EK-HS571-TM-001

# HSC50/70 Hardware Technical Manual

Prepared by Educational Services
of
Digital Equipment Corporation

**July 1986**



The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

| | | |
|---|---|---|
| DEC | DECnet | UDA50 |
| DECUS | DECsystem-10 | HSC |
| DIGITAL | DECSYSTEM-20 | MASSBUS |
| PDP | RSX | RA80 |
| UNIBUS | VMS | RA81 |
| VT | RA60 | KDA50-Q |
| RC25 | TA78-81 | |
| VAX | TU78-81 |  |
| RA80 | KDB50 | |

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

## TABLE OF CONTENTS

**FIGURES**

TABLES

# Preface

This manual provides a technical description of the HSC50/70 subsystem and is for use by support level field service representatives. It includes:

- A block diagram description of the HSC50 and HSC50 hardware
- A block diagram description of each module in the HSC50/70
- A description of the buses in the HSC50 and HSC70
- A description of the power system

All information in this manual is informational and designed to help field service personnel understand the HSC50 and HSC70 components and their interrelationships. Reader knowledge in the following areas is assumed:

- Digital Storage Architecture (DSA)
- Mass Storage Control Protocol (MSCP) and Tape Mass Storage Control Protocol (TMSCP)
- External cabling - Standard Disk Interface (SDI), Standard Tape Interface (STI) and Computer Interconnect (CI)
- Basic HSC50 and HSC70 operation

The following information is not included in this manual:

- User information
- Utilities and diagnostics
- Installation
- Service information
- Software information
- Special chip description
- PAL equations
- Boot PROM listings

For source material on these and other subjects not within the scope of this manual, refer to the related documentation list at the end of Chapter 1.

**NOTE**

**Because this manual covers both the HSC50 and HSC70, the acronym HSC is used when both servers perform the same function.**

# CHAPTER 1
# OVERVIEW

## 1.1 INTRODUCTION

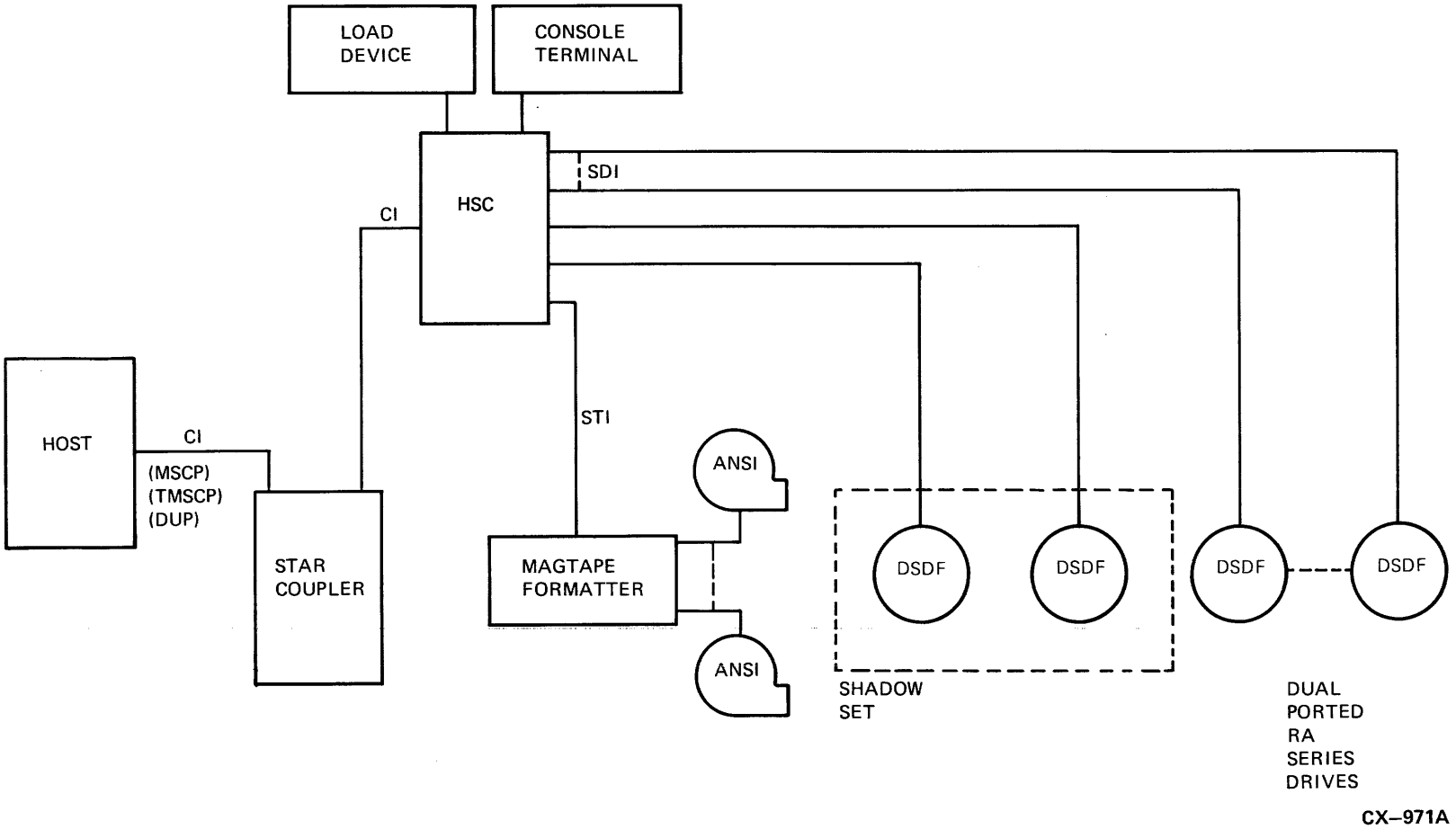
This chapter gives you an overview of what the HSC mass storage server is capable of doing and how it works in a subsystem and in a cluster.

## 1.2 THE HSC SUBSYSTEM

The HSC mass storage server is a stand-alone controller in the Digital Storage Architecture (DSA). See Figure 1-1. The HSC:

- Connects to the hosts(s) via a Computer Interconnect (CI) Bus.
- Follows the rules of System Communications Architecture (SCA) and System Communications Services (SCS) because it is a passive node on a cluster. (A passive node does not initiate commands on the CI Bus.)
- Receives Mass Storage Control Protocol (MSCP) commands from a host and returns MSCP end messages. It sends MSCP error messages and MSCP attention messages to the host when appropriate.
- Provides:
    - MSCP disk server
    - MSCP tape server
    - Diagnostics and Utilities Protocol (DUP) server
- Serves multiple hosts since it is part of a CI cluster. The HSC is the first DEC mass storage server to do so.
- Manages both disk units and tape formatters. By using the Digital Storage Architecture, you can add new types of mass storage units to the controller when the units become available.
- Runs its own self-contained tests on initialization. If the tests pass, it loads and runs more comprehensive self-tests. If these tests pass, it then boots the functional software or off-line diagnostic software from the load device.

**Figure 1-1 HSC Subsystem**



- Runs in-line diagnostics in parallel with the function of an MSCP server. Some in-line diagnostics test the disk and tape units and some test the HSC itself. These diagnostics can be initiated in several ways:
    - Automatically (spawned by error thresholds)
    - Periodically (started by internal software timer)
    - On Demand (initiated via the local terminal)
    - Via DUP commands (from the host)

  There are no host resident (on-line) diagnostics for either the HSC or the disk or tape units. However, the HSC echoes a CI Bus exerciser test packet sent from a host.
- Runs its own set of local utilities in parallel with the MSCP server. They can be initiated manually from the local terminal or via a DUP command from a host.
- Runs off-line diagnostics. These off-line diagnostics are loaded from the internal HSC load device and run in off-line mode. While in this mode, the server does not respond to the CI and, therefore, is not functioning as an MSCP server. These off-line diagnostics are selected and run by the operator via the local terminal. They test only the HSC itself, not the disk or tape units. The only way to get back to the normal user mode is to reboot using the HSC functional software.
- Manages the I/O operations by initiating seeks and head switches necessary to select the proper track. It also monitors rotational position, checks headers, reads/writes data, checks for errors, provides error handling, etc.
- Resequences the command queue for faster throughput.
- Fragments long transfers for faster throughput.
- Revectors bad blocks.
- Provides server-initiated bad block replacement (CIBBR).
- Provides local shadowing. Shadowing is maintaining data on two identical disk drives instead of only one. Thus, if a drive fails, the data stored on that drive is available from another member of the shadow set.
- Provides tape backup/restore without using host resources.
- Manages tape I/O operations.
- Ensures data integrity with the following types of data integrity checks:
    - Cyclic redundancy check (CRC) on all CI packets
    - Error detection check (EDC) on all data blocks
    - Error correction code (ECC) on all data blocks
    - Byte parity on all internal memory and buses
- Provides an Operator Control Panel (OCP) fault code display and a red/green LED display on most modules upon detecting a fatal error.
- Provides physical and electrical isolation. It is the first DEC mass storage server to be packaged in its own cabinet with its own backplane, power supplies, and cooling. The CI cables to the host(s) are point to point from the Star Coupler. The SDI/STI cables to the units are radially connected, and all are transformer isolated. This means some components of the subsystem can be disconnected and replaced without having to power down the other components.

- Has no knowledge of file structure. The HSC responds to any legal command from any host. File structure, file management, and file protection are host responsibilities.

## 1.3 HSC SUBSYSTEM COMMUNICATIONS

Figure 1-1 shows the different buses, protocol, and devices that communicate with the HSC subsystem. Within the subsystem, there are certain protocols that have been standardized. Table 1-1 describes what communications and/or protocol take place in the HSC subsystem.

**Table 1-1 HSC Subsystem Communications**

| Device | Description |
|---|---|
| CI | The CI bus carries three different protocols between the host and the controller:<br><br>Mass Storage Control Protocol (MSCP) defines the protocol used between the host and the HSC for disks.<br><br>Tape Mass Storage Control Protocol (TMSCP) defines the protocol used between the host and the HSC for the tape formatters.<br><br>Diagnostics and Utilities Protocol (DUP) defines the protocol used by the host/controller to communicate between the host terminal and the HSC software for running diagnostics and utilities. This allows an operator to run the diagnostics and utilities from a host terminal. |
| SDI | Standard Disk Interconnect (SDI) defines the:<br><br>Protocol used between the HSC and the disk unit.<br><br>Cable and signals used between the HSC and the disk unit.<br><br>Responsibilities of the HSC and disk unit. |
| DSDF | Digital Standard Disk Format (DSDF) defines the layout and format of the disk unit. |
| STI | Standard Tape Interconnect (STI) defines the:<br><br>Protocol used between the HSC and the tape formatter.<br><br>Cable and signals used between the HSC and the tape formatter.<br><br>Responsibilities of the HSC and tape formatter. |
| ANSI | ANSI defines the format used on mag tape. |
| Load Device | The HSC loads the function software or diagnostic software source into memory via the load device. |
| Console Terminal | The console terminal is used to run the diagnostics or utilities. |

## 1.4 THE HSC AS A NODE ON A CLUSTER

The HSC is a passive node on a cluster. Figure 1-2 shows a cluster with two HSCs cabled to dual ported disk drives and tape formatters. In this configuration, you can take an HSC off line without disturbing host computer or storage unit operations. A cluster can be made up of all VAXs or Large Computer Group (LCG) systems.

## 1.5 REFERENCE DOCUMENTATION

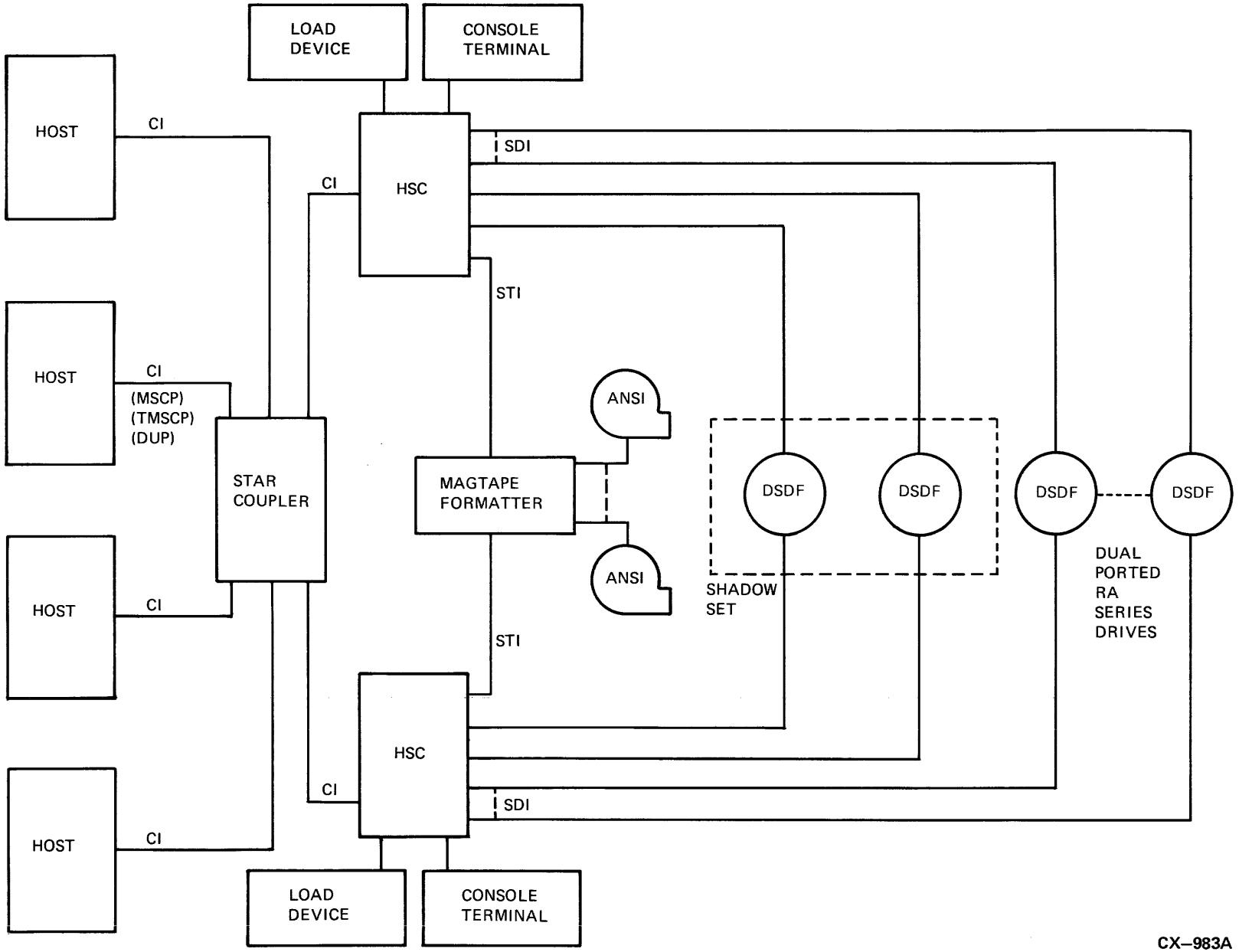
This manual contains HSC50 and HSC70 hardware technical description. Other documents cover different subjects. All are listed in Table 1-2.

**Table 1-2 Reference Documentation**

| Designation | Description |
|---|---|
| AA-GMEAA-TK | *HSC User Guide*—contains information on operator controls and utility programs. |
| EK-HSC70-SV | *HSC70 Service Manual* - contains information necessary for effective service and maintenance of the product. |
| EK-HSC70-MG | *HSC70 Pocket Maintenance Guide* - this guide is a subset of the service manual. |
| EK-HS571-TM | *HSC50/70 Hardware Technical Manual* - contains hardware theory of operation for the HSC50 and HSC70. Distribution is limited to internal support level personnel. |
| EK-HS572-TM | *HSC50/70 Software Technical Manual* - contains software descriptions and internal software and microcode flows for the HSC50 and HSC70. Distribution is limited to internal support level personnel. |
| EK-HS574-TM | *HSC50/HSC70 Chip Description Technical Manual* - contains information on special chips, PAL equations, and Boot PROM listings for the HSC50 and HSC70. |
| EK-HSC70-IP | *HSC70 Illustrated Breakdown*—a collection of illustrations that relate the part to the part number. |
| EK-881PC-UG | *881 Power Controller User Guide*—contains information on the operation of the 881 Power Controller. |
| MP-01422 | *HSC50 Field Print Set*—contains the engineering assembly drawings and circuit schematics of the HSC50. |
| MP-01426 | *HSC70 Field Print Set*—contains the engineering assembly drawings and circuit schematics of the HSC70. |

Manuals beginning with EK can be ordered from Printing and Circulation Services, 10 Forbes Road, Northboro, Massachusetts 01532. The rest of the manuals and print sets can be ordered from the Software Distribution Center, 20 Forbes Road (NR4), Northboro, Massachusetts 01532.

**Figure 1-2 HSC Subsystem in a Cluster**



# CHAPTER 2
# HSC50 AND HSC70 BLOCK DIAGRAM OVERVIEW

## 2.1 INTRODUCTION

The first part of this chapter describes the physical characteristics of the HSC. The second part of the chapter describes the functional characteristics of each module in the HSC.

## 2.2 PHYSICAL CHARACTERISTICS OF HSC

Figure 2-1 shows the physical components of the HSC. These components are:

- AC power controller
- Main power supply
- Auxiliary power supply
- Blower for the logic modules
- Airflow sensor for the blower
- Logic module set
- Four CI coax connectors
- SDI/STI port connectors
  - Twenty four for HSC50
  - Thirty two for HSC70
- Operator Control Panel (OCP)
- Load device
  - TU58 for HSC50
  - Floppy disk for HSC70
- Connector for a console terminal

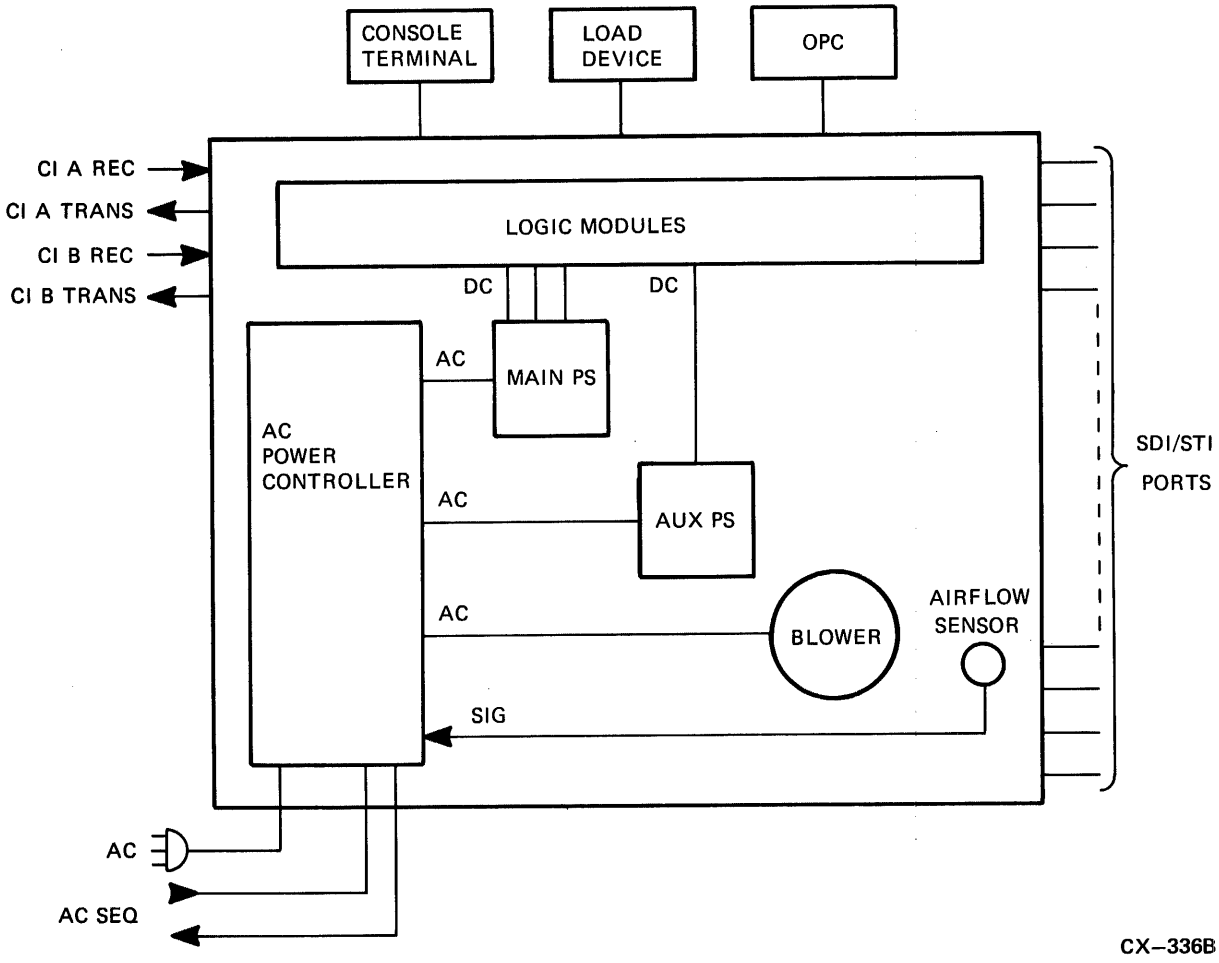
### 2.2.1 AC Power Controller

The three-phase input ac power controller provides single-phase ac power to the main power supply, auxiliary power supply, and the blower for the logic. The HSC has one of the following ac power controllers:

- HSC50-AA—70-19122-01 for 120/208 Vac
- HSC50-AB—70-20613-01 for 380/415 Vac
- HSC70-AA—881A for 120/208 Vac
- HSC70-AB—881B for 380/415 Vac

All four ac power controllers use the Digital Power Sequencing Bus for ac power sequencing. (For more information on the ac power controllers, refer to Chapter 13.)

**Figure 2-1 HSC Components**




### 2.2.2 Power Supplies

There are two power supplies in the HSC: the main and auxiliary. The main power supply is always installed in the HSC50. If the HSC50 has more than eight modules installed, you will see an auxiliary power supply. The HSC70 always has both power supplies. The two power supplies provide the following voltages:

+12 volts (main power supply only)
-5.2 volts (main power supply only)
+5 volts (both power supplies are in parallel)

You cannot adjust these power supplies in the field. (For more information on the power supplies, refer to Chapter 13.)

### 2.2.3 Blower And Airflow Sensor

The blower provides cooling for the logic modules. It receives ac power from the ac power controller.

The airflow sensor monitors the velocity of the exhausted air through the outlet duct. The airflow sensor switch is self heating and requires cool air to stay open. If the blower stops, the sensor switch closes and energizes a relay that does one of the following:

Trips the main circuit breaker in the ac power controller of the HSC50 to remove ac power
De-energizes a relay in the ac power controller of the HSC70 to remove ac power

### 2.2.4 Console Terminal

The HSC has its own console terminal. It allows you to:

Run diagnostics
Run utilities
Receive error printouts
Receive crash dumps

The following terminals are shipped with the HSC:

LA12 with HSC50
VT220 and LA50 with HSC70

### 2.2.5 Load Device

The HSC has a load device that performs the following functions:

Loads operational software
Stores system parameters
Loads diagnostics
Records crash exception information for HSC50
Records crash dumps for HSC70

The following are the two load devices:

TU58 for the HSC50
Floppy disk for the HSC70

### 2.2.6 Operator Control Panel

The Operator Control Panel provides an operator interface to the HSC. The panel contains the following indicators and switches:

State indicator
Power indicator
Init indicator/switch
Fault indicator/switch
Online indicator/switch
Two blank indicators/unused switches
Secure/Enable indicator/switch

## 2.3 FUNCTIONAL CHARACTERISTICS OF HSC

This section describes the functional characteristics of the HSC. First, let's look at the various names of modules and terms you will see in this section. Table 2-1 shows you the various names of the modules used in the HSC50 and HSC70.

**Table 2-1 Module Nomenclature**

| Module Name | Engineering Name | Module Designation |
|---|---|---|
| Port Link | LINK or Interprocessor Link Interface | L0100 |
| Port Buffer | PILA | L0109 |
| Port Processor | K.pli | L0107-YA |
| | Host Interface K.ci | Consists of Port Link, Port Buffer, and Port Processor Modules |
| Disk Data Channel | K.sdi | L0108-YA |
| Tape Data Channel | K.sti | L0108-YB |
| | Data Channel | Disk Data Channel<br>Tape Data Channel |
| Input/Output Control Processor (HSC50) | P.ioc | L0105 |
| Input/Output Control Processor (HSC70) | P.ioj | L0111 |
| Memory (HSC50) | M.std | L0106 |
| Memory (HSC70) | M.std2 | L0117 |
| | M.ctl | Control Memory |
| | M.prog | Program Memory<br>Private Memory |
| | M.data | Data Memory |
| | K.rx | Floppy Disk Controller |

Table 2-1 (Cont.) Module Nomenclature

| Module Name | Engineering Name | Module Designation |
|---|---|---|
| | Requestor | Any module that requests the use of an internal HSC bus |
| | K | A K.sdi, K.sti, or K.pli |
| | R0<br>R9 | Requestor 0 through 9 (lowest priority is 0, highest is 9) |
| | R0 | P.ioc<br>P.ioj |
| | R1 | K.pli |
| | R2—R9 | K.sdi or K.sti |

### 2.3.1 HSC50 Block Diagram

Figure 2-2 is a functional block diagram of the HSC50. The module set consists of the following modules:

Port Link Module (LINK)
Port Buffer Module (PILA)
Port Processor Module (K.pli)
Data Channel Module (K.sdi or K.sti)
Memory Module (M.std)
I/O Control Processor (P.ioc)

Also notice that the HSC50 has:

A console terminal
An Operator Control Panel (OCP)
A dual TU58 load device

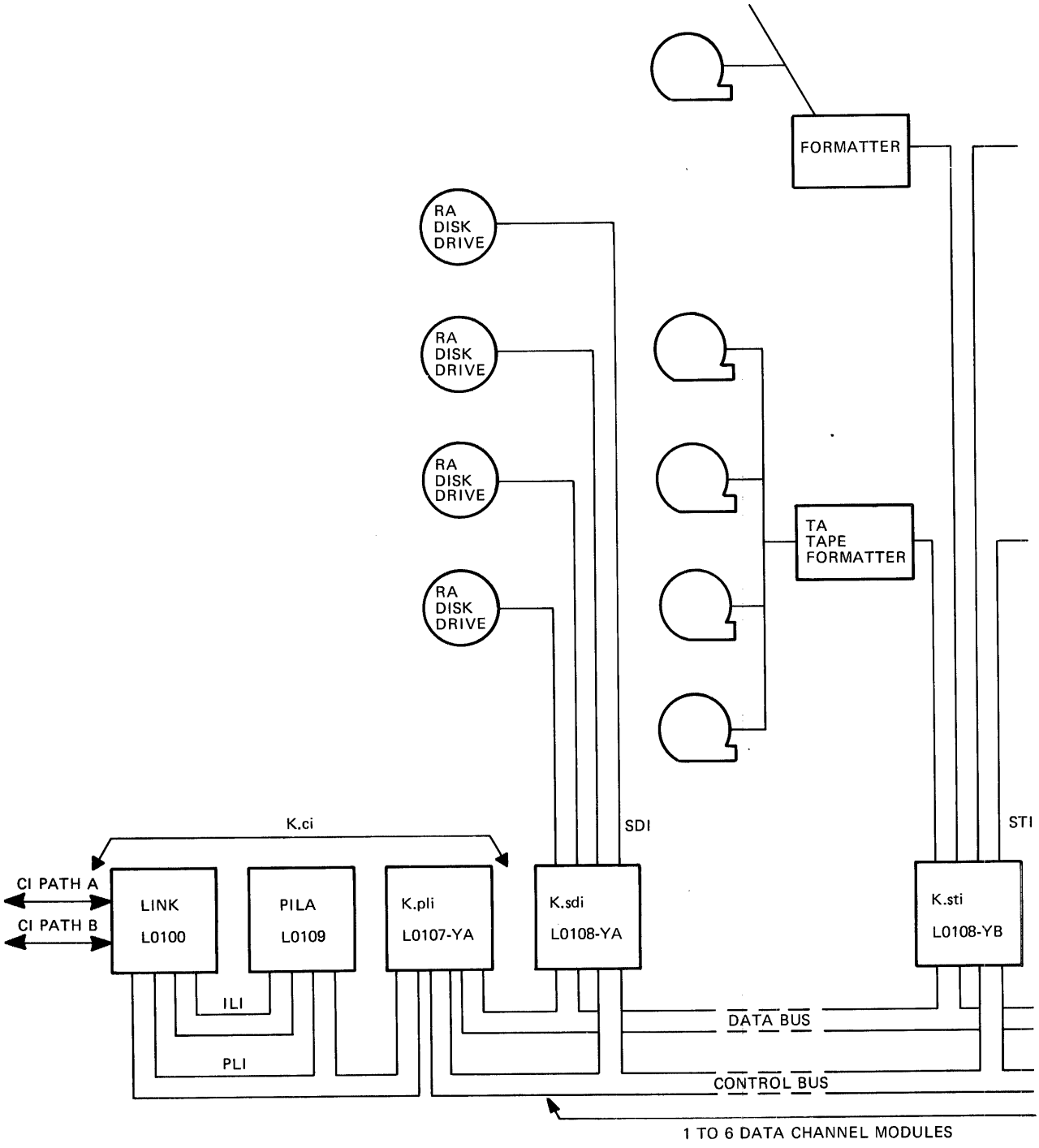
### 2.3.2 HSC70 Block Diagram

Figure 2-3 is a functional block diagram of the HSC70. The HSC70 has different features than the HSC50. The HSC70 features are described below:

- A new I/O Control Processor Module (P.ioj)
- A new Memory Module (M.std2)
- A new load device (floppy disk drive)
- Two more data channel modules (eight maximum)
- New backplane

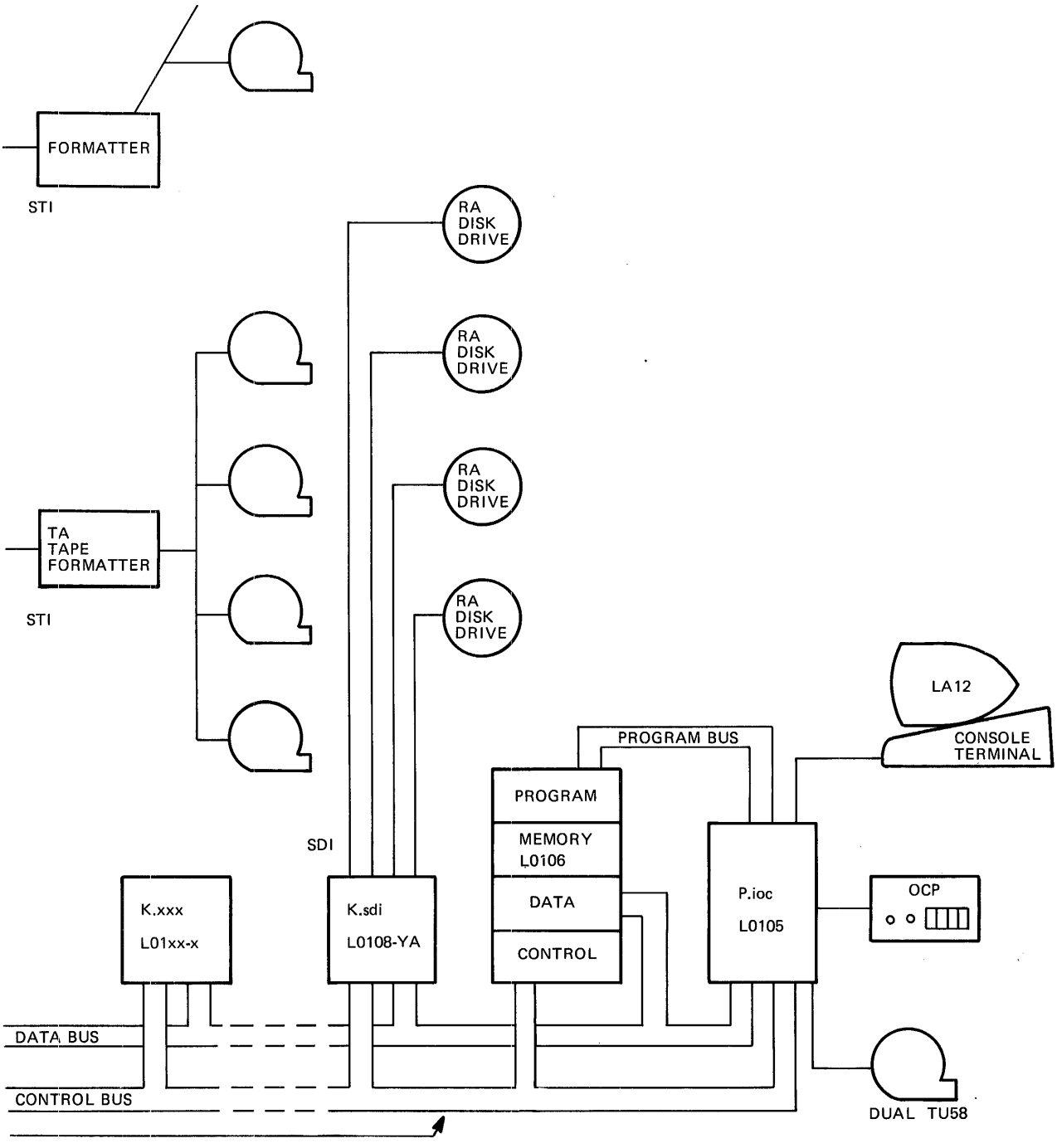
**NOTE**
**You cannot upgrade an HSC50 to an HSC70.**

**Figure 2-2 HSC50 Functional Block Diagram**



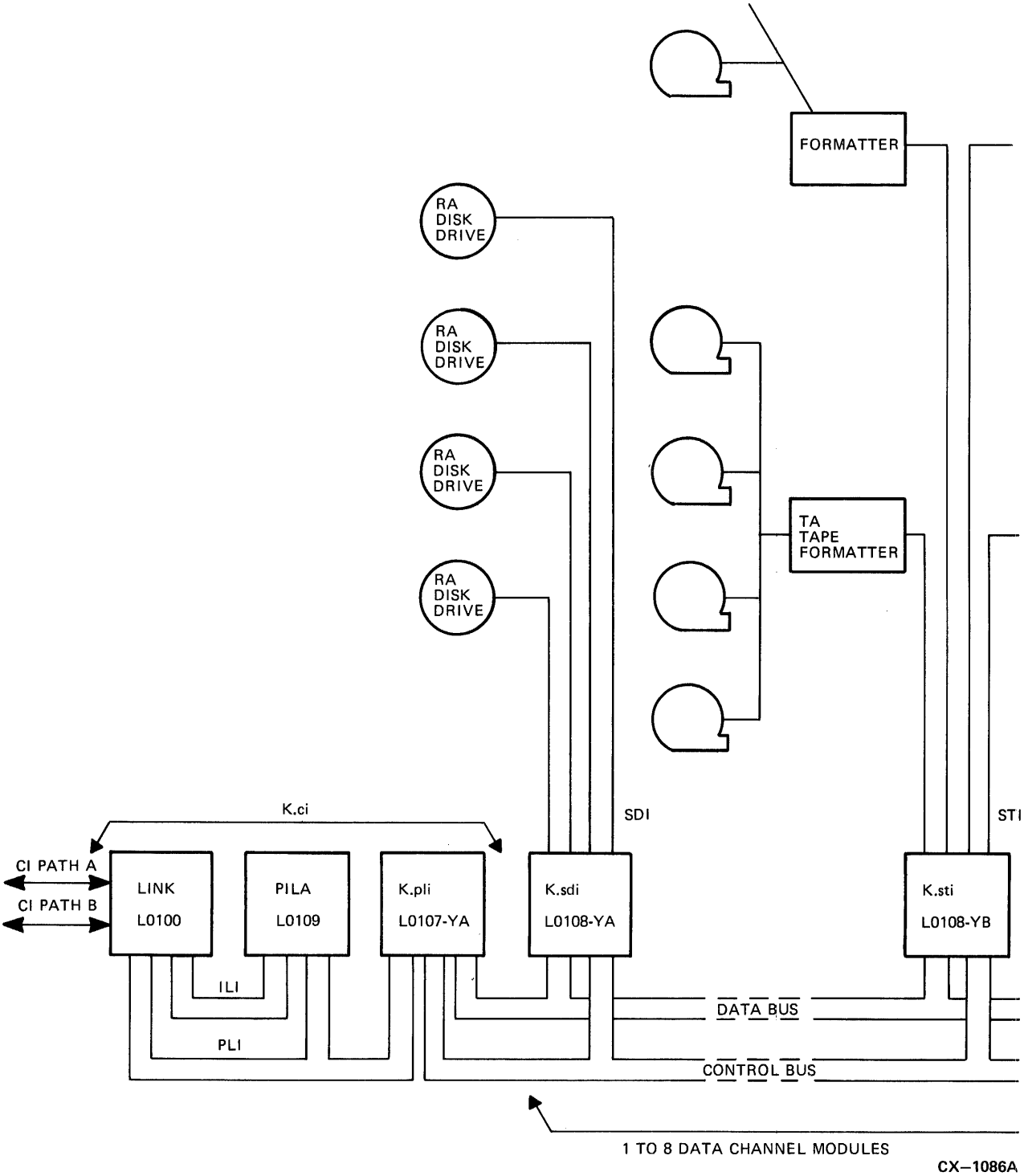

(Continued on next page)

Figure 2-2 (Cont.) HSC50 Functional Block Diagram



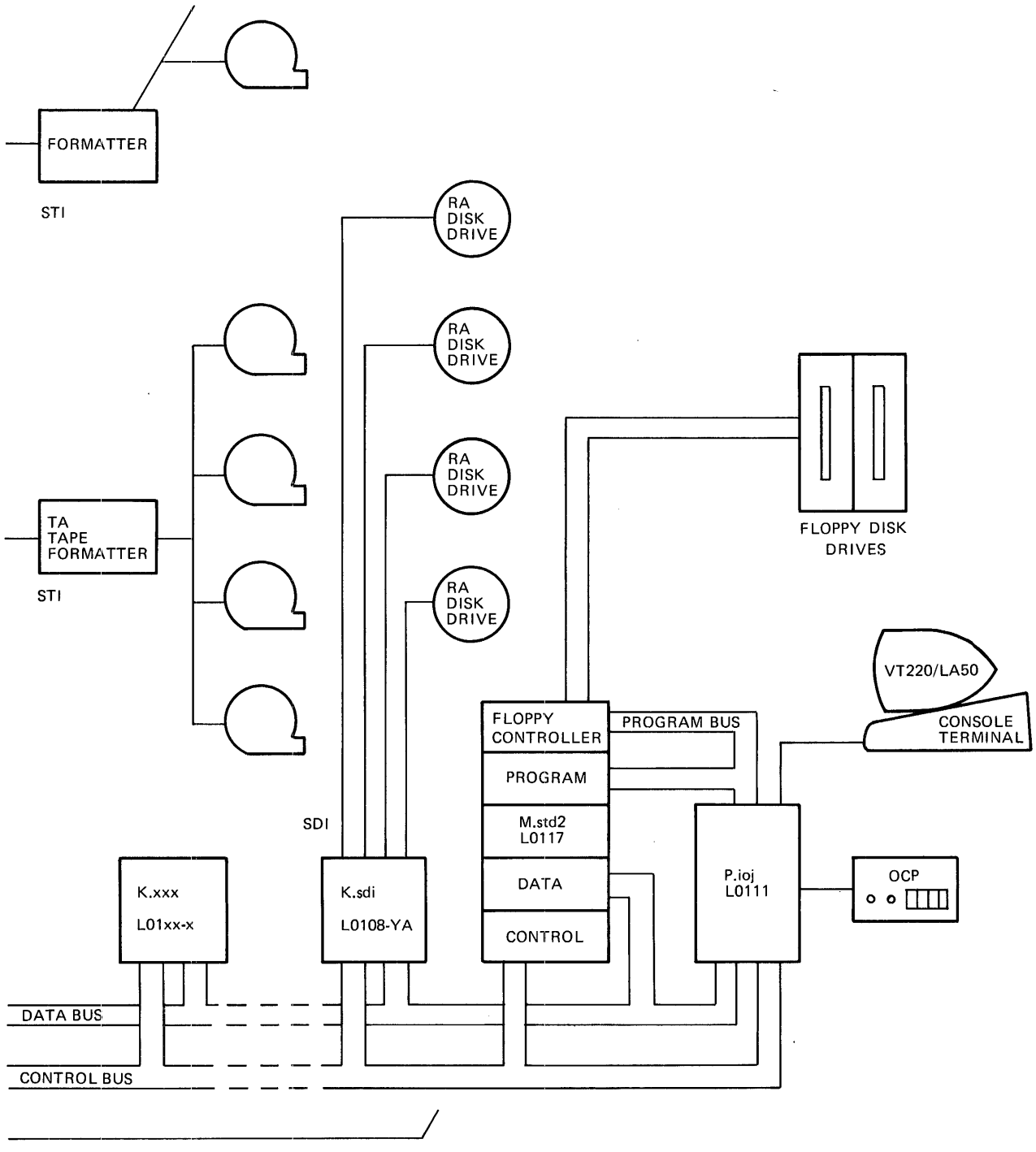

CX–1085A
Sheet 2 of 2

Figure 2-3 HSC70 Functional Block Diagram




(Continued on next page)

**Figure 2-3 (Cont.) HSC70 Functional Block Diagram**




CX–1086A
Sheet 2 of 2

### 2.3.3 Input/Output Control Processor Module (HSC50)

The HSC50 Input/Output Control Processor Module (P.ioc) provides a function similar to that of a CPU in a computer system. The CPU is an F-11 chip set that performs the standard PDP-11/34 instruction set. The L series hex height module (L0105) is in slot one of the backplane. This module contains the following (Figure 2-4):

- ROM (for self-test and booting software)
- Memory Management Unit (MMU)
- Line clock (fixed frequency)
- ID register (serial number)
- I/O page registers
- DC power supply voltage monitor
- Control and Data Bus arbitration logic
- Parity generation and checking for the three memories
- Control Memory Windows (simplifies the addressing scheme of the F-11 when accessing Control Memory)

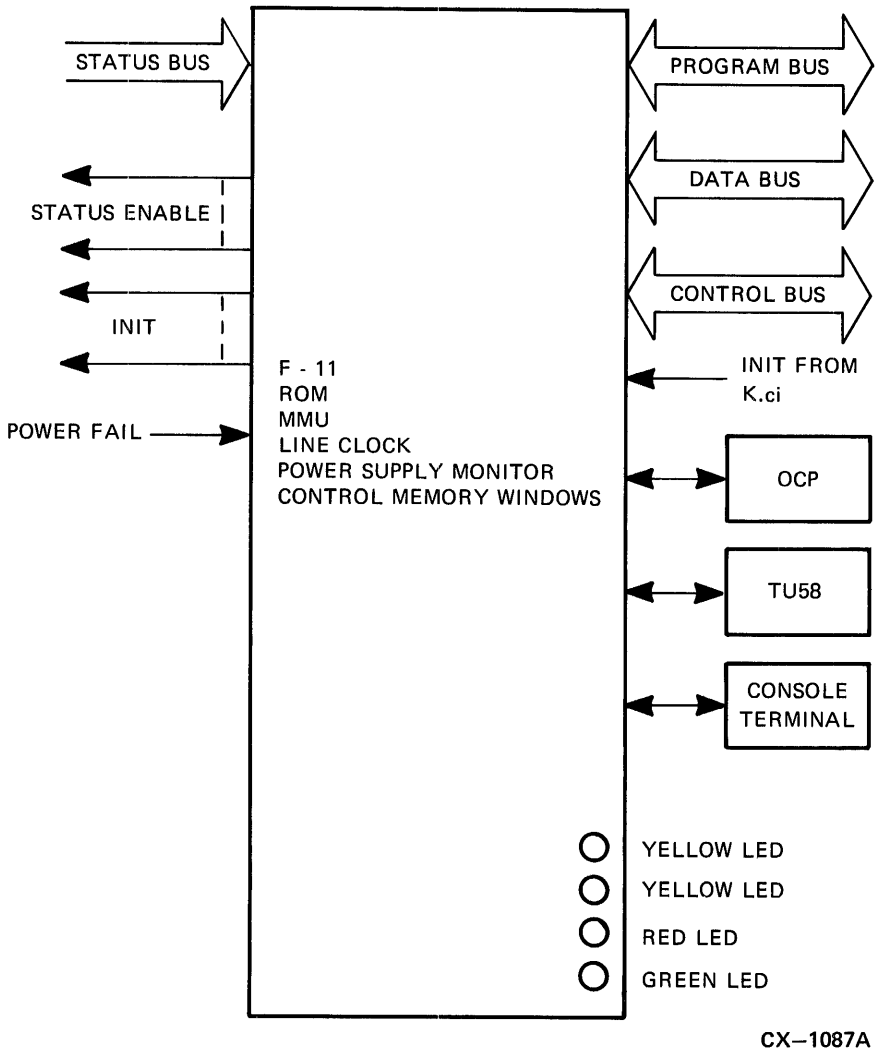
The P.ioc uses the following interfaces:

- Status Bus (status of each K)
- Status Enable (enable status of K to Status Bus)
- INIT (initialize each K)
- Power fail signal from the power supplies (ac dropping) for a standard PFI (trap to 24)
- Program Bus (termination and arbitration on P.ioc)
- Control Bus (termination and arbitration on P.ioc)
- Data Bus (termination and arbitration on P.ioc)
- INIT for K.pli (initialize the P.ioc)
- Operator Control Panel (OCP) through the switch/display register in the I/O page
- TU58 controller through a serial port
- Console terminal through a serial port
- Four LEDs for module status

The P.ioc has four LEDs that help you troubleshoot the HSC50. Figure 2-4 shows the location of the LEDs. The following describes the function of each LED:

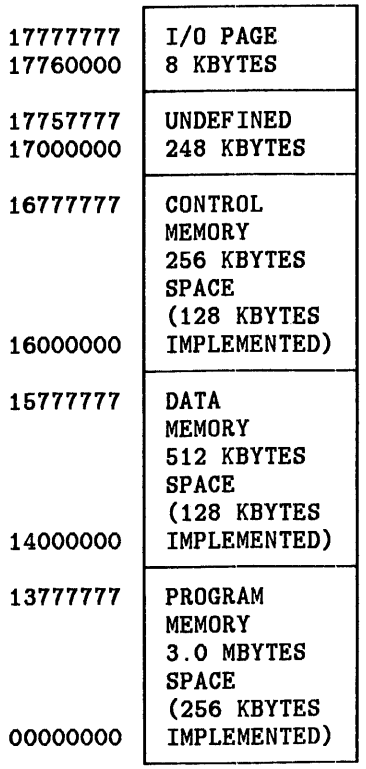
- The top yellow LED is the State indicator (this LED and the State indicator on the OCP blink to indicate normal operation)
- Next to the top yellow LED is the RUN indicator (blinks once each time the F-11 fetches an instruction)
- The red LED is the Diagnostic/Testing Fail indicator (initialized on and then turned off by the software after passing diagnostics)
- The green LED is the Diagnostic Passed indicator (turned on when the red LED is turned off)

Figure 2-4 Input/Output Control Processor Module (HSC50)



The 22-bit Address Bus allows the F-11 to address up to 4 Mbytes of memory. Figure 2-5 shows the starting address of the memories and the I/O page.

**Figure 2-5 22 Bit Address Allocation (HSC50)**

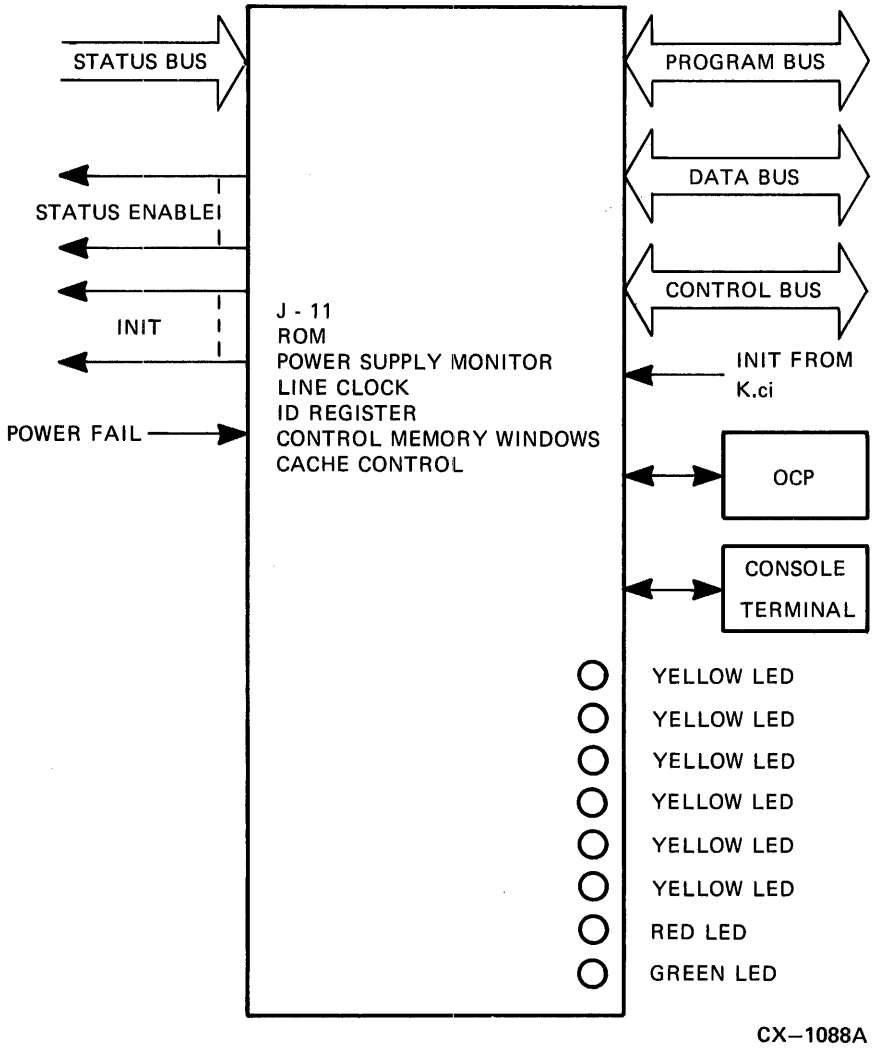



### 2.3.4 Input/Output Control Processor (HSC70)

The HSC70 Input/Output Control Processor Module (P.ioj) provides a function similar to that of a CPU in a computer system. The CPU is a J-11 chip set that performs the standard PDP-11/70 instruction set. The L series hex height module (L0111) is in slot one of the backplane. This module contains the following (Figure 2-6):

- ROM (for self-test and booting software)
- 8 Kbyte instruction cache
- Line clock (fixed frequency)
- ID register (serial number)
- I/O page registers
- DC power supply voltage monitor
- Control and Data Bus arbitration logic
- Parity generation and checking for the three memories

**Figure 2-6 Input/Output Control Processor Module (HSC70)**




- Control Memory Windows (simplifies the addressing scheme of the J-11 when accessing Control Memory)

The P.ioc uses the following interfaces:

- Status Bus (status of each K)
- Status Enable (enable status of K to Status Bus)
- INIT (initialize each K)
- Power fail signal from the power supplies (ac dropping) for a standard PFI (trap to 24)
- Program Bus (termination and arbitration on P.ioj)

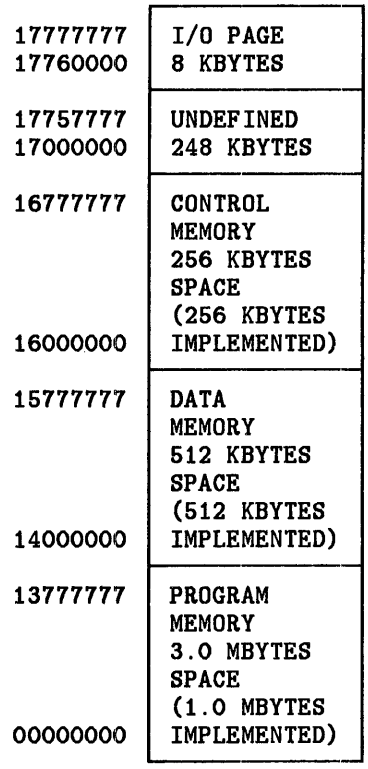
- Control Bus (termination and arbitration on P.ioj)
- Data Bus (termination and arbitration on P.ioj)
- INIT for K.pli (initialize the P.ioj)
- Operator Control Panel (OCP) through the Switch/Display Register in the I/O page
- Console terminal through a serial port
- Eight LEDs for module status

The P.ioj has eight LEDs that help you troubleshoot the HSC70. Figure 2-6 shows the location of the LEDs. The following describes the function of the LEDs.

- The top yellow LED is on at power up and then turned off by the initialization microroutine in the J-11. When the J-11 enters $\mu$ODT, the J-11 turns on the LED. When the J-11 exits $\mu$ODT, the J-11 turns off the LED.
- The next yellow LED is on at power up and then turned off by the initialization microroutine in the J-11. The microroutine turns the LED off after testing that Program Memory:
  - Location $00000000_8$ does not generate a non-existent memory (NXM)
  - Location $17777700_8$ does generate an NXM
- The next yellow LED is on at power up and then turned off by the initialization microroutine in the J-11. The microroutine turns the LED off after testing that the I/O address of the console terminal is not an NXM address.
- The next yellow LED is on at power up and then turned off by the initialization microroutine in the J-11. The microroutine turns the LED off after initializing the J-11 chip set and testing the CPU error register.
- The next yellow LED is the State indicator (this LED and the State indicator on the OCP blink to indicate normal operation)
- The next yellow LED is the RUN indicator (blinks once each time the J-11 fetches an instruction)
- The red LED is the Diagnostic/Testing Fail indicator (initialized on and then turned off by the software after passing diagnostics)
- The green LED is the Diagnostic Passed indicator (turned on when the red LED is turned off)

The 22-Bit Address Bus allows the J-11 to address up to 4 Mbytes of memory. Figure 2-7 shows the starting address of memories and the I/O page.

**Figure 2-7 22 Bit Address Allocation (HSC70)**




### 2.3.5 Memory Module (HSC50)

The HSC50 Memory Module (M.std) contains the following three memories (Figure 2-8):

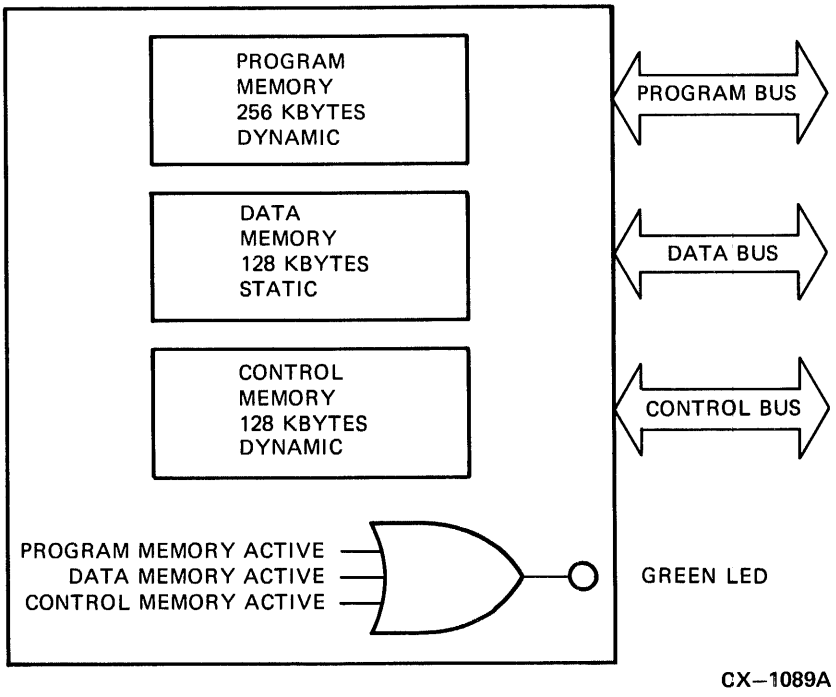
- Program Memory—used for program storage for the P.ioc—256 Kbytes
- Data Memory—used for data buffers—128 Kbytes
- Control Memory—used for control blocks - 128 Kbytes

This module is an L0106 hex height module and is in slot two of the backplane.

The Memory Module has no self-test; however, the following modules test the memory module:

P.ioc
K.sdi/K.sti
K.pli

Figure 2-8 Memory Module (HSC50)



The word structure is a 16-bit data word plus high-byte and low-byte parity bits. The Memory Module provides no parity generation or checking. Those responsibilities belong to the modules that access the memories. The memories interface to the following buses:

Program Memory Bus
Control Memory Bus
Data Memory Bus

The Memory Module has one green LED that indicates memory activity. (Refer to Figure 2-8.) The condition shown is an OR of the three memory cycles.

### 2.3.6 Memory Module (HSC70)

The HSC70 Memory Module (M.std2) contains the following logic (Figure 2-9):

- Program Memory—used for program storage for the P.ioj—1.0 Mbytes
- Data Memory—used for data buffers—256 Kbytes
- Control Memory—used for control blocks—512 Kbytes
- Floppy Disk Controller

This module is an L0117 hex height module and is in slot two of the backplane.

The Floppy Disk Controller controls the two floppy disk drives. The I/O Control Processor issues commands to and checks status of the controller through the I/O page via the Program Bus.

The Memory Module has no self-test; however, the following modules test the memory module:

P.ioj
K.sdi/K.sti
K.pli

The word structure is a 16-bit data word plus high-byte and low-byte parity bits. The Memory Module provides no parity generation or checking. Those responsibilities lay in the modules that access the memories. The memories interface to the following buses:

Program Memory Bus
Control Memory Bus
Data Memory Bus

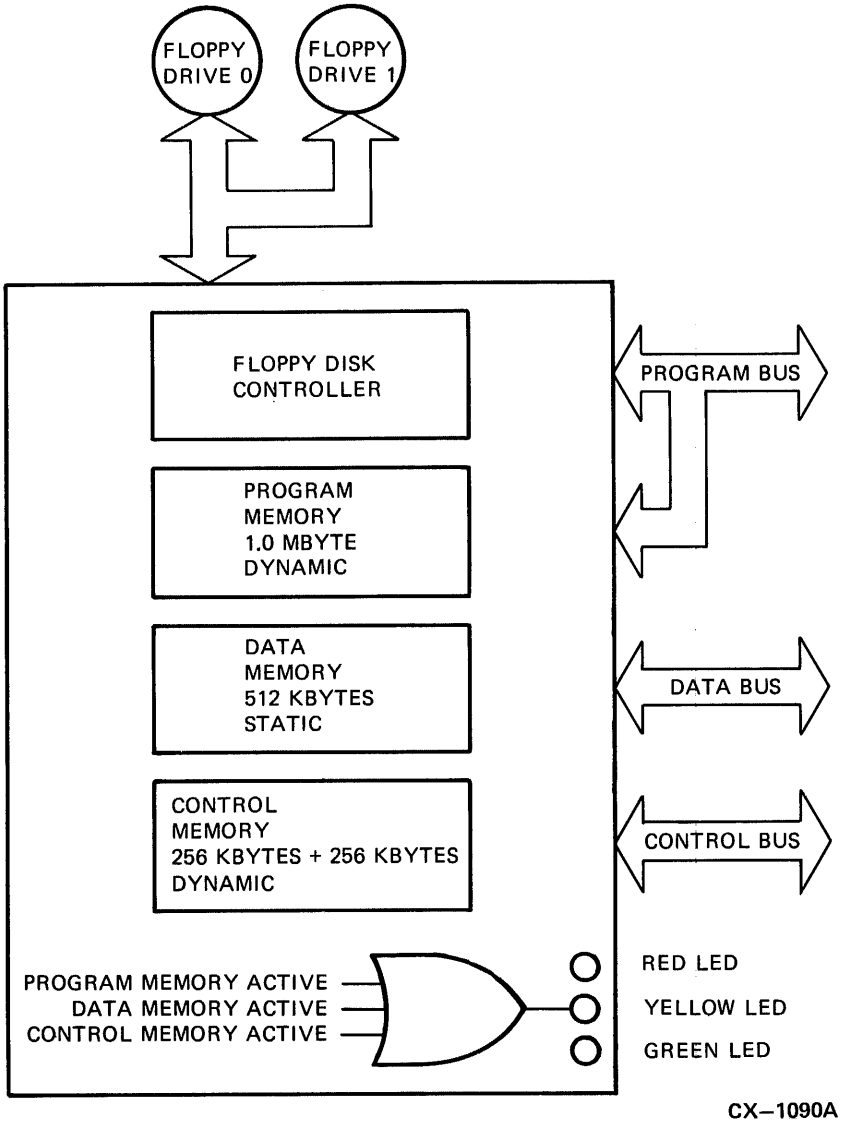
The Memory Module has three LEDs located near the bottom (Figure 2-9). The following describes the function of the LEDs:

- The red LED is the Diagnostic/Testing Fail indicator (initialized on and then turned off by the P.ioj after the memory module passes tests performed by the P.ioj, K.sdi/K.sti, and K.pli)
- The yellow LED is the memory active indicator (the condition shown is an OR of the three memory cycles)
- The green LED is the Diagnostic Passed indicator (turned on when the red LED is turned off)

### 2.3.7 Data Channel Module

The Data Channel modules are the interface between the disk and tape formatters and the HSC. The differences between the disk and tape modules are the contents of the ROMs and one jumper. The ROMs contain the microcode for the microprocessors.

Figure 2-9 Memory Module (HSC70)



There are two types of data channel modules:

Disk Data Channel—L0108-YA—HSC5X-BA—K.sdi
Tape Data Channel—L0108-YB—HSC5X-CA—K.sti

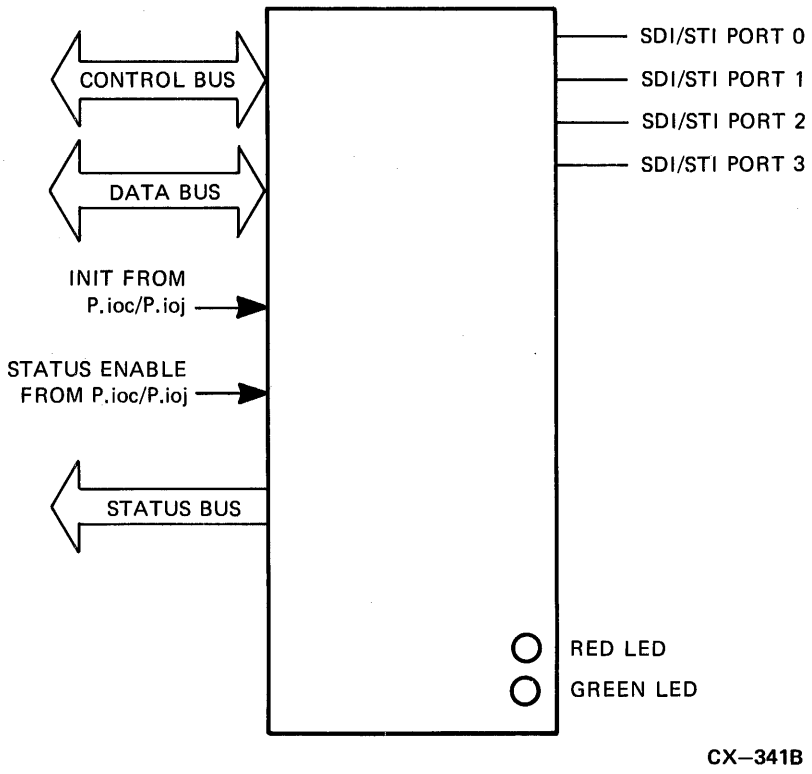
The Data Channel modules are in the following slots on the backplane:

4, 6, 7, 8, 9, or 10 for the HSC50
3, 4, 5, 6, 7, 8, 9, or 10 for the HSC70

The Data Channel uses the following interfaces (Figure 2-10):

- The Control Memory Bus is the interface to the Control Memory.
- The Data Memory Bus is the interface to the Data Memory.
- The INIT from the P.ioc/P.ioj initializes the Data Channel module. When the P.ioc/P.ioj asserts the INIT signal, the Data Channel is held in a cleared state. When the P.ioc/P.ioj negates the INIT signal, the Data Channel starts its self-test, and the result of the test is put into a Status Register on the Data Channel. This register is then enabled onto the Status Bus when the P.ioc/P.ioj asserts STATUS ENABLE.

**Figure 2-10 Data Channel Module**



- The STATUS ENABLE from the P.ioc/P.ioj enables the data channel status onto the Status Bus.
- The Status Bus is the interface to the P.ioc/P.ioj for the Data Channel status.
- The SDI/STI PORT is the interface to the disk drives or tape formatters. Each Data Channel has four ports. However, the data channel only transfers data on one port at a time, but it can be doing other operations on the other three ports (for example, a seek).

The HSC has a priority scheme between the Data Channels, K.pli, and P.ioc for Control and Data Memory access. The slot the module is in determines its priority. The Module Utilization Label (located above the modules) shows the requestor priority on the Control and Data Buses.

Figure 2-11 is an example of the Module Utilization Label on an HSC50. Notice that the priority is shown in the row *Req*. (The lowest priority is zero.) The P.ioc is priority zero (R0). The Data Channel Module in slot four is priority seven (R7). The R in R7 is another way of expressing the word requestor. You should read R7 as requestor 7.

**Figure 2-11 HSC50 Module Utilization Label Example**

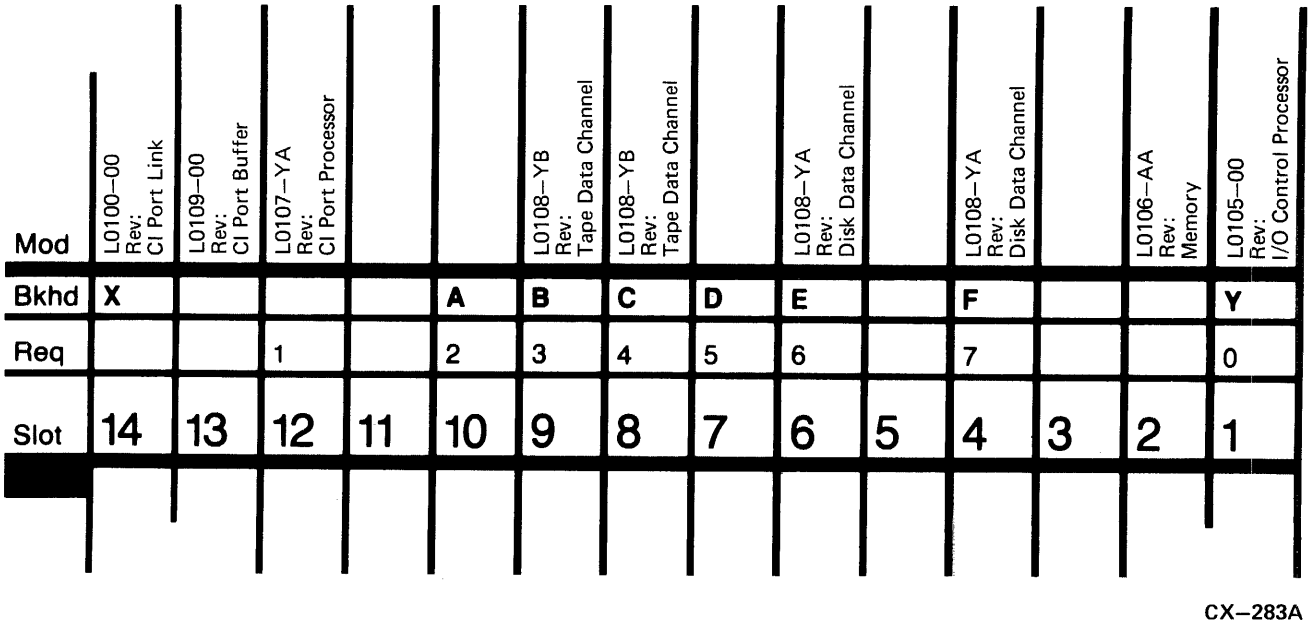
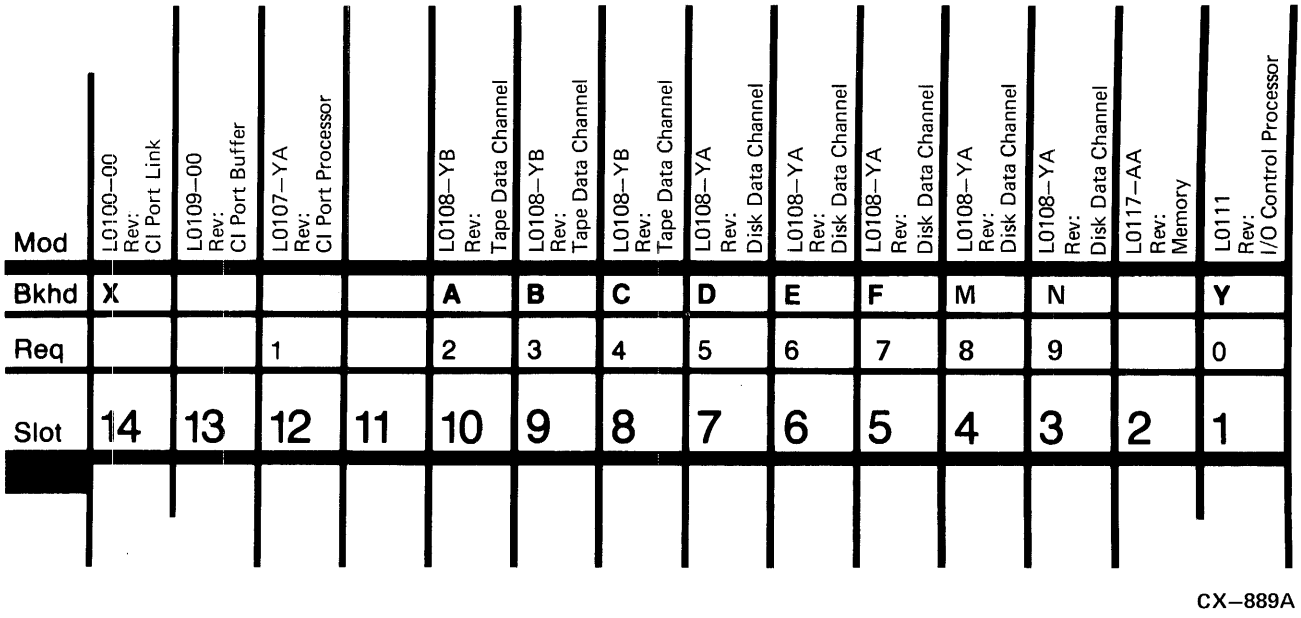



Figure 2-12 is an example of the Module Utilization Label on an HSC70. Notice that the priority is shown in the row *Req*. The P.ioj is priority zero (R0). The Data Channel Module in slot four is priority eight (R8).

**Figure 2-12 HSC70 Module Utilization Label Example**




CX–889A

During write and read operations, the K.sdi checks the error detection code (EDC) and generates or checks the error correction code (ECC). On a write operation, the K.sdi checks the EDC on each block of data it processes. It generates the ECC and appends it to each block (after the EDC) as it leaves the K.sdi on its way to the disk. On a read operation, the K.sdi checks EDC on each block of data from the disk then checks and strips the ECC. If there is an ECC error, the Disk Data Channel and the P.ioc/P.ioj make the correction.

The K.sti does not generate ECC, and it uses a different scheme for EDC. Because tape formatters do not use ECC, no ECC is generated by the K.sti. The K.sti does not write EDC with each record, but it generates an EDC for each 512 bytes during a write operation. The tape formatter generates and sends an EDC every 512 bytes on a read operation.

**NOTE**

**The EDC sent to a tape formatter is never written on the tape media. The tape formatter strips off the EDC before sending the data to the tape drive. The tape formatter appends the EDC to the data before sending it to the HSC.**

Figure 2-10 shows the two LEDs on the Data Channel Module. The following describes the function of the LEDs:

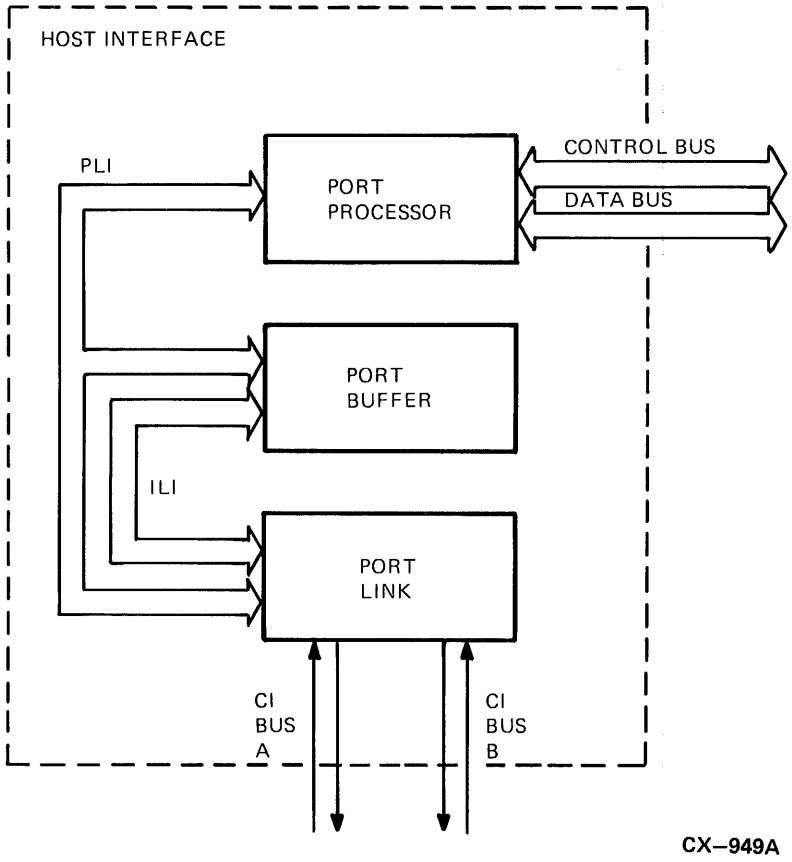
- The red LED is the Diagnostic/Testing Fail indicator (turned on when the P.ioc/P.ioj initializes the K.sdi/K.sti and turned off after the K.sdi/K.sti passes its diagnostics)
- The green LED is the Diagnostic Passed indicator (turned on when the red LED is turned off)

### 2.3.8 Host Interface

The modules in slots 12, 13, and 14 of the HSC are the Host Interface (Figure 2-13). They are the:

Port Processor Module (K.pli)
Port Buffer Module (PILA)
Port Link Module (LINK)

**Figure 2-13 Host Interface**




These three modules are the interface between the CI and the rest of the HSC. The modules have their own interface:

- The Port/Link Interface (PLI) is the data transfer path between the K.pli and PILA and is the control path for the PILA and LINK. The K.pli controls the PILA and LINK in a master slave relationship.
- The Interprocessor Link Interface (ILI) is the data path between the PILA and LINK.

### 2.3.9 Port Processor Module (K.pli)

The Port Processor Module (K.pli) provides the intelligence for the host interface modules. Also, this module is the interface between the K.ci and the rest of the HSC through the Control Bus, the Data Bus, and the Status Bus. This module is an L0107 hex height module and is in slot 12.

The K.pli uses the following interfaces (Figure 2-14):

- The PLI is the interface between the modules in the K.ci.
- The Control Memory Bus is the interface to the Control Memory.
- The Data Memory Bus is the interface to the Data Memory.
- The INIT is how the P.ioc/P.ioj initializes the K.pli. When the P.ioc/P.ioj asserts the INIT signal, the K.pli is held in a cleared state. When the P.ioc/P.ioj negates the INIT signal, the K.pli starts its self-test and the result of the test is put into a Status Register on the K.pli. This register is then enabled onto the Status Bus when the P.ioc/P.ioj asserts STATUS ENABLE.
- The STATUS ENABLE from the P.ioc/P.ioj enables the K.pli status onto the Status Bus.
- The Status Bus is the interface to the P.ioc/P.ioj for the K.pli status.
- The HOST INIT signal initializes the P.ioc/P.ioj causing it to reboot. A host sending a reset causes the K.pli to generate the HOST INIT signal.

The K.pli has three LEDs that help you troubleshoot the HSC. Figure 2-14 shows the location of the LEDs. The following describes the function of the LEDs:

- The red LED is the Diagnostic/Testing Fail indicator (turned on when the P.ioc/P.ioj initializes the K.pli and turned off after the K.pli passes its diagnostics)
- The yellow LED indicates that the P.ioc/P.ioj is asserting INIT (only on some early the C-rev etch modules)
- The green LED is the Diagnostic Passed indicator (turned on when the red LED is turned off)

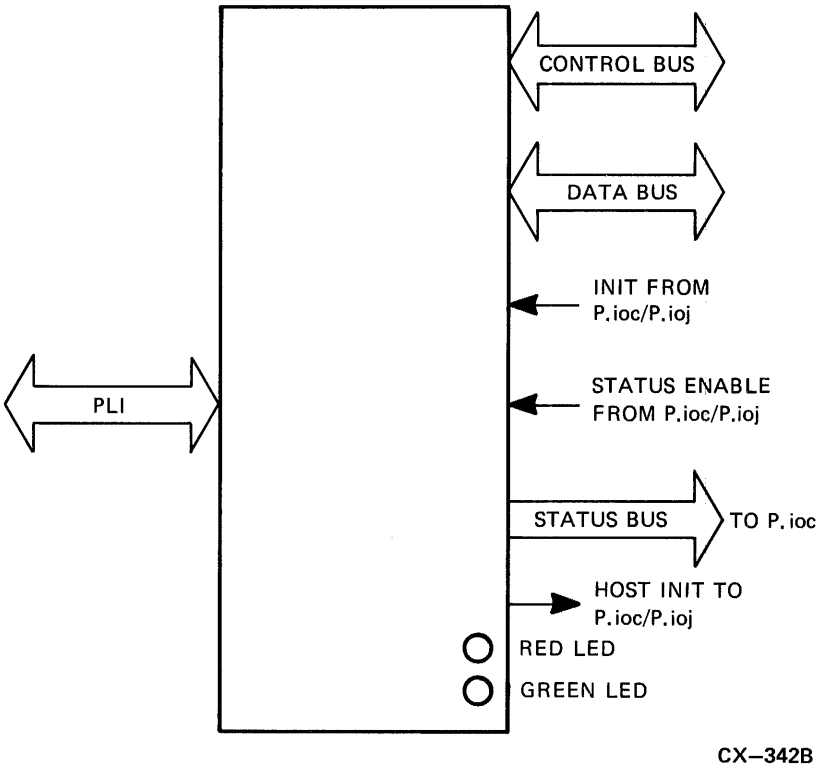
### 2.3.10 Port Buffer Module (PILA)

The Port Buffer Module (PILA) provides buffering for data and message packets. There are two 1-Kbyte buffers for transmit packets and two 1-Kbyte buffers for receive packets. The PILA communicates with the LINK via the ILI and with both the LINK and K.pli via the PLI. This L0109 hex height module is in slot 13.

The PILA has no self-test, but the K.pli tests it and lights the LEDs. Figure 2-15 shows the three LEDs on the PILA. The following describes the function of the LEDs:

- The top yellow LED is a test point indicator. This LED is only found on some early E-rev etch modules.
- The red LED is the Diagnostic/Testing Fail indicator. The K.pli controls this indicator.
- The green LED is the Diagnostic Passed indicator. The K.pli turns this LED on when it turns the red LED off.

Figure 2-14 Port Processor Module (K.pli)




### 2.3.11 Port Link Module (LINK)

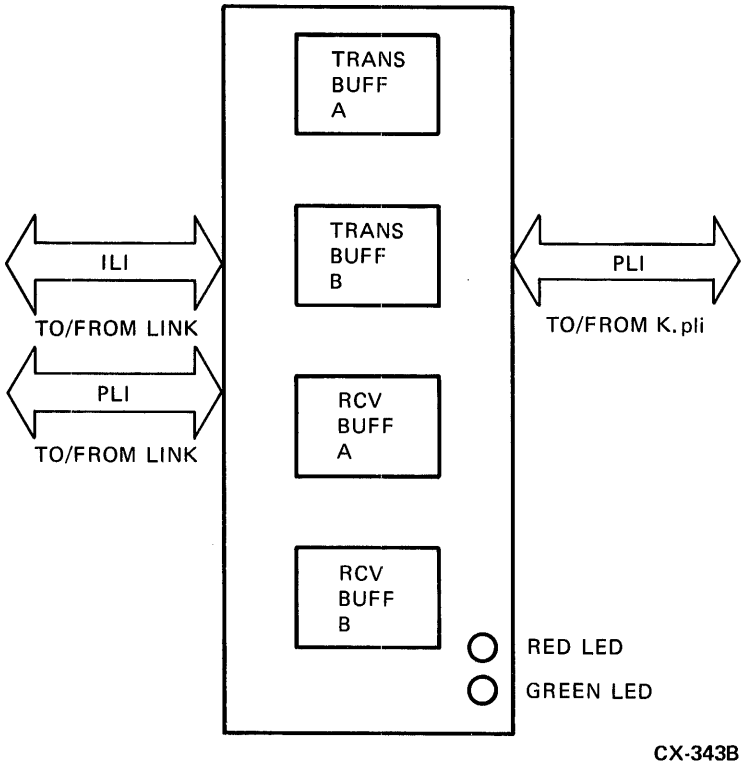
The Port Link Module (LINK) is the same module used in many of the products that interface to a CI. It performs the following functions:

- Interfaces to the dual-path CI bus
- Contains CI arbitration logic
- Contains the node address switches
- Generates/checks CI packet CRC
- Interfaces to the ILI and PLI buses

This L0100 hex height module is in slot 14.

This module has no self-test, but the K.pli tests it. Figure 2-16 shows the two LEDs on the LINK. The following describes the function of the LEDs:

Figure 2-15 Port Buffer Module (PILA)




- The red LED is the maintenance loop indicator. When the K.pli enables the loop back in the LINK, it lights this LED.
- The green LED is the local activity indicator. It lights when there is an incoming or outgoing packet.

### 2.3.12 Backplane

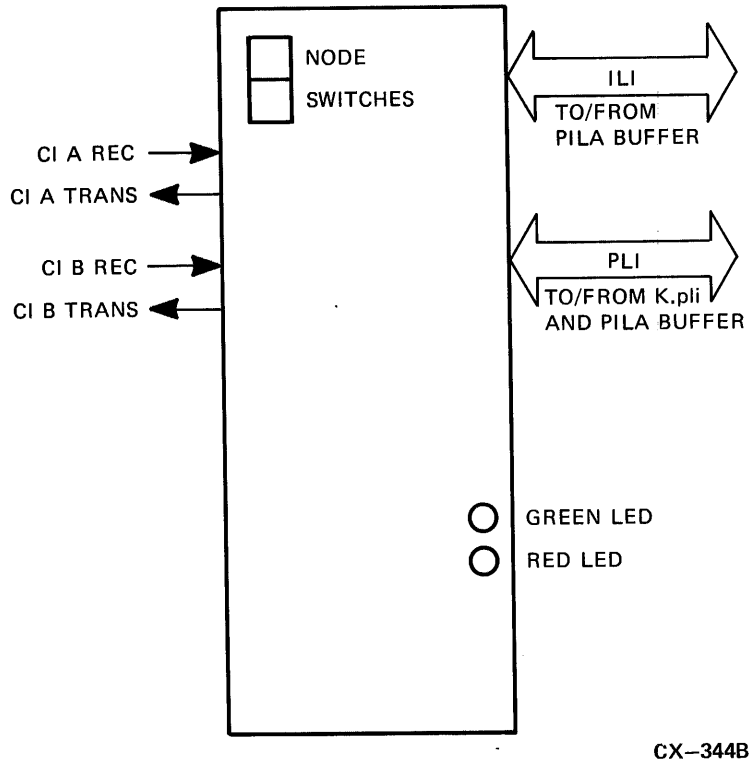
The backplane is a 14-slot, 4-layer, pressed pin arrangement that provides pinning for L series modules. It provides power for the modules and signal buses. The power voltages are:

+12 volts
+5 volts
-5.2 volts

The following describes the buses on the backplane:

- The Program Bus is the interface between Program Memory and the P.ioc/P.ioj. (The K.rx in the HSC70 also uses this bus.)

Figure 2-16 Port Link Module (LINK)




- The Data Bus is the interface between Data Memory and the following modules:

  P.ioc/P.ioj
  K.pli
  K.sdi
  K.sti

- The Control Bus is the interface between Control Memory and the following modules:

  P.ioc/P.ioj
  K.pli
  K.sdi
  K.sti

- The Status Bus is the interface between the P.ioc/P.ioj and the Status Bus Register of following modules:

  K.pli
  K.sdi
  K.sti

- The STATUS ENABLE signals enable the status of the Ks onto the Status Bus.

- The INIT signals initialize the Ks when asserted.
- The HOST INIT signal initializes the P.ioc/P.ioj.
- The ILI Bus is the interface between the LINK and the PILA.
- The PLI bus is the interface between the LINK, PILA, and K.pli.

There are four coax cables from the rear bulkhead CI connectors to the backplane pins for the LINK. There are four SDI/STI port cables, in a harness, from the rear bulkhead for each of the:

Six Data Channel slots (HSC50)
Eight Data Channel slots (HSC70)

#### 2.3.12.1 HSC50 Backplane Jacks

Figure 2-17 shows the location of the jacks on the back of the HSC50 backplane. The following describes the purpose of each jack:

- J11 goes to the following units:

  Console Terminal
  Operator Control Panel
  TU58 tape drives

- J12 has the following wires from the main power supply:

  +12.0 volts
  +12.0 volts, +5.0 volts, and -5.2 volts sense wires
  POWER FAIL signal

- J13 has the following wires from the auxiliary power supply:

  +5 volt sense wire
  POWER FAIL signal

#### 2.3.12.2 HSC70 Backplane Jacks

Figure 2-18 shows the location of the jacks on the back of the HSC70 backplane. The following describes the purpose of each jack:

- J11 goes to the console terminal
- J12 goes to the Operator Control Panel
- J13 has the following wires from the two power supplies:

  +12.0 volts
  +12.0 volts, +5.0 volts, and -5.2 volts sense wires
  POWER FAIL signals

- J18 goes to the floppy disk drives

Figure 2-17 HSC50 Backplane

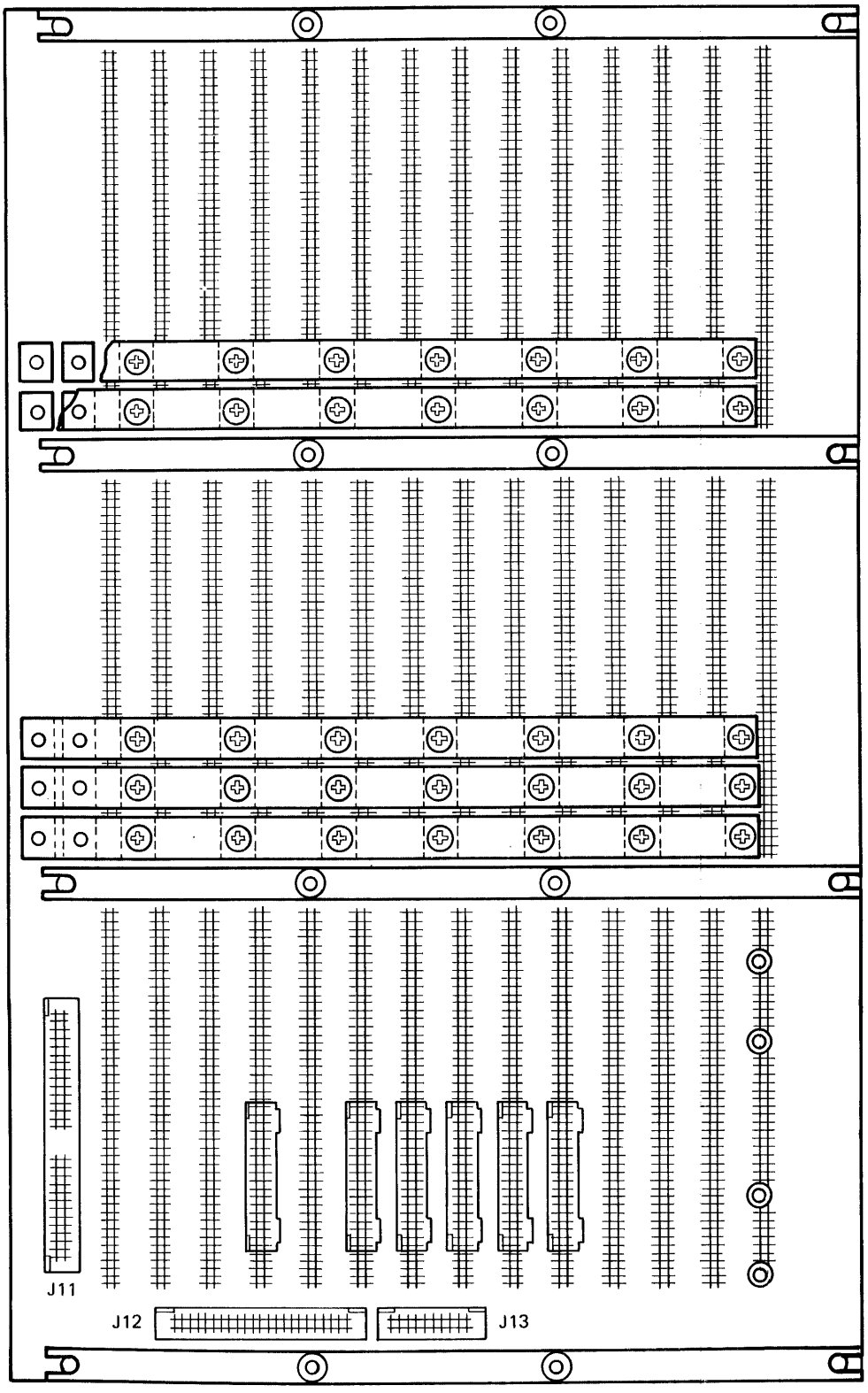
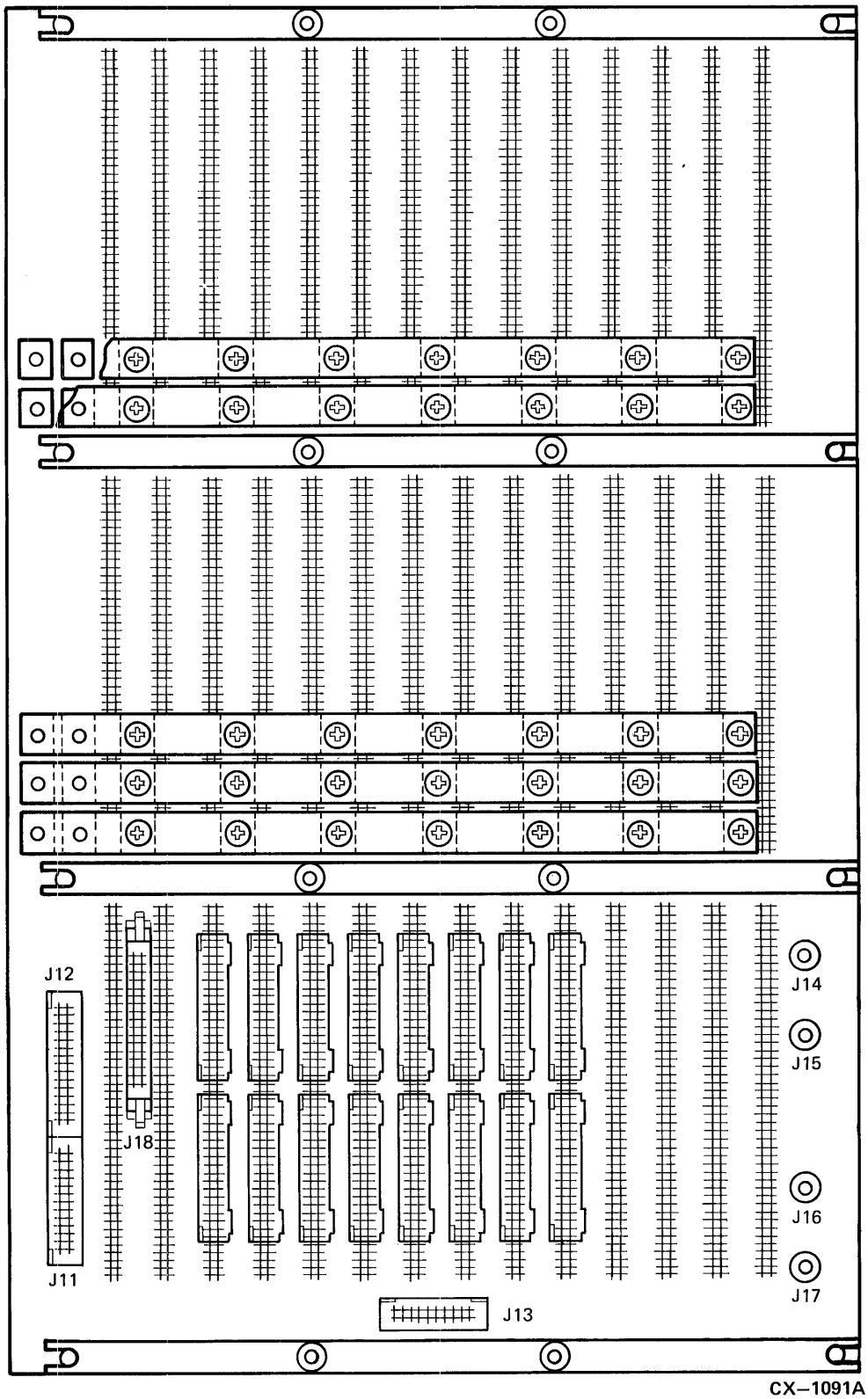


Figure 2-18 HSC70 Backplane



### 2.3.13 Operator Control Panel

The P.ioc/P.ioj reads the switches and controls the indicators on the Operator Control Panel (OCP). Figure 2-19 shows how the P.ioc/P.ioj controls the indicators on the OCP. Figure 2-20 shows how the P.ioc controls the indicators on the OCP. Figure 2-21 shows how the P.ioj controls the indicators on the OCP. For information on the function of each switch and indicator, refer to the *HSC User Guide* or *HSC70 Service Manual.*

The switches go to the following logic:

- The INIT and SECURE/ENABLE switch go to the following logic:
  - F-11 or J-11 to start the booting process
  - Terminal UARTs to do a terminal loopback
  - Control and Status Register (PIO CSR) for position of SECURE/ENABLE switch
- The rest of the switches are read into the Switch and Display Register (SWDR) where the software can act upon them

The indicators are controlled by the logic on the P.ioc/P.ioj. The Voltage Monitor Circuit controls the POWER indicator. If all voltages are within tolerance, the POWER indicator is lit. Notice in Figure 2-21 that the POWER FAIL signal also controls the POWER indicator. When asserted, the POWER FAIL signal turns off the POWER indicator. The software controls the other indicators through the following logic:

- Control and Status Register (PIO CSR) for the STATE indicator
- Switch Display Register (SWDR) for the other of the indicators

Figure 2-19 Operator Control Panel Switches

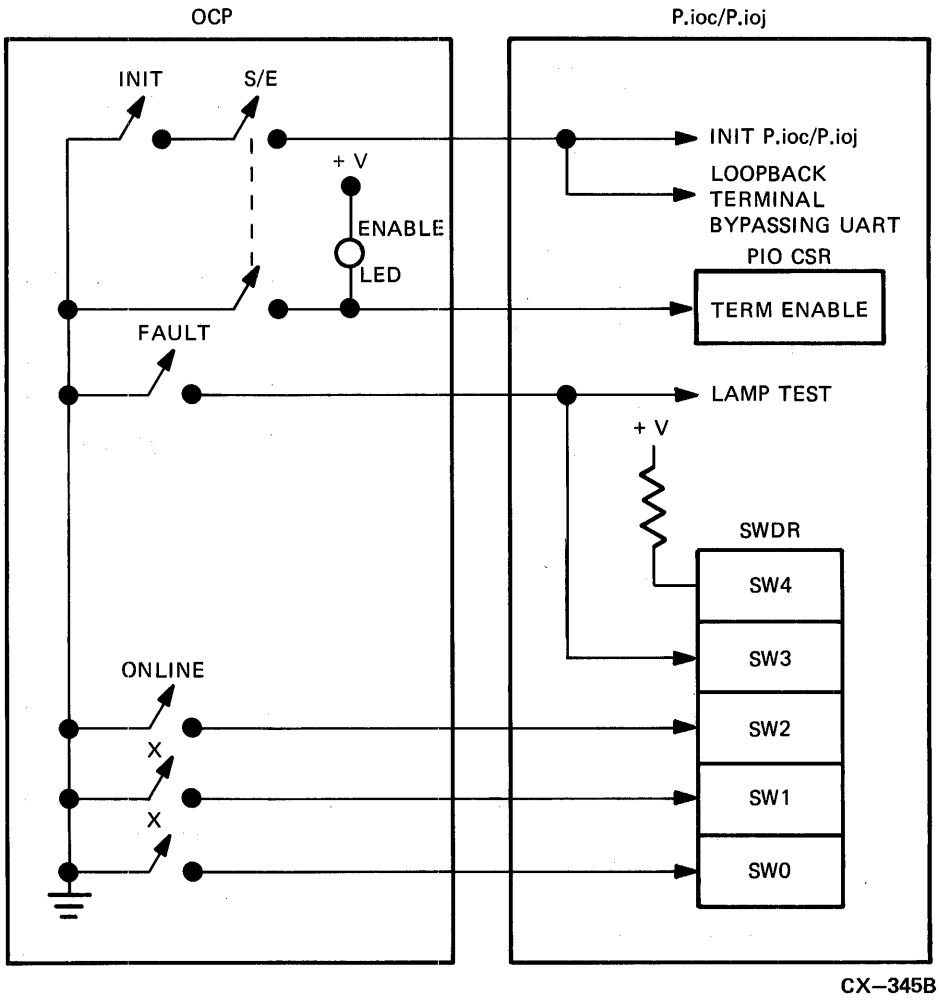


Figure 2-20 HSC50 Operator Control Panel Indicators

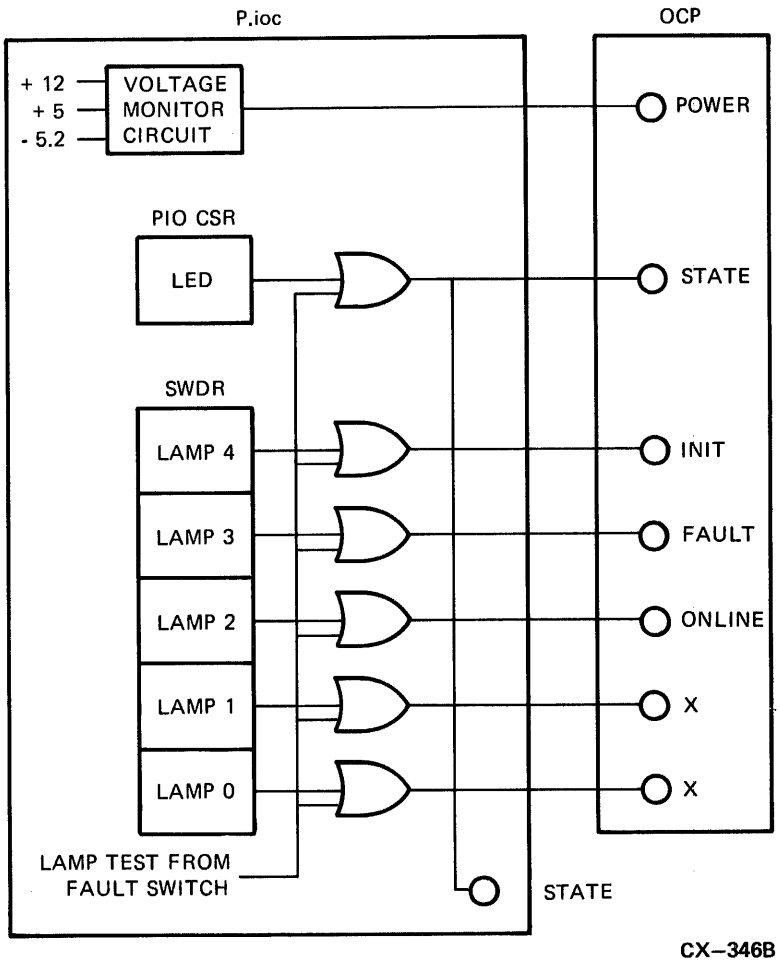
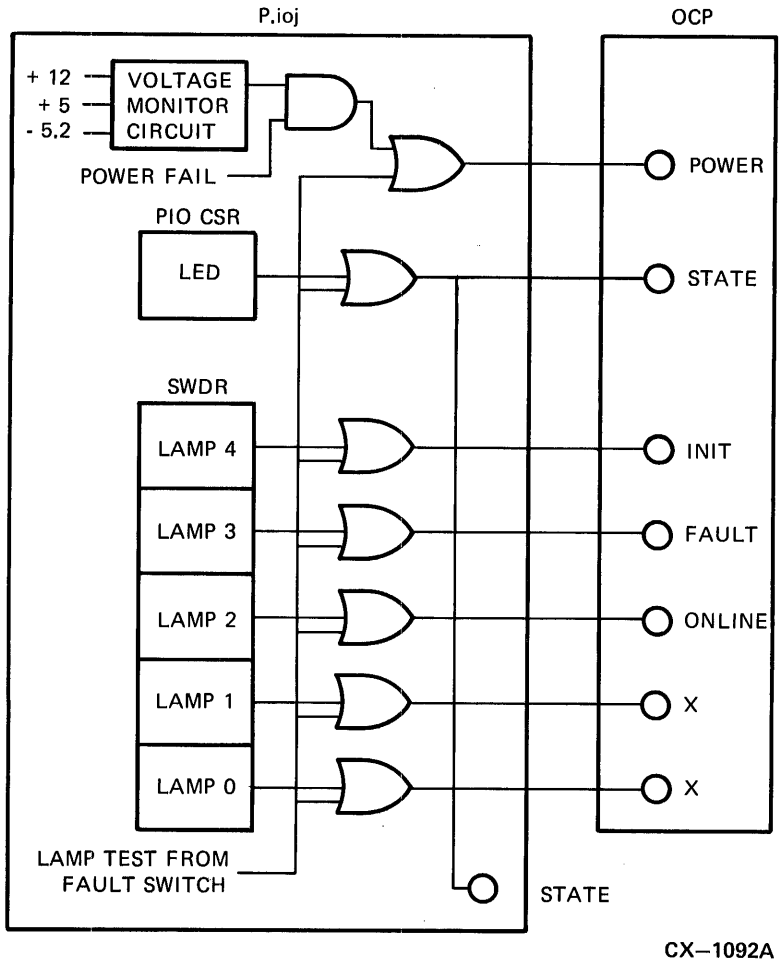


Figure 2-21 HSC70 Operator Control Panel Indicators



# CHAPTER 3
# BACKPLANE BUSES AND CONTROL SIGNALS

## 3.1 INTRODUCTION

This chapter describes the buses and other control signals physically located on the backplane of the HSC. All backplane signals are described with the following exceptions:

- The Floppy Disk Controller (K.rx) signals on connector C even numbered pins 6 through 38 are discussed in Chapter 11.
- The host interface (K.ci) signals on the B and C connectors of slots 13 and 14 and the C connector of slot 12 are discussed in Chapters 5, 6, and 7.
- The console terminal and Operator Control Panel (OCP) signals on the odd numbered pins of connector C in slot 1 are discussed in Chapter 8 and 9.

Descriptions and timing diagrams for the Program Bus, Control Bus, and Data Bus are presented in this chapter. The other miscellaneous control signals discussed are:

- S ENABLE i L
- STATUS <7:0> L
- CLK A H and CLK B H
- CLK CTL H
- CLEAR x H
- HOST CLR
- SLOW CYCLE
- Control and Data BASE IN and BASE OUT lines

## 3.2 HSC50/70 BUS DIFFERENCES

The buses on the backplanes of the HSC50 and HSC70 are functionally identical but differ in some signal names and in implementation of the arbitration logic. Table 3-1 lists each backplane signal name that has a different name on the HSC50 than on the HSC70. The arbitration for buses on the HSC50 is covered in Chapter 8 and for the HSC70 in Chapter 9 since the arbitrators are located on the P.ioc/P.ioj module.

**NOTE**

**The descriptions in this chapter are for both HSCs. Where the HSC50 signals differ from the HSC70 signals, a note will follow that description.**

Table 3-1 HSC50/70 Signal Differences

| HSC50 Signal Name | HSC70 Signal Name | Slot/Backplane Pin Number |
|---|---|---|
| CGRANT 0 L | CGRANT 9 L | B38[1] |
| CGRANT L | CGRANT 8 L | B36 |
| CREFR CLK L | SLOW CYCLE L | B86 |
| CREFR REQ L | SPARE 5 | B87 |
| CREQ 0 L | CREQ 9 L | B37 |
| CREQUEST L | CREQ 8 L | B35 |
| DGRANT 0 L | DGRANT 9 L | A52 |
| DGRANT L | DGRANT 8 L | A50 |
| DREQ 0 L | DREQ 9 L | A51 |
| DREQUEST L | DREQ 8 L | A49 |
| PBASE OUT 0 L | BIRQ 4 L | 1C76 |
| PBASE OUT 1 L | BIAK L | 1C78 |
| PBASE OUT 2 L | BDMR L | 1C80 |
| PBASE OUT 3 L | BDMGO L | 1C82 |
| PBASE OUT 4 L | BSACK L | 1C84 |
| PBASE OUT 5 L | SPARE 3 | 1C86 |
| PENA L | SPARE 4 | 1C88 |
| PBASE IN 0 L | BIRQ 4 L | 2C75 |
| PBASE IN 1 L | BIAK L | 2C77 |
| PBASE IN 2 L | BDMR L | 2C79 |
| PBASE IN 3 L | BDMGO L | 2C81 |
| PBASE IN 4 L | BSACK L | 2C83 |
| PBASE IN 5 L | SPARE 3 | 2C85 |
| S ENABLE 0 L | S ENABLE 9 L | B77 |
| S ENABLE i L | S ENABLE 8 L | B76 |
| SINGLE STEP L | TEST BOOTSTRAP L | C05 |

[1] Where slot numbers are not specified before a pin number, the signal is on multiple slots and the number given is for the P.ioc/P.ioj slot.

Table 3-1 (Cont.) HSC50/70 Signal Differences

| HSC50 Signal Name | HSC70 Signal Name | Slot/Backplane Pin Number |
|---|---|---|
| SWAP MEM BANK H | SWAP PMEM BANK H | C70 |
| SWAP MEM BOARD H | SWAP CMEM BANK H | C68 |
| T1 PRESENT + | AUX1 PRESENT + | 1C35 |
| T1 RCV + | AUX1 RCV + | 1C43 |
| T1 RCV - | AUX1 RCV - | 1C41 |
| T1 XMTR OUT + | AUX1 XMT + | 1C37 |
| T1 XMTR OUT - | AUX1 XMT - | 1C39 |
| T2 PRESENT + | AUX2 PRESENT + | 1C45 |
| T2 RCV + | AUX2 RCV + | 1C43 |
| T2 RCV - | AUX2 RCV - | 1C41 |
| T2 XMTR OUT + | AUX2 XMT + | 1C37 |
| T2 XMTR OUT - | AUX2 XMT - | 1C39 |
| | FLOPPY SPARE | 2C6 |
| | IN USE L | 2C8 |
| | DRV SEL 3 L | 2C10 |
| | INDEX L | 2C12 |
| | DRV SEL 0 L | 2C14 |
| | DRV SEL 1 L | 2C16 |
| | DRV SEL 2 L | 2C18 |
| | MOTOR ENB L | 2C20 |
| | DIR SEL L | 2C22 |
| | STEP L | 2C24 |
| | WRDATA L | 2C26 |
| | WR GATE L | 2C28 |
| | TRACK 0 L | 2C30 |

Table 3-1 (Cont.) HSC50/70 Signal Differences

| HSC50 Signal Name | HSC70 Signal Name | Slot/Backplane Pin Number |
|---|---|---|
| | WR PROT L | 2C32 |
| | RD DATA L | 2C34 |
| | SIDE SEL L | 2C36 |
| | READY L | 2C38 |

## 3.3 PROGRAM BUS

This section describes the Program Bus and the timing for its various cycles. The Program Bus is an asynchronous bus etched on the backplane between slots 1 and 2 on connector C.

Timing on the Program Bus for the HSC50 is about 10% faster than the Q-Bus implementation on the KDF11-A because some of the deskewing delays have been shortened. Timing on the Program Bus for the HSC70 is identical to the Q-Bus implementation on the KDJ11-A.

The Program Bus is a subset of the Q22 Bus, with several signals added. The Q22 Bus is a Q-Bus expanded to 22 addressing bits. In this chapter, where Program Bus differs from the Q-Bus, the differences will be pointed out.

**NOTE**

**You cannot plug a Q-Bus option into the backplane of an HSC.**

The Program Bus has 22 bits of addressing and a 16 bit data word. The address and data are multiplexed over the same lines during different times in the bus cycle. There are no parity lines for the address. The data lines have byte parity which is transmitted on lines 16 and 17 of the address/data lines during the data portion of the cycle.

The F-11/J-11s use the Program Bus to gain access to the following logic and buses:

- Program Memory (M.prog)
- Floppy Disk Controller (K.rx) (HSC70 only)
- Control Memory (through Control Bus cycles)
- Data Memory (through Data Bus cycles)
- Control Memory Windows
- Console terminal
- Auxiliary asynchronous ports (used for TU58 on HSC50)
- Bootstrap proms

- I/O page system registers which include:
  - PIOCSR
  - Switch/Display register
  - K Init and Status registers
  - Error Address registers
  - Serial Number register

**NOTE**

**You may want to refer to the block diagrams in Chapters 8 and 9 to see how the F-11 and J-11 use the Program Bus.**

However, the only devices directly on the Program Bus are the F-11/J-11, Program Memory, and K.rx. The data and address signals for the other devices and registers are interfaced to the Program Bus through the INBUS and OUTBUS drivers. On the HSC70 another set of transceivers, the BIBUS XCEIVERS, interface the serial ports and bootstrap proms to the data lines of the Program Bus.

The F-11/J-11 initiates a normal memory cycle on the Program Bus when addressing Control or Data Memories, but a separate bus cycle is performed on the Control or Data Bus *within the Program Bus cycle*. The extra bus cycle is transparent to the P.ioc/P.ioj.

The K.rx uses the Program Bus to access Program Memory using direct memory access (DMA).

**NOTE**

**This chapter covers only the activity on the Program Bus. For details about the P.ioc/P.ioj registers and interfaces see Chapters 8 and 9.**

### 3.3.1 Program Bus Master-Slave Relationship

Communication on the Program Bus is asynchronous but remains interlocked because of the master-slave relationship. The bus master has control of the bus. Only the P.ioc/P.ioj or K.rx may become bus master, but any device on the Program Bus may become slave.

The bus master initiates bus transactions, and the slave device responds by using bus protocol to acknowledge the transaction, receive data, or transmit data. Communication on the Program Bus is interlocked because the slave device must respond to the control signals issued by the bus master before the transfer is complete.

If communications on the Program Bus break down because the slave does not respond to the bus master, the master aborts the current bus cycle and traps to $004_8$ as a NXM error.

### 3.3.2 HSC50/70 Program Bus Differences

The HSC50/70 Program Buses have the following differences:

- The HSC70 bus has DMA capabilities; the HSC50 does not.
- The HSC50 and the HSC70 have the ability to swap banks for the Program Memory, but only the HSC70 can swap banks in Control Memory (M.ctl). The HSC50 also has circuitry to swap memory slots in the event that two memory boards are used.
- The HSC50 has PBASE OUT and PBASE IN signals which could be used for multiple memory boards. The HSC70 does not have the multiple memory module capability on the Program Bus.

NOTE

**There are no configurations of the HSC with multiple memory boards. The multiple memory board feature is described here for informational purposes only.**

### 3.3.3 Program Bus Signals

Table 3-2 summarizes the Program Bus signals on the HSC50/70.

**Table 3-2 Program Bus Signal Names**

| Quantity | Function | Bus Signal Mnemonic |
|---|---|---|
| 16 | Data/address lines | BDAL <15:00> |
| 2 | Parity/address lines | BDAL <17:16> |
| 4 | Address lines | BDAL <21:18> |
| 5 | Bus control lines | BSYNC<br>BDIN<br>BWTBT<br>BBS7<br>BRPLY |
| 3 | DMA control lines | BDMR*<br>BDMGO*<br>BSACK* |
| 2 | Interrupt control lines | BIRQ 4*<br>BIAK* |
| 4 | Processor control lines | BPOK<br>BDCOK<br>BINIT<br>PREFR CLK |
| 2 | Memory bank control lines | SWAP CMEM BANK*<br>SWAP PMEM BANK*<br>SWAP MEM BANK**<br>SWAP MEM BOARD** |
| 7 | Memory board base address | PBASE OUT <5:0>**<br>PBASE IN <5:0>**<br>PENA OUT** |

* HSC70 only

** HSC50 only

Table 3-3 describes the signals on the Program Bus which are identical to their counterparts on the Q-Bus.

Table 3-3 Program Bus Signals Which are Identical to Q-Bus Signals

| Signal Mnemonic | Signal Function |
|---|---|
| BDAL <15:00> L | Address lines for first part of bus cycle. Data lines for second part of cycle. |
| BDAL <21:18> L | Address lines 18 through 21. |
| BRPLY L | Slave devices assert BRPLY L in response to BDIN L or BDOUT L and (for non I/O page references) valid parity on BDAL <17:00> L, indicating their ability to transfer data according to the appropriate protocol. BRPLY L is also asserted during interrupt acknowledge (IAK) transactions. |
| BSYNC L | Synchronize - The bus master device asserts BSYNC L, indicating it has placed an address on the bus. The transfer is in process until the master negates BSYNC L. |
| BDOUT L | Data output - BDOUT, when asserted, implies that valid data is available on BDAL <15:00> L, and that an output transfer, with respect to the bus master device, is taking place. |
| BDIN L | Data input - Used for two types of bus operations. When asserted during BSYNC L time, BDIN L implies an input transfer with respect to the current bus master and requires a response, BRPLY L, from the addressed slave. Also, on the HSC70 the J-11 initiates interrupt service for the K.rx by asserting BDIN L followed by BIACK L. |
| BWTBT L | Write/Byte - BWTBT L is used in two ways to control a bus cycle. It is asserted during the address portion of a cycle to indicate that an output cycle rather than an input cycle is to follow. It is asserted during the Data portion of a write cycle to indicate a byte rather than a word transfer is to take place. |
| BBS7 L | Bank 7 select - When the bus master asserts an address, it asserts this signal to reference the I/O page. The I/O page address is placed on BDAL <12:00> L and the address bits from BDAL <21:13> are ignored. |
| BINIT L | Initialize - This signal provides a system reset. All devices on the bus are to return to a known, initial state. Registers are reset to zero, all bus drivers are disabled, and logic is reset to state 0, ready to be addressed for operations. |
| BIRQ 4 L* | Interrupt request priority level 4 is used by the K.rx to interrupt the J-11. |
| BIAK L* | Asserted by the J-11 to acknowledge an interrupt from K.rx. |
| BDMR L* | Direct Memory Access (DMA) request - The K.rx asserts this signal to request bus mastership. |

Table 3-3 (Cont.) Program Bus Signals Which are Identical to Q-Bus Signals

| Signal Mnemonic | Signal Function |
|---|---|
| BDMGO L* | Direct memory access grant - The J-11 asserts this signal to grant bus mastership to the K.rx in response to BDMR L. The K.rx acknowledges the grant by asserting BSACK L after BRPLY L and BSYNC L are both negated. |
| BSACK L* | The K.rx asserts this signal in response to the BDMGO L signal indicating it accepts bus mastership. The K.rx remains bus master until it negates BSACK L. |

* HSC70 only

Table 3-4 describes those signals that differ from their counterparts in the Q-Bus.

Table 3-4 Program Bus Signals Which Differ From Q-Bus Signals

| Signal Name | Description |
|---|---|
| PREFR CLK L | 66.7 KHz clock (15.0 microsecond period) is used by the Program Memory for timing Refresh Cycles. To simplify the design of the Memory Module, logic is provided on the P.ioc/P.ioj to eliminate any synchronization problems between this signal and BSYNC L at the Memory Module. |
| BDCOK H | Consists of three elements. The first is a *wakeup* circuit which keeps the P.ioc/P.ioj initialized until the +5.0 volt power is stabilized. The second is a dc voltage monitor circuit which checks for the existence of +5.0 volts, -5.2 volts, and +12.0 volts. If any voltage is missing, the P.ioc/P.ioj is held in the initialize state. When these two circuits are satisfied, the P.ioc/P.ioj is enabled and the Power lamp on the Operator Control Panel is turned on. The third component is a receiver input for the INIT switch on the Operator Control Panel; depressing this switch will initialize the P.ioc/P.ioj. |
| BDAL <17:16> L | During the data portion of a bus cycle, BDAL 17 contains the actual odd parity bit for BDAL <15:08> and BDAL 16 contains the odd parity bit for <07:00>. Parity on the Program Memory Bus is not encoded as on the Q-Bus. The parity bits are generated and received by the P.ioc/P.ioj or K.rx. |
| SWAP CMEM BANK H* | Used by the HSC70 Control Memory to select which half of the Control Memory DRAMs is used. |
| SWAP PMEM BANK H* | Used by the HSC70 Program Memory to swap the base address of its two banks. |
| SWAP MEM BOARD H** | Unused. This signal would have caused the first two Memory Boards in the HSC to swap base addresses if multiple Memory Modules had been used. |

Table 3-4 (Cont.) Program Bus Signals Which Differ From Q-Bus Signals

| Signal Name | Description |
|---|---|
| SWAP MEM BANK H** | Causes Program Memory to swap the base address of its two banks of Program Memory. |
| PBASE OUT <5:0>** | Asserted high on P.ioc. These signals were in the original design for more then one memory module. The PBASE OUT signals on the P.ioc become the PBASE IN signals to the Memory Module. The PBASE IN signals go through a base memory adder circuit on the Memory Module and then become PBASE OUT. |
| PBASE IN <5:0>** | Refer to PBASE OUT signals above. |
| PENA OUT** | Asserted low on P.ioc. This signal was in the original design for more then one memory module. |

*Signals only in the HSC70

** Signals only in HSC50

### 3.3.4 Program Bus Transactions

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

1. Most signals on the backplane etch are asserted low and referred to with a prefix character **B** and a suffix **L**. For example:
   - BSYNC L, BWTBT L, BBS7 L, BDAL <21:00> L
   - BPOK H and BDCOK H are asserted high
2. A signal name without a prefix or suffix is used when timing, polarity, and physical location are unimportant. For example:
   - SYNC for BSYNC
   - WTBT for BWBT
   - DAL <21:00> or the DAL lines for BDAL <21:00>
3. Receivers and drivers are considered part of the bus. Signal inputs to drivers are referred to with a prefix character **T** for transmit. For example:
   - TSYNC
   - TWTBT
   - TDAL <21:00>
4. Signal outputs of receivers are referred to with the prefix character *R* for received. For example:
   - RSYNC
   - RWTBT
   - RDAL<21:00>

   Program Memory signals feeding the receivers and drivers are labeled with a prefix character **PM** in the print sets. The prefix character *R* is used in this book.

5. If a reference is made to a Control or Data Bus address, a cycle will be requested, arbitrated, and performed on the respective bus. The timing diagrams do not take the other bus cycles into account. See Sections 3.5.2 and 3.6.2 for cycle information on the Control and Data Buses that occur within the Program Bus cycles.

The conventions listed in items 3, 4, and 5 above are used when timing is important. The timing is referenced to a receiver output or a driver input. For example:

> After receipt of the negation of RDIN, the slave negates its TRPLY (0 nanoseconds minimum, 8000 nanoseconds maximum). It must maintain valid data on its TDAL lines until 0 nanoseconds (minimum) after the negation of RDIN, and must negate its TDAL lines 100 nanoseconds (maximum) after the negation of its TRPLY.

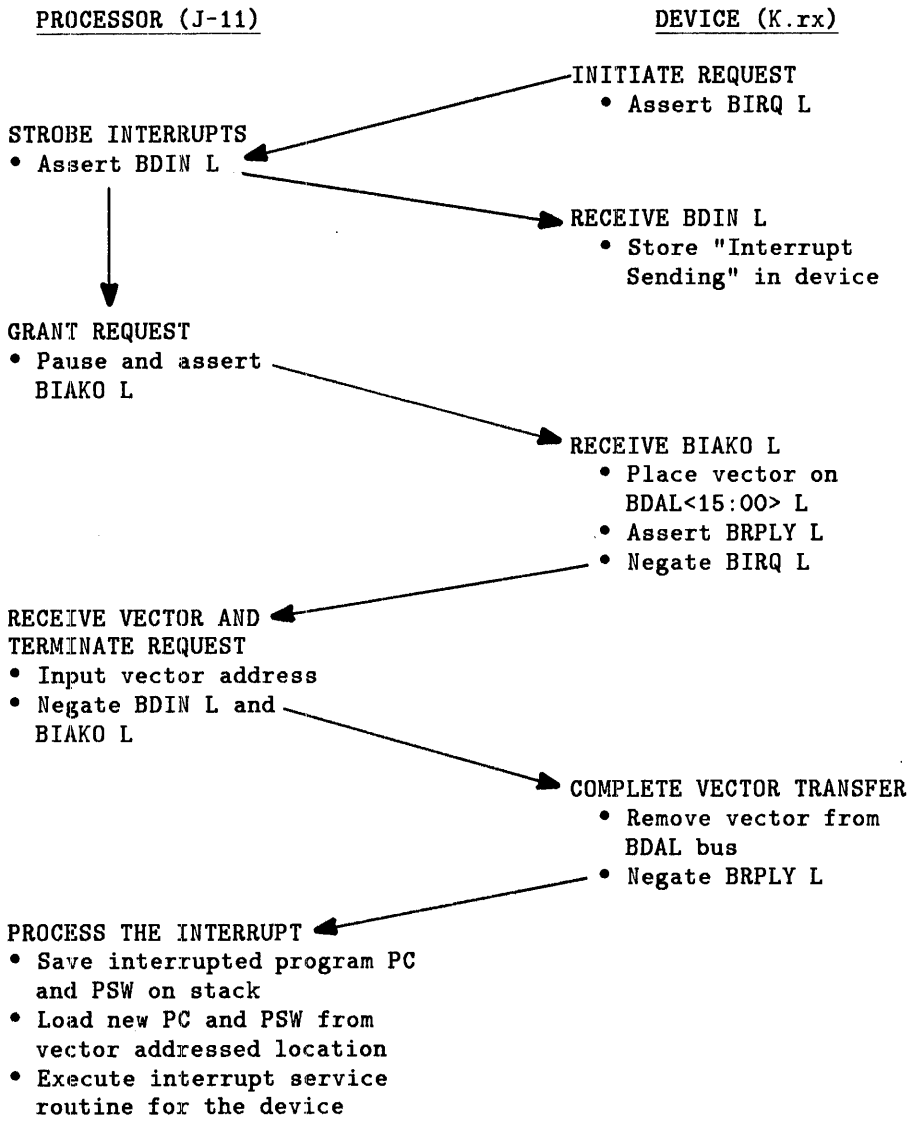
#### 3.3.4.1 Program Bus Interrupt Protocol (HSC70 Only)

The only interrupt used on the Program Bus is a level 4 interrupt generated by the K.rx. Control Bus and asynchronous line interrupts are contained within the P.ioj logic and do not use the BIRQ lines on the Program Bus.

The K.rx interrupts the P.ioj for service when it has completed work, is ready, or has an error. When the J-11 acknowledges the interrupt, the K.rx supplies an interrupt vector address ($230_8$) that the J-11 uses to retrieve a new processor status word (PSW) and a new program counter value (PC). The old PSW and PC are stored on the stack while the J-11 services the interrupt.

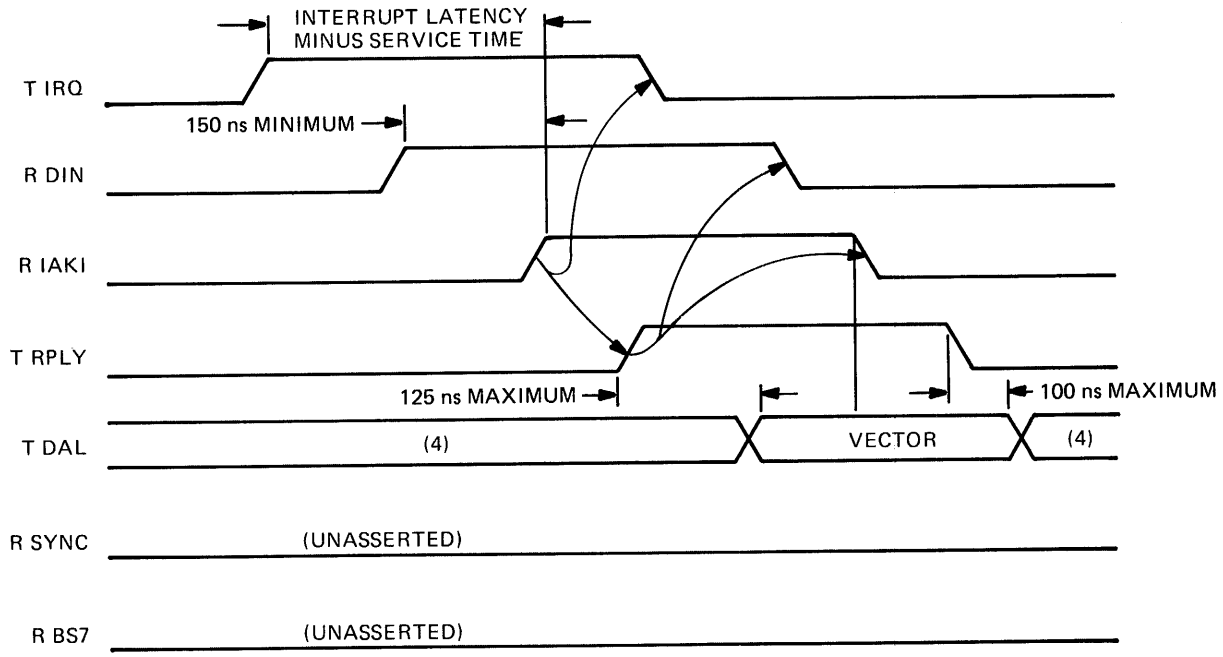
Interrupt protocol on the Program Bus has three phases: the interrupt request phase, the interrupt acknowledge phase, and the interrupt vector transfer phase. The operations performed by the processor and interrupting device are shown in Figure 3-1.

Figure 3-1 Program Bus Interrupt Request/Acknowledge Sequence




Interrupt protocol timing is shown in Figure 3-2. Following is a description of the timing during the interrupt.

**Figure 3-2 Program Bus Interrupt Protocol Timing**




NOTES:

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

CX–517A

The interrupt request phase starts when the K.rx asserts an interrupt request TIRQ4 L. During the interrupt acknowledge phase, the K.rx accepts the RIAKO signal on the leading edge of RDIN.

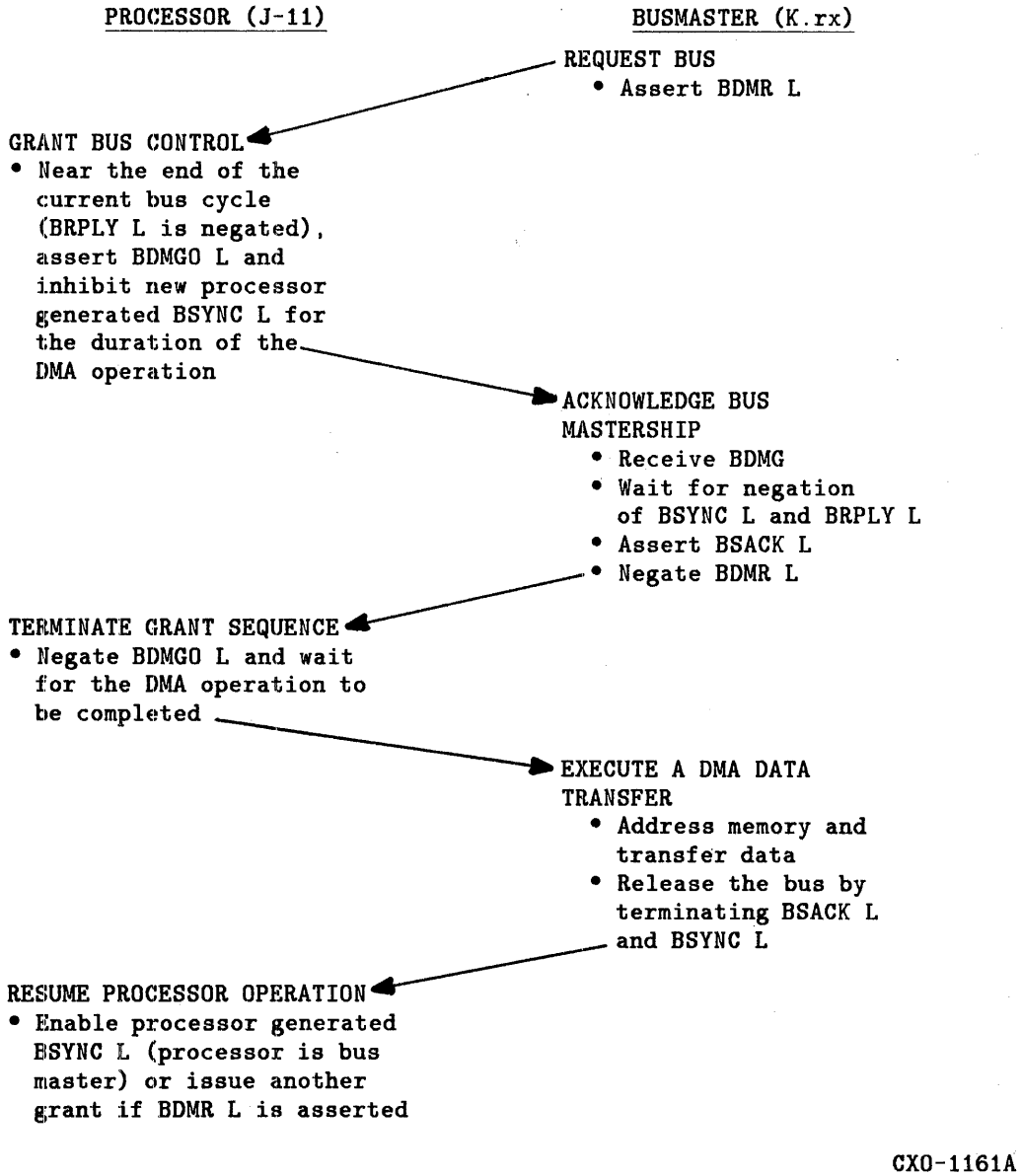
During the interrupt vector transfer phase, the K.rx asserts RRPLY. The vector address must be stable on BDAL <08:02> 125 nanoseconds (maximum) after TRPLY is asserted. The J-11 then receives the assertion of RRPLY within 200 nanoseconds (minimum). It then inputs the vector address and negates both TDIN and TIAKO. The K.rx negates TRPLY after the negation of RIAKI and removes the vector address from TDAL <08:02> 100 nanoseconds (maximum) after TRPLY negates. Because vector addresses must be between $000_8$ and $774_8$, the high order TDAL lines remain inactive.

### 3.3.4.2 Program Bus Mastership Protocol (HSC70 Only)

The DMA capability allows direct data transfers between the K.rx and memory. The K.rx becomes bus master by asserting BDMR L. The P.ioj then grants the bus to the K.rx by asserting BDMG L.

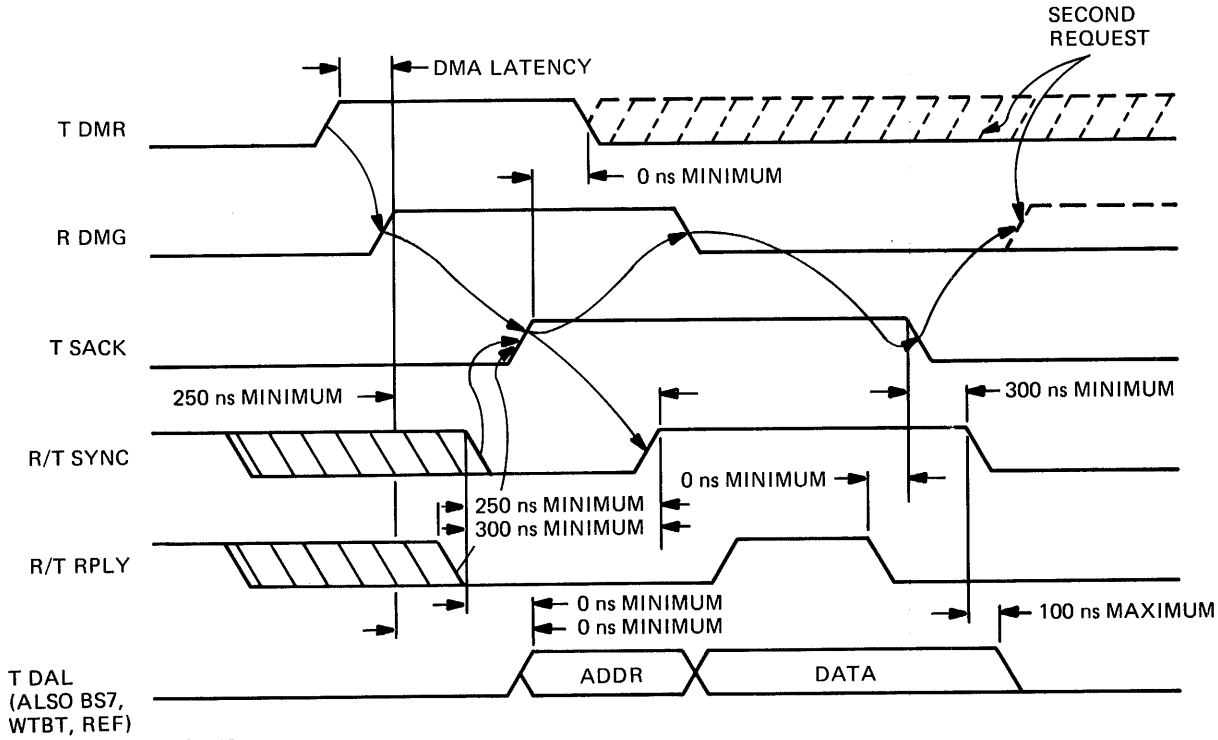
A DMA transaction is divided into three phases: the bus mastership acquisition, the data transfer, and the bus mastership relinquish. The protocol for a DMA transaction is shown in Figure 3-3.

Figure 3-3 Program Bus DMA Request/Grant Sequencing




The bus cycle timing is shown in Figure 3-4. Following is a description of the timing when the K.rx is requesting and releasing the Program Bus.

Figure 3-4 Program Bus Request/Grant Bus Cycle Timing




NOTES:

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

CX–520A

During bus acquisition, the K.rx requests the bus by asserting BDMR. The processor arbitration logic responds with TDMG. The K.rx asserts BSACK in response. If the processor does not receive a response to the DMA grant, it clears the grant and rearbitrates the bus.

During the data transfer phase, the K.rx keeps BSACK asserted. At the end of the data transfer, the K.rx relinquishes the bus by negating BSACK. TSYNC is negated 300 nanoseconds (maximum) after TSACK is negated.

### 3.3.4.3 Program Bus Data Transfers

The current bus master initiates a data transfer by placing the slave address and appropriate control signals on the bus. It then waits for a response from the slave before continuing with the transfer.

The Program Bus data word is 16-bits wide and is comprised of the low byte and the high byte. The low byte is BDAL <07:00>, and the high byte is BDAL <15:08>. Bits 16 and 17 supply odd parity. Bit 16 is the low byte parity bit and bit 17 is the high byte parity bit. Bits <21:18> are not active during the data portion of a cycle.

Program Bus addressing is accomplished using the BDAL lines and the BBS7 signal. If BBS7 is asserted during the address portion of the cycle, the bus master is addressing an I/O page address, and the address in the I/O page is found on BDAL lines <12:00>. If BBS7 is not asserted, the bus master is addressing memory, and the address is found on DAL <21:00>. The I/O addresses are in the top 8 Kbytes of addressing ($17770000_8$ through $17777777_8$).

The Program Bus supports five of the seven types of data transfers available on the Q-Bus. Table 3-5 lists the transfer types supported on the Program Bus.

**Table 3-5 Program Bus Data Transfer Types**

| Bus Cycle Mnemonic | Description | Function With Respect to Bus Master |
|---|---|---|
| DATI | Data word input | Read word |
| DATO | Data word output | Write word |
| DATOB | Data byte output | Write byte |
| DATIO | Data word input/output | Read word, modify, write word |
| DATIOB | Data word input/byte output | Read word, modify, write byte |

The address portion of the transfer is made up of an address set-up time and an address hold time. During the address set-up time the bus master does the following:

- Asserts TDAL<21:00> with the desired slave device address bits
- Asserts TBS7 if a device in the I/O page is being addressed
- Asserts TWTBT if the cycle is a data out bus cycle
- Asserts TSYNC 150 nanoseconds (minimum) after gating TDAL, TBS7 and TWTBT onto the bus

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

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

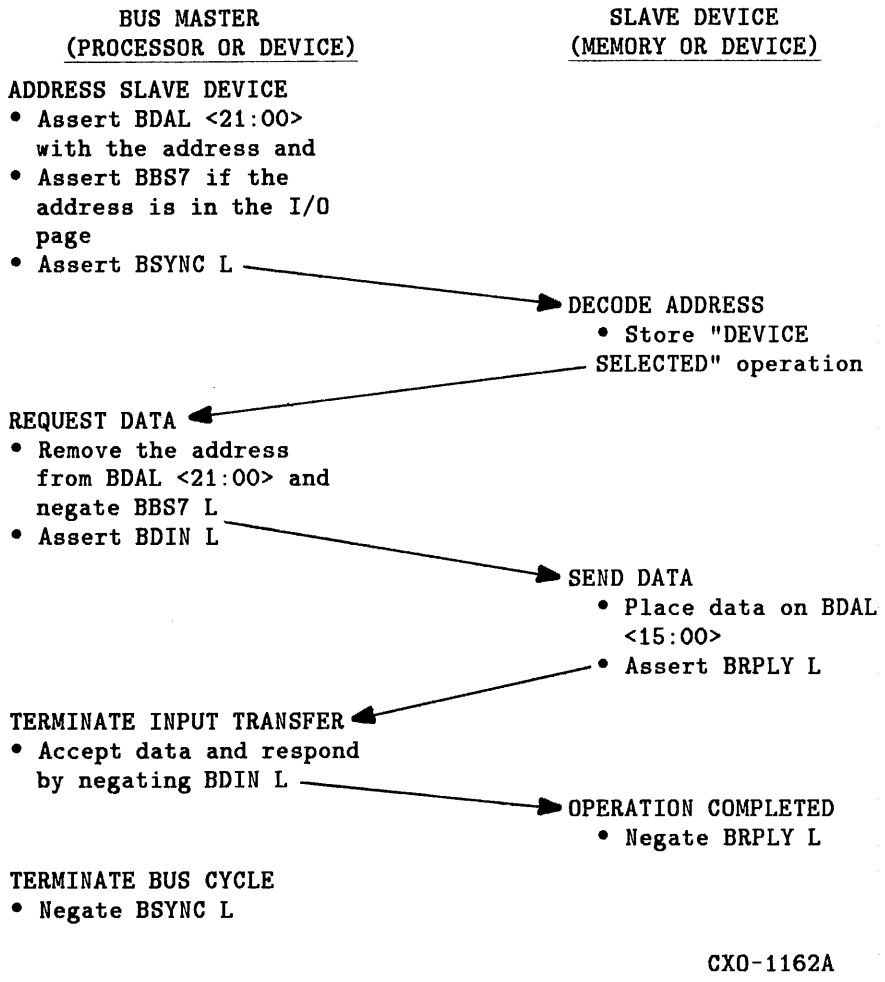
The following sections describe the cycles performed on the Program Bus. All cycles use the above addressing portion within the cycle. The addressing portion will not be repeated in the description of the cycle.

### 3.3.4.4 Program Bus DATI Cycle

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

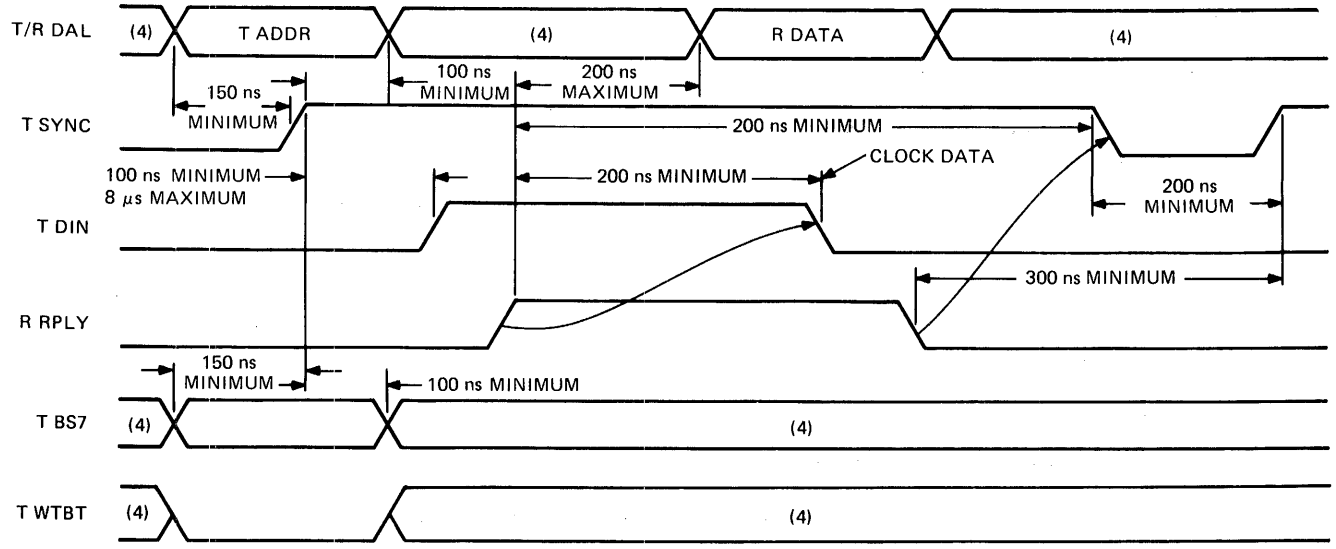
The DATI bus cycle operations are shown in Figure 3-5 in flow format.

**Figure 3-5 Program Bus DATI Flow Cycle**

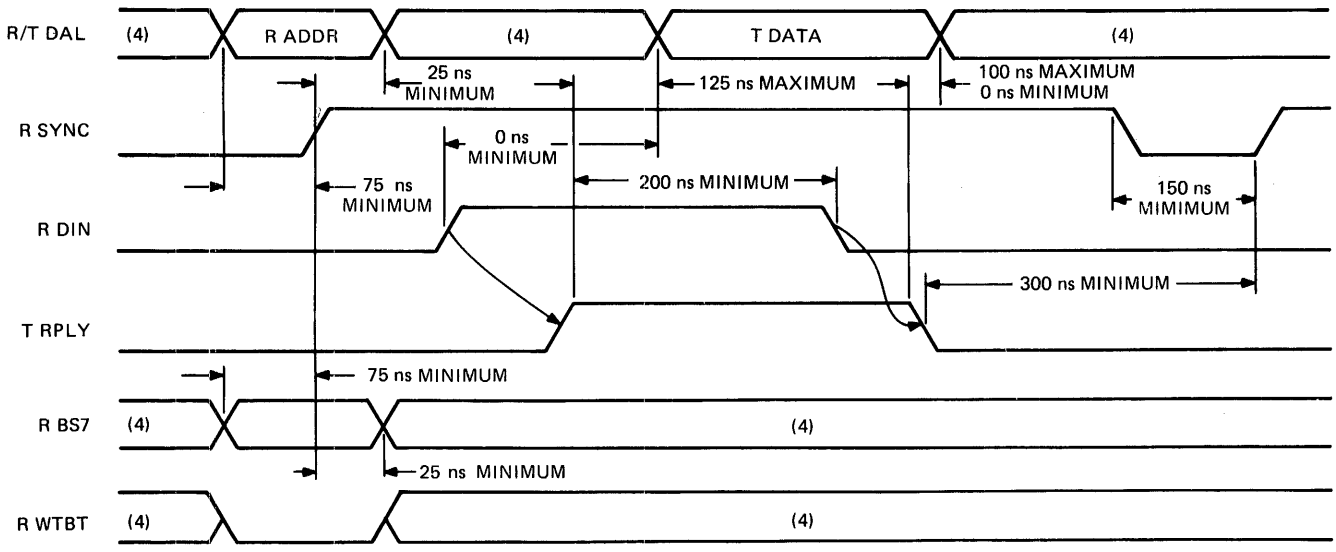



The timing diagram for a DATI Program Bus cycle is shown in Figure 3-6. Following is a description of the timing relationships.

## Figure 3-6 Program Bus DATI Timing Cycle




TIMING AT MASTER DEVICE




TIMING AT SLAVE DEVICE

NOTES:

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

CX–521A

During a DATI operation, the master:

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

When the slave receives DIN it:

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

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

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

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

- Negates TRPLY after RDIN is negated.
- Negates TDATA 100 nanoseconds (maximum) after it negates TRPLY.

To complete the transfer, the master:

- Negates TSYNC after RRPLY is negated.

#### 3.3.4.5 Program Bus DATO and DATOB Cycle

The DATO(B) bus cycle is a write operation because a 16-bit data word for DATO or 8-bit data byte for DATOB moves from the master to the slave.

Figure 3-7 gives the flow for a DATO(B) cycle.

