level 2, all accesses to the module are assumed to be directed to the microprocessor bus, and care must be taken when addressing node space, so that CASRAM is selected by the microprocessor address bus before accessing node space. This means that CASRAM must be accessed via window space before node space is freely accessible; if any non-CASRAM locations are then addressed via window space, a CASRAM address must be accessed via window space before re-accessing node space.

Failure to follow this procedure will result in invalid data being transferred on the VAXBI, for node space read transactions, and will result in the last accessed window space location being modified by node space write transactions.

Maintenance Level 3 is intended for board testing and should not be selected. In Maintenance Level 3, the CGA will request control of the microprocessor bus as in Maintenance Level 2, but will leave all microprocessor control lines disabled to permit the use of external test equipment to control the bus.

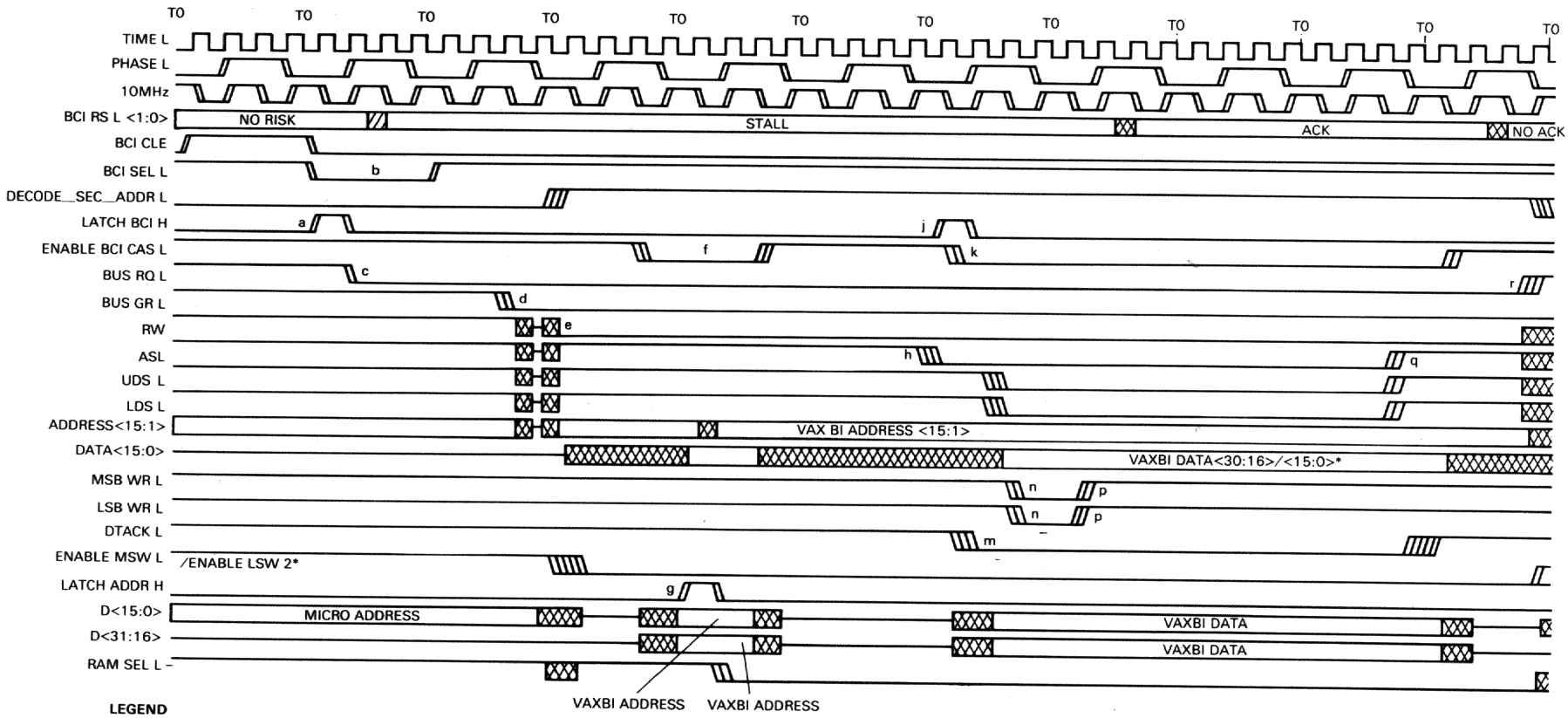
3.17.2 DMB32 Master Transactions

When it needs to perform a VAXBI transaction as master, the DMB32 microprocessor constructs a DMA file within CASRAM, writes to a file address register within the ATGA, updates a VAXBI mask register within the CGA (if a write mask command is to be requested) and finally, writes a command to the BI_COMM register in the CGA. This final write causes the BCI interface to initiate the transaction.

The CGA requests a transaction by asserting a request code on BCI_RQ<1:0>L. This may be for either a VAXBI transaction or a loopback transaction.

The CGA asserts the ATGA control signals RST_DMA_ADDR L and EN_DMA_ADDR L; this sets the DMA file address count field, RA<2:0> H, to 111 and enables the DMA file address onto RA<10:00> H. Thus the address on RA<10:00> H points to the location of the VAXBI address held in CASRAM. Once the address is latched, the DMA file address count field is incremented to 000 (to address the first data longword of the DMA file).

On receiving the request code, the BIIC arbitrates for the bus and asserts BCI_MDE L, to enable the VAXBI address from the data paths BCI_D<31:0> H, and the command from the CGA, BCI_I<3:0> L. This assertion of BCI_MDE L may occur as early as the cycle following the assertion of a request code, but may be delayed according to current BCI slave port activity. BCI_RAK L is now asserted to indicate that the BIIC has won control of the bus and is now bus master.

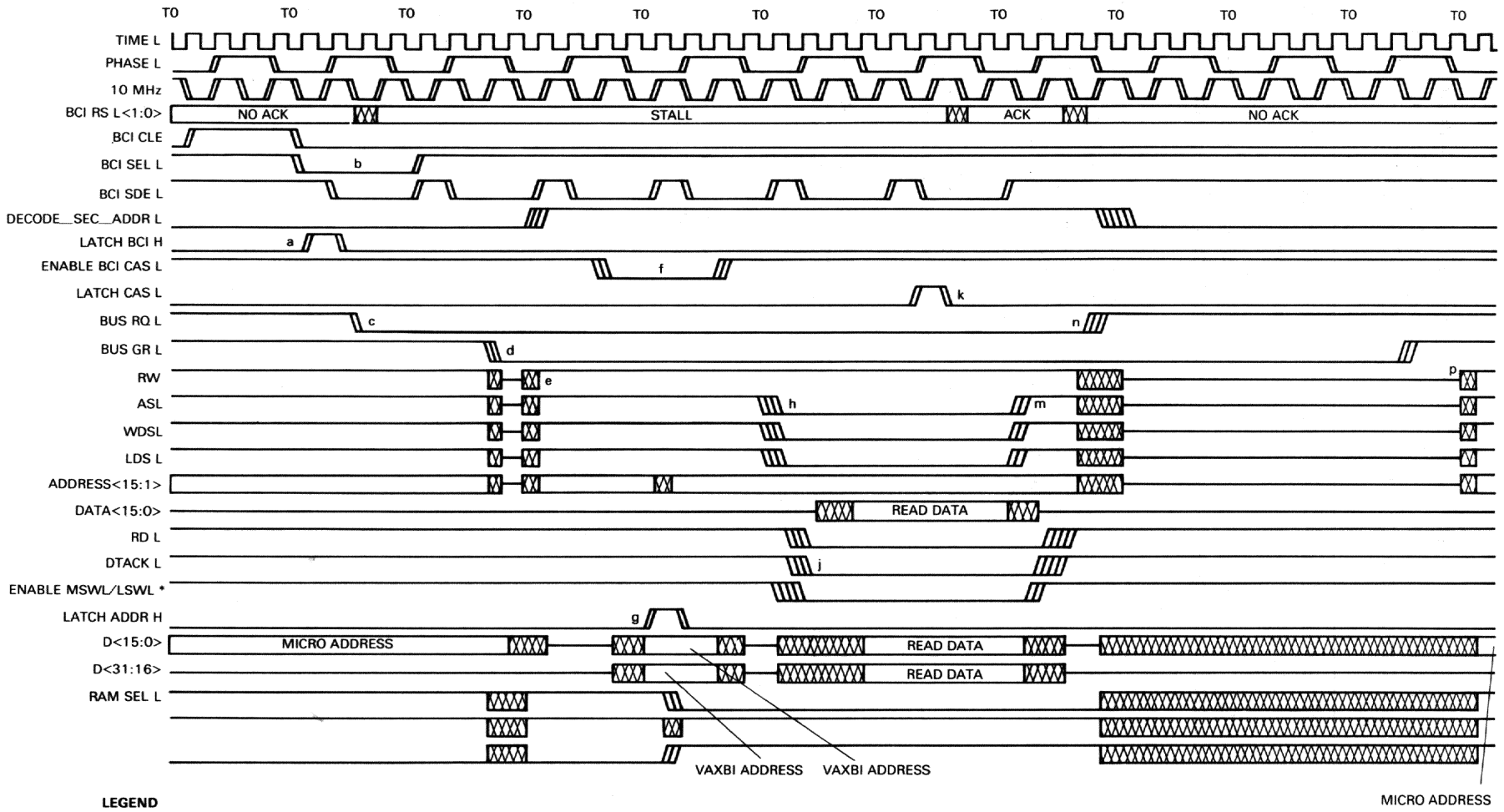


LEGEND

* ENABLE MSW L ASSERTED IF A, = 1)VAXBI CONVENTION
ENABLE LSW L ASSERTED IF A, = 0
a = VAXBI ADDRESS LATCHED INTO DATA PATHS
b = THIS NODE SELECTED
c = CGA REQUESTS CONTROL OF MICROPROCESSOR BUS
d = MICROPROCESSOR GRANTS REQUEST THEN RELEASES CONTROL, ADDRESS AND DATA LINES
e = CGA TAKES CONTROL OF MICROPROCESSOR BUS
f = CGA ENABLES VAXBI ADDRESS ONTO DATA BUS
g = CGA LATCHES VAXBI ADDRESS INTO BUS MUX (LS646s)
h = CGA STARTS MICROPROCESSOR BUS ACCESS ASSERTION OF ASL
j = CGA LATCHES VAXBI DATA LONGWORD INTO DATA PATHS
k = CGA ENABLES VAXBI DATA ONTO CASRAM DATA BUS AND ONE WORD ONTO MICROPROCESSOR BUS VIA BUS MUX (LS245s)
m = CGA ISSUES DTACK L WITH ZERO WAIT STATES (LOCAL RAM ACCESS) WHICH CONTROLS ACK RESPONSE TO BI
n = DATA WORD WRITTEN TO RAM (BYTE MASKABLE)
p = WRITE PULSES REMOVED AT END OF S6 AS PER MICROPROCESSOR ACCESS
q = ADDRESS AND DATA STROBES REMOVED, FIND MICROPROCESSOR CYCLES LATER
r = CGA RELEASES CONTROL OF MICROPROCESSOR BUS

RE1441

Figure 3-18 VAXBI Window Space Write to Microprocessor I/O Space (MAINT. MODE 1)



LEGEND

* = ENABLE MSW L AND ENABLE LSW L - BOTH ASSERTED FOR BAXBI READS
a = VAXBI ADDRESS LATCHED INTO DATA PATHS
b = THIS NODE SELECTED
c = CGA REQUESTS CONTROL OF MICROPROCESSOR BUS
d = MICROPROCESSOR GRANTS REQUEST AND RELEASES CONTROL, ADDRESS AND DATA LINES
e = CGA TAKES CONTROL OF MICROPROCESSOR BUS
f = CGA ENABLES VAXBI ADDRESS ONTO CASRAM DATA BUS
g = CGA LATCHES VAXBI ADDRESS INTO BUSMUX (LS646s)
h = CGA STARTS MICROPROCESSOR BUS READ CYCLE
j = CGA USES DTACK L WITH ZERO WAIT STATUS (LOCAL RAM ACCESS)
k = CGA LATCHES READ DATA FROM MICROPROCESSOR RAM INTO DATA PATHS
m = ADDRESS AND DATA STROBES REMOVED AT END OF ACCESS
n = CGA RELEASES CONTROL OF MICROPROCESSOR BUS
p = MICROPROCESSOR REGAINS CONTROL OF THE BUS

RE1442

Figure 3-19 VAXBI Window Space Read from Microprocessor RAM (MAINT. MODE 1)

The assertion of BCI_RAK L is always accompanied by the assertion of BCI_CLE H, indicating that a command/address cycle is present on the VAXBI bus. The slave port responds to the master port cycle by latching the command/address information into the CGA and ATGA. LATCH_BCI H latches the VAXBI address into the data path.

Any DMB32 master transaction can be aborted by BCI_MAB L; this signal is asserted by the CGA if any errors are detected during a master transaction or if a VAXBI retry timeout occurs.

3.17.2.1 Master Write Transaction – CAS_CS<3:0> L and CAS_OE L remain asserted throughout the transaction, to provide write data for the transaction. The data output to the CASRAM data bus will change as the CASRAM address changes.

When the BIIC is ready for data, it asserts BCI_NXT L. The CGA responds by latching the data with LATCH_CAS H, and incrementing the DMA file address with an assertion of INCR_DMA_ADDR H. The BIIC enables the data onto BCI_D<31:00> H , together with byte mask information from the CGA on BCI_I<3:0> H, by asserting BCI_MDE. The data is output on the VAXBI during the next VAXBI Cycle.

The sequence of INCR_DMA_ADDR H, BCI_NXT L, LATCH_CAS H and BCI_MDE L is repeated until all data cycles are complete, then the CASRAM is disabled and de-selected, and the DMA file address is removed from RA<10:00> H.

When the data is transferred and the BIIC receives two ACK cycles from the VAXBI slave; BCI_RAK L is de-asserted and the BIIC outputs a MASTER TRANSACTION COMPLETE code on BCI_EV<4:0> L. The CGA verifies that this code is present following the de-assertion of BCI_RAK L and updates an internal status register with a completion code.

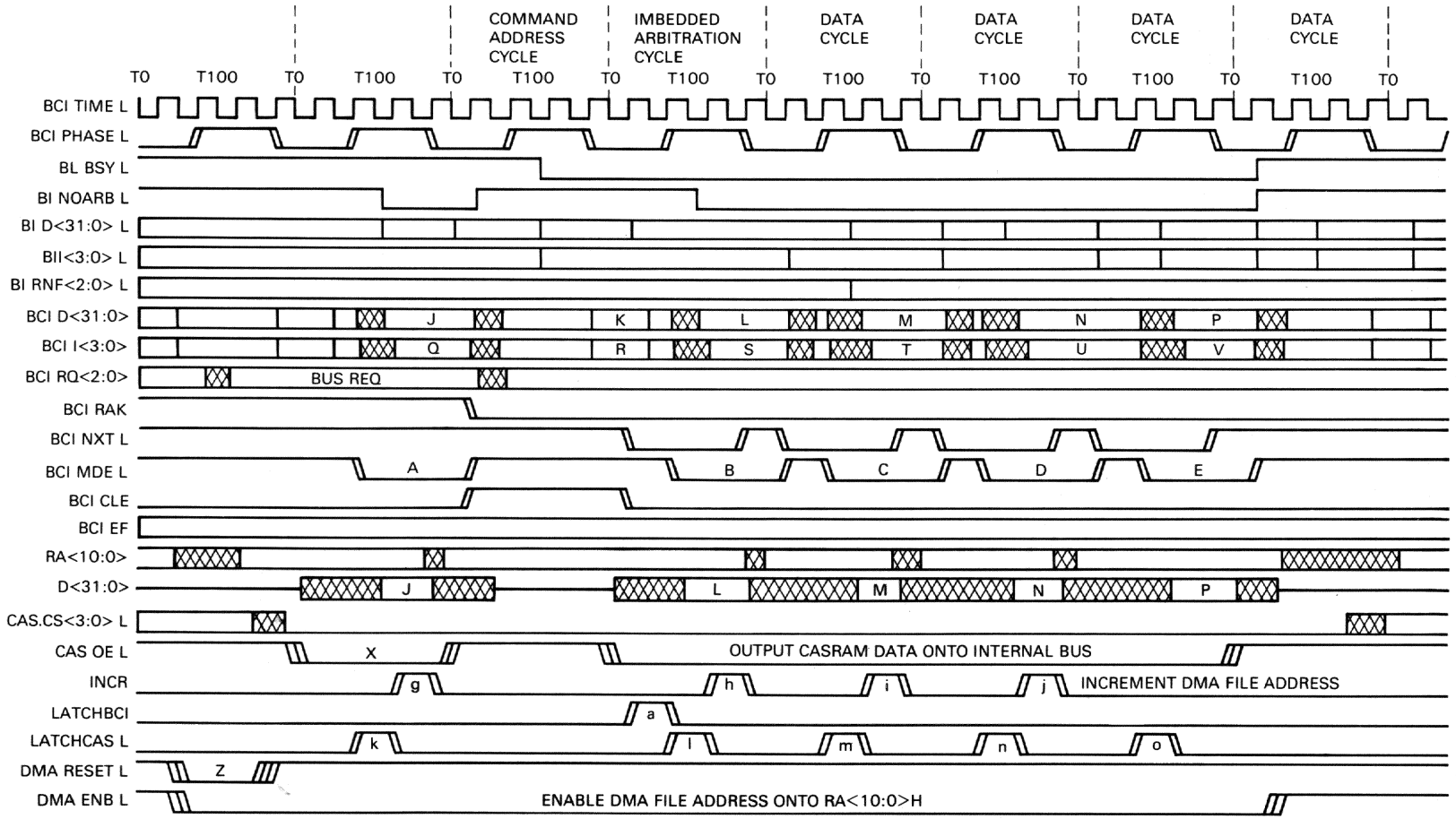
The example in Figure 3-20 is for an octaword write-mask transaction. A normal write transaction is identical, but BCI_I<3:0> L is ignored during data cycles. For a longword transaction only one data cycle occurs.

