EK-DB86X-TD-002

VAX 8600/8650 SBIA Technical Description

Prepared by Educational Services
of
Digital Equipment Corporation

1st Edition, May 1985
2nd Edition, January 1986

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

Printed in U.S.A.

The manuscript for this book was created on a VAX-11/780 system and, via a translation program, was automatically typeset by Digital's DECset Integrated Publishing System. The book was produced by Educational Services Development and Publishing in Marlboro, MA.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

FIGURES

FIGURES (Cont)

TABLES

TABLES (Cont)

CHAPTER 1
INTRODUCTION

NOTE

For simplicity, the word "system" is used throughout this manual and applies to both the VAX 8600 and VAX 8650 systems unless otherwise specified.

1.1 MANUAL SCOPE AND RELATED DOCUMENTS

This manual, written as a training and field resource, is a comprehensive description of the VAX 8600/8650 SBIA. The manual is written on three levels – general, functional, and detailed. Table 1-1 lists related hardware documentation.

1.2 GENERAL DESCRIPTION

A VAX 8600/8650 may contain two synchronous backplane interconnect adapters (SBIAs), SBIA 0 and SBIA 1. Each SBIA provides an interface between the MBox, via the ABus, and a synchronous backplane interconnect (SBI) that will do the following conversions.

Table 1-1 Related Hardware Manuals

Title	DIGITAL P.N.
Technical Descriptions	
VAX 8600/8650 Console Technical Description	EK-KA86C-TD
VAX 8600/8650 EBox Technical Description	EK-KA86E-TD
VAX 8600/8650 System Power Technical Description	EK-KA86P-TD
VAX 8600/8650 FBox Technical Description	EK-FP86X-TD
VAX 8600/8650 IBox Technical Description	EK-KA86I-TD
VAX 8600/8650 MBox/Memory Technical Description	EK-KA86M-TD
VAX 8600/8650 System Clocks Technical Description	EK-KA86K-TD
VAX 8600/8650 EMM Technical Description	EK-KA86V-TD
VAX 8600/8650 System Description and Processor Overview	EK-KA86S-TD
User's Guides	
VAX 8600/8650 System Diagnostics User's Guide*	EK-KA86D-UG
VAX 8600/8650 System Hardware User's Guide	EK-8600H-UG
VAX 8600/8650 System Maintenance Guide*	EK-86XV1-MG
VAX 8600/8650 System Installation Manual	EK-8600I-IN
VAX 8600/8650 System Fault Isolation Manual*	EK-8600S-MM

* For Internal Use Only

The SBIA enables the CPU, via the MBox, to read or write SBI nexus registers and SBIA registers (see Figure 1-1). In response to the CPU's reading a vector register, the SBIA initiates an interrupt summary read to determine interrupt vectors. In a like manner, in response to the CPU's writing the unjam register, the SBIA hardware carries out the SBI unjam sequence.

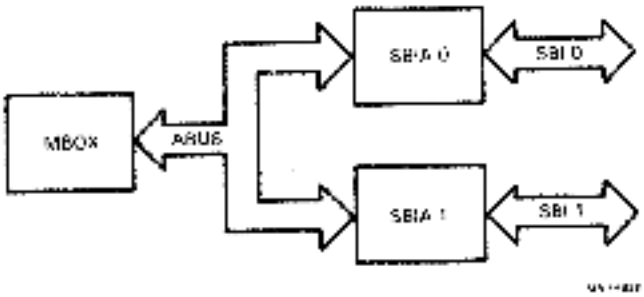


Figure 1-1 ABus/SBIA Interconnect

When an SBI nexus initiates a DMA write, after being set up by the CPU, the SBIA transfers the command/address and writes data to the MBox. For a DMA read, the SBIA transfers the command/address to the MBox, accepts the read data from the MBox, and then transfers the read data to the SBI.

1.3 PHYSICAL DESCRIPTION

The SBIAs are located on the five-slot ABus backpanel (Figure 1-2). There are two slots for each SBIA and one slot for an SBI/ABus terminator module. The ABus backpanel is specific for SBIAs; no other ABus interfaces can be used on this backpanel. A VAX 8600/8650 includes one SBIA. The DIGITAL part number for the second SBIA is DB86-AA.

1.3.1 Module Partitioning

The ABus interface module, SBA, an L0203, is installed in slot 3 for SBIA 0 and slot 5 for SBIA 1. The SBI interface module, SBS, an L0202, is installed in slot 2 for SBIA 0 and slot 4 for SBIA 1. The SBI/ABus terminator module, the STM module, an L0224, is installed in slot 1. The terminator provides termination for the ABus and one end of both SBIs. The SBI clock signals are not terminated by the STM, but on the far end of the SBI. The far end SBI terminator is an SBT for SBI 1 and an M9040 for SBI 0, unless SBI 0 is connected to an expansion cabinet. In that case, SBI 0 will also be terminated with an SBT.

1.3.1.1 SBA Module – The SBA module has the following characteristics.

1. Contains both ECL and TTL logic
2. Has a CPU/SBI state machine, a 1K × 12-bit PROM
3. Has ECL to TTL and TTL to ECL logic translators
4. Contains a 16 × 40-bit register file, the primary interface between the SBIA and the ABus.

1.3.1.2 SBS Module – The SBS module has the following characteristics.

1. With the exception of the clock translators, completely TTL logic
2. Contains the DC101 SBI priority arbitration chips
3. Has the following ROMs/PROMs:
   a. A 256 × 4-bit ROM for ABus commands
   b. A 32 × 8-bit PROM for SBIA error vectors
   c. A 256 × 8-bit PROM for zero fill
   d. Three 256 × 8-bit PROMs for register read/write control
   e. A 1K × 4-bit PROM used for address decoding.

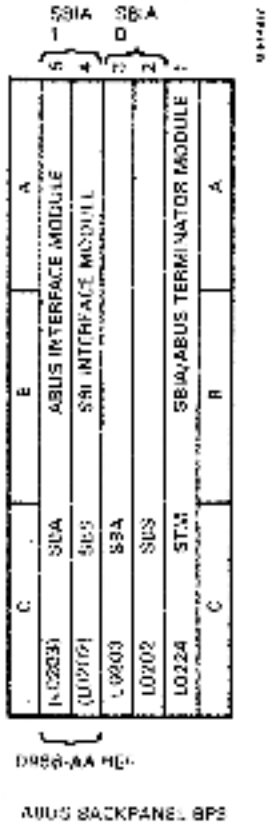


Figure 1-2 ABus Backpanel

1.4 SBIA DATA TRANSFERS

The SBIA is involved in all data transfers between the CPU and SBI nexus. A brief description of each type of transfer follows.

1.4.1 CPU Write

The CPU initiates a CPU write by sending a command/address, and write data longword over the ABus to the SBIA. They conform to ABus protocol. For an overview of ABus protocol and a list of ABus signals, see Appendix A. The command/address and write data are loaded into the SBIA register file. The address portion of the command/address is the target location for the write data. The SBIA removes the command/address and then the write data from the register file. If the address specifies an SBIA register, the data is written into the SBIA register.

If the address is for an SBI nexus, the SBIA modifies the command/address so that it conforms to SBI protocol (see Appendix B). When the SBIA can get control of the SBI at the CPU transfer request level (set at TR02, see Appendix C) it will transmit the modified command/address on the SBI. The write data is transmitted on the SBI on the following SBI cycle.

1.4.2 CPU Read

The CPU also initiates a CPU read by sending a command/address over the ABus to the SBIA. It is loaded into the register file as with the CPU write. Again, the address portion of the command/address indicates the target location for the read. The SBIA removes the command/address from the register file, and, if the address is for an SBIA register, gates the contents of the addressed register to the register file.

If the address indicates an SBI nexus, the SBIA modifies the command/address so that it conforms to SBI protocol. As with the CPU write, when the SBIA can get control of the SBI at the CPU transfer request level, it transmits the modified command/address on the SBI.

The addressed nexus recognizes the address and returns the requested read data on the SBI. The SBIA then takes the read data from the SBI, reformats it so that it conforms to ABus protocol, and loads it into the register file. When the read data is in the register file, either from an SBIA register or an SBI nexus, the SBIA informs the MBox that the read data is available.

The MBox takes the read data from the register file and transfers it to the EBox.

1.4.3 DMA Transfers

For DMA transfers, the CPU must provide the nexus with necessary information, such as the starting address of the data transfer and number of bytes to transfer. Interrupts must be enabled if the CPU is to be interrupted at completion of the transfer. The CPU writes nexus registers to prepare for the transfer.

1.4.3.1 DMA Write – Once the nexus has been programmed for a DMA write, it arbitrates for the SBI and transfers the command/address, followed by the write data longword or longwords for a quadword transfer. The SBIA modifies the command/address so that it conforms to ABus protocol and stores it in the register file. The write data is placed in the register file following the command/address. When the command/address and write data are in the register file, the SBIA asserts an IOA request to the MBox for service. The MBox reads the command/address and then the write data, storing the data in cache.

1.4.3.2 DMA Read – As for the DMA write, the nexus arbitrates for the SBI and transfers the command/address. The SBIA reformats the command/address, stores it in the register file, and then asserts an IOA request to the MBox. The MBox reads the command/address, obtains the requested data from cache/memory, and places the read data in the SBIA register file. The SBIA removes the read data from the register file and arbitrates for control of the SBI at the transfer request level assigned to the DMA, TR01. When the SBIA gains control of the SBI, it transfers the read data to nexus over the SBI.

1.5 PHYSICAL MEMORY ADDRESSES

The system physical memory allocation is shown in Figure 1-3. The memory allocation for each IO adapter is shown in Figure 1-4.

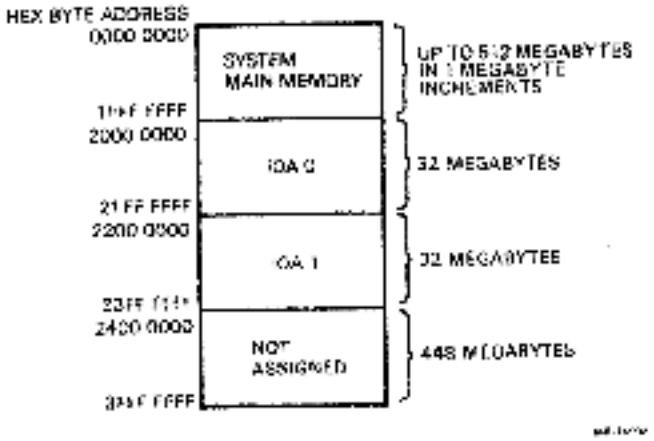


Figure 1-3 Physical Memory Address Allocation

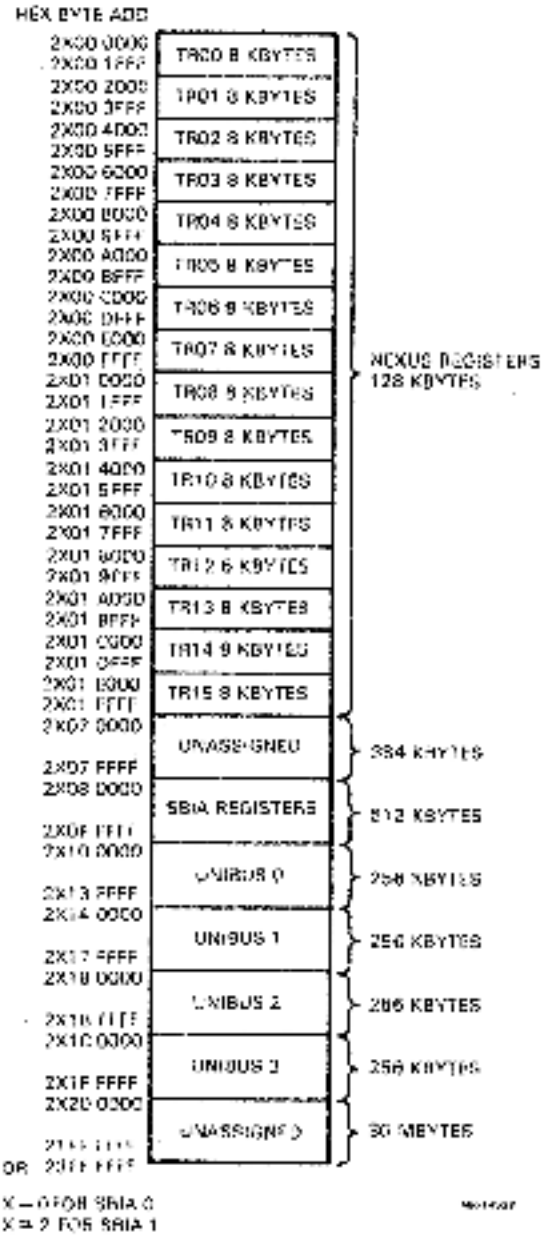


Figure 1-4 I/O Adapter Physical Address Allocation

The SBIA register addresses are shown in Table 1-2. Both the hex byte and hex longword addresses are shown. The MBox, before placing the physical address on the ABus, shifts the address right by two bits to provide a longword address. The SBIA decodes the hex longword address. Any address returning to the MBox from the SBIA is shifted left by two bits as it enters the MBox from the ABus. The SBIA registers are described in detail in Chapter 3.

Table 1-2 SBIA Register Addresses*

Hex Byte Register Name	Hex Longword Address	Address
Configuration register	2X08 0000	8Y2 0000
Control and status register	2X08 0004	8Y2 0001
Error summary register	2X08 0008	8Y2 0002
Diagnostic control register	2X08 000C	8Y2 0003
DMAI command/address register	2X08 0010	8Y2 0004
DMAI ID register	2X08 0014	8Y2 0005
DMAA command/address register	2X08 0018	8Y2 0006
DMAA ID register	2X08 001C	8Y2 0007
DMAB command/address register	2X08 0020	8Y2 0008
DMAB ID register	2X08 0024	8Y2 0009
DMAC command/address register	2X08 0028	8Y2 000A
DMAC ID register	2X08 002C	8Y2 000B
SBI silo	2X08 0030	8Y2 000C
SBI error register	2X08 0034	8Y2 000D
SBI timeout address register	2X08 0038	8Y2 000E
SBI fault/status register	2X08 003C	8Y2 000F
SBI silo comparator	2X08 0040	8Y2 0010
SBI maintenance register	2X08 0044	8Y2 0011
SBI unjam register	2X08 0048	8Y2 0012
SBI quadclear register	2X08 004C	8Y2 0013
Vector registers	2X08 0080	8Y2 0020
.	.	.
.	.	.
.	.	.
Vector registers	2X08 00B8	8Y2 002E

* For SBIA 0: X = 0 and Y = 0
For SBIA 1: X = 2 and Y = 8

1.6 SBIA ERROR DETECTION

The SBIA detects the following types of errors.

1. SBI parity errors
2. Parity errors on ABus data being transferred to the SBI, when it is removed from the register file
3. Timeouts
   a. The SBIA is unable to gain control of the SBI in 102.4 μs.
   b. An SBI nexus does not respond to a command/address.
   c. If, after the SBI nexus acknowledges the command/address for a CPU read, the SBIA does not receive read data in 102.4 μs.

4. SBI protocol errors

   a. The SBIA receives read data when a read is not pending.

   b. A command/address has indicated a write function, but there is no write data on the next SBI cycle.

   c. An SBI nexus attempts an interlock write without a previous interlock read.

   d. More than one SBI nexus transmits on the SBI at the same time.

The errors are described in Chapters 2 and 3. First, the errors are described in the data transfer description, at the time the error might be detected. The errors are also described, in detail, with the register bit descriptions, for the register bit that the particular error would set.

1.7 SBIA REGISTERS

The SBIA has 35 registers in the I/O address space (see Table 2-1). They include control registers, status registers, maintenance registers, error registers, and vector registers. The vector registers occupy addresses 2008 0080 (2208 0080, SBIA 1) to 2008 00B8 (2208 00B8, SBIA 1). The remaining registers are in addresses 2008 0000 (2208 0000, SBIA 1) to 2008 004C (2208 004C, SBIA 1). Each register is described in detail in Chapter 3.

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 CHAPTER OVERVIEW

This chapter covers the SBIA at the block diagram level. Each block is treated individually; the relationship between blocks is described and, if a particular block is used in a data transfer, that aspect is included. Chapter 3 contains detailed descriptions of the data transfers. The SBIA block diagram (Figure 2-1) provides the reference for most of the overview. The numbers in the blocks refer to to print set sheet designation. Flowcharts are used to describe the CPU read/write transactions and the possible error conditions that may arise during these transactions.

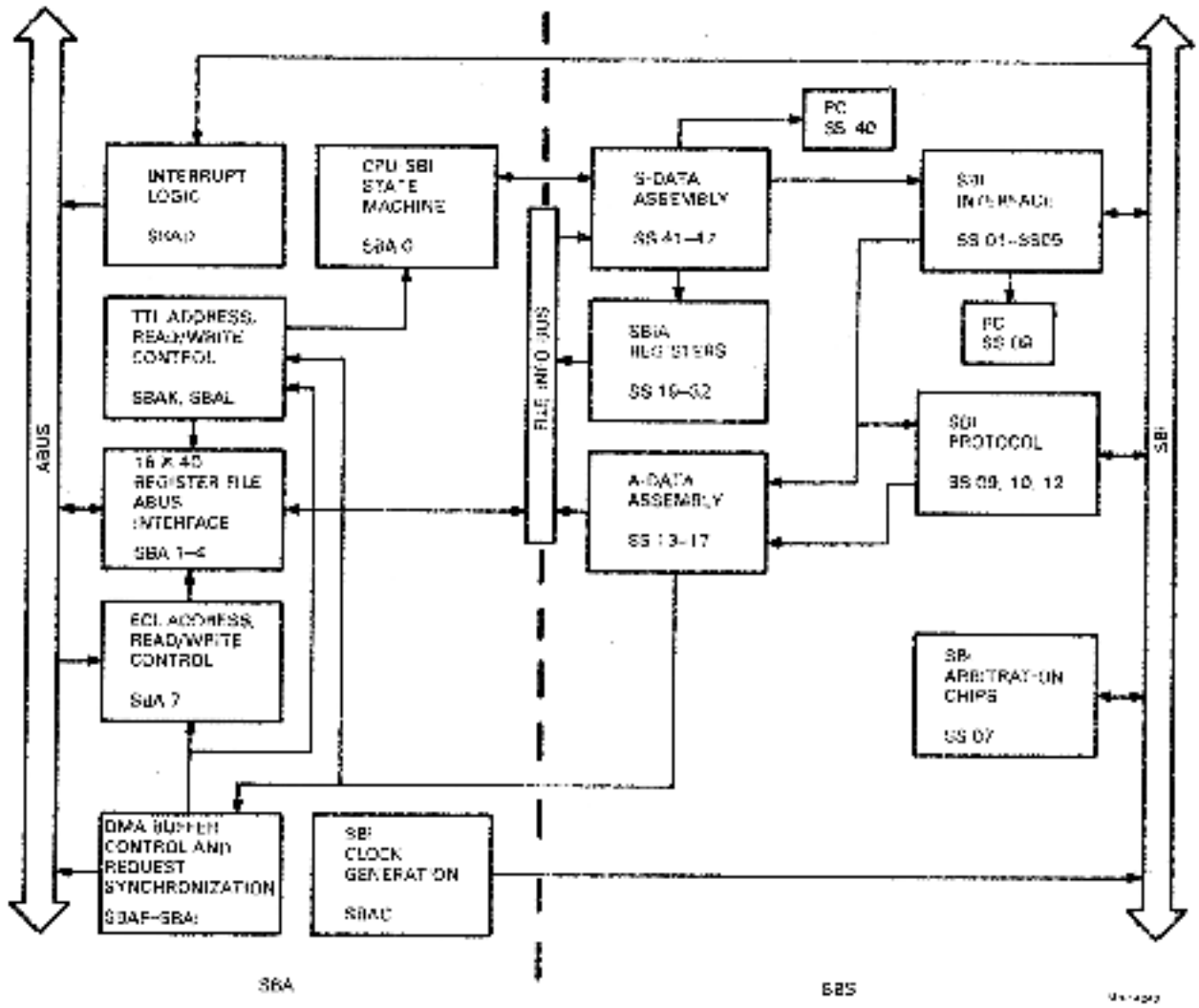


Figure 2-1 SBIA Basic Block Diagram

2.2 SBIA BASIC BLOCK DIAGRAM

Figure 2-1, the SBIA block diagram, is logically divided by the file info bus, which is the backpanel interconnect between the SBA and SBS modules. Everything to the left of the file info bus is on the SBA module. Everything to the right of the file info bus is on the SBS module.

The SBS print set pages are designated as SS00–SS48 and do not include any hex numbers. The SBA module print set is numbered from SBA00–SBAT. The alphabetic numbering starts as if the print set is numbered in hex, but continues through the alphabet after F.

2.2.1 CPU/SBI State Machine

The CPU/SBI state machine is a 1K × 12-bit PROM used to control CPU read/write transactions to SBI nexus registers and vector registers. The state machine is not used for CPU reads/writes to SBIA registers other than the vector registers. Its major function is to control the S-data assembly, during the transfer of register file data, from the ABus to the SBI. During this transfer, it controls the modification of ABus protocol to conform to SBI protocol. It also controls the monitoring of SBI bits to check for error conditions.

The CPU/SBI state machine microword contains an even parity bit, and parity is checked on every microword, even if the state machine is not involved with the transfer. Other outputs include four bits that are used to generate the next address (next state), the bits used to control the S-data assembly, and a bit to allow the SBIA to hold the SBI for an additional cycle for CPU writes to nexus registers.

The state machine does not leave the idle state for a CPU read/write unless the command/address and write data for a CPU write have been removed from the register file without parity errors. When the state machine leaves the idle state, it steps through a series of microinstructions to allow the read/write transaction to take place.

The state machine monitors the SBI confirmation bits by branching on these two bits, and uses these branch conditions to detect error or timeout conditions. The error conditions are covered in the flowcharts.

2.2.2 Register File

The most significant part of the ABus interface is the register file, a 16 × 40-bit dual port register file. This register file is capable of being addressed, written, or read simultaneously on each of its two ports. One port is on the ABus, and the other is on the file info bus, the backpanel bus that connects the SBA module to the SBS module. The register file contains both ECL and TTL logic, with the ABus side being ECL logic and the file info bus side TTL logic.

All information exchange over the ABus must go through the register file, and there are locations reserved for the various transactions. Two locations are reserved for CPU transactions, one location for the command/address word and one location for the read or write data. Two locations are also reserved for interlock read transactions, with one location for the command/address word and one for the read data. There is a block of locations reserved for each of the other DMA transactions – transaction buffers A (DMAA), B (DMAB), and C (DMAC). Each contains three locations, one for the command/address and the other two for the read or write data longwords. (Paragraph 3.1 covers the register file in detail, and Figure 3-1 shows the register file organization.)

If the MBox is writing information into the register file or reading data from the register file, the MBox sets up the register file address. If the SBIA is removing information from the register file for transfer to the SBI, or placing modified SBI information into the register file, the SBIA will control the address. There is a read/write address on the ABus side and a read/write address on the file info bus side.

On the ABus side, the register file can be read or written only if commanded by the MBox. On the TTL side, the register file is read every SBI T0 (see Appendix B, Figure B-10). It is written, when enabled, at SBI T2. Even though it is read every SBI cycle, the contents of the addressed location are not necessarily valid information. The SBIA does not process information read on the TTL side unless it is valid information that the MBox has loaded.

2.2.3 S-Data Assembly

When the register file is read, the data is transferred over the file info bus to the S-data assembly. If the data is to be transmitted on the SBI, the S-data assembly checks for proper parity, reformats each piece of information to insure that it will conform to SBI protocol, and passes the reformatted information to the SBI transceivers.

The S-data assembly contains parity checkers to insure that ABus data has reached the S-data assembly without errors. It also contains parity generation circuitry that will change the ABus odd parity to SBI even parity.

Every SBI T0T1, the contents of the register file, as addressed by the TTL address, will be read and transferred to the S-data assembly. If the MBox has not loaded pertinent information into the register file, the information just read will not get transferred to the SBI. However, a parity check is made in any case. At this point, the data is in ABus format, so the parity is checked accordingly. (See Appendix A for a review of ABus protocol.)

The S-data assembly uses multiplexers to reformat the data so that it will conform to SBI protocol. The multiplexers, and thus the format, are controlled by the state machine.

The S-data assembly also recalculates parity to provide even parity as required on the SBI.

2.2.4 SBI Interface

The SBI interface consists of 8646 4-bit transceiver latches. They are clocked to transmit every SBI T0, if enabled. The receive latches are opened every SBI T2 and closed at ~SBI T2. Each chip generates odd parity over the four bits it receives. These parity outputs are used by the SBI protocol logic to check for even SBI parity.

2.2.5 SBI Protocol

The SBI protocol logic combines the parity outputs from each of the SBI transceiver chips to check for SBI parity. Parity is checked over the following bits.

1. BUS SBI B<31:00>
2. BUS SBI M<03:00>
3. BUS SBI P<01:00>
4. BUS SBI TAG<02:00>
5. BUS SBI ID<04:00>.

Parity is not checked over the following bits.

1. BUS SBI TR<15:00>
2. BUS SBI CONF<01:00>
3. BUS SBI FAULT
4. BUS SBI INTLK
5. BUS SBI MP<02,01>
6. BUS SBI SPARE <01:00>.

The protocol logic monitors for other SBI faults as follows.

1. Unexpected read fault: The SBIA receives read data with the CPU ID but is not expecting read data.

2. Write sequence fault: The command/address indicated a write function, and the information received on the next SBI cycle has good parity but it is not write data.

3. Multiple transmitter fault: The SBIA just transmitted on the SBI, but the ID received does not compare to the ID transmitted.

4. Interlock sequence fault: The SBIA receives a command/address for an interlock write, but the interlock flip-flop is not set (there was no interlock read).

2.2.6 A-Data Assembly

The A-data assembly is primarily responsible for the reformatting of SBI data to insure that it conforms to ABus protocol. It receives the SBI information from the SBI transceivers, modifies it as necessary, and passes it over the file info bus to the register file for temporary storage. Parity will be recalculated to provide odd parity.

2.2.7 Clock Logic

There are two major parts of the clock logic, the ABus clock logic and the SBI clock generation circuitry.

2.2.7.1 ABus Clock Logic – The SBIA uses two clock signals from the clock module, CLK SBA[N] CLOCK5 141 B and CLK SBA[N] CLOCK5 141 D to generate the following ECL clocks.

1. SBA6 CLK6 PHASE T0B
2. SBA6 CLK6 PHASE T1B
3. SBA6 CLK6 PHASE T3B
4. SBA6 CLK6 PHASE T2D
5. SBA6 CLK3 T3D.

These five clocks provide timing for portions of the ABus interface logic (see Figure 2-2).

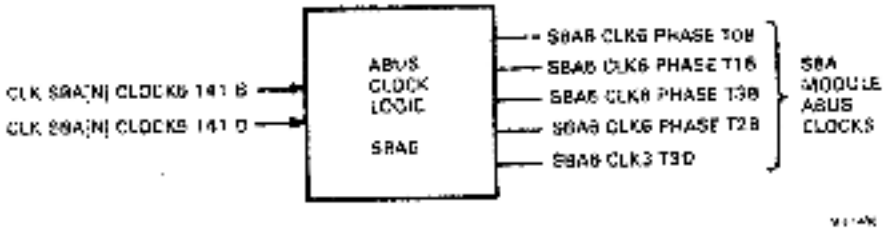


Figure 2-2 ABus Clock Logic

2.2.7.2 SBI Clock Generation – The SBI clock generation circuitry uses a 40 MHz crystal oscillator to provide the six SBI ECL clocks (see Appendix B). The six clocks used by the SBI nexus, including the SBIA, to derive the four 50 ns time states, T0, T1, T2, and T3 are:

1. BUS SBI TP H
2. BUS SBI TP L
3. BUS SBI PCLK H
4. BUS SBI PCLK L
5. BUS SBI PDCLK H
6. BUS SBI PDCLK L

The SBIA decodes these six SBI clocks to provide various ECL timing terms that are converted to TTL for use on the SBA module. Three of the ECL timing terms are converted to TTL on the SBS module for use on that module (see Figure 2-3).

The external clock input is for manufacturing use only. An input for an external clock, backpanel pin A06, must be enabled by grounding pin A01. Both of these pins are in slot 3 (SBIA 0) or slot 5 (SBIA 1).

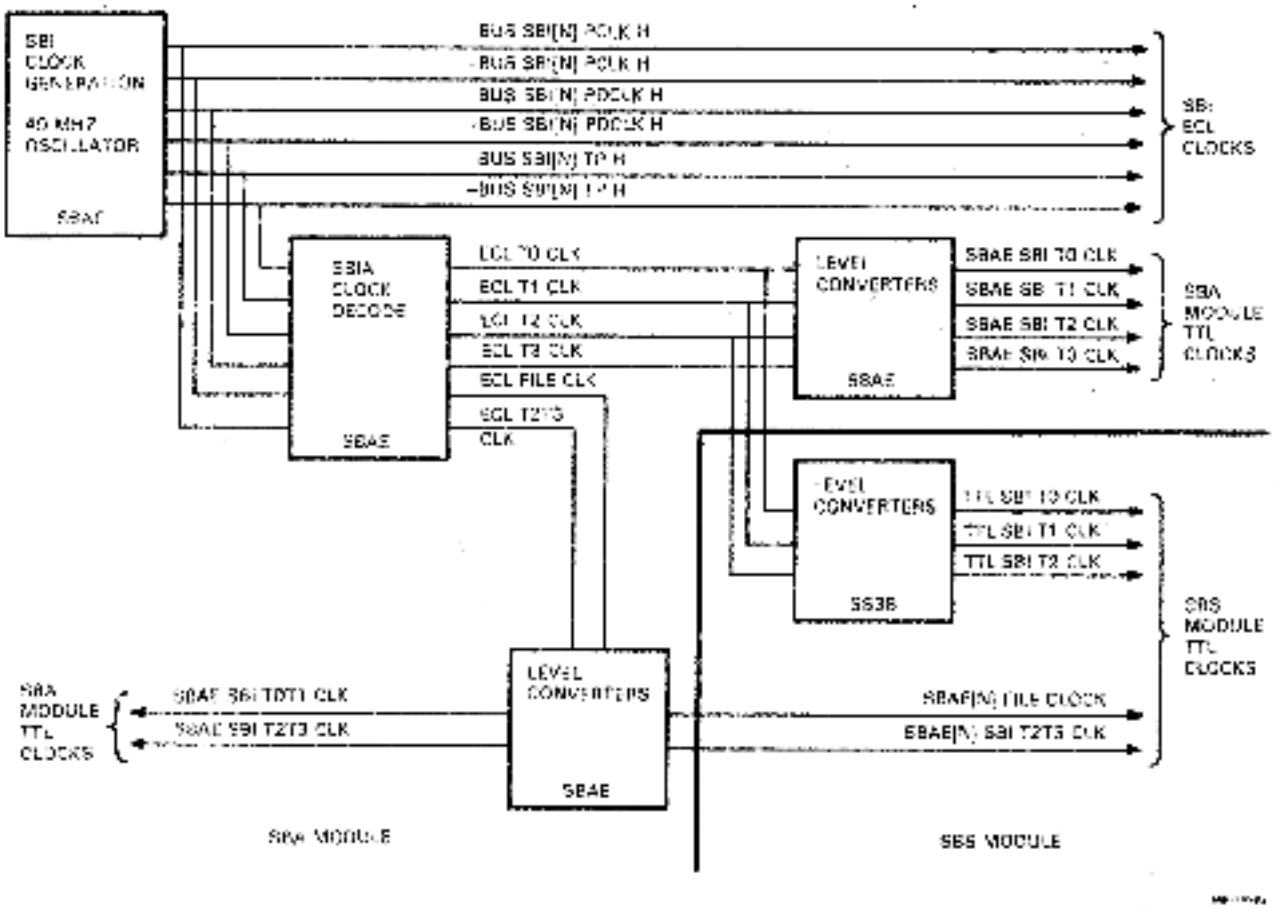


Figure 2-3 SBI Clock Generation

2.2.8 DMA Buffer Control and Request Synchronization

The register file contains locations dedicated to specific data transfer transactions, with four separate groups of locations reserved for the DMA transactions. DMA transaction buffers DMAA, DMAB, and DMAC are used for DMA transactions other than interlock reads. Transaction buffer DMAI is used only for interlock reads.

When a DMA transfer is initiated by an SBI nexus, the SBIA must insure that the information is loaded into a transaction buffer in the register file and that it will not be disturbed until the transfer is complete. The DMA buffer control and request synchronization logic determines the transaction buffer to be used next, provides this information to the address control logic, and, when there is a DMA queued in the register file, asserts the DMA request to the MBox. A DMA transaction is considered to be queued when the command/address and write data for a DMA write have been loaded into the register file. When the DMA transaction is complete, the transaction buffer is free for another DMA request.

2.2.9 ECL Address and Read/Write Control

The ECL address and read/write control logic generates the read and write address and enables reading or writing the register file from the ABus (see Paragraph 3.2.1).

The method of address generation depends on the type of data transaction. For DMA transfers, part of the address is generated by the SBIA logic and part by the MBox. For CPU reads, the entire address is generated by the MBox.

The register file can be written from the ABus only if the I/O adapter is selected, and then only if the MBox enables writing the register file. Likewise, to read the register file, the I/O adapter must be selected. The contents of the addressed location are gated to the ABus if the MBox does not enable writing the register file.

2.2.10 TTL Address and Read/Write Control

This block of logic is responsible for controlling the register file address on the SBI side of the register file. The address depends on the type of data transfer operation in progress. For DMA transactions, the DMA buffer control and request synchronization logic control the address. For a CPU read/write, the address is controlled by ABus signals.

2.2.11 Interrupt Logic

The interrupt logic receives the SBI and other interrupt requests, and, if interrupts are enabled (control and status register bit 31 set), establishes priority of the interrupts, and sends an encoding of the highest priority interrupt to the EBox on the IPR return lines (ABus IPR RETURN <04:00>).

The interrupt priorities are as follows. (See Table 3-9 for a list of interrupt priority levels and vectors.)

1. Fail
2. SBI fault or SBIA error
3. Alert
4. Silo compare
5. SBI request <07:04>.

2.2.12 SBIA Registers

The SBIA has 35 registers in the I/O address space -- 20 control, status, error, and maintenance registers, and 15 interrupt vector registers. The first group lies in the local address range of 2008 0000 to 2008 004C (SBIA 0) or 2208 0000 to 2208 004C (SBIA 1). The vector registers are addresses 2008 0080 to 2008 00B8 (SBIA 0) and 2208 0080 to 2208 00B8 (SBIA 1). Access to any other location within the local address space causes an error.

Any data to, or from, the SBIA registers must pass through the register file and over the ABus. The register contents pass through tri-state devices to the file info bus to prevent the possibility of register data and SBI traffic colliding on the file info bus. When the registers are read, if there are numerous bits that are not used, special logic (the zero fill logic) provides logic zeros for most unused bits. (The registers and their bit definitions are defined in detail in Paragraph 3.14, SBIA Registers).

2.2.13 SBI Arbitration Chips

Two DC101 priority arbitration chips allow the SBIA access to the SBI. One, used for DMA transactions, transfers the read data to the SBI; the other gives the SBIA control of the SBI for CPU read or write transactions. The DC101s and SBI arbitration is described in detail in Appendix C.

2.3 CPU TRANSACTION FLOWCHARTS

The following flowcharts describe the PROM flow for CPU transactions. Each flowchart is followed by a brief description. A detailed description may be found in Chapter 3, and the PROM code may be found in microfiche.

2.3.1 Starting the State Machine

The first flowchart, Figure 2-4, is used to determine which of the other four flowcharts to follow. When the VAX 8600/8650 is first powered-up, the ABus is not enabled (the console provides a continuous INIT to the SBIA). When the console enables the ABus by setting console register MCSR <01>, CL ABUS ENABLE is asserted to remove the INIT. The CPU/SBI state machine is forced to the idle state by an INIT.

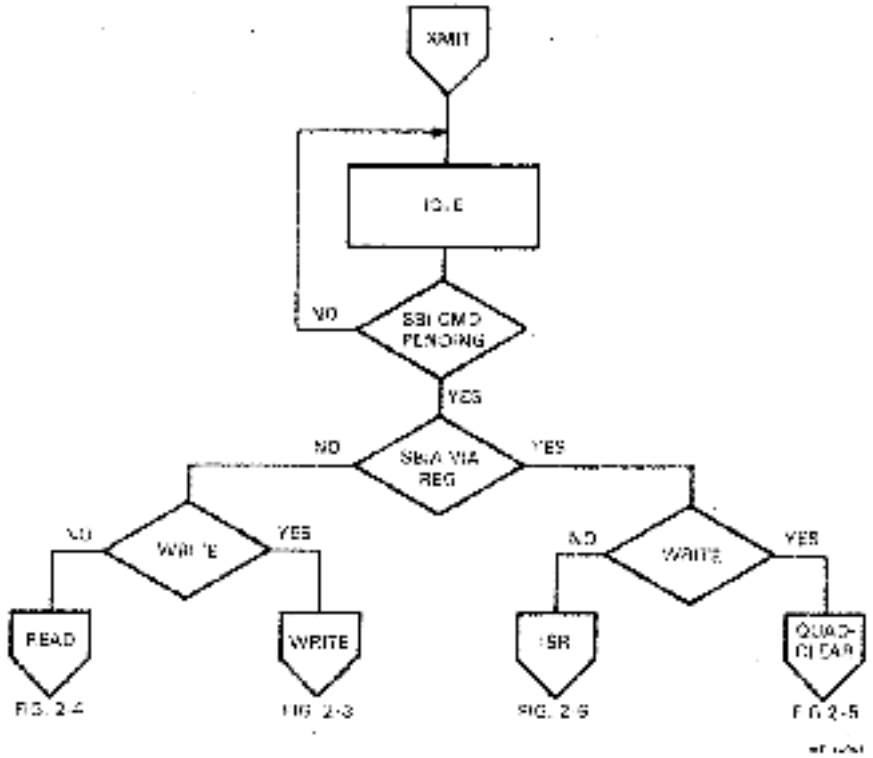


Figure 2-4 Starting the State Machine

Three functions are involved in getting the CPU-SBI state machine from the idle state to a read, write, ISR, or quadclear state: SBI CMD PENDING, SBIA VIA REG, and WRITE.

1. SBI CMD Pending: If there is no SBI command pending, the SBI state machine remains in the idle state. If an SBI command is pending, other control signals are monitored to determine what path to follow. An SBI command is pending under the following conditions.

   a. The MBox has loaded the register file with a command/address for a CPU read or write

   b. When the command/address is removed from the register file and is transferred to the S-data assembly, there are no address or control parity errors.

   c. For a CPU write, there are no data or control parity errors on the write data.

   d. The address has to be a valid SBI address with the SBIA enabled to access the SBI, or the address must be a valid SBIA vector register address.

   If there are any error conditions, the state machine will not leave the idle state. ABUS CPU BUF ERROR will be asserted to inform the MBox of the error, and the timeout address register, the SBI error register, and the error summary register <31:26> are latched to hold the error conditions.

2. SBI VIA REG: SBI VIA REG, one of the outputs of a PROM addressed by the command/address, is asserted if the CPU is writing the quadclear register or reading the vector registers for IPR <17:14>. The latter initiates an interrupt summary read.

3. WRITE: WRITE will be asserted if bit three of the CPU command in the command/address, is set.

Table 2-1 shows which operation is performed and which flowchart to follow, for the various combinations of WRITE and SBIA VIA REG.

Table 2-1 SBIA Operation Performed

Write	SBI VIA REG	Operation Performed	Figure No.
0	0	Read	2-6
0	1	Interrupt summary read	2-8
1	0	Write	2-5
1	1	Quadclear	2-7

2.3.2 CPU Write

When the state machine leaves the idle state it starts a timeout counter and goes to the ARB WAIT state (Figure 2-5). The SBIA will request control of the SBI by asserting the transfer request. The transfer request level is selected by connecting backpanel pins to backpanel pins at ground potential (see Appendix C). The state machine remains in the ARB WAIT state until ARB OK is received or a timeout occurs.

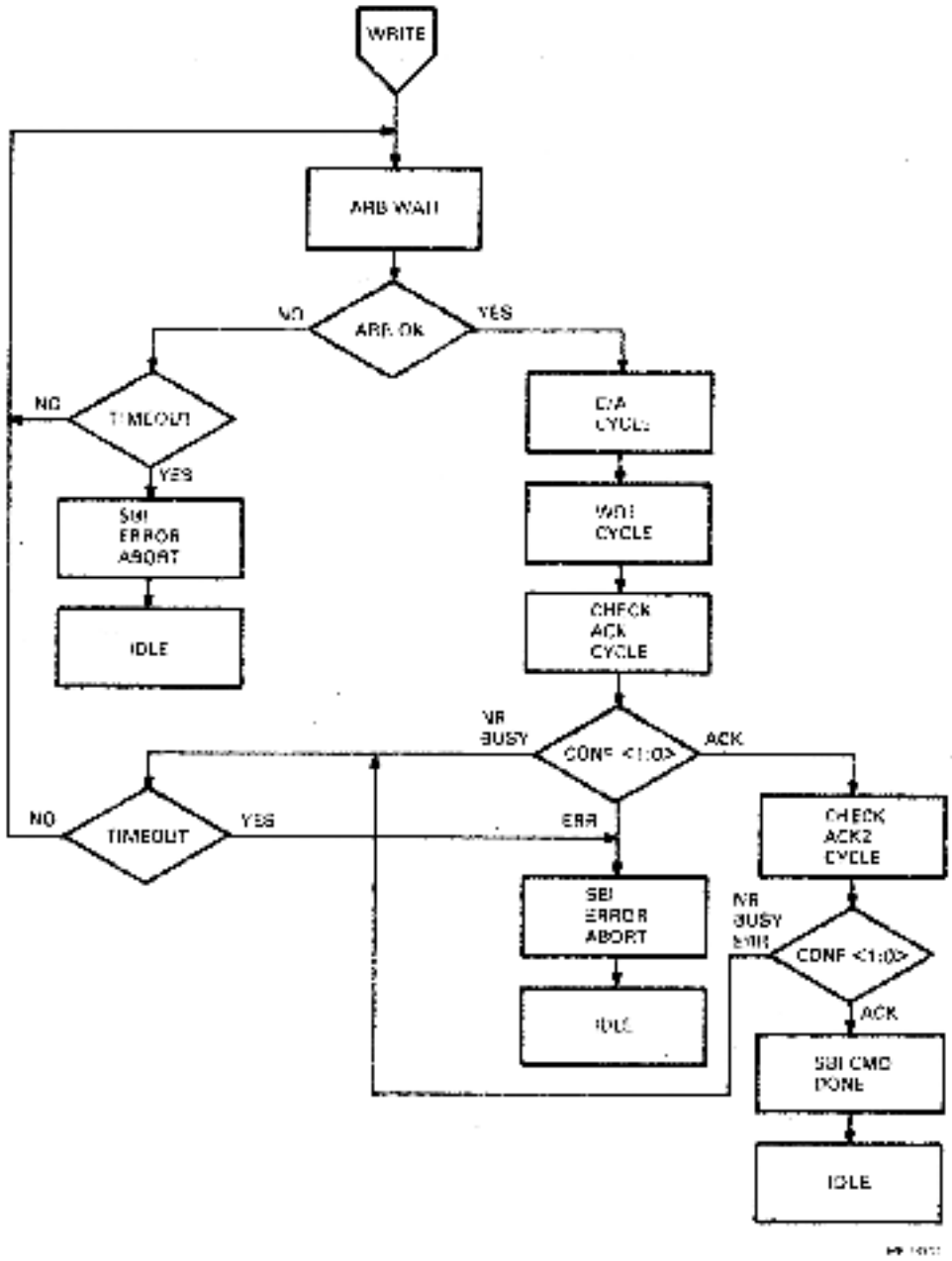


Figure 2-5 State Machine Flowchart: CPU Write

If ARB OK is not received within 512 SBI cycles (102.4 μs), the state machine goes to the error abort state and the command is aborted. ABUS CPU BUF ERROR is asserted to inform the MBox of the error condition, and the SBI error register, the timeout address register, and the error summary register <31:26> are latched to hold the error information. From the abort state, the state machine returns to the idle state. ABUS CPU BUF DONE is asserted simultaneously with ABUS CPU BUF ERROR. ABUS CPU BUF DONE tells the MBox that the SBIA has finished the transaction.

When the SBIA has gained control of the SBI, it transmits the command/address and then the write data on the SBI. The state machine then monitors for an acknowledge from the SBI nexus.

If there is no response, or there is a busy response, the SBIA returns to the ARB WAIT state to arbitrate for control of the SBI again. When the SBIA gains control of the SBI, the SBIA retransmits the command/address followed by the write data.

