Space Science
UW SSEC Publication
No. 89.11.M1

McIDAS

Man computer Interactive Data Access System

Multisourcerer Manual

November 1989

Space Science and Engineering Center
University of Wisconsin - Madison
1225 West Dayton Street
Madison, WI 53706
Telephone 608-262-2455
TWX 608-263-6738

Multisourcerer Manual

Contents Page

Contents

Page

GPCI Detailed Circuit Description

Contents **Page**

Contents Page

Contents Page

Contents **Page**

CDR Supplemental Data

Figures, Tables and Timing Diagrams

Contents Page

Figures

Tables

Contents Page

Timing Diagrams

Reader Response

We would like your comments on our manual. Please take a moment to fill out this self-mailing form and return it to SSEC - McIDAS User Services and Documentation. Thank you for your help.

Manual name (from the title page)

____

Did you find this manual helpful? Did it provide the required information?

____

____

____

Did you find errors or omissions in the manual? Please be very specific.

____

____

____

Please rate this manual on the following items from 1 to 5, with 1 as low and 5 as high.

Text	1	2	3	4	5
Graphics	1	2	3	4	5
Ease-of-use	1	2	3	4	5
Overall	1	2	3	4	5

What is the best feature of this manual?

____

____

____

What is the worst feature of this manual?

____

____

____

Your name and title ______________________________

Date ______________________________

McIDAS site ______________________________

Address ______________________________

____

Telephone ______________________________

Check here if you would like a written reply. ___

Fold Here

Fold Here

Space Science and Engineering Center
McIDAS User Services and Documentation, Room 611
University of Wisconsin - Madison
1225 West Dayton Street
Madison, WI 53706

Introduction

Introduction

SSEC's development of the Multisourcerer was motivated by the decision to reduce the proliferation of both the different types and physical number of I/O devices that are connected to the IBM I/O channel. In addition, the new generation of I/O devices to be built for the Multisourcerer will be more reliable, maintainable, and easier to use than the current generation of I/O devices.

The IBM I/O Channel is the means by which data is transferred to and from the central processing unit of several different kinds of IBM midsized computers, including the 43XX series around which the current McIDAS family is built. SSEC has developed a number of specialized I/O devices to connect to the I/O channel. A different interface was designed for each of these devices and optimized to the function performed by the device. This approach has some limitations. One is the growing number of channel interface designs for which SSEC must provide documentation, maintenance, and spare parts. Another drawback is the proliferation of physical boxes connected to the I/O channels of the computer.

To address these problems, SSEC developed the Multisourcerer. The main component of the Multisourcerer is the General Purpose Channel Interface (GPCI), a channel interface with sufficient generality to control the current generation of SSEC I/O devices. Future I/O devices developed at SSEC will be designed to match the standard established with the Multisourcerer.

The Multisourcerer is also a multi-device controller as shown in Figure 1 on the next page. That is, multiple ingestors and/or communication devices can be contained within a single Multisourcerer. Adding a new data stream to an existing McIDAS system can be as simple as plugging in a new board, connecting the data source to the back of the chassis, and perhaps upgrading the firmware PROMs on the Multisourcerer (the software must be present on the IBM computer to handle the new data source). This procedure contrasts with the former method of adding a new data stream by finding more rack space for another box, then connecting another set of heavy I/O cables as well as connecting the new data source.

In addition to the GPCI, the Multisourcerer has a seven-slot card cage with room for six devices. It also includes a keypad and single line display to allow operators and maintenance personnel easy use of the I/O devices. Data transfer within the Multisourcerer is faster than the IBM I/O channel. This allows us to use the full bandwidth of the channel.

The Multisourcerer combines performance, flexibility, and ease of use in one unified design that provides SSEC with a standard method of connecting our specialized devices to the IBM I/O channel.

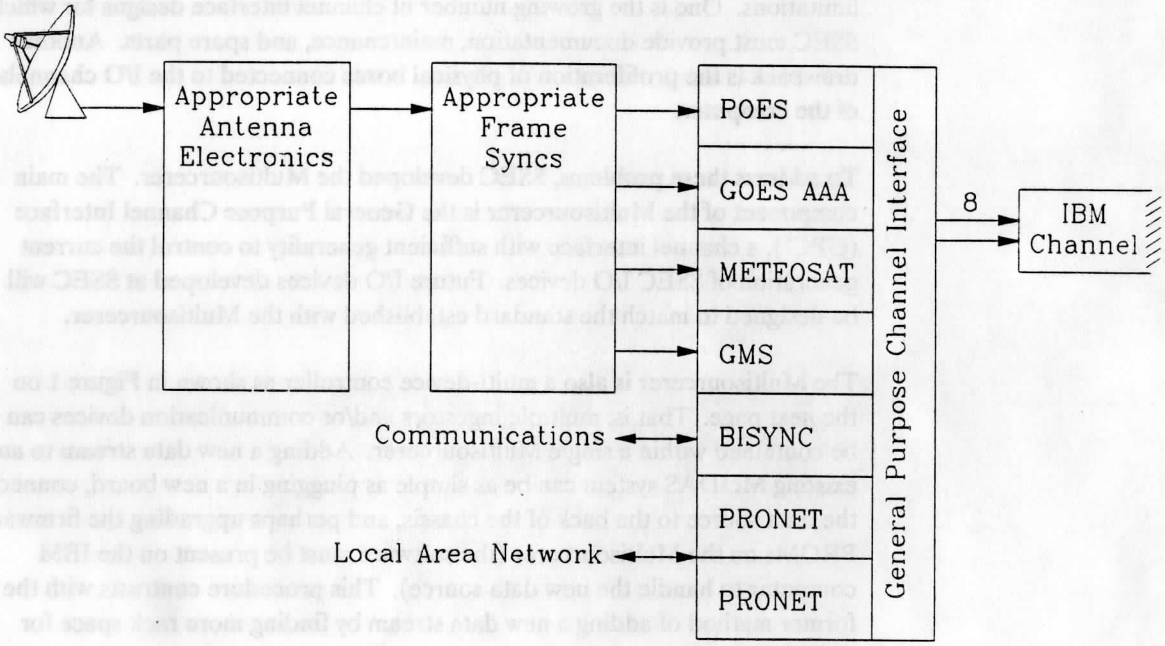


Figure 1. Multisourcerer Interface

Installation and Configuration

Installation

Use the instructions below to install the Multisourcerer.

1. Remove the Multisourcerer from its packing box.

2. Take off the top panel of the Multisourcerer and open the front panel. Remove the interior packing material, inspecting it for additional components.

3. In separate boxes, you will find the front panel single line display and the General Purpose Channel Interface (GPCI) card. Taking precautions to avoid static discharge, remove the display from its packing material.

4. Mount the display to the inside of the front panel using the plastic screws provided, inserted into the mounting posts.

5. Connect the power and signal cables to the display.

6. Attach the power connector with the bottom pins aligned. The red wire should be lined up with the bottom pin of the connector on the display.

7. Attach the signal cable with the brown wire on top.

8. Replace the top panel of the Multisourcerer.

9. Remove the GPCI from its packing material. Carefully place the GPCI in the bottom slot of the Multisourcerer card cage. Do not damage the insulation of the ribbon cables below the card.

10. There are three cables to attach to the GPCI card. Attach the cable from the front panel to the edge card connector J3. Attach the cable from the Driver Receiver card (the PC card mounted on the side of the chassis) to the edge card connector J4. Attach the cable from the test port to the edge card connector J5.

11. Insert the application cards and close the front panel.

12. Now configure the Multisourcerer using the Configuration information on page 2-3.

13. Following configuration, attach the Multisourcerer to the I/O channel. The cables from the computer are attached to the BUS OUT and TAG OUT connectors. Terminators are placed on the BUS IN and TAG IN connectors. The Multisourcerer can also work on a channel with other devices. If this feature is used, the Multisourcerer can be connected before or after the other device. If connected before the other device, the cables run from the TAG IN and BUS IN connectors to the next device. If connected after, the cables run from the other device to the Multisourcerer's BUS OUT and TAG OUT connectors. The bar across the back of the Multisourcerer provides both a place to relieve the strain on the channel cables attached to the Multisourcerer and a convenient handhold.

14. **After connecting the Multisourcerer to the I/O channel, attach the signal cables, apply power, and turn the Multisourcerer on. Each type of application card comes with a manual that describes the pin out and signal levels expected by that card. Please refer to the appropriate manual for further information.**

Configuration

Use the front panel keypad and display to configure the Multisourcerer. You can save entered parameters in a Non-Volatile RAM (NVRAM). The Multisourcerer remembers them even when the power is off and automatically recalls them during the power-up cycle. Therefore, configuration need only be performed during the initial installation of the Multisourcerer or when the application cards are changed. Configuration can be done only when the Multisourcerer is in diagnostic mode.

Use the procedure below to configure the Multisourcerer.

1. **The General Purpose Channel Interface (GPCI) card, in the bottom slot of the Multisourcerer, has two switches located at AB1. Switch 1 performs no function. Switch 2 controls the Multisourcerer's mode (operational or diagnostic). To place the Multisourcerer in diagnostic mode, open the front panel and move switch 2 to OFF.**

2. **Close the front panel and power up the Multisourcerer. The display should show the following prompt:**

```
PRESS D
```

If it does not, press the RESET switch. This is a prompt to remind you that you are in diagnostic mode.

3. **Using the keypad,**

    Press: D

The display response is:

```
WHICH DIAG. RTN
```

which means which diagnostic routine do you want.

4. Configuration and diagnostic routines are selected by pushing the appropriate keys on the keypad. You can view a menu of the available routines one routine at a time by pressing the S key to scroll through them. Pressing the R key allows a reverse step to review the previous item. When you see the routine you want to perform, press the key with the character that appears with that routine. The following is an example.

You want to run the GPCI diagnostic tests.

Press: S

until you see the following routine displayed:

```
1 GPCI DIAG
```

To execute the diagnostics,

Press: 1

Several of the routines require additional information and prompt you to select an item from a submenu or to answer yes (Y) or no (N). If the choice requires a submenu, the S and R keys scroll and reverse through the submenu.

The routines currently available are:

Key	Routine
1	test GPCI *
2	device tests
4	go to the Operational Mode
5	set the channel base address
6	store parameters in the NVRAM
9	display a device's characteristics
A	add a device to the list
B	delete a device from the list
C	store the Checksum of the PROM

*You will need to do a reset following this test.

To configure the Multisourcerer, you must set the channel base address (5), add the devices that are to be installed in the Multisourcerer (A), delete any unused devices (B), and store these parameters in the NVRAM (6). Each of these routines is described in greater detail below.

5. A host computer can address up to 256 devices on an I/O channel. These devices have distinct addresses which, in hexadecimal, range from 00 to FF. We call the first digit the channel base address. We call the second digit the device address. All of the devices installed in a Multisourcerer must have the same base address. To set the base address,

   Press: **5**
   Press: *the key for the chosen base address*

6. To add a device to the Multisourcerer,

   Press: **A**

   The prompt asks you to select the type of device that you want added to the list of devices present in the Multisourcerer. You can use the S and R keys to help you recall the number for the type of device. Any application card placed in the Multisourcerer but not added to this list is ignored. The device type numbers assigned to various devices are:

Number	Device Type
0	Bisync Communication Card
1	Ingestors (see number 7 below)
3	ProNET Communication Card

7. If you choose an ingestor as the device type, the next prompt asks you to select the ingestor type. The ingestor type can be viewed by pressing the S key and scrolling through the following choices.

Ingestor	Submenu
1	GOES AAA Ingestor
2	POES Ingestor
4	METEOSAT Ingestor
5	GMS Ingestor
6	GVAR Ingestor

   When you choose the ingestor type, the I/O port address is displayed for verification that the ingestor is configured properly, e.g.:

```
2, I/O = 90.CONT?
```

   To verify that it is properly addressed,

   Press: **Y**

8. After you choose which device type to add to the list, you are asked to select the device address. Following this, there is one more prompt. It is:

```
RWDPE?
```

which means Report any Write Data Parity Errors. In an operational setting, this is answered yes.

Press: **Y**

9. During initial installation, each device in the range 0-E not added to the list should be deleted. To do this,

Press: **B**
Press: *the address of the device to be deleted*

After initial installation, only delete devices if they are removed from the system. A device on the list, but physically not present, is ignored.

10. The final step in configuring the Multisourcerer is saving the new parameters. To do this,

Press: **6**
Press: **Y**

If this is not done the Multisourcerer reverts to the previous configuration when it is reset, or when it is powered down and then powered up again.

11. To inspect the results of your configuring,

Press: **9**
Press: *the number of the device you wish to check*

The display will show the information in the following format:

```
# t aa yyyyy
```

# is the device number.
t is the device type (see the Device Type list above).
aa is the device I/O port.
yyyyy is RWDP or NRWDP (reports write data parity errors or no report of write data parity errors).

12. **When the configuration is complete, return the Multisourcerer to operational mode by moving switch 2 on the GPCI back to the ON position. Then do a reset.**

13. **Return to number 13 on page 2-1 for further installation instructions.**

Note For Configuration For Bisync

The Multisourcerer can communicate with a McIDAS workstation, using the Bisynchronous (Bisync) Communications Protocol, by placing SSEC-designed Bisync cards in the Multisourcerer and workstation. The card in the workstation replaces the two ProNET cards. There are several links in this communications path and they must be mapped correctly to allow data to be transferred between the host and workstation.

A workstation is referenced by its terminal number (e.g., Txx, where the xx is a two-digit decimal number) inside the McIDAS software running on the host. A systems programmer must map the Txx number to a channel address when generating McIDAS. Check with the systems programmer to get a list of this mapping.

A second mapping occurs inside the Multisourcerer. Here the channel address is converted to the physical port attached to the cable going to the workstation. The cable may be extended indefinitely with a pair of modems. This mapping occurs in two stages. The first stage is performed by the GPCI, which converts the channel address to MULTIBUS®* memory and I/O addresses. This is done automatically once the device is added to the list of those present in the Multisourcerer. The second stage is done on the specific Bisync card which communicates with the addressed workstation. This mapping is performed by attaching the appropriate jumpers on the card edge connector to J2. See McIDAS design note DN3504-026.

The front panel of the Multisourcerer has eight LEDs labeled 0-7. Each one represents the corresponding channel address and is lit when a data transfer occurs from that address. The port accessed at this time is determined by the position of the Bisync card responding to that address. You should follow the simplest mapping and assign address 0 to slot A, 1 to B, etc. You may also find it useful to fill out a chart similar to the one that follows.

Channel	Txx	Port	Location
0		A	
1		B	
2		C	
3		D	
4		E	
5		F	
6		G	
7		H	

*MULTIBUS is a registered trademark of Intel Corporation.

System Operation

System Operation

Multisourcerer operation is controlled from the front panel. In addition to the keypad and display, there are two switches on the front panel. One is a RESET switch which is a momentary switch. The other is a DISCONNECT switch which is a toggle switch. RESET resets the GPCI and all application cards present in the Multisourcerer. It works only when DISCONNECT is off. This prevents accidental resets of the Multisourcerer when it is connected to the channel. The red LED above the DISCONNECT switch is lit when the Multisourcerer is powered up and the DISCONNECT switch is off. The LED goes out when the DISCONNECT switch is turned on and the Multisourcerer is connected to the channel.

Programs that try to access the Multisourcerer when it is disconnected will report the message "NO PATHS AVAILABLE" on the operator's console. All programs that are using the Multisourcerer when it is connected to the channel should be halted before the Multisourcerer is disconnected. The Multisourcerer should always be disconnected from the channel before it is powered down.

The following four functions can be controlled from the keypad when the Multisourcerer is in the operational mode:

- view the first bank of devices
- change the ready/not ready state of an individual application card
- reset an individual application card
- view the second bank of devices

If accidentally selected, the last two functions are aborted by pressing the N key. You can view a menu of these functions by using the S (scroll) and R (reverse) keys. These functions are:

Key	Routine
0	view the first bank of devices (0-7)
1	change ready/not ready for a single device
2	reset a single device
4	view the second bank of devices (8-E)

Each Multisourcerer application card (also called a device) is assigned a channel address. The front panel of the Multisourcerer has eight green LEDs, labeled 0-7. Each one represents the device at the corresponding channel address. An LED is lit when a data transfer is occurring between the host and that device. When the 0 key is pushed, the Multisourcerer shows a summary of the state of each device on the front panel display. Each device is assigned two display characters. The first character pair corresponds to device 0, the second character pair corresponds to device 1, and so on.

The first character of the pair indicates which type of application card is assigned that address. The second character of the pair indicates the state of the device. The character codes are:

Character	Type F Device
B	Bisync
A	GOES AAA Ingestor
T	POES Ingestor
M	METEOSAT Ingestor
P	ProNET
G	GMS Ingestor
V	GVAR Ingestor

A device has one of the five states listed in the table below.

Character	Device State
R	ready
N	not ready
E	empty
*	signal present

If the device is present, and the Multisourcerer is connected to the channel, then the device may be either ready (R) or not ready (N). A device that is ready is operational and responds to any legal command issued to it by performing the requested function. A device that is not ready also responds to commands, but indicates that operator intervention is required.

GPCI Overview

GPCI Overview

The General Purpose Channel Interface (GPCI) is a bus-to-bus converter. It provides the interface between the IBM System 370 I/O bus and the Intel Multibus. The GPCI is the main subassembly of the Multisourcerer. The GPCI design is sufficiently general and flexible to control the current generation of SSEC I/O devices. The design of future I/O devices developed at SSEC will match the standard established with the Multisourcerer.

Refer to Figure 2, the Multisourcerer Block Diagram, on the next page. The Channel Drivers and Receivers board is a printed circuit card. The card is vertically mounted on the Multisourcerer's upper-left panel. Functionally, this card is an extension of the GPCI. The card consists primarily of line drivers and receivers, and a small amount of control circuitry. The reasons for making a separate board for these circuits are:

- it frees up enough board space on the GPCI to allow all remaining GPCI circuits to be located on a single Multibus form-factor wire wrap board
- it allows required IBM channel I/O control hardware to remain intact, even when the GPCI is removed
- it contains circuitry that is design stable and unlikely to change

For additional information on the Channel Drivers and Receivers card, refer to Chapter 10.

As shown in Figure 2, the GPCI consists of two main sections, the I/O Interface Controller and the Device Controller. Dual processors control the GPCI. A bit-slice processor controls channel interface functions, and an 8085 microprocessor controls device functions. The design choice for the I/O Interface Controller section is a bit-slice processor because its speed is inherently higher than the channel's. Thus, the Multisourcerer can make use of the full bandwidth of the IBM channel. To the remainder of the GPCI and the Multibus, the I/O Interface Controller makes the IBM channel appear as an I/O port.

The Device Controller Section is based on an 8085 microprocessor design. The Device Controller can control up to six Multibus compatible devices (GPCI occupies the bottom slot of a 7-slot card cage). The Device Controller Section also provides operator interaction via a 20-key keypad and a 16-character single line display. The RS232 port allows you to use a monitor and keyboard during diagnostics and self testing. The port can also be used for maintenance troubleshooting.

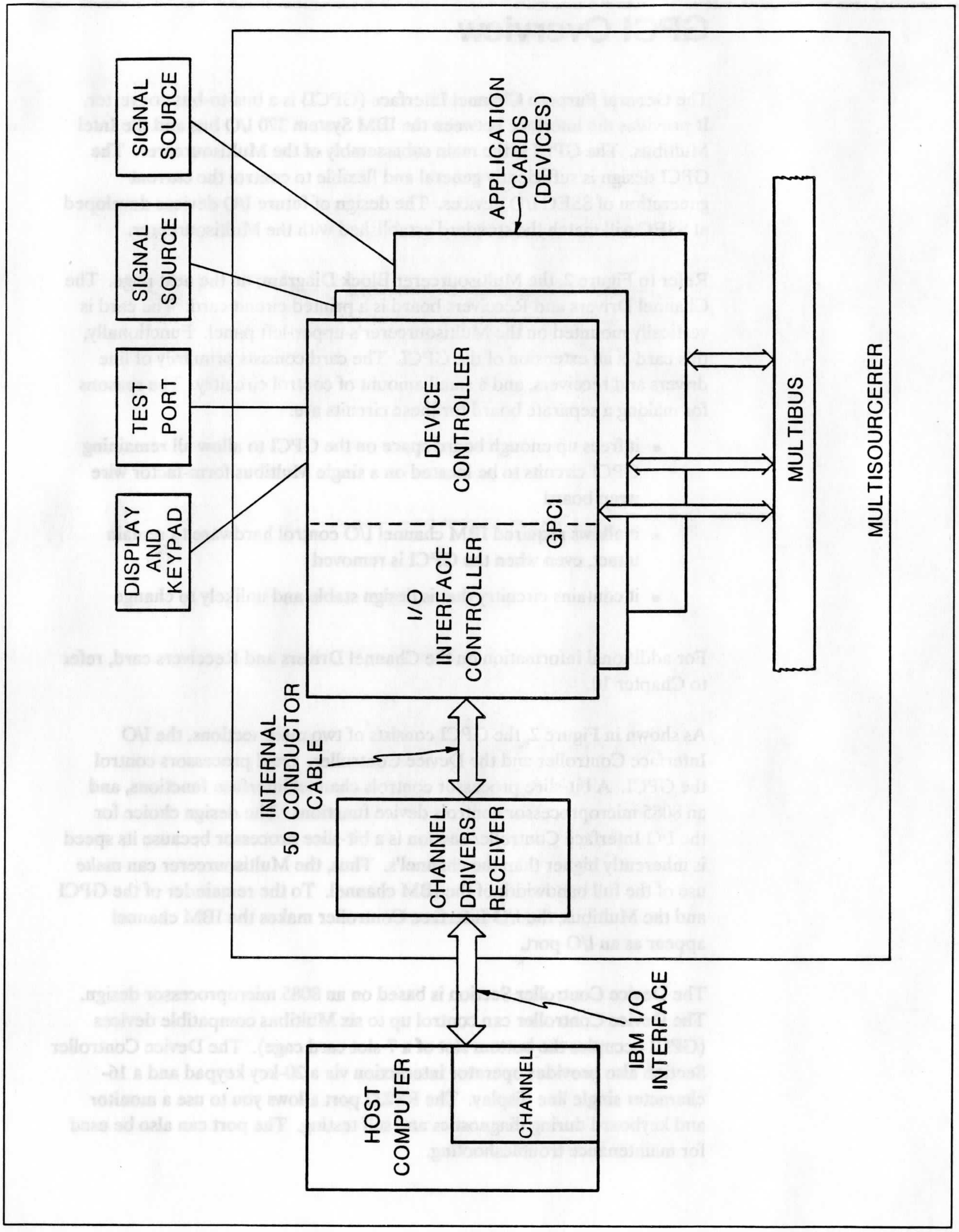


Figure 2. Multisourcerer Block Diagram

Because of the complexity of the GPCI, it is divided into four subsequent chapters. These include:

- Chapter 5 - GPCI Basic Functional Description (based on Figure 3)
- Chapter 6 - GPCI Detailed Functional Description (based on Figure 4)
- Chapter 7 - GPCI Detailed Circuit Description
- Chapter 8 - GPCI Bit-Slice Microcode Description

Chapters 5, 6 and 7 provide successively more detailed descriptions of the GPCI.

An understanding of the GPCI without an understanding of the IBM I/O interface requirements and operation would be difficult if not impossible. The *IBM System/360 and System/370 I/O Interface Channel to Control Unit Original Equipment Manufacturers' Information Manual* (hereafter referred to as the I/O Reference Manual) describes the interface requirements in detail.

The GPCI and its application cards function as an IBM "control unit" (type-3) and "devices" respectively. The I/O Channel Interface has several optional features that are not employed in the GPCI design. For example, the GPCI does not employ data streaming or the bus extension features. Data In and Data Out tag lines are very similar in operation, timing, and use to Service In and Service Out respectively. The GPCI uses only Data In and Data Out. However, Service In and Service Out are connected to the GPCI via the Channel Drivers and Receivers board. Section 3 of the I/O Reference Manual describes Data In and Data Out. These are the only features discussed in section 3 that apply to the GPCI. The GPCI does not use Metering In and Metering Out.

GPCI Basic
Functional Description

GPCI Basic Functional Description

The IBM interface system is a fully interlocked handshaking system. That is, when the channel or control unit (i.e., GPCI) raises a control signal, that signal is not allowed to fall until its receipt is acknowledged by the recipient's raising of another control signal. In turn, that acknowledge signal cannot drop until it is acknowledged, etc. The advantage of this interface is automatic matching of the channel speed to a wide range of data rates.

I/O Interface Controller

The I/O Interface Controller consists of:

- **Bit-Slice Processor**
- **Channel Input and Output Latches**
- **Condition Code Multiplexer**

Bit-Slice Processor

Refer to Figure 3 on page 5-3. The I/O Interface Controller is built around the Bit-Slice Processor. This block functions as a high speed (100 nanosecond instruction cycle time) 4-bit microprocessor. It is controlled by approximately 470 40-bit micro-instructions which are stored in PROMs.

Channel Input and Output Latches

The 8085 microprocessor, its associated buses, the IBM channel, and the Multibus all handle data in byte-wide formats. Therefore, all data paths between these sections or buses and the I/O Interface Controller require data latches for assembling and disassembling data.

The Channel Input Latches assemble the 4-bit words from the Bit-Slice Processor into 8-bit words. The Channel Output Latches disassemble the 8-bit channel Data Bus data into 4-bit words and pass them to the Bit-Slice Processor.

The I/O Interface Controller generates the Tag and Control signals by assembling 8-bit control words in the Channel Input Latch and transmitting them to the Channel Drivers and Receivers (CDR) board via the Octal Transceiver block. Two latches on the CDR board have their inputs wired in parallel. They are the Output Data Latch, and the Tag and Control Latch. The appropriate latch is selected and clocked by the I/O Interface Controller via the CDR Control lines. Thus, most Tag In and Control In lines originate on the CDR board.

Condition Code Multiplexer

The I/O Interface Controller monitors channel control (Control Out) signals via the Condition Code Multiplexer. This block monitors several other signals, that are not shown in Figure 3. These signals are discussed in the GPCI Detailed Circuit Description (Chapter 7).

Device Controller

The Device Controller consists of three sections:

- Microprocessor Section
- Shared Resources Section
- Multibus Interface Section

Microprocessor Section

The primary function of the Microprocessor Section is to act as an interface between the application devices and the I/O Interface Controller. In addition, it passes device status to the I/O Interface Controller, provides DMA control, displays status, and allows operator interaction via the keyboard. It also performs diagnostic testing on many GPCI sections and application cards.

The Microprocessor Section is conventional in design. It is an 8085 microprocessor based system, containing internal address and data buses which connect the microprocessor to its support components.

The Microprocessor Section is controlled by the program residing in its 16K byte EPROM (optionally 32K bytes). It contains a 1K byte static RAM for storing variables such as device state tables, and a nonvolatile RAM (NVRAM) for storing Multisourcerer configuration variables. The NVRAM retains its data even when power is off. The configuration variables are altered by the operator via the keyboard.

The I/O control block provides an interface between the microprocessor, keyboard, serial test port and display.

The Register File serves as a link between the I/O Interface Controller and the Device Controller. It passes status and device state information from one section to the other. To the Bit-Slice Processor, the Register File appears as a 64-nibble RAM (remember the I/O Interface Controller is a nibble-wide system). From the microprocessor's point of view, the Register File appears as a 32-byte RAM.

The Bit-Slice Processor passes device specific information to the Microprocessor Section by writing the data into the proper locations (each device has 4 nibbles assigned to it) in the Register File. Next, the device's assigned address is loaded into a specified address of the Register File. Finally, the Bit-Slice Processor interrupts the microprocessor, telling it to look in the Register File. When the microprocessor is interrupted, it retrieves the device's address and uses it as a pointer to locate the new information in the Register File. Data is passed from the Microprocessor Section to the Bit-Slice Processor by writing the data into the proper Register File location. The Bit-Slice Processor locates new data by sequentially checking each device's register set for new data, as there is no true interrupt capability for the Bit-Slice Processor.

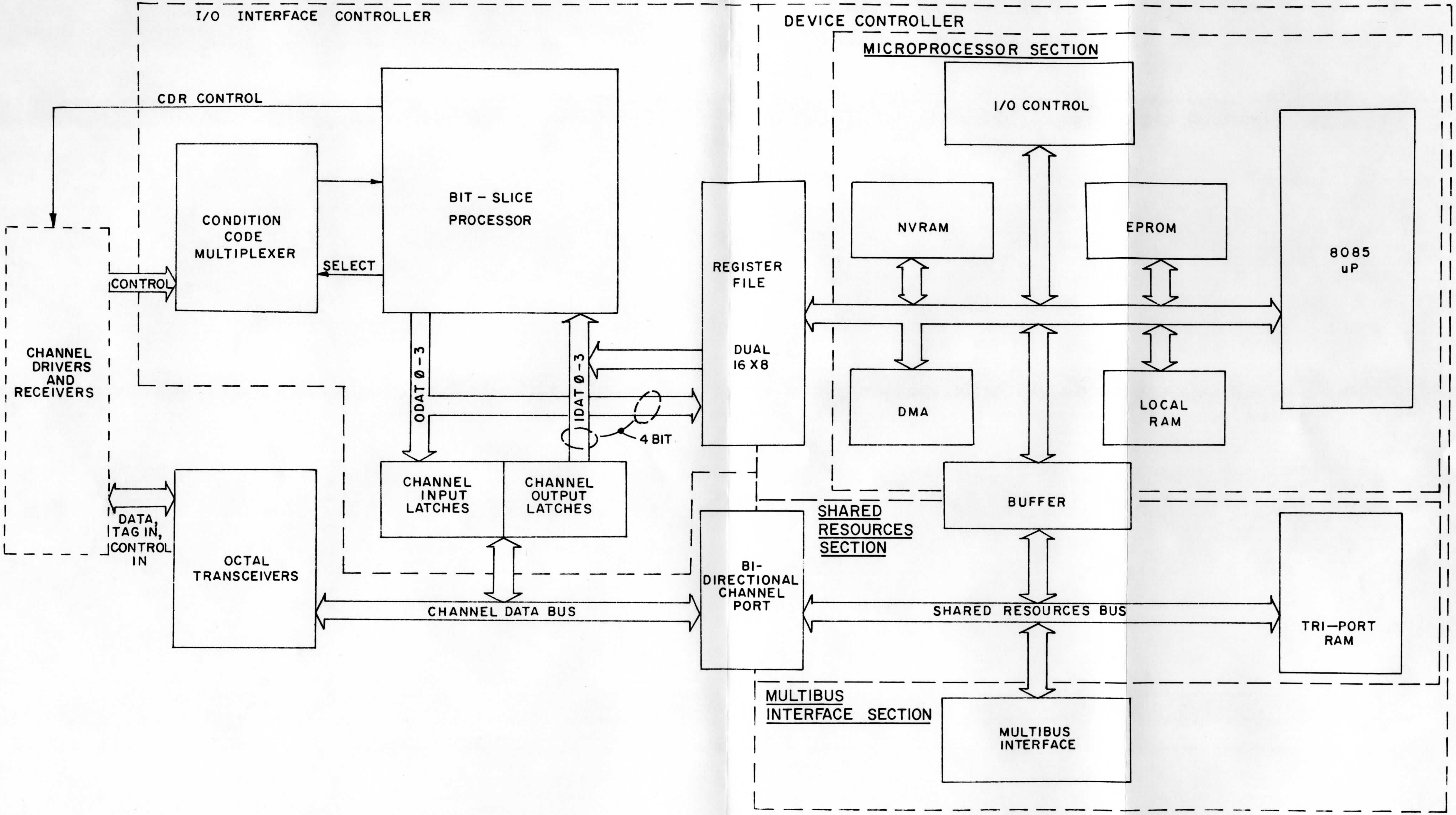


Figure 3. GPCI Basic Block Diagram

In summary, the Register File provides the following:

- bus width conversion
- status and device state temporary storage
- data passing (status and device states) between two unsynchronized systems
- data sharing between two systems operating at widely different data rates

Shared Resources Section

The Shared Resources Bus connects the Tri-Ported RAM, the Multibus, the Channel, and the Microprocessor Section together. Each of these sources contains a tri-state bus transceiver or is a tri-state device. At any one time, only one data source and one data sink use the bus. Before data can be transferred over the bus, the appropriate transceivers must be enabled, and the data flow direction through those transceivers must be set. The transceivers are controlled by bus arbiters and associated control hardware. Because of the large number of users on the bus, the bus arbitration and control logic is complex. Table 1 below shows the valid Shared Resources Bus Configurations.

From	To	Domain
Microprocessor	Channel Port	I/O
Channel Port	Microprocessor	I/O
Microprocessor	Tri-Port RAM	Mem
Tri-Port RAM	Microprocessor	Mem
Microprocessor	Multibus	I/O
Multibus	Microprocessor	I/O
Microprocessor	Multibus	Mem
Multibus	Microprocessor	Mem
Channel Port	Tri-Port RAM	DMA Mem
Tri-Port RAM	Channel Port	DMA Mem
Channel Port	Multibus	DMA I/O
Multibus	Channel Port	DMA I/O
Channel Port	Multibus	DMA Mem
Multibus	Channel Port	DMA Mem
Multibus	Tri-Port RAM	Mem
Tri-Port RAM	Multibus	Mem
Multibus*	Channel Port	Mem
Channel Port*	Multibus	Mem

* Channel Port masquerades (from the Multibus' view) as a memory block which is contiguous with the Tri-Port RAM.

Table 1. Shared Resources Bus Configuration

The Tri-Port RAM is a 4K byte static RAM. Its address bus can be driven by the Multibus, the DMA Address Generator, or the microprocessor's address bus. This is why it is called the Tri-Port RAM. The microprocessor and the DMA Generator address the Tri-Port RAM via addresses 7000H-7FFFH. During DMA transfers, the DMA Generator controls the Tri-Port RAM's read/write addressing, and must not exceed address 7FFFH. Multibus/Tri-Port RAM data transfers occur only when the GPCI is acting as a Multibus slave, using Multibus addresses 1000H-1FFFH to access the RAM.

