EK-RB730-TD-001

VAX-11/730 IDC Technical Description

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, September 1982

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

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

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CHAPTER 1 INTRODUCTION

1.1 GENERAL DESCRIPTION 1-1
1.2 PHYSICAL DESCRIPTION 1-1
1.3 POWER REQUIREMENTS 1-3
1.4 FUNCTIONAL DESCRIPTION 1-4
1.4.1 Disk Drive Select and Drive Status Monitor 1-4
1.4.2 Asserting Disk Drive Commands 1-4
1.4.3 Synchronization of IDC Operation 1-4
1.4.4 Address Location 1-4
1.4.5 Data Transfers 1-5
1.4.6 Verifying Data Integrity 1-5
1.4.7 Status Word Generation 1-5

CHAPTER 2 INTERFACES

2.1 IDC INTERFACES 2-1
2.2 IDC/PORT BUS AND UNIBUS INTERFACES 2-1
2.2.1 Control Words 2-3
2.2.1.1 IDC Control Word 2-3
2.2.1.2 Disk Drive Control Words 2-3
2.2.1.3 RL02 Get Status Command 2-3
2.2.1.4 RL02 Seek Command 2-3
2.2.1.5 R80 Seek Command 2-3
2.2.1.6 R80 Head Select Command 2-8
2.2.1.7 R80 Recalibrate Command 2-8
2.2.2 Address Information 2-8
2.2.3 Status Information 2-8
2.2.3.1 IDC Status Word 2-8
2.2.3.2 RL02 Status 2-8
2.2.3.3 R80 Status 2-8
2.2.4 Error Detection Information 2-8
2.2.5 Port Microinstruction Inputs 2-16
2.2.6 PORT INSTR Input 2-16
2.2.7 READ PORT and SEL ACC IN Inputs 2-16
2.2.8 CPU P2 and PORT CLOCK Inputs 2-16
2.3 IDC/R80 INTERFACE 2-16
2.3.1 R80 TAG 3:1 and R80 TAG BUS 9:0 2-16
2.3.2 ACLO, GND, and R80 INITIALIZE 2-21
2.3.3 R80 WRITE DATA and R80 WRITE CLOCK 2-21
2.3.4 R80 SECTOR COUNT 1, 2, 4, 8, and 16 2-21
2.3.5 R80 FAULT, R80 PLUG VALID, R80 SEEK ERROR, R80 ON CYLINDER, R80 DRIVE READY, and R80 WRITE PROTECT 2-21
2.3.6 R80 SELECT ADDRESS 1 and 2 2-21
2.3.7 R80 INDEX PULSE and R80 SECTOR PULSE 2-23
2.3.8 R80 READ DATA and R80 READ CLOCK 2-23

2.4 IDC/RL02 INTERFACE 2-23
2.4.1 RL DRIVE COMMAND and RL SYSTEM CLOCK 2-23
2.4.2 RL DRIVE SELECT 0 and 1 2-25
2.4.3 POWER FAIL (ACLO) 2-25
2.4.4 RL WRITE GATE and RL WRITE DATA 2-25
2.4.5 RL DRIVE READY 2-26
2.4.6 RL DRIVE ERROR 2-26
2.4.7 RL STATUS and RL STATUS CLOCK 2-26
2.4.8 RL SECTOR PULSE 2-26
2.4.9 RL READ DATA 2-26

CHAPTER 3 THEORY OF OPERATION

3.1 IDC FUNCTIONS 3-1
3.2 OVERALL IDC OPERATION 3-3
3.2.1 Initiating IDC Functions 3-3
3.2.1.1 Loading Required Inputs 3-3
3.2.1.2 Loading the IDC Control Word 3-3
3.2.2 IDC Operation 3-4
3.2.3 Transfer of Information and Data from IDC to CPU 3-4
3.2.3.1 IDC Status Information Transfer (IDC to CPU) 3-4
3.2.3.2 Disk Drive Status Information Transfer (IDC to CPU) 3-4
3.2.3.3 ECC/CRC Error Detection Information Transfer (IDC to CPU) 3-5
3.2.3.4 Current Address Information Transfer (IDC to CPU) 3-5
3.2.3.5 Data Transfer (IDC to CPU) 3-5
3.3 OVERALL IDC LOGIC FAMILIARIZATION 3-5
3.3.1 IDC Port Control Logic 3-5
3.3.2 Microcontroller 3-6
3.3.3 Y-Bus Transceivers 3-6
3.3.4 Disk Address Register 3-6
3.3.5 Data Input Register, Data Buffer, and Data Register Control Logic, Data Output Register, Read Data Tristate Drivers, and R80 Multiplexer 3-6
3.3.6 Control Status Register 3-6
3.3.7 Clock Control 3-9
3.3.8 TAG Bus Control 3-9
3.3.9 Serializer 3-9
3.3.10 Header/Data Comparator 3-9
3.3.11 Data Shift Register 3-9
3.3.12 NRZ Data Formatter 3-10
3.3.13 MFM Encoder 3-10
3.3.14 ECC/CRC Logic 3-10
3.3.15 Read Data Separator 3-10
3.3.16 Status/Data Gate 3-10
3.3.17 Disk Data Multiplexer 3-10
3.3.18 Data Synchronizer 3-10
3.3.19 Sector and Index Pulse Multiplexer and Synchronizer 3-10

3.4 IDC FUNCTIONAL THEORY OF OPERATION 3-11
3.4.1 Seek Functions 3-12
3.4.1.1 RL02 Seek 3-12
3.4.1.2 R80 Seek 3-12
3.4.2 RL02 Get Status 3-13
3.4.3 R80 Get Status 3-14
3.4.4 Read Header 3-14
3.4.4.1 RL02 Read Header 3-14
3.4.4.2 R80 Read Header 3-15
3.4.5 Write Data, Read Data, and Write Check Data 3-16
3.4.5.1 RL02 Write Data, Read Data, and Write Check 3-17
3.4.5.2 R80 Write Data, Read Data, and Write Check 3-24
3.4.6 Read Data Without Header Check 3-32
3.4.6.1 RL02 Read Data Without Header Check 3-32
3.4.6.2 R80 Read Data Without Header Check 3-34
3.4.7 Write Format 3-37
3.4.8 Idle Mode 3-38
3.5 DETAILED FUNCTIONAL DESCRIPTIONS 3-38
3.5.1 Disk Drive Selection and Drive Status Monitor 3-38
3.5.1.1 Generation of DRIVE SEL 0 and 1 3-39
3.5.1.2 Generation of RL02 and R80 3-39
3.5.1.3 Gating DRIVE RDY 3-39
3.5.1.4 Gating DRIVE ERR 3-39
3.5.2 TAG Bus Control Logic 3-39
3.5.2.1 Asserting R80 Seek, Head Select, and Recalibrate Commands 3-41
3.5.2.2 Asserting Read Gate 3-41
3.5.2.3 Asserting Write Gate 3-41
3.5.3 Clock Control Logic 3-41
3.5.3.1 Enable SYS CLOCK 3-46
3.5.3.2 Enable RL STATUS CLOCK or CPU CLOCK 3-46
3.5.3.3 Enable DISK CLOCK 3-47
3.5.4 Sync Byte Recognition Logic 3-47
3.5.5 RL02 Header Comparison Logic 3-50
3.5.6 R80 Header Comparison and Skip Sector Monitor Logic 3-52
3.5.6.1 R80 Header Comparison Logic 3-55
3.5.6.2 Skip Sector Monitor Logic 3-57
3.5.7 SKIP SECTOR CONTROL LOGIC 3-57
3.5.8 Write Check Data Comparison Logic 3-58
3.5.9 Interrupt Control Logic 3-62
3.5.9.1 UBUS BR5 3-62
3.5.9.2 PORT XFER REQ 3-63
3.5.10 IDC Control Register, Timeout Logic, and Status Logic 3-63
3.5.10.1 IDC Control Register 3-63
3.5.10.2 Timeout and Status Logic 3-67
3.5.11 Serializing Data from Data Buffer and Sync Byte Tristate Drivers 3-69
3.5.12 Formatting and Loading Disk Drive Read Data in Data Buffers 3-73

3.5.13 IDC/CPU Interface Logic 3-76
3.5.13.1 Loading CSR 3-78
3.5.13.2 Reading CSR 3-79
3.5.13.3 Loading Disk Address Register 3-80
3.5.13.4 Read Disk Address Register 3-81
3.5.13.5 Reading ECC/CRC Logic 3-82
3.5.13.6 Loading IDC Data Buffers 3-83
3.5.13.7 Reading IDC Data Buffers 3-86
3.5.13.8 Initializing/Clearing IDC and R80 Disk Drive 3-90
3.5.14 Microcontroller Branching, Loops, and Stalls 3-92
3.5.14.1 Microcontroller Branching 3-93
3.5.14.2 Microcontroller Loops 3-93
3.5.14.3 Microcontroller Stalls 3-93
3.5.15 Read Data Separator Operation 3-93
3.5.15.1 Phase Lock Loop (PLL) 3-95
3.5.15.2 Data Separator 3-97
3.5.16 MFM Encoding and Write Precompensation 3-98
3.5.16.1 MFM Encoding 3-98
3.5.16.2 Write Precompensation 3-100

CHAPTER 4 MAINTAINABILITY FEATURES

4.1 MAINTAINABILITY FEATURES 4-1
4.1.1 Maintenance Mode 4-1
4.1.2 Data Loopback 4-1
4.1.3 Write Inhibit and Timeout Inhibit 4-1
4.1.4 Defeatable Enables 4-1

CHAPTER 5 PROGRAM INTERFACE

5.1 BASIC SYSTEM OPERATION 5-1
5.2 PROGRAMMING OVERVIEW 5-1
5.2.1 IDC Registers 5-1
5.2.1.1 Control Status Register (CSR) 5-1
5.2.1.2 Bus Address Register (BAR) 5-6
5.2.1.3 Byte Count Register (BCR) 5-6
5.2.1.4 Disk Address Register (DAR) 5-7
5.2.1.5 Multipurpose Register (MPR) 5-7
5.2.1.6 ECC Position Register 5-8
5.2.1.7 ECC Pattern Register 5-8
5.2.1.8 IDC Initialization Register 5-8
5.3 COMMANDS 5-8
5.3.1 Positioning Commands 5-8
5.3.1.1 Seek Function 5-8
5.3.2 Data Transfer Commands 5-9
5.3.2.1 Read Header Function 5-9
5.3.2.2 Write Data Function 5-9
5.3.2.3 Read Data Function 5-9
5.3.2.4 Read Data Without Header Check Function 5-9

5.3.2.5 Write Check Function 5-9
5.3.2.6 Get Status Function 5-9
5.3.3 Housekeeping Commands 5-12
5.3.3.1 NOP Function 5-12
5.4 R80 ECC HANDLING 5-12
5.5 HARDWARE ERROR RECOVERY 5-12
5.6 SOFTWARE ERROR CORRECTION 5-12
5.7 R80 SKIP SECTOR OPERATION 5-13
5.7.1 R80 Bad Spot Problem 5-13
5.7.2 The Concept of Skip Sectoring 5-13
5.7.2.1 Software Handling of Skip Sector Errors 5-13
5.7.3 Skip Sectoring (with the Automatic Inhibit Bit Set) 5-13
5.8 R80 FORMATTING 5-13
5.9 EXAMPLES OF SYSTEM OPERATION 5-14
5.9.1 Seek Operation 5-14
5.9.2 Data Transfer Operation (Read/Write) 5-15

APPENDIX PROGRAMMED ARRAY LOGIC DEVICES (PALS) A-1

FIGURES

Figure No.	Title	Page
1-1	Interconnections for Possible Configuration of RB730 Disk Subsystem	1-2
2-1	IDC Signal Interfaces	2-2
2-2	IDC Control Word Data Format and Bit Significance	2-4
2-3	RL02 Get Status Command Data Format and Bit Significance	2-5
2-4	RL02 Seek Command Data Format and Bit Significance	2-6
2-5	R80 Seek Command Data Format and Bit Significance	2-7
2-6	R80 Head Select Command Data Format and Bit Significance	2-9
2-7	R80 Recalibrate Command Data Format and Bit Significance	2-10
2-8	RL02 Read/Write Data Address Data Format and Bit Significance	2-11
2-9	R80 Read/Write Data Address Data Format and Bit Significance	2-12
2-10	IDC Status Word Data Format and Bit Significance	2-13
2-11	RL02 Status Information Data Format and Bit Significance	2-14
2-12	R80 Status Information Data Format and Bit Significance	2-15

2-13 Port Microinstructions Format and Bit Significance 2-17
2-14 Timing Relationship of PORT CLOCK and CPU P2 Inputs to IDC 2-20
2-15 R80 Write Data Format and Data Transfer Timing: IDC to R80 2-22
2-16 R80 Sector Pulse and Index Pulse Timing 2-23
2-17 R80 Read Data Format and Data Transfer Timing: R80 to IDC 2-24
2-18 RL Write Data Format and Data Transfer Timing: IDC to RL02 2-25
2-19 Format and Bit Significance of RL02 Status Information Transfer: RL02 to IDC 2-26
2-20 RL Read Data Format and Data Transfer Timing: RL02 to IDC 2-27

3-1 IDC Functional Block Diagram 3-7
3-2 Disk Drive Selection and Drive Status Monitor 3-38
3-3 TAG Bus Control Logic Functional Diagram 3-40
3-4 Clock Control Logic Functional Block Diagram 3-41
3-5 Clock Control Logic Timing Diagram 3-42
3-6 Sync Byte Recognition Logic Functional Block Diagram 3-46
3-7 Sync Byte Recognition Logic Timing Diagram 3-47
3-8 RL02 Header Comparison Logic Functional Block Diagram 3-49
3-9 RL02 Header Comparison Logic Timing Diagram 3-51
3-10 R80 Header Comparison and Skip Sector Monitor Logic Functional Block Diagram 3-52
3-11 R80 Header Data Modification and Comparison Data Control Timing 3-54
3-12 Skip Sector Control Logic Functional Block Diagram 3-55
3-13 Skip Sector Example 3-56
3-14 Write Check Data Comparison Logic Functional Block Diagram 3-57
3-15 Write Check Data Comparison Logic Timing Diagram 3-59
3-16 UBUS BR5 Interrupt Control Logic Functional Block Diagram 3-60
3-17 PORT XFER REQ Logic Functional Block Diagram 3-61
3-18 IDC Control Register Timeout Logic and Status Logic Functional Block Diagram 3-62
3-19 Data and Sync Byte Serialization Control Logic Functional Block Diagram 3-68
3-20 Data and Sync Byte Serialization Control Logic Timing Diagram 3-70
3-21 Read Data Formatting and Storage Control Logic Functional Block Diagram 3-72
3-22 Formatting and Loading Read Data Input to FIFO: Timing Diagram 3-73

3-23 IDC Register Source and Destination for Data and Information Transferred between IDC and CPU via CPU Y-Bus ... 3-74
3-24 IDC/CPU Interface Logic Functional Block Diagram ... 3-75
3-25 IDC Control Word Transfer Timing (CPU to IDC) ... 3-76
3-26 IDC Status Word Transfer Timing (IDC to CPU) ... 3-77
3-27 Disk Drive Control Word and Read/Write Address Transfer Timing (CPU to IDC) ... 3-78
3-28 Current Read/Write Address Transfer Timing (IDC to CPU) ... 3-79
3-29 Data Error Information Transfer Timing (IDC to CPU) ... 3-80
3-30 Data Byte Transfer Timing (CPU to IDC) ... 3-82
3-31 Data Longword Transfer Timing (CPU to IDC) ... 3-83
3-32 Data Byte Transfer Timing (IDC to CPU) ... 3-86
3-33 Single Data Longword Transfer Timing (IDC to CPU) ... 3-87
3-34 Automode Data Longword Transfer Timing (IDC to CPU) ... 3-88
3-35 Initialize/Clear Logic Diagram ... 3-89
3-36 Microcontroller Functional Block Diagram ... 3-90
3-37 Read Data Separator Block Diagram ... 3-92
3-38a VCO Output at Twice Data Rate (Frequency Lock) Timing Diagram ... 3-93
3-38b VCO Output Less Than Twice Data Rate Timing Diagram ... 3-93
3-38c VCO Output More Than Twice Data Rate Timing Diagram ... 3-94
3-39 Loop Lock Settling Time ... 3-94
3-40 Data Separator Detailed Diagram ... 3-95
3-41 Data Separator Timing Diagram ... 3-95
3-42 MFM Encoding and Write Precompensation Logic Functional Block Diagram ... 3-97
3-43 MFM Encoding and Write Precompensation Timing Diagram ... 3-98
3-44 Write Precompensation Early/Late Bit Combinations ... 3-99

5-1 Disk Address Register ... 5-7

A-1 Basic PAL Logic Configuration ... A-1
A-2 XOR Logic Function Using PAL Logic ... A-2
A-3 Typical PAL Symbology ... A-3
A-4 PAL Plot Listing ... A-5
A-5 Logic Diagram: PAL16L8 ... A-7
A-6 Logic Diagram: PAL16R4 ... A-8
A-9 Logic Diagram: PAL16R6 ... A-9
A-10 Logic Diagram: PAL16R8 ... A-10

TABLES

Table No.	Title	Page
1-1	Related Documentation	1-1
1-2	CPU-Initiated IDC Functions	1-3
2-1	Port Microinstruction Functions	2-18
3-1	IDC Functions	3-1
5-1	IDC Registers	5-2
5-2	Control Status Register Bit Assignments	5-3
5-3	MPR Bit Assignments	5-7
5-4	RL02 Get Status	5-10
5-5	R80 Get Status	5-11
A-1	PAL Device Types Used in the VAX-11/730	A-3

CHAPTER 1
INTRODUCTION

1.1 GENERAL DESCRIPTION

The Integrated Disk Controller (IDC) is part of the RB730 disk subsystem, a hardware option of the VAX-11/730. The RB730 disk subsystem includes the IDC and up to four RL02 disk drives or the IDC, one R80 disk drive, and up to three RL02 disk drives. The IDC provides the interface between the VAX-11/730 CPU and the associated disk drives of the RB730 disk subsystem for the purpose of data storage and retrieval. This manual presents the IDC technical description. Other documents related to the RB730 disk subsystem of the VAX-11/730 are listed in Table 1-1.

1.2 PHYSICAL DESCRIPTION

The IDC is a single hex-size module (M8388) that plugs into the VAX-11/730 backplane. All electrical connections for interfacing the IDC with the CPU are provided via the VAX-11/730 backplane. The electrical connections for interfacing the IDC with an R80 disk drive are provided via connectors J2 and J3. Connector J1 provides the electrical connections for interfacing the IDC with the RL02 disk drive(s) in a daisy chain fashion. Figure 1-1 shows the electrical connections for one possible configuration of the RB730 disk subsystem.

Table 1-1 Related Documentation

Document	Document Number
IDC Field Maintenance Print Set	MP-01278
R80 Disk Drive Field Maintenance Print Set	MP-01419
RL02 Disk Drive Field Maintenance Print Set	MP-01332
RL01/RL02 Disk Drive Technical Manual	EK-RL012-TM
R80 Disk Drive Technical Description	EK-00R80-TD
VAX-11/730 Central Processing Unit Technical Description	EK-KA730-TD

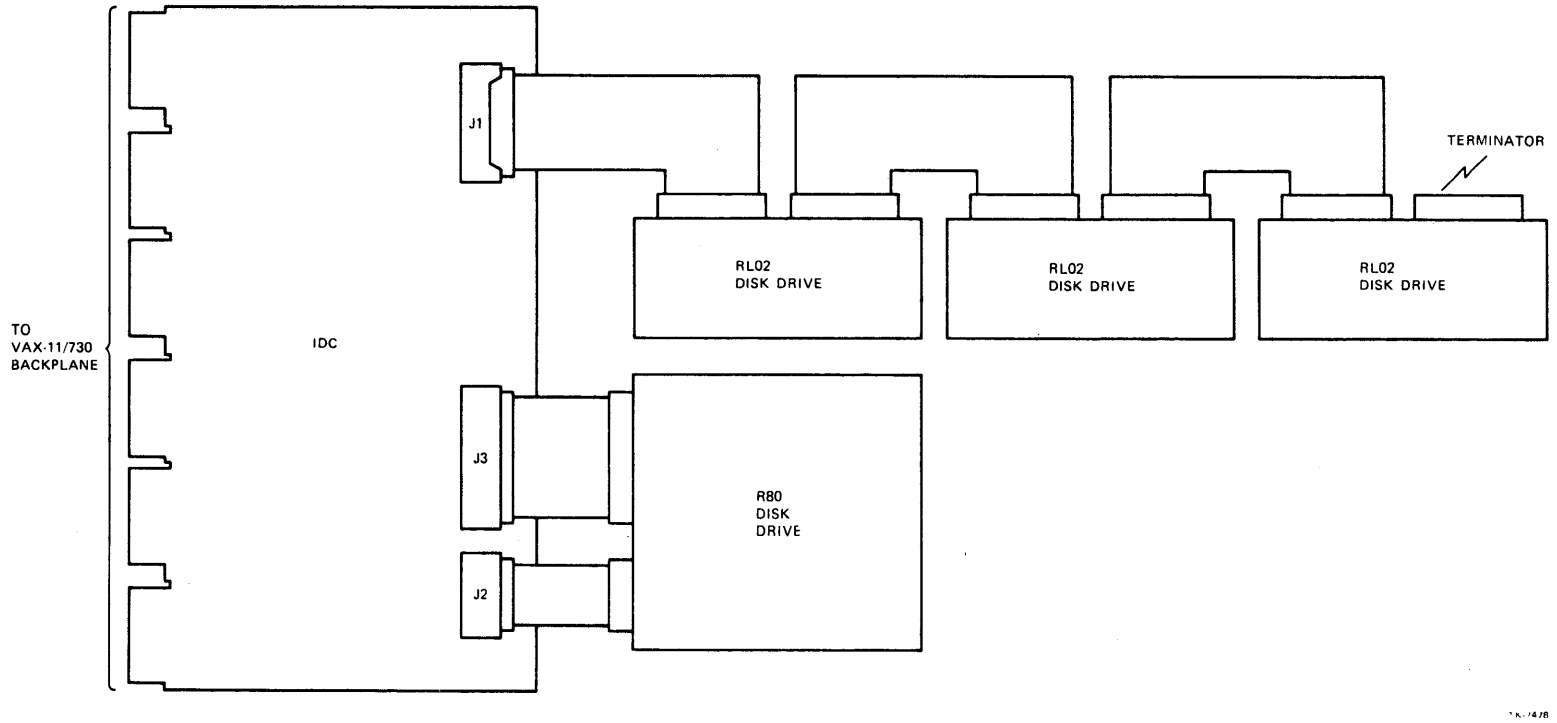


Figure 1-1 Interconnections for Possible Configuration of RB730 Disk Subsystem

1.3 POWER REQUIREMENTS

The IDC requires approximately 55 watts of dc power. The dc power requirements are as follows:

+5 V at 8.0 A
+15 V at 0.5 A
−15 V at 0.5 A

1.4 FUNCTIONAL DESCRIPTION

The IDC controls the operation of the associated disk drives of the RB730 disk subsystem to store and retrieve data. IDC operation is initiated by the CPU. The CPU loads the IDC with the information required to initiate and perform each of the functions necessary in storing and retrieving data from a specific address location of the selected disk drive. Once a function is specified by the CPU, the IDC controls the operation of the disk drive to perform the function. After the function has been completed, the IDC, if enabled, generates and asserts an interrupt to the CPU. Table 1-2 lists the functions that can be specified by the CPU and describes the purpose of each one.

Table 1-2 CPU-Initiated IDC Functions

Function Specified by CPU	Purpose
Seek	Position the specified disk drive read/write head over the specified cylinder and enable it.
Get status	Retrieve the status information from the specified disk drive and store it in the IDC data buffer.
Read header	Read from the specified disk drive the header information from the first sector encountered and store it in the IDC data buffer.
Write data	Write the data contained in the IDC data buffer at the specified read/write data address of the specified disk drive.
Read data	Read from the specified disk drive the data from the specified read/write data address and store the data in the IDC data buffer.
Read data without header check	Read from the specified disk drive the data from the first sector encountered and store the data in the IDC data buffer.
Write check	Read from the specified disk drive the data from the specified read/write data address and compare this data with data contained in the data buffers.
Write format (Used only with R80 disk drive)	Write new header data to each of the 32 sectors of the applicable R80 cylinder.
Maintenance	Place the IDC in the maintenance mode so that the IDC logic may be exercised by microdiagnostic routines.

When the IDC is not performing a CPU-specified function, it operates in the idle mode. In this mode, the IDC selects and monitors the status of each associated disk drive. If an operational status change is detected, the IDC alerts the CPU.

The IDC contains the control and monitoring circuitry for:

- Selecting the CPU-specified disk drive, monitoring disk drive operational status, and enabling the appropriate IDC data paths for interfacing to the selected disk drive
- Asserting the CPU-specified disk drive commands to control selection and positioning of the disk drive read/write heads
- Synchronizing IDC operation with the selected disk drive or the CPU
- Locating the address (sector) to or from which data is to be stored or retrieved
- Performing single or successive block data transfers between the CPU and the disk drives
- Verifying the integrity of the data through the storage and retrieval cycle
- Generating a status word that can be used by the CPU to identify data error detection, the reason the IDC could not complete a CPU-specified function, or disk drive status changes

1.4.1 Disk Drive Select and Drive Status Monitor

The IDC uses the disk drive select information specified by the CPU to select the desired disk drive. The IDC also monitors the operational status of the selected disk drive to make certain that the drive is operational and not busy before issuing further commands. The disk drive select information is also used within the IDC to select the proper data paths, specify the data buffer storage capacity, and gate the proper clocks for synchronization.

During the idle mode of operation, the IDC selects and monitors the operational status of the associated disk drives and records any detected operational status change. If a change is detected, the IDC alerts the CPU.

1.4.2 Asserting Disk Drive Commands

The IDC controls assertion of the CPU-specified disk drive commands in the format that is compatible with the selected disk drive. The RL02 disk drive commands are converted to a serial format and asserted to the RL02 disk drives; the R80 disk drive commands are gated to the R80 disk drive in a parallel format.

1.4.3 Synchronization of IDC Operation

Any one of six clocks may be selected by the IDC as the basic timing clock to ensure synchronous operation between the selected disk drive or CPU and the IDC.

1.4.4 Address Location

The IDC compares the read/write address specified by the CPU with the address information read from the selected disk drive to locate the address (sector) to or from which data is to be stored or retrieved.

1.4.5 Data Transfers

The IDC provides for single or successive block data transfers between the CPU and the disk drives. A block of data is defined as the data storage capacity of the disk for each addressable storage location (sector). Each sector of an RL02 disk drive provides the storage capacity for 256 bytes of data (one block). Each sector of the R80 disk drive provides the storage capacity for 512 bytes of data (one block).

The IDC provides buffering for all data to be written to or read from the disk. The IDC contains two data buffers: each data buffer provides storage for up to 512 bytes of data. Control of each of the data buffers is shared by the IDC and the CPU.

The CPU controls the data buffers to load the IDC with data to be written to the disk drives. The CPU also controls the data buffers to transfer the data contained in the data buffers from the IDC to the CPU.

The IDC controls the data buffers to store the data read from the disk drives until it is transferred to the CPU under CPU control. This dual IDC data buffer arrangement provides the capability for reading or writing successive sectors of data. While the IDC is reading or writing one sector of data using one of the IDC data buffers, the CPU can be using the other data buffer to transfer the data read from the previous sector or to load the data to be written in the next sector.

1.4.6 Verifying Data Integrity

The IDC verifies the integrity of the data throughout the storage and retrieval cycle. When data are being written to the disk, the IDC generates a coded word based on the configuration of data written to the disk. This coded word is also written on the disk during the write data cycle. When data are being read from the disk, the IDC generates a coded word based on the configuration of data read from the disk. After the data have been read, the coded word stored on the disk is compared with the coded word generated from the configuration of the data read from the disk. If the coded words are identical, data integrity has been maintained throughout the storage and retrieval cycle (the data read are identical to the data written).

1.4.7 Status Word Generation

During the performance of each CPU-specified function, the IDC tests the results of conditions and operations required to execute the function. The results of these tests are recorded and formatted to produce an IDC status word. During the idle mode, the IDC records any detected drive status change. Any recorded drive status change is provided as part of the IDC status word. The IDC status word can be read by the CPU.

CHAPTER 2
INTERFACES

2.1 IDC INTERFACES

The electrical connections between the IDC and the VAX-11/730 CPU (port bus and UNIBUS interfaces), the IDC and the R80 disk drive (R80 interface), and the IDC and the RL02 disk drive (RL02 interface) are shown in Figure 2-1.

All connections for the port bus and UNIBUS interfaces are provided via the VAX-11/730 backplane. The R80 interface is provided via two ribbon cables. A 60-wire ribbon cable connects J3 of the IDC with J201 of the R80 disk drive. A 26-wire ribbon cable connects J2 of the IDC with J202 of the R80 disk drive. The RL02 interface is provided via a 40-wire ribbon cable that connects J1 of the IDC with J12 of the RL02 disk drive.

All of the signal lines (except ACLO) at the R80 and RL02 interfaces are dual signal lines (indicated in Figure 2-1 by the dual listing of pin numbers at the interface connectors). The first pin number listed refers to the low level signal line; the second number refers to the high level signal line. These signal lines are driven or detected by differential line drivers or receivers.

The port bus interface BUS Y D31:D00 signal lines form a common bidirectional bus that interconnects the IDC and floating-point accelerator (FPA) with the Y-bus of the CPU data path module. These signal lines are driven/detected by octal transceivers on the IDC and the FPA. The rest of the signal lines at the port bus and UNIBUS interfaces are dedicated signal lines.

All of the input/output signals at the IDC/port bus, UNIBUS, R80, and RL02 interfaces are also shown in Figure 2-1. The following paragraphs discuss the characteristics and significance of the input/output signals at each of the interfaces.

2.2 IDC/PORT BUS AND UNIBUS INTERFACES

The input/output signals at the IDC/port bus and UNIBUS interfaces include BUS Y D31:D00, CSR17 and CSR14:10, PORT INSTR, READ PORT, SEL ACC IN, CPU P2, PORT CLOCK, PORT XFER REQ, XFER GRANT, BR5, ACLO, and DCLO.

The BUS Y D31:D00 signal lines are used to transfer control words, address information, and data from the CPU to the IDC, and to transfer IDC and disk drive status information, current address information, error detection information, and data from the IDC to the CPU. The CPU initiates and controls the transfer of all control words, information, and data between the IDC and the CPU via port microinstruction inputs to the IDC. The port microinstruction inputs are asserted via the CSR17 and CSR14:CSR10 signal lines.

The port microinstruction inputs to the IDC, together with the PORT INSTR input, are used to preset the IDC and to cause the transfer of control words, address information, and data from the CPU to the IDC. The port microinstruction inputs to the IDC together with the PORT INSTR, READ PORT, and SEL ACC IN inputs are used to cause the transfer of the IDC and disk drive status information, current address information, error detection information, and data from the IDC to the CPU. (The state of the SEL ACC IN input identifies the READ PORT signal as being applicable to the FPA or the IDC: a low SEL ACC IN signal indicates READ PORT is IDC specific.)

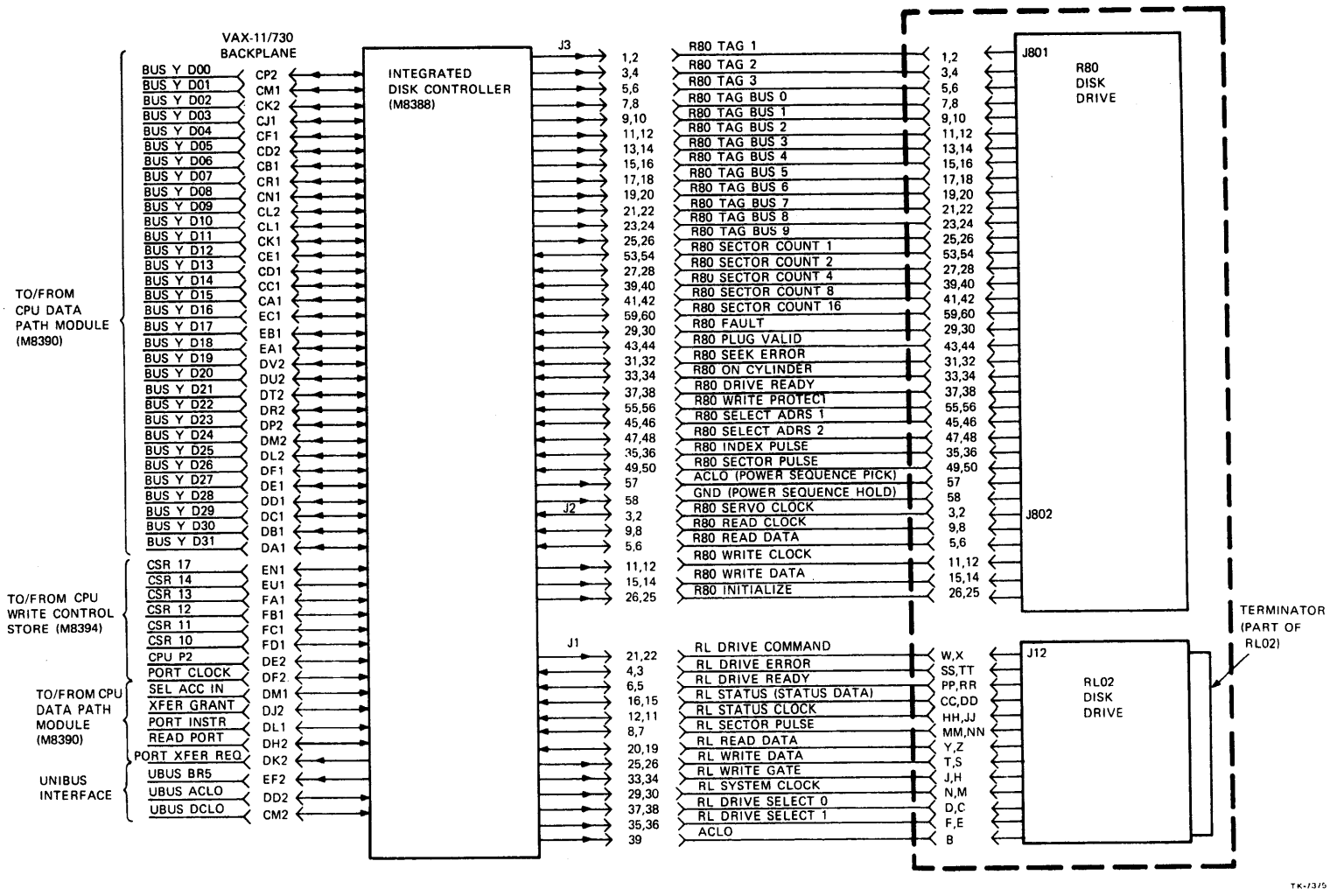


Figure 2-1 IDC Signal Interfaces

The CPU P2 and PORT CLOCK inputs to the IDC are the basic timing pulses that synchronize the operation of the IDC with the CPU.

The PORT XFER REQ and UBUS BR5 signals are the interrupt signals generated by the IDC. The PORT XFER REQ output is the fast interrupt output of the IDC. The PORT XFER REQ signal is asserted by the IDC to signal the CPU that the read data, write data, or write check function requested by the previous IDC control word input has been completed on the specified sector of data and that the IDC is waiting for further instructions. The CPU uses the XFER GRANT input to acknowledge the interrupt. The UBUS BR5 output is the slow interrupt and is asserted to the CPU via the UNIBUS. The UBUS BR5 signal specifies to the CPU that the IDC function requested by the IDC control word input has been completed, that one of the disk drives has changed operational status, or that the IDC operation has been halted due to an error.

The format and bit significance of the control words, address information, status information, and error detection information are discussed in Paragraphs 2.2.1 through 2.2.4. The data words transferred between the IDC and the CPU via the BUS Y D31:D00 signal lines may be in either byte (8-bit) or longword (32-bit) format. The format and bit decoding of the port microinstruction inputs applied via the CSR17 and CSR14:10 signal lines are discussed in Paragraph 2.2.5. The significance of the PORT INSTR input, READ PORT and SEL ACC IN inputs, and CPU P2 and PORT CLOCK inputs to the IDC are discussed in Paragraphs 2.2.6, 2.2.7, and 2.2.8, respectively.

2.2.1 Control Words

The control words input to the IDC via the BUS Y D31:D00 signal lines include the IDC control word and the disk drive control words.

2.2.1.1 IDC Control Word – The IDC control word specifies to the IDC the function to be performed, identifies the disk drive to be used, indicates whether R80 disk drive skip sectoring will be enabled, starts the specified function, and indicates if the IDC is to generate an interrupt (UBUS BR5) at completion. An IDC control word input must be applied to the IDC to initiate each of the IDC functions. The format and bit significance of the IDC control word are shown in Figure 2-2.

2.2.1.2 Disk Drive Control Words – The disk drive control words input to the IDC include the RL02 get status command, the RL02 seek command, the R80 seek command, the R80 head select command, and the R80 recalibrate command. The purpose of each of these commands is discussed in the following paragraphs.

RL02 Get Status Command

The RL02 get status command is used to cause the transfer of the RL02 status word (Paragraph 2.4.7) from the RL02 to the CPU via the IDC. The format and bit significance of the RL02 get status command are shown in Figure 2-3.

RL02 Seek Command

The RL02 seek command is used to reposition the RL02 read/write heads over another cylinder. The RL02 seek command specifies the direction and number of cylinders that the read/write heads are to move and which of the two heads is to be used. The format and bit significance of the RL02 seek command are shown in Figure 2-4.

R80 Seek Command

The R80 seek command is used to position the R80 read/write heads over the desired cylinder. The R80 seek command specifies the cylinder address over which the read/write heads are to be positioned. The format and bit significance of the R80 seek command is shown in Figure 2-5.

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	-	
BUS Y D01	(F0)	Function Select. These bits specify one of eight functions to be performed by the IDC. These bits are decoded as shown in the table in the figure.
BUS Y D02	(F1)	
BUS Y D03	(F2)	
BUS Y D04	-	
BUS Y D05	-	
BUS Y D06	(IE)	Interrupt Enable. When set, enables IDC to generate UBUS BR5 interrupt when applicable.
BUS Y D07	(CRDY)	Controller Ready. When reset, enables IDC to start function specified. Drive Select. These bits specify the address of the disk drive to be used to perform the function specified.
BUS Y D08	(DS0)	
BUS Y D09	(DS1)	
BUS Y D10	-	
BUS Y D11	-	
BUS Y D12	-	
BUS Y D13	-	
BUS Y D14	-	
BUS Y D15	-	
BUS Y D16	(ATTN0)	Attention bits. These bits are used to reset the attention bits asserted to the CPU via the IDC status word (see Figure 2-10).
BUS Y D17	(ATTN1)	
BUS Y D18	(ATTN2)	
BUS Y D19	(ATTN3)	
BUS Y D20	-	
BUS Y D21	-	
BUS Y D22	(SSEI)	Skip Sector Error Inhibit. When set, inhibits R80 skip sectoring.
BUS Y D23	(SSE FLAG)	Skip Sector Error Flag. This bit is used to reset the SSE flag asserted to the CPU via the IDC status word (see Figure 2-10).
BUS Y D24	-	
BUS Y D25	(MTN)	Maintenance. Used with Function Select bits F0, F1, and F2 to select maintenance function (see the table in this figure).
BUS Y D26	-	
BUS Y D27	(ASSI)	Automatic Skip Sector Inhibit. When cleared, allows automatic skip sectoring.
BUS Y D28	(WRT INH)	Write Inhibit. When set, inhibits writing to the disk drives and disables timeout from occurring. Used during maintenance function.
BUS Y D29	(R80 FORMAT)	R80 Format. Used with Function Select bits F0, F1, and F2 to specify R80 write format function (see the table in this figure).
BUS Y D30	-	
BUS Y D31	-	

* - Not used as part of IDC control word.

			Function Select			IDC Function
	R80	MTN	F2	F1	F0	Specified
	0	1	0	0	0	Maintenance
	1	0	0	0	0	R80 Write Format
	0	0	0	0	0	No Operation
	0	0	0	0	1	Write Check
	0	0	0	1	0	Get Status
	0	0	0	1	1	Seek
	0	0	1	0	0	Read Header
	0	0	1	0	1	Write Data
	0	0	1	1	0	Read Data
	0	0	1	1	1	Read Data Without Header Check

Figure 2-2 IDC Control Word Data Format and Bit Significance

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	(M)	Marker. Used at RL02 disk drive to indicate a new command word. (This bit must be set.)
BUS Y D01	(GS)	Get Status. When set, commands RL02 disk drive to gate RL02 status word to IDC.
BUS Y D02	-	
BUS Y D03	(RST)	Reset. When set, commands RL02 disk drive to clear its error register before gating RL02 status word to IDC. (BUS Y D03 must be a 1.)
BUS Y D04	-	
↕	↕	
BUS Y D31	-	

* - Not used as part of RL02 Get Status Word

Figure 2-3 RL02 Get Status Command Data Format and Bit Significance

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	(M)	Marker. Used at RL02 disk drive to indicate a new command word. (This bit must be a l.)
BUS Y D01	(GS)	Get Status. This bit must be cleared for RL02 seek instruction.
BUS Y D02	(DIR)	Direction. Indicates direction of movement of RL02 read/write heads. When cleared, indicates movement toward higher addresses; when set, movement toward lower addresses.
BUS Y D03	(RST)	Reset. When cleared, used at RL02 to indicate that a cylinder difference word is being applied.
BUS Y D04	(HS)	Head Select. Used at RL02 to identify read/write head to be used. When set, selects upper head; when cleared, selects lower head.
BUS Y D05	-	
BUS Y D06	-	
BUS Y D07	(DF0)	Difference. These bits are used at the RL02 disk drive to specify the number of cylinders the read/write heads are to move.
BUS Y D08	(DF1)	
BUS Y D09	(DF2)	
BUS Y D10	(DF3)	
BUS Y D11	(DF4)	
BUS Y D12	(DF5)	
BUS Y D13	(DF6)	
BUS Y D14	(DF7)	
BUS Y D15	(DF8)	
BUS Y D16	-	
↕	↕	
BUS Y D31	-	

* - Not used as part of RL02 seek command.

Figure 2-4 RL02 Seek Command Data Format and Bit Significance

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	(CA0)	Cylinder Address Bits. These bits are used at the R80 disk drive to identify the cylinder address over which the read/write heads are to be located.
BUS Y D01	(CA1)	
BUS Y D02	(CA2)	
BUS Y D03	(CA3)	
BUS Y D04	(CA4)	
BUS Y D05	(CA5)	
BUS Y D06	(CA6)	
BUS Y D07	(CA7)	
BUS Y D08	(CA8)	
BUS Y D09	(CA9)	
BUS Y D10	-	
BUS Y D11	-	
BUS Y D12	-	
BUS Y D13	(CA)	Cylinder Address. When set, specifies that this word is a cylinder address word.
BUS Y D14	(HS)	Head Select. This bit must be cleared for R80 seek command.
BUS Y D15	(CS)	Control Select. This bit must be cleared for R80 seek command.
BUS Y D16	-	
↕	↕	
BUS Y D31	-	

* - Not used as part of R80 seek instruction.

Figure 2-5 R80 Seek Command Data Format and Bit Significance

R80 Head Select Command
The R80 head select command specifies which one of the fourteen read/write heads of the R80 disk drive is to be enabled. The format and bit significance of the R80 head select command are shown in Figure 2-6.

R80 Recalibrate Command
This command is used to position the R80 disk drive read/write heads over cylinder 0. The format and bit significance of the R80 recalibrate command are shown in Figure 2-7.

2.2.2 Address Information
The read/write data address information input to the IDC is used to locate the initial sector of the disk drive cylinder to or from which data are to be written or read. The current address information output from the IDC is used to specify to the CPU the complete address of the last sector of data that was written or read. The format and bit significance of the RL02 and R80 read/write data address information are shown in Figures 2-8 and 2-9, respectively. The format and bit significance of the current address information are the same as the read/write data address input to the IDC.

2.2.3 Status Information
There are three status information outputs of the IDC; these include the IDC status, the RL02 status, and the R80 status.

2.2.3.1 IDC Status Word – The IDC status word specifies to the CPU the contents of the previous control word input (Figure 2-2), specifies whether the function selected by the previous control word input was executed successfully and completed within the time allowed by the IDC (approximately 150 milliseconds), informs the CPU about changes in the operational status of the disk drives, provides information that identifies the type of fault detected (if any), and indicates if the IDC had generated a slow interrupt request (asserted UBUS BR5). The format and bit significance of the IDC status word are shown in Figure 2-10.

2.2.3.2 RL02 Status – The RL02 status information specifies to the CPU the current operational state of the RL02 disk drive, the position of the disk brushes (over the disk or home), whether the read/write heads are over the disk, whether a fault condition has been detected, and if a new disk cartridge has been loaded. The RL02 status information is transferred to the CPU in byte format. The format and bit significance of the RL02 status information contained in each byte transferred to the CPU are shown in Figure 2-11.

2.2.3.3 R80 Status – The R80 status information specifies to the CPU the sector over which the read/write heads were located when the status data were output from the R80 to the IDC, if an address plug is installed, whether the drive is operational or has a fault, and the operational condition of the drive. The R80 status information is transferred to the CPU in byte format. The format and bit significance of the R80 status information contained in each byte transferred to the CPU are shown in Figure 2-12.

2.2.4 Error Detection Information
Error detection information (error position and error pattern) can be provided to the CPU following a detected error in the data read from the disk. The error position data are transferred to the CPU via the BUS Y D12:D00 signal lines. These data specify to the CPU the position (location within the sector of data being read) of the first data bit of the data burst in which the read error was detected. The error pattern data are transferred to the CPU via the BUS Y D10:D00 signal lines. The error pattern data specifies to the CPU the correction pattern for the 11-bit data burst in which the read error was detected. During the error pattern data transfer, the BUS Y D12:D11 signal lines are set to a low.

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	(HS0)	Head Select bits. These bits are used at the R80 disk drive to select one of the fourteen read/write heads.
BUS Y D01	(HS1)	
BUS Y D02	(HS2)	
BUS Y D03	(HS3)	
BUS Y D04	-	
BUS Y D05	-	
BUS Y D06	-	
BUS Y D07	-	
BUS Y D08	-	
BUS Y D09	-	
BUS Y D10	-	
BUS Y D11	-	
BUS Y D12	-	
BUS Y D13	(CA)	Cylinder Address. This bit must be cleared for R80 head select command.
BUS Y D14	(HS)	Head Select. When set, specifies that this word is a R80 head select command.
BUS Y D15	(CS)	Control Select. This bit must be cleared for R80 head select command.
BUS Y D16	-	
↕	↕	
BUS Y D31	-	

* - Not used as part of R80 head select instruction.

Figure 2-6 R80 Head Select Command Data Format and Bit Significance

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	-	
BUS Y D01	-	
BUS Y D02	-	
BUS Y D03	-	
BUS Y D04	-	
BUS Y D05	-	
BUS Y D06	(RTZ)	Return to Zero. When set, used at R80 disk drive to initiate positioning read/write heads over cylinder 0.
BUS Y D07	-	
BUS Y D08	-	
BUS Y D09	-	
BUS Y D10	-	
BUS Y D11	-	
BUS Y D12	-	
BUS Y D13	(CA)	Cylinder Address. This bit must be cleared for a R80 recalibrate command.
BUS Y D14	(HS)	Head Select. This bit must be cleared for a R80 recalibrate command.
BUS Y D15	(CS)	Control Select. When set, specifies that this word is a control function word. This bit must be set for R80 recalibrate command.
BUS Y D16	-	
↕	↕	
BUS Y D31	-	

* - Not used as part of R80 recalibrate instruction.

Figure 2-7 R80 Recalibrate Command Data Format and Bit Significance

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	(SA0)	Sector Address. These bits specify the address of one of the 40 sectors of the RL02 cylinder to/from which data are to be written/read.
BUS Y D01	(SA1)	
BUS Y D02	(SA2)	
BUS Y D03	(SA3)	
BUS Y D04	(SA4)	
BUS Y D05	(SA5)	
BUS Y D06	(HS)	Head Select. This bit specifies which of the two RL02 read/write heads is selected; when set, indicates lower head; when cleared, upper head.
BUS Y D07	(CA0)	Cylinder Address. These bits specify the address of the RL02 cylinder (one of 512) over which the read/write heads are located.
BUS Y D08	(CA1)	
BUS Y D09	(CA2)	
BUS Y D10	(CA3)	
BUS Y D11	(CA4)	
BUS Y D12	(CA5)	
BUS Y D13	(CA6)	
BUS Y D14	(CA7)	
BUS Y D15	(CA8)	
BUS Y D16	-	
↕	↕	
BUS Y D31	-	

* - Not used as part of RL02 read/write data address.

Figure 2-8 RL02 Read/Write Data Address Data Format and Bit Significance

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	(SA0)	Sector Address. The bits specify the address of one of the 32 sectors of the R80 cylinder (one of 561) over which the read/write heads are located.
BUS Y D01	(SA1)	
BUS Y D02	(SA2)	
BUS Y D03	(SA3)	
BUS Y D04	(SA4)	
BUS Y D05	(HS0)	Head Select. These bits specify which one of the 14 R80 read/write heads is selected.
BUS Y D06	(HS1)	
BUS Y D07	(HS2)	
BUS Y D08	(HS3)	
BUS Y D09	(CA0)	Cylinder Address. These bits specify the address of the R80 cylinder (one of 561) over which the read/write heads are located.
BUS Y D10	(CA1)	
BUS Y D11	(CA2)	
BUS Y D12	(CA3)	
BUS Y D13	(CA4)	
BUS Y D14	(CA5)	
BUS Y D15	(CA6)	
BUS Y D16	(CA7)	
BUS Y D17	(CA8)	
BUS Y D18	(CA9)	
BUS Y D19	-	
↕	↕	
BUS Y D31	-	

* - Not used as part of R80 read/write data address.

Figure 2-9 R80 Read/Write Data Address Data Format and Bit Significance

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	(DRDY)	Drive Ready. When set, indicates that the presently selected drive is operational and ready to receive further commands.
BUS Y D01	(F0)	Function Select. These bits specify the function selected by the previous IDC control word input (See Figure 2-2).
BUS Y D02	(F1)	
BUS Y D03	(F2)	
BUS Y D04	-	
BUS Y D05	-	
BUS Y D06	(IE)	Interrupt Enable. Indicates state of IE bit of previous IDC control word input.
BUS Y D07	(CRDY)	Controller Ready. When set, indicates controller is ready to perform a function.
BUS Y D08	(DS0)	Drive Select. These bits indicate disk drive address specified by the previous IDC control word input.
BUS Y D09	(DS1)	
BUS Y D10	(OPI)	Operation Incomplete
BUS Y D11	(DCK)	Data Check Error
BUS Y D12	(DTL)	Data Late Error
BUS Y D13	-	
BUS Y D14	(DE)	Drive Error
BUS Y D15	(ERR)	Composite Error
BUS Y D16	(ATTN0)	Attention Bits. When set, indicates associated disk drive has completed a previously specified function and is asserting drive ready or that the associated disk drive is reporting an error.
BUS Y D17	(ATTN1)	
BUS Y D18	(ATTN2)	
BUS Y D19	(ATTN3)	
BUS Y D20	(ECS0)	ECC status. These bits define the status of the ECC comparison as shown in table at right.
BUS Y D21	(ECS1)	
BUS Y D22	(SSEI)	Skip Sector Error Inhibit. Indicates state of SSEI bit of previous IDC control word input. When set, indicates R80 skip sectoring was inhibited.
BUS Y D23	(SSE FLAG)	Skip Sector Error Flag. When set, indicates that the sector read contains a skip sector flag because it or a previous sector was a bad sector.
BUS Y D24	(IR)	Interrupt Request. When set, indicates that the IDC asserted an interrupt request.
BUS Y D25	(MTN)	Maintenance. When set, indicates maintenance function was specified by previous IDC control word input.
BUS Y D26	(R80)	R80. When set, indicates R80 disk drive selected by previous IDC control word input.
BUS Y D27	(ASSI)	Automatic Skip Sector Inhibit. When set, indicates automatic skip sectoring was inhibited by previous IDC control word input.
BUS Y D28	(WRT INH)	Write Inhibit. When set, indicates that timeout was disabled and writing to disk drives was inhibited by previous IDC control word input.
BUS Y D29	(R80 FORMAT)	R80 Format. When set, indicates that the previous IDC control word input specified an R80 write format function.
BUS Y D30	-	
BUS Y D31	-	

Error Bits. These bits are encoded as shown in the table at right to specify the type of error detected.

OPI	DCK	DTL	DE	ERR	Indicated Error
0	1	0	0	1	ECC/CRC error in disk data field
1	1	0	0	1	ECC/CRC error in disk header field
1	0	0	0	1	Timeout error
1	0	1	0	1	Header not found
0	0	1	0	1	Data buffer empty during write or full during read
0	0	0	1	1	Disk drive reporting an error

ECS 0	ECS 1	Status
0	0	No Error
0	1	Data Error
1	0	Noncorrectable Error
1	1	Correctable Error

* - Not used as part of IDC status word.

Figure 2-10 IDC Status Word Data Format and Bit Significance

Data Format (BUS Y Data Bits)	Bit Significance*	
BUS Y D00	(STA)	State. These bits define the operational state of the applicable RL02 disk drive. State is encoded as shown in table at right.
BUS Y D01	(STB)	
BUS Y D02	(STC)	
BUS Y D03	(BH)	Brush Home. When set, indicates that brushes are not over the disk recording area.
BUS Y D04	(HO)	Heads Out. When set, indicates that the read/write heads are over the disk recording area.
BUS Y D05	(CO)	Cover Open. When set, indicates cartridge access cover open or cartridge dust cover is not in place.
BUS Y D06	(HS)	Head Select. Indicates currently selected head. When set, indicates lower head; when cleared, indicates upper head.
BUS Y D07		
BUS Y D00	(DSE)	Drive Select Error. When set, indicates multiple drives responding to one address.
BUS Y D01	(VC)	Volume Check. When set, indicates a new cartridge may have been mounted since the last time the drive was selected.
BUS Y D02	(WGE)	Write Gate Error. When set, indicates that during RL02 write data mode, drive not ready to read/write, drive write protected, sector pulse occurred, and/or drive was reporting an error.
BUS Y D03	(SPE)	Spin Error. When set, indicates spindle speed not reached within required time or spindle speed is too high.
BUS Y D04	(SKTO)	Seek Time Out. When set, indicates read/write heads not located over specified cylinder within required time during seek state or read/write signal lost when disk drive was in lock-on state.
BUS Y D05	(WL)	Write Lock. When set, indicates write protect condition selected by disk drive WRITE PROT switch.
BUS Y D06	(CHE)	Current in Head Error. When set, indicates write current detected in read/write heads when disk drive is not in write data mode.
BUS Y D07	(WDE)	Write Data Error. When set, indicates disk drive in write data mode, but no write data is asserted within the required time.

STA	STB	STC	State
0	0	0	Load Cartridge
0	0	1	Spin-Up
0	1	0	Brush Cycle
0	1	1	Load Heads
1	0	0	Seek
1	0	1	Lock-On
1	1	0	Unload Heads
1	1	1	Spin-Down

* - Not used as part of RL02 status information.

Figure 2-11 RL02 Status Information Data Format and Bit Significance

Data Format (BUS Y Data Bits)		Bit Significance*	
BUS Y D00		(SEC0)	Sector Count. These bits specify the sector address over which the R80 read/write heads were located when the status information was output from the R80 disk drive to the IDC.
BUS Y D01		(SEC1)	
BUS Y D02		(SEC2)	
BUS Y D03		(SEC3)	
BUS Y D04		(SEC4)	
BUS Y D05		-	
BUS Y D06		-	
BUS Y D07		-	
BUS Y D00	BYTE 2	(FLT)	Fault. When set, indicates dc power fault, head select fault, write fault, write or read while off cylinder, or write attempted during read function.
BUS Y D01		(PLGV)	Plug Valid. When set, indicates a logic plug installed in the R80 disk drive operation control panel.
BUS Y D02		(SKE)	Seek Error. When set, indicates R80 unable to complete seek within 500 microseconds, read/write heads outside recording area, or illegal address detected.
BUS Y D03		(ONCY)	On Cylinder. When set, indicates disk drive read/write heads are located over a cylinder.
BUS Y D04		(DRDY)	Drive Ready. When set, indicates disk drive is up to speed, read/write heads are loaded, and no fault exists in disk drive.
BUS Y D05		(WTP)	Write Protect. When set, indicates R80 is in write protect mode (write protect mode selected using WRITE PROT switch on R80 disk drive).
BUS Y D06		-	
BUS Y D07		-	

* - Not used as part of R80 status information.

Figure 2-12 R80 Status Information Data Format and Bit Significance

2.2.5 Port Microinstruction Inputs

The port microinstruction inputs to the IDC are used to preset the IDC logic and to cause the transfer of data and information between the IDC and the CPU. The port microinstructions reside in the writable control store (WCS) module in the CPU. The CSR17 and CSR14:10 signal lines contain the port microinstruction applicable to the IDC. The format and bit significance of the port microinstruction inputs are shown in Figure 2-13. Table 2-1 lists the port microinstruction functions.

2.2.6 PORT INSTR Input

The CPU outputs a PORT INSTR signal to indicate that a valid port command is being applied on the CSR signal lines. The PORT INSTR signal and the port command remain active for an entire CPU microcycle (270 nanoseconds). A high PORT INSTR input with CSR17 of the port microinstruction set to a high (CSR17 high indicates that the port microinstruction is IDC specific) enables the IDC to decode the port microinstruction input and to preset the IDC logic, or to cause data transfers between the CPU and the IDC.

2.2.7 READ PORT and SEL ACC IN Inputs

The READ PORT and SEL ACC IN signals are used to cause the transfer of information and data from the IDC or FPA to the CPU. The SEL ACC IN signal indicates if the READ PORT input is IDC or FPA specific. (If SEL ACC IN is low, READ PORT is IDC specific.)

2.2.8 CPU P2 and PORT CLOCK Inputs

The CPU P2 and PORT CLOCK inputs provide the basic timing pulses for synchronizing the IDC operation with CPU operation. The PORT CLOCK input is the basic 90-nanosecond CPU clock. The CPU P2 input is the gated CPU clock phase 2 output of the CPU. The CPU P2 signal is normally high during the last 90 nanoseconds of the 270-nanosecond CPU microcycle. Figure 2-14 shows the timing relationship of the PORT CLOCK and CPU P2 inputs relative to the CPU microcycle.

2.3 IDC/R80 INTERFACE

The interface signals at the IDC/R80 interface are shown in Figure 2-1. The IDC/R80 interface signals input to the R80 disk drive from the IDC include R80 TAG 3:1, R80 TAG BUS 9:0, ACLO (POWER SEQUENCE PICK), GND (POWER SEQUENCE HOLD), R80 WRITE CLOCK, R80 WRITE DATA, and R80 INITIALIZE. The IDC/R80 interface signals output from the R80 disk drive to the IDC include R80 SECTOR COUNT 1, 2, 4, 8, and 16, R80 FAULT, R80 PLUG VALID, R80 SEEK ERROR, R80 ON CYLINDER, R80 DRIVE READY, R80 WRITE PROTECT, R80 SELECT ADRS 1 and 2, R80 INDEX PULSE, R80 SECTOR PULSE, R80 SERVO CLOCK, R80 READ CLOCK, and R80 READ DATA. All of the signals at the IDC/R80 interface are discussed in detail in Paragraphs 2.3.1 through 2.3.8.