Figure 3-7 Program Bus DATO or DATOB Flow Cycle

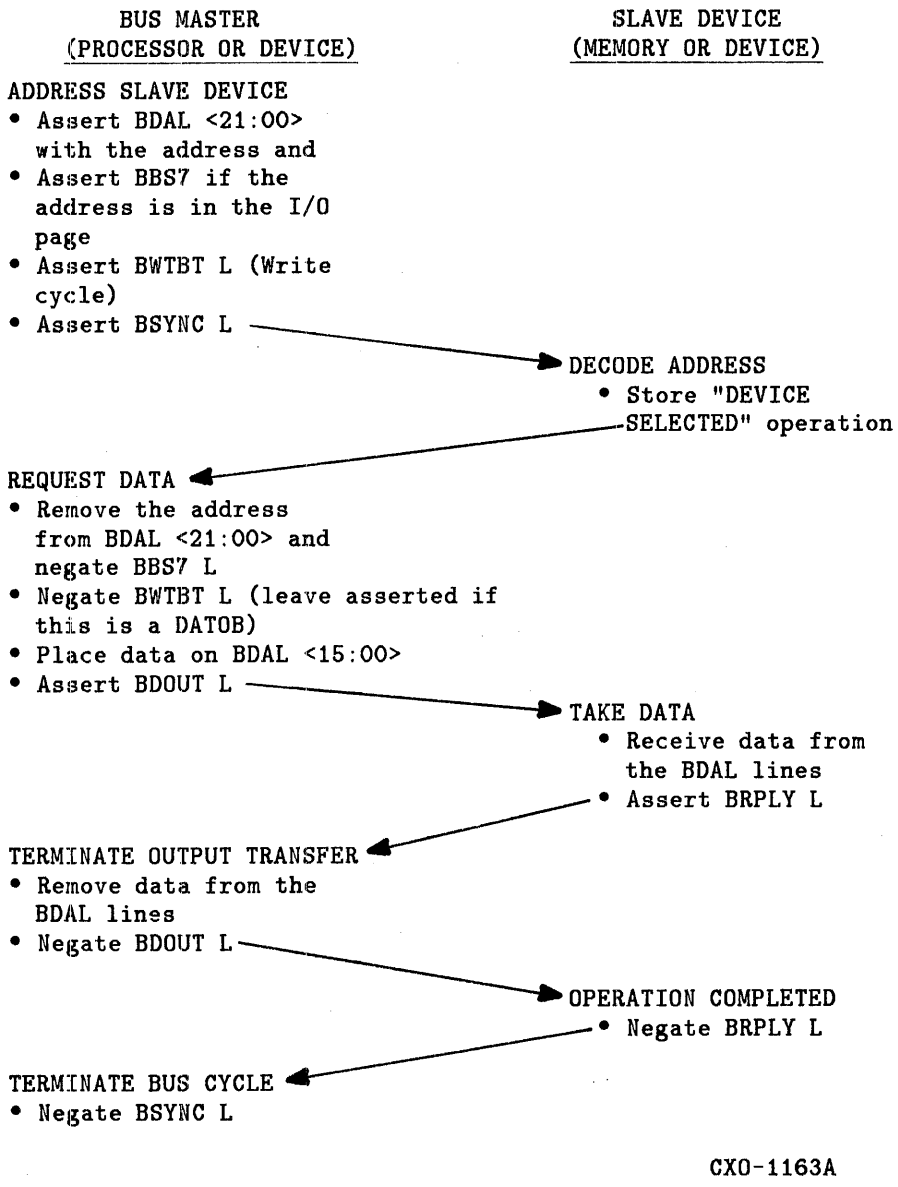
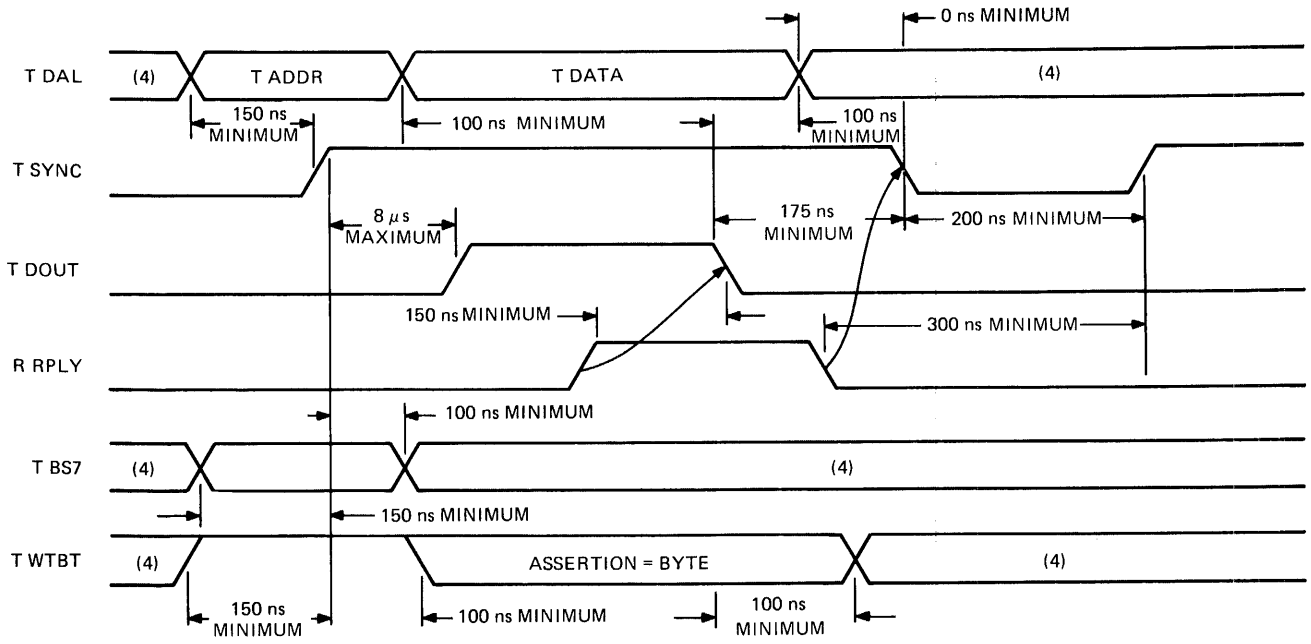
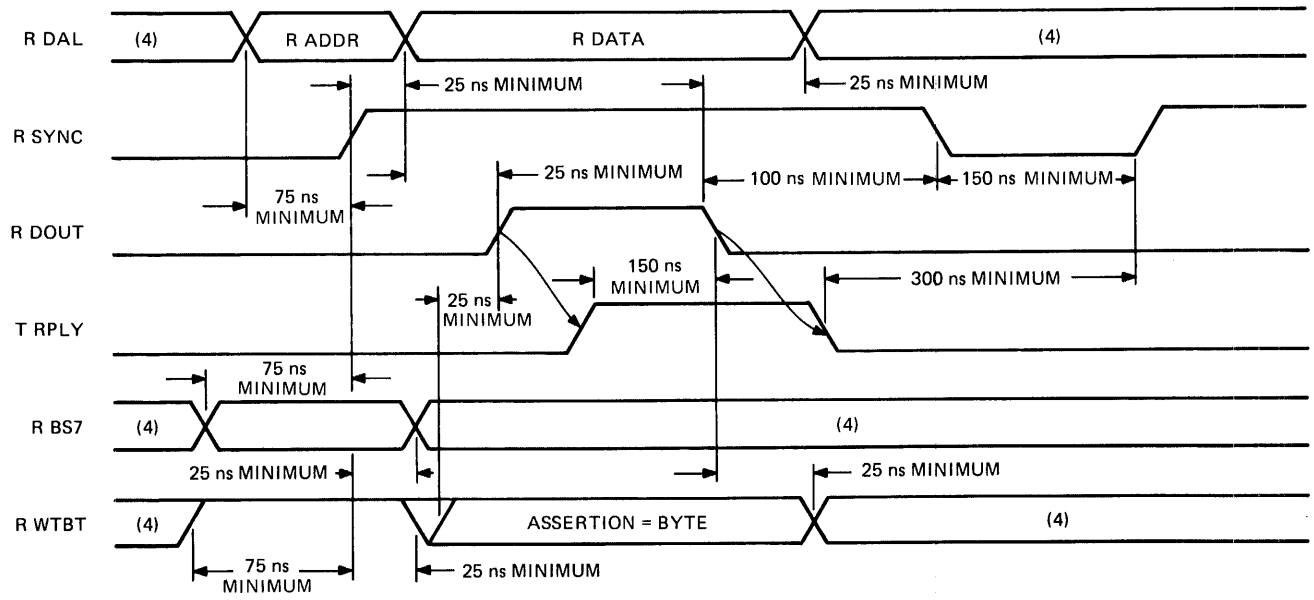



Figure 3-8 shows the timing diagram for the DATO(B) cycle. Following is a description of the timing relationships.

## Figure 3-8 Program Bus DATO or DATOB Timing Cycle




TIMING AT MASTER DEVICE




TIMING AT SLAVE DEVICE

NOTES:

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

CX–522A

The data transfer portion of a DATO(B) is made up of a data set-up time and a data hold time.

During the data set-up time, the master:

- Places the data on TDAL <15:00> 100 nanoseconds (minimum) after asserting TSYNC, and TDAL <17:16> supplies parity error detection.
- Negates TWTBT 100 nanoseconds (minimum) after TSYNC is asserted. Keeps TWTBT asserted for DATOB.
- Asserts TDOUT 100 nanoseconds (minimum) and 8 microseconds (maximum) after the TDAL and TWTBT lines are stable.

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

During the data hold time the following operations occur:

- The master receives RRPLY and negates TDOUT 150 nanoseconds (minimum) later.
- The master keeps the TDAL lines stable for at least 100 nanoseconds after negating TDOUT.
- The slave senses the negation of RDOUT and negates TRPLY.
- The master negates TSYNC after RRPLY is negated.

The DATO(B) bus cycle is now complete.

During the DATOB cycle, BWTBT is asserted and BDAL 0 L informs the slave to write either the upper or lower byte. If BDAL 0 L is not asserted, the lower byte will be written.

#### 3.3.4.6 Program Bus DATIO and DATIOB Cycles

The protocol for a DATIO(B) is identical to the addressing and data transfer portions of the DATI and DATO(B) bus cycles and is illustrated in Figure 3-9. After addressing the device, a DATI cycle is performed as explained earlier; however, BSYNC L is not negated. BSYNC L remains active for an output word or byte transfer. The bus master maintains at least 200 nanoseconds between BRPLY L negation during the DATI cycle and BDOUT L assertion. When the bus master negates BSYNC L, the cycle terminates as described for DATO(B).

## 3.4 BUS REQUESTORS

A requestor in the HSC is defined as a module which has the necessary hardware to request the Control or Data Bus. A requestor is any one of the following modules:

- I/O Control Processor (P.ioc/P.ioj)
- Port Processor Module (K.pli)
- Disk Data Channel (K.sdi)
- Tape Data Channel (K.sti)

There may be a maximum of ten requestors on the HSC70 or eight requestors on the HSC50. The requestor priorities on the Data Bus and Control Bus are determined by backplane slot location and the priorities of a given requestor are the same on both buses. (Control Bus requestor 5 is also Data Bus requestor 5.) Table 3-6 gives the slot and requestor number relationship.

Figure 3-9 Program Bus DATIO or DATIOB Bus Cycle

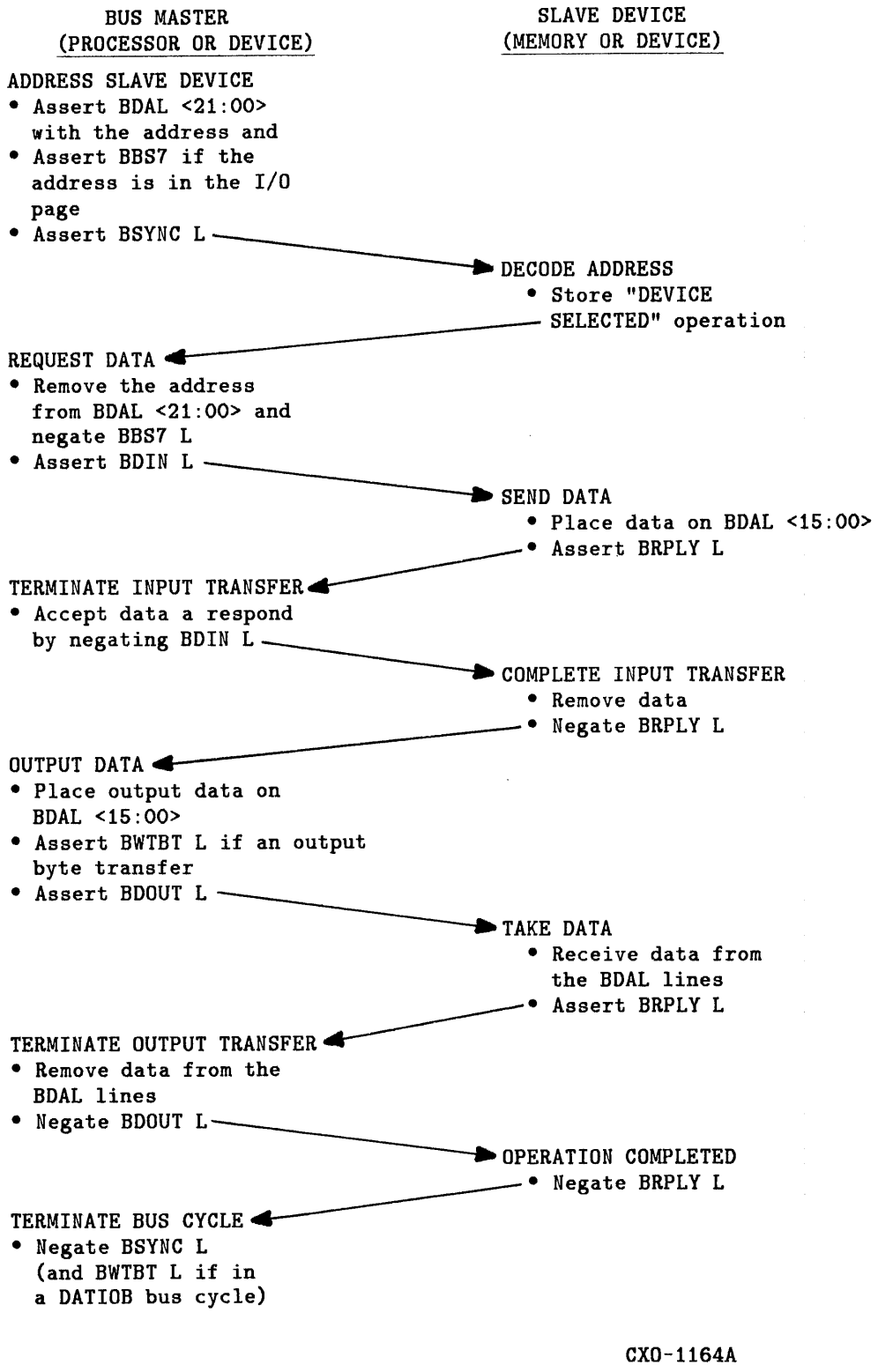


Table 3-6 Requestor Number Backplane Slots

| Slot Number | HSC50 Requestor | HSC70 Requestor | Requestor Type |
|---|---|---|---|
| 1 | 0 | 0 | P.ioc/P.ioj |
| 3 | - | 9 | K.sdi or K.sti |
| 4 | 7 | 8 | K.sdi or K.sti |
| 5 | - | 7 | K.sdi or K.sti |
| 6 | 6 | 6 | K.sdi or K.sti |
| 7 | 5 | 5 | K.sdi or K.sti |
| 8 | 4 | 4 | K.sdi or K.sti |
| 9 | 3 | 3 | K.sdi or K.sti |
| 10 | 2 | 2 | K.sdi or K.sti |
| 12 | 1 | 1 | K.pli |

All arbitration for the Control and Data Buses takes place on the P.ioc/P.ioj module in slot 1. In the HSC50, the P.ioc module gains ownership of the Control or Data Bus using the request and grant signals on the backplane the same as other requestors. In the HSC70, the P.ioj does not use any backplane pins to send and receive the request and grant signals to itself. Instead, the HSC70 P.ioj request and grant signals are contained entirely within the module and the extra backplane lines are redefined as signals for the two additional requestors.

The Memory Module is the only module on the Control and Data Buses that is not a requestor. The Port Link Module (LINK) and Port Buffer Module (PILA) do not connect to the Control and Data Buses.

## 3.5 CONTROL BUS

The Control Bus is a synchronous, 6.6 Mbyte/sec (300 nanosecond cycle) bus. The P.ioc/P.ioj, the K.pli, the K.sdi, and the K.sti modules use the Control Bus to access the control structures in Control Memory.

The Control Bus deals with word addresses and has 17 significant address bits with no parity. The data word on the Control Bus is 16 bits wide with two parity bits. For writing purposes, the data word is divided into two 9-bit bytes.

### 3.5.1 Control Bus Signals

Table 3-7 describes the Control Bus signals. All of the signals on the Control Bus, except CTIMING H, are asserted low.

**Table 3-7 Control Bus Signals**

| Signal | Description |
|---|---|
| CDATA <15:00> L | 16 data lines |
| CDATA HP L | Odd parity for CDATA <15:08> |
| CDATA LP L | Odd parity for CDATA <07:00> |
| CADR <16:00> L | 17 address lines (word address) |
| CWRTH L | Write enable line for CDATA <15:08,HP> |
| CWRTL L | Write enable line for CDATA <07:00,LP> |
| CREQ x L | 1 request line per requestor |
| CGRANT x L | 1 grant line per requestor |
| CACK L | Positive acknowledge from memory |
| CTIMING H | Timing for clocking data |
| CCYCLE <2:0> L | 3 encoded bits to indicate the desired cycle |
| CERR L | Indicates illegal CYCLE <2:0> line encoding |
| | **NOTE**<br>**x is 0 through 7 for the HSC50 and 1 through 9 for the HSC70** |

Termination of all Control Bus signals is provided on the P.ioc/P.ioj module. All lines are terminated with a resistor divider consisting of a 330 ohm pull-up resistor to +5 volts and a 680 ohm pull-down resistor to ground with the following exceptions:

- HSC50 only:
  - The signal CACK L is terminated with an equivalent 165 ohm pull-up resistor to +5 volts and a 680 ohm pull-down resistor to ground.
  - The signal CGRANT <7:0> L and CREQ <7:0> L are terminated with a 390 ohm pull-up resistor to +5 volts and a 620 ohm pull-down resistor to ground.
- HSC70 only—the signals CGRANT <9:1> L and CREQ <6:2> L are terminated with a 390 ohm pull-up resistor to +5 volts and a 620 ohm pull-down resistor to ground.

### 3.5.2 Control Bus Cycles

A Control Bus cycle is defined as the time a requestor has its associated CGRANT signal asserted from the arbitrator on the P.ioc/P.ioj module. Arbitration is performed on fixed 300 nanosecond boundaries. Once the requestor receives a grant from the arbitrator, it has control of the Control Bus for one 300 nanosecond cycle. No requestor may attempt to obtain two successive Control Bus cycles except during a Control Bus lock cycle. Each requestor must negate its associated CREQ L for at least one bus cycle after receiving an asserted CGRANT x L.

When granted access to the Control Bus, a requestor may perform one of four types of cycles. The cycles are determined by the code in CCYCLE <2:0>. Table 3-8 shows the coding for each of the cycles.

**Table 3-8 Control Bus Cycle Encoding**

| CCYCLE <2:0> 2 1 0 | Bus Cycle Definition |
|---|---|
| 0 0 1 | Normal control memory reference (read or write) |
| 0 1 0 | Interrupt cycle |
| 1 0 0 | Lock (read-and-set) cycle |
| 1 1 1 | NMA cycle |

The arbitrator contains logic to detect valid CCYCLE codes. If an illegal CCYCLE code (000, 011, 101, or 110) is detected on any Control Bus cycle, the CERR L line will be asserted by this logic on the arbitrator. The arbitrator then prevents the alteration of Control Memory by inhibiting the assertion of CTIMING H. Each requestor samples CERR L after performing a Control Bus cycle to verify proper operation of the CCYCLE lines.

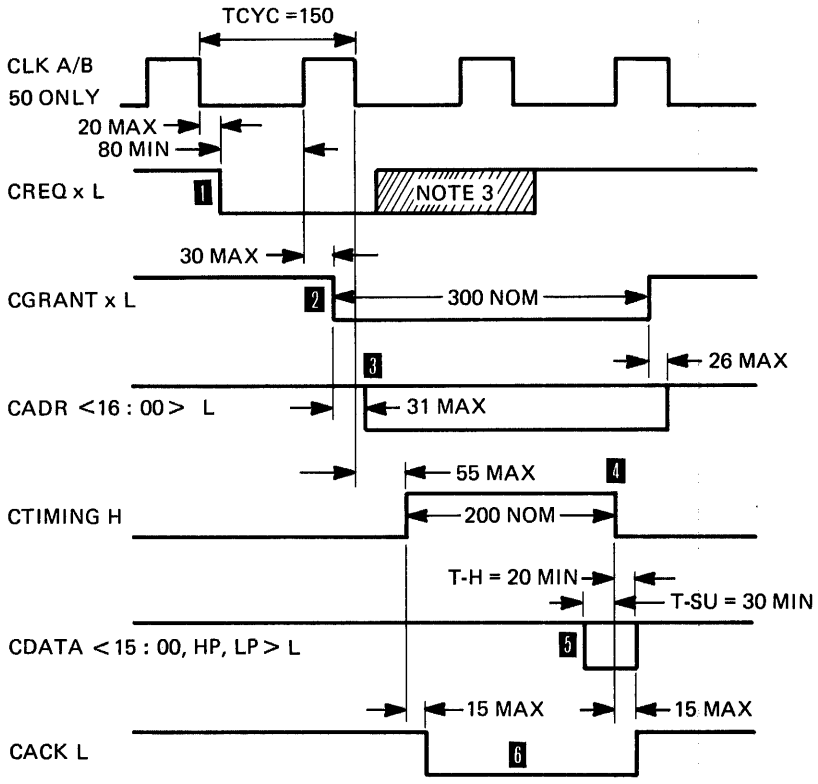
The following sections describe each type of cycle.

#### 3.5.2.1 Control Bus Read Cycle

Once a requestor is granted the Control Bus, a read cycle will be performed if the requestor puts a normal cycle code (001) on the CCYCLE lines and negates both CWRTH L and CWRTL L during the cycle. CWRTH L and CWRTL L are the only signals that distinguish a read cycle from a write cycle. If either is asserted, a write cycle will be performed instead of a read cycle. The data word is read from Control Memory on CDATA <15:00,HP,LP>.

The following numbered list refers to events in Figure 3-10 and gives a description of the timing during an HSC50 Control Bus read cycle.

Figure 3-10 HSC50 Control Bus Read Timing




NOTES: 1. ALL VALUES IN NANOSECONDS
2. ONE CHARACTER = 10 NANOSECONDS (APPROX.)
3. CREQ x L MAY BE NEGATED ANY TIME AFTER THE ASSERTION OF CGRANT x L
4. X = 0 THROUGH 7 FOR THE HSC50
X = 1 THROUGH 9 FOR THE HSC70

CX–1101A

1 When a cycle is desired, a requestor asserts its request line (CREQ x L) no later than 20 nanoseconds after the negative transition of the backplane clock and awaits the result of arbitration (performed on the P.ioc/P.ioj module).

2 When successful, the selected requestor receives an asserted grant line (CGRANT x L). The requestor now has control of the Control Bus until the grant is negated.

3 The requestor must immediately enable its CADR <16:00> L, and CCYCLE <2:0> L drivers onto the bus and leave them enabled for the duration of the grant. CCYCLE <2:0> must be equal to 1 for a read cycle.

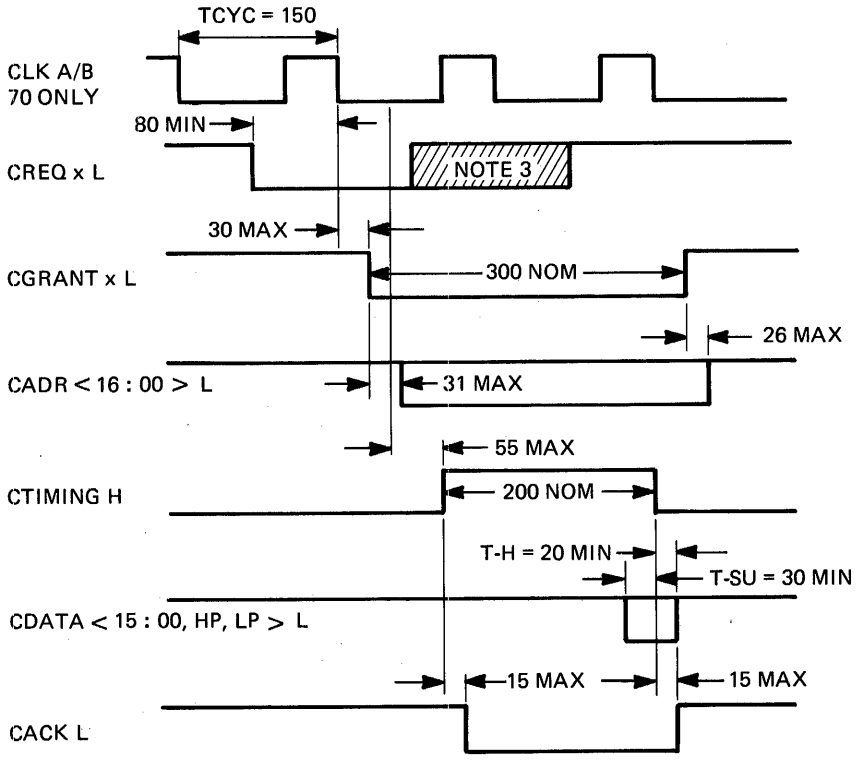
4 The requestor uses the negation of CTIMING H to strobe the data into its receiver latches.

5 The data from Control Memory is gated onto the CDATA <15:00,HP,LP> L lines.

6 During the bus cycle, the requestor must sample CACK L (to determine a successful memory reference). Note that CACK L is derived from CTIMING H.

The timing relationship for the HSC70 Control Memory cycle is different than the HSC50. Figure 3-11 shows the HSC70 Control Bus Timing. The differences are that CGRANT is qualified 50 nanoseconds later than on the HSC50 and the CGRANT signal is qualified on a different edge of CLK A/B. This is true for all HSC70 Control Memory cycles.

**Figure 3-11 HSC70 Control Bus Read Timing**




NOTES:
1. ALL VALUES IN NANOSECONDS
2. ONE CHARACTER = 10 NANOSECONDS (APPROX.)
3. CREQ x L MAY BE NEGATED ANY TIME AFTER THE ASSERTION OF CGRANT x L
4. X = 0 THROUGH 7 FOR THE HSC50
   X = 1 THROUGH 9 FOR THE HSC70

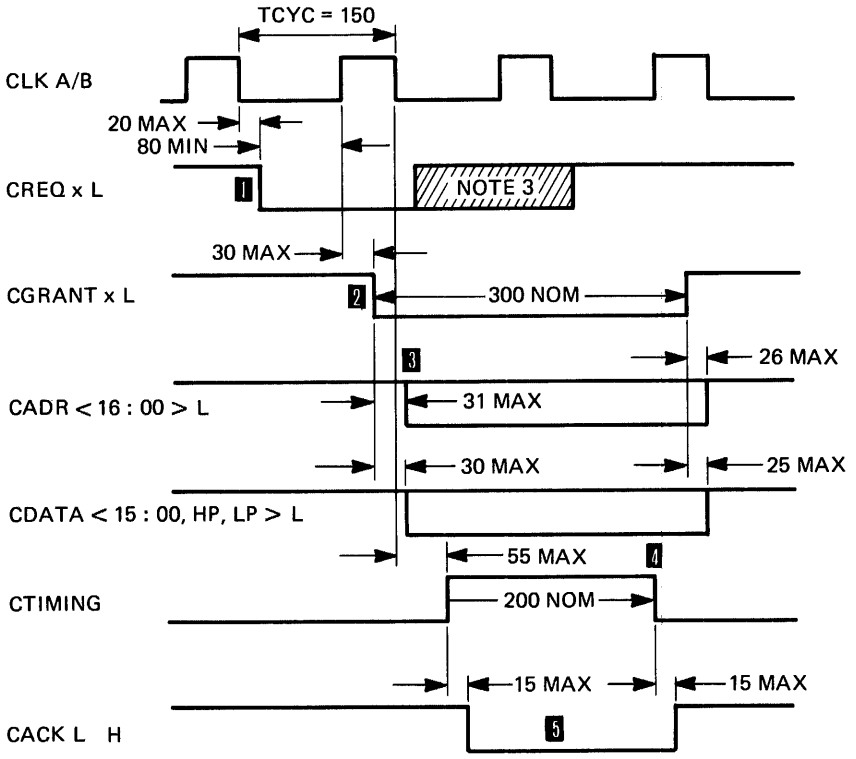
CX–1102A

### 3.5.2.2 Control Bus Write Cycle

Once a requestor is granted the Control Bus, a write cycle will be performed if the requestor puts a normal cycle code (001) on the CCYCLE lines and asserts either CWRTH L or CWRTL L during the cycle. If CWRTH L is asserted, CDATA <15:08> L will be written to Control Memory. If CWRTL L is asserted, CDATA <07:00> will be written to Control Memory. The upper and lower 9-bit bytes are written independently, though at the same time with the same control signals. CWRTH L and CWRTL L are the only signals that distinguish a write cycle from a read cycle.

The following numbered list refers to events in Figure 3-12 and gives a description of the timing during a Control Bus write cycle. The timing for an HSC70 is different for CLK A/B and CREQ x L (Figure 3-11).

**Figure 3-12 HSC50 Control Bus Write Cycle**




NOTES:
1. ALL VALUES IN NANOSECONDS
2. ONE CHARACTER = 10 NANOSECONDS (APPROX.)
3. CREQ x L MAY BE NEGATED ANY TIME AFTER THE ASSERTION OF CGRANT x L
4. X = 0 THROUGH 7 FOR THE HSC50
   X = 1 THROUGH 9 FOR THE HSC70

CX-1103A

1. When the cycle is desired, the requestor asserts its request line (CREQ x L) no later that 20 nanoseconds after the negative transition of the backplane clock and awaits the result of arbitration (performed on the P.ioc/P.ioj module).
2. When successful, the selected requestor receives an asserted grant line (CGRANT x L). The requestor now has control of the Control Bus until the grant is negated.
3. The requestor must immediately enable its CADR <16:00> L drivers, CCYCLE <2:0> L drivers, CDATA <15:00,HP,LP> L, and CWRTH/L L drivers onto the bus and leave them enabled for the duration of the grant. CCYCLE <2:0> L must be equal to 1 for a write operation. CWRTH L will be asserted if CDATA <15:08,HP> are to be written, and CWRTL L will be asserted if CDATA <07:00,LP> are to be written.
4. During the bus cycle, the requestor must sample CACK L (to determine a successful memory reference). Note that CACK L is derived from CTIMING H, so special attention must be paid when this signal is sampled.

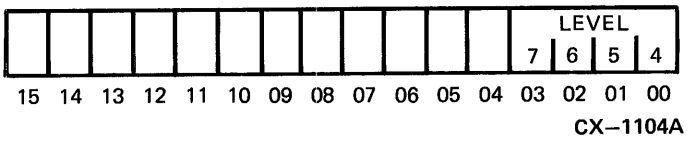
### 3.5.2.3 Control Bus Interrupt Cycle

The only means for any requestor to interrupt the P.ioc/P.ioj is through the Control Bus interrupt cycle. The K sends an interrupt mask that informs the P.ioc/P.ioj the BR level on which it is interrupting. The P.ioc/P.ioj then turns the interrupt level into an interrupt vector which it passes to the P.ioc/P.ioj through the interrupt logic. The P.ioc/P.ioj uses the interrupt vector as an entry into an interrupt service routine. Multiple interrupts may be generated on a single cycle.

Once it has been granted use of the Control Bus, the requestor performs an interrupt cycle by driving the interrupt code (010) onto the CCYCLE lines. The requestor drives the interrupt mask onto the data lines and the cycle is performed as a normal write-to-memory (though there will be no memory access). The P.ioc/P.ioj accepts the data lines but ignores parity. The CADR lines are unused during the interrupt cycle. The arbitrator is responsible for asserting CACK L at the proper time to prevent requestors from getting a non-existent memory error. The arbitrator disables memory access by not asserting CTIMING L.

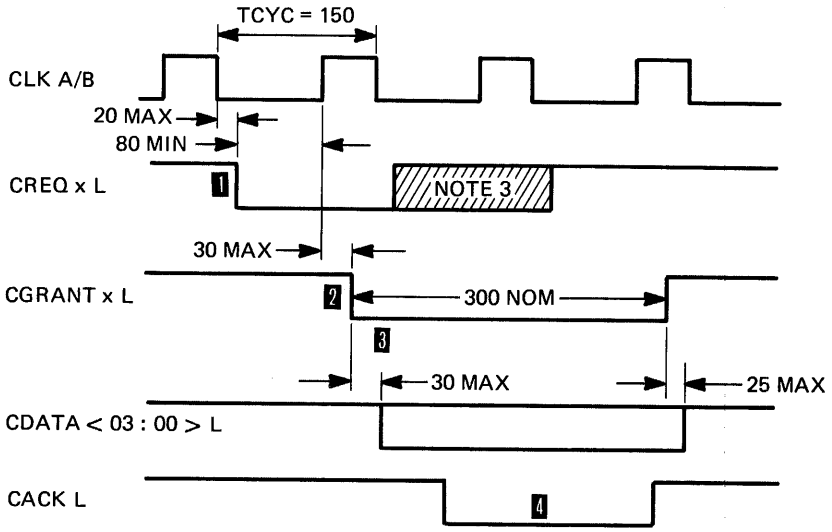
Figure 3-13 gives the bit definitions for the interrupt mask.

**Figure 3-13 P.ioc/P.ioj Control Bus Interrupt Mask**




The following numbered list refers to the events in Figure 3-14 and gives a description of the timing during a Control Bus interrupt cycle.

Figure 3-14 Control Bus Interrupt Cycle




NOTES: 1. ALL VALUES IN NANOSECONDS
2. ONE CHARACTER = 10 NANOSECONDS (APPROX.)
3. CREQ x L MAY BE NEGATED ANY TIME AFTER THE ASSERTION OF CGRANT x L
4. X = 0 THROUGH 7 FOR THE HSC50
X = 1 THROUGH 9 FOR THE HSC70

CX–1105A

1 When a cycle is desired, the requestor asserts its request line (CREQ x L) no later than 20 nanoseconds after the negative transition of the backplane clock and awaits the result of arbitration (performed on the P.ioc/P.ioj module).

2 When successful, the selected requestor receives an asserted grant line (CGRANT x L). The requestor now has control of the Control Bus until the grant is negated.

3 The requestor must immediately enable its CCYCLE <2:0> L drivers, CDATA <15:00> L, and CWRTL L drivers onto the bus and leave them enabled for the duration of the grant. CCYCLE <2:0> L must contain the interrupt code (010) and CDATA <03:00> L must have one or more bits set to interrupt at the desired BR level.

4 During the bus cycle, the requestor must sample CACK L (which has been generated by the arbitrator).

5 The interrupt logic on the P.ioc/P.ioj module turns the interrupt level into an interrupt vector and services the associated interrupt queue when the interrupt is recognized.

### 3.5.2.4 Control Bus Lock Cycle

The requestors use the Control Bus lock cycle to allocate a software structure in Control Memory so that another requestor may not access that structure. The requestor keeps a lock on a structure until it finishes its operation.

When the $100_8$ code is asserted on CCYCLE <2:0> during the Control Bus cycle, a lock cycle will be performed at the designated address. The lock cycle performs a read cycle for the requestor, but the contents of the addressed memory location are rewritten *by the arbitrator* with a value of $100000_8$. The writing of the addressed memory location occurs during the next successive cycle on the bus after the read cycle (under control of the arbitrator). These two cycles cannot interrupted by any requestor.

Refer to Figure 3-15 for timing relationships during a lock cycle.

Note that the GRANT line will remain asserted during both bus cycles, so that the requestor must leave the address on the bus for the write-to-memory portion of the cycle. Thus the requestor must not detect the completion of the request until CGRANT x L is negated. The requestor is responsible for clocking the data from the read-from-memory part of the request into its receiver latches with the first negation of CTIMING H during the cycle, just as in a normal read cycle.

Since a lock cycle appears to the memory to be a normal read cycle followed by a normal write, the memory need not do anything special to accommodate this feature. However, as with any normal cycle, the memory must respond with the assertion of CACK L, and the requestor must check for the assertion of CACK L.

### 3.5.2.5 Control Bus Non-Memory Access (NMA) Cycle

The requestors use the non-memory access cycle for diagnosing the operation of the CADR and CDATA lines.

Requestors execute an NMA cycle after being granted the Control Bus by putting the $111_8$ code out on the CCYCLE lines. All timing signals are used the same as for a normal cycle, except that the arbitrator inhibits the assertion of CTIMING H. Without the assertion of CTIMING H, all Control Memory is inhibited. A requestor may execute either a read NMA cycle or a write NMA cycle.

The arbitrator asserts CACK L during a Control Bus NMA cycle to prevent undesirable non-existent memory errors from occurring in any requestor.

### 3.5.2.6 Control Bus Refresh Cycle

Since dynamic RAMS are used in Control Memory, a refresh cycle is necessary. The refresh signals reserve a Control Bus cycle for all the RAMS on the Memory Module to perform a refresh. The refresh cycle occurs every 15 microseconds and has the highest priority on the Control Bus.

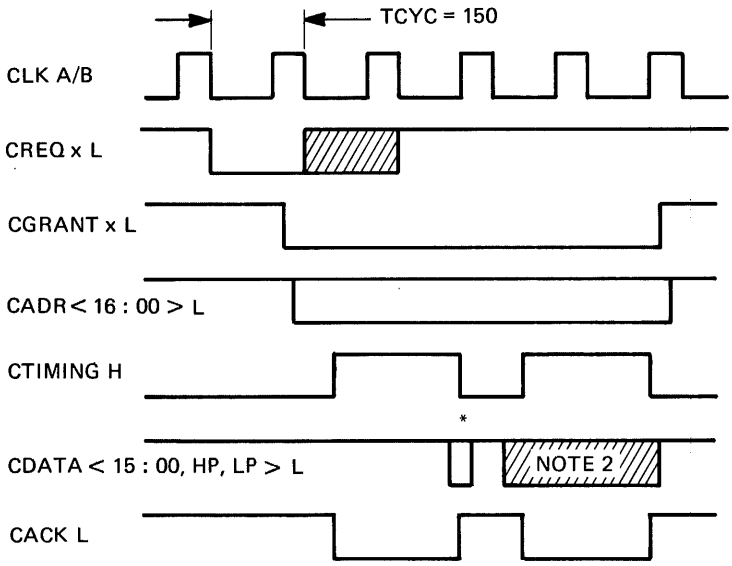
On the HSC70, only one signal (CREFR GRANT L) is present on the backplane for the refresh cycle. The arbitration request and the clock signal do not go onto the backplane. On the HSC50, CREFR REQ L and CREFR CLK L also are driven onto the backplane.

The P.ioc/P.ioj generates CREFRESH CLK H as a 66.7 KHz square wave (15 microsecond period) to indicate when a refresh cycle is to be performed. On the HSC70 this signal is used to directly request the Control Bus refresh cycle.

On the HSC50, the signal goes onto the backplane as CREFR CLK L on pin B86. In slot one on the HSC50, pin B86 is tied to pin B87 and comes back into the P.ioc module as CREFR REQ L. The Control Bus arbitrator then issues a refresh grant signal (CREFR GRANT L), with the same timing as any of the CGRANT n L signals.

The Control Memory logic uses the assertion of CREFR GRANT L to initiate an internal refresh cycle. The arbitrator may not assert any CGRANT signals while CREFR GRANT L is asserted.

Figure 3-15 Control Bus Lock Cycle




NOTES:
1. REFER TO FIGURES 3-10 AND 3-12 FOR ACTUAL TIMES
2. DATA SUPPLIED BY ARBITRATOR DURING WRITE PORTION OF CYCLE
3. * = REQUESTOR IS RESPONSIBLE FOR LATCHING DATA READ AT THIS POINT
4. X = 0 THROUGH 7 FOR HSC50
   X = 1 THROUGH 9 FOR HSC70
5. CCYCLE LINES MUST BE DRIVEN FOR THE ENTIRE LOCK CYCLE
6. THE MEMORY DRIVES THE DATA LINES DURING THE FIRST HALF OF THE CYCLE; THE ARBITRATOR DRIVES THE WRITE STROBES AND THE DATA LINES DURING THE SECOND HALF

CX–1106A

## 3.6 DATA BUS

The Data Bus is a synchronous, 13.3 Mbyte/second (150 nanosecond cycle) bus etched on the backplane from slot 1 to slot 12 within the A connector blocks. It is used by the P.ioc/P.ioj, the K.pli, the K.sdi, and the K.sti modules to access Data Memory (M.data).

The Data Bus deals with word addresses and has 18 significant address bits with no parity bits. The data word is 16 bits wide with 2 parity bits and, for writing purposes, is divided into two 9-bit bytes.

All arbitration for the Data Bus takes place on the P.ioc/P.ioj module in slot one. In the HSC50, the P.ioc module gains ownership of the Data Bus by using the request and grant signals on the backplane the same as other requestors. In the HSC70, the P.ioj DREQUEST and DGRANT signals are on the module and the extra backplane lines are redefined as signals for the two additional requestors.

### 3.6.1 Data Bus Signals

Table 3-9 lists and describes the Data Bus signals. All of the signals on the Data Bus, except DNMA, are asserted low.

**Table 3-9 Data Bus Signals**

| Signal | Description |
|---|---|
| DDATA <15:00> L | 16 data lines |
| DDATA HP L | High byte parity for DDATA <15:08> lines |
| DDATA LP L | Low byte parity for DDATA <07:00> lines |
| DADR <17:00> L | 18 address lines (word address) |
| DWRTH L | Write line for upper byte |
| DWRTL L | Write line for lower byte |
| DREQ x L | 1 request line per requestor |
| DGRANT x L | 1 grant line per requestor |
| DTIMING L | Timing reference from arbitrator for write cycles |
| DACK L | Acknowledge line (positive acknowledge from memory) |
| DNMA H | Non-memory-access line (used for diagnostic purposes) |
| | **NOTE**<br>**x is 0 through 7 for the HSC50 and 1 through 9 for the HSC70** |

The P.ioc/P.ioj module provides the termination for all Data Bus signals. All signals are terminated with a 330 ohm pull-up resistor to +5 volts and a 680 ohm pull-down resistor to ground, with the following exceptions:

- HSC50:
  - DREQUEST x L signals are terminated with 3.9K ohm pull-up resistors to +5 volts
  - DNMA H is terminated with an equivalent 165 ohm pull-up resistor to +5 volts and a 680 ohm pull-down resistor to ground
- HSC70:
  - DNMA H is terminated with a 200 ohm pull-up resistor to +5 volts
  - DWRTH L and DWRTL L are terminated with a 220 ohm pull-up resistor to +5 volts
  - DREQ <9:0> L are terminated with a 1K ohm pull-up resistor to +5 volts

### 3.6.2 Data Bus Cycles

A Data Bus cycle is defined as the time a requestor has its associated DGRANT signal asserted by the arbitrator on the P.ioc/P.ioj module. Arbitration is performed on fixed 150-nanosecond boundaries. Once the requestor receives a grant from the arbitrator, it has control of the Data Bus for one 150-nanosecond cycle. No requestor may attempt to obtain two successive Data Bus cycles. Each requestor negates DREQ x L for at least one clock cycle after receiving an asserted DGRANT x L.

When granted access to the Data Bus, a requestor may perform one of the following cycles:

- Read
- Write
- Non-memory access (NMA)

The following sections describe each type of cycle.

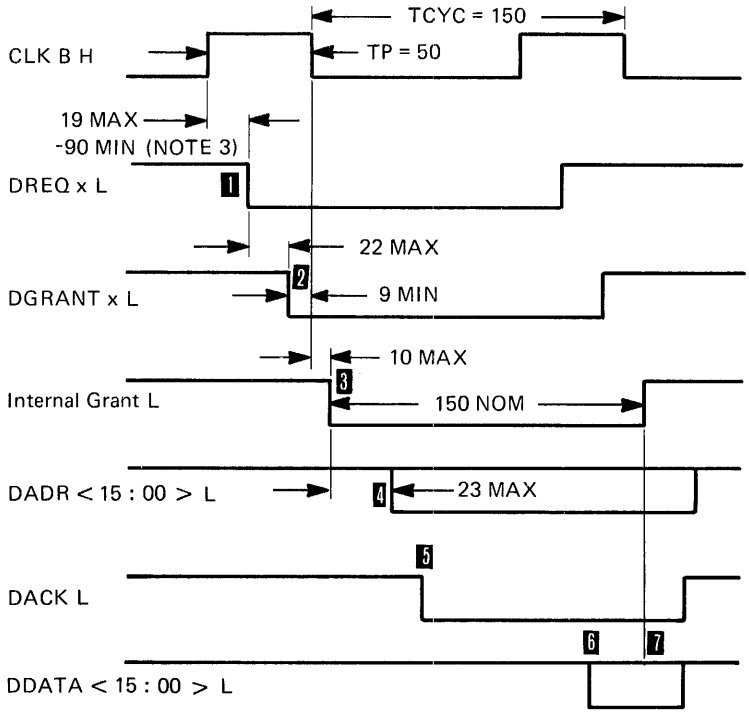
#### 3.6.2.1 Data Bus Read Cycle

Once a requestor has been granted the Data Bus, a read cycle will be performed if DWRTH L and DWRTL L are negated while the requestor asserts DADR <17:00>. The diagnostic signal DNMA H must also be negated for Data Memory to place data from the specified address on the DDATA <15:00,HP,LP> lines.

The following numbered list refers to the events in Figure 3-16 and gives a description of the timing during a normal Data Bus read cycle.

1. When a cycle is desired, a requestor asserts its request line (DREQ x L) no later than 19 nanoseconds after the positive transition of CLK B H and awaits the result of arbitration (performed on the P.ioc/P.ioj module).
2. When successful, the selected requestor receives an asserted grant line (DGRANT x L) at least 9 nanoseconds before the negative transition of CLK B H.
3. The requestor clocks this asserted grant directly into a flop with this negative transition. This synchronized grant defines a Data Bus cycle for that requestor, and it has possession of the Data Bus until the grant is dropped.
4. Using this internally synchronized grant, the requestor then asserts the desired address onto the appropriate lines and negates DNMA H, DWRTH L, and DWRTL L. These signals must remain stable for the duration of the bus cycle.
5. Data Memory asserts DACK L if the address is within its range.
6. Data Memory puts data out on the DDATA lines. The requestor uses the negation of its internal grant to strobe the data into its buffer.
7. At the end of the bus cycle, the requestor, using its internally synchronized grant, will latch the data and address from the bus lines and sample the DACK L signal. A requestor is responsible for post-checking parity accesses.

Figure 3-16 Data Bus Read Timing




NOTES: 1. ALL VALUES IN NANOSECONDS
2. ONE CHARACTER = 5 NANOSECONDS (APPROX.)
3. DREQ i L MAY BE ASSERTED AT LEAST 10 NANOSECONDS AFTER THE PRECEDING NEGATIVE TRANSITION OF CLK B H (UP TO 90 NANOSECONDS BEFORE THE POSITIVE TRANSITION OF CLK B H)

CX–1107A

### 3.6.2.2 Data Bus Write Cycle

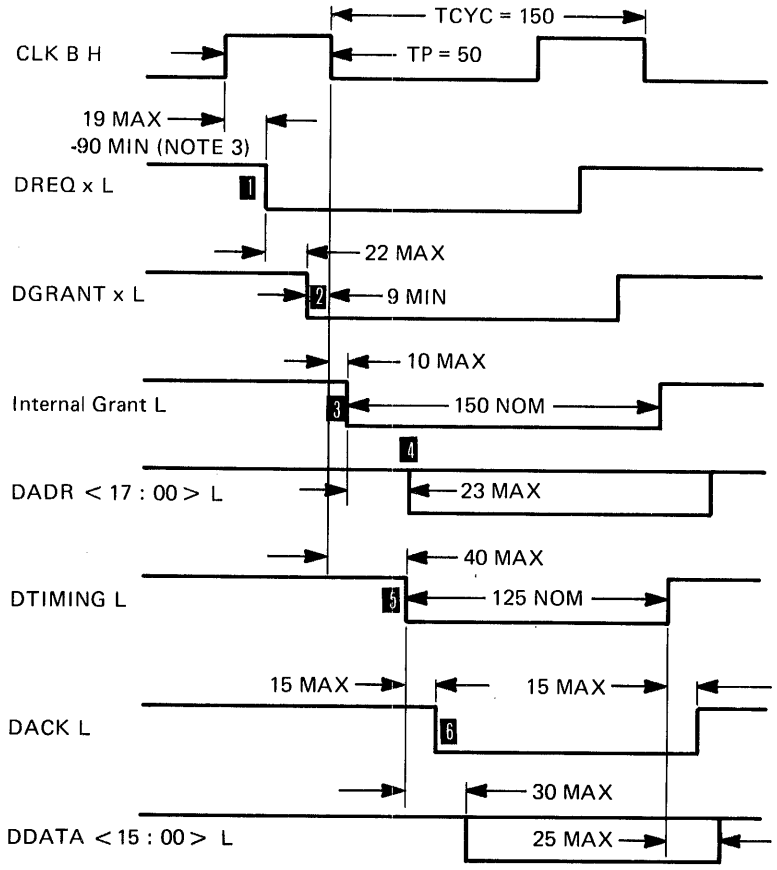
A write cycle is performed when the granted requestor asserts either DWRTH L or DWRTL L during the Data Bus cycle. With DWRTH L asserted, DDATA <15:08,HP> will be written to the high byte of the location in Data Memory specified by the address on the DADR <17:00> lines. When DWRTL L is asserted, DDATA <07:00,LP> are written to the low byte of the specified address. When both lines are asserted, DDATA <15:00,HP,LP> are all written into the memory word specified by the address lines.

A requestor is responsible for pre-generating parity on write-to-memory accesses. A parity bit is generated and stored separately for each byte. Data Memory does not check or generate parity, but only stores the parity bits as part of the data.

The following numbered list refers to the events in Figure 3-17 and gives a description of the timing during a normal Control Bus write cycle.

1. When a cycle is desire, a requestor asserts its request line (DREQ x L) no later than 19 nanoseconds after the positive transition of CLK B H and awaits the result of arbitration (performed on the P.ioc/P.ioj module).
2. When successful, the selected requestor receives an asserted grant line (DGRANT x L) at least 9 nanoseconds before the negative transition of CLK B H.
3. The requestor clocks this asserted grant directly into a flop with this negative transition. This synchronized grant defines a Data Bus cycle for that requestor, and it has possession of the Data Bus until the grant is dropped.
4. Using this internally synchronized grant, the requestor then asserts the desired address onto the appropriate lines and negates DNMA H.
5. The arbitrator on the P.ioc/P.ioj asserts DTIMING L to enable the drivers on the requestor for DDATA <15:00>.
6. Data Memory asserts the DACK L signal if the address is within its range.

Figure 3-17 Data Bus Write Timing




NOTES: 1. ALL VALUES IN NANOSECONDS
2. ONE CHARACTER = 5 NANOSECONDS (APPROX.)
3. DREQ x L MAY BE ASSERTED AT LEAST 10 NANOSECONDS AFTER THE PRECEDING NEGATIVE TRANSITION OF CLK B H (UP TO 90 NANOSECONDS BEFORE THE POSITIVE TRANSITION OF CLK B H)

CX–1108A

### 3.6.2.3 Data Bus NMA Cycle

An NMA cycle on the Data Bus is initiated by a requestor to check proper operation of its Data Bus transceivers. The requestor drives the data and address onto the bus and then reads it back.The NMA cycle may be performed on any Data Bus cycle.

For an NMA cycle, *all* timing signals are used in the same way as in a normal cycle, except that DNMA H is asserted. The assertion of DNMA H disables all Data Memory references and causes the arbitrator to assert DACK L with the proper timing. This latter event is necessary to prevent undesirable non-existent memory errors from occurring in any requestors.

### 3.7 S ENABLE i L LINES

These lines, one per requestor, enable the STATUS <7:0> conditions onto the backplane for use by the P.ioc/P.ioj. The line to each requestor is terminated on the P.ioc/P.ioj so as to default to the negated (high) state.

### 3.8 STATUS <7:0>

STATUS <7:0> are used by the P.ioc/P.ioj to determine the status of a K. The P.ioc/P.ioj reads the STATUS lines from each requestor slot by asserting the associated S ENABLE x L line. The STATUS lines are polled by the P.ioc/P.ioj during HSC initialization or when a requestor informs the P.ioc/P.ioj that an abnormal condition exists.

There is no hardware generation or checking for parity on the STATUS lines. However, the microcode on the K.pli, K.sdi, and K.sti codes parity into the status codes using bit 7. The software is then responsible for ensuring that odd parity exists in the received code. For instance, if a requestor is reporting a code $46_8$, the code in the STATUS <7:0> will be $046_8$. If the requestor is passing a code $47_8$, STATUS <7:0> will be encoded as a $247_8$.

During the initialization polling, if a requestor is not present in the associated slot, a $377_8$ will be returned. If a requestor is present but has failed its self test, a code of $1xx_8$ or $3xx_8$ will be returned. If the requestor has passed its self test, it returns a code indicating what type of requestor it is. For the codes and meanings of the status byte, refer to the service manual.

### 3.9 CLK A H and CLK B H LINES

The clock lines provide the base frequency for the majority of logic in the HSC so that it may run as a completely synchronous system. All requestors use CLK B H to clock both D REQ x L and D GRANT x L to eliminate any possibility of clock skew in this time-critical area.

### 3.10 CLK CTL H LINE

When asserted, the CLK CTL H line is used to define which CLK A/B H cycle is to be used for the upper (control) microsequencer as opposed to the lower (data) microsequencer in the Ks. This allows repeatable inter-K synchronization to assure that Ks make requests at predictable times with respect to other Ks.

This line is generated so that it changes synchronously with the rising edge of CLK A/B H. Ks strobe the CLK CTL signal into a flop with the falling edge of CLK A/B H prior to any use of the signal.

### 3.11 CLEAR x H LINE

The CLEAR x H line (where x is 1 through 7 for an HSC50 and 1 through 9 for an HSC70) is used as an unconditional clear and disable on each requestor. This clear is not dependent on a microsequencer controlled clear, and it clears any logic on the requestor that can be cleared. Since there is a unique version of this signal for every requestor, each requestor can be selectively cleared for diagnostic purposes or forcibly disabled for recovery purposes. Every requestor terminates this line so that it defaults to the asserted state. When there is no P.ioc/P.ioj module installed in the backplane, all requestors are cleared.

### 3.12 HOST CLR H LINE

The K.pli asserts HOST CLR H to clear out the P.ioc/P.ioj in response to a low level host interface clear. The termination for this signal is on the P.ioc/P.ioj module and defaults to the inactive (low) state.

The K.pli asserts HOST CLEAR H when a host CPU sends a specially defined sequence of two commands.

### 3.13 SLOW CYCLE L

Pin B86 slots 1 through 12 have the signal line SLOW CYCLE L in the HSC70 only, but the signal is not used and is grounded. Hardware circuits are present on the HSC70 Memory Module board to allow a faster cycle time on the Data Bus and Control Bus if the SLOW CYCLE L signal is not asserted.

### 3.14 BASE AND ENA IN AND OUT LINES

The CBASE IN, DBASE IN, CBASE OUT, DBASE OUT, ENA IN, AND ENA OUT lines are not used in any implementation of the HSC. These lines are for the possibility that more memory than is present on one memory module is required.

# CHAPTER 4
# HOST INTERFACE OVERVIEW

## 4.1 INTRODUCTION

This chapter introduces you to the function of the three host interface (K.ci) modules. The first part describes the format of the packets that the K.ci sends and receives over the Computer Interconnect Bus (CI). The second part of the chapter gives you an overview of how the three K.ci modules work together to process the packets.

You should think of a packet as a burst of information sent between nodes on the cluster. This packet may be a Mass Storage Control Protocol (MSCP) command, a datagram, or other information sent between nodes. Think of this packet as a letter that you want to send to someone. Before you send the letter, you must put it in an envelope. The envelope around the letter protects it and also carries the address of the sender and receiver. The following sections describe the format of the packets or envelopes.

## 4.2 PACKET FORMATS

The following sections describe the formats of the two types of packets: information and ACK/NACK (acknowledge/negative acknowledge).

### 4.2.1 Information Packet

Figure 4-1 illustrates the format of an information packet. The information packet carries both messages and data across the CI Bus. The LINK puts the synchronization bytes at the beginning of the packet and the Cyclic Redundancy Check (CRC) and trailer at the end. The Host Processor Module (K.pli) supplies the following parts of the packet:

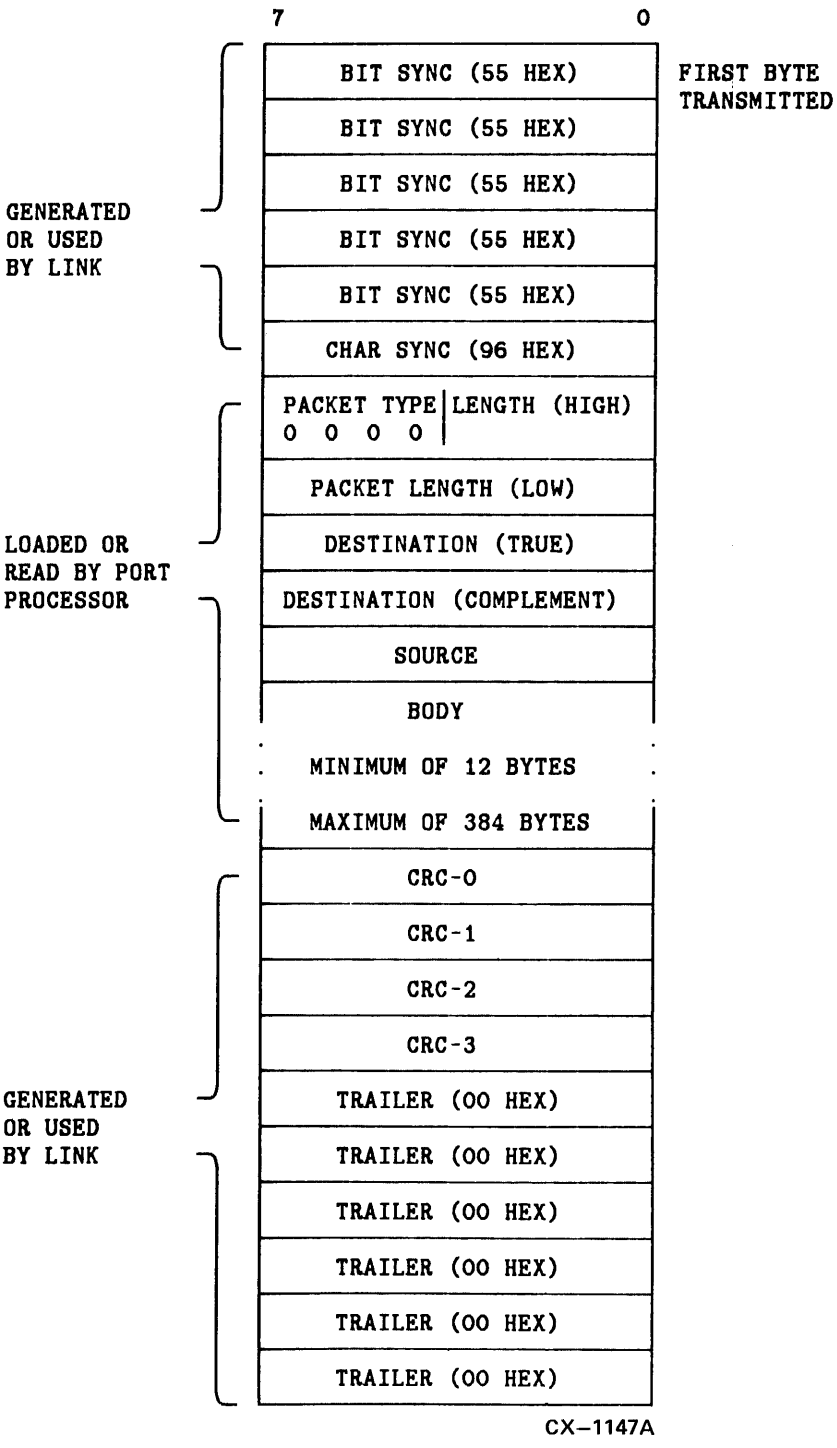
Packet type
Packet length
Destination
Source
Body

Note in Figure 4-1 that the body portion represents the letter and all of the other parts represent the envelope.

#### 4.2.1.1 Bit Synchronization

The first five bytes of the information packet contain bit synchronization (sync) bytes. The bytes are 55 hexadecimal, an alternating pattern of 1s and 0s. This pattern allows the LINK to detect that information is on the CI Bus and then synchronizes the Manchester Decoder for receipt of useful data. The LINK inserts these bit synchronization bytes into the packet during transmission.

Figure 4-1 Information Packet Format



#### 4.2.1.2 Character Synchronization

The character synchronization (Char Sync) byte (96 hexadecimal) indicates the start of useful data in the packet. When the LINK recognizes the synchronization character during packet reception, it starts the framing of the serial data into eight-bit bytes. The LINK inserts the synchronization character into the packet during transmission.

#### 4.2.1.3 Packet Type/Length (High)

The packet type and length (high) byte contains:

- The type of packet (information packet)
- The upper (high) four bits of a 12-bit packet length word

Bits <7:4> (packet type) contain 0s. Bits <3:0> contain the upper four bits of the 12-bit word that specifies the packet length. Information packets vary in length from seven bytes to a maximum of one Kbyte in one-byte increments. The packet length specified by the 12-bit packet length word includes all data from the packet type and length (high) byte up to and including the last byte of the body.

**NOTE**

**The buffers in the Port Buffer Module (PILA) limit the packets to one Kbyte.**

The Port Processor Module (K.pli) supplies the packet type and length (high) byte as part of the packet before sending it to the PILA.

#### 4.2.1.4 Packet Length (Low)

The packet length (low) byte contains the low eight bits of the 12-bit packet length word. The K.pli supplies this byte as part of the packet before sending it to the PILA.

#### 4.2.1.5 Destination (True And Complement)

The destination bytes contain the 8-bit address of the destination node. (The destination represents the address on an envelope.) The K.pli supplies two destination bytes:

- True node address value
- Complement of the true node address value

This redundant destination address eliminates a single logic failure in a node from replying to a packet that was not sent to it. With a single address decode circuit, a logic failure could cause a node to decode another node's address, resulting in both nodes transmitting an acknowledge packet at the same time. This type of failure would result in a collision on the CI Bus and would be seen as a no response by the transmitting node.

#### 4.2.1.6 Source

The source byte contains the 8-bit address of the HSC. The K.pli provides the source byte as part of the packet before sending it to the PILA. (The source is like the return address on the envelope.)

#### 4.2.1.7 Body

The body contains the data and port-processed protocol information. The K.pli supplies the body portion of the packet before sending it to PILA. The body has a minimum of 12 bytes, a maximum of 384 bytes. (The body is like the letter in the envelope.)

#### 4.2.1.8 Cyclic Redundancy Check (CRC) Bytes

The CRC bytes contain the coefficients of the CRC polynomial. During a packet transmission, the CRC logic generates the CRC coefficients (starting with the packet type and length (high) byte). These four bytes make up a 32-bit longword. During packet reception, the LINK regenerates the CRC coefficients and then compares them with the four incoming CRC bytes. An error-free packet results in an exact match between the two longwords. The LINK supplies the four CRC bytes of the packet during transmission.

#### 4.2.1.9 Trailer

The trailer consists of six bytes of all 0s. These bytes insure that all bits of a received packet have been shifted through the LINK before it senses the end of packet reception. The LINK supplies the trailer portion of the packet during transmission.

### 4.2.2 Acknowledge/Negative Acknowledge (ACK/NACK) Packet

Figure 4-2 illustrates the format of ACK and NACK packets. The LINK sends ACK and NACK packets to inform the transmitting node that the packet arrived without data loss or bus collision (CRC checked OK). If the HSC successfully accepted the packet into the buffers in the PILA, the LINK returns an ACK packet indicating a successful bus transaction and storage in the PILA. If the receiver buffers in the PILA were full, the LINK returns a negative acknowledge (NACK) packet indicating that the packet was successfully received but not accepted. The transmitting node must then retransmit the packet.

The LINK generates and transmits the entire ACK (or NACK) packet. An ACK/NACK packet differs from an information packet in the following three ways:

- The ACK/NACK packet has no body. An ACK/NACK packet only acknowledges reception of an information packet. It does not transfer messages or data as such.
- The ACK/NACK packet has no packet length word. All ACK and NACK packets are the same length. Therefore, bits <3:0> of the packet type and length (high) byte contain 0s and the packet length (low) byte contains 0s.
- The ACK/NACK packet type bits <7:4> are different. The packet type and length (high) byte specify the type of packet, as follows:
    - Bit 7 equals a 1 indicating an ACK/NACK packet
    - Bit 6 equals a 1 for an ACK packet, 0 for a NACK packet

## 4.3 HOST INTERFACE (K.ci)

The host interface (K.ci) consists of three modules:

- Port Link Module (LINK)
- Port Buffer Module (PILA)
- Port Processor Module (K.pli)

These modules are the interface between the HSC internal buses and the CI Bus. Figure 4-3 illustrates the host interface modules. Communication and data transfer between the LINK, PILA, and K.pli are over the PORT/LINK Interface (PLI). The K.pli controls all communications and data transfers between these three modules. The Interprocessor Link Interface (ILI) allows communication and data transfer between the PILA and the LINK.

Figure 4-2 ACK/NACK Packet Format

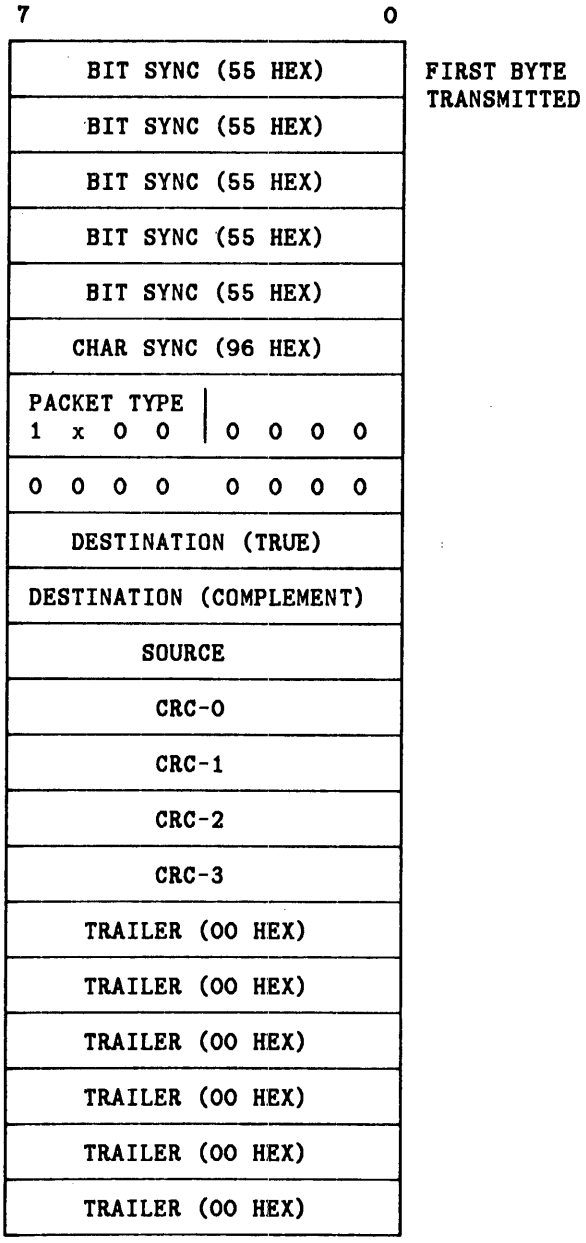
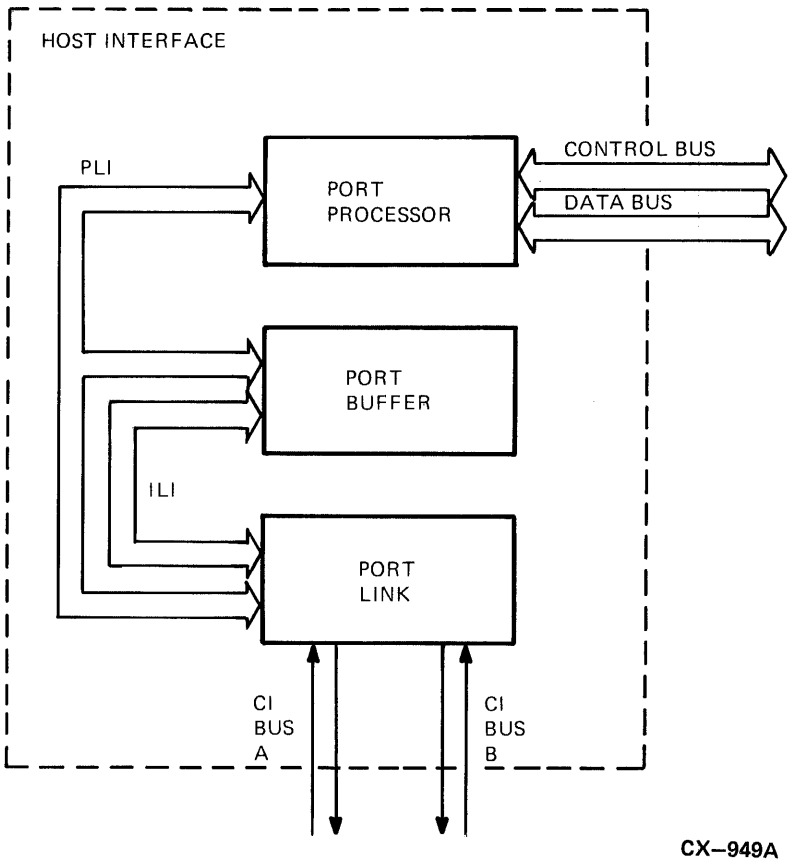


Figure 4-3 Host Interface (K.ci) Modules




The K.ci, an intelligent interface, performs the function of a buffered communications port. It utilizes control structures in the HSC memory to transfer messages and blocks of data between the HSC memory and other nodes within the CI cluster. By providing data buffering, address translation, and serial encoding and decoding, the K.ci reduces the amount of overhead software processing required to complete high-level intercomputer communications.

Figure 4-4 shows a block diagram of the K.ci, and the following sections describe this figure.

### 4.3.1 Port Link Module (Link)

The LINK provides the interface to the CI Bus and services both CI paths. This module contains a transmit path and a receive path with CRC functions shared between the two channels. The LINK can transmit or receive over only one CI path at a time due to the common CRC logic used by both channels.

The LINK gets data packets from the buffers in the PILA over the XMIT DATA BUS. The LINK appends to the packets the bit and character synchronization, CRC, and trailer.

The transmit path consists of the:

Transmitter
Manchester Encoder
Path Select

The LINK transmitter converts the data packet from a byte format to a 70-megabit-per-second serial format. The Manchester Encoder then encodes the information. The Manchester Encoder combines the serial data with a bit rate clock to produce a modulated (phase encoded) carrier for the CI Bus. The Path Select logic, under K.pli microcode control, selects the CI path (A or B) for the transmission.

The receiver path consists of the:

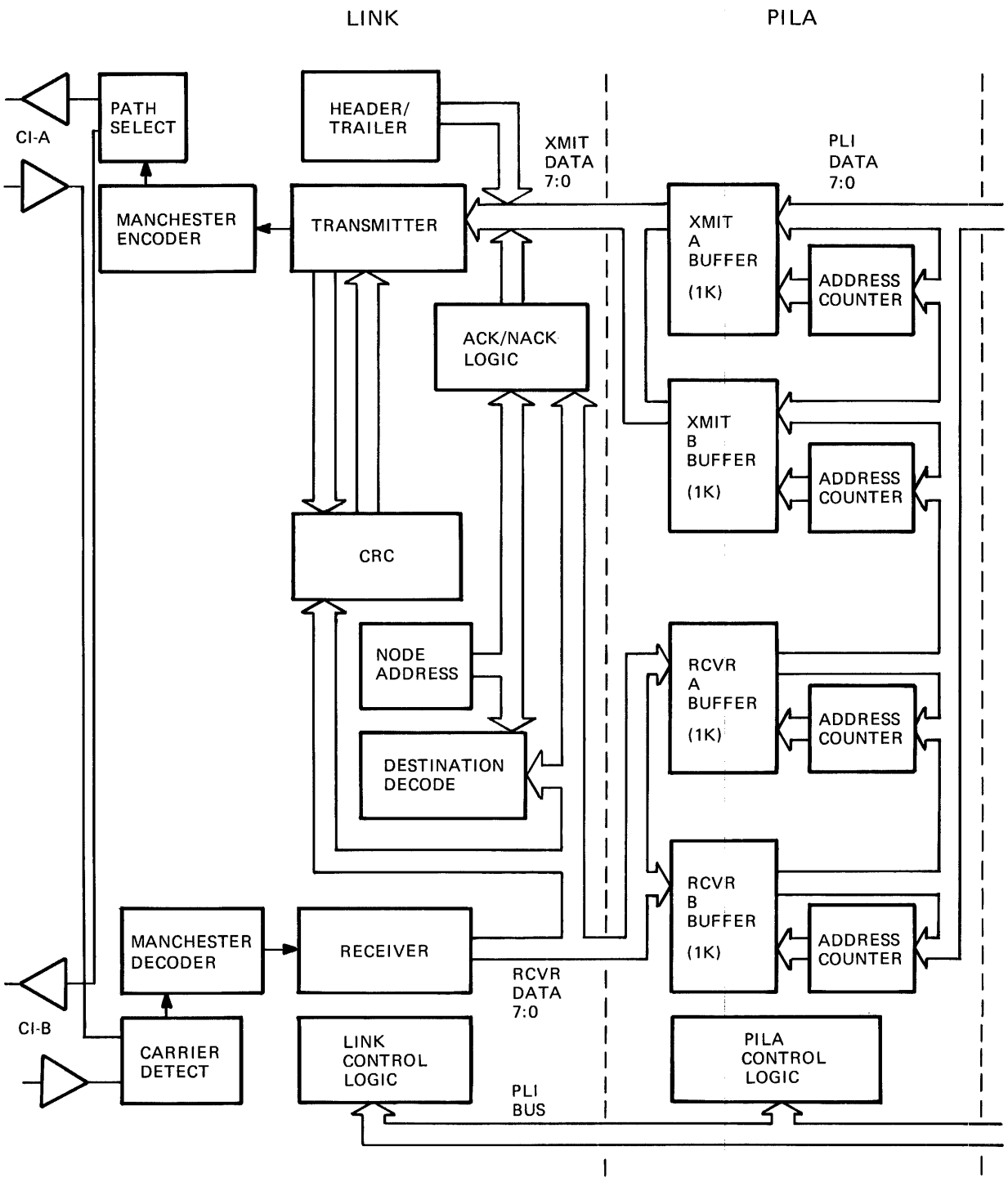
Carrier Detect
Manchester Decode
Receiver
Destination Decode

The Carrier Detect logic monitors the two CI paths and connects the receiver channel to the active path. The logic selects the active path and routes the packet to the Receiver. The Receiver converts the packet data from a 70 megabit-per-second serial format to a byte format. The K.pli routes the packet information to an empty PILA buffer over the RCVR DATA bus (part of the ILI Bus). The Destination Decode logic compares the destination byte of the packet against the node address to determine if the HSC should process the packet. If the HSC should not process the packet, the LINK stops processing the packet. If the HSC should process the packet, the LINK finishes processing the packet. The CRC logic validates the packet by checking the packet data against the packet CRC bytes.

After receiving the packet, the LINK can respond in three different ways:

- If buffers are available in the PILA and the CRC check passed, the LINK will return an ACK packet.
- If the PILA buffers are full but the CRC check passed, the LINK will return an NACK packet.
- If the CRC check does not pass, the LINK will not respond. Therefore, the sending node will send the packet again.

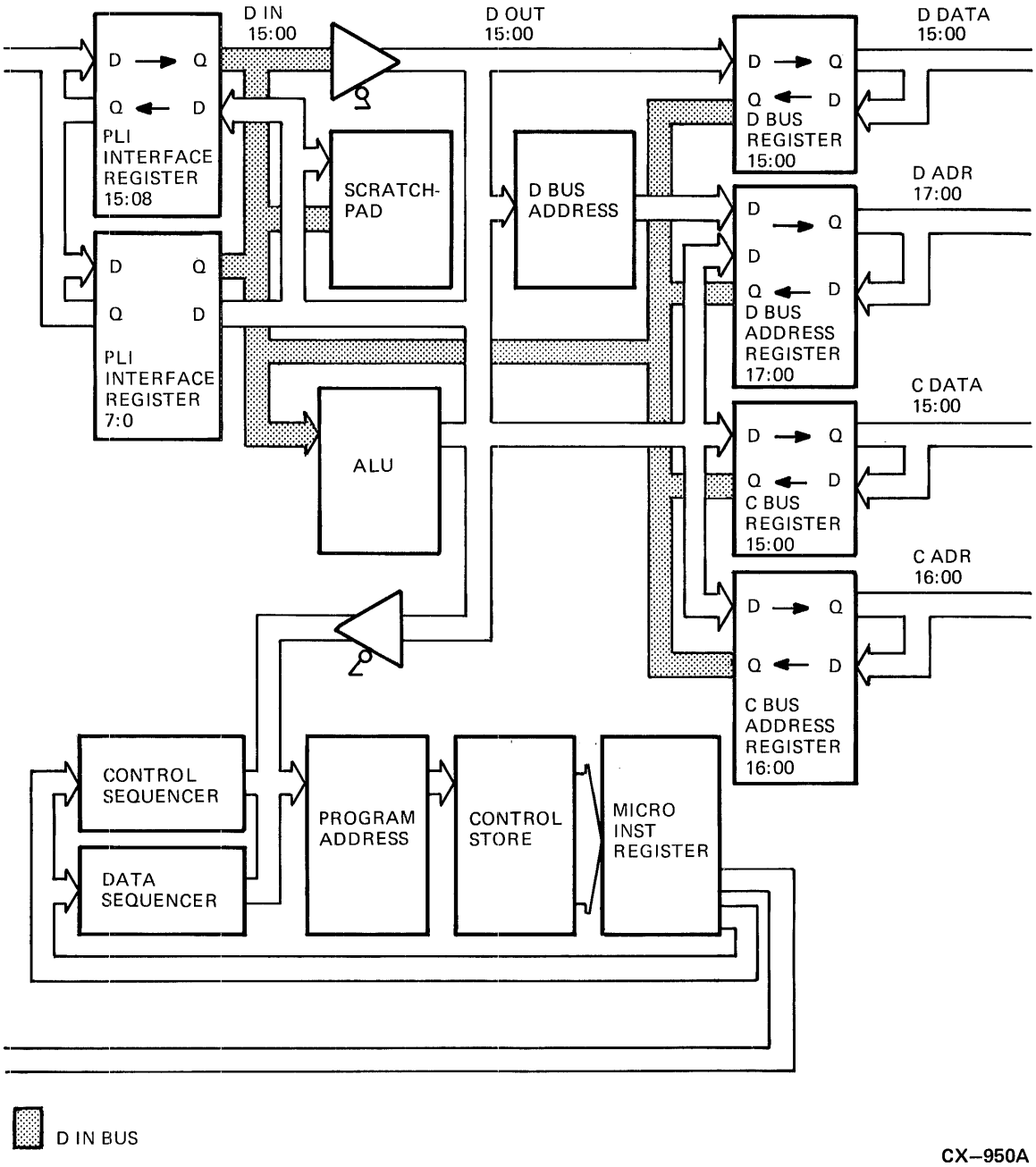
Figure 4-4 Host Interface (K.ci) Block Diagram




(Continued on next page)

**Figure 4-4 (Cont.) Host Interface (K.ci) Block Diagram**

K.pli




CX–950A
Sheet 2 of 2

### 4.3.2 Port Buffer Module (PILA)

The Port Buffer Module (PILA) provides buffering for the data packets transferred through the K.ci. The PILA contains two transmit and two receive buffers (A and B). Each buffer has a capacity of one Kbyte. When data packets are being transmitted, the K.pli loads the Transmit A Buffer over the PLI DATA bus. The K.pli loads Transmit B Buffer while the LINK transmits the packet in Transmit A Buffer. Likewise, the K.pli routes a received data packet to the Receive A Buffer over the RCVR DATA bus. If the LINK accepts another packet before Receive A Buffer empties, the K.pli routes the packet to Receive B Buffer. The control sequencer in the K.pli controls the packet transfers over the PLI and ILI Buses.

### 4.3.3 Port Processor Module

The port processor (K.pli) contains two logic groups:

- Two sequencers and a shared arithmetic logic unit (ALU)
- Interface between the PILA and the rest of the HSC

The two sequencers alternately and independently access the common control store. The two sequencers, in conjunction with the control store and ALU, control different portions of the external interfaces and have separate and independent functions. The data sequencer handles all of a data transfers between data memory and the PLI interface. The control sequencer has responsibility for all other work the K.pli must do. This work includes:

- Examining queues for work to do
- Acquiring and retiring buffers

The K.pli obtains or passes data to and from the PILA, Control Bus, and Data Bus. The Control and Data Buses are 16 bits wide, as are the two internal buses, D IN and D OUT. However, the PLI DATA bus is only 8-bits wide. The PLI Interface Registers convert from 8-bit to 16-bit on a transfer from the PILA to the K.pli. Also, the PLI Interface Registers convert from 16-bit to 8-bit on a transfer from the K.pli to the PILA.

# CHAPTER 5
# PORT LINK MODULE

## 5.1 PORT LINK MODULE FUNCTIONAL OVERVIEW

The Port Link Module (LINK) (L0100) provides the interface to the Computer Interconnect (CI) Bus. This module is functionally divided into a transmit path and a receive path with a cyclic redundancy check (CRC) function shared between the two paths. Due to the common CRC logic used by both paths, the LINK can transmit or receive over only one CI Bus (A path or B path) at a time. Figure 5-1 is a simplified block diagram of the LINK module.

The LINK receives data packets from the Port Buffer Module (PILA) over the XMIT DATA bus. The LINK appends a header and a trailer to each packet. Headers contain both source and destination information for the packet. Node address switches on the LINK provide the node with its address on the CI Bus. The packet header contains this address as a source identification. The trailer following the packet body keeps the receiver logic active while the LINK processes the last data bytes.

The CRC Logic uses the packet data bytes to generate four CRC bytes and appends them to the data packet. The receiving node checks the generated CRC bytes for transmission errors.

The LINK also converts the data packet from a byte format to a 70-megabit-per-second serial format using Manchester encoding. In order to produce a modulated (phase encoded) carrier for the CI Bus, the Manchester Encoder combines the serial data with the bit rate clock.

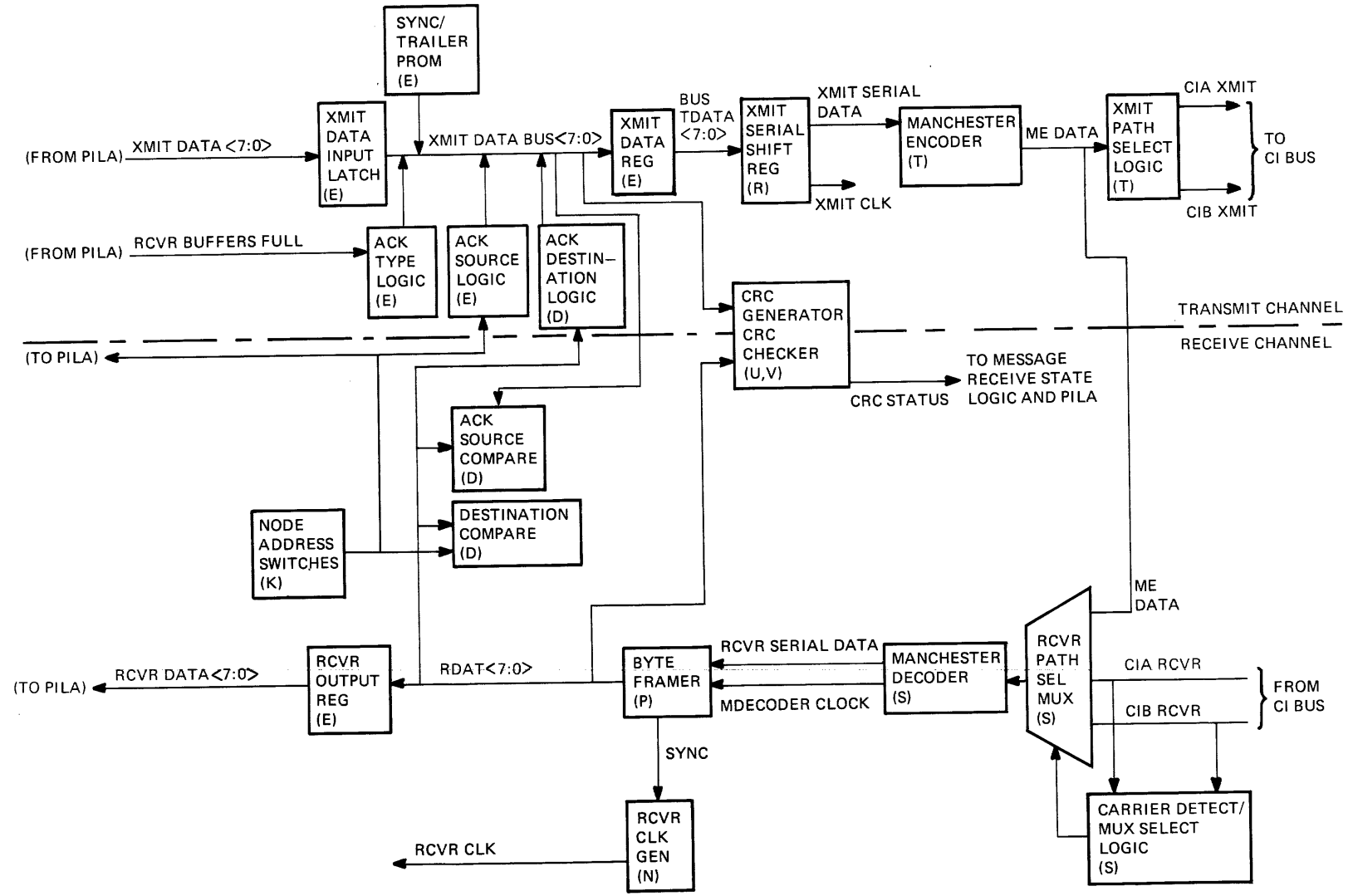
The Transmit (XMIT) Path Select Logic, under microcode control, selects the CI transmission path (A or B). Carrier Detection Logic monitors the two CI paths and connects the receiver channel to the active path.

Serial data from the CI is Manchester decoded, separating the signal into its clock and data components. These components are applied to the LINK receiver. It, in turn, converts the packet data from a 70 megabit-per-second serial format to a 8-bit parallel byte format. The LINK receiver then presents the packet data to the CRC Logic for validation. CRC Logic checks the packet data against the packet CRC bytes.

If a CRC error is detected, no response (ACK or NACK - acknowledge or no acknowledge) is returned to the transmitting node. If no CRC error exists, the packet is sent to the PILA over the RCVR DATA bus. If the PILA can accept the packet, the LINK returns an ACK message to the transmitting node. If the buffers on the PILA module are full and cannot accept the packet, the LINK returns a NACK message to the transmitting node. That node then retransmits the packet.

LINK Control Logic receives commands from the Port Processor Module (K.pli) to select and start LINK operations. This logic also senses signal conditions controlling data packet transfers through the LINK. For timing operations on the receive and transmit channels, a receive clock (RCVR CLK) and a transmit clock (XMIT CLK) are generated on the LINK.

Figure 5-1 LINK Simplified Block Diagram




NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX–951A

The following sections describe the four LINK operations with the transmit and receive channels functioning according to the specific packet type being processed. These four operations are described in the order of occurrence:

Reception of an information packet
Transmission of an ACK/NACK packet
Transmission of an information packet
Reception of an ACK/NACK packet

### 5.1.1 Information Packet Reception

Data packets on the CI Bus are in serial format at a serial bit rate of 70 MHz. Data is Manchester encoded (phase encoded) where the clock is incorporated into the modulated signal. CI paths A and B are input to a RCVR Path Select Mux in the LINK. Carrier Detect Logic monitors both CI paths. When the logic senses the initial presence of signals on one of the paths enabled by the K.pli, it switches the multiplexor to the active path, selecting CIA RCVR or CIB RCVR for the Manchester Decoder. The K.pli can also select an internal path where the multiplexor selects the output from the transmit channel and loops it back into the port. The K.pli uses this feature for maintenance operations.

The multiplexor output is applied to the Manchester Decoder where the signal clock is extracted from the modulated signal. The Manchester Decoder outputs the data (RCVR SERIAL DATA) and the clock (MDECODER CLOCK) to the Byte Framer containing the sync character detector.

The Byte Framer performs serial-to-parallel conversion of signal data. The sync character detector enables the framer. The framer, when enabled, outputs a data byte (RDAT <7:0>) for every eight serial bits received from the Manchester Decoder. A RCVR CLK generator develops RCVR CLK which times the data transfers through the LINK receive channel. SYNC from the Byte Framer synchronizes RCVR CLK with the data bytes and approximately centers RCVR CLK within the asserted time period of RDAT <7:00>. RDAT <7:00> data bytes are clocked into the RCVR output register and sent to the PILA as RCVR DATA <7:00>.

Correct packet destination is verified by the Destination Compare Logic. Packet destination bytes are compared to the node address set into the LINK node address switches. If they do not compare, the receiver is cleared and reception is terminated.

The packet source byte is extracted from the incoming packet and placed into the ACK destination register. When the LINK transmits an ACK packet in response to the information packet being received, it uses the address in that register (source of the information packet) as the ACK destination.

The packet bytes extending from the packet type and length (high) byte up to and including the last byte of the body are checked by the CRC. The CRC algorithm acts on the bytes and generates a 32-bit CRC longword. The four CRC bytes in the packet are compared to the generated longword. If the packet is error free, CRC STATUS is asserted to the message receive logic.

After the packet trailer passes through the LINK, the Carrier Detect Logic senses the end of the packet and informs the ACK Transmit Logic. This logic initiates transmission of the ACK packet.

### 5.1.2 ACK/NACK Packet Transmission

ACK/NACK packets are generated and transmitted entirely by the LINK. No packet information is received from the PILA as XMIT DATA <7:00>.

The ACK Transmit Logic initiates the transmit operation by enabling the Sync/Trailer PROM. This Sync /Trailer PROM outputs five 8-bit-sync bytes and a sync character byte onto the XMIT DATA bus (XMIT DATA BUS <7:00>). The ACK Type Logic is enabled and outputs the packet type byte onto the XMIT DATA bus. The PILA signal RCVR BUFFERS FULL is sampled at the start of information packet reception. If RCVR BUFFERS FULL is true, the PILA cannot accept the information packet just received. In this case, the ACK Type Logic outputs the code for a NACK packet. If RCVR BUFFERS FULL is false, the logic outputs the code for an ACK type packet.

LINK Control Logic then enables the output of the ACK Destination Register which outputs the two destination bytes onto the XMIT DATA bus. The destination value used is the source address taken from the information packet just received. ACK Source Logic is enabled and the node address from the node address switches is transferred to the XMIT DATA Bus as the source byte.

The ACK/NACK packet is transferred to the BUS TDATA Bus via the XMIT Data Register. Starting with the packet type byte, the packet is also input into the CRC generator where a 32-bit CRC longword is generated. After the source byte is input to the CRC generator, the LINK Control Logic gates the CRC longword (a byte at a time) onto the BUS TDATA Bus. The four CRC bytes are thus inserted into the ACK/NACK packet.

Finally, the ACK Transmit Logic reenables the Sync/Trailer PROM outputting six trailer bytes onto the XMIT DATA Bus and completing the ACK/NACK packet.

Sending the ACK/NACK packet from the BUS TDATA Bus through the XMIT Shift Register converts the signal data from parallel to serial. Data bytes input to the register are shifted out serially at a 70 MHz bit rate to the Manchester Encoder as XMIT SERIAL DATA. Serial Shift Register Logic also generates XMIT CLK. This signal times the data transfer through the LINK transmit channel. XMIT CLK and the serial data are synchronized within the shift register. The Manchester Encoder combines data from XMIT SERIAL DATA with the bit rate clock producing a phase-encoded carrier. It then outputs the modulated carrier (ME DATA) to the CI Bus. ACK Transmit Logic selects the same CI path used by the information packet just received. In addition, ME DATA can follow an internal loop path into the receive channel. This feature is used for maintenance testing and is possible only if the LINK is in internal loop mode and the receiver inputs from the CI Bus are disabled.

### 5.1.3 Information Packet Transmission

All information packets are generated by the K.pli and input to the LINK transmit channel from the PILA. Information packet bytes inserted by the LINK are:

Five 8-bit sync bytes
One 8-bit character sync byte
Four 8-bit CRC bytes
Six 8-bit trailer bytes

Information packet transfers use some of the functions described in the previous section, and they operate as described.

Using the Message Transmit Logic, the K.pli initiates the transmit operation. When the transmit operation is initiated, the LINK enables the Sync/Trailer PROM. This signal outputs five 8-bit sync bytes and an 8-bit sync character byte onto the XMIT DATA Bus. The packet type and length (high) byte, the packet length (low) byte, and the destination bytes are provided by the K.pli.

When the destination bytes are on the XMIT DATA Bus, the LINK enters the destination address into the ACK Source Compare Logic. When the ACK/NACK response packet is received from the target node, the packet source byte is compared with the contents of the compare logic. If the correct node responded, a match is obtained.

The K.pli, not the LINK, inserts the source byte. The K.pli reads the node address switches on the LINK. The source byte is an input to the XMIT DATA Bus from the PILA.

The K.pli generates the body before sending it to the LINK.

The CRC Generator produces the four 8-bit CRC bytes just as for an ACK/NACK transmission. However, the information packet body is also input to the CRC Generator and contributes to the CRC longword generation. Finally, the LINK Message Transmit Logic reenables the Sync/Trailer PROM. This signal outputs the six trailer bytes onto the XMIT DATA Bus, completing the information packet.

### 5.1.4 ACK/NACK Packet Reception

Transfer of an ACK/NACK packet through the receive channel utilizes most of the functions described in the previous section. These functions also operate as described. With regard to the LINK receive channel, the basic difference between an ACK/NACK packet and an information packet is the use of the packet source byte. In ACK/NACK reception, the source byte is not entered into the Destination Register but is applied to the ACK Source Compare Logic. This logic currently contains the destination address of the information packet just transmitted. The source byte is compared to the destination address and matches if the correct nodes are involved in the data transfer.

## 5.2 RECEIVE AND TRANSMIT CHANNELS

The following sections describe receive channel and transmit channel hardware. Hardware control is a function of K.pli commands, the type of operation being executed, and conditions sensed by the logic (for example, errors) during the operation. Hardware control is implemented by Programmable Array Logic (PAL) defining various hardware states during each operation. The states are represented in four diagrams in the engineering drawing set. (Operations represented in these diagrams are shown in Table 5-1 and described in Section 5.7.)

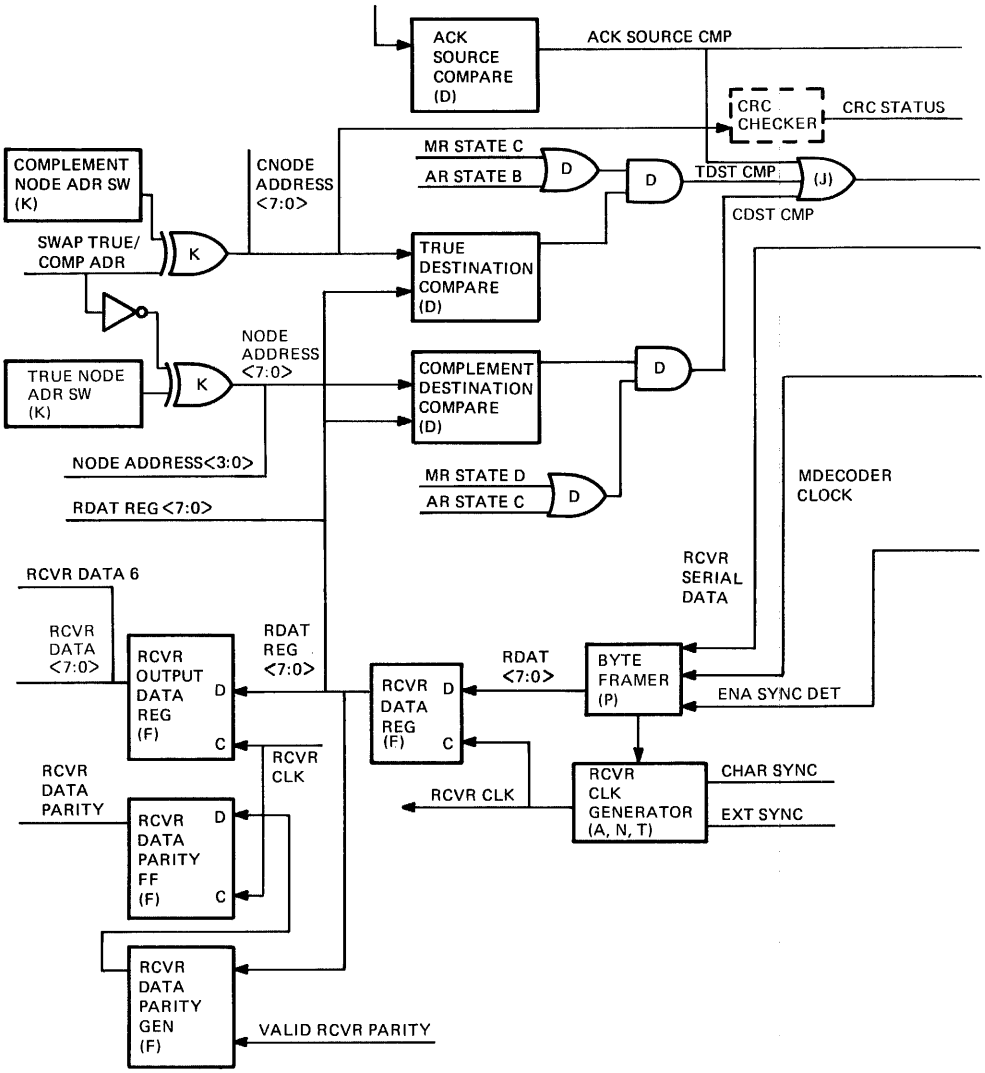
Table 5-1 LINK States

| Operation | Number of States |
|---|---|
| Information Packet Reception | 13 |
| ACK Packet Transmission | 8 |
| Information Packet Transmission | 13 |
| ACK Packet Reception | 8 |

### 5.2.1 Receive Channel

Figure 5-2 is a block diagram of the receive channel and should be referenced throughout this section.

## Figure 5-2 Receive Channel Block Diagram

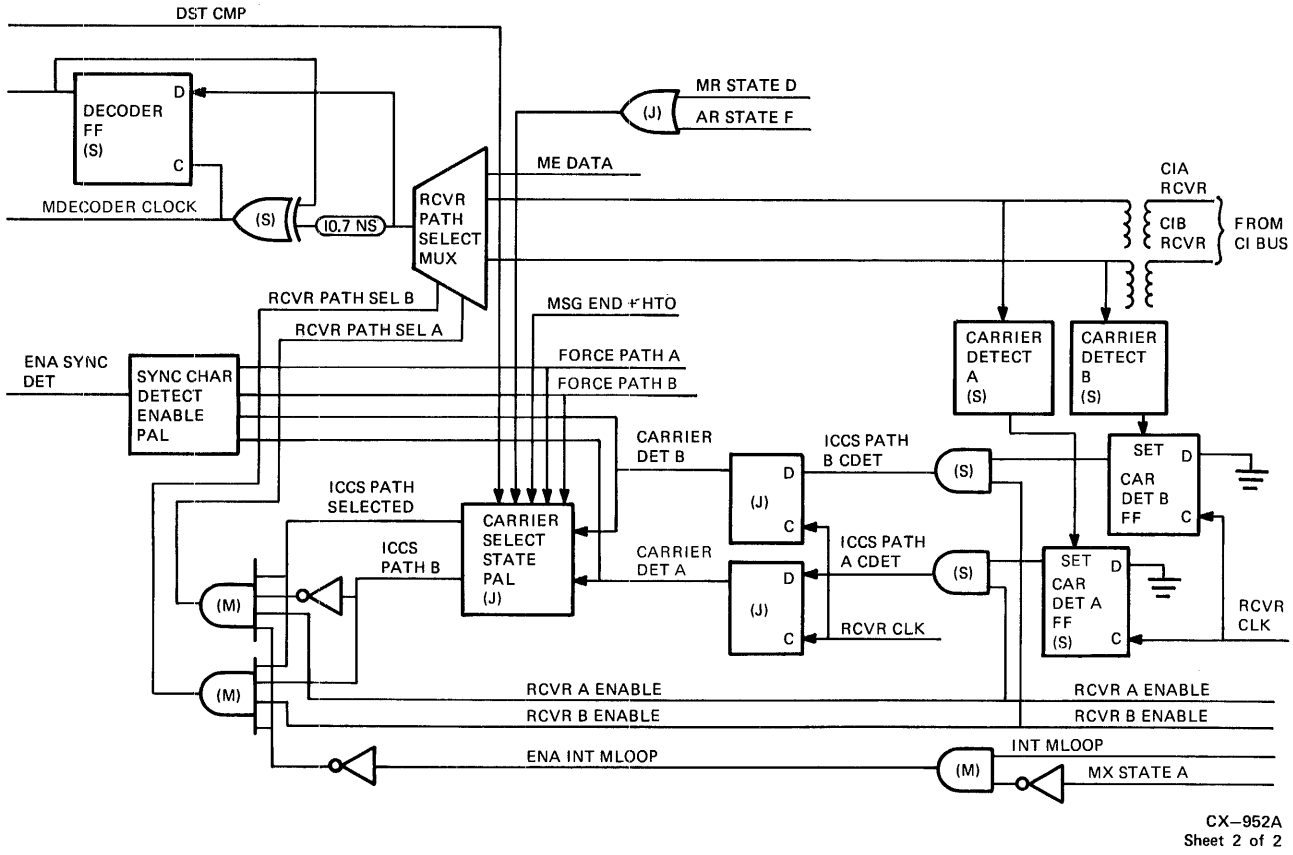



NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX–952A
Sheet 1 of 2

(Continued on next page)

Figure 5-2 (Cont.) Receive Channel Block Diagram



Receive channel hardware contains both transistor-transistor logic (TTL) and emitter-coupled logic (ECL). The Carrier Detection/Path Selection Logic, Manchester Decoder, Byte Framer and Sync Character Detector all use ECL logic. For those unfamiliar with ECL logic, the Receive Path Select Multiplexor is described in Section 5.2.1.3.

#### 5.2.1.1 CI Carrier Detection and Path Selection

The Carrier Detection/Path Selection Logic monitors CI Bus activity and, when activity is detected, selects the active path as an input to the LINK receive channel. The K.pli, through the LINK control PALs, specifies which receive channel can receive signal inputs from the CI Bus. These PALs enable the receive channels by asserting RCVR A ENABLE or RCVR B ENABLE.

#### 5.2.1.2 Carrier Detect Logic

The Carrier Detect Logic is identical on both paths A and B. If a carrier is present on CI path A, the Carrier Detect A Logic sets the carrier detect A flip-flop. If the K.pli has enabled channel A (RCVR A ENABLE true), ICCS PATH A CDET is asserted and a flip-flop on the next RCVR CLK asserts CARRIER DET A. The flip-flop outputs CARRIER DET A to the Carrier Select State PAL. If the current state of the port permits opening a receive channel, the Carrier State Select PAL asserts ICCS PATH SELECTED and negates ICCS PATH B. RCVR PATH SEL A directs the Receive Path Select Multiplexor to select CI path A for the multiplexor input. If activity is sensed on CI path B, similar logic selects CI path B for the multiplexor input (but not A and B at the same time.)

LINK Control Logic signals, FORCE PATH A and FORCE PATH B, force a corresponding path selection from the Carrier Select State PAL. When the K.pli commands a message transmission, the transmission path is reserved in the receive channel waiting to receive the ACK response.

The K.pli, through the LINK control PALs, can also select the internal maintenance loop (INT MLOOP) where ME DATA from the transmit channel is selected for the multiplexor input. The true state of INT MLOOP inhibits both RCVR PATH A and RVCVR PATH B causing the multiplexor to select the ME DATA input signal.

#### 5.2.1.3 Receive Path Select Multiplexor - ECL Logic

The Receive Path Select Multiplexor is shown in Figure 5-3 and also on sheet S of the engineering drawing set. If RCVR PATH SEL A is true, the output of OR gate A can follow the CIA RCVR signal input.

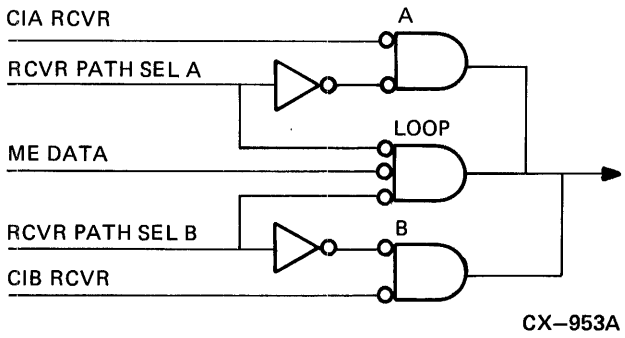
The signal RCVR PATH SEL B is false holding the output of OR gate B low. In ECL logic, a signal low is the non-active state and a high is the active state. Any gate connected to a common line can pull up the line to the active state. Thus, OR gate B is held inactive (low) while OR gate A transfers the CIA RCVR signal to the Manchester Decoder. The true state of RCVR PATH SEL A also holds the LOOP OR gate in the inactive state.

If RCVR PATH SEL B is true (RCVR PATH SEL A false), OR gate A and the LOOP OR gate are held inactive. OR gate B transfers CIB RCVR to the Manchester Decoder.

#### 5.2.1.4 Manchester Decoder

Phase encoding and Manchester Decoder Logic are discussed in the following sections.

Figure 5-3 Receive Path Select Multiplexor-ECL Logic




### 5.2.1.4.1 Phase Encoding

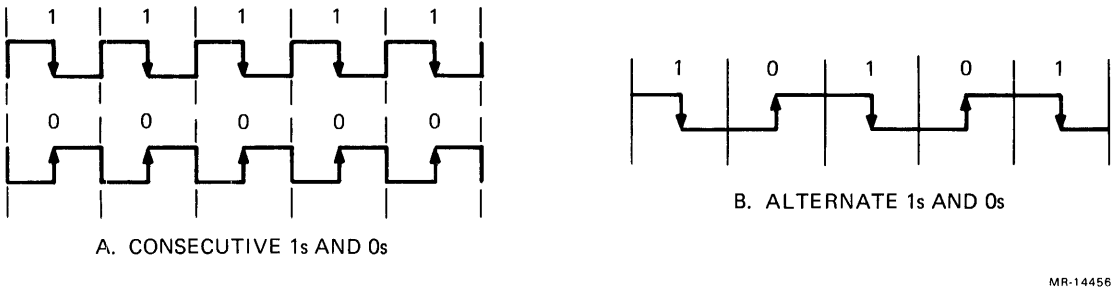
Phase encoding is a modulation technique where a signal phase reversal occurs for each bit of information (Figure 5-4).

- When the input is binary 1, there is a negative transition at bit cell center.
- When the input is binary 0, there is a positive transition at bit cell center.
- When the binary value (either 0 or 1) is the same as the previous cell, the cell must start with a level reversal.
- If the binary value is different than the previous cell, there is no reversal at the start of the cell.

Note in Figure 5-4 that consecutive ones or consecutive zeros cause phase reversals to occur at twice the data rate. Alternate ones and zeros cause signal transitions to occur at the data rate.

Phase (Manchester) encoding presents high read integrity, an advantage over other types of encoding. A positive pulse is always binary 0 and a negative pulse is always binary 1.

Figure 5-4 Phase Encoded Data



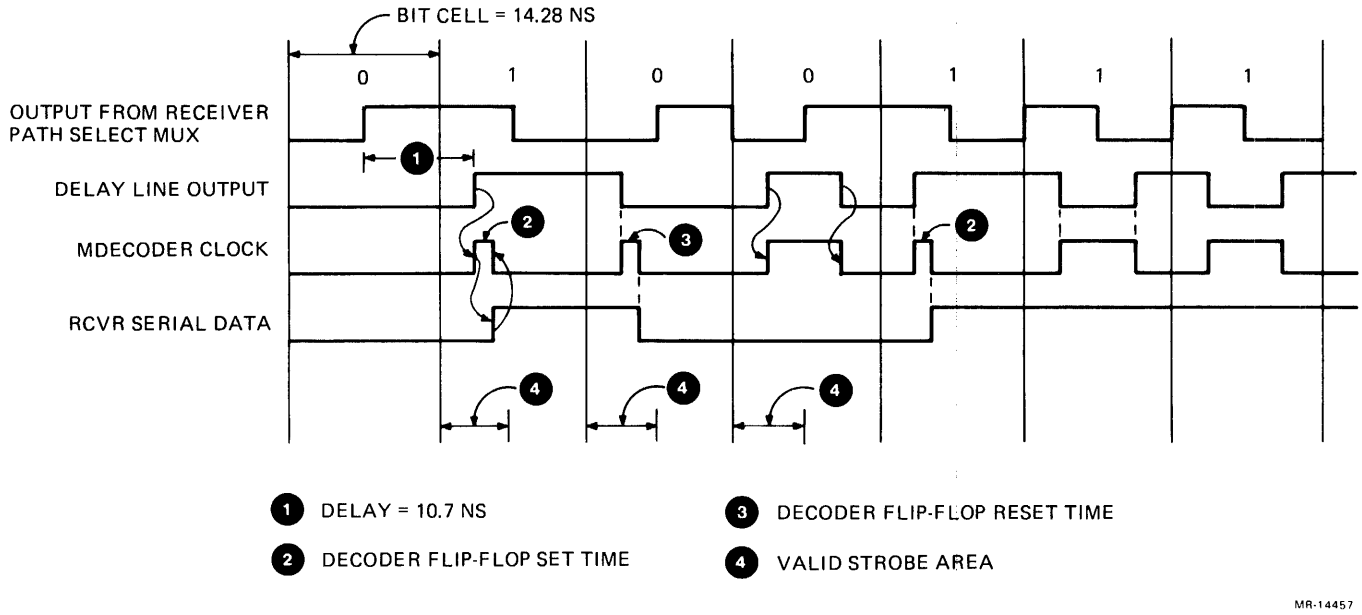
#### 5.2.1.4.2 Decoder Logic

The Manchester Decoder separates the 70 MHz bit rate clock (MDECODER CLOCK) and data (RCVR SERIAL DATA) from the input. The decoder consists of a flip-flop with the signal data from the Receive Path Select Multiplexor as the D input. The flip-flop clock input is derived from XORing the delayed output of the Receive Path Select Multiplexor (delayed 10.7 ns) with the output of the decoder flip-flop.

Figure 5-5 illustrates the action of the decoder logic. Signal data from the Receive Path Select Multiplexor is shown with one or zero transitions at the center of each bit cell. With a 70 MHz bit rate, the width of the bit cells is 14.28 ns. Delay line output is seen as the signal data delayed 10.7 ns. XORing the delayed data with the flip-flop output (RCVR SERIAL DATA) generates the MDECODER CLOCK waveform.

The MDECODER CLOCK is at 70 MHz with a 14.28 ns period. The XOR action generates the clock's rising edge one quarter into each bit cell. This centers the rising edge in the valid strobe area (first half of the bit cell).

**Figure 5-5 Manchester Decoder Timing Diagram**




#### 5.2.1.5 Sync Character Detect Enable PAL

The purpose of the Sync Character Detect Enable PAL is to assert ENA SYNC DET to the Byte Framer when a packet is expected. The PAL asserts ENA SYNC DET immediately after information packet transmissions in anticipation of the ACK (or NACK) response. The PAL monitors CARRIER DET A and CARRIER DET B and asserts ENA SYNC DET when it senses that a signal carrier is being received. The PAL negates ENA SYNC DET during node transmissions (FORCE PATH A, FORCE PATH B) so the LINK does not respond to its own transmissions.

The Byte Framer contains a sync detector enabled by ENA SYNC DET. When the sync detector finds a packet sync character it recognizes a packet is being received. When the detector recognizes the sync character, it enables the Byte Framer to process the packet bytes. By keeping the detector disabled except when a packet is detected, the Sync Character Detect PAL prevents the detector from erroneously

recognizing noise as a sync character. The Sync Character Detect Enable PAL is discussed in more detail in Section 5.7.2.2.

#### 5.2.1.6 Byte Framer

The Byte Framer is enabled when it receives the sync character byte. When the framer recognizes the sync character, it begins serial signal data conversion from the Manchester Decoder into 8-bit data bytes for the RDAT Bus.

As shown in Figure 5-6, RCVR SERIAL DATA is input to the RCVR Serial Shift Register. This register is held in the load state by the negated state of E197R2 so no data is shifted into the register. When the LINK receive channel senses data on the CI Bus, the Sync Character Detect Enable PAL also senses it.

If the PAL determines this is a valid time to receive a packet, it asserts ENA SYNC DET to the SYNC ENA flip-flop. On the next RCVR CLK, the flip-flop outputs SYNC ENA to another flip-flop that asserts E197-R2 to the RCVR serial shift register. The true state of E197-R2 enables the register by changing its state from load to shift. RCVR SERIAL DATA is shifted into the register at the 70 MHz bit rate by MDECODER CLOCK.

Figure 5-7 is a timing diagram of the RCVR Serial Shift Register.

The RCVR Serial Shift Register outputs 8-bit bytes onto a data bus and sends them to the RCVR input register. The sync detector monitors the data looking for the sync character byte on the bus. When the detector recognizes the sync character, it asserts E198-3 to the sync flip-flop. The next MDECODER CLOCK that clocks the last bit into the register also sets the sync flip-flop. SYNC is asserted when the sync character is in the shift register and not one clock pulse later.

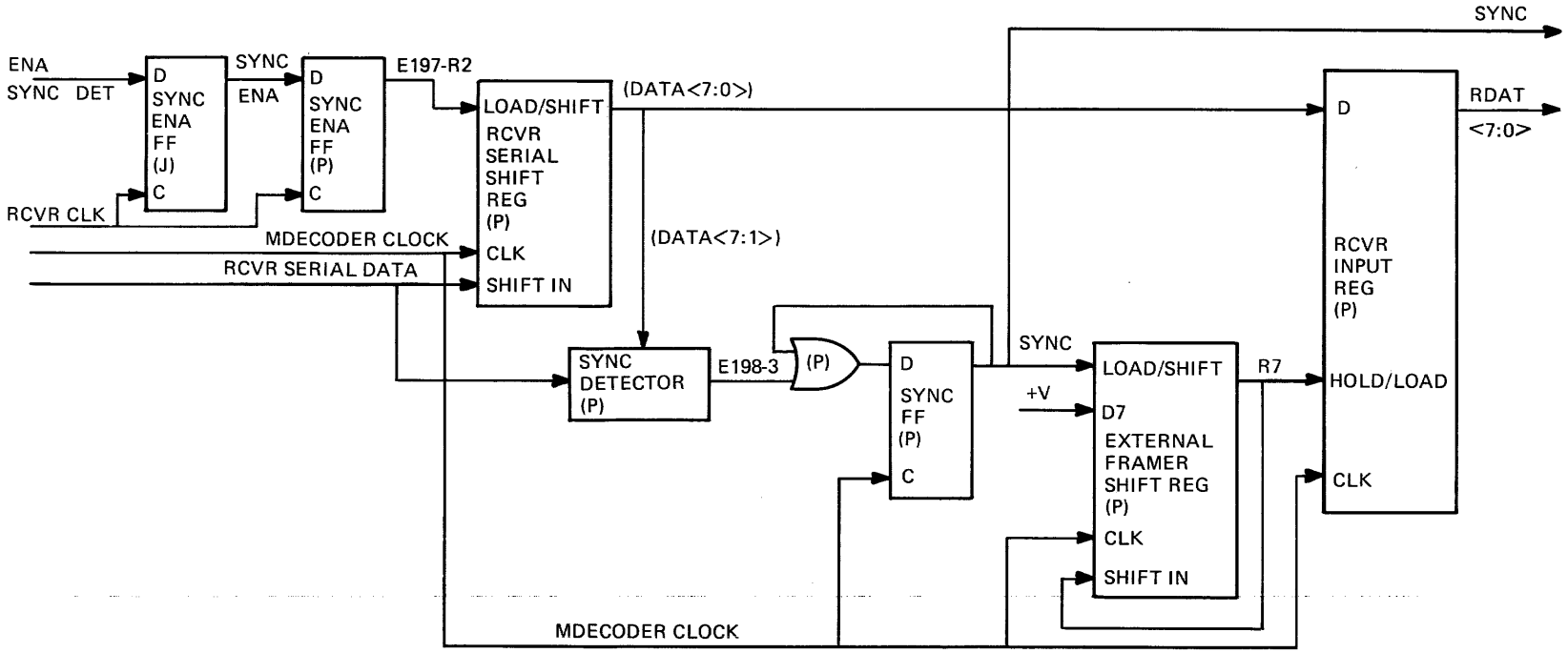
When SYNC is asserted, the External Framer Shift Register switches the RCVR Input Register from the hold state to the load state (for one clock pulse) every eight MDECODER CLOCK pulses. RCVR SERIAL DATA continues to shift into the RCVR Serial Shift Register. Every eight clock pulses a data byte is present in the shift register and on the data bus. At this time, the External Framer Shift Register switches the RCVR Input Register from hold to load. The next MDECODER CLOCK pulse loads the data byte into the register. The D7 input to the External Framer Shift Register is tied high. Before the assertion of SYNC, the framer register is in the load state and the R7 output is true. The true state of the R7 output keeps the RCVR Input Register in the load state. When SYNC is asserted, the framer shift register starts to shift. The 1 at R7 is shifted in and through the framer shift register. Every eight MDECODER CLOCK pulses, the 1 is shifted through to the R7 output, switching the RCVR input register to the load state for one clock pulse. As seen in Figure 5-7, a data byte is on the data bus when the RCVR Input Register is loaded. The timing for the first three bytes of a packet is shown in Figure 5-8.

#### 5.2.1.7 RCVR CLK Generator

Figure 5-9 is a block diagram of the RCVR CLK Generator. The RCVR CLK is derived from a crystal-controlled 70 MHz oscillator. The RCVR CLK pulses time and controls the operation of the receive channel logic. When a signal packet is received, the RCVR CLK and packet bytes are synchronized by the SYNC signal received from the Byte Framer.

The output from the 70 MHz oscillator is doubled to 140 MHz by a Frequency Doubler. (The 140 MHz is used by the Manchester Encoder in the transmit channel.) The 140 MHz is divided down to 35 MHz and applied to a shift register consisting of four flip-flops. The shift register divides the 35 MHz by 4, outputting RCVR CLK at a frequency of 8.75 MHz (time period equals 114.28 ns). Table 5-2 lists the frequency and periods of the LINK clocks. The XMIT CLK discussed in Section 5.2.2.7 is included in the table.

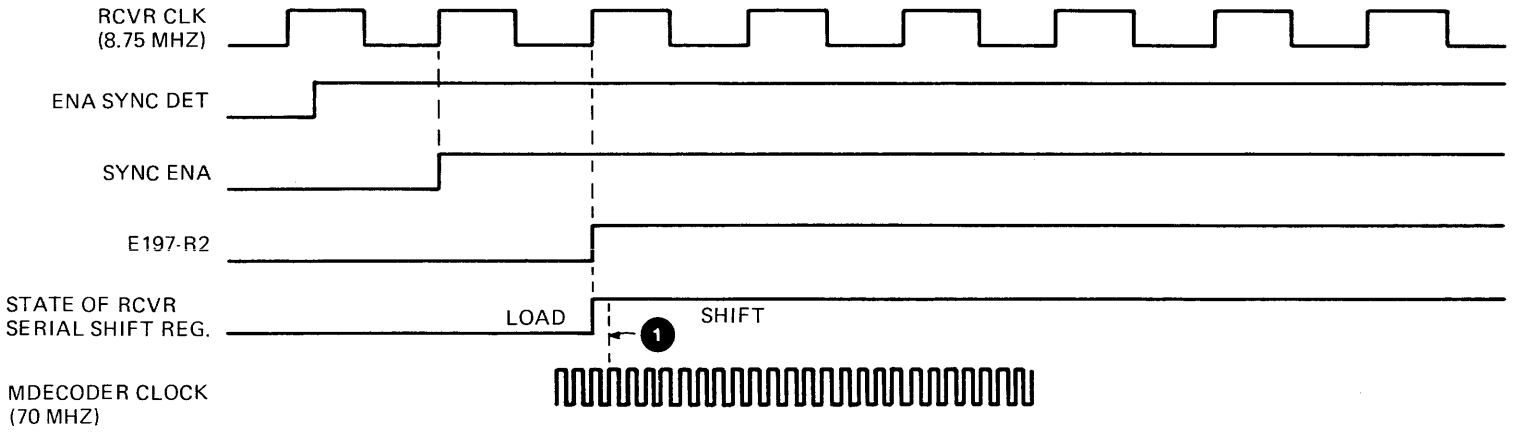
**Figure 5-6 Byte Framer Block Diagram**




NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX-956A

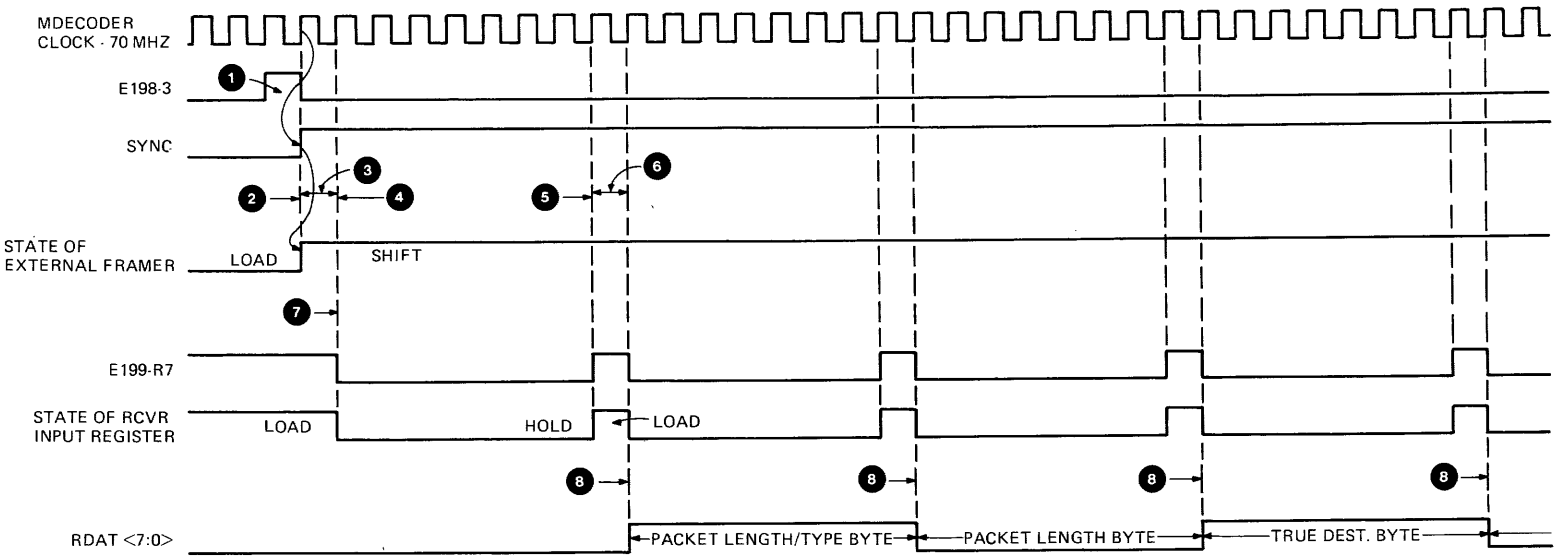
**Figure 5-7 RCVR Serial Shift Register (Enable)**




1 RCVR SERIAL SHIFT REGISTER STARTS TO SHIFT.

MR-14460

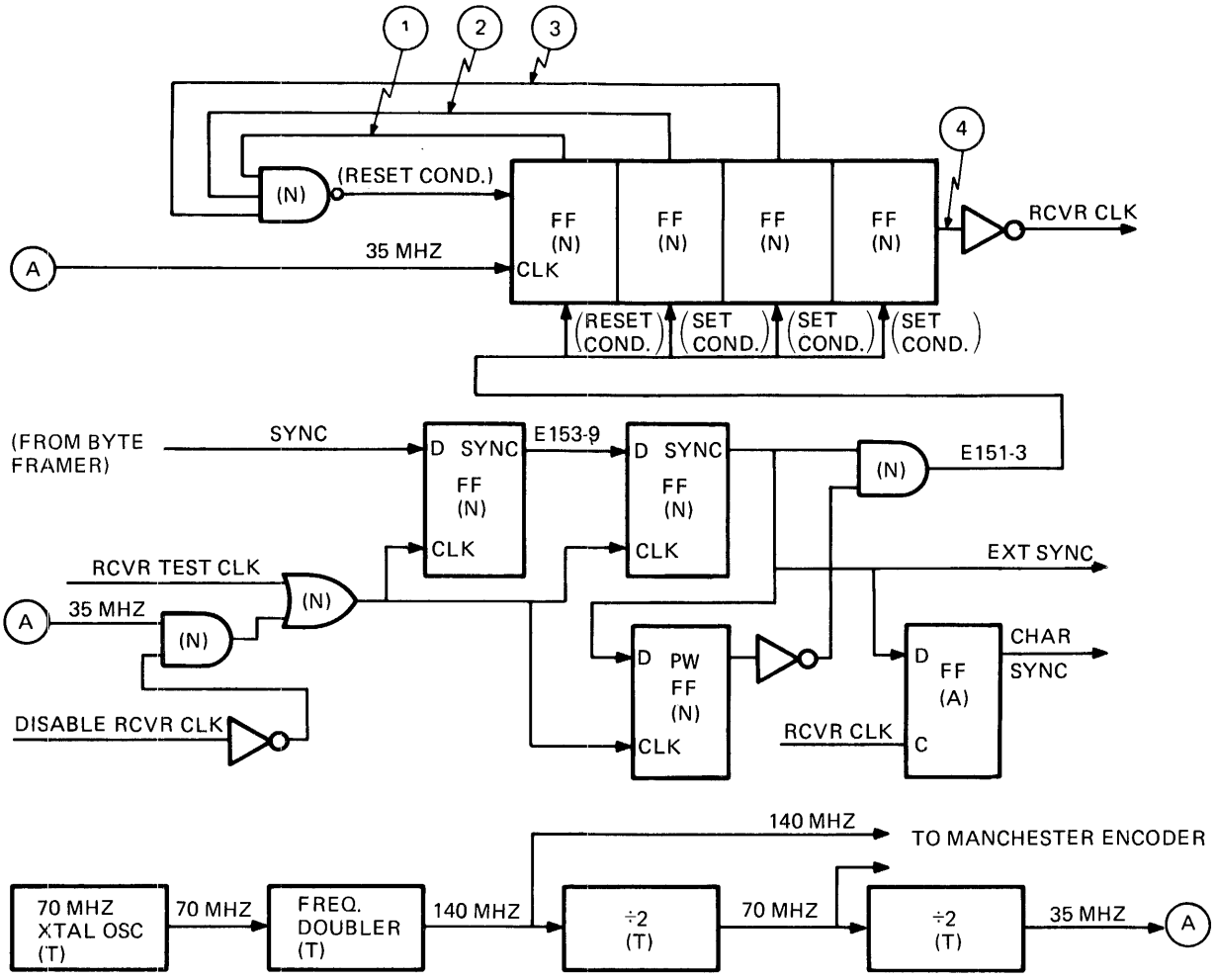
## Figure 5-8 Byte Framer Timing Diagram




1. SYNC CHARACTER BYTE IN SYNC DETECTOR.
2. LAST BIT OF SYNC CHARACTER BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER
3. SYNC CHARACTER BYTE IN RCVR SERIAL SHIFT REGISTER.
4. FIRST BIT OF PACKET LENGTH/TYPE BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER
5. LAST BIT OF PACKET LENGTH/TYPE BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
6. PACKET LENGTH/TYPE BYTE IN RCVR SERIAL SHIFT REGISTER.
7. EXTERNAL FRAMER SHIFT REGISTER STARTS TO SHIFT.
8. (DATA <7:0>) CLOCKED INTO RCVR INPUT REGISTER.

TK-8612

Figure 5-9 RCVR CLK Generator




NOTE:
LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX–1034A

Table 5-2 LINK Clocks

| Frequency (MHz) | Period (ns) | Clock |
|---|---|---|
| 70.00 | 14.28 | MDECODER CLOCK |
| 35.00 | 28.57 | — |
| 8.75 | 114.28 | RCVR CLK |
| 8.75 | 114.28 | XMIT CLK |

The shift register shifts a logic low through the flip-flop chain. When the low is in the rightmost flip-flop, the other three flip-flops are set. Outputs from the three set flip-flops are ANDed together conditioning the first flip-flop to reset on the next clock pulse and reinsert the low into the flip-flop chain. The cycle is then repeated.

The left and right portions of Figure 5-10 illustrate the operation cycle of the shift register. (The center portion illustrates the synchronization function.) Waveforms 1, 2, 3, and 4 relate to the corresponding points in Figure 5-9. Also shown are the MDECODER CLOCK and SYNC from the Byte Framer and the time periods when the RDAT (7:0) bytes are in the RCVR Input Register. These three signals are time related and are shown as they appear in the Byte Framer timing diagram (Figure 5-8). The 35 MHz clock and the shift register waveforms are time related to each other but are independent of the Byte Framer timing. The SYNC signal is used to synchronize shift register action with the data bytes from the Byte Framer.

As shown in Figure 5-9, when SYNC is asserted, the 35 MHz clock that asserts E151-3 sets two sync flip-flops. The next 35 MHz clock sets a pulse width flip-flop that negates E151-3, forming an E151-3 pulse to the shift register. The E151-3 pulse synchronizes the register by forcing a reset condition on the first flip-flop and a set condition on the other three flip-flops. The next 35 MHz clock places the register into the conditioned state, introducing a logic low into the first flip-flop. Regardless of where the register is in its cycle, it is restarted at the beginning of the cycle. The assertion of SYNC followed by the assertion of E151-3 is illustrated in Figure 5-10. Note that conditions forced onto the shift register by the E151-3 pulse are clocked in by the next 35 MHz clock pulse (the first flip-flop is reset and the other three are set). As seen in Figure 5-10, the logic low had reached the second flip-flop when the register cycle was interrupted and reset to its starting point. From this point on, the register cycles are in synchronization with the byte frame. This results in the generation of RCVR CLK pulses approximately centered in the time period when the packet bytes (RDAT <7:0>) are in the RCVR input register.

#### 5.2.1.8 CRC Check

The packet bytes on the RDATA Bus, up to and including the four 8-bit CRC bytes, are input to the CRC Checker. If no errors are detected by the checker, it asserts CRC STATUS to the Message Receive State Logic, indicating reception of a valid, error-free packet.

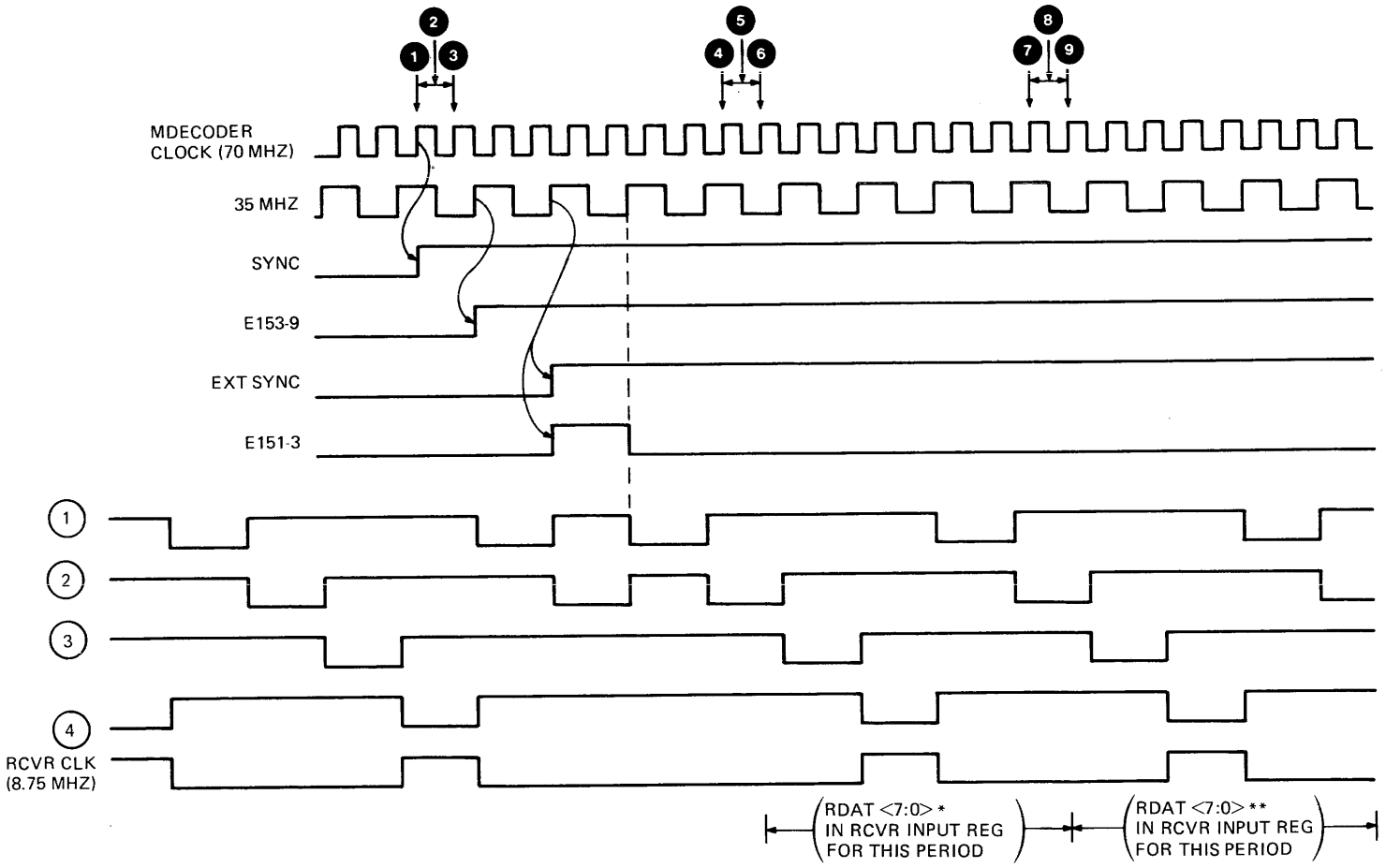
#### 5.2.1.9 Destination Compare

The node address and the complement of the node address are set into two 8-contact node address switches. Outputs from these address switches permit a destination compare. The 8-bit output of the complement node address switch is sent to the True Destination Compare Logic as CNODE ADDRESS <7:0>; the 8-bit output of the true node address switch is sent to the Complement Destination Compare Logic as NODE ADDRESS <7:0>.

True destination and complement destination bytes go from the RDAT Bus to both destination compare logic circuits. The state PALs enable the compare logic outputs so that when the true destination byte is on the RDAT Bus, the output of the True Destination Compare Logic is enabled. If the true destination byte matches CNODE ADDRESS <7:0> from the complement node address switch, TDST CMP is asserted indicating a true address match. Likewise, when the complement destination byte is on the RDAT Bus, the output of the Complement Destination Compare Logic is enabled. If the complement destination byte matches NODE ADDRESS <7:0> from the true node address switch, CDST CMP is asserted indicating a complementary address match. Both true and complement destination matches assert DST CMP to the message receive and ACK Receive State Logic.

A polarity reversal in the compare logic applies the output of the true node address switch to the Complement Destination Compare Logic and applies the output of the complement node address switch to the True Destination Compare Logic. The output of the node address switches is coupled by XOR gates to the compare logic. This allows swapping the true address and the complement address for maintenance testing.

## Figure 5-10 RCVR CLK Synchronization




1. LAST BIT OF SYNC CHARACTER BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
2. SYNC CHARACTER BYTE IN RCVR SERIAL SHIFT REGISTER.
3. FIRST BIT OF PACKET LENGTH/TYPE BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
4. LAST BIT OF PACKET LENGTH/TYPE BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
5. PACKET LENGTH/TYPE BYTE IN RCVR SERIAL SHIFT REGISTER.
6. FIRST BIT OF PACKET LENGTH BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
7. LAST BIT OF PACKET LENGTH BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
8. PACKET LENGTH BYTE IN RCVR SERIAL SHIFT REGISTER.
9. FIRST BIT OF TRUE DESTINATION BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.

* PACKET TYPE/LENGTH BYTE
** PACKET LENGTH BYTE

MR-14461

#### 5.2.1.10 ACK Source Comparison

ACK source compare logic is used only during ACK packet reception. An ACK packet is transmitted from its source acknowledging an information packet transmitted from this node. While the information packet is in the transmit channel, the destination address is saved and sent to the ACK source compare logic.

ACK Source Compare Logic receives inputs from the transmit channel and the RDAT Bus. When the source byte of the ACK packet is on the RDAT Bus, compare logic output is sampled. ACK SOURCE CMP is asserted if the source address of the ACK packet matches the destination address of the preceding information packet.

#### 5.2.1.11 Receive Data Parity and Channel Output

The Receiver Output Data Register transfers data bytes from the RDAT Bus to the Packet Buffer. These bytes are output from the register as RCVR DATA <7:0>. In addition, the data bytes are sent from the RDAT Bus into a Receiver Data Parity Generator where odd parity is generated on each byte. A ninth input to the parity generator (VALID RCVR PARITY) permits introduction of parity errors used for maintenance testing. Parity generator output is applied to a parity flip-flop that outputs RCVR DATA PARITY to the packet buffer.

### 5.2.2 Transmit Channel

The transmit channel takes the data on the XMIT DATA BUS <7:0> and puts it in a serial format for transmission over the CI data paths. Refer to Figure 5-11.

#### 5.2.2.1 Transmit Data Input

Transmit data (XMIT DATA <7:0>) from the PILA is put into the transmit channel by the XMIT Data Input Latch and then transferred to the XMIT Data Bus as XMIT DATA BUS <7:0>.

This latch is transparent because the data on the bus follows the XMIT DATA <7:0> as long as the latch is enabled. ENA XMIT DATA LATCH from the Transmit Control Logic and XMIT CLK high enable the XMIT data input latch. When XMIT CLK is low, this latch is disabled (closed).

#### 5.2.2.2 Bit Sync, Sync Character, and Trailer Bytes

Bit synchronization bytes, trailer bytes, and the sync character byte reside in a 32 by 8 PROM. ENA SYNC/TR from the Transmit Control Logic enables PROM output. A 5-bit address input to the PROM selects the output bytes placed onto the XMIT Data Bus.

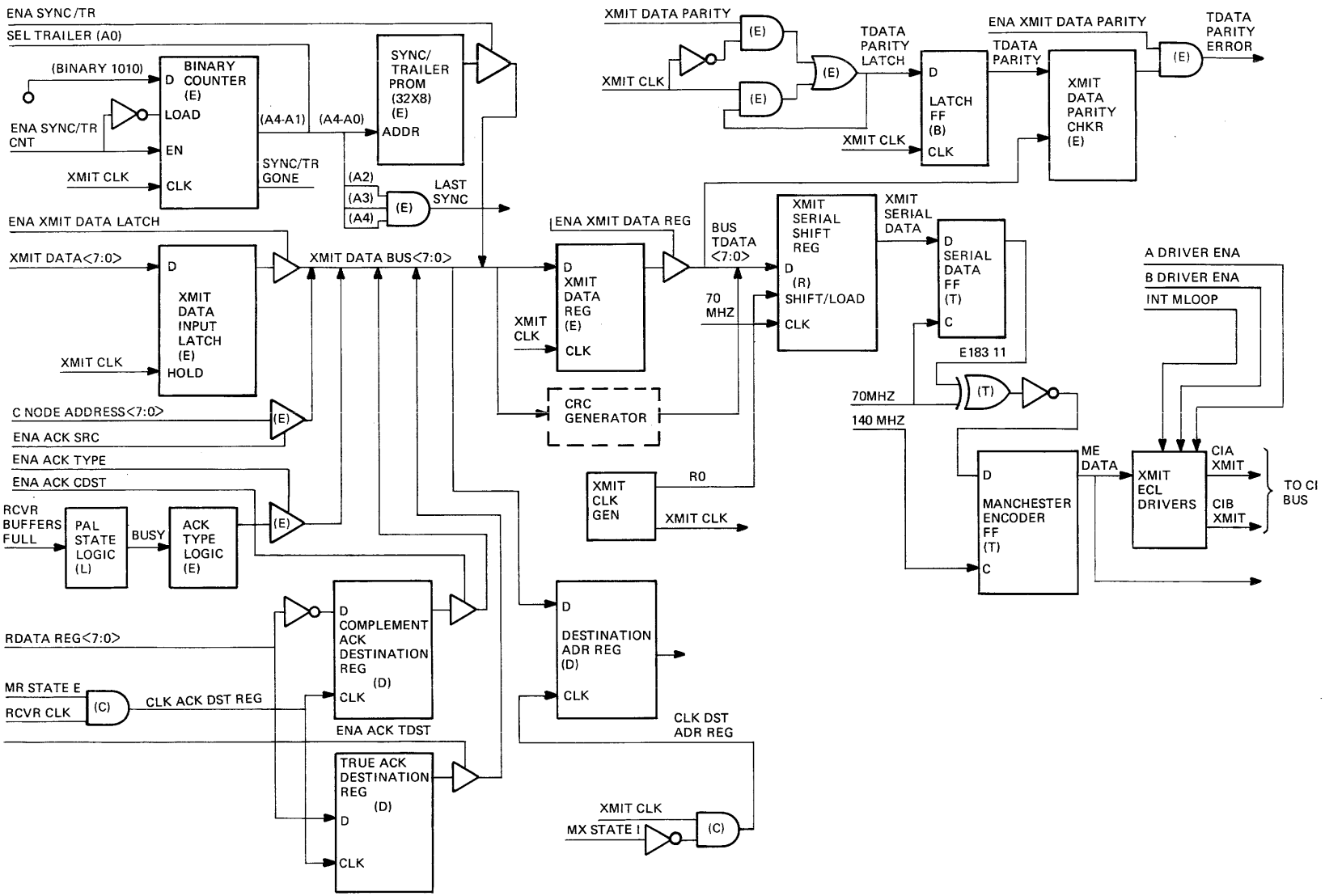
Figure 5-12 diagrams the 32 eight-bit locations in the Sync/Trailer PROM.

The five 8-bit sync bytes and the sync character byte are located in the upper area of PROM space. The lower part of PROM is reserved for a possible header extension to 16 bytes (15 bytes of bit synchronization and one sync character byte). Note in Figure 5-12 that the bit sync bytes, starting at address 10101, alternate with six trailer bytes starting at address 10100. PROM address bits <A4:A1> are obtained from a binary counter enabled by ENA SYNC/TR CNT from the Transmit Control Logic. When this signal is false, the counter is loaded with starting address 1010. When the signal is asserted, the counter counts up from 1010, addressing every other PROM location. The least significant address bit of the PROM (A0) is SEL TRAILER from the Transmit Control Logic. When SEL TRAILER is false, PROM sync bytes are addressed. When SEL TRAILER is true, PROM trailer bytes are addressed.

Address bits <A4:A2> are monitored and assert LAST SYNC to the Transmit Control Logic when all three bits are true. As shown in Figure 5-12, this occurs when the last sync byte (sync byte 5) is addressed.

When the binary counter has counted past the last trailer byte or past the sync character byte, it overflows and asserts SYNC/TR GONE to the PAL Logic.

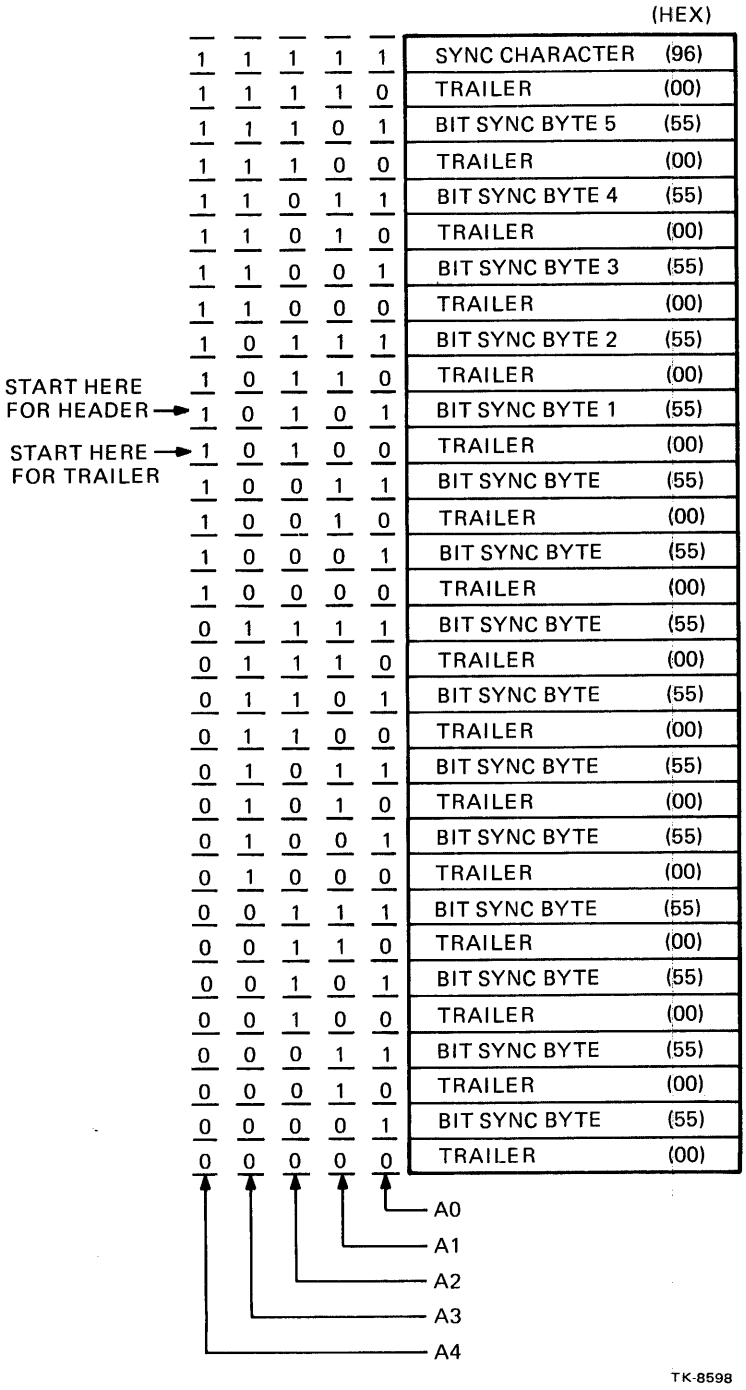
## Figure 5-11 Transmit Channel Block Diagram




NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX-961A

Figure 5-12 Sync/Trailer PROM Space



#### 5.2.2.3 ACK Packet Inserts

The LINK inserts the packet type, source, and destination bytes into the ACK packet. When information packets are transmitted, the K.pli inserts these bytes and LINK hardware is not involved.

##### 5.2.2.3.1 Packet Type Byte

The packet type byte is obtained from the ACK Type Logic outputting a one in bit position seven to signify an ACK (or NACK) packet. Bit position six is a function of BUSY, a signal derived from the packet buffer RCVR BUFFERS FULL. BUSY is asserted when the receive buffers in the PILA are full and cannot accept the information message just received. This causes a one in bit position six signifying NACK packet transmission. If BUSY is false, bit position six is zero indicating ACK packet transmission. Remaining bits in the ACK type logic <5:0> are always zero.

##### 5.2.2.3.2 Source Byte

The ACK source byte is the complement node address (CNODE ADDRESS <7:0>) obtained from the complement node address switch on the LINK. This source byte is gated onto XMIT Data Bus by ENA ACK SRC from the PAL State Logic.