The Bidirectional Channel Port block links the Shared Resources Bus, located in the Device Controller Section, to the Channel Data Bus. The Bidirectional Channel Port functions as dual back-to-back octal tri-stated latches. Incoming channel data or outgoing I/O Interface Controller data is latched into the Bidirectional Channel Port, and is readable by the Shared Resources Bus. Once on the Shared Resources Bus, this data is available to the Microprocessor Section, the Tri-Port RAM or the Multibus. For data passing from the Shared Resources Bus to the channel or Bit-Slice Processor, the data is latched by the data originator (Multibus or microprocessor) and read by the recipient (Bit-Slice Processor or the channel).

Multibus Interface Section

The Multibus Interface block allows bidirectional communication with the application cards (six maximum due to chassis limitation).

GPCI Detailed
Functional Description

GPCI Detailed Functional Description

The GPCI Functional Block Diagram is shown in Figure 4 on page 6-3. This figure is an expansion of Figure 3 and introduces the main control blocks. Note that the schematic page numbers are listed in each block. This should make the transition from the functional descriptions to the schematic diagrams easier.

I/O Interface Controller

As shown in Figure 4, the I/O Interface Controller consists of the following blocks:

- Bit-Slice Processor
- Channel Input and Output Latches
- Condition Code Multiplxer
- Command PROM Latch and Decode
- Register File Address Generator
- Register File
- Clock

Each of these is described below.

Bit-Slice Processor

The heart of the I/O Interface Controller Section is the Bit-Slice Processor. Figure 5 on page 6-6 is an expansion of this block. The Bit-Slice Processor consists of the Microsequencer, Microcode and Bit-Slice Arithmetic Logic Unit (ALU) blocks. These three blocks comprise a 4-bit microprocessor with a pipeline architecture. That is, all three blocks are cascaded and clocked by a common clock. Thus, the Microsequencer is always outputting the address that will be used by the Microcode block on the next clock pulse (the Microcode block has a registered output). The Microcode block's output is one clock pulse ahead of the ALU output. In this way, the Microsequencer, the Microcode block and the ALU form an instruction pipeline.

The Microsequencer is an address sequencer/selector that controls the execution sequence of the micro-instructions stored in the Microcode memory. The Next Instruction bus selects one of sixteen Microsequencer instructions. Each instruction can select one of four address sources. These sources are:

- direct input (ADDAT0--ADDAT8) which is used for conditional jumps and calls
- microprogram address register which usually contains an address one greater than the previous address
- a register/counter which retains data loaded during a previous micro-instruction (not presently used)
- a nine-deep last-in first-out (LIFO) stack which provides return address linkage for subroutine returns and looping

The Microcode block consists of five cascaded 1024-byte PROMs. The cascaded PROMs feature output data latches which are clocked by the Bit-Slice clock. The 40 output bits are used as follows:

- 9 bits for Bit-Slice ALU instructions
- 4 bits for Microsequencer instructions
- 9 bits for Bit-Slice Controller jump addresses/data
- 8 bits for ALU register addressing
- 1 bit for I/O control enable
- 3 bits for Channel Drivers and Receivers (CDR) control
- 5 bits for miscellaneous control
- 1 bit is unused

The Bit-Slice ALU is a 4-bit microprocessor slice and can be cascaded to any number of bits in groups of four bits. The ALU can perform addition, subtraction, and five logic functions, though only the ADD and three logic functions (AND, OR and EXOR) are currently used.

The ALU has 16 internal RAM registers. Each RAM register can be selected as a data source for an instruction. Also, zero can be selected as a data source as well as the data-in bus (IDAT0-IDAT3). The data sources, ALU function, and data destination are selected via outputs from the Microcode block. The ALU also generates four status flags, carry, overflow, zero and negative accumulator results. At present, only the zero flag is used. The ALU instruction has a data destination code embedded in it. The data can be written to the data out bus (ODAT0-ODAT3), one of the RAM registers, or to an internal shift register, which is not presently used.

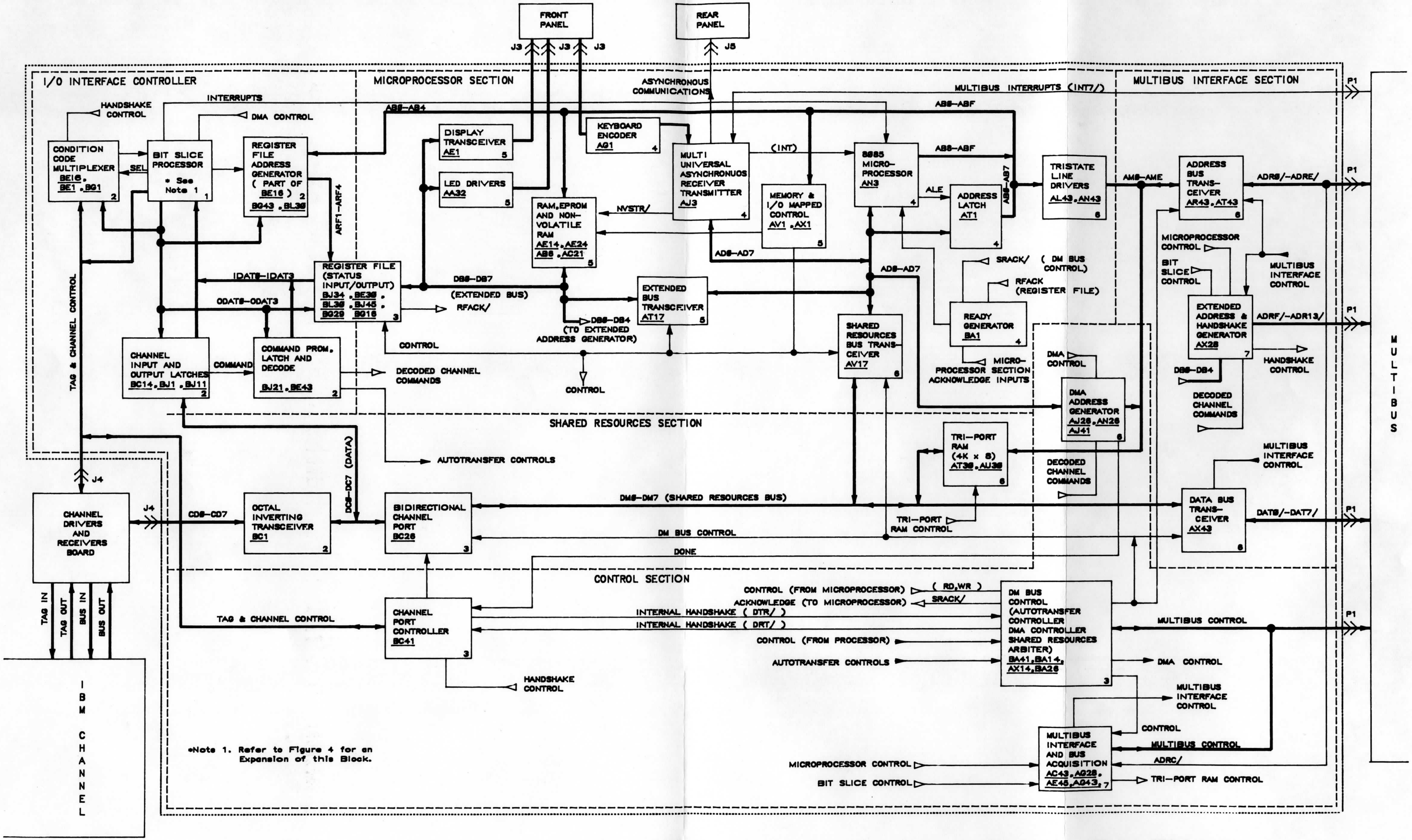

FRONT PANEL
REAR PANEL
J3
J3
J3
J5
I/O INTERFACE CONTROLLER
MICROPROCESSOR SECTION
MULTIBUS INTERFACE SECTION
HANDSHAKE CONTROL
INTERRUPTS
DMA CONTROL
AB8-AB4
ASYNCHRONOUS COMMUNICATIONS
MULTIBUS INTERRUPTS (INT7/)
AB8-ABF
P1
CONDITION CODE MULTIPLEXER BE16, BE1, BG1 2
SEL
BIT SLICE PROCESSOR * See Note 1 1
REGISTER FILE ADDRESS GENERATOR (PART OF BE16) 2 BG43, BL36
DISPLAY TRANSCEIVER AE1 5
LED DRIVERS AA32 5
KEYBOARD ENCODER AG1 4
MULTI UNIVERSAL ASYNCHRONUOS RECEIVER TRANSMITTER AJ3 4
(INT)
8085 MICRO-PROCESSOR AN3 4
AB8-ABF
ALE
ADDRESS LATCH AT1 4
AB8-AB7
TRISTATE LINE DRIVERS AL43, AN43 6
AM8-AME
ADDRESS BUS TRANS-CEIVER AR43, AT43 6
ADR8/-ADRE/
RAM, EPROM AND NON-VOLATILE RAM AE14, AE24, AB8, AC21 5
NVSTR/
MEMORY & I/O MAPPED CONTROL AV1, AX1 5
ARF1-ARF4
AD8-AD7
AD8-AD7
MICROPROCESSOR CONTROL
BIT SLICE CONTROL
MULTIBUS INTERFACE CONTROL
SRACK/ (DM BUS CONTROL)
RFACK (REGISTER FILE)
IDAT8-IDAT3
ODAT8-ODAT3
REGISTER FILE (STATUS INPUT/OUTPUT) BJ34, BE38, BL36, BJ45, BG29, BG16 3
DB8-DB7
(EXTENDED BUS)
RFACK/
DB8-DB4 (TO EXTENDED ADDRESS GENERATOR)
EXTENDED BUS TRANSCEIVER AT17 5
READY GENERATOR BA1 4
EXTENDED ADDRESS & HANDSHAKE GENERATOR AX28 7
ADRF/-ADR13/
HANDSHAKE CONTROL
TAG & CHANNEL CONTROL
CHANNEL INPUT AND OUTPUT LATCHES BC14, BJ1, BJ11 2
COMMAND
COMMAND PROM, LATCH AND DECODE BJ21, BE43 2
DECODED CHANNEL COMMANDS
CONTROL
CONTROL
SHARED RESOURCES BUS TRANS-CEIVER AV17 6
MICRO-PROCESSOR SECTION ACKNOWLEDGE INPUTS
DMA CONTROL
DMA ADDRESS GENERATOR AJ28, AN28, AJ41 6
DB8-DB4
DECODED CHANNEL COMMANDS
M U L T I B U S
SHARED RESOURCES SECTION
TRI-PORT RAM (4K x 8) AT38, AU38 6
DECODED CHANNEL COMMANDS
MULTIBUS INTERFACE CONTROL
AUTOTRANSFER CONTROLS
J4
J4
CHANNEL DRIVERS AND RECEIVERS BOARD
CD8-CD7
OCTAL INVERTING TRANSCEIVER BC1 2
DC8-DC7 (DATA)
BIDIRECTIONAL CHANNEL PORT BC28 3
DM8-DM7 (SHARED RESOURCES BUS)
DATA BUS TRANS-CEIVER AX43 6
DAT8/-DAT7/
TRI-PORT RAM CONTROL
DM BUS CONTROL
DONE
CONTROL SECTION
TAG IN
TAG OUT
BUS IN
BUS OUT
CONTROL (FROM MICROPROCESSOR)
(RD,WR)
ACKNOWLEDGE (TO MICROPROCESSOR)
SRACK/
INTERNAL HANDSHAKE (DTR/)
INTERNAL HANDSHAKE (DRT/)
CONTROL (FROM PROCESSOR)
AUTOTRANSFER CONTROLS
TAG & CHANNEL CONTROL
CHANNEL PORT CONTROLLER BC41 3
DM BUS CONTROL (AUTOTRANSFER CONTROLLER DMA CONTROLLER SHARED RESOURCES ARBITER) BA41, BA14, AX14, BA28 3
MULTIBUS CONTROL
DMA CONTROL
MULTIBUS INTERFACE CONTROL
HANDSHAKE CONTROL
CONTROL
I B M C H A N N E L
*Note 1. Refer to Figure 4 for an Expansion of this Block.
MULTIBUS INTERFACE AND BUS ACQUISITION AC43, AG28, AE48, AG43, 7
MULTIBUS CONTROL
ADRC/
MICROPROCESSOR CONTROL
BIT SLICE CONTROL
TRI-PORT RAM CONTROL

Figure 4. GPCI Functional Block Diagram

Channel Input and Output Latches

This block makes the 4-bit data in and data out buses of the I/O Interface Controller compatible with the 8-bit parallel data buses of the channel and microprocessor. All controls (latch clocks and output enables) for this block are generated within the I/O Interface Controller.

Condition Code Multiplexer

The Condition Code Multiplexer functions as a 1-of-22 line selector. The output of the multiplexer is connected to the CC/ input of the Microsequencer and affects the Microsequencer during its execution of a conditional instruction. If CC/ is low, the conditional instruction is executed. If CC/ is high, the next sequential instruction is fetched.

The Condition Code Multiplexer is controlled by six of the eight ALU register addressing lines from the Microcode block. Internally, the multiplexer presents the true and the inverted version of each signal to the selector logic. In addition, "Always" and "Never" are internally generated inputs to the selector logic. These conditions can be used to convert conditional instructions (i.e., conditional jump) into unconditional instructions (i.e., jump always or never jump).

Command PROM Latch and Decode

Commands from the channel are received by the GPCI during Initial Selection Sequences (refer to the I/O Reference Manual). Different Shared Resources Section bus hardware configurations may be required for the same channel command if directed to different devices. Therefore, the command must be converted to a device specific command by the Command PROM Latch and Decode block.

Prior to sending the command, the channel sends the base and device addresses to the Bit-Slice Processor. The Bit-Slice Processor stores the device address in one of its internal RAM registers. During the time that the command is valid on the channel data bus (DC0-DC7), the command is latched into the Channel Input Latches. When the command is sensed by the Bit-Slice Processor, it fetches the device address and uses it to read the "Device Type" code from the Register File. Together, these two inputs drive the 12 address lines of the Command PROM. The PROM is programmed to generate a 4-bit output code used by the I/O Interface Controller and Device Controller to configure hardware necessary for execution of the command. The microprocessor also uses the command to determine a specific sequence of instructions which, when issued to the device, causes the device to execute the command.

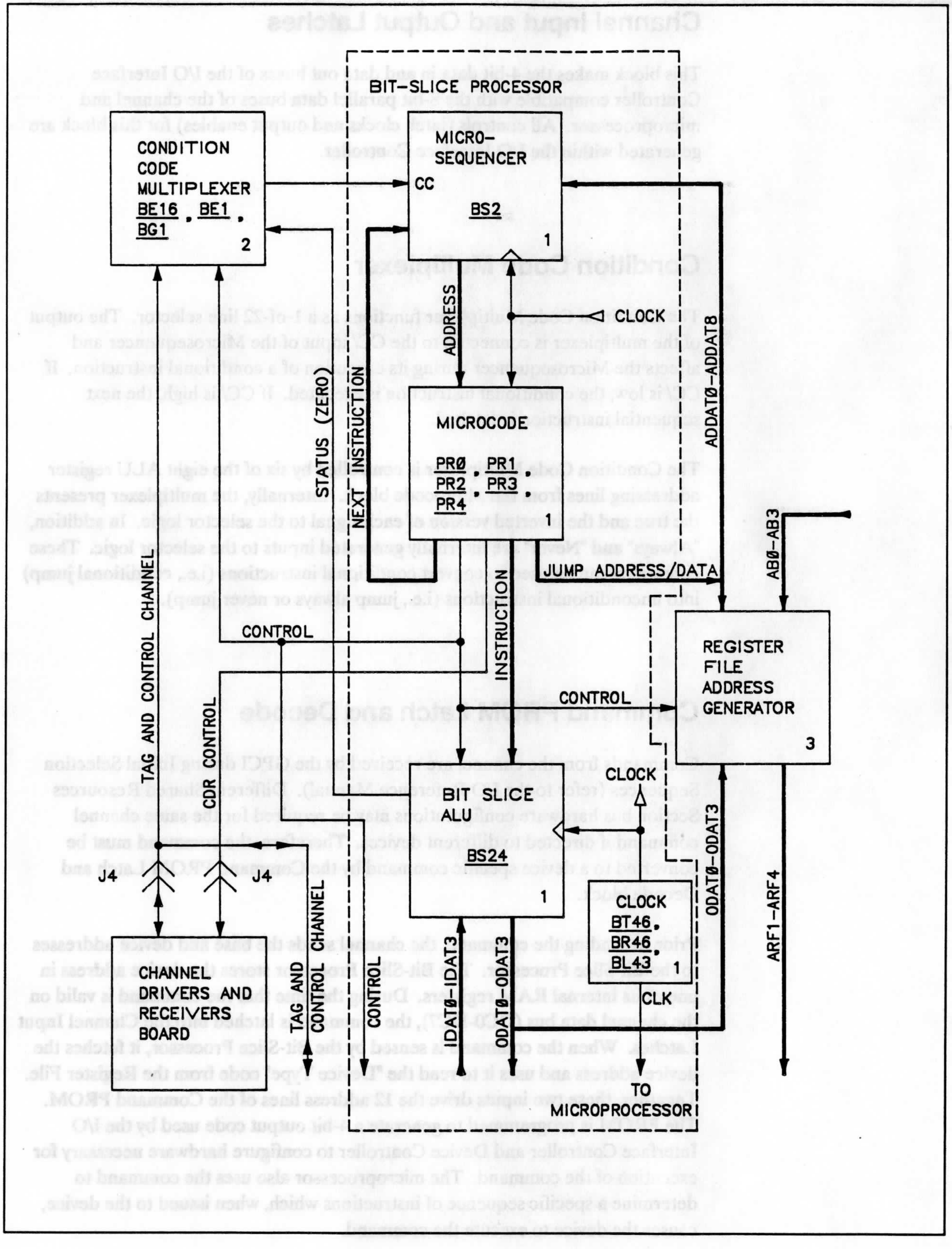


Figure 5. Bit-Slice Processor Block Diagram

Register File Address Generator

The Register File Address Generator is an address bus selector controlled by a bus arbiter. The Bit-Slice Processor and the Microprocessor Sections supply address information to the selector section of the Register File Address Generator. The arbiter section of the Register File Address Generator monitors several inputs from the Bit-Slice Processor and the Microprocessor Section to detect requests for the Register File. It resolves contention during simultaneous requests for the Register File in favor of the Bit-Slice Processor. Once control is granted, the section in control maintains control until its read or write evolution is completed.

Register File

The Register File consists of a 64-nibble static RAM, data multiplexers and latches. The function of the Register file is 2-way status passing between the I/O Interface Controller and Device Controller.

Though the Multisourcerer has slots for up to six applications cards (devices), the GPCI can control a maximum of 15 devices. Thus, the Register File requires 15 storage areas. A 16th storage area is provided for passing GPCI specific data between the bit-slice side and the microprocessor side of the GPCI. Each storage area consists of two 2-nibble blocks (four nibbles total). Nibbles 0 and 1 form block 0; nibbles 2 and 3 form block 1; ... nibbles 60 and 61 form block 30 (1EH); and nibbles 62 and 63 form block 31 (1FH). Each device installed in the GPCI is assigned a hexadecimal "device address." The device's address forms the least significant hexadecimal digit of that device's assigned Register File nibble-pair address. For example, device address 7 is assigned nibble blocks 07 and 17, and device address AH is assigned nibble blocks 0AH and 1AH, etc.

Nibble pairs map into bytes and vice versa during Register File writes and reads by the microprocessor. The lower addressed nibble forms the four LSBs of the byte. Data multiplexers and data latches, under control of the Register File Address Generator, assemble and disassemble the microprocessor data bytes. When controlled by the microprocessor, the lower five microprocessor address bits select the desired nibble pair. The Bit-Slice Processor selects the desired nibble by writing the device address to the Register File Address Generator and specifying one of the four possible nibbles via two address line inputs from the Jump Address/Data bus.

Clock

The clock generator consists of an 18.4 MHz crystal oscillator, frequency dividers and buffer amplifiers. The 18.4 MHz is buffered and used by the I/O Interface Controller and the Device Controller. Also, the 18.4 MHz is supplied to a divide-by-two counter and a divide-by-six counter. These counters produce a 9.216 MHz and a 3.07 MHz clock output. Both sides use the 9.216 MHz clock, while only the Microprocessor Section uses the 3.07 MHz clock.

Device Controller

The Device Controller, as shown in Figure 4 on page 6-3, consists of these four sections:

- Microprocessor Section
- Shared Resources Section
- Multibus Interface Section
- Control Section

Microprocessor Section

The Microprocessor Section is conventional in design. It is based on the INTEL 8085 microprocessor. Refer to the Microprocessor Section of Figure 4 on page 6-3 for the following discussion.

Microprocessor

The INTEL 8085 microprocessor hardware is documented in INTEL's *Microprocessor and Peripheral Handbook*. The *INTEL 8080/8085 Assembly Language Programming Manual* provides a good programming reference for the 8085. Therefore, the documentation here is limited to that which is necessary to discuss the remaining Microprocessor Section components.

Address Latch

The 8085 contains a dedicated 8-bit bus for the upper eight address lines (AB8-ABF). The lower eight address bits are multiplexed with data and output on the bidirectional address/data bus (AD0-AD7). The Address Latch demultiplexes the lower eight address bits using ALE as an "address valid" strobe. The output of the Address Latch is the least significant 8 bits of the address bus. The output of the latch joins the upper eight address bits to form the 16-bit wide address bus (AB0-ABF) which drives the address inputs of the:

- Scratch Pad RAM
- Non-volatile RAM (NVRAM)
- EPROM
- Register File
- Memory and I/O Mapped Control

Memory and I/O Mapped Control

This block produces 17 control signals for functional blocks of memory and I/O addresses. Nine of these signals are memory mapped and eight are I/O mapped. In turn, some of these control signals are inputs to other control generators, further expanding the memory and I/O mapping. The inputs to this block are the microprocessor address lines, and several microprocessor control lines (RD, WR, IO/M, MS0 and MS1).

RAM, EPROM and Non-Volatile RAM

These memory components are addressed by the microprocessor's address bus and have their tri-stated outputs enabled by enable outputs from the Memory and I/O Mapped Control block. The tri-stated outputs from the memory components are connected to the Extended Bus (DB0-DB7) which is connected to the microprocessor's data bus (AD0-AD7) via the Extended Bus Transceiver.

The RAM is a static 1K byte RAM which stores variables such as device state tables, etc. This RAM is also the Scratch Pad RAM.

The EPROM is currently a 16K byte EPROM. It stores the program (firmware) that the microprocessor executes. It also stores program constants and tables. The current version of the 8085 program requires nearly all of the 16K bytes of storage. The 16K byte EPROM can be replaced with a 32K byte EPROM simply by replacing the plug-in EPROM and moving a jumper. This will prevent major GPCI board changes due to possible future firmware changes.

The Non-Volatile RAM (NVRAM) provides permanent storage of Multisourcerer configuration variables. Internally, the NVRAM consists of a RAM mirrored by an EEPROM (Electrically-Erasable-Programmable-Read-Only Memory). During configuration of the Multisourcerer, the operator enters certain device specific parameters via the keyboard. The microprocessor reads the keyboard via the Keyboard Encoder and writes the data to the RAM section of the NVRAM. At this time, if power is lost, the newly entered data is lost. To prevent data loss, the RAM must be copied into the EEPROM section. This is accomplished by strobing a special storage pin on the NVRAM. This pin is driven by an output from the MUART (NVSTR/). In turn, the MUART is controlled by the microprocessor.

When the sequence described in the Configuration Chapter (Chapter 2) is executed, the microprocessor issues the EEPROM store command. During power-up, the EEPROM is downloaded into the RAM section of the NVRAM, making the configuration data available for microprocessor use. Thus, configuration need only be performed during initial installation or when application cards are changed.

Display Transceiver

The Display Transceiver connects the 16-character vacuum fluorescent display module to the 8-bit Extended Bus. The display module is intelligent. That is, it has its own on-board dedicated microprocessor allowing data to be written to the display in an ASCII format. Character entry position is programmed by sending a cursor position to the module. The Display Transceiver provides a read/write path between the microprocessor and the module. At present, the display read feature is not used.

Keyboard Encoder

The Keyboard Encoder sequentially drives the column wires on a row and column wired keypad while simultaneously monitoring the row wires. When a key is pressed, an intersection between a row and a column wire is created, activating one of the row wires. The Keyboard Encoder contains decoding logic to determine which key is pressed, based on the active row and column combination. The Keyboard Encoder encodes the intersection into a 5-bit binary code and generates an interrupt. Both the interrupt and the keycode are sent to the MUART. The MUART multiplexes the interrupt onto the INT interrupt line to the microprocessor. When the microprocessor is interrupted, it queries the MUART and determines that a keycode is present. The microprocessor then requests the keycode from the MUART.

MUART

The MUART (Multi-Universal Asynchronous Receiver Transmitter) block:

- provides an interface between a serial RS232 test port (J5) and the microprocessor's data bus
- provides an interface between the Keyboard Encoder's 5-bit parallel output and the microprocessor's data bus
- multiplexes the Multibus interrupt signal, INT7/, with internally generated interrupts and applies the multiplexed interrupt signal to the interrupt input of the microprocessor
- contains five microprocessor programmable counter/timers (can be cascaded)
- contains an on-board baud rate generator, programmable for 13 common baud rates
- drives the non-volatile store pin on the NVRAM

LED Drivers

Eight front panel mounted LEDs provide device activity monitoring. The LEDs correspond with the device address. That is, LED0-LED7 represent device addresses 0-7 respectively. If LED7 is on continuously, it indicates that LED0-LED6 represent device addresses 8H-EH (8-14 decimal). The operator can select which bank of devices (upper or lower) is monitored by the LEDs via the front panel keyboard.

Extended Bus Transceiver

The Extended Bus Transceiver is the connecting link between the DB bus (Extended Bus) and the AD bus (internal/external microprocessor data bus). The Extended Bus Transceiver is always enabled, and its data direction is controlled by the Memory and I/O Mapped Control block. Refer to Figure 4 on page 6-3. The Extended Bus Transceiver transfers data from left to right (from the DB bus to the AD bus) only while reading the EPROM, RAM, NVRAM, Register File and Display. At all other times, the Extended Bus Transceiver transfers data from the AD bus to the DB bus.

Shared Resources Bus Transceiver

This transceiver is similar to the Extended Bus Transceiver above, except it links the microprocessor's AD bus to the Shared Resources Bus. Data transfer direction is controlled by an output from the Memory and I/O Mapped Control block. The Shared Resources Bus Transceiver transfers data from the Shared Resources Bus to the AD bus when reading the Tri-Port RAM or the channel port. At all other times, the transceiver transfers data from the microprocessor to the Shared Resources Bus. For this reason, this transceiver's tri-state output is set to the high impedance state at all times except when the microprocessor is actually writing data to the Shared Resources Bus. The transceiver's tri-state control is driven by an output from the DM Bus Control block.

Address Bus Transceiver

The Address Bus Transceiver block is driven by the microprocessor's AB0-ABE address lines. The output of this block is connected to the AM bus (AM0-AME), which is the Shared Resources Address bus. This bus can be driven by the microprocessor (via the Address Bus Transceiver), the DMA Address Generator (refer to the DMA Address Generator description below), or the Multibus (via the Address Bus Transceiver). The AM bus can provide addressing for the Tri-Port RAM and the Multibus. All address sources that share the AM bus must be tri-state devices, as only one address source is allowed to drive the address bus at one time. The Address Bus Transceiver's tri-state control is driven by an output from the DM Bus Controller block.

Ready Generator

The 8085 microprocessor is connected to a wide range of circuits. These circuits include:

- all application cards (devices), via the Multibus
- Tri-Port RAM
- IBM channel
- NVRAM
- Scratch Pad RAM
- EPROM
- Display
- MUART
- LED Drivers
- Register File

These devices have a wide range of response times to the microprocessor's read or write commands. The microprocessor has an input pin labeled RDY (Ready). If this input is low during a microprocessor read or write cycle, the microprocessor waits until that line goes high before completing the cycle. Thus, slow circuits can force the microprocessor to wait until they have performed the read or write process, by holding the RDY pin low. The Ready Generator combines the individual "ready" responses from the circuits listed above and produces a single output to the RDY pin on the microprocessor.

Shared Resources Section

Tri-Port RAM

The Tri-Port RAM is a 4K byte static RAM. Its input/output data bus is connected to the Shared Resources Bus, and its address bus is connected to the AM bus. Thus, the RAM is a public storage device. The Multibus, channel port, and microprocessor can read and write data from and to the RAM. The RAM's "write" and "output enable" controls are driven by outputs from the Multibus Interface and Bus Acquisition block.

Data Bus Transceiver

This block is also part of the Multibus Interface and is described on the next page.

Shared Resources Bus Transceiver

This block is also part of the Microprocessor Section and is described on the adjacent page.

Bidirectional Channel Port

The Bidirectional Channel Port consists of back-to-back octal latches. Each latch has a separate latch control and tri-state enable. The Channel Port Controller latches channel data into the channel side octal latch. This data is available to the Shared Resources Bus under control of the DM Bus Controller. The DM Bus Controller latches Shared Resources Bus data into the Shared Resources side octal latch. This data is available to the DC bus under control of the Bit-Slice Processor.

Multibus Interface Section

All application cards (devices) connect to the GPCI via the INTEL Multibus. Therefore, the GPCI requires a Multibus Interface Section to make the GPCI compatible with the Multibus. The four groups of signals that make up the Multibus are address, data, bus acquisition and control signals. The Multibus Interface Section is complicated by the fact that the GPCI can act as either a Multibus master or slave.

Address Bus Transceiver

The AM bus is connected to the Multibus via the Address Bus Transceiver. When the GPCI acts as a bus master, the AM bus acts as a Multibus address source. During this time, the microprocessor or the DMA Generator provides input to the Address Bus Transceiver via the AM bus. The Address Bus Transceiver acts as a bus transmitter. When one of the application cards acts as a bus master (GPCI acts as a bus slave), the Address Bus Transceiver is configured as a bus receiver. The Multibus Interface and Bus Acquisition block (Figure 4, lower-right corner) drive the direction and tri-state controls on the Address Bus Transceiver.

Extended Address and Handshake Generator

The Address Bus Transceiver links AM0-AME to the least significant 15 bits of the Multibus, resulting in 32K bytes of addressing capability. This capability is expanded to 1M bytes by the Extended Address and Handshake Generator. This block drives five additional Multibus address lines, and controls several miscellaneous handshake lines. The microprocessor controls the extended address portion of the Extended Address and Handshake Generator by writing data to I/O port 30H. It controls the handshake signals by writing data to I/O port 31H. The handshake lines are functionally part of the Control Section. These handshake lines are used by the I/O Interface Controller and the DMA Address Generator. These signals are discussed in detail in the Detailed Circuit Descriptions (Chapter 7).

Data Bus Transceiver

The Data Bus Transceiver is nearly identical to the Address Bus Transceiver. It links the Shared Resources Bus to the Multibus. Its tri-state output is controlled by the Multibus Interface and Bus Acquisition block.

Control Section

The Control Section consists of the:

- DMA Address Generator
- Channel Port Controller
- DM Bus Control
- Multibus Interface and Bus Acquisition

The DM Bus Control block includes the Autotransfer Controller, the DMA Controller, and the Shared Resources Bus Arbiter. In simplest terms, the Control Section controls the Shared Resources Bus gateways, Multibus arbitration, and DMA address generation. Since all major sections of the GPCI, IBM channel, and Multibus have access to the Shared Resources Bus, the Control Section receives inputs from all of these sections.

DMA Address Generator

The DMA Address Generator is microprocessor programmable. It is programmed in response to a data transfer type channel command, requiring a DMA transfer. Whether or not a data transfer command results in a DMA transfer depends on the coding in the Command PROM portion of the Command PROM Latch and Decode block. At present, all DMA transfers controlled by the GPCI's DMA Address Generator must include the channel as a data source or sink. That is, only DMA data transfers between the channel and the Tri-Port RAM, or between the Channel and Multibus are allowed. Data transfers between the channel and the microprocessor occur, but these transfers are not DMA types.

DMAs can be either memory or I/O transfers, depending on the transfer source and/or destination. For memory type DMAs, the DMA Address Generator is programmed with a starting address and an ending address. Each time a byte is transferred, the generator increments its address and compares it to the ending address. The DMA transfer terminates when the two addresses match. Programming the generator for an I/O transfer involves sending a byte count to the DMA generator. Prior to transferring the first byte, the generator sets an internal counter to zero. After each byte transfer, the counter is incremented and compared to the initial byte count value. When a match occurs, the transfer is terminated.

Channel Port Controller and DM Bus

Ideally, the Channel Port Controller and DM Bus Control block would be combined to form a Shared Resource Controller. However, due to the complex nature of such a controller, no standard IC exists at the present time that could implement the entire task. The final design choice was to design three small controllers, interconnect them, and control them with a master arbiter. The DM Bus Control block contains the arbiter and two controllers, the Autotransfer Controller and the DMA Controller. The remaining controller is the Channel Port Controller.

During a data transfer involving the channel, two of the controllers are active. The Channel Port Controller is active during all channel data transfers. The DMA Controller is active only during DMA transfers involving the channel. The Autotransfer Controller is active during non-DMA transfers involving the channel. The IBM channel, Multibus, and microprocessor transfer data using interlocked handshake signals. The Autotransfer Controller and DMA Controller interact with the Multibus' and microprocessor's handshake signals. The Channel Port controller interacts with the channel's handshake signals.