2.3.1 R80 TAG 3:1 and R80 TAG BUS 9:0

The R80 TAG 3:1 and R80 TAG BUS 9:0 signal lines are used to transmit disk drive control signals from the IDC to the R80 disk drive. These signal lines are used to position the read/write heads over the desired cylinder, to select one of the fourteen read/write heads, and to initiate a disk drive read, write, or recalibrate function.

The R80 TAG 3:1 inputs to the R80 disk drive are used to identify the parallel inputs applied via the R80 TAG BUS 9:0 inputs (see Table 2-2). When the R80 TAG 1 signal is asserted, it identifies to the R80 disk drive that the R80 TAG BUS 9:0 inputs contain a binary-coded cylinder address and initiates the R80 disk drive seek function (repositions the R80 read/write heads over the cylinder having the address specified by the R80 TAG BUS 9:0 inputs).

When the R80 TAG 2 signal is asserted, it identifies to the R80 disk drive that the R80 TAG BUS 4:0 inputs contain binary coded R80 read/write head selection information and initiates selection of one of the fourteen read/write heads based on the state of the R80 TAG BUS 4:0 inputs.

Port Microinstruction Bits	Bit Significance
CSR 17	Port Device Select. This bit contains the address of the port device for which the port microinstruction is intended. The address for the IDC is CSR 17=1.
CSR 14 CSR 13	Command Identity. These bits specify the function of the command bits (CSR12:CSR10): read (transfer information or data from IDC to CPU), write (transfer information or data from CPU to IDC), or control (preset the IDC logic). These bits are encoded as follows:

CSR 14	CSR 13	Function
0	0	Read
0	1	Write
1	0	Control

CSR 12 CSR 11 CSR 10	Command Bits. These bits enable the IDC data paths that cause the transfer of information or data between the IDC and CPU, or preset the IDC logic. The command bits are decoded in the IDC to initiate the function(s) specified in Table 2-1.

Figure 2-13 Port Microinstructions Format and Bit Significance

Table 2-1 Port Microinstruction Functions

Command Identity (See CSR 13:14)	CSR 12	CSR 11	CSR 10	Function Decode	Function Description
Write	0	0	0	WRITE CSR	Load IDC control word (Figure 2-2) from the CPU into the IDC.
Write	0	0	1	WRITE DAR	Load one of the following disk drive control words from the CPU into the IDC: RL02 get status command (Figure 2-3), RL02 seek command (Figure 2-4), RL02 read/write data address (Figure 2-8), R80 seek command (Figure 2-5), R80 head select command (Figure 2-6), R80 recalibrate command (Figure 2-7), or R80 read/write data address (Figure 2-9).
Write	0	1	0	WRITE DATA BYTE	Load one data byte (BUS Y bits D00 through D07) from the CPU into the IDC.
Write	0	1	1	WRITE DATA WORD	Load one data word (BUS Y bits D00 through D31) from the CPU into the IDC.
Read	0	0	0	READ CSR	Output to CPU the IDC status word (Figure 2-10).
Read	0	0	1	READ DAR	Output to CPU the current RL02 read/write data address (Figure 2-8) or the current R80 read/write data address (Figure 2-9).
Read	0	1	0	READ DATA BYTE	Output to CPU one data byte (BUS Y bits D00 through D07) from IDC to CPU. Usually two successive read data byte commands are used to output to the CPU the RL02 status information or the R80 status information (Figure 2-11 or 2-12, respectively).

Table 2-1 Port Microinstruction Functions (Cont)

Command Identity (See CSR 13:14)	CSR 12	CSR 11	CSR 10	Function Decode	Function Description
Read	0	1	1	READ DATA WORD	Output to CPU one data word (BUS Y bits D00 through D31).
Read	1	0	0	READ PATTERN	Output to CPU a 13-bit word (BUS Y bits D00 through D12) that contains the 11-bit data burst in which a read error occurred.
Read	1	0	1	READ POSITION	Output to CPU a 13-bit word (BUS Y bits D00 through D12) that contains the address of the first bit of the data burst within which a read error occurred.
Control	0	0	0	CLEAR FIFO CNTR	Resets the counter that controls sequential loading and unloading of data from the IDC data buffers.
Control	0	0	1	RESET BR	Resets the UBUS BR5 interrupt request output of the IDC.
Control	0	1	1	CLEAR IDC	Presets the IDC and R80 disk drive. This function is also initiated by the ACLO input.
Control	1	0	0	SET AUTOMODE	Presets the conditions that allow successive data words to be gated to the CPU without issuing a READ DATA WORD port microinstruction for each data longword to be gated. However, a READ PORT signal is required for gating each data longword to the CPU.
Control	1	0	1	CLEAR AUTOMODE	Deselects the automode function.

Table 2-1 Port Microinstruction Functions (Cont)

Command Identity (See CSR 13:14)	CSR 12	CSR 11	CSR 10	Function Decode	Function Description
Control	1	1	0	SELECT FIFO A	Selects one of the two IDC data buffers to be used in the transfer of data between the IDC and the CPU.
Control	1	1	1	SELECT FIFO B	Selects one of the two IDC data buffers to be used in the transfer of data between the IDC and the CPU

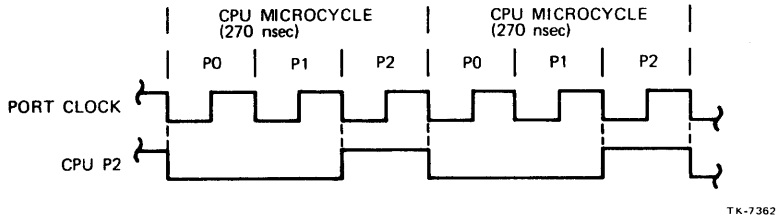


Figure 2-14 Timing Relationship of PORT CLOCK and CPU P2 Inputs to IDC

Table 2-2 R80 TAG Bus Bit Decoding

	R80 TAG 1 Asserted	R80 TAG 2 Asserted	R80 TAG 3 Asserted
R80 TAG Bus Bit	Cylinder Address (Binary Coded)	Read/Write Head Select (Binary Coded)	Control Select*
0	1	1	Write gate
1	2	2	Read gate
2	4	4	Not used
3	8	8	Not used
4	16	Not used	Not used
5	32	Not used	Not used
6	64	Not used	Recalibrate
7	128	Not used	Not used
8	256	Not used	Not used
9	512	Not used	Not used

* Only one of the ten TAG bus bits may be asserted at a time when R80 TAG 3 is asserted.

When the R80 TAG 3 signal is asserted, it identifies to the R80 disk drive that the TAG BUS 9:0 inputs specify a control select signal. The control select signals input to the R80 disk from the IDC include the R80 recalibrate write gate and read gate commands. Assertion of the R80 TAG 3 and R80 TAG BUS 6 inputs initiates the R80 recalibrate function (positions the R80 read/write heads over the cylinder having the address of 0). Assertion of the R80 TAG 3 and R80 TAG BUS 0 inputs enables the R80 write gate function. Assertion of the R80 TAG 3 and R80 TAG BUS 1 inputs enables the R80 read gate function.

2.3.2 ACLO, GND, and R80 INITIALIZE

The ACLO (POWER SEQUENCE PICK) input to the R80 disk drive is low when the VAX-11/730 system is operating normally. However, when the VAX-11/730 system experiences a low ac line level condition, the ACLO input to the IDC is asserted. The ACLO input is buffered by the IDC and asserted to the R80 disk drive as a high ACLO (POWER SEQUENCE PICK) signal. A high POWER SEQUENCE PICK signal input to the R80 disk drive causes the disk drive to spin down and inhibits any read/write operations. The POWER SEQUENCE HOLD input to the R80 disk drive is used to inhibit spinup of the drive while another R80 disk drive is in the spinup state. Since only one R80 disk drive is connected to the IDC, the POWER SEQUENCE HOLD input is clamped at ground in the IDC.

The R80 INITIALIZE input to the R80 disk drive is initiated by the IDC either in response to a CLEAR IDC port microinstruction input from the CPU or in response to the DCLO input during initial powerup or powerup following an input power interruption to the VAX-11/730. The R80 INITIALIZE input to the R80 disk drive causes the read/write heads to be deselected and to be positioned over cylinder 0.

2.3.3 R80 WRITE DATA and R80 WRITE CLOCK

The R80 WRITE DATA input to the R80 disk drive is used to apply serially the data to be written on the disk. The R80 WRITE CLOCK signal is generated by the IDC from the R80 SERVO CLOCK input to the IDC. The R80 WRITE CLOCK signal strobes each data bit applied via the R80 WRITE DATA input into the R80 disk drive.

The R80 WRITE DATA applied to the disk drive is in the format illustrated in Figure 2-15. Figure 2-15 also shows the timing relationship of the R80 WRITE DATA and R80 WRITE CLOCK outputs from the IDC during the transfer of one sector of data from the IDC to the R80 disk drive.

2.3.4 R80 SECTOR COUNT 1, 2, 4, 8, and 16

The R80 SECTOR COUNT 1, 2, 4, 8, and 16 inputs to the IDC from the R80 disk drive provide binary coded sector address information. The SECTOR COUNT inputs identify the correct sector from the 32 sectors of the selected cylinder over which the read/write heads are located. The SECTOR COUNT inputs change state at the leading edge of each R80 SECTOR or R80 INDEX PULSE. The SECTOR COUNT inputs to the IDC are reset to zero by the R80 INDEX PULSE and incremented by each R80 SECTOR PULSE.

2.3.5 R80 FAULT, R80 PLUG VALID, R80 SEEK ERROR, R80 ON CYLINDER, R80 DRIVE READY, and R80 WRITE PROTECT

These signal inputs to the IDC are used to indicate the operational or fault status of the R80 disk drive. The significance of each signal is detailed in Figure 2-12, which illustrates and discusses the makeup and bit significance of the R80 status information transferred to the CPU. The R80 FAULT, R80 PLUG VALID, R80 ON CYLINDER, and R80 DRIVE READY inputs to the IDC from the R80 disk drive are used to specify to the IDC any changes in R80 disk drive status (that the operational function requested of the disk drive was completed successfully or that a fault condition has developed).

2.3.6 R80 SELECT ADDRESS 1 and 2

These signal lines specify to the IDC the binary-coded unit number of the disk drive. This number is selectable by selection of the logic plug installed on the R80 operator panel.

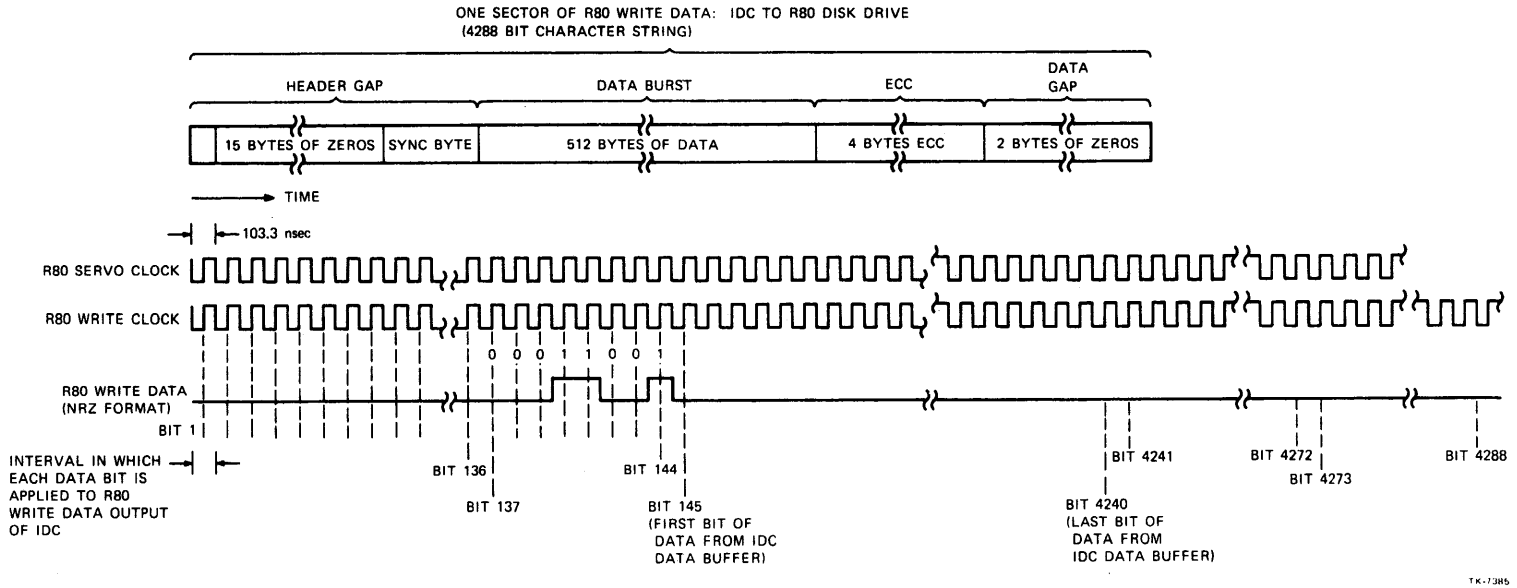


Figure 2-15 R80 Write Data Format and Data Transfer Timing: IDC to R80

2.3.7 R80 INDEX PULSE and R80 SECTOR PULSE

The R80 INDEX PULSE and R80 SECTOR PULSE inputs to the IDC are used to indicate the beginning of each sector of the R80 cylinder track. The R80 INDEX PULSE occurs once per R80 cylinder revolution with the leading edge of the pulse indicating the beginning of sector 0. An R80 SECTOR PULSE marks the beginning of each of the remaining 31 sectors for each R80 cylinder revolution. Figure 2-16 shows the timing relationship of the R80 SECTOR and R80 INDEX PULSES.

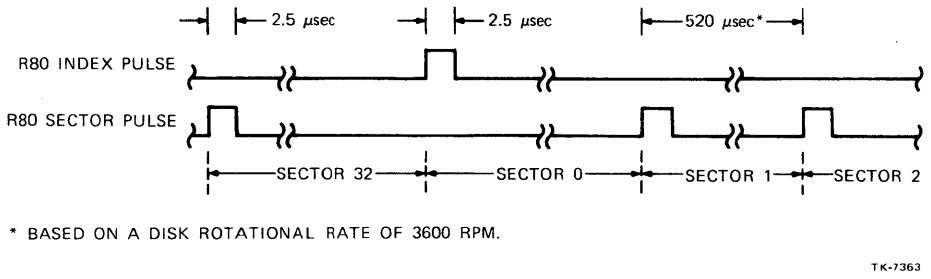


Figure 2-16 R80 Sector Pulse and Index Pulse Timing

2.3.8 R80 READ DATA and R80 READ CLOCK

The R80 READ DATA output is used to apply to the IDC the data read from the disk. The R80 READ CLOCK output is synchronized with the R80 READ DATA output to provide a timing pulse that defines the beginning of each interval in which each bit of the read data is applied.

The R80 READ DATA applied to the IDC is in the format illustrated in Figure 2-17. Figure 2-17 also shows the timing relationship of the R80 READ DATA and R80 READ CLOCK outputs of the R80 disk drive.

2.4 IDC/RL02 INTERFACE

The signals at the IDC/RL02 interface are shown in Figure 2-1. The IDC/RL02 interface signals input to the RL02 disk drive include RL DRIVE COMMAND, RL DRIVE DATA, and RL SYSTEM CLOCK. The IDC/RL02 interface signals output from the RL02 disk drive include RL DRIVE ERROR, RL DRIVE READY, RL STATUS, RL STATUS CLOCK, RL SECTOR PULSE and RL READ DATA. All of the signals at the IDC/RL02 interface are discussed in Paragraphs 2.4.1 through 2.4.9.

2.4.1 RL DRIVE COMMAND and RL SYSTEM CLOCK

The RL DRIVE COMMAND signal line is used to transfer serially a RL02 get status command or a RL02 cylinder difference word from the IDC to the RL02 disk drive. The RL02 get status command initiates the transfer of RL02 status data from the RL02 to the IDC. The RL02 cylinder difference word specifies to the RL02 the data required to reposition the read/write heads over the desired cylinder and selects the read/write head to be used. The RL02 get status command and RL02 cylinder difference word inputs are in a 16-bit serial data format and are transferred to the RL02 by the RL SYSTEM CLOCK. The RL SYSTEM CLOCK, a 4.1 megahertz clock signal generated by the IDC, transfers the RL02 cylinder difference word to the RL02 at a rate of 243.9 nanoseconds per bit. The RL SYSTEM CLOCK signal input to the RL02 disk drive is used also to synchronize the operation of the RL02 disk drive with the IDC.

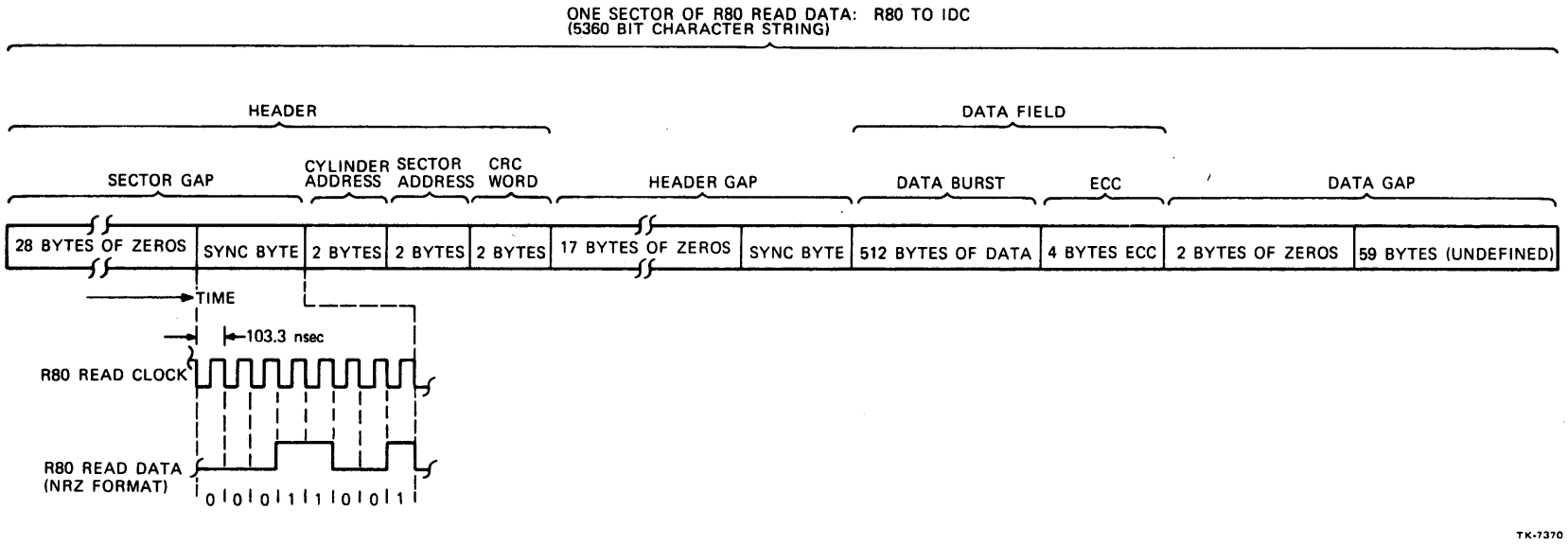


Figure 2-17 R80 Read Data Format and Data Transfer Timing: R80 to IDC

The structure and bit significance of the get status and cylinder difference word inputs to the RL02 disk drive are discussed in Figures 2-3 and 2-4, respectively. The data words shown in Figures 2-3 and 2-4 are serialized in the IDC and applied to the RL02 disk drive via the RL DRIVE COMMAND signal line. The bit identified as D00 in Figures 2-3 and 2-4 corresponds to the first bit of the 16 bits transferred to the RL02.

2.4.2 RL DRIVE SELECT 0 and 1

The RL DRIVE SELECT 0 and 1 inputs enable selection of one of the four RL02 disk drives that may be connected to the IDC. The RL DRIVE SELECT 0 and 1 inputs to the RL02 disk drives are generated by the IDC in response to the Drive Select bits of the IDC control word (Figure 2-2) input to the IDC from the CPU. Assertion of the Drive Select bits enables the selected drive to generate/respond to the signals at the IDC/RL02 interface.

2.4.3 POWER FAIL (ACLO)

This signal input to the RL02 disk drives is asserted low whenever a low line level or loss of the primary facility power input is detected. Assertion of the POWER FAIL signal causes all of the RL02 disk drives connected to the IDC to cycle down. When the POWER FAIL signal is deasserted (returns to a high), the RL02 disk drives spin up and the read/write heads are loaded and positioned over cylinder 0.

2.4.4 RL WRITE GATE and RL WRITE DATA

The RL WRITE GATE signal enables the write circuits in the selected RL02 disk drive. The data to be written on the selected RL02 disk are applied in a serial format via the RL WRITE DATA signal line. The data to be written are encoded in Modified Frequency Modulation (MFM) form. The RL WRITE DATA applied to the RL02 disk drive are in the format shown in Figure 2-18. Figure 2-18 also illustrates the timing relationship of the RL WRITE DATA and RL WRITE GATE outputs of the IDC during the transfer of one sector of data from the IDC to the RL02 disk drive.

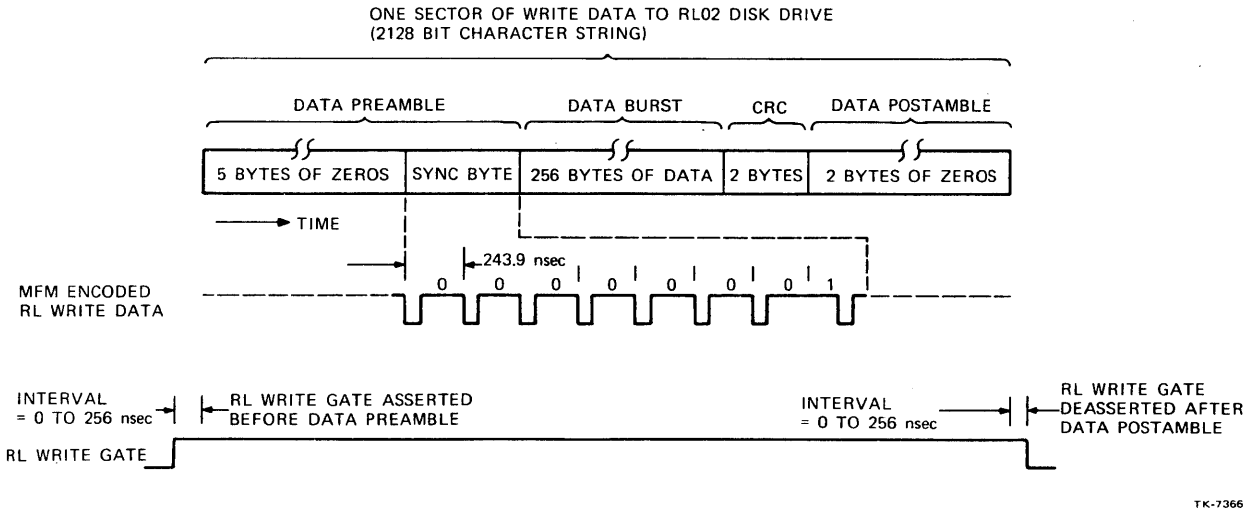


Figure 2-18 RL Write Data Format and Data Transfer Timing: IDC to RL02

2.4.5 RL DRIVE READY

The RL DRIVE READY input to the IDC is asserted high to indicate that the disk drive has successfully executed the previously asserted drive command (the read/write heads are located over the desired cylinder and loaded) and is ready to receive further drive commands or to write or read data to or from the disk. The RL DRIVE READY signal is deasserted after receiving a RL DRIVE command, on detection of an operational error within the disk drive, or when RL WRITE GATE is asserted (when data are being written on the disk).

2.4.6 RL DRIVE ERROR

The RL DRIVE ERROR input to the IDC is asserted high to indicate that the selected RL02 disk drive has developed an error condition. When the RL DRIVE ERROR output of the RL02 disk drive is asserted, the DRIVE READY output is deasserted. The type of error causing the assertion of RL DRIVE ERROR may be determined by examining the RL STATUS output of the applicable RL02 disk drive (Paragraph 2.4.7).

2.4.7 RL STATUS and RL STATUS CLOCK

The RL STATUS and RL STATUS CLOCK outputs of the RL02 disk drive are enabled by asserting an RL02 get status command to the disk drive via the RL drive command signal line. On receipt of the get status command, the selected disk drive transfers serially 16 bits of status information to the IDC via the RL STATUS signal lines. The format and bit encoding of the 16-bit RL status word input to the IDC are illustrated in Figure 2-19. The significance of each bit is as specified in Figure 2-11. The RL STATUS CLOCK output of the disk drive is derived from the 4.1 megahertz RL SYSTEM CLOCK input to the disk drive. The RL STATUS CLOCK output of the disk drive (a 4.1 megahertz clock) is asserted to the IDC in synchronization with each bit of the status information. The RL STATUS CLOCK output remains enabled until a new RL drive command is asserted or the disk drive is deselected.

2.4.8 RL SECTOR PULSE

The RL SECTOR PULSE signal input to the IDC is used to indicate the beginning of each of the 40 sectors of each RL02 cylinder track. The selected RL02 disk drive asserts a high 45 + 10 microsecond pulse once every 625 microseconds. The leading edge of the pulse indicates the beginning of a sector.

2.4.9 RL READ DATA

The data recorded on the disk are applied serially to the IDC via the RL READ DATA signal line. This signal line applies the recorded data to the IDC whenever the disk drive is selected (Paragraph 2.4.2) and the disk drive asserting a high RL DRIVE READY output (the drive is not performing a seek function, and does not have a detected error). The read data transferred via the RL READ DATA signal line are encoded in MFM form as shown in Figure 2-20. The data are transferred from the RL02 to the IDC at a 4.1 megahertz rate (243.9 nanoseconds per bit).

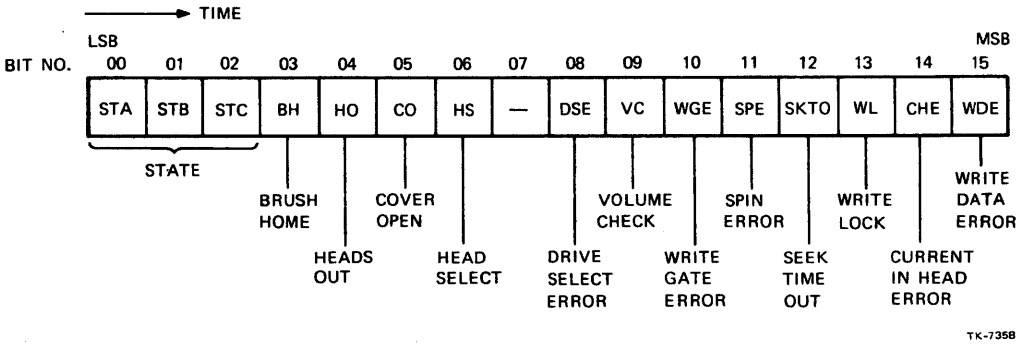


Figure 2-19 Format and Bit Significance of RL02 Status Information Transfer: RL02 to IDC

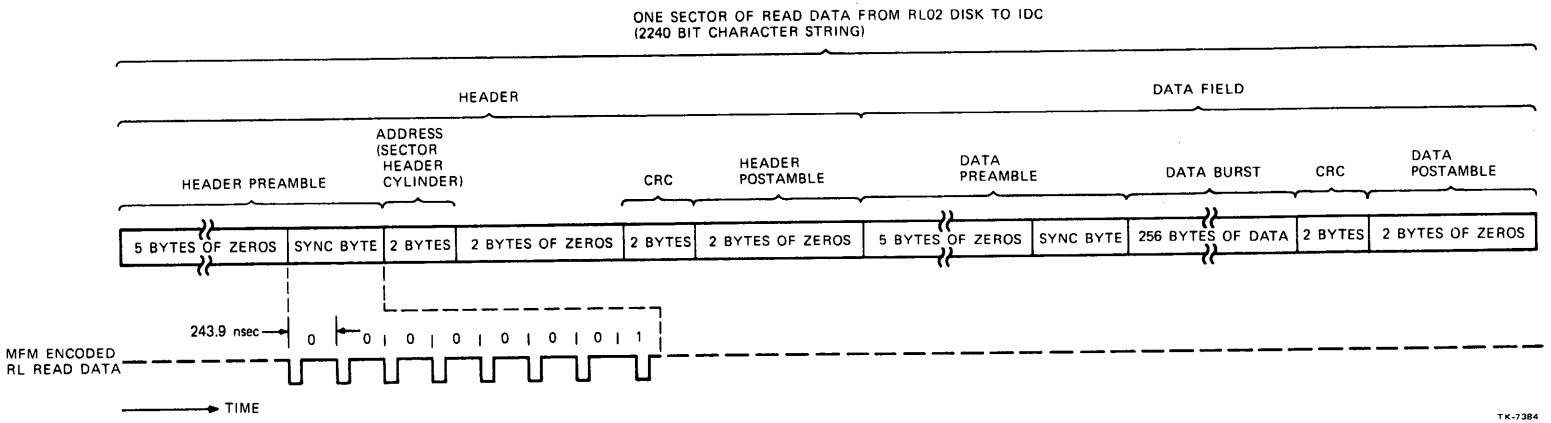


Figure 2-20 RL Read Data Format and Data Transfer Timing: RL02 to IDC

CHAPTER 3
THEORY OF OPERATION

3.1 IDC FUNCTIONS

The IDC and associated disk drive(s) make up the RB730 disk subsystem. The IDC interfaces the VAX-11/730 CPU with up to four RL02 disk drives or one R80 disk drive and up to three RL02 disk drives. The IDC executes the functions specified by the CPU to cause storage and retrieval of data from the disk drives of the RB730 disk subsystem. Table 3-1 lists the functions that can be specified with the IDC control words and describes the purpose of each function. It also lists the required inputs (disk drive control words, address information, and data) for each function.

Table 3-1 IDC Functions

Function Specified by IDC Control Word from CPU*	Required Inputs from CPU*	Purpose
Seek (RL02)	RL02 Seek Command	Controls positioning of the selected RL02 disk drive read/write heads over the desired cylinder track and enables the desired read/write head.
Seek (R80)	R80 Seek Command	Controls positioning of the R80 disk drive read/write heads over the desired cylinder.
Seek (R80)	R80 Head Select Command	Enables one of the fourteen read/write heads in the R80 disk drive.
Seek (R80)	R80 Recalibrate Command	Controls positioning of the R80 read/write heads over cylinder 0.
Get Status (for RL02)	RL02 Get Status Command	Controls gating the status information from the selected RL02 disk drive and storing it in the IDC data buffer.
Get Status (For R80)	None. (Information for selecting R80 status information is contained in IDC control word.)	Controls gating the status information from the R80 disk drive and storing the status information in the IDC data buffer.

* The format and bit significance of the IDC control word and the specified required inputs to the IDC are discussed in Chapter 2.

Table 3-1 IDC Functions (Cont)

Function Specified by IDC Control Word from CPU*	Required Inputs from CPU*	Purpose
Read Header	None. (Information for selecting disk drive from which header is to be read is contained in IDC control word.)	Controls reading from the selected disk drive the header information from the first sector encountered and storing it in the IDC data buffer.
Write Data	Read/Write Data Address. (CPU must load IDC data buffer with data to be written)	Controls writing of the data contained in the IDC data buffer at the specified read/ write data address of the selected disk drive.
Read Data	Read/Write Data Address	Controls reading from the selected disk drive the data from the specified read/write data address and storing of the data in the IDC data buffer.
Read Data Without Header Check	None. (Information for selecting disk drive from which data are to be re trieved is specified as part of the IDC control word.)	Controls reading from the selected disk drive the data from the first sector encountered and storing the data in the IDC data buffer.
Write Check	Read/Write Data Address. (CPU must load IDC data buffer with comparison data.)	Controls reading from the selected disk drive the data from the specified read/write data address and comparison of data read from memory with data contained in the data buffers.
Write Format (Used only with R80 disk drive)	Header Data. Performed after read/write heads of the selected disk drive have been positioned over the cylinder (CPU must load IDC data buffer with the header information for all 32 sectors of the cylinder.)	Controls writing of new header data from the IDC data buffer into each of the 32 sectors of the applicable R80 cylinder.
Maintenance	As specified in microdiagnostic routines.	Places the IDC in the maintenance mode such that the IDC logic may be exercised by microdiagnostic routines designed to detect faults or verify operational status of the IDC hardware.

* The format and bit significance of the IDC control word and the specified required inputs to the IDC are discussed in Chapter 2.

3.2 OVERALL IDC OPERATION

The IDC operates under CPU control. The CPU loads the required inputs (disk drive control word, address information, and/or data) and IDC control word needed to initiate each function of the IDC. Once an IDC function is initiated (when the IDC control word is loaded), the IDC operation is controlled by a microcontroller on the IDC. After the function has been completed or an error is detected, the IDC generates and asserts an interrupt signal to the CPU. The CPU then takes control of the IDC operation to transfer the desired information or data from the IDC to the CPU, or to load the IDC with the required input(s) and IDC control word needed to initiate another IDC function. When the IDC is not performing a function specified by the CPU, it operates in the idle mode. In this mode, the operation of the IDC is controlled by the microcontroller, which samples the operational status of the disk drive(s) and generates and asserts an interrupt to the CPU if an operational status change is detected.

3.2.1 Initiating IDC Functions

Each of the IDC functions listed in Table 3-1 is initiated under CPU control. The required inputs and IDC control word needed to initiate each IDC function are asserted to the IDC via the CPU Y BUS and are loaded into the IDC by the port microinstructions asserted to the IDC. The IDC decodes the port microinstructions and generates the control signals used to preset the IDC logic or to load the IDC registers and data buffers.

3.2.1.1 Loading Required Inputs

a. Disk Drive Control Word and Address Information – The required disk drive control word (RL02 seek command, RL02 get status command, R80 seek command, R80 head select command, or R80 recalibrate command) or address information (RL02 read/write address or R80 read/write address) is loaded into the IDC by asserting the applicable disk drive control word or address information to the IDC via the CPU Y BUS and simultaneously asserting a WRITE DAR port microinstruction (see Table 2-1) and a PORT INSTR input to the IDC. (A detailed discussion of how the disk drive control word and address information are loaded into the IDC is provided in Paragraph 3.5.13.3.)

b. Data – The required data input to the IDC is loaded into one of the two data buffers. Each data buffer has the capacity to store one full sector of data (512 bytes of R80 data or 256 bytes of RL02 data). When data are a required input, the CPU must load a full sector of data. If a partial sector is to be loaded, the CPU must load the rest of the sector with zeros. The data to be loaded into the data buffer(s) may be in either byte or longword format.

   Before the required data are loaded into the IDC, the CPU must assert a FIFO SEL port microinstruction (see Table 2-1) to select the data buffer to which the data are to be loaded. The CPU causes loading of each data byte or data longword into the selected data buffer by asserting the correct WRITE DATA BYTE or WRITE DATA WORD port microinstruction (see Table 2-1) and PORT INSTR input signal to the IDC and simultaneously asserting the data byte or data longword to be loaded via the CPU Y BUS. (A detailed discussion of how the data are loaded into the IDC is provided in Paragraph 3.5.13.6.)

3.2.1.2 Loading the IDC Control Word – The IDC control word is loaded into the IDC by asserting the IDC control word onto the CPU Y BUS and simultaneously asserting a WRITE CSR port microinstruction (see Table 2-1) and PORT INSTR input signal to the IDC. (A detailed discussion of how the IDC control word is loaded into the IDC is provided in Paragraph 3.5.13.1.)

3.2.2 IDC Operation

Each of the IDC functions listed in Table 3-1 is initiated when the CPU loads the applicable IDC control word into the IDC. The IDC control word input specifies the function to be executed, the address of the disk drive to be used, and whether an interrupt is to be generated at the completion of the specified function. If the R80 disk drive is to be used, the IDC control word also indicates if skip sectoring is to be enabled.

When the IDC control word is loaded into the IDC, the address bits are used to enable the appropriate RL02 disk drive (if an RL02 disk drive is specified) and to condition the IDC for operation with an RL02 or the R80 disk drive. The function bits of the IDC control word indicate to the IDC the function to be performed and are used to preset the IDC microcontroller. The operational sequence performed by the IDC is initiated by the CRDY bit of the IDC control word. (A detailed discussion of IDC operation during each function is presented in Paragraph 3.4.)

After the IDC has completed the function specified by the IDC control word, the IDC generates and asserts the applicable interrupt to the CPU. Then, if applicable, the IDC enters the idle mode of operation.

After the IDC has asserted the proper interrupt to the CPU, the CPU may specify another IDC function or, if data or information was requested by the previously specified IDC function, assert the applicable port microinstructions to transfer the requested information or data from the IDC to the CPU.

3.2.3 Transfer of Information and Data from IDC to CPU

The transfer of information (IDC status information, disk drive information, CRC/ECC error detection information, and current address information) and data from the IDC to the CPU is controlled by the CPU. To transfer information and data from the IDC to the CPU requires that the CPU assert the proper port microinstruction input(s) followed during a later CPU microcycle by a READ PORT signal. The IDC decodes the port microinstruction input(s) and generates the enable signals that make available to the CPU the requested information or data. The following READ PORT signal is used to enable the requested information or data to be asserted to the CPU via the CPU Y BUS.

3.2.3.1 IDC Status Information Transfer (IDC to CPU) – The CPU causes transfer of the IDC status word (Figure 2-10) from the IDC to the CPU by asserting a READ CSR port microinstruction (see Table 2-1), followed during a later CPU microcycle by a READ PORT signal. The IDC decodes the port microinstruction and generates the control signals required to make the IDC status word available for transfer to the CPU. The READ PORT input to the IDC port control logic is used to assert the IDC status word to the CPU via the CPU Y BUS. (A detailed discussion of how the IDC status information is transferred from the IDC to the CPU is provided in Paragraph 3.5.13.2.)

3.2.3.2 Disk Drive Status Information Transfer (IDC to CPU) – The two bytes of disk drive status information (Figure 2-11, RL02; Figure 2-12, R80) read from the disk drives by the IDC during a get status function are stored in the IDC data buffer. The CPU causes the transfer of the disk drive status information by asserting two READ DATA BYTE port microinstructions. Following each READ DATA BYTE port microinstruction, the CPU asserts a READ PORT signal. The IDC decodes each of the READ DATA BYTE port microinstructions and generates the control signals to cause the transfer of a single byte of data from the data buffer to the data output register, where it is available for transfer to the CPU.

Each of the READ PORT inputs to the IDC control logic is used to assert the byte of disk drive status information to the CPU via the CPU Y BUS. (A detailed discussion of how the disk drive status information is transferred from the IDC to the CPU is provided in Paragraph 3.5.13.7.)

3.2.3.3 ECC/CRC Error Detection Information Transfer (IDC to CPU) – The CPU causes the transfer of the ECC POSITION or ECC PATTERN from the IDC to the CPU by asserting the applicable READ POSITION or READ PATTERN port microinstruction (see Table 2-1), followed during a later CPU microcycle by a READ PORT signal. The IDC port control logic decodes the port microinstruction and generates the control signal required to make the ECC POSITION or ECC PATTERN information available for transfer to the CPU. The READ PORT input is used to assert the ECC POSITION or ECC PATTERN information to the CPU via the CPU Y BUS. (A detailed discussion of how the error detection information is transferred from the IDC to the CPU is provided in Paragraph 3.5.13.5.)

3.2.3.4 Current Address Information Transfer (IDC to CPU) – The CPU causes the transfer of the current address information to the CPU by asserting a READ DAR port microinstruction (see Table 2-1) followed during a later CPU microcycle by a READ PORT signal. The IDC port control logic decodes the port microinstruction and generates the control signals required to make the current address information available for transfer to the CPU. The READ PORT input to the IDC port control logic is used to assert the current read/write data address to the CPU via the CPU Y BUS. (A detailed discussion of how the current address information is transferred from the IDC to the CPU is provided in Paragraph 3.5.13.4.)

3.2.3.5 Data Transfer (IDC to CPU) – The CPU controls the transfer of data from the IDC buffers to the CPU. The data contained in the IDC data buffers may be transferred in either byte or longword format. The CPU causes the transfer of a data byte or single data longword by asserting a READ DATA BYTE or READ DATA WORD port microinstruction followed during a later CPU microcycle by a READ PORT signal. The IDC decodes the port microinstruction and generates the control signals that make available for transfer to the CPU a single data byte or a series of four contiguous data bytes arranged in a longword format. The READ PORT signal input to the IDC enables the data byte or data longword to be transferred to the CPU via the CPU Y BUS.

For transferring a series of data longwords, the CPU presets the IDC using an AUTOMODE port microinstruction, followed by a single READ DATA WORD port microinstruction. Presetting the IDC with the AUTOMODE and READ DATA WORD port microinstructions allows a series of data longwords to be transferred with a series of READ PORT signals (each successive READ PORT input signal causes the transfer of successive data longwords). (A detailed discussion of how the data are transferred from the IDC to the CPU is provided in Paragraph 3.5.13.7.)

3.3 OVERALL IDC LOGIC FAMILIARIZATION

Figure 3-1 is a block diagram of the IDC. Each block represents a grouping of components having the operational characteristics identified in that block.

3.3.1 IDC Port Control Logic

The IDC port control logic operates under CPU control. The CPU uses port microinstruction inputs to get control of the IDC. The port microinstructions are applied to the IDC port control logic via the CSR17 and CSR14:10 signal lines. When these signal lines contain a valid port device (IDC or FPA) instruction, the CPU also asserts a high PORT INSTR signal. When the PORT INSTR input is high, the IDC port control logic decodes the port microinstruction and generates the control signals to preset the IDC logic, to load the required input(s) or IDC control word into the IDC, or to make information or data contained in the IDC available for transfer to the CPU.

3.3.2 Microcontroller

The microcontroller, a combination of conditional addressing logic and associated PROMs, generates the proper sequence of microwords that control the operation of the IDC in causing the function specified by the IDC control word. Branch condition inputs from the control status register (CSR), data buffer and data register control logic, header/data comparator, and ECC/CRC logic determine the sequence of microwords generated by the microcontroller. Timing for the sequence of microwords generated is controlled by the sequence clock output of the clock control.

3.3.3 Y-Bus Transceivers

All control words, address information, error detection information, status information, and data are transferred between the CPU and IDC via the Y-bus transceivers. The READ IDC input to the transceivers is used to control the direction of signal flow. The READ IDC input is generated from the READ PORT input from the CPU. A low READ IDC signal enables the signals at the IDC bus I/O to be asserted on the CPU Y-bus. A high READ IDC input enables the signals on the CPU Y-bus to be asserted on the IDC bus I/O.

3.3.4 Disk Address Register

The disk address register is loaded under CPU control with the required disk drive control word or read/write data address. The read/write data address of the disk address register may be incremented by the microcontroller to update the read/write data address information as additional contiguous sectors of data are written or read. The contents of the disk address register may be transferred from the IDC to the CPU under CPU control.

3.3.5 Data Input Register, Data Buffer and Data Register Control Logic, Data Output Register, Read Data Tristate Drivers, and R80 Multiplexer

The data input register and the data buffer and data register control logic operate under CPU control to cause loading of the required data inputs into the IDC data buffers. The data output register and the data buffer and data register control logic operate under CPU control to cause the transfer of the disk drive status information, header information, or data contained in the data buffers from the IDC to the CPU. During a write function, the data buffer and data register control logic operates from microcontroller inputs to cause the transfer of data from the data buffers into the data shift register. During a read function, the data buffer and data register control logic, read data tristate drivers, and R80 multiplexer operate from microcontroller inputs to load the data buffers with the proper header information, status information, or data from the applicable disk drive.

3.3.6 Control Status Register

The control status register (CSR) is loaded under CPU control with the IDC control word. The CSR also operates under CPU control to cause the transfer of the IDC status word (the current IDC control word contained in the CSR and a summary of the current status of the IDC and disk drives) from the IDC to the CPU.

The CSR asserts the initial branch conditions (F0, F1, and F2) and the start signal (CRDY) to the microcontroller. The CSR also controls selection of the applicable disk drive and enables the appropriate read data paths of the IDC. Status information from the disk drives and from the IDC header/data comparator and ECC/CRC logic is asserted to the CSR, which makes this information available to the CPU in the form of the IDC status word output.

When the function specified by the IDC control word is completed or has been halted due to an error, the CSR operates from microcontroller inputs to generate and assert the applicable interrupt (UBUS BR5 or PORT XFER REQ) to the CPU.

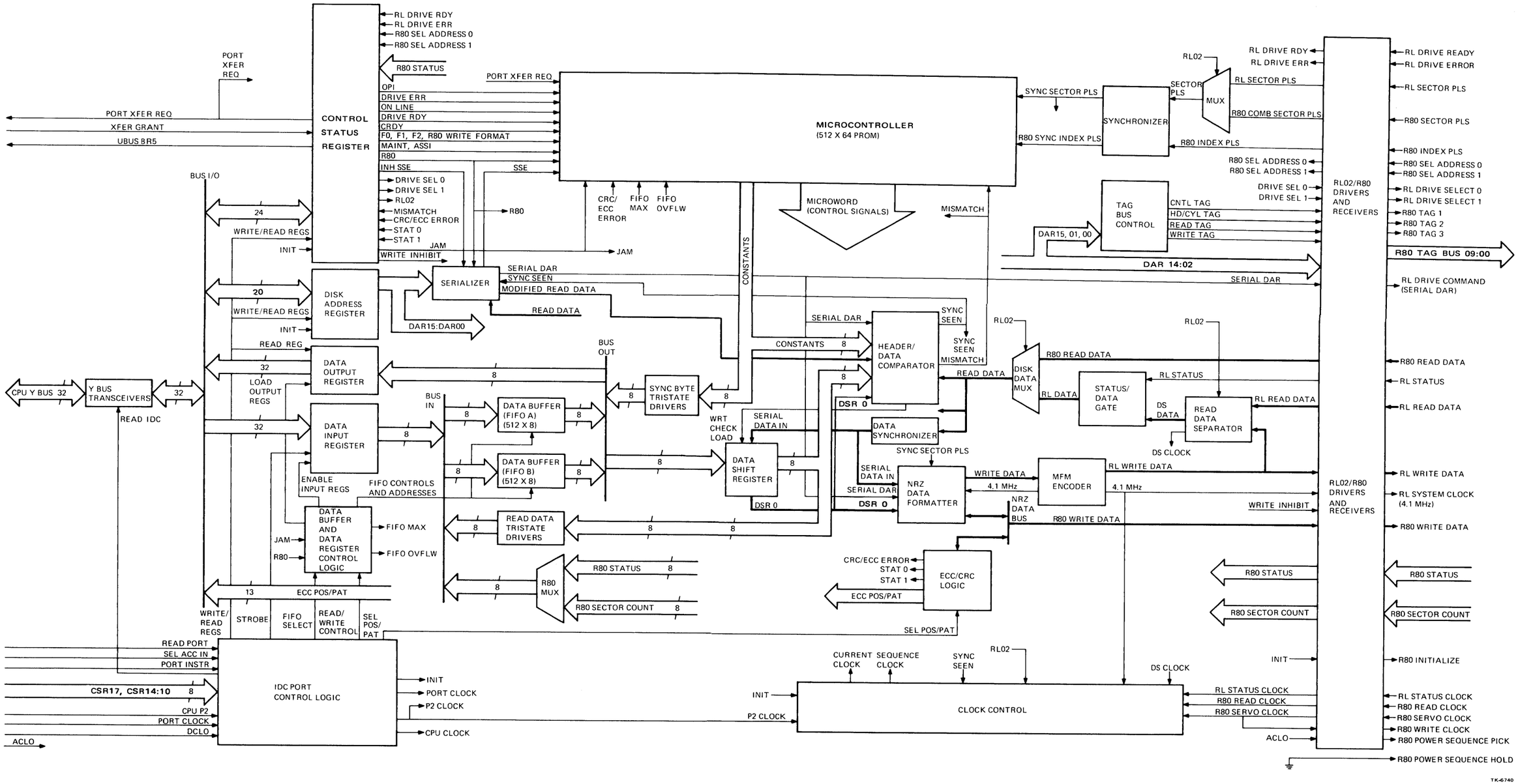


Figure 3-1 IDC Functional Block Diagram

3.3.7 Clock Control

The clock control synchronizes the operation of the IDC with either the selected disk drive or the CPU. Selection of the proper clock for use as the CURRENT and SEQUENCE CLOCK outputs is caused by inputs from the microcontroller.

3.3.8 TAG Bus Control

During an R80 seek function, the TAG bus control operates from microcontroller inputs to format and assert the disk drive control word from the disk address register to the R80 disk drive. During an R80 read or write function, the microcontroller inputs to the TAG bus control enable assertion of R80 read gate or write gate, as applicable.

3.3.9 Serializer

During an RL02 seek or get status function, the serializer operates from microcontroller inputs to format and assert the disk drive control word from the disk address register to the RL02 disk drives.

The serializer is also used during the read and write data functions to serialize the read/write data address contained in the disk address register so that the address can be compared with the address read from the disk drive. The address comparison is performed in the header/data comparator.

During the R80 read and write data functions, the serializer is also used to monitor the skip sector flag (bit 13) of the R80 header data and to assert a skip sector error (SSE) input to the microsequencer if a bad or displaced sector is encountered.

3.3.10 Header/Data Comparator

The header/data comparator operates from microcontroller inputs. During a read or write data function, the header/data comparator is used to locate the disk drive sector to or from which the data is to be written or read. During a write check function, the header/data comparator is used to perform a bit-by-bit comparison of data read from memory with the data contained in the data buffers.

3.3.11 Data Shift Register

The data shift register operates from microcontroller inputs during the read data, write data, and write check data functions. During the initial phases of the read data and write data functions, and during the write check data function, the data shift register is used with the header/data comparator to locate the header sync byte of the data read from the disk drive. Once the header sync byte has been located and a header match found, the operation of the data shift register depends on the function (write data, write check data, or read data).

During the write data function, the data shift register serializes the data bytes input from the data buffers and the sync byte input from the sync byte tristate drivers. The serialized output (DSR0) is asserted to the NRZ data formatter.

During the write check function, the data shift register serializes the data byte input from the data buffers and asserts the serialized data to the header/data comparator to allow a bit-by-bit comparison of the data contained in the data buffers with the data read from the disk drive.

During the read data function, the data shift register converts the serial read data input from the disk drive to a byte format for storage in the data buffers.

3.3.12 NRZ Data Formatter

The NRZ data formatter operates from microcontroller inputs. During the write and maintenance functions, the NRZ data formatter converts the DSR0 output of the data shift register into NRZ and WRITE DATA pulses. The NRZ data output is the write data input to the R80 disk drive and the data sample input to the ECC/CRC logic. The WRITE DATA pulses are input to the MFM encoder to produce the RL WRITE DATA pulses for the RL02 disk drives. When the CRC portion of the RL02 write data is to be written, the NRZ data formatter converts the NRZ output of the ECC/CRC logic to WRITE DATA pulse inputs to the MFM encoder. The CRC and ECC portions of the R80 WRITE DATA are written directly from the ECC/CRC logic by way of the NRZ data bus. During a read function, the NRZ data formatter converts the SERIAL DATA IN to an NRZ format for use in the ECC/CRC logic.

3.3.13 MFM Encoder

The MFM encoder converts the WRITE DATA inputs from the NRZ data formatter to MFM-encoded RL WRITE DATA. These data make up the write data inputs to the RL02 disk drives. During the IDC maintenance function (testing under microdiagnostic control), the RL WRITE DATA is applied to the read data separator to simulate the RL READ DATA output of the RL02 disk drives.

3.3.14 ECC/CRC Logic

The ECC/CRC logic operates from microcontroller inputs during the write and read functions. During both the write and read functions, the ECC/CRC logic generates both CRC and ECC data. During the write function, the ECC/CRC logic outputs this data, as applicable, to be written on the disk drive. During the read function, the ECC/CRC logic compares the CRC or ECC code generated with the ECC/CRC data received to validate the integrity of the data read from the disk drive. If an error is indicated by the comparison, an error signal is generated and the status of the error is indicated. The ECC/CRC logic may also be controlled by the CPU to cause the transfer of the ECC POS/PAT data from the IDC to the CPU.

3.3.15 Read Data Separator

The read data separator operates from microcontroller inputs to convert the MFM-encoded RL READ DATA to a format compatible with the IDC logic. The read data separator also generates the DS CLOCK, which is used to synchronize the IDC operation with the timing of the RL READ DATA input.

3.3.16 Status/Data Gate

The status/data gate operates from microcontroller inputs to enable either the RL STATUS input or DS DATA output of the read data separator to be applied as the RL DATA input to the disk data multiplexer.

3.3.17 Disk Data Multiplexer

The disk data multiplexer enables either the RL DATA or R80 READ DATA as the READ DATA input to the data synchronizer, serializer, and header/data comparator.

3.3.18 Data Synchronizer

The data synchronizer converts the READ DATA inputs to pulses having a pulse duration equal to the time interval between synchronizing clock pulses. [The applicable clock pulse used for synchronization (current clock) is selected by microcontroller inputs to the clock control.]

3.3.19 Sector and Index Pulse Multiplexer and Synchronizer

The SYNC SECTOR PLS and R80 SYNC INDEX PLS inputs to the microsequencer are controlled by a multiplexer and synchronizer. The multiplexer enables either the RL SECTOR PLS or the R80 COMB SECTOR PLS from the disk drives to be asserted as the SECTOR PLS input to the synchronizer.

The synchronizer conditions the SECTOR PLS and R80 INDEX PLS inputs such that the pulse duration of these inputs will be equal to the synchronizing clock pulses (current clock) asserted from the clock control.

3.4 IDC FUNCTIONAL THEORY OF OPERATION

Each of the IDC operations in causing a CPU-specified function is initiated by loading the IDC control status register (CSR) with an IDC control word with the CRDY bit reset. When the IDC control word is loaded into the CSR, the function bits (F0, F1, and F2) and the CRDY bit of the IDC control word input are asserted to the microcontroller (see Figure 3-1). The function bits specify to the microcontroller the function that is to be performed and are used to provide the branch condition inputs to preset the initial microword output of the microcontroller. The CRDY bit is the start command for the microcontroller. When the IDC is not busy performing a CPU-requested function, it operates in the idle mode of operation. While in the idle mode, the microcontroller sequentially enables and samples the operational status of each of the disk drives. After sampling the operational status of the disk drives, the microcontroller monitors the CRDY input from the CSR. If the CRDY input is reset, the microcontroller branches on the function bits (F0, F1, and F2) to preset the microword output of the microcontroller to initiate the CPU function specified.

The disk drive address bits of the IDC control word input are asserted on the DRIVE SEL 0 and DRIVE SEL 1 outputs of the CSR. The DRIVE SEL 0 and 1 outputs of the CSR are asserted on the RL DRIVE SEL 0 and RL DRIVE SEL 1 outputs of the IDC to enable, if applicable, one of the RL02 disk drives. Also, the disk drive address bits are used within the CSR. These bits are decoded to determine if the selected drive is an RL02 or the R80 and to enable the RL02 or R80 outputs of the CSR. Internal to the CSR, the RL02 and R80 signals couple the applicable RL DRIVE RDY, RL DRIVE ERROR, R80 DRIVE RDY, or R80 FAULT inputs to the CSR on the DRIVE RDY or DRIVE ERR inputs to the microcontroller. (Refer to Paragraph 3.5.1 for a more detailed description of disk drive select and drive status monitor.)

The R80 output of the CSR is asserted to the microcontroller, serializer, and data buffer and data register control logic. The R80 input to the microcontroller is used to control the sequence of microwords generated. The R80 input to the serializer determines the sequence in which the contents of the disk address register (DAR) is serialized. The R80 input to the data buffer and data register control logic is used to enable the FIFO MAX and FIFO OVFLW outputs after either 512 bytes of data (one full sector of R80 disk drive read data) or 256 bytes of data (one full sector of RL02 disk drive read data).

The RL02 output of the CSR is asserted to the clock control to enable the proper clock to be selected for synchronizing the IDC operation with the selected disk drive, and to the IDC multiplexers to enable either the RL02 or R80 data and sector pulse paths. The RL02 output conditions the read data separator.

The interrupt enable bit (IE) of the IDC control word input is used within the CSR to enable the UBUS BR5 interrupt signal on command by the microcontroller. The UBUS BR5 signal is asserted to the CPU to indicate that the function specified by the IDC control word input has been completed, or that the IDC operation has been halted due to a detected error.

The attention bits of the IDC control word input are used to reset the registered attention bits within the CSR.

The maintenance bit of the IDC control word input is asserted to the microcontroller to enable the IDC maintenance function.

The following paragraphs describe the operation of the IDC relative to each of the functions that can be specified by the IDC control word input (see Table 3-1). The functional operation of the IDC in the idle mode is also discussed. The following discussions are keyed to the functional block diagram in Figure 3-1. Where applicable, reference is made to more detailed discussions.

3.4.1 Seek Functions

Each of the four seek functions listed in Table 3-1 may be initiated by the CPU by loading the disk address register with the correct disk drive control word and loading the CSR with the IDC control word.

For the RL02 seek function, the disk address register is loaded with an RL02 seek command (Figure 2-4). For the R80 seek functions, the disk address register is loaded with one of the three R80 drive commands (seek command, Figure 2-5; head select command, Figure 2-6; or recalibrate command, Figure 2-7).

Since the sequence of IDC operations in initiating the seek functions and asserting the applicable drive commands depends on whether an RL02 or R80 disk drive is selected, the seek functions are discussed separately as follows.

3.4.1.1 RL02 Seek – When an RL02 seek function is specified by the IDC control word input, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output.

The microcontroller then checks the DRIVE RDY input to determine if the selected disk drive is ready (the selected disk drive is operational and not busy performing a seek). This check is performed because it is possible that a previous seek was issued to this drive and the seek has not yet been completed. If the DRIVE RDY input is present or when it is asserted, the microcontroller enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control, which synchronizes the IDC with the selected RL02 disk drive.

To avoid writing a command to the disk drive during the time that a sector pulse is present, the microcontroller loops until 50 microseconds after a sector pulse input (SYNC SECTOR PLS) from the disk drive has been asserted and has terminated. Then, the microcontroller enables the serializer to assert, serially, the contents of the DAR to the RL DRIVE COMMAND input of the applicable RL02 disk drive.

After the last bit of the DAR (DAR 15) has been asserted, the microcontroller selects the P2 CLOCK as the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control to allow the IDC to be synchronized with the CPU. Then, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word input was set, enables the UBUS BR5 signal output of the CSR. The UBUS BR5 signal signifies to the CPU that the seek command has been issued to the disk drive. The IDC then returns to the idle mode of operation.