##### 5.2.2.3.3 Destination Bytes

ACK destination bytes are derived from the source byte of the associated information packet. This source byte is taken from the RDAT Bus in the receive channel and clocked into the Destination Address Registers by CLK ACK DST REG. RDAT REG <7:0> is entered directly into the True ACK Destination Register and the inverse is entered into the Complement ACK Destination Register. The true destination byte and the complement destination byte are gated to the XMIT data bus by ENA ACK TDST and ENA ACK CDST, respectively. PAL State Logic asserts the gating signals for inserting the bytes into the ACK packet at the appropriate times.

#### 5.2.2.4 Destination Address Register

This register saves the destination address of the information packet being transferred. CLK DST ADR REG is asserted at the correct time to clock the true destination byte into the register. The destination byte is compared with the source of the ACK packet in the receive channel. A match indicates the correct node responded to the message transmission.

#### 5.2.2.5 Transmit Data Parity Check

Data from the BUS TDATA Bus is applied to the XMIT Data Parity Checker where packet bytes are checked. Parity bits, XMIT DATA PARITY, sent from the packet buffer, are applied to a latch flip-flop as TDATA PARITY LATCH. An OR feedback network holds TDATA PARITY LATCH true for both transmissions of XMIT CLK, allowing the latch flip-flop to set (if parity is one). The latch flip-flop outputs the parity bit (TDATA PARITY) to the parity checker. Parity is then checked when ENA XMIT DATA PARITY is asserted and enables the parity checker output. TDATA PARITY ERROR is asserted to the Message State Logic when a parity error occurs.

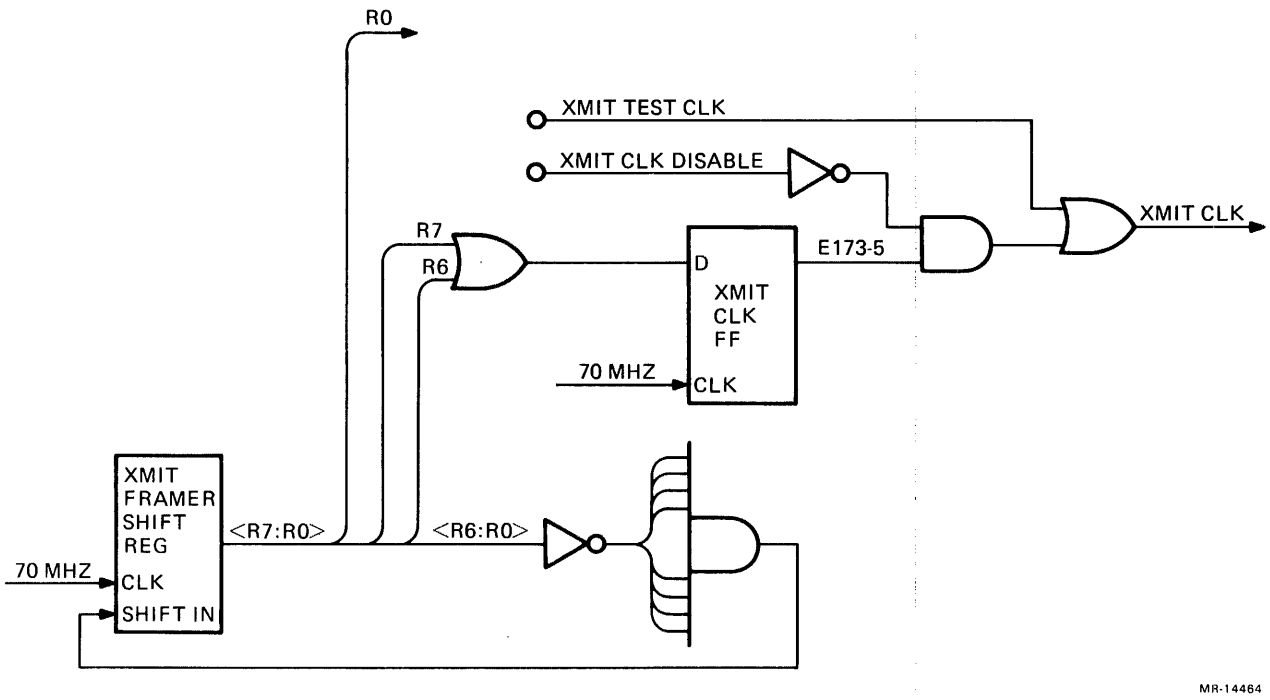
#### 5.2.2.6 CRC Generation

Packet bytes on the XMIT Data Bus, starting with the packet type byte and ending with the last byte of the body, are input to the CRC Generator. The generator produces a 32-bit CRC longword unique to the packet being transmitted. This longword is inserted into the packet one byte at a time at the end of the packet body.

### 5.2.2.7 XMIT CLK Generator

Figure 5-13 is a block diagram of the XMIT CLK Generator. XMIT CLK is derived from a 70 MHz input received from a crystal oscillator network in the RCVR CLK Generator. The Transmit Clock Generator produces XMIT CLK pulses at 8.75 MHz (period = 114.28 ns). It also outputs an R0 pulse to load the XMIT Serial Shift Register from the TDATA Bus.

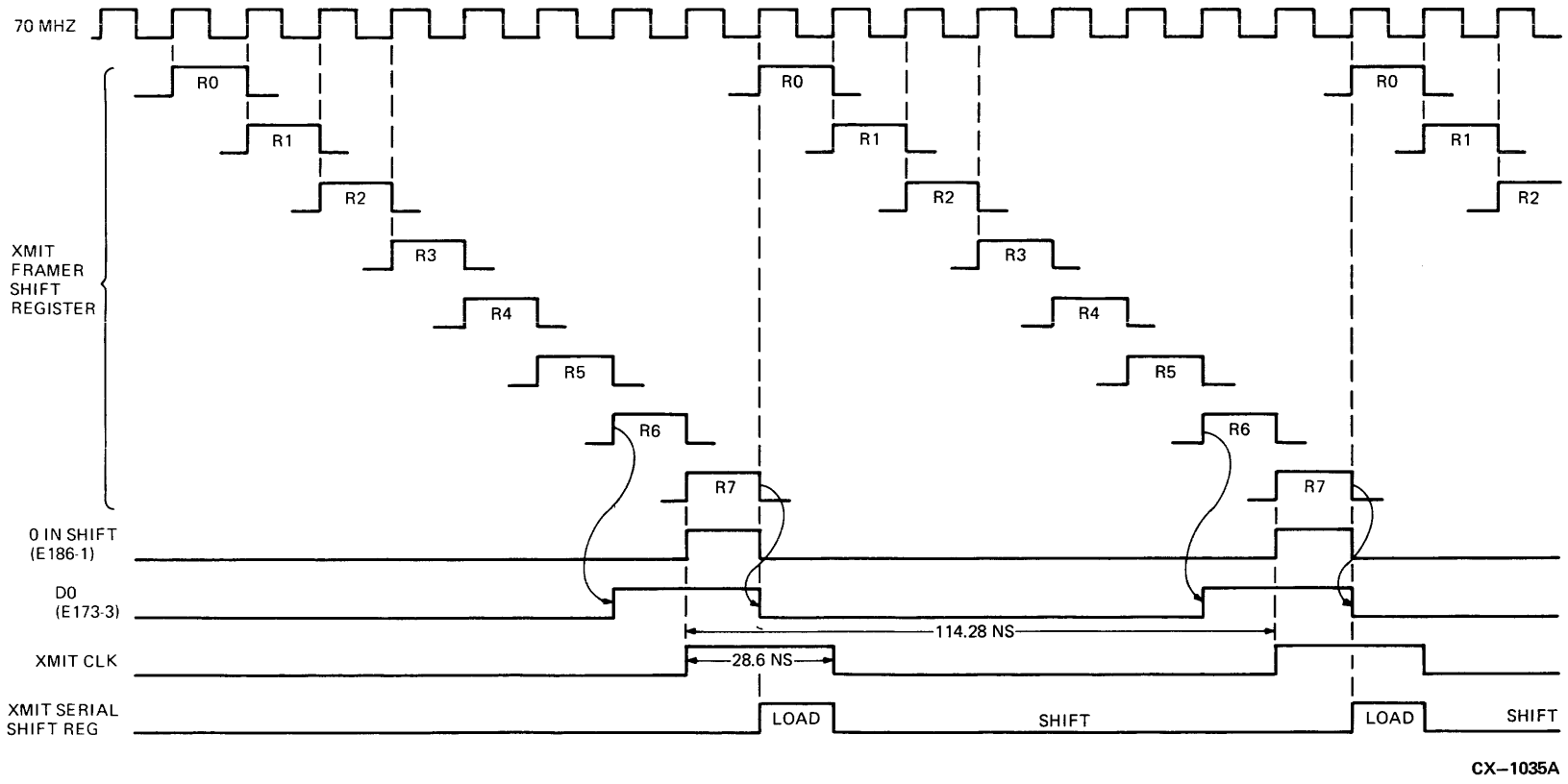
**Figure 5-13 XMIT CLK Generator Block Diagram**




Timing for the XMIT CLK Generator and the XMTR Framer Shift Register is illustrated in Figure 5-14. The XMTR Framer Shift Register is clocked at 70 MHz with an 8-bit parallel output <R7:R0>. The inverses of bits <R6:R0> are ANDed so that when all seven bits are false, a one is input to the Framer Shift Register. This one is clocked up to the R7 output, at which time another one is generated for the Framer Shift Register. Figure 5-14 illustrates the time relationship of XMIT CLK relative to Framer Shift Register outputs.

R6 and R7 from the Framer Shift Register are applied to the D input of the XMIT CLK flip-flop causing it to set for two 70 MHz clocks. Output of the flip-flop is XMIT CLK. For maintenance testing, the output of the XMIT CLK flip-flop can be disabled and an XMIT TEST CLK substituted.

**Figure 5-14 XMIT CLK Generator Timing Diagram**



#### 5.2.2.8 Parallel To Serial Data Conversion

Eight-bit data bytes from the TDATA Bus are input to the XMIT Serial Shift Register. Every eighth 70 MHz clock cycle R0 from the XMIT CLK Generator is asserted, loading the shift register with a data byte from the TDATA Bus. After loading, the register returns to the shift state, sending out the data byte a bit at a time as XMIT SERIAL DATA. As the last bit is shifted out, R0 is asserted again, loading the next packet byte into the serial shift register. Figure 5-14 illustrates load and shift timing of the serial shift register.

XMIT SERIAL DATA is applied to a serial data flip-flop clocked by 70 MHz. Output from this flip-flop (E183-11) is sent to the Manchester Encoder.

#### 5.2.2.9 Manchester Encoder

The Manchester Encoder modulates the serial data with the Data Bit Rate Clock, producing the signal format placed on the CI Bus. Encoder logic XORs E183-11 (serial data flip-flop output) with the 70 MHz clock. Output of the XOR gate is inverted and applied to the Manchester Encoder flip-flop. This flip-flop is clocked at 140 MHz or twice the data rate, as required for phase encoded data (Section 5.2.1.4). Output of the encoder flip-flop (ME DATA) is the packet data ready for transmission onto the CI Bus.

Manchester encoder timing is illustrated in Figure 5-15. E183-11 output of the serial data flip-flop is shown for the example data bits. After E183-11 is XORed with the 70 MHz clock, the inverse of the XOR output is used for the encoder flip-flop input. Clocking the encoder flip-flop at 140 MHz results in the ME DATA waveform. ME DATA signal format is identical to that of the serial data received from the CI Bus in Figure 5-5.

#### 5.2.2.10 XMIT ECL Drivers

Figure 5-16 diagrams an ME DATA transfer to the CI Bus by the XMIT ECL drivers. Note the XMIT drivers are divided into two channels feeding the CI Bus A and B paths. The K.pli selects the path using the Transmit Control Logic to enable the driver in the selected channel.

ME DATA is routed to drivers in both channels and through coupling transformers to the CI Bus as CIA XMIT and CIB XMIT. When the Transmit Control Logic selects channel A, signal A DRIVER ENA is asserted (B DRIVER ENA false) and in turn asserts E151-1 from the channel A AND gate. Assertion of E151-1 causes outputs from both channel A XOR gates and enables the channel A driver. In the same manner, assertion of B DRIVER ENA asserts E151-2 output of the channel B AND gate and enable the channel B driver.

Redundancy in driver enabling logic prevents a single component failure from causing simultaneous enabling of the A and B channels. If, due to a logic component failure, the outputs of both channel A and channel B are asserted (E151-1 and E151-2 true), one of the channel A XOR gates and one of the channel B XOR gates are inhibited. The enabling inputs to the channel drivers are held high inhibiting the drivers and isolating the node from the CI Bus. Operation of ECL logic is described in Section 5.2.1.3. During internal maintenance loop operation, the Port Control Logic disables both output drivers. In this state, ME DATA from the transmit channel is looped back into the receive channel. The port asserts INT MLOOP, inhibiting both E151 AND gates and shutting off the output drivers. In addition, signal lines into both A and B channel drivers are held high by INT MLOOP inhibiting any signal data variations into the drivers.

### 5.3 CRC GENERATOR AND CHECKER

The LINK is responsible for cyclic redundancy generation and checking. This CRC logic is common to both the receive and transmit paths of the CI Bus. Figure 5-17 is a block diagram of the CRC Generator and Checker.

**Figure 5-15 Manchester Encoder Timing Diagram**

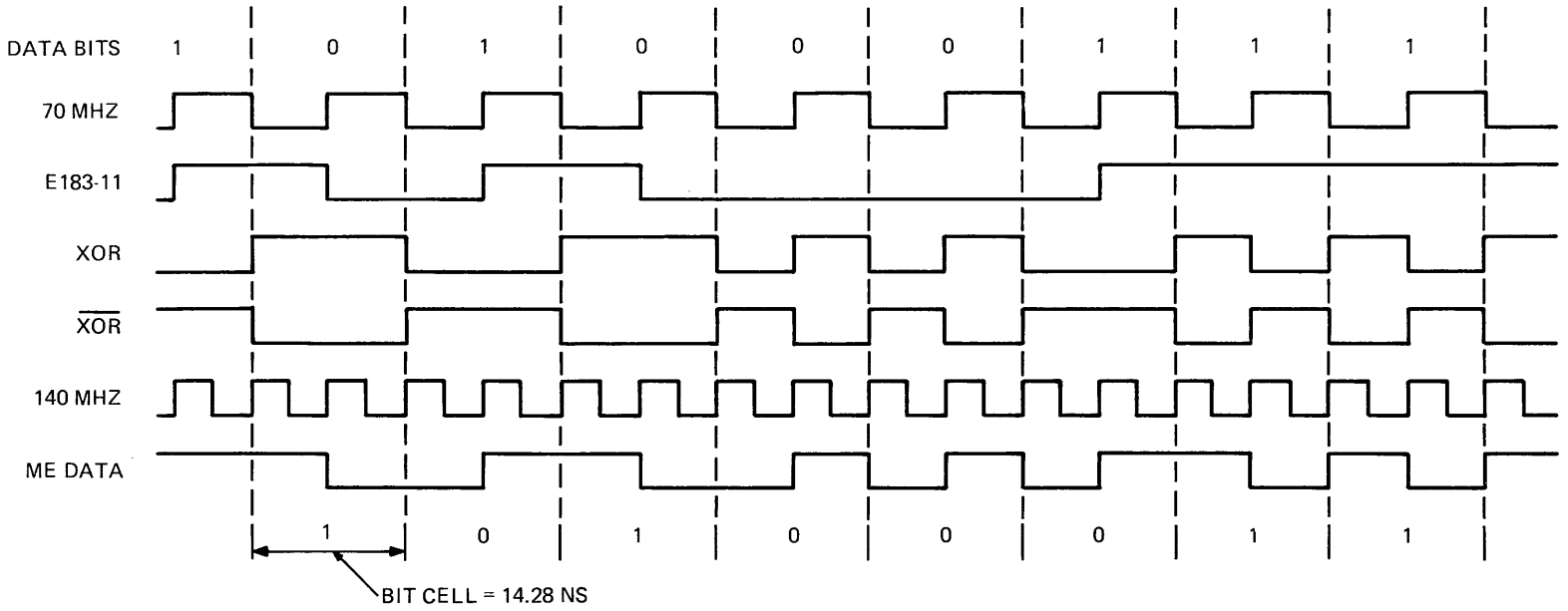
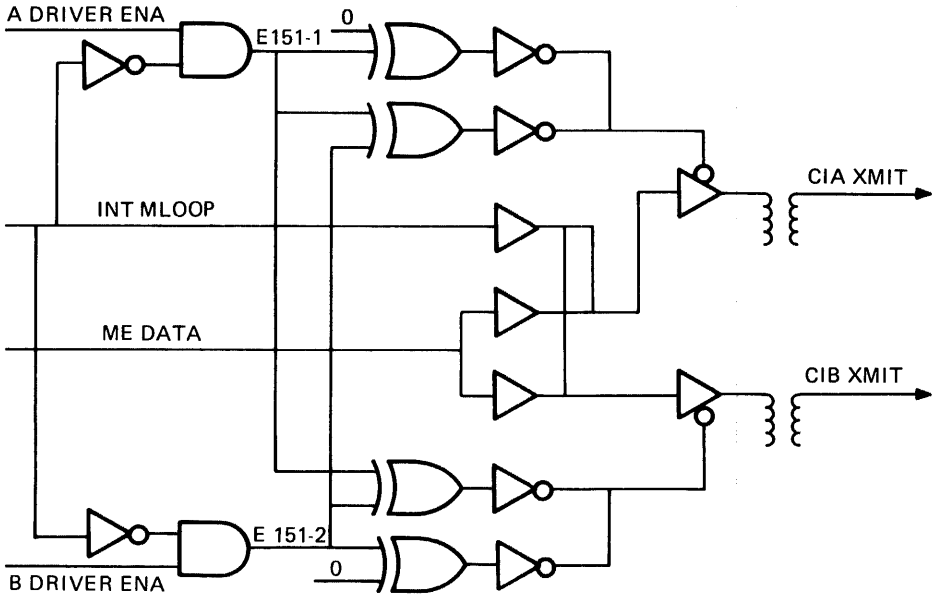


Figure 5-16 XMIT ECL Drivers




NOTE:
THE LOGIC IN THIS FIGURE IS CONTAINED
ON SHEET T OF THE ENGINEERING DRAWINGS.

MR-14466

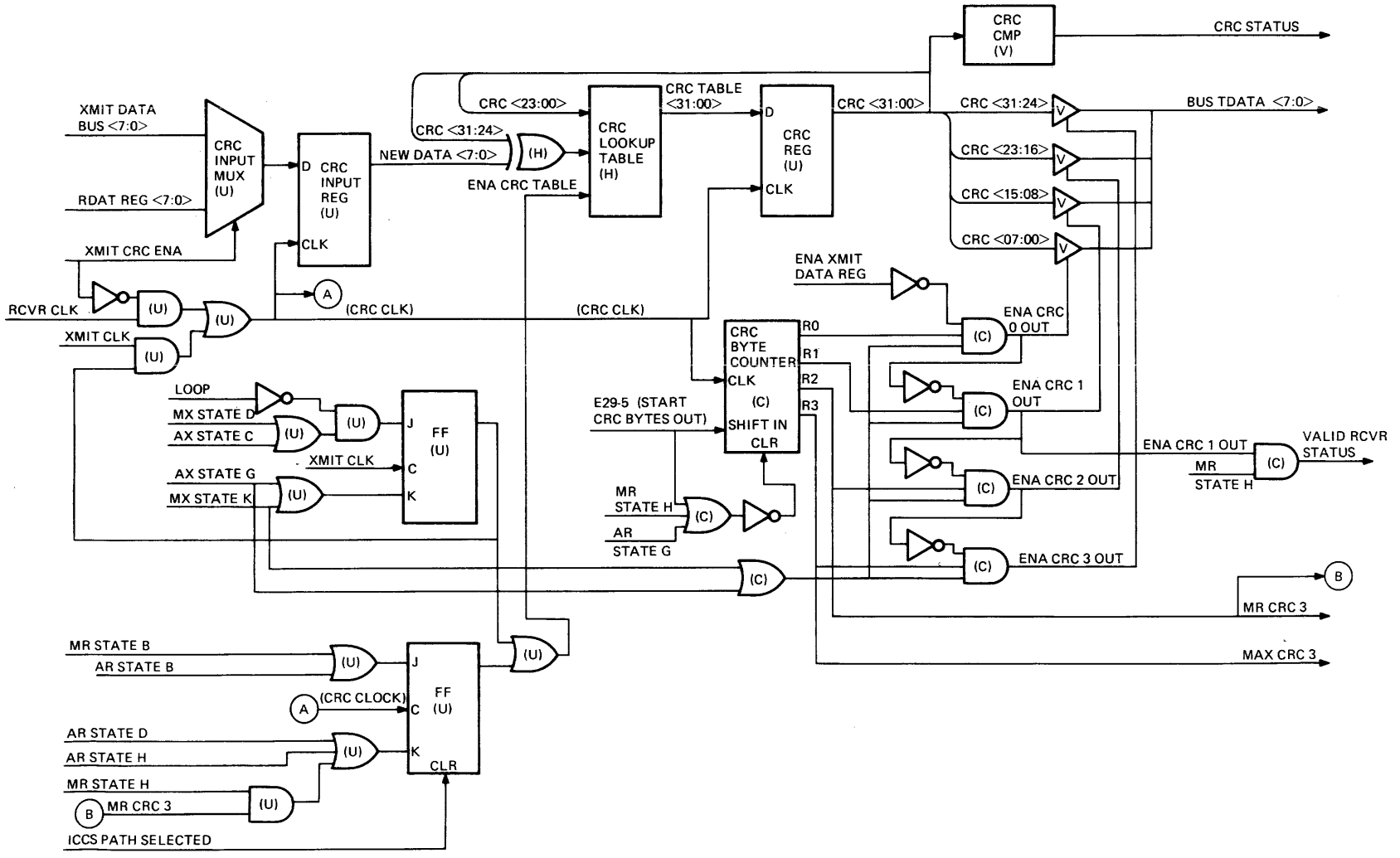
### 5.3.1 CRC Generator

Packet bytes from the XMIT Data Bus in the transmit channel are input to the CRC Input Multiplexor as XMIT DATA BUS <7:0>. Transmit Control Logic asserts XMIT CRC ENA to the multiplexor for selecting the bytes from the transmit channel. The multiplexor output is sent to the CRC Input Register that outputs the bytes as NEW DATA <7:0>.

Using an XOR gate, NEW DATA <7:0> is applied to a CRC lookup table. Lookup Table Logic is applied to a CRC Register that outputs CRC <31:00>. Note the CRC Register initially is preset to all ones. CRC <31:00> is looped back into the Lookup Table Logic in two parts: the first three bytes (CRC <23:00>) are applied directly to the logic, and the uppermost byte (CRC <31:24>) is XORed with the new input byte from the CRC Input Register. Thus, new data bytes are continuously integrated into the compilation of the previous data bytes. In this way, the CRC-generated longword is always a function of the packet bytes received from the transmit channel.

The CRC longword is also coupled by four sets of drivers to the BUS TDATA Bus in the transmit channel. When enabled, each driver places a byte onto the BUS TDATA Bus to insert a CRC byte into the packet being transmitted. These drivers are enabled from a CRC byte counter that receives a SHIFT IN input (E29-5) when the last packet body byte is on the BUS TDATA Bus. Input is shifted through the counter by CRC CLOCK asserting R0 through R3 in sequence. R0 through R3 are enabled at the appropriate time through the PAL State Logic.

## Figure 5-17 CRC Generator/Checker



In addition, ENA XMIT DATA REG must be false before the R0 AND gate is enabled to place the first CRC byte (CRC <7:0>) onto the BUS TDATA Bus. This ensures the BUS TDATA Bus is isolated from the XMIT Data Register before CRC logic is connected to the bus (Figure 5-11). Likewise, each AND gate must be disabled in sequence before the next AND gate can be enabled, ensuring only one source drives the TDATA Bus at a time. CRC Generator Logic is clocked by CRC CLOCK (seen as XMIT CLK during the transmit states).

### 5.3.2 CRC Checker

Packet bytes from the RDAT Register Bus in the receive channel are input to the CRC Input Multiplexor as RDAT REG <7:0>. Transmit Control Logic negates the XMIT CRC ENA input to the multiplexor selecting the bytes from the receive channel. The multiplexor output is sent to the CRC Input Register which outputs the bytes as NEW DATA <7:0>.

CRC logic generates the CRC longword as described in previous sections. The last four bytes input to the CRC become the CRC longword generated for the packet. When the CRC longword is entered into the CRC lookup table, if the packet is error free, an output of DEBB 28E3 (hex) results. The longword value is checked by a CRC comparator that asserts CRC STATUS if the proper value is obtained.

## 5.4 ARBITRATION

To prevent collisions on the bus, only one node at a time should transmit. When the K.pli commands a node to transmit an information packet, the LINK enters an arbitration process to gain control of the bus. This process is not necessary for ACK packet transmission as it is assumed the bus has already been acquired for such transfers. As explained in following sections, bus acquisition by a node is accomplished by taking turns between the competing nodes through the arbitration process. No hardware or software control allows a node to seize the bus and exclude other nodes.

### 5.4.1 Arbitration Process

In the arbitration process, a node counts down a specific number of bus quiet slots. A quiet slot is a time period approximately 800 ns in length with no activity on the bus. This is sufficient time for a one-way trip on the bus and to detect a carrier presence. This quiet slot is the time period allocated for an arbitrating node to detect transmission by another node.

When a node completes its quiet slot countdown (reaches zero), it wins the bus and may transmit. If the node detects bus activity before the countdown is completed, the arbitration process is interrupted and restarted when the bus is again quiet. When several nodes are competing for the bus, all arbitration countdowns other than that of the winner are interrupted. All losing nodes restart countdowns simultaneously when the bus is again quiet, placing them in synchronization with each other.

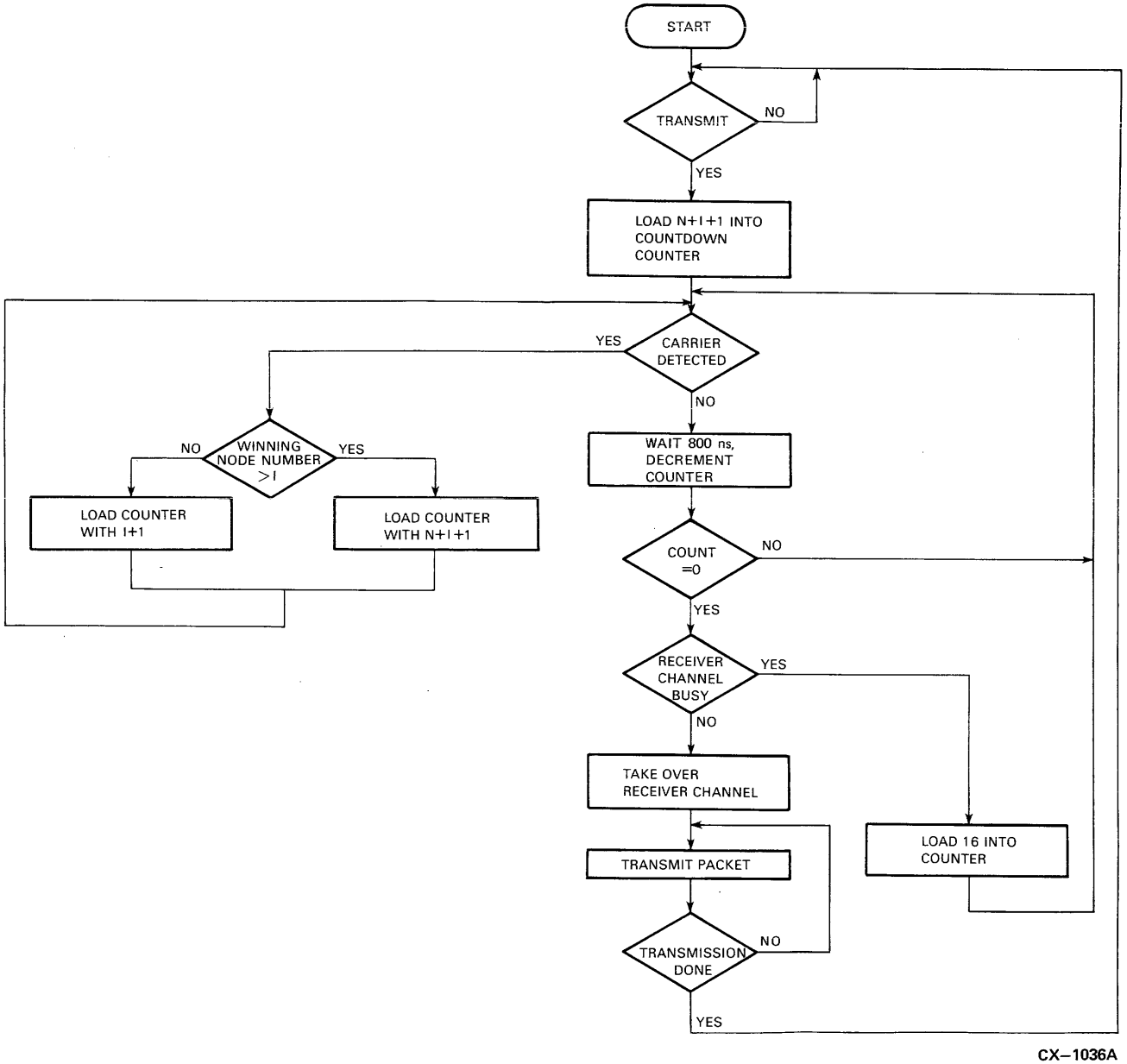
### 5.4.2 Arbitration Algorithm

The arbitration countdown is a dual count round robin algorithm giving the the bus to the lower numbered node when more than one node is attempting transmission. This algorithm is illustrated in Figure 5-18. Other nodes, however, can each access the bus before the winning node can gain the bus again. This process is implemented by the number of quiet slots each node must count.

The number of quiet slots counted down is determined by the number of the node attempting transmission or by the number of the node that last had the bus. A node may count N + I + 1 quiet slots or I + 1 slots where:

N = 16 (maximum number of allowable nodes)
I = node number

**Figure 5-18 CI Bus Arbitration Flowchart**



When a node begins arbitration, it counts N + I + 1. If the countdown is interrupted, the node determines the number of the winning node. When the winning node is a higher number, the node restarts an N + I + 1 countdown. If the winning node is a lower number, the node restarts its countdown at I + 1. When several nodes are competing for the bus, the lowest numbered node wins the bus but it must count down N + I + 1 to win the bus again. Higher nodes restart arbitration with the I + 1 countdown, and all win the bus before the first winner can once more gain the bus. As each node wins the bus, the N number is added to its countdown value and the next higher numbered nodes wins the bus. Thus, each competing node has a turn at the bus, starting with the lowest numbered node and going to the highest. Note whenever a node completes its countdown (reaches zero), it checks whether the receiver is free (ALT PATH BUSY false) before transmitting. Transmission from a node does not occur unless the node receiver is free to accept the ACK response. Although the node completed its countdown and gained one path on the bus, the node receiver could be busy receiving a packet on the other path. When this happens, the transmission is delayed by loading 16 into the node counter and continuing the countdown.

### 5.4.3 Arbitration Logic

Figure 5-19 is a block diagram of the Arbitration Logic. Prior to receiving a transmit command from the K.pli, the LINK is in the idle state (MX STATE A). In this state, LOAD ARB COUNT true loads the arbitrator for the quiet slot countdown. The basic slot counter is loaded with $1001_2$; the down counter is loaded with N + I + 1.

This down counter contains two sections: the lower four bits and the fifth bit. The four-bit section is loaded with the node address (NODE ADDRESS <3:0>). The fifth-bit section is loaded from an N load multiplexor that supplies the N term in the arbitration countdown expression. Multiplexor select inputs are shown in Table 5-3.

**Table 5-3 N Load Multiplexor Selection**

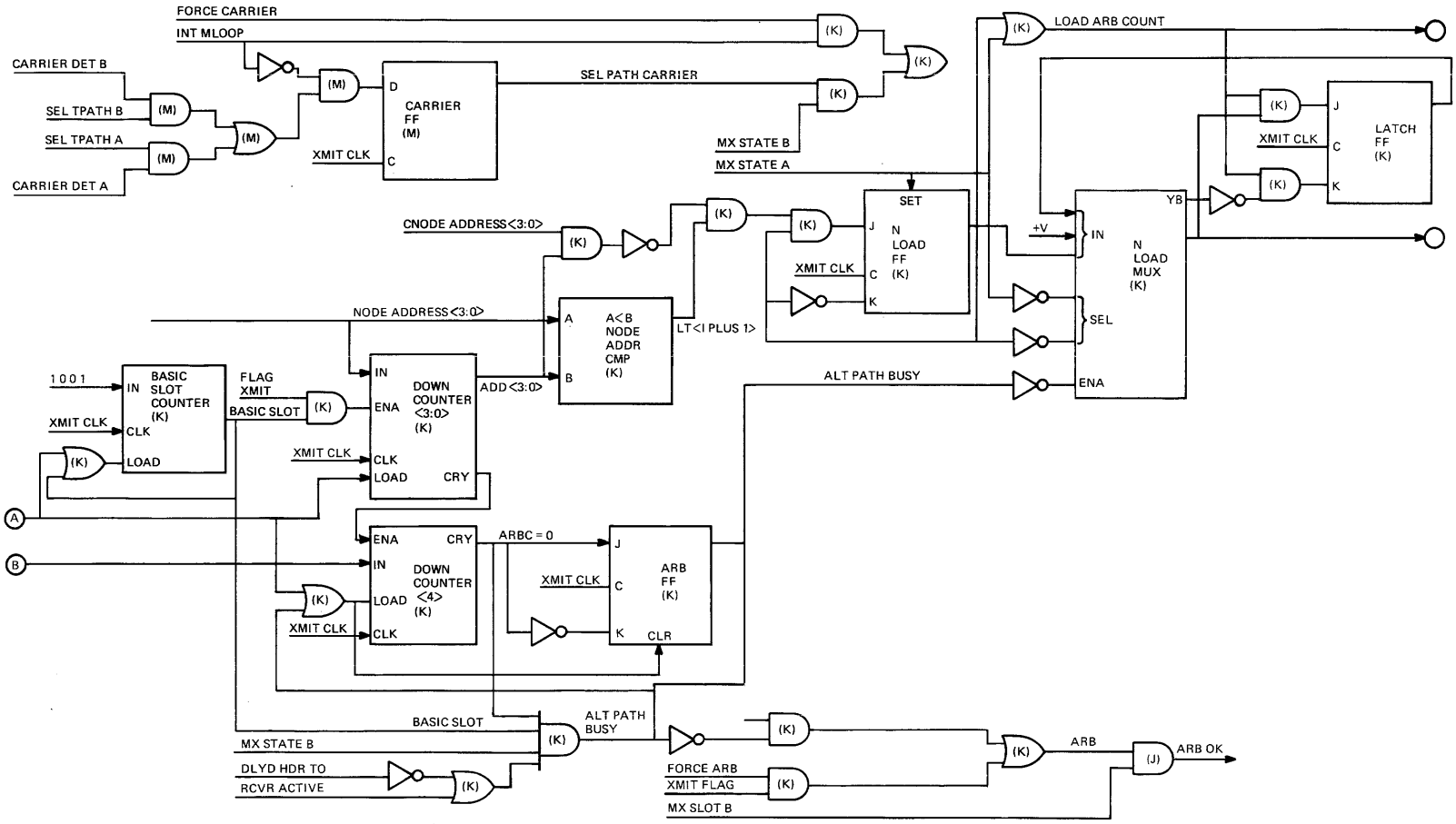
| Select Code MX STATE A | SEL PATH CARRIER | Input Selected |
|---|---|---|
| True | X | +V |
| True | X | +V |
| False | True (load arb.) | N Load FF |
| False | False | Latch FF |

X = Does not matter what the input is

While the LINK is idling in MX STATE A, the multiplexor selects the +V input to load a one into the fifth-bit section of the down counter. The one represents the N term in the N + 1 countdown expression. When the LINK shifts to MX STATE B, LOAD ARB COUNT negates and the arbitrator starts its countdown. The slot counter is clocked from $1001_2$ by XMIT CLK and outputs BASIC SLOT after seven clock pulses. BASIC SLOT is looped back, reloading the counter with $1001_2$ and the cycle is repeated. The time period of XMIT CLK is 114.28 ns so BASIC SLOT is asserted every 800 ns (7 X 114.28).

Each time BASIC SLOT is asserted, it enables the four-bit section of the down counter which is decremented by XMIT CLK. When this section of the down counter reaches zero, the next assertion of BASIC SLOT asserts the carry (CRY) output decrementing the fifth-bit section. If the fifth-bit section contains a one (N + I + 1 count), the one becomes a zero, the four-bit section becomes all ones, and the countdown continues. If the fifth-bit section contains a zero (I + 1 count), the CRY output goes true asserting

## Figure 5-19 CI Bus Arbitration Block Diagram




NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX-966A

ARBC = 0 (arbitration counter equals zero) setting the ARB flip-flop on the next XMIT CLK cycle. If the alternate bus path is not busy (ALT PATH BUSY false), ARB and ARB OK assert indicating a successful countdown and causing the LINK to shift to MX STATE C.

Note that when the counter reaches zero, one more assertion of BASIC SLOT is required to assert the CRY output and cause ARB to go true. This additional assertion of BASIC SLOT represents the one term in the two countdown expressions.

#### 5.4.3.1 Arbitration Countdown

If a carrier from another node is detected during the countdown, the arbitrator is reloaded and the countdown restarted. The node address comparator determines whether the interrupting (winning) node is a higher- or lower-numbered node to determine the new countdown value. (See Section 5.4 for a general description of the arbitrator.)

##### 5.4.3.1.1 LINK Arbitration Comparator

The comparator compares the node address with ARB CMP ADD <3:0> from the four-bit section of the down counter and asserts LT or less than (I PLUS 1) if this node number is less than the winning node number. For example, assume this node is number five and the winner is node number two. ARB CMP ADD <3:0> is counted down to three. The comparator A input is greater than the B input; therefore, LT (I PLUS 1) is false. This node is not less than the winning node. If the winning node is node seven, for instance, ARB CMP ADD <3:0> is 14 (as the fifth bit was decremented). Comparator A input is less than the B input; therefore, LT (I PLUS 1) is true. This node is less than the winning node. The LT (I PLUS 1) signal determines which countdown value is reloaded into the down counter for the next countdown.

#### 5.4.3.2 Carrier Detection

When a carrier is detected (interrupting the countdown), CARRIER DET A or CARRIER DET B is asserted. If the carrier is detected on the SEL TPATH selected by the LINK Control PAL, SEL PATH CARRIER is asserted. In turn, this asserts LOAD ARB COUNT and reloads the basic slot counter and both sections of the down counter. The fifth-bit section of the down counter is again loaded from the N Load Multiplexor. However, in this case the multiplexor selects its input from the N Load Flip-Flop (Table 5-3)

During the countdown, SEL PATH CARRIER false holds the N Load Flip-Flop reset. When this signal asserts, the J input to the flip-flop checks LT (I PLUS 1) from the node comparator. If LT (I PLUS 1) is true (this node less than the winning node), the flip-flop is set and a one is loaded into the fifth-bit section. If LT (I PLUS 1) is false, the opposite occurs. Output from the N Load Multiplexor is latched in a latch flip-flop. When SEL PATH CARRIER is negated, the N Load Multiplexor selects the output of the latch flip-flop, thus maintaining the fifth-bit selection after SEL PATH CARRIER is negated.

In the case where the winning node is greater than this node, LT (I PLUS 1) is false and the fifth bit section of the counter is loaded with a zero. Likewise when node zero is beaten by node 15, it must appear that it was beaten by a lower node and restart its countdown as I + 1. However, in this case, LT (I PLUS 1) is true. Logic in the N Load Flip-Flop forces a zero into the fifth bit section of the counter when node zero is beaten by node 15. When this is node zero (CNODE ADDRESS <3:0> equals all ones) and it has just been beaten by node 15 (ARB CMP ADD <3:0> equals all ones), the AND gate transferring LT (I PLUS 1) into the N Load Flip-Flop is inhibited and the flip-flop remains reset. A zero is reloaded into the fifth-bit section of the down counter and node 0 does an I + 1 countdown.

#### 5.4.3.3 Receive Channel Busy - Arbitration Condition

If the LINK receive channel is busy on the the alternate bus path, RCVR ACTIVE is true, causing ALT PATH BUSY true. This condition inhibits the assertion of ARB and loads 16 into the down counter. ALT PATH BUSY loads only the fifth-bit section of the down counter and the four-bit section remains enabled in count mode. ALT PATH BUSY generates the 16 by disabling the N Load Multiplexor causing a zero output into the fifth-bit section. With the countdown successfully completed, the four-bit section is all zeros. As the fifth-bit section is loaded with a zero, the four-bit section is decremented to all ones. Thus, when the entire counter is enabled again, it contains a count of 16.

#### 5.4.3.4 Successful Arbitration Inhibited

The true state of RCVR ACTIVE inhibits a successful arbitration by reasserting ALT PATH BUSY. RCVR ACTIVE negates after the message on the alternate path has been received. The transmission that is arbitrating on the bus, however, still cannot be allowed because the transmit channel must be used to transmit an ACK response. At this point in the message, the receive state sequence is state I. Hence, MR STATE I keeps ALT PATH BUSY true inhibiting the assertion of ARB.

The signal DLYD HDR TO being false also inhibits successful arbitration. This signal is false if a transmission is occurring from this node (A DRIVER ENA or B DRIVER ENA true).

### 5.5 LINK FUNCTIONS

LINK function commands are carried from the K.pli by four LINK control lines (LINK CONTROL <3:0>) and eight port data lines (PORT DATA <7:0>) (Figure 5-20). The K.pli asserts SELECT when a valid function exists on the LINK control lines. A function decoder decodes the LINK control lines and outputs the specific function commanded by the K.pli. LINK function commands are:

- XMIT FCN—initiates arbitration and transmission on one of the CI paths. The CI path used is selected by port data bit seven (0 = path A; 1 = path B).
- RESET XMIT STATUS—resets transmission status bits at the end of a transmission operation.
- ABORT XMIT FCN—aborts a currently active transmission operation.

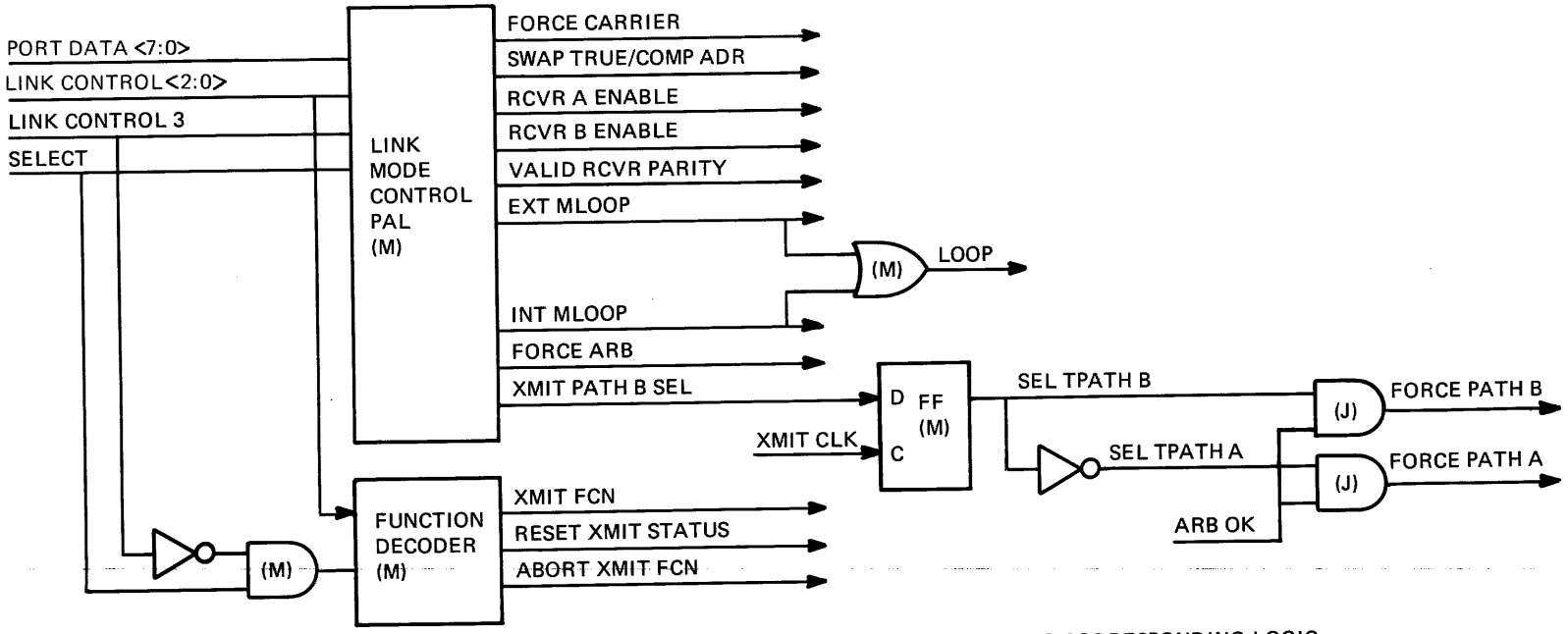
The LINK Mode Control PAL receives the LINK control and port data lines from the K.pli. The port data lines carry control information related to the function commands and specify various maintenance functions for the LINK.

#### 5.5.1 LINK Control Information and Maintenance Functions

LINK control information and maintenance functions are described as follows:

- XMIT PATH B SEL—This signal selects the CI path associated with the XMIT FCN command.
- RCVR A ENABLE—This signal enables path A in the LINK receiver making the node accessible on CI path A.
- RCVR B ENABLE—This signal enables path B in the LINK receiver making the node accessible on CI path B.
- EXT MLOOP—This maintenance function allows the LINK to receive its own transmission by looping on the selected CI path.
- INT MLOOP—This maintenance function allows the LINK to receive its own transmission by looping inside the transmit drivers and input receiver detectors. This operation does not interfere with the CI operation of other nodes.
- FORCE CARRIER—This maintenance function causes the LINK to see a detected carrier.
- FORCE ARB—This maintenance function causes the LINK to force a successful arbitration.

Figure 5-20 LINK Functions




NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX–967A

- VALID RCVR PARITY—This maintenance function generates parity errors in the receive channel.
- SWAP TRUE/COMP ADR—This maintenance function causes the true and complementary address sources to be swapped, resulting in an address mismatch.

SEL TPATH A or SEL TPATH B, the transmission path select signal, asserts a corresponding FORCE PATH signal after the node successfully arbitrates for the bus (ARB OK true). The FORCE PATH signal enables the corresponding path in the receive channel preparatory to ACK response reception.

## 5.6 LINK INTERFACE SIGNALS

Figure 5-21 shows the LINK interface signals, mostly used to interface with the PILA. Figure 5-22 and Figure 5-23 are flow diagrams of typical error-free transmit and receive operations. Interface signals are preceded by an asterisk, and some other signals, internal to the LINK, are included to complete the flow. These two flow diagrams utilize most of the LINK interface signals and explain their basic functions. Interface signals not included in the diagrams are the three clocks (XMIT CLK, PORT CLK, RCVR CLK), the node address (NODE ADDRESS <7:0>), and INITIALIZE, explained as follows:

- PORT CLK—received from the PILA. Used in both the PILA and the LINK.
- XMIT CLK—generated in the LINK. Used in both the PILA and the LINK.
- RCVR CLK—generated in the LINK. Used in both the PILA and the LINK.
- NODE ADDRESS <7:0> - Sent to the K.pli via the PILA and inserted into the transmitted packet as the source byte.
- INITIALIZE - used for system initialization.

## 5.7 LINK OPERATING STATES

The following descriptions of the four LINK operations utilize the state diagrams contained in the engineering drawing set. Various states are shown in the diagrams as circles. A path looping back into a circle holds the LINK in that state as long as the signal condition in that loopback path is true. The LINK goes to its next state if the signal condition shown in the connecting path to the next state is true. Where no loopback paths are shown, the LINK stays in that state for one clock pulse, performs the indicated task(s) and proceeds to the next state.

Also included in the drawing set is a XMIT/RCVR state flow diagram. This diagram indicates flows for the following:

- Message transmission
- ACK reception operation
- Message reception
- ACK transmission operation
- Use of PALs
- Sequence shifts from one PAL to another during operation execution

In addition, this diagram indicates two basic points in LINK operations:

- An ACK receive sequence is part of the message transmit sequence (sequence is not complete until the ACK receive sequence is done)
- An ACK transmit sequence is part of the message receive sequence (sequence is not complete until the ACK transmit sequence is done)

Figure 5-21 LINK Interface Signals

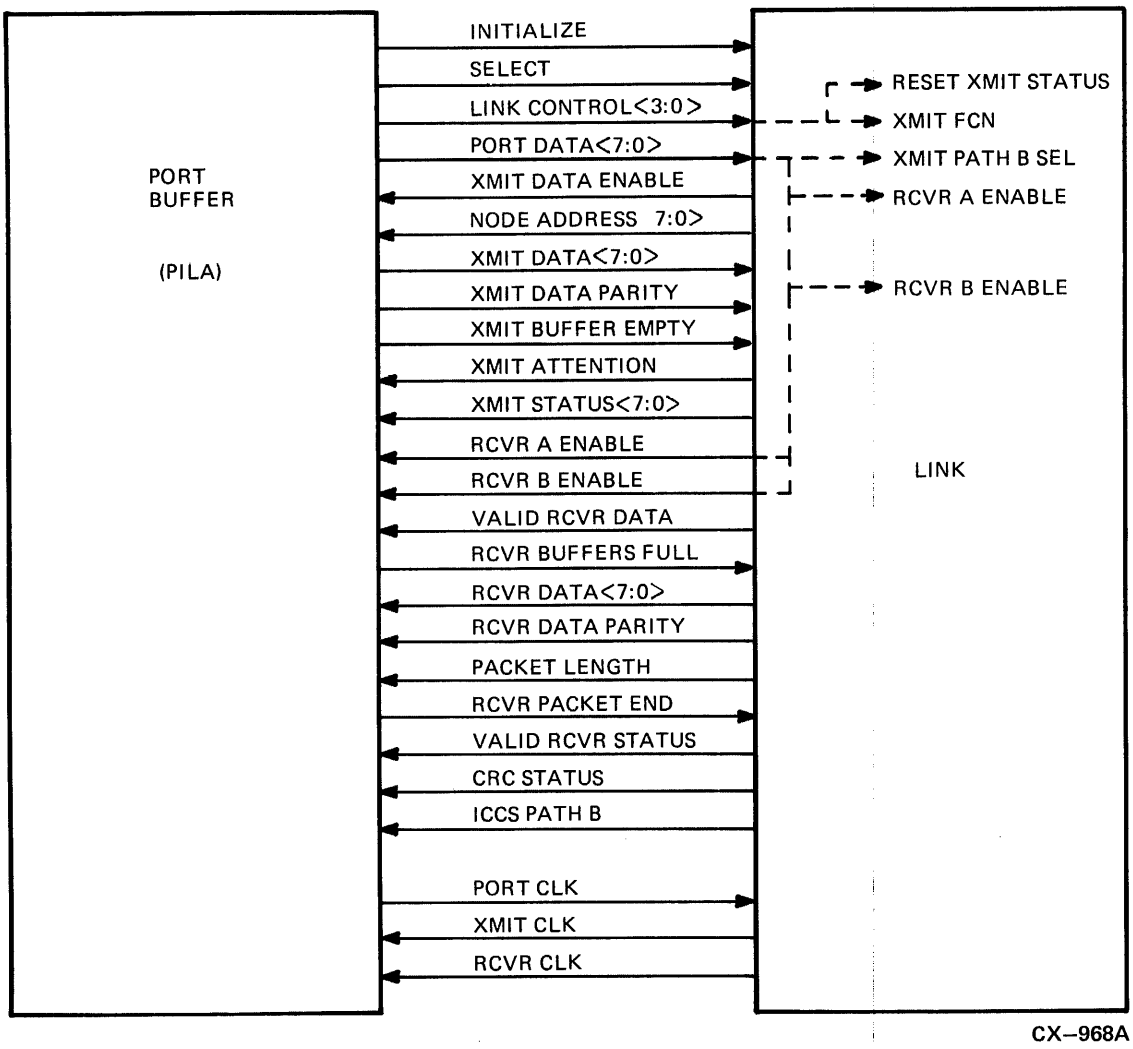
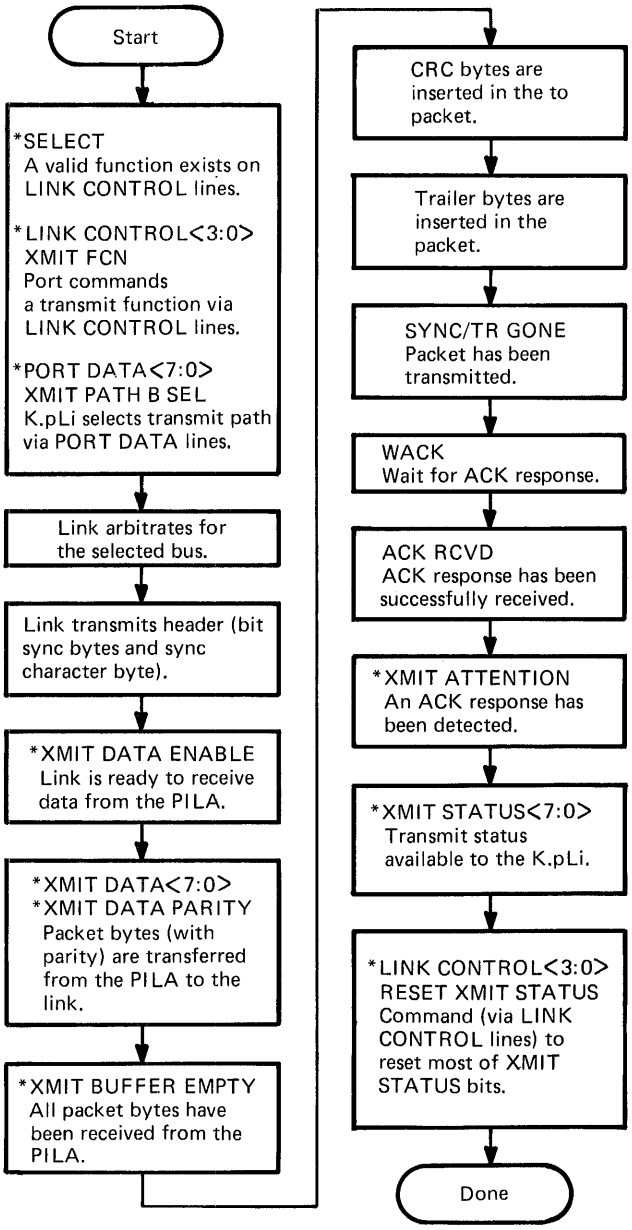


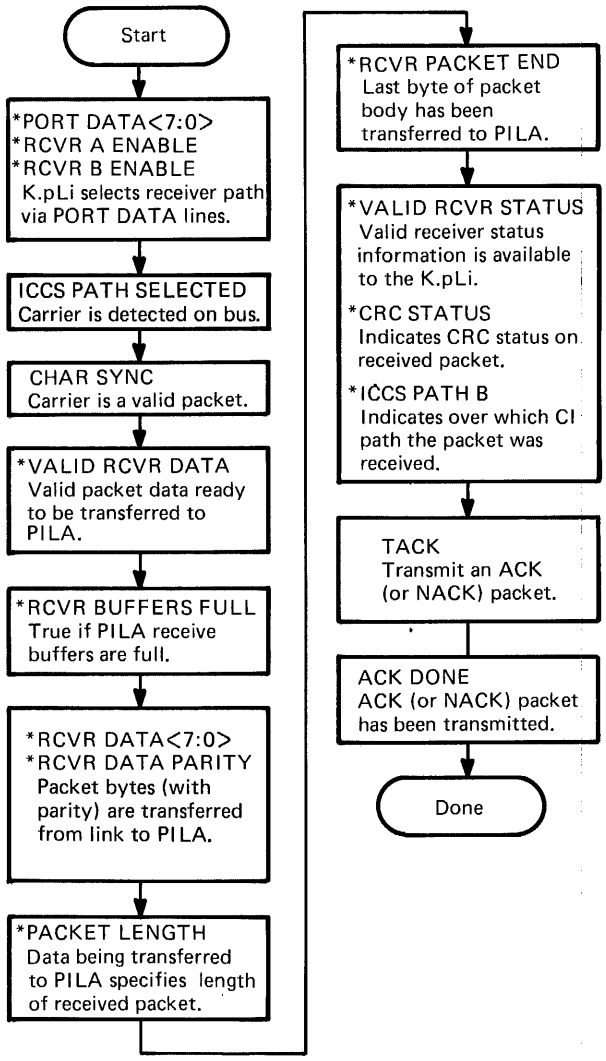
**Figure 5-22 Interface Flow Diagram - Transmit Operation**




*Interface signal

CX–969A

Figure 5-23 Interface Flow Diagram - Receive Operation




*Interface signal

CX–970A

### 5.7.1 Message Transmit

Figure 5-24 illustrates the Message Transmit State Logic and is used in conjunction with the MESSAGE XMIT state diagram in the engineering drawing set. Two PALs are used for the message XMIT state sequence.

The K.pli signal INITIALIZE asserts TINIT, initializing the LINK and asserting MX STATE A (transmit idle state) from PAL number one. When the K.pli commands a transmit function, XMIT FCN is asserted from the LINK Control PAL asserting TXMIT and transferring the LINK to state B. The LINK then arbitrates for the bus in state B. When the arbitration is successful, ARB is asserted and the LINK transfers to state C.

In state C, the LINK transmits the bit synchronization bytes and the sync character byte. After the sync character byte is transmitted, SYNC/TR GONE is asserted and moves the LINK to state D.

In state D the CRC Generator is enabled (except for maintenance loop operations), the second MSG XMIT State PAL is enabled, and the LINK goes to state E. PAL number one remains in state E for the remainder of the transmission if no parity error occurs. In case of parity error, PE is asserted and transfers the LINK to state F.

When the LINK is placed in state F, PAL number two is reset and XMIT ATTENTION asserted to the K.pli which terminates the transmission of the packet. The LINK then returns to state A.

PAL number two moves from its idle state (G) to state H when Pal number one asserts MX STATE D. From state H the LINK goes to state I where the destination byte is clocked into the destination address register. At this point, the LINK transfers to state J where it waits for transmission of the packet body. When the last byte of the body is transmitted, XMIT BUFFER EMPTY is received from the PILA transferring the LINK to state K (if this is not a maintenance operation). If this is a maintenance operation (LOOP true), the LINK goes directly to state L.

CRC bytes are transmitted in state K. MAX CRC 3 is asserted when the last CRC byte is transmitted, causing the LINK to transfer to state L. In this state, the packet trailer bytes are transmitted. Following this transmission, SYNC/TR GONE is asserted and transfers the LINK to state M.

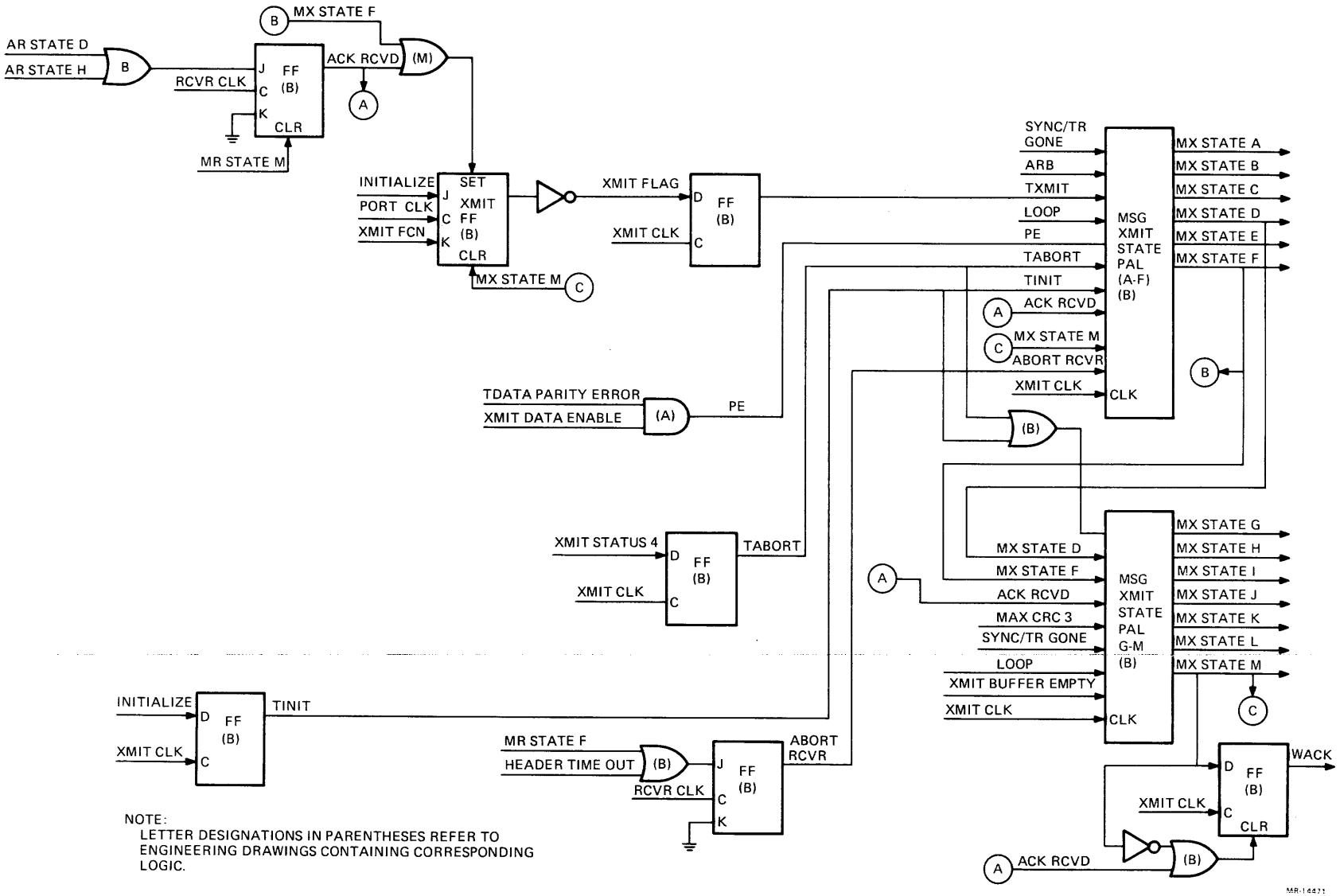
The LINK completes its transmission in state M and waits for completion of the ACK receive sequence. AR STATE D or AR STATE H (from the ACK Receive State Logic) asserted indicates the end of the ACK receive sequence. Either of these signals asserts ACK RCVD to both PALs, returning them to their idle states. ACK RCVD also negates TXMIT completing the message XMIT sequence.

When the LINK enters state M, WACK (Wait for ACK) is asserted to the ACK Receive State Logic enabling the ACK RCVR PAL to start the ACK receive sequence. When the ACK response is received, ACK RCVD is asserted and WACK is negated.

The K.pli can abort the transmission by asserting ABORT XMIT FCN over the LINK control lines. This signal asserts XMIT STATUS 4 and then TABORT with two flip-flops. TABORT resets both XMIT PALs to their idle states.

The MSG XMIT sequence is also reset by HEADER TIME OUT which asserts ABORT RCVR to PAL number one. The MSG RCVR State Logic asserts HEADER TIME OUT when a carrier is detected but SYNC CHAR does not occur.

**Figure 5-24 Message Transmit State Logic**




NOTE:
LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

### 5.7.1.1 Transmit Control Logic

Figure 5-25 illustrates the logic controlling the data flow through the transmit channel shown in Figure 5-11. The state signals generated by the XMIT State PALs regulate these control signals. Assertion and negation of the control signals can be related to the task(s) performed in the various states. (Refer to the XMIT state diagrams in the engineering print set.) ENA SYNC/TR CNT enables the Sync/Trailer Counter to start counting. ENA SYNC/TR gates the bit sync bytes and the trailer bytes onto the XMIT DATA Bus. This signal is negated by ENA XMIT DATA LATCH which gates the packet bytes from the PILA onto the XMIT DATA Bus and asserts ENA XMIT DATA PARITY.

The ENA XMIT DATA REG isolates the BUS TDATA Bus from the XMIT DATA Bus while the CRC bytes are being placed onto the TDATA Bus. TINIT initially asserts ENA XMIT DATA REG which passes the packet bytes onto the BUS TDATA Bus until XMIT BUFFER EMPTY is received from the PILA. XMIT BUFFER EMPTY negates ENA XMIT DATA REG which remains negated until all the CRC bytes are placed onto the BUS TDATA Bus. When this occurs, MX STATE L is asserted thereby reasserting ENA XMIT DATA REG for the trailer bytes. MX STATE L also asserts SEL TRAILER, gating the trailer bytes out of the Sync/Trailer PROM onto the XMIT DATA Bus.

A DRIVER ENA and B DRIVER ENA enable the drivers that output the transmitted packet onto the selected CI path. During a message XMIT operation, the selected DRIVER ENA signal is asserted by the SEL TPATH signal selected by the K.pli via the LINK control lines and by MX STATE C. When SYNC/TR GONE is asserted during the MX STATE L, the DRIVER ENA signal is negated. During an ACK XMIT operation, the selected DRIVER ENA signal is asserted by AX STATE B and LAST RCVR = B. LAST RCVR = B is true if the last message arrived on CI path B (ICCS PATH B true). In this case, B DRIVER ENA is asserted to transmit the ACK over the same path as it was received. Conversely, if the message was received on CI path A, A DRIVER ENA is asserted, transmitting the ACK over CI path A. When SYNC/TR GONE is asserted, the DRIVER ENA signal is negated during AX STATE H.

### 5.7.1.2 Transmit Status

Eight transmit status bits (XMIT STATUS <7:0>) indicate transmit operation status (Figure 5-26). These bits are available to the K.pli along with XMIT ATTENTION.

XMIT ATTENTION is asserted when an ACK or NACK response is received from the destination node. It is also asserted when the destination node does not respond (ACK packet timeout occurs), when a transmit parity error occurs, or when an abort transmission command is issued (ABORT XMIT FCN is asserted).

**Figure 5-25 Transmit Control Logic**

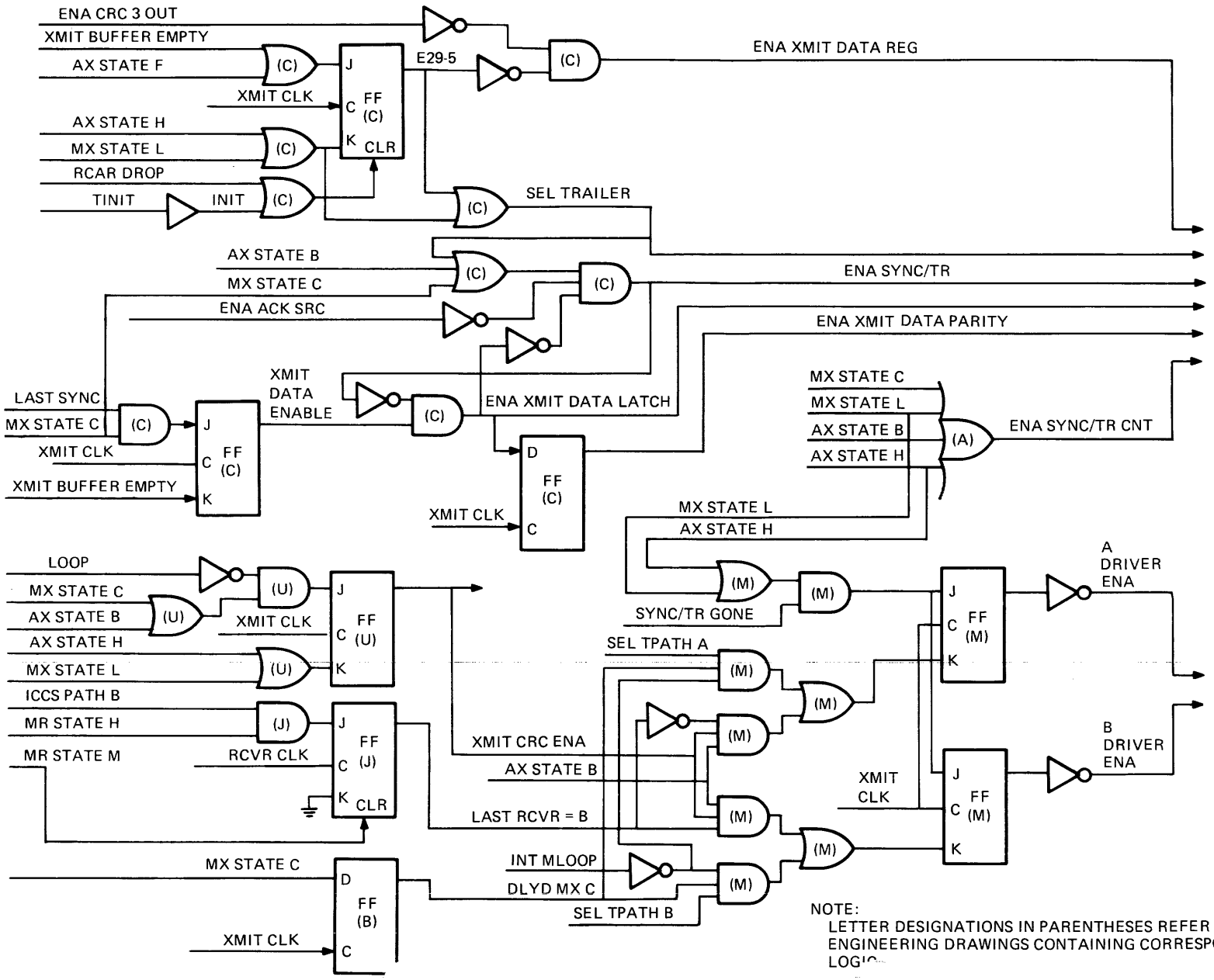
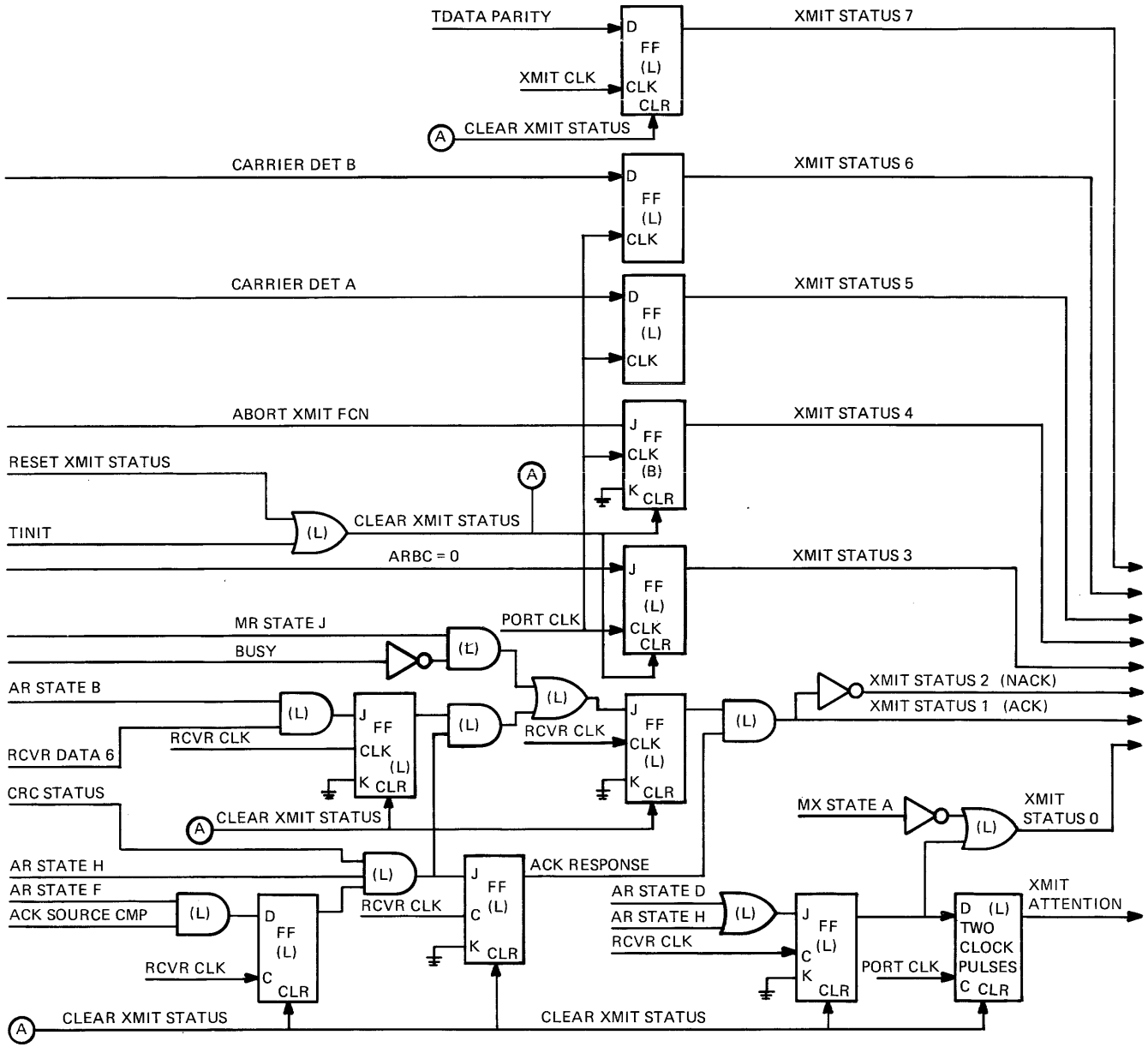


**Figure 5-26 Transmit Status**




NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX–972A

The XMIT STATUS bits are asserted as follows:

| | |
|---|---|
| XMIT STATUS 7 | This bit is set if a data parity error is detected on the BUS TDATA Bus in the transmit channel during a transmission. A parity error asserts XMIT ATTENTION to the K.pli which then aborts the transmission. |
| XMIT STATUS 6 | When set, this bit indicates a carrier present on CI Bus A. |
| XMIT STATUS 5 | When set, this bit indicates a carrier present on CI Bus B. |
| XMIT STATUS 4 | This bit is set when the K.pli commands a transmission abort (ABORT XMIT FCN) via the LINK control lines. |
| XMIT STATUS 3 | This bit is set when an arbitration countdown reaches zero. It does not necessarily mean a transmission will occur (Section 5.4). |
| XMIT STATUS 2 | This bit is set when a NACK is received from the destination node. A NACK response asserts XMIT ATTENTION to the K.pli. |
| XMIT STATUS 1 | This bit is set when an ACK is received from the destination node. An ACK response asserts XMIT ATTENTION to the K.pli. |
| XMIT STATUS 0 | This bit is set when a transmit operation is in progress or whenever XMIT ATTENTION is asserted. |

### 5.7.2 ACK Receive

Figure 5-27 illustrates the ACK Receive State Logic and is used in conjunction with the ACK RCVR state diagram in the engineering drawing set. Two PALs are used for the ACK RCVR state sequence.

#### 5.7.2.1 ACK Receive PAL States

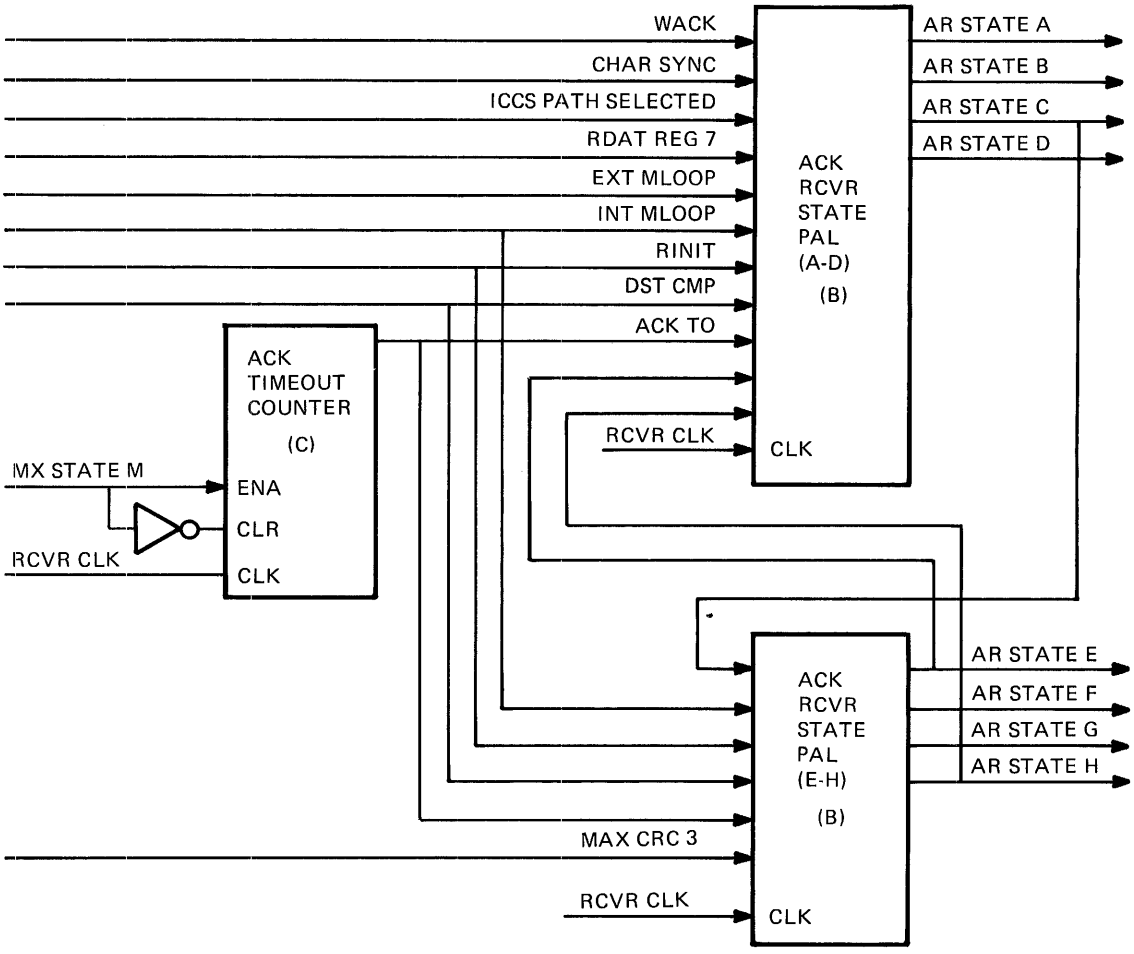
INITIALIZE from the K.pli initializes the receive channel and asserts RINIT in both ACK RCVR PALs placing them in their idle states (state A for PAL number one; state E for PAL number two). The LINK is transferred to AR STATE B when the following three conditions exist:

- PAL number one senses a valid packet is being received (CHAR SYNC true).
- The receiver is waiting for an ACK response (WACK true).
- The package is an ACK (RDAT REG 7 = 1) rather than a message packet.

The packet true destination byte is checked in state B. If a match is obtained (DST CMP true), the LINK transfers to state C. PAL number one remains in state C until the ACK RECVR state sequence is complete. In state E, ACK RCVR State PAL number two is enabled (AR STATE E is asserted) and the complement destination byte is checked. If a destination match is obtained (DST CMP true), PAL number two moves to state F.

State D of PAL number one is a receiver clear state, entered if an improper response is obtained in states A, B, or C. State D is entered from state A if CHAR SYNC and WACK are true but RDAT REG 7 = 0 (a message packet, not an ACK response). State D is entered from state B if a true destination mismatch occurred; from state C if a complementary destination mismatch occurred. After clearing the various receiver functions, PAL number one returns to the idle state A. In state F, the packet source byte is checked. If this is not a maintenance operation (INT MLOOP false), the LINK then passes to state G. If INT MLOOP is true (a maintenance operation), the LINK goes to state H.

Figure 5-27 ACK Receive State Logic




NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX–973A

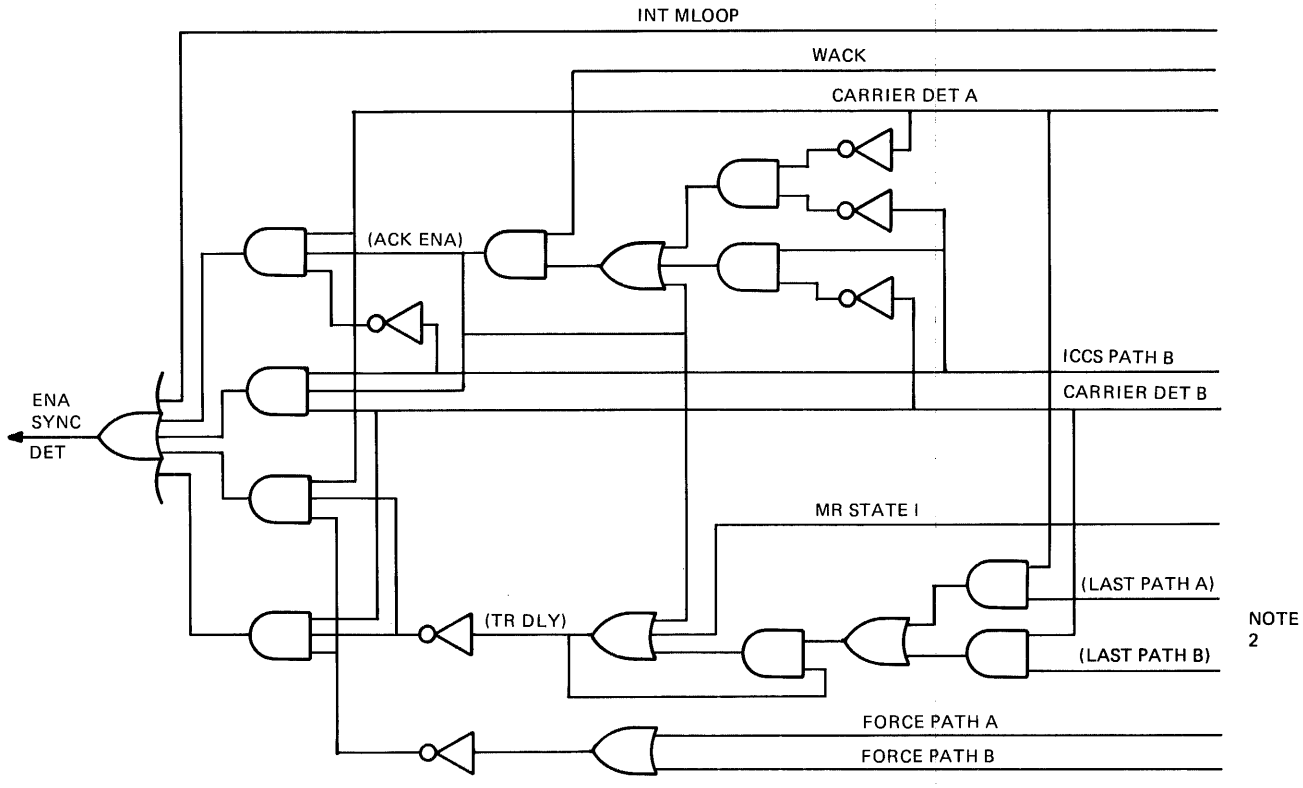
In state G the CRC bytes are input to the CRC Checker for CRC error checking. When MAX CRC 3 is asserted (last CRC byte into the CRC Checker), the LINK moves to state H. This is the last state in the ACK RCVR sequence. In this state, receive functions are cleared and then both PALs return to the idle state. The last state in a MESSAGE XMIT state sequence is state M. When MX STATE M is asserted, an ACK timeout counter is enabled and starts counting. After 3.66 microseconds, if the ACK RCVR sequence is not completed, the counter asserts ACK TO, terminating the sequence and returning both PALs to the idle state. The K.pli then reads status bits to determine the problem.

### 5.7.2.2 Sync Character Detect Enable PAL

The Sync Character Detect Enable PAL (Figure 5-28) enables the Sync Character Detector when a packet is expected and inhibits the detector when transmitting from this node. In the figure, ENA SYNC DET is asserted by any of five signals applied to an output OR gate. The Sync Character Detector should be enabled during the following times:

- During an internal maintenance loop operation
- After a transmission when an ACK packet is expected
- During message packet transmission from another node (taking care not to respond to transmissions from this node - ACK packet)

**Figure 5-28 Sync Character Detect Enable PAL**




NOTE: 1. THE LOGIC IN THIS FIGURE IS CONTAINED ON SHEET J OF THE ENGINEERING DRAWINGS.
2. SIGNALS GENERATED INTERNALLY

CX–974A

When in maintenance loop operation, INT MLOOP is true and enables the Sync Character Detector.

Two signals enable the Sync Detector when an ACK packet is received. One is generated by ANDing CARRIER DET A with the negated state of ICCS PATH B while the other is generated by ANDing CARRIER DET B with the asserted state of ICCS PATH B. Thus, the two gates search for a carrier presence in both CI paths. Enabling the two gates is restricted to ACK packets by ACK ENA that is asserted while waiting for an ACK packet (WACK true) and after a loss of carrier is sensed. (The carrier lost would be a message transmit carrier from this node.) ICCS PATH B (true or false) enables one of the AND gates in the ACK ENA logic. When that gate senses a loss of carrier (CARRIER DET is negated), ACK ENA is asserted and is latched. The next time a carrier is sensed (ACK response), the output AND gate is enabled and asserts ENA SYNC DET via the output OR gate.

These last two signals enable the sync detector when a message packet is received. They are generated by AND gates enabled when the node is not transmitting a message packet (both FORCE PATH signals false) and a carrier is detected on one of the CI paths. The gates are inhibited by trailer delay (TR DLY) true at ACK transmission end when the packet trailer is under transmission.

### 5.7.3 Message Receive

Figure 5-29 illustrates the Message Receive State Logic. It is used in conjunction with the MSG RCVR state diagram in the engineering drawing set. Two PALs are used for the message receiver state sequence.

The signal INITIALIZE from the K.pli asserts ABORT + INIT FCN which asserts RINIT. RINIT initializes the Receive Channel Logic and places the two MSG RCVR state PALs into their idle states (state A for PAL number one and state M for PAL number two). When the receiver is not disabled due to transmission from the transmit channel (RXMIT false), a valid packet is in the receive channel (CHAR SYNC true) and the packet is recognized as a message (RDAT REG 7 = 0) and not an ACK. PAL number one transfers to MR STATE B.

In MR STATE B, VALID RCVR DATA is asserted to the PILA indicating a valid packet is being received. PACKET LENGTH is asserted to the PILA indicating the byte being transferred contains packet length information. Also, the CRC Checker is enabled and starts receiving the packet bytes. The LINK moves to MR STATE C on the next clock pulse.

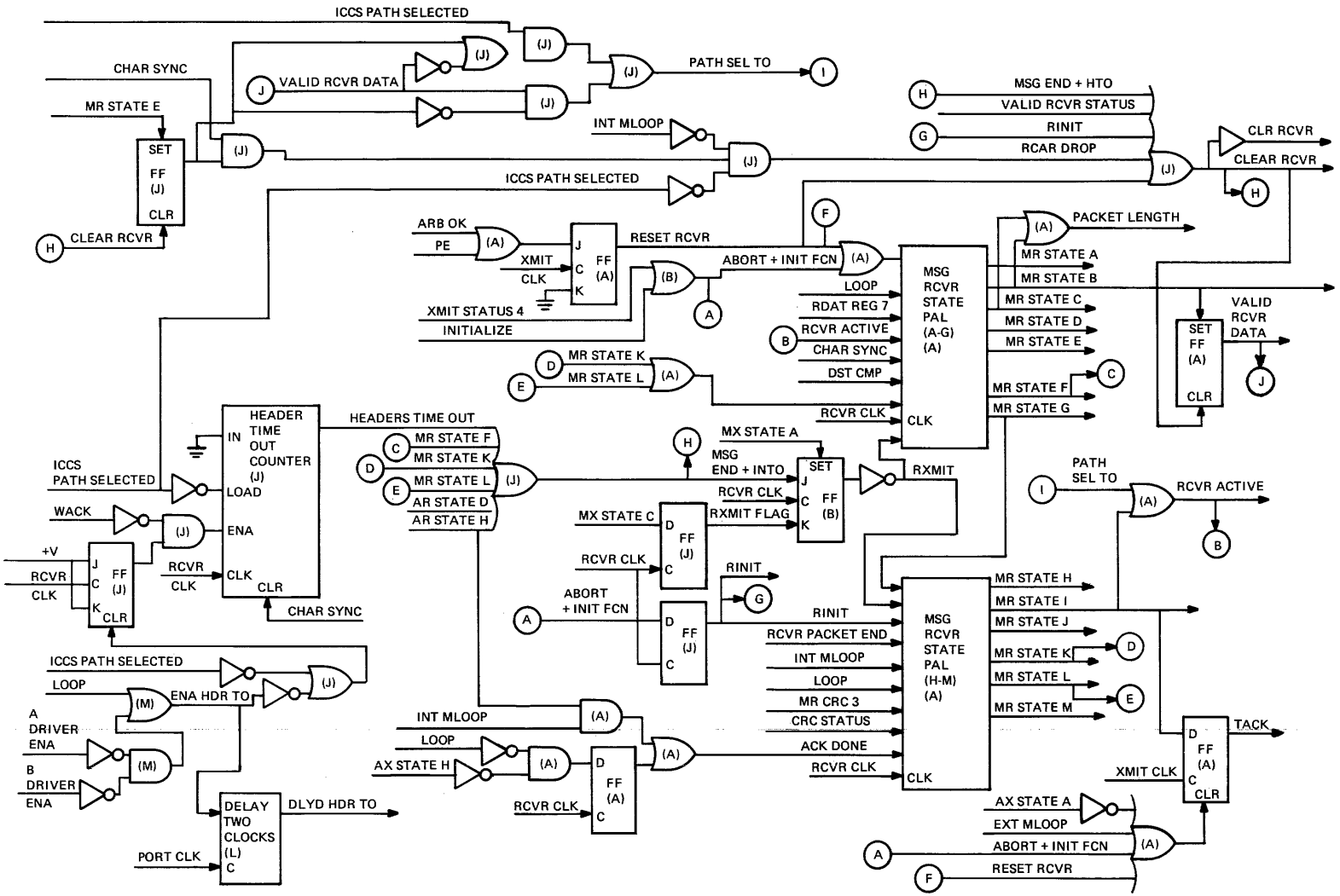
The true destination byte is checked in MR STATE C and if a match is obtained (DST CMP true), the LINK moves to state D. PACKET LENGTH remains asserted in state C as the byte being transferred to the PILA contains packet length information.

In MR STATE D the complement destination byte is checked and if a match is obtained (DST CMP true), the LINK moves to MR STATE E. In this state the packet source byte is clocked into the True and Complement ACK Destination Registers to serve as the ACK response destination. The next RCVR CLK pulse moves the LINK to MR STATE G. For the remainder of the MSG RCVR state sequence, PAL number one remains in MR STATE G. Assertion of MR STATE G moves PAL number two from its idle state (state M) to state H when its condition signal (RCVR PACKET END) is asserted.

If any of the following three condition tests made by PAL number one fails, the LINK is transferred to MR STATE F:

- While in state A with RXMIT false, CHAR SYNC is asserted but RDAT REG = 1 (this is an ACK packet).
- A true destination mismatch occurred in state C.
- A complement destination mismatch occurred in state D.

## Figure 5-29 Message Receive State Logic




NOTE: LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

CX-975A

Receive logic is cleared in MR STATE F and PAL number one returns to the idle state (state A) on the next RCVR CLK pulse. PAL number two remains in idle state (state M) while the packet body is transferred to the PILA. After the last body byte is sent to it, the PILA asserts RCVR PACKET END and PAL number two goes to MR STATE H. Packet CRC bytes are input to the CRC checker in state H. When the last byte is in the checker, MR CRC 3 is asserted. With no CRC error, CRC STATUS is true when MR CRC 3 is asserted and the LINK moves to state I. In case of CRC error, CRC OK is false and the LINK goes to state L. In state L the MSG RCVR state sequence is aborted. The receive channel is cleared and both PALs move to their idle states (PAL number one to state A and PAL number two to state M).

The message receive state sequence remains in state I while the LINK transmits the ACK response. MR STATE I is asserted and in turn asserts TACK (transmit ACK) to the ACK transmit state PAL, initiating the ACK transmit sequence. When ACK transmission is complete, AX STATE H is negated asserting ACK DONE to message receiver PAL number two. The assertion of ACK DONE moves the LINK to MR STATE K. In this state, the receive channel is cleared and PAL number one is returned to its idle state (state A). The next RCVR CLK pulse returns PAL number two to its idle state (state M).

The Message Receive State Logic contains a header timeout counter to prevent receive channel hangups. This counter is turned on by ICCS PATH SELECTED (removes the counter LOAD signal) and cleared by CHAR SYNC. It starts counting when a carrier is detected and is cleared when the carrier is recognized as a valid packet. If CHAR SYNC fails to assert, the counter times out in 3.66 microseconds and outputs HEADER TIME OUT. Assertion of that signal causes MSG END + HTO to assert, thereby asserting CLEAR RCVR to reset the receive logic.

A flip-flop enables and disables the header timeout counter at the RCVR CLK rate. The four-bit counter is extended to five bits, producing the 3.66 microsecond timeout period (32 X 114.28 ns = 3.66 microseconds). This counter is disabled by WACK. When the transmit channel is transmitting a message packet, WACK is asserted in MX STATE M. Thus, WACK prevents carrier detection from starting the header timeout period. Signals other than MSG END + HTO assert CLEAR RCVR. One such signal, RCAR DROP (receive carrier dropped), is asserted if a carrier is lost during a message reception. ICCS PATH SELECTED is asserted before and negated after CHAR SYNC. If a receive carrier is prematurely lost, ICCS PATH SELECTED is negated while CHAR SYNC is still true, causing assertion of RCAR DROP. Note CHAR SYNC is not applied to the ANDing operation until MR STATE E sets a flip-flop gating CHAR SYNC to the RCAR DROP signal AND gate. Delaying CHAR SYNC until MR STATE E allows the packet header to pass before the node looks for carrier dropout.

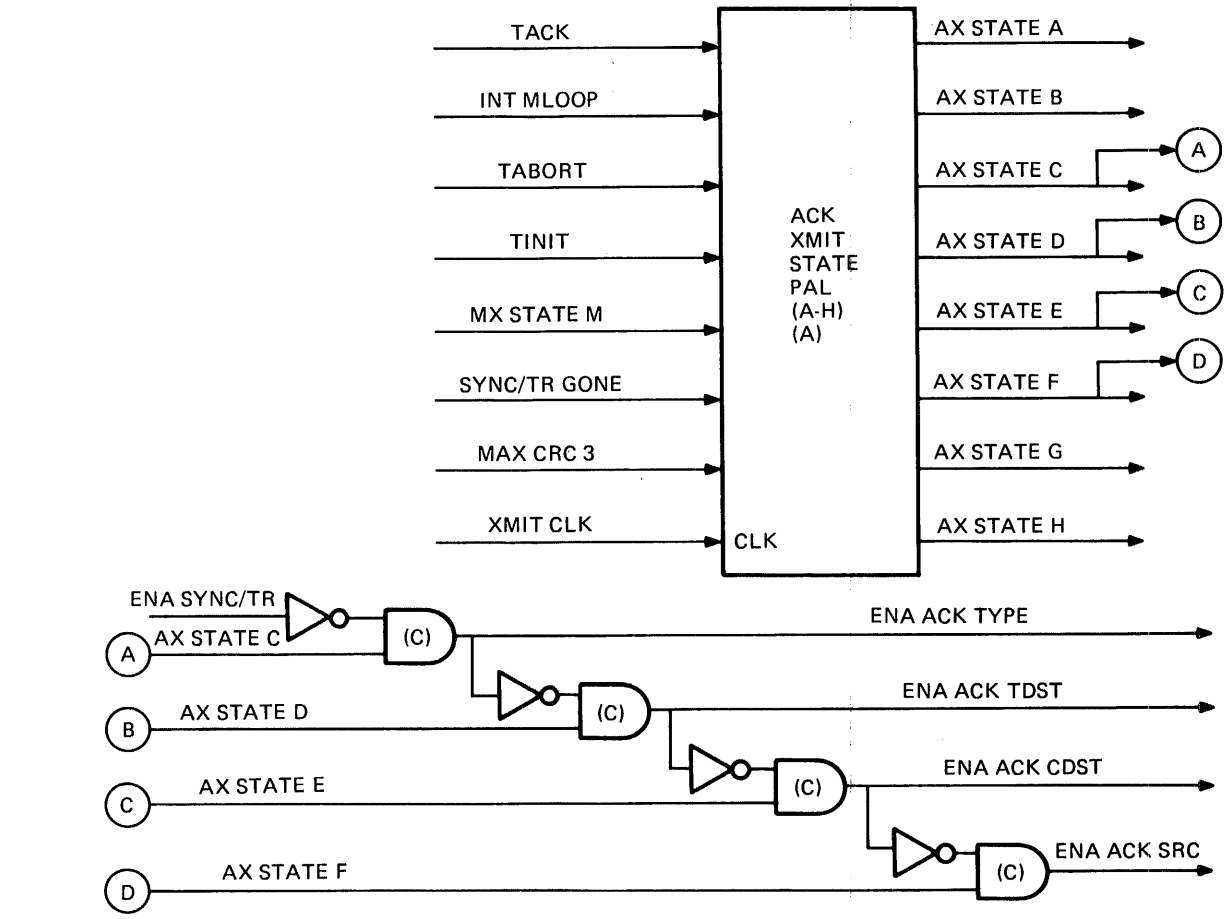
### 5.7.4 ACK Transmit

Figure 5-30 illustrates ACK Transmit State Logic and is used in conjunction with the ACK XMIT state diagram in the engineering drawing set.

The signal INITIALIZE from the K.pli asserts TINIT initializing the LINK and asserting AX STATE A from the ACK XMIT PAL. AX STATE A is the ACK transmit idle state. When TACK (transmit ACK) arrives from the MSG RCVR State PAL, the LINK goes to AX STATE B. In this state the Sync/Trailer PROM Logic is enabled and outputs the bit synchronization bytes and the sync character byte onto the XMIT DATA Bus. The selected transmit driver is also enabled. When SYNC/TR GONE is asserted, the LINK transfers to state C.

The LINK is in AX STATE C for one XMIT CLK pulse. While in this state, the ACK type byte is placed onto the XMIT DATA Bus and the CRC Generator is enabled. The next clock pulse moves the LINK to state D. In AX STATE D the ACK true destination byte is placed onto the XMIT DATA Bus. The LINK then advances to AX STATE E where the ACK complement destination byte is placed on the XMIT DATA Bus. The LINK then moves to state F. The ACK source byte is placed on the XMIT DATA Bus in AX STATE F. The LINK moves to state G where CRC bytes (generated by the CRC Generator) are output onto the BUS TDATA Bus. When the last CRC byte is placed onto the bus, MAX CRC 3 is asserted and moves the LINK to state H. In that state the Sync/Trailer PROM is enabled again and

Figure 5-30 ACK Transmit State Logic




NOTE:
LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

MR-14473

the packet trailer bytes are output from the PROM onto the XMIT DATA Bus. After the trailer bytes are placed on the bus, SYNC/TR GONE is asserted and returns ACK XMIT PAL to its idle state (state A).

Note in Figure 5-30 that assertion of each gate coupling a byte to the XMIT DATA Bus depends on negation of the gate that coupled the preceding byte to the bus. This ensures only one source drives the XMIT DATA Bus at one time.

## 5.8 STATUS LEDS

The LINK has two status LEDs: a green LED and a red LED.

The green LED is CI Active. The LED lights when the K.pli selects a CI Bus (A or B) path.

The red LED is Module Failure. When the K.pli puts the LINK in maintenance mode, the LED lights. In maintenance mode, the output of the transmitter is wrapped around into the receiver (shown on Figure 5-1 as ME DATA). The maintenance mode allows the K.pli to transmit data and check that data verifying the LINK logic.

# CHAPTER 6
# PORT BUFFER MODULE (PILA)

## 6.1 INTRODUCTION

The Port Buffer Module is one of three modules that make up the Host Interface. The Port Buffer Module, or Port Interface-to-Link Adapter (PILA), contains four 1 Kbyte (8-bit bytes) RAM buffers, two 1 Kbyte buffers for reception, and two 1 Kbyte buffers for transmission. The buffers provide temporary storage between the 70 megabit-per-second CI Bus and the Port Processor Module (K.pli).

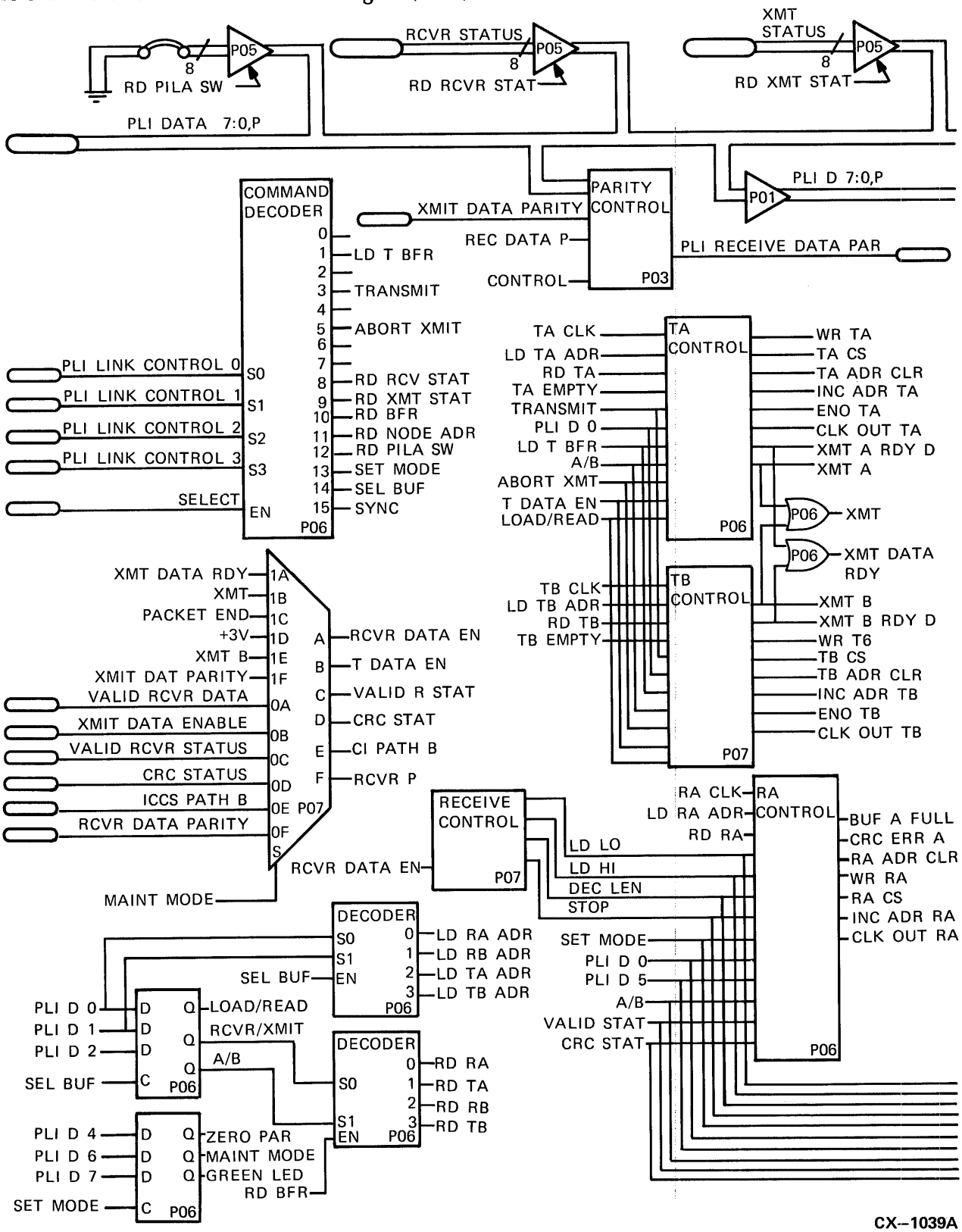
As the Link Interface Module (LINK) receives a packet, it loads the PILA receive buffers. The K.pli reads the receive buffers, reformats the data, and sends it to the HSC memory via the Control Bus or Data Bus.

The K.pli loads packet data into the transmit buffers. After the entire packet has been loaded, the K.pli commands the LINK to transmit the packet. The LINK then reads the transmit buffer as it transmits the packet.

The Port Link Interface (PLI) carries communications and data transfers between the K.pli and the PILA. All data and PLI functions are controlled by the K.pli.

The Interprocessor Link Interface (ILI) carries communications and data transfers between the PILA and the LINK. All data and ILI functions are controlled by the LINK. Figure 6-1 shows the block diagram of the PILA. The following paragraphs describe the interfaces and the operation of the PILA.

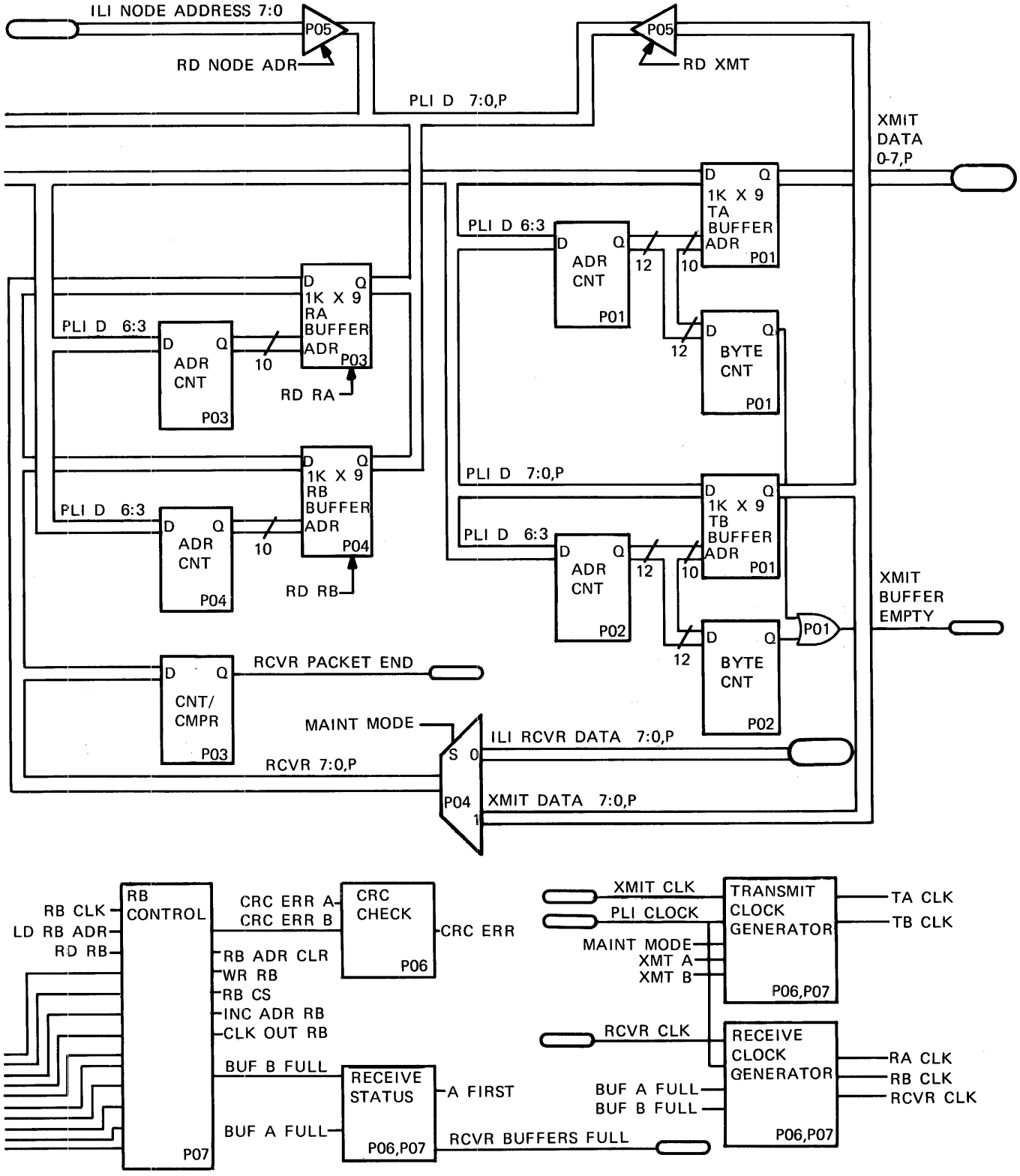
Figure 6-1 Port Buffer Module Block Diagram(PILA)




CX-1039A
Sheet 1 of 2

(Continued on next page)

Figure 6-1 (Cont.) Port Buffer Module Block Diagram(PILA)




CX–1039A
Sheet 2 of 2

## 6.2 PORT LINK INTERFACE (PLI)

The K.pli controls the PILA and LINK via the Port Link Interface (PLI). Also, the K.pli and PILA pass data over the PLI. Figure 6-2 shows the PLI signals, and Table 6-1 describes the PLI signals.

**Figure 6-2 PLI Signals**

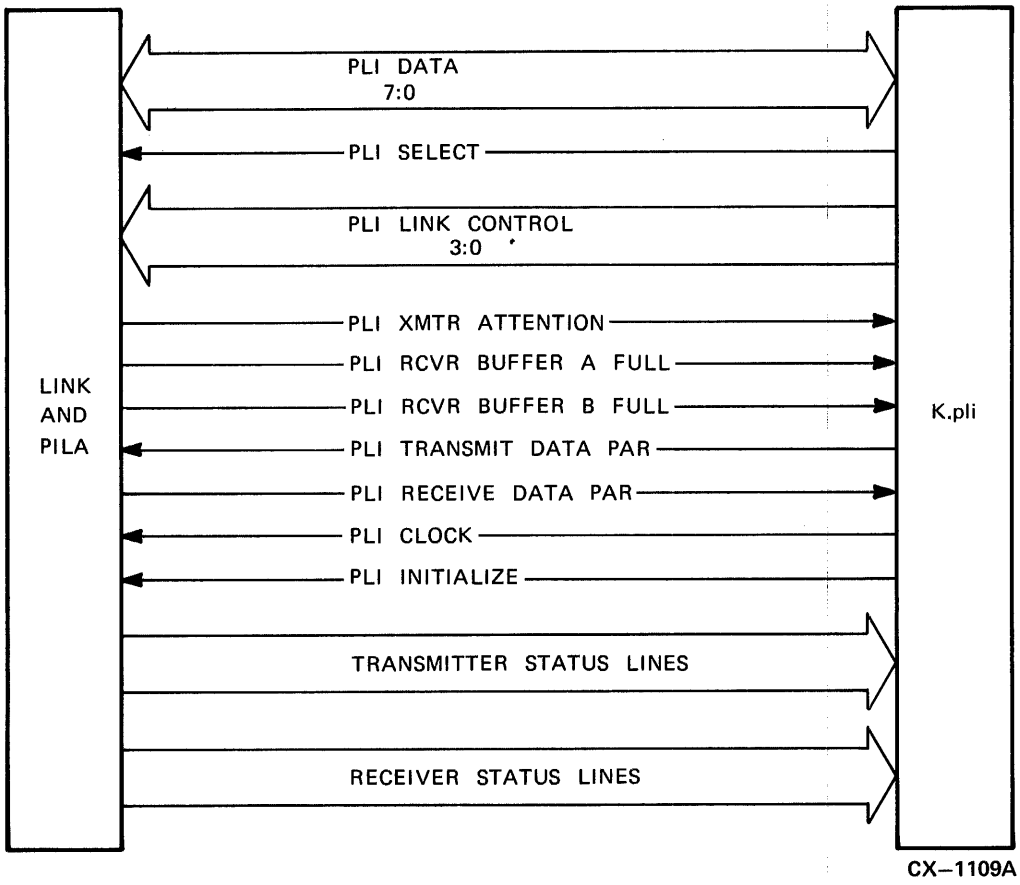


Table 6-1 PLI Signal Description

| Signal | Description |
|---|---|
| PLI DATA <7:0> | Transfers data from PILA to K.pli PLI Input Buffer and data from K.pli PLI Output Buffer to PILA. Also transfers control and status between the K.pli and LINK/PILA. The direction and type of information transferred is determined by the code on PLI LINK CONTROL <3:0> lines. |
| PLI SELECT | Asserted by the select PLI command. Acts as an enable for the PLI LINK CONTROL <3:0> lines. |
| PLI LINK CONTROL <3:0> | Asserted by a combination of D OUT <03:00> to the PLI Control Register on the K.pli. The PLI SELECT signal enables these lines. Section 6.4 describes the commands and functions encoded in these lines. |
| PLI XMTR ATTENTION | Asserts PLI XMTR ATTENTION in the K.pli. The LINK asserts PLI XMTR ATTENTION when there is a response to a transmit command in normal or loop modes. The K.pli can then read the transmitter status bits with the read transmitter status command (Section 6.4.8).<br><br>PLI XMTR ATTENTION is cleared by INITIALIZE and the reset transmit status command.<br><br>When set, PLI XMTR ATTENTION enables transmit status bits 7, 4, 2, and 1. When reset, PLI XMTR ATTENTION disables transmit status bits 7, 4, 2, and 1. |
| PLI RCVR BUFFER A FULL | Asserts PLI RCVR BUFFER A FULL in the K.pli. Asserted when a valid packet has been stored in Receive Buffer A. Asserted by INITIALIZE to prevent the LINK from putting packets in the buffer until it is enabled. Negated when the buffer is enabled by a set mode command from the K.pli. |
| PLI RCVR BUFFER B FULL | Asserts PLI RCVR BUFFER B FULL in the K.pli. Asserted when a valid packet has been stored in Receive Buffer B. Asserted by INITIALIZE to prevent the LINK from putting packets in the buffer until it is enabled. Negated when the buffer is enabled by a set mode command from the K.pli. |
| PLI TRANSMIT DATA PAR | The K.pli Data Parity Generation Logic generates the PLI TRANSMIT DATA PAR signal when it sends data to the PILA. The PILA stores the parity bit (odd). |
| PLI RECEIVE DATA PAR | The PLI RECEIVE DATA PAR bit (odd) is generated by the LINK when the receive buffer is written. The PILA does not check receive data parity; it is checked by the K.pli Data Parity Check Logic. The PILA generates parity on all read operations except the read buffer command (when it passes on the LINK generated parity). |

Table 6-1 (Cont.) PLI Signal Description

| Signal | Description |
|---|---|
| PLI CLOCK | The K.pli Control Logic generates the PLI CLOCK signal at the K.pli CLK3 time on every microcycle. All PILA flags and status signals are synchronized to this clock. Signals are asserted and strobed by the clock's trailing edge. |
| PLI INITIALIZE | Forces the initialization of the LINK and PILA. The PILA provides a maintenance mode that allows transmit data to be looped back into the receive buffer for diagnostic purposes. INITIALIZE puts the PILA into maintenance mode, setting both the Receive Buffer A Full Flag (BUFA FULL) and Receive Buffer B Full Flag (BUFB FULL) to prevent the LINK from loading the buffers until the K.pli enables the receive buffers. INITIALIZE also resets the status of CRC error (CRC ERR), first buffer (FIRST BUF), Buffer A Bus (BUSB BUFA), and Buffer B Bus (BUSB BUFB), and disables any read or write operations. |

## 6.3 OPERATIONS BETWEEN PILA AND LINK

The LINK loads the received packet into the receive buffers on the PILA and sends packets in the transmit buffers over the CI Bus. The PILA provides dual, independent 1 Kbyte receive buffers (RA and RB) and dual, independent 1 Kbyte transmit buffers (TA and TB) as temporary storage for packets. The following sections describe how the LINK loads a receive buffer during a receive operation and how the LINK gets the packet from a transmit buffer during a transmit operation.

### 6.3.1 Receive Operation from LINK to PILA

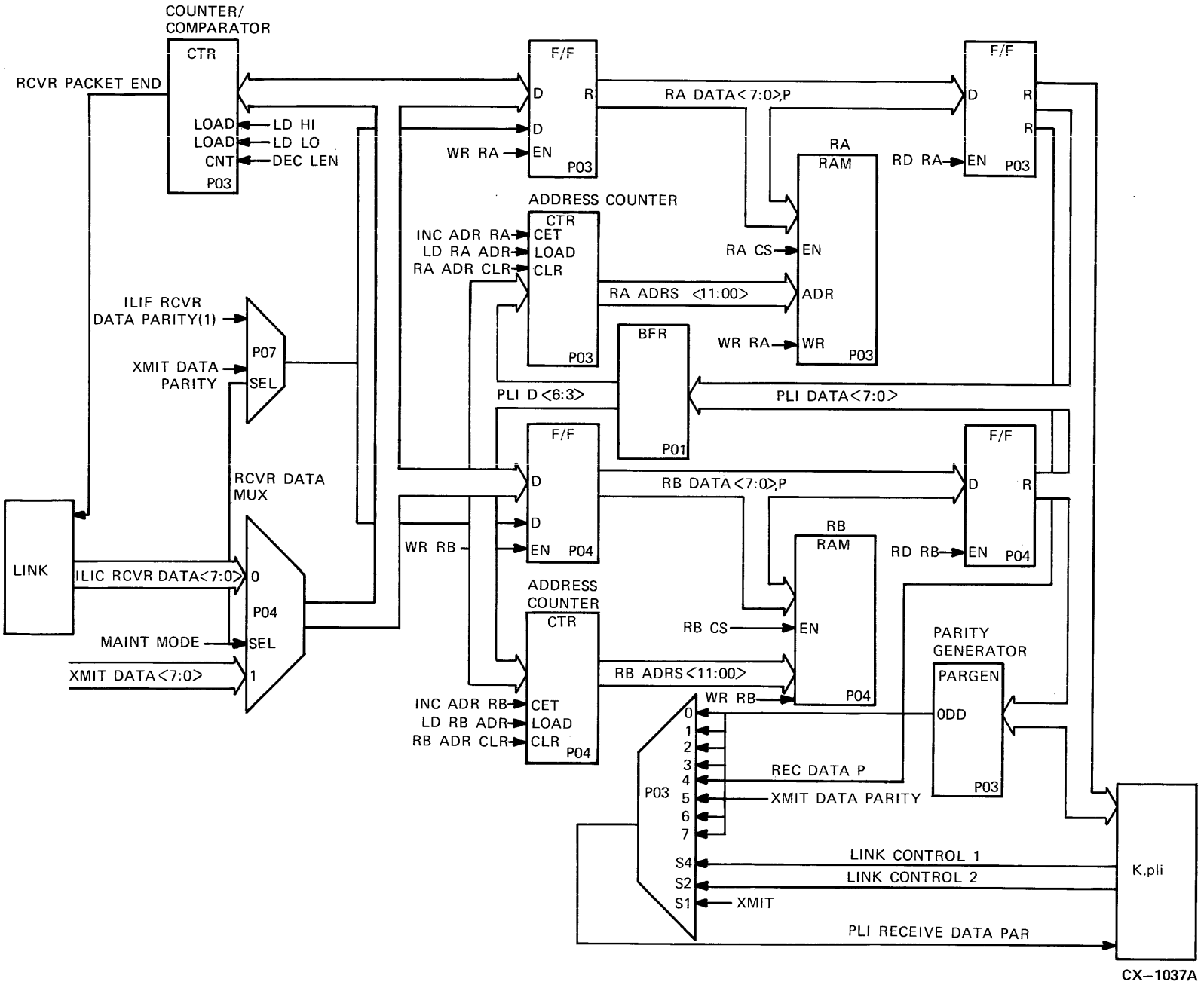
During a receive operation, the LINK puts the bytes of the incoming packet into one of the receive buffers. Figure 6-3 illustrates the receive buffer circuits and the following list describes the circuits:

- Counter/Comparator—loaded with the packet length from the incoming packet and then down counted as the logic puts each byte of the packet into the buffer. (Refer to Chapter 4 for the packet format.)
- Address Counter—initially zero. Incremented each time the logic puts a byte of the packet into the buffer.
- Buffer—RA and RB are the 1 Kbyte receive buffers for storage of the incoming packet bytes.

The following list describes a receive operation (a packet from the LINK going to Receive Buffer A):

1. The LINK recognizes that the incoming data is valid when it receives the CHAR SYNC.
2. The LINK assembles the first byte (PACKET TYPE/LENGTH (HIGH)) and places the byte on the ILIC RCVR DATA lines <7:0> to the Receiver Data Multiplexer.
3. The output of the multiplexer is an input to the Counter/Comparator and the input buffer of Receive Buffer A (RA).
4. The LINK then generates a clock pulse that asserts the following signals on the PILA:
   - RA ADR CLR clears the Address Counter for the buffer
   - LD HI strobes the packet length high byte into the Counter/Comparator

**Figure 6-3 Receive Buffers**



- WR RA strobes data from the multiplexer into location zero of the Receiver Buffer A
- INC RA ADR increments the address counter to location one

5. The LINK assembles the next byte, PACKET LENGTH (LOW), and generates the clock pulse that strobes the byte into the Counter/Comparator and into the buffer and then increments the Address Counter.
6. As the LINK strobes each received byte into the receive buffer and increments the Address Counter, the signal DEC LEN decrements the Counter/Comparator.
7. When the Counter decrements to two, the Comparator asserts the signal RCVR PACKET END.
8. RCVR PACKET END signals the LINK to check the packet CRC.
9. The LINK asserts RCVR BUFFER A FULL to the K.pli.

To learn how the K.pli removes the data from the buffer, refer to Section 6.4.9 and Chapter 7.

### 6.3.2 Transmit Operation from PILA to LINK

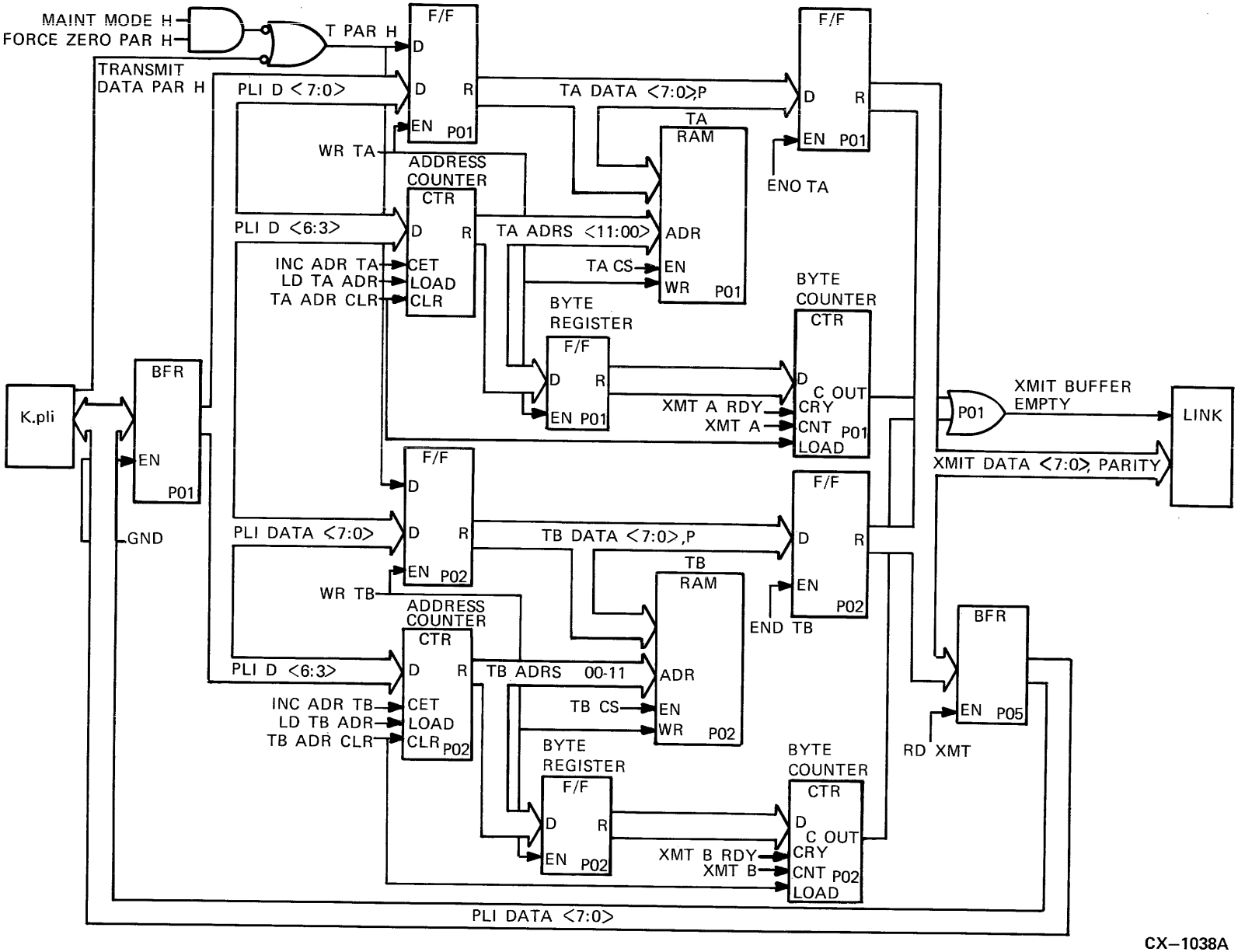
During a transmit operation, the LINK takes the bytes of a packet from one of the transmit buffers and sends the packet over the CI Bus. The K.pli puts the packet into a buffer before it commands the LINK to send the packet. (Filling the buffers is described in Section 6.4.1 and Chapter 7.) The LINK then gets the packet one byte at a time, serializes the bytes, and sends the packet over the CI Bus. Figure 6-4 illustrates the transmit buffers on the PILA, and the following list describes the circuits:

- Address Counter—initializes to zero and increments each time the LINK takes a byte from the buffer
- Byte Register—contains the byte count of the buffer
- Byte Counter—initializes to the count of the Byte Register and decrements each time the LINK takes a byte from the buffer
- Buffer—TA and TB are the 1-Kbyte transmit buffers for storage of packets

The following list describes a transmit operation (the LINK takes bytes of the packet from Transmit Buffer A on the PILA):

- Before starting the transmit operation, the K.pli loads the packet into Transmit Buffer A, and the Byte Register contains the number of bytes in the buffer
- When the K.pli commands the LINK to transmit, the transmit command asserts the following signals:
  - RA ADR CLR—clears the Address Counter and strobes the Byte Register into the Byte Counter
  - TA CS—enables the output of the buffer
  - ENO TA—enables to output of the buffer to the LINK
- The LINK strobes the byte on the XMIT DATA lines into its transmitter logic
- The LINK then asserts a signal to increment the Address Counter and decrement the Byte Counter
- When the Byte Counter reaches zero, the signal XMIT BUFFER EMPTY causes the LINK to append the CRC bytes

**Figure 6-4 Transmit Buffers**




CX–1038A

## 6.4 PLI LINK CONTROL

The K.pli controls the LINK and PILA via the LINK CONTROL <3:0> lines on the PLI. The following sections describe the commands encoded on these lines. Codes of LINK CONTROL <3:0> = 0000 and 0010 are not used. Figure 6-5 shows the command decoder on the PILA.

### 6.4.1 Load Transmit Buffer: LINK CONTROL <3:0> = 0001

Before issuing the load transmit buffer command to the PILA, the K.pli first selects the buffer with the select buffer command (code 1110). The load transmit buffer command then loads the data from the DATA <7:0> lines (and the associated parity bit for the DATA lines) into the selected buffer (Figure 6-4). As long as both the load transmit buffer command and SELECT are asserted, the K.pli loads bytes into the PILA with each PLI clock. The Address Counter is incremented at the end of each cycle during which the load command is asserted. The signal WR TA strobes the data into the buffer and also strobes the Address Counter into the Byte Register (which is the byte count).

The K.pli starts loading the transmit buffers at location 0 and goes to location n-1 (Figure 6-6). If XMIT OFFSET is set, then the transmit buffers are loaded from location 0 + offset to location n + offset-1, where n = the number of bytes loaded into the buffer by the K.pli.

Loading the transmitter buffers is described in Chapter 7.

### 6.4.2 Transmit: LINK CONTROL <3:0> = 0011

The transmit command performs the following operations:

- Selects the transmit buffer to be transmitted
- Clears its address counter to zero
- Initiates the LINK CI arbitration on the CI path selected by the command

Upon successful CI Bus arbitration, the selected buffer is transmitted. The LINK checks data parity. If a parity error occurs, the transmission is aborted. The path and buffer are selected by DATA lines 7 and 0 as shown in Table 6-2.

**Table 6-2 Path and Buffer Selection by Data Lines 7 and 0**

| Data Line 7 | Data Line 0 | CI Path | Transmit Buffer |
|---|---|---|---|
| 0 | 0 | A | A |
| 0 | 1 | A | B |
| 1 | 0 | B | A |
| 1 | 1 | B | B |

Figure 6-5 Link Control Command Decoder

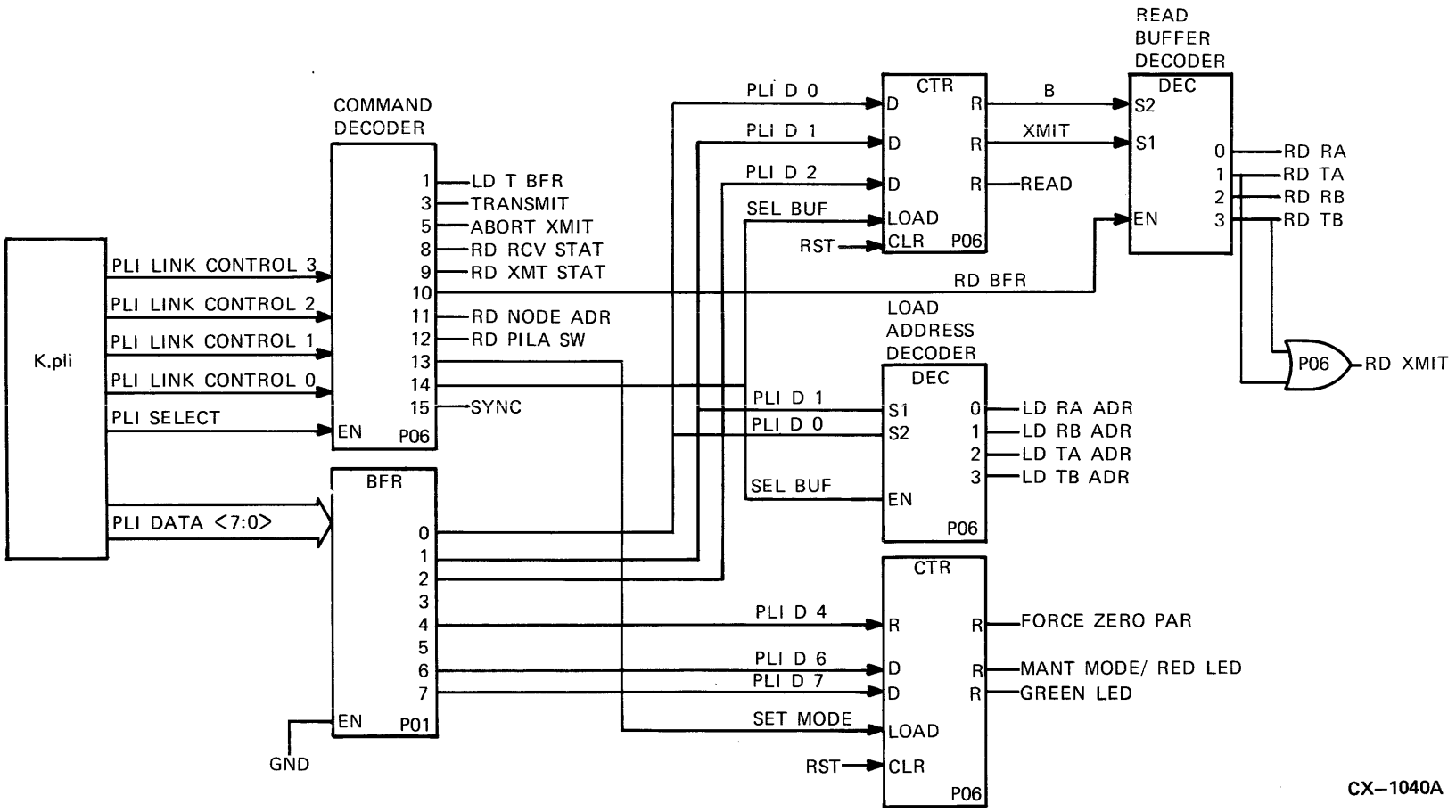
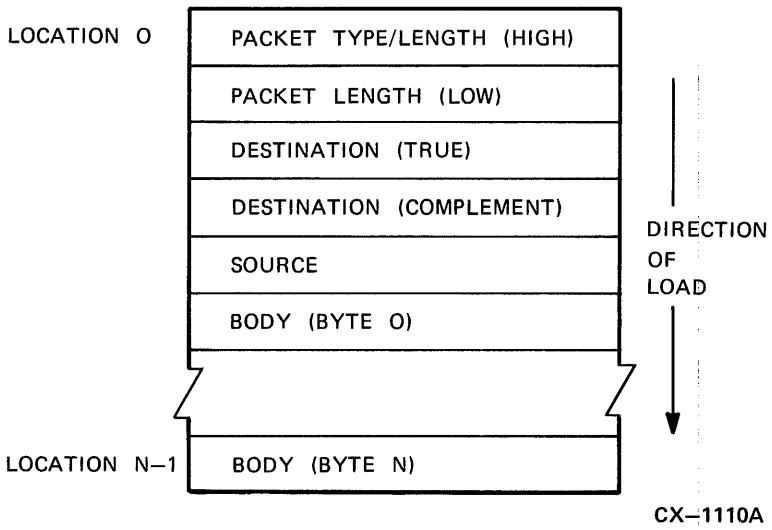


Figure 6-6 Transmit Buffer Load




The transmit command does one of the following:

1. Initiates transmission of a newly loaded buffer.
2. Retransmits a previously loaded buffer (according to the buffer and path indicated with the command and regardless of previous transmit commands on the same buffer):
    - On the same CI path (for retry)
    - On an alternate CI path (for recovery)

Only one transmit command can be issued and executed at a time. Therefore, the transmission must be completed or aborted, and the transmit status must be cleared before a second transmit command can be issued.

### 6.4.3 Reset Transmit Status: LINK CONTROL <3:0> = 0100

The reset transmit status command is not used by the PILA; it is used by the LINK. This command is always issued at the end of a transmission sequence—that is, in response to XMTR ATTN after reading the transmit status. It resets all transmit status bits except 5 and 6. The transmit status bits are described in Section 6.4.8.

### 6.4.4 Abort Transmission: LINK CONTROL <3:0> = 0101

The abort transmission command causes the PILA to abort a currently active transmission (by disabling the Transmit Address Counter and Output Register) and causes the link to set bit 4 (transmission aborted) in the transmit status. XMTR ATTN is also asserted by the LINK. If XMTR ATTN is asserted when the abort transmission command is issued, the abort is ignored.

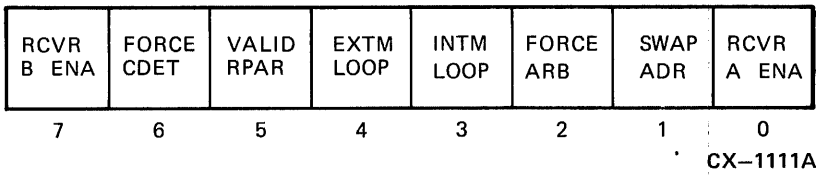
### 6.4.5 Enable Link Control: LINK CONTROL <3:0> = 0110

The enable link control command is not used by the PILA; it is used by the LINK. It enables the LINK controls indicated by ones in DATA <7:0> lines. The controls are shown in Figure 6-7 and described in Table 6-3.

**Table 6-3 Link Control Bit Description**

| Bit | State Of Bit | Description |
|---|---|---|
| 7 | 1 | Enables LINK receiver path B, making the node accessible to CI path B. |
| | 0 | Node ignores traffic on CI path B. INITIALIZE disables receiver path B. |
| 6 | 1 | Used in LINK internal maintenance loop mode. Forces the LINK to see a detected carrier, starting header time-out logic and stopping the arbitration sequence. |
| | 0 | Disables this function. Also disabled by INITIALIZE. |
| 5 | 1 | Forces the LINK to generate odd parity on received data. INITIALIZE also forces odd parity. |
| | 0 | Causes the LINK to generate even parity on received data. |
| 4 | 1 | External maintenance loop enable allows the LINK to receive its own transmission by looping on the selected CI path. |
| | 0 | Takes the LINK out of external maintenance loop mode. |
| 3 | 1 | Internal maintenance loop enable allows the LINK to receive its own transmission by looping inside the CI driver/receiver. |
| | 0 | Takes the LINK out of internal maintenance loop mode. |
| 2 | 1 | Causes the LINK to force successful arbitration and to transmit immediately when the transmit command (0011) is received from the K.pli. Should not be used on-line. |
| | 0 | Causes the LINK to perform normal arbitration in response to a transmit command. INITIALIZE causes the link to perform normal arbitration. |
| 1 | 1 | Causes the true and complement node addresses to be swapped. |
| | 0 | Causes the true and complement node addresses to be their normal values. INITIALIZE sets the node addresses to their normal values. |
| 0 | 1 | Enables LINK receiver path A, making the node accessible to CI path A. |
| | 0 | The node ignores traffic on CI path A. INITIALIZE disables receiver path A. |

Figure 6-7 Link Control Bits




### 6.4.6 Disable Link Control: LINK CONTROL <3:0> = 0111

The disable link control command is not used by the PILA; it is used by the LINK. It disables the LINK controls indicated by zeros in DATA <7:0> lines. The controls are shown in Figure 6-7 and described in Table 6-3.

### 6.4.7 Read Receiver Status: LINK CONTROL <3:0> = 1000

The read receiver status command enables the receiver status bits in the PILA onto the DATA <7:0> lines. The read receiver status command only reads eight of the receiver bits; CDA and CDB are read by the read transmitter status command (Section 6.4.8). All receiver status bits are synchronized to the PLI clock. Figure 6-8 shows the receiver status bits, and Table 6-4 describes the bits.

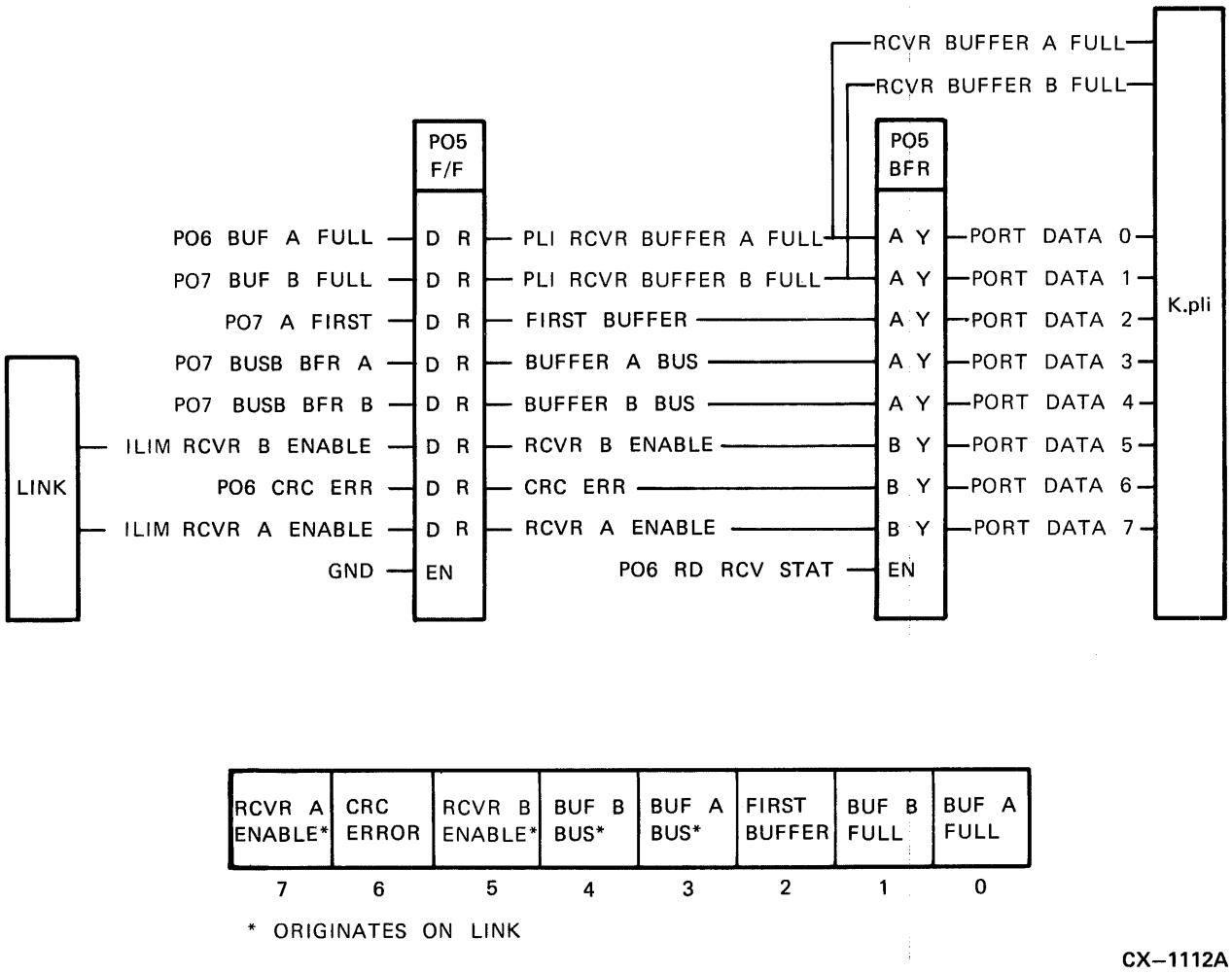
Table 6-4 Receiver Status Bits

| Bit | Name | Description |
|---|---|---|
| 7 | RCVR A ENABLE | Set by an enable link control command (0110) with bit 0 set. |
| 6 | CRC ERR | When set, indicates that the LINK detected a CRC error on the received message or data packet. CRC is neither generated nor checked on PILA internal loop mode ACK packets. This bit is never set in PILA maintenance mode.<br><br>The PILA uses dual CRC latches to indicate the CRC error for each receive buffer. Therefore, to select the CRC latch, a select buffer command (0011) must be issued before the read receiver status command.<br><br>CRC error is cleared by asserting INITIALIZE, or by releasing the buffer. |

Table 6-4 (Cont.) Receiver Status Bits

| Bit | Name | Description |
|---|---|---|
| 5 | RCVR B ENABLE | Set by an enable link control command (0110) with bit 7 set. |
| 4 | BUFFER B BUS | Indicates the CI path that carried the last received packet in Receiver Buffer B, where:<br><br>0 = CI path A<br>1 = CI path B<br><br>Valid only while PLI RCVR BUFFER B FULL is set. Cleared by INITIALIZE. |
| 3 | BUFFER A BUS | Indicates the CI path that carried the last received packet in Receiver Buffer A, where:<br><br>0 = CI path A<br>1 = CI path B<br><br>Valid only while PLI RCVR BUFFER A FULL is set. Cleared by INITIALIZE. |
| 2 | FIRST BUFFER | Indicates the buffer first filled when both RECEIVE BUFFER A FULL and RECEIVE BUFFER B FULL are either both set or both cleared, where:<br><br>0 = Buffer B was filled first<br>1 = Buffer A was filled first<br><br>FIRST BUFFER is cleared by INITIALIZE. |
| 1 | PLI RCVR BUFFER B FULL | Set when a valid packet has been stored in Receive Buffer B. This bit remains asserted until Receive Buffer B has been enabled (by set mode command - 1101). PLI RVCR BUFFER B FULL is also set by INITIALIZE (to prevent receptions until enabled). This status bit is also sent to the K.pli as a separate PLI line. |
| 0 | PLI RVCR BUFFER A FULL | Set when a valid packet has been stored in Receive Buffer A. This bit remains asserted until Receive Buffer A has been enabled (by set mode command—1101). PLI RCVR BUFFER A FULL is also set by INITIALIZE (to prevent receptions until enabled). This status bit is also sent to the K.PLI as a separate PLI line. |

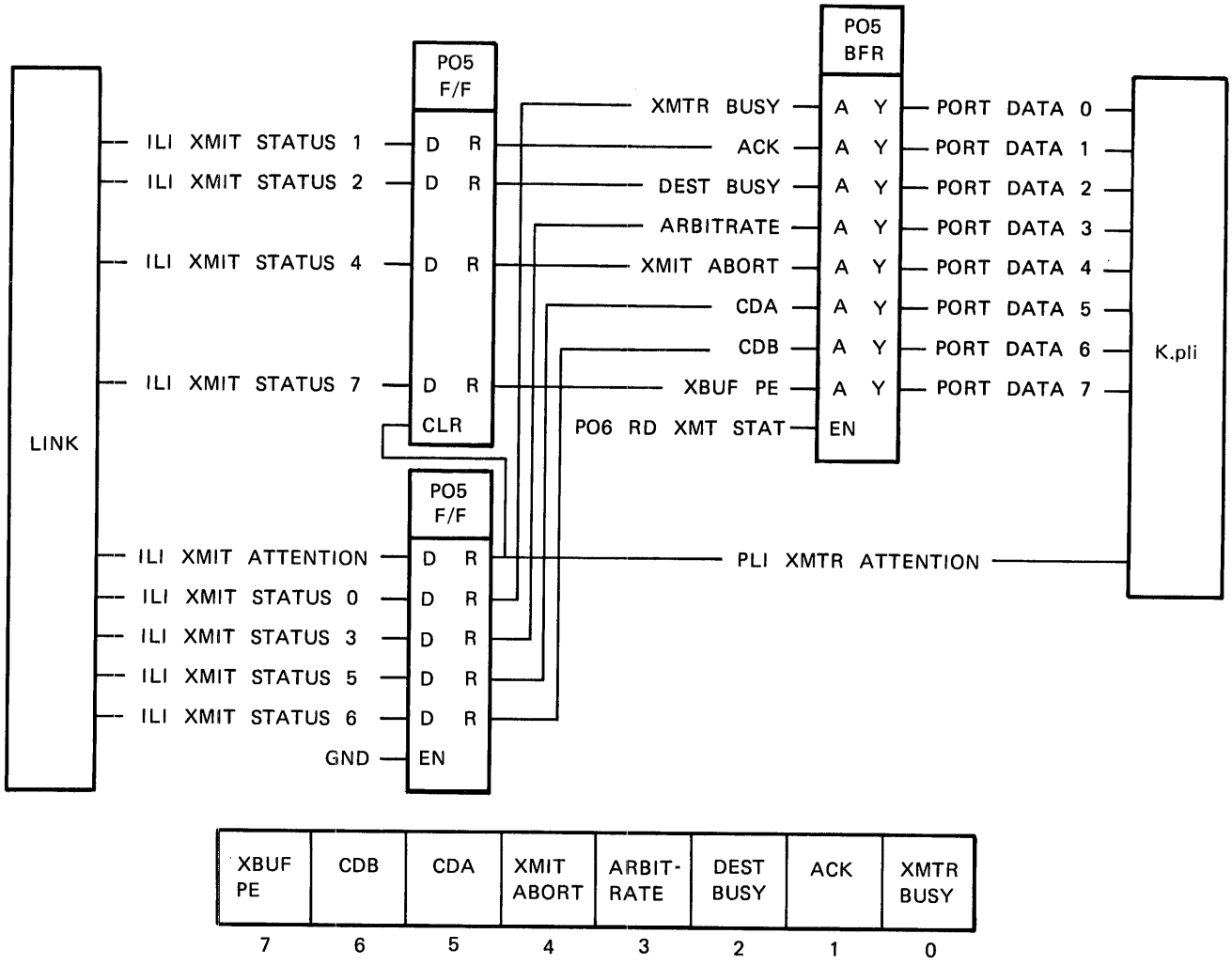
Figure 6-8 Receiver Status Bits




### 6.4.8 Read Transmitter Status: LINK CONTROL <3:0> = 1001

The read transmitter status command enables the transmit status from the PILA onto the DATA <7:0> lines. Bits CDB and CDA are receiver status bits. All the status bits originate on the LINK and are synchronized to the PLI clock. Figure 6-9 shows the transmitter status bits, and Table 6-5 describes the bits.