The purpose of each controller is to convert the resource-specific handshake signals into internal handshake signals. These signals are DRT/ and DTR/. DRT/ is an output of the Autotransfer and DMA Controllers and an input to the Bidirectional Channel Port. DTR/ is an output of the Bidirectional Channel Port and an input to the Autotransfer and DMA Controllers. In the example data transfer below, assume the transfer is from the microprocessor to the channel:

- the microprocessor tells the Autotransfer Controller it has data for the channel (WR goes high)
- the Autotransfer Controller tells the Channel Port Controller that a data transfer request is pending (DRT/ goes low)
- the Channel Port acknowledges the transfer request (DTR/ goes low)
- the Channel Port Controller tells the channel data is available (the Data In tag line goes high)
- the channel reads the Bidirectional Channel Port's latched data and acknowledges receipt of the data (the Data Out tag line goes high)
- the Channel Port Controller acknowledges the channel's receipt of data by dropping the Data In tag line
- the channel acknowledges the fall of Data In by dropping Data Out
- the Channel Port Controller tells the Autotransfer Controller that the channel read the data (DTR/ rises)
- the Autotransfer Controller tells the microprocessor that the channel received the data (SRACK/ goes low)
- the Autotransfer Controller completes the process by raising SRACK/

Shared Resources Arbiter

When a transfer command is received, the resource requests control of the DM bus via a control signal to the Shared Resources Arbiter. The Shared Resources Arbiter grants control if no other DM bus transfers are in progress.

Multibus Interface and Bus Acquisition

When a system containing more than one bus master is interconnected via a Multibus, a bus priority resolution scheme is required to ensure that the highest priority master is granted bus acquisition. In such a system, when two or more bus masters request the bus simultaneously, the bus priority resolution logic grants bus acquisition to the highest priority bus master.

The Multibus can be configured for serial or parallel priority resolution. Serial resolution is practical for resolving priority when a few bus masters are involved; parallel resolution resolves priority for a larger system, involving many bus masters. The GPCI uses a serial system. In a serially arbitrated Multibus system, requests for system bus access are ordered by priority on the basis of bus slot location. Each master on the bus notifies the next lower priority master when it needs to use the bus, and each master monitors the bus request status of the next higher priority master. Thus, the masters pass bus requests from one to the next in a daisy chain fashion.

Currently, the GPCI has the highest Multibus priority (Jumper selected). The Bus acquisition logic arbitrates bus requests for use of the Multibus from the Channel Port controller, DMA Controller and Microprocessor Section.

The Multibus Interface Section links the Multibus memory and I/O read and write control signals to the GPCI. When the GPCI acts as a bus master, the Multibus Interface generates these Multibus control signals. The GPCI responds to the Multibus memory and I/O read and write control signals when the GPCI acts as a bus slave. The DM Bus Control block plays a minor part in the Multibus interface by linking the transfer acknowledge signal (XACK/) to the GPCI. While operating as a bus master, the GPCI monitors XACK/; while acting as a bus slave, the GPCI acts as a source for XACK/.

GPCI Detailed
Circuit Description

GPCI Detailed Circuit Description

The schematic diagrams of the GPCI are shown on SSEC drawing #6450-0465 (Revision E, dated 3/13/87). For discussion purposes, the I/O Interface Controller and the Microprocessor sections are based on the I/O Interface Controller Detailed Functional Block diagram (Figure 6 on page 7-3) and the Microprocessor Section Detailed Functional Block Diagram (Figure 8 on page 7-19) respectively. These drawings are hybrids. They possess the advantages of block diagrams in that identical function chips are combined into single blocks, and buses are shown as single lines. However, they also show the source and destination of each control signal. The remaining sections of the GPCI are described using the schematic diagrams and the GPCI Functional Block Diagram (Figure 4 on page 6-3).

Schematic Conventions

The GPCI is built on a Multibus wire-wrap form factor board. Pin locations on the board are described by the column designator (alpha) and row number. Each IC schematic circuit symbol has a three- or four-character label that describes the column and row where pin 1 of that IC is located. The Detailed Circuit Description that follows refers to the ICs by this location label (i.e., a 74LS92 IC whose 1 pin is located in Column BR and row 46 is referred to as BR46). When reference is made to a schematic circuit symbol of a multiple section device, the symbol ID is used, followed by a hyphen and the letter designator. The symbol ID alone refers to single section ICs.

Logic Conventions

Logic signal names are indicated by all uppercase letters and numbers (i.e., ODAT0). A logic signal name ending with an overline or a trailing slash represents an active low signal (i.e., DRT or DRT/). The slash is used exclusively throughout this description, regardless of the active low symbol used on the schematic. A note of caution is appropriate here. There are several state machine diagrams in the following discussions. These diagrams show the logic signals that cause transitions from one state to another. These drawings use a bar over the signal to indicate a zero volt condition. Absence of an overbar indicates a "high" voltage condition.

Several conventions can be used to describe the state of a logic signal. Some of these conventions are true or false, high or low, one or zero, and active or inactive. In the remaining GPCI discussion, all logic states are described by using "high" and "low." This convention best describes the physical condition of a logic signal and is better suited for troubleshooting. The binary "1"s and "0" shown in the state machine diagrams represent "highs" and "lows" respectively.

The Tri-Port RAM was formerly called the Dual-Port RAM. The Tri-Port RAM's logic signals are acronyms based on its former name. Any logic signal beginning with DP (Dual-Port) involves the Tri-Port RAM. Also, DSGO (Dual-Port Slave Go) is a Tri-Port RAM signal name.

I/O Interface Controller

Bit-Slice Processor

Bit-Slice microprocessor theory is beyond the scope of this documentation. Additional information on bit-slice design is presented in Advanced Micro Devices' *Bipolar Microprocessor Logic and Interface Data Book*. In particular, refer to the AM2901 and AM2910 descriptions. Also, refer to the microcode listings and the GPCI Bit-Slice Microcode Descriptions in Chapter 8.

Refer to Figure 6 on the adjacent page and the Bit-Slice Processor section of the Detailed Functional Description (pages 6-1 and 6-2) as necessary. The Bit-Slice Processor consists of the Microsequencer, the Microcode PROMs (registered outputs) and the Bit-Slice ALU. Together, these components function as a pipeline architecture 4-bit microprocessor. Since the Bit-Slice Processor is conventional in design, this description focuses on control signals and interaction with other functional blocks of the GPCI.

The Bit-Slice Processor generates several control signals which are direct outputs from the Microcode block. This block consists of five 1024 byte PROMs cascaded together to form a 1024-word by 40-bit wide output. Nine of these bits are used for external control. Table 2 on page 7-5 identifies these nine control bits and their purpose.

CDCK/, CDDIR, CA0 and CA1 are inputs to the CDR's Board Control section. The Bit-Slice Processor drops CDCK/ when it needs to latch valid CD bus data into the appropriate latch on the CDR. CA0 and CA1 form a 2-bit address that selects the appropriate CDR latch when passing data from the GPCI to the CDR. CA0 and CA1 also control a buffer on the CDR board that allows channel input data to be read. CDDIR controls bidirectional latches and buffers on the CDR board. For additional information on CA0 and CA1, refer to the CDR section.

IOC functions as an enable signal for BN45, the Bit-Slice Control Logic block. The Control Logic block allows eight additional control signals to be developed under control of IOC. For additional information on BN45, refer to the Bit-Slice Control Logic block description below.

The remaining control signals shown in Table 2 are discussed in later sections.

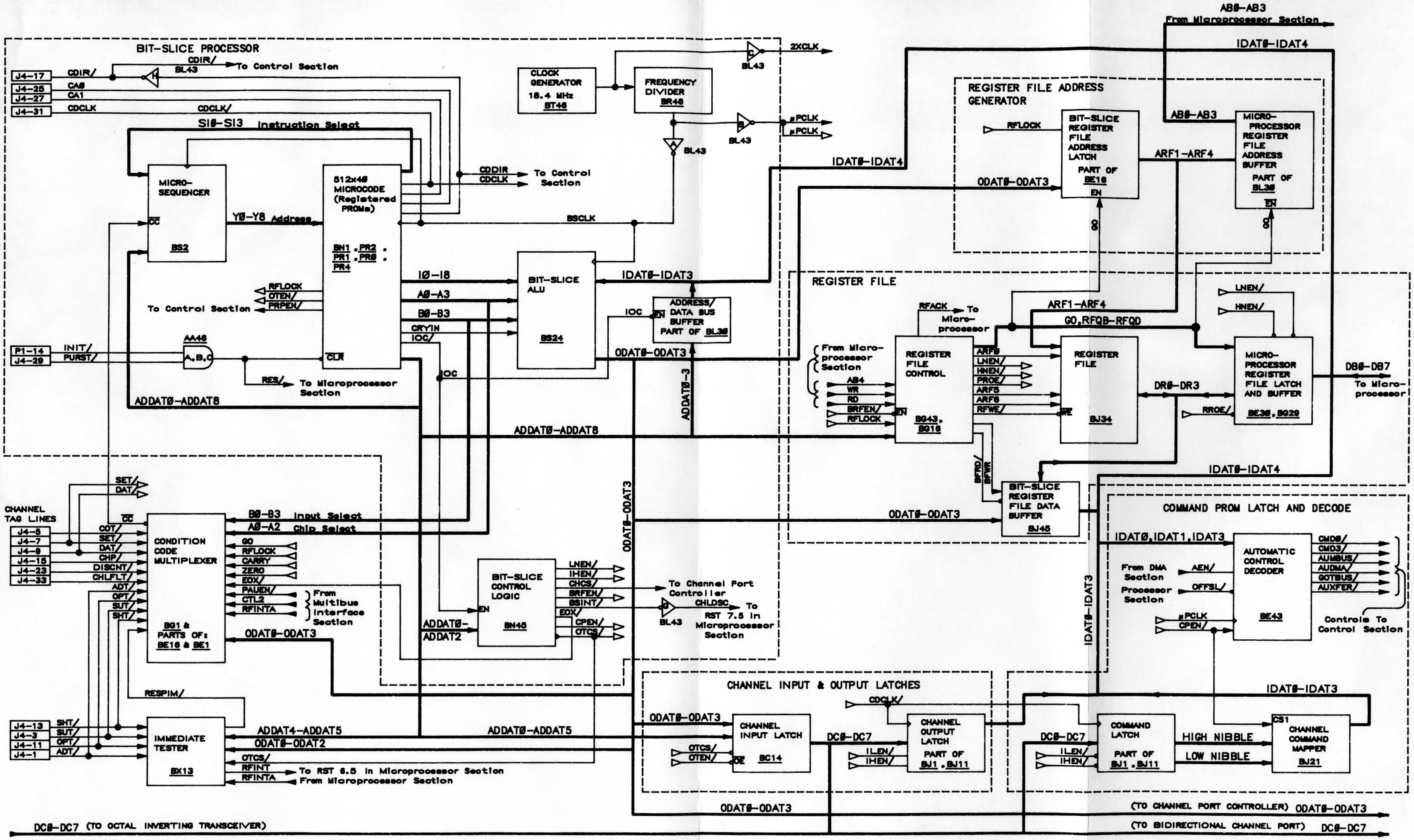


Figure 6. I/O Interface Controller Block Diagram

Control Signal	Purpose
IOC (I/O Control)	Enables the Bit-Slice Control Logic Block
CDCK/(Channel Data Clock)	Latches Data, Tag and Control signals on the CDR
CDDIR (Channel Data Direction)	Controls the Direction of the Data Flow on the CDR and GPCI's Octal Inverting Transceiver (part of the Shared Resources Section)
RFLOCK (Register File Lock)	Inhibits 8085 access of the Register File
OTEN/ (Output Enable)	Enables the Channel Input Latch (BC14)
PRPEN/ (Processor Port Enable)	Allows the Bit-Slice to read the DM Bus Data from the Bidirectional Channel Port (BC26)
CRYIN (Carry In)	Latches the Bit-Slice Data into the Bidirectional Channel Port (BC26)
CA0, CA1 (Channel Address)	CDR Control

Table 2. Microcode Block Originated Control Signals

Channel Input and Output Latches

The Channel Input Latch consists of PAL 4 located at BC14 (see sheet 2 of the schematics). BC14 assembles Bit-Slice nibbles into bytes for exporting to the DC bus. The output data from BC14 is passed to the IBM Channel. From the Channel's perspective, it is input data. This is where BC14 got its name. BC14 can also pass data to the Shared Resources Bus.

OTCS/ must be low while writing to either nibble of BC14. The Bit-Slice Processor uses ADDAT4 and ADDAT5 as nibble-select lines when loading data into BC14. If ADDAT4 and ADDAT5 are both low, ODAT0-ODAT3 are latched into the lower nibble of BC14. If ADDAT4 is high and ADDAT5 is low, ODAT0-ODAT3 are latched into the upper nibble of BC14. The lower nibble and upper nibble of BC14 are output to DC0-DC3 and DC4-DC7 respectively when OTEN/ goes low.

The Channel Output Latch consists of latches BJ1 and BJ11 which have the opposite function of BC14 above. That is, BJ1 and BJ11 simultaneously latch the upper and lower DC bus nibbles respectively when latch control CDCLK/ (Channel Data Clock) goes high. The I/O Interface Controller reads the lower or upper nibble by dropping ILEN/ (Input Low Enable) or IHEN/ (Input High Enable) respectively. BJ1 and BJ11 each have two sets of outputs, a tri-state set and a totem pole set. The tri-state sets are enabled by ILEN/ and IHEN/, as described above, and connect to IDAT0-IDAT3. The totem pole outputs are always active, and drive the least significant eight address pins of the Channel Command Mapper (see the Command PROM Latch and Decode section that starts on page 7-9).

Condition Code Multiplexer

The IBM channel is connected to the GPCI via a fully interlocked I/O interface. The I/O Interface Controller is a part of the GPCI end of that interlocked handshake scheme. At all times, the proper handshake action is indicated by interface signal conditions on the Tag and Control lines. Therefore, the I/O Interface Controller must monitor the Tag Out lines (Data Out, Command Out, and Address Out) and Control signals (Suppress Out, Operational Out, Service Out, Select Out, and Hold Out). The I/O Interface Controller monitors these lines via the Condition Code Multiplexer. In addition, the Condition Code Multiplexer monitors several status signals from other sections of the GPCI.

The Condition Code Multiplexer consists of PALs®* BG1, BE16 and BE1 (see sheet 2 of the schematics). The Condition Code Multiplexer can select either state (true or inverted) of 22 external inputs plus internally generated "Always" and "Never" conditions. Thus, the generator acts as a 1-of-46-line selector. It selects a particular line via the A and B buses. The A bus (A0 and A1 only) selects one of the three chips, while the B bus (B0-B3) selects one of 16 lines within that chip. Since the A and B buses are outputs of the Microcode block, output selection of the condition Code Generator is controlled by the Bit-Slice Processor. Table 3 on the next page summarizes the conditions which can cause CC/ to go low.

Immediate Tester

The Bit-Slice Processor must frequently test the status of more than one control line. Since the Bit-Slice Processor has no true interrupt capability, it must test these control line combinations frequently. One method of performing this function is to test the status of each control line of interest on a frequent basis to ensure that channel activity is monitored effectively. However, this approach is unsatisfactory since it requires excessive microcode execution time. A preferred approach is to provide programmable hardware that can monitor desired combinations of control lines and produce a single output to the Condition Code Multiplexer. Then the microcode can test the single output when desired. The Immediate Tester performs these tests. Its output signal is RESPIM/ (Respond Immediate).

The Immediate Tester (BX13 on sheet 2) is programmed by writing to I/O port 23H. The data on ODAT0-ODAT2 (Output Data bits 0-2) determines which test(s) will be performed by the Immediate Tester. Table 4 on the next page describes the programmable tests. If Operational Out falls, RESPIM/ will always go low, regardless of the programmed test.

The inputs to the Immediate Tester are LSUT/ (Latched Suppress Out), LOPT/ (Latched Operational Out), LADT/ (Latched Address Out), and LSHT/ (Latched SHT/-Select Out and Hold Out). These signals are outputs of BE1 (part of the Condition Code Multiplexer) because this chip had four unused D-latches available. Functionally, these latches are part of the Immediate Tester.

*PAL is a registered trademark of Monolithic Memories Inc.

A	B		B	
2	7	CARRY	F	CARRY/
	6	ZERO	E	ZERO/
	5	RFINTA	D	RFINTA/
	4	CTL2	C	CTL2/
	3	CHLFLT/	B	CHLFLT
	2	DISCNT/	A	DISCNT
	1	CHPAR/	9	CHPAR
	0	ALWAYS	8	NEVER
1	7	SUPOUT/	F	SUPOUT
	6	DAT/	E	DAT
	5	OPOUT/	D	OPOUT
	4	SEROUT/	C	SEROUT
	3	COMOUT/	B	COMOUT
	2	ADROUT/	A	ADROUT
	1	SHOUT/	9	SHOUT
	0	ALWAYS	8	NEVER
0	7	GO	F	GO/
	6	PAUEN/	E	PAUEN
	5	RESPIM/	D	RESPIM
	4	EOXL	C	EOXL/
	3	OUTDAT3	B	OUTDAT3/
	2	OUTDAT2	A	OUTDAT2/
	1	OUTDAT1	9	OUTDAT1/
	0	OUTDAT0	8	OUTDAT0

Table 3. GPCI Bit-Slice Condition Codes

ODAT2	ODAT1	ODAT0	PORT23	TEST
low	low	high	high	SHT/ going low
low	high	low	high	SUT/ going low
high	low	low	high	ADT/ going high
low	low	low	low	no change
low	low	low	high	SHT/ going high

ADT/ (Address Out) monitoring is not presently used.

Table 4. Immediate Tester Programming

Command PROM Latch and Decode

The Command PROM Latch and Decode consists of the totem pole output sections of BJ1 and BJ11, the Channel Command Mapper (BJ21), and the Automatic Control Decoder (BE43).

BJ1 and BJ11 latch the channel's command. The latched command drives the eight least significant address pins on BJ21. Prior to sending the command, the channel sends the device address which is stored in the Register File by the Bit-Slice Processor. When the new command is received and latched, the Bit-Slice Processor fetches the device address and uses it as a pointer to locate the device's "device type" code stored in the Register File. The address of a given cell in the Command PROM is selected by feeding the channel command to the eight LSBs and the device type to the four MSB addŕess pins of the Channel Command Mapper (BJ21). BJ21 produces a 4-bit device specific command code output (see Table 5 below). When the Bit-Slice Processor drops CPEN/, the 4-bit output code is written to the IDAT bus. IDAT0, IDAT1 and IDAT3 are inputs to the Automatic Control Decoder (BE43). BE43 is also enabled by CPEN/.

Mapper Output		Definition
0000	(0)	Immediate
0001	(1)	Not Assigned
0010	(2)	Not Assigned
0011	(3)	Not Assigned
0100	(4)	Test I/O
0101	(5)	Not Assigned
0110	(6)	Not Assigned
0111	(7)	Illegal
1000	(8)	Data XFER READ Micro (Sense)
1001	(9)	Data XFER READ Multibus
1010	(A)	Data XFER READ DMA (Memory)
1011	(B)	Data XFER READ DMA (I/O)
1100	(C)	Data XFER WRITE Micro
1101	(D)	Data XFER WRITE Multibus
1110	(E)	Data XFER WRITE DMA (Memory)
1111	(F)	Data XFER WRITE DMA (I/O)

Table 5. Command Code Definitions

When a new device type is added to the Multisourcerer, a new Command PROM map must first be made, assigning each command a value. Commands that are not used should be assigned a value of 7 (0111).

The Automatic Control Decoder (PAL 5, located at BE43) produces six outputs used by the Control Section. IDAT0, IDAT1, IDAT3, and control signals OFFSL/, CPEN/, AEN/, and AUTOST/ are inputs to BE43.

The outputs from BE43 are shown in Table 6 below, and are discussed in greater detail in the Control Section description which starts on page 7-31. Table 6 was developed from the PAL 5 equations located in the GPCI Supplemental Data Chapter (Chapter 9).

Signal	Description
CMD0	latched and inverted value of IDAT0 when CPEN/ goes low (LSB of the Command PROM output)
CMD3/	latched and inverted value of IDAT3 when CPEN/ goes low (MSB of the Command PROM output)
GOTBUS/	GPCI has use of the Multibus (bus master)
AUDMA/	active during DMA transfers between a channel and another resource
AUXFER/	active during non-DMA transfers between a channel and another resource
AUMBUS/	active during non-DMA transfers between a channel and the Multibus

Table 6. Automatic Control Decoder Outputs

Additional information on these control signals can be found in the Shared Resources description (page 7-28) and the Control Section description (starting on page 7-31).

Register File Section

The Register File Section, shown in Figure 4 on page 6-3, consists of the Register File Address Generator and the Register File. In turn, each of these blocks consists of several subsections. Figure 6 on page 7-3 shows the subsections which compose the Register File Address Generator and the Register File. Refer to the Register File Functional Description (page 6-7) as necessary.

Since the Register file is shared by the microprocessor and Bit-Slice Processor sections, the Register File Address Generator contains a control section (Register File Control block on Figure 6) which arbitrates contention for the Register File. When one of the requesting sections is granted access to the Register File, the control section allows that section's addressing bus (ODAT or AB) to address the Register File.

The Register File Control block consists of the Register File State Machine (PAL 6, located at BG43) and Register File Output Decode (PAL 7, located at BG16). Figure 7 on the next page shows the various states of the Register File State Machine. The states are defined by the levels on the state output signals RFQA/, RFQB/, RFQC/ and RFQD/.

Each of the bubbles in Figure 7 contains a 4-bit binary code. This code represents the physical output of BG43. The leftmost bit represents RFQA/; the rightmost bit represents RFQD/. As noted in the Logic Conventions, all logic signals are shown in physical convention. That is, a signal with no overline is read as "high," while a signal with an overline is "low."

Note in Figure 7 that state 1111 is common to two loops. One loop includes state 1110, while the other contains all other states. The loop containing 1110 is the Bit-Slice Processor's read/write loop; the other loop services the microprocessor. There is a separate microprocessor path for reading and writing. These separate read/write paths are required because the processor reads and writes 8-bit words, while the Register File stores 4-bit words. The WAIT states provide extra clock cycles required to clock latches and qualify chip enable strobes before processing the second 4-bit word.

External to PAL 6, RFQA/ is also called GO. GO is an active high signal and is applied to one of the inputs of the Condition Code Multiplexer. When the Bit-Slice Processor wants to access the Register File, it raises RFLOCK (Register File Lock). Assuming the state machine is currently servicing a microprocessor request, the state machine will finish the current request, and return to state 1111. Now, because RFLOCK is active, the state machine goes to state 1110. After raising RFLOCK, the Bit-Slice Processor enters a monitoring loop that tests GO. The Bit-Slice Processor recognizes that it has control of the Register File when GO goes high.

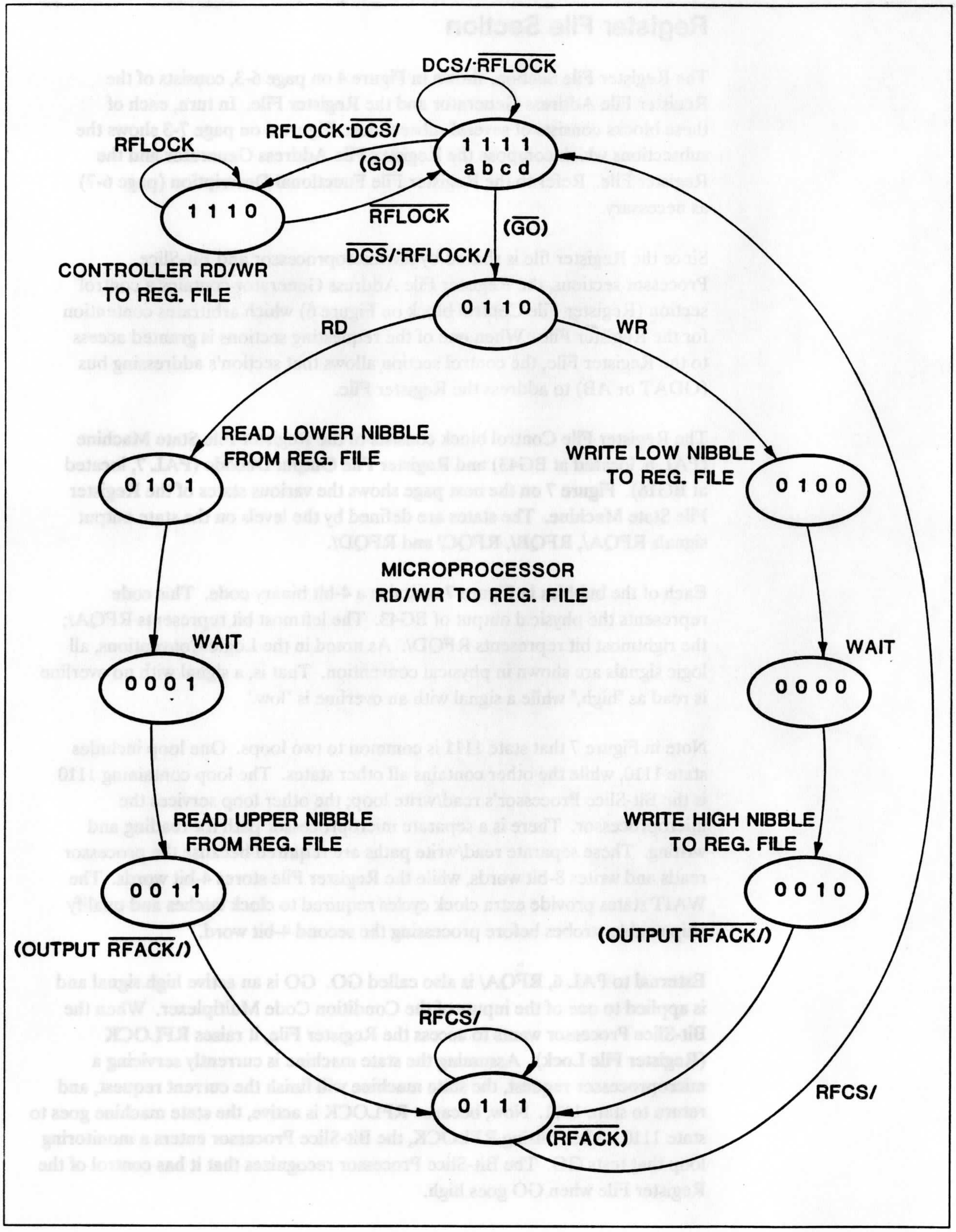


Figure 7. Register File State Machine (PAL 6A)

The Bit-Slice Processor performs a double read or write instruction after gaining access to the Register File. The reason for the double read or write instruction is to meet setup requirements for generating BRFRD/, BRFWR and RFWE/ (Bit-Slice Register File Read, Bit-Slice Register File Write, and Register File Write Enable, respectively). These control signals are outputs of PAL 6, as are the state output lines (RFQA/ - RFQD/).

Since RFWE/, BRFRD/ and BRFWR have internal inputs from the state outputs, two BSCLK (Bit-Slice Clock) pulses are required to clock data through the cascaded registers. Each Bit-Slice instruction requires one BSCLK pulse. This is why two read or write instructions to the Register File are required. The first BSCLK pulse latches the state outputs and qualifies the inputs to the control signal latches. The second BSCLK pulse latches the control signals. Additional information on BRFWR, BRFRD/ and RFWE/ can be found in the descriptions of the circuits in which they are used (refer to sheet 3 of the schematics).

The Register File Output Decode logic consists of PAL 7 located at BG16, and is shown on sheet 3 of the schematics. BG16 decodes the state outputs of BG43 to produce seven control signals. Four control signals (RFACK/, RROE/, LNEN/ and HNEN/) are used during microprocessor accesses of the Register File. The remaining output signals (ARF0, ARF5 and ARF6) are active during microprocessor or Bit-Slice Processor Control. They drive three Register File address pins (A0, A5 and A6 respectively). Table 7 below shows the signal source for each Register File address line.

Signal	Signal Source if 8085 is in Control	Signal Source if Bit-Slice is in Control
ARF1-ARF4	AB0-AB3	ODAT0-ODAT3
ARF0	State Machine	ADDAT4
ARF5	AB4	ADDAT5
ARF6	(none)	ADDAT6

Table 7. Register File Address Sources

The Register File Address Generator consists of part of BE16 (PAL 1 on sheet 2 of the schematics), and half of BL30. Together, these chips act as a four-line multiplexer. The output of the multiplexer, ARF1-ARF4, drives address input lines A1-A4 on the Register File. Both chips have tri-state output controls that are driven by GO. When GO is high, BE16's outputs are enabled. When GO is low, buffer BL30's outputs are enabled. BE16 latches data when RFLOCK goes high. Table 7 on page 7-13 summarizes the signal sources for ARF1-ARF4.

The Bit-Slice Register File Data Buffer consists of bidirectional buffer BJ45, located on sheet 3 of the schematics. This component transfers Register File output data to the Bit-Slice Processor's input data bus (IDAT0-IDAT3) during the Bit Slice Processor's Register File read cycles. BJ45 transfers Bit-Slice Processor output data from the ODAT bus to the Register File. BRFRD/ and BRFWR control the direction of data through BJ45. Table 8 below summarizes the data flow through BJ45.

BRFRD/	BRFWR	Action
0	0	DR bus goes to the IDAT bus (Read Cycle)
0	1	not allowed
1	0	tri-state
1	1	ODAT bus goes to DR bus (Write Cycle)

Table 8. Bit-Slice Register File Data Buffer Controls

The Microprocessor Register File Latch and Buffer consists of buffer BE30 and PAL 8 (located at BG29). Both chips are located on sheet 3 of the schematics.

When the microprocessor writes data to the Register File, it writes the address to the Register File Address Generator and writes the data onto the DB bus. It holds the data until it receives an acknowledge (RFACK/). The upper or lower nibble of the DB bus data is transferred to the DR bus by dropping the appropriate nibble enable line (HNEN/ or LNEN/ respectively - refer to the Register File Output Decode above). The low nibble (DB0-DB3) is stored first, then the upper nibble (DB4-DB7). After the upper nibble is stored in the Register File, the Register file Control logic drops RFACK/.

To read data from the Register File, the microprocessor writes the address to the Register File Address Generator and raises RD. The processor enters wait states until RFACK/ goes low, indicating that the data on the DB bus is valid. Refer to Figure 7 on page 7-12. When the Register File Control section receives a read request from the microprocessor, it must assemble the requested nibbles into a byte, and write the byte of data onto the DB bus. The byte is assembled in PAL 8, under control of the Register File State Machine. Inputs to PAL 8 are the state code outputs of PAL 6. PAL 8 uses state 0101 as a latch strobe to latch the lower nibble. It uses state 0011 as a latch strobe to latch the upper nibble. PAL 7 (Register File Output Decode) uses state 0011 to generate RFACK/.

The Register File consists of RAM BJ34, and is located on sheet 3 of the schematics. BJ34 is a 1024 by 4-bit static RAM. However, because its upper four address pins are grounded, it functions as a 128-word by 4-bit RAM. The addressing scheme for the Register File is described in the GPCI Detailed Functional Description Chapter (page 6-7).

Signal	Definition	Address/Port	Process
RFCS/	Register File Chip Select	6800H-69FFH	(RD & WR)
DPCS/	Tri-Port Chip Select	7000H-7FFFH	(RD & WR)
OFFSL/	OFF-BOARD Select	08000H-FFFFFH	(RD & WR)
OFFSL/	OFF-BOARD Select	I/O Ports 40H-7FH	(RD & WR)
SDDR/	Shared Resources Data Direction	7000H-FFFFH I/O Ports 18H-1FH	(RD) (RD)
MND/	Memory-No Delay	0000H-67FFH 6C00H-6FFFH	(RD) (RD & WR)
SPCS/	Scratch Pad RAM Chip Select	6C00H-6FFFH	(RD & WR)
MDDR/	Memory Data Direction	0000-6FFFH	(RD)
PMCS/	Permanent Memory Chip Select	0000-67FFH	(RD)
NVCS/	Non-Volatile RAM Chip Select	6A00H-6BFFH	(RD & WR)

Table 9. Memory Map

Device Controller

Microprocessor Section

Refer to Figure 8 on page 7-19. As stated in the GPCI Detailed Functional Description Chapter, the microprocessor chip (AN3) is not documented in detail here, since it is documented by its manufacturer.

Address Latch

Refer to Figure 8 and sheet 4 of the schematics. The Address Latch consists of AT1. AT1's D-inputs are driven by the microprocessor's bidirectional AD bus. AT1 is a transparent octal latch. That is, while its enable pin (pin 11) is high, its outputs follow its inputs. When pin 11 goes low, the D-inputs are latched. Pin 11 is driven by the ALE output from the microprocessor (pin 30). AB0-AB7, the outputs of AT1, join the A8-AF outputs from the microprocessor to form the 16-bit address bus (AB0-ABF).

Memory Address Decode

The Memory Address Decode block consists of PAL 13A, located at AV1 (sheet 5). Inputs to AV1 consist of the nine MSBs of the microprocessor's address bus (AB9-ABF), and microprocessor control signals RD, WR, IO/M, MS0 and MS1. RD and WR (Read and Write respectively) are outputs of inverter AG14, sections G and H respectively. MS0 and MS1 are simultaneously high during the microprocessor's "opcode fetch" cycle. These signals ensure that no opcode fetches occur, except from on-board PROM and RAM. The signal IO/M establishes the signal's domain (I/O or memory). IO/M is high during I/O port addressing and low during memory addressing.

Table 9 on the adjacent page shows the outputs of AV1, addresses, and their domain (Memory or I/O).

PMCS/, SPCS/ and NVCS/ enable the EPROM (AB6), the Scratch Pad RAM (AE14 and AE24), and the Non-Volatile RAM respectively (AC21).

SDDR/ controls the data flow direction of AV17 (Shared Resources Bus Transceiver) located between the Microprocessor Section and the Shared Resources Bus (DM bus).