3.17.2.2 Master Read Transaction – In a Master Read Transaction the data direction is from the BCI, so BCI_MDE L is not asserted during data cycles; for a read transaction BCI_NXT L indicates that valid read data is on BCI_D<31:00>L. In response to assertions of BCI_NXT L, the data is written to CASRAM.

The CGA latches data into the data paths by asserting LATCH_BCI H, firstly during the imbedded arbitration cycle and then in every following data cycle. The data latched during the imbedded arbitration cycle is not used by the BCI interface. With the assertion of LATCH_BCI H, during the imbedded arbitration cycle, the CGA asserts ENABLE_CAS_BCI L to enable the contents of the data paths onto the internal bus.

In the following cycle, the assertion of BCI_NXT L indicates that the slave node acknowledged the first DATA cycle and returned data and data status on the BCI. This data is latched onto the internal data bus by LATCH_BCI H, and written to the bottom of the DMA file by CAS_WE L. With the de-assertion of CAS_WE L the CGA asserts INCR_DMA_ADDR H to increment the DMA File address.

The sequence of INCR_DMA_ADDR H, BCI_NXT L, LATCH_BCI H and CAS_WE L is repeated until all data cycles are complete. The data paths and DMA file address are then disabled, and CASRAM is de-selected.



LEGEND

A = BIIC TAKES COMMAND/ADDRESS
B = BIIC TAKES DATA 1
C = BIIC TAKES DATA 2
D = BIIC TAKES DATA 3
E = BIIC TAKES DATA 4
k = LATCH VAXBI ADDRESS INTO DATA PATHS
l = LATCH DATA 1
m = LATCH DATA 2
n = LATCH DATA 3
r = LATCH DATA 4

J = VAXBI ADDRESS (OUT)
K = VAXBI ADDRESS (IN)
L = DATA 1
M = DATA 2
N = DATA 3
P = DATA 4
Q = VAXBI COMMAND (OUT)
R = VAXBI COMMAND (IN)
S = MASK 1
T = MASK 2

U = MASK 3
V = MASK 4
X = OUTPUT VAXBI ADDRESS ONTO INTERNAL BUS
Z = INITIALIZE DMA FILE ADDRESS COUNT
a = LATCH BI COMMAND ADDRESS INTO DATA PATHS (SLAVE PORT)
g = INCREMENT DMA FILE ADDRESS

RE1443

Figure 3-20 DMB32 Master Write Transaction

After the last data cycle, the master BIIC sends two ACKs on the VAXBI to indicate to the slave that data has been correctly received. THE BIIC deasserts BCI_RAK L as the final ACK is being enabled onto the VAXBI and asserts the MASTER TRANSACTION COMPLETE code on BCI_EV<4:0>L.

The example in Figure 3-21 is for an octaword read transaction. For a longword transaction only one data cycle occurs.

Data is written to the DMA file regardless of the receive data status or received parity. If a parity error occurs the BIIC aborts the transfer and supplies an error code to the CGA, which sets an error code within an internal summary register. Receiving a READ DATA SUBSTITUTE status code for any longword does not abort the transaction, but causes a READ DATA SUBSTITUTE code to be output by the BIIC, on BCI_EV<4:0> L, with the de-assertion of BCI_RAK L at the end of the transaction.

The CGA sets an error code within an internal summary register which the microprocessor reads to determine if the master transaction was successfully completed.

3.17.3 Loopback Transactions

The DMB32 uses loopback transactions during power up self test, to initialize certain BIIC registers, to check for BIIC self-test completion and to verify correct CGA master and slave port control signals.

Loopback transactions are identical to Intranode transactions via the VAXBI except that the only VAXBI signals asserted by the BIIC are VAXBI BSY L and VAXBI NOARB L. Loopback transactions do not require bus arbitration and therefore permit rapid access to the BIIC internal registers. The onboard self test will not use Intranode transactions via the VAXBI if any failure is detected with VAXBI Loopback transactions.

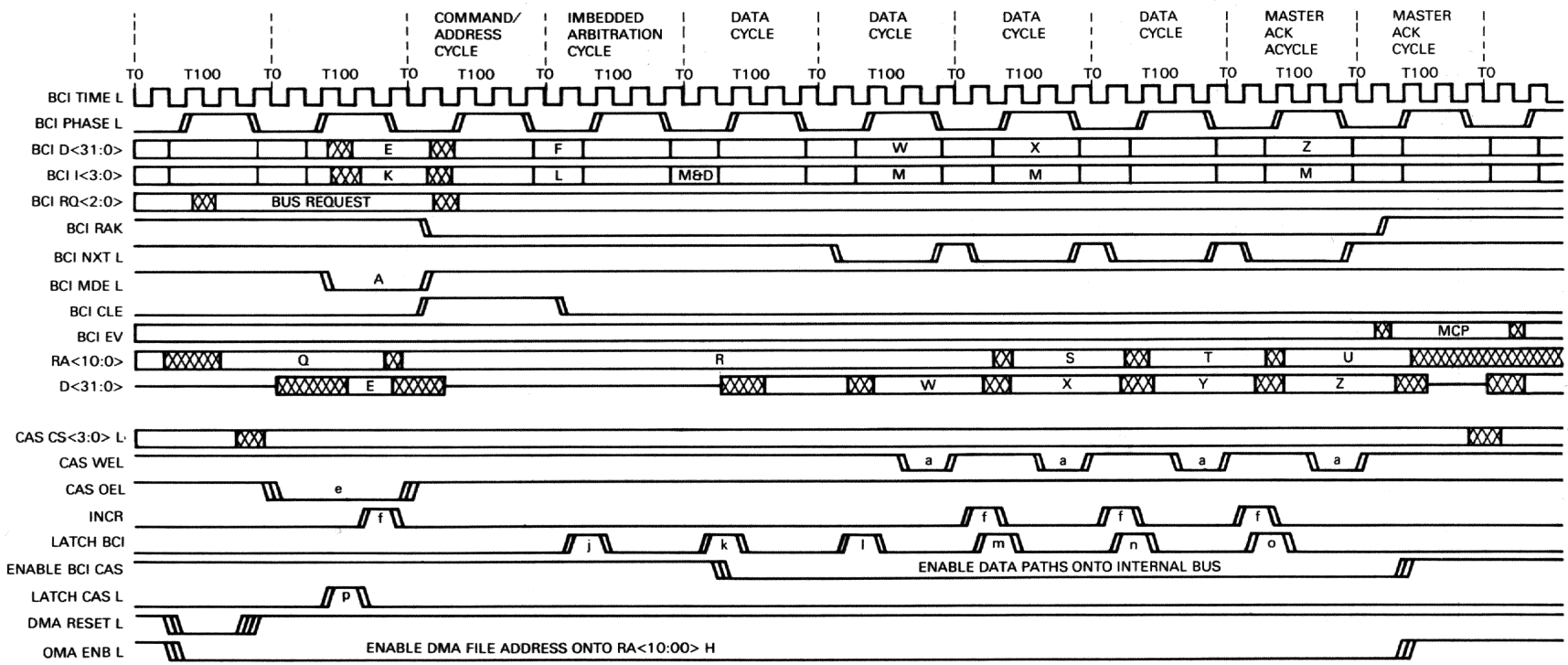
3.17.4 VAXBI INTR Transactions

VAXBI INTR commands are initiated by the DMB32 to request a host interrupt.

The DMB32 will always interrupt at level 4, the lowest VAXBI interrupt level. Interrupts are controlled via the CGA which monitors the interrupt enable flags within the sync, async, and printer CSRs. When the microprocessor sets a request bit within the BI_INTR register, and assuming that this interrupt is enabled, the CGA will synchronously assert a BCI INT L signal. The BIIC then requests control of the VAXBI and, upon becoming master,transmits a VAXBI INTR command.

Should this VAXBI INTR receive a NO ACK, or illegal confirmation from the slave node(s), the BIIC will inform the CGA via the BCI EV<4:0> L lines. This will cause the CGA to de-assert the BCI_INT L line and set the FIR (FAILED INTERRUPT EV CODE RECEIVED) within an internal status register. The CGA will re-assert the BCI INT L signal upon the microprocessor reading this register, permitting the event to be logged and the interrupt to be retransmitted.

A successful VAXBI INTR command transmission causes the host node to log the interrupt and respond, once the host processor priority level is lower than the pending interrupt, with a VAXBI IDENT command.



LEGEND

A = BIIC TAKES COMMAND/ADDRESS
K = BI COMMAND (OUT)
L = BI COMMAND (IN)
E = BI ADDRESS (OUT)
F = BI ADDRESS (IN)
Q = CASRAM ADDRESS OF VAXBI ADDRESS
M = DATA STATUS

R = CASRAM ADDRESS TO WRITE DATA 1
S = DATA 2 ADDRESS
T = DATA 3 ADDRESS
U = DATA 4 ADDRESS
W = DATA 1
X = DATA 2
Y = DATA 3
Z = DATA 4

a = WRITE DATA 1
e = OUTPUT VAXBI ADDRESS ONTO INTERNAL BUS
f = INCREMENT DMA FILE ADDRESS
g = 001
h = 010
i = 011
j = LATCH VAXBI ADDRESS INTO DATA PATHS
k = LATCH MASTER ID FROM BCII<3:0> (UNUSED)

l = LATCH DATA 1 INTO DATA PATHS
m = LATCH DATA 2 INTO DATA PATHS
n = LATCH DATA 3 INTO DATA PATHS
o = LATCH DATA 4 INTO DATA PATHS
p = LATCH VAXBI ADDRESS INTO DATA PATHS
MID = MASTER ID
MCP = MASTER COMPLETION EVENT CODE

RE200

Figure 3-21 DMB32 Master Read Transaction Timing

3.18 OVERALL CONTROL LOGIC OPERATION AND TIMING

3.18.1 Microprocessor Write to CASRAM

Figure 3-22 shows the waveforms for a Write To CASRAM by the microprocessor. Clock states S0 to S7 identify the microprocessor Write cycle. The sequence is described in following paragraphs.

At the end of S1, the microprocessor outputs the required CASRAM address on ADD<15:01> H. As the default state of the Busmux address drivers/latches is transparent in the direction microprocessor bus to internal bus, the address will appear on the internal bus.

A direct version of ADD<15:12> (not via the Busmux) is continually decoded by logic in the CGA. When a CASRAM address is decoded, the CGA asserts CAS_RQ L to prepare the ATGA secondary access port for the assertion of AS L. Both must be asserted to enable the port. They must remain asserted for the duration of a secondary access port transaction.

At the end of S2, AS L is asserted to enable the secondary access port. Thus the CASRAM address, translated as necessary, appears at RA<10:00> H. The address is latched in the ATGA by the de-assertion of LATCH_SEC_ADDR H, and is inhibited at the Busmux by the de-assertion of ENABLE_ADDRESS L.

After asserting AS L, the microprocessor outputs the data on DAT<15:00> H, asserts UDS L and LDS L to identify a word transaction, and de-asserts RW H to provide a Write Strobe. LSB_WR L and MSB_WR L, which are derived from RW H, UDS L and LDS L are generated, but these are not used for CASRAM accesses.

In Figure 3-22 the waveforms are for a least significant word write transaction. When ENABLE_LSW L is asserted, the data DAT<15:00> H is enabled onto the internal bus D<15:00> H.

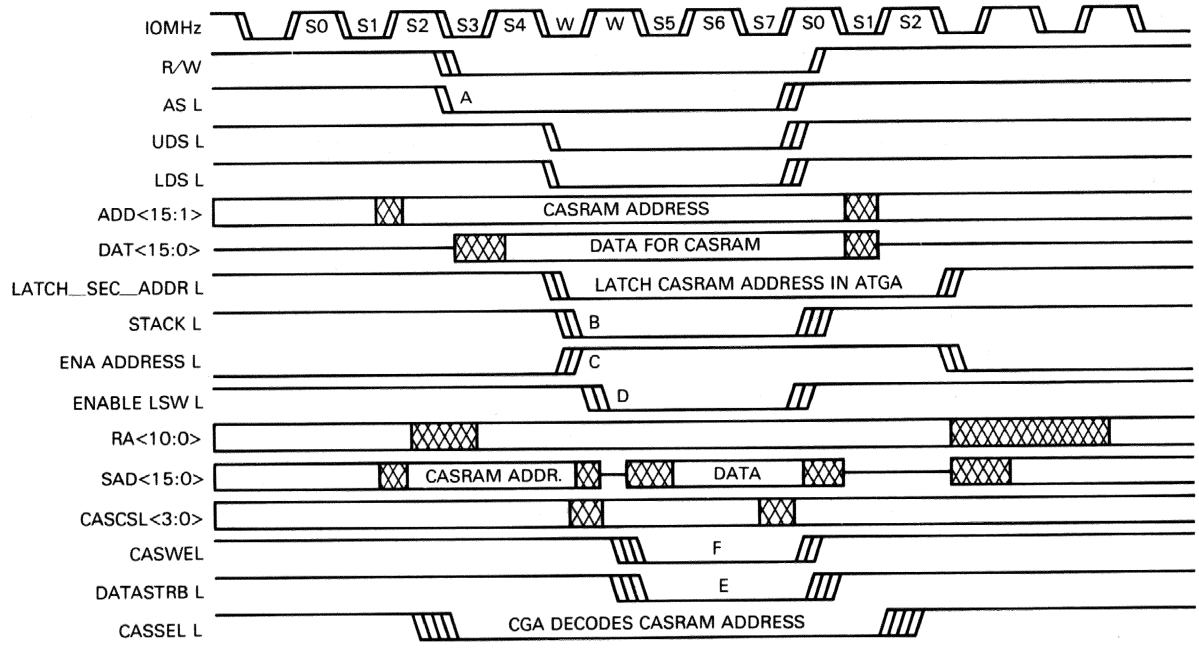
CAS_CS<1:0> L selects the low word of CASRAM, and the RAM is enabled by CAS_WE L; VALID_SEC_DATA L is also asserted. This signal enables the gate array data ports in case the data is also to be written to a gate array register. The Write action is completed by the de-assertion of CAS_WE L, and in the case of gate array data, VALID_SEC_DATA L. The CASRAM is de-selected by the de-assertion of CAS_CS<1:0> L.

In all microprocessor transactions, wait states are inserted according to values loaded in the MICRO_DELAY register in the CGA. These predetermined values are loaded at initialization time; they cannot be modified by the host.

The extension of the transaction is achieved by delaying the assertion of DTACK L. Unless this signal is asserted, the microprocessor will insert wait states. DTACK L is sampled during S4, and during every subsequent wait state. When DTACK L is detected, the transaction cycle will proceed.

At the end of the microprocessor transaction, ENABLE_LSW L, AS L, UDS L, LDS L and DTACK L are de-asserted, and the microprocessor address and data lines are tri-stated. Two clock cycles later, ENABLE_ADDRESS L is asserted to re-enable the Busmux address latches ready for the next microprocessor transaction.

LATCH_SEC_ADDR H is asserted, ready for the next secondary port access.



LEGEND

A = SELECT SECONDARY PORT
B = PERMIT MICROPROCESSOR TRANSACTION TO PROCEED
C = DISABLE LS646s
D = ENABLE DATA TO INTERNAL BUS
E = WRITE PULSE FOR ATGA
F = WRITE STROBE

RE201

Figure 3-22 Busmux Write Transaction Timing

3.18.2 Microprocessor Read from CASRAM

The example of Figure 3-23 illustrates how the command/address cycle of a primary port access (DMB32 slave) can occur during accessing by the secondary access port. The command/address information is latched into the primary access port by BCI_CLE H, without disrupting accessing by the secondary access port. The secondary access port transaction will be complete before the primary access port needs access to the internal bus.

The Microprocessor Read transaction starts with the CASRAM address on ADD<15:01> H. Then AS L, UDS L, LDS L and RW H are all asserted. Thus the Read strobe RD L is generated.