3.4.1.2 R80 Seek – When the disk address register is loaded, the DAR 09:02 outputs are asserted on the R80 TAG BUS 09:02 signal lines. The DAR 15, 01, and 00 outputs are applied to the TAG bus control. DAR 14 and 13 are asserted as conditioning inputs to the R80 TAG 2 and R80 TAG 1 signal line drivers.

When an R80 seek is specified by the IDC control word input and CRDY is reset, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output.

The microcontroller then checks the DRIVE RDY input to determine if the R80 disk drive is ready (the R80 disk drive is operational and not busy performing a seek). If the DRIVE RDY input is present or when it is asserted, the microcontroller asserts a seek instruction input to the TAG bus control. The seek instruction enables the DAR 01 and 00 outputs of the disk address register to be asserted on the R80 TAG BUS 01 and 00 signal lines, respectively, via the READ TAG and WRITE TAG outputs of the TAG bus control. Next, the microcontroller asserts a strobe input to the TAG bus control. The strobe input enables the HD/CYL TAG output of the TAG bus control, and if DAR 15 is H (the seek instruction in the DAR is a recalibrate command), it enables the CNTL TAG output of the TAG bus control. The HD/CYL TAG output is used with the DAR 13 and 14 inputs to the R80 TAG 1 and R80 TAG 2 signal line drivers to enable the applicable R80 TAG input to the R80 disk drive (R80 TAG 1 if the disk address register was loaded with an R80 seek command; R80 TAG 2 if the disk address register was loaded with an R80 head select command). If the disk address register was loaded with an R80 recalibrate command, the CNTL TAG output of the TAG bus control would assert the R80 TAG 3 input to the R80 disk drive (see Table 2-2). (A detailed discussion of the TAG bus control logic is presented in Paragraph 3.5.2.)

Assertion of one of the R80 TAG inputs loads the R80 TAG BUS 9:0 signals into the R80 disk drive. After the specified seek instruction has been asserted, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word input was set, enables the UBUS BR5 signal output of the CSR. The IDC then returns to the idle mode of operation.

3.4.2 RL02 Get Status

The RL02 get status function is initiated by loading the disk address register with an RL02 get status command (Figure 2-3) and loading the CSR with the applicable IDC control word.

When an RL02 get status function is specified by the IDC control word, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output.

The microcontroller then selects FIFO A and clears the FIFO A address counter. Next the microcontroller enables the RL WRITE CLOCK (4.1 megahertz) to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control, which synchronizes the operation of the IDC with the selected disk drive. Then the microcontroller enables the serializer to assert, serially, the RL02 get status command from the disk address register to the RL DRIVE COMMAND input of the selected RL02 disk drive. After the RL02 get status command has been asserted, the microcontroller deselects the RL WRITE CLOCK and enables the RL STATUS CLOCK on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control.

The 16 bits of status information from the selected RL02 disk drive are asserted to the IDC in synchronization with the RL STATUS CLOCK. (The format and bit significance of the RL02 status information are shown in Figure 2-19.) The RL02 status information is applied to the IDC via the RL STATUS input to the RL02 receivers. Each bit of the RL02 status information is coupled through the status/data gate, disk data multiplexer, and data synchronizer, and is asserted to the data shift register.

After the first eight bits of status information have been shifted into the data shift register, the microcontroller enables the read data tristate drivers, which asserts the parallel output of the data shift register to the FIFOs. Then, the microcontroller writes the first eight bits as a single byte into FIFO A and increments the FIFO A address. After the second eight bits of status information have been shifted into the data shift register, the microcontroller again enables the read data tristate drivers and writes the second eight bits as a single byte into FIFO A. (A detailed discussion of how the microcontroller causes writing of data to the data buffers is provided in Paragraph 3.5.12.)

After the two bytes of status information have been loaded into FIFO A, the microcontroller deselects the RL STATUS CLOCK and enables the CPU CLOCK (P2 CLOCK) to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. Then, the microcontroller clears the FIFO address, sets the CRDY output of the CSR, and, if the IE bit of the previous IDC control word was set, asserts a UBUS BR5 interrupt to the CPU to signal that the requested function has been completed. The RL02 status information is now ready for transfer from the IDC to the CPU. The CPU transfers the RL02 status information from the IDC to the CPU as discussed in Paragraph 3.5.13.

3.4.3 R80 Get Status

The R80 get status function is initiated by loading the CSR with the applicable IDC control word. When the IDC control word is loaded, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output.

Then, the microcontroller selects FIFO A and clears the FIFO A address counter. The 16 bits of R80 status information from the R80 disk drive are asserted to the IDC in parallel format. The R80 status information and R80 sector count inputs are applied to the R80 multiplexer. After FIFO A has been selected and the FIFO address counter has been cleared, the microcontroller enables the R80 SECTOR COUNT through the R80 multiplexer, writes the sector count information into FIFO A, and increments the FIFO A address. Then, the microcontroller enables the R80 STATUS through the R80 multiplexer and writes the R80 status information into FIFO A.

After the R80 status information has been written into FIFO A, the microcontroller clears the FIFO A address counter, sets the CRDY output of the CSR, and, if the IE bit of the previous IDC control word was set, asserts a UBUS BR5 signal to the CPU. The R80 status information is now ready for transfer from the IDC to the CPU. The CPU transfers the R80 status information from the IDC to the CPU as discussed in Paragraph 3.5.13.

3.4.4 Read Header

The read header function is initiated by loading the CSR with the applicable IDC control word. The sequence of operations performed by the IDC in executing the read header function depends on whether the header data is to be retrieved from one of the RL02 disk drives or the R80 disk drive. These alternatives are discussed separately as follows.

3.4.4.1 RL02 Read Header – When the IDC control word is loaded, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output.

The microcontroller then selects FIFO A and resets the FIFO A address counter. Next the microcontroller checks the DRIVE RDY input to determine if the selected RL02 disk drive is ready (the selected disk drive is operational and not busy performing a seek). If the DRIVE RDY input is present or when it is asserted, the microcontroller enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. (A detailed discussion of the clock control is provided in Paragraph 3.5.3.) This synchronizes the operation of the IDC with the selected RL02 disk drive. Then, the microcontroller loops until the leading edge of a SYNC SECTOR PLS is detected. (This pulse is generated from the RL SECTOR PLS from the selected RL02 disk drive.) After the leading edge of the RL SECTOR PLS has been detected, the microcontroller loops until eight microseconds after the RL SECTOR PLS has terminated. After the loop, the microcontroller enables the read data separator, and then loops again until after 32 RL READ DATA pulses have been asserted. When enabled, the read data separator converts the MFM-encoded RL READ DATA to an NRZ format. The read data separator is enabled during the header preamble portion of the RL READ DATA from the RL02 disk drive (see Figure 2-20) and uses the first four

bytes of zeros to synchronize itself with the RL READ DATA input. In addition to converting the RL READ DATA to an NRZ format, the read data separator generates a clock (DS CLOCK) that is synchronized with the DS DATA output. After the synchronization loop, the microcontroller clears the data shift register, presets the CONSTANTS (which will allow the header/data comparator to determine when the sync byte of the RL READ DATA is present), and enables the DS CLOCK output of the read data separator to be asserted on the CURRENT CLOCK output of the clock control.

The SEQUENCE CLOCK output of the clock control is inhibited until the sync byte of the RL READ DATA has been asserted. This causes the microcontroller to stall until the sync byte is located. (A detailed discussion of how the sync byte is located is provided in Paragraph 3.5.4.)

When the sync byte has been located, the SYNC SEEN signal is asserted to the clock control, which enables the DS CLOCK to be asserted on the SEQUENCE CLOCK output of the clock control. Resumption of the SEQUENCE CLOCK restarts the microcontroller.

After the first eight bits following the sync byte have been shifted into the data shift register, the microcontroller enables the read data tristate drivers, loads the contents of the data shift register into FIFO A, and increments the FIFO A address counter. The following eight bits are also shifted into the data shift register, loaded into FIFO A, and the FIFO A address counter incremented. Now the two bytes of the RL sector address are contained in FIFO A.

As shown in Figure 2-20, the two bytes of RL READ DATA following the two address bytes are zeros. Thus, these data are not loaded into the FIFO. However, the two bytes of CRC data that follow are loaded into FIFO A. (A detailed discussion of how the microcontroller causes writing of data to the data buffers is provided in Paragraph 3.5.12.)

After the CRC data have been loaded into FIFO A, the microcontroller clears the FIFO A address counter and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. This allows the IDC to be synchronized with the CPU. The microcontroller also sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word input was set, asserts a UBUS BR5 signal to the CPU.

The RL02 header data are now ready for transfer from the IDC to the CPU. The CPU transfers the RL02 header data from the IDC to the CPU as discussed in Paragraph 3.5.13.

3.4.4.2 R80 Read Header – When the IDC control word is loaded, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output.

The microcontroller then selects FIFO A. Next the microcontroller checks the DRIVE RDY input to determine if the R80 disk drive is ready (the disk drive is operational and not busy performing a seek). If the DRIVE RDY input is present or when it is asserted, the microcontroller enables the R80 SERVO CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. (A detailed discussion of the clock control is provided in Paragraph 3.5.3.) This synchronizes the IDC operation with the operation of the R80 disk drive. Then the microcontroller loops until the leading edge of the SYNC SECTOR PLS is detected. (The SYNC SECTOR PLS is generated from the R80 SECTOR PLS or R80 INDEX PLS input from the R80 disk drive.) After the leading edge of the SYNC SECTOR PLS is detected (indicating the beginning portion of the sector), the microcontroller clears the FIFO A address counter and loops until 60 R80 SERVO CLOCK pulses have been asserted. This loop is initiated to inhibit enabling the R80 read circuitry until the read/write heads are positioned over the sector gap portion of the R80 header data. The microcontroller then enables the TAG bus control to assert the READ TAG and CNTL TAG signals to the R80 drivers. These signals enable the R80 drivers to assert the R80 TAG BUS 01 and R80 TAG 3 outputs of the IDC. These outputs enable the R80 read gate and allow the R80 READ DATA and R80 READ CLOCK to be asserted to the IDC.

After the R80 TAG BUS 01 and R80 TAG 3 signals have been asserted, the microcontroller loops until after 88 R80 servo clock pulses have been asserted to allow the R80 disk drive to achieve phase lock. Phase lock is achieved by reading a sequence of zeros in the sector gap of the R8O READ DATA. (See Figure 2-17 for the R80 READ DATA format.) After the phase lock loop, the microcontroller clears the data shift register and presets the CONSTANTS output of the microcontroller to the R80 SYNC BYTE pattern. Then the microcontroller enables the R80 READ CLOCK to be asserted on the CURRENT CLOCK output of the clock control. The R80 READ CLOCK is not asserted on the SEQUENCE CLOCK output until after the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the header sync byte of the R80 READ DATA is located. (A detailed discussion of how the sync byte is located is provided in Paragraph 3.5.4.)

When the header sync byte has been located, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the R80 READ CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller, which then enables the read data tristate drivers.

After the first eight bits of header information (first byte of cylinder address) have been shifted into the data shift register, the microcontroller loads the parallel output of the data shift register into FIFO A and then increments the FIFO A address counter. The remaining 40 bits of header information are converted into byte format and loaded into FIFO A in the same manner as the first eight bits. (A detailed discussion of how the microcontroller causes writing of data to the data buffers is provided in Paragraph 3.5.12.)

After all six bytes of R80 header data have been loaded into FIFO A, the microcontroller resets the FIFO A address counter and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. This synchronizes IDC operation with the CPU. The microcontroller also deasserts the read gate output of the tag bus control, sets the CRDY output of the CSR, and, if the IE bit of the previous IDC control word was set, asserts a UBUS BR5 signal to the CPU.

The R80 header data are now ready for transfer from the IDC to the CPU. The CPU transfers the R80 header data from the IDC to the CPU as discussed in Paragraph 3.5.13.

3.4.5 Write Data, Read Data, and Write Check Data

The write data, read data, and write check functions are initiated by loading the data to be written to the disk drive into the FIFO, loading the disk address register with the applicable read/write data address, and loading the CSR with the applicable IDC control word.

The first sector (512 bytes, R80; 256 bytes, RL02) of data to be written is loaded into FIFO A. If two sectors are to be written, the second sector is loaded into FIFO B. If several contiguous sectors of data are to be written, the CPU loads FIFO A with the first sector, and FIFO B with the second sector. After the IDC has transferred the data from FIFO A, and while it is transferring the second sector from FIFO B, the CPU loads the third sector of data into FIFO A. After the IDC has transferred the contents of FIFO B, and while it is transferring the third sector from FIFO A, the CPU loads FIFO B with the fourth sector of data. This process may be repeated until all sectors of the cylinder (31 sectors for the R80 and 40 sectors for the RL02) have been written.

After the data have been loaded into the FIFO(s) and the read/write data address has been loaded into the disk address register, the CPU initiates the write data function by loading the applicable IDC control word into the CSR.

The operational sequence executed by the IDC in performing the write data, read data, and write check data functions depends on whether an RL02 or the R80 disk drive is selected. Therefore, the operational sequences are discussed separately in the following paragraphs.

3.4.5.1 RL02 Write Data, Read Data, and Write Check – When an RL02 write data, read data, or write check function is specified by the IDC control word, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output. The microcontroller then selects FIFO A and resets the FIFO A address counter. Next, the microcontroller checks the DRIVE RDY input to determine if the selected disk drive is ready (the disk drive is operational and not busy performing a seek function). If the DRIVE RDY input is present or when it is asserted, the microcontroller enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. This synchronizes the IDC operation with the selected RL02 disk drive. (A detailed discussion of the clock control is provided in Paragraph 3.5.3.)

The microcontroller then loops until the leading edge of the SYNC SECTOR PLS is detected. This pulse is generated by the RL SECTOR PLS input from the RL02 disk drive. Presence of the SYNC SECTOR PLS indicates that the applicable read/write head of the RL02 disk drive is positioned at the beginning portion of a data sector. After the leading edge of the SYNC SECTOR PLS is detected, the microcontroller loops until the trailing edge of the SYNC SECTOR PLS is detected. Then the microcontroller again loops until 32 RL SYSTEM CLOCK (4.1 megahertz) pulses have been asserted to the microcontroller via the SEQUENCE CLOCK output of the clock control. This second microcontroller loop is initiated to prevent the read data separator from trying to achieve phase lock on data that may contain glitches. After the loop, the microcontroller enables the read data separator, clears the ECC/CRC logic, clears the MISMATCH output of the header/data comparator, and then loops until after 32 RL SYSTEM CLOCK (4.1 megahertz) pulses have been asserted to the microcontroller via the SEQUENCE CLOCK output of the clock control. This loop is initiated to allow time for the read data separator to achieve phase lock on the data being read from the disk (RL READ DATA input). Phase lock is achieved by reading a sequence of four bytes of zeros in the header preamble of the RL READ DATA. (See Figure 2-20 for the RL READ DATA format.)

After the loop for phase lock, the microcontroller presets the conditions for locating the header sync byte of the RL READ DATA. The microcontroller also conditions the serializer such that after the sync byte has been located, the address portion of the RL READ DATA input can be compared with the read/write data address contained in the disk address register.

To preset the conditions for locating the header sync byte, the microcontroller clears the data shift register and presets the CONSTANTS output of the microcontroller to the header sync byte pattern. Then the microcontroller selects the DS CLOCK for syncronization. The DS CLOCK is generated from the RL READ DATA input and thus synchronizes the IDC with the selected RL02 disk drive data rate. When the DS CLOCK from the read data separator is selected, the DS CLOCK is asserted on the CURRENT CLOCK output of the clock control. The DS CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until after the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the header sync byte has been found. (A detailed discussion of how the header sync byte is located is provided in Paragraph 3.5.4.)

When the RL READ DATA header sync byte is found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the DS CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller which then enables the ECC/CRC logic. The SYNC SEEN signal is asserted also to the serializer to enable the contents of the disk address register to be asserted serially to the header/data comparator where it is compared bit-by-bit with the address information of the RL READ DATA. (A detailed discussion of the RL02 header comparisons is provided in Paragraph 3.5.5.)

The address information of the RL READ DATA is also asserted via the data synchronizer on the SERIAL DATA IN input of the NRZ data formatter. The NRZ data formatter couples the SERIAL DATA IN to the ECC/CRC logic via the NRZ data bus. While the address information is being compared in the header/data comparator and while the results of the comparison are being tested, the ECC/CRC logic generates a CRC word based on the configuration of the two bytes of address information and the two bytes of zeros that follow the address information.

After the 16 bits of address information of the RL READ DATA have been compared with the read/write data address, the microcontroller turns off the serializer and monitors the MISMATCH output of the header/data comparator. If the MISMATCH output is low (the address information of the RL READ DATA did not match the read/write data address in the disk address register), the microcontroller enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. Then the microcontroller loops until the next sector is encountered (the next SYNC SECTOR PLS is asserted) before reinitiating the header/data comparison. This process is repeated until a match is found or until TIMEOUT occurs. (Refer to Paragraph 3.5.10 for a discussion of TIMEOUT.)

If the MISMATCH output is high (the address information of the RL READ DATA matched the read/write data address in the disk address register), the microcontroller loops until the two bytes of zeros following the address information of the RL READ DATA have been asserted to the ECC/CRC logic. Then the microcontroller enables the ECC/CRC logic to load the header CRC word of the RL READ DATA. After the header CRC word is loaded, the microcontroller enables the ECC/CRC logic to compare the CRC word generated by the ECC/CRC logic from the address information and two bytes of zeros of the address information with the header CRC word of the RL READ DATA.

If a CRC error is indicated by the ECC/CRC logic (CRC/ECC ERROR is asserted to microcontroller), the microcontroller deselects the DS CLOCK and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. This synchronizes the operation of the IDC with the CPU. Then the microcontroller sets the Operation Incomplete (OPI) and CRDY bits in the CSR. Next, if the IE bit of the previous IDC control word was set, the microcontroller generates and asserts a UBUS BR5 interrupt to the CPU.

If no CRC error is detected, the microcontroller clears the ECC/CRC logic and branches on the F1 and F2 bits of the IDC control word input to initiate the operations associated with the RL02 write data function, RL02 read data function, or RL02 write check function.

a. RL02 Write Data

After the proper sector has been located and the CRC pattern verified, the microcontroller checks to make certain that the data to be written to the disk were loaded into the FIFO (FIFO OVFLW is asserted to the microcontroller) and that the selected RL02 disk drive is operational (DRIVE RDY is asserted to the microcontroller).

If the FIFO was not filled by the CPU, the microcontroller enables the P2 CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control, sets the Data Late (DLT) error and CRDY bits in the CSR, clears the MISMATCH output of the header/data comparator, and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU.

If the selected RL02 disk drive is not operational (DRIVE RDY is not asserted), the microcontroller enables the P2 CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control, sets the Operation Incomplete (OPI) and CRDY bits in the CSR, and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU.

If the FIFO is full and the selected RL02 disk drive is operational, the IDC continues with the write data function. First the microcontroller deselects the DS CLOCK and enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. Then, after a loop, the microcontroller clears the FIFO address counter, clears the data shift register, and enables the NRZ data formatter and MFM encoder. Next the microcontroller again loops until 40 bits (zeros), part of the data preamble, have been written to the selected RL02 disk drive. (The zeros are written by holding the DSR0 input to the NRZ data formatter low.) When the last bit of the data preamble has been written, the microcontroller generates and enables the CONSTANTS from the microcontroller to be loaded into the data shift register. (The CONSTANTS specify the sync byte pattern (80_{16}) to be written as part of the RL write data preamble.) Then the microcontroller enables the CONSTANTS to be asserted serially via the DSR0 output of the data shift register to the NRZ data formatter. The NRZ data formatter samples the DSR0 output of the data shift register at a 4.1 megahertz rate and generates an NRZ formatted pulse train, which is asserted to the ECC/CRC logic via the NRZ data bus. The NRZ data formatter also generates the WRITE DATA inputs of the MFM encoder.

The MFM encoder translates the WRITE DATA inputs to an MFM format and asserts these data to the selected RL02 disk drive via the RL WRITE DATA signal line.

After the last bit of the sync byte has been asserted to the NRZ data formatter, the microcontroller enables the first byte of data from FIFO A to be loaded into the data shift register and increments the FIFO A address counter. At the same time, the microcontroller enables the ECC/CRC logic, which samples the bit configuration of the 256 bytes of data as it is being transferred to the disk drive and generates a 16-bit CRC word representative of the bit configuration.

After the first byte of data has been loaded, the data shift register serially asserts bits 0 through 7 of the first data byte to the NRZ data formatter. After bit 7 of the first data byte has been asserted to the NRZ data formatter, the second byte of data from FIFO A is loaded into the DSR and the FIFO A address counter is again incremented. After bit 7 of the second data byte has been asserted to the NRZ data formatter, the third data byte from FIFO A is loaded into the data shift register and the FIFO A address counter is incremented. This process is repeated until all 256 bytes of data from FIFO A have been loaded into the data shift register and asserted to the RL WRITE DATA input of the selected RL02 disk drive via the NRZ data formatter and MFM encoder. (A detailed discussion of how the microcontroller causes transfer of data from the data buffers to the data shift register and data shift register operation in serializing the data is provided in Paragraph 3.5.11.)

After the 256 bytes of data have been asserted on the RL WRITE DATA signal line (the FIFO A address counter has been incremented to its maximum count and FIFO MAX is asserted to the microcontroller), the microcontroller enables the ECC/CRC logic to assert serially the 16-bit CRC word derived from the bit configuration of the 256 bytes of data on the RL WRITE DATA signal line. The CRC word is asserted on the RL WRITE DATA signal line via the NRZ data bus, NRZ data formatter and MFM encoder. After the last bit of the CRC word is asserted, the microcontroller inhibits the ECC/CRC logic, and then holds the NRZ data formatter enabled until 16 zeros (data postamble) have been written. After the 16 zeros have been written, the microcontroller inhibits the NRZ data formatter and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control, which synchronizes IDC operation with the CPU. Then the microcontroller clears the FIFO A address counter and enables the PORT XFER REQ output of the CSR to be asserted to the CPU.

If more data are to be written, the CPU asserts XFER GRANT to the CSR. The XFER GRANT input resets the PORT XFER REQ output of the CSR, which causes the microcontroller to select FIFO B and then monitor the CRDY output of the CSR. If more data are to be transferred, the CRDY output of the CSR will have remained cleared and the microcontroller will then increment the read/write data address in the disk address register and reset the function timer. Then the microcontroller reinitiates the RL02 write data function to cause the transfer of the data contained in FIFO B to the next sector of the RL02 disk drive.

If no more data is to be written, the CPU responds to the PORT XFER REQ input by loading an IDC control word with CRDY set and then asserting XFER GRANT. The XFER GRANT input to the IDC resets the PORT XFER REQ signal. When the PORT XFER REQ signal is reset, the microcontroller monitors the CRDY output of the CSR. If the CRDY output is set, indicating that no more data are to be transferred, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU.

b. RL02 Read Data

After the proper sector has been located and the CRC pattern verified, the microcontroller checks to make certain that the selected FIFO is empty. If the FIFO is full (FIFO OVFLW is asserted to the microcontroller), the microcontroller clears the MISMATCH output of the header/data comparator and sets the CRDY and Data Late (DLT) error bits in the CSR. If the IE bit of the previous IDC control word was set, the microcontroller also generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

If the FIFO is empty, the microcontroller deselects the DS CLOCK and enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. The microcontroller then loops until the write splice area within the header gap has passed the read/write heads of the RL02 disk drive. Then the microcontroller clears the ECC/CRC logic and enables the read data separator.

Next, the microcontroller loops until 32 RL READ DATA pulses have been asserted to the IDC. This loop is initiated to allow the read data separator to achieve phase lock on the data being read from the disk. After the phase lock loop, the microcontroller clears the selected FIFO address counter and presets and asserts the CONSTANTS output of the microcontroller to the header/data comparator. (The CONSTANTS output is preset to the bit configuration of the RL READ DATA data preamble sync byte.) The microcontroller then enables the DS CLOCK to be asserted on the CURRENT CLOCK output of the clock control. The DS CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the RL READ DATA data preamble sync byte has been found. (A detailed discussion of how the sync byte is located is provided in Paragraph 3.5.4.)

Detection of the sync byte of the data preamble signals the start of the data segment of the sector to be read. When the header preamble sync byte has been found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the DS CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller, which then enables the ECC/CRC logic, and begins converting the RL READ DATA into byte format and storing the 256 bytes of RL READ DATA in the selected FIFO. (A detailed discussion of how the READ DATA are converted to byte format and stored in the data buffers is provided in Paragraph 3.5.12.)

Each bit of the 256 bytes of RL READ DATA is used in the ECC/CRC logic to generate a 16-bit CRC word representative of the bit configuration of the RL READ DATA. After all 256 bytes of RL READ DATA have been loaded into the selected FIFO (FIFO MAX is asserted to the microcontroller), the microcontroller enables the 16-bit CRC word from the RL02 disk drive to be loaded into the ECC/CRC logic. After the CRC word has been loaded, the microcontroller enables the ECC/CRC logic to compare the CRC word generated from the 256 bytes of RL READ DATA with the CRC word read from the disk. Next the microcontroller deselects the DS CLOCK and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. Then the microcontroller monitors the CRC/ECC ERROR signal output of the ECC/CRC logic.

If a CRC/ECC error is indicated, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. Then the IDC returns to the idle mode of operation.

If no CRC/ECC ERROR is indicated, the microcontroller clears the selected FIFO address counter and generates and asserts the PORT XFER REQ output of the CSR to the CPU. This signal signifies that the IDC has completed reading a sector of data and that the data are ready for transfer to the CPU.

If more data are to be read, the CPU asserts a XFER GRANT signal to the CSR. When the XFER GRANT signal is asserted, the PORT XFER REQ output is reset. When the PORT XFER REQ is reset, the microcontroller changes the FIFO selected and monitors the CRDY output of the CSR. If the CRDY output of the CSR has remained cleared, the microcontroller increments the read/write data address in the disk address register, resets the timer, and reinitiates the RL02 read data function to read the next sector of RL READ DATA and store the data in the selected FIFO.

If no further data are to be read, the CPU responds to the PORT XFER REQ input by loading an IDC control word with CRDY set and then asserting XFER GRANT. The XFER GRANT input to the IDC resets the PORT XFER REQ signal. When the PORT XFER REQ signal is reset, the microcontroller monitors the CRDY output of the CSR. If the CRDY output is set, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

c. RL02 Write Check

After the proper sector has been located and the CRC pattern verified, the microcontroller checks to make certain that the data to be compared with the data from the disk drive were loaded into the FIFO (FIFO MAX is asserted to the microcontroller).

If the FIFO was not filled by the CPU, the microcontroller enables the P2 CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control, sets the Data Late (DTL) error and CRDY bits in the CSR, clears the MISMATCH output of the header/data comparator, and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU.

If the FIFO is full, the IDC continues with the write check function. First the microcontroller enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. Next, the microcontroller disables the read data separator and then loops until the write splice area within the header gap has passed the read/write heads of the RL02 disk drive. The read data separator is disabled to prevent the read data separator circuitry from being triggered by data glitches at the beginning of the header gap. (The data glitches were produced when the write heads were first turned on when the header gap was written.)

After the loop to allow the read/write heads of the RL02 disk drive to be positioned over the valid data in the header gap, the microcontroller again enables the read data separator. After the read data separator is enabled, the microcontroller loops until 32 RL SYSTEM CLOCK pulses have been asserted to the microcontroller via the SEQUENCE CLOCK output of the clock control. This loop is initiated to allow the read data separator to again achieve phase lock on the data being read from the disk.

After the phase lock loop, the microcontroller clears the FIFO address counter and enables the first byte of data from the selected data buffer to be asserted to the data shift register. The microcontroller also clears the MISMATCH output of the header/data comparator, and presets and enables the CONSTANTS from the microcontroller to be asserted to the header/data comparator. (The CONSTANTS output is preset to the bit configuration of the RL02 READ DATA data preamble sync byte.) The microcontroller then enables the DS CLOCK to be asserted on the CURRENT CLOCK output of the clock control. The DS CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the RL02 READ DATA data preamble sync byte has been found. (A detailed discussion of how the sync byte is found is provided in Paragraph 3.5.4.)

Detection of the RL02 READ DATA data preamble sync byte signals the start of the data segment of the sector on which the write check is to be performed. When the sync byte has been found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the DS CLOCK to be asserted onto the SEQUENCE CLOCK output. This restarts the microcontroller, which then enables the ECC/CRC logic.

In the write check mode, the WRT CHK LOAD output of the header/data comparator is enabled also when the RL02 READ DATA data preamble sync byte is found. The WRT CHK LOAD signal is asserted to the data shift register where it enables the first data byte from the selected FIFO to be loaded into the data shift register. The microcontroller then increments the selected FIFO address counter.

When the first data byte is loaded into the data shift register, bit 0 of the first data byte is asserted to the header/data comparator via the DSR0 output of the data shift register. The first bit of the data portion of the first data byte is asserted to the header/data comparator coincident with the first bit of the data portion of the RL READ DATA asserted from the disk drive. (Because the CURRENT CLOCK used by the data shift register is derived from the RL READ DATA input, the data loaded into the data shift register are serialized and asserted to the header/data comparator in sync with each bit of the RL READ DATA input.) The data shift register serializes and asserts bits 0 through 7 of the first data byte to the header/data comparator.

After bit 7 of the first data byte has been asserted to the header/data comparator, the microcontroller loads the second byte of data from the selected FIFO into the data shift register and increments the selected FIFO address counter.

After bit 7 of the second data byte has been serialized and asserted to the header/data comparator, the microcontroller loads the third data byte from the selected FIFO into the data shift register and increments the FIFO A address counter. This process is repeated until all 256 bytes of data from the selected FIFO have been serialized and asserted to the header/data comparator for comparison with the RL READ DATA input. (A detailed discussion of how the microcontroller and data shift register cause serialization of data from the data buffers is provided in Paragraph 3.5.11.)

The header/data comparator performs a bit-by-bit comparison of the RL READ DATA input with the DSR0 input to determine if the RL READ DATA matches the serialized data from the data shift register.

Each bit of the data asserted to the header/data comparator for comparison with the RL READ DATA is asserted also to the ECC/CRC logic via the NRZ data formatter and NRZ data bus. The ECC/CRC logic generates a 16-bit CRC word based on the configuration of the 2048 data bits asserted to the ECC/CRC logic via the DSR0 input to the NRZ data formatter.

After all 256 bytes of data from the FIFO have been serialized and asserted to the header/data comparator for comparison with the RL READ DATA input (the FIFO A address counter has been incremented to its maximum count and FIFO MAX is asserted to the microcontroller), the microcontroller strobes the header/data comparator to sample the results of the data comparison. If the data did not compare, the MISMATCH output of the header/data comparator will remain low. If the data matched the RL READ DATA, the MISMATCH output will be set high. The MISMATCH output is asserted to the status logic in the CSR and to the microcontroller.

The microcontroller also enables the ECC/CRC logic to load the 16-bit CRC word being read from the RL02 disk drive. After the CRC word has been loaded, the microcontroller enables the ECC/CRC logic to compare the two CRC words (the CRC word generated from the 2048 bits of data used for comparison with the 2048 bits of RL READ DATA with the CRC word read from the disk drive). Then the microcontroller enables the P2 CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control to synchronize IDC operation with the CPU.

If a CRC comparison error is indicated (the ECC/CRC ERROR signal is asserted to the microcontroller and CSR), the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal the CPU. Then the IDC returns to the idle mode of operation. [Note that if the results of the data comparison (the 2048 bits of RL READ DATA with the 2048 bits of data from the FIFO) did not match, then a CRC error will also occur. It is not, therefore, necessary to terminate the write check function when a data comparison error is detected. However, if the write check function is terminated by a CRC error, the CPU can determine if the error was CRC related or a data comparison error by reading the IDC status word. The results of the data comparison (MISMATCH) and CRC comparison (CRC/ECC ERROR) are made available to the CPU via the IDC status word.]

If no ECC comparison error is indicated, the microcontroller clears the selected FIFO address counter and enables the PORT XFER REQ output of the CSR to be asserted to the CPU. The PORT XFER REQ signal signifies to the CPU that the write check function has been performed on the requested sector of data and that the data comparison was valid.

If the write check function is to be performed on the next sector of data, the CPU asserts an XFER GRANT signal to the CSR. The XFER GRANT input resets the PORT XFER REQ output. When the PORT XFER REQ output is reset, the microcontroller changes the FIFO selected and monitors the CRDY output of the CSR. If the write check function is to be continued, the CRDY output of the CSR will have remained cleared, and the microcontroller will increment the read/write data address contained in the disk address register and reset the function timer. Then the microcontroller checks the DRIVE RDY input. If the selected RL02 disk drive is operational, the microcontroller reinitiates the RL02 write check function to compare the next sector of data from the selected RL02 disk drive with the data contained in the selected FIFO.

If the write check function is not to be continued, the CPU responds to the PORT XFER REQ input by loading an IDC control word with CRDY set and then asserting XFER GRANT. The XFER GRANT input to the IDC resets the PORT XFER REQ signal. When the PORT XFER REQ signal is reset, the microcontroller monitors the CRDY output of the CSR. If the CRDY output is set, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of the operation.

3.4.5.2 R80 Write Data, Read Data, and Write Check – When a R80 write data, read data, or write check function is specified by the IDC control word, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output. The microcontroller then selects FIFO A and resets the FIFO A address counter. Next the microcontroller checks the DRIVE RDY input to determine if the disk drive is ready (the R80 disk drive is operational and not busy performing a seek). If the DRIVE RDY input is present or when it is asserted, the microcontroller enables the R80 SERVO CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. This synchronizes the operation of the IDC with the R80 disk drive. (A detailed discussion of the clock control is provided in Paragraph 3.5.3.)

The microcontroller then loops until the leading edge of the SYNC SECTOR PLS is detected. This pulse is generated by the R80 SECTOR PLS or R80 INDEX PLS inputs from the R80 disk drive. Presence of the SYNC SECTOR PLS indicates that the applicable read/write head of the R80 disk drive is positioned at the beginning portion of the header data. After the leading edge of the SYNC SECTOR PLS is detected, the microcontroller loops until 60 R80 SERVO CLOCK pulses have been asserted. This microcontroller loop is initiated to prevent the R80 disk drive from trying to achieve phase lock on data that may contain glitches. After the loop, the microcontroller enables the TAG bus control to assert the READ TAG and CNTL TAG signals to the R80 drivers. These signals enable the R80 drivers to assert the R80 TAG BUS 01 and R80 TAG 3 outputs of the IDC (assert a read gate command to the R80 disk drive). The read gate command enables the R80 disk drive to read the data from the disk and assert the data to the IDC via the R80 READ DATA signal line. The R80 disk drive also generates and asserts the R80 READ CLOCK, which is synchronized with the READ DATA input to the IDC.

After the read gate command has been asserted, the microcontroller clears the ECC/CRC logic, clears the MISMATCH output of the header/data comparator, and then loops until after 88 R80 SERVO CLOCK pulses have been asserted to the IDC. This loop is initiated to allow time for the R80 disk drive to achieve phase lock on the data being read from the disk. Phase lock is achieved by reading a sequence of zeros in the sector gap of the R80 READ DATA. (See Figure 2-17 for the R80 READ DATA format.)

After the loop for phase lock, the microcontroller presets the conditions for locating the header sync byte of the R80 READ DATA. The microcontroller also conditions the serializer such that after the sync byte has been located, the address portion of the R80 READ DATA input can be compared with the read/write data address contained in the disk address register.

To preset the conditions for locating the header sync byte, the microcontroller clears the data shift register and presets the CONSTANTS output of the microcontroller to the header sync byte pattern. Then the microcontroller selects the R80 READ CLOCK for synchronization. The R80 READ CLOCK is generated within the R80 disk drive from the R80 READ DATA input and thus synchronizes the IDC with the selected R80 disk drive data rate.

When the microcontroller selects the R80 READ CLOCK, the R80 READ CLOCK is asserted on the CURRENT CLOCK output of the clock control. The R80 READ CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the R80 header sync byte has been found. (A detailed discussion of how the sync byte is located is provided in Paragraph 3.5.4.)

When the R80 READ DATA header sync byte is found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the R80 READ CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller, which enables the ECC/CRC logic. The SYNC SEEN signal is also asserted to the serializer to enable the contents of the disk address register to be asserted to the header/data comparator, where it is compared bit-by-bit with the address information of the R80 READ DATA.

The format of the R80 read/write data address contained in the disk address register is not the same as the format of the address information contained in the header of the R80 READ DATA input. In addition to address information, the header of the R80 READ DATA input contains unused bits, various flags, and skip sector information. During the R80 read/write address comparison, the serializer performs three functions:

1. Modifying the READ DATA input to mask the unused and various flag bits contained in the header of the READ DATA input order of each of the bits

2. Controlling assertion order of each of the bits of the read/write data address contained in the disk address register to enable these bits to be compared with the corresponding bits of the READ DATA input

3. Recording, if enabled, the status of the skip sector flag in the header of the READ DATA input

The header/data comparator performs a bit-by-bit comparison of the MODIFIED READ DATA input with the SERIAL DAR output of the serializer to determine if the MODIFIED READ DATA matches the read/write data address contained in the disk address register. (A detailed discussion of the R80 header data comparison including monitoring for the skip sector flag is provided in Paragraph 3.5.6.)

Each bit of the R80 header data asserted to the data synchronizer is asserted also to the SERIAL DATA IN input of the NRZ data formatter. The NRZ data formatter couples the SERIAL DATA IN to the ECC/CRC logic via the NRZ data bus. The ECC/CRC logic generates a CRC word based on the configuration of the 32 bits of the R80 header data.

After all 32 bits of the R80 header data have been compared, the microcontroller turns off the serializer and monitors the MISMATCH output of the header/data comparator. If the MISMATCH output is low (the address information of the R80 header data did not match the read/write data address in the disk address register), the microcontroller enables the R80 SERVO CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE clock outputs of the clock control. Then the microcontroller loops until the next sector is encountered (the next SYNC SECTOR PLS is asserted) before reinitiating the header data comparison. This process is repeated until a match is found or until TIMEOUT occurs. (A detailed discussion of the timeout logic is provided in Paragraph 3.5.10.)

If the MISMATCH output is high (the address information of the R80 header data matched the read/write data address in the disk address register), the microcontroller then monitors the SSE output of the serializer.

If the skip sector flag (SSE) of the R80 header data was set and the INH SSE bit of the IDC control word is not asserted, indicating that the sector being read is a bad or displaced sector, the serializer asserts a skip sector error (SSE) signal to the microcontroller.

If a skip sector error is indicated, the microcontroller sets the SSE output of the CSR and increments the sector address contained in the disk address register. (Only one bad sector may be encountered per cylinder; however, each sector following the bad sector will also be flagged because it will have been displaced. Provision for an additional sector (Sector 31) is provided on each cylinder; therefore, if a bad or displaced sector is encountered on the current cylinder, the microcontroller can inhibit monitoring for further skip sector flags during the remainder of the current function without degrading system performance.) Then the microcontroller loops until the next SYNC SECTOR PLS is asserted before reinitiating the header data comparison.

If the sector being read is not a bad or displaced sector, the microcontroller then enables the ECC/CRC logic to load the CRC word of the R80 header data. After the R80 header CRC is loaded, the microcontroller enables the ECC/CRC logic to compare the CRC word generated by the ECC/CRC logic from the R80 header data with the R80 header CRC word.

If a CRC error is indicated by the ECC/CRC logic, the microcontroller deselects the R80 READ CLOCK and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control to synchronize the operation of the IDC with the CPU. Then the microcontroller sets the Operation Incomplete (OPI) bit and sets the CRDY bit in the CSR. Next, if the IE bit of the previous IDC control word was set, the microcontroller generates and asserts a UBUS BR5 interrupt to the CPU.

If no CRC error is detected, the microcontroller clears the ECC/CRC logic and branches on the F0, F1, and F2 bits of the IDC control word input to initiate the operations associated with the R80 write data function, the R80 READ DATA function, or the R80 write check function.

a. R80 Write Data

After the proper sector has been located and the CRC pattern verified, the microcontroller selects the R80 SERVO CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. Next the microcontroller checks to make certain that the data to be written to the disk were loaded into the FIFO (FIFO MAX is asserted to the microcontroller). If the FIFO was not filled by the CPU, the microcontroller enables the P2 CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control, sets the Data Late (DLT) error and CRDY bits in the CSR, clears the MISMATCH output of the header/data comparator, and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU.

If the FIFO is full, the IDC continues with the write data function. First, the microcontroller disables the read gate signal output of the TAG bus control, which deasserts the READ GATE signal from the R80 disk drive. Then, after a loop, the microcontroller enables the TAG bus control to assert a write gate command to the R80 disk drive. The microcontroller also clears the FIFO address counter and clears the data shift register. Then the microcontroller again loops until 120 bits of zeros (header gap) have been written to the R80 disk drive. (The zeros are written by holding the R80 WRITE DATA output of the NRZ data formatter low.) When the last bit of the header gap is written, the microcontroller generates and enables the CONSTANTS from the microcontroller to be loaded into the data shift register. (The CONSTANTS specify the sync byte pattern to be written in the R80 header gap, that is, 19_{16}.) Then, the microcontroller enables the NRZ data formatter and asserts serially the sync byte data from the data shift register to the NRZ data formatter. The NRZ data formatter samples the DSR0 output of the data shift register and generates an NRZ formatted pulse train that is asserted on the R80 WRITE DATA signal line and to the ECC/CRC logic via the NRZ data bus.

After the last bit of the sync byte has been asserted to the NRZ data formatter, the microcontroller enables the first byte of data from FIFO A to be loaded into the data shift register and increments the FIFO A address counter. At the same time, the microcontroller enables the ECC/CRC logic, which samples the bit configuration of the 512 bytes of data as they are being transferred to the disk drive and generates a 32-bit ECC word representative of the bit configuration.

After the first byte of data has been loaded, the data shift register serializes and asserts bits 0 through 7 of the first data byte to the NRZ data formatter. After bit 7 of the first data byte has been asserted to the NRZ data formatter, the second byte of data from FIFO A is loaded into the data shift register and the FIFO A address counter is incremented. After bit 7 of the second data byte has been asserted to the NRZ data formatter, the third data byte from FIFO A is loaded into the data shift register and the FIFO A address counter is incremented. This process is repeated until all 512 bytes of data from FIFO A have been loaded into the data shift register and asserted to the R80 WRITE DATA input of the R80 disk drive via the NRZ data formatter. (A detailed discussion of how the microcontroller causes the transfer of data from the data buffers to the data shift register and data shift register operation in serializing the data is provided in Paragraph 3.5.11.)

After the 512 bytes of data from the FIFO have been asserted on the R80 WRITE DATA signal line (the FIFO A address counter has been incremented to its maximum count), FIFO MAX is asserted to the microcontroller. When FIFO MAX is asserted, the microcontroller enables the ECC/CRC logic to assert serially the 32-bit ECC word derived from the bit configuration of the 512 bytes of data asserted on the R80 WRITE DATA signal line via the NRZ data bus.

After the last bit of the ECC word is asserted, the microcontroller inhibits the ECC/CRC, holds the WRITE GATE output of the TAG bus control logic asserted to the R80 disk drive and the NRZ data formatter enabled until 16 zeros have been written to the data gap. After the 16 zeros have been written, the microcontroller inhibits the TAG bus control, which deasserts the WRITE GATE signal from the R80 disk drive. The microcontroller then enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control, which synchronizes IDC operation with the CPU. Next the microcontroller clears the FIFO A address counter and enables the PORT XFER REQ output of the CSR to be asserted to the CPU.

If more data are to be written, the CPU asserts XFER GRANT to the CSR. The XFER GRANT input resets the PORT XFER REQ output of the CSR, which causes the microcontroller to select FIFO B and then monitor the CRDY output of the CSR. If more data are to be transferred, the CRDY output of the CSR will have remained cleared and the microcontroller will then increment the read/write data address in the disk address register and reset the function timer. Then the microcontroller checks the DRIVE RDY input. If the R80 disk drive is operational, the microcontroller reinitiates the R80 write data function to cause the transfer of the data contained in FIFO B to the next sector of the R80 disk drive.

If no more data are to be written, the CPU responds to the PORT XFER REQ input by loading an IDC control word with CRDY set and then asserting XFER GRANT. The XFER GRANT input to the IDC resets the PORT XFER REQ signal. When the PORT XFER REQ signal is reset, the microcontroller monitors the CRDY output of the CSR. If the CRDY output is set, indicating that no more data are to be transferred, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

b. R80 Read Data

After the proper sector has been located and the CRC pattern verified, the microcontroller selects the R80 SERVO CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. Then the microcontroller checks to make certain that the FIFO is empty. If the FIFO is full (FIFO MAX is asserted to the microcontroller), the microcontroller clears the MISMATCH output of the header/data comparator and sets the CRDY and Data Late (DLT) error bits in the CSR. If the IE bit of the previous IDC control word was set, the microcontroller also generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

If the FIFO is empty, the microcontroller causes the TAG bus control to deassert the read gate command from the R80 disk drive. The microcontroller then loops until the write splice area within the header gap has passed the read/write heads of the R80 disk drive. Then the microcontroller enables the TAG bus control to reassert the read gate command to the R80 disk drive. Next the microcontroller clears the ECC/CRC logic.

After the read gate command is asserted to the R80 disk drive, the microcontroller loops until 88 R80 SERVO CLOCK pulses have been asserted to the IDC. This loop is initiated to allow the R80 disk drive to achieve phase lock on the data being read from the disk. After the phase lock loop, the microcontroller clears the selected FIFO address counter, and presets and asserts the CONSTANTS output of the microcontroller to the header/data comparator. (The CONSTANTS output is preset to the bit configuration of the R80 READ DATA header gap sync byte, that is, 19_{16}.) The microcontroller then enables the R80 READ CLOCK to be asserted on the CURRENT CLOCK output of the clock control. The R80 READ CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the R80 header gap sync byte has been found. (A detailed discussion of how the sync byte is found is provided in Paragraph 3.5.4.)

Detection of the sync byte of the R80 header gap signals the start of the data segment of the sector to be read. When the R80 header gap sync byte has been found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the R80 READ CLOCK to be asserted onto the SEQUENCE CLOCK output. This restarts the microcontroller, which then enables the ECC/CRC logic, and begins converting the R80 READ DATA into byte format and storing the 512 bytes of R80 READ DATA into the selected FIFO.

After the data shift register has been loaded with the first eight bits of R80 READ DATA, the microcontroller enables the parallel output of the data shift register to be asserted to the input of the FIFO(s) via the READ DATA tristate drivers, loads the data byte into the selected FIFO, and increments the selected FIFO address counter. This process (converting the R80 READ DATA to byte format and storing each byte) is repeated until all 512 bytes of R80 READ DATA have been written into the selected FIFO. (A detailed discussion of how the READ DATA are converted to byte format and stored in the data buffer is provided in Paragraph 3.5.12.)

Each bit of the 512 bytes of R80 READ DATA is used in the ECC/CRC logic to generate a 32-bit ECC word representative of the bit configuration of the R80 READ DATA. After all 512 bytes of R80 READ DATA have been loaded into the selected FIFO (FIFO MAX is asserted to the microcontroller), the microcontroller enables the 32-bit ECC word from the R80 disk drive to be loaded into the ECC/CRC logic. After the ECC word has been loaded, the microcontroller enables the ECC/CRC logic to compare the ECC word generated from the 512 bytes of R80 READ DATA with the ECC word read from the disk. Then the microcontroller clears the FIFO address counter and monitors the CRC/ECC ERROR signal output of the ECC/CRC logic.

If a CRC/ECC error is indicated, the microcontroller initiates an ECC correction routine. At the completion of the correction routine, the results of the correction computation are indicated in the STAT 0 and STAT 1 signals that are asserted to the status logic of the CSR. On completion of the correction computation, the microcontroller deselects the R80 READ CLOCK and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. Then the microcontroller clears the selected FIFO address counter, sets the CRDY output of the CSR, and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

If no CRC/ECC ERROR is indicated, the microcontroller deselects the R80 READ CLOCK and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. This synchronizes the IDC with the CPU. Then the microcontroller generates and asserts the PORT XFER REQ output of the CSR to the CPU. This signal signifies that the IDC has completed reading one sector of data and that the data are ready for transfer to the CPU.

If more data are to be read, the CPU asserts an XFER GRANT signal to the CSR. When the XFER GRANT signal is asserted, it resets the PORT XFER REQ output. When the PORT XFER REQ is reset, the microcontroller changes the FIFO selected and monitors the CRDY output of the CSR. If more data are to be read, the CRDY output of the CSR will have remained cleared, and the microcontroller will increment the read/write data address in the disk address register and reset the function timer. Then the microcontroller checks the DRIVE RDY input. If the R80 disk drive is operational, the microcontroller reinitiates the R80 READ DATA function to read the next sector of data from the R80 disk drive and store the data in the selected FIFO.

If no further data are to be read, the CPU responds to the PORT XFER REQ input by loading an IDC control word with CRDY set and then asserting XFER GRANT. The XFER GRANT input to the IDC resets the PORT XFER REQ signal. When the PORT XFER REQ signal is reset, the microcontroller monitors the CRDY output of the CSR. If the CRDY output is set, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

c. R80 Write Check

After the proper sector has been located and the CRC pattern verified, the microcontroller checks to make certain that the data to be compared with the data from the disk drive were loaded into the FIFO (FIFO OVFLW is asserted to the microcontroller).

If the FIFO was not filled by the CPU, the microcontroller enables the P2 CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. Then the microcontroller sets the Data Late (DLT) error and CRDY bits in the CSR, clears the MISMATCH output of the header/data comparator, and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU.

If the FIFO is full, the IDC continues with the write check function. First, the microcontroller selects the R80 SERVO CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. Next, the microcontroller disables the READ GATE signal output of the TAG bus control, which deasserts the READ GATE signal from the R80 disk drive. The READ GATE signal is deasserted to disable the R80 disk drive read circuitry from being triggered by data glitches at the beginning of the header gap. (The data glitches were produced when the write heads were first turned on when the header gap was written.) Then, after a loop to allow the read/write heads of the R80 disk drive to be positioned over the valid data in the header gap, the microcontroller enables the TAG bus control to reassert the read gate to the R80 disk drive. After the read gate command is asserted, the microcontroller loops until 88 R80 SERVO CLOCK pulses have been asserted to the IDC. This loop is initiated to allow the R80 disk drive to again achieve phase lock on the data being read from the disk. After the phase lock loop, the microcontroller clears the FIFO address counter and enables the first byte of data from the selected data buffer to be asserted to the data shift register. The microcontroller also clears the MISMATCH output of the header/data comparator, and presets and enables the CONSTANTS from the microcontroller to be asserted to the header/data comparator. (The CONSTANTS output is preset to the bit configuration of the R80 READ DATA header gap sync byte, that is, 19_{16}.) The microcontroller then enables the R80 READ CLOCK to be asserted on the CURRENT CLOCK output of the clock control. The R80 READ CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the R80 READ DATA header gap sync byte has been found. (A detailed discussion of how the sync byte is found is provided in Paragraph 3.5.4.)

Detection of the R80 READ DATA header gap sync byte signals the start of the data segment of the sector on which the write check is to be performed. When the header gap sync byte has been found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the R80 READ CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller, which then enables the ECC/CRC logic.

In the write check mode, the WRT CHK LOAD output of the header/data comparator is also enabled when the R80 READ DATA header gap sync byte is found. The WRT CHK LOAD signal is asserted to the data shift register where it enables the first data byte from the selected FIFO to be loaded into the data shift register. The microcontroller then increments the selected FIFO address counter.

When the first data byte is loaded into the data shift register, bit 0 of the first data byte is asserted to the header/data comparator via the DSR0 output of the data shift register. The first bit of the first data byte is asserted to the header/data comparator coincident with the first bit of the data portion of the R80 READ DATA asserted from the disk drive. (Because the CURRENT CLOCK used by the data shift register is derived from the R80 READ CLOCK input, the data loaded into the data shift register is serialized and asserted to the header/data comparator in sync with each bit of the R80 READ DATA input.) The data shift register serializes and asserts bits 0 through 7 of the first data byte to the header/data comparator.

After bit 7 of the first data byte has been asserted to the header/data comparator, the microcontroller loads the second byte of data from the selected FIFO into the data shift register and increments the selected FIFO address counter.

After bit 7 of the second data byte has been serialized and asserted to the header/data comparator, the microcontroller loads the third data byte from the selected FIFO into the data shift register and increments the FIFO A address counter. This process is repeated until all 512 bytes of data from the selected FIFO have been serialized and asserted to the header/data comparator for comparison with the data portion of the R80 READ DATA input. (A detailed discussion of how the microcontroller causes serialization of data from the data buffers is provided in Paragraph 3.5.11.)

The header/data comparator performs a bit-by-bit comparison of the data portion of the R80 READ DATA input with the DSR0 input to determine if the data stored on the disk match the serialized data from the data shift register.

Each bit of the data asserted to the header/data comparator for comparison with the R80 READ DATA is asserted also to the ECC/CRC logic via the NRZ data formatter and NRZ data bus. The ECC/CRC logic generates a 32-bit ECC word based on the configuration of the 4096 data bits asserted to the ECC/CRC logic via the DSR0 input to the NRZ data formatter.

After the 512 bytes of data from the FIFO have been serialized and asserted to the header/data comparator for comparison with the data portion of the R80 READ DATA input and to the ECC/CRC logic for generation of a 32-bit ECC word, (the FIFO A address counter has been incremented to its maximum count, FIFO MAX is asserted to the microcontroller), the microcontroller strobes the header data comparator to sample the results of the data comparison and enables the ECC/CRC logic to load the 32-bit ECC word of the R80 READ DATA input. If the data did not compare, the MISMATCH output of the header/data comparator will be low. If the data from the data buffer matched the data portion of the R80 READ DATA, the MISMATCH output will be high. The MISMATCH output is asserted to the status logic in the CSR and to the microcontroller. After the ECC word has been loaded, the microcontroller enables the ECC/CRC logic to compare the two ECC words (the ECC word generated from the 4096 bits of data used for comparison with the 4096 bits of R80 READ DATA with the ECC word read from the disk drive). Also, the microcontroller enables the P2 CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control to synchronize IDC operation with the CPU.

If an ECC comparison error is indicated (the ECC/CRC ERROR signal is asserted to the microcontroller and CSR), the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 to signal the CPU. Then the IDC returns to the idle mode of operation. [Note that if the results of the data comparison (the 4096 bits of R80 READ DATA with the 4096 bits of data from the FIFO) did not match, then an ECC error will also occur. Therefore, it is not necessary to terminate the write check function when a data comparison error is detected. However, if the write check function is terminated by an ECC error, the CPU can determine if the error was ECC related or a data comparison error by reading the IDC staus word. The results of the data comparison (MISMATCH) and ECC comparison (CRC/ECC ERROR) are made available to the CPU via the IDC status logic.]

If no ECC comparison error is indicated, the microcontroller clears the selected FIFO address counter and enables the PORT XFER REQ output of the CSR to be asserted to the CPU. The PORT XFER REQ signal signifies to the CPU that the write check function has been performed on the requested sector of data and that the data comparison was valid.

If the write check function is to be performed on the next sector of data, the CPU asserts an XFER GRANT signal to the CSR. The XFER GRANT input resets the PORT XFER REQ output. When the PORT XFER REQ output is reset, the microcontroller changes the FIFO selected and monitors the CRDY output of the CSR. If the write check function is to be continued, the CRDY output of the CSR will have remained cleared, and the microcontroller will increment the read/write data address in the disk address register and reset the function timer. Then the microcontroller checks the DRIVE RDY input. If the R80 disk drive is operational, the microcontroller reinitiates the R80 write check function to compare the next sector of the data from the R80 disk drive with the data contained in the selected FIFO.

If no further data are to be read, the CPU responds to the PORT XFER REQ input by loading an IDC control word with CRDY set and then asserting XFER GRANT. The XFER GRANT input to the IDC resets the PORT XFER REQ signal. When the PORT XFER REQ signal is reset, the microcontroller monitors the CRDY output of the CSR. If the CRDY output is set, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

3.4.6 Read Data Without Header Check

The read data without header check function specified in Table 3-1 may be initiated by the CPU by loading the CSR with the applicable IDC control word. Since the operational sequence of the IDC in performing the read data without header check function depends on whether an RL02 or R80 disk drive is selected, the read data without header check functions are discussed separately as follows.

3.4.6.1 RL02 Read Data Without Header Check – When an RL02 read data without header check function is specified in the IDC control word, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output. The microcontroller then selects FIFO A and resets the FIFO A address counter. Next the microcontroller checks the DRIVE RDY input to determine if the selected disk drive is ready (the disk drive is operational and not busy performing a seek function). If the DRIVE RDY input is present or when it is asserted, the microcontroller enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. This synchronizes the IDC operation with the selected RL02 disk drive.

The microcontroller then loops until the leading edge of the SYNC SECTOR PLS is detected. This pulse is generated by the RL SECTOR PLS input from the RL02 disk drive. Presence of the SYNC SECTOR PLS indicates that the applicable read/write head of the RL02 disk drive is positioned at the

beginning portion of a data sector. After the leading edge of the SYNC SECTOR PLS is detected, the microcontroller loops until the trailing edge of the SYNC SECTOR PLS is detected. Then the microcontroller loops until 32 RL SYSTEM CLOCK (4.1 megahertz) pulses have been asserted to the microcontroller via the SEQUENCE CLOCK output of the clock control. This second microcontroller loop is inititated to prevent the read data separator from trying to achieve phase lock on data that may contain glitches. After the loop, the microcontroller enables the read data separator and then loops until after 32 RL SYSTEM CLOCK (4.1 megahertz) pulses have been asserted to the microcontroller via the SEQUENCE CLOCK output of the clock control. This loop is initiated to allow time for the read data separator to achieve phase lock on the data being read from the disk (RL READ DATA input). Phase lock is achieved by reading a sequence of four bytes of zeros in the header preamble of the RL READ DATA. (See Figure 2-20 for the RL READ DATA format.)

After the loop for phase lock, the microcontroller presets the conditions for locating the header sync byte of the RL READ DATA.

To preset the conditions for locating the header sync byte, the microcontroller clears the data shift register and presets the CONSTANTS output of the microcontroller to the header sync byte pattern. Then the microcontroller selects the DS CLOCK for synchronization. The DS CLOCK is generated from the RL READ DATA input and thus synchronizes the IDC with the selected RL02 disk drive data rate. When the DS CLOCK from the read data separator is selected, the DS CLOCK is asserted on the CURRENT CLOCK output of the clock control.

The DS CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the header sync byte has been found. (A detailed discussion of how the header sync byte is located is provided in Paragraph 3.5.4.)

When the RL READ DATA header sync byte is found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the DS CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller, which then loops until the 48 bits comprising the address information, the 16 bits of the zeros that follow, and the CRC word of the RL02 header portion of the RL READ DATA have been bypassed.