If there is no response, or the the nexus continues to send a busy response and the timeout counter expires after 512 SBI cycles (102.4 μs), the state machine enters the error abort state and reports the error with ABUS CPU BUF ERROR. The state machine then returns to the idle state.

If the SBIA receives an error confirmation, the state machine enters the error abort state, reports the error, and then returns to the idle state.

If the SBIA receives an acknowledge for the command/address, the state machine then looks for an acknowledge for the write data. If the write data response is busy or error or there is no response, the state machine assumes an intermittent error and returns to the ARB WAIT state; it then retransmits the command/address and write data.

When the SBIA receives an acknowledge for the write data, before the expiration of the timer, the state machine goes to the command done state. It asserts ABUS CPU BUF DONE to inform the MBox that the command has been carried out and then returns to the idle state.

2.3.3 CPU Read

When the state machine leaves the idle state, it goes to the ARB WAIT state, asserts the transfer request, and waits for ARB OK (see Figure 2-6). If the SBIA does not gain control of the SBI within 512 SBI cycles, a timeout condition exists, the state machine goes to the abort state, asserts ABUS CPU BUF ERROR to report the error, and returns to the idle state.

When the SBIA gains control of the SBI, it transmits the command/address on the SBI. It waits one cycle and then checks for an acknowledge for the command/address. If there is no response, or the response is a busy response, the state machine returns to the ARB WAIT state, waits for control of the SBI once more, and then retransmits the command/address.

If the SBIA receives an error response for the command/address, or the timeout counter expires before receiving an acknowledgment, the state machine goes to the error abort state and asserts ABUS CPU BUF ERROR to report the error.

If an acknowledge is received, the SBIA restarts the 512 SBI cycle timeout counter. The SBIA waits for the nexus to transmit the read data on the SBI. If the SBIA does not receive the read data within 512 SBI cycles, the state machine goes to the error abort state to report the error.

If the read data is received within the allotted time, the SBIA reformats the data and loads it into the register file. The state machine then goes to the SBI CMD DONE state and asserts ABUS CPU BUF DONE. The MBox assumes that the transaction was carried out successfully when ABUS CPU BUF ERROR is not asserted. The state machine then returns to the idle state.

The MBox transfers the read data from the register file to the EBox to complete the CPU read transaction.

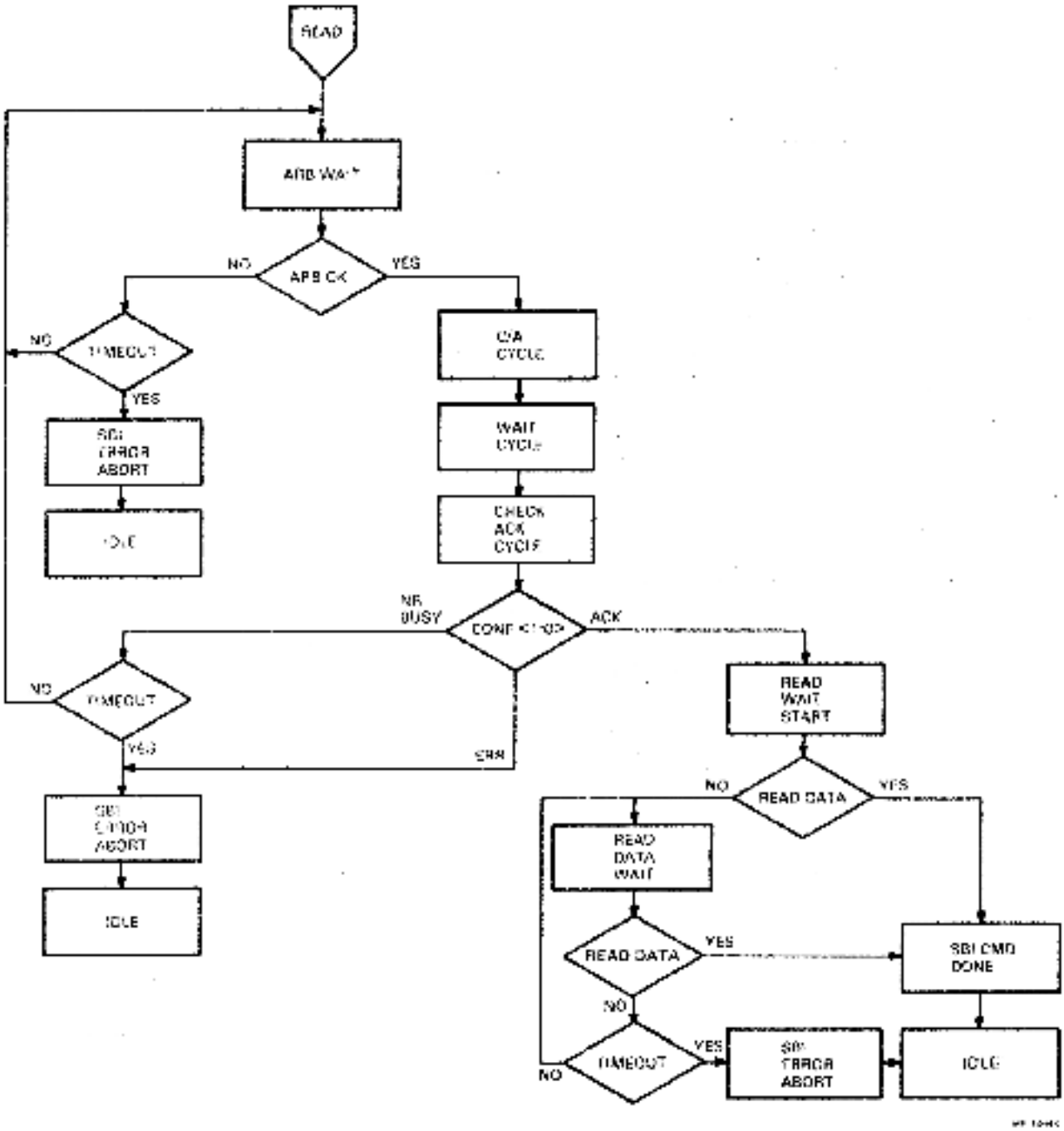


Figure 2-6 State Machine Flowchart: CPU Read

2.3.4 Quadclear

The purpose of the quadclear operation is to clear ECC errors in SBI memories (see Figure 2-7). For the VAX 8600/8650 system, microdiagnostics uses the quadclear operation extensively to perform DMA quadword loopback transfers.

The CPU writes the quadclear register. The command/address specifies a CPU write to the quadclear register. The write data contains the SBI command for an extended write masked (1011) and the address of the quadword to be cleared. The ABus write data is used to form the SBI command/address, and the SBI write data is two all-zero longwords. The flowchart for quadclear is much the same as for CPU write.

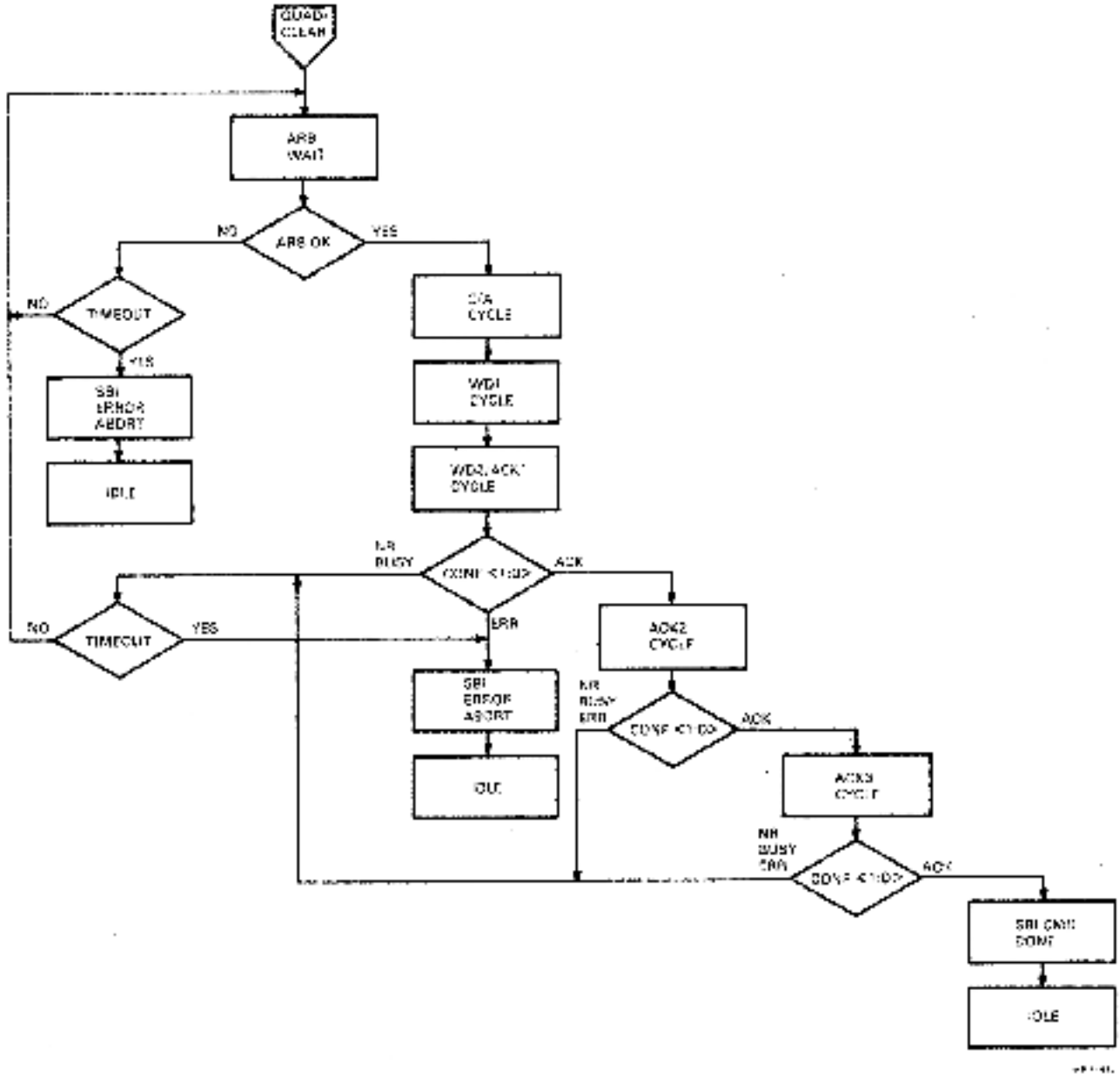


Figure 2-7 State Machine Flowchart: Quadclear

The state machine goes to the ARB WAIT state to await the availability of the SBI. As with the CPU read or write, the SBIA must gain control of the SBI within 512 SBI cycles, or a timeout condition will exist. If the SBIA does not gain control of the SBI within 512 SBI cycles, the state machine goes to the error abort state and asserts ABUS CPU BUF ERROR to report the error.

When the SBIA gains control of the SBI, it transmits the command/address developed from the ABus write data, followed by two all-zero write data cycles. On the second write data SBI cycle, the SBIA looks for an acknowledgment for the command/address from the nexus.

If there is no response, or the confirmation is a busy response, the state machine returns to the ARB WAIT state, waits for control of the SBI, and retransmits the command/address and write data. If the SBIA continues to receive busy responses or receives no response for 512 SBI cycles, the state machine goes to the error abort state to report the error. Also, if an error response is received for the command/address, the state machine goes to the error abort state.

When the state machine has seen an acknowledgment for the command/address, it looks for the acknowledgments for the write data cycles. An acknowledgment is expected for each write data cycle. If the SBIA receives an error or busy confirmation, or receives no response for either data cycle, the state machine will go to the ARB WAIT cycle to retransmit the command/address and write data. Again, it is assumed that the command/address was received properly, so there must have been an intermittent error.

When the state machine has detected the proper number of acknowledgements (three), it goes to the SBI CMD DONE state, asserts ABUS CPU BUF DONE, and then returns to the idle state.

2.3.5 Interrupt Summary Read

When the CPU is going to handle an interrupt, it reads the SBIA vector register corresponding to the IPR level of the active interrupt (see Figure 2-8). The SBIA initiates an interrupt summary read (ISR) in response to the CPU reading the SBIA register for an IPR of 14, 15, 16, or 17. (See Table 3-9 for a comprehensive list of IPR levels, addresses, or vectors.)

The command/address from the CPU indicates a read, and the address specifies the vector register.

During the SBI command/address cycle, the SBIA transmits data with a bit set according to the IPR level being handled – B<04> for IPR 14, B<05> for IPR 15, B<06> for IPR 16, and B<07> for IPR 17.

Any nexus interrupting at that IPR level is expected to transmit an interrupt summary response two cycles after receiving the ISR. The interrupt summary response will have two bits set, the first bit being the same as the nexus TR level, and the other bit equal to the nexus TR level plus 16. The extra bit is not used by the SBIA, but is needed to insure even parity on the SBI. For instance, if the interrupting nexus has a TR of 4, the nexus would set B<20> and B<04>. More than one nexus could be responding to the IPR, and no nexus can be expected to have control of the parity bits. Each nexus will transmit two bits, so parity remains even no matter the number of nexus responses to the ISR.

During an ISR the SBIA does not check the confirmation bits for an acknowledgment, error, or busy. The interrupt summary return is checked only for parity errors.

The state machine enters the ARB WAIT state until the SBIA can gain control of the SBI. If the SBI is not available within 512 SBI cycles, the state machine goes to the error abort state to report the error.

When the SBI is available, the state machine goes to the ISR command/address cycle to transmit the interrupt summary read word. The next state waits just one SBI cycle.

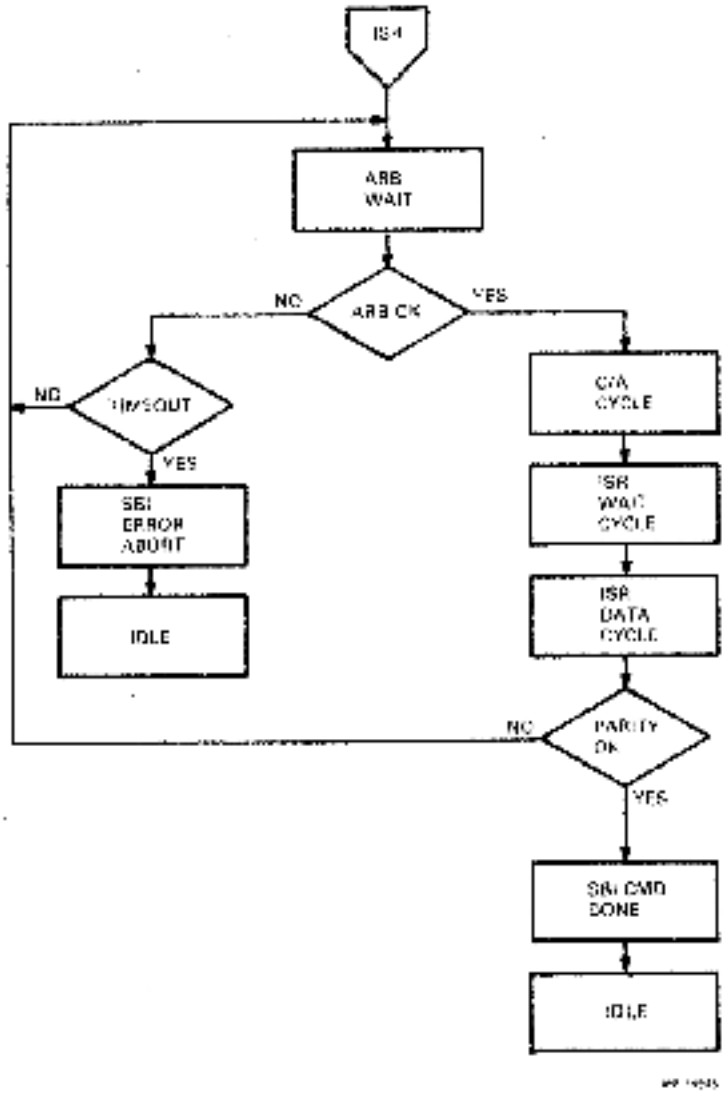


Figure 2-8 State Machine Flowchart: ISR

The state machine expects the interrupt summary return on the next SBI cycle and performs a parity check on the received data. If there is a parity error, the state machine returns to the ARB WAIT state for SBI availability, and retransmits the interrupt summary read word.

If there is no parity error, the SBIA generates the vector from the interrupt summary response and loads the vector into the register file (see Figure 3-16). The state machine goes to the command done state to inform the MBox that the transaction is complete and that the vector is in the register file. The state machine then returns to the idle state.

2.4 SBIA TRANSFERS NOT USING STATE MACHINE

2.4.1 SBIA Register Writes or Reads

When the CPU writes or reads an SBIA register, the state machine does not leave the idle state and therefore is not involved with the transfer. The MBox loads the register file with a command/address with a command indicating a write or a read, and an address specifying an SBIA register. For a CPU write to an SBIA register, the MBox loads the register file with the write data immediately following the command/address. For a read, the register contents are transferred to the register file. The MBox then reads the register file.

When the command/address and write data for a CPU write are transferred from the register file to the S-data assembly, parity is checked. If there are no parity errors, and the address is a valid SBIA register address, the write data is written into the addressed register. If there are no errors, the MBox is informed that the operation is complete with the assertion of ABUS CPU BUF DONE.

If there is a parity error over the command/address or the write data, or the address is an invalid SBIA register address, the SBIA asserts ABUS CPU BUF ERROR to inform the MBox of the error condition. As with the preceding errors, the SBI error register, the timeout address register, and the error summary register <31:26> are latched to hold the error information.

For a CPU read register, if there are no parity errors and the address is a valid register address, the contents of the addressed register are written into the register file. ABUS CPU BUF DONE informs the MBox that the register data is in the register file. The MBox reads the data from the register file and transfers it to the EBox.

2.4.2 Unjam

An unjam, issued to the SBI to clear a hung system, is initiated by the CPU writing the SBIA unjam register, address 2008 0048 or 2208 0048. The MBox loads the register file with a command/address indicating a write to the unjam register. Because it is a register write, the MBox also transfers write data to the register file. Although the write data is not used, it is needed to provide good parity on the write data to initiate the unjam sequence.

If there are no parity errors on the command/address and the write data, and the address is a valid SBIA register address, the decoding of the unjam register address initiates the hardware unjam sequencer, which issues SBI HOLD for 16 SBI cycles, SBI HOLD and SBI UNJAM for 16 SBI cycles, and SBI HOLD for another 16 SBI cycles.

If there is a parity error over the command/address or the write data, or the address is an invalid SBIA register address, the SBIA will assert ABUS CPU BUF ERROR to inform the MBox of the error condition. The SBI error register, the timeout address register, and the error summary register <31:26> are latched to hold the error information.

2.4.3 DMA Transactions

The CPU initiates DMA transactions by a series of CPU writes to the appropriate nexus registers. When properly programmed and enabled, the nexus starts the data transfer by placing a command/address on the SBI. With DMA write, the command/address is followed by the write data. With a DMA read, the nexus expects to receive the read data over the SBI later.

DMA transactions may be interlocked or noninterlocked. A DMA interlock read must precede an interlock write. An interlock write occurring without a preceding interlock read is an interlock sequence fault, and SBI FAULT is asserted to interrupt the CPU.

An interlock timeout exists if the interlock write does not occur within 512 SBI cycles (102.4 μs) of the interlock read. An interlock timeout asserts a DMA1 ERROR, which will interrupt the CPU.

2.4.3.1 DMA Write – The SBIA, upon receiving the command/address from the SBI nexus, reformats the command/address and writes it into the register file if the following conditions are met.

1. There are no SBI parity errors over the command/address.
2. The address is for a memory location; it is less than the contents of the configuration register.
3. The tag is for a command/address (011).
4. The function is a valid SBI function.
5. The SBIA is not expecting write data, which means that the previous SBI cycle was not a command/address for a DMA write.
6. There is an available transaction buffer in the register file.

The following SBI cycle(s) should contain the write data to be written into memory. The write data will be written into the register file if the following conditions are met.

1. The command/address indicated a write function or extended write function.
2. The tag indicates write data (101).
3. There are no parity errors on the write data.

If the SBIA detects an SBI parity error over the command/address or write data, the error condition is latched in the SBI fault/status register. The information is not written into the register file, and the EBox is notified of the error condition by an interrupt request.

When the command/address and write data have been written into the register file, the SBIA asserts ABUS IOA REQUEST [N] to request MBox attention. When ABUS IOA REQUEST [N] is asserted, ABUS WR CMD will be asserted to inform the MBox that it is a DMA write. ABUS MSKED CMD may also be asserted if it is a masked write. The assertion of the latter two signals allows the MBox microcode to branch early. The microcode does not have to wait until it gets the command/address to know what is expected.

The MBox reads the command/address and write data and stores the write data in cache/memory. If the MBox detects no errors it informs the SBIA that the transaction is complete by asserting MCC ABUS DMA DONE. If the MBox detects an error, it asserts MCC ABUS DMA ERROR at the same time it asserts MCC ABUS DMA DONE. The SBIA latches the error condition in the error summary register and the appropriate DMA command/address and DMA ID register. The SBIA asserts an interrupt request to inform the EBox of the DMA error.

2.4.3.2 DMA Read – The DMA read is initiated by the nexus in the same manner as the DMA write; it transmits the command/address on the SBI. The nexus monitors the SBI for the returning read data.

The SBIA, upon receiving the command/address, reformats it and writes it into the register file if the conditions listed in Paragraph 2.4.3.1 for a DMA write are met. If there is an SBI parity error, the error will be reported the same as for a DMA write.

When the command/address has been written into the register file, the SBIA asserts ABUS IOA REQUEST [N] to request MBox attention. Because ABUS WR CMD is not asserted, the MBox microcode assumes it is a DMA read. ABUS MSKED CMD (not asserted) is used to allow the MBox microcode to branch before it sees the command/address.

The MBox reads the command/address and obtains the read data from cache/memory. It will then write the read data into the SBIA register file. Then the MBox informs the SBIA that the read data is in the register file by the assertion of MCC ABUS DMA DONE. If the MBox detected an error, MCC ABUS DMA ERROR would also have been asserted.

The SBIA removes the read data from the register file and transfers it to the S-data assembly for parity check, reformatting, and transfer to the SBI. When the SBIA can get control of the SBI at TR1, the read data is reformatted to conform to the SBI format and transmitted on the SBI.

If the MBox detected a DMA error, the SBIA latches an error bit in the error summary register and latches the appropriate DMA command/address register and DMA ID register.

If a parity error is detected when the SBIA transfers the read data from the register file to the S-data assembly, the error condition is latched in the error summary register and the appropriate DMA command/address and DMA ID register.

The SBI nexus initiating the DMA read is informed of the error condition because the SBIA forces the mask bits to 0010, read data substitute. No other SBI nexus detects the error because the read data is transmitted on the SBI with even P0 and P1.

CHAPTER 3
DETAILED DESCRIPTION

3.1 REGISTER FILE ORGANIZATION

The SBIA register file contains 16 locations, each 40 bits wide. It is divided in five areas called transaction buffers. One transaction buffer, the CPU transaction buffer, is reserved for CPU reads and writes of SBIA or SBI nexus registers. The DMAI transaction buffer is reserved for DMA interlock reads. The remaining three transaction buffers, DMAA, DMAB, and DMAC, are reserved for DMA read and write transfers.

The information contained in the register file conforms to ABus protocol. Therefore, reformatting is necessary before placing the information on the SBI, or before placing SBI information into the register file. (See Appendix A, ABus Protocol, and Appendix B, SBI Protocol.)

3.1.1 CPU Transaction Buffer

The CPU transaction buffer consists of two locations. Location 2 is reserved for the command/address, and location 3 is reserved for the read or write data longword. Refer to Figure 3-1.

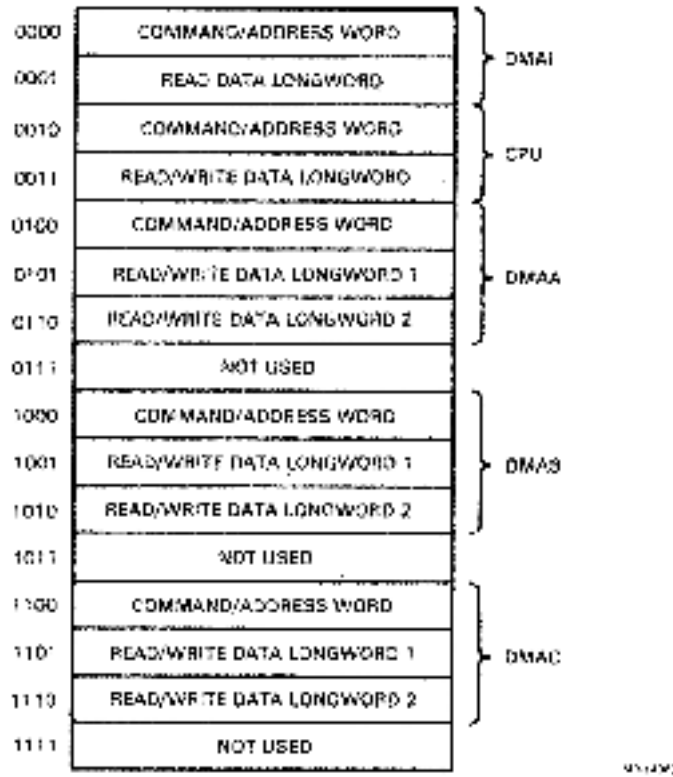


Figure 3-1 SBIA Register File

3.1.2 DMA1 Transaction Buffer

Register file location 0 is reserved for the command/address for a DMA read interlock. Location 1 is reserved for the read interlock data longword. The interlock is released with a write interlock, through the DMAA, DMAB, or DMAC transaction buffers.

3.1.3 DMAA, DMAB, or DMAC Transaction Buffers

Each of the three DMA transaction buffers, DMAA, DMAB, or DMAC, consists of three locations within the register file. DMAA uses locations 4, 5, and 6; DMAB uses locations 8, 9, and A; and DMAC uses locations C, D, and E. The command/address is stored in the first of the three locations, while the other two locations are reserved for the read or write data longwords. The upper two bits of the four-bit address is the same for each of the three locations for a DMA transaction buffer. DMAA = 01XX, DMAB = 10XX, and DMAC = 11XX. When a particular DMA transaction buffer is selected, the upper two bits can be held constant until that buffer is no longer in use. The lower bits can be manipulated to address the required location within the transaction buffer. (This process becomes more apparent as the CPU read, CPU write, and DMA transactions are investigated.)

3.2 CPU WRITE SBI NEXUS REGISTER

A CPU read or write cannot be initiated by the MBox if the SBIA has any DMA IOA requests pending (SB ABUS IOA REQUEST [N]).

The MBox creates the command/address word based upon the EBox request and address and transfers the command/address and write data to the SBIA. The SBIA transfers the command/address and data to the SBI and monitors the SBI for the proper number of acknowledgments.

The description of the CPU write nexus register begins with the MBox loading the command/address and write data into the register file and concludes with ABUS CPU BUF DONE after the write data has been received and acknowledged by the SBI device.

3.2.1 Loading CPU Command/Address

See Figure 3-2. The MBox selects the I/O adapter with MCC ABUS IOA SELECT [N]. The address latches are loaded by MCC ABUS ADDRESS CTRL<01:00> 1. – 00. The address is generated by these conditions:

1. The upper two bits of the register file ECL address are forced to 00 by the assertion of MCC ABUS CPU BUF SEL H.

2. The lower two bits of the register file ECL address are determined by MCC ABUS MBOX OUT H and MCC ABUS CPU BUF SEL H according to Table 3-1.

MCC ABUS MBOX OUT H enables the command/address to be written into the ECL portion of the register file as addressed by ECL FILE ADR <03:00>, which is 0010 for the command/address.

3.2.2 Loading CPU Write Data

The next ABus cycle keeps the IOA selected, but MCC ABUS ADDRESS CTRL <01:00> 1. will be changed to 10, which increments the address from 0010 to 0011. Because MCC ABUS MBOX OUT H is kept asserted, the write data will be written into the register file.

The register file now contains the command/address and write data, and the SBIA is responsible for passing this information to the SBI. The MBox deselects the IOA and waits for the SBIA to send ABUS CPU BUF DONE. The MBox services any DMA IOA requests that occur before the assertion of ABUS CPU BUF DONE.

Table 3-1 ECL File Adress <01:00>

MCC ABUS CPU BUF SEL H	MCC ABUS MBOX OUT H	ECL FILE ADR <01:00>
0	0	0 0
0	1	0 0
1	0	1 1
1	1	1 0

3.2.3 Addressing and Unloading the Register File for TTL Read

The register file must be addressed on the TTL side to enable reading the command/address for a CPU read/write, write data for a CPU write, or the read data for a DMA read. DMA transfers have highest priority, so a DMA request would set up the file read address for that particular DMA request. FILE READ ADDR <03:02> is controlled by the DMA requests and FILE READ ADDR <01:00> is controlled by the presence or absence of a DMA transfer request (SEND DMA TR), according to Table 3-2.

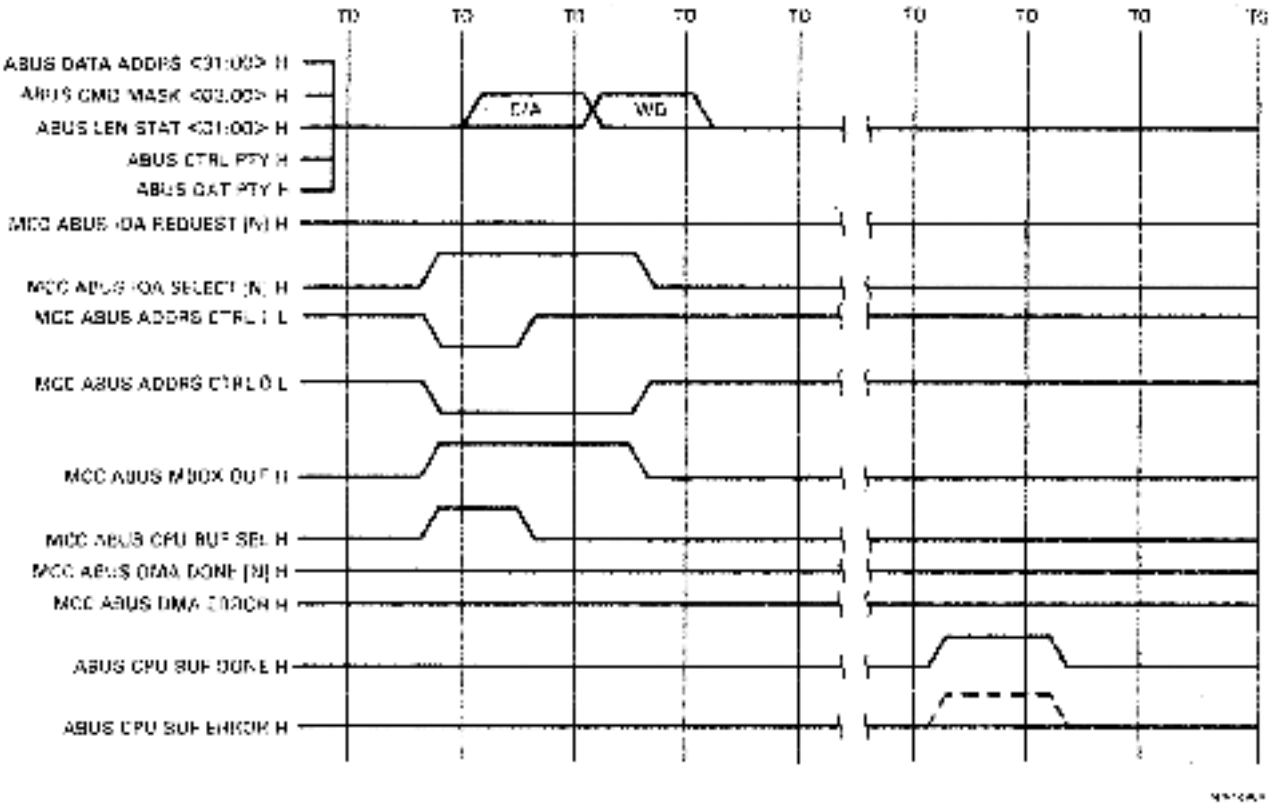


Figure 3-2 CPU Write: ABus Protocol

Table 3-2 Register File TTL Read Address

Transaction Buffer	FILE READ ADDR 03	02	01	00
DMAA	0	1	X	X
DMAB	1	0	X	X
DMAC	1	1	X	X
DMAI, CPU	0	0	X	X
DMA TR	X	X	0	1
-DMA TR	X	X	1	0

For example, the SBIA has received a quadword read command and the read longwords are in the register file in transaction buffer A (DMAA) waiting to be transferred to the SBI. When DMA transaction buffer A (DMAA) asserts SEND DMA TR, FILE READ ADDR <03:00> is loaded with 0101, the location for the first read data longword.

Likewise, for an interlock read, when DMAI asserts SEND DMA TR, to transfer the read longword to the SBI, FILE READ ADDR <03:00> will be 0001.

For a CPU write, the command/address is transferred to the SBI, followed by the write data. If no DMA transaction buffer has a command in progress for a DMA read, this fact, and the lack of a DMA TR, generate a FILE READ ADDR of 0010, the address of the command/address.

In any of these foregoing cases, the address will be loaded at T0 and can be incremented at the next T0, but cannot otherwise be changed until the next CPU read, CPU write, or DMA read is initiated. The address is held during that time when a valid file read is in progress.

3.2.3.1 Valid File Read – VALID FILE READ is used to hold a file read address when the address is valid, and to insure that only valid data is loaded into the command/address and write data latches. It is asserted under the following conditions:

1. The use of the SBI has been requested with a DMA TR for a DMA read.
2. The TTL file read address indicates that a CPU command/address is to be read from the register file.
3. The TTL file read address indicates that the CPU write data is to be read from the register file.
4. The second read data longword is to be read from the register file for a DMA quadword read.

The register file is read every SBI cycle, but the information is not always valid. VALID FILE READ will insure that the command/address latch is loaded only with a valid command/address, and that the write data latch is not loaded with stale data.

3.2.3.2 Double Unload – When two sequential locations in the register file have to be read, as for a DMA quadword read or a CPU write, FILE READ ADDR <03:00> must be incremented. DOUBLE UNLOAD is asserted in these cases, and when DMA ARB OK is received for a DMA quadword read, or FILE READ ADDR <03:00> = 0010 for a CPU write, FILE READ ADDR will be incremented on the next T0.

3.2.4 File Data Latch

The file data latch is loaded from the addressed register file location every SBI cycle, and latched at T1.5. Parity is checked over the address/data bits and control bits. The parity check will be used to enable the CPU-SBI state machine to leave the idle state (see Paragraph 3.2.7). The file data latch provides inputs to the command/address latch, write data latch, parity checkers, and multiplexers in the S-data assembly.

3.2.5 Loading the Command/Address Latch

The command/address latch is loaded from the file data latch if the register file address is 0010, the address for the CPU command/address, and VALID FILE READ is asserted. It is latched at –T1 and will not change until the next CPU command/address is transferred from the register file to the file data latch and then to the command/address latch.

3.2.6 Loading the Write Data Latch

The write data latch is loaded from the file data latch if the register file address is 0011, the address for the write data, and VALID FILE READ is asserted. It too is latched at –T1 and will not change until the next time CPU write data is transferred from the register file to the file data latch, then the write data latch.

3.2.7 Starting the CPU-SBI State Machine

The CPU-SBI state machine loops in the idle state, waiting for CMD GO. CMD GO causes the state machine to leave the idle state and execute a series of microinstructions that transfer the contents of the command/address and write data latches to the SBI.

CMD GO will be asserted if these conditions exist.

1. There was no control parity error or A/D parity error over the command/address word.
2. The address is an ISR register address or an SBI address and the SBI is enabled.
3. It is a CPU write and there is no control parity error or A/D parity error over the write word.

If any error conditions exist, the SBIA immediately sends ABUS CPU BUF ERROR to the MBox, which will generate an EBox microtrap.

When the state machine leaves the idle state, a binary counter is allowed to count every other SBI T0. If the counter counts 512 SBI cycles before completing the CPU write (102.4 μs), an SBI timeout occurs. This timer is disabled at the normal termination of a CPU write when the state machine returns to the idle state.

The microinstructions executed by the state machine upon leaving the idle state depend on the physical address and whether the CPU command is for a read or write.

3.2.8 CPU ARB WAIT

The SBIA must gain control of the SBI at the CPU TR level before it can place the command/address and write data on the SBI. CPU ARB OK is asserted by the DC10I priority arbitration chips to grant the SBIA control of the SBI for the CPU write. (See Appendix C, SBI Arbitration.)

Because the operation is a CPU write to an SBI nexus register, the SBIA must place the command/address and write data on the SBI in consecutive SBI cycles. Therefore, the SBIA must have control of the SBI for two SBI cycles. When CPU ARB OK is asserted, the state machine asserts CPU HOLD to cause the DC101s to assert TR0 for one cycle, holding the SBI for an additional cycle.

The state machine waits in the ARB WAIT cycle until CPU ARB OK is received. If CPU ARB OK is not received within 102.4 μs, a timeout occurs. The state machine aborts the CPU write operation and enters the ABORT STATE to assert ABUS CPU BUF ERROR (and ABUS CPU BUF DONE) to alert the MBox of the error condition. The MBox generates an EBox microtrap. The state machine then returns to the idle state. The SBIA will latch the timeout address error register, the SBI error register, and error summary register <31:26>.

3.2.9 CPU Write SBI Nexus Register: Command/Address Cycle

The PROM code enables multiplexers to gate the command/address to the SBI transceivers. In Figure 3-3, and all block diagrams that show data or control flow, only the enabled inputs to a multiplexer are shown. The other inputs will not be shown. In Figure 3-4, the ABus command, 1101, is for a write. The corresponding SBI function is write mask, 0010. Following is a detailed list of the inputs to the SBI transceivers.

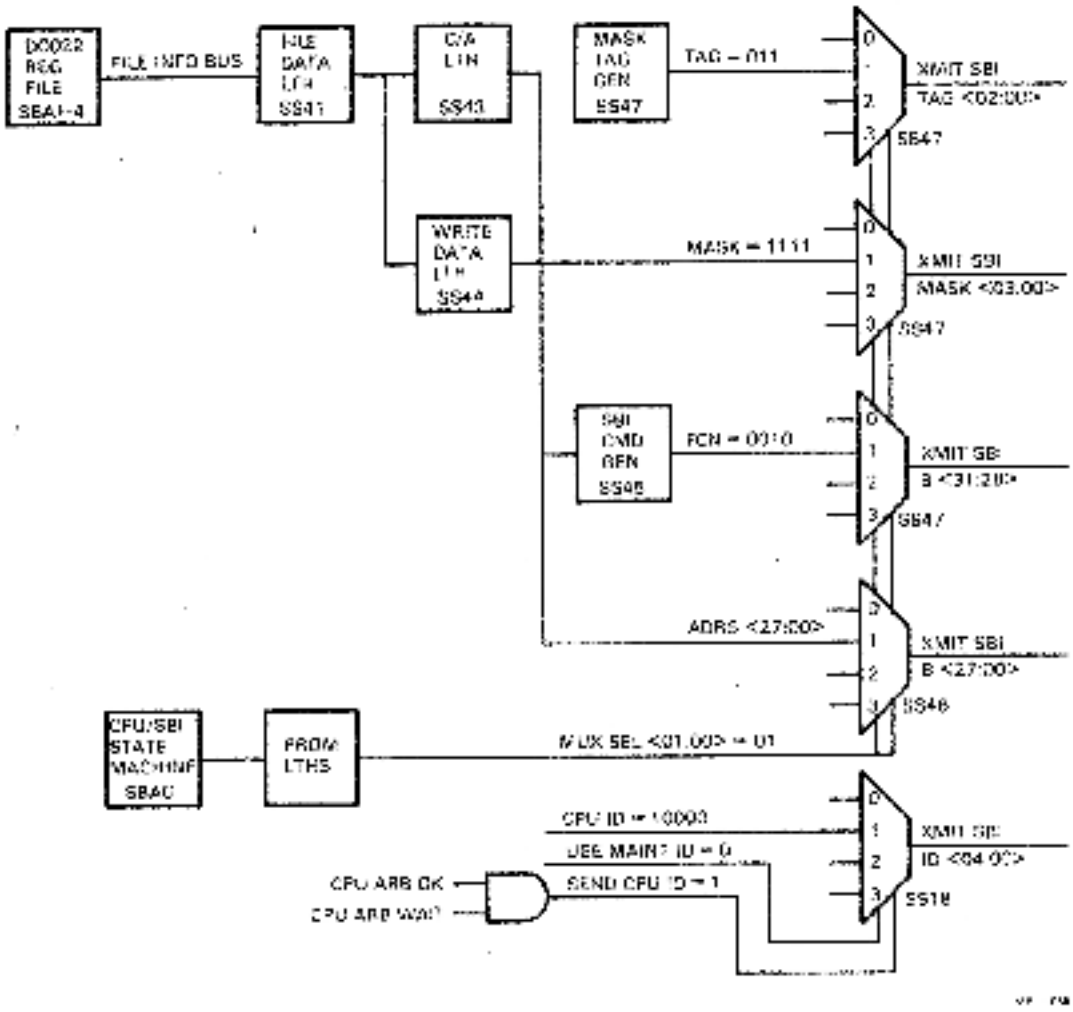


Figure 3-3 S-Data Assembly: CPU Write SBI Nexus Register Command/Address Cycle

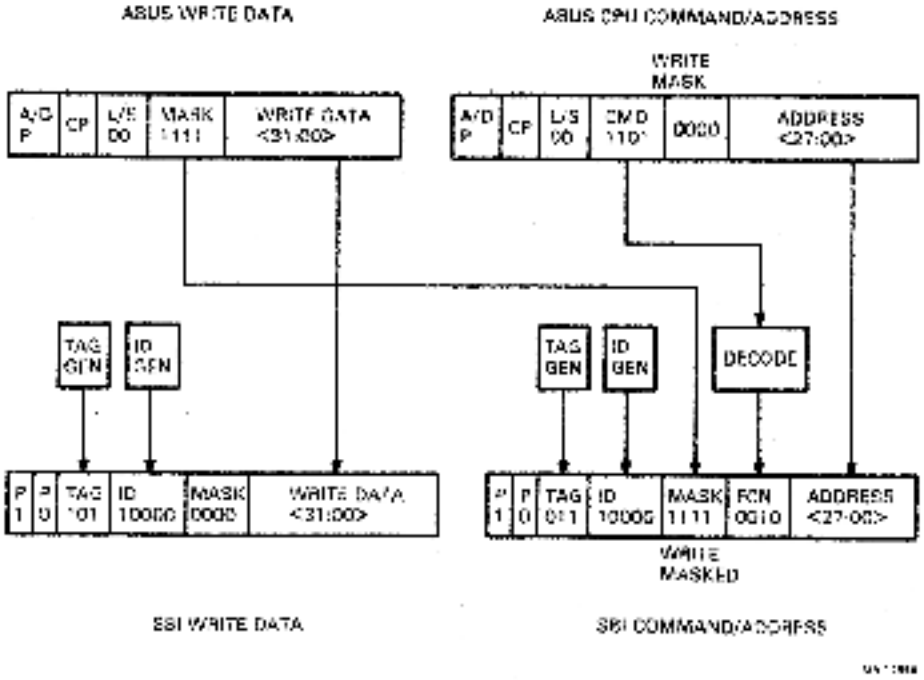


Figure 3-4 Command/Address and Write Data Transfer to SBI for CPU Write Nexus Register

1. XMIT SBI B<27:00>: The 28-bit physical address is gated to the SBI transceivers from the command/address latches.

2. XMIT SBI B<31:28>: CPU command bits in the command/address latches (CPU CMD<03,01>) are decoded by the SBI command generator to provide the SBI function. The function is passed by a multiplexer to SBI transceivers for SBI B<31:28> (see Table 3-3).

3. XMIT SBI MASK<03:00>: The PROM enables multiplexing the mask bits from the write data latches to the SBI transceivers.

4. XMIT SBI ID<04:00>: The CPU ID is hardwired to 10000, and passed by a multiplexer to the SBI transceivers.

5. XMIT SBI TAG<02:00>: The multiplexer for the tag bits is hardwired to generate a tag of 011, to indicate a command/address.

6. XMIT SBI P0: Parity for the ID bits is a hardwired input to a multiplexer and is always asserted. It is used with the write data latch control parity (parity over the length/status and mask bits) to generate XMIT SBI P0.

7. XMIT SBI P1: The address/data parity bit in the command/address latch is used as XMIT SBI P1. On the ABus, this parity bit provides odd parity over the address. The generated SBI function bits always consist of an odd number of bits. When they are concatenated with the address, the A/D parity remains correct.