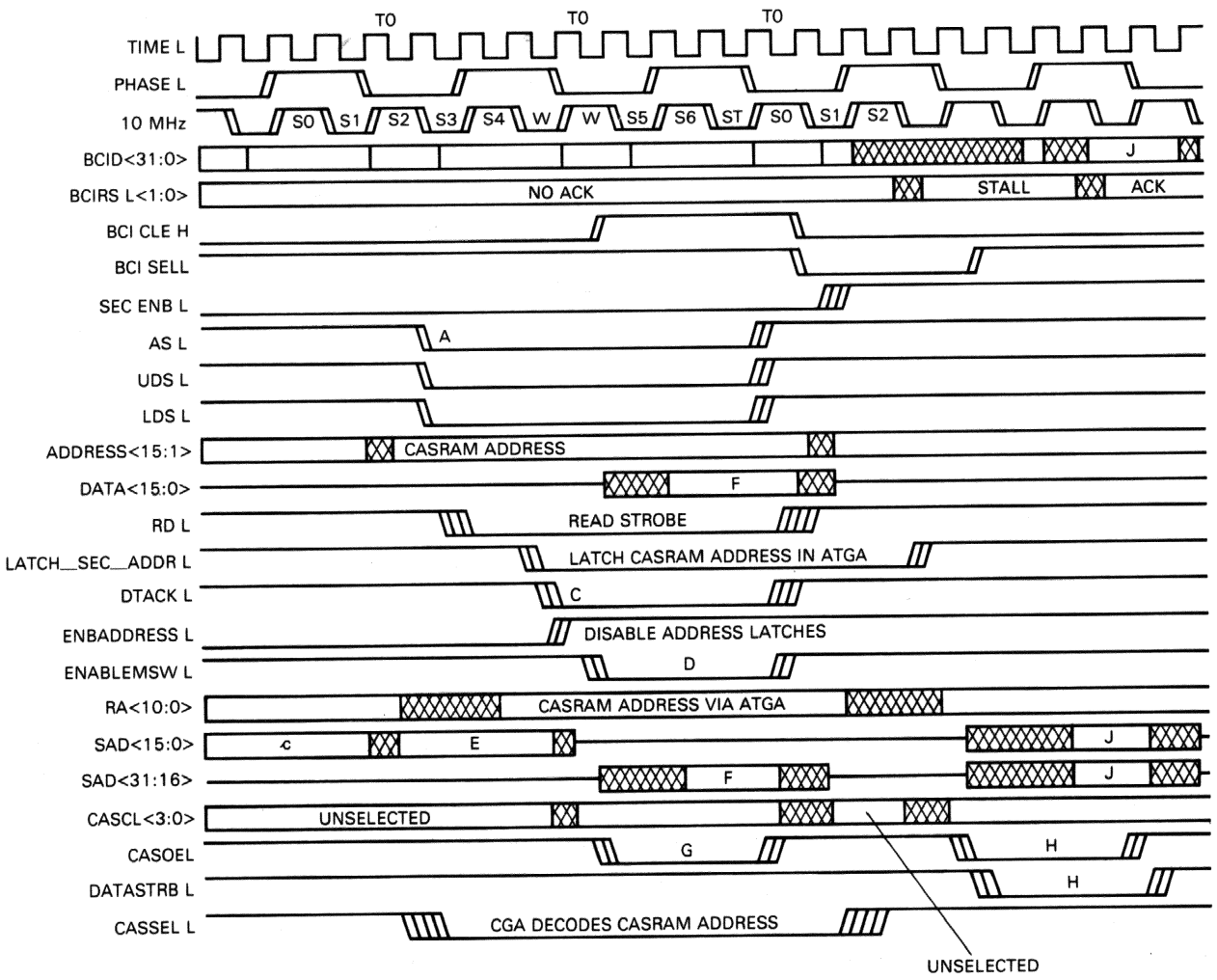
The CASRAM address is enabled into the ATGA by AS L, and latched by the de-assertion of LATCH_SEC_ADDR H.

In the Busmux, the address latches are disabled and the data latches enabled by the de-assertion of ENABLE_ADDRESS L and the assertion of ENABLE_MSW L. Note that in this case the most significant word is addressed by CASRAM address bit RA<1> H.

CAS_CS<3:2> L selects the most significant word and CAS_OE L enables it to the microprocessor via the internal bus and the Busmux. The data is latched internally before the microprocessor de-asserts AS L, UDS L and LDS L, and tri-states the address lines.

CAS_CS<3:2> L and CAS_OE L are de-asserted to disable the CASRAM.

Two clock cycles later LATCH_SEC_ADDR H is asserted. However, the latches remain disabled by ENABLE_ADDRESS L which remains de-asserted during the primary access port activity that is now in progress.



LEGEND

A = SELECT SECONDARY PORT
J = DATA OUT (FOR BI)
C = PERMIT MICROPROCESSOR TRANSACTION TO PROCEED
D = ENABLE DATA TO MICROPROCESSOR
E = CASRAM ADDRESS
F = DATA OUT (FOR MICROPROCESSOR)
G = CASRAM DATA TO INTERNAL BUS (FOR MICROPROCESSOR)
H = CASRAM DATA TO INTERNAL BUS (FOR BI)

-c = PREVIOUS CASRAM ADDRESS

RE202

Figure 3-23 Busmux Read Transaction Timing

APPENDIX A
IC DESCRIPTIONS

A.1 SCOPE

This appendix contains data on the following major ICs used on the DMB32:

- 68000 Microprocessor (Section A.2)
- 8530 Universal Synchronous/Asynchronous Receiver/Transmitter (USART) (Section A.3)
- 8536 Counter/Timer and Parallel I/O Unit (Section A.4)
- 2681 Dual UART (DUART) (Section A.5)

More detailed information on the ICs is given in the manufacturer's data sheets. The smaller, more common ICs are well described in semiconductor data books and are not included here.

A.2 68000 MICROPROCESSOR

A.2.1 Overview

The 68000 is a 16-bit microprocessor which has 32-bit internal architecture. Its main features are:

- 16-bit asynchronous data bus
- 23-bit asynchronous address bus, capable of addressing 16 Mbyte in conjunction with data strobes (UDS and LDS).
- Eight 32-bit data registers
- Seven 32-bit address registers
- Memory-mapped I/O
- Compatibility with 6800-series peripheral ICs
- Single +5 V power supply
- Mounted in a 64-pin DIL package.

The architecture of the 68000 is shown in simplified form in Figure A-1.

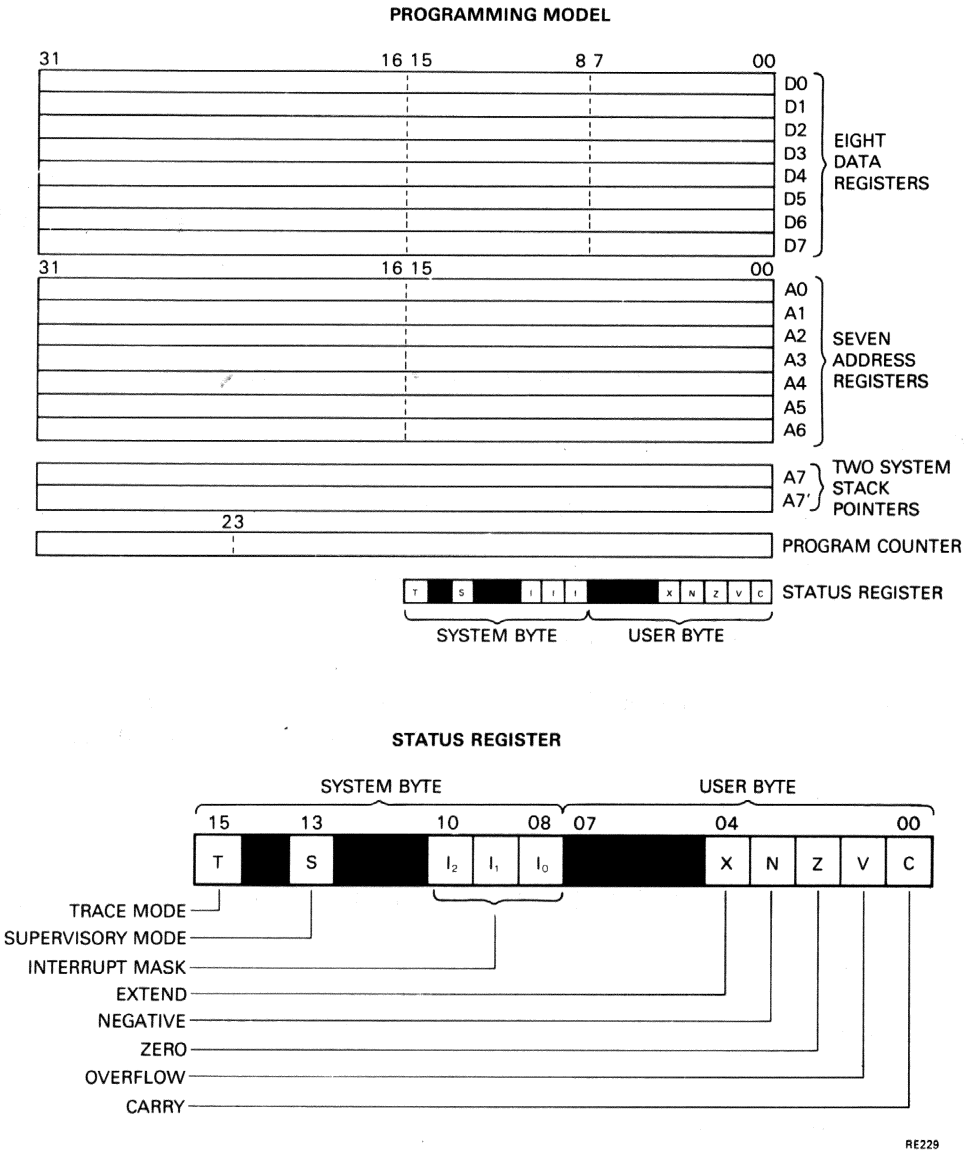


Figure A-1 68000 Architecture

A.2.2 Signals and Pinout

The signals to and from the 68000 microprocessor can be considered as being divided into groups. These groups are shown in Figure A-2. The functions of these groups and their signals are described in Table A-1, the power supply and ground connections are included.

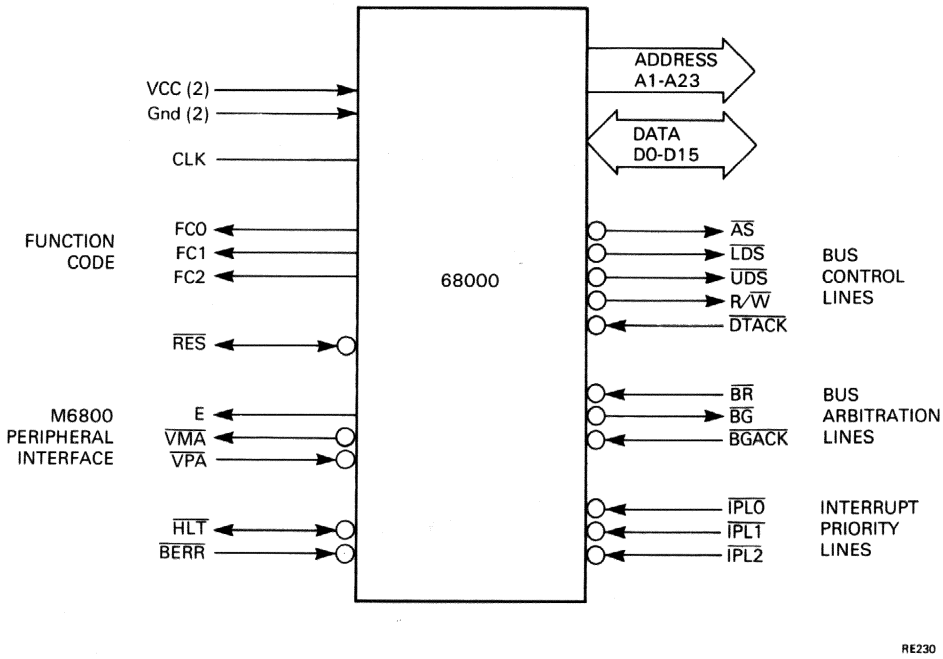


Figure A-2 68000 Input/Output Signals

The physical connections that correspond to the signals, and the power supply and ground connections, are identified in the pinout diagram Figure A-3.

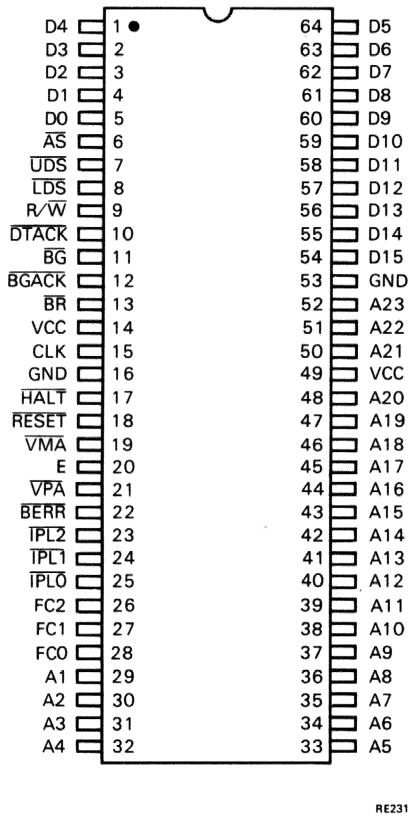


Figure A-3 68000 Pinout

Table A-1 68000 Signal Descriptions

Address and Data Bus	
Address Bus Lines (A1 to A23)	23-bit output bus to address 16 megabytes, in conjunction with UDS and LDS. Lines A1, A2 and A3 are also used to signal the interrupt level while an interrupt is being serviced.
Data Bus Lines (D0 to D15)	16-bit bidirectional bus to transfer data in words or bytes. Lines D0 to D7 are also used to receive a vector number during an interrupt-acknowledge cycle.
Bus Control	
Address Strobe (AS)	An output indicating that a valid address is on the address bus.
Data Strobes (LDS, UDS)	Outputs indicating whether data transfer is on the upper, the lower or both bytes of the data bus.
Read/Write (R/W)	An output indicating whether a data bus transfer is Read or Write, and also controlling external bus buffers.
Data Transfer Acknowledge (DTACK)	An input which extends the data bus cycle time until it is asserted, so allowing the data bus to synchronize with slow devices or memories.
Bus Arbitration	
Bus Request (BR)	An input from a device asking for control of the bus.
Bus Grant (BG)	An output from the 68000 granting control of the bus.
Bus Grant Acknowledge (BGACK)	An input from a device confirming that it has control of the bus.
Interrupt Priority	
Interrupt Priority Lines (IPL0, IPL1, IPL2)	Inputs which give the priority level of an interrupting device or process. The priority level is in the range 0 to 7; 0 is 'no interrupt' and 7 is the highest priority. IPL2 is the MSB.
Function Code	
Function Code Lines (FC0, FC1, FC2)	Outputs which indicate to external devices the status (User or Supervisor) and the type of cycle being executed.
M6800 Peripheral Interface	
Valid Peripheral Address (VPA)	An input that indicates to the 68000 that the device or memory region addressed is an M6800 type and that data transfer should be synchronized to the Enable signal (E).
Valid Memory Address (VMA)	An output in response to VPA which indicates that a valid address is on the address bus and that the 68000 is synchronized to the Enable Signal.
Enable (E)	An output which is the standard enable clock signal for M6800 systems.

Table A-1 68000 Signal Descriptions (continued)

System Control and Timing	
Bus Error (BERR)	An input from an external device that terminates the current bus cycle in the event of a problem. Also interacts with the Halt signal (HLT).
Reset (RES)	A bidirectional signal line that either receives an external reset signal or outputs a reset signal to external devices, causing either the 68000 or the external devices to perform an initialization sequence. Also interacts with the Halt signal (HLT).
Halt (HLT)	A bidirectional signal line that either receives an external halt signal or outputs a signal indicating to external devices that the 68000 has stopped. An external halt signal causes the 68000 to stop at the end of the current bus cycle, a halted 68000 can only be re-started by an external Reset. Also interacts with the Bus Error and Reset signals.
Clock (CLK)	The input to the 68000 from the master system clock, the frequency is 5 MHz.
Power Supply	
+5 volts (Vcc)	The single power supply input, connected to two pins.
Ground (GND)	The zero-voltage side of the power supply, connected to two pins.

A.3 8530 UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS RECEIVER/TRANSMITTER

A.3.1 Overview

The 8530 Universal Synchronous/Asynchronous Receiver/Transmitter (USART) is a peripheral IC for data communications. It can be software configured to handle several types of encoding and protocol. Its main features are:

- Two channels
- Programmable baud rates
- NRZ, NRZI and FM encoding
- HDLC and SDLC bit-oriented synchronous protocols
- Monosync and Bisync byte-oriented synchronous protocols
- CRC generation and checking
- Flag and zero insertion and checking
- 6-bit to 8-bit character lengths and residue handling
- Mounted in a 40-pin DIL package.

The architecture of the 8530 USART is shown in Figure A-4, and its register set is summarized in Figure A-5.

