EK–KDA5Q–TD–001

KDA50-Q

TECHNICAL DESCRIPTION

digital

EK-KDA5Q-TD-ØØ1

KDA5Ø-Q
TECHNICAL DESCRIPTION

FOR INTERNAL USE ONLY

Digital Equipment Corporation
Colorado Springs, Colorado

First Edition, February 1985

The material in this manual is for informational purposes and is subject to change without notice. DIGITAL assumes no responsibility for any errors which may appear in this manual.

Digital Equipment Corporation does not grant licenses to make, use or sell equipment as described in this manual, and makes no claim that use of its products with those of other manufacturers will not infringe on existing or future patent rights.

This equipment generates, uses, and may emit radio frequency energy. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC rules for operation in a commercial environment. If this equipment is operated in a residential area, the user, at his own expense, may be required to take corrective measures.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DEC	DECnet	RQDX
DECUS	DECsystem-1Ø	RC25
HSC5Ø	DECSYSTEM-2Ø	VT
digital	DECwriter	KDA5Ø-Q
PDP	RA8Ø	RSTS
UNIBUS	RSX	UDA5Ø
VAX	MASSBUS	VMS
RA81	RA6Ø	

CONTENTS

FIGURES

TABLES

CHAPTER 1
INTRODUCTION

1.1 KDA5Ø-Q DISK CONTROLLER

The KDA5Ø-Q is an intelligent controller which interfaces up to four 16-bit RA series disk drives with any CPU that utilizes a QBUS. The physical controller-to-drive interface is accomplished using a radial bus instead of the conventional daisy-chain (serial) method. Radial configuration means a separate I/O cable is attached to each disk drive. Figure 1-1 shows the KDA5Ø-Q subsystem configuration.

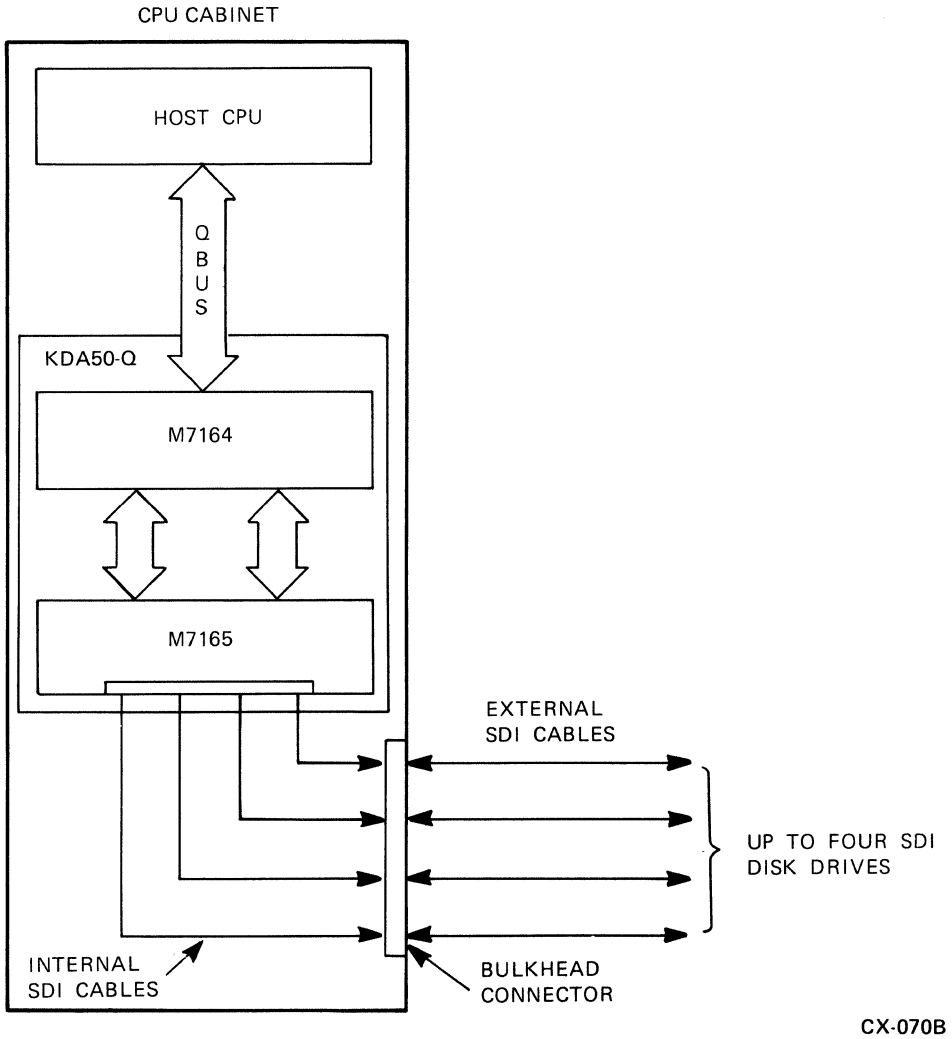


Figure 1-1 KDA5Ø-Q Disk Subsystem Configuration

1.1.1 KDA5Ø-Q Hardware

The SDI module and the processor module make up the KDA5Ø-Q. The SDI module is the communication interface between the KDA5Ø-Q microprocessor module and the disk drives. The processor module (M7164) is the control portion of the KDA5Ø-Q and is responsible for the QBUS interface.

1.1.1.1 SDI Module - Some of the circuitry and functions of the SDI module are as follows:

- Contains a 32K-byte high-speed buffer used during data transfers. The buffer allows controller-to-drive transfers to occur at a higher rate than controller-to-host transfers. This improves performance by minimizing missed disk revolutions due to a buffer-full condition.
- Converts the KDA5Ø-Q buffer format (parallel) to SDI format (serial) and vice versa.
- Generates the real-time error correction code (ECC). This code has a correction capability of up to eight 1Ø-bit error bursts per block (sector).
- Implements the real-time and electrical interface to the SDI, including error detection on the SDI and RAM.

1.1.1.2 Processor Module - Some of the circuitry and functions of the processor module are listed below:

- Performs all KDA5Ø-Q interaction with the QBUS via two registers: the initializing and polling (IP) register, and the status/address (SA) register. The IP register has a default octal address of 172150. The SA register address is always the IP register address plus 2 (172152 in this case). A switch pack is used to set the I/O page register address.
- Reports microcode detected errors through four LEDs on the processor module and four LEDs on the SDI module. The error code reported indicates the module to replace.

Also located on the processor module is a dual microprocessor. It is made up of two 12-bit microprogram sequencers which share a common 16-bit ALU. The combination of the sequencers and the shared ALU enables the dual microprocessor to execute two independent, interleaved microprograms (from Read Only Memory) at the same time. One of the sequencers controls the KDA5Ø-Q to host interaction, and the other controls the KDA5Ø-Q to disk drive interaction. For greater efficiency, one sequencer fetches an instruction while the other executes an instruction.

1.1.2 Functional Microcode

The functional microcode may be divided into two functional flows or streams: the drive control stream which manages the controller-to-disk-drive interface, and the QBUS control stream which manages the controller-to-host interface.

1.1.2.1 Drive Control Stream - Some of the functions that the drive control stream performs are listed below:

- Monitors Attention from the drives. When the drive attention bit has been set and detected, the drive control stream gets the drive status, compares it with the previous status, and takes appropriate action.
- Constructs and sends packets to the disk drives. The packets may be the result of a host request (Read, Write, Replace, etc.) or in response to a drive attention condition.
- Receives and validates packets from the drives.
- Monitors the drive status flags from the QBUS control stream. The drive status flags supply communication between the QBUS control stream and the drive control stream.

- Performs tasks as required by the drive status flags. Some of these tasks are as follows:
  - Initiates Read, Write, Seek, and Head-Select packets to the drive.
  - Reads and verifies the block (sector) header.
  - Performs data transfers between internal RAM and the disk drive.
  - Updates drive status and buffer use flags.
  - Performs data error analysis and recovery.
  - Performs bad block revectoring.

1.1.2.2 QBUS Control Stream - Some of the functions that the QBUS control stream performs are as follows:

- Vectors to the appropriate handling routine in the microcode when the host has a command to send to the KDA5Ø-Q or the KDA5Ø-Q has a response to send to the host.
- Exchanges information packets with the host in memory.
- Validates each packet from the host.
- Constructs the KDA5Ø-Q response packets for transmission to the host.
- Analyzes the drive packets and performs the functions listed below:
  - Decodes the logical block number (LBN) to cylinder, group, track, and sector information.
  - Optimizes seek selection from the outstanding commands.
  - Allocates data buffer space.
  - Computes and stores parameters for each sector transfer.
  - Performs packet error detection.

  - Performs memory mapping for mapped requests.
  - Transfers data between the host and internal memory (including automatic support for block mode memories).

- Performs ECC error correction.
- Polls the command queue at the completion of each command.
- Performs initialization
- Initiates drive control stream packet executions.

1.1.3 Digital Storage Architecture (DSA)

The KDA5Ø-Q belongs to the family of DSA products which implements the Standard Disk Interconnect (SDI). DSA defines the operating rules of mass storage subsystems and how they interface with the host computer. Some of the characteristics of DSA are as follows:

- I/O management has been moved from the host to the controller.
- The host views the disk subsystem as one contiguous string of sectors known as logical blocks. A logical block contains 512 bytes of information.
- The host is not concerned with disk geometry such as cylinder, track, sector, etc.
- Host and subsystem exchange messages using the Mass Storage Control Protocol (MSCP).

1.1.4 Mass Storage Control Protocol (MSCP)

The KDA5Ø-Q Disk Controller is a Mass Storage Control Protocol (MSCP) device, and MSCP is a communication protocol used with intelligent mass storage controllers. MSCP hides device-dependent requirements, such as disk geometry and error recovery strategies, from the host, and so enables one class driver to be replaced by several different device drivers.

To request an I/O operation, the host constructs an MSCP message and sends it to the controller. The MSCP message contains the drive address, the function to be performed, the starting logical block (sector) number, and the amount of data requested. The message does not contain drive geometry information because MSCP hides device-dependent requirements.

When the subsystem receives the request, it independently performs all drive management and data movement, as well as any necessary error recovery. For example, for the recovery technique called revectoring, the KDA50-Q accesses a replacement sector instead of the original sector found to be in error. The sector in error is then marked as bad and replaced via a cooperative process between the host software and the KDA50-Q. Upon completion, the subsystem gives status information by sending the host an MSCP response message. This flow differs from conventional subsystems in which host computer resources would be used to control the drive.

The KDA50-Q Disk Controller Specifications are described in Table 1-1.

Table 1-1 KDA5Ø Specifications

Characteristics	Specifications
Physical components	KDA5Ø module 1 (M7164) KDA5Ø module 2 (M7165) 5Ø-pin flat cable assembly 4Ø-pin flat cable assembly SDI cable assembly I/O bulkhead assemblies
Power consumption	7Ø watts nominal
Heat dissipation	Approximately 246 Btu/hour
Electrical voltage and current requirements	13.5 amps at +5 volts, 3Ø milliamps at +12 volts
QBUS Loading	3.Ø AC / .5 DC Standard Loads
Operating temperature range	1Ø° C to 4Ø° C (5Ø° F to 1Ø4° F) with a temperature gradient of 2Ø° C/hour (36° F/hour)
Operating relative humidity range	1Ø% to 9Ø% with a wet bulb temperature of 28° C (82° F) and a minimum dew point of 2° C (36° F)
Operating altitude range	Sea level to 24ØØ meters (8ØØØ ft). Derate the maximum allowable operating temperature by 1.8° C/1ØØØ meters (1° F/1ØØØ feet) for operation above sea level
Mounting restrictions	Mounts in two quad-height QBUS SPC slots in the following mounting boxes: BA23 rackmount with H349Ø BA11-S or BA11-N
Bulkhead Requirements	Two cutouts used to mount the SDI cable hardware. These cutouts are 2x3 on the H349Ø, but may be different sizes on other boxes. These other sizes must be adapted to the 2x3 size as described in the KDA5Ø-Q USER GUIDE.

1.2 RELATED DOCUMENTATION

DIGITAL customers may order the following KDA5Ø-Q related manuals:

- KDA5Ø-Q USER GUIDE (EK-KDA5Q-UG)
- KDA5Ø-Q SERVICE MANUAL (EK-KDA5Q-SV)
- KDA5Ø-Q TECHNICAL DESCRIPTION (EK-KDA5Q-TD)
- KDA5Ø-Q FIELD MAINTENANCE PRINT SET (MP-Ø1423)

Internal DIGITAL Customers: The User Guide, Service Manual and Technical Description may be ordered directly from Publication and Circulation Services, 1Ø Forbes Road, Nothboro, Massachusetts Ø1532 (RCS Code: NR12, Mail Code: NRØ3/W3).

The Field Maintenance Print Set may be ordered directly from the Software Distribution Center, 444 Whitney Street, Northboro, Massachusetts Ø1532 (RCS Code: MSDC, Mail Code: NRO2-1/J6).

Outside the United States, consult local DIGITAL offices.

CHAPTER 2
QBUS OVERVIEW

2.1 INTRODUCTION

The original LSI-11 bus has 16 bits of addressing, a 16-bit data word, and associated bus control signals. The addressing has been expanded to 18 bits (QBUS) and again to 22 bits (Q22 bus). The differences between these buses are:

- The available addressing.
- The 18- and 22-bit versions support Block mode transfers; the 16-bit version does not.

2.1.1 QBUS Master-Slave Relationship

Communication on the QBUS is asynchronous but remains interlocked because of the master-slave relationship. The bus master has control of the bus and is typically the processor or a DMA device. It initiates bus transactions, and the slave device responds by using bus protocol to acknowledge the transaction, receive data, or transmit data. Communication on the QBUS is interlocked because the slave device must respond to the control signals issued by the bus master before the transfer is complete.

If communications on the QBUS break down because the slave does not respond to the bus master, the master aborts the current bus cycle. On the KDA5Ø-Q, a software timer initiates the abort within ten microseconds after the slave fails to respond.

2.2 QBUS SIGNAL FUNCTIONS

The QBUS signal lines are functionally divided as shown in Table 2-1.

Table 2-1 Summary Of Signal Line Functions

Quantity	Function	Bus Signal Mnemonic
16	Data address lines	BDAL <15:Ø>
2	Memory parity/address lines	BDAL <17:16>
4	Address lines	BDAL <21:18>
6	Address and data transfer control lines	BSYNC, BDIN, BDOUT, BWTBT, BBS7, BRPLY
3	Direct memory access (DMA) control lines	BDMR, BDMG, BSACK
5	Interrupt control lines	BIRQ4, BIRQ5, BIRQ6, BIRQ7, BIAK
6	System control lines	BPOK, BDCOK, BINIT, BHALT, BREF, BEVNT

Address bits <15:ØØ> and the 16 bits of data are time multiplexed over the address/data lines <15:ØØ>. Two additional address bits and the memory parity bits are time multiplexed over address lines <17:16>.

Most signals on the QBUS are bi-directional and can be driven or received at any point along the signal line. When a bi-directional signal line is driven, the signal travels both from the driver to the near end bus terminator and from the driver to the far end bus terminator.

2.2.1 QBUS Priority Chains

The rest of the QBUS signal lines are unidirectional and travel in only one direction. These signals make up the interrupt and DMA grant daisy chains and are listed below:

- BIAKO L (Interrupt Acknowledge Out)
- BIAKI L (Interrupt Acknowledge In)
- BDMGO L (Direct Memory access Grant Out)

- BDMGI L (Direct Memory access Grant In)

The interrupt acknowledge and direct memory access lines are connected in a daisy-chain fashion. The output of one device (BIAKO L or BDMGO L) connects to the input of the next device (BIAKI L or BDMGI L), and the processor asserts BIAK or BDMG in response to interrupt or direct memory access requests. Each of the output signals (BIAKO or BDMGO) is received on a device input pin (BIAKI or BDMGI) and conditionally retransmitted via the device output pin (BIAKO or BDMGO).

2.2.1.1 Interrupt Priority Chain - As an example, assume a QBUS device asserts an interrupt request, but the request is not the highest priority interrupt pending. When the device receives BIAKI L, it retransmits (within 500 nanoseconds) BIAKO L to the next device in the daisy chain. If this QBUS device has the highest priority interrupt pending, it does not retransmit BIAKO L. Instead, the interrupt is accepted and proceeds according to the interrupt protocol described in section 2.3.2.

2.2.1.2 DMA Priority Chain - The DMA priority daisy chain is similar to the interrupt priority daisy chain except that its priority is determined solely by its position on the bus. In other words, the DMA device electrically closest to the processor has the highest priority. For example, the processor transmits BDMGO in response to a DMA request (BDMR L). The nearest (highest priority) DMA device receives BDMGO on the input pin BDMGI. If this device did not issue the DMA request, it transmits BDMGO to the next device (next lower priority) in the daisy chain. The bus mastership protocol is explained in greater detail in section 2.3.3.

2.2.2 QBUS Signals Functional Description

Table 2-2 describes all the QBUS signals and lists their associated bus pin. The signal mnemonic is also listed.

Table 2-2 QBUS Signal Functional Description

Bus Pin	Signal Mnemonic	Signal Function
AA1	BIRQ5 L	Interrupt request priority level 5
AB1	BIRQ6 L	Interrupt request priority level 6

Table 2-2 (cont.)

Bus Pin	Signal Mnemonic	Signal Function
AC1	BDAL16 L	Address line 16 during addressing protocol; memory error data line during data transfer protocol
AD1	BDAL17 L	Address line 17 during addressing protocol; memory error enable during data transfer protocol
AE1	SSPARE1 OR +5B	Special spare - This pin is not assigned or bused in DIGITAL cable or backplane assemblies. It is available for user connection. AE1 may be used for +5V battery (+5B) backup power to keep critical circuits alive during power failures. A jumper is required on LSI-11 bus options to open (disconnect) the +5B circuit in systems that use this line as SSPARE1
AF1	SSPARE2 OR SRUN	Special spare - Not assigned or bused in DIGITAL cable or backplane assemblies. It is available for user connection. In the highest priority device slot, the processor may use this pin for a signal indicating the processor RUN state.
AH1	SSPARE3	Special spare - Not assigned or bused in DIGITAL cable or backplane assemblies. It is available for user connection.
AJ1	GND	System signal reference and DC return.
AK1 AL1	MSPAREA	Maintenance spare - Normally connected together on the backplane at each option location (not a bused connection.)
AM1	GND	System signal reference and DC return.

Table 2-2 (cont.)

Bus Pin	Signal Mnemonic	Signal Function
AN1	BDMR L	Direct Memory Access (DMA) request - A device asserts this signal to request bus mastership.
AP1	BHALT L	Processor Halt - When BHALT L is asserted, the processor responds by going into its halt state (generally console ODT mode.)
AR1	BREF L	Memory refresh - Used during refresh protocol to override memory bank selection decoding and cause all banks to be selected. During block mode transfers, the slave device asserts BREF L with BRPLY L to indicate it can accept another block mode DIN or DOUT cycle.
AS1	+5B or +12B	+12V or +5V battery backup to keep critical circuits alive during power failures. This signal is not bused to BS1 in all DIGITAL backplanes. A jumper is required on all LSI-11 Bus options to open (disconnect) the backup circuit from the bus in systems that use this line at the alternate voltage.
AT1	GND	System signal reference and DC return.
AU1	PSPARE1	Power spare 1 - This pin is not assigned and customer usage is not recommended. If a backplane is busing -12 volts (on pin BB2) and a module is accidentally inserted upside down in the backplane, -12 volts appears on pin AU1. If AU1 is unused on the module, no damage will occur.
AV1	+5B	+5V. Battery backup power - To keep critical circuits alive during power failures.

Table 2-2 (cont.)

Bus Pin	Signal Mnemonic	Signal Function
BA1	BDCOK H	DC Power OK - The power supply generates this signal when sufficient DC voltage is available to sustain reliable system operation. It is part of the powerup, powerdown and boot protocol.
BB1	BPOK H	AC Power OK - The power supply asserts this signal when primary power is normal. If negated during processor operation, a power fail trap sequence is initiated. It is part of the powerup, powerdown, and boot protocol.
BC1 BD1 BE1 BF1	BDAL18 or SSPARE 4 BDAL19 or SSPARE 5 BDAL2Ø or SSPARE 6 BDAL21 or SSPARE 7	Special spares <7:4> in older LSI-11 Bus Systems. Not assigned or bused in non-Q22 BUS cable and backplane assemblies. Address lines <21:18> in Q22 BUS backplane and cable assemblies. CAUTION These pins may have been used as test points in some DIGITAL or customer options. These options must be modified or designated incompatible with Q22 BUS backplanes.
BH1	SSPARE 8	Special spare - not assigned or bused in DIGITAL cable or backplane assemblies available for user connection.
BJ1	GND	System signal reference and DC return.
BK1 BL1	MSPAREB MSPAREB	Maintenance Spares - Normally connected together on the backplane at each option location (not a bused connection).
BM1	GND	System signal reference and DC return.

Table 2-2 (cont.)

Bus Pin	Signal Mnemonic	Signal Function
BN1	BSACK L	A DMA device asserts this signal in response to the BDMGO L signal indicating it accepts bus mastership. The device remains bus master until it negates BSACK L.
BP1	BIRQ7 L	Interrupt request priority level 7.
BR1	BEVNT L	External Event Interrupt Request - The processor latches the leading edge and arbitrates as an interrupt. A typical use of this signal is a line time clock interrupt.
BS1	+12B	+12V DC battery backup power (not bused to AS1 in all DIGITAL backplanes)
BT1	GND	System signal reference and DC return.
BU1	PSPARE2	Power spare 2 - This signal is not assigned a function and is not recommended for use. If a backplane is busing -12 volts (on pin AB2) and a module is accidentally inserted upside down in the backplane, -12 volts appears on pin BU1. If BU1 is unused on the module, no damage will occur.
BV1	+5V	Normal +5V DC system power.
AA2	+5V	Normal +5V DC system power.
AB2	-12V	Optional -12V power for devices requiring this voltage.
AC2	GND	System signal reference and DC return.
AD2	+12V	Normal +12V DC system power.

Table 2-2 (cont.)

Bus Pin	Signal Mnemonic	Signal Function
AE2	BDOUT L	Data output - BDOUT, when asserted, implies that valid data is available on BDAL <15:ØØ> L, and that an output transfer, with respect to the bus master device, is taking place.
AF2	BRPLY L	Slave devices assert BRPLY L in response to BDIN L or BDOUT L, indicating their ability to transfer data according to the appropriate protocol. BRPLY L is also asserted during interrupt acknowledge (IAK) transactions.
AH2	BDIN L	Data Input - BDIN L is used for two types of bus operation: 1. When asserted during BSYNC L time, BDIN L implies an input transfer with respect to the current bus master and requires a response, BRPLY L, from the addressed slave. 2. The interrupt fielding processor initiates interrupt service by asserting TDIN L followed by TIACK L.
AJ2	BSYNC L	Synchronize - The bus master device asserts BSYNC L, indicating it has placed an address on the bus. The transfer is in process until the master negates BSYNC L. In block mode, BSYNC L remains asserted until the last transfer cycle is completed.
AK2	BWTBT L	Write/Byte - BWTBT L is used in two ways to control a bus cycle: 1. It is asserted during the address portion of a cycle to indicate that an output cycle is to follow (DATO, DATOB, DATBO) rather than an input cycle. 2. It is asserted during the Data portion of a DATOB or DATIOB bus cycle to indicate a byte rather than a work transfer is to take place.
AL2	BIRQ4 L	Interrupt request priority level 4.

Table 2-2 (cont.)

Bus Pin	Signal Mnemonic	Signal Function
AM2 AN2	BIAKI BIAKO	Interrupt Acknowledge - The processor asserts BIAKO L to acknowledge an interrupt. The bus transmits this to BIAKI L of the next priority device (electrically closest to the processor). This device accepts the interrupt acknowledge under two conditions: 1. The device requested the bus by asserting an interrupt, BIRQ(x) L. 2. The device had the highest priority interrupt request on the bus at the time of the preceding BDIN L assertion. If both of these conditions are not met, the device asserts BIAKO L to the next device on the bus. This process continues in a daisy-chain fashion until the device with the highest interrupt priority receives the Interrupt Acknowledge (IAK) signal and proceeds with Interrupt Protocol.
AP2	BBS7 L	Bank 7 select - When the bus master asserts TADDR, it asserts this signal to reference the I/O page (including that part of the I/O page reserved for nonexistent memory). The I/O page address is placed on BDAL <12:00> L and the address bits from BDAL <13> on up are ignored. During DATBI transfers, the bus master asserts this signal with the first data transfer until the last transfer indicates to the block mode slave that there will be subsequent transfers.

Table 2-2 (cont.)

Bus Pin	Signal Mnemonic	Signal Function
AR2 AS2	BDMGI BDMGO	Direct memory access grant - The bus arbitrator asserts this signal to grant bus mastership to a requesting device according to bus mastership protocol. The signal is passed in a daisy-chain from the arbitrator (as BDMGO L) along the bus to BDMGI L of the next priority device. This device accepts the grant only if it requested the bus (by assertion of BDMR L). If not, the device passes the grant (asserts BDMGO L) to the next device on the bus. This process continues until the requesting device acknowledges the grant by asserting BSACK L after BRPLY L and BSYNC L are both negated.
AT2	BINIT L	Initialize - This signal provides a system reset. All devices on the bus are to return to a known, initial state--i.e., registers are reset to zero, all bus drivers are disabled, and logic is reset to state Ø, ready to be addressed for operations. Exceptions should be completely documented in programming and engineering specifications for the device.
AU2	BDALØ L	Data/Address line ØØ. BDALO L is either the data bit Ø or the address bit Ø. During byte transfers, it specifies the high or low byte during address for DATOB and DATIOB cycles. If it is asserted, the transfer occurs on the high byte.
AV2	BDAL1 L	Data/Address line Ø1.
BA2	+5	+5V DC power.
BB2	-12	-12V DC power (optional, not required for DIGITAL LSI-11 or F11 hardware options).
BC2	GND	power supply return

Table 2-2 (cont.)

Bus Pin	Signal Mnemonic	Signal Function
BD2	+12	+12V. DC power
BE2	BDAL2 L	Data/Address line Ø2.
BF2	BDAL3 L	Data/Address line Ø3.
BH2	BDAL4 L	Data/Address line Ø4.
BJ2	BDAL5 L	Data/Address line Ø5.
BK2	BDAL6 L	Data/Address line Ø6.
BL2	BDAL7 L	Data/Address line Ø7.
BM2	BDAL8 L	Data/Address line Ø8.
BN2	BDAL9 L	Data/Address line Ø9.
BP2	BDAL1Ø L	Data/Address line 1Ø.
BR2	BDAL11 L	Data/Address line 11.
BS2	BDAL12 L	Data/Address line 12.
BT2	BDAL13 L	Data/Address line 13.
BU2	BDAL14 L	Data/Address line 14.
BV2	BDAL15 L	Data/Address line 15.

2.3 QBUS TRANSACTIONS

During the protocol specifications, bus signals are referred to in different ways. The following is a list of the conventions used:

1. The signal name without any prefixes or suffixes is used when timing, polarity and physical location are unimportant. For example:

   - SYNC, WTBT, BS7, DAL<21:ØØ> or the DAL lines.

2. Most signals on the backplane etch are asserted low and referred to with a prefix character B and a suffix L. For example:

   - BSYNC L, BWTBT L, BBS7 L, BDAL<21:ØØ> L.
   - BPOK H and BDCOK H are asserted high.

3. Receivers and drivers are considered part of the bus. Signal inputs to drivers are referred to with a prefix character T for transmit. For example:

   - TSYNC, TWTBT, TBS7, TDAL<21:ØØ>

4. Signal outputs of receivers are referred to with the prefix character R for received. For example:

   - RSYNC, RWTBT, RBS7, RDAL<21:ØØ>

The conventions listed in 3 and 4 are used when timing is important. The timing is referenced to a receiver output or a driver input. For example:

> After receipt of the negation of RDIN, the slave negates its TRPLY (Ø nanoseconds minimum, 8ØØØ nanoseconds maximum). It must maintain valid data on its TDAL lines until Ø nanoseconds (minimum) after the negation of RDIN, and must negate its TDAL lines 1ØØ nanoseconds (maximum) after the negation of its TRPLY.

2.3.1 Power-up/down Protocol

The power-up/down sequence begins when power is first applied to the system. The BPOK H, BDCOK H, and BINIT L signals initialize the devices on the bus, and the controlling processor becomes the bus master. The processor may then execute some boot program or other power-up option.

Power-up/down protocol is controlled by BDCOK H and BPOK H. These signals are driven by an external device (usually the power supply) and have the following definitions:

- BDCOK H is asserted 3 milliseconds after DC power is within specification, and remains asserted until power fails.

- BPOK H is asserted 70 milliseconds after the assertion of BDCOK H. BPOK H indicates power is stable and at least seven milliseconds of reserve DC power is available should a power failure occur.

- The negation of this line indicates that power is failing and that only four milliseconds of reserve DC power remains. The negation of this line during processor operation starts a power-fail trap sequence.

2.3.1.1 Power-up - Figure 2-1 shows the timing of a power-up sequence. During a power-up sequence the following events occur:

- Initially, there is no DC power, so all the signals are low.

- The power-up logic (located in the power supply) holds BDCOK H negated until DC power is present for three milliseconds (minimum).

- After the processor has nominal power (BDCOK H), BINIT L is asserted for zero nanoseconds (minimum) and then negated.

- The power-up logic (located in the power supply) keeps BPOK H negated until BDCOK H is asserted for 70 milliseconds (minimum). If BDCOK H does not remain asserted (power stable) for 70 milliseconds, BPOK H is not asserted. Critical operations of devices should not be attempted until both BDCOK H and BPOK H are asserted.

- Once asserted, BPOK H must remain so for a minimum of three milliseconds. BDCOK H must remain asserted for four milliseconds after BPOK H is negated.

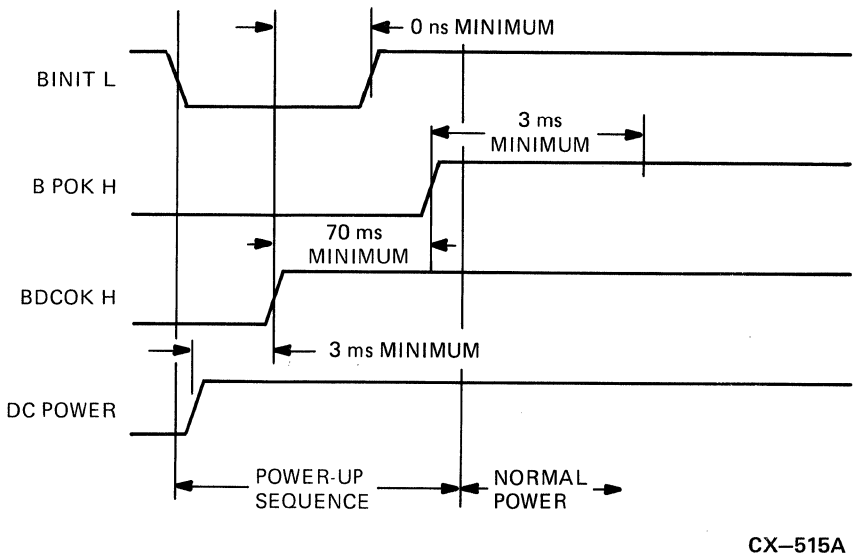


Figure 2-1 Power-up Timing

2.3.1.2 Power-down - Figure 2-2 shows the timing of a power-down sequence. During a power-down sequence the following events occur:

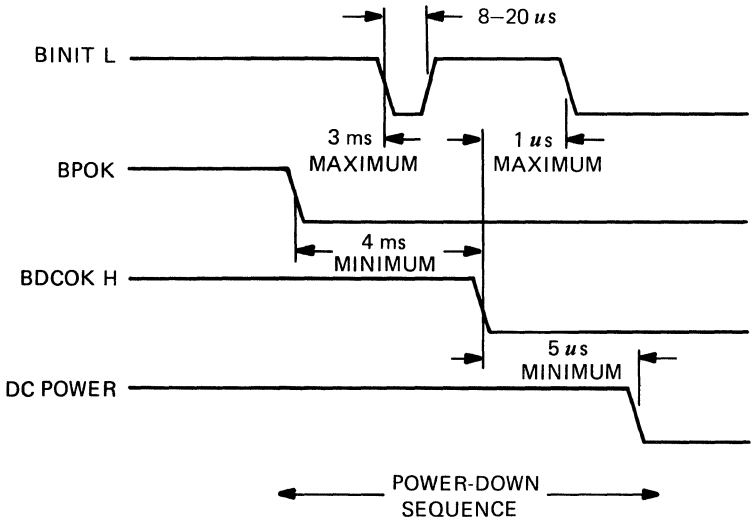
- BPOK H is negated if the ac voltage drops below 75 percent of the nominal value for one full line cycle (15 to 24 milliseconds). Once BPOK H is negated, the entire power-down sequence must be completed.