After the header portion of the RL READ DATA has been bypassed, the microcontroller checks to make certain that the selected FIFO is empty. If the FIFO is full (FIFO MAX is asserted to the microcontroller), the microcontroller clears the MISMATCH output of the header/data comparator and sets the CRDY and Data Late (DLT) error bits in the CSR. If the IE bit of the previous IDC control word was set, the microcontroller also generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

If the FIFO is empty, the microcontroller deselects the DS CLOCK and enables the RL SYSTEM CLOCK (4.1 megahertz) to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. The microcontroller then loops until the write splice area within the header gap of the RL READ DATA has passed the read/write heads of the RL02 disk drive. Then the microcontroller clears the ECC/CRC logic and enables the read data separator.

Next, the microcontroller loops until 32 RL READ DATA pulses have been asserted to the IDC. This loop is initiated to allow the read data separator to achieve phase lock on the data being read from the disk. After the phase lock loop, the microcontroller clears the selected FIFO address counter and presets and asserts the CONSTANTS output of the microcontroller to the header/data comparator. (The CONSTANTS output is preset to the bit configuration of the RL READ DATA header preamble sync byte.) The microcontroller then enables the DS CLOCK to be asserted on the CURRENT CLOCK output of the clock control. The DS CLOCK is not asserted on the SEQUENCE CLOCK output of the

clock control until the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the RL READ DATA header preamble sync byte has been found. (A detailed discussion of how the sync byte is located is provided in Paragraph 3.5.4.)

Detection of the sync byte of the header preamble signals the start of the data segment of the sector to be read. When the header preamble sync byte has been found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the DS CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller, which then enables the ECC/CRC logic, and begins converting the data portion of the RL READ DATA input into byte format and storing the 256 bytes of RL READ DATA into the selected FIFO. (A detailed discussion of how the READ DATA are converted to byte format and stored in the data buffers is provided in Paragraph 3.5.12.)

Each bit of the 256 bytes of the data portion of the RL READ DATA is used in the ECC/CRC logic to generate a 16-bit CRC word representative of the bit configuration of the RL READ DATA. After all 256 bytes of RL READ DATA have been loaded into the selected FIFO (FIFO MAX is asserted to the microcontroller), the microcontroller enables the 16-bit CRC word of the RL READ DATA input to be loaded into the ECC/CRC logic. After the CRC word has been loaded, the microcontroller enables the ECC/CRC logic to compare the CRC word generated from the 256 bytes of RL READ DATA with the CRC word read from the disk. Next, the microcontroller deselects the DS CLOCK and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. Then, the microcontroller monitors the CRC/ECC ERROR signal output of the ECC/CRC logic.

If a CRC/ECC error is indicated, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

If no CRC/ECC ERROR is indicated, the microcontroller clears the selected FIFO address counter and generates and asserts the PORT XFER REQ output of the CSR to the CPU. This signal signifies that the IDC has completed reading a sector of data and that the data are ready for transfer to the CPU.

If more data are to be read, the CPU asserts a XFER GRANT signal to the CSR. When the XFER GRANT signal is asserted, the PORT XFER REQ output is reset. When the PORT XFER REQ is reset, the microcontroller changes the FIFO selected and monitors the CRDY output of the CSR. If the CRDY output of the CSR has remained cleared, the microcontroller resets the function timer, and reinitiates the RL02 read data without header check function to read the next sector of RL READ DATA and store the data portion in the selected FIFO.

If no further data are to be read, the CPU responds to the PORT XFER REQ input by loading an IDC control word with CRDY set and then asserting XFER GRANT. The XFER GRANT input to the IDC resets the PORT XFER REQ signal. When the PORT XFER REQ signal is reset, the microcontroller monitors the CRDY output of the CSR. If the CRDY output is set, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

3.4.6.2 R80 Read Data Without Header Check – When an R80 read data without header check function is specified by the IDC control word, the microcontroller branches on the F0, F1, and F2 inputs to preset the microcontroller microword output. The microcontroller then selects FIFO A and resets the FIFO A address counter. Next, the microcontroller checks the DRIVE RDY input to determine if the

disk drive is ready (the R80 disk drive is operational and not busy performing a seek). If the DRIVE RDY input is present or when it is asserted, the microcontroller enables the R80 SERVO CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control. This synchronizes the operation of the IDC with the R80 disk drive.

The microcontroller then loops until the leading edge of the SYNC SECTOR PLS is detected. This pulse is generated by the R80 SECTOR PLS or R80 INDEX PLS inputs from the R80 disk drive. Presence of the SYNC SECTOR PLS indicates that the applicable read/write head of the R80 disk drive is positioned at the beginning portion of the header data. After the leading edge of the SYNC SECTOR PLS is detected, the microcontroller loops until 60 R80 SERVO CLOCK pulses (9.677 megahertz) have been asserted. The microcontroller loop is initiated to prevent the R80 disk drive from trying to achieve phase lock on data that may contain glitches. After the loop, the microcontroller enables the TAG bus control to assert the READ TAG and CNTL TAG signals to the R80 drivers. These signals enable the R80 drivers to assert the R80 TAG BUS 01 and R80 TAG 3 outputs of the IDC (assert a read gate command to the R80 disk drive). The read gate command enables the R80 disk drive to read the data from the disk and assert the data to the IDC via the R80 READ DATA signal line. The R80 disk drive also generates and asserts the R80 READ CLOCK, which is synchronized with the READ DATA input to the IDC.

After the read gate command has been asserted, the microcontroller loops until after 88 R80 SERVO CLOCK pulses have been asserted to the IDC. This loop is initiated to allow time for the R80 disk drive to achieve phase lock on the data being read from the disk. Phase lock is achieved by reading a sequence of zeros in the sector gap of the R80 READ DATA. (See Figure 2-17 for the R80 READ DATA format.)

After the loop for phase lock, the microcontroller presets the conditions for locating the header sync byte of the R80 READ DATA.

To preset the conditions for locating the header sync byte, the microcontroller clears the data shift register and presets the CONSTANTS output of the microcontroller to the header sync byte pattern. Then the microcontroller selects the R80 READ CLOCK for synchronization. The R80 READ CLOCK is generated within the R80 disk drive from the R80 READ DATA and thus synchronizes the IDC with the R80 disk drive data rate. When the R80 READ CLOCK is selected, the R80 READ CLOCK is asserted on the CURRENT CLOCK output of the clock control. The R80 READ CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until the sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the R80 READ DATA header sync byte has been found. (A detailed discussion of how the sync byte is located is provided in Paragraph 3.5.4.)

When the R80 READ DATA header sync byte is found, the SYNC SEEN output of the header/data comparator is asserted to the clock control to enable the R80 READ CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller, which then loops until the 48 bits comprising the address information and CRC word of the header portion of the R80 READ DATA have been bypassed.

After the header portion of the R80 READ DATA has been bypassed, the microcontroller selects the R80 SERVO CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. Then the microcontroller checks to make certain that the FIFO is empty. If the FIFO is full (FIFO MAX is asserted to the microcontroller), the microcontroller clears the MISMATCH output of the header/data comparator and sets the CRDY and Data Late (DLT) error bits in the CSR. If the IE bit of the previous IDC control word was set, the microcontroller also generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

If the FIFO is empty, the microcontroller causes the TAG bus control to deassert the read gate command from the R80 disk drive. The microcontroller then loops until the write splice area within the header gap has passed the read/write heads of the R80 disk drive. Then, the microcontroller enables the tag bus control to reassert the read gate command to the R80 disk drive. Next the microcontroller clears the ECC/CRC logic.

After the read gate command is asserted to the R80 disk drive, the microcontroller loops until 88 R80 SERVO CLOCK pulses have been asserted to the IDC. This loop is initiated to allow the R80 disk drive to achieve phase lock on the data being read from the disk. After the phase lock loop, the microcontroller clears the selected FIFO address counter, and presets and asserts the CONSTANTS output of the microcontroller to the header/data comparator. (The CONSTANTS output is preset to the bit configuration of the R80 READ DATA header gap sync byte.) The microcontroller then enables the R80 READ CLOCK to be asserted on the CURRENT CLOCK output of the clock control. The R80 READ CLOCK is not asserted on the SEQUENCE CLOCK output of the clock control until the R80 READ DATA header gap sync byte has been found (when SYNC SEEN from the header/data comparator is asserted to the clock control). Thus, the microcontroller is forced to stall until the R80 READ DATA header gap sync byte has been found. (A detailed discussion of how the sync byte is found is provided in Paragraph 3.5.4.)

Detection of the R80 READ DATA header gap sync byte signals the start of the data segment of the sector to be read. When the R80 READ DATA header gap sync byte has been found, the SYNC SEEN output of the header/ data comparator is asserted to the clock control to enable the R80 READ CLOCK to be asserted on the SEQUENCE CLOCK output. This restarts the microcontroller, which then enables the ECC/CRC logic, and begins converting the data portion of the R80 READ DATA into byte format and storing the 512 bytes of R80 READ DATA in the selected FIFO.

After the data shift register has been loaded with the first eight bits of R80 READ DATA, the microcontroller enables the parallel output of the data shift register to be asserted to the input of the FIFO(s) via the read data tristate drivers, loads the data byte into the selected FIFO, and increments the selected FIFO address counter. This process (converting the R80 READ DATA to byte format and storing each byte) is repeated until all 512 bytes of the data portion of the R80 READ DATA have been written into the selected FIFO. (A detailed discussion of how the READ DATA are converted to byte format and stored in the data buffer is provided in Paragraph 3.5.12.)

Each bit of the 512 bytes of R80 READ DATA is used in the ECC/CRC logic to generate a 32-bit ECC word representative of the bit configuration of the data portion of the R80 READ DATA. After all 512 bytes of R80 READ DATA have been loaded into the selected FIFO (FIFO MAX is asserted to the microcontroller), the microcontroller enables the 32-bit ECC word of the R80 READ DATA to be loaded into the ECC/CRC logic. After the ECC word has been loaded, the microcontroller enables the ECC/CRC logic to compare the ECC word generated from the 512 bytes of R80 READ DATA with the ECC word read from the disk. Then the microcontroller clears the FIFO address counter and monitors the CRC/ECC ERROR signal output of the ECC/CRC logic.

If a CRC/ECC error is indicated, the microcontroller initiates an ECC correction routine. At the completion of the correction routine, the results of the correction computation are indicated in the STAT 0 and STAT 1 signals that are asserted to the status logic of the CSR. On completion of the correction computation, the microcontroller deselects the R80 READ CLOCK and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT clock outputs of the clock control. Then the microcontroller clears the selected FIFO address counter, sets the CRDY output of the CSR, and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

If no CRC/ECC ERROR is indicated, the microcontroller deselects the R80 READ CLOCK and enables the P2 CLOCK to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs of the clock control. This synchronizes the IDC with the CPU. Then the microcontroller generates and asserts the PORT XFER REQ output of the CSR to the CPU. This signal signifies that the IDC has completed reading one sector of data and that the data are ready for transfer to the CPU.

If more data are to be read, the CPU asserts a XFER GRANT signal to the CSR. When the XFER GRANT signal is asserted, it resets the PORT XFER REQ output. When the PORT XFER REQ is reset, the microcontroller changes the FIFO selected and monitors the CRDY output of the CSR. If more data are to be read, the CRDY output of the CSR will have remained cleared, and the microcontroller will reset the function timer and reinitiate the R80 read data without header check function to read the next sector of data from the R80 disk drive and store the data in the selected FIFO.

If no further data are to be read, the CPU responds to the PORT XFER REQ input by loading an IDC control word with CRDY set and then asserting XFER GRANT. The XFER GRANT input to the IDC resets the PORT XFER REQ signal. When the PORT XFER REQ signal is reset, the microcontroller monitors the CRDY output of the CSR. If the CRDY output is set, the microcontroller sets the CRDY output of the CSR and, if the IE bit of the previous IDC control word was set, generates and asserts a UBUS BR5 signal to the CPU. The IDC then returns to the idle mode of operation.

3.4.7 Write Format

The write format is used only to format R80 disk drive headers. When the write format function is initiated, the IDC checks to make certain that the R80 disk drive is ready (that it is operational and not busy performing a seek). If the R80 disk drive is ready or when it becomes ready, the IDC waits until the R80 INDEX PLS from the R80 disk drive is detected (waits for the beginning of sector 0).

After the R80 INDEX PLS is detected, the IDC then

- writes a sequence of zeros (224),
- writes the header sync byte,
- writes four bytes of header data from the data buffer,
- writes the header CRC word, which was generated in the IDC from the header data written to the disk drive,
- writes a sequence of zeros (136),
- writes the header gap sync byte,
- writes a sequence of zeros (4096),
- writes the ECC word which was generated in the IDC while the sequence of zeros (4096) were being written, and
- writes a sequence of zeros.

The IDC then waits for the leading edge of the next sector pulse. After the next sector pulse is detected, the IDC repeats the write sequence just specified. This process is repeated until the R80 INDEX PLS is again detected. Then the IDC generates and asserts, if enabled, a UBUS BR5 interrupt to the CPU, which indicates that the specified function has been completed.

3.4.8 Idle Mode

The IDC idle mode of operation is entered automatically after the completion of a CPU-specified function. The IDC remains in the idle mode until another IDC function is specified by the CPU (the CRDY L input to the microcontroller is set to a high). When the idle mode is entered, the microcontroller generates and asserts the UDRV SEL 0, UDRV SEL 1, and UDRV SEL signals to the CSR (see Figure 3-1). These signal inputs are used to generate the DRIVE SEL 0 and 1 signals, which select which one of the disk drives is to be enabled. These signals also enable the operational status signal inputs from the selected disk drive (RL DRIVE RDY, RL DRIVE ERR, R80 DRIVE RDY, R80 PLUG VALID, R80 ON CYLINDER, and R80 FAULT) to control assertion of the DRIVE RDY and DRIVE ERR signals to the microcontroller. (A detailed discussion of how the disk drives are selected and the status signals are asserted to the microcontroller is provided in Paragraph 3.5.1.) The DRIVE SEL 0 and 1 signals also enable the appropriate ONLINE signal output of the CSR to be asserted to the microcontroller. (The ONLINE signal contained in the CSR is set and cleared by the microcontroller to provide a record of which drives are currently in use. A discussion of the ONLINE register contained in the CSR is provided in Paragraph 3.5.10.)

After the disk drive has been selected, the microcontroller branches on the DRIVE RDY, DRIVE ERR, and ONLINE signals asserted and generates the control signals (USET ATTN L, USET ONLINE L, and UCLEAR ONLINE L) to record the disk drive status. The USET and UCLEAR ONLINE signals are asserted to the CSR to record the sampled disk drive status during the monitoring period. The USET ATTN signal is asserted to the CSR to record that the enabled disk drive has changed operational status (has gone off-line, has come back on-line, or is reporting an error). (A detailed discussion of the function of the ATTN and ONLINE registers contained in the CSR is provided in Paragraph 3.5.10.)

After the operational status of the selected disk drive has been sampled and recorded, the microcontroller increments the address count (UDRV SEL 0 and 1) and reasserts USEL DRV to enable the next disk drive. When the next disk drive is enabled, the microcontroller again branches on the DRIVE ERR, DRIVE RDY, and ONLINE inputs and records the status of the enabled disk drive. After all disk drives have been sampled, the microcontroller checks the CRDY L input from the CSR to determine if the CPU has requested the IDC to perform a function. If no function has been requested (the CRDY L input is L), the microcontroller repeats the idle mode routine.

3.5 DETAILED FUNCTIONAL LOGIC DESCRIPTIONS

3.5.1 Disk Drive Selection and Drive Status Monitor

The disk drive selection and drive ready monitor logic is used to enable, if applicable, one of the RL02 disk drives, to condition the IDC logic for operation with either the R80 disk drive or an RL02 disk drive, and to monitor the status of the selected disk drive.

The RL02 disk drives are connected to the IDC in a daisy chain. Each RL02 disk drive is preprogrammed with a specific address by installing an address plug. Each RL02 disk drive is enabled by the IDC by asserting the configuration of DRIVE SEL 0 and 1 bits which matches the preprogrammed address. When an RL02 disk drive is enabled, it asserts to the IDC its operational status information (RL DRIVE RDY or RL DRIVE ERROR). The RL DRIVE RDY signal is only asserted if the selected disk drive is operational and is not busy performing a seek. When an RL02 disk drive is enabled, operational, and not performing a seek function, the selected RL02 disk drive enables the read data from the disk and the sector pulses to be asserted to the IDC via the RL READ DATA and RL SECTOR PLS signal lines.

The R80 disk drive is enabled at all times and continuously asserts its status information (including R80 ON CYLINDER and R80 DRIVE RDY or R80 FAULT), R80 SECTOR PLS, and R80 INDEX PLS outputs to the IDC. Unlike the RL02 disk drives, the R80 disk drive asserts its address information to the IDC via the R80 SEL ADDRESS 0 and 1 signal lines. When the configuration of DRIVE SEL 0 and 1 bits match the address of the R80 disk drive, the IDC generates an R80 signal. The R80 signal conditions the IDC logic for operation with the R80 disk drive. Again, unlike the RL02 disk drives, the R80 does not assert its READ DATA output to the IDC when it is selected. A separate command from the IDC TAG bus control (read gate) must be asserted before the R80 READ DATA output is asserted to the IDC.

A functional block diagram of the disk drive selection and drive ready monitor logic is presented in Figure 3-2. Figure 3-2 shows the maximum number and configuration of disk drives that may be connected to the IDC: three RL02 disk drives and one R80 disk drive, or four RL02 disk drives.

The disk drive selection and drive status monitor generates the appropriate DRIVE SEL 0 and 1 signals that are used to enable, if applicable, the appropriate RL02 disk drive, to control generation of the RL02 and R80 signals, and to enable the appropriate DRIVE RDY or DRIVE ERR input signal to be asserted to the microcontroller.

3.5.1.1 Generation of DRIVE SEL 0 and 1 – When performing a CPU-specified function, the IDC generates the DRIVE SEL 0 and 1 signals from bits 08 and 09 of the IDC control word input that is loaded into the control register of the CSR. When the IDC is operating in the idle mode, the DRIVE SEL 0 and 1 outputs are generated from the UDRV SEL, UDRV SEL 0, and UDRV SEL 1 outputs of the microcontroller.

3.5.1.2 Generation of RL02 and R80 – If the configuration of DRIVE SEL 0 and 1 signals asserted to the R80 address compare logic match the R80 SELECT ADDRESS 0 and 1 signals asserted from the R80 disk drive, and if an R80 PLUG VALID signal is asserted (the R80 disk drive is installed as part of the RB730 disk subsystem and an address plug is installed), the RL02 output will be a low. The RL02 output is inverted to produce the R80 signal. If the address does not compare or an R80 disk is not installed, the RL02 output will be high and the R80 signal will be low.

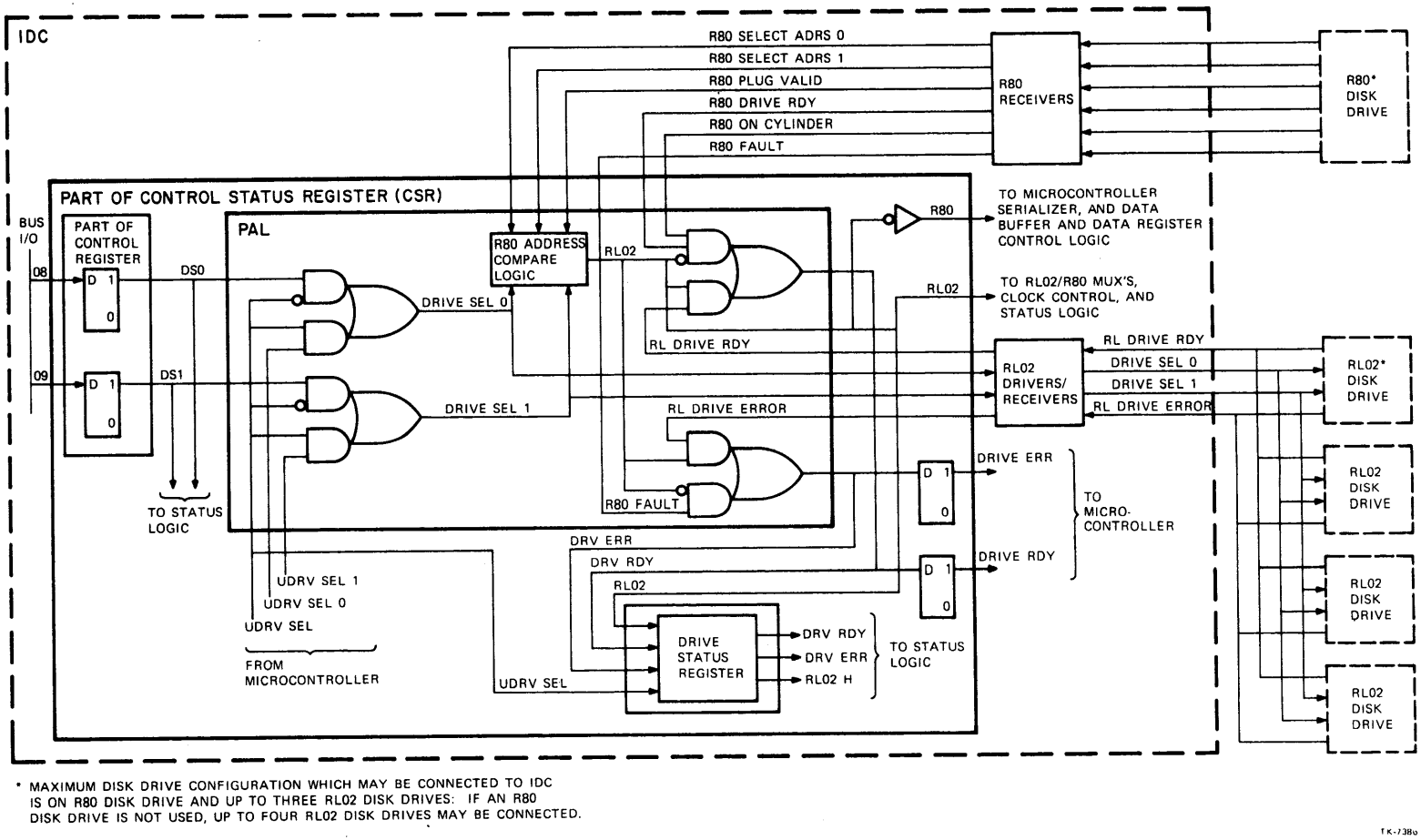
3.5.1.3 Gating DRIVE RDY – The RL02 output of the R80 address compare logic is used to enable either the R80 DRIVE RDY or RL DRIVE RDY input to be asserted to the DRIVE RDY input of the microcontroller.

3.5.1.4 Gating DRIVE ERR – The RL DRIVE ERROR or R80 FAULT output of the selected disk drive is enabled on the DRIVE ERR output of the disk drive selection and drive status monitor by the RL02 output of the R80 address compare logic. The DRIVE ERR output is asserted to the IDC microcontroller and to the IDC status logic.

3.5.2 TAG Bus Control Logic

The TAG bus control logic operates from microcontroller inputs to assert the following commands to the R80 disk drive via the R80 TAG and R80 TAG BUS signal lines.

R80 seek
R80 head select
R80 recalibrate
Read gate
Write gate



* MAXIMUM DISK DRIVE CONFIGURATION WHICH MAY BE CONNECTED TO IDC IS ON R80 DISK DRIVE AND UP TO THREE RL02 DISK DRIVES: IF AN R80 DISK DRIVE IS NOT USED, UP TO FOUR RL02 DISK DRIVES MAY BE CONNECTED.

TK-7380

Figure 3-2 Disk Drive Selection and Drive Status Monitor

3.5.2.1 Asserting R80 Seek, Head Select, and Recalibrate Commands – When the appropriate R80 seek (Figure 2-5), R80 head select (Figure 2-6), or R80 recalibrate (Figure 2-7) command is loaded into the IDC disk address register and an R80 seek function is specified by the IDC control word input, the microcontroller controls gating of the appropriate command to the R80 disk drive. A functional block diagram of the TAG bus control logic is shown in Figure 3-3.

To assert the appropriate seek, head select, or recalibrate command contained in the disk address register, the microcontroller asserts a USEEK INSTR signal and a UTAG STROBE signal to the TAG bus control logic. Timing of the USEEK INSTR and UTAG STROBE inputs to the TAG bus control logic is shown in Figure 3-3.

3.5.2.2 Asserting Read Gate – The read gate command input to the R80 disk drive is initiated by holding the R80 TAG 3 input high and asserting the R80 TAG BUS 1 input. The read gate command is terminated when the R80 TAG BUS 1 input is deasserted.

To initiate a read gate command, the microcontroller asserts a UTAG STROBE, which enables the R80 TAG 3 output (see Figure 3-3). Next, the microcontroller asserts a UENB LOOP LOCK signal, which enables the R80 TAG BUS 1 output. The microcontroller may terminate the read gate command by deasserting the UENB LOOP LOCK input to the TAG bus control logic. Figure 3-2 shows the timing relationship of the UTAG STROBE and UENB LOOP LOCK input signals.

3.5.2.3 Asserting Write Gate – The write gate command input to the R80 disk drive is initiated by holding the R80 TAG 3 input high and asserting the R80 TAG BUS 0 input. The write gate command is terminated when the R80 TAG BUS 0 input is deasserted.

To initiate a write gate command, the microcontroller asserts a UTAG STROBE, which enables the R80 TAG 3 output (see Figure 3-3). Next the microcontroller asserts a UWRITE GATE signal, which enables the R80 TAG BUS 0 output. The microcontroller may terminate the write gate command by deasserting the UWRITE GATE input to the tag bus control logic. Figure 3-2 shows the timing relationship of the UTAG STROBE and UWRITE GATE input signals.

3.5.3 Clock Control Logic

The clock control logic is used to synchronize operation of the IDC with the CPU, the R80 disk drive, or the RL02 disk drive, as applicable. It is also used to inhibit the sequence clock output and thus stall the IDC microcontroller until the sync byte of the read data input is detected.

A functional block diagram of the clock control logic is shown in Figure 3-4. A timing diagram showing the periods of the input clocks and the controlled gating of the clocks to the CURRENT CLOCK and SEQUENCE CLOCK outputs is presented in Figure 3-5.

As shown in Figure 3-4, the RL02 input to the clock control enables the appropriate RL02 or R80 disk drive clock inputs to be asserted as the SYS CLOCK or DISK CLOCK inputs to the CURRENT CLOCK and SEQUENCE CLOCK gates. The P2 CLOCK L and the RL STATUS CLOCK inputs are asserted directly to the CURRENT CLOCK and SEQUENCE CLOCK gates.

To change from one synchronizing clock to another, the microcontroller asserts a UCHANGE CLKSRC H input and the applicable clock select signal (USEL SYS CLK, USEL STATUS CLK, USEL CPU CLK, or USEL DSK CLK).

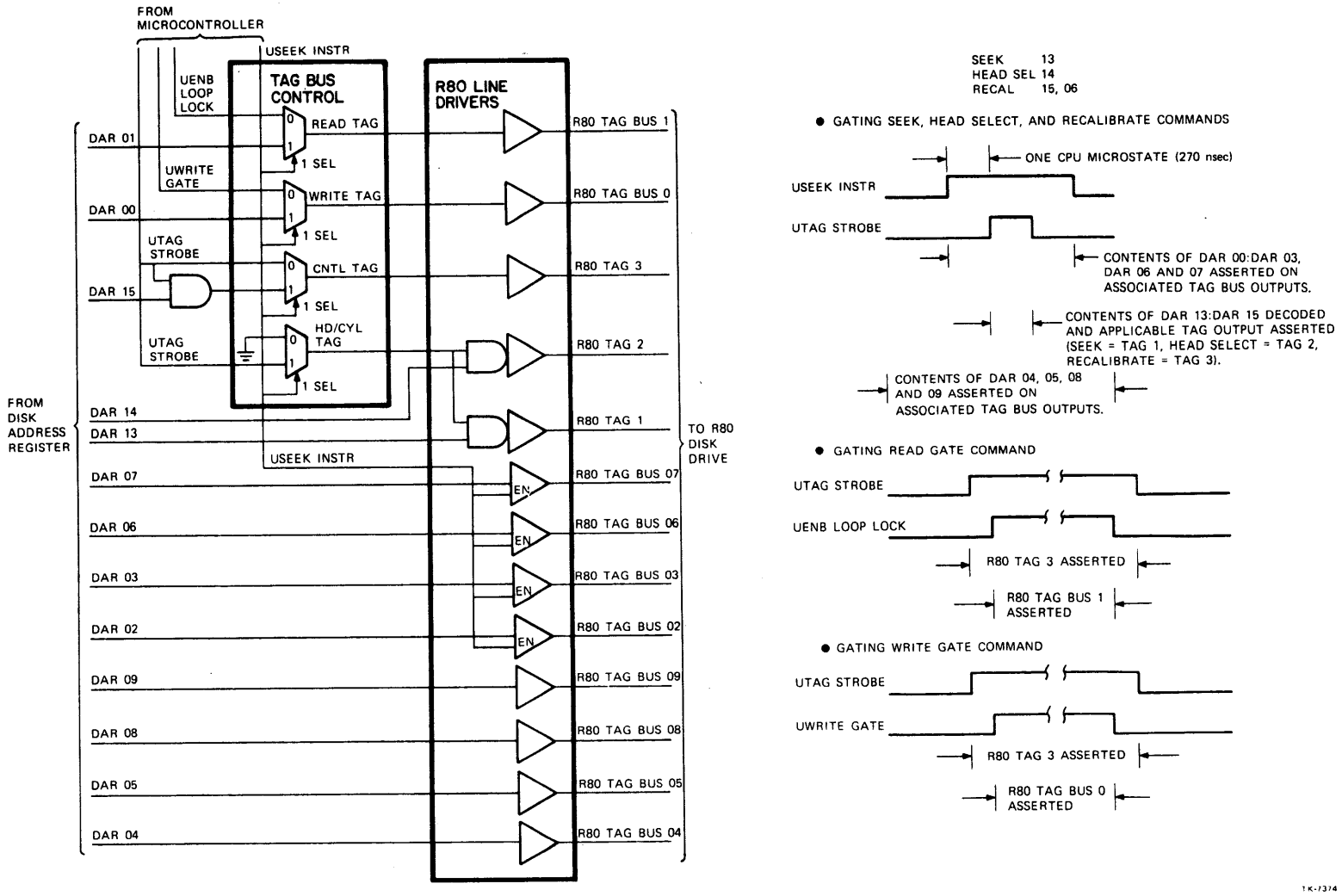


Figure 3-3 TAG Bus Control Logic Functional Block Diagram

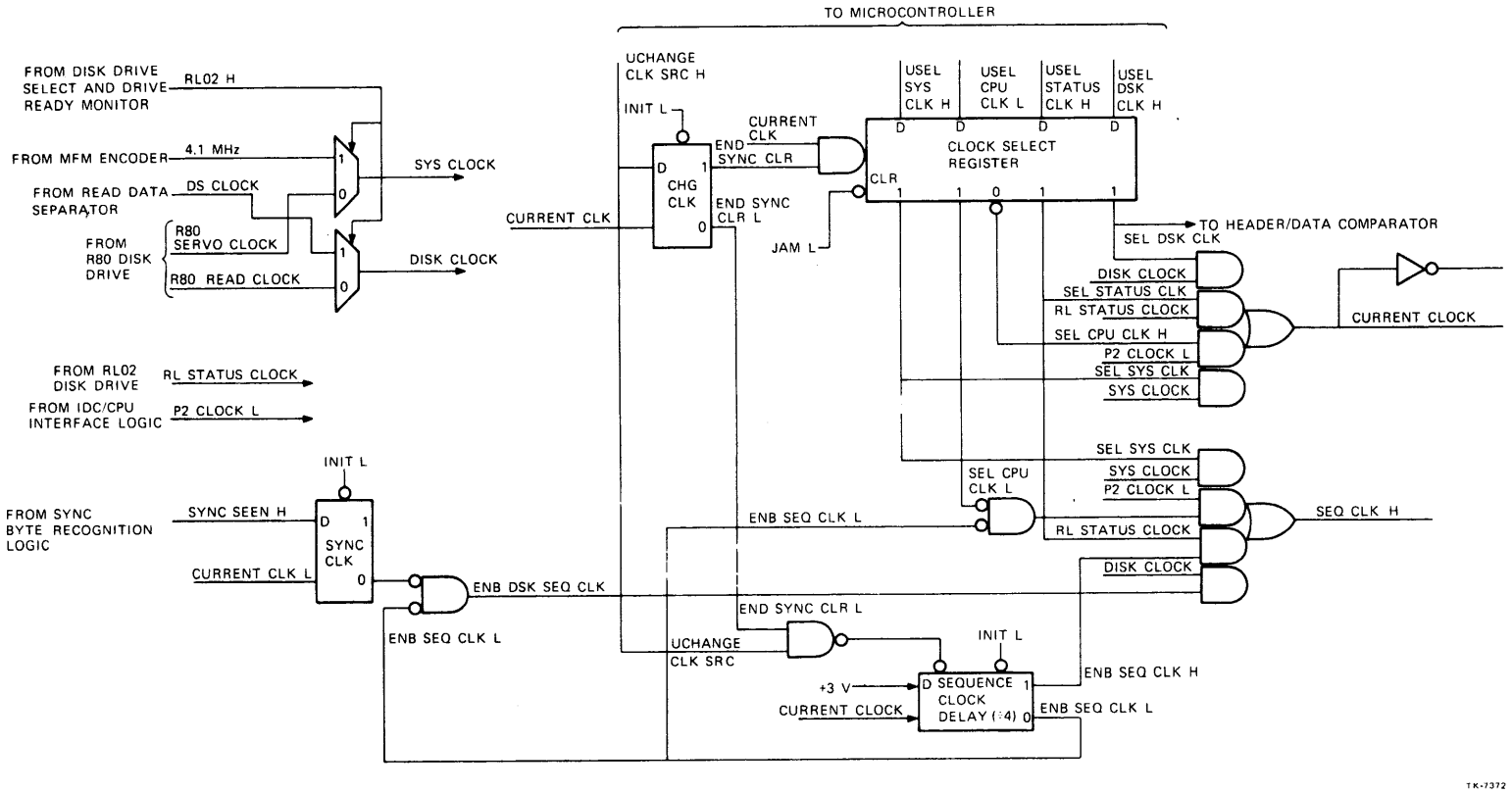


Figure 3-4 Clock Control Logic Functional Block Diagram

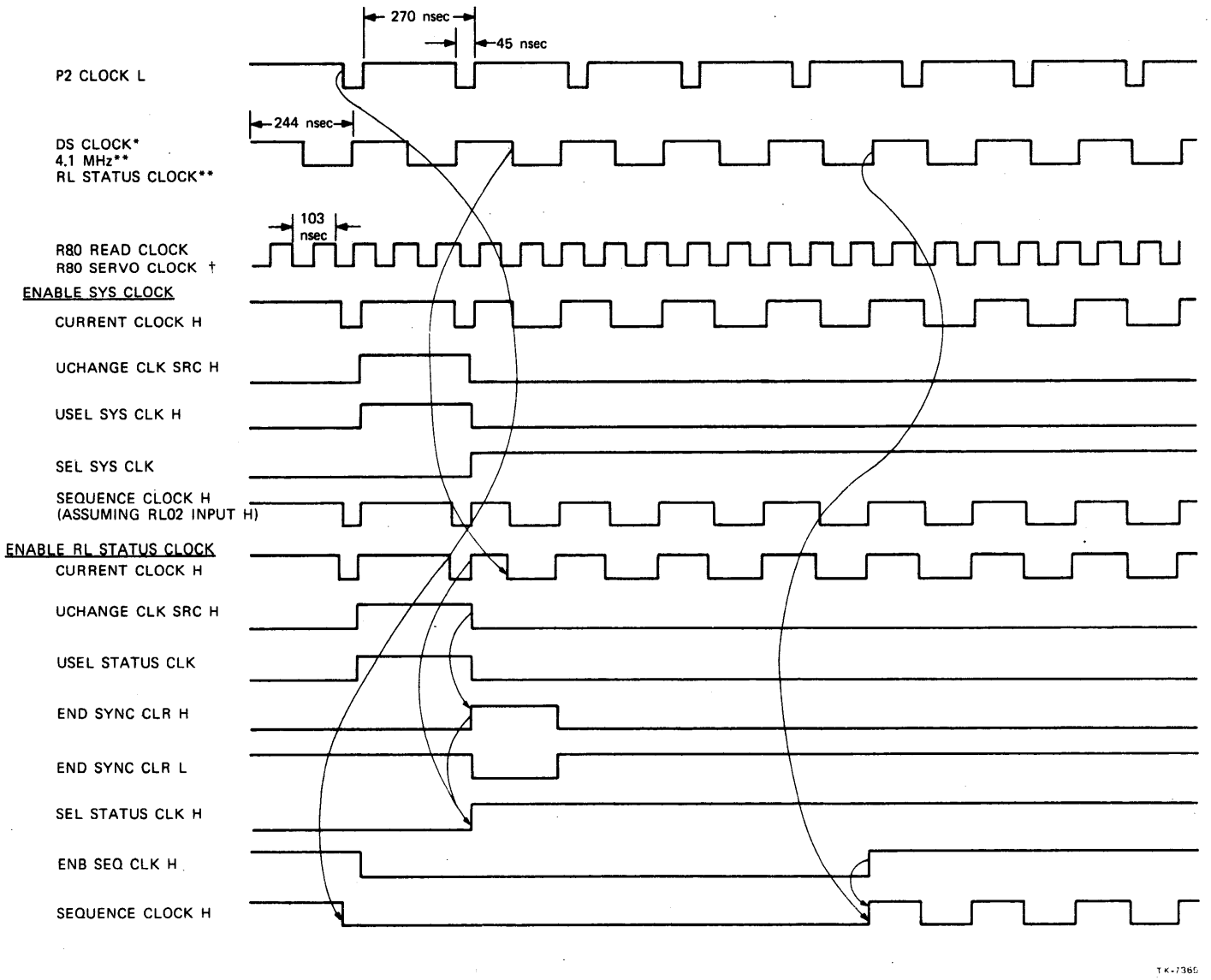
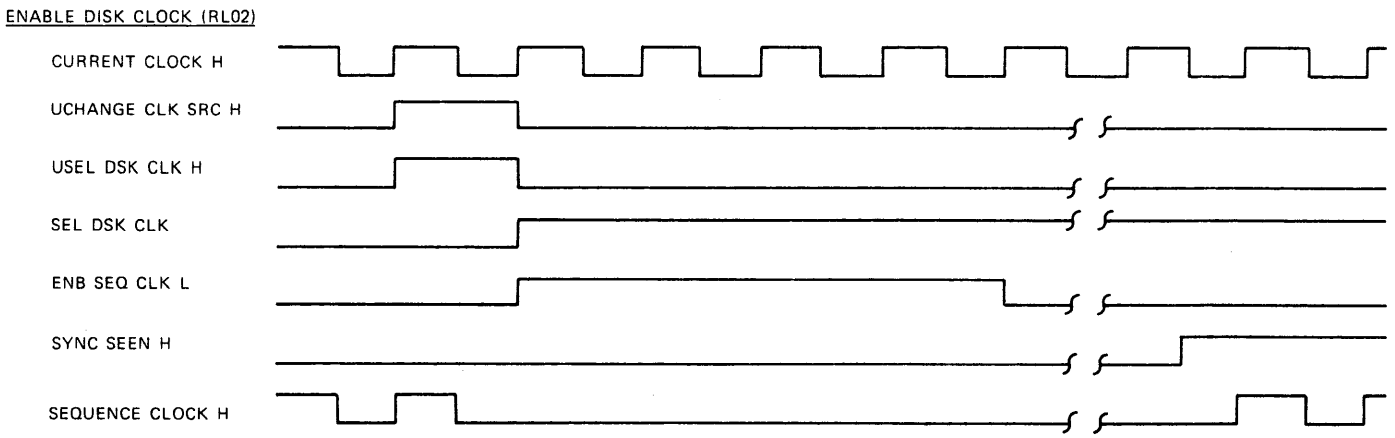


Figure 3-5 Clock Control Logic Timing (Sheet 1 of 2)



* DS CLOCK ALTHOUGH SHOWN TO OCCUR AT A 4.1 MHz RATE (NOMINAL) MAY VARY SLIGHTLY WITH DISK DRIVE SPEED.

** 4.1 MHz CLOCK AND RL STATUS CLOCK OCCUR AT THE 4.1 MHz RATE SHOWN FOR THE DS CLOCK; HOWEVER, THE PHASE RELATIONSHIP OF THESE CLOCKS VARY.
R80 SERVO CLOCK OCCURS AT THE RATE SHOWN FOR THE R80 READ CLOCK; HOWEVER, THE PHASE RELATIONSHIP OF THESE CLOCKS VARY.

† R80 SERVO CLOCK OCCURS AT THE RATE SHOWN FOR THE R80 READ CLOCK; HOWEVER, THE PHASE RELATIONSHIP OF THESE CLOCKS VARY.

NOTE: PHASE RELATIONSHIP OF CLOCK INPUTS TO THE IDC SHOWN ABOVE MAY OR MAY NOT BE AS ILLUSTRATED: THIS DIAGRAM ILLUSTRATES THE VARIOUS PERIODS OF THE CLOCK INPUTS ONLY.

TK-7364

Figure 3-5 Clock Control Logic Timing (Sheet 2 of 2)

3.5.3.1 Enable SYS CLOCK – To enable the SYS CLOCK to be asserted onto the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control logic, the microcontroller asserts UCHANGE CLKSRC H and USEL SYS CLK to the clock control (see Figure 3-4). The UCHANGE CLKSRC H input enables the CHG CLK flip-flop to be set with the following CURRENT CLOCK input. When the CHG CLK flip-flop is set, it asserts an END SYNC CLR signal to the clock gate of the clock select register, which initiates loading of the USEL SYS CLK signal. The resulting SEL SYS clock output is asserted to the CURRENT CLOCK and SEQUENCE CLOCK gates to enable either the 4.1 megahertz clock or R80 SERVO CLOCK inputs, as applicable, to be asserted on the SEQUENCE CLOCK and CURRENT CLOCK outputs. Figure 3-5 illustrates the timing relationships of input and output signals for changing the synchronization clock from the CPU CLK (P2 CLOCK) to the SYS CLOCK (4.1 megahertz clock).

3.5.3.2 Enable RL STATUS CLK or CPU CLOCK – To enable the RL STATUS CLK or CPU CLOCK to be asserted on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control, the microcontroller asserts UCHANGE CLKSRC and USEL RL STATUS CLK or USEL CPU CLK to the clock control (see Figure 3-4).

The UCHANGE CLKSRC H input is asserted to the CHG CLK flip-flop and to the reset gate of the sequence clock delay. The UCHANGE CLKSRC input together with the END SYNC CLR L output of the CHG CLK flip-flop resets the sequence clock delay.

When the sequence clock delay is reset, the ENB SEQ CLK outputs are deasserted from the sequence clock gates, which inhibits the currently selected clock (with the exception of the SYS CLK) from being asserted on the SEQUENCE CLOCK output.

When the UCHANGE CLKSRC signal is loaded into the CHG CLK flip-flop (when the CURRENT CLOCK input goes high), the END SYNC CLR H and L outputs are enabled. The END SYNC CLR H output initiates loading of the applicable clock select input (for this discussion, the USEL CPU CLK or USEL STATUS CLK input) from the microcontroller into the clock select register. The END SYNC CLR L output removes the reset signal from the sequence clock delay.

The applicable clock select (SEL STATUS CLK or SEL CPU CLK L and H) outputs of the clock select register are asserted to the sequence clock and current clock gates. The appropriate SEL STATUS CLK or SEL CPU CLK H output enables the selected clock input to be asserted directly on the CURRENT CLOCK output.

The selected clock input RL STATUS CLOCK or CPU CLOCK (P2 CLOCK L) is not enabled on the SEQUENCE CLOCK output until after the sequence clock delay asserts the applicable ENB SEQ CLK signal to the sequence clock gates.

As noted earlier, the ENB SEQ CLK outputs of the sequence clock delay were reset when the UCHANGE CLKSRC H input was asserted from the microcontroller and was held in the reset state until the UCHANGE CLKSRC input was loaded into the CHG CLK flip-flop. After the reset to the sequence clock delay has been removed, and four positive transitions of the CURRENT CLOCK have been asserted, the ENB SEQ CLK H and L outputs are enabled. These outputs enable the selected RL STATUS CLOCK or CPU CLOCK (P2 CLOCK L) inputs to be asserted on the SEQUENCE CLOCK output.

The timing diagram of Figure 3-5 illustrates the relationships of input and output signals required to change the synchronization clock from the CPU CLOCK to the RL STATUS CLOCK.

3.5.3.3 Enable DISK CLOCK – To enable the DISK CLOCK (R80 READ CLOCK or DS CLOCK) on the CURRENT CLOCK and SEQUENCE CLOCK outputs of the clock control, the microcontroller asserts UCHANGE CLKSRC H and USEL DSK CLK H to the clock control. The DISK CLOCK is enabled on the CURRENT CLOCK output of the clock control in much the same manner as discussed for enabling the CPU CLOCK or RL STATUS CLOCK (Paragraph 3.5.3.2). However, an additional input (SYNC SEEN) to the clock control is necessary before the DISK CLOCK is gated on the SEQUENCE CLOCK output. Gating of the DISK CLOCK to the SEQUENCE CLOCK output is delayed until after SYNC SEEN is asserted (see Figure 3-4). The timing diagram of Figure 3-5 illustrates the relationships of input and output signals required to change the synchronization clock from the SYS CLOCK (4.1 megahertz) to the DISK CLOCK (DS CLOCK).

3.5.4 Sync Byte Recognition Logic

The sync byte recognition logic is used to locate the sync byte of the READ DATA from the selected disk drive. There are two sync bytes in each sector of READ DATA asserted from the disk drives; one directly precedes the header portion of the READ DATA, the second directly precedes the data portion of the READ DATA. The process for locating each of these sync bytes is similar; thus, the following discussion is keyed to locating the sync byte that precedes the data portion of the READ DATA input. This was selected because, when the sync byte has been located and if the IDC is performing a write check function, the SYNC SEEN signal is used to initiate loading the data shift register with the first byte of data to be compared with the READ DATA input. (The SYNC SEEN signal is used to load the data shift registers because the microcontroller, which normally controls loading of the data shift registers, remains in a stall condition until after SYNC SEEN is generated, which would result in misalignment of the data to be compared.)

To locate the sync byte of the READ DATA input, the sync byte recognition logic converts the serial READ DATA input to a parallel format and compares the parallel formatted READ DATA with the sync byte pattern expected (CONSTANTS input from the microcontroller). When a match is determined, a SYNC SEEN signal is generated. A functional block diagram of the sync byte recognition logic is shown in Figure 3-6.

The microcontroller presets the conditions that enable the sync byte to be located. First, the microcontroller generates and asserts the CONSTANTS to the eight-bit checker. For locating the sync byte in the RL02 READ DATA, the CONSTANTS asserted are preset to 80_{16}. For locating the sync byte in the R80 READ DATA, the CONSTANTS are preset to 19_{16}. (CNST 7 of the CONSTANTS input is the most significant digit of the specified sync byte pattern asserted via the CNST 7:0 inputs.) Also, if the IDC is performing a write check function, the microcontroller asserts a UWRT CHK H signal. Then the microcontroller selects the DISK CLK to be asserted on the CURRENT CLOCK output of the clock control. When the DISK CLOCK is selected, the clock control asserts its SEL DSK CLK, CURRENT CLOCK H, and CURRENT CLOCK L outputs to the sync byte recognition logic. Because the DSK CLOCK asserted on the CURRENT CLOCK outputs of the clock control is derived from the READ DATA asserted, the CURRENT CLOCK H and L input are synchronized with each bit of the READ DATA input.

The READ DATA H input to the sync byte recognition logic is asserted to the eight-bit checker and read data synchronizer (see Figure 3-6). The READ DATA H input to the read data synchronizer is sampled at the midpoint of each data bit interval by the CURRENT CLOCK L input, and the condition of the READ DATA H input (a logical 0 or 1) is loaded into the read data synchronizer. A diagram showing the timing relationship of the signals and events discussed in the following paragraphs is presented in Figure 3-7.

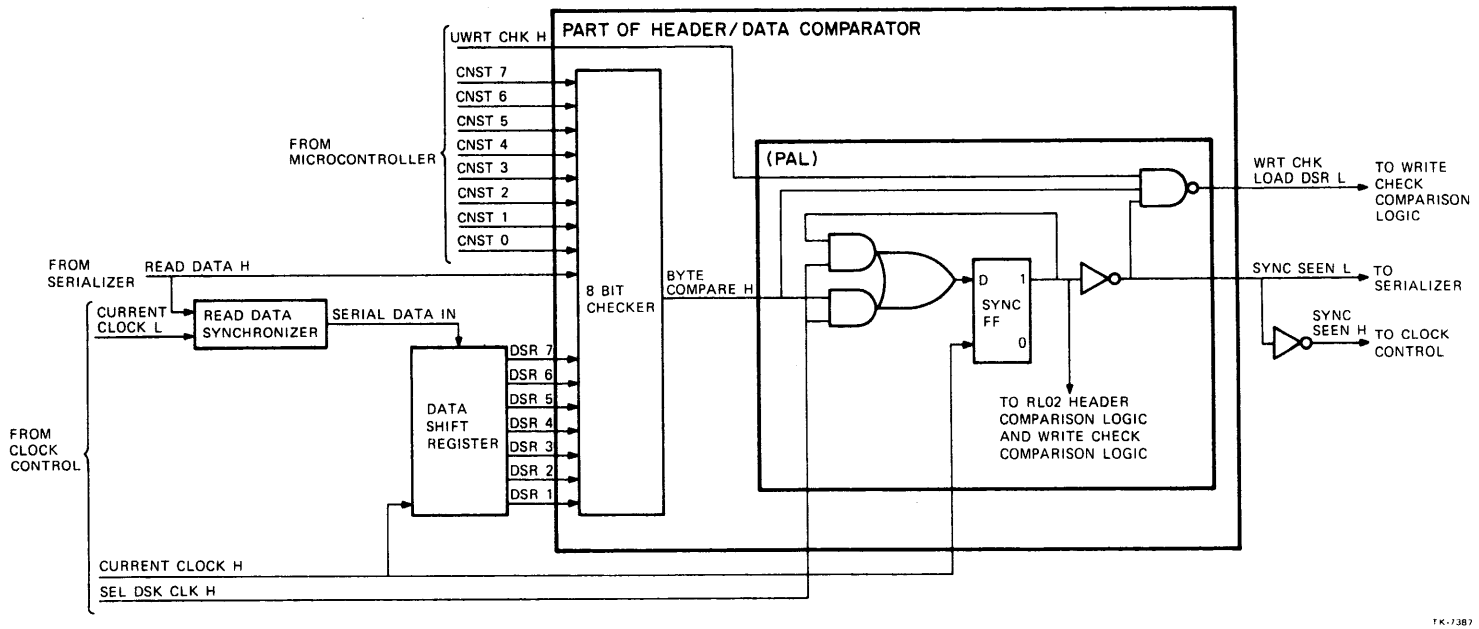


Figure 3-6 Sync Byte Recognition Logic Functional Block Diagram

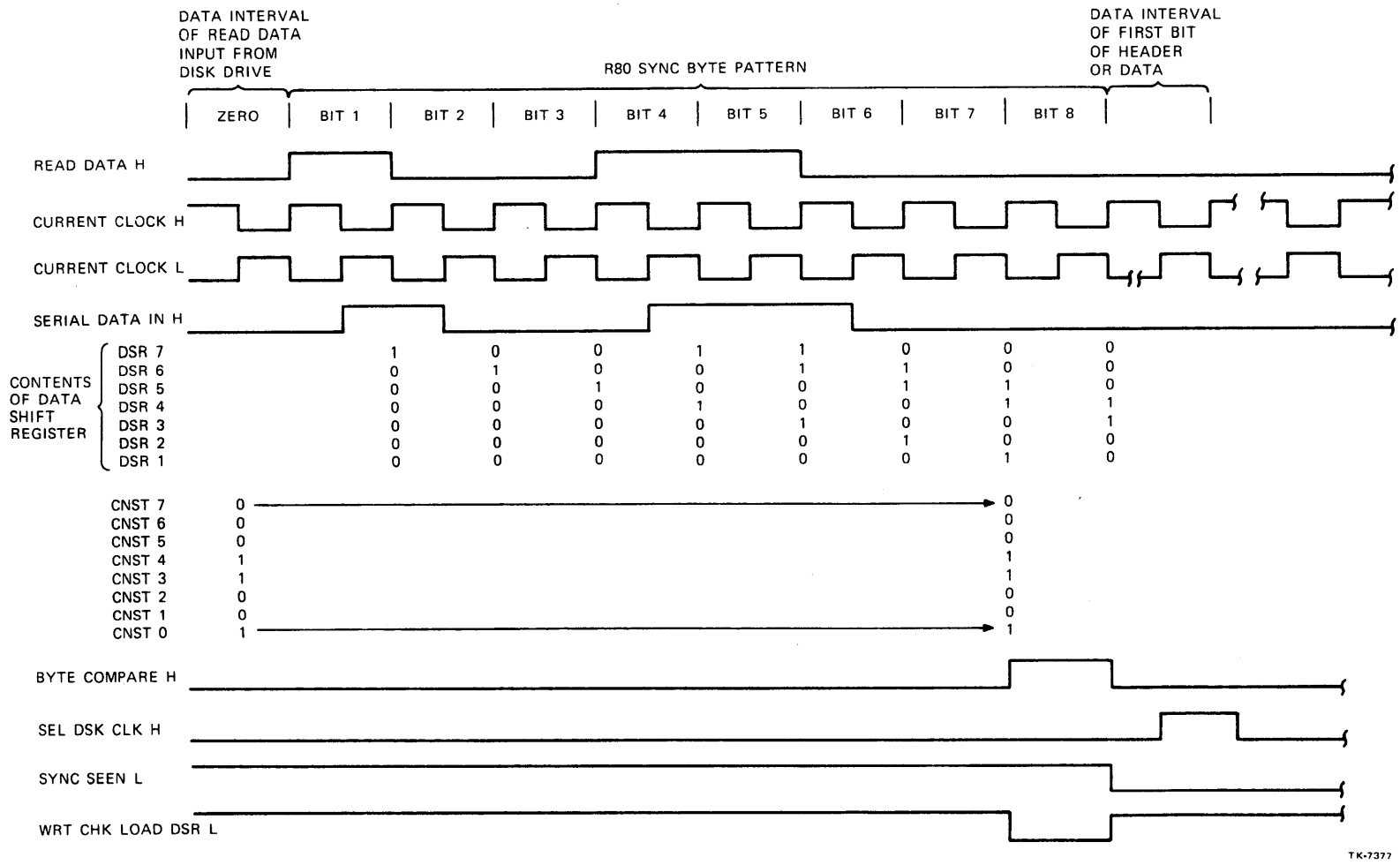


Figure 3-7 Sync Byte Recognition Logic Timing Diagram

When the read data synchronizer is loaded, it asserts the sampled condition of the READ DATA H input to the data shift register via the SERIAL DATA IN signal line. The CURRENT CLOCK H input to the data shift register loads the SERIAL DATA IN signal asserted into DSR7 of the data shift register and shifts the current contents of DSR 7:1 to DSR 6:0, respectively.

The parallel outputs DSR7:1 of the data shift register are asserted to the eight-bit checker where they are compared with the CONSTANTS (CNST 6:0, respectively). When a match is determined and the READ DATA H input matches the CNST 7 input, the BYTE COMPARE H output of the eight-bit checker is asserted to the input gates of the SYNC FF and to the write check load DSR gate. The BYTE COMPARE H and SEL DSK CLK H inputs to the SYNC FF input gates enable the SYNC FF to be set with the next positive transition of the CURRENT CLOCK H input, producing the SYNC SEEN H and L output signals. The BYTE COMPARE H signal enables the WRT CHK LOAD DSR L output. The WRT CHK LOAD DSR L output is asserted to the data shift register, where it is combined with the CURRENT CLOCK H input to load the data shift register.

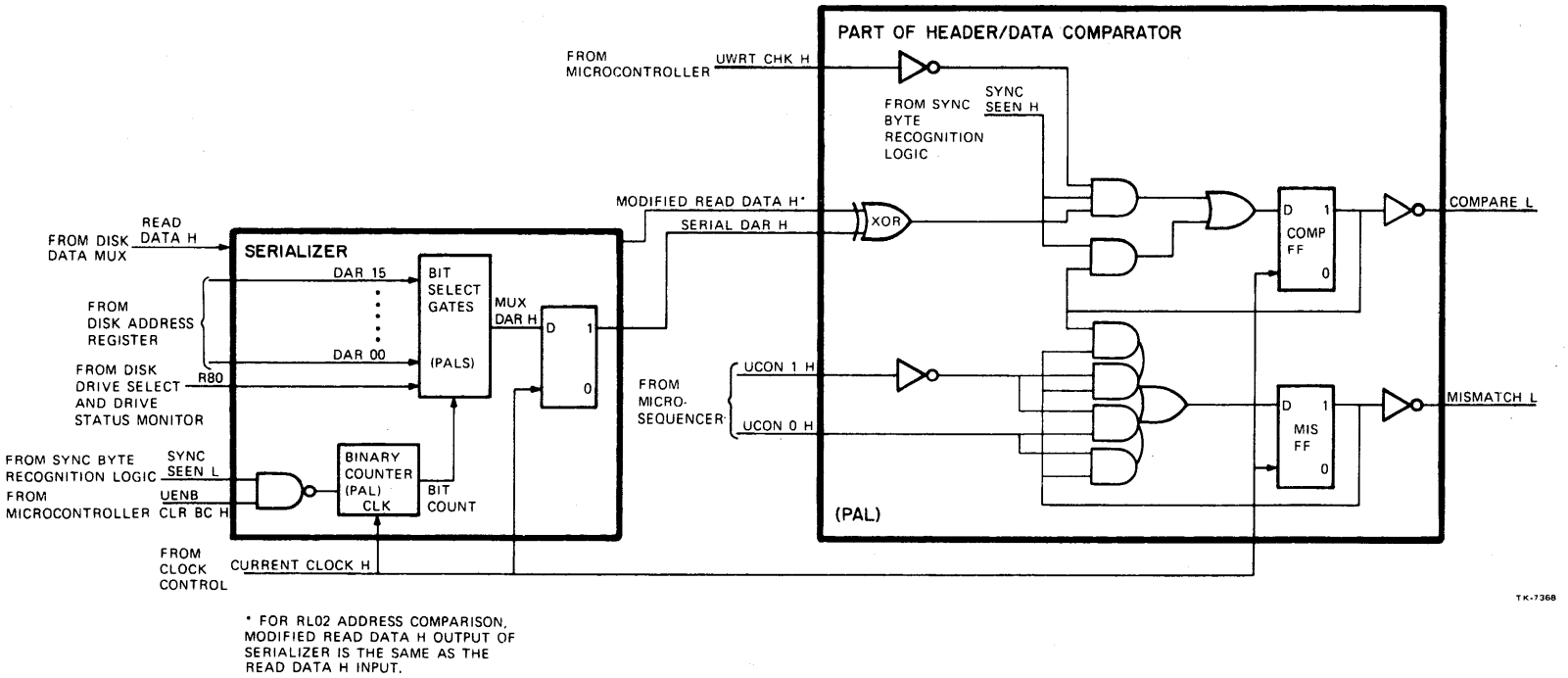
The SYNC SEEN H output of the SYNC FF is asserted to the input gates of the SYNC FF, where it is combined with the SEL DSK CLK H input from the clock control to inhibit the SYNC SEEN output from being reset until another clock is selected by the microcontroller. The SYNC SEEN L output of the SYNC FF is asserted to the serializer.