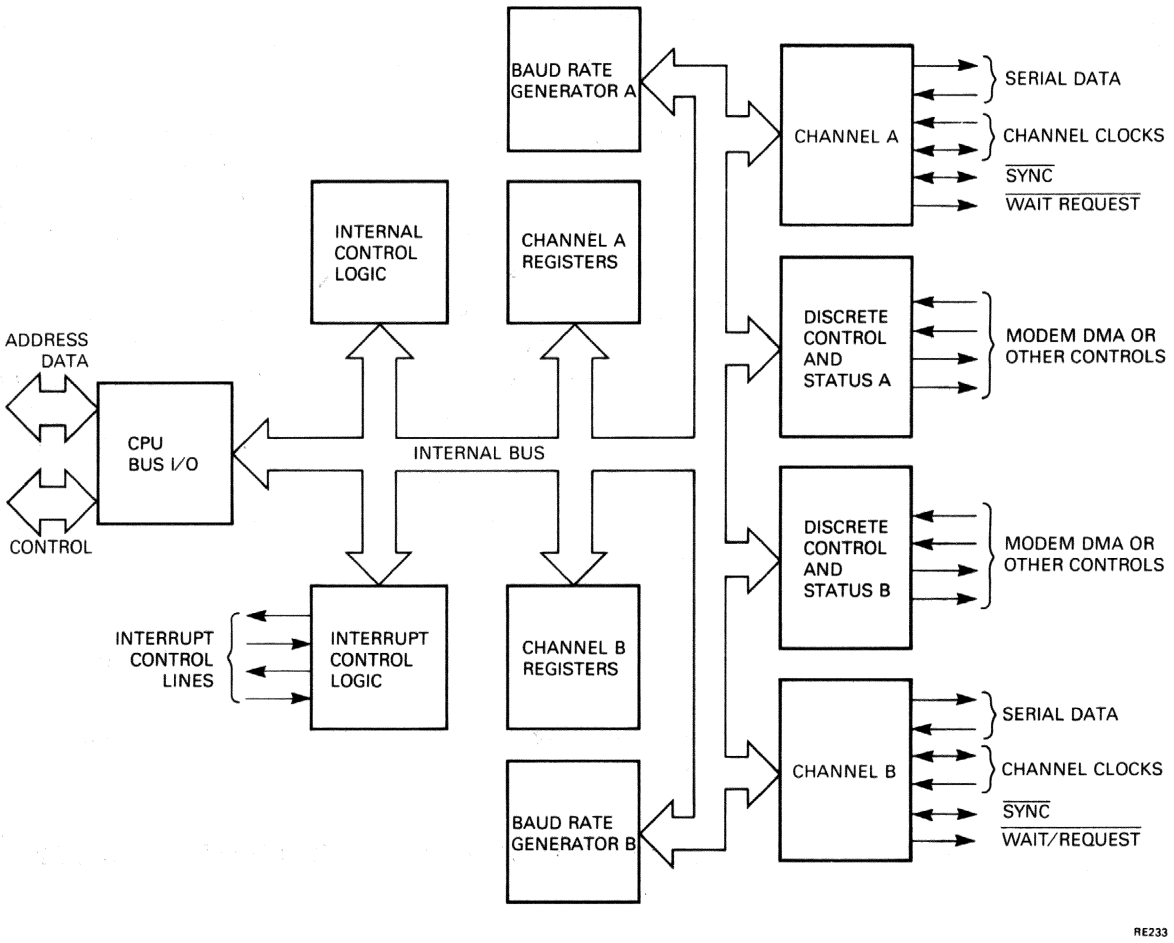


Figure A-4 8530 Architecture

READ REGISTER FUNCTIONS

RR0	TRANSMIT/RECEIVE BUFFER STATUS AND EXTERNAL STATUS
RR1	SPECIAL RECEIVE CONDITION STATUS
RR2	MODIFIED INTERRUPT VECTOR (CHANNEL B ONLY) UNMODIFIED INTERRUPT VECTOR (CHANNEL A ONLY)
RR3	INTERRUPT PENDING BITS (CHANNEL A ONLY)
RR8	RECEIVE BUFFER
RR10	MISCELLANEOUS STATUS
RR12	LOWER BYTE OF BAUD RATE GENERATOR TIME CONSTANT
RR13	UPPER BYTE OF BAUD RATE GENERATOR TIME CONSTANT
RR15	EXTERNAL/STATUS INTERRUPT INFORMATION

WRITE REGISTER FUNCTIONS

WR0	CRC INITIALIZE, INITIALIZATION COMMANDS FOR THE VARIOUS MODES, SHIFT RIGHT/SHIFT LEFT COMMAND
WR1	TRANSMIT/RECEIVE INTERRUPT AND DATA TRANSFER MODE DEFINITION
WR2	INTERRUPT VECTOR (ACCESSED THROUGH EITHER CHANNEL)
WR3	RECEIVE PARAMETERS AND CONTROL
WR4	TRANSMIT/RECEIVE MISCELLANEOUS PARAMETERS AND MODES
WR5	TRANSMIT PARAMETERS AND CONTROLS
WR6	SYNC CHARACTERS OR SDLC ADDRESS FIELD
WR7	SYNC CHARACTER OR SDLC FLAG
WR8	TRANSMIT BUFFER
WR9	MASTER INTERRUPT CONTROL AND RESET (ACCESSED THROUGH EITHER CHANNEL)
WR10	MISCELLANEOUS TRANSMITTER/RECEIVER CONTROL BITS
WR11	CLOCK MODE CONTROL
WR12	LOWER BYTE OF BAUD RATE GENERATOR TIME CONSTANT
WR13	UPPER BYTE OF BAUD RATE GENERATOR TIME CONSTANT
WR14	MISCELLANEOUS CONTROL BITS
WR15	EXTERNAL/STATUS INTERRUPT CONTROL

RE234

Figure A-5 8530 Register Summary

A.3.2 Signals and Pinout

The function of the signals to and from the 8530 USART are described in Table A-2, the power supply and ground connections are included for completeness. The physical connections that correspond to the signals, and the power supply and ground connections, are identified in the pinout diagram Figure A-6.

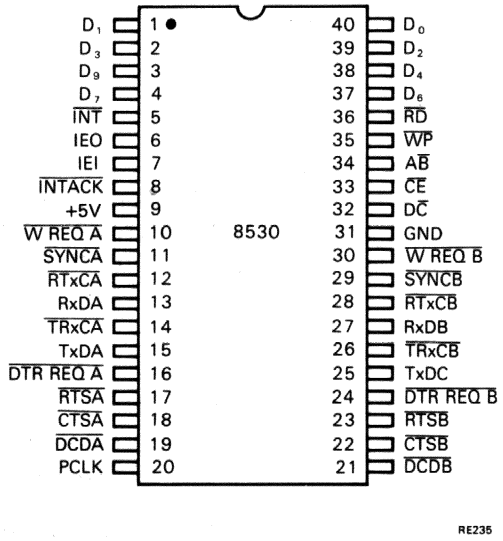


Figure A-6 8530 Pinout

Table A-2 8530 Signal Descriptions

Data Bus	
Data Bus Lines (D0 to D7)	8-bit bidirectional bus to transfer data in bytes.
Bus Timing and Reset	
Read (RD)	An input indicating that data is to be transferred to the 8530 via one of the serial channels, and enabling the 8530 bus drivers. Also used to transfer an interrupt vector to the data bus.
Write (WR)	An input indicating that data is to be transferred from the 8530, via one of the serial channels. If RD and WR are asserted together the 8530 will perform a Reset operation. (Note that both RD and WR are dependent on the CE signal).
Control	
Channel Select (A/B)	An input which selects whether Channel A or Channel B is to be used for a Read or Write operation.
Chip Enable (CE)	An input which enables the 8530 for a Read or Write operation.
Data/Control Select (D/C)	An input which defines the type of information to be transferred to or from the 8530. High assertion indicates a data transfer, low assertion indicates a command transfer.

Table A-2 8530 Signal Descriptions (continued)

Interrupt	
Interrupt Request (INT)	An output indicating that the 8530 needs to interrupt the 68000.
Interrupt Acknowledge	An input indicating that the 68000 is processing the 8530 interrupt. When the interrupt daisy-chain stabilizes, RD is asserted and the 8530 outputs the interrupt vector on the data bus.
Interrupt Enable In (IEI)	An input indicating that the 8530 can interrupt the 68000 microprocessor.
Interrupt Enable Out (IEO)	Not connected (normally used to output the Interrupt Enable signal to a lower priority device).
Serial Data (Channel A and Channel B)	
Transmit Data Line (TxDA, TxDB)	An output signal to transmit serial data at standard TTL levels.
Receive Data Line (RxDA, RxDB)	An input signal to receive serial data at standard TTL levels.
Channel Control (Channel A and Channel B)	
Synchronization (SYNCA, SYNCB)	Not connected (normally used to synchronize Read and Write operations).
Wait/Request (W/REQA, W/REQB)	A dual-function output that is programmable either as a Request line for DMA control or as a Wait line to synchronize the data rate of the 8530 to that of the 68000.
Data Terminal Ready/Request (DTR/REQA,DTR/REQB)	A dual-function output that is programmable (by the DTR bit) either as a general-purpose output or as a Request line for DMA.
Request to Send (RTSA, RTSB)	An output indicating that there is data ready to transmit on the serial line.
Clear To Send (CTSA, CTSB)	An input in response to the RTS signal indicating that data can be transmitted.
Channel Clocks (Channel A and Channel B)	
Transmit/Receive Clock (TRxCA, TRxCB)	Programmable in several modes.
Receive/Transmit Clock (RTxCA, RTxCB)	Programmable in several modes

Table A-2 8530 Signal Descriptions (continued)

System Clock	
Clock (PCLK)	An input to receive the master system clock 5 MHz signal.
Power Supply	
+5 volts (Vcc)	The power supply input.
Ground (GND)	The zero-voltage side of the power supply.

A.4 8536 COUNTER/TIMER

A.4.1 Overview

The 8536 Counter/Timer and parallel I/O (CIO) is a general-purpose peripheral support IC which has three I/O ports (PA, PB and PC) and three counter/timer circuits (CT1, CT2 and CT3). These are shown in the basic block diagram of Figure A-7. The ports and the counter/timer are programmed via the CPU interface.

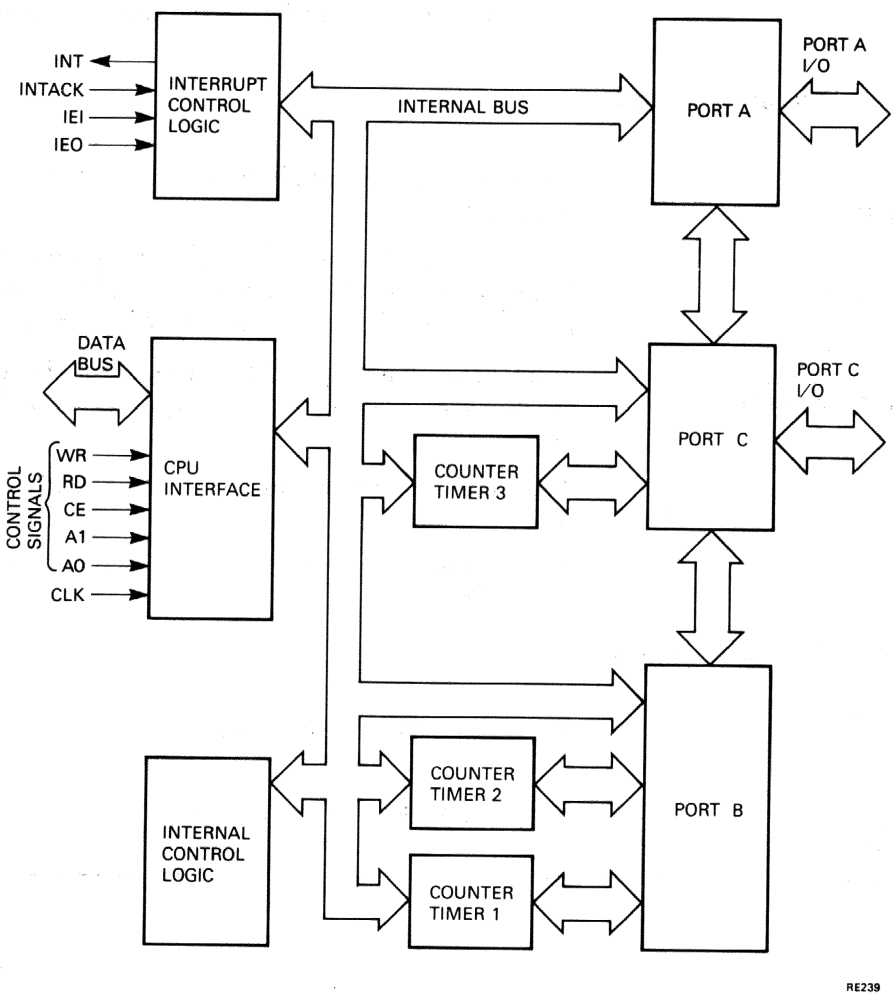


Figure A-7 CIO Architecture

There are five potential sources of interrupt for events in the CIO. These have the priority CT3, PA, CT2, PB, and CT1.

A.4.1.1 The CPU Interface – The CPU is accessed and programmed via control, status, and data registers in internal control logic, via the registers, data can be transferred, ports can be configured, counter/timers can be programmed and enabled, and status can be monitored.

Data registers for ports A, B, and C can be directly addressed and accessed by means of five control signals; CE, WR, RD, and A<1:0>. Status and control registers are accessed in a two-stage process via an address pointer register. Data to or from the addressed register is moved on the data bus.

A 5 MHz signal (CLK) provides a clock signal for the CIO logic. CLK is also divided by two to provide a count signal for the timer functions of counter/timers 1, 2 and 3.

A.4.1.2 I/O Ports – A and B are eight-bit I/O ports which can be programmed, as bytes or as individual bits. They can be input, output or bidirectional. Port C can be programmed as individual input, output or bidirectional bits. It can also provide handshake signals to support data transfer on ports A or B.

Ports B and C also have a counter/timer function. When so programmed, B<3:0>, B<7:4> and C<3:0> can be used as external connections to counter/timers 1, 2 and 3 respectively.

To ease the servicing needs of ports A and B when they are used for transferring data, these ports may be single or double buffered.

A.4.1.3 Counter/Timers – Via the CPU interface, counter/timer registers are loaded with a count value. The register is then counted down by a clock signal (timer function) or an event signal (counter function). If needed, the count value can be reloaded without further access by the CPU.

If it is programmed to use the associated I/O port, a counter/timer can be controlled and monitored externally. If the I/O port is not used, timeout must be reported by interrupt, or the CIO must be polled. Table A-3 shows the function of external access pins.

Table A-3 Counter/Timer External Access

Function	C/T1	C/T2	C/T3
Counter/Timer Output	PB 4	PB 0	PC 0
Counter Input	PB 5	PB 1	PC 1
Trigger Input	PB 6	PB 2	PC 2
Gate Input	PB 7	PB 3	PC 3

A.4.1.4 Interrupts – Interrupts are raised by the signal INT (see Figure A-7) and acknowledged by INTACK. However, for the CIO to raise an interrupt, the interrupt must be enabled by both software and hardware. The software enable is implemented by writing a bit to a CIO register, and interrupt enable in (IEI) and out (IEO) provide daisy-chaining connection for a hardware interrupt priority scheme. If IEI is positive, the IC can raise an interrupt. If the IC is not involved in an interrupt sequence, the positive level is extended to IEO. Thus the IC nearest the positive source has the highest priority, and the IC furthest away has the lowest priority.

In the DMB32, CIO interrupts are software disabled, so although the daisy chain is implemented, it has no effect.

A.4.1.5 Register Selection – There are 48 registers in the CIO, so they cannot all be directly addressed by bits A<1:0>. For this reason, only the data registers are directly addressed. Control and status registers can only be accessed after the address has previously been written to a 'pointer' register.

Table A-4 shows how the address lines are decoded. An 0, 1 or 2 code on A<1:0>, directly accesses data registers for ports C, B and A respectively. A 1,1 code specifies a control operation. Higher address lines are decoded elsewhere in order to generate CE L (The address is $F400_{16}$ in DMB32). All access is barred unless CE L is asserted.

Table A-4 CIO Register Selection

Register Accessed	CE L	A1	A0
Port C	L	0	0
Port B	L	0	1
Port A	L	1	0
Control	L	1	1
None	H	x	x

x = don't care

The two stages of the control operation are as follows.

1. Write the register address to the pointer register
2. Read or write the addressed status or control register

Between stages 1 and 2, many of the CIO internal operations are suspended and status registers are not updated, so stage 2 should immediately follow stage 1.

Table A-5 lists the register set for the 8536, and gives the address (to be supplied on DAT<5:0>) and the type of access for each register. Note that the data registers can also be accessed directly (without using the address pointer).