  If a device requests bus mastership before the power failure and has not become bus master, it should maintain the request until BINIT L is asserted or until the request is acknowledged. If the request is acknowledged, regular bus protocol is followed.

- Processor software should execute a RESET instruction three milliseconds (minimum) after the negation of BPOK H. This asserts BINIT L for eight to twenty microseconds. Processor software executes a halt instruction immediately following the reset instruction.

- BDCOK H must remain asserted for a minimum of four milliseconds after the negation of BPOK H. This four milliseconds allows mass storage and similar devices to protect themselves against erasures and false Writes during a power failure.

- The processor asserts BINIT L one microsecond (minimum) after the negation of BDCOK H.
- The DC power must be stable for a minimum of five microseconds after the negation of BDCOK H.
- BDCOK H must stay negated for a minimum of three milliseconds.



NOTE:
ONCE A POWER-DOWN SEQUENCE IS STARTED, IT MUST BE COMPLETED BEFORE A POWER-UP SEQUENCE IS STARTED.

CX–516A

Figure 2-2 Power-Down Timing

2.3.2 Interrupt Protocol

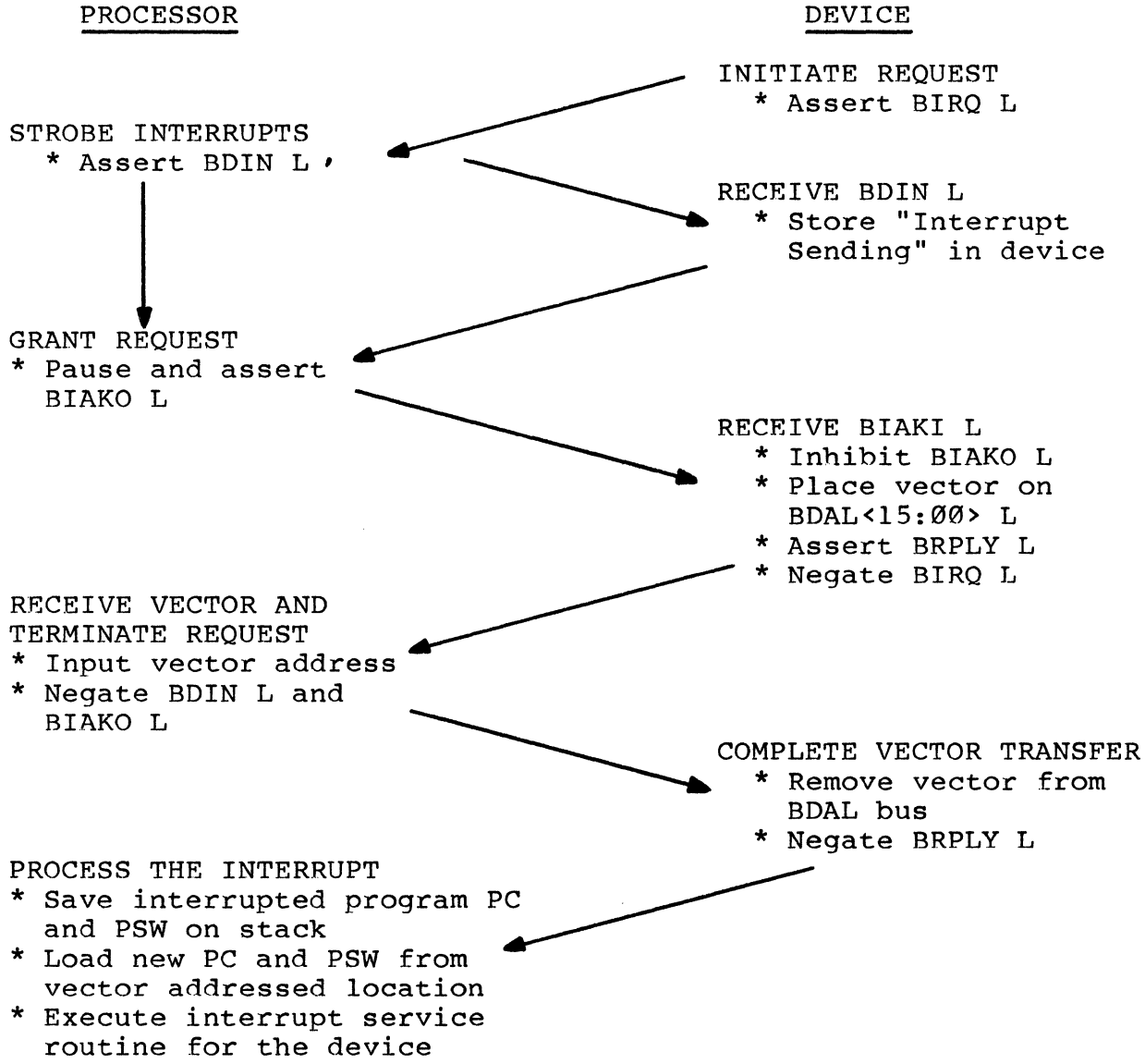
A device interrupts the processor for service because it is done, ready, has an error, or because of some other condition that requires processor intervention. When the processor acknowledges the interrupt, the slave device supplies an interrupt vector address that the processor uses to create a new processor status word (PSW) and a new program counter value (PC). The old PSW and PC are stored on the stack while the processor services the interrupt.

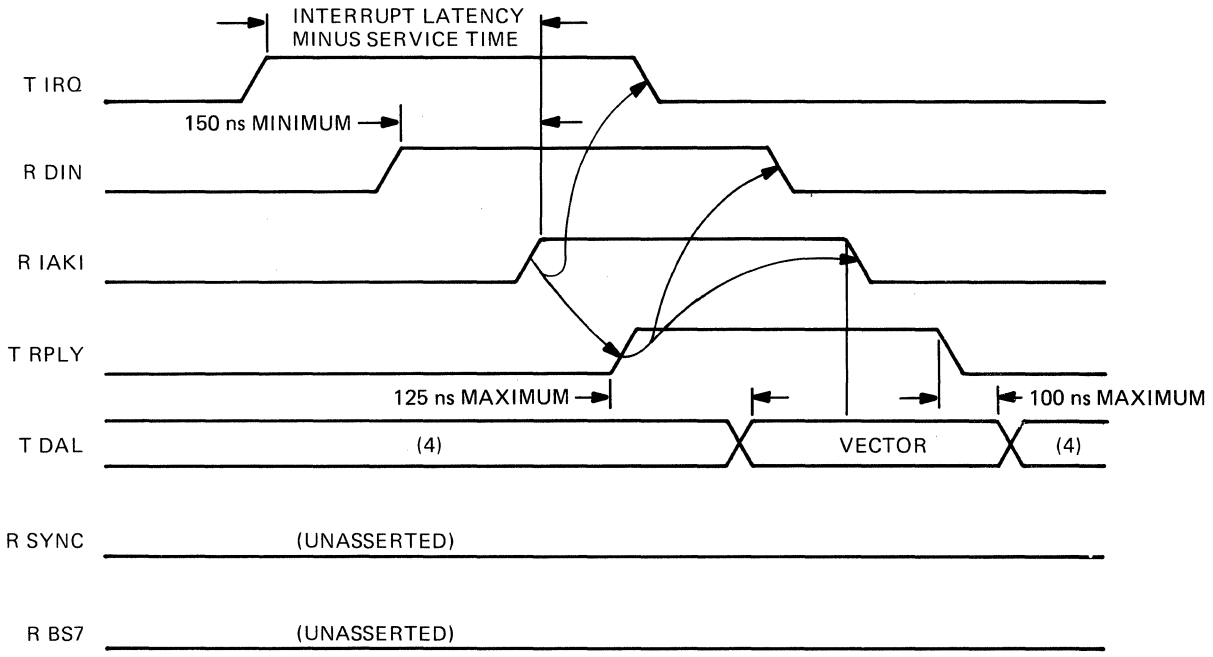
The new PSW contains information on the current status of the processor. This information includes the current processor priority, the condition codes describing the result of the last operation, and a trap bit for program debugging. Bits <Ø7:Ø5> of the new PSW can be programmed for interrupt priority levels Ø through 7. Only interrupts on a higher level than the PSW interrupt priority field are acknowledged.

The new PC value directs the processor to the starting location of the interrupt service routine. The processor executes the routine found at the new PC location and, upon completion, gets the original PC and PSW values from the stack and returns to the routine that was interrupted. See the appropriate processor handbooks for more information on the PC and PSW.

Interrupt protocol has three phases: the interrupt request phase, the interrupt acknowledge and priority arbitration phase, and the interrupt vector transfer phase. The operations performed by the processor and interrupting device are shown below. Interrupt protocol timing is shown in Figure 2-3.

Interrupt Request/Acknowledge Sequence





NOTES:

1. TIMING SHOWN AT REQUESTING DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

CX–517A

Figure 2-3 Interrupt Protocol Timing

2.3.2.1 Interrupt Request - The interrupt request phase starts when a device asserts an interrupt request via the interrupt request lines. BIRQ4 L is the lowest hardware priority level and must be asserted during all interrupts for compatibility with older systems. Table 2-3 shows the signal line combinations and the corresponding interrupt request levels.

Table 2-3 Interrupt Levels

Interrupt Level	Lines Asserted by Device
4	BIRQ4 L
5	BIRQ4 L, BIRQ5 L
6	BIRQ4 L, BIRQ6 L
7	BIRQ4 L, BIRQ6 L, BIRQ7 L

2.3.2.2 Interrupt Acknowledge - Devices capable of requesting interrupts decide whether or not to accept the RIAKI signal on the leading edge of RDIN. There are two cases to consider:

1. If the device does not support position-independent multilevel interrupts, it will accept RIAKI if:

   - It is requesting an interrupt, and does not monitor other interrupts (position dependent priority).

2. If the device does support position-independent multilevel interrupts, it will accept RIAKI if:

   - It is requesting an interrupt, and there is no other higher priority interrupt request pending when RDIN is asserted.

Figure 2-4 shows the position dependent interrupt priority configuration.

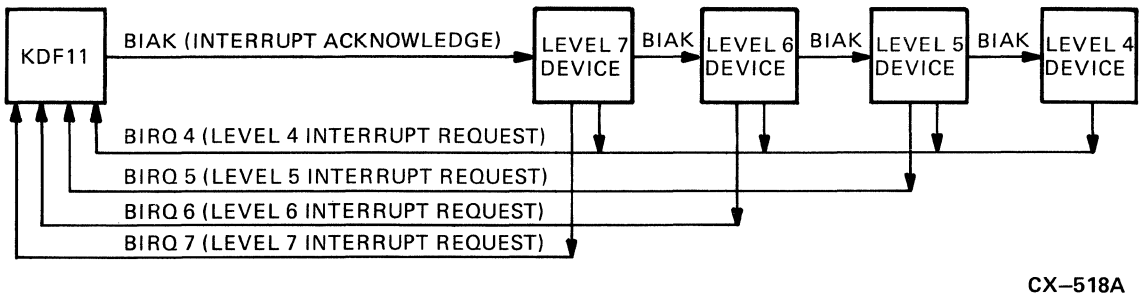


Figure 2-4 Position Dependent Configuration

Devices that support position-independent, multilevel interrupts assert from one to three IRQ lines when requesting an interrupt. These devices also monitor the other interrupt lines to determine if a higher level interrupt is pending. Table 2-4 lists the lines that must be asserted and monitored for each interrupt level. Figure 2-5 shows the position-independent interrupt priority configuration.

Table 2-4 Position-Independent Interrupts

Interrupt Level	IRQ Lines Asserted	IRQ Lines Monitored
4	TIRQ4	RIRQ5, RIRQ6
5	TIRQ4 L, TIRQ5 L	RIRQ6
6	TIRQ4 L, TIRQ6 L	RIRQ7
7	TIRQ4 L, TIRQ6 L, TIRQ7 L	

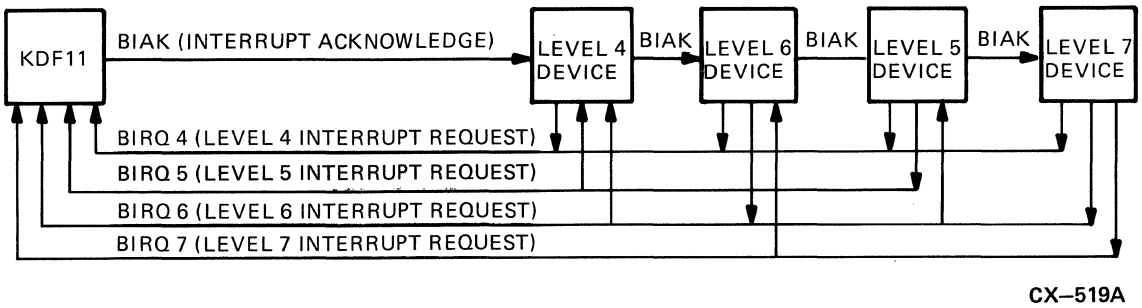


Figure 2-5 Position-Independent Configuration

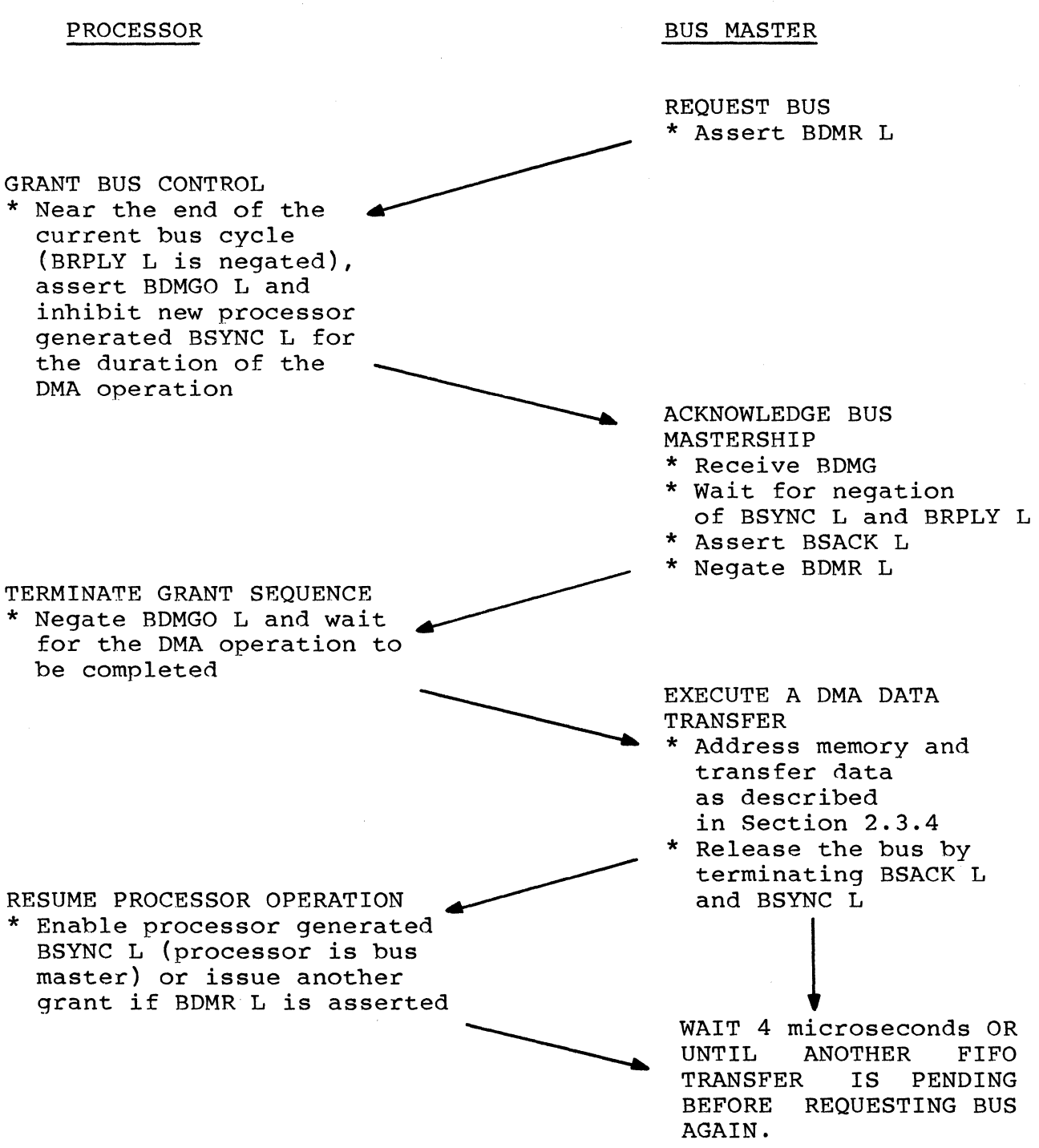
2.3.2.3 Vector Transfer - During the interrupt vector transfer phase, the responding interrupt device receives RIAKI and then asserts TRPLY. The vector address must be stable on TDAL<Ø8:Ø2> 125 nanoseconds (maximum) after TRPLY is asserted. The processor then receives the assertion of RRPLY, and 2ØØ nanoseconds (minimum) later it inputs the vector address and negates both TDIN and TIAKI. The interrupting device negates TRPLY after the negation of RIAKI, and removes the vector address from TDAL<Ø8:Ø2> 1ØØ nanoseconds (maximum)after TRPLY negates. Because vector addresses must be between ØØØ and 774 (octal), the high order TDAL lines remain inactive. The interrupt timing is shown in Figure 2-3.

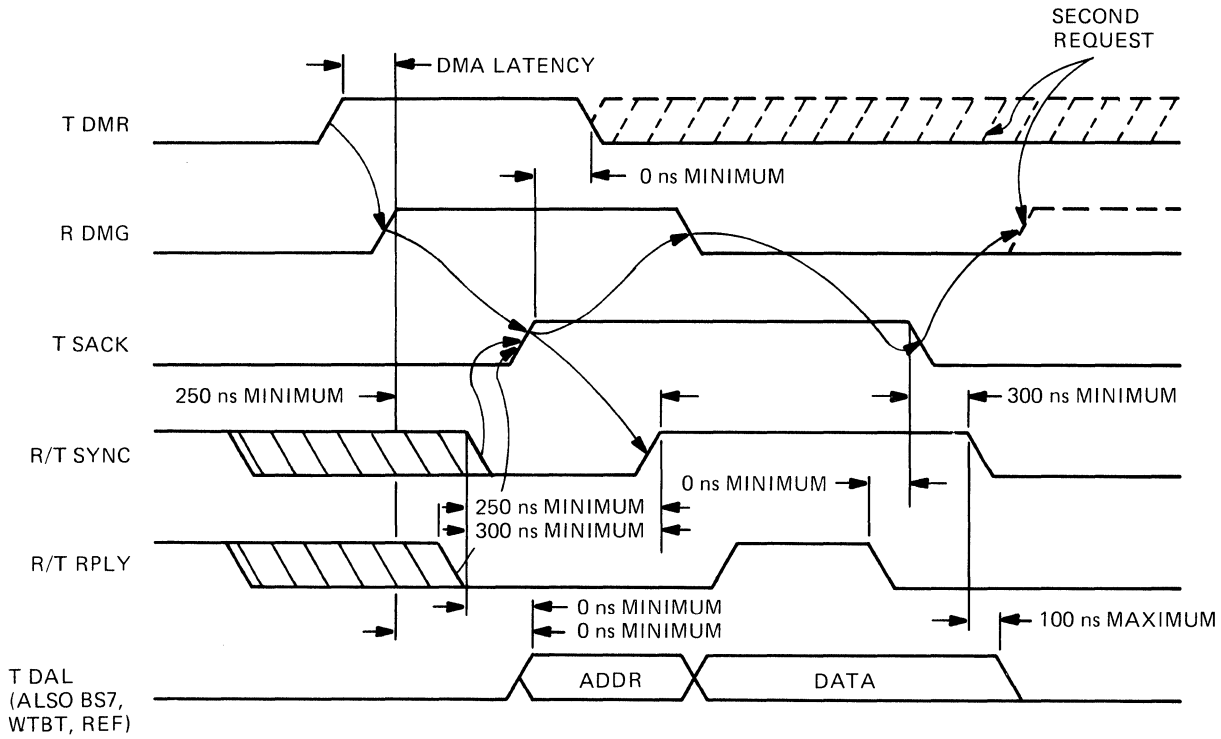
2.3.3 Bus Mastership Protocol

The DMA capability allows direct data transfers between I/O devices and memory, which is useful when using mass storage devices that move large blocks of data. In addition, DMA requests are assigned the highest priority level--that is, any DMA device may request the use of the bus and become bus master by asserting BDMR L. When this is done, the bus arbitration logic evaluates all requests and grants the bus by asserting BDMG L. Since the priority of a DMA device is determined solely by its position on the bus, the DMA device that is closest to the processor and has requested the bus will utilize the grant and remain bus master until it relinquishes its mastership.

A DMA transaction is divided into three phases: the bus mastership acquisition phase, the data transfer phase, and the bus mastership relinquish phase. The protocol for a DMA transaction is shown below, and the bus cycle timing is shown in Figure 2-6.

DMA Request/Grant Sequencing





NOTES:

1. TIMING SHOWN AT REQUESTING DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.

CX-520A

Figure 2-6 DMA Request/Grant Bus Cycle Timing

2.3.3.1 Bus Acquisition Phase - During bus acquisition, a DMA device requests the bus by asserting TDMR. The processor arbitration logic responds with TDMGO. The grant signal passes through all the DMA devices in the DMA daisy chain until it reaches the requesting device. The requesting device blocks the grant and asserts TSACK as a response to it. If the processor does not receive a response to the DMA grant, it clears the grant and rearbitrates the bus.

2.3.3.2 Data Transfer Phase - During the data transfer phase, the DMA device continues asserting BSACK. If this is a multiple word transfer (block mode transfer), the DMA device must limit the block mode transfer to eight words, unless it monitors RDMR to see if other devices need the bus. If the DMA device monitors RDMR, it can transfer up to 16 words.

2.3.3.3 Bus Relinquish Phase - At the end of the last data transfer, the DMA device relinquishes the bus by negating TSACK. TSYNC is negated 300 nanoseconds (maximum) after TSACK is negated.

2.3.4 Data Transfer Protocol

The current bus master initiates a data transfer by placing the slave address and appropriate control signals on the bus. It then waits for a response from the slave before continuing with the transfer. The master-slave relationship allows both fast and slow devices to communicate on the bus because a bus cycle is not complete until the required master-slave signal sequence occurs.

The QBUS has either 18 or 22 DAL lines (<17:00> or <21:00>), depending on the QBUS version examined. However, the control signals are the same on all versions.

The QBUS data word is 16-bits wide and is comprised of the low byte and the high byte. The low byte is bits <07:00>, and the high byte is bits <15:08>. Bits 16 and 17 supply parity error detection. Bit 17 indicates the master device should enable its parity error detection logic. Bit 16 indicates a parity error has occurred. Bits <21:18> are not active during the data portion of a cycle.

QBUS addressing is accomplished using the BDAL lines and the BBS7 signal. If BBS7 is asserted, the bus master is addressing an I/O device, and the device address in the I/O page is found on BDAL lines <12:00>, regardless of the QBUS version. If BBS7 is not asserted, the bus master is addressing memory, and the address is found on DAL<21:00>, DAL<18:00> or DAL<16:00>, depending on the QBUS version.

The I/O page defines the range of addresses available for I/O devices. In QBUS systems, the I/O page is the top 8K bytes of the address space. Table 2-5 lists the I/O page addresses for the 16, 18, and 22-bit QBUS versions.

Table 2-5 I/O Page Addresses

Address Space	I/O Page Addresses
16 bit	160000 through 177777
18 bit	760000 through 777777
22 bit	17760000 through 17777777

QBUS memory address space is different for the different QBUS versions. Table 2-6 lists the memory addresses for the 16, 18, and 22 bit QBUS versions.

Table 2-6 QBUS Memory Address Ranges

Address Space	Memory Address Ranges
16 bit	000000 through 157777
18 bit	000000 through 757777
22 bit	00000000 through 17757777

All systems reserve virtual addresses 000 through 376 for trap and interrupt vector addresses. Some larger systems also include virtual addresses 400 through 776 for their interrupt vectors.

2.3.5 Data Transfers

The QBUS supports seven types of data transfers. These transfers move 16-bit words or 8-bit bytes to or from slave devices. Block mode transfers start with an initial address and data word. Subsequent data words are not transferred with an address. Instead, these words are automatically transferred to the next sequential location in memory. Additional words are stored sequentially in this manner until the transfer is complete. In byte output operations, the data to be written in the destination byte is valid on the appropriate DAL lines.

Although the QBUS supports seven data transfer types, the KDA50-Q performs only four of them. Table 2-7 lists the transfer types the KDA50-Q performs, as well as the other three types.

Table 2-7 Data Transfer Types

Bus Cycle Mnemonic	Description	Function With Respect to Bus Master
DATI	Data word input	Read word
DATO	Data word ouptut	Write word
DATOB	Data byte output	Write byte - The KDA5Ø-Q does not perform the DATOB operation.
DATIO	Data word input/output	Read word, modify, write word - The KDA5Ø-Q does not perform the DATIO operation.
DATIOB	Data word input/byte output	Read word, modify, write byte - The KDA5Ø-Q does not perform the DATIOB operation.
DATBI	Data block input	Block mode read
DATBO	Data block output	Block mode write

2.3.5.1 Address Portion Of The Transfer - The address portion of the transfer is made up of an address set-up/deskew time and an address hold/deskew time. During the address set-up/deskew time the bus master does the following:

- Asserts TDAL<21:ØØ> with the desired slave device address bits.
- Asserts TBS7 if a device in the I/O page is being addressed.
- Asserts TWTBT if the cycle is a data out bus cycle.
- Asserts TSYNC 15Ø nanoseconds (minimum) after gating TDAL, TBS7 and TWTBT onto the bus.

The slave address, RBS7 and RWTBT are asserted at the slave bus receiver for at least 75 nanoseconds before RSYNC becomes active. Devices in the I/O page ignore the 9 high order address bits RDAL<21:13> and instead decode RBS7 along with the 13 low order address bits. An active RWTBT signal indicates a data-out operation follows, while an inactive RWTBT indicates a data-in or DATIO(B) operation.

The address hold/deskew time starts after RSYNC is asserted. The slave device uses the active RSYNC to clock RDAL address bits RBS7 and RWTBT into its internal logic. RDAL<21:00>, RBS7 and RWTBT will stay active for 25 nanoseconds (minimum) after RSYNC becomes active. TSYNC stays active for the duration of the bus cycle.

2.3.5.2 DATI - The DATI bus cycle is a Read operation because a 16-bit data word moves from the slave to the master. During a DATI operation the master:

- Gates the slave address to the bus and negates TWTBT.
- Asserts TSYNC 150 nanoseconds (minimum) after the slave address is gated.
- Asserts TDIN 100 nanoseconds (minimum) after the assertion of TSYNC.

When the slave receives DIN it:

- Asserts TRPLY 0 nanoseconds (minimum) after RDIN. A bus timeout occurs if TRPLY is not asserted within 8000 nanoseconds.
- Gates TDATA onto the bus 125 nanoseconds (maximum) after TRPLY.

In response to the reply signal from the slave, the master:

- Has valid RDATA 200 nanoseconds (maximum) after RRPLY.
- Negates TDIN 200 nanoseconds (minimum) after RRPLY.

After receiving the negation of the data in signal, the slave:

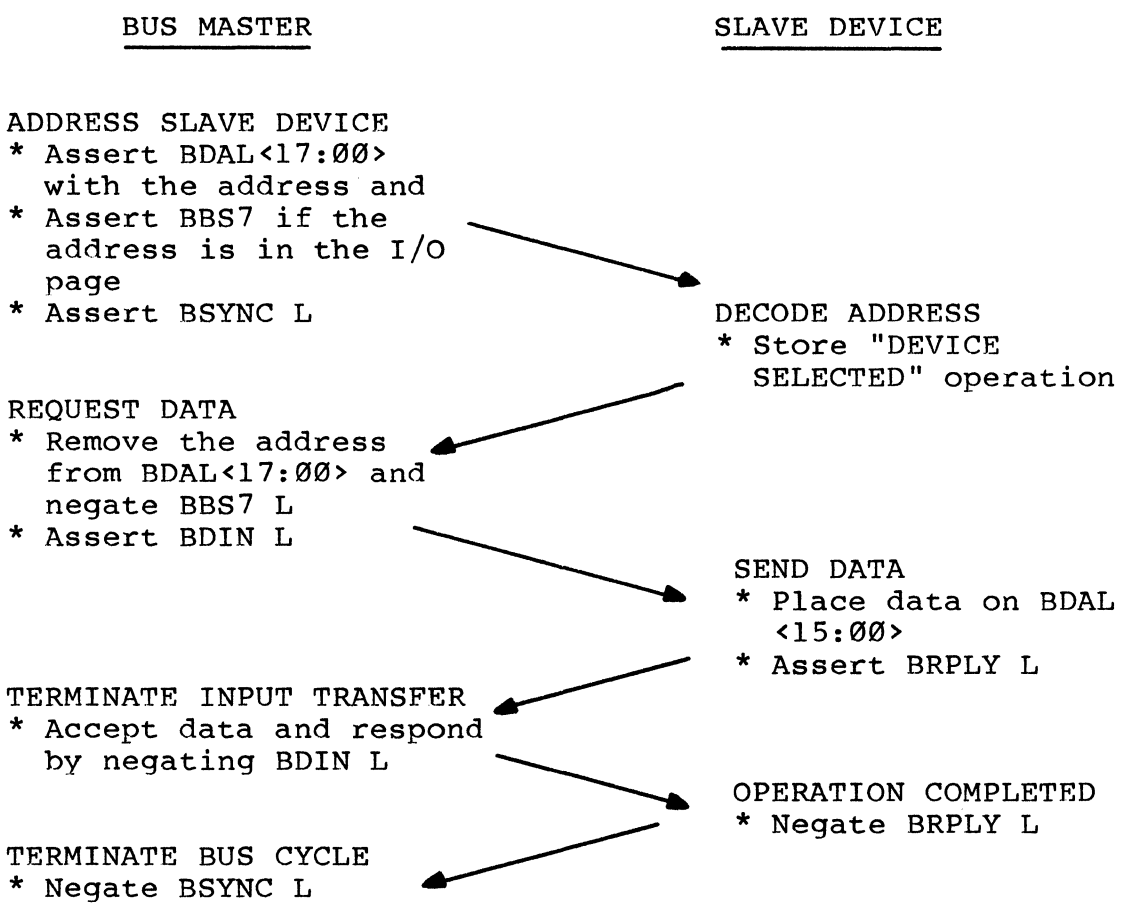
- Negates TRPLY Ø nanoseconds (minimum) after RDIN is negated.
- Negates TDATA Ø nanoseconds (minimum) and 1ØØ nanoseconds (maximum) after it negates TRPLY.