MDDR/ is combined with an output from the I/O Port Decode (IDDR/) to drive the direction control pin on the Extended Bus Transceiver (AT17). Refer to the IDDR/ description below. AT17 passes data between the microprocessor and the memory blocks (EPROM, RAM and Non-Volatile RAM) and the Register File via the Microprocessor Register File Latch and Buffer.

RFCS/ is used by the Register File (see Bit-Slice Processor Description) during microprocessor read/write operations involving the Register File.

DPCS/ is low while read/write operations involving the Tri-Port RAM are in progress (refer to the Shared Resources Section Description on page 7-28).

OFFSL/ is low during off-board memory and I/O read/write operations. This signal is low only when the GPCI is acting as a Multibus master.

MND/ generates a low input to the Ready Generator (BA1) when performing a read/write operation involving the EPROM or Scratch Pad RAM. MND/ prevents the microprocessor from entering a wait state when accessing these memory components. Note that the address space of the Non-Volatile Memory (slower access time) is not included in the makeup of MND/. The Non-Volatile RAM is slower, and requires one wait state for reliable operation.

I/O Port Decode

The I/O Port Decode consists of PAL 14, located at AX1 (sheet 5 of 7). Inputs to AX1 consist of AB3-AB7 (part of the microprocessor's address bus), and control signals RD, WR and IO/M. The control signals serve the same purpose as they do in the Memory Address Decode section. The 8085 microprocessor outputs the I/O address simultaneously on the upper (AB8-ABF) and lower (AB0-AB7) halves of its address bus. Thus, AB3-AB7 contain the five MSBs of an I/O port address during I/O reads and writes. Table 10 below shows the outputs of AX1, their I/O port addresses, and signal destination.

Signal	Definition	Port Address	Destination
IND/	I/O No Delay	00-0FH, 20H-3FH	BA1 (SHT. 4)
XIOCS/	Extra I/O Chip Select	30H-37H	AX28 (SHT. 7)
DMACS/	DMA Chip Select	20H-2FH	AJ41 (SHT. 6)
CHPS/	Channel Port Select	18H-1FH	AG28, AX14 (SHT. 7)
MUCS/	MUART Chip Select	00-0FH	AJ3 (SHT. 4)
LLCS/	LED Latch Chip Select	38H-3FH	AA32 (SHT. 5)
DSPEN/	Display Enable	10H-17H	Display, AE1 (SHT. 5) BA1 (SHT. 4)
IDDR/	I/O Data Direction	10-17H (RD)	AT17 via AE36 (SHT. 5)

Table 10. I/O Map

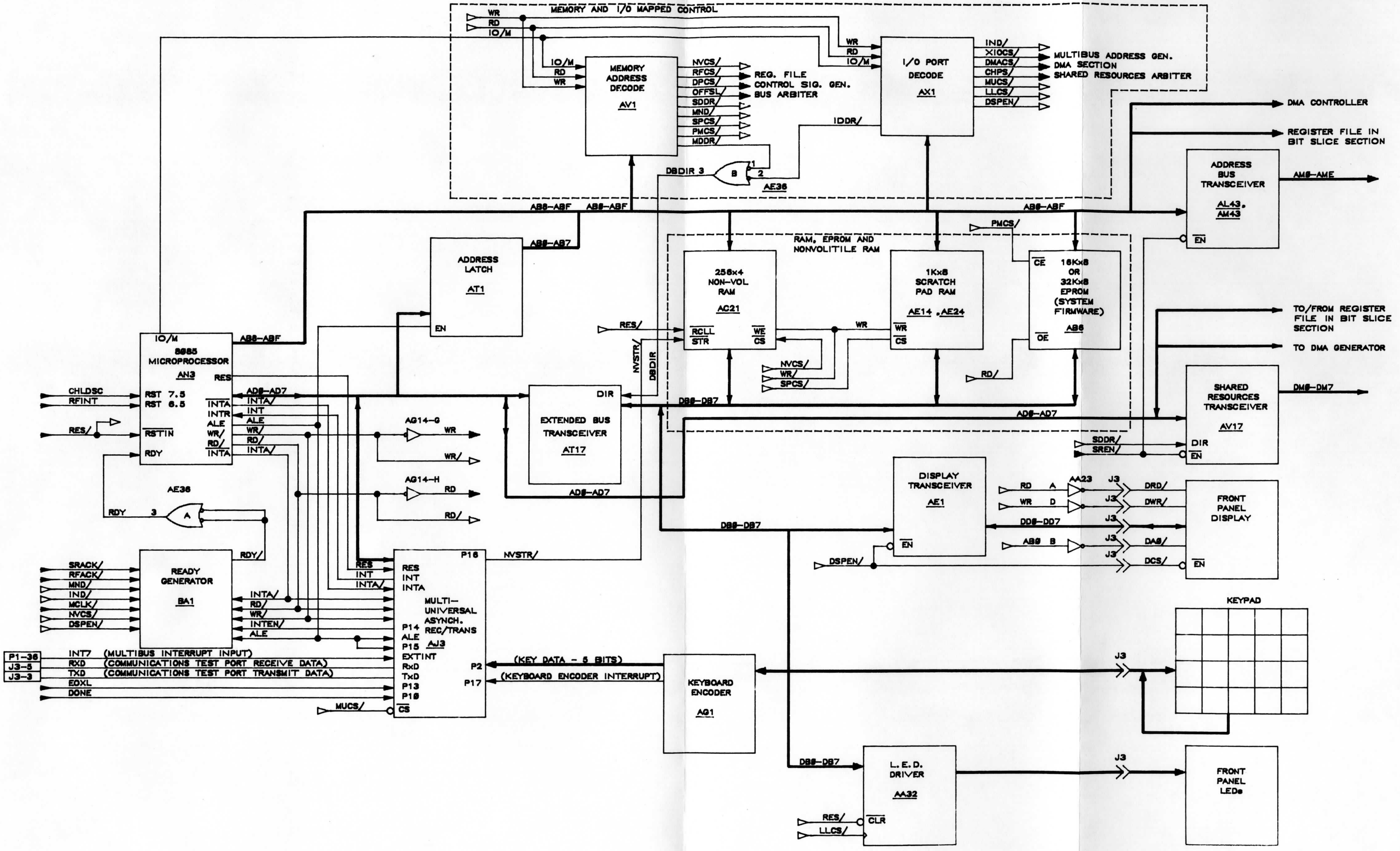


Figure 8. Microprocessor Section Detailed Functional Block Diagram

IND/ provides an immediate I/O acknowledge signal to the Ready Generator for all on-board I/O port reads and writes (ports 00-3FH), except ports 10H-1FH. Ports 10H-17H are used for communicating with the 16-character vacuum fluorescent front panel display. Ports 18H-1FH address the channel port.

Microprocessor/channel communication paths are via the DM bus. If the communication paths are busy during a microprocessor I/O read or write request, the microprocessor is forced to wait until the DM Bus Controller grants permission for communication to proceed. Because the channel access time is variable, IND/ cannot generate the acknowledge signal for microprocessor/channel communication. The microprocessor/channel I/O acknowledge signal (SRACK/) is generated by the DM Bus Controller.

IND/ cannot be used as the acknowledge signal when communicating with the display (refer to DSPEN/ below). The front panel display contains its own on-board microprocessor. If that microprocessor is busy, the 8085 microprocessor may have to wait until the display's microprocessor is ready to send or receive data (see DSPEN/ below).

XIOCS/ is used as a latch signal by the Extended Address and Handshake Generator (AX28). For additional information, refer to the Control Section or the Extended Address and Handshake Generator description.

DMACS/ enables DMA Address Generator programming by the microprocessor. Once the DMA Address Generator is programmed, DMACS/ is no longer required for a DMA transfer. Refer to the Control Section or the DMA Address Generator for additional information.

CHPS/ causes the DM Bus Control block to generate a Shared Resources Request (SRREQ/) when the 8085 microprocessor attempts to read or write data from or to the channel. The Shared Resources Bus Arbiter portion of the DM Bus Control block processes the SRREQ/ request signal, eventually granting the microprocessor access to the DM bus and the Bidirectional Channel Port (BC26). CHPS/ is also applied to AG28 (part of the Multibus Interface and Bus Acquisition Section) where it is combined with the microprocessor's RD and WR signals to form CHLEN/ (Channel Latch Enable) and CHLCK/ (Channel Latch Clock). CHLEN/ and CHLCK/ control the DM bus side of the Bidirectional Channel Port. For additional information, refer to the Control Section description that begins on page 7-31.

MUCS/ enables the MUART. See the MUART description on page 7-24.

LLCS/ latches data on the DB data bus onto the LED driver (AA32). The front panel LED channel activity indicators are driven by AA32.

DSPEN/ functions as a chip select for the front panel display's data bus transceiver (AE1), a transfer acknowledge for the Ready Generator, and a chip select for the display module. For additional information, refer to the Display Transceiver and Front Panel Display (page 7-23) and the Ready Generator (page 7-26) descriptions.

IDDR/ is ORed with MDDR/ (described in the Memory Address Decode Description above) by inverting input OR gate AE36, section B. The output of this gate (DBDIR) controls the direction of data flow through the Extended Bus Transceiver (AT17).

RAM

Refer to sheet 5 of the schematics. The RAM consists of two cascaded 4-bit by 1K static RAM chips located at AE14 and AE24. AE14 stores the four LSBs of the DB bus; AE24 stores the four MSBs of the DB bus. The DB bus is bidirectional and connects to the microprocessor's AD bus via AT17, the Extended Bus Transceiver. The chip select pin (pin 8) is driven by SPCS/, an output of the Memory and I/O Mapper. SPCS/ is low for memory addresses 6C00H-6FFFH. The write enable pins (pin 10) are driven by the microprocessor's WR/ output control signal (pin 31).

EPROM

The EPROM consists of AB6, and is shown on sheet 5 of the schematics. AB6 is organized as a 16K byte EPROM. It is enabled by the PMCS/ memory mapped control signal generated by AV1. PMCS/ is low for memory addresses 0000-67FFH. AB6's data output port connects to the DB bus (refer to the RAM description above).

NVRAM

The NVRAM consists of AC21, and is shown on sheet 5 of the schematics. AC21 is organized as a 256-nibble RAM, mirrored by a 256-nibble EEPROM (Electrically Erasable Programmable Read-Only Memory). The chip select pin is driven by NVCS/, an output of AV1. NVCS/ is low for memory addresses 6A00H-6BFFH. The write enable pin (pin 11) strobes DB bus data into the RAM portion of the NVRAM. The entire RAM is copied to the EEPROM portion of the NVRAM if pin 9 (Store) goes low. Pin 9 is driven by NVSTR/, an output of the MUART. The EEPROM contents are dumped to the RAM if pin 10 (RCLL - Recall) is strobed low. Pin 10 is driven by RES/ which goes low during board resets.

Display Transceiver and Front Panel Display

The Display Transceiver (AE1) is located on sheet 5 of the schematics. It connects the DB data bus to the front panel display. AE1 is enabled by DSPEN/, and is low during a port read or write to port address 10H-17H (refer to the I/O Port Decode on page 7-18). The direction of data through AE1 is controlled by the microprocessor's RD signal. While writing to the display, RD is low, causing AE1 to pass data from the DB bus to the display.

The front panel display is enabled by DSPEN/, and requires TTL data in 8-bit ASCII format. In addition to DSPEN/ and the 8-bit DD bus, the display requires RD, WR, and AB0 from the 8085 microprocessor. When AB0 is low, ASCII control or character data can be read from or written to the display using RD or WR respectively. When AB0 is high, commands can be written to the display by pulsing WR high; status can be read from the display by pulsing RD high.

Keyboard Encoder and Keypad

AG1, the Keyboard Encoder is located on sheet 4 of the schematics. The chip is enabled at all times since its output enable pin is grounded. The chip contains its own on-board oscillator, encoding circuity, switch debounce circuitry, and pull-ups. The front panel keyboard (20 keys) is wired as a matrix, with four columns and five rows. Each row wire (Y1 - Y5) is pulled high by one of the chip's internal pull-up resistors. The column wires (X1 - X4) are sequentially pulsed low, and the row wires are continuously monitored.

When a key is pressed, a connection is formed between one of the row wires and one of the column wires. If a key is pressed, the connected row wire goes high when the connected column wire is pulsed. When activity is sensed on a row wire, the Keyboard Encoder encodes the row and column intersection, latches the 5-bit code into the output register, and pulses its "Data Available" pin high. The Data Available signal is connected to an input port on the MUART and serves as a Keyboard Encoder interrupt. AG1's output register is also connected to the MUART. For additional information, refer to the MUART description below.

MUART

AJ3, the MUART (Multi-Universal Asynchronous Receiver Transmitter) is located on sheet 4 of the schematics. AJ3 consists of:

- two programmable, parallel 8-bit I/O ports (P10-P17 and P20-P27)
- five 8-bit programmable timer/counters; four can be cascaded to form two 16-bit timer/counters
- eight-level priority interrupt controller
- baud rate generator, programmable for 13 common baud rates up to 19.2K bits/second
- serial asynchronous communications interface

Port 2 (P20-P27) receives the 5-bit key code data from the Keyboard Encoder. These five bits are applied to P20-P24. P25 is grounded and P26-P27 are tied high.

Each pin on Port 1 (P10-P17) is programmed as an input or output pin in accordance with the Port 1 control register contents. If the value of a bit in the control register is "1", the corresponding Port 1 pin is an output pin; if the bit is "0", the corresponding pin is an input. Pin P17 is under control of a bit in Command Register 1. When this bit is set, P17 generates an interrupt when its input level goes from low to high. The Keyboard Encoder Interrupt (Data Available) output is connected to P17. Table 11 below describes the use of each pin on Port 1.

Port	Input/Output	Purpose
P10	Input	DONE - DMA Transfer Complete
P11	(not used)	
P12	(not used)	
P13	Input	EOXL - Channel Indicates End of Transfer
P14	Output	INTEN/ (Interrupt Enable) - see note below
P15	Input	Control Signal ALE - see note below
P16	Output	NVSTR/ - Non-Volatile RAM Store
P17	Input	Interrupt - Keyboard Encoder (see Port 2 above)

Note: If the 8085 attempts to write to or read from a nonexistent resource (i.e., a de-installed application card), the 8085 enters a permanent wait state because it will never receive an acknowledge input to its Ready Generator. INTEN/ is the output of Timer 5 which is reset and retriggered by the ALE signal applied to Port 15. Thus, Timer 5 is reset every time the 8085 does an opcode fetch cycle (ALE goes high). During a wait state, opcode fetch cycles do not occur. INTEN/ is generated if the acknowledge response time exceeds the delay programmed into the timer. INTEN/ is an input to the Ready Generator that causes a pseudo RDY input to the 8085. This pseudo RDY signal ends the wait state.

Table 11. MUART Port 1 Control Signal Summary

Signals DONE (DMA completed) and EOXL (channel End of Transfer Latched) are applied to Port 1 pins P10 and P13 respectively. Although these signals do not generate interrupts, they are readable by the 8085. During a DMA, the 8085 periodically reads (polls) Port 1 on the MUART and tests bit 0 to determine if the DMA is completed. Likewise, during a channel transfer, bit 3 is tested to determine if the channel has indicated an end of transfer.

The serial asynchronous communication port is used for serial communication with an external terminal. TXD and RXD are the transmit and receive data lines respectively. The data transmission baud rate is a function of AJ3's internal baud rate generator. The baud rate frequency is programmable, and controlled by the 8085. Currently, the asynchronous communication port is set for 19.2K baud, 8-bit ASCII, no parity, and one stop-bit.

The eight-level priority interrupt controller section of the MUART arbitrates eight sources of interrupts, including the external interrupt input (Multibus INT7/). Table 12 below shows the priority of interrupts, as determined by the MUART.

Source	Priority
Timer 1	0 (highest)
Port P17	1
INT 7/	2
Timer 3	3
Receiver Interrupt	4
Transmitter Interrupt	5
Timers 2 & 4	6
Timer 5	7 (lowest)

Table 12. MUART Interrupt Priority

LED Drivers

AA32, the LED Drivers chip, is located on sheet 5 of the schematics. AA32 is an octal D-type latch, having a master clear input (pin 1). AA32 is latched by LLCS/ which goes low for I/O port address 38H-3FH. Its D-inputs are driven by the DB bus. The anode of each front panel mounted LED is connected to the +5VDC supply; each cathode is connected to an output of AA32. An LED is "on" if its cathode is pulled low by its respective driver in AA32. AA32 is cleared by RES/.

Extended Bus Transceiver

The Extended Bus Transceiver consists of AT17 which is shown on sheet 5 of the schematics. AT17 connects the microprocessor's AD bus with the DB bus. AT17 is always enabled since pin 19, its enable pin, is grounded. The direction of data flow through AT17 is determined by DBDIR which is the output of an inverting input OR gate located at AE36-B. This gate produces a high output if MDDR/ or IDR/ goes low. DBDIR goes high while reading I/O ports 10H-17H (Front Panel Display), and reading memory addresses 0000-67FFH and 6C00H and 6C00-6FFFH (RAM, EPROM, and NVRAM).

Shared Resources Bus Transceiver

The Shared Resources Bus Transceiver consists of AV17, and is shown on sheet 6 of the schematics. It connects the DM bus (Shared Resources bus) with the microprocessor's AD bus. AV17 is enabled by SREN/, an output of the Shared Resources Arbiter. The direction control of AV16 (pin 1) is driven by SDDR/. SDDR/ is an output of the Memory and I/O Mapper shown on sheet 5 of the schematics. It goes low when reading I/O ports 18H-1FH (Channel Port) or reading addresses 7000H-FFFFH (Tri-Port RAM and off-board).

Address Bus Transceiver

The Address Bus Transceiver consists of AR43 and AT43. These chips are shown on sheet 6 of the schematics. They link the AM bus with the Multibus' address bus (ADR0/-ADRE/). AR43 and AT43 are enabled by BOEN/, an output of the Multibus Interface and Bus Acquisition circuits. The direction control for AR43 and AT43 is driven by SLAVEGO/, an output of the Shared Resources Arbiter. When SLAVEGO/ is high, the AM bus drives the Multibus address lines; when SLAVEGO is low, the Multibus address lines drive the AM bus.

Ready Generator

The Ready Generator consists of PAL 12, located at BA1. This circuit drives the RDY pin on the 8085, thereby controlling the number of wait states during a memory mapped or I/O mapped read or write cycle. Several inputs to BA1 are clocked into latches by MCLK (Microprocessor clock), resulting in a delay of up to one MCLK clock period. The remaining inputs to BA1 are ORed together, resulting in zero wait states. The latched signals are ORed with the zero wait state signals to produce RDY. Table 13 on the adjacent page lists the inputs to the Ready Generator, and the number of MCLK clock cycle delays.

Signal	Source	Delay (MCLK cycles)
MND/	Memory Address Decode (AV1)	0
IND/	I/O Port Decode (AX1)	0
NVCS/	Memory Address Decode (AV1)	1
DSPEN/	I/O Port Decode (AX1)	1
RFACK/	Register File Output Decode (BG16)	1 (see notes below)
SRACK/	DM Bus Controller (AX14)	1 (see notes below)
INTA/	Microprocessor (AN3)	0
INT	See Note in Table 12	1
INTEN	See Note in Table 12	

Notes: RFACK/ is a read/write acknowledge signal generated by the Register File upon completion of a read or write cycle. However, since the Register File is shared by the Bit-Slice Processor and the 8085, the Register File may already be busy when the 8085 attempts a read or write cycle. The arbiter portion of the Register File will prevent the 8085 from accessing the Register File until after the Bit-Slice completes its cycle, possibly resulting in a delay of RFACK/. After RFACK/ goes low, RDY goes high on the next rising edge of MCLK.

SRACK/ is a read/write acknowledge signal from the Shared Resources Arbiter portion of the DM Bus Controller. Since the DM bus is shared by several resources, it may be busy when the 8085 attempts to use the bus. Thus, SRACK/ may be delayed. After SRACK/ goes low, the Ready Generator generates the RDY signal on the next rising edge of the MCLK.

Table 13. Ready Generator Characteristics

Shared Resources Section

The GPCI's shared resources consist of the Tri-Port RAM, channel port, microprocessor and Multibus. Data transfers from one resource to another occur over the DM bus. Except for the Tri-Port RAM, all resources are connected to the DM bus via tri-state bus transceivers. The Tri-Port RAM contains a tri-state output, and is connected directly to the DM bus. The Control Section controls the DM bus by controlling the direction of data flow through the transceivers and activating the appropriate tri-state enables. Primarily, the Control Section controls the Shared Resources Section.

The Functional Description Section shows the various data transfer capabilities between the resources, as well as the data transfer domains. This section describes those transfers via timing diagrams which begin on page 7-52. The timing diagrams show the interlocked handshake signals and many of the control signals originating in the Control Section. They also show signal cause and effect. The diagrams show the transfer of two or more bytes of data, but do not attempt to show contention resolution when more than one resource simultaneously attempts to control the DM bus. Refer to the Control Section on page 7-31 for additional information on various control signals. See Table 14 below to determine which timing diagram describes a particular transfer.

From	To	Domain	Timing Diagram
Microprocessor	Channel Port	I/O	1
Channel Port	Microprocessor	I/O	2
Microprocessor	Tri-Port RAM	Mem.	3
Tri-Port RAM	Microprocessor	Mem.	3
Microprocessor	Multibus	I/O	4
Multibus	Microprocessor	I/O	5
Microprocessor	Multibus	Mem.	4
Multibus	Microprocessor	Mem.	5
Channel Port	Tri-Port RAM	DMA Mem.	6
Tri-Port RAM	Channel Port	DMA Mem.	7
Channel Port	Multibus	DMA I/O	8
Multibus	Channel Port	DMA I/O	9
Channel Port	Multibus	DMA Mem.	8
Multibus	Channel Port	DMA Mem.	9
Multibus	Tri-Port RAM	Mem.	10
Tri-Port RAM	Multibus	Mem.	10
Multibus*	Channel Port	Mem.	11 and 13
Channel Port*	Multibus	Mem.	12 and 13

*Channel masquerades (from the Multibus' view) as a memory block which is contiguous with the Tri-Port RAM.

Table 14. Shared Resources Transfer Timing Diagram Directory

Multibus Interface Section

The GPCI is linked to its application cards via the Multibus. The Multibus Interface Section links the remainder of the GPCI to the Multibus, and consists of three signal blocks (refer to Figure 4 on page 6-3). These blocks are the Address Bus Transceiver, the Data Bus Transceiver, and a Control Interface Section consisting of the Multibus Interface and Bus Acquisition block and part of the DM Bus Control block. The Extended Address portion of the Extended Address and Handshake, together with the Address Bus Transceiver, make up the Address Bus Interface.

Address Bus Interface

The GPCI can act either as a Multibus master or Multibus slave. As a bus master it drives the Multibus address lines to address memory and I/O ports located off-board on the various application cards. As a bus slave, the GPCI's Address Bus Interface must respond to off-board bus masters' requests for resource access. The resources on the GPCI that are accessible by off-board Multibus masters are the IBM channel port and the Tri-Port RAM. Thus, the Address Bus Interface must be bidirectional. The lower 15 bits of the address bus are linked to the GPCI's AM bus by octal inverting transceivers AR43 and AT43 (Address Bus Transceiver). The upper five bits are linked by the Extended Address and Handshake Generator.

Internally, AT43 and AR43 act as back-to-back inverting tri-state octal buffers. When the output enable (OE/) pin is high, both sets of buffers are in a high impedance state. When OE/ is low, the transfer (T) pin determines which set of buffers is active. When T is high, the transfer is from A to B; when T is low, the transfer is from B to A. The T pins on both chips are driven by SLAVEGO/. The OE/ pins are driven by BDEN/ (Board Enable). SLAVEGO/ is an output of the DM Bus Control block, and goes low when an off-board Multibus master is granted control of a GPCI resource. Thus, when control is granted to an off-board master, AR43 and AT43 act as address line receivers. BDEN/ is low anytime the GPCI is acting as a bus master or bus slave

Full compliance with the Multibus interface specifications requires 20 address lines. The 15 LSB address lines are linked by the Address Bus Transceiver. The Extended Address Generator links the five MSBs of the Multibus address bus to the GPCI. The Extended Address and Handshake Generator consists of PAL 16, located at AX28, and is shown on sheet 7 of the schematics

PAL 16, an AMPAL22V10, uses five registered outputs to drive the extended address lines, and four additional registered outputs for control (handshake). All registers are clocked by XIOCS/ (Extra I/O Chip Select). XIOCS/ is an output of the I/O Port Decode (AX1), and is active for port addresses 30H-37H. The five extended address registers are programmed by the microprocessor when writing data to I/O address 30H; the four control lines are programmed by writing data to port 31H.

Two important features of this PAL are that it has an external reset input (RES/) to each of its registers, and the active state (high or low) of each output is programmable. When RES/ goes low, each output goes to its inactive state. ADRF/ is programmed as an active high output (goes low during reset), while ADR10/ - ADR13/ go high. Therefore, a reset sets the Multibus default address to 08000H. This eliminates the need to reprogram the Extended Address Generator after a power-up reset, since the present GPCI configuration uses addresses 08000H-FFFFFH as off-board (Multibus) addresses.

The Extended Address Generator is programmed by the data on the five LSBs of the DB bus while writing to port address 30H (see Table 15 below).

DB4	DB3	DB2	DB1	DB0	Bit Asserted
0	0	0	0	0	ADRF/
0	0	0	1	0	ADR10/
0	0	1	0	0	ADR11/
0	1	0	0	0	ADR12/
1	0	0	0	0	ADR13/

Table 15. Extended Address Generator Programming

The tri-state outputs of the Extended Address Generator are enabled when BDEN/ is low and SLAVEGO/ is high. These conditions occur only when the GPCI is acting as a Multibus master.

Since the Handshake Generator portion of PAL 16 is functionally part of the GPCI Control Section, it is discussed in detail in that section.

Data Bus Transceiver

The Data Bus Transceiver consists of AX43, and is shown on sheet 6 of the schematics. AX43 is an inverting octal transceiver that links the Multibus data bus (DAT0-DM7) to the Shared Resources bus (DM0-DM7). AX43 is enabled by BDEN/ which is low anytime the GPCI acts as a bus master or bus slave. BDT/R drives the direction control of AX43. It is described in the Control Section description below.

Control Section

The Control Section was first introduced in Figure 4. In addition to the DM Bus Control (Shared Resources section control), the Control Section includes a DMA Address Generator, and a Multibus Interface and Bus Acquisition section. The Control Section Description is organized as follows:

- Control Section Overview
- General Control Section Originated Signals
- DMA Address Generator Signals
- Multibus Interface and Bus Acquisition Signals

Control signals which originate in other sections of the GPCI and are used outside of the originating section are also described here.

Control Section Overview

The Control Section controls resource access to the DM bus. The DM bus is common to the:

- Channel Port
- I/O Interface Controller
- Microprocessor Section
- Tri-Port RAM
- Multibus

The DM Bus Control section consists of the Shared Resources Arbiter and two sub-controllers: the DMA Controller and the Autotransfer Controller. Both sub-controllers and the arbiter function as separate finite state machines. Each state machine consists of a PAL. Depending on the type of transfer and the resources involved, one or more of the sub-controllers will be active. The resource arbiter (PAL 17A) controls sub-controller activity. In turn, the Shared Resources Arbiter receives inputs from the Automatic Control Decoder, a part of the Command PROM Latch and Decode block in Figure 4. Finally, the Automatic Control Decoder is responsive to channel commands, microprocessor access of Multibus memory and I/O space, Multibus bus master status, and channel data transfer status.

Figure 9 on the next page shows the interconnections between these blocks of control logic. It does not show all output control signals. The Shared Resources Arbiter, the Autotransfer Controller, and the DMA Controller are represented by state diagrams. Figures 10, 11 and 12 (on pages 7-33, 7-34 and 7-35) represent the Shared Resources Arbiter, the Autotransfer Controller and the DMA Controller, respectively. Note in Figure 10 that the Autotransfer Controller is enabled in Shared Resources Arbiter states b, c and e. The DMA Controller is enabled in state d. The timing diagrams, which begin on page 7-52, show the relationships between the various signals in Figure 9.

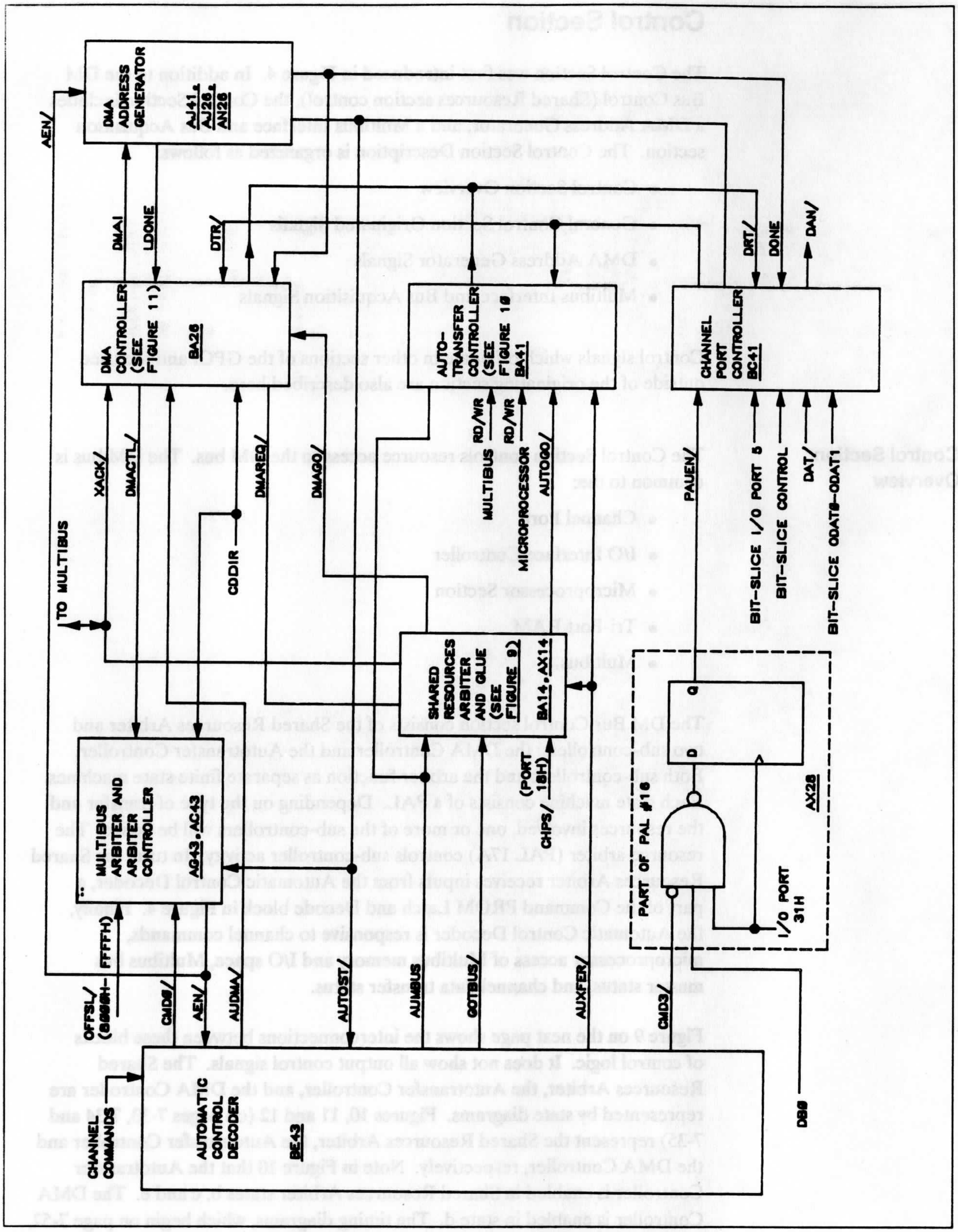


Figure 9. GPCI Control Section Interconnections

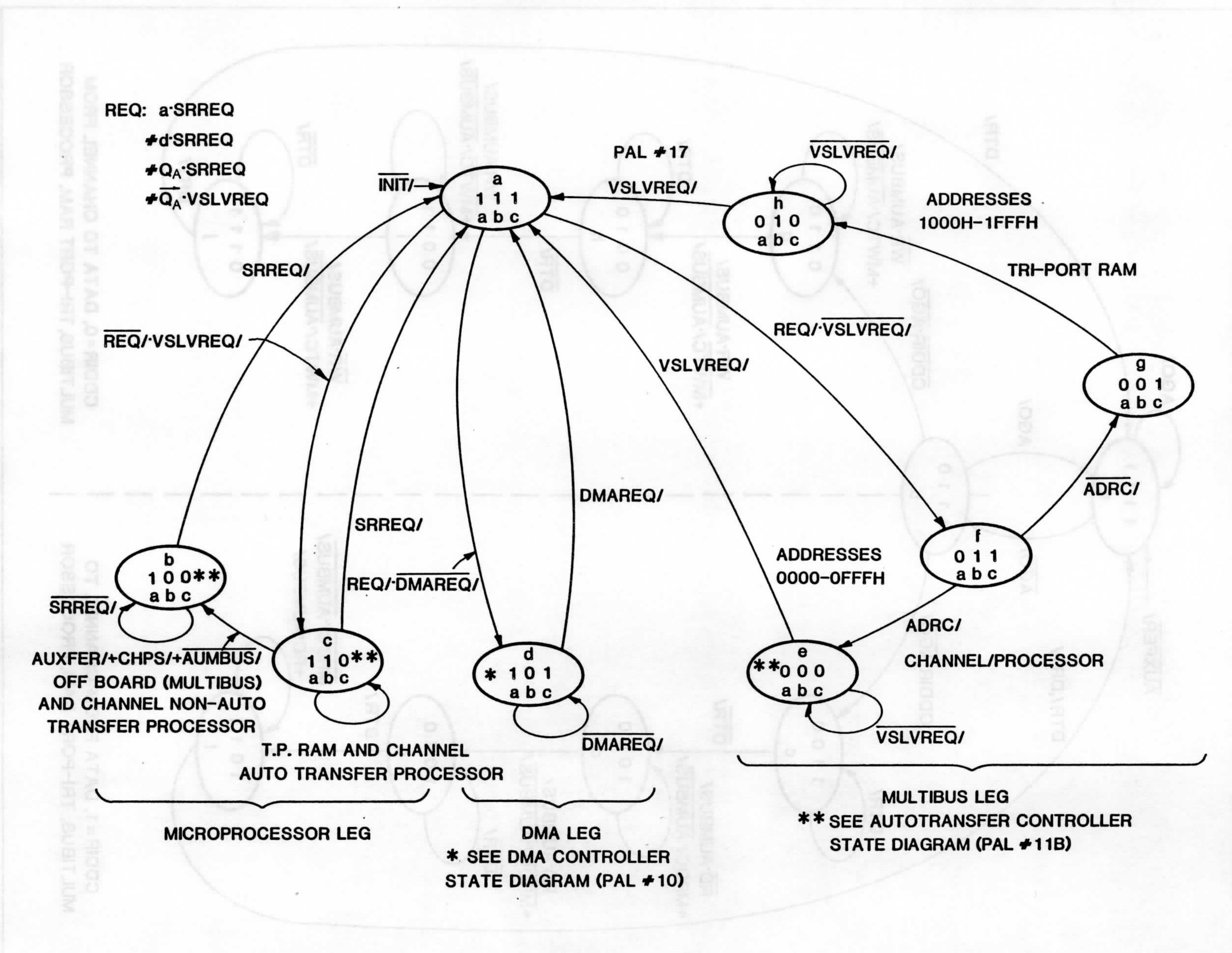


Figure 10. Shared Resources Bus Arbiter (PAL 17)