3.5.5 RL02 Header Comparison Logic

The RLO2 header comparison logic enables the IDC to locate the sector to or from which the data are to be written or read. The address to or from which the data are to be written or read is loaded into the IDC disk address register by the CPU. The parallel output of the disk address register is asserted to the serializer where it is converted into a serial format and asserted to the header/data comparator for comparison with the READ DATA input (the address portion of the RLO2 header data). A functional block diagram of the RL02 header comparison logic is shown in Figure 3-8.

The microcontroller disk drive select and drive status monitor and the sync byte recognition logic preset the conditions for performing the RL02 header comparison. The disk drive select and drive status monitor asserts a low R80 input to the bit select gates of the serializer to identify the selected drive as an RL02. (The R80 input identifies the order in which the parallel output of the disk address register is serialized). With the R80 input low, the bit select gates enable the DAR 00:15 inputs to be asserted sequentially to the SERIAL DAR H output (DAR 00 is asserted first and DAR 15 is asserted last).

When the search for the sync byte preceding the header data is initiated by the microcontroller, the microcontroller asserts a UENB CLR BC H input to the serializer binary counter. The UENB CLR BC H input and the SYNC SEEN L input (a high until the sync byte is found) hold the binary counter reset to the count of zero. While the binary counter is reset, the bit select gates continuously assert the DAR 00 output on the MUX DAR H output. (This enables the SERIAL DAR H output to provide the first bit of the header data to the header data comparator coincident with the first bit of the header data asserted via the READ DATA H input.) Before the sync byte is located, the SYNC SEEN H input to the header/data comparator is low. The low SYNC SEEN H input holds the compare flip-flop set until the SYNC SEEN H signal is asserted high. The UWRT CHK H input to the header/data comparator is a low when not performing a write check function, allowing the READ DATA H input to be compared with the SERIAL DAR H input once the sync byte has been located.



* FOR RL02 ADDRESS COMPARISON, MODIFIED READ DATA H OUTPUT OF SERIALIZER IS THE SAME AS THE READ DATA H INPUT.

Figure 3-8 RL02 Header Comparison Logic Functional Block Diagram

When the sync byte has been located, the SYNC SEEN H and SYNC SEEN L signals to the compare flip-flop enable gates and the reset gate of the serializer binary counter are asserted. The SYNC SEEH H signal enables the comparison of the first bit of the header data asserted via the READ DATA H input and the first bit of the read/write address from the disk address register (DAR 00) asserted via the SERIAL DAR H input to the header/data comparator. The SYNC SEEN L signal input to the serializer binary counter presets the binary counter to the bit count of 1. When the binary counter is preset by the SYNC SEEN L input, DAR 01 from the bit select gates is asserted on the MUX DAR H signal line.

With the first positive transition of the CURRENT CLOCK H input (following the assertion of the SYNC SEEN H and L signals), the results of the comparison of the first bit of the header data and the first bit of the desired read/write address are sampled at the compare flip-flop. Also, the first positive transition of the CURRENT CLOCK H input increments the serializer binary counter and enables the second bit of the read/write address to be loaded into the serializer flip-flop and asserted to the header/data comparator via the SERIAL DAR H input. (The binary counter is always one bit count ahead of the header address bit being compared. This enables each of the read/write address bits to be asserted to the header/data comparator in sync with the corresponding address bits asserted via the READ DATA H input.)

The SERIAL DAR H and READ DATA H inputs to the header/data comparator are compared via an exclusive OR in the enable gates of the compare flip-flop. If the SERIAL DATA H and READ DATA H inputs match, the output of the exclusive OR remains low and the compare flip-flop remains reset. However, if the inputs do not match, the compare flip-flop is set and a feedback loop from the compare flip-flop holds the compare flip-flop set for the remainder of the bit comparisons.

After all 16 bits of header have been compared, the microcontroller tests the comparison results by setting UCON 1 H at a high and UCON 0 at a low. This configuration of UCON signals inhibits three of the four enable gates at the input of the mismatch flip-flop. If during the bit comparisons all bits of the header data and the read/write address compared, the COMPARE H input to the fourth enable gate will be a low. Thus, when the other three enable gates are inhibited, the MISMATCH L output of the header/data comparator will be asserted high. However, if the bits did not compare, the COMPARE H input to the fourth enable gate holds the MISMATCH L output at a low when tested with the UCON signal inputs. The MISMATCH L signal output is asserted to the microcontroller. A timing diagram showing the relationship of signal inputs and events for the RL02 header comparison logic is shown in Figure 3-9.

3.5.6 R80 Header Comparison and Skip Sector Monitor Logic

The R80 header comparison and skip sector monitor logic enables the IDC to locate the sector to or from which the data are to be written or read and to determine if the sector is bad or displaced. The address to or from which the data are to be written or read is loaded into the IDC disk address register by the CPU. The parallel output of the disk address register is asserted to the serializer where it is converted into a serial format and asserted to the header/data comparator for comparison with the header portion of the READ DATA input (the R80 header data).

The R80 header data is not asserted via the READ DATA H input in the same bit configuration as the read/write data address contained in the disk address register. Also, the R80 header data contains unused bits, and various flag bits that are not significant to the R80 header comparison function. Therefore, for the R80 header comparison and skip sector monitor operation, the serializer is used to mask the unused bits and various flag bits of the R80 header data, to control assertion of the read/write address from the disk address register, and, if enabled, to record the status of the skip sector flag of the R80 header data. A functional block diagram of the R80 header comparison and skip sector monitor logic is shown in Figure 3-10.

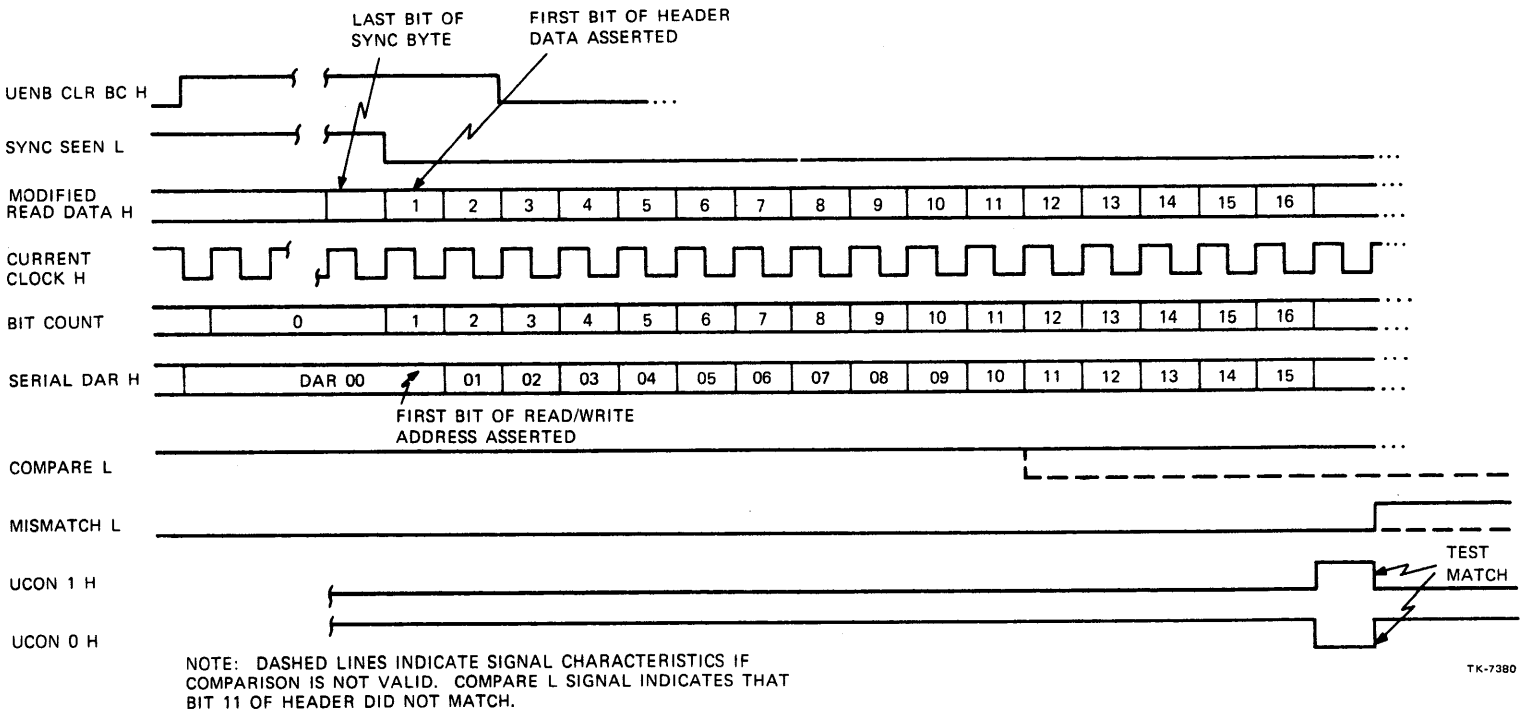


Figure 3-9 RL02 Header Comparison Logic Timing Diagram

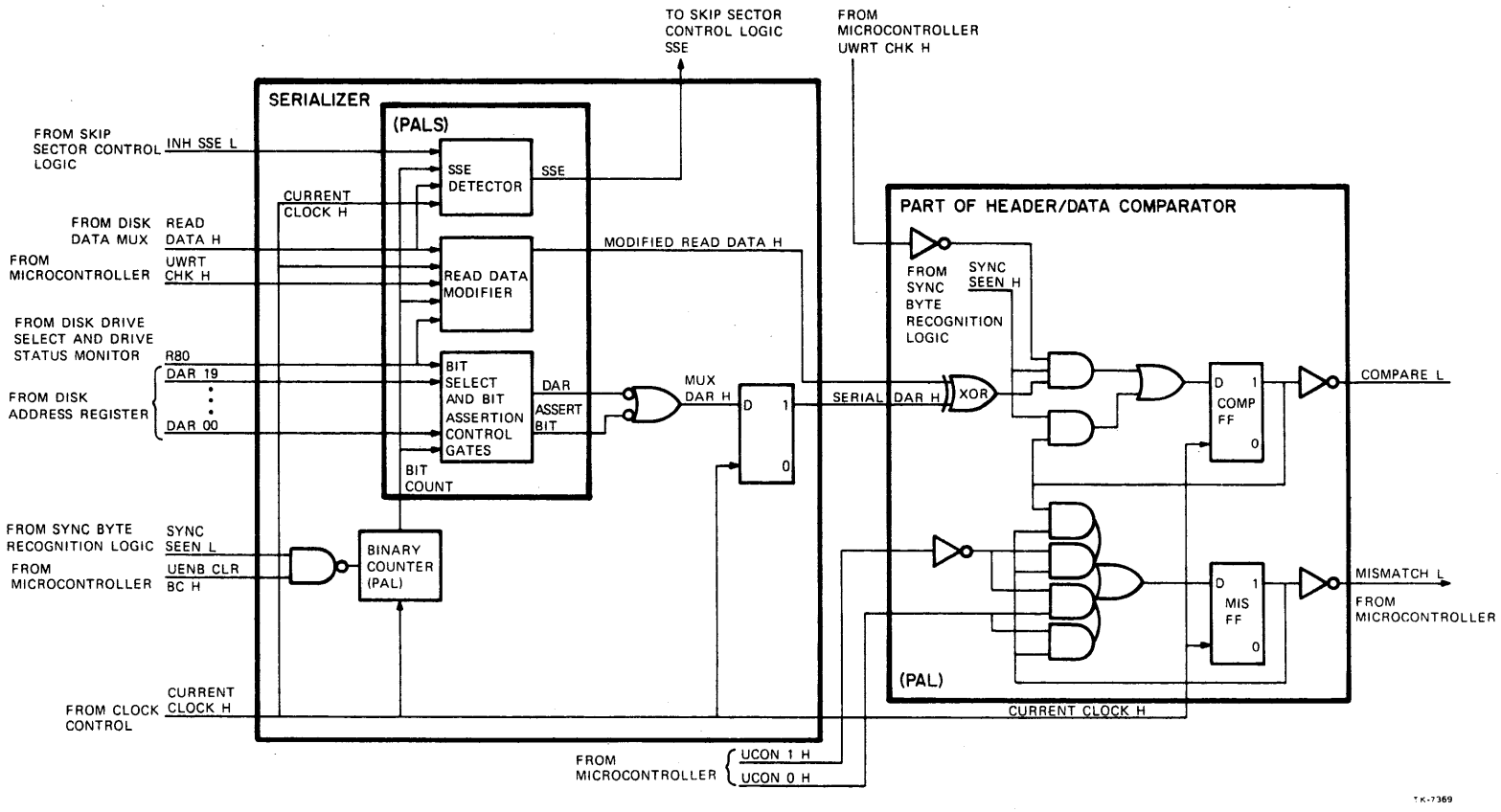


Figure 3-10 R80 Header Comparison and Skip Sector Monitor Logic Functional Block Diagram

3.5.6.1 R80 Header Comparison Logic – The microcontroller disk drive select and drive status monitor and the sync byte recognition logic preset the conditions for performing the R80 header comparison. The R80 input to the bit select and bit assertion control gates identifies the order in which the parallel output of the disk address register is to be serialized and enables the ASSERT BIT output with specific bit counts to mask the header data bits within the R80 header data that are not significant in locating the desired address. The R80 input is asserted also to the read data modifier.

With the R80 input asserted and the UWRT CHK H signal not asserted, the read data modifier masks out the header data bits that are of no significance in locating the desired address, by asserting the MODIFIED READ DATA H output to a high during the bit count interval in which these header data bits occur.

When the search for the sync byte preceding the header data is initiated, the microcontroller asserts a UENB CLR BC H input to the serializer binary counter. The UENB CLR BC H input and the SYNC SEEN L Input (a high until the sync byte is found) hold the binary counter reset to the count of zero. While the binary counter is reset, the bit select and bit assertion control gates continuously assert the DAR 09 output on the MUX DAR H and the SERIAL DAR H outputs. (This enables the SERIAL DAR H output to provide the first bit of the header data to the header data comparator coincident with the first bit of the R80 header data asserted via the MODIFIED READ DATA H input.)

Before the sync byte is located, the SYNC SEEN H input to the header/data comparator is low. The low SYNC SEEN H input holds the compare flip-flop set until the SYNC SEEN H signal is asserted. The UWRT CHK H input to the header/data comparator is an L when not performing a write check function, thus allowing the MODIFIED READ DATA H input to be compared with the SERIAL DAR H input once the sync byte has been located.

When the sync byte has been located, the SYNC SEEN H and SYNC SEEN L signals are asserted to the compare flip-flop enable gates and the reset gate of the serializer binary counter. The SYNC SEEN H signal enables the comparison of the first bit of the header data asserted via the MODIFIED READ DATA H input with the first bit of the read/write address from the disk address register to be asserted via the SERIAL DAR H input to the header/data comparator.
The SYNC SEEN L signal input to the serializer binary counter presets the binary counter to the bit count of 1. When the binary counter is preset by the SYNC SEEN L input, DAR 10 from the bit select and bit assertion control gates is asserted on the MUX DAR H signal line. With the first positive transition of the CURRENT CLOCK H input (the first positive transition following the assertion of the SYNC SEEN H and L signals), the results of the comparison of the first bit of the R80 header data (modified READ DATA) and the first bit of the desired read/write address (DAR 09) is sampled at the compare flip-flop. Also, the first positive transition of the CURRENT CLOCK H input increments the serializer binary counter and enables the second bit of the read/write address (DAR 10) to be loaded into the serializer flip-flop and asserted to the header/data comparator via the SERIAL DAR H input.

The sequence in which the read/write address bits (DAR 19:00) are enabled on the SERIAL DAR H input to the header/data comparator is controlled by the bit select and bit assertion control gates. Also, the bit select and bit assertion control gates enable a high to be asserted on the SERIAL DAR H signal line coincident with the BIT COUNT associated with the unused and various flag bits of the R80 header data. During the data interval (BIT COUNT) in which the unused and various flag bits of the R80 header data are being asserted on the READ DATA H signal line, the read data modifier forces the MODIFIED READ DATA H signal to a high. This allows the unused and various flag bits of the R80 header data to be masked from the comparison of the address information.

The timing diagram (Figure 3-11) shows the format of the R80 header data asserted via the READ DATA H input, the corresponding BIT COUNT output of the binary counter, the intervals (BIT COUNT) during which each DAR BIT is gated, the intervals during which the ASSERT BIT is enabled (L), and the intervals in which the READ DATA H input is modified to produce the MODIFIED READ DATA H output. Figure 3-11 also shows the resulting MODIFIED READ DATA H and SERIAL DAR H outputs.

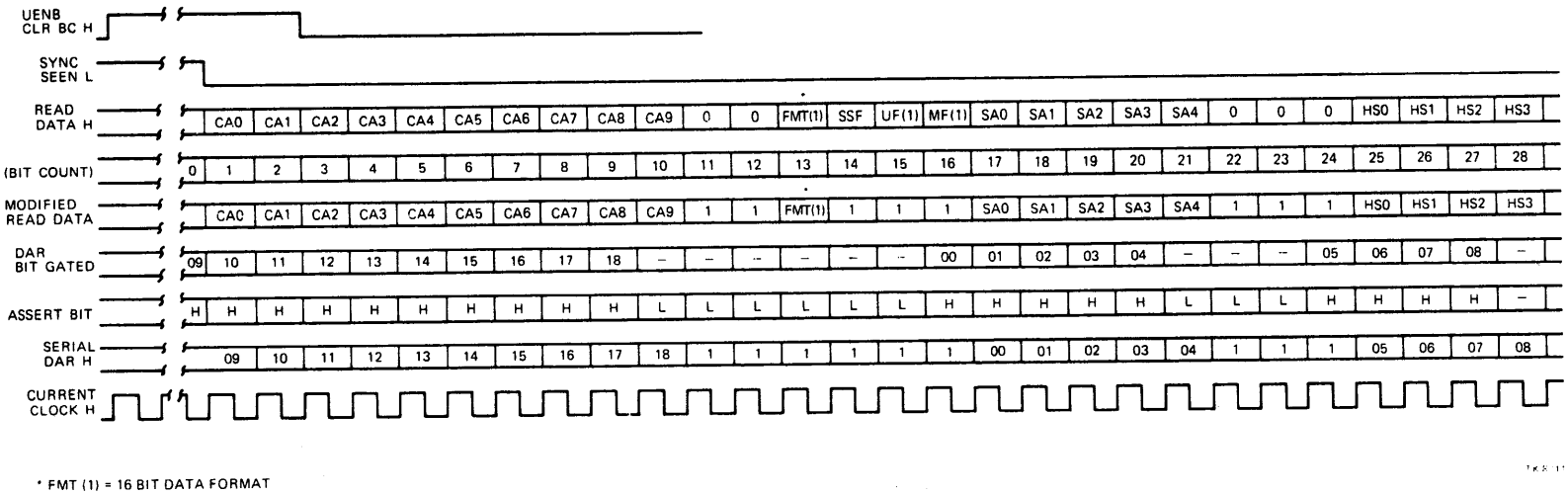


Figure 3-11 R80 Header Data Modification and Comparison Data Control Timing

The SERIAL DAR H and MODIFIED READ DATA H outputs of the serializer are compared in an exclusive OR in the header/data comparator. If the SERIAL DATA H and MODIFIED READ DATA H inputs match, the output of the exclusive OR remains low and the compare flip-flop remains reset. However, if the inputs do not match, the compare flip-flop is set and a feedback loop from the compare flip-flop holds the compare flip-flop set for the remainder of the bit comparisons.

After all bits of R80 header have been compared, the microcontroller tests the comparison results by setting the UCON 1 H at a high and UCON 0 at a low. This configuration of UCON signals inhibits three of the four enable gates at the input of the mismatch flip-flop. If during the bit comparisons, all bits of the header data and the read/write address compared, the COMPARE H input to the fourth enable gate will be a low. Thus, when the other three enable gates are inhibited, the MISMATCH L output of the header/data comparator will be asserted high. However, if the bits did not compare, the COMPARE H input to the fourth enable gate holds the MISMATCH L output at a low when tested with the UCON signal inputs. The MISMATCH L signal output is asserted to the microcontroller.

3.5.6.2 Skip Sector Monitor Logic – The SSE detector of the serializer is enabled during BIT COUNT 14 if the INH SSE L signal is not asserted L. When the SSE detector is enabled, the state of the READ DATA H input is loaded into the SSE detector at the end of BIT COUNT 14 by the CURRENT CLOCK H input. If the READ DATA H signal line were high during the interval, indicating that the sector is a bad or displaced sector, a high SSE signal is asserted to the skip sector control logic.

3.5.7 Skip Sector Control Logic

The skip sector control logic (Figure 3-12) enables the IDC to skip a bad or defective sector when writing or reading from the R80.

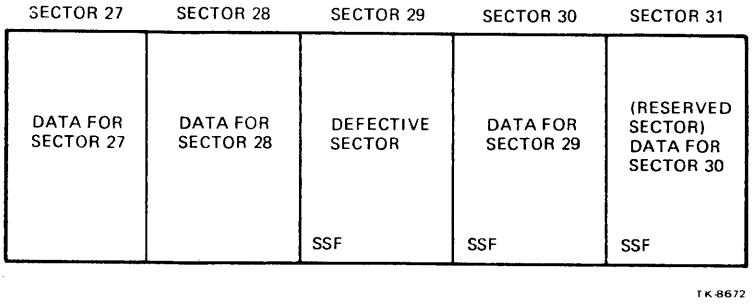


Figure 3-12 Skip Sector Control Logic
Functional Block Diagram

When the skip sector flag (bit 13 of the header word) is detected during a write operation, that sector is skipped. The information is then written in the next sector. Each following sector is displaced by one.

Figure 3-13 shows an example of the last five sectors of a track. In this example, sector 29 was found to be defective during the formatting process. The skip sector flag was set in sectors 29 and 30. Notice that the data for sector 29 were written in sector 30. The data for sector 30 are written in the reserve sector 31. The skip sector flag is set in all remaining sectors of the track. This is done in case a data transfer begins at a sector that is beyond the defective sector.

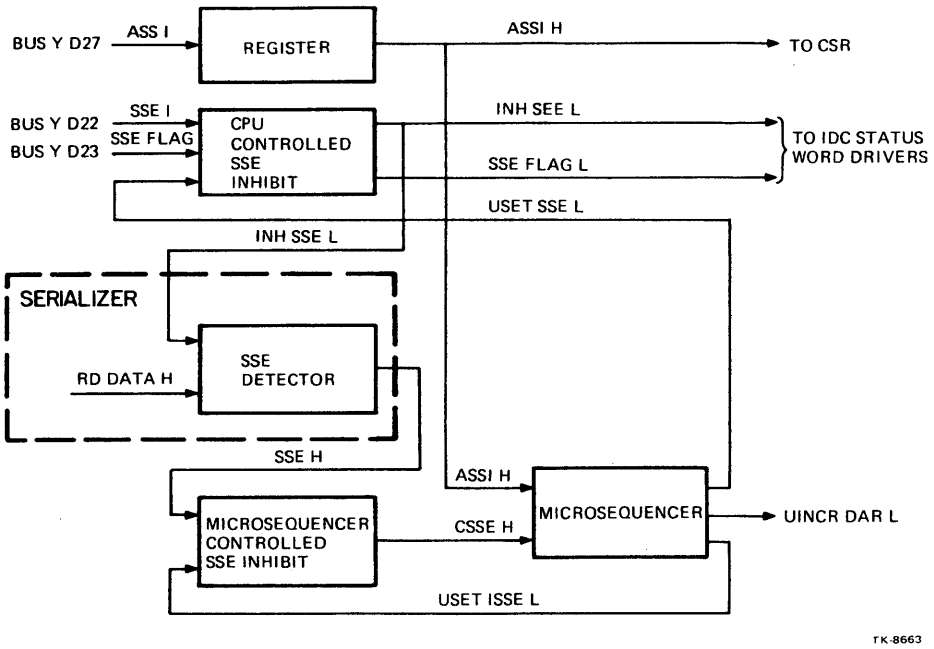


Figure 3-13 Skip Sector Example

During a read operation, the same type of process takes place. When a skip sector is detected as being set, the data are then read from the next sector.

When a skip sector error is detected by the skip sector monitor logic (Figure 3-10), it asserts skip sector error signal SSE H which is sourced to the skip sector control logic. Inside the skip sector control logic the skip sector error signal SSE H is ANDed with the microsequencer inhibit skip section error (USET ISSE L). Provided that the microsequencer has enabled skip sector errors (USET ISSE L deasserted), skip sector error signal SSE H is sourced to the microsequencer as CSSE H.

Provided that the CPU has disabled automatic skip sectoring (CSR bit 27 (ASSI) set), the microsequencer aborts the operation immediately. This results in the assertion of USET SSE L to flag the CPU (SSE FLAG L) that operation has been terminated due to a skip sector error and in the assertion of UINCR DAR L to increment the disk address register.

The driver software then sets CSR bit 23 (SSE FLAG) to clear the skip sector error, sets CSR bit 22 (SSEI) to inhibit further generation of skip sector errors, and clears CSR 7 (CRDY) to set the GO bit which continues the transfer.

The SSEI bit allows the IDC to finish the transfer without an interrupt from the skip sector flag. It also sets up the IDC to read sector 31, if necessary. The SSEI bit is cleared by the IDC at the end of each track. Therefore, the driver software must clear SSEI at the beginning of each data transfer.

3.5.8 Write Check Data Comparison Logic

The write check data comparison logic (Figure 3-14) performs a bit-by-bit comparison of the data portion of the R80/RL READ DATA input with the DSRO input to determine if the R80/RL READ DATA matches the serialized data from the data shift register.

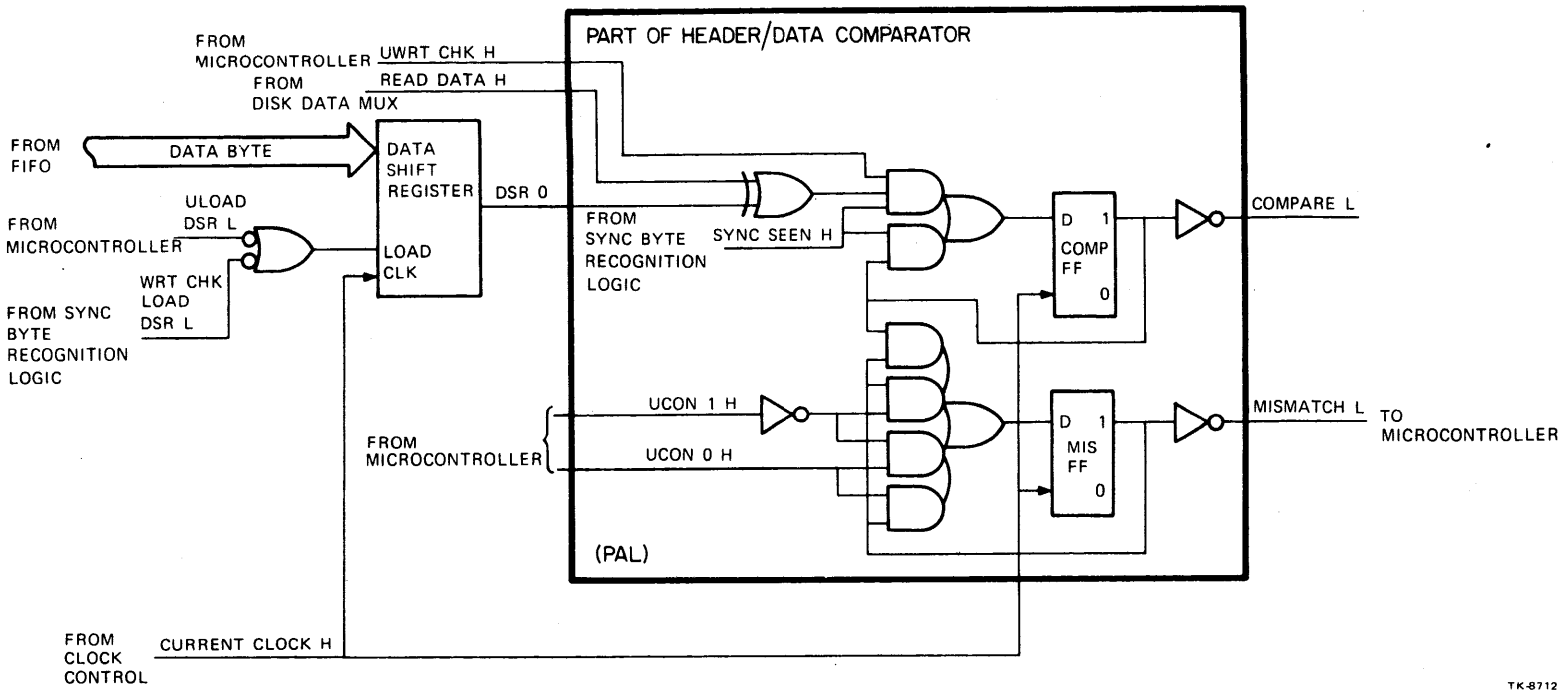


Figure 3-14 Write Check Data Comparison Logic Functional Block Diagram

In the write check mode, the WRT CHK LOAD DSRL output of the header/data comparator is enabled when the R80 READ DATA header gap/RL02 READ DATA data preamble sync byte is found. The WRT CHK LOAD DSRL signal is asserted to the data shift register where it enables the first data byte from the selected FIFO to be loaded into the data shift register. The microcontroller then increments the selected FIFO address counter.

When the first data byte is loaded into the data shift register, bit 0 of the first data byte is asserted to the header/data comparator via the DSRO output of the data shift register. The first bit of the first data byte is asserted to the header/data comparator coincident with the first bit of the data portion of the R80/RL READ DATA asserted from the disk drive (READ DATA H from the DISK DATA MUX). (Because the CURRENT CLOCK used by the data shift register is derived from the R80/RL READ DATA input, the data loaded into the data shift register is serialized and asserted to the header/data comparator in sync with each bit of the R80/RL READ DATA input.) The data shift register serializes and asserts bits 0 through 7 of the first data byte to the header/data comparator.

After bit 7 of the first data byte has been asserted to the header/data comparator, the microcontroller loads the second byte of data from the selected FIFO into the data shift register and increments the selected FIFO address counter.

After bit 7 of the second data byte has been serialized and asserted to the header/data comparator, the microcontroller loads the third data byte from the selected FIFO into the data shift register and increments the FIFO A address counter. This process is repeated until all 512/256 bytes of data from the selected FIFO have been serialized and asserted to the header/data comparator for comparison with the data portion of the R80/RL READ DATA input. (A detailed discussion of how the microcontroller causes serialization of data from the data buffers is provided in Paragraph 3.5.11.)

When the sync byte has been located, the SYNC SEEN H signal to the compare flip-flop enables gates is asserted. The SYNC SEEN H signal enables the comparison of the first bit of R80/RL READ DATA asserted via the READ DATA H input and the first bit of data of the serialized data from the data shift register asserted via the DSRO input to the header/data comparator.

With the first positive transition of the CURRENT CLOCK H input (the first positive transition following the assertion of SYNC SEEN H), the results of the comparison of the first bit of the R80/RL READ DATA and the first bit of the serialized data from the data shift register are sampled at the compare flip-flop.

The DSRO and READ DATA H inputs to the header/data comparator are compared via an exclusive OR in the enable gates of the compare flip-flop. If the DSRO and READ DATA H inputs match, the output of the exclusive OR remains low and the compare flip-flop remains reset. However, if the inputs do not match, the compare flip-flop is set and a feedback loop from the compare flip-flop holds the compare flip-flop set for the remainder of the bit comparisons.

After all bits have been compared, the microcontroller tests the comparison results by setting UCON 1 H at a high and UCON 0 at a low. This configuration of UCON signals inhibits three of the four enable gates at the input of the mismatch flip-flop. If during the bit comparisons, all bits of the data shift register and the R80/RL READ DATA compared, the COMPARE H input to the fourth enable gate will be a low. Thus, when the other three enable gates are inhibited, the MISMATCH L output of the header/data comparator will be asserted high. However, if the bits did not compare, the COMPARE H input to the fourth enable gate holds the MISMATCH L output at a low when tested with the UCON signal inputs. The MISMATCH L signal output is asserted to the microcontroller. A timing diagram showing the relationship of signal inputs and events for the write check data comparison logic is shown in Figure 3-15.

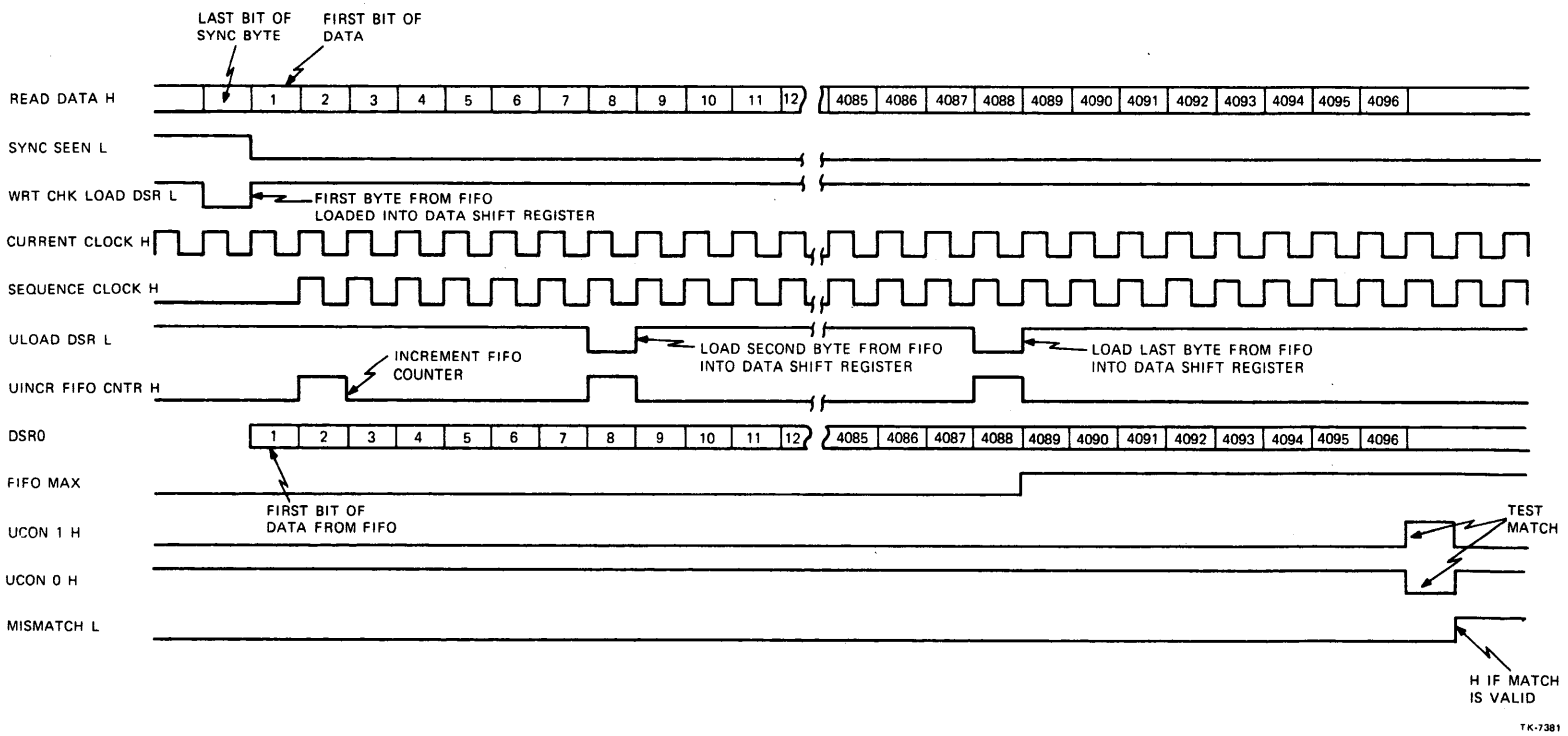


Figure 3-15 Write Check Data Comparison Logic Timing Diagram

3.5.9 Interrupt Control Logic

The IDC generates two interrupts: UBUS BR5 and PORT XFER REQ. The UBUS BR5 interrupt is generated, when enabled by the CPU, after the function requested by the CPU has been performed or in the idle mode of operation if a drive status change is detected. The PORT XFER REQ is a special interrupt signal that is used during the read, write, or write check functions. The PORT XFER REQ signals the CPU that IDC has read, written, or write checked one complete sector of data and is ready to read, write, or write check the next sector of data.

3.5.9.1 UBUS BR5 – The UBUS BR5 interrupt control logic is enabled by the CPU by setting the Interrupt Enable (IE bit) of the IDC control word input. When the IDC control word is loaded into the CSR, the IE bit is asserted to the UBUS BR5 interrupt control logic (see Figure 3-16). If the IE bit of the IDC control word is not set (the IE signal input is low) the generation of UBUS BR5 is inhibited. If the IE bit is set, the UBUS BR5 interrupt may be generated by the microcontroller by asserting a USET INT L signal, or by the IDC status logic by deasserting any one of the attention bits (ATTN3:0).

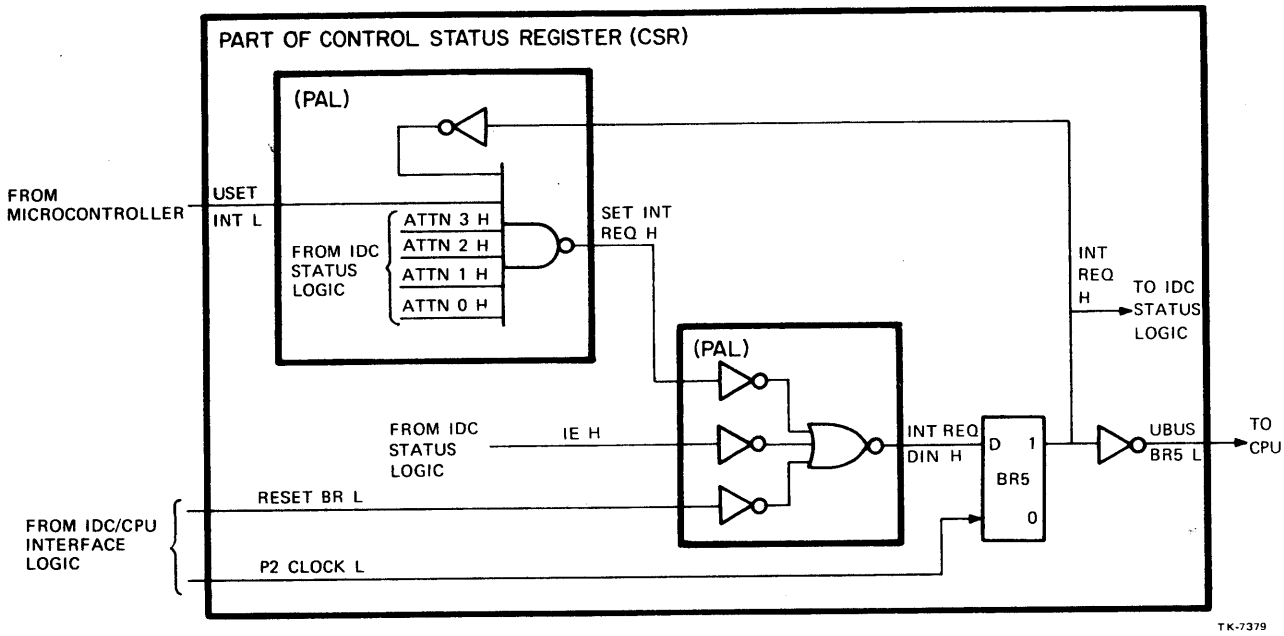


Figure 3-16 UBUS BR5 Interrupt Control Logic Functional Block Diagram

When the USET INT L signal from the microcontroller is asserted or one of the attention bits from the IDC status logic is deasserted, the SET INT REQ signal goes low producing a high INT REQ DIN signal input to the BR5 flip-flop. The high INT REQ DIN signal causes the BR5 flip-flop to be set with the next P2 CLOCK L input, producing an INT REQ H output. The INT REQ H output of the BR5 flip-flop is inverted to produce the UBUS BR5 L interrupt signal. The INT REQ H output of the BR5 flip-flop is also inverted and asserted to the control gates of the BR5 flip-flop to hold the UBUS BR5 L interrupt signal asserted until it is reset by the CPU.

The CPU resets the flip-flop by asserting a RESET BR port microinstruction to the IDC/CPU interface logic. The resulting RESET BR L signal causes the INT REQ DIN H input to the BR5 flip-flop to be deasserted, which enables the BR5 flip-flop to be reset with the P2 CLOCK L input. When the BR5 flip-flop is reset, the UBUS BR5 L signal output to the CPU is set H.

3.5.9.2 PORT XFER REQ – The PORT XFER REQ L interrupt signal output of the IDC is set by inputs from the microcontroller (see Figure 3-17). The microcontroller initiates the PORT XFER REQ L signal output by asserting UCMD2(H), UCMD1(L) and UCMD0(L). The UCMD inputs enable the PORT XFER REQ flip-flop to be reset. When the PORT XFER REQ flip-flop is reset, the low signal output is coupled back to the input gates to hold the flip-flop in the reset state. The low signal output of the flip-flop is inverted to produce an XFER REQ H signal. The XFER REQ H signal is inverted and asserted to the CPU as the PORT XFER REQ L (interrupt) signal. The PORT XFER REQ L signal is reset when the CPU asserts an XFER GRANT L signal or when the IDC is initialized (INIT L asserted).

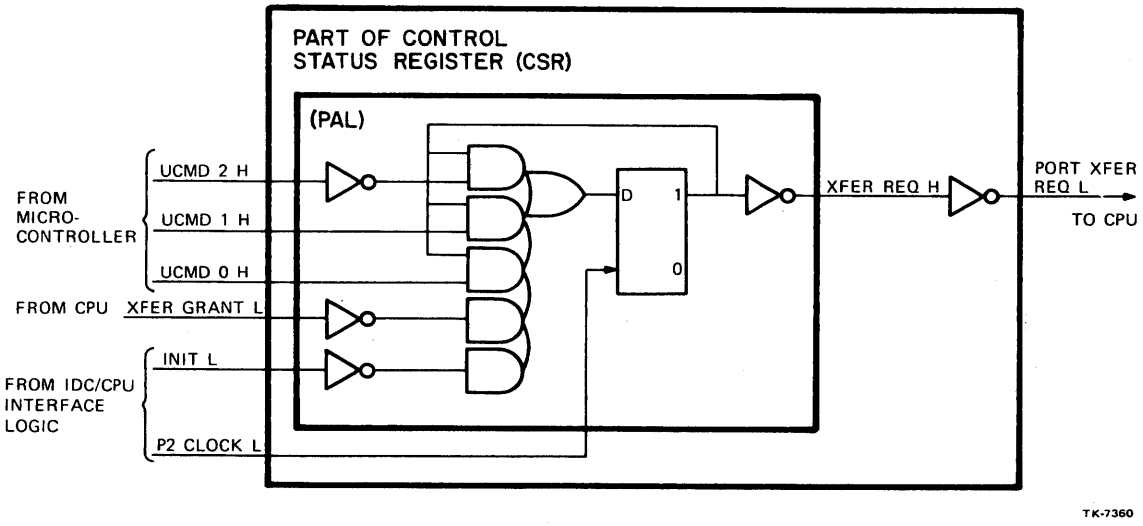


Figure 3-17 PORT XFER REQ Logic Functional Block Diagram

3.5.10 IDC Control Register, Timeout Logic, and Status Logic – The IDC control register, timeout logic, and status logic is contained in the control status register (CSR) shown in Figure 3-1.

3.5.10.1 IDC Control Register – The IDC control register portion of the IDC control register, timeout logic, and status logic registers the IDC control word input from the CPU. The registered IDC control word inputs (See Figure 3-18) are used to provide the following:

- Branch condition inputs to the microcontroller
- Drive select information to the disk drive select and drive status monitor
- Skip sector data to the skip sector control logic
- Interrupt enable signal to the UBUS BR5 interrupt control logic
- Presetting the IDC data paths for maintenance
- Resetting the IDC status timeout, OPI, and DLT control registers

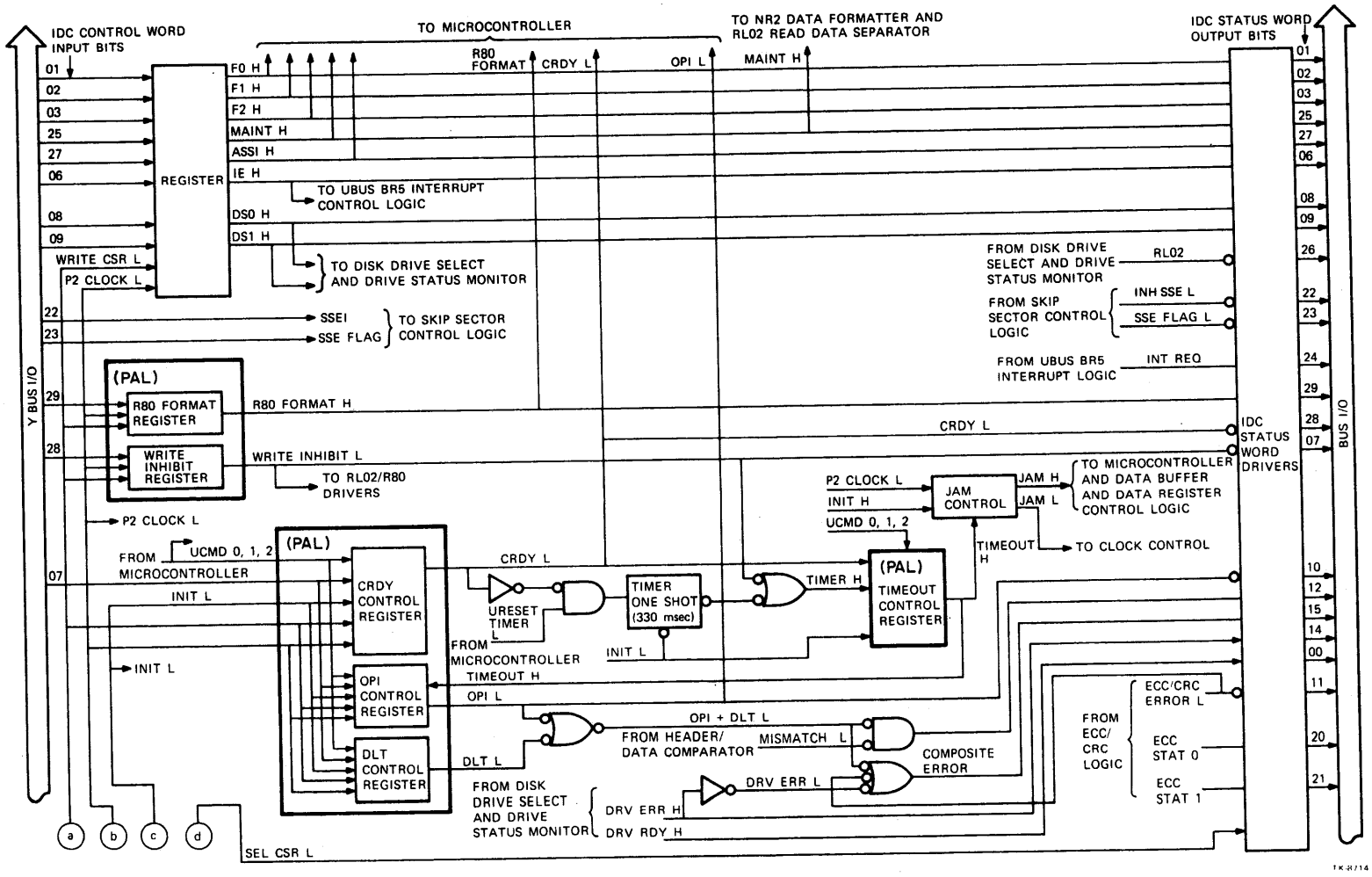


Figure 3-18 IDC Control Register Timeout Logic and Status Logic Functional Block Diagram (Sheet 1 of 3)

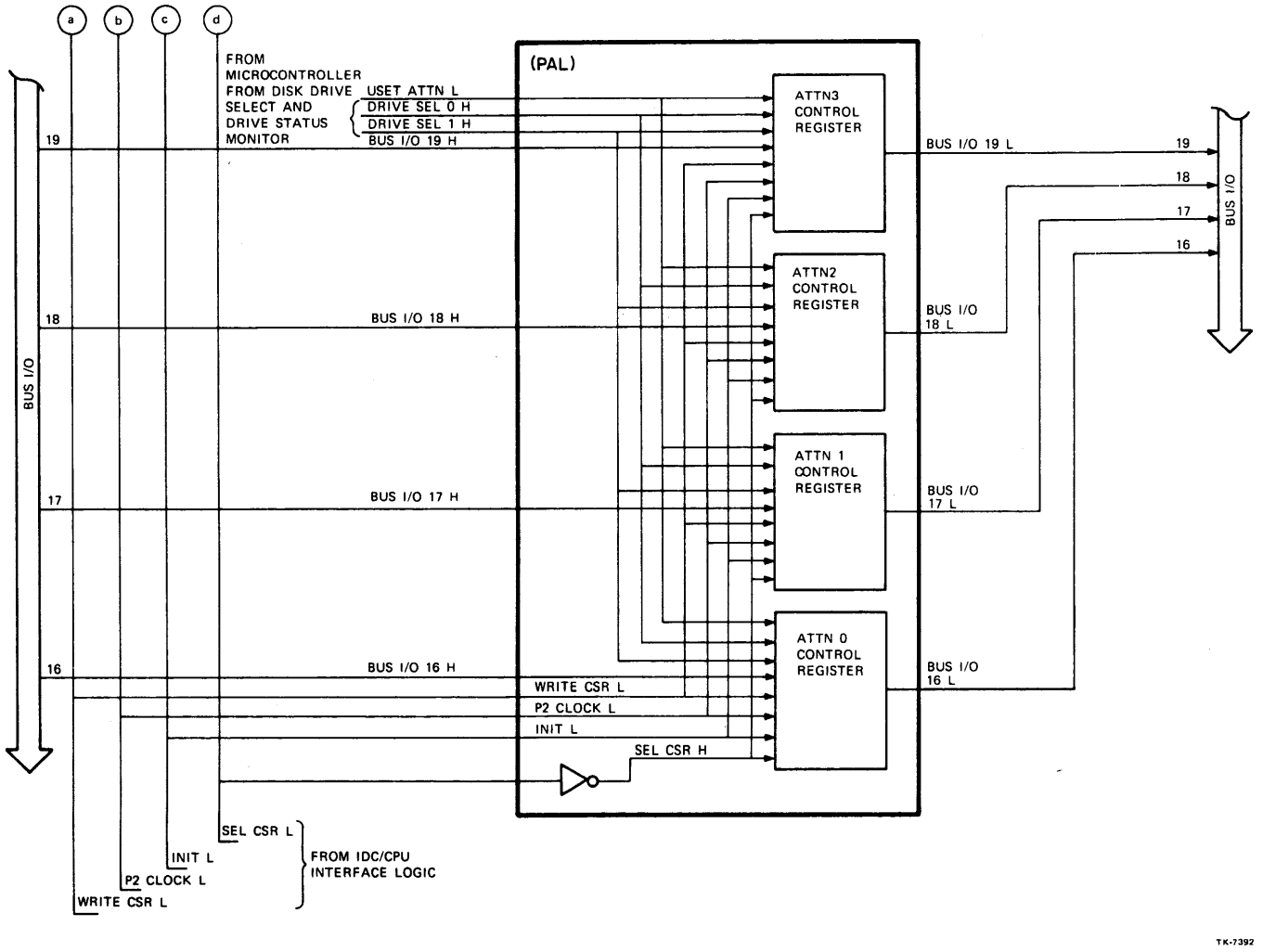


Figure 3-18 IDC Control Register Timeout Logic and Status Logic Functional Block Diagram (Sheet 2 of 3)

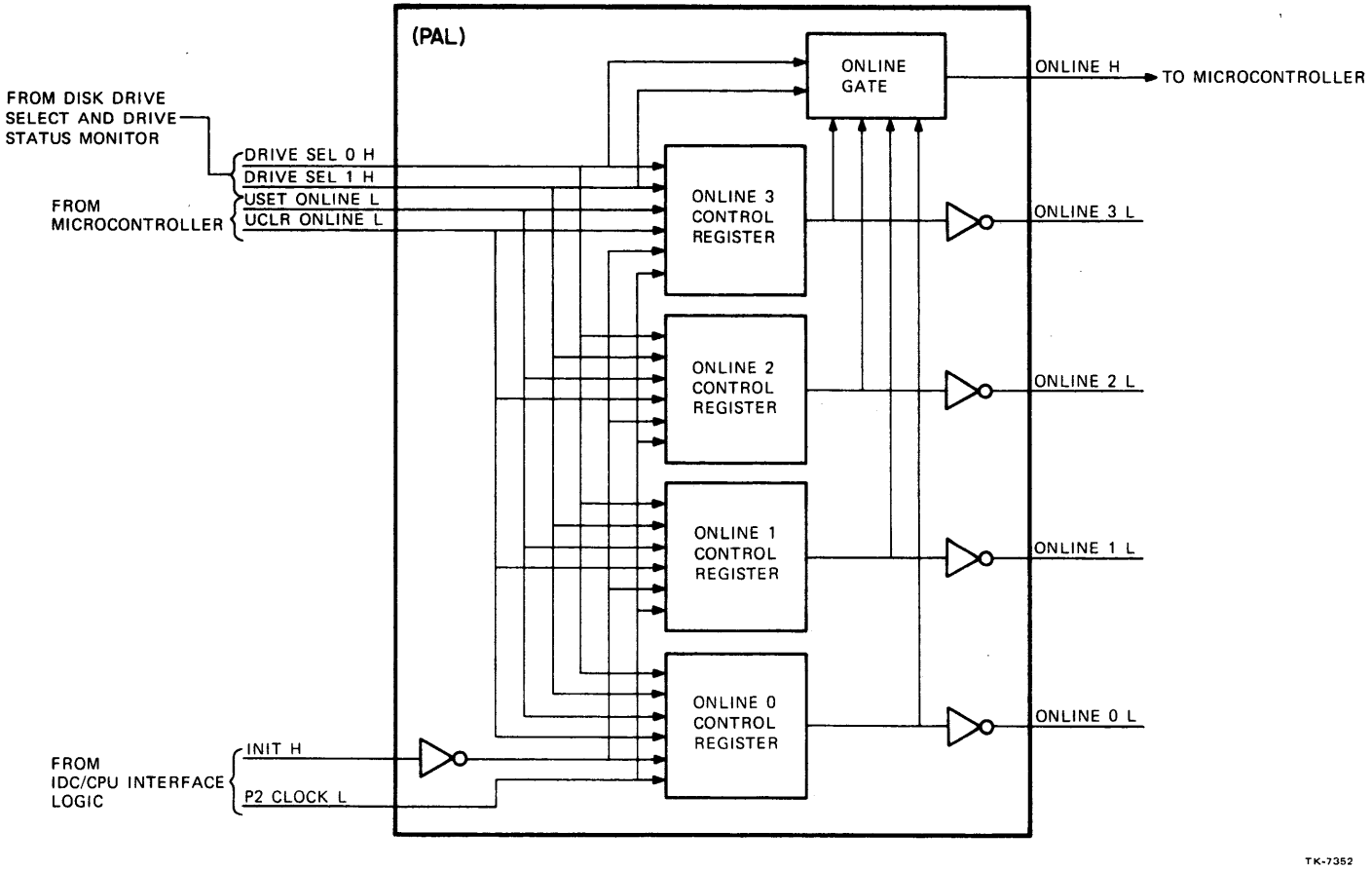


Figure 3-18 IDC Control Register Timeout Logic and Status Logic Functional Block Diagram (Sheet 3 of 3)

All registered IDC control word inputs are asserted to the IDC status word drivers and form part of the IDC status word output of the IDC.

There are four unregistered IDC control word inputs (BUS I/O 19:16). These bits are used to clear the attention (ATTN) control registers of the IDC status logic.

Each of the IDC control word is loaded into the IDC by the WRITE CSR L and P2 CLOCK L inputs from the IDC/CPU interface logic. Bits 01, 02, 03, 06, 08, 09, 25, 27, 28, and 29 are loaded directly into registers. These registered bits are asserted to the IDC status word drivers to provide (as part of the status word output) a record of the specified IDC control word input. These registered bits also provide branch condition inputs to the microcontroller (F0, F1, F2, MAINT, ASSI, R80 FORMAT), an enable bit (IE) to the UBUS BR5 interrupt control logic, disk drive select information (DS0 and DS1) to the disk drive select and drive status monitor, and, if a maintenance function is specified, a write inhibit signal to the timeout logic and to the R80/RL02 drivers. The write inhibit signal inhibits writing to the R80 or RL02 disk drives and inhibits timeout from occurring during a maintenance function.

Bits 22 and 23 are registered bits also. However, these bits are discussed as part of the skip sector control logic (refer to Paragraph 3.5.7).

Bits 16 through 19 of the IDC control word input are used to reset the attention control register associated with each disk drive. The attention control registers are discussed as part of the timeout and status logic.

Bit 07 of the IDC control word input is asserted to the CRDY, OPI, and DLT control registers. Bit 07 (the CRDY bit input) is registered in the CRDY control register. When registered, the CRDY output is asserted to the microcontroller where it enables the microcontroller to branch on the branch condition inputs and initiate the specified function. The CRDY output is asserted also to the timeout logic to start the timer. While bit 07 is being loaded into the CRDY control register, it is also being used to reset the OPI and DLT control registers of the status logic to clear any error information that may have been generated during the previously specified IDC function.

3.5.10.2 Timeout and Status Logic – The timeout and status logic of the control status register limits the time in which the IDC may attempt to perform a specified function (other than maintenance), registers IDC fault status (OPI and DLT), keeps track of the disk drives currently in use, and, if the IDC did not complete the specified function within the time constraints of the timeout logic, registers the reason for noncompletion. The status logic also formats and asserts to the IDC status word drivers the status information from the disk drive select and drive ready monitor, skip sector control logic, UBUS BR5 interrupt control logic, header/data comparator, and ECC/CRC logic.

The timeout of the control status register is enabled when the CRDY L output of the CRDY control register is set to a high (when an IDC control word is loaded). The low-to-high transition of the CRDY L signal triggers the timer oneshot (see Figure 3-18). The duration of the timer oneshot is set at 150 milliseconds, which allows sufficient time for the IDC to perform the function specified by the IDC control word input. After the 150 millisecond time limit, the timer oneshot output goes low producing a TIMER H signal input to the timeout control register. (If a maintenance function is specified, the WRITE INHIBIT L signal from the write inhibit register inhibits the generation of TIMER H). The TIMER H signal is asserted to the timeout control register where it is combined with the CRDY input. If the CRDY input has not been set to a L indicating that the specified function has not been completed, the timeout control register is set producing a high TIMEOUT H signal. The TIMEOUT H input to the OPI control register causes the OPI control register to be set, producing a low OPI output.