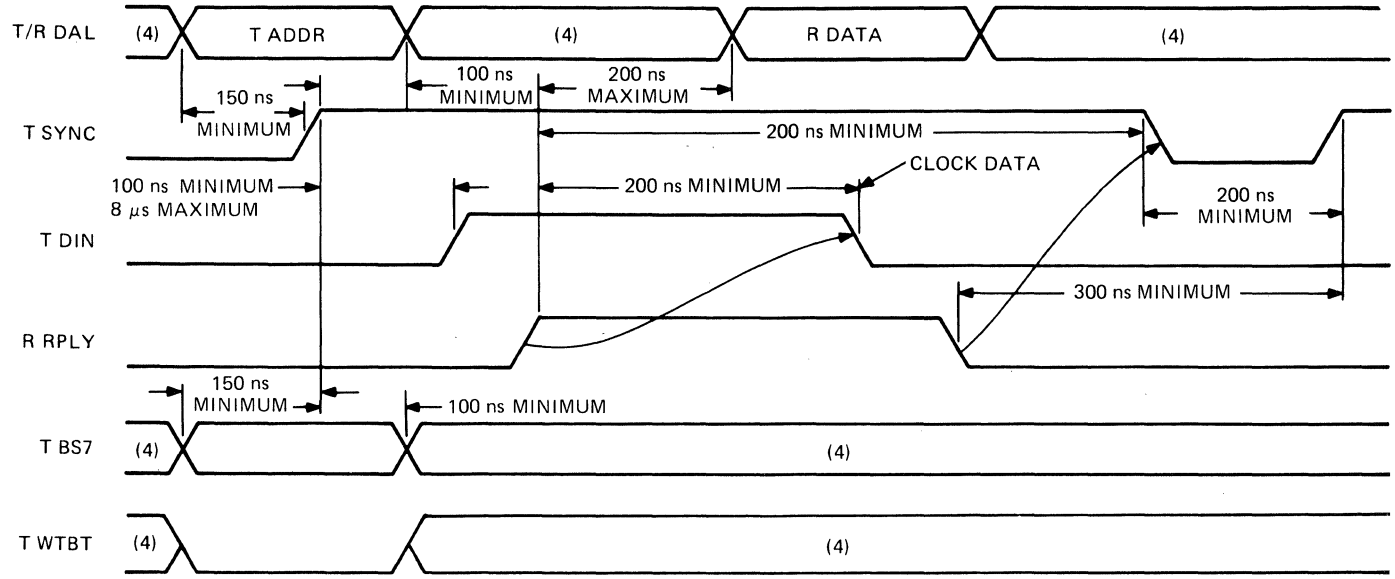
To complete the transfer, the master:

- Negates TSYNC Ø nanoseconds (minimum) after RRPLY is negated.

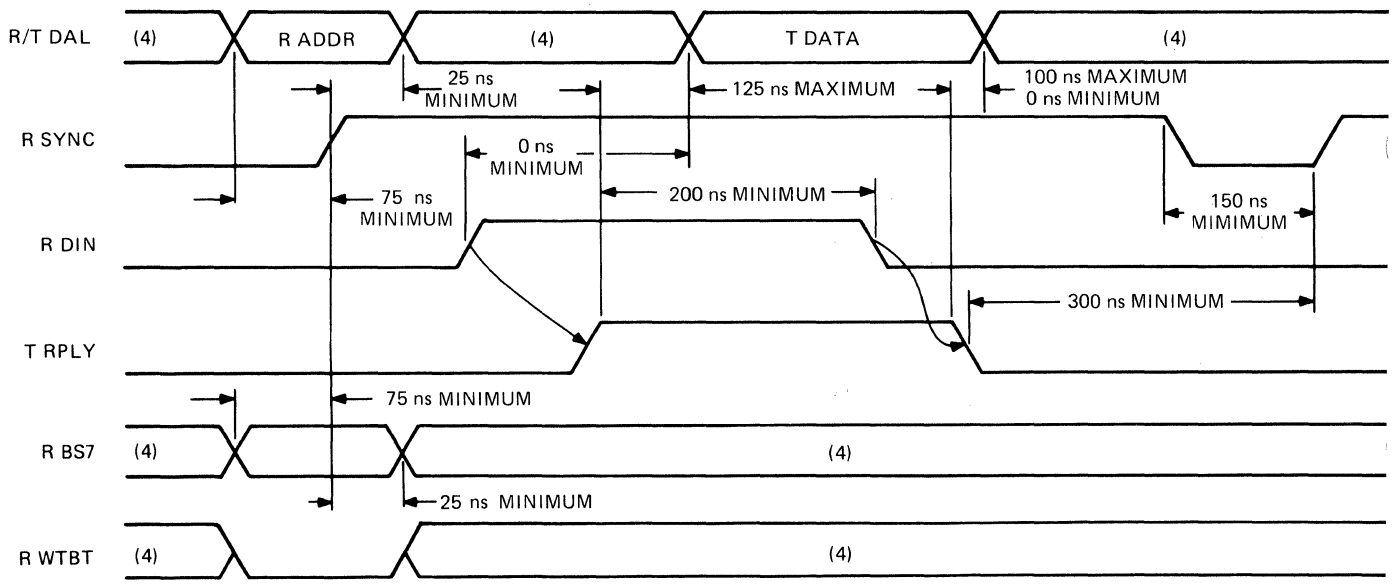
The DATI bus cycle operations are shown below in flow chart format, and the bus cycle timing is shown in Figure 2-7.

DATI Bus Cycle





TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES:

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

CX-521

Figure 2-7 DATI Bus Cycle Timing

2.3.5.3 DATO - The DATO bus cycle is a Read operation because a 16-bit data word moves from the master to the slave. The data output timing occurs after the addressing portion of a DATO bus cycle as shown in Figure 2-8.

The address portion of the DATO operation is described in Section 2.3.5.1. The data transfer portion of a DATO is made up of a data set-up/deskew time and a data hold/deskew time.

NOTE

The KDA5Ø does not perform DATOB operations.

During the data set-up/deskew time, the master:

- Places the data on TDAL <15:ØØ> 1ØØ nanoseconds (minimum) after asserting TSYNC, and TDAL <17:16> supplies parity error detection. If memory supports parity error detection and TDAL<16> L is asserted during data transfer time, a parity error is written into memory. TDAL <17> is not active during a Write.
- Negates TWTBT 1ØØ nanoseconds (minimum) after TSYNC is asserted.
- Asserts TDOUT 1ØØ nanoseconds (minimum) and 8 microseconds (maximum) after the TDAL and TWTBT lines are stable.

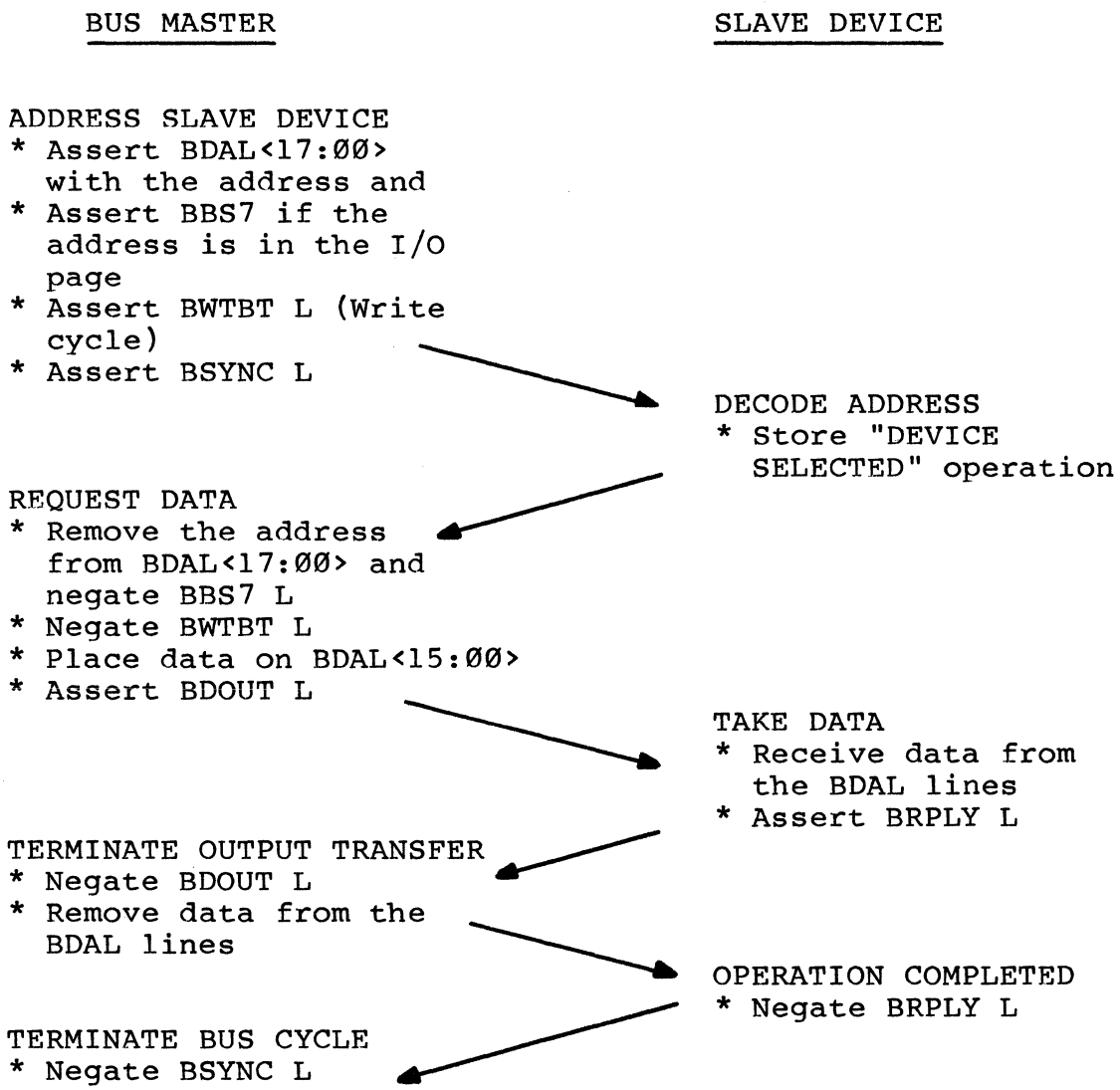
The slave responds to TDOUT by accepting the data and asserting TRPLY. This completes the data set-up/deskew time.

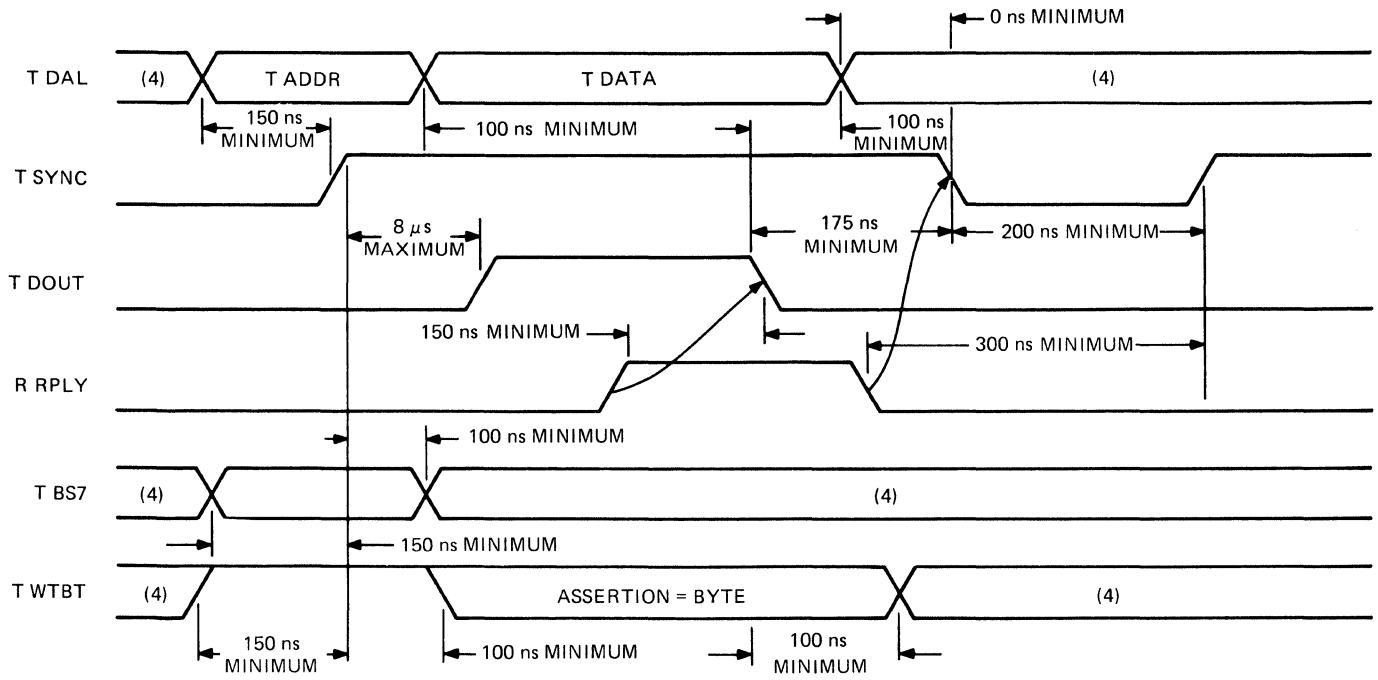
During the data hold/deskew time the following operations occur:

- The master receives RRPLY and negates TDOUT 15Ø nanoseconds (minimum) later.
- The master keeps the TDAL lines stable for at least 1ØØ nanoseconds after negating TDOUT.
- The slave senses the negation of RDOUT and negates TRPLY.
- The master negates TSYNC after RRPLY is negated.

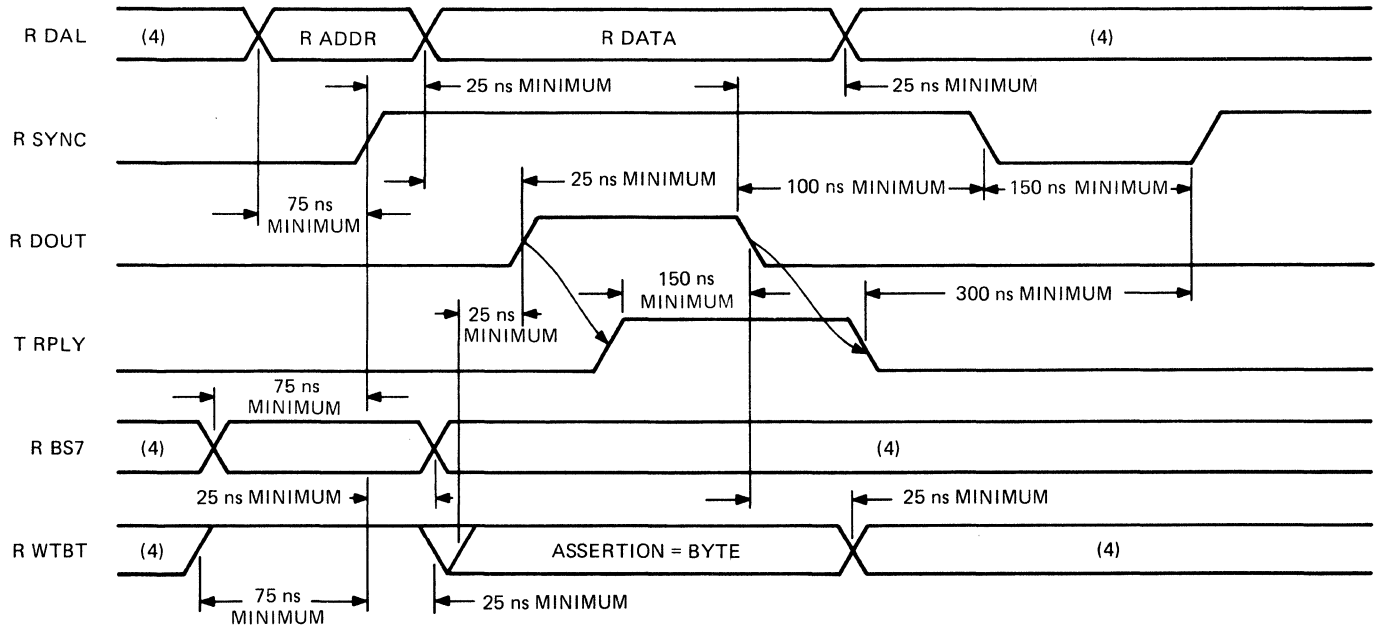
The DATO bus cycle is now complete.

DATO Bus Cycle





TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES:

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

CX–522A

Figure 2-8 DATO Bus Cycle Timing

2.3.5.4 Block Mode Transfers - Block mode transfers move multiple words of data in the same bus cycle. Eight data words are transferred to sequential locations by starting with an initial address and automatically incrementing it each time a word is transferred. The initial address is specified at the beginning of the transfer and is followed by the multiple data words.

If the master device monitors BDMR and it has not been asserted by the seventh word in the transfer, the transfer may continue for another block of eight 16-bit words. The bus master may continue to transfer up to 16 blocks of data as long as BDMR is not asserted by another device.

2.3.5.4.1 DATBI - The DATBI bus cycle is a block mode Read operation because blocks of data are moved from the slave to the master device.

During a DATBI operation, the master:

- Gates the slave address to the bus and negates TWTBT.
- Asserts TSYNC 150 nanoseconds (minimum) after the slave address is gated.
- (*) Asserts TDIN 100 nanoseconds (minimum) after the assertion of TSYNC.
- Asserts TBS7 within 50 nanoseconds of TDIN assertion.

When the slave receives RDIN, it:

- Asserts TRPLY 0 nanoseconds (minimum) after RDIN. (A bus timeout occurs if TRPLY is not asserted within 8000 nanoseconds.)
- Asserts TREF with TRPLY. This indicates the slave can accept another block mode DIN transfer.
- Gates TDATA onto the bus 125 nanoseconds (maximum) after TRPLY.

In response to the reply signal from the slave, The master:

- Has valid RDATA 200 nanoseconds (maximum) after RRPLY.
- Negates TDIN 200 nanoseconds (minimum) after RRPLY.

The slave responds to the negation of RDIN by:

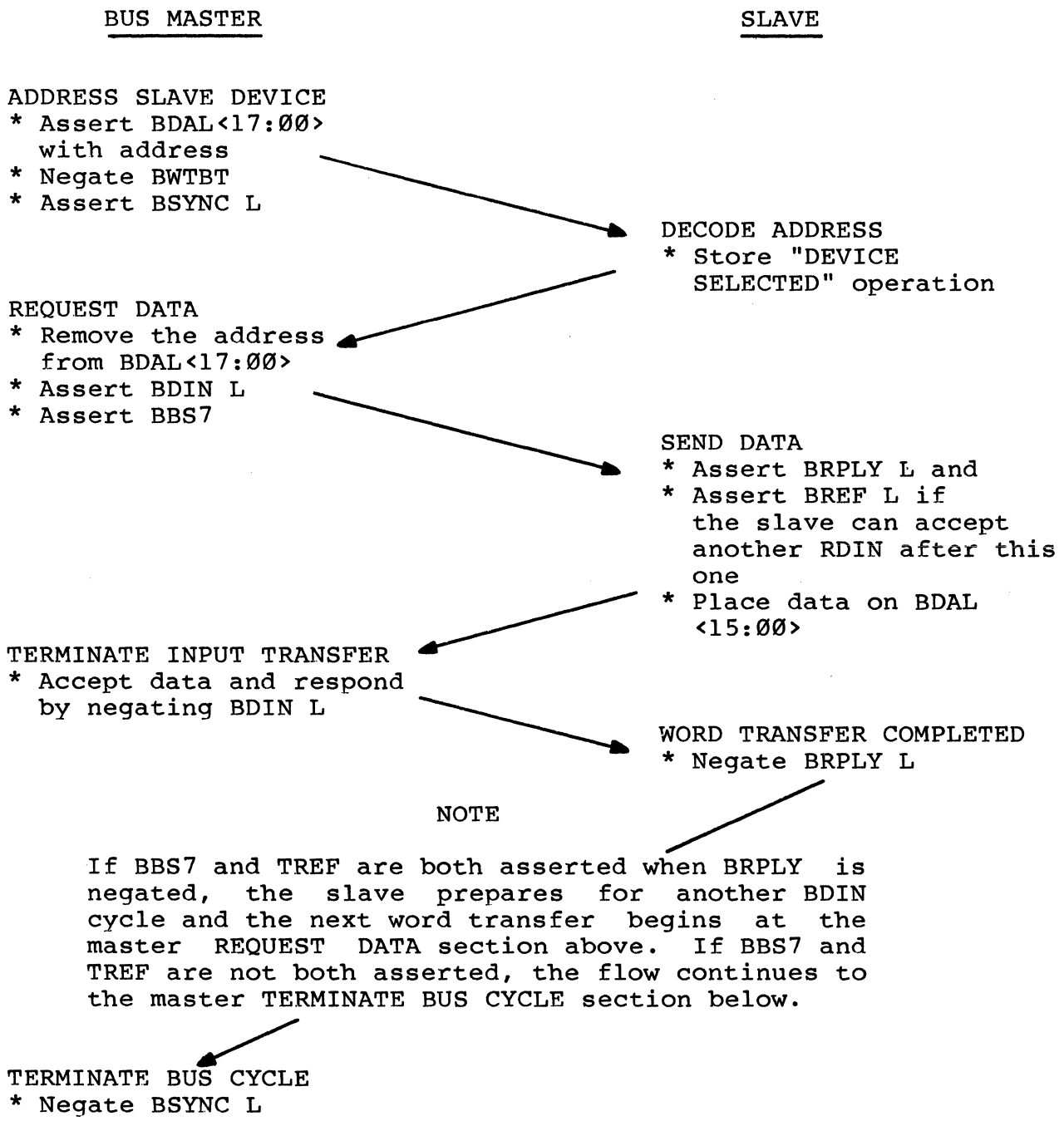
- Negating TRPLY Ø nanoseconds (minimum) after RDIN is negated.
- Preparing for another RDIN cycle if both RBS7 and TREF are still asserted, or removing TDATA from the bus 1ØØ nanoseconds (maximum) after negating TRPLY.

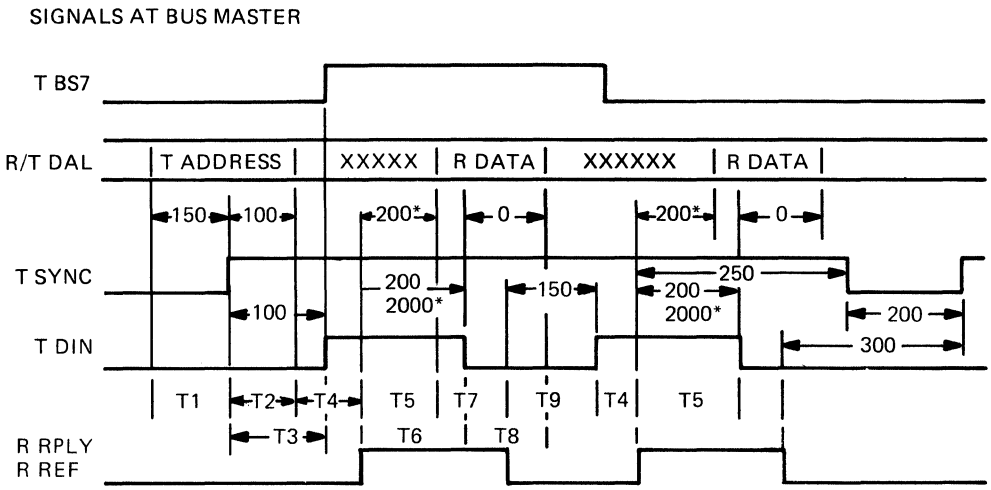
If the transfer is completed (BBS7 and BREF are not both asserted), the master responds by negating TSYNC Ø nanoseconds (minimum) after RRPLY is negated and the bus cycle is completed.

If the transfer is not complete (BBS7 and BREF are asserted), the master asserts TDIN 15Ø nanoseconds (minimum) after RRPLY is negated and the protocol continues with (*) above.

DATBI bus cycle operations are shown below in a flow chart format, and the bus cycle timing is shown in Figure 2-9.

DATBI Bus Cycle





T1	=	ADDRESS TO T SYNC	150 NANOSECONDS MINIMUM
T2	=	ADDRESS HOLD	100 NANOSECONDS MINIMUM
T3	=	T SYNC TO T DIN	100 NANOSECONDS MINIMUM
T4	=	T DIN TO R RPLY T (DRIVE) + T (PROP) + T (RECEIVE) + T (DELAY) + T (DRIVE) + T (PROP) + T (RECEIVE)	
T5	=	R RPLY TO DATA	200 NANOSECONDS MAXIMUM
T6	=	R RPLY TO T DIN	200 NANOSECONDS MINIMUM
T7	= = =	T DIN TO R RPLY T (DRIVE) + (PROP) + T (RECEIVE) + T (DELAY) + T (DRIVE) + T (PROP) + T (RECEIVE)	
T8	=	R RPLY TO DATA	0 NANOSECONDS MINIMUM
T9	=	R RPLY TO T DIN	150 NANOSECONDS MINIMUM
T CELL	=	T4 + T6 + T7 +T9	–SINCE T6 MUST BE> T5 FOR MASTER TO HAVE VALID DATA - AND T9>T8

NOTE:
TIMES ARE MINIMUM EXCEPT WHERE "*" DENOTES MAXIMUM

CX–523A

Figure 2-9 DATBI Bus Cycle Timing

2.3.5.4.2 DATBO - The DATBO bus cycle is a block mode Write operation because blocks of data are moved from the master to the slave device.

During a DATBO operation the master:

- Asserts the slave address on the TDAL lines, and also asserts TWTBT to indicate an output operation follows.
- Asserts TSYNC.
- Negates TWTBT 1ØØ nanoseconds (minimum) after TSYNC (because this is a word transfer) and gates TDATA onto the bus.
- Asserts TDOUT 1ØØ nanoseconds (minimum) after TDATA is gated onto the bus.

(*) In response, the slave:

- Accepts the data.
- Asserts TREF with TRPLY. This indicates the slave can accept another block mode DOUT transfer.

In response to the TRPLY signal from the slave, the master:

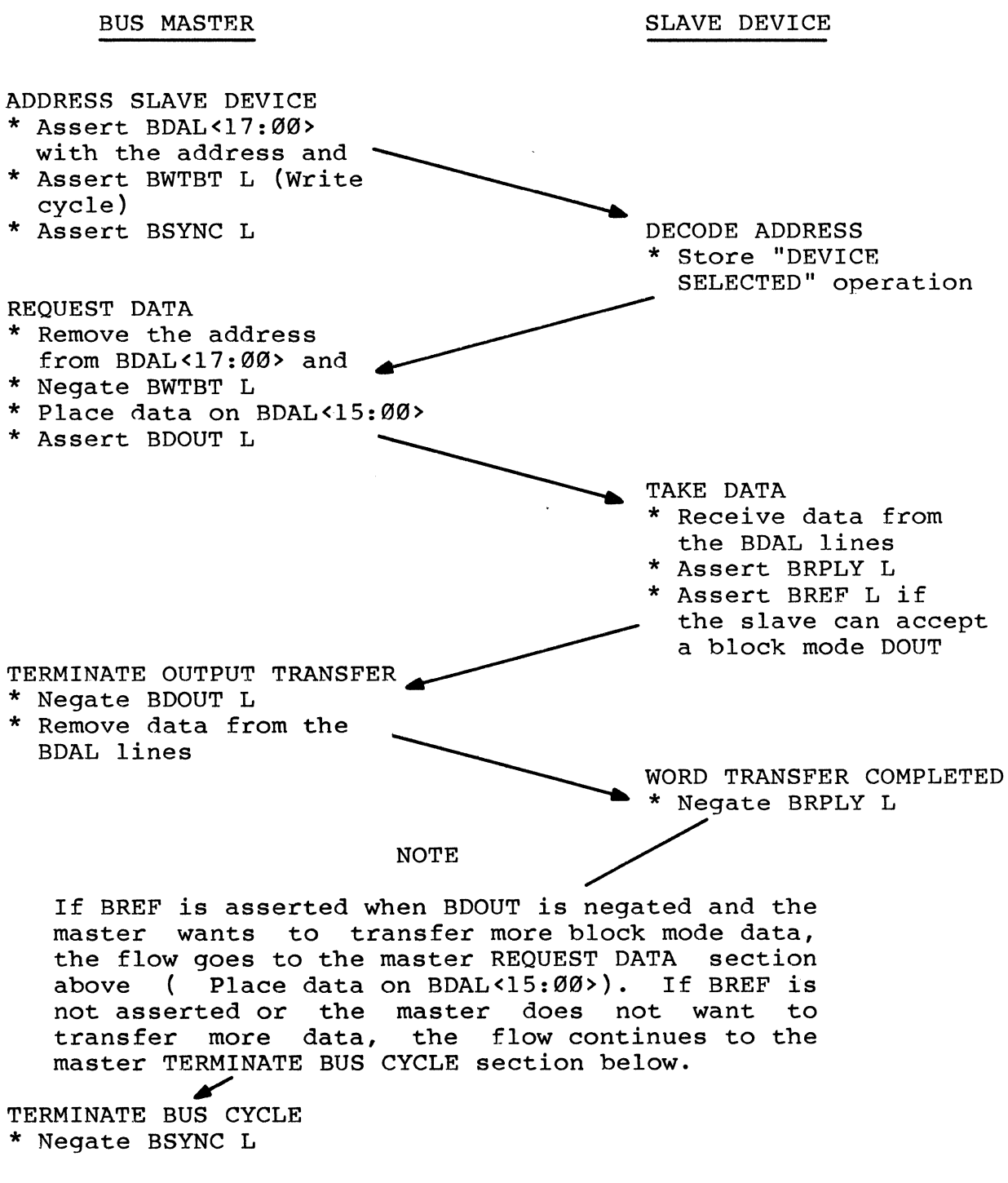
- Negates TDOUT 15Ø nanoseconds (minimum) after RRPLY is asserted.
- (**) Monitors the RREF line. If RREF is still asserted after TDOUT is negated, the master may transfer more block mode data to the slave. In this case, the master gates new TDATA onto the bus 1ØØ nanoseconds (minimum) after negating TDOUT.