Table 3-3 CPU Command Conversion to SBI Function Codes

CPU Command	CPU CMD Bits 03	02	01	00	ADR 27*	BUS SBI Bit 31	30	29	28	SBI Function Code	
Read	0	0	0	1	X	0	0	0	1	Read masked
Read lock	0	0	1	0	X	0	1	0	0	Interlock read masked
Read modify	0	1	0	0	0	0	0	0	1	Read masked
Read modify	0	1	0	0	1	0	1	0	0	Interlock read
Masked write mask	1	1	0	1	X	0	0	1	0	Write masked
Write mask unlock	1	1	1	0	X	0	1	1	1	Interlock write masked

* If ABus address bit 27 is not asserted (SBI memory), a read modify becomes a read mask. If ABus address bit 27 is asserted (SBI nexus), a read modify becomes an interlock read masked.

3.2.10 CPU Write SBI Nexus Register: Write Data Cycle

During the next SBI cycle, the PROM code will enable multiplexers to gate the write data to the SBI transceivers. Following is a detailed list of the inputs to the SBI transceivers. See Figures 3-4 and 3-5.

1. XMIT SBI B<31:00>: The CPU write data is transferred by a multiplexer to the SBI transceivers from the write data latch.

2. XMIT SBI MASK<03:00>: The PROM code disables mask bit multiplexers, forcing the mask bits to 0000.

3. XMIT SBI ID<04:00>: The CPU ID is hardwired to 10000 and passed by a multiplexer to the SBI transceivers.

4. XMIT SBI TAG<02:00>: The multiplexer for the tag bits is hardwired to generate a tag of XX1. Because the state machine is not in the ARB WAIT state, the tag will be 101 to designate write data.

5. XMIT SBI P0: The mask is all 0s, the ID is forced to 10000, and the tag is forced to 101, which is an odd number of logic 1s. P0 is hardwired through multiplexers to be a logic 1 to provide the SBI even parity.

6. XMIT SBI P1: The odd parity latched in the write data latch (with the write data word) is complemented to provide SBI even parity over the write data.

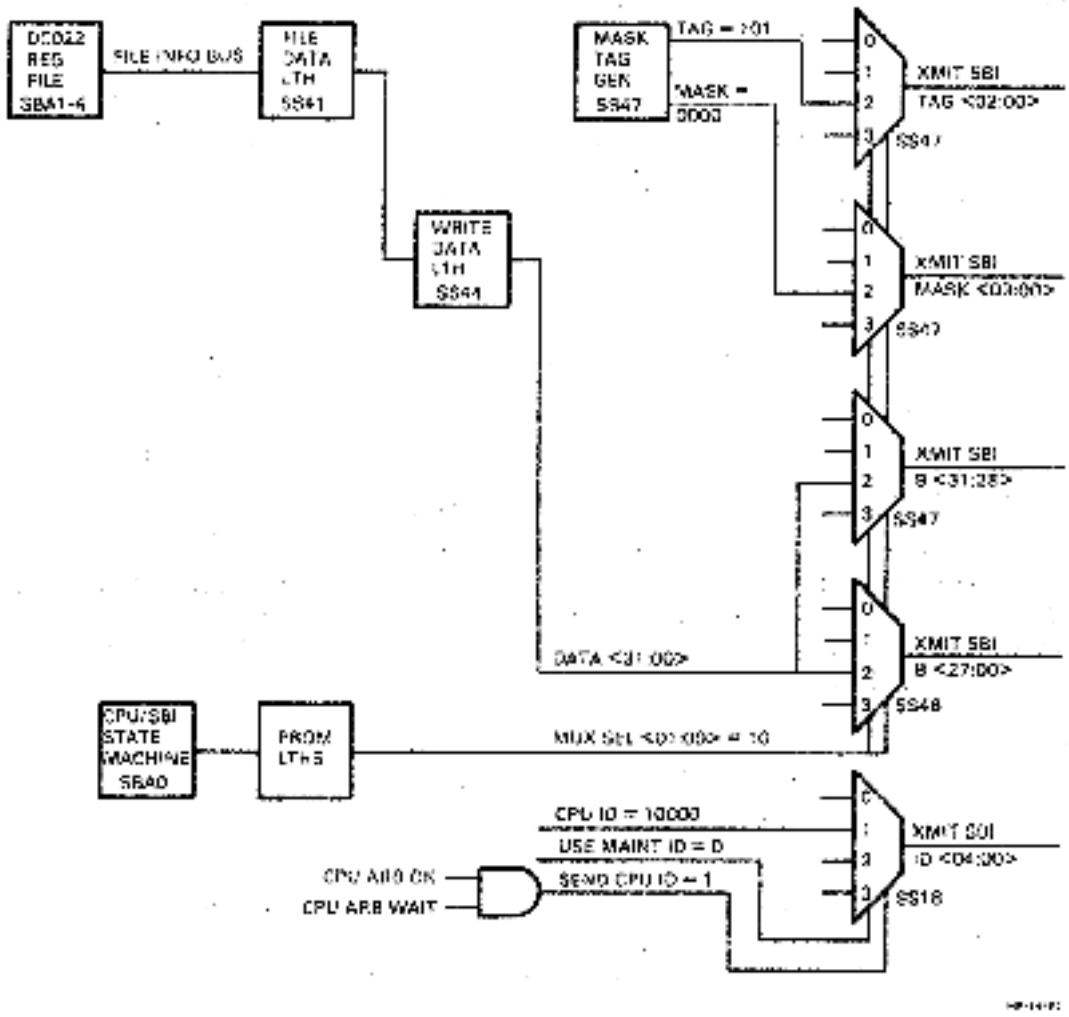


Figure 3-5 S-Data Assembly: CPU Write SBI Nexus Register Write Data Cycle

3.2.11 CPU Write SBI Nexus Register: Check ACK Cycle

Any SBI transfer requires the responding SBI nexus to transmit confirmation bits (acknowledge, busy, or error) on the second SBI cycle following the cycle in which the information was received.

The PROM code will branch on the received confirmation bits, REC SBI CONF<01:00> (see Table 3-4).

Table 3-4 SBI Confirmation Bits

SBI CONF <01:00>	Description
0 0	No response
0 1	Acknowledge
1 0	Busy
1 1	Error

1. No response and no timeout: The state machine returns to the ARB WAIT state to retransmit the command/address and write data when the SBI is available (CPU ARB OK).

2. Busy and no timeout: The state machine returns to the ARB WAIT state to retransmit the command/address and write data when the SBI is available (CPU ARB OK).

3. Error: The state machine enters the error abort state. ABUS CPU BUFF ERR will be asserted and transferred to the MBox (microtrap the EBox). Then the state machine will enter the idle state.

4. Timeout: If the state machine receives a busy response, or no response, for 102.4 μs, a timeout condition exists. The state machine enters the error abort state and asserts ABUS CPU BUF ERR. The state machine then returns to the idle state.

5. Acknowledge: The state machine goes to the CHECK ACK2 cycle to look for the acknowledge for the data word.

3.2.12 CPU Write SBI Nexus Register: Check ACK2 Cycle

The CPU-SBI state machine again branches on the confirmation bits, and in this case, if the response is busy or error, or there is no response, the state machine returns to the ARB WAIT state. It is assumed that one acknowledge indicates the device must be responding. Therefore, the SBIA will transmit the command/address and write data one more time.

If the response is an acknowledge, the CPU write function has been completed. The state machine will go to the SBI CMD DONE state, assert ABUS CPU BUF DONE to notify the MBox of the completion of the operation, and return to the idle state.

3.2.13 CPU Write SBI Nexus Register: Timeout

When the state machine leaves the idle state, a 102.4 μs timer is started. If the SBIA does not receive both acknowledges within this time, the command is aborted. The MBox is notified by the assertion of ABUS CPU BUF ERROR, and the state machine returns to the idle state.

3.3 CPU READ SBI NEXUS REGISTER

The CPU reads SBI nexus registers to respond to device interrupts or to verify register contents. The MBox generates the command/address from the EBox read request, passes the command/address to the SBIA, and then waits for the SBIA to receive the needed register information from the SBI nexus. While the MBox is waiting for CPU BUF DONE from the SBIA, it services DMA IOA requests.

The SBIA places the command/address on the SBI, and waits for the SBI nexus to acknowledge the command/address and to transmit the register data on the SBI. The SBIA takes the register data, the read data return word, and places it in the register file, and then notifies the MBox that the information is available.

When the SBIA asserts CPU BUF DONE, it is queued in the MBox arbitration logic. When the request is serviced, the MBox reads the data from the register file and passes the information to the EBox.

3.3.1 Loading CPU Command/Address for CPU Read SBI Nexus Register

The command/address word is transferred to the register file in the same manner as for a CPU write. (See Paragraph 3.2.1 and Figure 3-6).

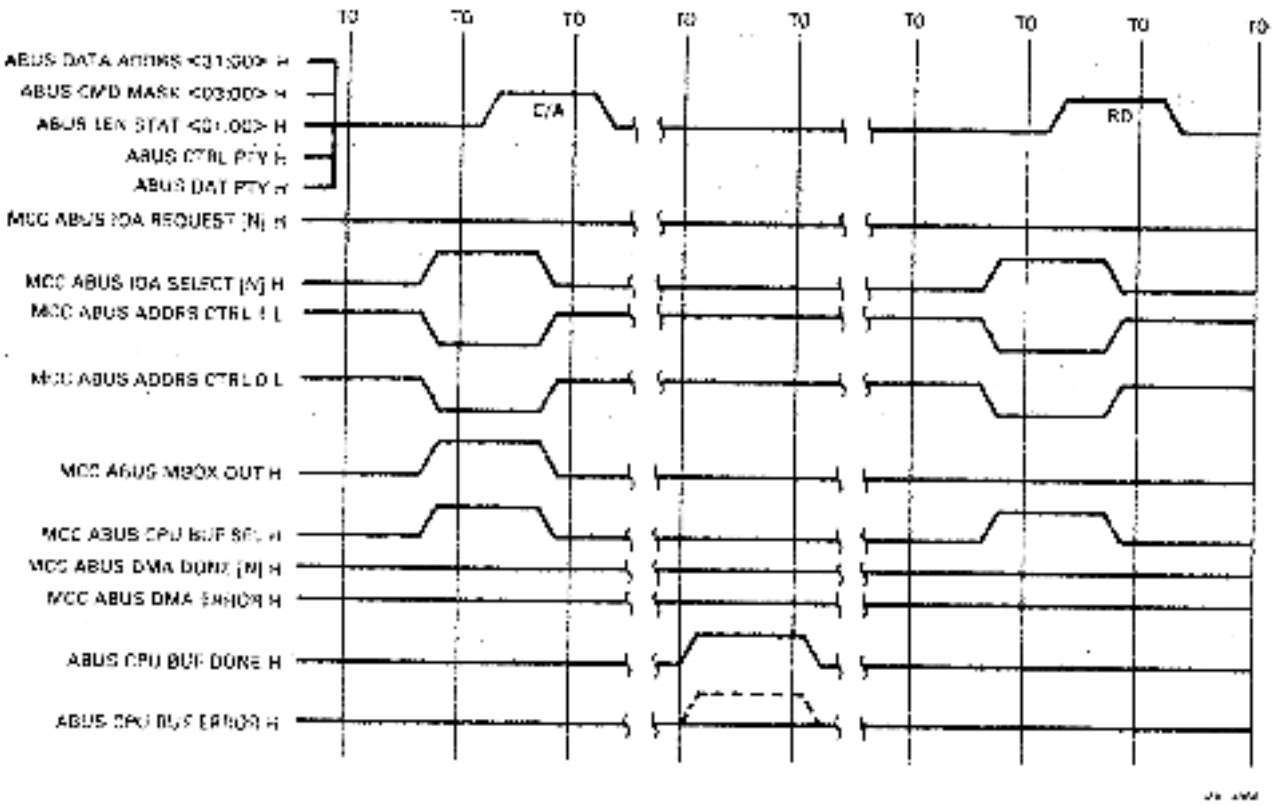


Figure 3-6 CPU Read SBI Nexus Register: ABus Protocol

3.3.2 Addressing the Register File for TTL Read Nexus Register

The register file is addressed for a TTL read in the same manner as for a CPU write, except that the address is not incremented. There is no need to increment the address because there is no write data.

3.3.3 File Data Latch

The file data latch is loaded from the addressed register file location every SBI cycle and latched at T1.5. Parity is checked over the address/data bits and control bits. The parity check is used to enable the CPU-SBI state machine to leave the idle state. The file data latch provides inputs to the command/address latch for a CPU read.

3.3.4 Loading the Command/Address Latch for CPU Read SBI Nexus Register

The command/address latch is loaded from the file data latch if the register file address is 0010, the address for the CPU command/address. It is latched at –T1 and will not change until the next CPU command/address is transferred from the register file to the file data latch, and then to the command/address latch.

3.3.5 Starting the CPU-SBI State Machine for CPU Read SBI Nexus Register

The CPU-SBI state machine will leave the idle state in the same manner as it does for a CPU write (see Paragraph 3.2.7). The difference for a CPU read is that command bit 3 is not set (to designate a read), changing one bit in the PROM address.

3.3.6 CPU Read SBI Nexus Register: CPU ARB Wait

As with a CPU write, the CPU read SBI nexus register cannot be carried out unless the SBIA can get control of the SBI at the CPU TR level. CUP ARB OK is asserted to allow the SBIA to transmit the command/address on the SBI.

Because the data transfer is a CPU read, only the command/address will be transmitted on the SBI. There is no need for the SBIA to hold the SBI for an additional cycle, therefore, CPU HOLD is not asserted by the CPU-SBI state machine.

The state machine waits in the CPU ARB WAIT state until it receives CPU ARB OK, or a timeout condition occurs. If CPU ARB OK is not received within 102.4 μs after leaving the idle state, the transaction is aborted, the state machine returns to the idle state, and ABUS CPU BUF ERROR is asserted to notify the MBox of the error condition. The MBox generates a microtrap to the EBox.

3.3.7 CPU Read SBI Nexus Register: Command/Address Cycle

When the SBIA gains control of the SBI, it transfers the command/address from the command/address latch to the SBI transceivers (see Figures 3-7 and 3-8). Figure 3-7 shows a command/address word for a CPU read of the odd word from an SBI nexus register. The multiplexers are enabled according to the following list.

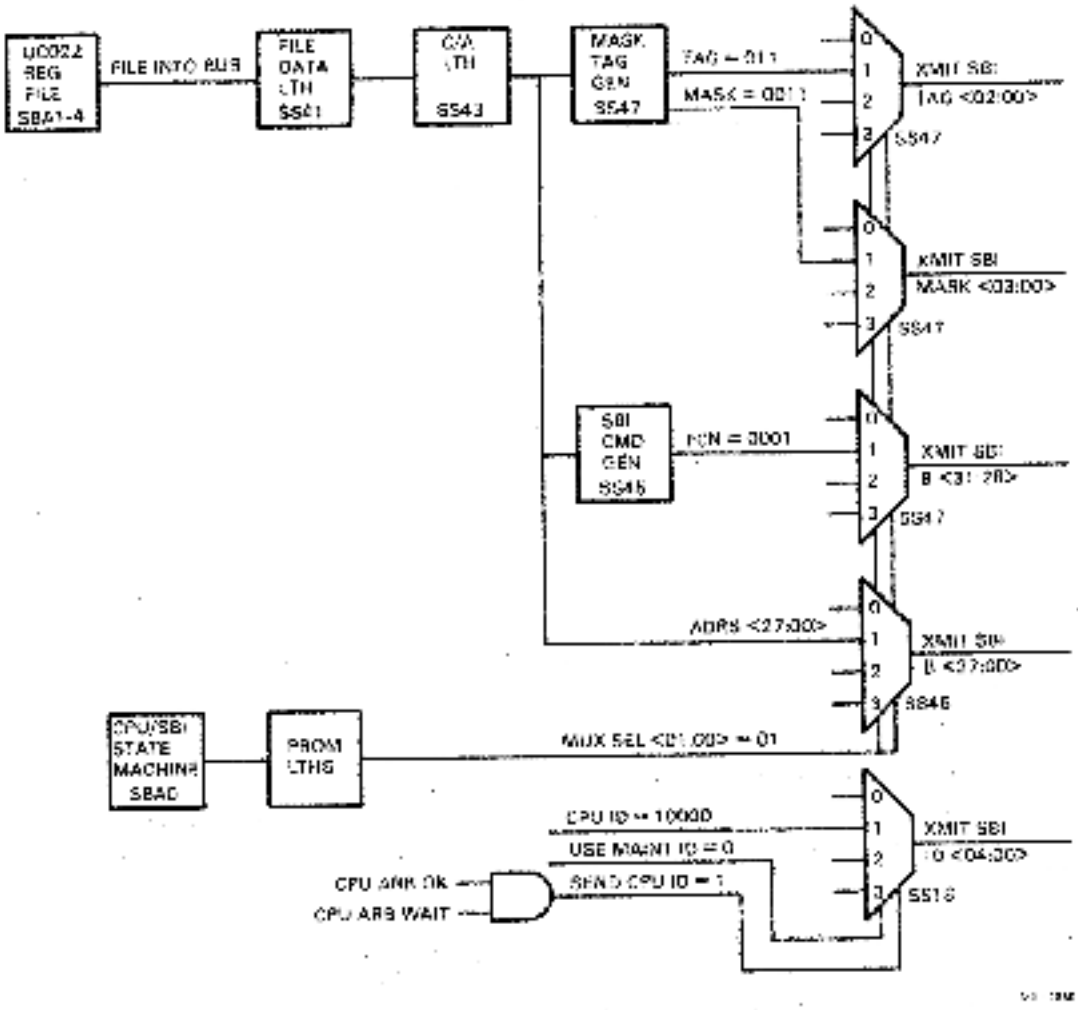


Figure 3-7 S-Data Assembly: CPU Read Nexus Register

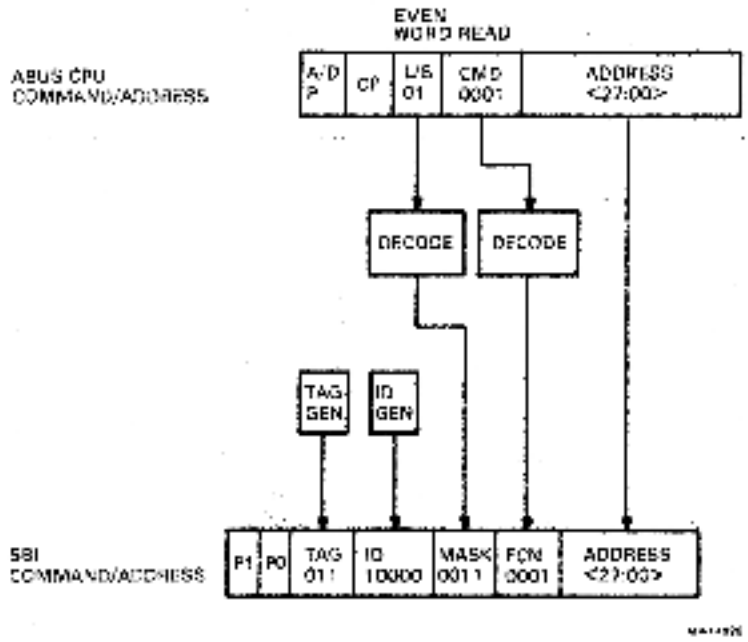


Figure 3-8 Command/Address Transfer to SBI for CPU Read SBI Nexus Register

1. XMIT SBI B<27:00>: Address bits <27:00> are transferred from the command/address latch to the SBI transceivers as the SBI address.

2. XMIT SBI B<31:28>: The CPU command (0001) is used to generate the SBI function code (0001), which is multiplexed as XMIT SBI B<31:28> to the SBI transceivers (see Table 3-5).

Table 3-5 SBI Mask Bits from CPU L/S Bits

CPU L/S Bits 01	CPU L/S Bits 00	SBI Mask 03	SBI Mask 02	SBI Mask 01	SBI Mask 00	Legend
0	0	0	0	0	0	
0	1	0	0	1	1	Read even word
1	0	1	1	0	0	Read odd word
1	1	1	1	1	1	Read long word

3. XMIT SBI MASK<03:00>: The length/status (L/S) bits in the command/address latch are multiplexed to provide the mask bits according to Table 3-5.

4. XMIT SBI ID<04:00>: The CPU ID is hardwired to 10000 and passed by a multiplexer to the SBI transceivers.

5. XMIT SBI TAG<02:00>: The multiplexer for the tag bits is hardwired to generate a tag of 011.

6. XMIT SBI P0: The mask bits will be 1100, 0011, or 1111. The tag is forced to 011, and the ID = 10000. All combinations provide an odd number of ones. XMIT SBI P0 is forced to a 1 to provide even parity over the TAG, ID, and mask bits.

7. XMIT SBI P1: The address/data odd parity latched in the command/address latch is complemented to provide SBI even parity over the SBI function and address. This can be done because both the ABus command and corresponding SBI function have an odd number of 1s. The address does not change between the ABus and SBI.

3.3.8 CPU Read SBI Nexus Register: Wait Cycle

SBI protocol requires that the responding device transmit the confirmation bits on the second SBI cycle after the received information was transmitted. Because the SBIA sends only a command/address word, it must wait one cycle before it can monitor for the confirmation.

3.3.9 CPU Read SBI Nexus Register: Check ACK Cycle

After waiting for one cycle, the CPU-SBI state machine will branch on the SBI confirmation bits (see Table 3-4).

1. No response and no timeout: If there is no response and no timeout, the state machine goes to the ARB WAIT state to wait until the SBI is available. The command/address will be retransmitted (CPU ARB WAIT).

2. Busy and no timeout: If the responding SBI device is busy, the state machine also returns to the ARB WAIT state to wait until the SBI is available. The command/address is retransmitted (CPU ARB WAIT).

3. Error: If the confirmation bits indicate an error condition, the state machine branches to the error abort state, and issues ABUS CPU BUF ERR to cause the MBox to generate an EBox microtrap. The state machine aborts the CPU read operation and returns to the idle state.

4. Timeout: If a timeout condition occurs (102.4 μs) while receiving a busy confirmation code, or no response, the state machine branches to the error abort state, the operation will be aborted, ABUS CPU BUF ERR will be asserted, and then the state machine will return to the idle state.

5. Acknowledge: Upon receipt of the expected response, acknowledge, the state machine waits for the read data return word.

3.3.10 CPU Read SBI Nexus Register: Read Wait Start

The state machine clears the timeout counter and then restarts it. It then checks for a timeout while waiting for the read return data.

3.3.11 CPU Read SBI Nexus Register: Read Data Wait

The state machine will cycle in this state until the SBIA receives the read data or a timeout occurs.

1. If the read data is not received within 102.4 μs, a timeout condition exists. The SBIA state machine branches to the error abort state, asserts ABUS CPU BUF ERR, and aborts the transaction. The state machine then goes to the idle state.

2. If the read data is received within 102.4 μs the state machine will branch to the CMD DONE state and assert ABUS CPU BUF DONE to inform the MBox that the read data is available. The state machine then goes to the idle state. ABUS CPU BUF DONE is delayed by two SBI cycles to insure that the SBIA has time to get the read data to the register file.

The SBIA monitors the SBI transceivers every SBI cycle. If the received SBI tag is 000 and the received ID is 10000, it is the read data. The read data is transferred from the SBI transceivers to the register file.

3.3.12 Sending Acknowledge for the Read Data Word

When the SBIA receives the read data, if there is no parity error, the tag will be decoded. If the tag is 000, indicating read data, and the ID is 10000, the CPU ID, then the SBIA will send an acknowledge. When the acknowledge is enabled, the read data is enabled to be sent to the register file, and, after a 30 ns delay, register file write is asserted.

If there is a parity error, the acknowledge is not enabled. The responding device assumes NO RESPONSE and retransmits the read data. The SBIA state machine will wait 102.4 μs for valid read data before it aborts.

3.3.13 CPU Read SBI Nexus Register: Read Data Transfer to Register File

The inputs to the register file, the file info bus, are generated from the read data in the SBI transceivers. The circuitry involved is the A-data assembly, which consists of registers, multiplexers, tri-state multiplexers, tri-state buffers, a PROM, and various gating and decoding logic.

The CPU-SBI state machine goes to the SBI CMD DONE state, then the idle state upon detection of the read data and has no control over the transfer of the read data to the register file.

The tri-state latches and buffers that gate the contents of the SBI transceivers to the file info bus are enabled at T2T3 when it is determined to acknowledge the read data. After a 30 ns delay, the file info bus data is written into the DC022 register file.

The multiplexers and buffers are enabled in the following way (see Figure 3-9).

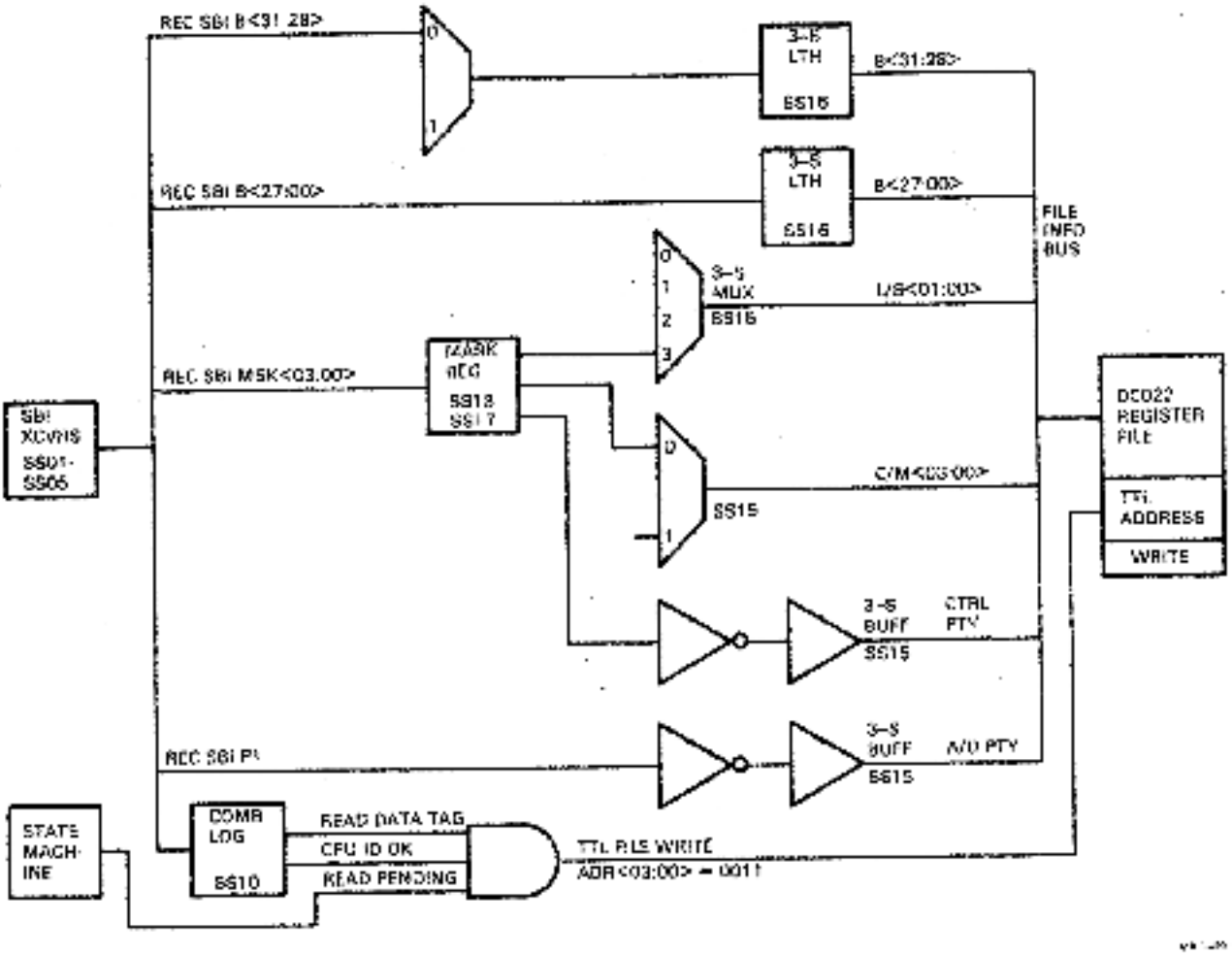


Figure 3-9 A-Data Assembly: CPU Read Nexus Register, Read Data

1. BUS FILE INFO B<31:28>: The tri-state latches contain REC SBI B<31:28> as long as the tag is not for a command/address word.

2. BUS FILE INFO B<27:00>: These tri-state latches are always loaded with REC SBI B<27:00>.

3. BUS FILE INFO L/S<01:00>: The mask register contains the mask bits. If mask bit 1 is set, it indicates an error condition on the SBI. If there is an error, BUS FILE INFO L/S<1:0> will both be set. If there is no error, neither bit is set.

4. BUS FILE INFO C/M<03:00>: Although not used by the MBox, the SBI command/mask bits are gated to the register file.

5. BUS FILE INFO CTR PTY: Because the L/S bits will be either 00 (no error) or 11 (error), they always have an even number of 1s. Therefore, the control parity depends on the command/mask parity. The command mask parity bit from the SBI transceivers is inverted and routed to the register file as control parity.

6. BUS FILE INFO A/D PTY: BUS SBI P1, the SBI parity bit over the data bits is inverted and as BUS FILE INFO A/D PTY is transferred to the register file.

3.3.14 CPU Read SBI Nexus Register: Register File TTL Write Address

Register file TTL write address bits <03:00> are forced to 0011 if the following conditions exist.

1. The received SBI data has a tag of 000, read data
2. The received SBI ID = 10000, the CPU ID
3. The state machine is waiting for read data and has asserted READ PENDING.

3.3.15 CPU Read SBI Nexus Register: ABUS CPU BUF DONE

The events that allow the sending of an acknowledge will also cause the SBIA state machine to leave the READ DATA WAIT state for the SBI CMD DONE state. The new state will initiate sending ABUS CPU BUF DONE to the MBox. When the MBox receives ABUS CPU BUF DONE, it knows that the read data is in the register file.

3.3.16 CPU Read SBI Nexus Register: MBox Reads the Register File

After the MBox services the CPU BUF DONE, it reads the SBIA register file to get the read data. The MBox must set up the register file read address and enable reading the register file (see Figure 3-6).

The MBox selects the applicable SBIA with MCC ABUS IOA SELECT [N]. MCC ABUS ADDRS CTRL<01:00> = 00 will enable loading the register file address. The file address least two significant bits are determined by MCC ABUS CPU BUF SEL and MCC ABUS MBOX OUT = 10 (See Figure 3-1). Because MCC ABUS CPU BUF SEL is asserted, the upper two address bits will be 00 to provide a register file address of 0011, the address for the read data word.

A register file read is enabled because MCC ABUS MBOX OUT is not asserted.

3.4 CPU WRITE SBIA REGISTER

The data transfer during a CPU write to an SBIA register is much like a CPU write to an SBI nexus register except the data is not transmitted on the SBI. The SBI transceivers are not enabled to transmit because the SBIA decodes the address as an address for an SBIA register.

Also, because the address is not for an SBI nexus register or vector register (interrupt status read, ISR), the state machine never leaves the idle state.

The MBox loads the command/address and write data into the register file in the same manner as for a CPU write to an SBI register (see Paragraphs 3.2.1 and 3.2.2).

The register file is addressed as for a CPU write to an SBI register. Also, the command/address and write data are transferred to the command/address latch and write data latch as for a CPU write to an SBI nexus register.

The command/address is used by the address decode and control logic to verify a valid address and generate the register write pulse. The contents of the write data latch will be written into the SBIA register that receives the write pulse (see Figure 3-10).

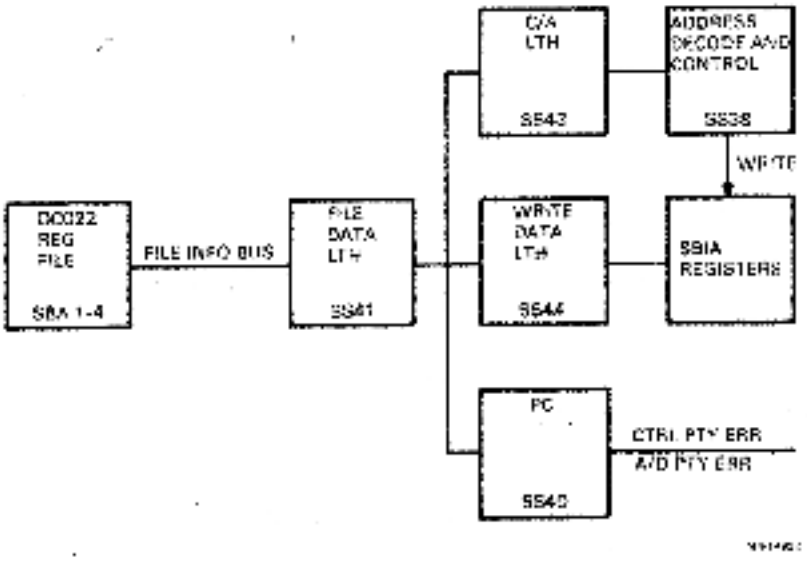


Figure 3-10 CPU Write SBIA Registers

3.4.1 SBIA Address Recognition

The register address decode logic verifies that the hex longword address is in the range of SBIA register addresses, 802 0000 to 802 FFFF. Of these SBIA addresses, the SBIA registers are at addresses 802 0000 to 802 0013. PROM E1 monitors the hex longword address, and one of the outputs, REG ADR OK, is asserted if the address is a valid SBIA register address (see Figure 3-11).

3.4.2 Selecting and Writing the SBIA Register

Other PROMs are addressed by address bits <05:00>. Three of the outputs of PROM E166 are decoded to provide the enabling for the register write pulse. The register is written at the next SBI T0 (see Figure 3-12). The decoder is enabled by LOCAL WRITE GO if the following conditions exist.

1. No control parity error on command/address word
2. No A/D parity error on command/address word
3. Command is for a CPU write
4. The register address is a valid address
5. No control parity error on data word
6. No A/D parity error on data word.

3.4.3 CPU Write SBIA Register: ABUS CPU BUF DONE

LOCAL WRITE GO is also used to generate ABUS CPU BUF DONE to notify the MBox that the operation is complete.

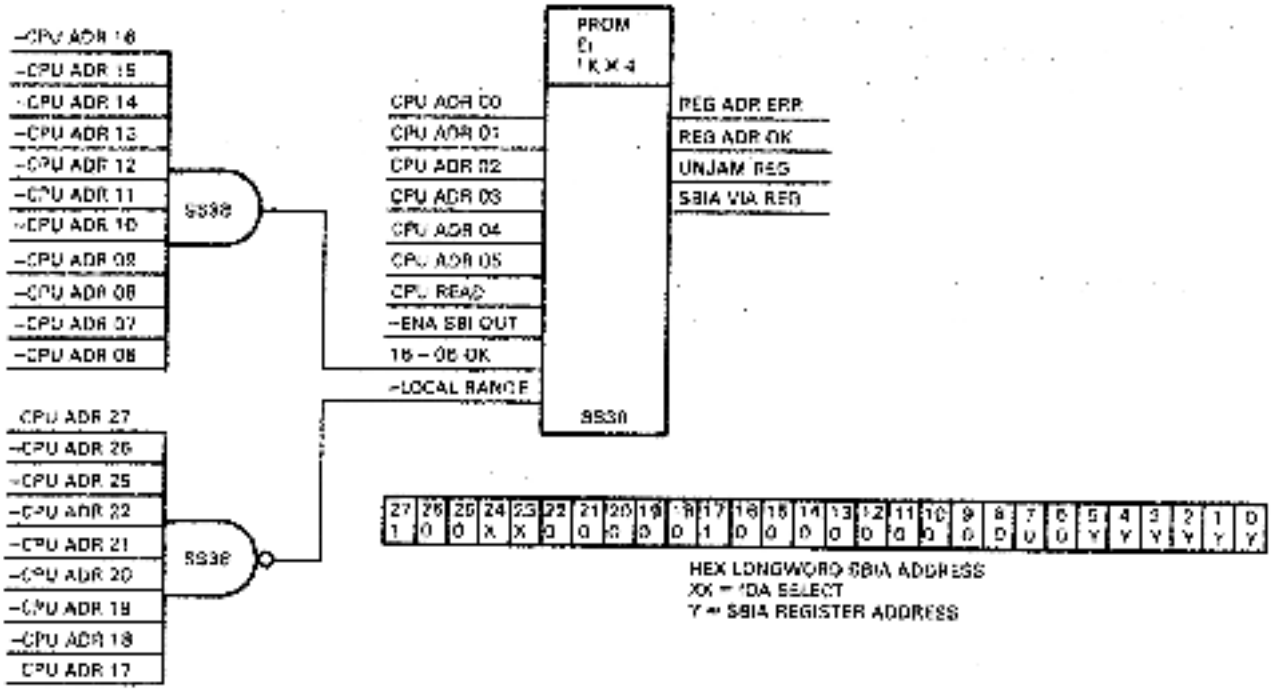


Figure 3-11 Register Address Decode Logic

3.4.4 CPU Write SBIA Register: ABUS CPU BUF ERROR

ABUS CPU BUF ERROR is sent to the MBox during a CPU write SBIA register if the SBIA detects any of the following errors.

1. CMD ERR: CMD ERR will be asserted if any one of the following conditions is true.
   a. Address not a valid SBIA register address
   b. A/D parity error on command/address cycle
   c. Control parity error on command/address cycle.
2. Data error: ABUS CPU BUF ERROR will be caused by a data error under the following circumstances if the command/address was not in error and the command is for a CPU write:
   a. A/D parity error on the write data cycle
   b. Control parity error on the write data cycle.

If there is an error on the register write, ABUS CPU BUF DONE will be asserted by ABUS CPU BUF ERROR.

3.5 CPU READ SBIA REGISTER

When the CPU wishes to read an SBIA register, the MBox will generate a command/address word from the EBox read request. The command/address will be written into the SBIA register file just as the command/address for a CPU read SBI nexus register. (See Paragraphs 3.3.1 and 3.3.2.)

The register file is addressed for a TTL read and the command/address word is transferred to the file data latch, and then to the command/address latch as for a CPU read SBI nexus register. (See Paragraphs 3.3.3 and 3.3.4.)

The register address is decoded as for a CPU SBIA register write, and the PROMs are addressed in the same manner. (See Paragraph 3.4.1 and Figures 3-11 and 3-12.)

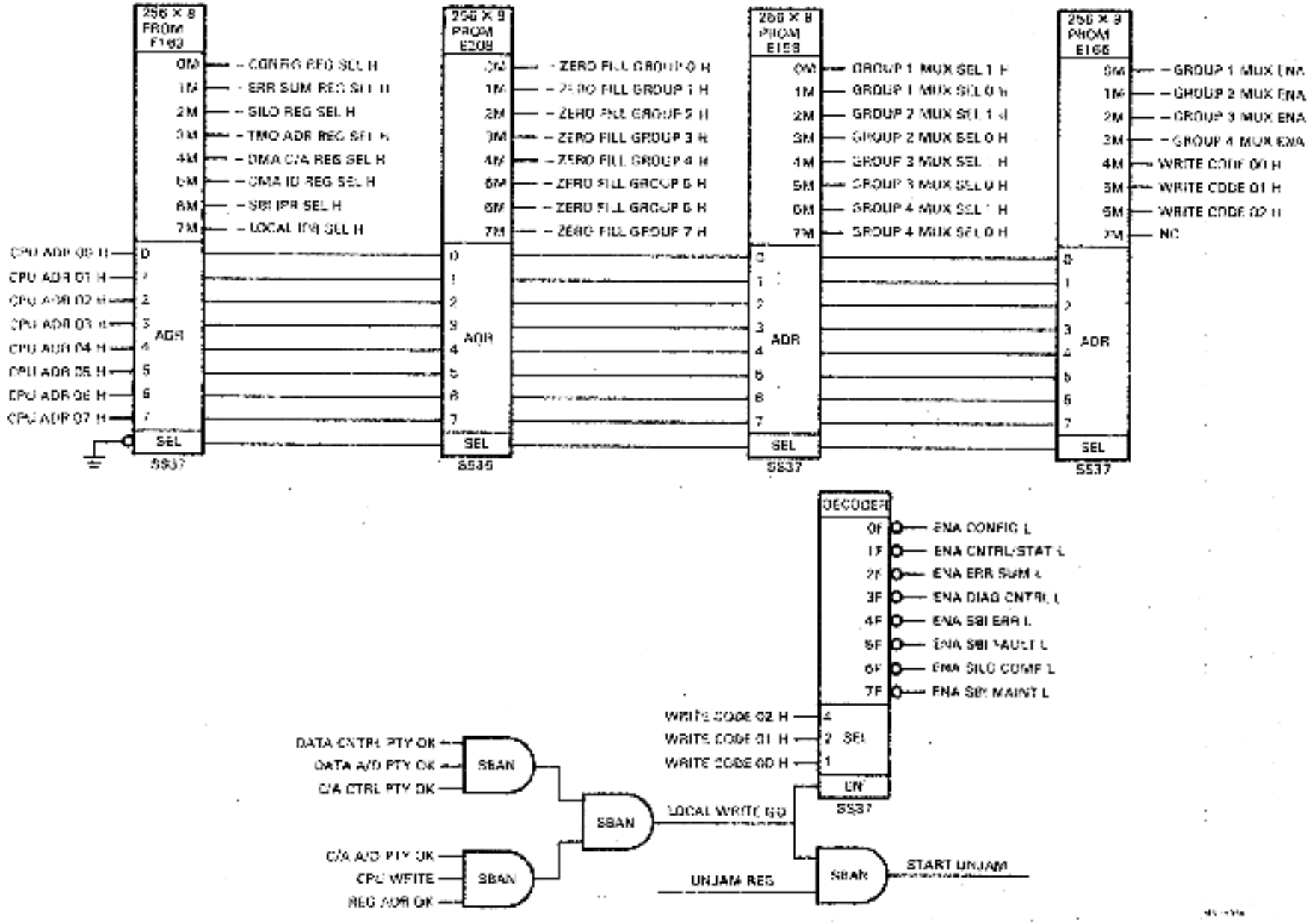


Figure 3-12 Register Selection, Zero Fill, and Write Enables

3.5.1 Register Data Bus

The SBIA register outputs are gated to the register data bus (see Figure 3-13), transferred to the file info bus by tri-state latches, and then written into the DC022 register file.

The SBIA registers with outputs used by the SBIA logic are made up of flip-flop and latch registers. These outputs are not tri-state. The SBIA registers that have outputs not used by SBIA logic are made up of tri-state latches, a tri-state PROM, a tri-state register file, and tri-state RAMs.

The registers made up of tri-state logic can be enabled directly to the register bus, but those registers that are not tri-state logic must be multiplexed by a tri-state multiplexer to the register bus.

PROMs E158 and E166 (Figure 3-12) are addressed by CPU ADR<05:00>. These PROMS enable the multiplexing of the the control and status register, error summary register, diagnostic control register, SBI error register, SBI fault status register, SBI silo comparator register, and SBI maintenance register to the register bus. The multiplexer is only a 4-to-1 multiplexer, but because all bits of the registers are not used, bits not used by one register, on one multiplexer input, can be used for another register. If GROUP 1 MUX ENA is asserted, GROUP 1 MUX SEL<01:00> controls which register is gated to register bus bits <31:24> (see Table 3-6). If GROUP 2 MUX ENA is asserted, GROUP 2 MUX SEL<01:00> controls which register is gated to register bus bits <23:20>.

Table 3-6 Register Bus Multiplexer Enabling

	GROUP 1 MUX ENA	GROUP 2 MUX ENA	GROUP 3 MUX ENA	GROUP 4 MUX ENA
GROUP [N]* MUX SEL<01:00>	**Register Bus <31:24>**	**Register Bus <23:20>**	**Register Bus <19:16>**	**Register Bus <15:00>**
00	Control status register	Error summary register	Error summary register	Error summary register
01	Maintenance register	Maintenance register	Diagnostic control	Diagnostic control
10	Fault status register	Fault status register	Fault status register	SBI error register
11	Silo comparator	Silo comparator	Silo comparator	Maintenance register

* GROUP [N] MUX SEL <01:00> refers to GROUP 1 MUX SEL<01:00> for GROUP 1 MUX ENA; to GROUP 2 MUX SEL<01:00> for GROUP 2 MUX ENA; to GROUP 3 MUX SEL<01:00> for GROUP 3 MUX ENA; and to GROUP 4 MUX SEL<01:00> for GROUP 4 MUX ENA.