Figure 6-9 Transmitter Status Bits




| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| XBUF PE | CDB | CDA | XMIT ABORT | ARBIT-RATE | DEST BUSY | ACK | XMTR BUSY |

CX–1113A

Table 6-5 Transmit Status Bit Description

| Bit | Name | Description |
|---|---|---|
| 7 | XBUF PE | Set if the LINK detects a parity error on the transmit data during a transmission. A parity error asserts XMTR ATTN and aborts the transmission.<br><br>Cleared by INITIALIZE and the reset transmit status command (0100). |
| 6 | CDB* | Indicates the state of the carrier on CI path B, where:<br><br>0 = Carrier B off<br>1 = Carrier B on |
| 5 | CDA* | Indicates the state of the carrier on CI path A, where:<br><br>0 = Carrier A off<br>1 = Carrier A on |
| 4 | XMIT ABORT | Set when a transmission is aborted by the abort transmission command (0101). Asserts XMTR ATTN.<br><br>Cleared by INITIALIZE and the reset transmit status command (0100). |
| 3 | ARBITRATE | Set when the arbitration count has expired after a transmit command (0011).<br><br>Cleared by INITIALIZE and the reset transmit status command (0100). |
| 2 | DEST BUSY | Set by a NACK packet from the destination node. Asserts XMTR ATTN.<br><br>Cleared by INITIALIZE and the reset transmit status command (0100). |
| 1 | ACK | Set by an ACK packet from the destination node. Asserts XMTR ATTN.<br><br>Cleared by INITIALIZE and the reset transmit status command (0100). |
| 0 | XMTR BUSY | Set by a transmit command (0011). Indicates that a transmit sequence is in progress or that XMTR ATTN is asserted. Not set in synchronization with the PLI clock.<br><br>Cleared by INITIALIZE and the reset transmit status command (0100). |

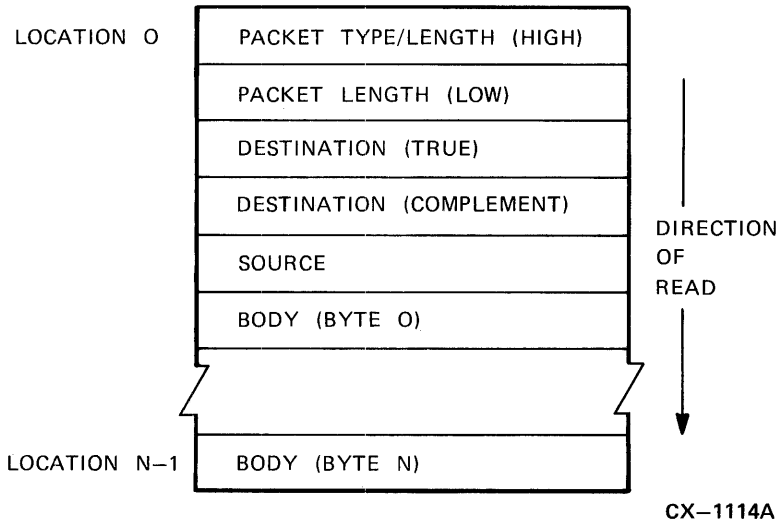
Note: Those bits marked with an asterisk (*) are receive buffer status bits.

### 6.4.9 Read Buffer: LINK CONTROL <3:0> = 1010

The read buffer command enables the contents of the currently addressed location in the receive or transmit buffer onto the DATA <7:0> lines. The buffer is selected by the last select buffer command (1110). The Address Counter for the selected buffer is incremented at the end of each cycle during which the read buffer command is asserted. The parity bit for the read data is passed to the K.pli on the PLI RECEIVE DATA PAR line. The read buffer command reads receive or transmit data starting at location 0 + offset. Offset can be any value from 0 to 15. The data is read sequentially starting with the first byte received or loaded and ending with the last byte received or loaded, as shown in Figure 6-10.

The read buffer command is also described in Chapter 7.

Figure 6-10 Receive or Transmit Buffer Read




### 6.4.10 Read Node Address: LINK CONTROL <3:0> = 1011

The read node address command reads the 8-bit CI node address from the LINK's node address dip switch onto the DATA <7:0> lines (Figure 6-11). Parity is generated by the PILA and passed on the PLI RECEIVE DATA PAR line.

### 6.4.11 Read Switches: LINK CONTROL <3:0> = 1100

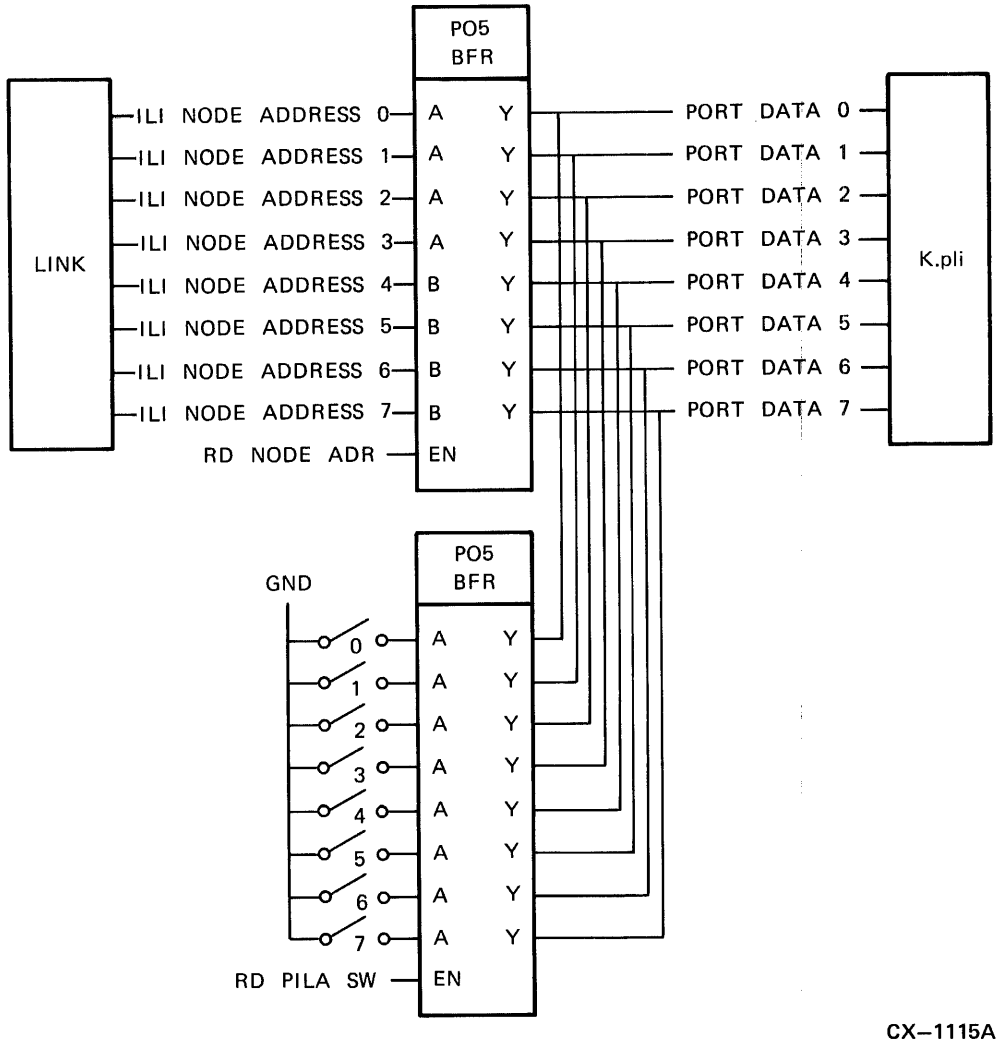
The read switches command reads the revision level of the PILA from the 8-bit dip switch onto the the DATA <7:0> lines (Figure 6-11). (All ones indicate the absence of the PILA.) Bits 7 through 5 indicate the module revision level as shown in Table 6-6.

Table 6-6 Module Revision Level

| 7 | 6 | 5 | Revision Letter |
|---|---|---|---|
| 0 | 0 | 0 | C |
| 0 | 0 | 1 | D |
| 0 | 1 | 0 | E |

Bits 4 through 0 indicate the rework level, with up to 32 engineering change orders allowed before a new etch is required. Parity is generated by the PILA and passed on the PLI RECEIVE DATA PAR line.

Figure 6-11 Read Switches and Node Address




### 6.4.12 Set Mode: LINK CONTROL <3:0> = 1101

The set mode command allows diagnostics to examine the PILA internal logic for fault isolation. The command functions are asserted on DATA <7:0> lines as shown in Figure 6-12 and described in Table 6-7.

Figure 6-12 PILA Mode Bits

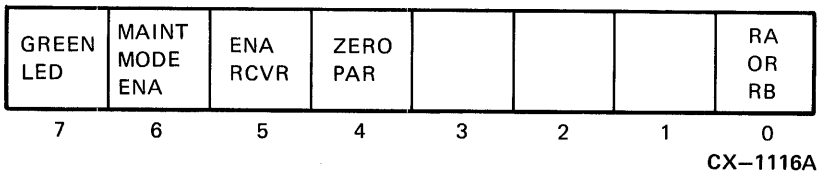



Table 6-7 Mode Bit Description

| Bit | Function | Description |
|---|---|---|
| 7 | Indicate Operational Status | K.pli sets and resets the green LED on the PILA to provide a visible indication of the PILA operational status. |
| 6 | Enable Maintenance Mode | Allows the transmit data to be looped into the receive buffer. Maintenance mode inhibits the SELECT line from the PILA to the LINK. Therefore, the LINK will not respond to any commands from the PILA. The red LED on the PILA indicates that the PILA is in maintenance mode. |
| 5 | Enable Receive Buffer | When set, releases Receive Buffer A or B (selected by bit 0) by clearing RECEIVER BUFFER A FULL or RECEIVER BUFFER B FULL. The buffer can then be used to store incoming CI packets. This signal also clears the CRC error flag.<br><br>This bit does not affect receive buffer full flags unless set. |
| 4 | Zero Parity | Valid only in maintenance mode. It must be zero to pass good transmit data parity. This bit is set to allow parity logic to be checked; when set, it forces the transmit buffer parity bit to zero (does not invert parity). |
| 3 | Not used | |
| 2 | Not used | |
| 1 | Not used | |
| 0 | Receive Buffer A or B | If not set, bit 0 releases Receive Buffer A; if set, bit 0 releases Receive Buffer B. Valid only when bit 5 is set. (See Table 6-8.) |

Table 6-8 Mode Bit 0 and 5

| Bit 0 | Bit 5 | Description |
|---|---|---|
| 0 | X | No effect on receive buffer full flags |
| 1 | 0 | Release Receive Buffer A and clear RECEIVE BUFFER A FULL |
| 1 | 1 | Release Receive Buffer B and clear RECEIVE BUFFER B FULL |

X = Does not matter whether it is a 1 or 0

### 6.4.13 Select Buffer: LINK CONTROL <3:0> = 1110

The select buffer command either selects the receive or transmit buffer. It selects the receive buffer to be read by subsequent read buffer commands (1010). It selects the transmit buffer to be loaded by subsequent load transmit buffer commands (0001). DATA <6:0> lines are asserted by the K.pli and defined in Table 6-9.

Table 6-9 Data Lines <6:0> Definitions

| Data Line 6 5 4 3 | 2 | 1 | 0 | Description |
|---|---|---|---|---|
| | 0 | 1 | 0 | Load Transmit Buffer A and load offset address |
| Offset | 0 | 1 | 1 | Load Transmit Buffer B and load offset address |
| Address | 1 | 0 | 0 | Read Receive Buffer A and load offset address |
| Bits | 1 | 0 | 1 | Read Receive Buffer B and load offset address |
| 3 2 1 0 | 1 | 1 | 0 | Read Transmit Buffer A and load offset address |
| | 1 | 1 | 1 | Read Transmit Buffer B and load offset address |

Offset address in DATA <6:3> are the offset from address 0. The K.pli cannot load an address offset greater than 15 into the address counter of the receive or transmit buffers and, therefore, cannot randomly access the buffers with an address greater than 15.

The two receive buffers are read-only; that is, they cannot be directly loaded by PLI commands. Either receive buffer can be loaded with data from either transmit buffer when the K.pli issues a transmit command and the packet buffer is in maintenance mode.

### 6.4.14 Sync: LINK CONTROL <3:0> = 1111

The sync command causes an output from the PILA command decoder. The K.pli can use this output to synchronize measurements.

## 6.5 STATUS LEDS

The yellow LED is a test point indicator. This LED is found only on some early E-rev etch modules. Associated with this LED is a test point that is accessible from the upper-right corner of the board (top handle-edge when module is inserted vertically). The LED and test point circuit allow you to detect logic levels. The circuit is capable of operating to 25 MHz, slowing down pulses to a rate of about 10 Hz. The LED being on indicates a logic high; the LED being off indicates a logic low.

The red LED is forced off by INITIALIZE and automatically turns on after INITIALIZE to visibly indicate that the PILA is in Maintenance Mode (the initial state after INIT). When the K.pli gives the SET MODE command to leave maintenance mode, this LED turns off. In maintenance mode, the output of the transmit buffer is wrapped around into the receive buffer. The diagnostic software in the K.pli tests the buffers in the PILA before testing the wrap mode in the LINK. When the K.pli puts the PILA in maintenance mode, it sets the the RECEIVE BUFFER A and B FULL so that it cannot receive anything from the LINK.

The green LED is forced off by INITIALIZE. Otherwise, it is set or reset by the K.pli diagnostic software to visibly indicate the PILA operational status.

# CHAPTER 7
# PORT PROCESSOR MODULE

## 7.1 INTRODUCTION

This chapter contains the logical and functional description of the Port Processor Module (K.pli). The K.pli is the controller for the Port Buffer Module (PILA) and the Port Link Module (LINK).

The K.pli is an L series extended hex module that controls the interconnect between the HSC internal Control and Data Buses and the Computer Interconnect Bus (CI). It uses several of the 2900-family bit-slice devices supported by medium/small scale integration transistor/transistor logic. The architecture of K.pli consists of 2900-family components organized as dual sequencers alternately accessing a common control store and supporting a single 16-bit data path.

The K.pli does not use control registers to communicate with the P.ioc and other controllers in HSC. Communications between K.pli and the P.ioc and other controllers of the subsystem are performed using memory resident control structures.

The functional and diagnostic microcode are resident in PROM Control Store.

## 7.2 STATUS LEDS

The red and green LEDs indicate the status of the K.pli module. The module powers up with the red LED on and the green LED off. Upon successful completion of the power-up diagnostics, the red LED is turned off and the green LED is turned on.

## 7.3 MICROINSTRUCTION WORD

The microinstruction word from Control Store controls the operation of the K.pli. The microinstruction word is 48 bits wide. Each word is divided into sixteen fields and each field controls different portions of the K.pli hardware. Figure 7-1 shows the fields and their positions in the microinstruction word. Table 7-1 contains a brief description of each field. Section 7.25 has a detailed description of each field.

Figure 7-1 Microinstruction Word

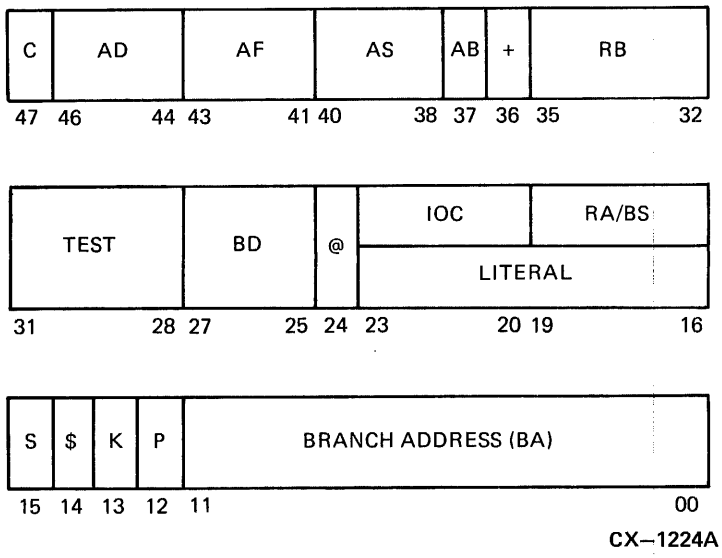



Table 7-1 Microinstruction Word Field Description

| Field | Description |
|---|---|
| C | The C field is the carry into the ALU for the current microinstruction. It steers the literal to the high byte for AF fields 3 through 7. |
| AD | The AD field, ALU destination, determines the routing of the ALU output data internal to the 2901. |
| AF | The AF field, ALU function, determines the basic operation to be performed by the ALU. |
| AS | The AS field, ALU source, selects two of the five possible sources for the ALU. |
| AB | The AB field, ALU to BUS, determines whether the D OUT lines will be driven by the 2901 ALU or if the D IN contents will be transferred to the D OUT lines to bypass the ALU. It also enables or disables the BS field decoders. |

Table 7-1 (Cont.) Microinstruction Word Field Description

| Field | Description |
|---|---|
| + | The + field, EXEC IF COND TRUE, is one bit long. This bit is combined with the following signals to generate INHIBIT EXEC (inhibit execute).<br>• TST COND TR (test condition true)<br>• COND EXC (conditional execute) ($ field)<br>INHIBIT EXEC prevents:<br>• Any ALU operation of each instruction for which it is active<br>• The IOC Latches from being enabled<br>• The destination decoders from being enabled |
| RB | The RB field, B Port Register, selects one of the sixteen internal 2901 registers to be read and/or written by the 2901 ALU. If the AS field contains a value of $5_8$ or $7_8$, the A Port Register field (RA field) is forced to the same value as the RB field. |
| TEST | The TEST field, TEST CONDITION <03:00>, selects one of sixteen signals for the conditional operations of the sequencers. The hardware includes two separate selectors, one for each of the two sequencers. |
| BD | The BD field, bus destination, selects the register that will be written with the data on the D OUT lines. |
| @ | The @ field, IOC ENABLE, permits the IOC field to modify the IOC Latches when set. This bit is not set when using the LITERAL field. |
| IOC | The IOC field, input/output control, if enabled by IOC ENABLE(@), modifies the IOC Latches. Bits <22:20> select the desired latch, and bit 23 sets or clears that latch. |
| RA | The RA field, A Port Register, selects one of the sixteen internal 2901 registers to be operated on by the ALU. The RA field is a shared field and is only enabled if the AS field does not equal 5 or 7 and bit AB does not specify the LITERAL field. |
| BS | The BS field, bus source, selects one of the eight K.pli registers which are external to the ALU. The contents of the selected register are enabled onto the D IN lines, routed through the input multiplexer, and presented to the D inputs of the 2901 ALU. |
| S | The S field, test sense, is XORed with the output of the Control and Data Test Select multiplexer to generate TST COND TR (test condition true). TST COND TR is then combined with branch address equal to $7777_8$ and JUMP CNTL 1 to determine whether the sequencer will output the Program Counter, the Stack, or the Branch Address. |

Table 7-1 (Cont.) Microinstruction Word Field Description

| Field | Description |
|---|---|
| $ | The $ field, conditional execute, enables all instructions when clear. When set, this bit enables the conditional execution of the IOC, bus destination, and ALU instructions. |
| K | The K field, jump control, is logically combined with TST COND TR and BR ADR <11:00> equal to all ones ($7777_8$) to control the next operation of the 2911 Sequencers. |
| P | The P field, instruction parity, is the single parity bit for the 48-bit microinstruction word. |
| BA | The BA field, branch address, provides the instruction address for conditional and unconditional branches and subroutine calls. Also, during the execution of certain instructions, this field provides supplemental data and control information. |

## 7.4 PORT PROCESSOR MODULE BLOCK DIAGRAM

Figure 7-2 is a block diagram of the K.pli showing the principal hardware elements and data paths. Each of these elements will be described in the following sections.

## 7.5 SEQUENCERS

In the lower-left-hand corner of Figure 7-2 you will see the two sequencers. The use of these two sequencers which alternately and independently access a common Control Store is the heart of the K.pli architecture. The two sequencers, in conjunction with the Control Store and Arithmetic and Logic Unit (ALU), are designated Control Sequencer and Data Sequencer. Each sequencer controls different portions of the external interfaces, and each has separate and independent functions.

The Data Sequencer handles all aspects of a data transfer between Data Memory and the PILA buffers. During a data transfer, the Data Sequencer is busy (at the maximum data rate); at all other times it is idle.

The Control Sequencer is responsible for all other work the K.pli does, including examining queues for work to do, acquiring and retiring buffers, and so on. The activity level of this sequencer is fairly high and fairly constant.

One sequencer does not help out or share the work assigned to the other. Each sequencer has its own job to do.

With a clock period of 150 nanoseconds, each sequencer has an apparent micro-cycle time of 300 nanoseconds. As a result, no effective time is lost by either sequencer when executing a branch since the branching sequencer is accessing Control Store while the other sequencer is executing.

The sequencers are the control section of the K.pli. Each sequencer consists of three 2911 bit-slice devices connected to form a 12-bit address. Figure 7-3 is a block diagram of one 2911. Each bit-slice is 4 bits wide and consists of the following logic:

- Program Counter Register
- Incrementer
- Four Register Stack
- Stack Pointer
- Next Address Multiplexer
- Internal logic to control these functions

The 12-bit address allows the sequencer to address 4 Kwords of Control Store. This address always points to the next instruction by automatically incrementing after the instruction fetch. The stack is 12-bits wide and 4 words deep and is used along with the stack pointer during call and return operations.

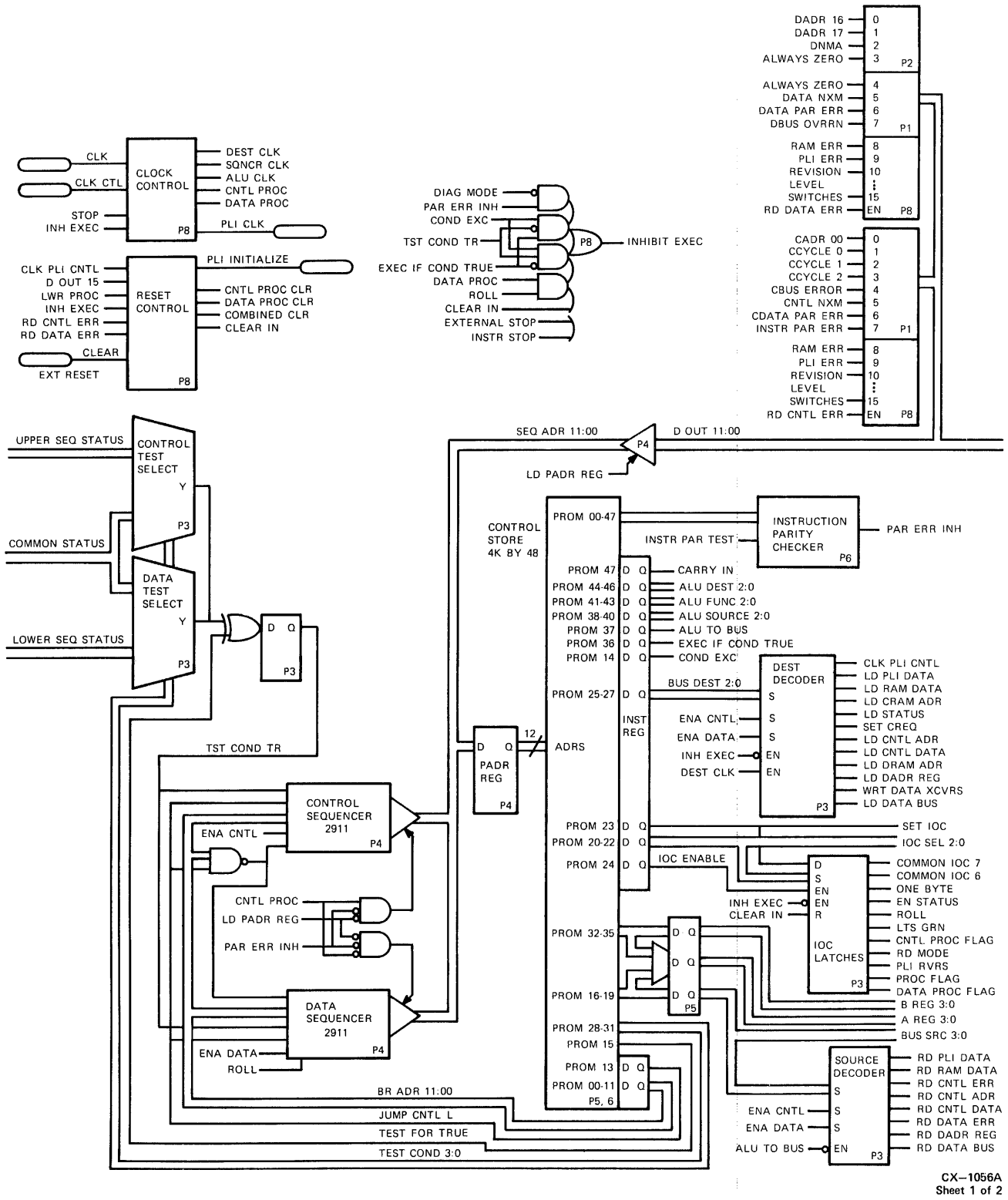
The internal multiplexer within the sequencers is capable of choosing one of the following for output to the program address register:

- The Stack
- The Program Counter Register
- The branch address supplied from the Control Store

As connected on the K.pli, the sequencers perform the following operations:

- CONTINUE—Increments the Program Counter Register by one and places the new address on the SEQ ADR <11:00> lines.
- RETURN FROM SUBROUTINE (pop stack)—Writes the Program Counter Register with the current value of the stack and gates this value to the SEQ ADR <11:00> lines. The Stack Pointer is then decremented.
- JUMP TO BRANCH ADDRESS—Writes the Program Counter Register with the current value of the branch address (supplied externally) and gates this address to the SEQ ADR <11:00>. The Stack and Stack Pointer are not affected.
- SUBROUTINE CALL (push stack)—Increments the Stack Pointer and pushes the current value of the Program Counter Register onto the Stack. Writes the Program Counter Register with the current value of BR ADR <11:00> and gates this address to SEQ ADR <11:00>.

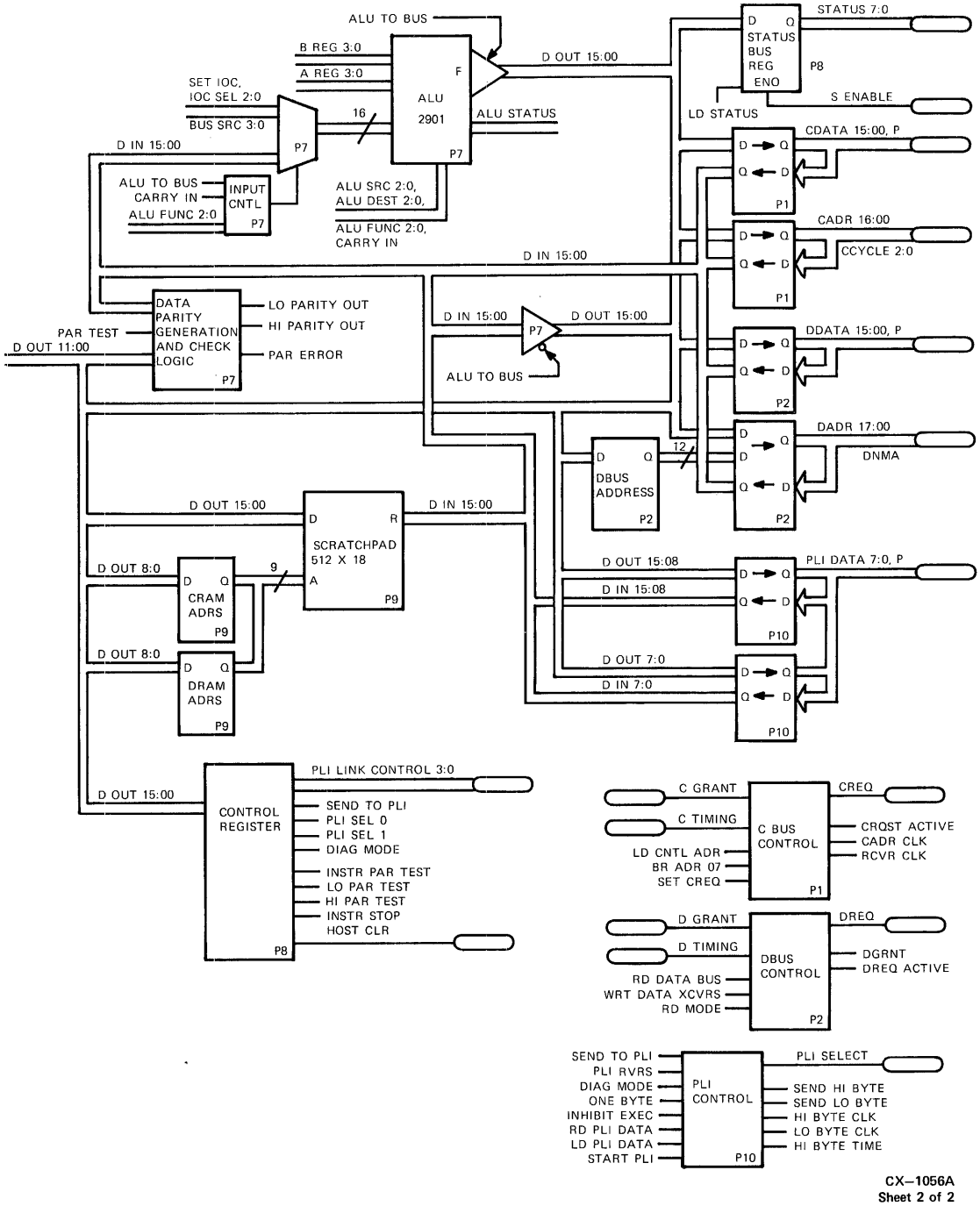
## Figure 7-2 Port Processor Module Block Diagram




CX–1056A
Sheet 1 of 2

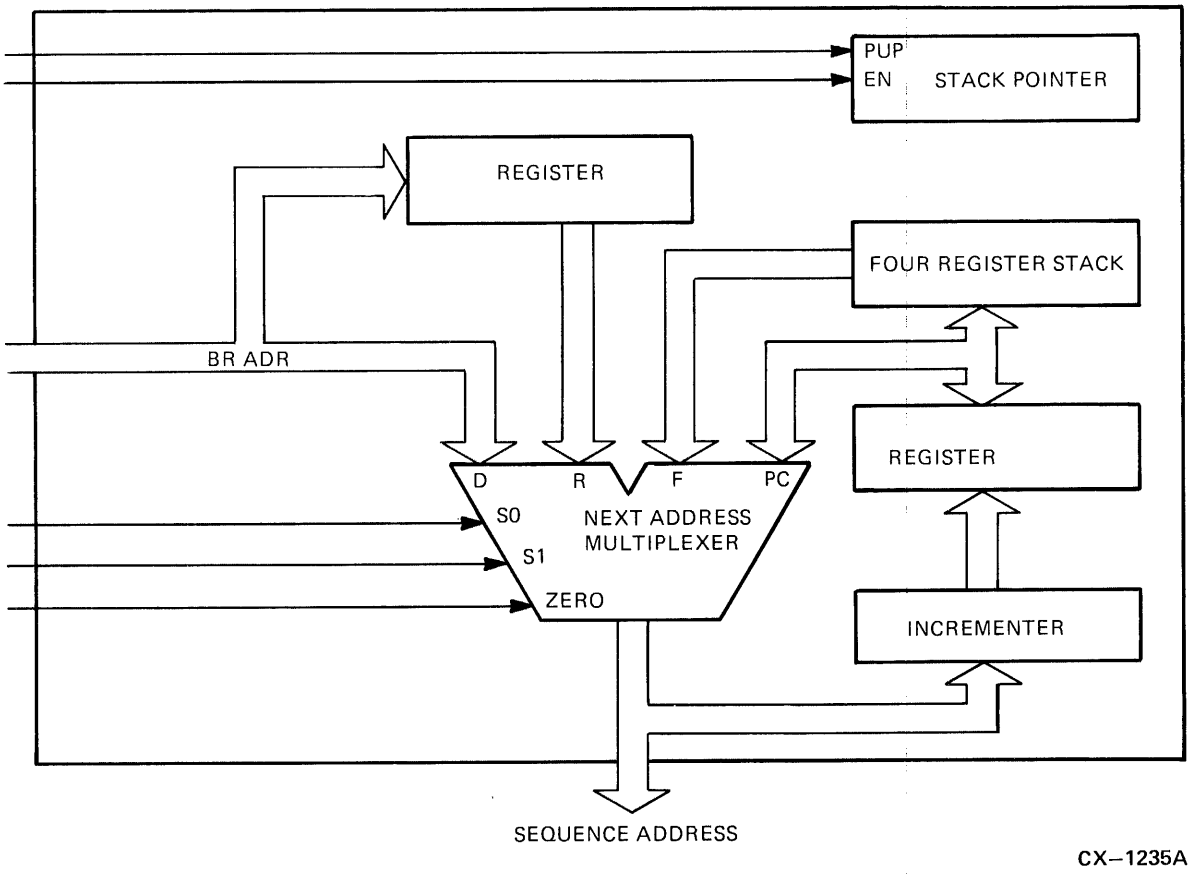
(Continued on next page)

Figure 7-2 (Cont.) Port Processor Module Block Diagram




CX–1056A
Sheet 2 of 2

Figure 7-3 2911 Sequencer Block Diagram



## 7.6 TEST MULTIPLEXER

The sequencers must have the ability to test different conditions on the K.pli, PILA, and LINK. Figure 7-2 shows two multiplexers just above the two sequencers. They are two separate 16-input multiplexers to select test conditions for the two sequencers. Bits 31 through 28 of the Control Store generate TEST COND <3:0> which select one of the 16 inputs. Either the true or the complement of the selected test condition may be selected. This is accomplished by the TEST FOR TRUE (bit 15 from the Control Store). A synchronization circuit is included to guarantee correct testing of asynchronous test conditions. Available test conditions include:

- PORT/LINK Interface (PLI) status
- Arithmetic Logic Unit (ALU) status
- Input/Output Control (IOC) Latches
- Error indicators

## 7.7 PROGRAM ADDRESS REGISTER

The sequencers address the Control Store through the Program Address Register shown in Figure 7-2. This register alternately accepts the outputs of one sequencer or the other on each clock cycle and presents the instruction address to the Control Store for the fetch operation. The fetch cycle for one sequencer occurs in parallel with the execute cycle of the other sequencer.

The sequencer outputs may be disabled and the Program Address Register forcibly written from the ALU by LD PADR REG. For example, the ALU may execute a calculated jump by calculating the jump address from a table and writing the data to the Program Address Register from D OUT <11:00>. During this time, the outputs of the sequencers are disabled.

## 7.8 CONTROL STORE

The PROM Control Store is shown in Figure 7-2 to the right of the sequencers. All microcode instructions are contained in the Control Store. It is 48 bits wide (including parity). The Control Store consists of twelve 2K by 8 PROMs, with a maximum access time of 90 nanoseconds. One microinstruction is fetched from the Control Store during each 150 nanosecond clock cycle. At the end of the clock cycle, the microinstruction is written into the latch where it is held for the following execution cycle.

The Control Sequencer has the ability, in diagnostic mode, to command the Data Sequencer to fetch and test parity consecutively in all Control Store locations. This is accomplished with the ROLL signal on the output of the IOC latches.

A detailed description of each of the bits in the microinstruction is covered in Section 7.25.

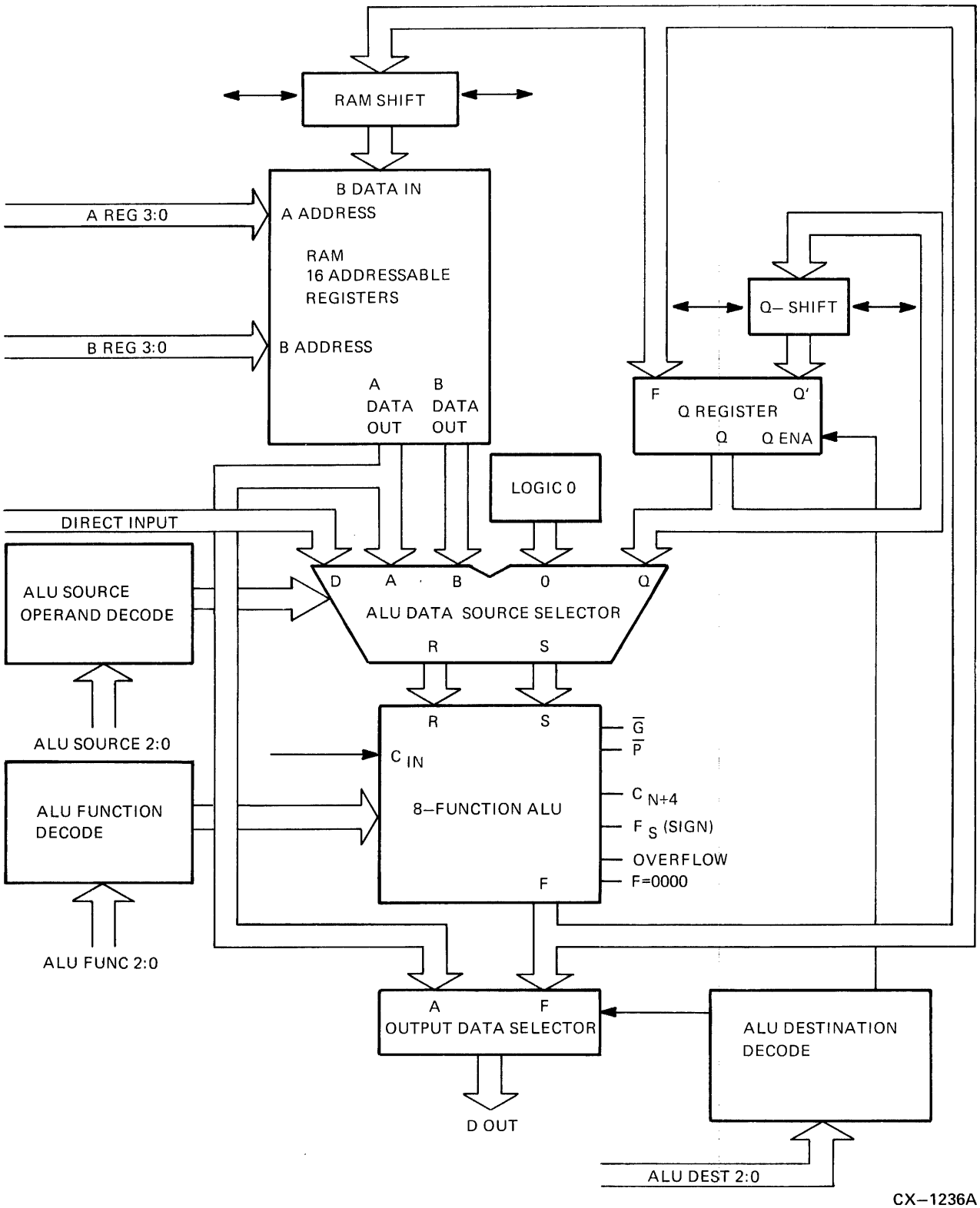
## 7.9 ARITHMETIC AND LOGIC UNIT

Figure 7-2 shows the Arithmetic and Logic Unit (ALU) in the upper-right-hand side. The ALU is implemented with four 2901 bit-slice devices, providing a 16-bit data path. The 2901 bit-slice ALU chip can perform addition, subtraction, and logic operations. Its internal data paths and registers are four bits wide (Figure 7-4). Nine control signals and eight register-select lines determine the ALU function, operand source, and destination.

The ALU has the following characteristics:

- A 2-port RAM comprised of 16 registers. The RAM registers are read via their A or B Data Out ports.
- A 4-bit shift register on the RAM data input. The output of this register is connected in ring counter fashion to the input.
- A 4-bit Q register for assisting with multiplication and division algorithms.

Figure 7-4 2901 ALU Block Diagram



- A 4-bit shift register on the Q register input.
- A multiplexer that selects the output of the Q register shifter or the ALU output for input to the Q Register.
- Two multiplexers that select the ALU inputs. These inputs may be the direct input, the A or B RAM, logic zero, or the Q Register.
- An output multiplexer capable of selecting the ALU output or port A of the RAM and gating it to D OUT.
- A 2-input ALU capable of performing the operations shown in Table 7-2.

Table 7-2 ALU Operations

| Operation | Operands Used |
|---|---|
| Addition | R + S<br>R + S + 1 |
| Subtraction | R—S<br>S—R<br>R—S—1<br>S—R—1 |
| Logical OR | R OR S |
| Logical AND | R AND S |
| Logical NOTRS | (NOT R) AND S |
| Logical XOR | R XOR S |
| Logical XNOR | R XNOR S |

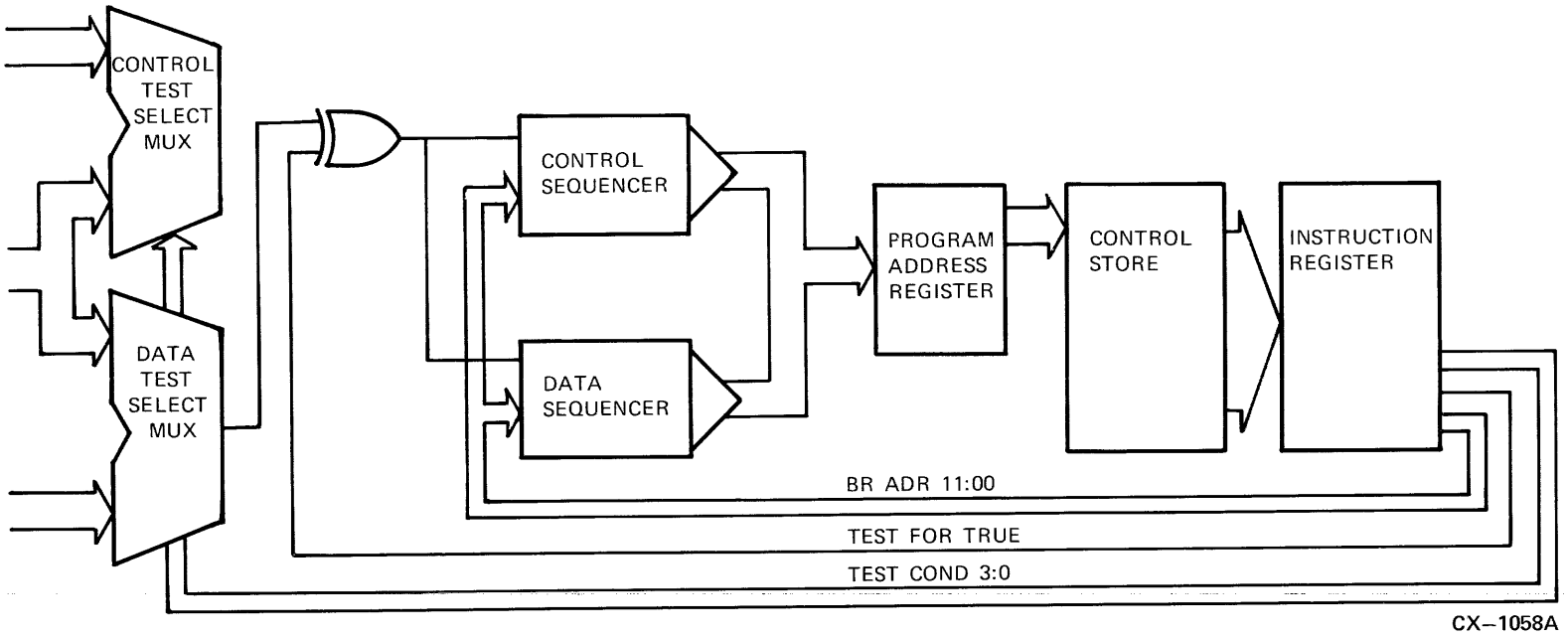
Organization of the 2901 is such that shift operations are separate from, and independent of, the arithmetic and logical operations and are effective only on the inputs to the internal Q and RAM registers. The results of ALU operations are not shifted at the Y outputs. The ALU state flags reflect the ALU results before any shift.

Shifts may be performed at the RAM register inputs or at the inputs of both the RAM and Q registers in one instruction. All are sixteen bit shifts, and the ALU carry is never affected by any shift operation. Shifts performed on the Q register are end off; that is, on a left shift the contents of bit 15 are lost, and on a right shift the contents of bit 00 are lost. Shifts performed on the RAM registers are circular; that is, on a left shift the contents of bit 15 appear in bit 00, and on a right shift the contents of bit 00 appear in bit 15.

## 7.10 PIPELINING CONCEPTS

Now that you have read about the elements that make up the computing power on the K.pli, let's look at how the sequencers, Program Address Register, Control Store, and the Instruction Register form a pipeline. Pipelined designs decrease instruction cycle time by performing fetch and execute operations in parallel (Figure 7-5). The Instruction Register holds the current microinstruction that controls the processing section and helps the control sections select the next address. Thus, the processing sections are operating on the current microinstruction while the control sections are fetching the next microinstruction from the Control Store.

Figure 7-5 K.pli Double Pipelining



## 7.11 K.pli PIPELINE

The K.pli uses a form of microinstruction/address/status pipelining (double pipelining). Registers are provided for the current microinstruction (Instruction Register), the sequencer address (Program Address Register), and the ALU/real-time status (Test Select multiplexers). These three registers provide the double pipelining design with three parallel paths. These paths simultaneously perform the following operations:

- The current microinstruction is gated into the Instruction Register. It controls the ALU along with the branch address of the Control and Data Sequencers.
- The sequencers generate a new value for the Program Address Register.
- The current value in the Program Address Register fetches the next microinstruction for the Control Store. Control Store bits <31:28> select the test conditions for the Test Select multiplexers.

## 7.12 PIPELINE TIMING

Figure 7-6 shows the timing for a instruction fetch-execute sequence. The SQNCR CLK supplies the timing for all the control timing. CNTL PROC is high when the Control Sequencer is active. CNTL PROC is low when the Data Sequencer is active. The timing in Figure 7-6 starts with the Control Processor gating the address on SEQ ADR <11:00>. Then the following steps occur:

- On the high to low transition of CNTL PROC:
  - The address from the Control Sequencer is clocked into the Program Address Register
  - The Data Sequencer gates its program address onto SEQ ADR <11:00>
  - The Control Store accesses the microinstruction for the Control Sequencer
- On the high to low transition the next CNTL PROC signal:
  - The microinstruction word is clocked into the Instruction Register to be used by the Control Sequencer in its next cycle
  - The program address of the Data Sequencer is clocked into the Program Address Register
  - The Control Sequencer gates its program address onto SEQ ADR <11:00>

The steps listed above tend to indicate an order of operation. However, SQNCR CLK occurs at the same time on the sequencers, Program Address Register, and Instruction Register. As a result, their operations occur simultaneously.

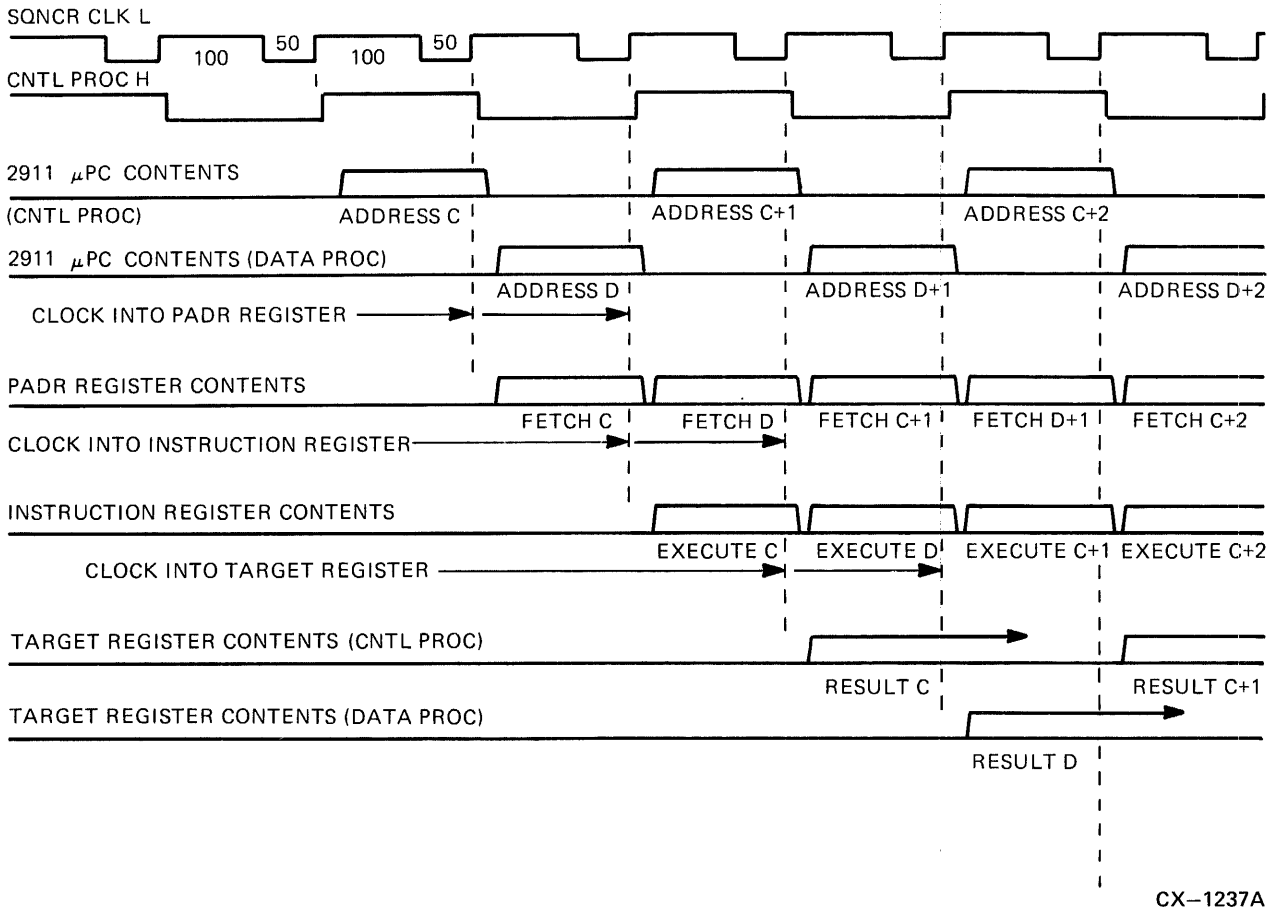
## 7.13 SCRATCHPAD MEMORY

The K.pli has a Scratchpad Memory of 512 words of 16 data bits and 2 parity bits. The memory is written or read under microcode control (Figure 7-2). Included are two 9-bit Scratchpad address registers, one for each of the two sequencers:

- CRAM ADRS
- DRAM ADRS

The address registers are loaded, but not read, by microcode. The Scratchpad data may be accessed by either sequencer.

Figure 7-6 Pipeline Timing for an Instruction Fetch-Execute Sequence

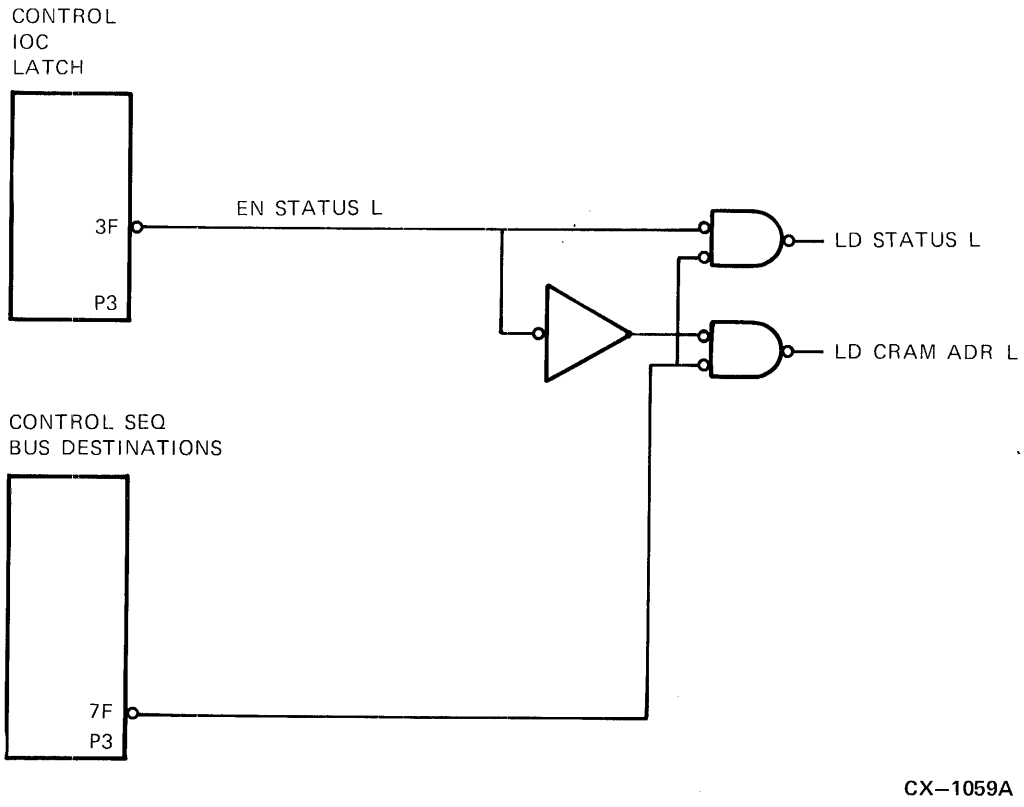



Because of the lack of decoding bits, some signals have to be shared. The generation of LD CRAM ADR is one of these signals (Figure 7-7). The decode of a 7 from the Control Sequencer Bus Destination Decoder enables LD STATUS or LD CRAM ADR depending on the state of the EN STATUS signal from the Control IOC Latch bit 3. If IOC 3 is set, the Control Sequencer Scratchpad Address Register is written to.

## 7.14 STATUS REGISTERS

The Control and Data Sequencers read the two Status Registers to determine status in the K.pli or PLI Interface. Figure 7-2 shows the two registers in the center top of the block diagram. Figure 7-8 shows the two registers. These two registers are made up of different pieces of logic throughout the K.pli.

Figure 7-7 Generation of LD CRAM ADR




The RD CNTL ERR signal reads the six status bits associated with Control Memory, Scratchpad, and PLI Interface accesses. It also reads the CCYCLE lines, the Control Memory Address least significant bit, and the six revision level bits. Control error bits 4, 5, 6, 7, and 8 are automatically cleared following a RD CNTL ERR. They are also cleared if the Control Sequencer writes the Control Register with data bit 15 set.

The RD DATA ERR signal reads the five status bits associated with Data Memory, Scratchpad, and PLI Interface accesses. It also reads the DNMA bit, the two most significant Data Address bits, and the six revision level bits. Data error bits 5, 6, 7, and 9 are automatically cleared following a RD DATA ERR signal. They are also cleared if the Data Sequencer writes the Control Register with data bit 15 set. Table 7-3 shows the bit assignments of these two registers.

Figure 7-8 Status Registers

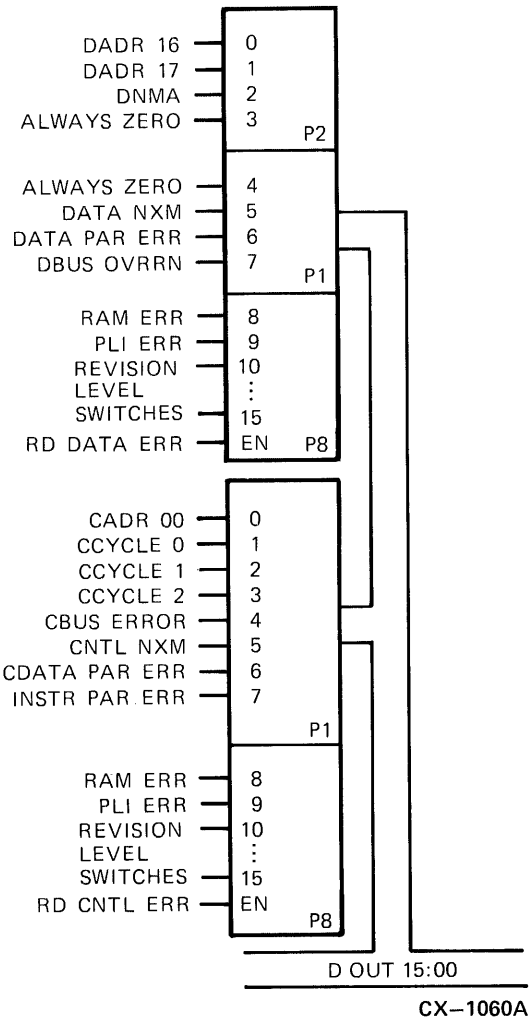


Table 7-3 Status Registers Bit Assignment

| Control Sequencer | Data Bit | Data Sequencer |
|---|---|---|
| CADR 00 | D IN 00 | DADR 16 |
| CCYCLE 0 | D IN 01 | DADR 17 |
| CCYCLE 1 | D IN 02 | DNMA |
| CCYCLE 2 | D IN 03 | Always zero |
| CBUS ERROR | D IN 04 | Always zero |
| CNTL NXM | D IN 05 | DATA NXM |
| CDATA PAR ERR | D IN 06 | DATA PAR ERR |
| INSTR PAR ERR | D IN 07 | DBUS OVRRN |
| RAM ERR | D IN 08 | RAM ERR |
| PLI ERR | D IN 09 | PLI ERR |
| Revision level | D IN <10:15> | Revision level |

The following error bits are ORed together into CNTL ERR SUM signal that is an input to the Control Test Select multiplexer:

- CBUS ERROR
- CDATA PAR ERR
- CNTL NXM
- INSTR PAR ERR
- RAM ERR

The following error bits are ORed together into DATA ERR SUM signal that is an input to the Data Test Select multiplexer:

- PLI ERR
- DBUS OVRRN
- RAM ERR
- DATA PAR ERR
- DATA NXM

The six revision level bits are encoded to represent K.pli revision levels shown in Table 7-4.

**NOTE**
**Switches 7 and 8 are not used.**

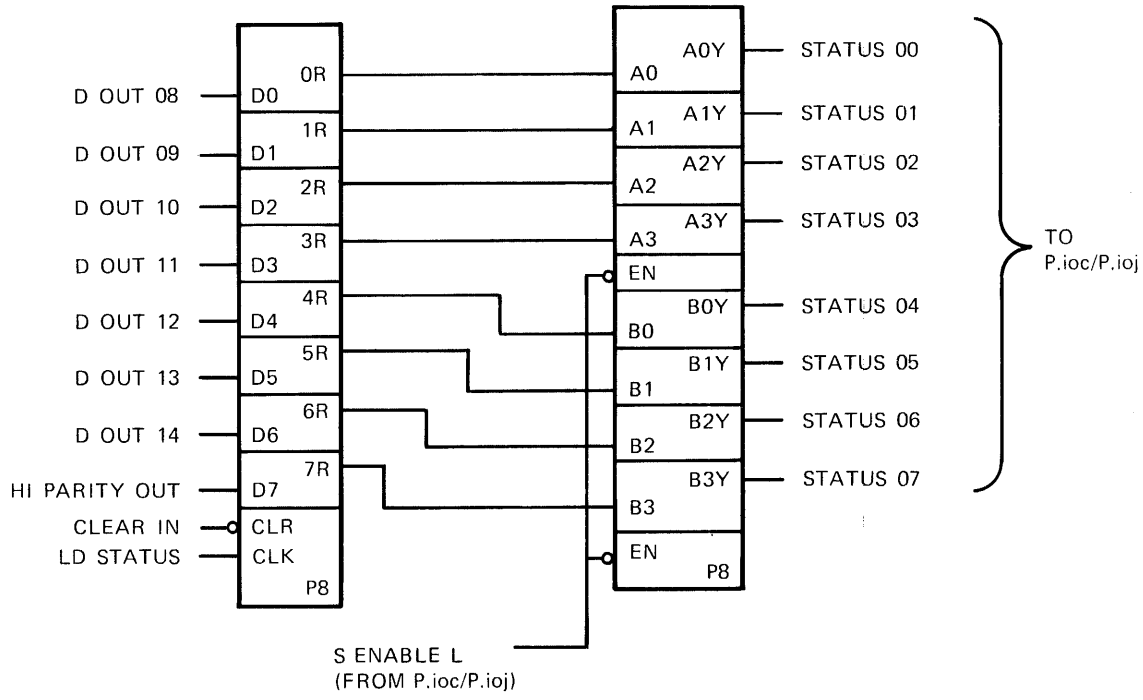
Table 7-4 K.pli Revision Level Switches

| Switch Number | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Board Rev. C | 0 | 0 | 0 | 1 | 0 | 0 |

## 7.15 STATUS BUS REGISTER

The Status Bus Register is gated onto the backplane by the P.ioc to check the status of the host interface modules. This register is written to by K.pli microcode after every major state change within the K.pli microcode. The Status Bus Register is shown in the upper-right-hand corner of Figure 7-2. The Status Bus Register is written to by microcode from D OUT <14:08,HP>. Notice that bit 7 is the parity bit for the other 7 bits. Figure 7-9 shows the Status Bus Register and Table 7-5 describes the bits.

Figure 7-9 Status Bus Register




CX–1061A

Table 7-5 Definitions of Status Bus Register Bits

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Status Bus Register Bits |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Initial state resulting from clearing the K. (Note the bad parity indication.) |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | Value resulting if no K is present. (Note the bad parity indication.) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | K.pli passed its internal diagnostics. |
| P | 0 | 1 | E | E | E | E | E | K.pli has declared a structure error in the system and performed an interrupt cycle on the Control Bus with priority 7. The error was detected by operational microcode. The E is the error code. Refer to the service manual for the error code. |
| P | 1 | 0 | 0 | F | F | F | F | A fault was detected by the diagnostics. The F is the failing test number. Refer to the service manual for the failing test numbers. |

P is the Parity bit

Because of the lack of decoding bits, some signals have to be shared. The generation of LD STATUS is one of these signals (Figure 7-7). The decode of a 7 from the Control Sequencer Bus Destination Decoder enables LD STATUS or LD CRAM ADR depending on the state of the EN STATUS signal from the Control IOC Latch bit 3. If IOC 3 is reset, the Status Bus Register is written to.

## 7.16 CONTROL BUS INTERFACE

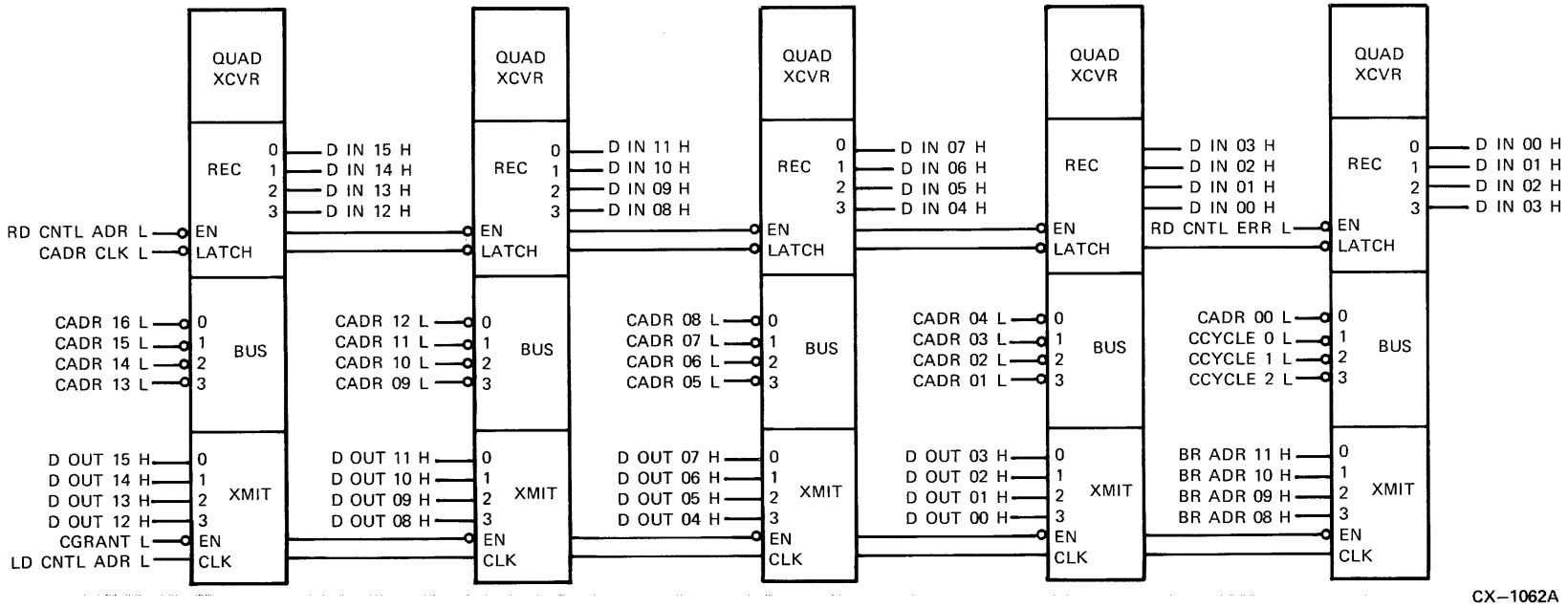
The Control Bus Interface is associated with the Control Sequencer and is used to access the HSC Control Memory that contains the control structures defining the work to be performed by K.pli. The Control Sequencer scans predefined areas of this memory for work and manipulates the control structures. The K.pli is capable of maintaining data rates up to 2.22 megabytes per second on this bus. For Control Bus timing information, refer to Chapter 3. The Control Bus Interface is shown in the upper-right-hand corner of Figure 7-2.

The interface is implemented with 2917A type latched bus transceivers plus a separate byte parity generator, which is shared with the:

- Data Memory Interface
- PLI Interface
- Scratchpad

The Control Bus Address Transceivers are read or written by microcode. When a Control Memory write is performed, the hardware provides an automatic wrap around so that the contents may be read and verified. These transceivers are latches only. The Control Bus Address must be incremented by microcode and rewritten for each Control Memory access. Note that the result obtained from reading the Control Bus Address does not have the same organization required when writing it (Figure 7-10). The value obtained from reading the transceivers is the address used in the last completed bus cycle.

Figure 7-10 Control Memory Address Register



The Control Bus Data Transceivers may be read or written by microcode. When a Control Memory write is performed, the hardware provides automatic wrap around so that the contents may be read and verified. This read is in exactly the same organization as a word that was gated to the data transceivers (including parity). The value obtained does not represent what is actually in memory as a result of the bus write (although this would be the typical result), but only what the K.pli transceivers placed on the backplane bus lines during the bus cycle.

Control Memory Cycles are initiated when SET CREQ is specified in Control Store BUS DEST <2:0>. During execution of this instruction BR ADR <10:8> determine the type of bus cycle to be performed, and BR ADR 11 controls the least significant address bit (Figure 7-10).

## 7.17 DATA BUS INTERFACE

The Data Bus Interface is similar to the Control Bus interface, except that it is associated with the Data Sequencer and is used to access the HSC Data Memory during data transfers. Data to be sent to the node on the CI is read from Data Memory, while data obtained from the node is written into Data Memory prior to being routed to the next process associated with the data transfer. The K.pli is capable of maintaining data rates up to 6.66 megabytes per second on this bus. For Data Bus timing information refer to Chapter 3.

### 7.17.1 Data Bus Data Transceivers

The Data Bus Data Transceivers are read or written by microcode, and the value obtained when reading the Data Transceivers has exactly the same format as that required to write them. When a Data Memory write is performed, the hardware provides an automatic wrap around so that the data and address contents may be read and verified. The value obtained does not represent what is actually in memory as a result of the bus write (although this would be the typical result), but only what the transceivers placed on the backplane bus lines during the bus cycle.

Data Bus cycles are initiated by either write loading (which initiates a write cycle) or reading (which initiates a read cycle) the Data Bus Data Transceivers.

The Data Bus Data Transceiver Drivers are controlled by the RD MODE signal from Data IOC Latch bit 3. If RD MODE H is low, these drivers will not drive the bus. If RD MODE H is high, these drivers will be gated onto the bus during each Data Bus cycle. This is true regardless of whether a cycle is initiated by a write load or a read of the data transceivers.

**NOTE**

**The term read refers to the PILA, not to Data Memory.**

### 7.17.2 Data Bus Address

This interface consists of 2917A-type latched bus transceivers, but it includes an up/down counter for the address. The Data Bus Address Transceivers and counter are written by microcode, and the transceivers may be read by microcode. Note that the result obtained from reading the Data Bus Address Transceivers does not have the same organization as that required when writing it (Figure 7-11). The value obtained from reading the transceivers is the address used in the last completed bus cycle.

The Data Bus address is 18 bits wide. The twelve least significant address bits are derived from a counter which is automatically incremented (or decremented) at the end of each Data Bus cycle. Normally, the address is written prior to beginning a block transfer, and it need not be rewritten while the block transfer is in progress. However, this counter will not cross 4 Kword boundaries.

## Figure 7-11 Data Bus Address Transceivers and Counter

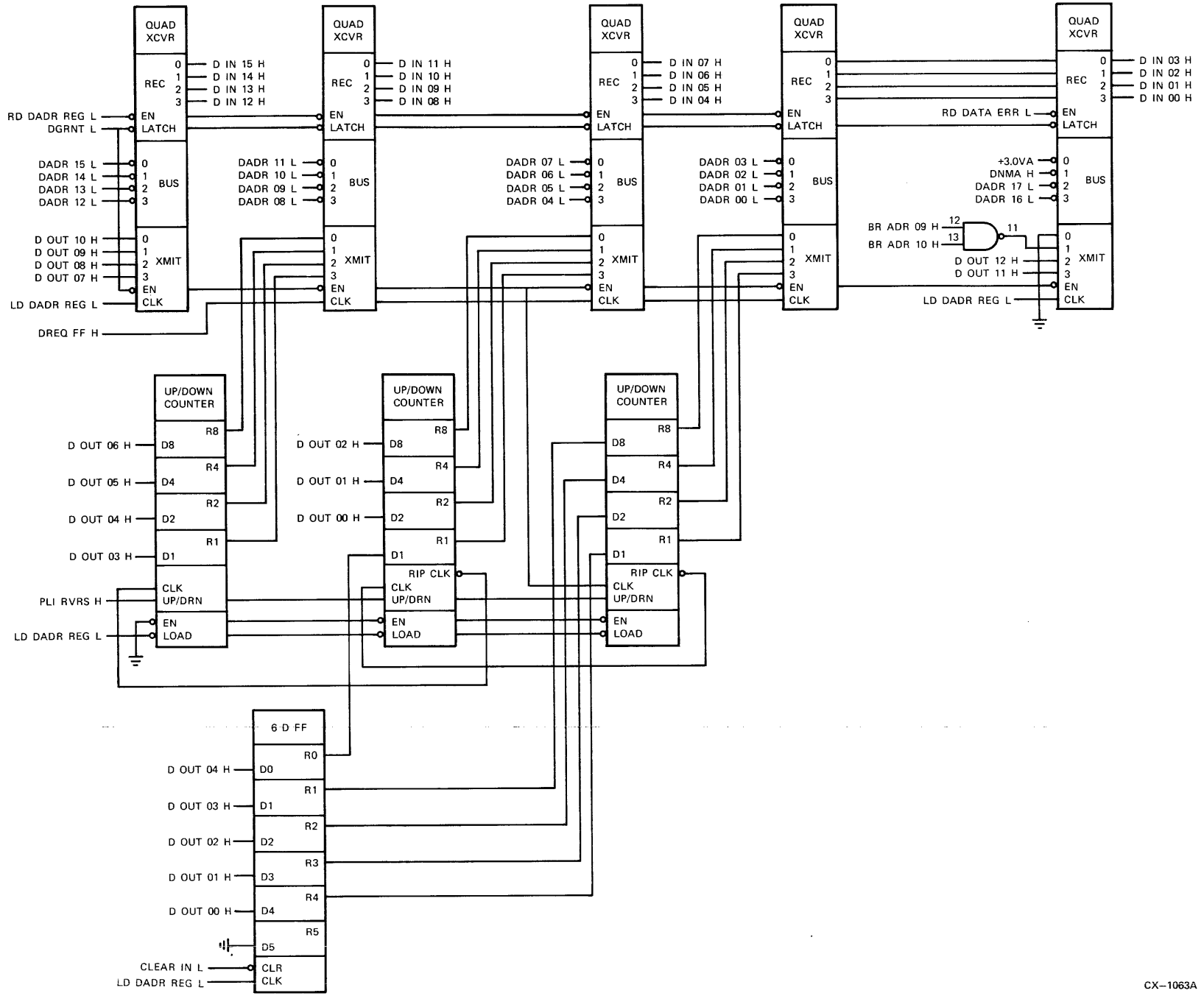
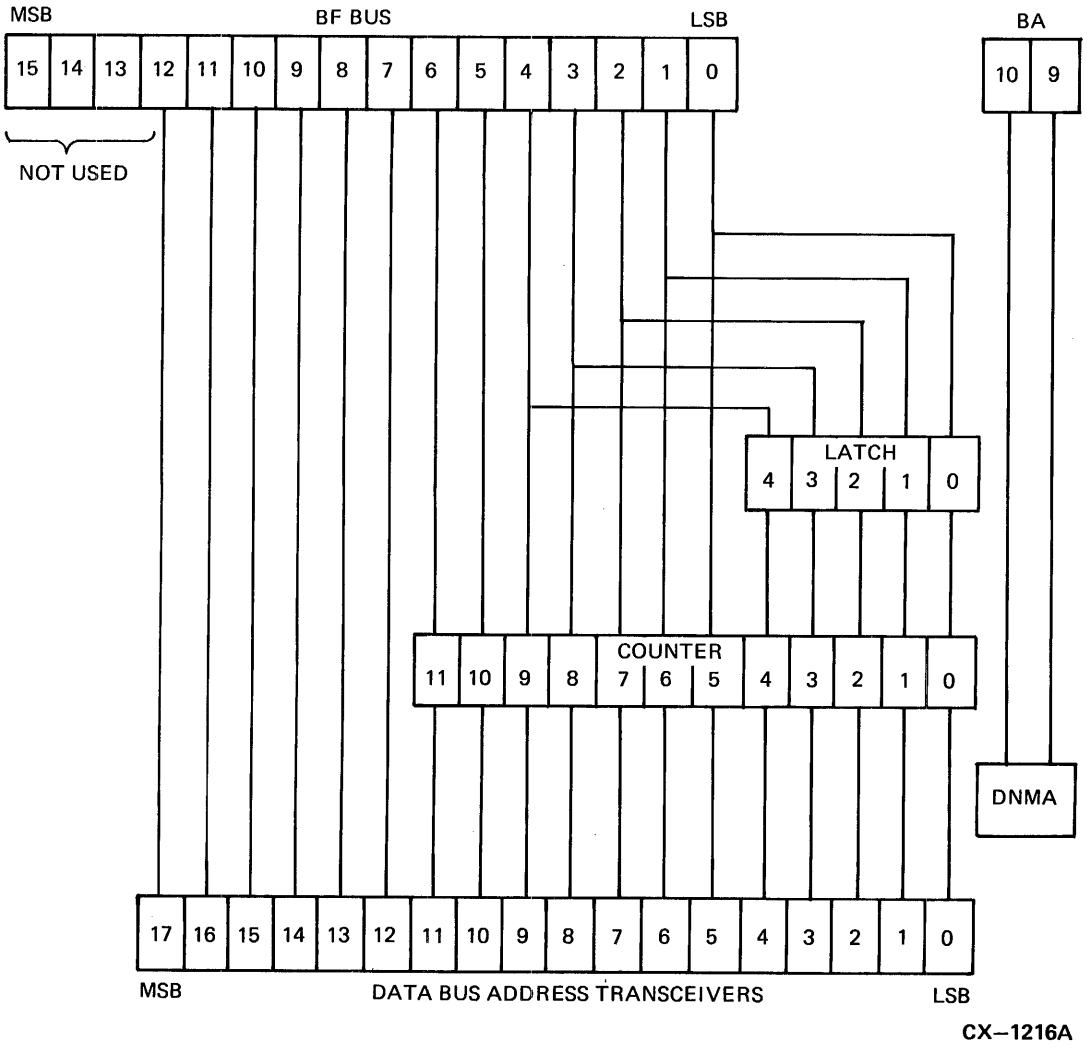


Figure 7-12 shows a simplified block diagram of how the Data Bus address is encoded. When the microcode asserts LD DADR REG, the following registers are written to:

- D OUT <04:00> are gated into the five bit latch
- D OUT <06:00> are gated into the counter bits <11:05>
- D OUT <12:07> are gated into Data Bus Address Transceivers
- The previous contents of the 5-bit latch are gated into counter bit <04:00>

**Figure 7-12 Generating the Data Bus Address**



Note that counter bits <04:00> are derived from the latch, not directly from the D OUT lines. In order to correctly write the entire address, LD DADR REG must be executed twice. The first write would place the desired contents of address <04:00> on D OUT <04:00>. The second write should place the desired contents of address <17:05> on D OUT <12:00>.

The procedure to start a data transfer is:

1. Write the least significant five address bits
2. Write the remainder of the address
3. Perform one dummy data read (if it is a memory read operation)
4. Move a data word, conditioned on completion of previous word
5. Repeat the above step until the transfer is complete

## 7.18 PLI INTERFACE

The PLI Interface interconnects the K.pli to the LINK and PILA which it controls. This interface provides paths between modules for:

- Data transfer
- Status transfer
- Control between K.pli, PILA, and LINK

This interface is capable of data rates up to 6.66 megabytes per second.

### 7.18.1 PLI Data Bus and PLI Control

The PLI Data Bus is bidirectional and consists of eight data lines and two parity lines. This bus is used to:

- Transfer data to and from the PILA buffers
- Receive status from the PILA and LINK
- Transfer commands to the PILA and LINK

The microcode determines how it is going to use the PLI Data Bus and sets the PLI LINK CONTROL <03:00> before issuing a read or a write to the PLI Data Bus.

The PLI Data Bus logic consists of a 16-bit PLI Data Transmit Register, a 16-bit Pli Data Receive Register, and the logic to:

- Transfer 16-bits from the PLI Data Transmit Register to the PILA over the 8-bit PLI Data Bus (This is accomplished by transferring two bytes of 8-bits each.)
- Transfer two 8-bit bytes from the PILA to the PLI Data Receive Register

The PLI Data Bus logic is located in the middle-right of Figure 7-2. The PLI Control logic is located in the lower-right corner of the block diagram.

The following three signals control how the data is transferred between the K.pli and the PILA (Figure 7-13).

- SEND TO PLI
- ONE BYTE
- PLI RVRS

## Figure 7-13 PLI Interface Registers and Control

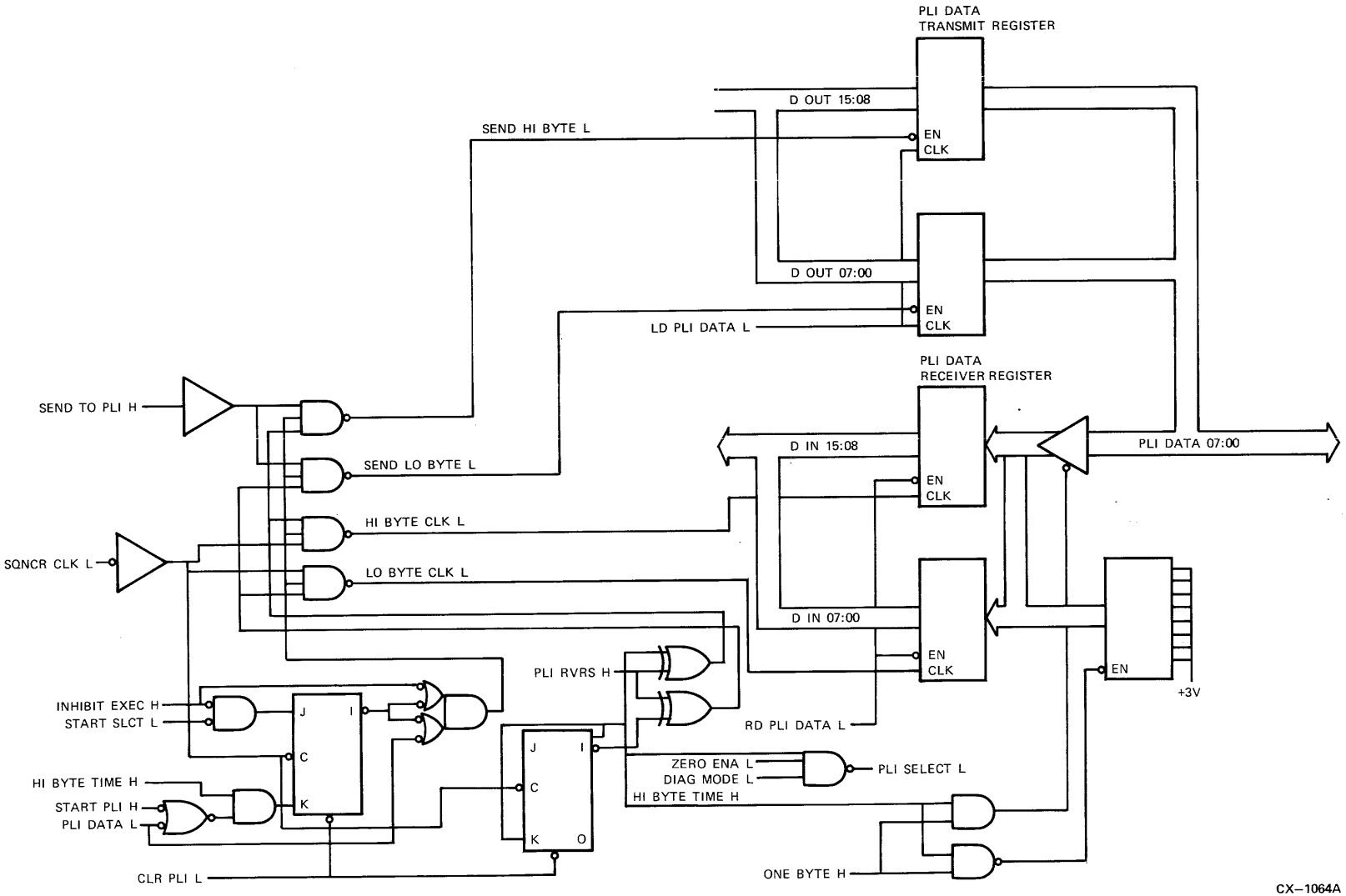


Table 7-6 describes what happens with the data for the different configurations of these three signals.

**Table 7-6 Control of Data Transfers Between the K.pli and the PILA**

| SEND TO PLI | ONE BYTE | PLI RVRS | Description |
|---|---|---|---|
| 0 | 0 | 0 | The RD PLI DATA signal initiates a two-byte transfer from the PILA to the K.pli. The first byte is gated into the low byte and the second byte is gated into the high byte of the 16-bit PLI Interface Receive Register. |
| 1 | 0 | 0 | The START PLI signal initiates a two-byte transfer from the K.pli to the PILA. The low byte (bits <07:00>) of the 16-bit PLI Interface Transmit Register is sent before the most significant byte (bits <15:08>). |
| 0 | 1 | 0 | The RD PLI DATA signal initiates a one-byte transfer from the PILA to the K.pli. The byte is gated into the low byte of the PLI Interface Receive Register. All zeros are gated into the high byte of the K.pli register. |
| 1 | 1 | 0 | The START PLI signal initiates a one-byte transfer from the K.pli to the PILA. The low byte of the PLI Interface Transmit Register is sent to the PILA. |
| 0 | 1 | 1 | The RD PLI DATA signal initiates a one-byte transfer from the PILA to the K.pli. The byte is gated into the high byte of the PLI Interface Receive Register. All zeros are gated into the low byte of the K.pli register. |
| 1 | 1 | 1 | The START PLI signal initiates a one-byte transfer from the K.pli to the PILA. The high byte of the PLI Interface Transmit Register K.pli register is sent to the PILA. |
| 0 | 0 | 1 | The RD PLI DATA signal initiates a two-byte transfer from the PILA to the K.pli. The first byte is gated into the high byte, and the second byte is gated into the low byte of the PLI Interface Receive Register. |
| 1 | 0 | 1 | The START PLI signal initiates a two-byte transfer from the K.pli to the PILA. The high byte of the PLI Interface Transmit Register is sent before the the low byte. |

The timing for double byte and single byte reads and writes are shown in the following figures:

- PLI Interface Double Byte Read (Figure 7-14)
- PLI Interface Single Byte Read (Figure 7-15)

- PLI Interface Double Byte Write (Figure 7-16)
- PLI Interface Single Byte Read (Figure 7-17)

### 7.18.2 PLI Status

In order to perform its job, the K.pli must check the status in the PILA and LINK. This is accomplished in one of two ways:

- Over the PLI Data Bus using one of the following commands generated by the Control Register:
  - Read receive status
  - Read transmitter status

  These two commands are described in Chapter 6.
- Through the Test Select multiplexers described in this section.

The Test Select multiplexer has 19 PLI status indicator inputs (Figure 7-18). Three status indicators are directly selected and generated from the Control Register (Section 7.19). The other 16 status indicators are multiplexed through multiplexers, PILA Status Select 0 and 1. The status pair to be selected is determined by:

- SEND TO PLI
- PLI SEL 1
- PLI SEL 0

Table 7-7 describes the function of these status indicators.

**Table 7-7 Directly Connected Status Indicators to the Test Select Multiplexers**

| Signal | Description |
|---|---|
| Enable ARB | Indicates whether the LINK Arbitration logic is enabled or disabled. |
| PLI RECEIVER STATUS 1 | This line is a spare input to the PLI Test Select multiplexer. |
| FIRST BUFFER | Indicates which buffer was filled first when both RECEIVE BUFFER A FULL and RECEIVE BUFFER B FULL are both set or both cleared, where:<br><br>0 = Buffer B was filled first<br>1 = Buffer A was filled first |
| BUFFER A BUS | Indicates the CI path that carried the last received packet in Receiver Buffer A, where:<br><br>0 = CI path A<br>1 = CI path B |
| XMTR BUSY | The LINK asserts this line when a transmit sequence is in progress or when XMTR ATTN is asserted. |
| ACK | The LINK asserts this line when it receives an ACK packet from the destination node. |

Table 7-7 (Cont.) Directly Connected Status Indicators to the Test Select Multiplexers

| Signal | Description |
|---|---|
| DEST BUSY | The LINK asserts this line when it receives a NACK packet from the destination node. |
| ARBITRATE | The LINK asserts this line when the arbitration count has expired after a transmit command. |
| RCVR A EN | The LINK asserts this line when receiver path A is enabled. |
| RCVR B EN | The LINK asserts this line when receiver path B is enabled. |
| BUFFER B BUS | Indicates the CI path that carried the last received packet in Receiver Buffer B on the PILA, where:<br><br>0 = CI path A<br>1 = CI path B |
| XMIT ABORT | The LINK asserts this line when a transmission is aborted by the abort transmission command. |
| CRC ERROR | The LINK asserts this line when it detects a CRC error on the received message or data packet. |
| CDA | Indicates the state of the carrier on CI path A, where:<br><br>0 = Carrier A off<br>1 = Carrier A on |
| CDB | Indicates the state of the carrier on CI path B, where:<br><br>0 = Carrier B off<br>1 = Carrier B on |
| XBUF PE | The LINK asserts this line when it detects a parity error in the transmit data during a transmission. A parity error asserts XMTR ATTN and aborts the transmission. |
| PLI XMTR ATTENTION | The LINK asserts this line when there is a response to a transmit command. The response may be caused by an ACK, NAK, transmit parity error, or transmit abort. |
| PLI RCVR BUFFER A FULL | The PILA asserts this line when Receive Buffer A, in the PILA, contains a valid block of data or a packet (in the PILA). |
| PLI RCVR BUFFER B FULL | The PILA asserts this line when Receive Buffer B, in the PILA, contains a valid block of data, or a packet. |

Figure 7-14 PLI Interface Double Byte Read Timing Diagram

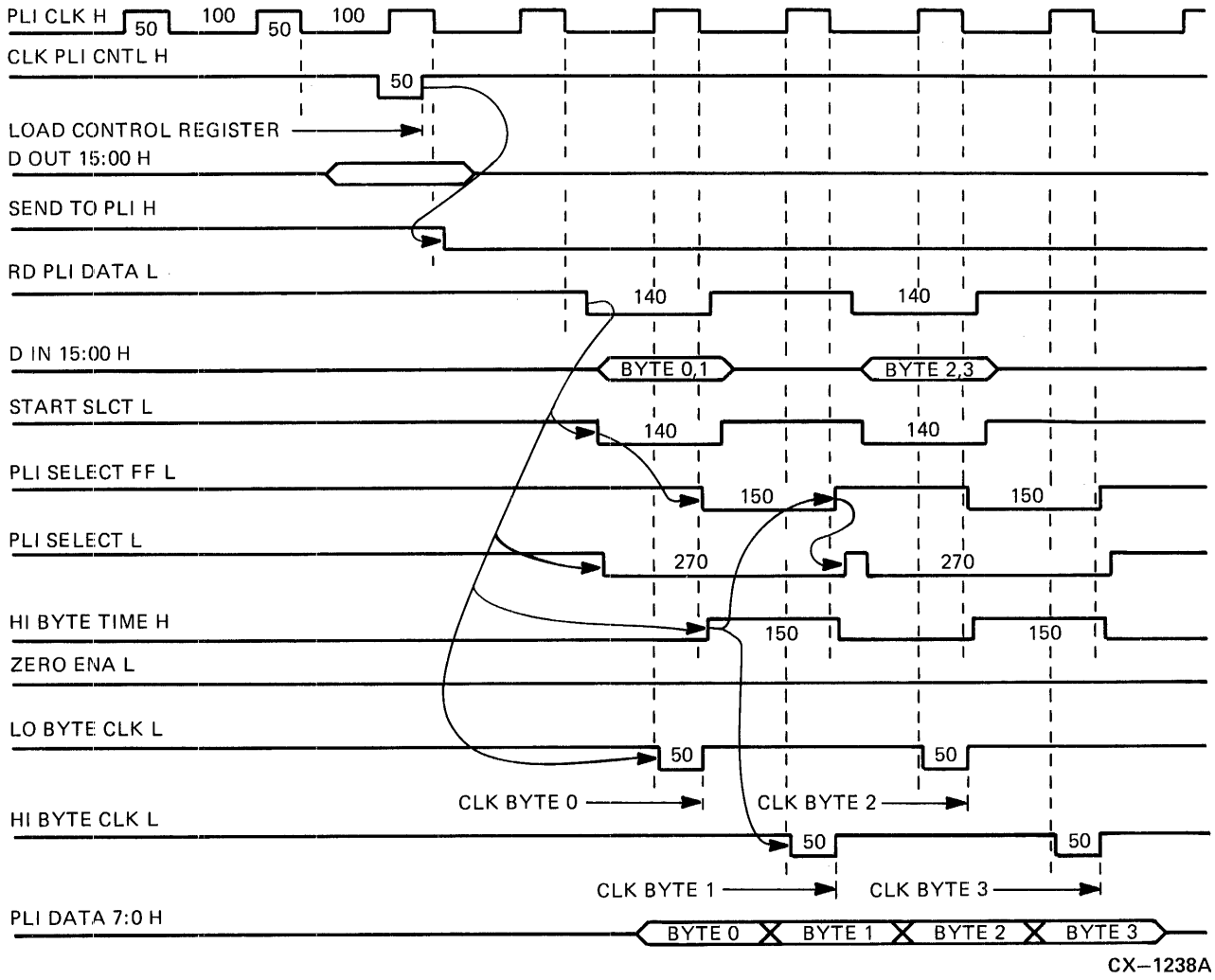


Figure 7-15 PLI Interface Single Byte Read Timing Diagram

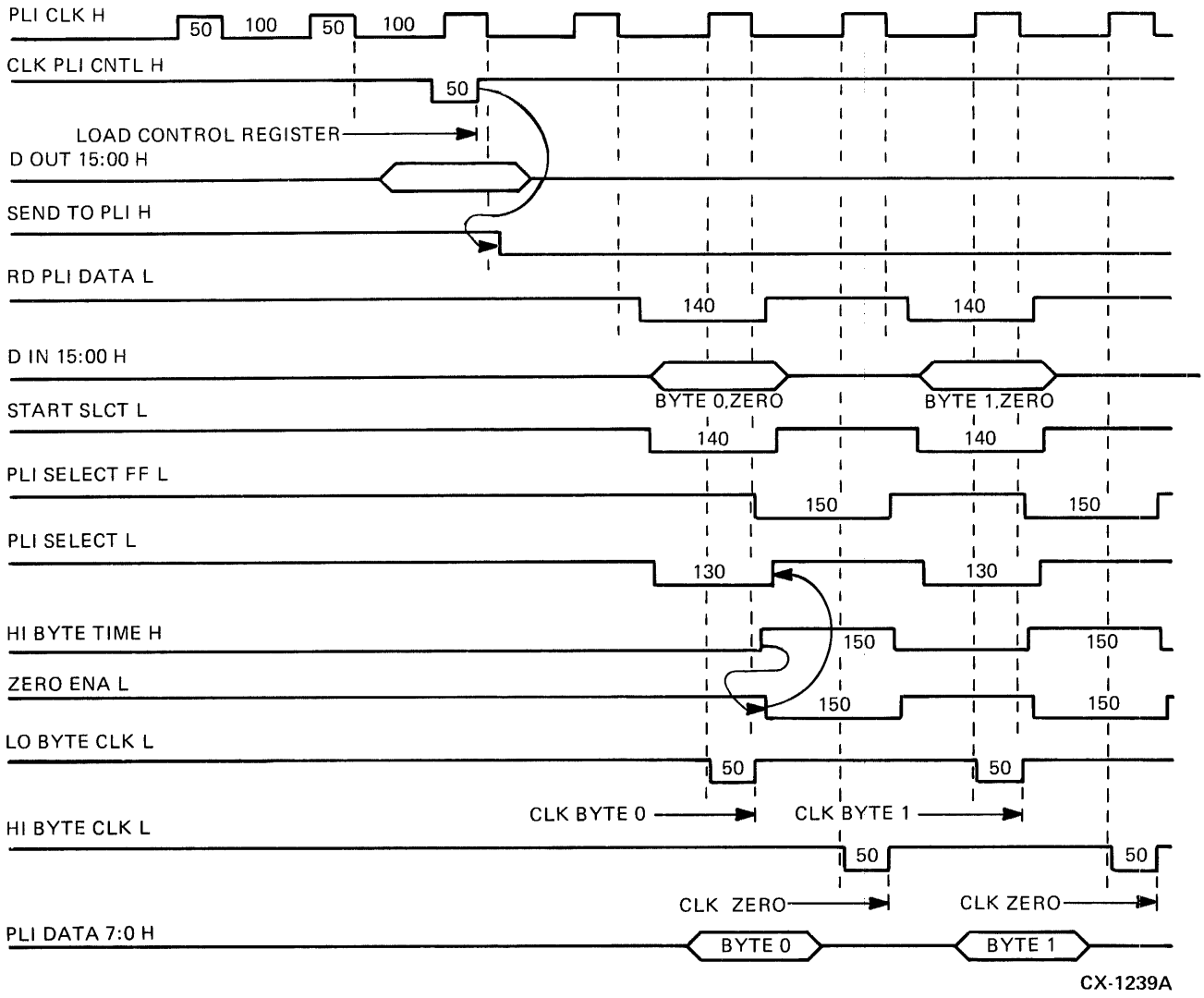


Figure 7-16 PLI Interface Double Byte Write Timing Diagram

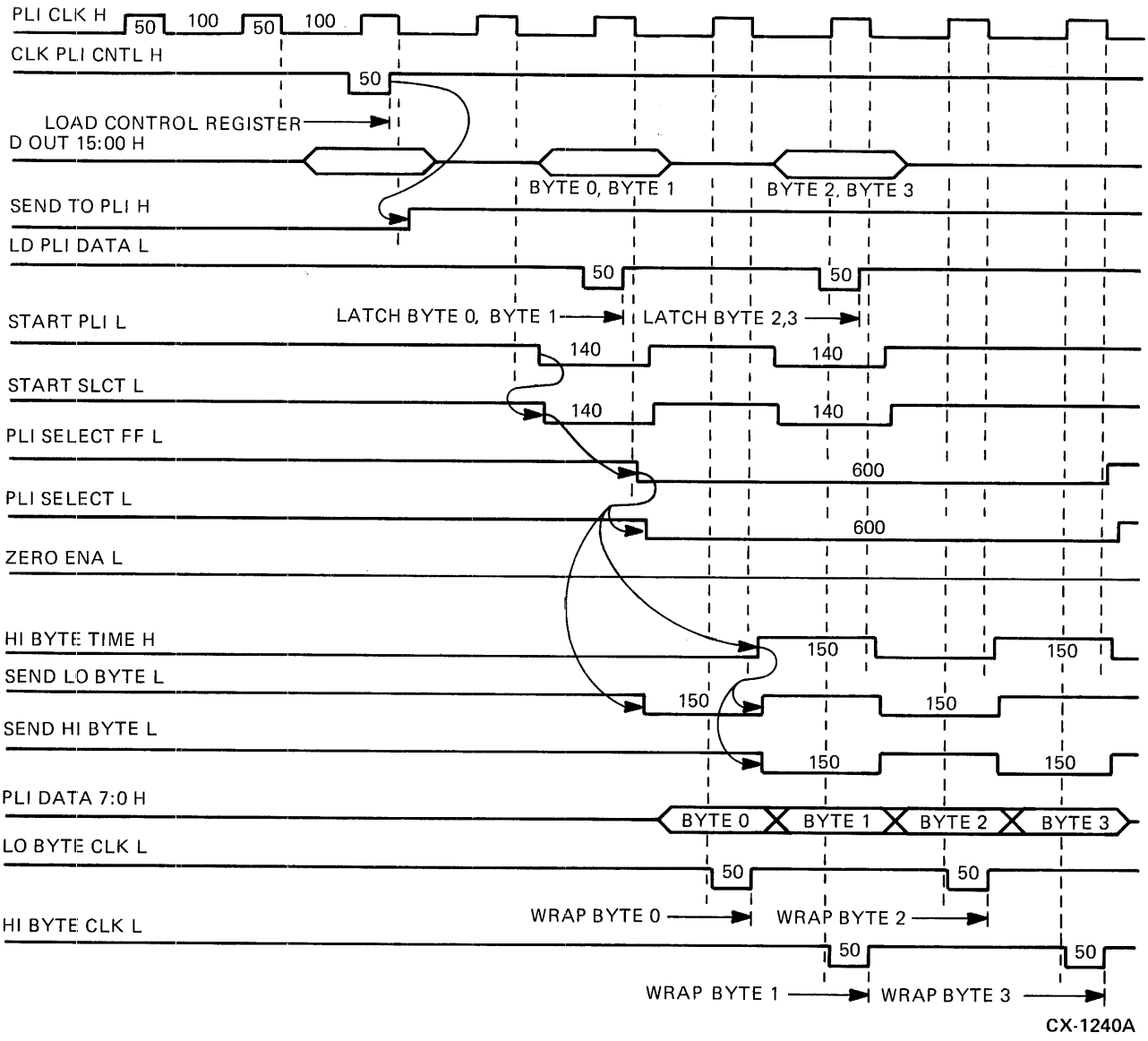


Figure 7-17 PLI Interface Single Byte Write Timing Diagram

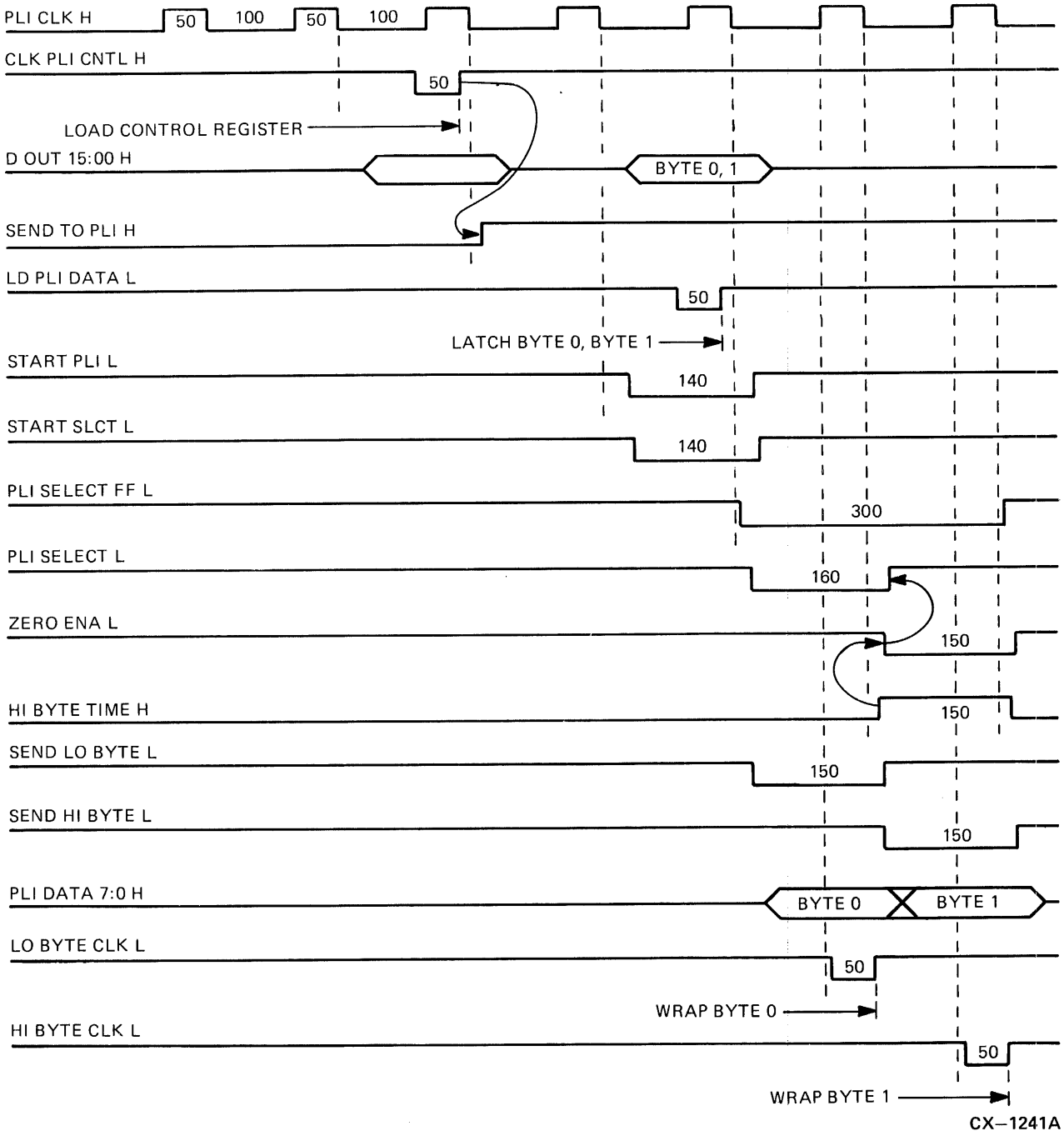
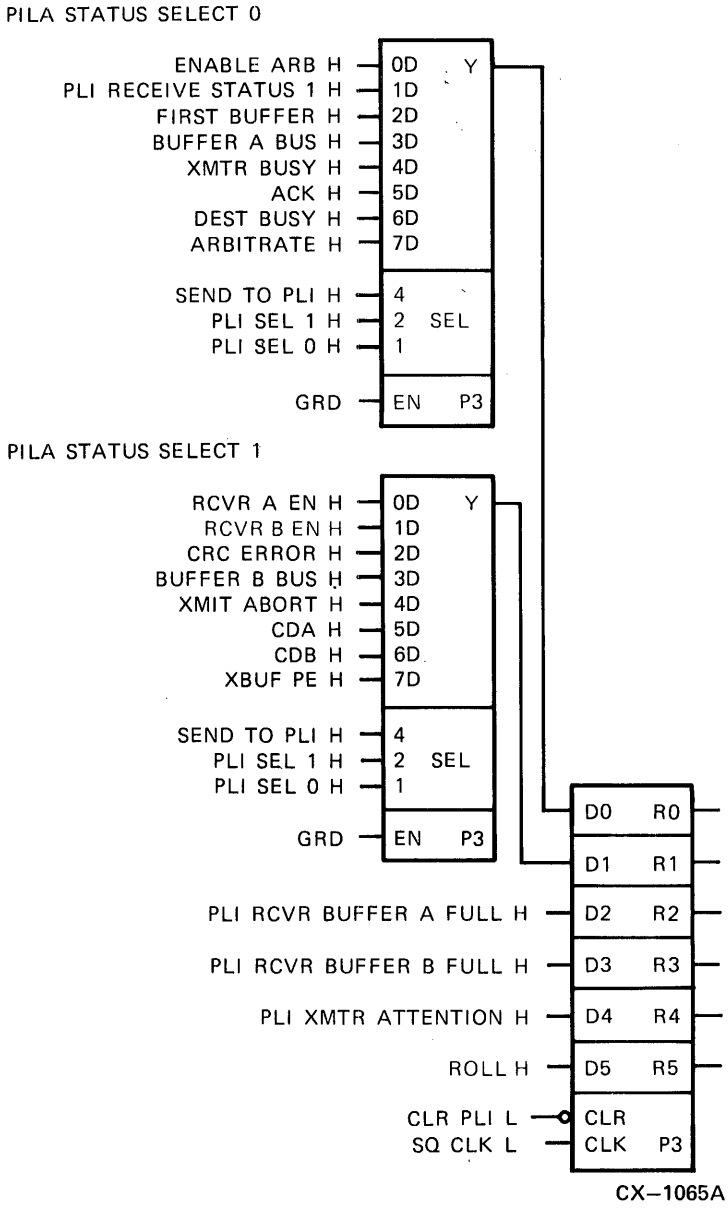


**Figure 7-18 PLI Portion of the Test Select Multiplexer**



### 7.18.3 PLI Interface Control

The PLI Interface is controlled by seven signals that are outputs from various blocks shown in Figure 7-2. These seven signals:

- Determine what is gated onto the PLI Data Bus
  - Data to or from a PILA buffer
  - Status from the LINK or PILA
  - Commands to the LINK or PILA
- Determine the direction of transfer on the PLI Data Bus
- Determine the correct time to clock the PLI Data Bus into the receivers
- Initialize the LINK and PILA

Table 7-8 describes the seven signals that control the PLI Interface.

**Table 7-8 PLI Interface Control Signals**

| Signal | Description |
|---|---|
| PLI LINK CONTROL <3:0> | These four signals control the LINK and PILA, including their activity on the PLI Interface. They are continuously driven but are acted upon only when PLI SELECT is asserted. Refer to Chapters 5 and 6 to learn how the modules use these four signals. These signals are shown on the block diagram (Figure 7-2) as an output of the Control Register. |
| SEND TO PLI | This signal determines the direction of data transfers on the PLI Data Bus. It also determines whether the PLI Receive Status or PLI Transmit Status lines are selected for microcode testing. This signal is shown on the block diagram as an output of the Control Register. |
| PLI SELECT | This signal is asserted for each data byte transferred. It is negated if the K.pli is in diagnostic mode. The signal is not directly controlled by microcode. It is asserted as part of the one- or two-byte sequence following a LD PLI DATA or RD PLI DATA. Except for INITIALIZE, the LINK and PILA will respond to the PLI Interface signals only when PLI SELECT is asserted. |
| PLI INITIALIZE | This signal forces the initialization of the LINK and PILA. It clears the error indicators, buffer status, interface status, and disables the receivers. This signal is asserted following a clear from the P.ioc or a clear from the Data or Control Sequencer. PLI INITIALIZE is shown in the upper-left corner of the block diagram (Figure 7-2) as an output from the Reset Control Logic. |

## 7.19 CONTROL REGISTER

The Control Register contains:

- PLI Interface Control
- P.ioc Reset
- Control Store Parity Test Logic
- Data Parity Test Logic

The Control Register is located in the lower right of Figure 7-2. Each control bit is described in Table 7-9.

**Table 7-9 Control Register Bit Definition**

| Bit | Definition |
|---|---|
| PLI LINK CONTROL <3:0> | These four bits control the LINK and PILA, including their activity on the PLI Interface. These four lines are continuously driven but are acted upon only when PLI SELECT is asserted. Refer to Chapters 5 and 6 to learn how the modules use these four signals. |
| SEND TO PLI | This bit determines the direction of data transfers on the PLI Data Bus. It also determines whether the PLI Receive Status or PLI Transmit Status lines are selected for microcode testing. |
| PLI SEL 0/1 | These two bits, with SEND TO PLI, select two of the sixteen PLI Status lines to place on the PILA Status Select 0, 1 lines for selection by the Test Select multiplexer logic. |
| DIAG MODE | This bit, when set, negates the PLI SELECT line (Table 7-8). It also permits writing of bits <15:08> of this register. This bit must be set in order to write bits <15:08> successfully. When cleared, this bit forcibly clears bits <15:08>. Diagnostic mode is not to be confused with the maintenance or diagnostic modes of the LINK or PILA, nor with software or microcode diagnostic modes. Setting this bit affects internal K.pli hardware only. |
| INSTR PAR TEST | This bit, when set, inverts the sensing of the Control Store parity bit, thus forcing instruction parity errors. |
| LO PAR TEST | This bit, when set, inverts the sensing of the low byte data parity test logic. |
| HI PAR TEST | This bit, when set, inverts the sensing of the high byte data parity test logic. |
| INSTR STOP | This bit, when set, stops all clocks in the K.pli. |
| HOST CLR | These two bits, if both are set, forcibly clear the P.ioc. |

## 7.20 DATA PARITY GENERATION AND CHECK LOGIC

The Data Parity Generation and Check Logic generates and checks parity for the two bytes on D OUT and the two bytes on D IN. The logic generates LO PARITY OUT and HI PARITY OUT that go along with the D OUT signals. The PAR ERROR is generated when either D IN byte fails parity. This logic is located in the right center of Figure 7-2.

## 7.21 INSTRUCTION PARITY CHECKER

The Instruction Parity Checker checks for odd parity over the 48 bits of the instruction from the Control Store. This parity bit is gated to the Test Select multiplexer. The odd parity bit for the 48 bits of the instruction word is bit 12. The Instruction Parity Checker is located at the top of the Control Store in Figure 7-2.

If a parity error is detected in the Control Store and the K.pli is in diagnostic mode, execution of that instruction is inhibited, and the sequencer active at that time is forced to a recovery routine. This trap forces the Program Address Register to $7776_8$ for the Control Sequencer or $7777_8$ for the Data Sequencer (the last two words of the Control Store), where branches to the appropriate service routine are located. Asserting the vector address is accomplished by PAR ERR INH inhibiting the output of the sequencer to the Program Address Register (Figure 7-19). This gates all ones into the Program Address Register inputs. However, if the parity error was from the Control Sequencer, SEQ ADR 00 is forced low by CNTL PROC and PAR ERR INH. This scheme gives us the two parity error trap vectors.

If a Control Store parity error is detected when K.pli is in normal mode, the internal clock will be stopped, resulting in a complete halt of all K.pli operations. Recovery will occur only when the P.ioc initializes the module.

## 7.22 DESTINATION DECODER

The Destination Decoder (DEST DECODER) decodes bits <27:25> from Control Store to write registers (for example, LD RAM DATA) or start timing (for example, SET CREQ). Signals ENA DATA and ENA CNTL control which destination decoder is selected. The Dest Decoder is located on the output of the Control Store in Figure 7-2.

## 7.23 IOC LATCHES

Each microcode instruction can set, or clear, one of the IOC Latches. These latches may not be set or cleared if the literal field, to the ALU, is being used. Modification of these latches is specified if IOC ENABLE is set. Table 7-10 describes each bit in the IOC Latches.

**Table 7-10 IOC Latch Bit Description**

| Bit | Definition |
|---|---|
| COMMON IOC 7/6 | These two bits are set or cleared by either sequencer. These bits are two of the inputs to the Test Select multiplexers of the two sequencers. They are used as microcode control and status flags or for communication between the two sequencers. |
| ONE BYTE | This bit is set or cleared by either sequencer. When set, a write or read of the PLI Data Register results in a single byte transfer on the PLI Data Bus. When this bit is clear, a write or read of the PLI Data Register results in a two byte transfer across the PLI Data bus. |

Table 7-10 (Cont.) IOC Latch Bit Description

| Bit | Definition |
|---|---|
| EN STATUS | This bit is set or cleared by the Control Sequencer only. It determines which of the two registers is written to by the Control Sequencer when BUS DEST decodes a seven. If this bit is clear, the Status Bus Register is written. If it is set, the Scratchpad Address register will be written. Operation of the Data Sequencer is not affected by this bit. |
| ROLL | This bit is set or cleared by the Control Sequencer only. When clear, both sequencers operate normally. When set, the Data Sequencer is prevented from executing all ALU, I/O, and branch operations. The Control Sequencer forces the Data Sequencer to fetch and check all instructions of the 4 Kword Control Store for proper parity. |
| LTS GRN | This bit is set or cleared by the Control Sequencer only. This bit controls the two LEDs. When set, the green LED is on and the red LED is off. When this bit is clear, the red LED is on and the green LED is off. |
| CNTL PROC FLAG | This bit is set or cleared by the Control Sequencer only, and it is inverted and used as an input to the Data Test Select multiplexer. This bit is used for microcode communication from the Control Sequencer to the Data Sequencer. |
| RD MODE | This bit is set or cleared by the Data Sequencer only. It controls the drivers of the K.pli Data Bus Data Transceivers. If set, this bit permits the gating of the drivers onto the Data Bus during any bus cycle. If clear, it prevents the gating of the drivers onto the Data Bus. |
| PLI RVRS | This bit is set or cleared by the Data Sequencer only. It controls the count direction of the Data Bus Address Counter and the odd/even byte sequence on the PLI Data Bus. When this bit is clear, the Data Bus Address Counter will increment, and the PLI Interface logic will transfer the low byte first. When it is set, the counter will decrement, and the PLI interface logic will transfer the high byte first. |
| DATA PROC FLAG | This bit is set or cleared by the Data Sequencer only, and it is an input to the Control Test Select multiplexer. This bit is used for microcode communication from the Data Sequencer to the Control Sequencer. |

## 7.24 SOURCE DECODER

The Source Decoder decodes bits <19:18> from Control Store to read from the different registers. Each sequencer has its own set of decoders (ENA CNTL or ENA DATA). The Source Decoder is located on the output of the Control Store in Figure 7-2.

## 7.25 MICROINSTRUCTION DETAILED DESCRIPTION

This section describes the organization and purposes of the fields within the microinstruction word. The microinstruction word is 48 bits wide. The microinstruction word is divided into sixteen fields, each of which controls different portions of the K.pli hardware. Figure 7-20 shows the fields and their positions within the microinstruction word. The following sections describe each of these fields.

Figure 7-19 Generation of Parity Error Vectors

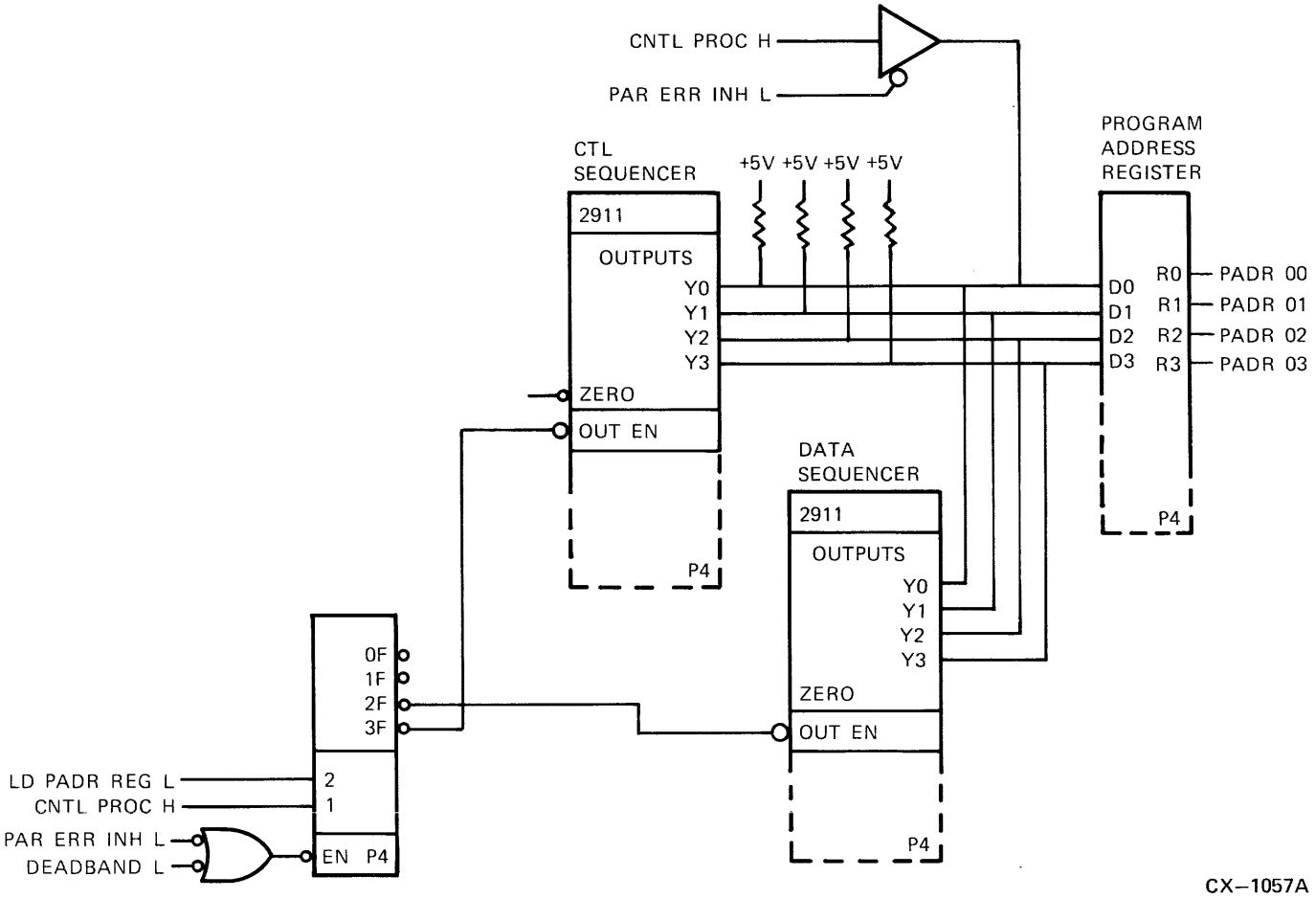
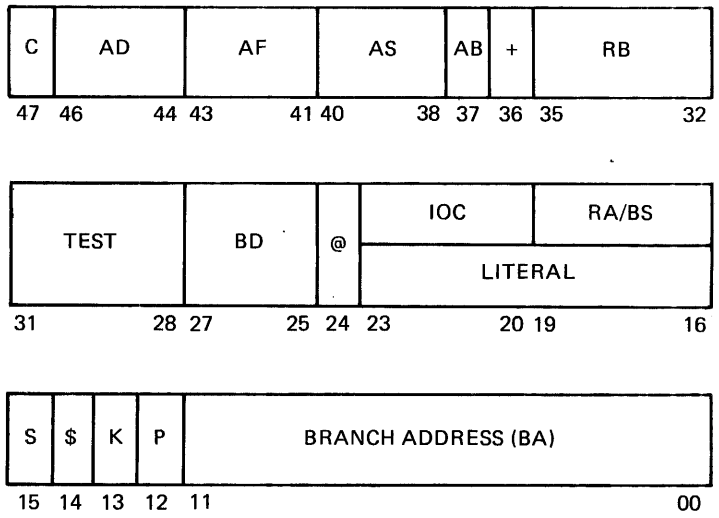


Figure 7-20 Microinstruction Word Description




### 7.25.1 C Field

The C field is the carry into the ALU for the current microinstruction. It steers the literal to the high byte for AF fields 3 through 7.

### 7.25.2 AD Field

The AD field, ALU destination, determines the routing of the ALU output data internal to the 2901. Table 7-11 describes the decoding of the three bits in the AD field.

Table 7-11 Decoding of Bits in the AD Field

| Decode | Description |
|---|---|
| 0 | ALU to Q Register |
| 1 | No destination (sets conditions only) |
| 2 | ALU to RB, output RA (RB if AS equals 5 or 7) |
| 3 | ALU to RB |
| 4 | ALU/2 to RB, Q/2 to Q (right shift) |
| 5 | ALU/2 to RB (right shift) |
| 6 | ALU*2 to RB, Q*2 to Q (left shift) |
| 7 | ALU*2 to RB (left shift) |

### 7.25.3 AF Field

The AF field, ALU function, determines the operation to be performed by the ALU. Table 7-12 describes the decoding of the three bits in the AF field. The R and S are the inputs to the ALU (Figure 7-4).

Table 7-12 Decoding of Bits in the AF Field

| Decode | Mnemonic | ALU Function |
|---|---|---|
| 0 | ADD | R plus S |
| 1 | SUBR | S minus R |
| 2 | SUBS | R minus S |
| 3 | OR | R OR S |
| 4 | AND | R AND S |
| 5 | NOTRS | (NOT R) AND S |
| 6 | XOR | R XOR S |
| 7 | XNOR | R EX-NOR S |

### 7.25.4 AS Field

The AS field, ALU source, selects two of the five possible sources for the ALU. Table 7-13 shows the combination of the three bits in the AS field. The R, S, A, D, B, 0, and Q are the inputs and outputs of the ALU Data Source Selector in the ALU (Figure 7-4).

Table 7-13 Decoding of Bits in the AS Field

| Decode | R | S |
|---|---|---|
| 0 | A | Q |
| 1 | A | B |
| 2 | 0 | Q |
| 3 | 0 | B |
| 4 | 0 | A |
| 5 | D | A |
| 6 | D | Q |
| 7 | D | 0 |

NOTE:

A and B are the contents of the internal registers selected by fields RA and RB, respectively

D is the data input to the ALU from outside the 2901

Q is the contents of the internal Q Register

0 is all zeros for that input.

### 7.25.5 AB Field

The AB field, ALU to BUS, determines if the D OUT lines will be driven by the 2901 ALU or if the D IN contents will be transferred to the D OUT lines, bypassing the ALU. It also enables or disables the BS field decoders. Table 7-14 describes the decoding of the bit in the AB Field.

Table 7-14 Decode of the Bit in the AB Field

| Decode | Description |
|---|---|
| 0 | D IN Bus source to D OUT |
| 1 | ALU to D OUT |

### 7.25.6 + Field

The + field, EXEC IF COND TRUE, is one bit long. This bit is combined with the following signals to generate INHIBIT EXEC (inhibit execute).

- TST COND TR (test condition true)
- COND EXC (conditional execute) ($ field)

INHIBIT EXEC prevents:

- Any ALU operation of each instruction for which it is active
- The IOC Latches from being enabled
- The destination decoders from being enabled

Table 7-15 describes the decoding of the bit in the + field along with EXEC IF COND TRUE and TST COND TR to generate INHIBIT EXEC.

**Table 7-15 Decode of the Bit in the + Field**

| COND EXC | EXEC IF COND TRUE | TST COND TR | INHIBIT EXEC | Condition |
|---|---|---|---|---|
| 0 | X | X | 0 | Execute all operations |
| 1 | 0 | 0 | 0 | Execute all operations |
| 1 | 0 | 1 | 1 | Stop ALU, IOC, destination operations |
| 1 | 1 | 0 | 1 | Stop ALU, IOC, destination operations |
| 1 | 1 | 1 | 0 | Execute all operations |

X means the state of the signal does not matter

### 7.25.7 RB Field

The RB field, B Port Register, selects one of the sixteen internal 2901 registers to be read and/or written by the 2901 ALU. If a value of $5_8$ or $7_8$ is in the the AS field, the A Port Register field (RA field) is forced to the same value as the RB field.

### 7.25.8 TEST Field

The TEST field, TEST CONDITION <03:00>, selects one of sixteen signals for the conditional operations of the sequencers. The hardware includes two separate selectors, one for each of the two sequencers. Table 7-16 shows the test conditions for the four bits of the TEST field.

Table 7-16 Decode of the Bits in the TEST Field

| Decode | Control Sequencer | Data Sequencer |
|---|---|---|
| 0 | 1 (True) always | 1 (True) always |
| 1 | Last Control Sequencer ALU result = 0 | Last Data Sequencer ALU result = 0 |
| 2 | Last Control Sequencer ALU carry out | Last Last Data Sequencer ALU carry out |
| 3 | Last Control Sequencer ALU MSB | Last Last Data Sequencer ALU MSB |
| 4 | Last Control Sequencer D OUT LSB | Last Last Data Sequencer D OUT LSB |
| 5 | Data Sequencer flag | Control Sequencer flag, inverted |
| 6 | Common Sequencer Flag 6 | Common Sequencer Flag 6 |
| 7 | Common Sequencer Flag 7 | Common Sequencer Flag 7 |
| 8 | Control Bus Error * | Data Bus Error * |
| 9 | Not Used | Not Used |
| 10 | PLI Test 0 ** | PLI Test 0 ** |
| 11 | PLI Test 1 ** | PLI Test 1 ** |
| 12 | Rcvr Buffer A Full | Rcvr Buffer A Full |
| 13 | Rcvr Buffer B Full | Rcvr Buffer B Full |
| 14 | Xmtr Attention | Xmtr Attention |
| 15 | Control Bus Request Active | Data Bus Request Active |

* Inclusive OR summation. Refer to Sections 7.17 and 7.16.

** Refer to Section 7.18.2.

### 7.25.9 BD Field

The BD field, bus destination, selects the register that will be written with the data that has been placed on the D OUT lines. Table 7-17 shows the decode of the three bits.

Table 7-17 Decode of the Bits in the BD Field

| Decode | Control Sequencer | Data Sequencer |
|---|---|---|
| 0 | NOP | NOP |
| 1 | Control Bus Data Xcvrs | Data Bus Data Xcvrs |
| 2 | Control Bus Address Xcvrs | Data Bus Address Xcvrs and Counter |
| 3 | Scratchpad Data | Scratchpad Data |
| 4 | PLI Bus Register | PLI Bus Register |
| 5 | Program Address Register | Program Address Register |
| 6 | Control Register | Control Register |
| 7 | IOC 3 SET; Scratchpad Adrs | Scratchpad Adrs |
| 7 | IOC 3 CLR; Status Bus Register | Scratchpad Adrs |

### 7.25.10 @ Field

The @ field, IOC ENABLE, permits the IOC field to modify the IOC Latches when set. This bit is not set when using the LITERAL field.

The IOC field has dual functions: it can be gated to the IOC Latches when the @ field is set, or the field can be the literal input to the ALU.

### 7.25.11 IOC Field

The IOC field, input/output control, if enabled by IOC ENABLE(@), modifies the IOC Latches. Bits <22:20> select the desired latch, and bit 23 sets or clears that latch. Table 7-18 describes the decoding of the bits in the IOC field.

Table 7-18 Decode of the Bits in the IOC field

| Decode | Control Sequencer | Data Sequencer |
|---|---|---|
| 0 | CNTL PROC FLAG | DATA PROC FLAG |
| 1 | LTS GRN | PLI RVRS |
| 2 | ROLL | Not Used |
| 3 | EN STATUS | RD MODE |
| 4 | Not Used | Not Used |
| 5 | ONE BYTE | ONE BYTE |
| 6 | COMMON IOC 6 | COMMON IOC 6 |
| 7 | COMMON IOC 7 | COMMON IOC 7 |

### 7.25.12 RA Field

The RA field, A Port Register, selects one of the sixteen internal 2901 registers to be operated on by the ALU. The RA field is a shared field and is only enabled if the AS field does not equal 5 or 7 and bit AB does not specify the LITERAL field.

### 7.25.13 BS Field

The BS field, bus source, selects one of the eight K.pli registers which are external to the ALU. The contents of the selected register are enabled onto the D IN lines, routed through the input multiplexer, and presented to the D inputs of the 2901 ALU. Table 7-19 describes the decoding of the bits in the BS field.

The BS field is a shared field and is only enabled if the AS field does not equal 5 or 7 and bit AB does not specify the LITERAL field.

**Table 7-19 Decode of the Bits in the BS field**

| Decode | Control Sequencer | Data Sequencer |
|---|---|---|
| 0 | NOP | NOP |
| 1 | Control Bus Data Xcvrs | Data Bus Data Xcvrs |
| 2 | Control Bus Address Xcvrs | Data Bus Address Xcvrs |
| 3 | Scratchpad Data | Scratchpad Data |
| 4 | PLI Bus Register | PLI Bus Register |
| 5 | Control Bus Errors | Data Bus Errors |
| 6 | Not Used | Not Used |
| 7 | Not Used | Not Used |

### 7.25.14 S Field

The S field, test sense, is XORed with the output of the Control and Data Test Select multiplexer to generate TST COND TR (test condition true). TST COND TR is then combined with branch address equal to $7777_8$ and JUMP CNTL 1 to determine whether the sequencer will output the Program Counter, the Stack, or the Branch Address. Table 7-20 describes what operation is performed in the sequencer depending on the selected test condition and the TST COND TR signal.

Table 7-20 Decode of the Selected Test Condition and TST COND TR

| Select Test Condition | TEST FOR TRUE | TST COND TR | Operation |
|---|---|---|---|
| 0 | 0 | 1 | Branch, Call, or Return |
| 0 | 1 | 0 | Increment Program Counter |
| 1 | 0 | 0 | Increment Program Counter |
| 1 | 1 | 1 | Branch, Call, or Return |

### 7.25.15 $ Field

The $ field, conditional execute, enables all instructions when clear. When set, this bit enables the conditional execution of the IOC, bus destination, and ALU instructions. Refer to Section 7.25.6 for a description of how this bit and TST COND TR and EXEC IF COND TRUE work together.

### 7.25.16 K Field

The K field, jump control, is logically combined with TST COND TR and BR ADR <11:00> equal to all ones ($7777_8$) to control the next operation of the 2911 Sequencers. (Refer to Section 7.25.14.) This combination controls the operation of the internal stack, and it determines whether the sequencer will output the:

- Program Counter
- Stack contents
- Branch address field

Table 7-21 describes the decode of the input signals for the sequencer operation.

Table 7-21 Decode of Input Signals for Sequencer Operation

| Branch Address | JUMP CNT 1 | TST COND TR | STACK | Y | Operation |
|---|---|---|---|---|---|
| *** | 0 | 0 | HOLD | PC | Increment |
| *** | 1 | 0 | HOLD | PC | Increment |
| $7777_8$ | 0 | 0 | HOLD | PC | Increment |
| $7777_8$ | 1 | 0 | HOLD | PC | Increment |
| *** | 0 | 1 | HOLD | BA | Branch |
| *** | 1 | 1 | PUSH | BA | Call |
| $7777_8$ | 0 | 1 | HOLD | BA | Branch |
| $7777_8$ | 1 | 1 | POP | STK | Return |

*** can be anything except $7777_8$

### 7.25.17 P Field

The P field, instruction parity, is the single parity bit for the 48-bit microinstruction word. This bit is calculated odd parity for the other 47 bits. If the number of 1 bits in the other 47 bits is even, then this bit is a 1, making the 48-bit microinstruction word odd parity. If the number of 1 bits in the other 47 bits is odd, then this bit is a 0 making the 48-bit microinstruction word odd parity.

### 7.25.18 BA Field

The BA field, branch address, provides the instruction address for conditional and unconditional branches and subroutine calls. Also, during the execution of certain instructions, this field provides supplemental data and control information. Refer to Section 7.25.19.

### 7.25.19 Destination and Branch Field Combinations

The BA field is not always a branch address. This field is also used for data and control for certain instructions. When the BD field equals two, these lines are used as data and control lines. These BA and BD field combinations are described in the following two sections.

#### 7.25.19.1 Writing Control Bus Address Transceivers

Writing the Control Bus Address Transceivers is done with D OUT and the BR ADR lines. Bus lines D OUT <15:00> are latched and then gated onto CADR lines <16:01>, respectively, during the Control Memory Cycle. BR ADR 11 is latched and gated onto CADR 00 (LSB). BR ADR <10:08> are latched and gated onto CCYCLE <0:2>, respectively. BR ADR 07 is latched. It then determines, during the Control Memory Cycle, whether the K.pli Control Bus Data Transceivers will be gated onto the CDATA lines. In order to write the register, the BD field must equal two and the Control Sequencer must be active.

Table 7-22 shows what Control Memory Cycles are generated from BR ADR <10:07>.

**Table 7-22 Control Memory Cycles Initiated from Decode of the BA Field**

| 10 | 09 | 08 | 07 | Control Memory Cycle |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | Initiate Control Memory Read Cycle |
| 1 | 0 | 0 | 1 | Initiate Control Memory Write Cycle |
| 0 | 1 | 0 | 1 | Initiate Interrupt Cycle |
| 0 | 0 | 1 | 0 | Initiate Lock Cycle |
| 1 | 1 | 1 | 1 | Initiate NMA Cycle |

#### 7.25.19.2 Writing Data Bus Address Transceivers and Counter

When the microcode writes the Data Bus Address, part of the address is written from the D OUT lines, and branch address bits BR ADR <10:09> are logically ANDed, latched, and then used to drive DNMA during the Data Bus Cycle (Figure 7-12).

Table 7-23 shows the decoding of BR ADR <10:09> to generate DNMA.

Table 7-23 Decoding of BR ADR <10:09> to Generate DNMA

| BR ADR 10 | BR ADR 09 | DNMA |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## 7.26 HARDWARE DETECTED ERRORS

Table 7-24 describes the K.pli hardware detected errors.

Table 7-24 Hardware Detected Errors

| Error | Description |
|---|---|
| Control Bus Error | This error is generated when an illegal cycle code is specified during a Control Bus Cycle. This error is part of the CNTL ERR SUM test condition and is cleared by a P.ioc clear command or a Control Sequencer clear. |
| Control Bus Parity Error | This error is generated when a value from Control Memory is received with bad parity (sensed when read from the data transceivers). This error is part of the CNTL ERR SUM test condition and is cleared by a P.ioc clear command or a Control Sequencer clear. |
| Control Bus NXM | This error is generated when a Control Bus Cycle did not assert CACK. This error is part of the CNTL ERR SUM test condition, and is cleared by a P.ioc clear command or a Control Sequencer clear. |
| Instruction Parity Error | This error is generated when a parity error is detected during an instruction fetch. This error is meaningful only when the Data Sequencer has been put into roll mode. This error is part of the CNTL ERR SUM test condition and is cleared by a P.ioc clear command or a Control Sequencer clear. |
| Scratchpad Parity Error | This error is generated when a parity error is detected during a read from Scratchpad. This error is part of the CNTL ERR SUM and DATA ERR SUM test conditions and is cleared by a P.ioc clear command or a Control Sequencer clear. |
| PLI Parity Error | This error is generated when one or more bytes read from the PLI Data Bus is received with bad parity. This error is part of the DATA ERR SUM test condition and is cleared by a P.ioc clear command or a Data Sequencer clear. |

Table 7-24 (Cont.) Hardware Detected Errors

| Error | Description |
|---|---|
| Data Bus Overrun | This error is generated when a Data Bus Cycle is initiated before completion of the preceding cycle. This error is part of the DATA ERR SUM test condition and is cleared by a P.ioc clear command or a Data Sequencer clear. |
| Data Bus Parity Error | This error is generated when one or more words with bad parity are read from Data Memory. This error is part of the DATA ERR SUM test condition and is cleared by a P.ioc clear command or a Data Sequencer clear. |
| Data Memory NXM | This error is generated when a Data Bus Cycle does not assert DACK. This error is part of the DATA ERR SUM test condition and is cleared by a P.ioc clear command or a Data Sequencer clear. |
| LINK Errors | The K.pli detects two errors generated in the LINK. Abort Transmission and Transmit Buffer Parity errors assert XMTR ATTENTION. The Control Sequencer tests XMTR ATTENTION at the Control Test Select multiplexer. The Control Sequencer must then issue a read transmitter status command to the LINK and check the bits of the returned status to see what caused XMTR ATTENTION. |

# CHAPTER 8
# HSC50 INPUT/OUTPUT CONTROL PROCESSOR MODULE

## 8.1 INTRODUCTION

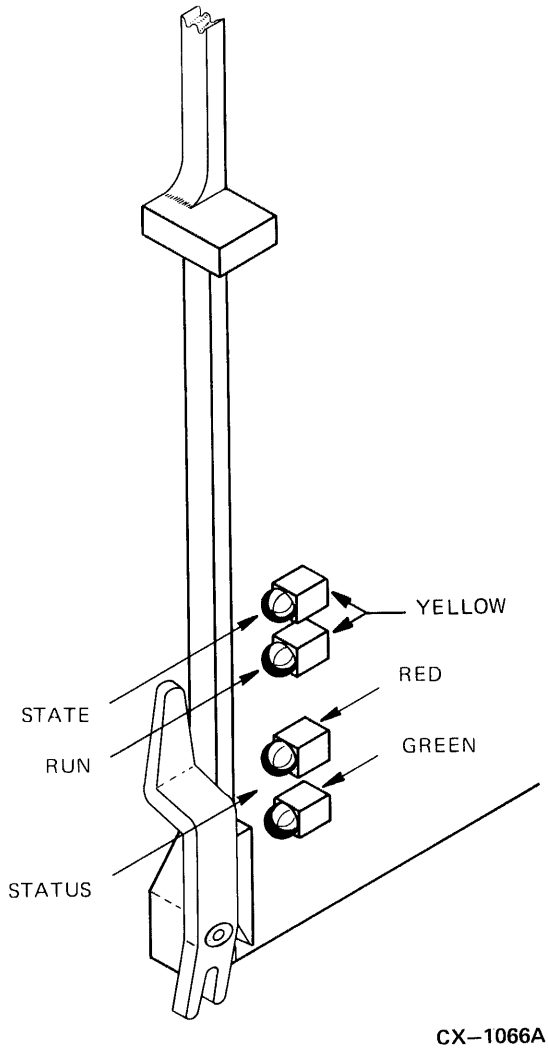
The HSC50 Input/Output Control Processor Module (F-11 P.ioc) is an extended hex module which contains:

- Four LEDs for status indications
- An F-11 processor and memory management
- Timing and interrupt logic for the F-11
- An interface to Program Memory
- A bootstrap PROM
- I/O page windows into Control Memory
- Control, timing, and arbitration for the HSC50 Data Bus and Control Bus
- Interface for two TU58 cassette drives
- Pio Control and Status Register
- An interface to the HSC50 Operator Control Panel (Switch/Display Register)
- K Init and Status Registers
- Error Address Register
- Serial Number Register
- Control Memory and Data Memory interfaces

## 8.2 LED INDICATORS

The P.ioc contains four LED indicators: one red, one green, and two yellow (Figure 8-1). These are described in the following sections.

Figure 8-1 P.ioc LEDs



### 8.2.1 P.ioc Diagnostic OK LEDs

The red and green LEDs are used to indicate whether the P.ioc successfully completed all of its initialization diagnostics. The module powers up with the red LED on and the green LED off. Upon successful completion of the power up diagnostics, the red LED is turned off and the green LED is turned on. These LEDs are controlled by a bit in the Switch/Display Register. (Refer to Section 8.13.)

### 8.2.2 RUN LED

The RUN LED, a yellow LED adjacent to the red Diagnostic OK LED, blinks once for each PDP-11 instruction fetch cycle. This is analogous to the RUN light on standard LSI-11 processor systems. When the F-11 is running, this LED is illuminated at about half-brilliance (compared to the other LEDs on the module).

### 8.2.3 State LED

The yellow State LED is controlled by bit <03> of the Pio Control and Status Register (PIOCSR) and is connected in parallel with the State indicator on the Operator Control Panel.

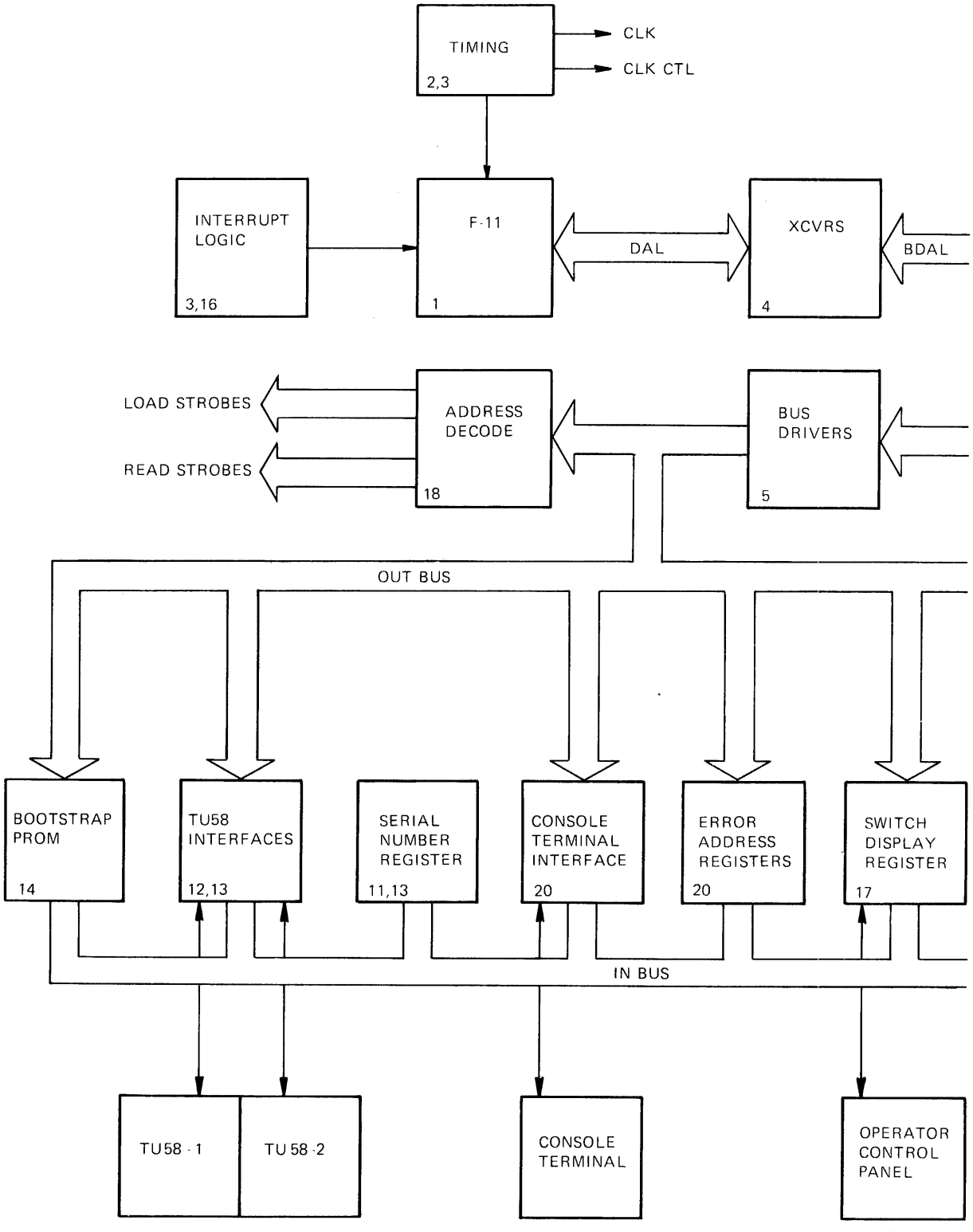
## 8.3 HSC50 INPUT/OUTPUT CONTROL PROCESSOR MODULE BLOCK DIAGRAM

Figure 8-2 is a block diagram of the P.ioc showing the principal hardware elements and data paths. Each of these elements will be described in the following sections.

## 8.4 F-11 BLOCK

The F-11 Block contains the F-11 processor and associated logic. The F-11 block is located on the upper-left side of Figure 8-2. The F-11 is implemented using three chips. Two MOS/LSI chips, data and control, implement the basic processor (Figure 8-3). The third chip, the memory management unit (MMU), provides a PDP-11/34 software-compatible memory management scheme.