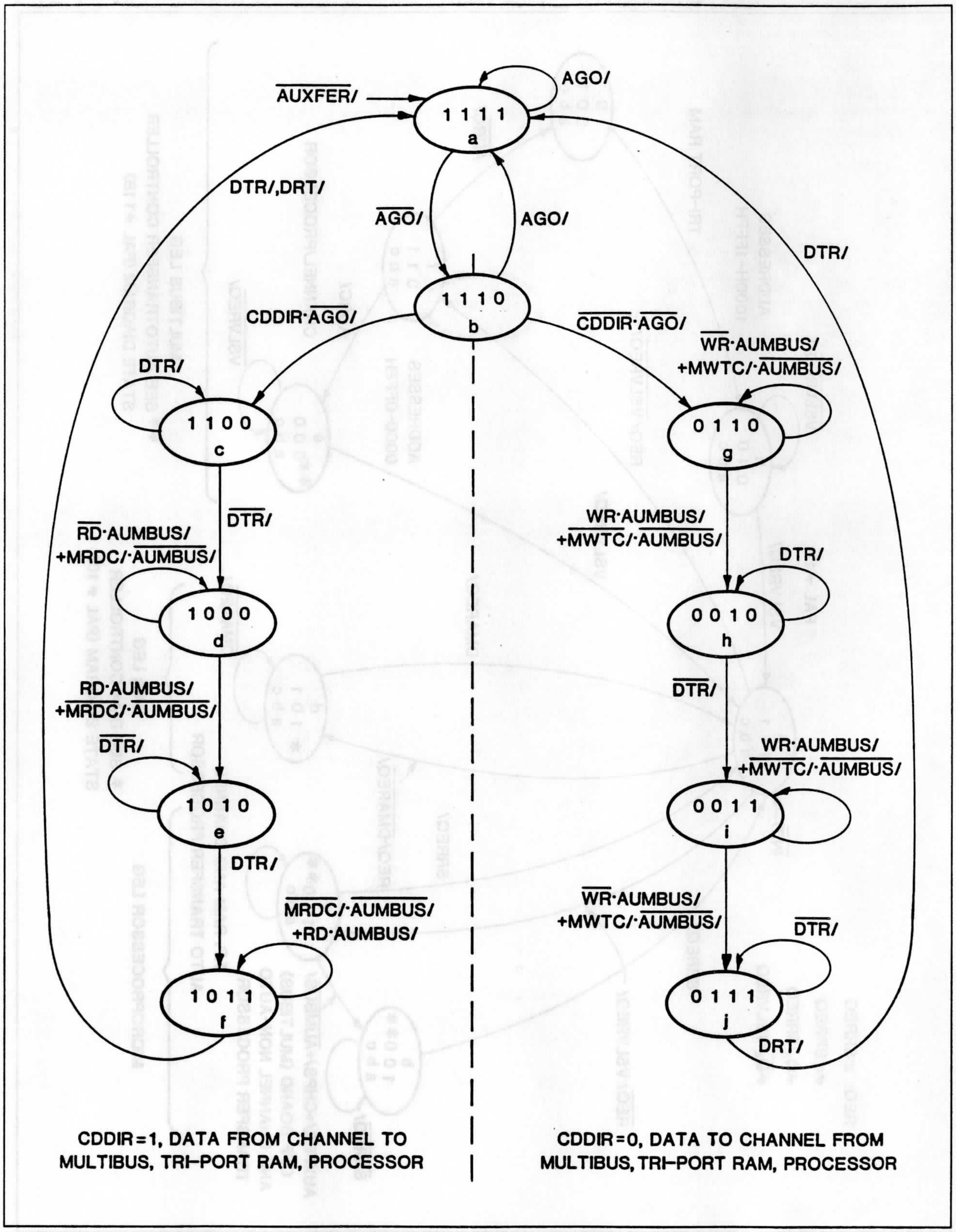

$\overline{AUXFER}$/
AGO/
1 1 1 1
a
DTR/,DRT/
$\overline{AGO}$/
AGO/
DTR/
1 1 1 0
b
CDDIR·$\overline{AGO}$/
$\overline{CDDIR}$·$\overline{AGO}$/
$\overline{WR}$·AUMBUS/
+MWTC/·$\overline{AUMBUS}$/
DTR/
1 1 0 0
c
0 1 1 0
g
$\overline{DTR}$/
$\overline{RD}$·AUMBUS/
+MRDC/·$\overline{AUMBUS}$/
WR·AUMBUS/
+$\overline{MWTC}$/·$\overline{AUMBUS}$/
DTR/
1 0 0 0
d
0 0 1 0
h
RD·AUMBUS/
+$\overline{MRDC}$/·$\overline{AUMBUS}$/
$\overline{DTR}$/
$\overline{DTR}$/
WR·AUMBUS/
+$\overline{MWTC}$/·$\overline{AUMBUS}$/
1 0 1 0
e
0 0 1 1
i
DTR/
$\overline{MRDC}$/·$\overline{AUMBUS}$/
+RD·AUMBUS/
$\overline{WR}$·AUMBUS/
+MWTC/·$\overline{AUMBUS}$/
$\overline{DTR}$/
1 0 1 1
f
0 1 1 1
j
DRT/
CDDIR=1, DATA FROM CHANNEL TO
MULTIBUS, TRI-PORT RAM, PROCESSOR
CDDIR=0, DATA TO CHANNEL FROM
MULTIBUS, TRI-PORT RAM, PROCESSOR

Figure 11. Autotransfer Controller (PAL 11D)

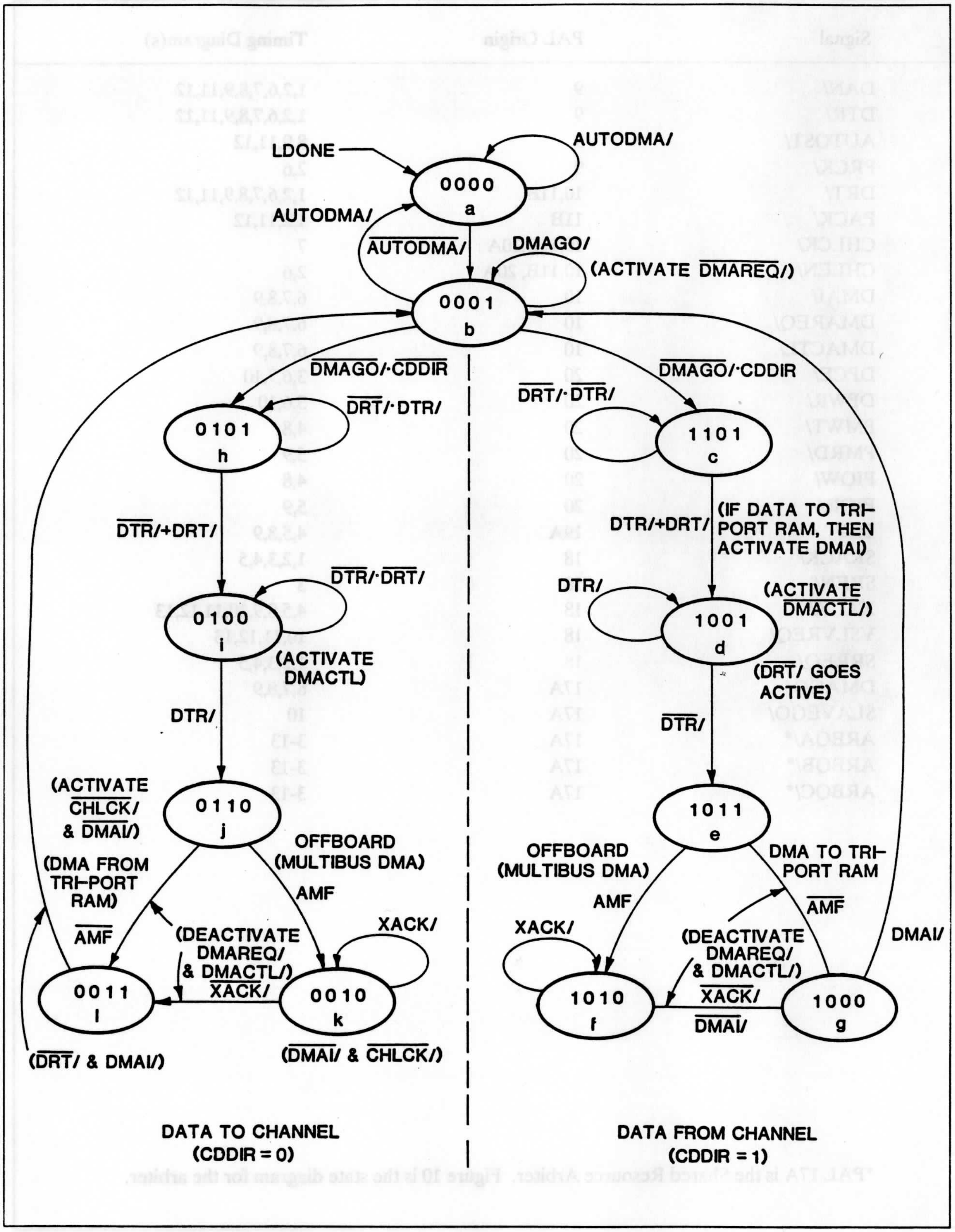

LDONE
AUTODMA/
0000
a
AUTODMA/
$\overline{AUTODMA/}$
DMAGO/
(ACTIVATE $\overline{DMAREQ/}$)
0001
b
$\overline{DMAGO/}$·$\overline{CDDIR}$
$\overline{DMAGO/}$·CDDIR
0101
h
$\overline{DRT/}$·DTR/
$\overline{DRT/}$·$\overline{DTR/}$
1101
c
$\overline{DTR/}$+DRT/
DTR/+DRT/
(IF DATA TO TRI-PORT RAM, THEN ACTIVATE DMAI)
$\overline{DTR/}$·$\overline{DRT/}$
DTR/
0100
i
(ACTIVATE DMACTL)
(ACTIVATE $\overline{DMACTL/}$)
1001
d
($\overline{DRT/}$ GOES ACTIVE)
DTR/
$\overline{DTR/}$
(ACTIVATE $\overline{CHLCK/}$ & $\overline{DMAI/}$)
0110
j
1011
e
(DMA FROM TRI-PORT RAM)
OFFBOARD (MULTIBUS DMA)
OFFBOARD (MULTIBUS DMA)
DMA TO TRI-PORT RAM
AMF
AMF
$\overline{AMF}$
$\overline{AMF}$
XACK/
XACK/
(DEACTIVATE DMAREQ/ & DMACTL/)
(DEACTIVATE DMAREQ/ & DMACTL/)
DMAI/
$\overline{XACK/}$
$\overline{XACK/}$
0011
l
0010
k
1010
f
1000
g
$\overline{DMAI/}$
($\overline{DRT/}$ & DMAI/)
($\overline{DMAI/}$ & $\overline{CHLCK/}$)
DATA TO CHANNEL (CDDIR = 0)
DATA FROM CHANNEL (CDDIR = 1)

Figure 12. DMA Controller (PAL 10-Partial)

Signal	PAL Origin	Timing Diagram(s)
DAN/	9	1,2,6,7,8,9,11,12
DTR/	9	1,2,6,7,8,9,11,12
AUTOST/	9	8,9,11,12
PRCK/	9	2,6
DRT/	10,11B	1,2,6,7,8,9,11,12
PACK/	11B	1,2,11,12
CHLCK/	10,11B,20A	7
CHLEN/	10,11B, 20A	2,6
DMAI/	10	6,7,8,9
DMAREQ/	10	6,7,8,9
DMACTL/	10	6,7,8,9
DPCE/	20	3,6,7,10
DPWR/	20	3,6,10
PMWT/	20	4,8
PMRD/	20	5,9
PIOW/	20	4,8
PIOR/	20	5,9
BS0/	19A	4,5,8,9
SRACK/	18	1,2,3,4,5
SREN/	18	3
XACK/	18	4,5,8,9,10,11,12,13
VSLVREQ/	18	10,11,12,13
SRREQ/	18	1,2,3,4,5
DMAGO/	17A	6,7,8,9
SLAVEGO/	17A	10
ARBQA/*	17A	3-13
ARBQB/*	17A	3-13
ARBQC/*	17A	3-13

*PAL 17A is the Shared Resource Arbiter. Figure 10 is the state diagram for the arbiter.

Table 16. PAL Generated Control Signals Described by Timing Diagrams

Control Section Signals

Most of the Control Section signals are generated by PALs. The PAL equations in the GPCI Supplemental Data Chapter (Chapter 9) describe these signals. Therefore, for those signals generated by PALs and shown on one or more of the Shared Resources Section timing diagrams, no additional discussion is provided other than a listing of the timing diagrams in which they appear. PAL generated control signals that are not shown in one or more timing diagrams are functionally described, but the PAL equations provide the detailed description. Control signals that are not generated by PALs and do not appear on any of the timing diagrams receive the greatest attention here.

Table 16 on the adjacent page lists those control signals that are PAL generated, and appear on one or more of the timing diagrams that begin on page 7-52. It also lists the timing diagram(s) that show the signal.

ARBQA/, ARBQB/ and ARBQC/ define the states of the arbiter, as shown in Table 17.

ARBQA/	ARBQB/	ARBQC/	STATE
1	1	1	a
1	0	0	b
1	1	0	c
1	0	1	d
0	0	0	e
0	1	1	f
0	0	1	g
0	1	0	h

1 = high, 0 = low

Table 17. Shared Resources Arbiter State Definitions

Note in Figure 10 that ARBQA/ is also called SLAVEGO/.

Table 18 on the next page shows those Control Section signals that are PAL generated, but do not appear on any timing diagrams.

Signal	PAL	Function
BDT/R	#19A	Multibus Data Transmit/Receive
BDEN/	#19A	Multibus Data Enable
LCDIR/	#19A	Latched Channel Data Direction
SREN/	#18	Shared Resources Enable
DSGO/	#17A	Dual Port Slave Go
LDONE	#15	Latched DMA Done
HI0/	#15	AJ26 Selection Input
HI1/	#15	AJ26 Selection Input
HI2/	#15	AJ26 Selection Input
LI0/	#15	AN26 Selection Input
LI1/	#15	AN26 Selection Input
LI2/	#15	AN26 Selection Input
ACI/	#15	DMA Address Counter Increment
WCI/	#15	DMA Word Counter Increment

Table 18. PAL Generated Control Signals Not Shown in Timing Diagrams

BDT/R (Multibus Data Transmit/Receive) controls the direction of the data flow through the Data Bus Transceiver. BDT/R is described in the Multibus Interface and Bus Acquisition description.

BDEN/ (Multibus Data Enable) drives the output enables on the Multibus Address Bus and Data Bus Transceivers. This term is low when the GPCI acquires the Multibus (GPCI is bus master), or when an off-board master is granted access to the GPCI (GPCI is bus slave). BDEN/ is described in the Multibus Interface and Bus Acquisition description (page 7-43).

LCDIR/ (Latched Channel Data Direction) is the latched channel direction control (CDDIR).

SREN/ (Shared Resources Enable) is low during shared Resource Arbiter states b and c (see Figure 10 on page 7-33). SREN/ controls the enable pin on the Shared Resources Bus Transceiver (AV17) located in the Microprocessor Section. When SREN/ is low, AV17 can pass data to or receive data from the DM bus.

DSGO/ (Dual-Port Slave Go) is low when the GPCI is acting as a Multibus slave, and an off-board master is performing a transfer to the Tri-Port RAM. This signal goes low upon entry into Shared Resource Arbiter state g, and stays low until VSLVREQ/ goes false (exit from state h). This signal is used in the generation of DPCE/ (Dual-Port Chip Enable) and DPWR/ (Dual-Port Write). These signals drive the Tri-Port RAM's CE/ and W pins respectively (see PAL 20 on sheet 7 of the schematic).

The remaining signals in Table 18 are all part of the DMA Address Generator.

DMA Address Generator

The DMA Address Generator consists of two cascaded 8-bit slice address generators (AJ26 and AN26), and a counter controller (AJ41). These components are shown on sheet 6 of the schematics. Together, AJ26 and AN26 function as a 16-bit programmable DMA address generator. Counter control of the DMA address generator is accomplished by AJ41. AJ41 is a PAL programmed as a 4-state state machine, having an integral state decoder and an instruction code generator.

AJ26 and AN26 are Advanced Micro Device's AM2940 DMA address generator chips. These chips generate sequential memory addresses for use in the sequential transfer of data to and from the IBM channel. They also maintain a word count and generates a DONE signal when a programmable terminal count or address is reached. The AM2940 can be programmed to increment or decrement the address in any of four control modes, and executes eight different instructions. The present version of the 8085 firmware sets the control mode and the address counter mode in accordance with the device involved in the DMA.

In the GPCI's application of the AM2940, AN26 functions as the least significant 8-bit DMA address generator and AJ26 functions as the most significant 8-bit DMA address generator. The two chips are cascaded by connecting the WCO/ and ACO/ (Word Carry Output and Address Carry Output, respectively) of AN26 to the WCI/ and ACI/ (Word Carry Input and Address Carry Input, respectively) of AJ26. Word count and address incrementation are controlled by the inputs to WCI/ and ACI/ on AN26. These inputs are driven by outputs from the controller. The controller also drives the three instruction selection pins (I0, I1 and I2) on each address generator chip.

LI0-LI2 drive instruction selection pins I0-I2 respectively on AN26; HI0-HI2 drive selection pins I0-I2 respectively on AJ26. Table 19 on page 7-41 shows the 13 valid instruction selection codes that can be generated by the controller, and the DMA function performed by each code.

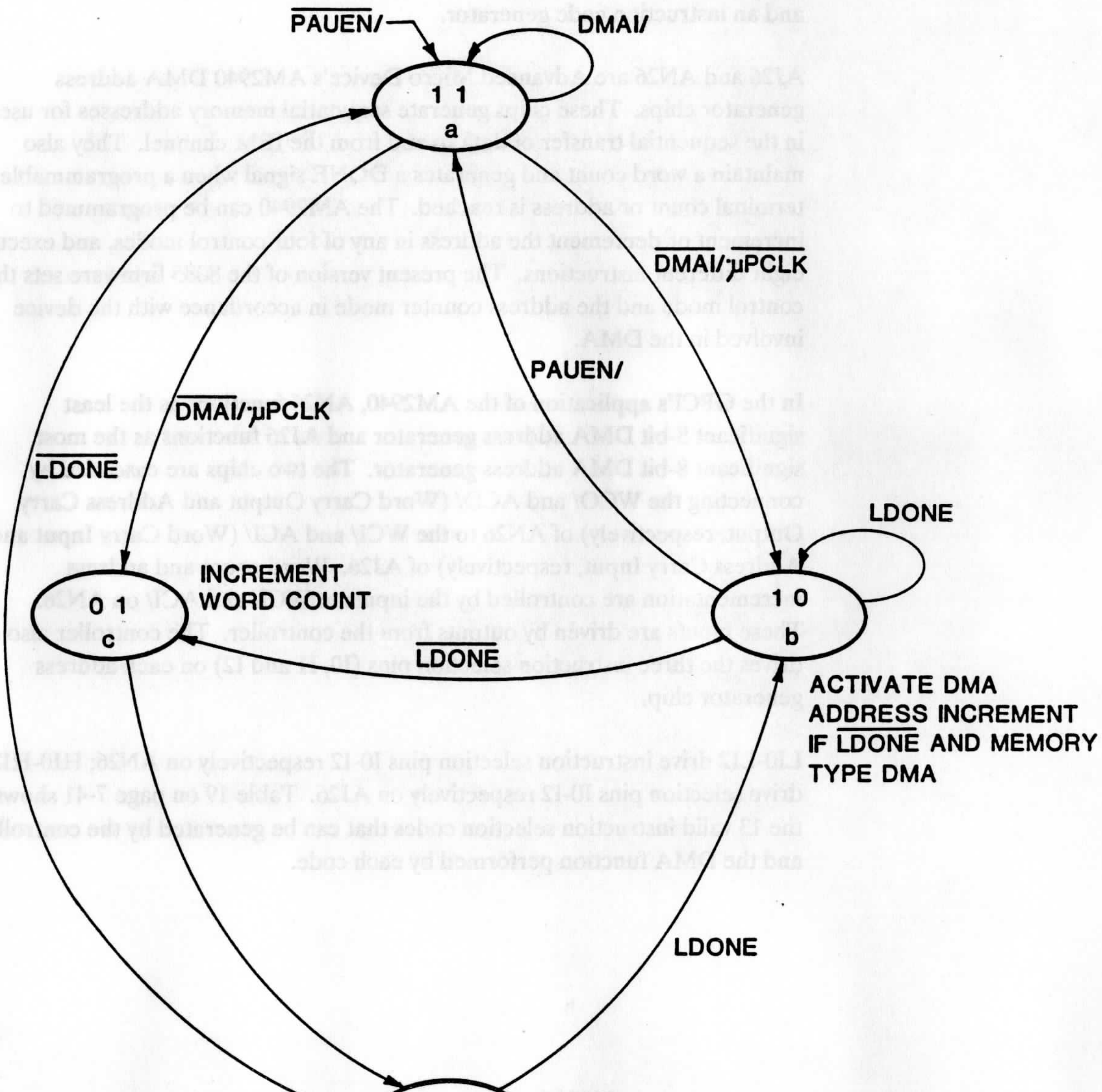


Figure 13. DMA Counter Decode State Diagram (PAL 15)

				H H H L L L
I/O	A A A			I I I I I I
Location	2 1 0	R/W	DMA Function	2 1 0 2 1 0
20	0 0 0	W	Load Control (Low)	H H H L L L
	0 0 0	R	Read Control (Low)	H H H L L H
21	0 0 1	W	Load Word CNT (Low)	H H H H H L
	0 0 1	R	Read Word CNT (Low)	H H H L H L
22	0 1 0	W	Load ADDR CNT (Low)	H H H H L H
	0 1 0	R	Read ADDR CNT (Low)	H H H L H H
23	0 1 1	W	REINIT (Low & High)	H L L H L L
	0 1 1	R	Illegal	
24	1 0 0	W	Load Control (High)	L L L H H H
	1 0 0	R	Read Control (High)	L L H H H H
25	1 0 1	W	Load Word CNT (High)	H H L H H H
	1 0 1	R	Read Word CNT (High)	L H L H H H
26	1 1 0	W	Load ADDR CNT (High)	H L H H H H
	1 1 0	R	Read ADDR CNT (High)	L H H H H H
27	1 1 1	W	Illegal	
	1 1 1	R	Illegal	

"Low" and "High" refer to the respective bytes of the 16-bit address.

Table 19. I/O Map for DMA Counter Control

The state machine portion of the counter controller drives the WCI/ and ACI/ inputs to the DMA generator chips, and captures and latches the DONE signal. Figure 13 on the adjacent page is the state diagram for the state machine. During the following discussion, refer to Figure 13 (DMA Counter Control State Diagram), and Timing Diagrams 14 (page 7-65) and 15 (page 7-66). Timing Diagram 14 shows DMA control during the DMA transfer shown in Timing Diagram 9 (page 7-60). Timing Diagram 15 shows DMA control during transfers shown in Timing Diagrams 6 through 8 (pages 7-57 through 7-59). Note in Timing Diagrams 14 and 15 that WCI/ is always active during state machine states c and d. Also note that ACI/ is active only while performing a memory type DMA transfer, allowing the address output to sequence upward only during a DMA memory transfer. During an I/O DMA transfer, the address output is frozen at its initial value.

Note that the 2940 chip is programmed the same for both I/O and Memory DMA transfers. The difference is in the control signals WCI/ and ACI/.

As shown in Timing Diagrams 6 through 9, it is possible for DMAI/ to occur in either phase of μPCLK. This presents a problem for latching the signal DONE. Because of the manner in which WCI/ is controlled, DONE goes true one address count sooner than desired, and returns false during the last transfer. The desired response is for DONE to go true immediately following the last required increment of the counter, and stay true until reset by an external signal. LDONE (Latched DONE) is formed by testing for DONE being high during state c, and if high, holding the signal until PAUEN/ (Processor Auto Enable) goes high. The processor raises PAUEN/ in recognition of the end of the DMA transfer.

One of the primary functions of the state machine is to delay WCI/ and ACI/ for one 2XCLK clock period if DMAI/ occurs in phase with μPCLK. By doing so, the relationship between WCI and DONE remains constant, regardless of the DMA mode, source, or destination. If the state machine detects DONE while μPCLK is high, it proceeds through state d as if DONE was not true, but then goes to state b. The state machine remains in state b until PAUEN/ goes false. LDONE is an enable input to all terms in the DMA Controller state machine, and causes the state machine to reset to its idle state when LDONE goes true. Since DMAI/ is an output from the DMA Controller, no additional DMAI/ pulses occur after LDONE goes true.

Multibus Interface and Bus Acquisition Signals

Refer to sheet 7 of the schematics. The Multibus Interface and Bus Acquisition block shown in Figure 4 on page 6-3, arbitrates Multibus access in a multi-master environment. This block controls the Address Bus Transceiver, the Data Bus Transceiver and the Multibus control signal interface. The Multibus control interface consists of parts of AX14, AG28 and AE45. Bus Acquisition is accomplished by AG43 and AC43.

Because the Multisourcerer is a multi-master system, Multibus arbitration is required to multiplex the GPCI onto the multi-master Multibus without contention problems. Multibus arbitration is performed by AC43, a Multibus Arbiter IC. Arbitration is based on priority resolution. There are two types of priority resolution schemes, serial and parallel. AC43 performs the priority resolution and can support either scheme, though the serial scheme is presently used. The GPCI is currently the highest priority master. This is established via a jumper between AB42 and AA42, grounding BPRN/ (Bus Priority In). The BPRO/ (Bus Priority Out) signal from AC43 is connected to the BPRN/ of the next highest priority master. This BPRN/ and BPRO/ daisy chaining scheme is extended to all bus masters, with the lowest priority bus master's BPRO/ open.

The CBRQ/ (Common Bus Request) pin from each bus master is wire-ORed together. When a bus master requires the bus, it pulls CBRQ/ low, informing the controlling master that a lower priority master wants the bus. When the controlling master completes its bus operation (if one was in progress), it drives BUSY/ high. BUSY/ is monitored by all bus masters. If BUSY/ is high, the next highest priority bus master gains control of the bus. If this bus master is the requesting master, it is granted access to the bus. If this bus master is not the requesting master, it passes the request to the next master. The requesting master drives BUSY/ low, preventing all other bus masters from controlling the bus until the controlling master completes its operation and drives BUSY/ high. AC43 acquires the Multibus, as described above, when its BS0/ input pin goes low. This pin is driven by AG43, the Bus Arbiter Controller.

AC43 drives AEN/ (Address Enable) low when the GPCI gains control of the Multibus. AEN/ is used by PAL 19A to form BDEN/ (Board Enable described below), and GOTBUS/ (see PAL 5 on sheet 2 of the schematics). GOTBUS/ is used by PAL 18A to form SRREQ/, VSLVREQ/ and SRACK/. SRREQ/ and VSLVREQ/ are input signals to the Shared Resources Arbiter (PAL 17A).

AG43, the Bus Arbiter Controller, drives BS0/ (pin 16) low when the GPCI requests the Multibus (GPCI wants to be the controlling master). When BS0/ goes low, AC43 gains control of the Multibus as described above. AG43 drives BS0/ low under the following conditions:

- the microprocessor attempts a non-DMA memory read or write from or to an application card in the Multisourcerer
- the channel requests a memory or I/O DMA read or write transfer involving an application card in the Multisourcerer

AG43 also produces the Address Bus and Data Bus Transceiver enable and direction controls. BDEN/ (Board Enable) enables both transceivers and the Extended Address Generator. BDEN/ goes low when the GPCI acts as the bus master or bus slave. BDT/R, the Data Bus Transceiver's direction control, goes low if:

- an off-board bus master writes to the GPCI or the channel
- the GPCI is a bus master and the microprocessor reads the Multibus
- the channel performs a DMA read involving the Multibus

The Multibus read/write control signals are:

- MWTC/ (Memory Write Control)
- MRDC/ (Memory Read Control)
- IOWC/ (I/O Write Control)
- IORC/ (I/O Read Control)
- XACK/ (Transfer Acknowledge)

MRDC/, MWTC/ and XACK/ are bidirectional; IOWC/ and IORC/ are unidirectional. As a bus master, the GPCI must drive MWTC/, MRDC/, IOWC/ and IORC/. The GPCI monitors XACK/ when acting as a bus master. As a bus slave, the GPCI responds only to MWTC/ and MRDC/, and acts as a source for XACK/. AG28 generates PMWT/, PMRD/, PIOW/ and PIOR/. These signals are the drive signals for MWTC/, MRDC/, IOWC/ and IORC/ respectively. These signals are inputs to four tri-state line drivers in AE45. These line drivers are active when GOTBUS/ goes low, i.e., the GPCI is the bus master.

CCLK/ (C Clock) and BCLCK/ (B Clock) are Multibus clock signals. All Multibus bus masters must be able to drive these clock signals, though in a particular system only one bus master is allowed to drive these signals. Usually, the highest priority bus master drives the signals. The GPCI is the highest priority bus master and drives the BCLCK/ and CCLK/ signals via NAND gates AE36-D and AE36-C respectively. The NAND gates function as inverters, and are driven by the 9.216 MHz μPCLK clock signal.

I/O Interface Controller Control Signals

The following signals originate in the I/O Interface Controller Section and are used externally to that section:

- EOXL (End of Transmission Latched)
- CDIR/ (Channel Direction)
- CDDIR (Channel Data Direction)
- PRPEN/ (Processor Port Enable)
- RFINT (Register File Interrupt)
- CHLDSC (Channel Disconnect)
- RES/ (Reset)
- CMD0/ (Command Zero)
- CMD3/ (Command Three)
- RFACK/ (Register File Acknowledge)
- CDCLK (Channel Data Clock)

EOXL

During a data transfer between the channel and another resource, the channel can signal the end of the transfer via specified signal sequences on the control lines. The Bit-Slice Processor monitors this condition via the Condition Code Multiplexer. When the Bit-Slice Processor recognizes the end of the transfer, it lowers EOX/ (End of Transfer). EOX/ is an input to PAL 1 located on sheet 2 of the schematic. PAL 1 inverts and latches EOX/ to form EOXL, and holds the signal until PAUEN/ (Processor Auto Enable) goes false. EOXL is an input to the MUART (Microprocessor Section) and the Channel Port controller. The microprocessor monitors EOXL (via the MUART) to determine when the transfer is complete. When the transfer is complete, the microprocessor drives PAUEN/ false, unlatching EOXL. The Channel Port controller uses EOXL to form AUTOST/ (Automatic State). AUTOST/ is used by the Channel Port controller, DMA Address Generator and Multibus interface logic.

CDIR/ and CDDIR

IC BL43-H (sheet 2 of the schematics) inverts CDDIR to produce CDIR/. CDIR/ and CDDIR are used by the Channel Drivers and Receivers board to control data flow through its transceivers. CDIR/ also controls the direction of data flow through BC1 (Octal Inverting Transceiver - see Figure 4). CDDIR is used by the Channel Port Controller to control the direction of data flow through the Bidirectional Channel Port.

PRPEN/

This signal is a direct output of the Microcode PROMs (BL1, located on sheet 1 of the schematics). PRPEN/ allows data that has been latched into the DM side of the Bidirectional Channel Port (BC26) to be enabled onto the DC bus.

RFINT

This signal is an output of the Immediate Tester (PAL 21, located on sheet 2 of the schematics). When the Bit-Slice Processor wants the microprocessor to look in the Register File for new data, it directs PAL 21 to raise RFINT. RFINT is connected to the RST 6.5 interrupt pin on the microprocessor.

CHLDSC

CHLDSC is connected to the RST 7.5 interrupt pin on the microprocessor. This signal is generated by the Bit-Slice Control Logic block (see Figure 6) when the Bit-Slice Processor determines that the Multisourcerer operator has placed the DISCONNECT switch in the "disconnect" position. The interrupt informs the microprocessor of the pending disconnect request, causing the microprocessor to perform various shutdown processes on active application cards.