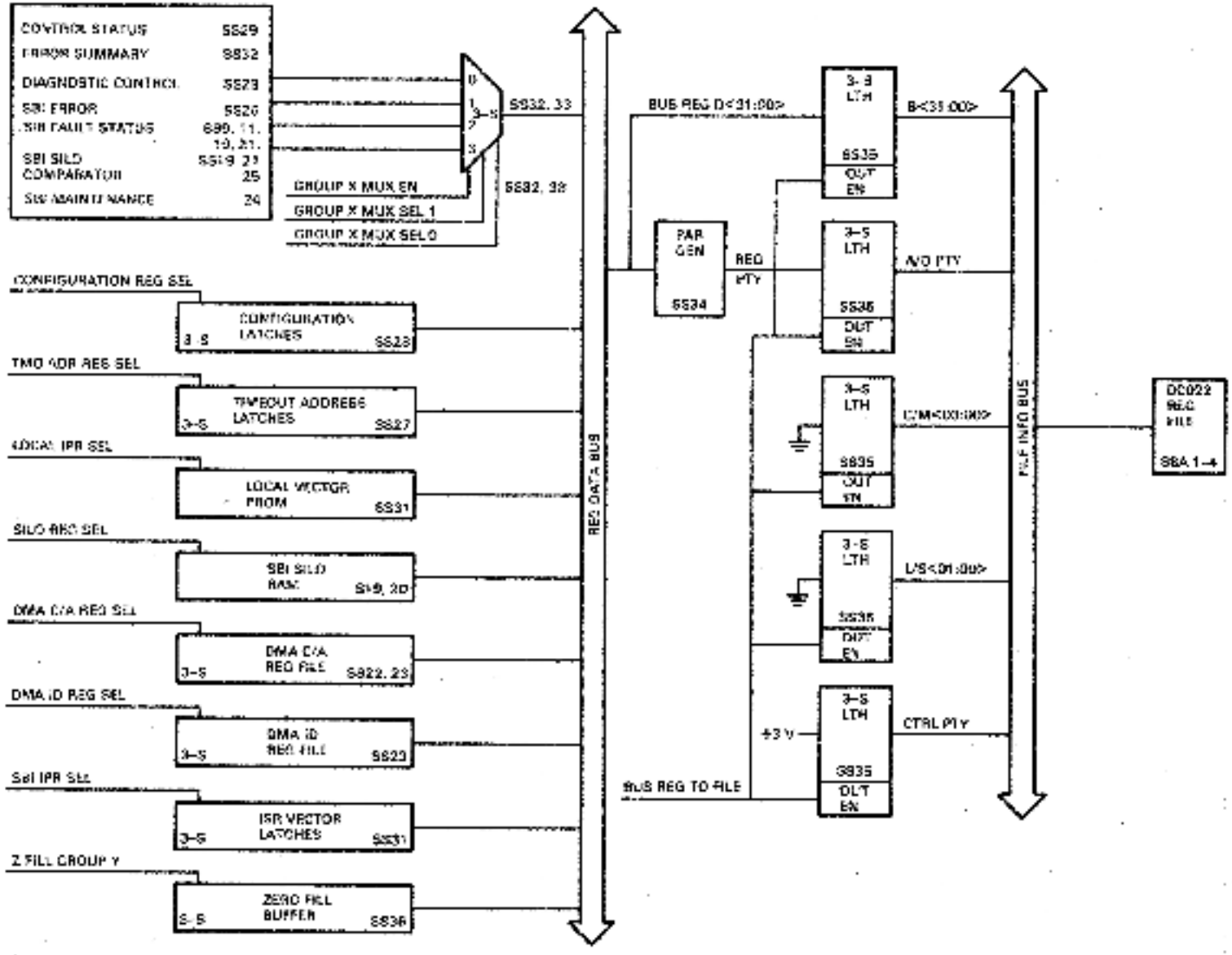


Figure 3-13 Read SBIA Registers

3.5.2 Zero Fill

Zeros must be placed on the register bus because not all of the register bits are used. The 0s are provided by tri-state buffers enabled by PROM E208 (Figure 3-12). This PROM is also addressed by CPU ADR<05:00>. The register bus bits that receive 0 from the tri-state buffer enabled by the zero fill PROM are shown in Table 3-7.

Table 3-7 Register Bus Zero Fill

Register Name	Bits Zero Filled
Configuration register	<19:08>
Control and status register	<23:00>
Error summary register	None
Diagnostic control register	<31:20>
DMA* command/address register	None
DMA* ID register	<31:08>
SBI silo register	None
SBI error register	<31:16>
SBI timeout address register	<31:28>
SBI fault status register	<15:00>
SBI silo comparator register	<15:00>
SBI maintenance register	<19:16>
SBI unjam register	<31:00>
SBI quadclear register	<31:00>
Vectors 24–27	<31:12>
Vectors 28- 2E	<31:08>

*DMAA, DMAB, DMAC, or DMAI

3.5.3 Enabling Register Data to File Info Bus

Tri-state latches are used to enable register data onto the file info bus because the file info bus may also carry information from the SBI. The file info bus contains address/data bits, command/mask bits, length/status bits, A/D parity, and control parity. When an SBIA register is read, only 32 bits of data are available from the register. The registers do not provide command/mask bits, length/status bits, or control parity (see Figure 3-13).

There are tri-state latches, with grounded inputs, to provide the command/mask and length/status bits. Because these bits will be all 0s on the file info bus, the control parity must be a logic 1. The tri-state latch that drives file info bus CTR PTY is tied high. A parity generator will monitor the register data bus and generate odd parity over the 32 bits. This parity bit will be enabled to the file info bus as A/D parity.

All of the tri-state latches are enabled by BUS REG TO FILE, which is asserted for an interrupt status read (ISR), by the state machine, or during a CPU read SBIA register if the following conditions exist (see Figure 3-14):

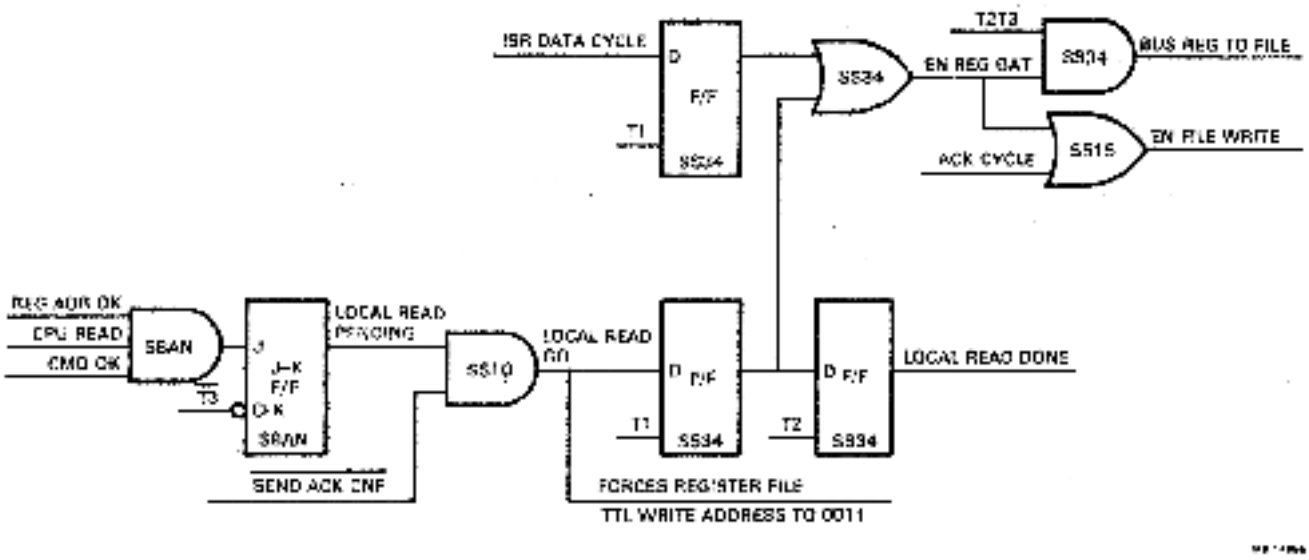


Figure 3-14 Enabling Register Data Bus and Local Read Done

1. The register address is valid.
2. The command is for a CPU read.
3. There is no A/D parity error over the command/address.
4. There is no control parity error over command/address.
5. SEND ACK CNF has not been asserted to send an acknowledge on the SBI. This signal is provided to prevent placing both register data and SBI data on the file info bus at the same time.

The register data is written into the DC022 register file 30 ns after EN FILE WRITE is asserted.

3.5.4 Register File TTL Write Address

LOCAL READ GO, one of the intermediate signals in Figure 3-14, is used to force the register file TTL write address to 0011, the location for the CPU read data.

3.5.5 CPU Read SBIA Register: ABUS CPU BUF DONE

LOCAL READ DONE (Figure 3-14) is used to assert ABUS CPU BUF DONE, which queues up the request in the MBox.

3.5.6 CPU Read SBIA Register: MBox Reads the Register File

The MBox will read the register data in the same way as for the register file for SBI nexus register data (see Paragraph 3.3.16).

3.5.7 CPU Read SBIA Register: ABUS CPU BUF ERROR

The MBox will be informed of a command error by the assertion of ABUS CPU BUF ERROR if any of the following conditions existed.

1. The address is not a valid SBIA register address.
2. Control parity error on the command/address word.
3. A/D parity error on the command/address word.

3.6 INTERRUPT SUMMARY READ

3.6.1 Interrupt Requests

If interrupts have been enabled by control and status register bit 31, the SBIA arbitrates SBI interrupt requests. When the EBox polls the SBIA with ABUS IOA SELECT [N], the SBIA provides the EBox with ABUS IPR RETURN <4:0>, an encoded priority of the interrupt requests. Table 3-8 lists the priority, with SBIA FAIL having the highest priority and SBI REQ 4 the lowest.

Table 3-8 Interrupt Priority

Request	IPR Level	ABUS IPR Return <4:0>
SBI FAIL	1E	11110
FAULT/ERR	1C	11100
SBI ALERT	1B	11011
COMP INT	19	11001
SBI REQ 7	17	10111
SBI REQ 6	16	10110
SBI REQ 5	15	10101
SBI REQ 4	14	10100

3.6.2 EBox IPR Arbitration

The EBox (EBC module) arbitrates ABUS IPR RETURN <4:0>, against pending interrupt requests. The highest priority request is held as the pending external interrupt. The highest priority external interrupt is compared to any active internal interrupt. If an external and internal interrupt of equal priority are active, the internal interrupt is given higher priority. The highest priority interrupt is then compared to the processor status longword interrupt priority level (PSL IPL). If the active interrupt priority is higher than the priority set for the CPU, the interrupt will be serviced.

3.6.3 EBox Microcode Generates the Read Address

For the EBox to service an external interrupt, it must determine the interrupt vector by reading an IOA vector register. The microcode builds the address for the register it will read. Table 3-9 lists the bytes and longword addresses, with vectors, for the external interrupts.

Table 3-9 Vector Register Addresses and Interrupt Vectors

Request	IPR Level	Hex Byte Address*	Hex Longword Address*	Non-Local Error	Local Error
SBI REQ 4	14	2008 0090 2208 0090	802 0024 882 0024	100–13C	
SBI REQ 5	15	2008 0094 2208 0094	802 0025 882 0025	140–17C	
SBI REQ 6	16	2008 0098 2208 0098	802 0026 882 0026	180–1BC	
SBI REQ 7	17	2008 009C 2208 009C	802 0027 882 0027	1C0–1FC	
COMP INT	19	2008 00A4 2208 00A4	802 0029 882 0029	50	50
SBI ALERT	1B	2008 00AC 2208 00AC	802 002B 882 002B	58	58
FAULT/ SBIA ERR	1C	2008 00B0 2208 00B0	802 002C 882 002C	5C	60
SBI FAIL	1E	2008 00B8 2208 00B8	802 002E 882 002E	64	64

*The first address is for SBIA 0, the second for SBIA 1

3.6.4 Command/Address

The MBox loads the command/address into the SBIA register file in the same manner as for a CPU read SBI nexus register. (See Paragraph 3.3.1 and Figure 3-6.) The register file is addressed, and the command/address transferred to the file data latch, and then to the command/address latch, as for a CPU read SBI nexus register. (See Paragraphs 3.3.2 through 3.3.4.)

3.6.5 Obtaining the Interrupt Vector for IPR 14–IPR 17

The action taken by the SBIA depends upon the register address. If the register address is for SBI interrupt request, IPR14–IPR17, an interrupt summary read (ISR) is required to obtain the interrupt vector. If the register address is for IPRs 19, 1B, 1C, or 1E, an ISR is not required, and the vector is read from a PROM.

3.6.5.1 IPR 14–IPR 17 – The register address decode logic will address PROM E1. Because it is a CPU read SBIA register, with the SBI enabled, the PROM address will be 164–167. The output in each case will be 1000, SBI VIA REG (see Figure 3-11). If the command/address does not have a parity error, SBI CMD GO will cause the CPU-SBI state machine to leave the idle state for the ISR CPU ARB WAIT state. If there is a command/address parity error, ABUS CPU BUF ERROR will be sent to the MBox along with ABUS CPU BUF DONE. If there is a command/address parity error, the state machine does not leave the idle state because SBI CMD GO is never asserted.

3.6.5.2 ISR CPU ARB Wait Cycle – The state machine awaits CPU ARB OK as for a CPU read SBI nexus register (see Paragraph 3.3.6). If a timeout occurs, the state machine goes to the error abort state and sends ABUS CPU BUF ERROR to the MBox. The state machine then returns to the idle state.

3.6.5.3 ISR C/A Cycle – The state machine asserts CPU HOLD to cause TR00 to hold the SBI. The SBI will be held for two cycles because the interrupting nexus is expected to respond during the second cycle after the command/address cycle. When the CPU gains control of the SBI, the command/address is transferred to the SBI by the S-data assembly, according to the following list (see Figure 3-15).

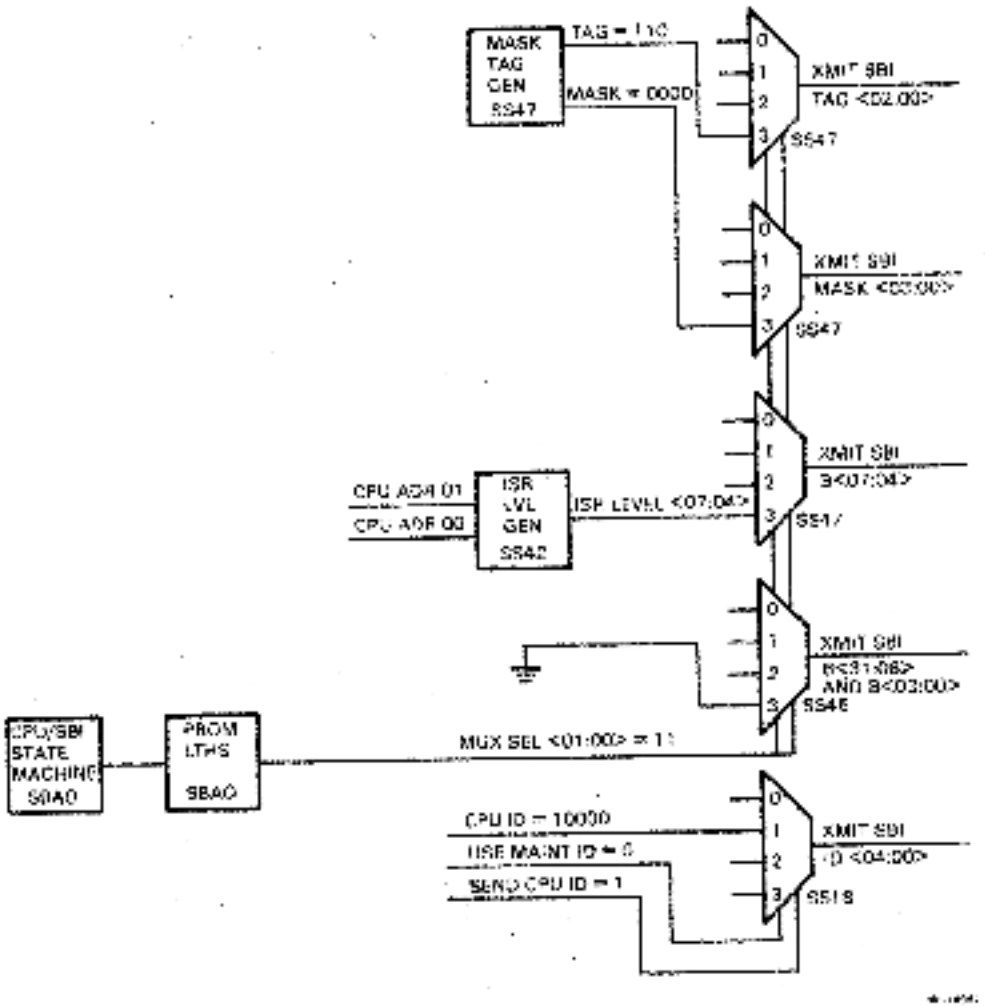


Figure 3-15 S-Data Assembly: Interrupt Summary Read

1. TAG <2:0> is forced to 110.
2. MASK <3:0> is forced to 0000.
3. B<31:08> and B<03:00> are all 0s because the multiplexer inputs are all grounded.
4. B<07:04> will have one bit set. The bit set corresponds to the interrupt priority level being serviced. For example, if the interrupt level is 5, B<05> is set. The bit set, <07:04>, corresponds to the least two significant bits of the register address (see Table 3-10).

Table 3-10 Setting SBI B<07:04> for ISR

CPU ADR <01:00>	SBI B<07:04>
0 0	0 0 0 1
0 1	0 0 1 0
1 0	0 1 0 0
1 1	1 0 0 0

5. ID<4:0> is forced to 10000.

6. P0 is forced high because the ID, TAG, and MASK bits always have an odd number of 1s.

7. P1 is also forced high because the data bits will always have one bit set – bit 7, 6, 5, or 4.

3.6.5.4 ISR Wait Cycle – The state machine waits one cycle. CPU HOLD is still asserted to keep TR00 asserted. The interrupt summary response is expected in the next cycle.

All nexus devices receive the ISR command/address cycle. Those nexus devices that are interrupting at the interrupt level being serviced, as indicated in SBI B<07:04>, will be required to respond two cycles after receiving the ISR command/address. Each responding nexus will set two data bits, one corresponding to the nexus TR level, and the other corresponding to the nexus TR level +16. It is not known how many nexus will respond, but by requiring each nexus to set two bits, the number of logic 1s remains even. Besides these bits, each responding nexus sets SBI ID<4:0> to 10000, the CPU ID, and SBI P0.

3.6.5.5 ISR Data Cycle – On the second SBI cycle after the command/address cycle, the SBIA checks parity on the interrupt summary response. P0 should be high and P1 should be low. If there is a parity error, and no timeout, the SBIA will return to the ARB WAIT cycle to retransmit the command/address.

If there is a timeout, the state machine aborts the transaction and asserts ABUS CPU BUF ERROR to notify the MBox of the error condition.

3.6.5.6 SBI CMD DONE – If there are no parity errors, the state machine proceeds to the SBI CMD DONE state and asserts ABUS CPU BUF DONE to inform the MBox that the vector is available. Then the state machine goes to the idle state.

3.6.5.7 Vector Transfer to the Register File – In the meantime, REC SBI B<15:01>, the TR levels for all interrupting devices, are arbitrated to determine which TR level has priority. The highest priority TR, along with the lower two bits of the interrupt priority request level (derived from the lower two bits of the register address), are gated to the register data bus by SBI IPR SEL L (see Figure 3-13).

SBI IPR SEL L is asserted by PROM E163 (see Figure 3-12). The PROM output is BF (1011 1111) for any ISR.

The vector and interrupt level are gated to the register data bus according to Figure 3-16. REG DATA BUS <31:12> receives all 0s from the zero fill tri-state buffers (see Figure 3-13 and Table 3-7).

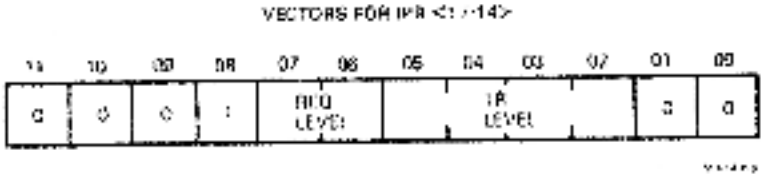


Figure 3-16 Vector Generation

3.6.5.8 ISR: TTL Register File Write Address – If the interrupt summary response data does not have any parity errors, the register file TTL write address is forced to 0011 (see Figure 3-14). The ISR data cycle enables the register data bus to the file info bus and enables writing the vector into the register file.

3.6.5.9 MBox Reads Vector – The MBox will set up the register file address and read the vector in the same manner as it would read any CPU read word.

3.6.6 Local Interrupt Vector

Reading the vector for a local interrupt is carried out in almost the same manner as a CPU read SBIA register. The MBox loads the command/address into the register file; the command/address is then transferred to the file data latch and to the command/address latch, as for a CPU read SBIA register.

The register address decode logic asserts REG ADR OK if the address is a valid SBIA register address (see Figure 3-11). PROM E163 will enable PROM E154 with LOCAL IPR SEL L (Figure 3-12) and the contents of the addressed PROM (E154) are gated to the register data bus (Figure 3-13).

The contents of PROM E154 is vector bits <07:00> (see Table 3-9). Bits <31:08> are forced to 0s by the zero fill tri-state buffers.

If the register address is valid (REG ADR OK, PROM E1) and there are no command/address parity errors, the vector is enabled to the file info bus and then written into the register file. The register file TTL write address is forced to 0011 (see Figure 3-14).

The MBox will read the vector from the register file as it would any CPU read word.

If there is a command/address parity error, the transaction is aborted. The MBox is notified by the assertion of ABUS CPU BUF ERROR.

3.7 QUADCLEAR

The purpose of the quadclear operation is to clear ECC errors in SBI memories. For the VAX 8600/8650 system, the quadclear operation is used extensively by microdiagnostics to perform DMA quadword loopback transfers.

The CPU initiates a quadclear operation by writing the SBIA quadclear register. The decoding of the quadclear register address allows the CPU-SBI state machine to control the quadclear operation. When the SBIA gains control of the SBI, the command/address is transferred to the SBI, followed by two SBI write data longwords, which contain all 0s. The address is the quadword aligned address of the quadword to be cleared.

The state machine sequences much like a CPU write SBI nexus register, except that there are two write data cycles instead of one. Also, the SBIA looks for confirmation for the extra cycle.

The MBox loads the register file with a command/address indicating a CPU write to the quadclear register. It will then load write data that contains a mask = 1111 and the quadword boundary address of the quadword to be cleared. The command/address will be transferred to the file data latch, and then to the command/address latch. The write data will be transferred from the register file to the file data latch, and then to the write data latch. When the address in the command/address latch is decoded by the address decode logic (see Figure 3-11), PROM E1 will assert SBIA VIA REG if the address is valid and the proper address. Recall that SBIA VIA REG was asserted by PROM E1 for an ISR also, but that was a CPU read of a vector register. This is a write, and it is the write bit in the command that directs the CPU-SBI state machine to the code for a quadclear.

If there is a command/address or data parity error, the state machine never leaves the idle state. ABUS CPU BUFF ERROR is asserted to inform the MBox of the error condition. If there are no parity errors, the state machine will leave the idle state to wait until the SBIA gains control of the SBI (CPU ARB OK).

The CPU-SBI state machine will assert CPU HOLD for two cycles, to cause the DC101 priority arbitration chips to assert TR00 for two SBI cycles to insure that the SBIA has control of the SBI for three bus cycles, the command/address cycle, and two write data cycles. During the third SBI cycle, arbitration for the bus will determine what nexus controls the bus for the following SBI cycle.

3.7.1 Quadclear Command/Address Cycle

The information held in the S-data assembly is transferred to the SBI transceivers in the following manner (see Figures 3-17 and 3-18).

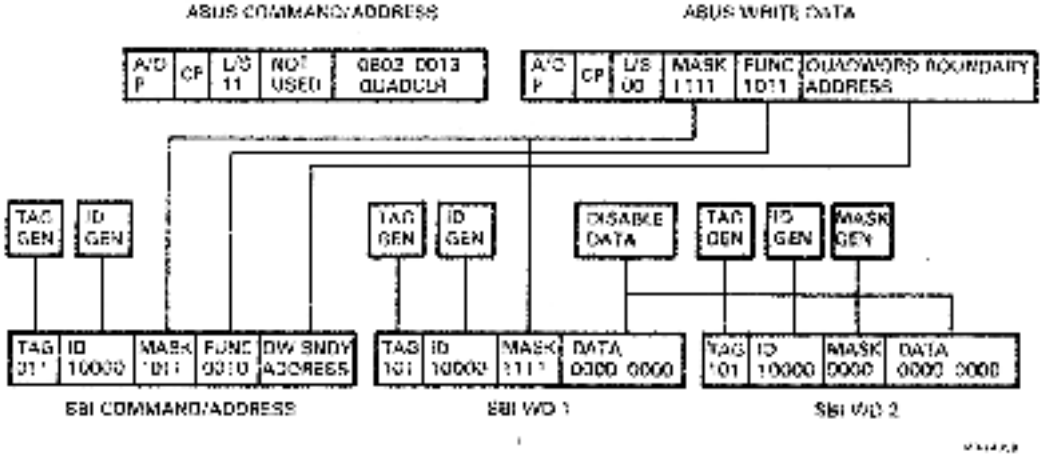


Figure 3-17 Quadclear Data Transfer to SBI

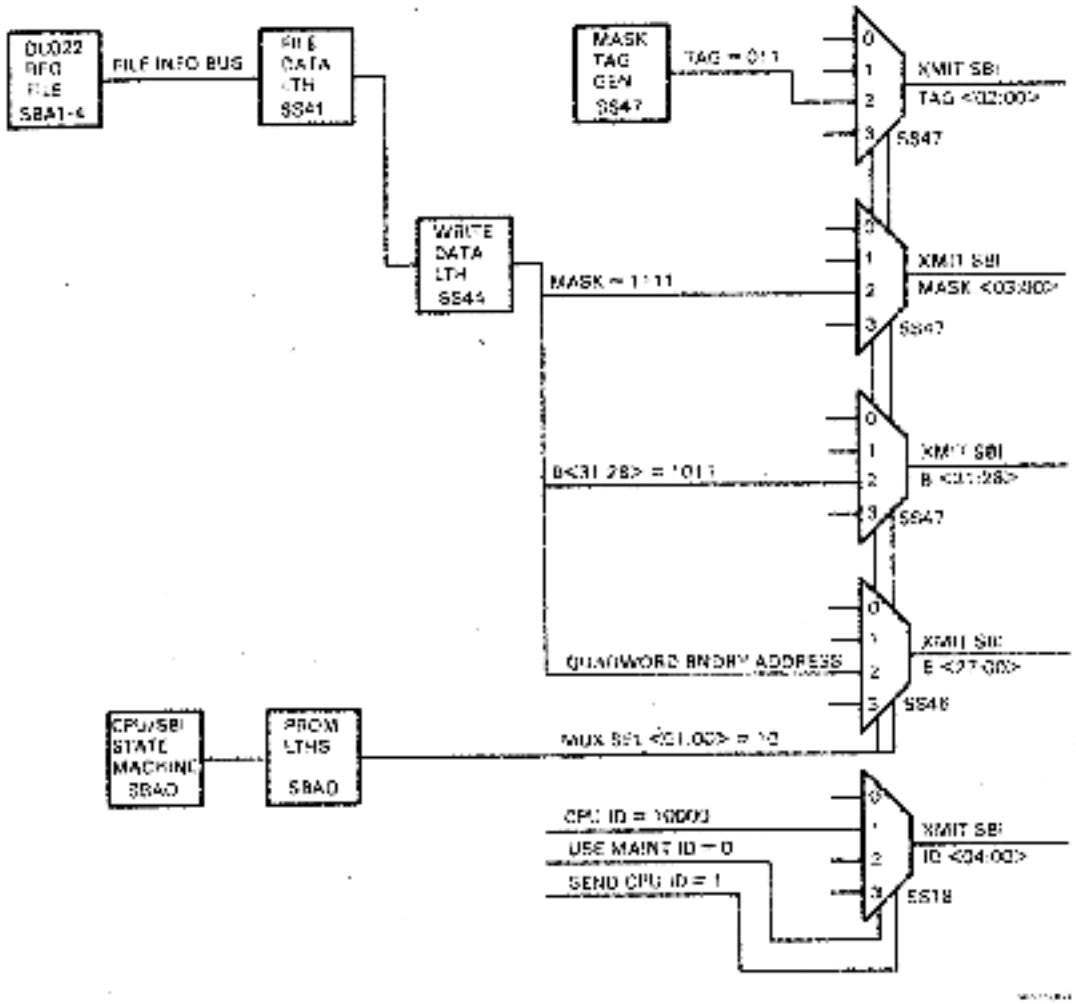


Figure 3-18 S-Data Assembly: Quadclear Command/Address Cycle

1. The quadword boundary address in the write data latch is transferred to SBI <27:00>.

2. Write data latch <31:28> contains the SBI function code, which for a quadclear will be 1011, extended write masked. These bits are multiplexed to SBI <31:28>. For diagnostics, a quadclear can be used to do quadword reads or writes. The command bits in the command/address word are not used for a quadclear.

3. The ABus write data mask, 1111, is multiplexed from the write data latch to SBI MASK <3:0>, also equal to 1111. This indicates a longword write.

4. The ID generator will provide the CPU's ID of 10000 to the SBI drivers.

5. A tag of 011, command/address, is provided by the tag generator.

6. SBI P0 is generated by XORing ID parity, forced to a logic 1, with parity over the L/S and MASK bits in the write data latch after it is inverted.

7. The data parity in the write data latch is inverted to provide P1 on the SBI.

3.7.2 Quadclear: Write Data Cycle 1

For the first write data cycle, the S-data assembly is enabled by the CPU-SBI state machine to transfer data to the SBI in the following way (see Figures 3-17 and 3-19).

1. The multiplexers that provide SBI <31:00> are disabled to insure that the data bits are all 0.
2. The mask bits in the write data latch are again transferred to the SBI mask bits.

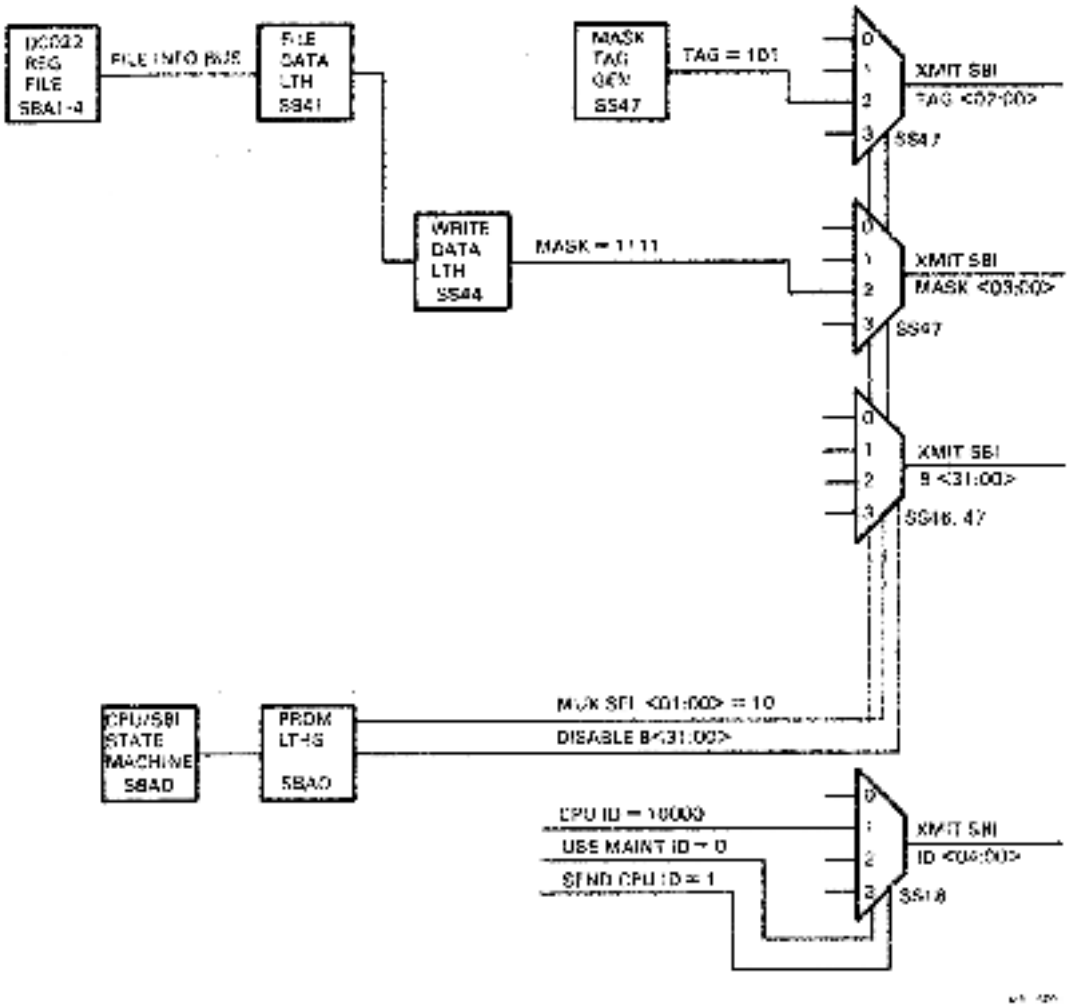


Figure 3-19 S-Data Assembly: Quadclear Write Data Cycle 1

3. The TAG generator provides a tag of 101 to indicate an SBI write data cycle.
4. The ID generator again provides the CPU ID of 10000.
5. SBI P0 is generated, as for the command/address cycle.
6. SBI P1 is forced to a logic 0 because the data field is all 0s.

3.7.3 Quadclear: Write Data Cycle 2/ACK 1

The S-data assembly is enabled by the CPU-SBI state machine in the following way (see Figures 3-17 and 3-20).

1. SBI <31:00> are again forced to 0 by the disabling of a multiplexer.

2. MASK <3:0> are also 0 because the multiplexers providing these bits are also disabled.

3. The TAG generator again provides a tag of 101, write data.

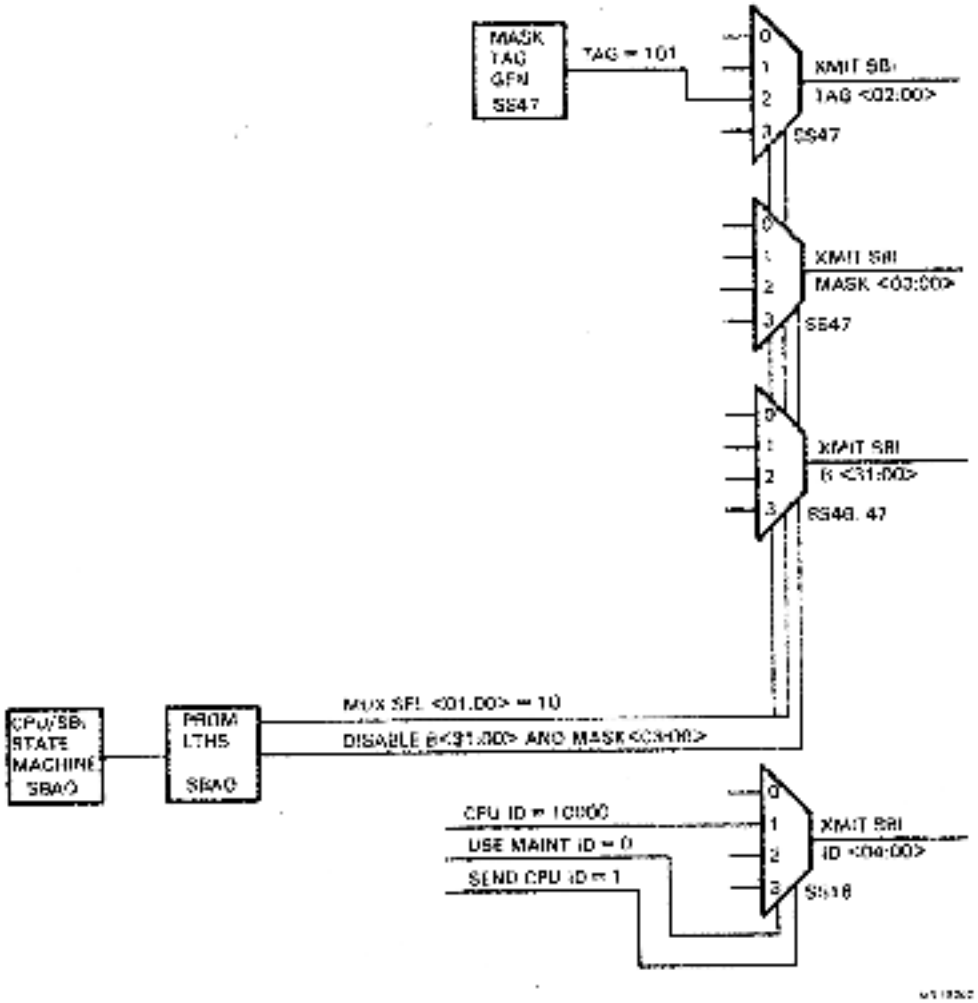


Figure 3-20 S-Data Assembly: Quadclear Write Data Cycle 2

4. The ID generator asserts the ID of 10000.

5. SBI P0 is provided by XORing a forced logic 1 for ID parity with a forced logic 0 for mask parity (the mask field is 0000).

6. SBI P1 is forced to a logic 0 because the data field is all 0s.

Write data cycle 2 is the second cycle after the command/address cycle so the SBIA should receive confirmation from the SBI nexus during this cycle. If there is no response from the nexus, or the confirmation code indicates that the nexus is busy, the state machine returns to the CPU ARB WAIT state and retransmits the command/address and write data when the SBIA can gain control of the SBI.

If the confirmation code indicates an error condition, the state machine enters the abort state, asserts ABUS CPU BUF ERROR, and returns to the idle state.

If an acknowledge is received, the state machine goes to the ACK2 cycle to monitor the confirmation bits for the second acknowledge.

3.7.4 Quadclear ACK2 Cycle

If the confirmation code is busy, error, or there is no response, the state machine returns to the ARB WAIT state to retransmit the command/address and write data.

If an acknowledge is received, the state machine looks for the third acknowledge.

3.7.5 Quadclear ACK3 Cycle

Again, if the confirmation is busy, error, or there is no response, the state machine returns to the ARB WAIT state to retransmit the command/address and write data.

If an acknowledge is received, the state machine goes to the COMMAND DONE state and asserts ABUS CPU BUF DONE to inform the MBox that the quadclear has been completed. The state machine then returns to the idle state.

3.7.6 Quadclear Timeout

If 512 SBI cycles elapse before the third acknowledge is received from the nexus, a timeout condition exists. The state machine goes to the abort state, asserts ABUS CPU BUF ERROR, then returns to the idle state.

3.8 QUADCLEAR FOR MICRODIAGNOSTICS

The quadclear operation is used by microdiagnostics, in conjunction with diagnostic control register <03> (FORCE QUAD DATA), to loop data back in the SBIA. (See Paragraph 3.14.4 for a description of the diagnostic control register.)

3.9 UNJAM

An UNJAM is issued to the SBI to clear a hung system. It is initiated by the CPU writing to the SBIA unjam register, address 2X08 0048. The unjam sequence consists of 16 cycles of SBI HOLD, 16 cycles of SBI HOLD and SBI UNJAM, then 16 more cycles of SBI HOLD. When the address for the unjam register is decoded, a special hardware sequencer initiates the unjam sequence.

The unjam hardware sequencer (see Figure 3-21) consists of two 4-bit binary counters. The carry from the first counter, after 16 SBI cycles, increments the second counter.

The MBox transfers a command/address to the SBIA register file. Because good parity is required for a data word, the MBox also writes a data word into the register file. The data is not needed except to provide a parity check. The command indicates a write and the address is for the unjam register. The register address decode logic addresses PROM E1, which provides an output of 0110. REG ADR OK and UNJAM REG are asserted.

UNJAM REG asserts START UNJAM if the following conditions (see Figure 3-12) are true.

1. No command/address control parity error
2. No command/address A/D parity error
3. No control parity error over the data word
4. No A/D parity error over the data word
5. The command is for a CPU write
6. REG ADR OK is asserted.

The assertion of START UNJAM loads both binary counters. Counter E96 is loaded with 0000, while counter E78 is preset to 1101. Every 16 SBI cycles the carry from counter E96 increments counter E78. Only four of the states of E78 are significant (see Table 3-11).

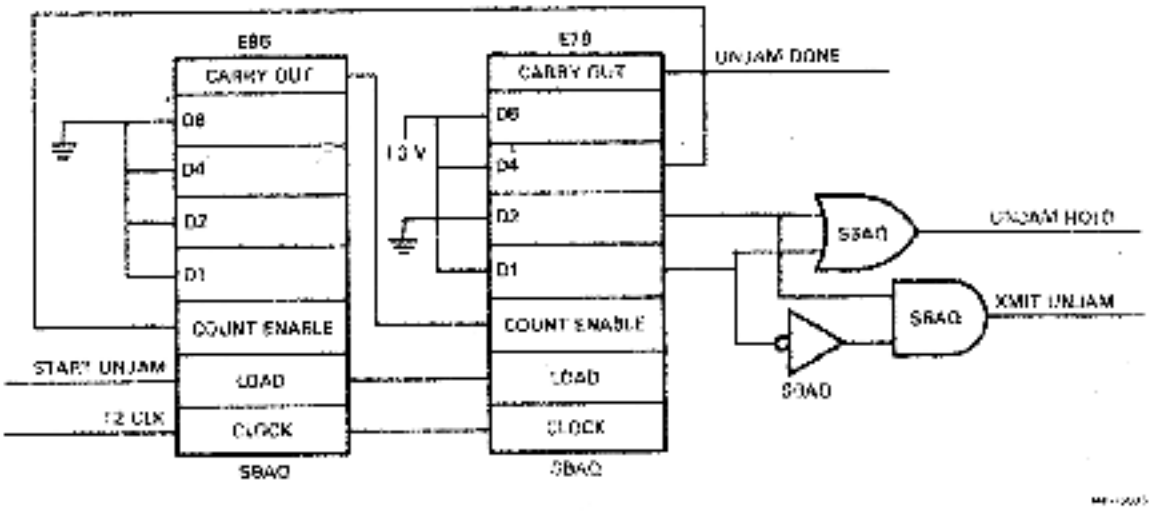


Figure 3-21 Unjam Sequencer

Table 3-11 Unjam Sequencer States

Counter E78 Output	Legend
1101	Enable E96 to count Assert SBI HOLD
1110	Enable E96 to count Assert SBI HOLD Assert SBI UNJAM
1111	Enable E96 to count Assert SBI HOLD
0000	Unjam done

When E78 is loaded, E96 is enabled to count. UNJAM HOLD asserts SBI HOLD for 16 SBI cycles. When E78 is incremented the first time, XMIT UNJAM causes SBI UNJAM to be asserted for 16 SBI cycles. UNJAM HOLD is still asserted so that SBI HOLD remains asserted. When E78 is incremented the second time, XMIT UNJAM goes low so SBI UNJAM is negated. SBI HOLD remains asserted for another 16 SBI cycles. When E78 is incremented for the final time, UNJAM DONE is asserted. It asserts ABUS CPU BUF DONE to alert the MBox of the completion of the unjam sequence.

3.10 DMA OVERVIEW AND BUFFER CONTROL

The CPU initiates a DMA transaction by loading SBI nexus registers. Once the SBI nexus has been programmed by the CPU, the nexus arbitrates for control of the SBI and transmits the command/address. If it is a DMA write transaction, the write data follows the command/address on the next successive SBI cycle(s). For a DMA read, after transmission and acknowledgment of the command/address, the nexus waits for the return of the read data.

The SBIA must recognize the command/address, and, for a write, the write data, and must transfer the information to the register file so that the MBox can read it. For a read, the SBIA transfers only a command/address to the register file. The MBox reads the command/address from the register file, then reads cache or memory, and transfers the read data to the SBIA register file. The SBIA gates the read data to the SBI for transfer to the nexus.

Before looking at the DMA transactions, the DMA buffer control and register file addressing must be investigated.

3.10.1 DMA Buffer Control

For noninterlocked DMA transactions or interlocked writes, there are only three DMA transaction buffers in the SBIA register file. Therefore, any command/address that is written must not write over an uncompleted DMA request. For a DMA interlock read, only one transaction buffer is needed because only one interlock read can be in process at one time.

Figure 3-22, a flowchart of noninterlocked DMA and DMA interlock write transactions, assumes no error conditions. (Error conditions will be mentioned in the DMA write or DMA read detailed description, and covered in detail in Paragraphs 3.11 and 3.12.) A description of the flowchart follows.