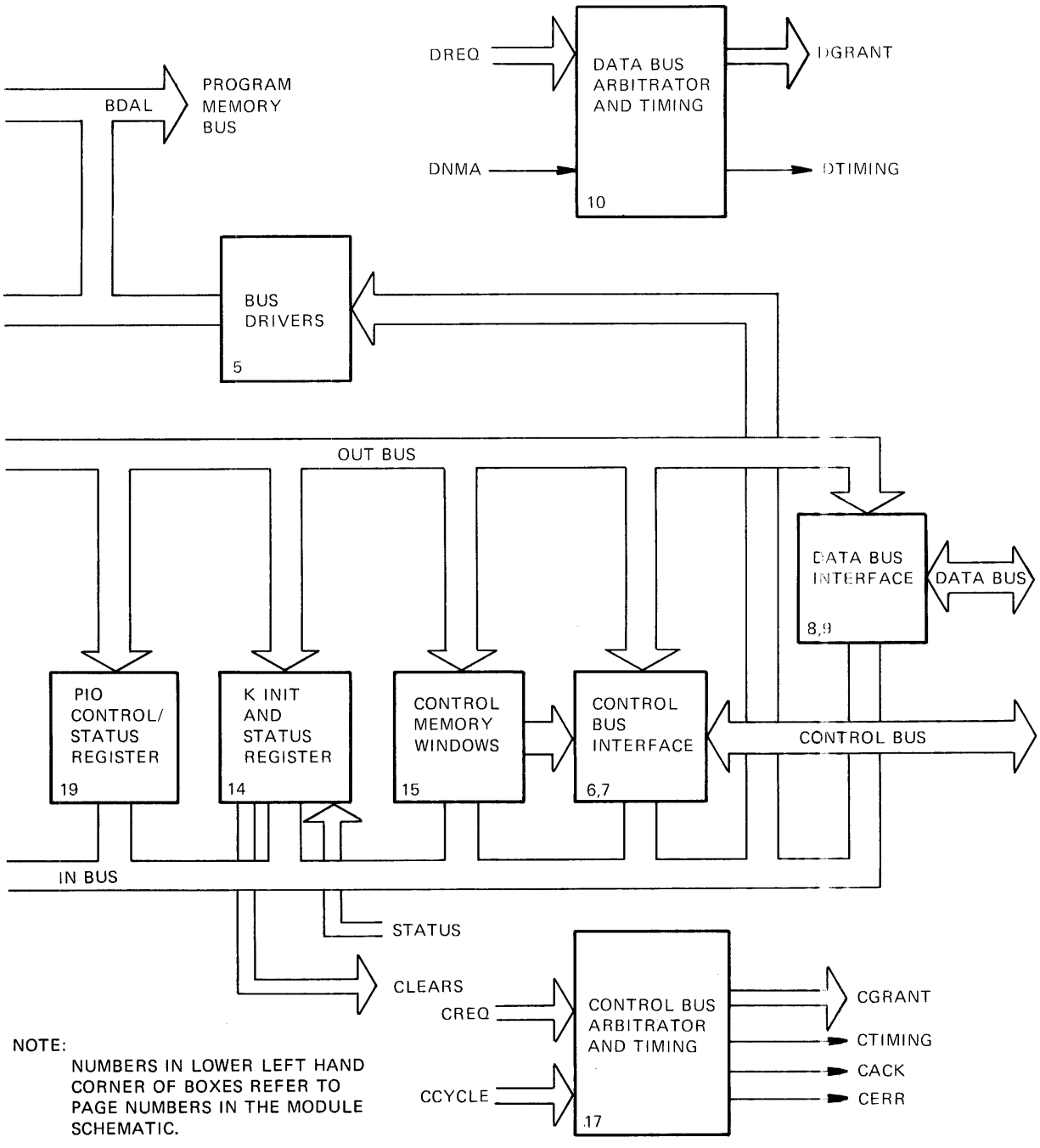
**Figure 8-2 HSC50 Input/Output Control Processor Module Block Diagram**




CX–1067A
Sheet 1 of 2

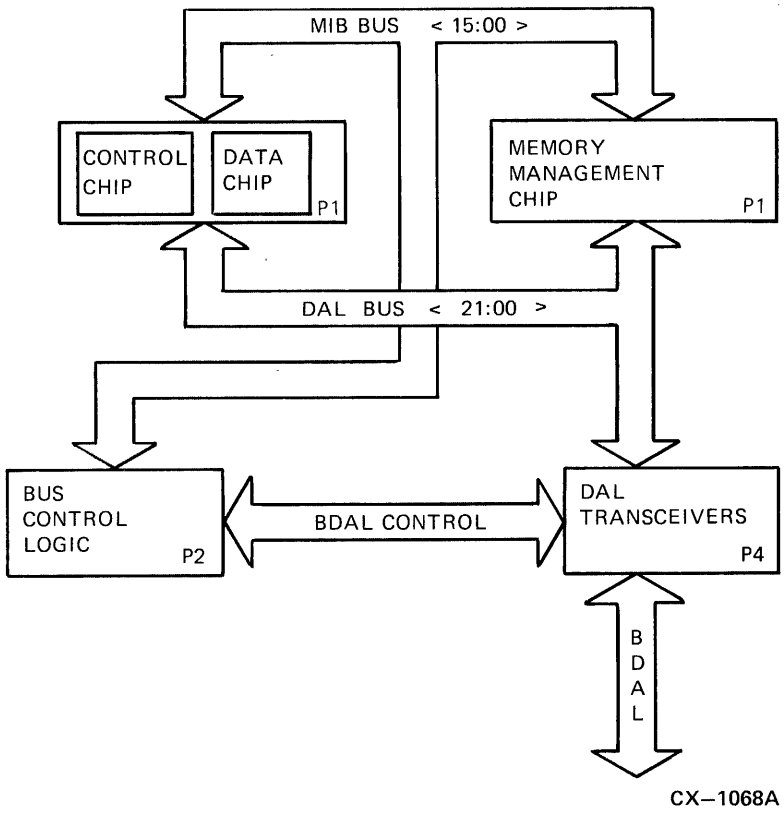
**(Continued on next page)**

**Figure 8-2 (Cont.) HSC50 Input/Output Control Processor Module Block Diagram**




CX–1067A
Sheet 2 of 2

Figure 8-3 F-11 Processor Functional Block Diagram




The data chip (DC302) performs the following functions:

- Performs all arithmetic and logical functions
- Handles data and address transfers with the external world
- Coordinates most interchip communication

The control chip (DC303) does microprogram sequencing for PDP-11 instruction decoding and contains the control store ROM. The data and control chips are both contained in one 40-pin package. The MMU chip (DC304) contains the registers for 22-bit memory addressing.

### 8.4.1 Memory Management Unit Chip

The Memory Management Unit (MMU) chip provides the memory management function. This chip provides dual mode (user and Kernel) address relocation of 22 bits. Sixteen-bit virtual addresses are received from the data chip via the Data Address Lines (DAL), relocated to the appropriate 22-bit physical address and then sent on the DAL to replace the original virtual address for transmission to the Program Memory Bus. The MMU chip contains the status registers and active page registers (PAR/PDR register pairs), as well as access protection and error detection capability.

The MMU chip is controlled by information received on the Microinstruction Bus (MIB) from both the data chip and the control chip, and by several discrete control inputs.

Aborts generated by protection hardware are vectored through kernel address $250_8$. The software uses status registers to determine why the abort occurred. Note that an abort to a location which is itself an invalid address will cause another abort. Thus, the kernel program must ensure the kernel virtual address $250_8$ is mapped into a valid physical address. Otherwise, an infinite loop occurs.

The Memory Management chip contains four status registers:

- Memory Management Register 0
- Memory Management Register 1
- Memory Management Register 2
- Memory Management Register 3

The following sections describe these four memory management registers.

#### 8.4.1.1 Memory Management Status Register 0

Memory Management Status Register 0 (SR0) contains abort error flags, memory management enable, and other essential information required to recover from an abort. The register is located at $17777572_8$. The format of SR0 is shown in Figure 8-4. Table 8-1 describes the bits in the SR0.

**Figure 8-4 Memory Management Status Register 0**

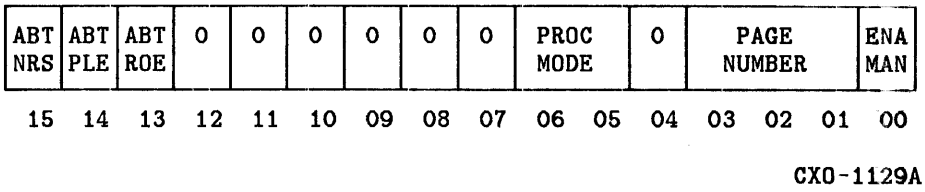



**Table 8-1 Memory Management Status Register 0 Bit Description**

| Bits | Description |
|---|---|
| 15 | ABORT NON-RESIDENT — This bit is set by attempting to access a page with an Access Control Field key equal to 0 or 4. |
| 14 | ABORT PAGE LENGTH ERROR — This bit is set by attempting to access a location in a page with a block number (virtual address bits <12:06>) outside the area authorized by the Page Length Field of the Page Descriptor Register for that page. |
| 13 | ABORT READ-ONLY ERROR — This bit is set by attempting to write in a read-only page. Read-only pages have access keys of 2. |
| 12:07,04 | These bits are read-only and are always 0. |
| 06:05 | PROCESSOR MODE — These bits indicate the processor mode (kernel=00, user=11) associated with the page causing the abort. |

Table 8-1 (Cont.) Memory Management Status Register 0 Bit Description

| Bits | Description |
|---|---|
| 03:01 | PAGE NUMBER — These bits indicate the virtual page number referenced. Pages, like blocks, are numbered from 0 upwards. The page number field is used by the error recovery routine to identify the page being accessed if an abort occurs. |
| 00 | ENABLE MANAGEMENT — When set, all addresses are relocated and protected by the Memory Management Unit. When reset, the Memory Management Unit is disabled and addresses are not relocated or protected. |

### 8.4.1.2 Memory Management Status Register 1

The Memory Management Status Register 1 (SR1) is implemented on some PDP-11 computers to provide additional capability. The HSC50 does not implement this register but does respond to its bus address, $17777574_8$, and reads it as all 0s. This information is provided for clarity only.

### 8.4.1.3 Memory Management Status Register 2

Memory Management Status Register 2 (SR2) is loaded with the 16-bit virtual address (VA) at the beginning of instruction fetch, but is not updated if the instruction fails. SR2 is the virtual address program counter. Upon abort, the result of SR0 bits 15, 14, or 13 being set freezes SR2 until the SR0 abort flags are cleared. The SR2 register is located at $17777576_8$ and is a read-only register.

### 8.4.1.4 Memory Management Register 3

Memory Management Register 3 (MMR3) enables or disables 22-bit mapping. MMR3 is located at $17772516_8$ and is cleared at power up, by a console start, and by a RESET instruction. Figure 8-5 shows the format of the MMR3. Table 8-2 describes the bits in this register.

Figure 8-5 Memory Management Register 3

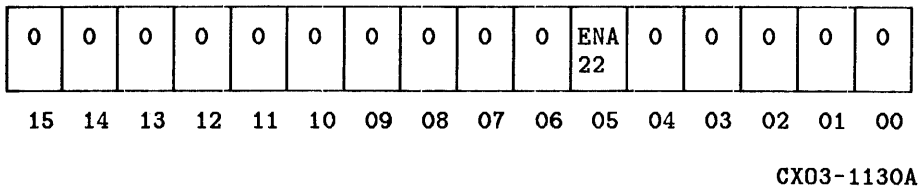


Table 8-2 Memory Management Register 3 Bit Description

| Bits | Description |
|---|---|
| 15:06<br>04:00 | These bits are read-only and are always 0. |
| 05 | ENABLE 22-BIT MAPPING — When set, this bit selects 22-bit relocation when MMR0 <00> is set. When cleared, 18-bit relocation is selected when MMR0 <00> is set. |

### 8.4.2 Bus Control Logic

The Bus Control Logic generates the necessary clock, strobe, and enables for the DAL transceivers.

### 8.4.3 Data-Address Lines

The Data-Address Lines (DAL) Bus is routed between all the MOS chips and the Program Memory Bus Transceivers. The 22-bit DAL Bus is time multiplexed. During clock-high time, the DAL bus transfers data from the data chip to the other MOS chips or between the rest of the module and the MOS chips. During clock-low time, the DAL Bus transfers service data (external and internal interrupt requests) from the module to the control chip.

### 8.4.4 Microinstruction Bus

The 16-bit Microinstruction Bus (MIB) is common to all data and control chips. A subset of the MIB is routed to the MMU because it does not need access to all MIB control signals.

The MIB is time-multiplexed and is used for different functions during clock high and low times. During clock-high time, the MIB transfers control information from the data chip to all control chips, the MMU, and the rest of the logic. During clock-low time, the MIB transfers microinstructions from the control chip to the data chip.

NOTE

**For more detail on the F-11, refer to the** *Microcomputer Processor Handbook 1979-1980.*

## 8.5 MICRO-ODT

Micro-ODT, as currently implemented in the F-11 microcode, accepts either 16- or 18-bit addresses, but not 22-bit addresses. Since the HSC50 utilizes the full 22-bit addressing capability of the MMU, and since special hardware is required to implement 18-bit μODT addressing, the P.ioc only allows 16-bit addressing in μODT. This provides access to the first 28 Kwords of Program Memory and the I/O page. However, since the I/O page is accessible, all available Control Memory is accessible using the Control Memory Windows. Data Memory cannot be accessed using μODT. Note that even if the MMU is enabled, the μODT microcode disables relocation.

## 8.6 INTERRUPT LOGIC

The Interrupt Logic generates the interrupt and vector address to the F-11 (shown on the left side of Figure 8-2). Table 8-3 shows the interrupt and trap vectors that are assigned in the F-11 P.ioc. Notice that the Control Bus can software interrupt on Level 7 through 4 (Section 8.16.4 and Chapter 3).

Table 8-3 Interrupt and Trap Vectors for F-11 P.ioc

| Trap Vector Address | Interrupt Level | Function |
|---|---|---|
| 000 | | (reserved) |
| 004 | Non-maskable | Bus error (non-existent memory) |
| 010 | None | Control error (non-existent F-11 Control Chip) |
| 014 | Non-maskable | BPT, breakpoint trap used for HSC ODT |
| 020 | Non-maskable | IOT, Input/Output Trap, used for HSC software inconsistency |
| 024 | Non-maskable | Power fail |
| 030 | Non-maskable | EMT, emulator trap, used for system service call |
| 034 | Non-maskable | TRAP instruction |
| 060 | 4 (3) | Console terminal, keyboard |
| 064 | 4 (4) | Console terminal, printer |
| 100 | 6 [1] | Line clock |
| 114 | Non-maskable | Parity error on read-from-memory |
| 120 | 4 (5) | Control Bus Interrupt, level 4 |
| 124 | 5 | Control Bus Interrupt, level 5 |
| 130 | 6 | Control Bus Interrupt, level 6 |
| 134 | 7 | Control Bus Interrupt, level 7 |
| 250 | Non-maskable | MMU abort |
| 300 | 6 [2] | TU58 1, receiver |
| 304 | 6 [3] | TU58 1, transmitter |
| 310 | 4 (1) | TU58 2, receiver |
| 314 | 4 (2) | TU58 2, transmitter |

Note that the numbers in parentheses () indicate the order of priority of level 4 interrupts. The numbers in brackets [] indicate the order of priority of level 6 interrupts.

## 8.7 ADDRESS DECODE

The Address Decode logic decodes addresses and provides the load strobes and read strobes for the I/O page registers located on the P.ioc module. The decode is shown on the left of Figure 8-2.

## 8.8 BOOTSTRAP PROM

The bootstrap PROM contains 1 Kwords starting at location $17773000_8$ (shown on the left side of Figure 8-2). This code is executed after:

- A power-up
- A subsystem INIT (from either the host or the Operator Control Panel)
- A software-initiated INIT

Two starting addresses are provided: the *cold* boot starts at $17773000_8$ and the *warm* boot starts at $17773010_8$. A cold boot happens when the HSC powers up or when the INIT switch is pushed. A warm boot happens when the system crashes or the set show utility requests that the HSC reboot. During a warm boot the software checks to see what caused the boot (for example, an NXM or parity error).

The F-11 is hard-wired for Power Up Mode 2. This mode causes automatic execution of the bootstrap at address $17773000_8$ whenever the F-11 is initialized.

## 8.9 TU58 INTERFACE

The TU58 Interface logic provides a parallel interface to the P.ioc and a serial interface with the TU58.

### 8.9.1 Parallel P.ioc Interface

The P.ioc has two separate TU58 interfaces. However, the HSC50 is shipped with only one TU58 Controller and two drives. This section describes only one of the interfaces. The interface has the same characteristics as a DL-11D. It contains the following four registers:

- Receiver Status Register (RCSR) $177520_8$
- Receiver Data Buffer Register (RBUF) $177522_8$
- Transmitter Status Register (XCSR) $177524_8$
- Transmitter Data Buffer Register (XBUF) $177526_8$

Figures 8-6, 8-7, 8-8, and 8-9 show the format of these four registers. Tables 8-4, 8-5, 8-6, and 8-7 describe the bits in the registers.

**Figure 8-6 Receiver Status Register**

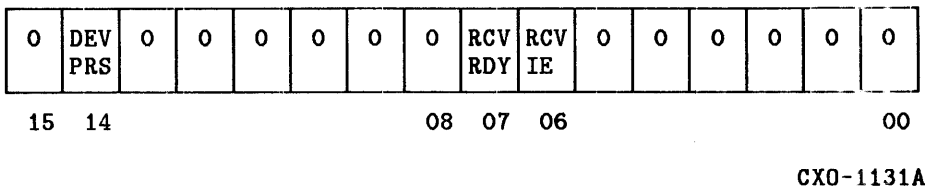


Table 8-4 Receiver Status Register Bit Description

| Bit | Description |
|---|---|
| 14 | DEVICE PRESENT — This read-only bit indicates a TU58 (or terminal) is plugged into this interface. This line is RS232C/RS423 compatible. |
| 07 | RECEIVER READY — This read-only bit indicates that there is a character in RBUF to be read. |
| 06 | RECEIVER INTERRUPT ENABLE — This read/write bit, when set, causes an interrupt to be generated on the assertion of RCV RDY. Note that setting this bit when RCV RDY is already asserted will cause an interrupt. |

Figure 8-7 Receiver Data Buffer Register

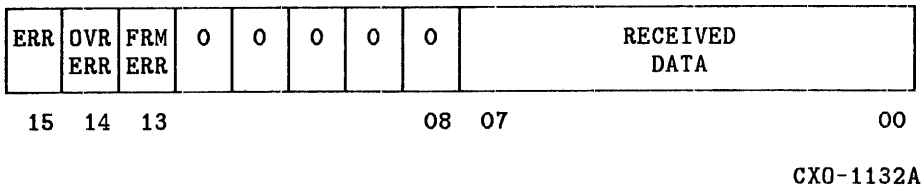



Table 8-5 Read Data Buffer Register Bit Description

| Bits | Description |
|---|---|
| 15 | ERROR — This read-only bit is the inclusive-OR of OVERRUN ERROR and FRAMING ERROR. |
| 14 | OVERRUN ERROR — This read-only bit indicates that the receiver attempted to set bit 07 in RCSR when it was already set. In other words, a new character was received and the preceding character had not yet been read. |
| 13 | FRAMING ERROR — This read-only bit indicates no stop bit was received. |
| 07:00 | RECEIVED DATA — These are the 8-bit characters just received. This byte is read-only. |

Note that there is no parity error bit implemented in this register. Parity is not checked or generated on any of the serial interfaces.

Figure 8-8 Transmitter Status Register

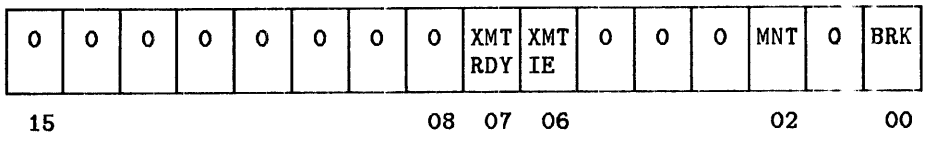



Table 8-6 Transmitter Status Register Bit Description

| Bit | Description |
|---|---|
| 07 | TRANSMITTER READY — This read-only bit, when set, indicates that XBUF can accept a character for transmission. |
| 06 | TRANSMITTER INTERRUPT ENABLE — This read/write bit, when set, causes an interrupt to be generated on the assertion of XMT RDY. Note that setting this bit when XMT RDY is already set will cause an interrupt to be generated. |
| 02 | MAINTENANCE — This read/write bit, when set, causes the serial output of the transmitter section to be connected to the input of the receiver section. Thus, a character loaded into XBUF can then be read back in RBUF. |
| 00 | BREAK — This read/write bit, when set, causes the transmitter to send a continuous SPACE which will be detected by the receiver as a framing error. |

Figure 8-9 Transmitter Data Buffer Register

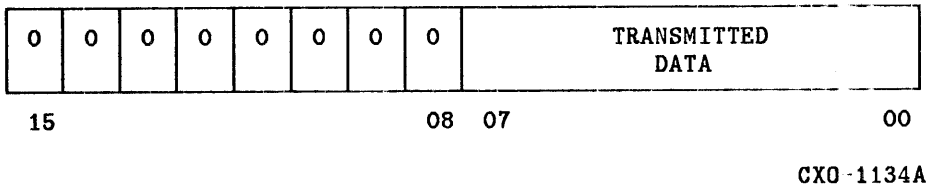



Table 8-7 Transmitter Data Buffer Register Bit Description

| Bits | Description |
|---|---|
| 07:00 | TRANSMITTED DATA — These write-only bits are loaded with the character to be transmitted. |

Data format for all three serial interfaces is 8 data bits, no parity, and 1 stop bit.

The first TU58 interface occupies addresses $17777520_8$—$17777526_8$ and uses $300_8$ (Rcvr) and $304_8$ (Xmtr) for its vectors. The second TU58 interface (not used) occupies addresses $17777530_8$—$17777536_8$ and uses $310_8$ (Rcvr) and $314_8$ (Xmtr) for its vectors. The first TU58 interface interrupts at level 6 and the second TU58 interface interrupts at level 4.

### 8.9.2 Serial Interface

The serial interface is an RS232C/RS423 compatible asynchronous full duplex serial line. Control commands are distinguished from binary data via the radial-serial bus protocol. Transmit and receive baud rates are the same and are jumper-selectable for 38.4, 19.2, or 9.6 Kbaud. Table 8-8 shows the configuration for baud rate selection jumpers. The baud rate is jumper-selectable according to the following table:

**Table 8-8 TU58 Serial Interface Baud Rate Selection**

| Jumper W4 | Jumper W5 | TU58 Baud Rate |
|---|---|---|
| IN | IN | 38.4K |
| IN | OUT | 9.6K |
| OUT | IN | 19.2K |
| OUT | OUT | 38.4K |

## 8.10 CONSOLE TERMINAL INTERFACE

The Console Terminal Interface is a EIA RS232C/RS423 serial interface. This interface is connected to the console terminal. The console terminal is the communications link between the processor and you.

The hardware interface for this terminal is also a DL-11D interface. The register set is the same as for the TU58, above, and occupies the addresses $17777560_8$—$17777566_8$. Vectors for this interface are $60_8$ (Rcvr) and $64_8$ (Xmtr). It interrupts at level 4. Available baud rates are 300 and 9600 baud. The baud rate is selected by Jumper W3: in for 9600 baud, out for 300 baud.

Reception of a *break* character (that is, detection of a framing error by the UART) on this interface will assert the processor's HALT line, if the Secure/Enable switch is in the ENABLE position, causing the F-11 to enter $\mu$ODT.

Note that the UART remembers a break character until a subsequent valid character is received. Therefore, if a break is received while the Secure/Enable switch is in the SECURE position, and no other valid character is subsequently received, the break will be remembered and subsequently switching the Secure/Enable switch to ENABLE will cause the F-11 to enter $\mu$ODT immediately. This may be caused by powering the console terminal on or off while leaving the HSC50 powered on.

An additional maintenance feature is included in this interface. If the INIT pushbutton on the Operator Control Panel is held depressed, any character received from the console keyboard is immediately echoed back to the terminal, thus providing a simple test of the serial line driver and receiver while bypassing the UART and associated hardware.

This terminal interface is also used to communicate with the F-11 $\mu$ODT.

## 8.11 SERIAL NUMBER REGISTER

The 16-bit Serial Number Register (shown just left of the middle in Figure 8-2) provides a unique identifiable number for each P.ioc module. The serial number is encoded by adding or removing jumpers on the module. The address of this register is $17770056_8$.

## 8.12 ERROR ADDRESS REGISTER

The Error Address Register consists of two locations: Low Error Address and High Error Address (shown in the middle of Figure 8-2). These two registers catch the full 22-bit physical address on the OUT BUS at the time a NXM or a parity trap occurs. These registers are frozen by the occurrence of the error and each, independent of the other, remains frozen until it is read. After being read, each register will resume catching the corresponding portion of the address. Following are the addresses of the two registers:

Low Error Register — $17770024_8$
High Error Register — $17770026_8$

Figures 8-10 and 8-11 show the format of these two registers.

**Figure 8-10 Low Error Address Register**

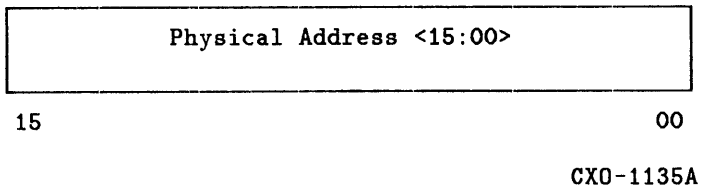
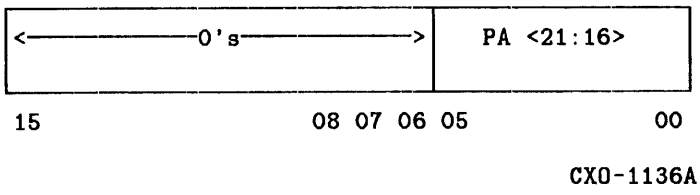



**Figure 8-11 High Error Address Register**

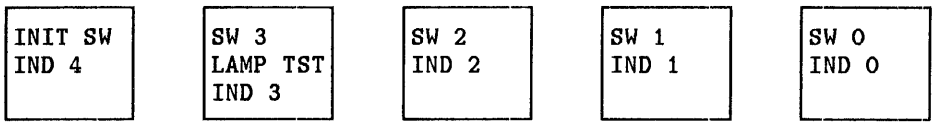



## 8.13 SWITCH/DISPLAY REGISTER

The Switch/Display Register (shown in the lower middle of Figure 8-2) is the interface between the software and the Operator Control Panel. This read/write register is located at $17770042_8$.

Figure 8-12 shows the arrangement of the five switches and indicators on the Operator Control Panel. Response to operation of these switches is handled entirely by software; no interrupts to the F-11 are generated by changes in state of the switches (except, of course, for INIT).

**Figure 8-12 Indicators and Switches on the Operator Control Panel**




CXO-1137A

The functions of the five panel switches and indicators are defined by software, except as follows:

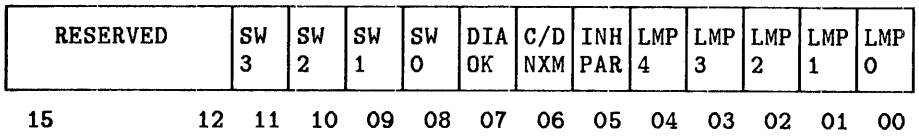
- INIT—a momentary push button which, when pushed and released, causes the F-11 to execute its bootstrap routine.
- SW 3—a momentary push button which, when pushed, provides a hardware lamp test.

In addition, there is a two-position rocker switch, SECURE/ENABLE. This switch provides three functions:

- Whenever the switch is in the ENABLE position, a break received from the terminal causes the F-11 to enter $\mu$ODT.
- If the F-11 is in Kernel mode and HALT instruction is decoded, the state of this switch is sampled. If the switch is in the ENABLE position, the F-11 enters $\mu$ODT. If the switch is in the SECURE position, the F-11 executes a trap to location $10_8$. If the F-11 is in user mode, a HALT instruction causes a trap to location $10_8$.
- In addition, if this switch is in the SECURE position, the INIT switch is disabled.

Figure 8-13 shows the format of the Switch/Display Register. Table 8-9 describes each bit of the register. The software must poll this register to detect changes in the state of the switches.

**Figure 8-13 Switch/Display Register**




CXO-1138A

Table 8-9 Switch/Display Register Bit Description

| Bits | Description |
|---|---|
| 15:12 | These bits are read-only and always zero. |
| 11:08 | SWITCHES <3:0> — These bits reflect the state of the corresponding switches on the Operator Control Panel. These bits are read-only. Note that no hardware debounce logic is provided; debouncing is done by software. |
| 07 | DIAGNOSTIC OK — This bit lights the Diagnostic OK LEDs. A zero lights the red LED and a one lights the green LED. This bit is zero on power up and is set to a one upon completion of the PROM bootstrap diagnostics. The intent of this bit is a simple visual GO/NO-GO indicator for the P.ioc. |
| 06 | CONTROL/DATA NON-EXISTENT MEMORY — This bit, when set, forces a NXM trap on any Control or Data Memory Bus access. |
| 05 | INHIBIT PARITY TRAP — This bit, when set, inhibits all parity traps, thus enabling examination of any memory location which has a hard parity error. |
| 04:00 | LAMPS <4:0> — These bits, when set, light the corresponding indicators on the Operator Control Panel. These bits are read/write. |

The last 8 writable bits are cleared by Host Clear, Bus Init, and I/O Clear.

## 8.14 PIO CONTROL AND STATUS REGISTER

The Pio Control and Status Register (PIOCSR) contains bits that control functions and provide status information on the P.ioc.

Figure 8-14 shows the format of the PIOCSR. Table 8-10 describes each bit of the register. Except as noted in Table 8-10, all bits are read/write and are cleared by either Bus Init or IO Clear. This register is located at $17770040_8$.

Figure 8-14 Pio Control and Status Register

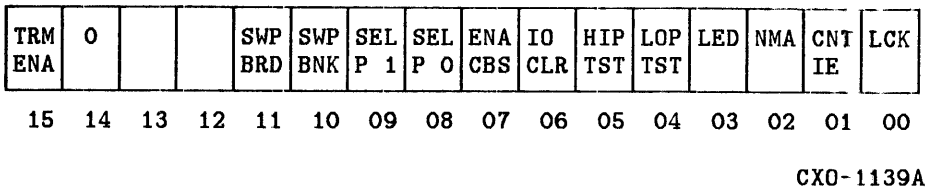


Table 8-10 Pio Control and Status Register Bit Description

| Bits | Description |
|---|---|
| 15 | TERMINAL ENABLE — This bit corresponds to the position of the Operator Control Panel SECURE/ENABLE switch. A 1 indicates the ENABLE position of the switch (read-only). |
| 14 | This bit is always zero. |
| 13 | This spare read/write bit is not used. |
| 12 | This spare read/write bit is not used. |
| 12 | Spare |
| 11 | SWAP BOARD — This bit, when set, causes the first two memory boards in the system to swap the base address for the Program Memory on each board (not used). |
| 10 | SWAP BANK — This bit, when set, causes the first memory board in the system to swap the addressing ranges between Program Memory banks on that board. These two bits are provided to maximize the probability that working memory is available starting at address 000000 for interrupt vectors. |
| 09:08 | SELECT P <1:0> — These bits provide a code which identifies which Control Bus interrupt field this P will respond to. These two bits are always zero in the HSC50. |
| 07 | ENABLE CONTROL BUS ARBITRATOR — This bit, when set, turns on the Control Bus Arbitrator for the seven other Control Bus requesters. When clear, only the P.ioc may access Control Memory. |
| 06 | IO CLEAR — This write-only bit is used to clear I/O page device registers. |
| 05 | HIGH-BYTE PARITY TEST — This bit, when set, causes the high byte parity tree to generate even parity on the next write-cycle to memory, therefore writing incorrect high-byte parity at that address. The bit is automatically cleared following that write cycle. This function applies to all three areas of memory (Program, Control, and Data). This bit is also cleared by BUS INIT, but not by I/O CLEAR. |
| 04 | LOW-BYTE PARITY TEST — This bit performs the same function as bit <05> except for the low byte. |
| 03 | LED — This bit lights a yellow LED, D6, located near the handle of the P.ioc and also drives the STATE lamp located on the Operator Control Panel. |
| 02 | NON-MEMORY-ACCESS — This bit, when set, causes the P.ioc to execute an NMA cycle on the very next Control Memory or Data Memory access it makes. This bit is cleared automatically at the conclusion of that Control or Data Memory access. This bit has no effect on and is not cleared by Program Memory accesses. |
| 01 | CONTROL MEMORY INTERRUPT ENABLE — This bit, when set, causes the P.ioc to execute an interrupt cycle on the very next Control Memory access it makes. This bit is cleared automatically at the conclusion of that Control Memory access. Note that if the P.ioc accesses Data Memory with this bit set, the bit will be cleared after that access with no interrupt generated. |

Table 8-10 (Cont.) Pio Control and Status Register Bit Description

| Bits | Description |
|---|---|
| 00 | CONTROL MEMORY LOCK — This bit, when set, causes the P.ioc to execute a lock cycle on the next Control Memory access it makes. This bit is cleared automatically at the conclusion of that memory access. Note that if the P.ioc accesses Data Memory with this bit set, the bit will be cleared after that access with no lock cycle generated. |

The last three bits (NMA, CNT IE, and LCK) have been prioritized such that, if more than one bit is set at a time, the highest priority bit will determine which type of cycle will occur. Following that cycle, *all* set bits will be cleared. The priority of these bits is:

1. NMA
2. CNT IE
3. LCK

## 8.15 K INIT AND STATUS REGISTERS

The K INIT and Status Registers initialize and hold the eight bit status of the:

Host Interface (K.pli)
Disk Data Channel (K.sdi)
Tape Data Channel (K.sti)

Figure 8-15 shows the format of the K INIT Register located at $17770044_8$. Table 8-11 describes the bits in the K INIT Register.

Figure 8-15 K INIT Register

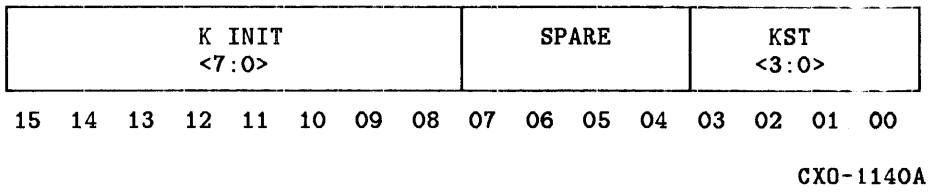



Table 8-11 K INIT Register Bit Description

| Bits | Description |
|---|---|
| 15:08 | K INITIALIZE <7:0> — These bits are backplane slot-specific clear signals. A zero in any bit position clears the module in the corresponding backplane slot. A one in any bit position negates the clear to that module and enables it to run. These bits are cleared to zero on power-up only. Note that since the P.ioc is considered Requester 0 in the HSC50, K INIT <0> has no significance. |

Table 8-11 (Cont.) K INIT Register Bit Description

| Bits | Description |
|---|---|
| 07:04 | These bits are spare bits. |
| 03:00 | SELECT K STATUS — These four bits are decoded to select the status of a requester. The codes refer to specific backplane slots. These bits are written first and then the Status Register is read to obtain the status. The HSC50 provides for only eight requesters (codes $00_8$ through $07_8$). Any code greater than $07_8$ generates a STATUS byte of $377_8$ or all ones in the Status Register. |

Figure 8-16 shows the format of the Status Register located at $17770046_8$. Table 8-12 describes the bits in the Status Register.

Figure 8-16 Status Register

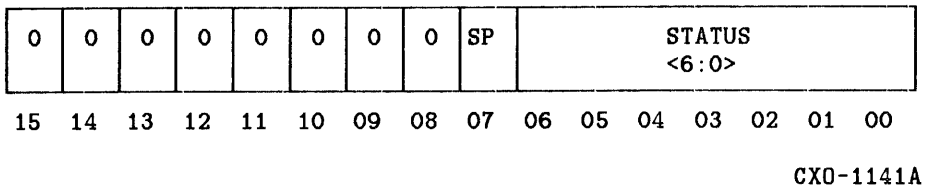



Table 8-12 Status Register Bit Description

| Bits | Description |
|---|---|
| 07 | STATUS PARITY — This bit is Status Parity. The Ks supply the parity bit from software (even or odd). The P.ioc checks the parity with software. |
| 06:00 | STATUS — These bits are the 7-bit status code returned from each K. Note that if a backplane slot is not occupied, reading the status for that slot returns a status byte of $377_8$ or all ones. |

NOTE

**Both the K INIT Register and Status Register are read/write. However, byte-writes to K INIT Register write all 16 bits of this register.**

## 8.16 THE INTERFACE TO CONTROL MEMORY

The interface to Control Memory consists of three blocks:

- Control Memory Windows
- Control Bus Interface
- Control Bus Arbitrator and Timing

The following sections describe these three blocks which are shown at the right side of Figure 8-2.

### 8.16.1 Control Memory Windows

The P.ioc software has very little need to access Data Memory because the Ks load and unload data buffers. But the software needs to access many different control blocks scattered all over Control Memory. In addition to the standard MMU, there is a need for a separate address relocation feature for Control Memory. A large number of page address relocation registers is needed, but page descriptor registers are not necessary.

The Control Window feature provides 1024 Window Address Registers (WADRs) They are used whenever there is a virtual address in the range $16xxxx_8$.

**NOTE**

**Throughout this discussion of Control Window Logic, a 16-bit address is used for ease of explanation. However, the MMU translates this 16-bit address to a 22-bit address, then gates it onto the BDAL Bus.**

Whenever an address is in the $16xxxx_8$ range:

- The hardware activates the Control Memory Window logic
- The appropriate Window Address Register is gated (along with part of the virtual address) to the Control Memory Address (CADR) lines
- The Control Bus is requested
- Control Memory is accessed

The software can still use the MMU to access Control memory, but this is rarely done.

The software can access the WADRs to load and read them by using I/O page addresses.

All of the hardware for this feature resides on the P.ioc module and consists of the following:

- 1024 WADRs contained in a 1K x 16 RAM
- WINDR $170020_8$ - Window Index Register
- WBUSR $170022_8$ - High Order Address History Register
- LOCADR $170054_8$ - Low Order Address History Register

The WADRs are the actual page address registers. The WINDR is an index register used as part of the WADR 1-of-1024 selection. The WBUSR and LOCADR are history registers that hold the CADR of the last Control Memory access for error recovery and diagnostic purposes.

#### 8.16.1.1 Control Memory Window Address Selection

The software uses the Control Memory window logic to change a 16-bit PDP-11 address into a 17-bit Control Memory address. This 17-bit address then points to a control block in Control Memory. Control blocks start on a double-word boundary and the address represents an offset from the start of Control Memory measured in increments of double words.

To change the 16-bit address to a 17-bit address the software performs the following four steps:

- Selects a window set
- Writes a double-word offset address into a WADR
- Reads back a virtual address
- Uses the virtual address to address Control Memory

Figure 8-17 is a sample PDP-11 macro program that shows the steps involved in changing the 16-bit address to a 17-bit address using the WINDR and WADR. The following describes the steps:

Figure 8-17 WADR Addressing

```
012767 000005 170020  MOV 1  #5,$WINDR     ;select window set #5 using the
                                            ;window index register
    .
    .
    .
016767 100000 170006  MOV 2  DBPTR,$WADR3 ;MOV the double word offset to WADR3
016700 170006         MOV 3  $WADR3,R0     ;get the virtual address in R0
    .
    .
    .
011002                MOV 4  (R0),R2       ;get the contents of Control Memory
                                            ;and put it in R2
    .
    .
    .
                                                              CXO-1142A
```

1 The first MOV writes a 5 in the WINDR which selects window set number 5 out of a possible 128 sets. Figure 8-18 shows how the 1 Kword by 16-bit RAM is divided into 128 window sets of eight WADRs.

The WINDR is loaded once prior to using a window set.

2 The second MOV writes the double-word offset address from the Control Memory virtual address of 0 to WADR3. The source is gated directly into the WADR without shifting the bits, as shown in the upper portion of Figure 8-19.

3 The third MOV reads back a *sort of* virtual address from WADR3. The low order seven bits of the WADR3 are shifted left two places and gated onto the IN BUS as show in the middle portion of Figure 8-19. The I/O page address is $170006_8$. Table 8-13 shows the register address of each WADR and the virtual I/O page address range each covers.

4 The fourth MOV uses the *sort of* virtual address obtained from step 3 above to actually address Control Memory. When the Control Window Logic is activated (by a virtual address in the $16Zxxx_8$ range), the high order nine bits of the WADR are shifted left one place and gated onto the CADR lines as shown in the bottom portion of Figure 8-19.

**NOTE**

**The Z in the virtual address $16Zxxx_8$ is the WADR register.**

Figure 8-18 Window Address Register Selection

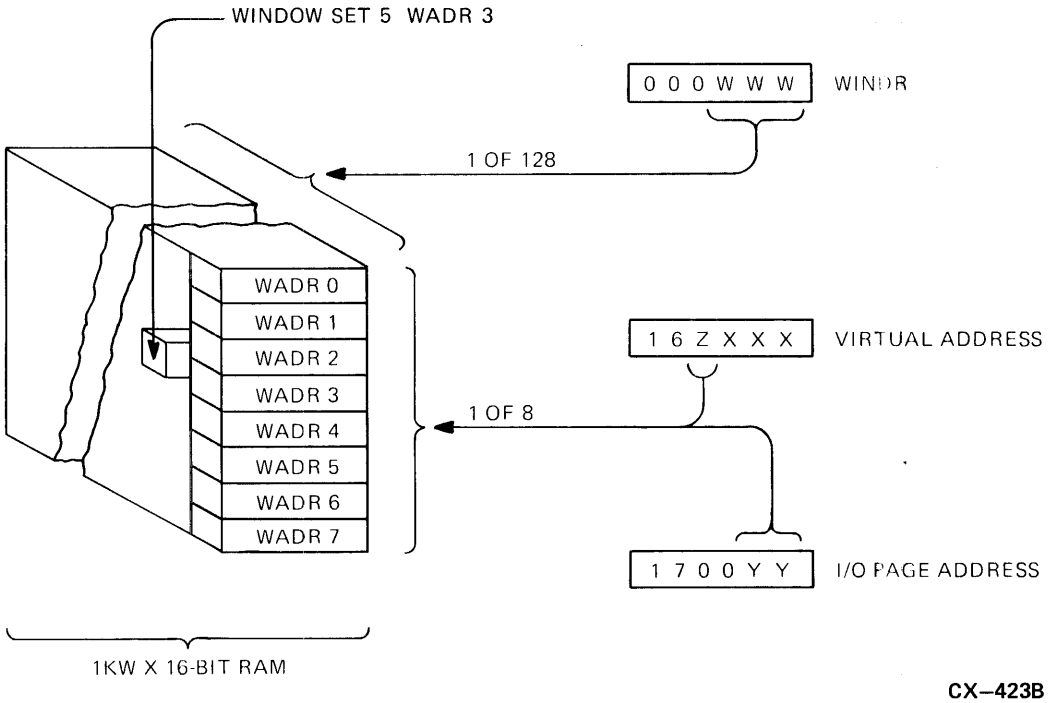


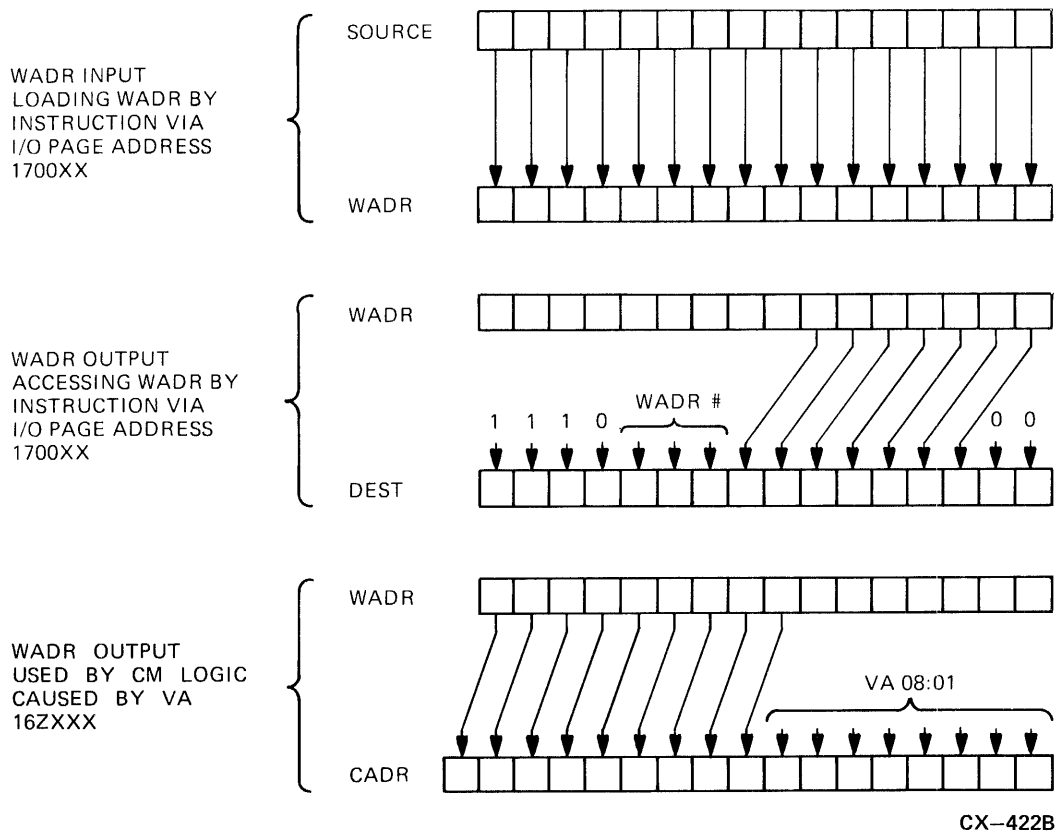

Table 8-13 I/O Page Address of Window Address Registers

| Register | Register Address | I/O Page Address |
|---|---|---|
| WADR0 | 170000 | 160000—160777 |
| WADR1 | 170002 | 161000—161777 |
| WADR2 | 170004 | 162000—162777 |
| WADR3 | 170006 | 163000—163777 |
| WADR4 | 170010 | 164000—164777 |
| WADR5 | 170012 | 165000—165777 |
| WADR6 | 170014 | 166000—166777 |
| WADR7 | 170016 | 167000—167777 |

Only 16-bit virtual addresses are shown here. However, the MMU would add bits to make 22-bit addressing.

Note that the third most significant digit of the I/O page address corresponds to the associated WADR.

Figure 8-19 WADR Gating




### 8.16.1.2 Control Window Example

The following example serves to illustrate the use of a Control Window. This explanation refers to Figure 8-20.

1. A control block is set aside in Control Memory by the P.ioc. The starting address of this control block is designated by a 16-bit pointer that represents the offset (in increments of double words) from the start of control memory.

   In this example, the pointer is $000410_8$ or an offset of $410_8$ double words from the start of Control Memory. Such an offset corresponds to Control Memory address $001020_8$, assuming word addressing started at $000000_8$. (An MMU access of the same address would use the physical 22-bit, byte-oriented address $16002040_8$.)

2. The software selects one of the WADRs to use. To illustrate, in this case it chose window set 5, WADR3.

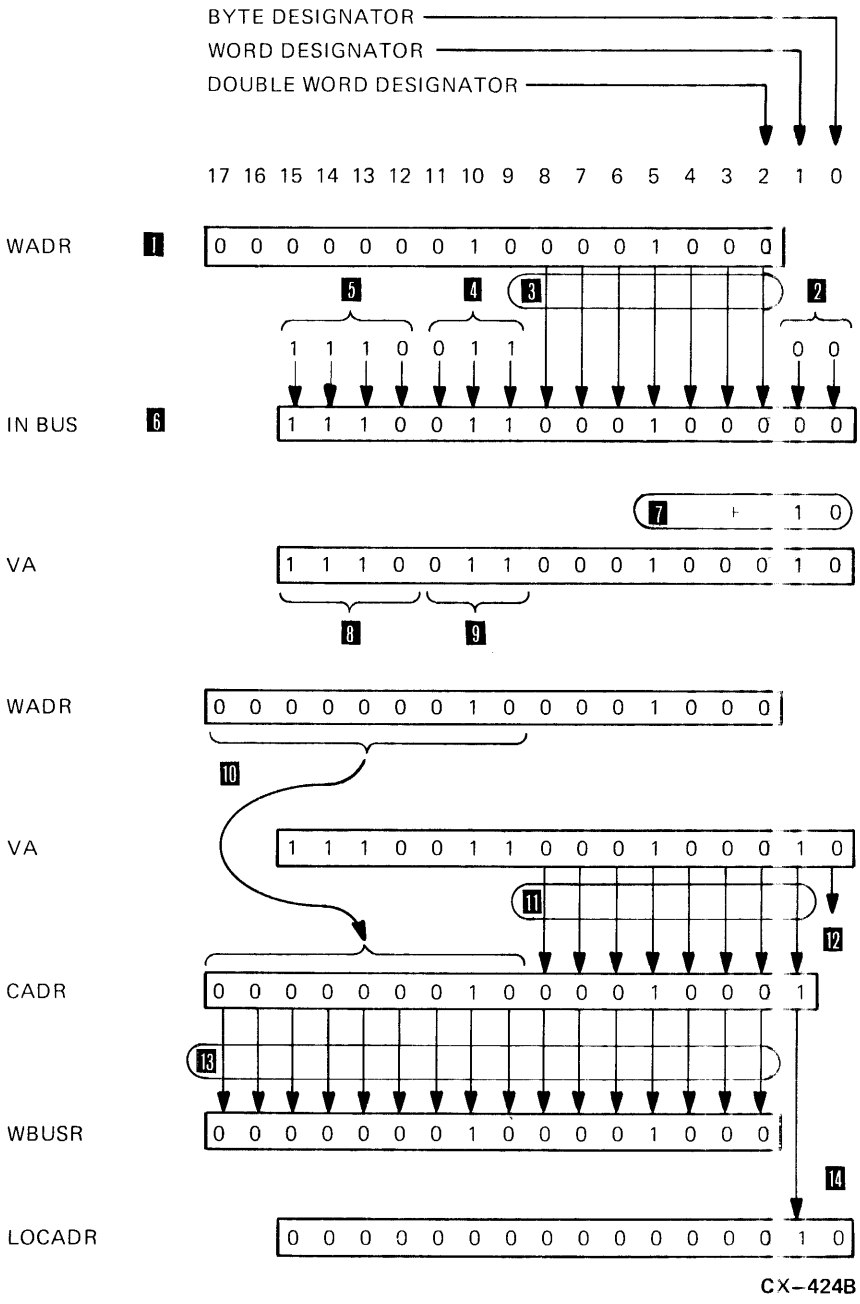
3. For selection of window set 5, the software deposits a $5_8$ in WINDR.

4. The software then deposits the pointer $000410_8$ into WADR3.

   ■ WADR3 contains a pointer $000410_8$.

   The software retrieves the contents of WADR3 with a standard PDP-11 instruction. WADR contents are gated as follows:

**Figure 8-20 Control Window Example**



2 Zeroes are gated onto IN BUS <01:00>, forcing a double word boundary.

3 Bits <06:00> of the WADR are gated onto IN BUS <08:02>. These bits shift left two places, aligning the low-order bit of the pointer with a double-word designation.

4 $011_2$ is gated onto IN BUS <11:09> because this is WADR3.

5 $1110_2$ is gated onto IN BUS <15:12> because a WADR was addressed and creates $16xxxxx_8$.

6 The result is $163040_8$. This will be used by the software as a 16-bit virtual address.

Virtual address $163040_8$ corresponds to the start of the control block in this example. If the software requires only the first control block of this word or byte, it would add nothing to this base address. Suppose the software wants to examine some information deeper in the control block.

7 For instance, to examine the third byte (low-order byte of the second word) software adds two to the virtual address.

The virtual address then contains $163042_2$, used by the software as the virtual address in a PDP-11 instruction. Because the hardware sees the virtual address in the $16xxxx_8$ range, the window logic is activated which forces use of the Control Bus. As a result, the hardware:

8 Sees $16xxxx_8$ and activates the Control Window.

9 Sees $xx3xxx_8$ and selects WADR3. (WINDR still selects window set 5.)

10 Gates WADR3 <15:07> onto CADR <16:08>.

11 Gates virtual address <08:01> onto CADR <07:00>.

12 Uses virtual address <00> as a byte designator, if a byte instruction is being executed.

CADR <16:00> ($001021_8$) on the Control Bus accesses a word in Control Memory. This is the second word of the control block.

Finally, the hardware captures CADR <16:00> and saves it in the history registers.

13 CADR <16:01> is captured in WBUSR <15:00>.

14 CADR <00> is captured in LOCADR <01>.

#### 8.16.1.3 Control Window Limitation

One limitation is placed on the Control Window feature. If the software is sequentially accessing information from a control block, and it crosses a $1000_8$ byte address boundary, the new address would have incremented the Z in virtual address $16Zxxx_8$. This would result in selecting a different WADR. So there is a $1000_8$ byte address boundary limitation. This limits the maximum size of a control block to 512 bytes assuming it starts at a $1000_8$ byte boundary; less if it starts at other than a $1000_8$ boundary.

### 8.16.2 Control Bus Interface

The Control Bus Interface block contains the Control Bus Data Transceivers and Control Bus Address Transceivers for the P.ioc (shown on the right side of Figure 8-2).

### 8.16.2.1 Control Bus Data Transceivers

The Control Bus Data Transmitter is 20 bits long:

- <15:00> is data
- Bit 16 is low byte parity
- Bit 17 is high byte parity
- Write low byte (CWRTL)
- Write high byte (CWRTH)

The transmitters are written from the OUT BUS at DOUT time if the address is between $160000_8$ and $167777_8$.

The Control Bus Data Receiver is 18 bits long:

- <15:00> is data
- Bit 16 is low byte parity
- Bit 17 is high byte parity

The receivers are enabled onto the IN BUS <15:00> at DIN time if the address is between $160000_8$ and $167777_8$.

### 8.16.2.2 Control Memory Address Transceivers

The Control Memory Address Transceiver (CADR) address inputs are controlled by the Control Memory Window logic (Section 8.16.1).

The Control Memory Address Receivers are two read-only registers:

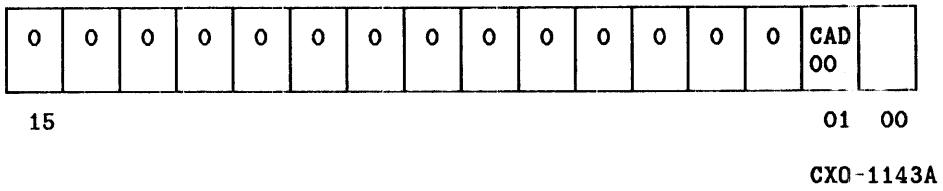
- Window Bus Register (WBUSR) — $170022_8$
- Low Control Memory Address Register (LOCADR) — $170054_8$

These two registers provide access to the address asserted on the Control Bus on each P.ioc cycle. They are updated after each Control Bus cycle by the P.ioc.

The Window Bus Register contains the high-order 16 bits of the 17-bit word address.

Figure 8-21 shows the format of the Low Control Memory Address Register. Bit <00> of this register is bit <00> of the 17-bit word address.

**Figure 8-21 Low Control Memory Address Register**



### 8.16.3 Control Bus Arbitrator and Timing

The P.ioc module contains the timing and arbitration logic for the Control Bus (shown in the lower right of Figure 8-2). For signal polarities and information about this bus, refer to Chapter 3.

The Control Memory consists of dynamic RAMs (DRAMs) which must be refreshed. To do this, the P.ioc generates an internal 66.7 KHz (15 $\mu$second period) clock to indicate when a refresh cycle is to be performed. This signal is input to the Control Bus Arbitrator as the highest priority Control Bus request. The arbitrator acknowledges the refresh cycle by the assertion of CREFR GRANT on the next available Control Bus cycle, with the same timing as any of the CGRANT n signals. The Control Memory logic thus uses the assertion of CREFR GRANT to initiate an internal refresh cycle.

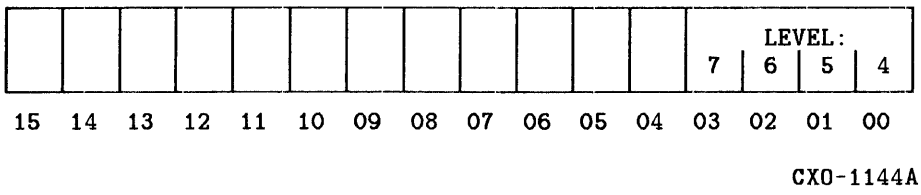
**NOTE**

**Refreshing is independent of the ENACBS bit in the PIOCSR.**

### 8.16.4 Interrupt Cycle

To execute a Control Bus Interrupt cycle, it is only necessary to set the CN IE bit in PIOCSR and then MOV the Interrupt Mask to ANY Control Memory address, using either the Control Memory Windows or Memory Management. A 1 in any bit position will queue an interrupt at that level to the P.ioc. Interrupts on more than one level may be generated by one Control Bus Interrupt cycle. The CNT IE bit will be cleared at the completion of the Interrupt cycle. Figure 8-22 shows the format of the Interrupt Mask.

**Figure 8-22 P.ioc Control Bus Interrupt Mask**




### 8.16.5 Lock Cycle

To execute a Control Bus Lock cycle, it is only necessary to set the LCK ENA bit in PIOCSR and then MOV the desired Lock Address to any destination where it may be examined to determine if the location is already locked. At the completion of the Lock cycle, the LCK ENA bit in PIOCSR is automatically cleared.

## 8.17 DATA MEMORY INTERFACE

The Data Memory Interface consists of two blocks of logic shown on the lower right of Figure 8-2:

- Data Bus Interface
- Data Bus Arbitrator and Timing

The P.ioc module contains the timing and arbitration logic for the Data Bus. In addition, the P.ioc is the lowest priority requester. The followings sections describe the Data Memory interface logic.

### 8.17.1 Data Bus Interface

The Data Bus Interface block contains the Data Bus Data Transceivers and the Data Bus Address Transceivers for the P.ioc. The following sections describe these two transceivers

#### 8.17.1.1 Data Bus Transceivers

The Data Bus Data Transmitter is 20 bits long:

- <15:00> is data
- Bit 16 is low byte parity
- Bit 17 is high byte parity
- Write low byte (DWRTL)
- Write high byte (DWRTH)

The transmitters are written from the OUT BUS at DOUT time if the address is between $14000000_8$ and $15777777_8$.

The Data Bus Data Receiver is 18 bits long:

- <15:00> is data
- Bit 16 is low byte parity
- Bit 17 is high byte parity

The receivers are written onto the IN BUS <15:00> at DIN time if the address is between $14000000_8$ and $15777777_8$.

#### 8.17.1.2 Data Bus Address Transceivers

The Data Bus deals with word addresses and has 18 significant address bits. This 18 bit address is handled differently in the Data Bus Address Transmitters than in the Data Bus Address Receivers.

##### 8.17.1.2.1 Data Bus Address Transmitters

The address on the OUT BUS <18:01> is clocked into the Data Bus Address Transmitters <17:00> when the address is between $14000000_8$ and $15777777_8$. Notice the shift of one bit to the right to make the address a word reference.

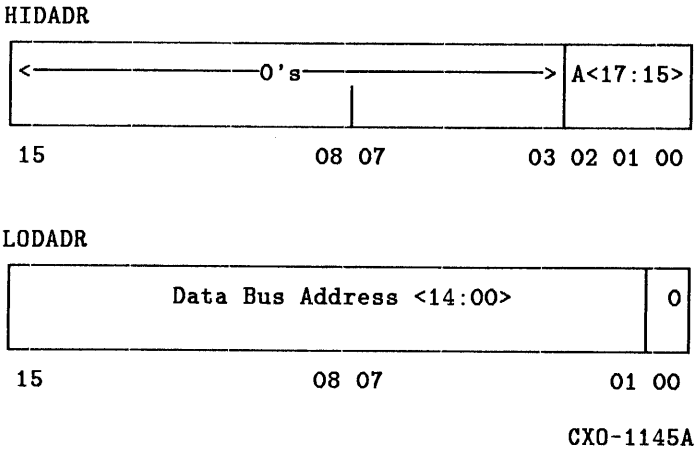
##### 8.17.1.2.2 Data Bus Address Receivers

The Data Bus Address Receivers are two read-only registers. These registers are:

- High Data Bus Address Register (HIDADR) located at $17770052_8$
- Low Data Bus Address Register (LODADR) located at $17770050_8$

After each Data Bus cycle by the P.ioc, the registers are updated with the 18-bit word address. Figure 8-23 shows the format of the two receiver registers.

Figure 8-23 Data Bus Address Receivers




### 8.17.2 Data Bus Arbitrator and Timing

The P.ioc module contains the timing and arbitration logic for the Data Bus. For signal polarities and information about this bus, refer to Chapter 3.

The P.ioc has the lowest priority Data Bus requester.

## 8.18 I/0 PAGE ADDRESS UTILIZATION

Table 8-14 is a list of the I/O page addresses assigned in the P.ioc.

Table 8-14 I/O Page Address Utilization

| Physical Address | Function |
|---|---|
| 17777776 | Processor Status Word |
| 17777656 | User PAR 7 |
| 17777654 | User PAR 6 |
| 17777652 | User PAR 5 |
| 17777650 | User PAR 4 |
| 17777646 | User PAR 3 |
| 17777644 | User PAR 2 |
| 17777642 | User PAR 1 |
| 17777640 | User PAR 0 |

Table 8-14 (Cont.) I/O Page Address Utilization

| Physical Address | Function |
|---|---|
| 17777616 | User PDR 7 |
| 17777614 | User PDR 6 |
| 17777612 | User PDR 5 |
| 17777610 | User PDR 4 |
| 17777606 | User PDR 3 |
| 17777604 | User PDR 2 |
| 17777602 | User PDR 1 |
| 17777600 | User PDR 0 |
| 17777576 | Memory Management Status Register 2 |
| 17777574 | Memory Management Status Register 1 |
| 17777572 | Memory Management Status Register 0 |
| 17777566 | Console Xmtr Data Buffer Register |
| 17777564 | Console Xmtr Status Register |
| 17777562 | Console Rcvr Data Buffer Register |
| 17777560 | Console Rcvr Status Register |
| 17777536 | TU58 2 Xmtr Data Buffer Register |
| 17777534 | TU58 2 Xmtr Status Register |
| 17777532 | TU58 2 Rcvr Data Buffer Register |
| 17777530 | TU58 2 Rcvr Status Register |
| 17777526 | TU58 1 Xmtr Data Buffer Register |
| 17777524 | TU58 1 Xmtr Status Register |
| 17777522 | TU58 1 Rcvr Data Buffer Register |
| 17777520 | TU58 1 Rcvr Status Register |
| 17776776 to 17773000 | BootStrap PROM |
| 17772516 | Memory Management Status Register 3 |
| 17772356 | Kernel PAR 7 |
| 17772354 | Kernel PAR 6 |
| 17772352 | Kernel PAR 5 |
| 17772350 | Kernel PAR 4 |
| 17772346 | Kernel PAR 3 |
| 17772344 | Kernel PAR 2 |
| 17772342 | Kernel PAR 1 |
| 17772340 | Kernel PAR 0 |

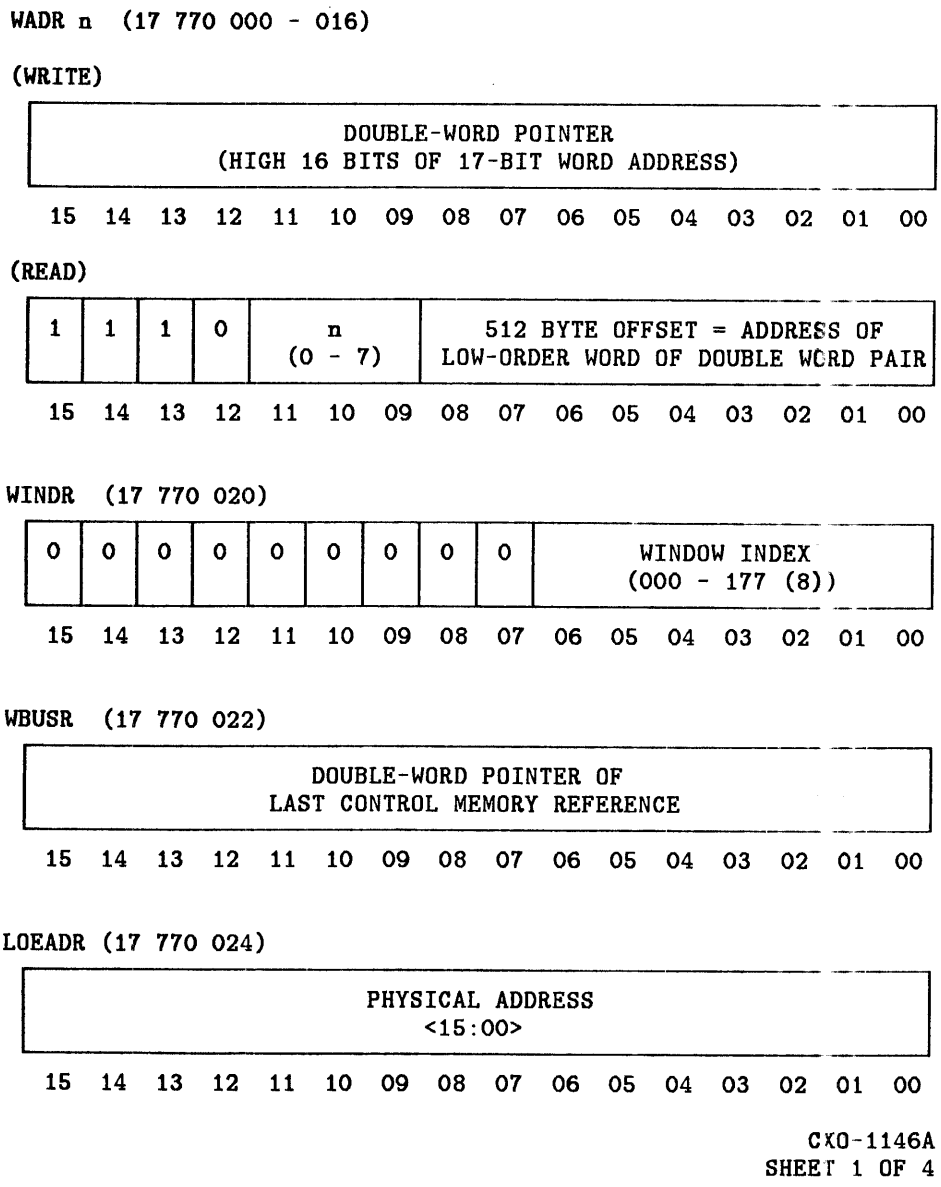
Table 8-14 (Cont.) I/O Page Address Utilization

| Physical Address | Function |
|---|---|
| 17772316 | Kernel PDR 7 |
| 17772314 | Kernel PDR 6 |
| 17772312 | Kernel PDR 5 |
| 17772310 | Kernel PDR 4 |
| 17772306 | Kernel PDR 3 |
| 17772304 | Kernel PDR 2 |
| 17772302 | Kernel PDR 1 |
| 17772300 | Kernel PDR 0 |
| 17770056 | Serial Number Register |
| 17770054 | Control Memory Low Address Register |
| 17770052 | Data Memory High Address Register |
| 17770050 | Data Memory Low Address Register |
| 17770046 | K Status Register |
| 17770044 | K Init Register |
| 17770042 | Switch/Display Register |
| 17770040 | Pio Control and Status Register |
| 17770026 | High Error Address Register |
| 17770024 | Low Error Address Register |
| 17770022 | Window Bus Register |
| 17770020 | Window Index Register |
| 17770016 | Window Address Register 7 |
| 17770014 | Window Address Register 6 |
| 17770012 | Window Address Register 5 |
| 17770010 | Window Address Register 4 |
| 17770006 | Window Address Register 3 |
| 17770004 | Window Address Register 2 |
| 17770002 | Window Address Register 1 |
| 17770000 | Window Address Register 0 |
| 17767776 | Control Memory Windows |
| 17760000 | |

## 8.19 I/O PAGE REGISTER FORMAT SUMMARY

Figure 8-24 is a summary of the I/O page registers on the P.ioc.

Figure 8-24 I/O Page Register Format Summary




(Continued on next page)

Figure 8-24 (Cont.) I/O Page Register Format Summary

HIEADR (17 770 026)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PHYSICAL ADDRESS <21:16> | | | | | |

PIOCSR (17 770 040)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRM ENA | 0 | 0 | 0 | SWP BRD | SWP BNK | SEL P1 | SEL P0 | ENA CBS | IO CLR | HIP TST | LOP TST | LED | NMA | CNT IE | LCK |

SWDR (17 770 042)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | SW 3 | SW 2 | SW 1 | SW 0 | DIA OK | C/D NXM | INH PAR | LMP 4 | LMP 3 | LMP 2 | LMP 1 | LMP 0 |

KINR (17 770 044)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K INIT <7:0> | | | | | | | | SPARE | | | | KST <3:0> | | | |

KSTR (17 770 046)

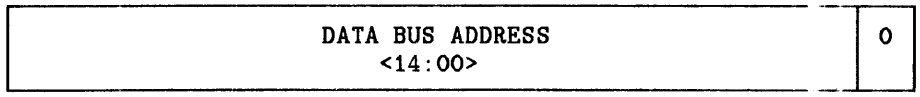
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SP | STATUS <6:0> | | | | | | |

CXO-1146A
SHEET 2 OF 4

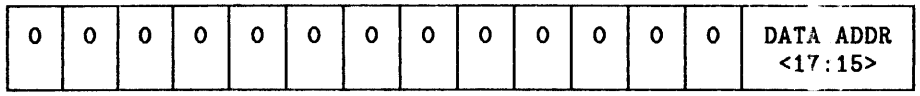
(Continued on next page)

Figure 8-24 (Cont.) I/O Page Register Format Summary

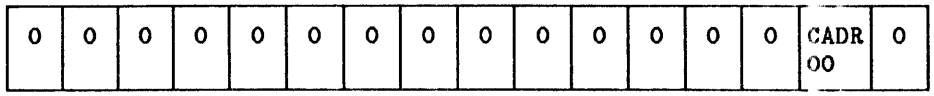
LODADR (17 770 050)



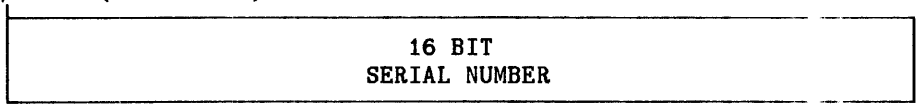

HIDADR (17 770 052)



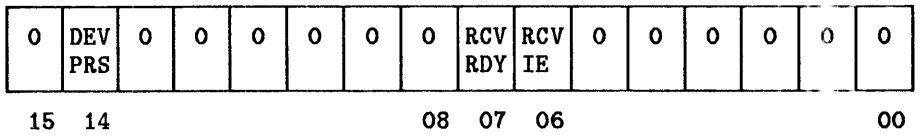

LOCADR (17 770 054)




S/NREG (17 770 056)




TU58-1 RCSR (17 777 520)
TU58-2 RCSR (17 777 530)
TERM RCSR (17 777 560)

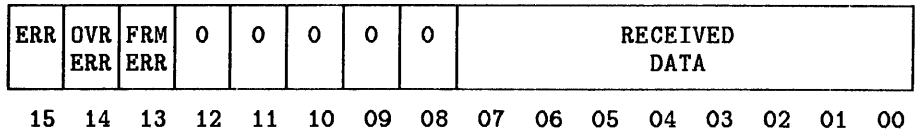



CXO-1146A
SHEET 3 OF 4

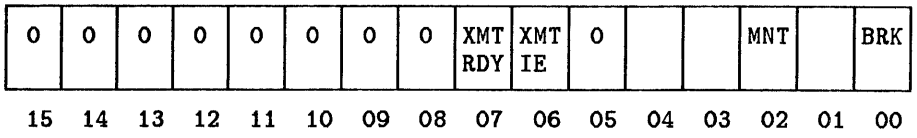
(Continued on next page)

Figure 8-24 (Cont.) I/O Page Register Format Summary

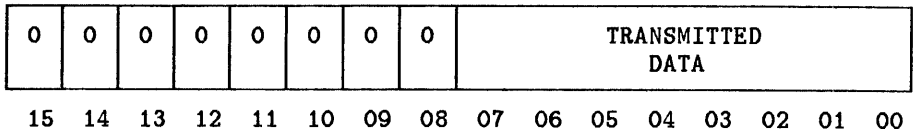
TU58-1 RBUF (17 777 522)
TU58-2 RBUF (17 777 532)
TERM RBUF (17 777 562)




TU58-1 XCSR (17 777 524)
TU58-2 XCSR (17 777 534)
TERM XCSR (17 777 564)




TU58-1 XBUF (17 777 526)
TU58-2 XBUF (17 777 536)
TERM XBUF (17 777 566)




CXO-1146A
SHEET 4 OF 4

# CHAPTER 9
# HSC70 INPUT/OUTPUT CONTROL PROCESSOR MODULE

## 9.1 INTRODUCTION

The HSC70 Input/Output Control Processor Module (J-11 P.ioj) is an extended hex module that contains the following logic:

- Eight LEDs for status indications
- A DCJ11 (J-11) processor (with integral memory management)
- Timing and interrupt logic for the J-11
- A 4-Kword instruction cache
- An interface to Program Memory, with integral direct memory access (DMA) and interrupt capability
- A bootstrap PROM
- I/O page windows into Control Memory
- Control, timing, and arbitration for the HSC70 Data Bus and Control Bus
- An RS-423 / DL-11 compatible interface for a console terminal for the J-11
- Pio Control and Status Register
- An interface to the HSC70 Operator Control Panel (Switch/Display Register)
- K Init and Status Registers
- Error Address Register
- Serial Number Register
- Control Memory and Data Memory interfaces

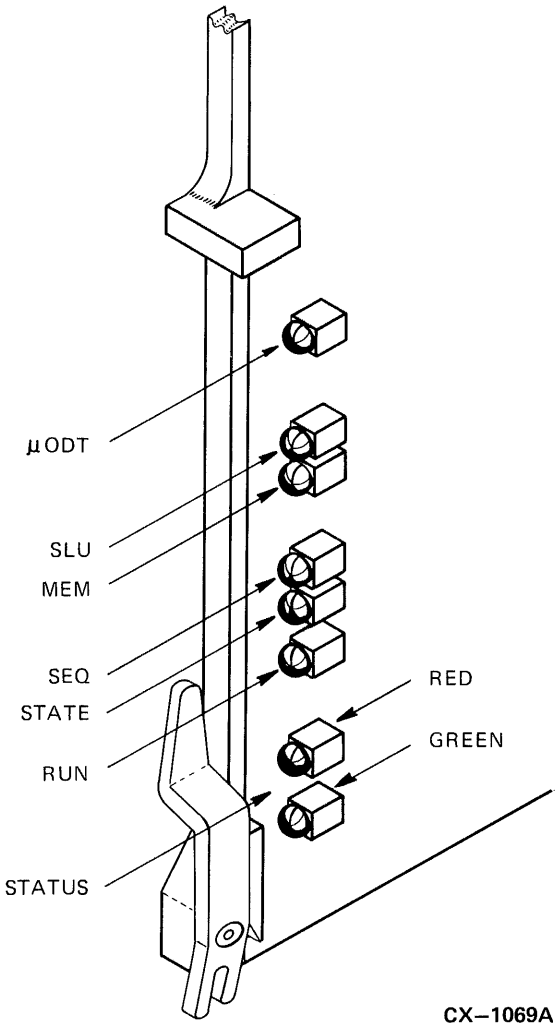
## 9.2 LED INDICATORS

The P.ioj contains eight LED indicators (Figure 9-1): one red, one green, and six yellow. These LEDs are described in the following sections.

### 9.2.1 P.ioj Diagnostic OK LEDs

The red LED (D7) and green LED (D8) are used to indicate whether the P.ioj successfully completed all of its initialization diagnostics. The module powers up with the red LED on and the green LED off. Upon successful completion of the power up diagnostics, the red LED is turned off and the green LED is turned on. This LED is controlled by a bit in the Switch/Display Register (Section 9.11).

Figure 9-1 P.ioj LEDs



### 9.2.2 RUN LED

A yellow LED, D6 (adjacent to the red Diagnostic OK LED), blinks once for each PDP-11 instruction fetch cycle. This is analogous to the RUN light on standard LSI-11 processor systems. When the J-11 is running, this LED will be illuminated at about half-brilliance (compared to the other LEDs on the module).

### 9.2.3 State LED

This yellow LED (D5) is controlled by bit <03> of the Pio Control and Status Register (PIOCSR) and is connected in parallel with the State indicator on the Operator Control Panel.

### 9.2.4 Status LEDs

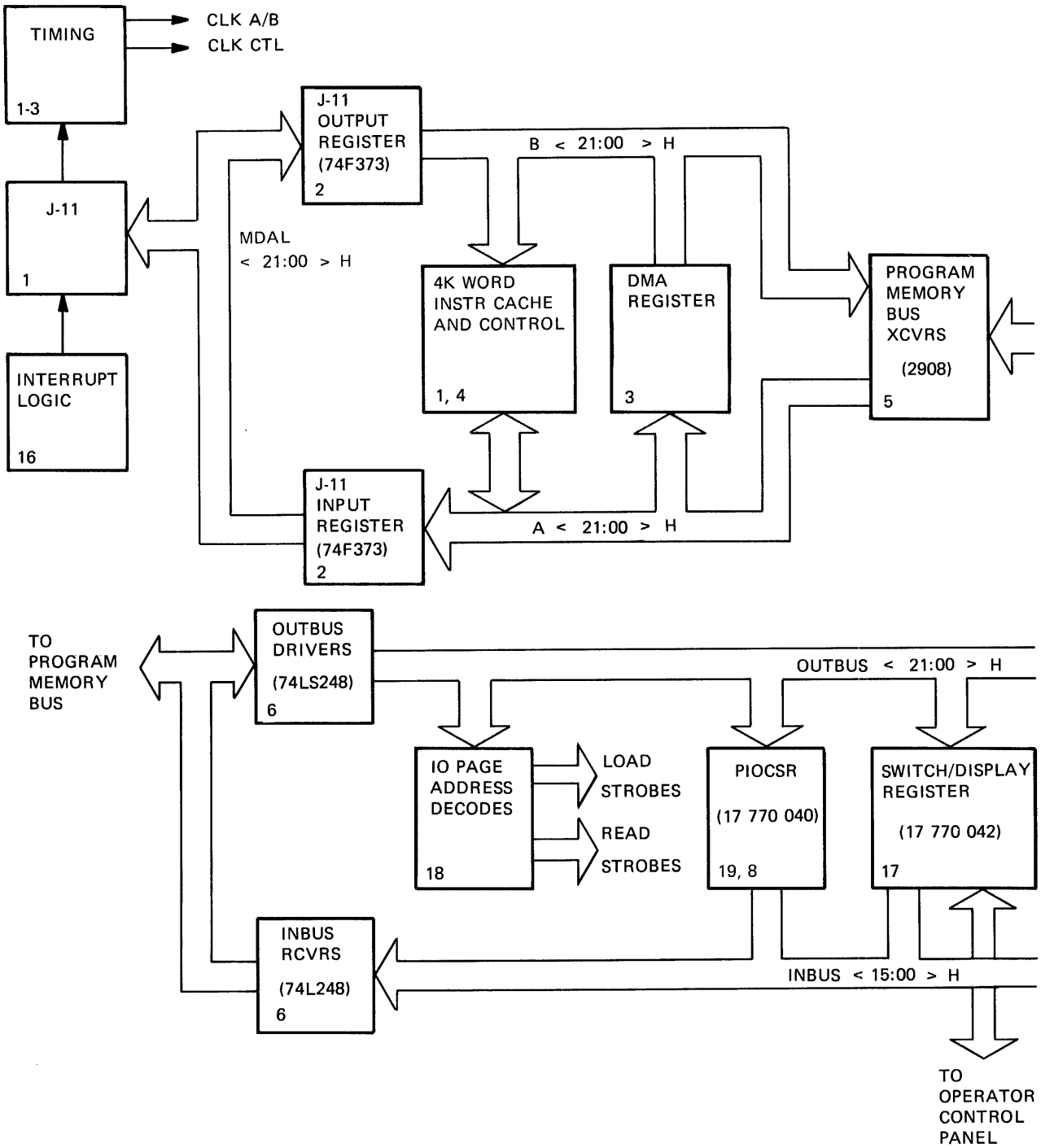
The four yellow Status LEDs (D1—D4) are controlled by the J-11 internal microcode. All four LEDs are initially on. As soon as the J-11 starts operating, it turns off the micro-ODT LED (D1). After going through several microcode steps, LED D4 is turned off, indicating that the J-11 is sequencing and has reached this point in the microcode. It then performs several memory operations and, if successful, turns off the MEMORY OK LED, D3. Finally, the J-11 tests whether the console terminal port is functional and, if so, turns off LED D2. The J-11 then proceeds to the bootstrap.

In addition, any time the J-11 is executing $\mu$ODT, LED D1 is on.

## 9.3 HSC70 INPUT/OUTPUT CONTROL PROCESSOR MODULE BLOCK DIAGRAM

Figure 9-2 is a block diagram of the P.ioj showing the principal hardware elements and data paths. Each of these elements will be described in the following sections.

Figure 9-2 HSC70 Input/Output Control Processor Module Block Diagram



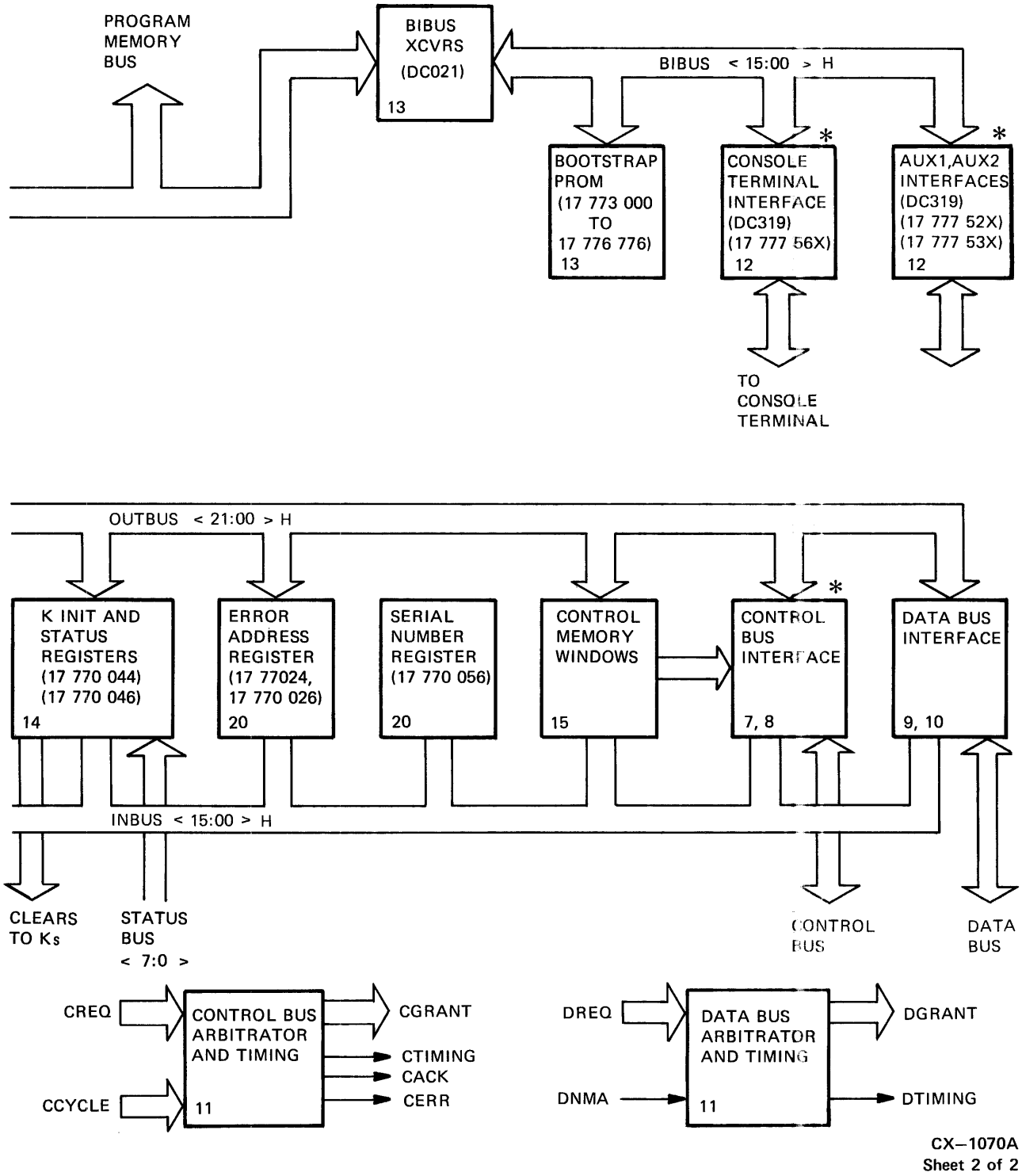

NOTES:

1. * THESE BLOCKS HAVE INPUTS TO THE INTERRUPT LOGIC
2. NUMBERS IN LOWER LEFT HAND CORNER OF BOXS REFER TO PAGE NUMBERS IN THE MODULE SCHEMATIC

CX–1070A
Sheet 1 of 2

(Continued on next page)

Figure 9-2 (Cont.) HSC70 Input/Output Control Processor Module Block Diagram




CX–1070A
Sheet 2 of 2

## 9.4 P.ioj CONTROL LOGIC

The P.ioj control logic controls the operation of the P.ioj. The control logic is located in the upper-left-hand corner of Figure 9-2. Figure 9-3 shows a detailed block of this logic. The block contains:

- DCJ11 Microprocessor
- Cache Data Path and Cache Memory
- State Sequencer
- Input/Output Control Logic
- Program Memory Input/Output Transceivers

Each of these blocks is described in the following sections.

**NOTE**

**For a detailed explanation of this logic, refer to the** *KDJ11-A CPU Module User's Guide* **(EK-KDJ1A-UG).**

### 9.4.1 DCJ11 Microprocessor

The DCJ11 Microprocessor has the complete PDP-11/70 instruction set plus PDP-11/44-11/70 memory management for addressing up to two-million words of memory. The complete processor is implemented in one 60-pin hybrid package. Figure 9-4 is a block diagram of the DCJ11 Microprocessor. It performs the following functions:

- Executes the PDP-11 instruction set
- Controls the memory management
- Supports the console micro-ODT
- Initiates all the data transfers and operations

**NOTE**

**For a detailed explanation of the DCJ11 Microprocessor, refer to the** *DCJ11 Microprocessor User's Guide* **(EK-DCJ11-UG).**

Besides the standard PDP-11 registers, the DCJ11 contains special registers. The way they are implemented in the P.ioj is described in the following sections.

#### 9.4.1.1 Cache Control Register

The Cache Control Register (CCR) contains information which is used to control the operation of Cache Memory. It is located at $17777746_8$. Figure 9-5 shows the format of the register, and Table 9-1 describes the bits. Unless otherwise noted, bits in this register are read/write.

Figure 9-3 P.ioj Control Logic Functional Block Diagram

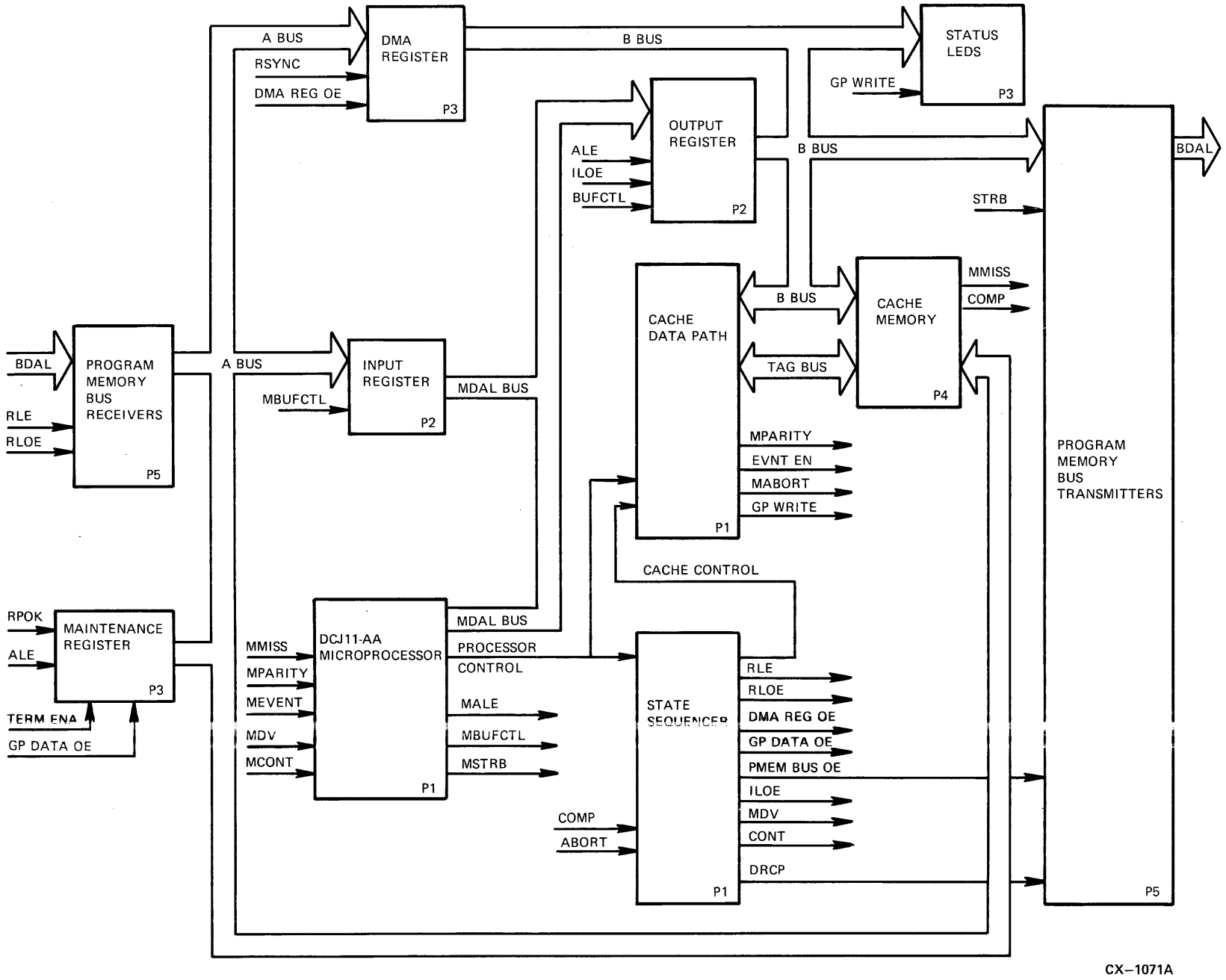


Figure 9-4 DCJ11 Block Diagram

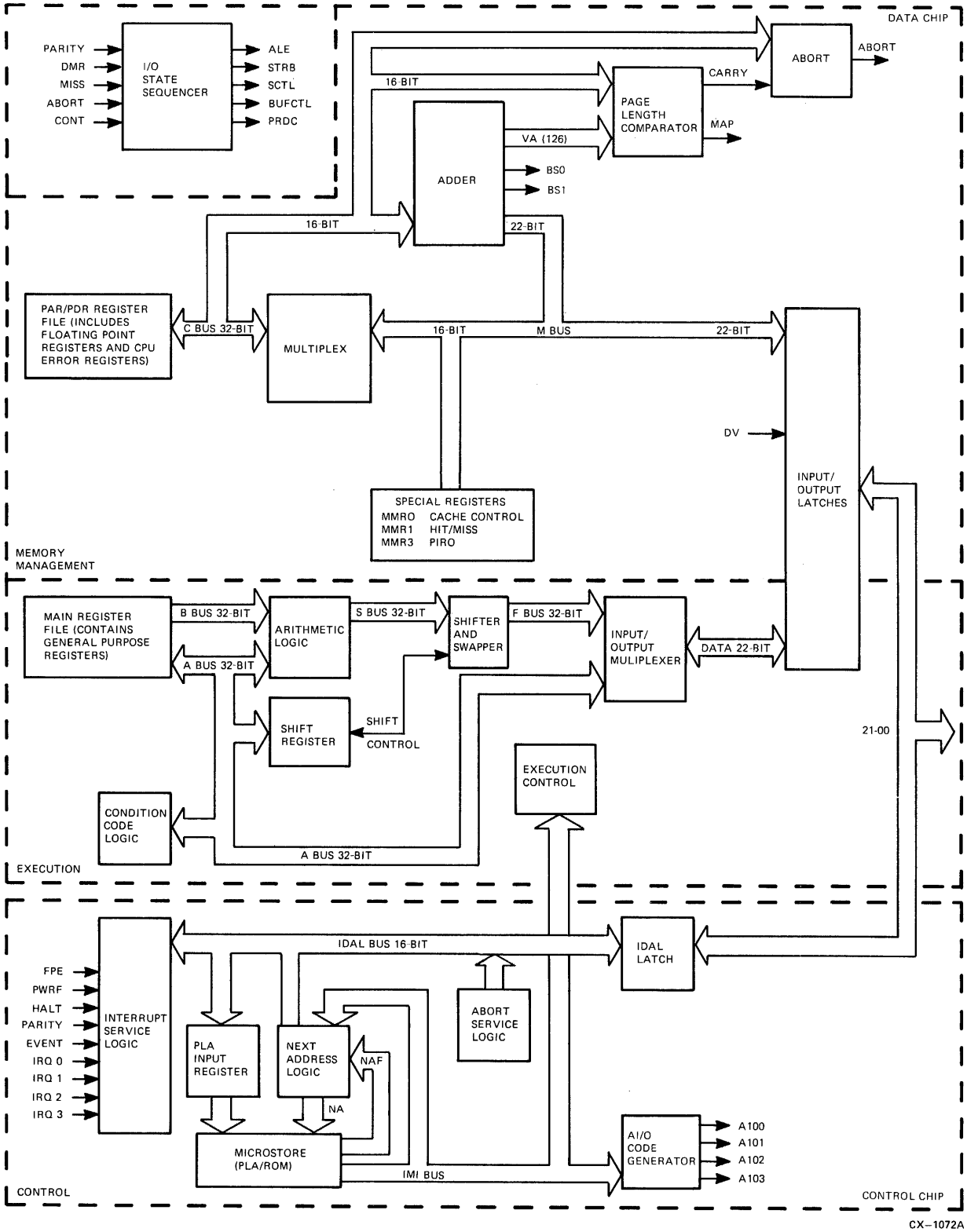


Figure 9-5 Cache Control Register

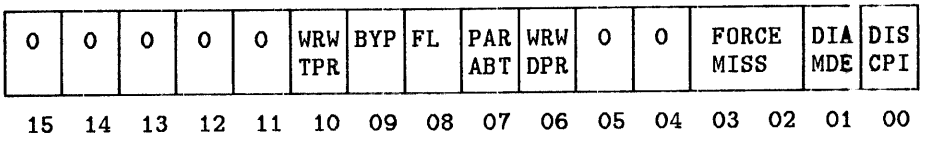



Table 9-1 Cache Control Register Bit Description

| Bits | Description |
|---|---|
| 15:11 | These read-only bits are always 0. |
| 10 | When set, WRITE WRONG TAG PARITY, this bit causes incorrect tag parity to be written on cache writes. A cache tag parity error will occur on the next access to that location. |
| 09 | When set, UNCONDITIONAL CACHE BYPASS, this bit forces all references to memory by the J-11 to go to Program Memory. Read or write hits will result in invalidation of those locations in the cache; misses will not change the contents. |
| 08 | When set, FLUSH CACHE, this bit causes the entire contents of the cache to become invalid. This will take approximately 1.7 milliseconds. Note that while the flush is taking place, DMA is not serviced. Resetting this bit causes no action. This write-only bit reads as 0. |
| 07 | If this bit, PARITY ERROR ABORT, is set and a cache parity error occurs, the cache reference aborts through vector $114_8$. The cache is updated. |
| 06 | When set, WRITE WRONG DATA PARITY, this bit causes incorrect data parity on both data bytes to be written on cache writes. A cache data parity error will occur on the next access to that location. |
| 05:04 | These read/write bits are not used. |
| 03:02 | When set, FORCE CACHE MISS, either bit reports the J-11 memory references as cache misses and disables cache parity checking. The cache is essentially removed from the system. |
| 01 | When set, DIAGNOSTIC MODE causes all non-bypass and non-forced-miss word writes to allocate the cache, regardless of nonexistent memory errors. |
| 00 | This bit, DISABLE CACHE PARITY INTERRUPT, is used in conjunction with CCR<07>. If CCR<07> is cleared, this bit determines what action is taken when a cache parity error occurs. If this bit is cleared and a cache parity error occurs, the cache is updated and an interrupt through vector $114_8$ is generated. If this bit is set and a cache parity error occurs, the cache is updated but no abort or interrupt is generated. |

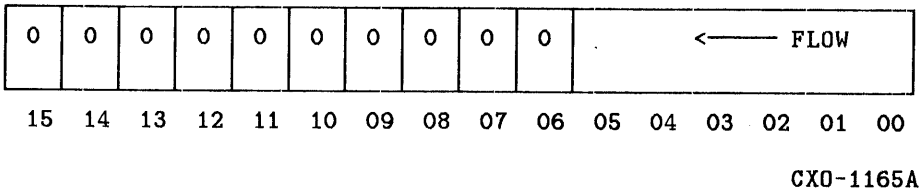
This register is cleared on power up or by a console start. It is not affected by a RESET instruction.

A cache parity interrupt or abort will only occur if a previous cache parity abort has been cleared. Specifically, MSER<07:05> must be cleared before new cache parity errors are recognized.

### 9.4.1.2 Cache Hit/Miss Register

The Cache Hit/Miss Register (CHMR) indicates whether the six most recent CPU memory references resulted in cache hits or cache misses. Bits enter from the bit <00> position and shift left. A one indicates a cache hit, a zero a cache miss. Figure 9-6 shows the format of the register. The register is located at $17777752_8$ and is read-only.

**Figure 9-6 Cache Hit/Miss Register**




The value of this register at power up is undefined. The register is not affected by console start or a RESET instruction.

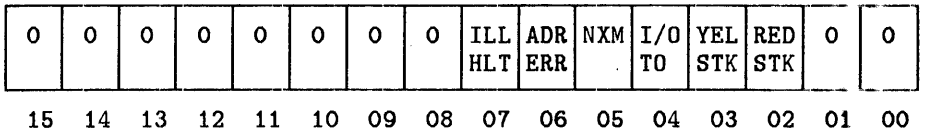
### 9.4.1.3 CPU Error Register

The CPU Error Register identifies the reason for any abort or trap through location $004_8$. It is located at $17777766_8$ and is read-only. Figure 9-7 shows the format, and Table 9-2 describes the bits.

**Table 9-2 CPU Error Register**

| Bits | Description |
|---|---|
| 15:08 | These bits are always 0. |
| 07 | When set, ILLEGAL HALT, this bit indicates that execution of a HALT instruction was attempted while in user mode, supervisor mode, or in kernel mode while the Operator Control Panel SECURE/ENABLE switch was in the SECURE position. |
| 06 | When set, ADDRESS ERROR, this bit indicates that word access to an odd byte or an instruction fetch from an internal register was attempted. |
| 05 | When set, NON-EXISTENT MEMORY, this bit indicates that a reference to Program, Control, or Data memory timed out. |
| 04 | When set, I/O PAGE TIMEOUT, this bit indicates that a reference to the I/O page timed out. Note that a timeout to Control Memory while using the Control Memory Windows will set this bit instead of nonexistent memory (NXM). |
| 03 | When set, YELLOW STACK VIOLATION, this bit indicates a yellow zone stack overflow trap occurred. |
| 02 | When set, RED STACK VIOLATION, this bit indicates a push to the kernel stack aborted while servicing an abort, interrupt, or TRAP instruction. |
| 01:00 | These bits are always 0. |

Figure 9-7 CPU Error Register




This register is cleared by power up, console start, and when it is written. It is not affected by a RESET instruction.

### 9.4.1.4 Programmable Interrupt Request Register

The Programmable Interrupt Request (PIRQ) Register provides seven levels of software interrupt capability. The software queues an interrupt request by setting one of the PIR bits (<15:9>) in the register which correspond to interrupt priority levels 7 through 1, respectively. Bits <7:5> and <3:1> are set by the DCJ11 to the encoded value of the highest pending request. When the program interrupt request is granted, the processor traps through a vector at virtual location $240_8$. It is the responsibility of the interrupt service routine to clear the appropriate bit in the PIRQ before exiting. The format of PIRQ is shown in Figure 9-8. Table 9-3 describes the bits in the PIRQ.

This register is cleared by power up, by console start, and by RESET instructions. The PIRQ is located at $17777772_8$.

Figure 9-8 Programmable Interrupt Request Register

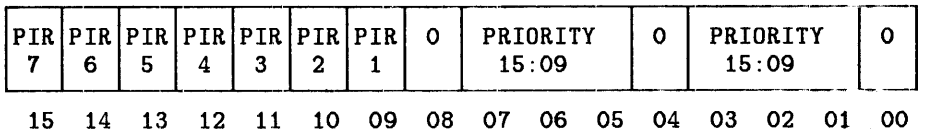



Table 9-3 Program Interrupt Request Register Bit Description

| Bits | Description |
|---|---|
| 15:09 | When set, PROGRAM INTERRUPT REQUEST <7:1>, these read/write bits queue a program interrupt request at levels 7 through 1. |
| 08 | This read-only bit is always 0. |
| 07:05 | These read-only bits are set by hardware to the encoded value of the highest pending request. |
| 04 | This read-only bit is always 0. |
| 03:01 | These read-only bits are set by hardware to the encoded value of the highest pending request. |
| 00 | This read-only bit is always 0. |

#### 9.4.1.5 Memory Management Status Registers

Aborts generated by the Memory Management Unit (MMU) are vectored through kernel virtual location $250_8$. Memory management registers 0, 1, 2, and 3 are used to determine why the abort occurred and to allow for easy program restarting. Note that an abort to a location which is in itself an invalid address will cause another abort. The following sections describe these four memory management registers.

##### 9.4.1.5.1 Memory Management Register 0

Memory Management Register 0 (MMR0) contains error flags, the page number whose reference caused the abort, and various other flags. The register is located at $17777572_8$. The format of MMR0 is shown in Figure 9-9. Table 9-4 describes the bits in the MMR0.

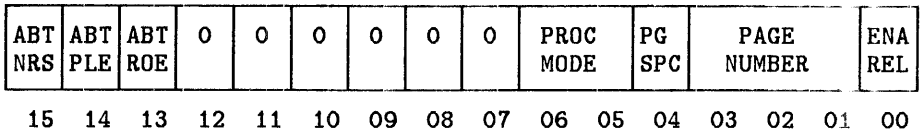
**Table 9-4 Memory Management Register 0 Bit Description**

| Bits | Description |
|---|---|
| 15 | This bit, ABORT NON-RESIDENT, is set by attempting to access a page with an Access Control Field key equal to 0 or 2. It it also set by attempting to use memory relocation with a processor mode of 2. |
| 14 | This bit, ABORT PAGE LENGTH ERROR, is set by attempting to access a location in a page with a block number (virtual address bits <12:06>) that is outside the area authorized by the Page Length Field of the Page Descriptor Register for that page. |
| 13 | This bit, ABORT READ-ONLY ERROR, is set by attempting to write in a read-only page. Read-only pages have access keys of 1. |
| 12:07 | These read-only bits are always 0. |
| 06:05 | These bits, PROCESSOR MODE, indicate the processor mode (kernel=00, supervisor=01, user=11, illegal=10) associated with the page causing the abort. If the illegal mode is specified, an abort is generated and MMR0 <15> is set. |
| 04 | This bit, PAGE SPACE, indicates the address space associated with the page causing the abort (I space = 0, D space = 1). |
| 03:01 | These bits, PAGE NUMBER, indicate the page number of the page causing the abort. |
| 00 | When set, ENABLE RELOCATION, this bit causes all addresses to be relocated. When cleared, memory management is inoperative and addresses are not relocated. |

Note: MMR0 <15:13> can be set by an explicit write; however, such an action does not cause an abort. Whether set explicitly or by an abort, MMR0 <15:13> cause the memory management logic to freeze the contents of MMR0 <06:01> Memory Management Register 1 and Memory Management Register 2. These registers remain frozen until MMR0 <15:13> are cleared by an explicit write or an initialization sequence.

MMR0 <15:13,00> are cleared by power up, by a console start, and by a RESET instruction. MMR0 <06:01> are undefined at power up and reflect the last unfrozen relocation operation after a console start or RESET instruction.

Figure 9-9 Memory Management Register 0

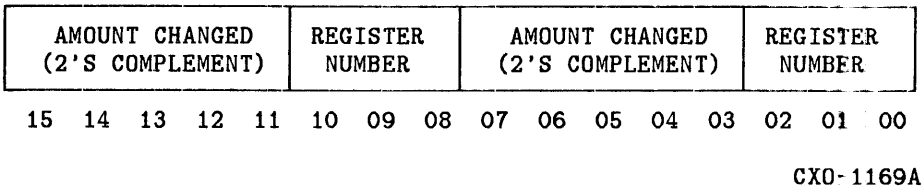



### 9.4.1.5.2 Memory Management Register 1

Memory Management Register 1 (MMR1) records any autoincrement/autodecrement of the general purpose registers. This register supplies information needed to recover from a Memory Management abort by allowing programmed register back up.

MMR1 is located at $17777574_8$ and is read-only. Its state at power up is undefined. Figure 9-10 shows the format of the MMR1.

Figure 9-10 Memory Management Register 1




If MMR0 <15:13> are non-zero, the contents of MMR1 are frozen. If MMR0 <15:13> are zero, then receipt of the signal MPRDC-L (start of macroinstruction) or an RDF microinstruction clears the register. Thereafter, selection of the half (bits <7:0> versus bits <15:8>) of MMR1 which is updated, when it is not frozen, toggles between the two halves, starting with the low half (bits <7:0>). Therefore, source operand register changes will always be recorded in bits <7:0>; however, destination operand register changes may be recorded in either half of MMR1 (depending on the mode of the source operand and the instruction type).

MMR1 is undefined at power up.

### 9.4.1.5.3 Memory Management Register 2

Memory Management Register 2 (MMR2) is loaded with the current 16-bit virtual address at the beginning of instruction fetch. MMR2, located at $17777576_8$, is a read-only register. Its state at power up is undefined.

#### 9.4.1.5.4 Memory Management Register 3

Memory Management Register 3 (MMR3) performs the following functions:

- Enables or disables the use of D space page address registers and page descriptor registers
- Enables or disables the use of 22-bit mapping
- Controls data on the time-multiplexed input MAP (pin 19 of the DCJ11)

MMR3, located at $17772516_8$, is cleared at power up, by a console start, and by a RESET instruction. Figure 9-11 shows the format of the MMR3. Table 9-5 describes the bits in this register.

**Figure 9-11 Memory Management Register 3**

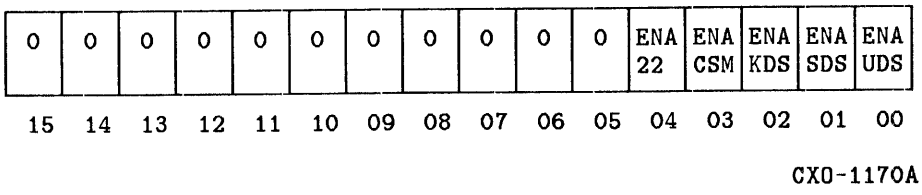



**Table 9-5 Memory Management Register 3 Bit Description**

| Bits | Description |
|---|---|
| 15:06 | These read-only bits are always 0. |
| 05 | This read-write bit is not used. |
| 04 | When set, ENABLE 22-BIT MAPPING, this bit selects 22-bit relocation when MMR0 <00> is set. When cleared, 18-bit relocation is selected when MMR0 <00> is set. |
| 03 | When set, ENABLE CSM INSTRUCTION, this bit allows recognition of the Call Supervisor Mode instruction. |
| 02:00 | When set, ENABLE (KERNEL, SUPERVISOR, USER) DATA SPACE, this bit enables Data Space mapping for the corresponding mode. |

#### 9.4.1.6 Micro-ODT

Micro-ODT, as currently implemented in the J-11 microcode, accepts 22-bit physical addresses. Therefore, all HSC70 memory is accessible using micro-ODT. Note that all I/O page references $17760000_8$ through $17777777_8$ require that all 22 bits be specified. However, leading 0s for either address or data are not necessary.

### 9.4.2 Input Register and Output Register

The Input Register and Output Register interface the A BUS and B BUS to the MDAL BUS. These registers allow the DCJ11 to transmit addressing and data information on the B BUS to the Program Memory Bus and receive data on the A BUS from the Program Memory Bus.

### 9.4.3 State Sequencer

The State Sequencer gate array generates most of the control and timing signals for the P.ioj control logic. Data transfers over the A BUS and B BUS are controlled by the array. Furthermore, the Program Memory Bus interface is sequenced by the array.

The State Sequencer runs only during external bus access. When the State Sequencer is inactive, the data paths are set up to do a high speed cache read operation. If the J-11 makes an external bus cycle, the State Sequencer runs and manipulates the P.ioj control logic and Program Memory Bus in order to perform external bus cycles:

- Memory input/output
- Interrupt vector reads
- Register input/output
- General purpose input/output
- Direct Memory Access arbitration

### 9.4.4 Cache Data Path

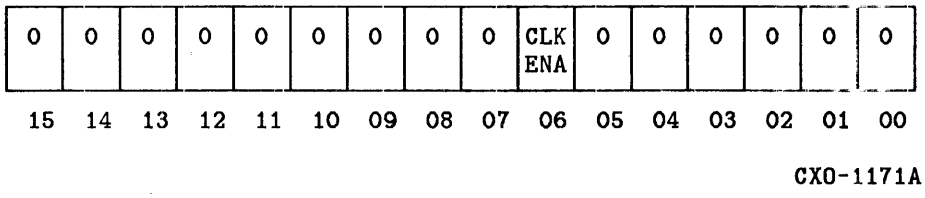
The Cache Data Path is a multifunction gate array that controls the 8 Kbyte direct map Cache Memory. It generates B BUS bits <21:13> as TAG data for the cache write transaction. Parity for the TAG data is generated, predicted, and checked by the gate array. The gate array also contains the flush address counter used to clear or flush the cache memory.

The Cache Data Path contains a Cache Control Register which is a shadow copy of the one in the DCJ11 (Section 9.4.1.1). It also contains the Clock Register and the Memory System Error Register which are described in the following sections.

#### 9.4.4.1 Clock Register

The Clock Register, CLKREG, allows software to enable the 50 Hz clock interrupt. The register, located at $17777546_8$, has the format as shown in Figure 9-12. Bit <06>, CLOCK ENABLE, is a read/write bit which, when set, enables the 50 Hz line clock interrupt. This bit is clear on power-up. All other bits are read-only and are zero.

Figure 9-12 Clock Register




#### 9.4.4.2 Memory System Error Register

The Memory System Error Register (MSER) is located at $17777744_8$. Figure 9-13 shows the format of the register, and Table 9-6 describes the bits. The entire register is read-only.

This register is cleared by any MSER write reference. It is also cleared on power up or by a console start (for example, by using Micro-ODT). It is unaffected by a RESET instruction.

Figure 9-13 Memory System Error Register

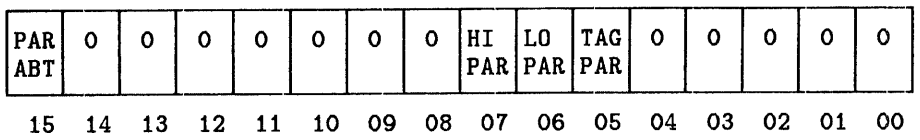



Table 9-6 Memory System Error Register Bit Description

| Bits | Description |
|---|---|
| 15 | This bit, PARITY ABORT, is set if a cache or memory parity error aborts an instruction. |
| 14:08 | These bits are always 0. |
| 07 | This bit, HIGH BYTE PARITY ERROR, is set if Cache Control Register (CCR) bit <07> is set and a cache parity error occurs in the high data byte. |
| 06 | This bit, LOW BYTE PARITY ERROR, is set if CCR<07> is set and a cache parity error occurs in the low data byte. |
| 05 | This bit, TAG PARITY ERROR, is set if CCR<07> is set and a cache parity error occurs in the tag field. |
| 07:05 | These bits are all set if CCR<07> is cleared and any type of cache parity error occurs. |
| 04:00 | These bits are always 0. |

### 9.4.5 Cache Memory

The Cache Memory is an 8-Kbyte, block size one, direct map (set size one) instruction cache. The Cache Data Path chip contains the control logic to support the Cache Memory. The Cache Memory is transparent to all programs and is designed with high-speed RAM memory. The memory locations currently being accessed from Program Memory are automatically stored in the Cache Memory. The next time these locations are accessed, the data is retrieved from the Cache Memory. This eliminates the time-consuming Program Memory Bus transaction. Full parity protection is provided for the Cache Memory and much of the parity calculations are done by the Cache Data Path chip.

The Cache Memory asserts COMP when an address scores a cache memory hit. MMISS is asserted when the address scores a cache memory miss. The memory read/write functions are controlled by the TAG BUS from the Cache Data Path chip. The data is written into the memory from the A BUS and read on the B BUS.

Figure 9-14 shows the logical subdivision of the 22-bit physical address.

**Figure 9-14 Logical Subdivision of a 22-bit Physical Address**

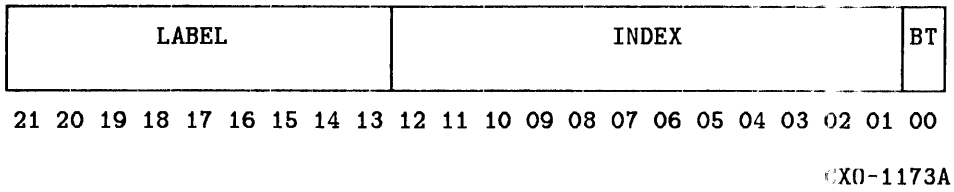
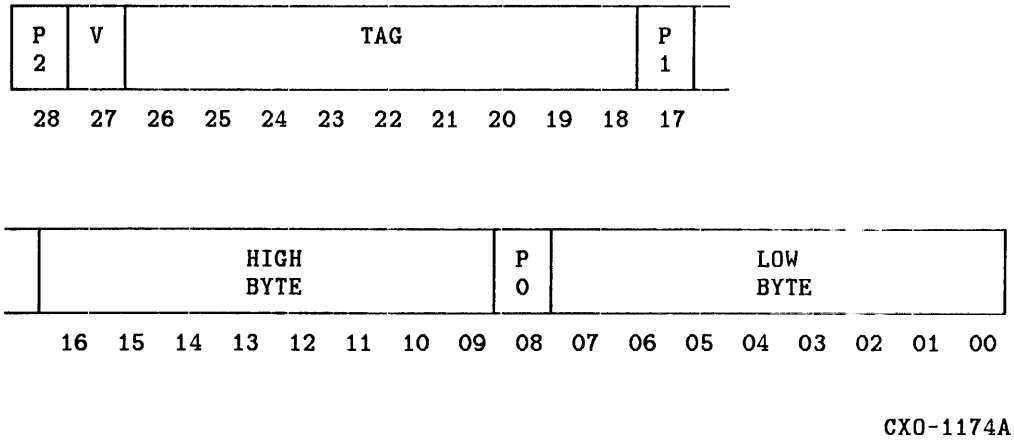



Figure 9-15 shows the organization of each 29-bit cache entry.

**Figure 9-15 Organization of each 29-bit Cache Entry**




P0 is the odd parity bit for the low data byte; P1 is the odd parity bit for the high data byte. P2 stores even parity for the tag and valid (V) bits.

In operation, the 12 bit index field from the physical address is used to access one of the 4-Kword locations in the cache. Then the 9-bit label field of the physical address is compared with the 9-bit tag field stored in the cache. If the two fields compare, then the valid bit is checked to contain a one and the P2 parity bit is checked for correct parity. If these conditions are met, a cache hit is declared and the data contents of that cache entry are used by the J-11. If any of these conditions is not met, then a cache miss is declared and the J-11 must then access Program Memory.

The response of the cache to various conditions is shown Table 9-7.

Table 9-7 Response of the Cache to Various Conditions

| | CPU Hit | CPU Miss | DMA Hit | DMA Miss |
|---|---|---|---|---|
| READ | Read cached data | Read memory - allocate cache | Read memory - no cache change | Read memory - no cache change |
| WRITE WORD | Write through cache to memory | Write memory - allocate cache | Invalidate cache - update memory | Update memory - no cache change |
| WRITE BYTE | Write through cache to memory | Write memory - no cache change | Invalidate cache - update memory | Update memory - no cache change |
| READ BYPASS | Read memory no cache change | Read memory - no cache change | — | — |
| WRITE BYPASS | Write memory - invalidate cache | Write memory - no cache change | — | — |
| READ FORCE MISS | Read memory - no cache change | Read memory - no cache change | — | — |
| WRITE FORCE MISS | Write memory - no cache change | Write memory - no cache change | — | — |

### 9.4.6 Direct Memory Access Register

The Direct Memory Access (DMA) Register is written with the address of each DMA write cycle. This address is monitored by the State Sequencer to ensure that the cache data does not become stale. If the DMA write address is in cache, then that cache entry is invalidated.

The DMA register receives an address from the Program Memory Bus via the A BUS and latches it into the register when RSYNC is asserted. The address is driven onto the B BUS to check it against the addresses in the cache memory when DMA REG OE is asserted.

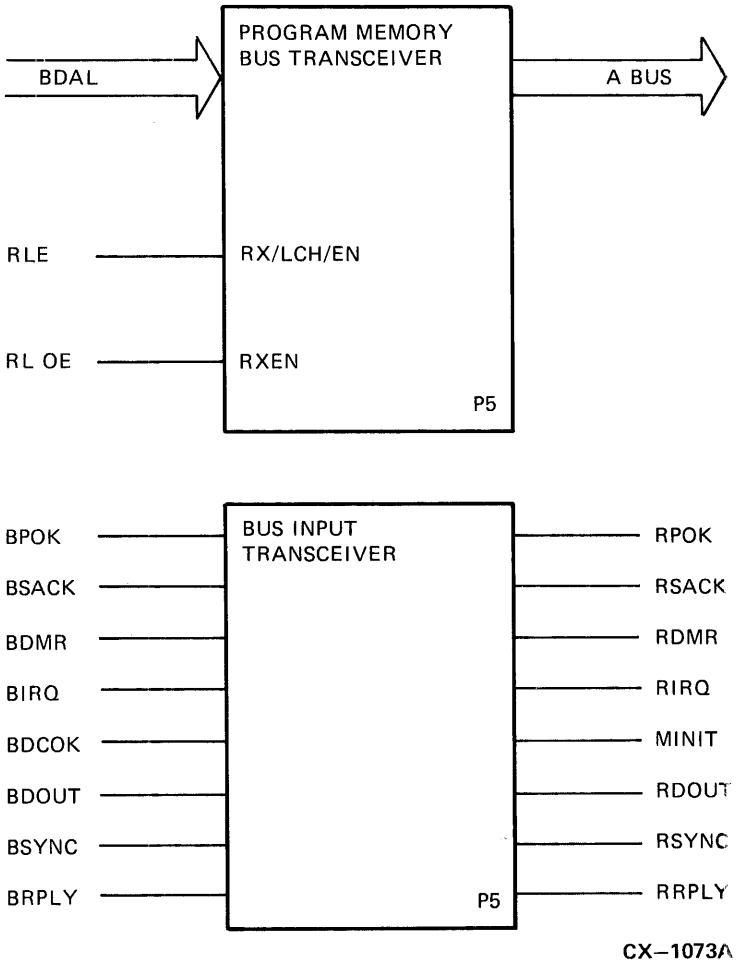
### 9.4.7 Program Memory Bus Receivers

The P.ioj control logic receives address and data from the Program Memory Bus via six 2908 bus transceivers as shown in Figure 9-16. The State Sequencer provides the control signals RLE and RL OE that transfer bus data to the A BUS. The data is latched when RLE is asserted. The output drivers are then enabled by RL OE, which enables the bus data to the A BUS.

### 9.4.8 Program Memory Bus Transmitters

The P.ioj control logic transmits address and data to the Program Memory Bus via six 2908 bus transceivers as shown in Figure 9-17. The address and data inputs are controlled by the LATCH input. The address is clocked into the transceiver when the STRB input from the DCJ11 is asserted. Write data is clocked into the transceiver when the DCRP input from the State Sequencer is asserted.

Figure 9-16 Program Memory Bus Receivers




The Program Memory control signals are transmitted by the Bus Output Transceivers. The State Sequencer provides most of the handshake protocol with the Program Memory Bus.

### 9.4.9 Maintenance Register

The Maintenance Register is configured for the J-11 to power up to the bootstrap address of $17773000_8$. This register also causes a trap through vector $004_8$ after a kernel mode HALT if the Secure/Enable switch is in the SECURE position. If the switch is in the ENABLE position, a kernel mode HALT causes the J-11 to enter micro-ODT. The register is located at $17777750_8$ and is read-only. Figure 9-18 shows the format of the register, and Table 9-8 describes the bits.

Figure 9-17 Program Memory Bus Transmitters

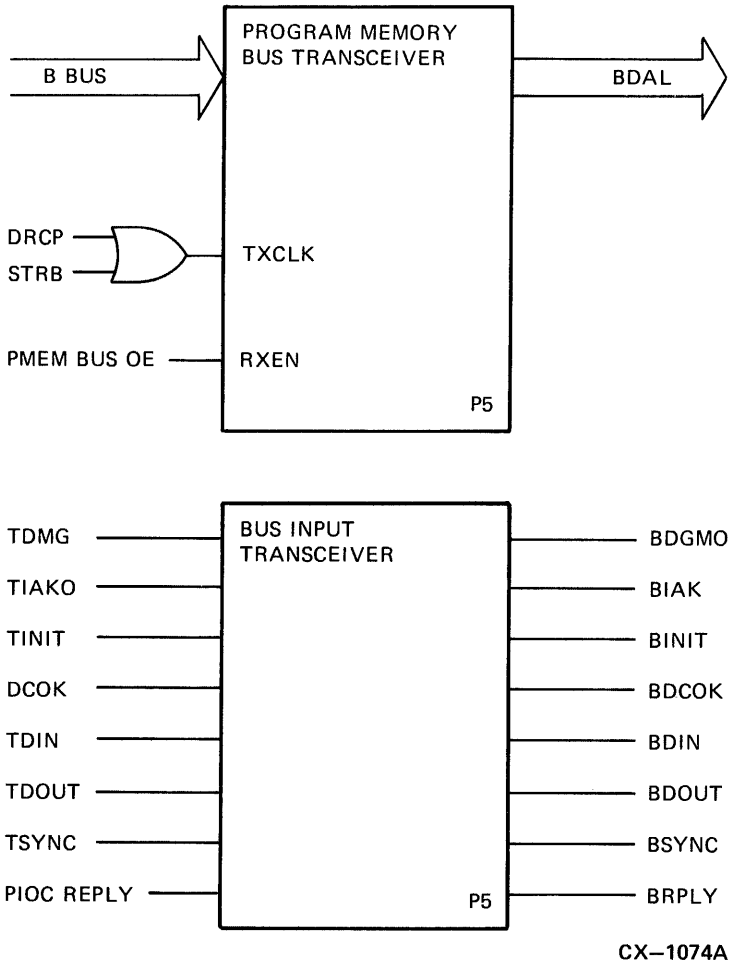



Figure 9-18 Maintenance Register

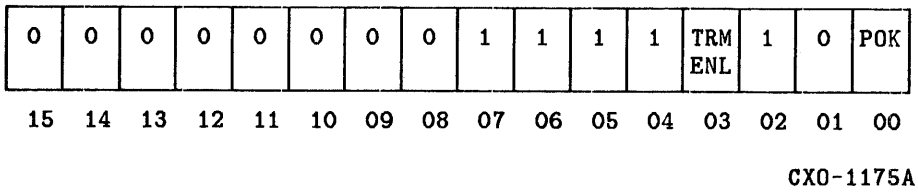


Table 9-8 Maintenance Register Bit Description

| Bits | Description |
|---|---|
| 15:08 | These bits are always 0. |
| 07:04 | These bits are always 1. This configuration indicates a J-11 module design unknown to standard PDP-11 operating systems. |
| 03 | This bit, TERMINAL ENABLE, is the inverse of Pio Control and Status Register bit <15>. This bit defines the action taken by the J-11 when a HALT instruction is encountered while in kernel mode. If the bit is a 1, a kernel HALT instruction will trap through vector $004_8$. If the bit is a 0, micro-ODT will be entered. |
| 02 | This bit is always 1. |
| 01 | This bit is always 0. |
| 00 | When set, POWER OK, this bit indicates that the power supplies have detected that the ac input power is satisfactory. |

## 9.5 INTERRUPT LOGIC

The Interrupt Logic generates the interrupt and vector address to the DCJ11 (located on the left side of Figure 9-2). Table 9-9 shows the interrupt and trap vectors that are assigned in the J-11 P.ioj. Notice that the Control Bus can software interrupt on level 7 through 4 (Section 9.15.4 and Chapter 3).

Table 9-9 Interrupt and Trap Vectors for J-11 P.ioj

| Trap Vector Address | Interrupt Level | Function |
|---|---|---|
| 000 | | (reserved) |
| 004 | Non-maskable | Bus error (non-existent memory) |
| 010 | None | Illegal and reserved instruction |
| 014 | Non-maskable | BPT, breakpoint trap used for HSC ODT |
| 020 | Non-maskable | IOT, input/output trap used for HSC software inconsistency |
| 024 | Non-maskable | Power fail |
| 030 | Non-maskable | EMT, emulator trap used for system service calls |
| 034 | Non-maskable | TRAP instruction |
| 060 | 4 (1) | Console Terminal Receiver |
| 064 | 4 (2) | Console Terminal Transmitter |
| 100 | 6 | Line Clock |

Table 9-9 (Cont.) Interrupt and Trap Vectors for J-11 P.ioj

| Trap Vector Address | Interrupt Level | Function |
|---|---|---|
| 114 | Non-maskable | Parity error on read-from-memory |
| 120 | 4 (7) | Control Bus interrupt, level 4 |
| 124 | 5 | Control Bus interrupt, level 5 |
| 130 | 6 | Control Bus interrupt, level 6 |
| 134 | 7 | Control Bus interrupt, level 7 |
| 230 | 4 * | K.rx interrupt |
| 240 | 7 to 1 | Programmable interrupt request |
| 250 | Non-maskable | MMU abort |
| 300 | 4 (3) | Auxiliary Serial Port 1 Receiver |
| 304 | 4 (4) | Auxiliary Serial Port 1 Transmitter |
| 310 | 4 (5) | Auxiliary Serial Port 2 Receiver |
| 314 | 4 (6) | Auxiliary Serial Port 2 Transmitter |

Note that the numbers in parentheses () indicate the order of priority of level 4 interrupts.

* The level 4 interrupt generated on the Program Bus is the lowest priority level 4 interrupt. All level 4 interrupts generated within the P.ioj module have higher priority than any off the module.

## 9.6 BOOTSTRAP PROM

The bootstrap PROM contains two Kwords in two 1K pages (shown in the upper-right corner of Figure 9-2.) Each page starts at location $17773000_8$ and extends through $17776776_8$. The BOOT PAGE 2 bit in the Pio Control and Status Register determines which page is active. (Refer to Section 9.10.) The code contained in this PROM is executed after power-up, a subsystem INIT (from either the host or the Operator Control Panel), or a software-initiated INIT.

The J-11 Power Up Mode 2 is hard-wired. This mode causes automatic execution of the bootstrap at address $17773000_8$ and automatic loading of the Program Status Word with the value $340_8$.

## 9.7 CONSOLE TERMINAL INTERFACE

The Console Terminal Interface provides an interface to the Console Terminal (shown in the upper-right corner of Figure 9-2.) This interface uses an EIA RS-423 signal-level serial interface. The terminal is used to perform console functions. Communications between the processor and the operator or field engineer is via a sequence of ASCII characters which are interpreted by the processor as console commands.

The Console Terminal Interface uses a DL-11 compatible asynchronous receiver/transmitter DC319 chip. This interface is a superset of the DL-11D interface. That is, there are four registers, as follows, but with some additional bits defined:

- Receiver Status Register (RCSR) $17777560_8$
- Receiver Data Buffer Register (RBUF) $17777562_8$

- Transmitter Status Register (XCSR) $17777564_8$
- Transmitter Data Buffer Register (XBUF) $17777566_8$

Figure 9-19, 9-20, 9-21, and 9-22 shows the format of each register. Table 9-10, 9-11, 9-12, and 9-14 describe the bits for each register.

**Figure 9-19 Receiver Status Register**

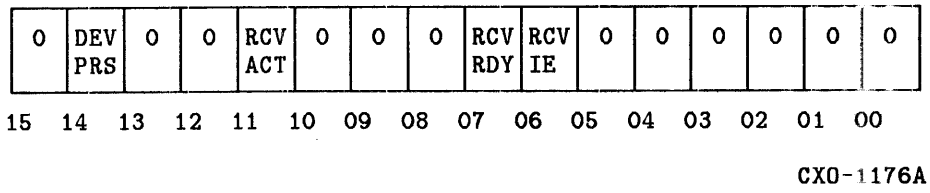



**Table 9-10 Receiver Status Register Bit Description**

| Bit | Description |
|---|---|
| 14 | When set, this bit indicates a terminal is connected to this interface. This bit, DEVICE PRESENT, is a read-only bit. This line is RS-232-C/RS-423 compatible. |
| 11 | When set, this bit indicates that a character is currently being received. This bit, RECEIVER ACTIVE, is a read-only bit. |
| 07 | When set, this bit indicates that there is a character in RBUF to be read. This bit, RECEIVER READY, is a read-only bit. |
| 06 | When set, this bit causes an interrupt to be generated on the assertion of RCV RDY. This bit, RECEIVER INTERRUPT ENABLE, is a read/write bit. Note that setting this bit when RCV RDY is already asserted will cause an interrupt. |

**Figure 9-20 Receiver Data Buffer Register**

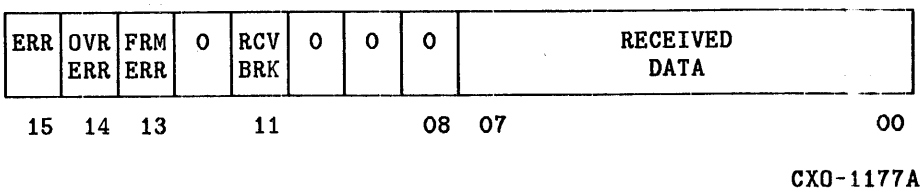


Table 9-11 Receiver Data Buffer Register Bit Description

| Bits | Description |
|---|---|
| 15 | This bit is the inclusive-OR of OVERRUN ERROR and FRAMING ERROR. This bit, ERROR, is a read-only bit. |
| 14 | When set, this bit indicates that the receiver attempted to set bit 07 in RCSR when it was already set. In other words, a new character was received and the preceding character had not yet been read by the J-11. |
| 13 | When set, this bit indicates no stop bit was received. This bit, FRAMING ERROR, is a read-only bit. |
| 11 | When set, this bit indicates the serial-in signal went from a mark to a space condition for 11 bit times after serial reception started. This bit, RECEIVED BREAK, is a read-only bit. |
| 07:00 | These eight bits are the character just received. These bits, RECEIVED DATA, are read-only bits. |

Note that there is no Parity Error bit implemented in this register. Parity is not checked or generated on any of the serial interfaces.

Figure 9-21 Transmitter Status Register

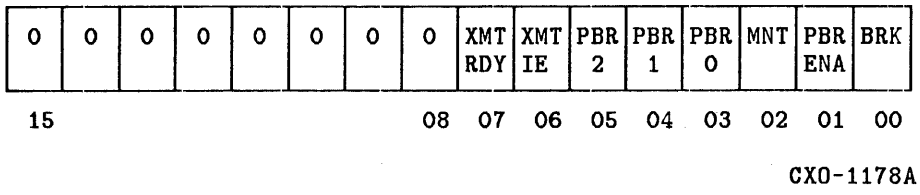



Table 9-12 Transmitter Status Register Bit Description

| Bits | Description |
|---|---|
| 07 | When set, this bit indicates that XBUF can accept a character for transmission. This bit, TRANSMITTER READY, is a read-only bit. |
| 06 | When set, this bit causes an interrupt to be generated on the assertion of XMT RDY. Note that setting this bit when XMT RDY is already set will cause an interrupt to be generated. This bit, TRANSMITTER INTERRUPT ENABLE, is a read/write bit. |
| 05:03 | When enabled by PBR ENA, bit <01> in this register, these bits select the baud rate for the serial port as shown in Table 9-13. |

Table 9-12 (Cont.) Transmitter Status Register Bit Description

| Bits | Description |
|---|---|
| 02 | When set, this bit causes the serial output of the transmitter section to be connected to the input of the receiver section. Thus, a character loaded into XBUF can then be read back in RBUF. This bit, MAINTENANCE, is a read/write bit. |
| 01 | When set, this bit overrides the default (hardware determined) baud rate of the serial port and allows the Programmable Baud Rate Select bits to determine the baud rate. This bit, PROGRAMMABLE BAUD RATE ENABLE, is a read/write bit. |
| 00 | When set, this bit causes the transmitter to send a continuous SPACE, which will be detected by the receiver as a framing error. This bit, BREAK, is a read/write bit. |

Table 9-13 Baud Rate Select for a DC319

| PBR 2 | PBR 1 | PBR 0 | Baud Rate |
|---|---|---|---|
| 0 | 0 | 0 | 300 |
| 0 | 0 | 1 | 600 |
| 0 | 1 | 0 | 1200 |
| 0 | 1 | 1 | 2400 |
| 1 | 0 | 0 | 4800 |
| 1 | 0 | 1 | 9600 |
| 1 | 1 | 0 | 19.2K |
| 1 | 1 | 1 | 38.4K |

Figure 9-22 Transmitter Data Buffer Register

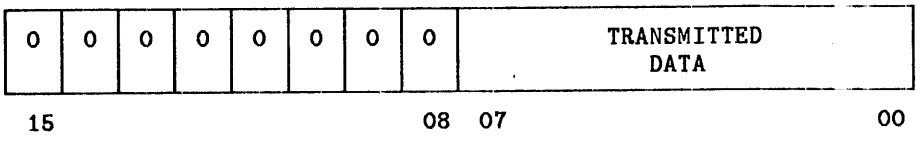


Table 9-14 Transmitter Data Buffer Register Bit Description

| Bits | Description |
|---|---|
| 07:00 | These eight bits are the characters to be transmitted. These bits, TRANSMITTED DATA, are write-only bits. |

Data format for this interface is 8 data bits, no parity, and 1 stop bit.

Vectors for this interface are $060_8$ for the receiver and $064_8$ for the transmitter. It interrupts at level 4.

The default baud rate for this port is 9600 baud.

A maintenance feature is included in this interface. If the INIT button on the Operator Control Panel is held depressed, any character received from the console keyboard is immediately echoed back to the terminal. This provides a simple test of the serial line driver and receiver, while bypassing the UART and associated hardware.

## 9.8 AUXILIARY SERIAL INTERFACES

There are two additional serial ports provided to the J-11 P.ioj (shown in the upper-right corner of Figure 9-2). These interfaces are also a superset of the DL-11D. The register set is the same as for the console terminal. (Refer to Section 9.7.)

The first Auxiliary Serial Port (AUX1) occupies addresses $17777520_8$ - $17777526_8$ and uses $300_8$ and $304_8$ for its Receiver and Transmitter vectors, respectively. The second Auxiliary Serial Port (AUX2) occupies addresses $17777530_8$ - $17777536_8$ and uses $310_8$ and $314_8$ for its Receiver and Transmitter vectors, respectively. Both interfaces interrupt at level 4.

The serial interface is electrically compatible with EIA RS-423 signal levels. It provides for asynchronous, full duplex communication. Transmit and receive baud rates are the same.

The default baud rate for both Auxiliary Serial Ports is 9600 baud.

## 9.9 I/O PAGE ADDRESS DECODES

The I/O Page Address Decodes decode the OUT BUS and provide the load strobes and read strobes for the I/O page addresses. The decode is shown in the left of Figure 9-2.

## 9.10 PIO CONTROL AND STATUS REGISTER

The Pio Control and Status Register (PIOCSR) contains bits that control functions on the P.ioj. It also provides status information.

Figure 9-23 shows the format of the PIOCSR. Table 9-15 describes each bit of the register. Except as noted in Table 9-15, all bits are read/write and are cleared by TINIT. This register is located at $17770040_8$.

Figure 9-23 Pio Control and Status Register

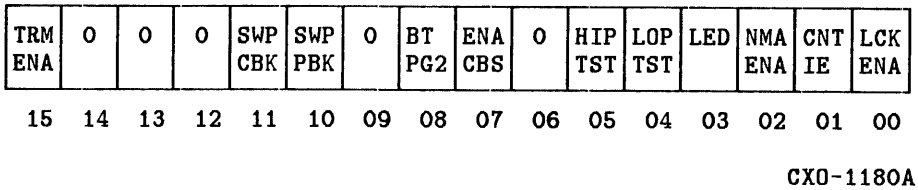


Table 9-15 Pio Control and Status Register Bit Description

| Bits | Description |
|---|---|
| 15 | This bit corresponds to the position of the Operator Control Panel Secure/Enable switch. A one indicates the ENABLE position of the switch (read-only). |
| 14 | This read-only bit is always zero. |
| 13:12 | These read/write bits are not used. |
| 11 | This bit allows the software to swap Control Memory banks. When set, it causes the HSC70 Memory Module to select the upper half of the Control Memory DRAMs as the active bank. |
| 10 | This bit allows the software to swap Program Memory banks. When set, it causes the HSC70 Memory Module to swap the addressing ranges between Program Memory banks on that module. This bit maximizes the probability that working memory is available starting at address $000000_8$ for interrupt vectors. |
| 09 | This read/write bit is not used. |
| 08 | This bit determines which page of the Bootstrap PROM is selected. When reset, the first 1 Kwords of the Bootstrap PROM are accessible in the range $17773000_8$ through $17776776_8$. When set, it causes the second 1 Kwords of the Bootstrap PROM to be accessible in the range $17773000_8$ through $17776776_8$. |
| 07 | This bit enables the Control Bus Arbitrator. When set, the Control Bus Arbitrator is enabled for the nine other Control Bus requesters. When clear, only the P.ioj may access Control Memory. |
| 06 | This read-only bit is always zero. |
| 05 | This bit is the high-byte parity test. When set, it causes the high byte parity tree to generate even parity on the next write-cycle to memory, therefore writing incorrect high-byte parity at that address. The bit is automatically cleared following that write cycle. This function applies to all three areas of memory (Program, Control, and Data). This bit is also cleared by TINIT. |
| 04 | This bit is the low-byte parity test. It performs the same function as bit <05> for the low byte. |
| 03 | This bit lights the yellow LED, D5, located near the handle of the P.ioj. It also drives the STATE indicator located on the Operator Control Panel. |
| 02 | This bit is the non-memory-access enable. When set, it causes the P.ioj to execute a Non-Memory Access (NMA) cycle on the next Control Memory or Data Memory access. This bit is cleared automatically at the conclusion of that Control or Data Memory access. This bit has no effect on and is not cleared by Program Memory accesses. |

Table 9-15 (Cont.) Pio Control and Status Register Bit Description

| Bits | Description |
|---|---|
| 01 | This bit is the Control Memory interrupt enable. When set, it causes the P.ioj to execute an interrupt cycle on the next Control Memory access. This bit is cleared automatically at the conclusion of that Control Memory access. Note that if the P.ioj accesses Data Memory with this bit set, the bit will be cleared after that access with no interrupt generated. |
| 00 | This bit is the Control Memory lock enable. When set, it causes the P.ioj to execute a lock cycle on the next Control Memory access. This bit is cleared automatically at the conclusion of that memory access. Note that if the P.ioj accesses Data Memory with this bit set, the bit will be cleared after that access with no lock cycle generated. |

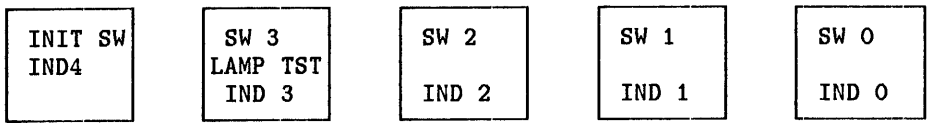
NOTE: A different priority scheme from the F-11 P.ioc implementation has been used for these last three bits (NMA ENA, CNT IE, and LCK ENA). In this design, setting NMA ENA will cause an NMA cycle to occur, regardless of the state of the other two bits. However, setting both CNT IE and LCK ENA simultaneously will generate an illegal cycle code on the Control Bus, providing a mechanism for diagnostics to check this logic. Under normal circumstances, only one of these three bits should be set at a time.

## 9.11 SWITCH/DISPLAY REGISTER

The Switch/Display Register (shown in the lower middle of Figure 9-2) is the interface between the software and the Operator Control Panel. This read/write is register located at $17770042_8$.

Figure 9-24 shows the arrangement of the five switches and indicators on the Operator Control Panel. Response to operation of these switches is handled entirely by software; no interrupts to the J-11 are generated by changes of state of the switches (except, of course, for INIT).

Figure 9-24 Indicators and Switches on the Operator Control Panel




CXO-1181A

The functions of the five panel switches and indicators are defined by software, except as follows:

- INIT—a momentary push button which, when pushed and released, causes the J-11 to execute its bootstrap routine.
- SW 3—a momentary push button which, when pushed, provides a hardware lamp test.

In addition, there is a two-position rocker switch, SECURE/ENABLE. This switch provides three functions:

- When the switch is in the ENABLE position, a break received from the terminal causes the J-11 to enter $\mu$ODT.
- If the J-11 is in Kernel mode and HALT instruction is decoded, the state of this switch is sampled. If the switch is in the ENABLE position, the J-11 enters $\mu$ODT. If the switch is in the SECURE position, the J-11 executes a trap to location $10_8$. If the J-11 is in user mode, a HALT instruction causes a trap to location $10_8$.
- In addition, if this switch is in the SECURE position, the INIT switch is disabled.

Figure 9-25 shows the format of the Switch/Display Register. Table 9-16 describes each bit of the register. This register must be polled by software to detect changes in the state of the switches.

**Figure 9-25 Switch/Display Register**

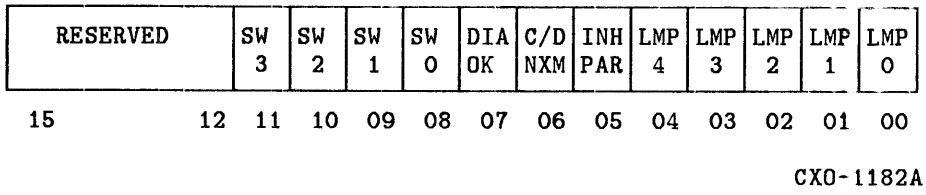



**Table 9-16 Switch/Display Register Bit Description**

| Bits | Description |
|---|---|
| 15:12 | These read-only bits are always zero. |
| 11:08 | These read-only bits reflect the state of the corresponding switches on the Operator Control Panel. Note that no hardware debounce logic is provided; debouncing is done by software. |
| 07 | This bit lights the DIAGNOSTICS TESTING/FAILED LED or the DIAGNOSTICS PASSED LED. A zero lights the DIAGNOSTICS TESTING/FAILED LED (red) and a one lights the DIAGNOSTICS PASSED LED (green). This bit powers up as zero and is set to a one by the successful completion of the PROM bootstrap diagnostics. The intent of this bit is a simple visual GO/NO-GO indicator for the P.ioj. |
| 06 | This bit, when set, forces a NXM trap on any Control or Data Memory Bus access. |
| 05 | This bit, when set, inhibits all parity traps, thus enabling examination of any memory location which has a hard parity error. |
| 04:00 | These read/write bits, when set, light the corresponding indicators on the Operator Control Panel. |

Note that the eight writable bits are cleared by TINIT.

## 9.12 K INIT AND STATUS REGISTERS

The K INIT and Status Registers initialize and hold the eight bit status of the:

Host Interface (K.pli)
Disk Data Channel (K.sdi)
Tape Data Channel (K.sti)

Figure 9-26 shows the format of the K INIT Register located at $17770044_8$. Table 9-17 describes the bits in the K INIT Register.

**Figure 9-26 K INIT Register**

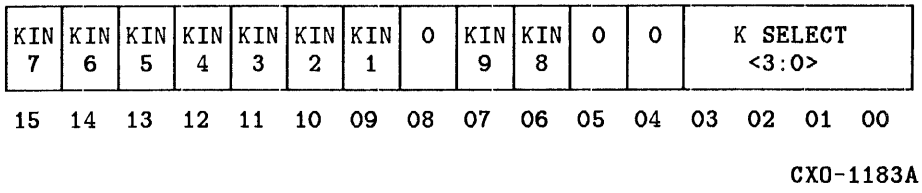



**Table 9-17 K INIT Register Bit Description**

| Bits | Description |
|---|---|
| 15:09<br>07:06 | These bits are backplane slot-specific clear signals. A zero in any bit position clears the module in the corresponding backplane slot. A one in any bit position negates the clear to that module and enables it to run. These bits are cleared to zero on power-up or after a subsystem INIT from the Operator Control Panel, but not on a subsystem INIT from a host processor. Note that since the P.ioj is considered Requester 0 in the HSC70, there is no KIN <0>. |
| 08,05,04 | These are spare bits. |
| 03:00 | These four bits are decoded to select the status of a requester. The codes refer to specific backplane slots. These bits are written first, and then the Status Register is read to obtain the status. This implementation provides for 10 requester codes (codes $00_8$ through $11_8$); codes $12_8$ through $17_8$ generate a K STATUS byte of $377_8$ or all ones. |

Figure 9-27 shows the format of the Status Register located at $17770046_8$. Table 9-18 describes the bits in the Status Register.

**Figure 9-27 Status Register**

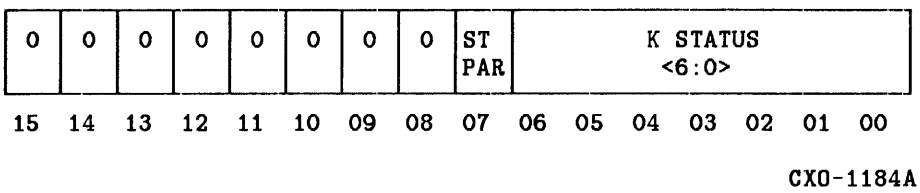


Table 9-18 Status Register Bit Description

| Bits | Description |
|---|---|
| 07 | STATUS PARITY — This bit is Status Parity. The Ks supply the parity bit from software (even or odd). The P.ioj checks the parity with software. |
| 06:00 | These bits, K STATUS, are the 7-bit status code returned from each Bus Requester. Note that if a backplane slot is not occupied, reading the status for that slot will return a status byte of $377_8$ or all ones. |

**NOTE**
**Both the K INIT Register and Status Register are read/write. However, byte-writes to K INIT Register write all 16 bits of this register.**

## 9.13 ERROR ADDRESS REGISTERS

The Error Address Registers (shown in the middle of Figure 9-2) consist of the Low Error Address and High Error Address. These two registers catch the full 22-bit physical address on the OUT BUS at the time a NXM or parity trap occurs. The J-11 could have been addressing the:

Program Memory
Control Memory
Data Memory

These registers are frozen by the occurrence of the error and each, independent of the other, remains frozen until it is read. After being read, each register will resume catching the corresponding portion of the address. The following are the addresses of the two registers:

Low Error Register — $17770024_8$
High Error Register — $17770026_8$

Figures 9-28 and 9-29 show the format of these two registers.

## 9.14 SERIAL NUMBER REGISTER

The 16-bit Serial Number Register provides a unique identifying number for each P.ioj module. (The Serial Number Register is shown just right of the middle of Figure 9-2.) The serial number is encoded by adding or removing jumpers on the module. The address of this register is $17770056_8$.

Figure 9-28 Low Error Address Register

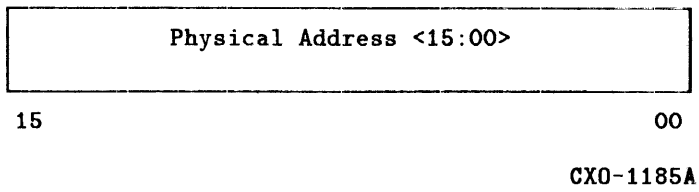
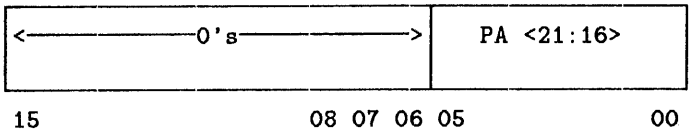


Figure 9-29 High Address Register




## 9.15 CONTROL MEMORY INTERFACE

The Control Memory interface consists of three blocks on:

- Control Memory Windows
- Control Bus Interface
- Control Bus Arbitrator and Timing

The following sections describe these three blocks which are shown at the right side of Figure 9-2.

### 9.15.1 Control Memory Windows

The software has very little need to access Data Memory because the Ks load and unload data buffers. But the software needs to access many different control blocks scattered through Control Memory. In addition to the standard MMU, there is a need for a separate address relocation feature for Control Memory. A large number of page address relocation registers are needed, but page descriptor registers are not necessary.

The Control Window feature provides 1024 Window Address Registers (WADRs). They are used whenever there is a virtual address in the range $16xxxx_8$.