RES/

RES/ is produced by wire-ORing PURST/ (Power Up Reset) and INIT/. PURST/ is generated on the Channel Drivers and Receivers board and is described in that chapter. INIT/ is the Multibus initialization signal. RES/:

- clears the Microcode PROM registers
- forms PACK/ (see PAL 11D on Sheet 3)
- resets the microprocessor
- resets the Shared Resources Arbiter (PAL 17A)
- resets the Extended Address and Handshake Generator (PAL 16)

CMD0/

This signal is generated by BE43, the Automatic Control Decoder portion of the Command PROM Latch and Decode logic (see Figures 4 and 6). CMD0/ is the output of a latch strobed by CPEN/ (Command PROM Enable). The data input to the latch is ID0. When CPEN/ is true, the ID0 line is driven by the LSB output of the Command PROM. Thus, CMD0/ is the latched and inverted LSB of the most recent command lookup code. CMD0/ is an input to the DMA Address Generator, and the Multibus Interface and Bus Acquisition logic.

CMD3/

This signal is generated in a manner similar to that of CMD0/ above, except the data input is ID3. CMD3/ indicates a data transfer channel command. It is used by PAL 16 (sheet 7) to form PAUEN/. PAUEN/ is active only during data transfer commands.

RFACK/

This signal is an output from BG16 in the Register File Control portion of the Register File (see Figure 6). When the microprocessor reads data from or writes data to the Register File, an unspecified delay may occur as a result of contention with the Bit-Slice Processor. As a result, the microprocessor may require forced wait states to prevent it from continuing on to its next instruction before its present instruction is successfully completed. The microprocessor enters wait states if its RDY pin is low. RDY is driven by the Ready Generator (BA1). One of the inputs to the Ready Generator is RFACK/. The Register File State Machine (PAL 6) generates RFACK/ only after the write process is completed, or read data is valid and stable.

CDCLK/

This signal is generated by a Microcode PROM (PR4). It clocks the channel command into the Command Latch portion of the Command Latch and Decode logic (see Figure 6 and sheet 2). CDCLK/ also clocks data into the Channel Port Controller (BC41).

Microprocessor Section Control Signals

The following control signals originate in the Microprocessor Section:

- RFCS/ (Register File Chip Select)
- DPCS/ (Dual Port Chip Select)
- XIOCS/ (Transfer I/O Chip Select)
- DMACS/ (DMA Chip Select)
- CHPS/ (Channel/Port Select)
- BLOK (BUS LOCK)
- DPLCK/ (Dual Port Lock)

RFCS/

This signal is an output of the Memory Address Decode chip (AV1 - see sheet 5 and Figure 8 on page 7-19), and is mapped to memory addresses 6800H-69FFH. RFCS/ is an input to the Register File state machine (PAL 6A, located at BG43), and must be low for the state machine to enter the microprocessor read/write loops (see Figure 7 on page 7-12).

DPCS/

This signal is an output of the Memory Address Decode chip (AV1 - see sheet 5 and Figure 8), and is mapped to memory addresses 7000H-7FFFH. DPCS/ is used to form DPCE/ (Dual-Port Chip Enable), the Tri-Port RAM's Chip enable. DPCE/ is required during microprocessor write and read operations involving the Tri-Port RAM. DPCS/ is also applied to PAL 18A to form SRREQ/ (Shared Resources Request). SRREQ/ is a control signal used by the Shared Resources Arbiter.

XIOCS/

This signal is an output of the I/O Port Decode chip (AX1 - see sheet 5 and Figure 6 on page 7-3), and is mapped to I/O port addresses 30H-37H. XIOCS/ is used as a latch clock by the Extended Address and Handshake Generator (PAL 16 - see sheet 7). Refer to the Extended Address and Handshake Generator description (page 7-29) for more information.

DMACS/

This signal is an output of the I/O Port Decode chip (PAL 14, located on sheet 5), and is mapped to I/O port addresses 20H-2FH. DMACS/ is an input to the DMA Address Generator Controller (PAL 15 - see sheet 6). DMACS/ must be low for the microprocessor to program LI0-LI2 and HI0-HI2, which are instruction select lines used by the DMA Address Generator chips. For more information, refer to the DMA Address Generator description (page 7-39).

CHPS/

This signal is an output of the I/O Port Decode chip (PAL 14, located on sheet 5), and is mapped to I/O port addresses 18H-1FH. CHPS/ is used by the Shared Resources Arbiter (PAL 17A) during a microprocessor/channel data transfer. Refer to Timing Diagrams 1 and 2 on pages 7-52 and 7-53, and Figures 8 and 9 on pages 7-19 and 7-32.

BLOK

BLOK is driven by the microprocessor's SOD (Serial Output Data) pin, and is controlled by firmware. This signal is applied to AC43 pin 16 (LOCK/). AC43 is the Multibus bus arbiter. BLOK prevents the GPCI from surrendering the Multibus to any other bus master, regardless of bus priority.

DPLCK/

DPLCK/ is asserted by the 8085 when it writes data to I/O port 31H that drives DB2 high. When asserted, this signal prevents the Multibus from accessing the Tri-Port RAM.

Handshake Generator Control Signals

Control signals that originate in the Multibus Interface Section are generated by the Handshake Generator portion of the Extended Address and Handshake Generator block (see Figure 4 on page 6-3).

The four control signals from the Handshake Generator section (PAL 16) are:

- PAUEN/ (Processor Auto Enable)
- RFINTA (Register File Interrupt Acknowledge)
- CTL2 (Control Latch 2)
- DPLCK/ (Dual Port Lock)

These control signals are programmed by data on the five LSBs of the DB bus while writing to port 31H. See Table 20 below.

DB4	DB3	DB2	DB1	DB0	Control Signal
0	0	0	0	0	(not used)
0	0	0	0	1	*PAUEN/
0	0	0	1	0	(not used)
0	0	1	0	0	DPLCK/
0	1	0	0	0	RFINTA
1	0	0	0	0	CTL2

* The state of PAUEN/ is determined by CMD3/. If CMD3/ is low, PAUEN/ will go true (low). CMD3/ is low while processing a data transfer command. Refer to the Command PROM Latch and Decode Section description (pages 7-9 and 7-10).

Table 20. Handshake Generator Programming

PAUEN/ is an input to the Condition Code Multiplexer, the Channel Port Controller and the DMA Address Generator. PAUEN/ is the microprocessor's acknowledgement of a data transfer command from the channel. It informs other sections of the GPCI that the microprocessor is ready to support the transfer. The Bit-Slice Processor detects this condition by monitoring the PAUEN/ input to the Condition Code Multiplexer.

When PAUEN/ goes low, the Channel Port Controller terminates the data transfer sequence. The Channel Port Controller combines PAUEN/ with three other transfer terminator signals to produce AUTOST/ (Automatic State), one of the main control signals used by the Channel Port Controller. For additional information on AUTOST/, refer to the timing diagrams that start on the next page. Finally, PAUEN/ serves as an enable for the DMA Address Generator's state machine (refer to the DMA Address Generator description on pages 7-39 through 7-42).

The Tri-Port RAM was originally called the "Dual-Port RAM." DPLCK/ (Dual Port Lock) is an acronym based on the Tri-Port RAM's former name for a control signal that can lock out the Multibus from accessing the Tri-Port RAM. Also, when DPLCK/ is low (Multibus cannot access the Tri-Port RAM), it enables the transfer acknowledge signal (XACK/) each time the Multibus attempts to read from or write to the Tri-Port RAM.

RFINTA is the microprocessor's acknowledgment of an interrupt generated by the Register File state machine. When the Bit-Slice Processor has new status information to pass to the microprocessor it loads the status into the appropriate cell in the Register File and interrupts the microprocessor. Then the Bit-Slice Processor enters a loop that monitors for an acknowledgment of that interrupt. RFINTA/ (Register File Interrupt Acknowledge) is the microprocessor's acknowledgment of the Bit-Slice Processor's interrupt. RFINTA/ is applied to the Condition Code Multiplexer, where it can be tested by the Bit-Slice Processor. When RFINTA/ goes low, it informs the Bit-Slice Processor that the status has been read.

At the time this manual went to press, the Timing Diagrams were not released from Engineering. Therefore, we could not include them in this document. When they are released, we will forward them to you. We apologize for any inconvenience this may cause.

GPCI Bit-Slice Microcode Description

GPCI Bit-Slice Microcode Description

The I/O Interface Controller causes the IBM channel to appear to the remainder of the GPCI as an I/O port. The I/O Interface Controller performs those IBM channel-specific interface activities required for bidirectional transmission of data and status. The I/O Interface Controller is under control of the microcode stored in five PROMs.

The GPCI and the Multisourcerer's application cards function as an IBM I/O control unit and I/O devices respectively. The *IBM System/360 and System/370 I/O Interface Channel to Control Unit Original Equipment Manufacturers Information Manual* (hereafter referred to as the "I/O Reference Manual") describes the interface requirements between the IBM channel and control units (i.e., GPCI). The GPCI's hardware and microcode are in compliance with this reference.

Each Multisourcerer is assigned a contiguous block of 16 I/O addresses by assigning it the MSD of a two-digit hexadecimal I/O address. The MSD is referred to as the "base address," while the LSD is the "device address." Application cards are assigned a unique device address (0H-EH); device address FH is reserved for the GPCI. The device address forms the LSD of the I/O address block. The GPCI can service up to 15 I/O devices.

The Bit-Slice Processor contains 16 bytes of internal RAM contained in the ALU. This RAM is referenced throughout this description as "Register x" (Register 0-Register FH). These registers are defined in Table 21 on the next page.

Registers	Definition	Register	Definition
Register 0	Device State	Register 8	Extra Status
Register 1	General Purpose Write	Register 9	Device Type
Register 2	Command Register Address	Register A	Device Address
Register 3	Command Code	Register B	Base Address
Register 4	Channel Control (Low)	Register C	Not Assigned
	3 ADDRIN	Register D	Polling Address
	2 OPIN		
	1 REQIN	Register E	State Flags
	0 PSOUT		3 Undefined
			2 Address Enable
Register 5	Channel Control (High)		1 Supress Enable
	3 SL3 (Relay)		0 Shout Enable
	2 SL2 (Relay)		
	1 SL1 (Relay)	Register F	State Flags
	0 STATIN		3 General Purpose Flag*
			2 Disconnecting
Register 6	Status (Low)		1 Stacked Status**
	3 Channel End		0 Pending**
	2 Device End		
	1 Unit Check		
	0 Unit Exception		
Register 7	Status (High)		
	3 Attention		
	2 Status Modifier		
	1 Control-Unit End		
	0 Busy		

* This flag has two different meanings depending on context. In the write data routine, this flag is set to indicate that a parity error is to be reported if detected. Elsewhere in the code the flag indicates that a system reset is being performed.

** When both of these are set it means the previous command had indicated chaining. The Bit-Slice processor suspends the polling loop and waits for the next command from the Channel.

Table 21. GPCI Bit-Slice Register Definitions

Communication between the I/O Interface Controller Section and the Device Controller Section is via the dual port Register File. The Register File allows status to be passed from one section to the other. The Register File is defined in Table 22 on the next page and provides two bytes (four nibbles) of status for each device. Though the Register File is accessible by both sections, all reference to particular locations within the Register File is by its Microprocessor Section address (6800H-681FH upper or lower nibble). Table 22 shows the Microprocessor Section address mapping for the Register file.

Actual data transfers from and to the channel are not handled directly by the Bit-Slice Processor. The data transfers are accomplished by hardware. The Bit-Slice Processor monitors these transfers and controls pre-transfer and post-transfer sequences as well as fault sequences before, during and after the transfers.

This discussion does not attempt to document the microcode on an instruction by instruction basis. This approach would be tedious and excessively detailed. Instead, a Functional Processing Flow analysis (Refer to Figure 14 on page 8-6), a Detailed Processing Flow analysis and microinstruction listings (located in the GPCI Supplemental Data Chapter) are provided. Also provided is a brief description of each routine and subroutine.

Microprocessor Address	Bit-Slice Register	Definition	
6800H-680EH	Registers 0-E	Processor Status	
		3 AT (Attention)	
		2 DE (Device End)	
		1 UC (Unit Check)	
		0 UE (Unit Exception)	
		Device State	
		3 DS3	
		3 DS3	
		2 DS2	
		1 DS1	
		0 DS0	
6810H-681EH		Extra Status	
		3 CE (Stacked Channel End)	
		2 BR (Busy Reported)	
		1 UC (Stacked Unit Check)	
		0 N/A	
		Device Type	
		3 DT3	00 = Bisync
		2 DT2	01 = Ingestor
		1 DT1	02 = ProNET
		0 DT0	
680FH	Register F	Device Modifier	
		3 DM3	
		2 DM2	
		1 DM1	
		0 DM0	
		Device Address	
		3 DA3	
		2 DA2	
		1 DA1	
		0 DA0	
681FH		Auxiliary State *	
		3 Command Reject	
		2 Intervention Required	
		1 Bus-out Check	
		0 Equipment Check	
		Channel Address (Read Only)	
		3 CA3	
		2 CA2	
		1 CA1	
		0 CA0	

* When the entire nibble equals zero, the meaning of the abnormal indication is to perform a selective reset on the specified device.

Table 22. GPCI Register File Bit Map

GPCI Bit-Slice Microcode Functional Description

Refer to Figure 14 on the next page. When power is first applied or the RESET switch on the front panel of the Multisourcerer is pressed, the I/O Interface Controller begins executing HSTART (Hard Start). HSTART initializes the ALU registers and the Device States located in the Register File. Then, HSTART exits to WATCON (Wait for Connect to Channel) which monitors the front panel switch marked DISCONNECT. As documented in pages 2-21 through 2-23 of the I/O Reference Manual, connection to and disconnection from the IBM channel must not disrupt other devices connected to the channel. Also, you should perform disconnections only after all pending I/O device activity is completed.

The status of the DISCONNECT switch informs the I/O Interface Controller of the operator's intentions. WATCON prevents the I/O Interface Controller from activating any channel tag or control line until after the DISCONNECT switch has been switched to the "connect" position. Then WATCON sequences three relays on the Channel Drivers and Receivers board. These relays connect the GPCI to the channel without causing disruption of any channel tag or control signals. This process is referred to as a physical connection. After the connecting process is completed, WATCON exits to STPOL.

STPOL (polling loop) is the I/O Interface Controller's idle routine. It performs two major functions. First, it monitors the channel's control lines to determine if the channel is requesting an Initial Selection Sequence or acknowledging a control unit's request for service. Second, STPOL sequentially tests (polls) the device states stored in the Register File for requested service.

Device State monitoring is not as time critical as channel monitoring. Therefore, Device State monitoring consists of fetching a particular device state from the Register File, testing the device state for requested service and, if service is requested, beginning a Control Unit Initiated Sequence by activating the Request In line. Until the requested services are completed, the polling address is frozen, preventing further testing of device states. Once the requested service has been completed, STPOL increases the polling address and repeats the process described above for the next device.

To achieve a high data throughput, channel service requests must be handled rapidly. The Bit-Slice Processor has no true interrupt capability and must, therefore, emulate this feature. Programmable hardware monitors several channel control lines and generates a single output (RESPIM/) which can be monitored by the Condition Code Multiplexer. RESPIM/ is tested every few lines within the STPOL code. If RESPIM/ is low, subroutine IMRES is invoked to interpret the channel's control line activity. Because of the Bit-Slice Processor's speed and the frequency with which RESPIM/ is tested, the Bit-Slice Processor performs as if it was interrupt driven.

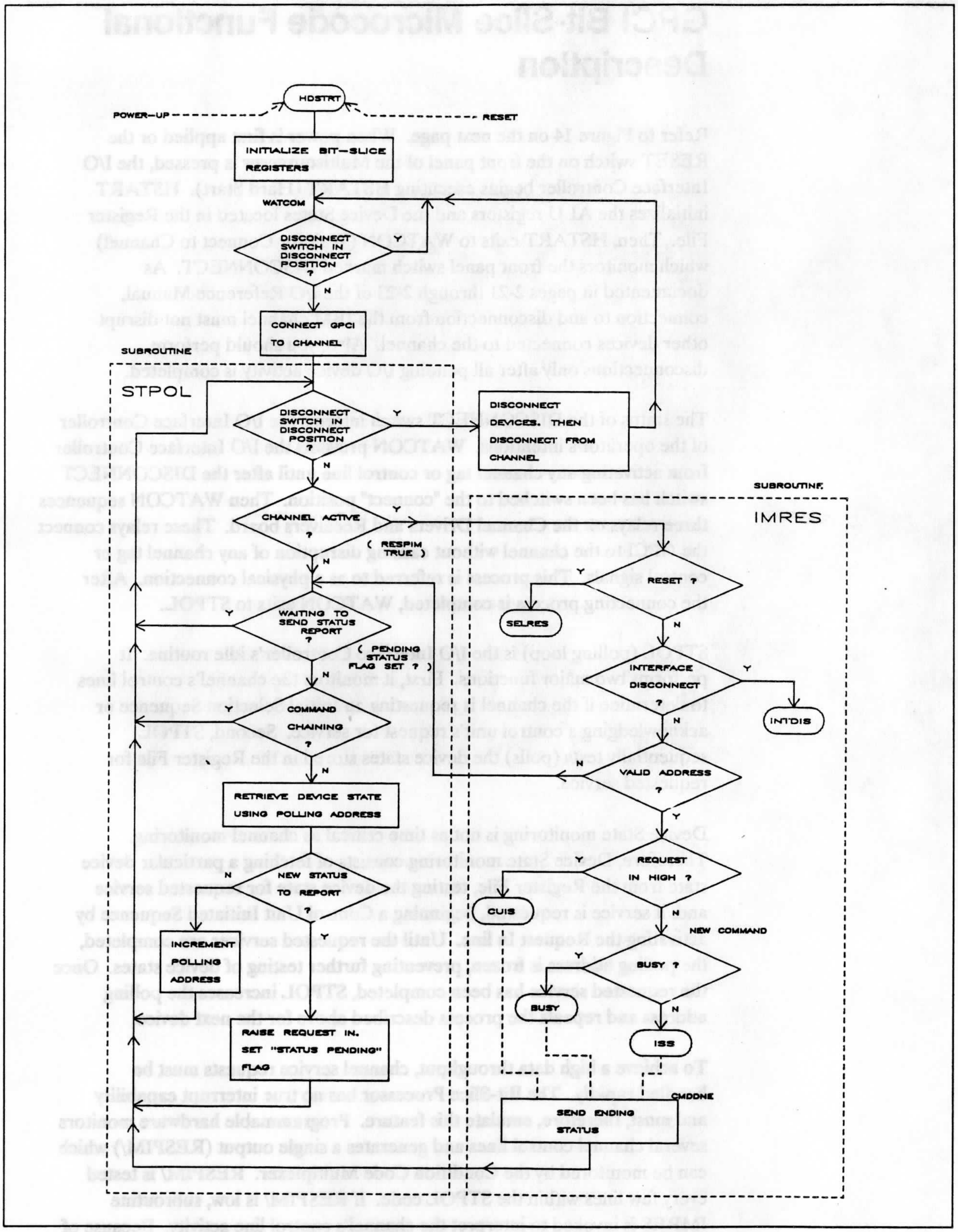

HDSTRT
POWER-UP
RESET
INITIALIZE BIT-SLICE REGISTERS
WATCOM
DISCONNECT SWITCH IN DISCONNECT POSITION ?
Y
N
CONNECT GPCI TO CHANNEL
SUBROUTINE
STPOL
DISCONNECT SWITCH IN DISCONNECT POSITION ?
Y
N
DISCONNECT DEVICES, THEN DISCONNECT FROM CHANNEL
CHANNEL ACTIVE ?
(RESPIM TRUE)
N
WAITING TO SEND STATUS REPORT ?
Y
(PENDING STATUS FLAG SET ?)
COMMAND CHAINING ?
Y
N
RETRIEVE DEVICE STATE USING POLLING ADDRESS
NEW STATUS TO REPORT ?
N
Y
INCREMENT POLLING ADDRESS
RAISE REQUEST IN. SET "STATUS PENDING" FLAG
SUBROUTINE
IMRES
RESET ?
Y
SELRES
N
INTERFACE DISCONNECT ?
Y
INTDIS
N
VALID ADDRESS ?
N
Y
REQUEST IN HIGH ?
Y
CUIS
N
NEW COMMAND
BUSY ?
Y
BUSY
N
ISS
CMDDNE
SEND ENDING STATUS

Figure 14. Functional Processing Flow Diagram

When the channel wants to send a command to one of the devices, it places the base and device addresses on the Bus Out lines and raises Address Out, Select Out and Hold Out. The rise of Select Out and Hold Out causes RESPIM/ to go active, which calls IMRES. Also, before the GPCI sends data to the channel, it raises Request In. The channel responds to the raised Request In by raising Select Out and Hold Out. Thus, regardless of who initiates a request for communications, Select Out and Hold Out rise, calling IMRES. The hardware that generates RESPIM/ also monitors the channel control lines for Selective Resets (resets the connected device), System Resets (resets all devices), and Interface Disconnect (channel terminates an ongoing transaction) sequences. Each of these sequences causes RESPIM/ to go low, thus envoking IMRES. Thus, IMRES must perform several tests to determine the exact cause of RESPIM/.

Upon entry to IMRES, a test for resets is made. All resets are handled by SELRES (Selective Reset) which does additional testing and passes system resets to SYSRES for processing. After testing for resets, IMRES tests for Interface Disconnects. If the test passes, INTDIS is called. INTDIS freezes the status and informs the microprocessor of the event with an interrupt.

Then, IMRES checks Request In. If this line is high, the channel is responding to a GPCI request for communications. This response is processed by CUIS (Control Unit Initiated Sequence). Next, IMRES tests for a channel request for communications. For this request to be valid, the base address must be valid and the device cannot be in a Busy state. If the base address is invalid, IMRES passes Select Out to the next control unit on the channel and returns. If the base address is valid, but the device is Busy, processing is passed to BUSY. BUSY checks the actual command and the device's device state to determine what action should be taken.

Knowledge of device states and device state codes is crucial to understanding the Bit-Slice Processor's microcode. These codes are listed in Table 23 below and defined in the following paragraphs.

Code (Binary)		
BIT 3210	Definition	Set By
0000	Device Available	B-S (Bit-Slice)
0010	Going Ready	U-P (Microprocessor)
0011	Going Not Ready	U-P
0100	Pending Status	U-P
0101	Pending Status Request	U-P
0110	Stacked Status	B-S
0111	Stacked Status Request	B-S
1000	Device Busy	B-S
1010	Device Not Ready	U-P
1011	Not Present (power up)	B-S

Table 23. The Device State Table

Device Available (code 0000B)

This is the quiescent (not busy) state for an I/O device. A device must be in this state to accept a new command. This state is set at the completion of a command after the ending status is sent to the channel.

Going Ready (code 0010B)

When a device receives a Selective Reset, that device goes to a Busy state (code 1000B) until the reset process is complete. Upon completion, the microprocessor sets the Going Ready state or Going Not Ready (0011B) state. Unless the device was in a Device Not Ready state before the reset sequence was received, the Going Ready state is set.

A device can receive a new command during reset. If this happens, BUSY sends a status of Busy to the channel, and sets the Busy Reported flag. The channel suspends all communications to that device until it receives another status of Device End. The Device End status reactivates the communications path for that device.

The polling loop (STPOL) uses the device state of Going Ready or Going Not Ready to direct processing into a special branch of STPOL that tests the Busy Reported flag. If the Busy Reported flag is set, it means the Busy status was transmitted during the reset. If a Busy status was sent, STPOL sets a Pending Status device state if the device state was Going Ready, or a Pending Status Request if the device state was Going Not Ready. The Pending Status or Pending Status Request causes STPOL to raise Request In during the next cycle through STPOL. This results in the initiation of a CUIS sequence which sends the Device End status to the channel.

Going Not Ready (code 0011B)

This state is generated by the microprocessor at the completion of a Selective Reset to a device in the Device Not Ready state. Refer to the Going Ready description above.

Pending Status (code 0100B)

All commands require a Channel End and a Device End status to complete the command. In the GPCI, the I/O Interface Controller controls the interface to the channel, and the microprocessor controls the interface to the devices. The I/O Interface Controller has no way of knowing when a device actually finishes a command, except by checking the device state. However, the I/O Interface Controller determines when use of the channel is no longer required for execution of the command. Therefore, the I/O Interface Controller supplies the Channel End status.

When the channel receives the Channel End status, it disconnects from the device. The channel does not attempt to communicate with any device which has not reported a Device End status. In other words, as far as the channel is concerned, the device is busy. However, the channel is available for communication with other devices.

When the microprocessor determines that a command (data transfer, etc.) is complete, it informs the I/O Interface Controller that a Device End must be sent. To do this, the microprocessor sets the device state in the Register File to Pending Status, and sets the status in the Register File to Device End. When STPOL discovers a device with a Pending Status, it raises Request In and sets the Pending Status flag. This initiates a CUIS cycle which results in transmission of a Device End status to the channel or a Stacked Status device state.

The Stacked Status device state is set if the channel indicates Busy when the CUIS cycle attempts to send the status. If this happens, STPOL waits until the channel is not busy and tries to send the Device End again. When the Device End status transmission is successful, CMDDNE changes the device state to Device Available.

Pending Status Request (code 0101B)

Refer to the Going Not Ready description above.

Stacked Status (code 0110B)

Refer to the Pending Status description above.

Stacked Status Request (code 0111B)

This state is set by CMDDNE when attempting to send a status of Device End to a channel while it is busy.

Device Busy (code 1000B)

With the exception of the Test I/O command, all commands result in a device state of Device Busy. This state is set by ISS. The microprocessor changes the state to Pending Status as described in the Pending Status description above.

Device Not Ready (code 1010B)

This state is set by the microprocessor when a device is present but is not yet ready to process commands. It can provide status to the channel upon request (via a Test I/O command), and it can be selectively reset. Refer to the Going Not Ready description above.

Device Not Present (code 1011B)

The device state of all devices is initially set to Device Not Present during power-up. The microprocessor examines the Non-Volatile RAM (NVRAM) to determine which devices are present. The device state of those devices which are present is then changed to Device Not Ready.

Detailed Bit-Slice Microcode Description

The Detailed Bit-Slice Microcode Description is based on Figures 15 and 16 (sheets 1-3). Figure 15 consists of microcode initialization and STPOL. Figure 16 consists of IMRES.

Routine HSTART (Hard Start)

Refer to Figure 15 on the adjacent page. First, HSTART sets the following ALU registers to zero:

- Register 4 (Channel Control-low)
- Register 5 (Channel Control-high)
- Register 6 (Status-low)
- Register 7 (Status-high)
- Register 8 (Extra Status)
- Register A (Device Address)
- Register B (Base Address)
- Register D (Polling Address)
- Register E (State Flags)
- Register F (State Flags)

Next, the Bus In and Tag In lines are lowered by writing the contents of the Status and Control registers to the Bus In and Tag In latches respectively. Then, HSTART loads a value of BH into each device state nibble (upper nibble of 6800H-680EH) of the Register File. A device state of BH is defined as Not Present.

The microprocessor determines what devices are available by examining its Non-Volatile RAM (NVRAM - refer to the Microprocessor Section Detailed Circuit Description). The microprocessor resets each available I/O device, and, after a successful reset, modifies the device states of those devices to Pending Status. The microprocessor sets the status to Attention. The Attention Status is passed to the channel by STPOL, informing the channel of available devices (refer to STPOL description below). HSTART unconditionally jumps to the routine WATCON (Wait for Connect to Channel).

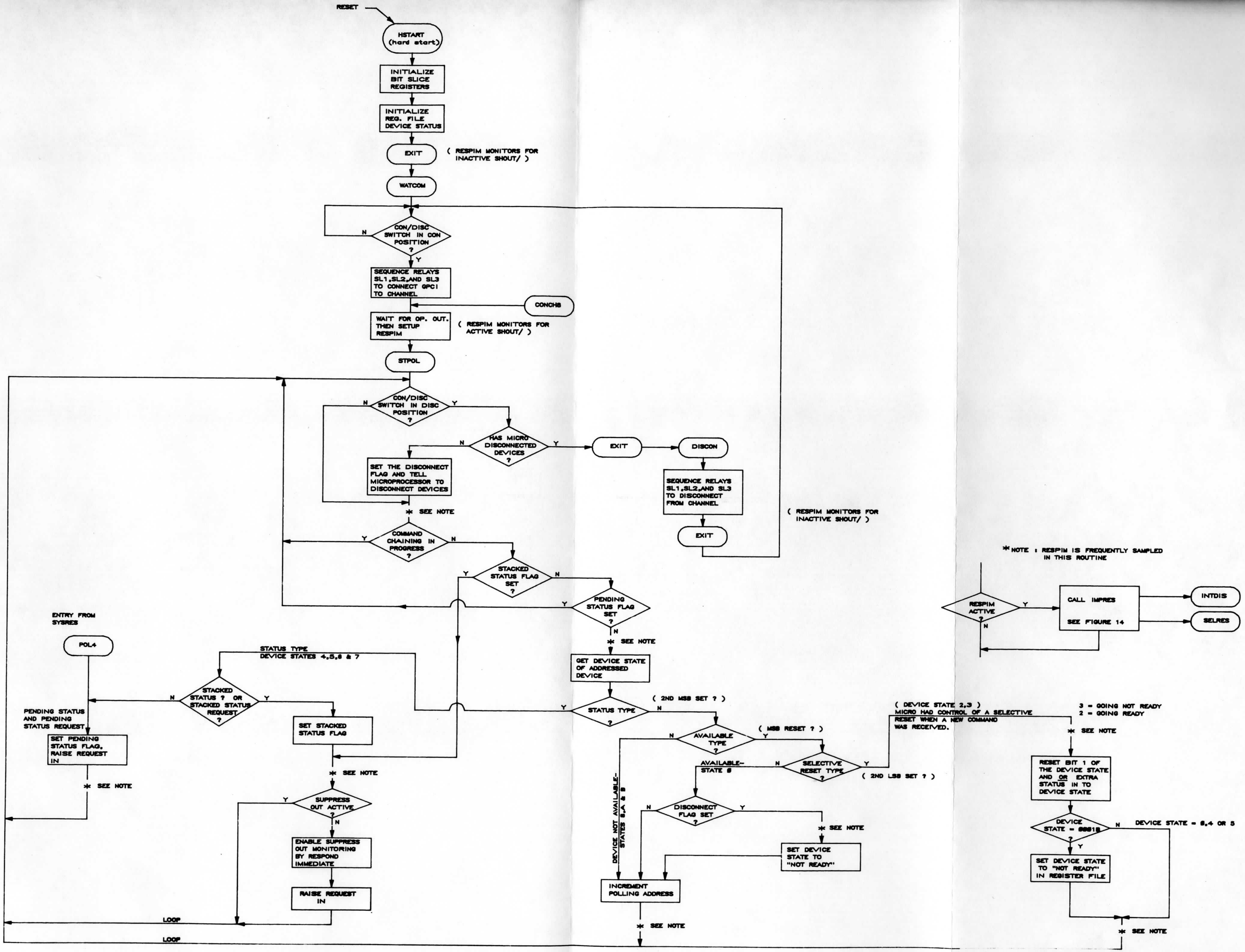

RESET
HSTART (hard start)
INITIALIZE BIT SLICE REGISTERS
INITIALIZE REG. FILE DEVICE STATUS
EXIT
(RESPIM MONITORS FOR INACTIVE SHOUT/)
WATCOM
CON/DISC SWITCH IN CON POSITION ?
N
Y
SEQUENCE RELAYS SL1,SL2,AND SL3 TO CONNECT GPCI TO CHANNEL
CONCHS
WAIT FOR OP. OUT. THEN SETUP RESPIM
(RESPIM MONITORS FOR ACTIVE SHOUT/)
STPOL
CON/DISC SWITCH IN DISC POSITION ?
N
Y
HAS MICRO DISCONNECTED DEVICES ?
N
Y
EXIT
DISCON
SEQUENCE RELAYS SL1,SL2,AND SL3 TO DISCONNECT FROM CHANNEL
(RESPIM MONITORS FOR INACTIVE SHOUT/)
EXIT
SET THE DISCONNECT FLAG AND TELL MICROPROCESSOR TO DISCONNECT DEVICES
* SEE NOTE
COMMAND CHAINING IN PROGRESS ?
Y
N
STACKED STATUS FLAG SET ?
Y
N
PENDING STATUS FLAG SET ?
Y
N
* SEE NOTE
GET DEVICE STATE OF ADDRESSED DEVICE
ENTRY FROM SYSRES
POL4
STATUS TYPE
DEVICE STATES 4,5,6 & 7
STACKED STATUS ? OR STACKED STATUS REQUEST ?
N
Y
PENDING STATUS AND PENDING STATUS REQUEST
SET PENDING STATUS FLAG, RAISE REQUEST IN
* SEE NOTE
SET STACKED STATUS FLAG
* SEE NOTE
SUPPRESS OUT ACTIVE ?
Y
N
ENABLE SUPPRESS OUT MONITORING BY RESPOND IMMEDIATE
RAISE REQUEST IN
LOOP
LOOP
STATUS TYPE ?
Y
N
(2ND MSB SET ?)
AVAILABLE TYPE ?
N
Y
(MSB RESET ?)
SELECTIVE RESET TYPE ?
N
Y
(2ND LSB SET ?)
AVAILABLE-STATE Ø
DEVICE NOT AVAILABLE-STATES Ø,A & B
DISCONNECT FLAG SET ?
N
Y
* SEE NOTE
SET DEVICE STATE TO "NOT READY"
INCREMENT POLLING ADDRESS
* SEE NOTE
(DEVICE STATE 2,3)
MICRO HAD CONTROL OF A SELECTIVE RESET WHEN A NEW COMMAND WAS RECEIVED.
3 = GOING NOT READY
2 = GOING READY
* SEE NOTE
RESET BIT 1 OF THE DEVICE STATE AND OR EXTRA STATUS IN TO DEVICE STATE
DEVICE STATE = ØØØ1B ?
N
Y
DEVICE STATE = Ø,4 OR 5
SET DEVICE STATE TO "NOT READY" IN REGISTER FILE
* SEE NOTE
*NOTE : RESPIM IS FREQUENTLY SAMPLED IN THIS ROUTINE
RESPIM ACTIVE ?
Y
N
CALL IMPRES
SEE FIGURE 14
INTDIS
SELRES

Figure 15. Initialization and Idle Loop Flow Diagram

Routine WATCON (Wait for Connect to Channel)

WATCON is entered from HSTART or DISCON (disconnect). WATCON monitors the status of the Multisourcerer's front panel DISCONNECT switch. This switch and the routine WATCON direct the Multisourcerer to make an orderly connection to or disconnection from the IBM channel.