The TIMEOUT H signal is also asserted to the jam control and to the OPI control register. The TIMEOUT H input to the jam control initiates generation of the JAM H and JAM L outputs. The JAM H output is asserted to the microcontroller, to force the next address to 1FF and to the data buffer and data register control logic to inhibit reading and writing to the data buffers. The JAM L output is asserted to the clock control to deselect the clock selected and to select the CPU clock. When the microcontroller is set to 1FF, the UCMD 0,1, and 2 outputs from the microcontroller set the CRDY control register producing a low CRDY L output. Also, the microcontroller generates a USET INT L signal which causes a UBUS BR5 interrupt signal to be asserted to the CPU (Refer to Paragraph 3.5.9.1).

If the function specified by the IDC control word input is to be extended (for example, if the current read function is to be performed for reading more than one sector of data), the microcontroller retriggers the timer oneshot by asserting a URESET TIMER L pulse. Retriggering the timer oneshot inhibits the timeout from occurring as a result of extended operations by extending the timer cycle an additional 150 milliseconds.

The IDC fault status registers include the OPI control register and the DLT control register. The OPI and DLT control registers are reset when an IDC control word specifying a function to be performed is loaded (where the CRDY bit, bit 07, is low and WRT CSR L is asserted low) or the IDC is initialized. When the OPI and DLT control registers are reset, the OPI L and DLT L outputs are set high. The microcontroller causes setting of the OPI L and DLT L outputs. The OPI control register may be set also by the TIMEOUT H signal from the timeout control register as discussed previously.

The microcontroller causes setting of the the OPI and DLT control registers by asserting the proper UCMD 0, 1, and 2 codes. If the IDC does not locate the proper header before timeout occurs, the microcontroller asserts the UCMD code to set the DLT control register. If an ECC/CRC error is found in the disk header data, the microcontroller asserts the UCMD code to set the OPI control register. If during a write function, the requisite data needed has not been loaded, the microcontroller sets the DLT control register. The OPI and DLT L outputs are asserted to the status formatting logic where it is encoded to provide error information to the CPU via the IDC status word output. The format of the IDC status word output is presented in Figure 2-10.

The timeout and status logic of the control status register also keeps track of the disk drive status through the attention (ATTN) and on-line control registers (See Figure 3-16). One attention control register and one on-line control register is provided for each of the four disk drives that may be used with the IDC. The on-line registers record that when last monitored, the applicable disk drive was in use (performing a function) or not in use. The attention registers are used to signal the CPU that the associated drive is currently in use, has completed the function it had been performing, or is reporting an error.

During the idle mode of operation, the microcontroller samples disk drive status. When in the idle mode of operation, the microcontroller generates the UDRV SEL 0 and 1 and UDRV SEL signals used by the disk drive select and drive status monitor. The resulting disk drive address bits (DRIVE SEL 0 and 1) from the disk drive select and drive status monitor are asserted to each of the ATTN 3:0 control registers, the on-line 3:0 control registers, and the on-line gate. The disk drive select and drive status monitor gates the DRIVE RDY and DRIVE ERR signals from the appropriate disk drive to the microcontroller. The DRIVE SEL 0 and 1 inputs to the on-line gate enables the appropriate ONLINE signal to be asserted to the microcontroller.

The microcontroller, after asserting the UDRV SEL 0 and 1 and UDRV SEL outputs, branches on the DRIVE RDY, DRIVE ERR, and ONLINE inputs to control the on-line and attention control registers associated with the addressed disk drive.

If the selected disk drive is not reporting an error and DRIVE RDY is not present (the disk drive is performing a function), the microcontroller asserts a UCLEAR ONLINE L signal to the on-line control registers. The UCLEAR ONLINE signal clears the appropriate on-line register to provide a record that during the monitoring period, the disk drive was busy.

If the selected disk drive is not reporting an error, DRIVE RDY is present, and the appropriate ON-LINE control register is set (indicating that during the previous monitoring period the disk drive was not busy), the microcontroller enables the next sequential UDRV SEL 0 and 1 address and UDRV SEL signals.

If the selected disk drive is not reporting an error, DRIVE RDY is present, and the on-line control register is reset (indicating that during the previous monitoring period the disk drive was busy), the microcontroller asserts a USET ONLINE L signal to the on-line control registers and a USET ATTN L signal to the attention control registers. The USET ONLINE signal sets the appropriate on-line control register to record that during the monitoring period, the disk drive was not busy. The USET ATTN L signal sets the applicable attention control registers.

If the selected disk drive is reporting an error (DRIVE ERR is asserted), the microcontroller asserts a USET ATTN L signal to the attention control registers. Also, if the associated on-line control register is presently cleared (indicating that the disk drive had been busy performing a function during the previous monitoring period), the microcontroller asserts a USET ONLINE signal to the on-line control registers.

The USET ATTN L signal sets the applicable attention control register. The USET ONLINE signal sets the applicable on-line control registers to record that during the previous monitoring period the drive was not busy or was reporting an error.

As indicated in Figure 3-18, the on-line control registers may be cleared and set by the microcontroller or when the IDC is initialized. However, the attention control registers may be cleared only by the CPU through an IDC control word input or when the IDC is initialized.

3.5.11 Serializing Data from Data Buffer and Sync Byte Tristate Drivers – The sync byte tristate drivers, data buffers (FIFO A and FIFO B), and data shift register are used to serialize the sync byte and data to be written to the disk drive during a write data function. During a write check function the data buffers and data shift register are used to serialize the data from the selected FIFO such that it may be compared with the data portion of the READ DATA input from the disk drive.

When a write data or write check function is specified by the IDC control word input to the IDC, the microcontroller selects the FIFO to be used by asserting the appropriate USEL FIFO (A or B) signal to the data buffer and data register control logic (see Figure 3-19).

During the write data function, the inputs from the microcontroller cause the assertion of the DSR 0 output of the data shift register to the NRZ data formatter to control assertion of a series of zeros and sync byte (data preamble), and the sector of data contained in the selected data buffer. After the proper sector has been located (header found and ECC or CRC pattern verified), the microcontroller asserts a UCLR FIFO CNTR H pulse to the data buffer and data register control logic. The UCLR FIFO CNTR H input causes the ADDRESS asserted to the selected FIFO to be reset to zero. The microcontroller also clears the data shift register by asserting a UCLR DSR L pulse. After the data shift register is cleared (the DSR 0 output has been reset), the microcontroller loops until the data intervals required to write the series of zeros to the data preamble have been asserted to the disk drive.

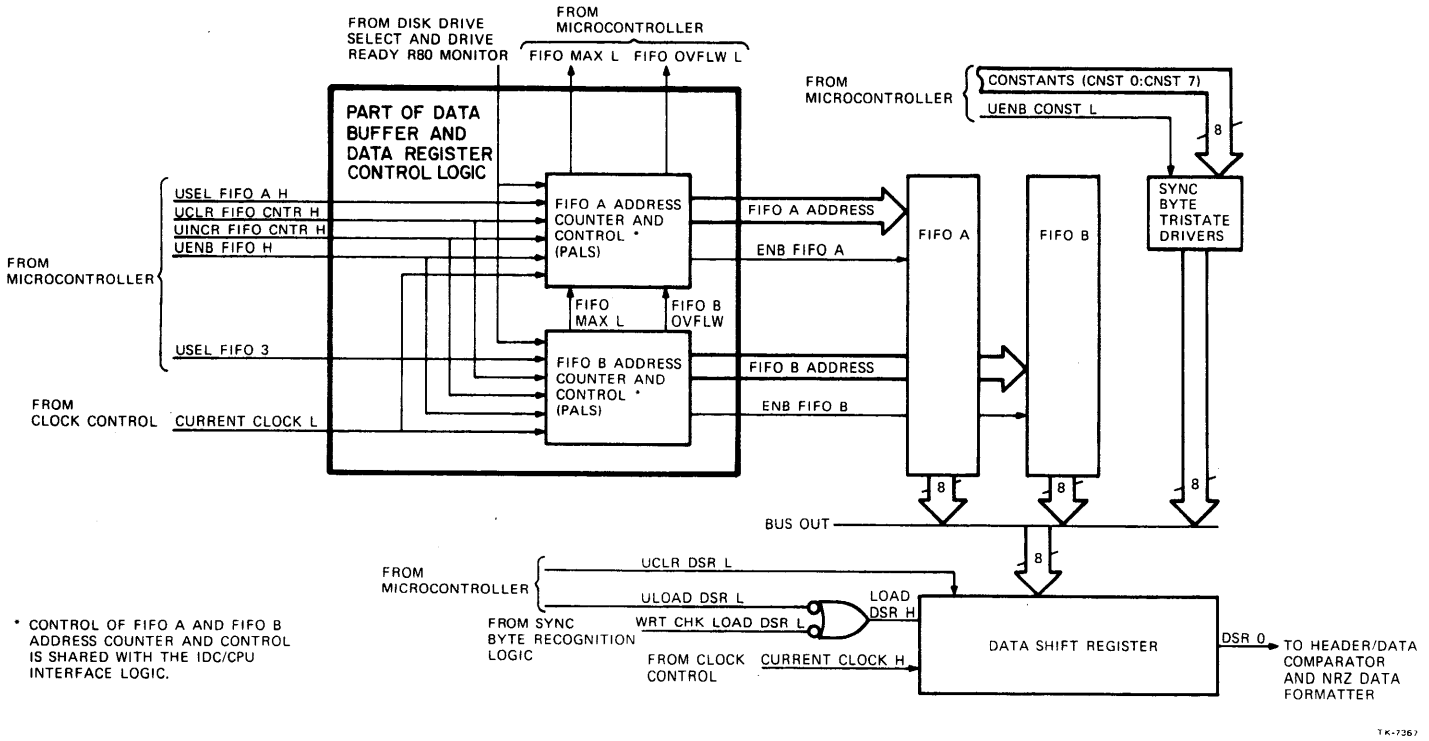


Figure 3-19 Data and Sync Byte Serialization Control Logic Functional Block Diagram

As the last zero bit of the data preamble is being written, the microcontroller asserts a UENB CONST L signal to the sync byte tristate drivers that enables the CONSTANTS output of the microcontroller (which has been preset to the appropriate sync byte pattern) to be asserted to the parallel input of the data shift register. The microcontroller also asserts a ULOAD DSR L pulse which causes the sync byte pattern to be loaded into the data shift register with the next positive transition of the CURRENT CLOCK H input. When the data shift register is loaded, the first bit of the sync byte is asserted on the DSR0 output. Then, with the leading edge of each CURRENT CLOCK H input, each successive bit of the sync byte pattern is asserted on the DSR0 output.

During the interval that the last sync byte bit is being asserted, the microcontroller asserts the UENB FIFO H signal and UINCR FIFO CNTR H pulse to the data buffer and data register control logic and a ULOAD DSR L pulse to the data shift register. The UENB FIFO H input to the data buffer and data register control logic is combined with the USEL FIFO (A or B) input to generate the ENB FIFO (A or B) output. The ENB FIFO output is asserted to the selected FIFO where it enables the data byte stored at the current FIFO ADDRESS location specified (ADDRESS 0) to be asserted to the parallel input of the data shift register. At the data shift register, the ULOAD DSR L input enables the data byte asserted from the selected data buffer to be loaded with the next positive transition of the CURRENT CLOCK H input.

Coincident with the loading of the data byte into the data shift register, the UINCR FIFO CNTR H pulse and CURRENT CLOCK L inputs to the selected FIFO address counter and control are combined to increment the FIFO ADDRESS. When the data byte is loaded (directly after the data interval in which the last bit of the sync byte was asserted to the DSR 0 output), the first bit of the data byte is asserted on the DSR 0 output. Then, with the leading edge of each CURRENT CLOCK H input, each successive bit of the first data byte is asserted on the DSR 0 output.

During the data interval in which bit 7 of the first data byte and bit 7 of each successive data byte is being asserted on the DSR 0 output, the microcontroller asserts a UINCR FIFO CNTR pulse to FIFO A and B address counter and control and a ULOAD DSR L signal to the data shift register. The ULOAD DSR L signal input to the DSR enables the data byte from the current FIFO ADDRESS to be loaded into the disk address register with the next positive transition of the CURRENT CLOCK H signal. The UINCR FIFO CNTR H pulse enables the FIFO ADDRESS asserted to the selected FIFO to be incremented. When the FIFO ADDRESS has been incremented to 256, if an RL02 disk drive is selected, or 512, if the R80 disk drive is selected, the FIFO A address counter and control asserts a FIFO MAX L signal to the microcontroller This signal signifies that after the next seven bits are asserted, the entire sector of data has been serialized and asserted on the DSR 0 output of the data shift register. A timing diagram showing the relationship of the control signals used in serializing the data from the data buffers and sync byte tristate drivers is presented in Figure 3-20.

During the write check function, the inputs from the microcontroller cause serialization of bytes 1 through 255 (RL02) or 511 (R80) of the data contained in the selected FIFO in the same manner as discussed in the preceding two paragraphs. However, byte 0 of the data contained in selected FIFO is loaded into the data shift register and the FIFO address counter is incremented as discussed in the following paragraph.

The microcontroller is in a stall condition until after the sync byte preceding the read data to be compared with the data from the selected FIFO has been located. Thus, before setting up the conditions for locating the sync byte, the microcontroller asserts a UCLR FIFO CNTR H pulse to the data buffer and data register control logic. The UCLR FIFO CNTR H input causes the ADDRESS asserted to the selected FIFO to be reset to zero. The microcontroller then generates and asserts a UENB FIFO signal to the data buffer and data register control logic. The UENB FIFO signal is combined with the USEL FIFO (A or B) input to generate the ENB FIFO (A or B) output asserted to the selected FIFO. The ENB FIFO signal enables the data contained at the current FIFO ADDRESS specified (ADDRESS 0) to be asserted to the parallel input of the data shift register. At the same time that the microcontroller asserts the UENB FIFO signal, it stalls until after the sync byte is found.

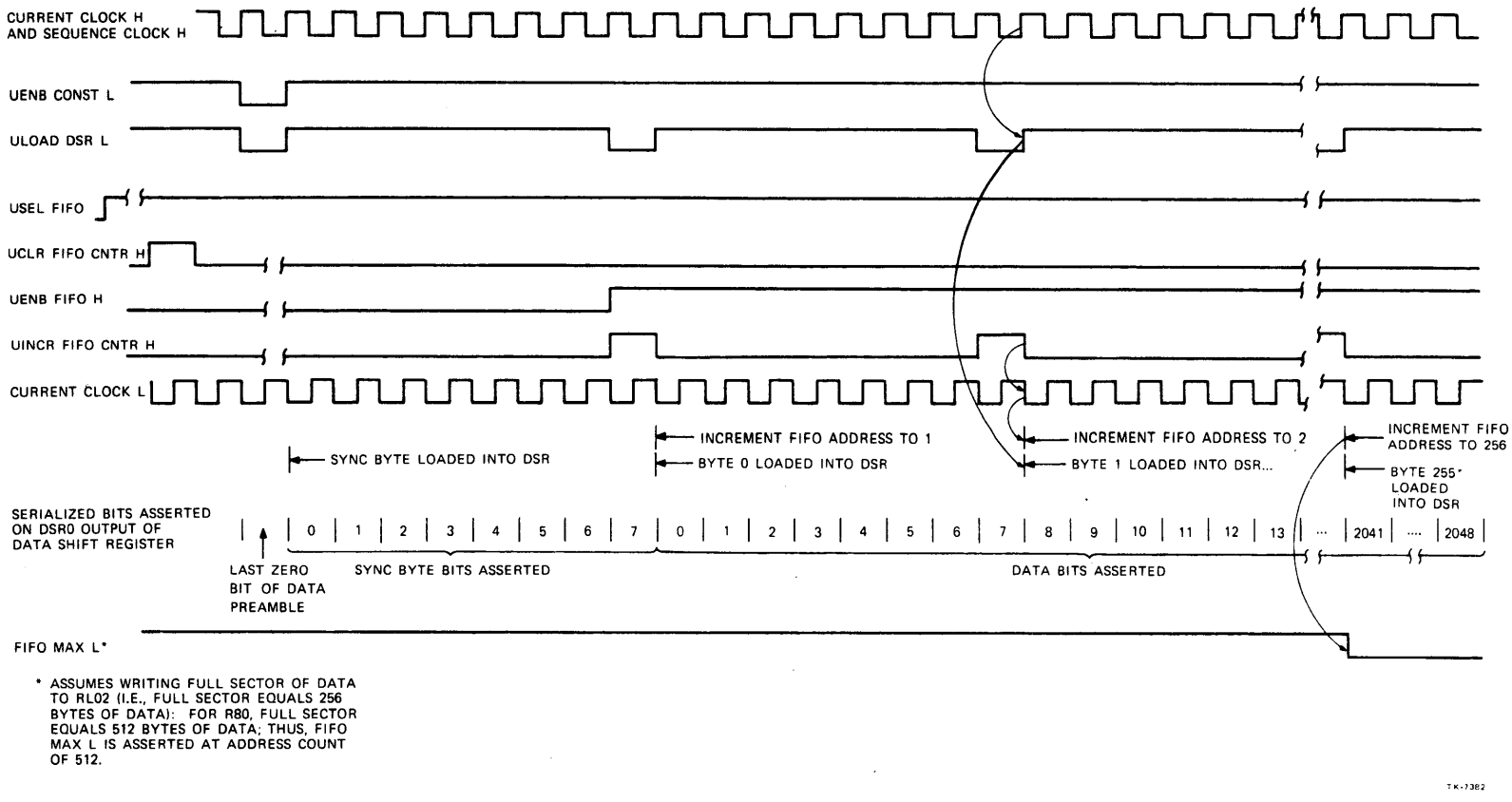


Figure 3-20 Data and Sync Byte Serialization Control Logic Timing Diagram

When the sync byte is found, the sync byte recognition logic generates and asserts a WRT CHK LOAD DSR L signal to the data shift register. The WRT CHK LOAD DSR L signal together with the next positive transition of the CURRENT CLOCK H input loads the data byte asserted from the selected FIFO into the data shift register. When the first byte is loaded, bit zero of the first data byte is asserted on the DSR 0 output. With each successive positive transition of the CURRENT CLOCK H input, each successive bit of the first data byte is asserted on the DSR 0 output.

While bit 1 of the first data byte is being asserted on the DSR 0 output, the microcontroller is again started. When started, the microcontroller asserts a UINCR FIFO CNTR H signal to the data buffer and data register control logic. The UINCR FIFO CNTR H signal is combined with the USEL FIFO (A or B) and CURRENT CLOCK L inputs to increment the selected FIFO address counter, which enables the second data byte from the FIFO to be asserted to the parallel input of the data shift register.

While bit 7 of the first and each successive data byte is being asserted on the DSR 0 output of the data shift register, the microcontroller generates and asserts a ULOAD DSR L pulse to the data shift register and a UINCR FIFO CNTR H pulse to the data buffer and data register control logic. These signals cause loading of the data shift register and incrementing the FIFO ADDRESS as discussed for serializing the data from the data buffers during the write data function. This process is continued until the FIFO MAX L signal from the FIFO A address counter and control is asserted to the microcontroller. The FIFO MAX L signal indicates that the full sector of DATA contained in the selected FIFO has been loaded into the data shift register.

3.5.12 Formatting and Loading Disk Drive Read Data in Data Buffers

During a read data function the read data tristate drivers and data buffer (FIFO A or FIFO B) are used to convert the serial READ DATA input from the disk drive to byte format and to load the formatted data into the selected FIFO. When a read data function is specified by the IDC control word input to the IDC, the microcontroller selects the FIFO to be used by asserting the appropriate USEL FIFO (A or B) signal to the data buffer and data register control logic (see Figure 3-21).

After the proper data sector has been located (header found and ECC or CRC pattern verified), the microcontroller asserts a UCLR FIFO CNTR H pulse to the data buffer and data register control logic. The UCLR FIFO CNTR H input causes the address asserted to the selected FIFO to be reset to zero. Then the microcontroller stalls until after the SYNC BYTE preceding the data portion of the READ DATA has been located.

While the IDC is looking for the sync byte and after the sync byte is found, each bit of the data read from the selected disk drive is asserted on the READ DATA H input to the read data synchronizer. The READ DATA H input to the read data synchronizer is sampled at the midpoint of each data bit interval by the CURRENT CLOCK L input, and the condition of the READ DATA H input (a logical 0 or 1) is loaded into the read data synchronizer. A diagram showing the timing relationship of the signals and events discussed in the following paragraphs is presented in Figure 3-22.

When the read data synchronizer is loaded, it asserts the sampled condition of the READ DATA H input to the data shift register via the SERIAL DATA IN signal line. The CURRENT CLOCK H input to the data shift register loads the SERIAL DATA IN signal asserted into DSR7 of the data shift register and shifts the current contents of DSR7:1 to DSR6:0, respectively.

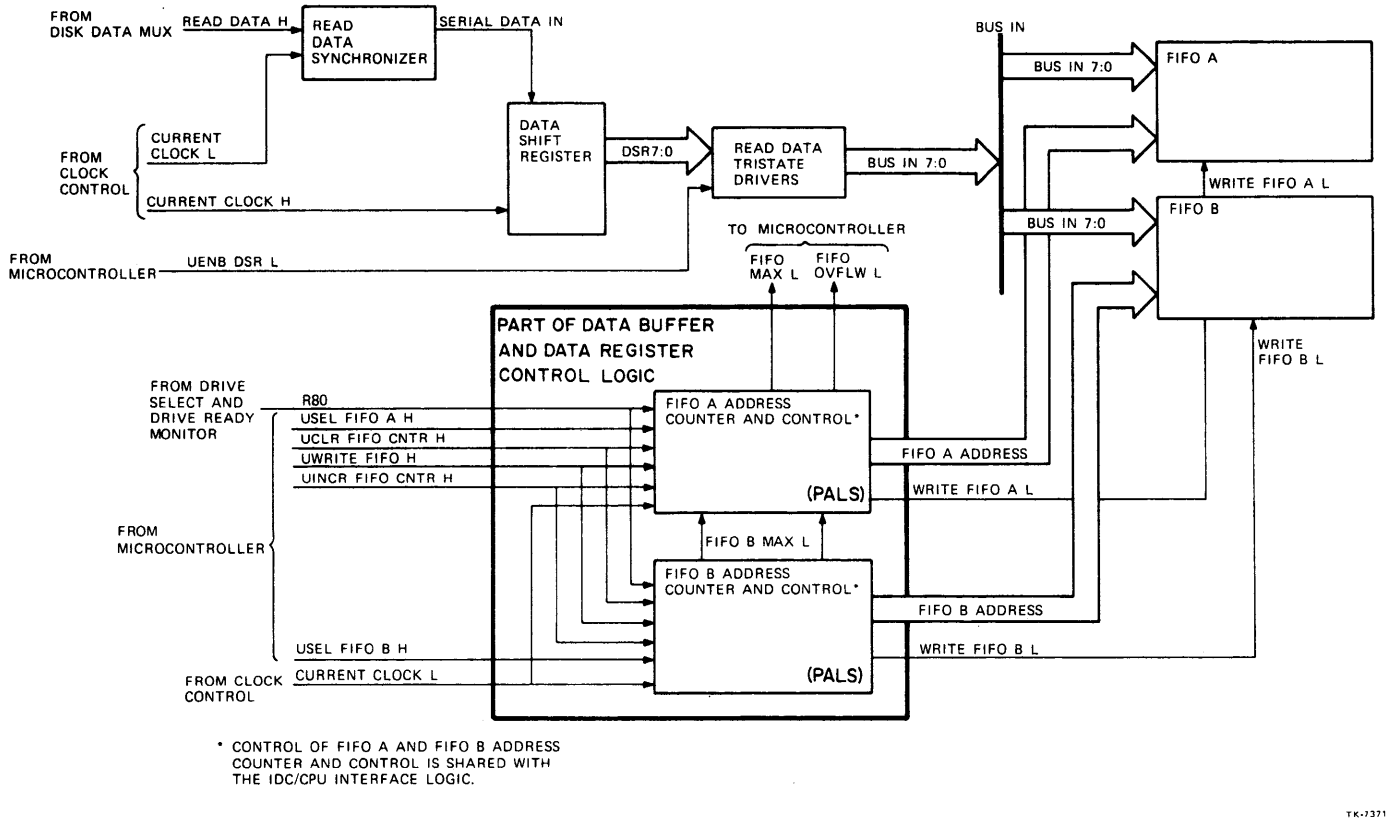


Figure 3-21 Read Data Formatting and Storage Control Logic Functional Block Diagram

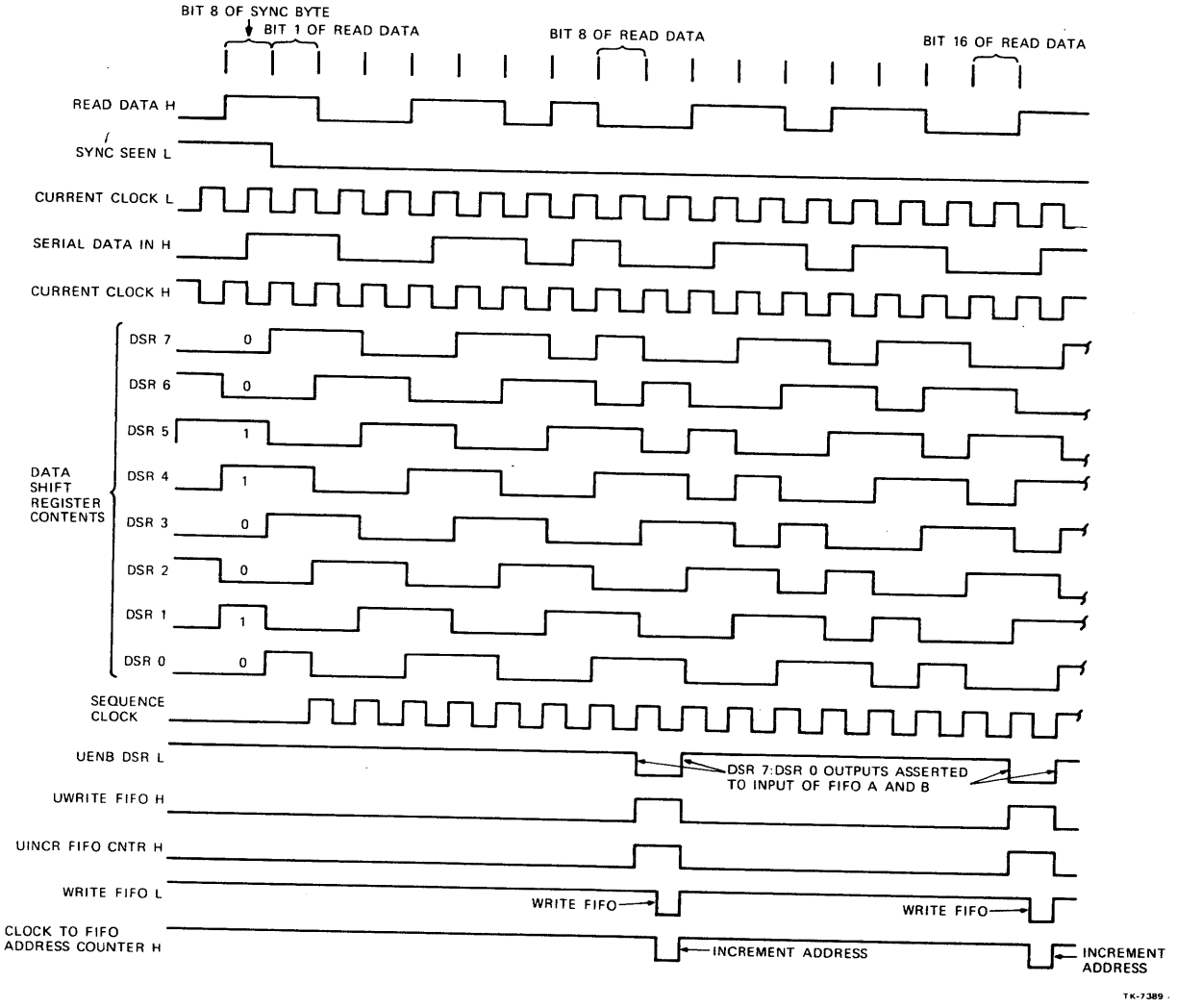


Figure 3-22 Formatting and Loading Read Data Input to FIFO: Timing Diagram

After the sync byte has been found, the microcontroller is restarted. The microcontroller is restarted at the same time that the first data bit of the READ DATA H input is loaded into DSR7 of the data shift register. Once the microcontroller is restarted, it counts the number of CURRENT CLOCK H pulses asserted to control assertion of the UENB DSR L, UWRITE FIFO H, and UINCR FIFO CNTR H outputs. (The CURRENT CLOCK H pulses are derived from the read data input and thus are synchronized each data bit interval.)

During the data interval in which each eighth data bit of the READ DATA is being loaded into the data shift register, the microcontroller generates and asserts a UENB DSR L signal to the read data tristate drivers, and the UWRITE FIFO H and UINCR FIFO CNTR signals to the FIFO A and FIFO B address counter and control of the data buffers and data register control logic.

The UENB DSR L input to the read data tristate drivers enables the parallel output of the data shift register (DSR7:0) to be asserted to the input of the FIFO A and FIFO B data buffers. The UWRITE FIFO H and UINCR FIFO CNTR inputs to the FIFO A and FIFO B address counter and control are combined with the USEL FIFO (A or B) and CURRENT CLOCK L inputs to generate the WRITE FIFO (A or B) L signal, which loads the data byte asserted from the read data tristate drivers into the selected FIFO, and the clock input to the selected FIFO address counter to increment the ADDRESS asserted to the selected FIFO. This process (sampling the READ DATA H input, shifting the sampled data into the data shift register, enabling and loading the parallel data output of the data shift register into the selected FIFO, and incrementing the FIFO ADDRESS counter) is repeated with each eight bits sampled until the FIFO ADDRESS has been incremented to 255 (if the data are being read from an RL02 disk drive) or 511 (if the data are being read from the R80 disk drive). When the FIFO address counter has been incremented to a count of 255 or 511 (depending on the state of the R80 signal input to the address counter and control), the address counter and control asserts a FIFO MAX L signal to the microcontroller. The FIFO MAX L signal indicates that the data portion of one sector of READ DATA has been converted to byte format and has been loaded into the selected data buffer.

3.5.13 IDC/CPU Interface Logic

The IDC/CPU interface logic enables the CPU to control loading the CSR, disk address register, and data buffers (FIFO A and FIFO B), and to control reading the CSR, disk address register, data buffers (FIFO A and FIFO B), and ECC/CRC logic. (Figure 3-23 defines the type of words, data, or information that is loaded into or read from the IDC registers and buffers.) Also, the IDC/CPU interface logic enables the CPU to initialize the IDC logic and R80 disk drive and to reset the UBUS BR5 interrupt signal. A functional block diagram of the IDC/CPU interface logic is shown in Figure 3-24.

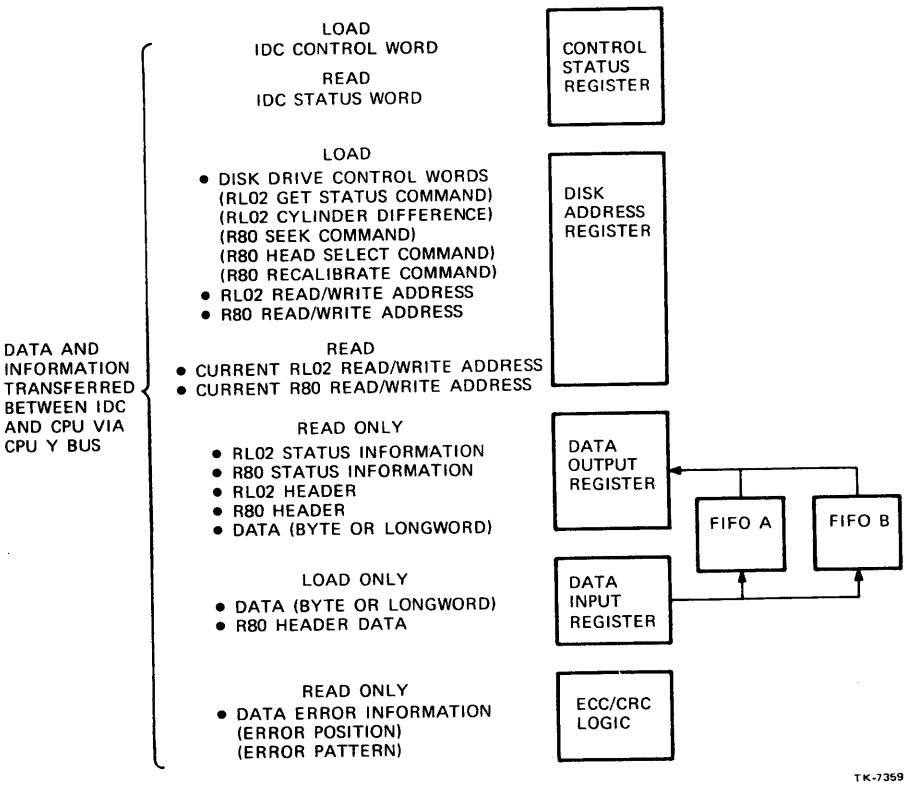


Figure 3-23 IDC Register Source and Destination for Data and Information Transferred between IDC and CPU via CPU Y-bus

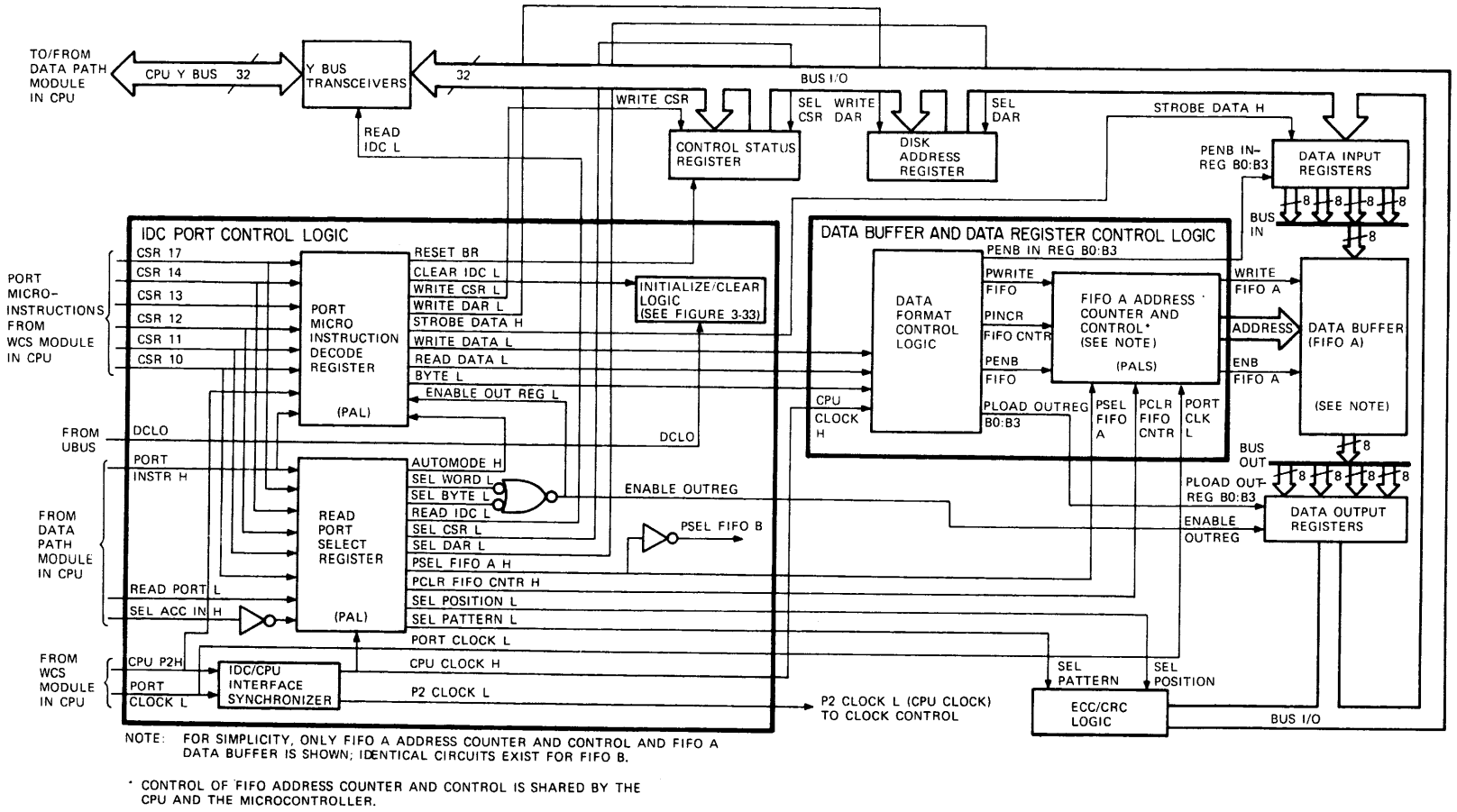


Figure 3-24 IDC/CPU Interface Logic Functional Block Diagram

For simplification, only the signals and control logic used for the control of one of the data buffers (FIFO A) is shown (the FIFO A address counter and control and the FIFO A data buffer). Identical logic exists for the control of FIFO B. Control of the FIFO A and FIFO B address counter and control is shared by the microcontroller and the CPU. This allows the microcontroller to cause loading or reading of the data buffers while the CPU is loading or reading the other data buffer. The microcontroller-initiated signal inputs to the FIFO A address counter and control are not shown in Figure 3-24. (CPU control of the data buffers is discussed in Paragraphs 3.5.11 and 3.5.12).

The IDC/CPU interface logic is synchronized with the CPU by the CPU timing signal inputs (CPU P2 H and PORT CLOCK L). The P2 CLOCK L output of the IDC/CPU interface logic is the basic CPU CLOCK signal used by the clock control to synchronize IDC operation with the CPU.

3.5.13.1 Loading CSR – The CPU causes loading of the CSR by asserting a WRITE CSR port microinstruction and a PORT INSTR signal to the port microinstruction decode register, and simultaneously asserting the word to be loaded via the CPU Y BUS (see Figure 3-24).

The port microinstruction decode register decodes the PORT MICROINSTRUCTION input and generates and asserts a WRITE CSR L signal to the CSR. The low-to-high transition of the WRITE CSR L signal input loads the word asserted on the BUS I/O via the CPU Y BUS and the Y-bus transceivers into the CSR.

Figure 3-25 shows a timing diagram illustrating the relationship of the PORT MICROINSTRUCTION, PORT INSTR, and CPU timing signal inputs to the IDC and the resultant signal (WRITE CSR L) that loads the CSR.

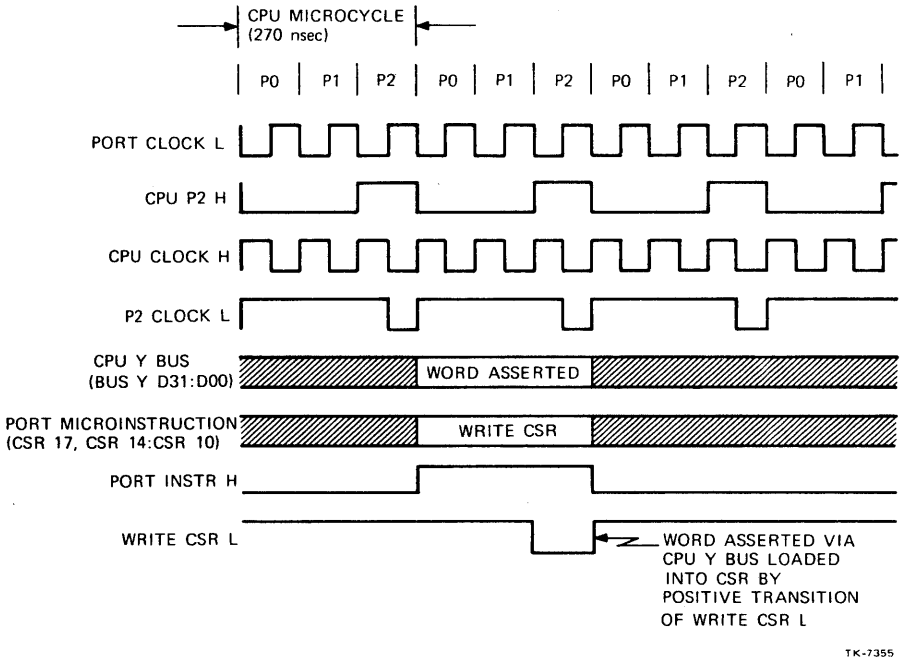


Figure 3-25 IDC Control Word Transfer Timing (CPU to IDC)

3.5.13.2 Reading CSR – To read and transfer the contents of the CSR to the CPU, the CPU asserts a READ CSR port microinstruction and a PORT INSTR signal to the read port interface select register, followed during a later CPU microcycle by a READ PORT L signal (see Figure 3-24). The READ CSR port microinstruction is loaded into the read port interface select register during clock phase 2 (CPU P2 H asserted) by the CPU CLOCK H input.

This conditions the read port select register such that a SEL CSR L output signal will be enabled by the READ PORT L signal input. When the READ PORT L signal is asserted and the SEL ACC IN H signal is not asserted (indicating that the READ PORT L signal is applicable to the IDC), the read port interface select register generates the SEL CSR L and READ IDC L outputs. The SEL CSR L output is asserted to the CSR where it enables the contents of the CSR to be asserted on the BUS I/O. The READ IDC L output is asserted to the Y-bus transceivers, where it enables the word asserted on the BUS I/O to be asserted to the CPU via the CPU Y BUS.

Figure 3-26 shows the timing relationship of the PORT MICROINSTRUCTION, PORT INSTR, READ PORT L, and CPU timing signals input to the IDC, the resulting IDC control signals, and the period during which the contents of the CSR are asserted to the CPU.

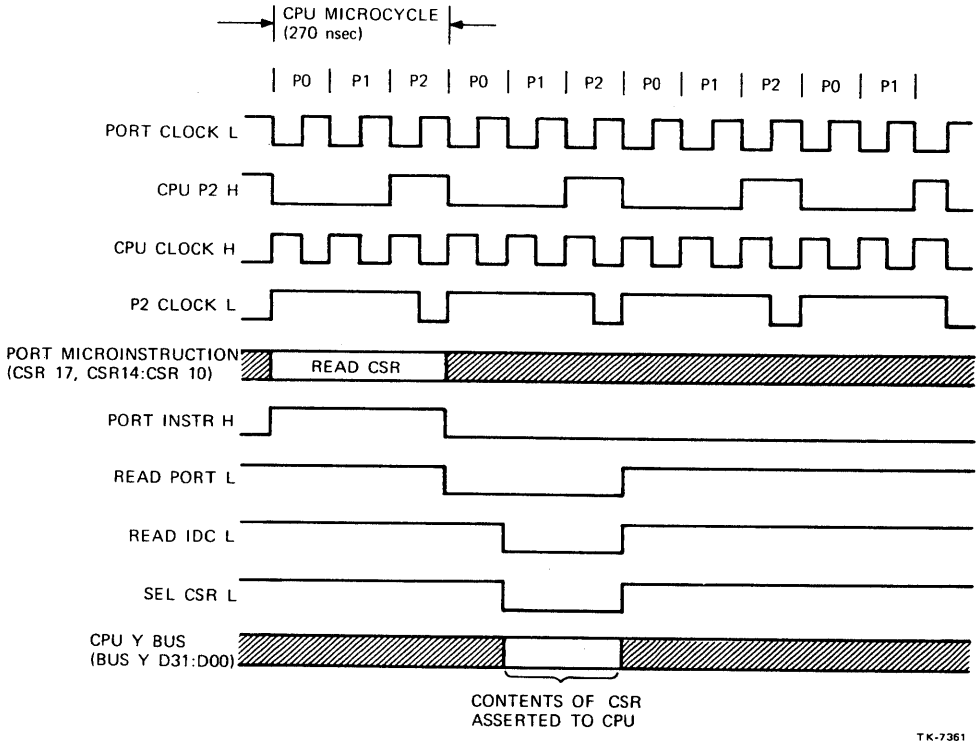


Figure 3-26 IDC Status Word Transfer Timing (IDC to CPU)

3.5.13.3 Loading Disk Address Register – The CPU loads the disk address register by asserting a WRITE DAR port microinstruction and a PORT INSTR signal to the port microinstruction decode register, and simultaneously asserting the word to be loaded via the CPU Y BUS (see Figure 3-24).

The port microinstruction decode register decodes the PORT MICROINSTRUCTION input and generates and asserts a WRITE DAR L signal to the disk address register. The low-to-high transition of the WRITE DAR L signal input loads the word asserted on the BUS I/O via the CPU Y BUS and the Y-bus transceivers into the disk address register.

Figure 3-27 shows a timing diagram illustrating the relationship of the PORT MICROINSTRUCTION, PORT INSTR, and CPU timing signal inputs to the IDC and the resulting signal (WRITE DAR L) that loads the disk address register.

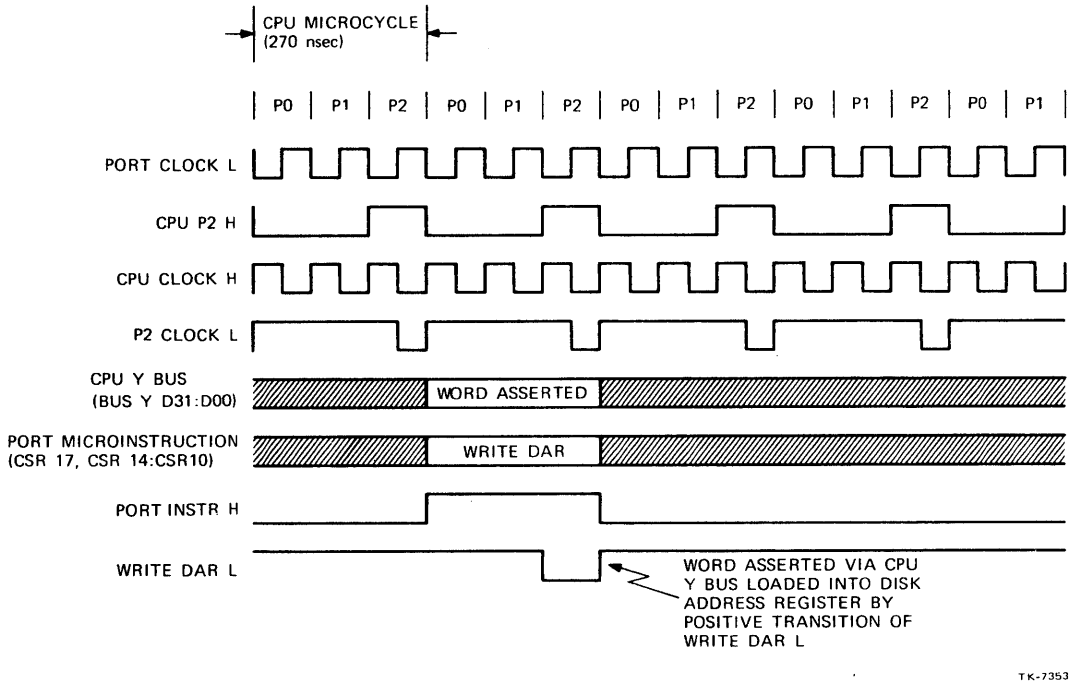


Figure 3-27 Disk Drive Control Word and Read/Write Address Transfer Timing (CPU to IDC)

3.5.13.4 Reading Disk Address Register – To read and transfer the contents of the disk address register to the CPU, the CPU asserts a READ DAR port microinstruction and a PORT INSTR signal to the read port interface select register, followed during a later CPU microcycle by a READ PORT L signal (see Figure 3-24). The READ DAR port microinstruction is loaded into the read port interface select register during clock phase 2 (CPU P2 H asserted) by the CPU CLOCK H input. This conditions the read port select register such that a SEL DAR L output signal will be enabled by the READ PORT L signal input. When the READ PORT L signal is asserted and the SEL ACC IN H signal is not asserted (indicating that the READ PORT L signal is applicable to the IDC), the read port interface select register generates the SEL DAR L and READ IDC L outputs. The SEL DAR L output is asserted to the disk address register where it enables its contents to be asserted on the BUS I/O. The READ IDC L output is asserted to the Y-bus transceivers, where it enables the word asserted on the BUS I/O to be asserted to the CPU via the CPU Y BUS.

Figure 3-28 shows the timing relationship of the PORT MICROINSTRUCTION, PORT INSTR, READ PORT L, and CPU timing signals input to the IDC, the resultant IDC control signals, and the period during which the contents of the disk address register are asserted to the CPU via the CPU Y BUS.

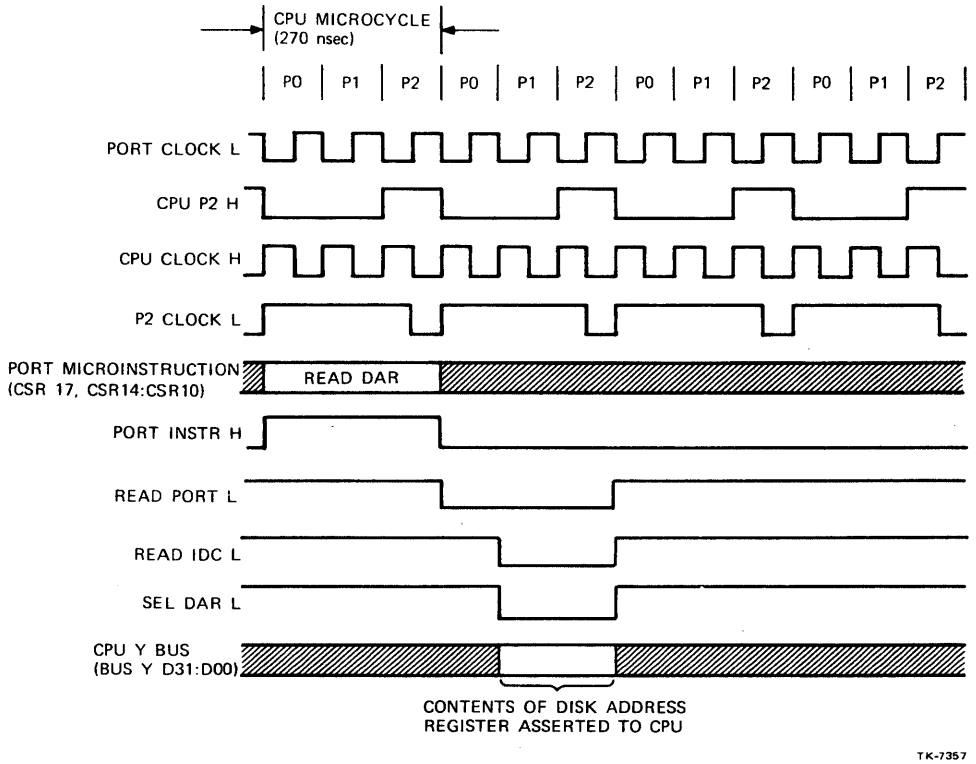


Figure 3-28 Current Read/Write Address Transfer Timing (IDC to CPU)

3.5.13.5 Reading ECC/CRC Logic – To read and transfer the contents of the ECC/CRC logic to the CPU, the CPU asserts a READ POSITION or READ PATTERN port microinstruction, as applicable, and a PORT INSTR signal to the read port interface select register, followed during a later CPU microcycle by a READ PORT L signal (see Figure 3-24). The READ POSITION or READ PATTERN port microinstruction is loaded into the read port interface select register during clock phase 2 (CPU P2 H asserted) by the CPU CLOCK H input. This conditions the read port select register such that a SEL POSITION L or SEL PATTERN L, as applicable, output signal will be enabled by the READ PORT L signal input. When the READ PORT L signal is asserted and the SEL ACC IN H signal is not asserted (indicating that the READ PORT L signal is applicable to the IDC), the read port interface select register generates the appropriate SEL POSITION L or SEL PATTERN L and READ IDC L outputs. The SEL POSITION or SEL PATTERN L output is asserted to the ECC/CRC logic where it enables the contents of the ECC position register or ECC pattern register, as applicable, to be asserted on the BUS I/O. The READ IDC L output is asserted to the Y-bus transceivers, where it enables the word asserted on the BUS I/O to be asserted to the CPU via the CPU Y BUS.

Figure 3-29 shows the timing relationship of the PORT MICROINSTRUCTION, PORT INSTR, READ PORT L, and CPU timing signals input to the IDC, the resulting IDC control signals, and the period during which the contents of the ECC/CRC logic are asserted to the CPU.

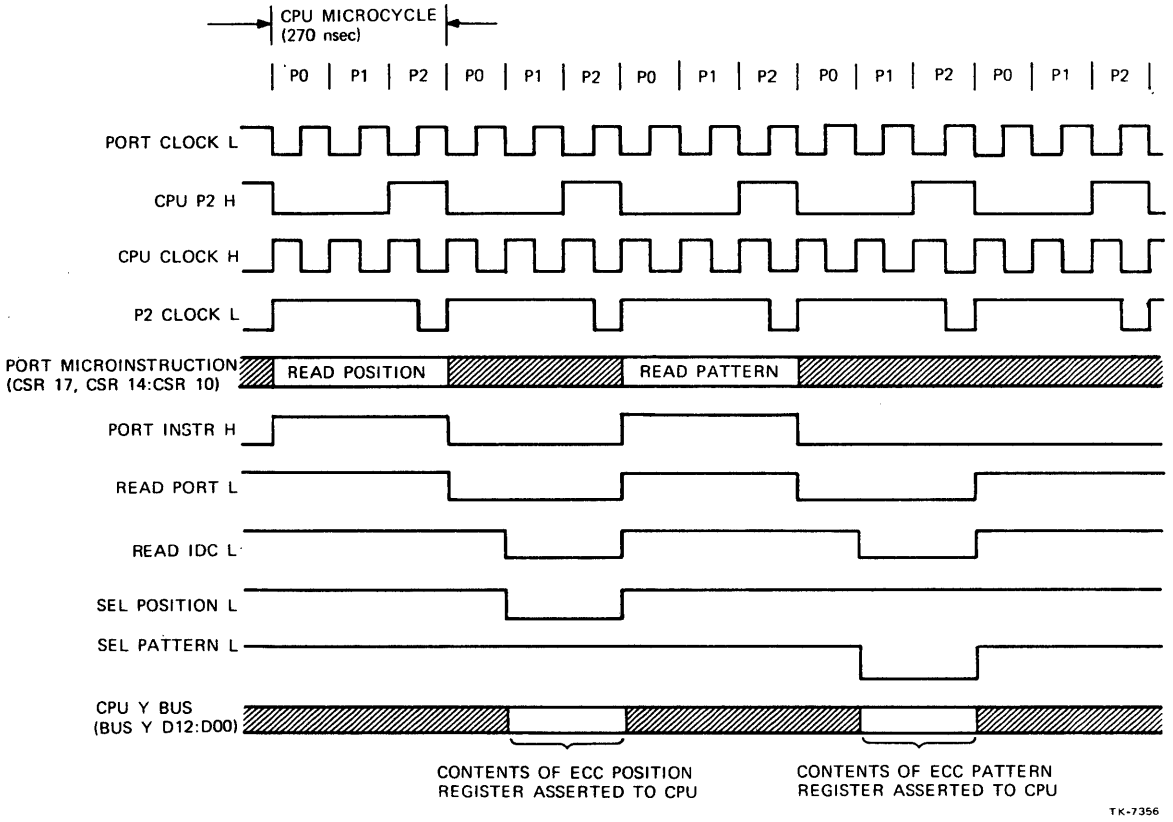


Figure 3-29 Data Error Information Transfer Timing (IDC to CPU)

3.5.13.6 Loading IDC Data Buffers – The CPU to load the IDC data buffer(s), the CPU selects the data buffer to be used (asserts a SELECT FIFO A or SELECT FIFO B port microinstruction and PORT INSTR signal) and clears the FIFO address counter and control logic (asserting a CLEAR FIFO CNTR port microinstruction and PORT INSTR signal). Then the CPU asserts a WRITE DATA BYTE or WRITE DATA WORD port microinstruction and PORT INSTR signal while simultaneously asserting via the CPU Y BUS the data byte or data longword to be loaded. The CPU must load the data buffer with a full sector of data [256 data bytes (64 data longwords), one full sector of RL02 data; or 512 data bytes (128 data longwords), one full sector of R80 data].

The read port select register decodes the SELECT FIFO A port microinstruction and generates and asserts a PSEL FIFO A H signal to the FIFO A address counter and control (see Figure 3-24). The read port select register decodes the CLEAR FIFO CNTR port microinstruction and generates and asserts a PCLR FIFO CNTR H pulse to the FIFO A and FIFO B address counter and control.

The PSEL FIFO A H signal and PCLR FIFO CNTR H pulse initiates resetting of the FIFO A ADDRESS asserted to the data buffer (FIFO A).

The port microinstruction decode register decodes the WRITE DATA BYTE or WRITE DATA WORD PORT MICROINSTRUCTION input and generates and asserts a STROBE DATA H pulse to the data input registers and a WRITE DATA L pulse to the data format control logic. If the PORT MICROINSTRUCTION input was WRITE DATA BYTE, the port microinstruction decode register also generates and asserts a BYTE L pulse to the data format control logic.

The STROBE DATA H pulse loads the data longword or data byte asserted on the BUS I/O via the CPU Y BUS and Y-bus transceivers into the data input registers. THE WRITE DATA L and BYTE L inputs to the data format control logic enable the proper sequence of control signals required to load the data byte input into the selected FIFO, or to convert the data longword input to four data bytes and load each of the four bytes into four contiguous storage locations of the selected FIFO.

If a data byte is to be loaded into FIFO A, the WRITE DATA L and BYTE L inputs to the data format control logic are used with the CPU CLOCK H input to enable the PENB INREG BO, PWRITE FIFO, and PINCR FIFO CNTR outputs. The PENB INREG BO signal enables the contents of INREG BO of the data input registers to be asserted to the inputs of FIFO A and FIFO B. The PWRITE FIFO and PINCR FIFO signals are asserted to the FIFO A address counter and control where they are used with the PSEL FIFO A and PORT CLOCK L inputs to produce a WRITE FIFO A L signal. The WRITE FIFO A L signal loads the data byte asserted from the data input register. The PSEL FIFOA and PORT CLOCK L inputs also produce a clock input to the FIFO A address counter to increment the ADDRESS asserted to FIFO A.

Figure 3-30 shows the timing relationship of the CPU timing signals, PORT MICROINSTRUCTION, PORT INSTR, and CPU Y BUS inputs to the IDC and the resulting control signals that are generated in loading FIFO A with the first data byte of the sector of data to be loaded.

If a data longword is to be loaded, the WRITE DATA L input to the data format control logic enables the series of PENB INREG B0:B3, PWRITE, and PINCR FIFO CNTR signals that enable the data longword input to be assembled into four data bytes and loaded into four contiguous storage locations within FIFO A.