Table A-5 8536 Registers

Internal Address (Binary)	Read/Write	Register Name
$A_5...A_0$		**Main Control Registers**
000000	R/W	Master Interrupt Control
000001	R/W	Master Configuration Control
000010	R/W	Port A Interrupt Vector
000011	R/W	Port B Interrupt Vector
000100	R/W	Counter/Timer Interrupt Vector
000101	R/W	Port C Data Path Polarity
000110	R/W	Port C Data Direction
000111	R/W	Port C Special I/O Control
		Most Often Accessed Registers
001000	*	Port A Command and Status
001001	*	Port B Command and Status
001010	*	Counter/Timer 1 Command and Status
001011	*	Counter/Timer 2 Command and Status
001100	*	Counter/Timer 3 Command and Status
001101	R/W	Port A Data**
001110	R/W	Port B Data**
001111	R/W	Port C Data**
		Counter/Timer Related Registers
010000	R	Counter/Timer 1 Current Count MS Byte
010001	R	Counter/Timer 1 Current Count LS Byte
010010	R	Counter/Timer 2 Current Count MS Byte
010011	R	Counter/Timer 2 Current Count LS Byte
010100	R	Counter/Timer 3 Current Count MS Byte
010101	R	Counter/Timer 3 Current Count LS Byte
010110	R/W	Counter/Timer 1 Time Constant MS Byte
010111	R/W	Counter/Timer 1 Time Constant LS Byte
011000	R/W	Counter/Timer 2 Time Constant MS Byte
011001	R/W	Counter/Timer 2 Time Constant LS Byte
011010	R/W	Counter/Timer 3 Time Constant MS Byte
011011	R/W	Counter/Timer 3 Time Constant LS Byte
011100	R/W	Counter/Timer 1 Mode Specification
011101	R/W	Counter/Timer 2 Mode Specification
011110	R/W	Counter/Timer 3 Mode Specification
011111	R	Current Vector
		Port A Specification Registers
100000	R/W	Port A Mode Specification
100001	R/W	Port A Handshake Specification
100010	R/W	Port A Data Path Polarity
100011	R/W	Port A Data Direction
100100	R/W	Port A Special I/O Control
100101	R/W	Port A Pattern Polarity
100110	R/W	Port A Pattern Transition
100111	R/W	Port A Pattern Mask
		Port B Specification Registers
101000	R/W	Port B Mode Specification
101001	R/W	Port B Handshake Specification
101010	R/W	Port B Data Path Polarity
101011	R/W	Port B Data Direction
101100	R/W	Port B Special I/O Control
101101	R/W	Port B Pattern Polarity
101110	R/W	Port B Pattern Transition
101111	R/W	Port B Pattern Mask

* All bits can be read and some bits can be written.
** Also directly addressable in Z8536 using pins A_0 and A_1.

RE244

A.4.2 Signal and Pinout

The functions of the signals to and from the 8536 CIO are described in Table A-6, the power supply and ground connections are included for completeness. The physical connections that correspond to the signals are identified in the pinout diagram Figure A-8.

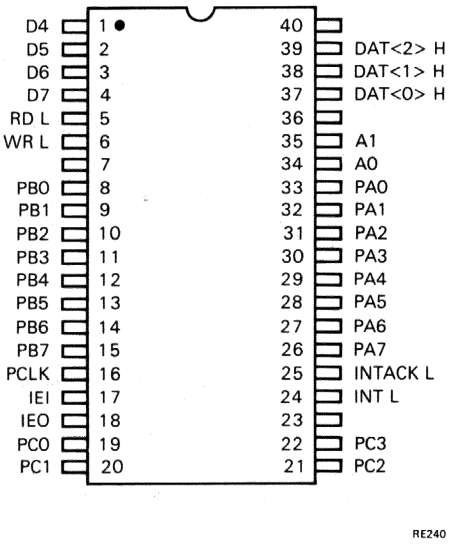


Figure A-8 CIO Pinout Details

Table A-6 8536 Signal Descriptions

Address Lines (A0, A1)	These two lines are used to select the register involved in the CPU transaction: Port A Data register, Port B Data register, Port C Data register, or a control register.
Chip Enable (CE)	A low level on this input enables the CIO to be read from or written to.
Data Bus Lines (D0-D7)	These eight data lines are used for transfers between the CPU and the CIO.
Interrupt Enable In (IEI)	IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.
Interrupt Enable Out (IEO)	IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only) IEO is connected to the next lower priority devices IEI input and thus inhibits interrupts from lower priority devices.
Interrupt Request (INT)	This signal is pulled Low when the CIO requests an interrupt.
Interrupt Acknowledge (INTACK)	This input indicates to the CIO that an Interrupt Acknowledge cycle is in progress INTACK must be synchronized to PCLK, and it must be stable throughout the Interrupt Acknowledge cycle.

Table A-6 8536 Signal Descriptions (continued)

Port A I/O Lines (PA0 to PA7)	These eight I/O lines transfer information between the CIO Port A and external devices.
Port B I/O Lines (PB0 to PB7)	These eight I/O lines transfer information between the CIO Port B and external devices. May also be used to provide external access to the Counter/Timers 1 and 2.
Port C I/O Lines (PC0 to PC3)	These four I/O lines are used to provide handshake, WAIT, and REQUEST lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the CIO Port C.
Peripheral Clock (PCLK)	This is the clock used by the internal control logic and the counter/timers in timer mode. It does not have to be the CPU clock.
Read (RD)*	This signal indicates that a CPU is reading from the CIO. During an Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the data bus if the CIO is the highest priority device requesting an interrupt.
Write (WR)*	This signal indicates a CPU write to the CIO.

* When RD and WR are detected low at the same time (normally an illegal condition), the CIO is reset.

A.5 2681 DUART

A.5.1 Overview

Figure A-9 shows the functional blocks of the 2681 Dual Universal Asynchronous Receiver Transmitter (DUART). There are control registers in every block except the bus buffer (which is a parallel holding register). It is via these registers that the DUART is programmed and monitored.

When the chip enable line (CEN) is low, the registers can be accessed by read or write actions of the host. Address lines A3 to A0 provide the address. RDN or WRN provides the timing and control. Commands, status, or data are transferred on the data lines D7 to D0. The operational control block manages these parallel operations.

Two serial data channels (A and B) perform the parallel/serial and serial/parallel conversion. Each TRANSMIT channel has a 2-byte buffer. This allows the next character to be loaded while the previous one is being transmitted. Each RECEIVE channel has a 4-byte buffer to allow for delays in interrupt response.

Also related to the serial interface are a 7-bit input port and an 8-bit output port. These lines can be used as individual, sense, and flag lines. Each line has a secondary function which may be used to provide modem control for the serial data lines.

A 3.6864 MHz crystal provides the basic timing for the timing block. This section contains a programmable counter/timer which can be programmed for many RECEIVE and TRANSMIT data rates. The counter timer can also be clocked by input port 2.

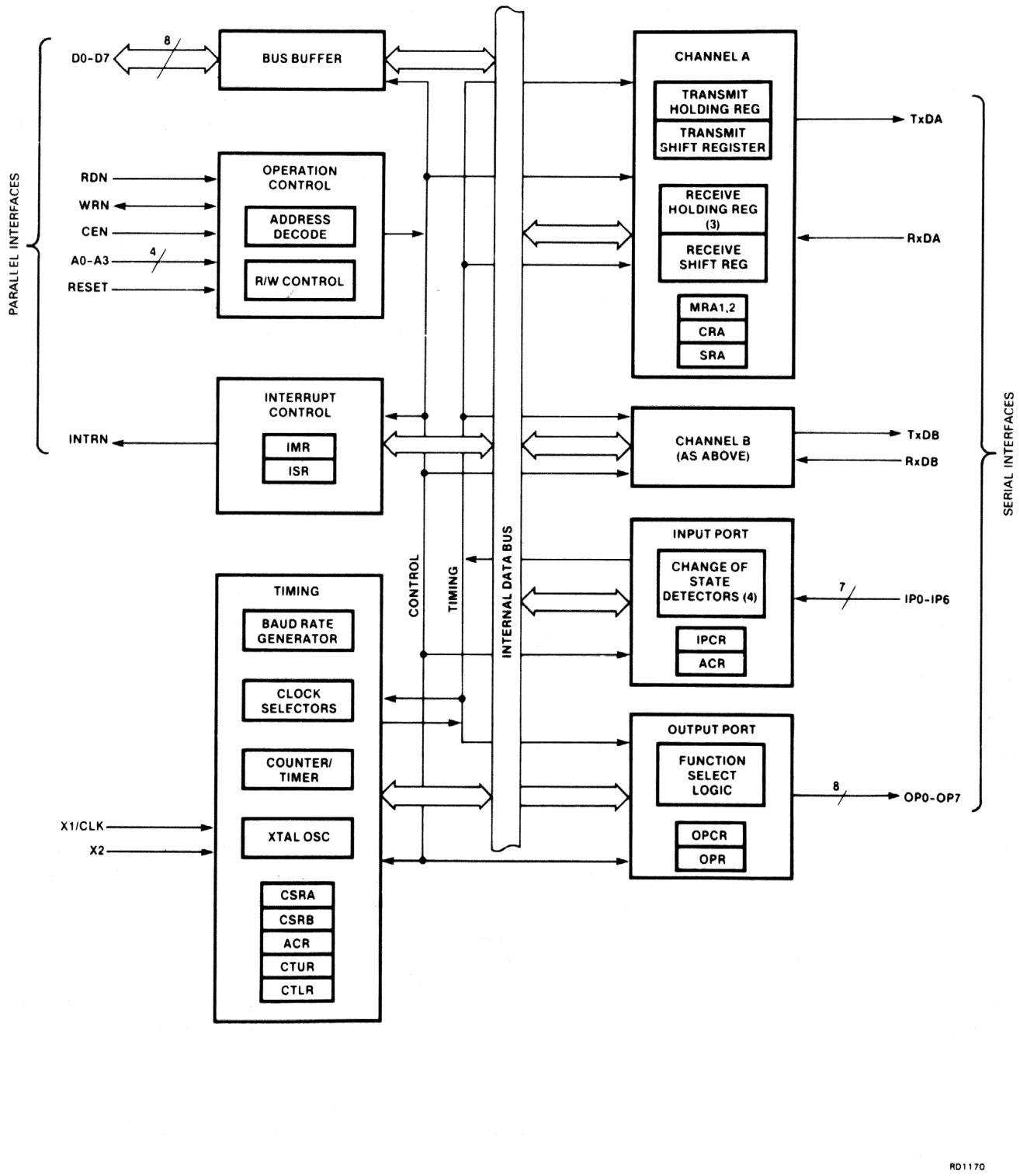


Figure A-9 2681 Dual Universal Asynchronous Receiver Transmitter

Interrupts are generated when at least one of eight maskable interrupt conditions occurs. INTRN will inform the controlling processor of changes in the DUART status. The interrupt routine should read status and then take the appropriate action.

The DUART can also be operated in the polled mode.

Characters to be transmitted must be written to the appropriate transmit holding register.

Received characters must be read from the appropriate receive holding register.

A.5.2 Signals and Pinout

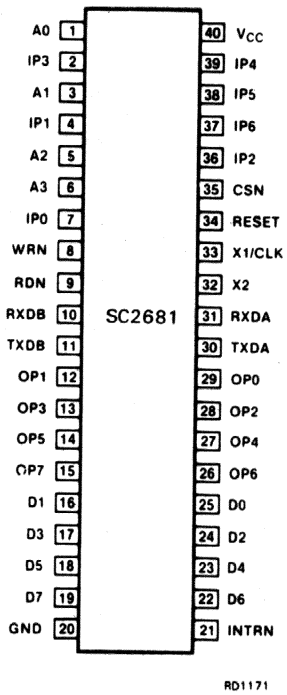


Figure A-10 2681 Pinout Diagram

A pinout diagram is provided in Figure A-10. The related signal functions are listed in Table A-7. This information applies only to the 40-pin DIL version of the 2681.

Table A-7 2681 Signal Descriptions

Data Bus (D0 to D7)	Used to transfer commands, data, and status between the DUART and the CPU. D0 is the least significant bit.
Chip Enable (CEN)	When low, data transfers between the CPU and the DUART are enabled on D0 to D7 as controlled by the WRN, RDN, and A0 to A3 inputs. When high, places the D0 to D7 lines in the 3-state condition.
Write Strobe (WRN)	When low, and CEN is also low, the contents of the data bus are loaded into the addressed register. The transfer occurs on the positive-going edge of the signal.
Read Strobe (RDN)	When low and CEN is also low, causes the contents of the addressed register to be placed on the data bus. The read cycle starts on the negative-going edge of RN.
Address Inputs (A0 to A3)	Select the DUART internal registers and ports for read/write operations.
Reset (RESET)	A high level clears the internal registers (SRA, SRB, IMR, ISR, OPR, OPCR), puts OP0 to OP7 in the high state, stops the counter/timer, and puts channels A and B in the inactive state with the TxDA and TxDB outputs in the mark (high) state.
Interrupt Request (INTRN)	A low level signals to the CPU that one or more of the eight maskable interrupting conditions are true.
Crystal 1 (X1/CLK)	Crystal or external clock input.
Crystal 2 (X2)	Connection for the other side of the crystal.
Channel A Receiver Serial Data Input (RxDA)	The least significant bit is received first. Mark is high, space is low.
Channel B Receiver Serial Data Input (RxDB)	The least significant bit is received first. Mark is high, space is low.
Channel A Transmitter Serial Data Output (TxDA)	The least significant bit is transmitted first. This output is held in the mark condition when the transmitter is disabled, idle, or when operating in local loopback mode. Mark is high, space is low.
(OP0 to OP7)	General purpose outputs.
(IP0 to IP6)	General purpose inputs.
(Vcc)	Power supply +5 V supply input.
(GND)	Ground.

APPENDIX B
GATE ARRAYS

B.1 OVERVIEW

This appendix contains brief functional descriptions, and signal information for the gate arrays used on the DMB32.

These arrays are:

- Address Translation Gate Array (ATGA) - Appendix B-2
- Control Gate Array (CGA) - Appendix B-3

B.2 ADDRESS TRANSLATION GATE ARRAY (ATGA)

B.2.1 Scope

The ATGA is a CMOS array designed to perform address translation tasks and to multiplex addresses for the DMB32 common address store, CASRAM. Arbitration between ports is not an ATGA task, but is done by the CGA and BIIC.

A pinout diagram of the ATGA is shown in Figure B-1.

B.2.2 Derivation Of CASRAM Addresses

Address translation functions performed by the ATGA are shown in Figure B-2. The inversion function applied to the secondary address is to correct for differences between VAX-11 and 68000 addressing structures.

Address outputs RA<10:00> from the ATGA can be of eight types.

1. Sync FIFO Address
2. RX FIFO Address
3. TX Action FIFO Address
4. DMA File Address
5. '0' Address (all zeroes)
6. Index from Async Base Offset
7. Index from Sync Base Offset
8. Direct (not translated)

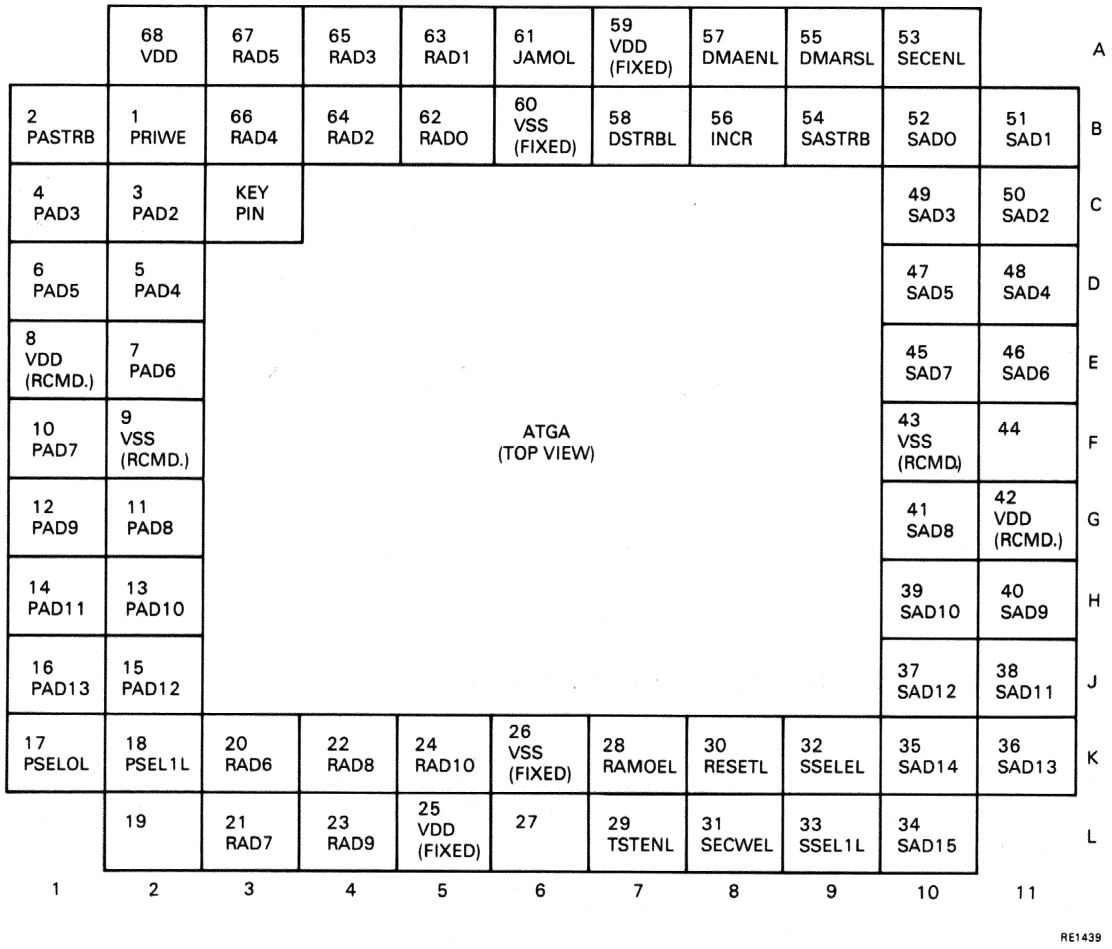


Figure B-1 ATGA Pinout Diagram

Primary (VAXBI) addresses are decoded to identify an address:

- Of a directly addressed register (no translation)
- Of a channel-specific register
- Of a FIFO

If the address is that of a directly addressed register, it is passed untranslated to RA<10:00>.