1. When no DMA transactions are occurring, the SBIA DMA buffer control is in the idle state, and the number of commands queued and commands in progress is 0.

2. The SBIA monitors the SBI transceivers, looking for a command/address tag (011). When a command/address tag is received, if there are no parity errors and the address is within the bounds of memory address (the address is less than the address contained in the configuration register), the SBIA checks for a valid function code.

3. If, upon receiving a valid function code, there are already two commands queued or three commands in progress, the SBIA transmits BUS SBI CONF <1:0> = 10 (BUSY), to inform the nexus that the SBIA cannot accommodate the request at that time. The nexus retransmits the command/address (and write data if for a DMA write) when it is able to regain control of the SBI.

4. If the SBIA is not busy, the SBIA loads the command/address (and write data for a DMA write) into a transaction buffer in the register file.

5. When the command/address (and write data for a DMA write) have been loaded into the register file, the following events take place.

   a. Number of commands queued is incremented by 1.
   b. Number of commands in progress is incremented by 1.
   c. The SBIA sends a DMA request to the MBox by asserting SB ABUS IOA REQUEST [N].

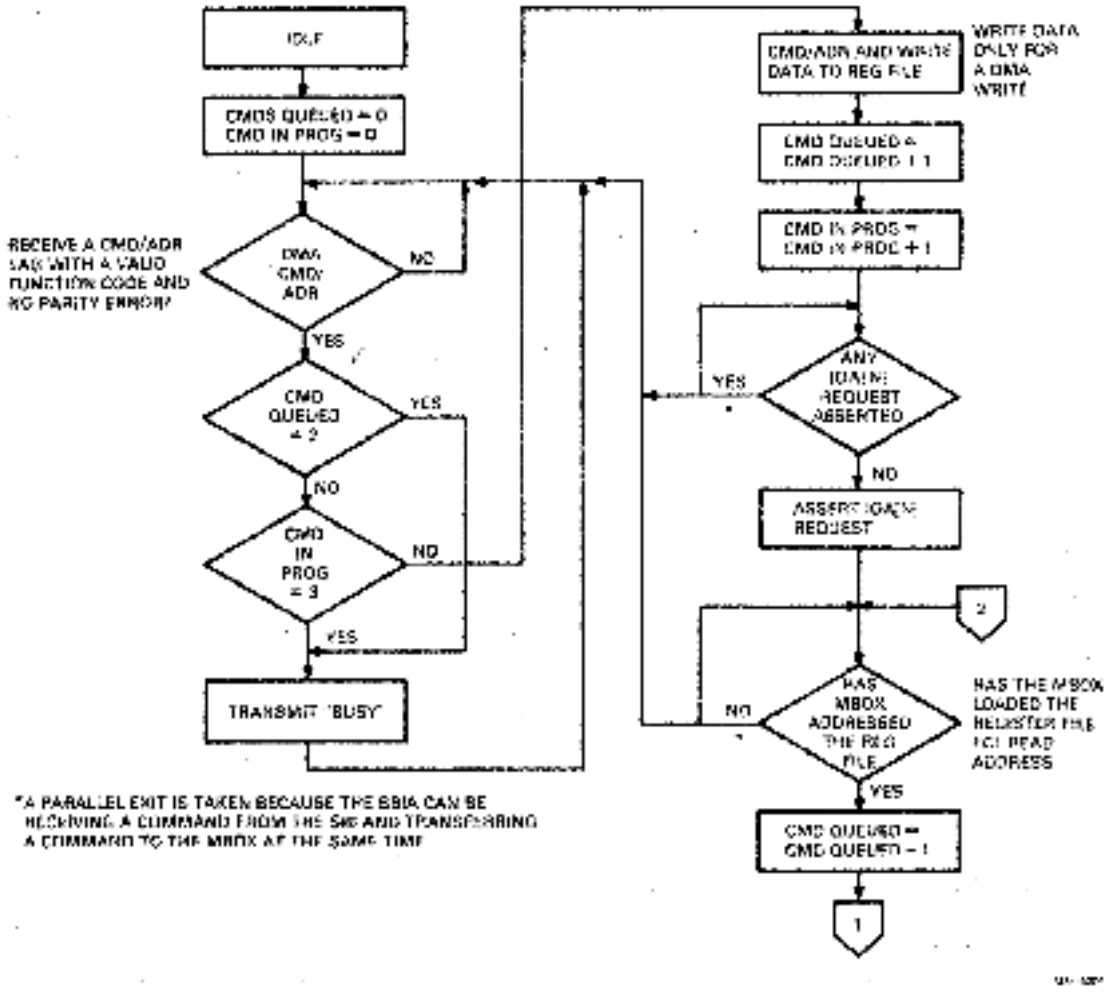


Figure 3-22 DMA Buffer Control (Sheet 1 of 2)

6. The SBIA waits for the MBox to load the ECL register file read address, which enables the MBox to read the command/address (and write data for a DMA write). The SBIA can be receiving DMA requests from the SBI and transferring DMA requests to the MBox at the same time, so the flowchart shows a parallel exit.

7. When the MBox has loaded the ECL register file read address, an indication that a command will be processed, the number of commands queued is decremented by 1.

8. If it is a DMA write, the MBox reads the command/address and the write data.

9. If it is a DMA read, the MBox reads the command/address, and the addressed data from cache or memory, and then writes the read data into the register file.

10. The SBIA waits for the MBox to assert MCC ABUS DMA DONE [N], which for a write indicates that the operation is finished, but a DMA read is not finished.

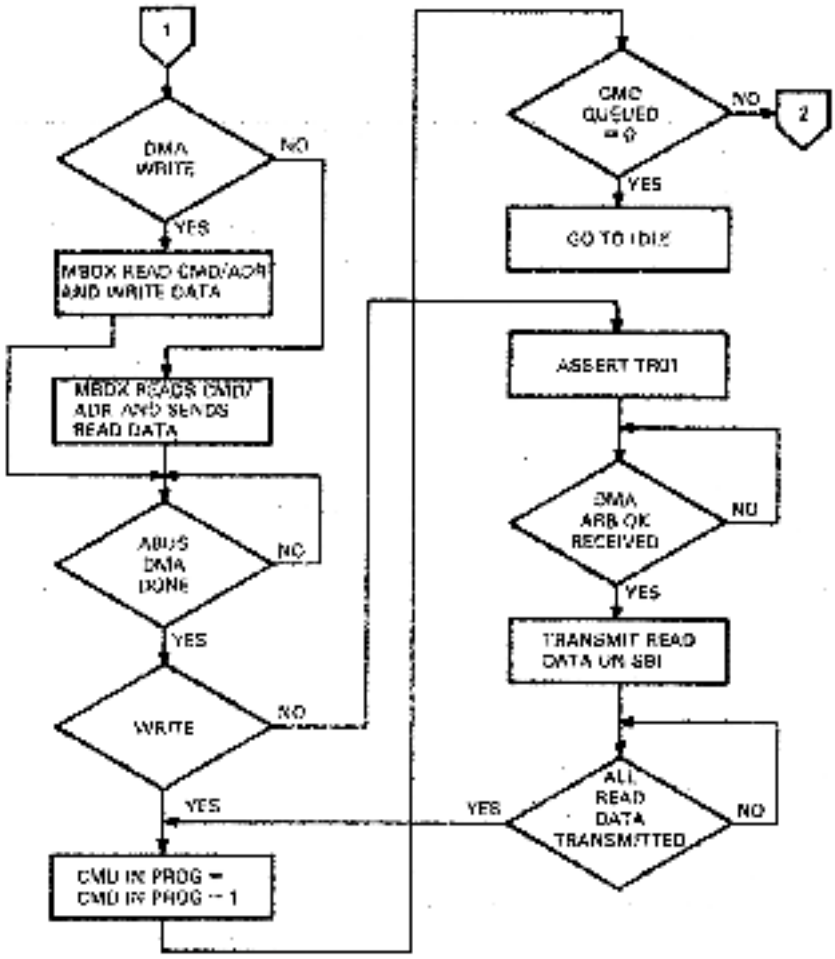


Figure 3-22 DMA Buffer Control (Sheet 2 of 2)

11. With a DMA write, the SBIA, upon the reception of DMA done or DMA error, reduces the number of commands in progress by 1 to free one of the DMA transaction buffers. If there are more commands queued, they will be attended to; if not, the DMA goes to the idle state.

12. For the DMA read, MCC ABUS DMA DONE [N] simply indicates that the read data is in the register file and must be transferred to the SBI. Upon receipt of DMA done, the SBIA requests the SBI at TR01, the TR level for DMA transactions.

13. When the SBIA receives DMA ARB OK, the read data is removed from the register file and transmitted on the SBI. For extended reads, two longwords are transferred.

14. When all read data has been transmitted on the SBI, the number of commands in progress will be decreased by 1. If there are more commands queued, another IOA request must be asserted to request MBox service; otherwise, the DMA goes to the idle state.

3.10.2 DMA Transaction Buffer Selection

Whenever the SBIA receives a DMA command/address from an SBI nexus, the command/address, or command/address and write data for a DMA write, must be loaded into an empty transaction buffer. Figure 3-22 shows that, if there are two DMA commands queued or three commands in progress, the SBIA would transmit BUS SBI CONF<1:0> = 10, to indicate that the SBIA is busy.

For a DMA write, a command is in progress if the command/address and associated write data have been loaded into the register file. A command is queued if it is in progress and if the MBox has enabled reading the command/address from the register file (that is, the MBox is acting on the command). The command in progress signals are used to determine the next transaction buffer to be used.

When the first DMA command/address is received, there are no commands in progress. DMA transaction buffer A is loaded with the command. Transaction buffer DMAA now has a command in progress.

If another DMA command/address is received, because DMAA has a command in progress, DMA transaction buffer B will be used. If the MBox has not started to read the command/address from transaction buffer A, there are now two commands queued. No more DMA command/addresses, except for a DMA interlock read, are accepted. The SBIA is busy, and BUS SBI CONF<1:0> = 10, is transmitted on the SBI.

When the MBox acts on DMA transaction buffer A, another DMA command/address is accepted and placed in DMA transaction buffer C. There are now three commands in progress; the SBIA is busy; and again BUS SBI CONF<1:0> = 10 is transmitted on the SBI.

Transaction buffer A has priority. If it is empty, it is loaded, regardless of the other two transaction buffers. If A is full, transaction buffer B is used. If both A and B are full, then C is used.

The logic that determines which transaction buffer to use also controls the upper two bits of the register file address. This will be explained in the next paragraph.

3.11 DMA WRITE

The DMA write must be set up by CPU writes to the SBI nexus. When the proper registers have been loaded, the SBI nexus carries out the DMA write transaction. When the nexus gains control of the SBI, it transmits an SBI command/address followed by the write data to the SBIA, which loads the command/address into a transaction buffer in the register file. The SBIA requests MBox service by asserting SB ABUS IOA REQUEST [N].

The MBox, after arbitration, in response to the IOA request, reads the command/address and write data from the SBIA register file. The MBox then stores the write data in cache or memory.

3.11.1 DMA Write: Command/Address Reception

The SBIA, like all SBI devices, is latching the SBI transceivers every --T2. If TAG <2:0> = 011, the information in the transceivers is a command/address, and the command is loaded into the command register. The command register is loaded with the following information.

1. REC SBI <31>, to indicate an extended, or quadword transfer
2. REC SBI <30>, to indicate an interlocked DMA transfer
3. REC SBI <29>, to indicate that the command is for a write
4. CMD/ADR MASKED, a NAND condition of all the SBI mask bits. If any SBI mask bit is not set, CMD/ADR MASKED is set. If all mask bits are set, it is not a masked operation and CMD/ADR MASKED will not be set.

The contents of the command register are held until the next command/address is received in the SBI transceivers. The contents of the command register are used later in the DMA to address the command PROM.

The upper bits of the received address, REC SBI B<27:18>, are compared with the contents of the configuration register to insure that the DMA address is within the bounds of memory. The results of the address comparison will be used to enable a function check.

Each SBI transceiver generates parity over the four bits it receives. All of the generated parity bits are combined and compared to SBI parity to check for an SBI parity error.

If there are no SBI parity errors, the address is within bounds, and if the tag indicates a command/address, REC SBI B<31:28> are decoded to check for valid functions. If there are not two commands queued or three commands in progress, the command address is written into the register file. (See Appendix B, SBI Protocol, for valid SBI command/address functions.)

If the function (REC SBI B<31:28>) is not valid the SBIA transmits an SBI error confirmation to notify the nexus of the error condition.

If the SBIA detects an SBI parity error, it asserts SBI FAULT to notify all nexus to latch their error registers. The SBIA, upon receiving SBI FAULT, sets the fault latch, and if the fault error is enabled by SBI FAULT REG<18>, the CPU is interrupted.

3.11.2 DMA Write: Register File TTL Write Address Generation

The register file cannot be loaded in a straightforward manner. On the SBI, the mask bits precede the write data by 1 SBI cycle, but on the ABus, the mask bits are transferred with the write data. Therefore, when the command/address is loaded into the register file, the mask bits that accompanied the SBI command/address must be loaded into the register file location for the first write data longword. When the first write data longword is loaded into the register file, the SBI mask bits that accompanied the first write data longword must be loaded into the register file location for the second write data longword. When the second write data longword is loaded into the register file, the ABus command is loaded into the register file location for the command/address. (See Figure 3-23 and Tables 3-12 and 3-13.)

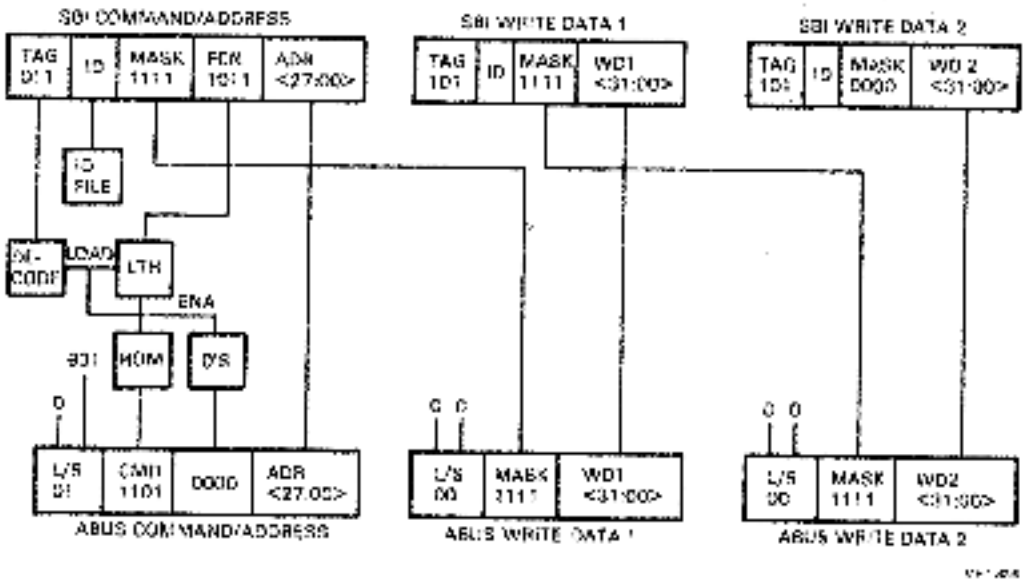


Figure 3-23 DMA Quadword Write Data Transfer

The DMA write SBI information is written into the register file as follows.

1. When the command/address is written into the register file, all bits but the command/mask are loaded into XX00, where XX is determined by the transaction buffers that have commands in progress. The mask is loaded into XX01, the location for write data 1.

2. When write data 1 is written into the register file, all bits but the command/mask are loaded into XX01. The mask is loaded into XX10, the location for write data 2.

3. When write data 2 is written into the register file, all bits but the command are written into XX10. The ABus command is written into XX00, the location for the command/address.

Table 3-12 Register File TTL Write Address <03:02>

Command in Progress	TTL FILE ADR <3:2>	TTL CMD/MSK ADR <3:2>
Not DMAA	0 1	0 1
DMAA	1 0	1 0
DMAA and DMAB	1 1	1 1

Table 3-13 Register File TTL Write Address <01:00>

SBI Cycle	TTL FILE ADR <1:0>	TTL CMD/MSK ADR <1:0>
Command/Address	0 0	0 1
Write Data 1	0 1	1 0
Write Data 2	1 0	0 0

3.11.3 DMA Write: A-Data Assembly Command/Address Transfer

The command address is transferred to the register file by the A-data assembly, according to Figure 3-24 in the following manner (see also Figure 3-23).

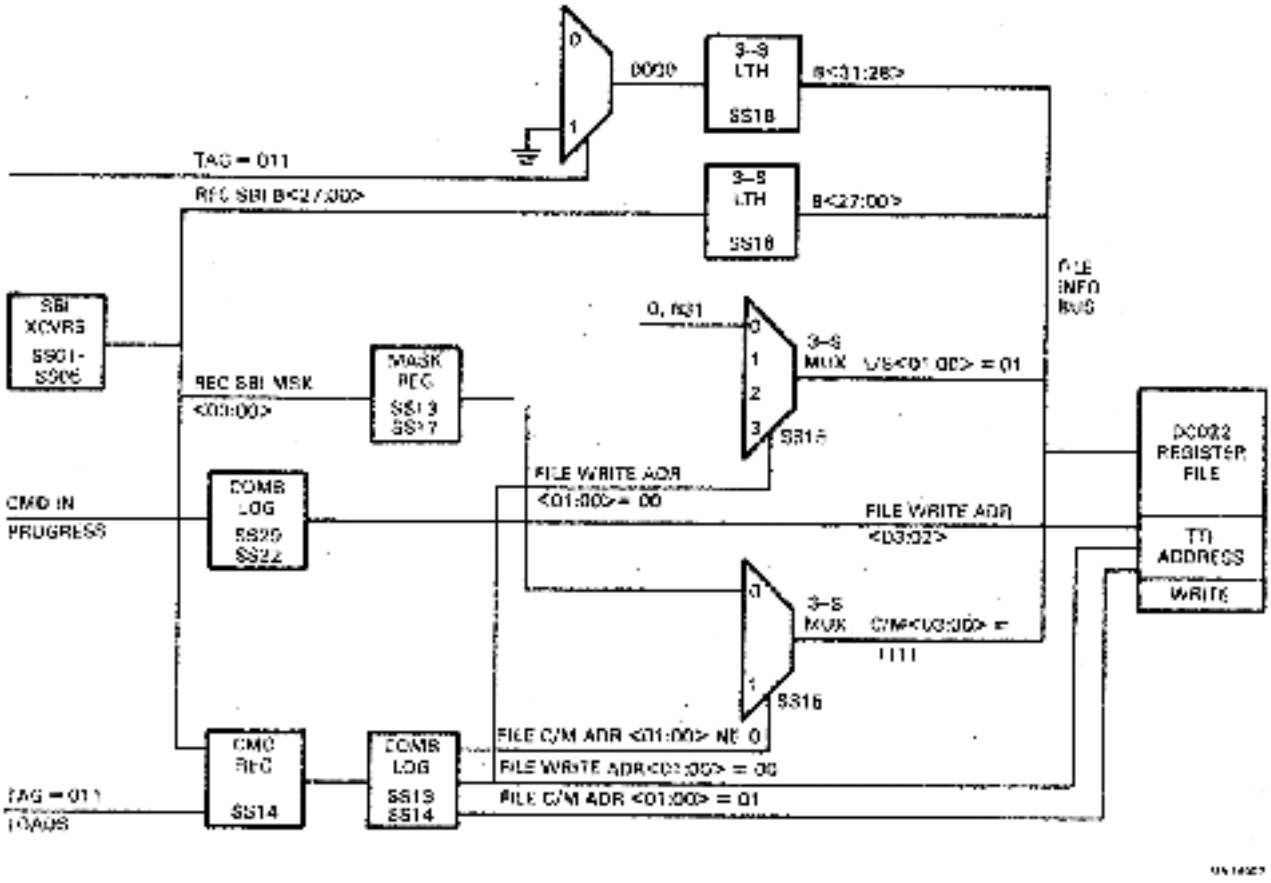


Figure 3-24 DMA Write, A-Data Assembly Command/Address Transfer

1. If the SBI tag is for a command/address, the command register is loaded (see Paragraph 3.14.3.1). It is held until the next command/address tag. The command register is used during the last write data cycle to address the command PROM, which provides a conversion from the SBI function to the ABus command.

2. The command address tag will enable input 1 to multiplexers with that input grounded. This will set file info bus B<31:00> to 0000.

3. REC SBI B<27:00> are latched in tri-state latches to be driven to file info bus B<27:00>.

4. The set of multiplexers that provides L/S <1:0> is enabled by FILE WRITE ADR <1:0>, which is 00 for the command/address. The inputs are 0 and REC SBI B<31>. Bit 31 is set for extended writes and provides a length/status field of 01, which on the ABus indicates a quadword transfer.

5. REC SBI MASK<3:0> are latched and enabled to the file info bus as C/M <3:0> because the enabling input C/M ADR <1:0> does not equal 0.

6. The ABus command always has an odd number of 1s, so control parity depends entirely on what the L/S field is. If the transfer is for an extended write, REC SBI B<31> is asserted, which makes the L/S field equal to 01. If REC SBI B<31> is not asserted, then the L/S field equals 00. Therefore, if REC SBI B<31> is asserted, then FILE INFO CNTRL PTY is also asserted.

7. The SBI function always has an odd number of 1s. If the address also has an odd number of 1s, the total number will be even and SBI P1 will be a 0. On the other hand, if the address has an even number of 1s, the total number will be odd and SBI P1 will be asserted. When the command address is transferred to the register file, the only concern is for the address bits, bits <27:00>. Bits <31:28> are forced to 0000. Therefore, SBI P1 need only be complemented to provide proper parity to the register file, FILE INFO A/D PTY.

The preceding information is held in latches and is enabled to the file info bus when an acknowledge is enabled for the command/address word. This same enabling signal is delayed and used to generate the register file write pulse.

If the command/address is for an extended write, the address boundary is constrained to be a quadword boundary; B<00> must be 0. If SBI B<00> is not 0, it will be cleared. If B<00> has to be cleared, the A/D parity bit is toggled to correct the parity.

3.11.4 DMA Write A-Data Assembly Transfer of Write Data 1

The SBI nexus transfers the first write data on the SBI cycle following the command/address cycle. The data will be transferred by the A-data assembly to the file info bus as follows (see Figures 3-25 and 3-23).

1. REC SBI B<31:28> are multiplexed to FILE INFO BUS B<31:28> because the tag is not 011, command/address. To be valid write data, the tag must be 101.

2. REC SBI B<27:00> are latched and are driven to FILE INFO BUS B<27:00>.

3. The multiplexer that provides FILE INFO BUS L/S <1:0> has input 01 enabled because FILE WRITE ADR <1:0> equals 01. This input is grounded for both bits.

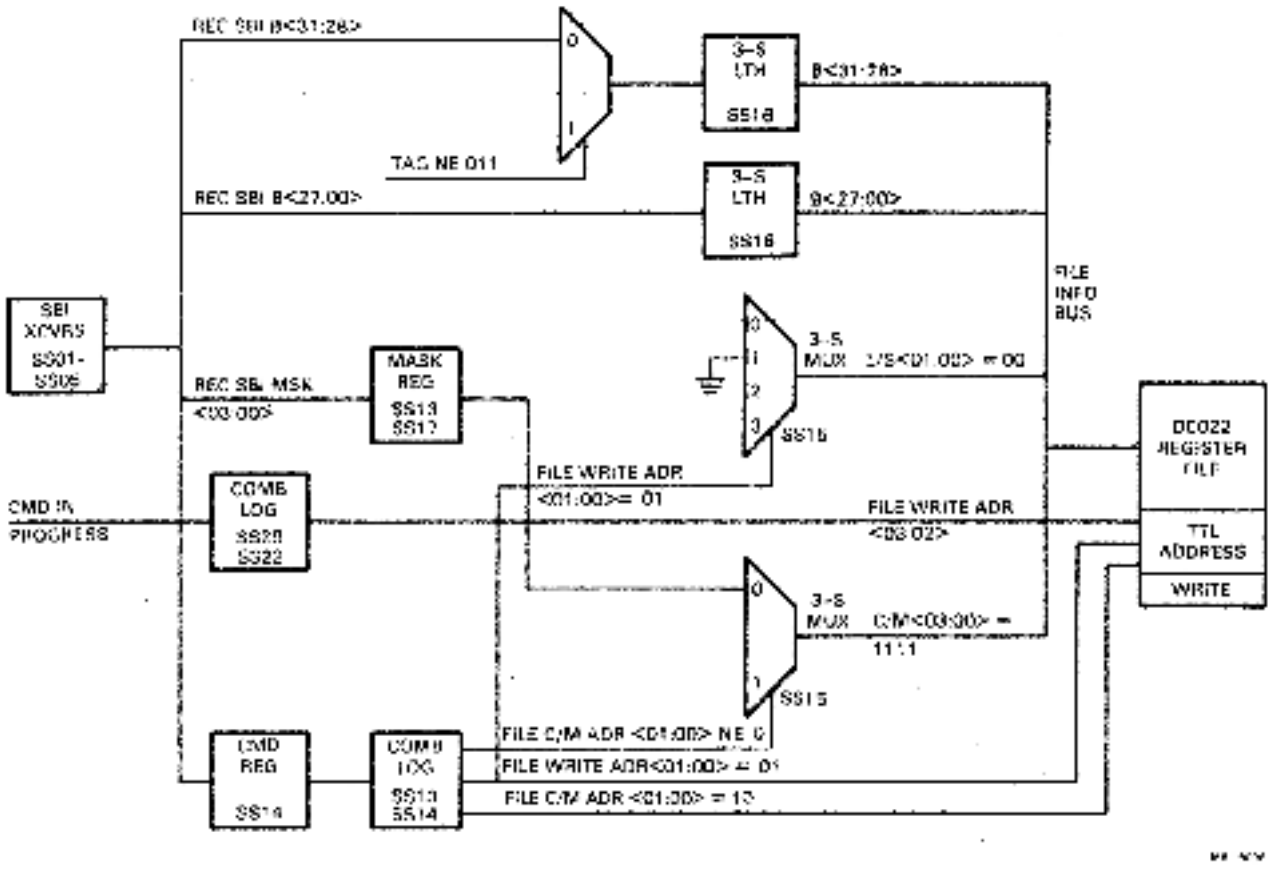


Figure 3-25 DMA Write: A-Data Assembly Transfer of Write Data 1

4. FILE C/M ADR <1:0> does not equal 00, therefore REC SBI MASK<3:0> is multiplexed to FILE INFO BUS C/M <3:0>.

5. The L/S bits will always equal 00 for the ABus write data cycles, so FILE INFO CTR PTY depends only on the mask bits. The SBI transceivers generate an even parity bit over the mask bits. However, the mask bits arrive in the SBIA one SBI cycle before the parity bits are written into the register file. Mask parity is latched and held for one SBI cycle, and then inverted to be used as FILE INFO CTR PTY.

6. REC SBI P1, parity over REC SBI B<31:00>, is inverted to provide parity over FILE INFO BUS B<31:00>, FILE INFO A/D PTY.

The tri-state multiplexers and tri-state latches are enabled to place the write data on the file info bus when an acknowledge is enabled for the write data word. The enabling signal is delayed to provide the register file write pulse.

3.11.5 DMA Write: A-Data Assembly Transfer of Write Data 2

Write data is transferred to the file info bus according to Figures 3-26 and 3-23 as follows.

1. FILE INFO BUS B<31:00> is transferred as with write data 1.

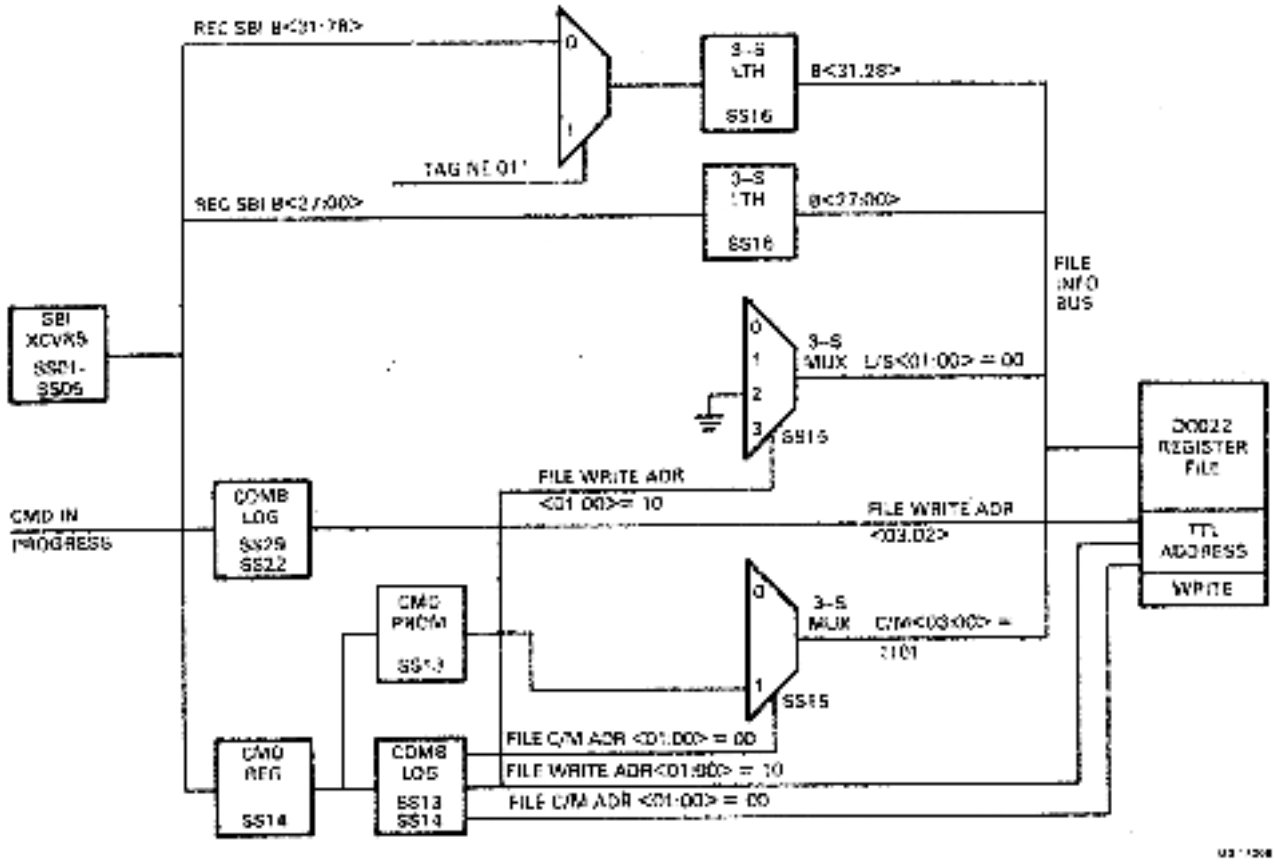


Figure 3-26 DMA Write: A-Data Assembly Transfer of Write Data 2

2. FILE INFO BUS L/S <1:0> is again 00. The only difference is that multiplexer input 2 is enabled because FILE WRITE ADR <1:0> = 10.

3. The contents of the command register address the command PROM, which will provide the ABus command on FILE INFO BUS C/M <3:0>. The command is routed through multiplexer input 1 because FILE C/M ADR <1:0> = 00. This will enable placing the command in the register file with the command/address.

4. FILE INFO A/D PTY is the same as for write data 1.

5. FILE INFO CTR PTY is the same as for write data 1.

The tri-state multiplexers and tri-state latches are enabled as for write data 1.

3.11.6 DMA Write: Acknowledge

The SBIA transmits an acknowledge on the SBI, two SBI cycles after receiving the command/address and write data longwords. An acknowledge is transmitted as SBI CONF <1:0> = 01 for the command address if the following conditions exist.

1. The tag equals 011, command/address.
2. There are no parity errors.
3. The address is within the bounds of memory as determined by the configuration register.
4. The function is a valid function.
5. There is no interlock sequence fault.

An acknowledge is transmitted for the write data longwords if the following conditions exist.

1. The command/address was a write function (expecting write data).
2. The tag = 101, write data.
3. There are no write data parity errors.

3.11.7 DMA Write: Sending IOA Request to the MBox

At approximately the same time that the second write data longword is being written into the register file, the circuitry that enables sending the acknowledge queues up the DMA requests by setting a command ready flip-flop. If no other DMA request has issued an IOA request to the MBox, an IOA request is asserted as SB ABUS IOA REQUEST [N]. If another DMA request has issued an IOA request, the present request remains queued until the first request is satisfied, at which time the request is honored.

3.11.8 DMA Write: MBox Reads the Register File

The MBox, in response to the IOA request, reads the register file to determine what it is expected to do. If the MBox were to wait until it received the command/address and then branch on the command, valuable time would be lost. To increase response time, as soon as the MBox selects the SBIA with MCC ABUS IOA SELECT [N], ABUS WR CMD and ABUS MSKED CMD are gated to the MBox. The MBox microcode is able to branch on these conditions without waiting for the command/address to be decoded.

The MBox reads the command/address and both of the write data longwords from the register file in much the same manner as for CPU read data (see Figure 3-27).

After the MBox has received and arbitrated the IOA request, it selects the SBIA with MCC ABUS IOA SELECT [N]. MCC ABUS ADDRS CTRL <1:0> = 00 enables loading the register file ECL read address. Address bits <1:0> are selected as 00 because MCC ABUS CPU BUF SEL is not asserted (Table 3-1). Address bits <3:2> are selected according to the transaction buffer that was queued up to request DMA service (Table 3-14). Because MCC ABUS MBOX OUT is not asserted, data is read from the register file.

On the ABus cycle following the loading of the address, the MBox reads the command address and drops MCC ABUS ADDRS CTRL <1> (asserted low), which causes the register file address to increment to XX01, the location for the first write data. On the next ABus cycle, write data 1 is gated to the MBox. The register file address is incremented again, this time to XX10, the address for the second write data. On the following ABus cycle, write data 2 is gated to the MBox. The MBox then drops MCC ABUS IOA SELECT [N] and MCC ABUS ADDRS CTRL <0>.

The MBox monitors the command/address for the cache/memory address, and stores the write data. If there are no errors, the MBox informs the SBIA by the assertion of MCC ABUS DMA DONE [N]. The reception of DMA DONE allows the SBIA to free the DMA transaction buffer that was tied up during the transaction.

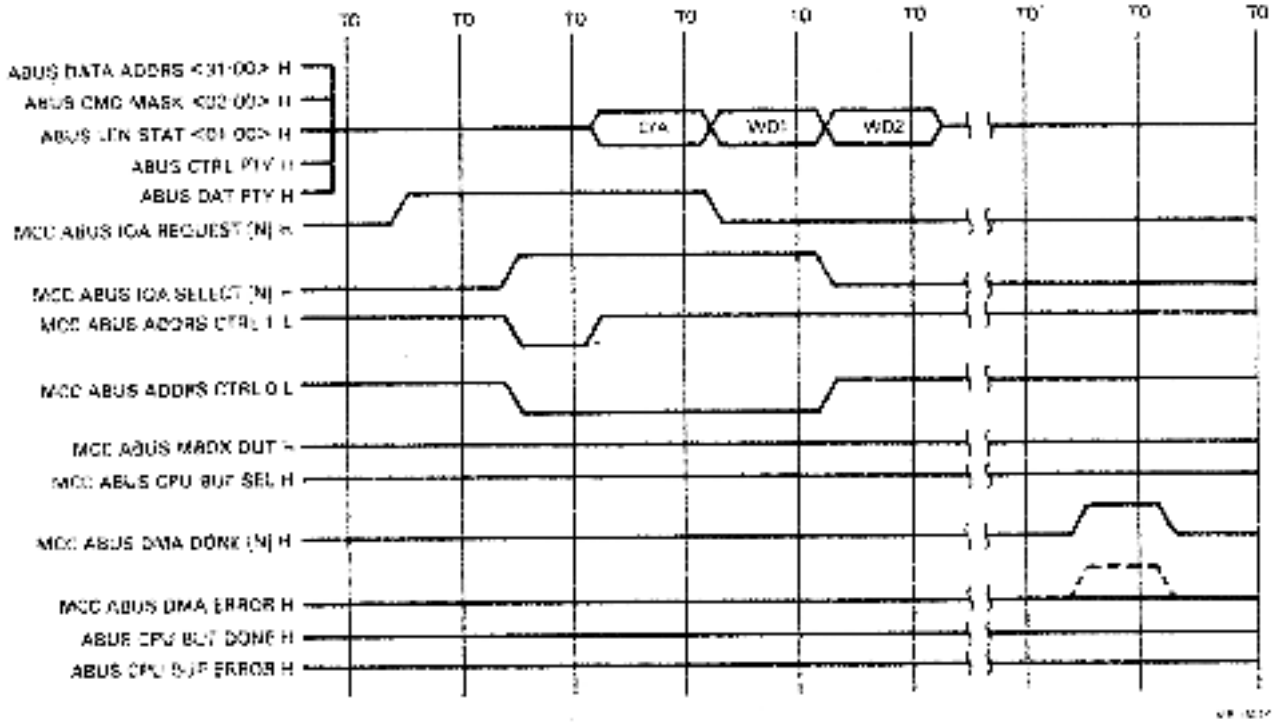


Figure 3-27 DMA Quadword Write, ABus Protocol

Table 3-14 Register File ECL Read Address <03:02>

Transaction Buffer Request	ECL FILE ADR <03:02>
DMAA	0 1
DMAB	1 0
DMAC	1 1
DMAI	0 0

If there is an error, the MBox responds with MCC ABUS DMA ERROR. The reception of the DMA error frees up the DMA transaction buffer as the DMA DONE would have, and also generates an interrupt.

3.12 DMA READ

Like the DMA write, the DMA read must be set up by CPU writes to the SBI nexus. When the proper registers have been loaded, the SBI nexus carries out the DMA read transaction. When the nexus gains control of the SBI, it transmits an SBI command/address to the SBIA, which loads the command/address into a transaction buffer in the register file. The SBIA requests MBox service by asserting SB ABUS IOA REQUEST [N].

The MBox, after arbitration, in response to the IOA request, reads the command/address word from the SBIA register file. Because the command is for a read, the MBox obtains the data from cache or memory and places the read data in the SBIA register file, in the same DMA transaction buffer that holds the command/address. When the SBIA can get control of the SBI, it transfers the read data from the register file to the SBI, to be consumed by the nexus that originated the DMA transaction.

3.12.1 DMA Read: Command/Address Reception

If the function (REC SBI B<31:28>) is not valid, the SBIA transmits an SBI error confirmation to notify the nexus of the error condition.

If the SBIA detects an SBI parity error, it asserts SBI FAULT to notify all nexus to latch their error registers. The SBIA, on receiving SBI FAULT, sets the fault latch, and if the fault error is enabled by SBI FAULT REG<18>, the CPU is interrupted. The command/address is received in the same manner as for a DMA write. The errors on command/address reception, SBI parity error, and invalid function are also the same.

The two main differences between the DMA write and DMA read are as follows (see Paragraph 3.11.1).

1. When the function is loaded into the command register, REC SBI <29> is not set because the command is for a read.

2. The contents of the command register are used to address the command PROM, but there is no delay in transferring the command to the register file as with a DMA write.

3.12.2 DMA Read: Register File TTL Write Address Generation

The most significant bits of the register file TTL write address, TTL FILE ADR <03:02>, are generated in the same manner as for a DMA write. These bits depend on the DMA transaction buffer that is going to be used (see Table 3-12). With a DMA read, the mask bits do not have to be manipulated as with a DMA write, and there is no write data to load into the register file. Because the only data loaded into the register file is the command/address, the least significant bits are 00, the location for the command/address for any DMA transaction buffer.

3.12.3 DMA Read: A-Data Assembly Command/Address Transfer

If the conditions described in Paragraph 3.9.1, DMA Buffer Control, are met, the command/address is transferred to the file info bus according to Figures 3-28 and 3-29. The SBI command/address is routed from the SBI transceivers to the A-data assembly, and to the register file in the following manner.

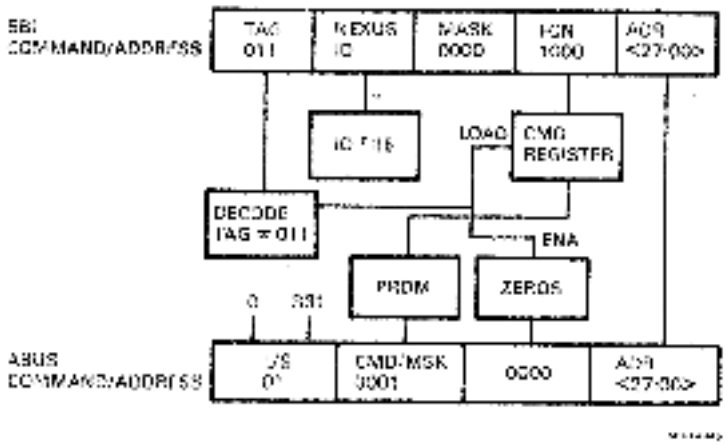


Figure 3-28 DMA Quadword Read: Command/Address Transfer

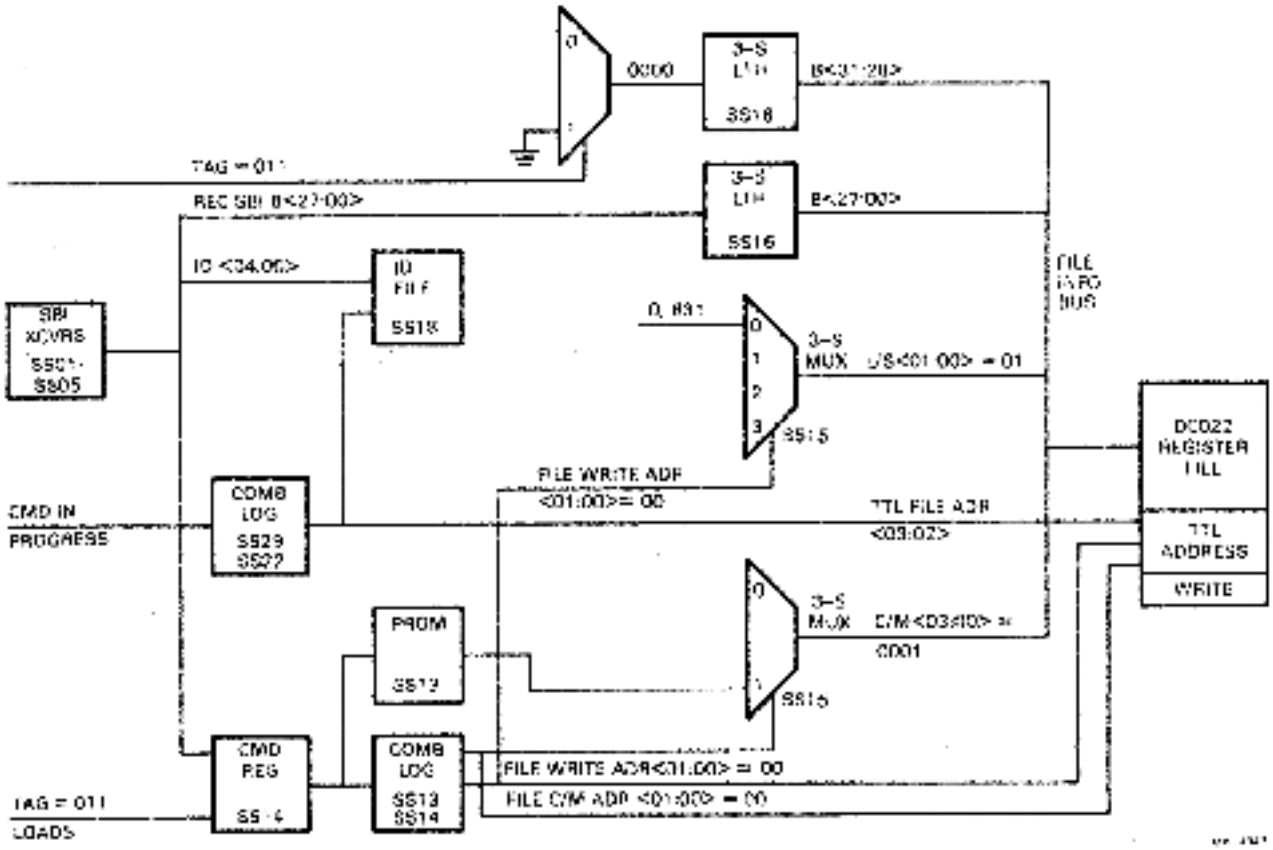


Figure 3-29 DMA Quadword Read: C/A Transfer to DC022

1. REC SBI <27:00> is transferred directly to FILE INFO B<27:00> as the read address.

2. The decoding of command/address tag, 011, with no parity errors, forces FILE INFO BUS <31:28> to 0000.

3. The SBI function, 1000, will be latched in the command register when the command/address tag, 011, is decoded. The contents of the command register are used to address the PROM, whose contents will be directed to FILE INFO C/M<3:0>. In this case, because the SBI function is an extended read, the PROM output is 0001, an ABus read command.