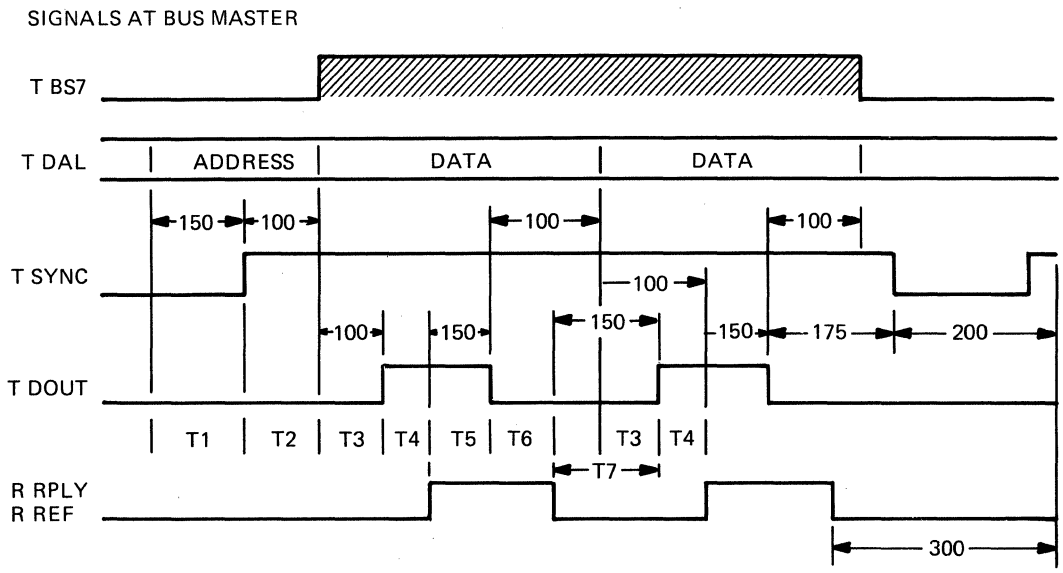
The slave negates TRPLY Ø nanoseconds (minimum) after TDOUT is negated. The master then gates new TDATA onto the bus and asserts TDOUT 1ØØ nanoseconds (minimum) later. The timing continues with (*) above.

If the master did not want to continue the transfer, or the RREF signal was not present in (**) above, the master would have negated TSYNC and the DATBO bus cycle would complete.

The DATBO bus cycle is illustrated in flow chart form below and the timing is shown in Figure 2-1Ø.

DATBO Bus Cycle





T1	=	ADDRESS TO T SYNC	
T1	=	ADDRESS TO T SYNC	150 NANOSECONDS MINIMUM
T2	=	ADDRESS HOLD	100 NANOSECONDS MINIMUM
T3	=	DATA TO T DOUT	100 NANOSECONDS MINIMUM
T4	=	T DOUT TO R RPLY	
	=	T (DRIVE) + T (PROP) + T (RECEIVE) + T (DELAY) + T (DRIVE) + T (PROP) + T (RECEIVE)	
T5	=	R RPLY TO T DOUT	150 NANOSECONDS MINIMUM
T6	=	T DOUT TO R RPLY	
	=	T (DRIVE) + T (PROP) + T (RECEIVE) + T (DELAY) + T (DRIVE) + T (PROP) + T (RECEIVE)	
T7	=	R RPLY TO T DOUT	150 NANOSECONDS MINIMUM
T CELL	=	T3 + T4 + T5 + T6 + T7	–SINCE T3 < T7

NOTE:
TIMES ARE MINIMUM EXCEPT WHERE "*" DENOTES MAXIMUM

CX–524A

Figure 2-1Ø DATBO Bus Cycle Timing

CHAPTER 3
KDA5Ø-Q FUNCTIONAL BLOCK DIAGRAM

3.1 INTRODUCTION

The functional block diagram for the KDA5Ø-Q is divided into the processor and SDI sections. The processor section, found on KDA5Ø-Q processor module M7164, contains the following circuitry:

- QBUS Interface
- DPROC and UPROC sequencers
- Shared ALU
- Pipeline Register
- Control Store (CROM)
- Clocks and control circuitry
- QBUS Control Register (LEDs)

The SDI section, found on KDA5Ø-Q SDI module M7165, contains the following circuitry:

- SERDES 1Ø and 16
- Error Correction Code (ECC) Generator
- Serial-to-parallel RTDS conversion
- Parallel-to-serial RTCS conversion

- 32K byte data buffer
- High and Low byte parity checking/generation
- Electrical interface to the SDI
- Drive Control Register (LEDs)

3.2 PROCESSOR BLOCK DIAGRAM

Figure 3-1 is a block diagram of the KDA5Ø-Q processor module (M7164). The 16-bit Bidirectional Data Bus is common to the processor and SDI modules. The following paragraphs provide a functional description of the blocks shown.

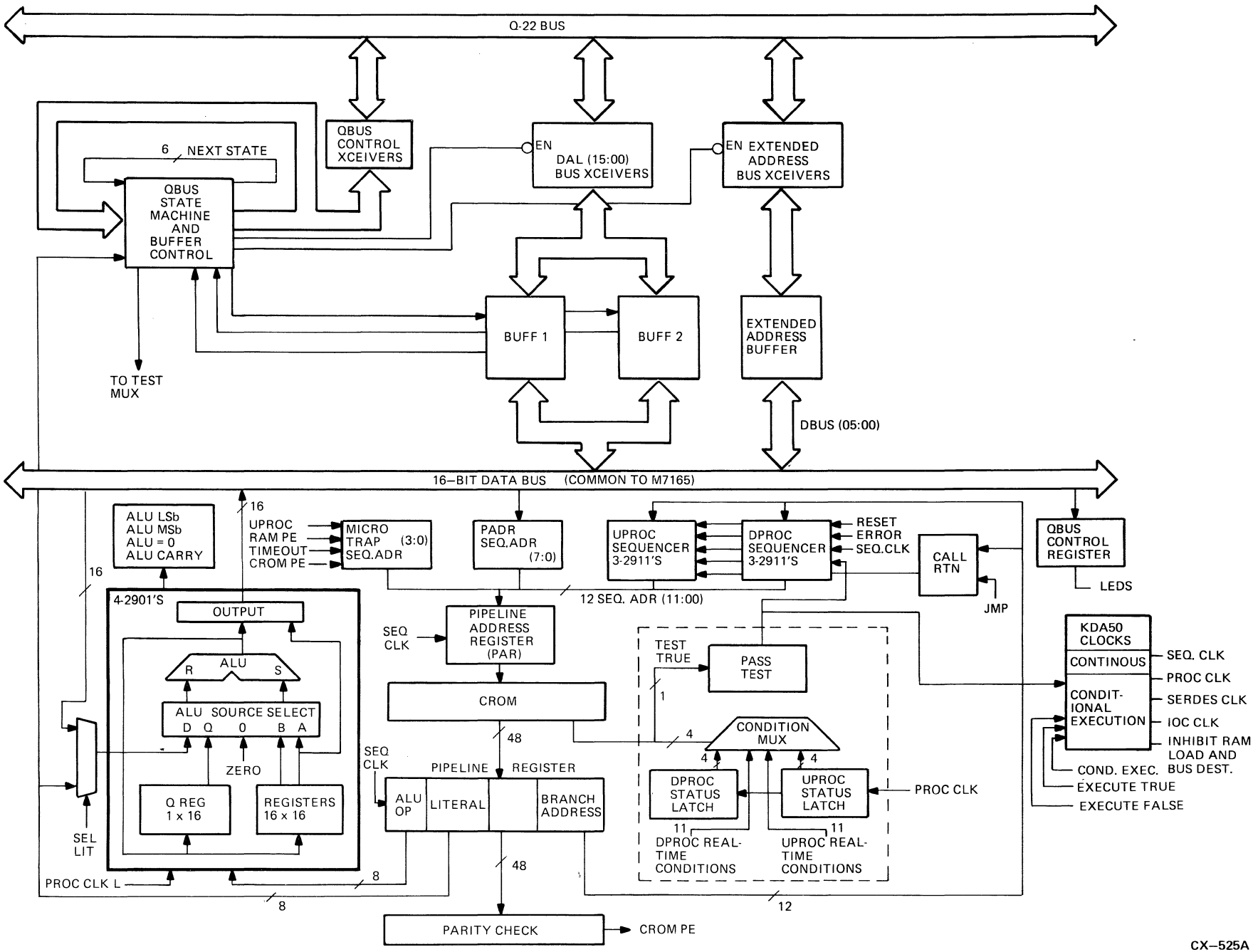


Figure 3-1 M7164 Block Diagram

3.2.1 Bus Transceivers

The Bus Transceivers transmit and receive data between the QBUS and the KDA5Ø-Q internal DAL bus and control lines. When transmitting, they provide sufficient output current to charge and discharge the bus capacitances quickly. When receiving, they respond to high speed bus transitions and provide low propagation delays.

3.2.2 Address/Data Buffers

The processor module has two 16-bit address/data buffers (BUFF 1 and BUFF 2) plus a separate Extended Address Register for communication between the QBUS transceivers and the internal KDA5Ø-Q data bus. Buff 1 and Buff 2 move information between the 16-bit DBUS and DAL lines <15:ØØ>, which may contain data or address information depending on the operation being performed. The Extended Address Register allows the 16-bit DBUS to transmit 21-bit-wide addresses on the QBUS. The register is placed between DBUS <Ø5:ØØ> and DAL <21:16>. During an addressing operation, the Extended Address Register and Buff 1 or Buff 2 are loaded as separate and distinct registers.

The address/data buffers contain two internal registers used during data movement. Figure 3-2 shows the use of the internal registers during the different types of data movement.

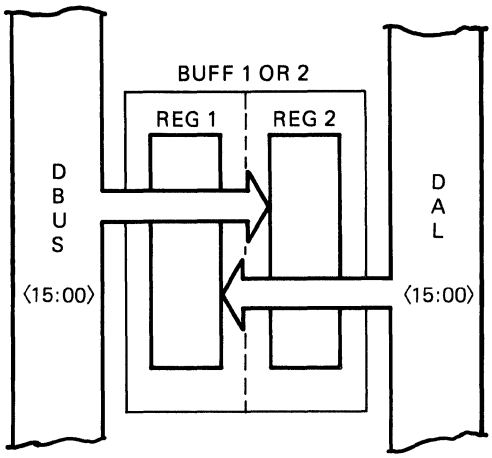


Figure 3-2 Buffer Operation

NOTE

The names REG 1 and REG 2 are used for explanation purposes only.

When storing data from an input bus to the internal registers, the buffers move data from the DAL lines to REG 1 or from the DBUS to REG 2. When transferring the stored data from the internal registers to the output bus, the buffers move data from REG 1 to the DBUS or from REG 2 to the DAL lines. A bus to bus transfer requires that data first be moved from the input bus to an internal register, and then from the internal register to the output bus.

3.2.3 Buffer Control Logic

Control and status portions make up the buffer control logic. The control portion performs the following operations:

- Enables and disables the buffers for both the state machine and the UPROC.
- Generates a clock to transfer data from the KDA5Ø-Q internal data bus into one of the buffers.

The status portion performs the following operations:

- Reports the status of BUFF 1 and BUFF 2.
- Reports the availability of both buffers to the control portion. Availability is defined differently for a read from the buffer register than for a write to the buffer register. When reading, the buffer is considered available when it is full. When writing, the buffer is considered available when it is empty.
- Supplies the clock to transfer data from the QBUS to the data buffers.

As an example, the following describes the operation of BUFF 1, BUFF 2, and the buffer control logic during a one word DATO operation. In this example, the KDA5Ø-Q is bus master and is transferring data to a slave device on the QBUS. Initial conditions are:

- The buffer control logic is in DATO mode.
- The UPROC has buffers available (empty) to accept data from the KDA5Ø-Q internal bus.
- The State Machine does not have any available (full) buffers to transfer to the QBUS.

- Neither BUFF 1 or BUFF 2 is currently active.

The following steps complete a one word DATO transfer:

1. Using the control portion of the buffer control logic, the UPROC clocks address information into BUFF 1. This results in the following status conditions:

   - BUFF 1 active.

   - Buffer Available for the State Machine and for the UPROC.

2. Using the control portion of the buffer control logic, the UPROC clocks data information into BUFF 2. This results in the following status conditions:

   - BUFF 1 and BUFF 2 active.

   - Buffer available for the State Machine, but not for the UPROC.

3. Using the control portion of the buffer control logic, the State Machine enables the address information onto the QBUS via BUFF 1. This results in the following status conditions:

   - BUFF 2 active; BUFF 1 not active.

   - Buffer available for both the State Machine and the UPROC.

4. Using the control portion of the buffer control logic, the State Machine enables the data onto the QBUS via BUFF 2. This results in the following status conditions:

   - BUFF 1 and BUFF 2 not active.

   - Buffer available for the UPROC, but no buffer is available for the State Machine.

3.2.4 QBUS State Machine

The QBUS State Machine implements the QBUS protocols discussed in Chapter 2. The state machine functions are listed below:

- Operates in idle loop mode.
- Performs DATI operations as master.
- Performs DATO operations as master.
- Performs DATI operations as slave.
- Performs DATO operations as slave.
- Issues interrupts to the processor.
- Passes DMA grants.
- Provides buffer control as discussed in the preceding section.

During the idle loop mode, the State Machine monitors input conditions from the QBUS and the KDA5Ø-Q to determine if it should stay in the idle loop, branch to the bus slave protocol, or branch to one of the DMA operations (DATI or DATO as master). If the KDA5Ø-Q is bus master, the State Machine monitors responses from the slave device and issues corresponding bus control signals. Part of the State Machine output is fed back to the input and helps determine the next state machine address.

3.2.5 29Ø1 ALU Slice

The 29Ø1 bit-slice ALU chip can perform addition, subtraction and logic operations. Its internal data paths and registers are four bits wide. Nine control signals and eight register-select lines determine the ALU function, operand source and destination.

The bit-slice concept allows many different configurations of micromachine components. For the KDA5Ø-Q, a 16-bit wide ALU is created by connecting four 29Ø1s in parallel. By itself, this ALU is not a microprocessor because it lacks control logic such as a program counter to determine the next instruction address, or a stack pointer to return from a branch or subroutine. As configured on the KDA5Ø-Q, the ALU has the following characteristics:

- A 2-port RAM comprised of sixteen 16-bit registers. The RAM registers are read via their A or B output ports.
- A 16-bit shift register on the RAM data input. The output of this register is connected in ring counter fashion to the input.

- A 16-bit Q-register for assisting with multiplication and division algorithms.
- A 16-bit shift register on the Q-register input.
- A multiplexer that selects the output of the Q-register shifter or the ALU output for input to the Q-register.
- Two multiplexers that select the ALU inputs. These inputs may be the data bus, the A or B RAM ports, the Q register, or a logical zero.
- An output multiplexer capable of selecting the ALU output or port A of the RAM and gating it to the data bus.
- A 2-input ALU capable of performing the following operations:

OPERATION	OPERANDS USED
Addition	R + S
	R + S + 1
Subtraction	R - S
	S - R
	R - S - 1
	S - R - 1
Logical OR	R or S
Logical AND	R and S
Logical NAND	NOT R and S
Logical XOR	R xor S
Logical XNOR	R xnor S

3.2.6 2911 Sequencer Slice

The 2911 Sequencer, like the 2901 ALU, is a bit-slice device which can be connected in parallel for wider data paths. Six 2911s form the UPROC and DPROC microprogram sequencers. Each resulting sequencer is 12 bits wide and is formed by connecting three 2911s in parallel. The following description refers to the sequencers as they are connected on the KDA50-Q.

The sequencers are the control section of the processor. They provide the program counter, the register file (stack), the stack pointer, and the internal logic to control these functions. The program counter is 12 bits wide and points to sequential locations in memory (CROM). It always points to the next instruction by automatically incrementing after the instruction fetch. The stack is 12 bits wide and 4 words deep and is used along with the stack pointer during branch operations.

Internal multiplexers within the sequencers are capable of choosing one of the following for output on the SEQADR lines <11:00>:

- The Branch Address supplied by the Pipeline Register.
- The Stack.
- The Program Counter.

NOTE

A reset returns the SEQADR lines to zero.

As connected on the KDA50-Q, the Sequencers can perform the operations listed below:

- CONTINUE - Increments the PC and places the new address on the SEQADR lines.
- RETURN FROM SUBROUTINE (Pop Stack) - Loads the PC with the current value of the stack and gates this value to SEQADR <11:00>. The stack pointer decrements.
- JUMP TO BRANCH ADDRESS - Loads the PC with the current value of the Branch Address and gates this address to SEQADR <11:00>. The stack and stack pointer are not affected.

- SUBROUTINE CALL (Push Stack) - Increments the stack pointer and pushes the current value of the PC onto the stack. Loads the PC with the current value of the Branch Address and gates this address to SEQADR <11:ØØ>.

- VECTOR TO ERROR - Loads SEQADR <11:ØØ> with FFX hex. This operation is explained in Section 3.2.7.

3.2.7 Micro Trap

The Micro Trap is an error-reporting circuit which monitors RAM parity error, Timeout, CROM parity error, and UPROC. Upon error, the Micro Trap circuit gates these signals to SEQ ADR <Ø3:ØØ> and disables the sequencer outputs, causing the assertion of SEQ ADR <11:Ø4>. The error vector then becomes FFX hex, and the four error vector signals determine the value of the don't cares (X). The following list describes the error vector signals:

- The RAM parity error occurs if a corrupted data word is read from the data buffer on the SDI module. Upon an error, RAM PEl L is gated to SEQADR <ØØ>, and the error vector becomes FFE hex.

- The Timeout error occurs when the DPROC sequencer does not execute an instruction within a timeout period of approximately 9Ø.7 milliseconds. If the DPROC tries a conditional execute instruction and the condition is not met,the DPROC does not execute the instruction. If this occurs a number of times, the timeout period expires. TIMEOUT L is gated to SEQADR <Ø1>, and the error vector becomes FFD hex. The timeout is not implemented for the UPROC.

- The CROM parity error is asserted when a corrupted instruction is read from CROM. Upon error, CROM PE L is gated to SEQADR <Ø2>, and the error vector becomes FFB hex.

- The UPROC line determines which sequencer was active when the error occurred. Upon error, UPROC H is gated to SEQADR <Ø3>, and the error vector becomes FF7 hex.

3.2.8 Pipeline Address/Data Register (PADR)

The ALU uses the PADR to place an address on the SEQADR lines <Ø7:ØØ>. For example, the ALU may execute a calculated jump by calculating the jump address from a table and writing the data to the PADR from the data bus. During this time, the outputs of the sequencers are disabled and the upper four SEQADR lines <11:Ø8> are asserted. The jump address becomes FXX hex. The don't cares (X) are determined by the eight PADR lines.

3.2.9 Pipeline Address Register

The Pipeline Address Register accepts 12 bits of address from the SEQADR lines. SEQ CLK L clocks the address to the CROM.

3.2.1Ø Control Store (CROM)

The CROM is an 8K x 48-bit Read Only Memory (ROM). It stores the KDA5Ø-Q microcode for the UPROC and DPROC sequencers. The allocated space for the UPROC is from address ØØØØ to 4Ø95, and for the DPROC from address 4Ø96 to 8191. The CROM is always enabled and, when addressed, the 48-bit microinstruction appears on the inputs of the Pipeline Register after the CROM propagation delay. Also, CROM lines <31:28> select the test conditions for the Test Multiplexer.

3.2.11 CROM Parity

The CROM parity circuit checks the microinstruction for parity. If the microinstruction has even parity (even number of ones), a CROM parity error is generated and the Micro Trap circuit reports the error as described in Section 3.2.7.

3.2.12 ALU Status Latches

The ALU Status Latches hold the current ALU status so the following ALU operation does not destroy it before it is tested. For example, if the ALU status was not latched and the UPROC stream set two of the ALU status lines, the next ALU operation (which the DPROC stream implements) would change the UPROC ALU status before it could be tested. PROCCLK L gates the latched data to the Test Mux input.

3.2.13 Test Multiplexer

The Test Multiplexer is the status check point of the system. It may be thought of as two test multiplexers: one for the UPROC, and the other for the DPROC. Using 4 bits from the CROM (before the pipeline), the Test Multiplexer chooses one of 16 possible test conditions for each sequencer. Four of these conditions are from the ALU and the rest are real-time conditions relating either to the UPROC or DPROC microcode stream.

During operation, the Multiplexer selects one of the ALU or real-time conditions and, if the test passes, supplies the PASS TEST H output. The Multiplexer may also test the input conditions for their true or false state. For example, if testing for the false state, the Multiplexer asserts PASS TEST H when the condition tested does not exist or is false. If testing for the true state, the Multiplexer asserts PASS TEST H when the condition tested exists or is true.

3.2.14 Pipeline Register

The Pipeline Register is 48 bits wide and gates microinstructions from CROM. The ALU and Sequencers process the Pipeline Register output, which is known as the current microinstruction. Also, the current microinstruction supplies the IOC selects along with the external source and destination selects. These select signals determine the IOC operation to be performed and the external source and destination locations.

3.2.15 UPROC Control Register

The UPROC Control Register is a write only register. The ALU writes the current value of the LED display to this register. A separate register stores the current value of the QBUS Control Register. By using this register, the value of the QBUS Control Register may be read.

3.2.16 Pipelining Concepts

Most of the processor module block diagram described in the preceding paragraphs is part of a pipelined system. Pipelined designs decrease instruction cycle time by performing fetch and execute operations in parallel. The Pipeline Register holds the current microinstruction that controls the processing section and helps the control section select the next address. Thus, the processing section is operating on the current microinstruction while the control section is fetching the next microinstruction from CROM.

3.2.16.1 KDA5Ø-Q Pipeline - The KDA5Ø-Q uses a form of Microinstruction/Address/Status Pipelining (Double Pipelining). Registers are provided for the current microinstruction (Pipeline Register), the sequencer address (Pipeline Address Register), and the ALU/real-time Status (Test Multiplexer). Due to these three registers, the Double Pipeline design has three parallel paths. These paths simultaneously perform the following operations:

- The current microinstruction is gated into the pipeline register. It controls the ALU along with the branch address of the UPROC and DPROC sequencers.
- The Sequencers generate a new value for the Pipeline Address Register.
- The current value in the Pipeline Address Register fetches the next microinstruction from CROM. CROM <31:28> selects the test conditions for the Test Multiplexer.

3.2.16.2 Pipeline Timing - Figure 3-3 shows the double pipeline design of the KDA5Ø-Q. The signals present at the input or output of each block result from the UPROC Sequencer or the DPROC Sequencer. For example, the output of the Pipeline Address Register is either the UPROC SEQ ADR or the DPROC SEQ ADR.

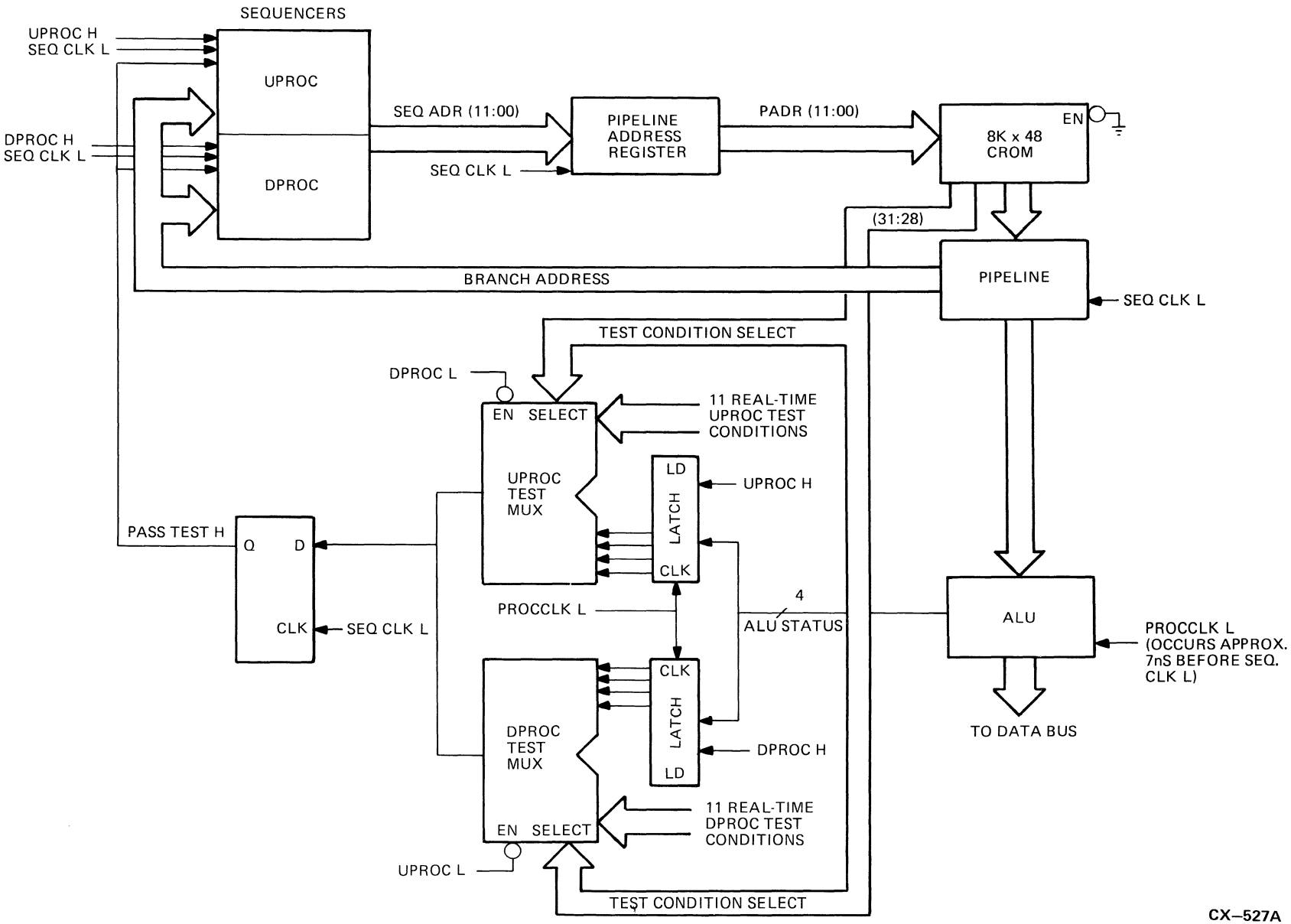


Figure 3-3 KDA5Ø-Q Double Pipelining

SEQ CLK L supplies the timing for all the control signals shown on the block diagram. Figure 3-4 shows the timing relationship of SEQ CLK L, PROCCLK L, UPROC H and DPROC H.

NOTE

The DPROC H signal name is used to help clarify the timing relationships. In the actual circuit schematics, the UPROC L signal performs the activities assigned to DPROC H. This may be done because UPROC L has the same timing characteristics as DPROC H.

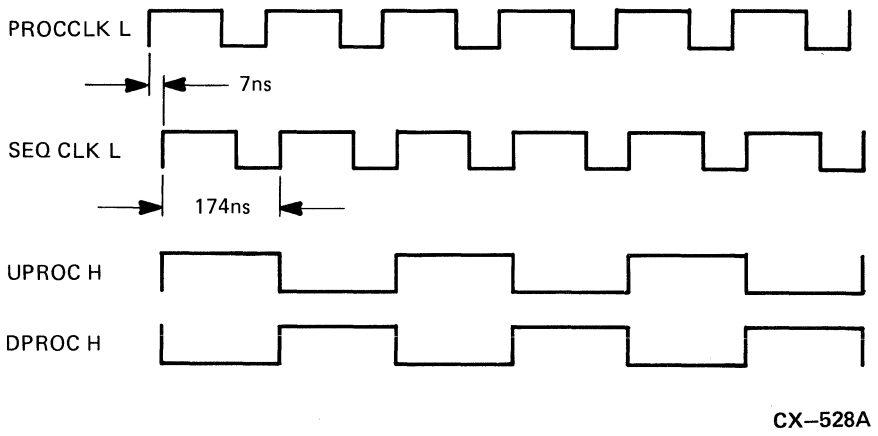


Figure 3-4 Pipeline Timing

The signals shown in Figure 3-4 enable, disable, and clock the circuits shown in Figure 3-3. UPROC H and DPROC H provide enabling and disabling signals; SEQ CLK L and PROCCLK L provide the clocking. The following paragraphs describe their effects on each of the blocks.

UPROC H, DPROC H, and SEC CLK L control the Sequencer timing. UPROC H and DPROC H enable their corresponding sequencers. A low-to-high transition of SEQ CLK L gates the data from the enabled sequencer to SEQ ADR <11:ØØ>.

SEQ CLK L controls the Pipeline Address Register timing. A low-to-high transition of SEQ CLK L gates the input of the register to PADR <11:ØØ>.

The CROM is not clocked and is always enabled. Therefore, it is not affected by the control signals. However, it does supply some control signals to the UPROC and DPROC Test Mux. These signals select the desired test condition from the 16 (15 actual conditions and the last which always equals a logical one) available at the input of the Test Mux.

SEQ CLK L controls the Pipeline Register timing. A low-to-high transition of SEQ CLK L gates the input of the register to the branch address lines and to the ALU. This microword is called the current microinstruction.

PROCCLK L controls the internal ALU timing. PROCCLK L occurs approximately 7 nanoseconds before SEQ CLK L to avoid race conditions on the ALU.

UPROC H and DPROC H cause the Test Mux latches to perform a load operation. The present ALU status is latched so the next ALU operation cannot alter it before it is tested. A low-to-high transition of PROCCLK L gates the latched data to the Test Mux input.

The condition select lines from the CROM choose one of the input conditions for the Test Mux. UPROC L and DPROC L enable the selected condition to the test circuit which determines if the selected bit is true or false. For simplicity, the test circuit is considered part of the Test Mux in the block diagram.

A low-to-high transition of SEQ CLK L gates the tested condition through the flip flop to the sequencers. This last step synchronizes the tested condition with the system.

3.2.16.3 Pipeline Operation - As previously described, either the UPROC or the DPROC sequencer generates the output of any of the blocks shown in Figure 3-3. For example, assume UPROC H is true and a low-to-high transition occurs on SEQ CLK L. The following conditions appear on the block diagram:

1. SEQ ADR <11:ØØ> will contain data from the UPROC sequencer. The sequencer determines this data from the UPROC branch address and UPROC test bit described in steps 4 and 7, respectively.

2. PADR <11:ØØ> contains data from the DPROC sequencer. This data is addressing the CROM and constitutes a fetch operation from memory.

3. The CROM output is a DPROC microinstruction because the PADR contains an address from the DPROC sequencer. This output selects the test conditions for the DPROC Test Mux and is applied to the input of the Pipeline Register.

4. The Pipeline Register output is a UPROC microinstruction. The pipeline sends the branch address portion of the microinstruction to the UPROC Sequencer and the ALU control portion to the ALU. Notice the current microinstruction (UPROC) corresponds to the enabled sequencer.

5. The ALU operates on the current microinstruction and supplies the UPROC ALU status to the UPROC Test Mux Latch.

6. Because UPROC H is true, the UPROC Test Mux Latches the UPROC ALU status. This status is sent to the UPROC Test Mux on the next low-to-high transition of PROCCLK L.

7. UPROC L is true and enables the output of the DPROC Test Mux. Using the test condition select lines, the DPROC Test Mux chooses the test condition. The select lines on the DPROC Test Mux are the output of the CROM given in step 3 above. Notice these select lines are generated by the DPROC sequencer.

8. Because of the low-to-high transition of SEQ CLK L, PASS TEST H is a UPROC-generated test bit.

The steps listed above tend to indicate an order of operation. However, SEQ CLK L occurs at the same time on the Sequencer, Pipeline Address register, Pipeline Register, and FF blocks. Because of this, their operations occur simultaneously or in parallel.