If the address is that of a channel-specific register, it is indexed by the channel number held in the ACSR or SCSR registers.

If the address is that of a FIFO, the appropriate FIFO controller is enabled, and the address of the next available location in that FIFO is output.

Except when they point to a FIFO, all secondary addresses are direct, and are passed, after inversion, directly to RA<10:0>.

FIFO addresses supplied on the secondary port are decoded to identify the FIFO. The appropriate FIFO is then enabled.

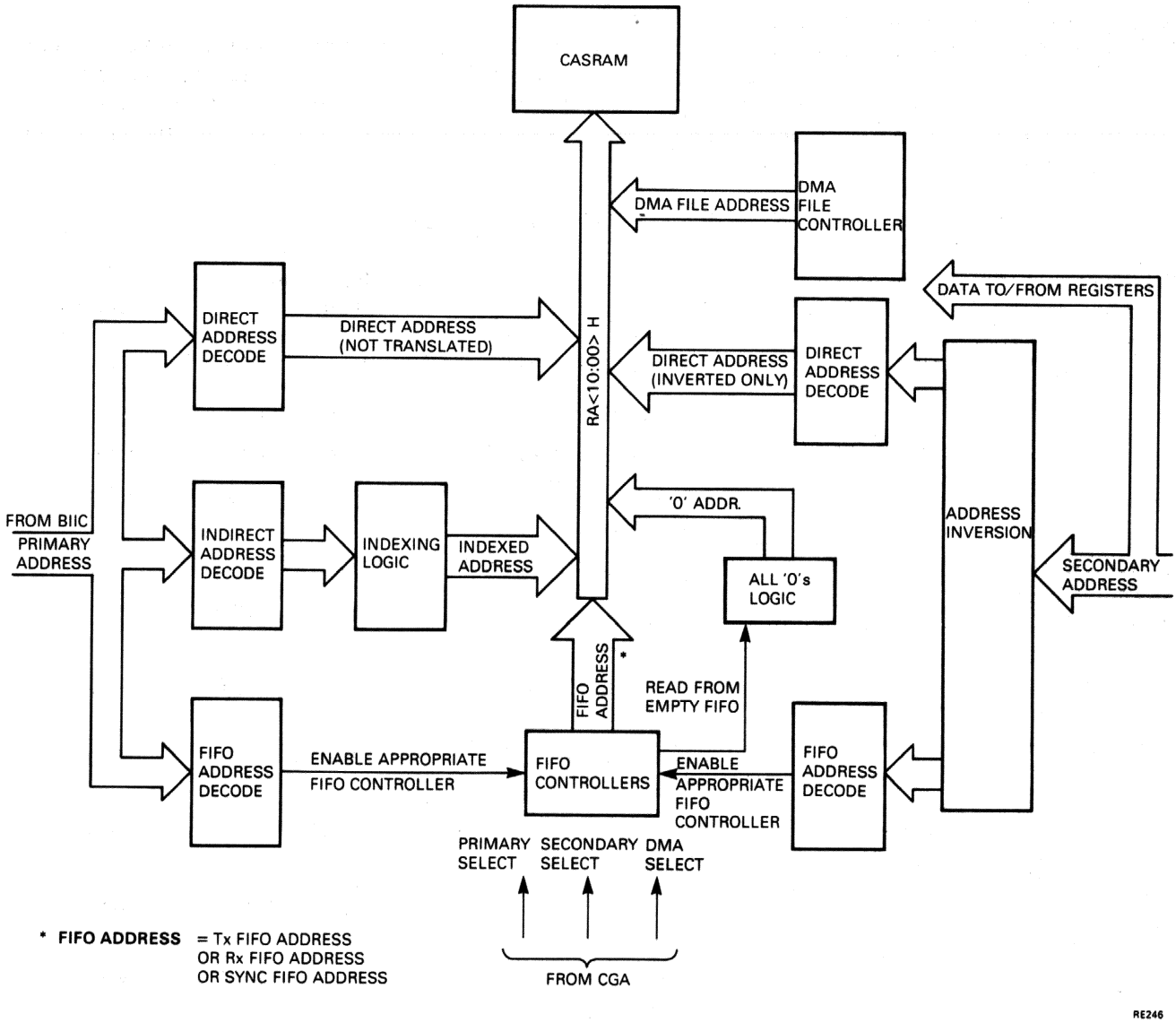


Figure B-2 Representation of Address Translation Functions

In DMA transactions, the DMA file controller provides a pointer to a DMA buffer address stored in CASRAM. The controller then functions as an address pointer for a DMA file established in CASRAM. Before initiating a DMA transaction, the microprocessor loads the DMA file register with the appropriate pointer address.

'0' address is an 'all zero' address that is output if JAM "0" L is asserted or an empty FIFO is read.

To access the ATGA internal registers, data is transferred via the multiplexed secondary address/data bus. Data is transferred without being inverted (see Figure B-2), to and from registers in the blocks marked *.

NOTE

The primary-port normally has no access to ATGA registers, but when certain registers in CASRAM are written, the data is also written to the appropriate ATGA register.

B.2.3 ATGA Register Overview

To support its address translation functions, the ATGA holds address information. For example, to index an async address, the ATGA must hold the ASYNC.IND.ADDR bits from the ACSR. This information is held in the internal ATGA registers listed in Table B-1, which also indicates the addresses and access restrictions for both the primary and secondary ports. Note that secondary addresses are the inverse of primary addresses.

Table B-1 ATGA Register Map

Primary Address*	Access	Register	Secondary Address**	Access
Base + 104	WO	Async CSR (ACSR)	Base + 1EF8	RO
Base + 108	WO	Sync CSR (SCSR)	Base + 1EF4	RO
Base + 220	RO	*** FIFO Status (FSTAT)	Base + 1DDC	RO
Base + 224	WO	*** DMA File Address (DFADD)	Base + 1DD8	WO
Base + 228	WO	*** FIFO Reset (RESET)	Base + 1DD4	WO
Base + 22C	RO	*** Last RAM Address (LRADD)	Base + 1DD0	RO
Base + 230	WO	*** Tx Act FIFO Base (TABASE)	Base + 1DCC	WO
Base + 234	WO	*** Sync FIFO Base (SFBASE)	Base + 1DC8	WO
Base + 238	WO	*** Rx FIFO Base (RFBASE)	Base + 1DC4	WO
Base + 23C	WO	*** Sync CSR Base (SCBASE)	Base + 1DC0	WO
Base + 240	WO	*** Async CSR Base (ACBASE)	Base + 1DBC	WO

* Base is decode of VAXBI ID

** Base is decode of 68000 address bits A<15:12>

*** Accessible in maintenance mode only

WO = Write only. RO = Read Only.

NOTE

ATGA registers are only 16 bits wide. The most significant word of the longword address is in CASRAM.

The async and sync CSRs exist in the CASRAM, but they are also duplicated in the ATGA. The ATGA registers hold only a subset of the full CSR contents.

B.2.4 SIGNAL DESCRIPTIONS

The ATGA has 55 Signal connections and 8 power and ground connections. Each of the signals is described in Table B-1.

Table B-2 ATGA Signal Descriptions

VDD	+5 V power connections (4 off)
VSS	Ground connections (4 off)
PA<12:02> H	The primary address port consists of 11 lines which are used to pass address information to the ATGA. When interfaced to the BIIC signals PA<12:02> H have a one to one correspondence to BCI D<12:02> H.
PA<13> H	PA<13> is an additional primary address input and is used to enable direct mapping of all 2 Kbyte Common RAM Store locations. On the DMB32, PA<13> is connected to BCI D<22> H and is used to detect 'Adapter Window Space' accesses. This feature is provided for diagnostic use.
SAD<15:00> H	The secondary address port consists of 16 bidirectional lines which provide a multiplexed information path, used to pass address and data information to and from the ATGA. The input address is assumed to run from A0 to A15, thus the interface from the 68000 should drive the least significant bit to a logic 0 when accessing the Common Address Store. This is required since the 68000 does not provide an A0 output, but uses dual data strobes. The ATGA will not use the A0 input since it only produces Longword address information for Common RAM Store Access, and permits only longword or word length accesses to internal registers. Byte and word accesses of CASRAM are controlled through the derivation of the RAM selects which on the DMB32 module are generated by the CGA.
DATASTRB L	The data strobe signal is an active low input which is used to enable data information onto the secondary bus. The direction of data flow on this bus is determined by SECWE L for secondary port access and PRIWE L for primary port access. The DATASTRB L signal must be active during data transfers initiated from either port, and for both input and output. Data I/O to the BIIC passes via the secondary bus, not directly onto the primary bus. On the DMB32 module this permits the data hold time to be controlled by the CGA because the data is held in the latches which separate the primary and secondary buses.
PASTRB H	The primary port address strobe is an active high signal which indicates the start of a cycle on the PA<15:00> H lines. This signal allows an address to be latched by the primary port during a secondary port access. This condition arises when DTACK has been issued to the microprocessor by the CGA. PASTRB H is directly used to latch the valid address information. In the DMB32, PASTRB H is connected to BCI CLE H from the BIIC.

Table B-2 ATGA Signal Descriptions (continued)

SASTRB H	The secondary address strobe is an active high signal which, when asserted, causes the secondary port address latches to become transparent, and latches the SAD<15:0> lines when de-asserted.
SECENB L	The secondary port enable signal indicates which of the two ports is allowed to access the CASRAM. A high logic level indicates that the primary port has access.
PRIWE H	PRIWE H is the primary port write enable signal. This signal indicates whether the present cycle is a Read or Write access. In the DMB32, PRIWE H is connected to BCI I<2> H and is latched by the de-asserting edge of PASTRB. The use of BCI I<2> as a WE H signal causes the following VAXBI commands to be interpreted as WRITE commands; WRITE, WCI, UNLKM, WMCI, INIT, INVAL, BROADCAST and IP INTR.
SECWE L	The secondary port write enable signal indicates whether a current secondary port cycle is a Read or Write (asserted = write).
INCR H	This is an active high level input signal, the asserting edge of which causes the DMA file address register value to be incremented by one (longword) address. (Providing that DMA ENB L is asserted).
RA<10:00> H	The 11 RAM address lines form the translated address information derived from the incoming address of either port. Note that address information passing via the ATGA from the secondary port is inverted prior to output on the RA<10:00> lines. This feature caters for the differences between the data structures of the 68000 microprocessor and the host.
SECSEL<1:0> L	The secondary port select signals are active low inputs and when both asserted, enable the ATGA to translate the address information on the secondary port. These signals must remain asserted throughout the information transfer. If the address bit SAD<13> is set to a logic "1" then the 8 Kbytes of Common RAM Store will be directly addressed. (The FIFOs and internal CSRs will be disabled and all CASRAM locations will be directly mapped). For primary port transactions, the CGA determines whether the addressed location is accessible. (The CGA has control of the data path routing logic and the Common RAM Store enable signals).

Table B-2 ATGA Signal Descriptions (continued)

PRISEL<1:0> L	If either of the two primary port select signals are asserted, the ATGA is enabled to decode the primary port address. Use of two input selects permits the use of the BIIC select code outputs. On the DMB32, PRISEL<0> L is connected to BCI SC<0> L, to enable detection of VAXBI accesses to user CSR Space. PRISEL<1> L is connected to BCI SC<1> L to enable detection of VAXBI accesses to VAXBI adapter window space. The PRISEL <1:0> L signal is not required to be active throughout a transfer. The current transfer is determined to be completed on the de-asserting edge of the DATASTRB L signal, whereupon the RA<10:00> H lines shall return to their default "0" address.
DMAENB L	DMA enable is an active low input signal which when asserted, causes the contents of the DMA file address register to be driven onto the RA<10:00> output lines. Assertion of this signal enables the DMA file address counter to be incremented by INCR H.
JAM "0" L	Assertion of the signal forces all 0s on the RA<10:00> lines.
DMARST L	When asserted, DMARST L causes the DMA file address counter to be reset to its initial condition. In the event of an error this enables the CGA to retry certain transfers, without intervention from the microprocessor.
RESET H	When asserted, RESET H initiates a total chip reset. This signal must be asserted at power-on. This operation is externally identical to the soft reset provided by the chip reset bit within the internal reset register.
ROE L	RAM address output enable is an active low signal, and, when asserted enables the RA<10:00> H lines. If this signal is deasserted, these address lines adopt a high impedance state. This feature is not used on the DMB32, but can be used during device test. An internal pull down resistor is provided, so that the RA<10:00> H lines are enabled if left open circuit.
TESTENB L	This is an active low signal, which, if asserted, enables the test functions within the ATGA. An internal pull up resistor is provided and the signal is deasserted if left open circuit. Assertion of TESTENB affects the operation of RXFIFO and is used only in device testing. If the signal is asserted with DMARSR L, then PA<22>, PA<12:02> and PRIWE become outputs, providing a test facility for the address decode logic.

B.3 CONTROL GATE ARRAY (CGA)

B.3.1 Scope

The CGA is a large scale integrated circuit, designed to provide an interface to the BIIC, as required by I/O nodes such as the DMB32. The major components of the CGA are identified as follows:

- BCI Slave Port Sequencer
- BCI Master Port Sequencer
- Busmux Sequencer

An additional feature of the CGA is that it is able to handle the majority of the DMB32 68000 memory/peripheral decoding tasks. The user interface to the CGA is via a 16-bit address/data multiplexed bus, whose control is internal to the CGA.

A pinout diagram of the CGA is shown in Figure B-3

120 BCIEVL0	117 BCIEVL3	116 BCIEVL4	113 RDL	111 ENBMICP OLSWL	108 IOSELL	107 RAMSELL	104 A12	101 A15	99 BUSRQL	96 LTCHUPA DD	93 ENBBCIC ASL	90 BUSCEL
3 LDSL	2 UDSL	119 BCIEVL1	115 WRLDSL	112 DATACAS UP	109 INTRGL	105 VDD (FIXED)	103 A13	100 CASSELL	97 ENBUFAD DRL	95 ADDRCAS UP	92 LATCHCA SL	87 FC1
6 BCI IDLO	5 SECWEL	1 VSS (RCMD)	118 BCIEVL2	114 WRUDSL	110 ENBMICR OMSWL	106 VSS (FIXED)	102 A14	98 DTACKL	94 LATCHBC IL	91 VSS (RCMD)	89 INTACKL	86 FCO
9 BCIMABL	7 AIOMHZ	4 ASL	KEY PIN							88 FC2	85 ROEL	83 RCEL3
11 BCIIDL2	10 RST	8 BCIIDL1								84 RWEL	82 RCEL2	81 RCEL1
14 BCITIME L	13 BCIDCLO L	12 BCIPHAS EL								80 RCEL0	79 DECODEE NB	78 TESTENB L
17 BCIRSLO	15 VDD (FIXED)	16 VSS (FIXED)				CGA (TOP VIEW)				76 VSS (FIXED)	75 VDD (FIXED)	77 BCIINTL 4
18 BCIRGLO	19 BCIRGLO	20 BCIRSL1								72 CAD13	73 CAD14	74 CAD15
21 BC)IDL3	22 BCINXTL	24 BCISELL								68 SAD9	70 SAD11	71 SAD12
23 BCIRAKL	25 BCIMDEL	28 BCII3								64 BCID9	67 SAD8	69 SAD10
26 BCII2	29 BCIIO	31 VSS (RCMD.)	34 JAMOL	38 INCR	42 BCID11	46 VSS (FIXED)	50 BCIDI	54 SAD1	58 SAD5	61 VSS (RCMD.)	65 BCID30	66 BCID31
27 BCII1	32 BCICLE	35 SECENBL	37 DMARESE TL	40 DATASTR BL	43 BCID10	45 VDD (FIXED)	49 BCID6	52 BCID7	55 SAD2	59 CAD6	62 BCID8	63 BCID3
30 BCISDEL	33 DIAGFAI LM	36 DMAENBL	39 SASTRB	41 BCID12	44 BCID22	47 BCID4	48 BCID5	51 BCID2	53 SAD0	56 SAD3	57 SAD4	60 SAD7

RE1438

Figure B-3 CGA Pinout Diagram

B.3.1.1 Overview – VAXBI slave port transfers are conducted by the BIIC independent of any external control, while VAXBI master port transfers require only the setting up of an address/data file and a write to the BCOM register to initiate the transfer.