4. FILE INFO L/S<1> is forced to a 0 and FILE INFO L/S<0> receives REC SBI B<31>, a logic 1. A L/S of 01 indicates a quadword data transfer.

5. FILE INFO A/D PTY is based upon REC SBI P1. Because the SBI function always contains an odd number of logic 1s and the corresponding bits written into the register file are 0s, REC SBI P1 can be used as FILE INFO A/D PTY except for an extended read with address bit 00 set. In this case, bit 00 is reset, and FILE INFO A/D PTY is complemented.

6. FILE INFO CTR PTY is dependent on REC SBI B<31>. The command/mask bits written into the register file for a DMA read always contain an odd number of 1s. The only other bit involved with control parity is L/S <00>, which is just REC SBI B<31>. If the DMA read is for a quadword, an extended read, REC SBI B<31> is set, as is will L/S <00>.

   In this case FILE INFO CTR PTY is asserted. If REC SBI B<31> is not asserted, L/S <00> will not be asserted. There will be an odd number of 1s, all of them in the command, so FILE INFO CTR PTY will be equal to 0.

3.12.4 DMA Read: ID File

The ID field in the command/address, ID <04:00>, designates the source of the command. When the read data is transmitted onto the SBI, it must contain the same ID to enable the proper nexus to receive the read data.

The ID file is addressed by the upper two bits of the register file TTL write address, TTL FILE ADR <03:02>, and will be loaded with ID <04:00> from the command/address. This same ID is transmitted with each read data word.

To enable generating proper parity when the read data is transmitted on the SBI, the ID file must contain parity for the ID that is being stored. Each SBI bus transceiver can handle four bits, and, because there are five ID bits, it takes more than one SBI bus transceiver. When grouped with the three tag bits, two SBI transceivers are sufficient.

It is a command/address cycle, so the tag is known to be 011. Therefore, the received parity bits from the two transceivers can be exclusive ORed to provide even parity over the ID bits. This even parity bit is stored in the ID file to be transmitted with the read data.

3.12.5 DMA Read: Acknowledge

The SBIA will transmit an acknowledge, SBI CONF <01:00> = 10, two SBI cycles after receiving the command/address if the following conditions exist.

1. The tag equals 011, command/address.
2. There are no parity errors.
3. The address is within the bounds of memory as determined by the configuration register.
4. There is no interlock sequence fault.

3.12.6 DMA Read: IOA Request

At about the same time the acknowledge is being transmitted on the SBI, the command/address is being written into the register file; the circuitry that enables sending the acknowledge queues up the DMA read request by setting a command ready flip-flop. If no other DMA transaction buffer has issued an IOA request to the MBox, SB ABUS IOA REQUEST [N] will be asserted. If another DMA transaction buffer has an IOA request in progress, the current request waits until the previous request has been satisfied.

3.12.7 DMA Read: MBox Reads the Register File

The MBox arbitrates the IOA request, and when it is ready to service the DMA request, it reads the SBIA register file to obtain the command/address (see Figure 3-30).

When the MBox selects the SBIA with MCC ABUS IOA SELECT [N], the SBIA sends ABUS WR CMD and ABUS MSKED CMD (both equal zero for an extended read) to inform the MBox of the type of operation. This allows the MBox microcode to branch before it receives the command/address.

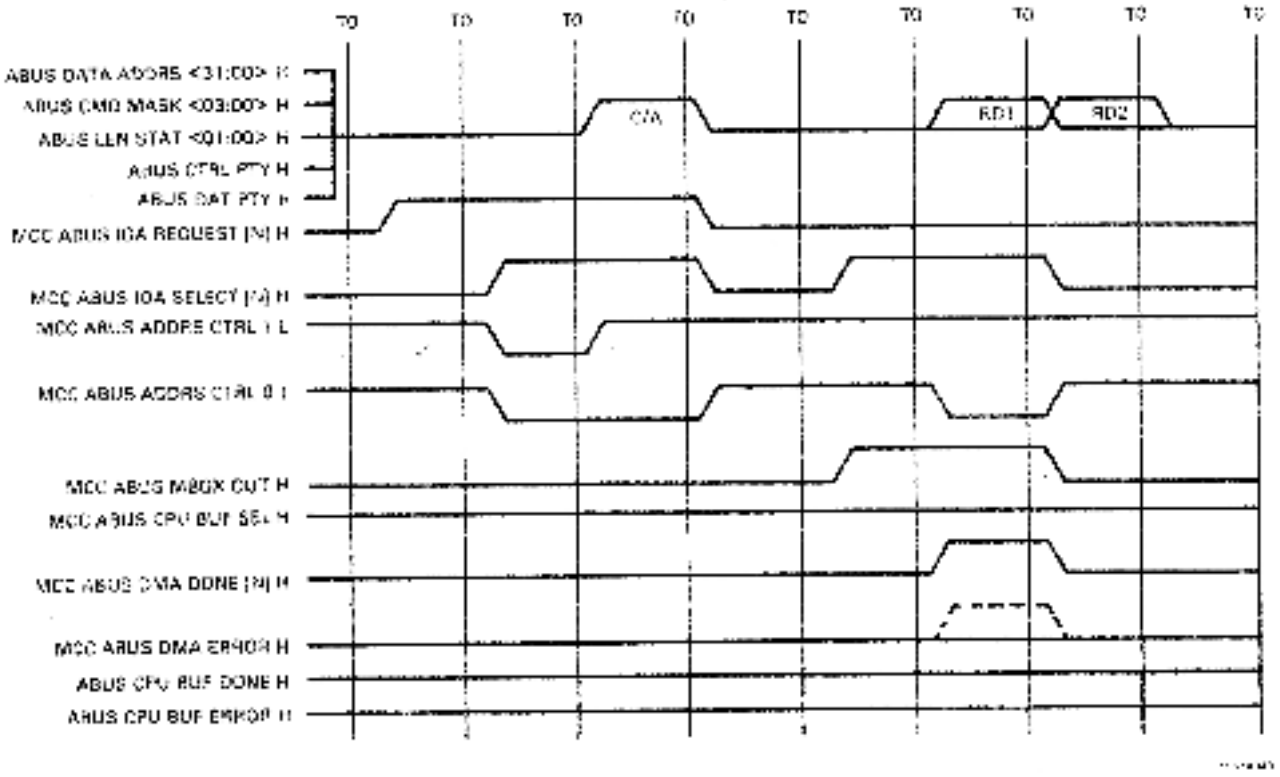


Figure 3-30 DMA Quadword Read ABus Protocol (with cache hit)

The register file ECL read address is selected in the same manner as for a DMA write (see Table 3-14 and Paragraph 3.10.8). On the ABus cycle following the loading of the ECL FILE ADR, the MBox reads the command/address word and drops MCC ABUS ADDRS CTRL 1 L (= 1). This will cause the ECL FILE ADR to increment by 1 to the location of the first read word. MCC ABUS ADDRS CTRL 0 L is then dropped (= 1), which causes the address to be held.

The MBox drops IOA select after reading the command/address. It now gets the read data from cache or memory. Figure 3-30 assumes a cache hit, but whatever the case, the MBox does not assert IOA select until the data is available. The first read data longword is placed on the ABus and written into the register file at location XX01. MCC ABUS ADDRS CTRL <01:00> equal 10 will increment the ECL FILE ADR to XX10, and the second read data longword is written into the register file.

At this point, both read data longwords are stored in the SBIA register file awaiting transfer to the SBI. They are in locations 2 and 3 of the transaction buffer that initiated the IOA request.

3.12.8 DMA Read: DMA DONE/ERROR

When the MBox transfers the first read data longword to the ABus, it also asserts MCC ABUS DMA DONE [N] to notify the SBIA that the data is on the ABus. If the MBox had detected either an address or command parity error on the command/address, it would also assert MCC ABUS DMA ERROR [N] at the same time.

When the SBIA receives DMA DONE, it requests the SBI by asserting DMA TR (transfer request). If there is an error, the transfer is aborted by clearing the DMA request in progress.

3.12.9 DMA Read: Register File TTL Read Address

The register file TTL read address must be set up to read the data and transfer it to the SBI. The file read address is generated according to the DMA transaction buffer with a request in progress (IOA request to the MBox) and DMA TR (see Table 3-2 and Paragraph 3.2.3). If DMA transaction buffer A contained the read data, FILE READ ADR <03:00> would equal 0101, the location for the first read data longword.

The address is loaded and held until the read data has been transferred to the SBI. The address is incremented to 0110 when the first longword is transferred to the SBI.

3.12.10 DMA Read: DMA Read Data Transfer to the SBI

Once the register file TTL read address has been set up, the contents of the addressed location are read out and written into the file data latch every SBI cycle. However, the SBI transceivers will not be enabled until the SBIA has received DMA ARB OK, which signifies that the SBIA has control of the SBI for a DMA transfer of read data. When DMA ARB OK is received, the SBIA holds the SBI for an extra cycle by asserting SEND DMA HOLD. This will cause BUS SBI TR00 to be asserted. The read data longwords are read from the register file and driven onto the SBI (see Figure 3-31) according to the following list.

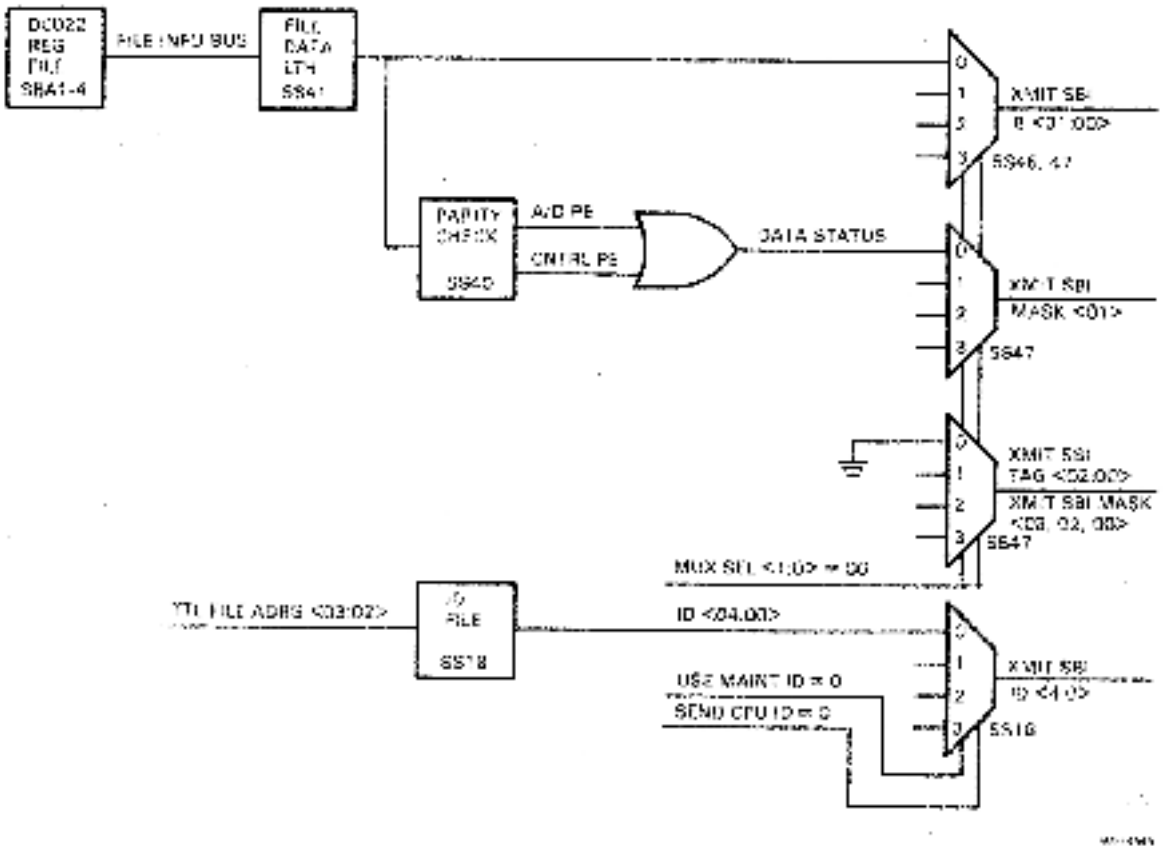


Figure 3-31 DMA Quadword Read: S-Data Assembly Transfer of Read Data

1. XMIT SBI B<31:00>: The contents of the addressed location in the register file is latched in the file data latch. Bits <31:00> are passed through the zero input of multiplexers to the SBI transceivers.

2. XMIT SBI TAG <02:00> = 000 because the multiplexer inputs are at ground potential.

3. XMIT SBI MASK <03:02, 00> = 000 because the multiplexer inputs are also at ground potential.

4. XMIT SBI MASK <01> depends on the presence or lack of data or control parity errors. Parity is checked over the contents of the file data latch, and a parity error causes the assertion of this mask bit. A MASK field of 0010, when sent with the read data, indicates read data substitute, an error condition, and allows the requesting device to detect the error.

5. XMIT SBI ID <04:00>: During the command/address cycle, the ID file was loaded with the ID from the command/address – the ID of the nexus that initiated the DMA transaction (see Paragraph 3.12.4). The ID file is being addressed by TTL FILE ADRS <03:02>. The ID is transferred to the SBI drivers through the zero input of multiplexers.

6. XMIT SBI P0: The ID file contains ID PARITY, even parity over the ID bits. It is routed to the SBI as P0. This may be done because the tag and mask bits are all 0s. If there is a data parity error, control parity error, or the MBox indicates an ABus error (either L/S bit set), MASK<01> is set (read data substitute) to inform the requesting device of the error. But, the mask field now equals 0010, and the parity bit is now incorrect. Therefore, for a data or control parity error, the control parity bit, P0, is toggled to insure that control parity is correct for the control field. This insures that only the requesting device detects the error.

7. XMIT SBI P1: If there is no parity error over the data bits, file data latch A/D parity (odd parity) is toggled to provide even parity for the SBI. If there is a parity error over the data bits, the parity bit is already even parity. It is not changed. The data on the SBI will be bad data, but because the parity is correct, only the requesting device will detect the error (MASK = 0010, read data substitute).

3.12.11 DMA Read Clear

When DMA ARB OK is received from the SBI priority arbitration chips, the DMA transaction being serviced is removed from the command in progress state, to free the transaction buffer for further usage. The read data cannot be destroyed before it is transferred to the SBI because the DMA has control of the SBI for the next two cycles.

3.12.12 DMA Read: Second Read Data Longword

The register file TTL read address is incremented to the location of the second read data longword, which is transferred to the SBI in the same manner as the first read data longword. When this data has been transmitted on the SBI, the SBIA circuitry is in a passive state waiting for another DMA or CPU transaction to be initiated.

3.13 SBIA SILO

The SBIA silo consists of RAMs providing a 16 location × 32-bit recorder that is loaded with selected SBI signals during each SBI cycle. When an SBI fault is detected by any SBI nexus, including the SBIA, the silo is locked. The CPU can read the silo to determine the sequence of events that led to the fault condition. The silo may also be locked, for maintenance purposes, by the silo comparator (see Figure 3-32).

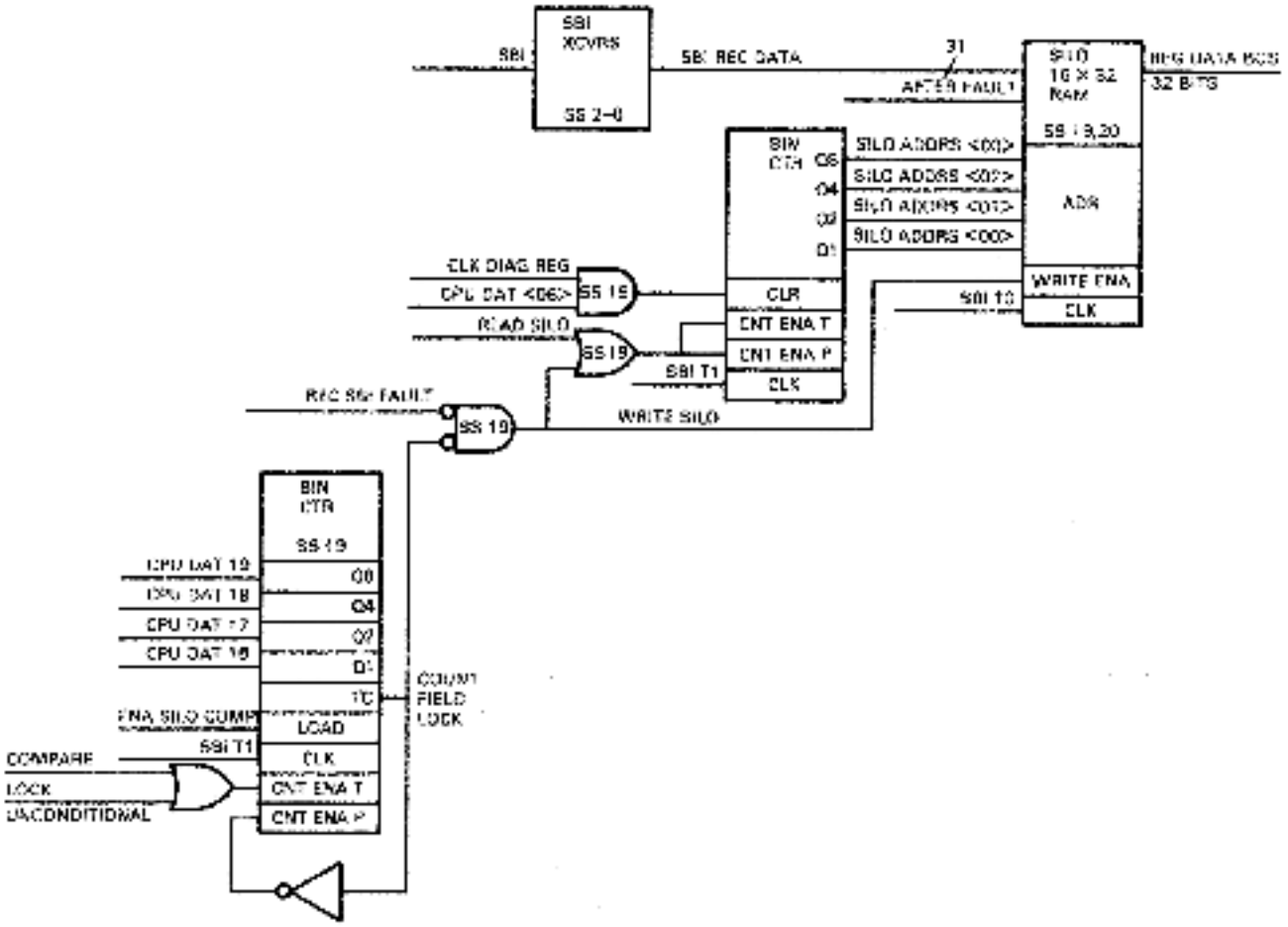


Figure 3-32 SBIA Silo

3.13.1 Silo Contents

The silo, read through the silo register, is loaded with the following SBI signals.

1. Silo register <31>: AFTER FAULT. Asserted the cycle after the SBI FAULT is cleared; loaded into the silo in the first location loaded following a fault.
2. Silo register <30>: SBI INTLK
3. Silo register <29:25>: SBI ID<04:00>
4. Silo register <24:22>: SBI TAG<02:00>

5. Silo register <21:18>: SBI B<31:28> or SBI MASK<03:00>. If SBI TAG = 011, command/address, the SBI command, B<31:28> will be loaded into the silo. Otherwise, the SBI mask will be loaded.

6. Silo register <17:16>: SBI CONF<01:00>.

7. Silo register <15:00>: SBI TR<15:00>.

3.13.2 Locking the Silo

The silo may be locked for two reasons:

1. The SBIA receives SBI FAULT or the SBIA detects an SBI fault for any of the following conditions.

   a. Interlock sequence fault
   b. Unexpected read fault
   c. Write sequence fault
   d. Multiple transmitter fault
   e. SBI parity fault.

2. The SILO comparator has detected that a predetermined number of SBI events have been written into the silo (used as a maintenance tool).

3.13.3 Silo During Normal System Operation

Every SBI cycle, at T0, the silo is loaded with the contents of the SBI transceivers and AFTER FAULT. The silo address is incremented on the following T1, to insure that the next SBI cycle is loaded into the next sequential silo location. The sequence continues indefinitely as long as there are no SBI faults.

When an SBI nexus detects an SBI fault, that nexus transmits BUS SBI FAULT on the SBI, which is received by the SBIA at T2 and latched at −T2, asserting REC SBI FAULT. The assertion of REC SBI FAULT prevents writing further data into the silo and disables incrementing the silo address.

NOTE

The present SBI cycle will not be written into the silo. The latest silo data is the SBI cycle previous to the assertion of BUS SBI FAULT.

In response to the FAULT interrupt, the CPU may read the contents of the silo by reading the silo register. Each time the contents of the silo register are transferred to the register file, the silo address is incremented. If the CPU does not clear the silo address, but starts reading at the present silo address, the sixteenth location read is the last SBI cycle loaded into the silo. The fifteenth location read is the next-to-the-last SBI cycle loaded into the silo, and so on. Also, if the number of SBI bus cycles was less than 16 since the last time the silo was locked, the first location loaded after the last fault was cleared has bit 31, AFTER FAULT, asserted.

3.13.4 Silo During Maintenance

When used for maintenance, the silo is capable of being locked in two ways, in addition to the unconditional lock for SBI FAULT. First, it can be locked after a predetermined number of SBI cycles, ranging from 1 to 16. Also, it can be locked after a predetermined number of SBI cycles after a particular SBI event has taken place.

3.13.4.1 Silo Unconditional Lock – The CPU loads the silo counter, silo comparator register <19:16>, with the 1's complement of the number of SBI cycles to load into the silo. At the same time, the CPU sets LOCK UNCONDITIONAL, silo comparator register <29>. LOCK UNCONDITIONAL enables the counter to count once each SBI cycle after the silo has been loaded. When the count reaches all 1s, F, the silo is locked. The address is not incremented and no further data is loaded into the silo.

When the silo is locked, a compare interrupt flip-flop is set. If the CPU has set silo lock interrupt enable (SILO LOCK INT ENA), silo comparator register <30>, COMP INTR will interrupt the CPU. The compare interrupt is cleared when the CPU loads the silo counter with a count other than F.

3.13.4.2 Silo Conditional Lock – In this silo maintenance mode, the silo is loaded every SBI cycle, as in normal operation, with the silo address being incremented after each SBI cycle. However, the count will be incremented only after a predetermined SBI event has been detected. A silo comparator will compare SBI conditions with the following SBIA register contents.

1. SBI maintenance register <27:23>: Maint ID<04:00>
2. Silo comparator register <26:23>: Maint command/mask <03:00>
3. Silo comparator register <22:20>: Maint TAG <02:00>.

The comparator can be programmed to check for the following comparisons.

1. The SBI ID equals the maintenance ID.
2. The SBI ID and SBI TAG equal the maintenance ID and TAG.
3. The SBI ID, SBI TAG, and SBI command/mask equal the maintenance ID, TAG, and command/mask. If the SBI TAG = 011, command mask, the comparison is for commands; otherwise, the masks are compared.

SBI silo comparator register <28:27>, COND LOCK CODE <01:00> controls which comparison will be made. Table 3-15 shows how the comparisons are controlled by COND LOCK CODE.

Table 3-15 COND LOCK CODE Control of Silo Comparisons

COND LOCK CODE <01:00>	Function
00	No compare
01	ID equal
10	ID and tag equal
11	ID and tag and command/mask equal

The CPU sets the bits according to the particular SBI ID, TAG, and command/mask the comparator is to look for. It then loads the counter with the 1s complement of the number of SBI cycles that are to be recorded after the enabled SBI conditions have been detected. If the CPU wishes to be interrupted when the silo is locked, it also sets LOCK INT EN.

The silo is loaded at T0 of each SBI cycle, with the address being incremented after the silo is loaded. If the SBI data matches for the enabled comparison, the compare latch is set, enabling the silo counter to start counting.

When the counter reaches F, the silo is locked, and if the interrupt is enabled, COMP INT will be asserted to interrupt the CPU.

When the contents of the silo are read back, if the initial count field was 0, the first entry read from the silo is the cycle that satisfied the comparison. The next 15 locations will be the next 15 SBI cycles.

3.14 SBIA REGISTERS

Each SBIA register will be shown with the addresses for SBIA 0 and SBIA 1. Each illustration is made up of the following.

1. A bit map
2. A descriptive name for each bit, with a 0 or a 1 if they are always a logic 0 or 1
3. An indication of whether the bit is read/write, read only, or write only.

Each table is laid out as follows.

1. Bit numbers
2. The descriptive name used in the bit map, 0 for zero
3. The actual print set name for the signal
4. A brief description of the bit.

3.14.1 Configuration Register

The configuration register bit map is given in Figure 3-33 and defined in Table 3-16.

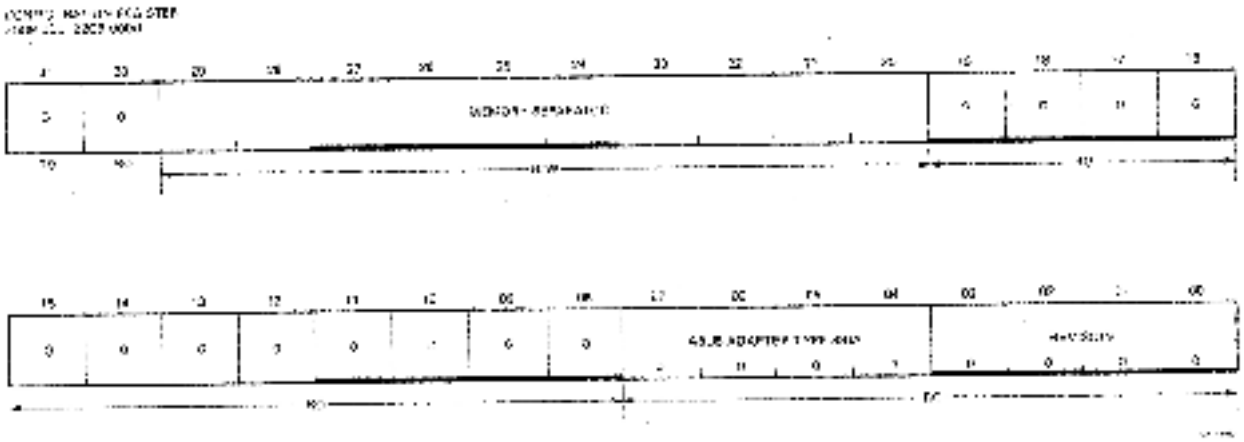


Figure 3-33 Configuration Register

Table 3-16 Configuration Register Bit Definition

Bit	Name	Definition
<31:30>	ZERO	(SS28) – Read only as zero
<29:20>	MEMORY SEPARATOR	SS28 MSR <27:18> – Defines the memory address boundary. Is equal to the number of megabytes of memory addressable over the ABus. If bit 29 is asserted, there are 512 Mbytes of memory, and bits <28:20> are disregarded when the hardware checks the DMA address. These bits are bits <29:20> in the memory separator register, but within the SBIA they are shifted right by two bits to match the physical address.
<19:08>	ZERO	(SS36) – Read as 0s provided by the zero fill logic.
<07:04>	ABUS ADAPTER TYPE	BUS REG D<07:04> (SS28) – Identify the type of ABus adapter, 0001 for the SBIA.
<03:00>	ABUS ADAPTER REVISION	BUS REG D<03:00> (SS28) – Least significant bits identify the revision of the ABus adapter, hardwired.

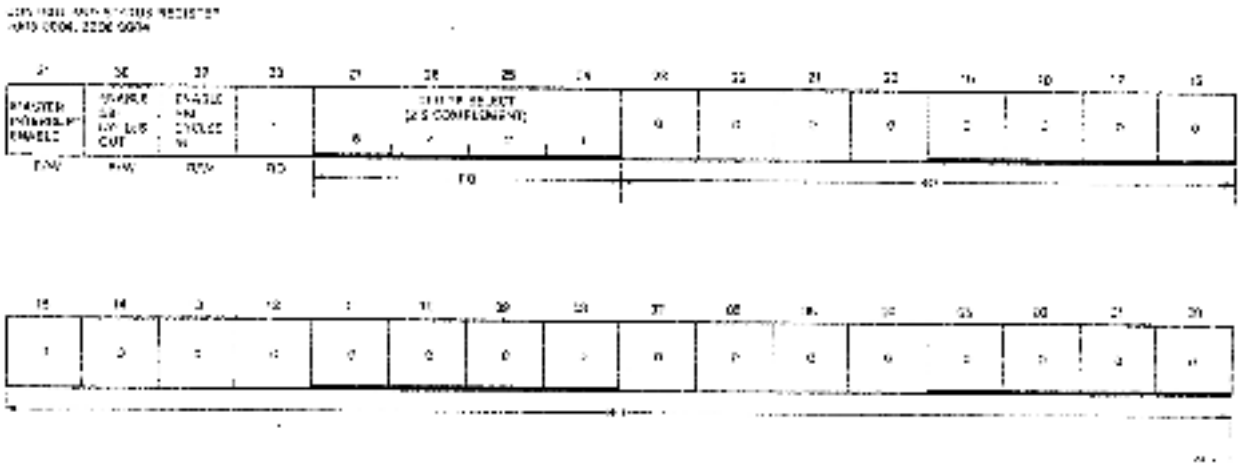


Figure 3-34 Control and Status Register

3.14.2 Control and Status Register

The control and status register bit map is given in Figure 3-34 and defined in Table 3-17.

Table 3-17 Control and Status Register Bit Definitions

Bit	Name	Definition
<31>	MASTER INTERUPT ENABLE	SS29 MSTR INTR ENA – When set, enables the SBA module to establish priority of interrupts and generates the appropriate interrupt priority level for CPU polling.
<30>	ENA SBI CYCLES OUT	SS29 ENA SBI OUT – Must be set for normal operation. Enables CPU to access SBI nexus registers. If the CPU attempts to access an SBI nexus register with this bit reset, an error condition occurs, and error summary register bits 20 and 19 are set (see description of the error summary register. Paragraph 3.14.3).
<29>	ENA SBI CYCLES IN	SS29 ENA SBI IN – Must also be set for normal operation. Enables all DMA activity through the SBIA. If this bit is not set, the SBIA will not recognize SBI function codes and will not respond to SBI commands (SBI confirmation is 00, no response).
<28>	ZERO	(SS33) - Read-only bit, will always be zero.
<27:24>	CPU TR SELECT <08:04>	CPU TR SEL <08:04> (SS07) -- Provide backpanel visibility of the jumpers used to select the SBI TR for CPU transactions. This field is the 2's complement of the TR level.
<23:00>	ZERO	(SS36) – Always read as 0s provided by the zero fill logic.

3.14.3 Error Summary Register

The error summary register bit map is given in Figure 3-35 and defined in Table 3-18.

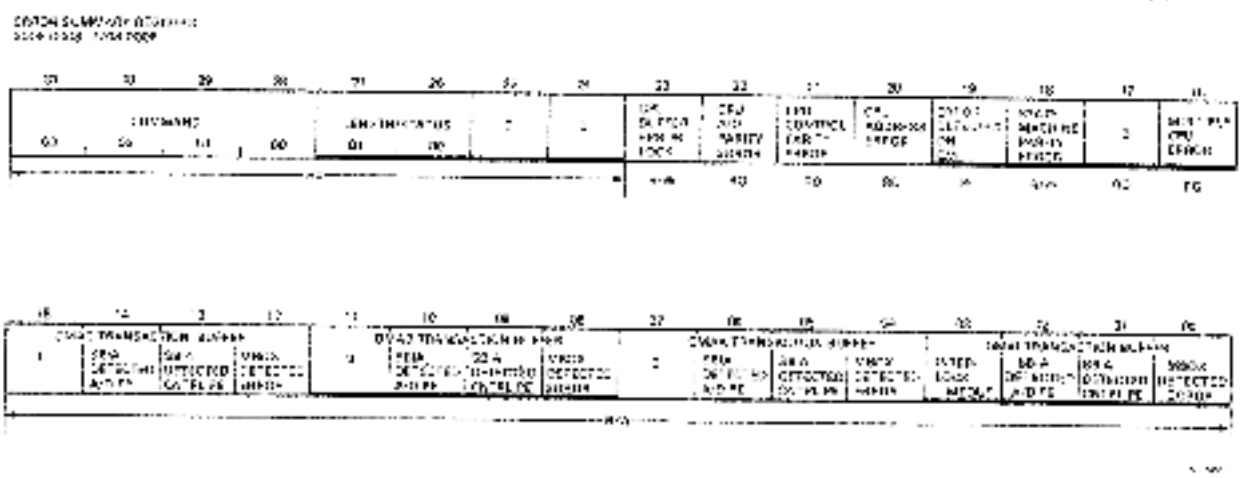


Figure 3-35 Error Summary Register

Table 3-18 Error Summary Register Bit Definitions

Bit	Name	Definition
<31:28>	COMMAND <03:00>	BUS REG D<31:28> (SS26) – The ABus command bits for a CPU I/O register read/write. Loaded every time the command/address latch is loaded, and latched by CPU ERROR LOCK, bit 23.
<27:26>	LENGTH/STATUS <01:00>	BUS REG <27:26> (SS26) – The ABus data length for a CPU I/O register read/write. Also loaded every time the command/address latch is loaded, and latched by CPU ERROR LOCK, bit 23.
<25:24>	ZERO	(SS26) – Read-only bits. Hardwired to a logic 0.
<23>	CPU BUFFER ERROR LOCK	SS37 CPU ERROR LOCK – Asserted for any of the following errors on a CPU I/O register read/write. 1. A/D parity error (bit 22) 2. Control parity error (bit 21)

Table 3-18 Error Summary Register Bit Definitions (Cont)

Bit	Name	Definition
		3. Address error (bit 20) 4. CPU read/write timeout on SBI (SBI error register bit 12) 5. SBI error (SBI error register bit 08). If this bit is set, error summary register <31:26>, the SBI error register, and the time-out address register are latched. If clear, these bits represent the most recent transaction. Writing this bit clears error summary register <22:19, 16>.
<22>	CPU A/D PARITY ERROR	SBAN A/D PTY BAD – Set if a parity error is detected on the address/data bits of the command/address or write data for a CPU I/O register read/write. If the error is detected on the command/address, bit 19 is also set. Parity is checked on the output of the file data latch. If this bit is set, bit 23 is set. Cleared when the CPU writes bit 23.
<21>	CPU CONTROL PARITY ERROR	SBAN CNTRL PTY BAD – Set if a parity error is detected on the control field of the command/address or write data for a CPU I/O register read/write. If the error is detected on the command/address cycle, bit 19 is also set. Parity is checked on the output of the file data latch. If this bit is set, bit 23 is also set. Also cleared when the CPU writes bit 23.
<20>	CPU ADDRESS ERROR	SS38 LOCAL ADR ERR – Set if the CPU accesses a nonexistent SBIA register or when an SBI nexus register is accessed when the control and status register bit 30 is clear (CPU access to the SBI is disabled). When it is set, it will set bit 23, and it is cleared when the CPU writes bit 23. This error is detected when the command/address word is available, so bit 19 should also be set.
<19>	ERROR DETECTED ON C/A	SBAN ERR ON C/A – Read-only bit set if an address/data purity error, a control parity error, or an address error is detected on the command/address cycle. The setting of this bit will set bit 23. This bit will be reset when the CPU writes bit 23.

Table 3-18 Error Summary Register Bit Definitions (Cont)

Bit	Name	Definition
<18>	STATE MACHINE PARITY ERROR	SBAO FORCE PARITY TRAP – Set if the state machine microword does not contain even parity. The occurrence of this error causes a CPU transaction to be aborted, if one is in progress, and generates an interrupt. A state machine parity error can occur if no CPU transaction is in progress, so it will not set bit 23.
<17>	ZERO	(SS33) – Read-only bit, hardwired to a logic 0.
<16>	MULTIPLE CPU ERROR	SBAN MULT CPU ERR – Can be set only if bit 23 is already set and a CPU addressing error is detected on the command/address cycle or there is an address/data or control parity error on the command/address or write data. Not set for a write data parity error for the transaction that sets bit 23, but for a subsequent transaction. Bit 16 is reset when the CPU writes bit 23.
<15>	ZERO	(SS32) – Read-only bit, hardwired to a logic 0.
<14>	DMAC TRANSACTION BUFFER SBIA DETECTED A/D PE	SS30 DMAC A/D ERR – Set for a data parity error when the read data is being transferred from transaction buffer C to the SBI during a DMA read. Cannot be set if bits 13 or 12 have been previously set. Cleared by the CPU writing it. The DMAC command/address register and DMAC ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<13>	DMAC TRANSACTION BUFFER SBIA DETECTED CNTRL PE	SS30 DMAC CNTRL ERR – Set for a control parity error when the read data is being transferred from transaction buffer C to the SBI during a DMA read. Cannot be set if bits 14 or 12 have been previously set. Cleared by the CPU writing it. The DMAC command/address register and DMAC ID register are locked if this bit is set. Setting this bit generates a local interrupt.

Table 3-18 Error Summary Register Bit Definitions (Cont)

Bit	Name	Definition
<12>	DMAC TRANSACTION BUFFER MBOX DETECTED ERROR	SS30 DMAC MBOX ERR – Set if the MBox detects a parity error or NXM on the transfer of command/address from the DMAC transaction buffer. Cannot be set if bits 14 or 13 have been previously set. Cleared by the CPU writing it. The DMAC command/address register and DMAC ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<11>	ZERO	(SS32) – Read-only bit, hardwired a logic 0.
<10>	DMAB TRANSACTION BUFFER SBIA DETECTED A/D PE	SS30 DMAB A/D ERR – Set for a data parity error when the read data is being transferred from transaction buffer B to the SBI during a DMA read. Cannot be set if bits 09 or 08 have been previously set. Cleared by the CPU writing it. The DMAB command/address register and DMAB ID register will be locked if this bit is set. Setting this bit generates a local interrupt.
<09>	DMAB TRANSACTION BUFFER SBIA DETECTED CNTRL PE	SS30 DMAB CNTRL ERR – Set for a control parity error when the read data is being transferred from transaction buffer B to the SBI during a DMA read. Cannot be set if bits 10 or 08 have been previously set. Cleared by the CPU writing it. The DMAB command/address register and DMAB ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<08>	DMAB TRANSACTION BUFFER MBOX DETECTED ERROR	SS30 DMAB MBOX ERR – Set if the MBox detects a parity error or NXM on the transfer of command/address from the DMAB transaction buffer. Cannot be set if bits 10 or 09 have been previously set. Cleared by the CPU writing it. The DMAB command/address register and DMAB ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<07>	ZERO	(SS32) – Read-only bit, hardwired to a logic 0.

Table 3-18 Error Summary Register Bit Definitions (Cont)

Bit	Name	Definition
<06>	DMAA TRANSACTION BUFFER SBIA DETECTED A/D PE	SS30 DMAA A/D ERR – Set for a data parity error when the read data is being transferred from transaction buffer A to the SBI during a DMA read. Cannot be set if bits 05 or 04 have been previously set. Cleared by the CPU writing it. The DMAA command/address register and DMAA ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<05>	DMAA TRANSACTION BUFFER SBIA DETECTED CNTRL PE	SS30 DMAA CNTRL ERR – Set for a control parity error when the read data is being transferred from transaction buffer A to the SBI during a DMA read. Cannot be set if bits 06 or 04 have been previously set. Cleared by the CPU writing it. The DMAA command/address register and DMAA ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<04>	DMAA TRANSACTION BUFFER MBOX DETECTED ERROR	SS30 DMAA MBOX ERR – Set if the MBox detects a parity error or NXM on the transfer of command/address from the DMAA transaction buffer. Cannot be set if bits 06 or 05 have been previously set. Cleared by the CPU writing it. The DMAA command/address register and DMAA ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<03>	DMAI TRANSACTION BUFFER INTERLOCK TIMEOUT	SS30 DMAI TIMEOUT – Set if an interlock write masked does not occur within 512 SBI cycles (102.4 μs) after an interlock read masked. Cannot be set if bits 02, 01, or 00 have been previously set. Cleared by the CPU writing it. The DMAI command/address register and DMAI ID register are locked if this bit is set. Setting this bit generates a local interrupt.

Table 3-18 Error Summary Register Bit Definitions (Cont)

Bit	Name	Definition
<02>	DMAI TRANSACTION BUFFER SBIA DETECTED A/D PE	SS30 DMAI A/D ERR – Set for a data parity error when the read data is being transferred from transaction buffer 1 to the SBI during a DMA interlock read. Cannot be set if bits 03, 01, or 00 have been previously set. Cleared by the CPU writing it. The DMAI command/address register and DMAI ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<01>	DMAI TRANSACTION BUFFER SBIA DETECTED CNTRL PE	SS30 DMAI CNTRL ERR – Set for a control parity error when the read data is being transferred from transaction buffer 1 to the SBI during a DMA interlock read. Cannot be set if bits 03, 02, or 00 have been previously set. Cleared by the CPU writing it. The DMAI command/address register and DMAI ID register are locked if this bit is set. Setting this bit generates a local interrupt.
<00>	DMAI TRANSACTION BUFFER MBOX DETECTED ERROR	SS30 DMAI MBOX ERR – Set if the MBox detects a parity error or NXM on the transfer of command/address from the DMAI transaction buffer. Cannot be set if bit 03, 02, or 01 have been previously set. Cleared by the CPU writing it. The DMAI command/address register and DMAI ID register are locked if this bit is set. Setting this bit generates a local interrupt.

3.14.4 Diagnostic Control Register

The diagnostic control register bit map is given in Figure 3-36 and defined in Table 3-19.

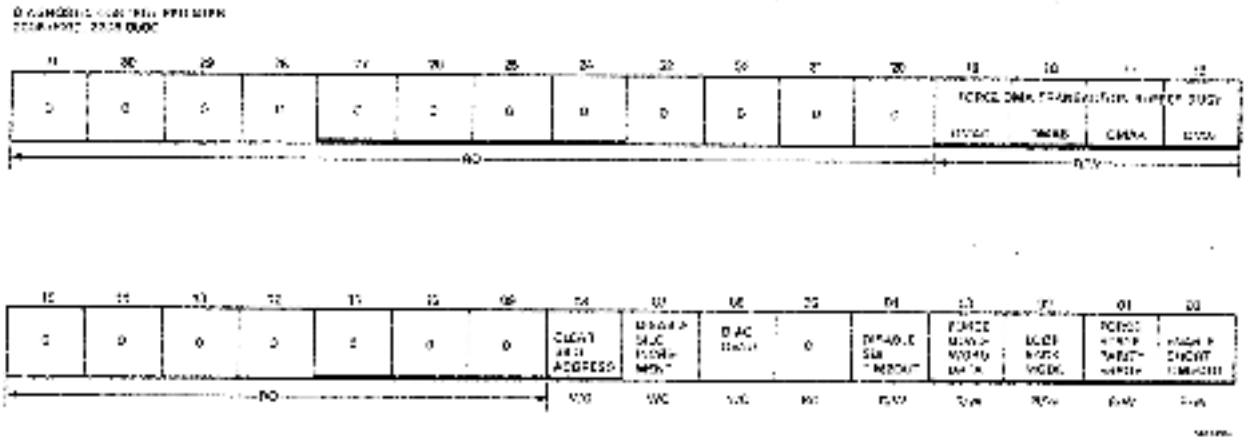


Figure 3-36 SBI Diagnostic Control Register

Table 3-19 SBI Diagnostic Control Register Bit Definition

Bit	Name	Definition
<31:20>	ZERO	<SS36) – Provided by the zero fill logic.
<19:16>	FORCE DMA TRANSACTION BUFFER BUSY	SS29 FORCE DMAC (DMAB, DMAA, DMAB) BUSY – Used to direct DMA traffic into specific DMA transaction buffers by forcing other buffers to be busy. The state of these bits has no effect on a DMA transaction already in progress.
<15:09>	ZERO	<SS32> – Hardwired to logic 0s.
<08>	CLEAR SILO ADDRESS	SS19 CLR SILO ADR – Clears the silo address upon setting. When this register is read, this bit is always 0 (hardwired).
<07>	DISABLE SILO INCREMENT	SS29 DISABLE SILO INC – When set, prevents the silo address from incrementing. Reset during normal operations to allow the silo address to increment. Also read as 0.
<06>	DIAG DEAD	SS29 DIAG DEAD – When set, simulates ABUS DEAD, interrupting the console and causing a reboot. ABUS DEAD is normally asserted by SBI FAIL. Also read as 0.
<05>	ZERO	(SS30) – Hardwired to 0.