**NOTE**

**Throughout the discussion of Control Window Logic, a 16-bit address is used for ease in explanation. However, the MMU in the DCJ11 translates this 16-bit address to a 22-bit address before gating it to the B BUS.**

Whenever an address is in the $16xxxx_8$ range:

- The hardware activates the Control Window logic
- The appropriate Window Address Register is gated (along with part of the virtual address) to the Control Memory Address (CADR) lines
- The Control Bus is requested
- Control Memory is accessed

The software can still use the MMU to access Control memory, but this is rarely done.

The software can load and unload the WADRs by using I/O Page addresses.

All of the hardware for this feature resides on the P.ioj module and consists of the following:

- 1024 WADRs contained in a 1K x 16 RAM
- WINDR $170020_8$ — Window Index Register

- WBUSR $170022_8$ - Hi Order Address History Register
- LOCADR $170054_8$ - Lo Order Address History Register

The WADRs are the actual page address registers. The WINDR is a index register used as part of the WADR 1-of-1024 selection. The WBUSR and LOCADR are history registers that hold the address of the last Control Memory access for error recovery and diagnostic purposes.

#### 9.15.1.1 Control Memory Window Address Selection

The software uses the Control Memory window logic to change a 16-bit PDP-11 address into a 17-bit Control Memory address. This 17-bit address then points to a control block in Control Memory. Control blocks start on a double-word boundary and the address represents an offset from the start of Control Memory measured in increments of double words.

To change the 16-bit address to a 17-bit address the software performs the following four steps:

- Selects a window set
- Writes a double-word offset address into a WADR
- Reads back a virtual address
- Uses the virtual address to address Control Memory

Figure 9-30 is a sample PDP-11 macro program that shows the steps involved in changing the 16-bit address to a 17-bit address using the WINDR and WADR. The following describes the steps:

**Figure 9-30 WADR Addressing**

```
012767 000005 170020  MOV 1  #5,$WINDR     ;select window set #5 using the
                                            ;window index register
   .
   .
   .
016767 100000 170006  MOV 2  DBPTR,$WADR3 ;MOV the double word offset to WADR3
016700 170006         MOV 3  $WADR3,R0     ;get the virtual address in R0
   .
   .
   .
011002                MOV 4  (R0),R2       ;get the contents of Control Memory
                                            ;and put it in R2
   .
   .
   .
```

CXO-1269A

1. The first MOV writes a 5 in the WINDR which selects window set number 5 out of a possible 128 sets. Figure 9-31 shows how the 1 Kword by 16-bit RAM is divided into 128 window sets of eight WADRs.

   The WINDR is loaded once prior to using a window set.

2. The second MOV writes the double-word offset address from the Control Memory virtual address of 0 to WADR3. The source is gated directly into the WADR without shifting the bits, as shown in the upper portion of Figure 9-32.

3. The third MOV reads back a *sort of* virtual address from WADR3. The low order seven bits of the WADR3 are shifted left two places and gated onto the IN BUS as shown in the middle portion of Figure 9-32. The I/O page address is $170006_8$. Table 9-19 shows the register address of each WADR and the virtual I/O page address range each covers.

4. The fourth MOV uses the *sort of* virtual address obtained from step 3 above to actually address Control Memory. When the Control Window Logic is activated (by a virtual address in the $16Zxxx_8$ range), the high order nine bits of the WADR are shifted left one place and gated onto the CADR lines as shown in the bottom portion of Figure 9-32.

**NOTE**

**The Z in the virtual address $16Zxxx_8$ is the WADR register.**

Table 9-19 I/O Page Address of Window Address Registers

| Register | Register Address | I/O Page Address |
|---|---|---|
| WADR0 | 170000 | 160000—160777 |
| WADR1 | 170002 | 161000—161777 |
| WADR2 | 170004 | 162000—162777 |
| WADR3 | 170006 | 163000—163777 |
| WADR4 | 170010 | 164000—164777 |
| WADR5 | 170012 | 165000—165777 |
| WADR6 | 170014 | 166000—166777 |
| WADR7 | 170016 | 167000—167777 |

Only 16-bit virtual addresses are shown here. However, the MMU would add bits to make 22-bit addressing.

Note that the third most significant digit of the I/O page address corresponds to the associated WADR.

Figure 9-31 Window Address Register Selection

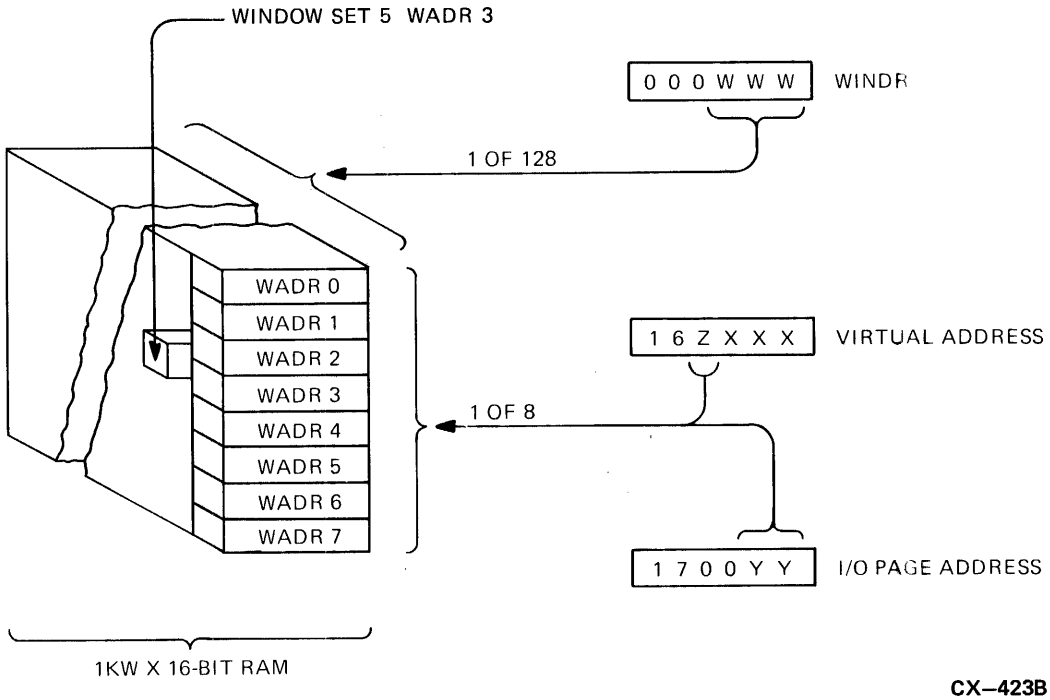
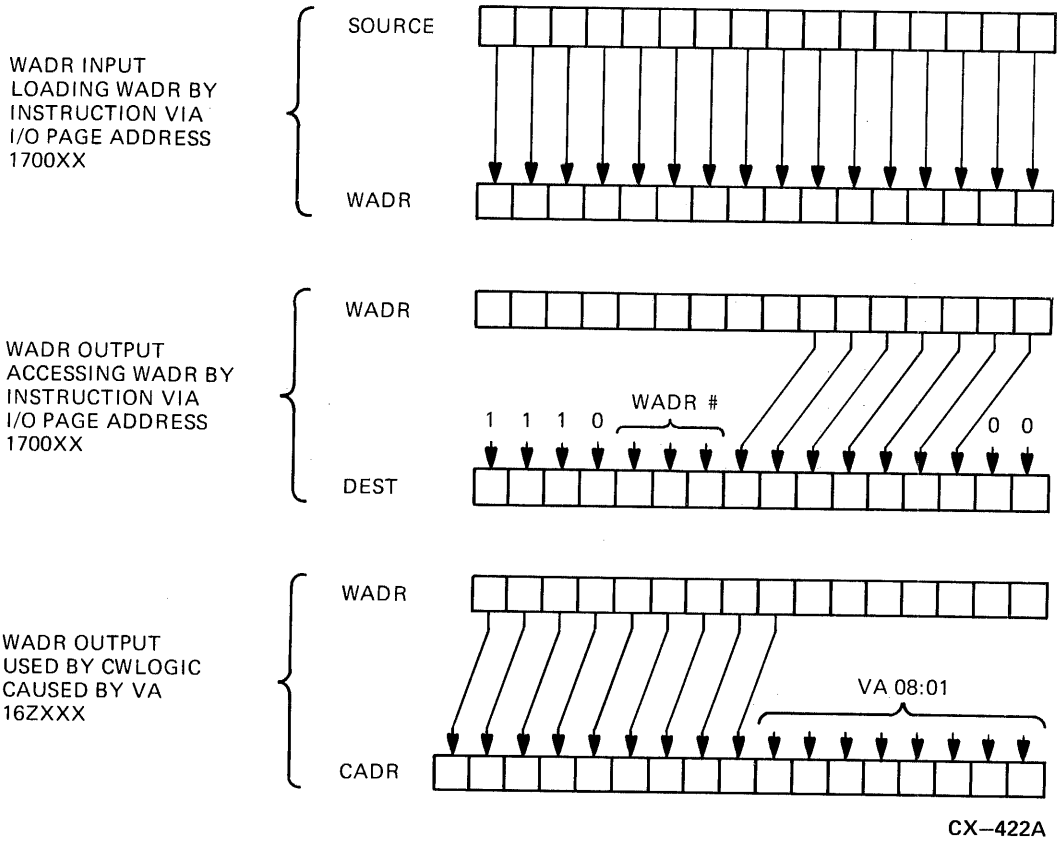


Figure 9-32 WADR Gating




### 9.15.1.2 Control Window Example

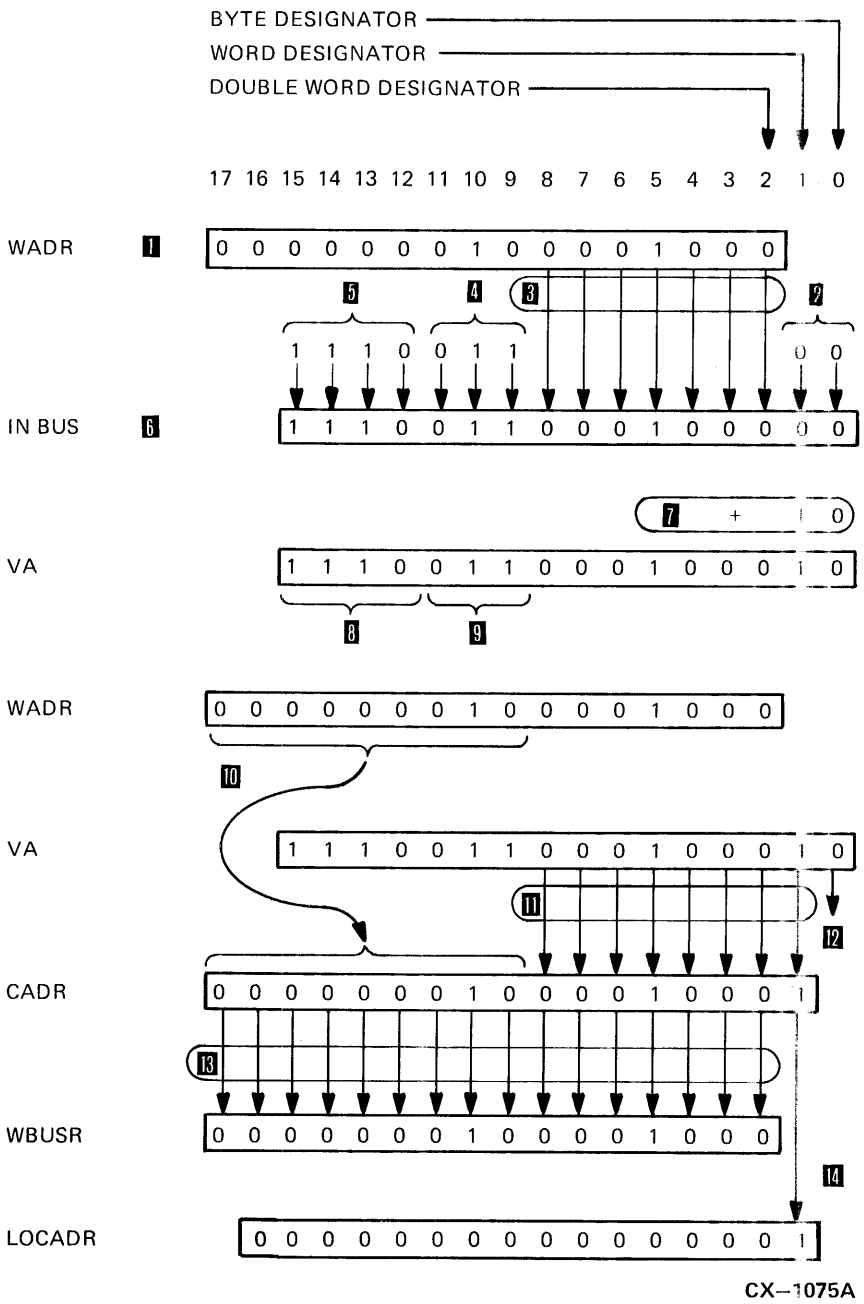
The following example serves to illustrate the use of a Control Window. Refer to Figure 9-33.

1. A control block is set aside in Control Memory by the P.ioj. The starting address of this control block is designated by a 16-bit pointer that represents the offset (in increments of double words) from the start of control memory.

   In this example, the pointer is $000410_8$ or an offset of $410_8$ double words from the start of Control Memory. Such an offset corresponds to Control Memory address $001020_8$, assuming word addressing starts at $000000_8$. (An MMU access of the same address would use the physical 22-bit, byte-oriented address $16002040_8$.)

2. The software selects one of the WADRs to use. In this case, it chose window set 5, WADR3.

Figure 9-33 Control Window Example



3. For selection of window set 5, the software deposits a $5_8$ in WINDR.
4. The software then deposits the pointer $000410_8$ into WADR3.

   1 WADR3 contains a pointer $000410_8$.

   The software retrieves the contents of WADR3 with a standard PDP-11 instruction. WADR contents are gated as follows:

   2 Zeros are gated onto IN BUS <01:00>, forcing a double word boundary.

   3 Bits <06:00> of the WADR are gated onto IN BUS <08:02>. These bits shift left two places, aligning the low-order bit of the pointer with a double-word designation.

   4 $011_2$ is gated onto IN BUS <11:09> because this is WADR3.

   5 $1110_2$ is gated onto IN BUS <15:12> because a WADR was addressed and creates $16xxxxx_8$.

   6 The result is $163040_8$. This will be used by the software as a 16-bit virtual address.

   Virtual address $163040_8$ corresponds to the start of the control block in this example. If the software requires only the first control block word or byte, it would add nothing to this base address.

   If the software wants to examine some information deeper in the control block, it adds to this base address. For instance, to examine the third byte (low-order byte of the second word), software adds two to the virtual address.

   The virtual address then contains $163042_2$, used by the software as the virtual address in a PDP-11 instruction. Because the hardware sees the virtual address in the $16xxxx_8$ range, the window logic is activated which forces use of the Control Bus. As a result, the hardware:

   7 Sees $16xxxx_8$ and activates the Control Window.

   8 Sees $xx3xxx_8$ and selects WADR3. (WINDR still selects window set 5.)

   9 Gates WADR3 <15:07> onto CADR <16:08>.

   10 Gates virtual address <08:01> onto CADR <07:00>.

   11 Uses virtual address <00> as a byte designator, if a byte instruction is being executed.

   CADR <16:00> ($001021_8$) on the Control Bus accesses a word in Control Memory. This is the second word of the control block.

   Finally, the hardware captures CADR <16:00> and saves it in the history registers.

   12 CADR <16:01> is captured in WBUSR <15:00>.

   13 CADR <00> is captured in LOCADR <00>.

#### 9.15.1.3 Control Window Limitation

One limitation is placed on the Control Window feature. If the software is sequentially accessing information from a control block and it crosses a $1000_8$ byte address boundary, the new address would have incremented the Z in virtual address $16Zxxx_8$. This would result in selecting a different WADR. So there is a $1000_8$ byte address boundary limitation. This limits the maximum size of a control block to 512 bytes assuming it starts at a $1000_8$ boundary; less if it starts at other than a $1000_8$ boundary.

### 9.15.2 Control Bus Interface

The Control Bus Interface block (located on the right side of Figure 9-2) contains the Control Bus Data Transceivers and Control Bus Address Transceivers for the P.ioj.

#### 9.15.2.1 Control Bus Data Transceivers

The Control Bus Data Transmitter is 20 bits long:

- <15:00> is data
- Bit 16 is low byte parity
- Bit 17 is high byte parity
- Low byte write enable (CWRTL)
- High byte write enable (CWRTH)

The transmitters are written from the OUT BUS at TDOUT time if the address is between $160000_8$ and $167777_8$.

The Control Bus Data Receiver is 18 bits long:

- <15:00> is data
- Bit 16 is low byte parity
- Bit 17 is high byte parity

The receivers are written onto the IN BUS <15:00> at TDIN time if the address is between $160000_8$ and $167777_8$.

#### 9.15.2.2 Control Memory Address Transceivers

The Control Memory Address Transceiver (CADR) address inputs are controlled by the Control Memory Window logic (Section 9.15.1).

The Control Memory Address Receivers are two read-only registers:

- Window Bus Register (WBUSR) — $170022_8$
- Low Control Memory Address Register (LOCADR) — $170054_8$

These two registers provide access to the address asserted on the Control Bus on each P.ioj cycle. They are updated after each Control Bus cycle by the P.ioj.

The Window Bus Register contains the high-order 16 bits of the 17-bit word address.

Figure 9-34 shows the format of the Low Control Memory Address Register. Table 9-20 describes the bits in this register.

**Figure 9-34 Low Control Memory Address Register**

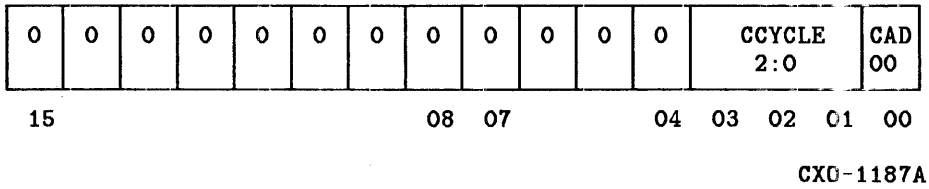


Table 9-20 Low Control Memory Address Register Bit Description

| Bits | Description |
|---|---|
| 15:04 | These bits are always 0. |
| 03:01 | These bits, CCYCLE <2:0>, are the cycle code bits corresponding to the last P.ioj Control Memory cycle. |
| 00 | This bit, CONTROL BUS ADDRESS <00>, is the least significant word address bit of the last P.ioj Control Memory cycle. |

### 9.15.3 Control Bus Arbitrator and Timing

The P.ioj module contains the timing and arbitration logic for the Control Bus (located in the lower middle of Figure 9-2). For signal polarities and information about this bus, refer to Chapter 3.

The J-11 P.ioj does not use any backplane pins to generate and receive the CREQUEST and CGRANT signals, as in the F-11 implementation. Instead, the J-11 P.ioj CREQUEST and CGRANT signals are contained entirely within the module and the extra backplane fingers are redefined to add two additional Control Bus Requesters. The new requesters are 8 and 9, which are higher priority than the original eight. The P.ioj is the lowest priority Control Bus requester.

The Control Memory consists of dynamic RAMs (DRAMs) which must be refreshed. To do this, the P.ioj generates an internal 66.7 KHz (15 $\mu$second period) clock to indicate when a refresh cycle is to be performed. This signal is input to the Control Bus Arbitrator as the highest priority Control Bus request. The arbitrator acknowledges the refresh cycle by the asserting CREFR GRANT on the next available Control Bus cycle, with the same timing as any of the CGRANT n signals. The Control Memory logic thus uses the assertion of CREFR GRANT to initiate an internal refresh cycle.

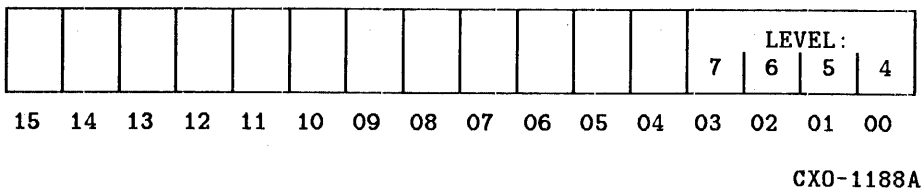
**NOTE**

**Refreshing is independent of the ENACBS bit in the PIOCSR.**

### 9.15.4 Interrupt Cycle

To execute a Control Bus Interrupt cycle, it is only necessary to set the CNT IE bit in PIOCSR and then MOV the Interrupt Mask to ANY Control Memory address, using either the Control Memory Windows or Memory Management. A one in any bit position will queue an interrupt at that level to the P.ioj. Interrupts on more than one level may be generated by one Control Bus Interrupt cycle. The CNT IE bit will be cleared at the completion of the Interrupt cycle. Figure 9-35 shows the format of the Interrupt Mask.

Figure 9-35 P.ioj Control Bus Interrupt Mask



### 9.15.5 Lock Cycle

To execute a Control Bus Lock cycle, it is only necessary to set the LCK ENA bit in PIOCSR and then MOV the desired Lock Address to any destination where it may be examined to determine if the location is already locked. At the completion of the Lock cycle, the LCK ENA bit in PIOCSR is automatically cleared.

## 9.16 DATA MEMORY INTERFACE

The Data Memory Interface consists of two blocks of logic shown in the lower right of Figure 9-2:

- Data Bus Interface
- Data Bus Arbitrator and Timing

The P.ioj module contains the timing and arbitration logic for the Data Bus. In addition, the P.ioj is the lowest priority requester. The following sections describe the Data Memory interface logic.

### 9.16.1 Data Bus Interface

The Data Bus Interface block contains the Data Bus Data Transceivers and the Data Bus Address Transceivers for the P.ioj. The following sections describe these two transceivers.

#### 9.16.1.1 Data Bus Transceivers

The Data Bus Data Transmitter is 20 bits long:

- <15:00> is data
- Bit 16 is low byte parity
- Bit 17 is high byte parity
- Low byte write enable (DWRTL)
- High byte write enable (DWRTH)

The transmitters are written from the OUT BUS at TDOUT time if the address is between $14000000_8$ and $15777777_8$.

The Data Bus Data Receiver is 18 bits long:

- <15:00> is data
- Bit 16 is low byte parity
- Bit 17 is high byte parity

The receivers are written onto the IN BUS <15:00> at TDIN time if the address is between $14000000_8$ and $15777777_8$.

#### 9.16.1.2 Data Bus Address Transceivers

The Data Bus deals with word addresses and has 18 address bits. This 18-bit address is handled differently in the Data Bus Address Transmitters than in the Data Bus Address Receivers.

##### 9.16.1.2.1 Data Bus Address Transmitters

The address on the OUT BUS <18:01> is clocked into the Data Bus Address Transmitters <17:00> when the address is between $14000000_8$ and $15777777_8$. Notice the shift of one bit to the right to make the address a word reference.

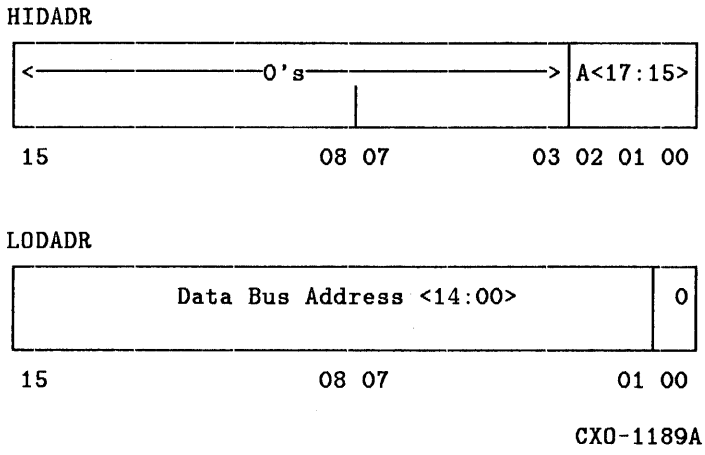
#### 9.16.1.2.2 Data Bus Address Receivers

The Data Bus Address Receivers are two read-only registers:

- High Data Bus Address Register (HIDADR) located at $17770052_8$
- Low Data Bus Address Register (LODADR) located at $17770050_8$

After each Data Bus cycle by the P.ioj, the registers are updated with the 18-bit word address. Figure 9-36 shows the format of the two receiver registers.

**Figure 9-36 Data Bus Address Transceivers**




### 9.16.2 Data Bus Arbitrator and Timing

The P.ioj module contains the timing and arbitration logic for the Data Bus. For signal polarities and information about this bus, refer to Chapter 3.

The J-11 P.ioj does not use backplane pins to generate and receive the DREQUEST and DGRANT signals, as in the F-11 implementation. Instead, the J-11 P.ioj DREQUEST and DGRANT signals are contained entirely within the module, and the extra backplane fingers are redefined to add two additional Data Bus Requesters. The new requesters are 8 and 9 (higher priority than the original eight). The P.ioj has the lowest priority Data Bus requester.

## 9.17 I/0 PAGE ADDRESS UTILIZATION

Table 9-21 is a list of the I/O page addresses assigned in the P.ioj.

## 9.18 I/O PAGE REGISTER FORMAT SUMMARY

Figure 9-37 is a summary of the I/O page registers on the P.ioj.

Table 9-21 I/O Page Address Utilization

| Physical Address | Function |
|---|---|
| 17777776 | Processor Status Word |
| 17777772 | Program Interrupt Request Register |
| 17777766 | CPU Error Register |
| 17777752 | Cache Hit/Miss Register |
| 17777750 | Maintenance Register |
| 17777746 | Cache Control Register |
| 17777744 | Memory System Error Register |
| 17777676 | User Data PAR 7 |
| 17777674 | User Data PAR 6 |
| 17777672 | User Data PAR 5 |
| 17777670 | User Data PAR 4 |
| 17777666 | User Data PAR 3 |
| 17777664 | User Data PAR 2 |
| 17777662 | User Data PAR 1 |
| 17777660 | User Data PAR 0 |
| 17777656 | User Instruction PAR 7 |
| 17777654 | User Instruction PAR 6 |
| 17777652 | User Instruction PAR 5 |
| 17777650 | User Instruction PAR 4 |
| 17777646 | User Instruction PAR 3 |
| 17777644 | User Instruction PAR 2 |
| 17777642 | User Instruction PAR 1 |
| 17777640 | User Instruction PAR 0 |
| 17777636 | User Data PDR 7 |
| 17777634 | User Data PDR 6 |
| 17777632 | User Data PDR 5 |
| 17777630 | User Data PDR 4 |
| 17777626 | User Data PDR 3 |
| 17777624 | User Data PDR 2 |
| 17777622 | User Data PDR 1 |
| 17777620 | User Data PDR 0 |
| 17777616 | User Instruction PDR 7 |
| 17777614 | User Instruction PDR 6 |
| 17777612 | User Instruction PDR 5 |
| 17777610 | User Instruction PDR 4 |
| 17777606 | User Instruction PDR 3 |
| 17777604 | User Instruction PDR 2 |
| 17777602 | User Instruction PDR 1 |
| 17777600 | User Instruction PDR 0 |

Table 9-21 (Cont.) I/O Page Address Utilization

| Physical Address | Function |
|---|---|
| 17777576 | Memory Management Status Register 2 |
| 17777574 | Memory Management Status Register 2 |
| 17777572 | Memory Management Status Register 0 |
| 17777566 | Console Xmtr Data Buffer Register |
| 17777564 | Console Xmtr Status Register |
| 17777562 | Console Rcvr Data Buffer Register |
| 17777560 | Console Rcvr Status Register |
| 17777546 | Clock Register |
| 17777536 | Aux Serial Port 2 Xmtr Data Buffer Register |
| 17777534 | Aux Serial Port 2 Xmtr Status Register |
| 17777532 | Aux Serial Port 2 Rcvr Data Buffer Register |
| 17777530 | Aux Serial Port 2 Rcvr Status Register |
| 17777526 | Aux Serial Port 1 Rcvr Status Register |
| 17777524 | Aux Serial Port 1 Xmtr Data Buffer Register |
| 17777522 | Aux Serial Port 1 Xmtr Status Register |
| 17777520 | Aux Serial Port 1 Rcvr Data Buffer Register |
| | Aux Serial Port 1 Rcvr Status Register |
| 17777400 | Command/Status Register |
| 17777402 | Memory Address Register 0 |
| 17777404 | Memory Address Register 1 |
| 17777406 | Memory Address Register 2 |
| | Memory Address Register 3 |
| 17776776 to 17773000 | Bootstrap PROM (both pages) |
| 17772516 | Memory Management Status Register 3 |
| 17772376 | Kernel Data PAR 7 |
| 17772374 | Kernel Data PAR 6 |
| 17772372 | Kernel Data PAR 5 |
| 17772370 | Kernel Data PAR 4 |
| 17772366 | Kernel Data PAR 3 |
| 17772364 | Kernel Data PAR 2 |
| 17772362 | Kernel Data PAR 1 |
| 17772360 | Kernel Data PAR 0 |

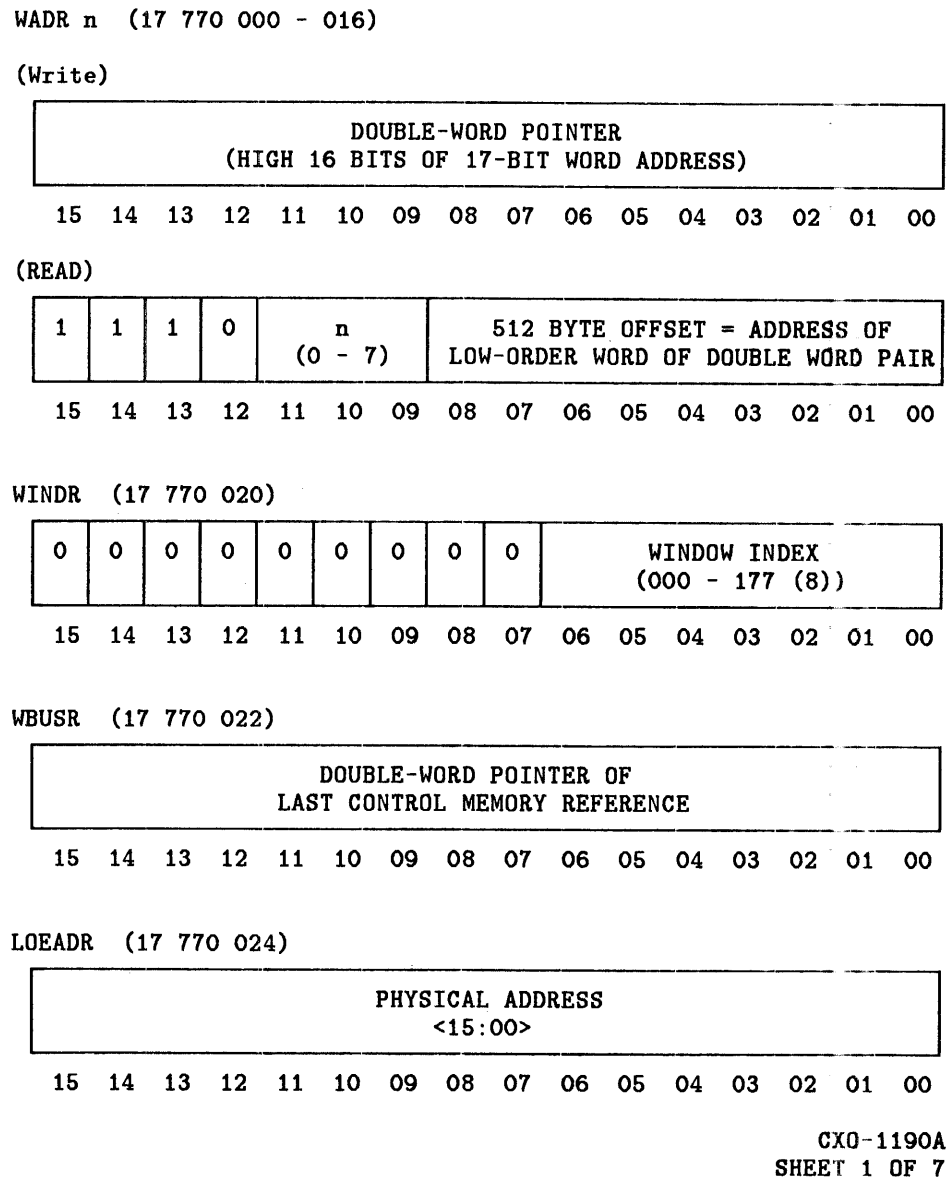
Table 9-21 (Cont.) I/O Page Address Utilization

| Physical Address | Function |
|---|---|
| 17772356 | Kernel Instruction PAR 7 |
| 17772354 | Kernel Instruction PAR |
| 17772352 | Kernel Instruction PAR |
| 17772350 | Kernel Instruction PAR |
| 17772346 | Kernel Instruction PAR |
| 17772344 | Kernel Instruction PAR |
| 17772342 | Kernel Instruction PAR |
| 17772340 | Kernel Instruction PAR |
| 17772336 | Kernel Data PDR 7 |
| 17772334 | Kernel Data PDR 6 |
| 17772332 | Kernel Data PDR 5 |
| 17772330 | Kernel Data PDR 4 |
| 17772326 | Kernel Data PDR 3 |
| 17772324 | Kernel Data PDR 2 |
| 17772322 | Kernel Data PDR 1 |
| 17772320 | Kernel Data PDR 0 |
| 17772316 | Kernel Instruction PDR 7 |
| 17772314 | Kernel Instruction PDR 6 |
| 17772312 | Kernel Instruction PDR 5 |
| 17772310 | Kernel Instruction PDR 4 |
| 17772306 | Kernel Instruction PDR 3 |
| 17772304 | Kernel Instruction PDR 2 |
| 17772302 | Kernel Instruction PDR 1 |
| 17772300 | Kernel Instruction PDR 0 |
| 17772276 | Supervisor Data PAR 7 |
| 17772274 | Supervisor Data PAR 6 |
| 17772272 | Supervisor Data PAR 5 |
| 17772270 | Supervisor Data PAR 4 |
| 17772266 | Supervisor Data PAR 3 |
| 17772264 | Supervisor Data PAR 2 |
| 17772262 | Supervisor Data PAR 1 |
| 17772260 | Supervisor Data PAR 0 |
| 17772256 | Supervisor Instruction PAR 7 |
| 17772254 | Supervisor Instruction PAR 6 |
| 17772252 | Supervisor Instruction PAR 5 |
| 17772250 | Supervisor Instruction PAR 4 |
| 17772246 | Supervisor Instruction PAR 3 |
| 17772244 | Supervisor Instruction PAR 2 |
| 17772242 | Supervisor Instruction PAR 1 |
| 17772240 | Supervisor Instruction PAR 0 |

Table 9-21 (Cont.) I/O Page Address Utilization

| Physical Address | Function |
|---|---|
| 17772236 | Supervisor Data PDR 7 |
| 17772234 | Supervisor Data PDR 6 |
| 17772232 | Supervisor Data PDR 5 |
| 17772230 | Supervisor Data PDR 4 |
| 17772226 | Supervisor Data PDR 3 |
| 17772224 | Supervisor Data PDR 2 |
| 17772222 | Supervisor Data PDR 1 |
| 17772220 | Supervisor Data PDR 0 |
| 17772216 | Supervisor Instruction PDR 7 |
| 17772214 | Supervisor Instruction PDR 6 |
| 17772212 | Supervisor Instruction PDR 5 |
| 17772210 | Supervisor Instruction PDR 4 |
| 17772206 | Supervisor Instruction PDR 3 |
| 17772204 | Supervisor Instruction PDR 2 |
| 17772202 | Supervisor Instruction PDR 1 |
| 17772200 | Supervisor Instruction PDR 0 |
| 17770056 | Serial Number Register |
| 17770054 | Control Memory Low Address Register |
| 17770052 | Data Memory High Address Register |
| 17770050 | Data Memory Low Address Register |
| 17770046 | K Status Register |
| 17770044 | K Init Register |
| 17770042 | Switch/Display Register |
| 17770040 | Pio Control and Status Register |
| 17770026 | High Error Address Register |
| 17770024 | Low Error Address Register |
| 17770022 | Window Bus Register |
| 17770020 | Window Index Register |
| 17770016 | Window Address Register 7 |
| 17770014 | Window Address Register 6 |
| 17770012 | Window Address Register 5 |
| 17770010 | Window Address Register 4 |
| 17770006 | Window Address Register 3 |
| 17770004 | Window Address Register 2 |
| 17770002 | Window Address Register 1 |
| 17770000 | Window Address Register 0 |
| 17767776 ⟩ 17760000 | Control Memory Windows |

Figure 9-37 I/O Page Register Format Summary




(Continued on next page)

Figure 9-37 (Cont.) I/O Page Register Format Summary

HIEADR (17 770 026)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PHYSICAL ADDRESS <21:16> | | | | | |

PIOCSR (17 770 040)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRM ENA | 0 | 0 | 0 | SWP BRD | SWP BNK | 0 | BT PG2 | ENA CBS | 0 | HIP TST | LOP TST | LED | NMA ENA | CNT IE | LCK ENA |

SWDR (17 770 042)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | SW 3 | SW 2 | SW 1 | SW 0 | DIA OK | C/D NXM | INH PAR | LMP 4 | LMP 3 | LMP 2 | LMP 1 | LMP 0 |

KINR (17 770 044)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KIN 7 | KIN 6 | KIN 5 | KIN 4 | KIN 3 | KIN 2 | KIN 1 | 0 | KIN 9 | KIN 8 | 0 | 0 | K SELECT <3:0> | | | |

KSTR (17 770 046)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ST PAR | K STATUS <6:0> | | | | | | |

CXO-1190A
SHEET 2 OF 7

(Continued on next page)

Figure 9-37 (Cont.) I/O Page Register Format Summary

LODADR (17 770 050)

| DATA BUS ADDRESS <14:00> | 0 |
|---|---|

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

HIDADR (17 770 052)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DATA ADDR <17:15> |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

LOCADR (17 770 054)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CCYCLE <2:0> | CAD 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

S/N REG (17 770 056)

| 16 BIT SERIAL NUMBER |
|---|

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

Command/Status Register (CSR) (17777400)

| PAR ERR | NXM ERR | INTR ENB | DIS REQ | PAR TST | BUSY LED | MOTR ENB | DRV SEL | CS7 | CS6 | CS5 | CS4 | CS3 | CS2 | CS1 | CS0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

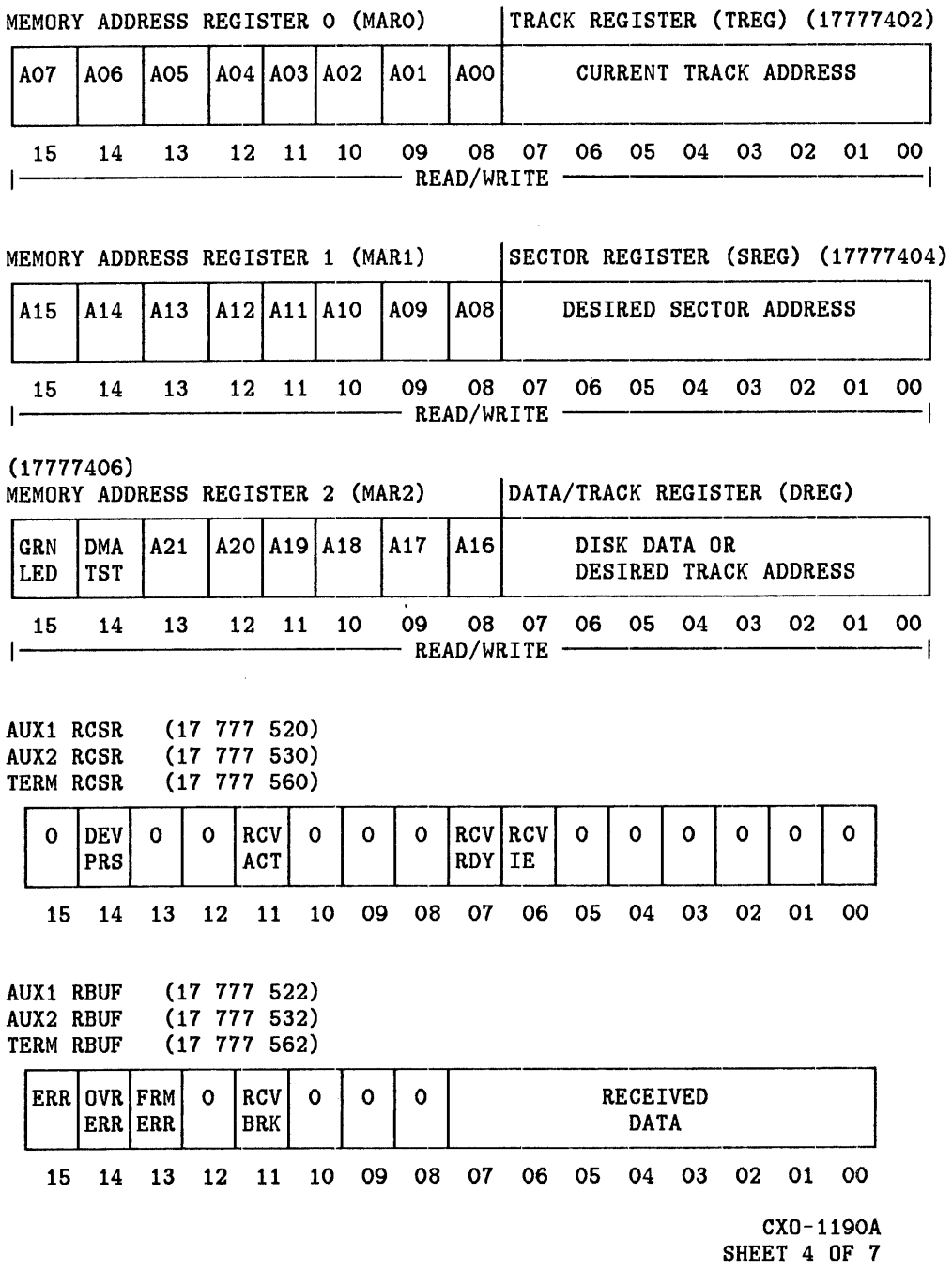
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

READ ONLY | READ ONLY | READ/WRITE |

CXO-1190A
SHEET 3 OF 7

(Continued on next page)

Figure 9-37 (Cont.) I/O Page Register Format Summary




(Continued on next page)

Figure 9-37 (Cont.) I/O Page Register Format Summary

AUX1 XCSR (17 777 524)
AUX2 XCSR (17 777 534)
TERM XCSR (17 777 564)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XMT RDY | XMT IE | PBR 2 | PBR 1 | PBR 0 | MNT | PBR ENA | BRK |

AUX1 XBUF (17 777 526)
AUX2 XBUF (17 777 536)
TERM XBUF (17 777 566)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TRANSMITTED DATA | | | | | | | |

CLKREG (17 777 546)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLK ENA | 0 | 0 | 0 | 0 | 0 | 0 |

MSER (17 777 744)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PAR ABT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | HI PAR | LO PAR | TAG PAR | 0 | 0 | 0 | 0 | 0 |

CACHE CONTROL REG (17 777 746)

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | WRW TPR | BYP | FL | PAR ABT | WRW DPR | 0 | 0 | FORCE MISS | | DIA MDE | DIS CPI |

CXO-1190A
SHEET 5 OF 7

(Continued on next page)

Figure 9-37 (Cont.) I/O Page Register Format Summary

MAINT REG (17 777 750)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | TRM ENL | 1 | 0 | POK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |

HIT/MISS REG (17 777 752)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | <——— FLOW |
|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 04 03 02 01 00 |

CPU ERR (17 777 766)

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ILL HLT | ADR ERR | NXM | I/O TO | YEL STK | RED STK | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |

PIRQ REG (17 777 772)

| PIR 7 | PIR 6 | PIR 5 | PIR 4 | PIR 3 | PIR 2 | PIR 1 | 0 | PRIORITY <15:09> | 0 | PRIORITY <15:09> | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 06 05 04 | 03 | 02 01 00 | |

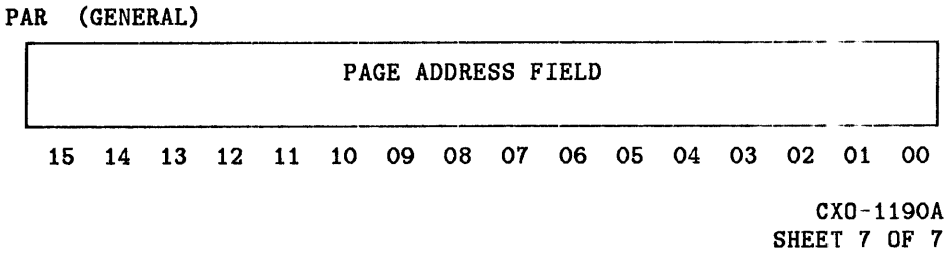
PDR (GENERAL)

| BYP CCH | PAGE LENGTH FIELD | 0 | PG WRT | 0 | 0 | EXP DIR | ACC CTL FIELD | 0 |
|---|---|---|---|---|---|---|---|---|
| 15 | 14 13 12 11 10 09 08 | 07 | 06 | 05 | 04 | 03 | 02 01 00 | |

CXO-1190A
SHEET 6 OF 7

(Continued on next page)

Figure 9-37 (Cont.) I/O Page Register Format Summary




CXO-1190A
SHEET 7 OF 7

# CHAPTER 10
# HSC50 MEMORY MODULE

## 10.1 INTRODUCTION

The HSC50 Memory Module (M.std) is an extended-hex size module residing in slot two of the HSC50, adjacent to the F-11 I/O Control Processor Module (P.ioc). The module number, L0106, is stamped on the handle. The M.std contains three separate and independent memories, each on its own bus. The three memories are:

- Control Memory or M.ctl (residing on the Control Bus)
- Data Memory or M.data (residing on the Data Bus)
- Program Memory or M.prog (residing on the Program Bus)

Each of the memory arrays includes byte parity.

Refer to Figure 10-1 for a block diagram of the M.std module and its three basic components.

The three memories each use different RAM types and quantities as shown in Table 10-1.

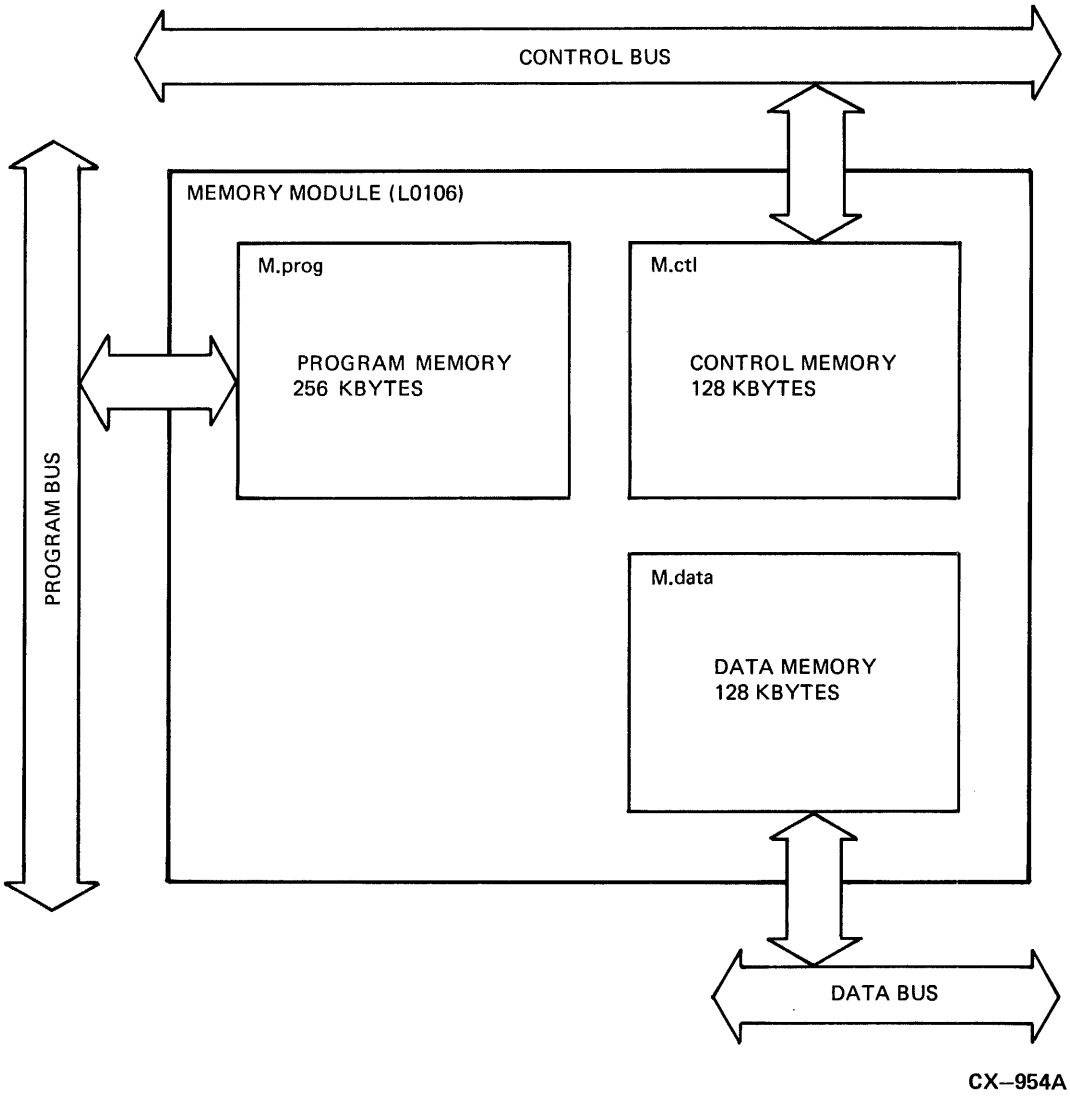
**Table 10-1 Size of Memory and Type of RAMs**

| Memory | Size | Number Of Chips | Type Of Chips | Access Time |
|---|---|---|---|---|
| M.ctl | 128 Kbytes<br>64 Kwords | 18 | 64KX1 Dynamic RAMs | 150 nanoseconds |
| M.data | 128 Kbytes<br>64 Kwords | 72 | 16KX1 Static RAMs | 70 nanoseconds |
| M.prog | 256 Kbytes<br>128 Kwords | 36 | 64KX1 Dynamic RAMs | 150 nanoseconds |

**NOTE**

**The HSC50 was originally designed to accept more than one memory module and the memory module was designed to permit different configurations of memory size. However, at present, the HSC50 uses only a single memory module with the configurations shown in Table 10-1.**

Figure 10-1 HSC50 Memory Module Block Diagram




## 10.2 STATUS LED

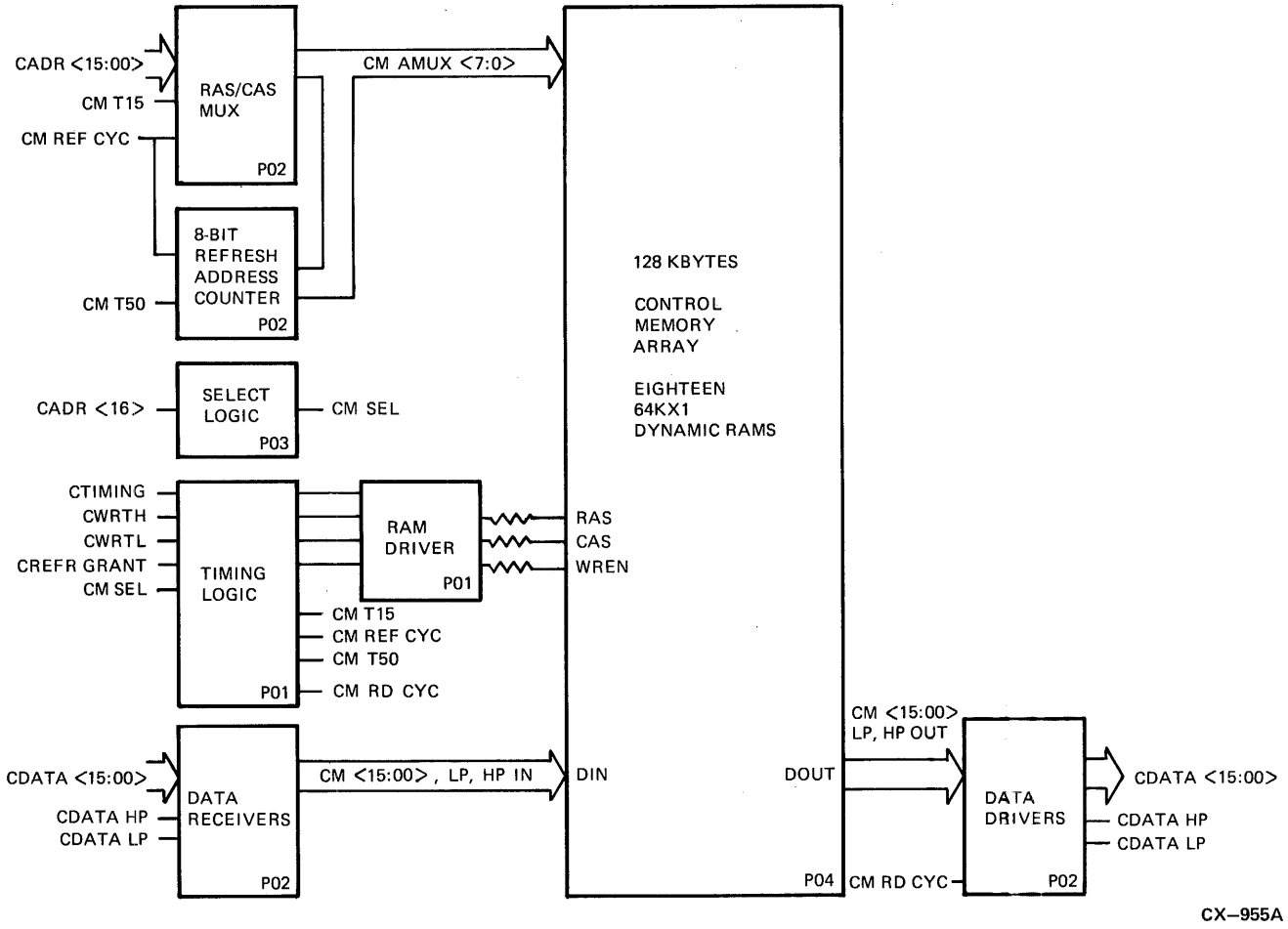
The yellow LED on the Memory Module indicates memory activity. It lights on every memory cycle to Control, Data, or Program Memory. The LED becomes brighter with increased memory activity.

## 10.3 CONTROL MEMORY

Control Memory, which resides on the Control Bus, performs a read or write cycle as determined by the requestor that has been granted the Control Bus for that cycle. To ensure data retention of the dynamic RAMs, Control Memory also performs refresh cycles once every 15 microseconds. It is logically organized as one bank that is 18-bits wide (16 data bits and 2 parity bits) and 64 Kwords deep.

Figure 10-2 is a block diagram of Control Memory. The following sections describe each block in the diagram.

Figure 10-2 Control Memory Block Diagram



### 10.3.1 RAS/CAS Multiplexor

The RAS/CAS Multiplexor supplies row and column addresses to the RAMs during read and write cycles. It receives the desired Control Memory address from the active requestor on the CADR <15:00> signal lines. The outputs of the RAS/CAS Multiplexor go to the RAMs themselves and supply the RAMs with the desired address.

NOTE

**The 16 bits of CADR <15:00> address all 128 Kbytes, or 64 Kwords. The 16-bit address is on word boundaries. Each read from Control Memory reads a 16-bit word. Therefore, the 16 bits select a word. The P.ioc shifts its address right one bit before it puts the address on the CADR. The Ks always do a word write. The P.ioc can do a byte write using the CWRTH and CWRTL signals.**

During a refresh cycle, the signal CM REF CYC disables the output of this multiplexor. The row and column address signals are supplied by the 8-Bit Refresh Address Counter located below the multiplexor.

### 10.3.2 8-Bit Refresh Address Counter

The 8-Bit Refresh Address Counter supplies refresh row addresses to the RAMs during refresh cycles. The signal CM REF CYC increments the counter and also enables the counter outputs. The outputs of the counter are connected to the RAS/CAS Multiplexor output lines. The outputs of the multiplexor are disabled during refresh cycles.

### 10.3.3 Select Logic

The Select Logic monitors CADR <16> and generates CM SEL if CADR <16> is zero.

NOTE

**The Select Logic contains other logic for the implementation of more than one Memory Module and more than one bank of memory.**

### 10.3.4 Timing Logic and RAM Drivers

The Timing Logic circuitry generates the required RAM timing and bus interface timing for Control Memory. CTIMING is the clock from the P.ioc that starts the timing circuitry for Control Memory. The timing is provided by a tapped delay line. The signals CWRTH and CWRTL, from the active requestor, determine if the cycle is to be a read or a write. Also CWRTH (high byte) and CWRTL (low byte) determine which byte is written for a write cycle. The signal CREFR GRANT determines if the cycle is to be a refresh cycle. CM SEL is asserted if the address is for the Control Memory.

The RAM Drivers supply the RAMs the row address strobe (CM RAS 0 and CM RAS 1), column address strobe (CM CAS 0), and the two write enables (CM WRT HB and CM WRT LB). The write enables allow either byte or word writes.

NOTE

**The design allows for two banks of Control Memory, so there are signals that control both banks. However, with this configuration of 128 Kbytes, only Bank 0 is used.**

### 10.3.5 Data Receivers

The Data Receivers supply data to the RAMs on a write cycle. The receivers are always enabled. The data to be written, as well as the high and low parity bits, are present on the CDATA lines. The receiver output lines (CM <15:00>,HP,LP IN) drive the Data-In lines (DIN) of the RAMs.

### 10.3.6 Data Drivers

The Data Drivers supply data to the Control Bus out of RAM on a read cycle. They are enabled by the signal CM RD CYC. When enabled, the data to be read is taken from the RAM outputs (CM <15:00>,HP,LP OUT), through the drivers to their outputs, and out on the backplane to the active requestor.

### 10.3.7 Control Memory Cycles

During a Control Bus cycle, the Control Memory is executing one of four cycles:

Idle
Write
Read
Refresh

The P.ioc decodes the CCYCLE lines on the Control Bus and determines if the cycle is a write or read. The P.ioc initiates the refresh cycle. The following sections explain each cycle type. Refer to the timing diagrams supplied for read, write, and refresh.

#### 10.3.7.1 Control Memory Idle Cycle

An idle cycle occurs when there is no active requestor or when a Non-Memory Access (NMA) cycle or an Interrupt (INTR) cycle occurs. The Control Memory remains idle because the P.ioc does not generate CTIMING. Therefore Control Memory does not generate CM SEL or the acknowledge (CACK) for any NMA, INTR, or idle Control Bus cycle.

#### 10.3.7.2 Control Memory Write Cycle

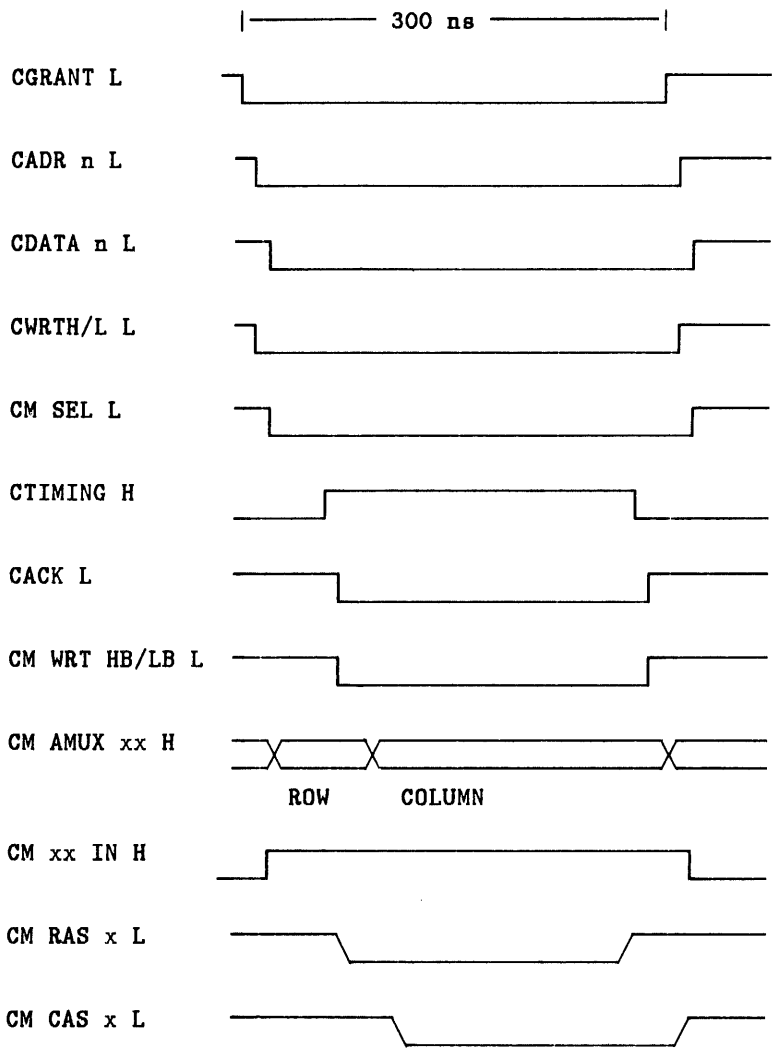
The Control Memory Write Cycle cannot begin until a requestor has been granted bus control with CGRANT. The P.ioc Control Bus Arbitration Logic acknowledges the request with CGRANT and the requestor becomes active for that cycle.

Refer to Figure 10-3 for a timing diagram of the Control Memory Write Cycle.

The following list describes the Control Memory Write Cycle:

1. When CGRANT is asserted, the active requestor asserts:
   - The address to be written on the CADR lines which are inputs to the CAS/RAS Multiplexor and the Select Logic
   - The data to be written on the CDATA lines which are inputs to the Data Receivers asserting CM xx IN
   - The command to write, CWRTH and/or CWRTL, generates the WREN for the RAMs
2. When the Control Memory receives the assertion of CTIMING from the P.ioc, the memory begins its cycle
3. The acknowledge signal, CACK, is asserted to tell the active requestor the selected address is an existing address in Control Memory
4. The output of the RAS/CAS Multiplexor (CM AMUX xx) enables the row address to the RAMs
5. The signal CM RAS is asserted

Figure 10-3 Control Memory Write Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any given time. The lines here are shown in the high state.

CX-1151A

6. The RAS/CAS Multiplexor output enables the column address to the RAMs 15 nanoseconds later
7. The signal CM CAS is then asserted to strobe the data and parity into the RAM from the active requestor on the CDATA lines of the data receivers

#### 10.3.7.3 Control Memory Read Cycle

The Control Memory Read Cycle cannot begin until a requestor has been granted bus control with CGRANT. The P.ioc Control Bus Arbitration Logic acknowledges the request with CGRANT, and the requestor becomes active for that cycle.

Refer to Figure 10-4 for a timing diagram of the Control Memory Read Cycle.

The following list describes the Control Memory Read Cycle:

1. When CGRANT is asserted, the active requestor asserts the address to be read on the CADR lines which are inputs to the RAS/CAS Multiplexor
2. When the Control Memory receives the assertion of CTIMING from the P.ioc, the memory cycle begins
3. Neither CWRTH or CWRTL being asserted indicates a read cycle
4. The CADR lines are gated through the RAS/CAS Multiplexor and generate CM AMUX xx H for the row address
5. CTIMING generates:
   - CM RD CYC
   - CM RAS
6. The column address is generated from the CADR lines 15 nanoseconds later
7. CM CAS is then generated
8. The data is read out of the RAM and is enabled onto the Control Bus by CM RD CYC

#### 10.3.7.4 Control Memory Refresh Cycle

The dynamic RAMs used in Control Memory require that all rows within the RAM be refreshed every 4 milliseconds to ensure data retention. A refresh timer circuit on the P.ioc provides a signal called CREFRESH CLK approximately every 15 microseconds to ensure that all 128 rows are refreshed within the required 2-millisecond period.

Refresh requests are given the highest priority in the arbitration logic on the P.ioc. No data is transferred across the Control Bus during a refresh cycle, nor does Control Memory generate CACK or drive the CDATA lines.

The arbitrator asserts CREFR GRANT, notifying the Control Memory that a refresh cycle is to be performed. The refresh cycle begins when CTIMING is asserted. The 8-Bit Refresh Address Counter supplies the address of the row to be refreshed. The counter is enabled by CM REF CYC and is incremented by the negation of the CM REF CYC signal. (CM REF CYC is the complement of CREFR GRANT from the Timing Logic.)

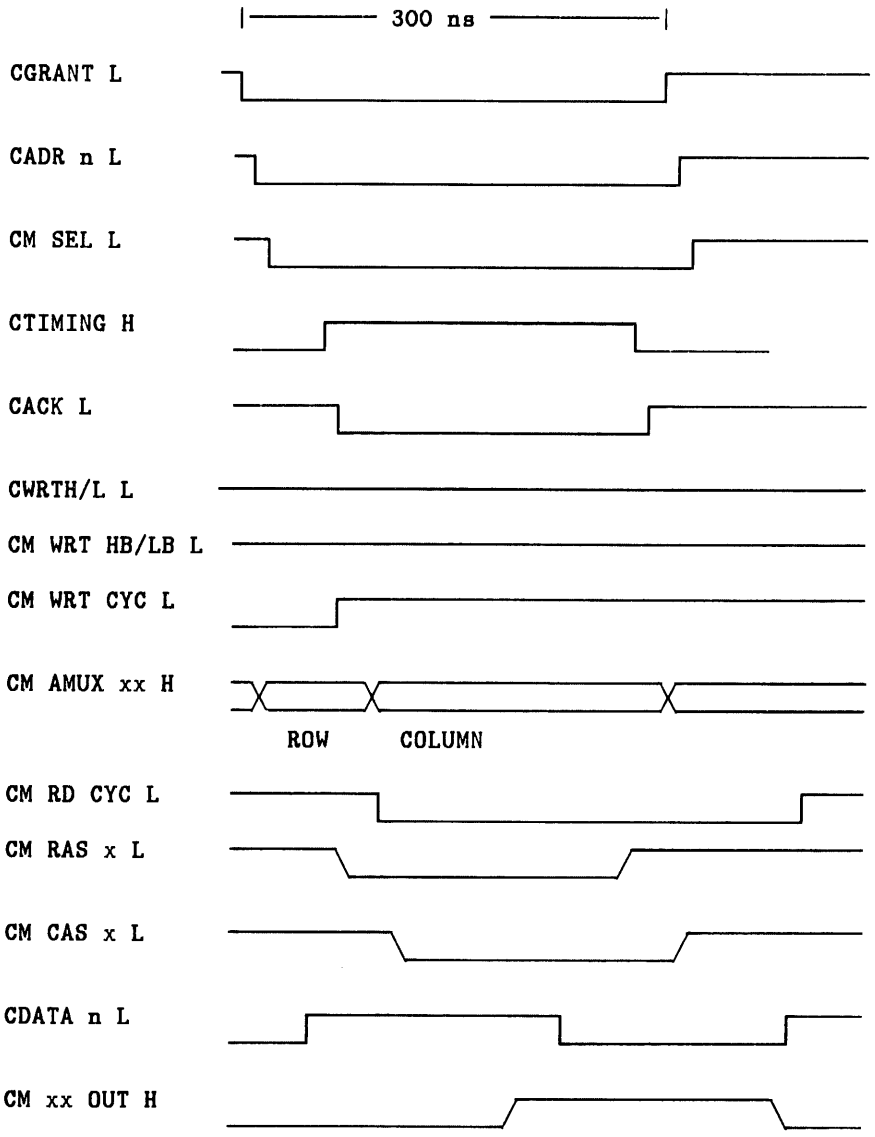
Refer to Figure 10-5 for a timing diagram of the Control Memory Refresh Cycle.

### 10.4 DATA MEMORY

Data Memory is 128 Kbytes of memory and resides on the Data Bus. Data Memory uses 16Kx1 static RAMs because of the high speed requirements of the Data Bus. It is organized as 4 banks of 18 RAMs apiece designated as banks zero, one, two, and three. Each bank is 18 bits wide (16 data bits plus 2 parity bits) and 16 Kwords deep. Either data byte within a word may be written separately using the signals DWRTH and DWRTL.

Data Memory is used by both the P.ioc and the Ks to perform data transfers. All data which flows through the HSC50 subsystem goes through the Data Memory.

Figure 10-4 Control Memory Read Cycle Timing

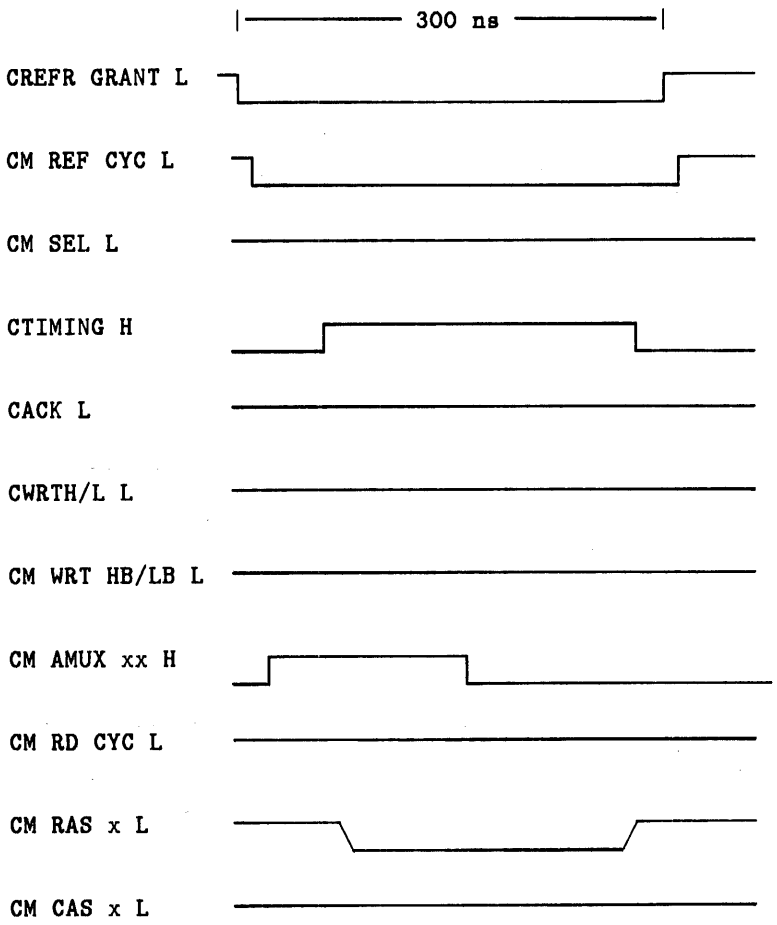



NOTE: The address or data lines can be in the high state or low state at any given time. The lines here are shown in the high state.

CX–1149A

The Data Memory block diagram (Figure 10-6) shows the components of Data Memory. The following sections describe each of those components.

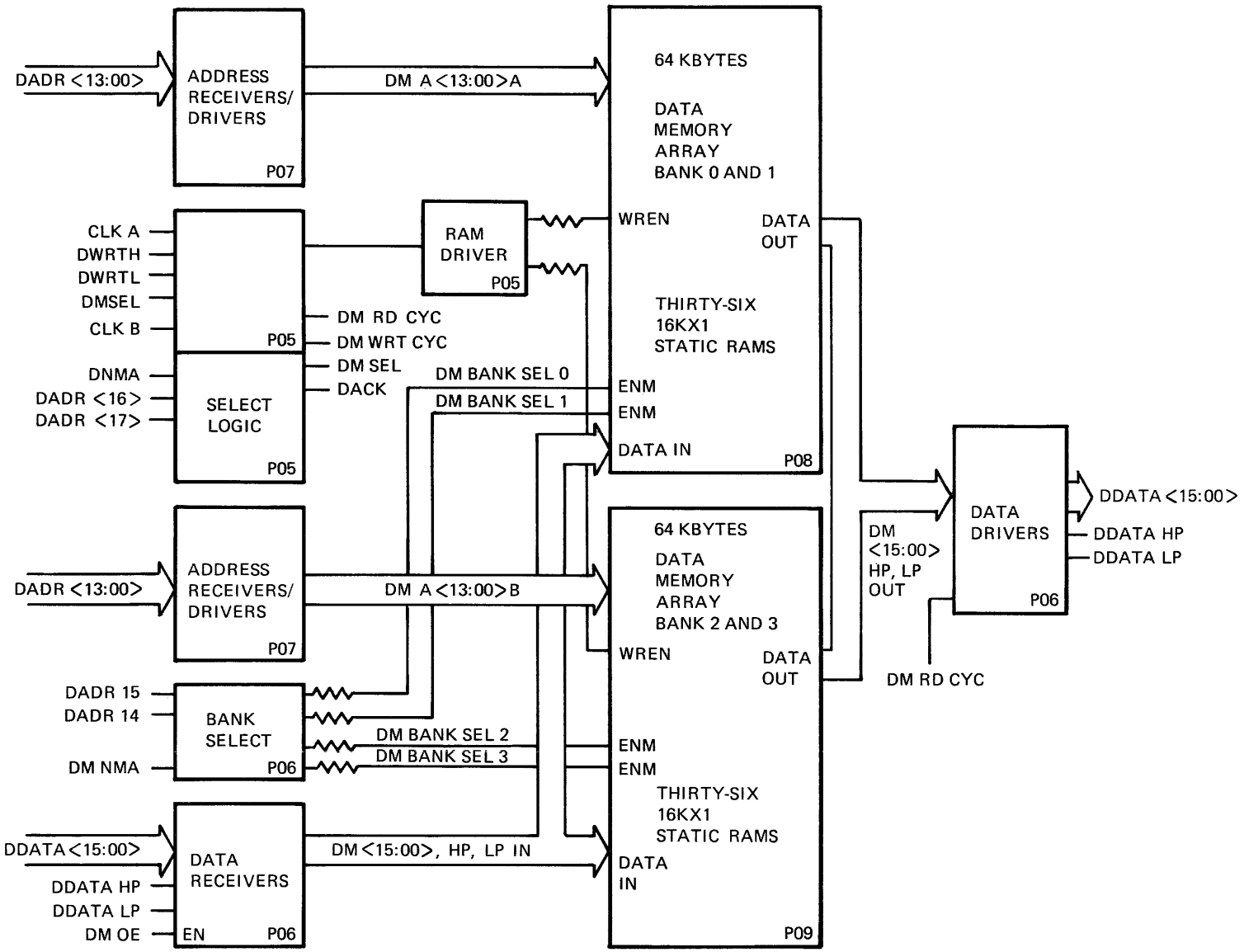
Figure 10-5 Control Memory Refresh Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any given time. The lines here are shown in the high state.

CX-1150A

Figure 10-6 Data Memory Block Diagram



### 10.4.1 Address Receiver/Driver

The Data Memory Address Receiver/Drivers supply addresses to the RAMs for read and write cycles. The inputs to the receivers are DADR <13:00>. The drivers in the upper left of Figure 10-6 provide DM A<13:00> A to Banks zero and one. The drivers in the middle of the figure provide DM A<13:00> B to Banks two and three. The receivers/drivers are always enabled.

**NOTE**

**DADR <15:14> are inputs to the Bank Select logic.**

### 10.4.2 Timing Logic and RAM Drivers

The Data Memory Timing Logic generates the required RAM timing and bus interface timing for Data Memory. The Timing Logic has delay lines that provide discrete timing signals; CLK A and CLK B from the P.ioc are inputs to these delay lines. The Data Bus cycles are synchronized by the requestor having been granted control of the bus with the signal CLK A or CLK B which have 150 nanoseconds periods.

The Ram Driver drives the write signals DWRTH and DWRTL to the RAMs for writing either or both bytes.

### 10.4.3 Select Logic

The Select Logic compares the address on the DADR lines and generates DM SEL and DACK if the address is in the range of the installed memory. DNMA H disables the select logic because Data Memory is not selected for a non-memory access cycle. (The requestors use the non-memory access cycle to check the operation of the DADR and DDATA lines.)

### 10.4.4 Bank Select

The Bank Select logic selects one of four banks of RAM available to Data Memory. It is enabled by the negation of DM NMA (a non-memory access cycle is not in progress). Signals DADR 15 and DADR 14 select one of the four banks as shown in Table 10-2.

Table 10-2 Data Memory Bank Selection

| DADR 15 | DADR 14 | Bank Selected |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

### 10.4.5 Data Receivers

The Data Receivers supply data to the RAMs on a write cycle. They are always enabled by the signal DM OE. The data and parity bits to be written are driven onto the DDATA <15:00>,HP,LP lines by the active requestor and passed through the driver to the RAM inputs DM <15:00>,HP,LP IN.

### 10.4.6 Data Drivers

The Data Drivers supply data to the Data Bus out of RAM on a read cycle. The drivers are enabled only on a read cycle by the signal DM RD CYC. When the drivers are enabled, the data as well as the parity bits are driven onto the Data Bus lines, DDATA <15:00>,HP,LP.

### 10.4.7 Data Memory Cycles

During any given Data Bus cycle, Data Memory is executing one of three cycles:

Idle
Write
Read

The static RAMs used in Data Memory do not require refresh cycles. Each of the three cycles is covered in the following sections. Refer to the timing diagrams supplied for the read and write cycles.

#### 10.4.7.1 Idle Cycle

An idle cycle occurs when there is no active requestor or when an NMA (non-memory access) cycle occurs. During these cycles the signal DNMA H forces the Data Memory select logic not to select Data Memory and forces the bank select logic not to select any bank of Data Memory.

#### 10.4.7.2 Write Cycle

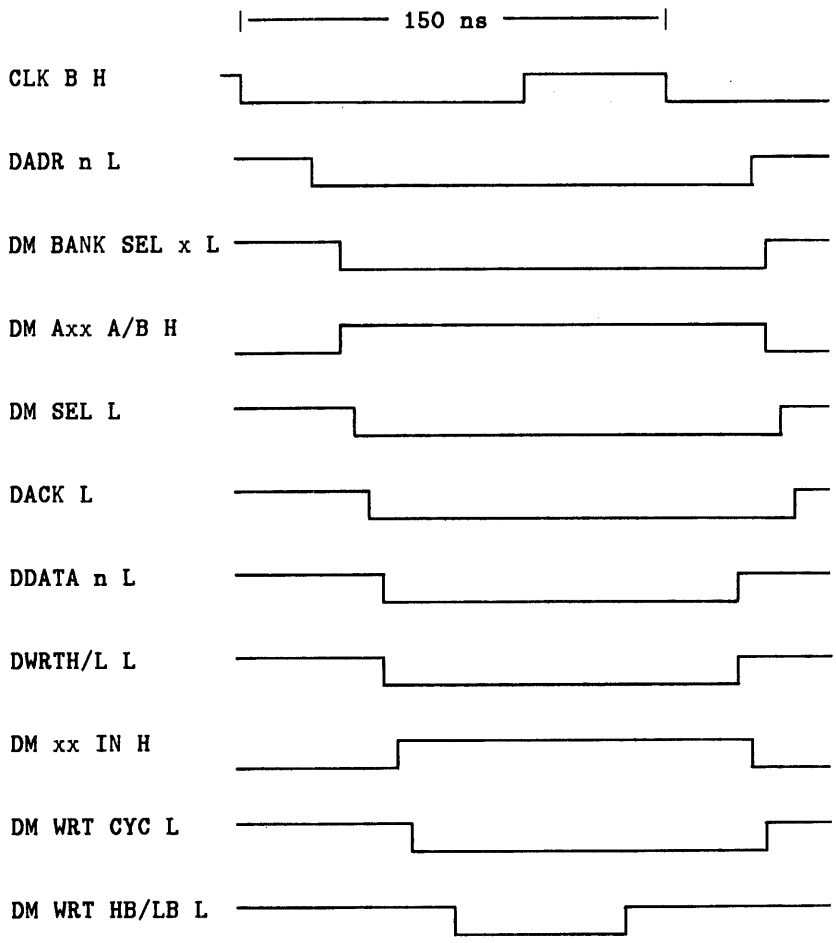
To write to Data Memory, a requestor must first acquire the Data Bus by asserting its request to the bus arbitrator on the P.ioc. Once the requestor receives the assertion of DGRANT from the P.ioc, it has possession of the Data Bus for that cycle.

Refer to Figure 10-7 for a timing diagram of the Data Memory write cycle.

The following list describes the Data Memory write cycle:

- After the P.ioc asserts DGRANT, the high to low transition of the CLK B H signal clocks the active requestor's DGRANT flip-flop asserting the address to be written on the DADR lines
- With the assertion of the DADR lines:
  - DM BANK SEL x is asserted from the Bank Select logic
  - DM Axx A/B is asserted from the Address Receivers/Drivers
  - DM SEL is asserted from the Select Logic
  - DACK is asserted from the Select Logic
- The requestor then asserts the DDATA lines and DWRTH/L lines
- The Data Receivers assert DM xx IN to the RAMs
- The Timing Logic asserts DM WRT CYC from DWRTH/L
- The RAM Driver asserts DM WRT HB/LB
- The data is written into the RAMs

Figure 10-7 Data Memory Write Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any given time. The lines here are shown in the high state.

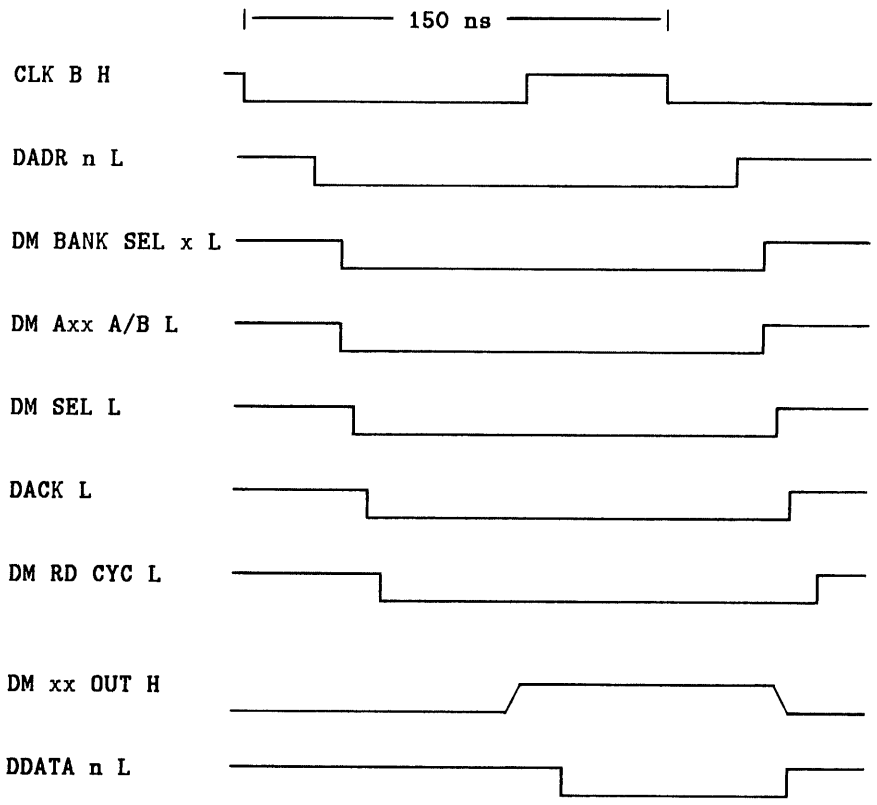
CX-1153A

### 10.4.7.3 Data Memory Read Cycle

To read from Data Memory, a requestor must first become active by asserting its request to the P.ioc arbitration logic and receiving the asserted DGRANT signal.

Refer to Figure 10-8 for a timing diagram of the Data Memory read cycle.

Figure 10-8 Data Memory Read Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any given time. The lines here are shown in the high state.

CX-1152A

The following list describes the Data Memory read cycle:

- After the P.ioc asserts DGRANT, the high to low transition of the CLK B H signal clocks the active requestor's DGRANT flip-flop asserting the address to be written on the DADR lines
- With the assertion of the DADR lines:
  - DM BANK SEL x is asserted from the Bank Select logic

    - DM Axx A/B is asserted from the Address Receivers/Drivers
    - DM SEL is asserted from the Select Logic
    - DACK is asserted from the Select Logic
- The DM RD CYC signal is asserted from the Timing Logic
- The outputs from the RAMs are enabled through the Data Drivers by DM RD CYC
- When CLK B goes HI, the data is strobed into the receivers on the requestor

## 10.5 PROGRAM MEMORY

The P.ioc loads the operational and diagnostic software into Program Memory via the TU58 tape unit. The P.ioc is the only module that uses Program Memory.

Program Memory uses 36 dynamic 64Kx1 RAMs logically organized as two banks (bank 0 and bank 1). Each bank is 18 bits wide (1 parity bit per byte) and 64 Kwords deep. The P.ioc can assign the starting address of $00000000_8$ to either bank 0 or bank 1. This capability allows the P.ioc to operate even if one bank fails (only 128 Kbytes). Remember that the F-11 processor on the P.ioc requires the standard PDP-11 trap and interrupt vectors to start at address $00000000_8$. However, the P.ioc does not dynamically swap banks when the HSC50 is online.

The Program Memory block diagram (Figure 10-9) shows the components of the memory. The following sections describe each of these components.

### 10.5.1 BDAL Buffer

The BDAL Buffer receives the address or data on the BDAL lines. PM SYNC clocks the address into the CAS and RAS Address Driver. After PM SYNC latches the address into the CAS and RAS Address Driver, the BDAL Buffer receives the data for the RAMs if a write cycle is to be performed to memory. The data is then available at the DIN input of the memory array.

### 10.5.2 CAS and RAS Address Drivers

The CAS and RAS Address Drivers supply row and column addresses to the RAM Address Driver. The input signals are:

PM <15:00> IN—address from BDAL Buffer
PM SYNC—clock
PM COL ADR EN—enable for CAS Address Driver
PM ROW ADR EN—enable for RAS Address Driver

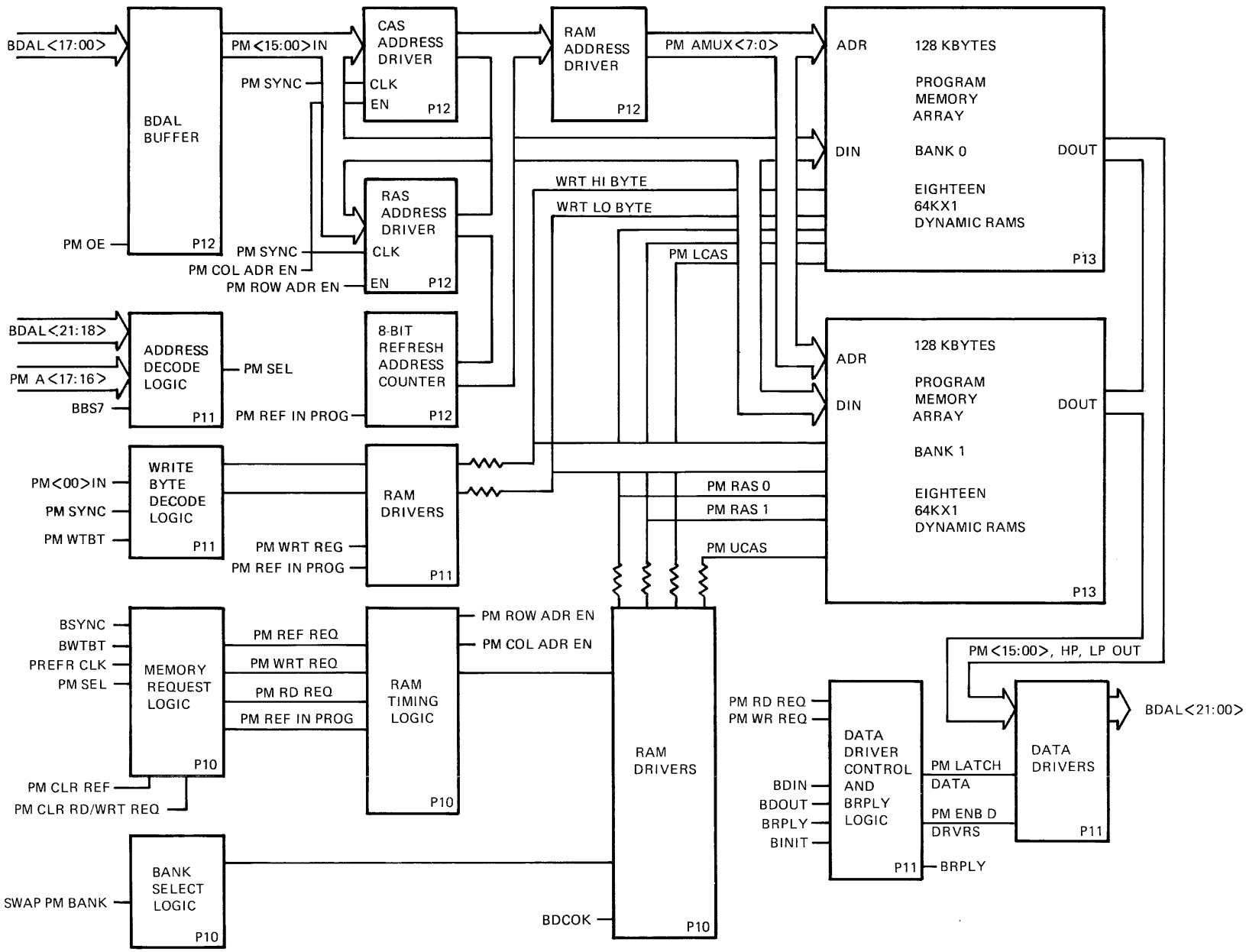
### 10.5.3 RAM Address Driver

The RAM Address Driver supplies the addresses to RAM for read, write, and refresh cycles. The CAS and RAS Address Drivers supply the saved address during a write or read cycle. The 8-Bit Refresh Address Counter supplies the address during a refresh cycle.

### 10.5.4 8-Bit Refresh Address Counter

The 8-Bit Refresh Address Counter supplies the refresh row address to the RAMs during refresh cycles. A refresh cycle disables the outputs of the CAS and RAS Address Drivers. The signal PM REF IN PROG enables the refresh counter output and increments the counter at the end of the refresh cycle.

**Figure 10-9 Program Memory Block Diagram**




CX-958A

### 10.5.5 Address Decode Logic

The Address Decode Logic selects program memory for address combinations that do not assert BBS7 and BDAL <21:18>. The output is PM SEL.

### 10.5.6 Write Byte Decode and RAM Drivers

The Write Byte Decoder supplies the inputs to the RAM Drivers depending on the state of PM 00 IN and PM WTBT. If the P.ioc does not assert PM WTBT (write byte), both bytes of the word are written into Program Memory. If the P.ioc asserts PM WTBT, the PM 00 IN (bit 00 of the address) signal selects which byte (HI or LO) is written into Program Memory.

The RAM Driver provides the write enable to the RAMs. If a refresh cycle is not in progress (PM REF IN PROG), PM WRT REQ enables the output of the RAM Driver.

### 10.5.7 Memory Request Logic

The Memory Request Logic consists of several flip flops servicing the types of requests to which Program Memory can respond. The valid requests for Program Memory are:

Read
Write
Read-modify-write
Refresh

### 10.5.8 RAM Timing Logic

The RAM Timing Logic provides the timing for the Program Memory via a delay line.

### 10.5.9 Bank Select Logic and Program Memory RAM Driver

The Bank Select Logic selects which of the two banks of Program Memory has a starting address of $00000000_8$. The signal SWAP PM BANK from the P.ioc selects that bank.

The RAM Drivers supply the CAS and RAS signals to the RAMs. The output from the RAM Timing Logic generates PM RAS 0 (low byte) and PM RAS 1 (high byte). Depending on the output from the Bank Select Logic, the RAM driver asserts PM LCAS (bank 0) or PM UCAS (bank 1).

### 10.5.10 Data Driver Control and BRPLY Logic

The Data Driver Control and BRPLY Logic control the reading and writing of data through the Program Memory data drivers. The following are the input signals:

PM RD REQ—read request
PM WR REQ—write request
BDIN—read control
BDOUT—write control

The output signal PM LATCH DATA clocks the data from the RAMs into the Data Drivers. The signal PM ENB D DRVRS enables the output of the Data Drivers onto the BDAL lines.

The BRPLY Logic asserts the signal BRPLY at the completion of a memory cycle. The signal BDIN gates BRPLY for a read cycle. The signal BDOUT gates BRPLY for a write cycle.

### 10.5.11 Data Drivers

The Data Drivers supply data to the Program Bus during a a read cycle. The signal PM LATCH DATA clocks the data from the RAMs into the Data Drivers. The signal PM ENB D DRVRS enables the output of the Data Drivers onto the BDAL lines. On a read cycle, the BDAL <17:00> L lines receive the data plus the parity bits, PM <15:00>,HP,LP OUT, from the RAMs.

### 10.5.12 Program Memory Cycles

As stated earlier, Program Memory performs the following four cycles:

Write
Read
Read-modify-write
Refresh

The following sections describe these cycles.

#### 10.5.12.1 Program Memory Write Cycle

Refer to Figure 10-10 for a timing diagram of the Program Memory write cycle.

The following list describes the Program Memory Write Cycle:

1. The P.ioc asserts the address on the BDAL lines while asserting BWTBT (write cycle).
2. The signal BSYNC latches the address into the CAS and RAS Address Drivers (PM SYNC).
3. The RAM Timing Logic asserts PM ROW ADR EN to enable the output of the RAS Address Driver.
4. The P.ioc asserts the write data on the BDAL lines.
5. The P.ioc asserts or negates BWTBT, depending on the type of cycle (word or byte).
6. The signal BDOUT initiates the Program Memory write cycle by generating PM WRT REQ.
7. The RAM Timing Logic generates the timing for the RAM Drivers that assert PM RAS 0 and PM RAS 1.
8. The RAM Timing Logic negates PM ROW ADR EN, disabling the output of the RAS Address Driver.
9. The RAM Timing Logic asserts PM COL ADR EN, enabling the output of the CAS Address Driver.
10. The RAM Timing Logic generates the timing for the RAM Drivers asserting PM CAS.
11. The Write Byte Decode Logic, through the RAM Drivers, generates the write enable (WRT HI BYTE and WRT LO BYTE) for the RAMs.
12. The Data Driver Control and BRPLY Logic negate BRPLY, finishing the write cycle.

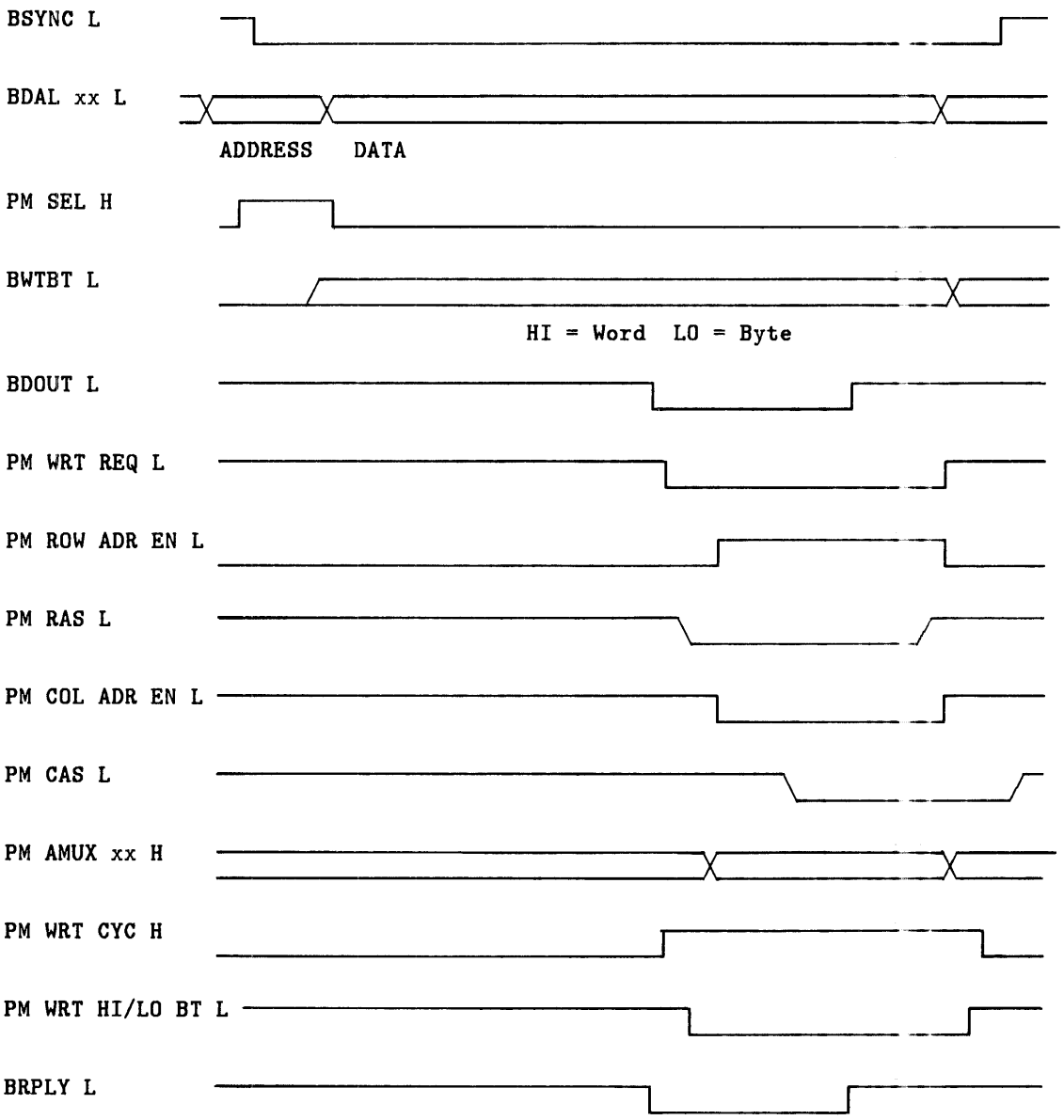
#### 10.5.12.2 Program Memory Read Cycle

Refer to Figure 10-11 for a timing diagram of the program memory read cycle.

The following list describes the Program Memory Read Cycle:

1. The P.ioc asserts the read address on the BDAL lines while negating BWTBT (read cycle).
2. The signal BSYNC latches the address into the CAS and RAS Address Drivers (PM SYNC).
3. The Memory Request Logic asserts PM RD REQ when the P.ioc asserts BSYNC.
4. The RAM Timing Logic asserts PM ROW ADR EN, enabling the output of the RAS Address Driver.
5. The RAM Timing Logic generates the timing necessary for the RAM Drivers to assert PM RAS 0 and PM RAS 1.
6. The RAM Timing Logic negates PM ROW ADR EN, disabling the output of the RAS Address Driver.

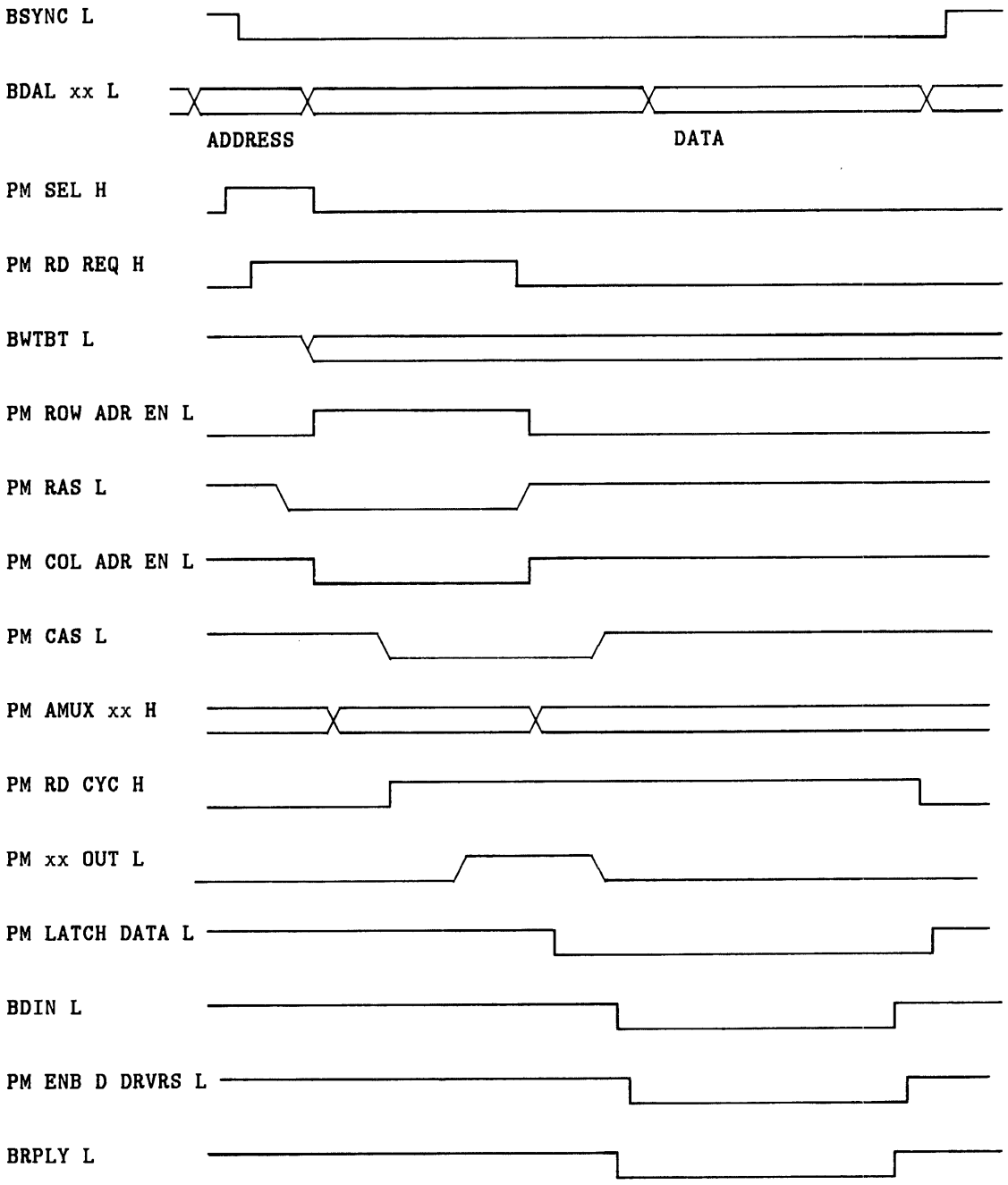
Figure 10-10 Program Memory Write Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any given time. The lines here are shown in the high state.

CX-1156A

Figure 10-11 Program Memory Read Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any given time. The lines here are shown in the high state.

CX–1154A

7. The RAM Timing Logic asserts PM COL ADR EN, enabling the output of the CAS Address Driver.
8. The RAM Timing Logic generates the timing necessary for the RAM Drivers to assert PM CAS.
9. PM LATCH DATA latches the data output of Program Memory (PM xx OUT) into the Data Drivers.
10. The signal BDIN asserts PM ENB D DRVRS, enabling the data from the Data Drivers onto the BDAL lines.
11. The Data Driver Control and BRPLY Logic negate BRPLY, finishing the read cycle.

#### 10.5.12.3 Program Memory Read-Modify-Write Cycle

Program Memory is capable of performing a read-modify-write cycle to the same address. The read-modify-write cycle is a single bus cycle that generates two memory cycles. This cycle is similar to a read cycle followed by a write cycle. However, in a read-modify-write cycle, the address is sent to the memory only once (during the read cycle). During the write cycle, the memory uses the address it latched into the CAS and RAS Address Drivers.

The P.ioc asserts BSYNC L during the entire cycle.

Although this cycle is a continuous cycle on the Program Bus, a refresh cycle can occur between the read and the write cycles. The refresh cycle will delay the write cycle.

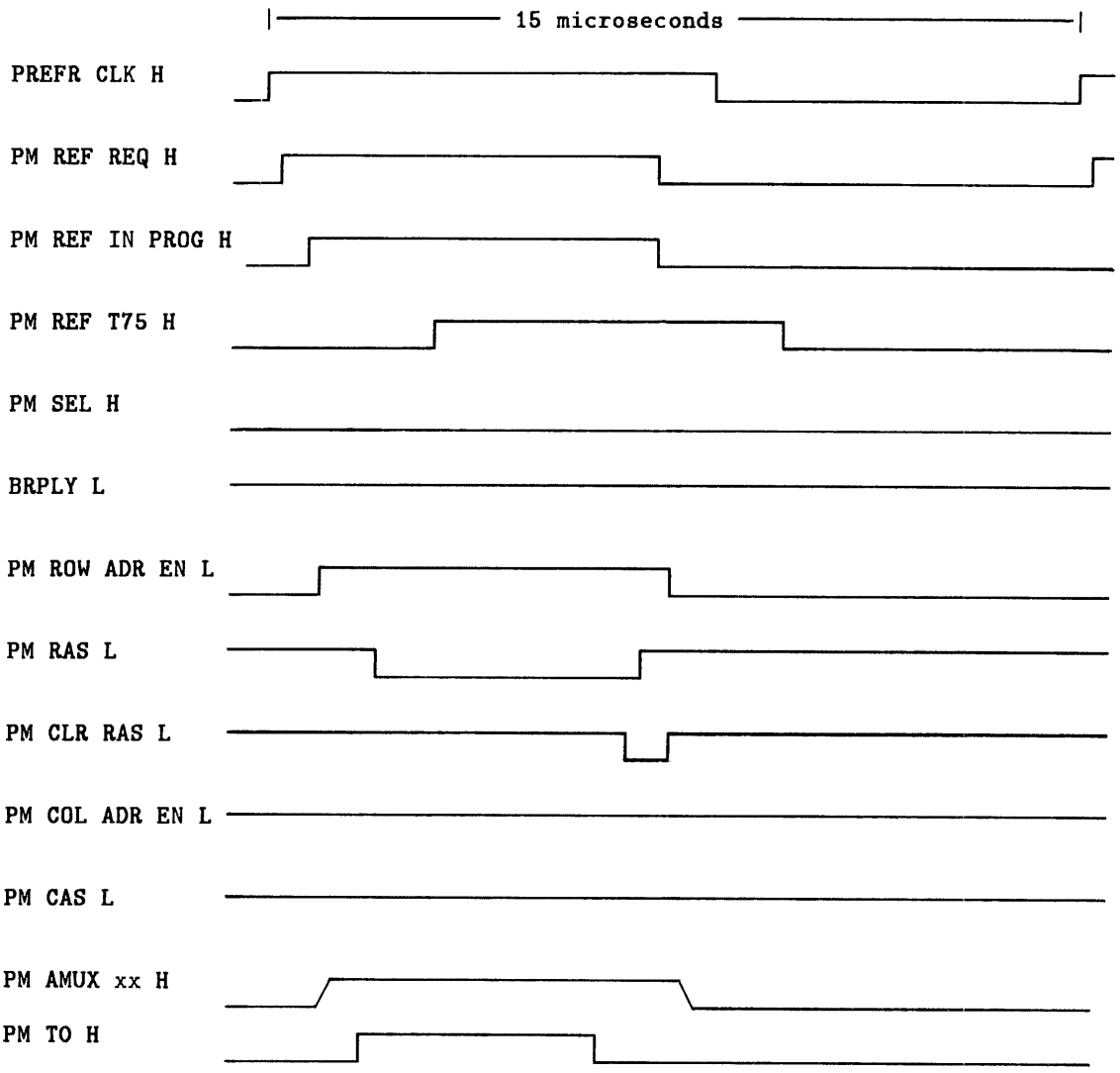
#### 10.5.12.4 Program Memory Refresh Cycle

Program Memory RAMs require refresh cycles to ensure data retention in the dynamic RAMs. The overflow of the counter on the P.ioc asserts PREFR CLK every 15 microseconds to the Program Memory. If a read or write cycle is in progress when the P.ioc asserts PREFR CLK, the Program Memory does not initiate the refresh cycle until the read or write cycle finishes. Also, the program memory will not initiate a read or write cycle until any refresh cycle in progress is complete.

The 8-bit Refresh Address Counter provides the refresh row address. The end of the refresh cycle increments the counter. A refresh cycle does not use the column address strobe (CAS).

Refer to Figure 10-12 for a timing diagram of the Program Memory refresh cycle.

Figure 10-12 Program Memory Refresh Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any given time. The lines here are shown in the high state.

CX–1155A

## 10.6 JUMPER AND DIPSHUNT CONFIGURATION

There are various jumpers and dipshunts on the HSC50 Memory Module. Table 10-3 describes the purpose and normal configuration of each jumper and dipshunt for the HSC50 Memory Module.

Table 10-3 HSC50 Memory Module Jumper and Dipshunt Configuration

| Jumpers or Dipshunts | Normal Configuration | Description |
|---|---|---|
| W1 | Out | Install for 16K RAMs in Control Memory |
| W2 | Out | Install for 16K RAMs in Control Memory |
| W3 | In | Install for 1 bank of 64K RAMs in Control Memory |
| W4 | Out | Install for 2 banks of 64K RAMs in Control Memory |
| W5 | In | Install for 64K RAMs in Control Memory |
| W6 | Out | Install for 16K RAMs in Control Memory |
| W7 | Out | Install for 16K RAMs in Control Memory |
| W8 | In | Install for 1 or 2 banks of 16K RAMs or 1 bank of 64K RAMs in Control Memory |
| W9 | In | Install for 64K RAMs in Control Memory |
| W10 | In | Install for 64K RAMs in Control Memory |
| E2 | In | Install 8-position dipshunt for 64KW in Data Memory |
| E8 | Out | Install 8-position dipshunt for 16KW in Data Memory |
| E33 | Out | Install 8-position dipshunt for 32KW in Data Memory |
| E112 | Out | Install 5-position dipshunt for 16KW in Control Memory |
| E113 | In | Install 5-position dipshunt for 32KW, 64KW, or 128KW in Control Memory |
| E223 | Out | Install 5-position dipshunt for 64KW in Program Memory |
| E229 | Out | Install 8-position dipshunt for 32KW in Program Memory |
| E239 | In | Install 8-position dipshunt for 64KW or 128KW in Program Memory |
| E242 | In | Install 5-position dipshunt for 32KW or 128KW in Program Memory |

# CHAPTER 11
# HSC70 MEMORY MODULE

## 11.1 INTRODUCTION

The HSC70 memory module, M.std2 (Memory Standard 2), is an extended-hex size module residing in slot two of the HSC70, adjacent to the J-11 I/O Control Processor Module (P.ioj). The module number L0117 is stamped on the handle. The M.std2 contains three separate and independent memories, each on its own bus, as well as a controller for the RX33 floppy disk drives. The three memories and the floppy disk controller are :

- Control Memory or M.ctl (residing on the Control Bus)
- Data Memory or M.data (residing on the Data Bus)
- Program Memory or M.prog (residing on the Program Bus)
- Floppy Disk Controller or K.rx (residing on the Program Bus)

Each of the memory arrays includes byte parity.

Refer to Figure 11-1 for a block diagram of the M.std2 module and its four components.

The three memories each use different RAM types and quantities as follows:

- M.ctl = 256 Kbytes (128 Kwords) uses 18 256Kx1 dynamic RAMs
- M.data = 512 Kbytes (256 Kwords) uses 72 64Kx1 static RAMs
- M.prog = 1 Mbyte (512 Kwords) uses 36 256Kx1 dynamic RAMs

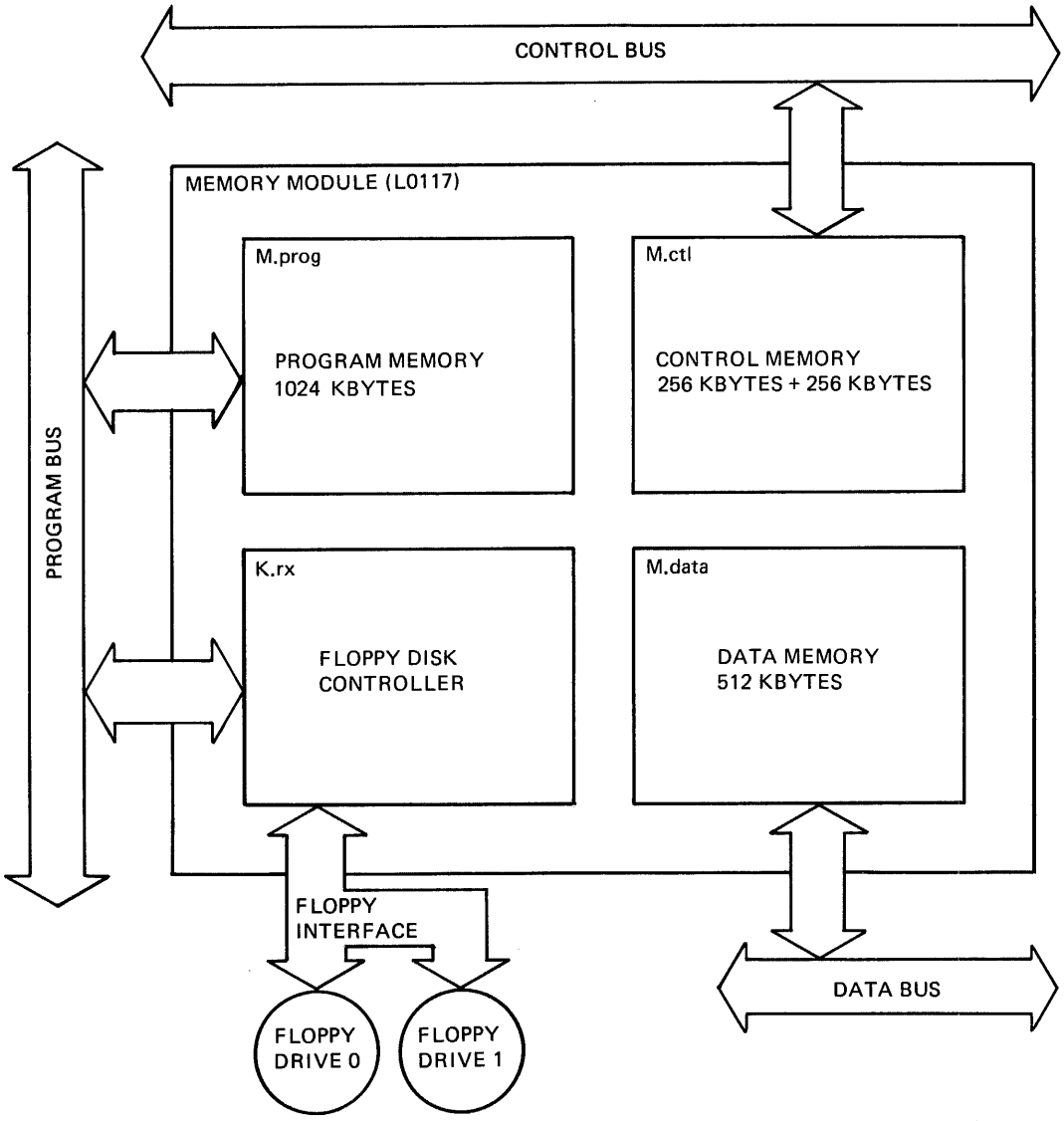
## 11.2 STATUS LEDs

There are three LEDs on the Memory Module. They display the status of the M.std2.

The yellow LED is the memory active LED. It lights on every memory cycle to Control, Data, or Program Memory. The the yellow LED becomes brighter with increased memory activity.

The red and green LEDs are controlled by the Module OK bit in the MAR2 (Memory Address Register 2) of the floppy controller. Setting the bit will turn on the green LED (module OK) and turn off the red LED (module BAD). Following power up, the bit will be cleared and the red LED will be on. After passing diagnostics, the green LED comes on (module OK bit). The memory address register will be discussed in Section 11.7.1.4.

Figure 11-1 M.std2 Block Diagram



## 11.3 CONTROL MEMORY

Refer to Figure 11-2 and to the print set while reading the following.

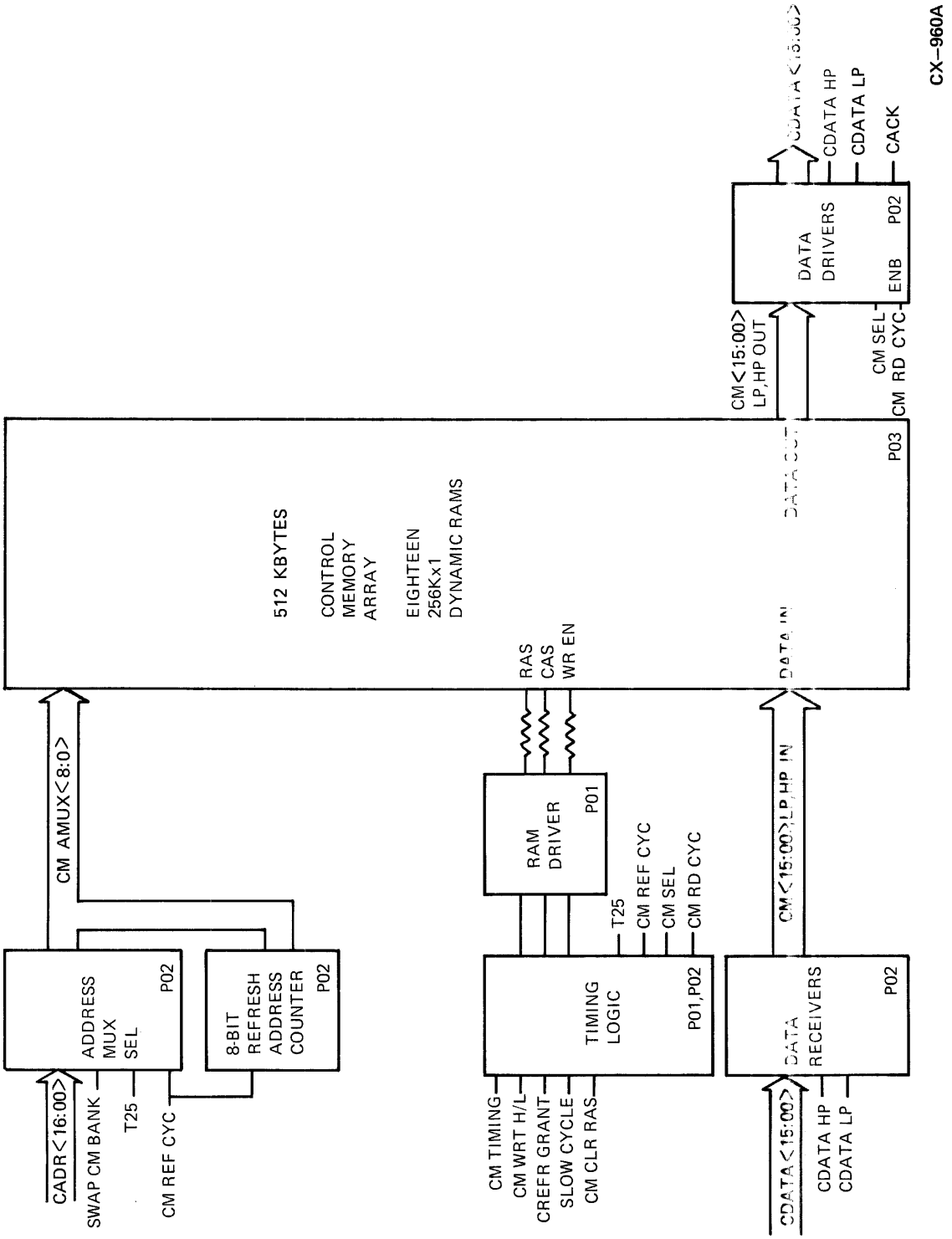



**Figure 11-2 Control Memory Block Diagram**

Control Memory, residing on the Control Bus, performs a read or a write cycle as determined by the requestor that has been granted the Control Bus for that cycle. To ensure data retention of the dynamic RAMs, control Memory also performs refresh cycles once every 15 microseconds. Control Memory is logically organized as one bank 18 bits wide and 128 Kwords deep. Since there are eighteen 256Kx1 dynamic RAMs in this design, there is a second bank of 128 Kwords contained within the same RAMs. This second bank may only be accessed by the P.ioj asserting the signal SWAP CM BANK H.

In use, the M.std2 stores 16 data bits and 2 parity bits which have identical paths for both data and timing. In addition, either data byte may be written independently.

The following sections describe each of the components shown in the Control Memory block diagram.

### 11.3.1 Address Multiplexer

The Control Memory Address Multiplexer supplies row and column addresses to the RAMs during read and write cycles. The address multiplexer is actually three 74F258 quad multiplexers. It receives the desired Control Memory address from the active requestor on the CADR <16:00> L signal lines. The outputs of the address multiplexer go to the RAMs themselves and supply the RAMs with the desired address. The signal SWAP CM BANK H, if asserted by the P.ioj, is also an input to the address multiplexer and is used to access the second bank of RAM in Control Memory.

### 11.3.2 Control Memory Eight Bit Refresh Address Counter

The Eight Bit Refresh Address Counter supplies refresh row addresses to the RAMs during refresh cycles. The counter is a 74LS590. The signal P01 CM REF CYC L enables the counter outputs. The counter outputs are connected to the address multiplexer output lines because the address multiplexer is not enabled during refresh cycles. The Refresh Address Counter is incremented at the end of each refresh cycle.

### 11.3.3 Control Memory Timing Logic and RAM Drivers

The timing logic circuitry generates required RAM timing and bus interface timing for Control Memory. The timing logic uses a 74F158 multiplexer to select taps of a delay line to provide timing. The multiplexer is steered by the signal SLOW MODE H. CTIMING H is the clock from the P.ioj that starts the timing circuitry for Control Memory. The signals CWRTH L and CWRTL L, from the active requestor, determine if the cycle is to be a read or a write. The signal CREFR GRANT L determines if the cycle is to be a refresh cycle.

Two 74F00s supply the RAMs with the row address strobe (CM RAS L), column address strobe (CM CAS L), and the two write enables (CM WRT HB L and CM WRT LB L). The write enables allow either byte or word writes.

### 11.3.4 Control Memory Data Receivers

The Control Memory Data Receivers, consisting of three 74F240 buffered line drivers, supply data to the RAMs on a write cycle. The receivers are always enabled. The data to be written, as well as the high and low parity bits, are present on the CDATA lines. The receiver output lines (CM <15:00,HP,LP> IN H) drive the Data-In lines (D IN) of the RAMs.

### 11.3.5 Control Memory Data Drivers

The Control Memory Data Drivers, consisting of three 74F240 buffered line drivers, supply data to the Control Bus out of RAM on a read cycle. They are enabled by the signal CM RD CYC L. When enabled, the data to be read is taken from the RAM outputs (CM <15:00,HP,LP> OUT H), through the drivers to their outputs, and out on the backplane to the active requestor.

One of the data drivers also drives the acknowledge (CACK L) signal, which indicates that the Control Memory recognizes and is responding to the address received from the active requestor.

## 11.4 CONTROL MEMORY CYCLES

During a Control Bus cycle, the Control Memory executes one of the following four cycles:

Idle
Write
Read
Refresh

The P.ioj decodes the CCYCLE lines on the Control Bus and determines if the cycle is a write or read. The P.ioj initiates the refresh cycle. Each cycle is covered in the following sections. Refer to the timing diagrams supplied for read, write, and refresh cycles.

### 11.4.1 Control Memory Idle Cycle

An Idle Cycle occurs when there is no active requestor or when a Non-Memory Access (NMA) cycle or an Interrupt (INTR) cycle occurs. The Control Memory remains idle during these types of cycles because the P.ioj does not generate CTIMING H. Therefore, Control Memory does not generate CM SEL L or the acknowledge (CACK L) for any idle, NMA, or INTR, Control Bus cycle.

### 11.4.2 Control Memory Write Cycle

In order to write to Control Memory, a requestor must first have been granted bus control by sending its request to the Control Bus arbitrator on the P.ioj module. The P.ioj acknowledges the request with the assertion of CGRANT L, and the requestor becomes active for that cycle.

At this point refer to Figure 11-3 for a timing diagram of the Control Memory write cycle.

With the assertion of CGRANT L, the active requestor asserts the address to be written on the CADR <16:00> L lines of the address multiplexer. CTIMING H is then asserted. The P.ioj asserts the write enables, CWRTH L or CWRTHL L, depending upon which byte(s) are to be written. The signal C CAS L is then asserted to strobe the data and parity into the RAM from the active requestor on the CDATA <15:00,HP,LP> L lines of the data receivers. The acknowledge signal CACK L is asserted to tell the active requestor the selected address exists in Control Memory.

When the Control Memory receives the assertion of CTIMING H from the P.ioj, it begins a write cycle to the address received on the CADR lines. The signal CTIMING H is asserted by the P.ioj and is gated through the Control Memory to become CM SEL L. The data on the CDATA lines plus parity is gated through the data receivers onto the D IN lines of the RAMs and written in the selected address.

### 11.4.3 Control Memory Read Cycle

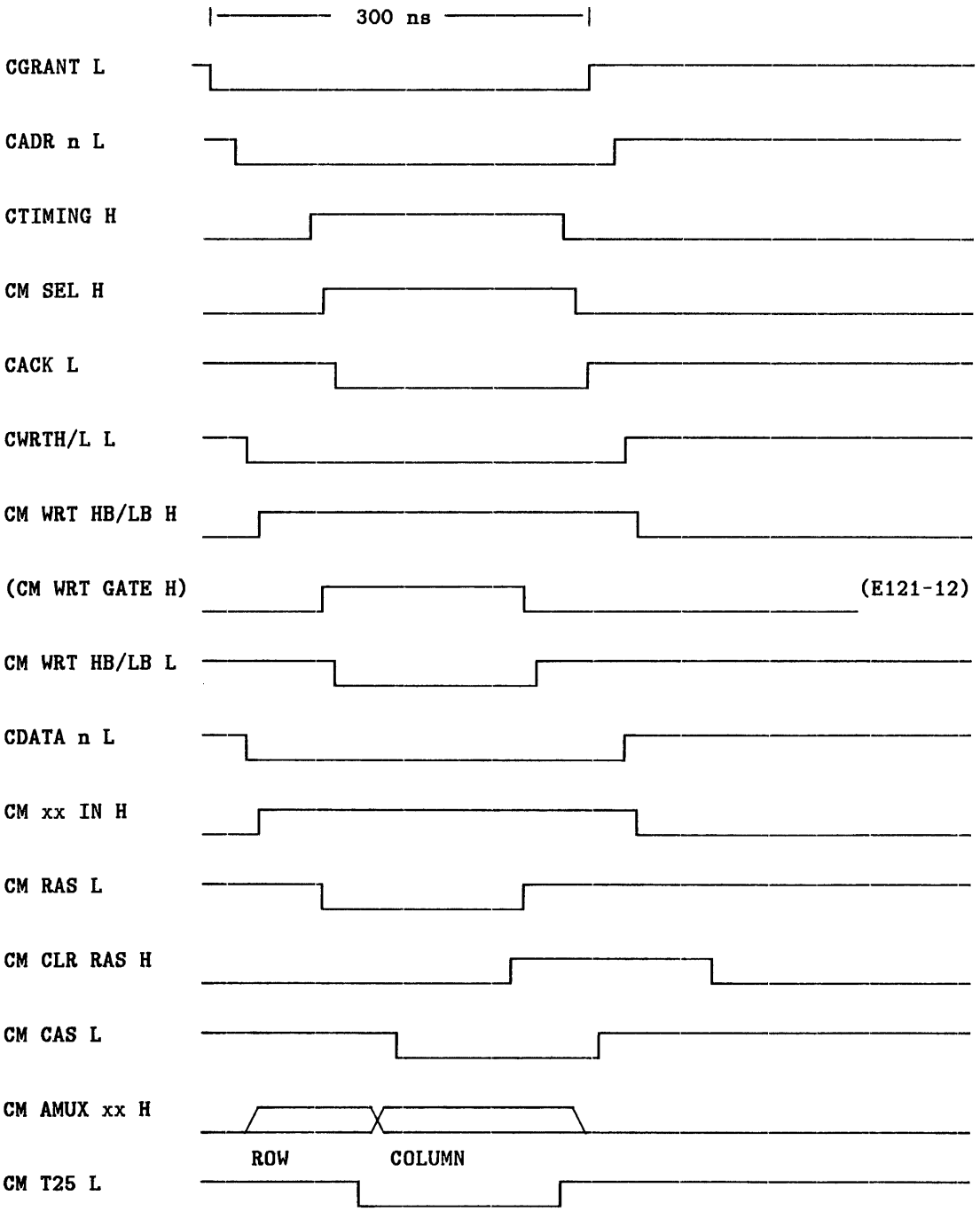
To read from Control Memory, a requestor must have been granted the Control Bus for that cycle by sending a request to the control bus arbitrator on the P.ioj and receiving back the assertion of the CGRANT L signal. The requestor is then active.

At this point refer to Figure 11-4 for a timing diagram of the Control Memory read cycle.

With the assertion of CGRANT L, the requestor asserts the desired address with the signals CADR <16:00> L. Both write enable lines, CWRTH L and CWRTL L, are negated. The Control Bus Arbitrator on the P.ioj decodes the cycle as normal and asserts the signal CTIMING H.

When Control Memory receives the assertion of the CTIMING H signal with the write enable lines negated, it starts a read cycle. The row address is selected with CM RAS L. The column address is valid, but not strobed onto the address lines until CM CAS L is asserted. CACK L is generated, telling the active requestor this is a valid Control Memory address. The data read from RAM is output onto the CDATA lines and sent to the active requestor. CM RAS L is negated and then CM CAS L and the data are negated. The CACK L signal is then negated to signal the end of the Control Memory read cycle.

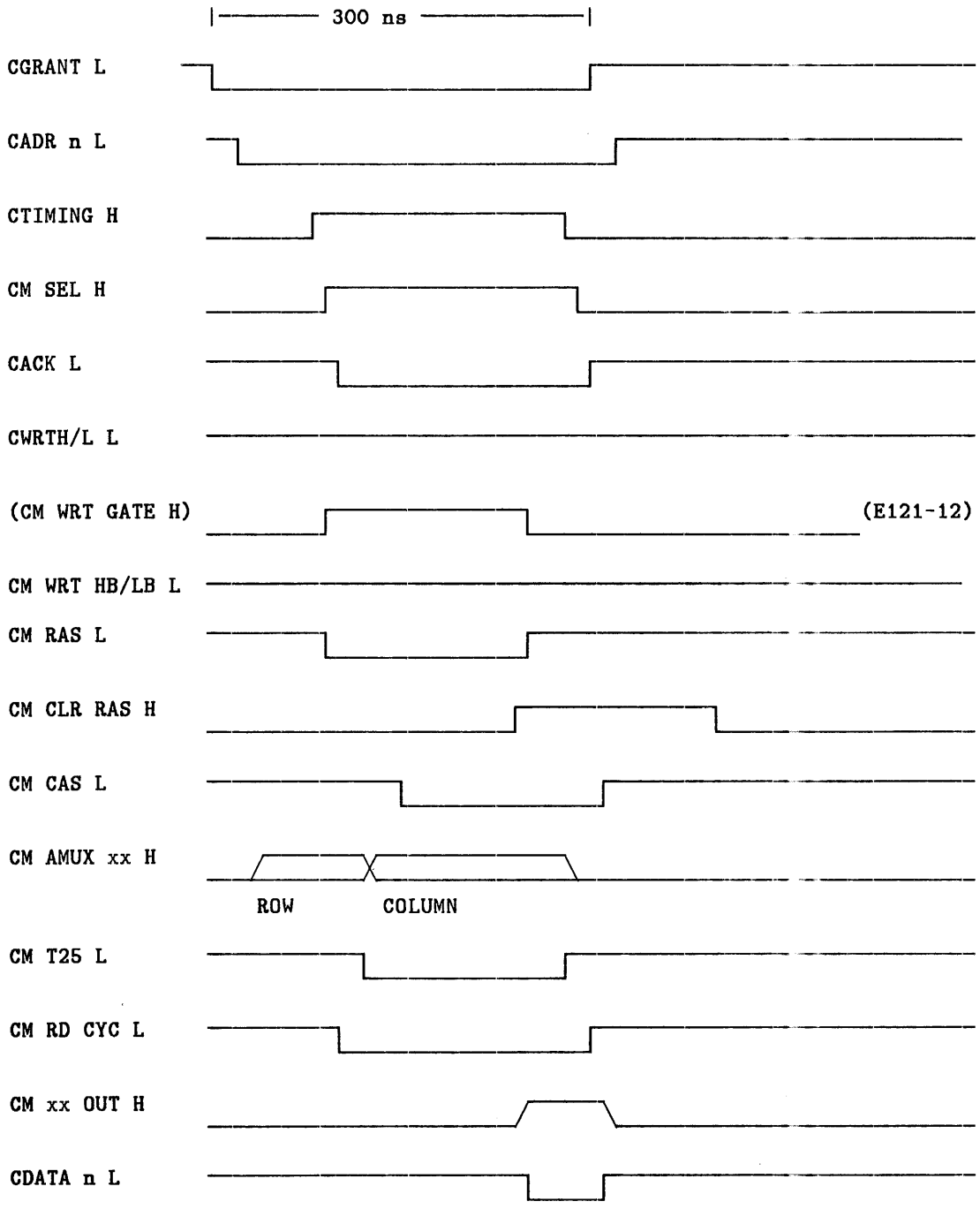
Figure 11-3 Control Memory Write Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any give time. The lines here are shown in the high state.

CXO-1225A

Figure 11-4 Control Memory Read Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any give time. The lines here are shown in the high state.

CXO-1226A

### 11.4.4 Control Memory Refresh Cycle

The dynamic RAMs used in Control Memory require that all rows within the RAM be refreshed every four milliseconds to ensure data retention. A refresh timer circuit on the P.ioj provides a signal called CREFRESH CLK H approximately every 15 microseconds to ensure that all 128 rows are refreshed within the required 2-millisecond period.

Refresh requests are given the highest priority in the arbitration logic on the P.ioj. No data is transferred across the Control Bus during a refresh cycle, nor does Control Memory generate CACK L or drive the CDATA lines.

The arbitrator asserts CREFR GRANT L, notifying the Control Memory that a refresh cycle is to be performed. The refresh cycle begins when CTIMING H is asserted. An eight-bit refresh address counter on the Control Memory supplies the address of the row to be refreshed. The counter is enabled by CREFR GRANT L and is incremented by the negation of the CREFR GRANT L signal.

Refer to Figure 11-5 for a timing diagram of the Control Memory refresh cycle.

## 11.5 DATA MEMORY

Data Memory resides on the Data Bus. Refer to Figure 11-6 and the print set.

Data Memory supports 256 Kwords of memory. It uses 64Kx1 static RAMs because of the high speed requirements of the Data Bus. It is organized as 4 banks of 18 RAMs each designated as banks zero, one, two, and three.

Each bank is 18 bits wide (16 data bits plus 2 parity bits) and 64 Kwords deep. Either data byte may be written separately using the signals DWRTH L and DWRTL L.

The following sections describe each of the components shown in the Data Memory block diagram.

### 11.5.1 Data Memory Address Receiver/Driver

The Data Memory Address Receiver/Drivers, consisting of four 74F240 buffered line drivers, supply addresses to the RAMs for read and write cycles. Two supply addresses to banks zero and one, and two supply addresses to banks two and three. They are always enabled. The address lines on the Data Bus are labeled DADR <15:00> L.

### 11.5.2 Data Memory Timing Logic and RAM Drivers

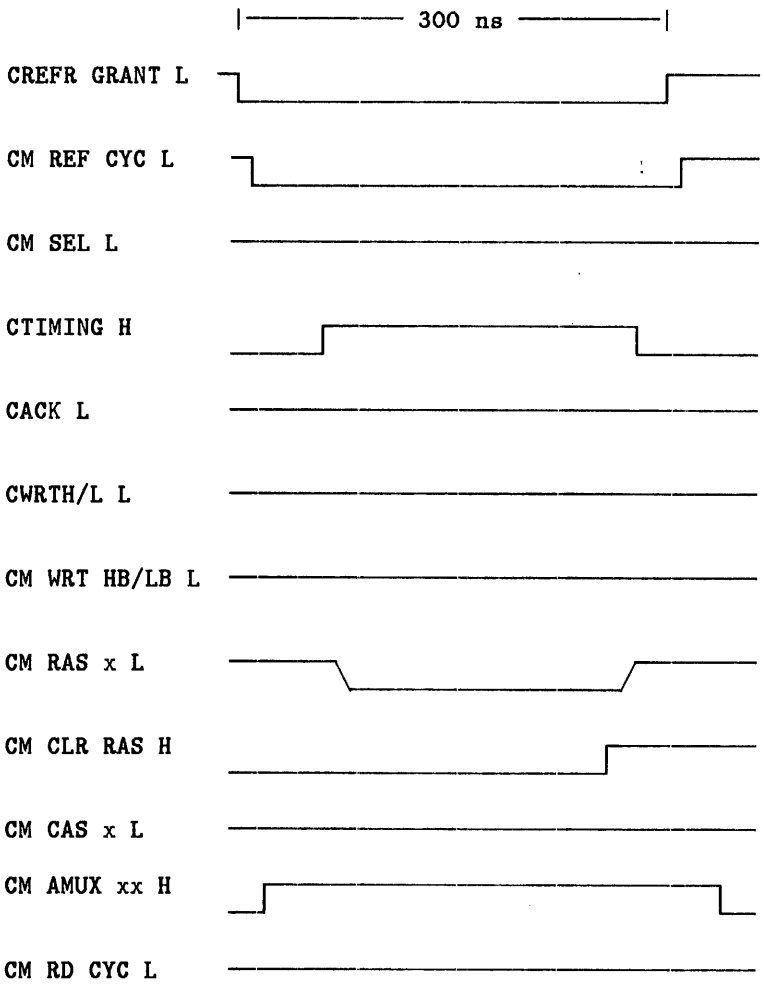
The Data Memory Timing Logic generates required RAM timing and bus interface timing for Data Memory. Data Memory timing depends on a 74F158 multiplexer, steered by the SLOW CYCLE L signal, to select taps of a delay line for discrete timing modes. The Data Bus cycles are synchronized by the requestor which has been granted control of the bus with the signal CLK B H. Both are 150 nanoseconds long.

A 74F00 is used to drive the write signals DWRTH L and DWRTL L to the RAMs for writing either or both bytes.

### 11.5.3 Data Memory Select Logic

The select logic compares the address on the DADR lines and generates DM SEL and DACK if the address is in the range of the installed memory. DNMA H disables the select logic because Data Memory is not selected for a non-memory access cycle. (The requestors use the non-memory access cycle for diagnostic purposes in checking the operation of the DADR and DDATA lines.)

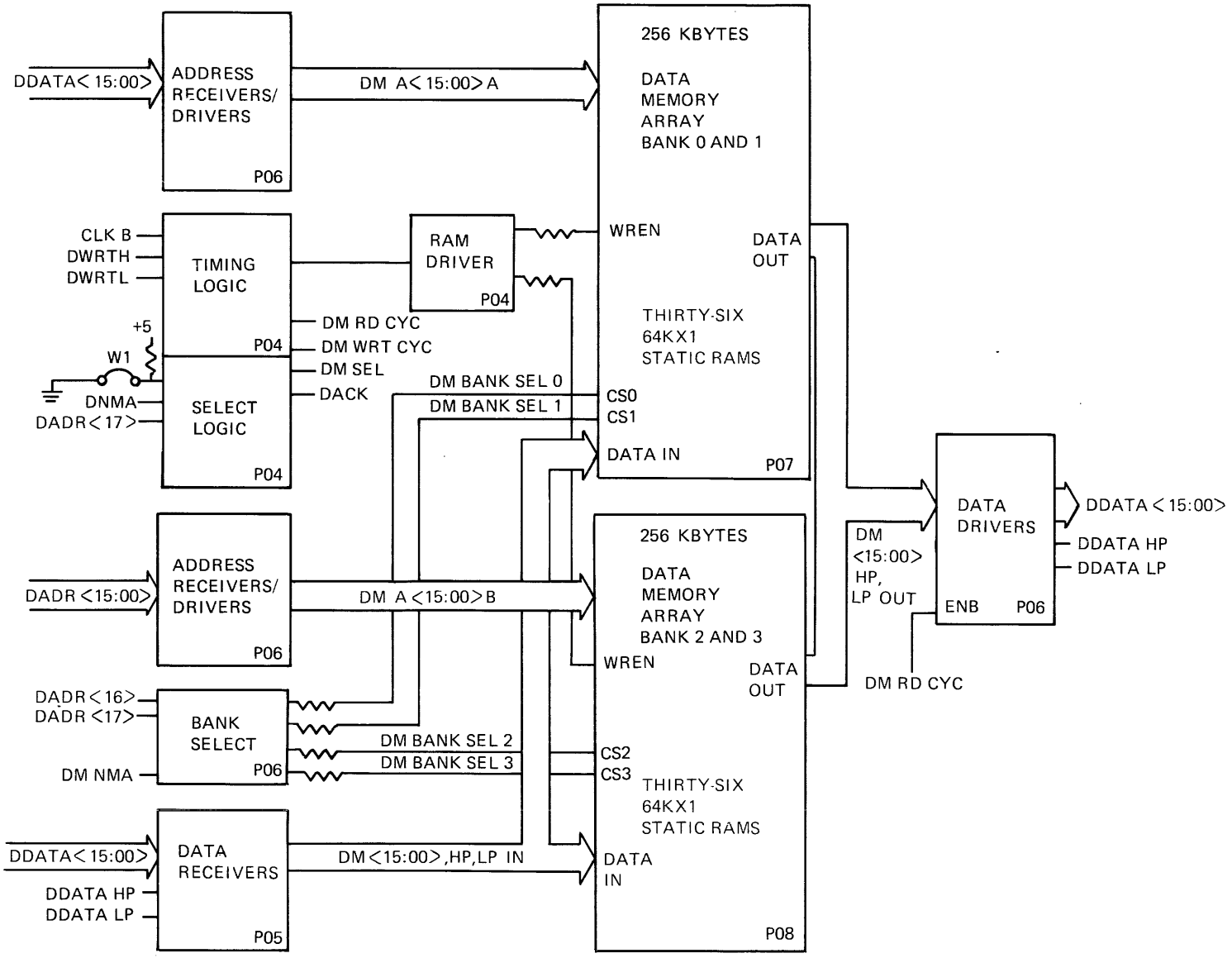
Figure 11-5 Control Memory Refresh Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any give time. The lines here are shown in the high state.

CXO-1227A

**Figure 11-6 Data Memory Block Diagram**




CX–962A

### 11.5.4 Data Memory Bank Select Logic

The Data Memory bank select logic selects one of four banks of RAM available to Data Memory. The bank select logic consists of one 74F139 one-of-four decoder. It is enabled by the negation of DNMA H (a non-memory access cycle is not in progress). Signals DADR 16 L and DADR 17 L select one of the four banks as shown in Table 11-1.

Table 11-1 Memory Bank Selection

| DADR 17 | DADR 16 | Bank Selected |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

### 11.5.5 Data Memory Data Receivers

The Data Memory data receivers, consisting of three 74F240 buffered line drivers, supply data to the RAMs on a write cycle. They are always enabled by the signal DM OE L. The data and parity bits to be written are driven onto the DDATA <15:00,HP,LP> L lines by the active requestor and passed through the driver to the RAM inputs DM <15:00,HP,LP> IN H.

### 11.5.6 Data Memory Data Drivers

The Data Memory data drivers supply data to the Data Bus out of RAM on a read cycle. The Data Memory data drivers, three 74F240 buffered line drivers, are enabled only on a read cycle by the signal DM RD CYC L. When the drivers are enabled, the data as well as the parity bits are driven onto the Data Bus lines, DDATA <15:00,HP,LP> L.

### 11.5.7 DATA MEMORY CYCLES

During a Data Bus cycle, the Data Memory executes one of the following three cycles:

Idle
Write
Read

The static RAMs used in Data Memory do not require refresh cycles. Each of the three cycles are covered in the following sections. Refer to the timing diagrams supplied for write and read cycles.

#### 11.5.7.1 Data Memory Idle Cycle

An idle cycle occurs when there is no active requestor or when an NMA (non-memory access) cycle occurs. During these cycles, the signal DNMA H forces the Data Memory select logic not to select Data Memory and forces the bank select logic not to select any bank of Data Memory.

#### 11.5.7.2 Data Memory Write Cycle

To write to Data Memory, a requestor must first acquire the Data Bus by asserting its request to the bus arbitrator on the P.ioj. Once the requestor receives the assertion of DGRANT L from the P.ioj, it has possession of the Data Bus for that cycle.

At this point refer to Figure 11-7 for a timing diagram of the Data Memory write cycle.

With the assertion of DGRANT L, the active requestor asserts the address to be written on the DADR <17:00> L inputs of the address receiver/drivers which supply the address to the RAM address input lines. The data and parity bits to be written are put on the data receiver input lines (DDATA <15:00,HP,LP> L) and the write enables (DWRTH L and DWRTL L) are asserted.

When the RAMs receive the address and one or both of the write enable lines are asserted, the cycle begins writing to the address received on the DADR lines.

Data Memory generates write enable pulses for one or both bytes as needed. CLK B H is used with a delay line to generate the DM WRT PULSE H signal.

#### 11.5.7.3 Data Memory Read Cycle

To read from Data Memory, a requestor must first become active by asserting its request to the P.ioj arbitration logic and receiving the asserted DGRANT L signal.

At this point refer to Figure 11-8 for a timing diagram of the Data Memory read cycle.

With the assertion of DGRANT L, the requestor asserts the address to be read on the DADR <17:00> L lines of the Data Memory Address Receiver/Drivers. The output of the receiver/drivers are to the RAM address input lines (DM A <15:00> H). The write enables, DWRTH L and DWRTL L, are negated as is the DNMA signal.

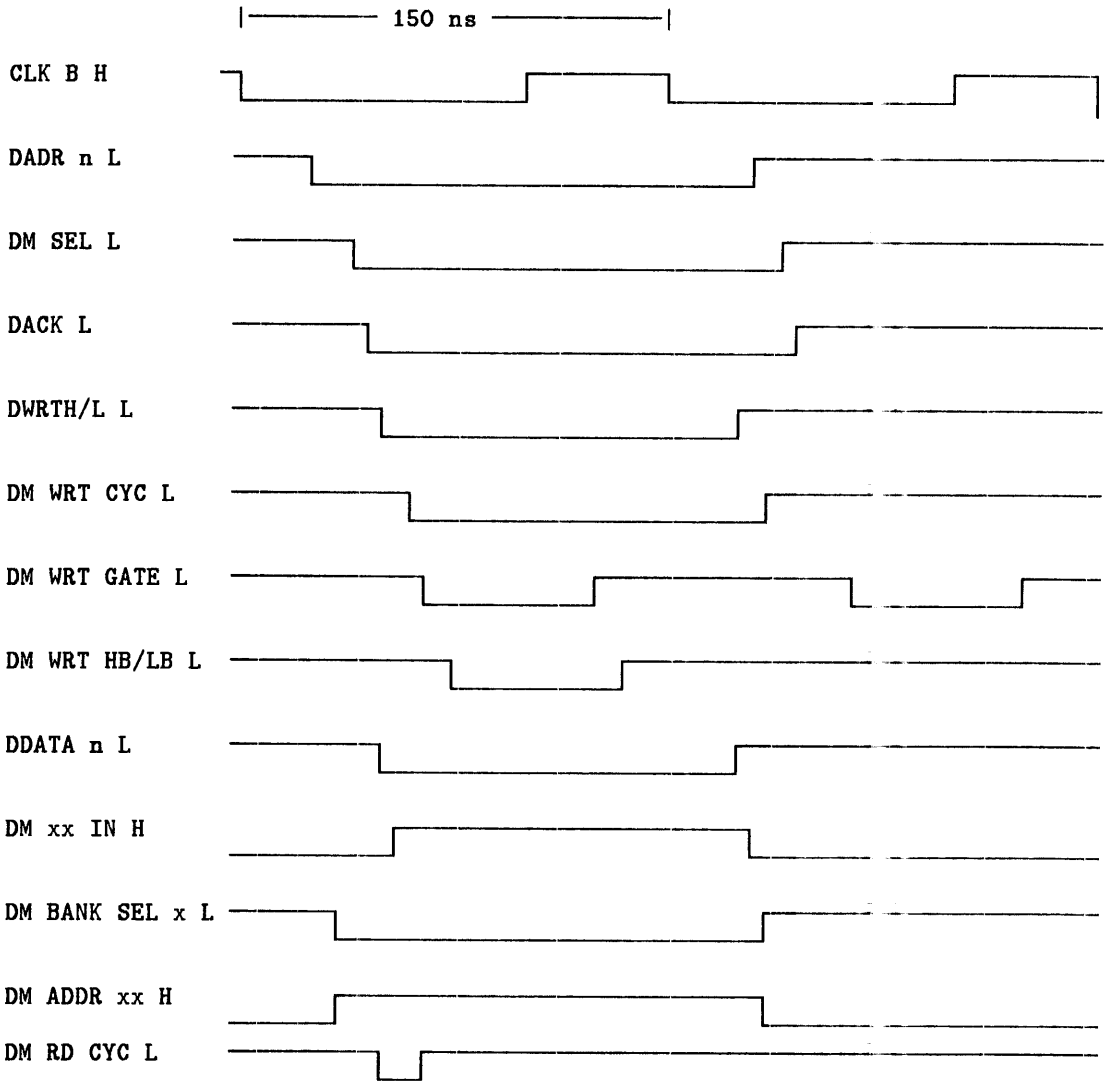
When Data Memory receives the address with the write enable lines negated, it begins the read cycle from the address on the DADR lines. The memory drives the data read, plus parity, onto the DDATA lines as long as both write enables and the DNMA signals remain negated and the address remains valid.