Figure 3-5 shows the pipeline in the state described in the previous eight steps.

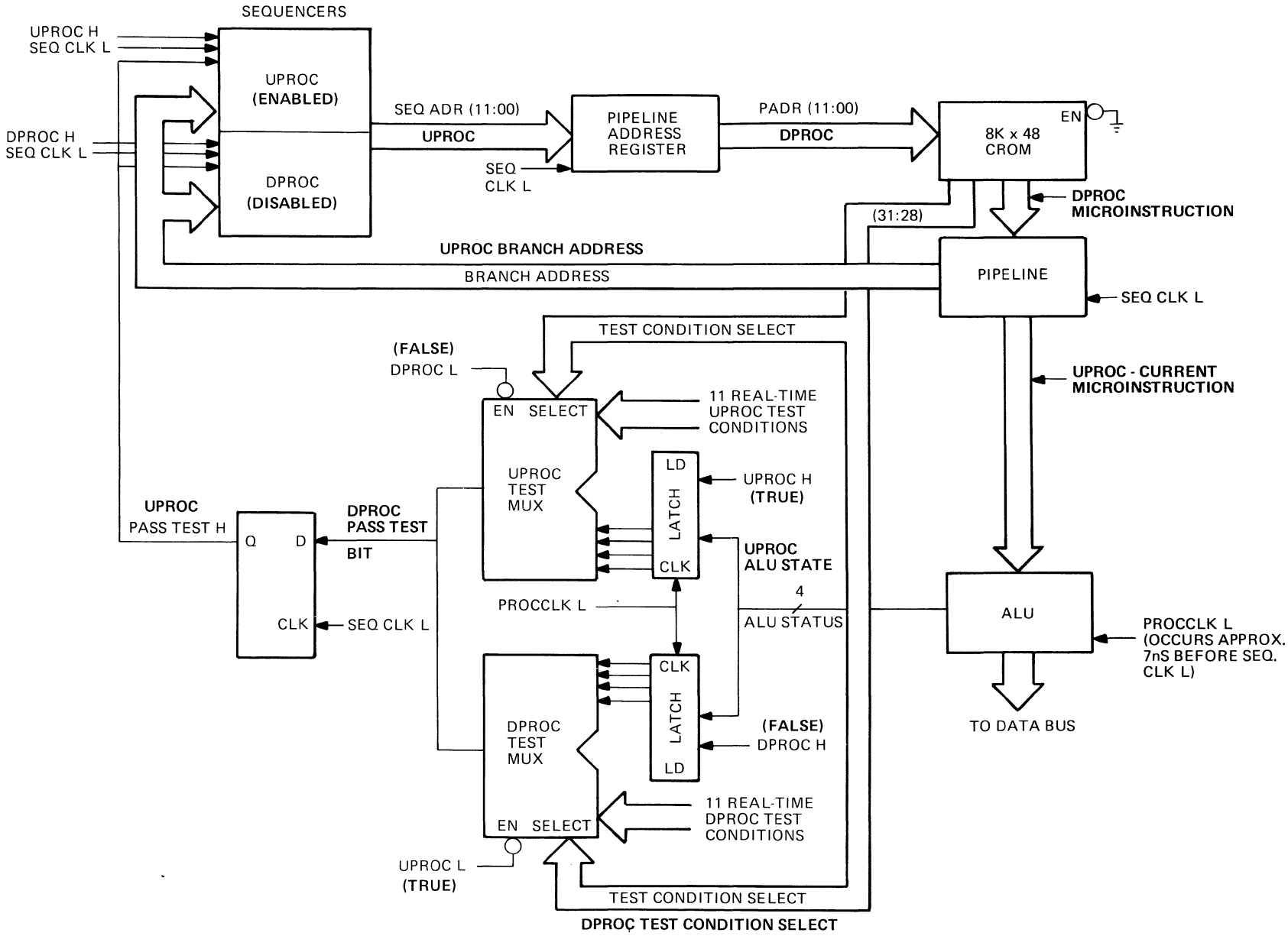


Figure 3-5 Pipeline Data

On the next SEQ CLK L, the other sequencer determines the outputs of the blocks in Figure 3-5. For example, each output labeled UPROC is changed to DPROC. Also, everything previously enabled is disabled and everything previously disabled is enabled.

3.2.16.4 Pipeline Data Movement - Figure 3-6 illustrates the data movement found in the double pipelined design. In the figure, "U" stands for UPROC and "D" stands for DPROC. The numbers keep track of the microcode stream that caused the address or result. For example, D3 results from the DPROC stream and it is the third such result shown in the figure.

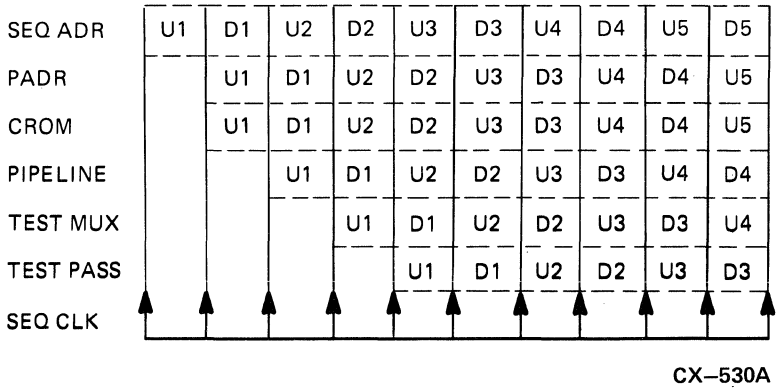


Figure 3-6 Pipeline Data Movement

The last column in Figure 3-6 shows the DPROC sequencer generating a new address for the SEQ ADR lines. This address, labeled D5, is D4 + 1 unless the DPROC is executing a subroutine call, a return from subroutine, or a jump instruction. If executing one of these instructions, the branch address is supplied by the D4 instruction in the pipeline and the condition tested is the D3 Pass Test bit.

3.3 SDI BOARD BLOCK DIAGRAM

Figure 3-7 is a block diagram of the KDA5Ø-Q SDI module (M7165). The 16-bit Bidirectional Data Bus is common between the SDI module and the processor module. The following paragraphs provide a functional description of the blocks shown.

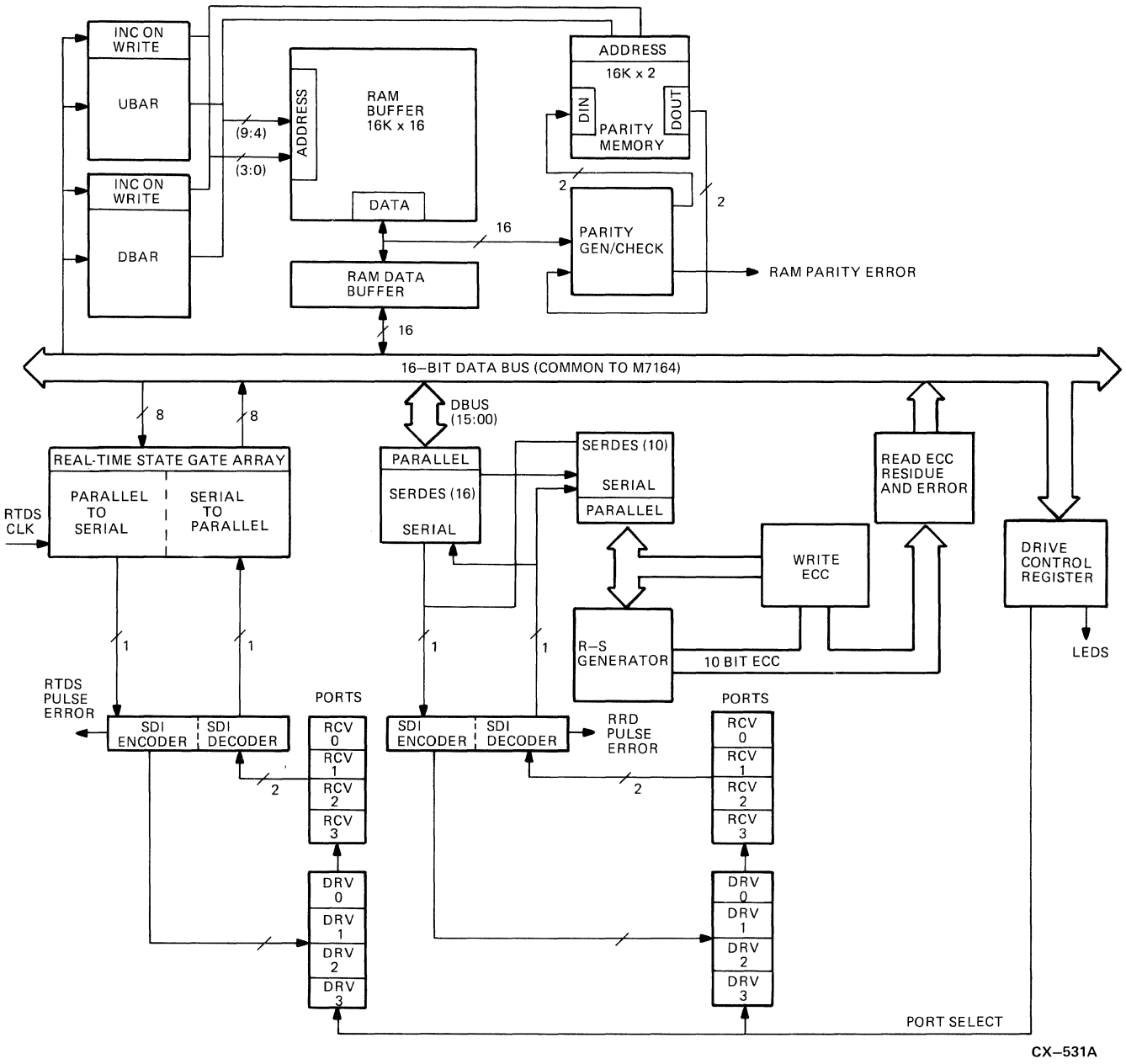


Figure 3-7 M7165 Block Diagram

3.3.1 U And D Buffer Address Register

The UBAR or DBAR gate the address word from the 16-bit Data Bus to the 16K x 16-RAM buffer and Parity Memory. The lower four bits of each register increment during a Write to the RAM buffer. Thus, up to 16 words can be written to sequential locations using a single starting address. Incrementing is possible only during Writes to the buffer, not during a read operation.

3.3.2 RAM Data Buffer

The RAM data buffer moves 16-bit data words between the data bus and the RAM buffer. In practice, two registers make up the RAM data buffer: one for data transfers from the RAM buffer to the DBUS, and the other for data transfers from the DBUS to the RAM buffer.

The controlling signals and their operation for both the registers in the RAM Data Buffer are listed below:

- READ H -- When READ H is low or false it enables writes to the RAM buffer.
- LD BUFFER L -- During a write to the RAM buffer, the rising edge of LD BUFFER L gates data to the RAM buffer. When asserted low, this signal disables READ H causing it to be false.
- RD BUFFER L -- RD BUFFER L enables reads to the RAM buffer when low or true.
- CLK BDR L -- During a read from the buffer, the rising edge of CLK BDR L gates data from the RAM buffer.

3.3.3 RAM Buffer

The RAM Buffer has 16,384 locations, and each is 16-bits wide (16K x 16 or 32K bytes). It provides temporary storage for the UPROC and DPROC. During disk data transfers, it compensates for the difference in speed between the disk drive and the host. All data transfers must pass through this buffer.

3.3.4 Parity Circuit

The Parity Circuit is comprised of the Parity Generator/Checker (PGC) and the Parity Memory. These circuits generate and check parity on the high and low bytes of the RAM data word. The PGC and the Parity Memory operate differently during Read RAM buffer operations than during Write RAM Buffer operations.

During a write, the following events occur:

- The Buffer Address Register (BAR) addresses the RAM Buffer and the Parity Memory.
- Sixteen bits of data are moved to the input of the RAM Buffer and the PGC.
- The PGC generates two parity bits--one for each byte in the data word. These bits are sent to the data input of the Parity Memory.
- Both the RAM Buffer and the Parity Memory store their input data at the address supplied by the BAR.

During a read, the following events occur:

- The BAR addresses the RAM Buffer and the Parity Memory.
- The RAM Buffer sends data to the RAM Data Buffer and the PGC.
- Parity Memory sends the two parity bits to the PGC.
- The PGC checks the parity on the high and low bytes of the RAM buffer word using the buffer word plus the two parity bits supplied by the Parity Memory.
- If parity has not changed since it was originally written, the PGC does not generate a parity error.

NOTE

The parity on the high and low bytes of the RAM Buffer word is usually odd (odd number of ones). The microcode may set the Parity Generator/Checker so it tests for even parity (even number of ones). This is usually done during diagnostics to ensure the parity circuit is operating correctly.

3.3.5 Real-Time State Gate Array

The Real-Time State Gate Array is a communications interface between the parallel format of the Data Bus and the serial format of the real-time state lines (RTCS and RTDS). The RTCS side of the Gate Array converts eight bits of parallel controller state information to an NRZ serial data stream. The RTDS side of the Gate Array converts the NRZ serial drive state information to 8-bit parallel format for the data bus.

3.3.6 SDI Encoder/Decoder (ENC/DEC)

The two SDI ENC/DEC blocks convert data between serial NRZ format and serial SDI format. They also monitor all the SDI lines and report errors in the electrical characteristics of the SDI data.

3.3.7 SERDES 1Ø, SERDES 16, And R-S Gen

The SERDES 1Ø, the SERDES 16, and the R-S Gen work together to provide serial-to-parallel data conversion and ECC calculation. The SERDES 1Ø and 16 convert data from serial-to-parallel and from parallel-to-serial, and operate on the write command (WCD) and read response data (RRD) lines. The R-S Gen calculates a 17Ø-bit ECC (17 1Ø-bit symbols) for every sector of data written to the disk. This calculated ECC is appended to the data and written at the end of the disk sector. Later, as the sector is read, the R-S Gen recalculates the ECC from the read data. Then, using an EXCLUSIVE-OR process, the R-S Gen compares the two ECC values. If the ECC values do not agree, the R-S Gen supplies the result of the compare operation to the UPROC. The UPROC then uses an algorithm to correct the errors that it is capable of correcting--that is, any number of errors contained in up to eight of the 1Ø-bit symbols in the data field. The following paragraphs describe the operation of the SERDES 1Ø, SERDES 16, and the R-S Gen during message and data transfers.

NOTE

Detailed descriptions of the SERDES and the R-S Gen chips are provided in the appendix.

During a command, the SERDES 1Ø and the R-S Generator are not used and no ECC is written or calculated. RRD is read by putting the SERDES 16 in serial-to-parallel mode. Data appearing at the SERDES 16 input from the SDI ENC/DEC is converted to parallel format and stored in the RAM Buffer. WCD is written by putting the SERDES 16 in parallel-to-serial mode. The data is moved from the RAM Buffer, converted to serial format at the SERDES 16, and sent to the SDI ENC/DEC.

When the SERDES is involved in a data transfer, an ECC must be calculated. During a Write, the calculated ECC is sent with the data to the disk drive. During a Read, the calculated ECC is compared with the ECC from the drive to ensure the data is correct.

The following occurs during a write data operation:

- The SERDES 16 is in parallel-to-serial mode.
- Parallel data is read from the buffer to the SERDES 16 and converted to serial data. The serial data is transferred to the SERDES 1Ø.
- The SERDES 1Ø is in serial-to-parallel mode and does the following:

  1. Sends the data out serially to the SDI ENC/DEC.
  2. Converts the data to 1Ø-bit parallel quantities and transfers it to the R-S Generator.

- The R-S Generator calculates the ECC from the 1Ø-bit quantities and stores the ECC characters in its internal memory.
- At word number 256 in the data transfer, the SERDES 1Ø switches modes and becomes a parallel-to-serial converter.
- The ECC characters are read from the R-S generator via the WRITE ECC register. The SERDES 1Ø converts the 1Ø-bit parallel format to serial format and sends the ECC to the SDI ENC/DEC.

The following occurs during a read data operation:

- The SERDES 16 is in serial-to-parallel mode.
- The SERDES 16 converts the serial data from the SDI ENC/DEC block to 16-bit parallel quantities. This 16-bit parallel word is written into the RAM Buffer.
- During this time, SERDES 1Ø converts the serial data from the SDI ENC/DEC to 1Ø-bit parallel quantities and sends this data to the R-S Generator.
- The R-S Generator calculates an ECC based on this data.
- When the ECC characters from the disk drive arrive at the R-S generator, the generator compares the calculated ECC with the ECC from the drive. Two possible results follow:

  1. The calculated and actual ECC characters are the same and no error is generated.

  2. A difference occurs between the calculated and actual ECC. In this case, the DPROC reads the R-S generator contents via the READ ECC RSDU register. The DPROC informs the UPROC of the error and the UPROC then performs ECC error correction via an error handler in the microcode.

3.3.8 Drive Control Register

The Drive Control Register (DCR) is a write only register. The UPROC or DPROC microcode stream writes the current value of the LED display to this register. A separate register stores the current value of the DCR. By using this register, the value of the DCR may be read.

CHAPTER 4
KDA5Ø-Q MICROINSTRUCTION FORMAT

4.1 INTRODUCTION

The KDA5Ø-Q microinstruction is 48-bits wide. Included in this instruction word are branch addresses, ALU control, Data Bus control, input/output control (IOC), test condition select, and other test and control bits. Because one microinstruction performs so many functions, it is termed a horizontal microinstruction. This format differs from vertical instruction formats found in traditional microprocessors which perform one function per microinstruction. The following list describes some of the differences between vertical and horizontal microinstruction formats:

- A horizontal microinstruction contains many different fields that control a part of the data flow. However, a vertical microinstruction contains relatively few fields. Listed in the preceding paragraph are some of the individual fields in a KDA5Ø-Q microinstruction. In comparison, a vertical microinstruction may contain only control and data or register fields.

- A horizontal microinstruction may perform many operations in parallel. A vertical microinstruction may perform only a single sequence of events.

- Horizontal formats are usually faster than vertical designs.

4.2 KDA5Ø-Q MICROINSTRUCTION

Figure 4-1 shows the KDA5Ø-Q microinstruction. The following paragraphs describe the fields and their function.

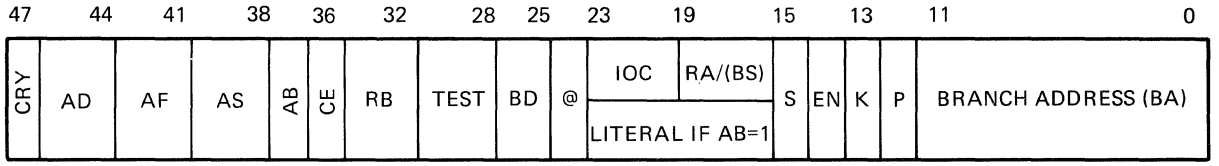


Figure 4-1 KDA5Ø-Q Microinstruction Format

4.2.1 CRY Field

If ALU function bits are zero through two, the CRY bit is the carry into the LSB of the ALU. If the ALU function bits are three through seven, the CRY bit, when set, also enables the high byte of the literal word to the ALU data inputs.

4.2.2 AD Field

The three AD bits determine the 29Ø1 ALU-operand-destination address and some of the 29Ø1's internal operations. The following list describes the possible destinations:

- AD = Ø--The ALU output goes to the internal Q-register and is gated to the Data Bus <15:ØØ>
- AD = 1--The ALU output is gated to the Data Bus <15:ØØ>.
- AD = 2--The ALU output is written to internal RAM at the location specified on the RB bits. The data at the A port of the internal RAM is gated to the Data Bus <15:ØØ>.
- AD = 3--The ALU output is written to internal RAM at the location specified on the RB bits. The ALU output is gated to the Data Bus <15:ØØ>.
- AD = 4--The ALU output is gated to the Data Bus <15:ØØ>. It is also shifted right (ALU/2) and the result written to internal RAM at the location specified on the RB bits. The internal Q-register is shifted right (Q/2).

- AD = 5--The ALU output is gated to the Data Bus <15:ØØ>. It is also shifted right (ALU/2) with the result written to internal RAM at the location specified on the RB bits.

- AD = 6--The ALU output is gated to the Data Bus <15:ØØ>. It is also shifted left (ALU x 2) with the result written to internal RAM at the location specified on the RB bits. The internal Q-register is shifted left (Q x 2).

- AD = 7--The ALU output is gated to the Data Bus <15:ØØ>. It is also shifted left (ALU x 2) with the result written to internal RAM at the location specified on the RB bits.

NOTE

The 29Ø1 outputs are enabled only if bit 37 (AB) is set in the current microinstruction.

4.2.3 AF Field

The three AF bits determine the function the ALU performs. Listed below are the possible AF codes:

- AF = Ø; ADD R + S If carry in is high, R + S + 1
- AF = 1; SUB S - R If carry in is low, R - S - 1
- AF = 2; SUB R - S If carry in is low, S - R - 1
- AF = 3; Logical Or R OR S
- AF = 4; Logical And R AND S
- AF = 5; Logical Nand Not R AND S
- AF = 6; Exclusive Or R XOR S
- AF = 7; Exclusive Nor R XNOR S

4.2.4 AS Field

The three AS bits determine the 2901 ALU source operands. These operands are sent to the two inputs (R and S) of the ALU. The following table shows the possible source operands.

Table 4-1 ALU Source Operands

AS Value	R Operand	S Operand
0	Data from internal RAM A port	Data from Q-register
1	Data from internal RAM A port	Data from internal RAM B port
2	Logical 0	Data from Q-register
3	Logical 0	Data from internal RAM B port
4	Logical 0	Data from internal RAM A port
5	If AB = 0, the operand is Data Bus <15:00>. If AB = 1, the operand is the literal word. This AS value makes the address and data of the RAM A port equal the RAM B port.	Data from internal RAM A port.
6	If AB = 0, the operand is Data Bus <15:00>. If AB = 1, the operand is the literal word.	Data from the Q-register
7	If AB = 0, the operand is Data Bus <15:00>. If AB = 1, the operand is the literal word. This AS value makes the address and data of the RAM A port equal the RAM B port.	Logical 0

4.2.5 AB Field

When set, the AB bit enables the ALU output to Data Bus <15:ØØ> and gates the literal word to the ALU input bus. When reset, it gates Data Bus <15:ØØ> to the ALU input bus, indicates the bus source data is on Data Bus <15:ØØ>, and disables the ALU output.

4.2.6 CE Field

The conditional execute bit (CE) determines if an operation should be executed on a true condition or a false condition from the Test Multiplexer. If the CE bit is reset, the operation executes on a false condition. If the CE bit is set, the operation executes on a true condition. The EN bit must be true or the CE bit has no effect.

4.2.7 RB Field

The four RB bits determine the B port RAM address. During a RAM read operation, the RAM gates data from this location through the B port. During a RAM write operation, the RAM writes data at the location the RB bits specify.

4.2.8 TEST Field

The four TEST bits determine the input bit select on the Test Multiplexer. They may choose any of the 16 possible test conditions (Ø - 15) for each sequencer.

4.2.9 BD Field

The three bus destination bits (BD) enable write ports on the 16-bit Data Bus. These ports are located on both the processor module and the SDI module. The following table shows the bus destination address and the port it enables.

Table 4-2 Bus Destinations

BD Value	Port Enabled	Comment
Ø	NONE	
1	NONE	
2	PAR ENA	This signal enables the PAR, which gates Data Bus <7:Ø> to SEQ ADR <7:Ø>.
3	LD Buffer	This Signal gates data from Data Bus <15:ØØ> to the RAM Buffer data input on the SDI module.

The following port selects depend on the state of UPROC and DPROC. The same BD value causes two port selects. One for the UPROC state and one for the DPROC state.

4	LD DCR and LD QCR	The Drive Control Register (DCR) is loaded during the DPROC state. The QBUS Control Register (QCR) is loaded during the UPROC State
5	LD RTCS and NONE	LD RTCS gates data from Data Bus <15:ØØ> to the real-time state gate array on the SDI module and occurs during the DPROC state. During the UPROC state, this BD value does not select a port.
6	LD SERDES and LD QDAR	LD SERDES gates data from Data Bus <15:ØØ> to the SERDES 16 and occurs during the DPROC state. LD QDAR gates the extended address bits from Data Bus <7:Ø> to the extended address transceiver and occurs during the UPROC state.

Table 4-2 (cont.)

BD Value	Port Enabled	Comment
7	LD DBAR and LD UBAR	LD DBAR gates data from the Data Bus <15:ØØ> to the BAR lines and occurs during the DPROC state. LD UBAR gates data from the Data Bus <15:ØØ> to the BAR lines and occurs during the UPROC state.

4.2.1Ø @ Field

The Input Output Control enable bit, @, enables the IOC functions on the SDI module and on the QBUS state machine.

4.2.11 IOC Field

The four IOC bits determine the IOC function. For example, an IOC of one enables the R-S Generator on the SDI module.

4.2.12 LITERAL Field

If CRY = Ø and AF = 1-2, the 8 LITERAL bits move to the low-byte data input of the ALU. If CRY = 1 and AF = 3 through 7, the 8 LITERAL bits move to the high-byte data input of the ALU.

4.2.13 RA Field

The four RA bits determine the A port RAM address. During a RAM read operation, the RAM gates data from this location through the A port.

4.2.14 [BS] Field

If AB = Ø, the four bus destination bits [BS] enable read ports on the 16-bit Data Bus. These ports are located on both the processor module and the SDI module. Table 4-3 shows the bus destination address and the port it enables. BUS SRC 3 is a direction bit that determines the direction of data movement between Data Bus <15:ØØ> on the SDI module and Data Bus <15:ØØ> on the processor module. If set, the BUS SRC 3 bit indicates a transfer from Data Bus <15:ØØ> on the processor module to Data Bus <15:ØØ> on the SDI module.

Table 4-3 Bus Sources

[BS] Value (octal)	Port Enabled	Comment
Ø	NONE	The Data Bus <15:ØØ> enabled direction is from the SDI module to the processor module.
1	NONE	The Data Bus <15:ØØ> enabled direction is from the SDI module to the processor module.
2	RD RTDS	RD RTDS gates the real-time drive status from the RTS gate array on the SDI module to seven bits of the Data Bus. The Data Bus <15:ØØ> enabled direction is from the SDI module to the processor module.
3	RD BUFFER	RD BUFFER gates data from the RAM data buffer on the SDI module to Data Bus <15:ØØ>. The Data Bus <15:ØØ> enabled direction is from the SDI module to the processor module.
4	NONE	The Data Bus <15:ØØ> enabled direction is from the SDI module to the processor module.
5	NONE	The Data Bus <15:ØØ> enabled direction is from the SDI module to the processor module.

Table 4-3 (cont.)

[BS] Value (octal)	Port Enabled	Comment
6	RD SERDES	RD SERDES gates data from the SERDES 16 to Data Bus <15:ØØ>. The Data Bus <15:ØØ> enabled direction is from the SDI module to the processor module.
7	RD ECC RSDU	RD ECC RSDU gates data from the ECC RSDU register on the SDI module to Data Bus <15:ØØ>. The Data Bus <15:ØØ> enabled direction is from the SDI module to the processor module.
1Ø	SAME AS Ø EXCEPT:	The Data Bus <15:ØØ> enabled direction is from the processor module to the SDI module.
11	SAME AS 1 EXCEPT:	The Data Bus <15:ØØ> enabled direction is from the processor module to the SDI module.
12	SAME AS 2 EXCEPT:	The Data Bus <15:ØØ> enabled direction is from the processor module to the SDI module.
13	SAME AS 3 EXCEPT:	The Data Bus <15:ØØ> enabled direction is from the processor module to the SDI module.
14	SAME AS 4 EXCEPT:	The Data Bus <15:ØØ> enabled direction is from the processor module to the SDI module.

Table 4-3 (cont.)

[BS] Value (octal)	Port Enabled	Comment
15	SAME AS 5 EXCEPT:	The Data Bus <15:ØØ> enabled direction is from the processor module to the SDI module.
16	SAME AS 6 EXCEPT:	The Data Bus <15:ØØ> enabled direction is from the processor module to the SDI module.
17	SAME AS 7 EXCEPT:	The Data Bus <15:ØØ> enabled direction is from the processor module to the SDI module.

4.2.15 S Field

The Test True (S) bit directs the Test Multiplexer to test for a true condition if set and test for a false condition if reset.

4.2.16 EN Field

The EN bit enables conditional execution based on the result of the test performed by the test multiplexer. See CE bit.

4.2.17 K Field

The K bit performs the following three functions if the test at the test multiplexer passes:

- Call-to subroutine. This operation occurs when K = 1 and the BA field does not equal 777 (Octal).
- Return-from subroutine. This operation occurs when K = 1 and the BA field does equal 777 (Octal).
- Jump. This operation occurs if K = Ø.

The call-to subroutine function pushes the current PC value onto the stack and gets a new PC value from the BA field. The return-from subroutine operation pops the new PC value from the stack and returns to the calling routine. The jump operation gets a new PC value from the BA field; however, it does not utilize the stack.

4.2.18 P Field

The P bit supplies odd CROM parity (odd number of ones in the CROM word).

4.2.19 BA Field

The BRANCH ADDRESS (BA) bits supply the address to branch to in case of a jump or return from the subroutine.

4.3 SAMPLE MICROINSTRUCTIONS

The following sample microinstructions illustrate some of the functions performed by the microinstruction fields. The microinstructions are given in binary and are followed by paragraphs that help decode them.

4.3.1 Sample 1

This sample is a move instruction. Data moves from the 16-bit internal data bus to RAM register #3 (R3) within the 2901 ALU. The data on the data bus is not changed.

47 32	31 16	15 Ø
ØØ11Ø1111ØØØØ11	ØØØØØØØØØØØØØ11	ØØØØØØØØØØØØØØØØ

The following list shows the fields previously discussed and their values for this example:

- CRY = Ø
- AD = 3
- AF = 3

- AS = 7
- AB = Ø
- CE = Ø
- RB = 3
- TEST = Ø
- BD = Ø
- @ = Ø
- The IOC/LITERAL field = Ø
- RA = 3
- S = Ø
- EN = Ø
- K = Ø
- P = Ø
- BA = Ø

The AD field supplies the ALU destination and also some of the internal ALU functions. According to this field, the ALU output is written to internal RAM at the location given by RB. Also, according to this field, the ALU output is written to the data bus. However, the ALU output is not written to Data Bus <15:ØØ> because the AB field is reset, disabling the external ALU output.

The AF field supplies the ALU function. In this case, the ALU is performing a logical OR operation.

The AS field supplies the two operands for this operation. Table 4-1 describes the AS fields. The AS value of 7 causes an R operand of Data Bus <15:ØØ> and an S operand of Ø. As a result of this operation, the ALU output is the value of Data Bus <15:ØØ>.

This example microinstruction performs a move operation. The data on Data Bus <15:ØØ> is ORed with Ø (do nothing) and written to RAM register 3 (R3). This instruction may be written:

MOV (Data Bus), R3.

4.3.2 Sample 2

Sample 2 performs a wait loop. It also illustrates one of the differences between horizontal and vertical microinstructions.

Using vertical microinstruction formats, a wait loop may require as many as four instructions to be implemented. In a typical wait loop, the first instruction loads the wait value in a register, the second instruction decrements the register, the third instruction determines if the register is at zero, and the fourth instruction supplies the branch address. When the register contents equal zero, the loop ends and the machine moves on to the next microinstruction.