Table 3-19 SBI Diagnostic Control Register Bit Definition (Cont)

Bit	Name	Definition
<04>	DISABLE SBI TIMEOUT	SS29 DISABLE SBI TMO – When set for diagnostics, prevents a timeout condition while waiting for the SBIA to gain control of the SBI, for an acknowledgment from a nexus, or for CPU read data.
<03>	FORCE QUADWORD DATA	SS29 FORCE QUAD DATA – Used by microdiagnostics with bit 2 (loopback mode) to provide a way to use a quadclear to loop data back on the SBI. FORCE QUAD DATA is set, and then the CPU executes a quadclear. For microdiagnostics, the address is a memory address instead of an SBI address (bit 27 is clear). The ABus command/address is the same as for the quadclear data transfer to the SBI (see Figure 3-17). It specifies a CPU write to the quadclear register. The ABus write data is the same except for the address, which is for a memory (cache) address. When the command/address is transmitted on the SBI it will be received by the SBIA, as it always is, but in this case, the address is less than the configuration register. To the SBIA it appears as a DMA extended write mask to memory and is handled as such. For a normal quadclear, the write data is forced to all 0s. In this case, FORCE QUAD DATA enables the A-data assembly multiplexers to transfer the contents of the write data latch (the ABus write data, 10:1 and the quadword boundary address) to the SBI. When the second write data longword is transferred to the SBI transceivers, bits 30 and 27 will be toggled (set) to allow the setting of all data bits on the SBI.
<02>	LOOP BACK MODE	SS29 LOOP BACK MODE – Used by microdiagnostics to allow a CPU read or write to be looped back in the SBIA. The PAMM has to be configured such that a memory (cache) address is mapped to an I/O adapter, and the same PAMM address, but with bits 27 and 28 inverted, is mapped to a memory address.

Table 3-19 SBI Diagnostic Control Register Bit Definition (Cont)

Bit	Name	Definition
		In the SBIA, LOOP BACK MODE inverts address bits 25 and 26 if bit 27 is reset, which will be the case if the CPU write is to a memory address. When the CPU writes a memory location that is mapped to an I/O adapter, the MBox writes the command/address and write data longword into the register file. The SBIA will carry out the command as a normal CPU write. When the command/address is transferred from the command/address latch to the SBI, because LOOP BACK MODE is set and address bit 27 is reset, address bits 25 and 26 are inverted. The command/address, followed by the write data, is transmitted on the SBI. When the SBIA clocks the SBI receivers and looks at the received data, if the address is less than the memory separator (in the configuration register), it will transfer the command/address and write data to the register file and request MBox service. The MBox writes the data into memory because the address, with bits 27 and 28 inverted (bits 25 and 26 in the SBIA), addresses a different PAMM location. This location is mapped to memory. This diagnostic bit can also be used with a CPU read in a similar manner for a further check of the SBIA logic.
<01>	FORCE STATE PARITY ERROR	SS29 FORCE STATE PTY – If set, a state machine parity error is forced during the CPU ARB WAIT state.
<00>	ENABLE SHORT TIMEOUT	SS29 ENA SHORT TIMEOUT – When set enables an SBI timeout in 8 SBI cycles instead of the normal 512 cycles. Read as a 0.

3.14.5 DMA Command/Address Registers

The DMA command/address register bit map is given in Figure 3-37 and defined in Table 3-20.

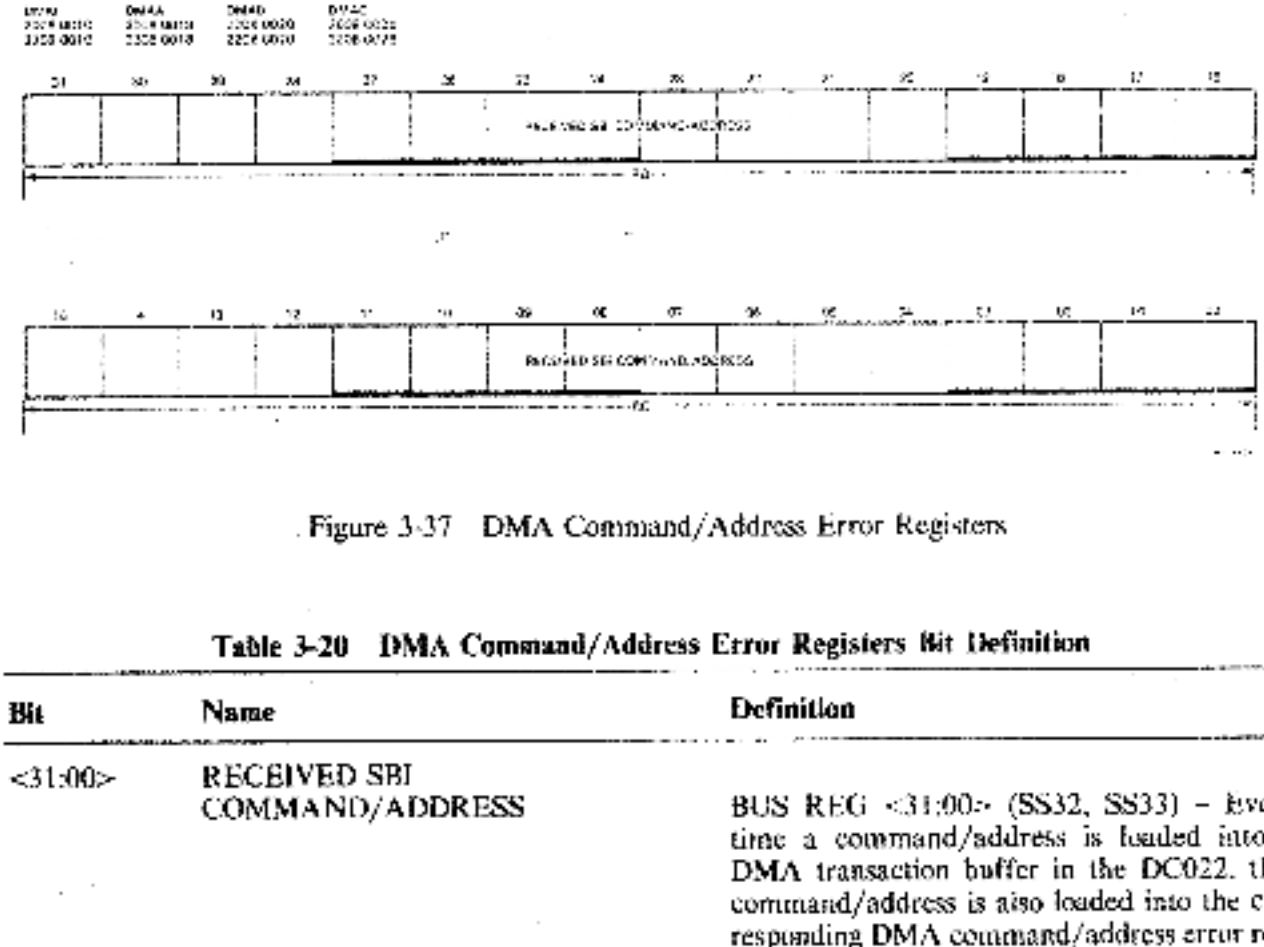


Figure 3-37 DMA Command/Address Error Registers

Table 3-20 DMA Command/Address Error Registers Bit Definition

Bit	Name	Definition
<31:00>	RECEIVED SBI COMMAND/ADDRESS	BUS REG <31:00> (SS32, SS33) – Every time a command/address is loaded into a DMA transaction buffer in the DC022, that command/address is also loaded into the corresponding DMA command/address error register. These error registers are actually TTL register files addressed by the upper two bits of the DC022 write address. SBI B<31:00> are written in these registers, with bits <31:28> being the SBI command codes and bits <27:00> the longword address. These error registers are locked if the SBIA or MBox detects a DMA error.

3.14.6 DMA ID Registers

The DMA ID register bit map is given in Figure 3-38 and defined in Table 3-21.

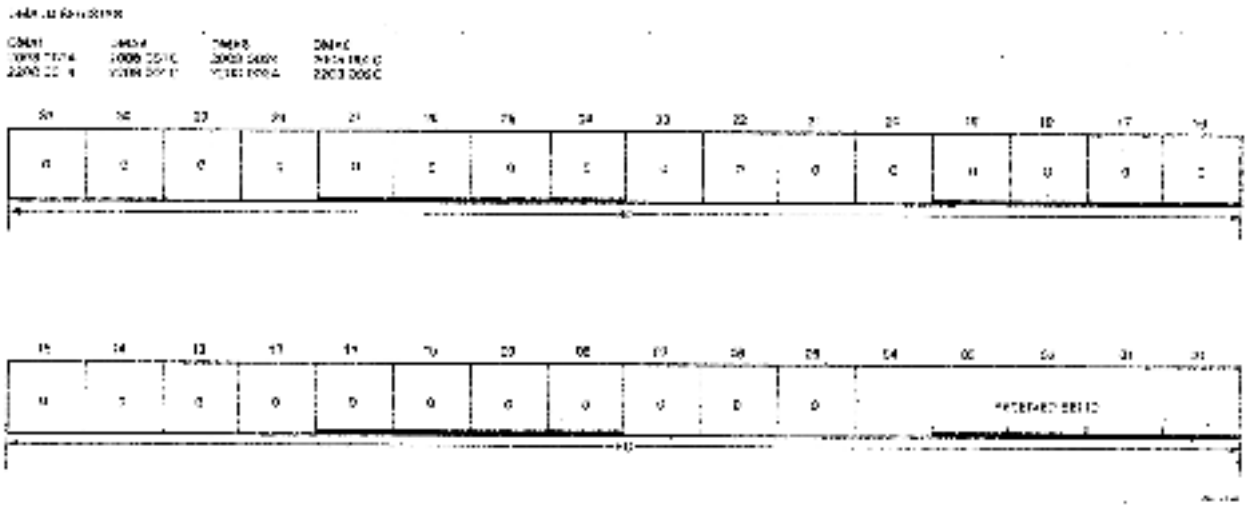


Figure 3-38 DMA ID Error Registers

Table 3-21 DMA ID Error Register Bit Definition

Bit	Name	Definition
<31:08>	ZERO	(SS36) -- Forced to 0 by the zero fill logic.
<07:00>	RECEIVED SBI ID	BUS REG <07:00> <SS23) – Each time a command/address is loaded into a DMA transaction buffer in the DC022, the SBI ID is also loaded into the corresponding DMA ID error register, an extension of the DMA command/address error registers. These error registers, like the command/address error registers, are TTL register files and are addressed in the same way, by the upper two bits of the DC022 write address. Bits <07:05> have the inputs at ground potential so they will always be read as 0. Bits <04:00> are loaded with REC SBI ID <04:00>. Like the DMA command/address error registers, these registers are locked if the SBIA or MBox detects a DMA error.

3.14.7 SBI Silo Register

The SBI silo register bit map is given in Figure 3-39 and defined in Table 3-22.

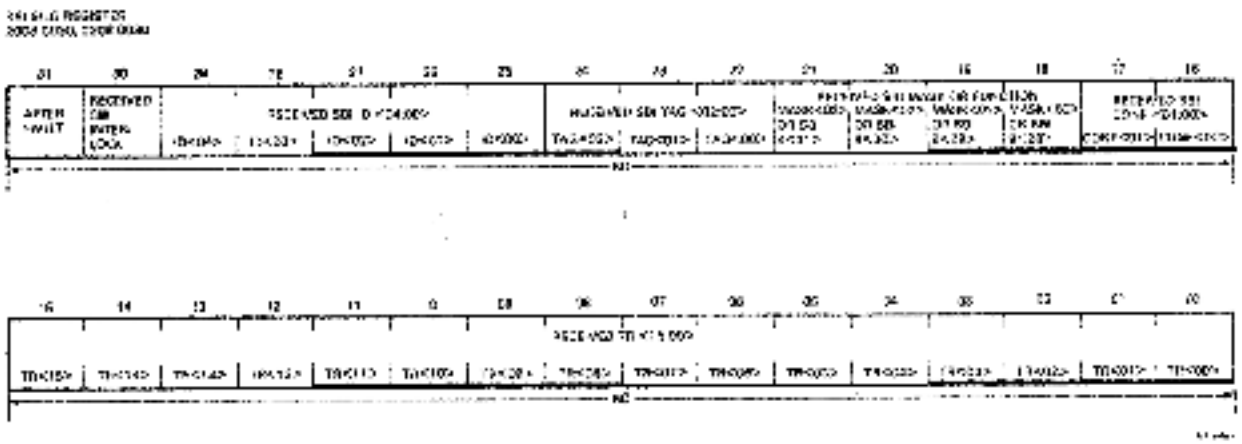


Figure 3-39 SBI Silo Register

Table 3-22 SBI Silo Register Bit Definition

Bit	Name	Definition
<31>	AFTER FAULT	BUS REG D<31> (SS20) – Loaded with AFTER FAULT, an indication that the SBI fault condition has cleared. AFTER FAULT is asserted for only one SBI cycle and is written into the first silo location after the fault clears. May be used to recognize frequently occurring fault conditions.
<30>	RECEIVED SBI INTERLOCK	BUS REG D<30> (SS20) – Loaded with REC SBI INTLK from the SBI transceivers.
<29:25>	RECEIVED SBI ID<04:00>	BUS REG D<29:25> (SS20) – Loaded with REC SBI ID<04:00>, an indication of which nexus has control of the SBI.
<24:22>	RECEIVED SBI TAG<02:00>	BUS REG D<24:22> (SS20) – Loaded with REC SBI TAG<02:00>, an indication of the type of SBI cycle as follows. 1. 000: Read data 2. 011: Command/address 3. 101: Write data 4. 110: Interrupt summary read 5. 111: Diagnostic tag.

Table 3-22 SBI Silo Register Bit Definition (Cont)

Bit	Name	Definition
<21:18>	RECEIVED SBI MASK or FUNCTION	BUS REG D<21:18> (SS20) – Contents depend on the SBI tag. If the tag is 011, command/address, the silo is loaded with the SBI function from bits <31:28>. Otherwise, the silo is loaded with the mask bits. The function codes are decoded as follows. 1. 0001: Read masked 2. 0010: Write masked 3. 0100: Interlock read masked 4. 0111: Interlock write masked 5. 1000: Extended read 6. 1011: Extended write masked. 1. Mask <03> = 1: Read or write to byte 3 2. Mask <02> = 1: Read or write to byte 2 3. Mask <01> = 1: Read or write to byte 1 4. Mask <00> = 1: Read or write to byte 1. The mask bit meanings for read data are as follows. 1. 0000: Valid read data 2. 0001: Corrected read data 3. 0010: Uncorrectable error.
<17:16>	RECEIVED SBI CONF <01:00>	BUS REG D<17:16> (SS20) – Loaded with the SBI confirmation bits, which have the following meanings. 1. 00: No response 2. 01: Acknowledge 3. 10: Busy 4. 11: Error on the command/address.
<15:00>	RECEIVED SBI TR<15:00>	BUS REG D<15:00> (SS19) – Loaded with any SBI transfer requests that may be asserted.

3.14.8 SBI Error Register

The SBI error register bits are given in Figure 3-40 and defined in Table 3-23.

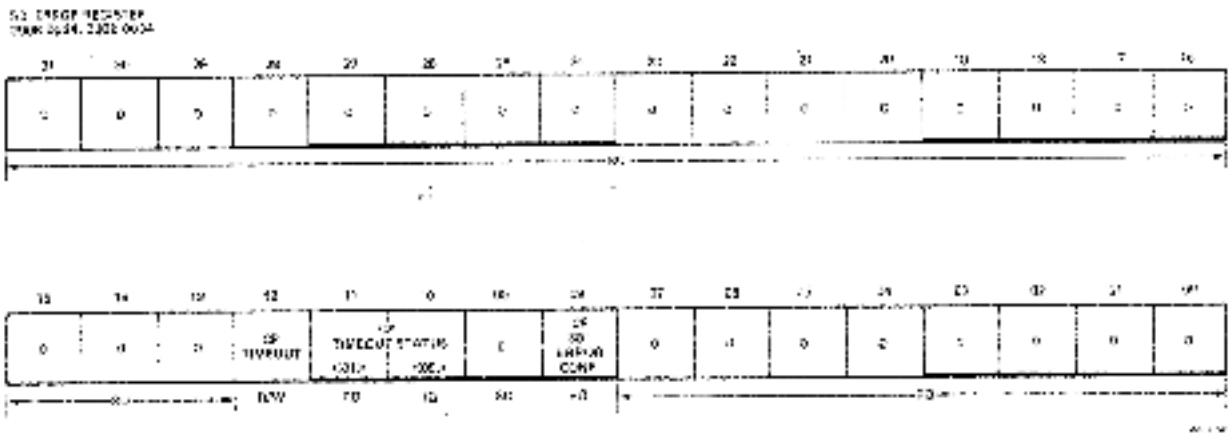


Figure 3-40 SBI Error Register

Table 3-23 SBI Error Register Bit Definitions

Bit	Name	Definition
<31:16>	ZERO	(SS36) – Read as 0s provided by the zero fill logic.
<15:13>	ZERO	(SS32) – Read-only bits, forced to 0 by hardware ground potential.
<12>	CP TIMEOUT BUS REG D<12>	(SS32) – Set when there is an SBI timeout on a CPU reference for one of the following reasons. 1. Unsuccessful access: When the SBIA does not receive an acknowledge confirmation for a CPU command/address or write data within 512 SBI cycles (102.4 μs) from the time the SBIA first requests the SBI. Unsuccessful access can be caused by the following a. SBIA is unable to win the SBI through bus arbitration. b. Target nexus is always busy when accessed. c. The address is for a nonexistent device or address. d. Combinations of the first two.

Table 3-23 SBI Error Register Bit Definitions (Cont)

Bit	Name	Definition
		2. If the SBIA does not receive the read data within 512 SBI cycles of the acknowledge for the command/address, it is a read data timeout. When this bit is set, error summary register 23 is set, which locks error summary register <31:26> (type of reference), SBI error register <11:10, 08>, and the timeout address register (referenced address). Reset when the CPU writes it to a 1. This will also reset <11:10, 08>.
<11:10>	CP TIMEOUT STATUS <01:00>	BUS REG D<11:10> (SS32) – Timeout status bits are made up of two signals as follows. 1. Bit <01>: State machine is in the read pending state. 2. Bit <00>: SBI confirmation equals 01, busy. Together, these two signals indicate the type of SBI timeout. 1. 00: SBI nexus did not respond (no response). 2. 01: Device was busy (busy). 3. 10: Waiting for read data. 4. 11: Cannot happen. These bits are locked by error summary register bit 23, and reset when the CPU writes SBI error register 12.
<09>	ZERO	(SS32) – Read-only bits, forced to 0 by hardware ground potential.
<08>	CPU SBI ERROR CONF	BUS REG D<08> (SS32) – Set when the SBI state machine enters the error abort state if the SBI nexus has returned an error confirmation on a CPU read/write command/address cycle. If this bit is set, error summary register bit 23 is set, locking the timeout address register, error summary register <31:26>, and bits <12:10> of this register. Reset when the CPU writes bit 12.
<07:00>	ZERO	(SS32) – Read-only bits, forced to 0 by hardware ground potential.

3.14.9 SBI Timeout Address Register

The SBI timeout address register bits are given in Figures 3-41 and defined in Table 3-24.

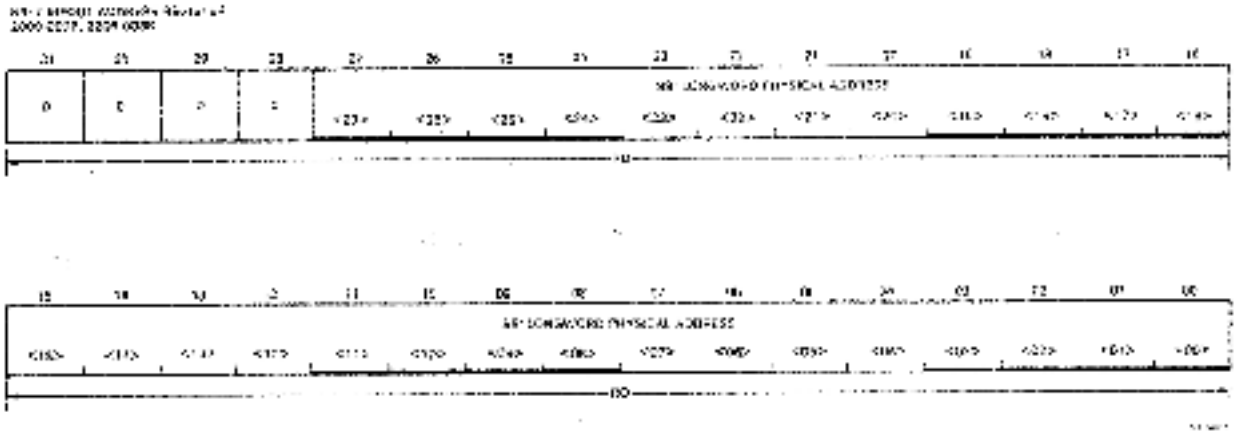


Figure 3-41 SBI Timeout Address Register

Table 3-24 SBI Timeout Address Register Bit Definition

Bit	Name	Definition
<31:28>	ZERO	(SS36) -- Forced to 0 by the zero fill logic.
<27:00>	SBI LONGWORD PHYSICAL ADDRESS	BUS REG D<27:00> (SS27) – Timeout address register is loaded with the physical address, for the CPU command, every time a command/address is transferred from the file data latch to the command/address latch. Locked if error summary register bit 23 is set.

3.14.10 SBI Fault/Status Register

The SBI fault/status register bits are given in Figure 3-42 and defined in Table 3-25.

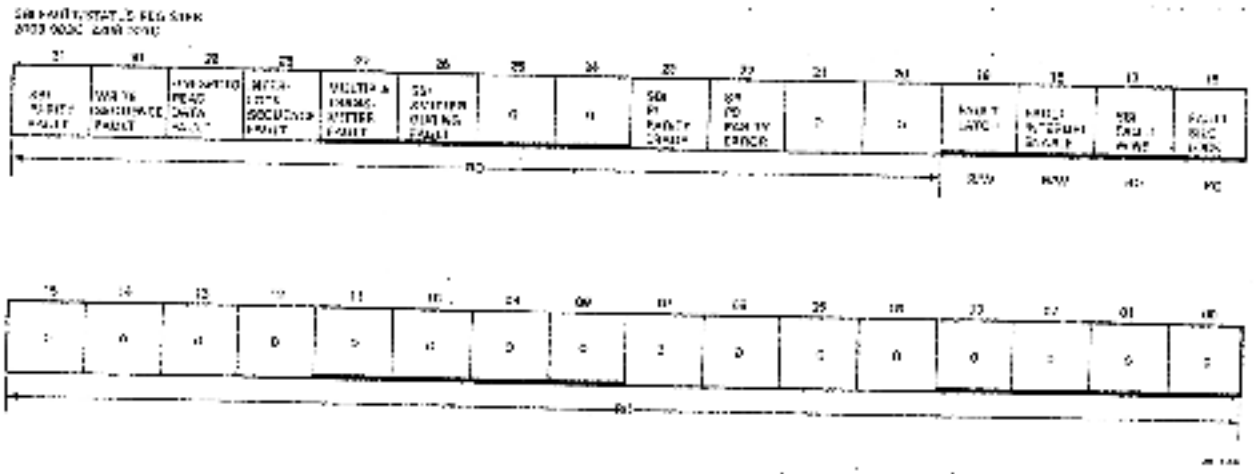


Figure 3-42 SBI Fault/Status Register

Table 3-25 SBI Fault/Status Register Bit Definitions

Bit	Name	Definition
<31>	SBI PARITY FAULT	SS1I FAULT REG B<31> – Set if the SBIA detects an SBI parity error on received SBI information. Bits <23:22> indicate an address/data or a control parity error. Register written every SBI cycle and locked when SBI FAULT is asserted. Cleared when SBI FAULT is deasserted. Valid only if bit 19 is set.
<30>	WRITE SEQUENCE FAULT	SS1I FAULT REG B<30> – Set if the SBIA is expecting write data and receives SBI information with no parity error, but the tag does not indicate write data (101). Also locked when SBI FAULT is asserted and clears when SBI FAULT clears. Valid only if bit 19 is set.
<29>	UNEXPECTED READ DATA	SS1I FAULT REG B<29> – Set if the SBIA receives information with the SBIA ID (10000) with a read data tag (000), but the SBIA is not expecting read data (no read pending). Locked when SBI FAULT is asserted and clears when SBI FAULT clears. Valid only if bit 19 is set.

Table 3-25 SBI Fault/Status Register Bit Definitions (Cont)

Bit	Name	Definition
<28>	INTERLOCK SEQUENCE FAULT	SS11 FAULT REG B<28> – Set if the SBIA receives a valid command/address for an interlock write masked but the interlock flip-flop is not set (an interlock read has not occurred). Locked when SBI FAULT is asserted and clears when SBI FAULT clears. Valid only if bit 19 is set.
<27>	MULTIPLE XMITTER FAULT	SS11 FAULT REG B<27> – Set if the SBIA detects an ID that is not the same as the ID it transmitted on the SBI. Locked when SBI FAULT is asserted and clears when SBI FAULT clears. Valid only if bit 19 is set.
<26>	SBI TRANSMITTER DURING FAULT	SS11 FAULT REG B<26> – Sets when the SBIA was the nexus transmitting on the SBI. Locked when SBI FAULT is asserted and clears when SBI FAULT clears. Valid only if bit 19 is set.
<25:24>	ZERO	(SS33) – Read as 0 provided by hardware ground potentials.
<23>	SBI P1 PARITY ERROR	SS11 FAULT REG B<23> – Indicates an SBI parity error over SBI B<31:00>. Valid only if bit 19 is set. Locked when SBI FAULT is asserted and clears when SBI FAULT clears.
<22>	SBI P0 PARITY ERROR	SS11 FAULT REG B<22> – Indicates an SBI parity error over SBI TAG <02:00>, SBI ID <04:00>, or SBI MASK <03:00>. Valid only if bit 19 is set and is locked when SBI FAULT is set. Clears when SBI FAULT clears.
<21:20>	ZERO	(SS33) – Forced to 0 by ground potentials.
<19>	FAULT LATCH	BUS REG D<19> (SS33) – If an SBI nexus (including the SBIA) detects an SBI fault, the nexus asserts SBI FAULT. The SBIA, upon reception of SBI FAULT, sets the fault latch, which keeps SBI FAULT asserted. It remains asserted until the CPU clears the fault latch by writing a 1 to bit 19. SBI FAULT is asserted for the following SBI error conditions.

Table 3-25 SBI Fault/Status Register Bit Definitions (Cont)

Bit	Name	Definition
		1. Interlock sequence fault 2. Unexpected read fault 3. Write sequence fault 4. Multiple transmitter fault 5. Parity fault. When this bit sets, fault/status register <31:26> and <23:22> are locked.
<18>	FAULT INTERRUPT ENABLE	SS21 FAULT INTR ENA – CPU sets this bit to enable an SBI fault to generate an interrupt. The interrupt is asserted if the fault latch, bit 19, is set.
<17>	SBI FAULT WIRE	SS33 BUS REG D<17> – Indicates the state of the SBI FAULT signal.
<16>	FAULT SILO LOCK	SS33 BUS REG D<16> – Set when the silo locks due to an SBI fault. Reset when the CPU resets the fault latch, bit 19.
<15:00>	ZERO	(SS36) – Forced to 0 by the zero fill logic.

3.14.11 SBI Silo Comparator Register

The SBI silo comparator register bits are given in Figure 3-43 and defined in Table 3-26.

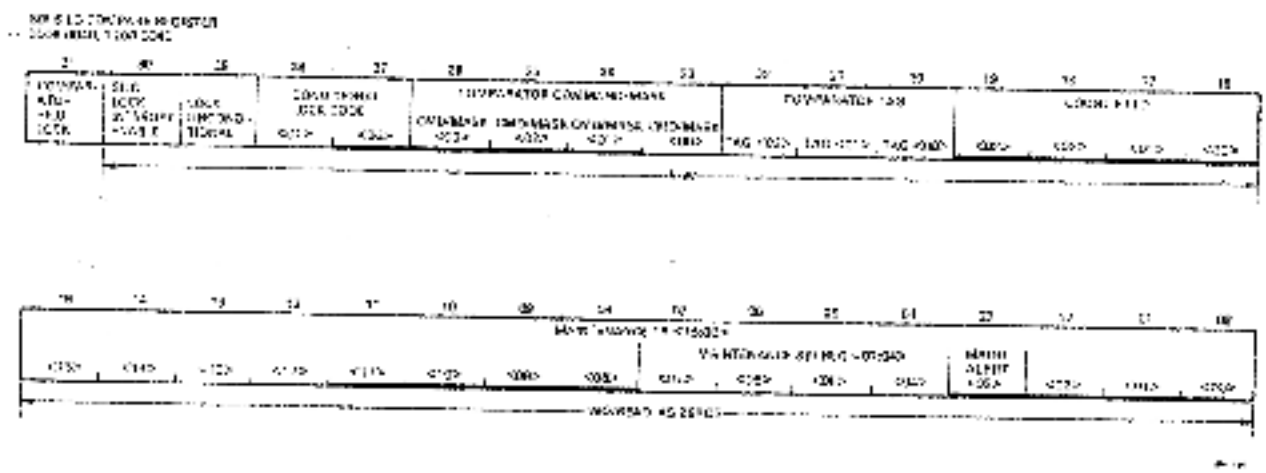


Figure 3-43 SBI Silo Comparator Register

Table 3-26 SBI Silo Comparator Register Bit Definition

Bit	Name	Definition
<31>	COMPARATOR SILO LOCK	SS2J CMP SILO LOCK – Set if the count in the silo counter has reached F. When this bit is set, the CPU is interrupted if bit 30 is set. Cleared when the CPU loads the silo count field with a count other than F.
<30>	SILO LOCK INTERRUPT ENABLE	SS25 SILO LOCK INTR EN – The CPU sets this bit to enable an interrupt when bit 31, CMP SILO LOCK, is set.
<29>	LOCK UNCONDITIONAL	SS25 LOCK UNCOND – When set, the silo counter counts on each SBI cycle (no comparison is made). Causes a silo lock within 16 SBI cycles, depending on the count loaded into the silo count field.
<28:27>	CONDITIONAL LOCK CODE <01:00>	SS25 COND LOCK CODE <01:00> – Determine the comparisons that enable counting the silo counter to achieve a silo lock. If the SBI data matches the silo comparator bits, for the enabled comparison, the counter is enabled to increment. The conditions are as follows. 1. 00: No compare (no comparison is made) 2. 01: SBI ID 3. 10: SBI ID and SBI TAG 4. 11: SBI ID and SBI TAG and SBI COMMAND/MASK.
<26:23>	COMPARATOR COMMAND/MASK	SS25 COMP CMD/MSK <03:00> – Provide the basis for the silo comparison, when it is enabled to compare the command/mask. If the SBI tag is 011, command/address, this field is compared with SBI B<31:28>, the SBI function. If the SBI tag is other than 011, this field is compared to the SBI mask bits.

Table 3-26 SBI Silo Comparator Register Bit Definition (Cont)

Bit	Name	Definition
<22:20>	COMPARATOR TAG	SS25 COMP TAG <02:00> – Provide the basis for the silo comparison when enabled to compare the tag. This field is compared with SBI TAG <02:00>.
<19:16>	COUNT FIELD	SS19 COUNT FIELD – The CPU loads the silo counter with the 1's complement of the number of SBI cycles to be loaded into the silo after a comparison is made. When the count reaches F, the silo is locked.
<15:00>	MAINTENANCE TR <15:00>	SS25 MAINT TR <15:00> – Provide the means to simulate SBI transfer requests, SBI interrupt requests, and SBI alert for diagnosing the interrupt logic and SBI priority arbitration logic. Also provide a means of testing the lower 16 bits of the silo. Controlled by SBI maintenance register bits <04:02> as follows. 1. The asserted MAINT TR <07:04> bit causes the corresponding SBI REQ <07:04> bit to be asserted if SBI maintenance register <04>, MAINT REQ ENA, is set. 2. If MAINT TR <03> and SBI maintenance register <04> are both set, SBI ALERT will be asserted. 3. If SBI maintenance register <03> is set, the asserted MAINT TR <15:00> causes the corresponding SBI TR <15:00> to be asserted when a CPU command/address is transmitted on the SBI. (See SBI maintenance register bit 03.) 4. MAINT TR <15:00> causes the corresponding SBI TR <15:00> to be asserted if SBI maintenance register bit 02, FORCE MAINT TR, is set.

3.14.12 SBI Maintenance Register

The SBI maintenance register bits are given in Figure 3-44 and defined in Table 3-27.

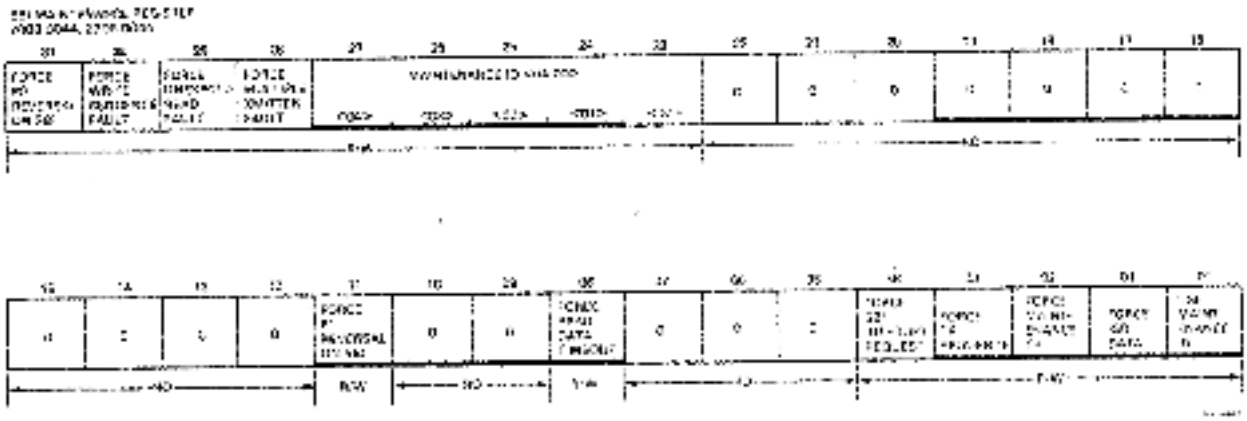


Figure 3-44 SBI Maintenance Register

Table 3-27 SBI Maintenance Register Bit Definition

Bit	Name	Definition
<31>	FORCE P0 REVERSAL ON SBI	SS24 FRC P0 REV ON SBI – When set, this causes the SBIA to transmit bad P0 parity on the SBI for all SBIA to SBI transactions including CPU read/write and DMA read data.
<30>	FORCE WRITE SEQUENCE FAULT	SS24 FORCE WSQ FAULT – When set, forces SBI TAG 01 to a logic 1. When used with a CPU write to an SBI nexus register, it forces the write data tag to 111, the diagnostic tag, causing a write sequence fault because SBI devices are looking for a tag of 101, write data.
<29>	FORCE UNEXPECTED READ FAULT	SS24 FORCE UNEXP READ – When set, the maintenance ID, bits <27:23>, with a tag of zero, are repeatedly transmitted on the SBI (the data is undefined). When the nexus, as selected by the maintenance ID, receives read data (TAG = 0), it should assert BUS SBI FAULT because of the unexpected read data.

Table 3-27 SBI Maintenance Register Bit Definition (Cont)

Bit	Name	Definition
<28>	FORCE MULTIPLE TRANSMITTER FAULT	SS24 FORCE MULTI XMIT – Used to force a multiple transmitter fault in any selected nexus. The CPU will load the maintenance ID with the ID of the selected nexus, then read that nexus configuration register. On the cycle after the command/address is transmitted on the SBI, the SBIA will enable the SBI to continually transmit a TAG = 111 with the maintenance ID (the data is undefined). When the nexus transmits the read data, the ID transmitted by the nexus is the SBIA's ID. It is ORed with the maintenance ID and, as long as the bits are not masked, causes the nexus to detect a multiple transmitter fault.
<27:23>	MAINTENANCE ID <04:00>	SS24 MAINT ID <04:00> – Used to generate the maintenance ID in the following instances. 1. Generation of unexpected read fault 2. Generation of multiple transmitter fault 3. Used by the silo as the compare ID 4. Used to check ID logic.
<22:20>	ZERO	(SS33) – Forced to 0 by ground potentials.
<19:16>	ZERO	(SS36) – Forced to 0 by the zero fill logic.
<15:12>	ZERO	(SS32) – Forced to 0 by ground potentials.
<11>	FORCE PI REVERSAL ON SBI	SS24 FRC PI REV ON SBI – When set, causes the SBIA to transmit bad PI parity on the SBI for all SBIA to SBI transactions. This includes CPU read/write and DMA read data.
<10:09>	ZERO	(SS32) – Forced to 0 by ground potentials.
<08>	FORCE READ DATA TIMEOUT	SS24 FORCE TIMEOUT – Presets the state machine timeout counter to all 1s when the state machine enters the read wait start state. The timer expires on the first count, generating a timeout condition while waiting for CPU read data.

Table 3-27 SBI Maintenance Register Bit Definition (Cont)

Bit	Name	Definition
<07:05>	ZERO	(SS32) – Forced to 0 by ground potentials.
<04>	FORCE SBI INTERRUPT REQUEST	SS24 MAINT REQ ENA – When set, enables SBI silo comparator register <07:04> and 03 to force interrupt requests and ALERT on the SBI.
<03>	FORCE TR SEQUENCE	SS24 FORCE TR SEQ – Enables SBI silo comparator register <15:00> to assert TR <15:00> on the SBI when a CPU command/address is transmitted. Used in conjunction with a loopback read to test the SBIA and SBI nexus arbitration logic.
<02>	FORCE MAINTENANCE TR	SS24 FORCE MAINT TR – Unconditionally asserts the TR corresponding to SBI silo comparator register <15:00>.
<01>	FORCE ISR DATA	SS24 FORCE ISR DATA – Used to enable the SBIA to respond to an interrupt summary read (ISR) to check the circuitry that sets priorities on the ISR data and generates the vectors. During the response cycle of an ISR, the SBIA enables the write data latch to be transmitted on the SBI along with a TAG, MASK, and ID of 0.
<00>	USE MAINTENANCE ID	SS24 USE MAINT ID – Enables the use of the maintenance ID, bits <27:23> for diagnostic purposes (see bits <27:23>).

3.14.13 SBI Unjam Register

The SBI unjam register bits are given in Figure 3-45 and defined in Table 3-28.

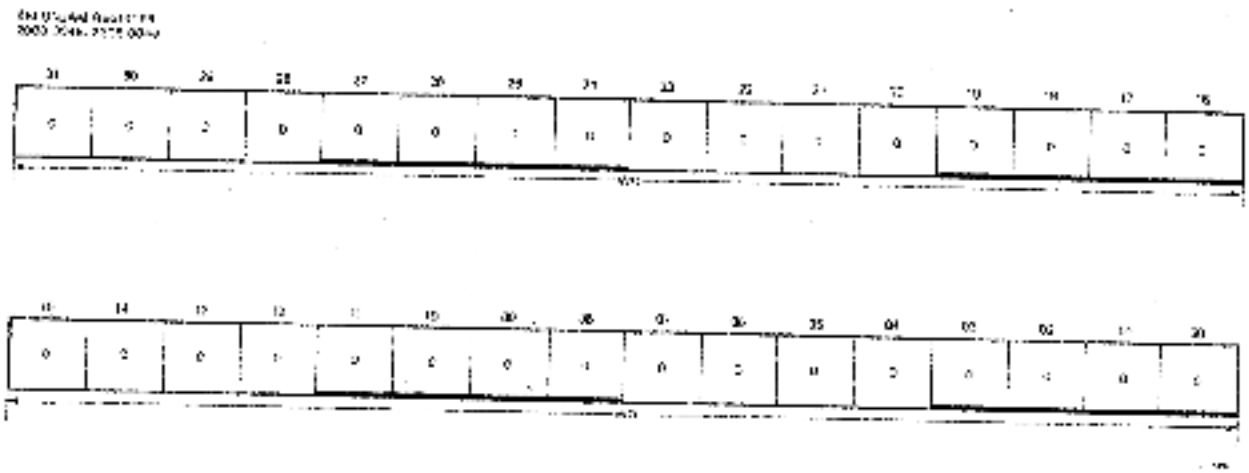


Figure 3-45 SBI Unjam Register

Table 3-28 SBI Unjam Register Bit Definitions

Bit	Definition
<31:00>	(SBAQ) – The SBI unjam register does not exist as a hardware register. When the SBIA decodes the unjam register address for a CPU write to the unjam register, the unjam sequence is initiated. If this register is read, the contents show as all 0s, provided by the zero fill logic on SS36. For a read, the unjam sequence will not be done.

3.14.14 SBI Quadclear Register

The SBI quadclear register bits are given in Figure 3-46 and defined in Table 3-29.

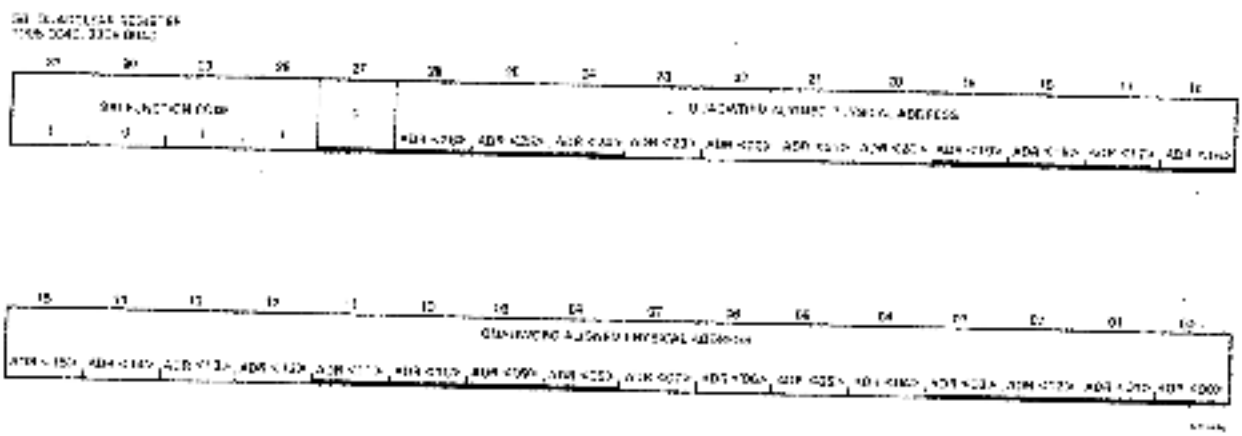


Figure 3-46 SBI Quadclear Register

Table 3-29 SBI Quadclear Register Bit Definition

Bit	Name	Definition
<31:28>	SBI FUNCTION CODE	The purpose of the quadclear register is to clear ECC errors in SBI memory. The quadclear register does not exist as a physical register. When the CPU writes the SBI quadclear register, the CPU register address, in the command/address, is decoded as a quadclear operation. The write data is used to generate the SBI command/address. Two longwords of all zero data are supplied by the SBIA. Bits <31:28> must be 1011, as they become the SBI function code, extended write masked. If this register is read, the quadclear is not done, and the read data is all 0s supplied by the zero fill logic.
<27>		Written as 0 because the address must be a memory address, not an I/O address. Also read as a 0 provided by the zero fill logic.
<26:00>	QUADWORD ALIGNED PHYSICAL ADDRESS	ADR <26:00> – Become the quadword address in the command/address. Also read as a 0 provided by the zero fill logic.

3.14.15 SBI Vector Register

The SBI vector register bits are given in Figure 3-47 and defined in Table 3-30.