Figure 3-31 shows the timing relationship of the CPU timing signals, PORT MICROINSTRUCTION, PORT INSTR, and CPU Y BUS inputs to the IDC and the resulting control signals that cause loading into the data buffer the first data longword of the sector of data to be loaded.

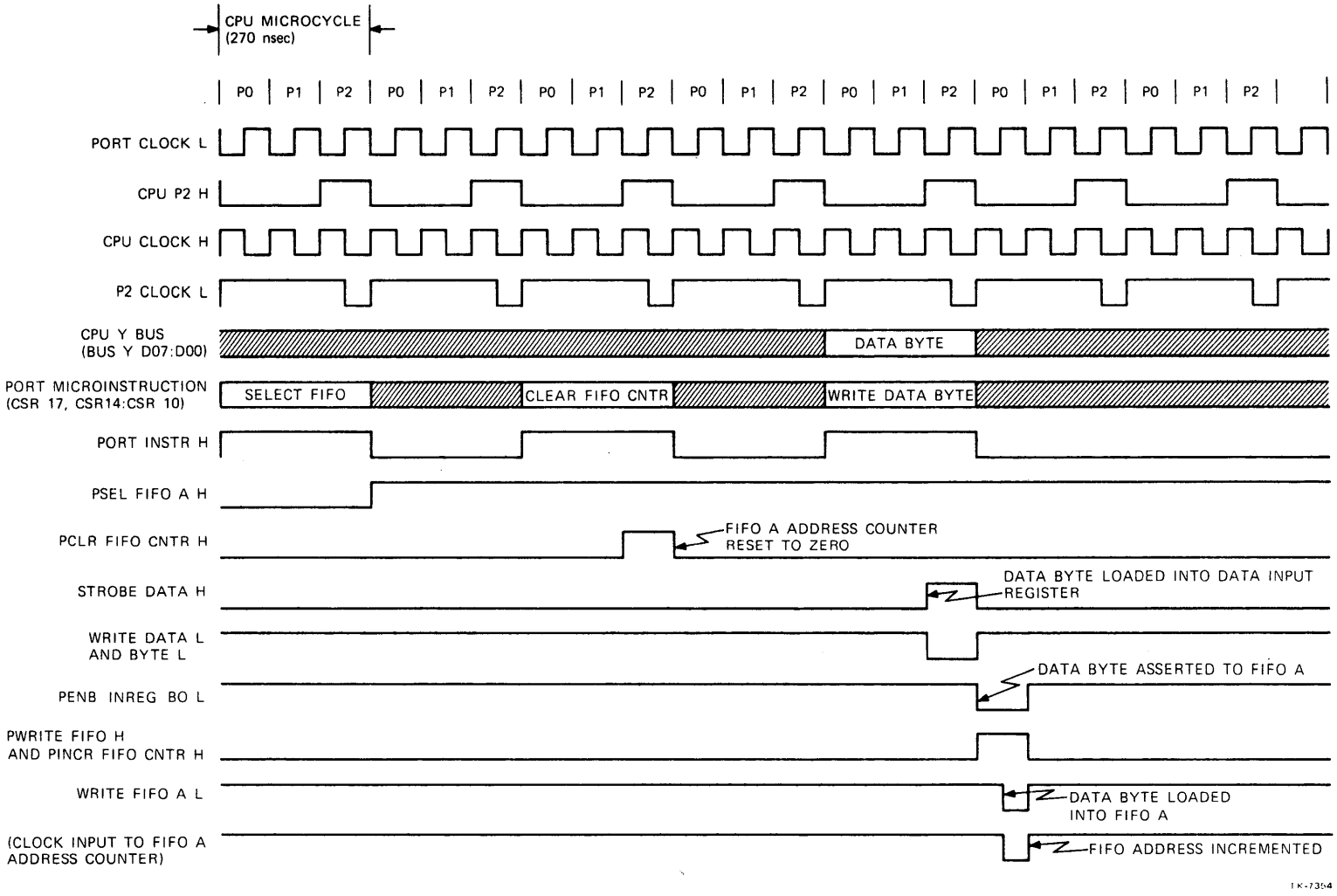


Figure 3-30 Data Byte Transfer Timing (CPU to IDC)

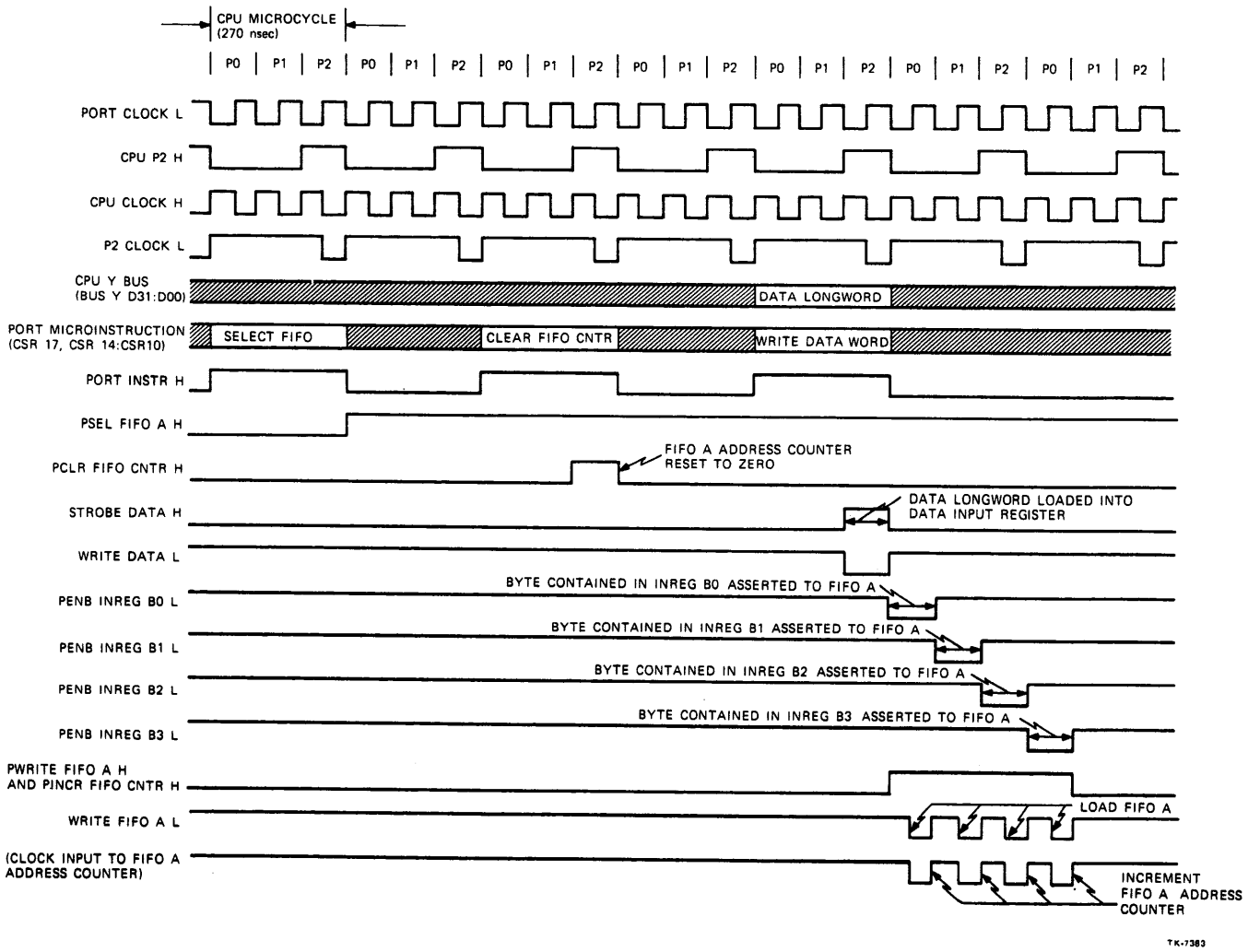


Figure 3-31 Data Longwood Transfer Timing (CPU to IDC)

3.5.13.7 Reading IDC Data Buffers – The CPU reads the IDC data buffer(s) by selecting the data buffer to be used (asserting a SELECT FIFO A or SELECT FIFO B port microinstruction and PORT INSTR signal), and clearing the address counter and control logic (asserting a CLEAR FIFO CNTR port microinstruction and PORT INSTR signal). Then the CPU asserts a READ DATA BYTE or READ DATA WORD port microinstruction and PORT INSTR, followed during a later CPU microcycle by a READ PORT L signal.

After the CPU selects the data buffer to be read and clears the FIFO address counter, the CPU may read a single byte or data longword or a series of them by asserting the applicable READ DATA BYTE or READ DATA WORD port microinstruction and PORT INSTR signal, followed during a later CPU microcycle by a READ PORT L signal for each data byte or data longword to be read. For reading a series of data longwords, the CPU may preset the CPU interface logic to the AUTOMODE, which enables a series of data longwords to be read by asserting only a READ PORT L signal for each successive data longword to be read. The CPU may preset the AUTOMODE function by asserting a SET AUTOMODE port microinstruction and PORT INSTR signal.

The read port select register decodes the SELECT FIFO A PORT MICROINSTRUCTION input and generates and asserts a PSEL FIFO A H signal to the FIFO A address counters and control (see Figure 3-24). The read port select register decodes the CLEAR FIFO CNTR PORT MICROINSTRUCTION input and generates and asserts a PCLR FIFO CNTR H pulse to the FIFO A and FIFO B address counter and control. The PSEL FIFO A H signal and the PCLR FIFO CNTR H pulse initiates resetting of the FIFO A ADDRESS asserted to the data buffer (FIFO A).

The READ DATA BYTE or READ DATA WORD port microinstruction is decoded by the port microinstruction decode register to produce the READ DATA L and BYTE L outputs or READ DATA L output, respectively. The READ DATA BYTE or READ DATA WORD port microinstruction is also loaded into the read port select register as a conditioning input to enable the SEL BYTE L or SEL WORD L output, respectively, when the READ PORT L signal is asserted during a following CPU microcycle.

The READ DATA L and BYTE L outputs of the port microinstruction decode register are asserted to the data format control logic. The READ DATA L and or BYTE L inputs enable the data format control logic to generate the proper sequence of output signals that cause gating of a single data byte from FIFO A and loading the data byte into the data output registers or cause gating of a series of four data bytes from four contiguous storage locations within the FIFO and loading the four data bytes in a longword format into the data output registers.

If a data byte is to be read, the READ DATA L and BYTE L inputs to the data format control logic are used with the CPU CLOCK H input to enable the PENB FIFO, PLOAD OUTREG BO, and PINCR FIFO CNTR outputs. The PENB FIFO signal is asserted to the FIFO A address counter and control where it is combined with the PSEL FIFO A input to produce the ENB FIFO A output. The ENB FIFO A signal is asserted to FIFO to enable the data byte contained in the current address location (specified by the ADDRESS input) to be asserted to the input of the data output registers. The PLOAD OUTREG BO output of the data format control logic is asserted to the data output registers to cause loading of the data byte asserted from FIFO A into register BO of the data output registers. The PINCR FIFO CNTR output of the data format control logic is asserted to the FIFO A address counter and control where it is used with the PSEL FIFO A and PORT CLOCK L inputs to generate a clock signal to increment the ADDRESS asserted to FIFO A.

If a data longword is to be read, the READ DATA L input to the data format control logic is used with the CPU CLOCK H input to enable the PENB FIFO, PLOAD OUTREG B0:B3, and PINCR FIFO CNTR outputs. The PENB FIFO and PINCR FIFO signals are asserted to the FIFO A address counter and control. The PENB FIFO signal is combined with the PSEL FIFO A input to produce the ENB FIFO A output, which is asserted to FIFO A to enable the data byte contained in the address location specified by the ADDRESS input to be asserted to the input of the data output registers. The PINCR FIFO CNTR signal is combined with the PSEL FIFO A and PORT CLOCK L inputs to generate a series of four clock pulses to sequentially increment the ADDRESS asserted to FIFO A and thus enable the data bytes from four contiguous address locations to be asserted to the data output registers. The PLOAD OUTREG B0 through PLOAD OUTREG B3 outputs of the data format control logic are enabled sequentially to enable the data bytes from the four contiguous addresses to be assembled into a data longword format in data output registers.

When the READ PORT L signal is asserted, following the READ DATA WORD or READ DATA BYTE port microinstruction, and the SEL ACC IN H is not asserted (indicating that the READ PORT L signal is applicable to the IDC), the read port select register generates the applicable SEL BYTE L or SEL WORD L and READ IDC L outputs. The SEL BYTE L or SEL WORD L output produces an ENABLE OUTREG L signal. The ENABLE OUTREG L signal is asserted to the data output registers to enable the contents of the data output registers to be asserted onto the BUS I/O. The ENABLE OUTREG L signal is also asserted to the port microinstruction decode register. The READ IDC L output is asserted to the Y-bus transceivers to enable the data byte or data word on the BUS I/O from the data output registers to be asserted to the CPU via the CPU Y BUS.

If the AUTOMODE function had been preset by a PORT MICROINSTRUCTION input to the read port select register, then the AUTOMODE H and the ENABLE OUTREG L inputs to the port microinstruction decode register will initiate a WRITE DATA L output signal. The WRITE DATA L output reinitiates loading of the data output registers with the next data longword to be read.

Figures 3-32 and 3-33 show the timing relationship between the CPU PORT MICROINSTRUCTION, PORT INSTR, READ PORT L and timing signal inputs to the IDC, the resulting IDC control signals and the period during which the requested data byte or data longword, respectively, is asserted to the CPU via the CPU Y BUS. Figure 3-34 shows the timing for data longword transfers to the CPU using the AUTOMODE function.

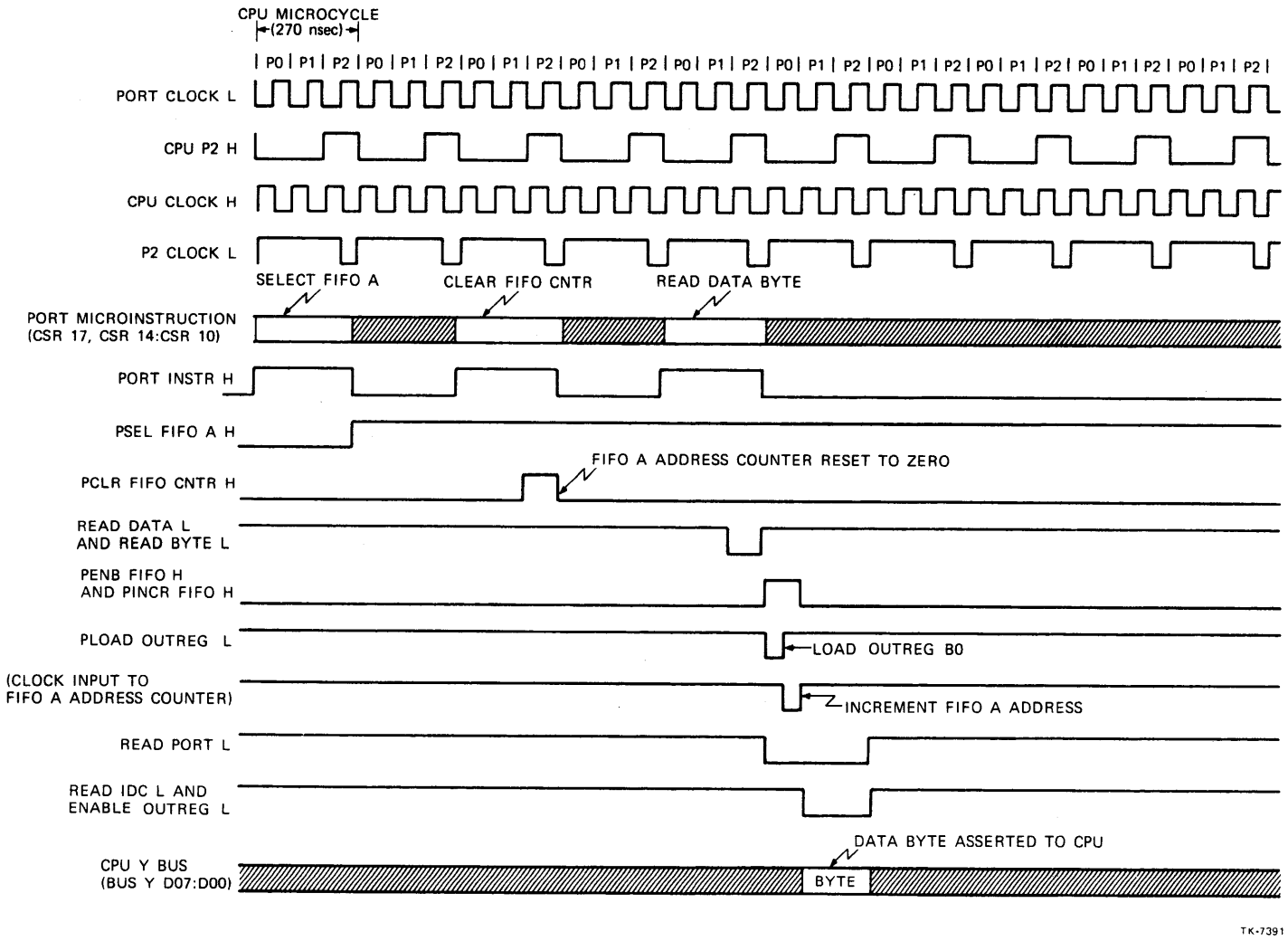


Figure 3-32 Data Byte Transfer Timing (IDC to CPU)

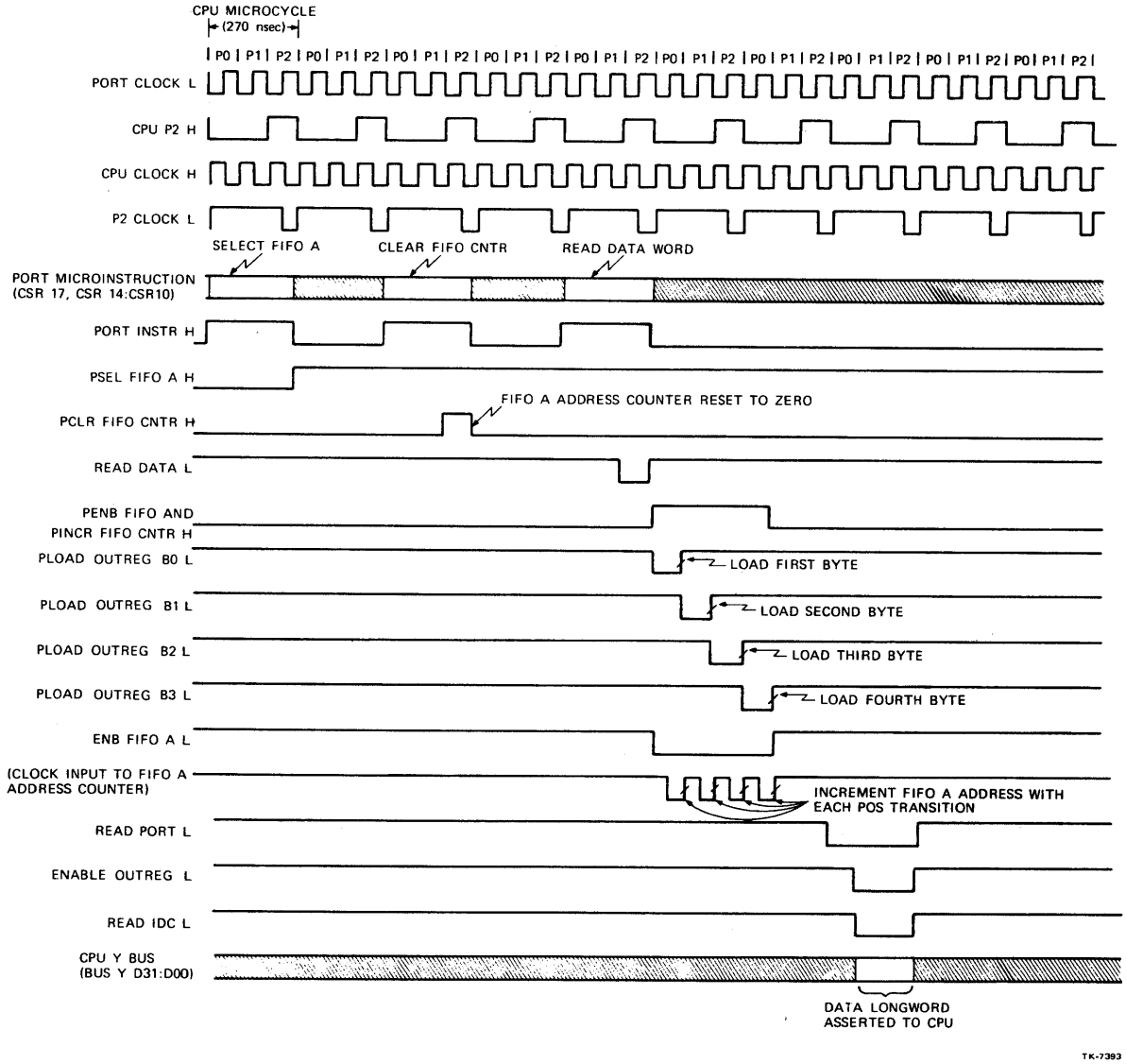


Figure 3-33 Single Data Longword Transfer Timing (IDC to CPU)

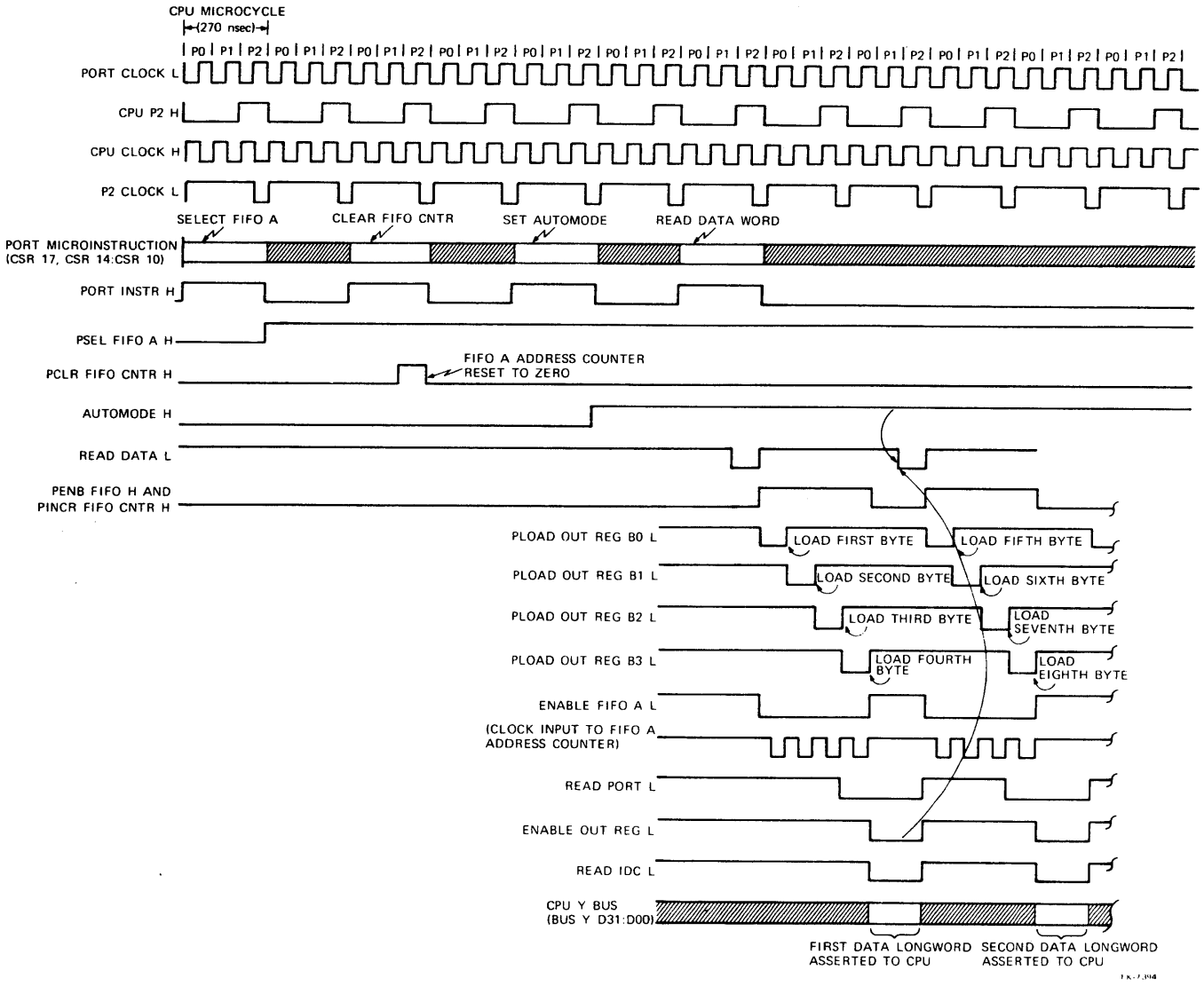


Figure 3-34 Automode Data Longword Transfer Timing (IDC to CPU)

3.5.13.8 Initializing/Clearing IDC and R80 Disk Drive – The IDC and R80 disk drive may be initialized under CPU control or initialized automatically following an interruption of operating voltages from the VAX-11/730 power system. A logic diagram of the initialize/clear logic is shown in Figure 3-35.

When a CLEAR IDC port microinstruction is asserted to the IDC, the resulting CLEAR IDC L signal, a 270 nanosecond pulse, is asserted to the initialize/clear logic. The CLEAR IDC L input is loaded into flip-flop 1 by the P2 CLOCK L input and transferred to flip-flop 2 by the second P2 CLOCK L input. The output of flip-flop 2, a 270 nanosecond positive pulse, is the initialize signal for the IDC and R80 disk drive.

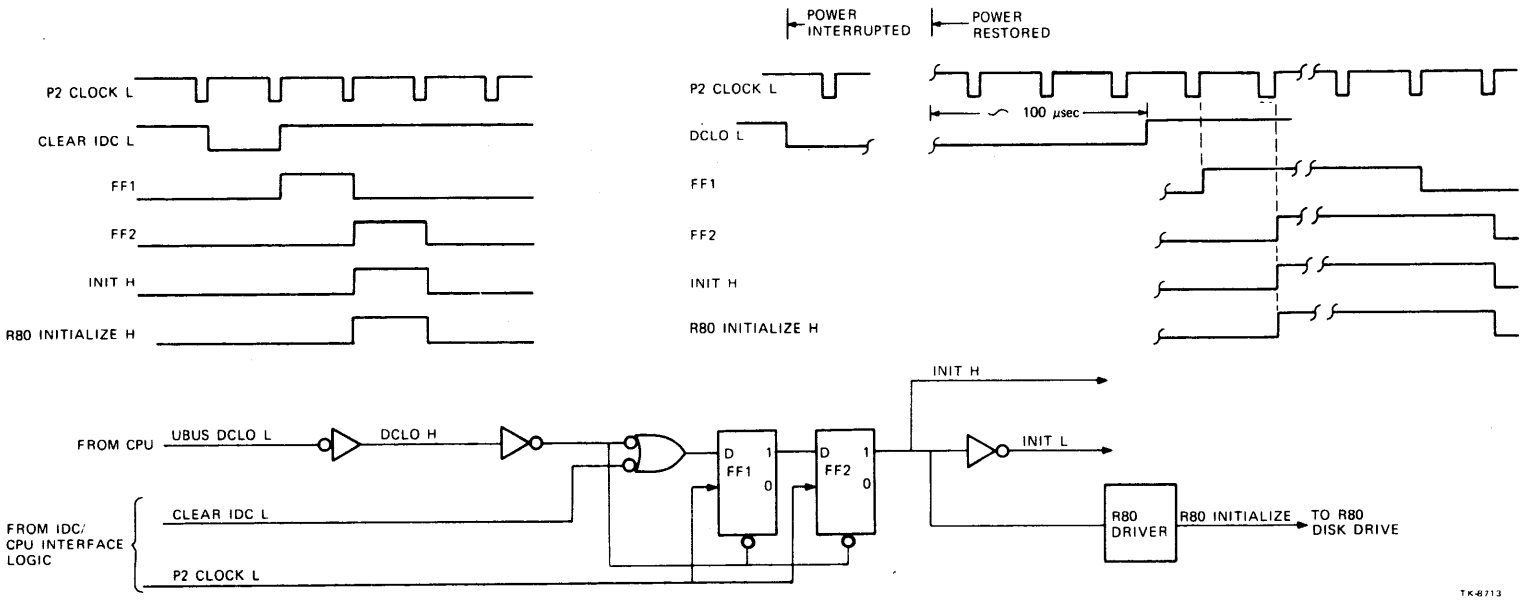


Figure 3-35 Initialize/Clear Logic Diagram

The UBUS DCLO L input to the initialize/clear logic is asserted when dc power to the VAX-11/730 is interrupted or goes out of tolerance. Following the power interruption, DCLO L is held asserted until approximately 100 microseconds after dc power has been restored. During the period between restoration of dc power and deassertion of DCLO L, and for two CPU microcycles following deassertion of DCLO L, the initialize/clear logic holds the INIT H, INIT L, and R80 INITIALIZE signal outputs asserted.

The timing diagram in Figure 3-35 shows the interval of assertion of the INIT and R80 INITIALIZE outputs relative to the CLEAR IDC L and DCLO L signals asserted.

3.5.14 Microcontroller Branching, Loops, and Stalls

The microcontroller consists of eight 512 × 8 PROMs, branch enable multiplexers, loop counter, and microfunction decoders (see Figure 3-36). The eight 512 × 8 PROMs are addressed in parallel, which provides a 512 × 64 PROM with 512 addressable locations. Each address provides a unique 64-bit microword output. The microword output is made up of the following:

- Nine "next address" bits (NAD 8:0)
- Two loop counter control bits (ULOAD LOOP CNTR and UINCR LOOP CNTR)

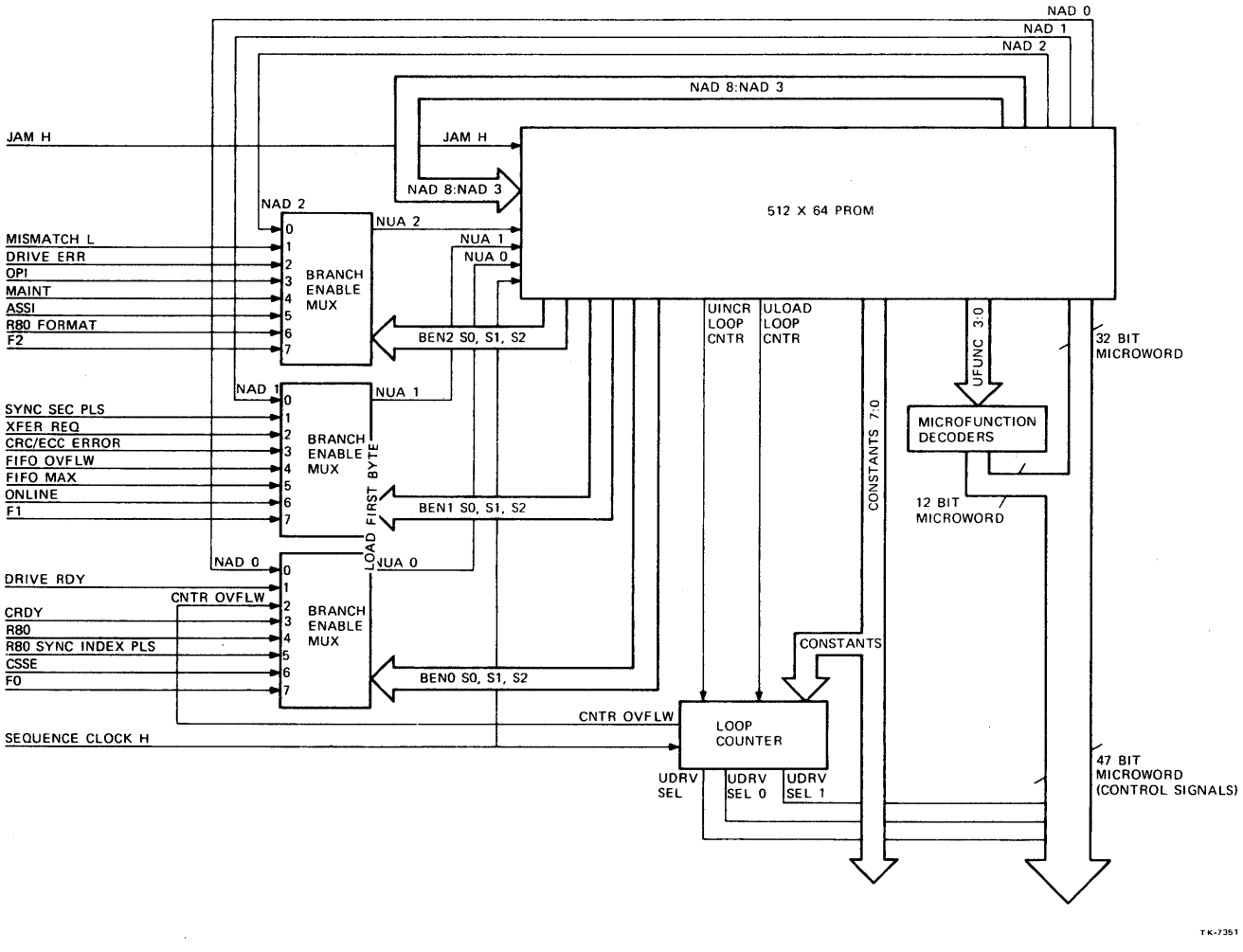


Figure 3-36 Microncontroller Functional Block Diagram

- An eight-bit selectable constant (CNST 7:0)
- Nine branch enable bits (BEN 2 S0, S1, and S2; BEN 1 S0, S1, and S2; and BEN 0 S0, S1, and S2)
- Four microfunction bits (UFUNC 3:0), a 47-bit microword

The 47-bit microword output is made up of a 32-bit microword from the 512 × 64 PROMs, the UDRV SEL, UDRV SEL 0, and U DRV SEL 1 bit outputs of the loop counter, and the 12-bit microword output of the microfunction decodes.

The nine "next address" bits are used as the base address in formulating the next address for the 512 × 64 PROM. The nine branch enable bits specify which (if any) branch conditions are examined to modify the three least significant bits of the base "next address."

The two LOOP CNTR control bits are used to load the loop counter with a selectable constant and enable the loop counter to be incremented by the SEQUENCE CLOCK input.

The eight-bit CONSTANTS output is used to present the loop counter and to generate the sync byte patterns.

The four microfunction bits are decoded in the microfunction decoders to provide a 12-bit microword.

The 12-bit microword output of the microfunction decoders, the drive select bit outputs of the loop counter, and the 32-bit microword output of the 512 × 64 PROM provide a 46-bit microword output of the microcontroller, which provide the control signals for the IDC.

3.5.14.1 Microcontroller Branching – The microword output from the IDC microcontroller is determined by the address input to the 512 × 64 PROM. The address is derived from the six most significant "next address" bits output from the PROMS (NAD 8:3) and the three "next microaddress" bits output from the branch enable multiplexers (NUA 2:0). The next microaddress bits are derived from the next address (NAD 2:0) or any one of the conditional inputs asserted to the microcontroller branch enable multiplexers. Each of the three branch enable multiplexers may select any one of the inputs to be asserted on the associated NUA signal line. The signals selected allow the three least significant address bits to enable an eight-location multiway branch.

3.5.14.2 Microcontroller Loops – The microcontroller may be made to loop on one or a series of addresses until a specific branch condition is satisfied.

3.5.14.3 Microcontroller Stalls – Microcontroller stalls are caused by interrupting the SEQUENCE CLOCK input. Once the microcontroller is stalled, it remains in the microstate initiated before the clock was interrupted until the SEQUENCE CLOCK input is reasserted.

3.5.15 Read Data Separator Operation

The read separator (Figure 3-37) converts MFM-encoded RL READ DATA into an IDC compatible format (NRZ). It also generates the DS CLOCK used to synchronize IDC operation with the timing of RL READ DATA inputs.

The data stream transferred from the disk (RL02) to the IDC consists of composite clock and data bits. With single density, a data bit is decoded by a data window that is generated from the clock bit. In double density, the lack of consistent clock bits makes it impossible to generate a data window in this manner. Instead, to separate clock and data bits the data separator circuit must first determine their nominal position and then generate a clock and data window that is centered around the bit positions.

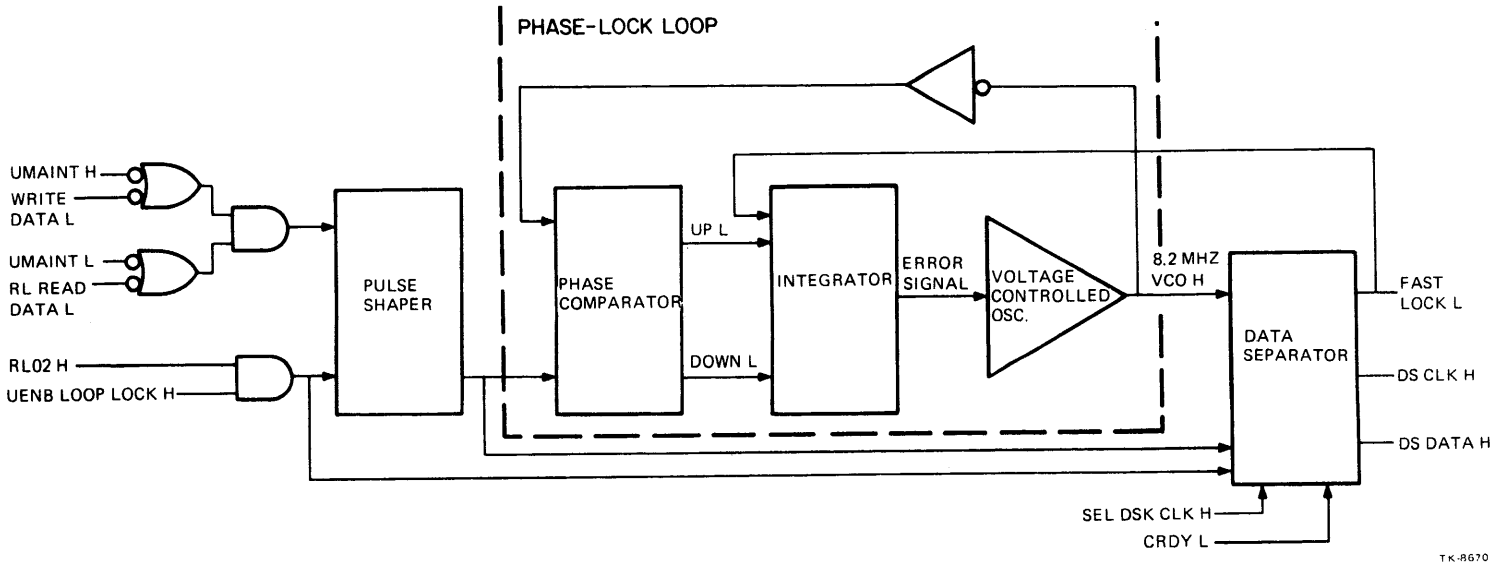


Figure 3-37 Read Data Separator Block Diagram

To determine the nominal bit position around which to center the window, the data separator must track data bit frequency changes. It uses the phase relationship between a bit and its window to vary the position of the window. In this way, even if an unpredictable bit shift occurs, the read data separator can adjust the window's position to compensate for the change.

Since a data pulse may occur at either the center or boundary of a bit cell, its location remains unpredictable for random data patterns. The only consistent pattern that may be used as the basis for data separation (MFM to NRZ) is the fact that MFM encoding guarantees that there will be at least one flux reversal on the disk for every two bit cells. This fundamental frequency makes it feasible to use phase lock loop techniques to form a self-clocking read data separator.

3.5.15.1 Phase Lock Loop (PLL) – The PLL is a closed loop circuit that locks onto the basic frequency of data bits (RL READ DATA L) read off the disk, and provides an output (8.2 megahertz VCO H) that is in phase anad frequency locked with the input data. Its output frequency is twice that of the incoming RL READ DATA bit rate. A simplified block diagram is shown in Figure 3-37. The input data to the pulse shaper can come from two different sources. The WRITE DATA L line provides a data path between the write precompensation circuit and the pulse shaper. This data path is used during the maintenance command as a means of sending write data back into the read circuits. The PL READ DATA L line is the data path followed when reading data off the disk. The RL READ DATA L pulses are standardized in the pulse shaper to a uniform 60 nanosecond pulse width and applied to the phase comparator and data separator. The other input to the phase comparator is the inverted output of the voltage-controlled oscillator (8.2 megahertz VCO H).

In the phase comparator, the phase of the RL READ DATA L pulse is compared with that of the VCO output (8.2 megahertz VCO H) as is illustrated in Figure 3-38A. The phase comparator then generates two signals (pulses) equal in duration to the phase difference: UP L if the VCO output (8.2 megahertz VCO H) is less than twice the data rate (Figure 3-38B) or DOWN L if the VCO output is more than twice the data rate (Figure 3-38C). If the VCO output is less than twice the data rate the VCO should speed up. Hence UP L is asserted and its width represents the magnitude of the speedup required. The same is true for slowdown if the VCO output is more than twice the data rate, (DOWN L asserted). These phase error outputs (UP L and DOWN L) are applied to the integrator, which generates the small error offset voltages required to control the VCO frequency and maintain loop lock.

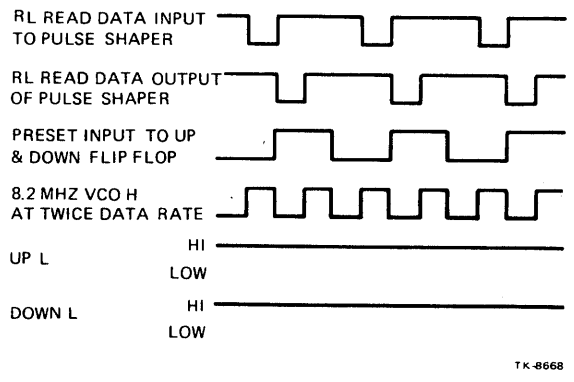


Figure 3-38a VCO Output at Twice Data Rate (Frequency Lock) Timing Diagram

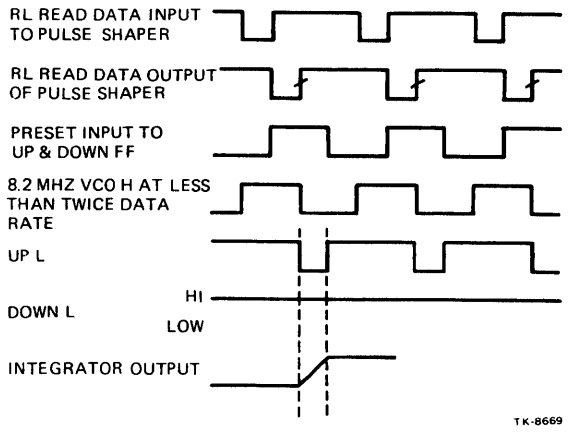


Figure 3-38b VCO Output Less Than Twice Data Rate Timing Diagram

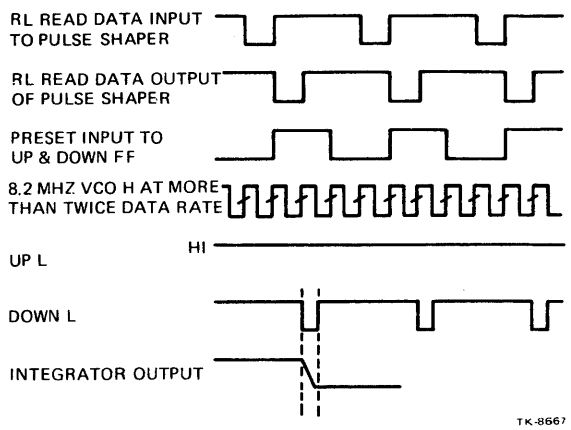


Figure 3-38c VCO Output More Than Twice Data Rate Timing Diagram

The integrator converts the UP L and DOWN L signals to an error voltage. The output (errror voltage) of the integrator is raised or lowered proportionally to the area under the UP L or DOWN L pulse (the integral of the pulse). Since the amplitude of the UP L and DOWN L pulses is fixed and the duration represents the amount of phase error, the change in error voltage due to the area under the pulse is also proportional to the phase error. The integration time constant (the rate at which the error voltage is allowed to change) is chosen such that the system will track long term frequency variations of the input but not respond to individual peak-shifted bits.

The voltage-controlled oscillator generates an output signal (8.2 megahertz VCO H) whose frequency is proportional to the voltage (error voltage) applied.

Figure 3-39 illustrates the relationship between the read data and the phase lock loop settling time. The phase lock loop is designed to lock onto read data frequency within four byte times.

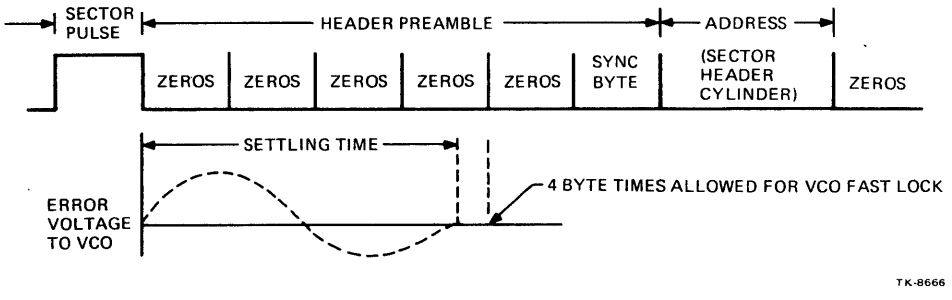


Figure 3-39 Loop Lock Settling Time

3.5.15.2 Data Separator – The data separator examines the incoming data stream and separates the pulses into DS DATA or DS CLOCK. A detailed diagram of the data separator is illustrated in Figure 3-40 and its timing sequence is illustrated in Figure 3-41.

When a read header command is decoded and the SYNC SECTOR PLS is detected, the header preamble from the selected drive appears on the RL READ DATA L line. The assertion of RLC2 H and UENB LOOP LOCK H allows the header preamble to enter the phase lock loop. The phase lock loop performs a fast lock using the first four bytes of zeros to synchronize itself with the RL READ DATA frequency. This fast lock is enabled by the assertion of FAST LOCK L from the enable flip-flop.

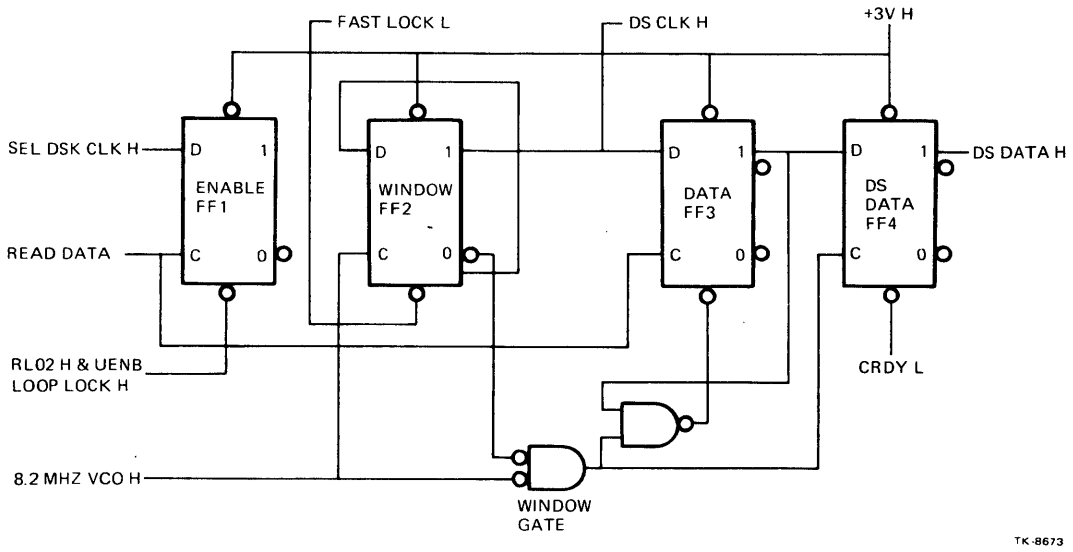


Figure 3-40 Data Separator Detailed Diagram

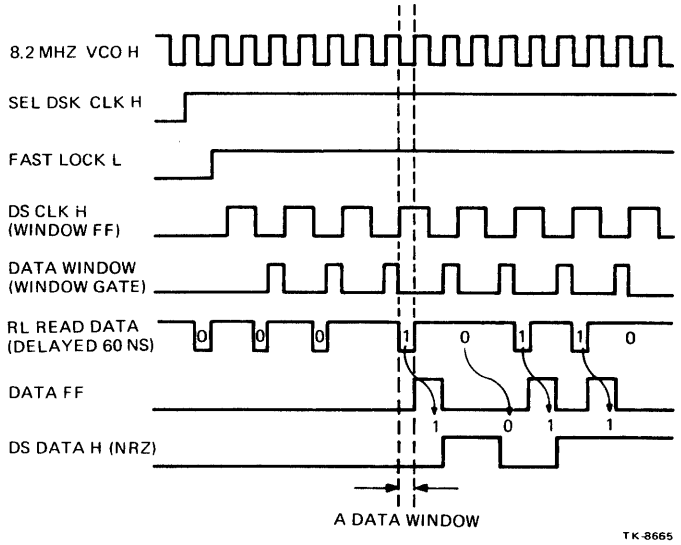


Figure 3-41 Data Separator Timing Diagram

After synchronization (fast lock), SEL DSK CLK H is asserted to enable the data separator. The enable flip-flop will set on the detection of the first data pulse and remains set until the data separator is disabled by the negation of SEL DSK CLK H.

The window flip-flop is now allowed to toggle under control of the 8.2 megahertz VCO H input producing two outputs (normal and inverted) at 4.1 megahertz each. The normal output (DS CLK H) is synchronized with DS DATA H and is asserted on the CURRENT CLOCK output of the clock control. When the window flip-flop is set, the inverted (low) output indicates a window time during which data pulses are interpreted as cell center pulses (data ones). When reset, it indicates a window time during which data pulses are interpreted as cell boundary pulses (data zeros).

The data flip-flop sets only when a data one occurs during the assertion of the window (window flip-flop set).

The window gate generates a pulse that simultaneously clocks the DS data flip-flop and clears the data flip-flop.

The DS data flip-flop is set by data ones (data flip-flop set) but is synchronized to the window gate.

3.5.16 MFM Encoding and Write Precompensation

The MFM encoding and write precompensation logic (Figure 3-42) performs two major functions:

1. Converting serial digital data (DSRO H) to a modified frequency modulation (MFM) format (MFM DATA L)

2. Preshifting the MFM data pulses to precompensate for magnetic peak shift phenomena (WRITE DATA L/H)

Figure 3-43 illustrates the timing relationship for the MFM encoding and write precompensation logic.

3.5.16.1 MFM Encoding – The RL02 uses a modified frequency modulation (MFM) encoding technique to magnetically record digital data on the disk surfaces. With this technique, each logical one produces a flux reversal in the center of its bit cell. Two successive logical zeros produce a flux reversal at the boundary of each bit cell containing a logical zero following a logical zero. This technique has the advantage of putting at least one flux reversal on the disk for every two bit cells, making it feasible to use phase lock loop techniques to form a self-clocking data recovery system.

During the write (UDISK WRITE H) or maintenance (UMAINT H) functions, the NRZ enable logic allows the DSRO H input from the data shift register to appear as NRZ WRT DATA H.

The write data shift register converts the serial write data (NRZ WRT DATA H) input to parallel write data (WRT DATA ⟨3:0⟩ H) using the 4.1 megahertz L clock. This allows the write data to be viewed as follows:

- WRT DATA3 H (next bit to be written)
- WRT DATA2 H (bit to be written)
- WRT DATA1 H (least significant preceding bit)
- WRT DATA0 H (most significant preceding bit)

The ALLOW 1 logic, using both 4.1 megahertz clocks, monitors the present write data one (WRT DATA2 H) and the previous write data one (ALLOW1 H). If either signal is a one, ALLOW1 H is asserted or remains asserted to the encoder, creating a window for the generation of an MFM-encoded logical one. The LOAD REG L output is used to clock in write data (WRT DATA ⟨3:0⟩ H) for use within the ALLOW 1 and the write early/write late bit comparison logic.

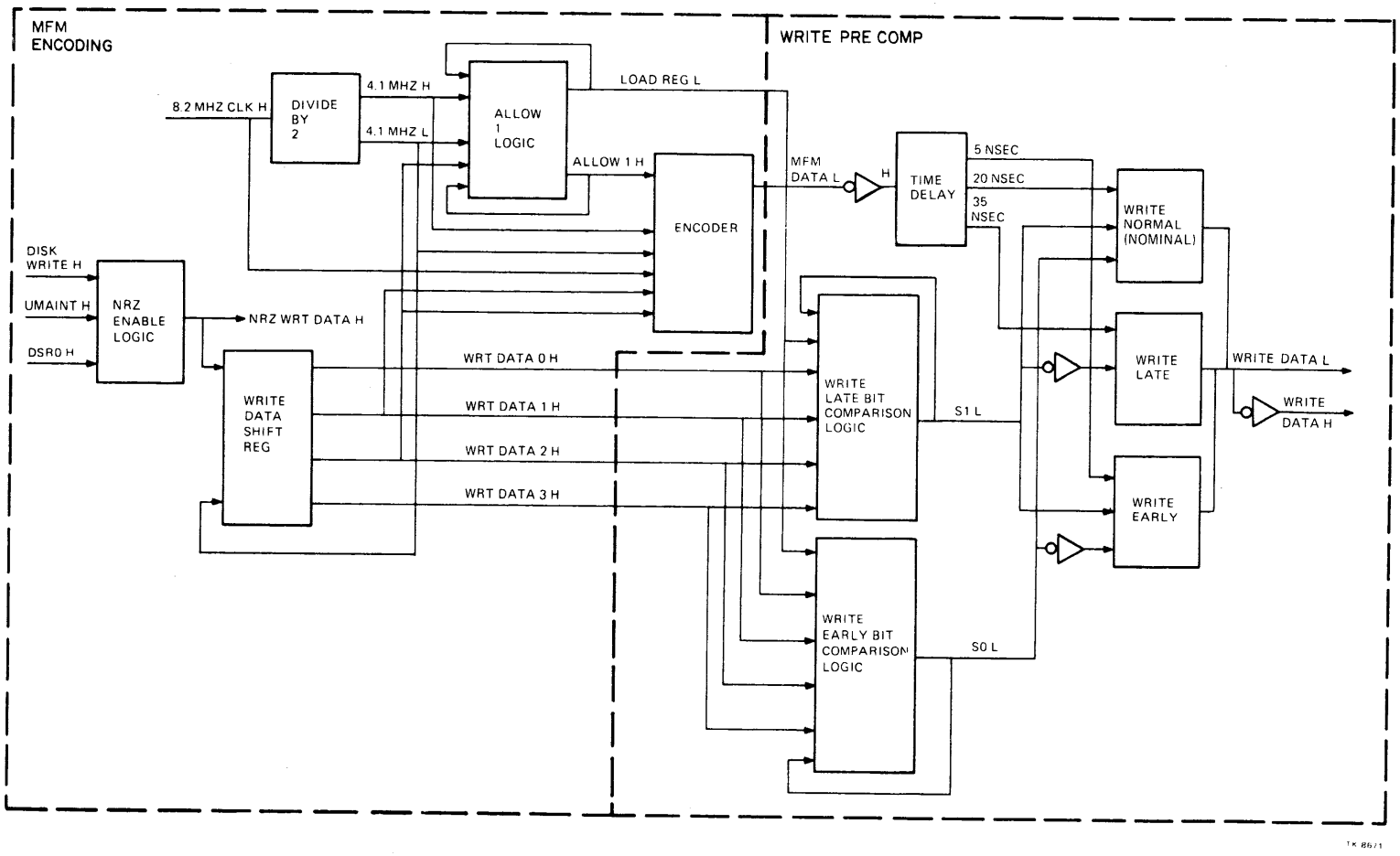


Figure 3-42 MFM Encoding and Write Precompensation Logic Functional Block Diagram

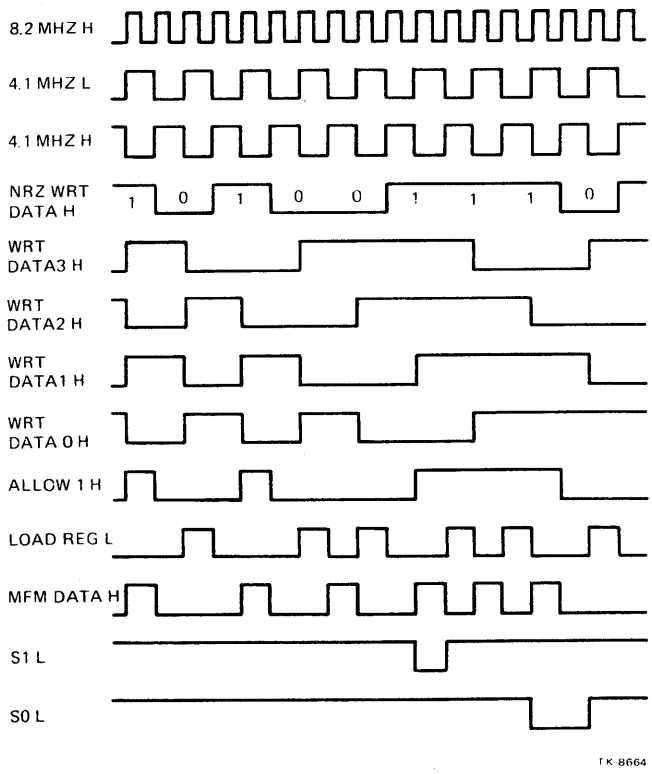


Figure 3-43 MFM Encoding and Write Precompensation Timing Diagram

The encoder monitors the following write data (WRT DATA ⟨2:1⟩ H) conditions:

- Logical one (ALLOW1 H)
- Two successive logical zeros (the complement of WRT DATA ⟨2:1⟩ H)

The encoder, in synchronization with the clock inputs (8.2 megahertz H, 4.1 megahertz L, 4.1 megahertz H), will assert MFM DATA L in the center of its bit cell for each write data logical one (ALLOW H). For two successive write data logical zeros (the complement of WRT DATA ⟨2:1⟩ H), the encoder will assert MFM DATA L at the boundary of each bit cell containing a logical zero following a logical zero.

3.5.16.2 Write Precompensation – One of the problems associated with double density magnetic recording is a phenomenon called peak shift, in which flux reversals written on the disk tend to repel one another. Because of this, the flux reversals appear displaced from where they were written. This can cause pattern sensitive data recovery problems.

The write precompensation logic offsets the harmful effects of peak shift. This logic displaces the MFM-encoded data pulses (MFM DATA L) by 15 nanoseconds in one direction or the other (early or late) before they are written on the disk. This allows the peak shift phenomenon to displace the flux reversal to the desired position.