Using a horizontal format, the same wait loop may be accomplished using two microinstructions. The first loads the wait value in a register, and the second decrements the register, determines if the register is at zero, and supplies the branch address.

For the following sample, assume RØ is a register containing a previously loaded wait value and the instruction is at address 77 (Octal). This sample will decrement RØ, test to determine if it has reached zero, and supply the branch address (77 Octal). The instruction branches to itself until RØ is decremented to zero and then continues with the next instruction.

47 32	31 16	15 Ø
ØØ11ØØ11ØØ1ØØØØØ	ØØØ1ØØØØØØØØØØØ	ØØØ1ØØØØØ111111

The following list shows the microword fields and their values for this example. A brief description of the function of each field is also given.

- CRY = Ø. Because the ALU Function (AF) is between zero and two, this bit is the carry bit into the LSB of the ALU.
- AD = 3. This value causes the ALU output to move into the internal RAM register specified by the RB field (in this case, RØ). It also causes the ALU output to move to the 16-bit internal data bus.
- AF = 1. This value causes the ALU to do a subtract operation. If the CRY bit is low, as it is in this sample, the ALU performs R - S - 1.
- AS = 4. This value causes an R operand of logical zero. The S operand is the data in internal RAM at the location specified in the RB field (RØ).

- AB = 1. This gates the ALU output to the 16-bit internal data bus.
- CE = Ø. Since the EN bit is reset, the CE bit has no effect.
- RB = Ø. RB is the address of the internal RAM register where the ALU result is written.
- TEST = 1. This tests the ALU output to determine if it is equal to zero. The tested signal is an ALU condition code: F = Ø, true when high.
- BD = Ø. A BD value of Ø does not enable any write ports.
- @ = Ø. The IOCs are not enabled.
- LITERAL = Ø. Although the literal word is gated to the ALU input bus per the AB field, it has no effect because the ALUs input bus is not one of the operands specified by the AS field.
- [BS]. This field is not used if the AB field equals one.
- RA = Ø. RA is the address of the internal RAM register where the S operand is located. This value is RØ.
- S = Ø. The S field, when reset, instructs the Test Multiplexer to test for a false condition.
- EN = Ø. If the EN field is reset, conditional execution is disabled.
- K = Ø. The K field, when reset, allows a jump to the address supplied in the BA field. In this example, the jump address is 77 (Octal).

In this sample, the wait value resides in RØ and is decremented because of the AF and AS fields. The AF field supplies the (R - S - 1) operation. The AS field further defines this operation as [Ø - (the contents of RØ) - 1], and effectively performs a decrement RØ function.

The TEST field determines the bit or condition code tested by the test multiplexer. In this sample, the condition code F = Ø is tested for the false state because the S field is equal to zero. Therefore, until the condition code is true (ALU output equals zero), the test multiplexer supplies the Pass Test signal. The operation of the test multiplexer is described in Chapter 3, Section 3.2.13.

NOTE

The condition code tested at the test multiplexer results from the previous instruction. For example, if the result of a UPROC instruction asserts a condition code, that condition code is tested during the next UPROC instruction.

The jump operation is performed because the K field equals zero and the test passes until RØ is decremented to zero. When RØ is equal to zero, the next instruction tests the condition code and the test fails because F = Ø is true. At this time, the wait loop has been departed from and the processor performs the microinstruction at location 78 (Octal). This instruction may be written:

```
77 DEC  RØ  JNZRO 77
```

CHAPTER 5
KDA5Ø-Q ERROR HANDLING

5.1 INTRODUCTION

This chapter describes the possible KDA5Ø-Q real-time errors found in the status/event codes of the error log printout. A table containing the status/event codes may be found in the appendix. Each error entry contains the following information:

- A brief description of the error.
- Where the error is detected.
- How the error is reported.
- If the error is retried and, if so, the amount of retries allowed.
- The error threshold points (where applicable). For example, at what point does the error become significant enough to report to the host?
- Some possible causes for the error.

Figure 5-1 shows the hardware relationship between the host and the KDA5Ø-Q. Figure 5-2 shows the software/firmware relationship between the host software and the KDA5Ø-Q firmware. These figures show the responsibilities of both the host and KDA5Ø-Q in a working system, and help when analyzing the errors listed in this chapter.

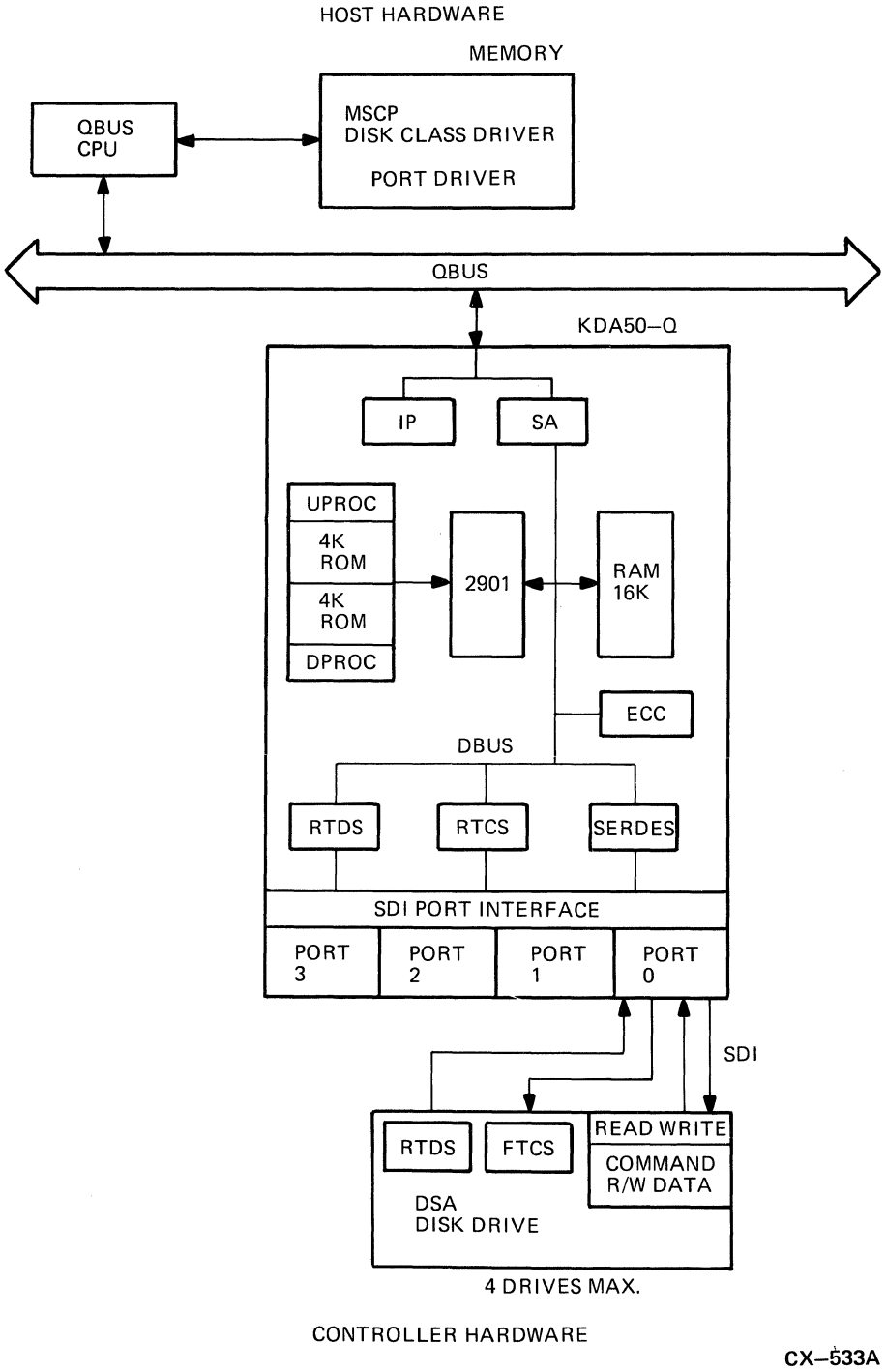


Figure 5-1 System Hardware Relationship

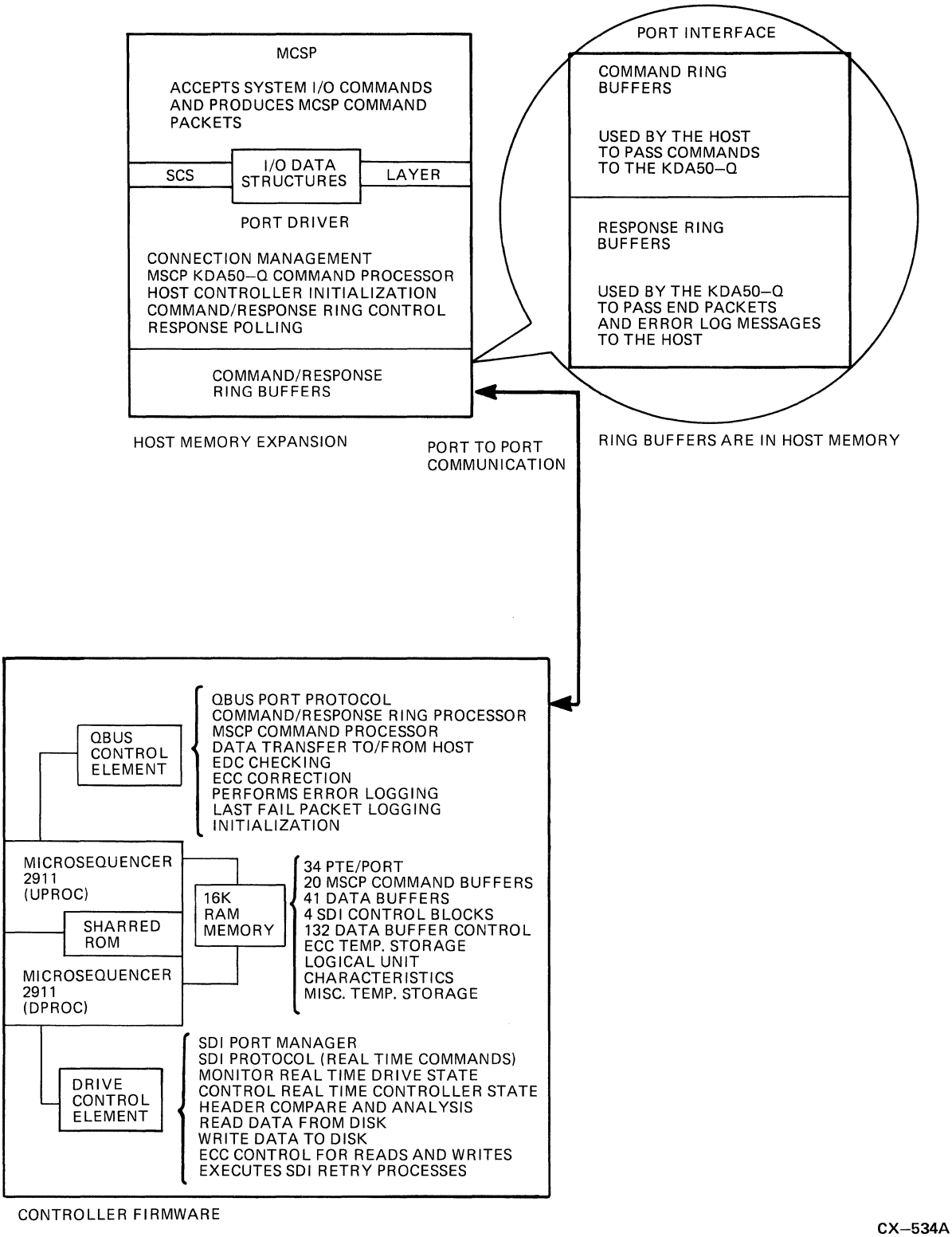


Figure 5-2 System Software/Firmware Relationship

5.2 MSCP PACKET KDA5Ø-Q TO HOST OP CODES

In an error log entry, the following four format codes are possible:

- Controller Error.
- QBUS Access Error.
- Disk Transfer Error.
- SDI Error.

This chapter describes the possible errors that may be observed for each category.

5.3 LAST FAIL PACKET FORMAT

Sections 5.3.1 to 5.3.16 describe the possible internal KDA5Ø-Q errors. All of the internal errors are logged within the Last Fail Packet. Figure 5-3 shows the format of this packet.

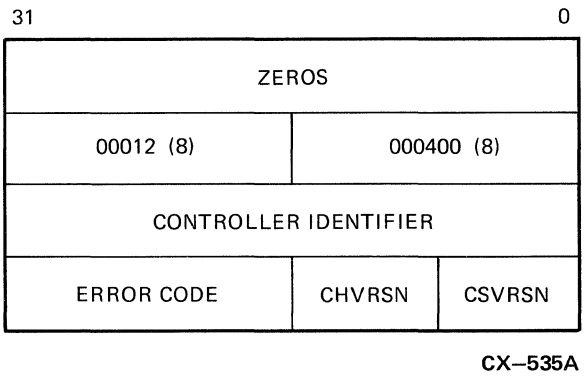


Figure 5-3 MSCP Last Fail Packet Format

5.3.1 QBUS Packet Read Error (ER.PRD)

A QBUS packet Read error occurs when the KDA5Ø-Q microcode times-out or detects a QBUS parity error while attempting to read the MSCP command packet from the host. The microcode has already read the command descriptor from the command ring, and should have a valid MSCP Packet pointer. The following are characteristics of this error:

NOTE

SA=1ØØØØ1 (octal) for this error.

- The UPROC microcode stream detects the error if it takes too long to retrieve any word within the packet or a QBUS parity error has occurred.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize command from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last-fail error packet if error logging is enabled.

5.3.1.1 Possible Causes - The following are possible causes for the QBUS Packet Read Error:

- KDA5Ø-Q processor module (M7164).
- Controller to host interface.
- Inconsistent descriptor in the command ring buffer.

5.3.2 QBUS Packet Write Error (ER.PWR)

A QBUS packet Write error occurs when the KDA5Ø-Q microcode times-out or detects a QBUS parity error while attempting to write an MSCP response packet to the host. The microcode has already read the command descriptor from the command ring, and should have a valid MSCP Packet pointer. The following are characteristics of this error:

NOTE

SA=1ØØØØ2 (octal) for this error.

- The UPROC microcode stream detects the error if it takes too long to write any word within the packet to the host.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize command from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.2.1 Possible Causes - The following are possible causes for the QBUS Packet Write Error:

- KDA5Ø-Q processor module (M7164)
- Controller to host interface.
- The KDA5Ø-Q has tried to write a packet to an illegal address in the host memory.

5.3.3 KDA5Ø-Q ROM And RAM Parity Error (ER.RRP)

A KDA5Ø-Q ROM and RAM parity error occurs when the CROM parity error and the RAM parity error signals are present at the input of the Micro Trap Circuit. (The Micro Trap Circuit is described in Chapter 3, Section 3.2.7). The Micro Trap circuit forces the UPROC or DPROC to an error handler in the KDA5Ø-Q microcode. The following are characteristics of this error:

NOTE

SA=1ØØØØ3 (octal) for this error.

- The error is detected by hardware (microtrap circuit).
- The hardware forces the microcode stream (UPROC or DPROC) to an error handler.
- If the error occurred when the DPROC was active, the DPROC saves the eventual SA error code value and informs the UPROC of a fatal controller error.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC microcode stream then goes into a wait loop until it receives an initialize command from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.3.1 Possible Causes - The following are possible causes for the KDA5Ø-Q ROM/RAM parity error:

- KDA5Ø-Q processor module (M7164) and the KDA5Ø-Q SDI module (M7165). The ROM resides on the processor module and the RAM resides on the SDI module.

5.3.4 KDA5Ø-Q ROM Parity Error (ER.ROP)

A KDA5Ø-Q ROM parity error occurs when the CROM parity error signal is present at the input of the Micro Trap Circuit. (The Micro Trap Circuit is described in Chapter 3, Section 3.2.7). The Micro Trap circuit forces the UPROC or DPROC to an error handler in the KDA5Ø-Q microcode. The following are characteristics of this error:

NOTE

SA=1ØØØØ5 (octal) for this error.

- The error is detected by hardware (microtrap circuit).
- The hardware forces the microcode stream (UPROC or DPROC) to an error handler.
- If the error occurred when the DPROC was active, the DPROC saves the eventual SA error code value and informs the UPROC of a fatal controller error.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC microcode stream then goes into a wait loop until it receives an initialize command from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.4.1 Possible Causes - The following are possible causes for the KDA5Ø-Q CROM Parity error:

- KDA5Ø-Q Processor Module (M7164)

5.3.5 KDA5Ø-Q RAM Parity Error (ER.RAP)

The KDA5Ø-Q RAM parity error occurs when the RAM parity error signal is present at the input of the Micro Trap Circuit. (The Micro Trap Circuit is described in Chapter 3, Section 3.2.7). The Micro Trap circuit forces the UPROC or DPROC to an error handler in the KDA5Ø-Q microcode. The following are characteristics of this error:

NOTE

SA=1ØØØØ4 (octal) for this error.

- The error is detected by hardware (microtrap circuit).
- The hardware forces the microcode stream (UPROC or DPROC) to an error handler.
- If the error occurred when the DPROC was active, the DPROC saves the eventual SA error code value and informs the UPROC of a fatal controller error.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC microcode stream then goes into a wait loop until it receives an initialize command from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.5.1 Possible Causes - The following are possible causes for the KDA5Ø-Q RAM Parity Error:

- The KDA5Ø-Q SDI module (M7165).

5.3.6 QBUS Ring Read Error (ER.RRD)

A QBUS ring Read error occurs when the KDA5Ø-Q microcode times-out or a QBUS parity error is observed while attempting to read the command descriptor from the host. (The command descriptor resides in the command and response ring buffer.) The following are characteristics of this error:

NOTE

SA=1ØØØØ6 (octal) for this error.

- The UPROC microcode stream detects this error if it takes too long to retrieve any word of the descriptor.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an init from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.6.1 Possible Causes - The following are possible causes for the QBUS Ring Read Error:

- KDA5Ø-Q processor module (M7164).
- Controller to host interface.
- A problem with the host ring buffer.

5.3.7 QBUS Ring Write Error (ER.RWR)

A QBUS ring Write error occurs when the KDA5Ø-Q microcode times out while attempting to update response or command descriptors in the host ring buffer. The descriptors are updated by a DMA Write from the KDA5Ø-Q. The descriptors are updated in the following cases:

NOTE

SA=1ØØØØ7 (octal) for this error.

- When the KDA5Ø-Q has finished reading the command packet, it changes the ownership bit so the host again owns the command descriptor.
- When the KDA5Ø-Q has a response message for the host, it updates a response descriptor by changing the ownership bit.

The following are characteristics of this error:

- The UPROC microcode stream detects this error if it takes too long to transfer the new descriptor value to the host ring buffer.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an init from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.7.1 Possible Causes - The following are possible causes for the QBUS ring Write error:

- KDA5Ø-Q processor module (M7164).
- Controller to host interface.
- A problem with the host ring buffer.

5.3.8 QBUS Interrupt Master Failure (ER.INT)

When the KDA5Ø-Q attempts to interrupt the host, there is a possibility it will never gain bus mastership. So, at the beginning of every interrupt sequence, the KDA5Ø-Q stores this error code in RAM in case the error occurs. If the error does occur, the KDA5Ø-Q waits for the bus until the host intervenes by initializing the KDA5Ø-Q. The host performs this initialization when its controller timeout value expires. The following are characteristics of this error:

NOTE

SA=1ØØØ1Ø (octal) for this error.

- The UPROC microcode stream writes this error code in RAM in case the error occurs.
- If the error occurs, the host initializes the KDA5Ø-Q when the hosts controller timeout value expires.
- The UPROC microcode stream creates a last fail error log packet if a last fail report was requested by the host during step four of initialization.
- The UPROC stream then goes into its idle loop, awaiting host interaction.

5.3.8.1 Possible Causes - The following are possible causes for the QBUS Interrupt Master Failure:

- The KDA5Ø-Q to host interface.
- The processor module (M7164).

5.3.9 Host Access Timeout Error (ER.HTO)

Normally, drives UNIT-ONLINE to a first controller are inaccessible to a second controller. This inaccessibility can be a problem if the communications path between a host and the first controller fails. The host access timeout, if enabled, eliminates this problem by causing the first controller to automatically release all drives if it doesn't receive a command within a specified host timeout interval (for the KDA5Ø-Q, this is one minute). Thus, the host access time-out error guarantees that dual port drives will be accessible from a second controller.

The UPROC microcode stream detects this error while in its idle loop. The error occurs if the KDA5Ø-Q has completed its last outstanding command and has been idle for a period greater than the host timeout value. The following are characteristics of the Host Timeout Error:

NOTE

SA=1ØØØ11 (octal) for this error.

- The error is detected when a timer, set up during the UPROC idle loop, times out.
- The KDA5Ø-Q initializes the on-line disk drives so they are available to a second controller.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize command from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.9.1 Possible Causes - The following are possible causes for the host timeout error:

- The controller to host interface has failed.
- The KDA5Ø-Q processor module (M7164)

5.3.1Ø Host Exceeded Command Limit (ER.NIM)
This error occurs if the KDA5Ø-Q determines a command is loaded in RAM log packet space and it is not an immediate command. The following are the characteristics of this error:

NOTE

SA=1ØØØ12 (octal) for this error.

- The UPROC microcode stream detects this error if it finds a non-immediate command in log packet space.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an init from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.1Ø.1 Possible Causes - The following are possible causes for the host exceeded command limit error:

- KDA5Ø-Q processor module (M7164).

5.3.11 QBUS Bus Master Failure (ER.MST)

This error occurs at the beginning of either a QBUS Read or a QBUS Write DMA transfer if the KDA5Ø-Q does not achieve bus mastership. The following are characteristics of this error:

NOTE

SA=1ØØØ13 (octal) for this error.

- The UPROC microcode stream detects this error if the KDA5Ø-Q fails to become bus master.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize command from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last-fail error packet if error logging is enabled.

5.3.11.1 Possible Causes - The following are possible causes for the QBUS bus-master failure:

- The controller to host interface.
- The KDA5Ø-Q processor module (M7164)

5.3.12 DM XFC Fatal Error (ER.DMX)

This error is possible when the KDA5Ø-Q is in Diagnostic Machine (DM) mode. DM mode is used to write drive-oriented diagnostics, and is down-line loaded from host memory into KDA5Ø-Q RAM space.

If a Read or a Write from or to QBUS memory is attempted and the buffer address where RAM is to be accessed is less than the beginning of the DM RAM memory space, the DM XFC Fatal Error occurs. The following are characteristics of this error:

NOTE

SA=1ØØØ14 (octal) for this error.

- The UPROC microcode stream detects this error during DM XFC 11 or 12.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last-fail error packet if error logging is enabled.

5.3.12.1 Possible Causes - The following are possible causes for the DM XFC Fatal Error:

- An invalid address was requested for the Read or Write.
- KDA5Ø-Q SDI module (M7165)

5.3.13 KDA5Ø-Q Hardware Error (ER.TMO)

This error can occur during a diagnostic check of the DPROC timeout circuit or during a Read or Write operation. During the diagnostic check, the error indicates the timeout circuit is not accurate. During the Read or Write operation, it indicates the DPROC sequencer on the SDI module has experienced an instruction timeout. The following are characteristics of this error:

NOTE

SA=1ØØØ15 (octal) for this error.

- The error is detected by the DPROC microcode stream if an instruction timeout occurs on the DPROC during a Read or Write operation or if the timeout circuit is inaccurate during diagnostics.
- The DPROC microcode stream displays 9 (HEX) in the LEDS on the KDA5Ø-Q SDI module.
- The DPROC microcode stream informs the UPROC microcode stream of the error and waits for action from the UPROC.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.13.1 Possible Causes - The following are possible causes for KDA5Ø-Q Hardware Error:

- KDA5Ø-Q processor module (M7164)

5.3.14 Invalid Virtual Circuit Identifier (ER.VCI)

This error occurs if an MSCP command is attempted when the virtual circuit is zero or if a DUP command is attempted when the virtual circuit is one. The following are characteristics of the Invalid Virtual Circuit Identifier error:

NOTE

SA=100016 (octal) for this error.

- The UPROC microcode stream detects this error when it is verifying an op code for an MSCP packet and when validating DUP commands.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize command from the host.
- The host detects bit 15 of the SA register, indicating the KDA50-Q has an error.
- The host initializes the KDA50-Q by writing the IP register.
- The KDA50-Q goes through the initialize sequence.
- The KDA50-Q logs a last fail error packet if error logging is enabled.

5.3.14.1 Possible Causes - The following are possible causes for the Invalid Virtual Circuit Identifier error:

- Controller to host interface.
- The KDA50-Q processor module (M7164)

5.3.15 Interrupt Write Error On QBUS (ER.IWR)

This error occurs when the KDA5Ø-Q tries to write a value to the response ring to set it not empty and a fatal QBUS error occurs. The Write is a one word DMA transfer to the host. The following are characteristics of this error:

NOTE

SA=1ØØØ17 (octal) for this error.

- The UPROC microcode stream detects this error if the DMA transfer fails.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last-fail error packet if error logging is enabled.

5.3.15.1 Possible Causes - The following are possible causes for the Interrupt Write Error on QBUS:

- The KDA5Ø-Q to host interface.
- KDA5Ø-Q processor module (M7164).

5.3.16 Too Many Subunits On KDA5Ø-Q (ER.SUN)

The KDA5Ø-Q does not implement this error.

5.3.17 Mapping Register Read Error (ER.MRR)

The KDA5Ø-Q generates this error when a fatal QBUS error (either timeout or parity) occurs while the KDA5Ø-Q tries to read the value of the mapping register via a DMA transfer from host memory. The following are characteristics of this error:

NOTE

SA=1ØØØ26 (octal) for this error.

- The UPROC microcode stream detects this error if a parity or timeout error occurs on the QBUS during the Read from host memory.
- The UPROC microcode stream loads the SA register with the error code and also sets bit 15 (error bit) to inform the host an error has occurred.
- The UPROC stream then goes into a wait loop until it receives an initialize from the host.
- The host detects bit 15 of the SA register, indicating the KDA5Ø-Q has an error.
- The host initializes the KDA5Ø-Q by writing the IP register.
- The KDA5Ø-Q goes through the initialize sequence.
- The KDA5Ø-Q logs a last fail error packet if error logging is enabled.

5.3.17.1 Possible Causes - The following are possible causes for the mapping register Read error:

- KDA5Ø-Q processor module (M7164).
- Controller to host interface.
- A problem with the host ring buffer.

5.4 ERROR LOG FORMAT Ø (CONTROLLER ERROR)

The following paragraphs describe the errors reported via the controller error log packet. In addition, figure 5-4 shows the format for a controller error log packet. These entries further define the Controller Error entry in the status/event code field of the error log packet.

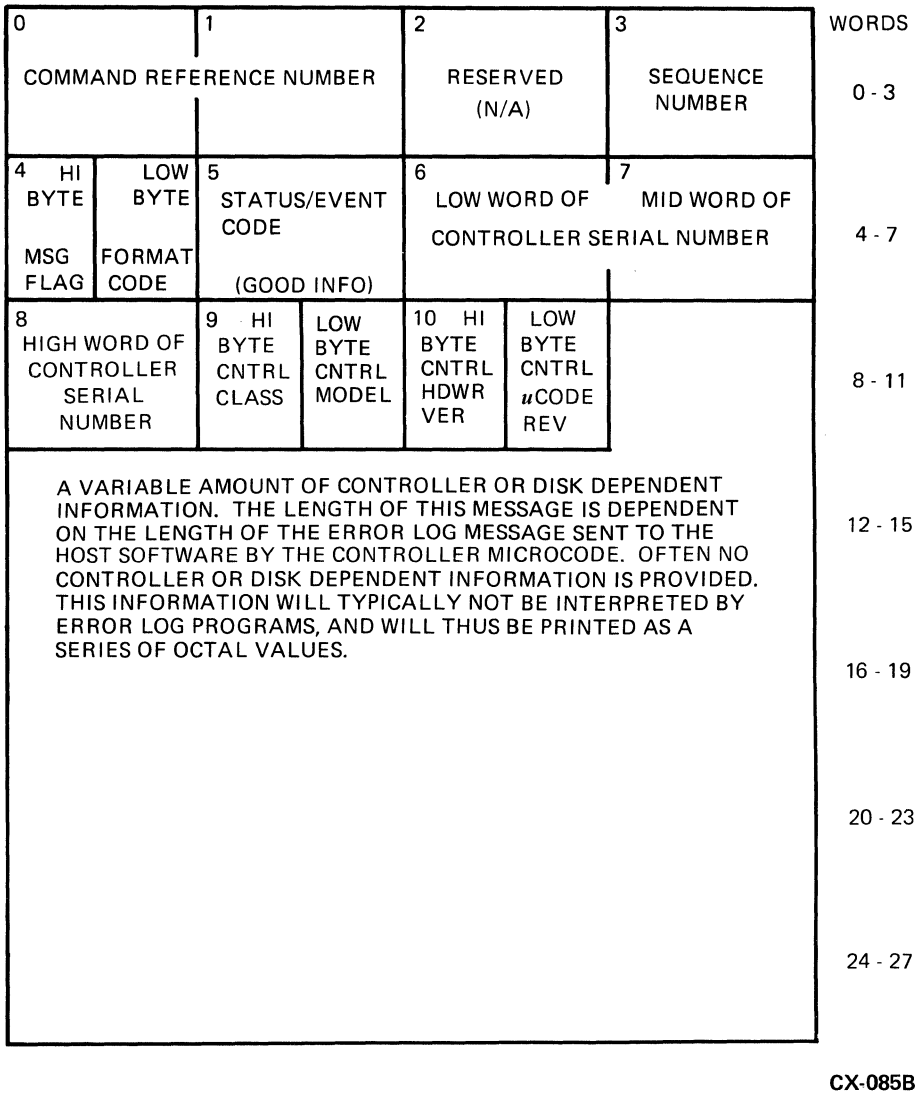


Figure 5-4 MSCP Controller Error Log Packet Format

5.4.1 SERDES Overrun Error (SC.OVR)

This error occurs at the end of a sector transfer when the target drive has a transfer rate in excess of the KDA5Ø-Q limit or there has been a hardware failure internal to the KDA5Ø-Q. The following are characteristics of this error:

NOTE

Status/event code=52 (octal) for this error.

- The SERDES 16 detects this error and flags it via the SERDES O/UR bit.
- If the O/UR bit is set, indicating an error, the DPROC saves the SERDES overrun error code and informs the UPROC of the error.
- The DPROC issues a MASTER SERDES RESET command.
- The DPROC issues a SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated.
- End and log error packets are sent to the host.

5.4.1.1 Possible Causes - The following are possible causes for the SERDES Overrun Error:

- The KDA5Ø-Q SDI module. (M7165)

5.4.2 EDC Error (SC.EDC)

The error detection code, attached to every disk sector, is revalidated when data transfers to host memory are performed. An EDC error detected at this time indicates a hardware failure in the data path of the KDA5Ø-Q. The following are initial conditions for this error:

NOTE

Status/event code=112 (octal) for this error.

- Data has been read from the disk and Read Gate is negated.
- The DPROC verifies the ECC for this sector using the R-S Gen.
- The DPROC saves the ECC residue in the event of an error.
- The DPROC notifies the UPROC a sector of data has been read.
- The UPROC checks for ECC errors and performs ECC correction if necessary.