WATCON implements the connect/disconnect sequences described by IBM and explained in pages 2-21 through 2-23 of the I/O Reference Manual.

WATCON waits for the DISCONNECT switch to be placed in the "connect" position. It monitors the status of this switch via an input (DISCNT/) to the Condition Code Multiplexer.

Once the DISCONNECT switch is placed in the "connect" position, WATCON tells the microprocessor to generate a time delay. The time delay request is initiated by writing a "timeout" code (F6H) to address 680FH in the Register File. The MSD (FH) signifies a GPCI request, while the six is a modifier which is interpreted by the microprocessor as a "timeout." The Bit-Slice Processor interrupts the microprocessor by calling TUPR0 (see subroutine description).

When the microprocessor is interrupted, it reads Register File address 680FH and determines it must begin a timeout. After generating the interrupt, WATCON energizes relays SL2 and SL3, located on the Drivers and Receivers card. At this point, WATCON enters a wait loop where it waits for an indication from the microprocessor that the timeout is completed. The purpose of this timeout is to allow the relays to settle (debounce) before proceeding. At this time, the Select Out signal is still coupled directly from the incoming Select Out pin to the outgoing Select Out pin via the normally closed contacts of SL1. However, because SL2 and SL3 are energized, the operational path is now functional. Thus, at this time, both paths operate in parallel.

Now, Select Out is propagated by raising PSOUT. Another timeout is initiated by writing F6H to 680FH, and relay SL1 is energized. This opens the Select Out bypass route. At this point, the GPCI is physically connected to the channel. The microprocessor is told of the connection by writing an F0H to address 680FH, followed by an interrupt. Finally, WATCON waits for the channel to go to a quiescent state before calling subroutine IMRES4. IMRES4 sets the programmable channel control signal monitor (RESPIM/) before exiting to STPOL.

Routine STPOL (Polling Loop)

STPOL is the Bit-Slice Processor's idle loop. There are several entry points into STPOL, though once entry is made, the only exit from the loop is by conditionally calling IMRES. IMRES is called when a test of the signal RESPIM/ passes. This test is performed every few lines of code throughout STPOL.

RESPIM/ is generated by hardware that monitors the following channel originated control signals:

- LOPT/ (Latched Operational Out)
- LSUT/ (Latched Suppress Out)
- LADT/ (Latched Address Out)
- LSHT/ (Latched SHOUT-Select Out ORed with Hold Out)

The Bit-Slice Processor can mask the RESPIM/ generator to respond to a single input only or a combination of inputs. Prior to entry into STPOL from WATCON, WATCON calls subroutine IMRES4 (part of IMRES) which programs the RESPIM/ generator to monitor SHOUT and Operational Out. Operational Out is always monitored and is non-maskable. RESPIM/ can be programmed to go low on the rising or falling edge of LSHT/(Latched SHOUT), the rising edge of LOPT/ (Latched Operational Out), or the falling edge of LSUT/ (Latched Suppress Out) or LADT/ (Latched Address Out). The last four of these signals are maskable. IMRES4 programs RESPIM/ to go low on the rising edge of SHOUT. Additional information on IMRES is available in the subroutine descriptions discussed later in this section.

Disconnect Monitoring

The first test performed by STPOL is a test of the DISCONNECT switch status. This switch has to be in the "connect" position to exit WATCON (see WATCON above). If you need to take the Multisourcerer off line, place the DISCONNECT switch in the "disconnect" position prior to removing power. The reason for this is that any devices which are currently active must have a status of Device End transmitted to the channel. Failure to adhere to this requirement may "hang" the channel.

When the switch is placed in the disconnect position, the Bit-Slice Processor informs the microprocessor of the condition by interrupting it. The microprocessor writes a status of Device End for each active device. Then, STPOL transmits the Device End status via the normal polling process. When all active channels have had a Device End status transmitted, the microprocessor informs the Bit-Slice Processor that all devices are logically disconnected. Then, STPOL exits by jumping to DISCON. DISCON enables the Select Out bypass path by de-energizing all relays, physically disconnecting the GPCI from the channel. Then, DISCON exits to WATCON where the Bit-Slice Processor waits for the DISCONNECT switch to be placed back into the connect position.

Command Chaining

If the DISCONNECT switch is in the connect position, STPOL tests RESPIM/ and conditionally calls IMRES as described above. Next, STPOL tests for command chaining. For information on command chaining refer to page 2-10 of the I/O Reference Manual. Command chaining is indicated during the completion of the present command. It informs the control unit of a pending new command immediately following the completion of the present command. When command chaining occurs, polling is bypassed since immediate reservicing of the present device is indicated.

Device State Groups

Refer to Figure 15 on page 8-11. The polling loop is bypassed if either the Pending Status flag or the Stacked Status flag is set. For the present, assume that neither flag is set. Now, STPOL retrieves the device state of the I/O device whose address is stored in the Polling Address register (Register A). Table 23 on page 8-7 is a list of the possible device states. Device states are analyzed and separated into four groups. These groups are:

- Device Not Available (Device Busy, Device Not Ready and Not Present)
- Status (Pending Status, Pending Status Request, Stacked Status and Stacked Status Request)
- Device Available (Available)
- Going Not Ready, Going Ready (Microprocessor had control during a Selective Reset)

Device Not Available Group

The Device Not Available group consists of Device Busy (1000B), Device Not Ready (1010B) and Not Present (1011B). Refer to the Device Not Available path in Figure 14 on page 8-6. If examination of the device state reveals a Device Not Available type, STPOL simply increments the polling address in preparation for testing the next device. Then STPOL tests RESPIM/ and loops back to the top of STPOL. In effect, STPOL skips any devices which fall into the Device Not Available group.

Status Group

The Status group consists of Pending Status (0100B), Stacked Status (0110B), Pending Status Request (0101) and Stacked Status Request (0111B). When the microprocessor completes a command (i.e., data transfer), it writes a status into the appropriate Register File address. Then it changes the device state from Busy to Pending Status. Assume the device, whose address is in the Polling Address register, has a Pending Status device state. The first test performed by the Status group path checks to see if the status is stacked or pending. Since we have a Pending Status, we enter the "Pending Status" branch. Here, the Pending Status flag (bit 0 of Register F) is set and the Request In control line to the channel is raised. This control line informs the channel that the GPCI has information to pass to the channel (status in this case). Once the Request In line is raised, STPOL checks for channel activity and returns to the top of STPOL. Now, notice that the Pending Status flag prevents access to the lower portion of STPOL. Thus, at this time, STPOL can only test for:

- DISCONNECT switch status
- channel activity
- command chaining
- Stacked Status
- Pending Status

The Stacked Status flag test will fail at this time since the Pending Status flag is set. Also, since command chaining is indicated only when both status flags (Pending and Stacked) are set, the Command Chaining flag test will also fail. Therefore, during a Pending Status condition, STPOL tests only for channel activity and a disconnect indication.

The channel responds to the raised Request In control signal by raising Select Out and Hold Out SHOUT goes high and (LSHT/ goes low). LSHT/ causes RESPIM/ to go low, resulting in a call of subroutine IMRES. IMRES attempts to send the device status to the channel. If the channel is currently not busy, it will accept the status. However, if it is busy, it tells the GPCI to stack the status. If this happens, IMRES clears the Pending Status flag and sets the Stacked Status flag.

If the channel accepts the status, IMRES clears the Pending Status flag unless the channel indicates command chaining. If command chaining is indicated, IMRES sets the Stacked Status flag and the Pending Status flag remains set.

Assume that the status is stacked (Stacked Status flag is set). Now, when control is returned to STPOL, processing continues down through STPOL until the Stacked Status flag is tested. This test passes and STPOL directs processing to the Stacked Status branch of the Status group of device states (See Figure 15 on page 8-11). Here, STPOL tests for channel activity and then tests SUT/. SUT/ (inverted Suppress Out) went low to tell the GPCI to stack its status, and remains low until the channel is able to accept the Stacked Status. Therefore, if Suppress Out is still low, you simply loop to the top of STPOL to repeat the cycle. Eventually, Suppress Out goes high, allowing STPOL to raise Request In. CUIS is initiated as described above when Request In goes high. Command chaining is always broken when status is stacked. Both flags (Stacked Status and Pending Status), are reset when the channel accepts the Stacked Status.

In STPOL, a Pending Status Request is handled exactly like a Pending Status, resulting in a transfer to CUIS. CUIS will either transmit a Device End if status is accepted or stack the status if it is unaccepted. In the latter case, the device state is changed to Stacked Status Request.

Device Available

Refer to Figure 15. The Device Available path in STPOL increments the Polling Address and returns to the top of STPOL. A device enters the Device Available state when it is available for service. Under this condition, the device requires no action on the part of the Bit-Slice Processor. Therefore, except during a disconnect, STPOL simply sets up the polling address to check the next device.

The Disconnect flag is tested within the Device Available path in STPOL. This flag is set near the beginning of STPOL if the DISCONNECT switch is in the disconnect position. When the switch is in this position, the microprocessor loads a Device End status into each busy device's status register, sets the device state to Pending Status, and monitors the device state.

During its polling cycles, STPOL detects the Pending Status, transmits the status to the channel, and resets the Pending Status flag as described above. At this time, the device's state is Device Available. During a Disconnect process, the test of the Disconnect Flag passes, resulting in a device state change from Device Available to Not Ready. When the microprocessor determines that all active devices are in a state of Not Ready, it activates the control signal CTL2. STPOL tests CTL2 immediately after testing the DISCONNECT switch. When both tests pass, it transfers control to DISCON which de-energizes all relays on the Channel Drivers and Receivers board, completing the interface disconnect process.

Once a device goes Not Ready, the microprocessor reactivates it by sending an Attention status to the channel.

Going Not Ready Going Ready Group

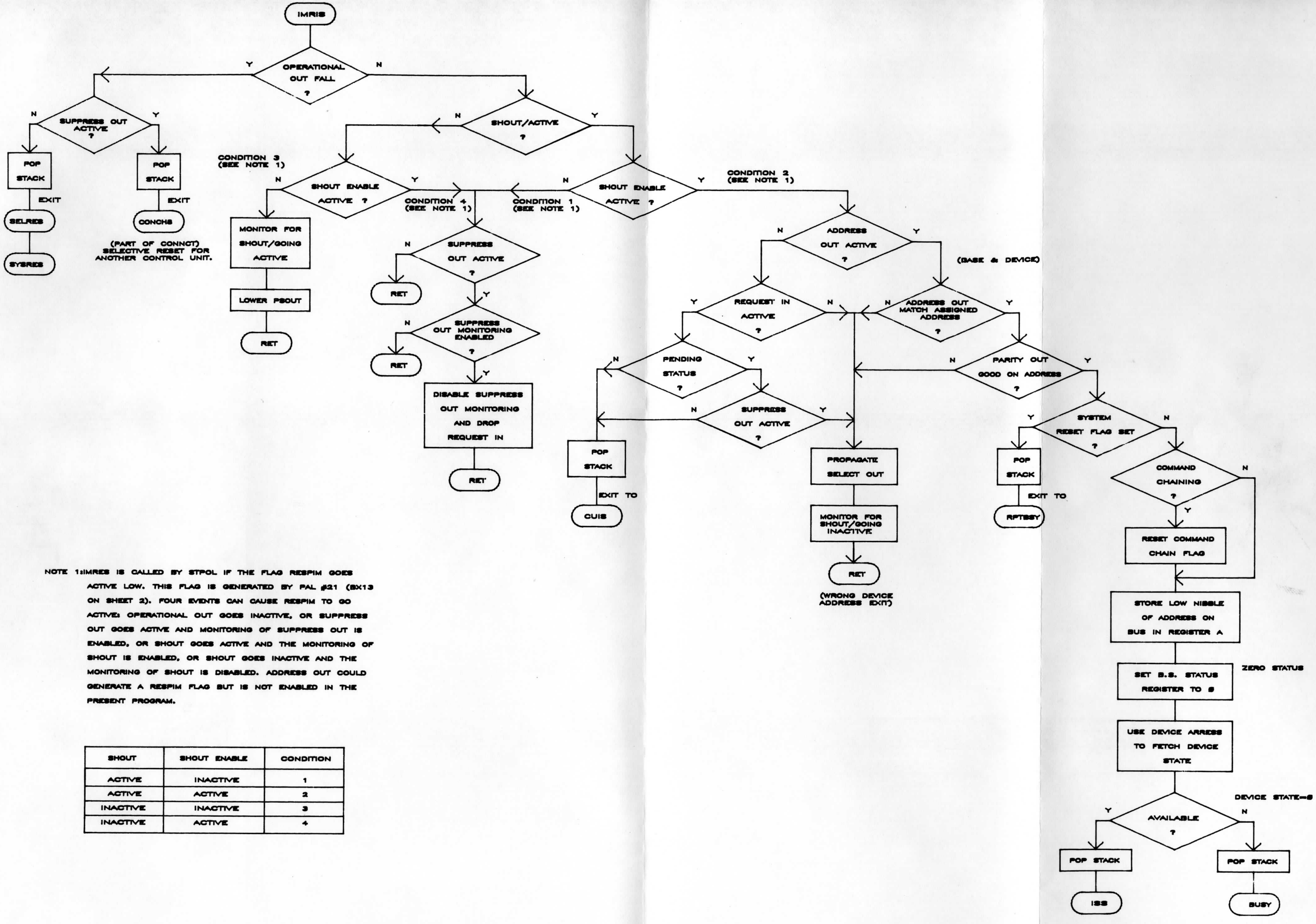
The last two device states, Going Not Ready and Going Ready, occur as a result of a Selective Reset. The channel (not the host) performs a Selective Reset when it does not receive expected results from a device within a predetermined period of time. A Selective Reset is indicated when the channel drops Operational Out (OPT/ rises) while Suppress Out is high (SUT/ is low). When OPT/ rises, IMRES is called as a result of the next test of RESPIM/. IMRES does additional testing to separate Selective Resets from System Resets. During a Selective Reset, IMRES transfers control to SELRES. SELRES sets the device state to Busy and tells the microprocessor of the reset.

After the microprocessor performs the Selective Reset, the device can go to a Ready (Device Available) or Not Ready (Device Not Available) state. The Ready/Not Ready status will normally be the same after the reset as it was before the reset. The one exception is if you changed the Ready/Not Ready state via the keyboard prior to the reset, but the microprocessor was unable to change the state prior to the reset. In this case, the microprocessor changes the state after the reset.

SELRES initially sets the device state to Busy. After completion of the reset, the microprocessor clears the device's status registers and sets the device state to Going Ready or Going Not Ready. Since there is no status response to a Selective Reset, the channel has no way of knowing when the reset is complete. Thus, it is possible for the channel to send a command to the device while it is still being reset. Should this happen (at this time, the device state is Busy), the GPCI would respond with a status of Busy and set the Extra Status register (Register 8) to Busy Reported.

The Going Ready Going Not Ready branch of STPOL has two purposes. It changes the Going Ready or Going Not Ready device state to Device Available or Device Not Available respectively, and it clears the Busy report to the channel if one was sent. If Busy was sent, a Device End status must be sent to clear the Busy condition.

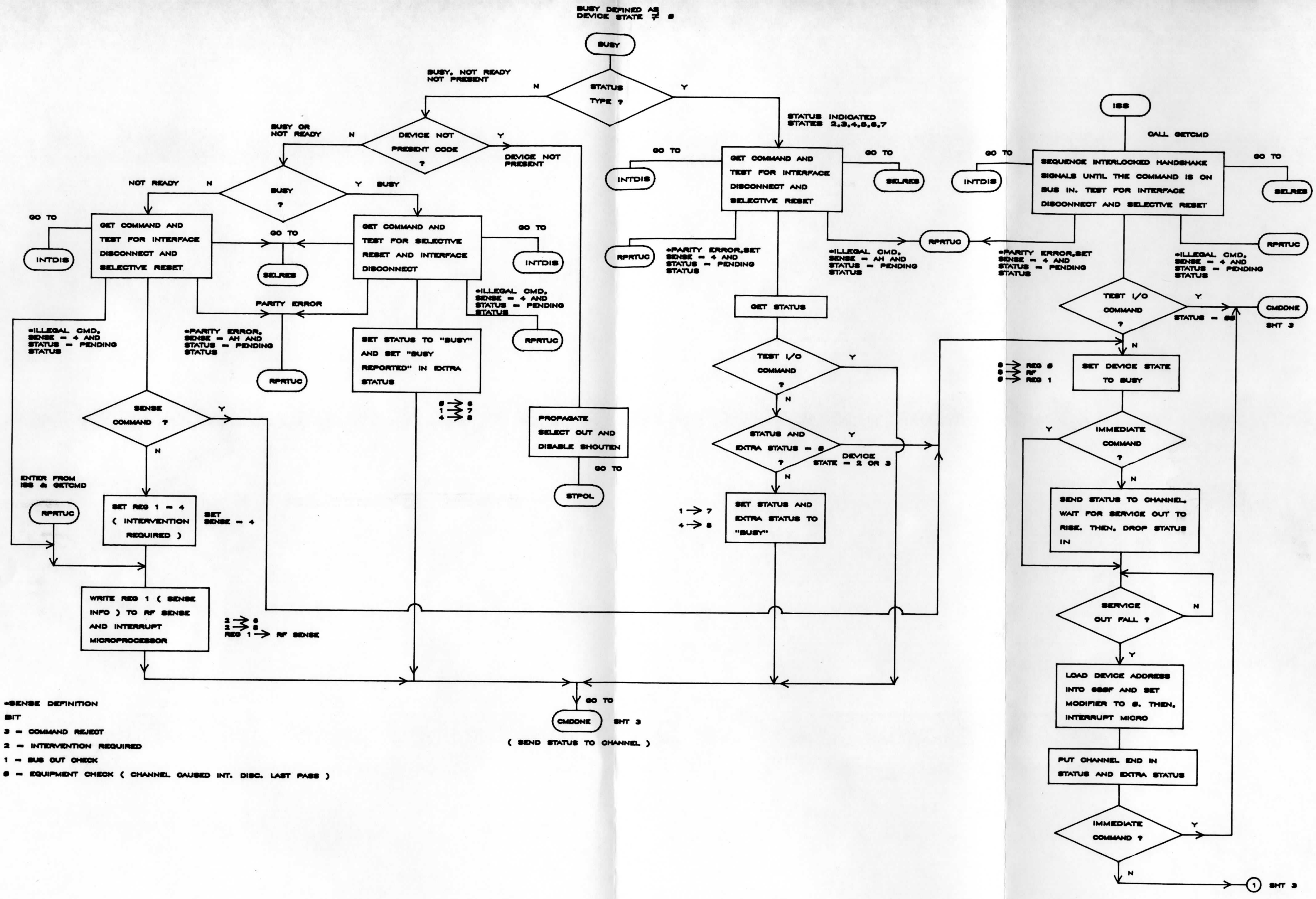
First, the device state codes for Going Ready (code 0010B) and Going Not Ready (code 0011B) are changed to device state codes 0000B and 0001B respectively, by resetting the second LSB. Then, Extra status is ORed with the device state code. At this time, Extra Status will be 0000B unless a status of Busy was sent to the channel. In that case Extra Status will be 0100B. By combining the Device State code and Extra Status, a Pending Status or Pending Status Request is formed only if a Busy Reported condition exists. If a Busy Reported condition doesn't exist, the 0001B (Going Not Ready) code is changed to 1010B, and the 0000B remains unchanged. The Pending Status and Pending Status Request (codes 0100B and 0101B respectively) cause STPOL to process these pending status codes as described above for the Status group of device states.



NOTE 1:IMRES IS CALLED BY STPOL IF THE FLAG RESPIM GOES ACTIVE LOW. THIS FLAG IS GENERATED BY PAL #21 (BX13 ON SHEET 2). FOUR EVENTS CAN CAUSE RESPIM TO GO ACTIVE: OPERATIONAL OUT GOES INACTIVE, OR SUPPRESS OUT GOES ACTIVE AND MONITORING OF SUPPRESS OUT IS ENABLED, OR SHOUT GOES ACTIVE AND THE MONITORING OF SHOUT IS ENABLED, OR SHOUT GOES INACTIVE AND THE MONITORING OF SHOUT IS DISABLED. ADDRESS OUT COULD GENERATE A RESPIM FLAG BUT IS NOT ENABLED IN THE PRESENT PROGRAM.

SHOUT	SHOUT ENABLE	CONDITION
ACTIVE	INACTIVE	1
ACTIVE	ACTIVE	2
INACTIVE	INACTIVE	3
INACTIVE	ACTIVE	4

Figure 16. Subroutine IMRES Flow Diagram (Sheet 1 of 3)



*SENSE DEFINITION

BIT

3 = COMMAND REJECT

2 = INTERVENTION REQUIRED

1 = BUS OUT CHECK

Ø = EQUIPMENT CHECK (CHANNEL CAUSED INT. DISC. LAST PASS)

Figure 16. Subroutine IMRES Flow Diagram (Sheet 2 of 3)

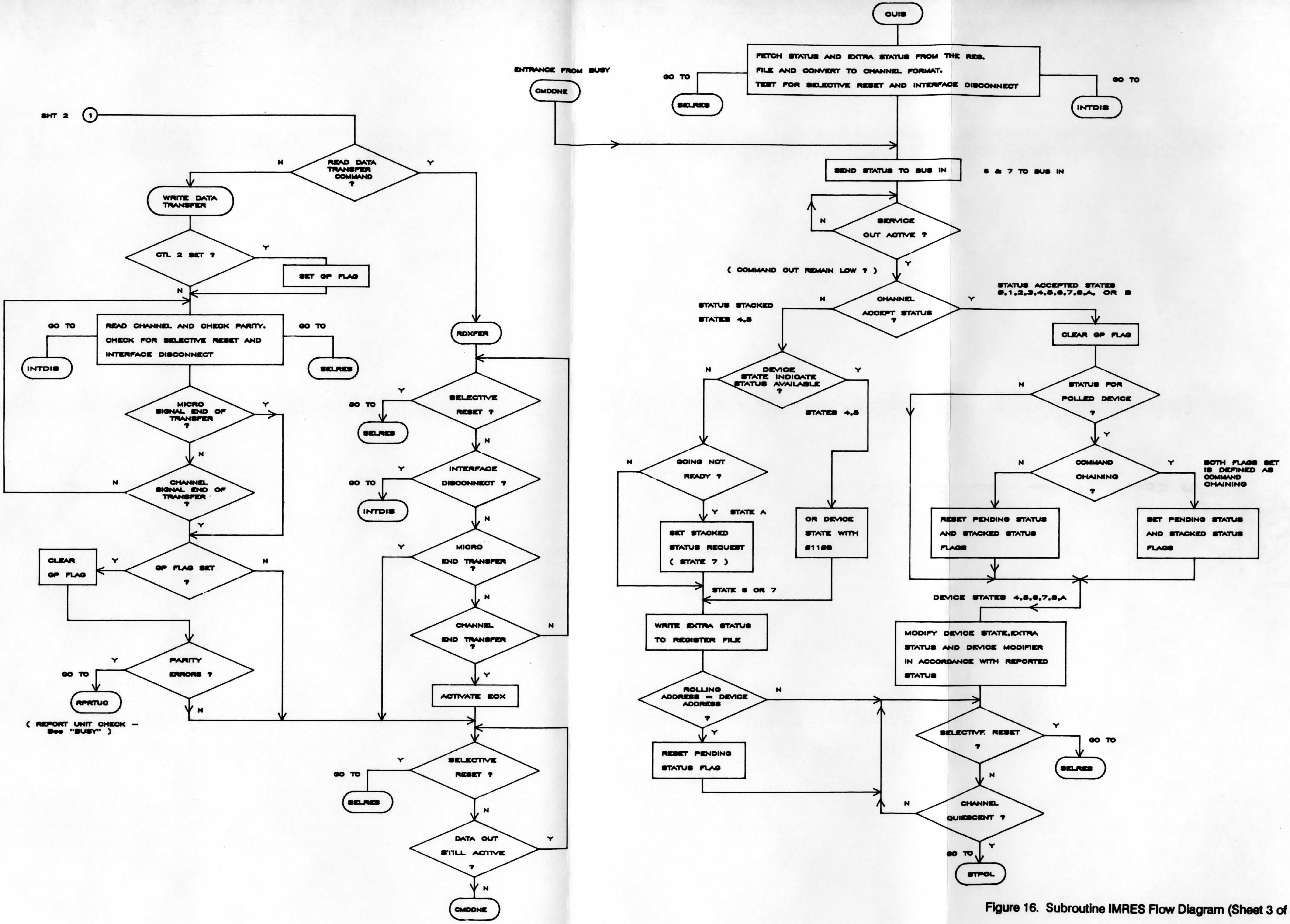

SHT 2
1
READ DATA TRANSFER COMMAND ?
N
Y
WRITE DATA TRANSFER
CTL 2 SET ?
Y
SET GP FLAG
N
GO TO
READ CHANNEL AND CHECK PARITY. CHECK FOR SELECTIVE RESET AND INTERFACE DISCONNECT
GO TO
INTDIS
SELRES
MICRO SIGNAL END OF TRANSFER ?
Y
N
CHANNEL SIGNAL END OF TRANSFER ?
N
Y
CLEAR GP FLAG
Y
GP FLAG SET ?
N
PARITY ERRORS ?
Y
GO TO
RPRTUC
(REPORT UNIT CHECK — See "BUSY")
N
RDXFER
SELECTIVE RESET ?
Y
GO TO
SELRES
N
INTERFACE DISCONNECT ?
Y
GO TO
INTDIS
N
MICRO END TRANSFER ?
Y
N
CHANNEL END TRANSFER ?
N
Y
ACTIVATE EOX
SELECTIVE RESET ?
Y
GO TO
SELRES
N
DATA OUT STILL ACTIVE ?
Y
N
CMDDONE
ENTRANCE FROM BUSY
CMDDONE
CUIS
FETCH STATUS AND EXTRA STATUS FROM THE REG. FILE AND CONVERT TO CHANNEL FORMAT. TEST FOR SELECTIVE RESET AND INTERFACE DISCONNECT
GO TO
SELRES
GO TO
INTDIS
SEND STATUS TO BUS IN
6 & 7 TO BUS IN
N
SERVICE OUT ACTIVE ?
Y
(COMMAND OUT REMAIN LOW ?)
N
CHANNEL ACCEPT STATUS ?
Y
STATUS STACKED STATES 4,5
STATUS ACCEPTED STATES 0,1,2,3,4,5,6,7,8,A, OR B
CLEAR GP FLAG
N
DEVICE STATE INDICATE STATUS AVAILABLE ?
Y
STATES 4,5
N
STATUS FOR POLLED DEVICE ?
Y
N
GOING NOT READY ?
Y STATE A
SET STACKED STATUS REQUEST (STATE 7)
OR DEVICE STATE WITH 81199
N
COMMAND CHAINING ?
Y
BOTH FLAGS SET IS DEFINED AS COMMAND CHAINING
RESET PENDING STATUS AND STACKED STATUS FLAGS
SET PENDING STATUS AND STACKED STATUS FLAGS
STATE 6 OR 7
DEVICE STATES 4,5,6,7,8,A
WRITE EXTRA STATUS TO REGISTER FILE
MODIFY DEVICE STATE,EXTRA STATUS AND DEVICE MODIFIER IN ACCORDANCE WITH REPORTED STATUS
ROLLING ADDRESS = DEVICE ADDRESS ?
N
Y
RESET PENDING STATUS FLAG
SELECTIVF. RESET ?
Y
GO TO
SELRES
N
N
CHANNEL QUIESCENT ?
Y
GO TO
STPOL

Figure 16. Subroutine IMRES Flow Diagram (Sheet 3 of 3)

Subroutine IMRES

IMRES directs all processing external to STPOL. IMRES is called by STPOL when RESPIM/ is low. For additional information on RESPIM/, refer to the Routine STPOL description on page 8-14.

IC BX13 generates RESPIM/ under one or more of these conditions:

- LOPT/ goes high, or
- LSHT/ goes low and LSHT/ monitoring is enabled, or
- LSHT/ goes high and LSHT/ monitoring is disabled, or
- LADT/ goes low and LADT/ monitoring is enabled, or
- SUT/ is low and SUT/ monitoring is enabled

Since any of the above conditions can cause RESPIM/ to go low, much of IMRES consists of tests that determine which of the above conditions actually caused RESPIM/ to go low.

Figure 16 on pages 8-19 through 8-24, shows the program flow of IMRES. First, IMRES tests Operational Out (LOPT/). If LOPT/ is high, it indicates either a System Reset for all control units and devices or a Selective Reset for another control unit. A System Reset is indicated when Operational Out falls (LOPT/ goes high) and Suppress Out is down (SUT/ is high), and the GPCI is on line (physically connected). A Selective Reset is indicated when LOPT/ rises while SUT/ is low (Suppress Out is high), and is valid only if Operational In is high. Thus, a System Reset can occur at any time, while a Selective Reset only occurs when a device is logically connected.

Normally, a device is never logically connected when IMRES is called, since IMRES transfers requests for logical connections to CUIS or ISS. Therefore, IMRES assumes that a Selective Reset must be intended for a device belonging to another control unit (i.e., another Multisourcerer). If Suppress Out is inactive (System Reset), IMRES exits to SELRES. SELRES lowers all tag lines and exits to SYSRES. If Suppress Out is active, IMRES exits to CONCH8 (Selective Reset for a device belonging to another control unit). CONCH8 (part of WATCON) waits for Operational Out to go high again, then exits to STPOL.

If LOPT/ is low (no resets), LSHT/ is tested. SHT/ is generated on the Channel Drivers and Receivers board and is formed by ANDing Select Out and Hold Out. LSHT/ is formed by PAL 2, located at BE1. BE1 latches SHT/ to form LSHT/. The RESPIM/ generator (IC BX13) is programmed by writing the contents of Register E to it (BX13). Register E contains the mask for BX13. BX13 can generate RESPIM/ either on the rising or falling edge of LSHT/. The LSB of Register E (SHEN) determines which edge of SHOUT/ will cause RESPIM/ to go low. If SHEN is high, the falling edge of LSHT/ drives RESPIM/ low. Conversely, if SHEN is low, RESPIM/ activates on the rising edge of LSHT/. Therefore, both conditions of SHEN must be tested, as well as both states of LSHT/.

Table 24 below shows all possible combinations of LSHT/ and SHEN, and the processing performed in each of the four paths.

Condition	SHOUT/*	SHEN	Processing
1	low	low	Test Suppress Out
2	low	high	Test for valid CUIS or ISS
3	high	low	Enable SHEN, drop propagation of Select Out and Return
4	high	high	Test Suppress Out

*LSHT/ is active low

Table 24. RESPIM/ Programming

During conditions 1 and 4, RESPIM/ is not caused by SHOUT/. Thus, SUT/ is tested to determine if it caused RESPIM/ to go low. If SUT/ is high, or if it is low but SUT/ monitoring is not enabled, RESPIM/ is assumed to be caused by activity between the channel and another control unit on the same channel. Conditions 2 and 3 result when LSHT/ goes low or high respectively. Both states of LSHT/ need to be detected. The GPCI is designed to operate in conjunction with other control units on the same channel. Since all bus, tag, and control signals (except Select Out) are parallel connected, the GPCI must be able to identify interface sequences intended for other control units as well as those sequences intended for itself. The following possibilities exist:

- ISS is intended for this GPCI
- CUIS is initiated by this GPCI
- ISS is intended for another control unit
- CUIS is initiated by another control unit

If either a CUIS or an ISS is intended for another control unit, IMRES must pass Select Out to the next downstream control unit, set up RESPIM/ to monitor for LSHT/ going high (condition 3), and return to STPOL. When the process (CUIS or ISS) between the channel and another control unit ends, RESPIM/ goes low (LSHT/ goes high), calling IMRES again. This time, IMRES stops the propagation of Select Out and re-enables monitoring for LSHT/ going low (condition 2). Thus, IMRES assumes that LSHT/ is destined for this GPCI unless additional processing proves otherwise. When control is relinquished to another control unit, the GPCI monitors the channel control lines to determine when the other control unit is finished. When the other control unit finishes, this GPCI assumes the next falling edge of LSHT/ will be for it.

Refer to Figure 16 (pages 8-19 through 8-24). When the channel attempts an ISS, IMRES verifies that the base and device address is correct and free of parity errors. Then the System Reset flag is tested. This flag is set at the beginning of a System Reset and reset at its completion. If IMRES is called during a System Reset, this flag will be set. The reason for this procedure is that no report of reset completion is sent to the channel after a reset is completed. Therefore, the channel has no way of knowing when it can resume communications following a reset.

If the GPCI is still busy with the reset when the channel attempts to initiate an ISS, IMRES detects the set System Reset flag and transfers processing to RPTBSY (Report Busy). RPTBSY sends a status of Control Unit Busy to the channel, ending any further channel initiated activity until the GPCI sends a status of "Control Unit End." RPTBSY also sets a flag indicating that Control Unit Busy was sent. This flag is tested upon completion of the reset. If the flag is set, the GPCI sends the Control Unit End status. Refer to the SYSRES description on page 8-53 for more information.

Assuming the System Reset flag is not set, the Command Chaining flag is tested. If the Command Chaining flag was set, it is now reset. The Command Chaining flag will be set again prior to completion of the command if command chaining continues (see routine CMDDNE below).