Master and slave sequencers are independent in their operation and will handle transfers from longword through to octaword in length. This independence permits the use of loopback transactions, either locally, using the BIIC LOOPBACK, or by the node initiating a VAXBI transaction addressing its own node/window space.

The CGA provides a programmable number of STALL cycles for VAXBI, Busmux and 68000 transactions.

Unimplemented CSRs receive NOACKs on the VAXBI during normal operation, although all 8 Kbytes of node space is accessible in maintenance mode. In addition up to 256 Kbytes of the 68000 memory space, are accessible via VAXBI window space in maintenance mode.

VAXBI interrupts are supported at one of 3 programmable levels via the BCI INT<7:4> L lines, the resulting levels being dependent on the connection of these three lines.

BIIC external vectors are supported via a CGA internal interrupt vector register capable of supplying one of 3 prioritized codes, dependent on the interrupting condition defined by the BINT register within the CGA.

A single interrupt is provided for use by the 68000, which may be masked for general VAXBI and/or specific CSR activity.

Figure 3-14 in Chapter 3 shows how the CGA may be interfaced to the BCI and a 68000 bus, and shows its use on the DMB32 module.

B.3.2 CGA Register Overview

The internal registers of the CGA are listed in Table B-3. The table gives the addresses and access restrictions for both the primary and secondary ports. Note that secondary addresses are the inverse of primary addresses.

Table B-3 CGA Register Map

Primary Address*	Access	Register	Secondary Address**	Access
Base + 100	RW	Maint CSR (MAINT)	Base + 1EFC	RW
Base + 104	WO	Async CSR (ACSR)	Base + 1EF8	WO
Base + 108	WO	Sync CSR (SCSR)	Base + 1EF4	WO
Base + 10C	WO	Printer CSR (PCSR)	Base + 1EF0	WO
Base + 280	RO	*** Async Line Param (ALPRC)	Base + 1D7C	RO
Base + 284	RO	*** Async Line Control (ALC)	Base + 1D78	RO
Base + 288	RO	*** Async Preempt (APC)	Base + 1D74	RO
Base + 28C	RO	*** CSR Change (CSRC)	Base + 1D70	RO
Base + 290	WO	*** Device Setup CRS (DSCSR)	Base + 1D6C	WO
Base + 294	WO	*** VAXBI Command (BCOM)	Base + 1D68	WO
Base + 298	WO	*** VAXBI Data Mask (BMSK)	Base + 1D64	WO
Base + 29C	WO	*** VAXBI Vector (BVEC)	Base + 1D60	WO
Base + 2A0	RW	*** VAXBI Interrupt (BINT)	Base + 1D5C	WO
Base + 2A4	WO	*** Micro Delay (MDEL)	Base + 1D58	RW
Base + 2A8	RO	*** VAXBI Summary (BSUM)	Base + 1D54	RO
Base + 2AC	WO	*** VAXBI Delay (BDEL)	Base + 1D50	WO
Base + 2B0	WO	*** Micro Interrupt (MIMK)	Base + 1D4C	WO

* Base is decode of VAXBI ID

** Base is $C000_{16}$, if internal CGA decode is selected

*** Accessible in maintenance mode only

WO = Write only. RO = Read Only. RW = Read/Write.

NOTE

CGA registers are only 16 bits wide. The most significant word of the longword address is in CASRAM.

The async and sync CSRs exist in the CASRAM, but they are also duplicated in the CGA. The CGA registers hold only a subset of the full CSR contents.

B.3.3 Signal Descriptions

The Control Gate Array has 108 Signal connections and 12 power and ground connections. Each of the signals are described in Table B-4.

Table B-4 CGA Signal Descriptions

Data Signals	
PA<12:01> H	The primary address port consists of 11 lines which are used to pass address information to the CGA. When interfaced to the BIIC signals PA<12:01> H have a one-to-one correspondence to BCI D<12:01> H PA<9:3> H are additionally used as device test outputs.
PA<13> H	The PA<13> H is an additional primary address input and is used to enable direct mapping of all 2 Kbyte Common RAM Store locations. On the DMB32 module PA<13> H is connected to BCI D<22> H to provide recognition of adapter window space access as distinct from node space access. This feature is provided for diagnostic use.
PA<15:14> H	PA<15:14> H indicate the number of longwords associated with a VAXBI transaction: **15 14 Transfer length** 0 0 Illegal 0 1 Longword 1 0 Quadword 1 1 Octaword PA<15:14> are connected to BCI<31:30> on the DMB32.
SAD<15:00> H	The secondary address port consists of 16 bidirectional lines which provide a multiplexed information path, used to pass address and data information to and from the CGA. On the DMB32 module this bus is interfaced to a microprocessor bus via an external interface. The timing of this multiplexed bus is controlled by the data and address strobe signals generated by the CGA. The input address is assumed to run from A0 to A15, thus the interface from the 68000 should enable the A1 address bit onto SAD<1>, SAD<0> being ignored.

Table B-4 CGA Signal Descriptions (continued)

Busmux Interface Control Signals	
LTCH UP ADDR H	This signal is an active high output which is used to latch primary port addresses into the Busmux interface address latches. Address information is latched on the asserting edge of LTCH UP ADDR H. This signal will be asserted during primary port accesses of the 68000 private bus.
DATA CAS UP H	This is an active high output signal and when asserted causes the Busmux interface data transceivers to pass data from the Busmux to the 68000 private bus, and vice-versa when deasserted.
ADDR CAS UP H	This is an active high output signal and when asserted causes the Busmux interface address latches to pass information from the Busmux to the 68000 private bus, and vice-versa when deasserted.
ENB UP ADDR L	This active low output signal is used to enable the Busmux interface address latches. The direction of information flow is controlled by ADDR CAS UP H.
ENB MICRO LSW L	This active low signal permits the data transceivers of the Busmux interface connected to the least significant word of the 32-bit CAS BUS. The direction of data flow through the interface is controlled by DATA CAS UP H.
ENB MICRO MSW L	This active low signal permits the data transceivers of the Busmux interface connected to the most significant word of the 32-bit CAS BUS. The direction of data flow through the interface is controlled by DATA CAS UP H.
Primary Port (BCI) Data Path Control Signals	
LATCH CAS L	The asserting edge of this active low signal is used to latch information from the 32-bit CAS BUS into the BCI input latches.
LATCH BCI L	The deasserting edge of this active low signal is used to latch information from the BCI D<31:0> H lines into the BCI output latches.
ENB BCI CAS L	This active low signal is used to enable the BCI output latches onto the 32-bit CAS BUS. The associated enabling of the BCI input latches is not under the control of the CGA but these latches must be enabled by the assertion of BCI MDE L or BCI SDE L.

Table B-4 CGA Signal Descriptions (continued)

CASRAM Control Signals	
CAS OE L	This signal is used to provide the output enable control for the Common Address Store. The period of assertion of CAS OE is controlled by the BDEL register for VAXBI slave accesses, and the MDEL Register for Busmux accesses. For master sequencer reads of the CASRAM the CAS OE L assertion period is dependent on the slave node response, but has a minimum pulse width of 200 ns (nominal).
CAS WE L	This signal is used to provide the write enable control of the Common Address Store RAM. The period of assertion of CAS WE is controlled by the BDEL register for VAXBI slave accesses, and the MDEL register for Busmux accesses. VAXBI master sequencer writes to the CASRAM use a fixed 75 ns (nominal) CAS WE assertion period.
CAS CS<3:0> L	These signals are used to provide the chip select control of the Common Address Store RAM. The period of assertion of CAS CS<3:0> is controlled by the BDEL register for VAXBI slave accesses, and the MDEL register for Busmux accesses. For master sequencer reads of the CASRAM the CAS CS<3:0> L assertion period is dependent on the slave node response, but has a minimum pulse width of 200 ns (nominal).
68000 Control Signals	
SECWE L	The secondary write enable signal indicates whether a current secondary port cycle is a read or write (active low) operation. This signal is an input at power up, and during normal operation, but is an output during mastership of the 68000 private bus. This signal is connected to the 68000 R/W signal.
SECSEL<1:0> L	The secondary port select signals are bidirectional signals. When configured as inputs they are active low signals, which, when asserted enable the CGA Busmux sequencer to initiate a transfer via the secondary port. These signals must remain asserted throughout the information transfer. SECSEL<0> is connected to the 68000 address strobe (AS L), and is bidirectional, being driven by the CGA during mastership of the 68000 private bus. SECSEL<1> is decoded from the 68000 address lines. The required decode logic for DMB32 is provided within the CGA, and this signal is an output, when DEC ENB H is asserted. If the signal DEC ENB H is deasserted SECSEL<1> is an input, permitting the use of any external decode.

Table B-4 CGA Signal Descriptions (continued)

INTRQ L	This is an active low signal, which, when asserted causes an interrupt to the 68000 microprocessor. The assertion of this signal may be prevented through use of the MIMK register in the CGA. The signal will remain asserted until a read of the interrupt register has occurred, indicating that the interrupt service routine has been entered. This implementation of interrupt acknowledgment does not support "daisy chaining" of interrupting devices, nor the programmed vector capability of the 68000. On the DMB32 the INTRQ L line will be connected to the interrupt line from the four 2681 DUARTs on the board.
UDS L	This active low signal is used to indicate participation of the upper byte in any 68000 transfer. It is an input at power-up and under normal operating conditions. During CGA mastership of the 68000 private bus, only possible in maintenance modes, it is an output fulfilling the 68000 timing requirements.
LDS L	This active low signal is used to indicate participation of the lower byte in any 68000 transfer. It is an input at power-up and under normal operating conditions. During CGA mastership of the 68000 private bus, only possible in maintenance modes, this signal is an output fulfilling the 68000 timing requirements.
DTACK L	This active low output signal is used to provide acknowledgment for 68000 bus cycles. If the input DEC ENB H is asserted, DTACK is produced for all internally generated select signals (memory and I/O). The number of WAIT cycles involved in memory/peripheral accesses is programmable: • 0 to 3 cycles for each of two banks of I/O • 0 or 1 cycles for ROM and RAM • 1 to 7 cycles for CAS • 6 or 7 cycles for INTACK acknowledge cycles through use of the MDEL register If the input DEC ENB H is not asserted then DTACK L will be issued only for accesses to CASRAM, (CAS SEL L assertion) thus permitting the use of external decode for all devices on the 68000 private bus.
BUSGR L	This active low signal is used to indicate that the 68000 will release control of its bus at the end of the current bus cycle. Therefore all related CGA output signals remain tri-stated until AS L is also deasserted.

Table B-4 CGA Signal Descriptions (continued)

BUSRQ L	This active low output signal is used to request mastership of the 68000 private bus, to permit access to this bus from the VAXBI. The signal will remain asserted throughout CGA bus mastership, and the 68000 acknowledgment signal BGACK L is not used. Maintenance modes 1 and 2 enable the 68000 private bus to be accessed from the VAXBI, the CGA taking control of the 68000 bus through BUSRQ L. In maintenance mode 1, control of the 68000 bus will be requested for any access to window space only and released after access is completed. Maintenance mode 2 causes the CGA to request control of the 68000 bus. Control will be retained until maintenance mode 1, or no maintenance mode, is selected. Maintenance mode 3 causes the CGA to request control of the 68000 private bus, but prevents any control signals from being driven. This allows an external test device to drive the 68000 control signals and verify the operation of the CGA micro-interface logic, the CGA Busmux sequencer, all devices on the 68000 private bus and the CASRAM.
WR UDS L	Write upper data strobe is an active low signal generated from the logical AND of UDS L with SECWE L (68K R/W). It is synchronized to the 10 MHz H output.
WR LDS L	Write lower data strobe is an active low signal generated from the logical AND of LDS L with SECWE L(68K R/W). It is synchronized to the 10 MHz H output.
RD L	Read is an active low output signal generated during 68000 read cycles. For accesses to memory, RD L is the logical AND of AS L with the inverse of SECWE L(68K R/W). For devices in I/O bank 2, the time from the 68K AS L assertion to RD L assertion is programmable, controlled by a field in the MDEL register.
INTACK L	Interrupt acknowledge is an active low signal generated from the logical AND of the 68000 function code signals (FC<2:0>) with the inverse of AS L. Note that FC<2:0> L are not driven when the CGA has control of the 68000 bus, and INTACK L is gated off by the assertion of BUSGR L to avoid this output being indeterminate.
RAM SEL L	RAM select is an active low signal generated during 68000 accesses to the address range 32 to 40 Kbytes. RAM SEL L is derived purely from 68000 address bits 13 through 15 (A<15:13>).

Table B-4 CGA Signal Descriptions (continued)

I/O SEL L	I/O select is an active low signal generated during 68000 accesses to the address range 56 to 64 Kbytes. I/O SEL L is derived purely from 68K address bits 13 through 15 (A<15:13>).
FC<2:0> H	The three function code bits are used to determine when the 68000 is executing an interrupt acknowledge cycle, refer to INTACK L.
A<15:12> H	Address bits 13 through 15 are used to derive the I/O SEL L and RAM SEL L output signals. A<12> is used internal to the CGA to enable different DTACK L and RD L Signal delays for each of two I/O banks.
10MHZ H	This is a 10 MHz clock signal derived from the 20 MHz clock, TIME L, and synchronized to the 5 MHz clock, PHASE L. 68000 bus cycles are synchronized to VAXBI bus cycles by use of this signal as the 68000 clock input.
ATGA Control Signals	
DATASTRB L	The data strobe signal is an active low level output used to enable data information onto the secondary bus. The direction of data flow on this bus is determined by the WE L signal, (SECWE L for secondary port access and PRIWE L for primary port access). Therefore, the DATASTRB L signal must be active during data transfers initiated from either port, and for input and output. DATASTRB L is asserted for writes to the least significant byte or for any read of the least significant word, of any CASRAM longword CSR. Exceptions to this are accesses to RAM by the master sequencer, when there will be no assertion of DATASTRB L, and during VAXBI IDENT operations with BIIC external vector fetch. Data I/O to the BIIC passes via the secondary bus rather than directly onto the primary bus. This permits a reduction in the gate array I/O count and assists in reducing the complexity
SECENB L	The secondary port enable signal indicates which of the two ports has been permitted access to the RAM Store, a high logic level indicating that the primary port presently has access. By default this signal is asserted, being deasserted only during primary port accesses.
SASTRB H	Secondary address strobe is an active high signal which is used to latch address information from the Busmux, during a secondary port access (68K) of CAS, on its deasserting edge.
DMAENB L	DMA enable is an active low output signal, used to enable the DMA file address register, within the ATGA, onto the CASRAM address lines.

Table B-4 CGA Signal Descriptions (continued)

DMARST L	DMA reset is an active low output signal and when asserted causes the least significant 3 bits of the DMA file address to be reset to their initial value (111).
INCR H	This is an active high level output signal, which, on the asserting edge, causes the DMA file address value, held within the ATGA, to be incremented by one (longword) address. This is providing that the DMAENB L signal is asserted. When this output is asserted within DMAENB L, the slave sequencer within the CGA attempts to increment a longword count for slave transfers of quad/octa word length. The assertion of INCR without DMAENB causes no action.
RST OUT H	Reset out is an active high output signal generated from BCI DCLO L, or the programmed reset bit within the MAINT Register. The period of RST OUT H assertion is equal to that of BCI DCLO L, or, for a programmed reset, is a minimum of 800 ns.
JAM 0 L	This is an active low output signal which will be asserted by the CGA during VAXBI IDENT transactions involving the module. This provides for BIIC external vector support, in that the most significant word of the external vector is fetched from location '0' of CASRAM. The ATGA will jam all 0s onto the CASRAM address lines in response to the assertion of JAM0 L.
BIIC Interface Signals	
PASTRB H	The primary port address strobe is an active high signal which indicates the start of a cycle on the PA<15:00> H lines. This signal permits an address to be latched by the primary port during a secondary port access. This condition arises when an acknowledgment, DTACK L, has been issued to the active device on the secondary port. PASTRB H is used directly to latch the valid address information and on the DMB32 module, is connected to BCI CLE H from the BIIC.
BCI SEL L	The BCI select signal is an input to the CGA, and when asserted, indicates the valid selection of this node by a VAXBI transaction. The BIIC checks for bus errors, verifies node ID, and asserts the SEL signal if the transaction type has been enabled via the BCI control register.

Table B-4 CGA Signal Descriptions (continued)

BCI RS<1:0> L

The response codes RS<1:0> L are active low outputs controlled by the CGA slave sequencer.

1 0	Description	Possible BIIC outputs
H H	NOACK	NOACK
H L	ACK	ACK, NOACK
L H	STALL	STALL, ACK or NOACK
L L	RETRY	RETRY, NOACK

The default output for the CGA slave sequencer is NOACK, and the RETRY code cannot be generated by the CGA.

BCI RQ<1:0> L

The request lines, RQ<1:0>, are active low output signals enabled by the CGA master sequencer, the enabled value being determined from the content of the BCOM register.