The following are characteristics for this error:

- The UPROC transfers the data to host memory and does the following:
  - Recalculates the EDC value for this data.
  - Compares the recalculated EDC with the original EDC.
- The UPROC detects this error if the EDC values are not the same.
- The number of retries for this error is drive dependent.
- If the retries are not successful, the UPROC sends an MSCP log packet.

5.4.2.1 Possible Causes - The following are possible causes for the KDA5Ø-Q EDC error:

- The KDA5Ø-Q SDI module (M7165).

5.4.3 Inconsistent Controller State (SC.CNT)

NOTE

This error is not implemented on the KDA5Ø-Q.

5.5 ERROR LOG FORMAT 1 (HOST ACCESS ERROR)

These entries are all KDA5Ø-Q/host communication problems. Generally, they are host bus problems such as a QBUS parity error or an odd transfer address. The following paragraphs describe the errors contained within Error Log format 1. Figure 5-5 shows the format for a Host Access Error Log Packet.

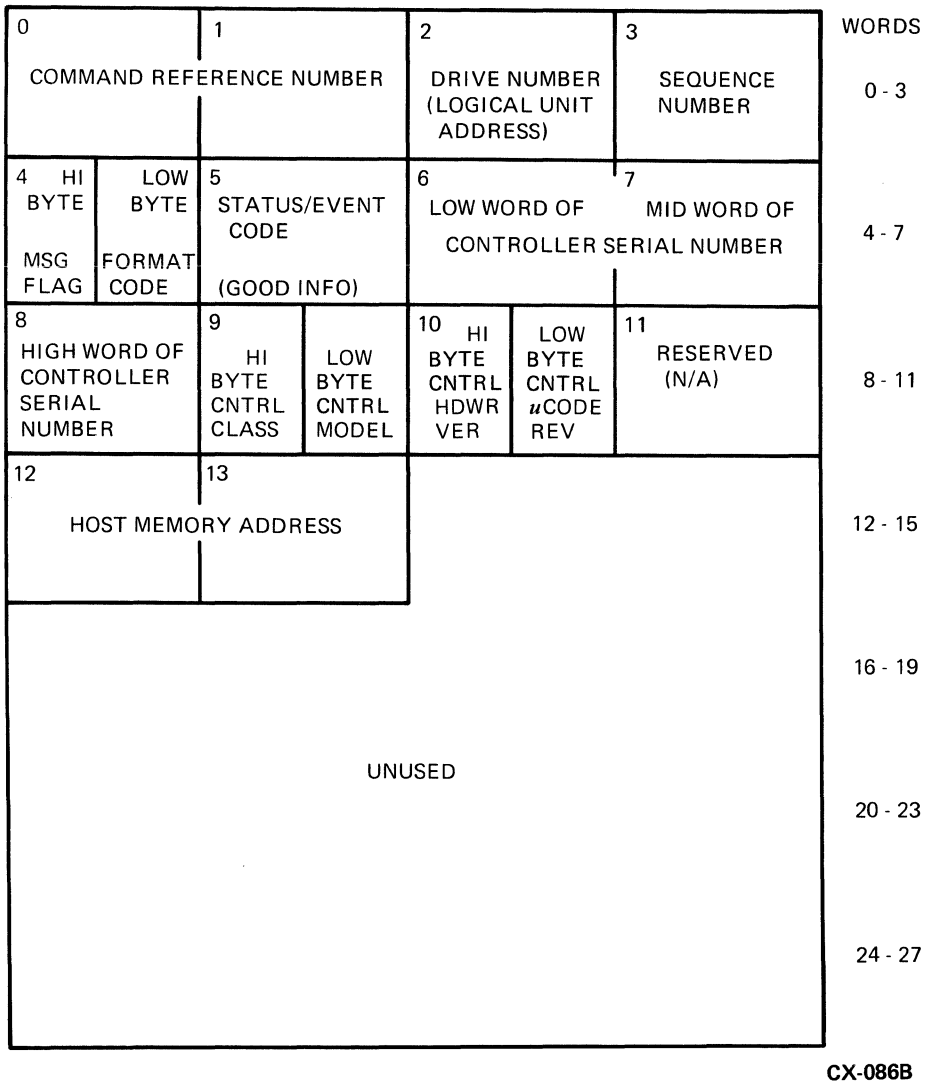


Figure 5-5 MSCP Host Memory Access Error Log Packet Format

5.5.1 QBUS Error Subcode Values

The following paragraphs describe the QBUS Error Subcode Values. These values further define the Host Buffer Access Error entry in the status/event code field of the error log packet.

5.5.2 Odd Transfer Address (SC.ODT)

This error occurs when the buffer address supplied by the host buffer descriptor within the MSCP command is odd. Because the KDA5Ø-Q transfers words only, an odd buffer address is illegal. The following are characteristics of this error:

NOTE

Status/event code=51 (octal) for this error.

- The UPROC microcode stream detects this error by testing the LSB of the buffer address. If the LSB is set, indicating an odd address, the UPROC microcode stream saves the odd transfer address error code in RAM.
- The UPROC microcode stream creates an error log packet and stores this packet in RAM.
- The UPROC microcode stream queues the error log packet to the host.

5.5.2.1 Possible Causes - The following are possible causes for the odd transfer address error:

- The KDA5Ø-Q to host interface.
- KDA5Ø-Q processor module (M7164)

5.5.3 Odd Byte Count (SC.ODB)

This error can occur during data transfers from a disk drive to the KDA5Ø-Q. After a sector is transferred, the KDA5Ø-Q checks the byte count and asserts this error if the byte count is odd. An odd byte count is illegal because the KDA5Ø-Q supports only the 512-byte format. Also, an odd byte count would require an odd transfer address. The following are characteristics of this error:

NOTE

Status/event code=111 (octal) for this error.

- After the sector has been transferred, the UPROC microcode stream detects this error by testing the LSB of the byte count field of the MSCP command packet. If the LSB is set, indicating an odd byte transfer is requested, the UPROC microcode stream saves the odd byte count error code in RAM.
- The UPROC microcode stream creates an error log packet and stores this packet in RAM.
- The UPROC microcode stream queues the error log packet to the host.

5.5.3.1 Possible Causes - The following are possible causes for the odd byte count error:

- Host to KDA5Ø-Q interface.
- KDA5Ø-Q processor module (M7164).

5.5.4 QBUS Nonexistent Memory Error (SC.NOM)

This error occurs if the KDA5Ø-Q times-out while attempting to transfer data between the KDA5Ø-Q and the host. If the transfer involves a host memory location that does not exist or is illegal, the transfer does not complete causing the timeout. The following are characteristics of the QBUS Nonexistent Memory Error:

NOTE

Status/event code=151 (octal) for this error.

- The UPROC microcode stream detects this error when the data transfer times out.
- The UPROC microcode stream sets up a flag so the microcode error handler can tell the difference between this error and the QBUS parity error.
- The UPROC microcode stream creates an error log packet and stores this packet in RAM.
- The UPROC microcode stream queues the error log packet to the host.

5.5.4.1 Possible Causes - The following are possible causes for the QBUS nonexistent memory error:

- The KDA5Ø-Q to host interface.
- The KDA5Ø-Q processor module (M7164).

5.5.5 QBUS Parity Error (SC.PAR)

This error occurs if the KDA5Ø-Q determines there is a QBUS parity error during a DATI operation from host memory. The KDA5Ø-Q checks the parity lines (BDAL 17:16) at the end of every Read and asserts this error if they are both set. (The operation of BDAL 17:16 is described in Chapter 2, Table 2-2.) The following are characteristics of this error:

NOTE

Status/event code=211 (octal) for this error.

- The UPROC microcode stream detects this error at the completion of a Read operation by testing BDAL 17:16 via the UPROC test multiplexer.
- The UPROC sets up a flag so the microcode error handler can tell the difference between this error and a nonexistent memory error.
- The UPROC microcode stream creates an error log packet and stores this packet in RAM.
- The UPROC microcode stream queues the error log packet to the host.

5.5.5.1 Possible Causes - The following are possible causes for the QBUS parity error:

- KDA5Ø-Q to host interface.
- KDA5Ø-Q processor module (M7164).

5.6 ERROR LOG FORMAT 2 (DISK TRANSFER ERROR)

These entries all indicate problems in transferring data between the KDA5Ø-Q and the disk drives. The errors generally indicate problems in finding the data or problems with the data, itself. The following paragraphs describe the errors contained within this error log format. Figure 5-6 shows the format for a Disk transfer error Log Packet.

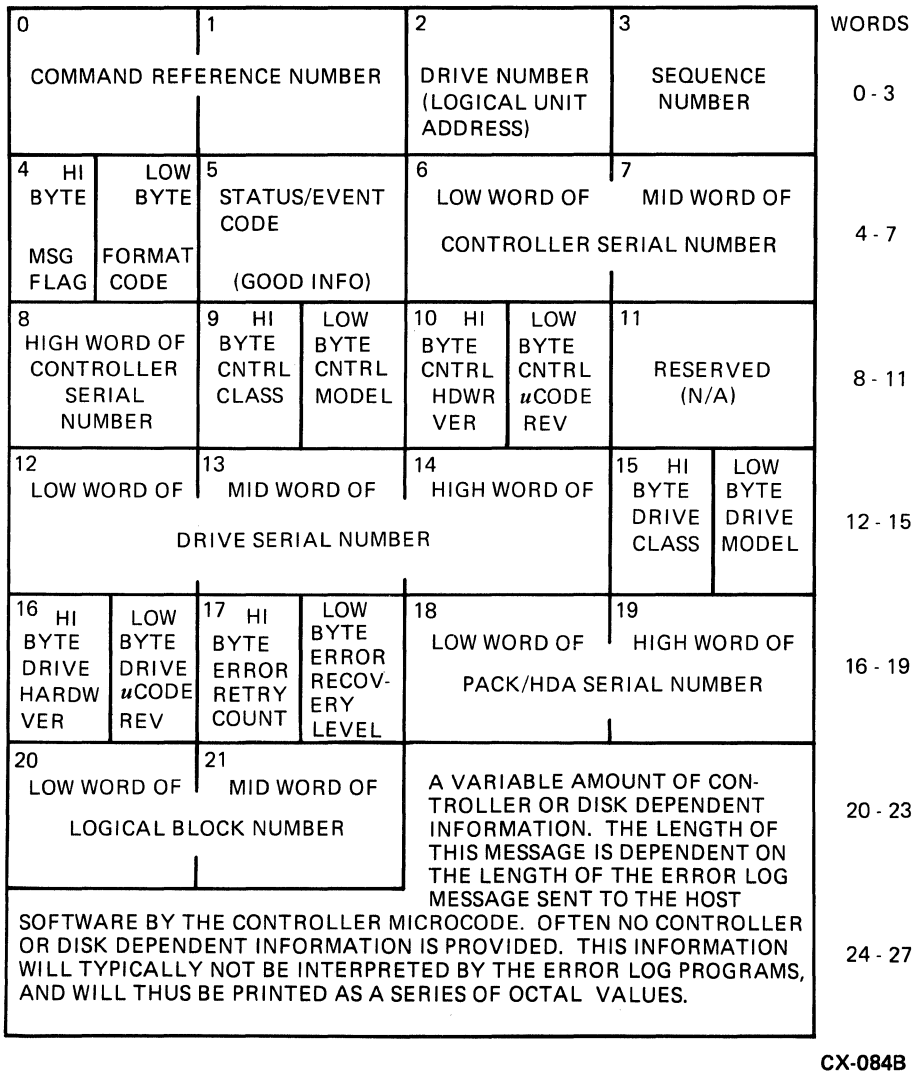


Figure 5-6 MSCP Disk Transfer Error Log Packet Format

5.6.1 Data Error Subcode Values

The following paragraphs describe the Data Error Subcode Values. These values further define the Data Error entry in the status/event code field of the error log packet.

5.6.2 Forced Error (SC.FER)

The forced error indicator is asserted in a replacement block when the data it contains is bad. The indicator is the one's complement of the calculated EDC for the sector. It is written during the bad block replacement (BBR) process if the replaced data could not be recovered. However, the block itself contains no errors. The following are characteristics of the forced error:

NOTE

Status/event code=1Ø (octal) for this error.

- The UPROC transfers the data block to the host and calculates the EDC.
- The UPROC microcode stream detects the forced error condition after finding the calculated EDC is the one's complement of the actual EDC from the disk drive.
- There are no retries for this condition.
- The UPROC creates an end packet which contains the forced error status (forced errors are not reported in the error log.)
- The forced error indicator remains in the sector until the sector is written with new data.

5.6.2.1 Possible Resolutions - The following are possible resolutions to a forced error condition:

- Perform a back-up operation on the files that contain the forced error.

5.6.3 Header Compare Error (SC.HDR)

This error occurs when the KDA5Ø-Q header-compare algorithm fails to find the desired header on the disk. The following are characteristics of the header compare error:

NOTE

Status/event code=11Ø (octal) for this error.

- The DPROC microcode stream detects this error when a header cannot be matched with the desired value.
- The DPROC searches the RCT to determine if the block has been revectored. If it has, the operation is performed on the revectored block and no error is reported. If not, the DPROC saves the header compare error code.
- The DPROC sets a bit in a status buffer, indicating a data buffer is full and the UPROC may now transfer it to the host. A full buffer is indicated as part of the DPROC to UPROC communication process.
- The DPROC informs the UPROC an error has occurred.
- The UPROC creates and sends an error log message to the host.

5.6.3.1 Possible Causes - The following are possible causes for the header compare error:

- A disk drive error.
- The SDI interface.
- The KDA5Ø-Q SDI module (M7165).

5.6.4 Data SYNC Timeout (SC.DSY)

This error occurs when the SERDES 16 does not detect the SYNC character (26BC Hex) that immediately precedes the actual data from the disk drive. The KDA5Ø-Q has already read a valid header and is awaiting the SYNC character. The following are characteristics of this error:

NOTE

Status/event code=15Ø (octal) for this error.

- The DPROC microcode stream detects this error while attempting to read data from a disk sector to a data buffer. If the SERDES 16 does not generate word rate clock (described in the Appendix, Table A-1), the DPROC microcode stream times-out and goes to an error handler.
- The DPROC sets a bit, indicating data-sync-not-found, in a status buffer that the UPROC later reads.
- The DPROC sets a bit in a status buffer, indicating a data buffer is full and the UPROC may now transfer it to the host. A full buffer is indicated as part of the DPROC to UPROC communication process.
- When the UPROC realizes the RAM data buffer is full via the buffer-full bit from the DPROC, it starts to transfer that data to the host. However, before the actual data is transferred, the UPROC checks the status buffer to see if any errors occurred.
- In this case, the UPROC finds data-sync-not-found asserted in the status buffer from the DPROC.
- The UPROC attempts to re-read the sector a drive-specified number of times.
- If the retry attempts fail, the UPROC creates and sends an error log messageto the host.

5.6.4.1 Possible Causes - The following are possible causes for the data sync timeout error:

- Hardware on the KDA5Ø-Q SDI module (M7165).
- A bad location on the disk.
- SDI interface.

5.6.5 ECC Errors

The following error description covers all of the ECC error types listed below:

- ECC Error uncorrectable (SC.ECC). The Status/event code=350 (octal) for this error.
- One Symbol ECC Error (SC.EC1). The Status/event code=410 (octal) for this error.
- Two Symbol ECC Error (SC.EC2). The Status/event code=450 (octal) for this error.
- Three Symbol ECC Error (SC.EC3). The Status/event code=510 (octal) for this error.
- Four Symbol ECC Error (SC.EC4). The Status/event code=550 (octal) for this error.
- Five Symbol ECC Error (SC.EC5). The Status/event code=610 (octal) for this error.
- Six Symbol ECC Error (SC.EC6). The Status/event code=650 (octal) for this error.
- Seven Symbol ECC Error (SC.EC7). The Status/event code=710 (octal) for this error.
- Eight Symbol ECC Error (SC.EC8). The Status/event code=750 (octal) for this error.

ECC errors occur when the data read from the disk does not agree with the data that was written. When data is written to the disk, an ECC is calculated (by the R-S Gen) and appended to the end of the sector. When the data is subsequently read from the sector, the ECC is revalidated. In general, there are two possible results:

1. The data error falls within the error correction capability of the ECC and data correction is performed. In this case, no error is reported and no data errors are experienced.
2. The data error does not fall within the error correction capability of the ECC and the error is retried according to drive dependent parameters. Each time the ECC error is retried, an error log packet is generated. If all of the retries fail, a bad block is reported via an end packet.

The following are characteristics of the ECC errors:

- When reading data from the disk, the R-S GEN (described in Chapter 3, Section 3.3.7) acts as a comparator. If the Read ECC from the disk does not agree with the newly calculated value, the R-S GEN asserts an error signal.

- The DPROC tests the ECC error bit from the R-S GEN and, if asserted, saves the ECC residue in the RAM data buffer.

- The DPROC informs the UPROC a buffer of data is available for transfer to the host, but contains an ECC error. At this point, the severity of the error has not been determined.

- The UPROC performs ECC correction on the buffer data.

- If the correction succeeds, the operation is completed and no error is reported.

- If the correction fails, the KDA5Ø-Q performs retries based on the drive's recommended retry count and the number of error recovery levels the drive supports. These parameters are implemented in the following ways:

  - The KDA5Ø-Q attempts retries up to two times the number given in the drive's recommended retry count.

  - The KDA5Ø-Q repeats the above retries for each recovery level the drive supports. For example, if the drive had a retry count of five and supported a recovery level of two, the KDA5Ø-Q would retry the operation up to 3Ø times (levels zero, one and two) before giving up.

- The UPROC reports an Unrecoverable ECC error in the error log if the retry attempts do not succeed.

5.6.5.1 Possible Causes - The following are possible causes for ECC errors:

- Disk drive error.
- KDA5Ø-Q SDI module.
- SDI interface.

5.7 ERROR LOG FORMAT 3 (SDI ERROR)

These entries indicate most drive problems. The following paragraphs describe the errors contained within Error Log Format three. Figure 5-7 shows the format for an SDI Error Log Packet.

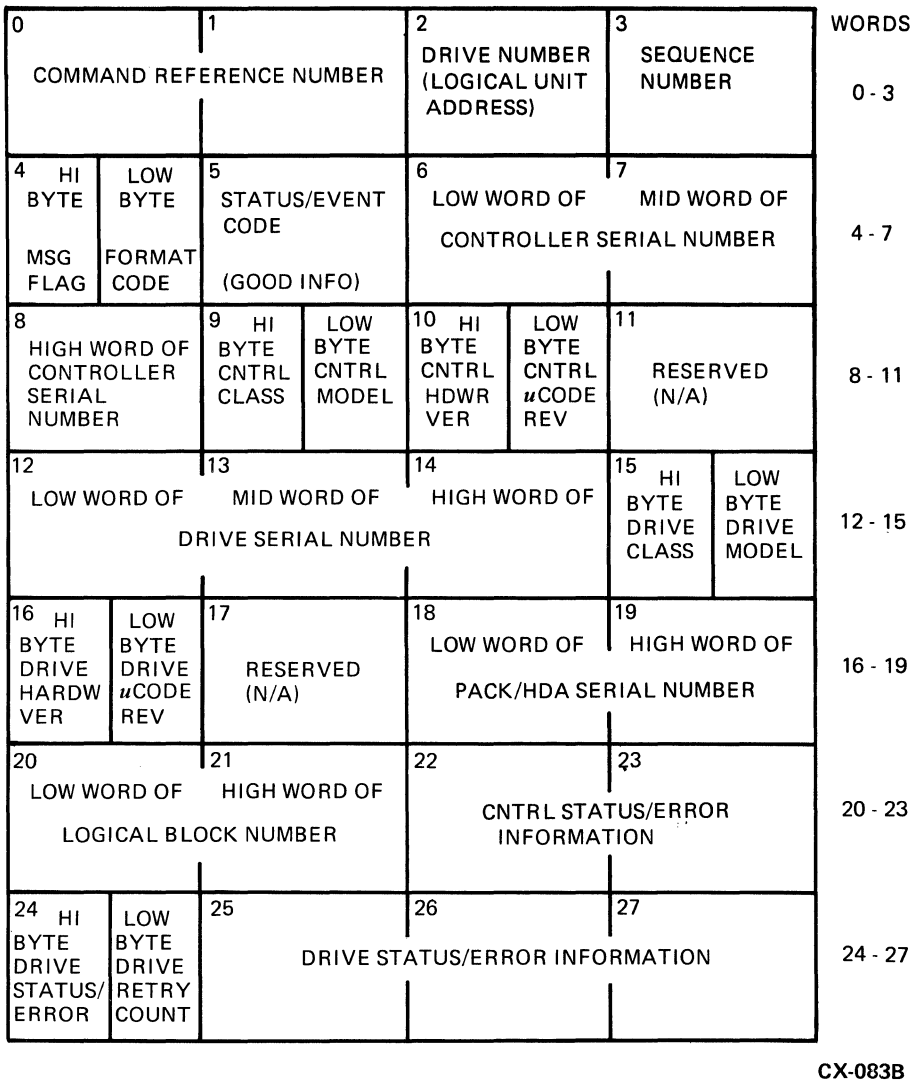


Figure 5-7 MSCP SDI Error Log Packet Format

5.7.1 Drive Error Subcode Values

The following paragraphs describe the Drive Error Subcode Values. These values further define the Drive Error entry in the status/event code field of the error log packet.

5.7.2 SDI Response Timeout (SC.STO)

This error occurs when the KDA5Ø-Q times-out while waiting for a response from the drive to a REAL-TIME command. The following are characteristics of the SDI Response Timeout Error:

NOTE

Status/event code=53 (octal) for this error.

- The DPROC attempts a REAL-TIME command.
- The DPROC tests for a response from the drive and if one is not received within a time-out value, the SDI response timeout error code is saved.
- The DPROC issues an SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated and the drive is marked offline.
- End and log error packets are sent to the host.

5.7.2.1 Possible Causes - The following are possible causes for the SDI response timeout error:

- A drive error.
- SDI interface.
- KDA5Ø-Q SDI module (M7165).

5.7.3 Invalid SDI Response (SC.INV)

The Invalid SDI Response error is reported if the following conditions are met:

- An invalid response from the drive.
- An inappropriate response to the command sent.
- A response with an invalid length.

The following are characteristics of the Invalid SDI Response error:

NOTE

Status/event code=113 (octal) for this error.

- The DPROC tests for a response from the drive and sets this error if the response message meets the criteria listed at the beginning of Section 5.7.3.
- The DPROC issues an SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated.
- End and log error packets are sent to the host.

5.7.3.1 Possible Causes - The following are possible causes for the invalid SDI response error:

- A drive error.
- SDI interface.
- KDA5Ø-Q SDI module (M7165).

5.7.4 Positioner Error (SC.POS)

This error is reported when the headers read from the drive indicate the drive has positioned to the wrong cylinder without indicating a Seek error. The following are characteristics of this error:

NOTE

Status/event code=153 (octal) for this error.

- A positioning command has resulted in no header match.
- The DPROC analyzes the header compare problem and determines there are no headers on this track that match the desired header. Therefore, the drive has a positioning error.
- The DPROC issues an SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated.
- End and log error packets are sent to the host.

5.7.4.1 Possible Causes - The following are possible causes for the positioner error:

- A drive error.
- SDI interface.
- KDA5Ø-Q SDI module (M7165).

5.7.5 Lost Read/Write Ready (SC.RWR)

This error occurs when drive Read/Write Ready is negated during a sector transfer. The following are characteristics of the Lost Read/Write Ready error:

NOTE

Status/event code=213 (octal) for this error.

- When looking for the header of the sector to transfer, the DPROC cannot match a header with the desired value.
- The DPROC analyzes the header compare problem and determines there are no errors and keeps looking for a valid header. It does a routine check of the RTDS bits and determines the disk drive has negated Read/Write Ready, possibly indicating the heads have gone off track.
- The DPROC issues an SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated.
- End and log error packets are sent to the host.

5.7.5.1 Possible Causes - The following are possible causes for the Lost Read/Write error:

- A drive error.
- SDI transmission problems.
- KDA5Ø-Q SDI module (M7165).

5.7.6 Lost Drive Clock Operation (SC.DCL)

Lost Drive Clock Operation occurs if the KDA5Ø-Q detects a Drive Clock failure during data transfer attempts. The drive clock check is performed by monitoring parallel data ready from the SERDES (refer to Appendix Table A-1). If the SERDES does not assert parallel data ready within a timeout value, it is assumed the drive clock has stopped operation. The timeout value is determined by the timeout portion of the microtrap circuit described in Chapter 3, Section 3.2.7. The following are characteristics of this error:

NOTE

Status/event code=253 (octal) for this error.

- The KDA5Ø-Q detects this error after it is synchronized to the disk drive word rate.
- The error is detected by the DPROC microcode stream when a timeout occurs while waiting for parallel data ready from the SERDES.
- The DPROC issues an SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A Seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated.
- End and log error packets are sent to the host.

5.7.6.1 Possible Causes - The following are possible causes for the Lost Drive Clock Operation error:

- A drive error.
- SDI interface.
- KDA5Ø-Q SDI module (M7165).

5.7.7 Lost Drive Receiver Ready (SC.RRD)

This error is detected when the KDA5Ø-Q attempts to transmit a real-time Read or Write command immediately before the target sector, and the RTDS indicates the drive has negated receiver ready. The following are characteristics of this error:

NOTE

Status/event code=313 (octal) for this error.

- The DPROC is prepared for a data transfer to or from the disk drive.
- The DPROC tests for receiver ready before issuing a REAL-TIME command. If receiver ready from the drive does not assert within a timeout value, this error code is saved.
- The DPROC issues an SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A Seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated.
- End and log error packets are sent to the host.

5.7.7.1 Possible Causes - The following are possible causes for the Lost Drive Receiver Ready error:

- A drive error.
- SDI transmission error.
- KDA5Ø-Q SDI module (M7165).

5.7.8 Drive Detected Error (SC.DDE)

The Drive Detected Error occurs when a drive reports an attention condition that is not related to an outstanding MSCP command. The following are characteristics of this error:

NOTE

Status/event code=353 (octal) for this error.

- The drive sends attention to the KDA5Ø-Q to report a change in its condition.
- The DPROC receives attention and saves the drive detected error code.
- The DPROC issues an SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated.
- End and log error packets are sent to the host.

5.7.8.1 Possible Causes - The following are possible causes for the Drive Detected Error:

- Drive Error.
- SDI transmission error.
- KDA5Ø-Q SDI module (M7165)

5.7.9 RTDS Pulse Or Data Parity Error (SC.LVØ)

This error is detected at the end of a sector transfer and indicates the loss of RTDS during a Read or Write operation. The following are characteristics of this error:

NOTE

Status/event code=413 for this error.

- The DPROC senses the loss of RTDS and saves this error code.
- The DPROC issues an SDI INITIALIZE command.
- The DPROC notifies the UPROC of the error condition.
- The UPROC performs the following retry operations:
  - An SDI GET STATUS command is issued.
  - An SDI DRIVE CLEAR is issued.
  - A seek to the previous cylinder is issued.
- These retry operations are performed two times from the point of the failing LBN.
- If the retries do not succeed, the MSCP request is terminated.
- End and log error packets are sent to the host.

5.7.9.1 Possible Causes - The following are possible causes for the RTDS pulse or data parity error:

- Drive error.
- SDI transmission failure.
- The KDA5Ø-Q SDI module (M7165).

APPENDIX A

SELECTED CHIP DESCRIPTIONS AND STATUS/EVENT CODES

A.1 SERDES 1Ø & 16

The Serializer-Deserializer (SERDES) chip is a special chip designed by Digital Equipment Corporation to perform data conversion between serial and parallel data formats. The data conversion may occur at a maximum frequency of approximately 28 MHZ.

The SERDES contains an automatic synchronization feature that allows it to synchronize a serial bit stream without any external control. Using an auto correlation technique, the SERDES detects a 12-bit pattern and uses this pattern as a synchronization point. After this point is detected, the SERDES begins the serial to parallel conversion. Three possible parallel word lengths are 8-bit, 1Ø-bit (SERDES 1Ø), and 16-bit (SERDES 16).

The SERDES has one parallel output port and two serial output ports. The parallel port and one of the serial ports are tri-state driven. These tri-state ports have separate external enable signals. The remaining serial port is a standard TTL totem-pole output.

Table A-1 provides a functional description of the pins on the SERDES. Figure A-1 shows the pin layout of the SERDES and figure A-2 is a functional block diagram of the SERDES.

Table A-1 SERDES Signal Definitions And Functions

Pin #	Pin Name	Input/ Output	Signal Name/Function
1	TTL G	----	TTL Ground. Pins 1 and 2Ø must be connected using the shortest possible path.

For Internal Use Only

Table 1 (cont.)

Pin #	Pin Name	Input/ Output	Signal Name/Function
2	ACD L	Output	AUTO Correlator. The SERDES asserts this signal when the sync byte is detected. This signal is for test purposes only and should not be used.
3	SCI H	Input	Serial Clock In. The positive edge of this signal shifts data into the shift register.
4	O/UR H	Output	Over/Under Run. This signal is cleared by Master Reset. It is asserted when a parallel transfer on the parallel port is missed. The missed parallel transfer occurs in one of the following ways: 1. In parallel out mode (POM asserted), the SERDES asserts PRDY when the parallel output data is ready. If a corresponding DR (data recieved) signal is not returned within the word boundary, O/UR is asserted, indicating an overrun condition. 2. In parallel in mode (POM negated), the SERDES asserts PRDY when it is ready to be parallel loaded. If a corresponding PLS (parallel input strobe) is not received within the word boundary, O/UR is asserted, indicating an underrun condition.

For Internal Use Only

Table 1 (cont.)

Pin #	Pin Name	Input/ Output	Signal Name/Function
5	PRDY H	Output	Parallel Ready. NOTE: The following assumes the tristate ports are enabled (POTSE asserted): PRDY helps initiate and complete asynchronous handshaking routines between the parallel port on the SERDES and a corresponding parallel device. In parallel out mode (POM asserted), the SERDES asserts POM when the parallel output word is stable on the I/O pins. However, if the sync character is present on the I/O lines, PRDY is not asserted. In parallel in mode (POM negated), the SERDES asserts PRDY when it is ready to receive a parallel word. PRDY negates in one of the following ways: 1. If POM is asserted, a negative to positive transition on DR (data received) negates PRDY. 2. If POM is negated, a positive to negative transition PLS (parallel input strobe) negates PRDY. 3. If neither 1 or 2 above occurs, the SERDES itself drops PRDY just before the parallel data becomes invalid and latches an O/UR.
6	DR H	Input	Data Received. In Parallel out mode (POM asserted), the SERDES receives this signal from the corresponding device, indicating the device has received the data. This is part of the asynchronous parallel handshake. The handshake sequence is as follows: The SERDES asserts PRDY when the parallel output data is stable. After the user has taken the data, it asserts DR H. Upon receiving DR H, the SERDES negates PRDY.

For Internal Use Only

Table 1 (cont.)