### 11.6 PROGRAM MEMORY

Program Memory, residing on the Program Bus, performs a read or a write cycle as directed by the P.ioj or the K.rx, whichever has been granted control of the Program Bus. Refer to Figure 11-9 and the print set. Program memory also performs refresh cycles every 15 microseconds as directed by the P.ioj signal PREFR CLK H.

Program Memory is logically organized as two banks, each 18 bits wide (16 data bits and 2 parity bits) and 256 Kwords deep. Either data byte may be written to separately with the signals BWTBT L and BDAL 00 L. The signals BDAL 19 (which becomes PM 19 IN L) and SWAP PM BANK are used to select one of the two banks as shown in Table 11-2.

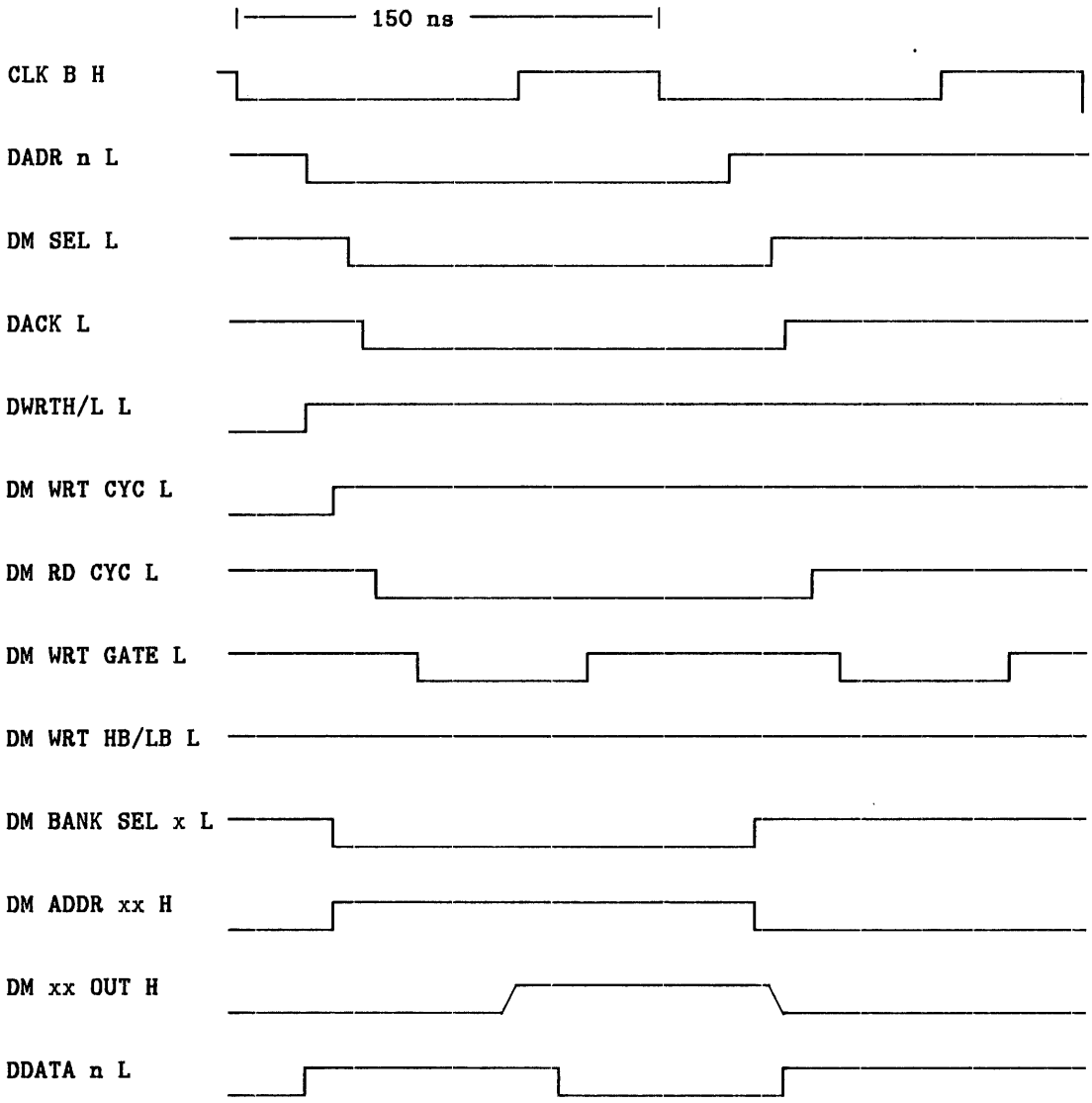
Figure 11-7 Data Memory Write Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any give time. The lines here are shown in the high state.

CXO-1228A

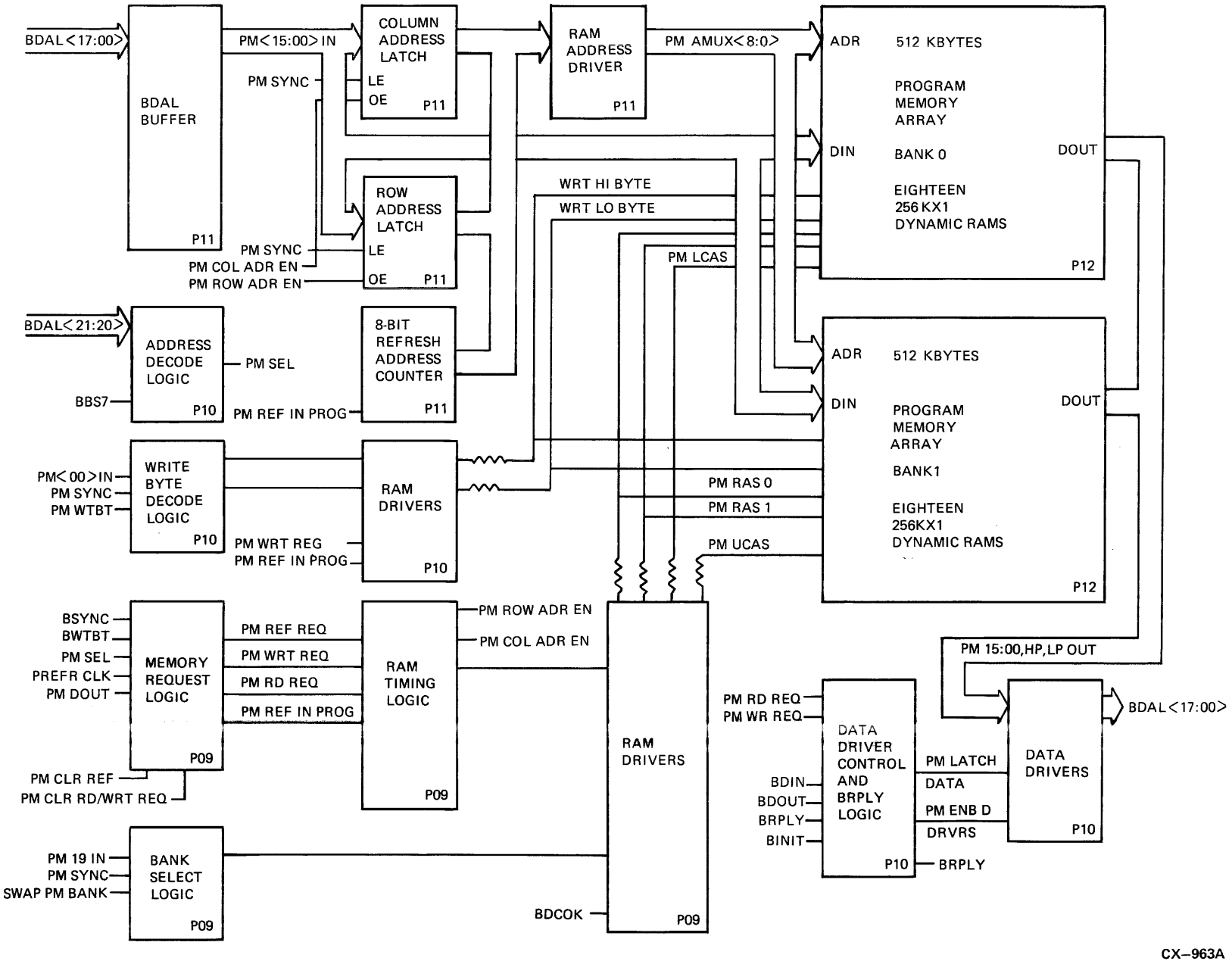
Figure 11-8 Data Memory Read Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any give time. The lines here are shown in the high state.

CXO-1229A

Figure 11-9 Program Memory Block Diagram




CX-963A

Table 11-2 Program Memory Bank Select

| BDAL 19 | SWAP PM BANK | Logical Bank Selected | Physical Bank Selected |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

The following sections describe each of the components shown in the Program Memory block diagram.

### 11.6.1 Program Memory BDAL Buffer

The BDAL buffer is three 74LS244 octal buffers. They receive an address from the P.ioj or K.rx, whichever is in control of the Program Bus. The buffers are always enabled by the signal PM OE L. The outputs of the BDAL buffers (PM <15:00> IN L) are applied to the row and column address latches. After the address has been latched into the row and column address latches, the BDAL lines receive data from the P.ioj or K.rx to be written to the RAMs.

### 11.6.2 Program Memory Row and Column Address Latches

The Program Memory Row and Column Address Latches supply row and column addresses to the RAM Address Driver. The row and column address selection is done with four 74S373 eight-bit latches. The signals PM <15:00> IN L are the inputs of the latches. The enable is the signal PM ROW ADR EN L for the row address latch and PM COL ADR EN L for the column address latch. The clock for both latches is PM SYNC L. The enables and clock for the row and column address selection come from the timing and Program Memory selection logic.

### 11.6.3 Program Memory RAM Address Driver

The RAM Address Driver supplies the addresses to RAM for read, write, and refresh cycles. The Program Memory RAM Address Driver is a series of 74S37 drivers. The outputs of the row and column address latches are applied to the RAM address inputs as signals called PM AMUX <8:0> H.

### 11.6.4 Program Memory Data Driver Control and Reply Logic

The Data Driver Control Logic controls the reading and writing of data through the Program Memory data drivers. The read request and write request signals, as well as the read and write control signals, are inputs to the logic. The outputs of the logic control the clock and enable of the data drivers.

The reply logic signal BRPLY L is dependent on a P.ioj or K.rx signal BDIN for a read cycle or BDOUT for a write cycle. This logic contains a flip flop for the read path and a flip flop for the write path.

### 11.6.5 Program Memory Data Drivers

The Program Memory Data Drivers supply data out of RAM to the Program Bus on a read cycle. The data driver logic consists of three 74S373 eight-bit latches. They are enabled by the signal PM ENB D DRVRS L and clocked by the signal PM LATCH DATA L. On a read cycle, data plus parity is taken from the RAM outputs, PM <15:00,HP,LP> OUT L, and clocked out on the BDAL <17:00> L lines to the P.ioj or K.rx.

### 11.6.6 Program Memory Eight Bit Refresh Address Counter

The Program Memory Refresh Address Counter supplies refresh row addresses to the RAMs during refresh cycles.

The refresh counter is a 74LS590 eight-bit counter. The outputs of the row and column address drivers are disabled for a refresh cycle. The signal PM REF IN PROG L enables the refresh counter output and causes the row and column address drivers to supply a refresh address rather than a memory address to the RAMs. This refresh address is incremented at the end of every refresh cycle.

### 11.6.7 Program Memory Address Decode Logic

Program Memory is selected for all address combinations that do not assert BBS7 L, BDAL20 L, or BDAL21 L.

### 11.6.8 Program Memory Write Byte Decode and RAM Driver Logic

The Write Byte Decoder (byte select logic) is a 74LS74 flip flop and two 74S00 NAND gates. During a word write, the flip flop is ignored and both bytes are written. During a byte write, the flip flop captures the state of address bit zero (bit zero selects which of the two bytes is written).

Logic consisting of a 74S37 NAND gate applies the outputs of the byte select flip flop plus the signal PM WRT ENB H to the selected RAMs for writing the high or low bytes.

### 11.6.9 Program Memory Request Logic

The Program Memory Request Logic consists of several flip flops servicing the types of requests Program Memory can respond to. The requests valid for Program Memory are:

Read
Write
Refresh

### 11.6.10 Program Memory RAM Timing Logic

The RAM Timing Logic consists of a 74S64 AND-OR-INVERT gate and delay lines. The types of requests available from the request logic are monitored by the gate, and the applicable timing is generated depending on the gate input that is true.

### 11.6.11 Program Memory Bank Select Logic and Program Memory RAM Driver

The Bank Select Logic selects one of the two banks of Program Memory available. The state of the signal PM 19 IN L determines which bank is selected. The bank select flip flop captures the state of PM 19 IN L and applies it to two NAND gates for the selection of the upper or lower bank. If the signal SWAP PM BANK H is asserted, the signal PM 19 IN L is inverted and the other bank of RAM is selected.

The Program Memory RAM Driver logic performs two functions. The output of the Bank Select Logic selects the lower or upper banks of Program Memory, and the output of the RAS flip flop strobes the desired row address. The row address strobe signals are PM RAS 0 L and PM RAS 1 L. The output signals for bank selection are PM LCAS L and PM UCAS L. Once the row has been selected, the bank select signals are supplied to select either the lower or the upper bank of RAMs.

### 11.6.12 PROGRAM MEMORY CYCLES

As stated earlier, Program Memory performs a write, read, or refresh cycle. The write and read cycles are initiated by either the P.ioj or the K.rx. The refresh cycle is initiated by the P.ioj. Refer to the timing diagrams supplied for write and read cycles.

#### 11.6.12.1 Program Memory Write Cycle

Refer to Figure 11-10 for a timing diagram of the Program Memory write cycle.

To initiate a write cycle, the bus master for that cycle (P.ioj or K.rx) asserts an address to be written to on the BDAL lines while asserting BWTBT L. The bus master then asserts BSYNC L. The address on the BDAL lines is latched into the address latches. The bus master then asserts the data and parity bits to be written on the BDAL lines. The signal BDOUT L is asserted and the Program Memory write cycle begins. The signal BRPLY L is asserted at this time.

If a refresh is not in progress, the write cycle begins as soon as BDOUT L is received. If a refresh cycle is in progress, the write cycle is delayed as much as 330 nanoseconds, one internal memory cycle, until the refresh is completed. Byte writes are accomplished by asserting BWTBT L before issuing BDOUT L. The BDAL 00 L signal determines which byte is to be written.

When the write cycle is finished, BDOUT L is negated. Program memory then negates BRPLY L.

#### 11.6.12.2 Program Memory Read Cycle

Refer to Figure 11-11 for a timing diagram of the program memory read cycle.

To initiate a read cycle, the bus master asserts an address on the BDAL lines to read from. BWTBT L is negated to indicate a read cycle and BSYNC L is asserted. This starts the internal timing sequence for a read cycle.

If a refresh is not in progress, the data is read from RAM and latched in the data drivers. The data is held in the data driver latch until receiving BDIN L, at which time the data is driven out onto the BDAL lines. The reply signal, BRPLY L, is then asserted.

If a refresh was in progress, the read cycle is delayed as much as 330 nanoseconds, or one complete memory cycle, until the refresh cycle is completed.

When the data has been gated onto the BDAL lines, BDIN L is negated and BRPLY L is negated. The data is removed from the BDAL lines.

#### 11.6.12.3 Program Memory Refresh Cycle

Program Memory RAMs require refresh cycles to ensure data retention in the dynamic RAMs. A refresh circuit on the P.ioj supplies refresh requests to the Program Memory every 15 microseconds. If a refresh request is supplied while a read or write cycle is in process, the request is queued until the present cycle is finished. Similarly, if a read or write cycle is received while a refresh is in process, the read or write request is queued until the refresh cycle is complete.

An Eight Bit Refresh Address Counter in the Program Memory provides the row address to be refreshed. This address is incremented at the end of every refresh cycle. The column address strobe (CAS) is not needed for refresh.

Refer to Figure 11-12 for a timing diagram of the Program Memory refresh cycle.

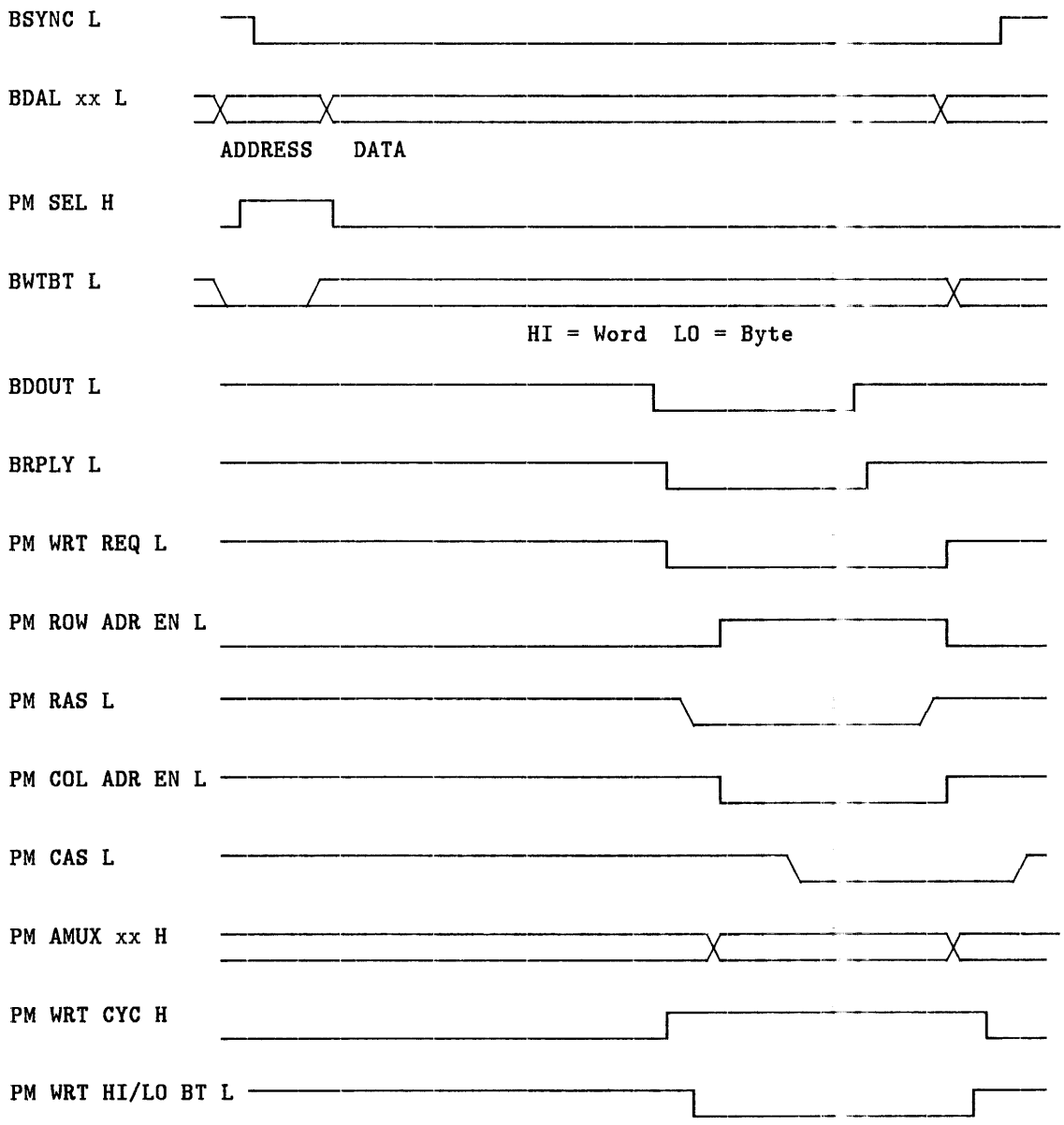
### 11.7 FLOPPY DISK CONTROLLER

The Floppy Disk Controller, or K.rx, residing on the Program Bus, performs direct memory access (DMA) word transfers when reading or writing. The minimum data transfer size in one command is 512 bytes (one sector). The maximum data transfer size in one command is one track (512 bytes x 15 sectors). At the end of a data transfer an interrupt is generated.

The main controlling chip for the K.rx is the 2797. It is also known as the Floppy Disk Controller (FDC) and will be referred to as the FDC in this document. It controls the software-to-hardware and hardware-to-software conversions between the RX33 floppy disk drives and the K.rx.

Pre-programmed PALs are used to control the steering and timing of the K.rx internal data path.

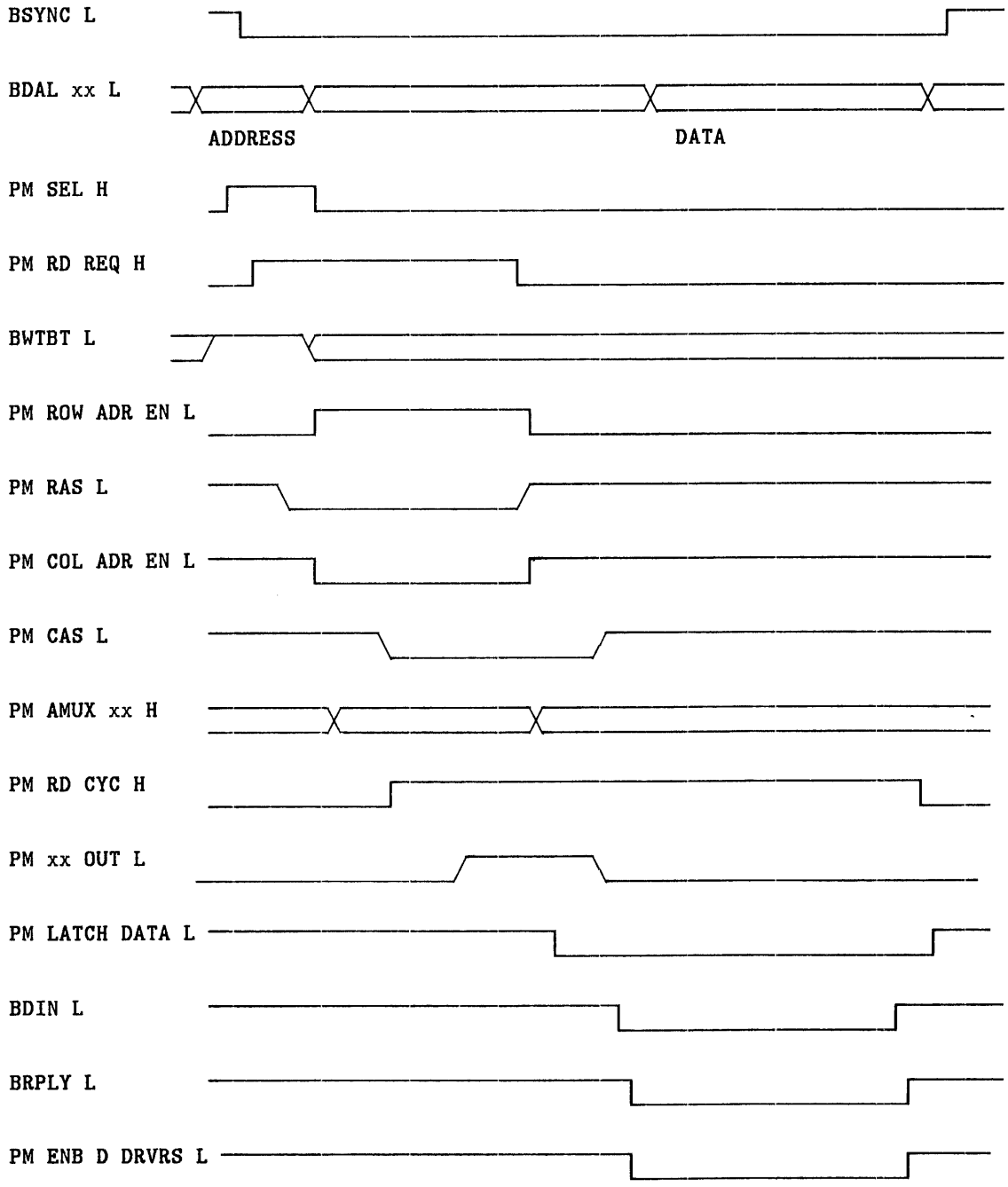
Figure 11-10 Program Memory Write Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any give time. The lines here are shown in the high state.

CXO-1230A

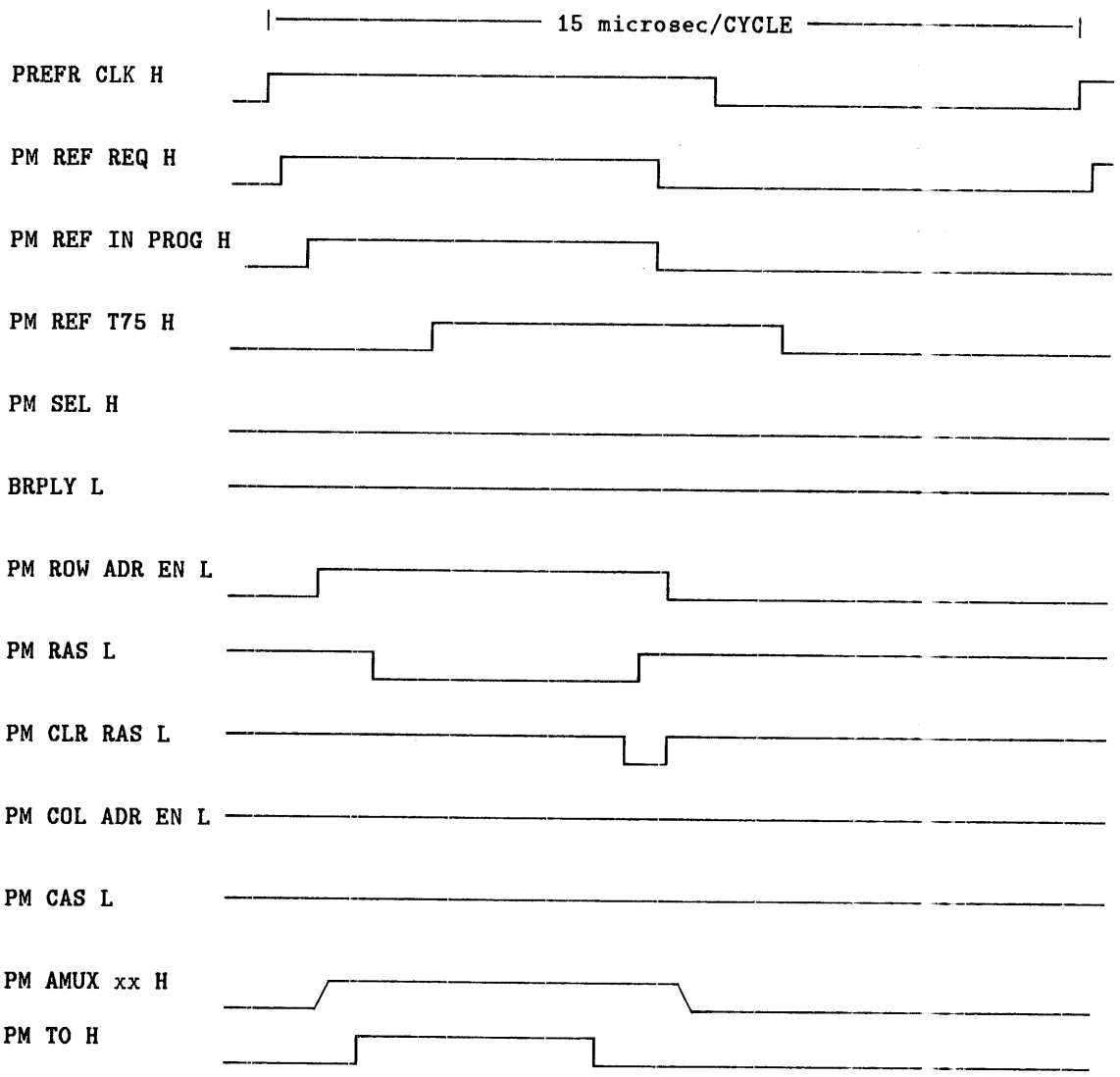
Figure 11-11 Program Memory Read Cycle Timing




NOTE: The address or data lines can be in the high state or low state at any give time. The lines here are shown in the high state.

CXO-1231A

Figure 11-12 Program Memory Refresh Timing




NOTE: The address or data lines can be in the high state or low state at any give time. The lines here are shown in the high state.

CXO-1232A

The starting memory word address for any data transfer is loaded into a 22-bit Memory Address Register (MAR). It is a physical memory address as opposed to a virtual memory address. The MAR will auto-increment by two at the end of each word transfer.

Byte parity is checked or generated for all data transfers to and from memory. Odd parity is used.

A non-existent memory error (NXM) will be detected if a memory reply is not received in response to each word transferred.

Lost data errors will be detected if the data transfer rate of the K.rx is not maintained by the Program Bus during a data transfer.

### 11.7.1 Floppy Controller Registers

The K.rx uses four 16-bit registers to control the floppy disk drives. The low byte of each register is implemented within the FDC chip while the high byte of each register is implemented external to the FDC chip. All four registers are word addressable only. These registers are:

- Command and Status Register, High Byte (CSR HIGH BYTE)
- Command and Status Register, Low Byte (CSR LOW BYTE *)
- Current Track Register (TRK REG *)
- Sector Register (SEC REG *)
- Data Register, Track Address Register (DAT/TRK REG *)
- Memory Address Register, Low Byte (MAR 0)
- Memory Address Register, Middle Byte (MAR 1)
- Memory Address Register, High Byte (MAR 2)

**NOTE**

**The asterisk (*) denotes the registers implemented in the FDC chip.**

The registers and their bit assignments and addresses are shown in Figure 11-13. Notice the division between bytes designating actual registers.

A description of each register follows.

#### 11.7.1.1 Command and Status Register (CSR)

The CSR register is the most frequently used. The low byte of the CSR is part of the FDC chip. The high byte is implemented with an AM2952 eight-bit bi-directional register.

The Command and Status Register high byte is described in Table 11-3.

Figure 11-13 Floppy Register Bit Assignments

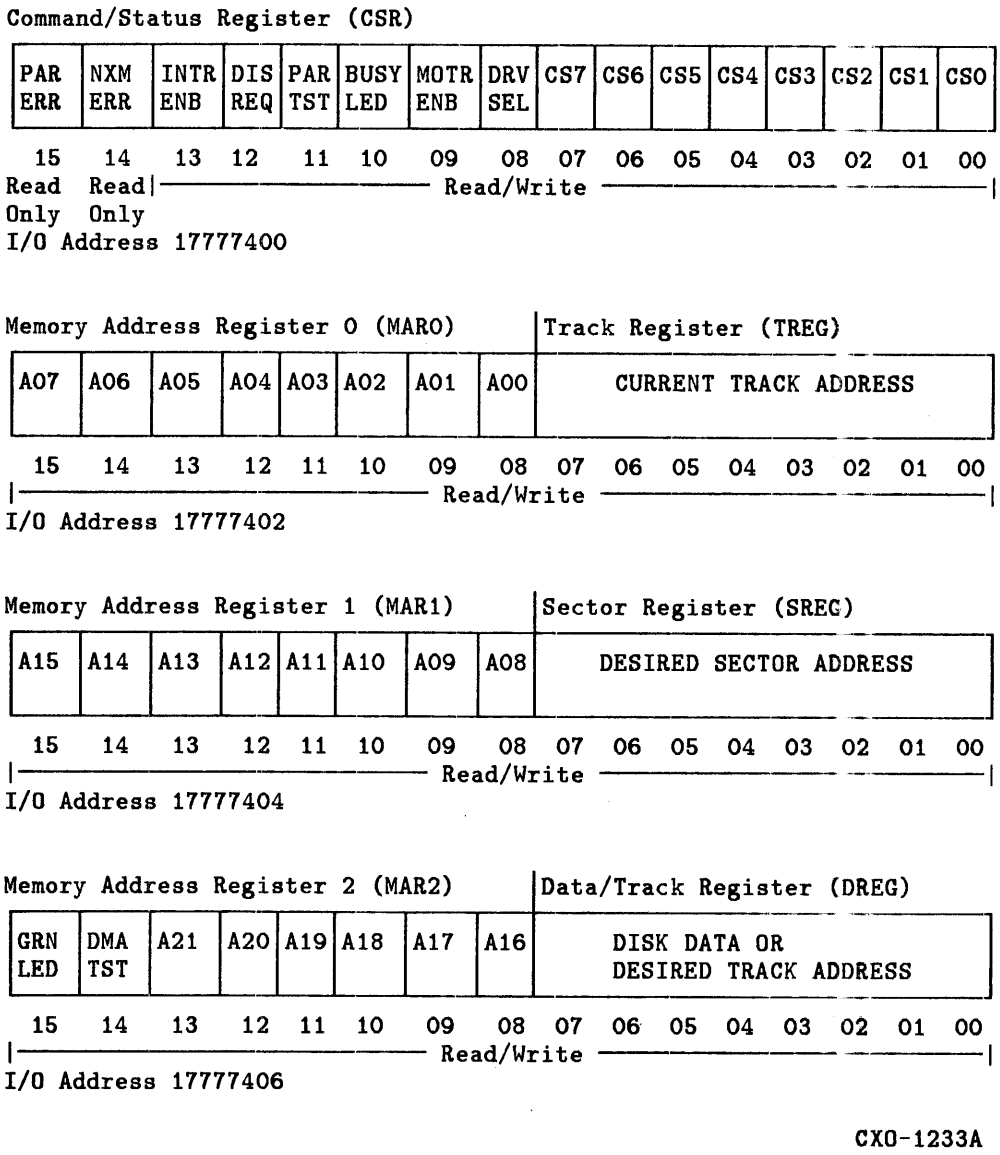


Table 11-3 Command and Status Register Bit Assignments (High Byte)

| Bit | Name | Read/Write | Description |
|---|---|---|---|
| 15 | Parity Error | Read Only | Sets when bad parity is received in one or both bytes during a DMA read from memory. Clears on power-up, when a new command is loaded into the CSR, when B INIT L is asserted or when the motor enable bit, bit 9, is cleared. |
| 14 | NXM Error | Read Only | Sets when a reply is not received from memory within ten microseconds of the assertion of BSYNC L by K.rx during a DMA read or write. Clears by power-up, when a new command is loaded into the CSR, when B INIT L is asserted or when the motor enable bit is cleared. |
| 13 | Interrupt Enable | Read/Write | Prevents unwanted interrupts from K.rx. Must be set before the interrupt condition occurs or no interrupt will be generated. |
| 12 | Disable DMA Request | Read/Write | Used by diagnostics to inhibit DMA requests. Not set in normal application. |
| 11 | Parity Test | Read/Write | Used by diagnostics to force bad parity for both bytes when doing DMA writes to memory. Also detects parity when doing DMA reads from locations that have good parity. |
| 10 | Drive Busy LED | Read/Write | Controls the LED on front of floppy disk drive. |
| 09 | Motor Enable | Read/Write | When clear, keeps K.rx in initialized state by generating an internal reset similar to the assertion of B INIT L from the P.ioc. When set, both drives are spun up and a seek to track zero is performed. |
| 08 | Drive Select 0/1 | Read/Write | Selects one of the two floppy disk drives. Clear selects drive zero, set selects drive one. |

The CSR Registers low byte, CSR BITS <7:0>, when written to, represent the Command Register. When read back they represent the Status Register. The Status Register bits are dynamically defined; they change context depending on the type of command last issued. Commands will be discussed in more detail in Section 11.7.4.

#### 11.7.1.2 Memory Address Register 0/Track Register (MAR0/TREG)

MAR0 and the Track Register are byte-wide registers paired as a single, word-wide register, jointly accessible as a word. MAR0 is the high byte of the word.

#### 11.7.1.2.1 MAR0 <15:8>, Read/Write

MAR0 is loaded with the lowest eight bits of the desired memory transfer address, A<07:00>, prior to issuing a DMA read or write command. It auto-increments by two at the end of each DMA word transfer. The carry out of MAR0 is the carry in to MAR1.

This register is cleared on power up or when BINIT L is received.

#### 11.7.1.2.2 Track Register <07:00>, Read/Write

The Track Register is read to determine the head location on a selected drive. It increments by one every time the head is stepped in (away from track zero) and decrements by one when the head is stepped out (toward track zero).

This register is not directly initialized by power up, BINIT L, or clearing the motor enable bit, but the seek resulting from the power up or BINIT L loads the track register with zeros.

### 11.7.1.3 Memory Address Register 1/Sector Register (MAR1/SREG)

MAR1 and the Sector Register are byte-wide registers paired as a single, word-wide register, jointly accessible as a word. MAR1 is the high byte of the word.

#### 11.7.1.3.1 MAR1 <15:08>, Read/Write

MAR1 is loaded with the middle eight bits of the desired memory transfer address, A <15:08>, prior to issuing a DMA word transfer. It auto-increments by one at the end of a DMA word transfer if the carry-out bit of MAR0 is asserted. The carry out of MAR1 is the carry in of MAR2.

This register is cleared on power up or when BINIT L is received.

#### 11.7.1.3.2 Sector Register <07:00>, Read/Write

This register is loaded with the desired sector position. The register contents are compared with the recorded sector number in the ID field of the media during floppy read or write operations. The Sector Register will auto-increment during multi-sector transfers until it contains a sector number that is one greater than the number of sectors per track (for example: $20_8$) causing a record-not-found error.

This register is initialized to $001_8$ on a power up, when BINIT L is asserted, or when the Motor Enable bit goes from clear to set.

### 11.7.1.4 Memory Address Register 2/Data Register (MAR2/DREG)

MAR2 and the Data Register are byte-wide registers paired as a single word-wide register and are only accessible jointly as a word. MAR2 is the high byte of the word.

#### 11.7.1.4.1 MAR2 <21:16>

MAR2 is loaded with the highest six bits of the desired memory transfer address, A <21:16>, prior to issuing a DMA read or write command. It auto-increments by one at the end of a DMA word transfer if the carry-out bit of MAR1 is asserted.

#### 11.7.1.4.2 MAR2, Bit 15, Module OK LED

Bit 15 is the module OK bit. It is set to light the green LED (module OK) on the M.std2 module. Bit 15 is cleared to light the red LED (module BAD). It is not an address bit.

#### 11.7.1.4.3 MAR2, Bit 14, Enable DMA Test

Bit 14 is the Enable DMA Test bit. It is set to test DMA transfers without having a floppy disk drive connected to the FDC. When set, the FDC does not drive the floppy interface signals. This guarantees that a DMA test doing a write sector operation cannot inadvertently write on the floppy media. Although the motor enable bit may be set, DMA Test will spin down the motors. Bit 14 is not an address bit.

All bits of the MAR2 register are cleared on power up or when BINIT L is asserted.

### 11.7.2 Data Register

The Data Register is loaded with the address of the desired track position on a seek command. The contents of this register are compared with the current track number in the Track Register to determine in which direction and how far to step the head.

During the read or write operations, the Data Register is used as a holding register and serves as the port for data byte loading and unloading to the Memory Data Register (MDR). The MDR, sometimes known as the XBUF/RBUF, is discussed in Section 11.7.3.4.

### 11.7.3 K.rx Floppy Controller Block Diagram

The following paragraphs describe the K.rx components. Refer to Figure 11-14 and the print set.

#### 11.7.3.1 Program Bus Register Selector, Synchronizer, Register Control PAL

The Program Bus Register Selector is a DC004 which receives the BDAL <02:00> L lines and the bus control signals, BSYNC L, BWRTBT L, BDIN L, and BDOUT L. These signals are used for reading and writing the K.rx registers. The Register Selector is enabled by the signal RX MATCH H. The outputs of the DC004 are the inputs to a bank of D flip flops contained in a 74F374 labeled the Synchronizer on the block diagram. The outputs of the Synchronizer are fed into two 16R8 PALs, the Register Control PAL (REGCTL), and the RX Control PAL (RX CTL). The Synchronizer, the Register Control PAL, and the RX Control PAL are clocked by the signal PAL CLK H and are always enabled by the signal TS OE L. The Register Control PAL takes the REG SEL x L signals from the Synchronizer and outputs the signals needed to read and write the Floppy Disk Controller Memory Address Registers (MARs). The RX Control PAL uses the same signals from the Synchronizer for reading and writing the Control and Status Register (CSR.)

#### 11.7.3.2 Program Bus Transceivers and Vector Control, High Address Driver

The Program Bus transceivers consist of four DC005 bi-directional chips. When receiving, the transceivers receive the BDAL <15:00> L signals from the Program Bus and turn them into RX D<15:00> H signals that the K.rx can use as data lines. When transmitting the RX D<15:00> H signals are driven onto the Program Bus BDAL lines. The DC005s are labeled BUS INTERFACE. The high address driver (74LS240) takes the RX D<21:16> H signals plus a write command and converts them back to BDAL signals and a BWTBT L signal for transmission on the Program Bus.

#### 11.7.3.3 Parity Generator/Checker

The K.rx parity generator/checker consists of two 74LS280 parity generators. They take the RX D<15:00> H signals from the Program Bus transceivers and generate the odd parity bits BDAL 17 L and BDAL 16 L. The checking is done by routing the parity bits, now called RDAL<17:16>, back to the 74LS280s through exclusive OR gates. The signal PAR TST H, a bit from the Control and Status Register, is used during diagnostics to force reading or writing with bad parity. This is a check for the 74LS280s.

#### 11.7.3.4 Interrupt Control and Vector Timing

Interrupt control and vector timing is done with a DC003 labeled on the print set as UNIBUS INTERRUPT. The signal INTR REQ H from the FDC chip initiates an interrupt if INTR ENB H is set in the CSR. The P.ioj responds with BIAKI L (interrupt acknowledge) and BDIN L. The K.rx sends the interrupt vector of 230 (octal) to the P.ioj along with BRPLY L. Interrupts are generated at the end of a command if the INTER ENB H bit is set in the CSR. The interrupt may indicate successful completion or an error. The CSR must be examined to determine the results.

#### 11.7.3.5 DMA Control

DMA control is done with a DC010 and is labeled DIRECT MEM ACCESS on the print set. The signal DMA REQ H, originally from the Disk Read and Disk Write PALs, initiates the DMA cycle by asserting BDMR L. After asserting BDMGI L, the signals MASTER H, TSYNC H, TDIN H, TDOUT H, ADR EN H, and DATA EN H are generated appropriately. These signals are applied to a bus driver (74LS240) to generate the bus control signals, BSYNC L, BDIN L, BDOUT L, and BSACK L.

#### 11.7.3.6 Disk Read PAL, Disk Write PAL

The disk read and disk write logic is contained in two separate 16R8 PALs labeled respectively DISKRD and DISKWR on the print set. They are clocked by the PAL CLK H signal and always enabled by TS OE L. The Disk Read PAL receives key timing events applicable to reads and outputs read data path control signals. The Disk Write PAL receives key timing events applicable to writes and outputs write data path control signals.

#### 11.7.3.7 Memory Address Register (MAR)

The three memory address registers are 20X8 PALs. Each is individually loaded by the register control PAL with the signals LD MAR x L. They are clocked by PAL CLK H and always enabled by TS OE L. The inputs are the RX data lines RX D<15:8> H. These inputs are used to load the starting memory address for the data transfer about to begin. After a word is transferred, the MAR is auto-incremented by two.

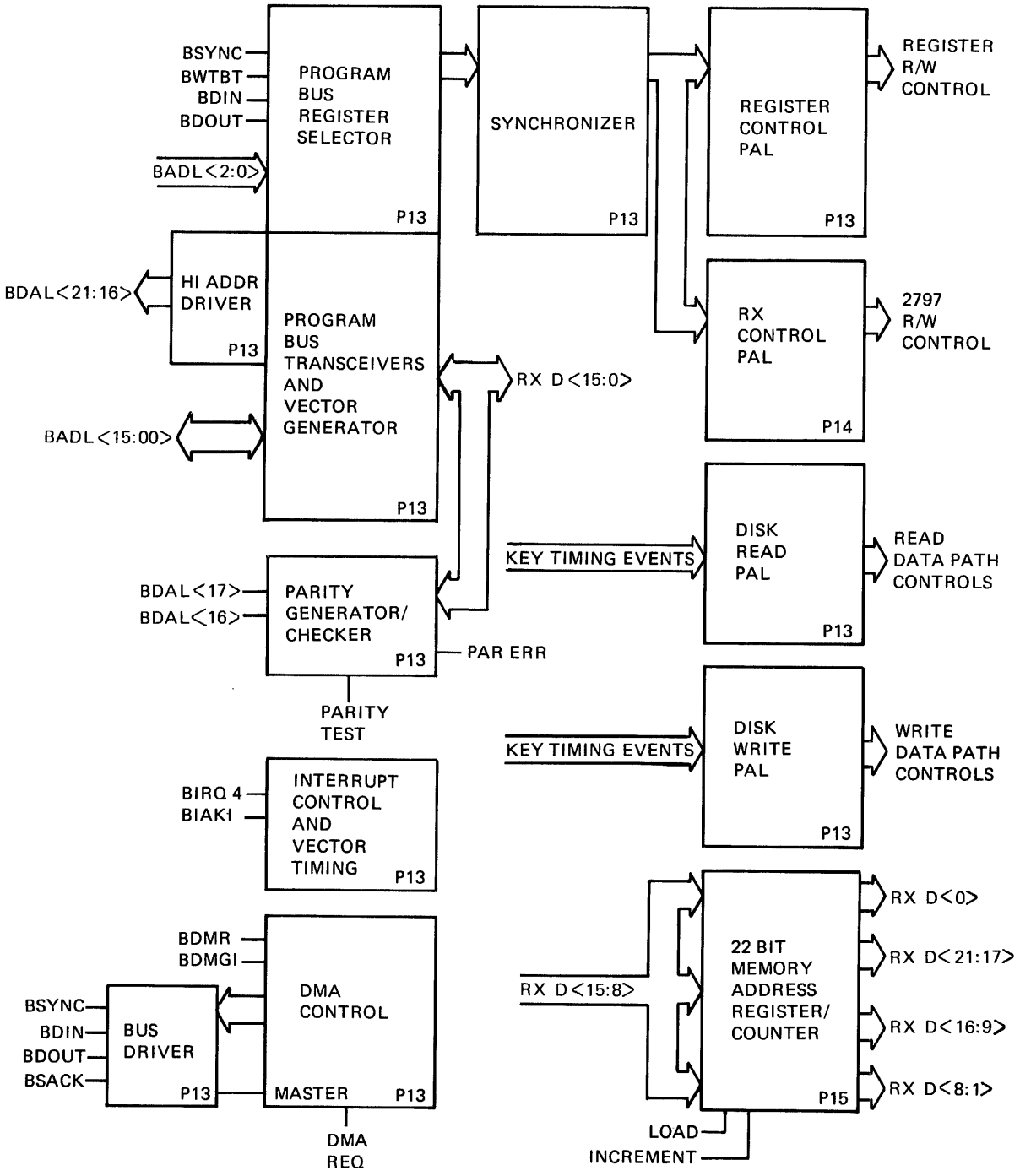
#### 11.7.3.8 Non-existent Memory Detection

The non-existent memory (NXM) detection logic consists of a 74LS123 one-shot and a 74LS74 flip-flop. The flip-flop detects the timeout of the one-shot. A timeout occurs when K.rx transfers a word to or from memory and a memory reply is not received within 10 microseconds. When a timeout is detected, the signal NXM ERR L is generated.

#### 11.7.3.9 Floppy Disk Controller 2797 (FDC)

The 2797 Floppy Disk Controller is the main controlling chip for the K.rx. The 2797 is an LSI chip designed to be a one chip floppy interface controller. This chip provides the hardware/software interface by converting software commands loaded into the data/track register, sector register, track register, and CSR into signals which are sent to the floppy drive to cause it to seek, read, or write. Hardware signals from the floppy drive are converted into data and status which are read by software. The RX D<7:0> H signals provide the software/hardware interface signals.

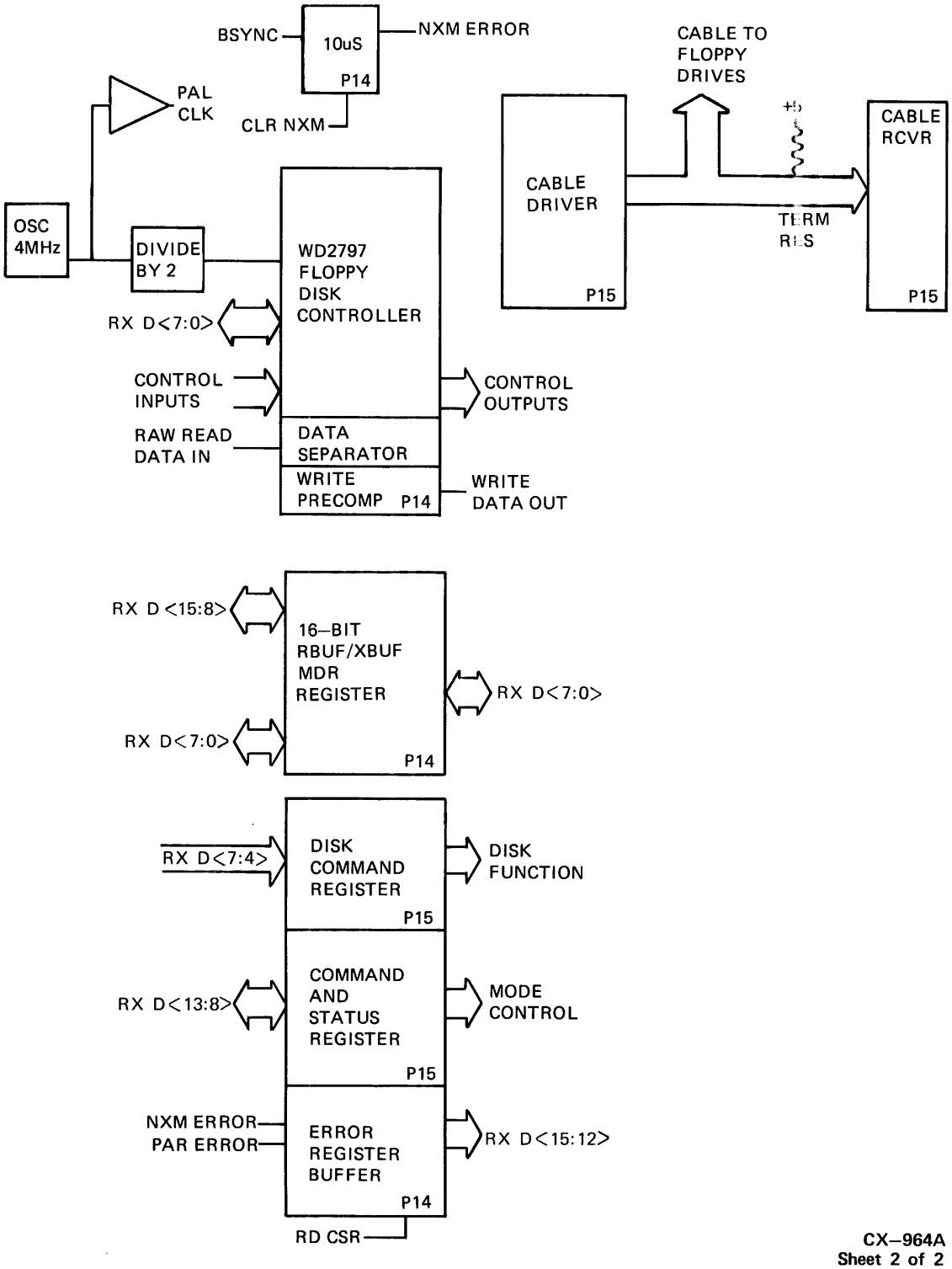
Figure 11-14 K.rx Controller Block Diagram




CX-964A
Sheet 1 of 2

(Continued on next page)

Figure 11-14 (Cont.) K.rx Controller Block Diagram




CX–964A
Sheet 2 of 2

The FDC also generates and checks cyclic redundancy checks (CRC) for all data transfers. An on-chip data separator receives encoded read data and generates separate data and clock on a read cycle. A write precompensation circuit compensates for predictable magnetic shifting effects when writing to the floppy drive.

A 4 MHz crystal oscillator with a divide-by-two flip-flop provides the timing for the FDC.

**NOTE**
**The 4 MHz oscillator also provides the PAL CLK.**

The Floppy Disk Controller has two potentiometers and a switch that are set at the factory and are not to be adjusted in the field. The potentiometer connected to WPW (pin 33) adjusts the write precompensation. The potentiometer connected to RPW (pin 18) adjusts the data separator. The switch connected to VCO (pin 26) adjusts the data separator.

### 11.7.3.10 RBUF/XBUF Register (MDR)

The 16-bit RBUF/XBUF register consists of two 8-bit 2952 registers. It is also known as the Memory Data Register or MDR. It is used to latch receive or transmit data onto the K.rx data lines (RX D<15:00> H).

### 11.7.3.11 Disk Command Register

The Disk Command Register is a 16R8 PAL. It is labeled in the print set as the FUNDEC, or function decoder. The Disk Command Register takes as inputs the RX D<7:4> H signals and provides disk functions as outputs. These disk functions are:

DISK R/W L
LD CSR DLY L
DISK READ L
DISK WRITE L

The PAL is clocked with PAL CLK H and is always enabled with TS OE L.

### 11.7.3.12 Command and Status Register (CSR)

The command and status register is an 8-bit 2952 read/write register. It is not labeled in the print set but can be identified by the enable inputs, LD CSR L and RD CSR L. The CSR is clocked by PAL CLK H. It takes as inputs the RX D <13:8> H lines, and outputs the first six bits of the high byte of the CSR. On a read, the high byte of the CSR is read and output on the RX D <13:8> H lines.

### 11.7.3.13 Error Register Buffer

The error register buffer is a 74LS240 eight-line driver. Its output control is enabled by RD CSR L. It passes the NXM ERR L and PAR ERR L signals and loads them into the two most significant bits of the CSR. These bits are read only.

#### 11.7.3.14 Cable Driver and Cable Receiver

The cable driver is a DC021 bus transceiver that takes applicable floppy drive signals from the FDC and drives them over a cable to the floppy disk drives. It is enabled as long as ENB DMA TEST H is not asserted. Signals to the floppy disk drives are:

WR DATA L
WR GATE L
DRV SEL 0 L
DRV SEL 1 L
SIDE SEL L
MOTOR ENB L
DIR SEL L
STEP L
IN USE L

The cable receiver, a 74LS244 buffer, receives signals from the floppy disk drives over the cable and delivers them to the FDC. The receiver is always enabled by the signal TS OE L. Signals from the floppy disk drives are:

INDEX L
TRACK 0 L
WR PROT L
RD DATA L
READY L

### 11.7.4 Floppy Disk Controller Commands

The 2797 Floppy Disk Controller accepts eleven commands. They should be loaded into the CSR only when the busy status bit, bit 0, is clear. The one exception to this is the force interrupt command which will be discussed in Section 11.7.4.4. Commands are divided into four types, they are:

- Seek Commands (type I commands)
- Read/Write Sector Commands (type II commands)
- Read Address and Read/Write Track Commands (type III commands)
- Force Interrupt Command (type IV commands)

Commands and types are summarized in Table 11-4. The bit assignments for each type of command are discussed in the following sections. The variables in columns 0 through 4 will also be explained.

#### 11.7.4.1 Floppy Disk Controller Disk Seeks (Type I Commands)

Five commands are recognized by the floppy controller as seek or head positioning commands. These commands are loaded into the command and status register (CSR) when the busy status bit, bit 0, is clear. These head positioning commands are shown in Table 11-5.

Table 11-4 Floppy Disk Controller Command Summary

| | | 2797 Bit Assignments | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Type | Command | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | Restore | 0 | 0 | 0 | 0 | h | v | r1 | r0 |
| 1 | Seek | 0 | 0 | 0 | 1 | h | v | r1 | r0 |
| 1 | Step | 0 | 0 | 1 | T | h | v | r1 | r0 |
| 1 | Step In | 0 | 1 | 0 | T | h | v | r1 | r0 |
| 1 | Step Out | 0 | 1 | 1 | T | h | v | r1 | r0 |
| 2 | Read Sector | 1 | 0 | 0 | m | L | E | U | 0 |
| 2 | Write Sector | 1 | 0 | 1 | m | L | E | U | 0 |
| 3 | Read Address | 1 | 1 | 0 | 0 | 0 | E | U | 0 |
| 3 | Read Track | 1 | 1 | 1 | 0 | 0 | E | U | 0 |
| 3 | Write Track | 1 | 1 | 1 | 1 | 0 | E | U | 0 |
| 4 | Force Interrupt | 1 | 1 | 0 | 1 | I3 | I2 | I1 | I0 |

Table 11-5 Head Positioning Commands

| Command | Description | If Success | If Failure |
|---|---|---|---|
| Restore | Seek Track 0 | Track Reg = 0 | Seek Error Set |
| Seek | Seek Track n | Track Reg = n | Seek Error Set |
| Step | One Step, Same Dir | Track Reg = n+1/-1 | Seek Error Set |
| Step In | One Step In | Track Reg = n+1 | Seek Error Set |
| Step Out | One Step Out | Track Reg = n-1 | Seek Error Set |

The head positioning commands accept four arguments as valid in the low byte of the CSR. Table 11-6 describes these four arguments.

Table 11-6 Head Positioning Command Arguments

| Bits | Action | Location |
|---|---|---|
| r1r0 | Selects stepping rate of motor | Bits 0 and 1 of CSR |
| V | Perform track number verify | Bit 2 of CSR |
| h | Head load flag | Bit 3 of CSR |
| T | Track update bit | Bit 4 of CSR |

#### 11.7.4.1.1 Restore Command

A Restore command is actually a seek to track zero. It is executed following power-up, the assertion of BINIT L, or when bit 9 of the CSR (motor enable) goes from clear to set. Upon receipt of this command, the TRACK 0 L signal to the FDC is sampled. If the signal is low, indicating the head is positioned over track zero, the track register is loaded with zeros and an interrupt is generated. If the signal is not low (head not over track zero), stepping pulses at the rate specified by the r1r0 field are issued from the FDC until the track zero input is activated. The track register is then loaded with zeros and the interrupt is generated. If the TRACK 0 L signal is not asserted within 255 stepping pulses, the FDC terminates the operation, interrupts, and sets the seek error status bit. If the V bit is set in the CSR, a verification will take place. The h bit allows the head to be loaded at the start of the command.

#### 11.7.4.1.2 Seek Command

A Seek command requires that the desired track number be loaded into the Data Register before the Seek command is initiated. A seek error will be generated if a non-existent track number is used. The track register keeps the number of the current track position. Track zero is the outermost track. The FDC will update the track register and issue stepping pulses in the appropriate direction until the contents of the track register are equal to the data register (the desired track location). A verification takes place if the V bit is set. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

#### 11.7.4.1.3 Step Command

Upon receipt of the Step command, the FDC issues a one-step pulse to the disk drive. The step motor direction is the same as the previous step command. A verification takes place if the V bit is set. The track register is updated if the T bit is set. The h bit allows the head to be loaded on the start of the command. An interrupt is generated at the completion of the command.

#### 11.7.4.1.4 Step In Command

Upon receipt of the Step In Command, the FDC issues a one-step pulse in the direction of the innermost track. If the T bit is set, the track register is incremented by one. A verification takes place if the V bit is set. The h bit allows the head to be loaded on the start of the command. An interrupt is generated at the completion of the command.

#### 11.7.4.1.5 Step Out Command

Upon receipt of the Step Out command, the FDC issues a one-step pulse in the direction of the outermost track (track zero). If the T bit is set, the track register is decremented by one. A verification is done if the V bit is set. The h bit allows the heads to be loaded at the start of the command. An interrupt is generated at the completion of the command.

### 11.7.4.2 Floppy Disk Controller Disk Reads and Disk Writes (Type II Commands)

The type two commands are:

Read Sector
Write Sector

Each will be discussed separately.

#### 11.7.4.2.1 Read Sector Command

The Read Sector command accepts four arguments as valid in the low byte of the CSR. Table 11-7 describes these four arguments.

**Table 11-7 Read Sector Command Arguments**

| Bit | Location | Action |
|---|---|---|
| m | Bit 4 of CSR | 0 = single sector<br>1 = multi-sector transfer |
| L | Bit 3 of CSR | 0 = 1024 bytes/sector<br>1 = 512 bytes/sector |
| E | Bit 2 of CSR | 0 = sample head load timing, no delay<br>1 = sample head load timing, 15ms delay |
| U | Bit 1 of CSR | 0 = select side zero<br>1 = select side one |

Note: The L bit of the CSR should always have a 1 in that bit position.

Once the head has been positioned over the desired track (cylinder), data may be transferred from the floppy disk to memory with the Read Sector command. The three bytes of the Memory Address Register (MAR) must be loaded with the 22-bit starting memory address, and the desired sector address must be loaded into the Sector Register.

Multiple sector transfers are accomplished by setting the m bit in the command word of the CSR. Multiple sector transfers are of the form sector x through end of track, x being the value contained in the Sector Register. Spiraled transfers or transfers crossing track boundaries require multiple read sector commands.

As each bit arrives serially from the disk, it is shifted through a data shift register until eight bits have arrived. The eight bits are then transferred to the Data Register (DR), and Data Request (DRQ) is set. The first byte is read from the the DR and loaded into the low byte of the MDR (Memory Data Register) that acts as a staging register. The second byte is read and loaded into the MDR high byte and a DMA request is issued.

Once the K.rx receives the assertion of BDMG I L from the P.ioj, it transfers the assembled word to memory along with correct parity for each byte. Upon completion of the transfer, the MAR is incremented by two. This sequence continues until all 512 bytes of the sector have been transferred to memory.

The busy bit of the CSR will be set while the read sector command is in progress. A two-byte CRC is computed as the data is read from the floppy and compared to the CRC read at the end of the data field. A CRC error will set if the computed CRC and the data field CRC do not match. An interrupt is generated at the completion of the read sector command. The low order bits of the CSR are redefined upon completion of the read sector command and must be examined for any errors. They are then reset when rewritten. These bits are shown in Table 11-8.

**Table 11-8 Status for Type II and Type III Commands**

| Bit | Name | Meaning |
|---|---|---|
| S7 | Not Ready | This bit, when set, indicates the drive is not ready. When reset, it indicates that the drive is ready. The type II and III commands will not execute unless the drive is ready. |
| S6 | Write Protect | On Read Sector: not used. On Read Track: not used. On any write: indicates a write protect. This bit is reset when updated. |
| S5 | Record Type | On Read Sector: indicates the sector type code from data field address mark. 1 = deleted data mark. 0 = data mark. On any write: forced to a zero. |
| S4 | Record Not Found (RNF) | When set, indicates the desired track, sector, or side was not found. This bit is reset when rewritten. |
| S3 | CRC Error | If S4 is set, an error was detected in the ID field; otherwise, the error was in the data field. This bit is reset when rewritten. |
| S2 | Lost Data | When set, indicates no response to the data request output (DRQ). This bit is reset to zero when rewritten. |
| S1 | Data Request | A copy of the DRQ output. When set, it indicates the data register (DR) is full on a read or empty on a write. This bit is reset to zero when rewritten. |
| S0 | Busy | When set, a command is being executed. When not set, a command is not being executed. |

#### 11.7.4.2.2 Write Sector Command

The Write Sector command accepts five arguments as valid in the low byte of the CSR. Table 11-9 describes these four arguments.

Once the head has been positioned over the desired track (cylinder), data may be transferred to the floppy disk from memory with the Write Sector command. The three bytes of the Memory Address Register (MAR) must be loaded with the 22-bit starting memory address, and the desired sector address must be loaded into the Sector Register.

Multiple sector transfers are accomplished by setting the m bit in the command word of the CSR. Multiple sector transfers are of the form sector x through end of track, x being the value contained in the Sector Register. Spiraled transfers or transfers crossing track boundaries require multiple write sector commands.

Table 11-9 Write Sector Command Arguments

| Bit | Location | Action |
|---|---|---|
| m | Bit 4 of CSR | 0 = single sector<br>1 = multi-sector transfer |
| L | Bit 3 of CSR | 0 = 1024 bytes/sector<br>1 = 512 bytes/sector |
| E | Bit 2 of CSR | 0 = sample head load timing, no delay<br>1 = sample head load timing, 15ms delay |
| U | Bit 1 of CSR | 0 = select side zero<br>1 = select side one |
| a0 | Bit 0 of CSR | 0 = write data address mark<br>1 = write deleted data address mark |

When the Write Sector command is loaded, the DRQ bit of the FDC sets and the K.rx issues a DMA request. Once K.rx receives the assertion of BDMG I L from the P.ioj, it transfers one word from memory. That word is loaded into the Memory Data Register (MDR). Both bytes of the word are checked for correct parity.

The data in the lower byte of the MDR is loaded into the Data Register (DR) of the FDC which clears the Data Request bit (DRQ) of the FDC. The FDC moves the byte into the data shift register which shifts each bit serially onto the floppy disk. The second byte of data is fetched from the high byte MDR and loaded directly into the DR.

When the third byte is requested, the DMA sequence is repeated and another word is fetched from memory. At the completion of each DMA word transfer, the MAR is incremented by two. This sequence continues until all 512 bytes of the sector have been transferred from memory to the floppy. A two-byte CRC is computed as the data is written to the floppy and appended to the data field.

If the data has not been loaded by the required time, a byte of zeros is written to the floppy and the Lost Data bit will be set.

The busy bit of the CSR is set while the Write Sector command is in progress. Again the low order bits of the CSR are redefined for a Write Sector command upon completion of the command (Table 11-8).

Partial writing of sectors is accomplished by writing the data and filling the balance with zeros.

Write precompensation is enabled whenever the track number is greater than $43_{10}$.

An interrupt is generated at the completion of the Write Sector command.

**NOTE**

**If the floppy disk drive is write protected, an interrupt is automatically generated if a Write Sector command is received.**

#### 11.7.4.3 Type III Commands

The type III commands are:

Read Address
Read Track
Write Track (Format)

Each will be covered separately.

##### 11.7.4.3.1 Read Address Command

Upon receipt of the Read Address command, the head is loaded and the busy status bit is set. The next encountered ID field is read from the disk into the DR, and a DRQ is generated. The FDC checks for validity, and the CRC status bit is set if there is an error. The track address of the ID field is written into the Sector Register for comparison. Upon completion of the command, an interrupt is generated and the busy status bit is cleared.

##### 11.7.4.3.2 Read Track Command

Upon receipt of the Read Track command, the head is loaded and the busy status bit is set in the CSR. Reading starts when an index pulse is detected and continues until the next index pulse is detected. All gap, header, and data bytes are assembled and transferred to the DR, and DRQs are generated for each byte. The accumulation of bytes is synchronized to each address mark detected. An interrupt is generated at the completion of the command.

##### 11.7.4.3.3 Write Track Command (Formatting)

Formatting the disk is accomplished by positioning the head over the desired track and issuing the Write Track command.

Upon receipt of the Write Track command, the head is loaded and the busy status bit is set in the CSR. Data Request is set immediately, and a word is fetched from memory and loaded into the DR. Writing starts when the first index pulse is detected and ends when the next index pulse is detected, at which time the interrupt is generated. If the DR has not been loaded by the time the index pulse has been detected, the operation is terminated making the device not busy. The Lost Data status bit is set and an interrupt is generated. This sequence continues from index pulse to index pulse.

##### 11.7.4.3.4 Status Register Summary

Table 11-10 is a summary of the Status Register bits for all commands discussed so far.

**Table 11-10 Status Register Summary**

| Bit | All Type I Commands | Read Address | Read Sector | Read Track | Write Sector | Write Track |
|---|---|---|---|---|---|---|
| S7 | Not Ready | Not Ready | Not ready | Not Ready | Not Ready | Not Ready |
| S6 | Write Protect | 0 | 0 | 0 | Write Protect | Write Protect |
| S5 | Head Loaded | 0 | Record Type | 0 | 0 | 0 |

Table 11-10 (Cont.) Status Register Summary

| Bit | All Type I Commands | Read Address | Read Sector | Read Track | Write Sector | Write Track |
|---|---|---|---|---|---|---|
| S4 | Seek Error | RNF | RNF | 0 | RNF | 0 |
| S3 | CRC Error | CRC Error | CRC Error | 0 | CRC Error | 0 |
| S2 | Track 0 | Lost Data | Lost Data | Lost Data | Lost Data | Lost Data |
| S1 | Index Pulse | DRQ | DRQ | DRQ | DRQ | DRQ |
| S0 | Busy | Busy | Busy | Busy | Busy | Busy |

#### 11.7.4.4 Type IV Commands

The Force Interrupt command is the only type IV command. It is used to terminate a multiple sector read or write or to ensure type I status in the Status Register. This command can be loaded into the Command Register at any time. If a command is being executed (busy status bit set), that command is terminated and the busy status bit is reset.

The lower four bits of the command determine the conditional interrupt as follows:

- I0 = Not-Ready to Ready Transition
- I1 = Ready to Not-Ready Transition
- I2 = Every Index Pulse
- I3 = Immediate Interrupt

The conditional interrupt is enabled when the corresponding bit positions of the command (I0-I3) are set to a one. When the condition for interrupt is met, the INTR REQ H line of the FDC is set, indicating that the condition specified has occurred. If I0-I3 are all set to zero, no interrupt occurs but the command presently being executed will be terminated immediately.

When using the immediate interrupt command (I3 = 1), an interrupt will be immediately generated and the current command terminated. Reading the status or writing to the CSR will not clear this command. Setting I0-I3 to zeros is the only way to clear the immediate interrupt command.

More than one condition may be set at a time. If for example, I1 = 1 and I2 = 1, an OR function is performed so that the interrupt is generated on either a ready to not-ready transition or on the next index pulse.

## 11.8 JUMPER CONFIGURATION

There are five jumpers on the HSC70 Memory Module. Table 11-11 describes the purpose and normal configuration of each jumper for this module.

**NOTE**

**The purpose of the factory set potentiometers and switch is described in Section 11.7.3.9.**

Table 11-11 HSC70 Memory Module Jumper Description

| Jumper | Normal Configuration | Description |
|---|---|---|
| W1 | Out | Installed for two banks of Data Memory<br>Removed for four banks of Data Memory |
| W2 | In | Installed for normal operation<br>Removed when using external clock input |
| W3 | In | Installed for 2 MHz clock (500 Kbytes/second)<br>Removed for 1 MHz clock |
| W4 | Out | Installed for 1 MHz clock (250 Kbytes/second)<br>Removed for 2 MHz clock |
| W5 | Out | Installed for 250 Kbytes/second transfer rate<br>Removed for 500 Kbytes/second transfer rate |

# CHAPTER 12
# DATA CHANNEL MODULE

## 12.1 INTRODUCTION

This chapter contains the logical and functional description of the Data Channel Module. There are two kinds of Data Channel Modules:

- Disk Data Channel—L0108-YA—HSC5X-BA—K.sdi
- Tape Data Channel—L0108-YB—HSC5X-CA—K.sti

The differences between the two modules are the microcode in the PROM and a jumper.

**NOTE**

**When talking about the Data Channel Module in general, the mnemonic K.sdi/K.sti will be used. When talking about a specific data channel, the mnemonics K.sdi or K.sti will be used.**

The K.sdi/K.sti is an L series extended-hex module which is the interface between the HSC and the Standard Disk/Tape Interconnect Bus (SDI/STI). It uses several of the 2900-family bit-slice devices, supported by medium/small scale integration transistor/transistor logic. The architecture of K.sdi/K.sti consists of 2900-family components organized as dual sequencers alternately accessing a common control store and supporting a single 16-bit data path.

The K.sdi supports up to four disk drives, each connected radially via the SDI Bus. The maximum serial data rate from any drive is 25 Mbit/second. Different drive types with different serial data rates may be simultaneously connected to one K.sdi. However, all disks must adhere to the sector formats for 16-bit and 18-bit data.

The K.sti supports up to four tape formatters, each connected radially via the STI Bus.

The K.sdi/K.sti does not use control registers to communicate with the P.ioc and other controllers in the HSC. Communications between the K.sdi/K.sti and the P.ioc and other controllers of the HSC are performed using memory resident control structures.

Go/No-Go diagnostics for K.sdi/K.sti are microcoded and are entirely resident in PROM along with the functional microcode.

## 12.2 STATUS LEDS

The red and green LEDs indicate the status of the K.sdi/K.sti module. The module powers up with the red LED on and the green LED off. Upon successful completion of the power up diagnostics, the red LED is turned off and the green LED is turned on.

## 12.3 MICROINSTRUCTION WORD

The microinstruction word from Control Store controls the operation of the K.sdi/K.sti. The microinstruction word is 48 bits wide. Each word is divided into sixteen fields, and each field controls different portions of the K.sdi/K.sti hardware. Figure 12-1 shows the fields and their position within the microinstruction word. Table 12-1 contains a brief description of each field. Section 12.7 has a detailed description of each field.

**Figure 12-1 Microinstruction Word**

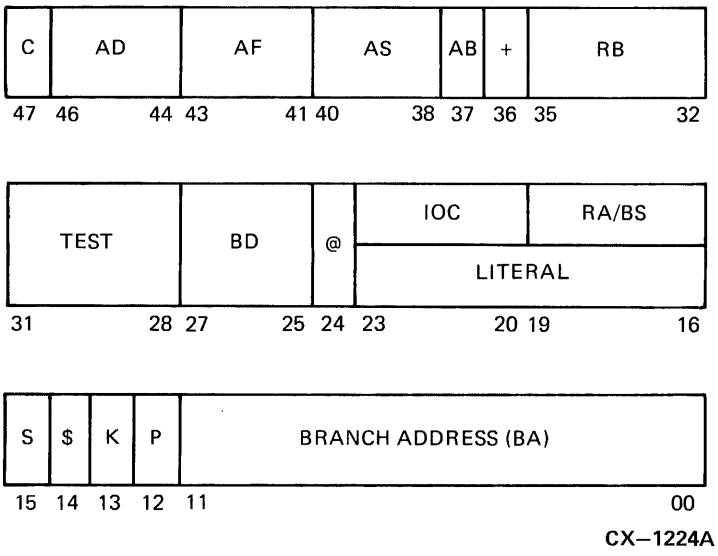


Table 12-1 Microinstruction Word Field Description

| Field | Description |
|---|---|
| C | The C field is the carry-in into the ALU for the current microinstruction. It steers the literal to the most significant byte for AF fields 3 through 7. |
| AD | The AD field, ALU destination, determines the routing of the ALU output data internal to the 2901. |
| AF | The AF field, ALU Function, determines the operation to be performed by the ALU. |
| AS | The AS field, ALU source, selects two of the five possible sources for the ALU. |
| AB | The AB field, ALU to BUS, determines whether ALU or the bus source decoder is the source for the data on the BUS. |
| + | The + field, EXEC IF COND TRUE, is one bit long. This bit is combined with the following signals to generate INHIBIT EXEC (inhibit execute).<br>• TST COND TR (test condition true)<br>• COND EXC (conditional execute)<br><br>INHIBIT EXEC prevents:<br>• Any ALU operation of each instruction for which it is active<br>• The IOC Control latches from being enabled<br>• The destination decoders from being enabled |
| RB | The RB field, B Port Register, selects one of the sixteen internal registers to be read and/or written by the 2901 ALU. If a value of $5_8$ or $7_8$ is in the AS field, the A Port Register field (RA field) is forced to the same value as the RB field. |
| TEST | The TEST field, TEST CONDITION <03:00>, selects one of 16 signals for the conditional operations of the sequencers. The hardware includes two separate selectors, one for each of the two sequencers. |
| BD | The BD field, bus destination, selects the register to be loaded with the data on the BUS lines. |
| @ | The @ field, IOC enable, when set, permits the IOC field to modify the IOC latches. This bit is not set when using the LITERAL field. |
| IOC | The IOC field, input/output control, if enabled by IOC ENABLE, modifies the IOC Latches. Bits <22:20> select the desired latch, and bit 23 sets or clears that latch. |

Table 12-1 (Cont.) Microinstruction Word Field Description

| Field | Description |
|---|---|
| RA | The RA field, A Port Register, selects one of the sixteen internal 2901 registers to be operated on by the ALU. The RA field is a shared field and is only enabled if the AS field does not equal $5_8$ or $7_8$ and bit AB does not specify the LITERAL field. |
| BS | The BS field, bus source, selects one of the K.sdi/K.sti registers which are external to the ALU. The contents of the selected register are enabled onto the BUS lines, routed through the input multiplexer, and presented to the D inputs of the 2901 ALU. |
| S | The S field, test sense, is XORed with the output of the Test Multiplexer to generate TST COND TR (test condition true). TST COND TR is then combined with branch address equal to $7777_8$ and JUMP CNTL 1 to determine whether the sequencer will output the Program Counter, the Stack, or the Branch Address. |
| $ | The $ field, conditional execute, enables all instructions when clear. When set, this bit enables the conditional execution of the IOC, bus destination, and ALU instructions. |
| K | The K field, jump control, is logically combined with TST COND TR and BR ADR <11:00> equal to all ones ($7777_8$) to control the next operation of the 2911 Sequencers. This combination controls the operation of the internal Stack, and it determines whether the sequencer will output the:<br>• Program counter<br>• Stack contents<br>• Branch address field |
| P | The P field, instruction parity, is the single parity bit for the 48-bit microinstruction word. This bit is calculated odd parity for the other 47 bits. If the number of 1 bits in the other 47 bits is even, then this bit is a 1 making the 48-bit microinstruction word odd parity. If the number of 1 bits in the other 47 bits is odd, then this bit is a 0 making the 48-bit microinstruction word odd parity. |
| BA | The BA field, branch address, provides the instruction address for conditional and unconditional branches and subroutine calls. Also, during the execution of certain instructions, this field provides supplemental data and control information. |

## 12.4 DATA CHANNEL MODULE BLOCK DIAGRAM

Figure 12-2 is a block diagram of the K.sdi/K.sti showing the principal hardware elements and data paths. Each of these elements are described in the following sections. These sections describe the block diagram as if it were made up of two different components. The left side of the block diagram is control and interface to the rest of the HSC. The right side of the block diagram is input and output to the Standard Disk/Tape Interconnect.

NOTE

**Notice on Figure 12-2 the bus labeled K.SDI BUS (BF BUS, KL BUS, INV BUS) D15:00. The names represent the same bus with different buffers because of dc loading, ac loading, and the way the Scratchpad RAM inverts the bus. This chapter will consistently refer to this bus as BUS D<15:00>.**

**To differentiate the various serializer/deserializers, encoder/decoders, receiver/drivers, and multiplexers on the right side of the block diagram, a name, a coordinate, and a page number appears in each block. For example, the block containing SER DESER, D7, P12 will be referred to as the Upper Real-Time Controller State Serializer (D7 P14).**

## 12.5 CONTROL AND INTERFACE THE TO REST OF THE HSC

The control and interface to the rest of the HSC of the K.sdi/K.sti has almost all the same components as the K.pli. If you look at the left side of Figure 12-2 you will see some of the same blocks that are in the K.pli block diagram. In the following sections we will cover the control portion of the K.sdi/K.sti:

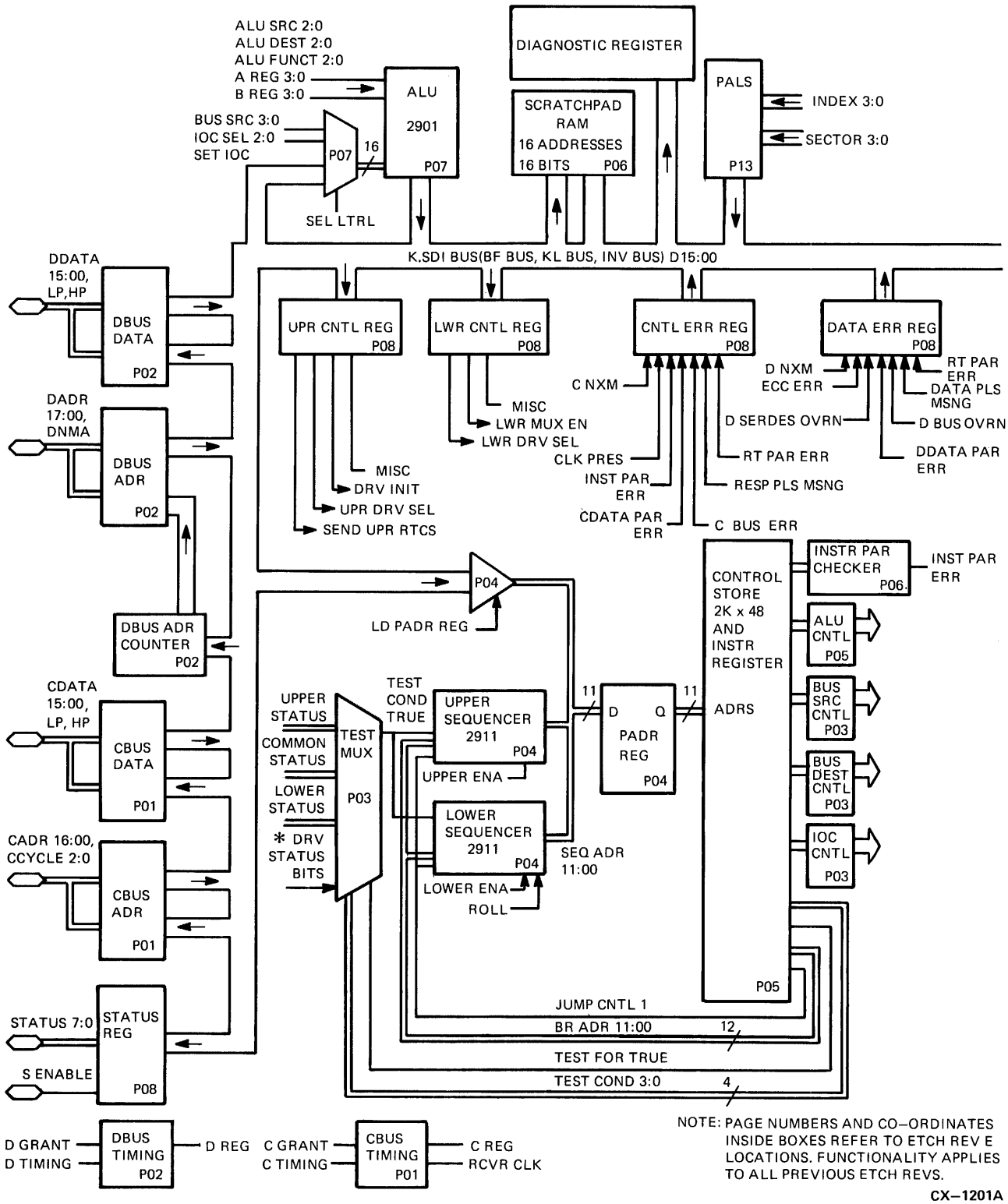
- Sequencers
- Test Multiplexer
- Program Address Counter
- Control Store and Instruction Register
- Arithmetic and Logic Unit (ALU)
- Scratchpad RAM
- Upper and Lower Sequencer Control Registers
- Control and Data Error Registers
- PALs

The PALS will be discussed along with the SDI/STI interface.

The interfaces to the rest of the HSC are on the left side of the block diagram. These elements are:

- Data Bus Interface
- Control Bus Interface
- Status Bus Interface

## Figure 12-2 Data Channel Module Block Diagram

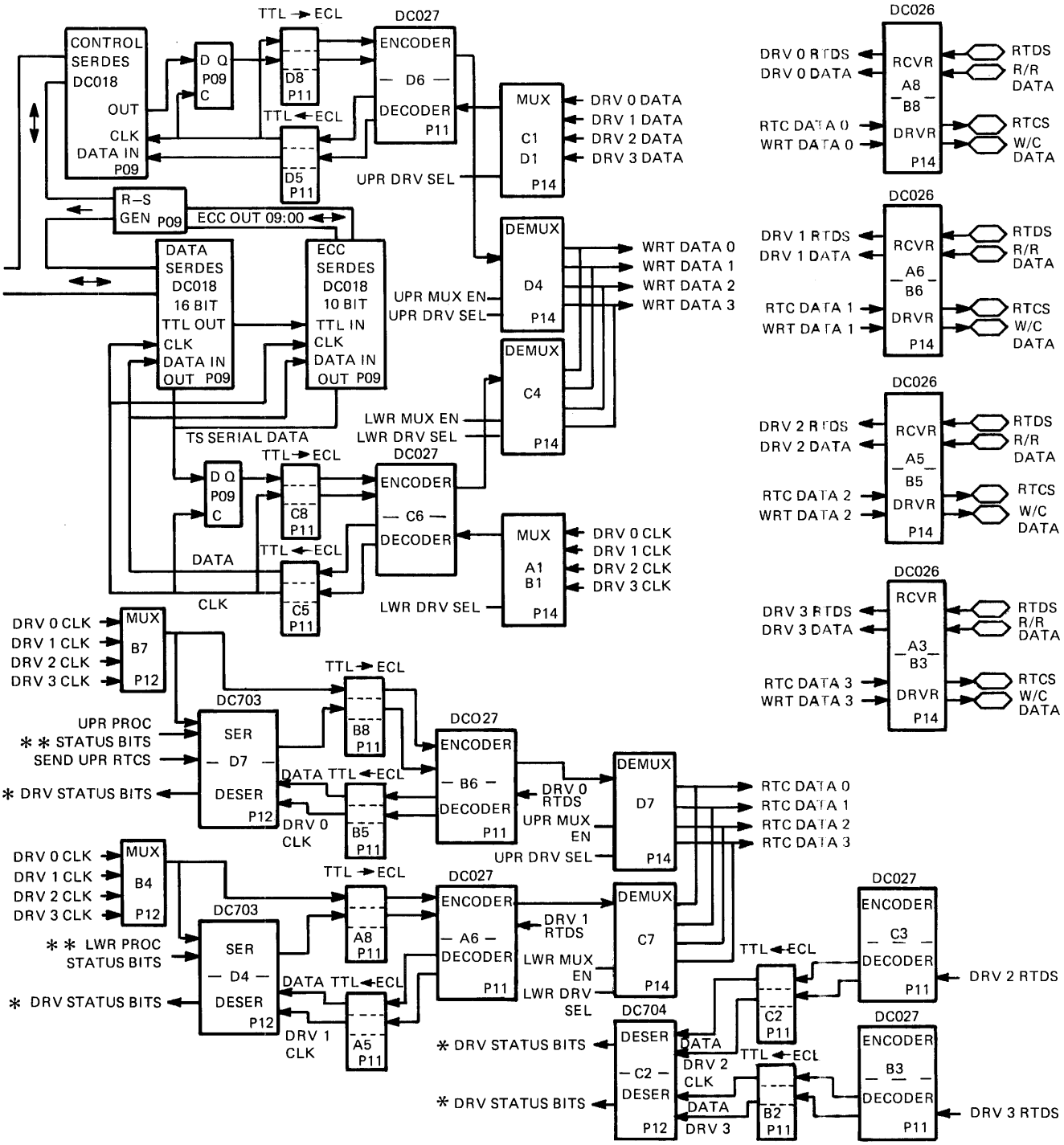



NOTE: PAGE NUMBERS AND CO-ORDINATES INSIDE BOXES REFER TO ETCH REV E LOCATIONS. FUNCTIONALITY APPLIES TO ALL PREVIOUS ETCH REVS.

CX-1201A
Sheet 1 of 2

(Continued on next page)

Figure 12-2 (Cont.) Data Channel Module Block Diagram




* DRV STATUS BITS: AVAIL, R/W RDY, ATTN, RDY TO REC, INDEX, SECTOR

** RTCS BITS: DRV INIT, RCVR RDY, RD GATE, WRT GATE

CX–1201A
Sheet 2 of 2

### 12.5.1 Sequencers

In the lower-left corner of Figure 12-2 you will see the two sequencers. The use of these two sequencers which alternately and independently access a common Control Store is the heart of the architecture of the K.sdi/Ksti. The two sequencers, in conjunction with the Control Store and Arithmetic and Logic Unit (ALU), are designated Upper Sequencer and Lower Sequencer. The two sequencers control different portions of the external interfaces, and they have separate and independent functions.

**NOTE**

**In the K.pli, the sequencers are named Control Sequencer and Data Sequencer. The two sequencers in the K.sdi/K.sti have the same function as those in the K.pli; however, they are named Upper and Lower. The Upper Sequencer has the same function as the Control Sequencer and the Lower Sequencer has the same function as the Data Sequencer.**

The Upper Sequencer is responsible for initiating the work that the K.sdi/K.sti does in response to commands for the P.ioc/P.ioj. This includes examining queues for work to do, acquiring and retiring buffers, and so on. The activity level of this sequencer is fairly high and fairly constant.

The Lower Sequencer handles all aspects of a data transfer between Data Memory and the disk/tape drive. During a data transfer, the Lower Sequencer is busy; at all other times it is idle.

One sequencer does not help out or share the work assigned to the other sequencer. Each sequencer has its own job to do.

With a basic clock period of 150 nanoseconds, each sequencer has an apparent micro-cycle time of 300 nanoseconds. One advantage of this technique is that no effective time is lost by either sequencer when executing a branch, since the Control Store is accessed by the branching sequencer while the other sequencer is executing.

The sequencers are the control section of the K.sdi/K.sti. Each sequencer consists of three 2911 bit-slice devices connected to form a 12-bit address. Figure 12-3 is a block diagram of one 2911. Each bit-slice is 4 bits wide and consists of the following logic:

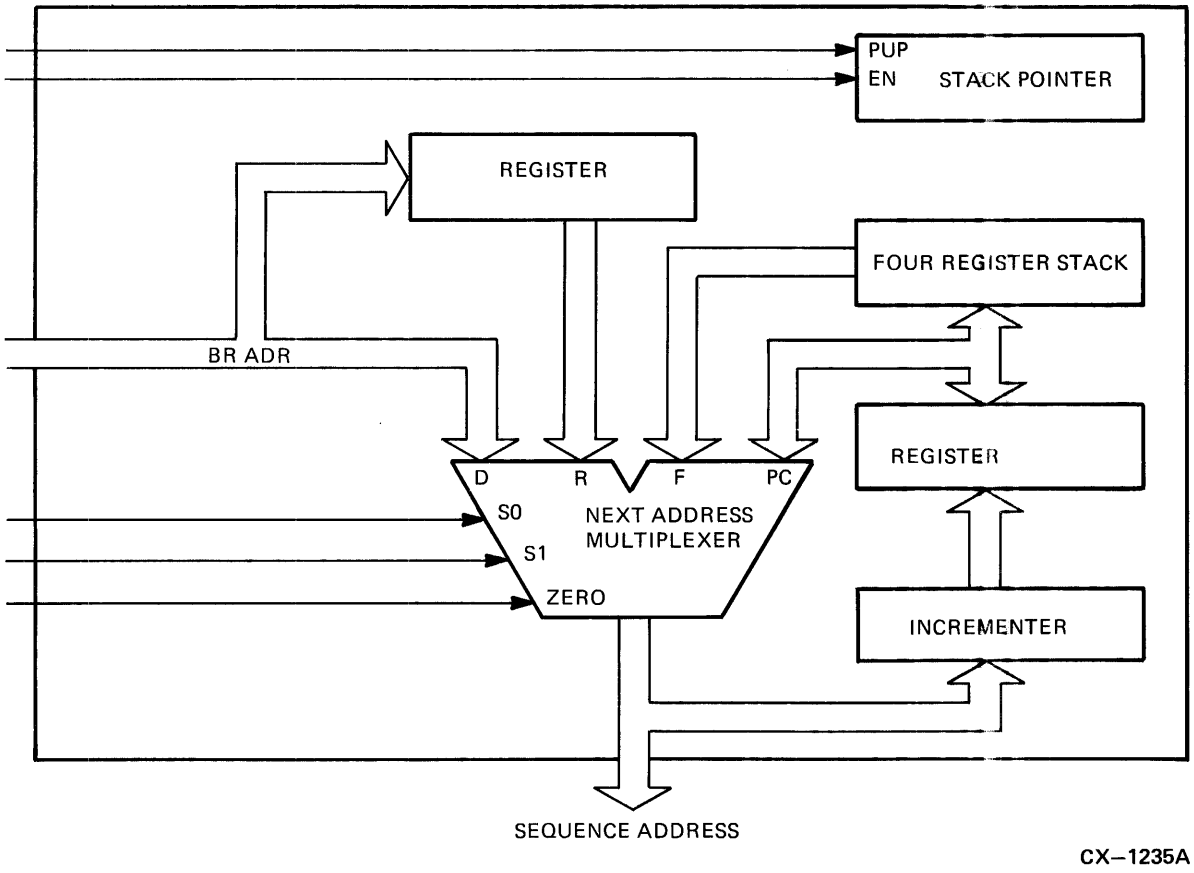
- Program Counter Register
- Incrementer
- Four Register Stack
- Stack Pointer
- Next Address Multiplexer
- Internal logic to control these functions

The sequencers in the K.sdi/K.sti connect three 2911s for a 12-bit wide address. This 12-bit address allows the sequencer to address 4 Kwords of Control Store. This address always points to the next instruction by automatically incrementing after the instruction fetch. The stack is 12-bits wide and four words deep and is used along with the stack pointer during branch operations.

The internal multiplexer within the sequencers is capable of choosing one of the following for output to the program address register:

- The Stack
- The Program Counter Register
- The branch address supplied from the Control Store

Figure 12-3 2911 Sequencer Block Diagram




As connected on the K.sdi/K.sti, the sequencers perform the following operations:

- CONTINUE—Increments the Program Counter Register and places the new address on the SEQ ADR <11:00> lines.
- RETURN FROM SUBROUTINE (pop stack)—Loads the Program Counter Register with the current value on the top of the stack and gates this value to the SEQ ADR <11:00> lines. The Stack Pointer is then decremented.
- JUMP TO BRANCH ADDRESS—Loads the Program Counter Register with the current value of the branch address and gates this address to the SEQ ADR <11:00>. The Stack and Stack Pointer are not affected.
- SUBROUTINE CALL (push stack)—Increments the Stack Pointer and pushes the current value of the Program Counter Register onto the Stack. Loads the Program Counter Register with the current value of BR ADR <11:00> and gates this address to SEQ ADR <11:00>.

### 12.5.2 Test Multiplexer

The sequencers must have the ability to test different conditions on the K.sdi/K.sti. Figure 12-2 shows a test multiplexer to the left of the two sequencers. This test multiplexer contains two separate 16-input multiplexers to select test conditions for the two sequencers. Bits 31 through 28 of the Control Store generate TEST COND <3:0> that select one of the 16 inputs. Either the true or the complement of the selected test condition may be selected. This is accomplished by the TEST FOR TRUE (bit 15 from the Control Store). A synchronization circuit is included to prevent incorrect testing of asynchronous test conditions. Available test conditions include:

- Drive status
- ALU status
- IOC Latches
- Error indicators

Section 12.7.8 contains a description of each input to the test multiplexers.

### 12.5.3 Program Address Register

The sequencers must address the Control Store through the Program Address Register (PADR REG) shown in Figure 12-2. This register accepts the sequencer outputs on each clock and presents the instruction address to the Control Store for the fetch operation.

The sequencer outputs may be disabled and the Program Address Register forcibly loaded from the ALU by LD PADR REG. For example, the ALU may execute a calculated jump by calculating the jump address from a table and writing the data to the Program Address Register from BUS D<11:00>. During this time, the outputs of the sequencers are disabled.

The Upper Sequencer microcode can force the Lower Sequencer to trap to address $3777_8$. The Upper Sequencer microcode may do this if it failed to communicate with the Lower Sequencer through the Scratchpad RAM. The Lower Sequencer may be stuck in a loop. The logic in Figure 12-4 shows that the decoder that loads the Program Address Register is disabled when a FORCE LWR TRAP is asserted. When the Lower Sequencer tries to load the Program Address Register, the decoder is disabled and the outputs from the 2911 are pulled high and a $3777_8$ is loaded into the Program Address Register.

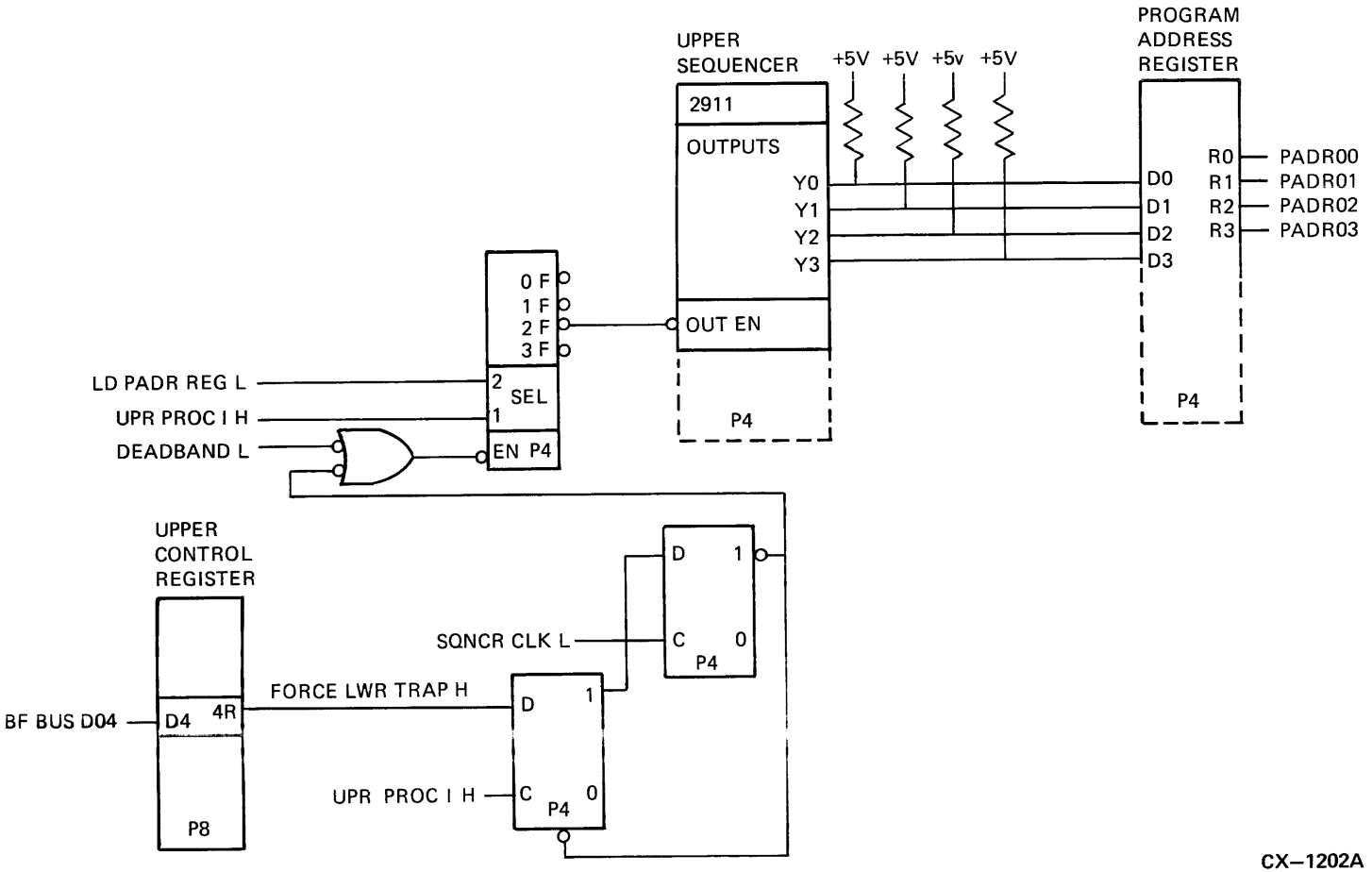
### 12.5.4 Control Store and Associated Logic

Figure 12-2 shows the Control Store and Instruction Register located to the lower left of center. The Instruction Register consists of latches on the output of the Control Store. These latches drive the following control logic:

- ALU Control
- Bus Source Control
- Bus Destination Control
- IOC Control

Along with the Instruction Register and control logic, we will look at the Instruction Parity Checker logic in this section.

**Figure 12-4 Force Lower Sequencer to Vector $3777_8$ Logic**



#### 12.5.4.1 Control Store and Instruction Register

Figure 12-2 shows the PROM Control Store to the right of the sequencers. All microcode instructions are contained in the Control Store. This Control Store is 48 bits wide (including parity). The Control Store consists of twelve 2K by 4 PROMs. One microinstruction is fetched from the Control Store during each 150 nanosecond clock cycle. At the end of the clock cycle, the microinstruction is loaded into a latch (Instruction Register) where it is held for the following execution cycle.

The Upper Sequencer has the ability, in diagnostic mode, to command the Lower Sequencer to consecutively fetch and test parity in all Control Store locations. This is accomplished with the ROLL signal on the output of the IOC Control latches. (Refer to Section 12.5.4.6.)

A detailed description of each bit of the microinstruction word is contained in Section 12.7.

#### 12.5.4.2 Instruction Parity Checker

The Instruction Parity Checker checks for odd parity over the 48 bits of the instruction from the Control Store. This parity bit is gated to the test select multiplexer. Bit 12 is the odd parity bit for the 48 bits of the instruction word. The Instruction Parity Checker is located at the top of the Control Store in Figure 12-2.

If a Control Store parity error is detected when K.sdi/K.sti is in normal mode, the sequencers are stopped, resulting in a complete halt of all K.sdi/K.sti operations. Recovery occurs only after the P.ioc initializes the module.

#### 12.5.4.3 ALU Control

The ALU Control (ALU CNTL) logic latches the bits from the Control Store to control the ALU by:

- Determining the routing of the ALU output internal to the 2901
- Determining the operation to be performed by the ALU
- Selecting two of the possible five inputs to the ALU
- Controlling the output of the ALU to the BUS D<15:00>

Figure 12-5 shows which Control Store bits generate which ALU control. Each bit is described in Section 12.7.

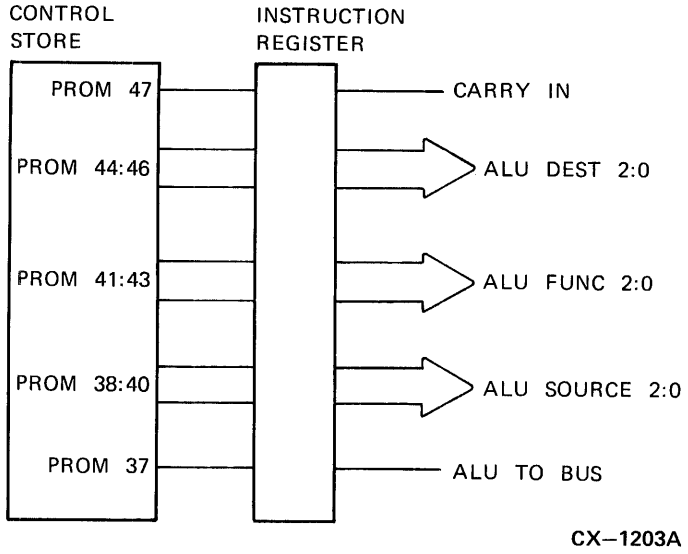
#### 12.5.4.4 Bus Source Control

The Bus Source Control (BUS SRC CNTL) logic decodes Control Store bits <18:16> to enable the different sources to BUS D<15:00> for each sequencer. The Bus Source Control logic is located on the output of the Control Store in Figure 12-2. Figure 12-6 is a block diagram of the Bus Source Control logic.

ALU TO BUS disables the two decoders. ALU TO BUS is generated by the ALU Control logic. (Refer to Section 12.5.4.3.)

Each sequencer reads from its own sources. The signals UPR PROC 0 and UPR PROC 1 determine which decoder is enabled. Notice that the Scratchpad RAM is the only source that can be read by both sequencers (RD SCRATCH REG).

Figure 12-5 ALU Control Logic




### 12.5.4.5 Bus Destination Control

The Bus Destination Control (BUS DEST CNTL) logic decodes Control Store bits <27:25> from Control Store to load registers (for example, LD STATUS REG) and start timing (for example, SET C REQ). Signals UPR PROC 0 and UPR PROC 1 control which destination decoder is selected. The Bus Destination Control logic is located on the output of the Control Store in Figure 12-2. Figure 12-7 is a block diagram of the Bus Destination Control logic.

Each sequencer writes to its own destinations. The signals UPR PROC 0 and UPR PROC 1 determine which decoder is enabled. Notice that there is a common destination decoder which allows either sequencer to write to Scratchpad RAM, load the Diagnostic Register, or load the Program Address Register.

Another enabling signal to the Bus Destination Control logic is destination clock (DEST CLK). Notice that a qualifier to DEST CLK is INHIBIT EXEC. The test multiplexers assert INHIBIT EXEC when a test is false. The microcode uses this feature to inhibit loading a register if the test is not met. An example would be the microcode moving data from the Scratchpad RAM to the Control Bus Data Register but first testing to see if the bus is busy. (This operation can be coded in one instruction.) If the bus is busy when the sequencer executes this instruction, the Scratchpad RAM data would be gated onto the bus but would not be clocked into the Control Bus Data Register. The signal INHIBIT EXEC would disable DEST CLK.

Figure 12-6 Bus Source Control Logic

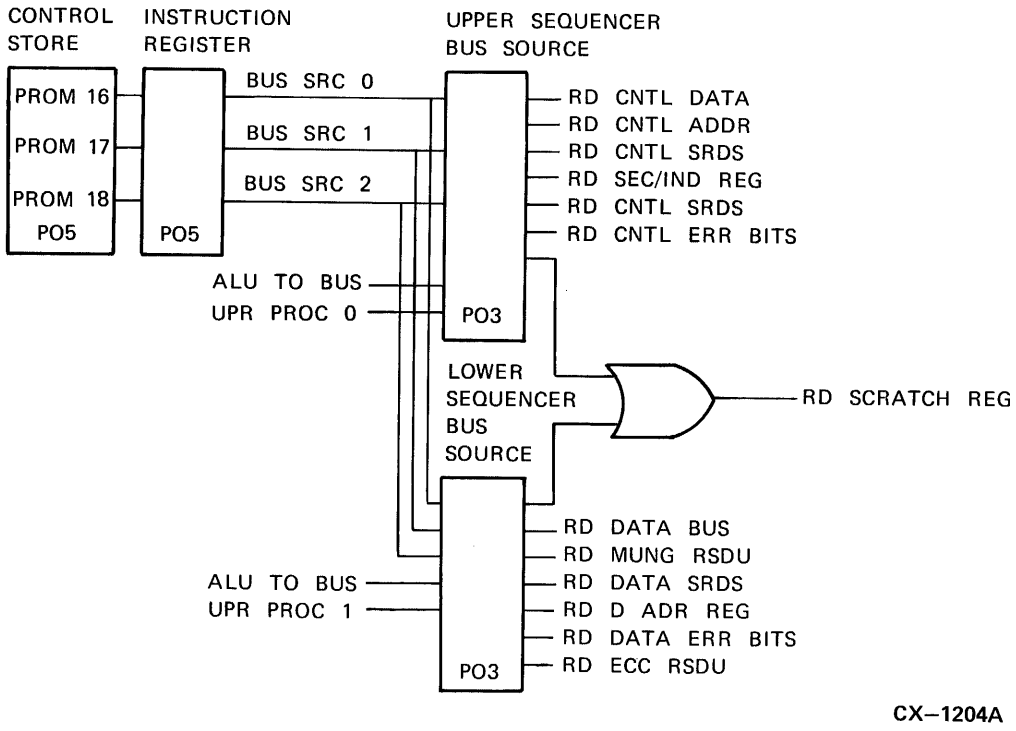
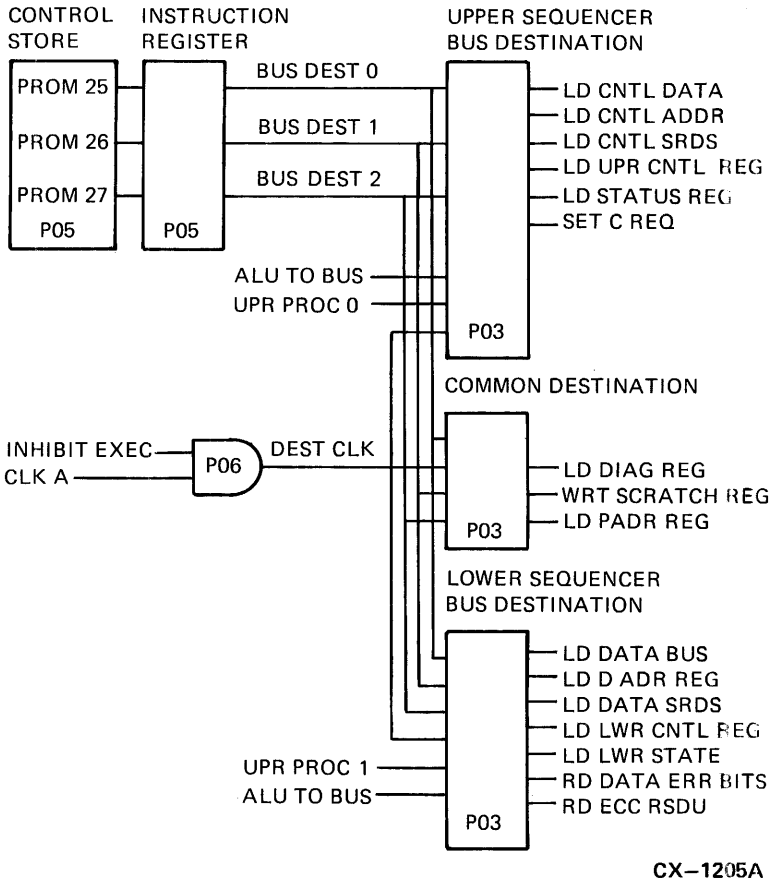


**Figure 12-7 Bus Destination Control Logic**




### 12.5.4.6 IOC Control

The term IOC stands for Input/Output Control. The IOC Control (IOC CNTL) logic, shown to the right of the Control Store in Figure 12-2, decodes Control Store bits <22:20> from the Control Store and sets or clears the IOC Control latches. These latches may not be set or cleared if the LITERAL field, to the ALU, is being used. Modification of these latches is specified if IOC ENABLE is set. Figure 12-8 is a block diagram of the IOC Control logic. Table 12-2 describes each bit in the IOC Control latches.

Figure 12-8 IOC Control Logic

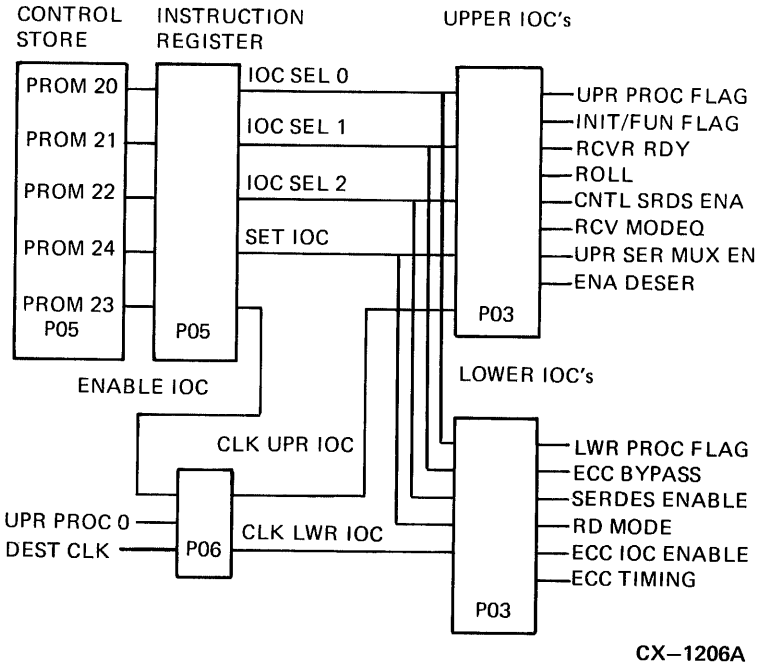



Table 12-2 IOC Control Bit Description

| Bit | Definition |
|---|---|
| UPR PROC FLAG | This bit is set or cleared by the Upper Sequencer and is an input to the Lower Test Multiplexer. This bit is used for microcode communication from the Upper Sequencer to the Lower Sequencer. |
| INIT/FUN FLAG | This bit is set or cleared by the Upper Sequencer. This bit controls the two LEDs. When set, the green LED is on and the red LED is off. When clear, the red LED is on and the green LED is off. This bit is available for the Upper Sequencer to test through the test multiplexer. |
| RCVR RDY | This bit is set or cleared by the Upper Sequencer. When set, this bit indicates that the K.sdi/K.sti is ready to receive a level 2 response from the disk unit or tape unit. (Refer to Section 12.6 for definition of level 2 response.) When cleared, it indicates that the K.sdi/K.sti is not ready to receive a level 2 response. |

Table 12-2 (Cont.) IOC Control Bit Description

| Bit | Definition |
|---|---|
| ROLL | This bit is set or cleared by the Upper Sequencer. When set, it prevents the Lower Sequencer from executing all ALU, I/O, and branch operations. The Lower Sequencer is forced to increment its address through Control Store in order to fetch and check instruction parity. When clear, both sequencers operate normally. |
| CNTL SRDS ENA | This bit is set or cleared by the Upper Sequencer. When set, this bit enables the Control SERDES. When reset, this bit resets all the internal registers and counters of the Control SERDES. |
| RCV MODE | This bit is set or cleared by the Upper Sequencer. This bit switches the Control SERDES between receive mode and transmit mode. When set, the Control SERDES is in receive mode; when clear, the Control SERDES is in transmit mode. |
| UPR SER MUX EN | This bit is set or cleared by the Upper Sequencer. When set, this bit enables WRT DATA <3:0> and RTC DATA <3:0> multiplexers. WRT DATA <3:0> and RTC DATA <3:0> are wired ORed for information from either the Upper Sequencer or Lower Sequencer. When cleared, the output of the multiplexer is disabled. |
| ENA DESER | This bit is set or cleared by the Upper Sequencer only. When set, this bit enables the output of the real-time drive state from the Real-Time State Deserializer for a selected port. When cleared, this bit disables the output of the Real-Time State Deserializer. |
| DATA PROC FLAG | This bit is set or cleared by the Lower Sequencer and is an input to the Upper Test Multiplexer. This bit is used for microcode communication from the Lower Sequencer to the Upper Sequencer. |
| ECC BYPASS | This bit is set or cleared by the Lower Sequencer. When set, this bit disables the output of the Data SERDES to the encoders and enables the output of the ECC SERDES to the encoders. When clear, this bit enables the output of the Data SERDES to the encoders and disables the output of the ECC SERDES to the encoders. |
| SERDES ENABLE | This bit is set or cleared by the Lower Sequencer. When set, this bit enables the Data SERDES. When reset, this bit resets all the internal registers and counters of the SERDES. |

Table 12-2 (Cont.) IOC Control Bit Description

| Bit | Definition |
|---|---|
| RD MODE | This bit is set or cleared by the Lower Sequencer. When set, this bit is an enabling signal to Data Bus Transceiver and configures the Data SERDES and ECC SERDES to receive data from the disk or tape unit. When clear, this bit configures the Data SERDES and ECC SERDES to receive data from the Data Bus Transceivers. |
| ECC IOC ENABLE | This bit is set or cleared by the Lower Sequencer. When set, this bit enables the ECC logic and ECC SERDES. When clear, this bit disables the ECC logic and resets all the internal registers and counters of the SERDES. |
| ECC TIMING | This bit is set or cleared by the Lower Sequencer. When set, this bit enables the ECC logic. When cleared, this bit disables the ECC logic. |

### 12.5.5 Arithmetic and Logic Unit

Figure 12-2 shows the Arithmetic and Logic Unit (ALU) in the upper-left corner. The ALU is implemented with four 2901 bit-slice devices, providing a 16-bit data path. The 2901 bit-slice ALU chip can perform addition, subtraction and logic operations. Its internal data paths and registers are four bits wide (Figure 12-9). Nine control signals and eight register-select lines determine the ALU function, operand source, and destination. The ALU has the following characteristics:

- A 2-port RAM comprised of 16 registers. The RAM registers are read via their A or B Data Out ports.
- A 4-bit shift register on the RAM data input. The output of this register is connected in ring counter fashion to the input.
- A 4-bit Q register for assisting with multiplication and division algorithms.
- A 4-bit shift register on the Q register input.
- A multiplexer that selects the output of the Q register shifter or the ALU output for input to the Q Register.
- Two multiplexers that select the ALU inputs. These inputs may be the direct input, the A or B RAM, logic zero, or the Q Register.
- An output multiplexer that selects the ALU output or port A of the RAM and gates it to BUS.
- A 2-input ALU capable of performing the operations shown in Table 12-3.

Figure 12-9 2901 ALU Block Diagram

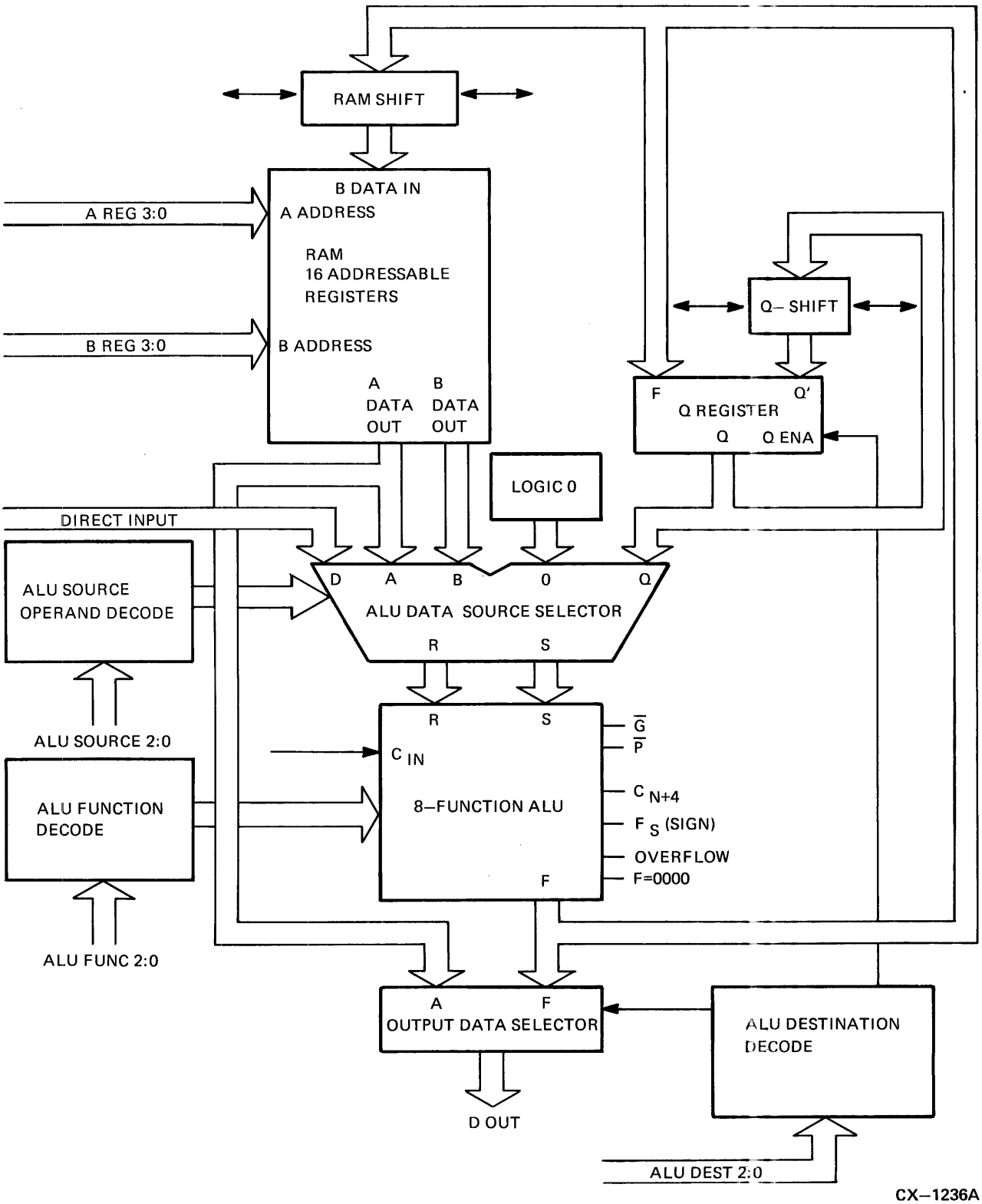


Table 12-3 ALU Operations

| Operation | Operands Used |
|---|---|
| Addition | R + S<br>R + S + 1 |
| Subtraction | R—S<br>S—R<br>R—S—1<br>S—R—1 |
| Logical OR | R OR S |
| Logical AND | R AND S |
| Logical NOTRS | (NOT R) AND S |
| Logical XOR | R XOR S |
| Logical XNOR | R XNOR S |

Organization of the 2901 is such that shift operations are separate from, and independent of, the arithmetic and logical operations, and are effective only on the inputs to the internal Q and RAM registers. The results of ALU operations are not shifted at the Y outputs. The ALU state flags reflect the ALU results before any shift.

Shifts may be performed at the RAM register inputs, or at the inputs of both the RAM and Q registers in one instruction. All are 16-bit shifts, and the ALU carry is never affected by any shift operation. Shifts performed on the Q register are end off. That is, on a left shift the contents of bit 15 are lost and 0 is shifted into bit 00. On a right shift the contents of bit 00 are lost and a 0 is shifted into bit 15. Shifts performed on the RAM registers are circular. That is, on a left shift the contents of bit 15 appear in bit 00, and on a right shift the contents of bit 00 appear in bit 15.

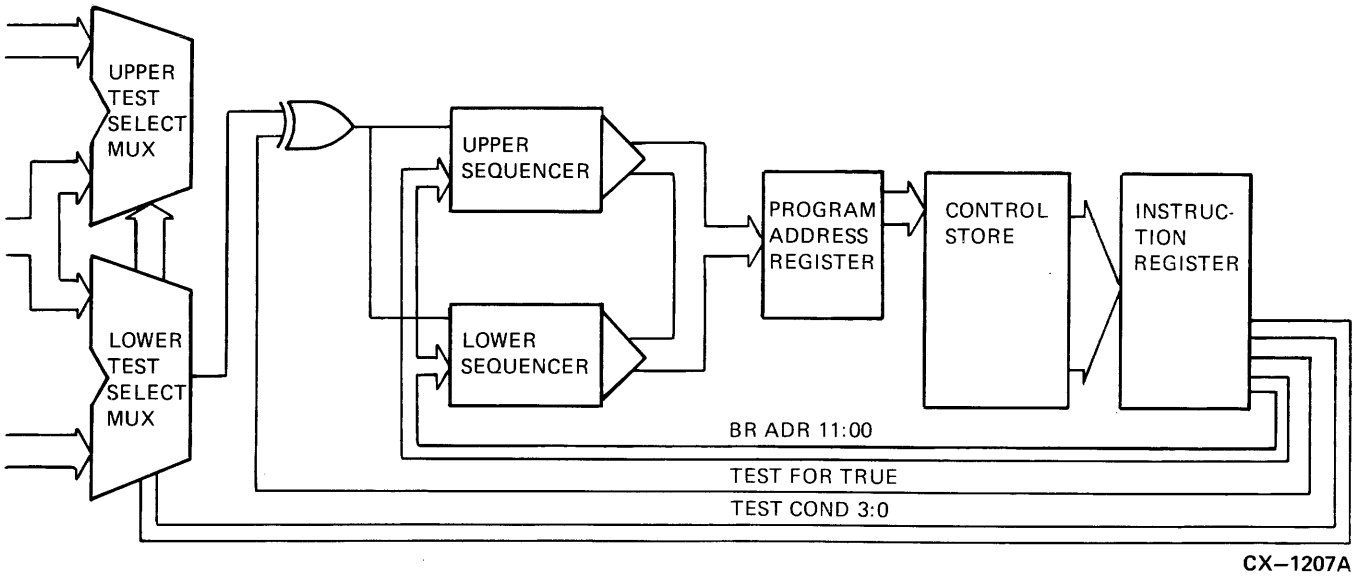
### 12.5.6 Pipelining Concepts

Now that you have read about the elements that make up the computing power on the K.sdi/K.sti, let's look at how the sequencers, Program Address Register, Control Store, and Instruction Register form a pipeline. Pipelined designs decrease instruction cycle time by performing fetch and execute operations in parallel (Figure 12-10). The Instruction Register holds the current microinstruction that controls the processing section and helps the control sections select the next address. Thus, the processing section is operating on the current microinstruction while the control section is fetching the next microinstruction from the Control Store.

### 12.5.7 K.sdi/K.sti Pipeline

The K.sdi/K.sti uses a form of microinstruction/address/status pipelining (double pipelining). Registers are provided for the current microinstruction (Instruction Register), the sequencer address (Program Address Register), and the ALU/real-time status (test multiplexers). Due to these three registers, the double pipelining design has three parallel paths. These paths simultaneously perform the following operations:

Figure 12-10 K.sdi/K.sti Double Pipelining

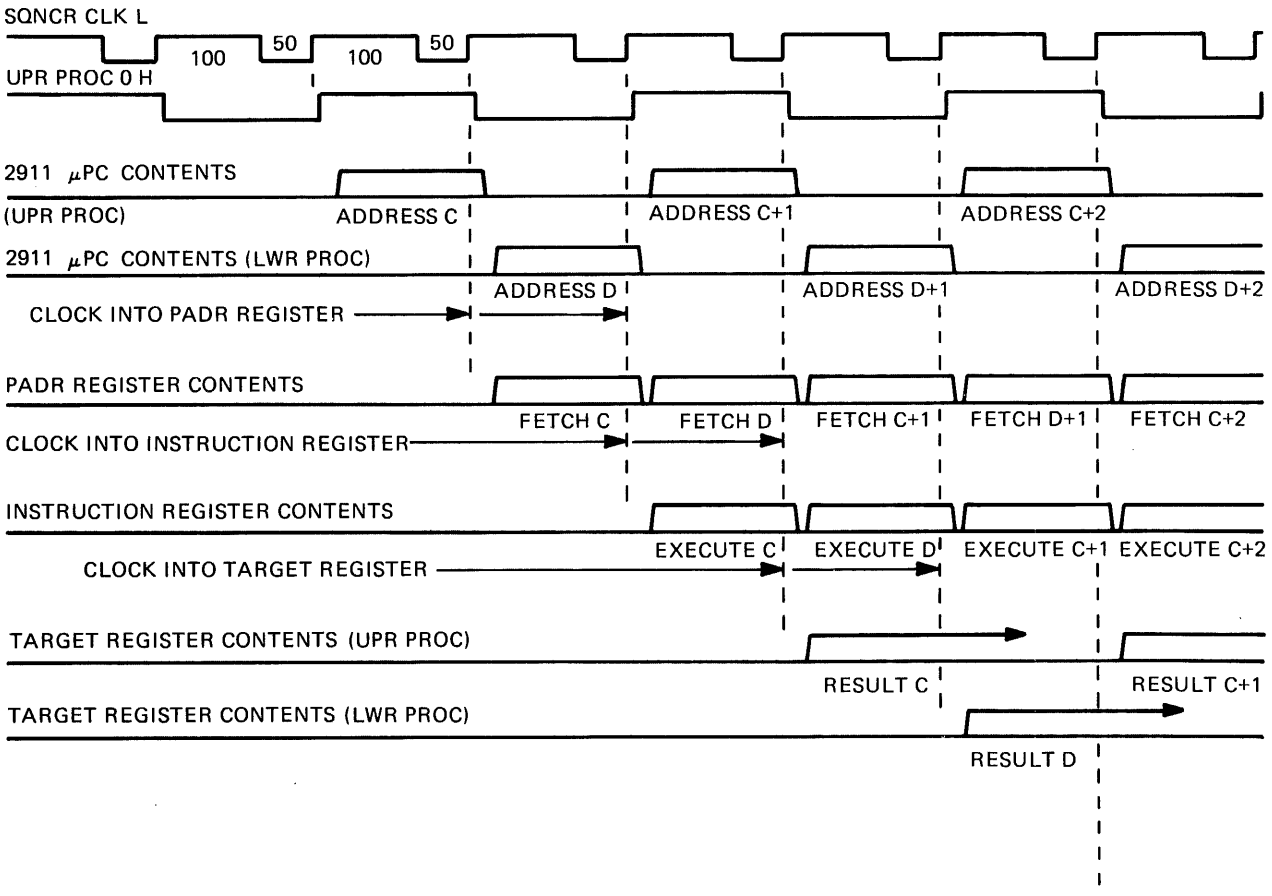


- The current microinstruction is clocked into the Instruction Register. It controls the ALU along with the branch address of the Upper and Lower Sequencers.
- The sequencers generate a new value for the Program Address Register.
- The current value in the Program Address Register fetches the next microinstruction for the Control Store. Control Store bits <31:28> select the test conditions for the test multiplexers.

### 12.5.8 Pipeline Timing

Figure 12-11 shows the timing for an instruction fetch-execute sequence. The SQNCR CLK supplies the timing for all the control timing. UPR PROC 1 is high when the Upper Sequencer is active. UPR PROC 1 is low when the Lower Sequencer is active. The timing in Figure 12-11 starts with the Upper Processor gating the address on SEQ ADR <11:00>. Then the following steps occur:

**Figure 12-11 Pipeline Timing for an Instruction Fetch-Execute Sequence**




CX–1234A

- On the high-to-low transition of the UPR PROC 0 signal:
  - The address from the Upper Sequencer is gated into the Program Address Register
  - The Control Store starts selecting the microinstruction for the Upper Sequencer
  - The Lower Sequencer gates its program address onto SEQ ADR <11:00>
- On the next high-to-low transition of the UPR PROC 0 signal:
  - The microinstruction word is clocked into the Instruction Register to be used by the Upper Sequencer
  - The Upper Sequencer gates its program address onto SEQ ADR <11:00>
  - The program address of the Lower Sequencer is gated into the Program Address Register

The steps listed above tend to indicate an order of operation. However, SQNCR CLK occurs at the same time on the sequencers, Program Address Register, and Instruction Register.

### 12.5.9 Scratchpad RAM

The K.sdi/K.sti has a Scratchpad RAM that has 16 memory locations of 16 bits. There is no parity protection. The two sequencers use the Scratchpad RAM:

- As a storage for variables
- As a communications mechanism between each other

Since there are only 16 words in Scratchpad RAM we only need to decode 4 bits. These 4 bits come from the Instruction Register and are:

IOC SEL 0
IOC SEL 1
IOC SEL 2
SET IOC

The signal to read from memory, RD SCRATCH REG, comes from the Bus Source Control logic. The signal to write to memory, WRT SCRATCH REG, comes from the Bus Destination Control logic.

### 12.5.10 Upper Control Register

The Upper Control Register (UPR CNTL REG) is shown just below the ALU in Figure 12-2. It is controlled by the Upper Sequencer. Figure 12-12 shows the bits of the register. Table 12-4 describes the bits in this register.

Figure 12-12 Upper Control Register

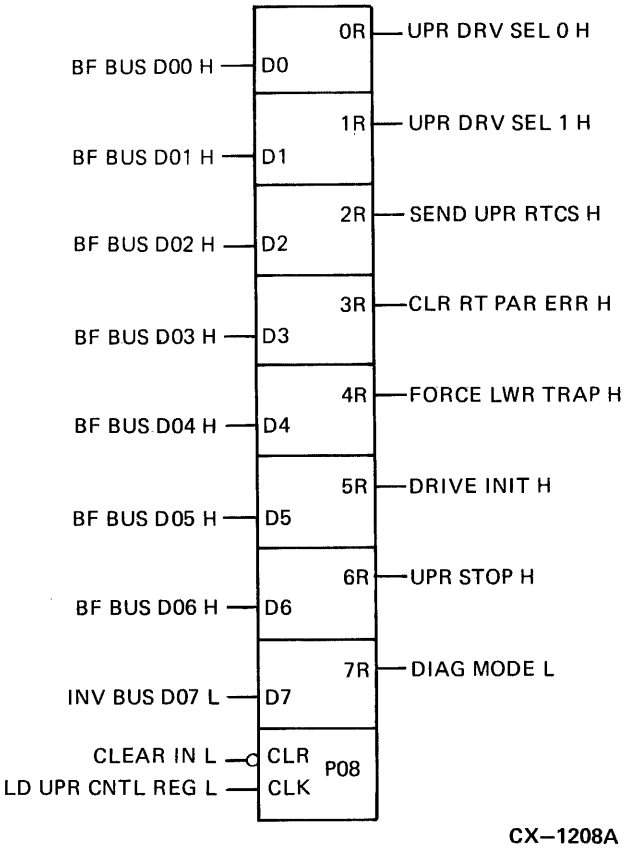


Table 12-4 Upper Control Register Bit Description

| Bit | Definition |
|---|---|
| UPR DRV SEL <1:0> | These two bits select the destination drive for the Control SERDES and real-time controller state for the Upper Sequencer. |
| SEND UPR RTCS | When set, this bit provides the sync bit for the Upper Real-Time Controller State Serializer (P12 D7). |
| CLR RT PAR ERR | When set, this bit clears the real-time drive state error in the Real-Time Drive State Deserializers. The UPR DRV SEL 0/1 bits select which drive deserializer is cleared. |
| FORCE LWR TRAP | When set, this bit forces the Lower Sequencer to address $3777_8$ (Section 12.5.3). |
| DRIVE INIT | When set, this bit asserts DRIVE INIT to the Real-Time Controller State Serializer. This bit initializes the drive presently selected by the the UPR DRV SEL 0/1. |
| UPR STOP | When set, this bit halts the K.sdi/K.sti immediately. This bit gives the microcode the ability to halt program execution due to the detection of a catastrophic error (an error which could possibly cause the K.sdi/K.sti to write bad data on the disk or tape). |
| DIAG MODE | When clear, this bit puts the K.sdi/K.sti in diagnostic mode. When in diagnostic mode the following K.sdi/K.sti hardware changes occur: The hardware has control of the real-time drive status bits and the bits capable of forcing errors on the Control and Data buses. It is capable of doing loop-back testing of the Control and Data SERDES. The Control Real-Time Control Serializer is capable of writing into the Real-Time Drive Deserializers. Control of the real-time drive status bits and the Control and Data bus error forcing ability is via the Diagnostic Register. (Refer to Section 12.5.12.) |

Notice in Figure 12-12 that the input to bit 7 (DIAG MODE L) is INV BUS D07 L. The INV BUS inverts the signal on BUS D07. Therefore, when the Control Sequencer writes a 1 on BUS D07, it is inverted and clears bit 7 of the Upper Control Register which puts the K.sdi/K.sti in diagnostic mode. Also, when the K.sdi/K.sti is initialized from the P.ioc, the register is cleared by CLEAR IN L which puts the K.sdi/K.sti in diagnostic mode.

### 12.5.11 Lower Control Register

The Lower Control Register (LWR CNTL REG) is shown just below the ALU in Figure 12-2. It is controlled by the Lower Sequencer. Figure 12-13 shows the bits of the register. Table 12-5 describes the bits in this register.

Figure 12-13 Lower Control Register

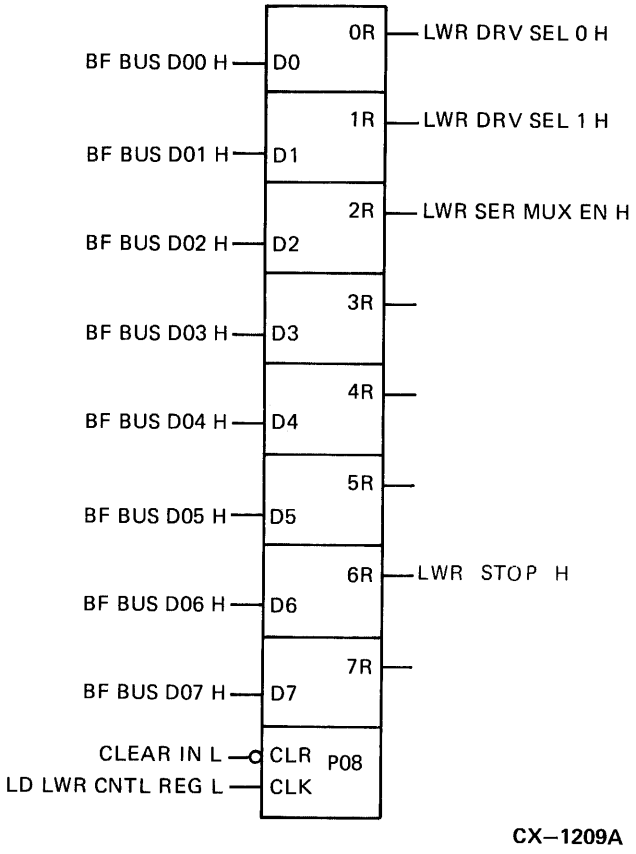


**Table 12-5 Lower Control Register Bit Description**

| Bit | Definition |
|---|---|
| LWR DRV SEL <1:0> | These two bits select the destination drive for the Data SERDES and real-time controller state for the Lower Sequencer. These two bits also select which drive is selected for input to the Data SERDES. |
| LWR SER MUX EN | This bit is set or cleared by the Lower Sequencer. When set, this bit enables WRT DATA <3:0> and RTC DATA <3:0> multiplexers. WRT DATA <3:0> and RTC DATA <3:0> are wired ORed for information from either the Upper Sequencer or Lower Sequencer. When cleared, the output of the multiplexer is disabled. |
| LWR STOP | When set, this bit halts the K.sdi/K.sti immediately. This bit gives the microcode the ability to halt program execution due to the detection of a catastrophic error (an error which could possibly cause the K.sdi/K.sti to write bad data on the disk or tape). |

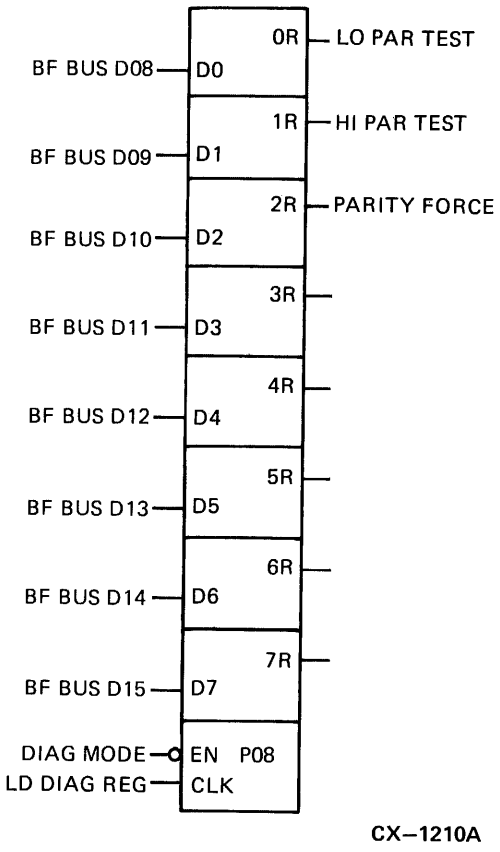
### 12.5.12 Diagnostic Register

The diagnostic register is an 8-bit register that allows the microcode to generate signals to test the parity check circuits during diagnostic mode. The Diagnostic Register is shown just above the Scratchpad RAM in Figure 12-2 and is writable by the Upper and Lower Sequencers. Figure 12-14 shows the bits in this register. Table 12-6 describes the bits in this register.

**Table 12-6 Diagnostic Register Bit Description**

| Bit | Description |
|---|---|
| LO PAR TEST | **C, D, E-rev etch**<br>Forces bad parity to be written into Data or Control Memory, depending on which is being written. This bit forces bad parity into the low byte only. When the system receives an initialize, the bit comes up asserted low forcing parity errors on the bus. To access the bus with no parity errors, this bit is set 1. This applies to diagnostic mode only.<br><br>**F-rev etch**<br>Any time this bit is a 1, bad parity is generated. |
| HI PAR TEST | Same as LO PAR TEST, only this bit controls the high byte. |
| PARITY FORCE | Forces bad parity to be generated across the output of the Control Store instruction word. |

**Figure 12-14 Diagnostic Register**



### 12.5.13 Control Error Register

Once the Upper Sequencer checks the Upper Test Multiplexer signal CNTL ERRSUM and finds it asserted, it then can read the Control Error Register and determine which error generated the CNTL ERRSUM. The following errors can assert CNTL ERRSUM:

CNTL NXM
C DATA PAR ERR
C BUS ERROR
RESP PLS MSNG
INSTR PAR ERR

The error register also includes some Control Bus status signals as shown in Figure 12-15. Table 12-7 describes the bits in this register. The Control Error Register is shown just above the Control Store in Figure 12-2.

**Figure 12-15 Control Error Register**

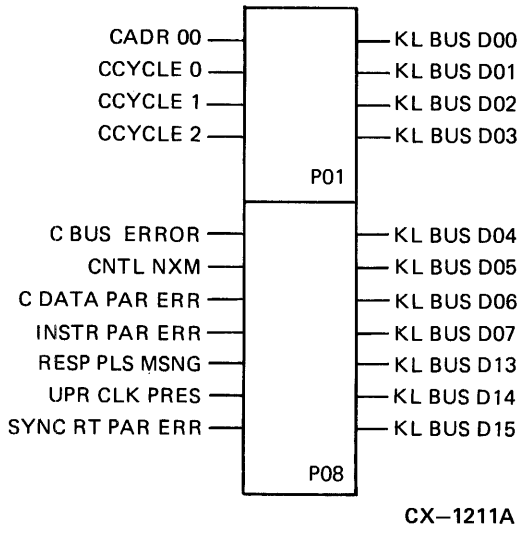


Table 12-7 Control Error Register Bit Description

| Bit | Description |
|---|---|
| CADR 00 | This bit is the least significant bit of the Control Bus Address used on the last Control Bus Cycle by this module. |
| CCYCLE <2:0> | These three bits define what type of cycle was performed on the last Control Bus Cycle by this module. |
| C BUS ERROR | This bit is the state of the CERR signal from the last Control Bus Cycle by this module. Once set, this bit remains set until cleared by reading this register. |
| CNTL NXM | This bit is the state of the CACK signal from the last Control Bus Cycle by this module. Once set, this bit remains set until cleared by reading this register. |
| C DATA PAR ERR | This bit indicates that there was a parity error on the Control Bus. Once set, this bit remains set until cleared by reading this register. |
| INSTR PAR ERR | This bit indicates that there was a parity error in the instruction word coming from the Control Store. |
| RESP PLS MSNG | This bit indicates that a transmission error from the drive occurred. This could be an extra pulse or a missing pulse over the SDI/STI connection. |
| UPR CLK PRES | This bit is not actually an error, but it is used to determine if a drive is in the off or offline state. In the offline state, clock transitions are present on the Real-Time Drive State line. In the off state or during drive initialization, clock transitions are not present. |
| SYNC RT PAR ERR | This bit indicates that a parity error or a transmission error occurred in the real-time state serializer/deserializers. |

### 12.5.14 Data Error Register

The Lower Sequencer reads the Data Error Register to determine if any errors occurred in the logic it controls. There is no ORing of data errors like the CNTL ERRSUM for the Upper Sequencer. The error register also includes some Data Bus status signals as shown in Figure 12-16. Table 12-8 describes the bits in this register. The Data Error Register is shown just above the Control Store in Figure 12-2.

Figure 12-16 Data Error Register

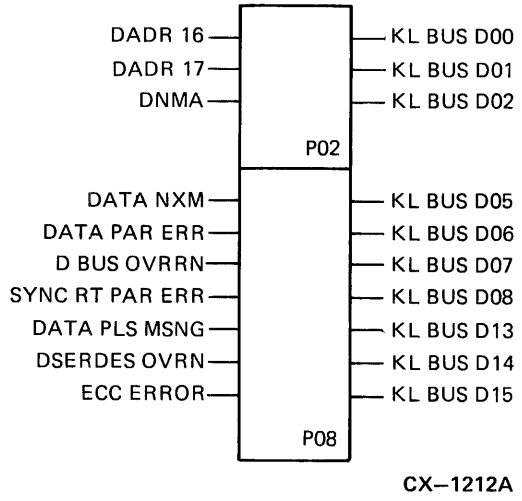



Table 12-8 Data Error Register Bit Description

| Bit | Description |
|---|---|
| DADR <17:16> | These two bits contain the two MSBs of the Data Bus Address used on the last Data Bus Cycle by this module. |
| DNMA | This bit reflects the state of the DNMA signal from the last Data Bus Cycle by this module. |
| DATA NXM | This bit is the state of the DACK signal from the last Data Bus Cycle by this module. Once set, this bit remains set until cleared by reading this register. |
| DATA PAR ERR | This bit indicates that there was a parity error on a previous Data Bus cycle by this module. Once set, this bit remains set until cleared by reading this register. |
| D BUS OVRRN | This bit indicates that a Data Bus Cycle was requested when the previously requested cycle had not yet completed. Once set, this bit remains set until cleared by reading this register. |

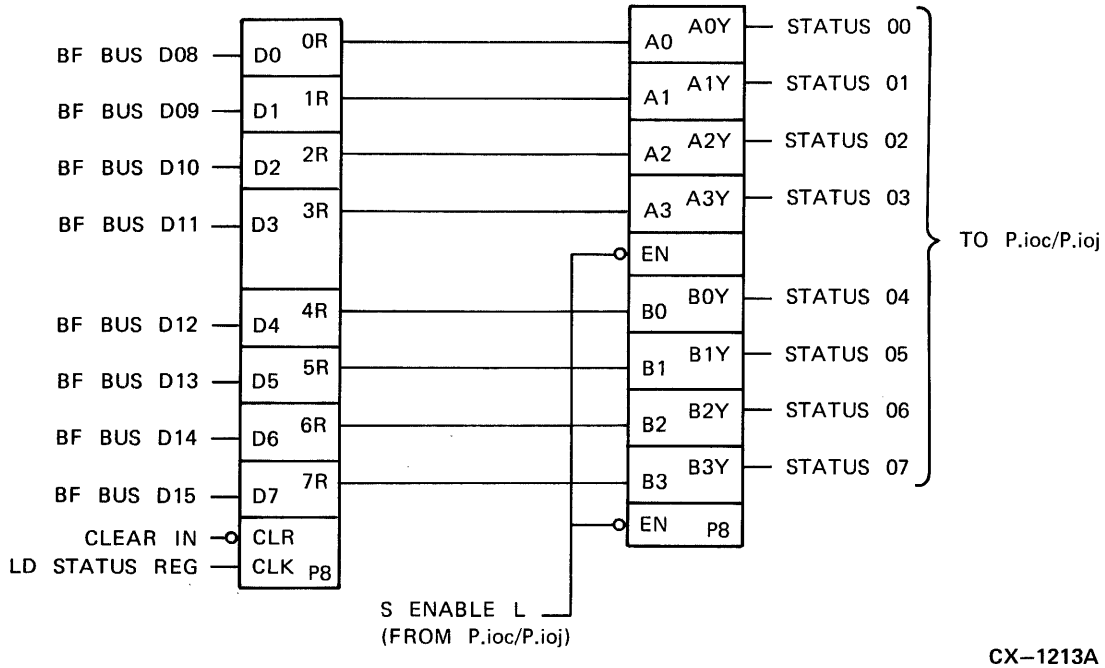
Table 12-8 (Cont.) Data Error Register Bit Description

| Bit | Description |
|---|---|
| SYNC RT PAR ERR | This bit indicates that a parity error or a transmission error occurred in the real-time state serializer/deserializers. |
| DATA PLS MSNG | This bit indicates that a data transmission error from the drive occurred. This could be an extra pulse or a missing pulse over the SDI/STI connection. |
| DSERDES OVRN | This bit indicates that there was a Data SERDES overrun. The Lower Sequencer did not transfer the data to or from the Data SERDES fast enough. |
| ECC ERROR | This bit indicates that there was an Error Correction Code (ECC) error on a data read operation. |

### 12.5.15 Status Register

The Status Register is gated onto the backplane by the P.ioc to check the status of the K.sdi/K.sti. The Status Register is shown in the lower-left-hand corner of Figure 12-2. The Status Bus Register is loaded by microcode from BF BUS D<15:08> (Figure 12-17).