To determine if an MFM-encoded data pulse is to be displaced from its nominal position (20 nanoseconds), the following rule is used. A pulse is preshifted only if:

- It is bounded on one side by a pulse that is not more than one bit cell away, and
- It is bounded on the other side by a pulse that is more than one bit cell away

The direction (early or late) of the preshift depends on the write data bit combinations that precede (WRT DATA ⟨1:0⟩ H) and/or follow (WRT DATA3 H) the bit to be preshifted (WRT DATA2 H).

The write precompensation logic is concerned with only four write data bit combinations, as shown in Figure 3-44. All other bit combinations preclude the need for preshifting by 15 nanoseconds and, therefore, the nominal position (20 nanoseconds) is used. The write late and write early bit comparison logic monitors the four write data bits (WRT DATA ⟨3:0⟩ H) looking for one of the four bit combinations that require preshifting. After monitoring the bit combinations, the logic causes one of the following to occur:

- Leave the pulse to be written in its nominal position (S1 L, SO L deasserted, and the 20-nanosecond tape asserted)
- Write the pulse 15 nanoseconds early (20 nanoseconds minus 15 nanoseconds) (S1 L deasserted, SO L, and the five-nanosecond tape asserted)
- Write the pulse 15 nanoseconds late (20 nanoseconds plus 15 nanoseconds) (SO L deasserted, S1 L, and 35-nanosecond tape asserted)

Thus, for the write data bit combination 1000 as shown in Figure 3-44, the MFM-encoded data pulse must be preshifted 15 nanoseconds to the right (late) to compensate for a peak shift to the left. For the bit combination 0001, the MFM-encoded data pulse must be preshifted 15 nanoseconds to the left (early) to compensate for a peak shift to the right.

The MFM-encoded and write-precompensated data (WRITE DATA L/H) are made available to the RL02 disk drive.

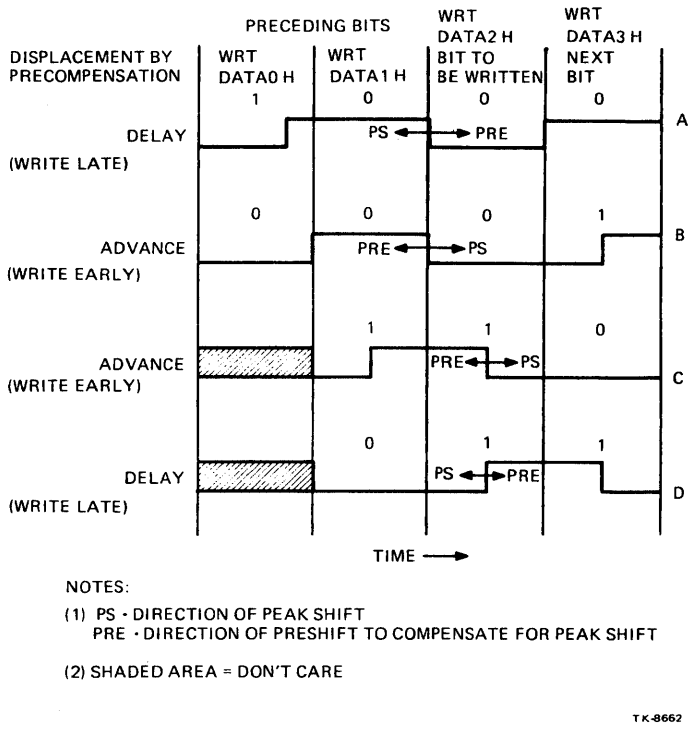


Figure 3-44 Write Precompensation Early/Late Bit Combinations

CHAPTER 4
MAINTAINABILITY FEATURES

4.1 MAINTAINABILITY FEATURES

The circuitry of the IDC contains several features that enhance maintenance. These are:

- Maintenance mode
- Data loopback
- Write inhibit
- Timeout inhibit
- Defeatable enables

4.1.1 Maintenance Mode

When bit 25 of the CSR is set, the IDC is placed in maintenance mode, where it is used to redefine instructions and to allow initiation of diagnostic software.

4.1.2 Data Loopback

When testing the IDC with microdiagnostics, the IDC provides a simulated RL READ DATA input (See Figure 3-1). When a maintenance function is specified by the CPU, the microcontroller selects the RL WRITE DATA output of the MFM encoder as the RL READ DATA input to the IDC read data separator. This allows the capability of performing testing of the IDC using known data configurations.

4.1.3 Write Inhibit and Timeout Inhibit

During microdiagnostic testing of the IDC, the timeout logic is inhibited and writing to the disk drives is inhibited. The timeout inhibits prevent the IDC from terminating diagnostic operations requiring more than 150 milliseconds to perform. The write inhibit enables the IDC to be tested without corrupting the data stored in the associated disk drives.

4.1.4 Defeatable Enables

The enable input to each of the PALs (the GND input that enables the output of the PALs to be asserted) is applied through a resistor. Thus, the enable may be manually defeated by asserting a +3 Vdc level at the enable input of the PAL.

CHAPTER 5
PROGRAM INTERFACE

5.1 BASIC SYSTEM OPERATION

Five basic decisions are made by the CPU when a data transfer occurs:

1. The drive to be used (drive number)
2. Where on the disk the desired data are located (cylinder, sector, track)
3. The direction of data transfer (read or write)
4. Where in memory the data to be read from or written to are found (starting memory address)
5. The amount of data to be transferred (number of words)

The commands generated by the CPU to make these decisions are applied to the selected disk through the IDC.

Up to four RL02 disk drives or one R80 disk drive and up to three RL02 disk drives can be connected to the IDC. However, since all of the drives have the same bus address, the CPU must designate (to the IDC) which drive to select. Once selected, only that particular drive can decode and respond to the operational commands.

Once selected, the drive starts its mechanical positioning after receiving the cylinder, sector, and track values plus a seek command containing a GO bit. The drive notifies the CPU, through the IDC, when the desired position is located. If no error condition exists, the CPU initiates the data transfer sequence.

When the heads are at the correct location and the command is a read operation, serial data are read from the disk, converted to parallel in the IDC, and then transmitted to the CPU. At the completion of a read or write operation, the IDC interrupts the CPU (providing the interrupt enable is set) to indicate that the data transfer is complete.

5.2 PROGRAMMING OVERVIEW

The communication of control commands, status data, error conditions, and maintenance information is accomplished through registers contained in the IDC. The IDC contains eight registers required for drive operation. Table 5-1 lists these registers, their mnemonics, their address, the type of register (read only or read/write), and their basic functions.

5.2.1 IDC Registers

5.2.1.1 Control Status Register (CSR) – The control status register (address F26200) is loaded under CPU control with the IDC control word. The CSR also operates under CPU control to cause the transfer of the IDC status word (the current IDC control word contained in the CSR and a summary of the current status of the IDC and disk drives) from the IDC to the CPU.

Table 5-1 IDC Registers

Register Name	Address	Type	Function
Control Status (CSR)	F26200	R/W	IDC control and status interface.
Bus Address (BAR)	F26204	R/W	Contains the UNIBUS virtual address of first byte to be transferred.
Byte Count (BCR)	F2608	R/W	Contains 2's complement of number of bytes to be transferred.
Disk Address (DAR)	F2620C	R/W	Contains disk cylinder, sector, and track address (head number) where transfer is to occur.
Multipurpose (MPR)	F26210	R/W	RL02 get status.
ECC Position	F26214	R	Contains the starting bit position of a correctable ECC error encountered during an R80 read.
ECC Pattern	F26218	R	Contains the bit (11) pattern to be used to correct the error.
IDC Initialize	F2621C	R/W	When written with a negative one, causes the entire IDC to be initialized.

The CSR asserts the initial branch conditions (F0, F1, and F2) and the start signal (CRDY) to the microcontroller. The CSR also controls selection of the applicable disk drive and enables the appropriate read data paths of the IDC. Status information from the disk drives and from the IDC header/data comparator and ECC/CRC logic is asserted to the CSR, which makes this information available to the CPU in the form of the IDC status word output.

When the function specified by the IDC control word is completed, is waiting for a data transfer to or from the CPU, or has been halted due to an error, the CSR operates from microcontroller inputs to generate and assert the applicable interrupt (UBUS BR5 or PORT XFER REQ) to the CPU.

Table 5-2 provides a description of each bit of the CSR.

Table 5-2 Control Status Register Bit Assignments

Bit Position	Name	Description
00	Drive Ready (DRDY)	Indicates the drive is ready to receive a command. It is cleared during a seek or head select operation and is reasserted when the operation is completed.
03:01	Function (F2:F0)	These bits are set by software to indicate the function to be performed when CRDY is cleared. Cleared by INIT. Commands are as follows:

R80

FMT	F2	F1	F0	Command
0	0	0	0	No drive operation
0	0	0	1	Write check data
0	0	1	0	Get status
0	0	1	1	Seek
0	1	0	0	Read header
0	1	0	1	Write data
0	1	1	0	Read data
0	1	1	1	Read data without header check
1	0	0	0	R80 write format function

Commands are described in detail in Paragraph 5.3.

05:04	Not used	When set, the CPU is interrupted at the normal or error termination of any function.
06	Interrupt Enable (IE)	Any asynchronous condition, such as a drive coming on line or completing a seek or asserting error, causes an interrupt due to the setting of the attention flop associated with that drive. This bit is set and cleared by the software writing the register. It is cleared by INIT.
07	Controller (IDC) Ready (CRDY)	This bit is cleared by the software to indicate that the function contained in bits ⟨03:01⟩ is to be performed. It is set by the IDC at the completion of the requested function, at the detection of an error, or by INIT.
09:08	Drive Select (DS1:DS0)	Selects one of four drives (3 through 0) connected to the controller. Cleared by INIT.

Table 5-2 Control Status Register Bit Assignments (Cont)

Bit Position	Name	Description
10	Operation Incomplete (OPI)	When set, indicates that the function did not complete within the OPI timeout or that a function was stopped because of a header CRC or skipped sector error.
11	Data Check Error (DCK)	If OPI is cleared, the ECC error occurred on the data (DCK). If OPI is set, the error occurred on the header (HCRC).
12	Data Late (DLT)	Indicates on a write, if OPI is cleared, that the CPU did not respond in time with accepting read data or passing write data. When OPI is set, the same bit indicates header not found (HNF). This indicates that the OPI timeout occurred while the IDC was searching for the correct sector to read, write, or write check.
13	Nonexistent Memory (NXM)	Indicates that the IDC was unable to access the memory address shown in the BAR. OPI and DLT are usually set when this error occurs.
		NOTES **1. In bits 13:10, the CRC check is performed on both header words, even through the second header word on the RL02 is always 0.** **2. Bits 13:10, if caused by DRIVE ERROR, are cleared by INIT or by initiating a function. CRDY is set by INIT.**
14	Drive Error (DE)	Indicates that the selected drive has flagged an error. The source of the error can be determined by performing a get status function. This error can be cleared for the RL02 by the RST bit during a get status function. DE will not cause ERR and CRDY until the normal occurrence of CRDY.
15	Composite Error (ERR)	Indicates that at least one of the error bits 14:10 is set. When ERR is set, an operation will terminate and interrupt if IE is set.

Table 5-2 Control Status Register Bit Assignments (Cont)

Bit Position	Name	Description
19:16	Attention (ATTN3:ATTN0)	An attention bit is provided for each drive to signal that a seek has been completed or that the drive has changed status. A status change is defined as asserting Drive Ready and removing Drive Ready while not doing a seek. These changes in drive status are scanned by the IDC whenever it is not occupied with performing a function. These bits are cleared by writing a one to the appropriate attention bit.
21:20	R80 ECC Status (ECS1:ECS0)	These two status lines are encoded as follows: 00 – No Errors. This is the initial state of the status lines. The 00 state is maintained unless a read data error is encountered. 01 – Data Error. The 01 state is entered following the check field of a read operation if the data is nonzero. This bit indicates that the correction determination is in progress. 11 – Correctable Error. The 11 state is entered at the completion of the correction computation if the computation is successful. 10 – Hard Error. If the correction computation operation is not successful, the 10 state is entered. **NOTE** **The 01 state indicates that an error has occurred (ECC or CRC mode) and that a correction computation is in progress (ECC mode). STAT 1 serves as a "correction computation complete" signal.**
22	R80 Skip Sector Inhibit (SSEI)	This is a read/write bit used to inhibit skip sector errors. When written as a 1, skip sector errors are inhibited from occurring until the bit is cleared. This bit can be cleared by writing it to a zero or by INIT. It should not be used when in automatic skip sector mode (bit 27 cleared).
23	R80 Skip Sector Error (SSE)	This bit can be read or written and can be set to either a 0 or 1. It is set when bit 13 of the R80 header is read as a one, indicating that the sector being read is a displaced sector because it or a previous sector contained a bad spot. This error can be cleared by writing a 0 in the bit position or by INIT. This bit should not be used when in automatic skip sector mode (bit 27 cleared).

Table 5-2 Control Status Register Bit Assignments (Cont)

Bit Position	Name	Description
24	Interrupt Request (IR)	Indicates that it is the IDC that has asserted BR5 and is requesting an interrupt. This bit can be cleared by writing a one to bit 24.
25	Maintenance (MTN)	Places the IDC in maintenence mode, where it is used to redefine instructions and allow initiation of diagnostic software. It can be read and written to a 1 or 0.
26	R80	This bit is asserted when DS 1:0 has selected the R80 disk drive.
27	Automatic Skip Sector Inhibit (ASSI)	When this bit is cleared, the IDC automatically handles skip sector errors. During this state, CSR bits 22 and 23 are undefined and should not be altered by software. Setting this bit disables the automatic handling of skip sector errors. Bits 22 and 23 assume the meanings just described and are used to control SSEs in software.
28	Time Out Inhibit (TOI)	When set, this bit disables the IDC on-board timeout clock. This bit is primarily used by microdiagnostics.
29	R80 Format (R80 FMT)	This bit, in combination with a function code of zero, selects the R80 format function to be performed after clearing CRDY.
30	Not used	
31	Mask Attention (MASK ATTN)	When set, any writes to the CSR are masked, so as not to clear the attention bits.

5.2.1.2 Bus Address Register (BAR) – The bus address register (address F26204) is loaded with the UNIBUS virtual address of the first byte to be transferred. Bits (31:18) are ignored.

5.2.1.3 Byte Count Register (BCR) – The byte count register (address F26208 is loaded with the two's complement of the number of bytes to be transferred.

5.2.1.4 Disk Address Register (DAR) –The disk address register (address F2620C) is loaded under CPU control with the required disk drive control word or read/write data address. The read/write data address of the disk address register may be incremented by the microcontroller to update the read/write data address information as additional contiguous sectors of data are written or read. The contents of the disk address register may be transferred from the IDC to the CPU under CPU control.

The format of this 32-bit register is shown in Figure 5-1.

The DAR must be loaded immediately before seek or data transfer commands. Since the R80 and RL02 have different geometries, the drive to be commanded must be selected before loading this register.

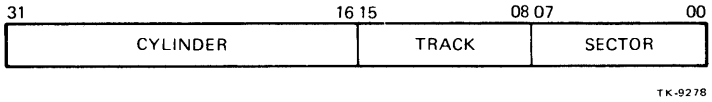


Figure 5-1 Disk Address Register

5.2.1.5 Multipurpose Register (MPR) – The multipurpose register (address F26210), when used with the RL02 get status function, is loaded with a get status command. The RL02 drive status word is obtained by loading the MPR with a get status command and then loading the CSR with a get status function. The IDC must be ready (CRDY) before loading the MPR. With the R80, get status is initiated by simply loading the CSR with the get status function.

Table 5-3 provides a description of each bit of the MPR.

Table 5-3 MPR Bit Assignments

Bit Position	Name	Description
0	Marker (M)	Used by the drive to tell when a new serial command word has arrived. Must be a 1.
1	Get Status (GS)	Must be a 2, indicating to the drive that the status word is being requested. At the completion of a get status command, the drive status word can be read from the MPR. With this bit set, bits 15:8 are ignored by the drive.
2	Not used	
3	Reset (RST)	If set, the RL02 drive clears its error register before sending the status.
15:4	Not used	

5.2.1.6 ECC Position Register – The ECC position register (address F26214) is a 13-bit register that indicates the starting bit position of a correctable ECC error encountered during an R80 read function.

5.2.1.7 ECC Pattern Register – The ECC pattern register (address F26218) is an 11-bit register indicating the bits to be used to correct the error. It is valid only during an R80 read that contains a correctable ECC error.

5.2.1.8 IDC Initialization Register – When written with a negative one (-1), the IDC initialization register (address F2621C) will cause the entire IDC to be initialized.

5.3 COMMANDS

Program operations are initiated by the combination of the actions listed below:

- Selecting a drive
- Loading the CSR with a function code
- Setting the GO bit (CRDY)

The function code identifies a specific command. On assertion of the GO bit (CRDY), the drive proceeds to execute the command. The commands can be divided into three categories:

1. Positioning
2. Data transfer
3. Housekeeping

These commands and their corresponding function codes are described in the following paragraphs.

5.3.1 Positioning Commands

Positioning commands direct the logic that controls the amount of mechanical movement required to position the heads over the recording media. These commands assert the ATTN bit after their normal completion.

5.3.1.1 Seek Function (F2:F0 = 3) – A seek is initiated to a drive by selecting it via the CSR, loading the DAR with the desired disk address, and issuing a seek command.

RL02 Seek – When a seek command is encountered, the IDC will set CRDY as soon as the drive receives the command and interrupt if IE is set. On receiving the cylinder address, the RL02 drive will seek and/or select a new head as indicated. Other combinations of DAR (0,1) will cause undefined results.

If a cylinder address is provided that attempts to move the head past the innermost (track 511) or outermost (track 0) limits, the head will come to rest on either track 0 or 510.

If software discovers that a seek was unsuccessful or that the RL02 is not selecting the proper cylinder, the execution of a read header command followed by a seek to the desired cylinder will resynchronize the IDC to the proper cylinder.

R80 Seek – The DAR must be loaded prior to the start of the seek function. The clearing of CRDY will then initiate the desired operation. As soon as the transfer is complete, CRDY will be set and the IDC will interrupt if IE is set.

NOTE

When −1 is written to the DAR, the microcode will issue a recalibrate command R80. This command positions the heads over cylinder 0.

5.3.2 Data Transfer Commands

Data transfer commands involve the transfer of data to or from the disk. This also includes the transfer of status information.

5.3.2.1 Read Header Function (F2:F0 = 4) – When a read header function is decoded with CRDY cleared, the IDC will read and buffer in the first header encountered on the selected drive. The IDC will set CRDY and interrupt if IE is set. Software can then read the two header words and the CRC word from the MPR with three successive reads to determine the current cylinder, head, or sector location of the drive.

5.3.2.2 Write Data Function (F2:F0 = 5) – When the write data function is decoded with CRDY cleared, the IDC begins reading successive header words and compares them to the DAR. When a match is found, the header CRC is checked and, if correct, the sector is written with the words provided by the CPU. For partial sector writes, the remamining sector area is filled with 0s. At the end of the sector, the DAR is incremented and the next sector is written if there is count remaining. At the end of the transfer, CRDY is set and an interrupt made if IE is set.

5.3.2.3 Read Data Function (F2:F0 = 6) – When this function is decoded with CRDY cleared, the IDC begins reading successive header words and comparing them to the DAR. When a match is found, the header CRC is checked and, if correct, that sector is read and the words requested are buffered for transfer to memory by the CPU. Data ECC (or CRC for RL02) is then checked and the DAR incremented. If the desired number of words has not been transferred, the next sector is read. Otherwise, CRDY is set and an interrupt made if IE is set.

5.3.2.4 Read Data Without Header Check Function (F2:F0 = 7) – When this function is decoded with CRDY cleared, the data portion of the sector following the next sector pulse is read and the words requested are buffered for transfer to memory by the CPU. The header is neither compared nor checked for CRC errors. Data ECC (or CRC for RL02) is checked at the end of a sector. If the desired number of words has not been transferred, the next sector is read. Otherwise, CRDY is set and an interrupt made if IE is set.

5.3.2.5 Write Check Function (F2:F0 = 1) – This function is used to perform a bit-by-bit comparison between data in main memory and data on the disk. When the function is decoded with CRDY cleared, the IDC starts reading headers and compares them to the DAR. When a match is found, the header CRC is checked and, if correct, that sector is read and the data are compared in the controller with data that has been fetched from main memory by the CPU. At the end of a sector, if a data comparison error or a data CRC/ECC error has been sensed, the Data Check (DCK) error bit will be set in the CSR.

5.3.2.6 Get Status Function (F2:F0 = 2)

RL02 Get Status – If the Get Status bit (bit 1 in the MPR) is set, the RL02 drive will send its status word via the status line to the IDC. When the drive status word is received, the IDC will set CRDY and interrupt if IE is set. The CPU can then read the RL02 status word by reading the MPR. This function may be performed any time the controller is ready, even though the drive is not (during a seek or when in the load state).

The operation is undefined if the Get Status bit is a 0. If bit 3 in the MPR is set, the drive will attempt to clear its error bits before sending the status word.

The contents of the RL02 status word are listed in Table 5-4.

Table 5-4 RL02 Get Status

Bit Position	Name	Description
2:0	State (ST)(A, B, or C)	These bits define the state of the drive: C B A 0 0 0 Load State 0 0 1 Spin-Up 0 1 0 Brush Cycle 0 1 1 Load Heads 1 0 0 Seek Track Counting 1 0 1 Seek Linear Mode 1 1 0 Up Load Heads 1 1 1 Spin Down
3	Brush Home (BH)	Asserted when the brushes are not over the disk.
4	Heads Out (HO)	Asserted when the heads are over the disk.
5	Cover Open (CD)	Asserted when the cover is open or the dust cover is not in place.
6	Head Select (HS)	Indicates the currently selected head.
7	Reserved	
8	Drive Select Error (DSE)	Indicates multiple drive selection is detected.
9	Volume Check (VC)	Indicates the transition from a head load state to a head on track state.
10	Write Gate Error (WGE)	Indicates the drive sensed Write Gate asserted when the sector pulse is asserted.
11	Spindle Error (SPD)	Indicates the spindle is not reaching speed in the required time, or is overspeeding.
12	Seek Timeout (SKTO)	Indicates the heads did not come on track in the required time during a seek command.
13	Write Lock (WL)	Indicates write lock status of the selected drive.
14	Head Current Error (HCE)	Indicates write current was detected in the heads when write gate was not asserted.
15	Write Data Error (WDE)	Indicates Write Gate was asserted, but no transitions were detected on the write data line.

R80 Get Status – The R80 sends its status word to the IDC in parallel via the interface lines. When the drive status word is ready, the IDC will set CRDY and interrupt if IE is set. The CPU can then read the R80 status word by reading the MPR. This function may be performed any time the IDC is ready, even though the drive is not (during a seek or when in the load state).

The contents of the R80 status word are listed in Table 5-5.

Table 5-5 R80 Get Status

Bit Position	Name	Description
4:0	Sector Count (SEC4:0)	Sector count will change on leading edge of sector or index. Timing integrity is maintained throughout seek operation.
7:5	Not used	
8	Fault (FLT)	Signals fault condition. The following types of faults are detected: dc power fault, head select fault, write fault, write or read while off cylinder, and write gate during a read operation. May be cleared by INIT or FAULT CLEAR on the operator panel.
9	Plug Valid (PLGV)	Bit indicates that a logic plug is installed in the operator panel.
10	Seek Error (SKE)	Indicates that R80 was unable to complete a seek within 500 milliseconds, that the carriage has moved to a position outside the recording field, or that an illegal address has been detected.
11	On Cylinder (ONCY)	Indicates that the servo has positioned the heads over a track. The status is cleared by any seek instruction causing carriage movement or zero track seek.
12	Drive Ready (DRDY)	Indicates that the unit is up to speed, the heads are loaded, and a no fault condition exists within the drive.
13	Write Protect (WTP)	Signals that the write protect switch has been enabled. Attempting to write at this time causes a fault to be issued.
15:14	Not used	

5.3.3 Housekeeping Commands

Housekeeping commands are used to place the drive logic into a known or initial state. ATTN is not raised at the completion of the housekeeping command unless there is a persistent error condition.

5.3.3.1 NOP Function (F2:F0 = 0)

This function is a NOP, except in the case of the R80 bit being set, and an R80 selected by the CSR. In this case an R80 Format function is performed.

5.4 R80 ECC HANDLING

The IDC has the ability to detect errors that occurred while reading the data field and to provide information for software to recover the data. The ECC code that will be used, called burst error correcting fire code, will locate an error that falls within an error burst of length 11 bits or less. Any errors outside the specified burst length are guaranteed to be detected, but not to be correctable.

The IDC logic will do the following:

- Find the 11-bit burst within which the read error is included.
- Determine the exact location of the burst within the data field.

This information will be provided to the software via the following two IDC registers.

- ECC pattern register: will contain the actual error burst
- ECC position register: will contain the address of the first bit of the error burst within the data field

5.5 HARDWARE ERROR RECOVERY

If an ECC error is detected, the IDC will simultaneously clock the ECC shift register and increment the position counter. When the counter overflows, the correction computation enters a second phase searching for a correctable error pattern. This is done by clocking the shift register bits (20:0) and simultaneously keeping a count of the number of shifts in the position counter. When an all zero condition is found, shifting and counting stop and ECC STAT (1:0) is set to 11, which indicates a correctable error pattern has been found. The error pattern and the error position can then be read via the specific registers.

There is one condition under which a correctable error pattern cannot be computed: an all zero condition is not found within n shifts, where n equals the number of data bits plus check bits: 4096 + 32 = 4128. Under this condition, STAT (1:0) is set to 10, indicating a hard error.

5.6 SOFTWARE ERROR CORRECTION

Error correction is accomplished by the software as follows (not necessarily in this order):

- Software reads the position register.
- Software counts from the beginning of its data field the number of bits as specified by the position register and extracts 11-bits, which represents the burst within which the error occurred.
- Software reads the pattern register.
- Software performs a logical "exclusive OR" of the 11-bit error burst with the contents of the pattern register. The result is the corrected 11-bit data burst, which is now put back into storage.

5.7 R80 SKIP SECTOR OPERATION

5.7.1 R80 Bad Spot Problem

The advent of the 3350 type high capacity drives has caused an increase in the number of bad spots appearing on the media. It is projected that the R80 could have up to 350 bad spots per head disk assembly (HDA). DEC STD 144, which is currently the only specification covering bad spots on disk drives, falls short of handling this large a number of defects. The R80 thus uses a skip sectoring approach that presents bad block information to the driver level software.

5.7.2 The Concept of Skip Sectoring

On each track of the disk, one sector is reserved that will be used as a replacement sector in the event of a bad spot on the track. This replacement is done by sliding each sector down by one, starting at the bad spot, such that the last sector at the end of the track is now the reserved sector. If more than one error occurs on a track, the second bad spot will be logged in the bad block file described in DEC STD 144.

The IDC automatically handles skip sector errors and continues the data transfer if the Inhibit bit (bit 27) in the CSR is cleared. Following is a description of software handling if the Inhibit bit is set.

5.7.2.1 Software Handling of Skip Sector Errors – The responsibility of the IDC is to notify the software that it is trying to read a sector that has been displaced. The responsibility of the software is to restart the transfer at the next sector (n + 1).

5.7.3 Skip Sectoring (with Automatic Inhibit Bit Set) – When the IDC driver receives a request for data, a logical block number and extent is supplied to determine where the transfer will take place on the disk. With skip sectoring, the transfer is initiated as usual by converting the block number to a physical address, loading the word count and address, and initiating the transfer. If no errors occur or an error other than an SSE occurs, the transfer is handled as in the past. If an SSE occurs, as indicated by a 1 in bit 23 of the CSR (MSB of byte 2), it indicates that a sector has been encountered that is physically displaced by one on the disk. This error could occur immediately at the beginning of a transfer, if it started after a bad spot on a track, or in the middle of a transfer if the operation was started before a bad spot and continued beyond it.

The software must first set SSE Inhibit (bit 22) of the CSR. This inhibits further generation of SSEs and allows the operation to continue without further interrupts from SSEs for the rest of the track. The software must then restart the operation. When the operation was aborted in the IDC, because 13 was set in the header, the disk address was incremented by 1. This is exactly where we want to start the operation again when skip sectoring occurs, so no modification of the disk address is necessary. Also, since the IDC aborted the operation as soon as the SSE bit in the header was detected, no data from the sector that generated the error was transferred. This means that the word count and address for the rest of the transfer are correct. So, to restart the transfer, all that is necessary is to set the GO bit (clear CRDY).

5.8 R80 FORMATTING

Provisions have been made within the IDC to format the R80. The following procedure is required to format the disk, one track at a time.

a. Select cylinder and head.

b. Set up registers as in a write data function, supplying four bytes of header for each of the 32 sectors on a track (128 bytes).

c. Initiate the write format function.

The IDC will:

a. Search for the leading edge of the index pulse (sector 0).

b. Immediately bring up write gate and start writing zeros.

c. Write all zeros for head scatter and PLO sync areas (27 bytes).

d. Write a sync pattern, four header bytes, and check word.

e. Write all zeros for write splice gap and PLO sync field (12 bytes).

f. Write a sync pattern, the data field, the four-byte data ECC word, and a two-byte pad at the end of the check word.

g. Wait for the leading edge of the next sector pulse and repeat steps a through f.

h. Continue until the index pulse is detected once again.

i. Set CRDY, interrupt, and return to idle.

5.9 EXAMPLES OF SYSTEM OPERATION

5.9.1 Seek Operation

The following is an example of the sequence involved in a seek function.

a. Select drive and function.

b. Load DAR with desired cylinder, track, and sector.

c. Issue seek function to drive and wait for interrupt. Seek will cause two interrupts, one when the seek has been issued to the drive (CRDY sets) and one when the seek completes. The drive doing the seek asserts DRDY when the seek is complete. The controller, when it is not busy performing functions, checks all drives that have been issued seeks to see if they have asserted DRDY. Respective attention flops are set by the microsequencer for those drives that have done so.

d. Check error flag to complete the seek operation.

NOTE

Since the controller becomes ready and interrupts as soon as a seek is issued, it is possible to issue seeks to additional drives while the first is seeking. An attention interrupt is provided for each drive as each drive completes its seek. The software must know which drives are doing seeks so that it will know why the Attention bit has been asserted.

5.9.2 Data Transfer Operation (Read/Write)

When the seek is completed, the CPU can issue a data transfer command. One drive can be doing a seek at the same time a data transfer command is issued to another drive. Once a data transfer has started, no further commands can be issued to a controller until the transfer is completed either normally or by error.

The read data operation is as follows:

a. Select drive and function. Load byte count, bus address, DAR, and issue read function via CSR.

b. DAR is compared to disk headers until a match is found.

c. The CPU will transfer data into memory using the BAR as a UNIBUS virtual address.

d. The controller will interrupt when the transfer is completed. Software will check error flag in CSR.

e. Select drive and function. Load BCR, BAR, and DAR and issue write function via the CSR.

f. DAR is compared to disk headers until a match is found.

g. The CPU will tansfer data from memory to the drive.

h. The controller will interrupt when the transfer is completed. Software will check error flag in CSR.

APPENDIX
PROGRAMMED ARRAY LOGIC DEVICES (PALS)

A.1 INTRODUCTION TO PALS

Programmed array logic devices (PALs) are logic arrays that may be programmed to give a custom-designed chip unique to a specific requirement.

The basic logic configuration used in PALs is shown in Figure A-1. The circuitry consists of a programmable AND array connected to a fixed OR array. Note that the AND array shown in the basic logic configuration has only four programmable (fusable link) inputs and two fixed OR outputs. In the PAL circuits used in the VAX-11/730, up to 32 programmable AND inputs and up to eight fixed OR inputs are used per output.

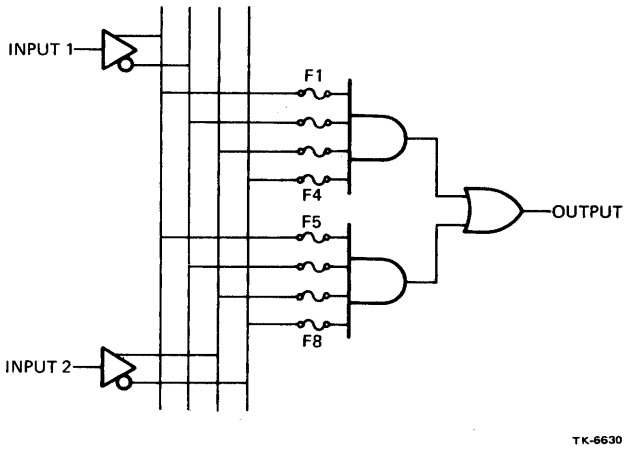


Figure A-1 Basic PAL Logic Configuration

An unprogrammed PAL has all fuses intact, as indicated in Figure A-1. A PAL is programmed by "blowing" the links for the unused AND inputs to give the desired AND before OR logic configuration. For example, the top half of Figure A-2 shows the links "blown" to implement the XOR function $A\overline{B}$ V $\overline{A}B$ in the basic PAL logic configuration. This same logic function may also be represented as shown in the bottom half of Figure A-2 where an X represents the links that are left intact to perform the logical AND. This last method of showing an AND array configuration is the convention used in the PAL plot listings provided in the VAX-11/730 microfiche set.

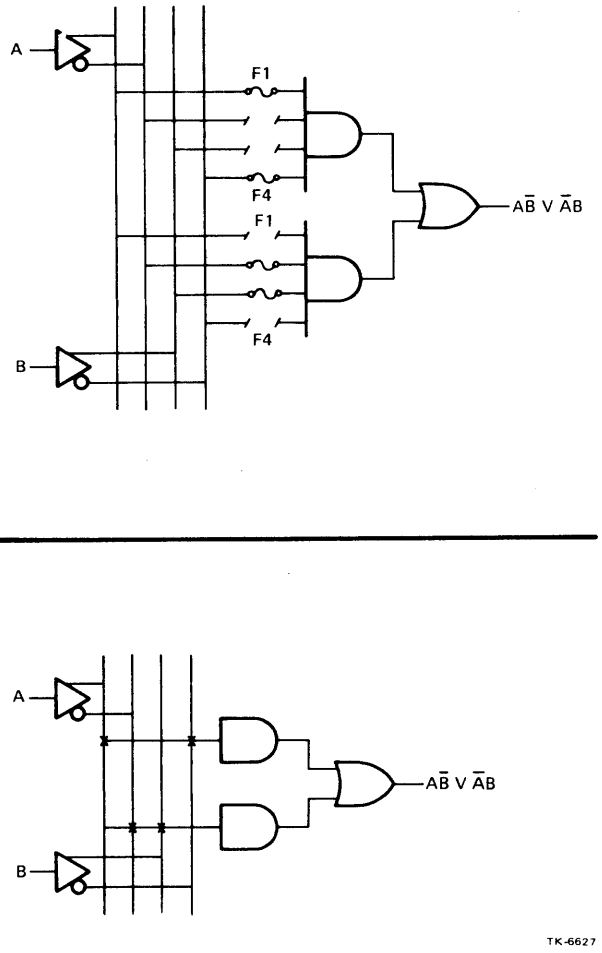


Figure A-2 XOR Logic Function Using PAL Logic

A.2 PAL DEVICE TYPES

The four types of PALs used in the VAX-11/730 are listed in Table A-1. Logic diagrams for each PAL are given at the end of this appendix.

It can be seen from the logic diagrams that the four PAL devices all use the basic AND before OR logic configuration discussed in Paragraph A.1. However, outputs from the l6L8 are inverted and six of the eight outputs feed back to the AND arrays for added functionality. In addition, the output inverters for these six outputs may be turned on and off by the AND arrays (programmable I/0). This provides added logic capability. It also allows the corresponding output pin to be used (when the inverter is turned off) as an input to the AND array just like a designated input pin.

Also note from the logic diagrams that the 16R8 has register outputs (D-type flip-flops) and no gate outputs. Again, outputs are fed back to the AND array but not directly by way of the output pins. Instead, the 0 outputs of the flip-flop drive the array. As a result, the output pins cannot be used as input pins as for a 16L8. The other two PAL types, the 16R6 and 16R4, have varying combinations of both gate and register outputs on the same chip.

Table A-1 PAL Device Types Used in the VAX-11/730

PAL Device Type	Inputs	Outputs	Program IO	Register Outputs	Description
16L8	16	8	6	0	AND-OR gate array
16L8	16	8	0	8	AND-OR array with registers
16R6	8	8	2	6	AND-OR array with registers
16R4	8	8	4	4	AND-OR array with registers

A.3 PAL SYMBOLOGY

A typical PAL as represented in the VAX-11/730 Engineering Print Set is shown in Figure A-3. Information within the symbol includes the device type, part number, and location. For example, the PAL in Figure A-3 is a 16R4 located at E50 with a part number equal to 010K3. The PAL part number distinguishes one programmed PAL from another.

Inputs to the designated PAL input pins are shown at the left of the PAL symbol. Outputs appear at the right. When an output pin is used as an input pin as discussed in Paragraph A.2, the input signal is entered at the left of the symbol and a dotted line (drawn across the PAL symbol) is used to show the connection to the output pin on the right. Pins having both input and output capability are labeled as I/O on the PAL symbol. Gate outputs not having both input/output capability are labeled with an O. Register outputs are identified by an R. Finally, designated input pins are specified by a D.

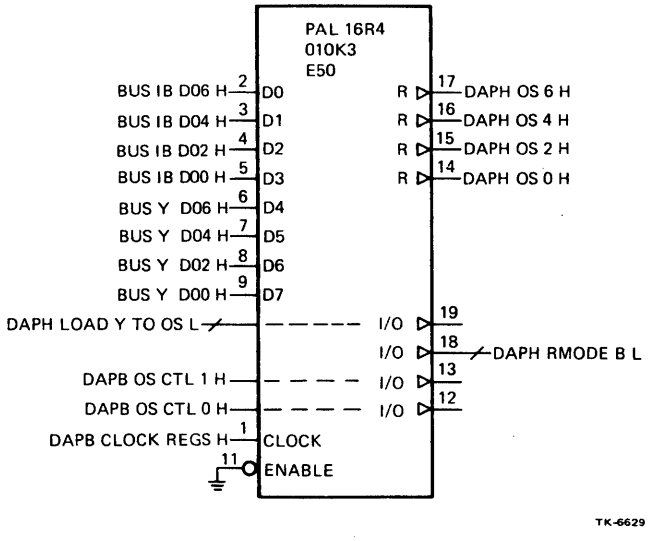


Figure A-3 Typical PAL Symbology

A.4 READING THE PAL PLOT LISTING

An example of a PAL plot listing is shown in Figure A-4. The part number and PAL device type (a 16R6 in this case) are at the top of the listing. The input or output associated with each PAL pin is given next. (NC indicates no connection; VCC indicates the +5 volt power source.) A low assertion level for input/output signals on the listing is indicated by a slash (/) immediately preceding the signal name. If there is no slash, the signal is asserted high. Input/output signal names on the listing are sometimes abbreviated and may not be exactly the same as in the Engineering Print Set.

The rest of the listing consists of the AND array plots for each output pin. An X represents the fusable links left intact; a dash (−) represents a "blown" link. To the right of each line in a plot is the list of signals selected by the intact links that make up the AND inputs. Because these individual AND terms are ORed by the PAL logic, the list of AND terms in the listing (ORed together) result in an easily read Boolean expression that represents the logic function performed. For example, output pin 12, which is a gate output (refer to the 16R6 logic diagram) and the last plot in the listing, has the following input:

```
VCC
START_8085_CYC*IO*A14*/RAS
/RAS*STATE
```

The underlines in the expression above only represent a space (a blank character) in the signal name. An asterisk (*) between signal names specifies the logical AND operation. Discounting the enable level for the output inverter, which in this case is always asserted, this input expression for output pin 12 (/UART_CHIP_SEL) may be read as follows:

```
UART CHIP SEL L = START 8085 CYC H IO H A14 H RAS H
                  RAS H V STATE H
```

For a register output, the Boolean expression read from the listing specifies the output signal just as for a gate output. The output pin is not asserted or negated until the register flip-flop is clocked. Flip-flops are clocked by the positive-going transition of the clock.

A.5 PAL LOGIC DIAGRAMS

The logic diagrams for the 16L8, 16R4, 16R6, and 16R8 PAL devices are shown in Figures A-5 through A-8.

```
PART NUMBER: 23-004K4-0-0

DEVICE TYPE: PAL16R6

PIN NUMBER = SYMBOL TABLE:

1= CLOCK                 8= SEL_9600_BAUD      15= STATE
2= ALE                   9= RESET              16=/RAS
3= REQUEST_REFR         10= GROUND             17= REFRESH_DONE
4= IO                   11= OUT_EN             18=/START_8085_CYC
5= A14                  12=/UART_CHIP_SEL      19=/LONG_CYCLE
6= 9600_BAUD            13=/9600_300_BAUD      20= VCC
7= 300_BAUD             14= REFRESH_CYC


FUSE PLOT:          (X = FUSE INTACT , - = FUSE BLOWN)

OUTPIN 19  ---- ---- ---- ---- ---- ---- ---- ---- VCC
           ---- ---X ---- X--- ---- ---- ---- ---- START_8085_CYC*A14
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 18  X--- ---- ---- ---- ---- ---- ---- ---- ALE
           ---- ---X ---- X--- ---- --X- ---- ---- REFRESH_CYC*START_8085_CYC*A14
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 17  ---- -X-- ---- ---- ---- ---- ---- ---- /REQUEST_REFR
           ---- ---- ---X ---- ---- ---X ---- ---- /REFRESH_DONE*/REFRESH_CYC
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 16  ---- ---- ---- --X- ---- --X- ---- ---- /RAS*REFRESH_CYC
           ---- ---- ---- ---X --X- ---- ---- ---- RAS*STATE
           ---- ---X -X-- X-X- ---- ---- ---- ---- START_8085_CYC*/RAS*/IO*A14
           ---- ---- ---- ---- ---- ---- ---- X--- RESET
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
           XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 15  ---- --X- ---- ---- ---- ---- ---- ---- /START_8085_CYC
```

Figure A-4 PAL Plot Listing (Sheet 1 of 2)

```
          ---- ---- ---- ---X ---- ---- ---- ---- RAS
          ---- ---- ---- -X-- ---- ---- ---- ---- /A14
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 14 ---- ---X ---- ---- ---- ---- ---- ---- START_8085_CYC
          ---- ---- ---- ---X ---- --X- ---- ---- RAS*REFRESH_CYC
          ---- ---- ---- ---X --X- ---- ---- ---- RAS*STATE
          ---- -X-- ---- ---- ---- ---X ---- ---- /REFRESH_CYC*/REQUEST_REFR
          ---- ---- --X- ---- ---- ---X ---- ---- /REFRESH_CYC*REFRESH_DONE
          X--- ---- ---- --X- ---X ---X ---- ---- /REFRESH_CYC*/RAS*ALE*/STATE
          ---- ---- ---- ---- ---- ---- ---- X--- RESET
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 13 ---- ---- ---- ---- X--- ---- X--- ---- SEL_9600_BAUD*9600_BAUD
          ---- ---- ---- ---- ---- X--- -X-- ---- /SEL_9600_BAUD*300_BAUD
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 12 ---- ---- ---- ---- ---- ---- ---- ---- VCC
          ---- ---X X--- X-X- ---- ---- ---- ---- START_8085_CYC*IO*A14*/RAS
          ---- ---- ---- --X- --X- ---- ---- ---- /RAS*STATE
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

Figure A-4 PAL Plot Listing (Sheet 2 of 2)

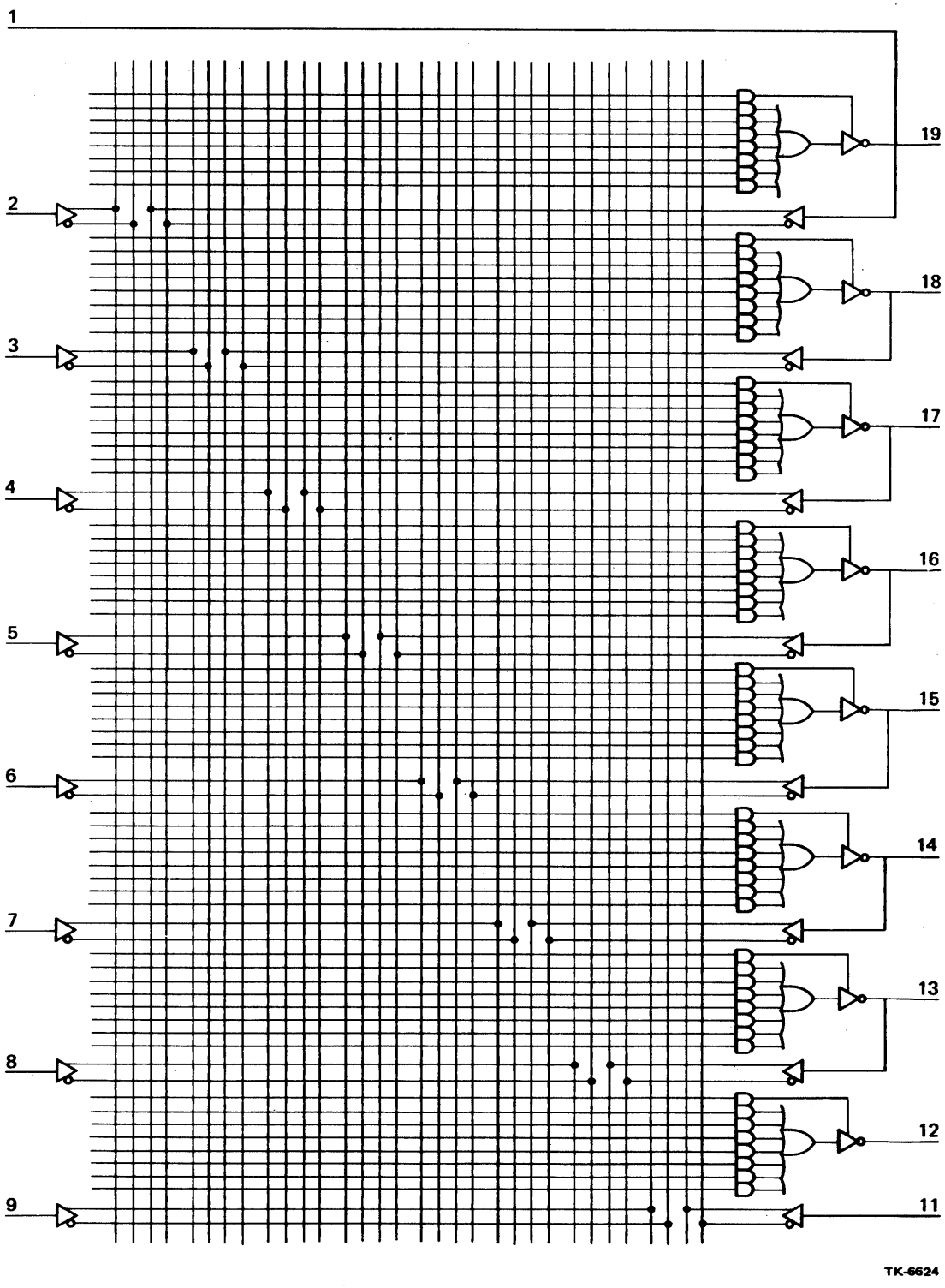


Figure A-5 Logic Diagram: PAL16L8

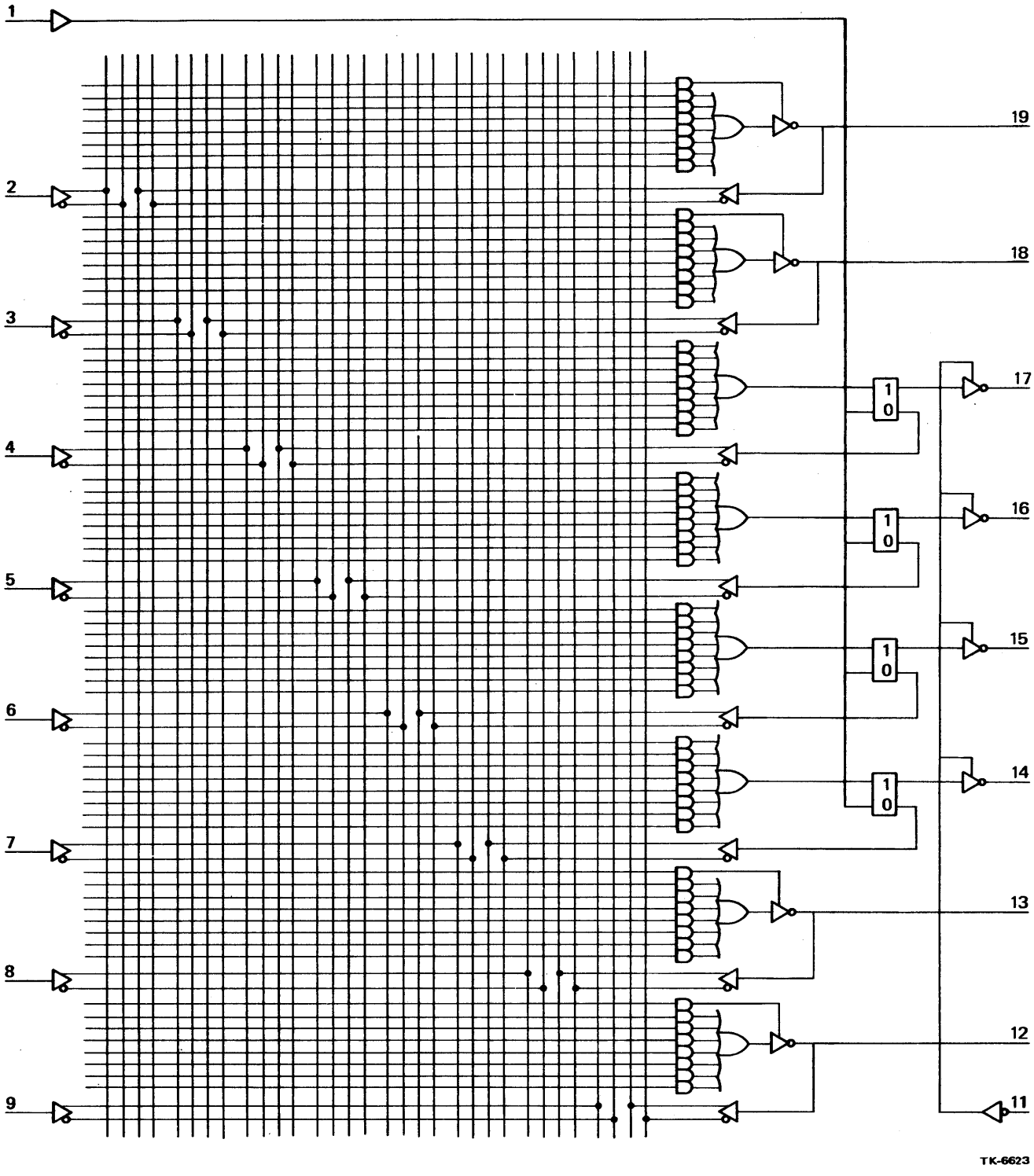


Figure A-6 Logic Diagram: PAL16R4

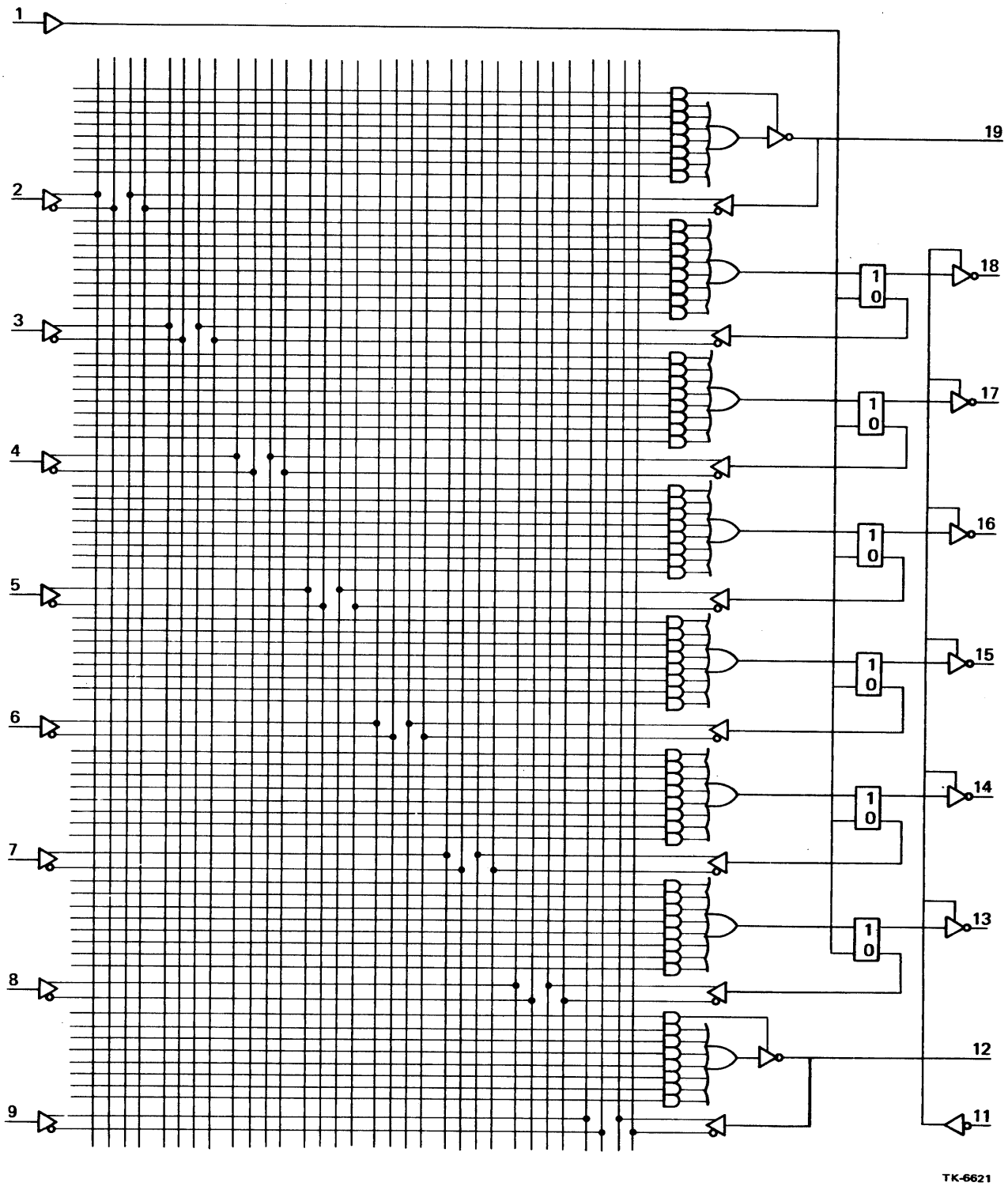


Figure A-7 Logic Diagram: PAL16R6

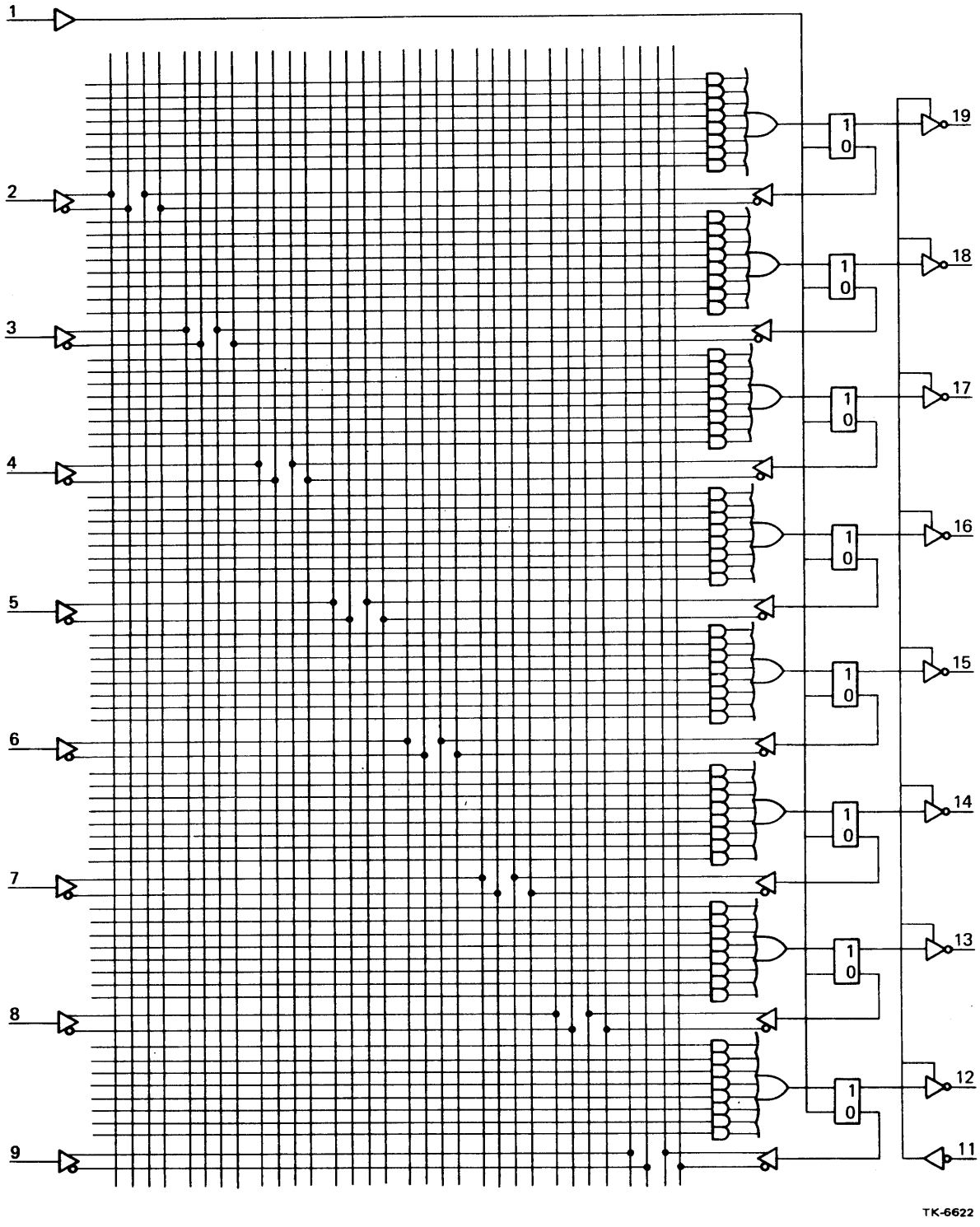


Figure A-8 Logic Diagram: PAL16R8

VAX-11/730 IDC Technical Description

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc? Is it easy to use? _____

What features are most useful? _____

What faults or errors have you found in the manual? _____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

Please send me the current copy of the *Documentation Products Directory*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
Accessories and Supplies Group
P.O. Box CS2008
Nashua, New Hampshire 03061

Attention: Documentation Products
Telephone: 1-800-258-1710

Order No. EK-RB730-TD-001

TW

Fold Here

Do Not Tear – Fold Here and Staple

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MA.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services/Quality Assurance
12 Crosby Drive, BU/E08
Bedford, MA 01730

Digital Equipment Corporation • Bedford, MA 01730