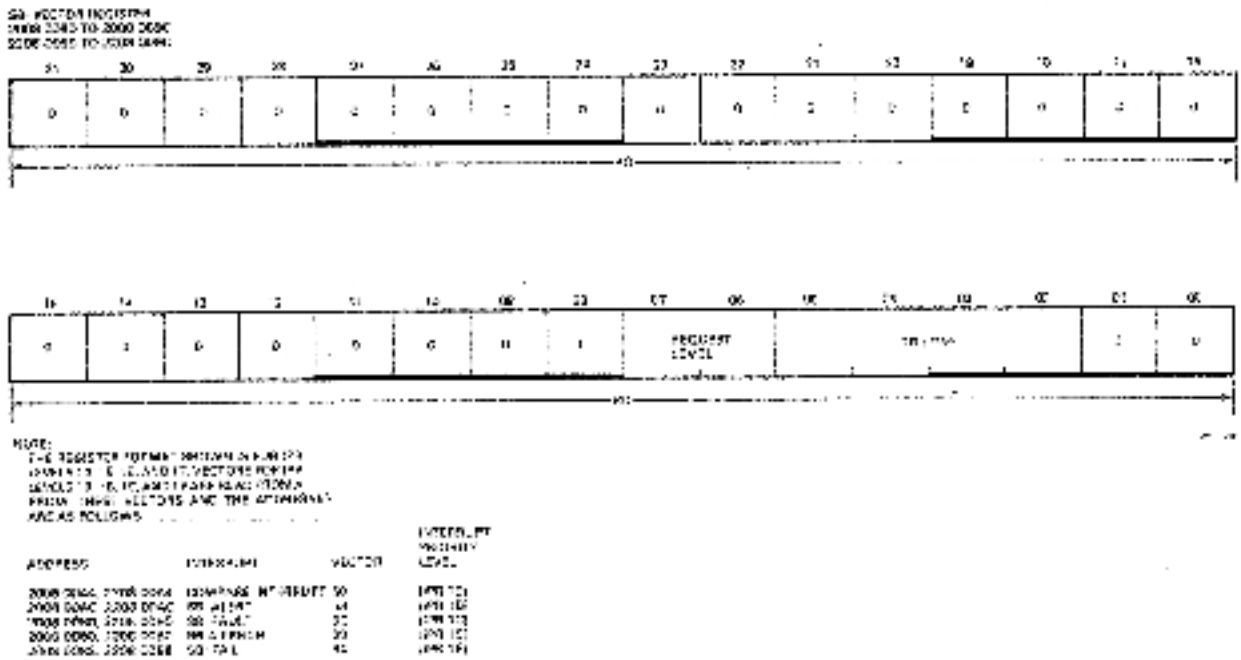


Figure 3-47 SBI Vector Register

Table 3-30 SBI Vector Register Bit Definitions

Bit	Name	Definition
<31:00>	VECTOR	The CPU reads the appropriate vector register in response to the arbitrated interrupt priority requests, which it uses, along with the I/O adapter number, to build the vector register. If the interrupt being serviced originated on the SBI, the vector is made up of the interrupt priority request level and the TR level. If the interrupt being serviced originated on the SBIA, the vector is read from a 32 × 8 PROM (see Tables 3-8 and 3-9 and Figure 3-16).
<31:12>	ZERO	(SS36) – Provided by the zero fill logic.
<11:09>	ZERO	(SS36 or SS31) – If the interrupt being serviced originated on the SBI, these 0s are forced by ground potentials in the hardware on SS31. If the interrupt is a local SBIA interrupt, these 0s are provided by the zero fill logic.
<08>	LOGIC ONE	(SS31) SS31 BUS REG <08> – Tied to +3 V for an SBI interrupt and always read as a logic 0 from the PROM.
<07:06>	REQUEST LEVEL	SS31 BUS REG <07:06> – Provided by the two least significant address bits, which correspond to the request level, for an SBI interrupt. Provided by the PROM for a local SBIA interrupt.
<05:02>	TR LEVEL SS31	BUS REG <05:02> – The TR level is represented by bits <05:02> if the interrupt is an SBI interrupt. If the interrupt is a local SBIA interrupt, these bits are provided by the PROM.
<01:00>	ZERO	SS31 BUS REG <01:00> – Read as 0s provided by the PROM for local SBIA interrupts, or inverting +3 V for an SBI interrupt.

APPENDIX A
ABUS PROTOCOL

A.1 ABUS INTERFACE

Appendix A is a review of the ABus interface signals as shown in Figure A-1. The ABus signals contained in the command/address, write data, or read data may be found in Figures A-2, A-3 and A-4.

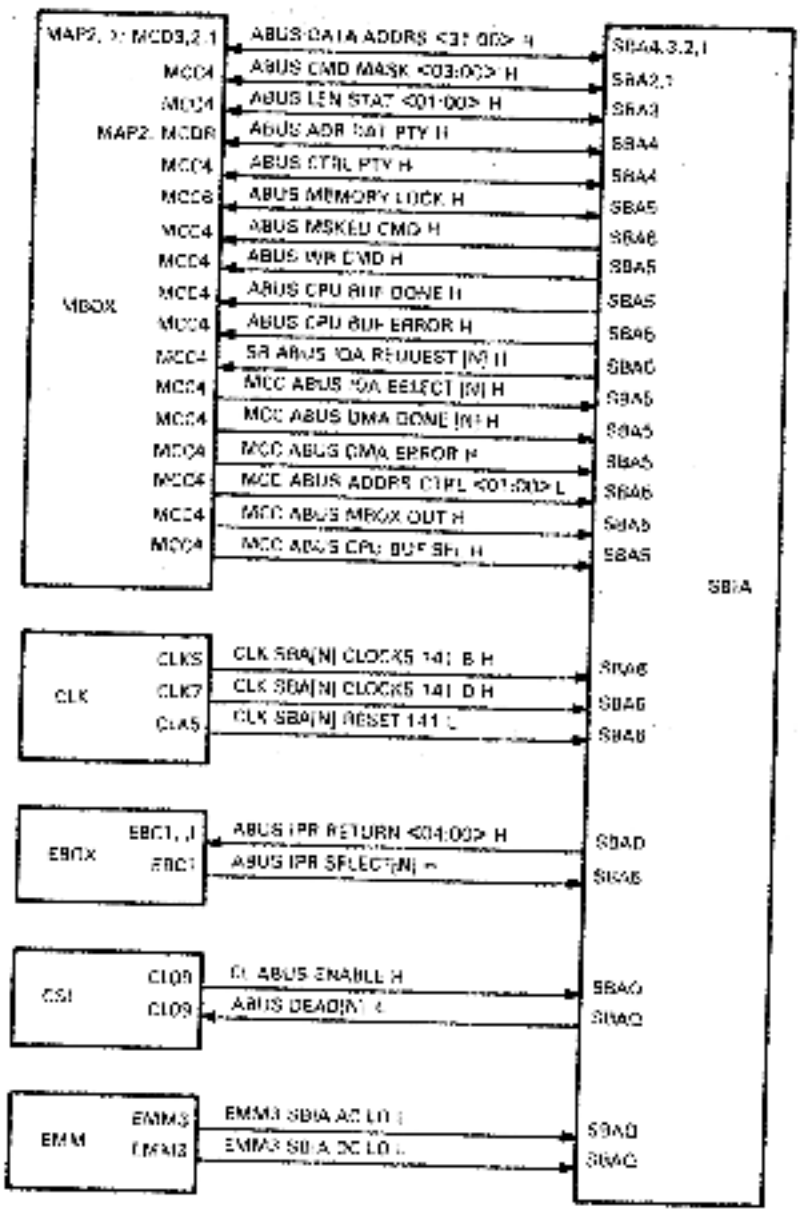


Figure A-1 ABus Interface

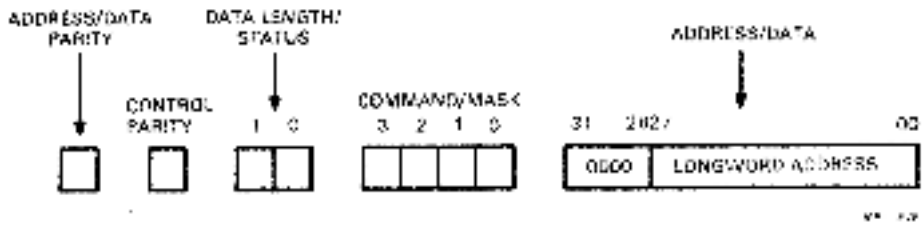


Figure A-2 ABus Command/Address Cycle Format

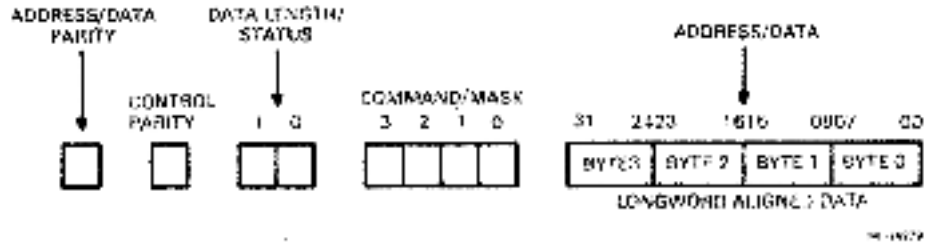


Figure A-3 ABus Write Data Cycle Format

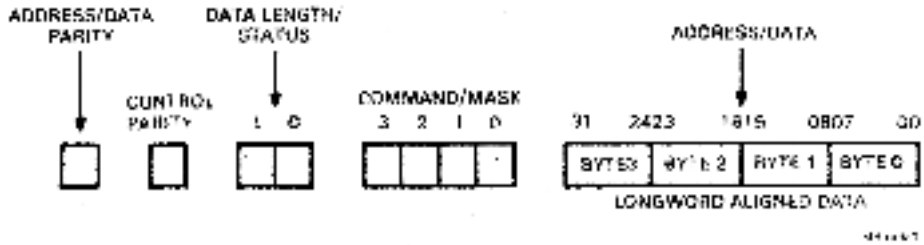


Figure A-4 ABus Read Data Cycle Format

A.1.1 MBox/SBIA Interface

ABUS DATA ADDRS <31:00> H – These 32 lines carry 32 bits of data for read data or write data cycles, or a 28-bit longword address for command/address cycles.

ABUS CMD MASK <03:00> H – The meaning of the command/mask bits depends on the type of ABus cycle as follows.

1. Command/address cycle: The command/mask bits designate an ABus command when transferred on the command/address cycle according to Table A-1 and Figure A-2. A DMA command may differ from a CPU command.

Table A-1 ABus Commands

Command/Mask <03:00>	Command	Command Application
0001	Read	CPU/DMA
0010	Read lock	CPU/DMA
0100	Read modify	CPU
1000	Write	DMA
1101	Write mask	CPU/DMA
1110	Write mask unlock	CPU/DMA

2. Write data cycle: The mask bits indicate which byte is to be written. If MASK<03> is set, byte 3 (bits <31:24>) is to be written. If MASK<01> is set, byte 1 (bits <08:15>) is to be written, and so on (see Figure A-2).

3. Read data return cycle: The command/mask field is not used for the read data return cycle (see Figure A-4 for the ABus read data cycle format).

ABUS LEN STAT <01:00> H – The use of this field also depends on the type of ABus cycle (see Figures A-2, A-3, and A-4).

1. CPU command/address cycle: The length/status bits are used for a CPU read or write to designate which words, odd or even, or if a longword is to be read, according to Table A-2.

Table A-2 Length/Status for CPU Read/Write

LEN STAT <01:00>	CPU Read/Write Function
01	Even word
10	Odd word
11	Longword

2. DMA command/address cycle: During a DMA command/address cycle, the length/status bits indicate the number of bytes to be involved in the DMA transfer according to Table A-3.

Table A-3 Length/Status for DMA Command/Address Cycle

LEN STAT <01:00>	DMA Command/Address Function
00	Longword
01	Quadword
10	Octaword

3. Write data or read data return cycle: During either of these two ABus cycles, the length/status bits indicate whether the data is good or not according to Table A-4.

Table A-4 Length/Status for Data Cycles

LEN STAT <01:00>	Data Cycle Function
00	Good data
11	Bad data (uncorrectable error)

ABUS ADR DATA PTY H – ABUS ADR DATA PTY H is odd parity computed over the longword address (command/address cycle) or data bits (data cycle), ABUS DATA ADDRS <31:00>.

ABUS CTRL PTY H – This odd parity bit is computed over the command/mask and length/status bits.

ABUS MEMORY LOCK H – ABUS MEMORY LOCK is asserted by the MBox when it processes a CPU read lock request. It is asserted by the SBIA for a DMA interlock read. ABUS MEMORY LOCK will remain asserted until a write unlock is received or a timeout occurs.

ABUS MSKED CMD H – This bit, when set, tells the MBox that the ABus command is for a masked operation. It allows the MBox microcode to branch before decoding the command field in the command/address.

ABUS WR CMD H – Like the previous bit, ABUS WR CMD allows the MBox microcode to branch before decoding the command field in the command/address. This bit indicates that the ABus command is for a write.

ABUS CPU BUF DONE H – ABUS CPU BUF DONE is asserted by the SBIA to inform the MBox that a CPU read or write transaction is complete. A CPU write is complete when the nexus has acknowledged the command/address and write data. A CPU read is complete when the read data is placed in the register file.

ABUS CPU BUF ERROR H – ABUS CPU BUF ERROR is asserted any time a CPU read/write is aborted for any of these conditions.

1. There is a parity error when the command/address or write data is removed from the register file for a CPU read/write.

2. The address specified in the CPU command/address is an SBI address but the SBI is not enabled, or the address is not a valid SBI or SBIA register address.

3. The SBIA is unable to gain control of the SBI at the CPUs transfer request level within 102.4 μs.

4. The addressed SBI nexus continually transmits a busy response, or does not transmit any response, to a CPU command/address for 102.4 μs after the CPU/SBI state machine leaves the idle state.

5. The SBIA receives an error response for a CPU command/address from the addressed nexus.

6. The SBIA does not receive the proper number of acknowledges for command/address and write data within 102.4 μs.

7. The SBIA does not receive read data, for a CPU read, within 102.4 μs of the acknowledge for the command/address.

SB ABUS IOA REQUEST [N] H – Each SBIA asserts its SB ABUS IOA REQUEST line when it needs MBox service for a DMA transfer. For a DMA write, it will be asserted when both the command/address and write data have been written into the register file. For a DMA read it is asserted when the command/address has been written into the register file. It is dropped when the MBox loads the register file address to read the command/address for the DMA request.

MCC ABUS IOA SELECT [N] H – The MBox will assert MCC ABUS IOA SELECT for the SBIA adapter it will service. This signal is used to enable access to the register file.

MCC ABUS DMA DONE [N] H – The MBox asserts MCC ABUS DMA DONE to notify the SBIA being serviced that the MBox has completed its portion of the DMA transaction. For a DMA write, the MBox has successfully stored the write data. For a DMA read, the MBox has loaded the read data return words into the register file.

MCC ABUS DMA ERROR H – The MBox asserts MCC ABUS DMA ERROR when it determines that it cannot process the command/address received from the SBIA. Its assertion will clear the request in the SBIA.

MCC ABUS ADDRS CTRL <01:00> L – The address control bits are used in the selected SBIA (MCC ABUS IOA SELECT [N] asserted) to control the loading, holding, or incrementing of the register file address according to Table A-5. These bits are asserted low, and in the table a logic 1 indicates an asserted signal.

Table A-5 MCC ABUS ADDRS CTRL

ADDRS CTRL <01:00> L	Register File Function
0 0	Hold address
0 1	Increment address
1 0	Not used
1 1	Load address

MCC ABUS MBOX OUT H – MBox out controls whether the SBIA register file, for the SBIA selected by SB ABUS IOA SELECT [N], will be read or written. It is also used, with MCC ABUS CPU BUF SEL H, to control the two least significant bits of the register file address being loaded by the MBox (see Table A-6). If MCC ABUS MBOX OUT H is asserted (a logic 1), it indicates that data will be coming out of the MBox and the register file will be written. If MCC ABUS MBOX OUT H is a logic 0, data will be going to the MBox, and the register file will be read. If MBox out is not asserted, the SBIA cannot drive the ABus data lines.

MCC ABUS CPU BUF SEL H – This signal is used, along with MCC ABUS MBOX OUT H, to control the register file address loaded by the MBox in the selected SBIA. For DMA transactions, the two most significant bits select the DMA transaction buffer that is responsible for the DMA request (see Table A-6). All addresses in the table are assumed to be the address loaded when both address control bits are asserted.

A.1.2 Clock/SBIA Interface

CLK SBA[N] CLOCK5 141 B H – The ECL clock phase B to the SBIA is used to clock the ECL logic on the SBA module. This clock line is called CLK SBIA[N] CLOCK5 141 B H on the clock module.

CLK SBA[N] CLOCK5 141 D H – The ECL clock phase D to the SBIA is used to clock the ECL logic on the SBA module. This clock line is called CLK SBIA[N] CLOCK5 141 D H on the clock module.

CLK SBA RESET 141 L – This is a master reset from the clock module.

A.1.3 EBox/SBIA Interface

ABUS IPR SELECT[N] H – ABUS IPR SELECT is used by the EBox to poll the IO adapters for pending interrupts. The EBox polls the adapters in modulo 4 order; therefore, even though there are only two SBIAs, each SBIA will be polled every fourth ABus cycle.

ABUS IPR RETURN <04:00> H – ABUS IPR RETURN <04:00> is the encoded value of the highest priority interrupt the IO adapter has pending, when polled by the EBox, with the assertion of ABUS IPR SELECT[N].

A.1.4 Console/SBIA Interface

CI ABUS ENABLE H – This bit is controlled by the console with MCSR<01>. When this bit is asserted, the IO adapters are enabled. When the register bit is reset, it forces an initialize in both SBIAs. It is used while the system is being booted up to prevent undefined states from having an effect on the IO adapters. It is also used to prevent CPU diagnostics from disturbing the SBI nexus.

ABUS DEAD[N] L – ABUS DEAD is asserted as a result of the assertion of SBI FAIL or for diagnostics, by the setting of diagnostic register bit <06>. It is used to forward a reboot request from another processor on the CI to the console, or to inform the console of an SBI power failure. It interrupts the console.

A.1.5 EMM/SBIA Interface

EMM3 SBIA AC LO L – AC LO from the EMM is used to assert BUS SBI FAIL on the SBI.

EMM3 SBIA DC LO L – DC LO from the EMM is used to assert BUS SBI DEAD and force an initialize within the SBIA.

Table A-6 Register File ECL Address Control

MCC ABUS CPU BUF SEL H	MCC ABUS MBOX OUT H	Register File ECL Address	Comments
0	0	00*	Prepare to read the DMA command/address
0	1	00*	Prepare to write a CPU command/address for an ABus diagnostic cycle
1	0	0011	Prepare to read the CPU read data return word
1	1	0010	Prepare to write the CPU command/address

* Bit determined by which DMA transaction buffer requested service. For an ABus diagnostic cycle, these bits equal 00.

APPENDIX B
SBI PROTOCOL

B.1 SBI SIGNAL NAMES

Appendix B is a brief review of the SBI. The technician should have a working knowledge of the SBI. If not, refer to the technical description for one of the VAX 11/780 SBI adapters.

Table B-1 contains a list of SBI signals and a brief description of each. The signals are separated by signal groups. These same signals are shown in Figure B-1.

Table B-1 SBI Signal Names and Description

Field	Description
Arbitration Group	
Arbitration Field [TR <15:00>]	Establishes a fixed priority among nexus for access to and control of the information transfer path
Information Transfer Group	
Information Field [B<31:00>]	Bidirectional lines that transfer data, command/address, and interrupt information between nexus
Mask Field [M<03:00>]	Encoded to indicate that a particular byte within the data field is to be read or written. With the read data, the mask bits indicate if the data is correct or in error
Identifier Field [ID<02:00>]	Indicates the logical source or destination of information contained in B<31:00>
Tag Field [TAG<02:00>]	Determines if the SBI cycle is for a command/address, read data, write data, or ISR

Table B-1 SBI Signal Names and Description (Cont)

Field	Description
Response Group	
Confirmation Field [CNF<01:00>]	The receiving nexus specifies its response to an SBI cycle: 00 = no response; 01 = acknowledge; 10 = busy; and 11 = error
Function Field [F<03:00>]	Specifies the command code. The function field is bits <31:28> of the command/address (see Figure B-4)
Parity Field [P<01:00>]	Indicates even parity over the SBI information. P0 is computed over the information in B<31:00> while P1 is computed over M<03:00>, ID<04:00>, and TAG<02:00> (see Figure B-2)
Fault Field [FAULT]	A cumulative SBI error line indicating that a nexus has detected an error condition
Interrupt Request Group	
Request Field [REQ<07:04>]	Each signal represents a level of priority whereby a nexus requests interrupt service
Alert Field [ALERT]	A signal that allows SBI memory to interrupt due to a power loss
Control Group	
Clock Field [CLOCK]	Six control lines that are used to generate the SBI clock signals
Fail Field [FAIL]	The assertion of AC LO within a nexus causes the assertion of FAIL on the SBI
Dead Field [DEAD]	The assertion of DC LO within a nexus causes the assertion of DEAD on the SBI
Unjam Field [UNJAM]	A reset signal to all SBI nexus
Interlock Field [INTLK]	A signal that indicates that a shared location is being modified by a nexus

Table B-1 SBI Signal Names and Description (Cont)

Field	Description
Unused Signals	Two unused lines
Multiprocesser [MP<02,01>]	Two additional unused lines
Spare [SPARE<01:00>]	

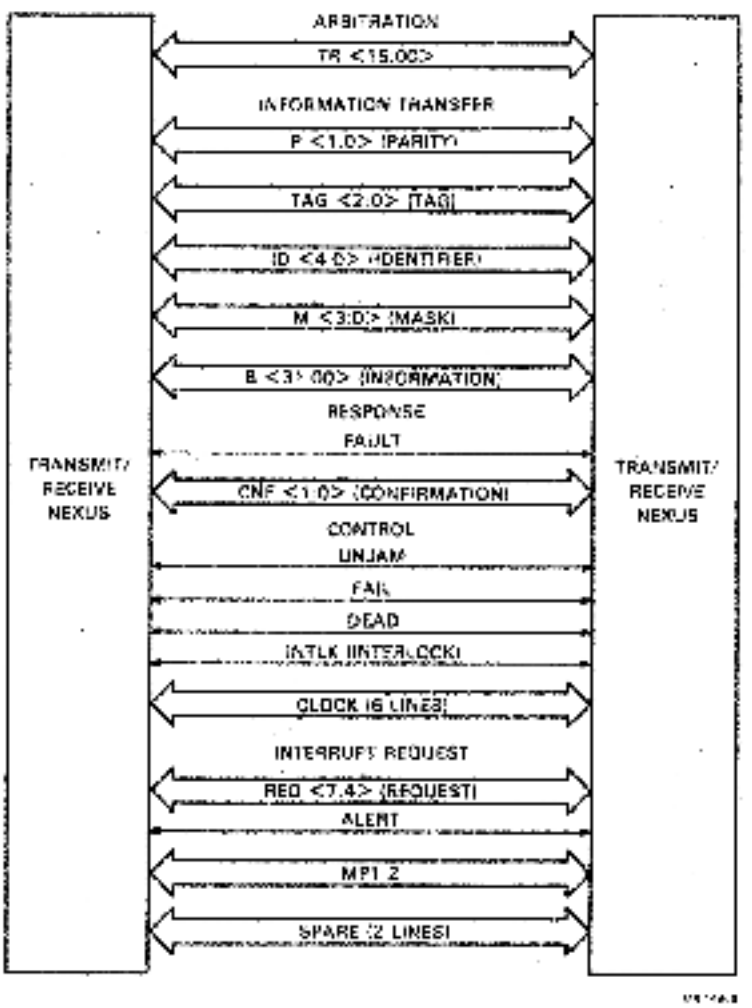


Figure B-1 SBI Signal Names

B.2 SBI PARITY COMPUTATION

Figure B-2 shows how even parity is computed over the SBI information.

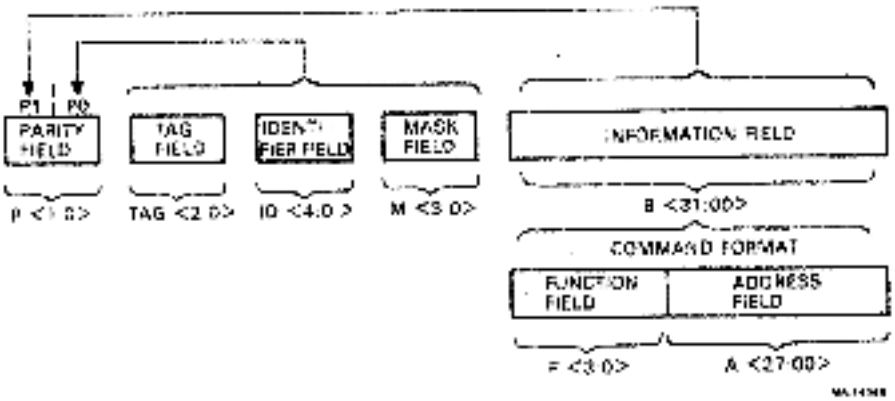


Figure B-2 SBI Parity Field Configuration

B.3 COMMAND ADDRESS FORMAT

Figure B-3 shows the format for a command/address cycle, and Figure B-4 shows the various SBI functions (command codes) and indicates whether the mask field is used or not.

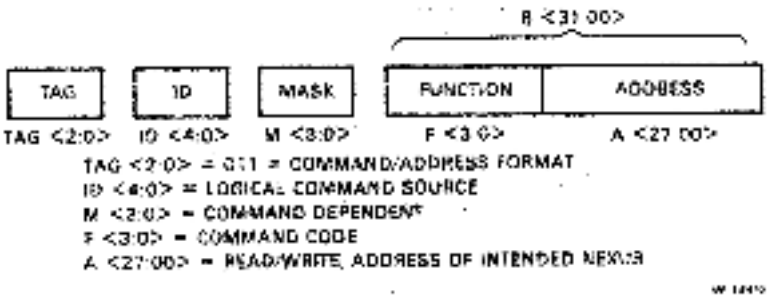
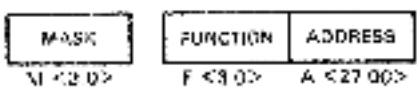


Figure B-3 SBI Command/Address Format



MASK USE	FUNCTION CODE	FUNCTION DEFINITION
IGNORED	0000	RESERVED
USED	0001	READ MASKED
USED	0010	WRITE MASKED
IGNORED	0011	RESERVED
USED	0100	INTERLOCK READ MASKED
IGNORED	0101	RESERVED
IGNORED	0110	RESERVED
USED	0111	INTERLOCK WRITE MASKED
IGNORED	1000	EXTENDED READ
IGNORED	1001	RESERVED
IGNORED	1010	RESERVED
USED	1011	EXTENDED WRITE MASKED
IGNORED	1100	RESERVED
IGNORED	1101	RESERVED
IGNORED	1110	RESERVED
IGNORED	1111	RESERVED

Figure B-4 SBI Command Codes

B.4 READ DATA FORMATS

Figure B-5 shows the three formats for read data.

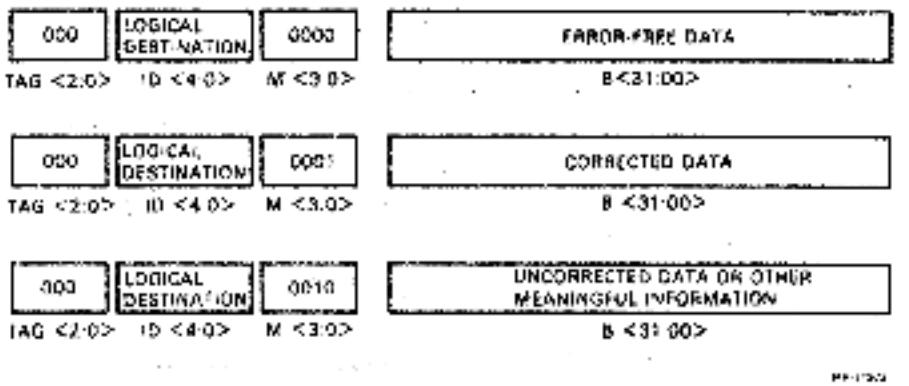


Figure B-5 Read Data Format

B.5 WRITE DATA FORMAT

Figure B-6 shows the format for SBI write data.

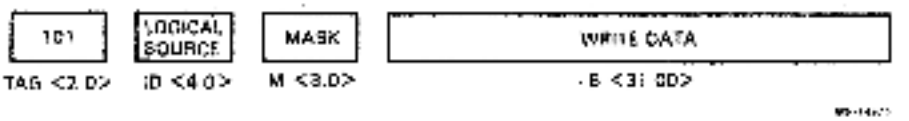


Figure B-6 Write Data Format

B.6 INTERRUPT SUMMARY READ FORMAT

Figure B-7 shows the format for interrupt summary read (ISR) and the response.

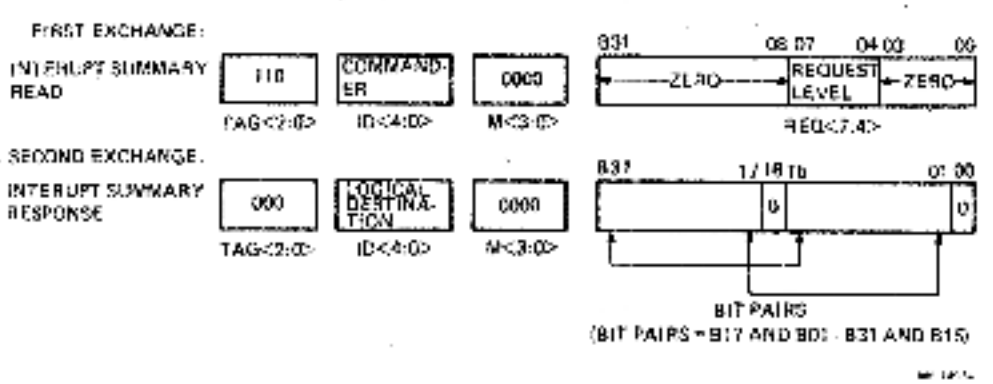


Figure B-7 Interrupt Summary Formats

B.7 EXTENDED READ FORMAT

Figure B-8 shows the SBI cycles for an extended read. Note that the mask bits are not used with an extended read.

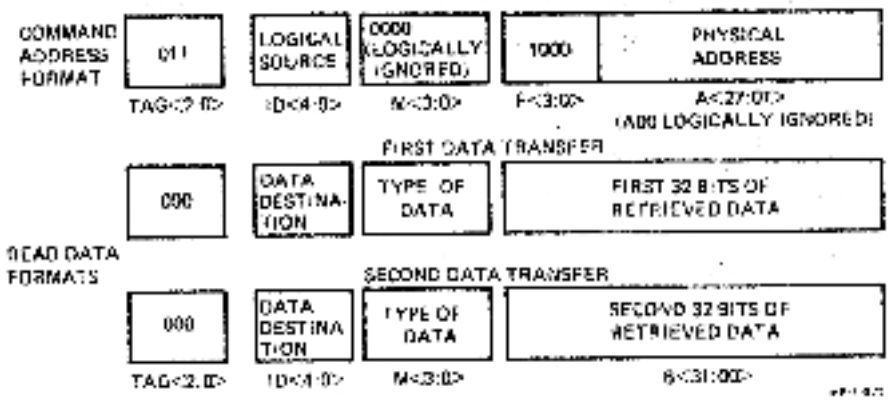


Figure B-8 SBI Cycles for Extended Read

B.8 EXTENDED WRITE FORMAT

Figure B-9 shows the SBI cycles for an extended write masked. The mask bits for the first write data longword are with the command/address. The mask bits for the second write data longword are with the first write data longword.

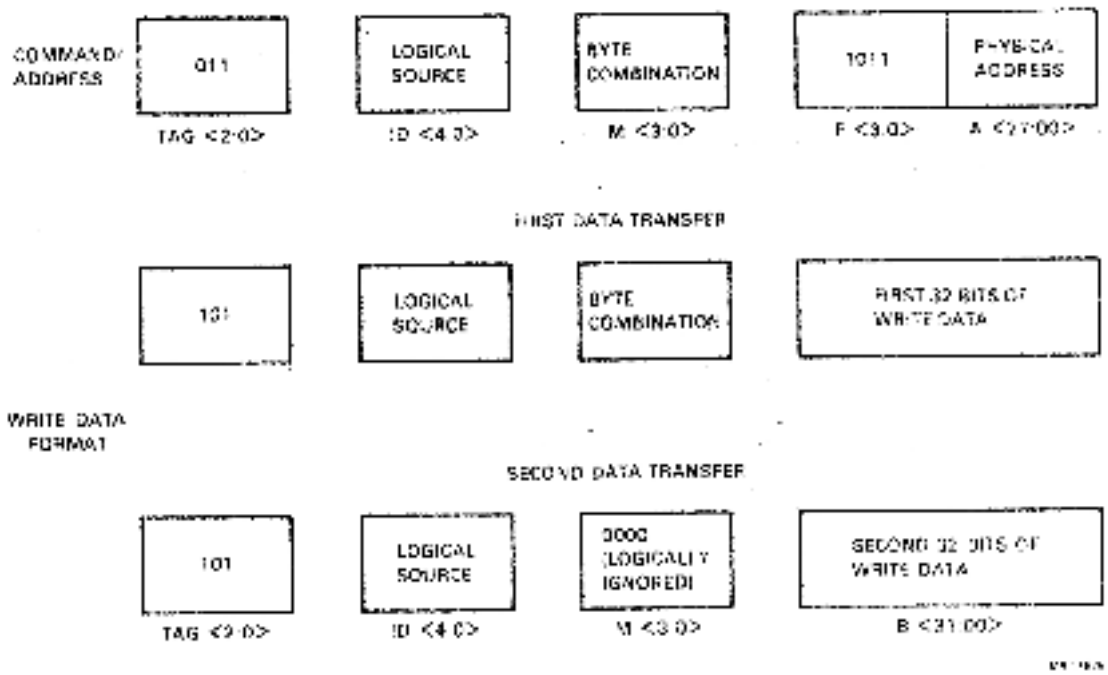


Figure B-9 SBI Cycles for Extended Write Masked

B.9 CLOCK SIGNALS

Figure B-10 shows the six SBI clock signals (TPH, TPL, PCLKH, PCLKL, PDCLKH, PDCLKL) and how they are used to generate the four SBI clock phases (T0, T1, T2, T3).

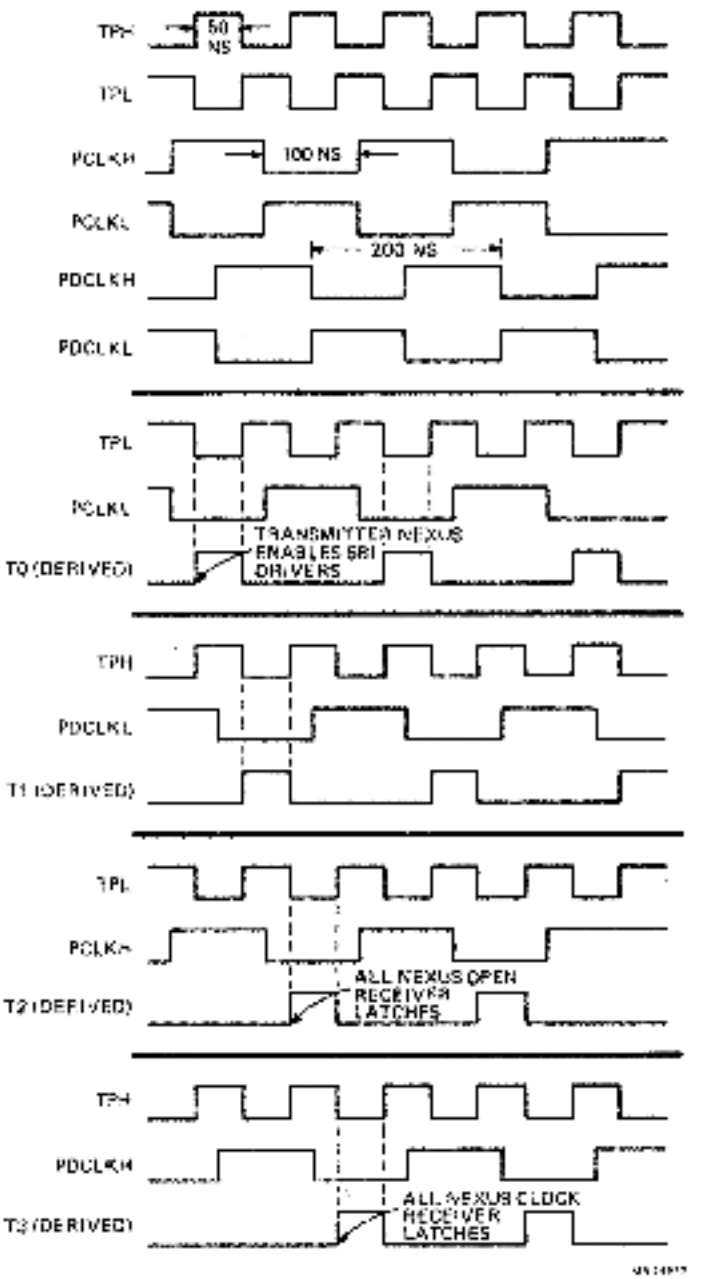


Figure B-10 SBI Clock Signals

APPENDIX C
SBI ARBITRATION

C.1 SBI PRIORITY ARBITRATION

Appendix C covers the method used to allow SBI nexus access to the SBI. The primary hardware involved is the DC101 priority arbitration chip. Each SBI nexus capable of accessing the SBI must have a DC101. The SBIA has two DC101s because the SBIA can access the SBI for CPU read/writes and DMA reads.

Figure C-1, DC101 Priority Arbitration Chips, shows the two DC101s in the SBIA – one for CPU transfers, the other for DMA reads. Refer to Figure C-1 for detail of the DC101s.

C.1.1 Priority Selection

Each SBI nexus is assigned an SBI priority, the nexus transfer request level (TR). Refer to the *VAX 8600/8650 System Maintenance Guide* (EK-86XVI-MG), for recommended priority levels. SBI transfer requests range from 0 to 16, with TR00 having the highest priority and TR16 the lowest priority. No SBI nexus is assigned priority TR00, which is reserved, to allow the nexus to hold the SBI for an additional cycle. Also, although there is no TR16 request line, if no TR line (TR00 through TR15) is asserted, a nexus assigned TR16 can get control of the SBI.

Priorities are selected according to the 2's complement of TR SEL 8, TR SEL 4, TR SEL 2, and TR SEL 1. Backpanel jumpers are used to connect these lines to ground or +3 V potential. In the SBIA, the DC101 for DMAs has each TR SEL line hardwired to +3 V, or 1111, to provide a priority of TR01. The TR SEL lines for the CPU are connected to +3 V, +3 V, +3 V, and ground, or 1110, which provides a priority of TR02. Table C-1 shows the binary configuration for various TR levels. It also indicates the jumper connection necessary to connect XMIT TR (DC101) to the BUS SBI TR line for the CPU TR. The XMIT TR output for the DMA DC101 is hardwired to BUS SBI TR01, C47, so no jumper is necessary.

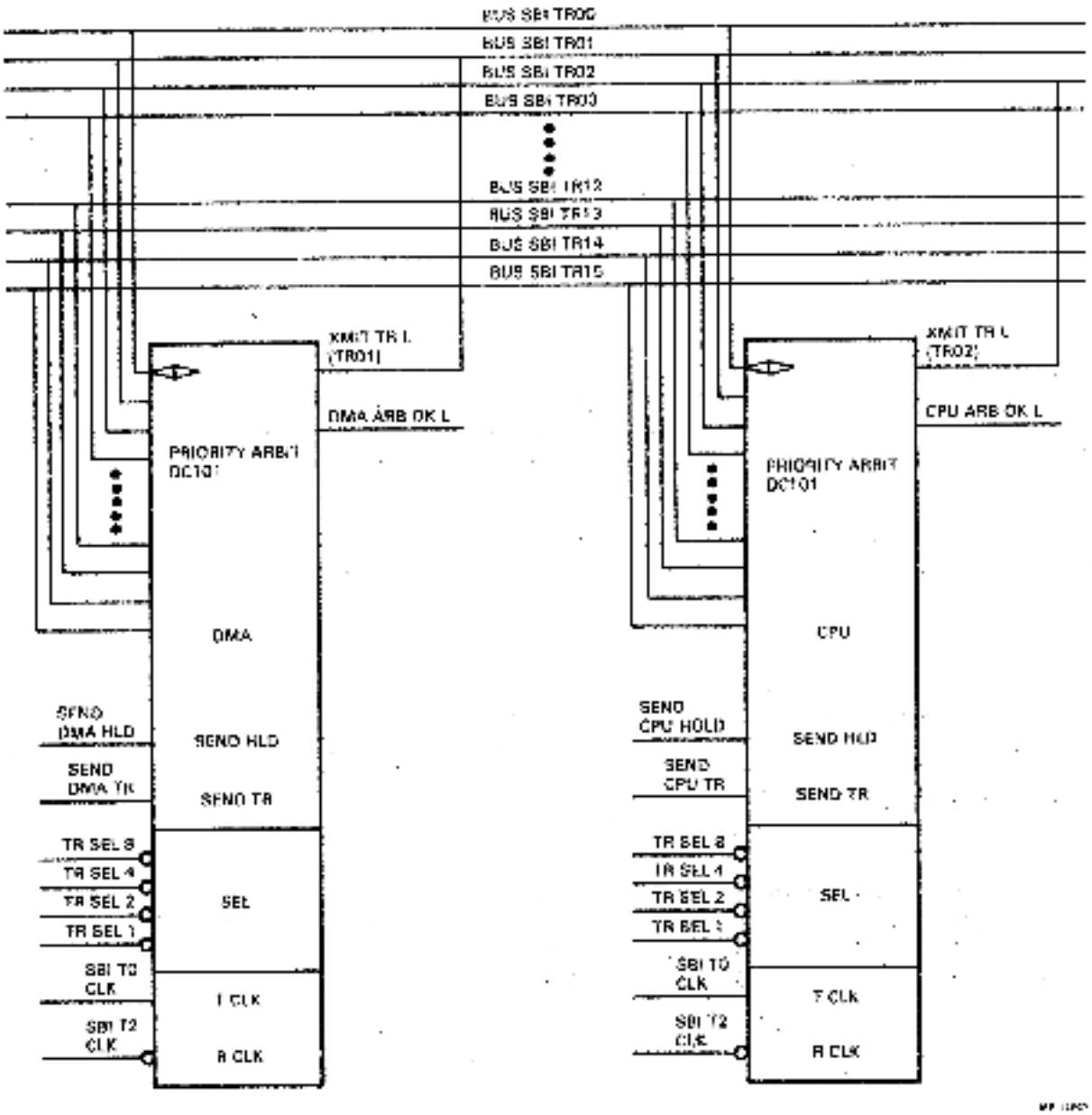


Figure C-1 DC103 Priority Arbitration Chips

Table C-1 XMIT TR Jumpers

TR SEL 8 4 2 1	DEVICE TR	Transmit TR Jumper Needed	Jumper from XMIT TR L, C83* to BUS SBI TR ____ L on Pin ____	
L L L L	16	No	–	–
L L L H	15	Yes	15	C84
L L H L	14	Yes	14	C77
L L H H	13	Yes	13	C73
L H L L	12	Yes	12	C75
L H L H	11	Yes	11	C71
L H H L	10	Yes	10	C69
L H H H	09	Yes	09	C67
H L L L	08	Yes	08	C65
H L L H	07	Yes	07	C63
H L H L	06	Yes	06	C59
H L H H	05	Yes	05	C57
H H L L	04	Yes	04	C55
H H L H	03	Yes	03	C51
H H H L	02†	Yes	02	C53
H H H H	01‡	Yes	01	C47

*Jumper connection made on SBS backpanel
†Normal configuration for CPU DC101
‡Normal configuration for DMA DC101

C.1.2 DC101 Operation

The DC101s operate according to Figure C-1 and the following description.

When a nexus desires to transfer information on the SBI, it must first gain control of the SBI. The assertion of the SEND TR input to the DC101 causes XMIT TR L to be asserted. This output is jumpered to the BUS SBI TR line for the required transfer request priority. In Figure C-1, SEND TR is SEND DMA TR for DMA read transactions or SEND CPU TR for CPU reads/writes.

The DC101s receive BUS SBI TRs at T2, latching them at -T2. Each DC101 that has SEND TR asserted compares the received BUS SBI TR lines to the priority level selected for that nexus. If the nexus priority level is higher than the received BUS SBI TR (lower number), the nexus gains control of the SBI. ARB OK L is asserted to indicate that the nexus has control of the SBI and can transmit on the next cycle. The nexus has control of the SBI for only one cycle.

There are cases when a nexus needs control of the SBI for more than one cycle. For example, a CPU write transfers a command/address followed by write data. In this case, the SBIA needs control of the SBI for two cycles.

The SBIA (SEND CPU TR) gains control of the SBI for the first cycle (TR02) by arbitrating for bus control. When the SBI is needed for an additional cycle, SEND HOLD (SEND CPU HOLD for CPU transactions) is asserted. This causes the assertion of TR00. Notice the diamond in Figure C-1 next to the BUS SBI TR00 line; this is a bidirectional line. The transmittal of TR00 by the CPU and reception of TR00 by any other nexus prevent all nexuses from asserting ARB OK for as long as TR00 is asserted. In the SBIA, SEND CPU HOLD is asserted for one cycle to allow the SBIA to transfer the write data.