1 0	Description
H H	No Request
H L	VAXBI Transaction Request
L H	Loopback Request
L L	Not Used

BCI EV<4:0> L

These 5 lines are used to provide information on transfer completion/errors on the VAXBI bus, as detected by the BIIC.

The codes detected are:

Value (Hex)	Mnemonic	Description
00	NO EVENT	
01	MCP	Master Action Complete
02	AKRSD	ACK Rcv for Slave Read Data
03	BTO	Bus Time Out
04	STP	Self Test Pass
05	RCR	Retry CNF Rcv
06	IRW	BIIC Internal Reg Written
07	ARCR	Advanced Retry
08	NICI	NOACK/ILLEGAL CNF for INTR
09	NICIPS *	NOACK/ILLEGAL IP INTR/STOP
0A	AKRE	ACK Rcv for ERROR VECTOR
0B	IAL	IDENT ARB Lost
0C	EVS4	Ext. Vector Sel @ level 4
0D	EVS5	Ext. Vector Sel @ level 5
0E	EVS6	Ext. Vector Sel @ level 6
0F	EVS7	Ext. Vector Sel @ level 7
10	STO	Stall Time Out (Slave Error)

Table B-4 CGA Signal Descriptions (continued)

11	BPS	Bad Parity For Slave
12	ICRSD	Illegal CNF for Slave Data
13	BBE	VAXBI BUSY error
14	AKRNE4	ACK Rcv Non-Err Vect @level 4
15	AKRNE5	ACK Rcv Non-Err Vect @level 5
16	AKRNE6	ACK Rcv Non-Err Vect @level 6
17	AKRNE7	ACK Rcv Non-Err Vect @level 7
18	RDSR	RD DATA SUBS. or reserved code
19	ICRMC	Illegal CNF for Command
1A	NCRMC	NOACK Rcv for Command
1B	BPR **	BAD PARITY received
1C	ICRMD	Illegal CNF for data
1D	RTO	Retry Time out
1E	BPM	Bad Parity, Master
1F	MTCE	Master Xmit Check error

* NICIPS only output if IPINTR FORCE bit in the BCI control register is used; if initiated via master port then NCRMC or ICRMC will be output.

**BAD PARITY is internally gated to determine whether the error was slave or master related.

AKRNEx is used to clear the CGA interrupt pending flop which deasserts the CGA interrupt request signal to the BIIC.

BCI INT L

BCI interrupt is an active low signal which, when asserted causes the BIIC to initiate an interrupt transaction (INTR) on the VAXBI bus. Two other bidirectional signals, DEC ENB H and TESTENB L, provide additional interrupt request lines. On the DMB32 the CGA output BCI INT L is connected to the lowest priority level, VAXBI Level 4 (VAX IPL 14).

BCI DCLO L

BCI DCLO L is an active low signal which, when asserted causes the CGA to be reset, and to assert its RST OUT H signal to reset the remaining node logic.

BCI NXT

BCI NXT is an active low input signal, sourced from the BIIC, which provides information about data flow on the VAXBI during MASTER port transactions.

BCI MAB L

Master abort is an active low output signal which is derived from the VAXBI master sequencer within the CGA. BCI MAB L will be asserted if a retry timeout occurs on the VAXBI or in the special case of a slave write being received during the initiation of a master write cycle.

Table B-4 CGA Signal Descriptions (continued)

BCI I<3:0> H

The BCI information lines, BCI I<3:0> H, pass command information to the slave sequencer (from the BIIC) and from master sequencer (to the BIIC) within the CGA.

Value (Hex)	Description	Mnemonic	CGA Action	
0	RESERVED CODE	–	–	
1	READ	READ	READ	
2	INTERLOCK READ WITH CACHE INTENT	IRCI	READ	
3	READ WITH CACHE INTENT	RCI	READ	
4	WRITE	WRITE	WRITE	
5	WRITE WITH CACHE INTENT	WCI	WRITE	
6	UNLOCK WRITE MASK WITH CACHE INTENT	UWMCI	WRITE MASK	
7	WRITE MASK WITH CACHE INTENT	WMCI	WRITE MASK	
8	INTERRUPT	INTR	BI INTR	*
9	IDENTIFY	IDENT	IDENT	
A	RESERVED CODE	–	–	
B	RESERVED CODE	–	–	
C	STOP	STOP	STOP	*
D	INVALIDATE	INVAL	–	*
E	BROADCAST	BDCST	–	
F	INTERPROCESSOR	IPINTR	–	

* INVAL and INTR commands will be acknowledged by the CGA (if enabled in the BIIC), although INVAL performs no action and INTR only causes the VAXBI interrupt received flag to be set in the BSUM register.

Reception of the STOP command causes the CGA to request mastership of the 68000 bus, by assertion of BUSRQ L. The CGA will output STALL on the BCI RS<1:0> L lines until the input BUSGR L is asserted, completing the action taken on receipt of a STOP command. The BIIC will translate the first STALL response into an ACK response on the VAXBI, since ACK and NOACK are the only legal responses, and will then hold the VAXBI BSY L signal until the BCI STALL code is removed.

During slave read data cycles the BCI I<3:0> H lines pass "Read Status" information to the BIIC:

Status Code (Hex)	Description	Mnemonic
1	READ DATA	RD

Table B-4 CGA Signal Descriptions (continued)

	Read Data is the only read status code supplied by the CGA, since Corrected Read Data and Read Data Substitute have no meaning without ECC checking of RAM contents. BCI I<2> H is ignored (don't care) by the master node decode, within the BIIC. The CGA does not use the read status information, but always outputs the RD code on read data cycles, with BCI I<2> H driven to "L". The VAXBI transaction continues even if a RDS (Read Data Substitute) code is received. Reception of an RDS code is logged in the BSUM register through use of the BIIC event codes. Thus invalid longword(s) will have been written to the DMA File in CASRAM with no information as to which longword(s) is invalid. Therefore the occurrence of this error indicates that the DMA transfer must be repeated. During an assertion of BCI DCLO L the BCI I<3:0> L lines are used for passing "ID" information, as presented at BI ID<3:0> H, to the BIIC. The BCI I<3:0> L lines are also used for passing byte mask information during master write transactions. An assertion of any of BCI I<3:0> L will mask in the associated data byte, permitting it to be written to CASRAM.
BCI SDE L	Slave data enable is an active low input signal only used to enable the read data status, on BCI I<3:0> L, during slave read transactions.
BCI MDE L	Master data enable is an active low input signal used to enable the byte mask information, onto BCI I<3:0> L, during master write transactions. It is also an input to the master sequencer control.
BCI RAK L	Request acknowledge is an active low input signal which indicates that the BIIC has VAXBI bus mastership and is participating in a bus transaction.
VAXBI Interface Signals	
BI ID<3:0> H	The ID lines are active high input signals hard-wired on the VAXBI backplane. These signals are enabled onto BCI I<3:0> L during a BCI DCLO L assertion.
DIAG FAIL L	Diagnostic fail is an active low output signal asserted in conjunction with BCI DCLO L. This signal is asserted upon setting the programmed reset bit within the MAINT register unless the skip self-test bit, within the same register, is also set. This signal will be deasserted by reseting the "DIAG FAIL" bit within the MAINT register.

Table B-4 CGA Signal Descriptions (continued)

BCI TIME L	This signal is the input 20 MHz clock and is connected to BCI TIME L, as connected to the BIIC, from the VAXBI clock receiver chip.
BCI PHASE L	This signal is the input 5 MHz clock and is connected to BCI PHASE L, as connected to the BIIC, from the VAXBI clock receiver chip.
Other Signals	
TEST L	Test is a programmable signal which, following an assertion of BCI DCLO L is an active low input signal. This input is only used during device test, and enables internal nodes to be monitored. The pin may also be programmed to be an output (via the DSCSR register). The special test operation of the device will be disabled when this pin is selected to be an output signal. The output provided will be the internal BCI INT L signal, permitting a selection of interrupt levels. On the DMB32 this signal is used as an output connected to the BIIC signal BCI INT<5> L.
DECODE ENB H	DECODE ENB H is a programmable pin which, following an assertion of BCI DCLO L is an active high input signal. When asserted this input enables the use of internal CGA decode to derive CAS SEL L, and the generation of the acknowledgment signal, DTACK L, for all 68000 address space. If DEC ENB H is deasserted, the signal CAS SEL L is treated as an input signal, requiring external 68000 address decode. In addition, DTACK is only generated in response to CAS SEL L and thus external logic must generate the DTACK for other peripheral and memory devices. This pin can also be programmed to be an output (via the DSCSR register). The internal CGA decode is enabled when this pin is selected as an output signal. The output provided will be the internal BCI INT L signal, permitting a selection of interrupt levels. On the DMB32 this signal is used as an output connected to the BIIC signal BCI INT<6> L.
VDD	+5 V power connections (4 off)
VSS	Ground connections (8 off)

APPENDIX C
GLOSSARY

This glossary defines terms used to describe the VAXBI bus, the BIIC, and the DMB32 option.

Adapter
A node that interfaces other buses, communication lines, or peripheral devices to the VAXBI bus.

Asserted
To be in the "true" state.

Asynchronous
A method of serial data transmission in which data is preceded by a start bit and followed by a stop bit. The receiver provides the intermediate timing to identify the data bits.

Auto-answer
Facility of a modem or terminal to automatically answer a call.

Auto-flow
Automatic flow control. Method by which the DMB32 controls the flow of data by means of special characters within the data stream.

Backward Channel
Channel which transmits in the opposite direction to the usual data flow. Normally used for supervisory or control signals.

Base Address
The address of the CSR.

BCI
VAXBI chip interface; a synchronous interface bus that provides for all communication between the BIIC and the user interface.

BIIC
Backplane interconnect interface chip; a chip that serves as a general purpose interface to the VAXBI bus.

BIIC CSR Space
The first 256 bytes of the 8 Kbyte nodespace, which is allocated to the BIIC's internal registers. See also **Nodespace.**

Bus Adapter
A node that interfaces the VAXBI bus to another bus.

CCITT
Comite Consultatif International de Telephonie et de Telegraphie. An international standards committee for telephone, telegraph and data communications networks.

Dataset
See **Modem**

Deasserted
To be in the "false" state.

Decoded ID
The node ID expressed as a single bit in a 16-bit field.

Device Type
A 16-bit code that identifies the node type. This code is contained in the BIIC's Device Register.

DIL
Dual-In-Line. The term describes ICs and components with two parallel rows of pins.

DMA
Direct Memory Access. Method which allows a bus master to transfer data to/from system memory without using the the host CPU.

DMA adapter
An adapter that directly performs block transfers of data to and from memory.

DUART
Dual Universal Asynchronous Receiver Transmitter. IC used for transmission and reception of serial asynchronous data on two channels.

Duplex
Method of transmitting and receiving on the same channel at the same time.

EIA
Electrical Industries of America. American organization with the same function as CCITT.

EMC
Electro-Magnetic Compatibility. The term denotes compliance with field-strength, susceptibility and static discharge standards.

Encoded ID
The node ID expressed as a 4-bit binary number. The encoded ID is used for the master's ID transmitted during an imbedded ARB cycle.

Even Parity
The parity line is asserted if the number of asserted lines in the data field is an odd number.

FCC
Federal Communications Commission. American organization which regulates and licenses communications equipment.

FIFO
First In First Out. The term describes a register or memory from which the oldest data is removed first.

Floating Address
CSR address assigned to an option which does not have a fixed address allocated. The address is dependent upon other floating address devices connected to the bus.

Floating vector
Interrupt vector assigned to an option which does not have a fixed vector allocated. The vector is dependent upon other floating vector devices connected to the bus.

FRU
Field Replaceable Unit

GO/NOGO
Test or indicator which defines an 'error' or 'no error' condition only.

H
Designates a high-voltage logic level (that is, the logic level closest to Vcc). Contrast with **L**.

IC
Integrated Circuit

Interrupt Vector
In VAX/VMS systems, an unsigned binary number used as an offset into the system control block. The system control block entry pointed to by the VAXBI interrupt vector contains the starting address of an interrupt-handling routine. (The system control block is defined in the *VAX-11 Architecture Reference Manual.*)

I/O
Input/Output

L
Designates a low-voltage logic level (that is, the logic level closest to ground). Contrast with *H*.

LSB
Least Significant Bit

LSI-11 bus
Another name for the Q-bus

Master
The node that gains control of the VAXBI bus and initiates a VAXBI or loopback transaction. See also **Pending Master**.

Master Port
Those BCI signals used to generate VAXBI or loopback transactions.

Master Port Transaction
Any transaction initiated as a result of a master port request.

Microcomputer
An IC which contains a microprocessor and peripheral circuitry such as memory, I/O ports, timers, and **UARTs**.

Modem
The word is a contraction of MOdulator DEModulator. A modem interfaces a terminal to a transmission line. A modem is sometimes called a dataset.

Module
A single VAXBI card that attaches to a single VAXBI connector.

MSB
Most Significant Bit

Multiplexer
A circuit which connects a number of lines to one line.

Node
A VAXBI interface that occupies one of sixteen logical locations on a VAXBI bus. A VAXBI node consists of one or more VAXBI modules.

Node ID
A number that identifies a VAXBI node. The source of the node ID is an ID plug attached to the backplane.

Node Reset
A sequence that causes an individual node to be initialized; initiated by the setting of the Node Reset bit in the VAXBI Control and Status Register.

Nodespace
An 8 Kbyte block of I/O addresses that is allocated to each node. Each node has a unique nodespace based on its node ID.

Null Modem
A cable which allows two terminals which use modem control signals to be connected together directly. Only possible over short distances.

Odd Parity
The parity line is asserted if the number of asserted lines in the data field is an even number.

PCB
Printed Circuit Board

Power-down/Power-up Sequence
The sequencing of the BI AC LO L and BI DC LO L lines upon the loss and restoration of power to a VAXBI system. See also **System Reset.**

Protocol
Set of rules which define the control and flow of data in a communications system.

PSTN
Public Switched Telephone Network

Q-bus
Global term for a specific DIGITAL bus on which the address and data are multiplexed.

Q22, Q18 and Q16
Terms used to define 22-, 18- and 16-bit address versions of Q-bus.

RAM
Random Access Memory

RESERVED code
A code reserved for use by DIGITAL.

RESERVED field
A field reserved for use by DIGITAL. The node driving the bus must ensure that all VAXBI lines in the RESERVED field are deasserted. Nodes receiving VAXBI data must ignore RESERVED field information. This requirement provides for adding functionality to future VAXBI node designs without affecting compatibility with present designs. Example: The BI D<31:0> L and BI I<3:0> L lines during the third cycle of an INTR transaction are RESERVED fields.

Reset Module
In a VAXBI system, the logic that monitors the BI RESET L line and any battery backup voltages and that initiates the system reset sequence.

Resetting Node
The node that asserts the BI RESET L line.

RFI
Radio Frequency Interference

ROM
Read Only Memory

Slave
A node that responds to a transaction initiated by a node that has gained control of the VAXBI bus (the master).

Slave Port
Those BCI signals used to respond to VAXBI and loopback transactions.

Split-speed
Facility of a data communications channel which can transmit and receive at different data rates at the same time.

System Reset
An emulation of the power-down/power-up sequence that causes all nodes to initialize themselves; initiated by the assertion of the BI RESET L line.

Transaction
The execution of a VAXBI command. The term "transaction" includes both VAXBI and loopback transactions.

UART
Universal Asynchronous Receiver Transmitter. **IC** used for transmission and reception of serial asynchronous data on a channel.

UNDEFINED Field
A field that must be ignored by the receiving node(s). There are no restrictions on the data pattern for the node driving the VAXBI bus. Example: The BI D<31:0> L and BI I<3:0> L lines during read STALL data cycles and vector STALL data cycles are UNDEFINED fields.

User Interface
All node logic exclusive of the BIIC.

User Interface CSR Space
That portion of each nodespace allocated for user interface registers. The user interface CSR space is the 8 Kbyte nodespace minus the lowest 256 bytes, which comprise the BIIC CSR space.

VAX Interrupt Priority Level (IPL)
In VAX/VMS systems, a number between 0 and 31 that indicates the priority level of an interrupt with 31 being the highest priority. When a processor is executing at a particular level, it accepts only interrupts at a higher level, and on acceptance starts executing at that higher level.

VAXBI Primary Interface
The portion of a node that provides the electrical connection to the VAXBI signal lines and implements the VAXBI protocol; for example, the BIIC.

VAXBI Request
A request for a VAXBI transaction from the **Master Port** interface that is asserted on the BCI RQ<1:0> L lines.

VAXBI System
All VAXBI cages, VAXBI modules, reset modules, and power supplies that are required to operate a VAXBI bus. A VAXBI system can be a subsystem of a larger computer system.

VAXBI Transaction
A transaction in which information is transmitted on the VAXBI signal lines.

Window Space
A 256 Kbyte block of I/O addresses allocated to each node based on node **ID** and used by bus adapters to map VAXBI transactions to other buses.

XOFF
Control code (23 octal) used to disable a transmitter. Special hardware or software is needed for this function.

XON
Control code (21 octal) used to enable a transmitter which has been disabled by an **XOFF** code.

Digital Equipment Corporation • Bedford, MA 01730