Pin #	Pin Name	Input/ Output	Signal Name/Function
7-14 and 27-34	I/O 15:ØØ	Input/ Output	Parallel Data I/O. These lines contain parallel data and their I/O function is determined by the state of POM. Their output is enabled by the POTSE (parallel output tristate enable) signal.
15	SDI1 H	Input	Serial Data In 1. This is serial input data and is clocked into the shift register if SDIC is asserted or high.
16	SDIØ H	Input	Serial Data In Ø. This is serial input data and is clocked into the shift register if SDIC is negated or low.
17	SDIC H	Input	Serial Data Input Control. This signal determines which serial input is shifted into the shift register. If it is low, SDIØ data is moved into the shift register. If it is high, SDI1 data is moved into the shift register.
18	POTSE L	Input	Parallel Out Tristate Enable. If negated, this signal disables all the parallel outputs (I/O 15:ØØ).
19	CE H	Input	Counter Enable. With POM negated, CE enables the internal bit rate counter on the next positive edge of SCI. With POM asserted, CE permits the detection of the sync pattern in the SDI stream to start the bit rate counter.

The sync pattern is as follows:

Preamble Ø's	sync pattern	Data
ØØØØØØØØØØ	111101011001100	XXXXXX

For Internal Use Only

Table 1 (cont.)

Pin #	Pin Name	Input/ Output	Signal Name/Function
2Ø	GND	----	Ground. Pins 1 and 2Ø must be connected by the shortest possible path.
21 22	WLSØ WLS1	Input Input	Word Length Select Ø and 1. These two select lines determine the length of the parallel word in or out of the SERDES. The following describes the possible parallel word lengths:

	16-bit	1Ø-bit	8-bit
WLSØ	H	H	L
WLS1	H	L	H

23	MR L	Input	Master Reset. This signal resets all the internal registers and counters.
24	I/O GND	----	I/O Ground. Isolated ground for the I/O port output buffers (I/O 15:ØØ).
25	POM H	Input	Parallel Out Mode. When asserted, POM causes the SERDES inputs and outputs to be as follows: serial in, serial out and parallel out. When negated, POM causes the SERDES inputs and outputs to be as follows: parallel in, serial out.
26	PLS L	Input	Parallel Input Strobe. When the SERDES is in parallel in mode (POM negated), PLS completes the parallel handshake. For example: The SERDES asserts PRDY, indicating it is ready to be parallel loaded. After the sending device has stable data on the I/O lines, it asserts PLS. This loads the SERDES registers and causes the SERDES to negate PRDY.
35	SOTSE L	Input	Serial Out Tristate Enable. If negated, this signal disables the two serial tristate outputs: TSSDO and TSSCO.

Table 1 (cont.)

Pin #	Pin Name	Input/ Output	Signal Name/Function
36	WRC H	Output	Word Rate Clock. WRC is intended to be useful in implementing the parallel port. It is decoded from the Modulo-N (N=8, N=1Ø, N=16) counter. It has a 5Ø% duty cycle and a period of N bit times. It is always asserted when the counter is equal to zero and is negated after N/2 bit times.
37	SDO H	Output	Serial Data Out. SDO has the same delay characteristics from SDI (Ø or 1) and I/O 15:ØØ as described in TSSDO. SDO is not a tristate output and is intended for serially cascading two SERDES running on the same Serial Clock In (SCI).
38	TSSDO H	Output	Tristate Serial Data Out. TSSDO is enabled by SOTSE (Serial Out Tristate Enable). Data transitions on this line are carefully matched in time to positive edges of TSSCØ (Tristate Serial Clock Out). If POM is asserted, the delay from SDIØ or SDI1 to TSSDO varies with the word length selected via WLSØ and WLS1. For a word length of 16 bits, the delay is 37-bit times; a word length of 1Ø bits, the delay is 25-bit times; a word length of 8 bits, the delay is 21-bit times. If POM is negated and PLS has been asserted (loading data into the internal register), then the serial output data will be available at the serial output ports exactly 4.5 bit times after PRDY becomes asserted. From then on, serial output data will always be 4.5 bit times delayed from PRDY.

Table 1 (cont.)

Pin #	Pin Name	Input/ Output	Signal Name/Function

NOTES FOR TSSDO H

1. With POM negated, the serial output port is not under SERDES control until data is loaded through the parallel port and CE is asserted. If the SERDES is left in idle mode while POM is negated, any data found at SDIØ or SDI1 will be shifted to SDO and TSSDO.

2. With POM asserted, the SERDES will be in search mode until it finds the sync byte. In the 16-bit and 1Ø-bit modes, the SERDES shifts out the complete sync byte and any data that follows it. In the 8-bit mode, however, the sync byte is truncated by one bit. This causes the serial data train coming from TSSDO and SDO to be one bit short. However, the data that follows the sync byte is not affected.

39	TSSCO H	Output	Tristate Serial Clock Out. This output is a delayed version of SCI. It is delayed to match the propagation delays of TSSDO, and therefore provides a positive-going edge very precisely matched to the transistion time of the TSSDO data bits. It is a tristate signal and is controlled by Serial Output Tristate Enable (SOTSE).
4Ø	VCC	----	Power Supply. This is the +5 Volts supply.

For Internal Use Only

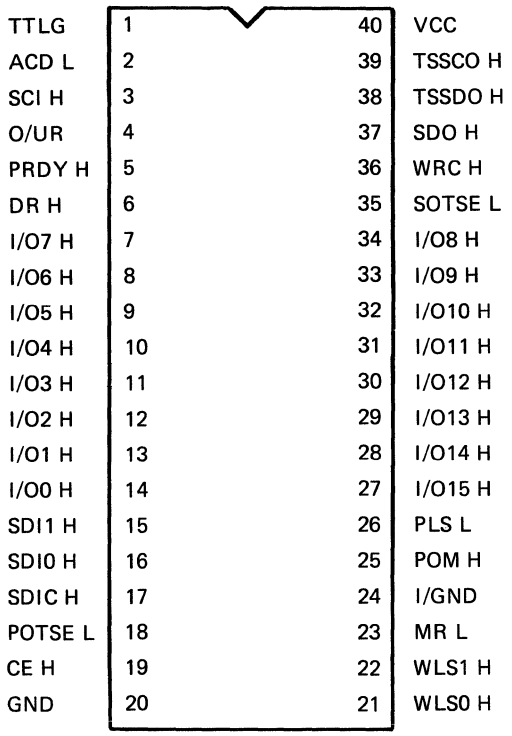


Figure A-1 SERDES Pin Layout

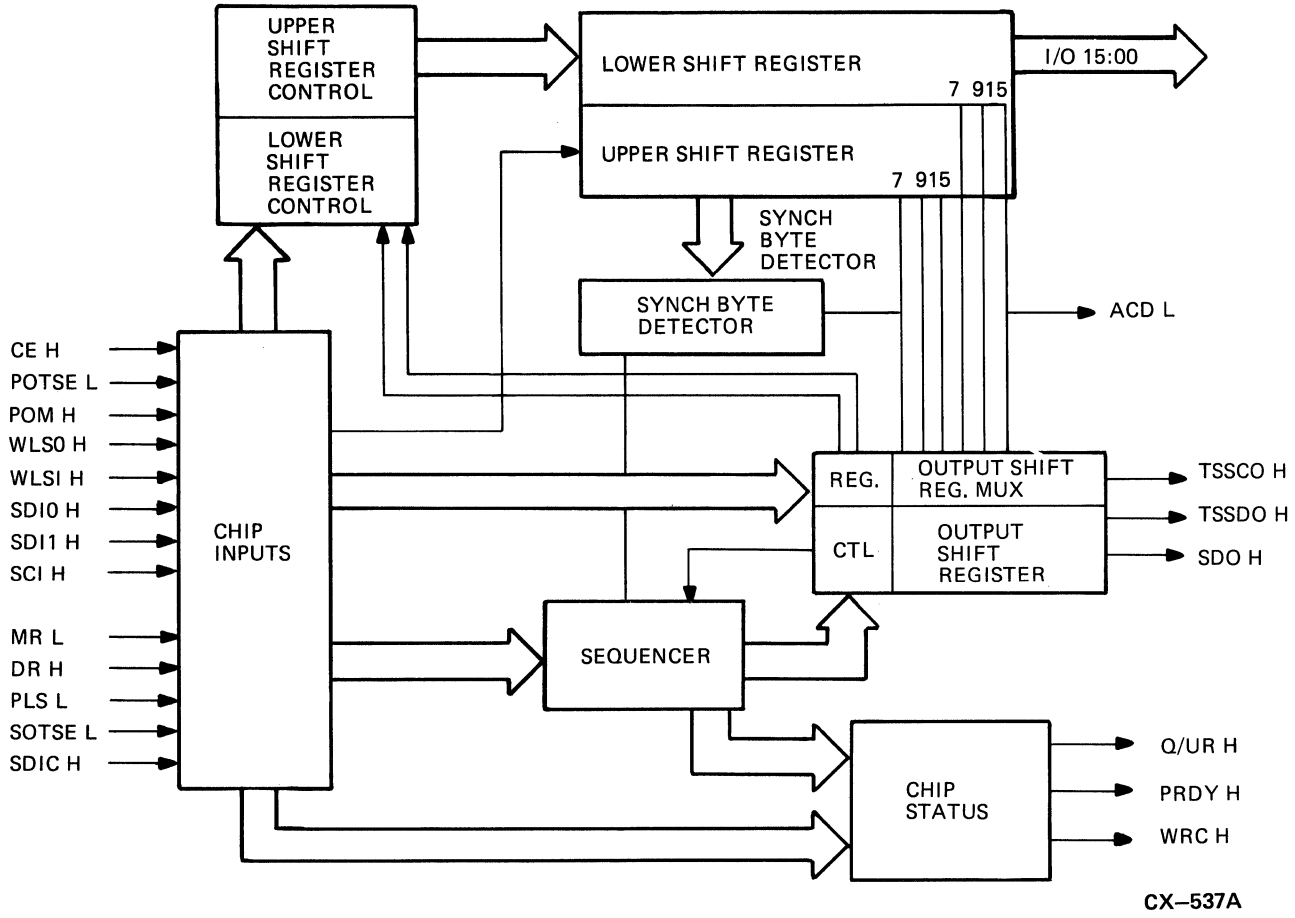


Figure A-2 SERDES Functional Block Diagram

A.2 REED SOLOMON GENERATOR (R-S GEN)

The R-S Gen is a dynamic N-MOS LSI device that implements a Reed-Solomon error correction code. The chip accepts up to 10060 bits of data in 10-bit parallel (symbol) form and generates from these symbols 170 bits (17 10-bit symbols) of ECC information. The chip also contains hardware to compare 170 bits of calculated ECC with 170 bits of Read ECC. This compare is accomplished via an EXCLUSIVE-OR operation. If the data does not compare, the result allows external correction of invalid data. The algorithm is capable of correcting any number of errors contained in up to eight of the 10-bit symbols in the data field. Table A-2 describes the signal names and their definition for all of the pins on the R-S Gen.

For Internal Use Only

Table A-2 Reed-Solomon Generator Signal Definitions And Functions

Pin #	Pin Name	Input/ Output	Signal Name/Function
25	DIØ H	Input	DATA IN 9:Ø. These 1Ø lines are
2Ø	DI1 H	Input	the input data lines.
26	DI2 H	Input	
18	DI3 H	Input	
2	DI4 H	Input	
15	DI5 H	Input	
4	DI6 H	Input	
13	DI7 H	Input	
7	DI8 H	Input	
1Ø	DI9 H	Input	
24	DOØ H	Output	DATA OUT 9:Ø. These 1Ø lines are
21	DO1 H	Output	the output data lines. They
27	DO2 H	Output	contain either calculated ECC or
19	DO3 H	Output	residue from an ECC comparison.
1	DO4 H	Output	
16	DO5 H	Output	
5	DO6 H	Output	
12	DO7 H	Output	
6	DO8 H	Output	
11	DO9 H	Output	
22	FBE H	Input	Feedback Enable. FBE is asserted when ECC is calculated. It is negated when ECC is output or when ECC is compared.
23	INE H	Input	Input Enable. Input enable must be asserted when the R-S Gen is calculating ECC on data input and when the R-S Gen is calculating the compare residue on ECC input. It must be negated when the R-S Gen is outputting ECC.
17	CLR L	Input	Clear. This signal is synchronous with CLK H and is asserted for a full clock period. It initializes the R-S Gen.
8	CLK H	Input	Clock. This signal provides the timing for the R-S Gen. During normal operation, input/output signals are latched/available on the falling edge of the clock.

Table 2 (cont.)

Pin #	Pin Name	Input/ Output	Signal Name/Function
3	ERR L	Output	Error Out. This signal is negated by CLR and asserted when the ECC residue is not equal to zero indicating an ECC error. The latch is enabled when FBE is negated and INE is asserted.

Figure A-3 shows the R-S Gen functional block diagram. The Clear signal zeros the seventeen 10-bit register stages and clears the error detector. The clock generator produces a two phase clock for each of the registers. The following operations are performed to generate an ECC:

- 10-bit parallel data enters through the input gate when the input enable signal is asserted.
- The data passes through the feedback gate to the alpha generator when the feedback enable signal is asserted.
- The alpha generator produces eight 10-bit terms composed of three to seven input bits.
- The terms are EXCLUSIVE-ORed with the register outputs and the result clocked into the next register stage. This process continues until all data words are entered.

At this point, the ECC is stored at the outputs of the 17 register stages. To output the ECC symbols, the input and feedback enables are negated and the register data shifts out at the clock frequency.

To verify input data from the disk, the R-S Gen generates an ECC based on the input data. Then, the internally generated ECC is shifted through the register stages and an EXCLUSIVE-OR operation is performed with the input ECC. The result goes to the output and the error generator. If the result is not zero (both ECC fields were the same), the error generator asserts the error signal.

For Internal Use Only

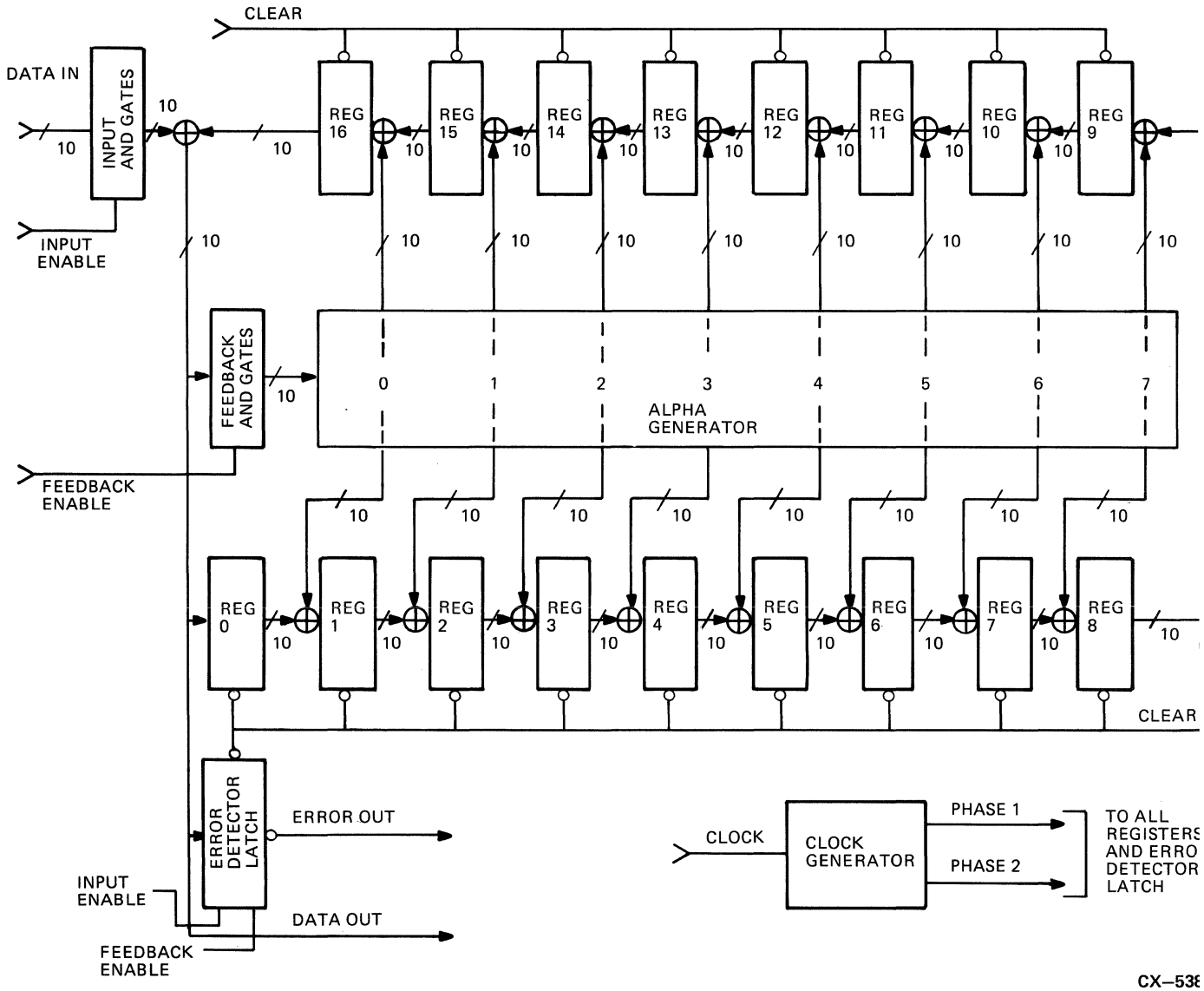


Figure A-3 R-S Gen Functional Block

Figure A-4 shows the timing of a typical Write operation. Figure A-5 show the timing of a typical Read operation.

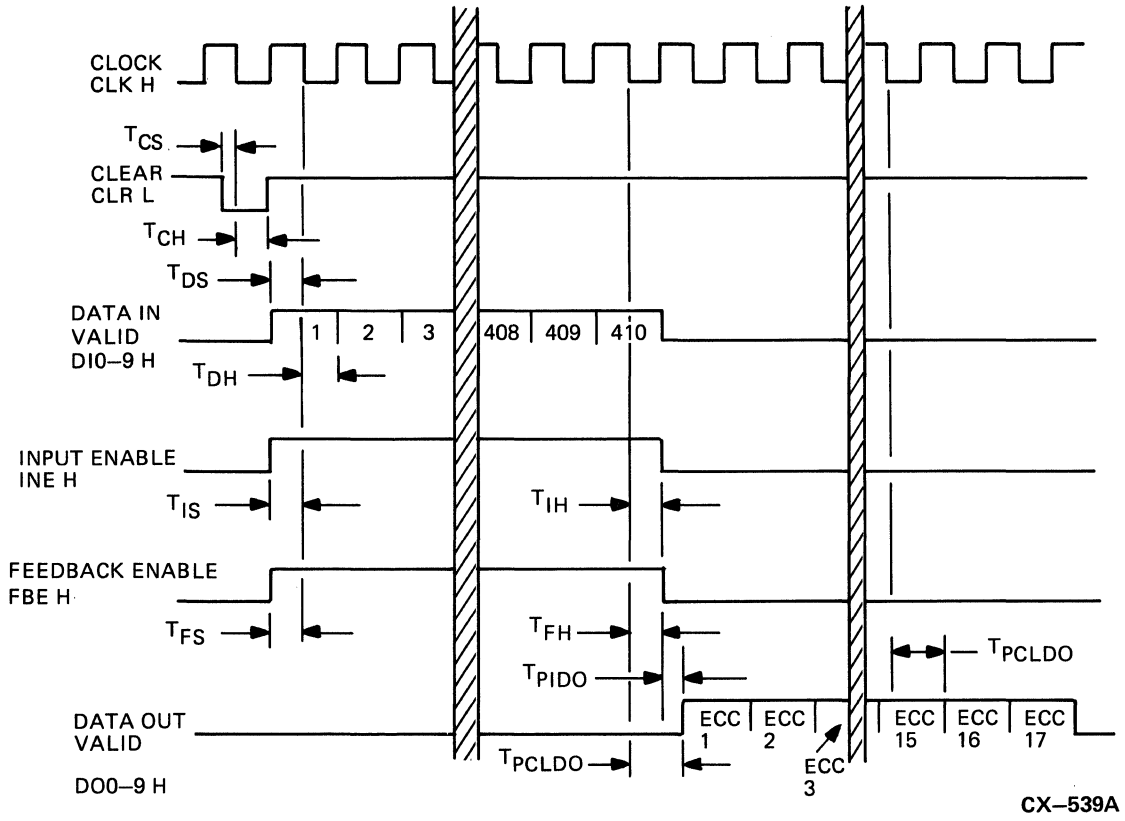


Figure A-4 R-S Gen Typical Write Operation

For Internal Use Only

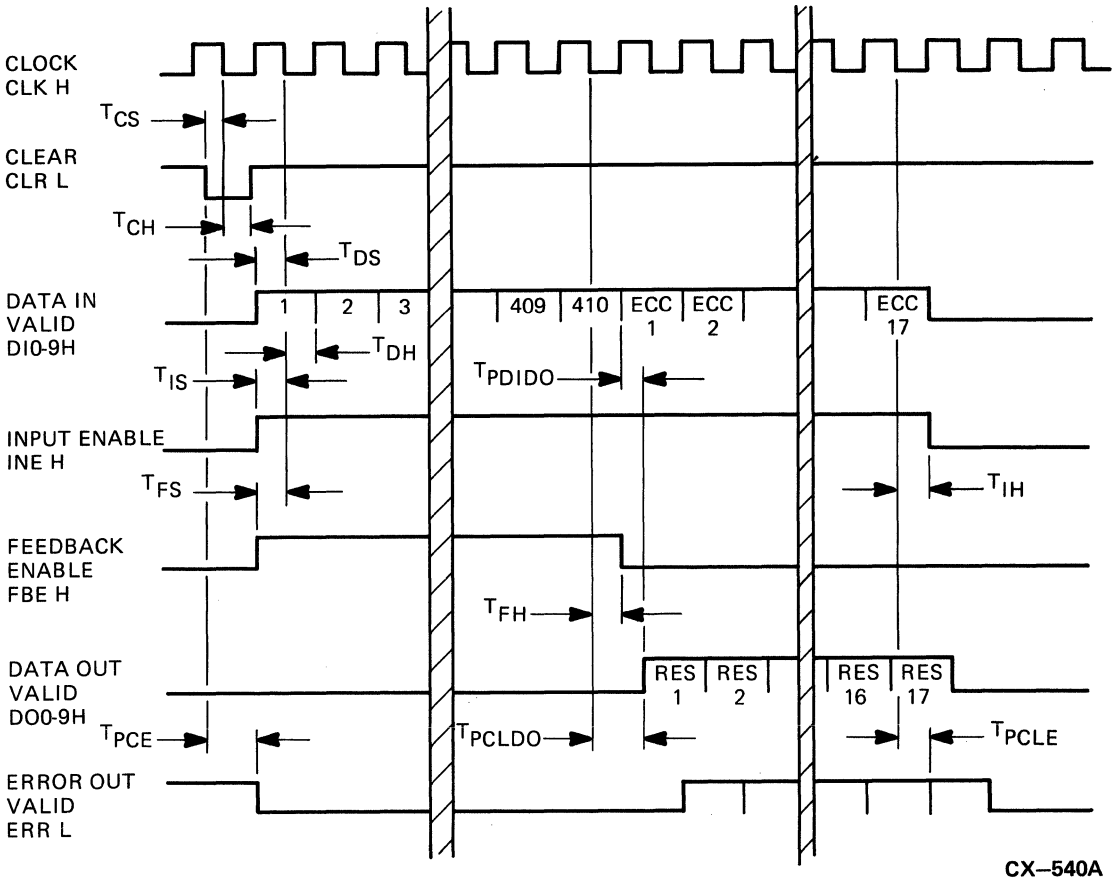


Figure A-5 R-S Gen Typical Read Operation

A.3 STATUS/EVENT CODES

Table A-3 lists the status/event codes. These codes are found in the status/event field of the error log.

Table A-3 Error Log Status Event Codes

Hex Code	Octal Code	Description

The first list in the following table is a group of codes that determine the major status or event being reported--that is, a media format error or a drive error, etc. Within these major categories are more specialized subcodes that break down the major category further. For example, if a hex code of (B) is a drive error, a hex code of (AB) reveals that there is also a drive clock dropout. A separate list is given for each of the subcode values.

For Internal Use Only

Table 3 (cont.)

Hex Code	Octal Code	Description
Ø	Ø	Success
1	1	Invalid command
2	2	Command aborted
3	3	Unit off-line
4	4	Unit available
5	5	Media format error
6	6	Write protected
7	7	Compare error
8	1Ø	Data error
9	11	Host buffer access error
A	12	Controller error
B	13	Drive error
1F	37	Status/event code mask
Success Subcode (Hex Ø)		
2Ø	4Ø	Spin-down ignored
4Ø	1ØØ	Still connected
8Ø	2ØØ	Duplicate unit number
1ØØ	4ØØ	Already online
2ØØ	1ØØØ	Still online
Invalid Command Subcode (Hex 1)		
1	1	Invalid message length
Command Aborted Subcode (Hex 2)		**NOT USED**
Unit Off-line Subcode (Hex 3)		
3	3	Unit unknown or online to another controller
23	43	No volume mounted or drive disabled via RUN/STOP switch

Table 3 (cont.)

Hex Code	Octal Code	Description
43	1Ø3	Unit inoperative - For SDI drives, the controller has marked the drive inoperative due to an unrecoverable error in a previous level 2 exchange, or the drive has a duplicate unit identifier.
83	2Ø3	Duplicate unit number
1Ø3	4Ø3	Unit disabled by field service or diagnostic. For SDI drives, the DD bit is set.

Unit Available Subcode (Hex 4) NOT USED

Media Format Error Subcode (Hex 5)

A5	245	Format mismatch - Disk is not formatted with 512 byte sectors. The disk's FCT indicates it is formatted with 576 byte sectors, and either the controller or the drive only supports 512 byte sectors.
C5	3Ø5	FCT corrupted - Disk is not formatted or the FCT is corrupted.
1Ø5	4Ø5	RCT corrupted - The RCT search algorithm encounters an invalid RCT entry.
125	445	No replacement block available

Write-Protected Subcode (Hex 6)

1ØØ6	1ØØØ6	Unit is software write protected.
2ØØ6	2ØØØ6	Unit is hardware write protected.

Compare Error Subcode (Hex 7) NOT USED

Data Error Subcode (Hex 8)

Table 3 (cont.)

Hex Code	Octal Code	Description
8	1Ø	Sector written with "Force Error" modifier.
48	11Ø	Invalid header - The subsystem reads an invalid or inconsistent header for the requested sector. Causes of an invalid header include header mis-sync, header sync time-out, or an inconsistent header.
68	15Ø	Data sync time-out - Data sync is not found.
88	21Ø	Correctable error in ECC field - A transfer encounters a correctable error in which only the ECC field is affected.
E8	35Ø	Uncorrectable ECC error - A transfer encounters an ECC error that exceeds the correction capability of the subsystem's error correction algorithm.
1Ø8	41Ø	One symbol ECC error
128	45Ø	Two symbol ECC error
148	51Ø	Three symbol ECC error
168	55Ø	Four symbol ECC error
188	61Ø	Five symbol ECC error
1A8	65Ø	Six symbol ECC error
1C8	71Ø	Seven symbol ECC error
1E8	75Ø	Eight symbol ECC error - A transfer encounters a correctable ECC error with the specified number of ECC symbols in error. The number of symbols in error corresponds to the severity of the error.
Host Buffer Access Error Subcode (Hex 9)		
9	11	Host buffer access error - The controller is unable to access a host buffer to perform a transfer and has no visibility into the cause of the error.

Table 3 (cont.)

Hex Code	Octal Code	Description
29	51	Odd transfer address
49	111	Odd byte count
69	151	Nonexistent memory error
89	211	Host memory parity error
Controller Error Subcode (Hex A)		
A	12	Reserved for host command time-out expired
2A	52	SERDES overrun or underrun error - Either the drive is too fast for the controller, or a controller hardware fault has prevented the controller microcode from being able to keep up with the data transfer to or from the drive.
4A	112	EDC error - The sector is read with correct or correctable ECC and an invalid EDC. There is most likely a fault in the ECC logic of this controller or the controller that last wrote the sector.
6A	152	Inconsistent internal control structure - Some high level check detects an inconsistent data structure. For example, a reserved field contains a non-zero value, or the value in a field is outside its valid range. This error usually implies the existence of a microcode bug.
8A	212	Internal EDC error - Some low level check detects an inconsistent data structure. For example, a microcode implemented checksum or vertical parity (hardware parity is horizontal) associated with internal sector data is inconsistent. This error usually implies a fault in the memory addressing logic of one or more of the controller's processing elements. It may also result from a double bit error or other error that exceeds the error detection capability of the controller's hardware memory checking circuitry.

For Internal Use Only

Table 3 (cont.)

Hex Code	Octal Code	Description
10A	412	Data bus overrun - The controller attempts to perform too many concurrent transfers, causing one or more of them to fail due to a data overrun or underrun.
12A	452	Data memory error - The controller detects an error in an internal memory, such as a parity error or a nonresponding address. This subcode only applies to errors that are not reported via MSCP. These errors do not affect the controller's ability to properly generate end and error log messages. For most controllers, this subcode only is returned for controller memory errors in data or buffer memory and noncritical control structures. If the controller has several such memories, the specific memory involved is reported as part of the error address in the error log message.
14A	512	PLI reception buffer parity error
16A	552	PLI transmission buffer parity error
Drive Error Subcode (Hex B)		
2B	53	Drive command time-out - For SDI drives, the controller's time-out expires for either a level two exchange or the assertion of read/write ready after an initiate seek.
4B	113	Controller-detected transmission error - For SDI drives, the controller detects an invalid framing code or a checksum error in a level two response from the drive. The KDA50-Q also returns this subcode for controller detected protocol errors. All other SDI controllers return subcode 9 for protocol errors.
6B	153	Positioner error (mis-seek) - The drive reports that a seek operation is successful, but the controller has determined that the drive has positioned itself to an incorrect cylinder.

Table 3 (cont.)

Hex Code	Octal Code	Description
8B	213	Lost read/write ready during or between transfers - For SDI drives, read/write ready is negated when the controller attempts to initiate a transfer or at the completion of a transfer. Read/write ready is previously asserted indicating the completion of the previous seek. This usually results from a drive detected transfer error, in which case an additional error log message may be generated containing the "drive detected error" subcode.
AB	253	Drive clock dropout - For SDI drives, either data clock or state clock is missing when it should be present. This is usually detected by a time-out.
CB	313	Lost receiver ready for transfer - For SDI drives, receiver ready is negated when the controller attempts to initiate a transfer or does not assert at the completion of a transfer. This includes all cases of the controller's time-out expiring for a transfer operation (level one real-time command).
EB	353	Drive-detected error - For SDI drives, the controller receives a get status or unsuccessful response with the EL flag set. The controller may also receive this response with the DR flag set. It does not support automatic diagnosis for that drive type.
10B	413	Controller-detected pulse or data parity error - For SDI drives, the controller detects a pulse error on either the state or data line, or the controller detects a parity error in a state frame.
12B	453	Drive-requested error log (EL bit set)
14B	513	Response length or opcode error - For SDI drives, a level two response from the drive has an invalid opcode, has an improper length, or is not a possible response in the context of the exchange.
16B	553	Clock resumption fails after initialization - For SDI drives, the drive clock does not start after a controller attempt to initialize the drive.

For Internal Use Only

Table 3 (cont.)

Hex Code	Octal Code	Description
18B	613	Clock persists after initialization - For SDI drives, the drive clock continues beyond drive initialization.
1AB	653	Receiver-ready collision - For SDI drives, the controller attempts to assert its receiver ready (to receive a response) and the drive's receiver ready is still asserted (to receive a command).
1CB	713	Response overflow

-L-

-M-

Digital Equipment Corporation • Colorado Springs, CO 80919