Figure 12-17 Status Register



### 12.5.16 Control Bus Interface

The Control Bus Interface is associated with the Upper Sequencer. The Upper Sequencer uses this bus to access the HSC Control Memory which contains the control structures defining the work to be performed by K.sdi/K.sti. The Upper Sequencer scans predefined areas of Control Memory for work and manipulates the control structures. The K.sdi is capable of maintaining data rates up to 2.22 megabytes per second on this bus. For Control Bus timing information, refer to Chapter 3. The Control Bus Interface is shown in the lower-left-hand corner of Figure 12-2.

The interface is implemented with 2917A-type latched bus transceivers plus a separate byte parity generator which is shared with the Data Bus.

The Control Bus Address Transceivers are read or written by microcode. When a Control Memory write is performed, the hardware provides an automatic wrap around so that the contents may be read and verified. These transceivers are latches only. The Control Bus Address must be incremented by microcode and reloaded for each Control Memory access. The value obtained from reading the transceivers is the address used in the last completed bus cycle.

Notice in Figure 12-18 that the result obtained from reading the Control Bus Address does not have the same organization as required when loading it. The signals BUS D <15:00> go to CADR <16:01>. The signals BR ADR <11:08> go to CADR 00 and CCYCLE <2:0>. When the Upper sequencer reads the register, CADR 00 and CCYCLE <2:0> go to KL BUS <03:00>.

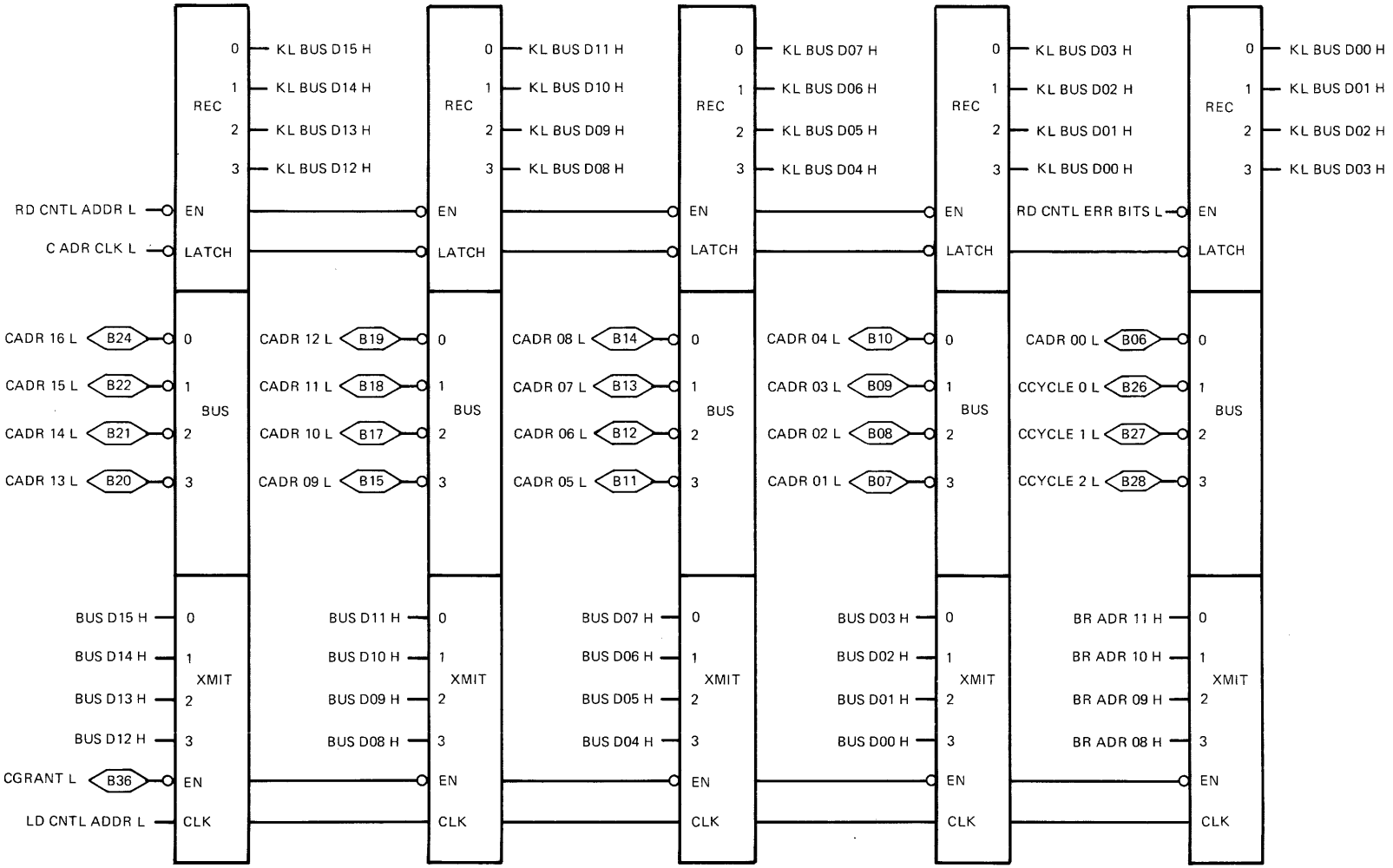
The Control Bus Data Transceivers may be read or written by microcode. When a Control Memory write is performed, the hardware provides automatic wrap around so that the contents may be read and verified. This read is organized exactly the same as the word that was gated to the data transceivers (including parity). The value obtained does not represent what is actually in memory as a result of the bus write (although this would be the typical result), but only what the K.sdi/K.sti transceivers placed on the backplane bus lines during the bus cycle.

Control Memory cycles are initiated when SET C REQ is specified in Control Store BUS DEST <2:0>. During execution of this instruction BR ADR <10:8> determine the type of bus cycle which will be performed, and BR ADR 11 controls the least significant address bit (Figure 12-18).

### 12.5.17 Data Bus Interface

This interface is similar to the Control Bus interface, except that it is associated with the Lower Sequencer. It is used to access the HSC Data Memory during data transfers. Data to be sent to the disk/tape is read from Data Memory, while data obtained from the disk/tape is written into Data Memory prior to being routed to the next process associated with the data transfer. The K.sdi/K.sti is capable of maintaining data rates up to 6.66 megabytes per second on this bus. For Data Bus timing information, refer to Chapter 3. The Data Bus Transceivers and Address Register are shown on the left side of Figure 12-2.

## Figure 12-18 Control Memory Address Register



### 12.5.17.1 Data Bus Data Transceivers

The Data Bus Data Transceivers are read or written by microcode, and the value obtained when reading the Data Transceivers has exactly the same format as required to write them. When a Data Memory write is performed, the hardware provides an automatic wrap around so that the data and address contents may be read and verified. The value obtained does not represent what is actually in memory as a result of the bus write (although this would be the typical result), but only what the transceivers placed on the backplane bus lines during the bus cycle.

Data Bus cycles are initiated by either writing (which initiates a write cycle) or reading (which initiates a read cycle) the Data Bus Data Transceivers.

The Data Bus Data Transceiver Drivers are controlled by the RD MODE signal from Lower IOC Control Latch bit 5. If RD MODE is clear, these drivers will not drive the bus. If RD MODE is set, these drivers will be gated onto the bus during each Data Bus cycle. This is true regardless whether a cycle is initiated by a load or a read of the data transceivers.

**NOTE**

**The term *read* is in reference to the disk/tape, not to Data Memory.**

### 12.5.17.2 Data Bus Address

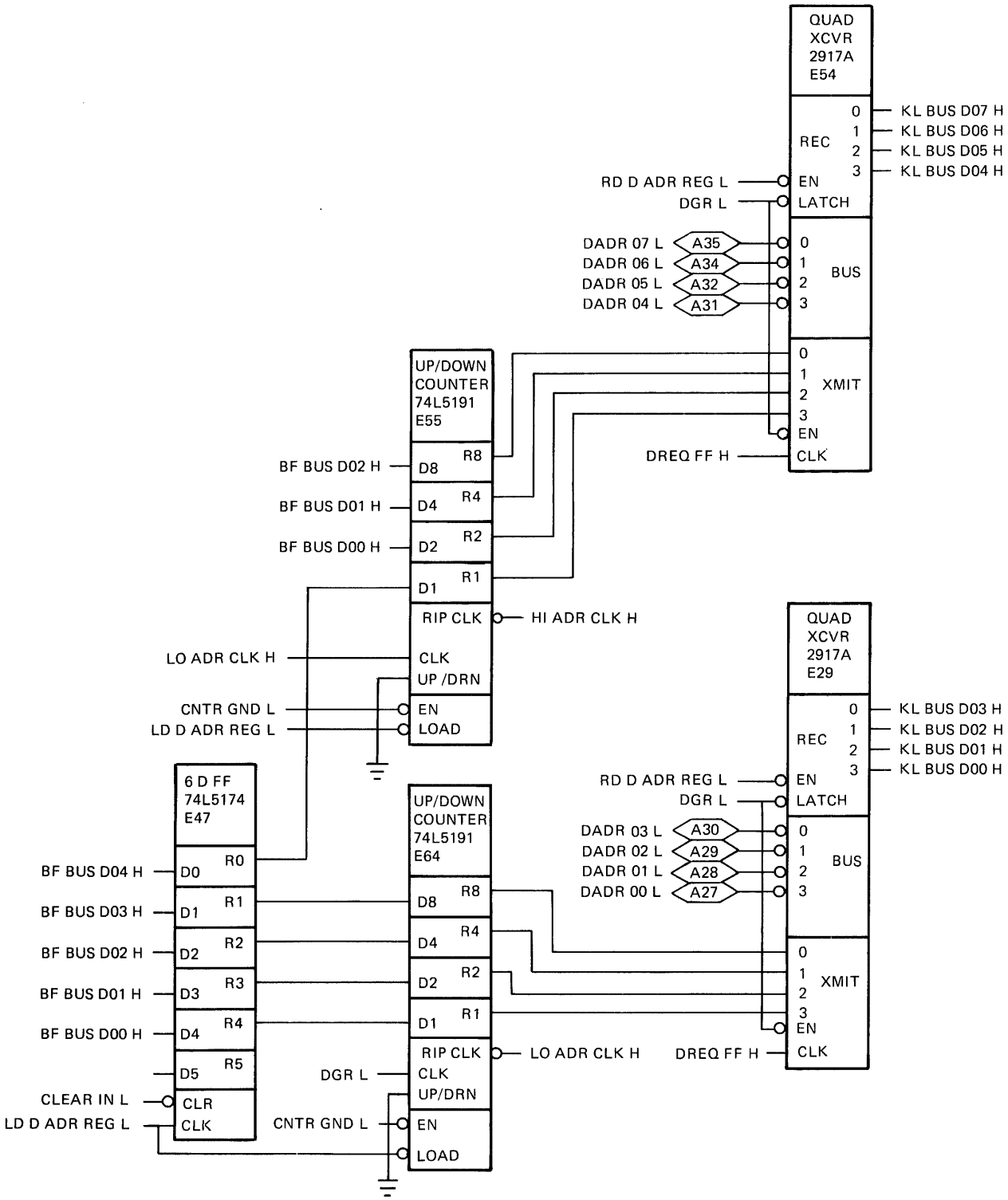
This interface consists of 2917A-type latched bus transceivers, but includes an up/down counter for the address. The Data Bus Address Transceivers and counter are loaded by microcode, and the transceivers may be read by microcode. Note that the result obtained from reading the Data Bus Address Transceivers does not have the same organization as required when loading it (Figure 12-20). The value obtained from reading the transceivers is the address used in the last completed bus cycle.

The Data Bus address is 18-bits wide. The 12 least significant address bits are derived from a counter which is automatically incremented (or decremented) at the end of each Data Bus cycle. Normally, the address is written prior to beginning a block transfer, and it is not rewritten while the block transfer is in progress. However, this counter will not cross 4 Kword boundaries.

Figure 12-19 shows a simplified diagram of how the Data Bus address is encoded. The microcode must assert the signal LD D ADR twice to generate the Data Bus address. On the first LD D ADR, BF BUS <4:0> is clocked into the 5 bit latch. On the second LD D ADR, the following registers bits are clocked:

- The 5-bit latch is clocked into counter bits <04:00>
- The BF BUS D<06:00> are clocked into the counter bits <11:05>
- The BF BUS D<12:07> are clocked into Data Bus Address Transceivers

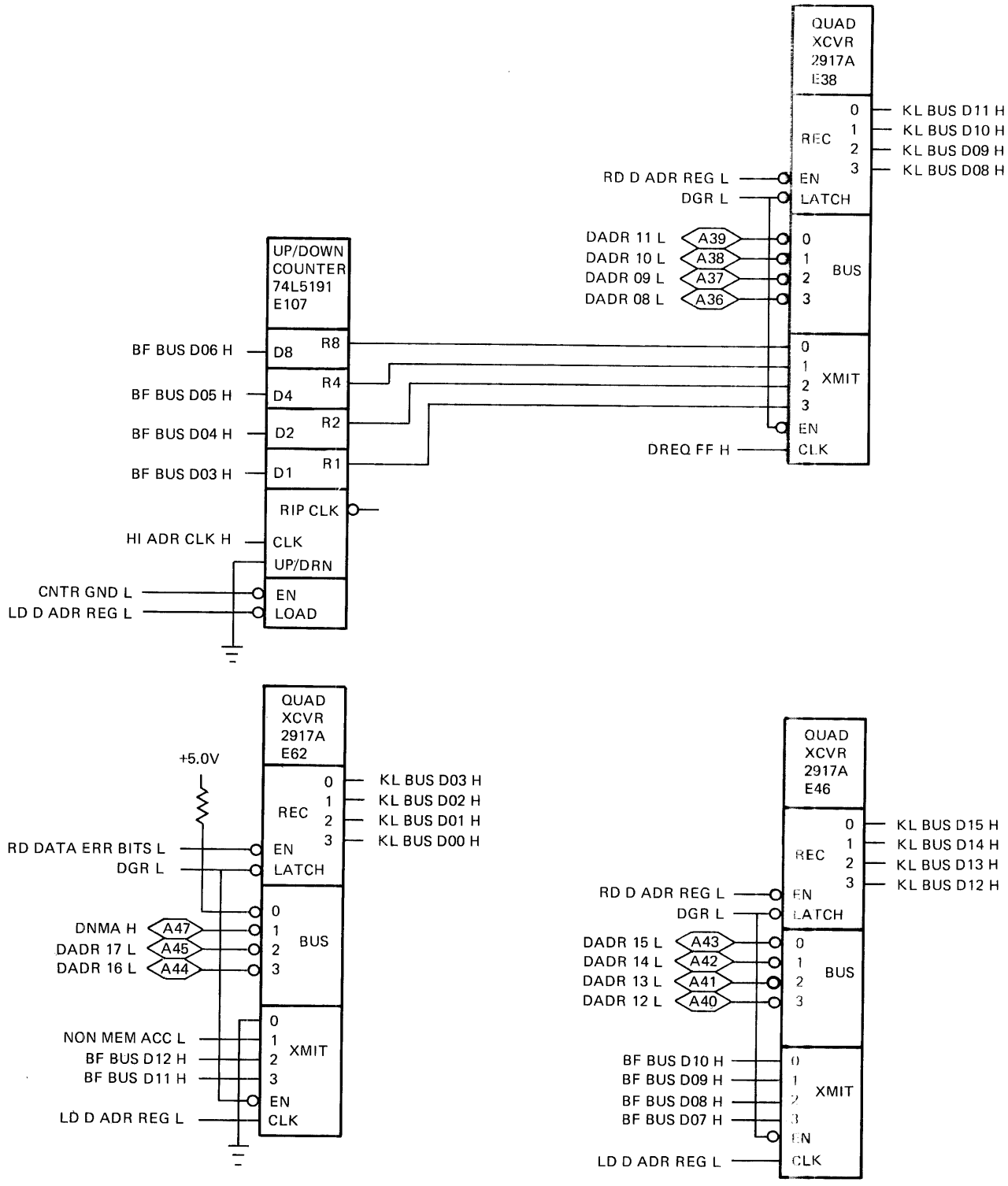
Figure 12-19 Data Bus Address Transceivers and Counter




CX–1215A
Sheet 1 of 2

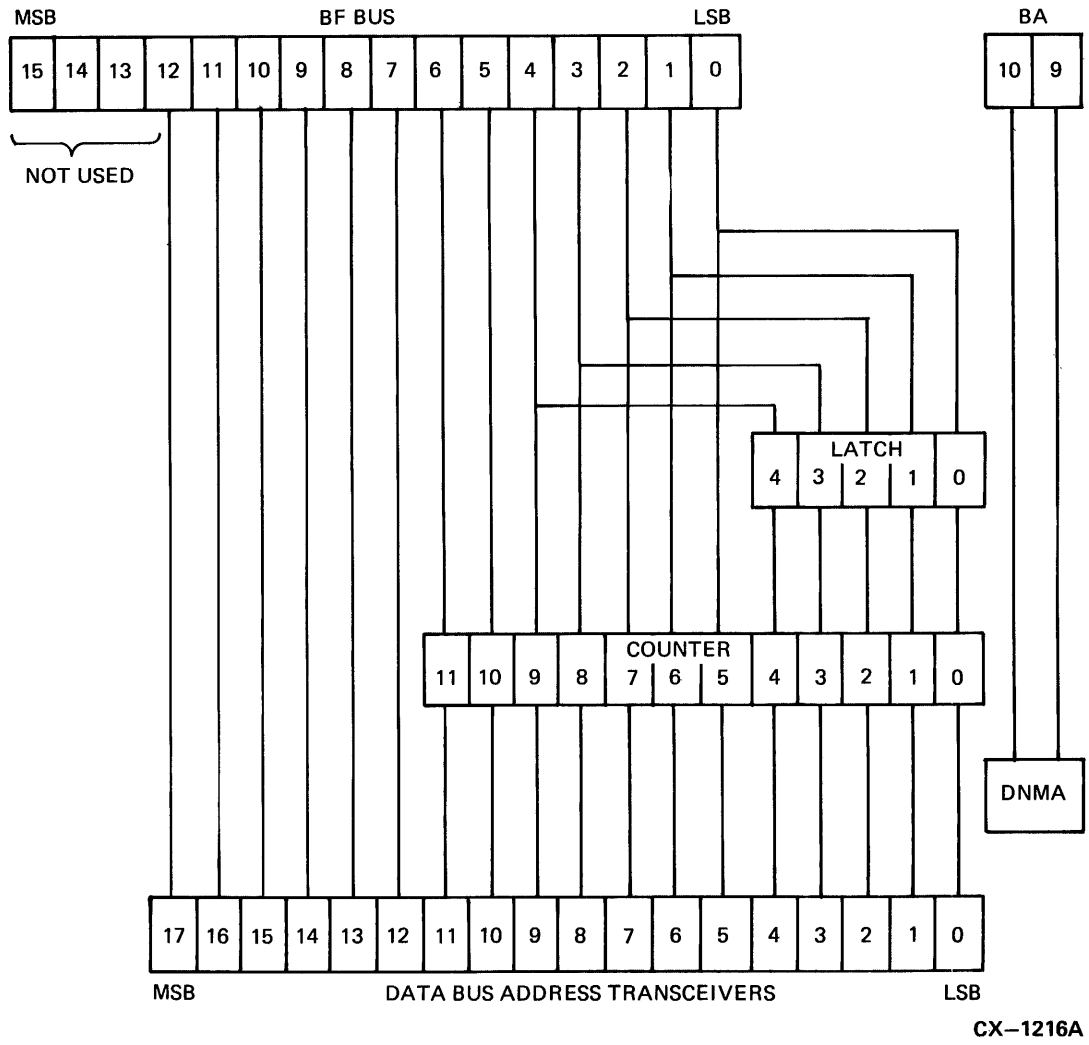
(Continued on next page)

Figure 12-19 (Cont.) Data Bus Address Transceivers and Counter




CX–1215A
Sheet 2 of 2

Figure 12-20 Generating the Data Bus Address



Note that counter bits <04:00> are derived from the latch, not directly from the BF BUS lines. In order to correctly write the entire address, LD D ADR REG must be executed twice. The first write places the desired contents of address <04:00> on BF BUS <04:00>. The second write places the desired contents of address <17:05> on BF BUS <12:00>.

The procedure to start a data transfer is:

1. Write the least significant five address bits
2. Write the remainder of the address
3. Perform one dummy data read (if it is a memory read operation)
4. Move a data word, conditioned on completion of previous word
5. Repeat the above step until the transfer is complete

### 12.5.18 Sector/Index PALs

The Sector/Index PALs latch the occurrence of the sector and index pulses so that microcode can deal with them via polling. The PALs are shown to the right of the Scratchpad RAM in Figure 12-2. The index <3:0> and sector <3:0> come from the real-time drive state deserializers shown in the lower-right hand corner of the block diagram. The PALs have 16 outputs which are 4 groups of 4. Figure 12-21 shows the Sector/Index Register, and Table 12-9 describes the bits in the register.

One input to the PALs is from two jumpers (W1 and W2). These two jumpers configure the Data Channel Module as an K.sdi or an K.sti.

**Figure 12-21 Sector/Index Register**

| IS D3 | IS D2 | IS D1 | IS D0 | ID 3 | ID 2 | ID 1 | ID 0 | IV 3 | IV 2 | IV 1 | IV 0 | SI S3 | SI S2 | SI S1 | SI S0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

CXO-1274A

**Table 12-9 Sector/Index Register Bit Description**

| Bits | Description |
|---|---|
| ISD<3:0> | Index or Sector pulse detected from the corresponding disk drive. The Control Sequencer uses these bits to keep track of the rotational position of each drive. When the Control Sequencer reads this register, any set bits are cleared. The PAL logic will not attempt to clear the sector or index bits if the bits are not set.<br><br>Not used for K.sti. |
| ID<3:0> | Index pulse detected from the corresponding disk drive. Similar to ISD<3:0>.<br><br>This bit is not used for an on-line Tape Formatter.<br><br>This bit is the latched low order bit of the event counter from an off-line Tape Formatter. The Upper Sequencer tests this bit and the SI bit for a state change in the Tape Formatter. |
| IV<3:0> | Index valid from the corresponding drive. These bits are set by the assertion of Index pulse and remain set until either sequencer issues a clear or until a Sector pulse is received and that drive's ISD bit is still set. For example, the sector counter for that drive is no longer valid because of a previously unrecognized Sector pulse.<br><br>Not used for K.sti. |

Table 12-9 (Cont.) Sector/Index Register Bit Description

| Bits | Description |
|---|---|
| SIS<3:0> | Synchronized Index or Sector pulse from the corresponding drive. The Data Sequencer uses these bits to detect sector boundaries.<br><br>This bit has the following two meanings in an K.sti:<br><br>Tape Formatter on line: A latch copy of the Acknowledge bit from the Tape Formatter. The Acknowledge bit indicates whether a data transfer completed successfully.<br><br>Tape Formatter off line: This bit is the latched second least significant low order bit of the event counter from an off-line Tape Formatter. The Upper Sequencer tests this bit and the ID bit for a state change in the Tape Formatter. |

## 12.6 INPUT AND OUTPUT TO THE STANDARD DISK/TAPE INTERCONNECT

The right side of Figure 12-2 shows the logic that the K.sdi/K.sti uses to communicate with the drive over the Standard Disk Interconnect (SDI) or the Standard Tape Interconnect (STI). The SDI/STI is shown on the upper-right of the block diagram. The K.sdi/K.sti has four ports and there are four hybrid receiver/driver circuits that drive the SDI/STI ports.

The K.sdi/K.sti performs seven different functions over the SDI/STI:

- Sends level 2 commands to a drive. (A level 2 command requires two or more bursts of information to transfer the command to a disk drive or a tape formatter. It also requires a response. An example of a level 2 command would be a seek command to a disk drive or a position tape command for a tape formatter.)
- Receives level 2 responses from a drive. (A level 2 response from a disk drive or tape formatter is two or more bursts of information in response to a level 2 command from the controller.)
- Sends level 1 commands to a drive. (A level 1 command is one burst of information to a disk drive or tape formatter. This command does not require a response. Examples of a level 1 command are select group for a disk drive and read for a tape formatter.)
- Sends data to to be written on a drive.
- Receives data read from a drive.
- Sends real-time controller state to a drive.
- Receives real-time drive state from a drive.

Instead of describing the individual blocks on the right-hand side of Figure 12-2, the following sections and figures describe the blocks as functions. The descriptions start with the top block and work toward the bottom.

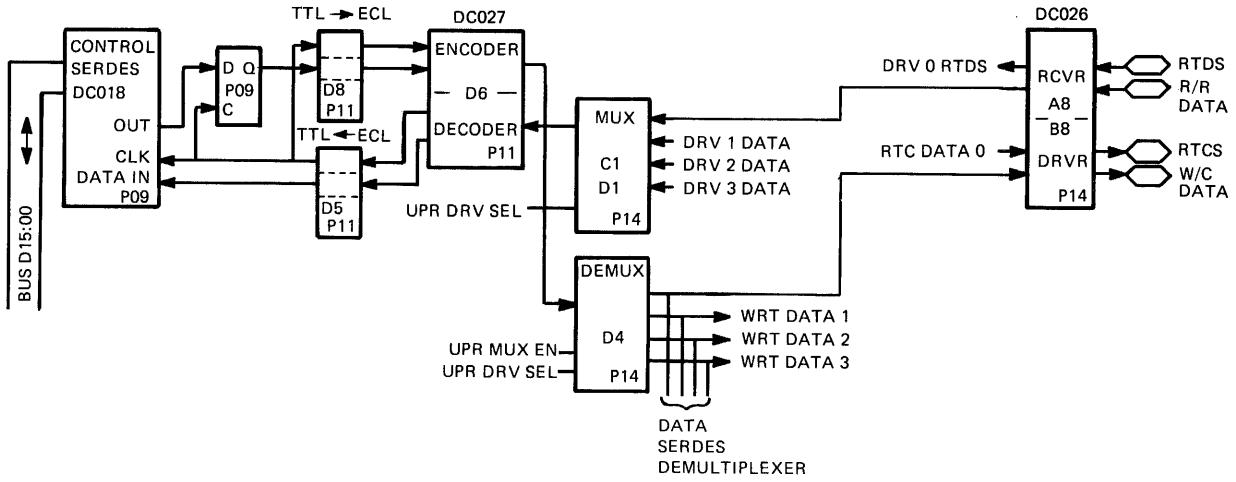
NOTE

**Any special chip used on the K.sdi/K.sti is described in detail in** *HSC50/70 Chip Descriptions Technical Manual* **(EK-HS574-TM).**

### 12.6.1 Send Level 2 Commands to a Drive

The Upper Sequencer can send a level 2 SDI/STI command to a drive. The Upper Sequencer gets the level 2 commands (INITIATE SEEK, GET STATUS, . . . ) from Control Memory and controls the logic that passes them from the Control Memory Transceivers to the Write/Command Data line on the SDI /STI. You have already read how the Upper Sequencer gets data from the Control Memory Transceivers. In this section, you will read how the Upper Sequencer controls the logic that turns the parallel data BUS D<15:00> into pulse-encoded information on the SDI/STI. The level 2 commands go from the Control SERDES to the Write/Command Data Line through the logic shown in Figure 12-22.

**Figure 12-22 Send Level 2 Commands to a Drive Block Diagram**




CX–1217A

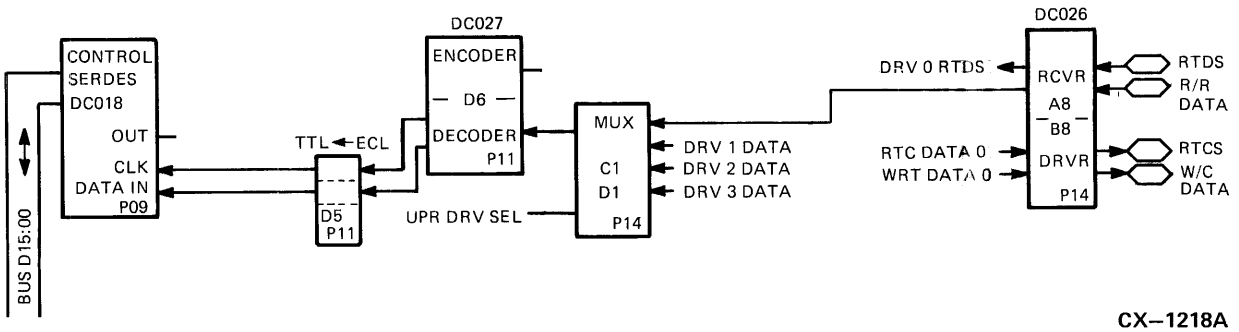
Each word of the command goes from BUS D<15:00> to the SDI/STI through the following logic:

1. The Control SERDES serializes the parallel information from BUS D<15:00>. The clock for the SERDES chip is derived from the Read/Response Data line on the SDI/STI.
2. The serial output of the SERDES is synchronized by a flip/flop to produce NRZ information.
3. This NRZ information is then converted from TTL to ECL before being sent to the encoder (D6 P11).
4. The encoder then encodes the NRZ information to the pulse encoding for the SDI/STI.
5. The output of the encoder is then sent to the demultiplexer (D4 P14).
6. The UPR DRV SEL signals, from the Upper Control Register, select which SDI/STI Driver the information is going to. The output of the demultiplexer is wire ORed with the output of the Data SERDES Demultiplexer. The signal UPR MUX EN, from the Upper IOC latch, selects this demultiplexer.
7. The output of the demultiplexer is sent to the SDI/STI Driver where it is then sent over the Write/Command Data line. In this block diagram the driver selected was (B8 P14) for port 0.

### 12.6.2 Receive Level 2 Responses from a Drive

The Upper Sequencer controls receiving the serial level 2 responses from the drive, converts it to parallel, and then transfers it to Control Memory. You have already read how the Upper Sequencer loads the address and data for Control Memory. In this section, you will read how the Upper Sequencer controls the logic that receives the serial data and turns it into parallel information for BUS D<15:00>. A level 2 response from the drive goes from the Read/Response line to BUSD<15:00> through the logic shown in Figure 12-23.

**Figure 12-23 Receive Level 2 Responses from a Drive Block Diagram**



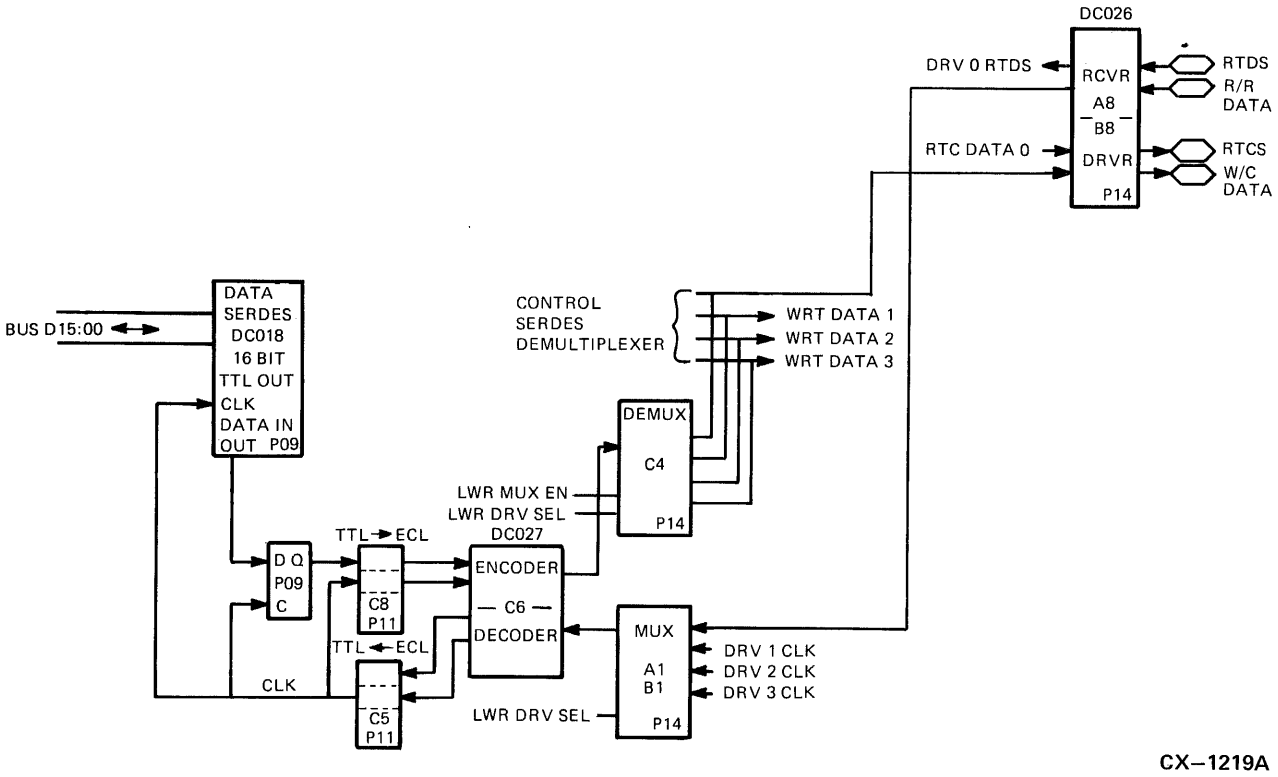

Each word of the response comes from the SDI/STI (right side of block diagram) to BUS D<15:00> through the following logic:

1. The response information comes into the SDI/STI Receiver (A8 P14) from the Read/Response Data (R/R DATA) line (port 0 in this block diagram).
2. The information from the SDI/STI Receiver (DRV x DATA) is one of four inputs to the multiplexer (C1 P14). The UPR DRV SEL signals from the Upper Control Register select which input to gate through the multiplexer.
3. The output from the multiplexer is an input to the decoder (D6 P11). The decoder decodes the pulse information and generates clock and data. This information is changed from ECL to TTL. The Control SERDES changes it from serial to parallel.
4. When the information is ready, the Upper Sequencer is signaled, and it then moves the information from the Control SERDES over BUS D<15:00> to the Control Bus.

### 12.6.3 Send Level 1 Commands to a Drive

The Lower Sequencer can send a level 1 SDI/STI command to a drive. The components of the path are the same type as those described in Section 12.6.1. The Lower Sequencer gets the level 1 SDI/STI command from Scratchpad RAM and controls the logic that passes it to the Write/Command Data line on the SDI/STI. In this section, you will read how the Lower Sequencer controls the logic that turns the parallel data BUS D<15:00> into pulse-encoded information on the SDI/STI. A level 1 command goes to the Write/Command Data line on the SDI/STI through the logic shown in Figure 12-24.

**Figure 12-24 Send Level 1 Commands to a Drive Block Diagram**



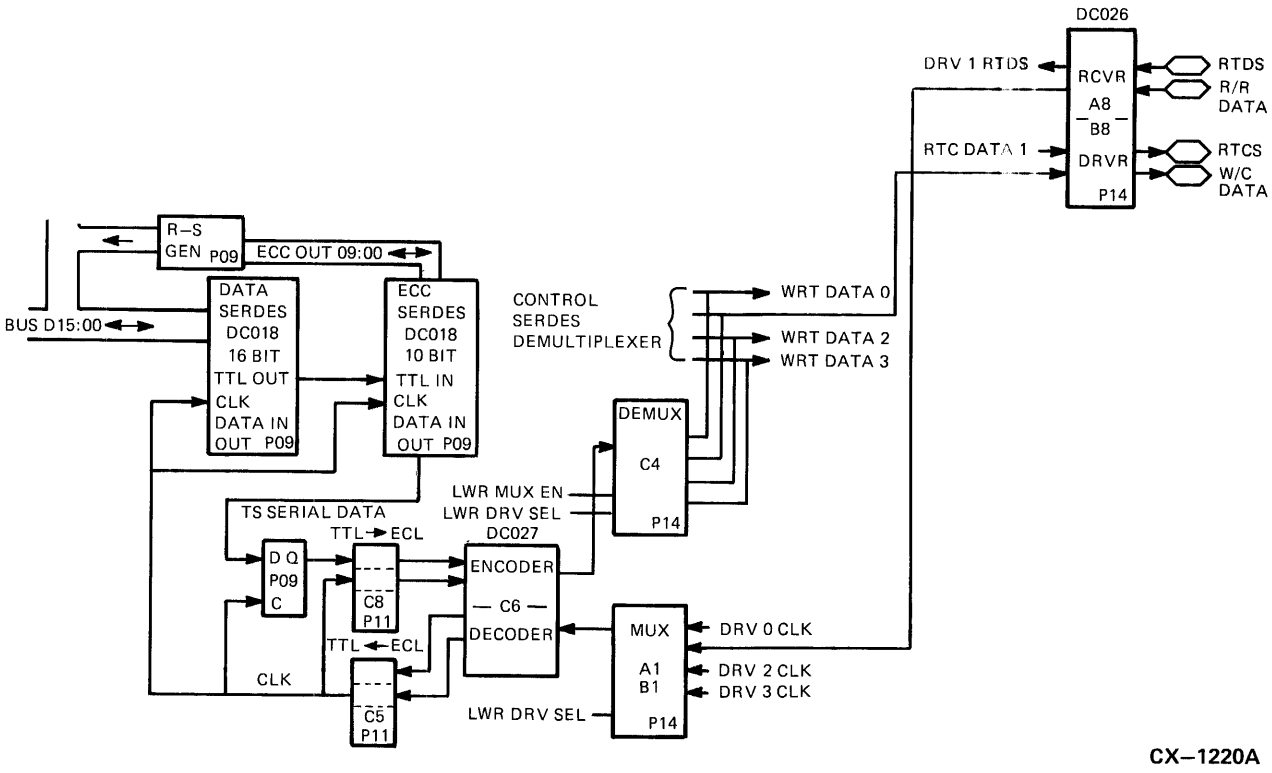

The level 1 SDI/STI command goes from BUS D<15:00> to the SDI/STI through the following logic:

1. The Data SERDES serializes the parallel information from BUS D<15:00>. The clock for the SERDES chip is derived from the Read/Response Data line on the SDI/STI.
2. The output of the SERDES is synchronized by a flip/flop.
3. This NRZ information is then converted from TTL to ECL before being sent to the encoder (C6 P11).
4. The encoder encodes the NRZ information to the pulse encoding for the SDI/STI.
5. The output of the encoder is sent to the demultiplexer (C4 P14).
6. The LWR DRV SEL signals, from the Lower Control Register, select which SDI/STI Driver the information is going to. The output of the demultiplexer is wire ORed with the output of the Control SERDES Demultiplexer. The signal LWR MUX EN, from the Lower Control Register, selects this demultiplexer.
7. The output of the demultiplexer is sent to the SDI/STI Driver where it is then sent out over the Write/Command Data line. In this block diagram the driver selected was (B8 P14) for port 0.

### 12.6.4 Send Data to a Drive

The Lower Sequencer is the only sequencer that can send data to a drive. It uses the Data SERDES. However, the data must also pass through the ECC SERDES (Figure 12-25). The Lower Sequencer gets the data from Data Memory and moves it from the Data Bus Transceivers to the Data SERDES over BUS D<15:00>.

**Figure 12-25 Send Data to a Drive Block Diagram**




The data goes from BUS D<15:00> to the SDI/STI through the following logic:

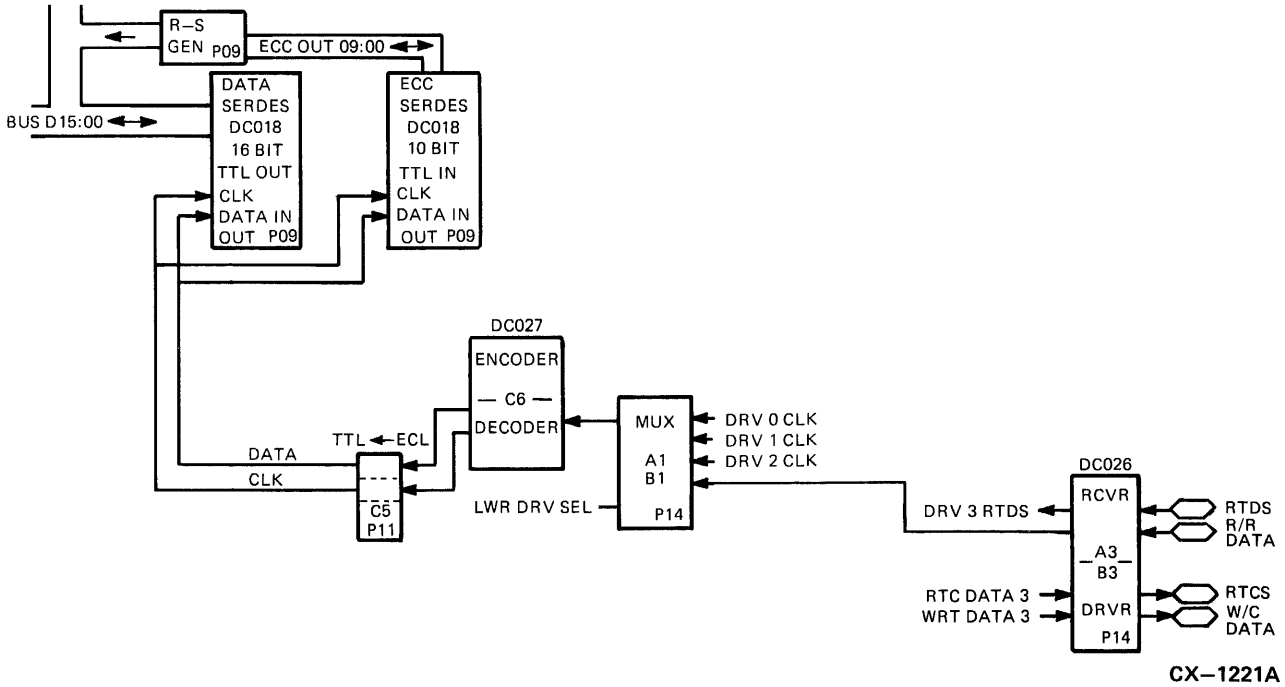
- The Data SERDES serializes the parallel data from BUS D<15:00>. The clock for the SERDES chips is derived from the Read/Response Data line on the SDI/STI. This time the data goes out the TTL OUT line to the ECC SERDES.
- The ECC SERDES does two things:
  - Converts the serial data to 10-bit parallel data and writes it to the Reed Solomon ECC Generator (R-S Gen)
  - Synchronizes the serial data with a flip/flop
- This NRZ data is converted from TTL to ECL before being sent to the encoder (C6 P11).
- The encoder encodes the NRZ data to the pulse encoding for the SDI/STI.
- The output of the encoder is sent to the demultiplexer (C4 P14).

- The LWR DRV SEL signals, from the Lower Control Register, select which SDI/STI Driver the information is going to. The output of the demultiplexer is wire ORed with the output of the Control SERDES Demultiplexer. The signal LWR MUX EN, from the Lower Control Register, selects this demultiplexer.
- The output of the demultiplexer is sent to the SDI/STI Driver where it is then sent out over the Write/Command Data line. In this block diagram the driver selected was (B6 P14) for port 1.
- The above sequence is repeated until the end of the sector or record. Then the Lower Sequencer puts the ECC SERDES in parallel to serial mode. It then starts the R-S Generator sending its 170 bit ECC calculation to the ECC SERDES. The R-S Generator sends 10-bit symbols, one at a time, to the ECC SERDES. The ECC SERDES converts the parallel 10-bit ECC symbols to serial and sends them right behind the data to the drive.

### 12.6.5 Receive Data from a Drive

The Lower Sequencer controls receiving the serial read data from the drive, converts it to parallel, and then transfers it to Data Memory. You have already read how the Lower Sequencer loads the address and data for Data Memory. In this section, you will read how the Lower Sequencer controls the logic that receives the serial data and turns it into parallel data for BUS D<15:00>. Figure 12-26 shows the logic that receives the data, decodes it, and converts it to parallel.

**Figure 12-26 Receive Data from a Drive Block Diagram**



The data goes from the SDI/STI (right side of block diagram) to BUS D <15:00> through the following logic:

1. The read data comes into the SDI/STI Receiver (A3 P14) from the Read/Response Data (R/R DATA) line (port 3 in this block diagram).
2. The data from the SDI/STI Receiver (DRV x DATA) is one of four inputs to the multiplexer (A1/B1 P14). The LWR DRV SEL signals from the Lower Control Register select which input to gate through the multiplexer.
3. The output from the multiplexer is an input to the decoder (C6 P11). The decoder decodes the pulse information and generates clock and serial data. This information is changed from ECL to TTL.
4. The serial data is then clocked into the Data SERDES and ECC SERDES at the same time.
5. The Data SERDES converts the serial data to 16-bit parallel data and signals the Lower Sequencer that a word of data is ready. The Lower Sequencer moves the data to the Data Bus Transceivers.
6. As the Data SERDES is converting, the ECC SERDES is converting the serial data to 10-bit parallel symbols for the R-S Generator.
7. At the completion of the transfer, the R-S Generator checks to see if it has a zero ECC. If the ECC is zero, it is correct. If the ECC is not zero, the ECC ERROR bit is set in the Data Error Register.

### 12.6.6 Send Real-Time Controller State to a Drive

The lower-right side of Figure 12-2 is where the logic converts the real-time controller state to serial information and sends it over the SDI/STI. The Upper Sequencer or Lower Sequencer sends real-time controller state to the drive. They cannot both send the state at the same time. Figures 12-27 and 12-28 show the inputs to the Real-Time State Serializers for the K.sdi and K.sti and the Real-Time Controller State line for the given input.

Figure 12-27 K.sdi Real-Time Controller State

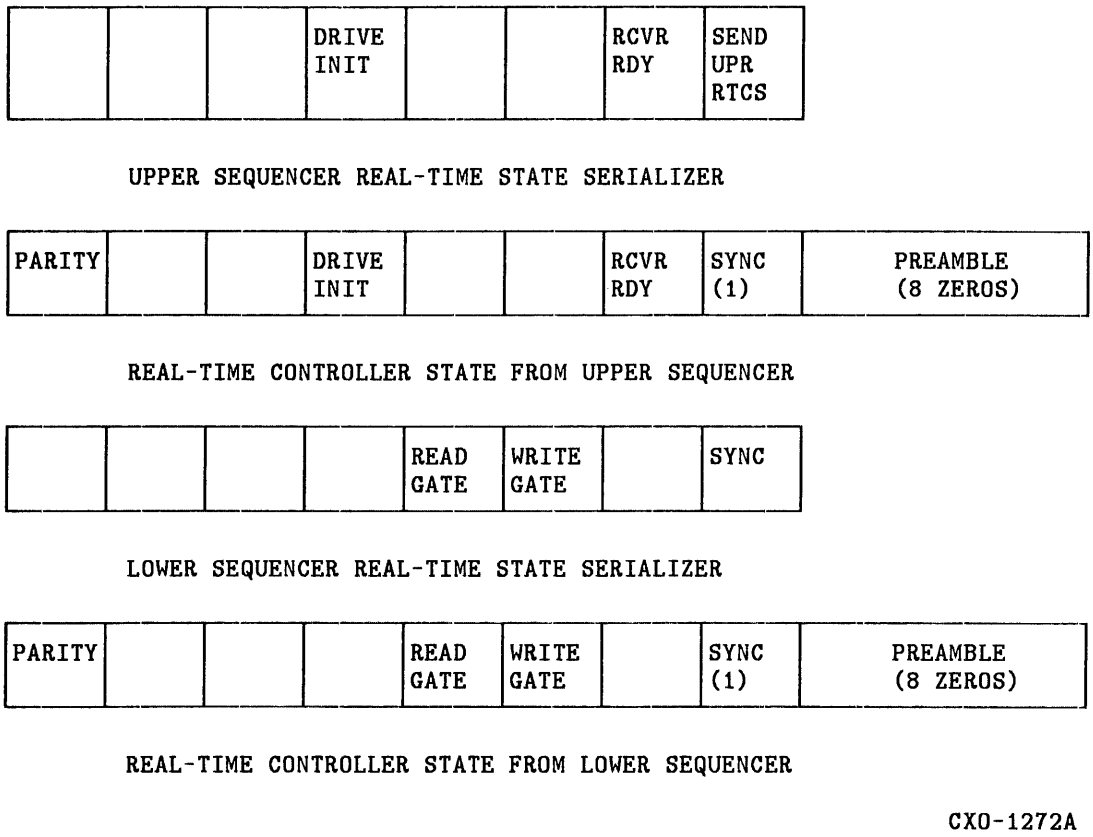
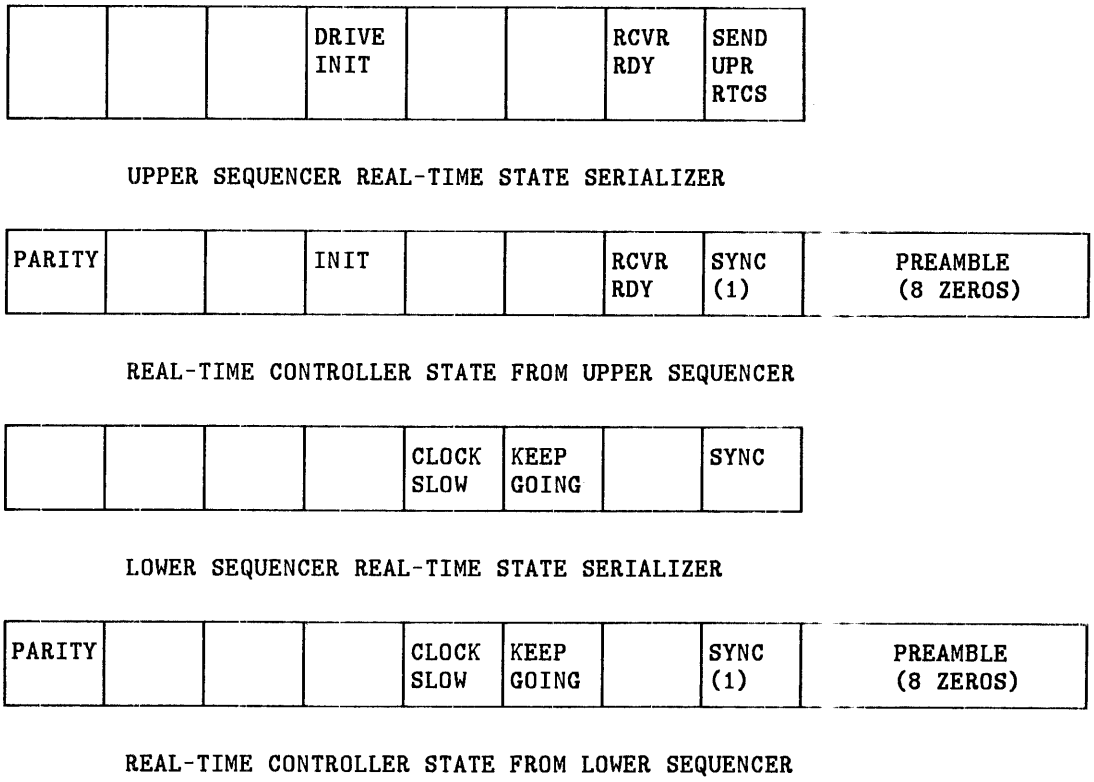


Figure 12-28 K.sti Real-Time Controller State




The Upper Sequencer sends two signals to the drive:

- RCVR RDY lets the drive know when it can send a response or attention message
- DRIVE INIT initializes the drive

The Lower Sequencer sends two signals to a disk drive:

- WRITE GATE asserts logic in the drive to do a write data operation
- READ GATE asserts logic in the drive to read data back to the K.sdi/K.sti

The Lower Sequencer sends two signals to a tape drive:

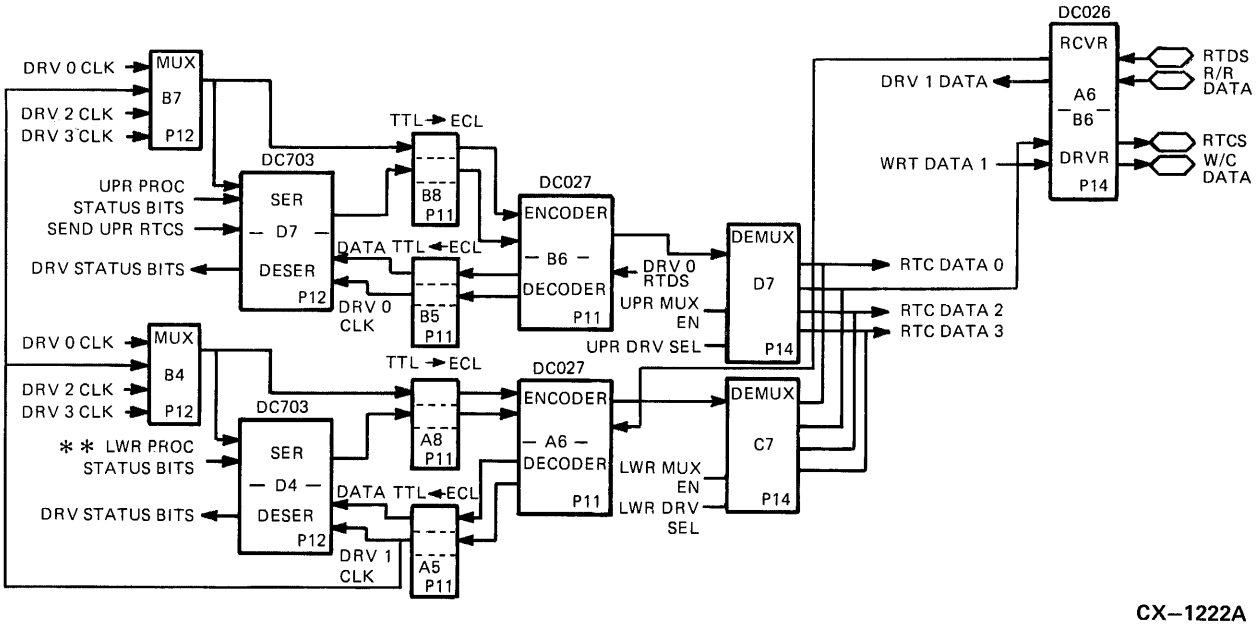
- KEEP GOING optimizes the performance of streaming tape drives.
- CLOCK SLOW indicates to the formatter that the K.sti is temporarily unable to support an effective clock rate in excess of 8.8 MHz.

Figure 12-29 shows the logic to send real-time controller status to a drive. There is an Upper Sequencer Real-Time Controller State Serializer (D7 P12) and a Lower Sequencer Real-Time Controller State Serializer (D4 P12).

**NOTE**

**The Real-Time Controller State serializers send the state when the least significant bit is set as an input to the serializers. This bit is also the SYNC bit on the Real-Time Controller State line.**

**Figure 12-29 Send Real-Time Controller State to a Drive Block Diagram**




CX–1222A

The real-time controller state for the Upper Sequencer goes through the following logic:

- The clock for the serializer (D7 P12) comes from the Real-Time Drive State line on the SDI/STI.
- The UPR PROC STATUS BITS to the serializer (D7 P12) come from IOC Control logic (RCVR RDY) and Upper Control Register (DRIVE INIT).

- When the Upper Sequencer sets the SEND UPR RTCS bit in the Upper Control Register, the serializer starts sending the real-time controller state until the bit is cleared. This bit is the least significant bit of the serializer input.
- The output of the serializer is changed from TTL to ECL and then pulse encoded (B6 P11).
- The multiplexer (D7 P14) is selected by UPR MUX EN from the IOC Control Logic. The port driver is selected by UPR DRV SEL bits from the Upper Control Register (for this example, port 1). Notice that the output of the multiplexer is wire ORed with the output from the Lower Sequencer's multiplexer (C7 P14). Only one multiplexer can be enabled at a time.
- The selected drive (B6 P14) then sends the status over the Real-Time Controller Status (RTCS) line on the SDI/STI.

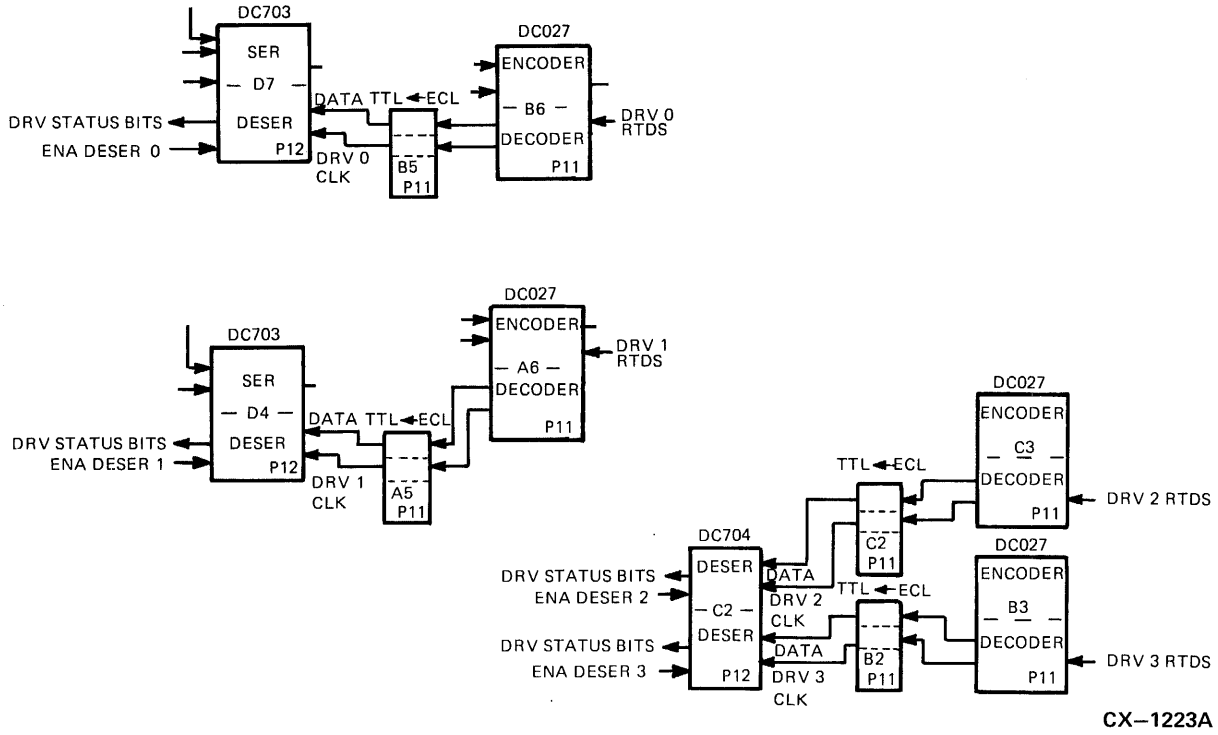
The Lower Sequencer does not set bits in a register to enable the Lower Real-Time Controller State Serializer (D4 P12). The status bits along with SYNC are generated through the ALU by the LITERAL field and then written into the serializer.

### 12.6.7 Receiving Real-Time Drive State from a Drive

The drives send their real-time drive state over the Real-Time Drive State (RTDS) line to the port receivers shown on the right side of Figure 12-2. The output of the receivers is DRV x RTDS. Figure 12-30 shows the logic for the four ports receiving real-time drive state:

- The real-time drive state comes into the receivers (not shown) whose output it DRV x RTDS.
- The decoders (B6,A6,C3, and B3 P11) change the pulse encoded SDI/STI state into NRZ. Then the signal level is changed from ECL to TTL.
- The state is then deserialized. There are two types of deserializers. The deserializers for port 0 and 1 are also serializers for real-time controller state which you saw in Section 12.6.6. Ports 2 and 3 are in one chip (C2 P12).
- The Upper Sequencer or Lower Sequencer can read the DRV STATUS BITS for a particular port by asserting ENA DESER x.
- Some of the DRV STATUS BITS are inputs to the test multiplexer.

Figure 12-30 Receiving Real-Time Drive State from a Drive Block Diagram



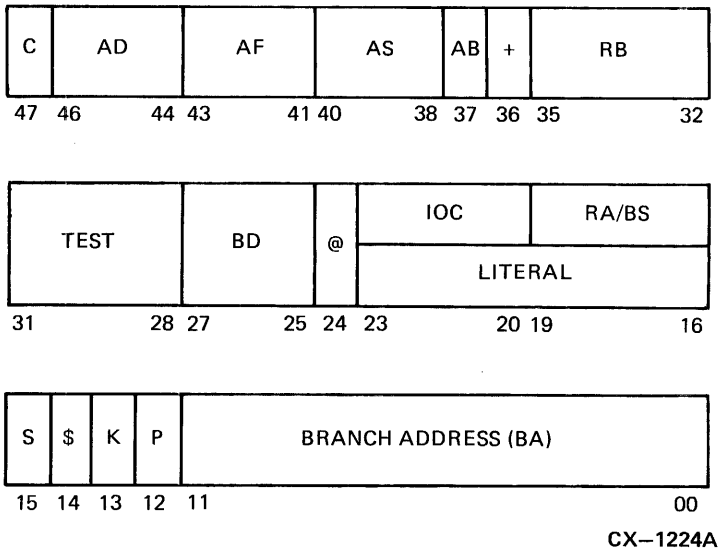

## 12.7 MICROINSTRUCTION

This section describes the organization and purposes of the fields within the microinstruction word. The microinstruction is 48-bits wide. The microinstruction word is divided into 16 fields, each of which controls different portions of the K.sdi/K.sti hardware. Figure 12-31 shows the fields and their position within the microinstruction word. The following sections describe each of these fields.

### 12.7.1 C Field

The C field is the carry-in into the ALU for the current microinstruction. It steers the literal to the most significant byte for AF fields 3 through 7.

Figure 12-31 Microinstruction Word Description




### 12.7.2 AD Field

The AD field, ALU destination, determines the routing of the ALU output data internal to the 2901. Table 12-10 describes the decoding of the three bits in the AD field.

Table 12-10 Decoding Bits in the AD Field

| Decode | Description |
|---|---|
| 0 | ALU to Q Register |
| 1 | No write |
| 2 | ALU to RB, output RA (RB if AS = 5 or 7) |
| 3 | ALU to RB |
| 4 | ALU/2 to RB, Q/2 to Q (right shift) |
| 5 | ALU/2 to RB (right shift) |
| 6 | ALU*2 to RB, Q*2 to Q (left shift) |
| 7 | ALU*2 to RB (left shift) |

### 12.7.3 AF Field

The AF field, ALU Function, determines the operation to be performed by the ALU. Table 12-11 describes the decoding of the three bits in the AF field. The R and S are the inputs to the ALU (Figure 12-9).

**Table 12-11 Decoding Bits in the AF Field**

| Decode | Mnemonic | ALU Function |
|---|---|---|
| 0 | ADD | R plus S |
| 1 | SUBR | S minus R |
| 2 | SUBS | R minus S |
| 3 | OR | R OR S |
| 4 | AND | R AND S |
| 5 | NOTRS | (NOT R) AND S |
| 6 | XOR | R XOR S |
| 7 | XNOR | R EX-NOR S |

### 12.7.4 AS Field

The AS field, ALU source, selects two of the five possible sources for the ALU. Table 12-12 shows the combination of the three bits in the AS field. The R, S, A, D, B, 0, and Q are the inputs and outputs of the ALU Data Source Selector in the ALU (Figure 12-9).

**Table 12-12 Decoding Bits in the AS Field**

| Decode | R | S |
|---|---|---|
| 0 | A | Q |
| 1 | A | B |
| 2 | 0 | Q |
| 3 | 0 | B |
| 4 | 0 | A |
| 5 | D | A |
| 6 | D | Q |
| 7 | D | 0 |

### 12.7.5 AB Field

The AB field, ALU to BUS, determines whether ALU or the bus source decoder is the source for the BUS. Table 12-13 describes the decoding of the bit in the AB Field.

**Table 12-13 Decode the Bit in the AB Field**

| Decode | Description |
|---|---|
| 0 | Bus source to BUS D<15:00>, D input to ALU is BUS D<15:00> |
| 1 | ALU to BUS D<15:00>, D input to ALU is the LITERAL field |

### 12.7.6 + Field

The + field, EXEC IF COND TRUE, is one bit long. This bit is combined with the following signals to generate INHIBIT EXEC (inhibit execute).

- TST COND TR (test condition true)
- COND EXC (conditional execute (Refer to Section 12.7.15))

INHIBIT EXEC prevents:

- Any ALU operation of each instruction for which it is active
- The IOC Control latches from being enabled
- The destination decoders from being enabled

Table 12-14 describes the decoding of the bit in the + field along with EXEC IF COND TRUE and TST COND TR to generate INHIBIT EXEC.

**Table 12-14 Decode the Bit in the + Field**

| COND EXC | EXEC IF COND TRUE | TST COND TR | INHIBIT EXEC | CONDITION |
|---|---|---|---|---|
| 0 | X | X | 0 | Execute all operations |
| 1 | 0 | 0 | 0 | Execute all operations |
| 1 | 0 | 1 | 1 | Stop ALU, IOC, destination operations |
| 1 | 1 | 0 | 1 | Stop ALU, IOC, destination operations |
| 1 | 1 | 1 | 0 | Execute all operations |

X = the state of the signal does not matter

### 12.7.7 RB Field

The RB field, B Port Register, selects one of the sixteen internal registers to be read and/or written by the 2901 ALU. If a value of $5_8$ or $7_8$ is in the AS field, the A Port Register field (RA field) is forced to the same value as the RB field.

### 12.7.8 TEST Field

The TEST field, TEST CONDITION <03:00>, selects one of 16 signals for the conditional operations of the sequencers. The hardware includes two separate selectors, one for each of the two sequencers. Table 12-15 shows the test conditions for the four bits of the TEST field. Tables 12-16 and 12-17 describe what each bit is.

**Table 12-15 Decode the Bits in the TEST Field**

| Decode | Upper Sequencer | Lower Sequencer |
|---|---|---|
| 0 | 1 (True) always | 1 (True) always |
| 1 | Last Upper Sequencer ALU result = 0 | Last Lower Sequencer ALU result = 0 |
| 2 | Last Upper Sequencer ALU carry out | Last Lower Sequencer ALU carry out |
| 3 | Last Upper Sequencer ALU MSB | Last Lower Sequencer ALU MSB |
| 4 | Upper Sequencer internal bus LSB | Lower Sequencer internal Bus LSB |
| 5 | Lower Sequencer flag | Upper Sequencer flag |
| 6 | Not used | Real-Time sector or index |
| 7 | Init/Fun Flag | Delayed zero test |
| 8 | Control Bus Error * | Data word clock |
| 9 | Control SERDES ready | Data SERDES ready |
| 10 | Drive attention | ECC symbol ready |
| 11 | Drive ready to receive | Drive ready to receive |
| 12 | Drive available | R/T transmit ready |
| 13 | Drive Rd/Wrt ready | Drive Rd/Wrt ready |
| 14 | Not used | Zero ready |
| 15 | Control Bus Request Active | Data Bus Request active |

* Inclusive or summation. Refer to Sections 12.5.17 and 12.5.16.

Table 12-16 Description of the Bits to the Upper Test Multiplexer

| Bit | Description |
|---|---|
| 0 | Always a logical 1 |
| 1 | Output of ALU equals a zero |
| 2 | Carry out of ALU from last Upper Sequencer operation |
| 3 | Most significant bit of the output from the ALU from last Upper Sequencer operation |
| 4 | Least significant bit of the BUS from last Upper Sequencer operation |
| 5 | This bit is writable by the Lower Sequencer and readable by the Upper Sequencer. Used by the microcode to synchronize software operations between the Upper and Lower Sequencers. |
| 6 | Not used |
| 7 | This bit indicates whether the microcode is running diagnostics or functional code. The system comes up with this bit cleared. This indicates that the K.sdi/K.sti is running diagnostic code. When the diagnostic code is completed, this bit is set. This bit is set or cleared by the microcode through the IOC Control logic on the output of the Control Store. |
| 8 | This bit is a summary of the following hardware detected error conditions:<br>Control Non Existent Memory<br>Control Data Parity Error<br>Control Bus Error<br>Drive Response Error<br>Instruction Parity Error<br>Refer to Section 12.5.13 for a description of the above errors. |
| 9 | This bit indicates that the Control SERDES is ready for a parallel I/O transfer depending on the mode (read or write). |
| 10 | This bit is the drive attention bit from the selected port. |
| 11 | This bit is the the ready to receive bit from the selected port. |
| 12 | This bit is the port available bit from the selected port. |
| 13 | This bit is the read/write ready bit from the selected port. When this bit is asserted, it indicates the drive may be written to or read from. If during a data transfer the drive detects an error, the data transfer is aborted and the read/write ready bit is negated. |
| 14 | Not used |
| 15 | This bit indicates that a Control Bus Cycle is in progress. This bit is set when the request is initiated and cleared at the completion of the data transfer. |

Table 12-17 Description of the Bits to the Lower Test Multiplexer

| Bit | Description |
|---|---|
| 0 | Always a logical 1 |
| 1 | Output of ALU equals a zero from the last Lower Sequencer operation |
| 2 | Carry out of ALU from the last Lower Sequencer operation |
| 3 | Most significant bit of the output from the ALU from the last Lower Sequencer operation |
| 4 | Least significant bit of the BUS from the last Lower Sequencer operation |
| 5 | This bit is writable by the Upper Sequencer and readable by the Lower Sequencer. Used by the microcode to synchronize software operations between the Upper and Lower Sequencers. |
| 6 | This bit is the real-time sector/index pulse for the drive selected by the Lower Sequencer. |
| 7 | This bit is the zero test bit from the ALU delayed one Lower Sequencer cycle. |
| 8 | This bit is data word clock (DATA WORD CLK) ORed with ALU carry (C16). This bit is used at the beginning of a data transfer to prevent the K.sdi/K.sti microcode from hanging. Data word clock negates soon after the detection of the sync pattern. The microcode then monitors the SERDES signal DATA WD RDY. If the drive fails and does not send a word to the SERDES, the microcode hangs. If the data word clock is not detected within an allotted time, the ALU carry is generated from a software timer that the microcode is looping on waiting for the data from the drive. The ALU carry signal then allows the microcode to get out of the loop. |
| 9 | This bit indicates that the Data SERDES is ready for a parallel I/O transfer depending on the mode (read or write). |
| 10 | This bit indicates that the RS Generator Chip has an ECC symbol ready to read. |
| 11 | This bit is the ready to receive bit from the selected port. |
| 12 | This bit indicates that another word can be parallel loaded into the Lower Real-Time Controller State Serializer. |
| 13 | This bit is the read/write ready bit from the selected port. When this bit is asserted, it indicates the drive may be written to or read from. If during a data transfer the drive detects an error, the data transfer is aborted and the read/write ready bit is negated. |
| 14 | **C, D, E-rev etch**<br>This bit is zero test delayed (ZERO TST DLY) (Bit 7) ANDed with the sector pulse (SECTOR H) from the selected drive. This bit allows the software to set a counter and then loop looking for a Sector pulse while decrementing the counter. If the drive is not working correctly and does not send a Sector pulse, the counter goes to zero and the software exits the loop.<br><br>**F-rev etch**<br>This bit is the AND of DATA WD RDY from the Data SERDES and DREQ FREE (the data bus is not in use). |
| 15 | This bit indicates that a Data Bus Cycle is in progress. This bit is set when the request is initiated and cleared at the completion of the data transfer. |

### 12.7.9 BD Field

The BD field, bus destination, selects the register that will be loaded with the data that has been placed on the BUS lines. Table 12-18 shows the decode of three bits.

**Table 12-18 Decode the bits in the BD Field**

| Decode | Upper Sequencer | Lower Sequencer |
|---|---|---|
| 0 | NOP | NOP |
| 1 | Control Bus Data Xcvrs | Data Bus Data Xcvrs |
| 2 | Control Bus Address Xcvrs | Data Bus Address Xcvrs and Counter |
| 3 | Control SERDES | Data SERDES |
| 4 | Upper Control Register * | Lower Control Register * |
| 5 | Program Address Register | Program Address Register |
| 6 | Status Bus Register | Lower Real-Time Control State |
| 7 | Scratchpad RAM | Scratchpad RAM |

* Also the Diagnostic Register

### 12.7.10 @ Field

The @ field, IOC enable, when set, permits the IOC field to modify the IOC latches. This bit is not set when using the LITERAL field.

**NOTE**

**The IOC field has dual functions: it can be gated to the IOC Control latches when the @ field is set, or the field can be the literal input to the ALU.**

### 12.7.11 IOC Field

The IOC field, input/output control, if enabled by IOC ENABLE, modifies the IOC Latches. Bits <22:20> select the desired latch, and bit 23 sets or clears that latch. Table 12-19 describes the decoding of the bits in the IOC field.

Table 12-19 Decode the Bits in the IOC Field

| Decode | Upper's IOC | Lower's IOC |
|---|---|---|
| 0 | UPR PROC FLAG | LWR PROC FLAG |
| 1 | INIT/FUN FLAG | Not Used |
| 2 | RCVR RDY | Not Used |
| 3 | ROLL | ECC BYPASS |
| 4 | CNTL SERDES ENA | DATA SERDES ENABLE |
| 5 | RCV MODE | RD MODE |
| 6 | UPR SER MUX EN | ECC IOC ENABLE |
| 7 | ENA DESER | ECC TIMING |

### 12.7.12 RA Field

The RA field, A Port Register, selects one of the sixteen internal 2901 registers to be operated on by the ALU. The RA field is a shared field and is only enabled if the AS field does not equal $5_8$ or $7_8$ and bit AB does not specify the LITERAL field.

### 12.7.13 BS Field

The BS field, bus source, selects one of the K.sdi/K.sti registers which are external to the ALU. The contents of the selected register are enabled onto the BUS lines, routed through the input multiplexer, and presented to the D inputs of the 2901 ALU. Table 12-20 describes the decoding of the bits in the BS field. The BS field is a shared field and is only enabled if the AS field equals 5 or 7 and AB equals 0.

Table 12-20 Decode the Bits in the BS Field

| Decode | Upper Sequencer | Lower Sequencer |
|---|---|---|
| 0 | NOP | NOP |
| 1 | Control Bus Data Xcvrs | Data Bus Data Xcvrs |
| 2 | Control Bus Address Xcvrs | Low Order ECC Residue |
| 3 | Control SERDES | Data SERDES |
| 4 | Sector/Index Register | Data Bus Address Xcvrs |
| 5 | Control Error Register | Data Error Register |
| 6 | Not Used | High Order ECC Residue |
| 7 | Scratchpad RAM | Scratchpad RAM |

### 12.7.14 S Field

The S field, test sense, is XORed with the output of the Test Multiplexer to generate TST COND TR (test condition true). TST COND TR is then combined with the branch address equal to $7777_8$ and JUMP CNTL 1 to determine whether the sequencer will output the Program Counter, the Stack, or the Branch Address. Table 12-21 describes what operation is performed in the sequencer, depending on the selected test condition and the TST COND TR signal.

**Table 12-21 Decode the Selected Test Condition and TST COND TR**

| Select Test Condition | TEST FOR TRUE | TST COND TR | Operation |
|---|---|---|---|
| 0 | 0 | 1 | Branch, Call, or Return |
| 0 | 1 | 0 | Increment |
| 1 | 0 | 0 | Increment |
| 1 | 1 | 1 | Branch, Call, or Return |

### 12.7.15 $ Field

The $ field, conditional execute, enables all instructions when clear. When set, this bit enables the conditional execution of the IOC, bus destination, and ALU instructions. Refer to Section 12.7.6 for a description of how this bit and TST COND TR and EXEC IF COND TRUE work together.

### 12.7.16 K Field

The K field, jump control, is logically combined with TST COND TR and BR ADR <11:00> equal to all ones ($7777_8$) to control the next operation of the 2911 Sequencers (Section 12.7.14). This combination controls the operation of the internal Stack, and it determines whether the sequencer will output the:

- Program counter
- Stack contents
- Branch address field

Table 12-22 describes the decode of the input signals for the sequencer operation.

Table 12-22 Decode Input Signals for Sequencer Operation

| | JUMP | TST COND | | | |
|---|---|---|---|---|---|
| Branch Address | CNT 1 | TR | STACK | Y | Operation |
| *** | 0 | 0 | HOLD | PC | Increment |
| *** | 1 | 0 | HOLD | PC | Increment |
| $7777_8$ | 0 | 0 | HOLD | PC | Increment |
| $7777_8$ | 1 | 0 | HOLD | PC | Increment |
| *** | 0 | 1 | HOLD | BA | Branch |
| *** | 1 | 1 | PUSH | BA | Call |
| $7777_8$ | 0 | 1 | HOLD | BA | Branch |
| $7777_8$ | 1 | 1 | POP | STK | Return |

*** can be anything except $7777_8$

### 12.7.17 P Field

The P field, instruction parity, is the single parity bit for the 48-bit microinstruction word. This bit is calculated odd parity for the other 47 bits. If the number of 1 bits in the other 47 bits is even, then this bit is a 1 making the 48-bit microinstruction word odd parity. If the number of 1 bits in the other 47 bits is odd, then this bit is a 0 making the 48-bit microinstruction word odd parity.

### 12.7.18 BA Field

The BA field, branch address, provides the instruction address for conditional and unconditional branches and subroutine calls. Also, during the execution of certain instructions, this field provides supplemental data and control information (Section 12.7.19).

### 12.7.19 Destination and Branch Field Combinations

The BA field is not always a branch address. This field is also used for data and control for certain instructions. When the BD field equals 2, these lines are used as data and control lines. These BA and BD field combinations are described in the following two sections.

#### 12.7.19.1 Write Control Bus Address Transceivers

Loading of the Control Bus Address Transceivers is done with BUS and the BR ADR lines. Bus lines BUS D<15:00> are latched and then gated onto CADR lines <16:01>, respectively, during the Control Memory Cycle. BR ADR 11 is latched and gated onto CADR 00 (LSB). BR ADR <10:08> are latched and gated onto CCYCLE <0:2>, respectively. BR ADR 07 is latched and then determines, during the Control Memory Cycle, whether the K.sdi/K.sti Control Bus Data Transceivers will be gated onto the CDATA lines. In order to load the register, the BD field must equal two and the Upper Sequencer must be active.

Table 12-23 shows what Control Memory cycles are generated from BR ADR <10:07>.

**Table 12-23 Control Memory Cycles Initiated from Decode the BA Field**

| 10 | 09 | 08 | 07 | Control Memory Cycle |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | Initiate Control Memory Read Cycle |
| 1 | 0 | 0 | 1 | Initiate Control Memory Write Cycle |
| 0 | 1 | 0 | 1 | Initiate Interrupt Cycle |
| 0 | 0 | 1 | 0 | Initiate Lock Cycle (Read and Force) |
| 1 | 1 | 1 | 1 | Initiate NMA Cycle |

### 12.7.19.2 Write Data Bus Address Transceivers and Counter

When the microcode loads the Data Bus Address, part of the address is loaded from the BUS lines, and branch address bits BR ADR <10:09> are logically ANDed, latched, and then used to drive DNMA during the Data Bus Cycle (Figure 12-19).

Table 12-24 shows the decoding of BR ADR <10:09> to generate DNMA.

**Table 12-24 Decoding BR ADR <10:09> To Generate DNMA**

| BR ADR 10 | BR ADR 09 | DNMA |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# CHAPTER 13
# POWER SYSTEM

## 13.1 INTRODUCTION

The HSC power system consists of an ac power controller, an airflow sensor, dc power supplies, and power switches and indicators.

The HSC50 has an ac power controller and one or two power supplies depending on the number of data channels installed. The HSC70 has a different power controller and always has both power supplies installed. The following sections describe the HSC power system.

## 13.2 AC POWER CONTROLLER

The HSC has four ac power controllers. The HSC50 uses the following two:

HSC50-AA—70-19122-01 for 120/208 Vac
HSC50-AB—70-20613-01 for 380-415 Vac

The HSC70 uses the following ac power controllers:

HSC70-AA—881A for 120/208 Vac
HSC70-AB—881B for 380-415 Vac

The ac power controller provides ac power to the power supplies and blower motor. Figures 13-1 and 13-2 show the location of the ac power controllers in the cabinets. Figures 13-3 to 13-8 show various views of the power controllers. Each controller is described in the service manual.

Each power supply and the blower motor should be plugged into a different input phase. On the rear of the HSC50 ac power controller, you will see the phase outlets marked by L1, L2, and L3 (Figures 13-4 and 13-6). On the HSC70 ac power controller, you will see the phase outlets marked by Ø1, Ø2, and Ø3 (Figure 13-8).

## 13.3 AIRFLOW SENSOR

The airflow sensor is mounted in the exhaust duct of the cabinet and monitors the velocity of the exhausted air through the exhaust duct (Figures 13-1 and 13-2).

The airflow sensor switch is self heating and requires moving air to keep the switch open. Power to heat the sensor comes from the -5.2 volts on the backplane and +12 volts on the main power supply (Figure 13-9). If the blower fails, the sensor energizes the airflow sensor relay (K1) through the same source and interrupts the ac power through the total off circuit in the ac power controller.

Figure 13-1 HSC50 Inside Rear View

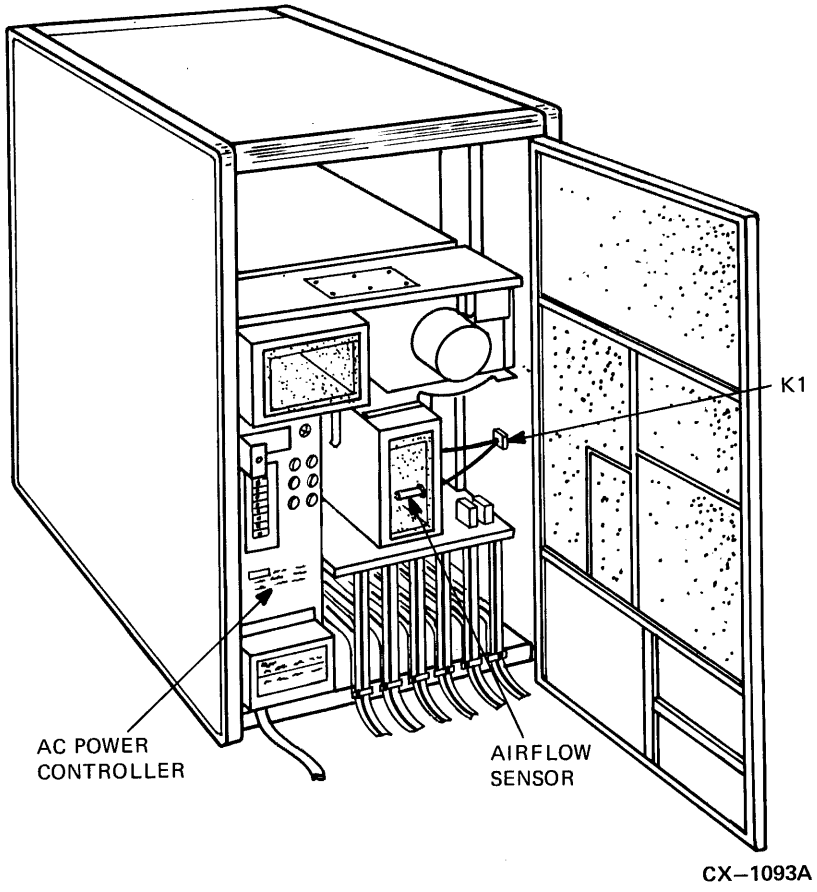


Figure 13-2 HSC70 Inside Rear View

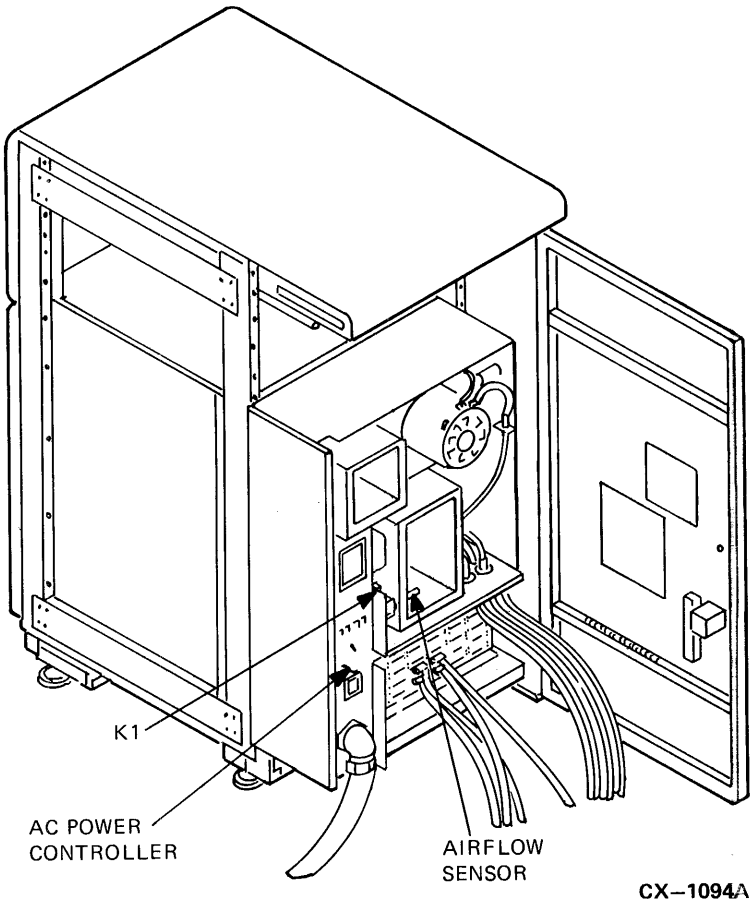


Figure 13-3 HSC50 AC Power Controller (120/208 Vac)—Front View

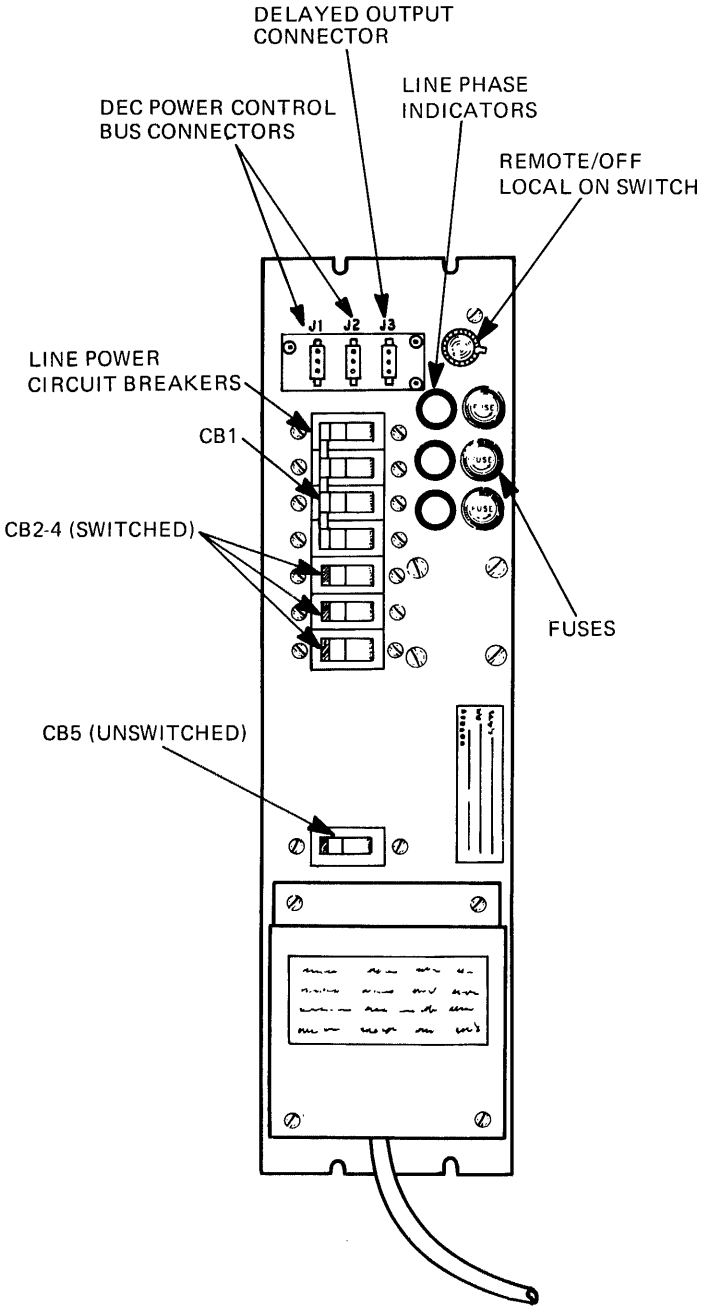
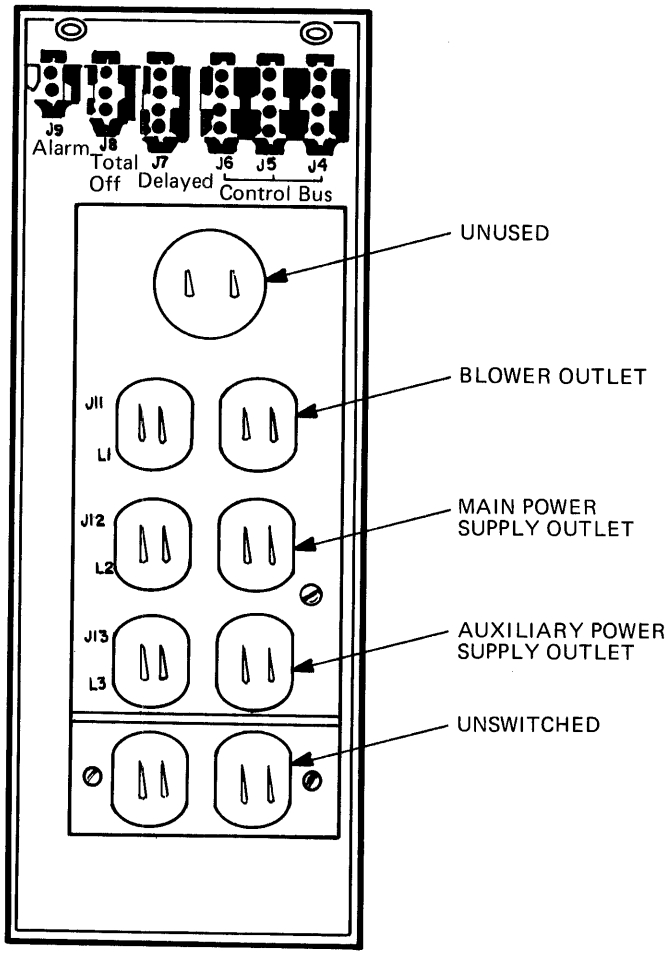


Figure 13-4 HSC50 AC Power Controller (120/208 Vac)—Rear View




CX–411A

Figure 13-5 HSC50 AC Power Controller (380-415 Vac)—Front View

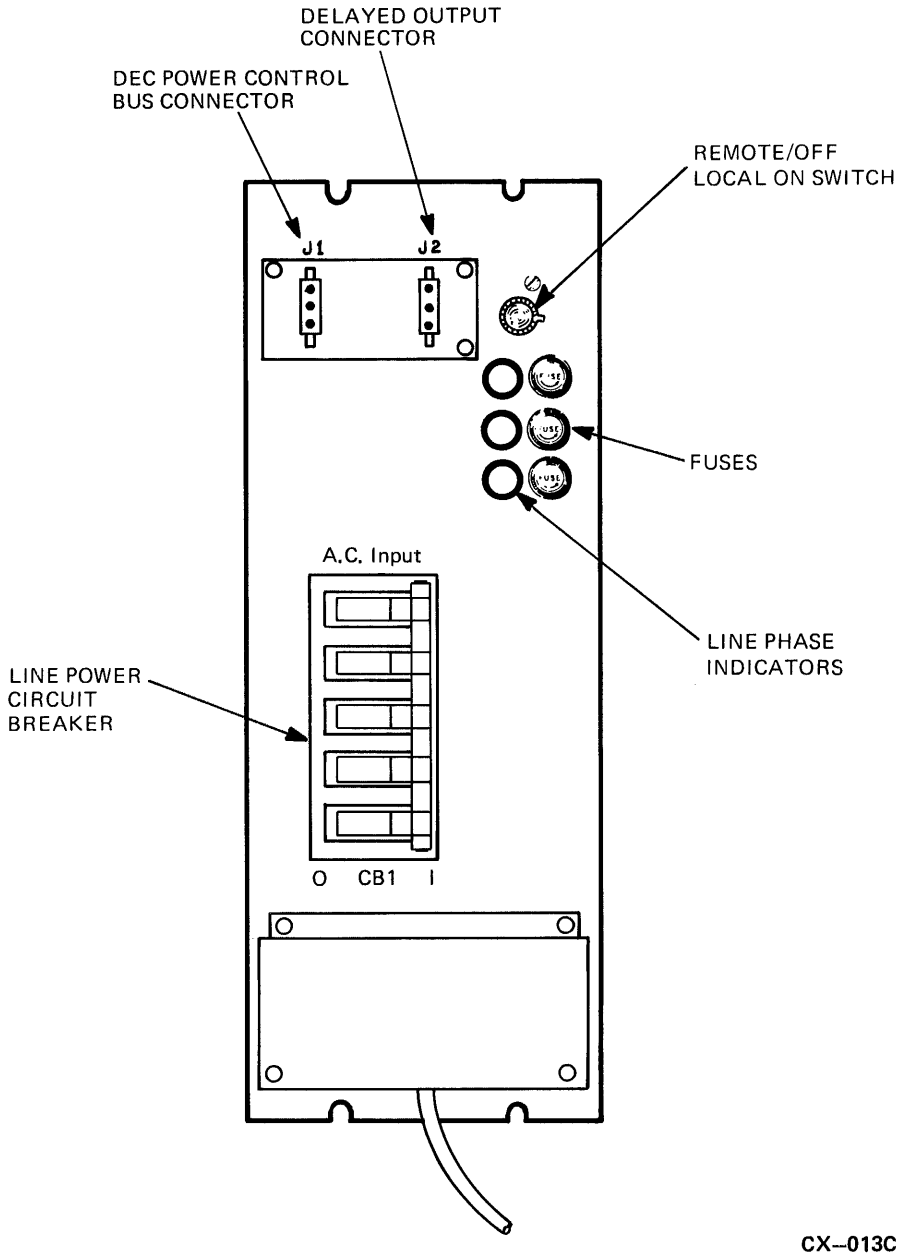


Figure 13-6 HSC50 AC Power Controller (380-415 Vac)—Rear View

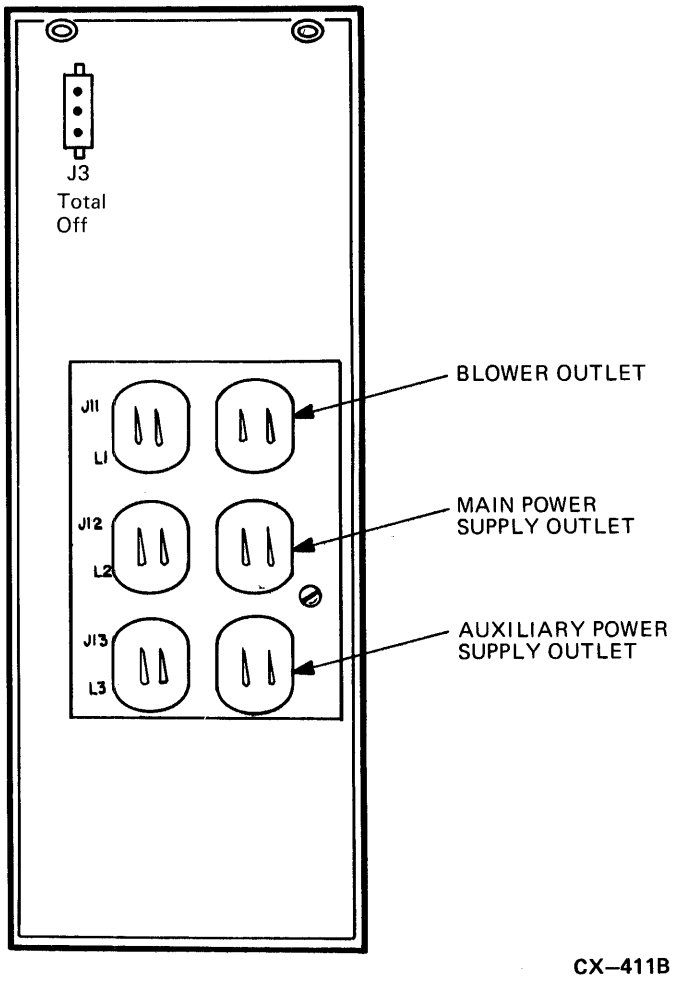


Figure 13-7 HSC70 AC Power Controller—Front View

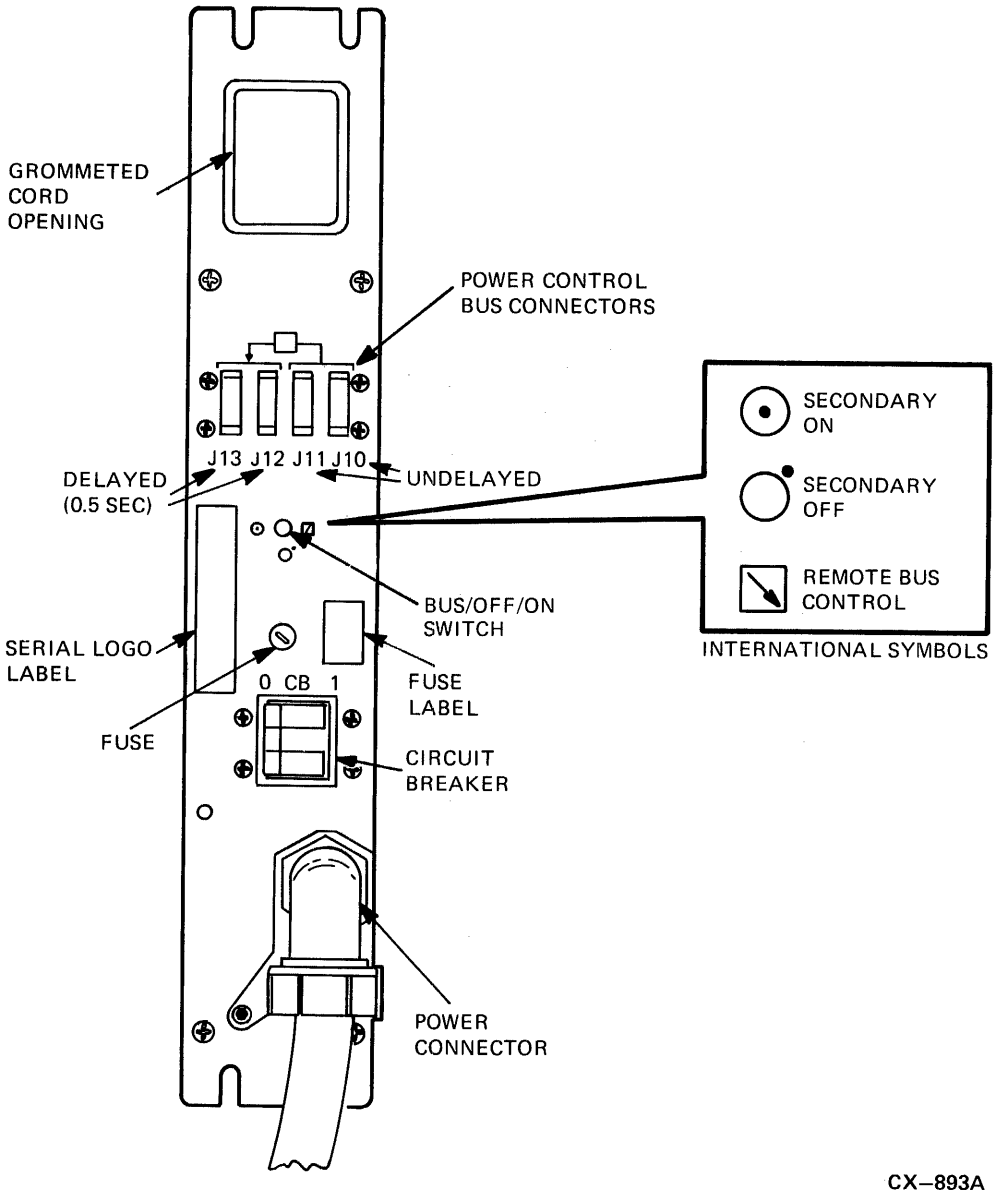


Figure 13-8 HSC70 AC Power Controller—Top View

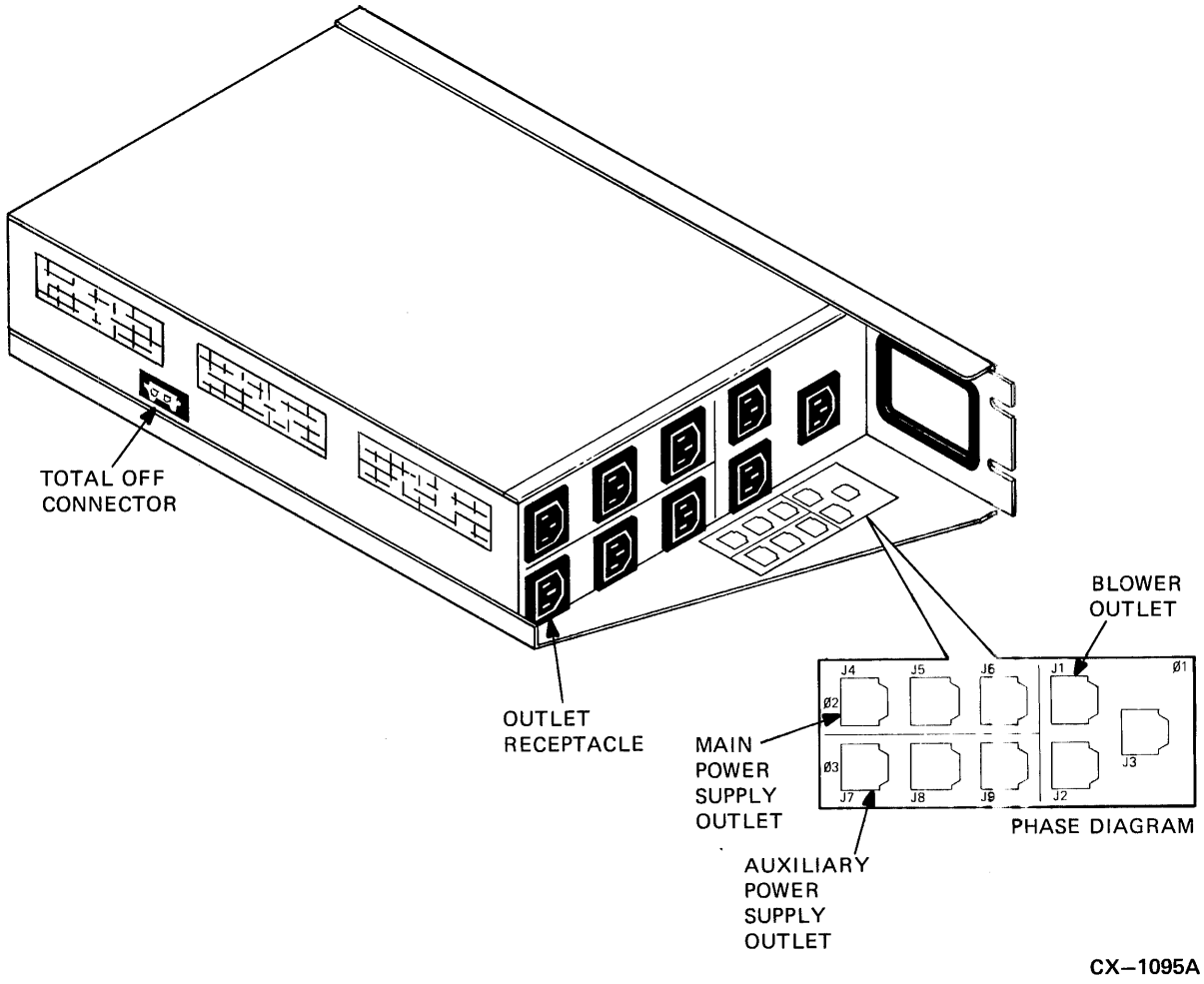
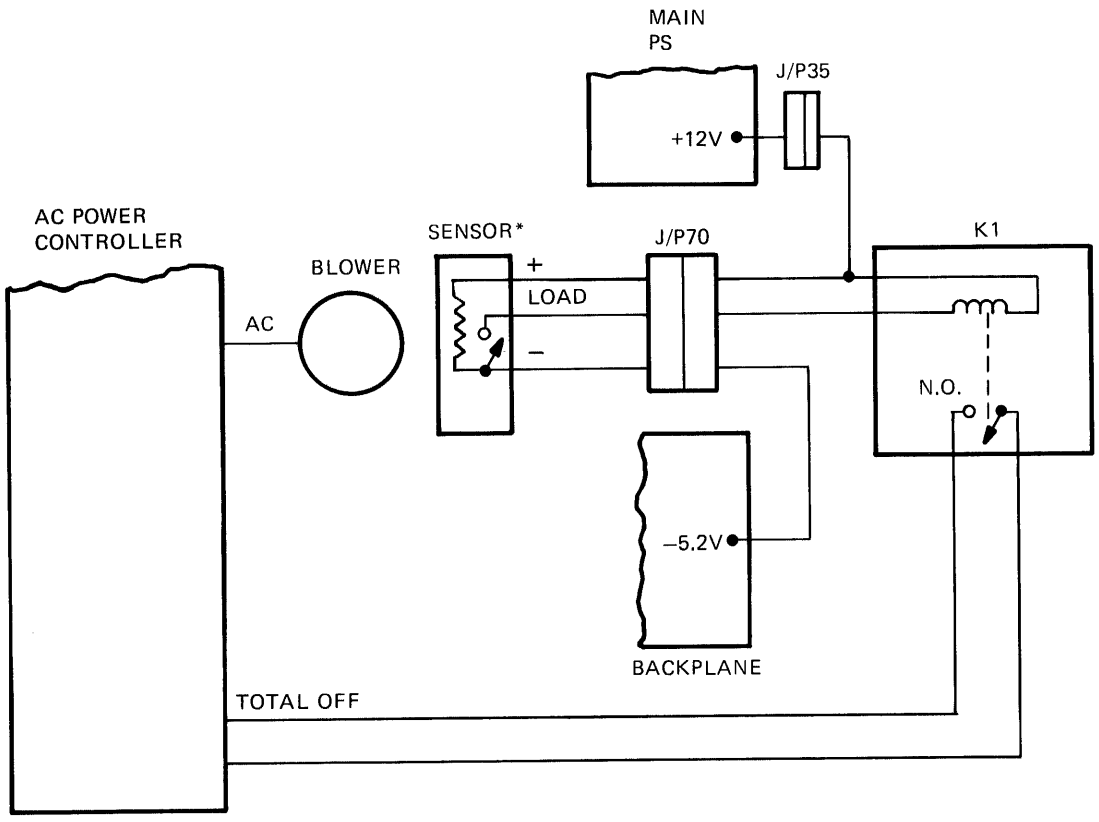


Figure 13-9 Airflow Sensor Cabling




The total off circuits are different in the HSC50 and HSC70. The total off circuit in the HSC50 trips the main circuit breaker in the ac power controller to remove ac power. The total off circuit in the HSC70 deenergizes a relay in the ac power controller to remove ac power.

**NOTE**

**The ac must be turned off and then back on to reset this circuit.**

When the total off circuit removes ac power, the power supplies sense the loss and assert the power fail circuit to the P.ioc or P.ioj. Asserting power fail allows for an orderly shutdown of the HSC.

## 13.4 POWER SUPPLIES

The HSC has two power supplies: the main power supply and the auxiliary power supply. Figures 13-10 and 13-11 show the location of the power supplies. These two power supplies are the switching regulator type. The output voltage is maintained through sense line feedback from the backplane. Figures 13-12 and 13-13 show the connections between the power supplies and backplane.

The need for an auxiliary power supply in the HSC50 depends on the number of data channel modules installed. If the HSC50 has more than three data channels, the auxiliary power supply is installed. However, the HSC70 always contains both power supplies.

**NOTE**

**These power supplies are not field adjustable.**

The following sections describe the two power supplies.

### 13.4.1 Main Power Supply

The main power supply is rated at 750 watts and provides the following voltages to the backplane:

+5.0 volts
-5.2 volts
+12 volts

The output current depends on how the power supplies are configured. If the auxiliary power supply is not installed on an HSC50, the main power supply operates in mode 1. If the auxiliary power supply is installed, the main power supply operates in mode 2.

#### 13.4.1.1 Main Power Supply—Mode 1

Mode 1 is when the main power supply supplies all of the power for the HSC50 (no auxiliary power supply). Table 13-1 shows the current outputs of the main power supply in mode 1.

**Table 13-1 Main Power Supply Outputs—Mode 1**

| Voltage | Current |
|---|---|
| +5.0 volts | 100 amperes |
| -5.2 volts | 35 amperes |
| +12 volts | 2 amperes (70-20033-01 and 70-20033-02)<br>4 amperes (70-20033-03 and 70-20033-04) |

**CAUTION**

**Never install a 70-20033-01 or 70-20033-02 main power supply in an HSC70 because it does not supply enough current. Use a 70-20033-03 or 70-20033-04 main power supply with a 4 ampere rating for the +12 volts. However, any of the main power supplies may be used in an HSC50.**

Figure 13-10 HSC50—Inside Front View

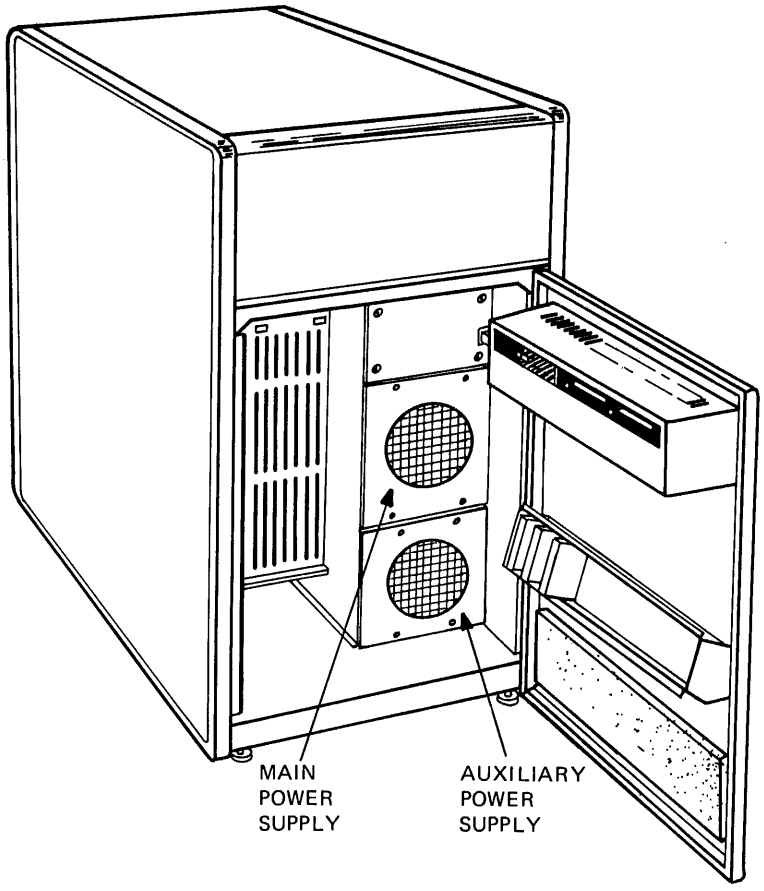
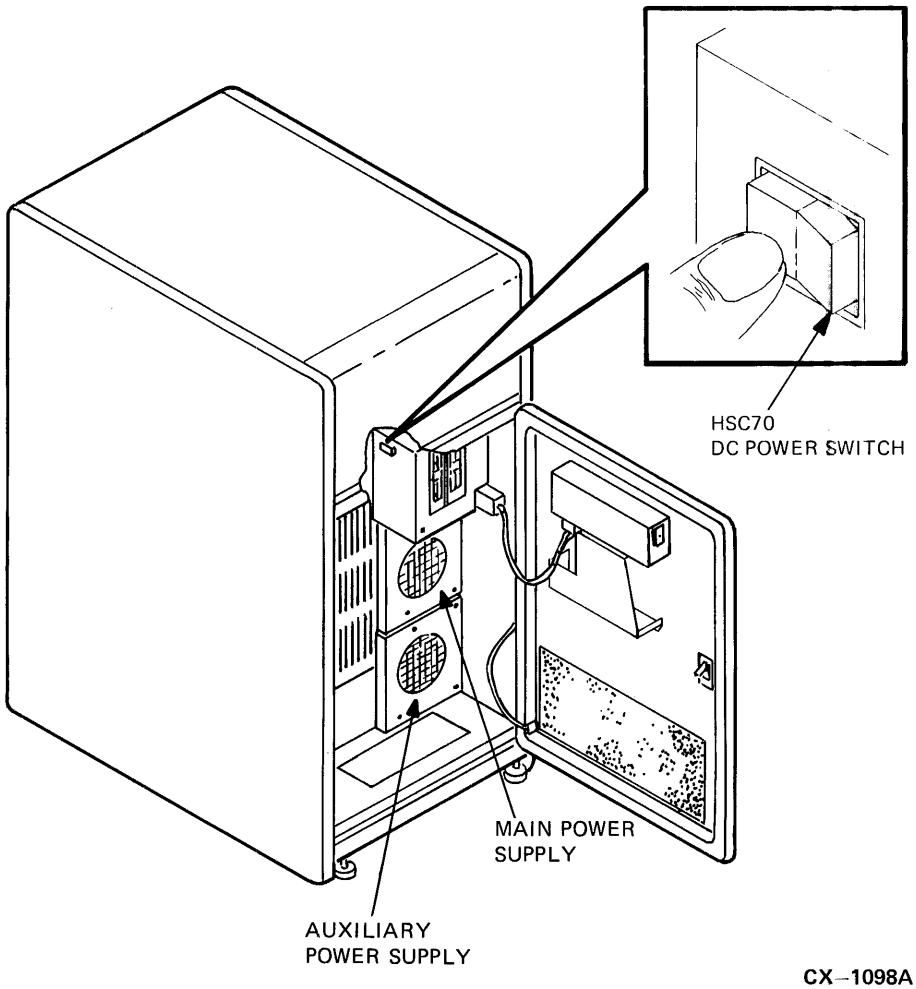
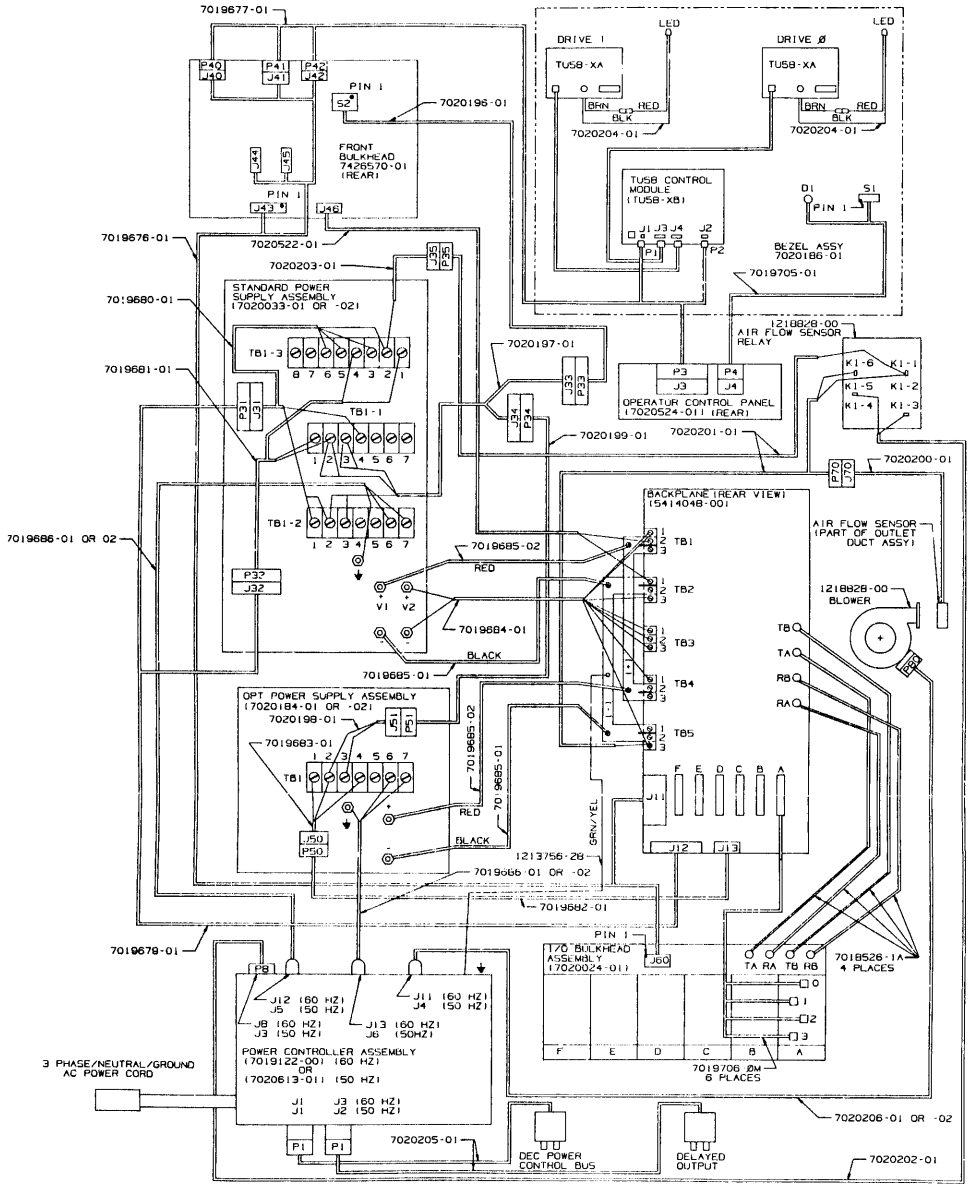


Figure 13-11 HSC70—Inside Front View

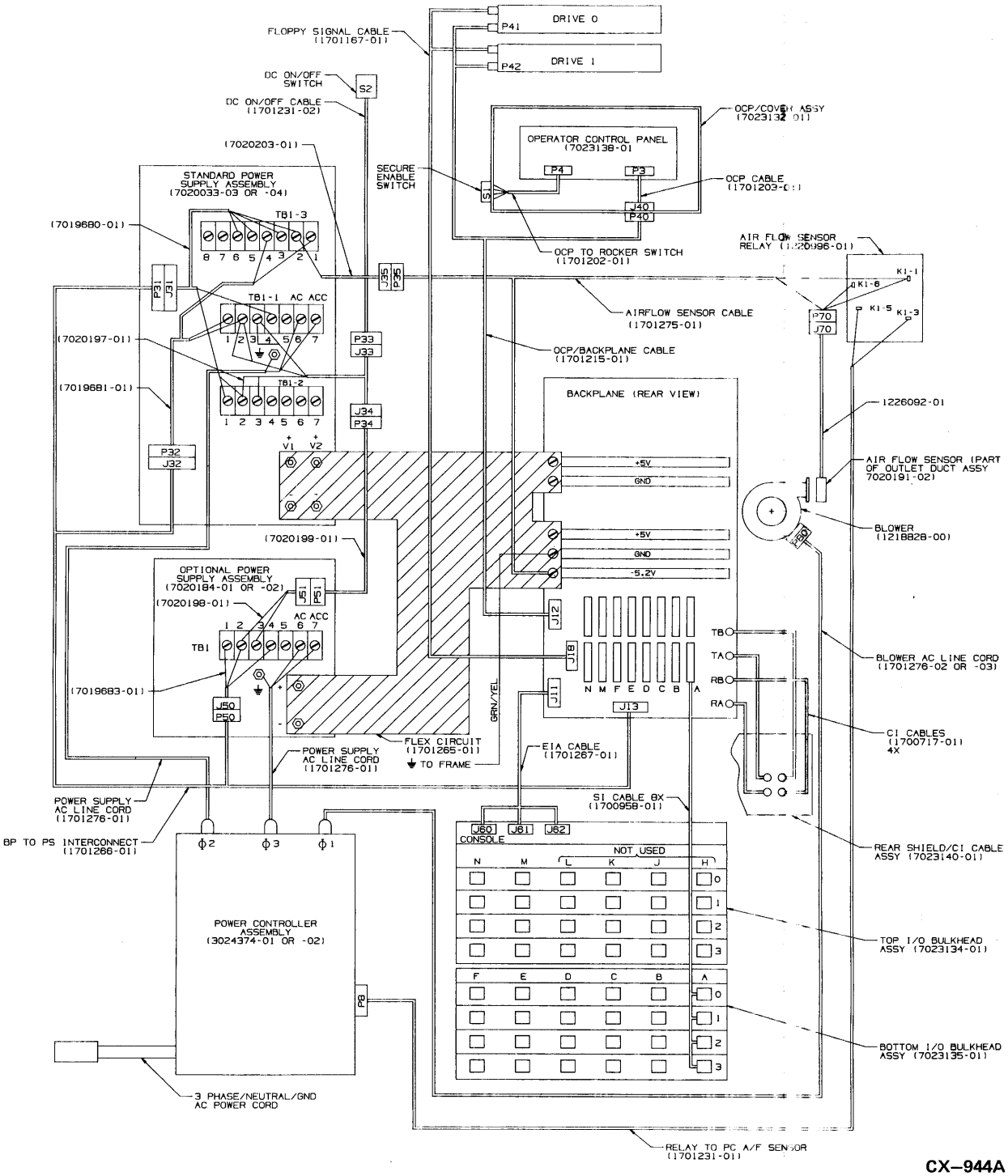


## Figure 13-12 HSC50 Internal Cabling




CX-051A

## Figure 13-13 HSC70 Internal Cabling




CX-944A

### 13.4.1.2 Main Power Supply—Mode 2

The main power supply automatically operates in mode 2 when there is an auxiliary power supply present in the HSC. In mode 2 the auxiliary power supply supplies up to 135 amperes for the +5.0 voltage. After 135 amperes, the main power supply also starts supplying current for the +5.0 volts. Refer to Section 13.5 for the interaction between the two power supplies. Table 13-2 shows the current outputs of the main power supply in mode 2.

**Table 13-2 Main Power Supply Outputs—Mode 2**

| Voltage | Current |
|---|---|
| +5.0 volts | 65 amperes |
| -5.2 volts | 70 amperes |
| +12 volts | 2 amperes (70-20033-01 and 70-20033-02)<br>4 amperes (70-20033-03 and 70-20033-04) |

Notice that the current capacity in mode 2 is increased for the -5.2 volts and decreased for the +5.0 volts. This is automatic. The power supply is still rated at 750 watts.

**NOTE**

**The +12 volt power supply is a slave of the -5.2 volt power supply. Therefore, the -5.2 volt power supply must be generating current for the +12 volts power supply to work. This means that you must have at least one module installed with ECL logic on it (LINK, K.sdi, or K.sti).**

### 13.4.2 Auxiliary Power Supply

The auxiliary power supply has a single +5.0 volt output. In the HSC50, the auxiliary power supply is added if there are more than three data channels present. The HSC70 always has an auxiliary power supply.

The auxiliary power supply is rated at +5.0 volts and 135 amperes. Section 13.5 describes how the two power supplies interact to supply the current for the HSC.

## 13.5 INTERACTION BETWEEN MAIN AND AUXILIARY POWER SUPPLIES

The main and auxiliary power supplies interact to supply the needed current for the HSC. The configuration of the power supplies make this interaction automatic. In brief, the outputs of the +5.0 volts are connected in parallel and the auxiliary power supply is adjusted to +5.1 volts. The auxiliary power supply supplies all of the needed current for the +5.0 volts until the demand from the HSC exceeds its rated output of 135 amperes. The main power supply is adjusted for +5.0 volts; therefore, it does not supply any current when the output from the auxiliary power supply is above +5.0 volts. When the demand exceeds 135 amperes, the voltage output of the auxiliary power supply starts to drop. When the output voltage drops to +5.0 volts, the main power supply starts to supply current for the +5.0 voltage.

If you measure +5.1 volts at the backplane, the auxiliary power supply is supplying all of the +5.0 volts.

## 13.6 PROTECTION CIRCUITS

Each HSC power supply has the following three protection circuits:

- Over voltage protection
- Over current protection
- Over temperature protection

These three protection circuits are explained in the following sections.

### 13.6.1 Over Voltage Protection

The over voltage protection circuits in each power supply provide protection for the modules. When the voltage rises above the values stated in Table 13-3, an SCR shorts the output to ground. You must remove the ac power and restore it before the power supply will work again.

**Table 13-3 Over Voltage Specifications**

| Voltage | Over Voltage Protection Limit |
|---|---|
| +5.0 volts | +6.50 volts |
| -5.2 volts | -6.76 volts |
| +12 volts | +15.0 volts |

NOTE

**If a remote sense wire becomes open-circuited, the voltage increases and triggers the over voltage protection circuit.**

### 13.6.2 Over Current Protection

The over current protection circuits keep the HSC power supplies from destroying themselves. When the output current exceeds the rated output, the output voltage starts to drop (foldback). Table 13-4 shows what current the power supplies supply with a dead short.

WARNING

**You should remove all rings and metal watches when working around the the backplane or power supplies in the HSC. The power supplies have a high output current capacity. A metal object could become red hot if shorted between ground and a voltage.**

Table 13-4 Dead Short Current of the Power Supplies

| Voltage | Maximum Dead Short Current |
|---|---|
| +5.0 volts | |
| Main | 50 amperes |
| Auxiliary | 65 amperes |
| -5.2 volts | 35 amperes |
| +12 volts | 2 amperes |

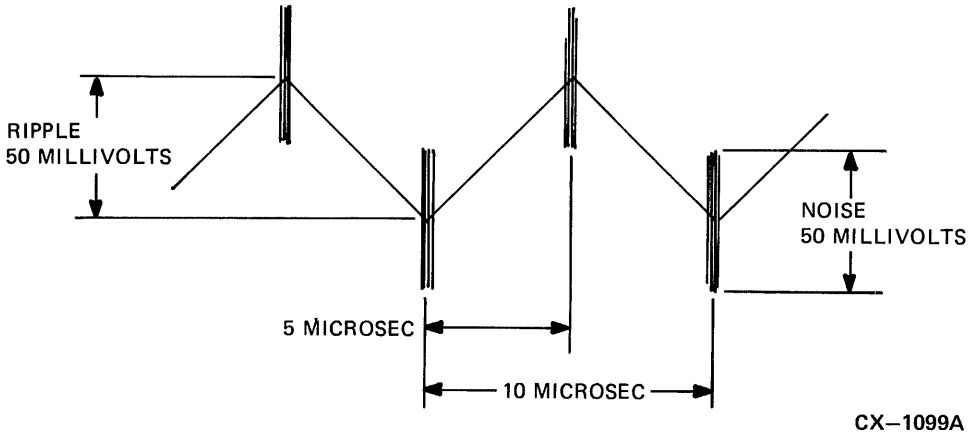
### 13.6.3 Over Temperature Protection

Each power supply has an over temperature protection circuit. When the internal temperature is exceeded, the power supply disables the output and a power fail is detected by the P.ioc or P.ioj.

### 13.6.4 Ripple and Noise

Figure 13-14 shows the ripple and noise present on the dc outputs of the power supplies. *Ripple* is the low frequency component in the waveform. The regulators switch at 100 KHz. *Noise* is the high frequency component of the waveform. The transistors turning on and off at the minimum and maximum points of the ripple cause noise. The maximum allowable ripple is 50 millivolts peak to peak. The maximum allowable noise is 50 millivolts peak to peak.

Figure 13-14 Power Supply Ripple and Noise Specifications



NOTE

**You should not attempt to measure the ripple of either power supply with an ordinary scope. You need special equipment to measure ripple of these power supplies accurately.**

## 13.7 POWER FAIL

Each power supply (main or auxiliary) has a POWER FAIL signal output. This signal indicates the imminent loss of dc voltage due to loss of ac input or internal thermal shutdown. The POWER FAIL signal can be generated by either the main or auxiliary supply. This POWER FAIL signal from each power supply is wire ORed together on the HSC backplane. Figure 13-15 shows where you can check the POWER FAIL signal on the power supplies.

The F11 or J11 detects the POWER FAIL signal and executes a standard PDP-11 Trap to 24 power fail interrupt. You should note following POWER FAIL signal troubleshooting aids:

1. The POWER FAIL signal output from one supply is not an input to the other supply. For example, the main power supply does not know when the auxiliary supply is generating a POWER FAIL. Therefore one supply can be generating a POWER FAIL while the other is supplying voltages. If you suspect a problem with the power system, check the POWER FAIL signal for a low (ground).
2. The POWER FAIL signal affects the HSC50 and HSC70 differently as described below:
   - On the HSC50, the POWER FAIL signal is not an input to the Voltage Monitor Circuit on the P.ioc. Therefore, if the auxiliary power supply is generating POWER FAIL, it is quite likely that the POWER indicator will still be on.
   - On the HSC70, the POWER FAIL signal is an input to the Voltage Monitor Circuit on the P.ioj. Therefore, if a power supply asserts the POWER FAIL signal, the POWER indicator on the OCP will be off.

## 13.8 DC POWER SWITCH

You can manually turn the power supplies off and on using dc power switch on the front bulkhead (Figures 13-16 and 13-11). This switch disables the switching regulators in the power supplies. It also allows you to turn dc off without turning off the blower.

NOTE

**Turning off the dc power switch does not activate the POWER FAIL to give any advanced warning that dc is going away. The dc power switch is intended for service purposes only, and you should only use the switch if the HSC is offline to the host. Otherwise, you may lose data.**

## 13.9 POWER INDICATOR

The POWER indicator on the Operator Control Panel is driven from a dc comparator circuit on the P.ioc or P.ioj. This circuit is different for the HSC50 and HSC70:

- On the HSC50, the circuit only monitors the +5.0, -5.2, and +12.0 volts. If the circuit detects that any of these voltages have dropped approximately one third, the indicator goes off.
- On the HSC70, the circuit monitors the +5.0 volts, -5.2 volts, +12.0 volts, and the POWER FAIL signal. If the circuit detects that any of the voltages have dropped approximately one third, or if the power supplies assert POWER FAIL, the indicator goes off.

It is important to note that this indicator being on does not mean the voltages are within specifications.

Figure 13-15 Power Fail Test Points

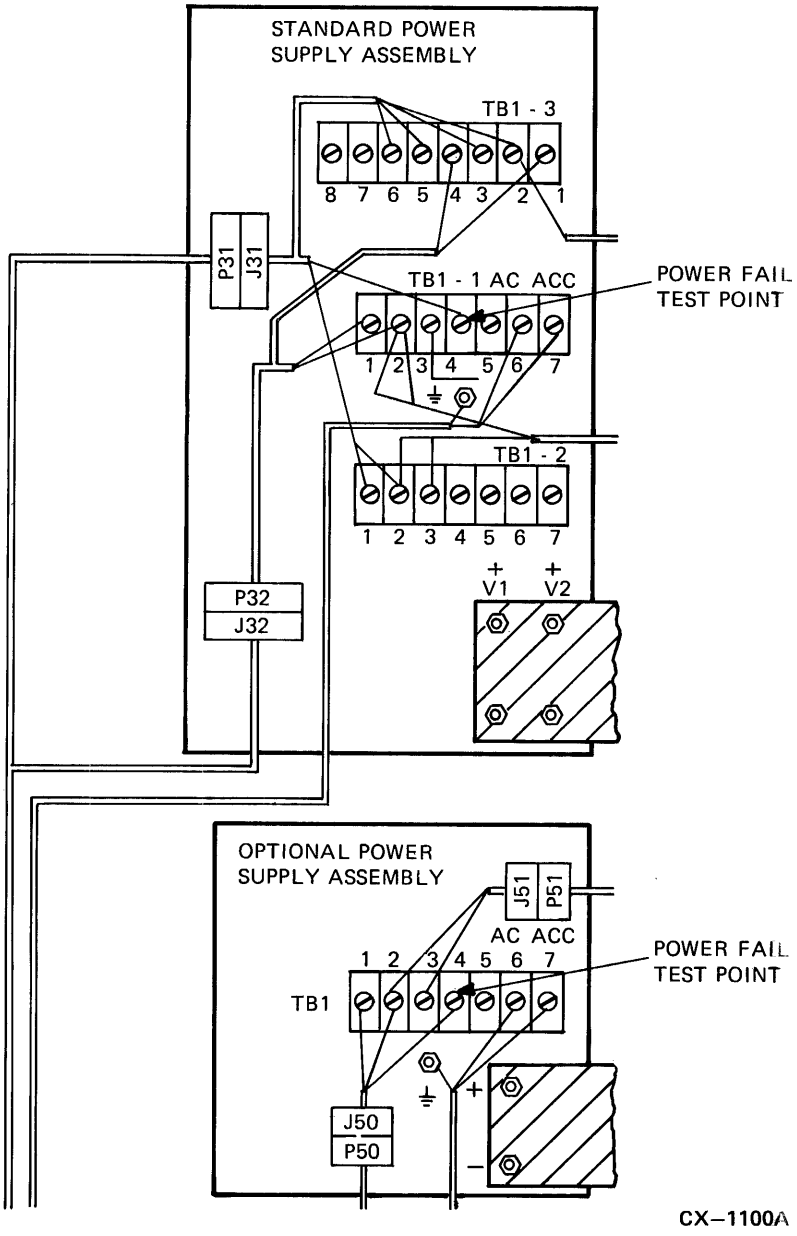
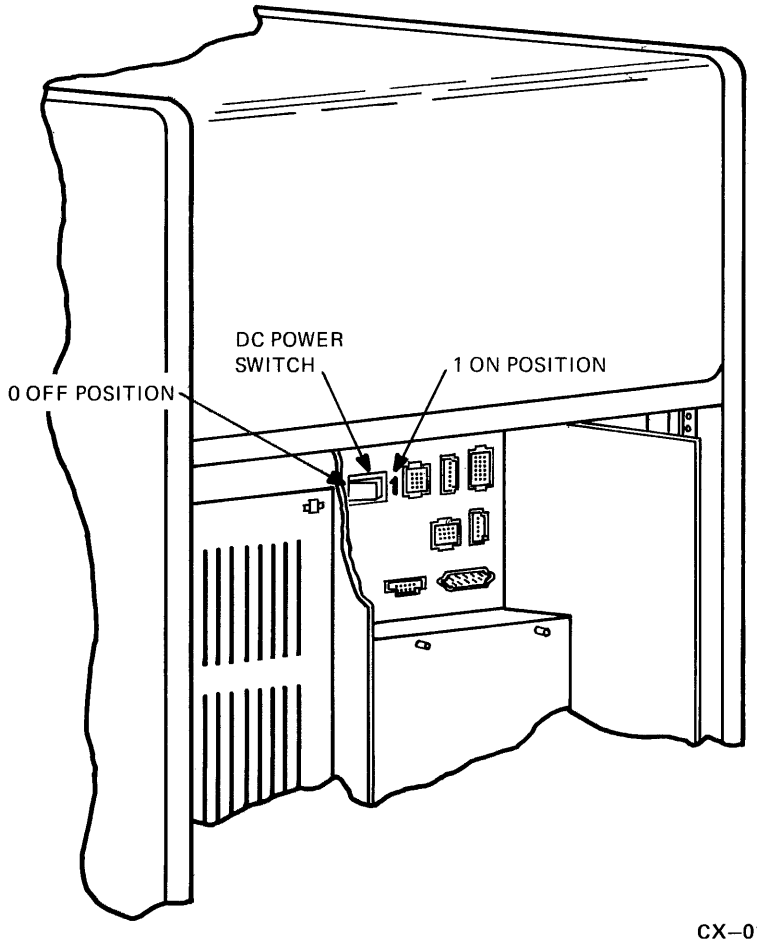


Figure 13-16 HSC50 DC Power Switch Location



# Index

## A



## B

## C

## D

## E

## F



## H

## I

## K

## L



## M

## N



## O



## P

## R

## S



## T

## V



## W



## X

# INSTALLING AND RUNNING THE CI EXERCISER SOFTWARE (AV-T637B-TE)

CONTENTS PAGE

## 1.0 ABSTRACT

The CI Exerciser is an online tool used to detect errors within a CI cluster and isolate a problem to a specific node. The software package consists of a user mode diagnostic that runs under the diagnostic supervisor, a controller class driver, and a responder class driver.

When running the exerciser in a cluster, at least one node must be a controller; any or all of the nodes can be responders. The controller sends commands to the responder, the responder takes the specified action and returns responses to the controller. In this manner, the controller can systematically check out the cluster and isolate any problems to a specific node.

All error printouts are made at the controlling node(s); there is no user interface to the responder.

## 2.0 PREPARING TO RUN THE CI EXERCISER

### 2.1 Process Quotas And Privileges

When running the CI Exerciser, the user must have specific privileges and quotas. The required values can be given to an account by running the authorize program. The privileges and quotas must at least meet the minimum shown in appendix B.

### 2.2 System Resources

The SYSGEN utility can be used to modify the parameters specified below. Since most are non-dynamic, the system must be REBOOTED before the new parameters values take effect.

IT IS STRONGLY RECOMMENDED THAT THE SYSTEM MANAGER BE NOTIFIED BEFORE MODIFYING THE SYSGEN PARAMETERS AND REBOOTING THE SYSTEM.

### 2.3 Modifying The System Parameters

The CI Exerciser (and associated drivers) require a certain amount of system resources. Specifically, the values for nonpaged pool, IRP count, SRP count, LRP count, scsconncnt and scsbuffcnt should be increased before loading and running the exerciser. Appendix A shows how to calculate the required values. The CIEPARAMS command file will modify these values for you. DO NOT RUN THIS COMMAND FILE WITHOUT FIRST NOTIFYING THE SYSTEM MANAGER. There is a second command file that can be run to put these sysgen values back to the values prior to running the CIEPARAMS command file, it is called CIEPARRST. If CIEPARAMS is run more than once, CIEPARRST will only return the values to the last set. Remember that once the sysgen parameters are changed, rebooting will not restore them to there original values. If CIEPARAMS is run twice without running CIEPARST, the values must be changed manually. Refer to appendix A to calculate the amount by which CIEPARAMS will change the sysgen parameters and for instructions to execute the

command files. CIEPARAMS and CIEPARST command files are put into the sysmaint area by the installation proceedure explained below.

The cluster configuration is important in modifying these parameters. Controller/Responder nodes require greater increases to the parameters than Responder nodes, and larger clusters require greater increases to the parameters than small clusters.

Refer to appendix A for a table that shows the additions that should be made to the system parameters as a function of node type and cluster configuration.

## 2.4 SCS System ID

Each node in the CI cluster must have a unique, NON-ZERO SCS system identification. This parameter is strictly a software entity used by the CI software to distinguish between nodes in the cluster, and should not be confused with the system configuration register. The SCS System ID is modified by the sysgen parameter SCSSYSTEMID. If more than one node in a cluster has the same SCS system ID, the diagnostic will NOT run correctly.

## 2.5 Polling Interval

The interval at which the CI port driver polls should be kept at 15 seconds or less. This restriction helps avoid excessively long timeout intervals which can occur if an error is detected by the exerciser. The polling interval is controlled by the sysgen parameter PAPOLLINTERVAL.

# 3.0 INSTALLING THE CI EXERCISER SOFTWARE

The CI Exerciser software resides on floppy disk (AS-T637A-TE) for a CI780 or on TU58 (BE-T645A-DE) for a CI750. Installing the software simply involves executing a command procedure on the media which copies the appropriate files into the SYS$MAINTENANCE area. To install the CI exerciser software on a VAX do the following:

- Log into the field service or system manager account
- Remove the console media from the console device
- Insert the CI Exerciser media into the console device
- Mount the CI Exerciser media by typing "MOUNT CS1 CIE"

  Note:
  If the attempt to mount the CIE media fails due to no device, then the console must be connected using sysgen. Type the following and then continue the installation:

```
$ MCR SYSGEN
SYSGEN> CONNECT CONSOLE
```

```
SYSGEN> EXIT
```

- Perform the installation by typing "@CS1:[CIE]CIEINSTAL"
- Dismount the CI Exerciser media by typing "DISMOUNT CS1"
- Remove the CI Exerciser media from the console device
- Replace the console media in the console device

The latest version of the Diagnostic Supervisor should also be put into the sysmaint area and used to run EVXCI. Diagnostic supervisor media is included in the installation kit.

## 4.0 LOADING THE CI EXERCISER SOFTWARE

Before running the CI exerciser, a second command procedure must be executed to load the appropriate class driver(s) into nonpaged pool. This procedure, CIELOAD.COM, prompts the user as to whether the node is to be a controller or just a responder, and loads the driver(s) appropriately. Running the controller in a node will cause moderate degradation of system performance, while running just the responder will have less of a degrading effect.

The drivers must be loaded each time the system is rebooted. To load the drivers, perform the following steps:

- Log into the field service or system manager account
- Establish CMKRNL privilege by typing "SET PROC/PRIV=CMKR" (Not necessary if the system manager account is being used)
- Load the software by typing "@SYS$MAINTENANCE:CIELOAD"

## 5.0 RUNNING THE CI EXERCISER

Once the CI class drivers have been loaded on the system, the responder is ready to participate in cluster test activity. The following additional commands must be issued to run the controller software. Note that an ATTACH and SELECT command must be issued for each port that is to be tested.

For a CI780...

```
$ RUN ESSAA
DS> LOAD EVXCI
DS> ATT CI780 SBI PAA0 tr br port_number
DS> ATT CI_NODE PAA0 node_name VAX780 port_number
DS> attach additional ports...
DS> SELECT PAA0
DS> select additional ports...
DS> ST
```

For a CI750...

```
$ RUN ECSAA
DS> LOAD EVXCI
DS> ATT CI750 HUB PAA0 slot br port_number
DS> ATT CI_NODE PAA0 node_name VAX750 port_number
DS> attach additional ports...
DS> SELECT PAA0
DS> select additional ports...
DS> ST
```

Note:

If running the generate activity test, test 20, control C may take up to 30 seconds to respond. when the DS prompt is returned, the desired action may be taken.

Note:

Only one controller should run either the generate activity test (test 20) or the monitor performance counters test (test 21) in a cluster at one time.

A help facility is also available with the CI Exerciser. Type "HELP EVXCI" to find the topics for which help information is available, and "HELP EVXCI topic" for information about a particular topic.

# APPENDIX A

## SYSGEN PARAMETERS

Below are two table, the first for nodes that are to be controllers and responders and the other for nodes that are to be only responders. A controller node must be a responder. The values are calculated by first determining the maximum number of controllers and responders that will be running in the system. Then plug in the values and determine the amount to increase the parameter.

C = maximum number of controllers in the cluster.

R = maximum number of responders in the cluster.

+ add

* multiply

| Parameter | Value to add to existing parameter |
|---|---|
| NPAGEDYN | 29k + ( 2.5k * C ) bytes |
| SRPCOUNT | 66 + ( 2 * R ) + ( 5 * C ) |
| IRPCOUNT | ( 8 * R ) + ( 8 * C ) |
| LRPCOUNT | ( 7 * R ) + ( 2 * C ) |
| SCSCONNCNT | ( 2 * R ) + C |
| SCSBUFFCNT | 2 + ( 2 * C ) |

Table 1 - Controller / Responder Node

C = maximum number of controllers in the cluster.

R = maximum number of responders in the cluster.

+ add

* multiply

| Parameter | Value to add to existing parameter |
|---|---|
| NPAGEDYN | 5k + ( 2.5k * C ) bytes |
| SRPCOUNT | 66 + R + ( 5 * C ) |
| IRPCOUNT | ( 4 * R ) + ( 8 * C ) |
| LRPCOUNT | ( 2 * R ) + ( 2 * C ) |
| SCSCONNCNT | R + C |
| SCSBUFFCNT | 2 * C |

Table 2 - Responder Only Node

C = maximum number of controllers in the cluster.

R = maximum number of responders in the cluster.

+ add

* multiply

Example -

Suppose one controller node is being run in a three node cluster, with all three nodes being selected for test. Then, the following increases should be made to the sysgen parameters:

C = 1
R = 2

In the controller/responder node:

- NPAGEDYN 29k + (2.5 * 1) = 31.5k bytes
- SRPCOUNT 66 + (2 * 2) + (5 * 1) = 75
- IRPCOUNT (8 * 2) + (8 * 1) = 24
- LRPCOUNT (7 * 2) + (2 * 1) = 16
- SCSCONNCNT (2 * 2) + 1 = 5
- SCSBUFFCNT 2 + (2 * 1) = 4

The following increases should be made to the sysgen parameters in the two responder nodes:

- NPAGEDYN 5k + (2.5 * 1) = 7.5k bytes
- SRPCOUNT (1 * 2) + (5 * 1) = 7
- IRPCOUNT (4 * 2) + (8 * 1) = 16
- LRPCOUNT (2 * 2) + (2 * 1) = 6
- SCSCONNCNT 2 + 1 = 3
- SCSBUFFCNT (2 * 1) = 2

# APPENDIX B

## PROCESS QUOTAS AND PRIVILEGES

The following is a table of minimum and typical process quotas and privileges needed to run the CI exerciser. These values can be modified by running the Authorize program. Authorize has a help facility to aid in making any necessary changes.

| | | | | | |
|---|---|---|---|---|---|
| PRIO: | 4 | BYTLM: | 65000 | BIOLM: | 1000 |
| PRCLM: | 2 | PBYTLM: | 0 | DIOLM: | 1000 |
| ASTLM: | 10 | WSDEFAULT: | 150 | FILLM: | 20 |
| ENQLM: | 20 | WSQUOTA: | 1024 | SHRFILLM: | 0 |
| TQELM: | 100 | WSEXTENT: | 1024 | CPU: | no limit |

Privileges

CMKRNL,DIAGNOSE,LOG_IO,PSWAPM,PHY_IO

Digital Equipment Corporation • Colorado Springs, CO 80919