After testing and clearing the Command Chaining flag, IMRES fetches the device state from the Register File to determine if the device is busy (any device state other than Device Available). If the device is busy, processing transfers to BUSY; if the device state is not busy, processing transfers to ISS. Refer to the BUSY or ISS descriptions if necessary.

IMRES determines that STPOL initiated a CUIS process by testing Address Out (ADT/) and Request In. If ADT/ is high, either the GPCI or another control unit raised its Request In, thereby initiating a CUIS process. Therefore, upon finding ADT/ high, IMRES tests Request In. If Request In is high, the requested CUIS sequence was initiated by this GPCI. If Request In is low, another control unit initiated the sequence. When another control unit initiates the sequence, Select Out is propagated and RESPIM/ is reconfigured to monitor for LSHT/ going high (condition 3). IMRES transfers control to CUIS when Request In is high, providing the channel is not indicating a busy condition.

Routine CUIS (Control Unit Initiated Sequence)

When invoking a call routine, the Bit-Slice Processor saves its return address by pushing it onto an internal stack. The normal exit from a call routine is a return instruction. A return instruction automatically "pops" the stored return address from the stack, and loads it into the program counter. If a call routine is exited by a jump instruction, the return address must be removed from the stack manually. Otherwise, the stack memory will soon overflow.

CUIS is entered from IMRES using a conditional jump instruction. This occurs as a result of the channel's response to the raised Request In control line. All exits from CUIS consist of jumps (with a stack pop) to the top of STPOL, thereby reentering the polling cycle at a known point. Therefore, IMRES (a call routine) requires a stack pop to clear the Bit-Slice Processor's stack pointer.

Although CUIS does not actually send status to the channel, it allows the GPCI to send it via CMDDNE (Command Done). CUIS merely retrieves the device status from the proper Status and Extra Status locations in the Register File, converts the status to channel format, and stores the status in the ALU's status working registers (Registers 6 and 7). To access the proper Register File addresses, CUIS fetches the polling address (refer to STPOL) and uses it as the Register File address. In addition to fetching and formatting the status, CUIS advances into the Channel I/O sequence by sending the Base and Device address to the channel (refer to page C-11 of the I/O Reference Manual).

While waiting for the channel to accept the address, CUIS monitors for Interface Disconnects, and System or Selective Resets. If a reset is indicated, CUIS jumps to SELRES which handles System and Selective Resets. The channel can signal the control unit (GPCI) to terminate ongoing I/O operations by performing an Interface Disconnect. Refer to the I/O Reference Manual for information on how the channel indicates an Interface Disconnect. If CUIS detects an Interface Disconnect, it transfers control to INTDIS.

Assuming no requests for resets or disconnects exist, CUIS jumps to CMDDNE. CMDDNE sends the status, located in Registers 6 and 7, to the channel. For more information on CMDDNE, refer to Routine CMDDNE on page 8-35.

Routine ISS (Initial Selection Sequence)

This routine is accessed by IMRES when LADT/ and LSHT/ go low. Also, a parity free address on Bus Out must be a valid address for this GPCI. Before IMRES jumps to ISS, it performs a stack pop. The purpose of this stack pop is the same as described in the first and second paragraphs of the CUIS description on the adjacent page.

The channel uses the ISS procedure to issue a command to the GPCI. ISS begins by calling GETCMD (Get Command), which sequences interlocked control signals until the command is on Bus Out. In a manner similar to CUIS, GETCMD also checks for Interface Disconnects and System and Selective Resets. GETCMD sets up GPCI hardware that maps the 8-bit command into a 4-bit code, and then stores the code in Register 3. GETCMD accomplishes the mapping process by using the channel command and the Device Type (stored in the Register File) as address inputs to a mapping PROM. Table 25 below shows all the possible codes.

4-Bit Code	HEX Value	Function
0000	0	Immediate
00001	1	Not Assigned
0010	2	Not Assigned
0011	3	Not Assigned
0100	4	Test I/O
0101	5	Not Assigned
0110	6	Not Assigned
0111	7	Illegal
1000	8	Data Transfer Read Micro (Sense)
1001	9	Data Transfer Read Multibus
1010	A	Data Transfer Read DMA (Memory)
1011	B	Data Transfer Read DMA (I/O)
1100	C	Data Transfer Write Micro
1101	D	Data Transfer Write Multibus
1110	E	Data Transfer Write DMA (Memory)
1111	F	Data Transfer Write DMA (I/O)

Table 25. Command Look-Up Table

Table 26 below lists the Device Types.

4-Bit Code	Device
0000	Bisync
0001	Ingestor
0010	ProNET
0011-1111	not used

Table 26. Device Types

GETCMD exits to RPRTUC (Report Unit Check) if the command contains parity errors or if it is an illegal command. If the command is parity error free and legal, GETCMD returns to the calling routine (ISS). For additional information, refer to the GETCMD subroutine information on page 8-41.

The remainder of ISS consists of decoding the command code and directing processing to the appropriate command handler.

The command is tested to determine if it is a Test I/O command. Test I/O commands send device status to the channel. Therefore, if the command is a Test I/O type, ISS simply jumps to CMDDNE. CMDDNE transmits the status, located in Registers 6 and 7, to the channel. For all other commands, the Device State register (Register 0) is set to Busy (8H). The appropriate device state nibble in the Register File is also set to 8H.

Next, the command is tested to determine if it is an Immediate command type. An Immediate command is any command which causes the I/O device to signal Channel End as its initial status during an Initial Selection Sequence. However, Test I/O is not an Immediate command. An Immediate command must meet the following requirements:

- its execution requires no more information than that contained in the command itself; that is, no data bytes are transferred
- the Channel End status is presented as initial status; Device End status may accompany Channel End

For all other command types, an initial status of 00 is loaded into Registers 6 and 7. If the command is Immediate, the 00 status transmission process is bypassed. Next, the Bit-Slice Processor interrupts the microprocessor to inform it that a command is pending. Next, a Status of Channel End is written to the status and Extra Status registers (Registers 6 and 8 respectively). The command is again tested to determine if it is an Immediate command. Immediate commands are completed by jumping to CMDDNE which writes the contents of Register 6 (Channel End) to the channel. The Channel End status is sent to the channel upon completion of the command (data transfers).

If the command is not an Immediate command, by default, it is a data transfer command type. The data transfer commands are sorted into read and write transfers. All read transfers are handled by RDXFER (Read Transfer); all write data transfers are handled by Write Data Transfer. The actual data transfer is accomplished by hardware. The read and write data transfer routines merely monitor the transfer processes, regardless of data sources and sinks.

There are several data source and sink combinations, and the hardware must be configured for each combination. The hardware configuration on the Bit-Slice Processor end is handled directly by hardware that decodes the command. The microprocessor end is handled primarily by the microprocessor. In fact, the Bit-Slice Processor interrupts the microprocessor to allow the microprocessor to decode the command and configure its share of the hardware so the transfer can proceed.

RDXFER consists of a loop which tests for Selective Resets, Interface Disconnects, and the end of the transfer. Both the channel and the device can end data transfers. Therefore, RDXFER must test for both causes of transfer terminations. If neither the channel nor the device indicates an end of transmission, the loop is repeated. If the channel terminates the transfer (Command Out goes high - COT/ goes low), RDXFER activates the EOX (End of Transfer) control signal. EOX is one of the controlling signals used to configure hardware during a data transfer. The device indicates the end of transfer via the microprocessor. That is, when the device has transferred the last byte of data to the channel, the microprocessor drops RFINTA, which is monitored by RDXFER. Regardless of who terminates the transfer, ending status must be presented to the channel. Therefore, RDXFER exits to CMDDNE.

Write Data Transfer is similar to RDXFER. Like RDXFER, it consists of a loop that tests for Interface Disconnects, Selective Resets, and an end of data transfer. Transfers can be terminated by the channel or the device. Channel termination is indicated when the channel raises Command Out (COT/ falls) in response to the GPCI's raised Data In line. The microprocessor indicates the end of the transfer by dropping RFINTA. Write Data Transfer checks each data byte for parity. Parity error reporting is an option the operator may select using the Multisourcerer's keypad. Upon entry to Write Data Transfer, the microprocessor control signal, CTL2, is tested. If CTL2 is high (report parity errors), the GP (General Purpose) flag (Register F, bit 3) is set and flags error reporting after the transfer.

During the transfer, if one or more bytes have parity errors, Register 1 is set to a value of 0100B (four). Upon completion of the transfer, the GP flag is tested. If it is set, Register 1 is tested for a value of four. If it contains a four, Write Data Transfer jumps to RPRTUC (Report Unit Check). RPRTUC stores the 0100 code in the Register File as a Sense byte that will be requested by the channel during a subsequent Sense command (Immediate command). The channel will issue the Sense command because RPRTUC sends a status of Unit Check to the channel instead of the expected Channel End.

If there are no parity errors, or if parity error reporting is not selected, Write Data Transfer jumps to CMDDNE. CMDDNE outputs a Channel End status and completes the Ending Sequence. For more information on CMDDNE, refer to the CMDDNE routine description which begins on page 8-35. For more information on the Ending Sequence or data transfers, refer to the I/O Reference Manual.

Routine BUSY

IMRES transfers control to BUSY instead of ISS when all conditions for a transfer to ISS are met, except when the device has a device state other than "Device Available" (code 0000B). In other words, any legal command (refer to Table 5 on page 7-9) directed to any device which has a device state other than Device Available, will result in a program transfer to BUSY. BUSY is a failure recovery routine. That is, during normal operations, conditions which invoke BUSY do not normally occur.

Refer to the device code descriptions on pages 8-8 and 8-9 if necessary.

There are 10 valid command codes (see Table 5) and nine device states, excluding Device Available. Thus, approximately 90 command/device state combinations are possible. The combinations are further expanded by the possibility of a System Reset, Selective Reset, or Interface Disconnect occurring while a device is busy. These sequences are not only unpredictable in their occurrence, but their durations are also variable. Each combination possibility must result in a suitable action if satisfactory operation is to be attained. Most combinations require a response to the channel (status). Fortunately, only a few types of responses are needed to handle all combination possibilities. Therefore, BUSY must sort the combinations into groups which require the same or similar responses.

BUSY performs much of the sorting process using the device state codes alone. Refer to Figure 16 on pages 8-19 through 8-23. Analysis of the device state code results in four major paths. Within some of these paths, additional decoding is performed by analyzing the command codes and/or testing for resets and Interface Disconnects.

BUSY begins by testing the MSB of the device state (refer to Figure 16 and Table 7, the Device State Table). If the MSB is zero, device codes 0010B through 0111B are processed. Remember, device code 0000B will never be processed by BUSY. Except for device states 0010B and 00011B, this group of device codes is status types. At this time, the device may not be busy, but the channel requires a status report. Normally, the channel will not issue a new command to a device which still needs to send a Device End. However, if the channel is reset after issuing a command and before the Device End is received, a Pending Status or Pending Status Request could occur (codes 0100B and 0101B respectively).

The channel can perform a Selective Reset on a device and issue a new command after the microprocessor changes the state from BUSY to Going Ready or Going Not Ready. If this happens, the device state could be 0010B, 0011B, 0100, or 0101B, depending on when the command occurred and whether the device state was Going Ready or Going Not Ready.

If the device state's MSB is zero, BUSY fetches the new command and tests for Selective and System Resets, and Interface Disconnects. Then, it retrieves the device's status from the Register File and stores it in the ALU's working status registers (Registers 6 and 7). Next, it tests the command to see if it is a Test I/O. The purpose of a Test I/O command is to retrieve device status. Thus, if the command is a Test I/O command, BUSY simply exits to CMDDNE. CMDDNE sends the status, located in the status working registers, to the channel.

If the command is not a Test I/O command, the status and extra status are tested for "zero." A device which received a Selective Reset will have a status of zero if it didn't receive the new command while the device state is still BUSY. In other words, if the device state is 0010B or 0011B, and a command wasn't received during the reset, the test on the status and extra status will pass. When this happens, BUSY transfers processing to ISS, as the device is able to act on the new command.

If the MSB of the device state is a "one," the device state will be Busy (code 1000B), Device Not Ready (code 1010B), or Not Present (code 1011B). These states are separated into three paths by two tests.

If the state is Device Not Present, BUSY propagates Select Out and disables LSHT/ monitoring. BUSY interprets this situation as a command for another control unit. Two Multisourcerers can be connected to the same channel and have the same base address, and still function normally, providing the device addresses of both units are mutually exclusive.

If the state is busy, BUSY retrieves the command, testing for resets and Interface Disconnects. Then it sets the status to Busy, sets the Busy Reported flag, and exits to CMDDNE. CMDDNE transmits the Busy status to the channel. The Busy Reported flag is converted into a Device End status after the in-progress command is completed. After the channel receives the Device End status, it reissues the command that forced the Busy status transmission.

If the device state is Device Not Ready, BUSY fetches the command, as described above. Then it tests for the Sense command, which is the only command that can be processed by a device in a Device Not Ready state. This command retrieves data from the device's sense indicators. This sense data is detailed enough for the channel to determine the actual state of the device, and any unusual conditions associated with the execution of the I/O operation during which the error was detected.

If any command other than the Sense command is received for a device in the Device Not Ready state, BUSY writes Sense information (Intervention Required) to the Register File and interrupts the microprocessor. Prior to the interrupt, BUSY writes a status of Unit Check to the Register File and to the ALU's working status registers. The interrupt informs the microprocessor of the forthcoming Sense command, giving it time to collect the Sense data. CMDDNE sends the Unit Check status to the channel. When the channel receives a Unit Check status, it usually sends a Sense command to determine the problem.

Routine CMDDNE (Command Done)

All IBM commands require an initial status and an ending status. Also, the channel normally requires a Channel End status and a Device End status for each command. Depending on the task (command) to be performed and the type of control unit used, Device End and Channel End may be simultaneously sent as the ending status. In the Multisourcerer, the Bit-Slice Processor is responsible for the Channel End. The microprocessor is responsible for informing the Bit-Slice Processor when the Device End can be sent. The result of this division of responsibility is three status byte transmission, an initial status of "00" (acknowledges the command), a Channel End status (the channel is no longer required), and a Device End (the device has completed the command). The Bit-Slice Processor sends all status bytes to the channel.

The initial status is sent by ISS during the Initial Selection Sequence. This status can never be refused (stacked) by the channel. CMDDNE sends the Channel End and Device End status to the channel. This status is stackable meaning the channel can refuse it. The source for this status is Registers 6 and 7. Thus, these registers must be loaded with the proper status before jumping to CMDDNE.

Besides transmitting status, CMDDNE also updates the device state and sets or resets the Pending Status, Stacked Status, and Command Chaining flags. CMDDNE begins by sending the status, stored in Registers 6 and 7, to the channel. Then it waits for SET/ (Service Out) to fall (the channel's acknowledgement of receipt of the status). After SET/ falls, CMDDNE checks to see if the channel raised Command Out, thereby informing the GPCI to stack the status.

CMDDNE's two major processing paths are Status Accepted and Status Stacked. Each path modifies the device states, status flags, and device status in accordance with the present device state, and indicates whether or not the status was accepted. These device states are discussed on pages 8-37 and 8-38. The Stacked Status path begins by testing the device state. If a status type device state is found (device state bit 2 is set), CMDDNE ORs the device state with 0110B. Thus, a device state of Pending Status or Pending Status Request is changed to Stacked Status or Stacked Status Request respectively. STPOL retransmits the status as soon as the channel indicates not busy.

The device states that are not status types are 0000B, 0010B, 0011B, 1000B, 1010B and 1011B. Except for 1011B, these states occur only during presentation of initial status to the channel. Since initial status is nonstackable, the status of devices in these device states is processed by the Status Accepted branch of CMDDNE. Device state 1011B is aborted by BUSY and thus cannot exist at the entrance to CMDDNE. Therefore, in Figure 16, the non-status branch of the Stacked Status leg is not used. It does provide a means of recovery if one of the non-status device states becomes stacked. Finally, the Stacked Status branch writes the Extra Status to the Register File, resets the Pending Status flag if this is the polled device, and joins the output of the Status Accepted branch.

The Status Accepted branch of CMDDNE compares the polling address with the device address to determine if a command chaining test should be performed. A Device End status was sent to the channel if the polling address equals the device address. Command chaining is indicated by the channel during the presentation of a Device End status. This status transmission results from the Pending Status device state. The channel indicates command chaining by raising Suppress Out at the same time it acknowledges the status byte. The status byte is acknowledged by raising Service Out. Also, for command chaining to be valid, the status must indicate Device End and no BUSY Reported. If command chaining is indicated, the Pending Status and Stacked Status flags are set. Otherwise, both flags are reset.

The last block of the Status Accepted branch of CMDDNE modifies the device state and Extra Status register, and conditionally clears the Device Modifier register. Table 27 shows the results of this block under various device state inputs.

Device State at Entry	Device State at Exit	Comments
0	0	Clears Device Modifier & Extra Status
2*	0	Clears Device Modifier & Extra Status
3*	A	Clears Device Modifier & Extra Status
4	0	Clears Device Modifier & Extra Status
5	A	Clears Device Modifier & Extra Status
6	0	Clears Device Modifier & Extra Status
7	A	Clears Device Modifier & Extra Status
8	8	Clears Extra Status
A	0	Clears Device Modifier & Extra Status
B**	0	Clears Device Modifier & Extra Status Device States 0010B

* Test I/O instruction only

**All commands except Sense

Table 27. Modification of Device State and Status by CMDDNE

Now, the Status Accepted branch combines with the Status Stacked branch. The remainder of CMDDNE consists of testing for Selective and System Resets while waiting for Service Out to fall. When Service Out falls SET/rises, and CMDDNE exits to the top of STPOL.

Device State Summary

Device State 0000B (Device Available)

All commands sent to devices in device state 0000B are processed by ISS. For all commands except Test I/O, ISS sets the device state to 1000B (Device Busy). A Test I/O command sent to a device in a Device Available state is the only way a device state of 0000 will exist upon entry to CMDDNE. For these conditions, the status which CMDDNE sends will be initial status and cannot be stacked by the channel.

Device States 0011B and 0011B

Device States 0010B and 0011B are Ready and Going Not Ready respectively. These codes exist only in the Register File from the time a Selective Reset ends until the next time STPOL polls the device. If a new command is received for a device in either of these states during this time, the processing is turned over to BUSY. Since the MSB of the device state is "0", the status type path in BUSY is used. If the new command is a Test I/O command, it transfers control to CMDDNE. CMDDNE will send any status contained in the Status and Extra Status registers, located in the Register File. For all other commands, the Status and Extra Status are tested for zeros.

If the status bytes are zero, processing is transferred to ISS. ISS changes the 0010B or 0011B state code to Device Busy (1000B). Thus, CMDDNE processes a Test I/O command sent to a device in one of these states (0010B or 0011B). The status which CMDDNE sends will be initial status and cannot be stacked by the channel.

Device State 0100B (Pending Status)

The microprocessor sets this state to inform the Bit-Slice Processor that the device has completed the command. Except for Test I/O, all commands result in a Pending Status.

Device State 0101B (Pending Status Request)

STPOL generates a Pending Status Request when a device receives a command while a Selective Reset is in progress, and when that device was in a Going Ready state prior to the reset. If the device was in a Going Not Ready state prior to the reset, STPOL sets the device state to Stacked Status Request.

Device State 0110B (Stacked Status)

If CMDDNE attempts to send status to the channel and the channel is currently busy, CMDDNE changes the state to Stacked Status.

Device State 0111B (Stacked Status Request)

If the GPCI attempts to send status to a busy channel as a result of a Pending Status Request, CMDDNE changes the status to Stacked Status Request.

Device State 1000B (Device Busy)

All commands, except Test I/O, sent to a device in a ready state, cause a device state of Device Busy. Device Busy exists from the time that ISS sets the Device Busy state until the microprocessor indicates that the device has finished the command. If, for some reason, a device in a Device Busy state receives a new command, that command is processed by BUSY. BUSY sets the status register to Busy and sets the Busy Reported flag. Then BUSY exits to CMDDNE. CMDDNE sends an initial status which cannot be stacked by the channel.

Device State 1010B (Device Not Ready)

BUSY processes commands for devices in this state. If the command is Sense, it is turned over to ISS for processing. ISS changes the state to Device Busy. For all other commands, BUSY sets the status to Unit Check and exits to CMDDNE. CMDDNE sends an initial status which cannot be stacked by the channel.

Device State 1011B (Device Not Present)

This device state is aborted by BUSY.

Subroutine Descriptions

There are 21 subroutines in the Bit-Slice Microcode. IMRES is covered in detail in the previous section and is not addressed again here. Subroutine documentation consists of set-up requirements, a brief process description, outputs, called subroutines, and Bit-Slice ALU register modifications. The following is a complete list of the subroutines used in the GPCI.

- CHKPAR - Check Parity
- FETSTA - Fetch Status
- GETCMD - Get Command
- IMRES - Immediate Response
- IMRES4 - Immediate Response (section 4)
- MADBIN - Move Address to Bus In
- MCLTIN - Move Control to Tag In
- MSTBIN - Move Status to Bus In
- RDRFDS - Read Register File Device State
- RDRFDT - Read Register File Device Type
- RDRFEX - Read Register File Extra Status
- RDRFPS - Read Register File Processor Status
- TCANSD - Test Command and Service Down
- TCORSU - Test Command Out or Service Out Up
- TELLUP - Tell Microprocessor (about an operation)
- TUPR0 - Tell Microprocessor (about a Reset or Interface Disconnect)
- TUPRST - Tell Microprocessor of Reset
- WRRFDA - Write Register File Device Address
- WRRFDM - Write Register File Device Modifier
- WRRFDS - Write Register File Device State
- WRRFEX - Write Register File Extra Status

Subroutine CHKPAR (Check Parity)

Set-Up

Register 1 must be cleared by the calling routine. Data Out (DAT/) must be low.

Called By

Write Data Transfer

Process Description

CHKPAR tests the channel parity bit. If an error is detected, CHKPAR tests Data Out (DAT/). If DAT/ is still low, CHKPAR sets Register 1 to a value of 4 (0100B).

Subroutines Called

None

Registers Modified

Register 1 is set to a value of 4 if a parity error is detected.

Subroutine FETSTA (Fetch Status)

Set-Up

None

Called By

CUIS, BUSY

Process Description

FETSTA extracts the Processor Status and Extra Status from the Register File, reformats this status, and loads it into Bit-Slice ALU registers 6, 7 and 8. Processor Status is determined by the microprocessor; Extra Status is determined by the Bit-Slice Processor.

FETSTA has two special purpose paths, a System Reset flag path and a Disconnect flag path. FETSTA sets the Control Unit End status if the System Reset flag is set. The System Reset flag is set by SYSRES if the channel attempts to send a command between the start of a system reset and its completion. If this happens, branch routine RPTBSY (entered from IMRES) sends a status of Control Unit Busy to the channel. A status of Control Unit Busy suspends all channel activity to the GPCI until a Control Unit End status is sent to the channel. FETSTA sets up this status if the System Reset flag is set. Then it returns to the calling routine (CUIS, in this case).

When disconnecting, FETSTA sets the Unit Exception status bit and returns.

Subroutines Called

RDRFPS, RDRFEX

Registers Modified

Registers 6, 7 and 8

Subroutine GETCMD (Get Command)

Set-Up

Used only during an Initial Selection Sequence.

Called By

BUSY, ISS

Process Description

GETCMD proceeds through the Initial Selection sequence until the command (on Bus Out) can be latched into Bit-Slice Processor hardware. Then, GETCMD fetches the device type from the Register File. The device types are listed below.

CODE	Device Type
00B	Bisync
01B	Ingestor
10B	ProNET

Together, the command and device type address a command look-up table, whose output is a 4-bit device specific command code. This code is held in Register 3 upon return to the calling routine. Once the device specific code is generated, the state of the specified device is tested to see if it is "ready". If it isn't in a ready state, GETCMD returns.

If the device is ready but the command contains a parity error, Register 1 is loaded with AH, the device state is changed to Pending Status, the return address is popped from the stack, and GETCMD exits to RPRTUC (Report Unit Check).

If the command contains no parity errors but is not a legal command for that device, the exit procedure is as described for a parity error except Register 1 contains an 8H. If the command was legal, GETCMD returns. While GETCMD is executing, it frequently tests for Interface Disconnects and Selective Resets. If either condition occurs, GETCMD pops the return address from the stack and jumps to INTDIS (see page 8-49) or SELRES (see page 8-51).

Subroutines Called

MCLTIN, MADBIN, RDRFDT

Registers Modified

Register 3 contains the device specific command.
Register 1 = AH (if parity error)
Register 1 = 8H (if illegal command)
Register 4 bit 2 set

Subroutine IMRES (Immediate Response)

See page 8-25.

Subroutine IMRES4 (Immediate Response Section 4)

Set-Up

None

Called By

CONTCH, used as in-line code by IMRES

Process Description

IMRES4 is a part of IMRES. It programs PAL 21 to drive RESPIM/ low if LSHT/ goes low. It also suspends propagation of Select Out by lowering PSOUT.

Subroutines Called

MCLTIN

Registers Modified

Register E bit 0 set
Register 4 bit 1 reset

Subroutine MADBIN (Move Address to Bus In)

Set-Up

Register AH contains the device address.
Register BH contains the base address.

Called By

CUIS, GETCMD

Process Description

MADBIN latches the base and device address into Bit-Slice hardware, forming an 8-bit I/O address. It transfers this 8-bit address to Bus In via the Channel Drivers and Receivers board.

Subroutines Called

None

Registers Modified

None

Subroutine MCLTIN (Move Control to Tag In)

Set-Up

Register 4 contains the low nibble of the channel control word.
Register 5 contains the high nibble of the channel control word.

Called By

HSTART, DISCON, CONTCH, STPOL, IMRES, CUIS, CMDDNE, GETCMD, ISS, INTDIS, SELRES, SYSRES

Process Description

MCLTIN latches Registers 4 and 5 into Bit-Slice hardware to form an 8-bit wide control byte. This control byte is transferred to the Tag and Control Latch on the Channel Drivers and Receivers board.

Subroutines Called

None

Registers Modified

None

Subroutine MSTBIN (Move Status to Bus In)

Set-Up

Register 6 contains the four LSBs of the Status byte.
Register 7 contains the four MSBs of the Status byte.

Called By

HSTART, DISCON, CMDDNE, ISS, SYSRES

Process Description

Same as MCLTIN above, except the byte is transferred to Bus In.

Subroutines Called

None

Registers Modified

None

Subroutines RDRFEX, RDRFDS, RDRFDT, RDRFPS

Set-Up

Prior to calling, the device address must be written to the Bit-Slice Processor's output data bus (ODAT0-ODAT3).

Process Description

RDRFEX, RDRFDS, RDRFDT and RDRFPS fetch Extra Status, Device State, Device Type, and Processor Status respectively from the Register File. Table 28 below shows destination registers and calling routines. None of these subroutines call other subroutines.

Subroutine	Data Destination Register	Calling Routines
RDRFEX	8	STPOL, FETSTA, INTDIS
RDRFDS	0	STPOL, IMRES, CUIS
RDRFDT	9	CONTCH, GETCMD
RDRFPS	6	FETSTA

Table 28. Read Subroutines Summary

Subroutine TCANSD (Test Command and Service Down)

Set-Up

None

Called By

CMDDNE, ISS

Process Description

TCANSD is a loop that exits to the calling routine only when Command Out and Service Out are simultaneously low (COT/ and SET/ are simultaneously high). While waiting, TCANSD tests for Interface Disconnects and Selective Resets.

Subroutines Called

None (conditionally jumps to INTDIS or SELRES after popping the return address from the stack)

Registers Modified

None

Subroutine TCORSU (Test Command Out or Service Out Up)

Set-Up

None

Called By

CUIS, CMDDNE, ISS

Process Description

TCORSU is a loop which exits to the calling routine if Command Out or Service Out is high (COT/ or SET/ is low). Like TCANSD, TCORSU also tests for Interface Disconnects and Selective Resets.

Subroutines Called

None (conditionally jumps to INTDIS or SELRES after popping the return address from the stack)

Registers Modified

None

Subroutine TELLUP (Tell Microprocessor)

Set-Up

Register 1 is loaded with a device modifier code and Register 9 is loaded with the address of the device for which the command is intended.

Process Description

TELLUP informs the microprocessor of a new command by writing the device modifier and device address to address 680FH in the Register File, then interrupting it. TELLUP begins by entering the first of two loops. The first loop waits for the microprocessor to complete any in-progress interrupt. TELLUP drives RFINT high, interrupting the microprocessor. Then it enters the second loop, where it waits for acknowledgement of receipt of the interrupt.

Within both loops, TELLUP monitors for Selective Resets and Interface Disconnects. When the microprocessor is interrupted, it reads address 680FH to determine which device gets the new command, and uses the device modifier as an address offset in an interrupt handler look-up table. Only two device modifiers are currently acceptable, 0000B and 0010B. Respectively, these modifiers cause vectoring into "normal" and "abnormal" command handlers within the 8085 firmware.

Subroutines Called

WRRFDM, WRRFDA (can jump to SELRES or INTDIS)

Registers Modified

None

Subroutine TUPR0

See subroutine TUPRST below.

Subroutine TUPRST (Tell Microprocessor about a Reset)

Set-Up

Registers 1 and 9 are set up the same as TELLUP. See Table 29 below.

Called By	Device Modifier	Device Address
SYSRES	8	F
SELRES	2	0-E
INTDIS *	2	*0-E
DISCON	6 (Time Out)	-
CONTCH	6 (Time Out)	

*INTDIS also sets the Equipment Check bit (LSB of 680FH) to distinguish Interface Disconnect from Selective Reset.

Table 29. Subroutine TUPRST Summary

Called By

TUPRST: INTDIS, SELRES, SYSRES
TUPR0: DISCON, CONTCH

Process Description

These subroutines are identical to TELLUP, except there is no testing for Selective Resets or Interface Disconnects. TUPR0 is a second entry point in TUPRST. The only difference between them is that TUPR0 does not pass a device address to the Register File.

Subroutines Called

WRRFDA, WRRFDM

Subroutines WRRFDA, WRRFEX, WRRFDM, WRRFDS

Set-Up

Prior to calling, the device address must be written to the Bit-Slice output data bus (ODAT0-0DAT3). Except for WRRFDA, Register 1 must be loaded with data to be written. The data source register for WRRFDA is Register 9.

Process Description

These subroutines transfer data nibbles to the Register File. Table 30 below shows which subroutine writes to which nibble of the Register File.

Nibble	6800-680F	6810-681F
High Nibble	WRRFDA* WRRFDS	(Microprocessor only)
Low Nibble	WRRFDM	WRRFEX

* WRRFDA uses Register 9 as the data source and is used to write to 6800H-680EH. WRRFDS uses Register 1 as the data source and is used to write to 680FH only.

Table 30. Write Subroutines Data Transfers

Table 31 below shows source registers and calling routines.

Source Subroutine	Register	Calling Routines
WRRFDA	9	TELLUP, TUPRST
WRRFEX	1	CMDDNE, BUSY, INTDIS, SELRES
WRRFDM	1	CMDDNE, TELLUP, TUPR0
WRRFDS	1	HSTART, CONTCH, STPOL, CMDDNE, ISS, INTDIS, SELRES

Table 31. Write Subroutines Summary

Branch Routine Descriptions

Except for entries to and exits from these routines, they are very similar to subroutines. They are entered from other routines via a jump instruction. The exit from these routines is via a jump to a particular location in the program, such as the beginning of STPOL.

The branch routines discussed in the Detailed Bit-Slice Microcode description will not be addressed here. This section describes the following routines:

- INTDIS (Interface Disconnect)
- RPRTUC (Report Unit Check)
- SELRES (Selective Reset)
- SYSRES (System Reset)

Branch Routine INTDIS (Interface Disconnect)

Entered From

GETCMD, Write Data Transfer, RDXFER, TELLUP, TCANSD, TCORSU

Set-Up

None

Process Description

The channel can terminate an ongoing I/O operation at any time by sending an Interface Disconnect sequence to the GPCI. The two ways the channel can indicate an Interface Disconnect are:

- if Hold Out is down and Address Out rises, or
- if Address Out is high and Hold Out falls

When one of the routines detects one of the INTDIS Interface Disconnect sequences, it jumps to INTDIS (subroutines must pop return addresses from the stack first). INTDIS shuts off the autotransfer hardware by raising EOX. Next, it saves all Extra-Status by ORing the ALU's Extra Status register (Register 8) with the device's extra status in the Register File. The ORed value is stored in the Register File. Now, all tag lines and Operational In are lowered, telling the channel that the disconnect has been performed.

Next, the device state is stored (as is) in the Register File, and the Pending Status and Stacked Status flags are lowered. Finally, INTDIS sets the Equipment Check bit in the Register File, which informs the microprocessor of the disconnect by calling TUPRST.

Subroutines Called

RDRFEX, WRRFEX, WRRFDS, MCLTIN, TUPRST

Exits To

STPOL, or SELRES if the channel performs a Selective Reset while INTDIS is waiting for Address Out to fall or Hold Out to rise.

Registers Modified

Register 1 = 2H
Register 4 = 0H
Register 5 - LSB is reset (Operational In)
Register F - the 2 LSBs are reset

Branch Routine RPRTUC (Report Unit Check)

Entered From	GETCMD, Write Data Transfer, BUSY
Set-Up	Register 1 is loaded with Sense data before jumping to RPRTUC.
Process Description	RPRTUC stores the Sense data in the Register File, sets up the "abnormal command" indicator (Device Modifier = 2) and interrupts the microprocessor by calling TELLUP.
Subroutines Called	WRRFEX, TELLUP
Exits To	CMDDNE
Registers Modified	Registers 6 and 8 are ORed with 00100 (masks Unit Check into Status and Extra Status).

Branch Routine SELRES (Selective Reset)

Used By

IMRES, CUIS, GETCMD, Write Data Transfer, RDXFER, INTDIS, RPTBSY, TELLUP, TCANSD, TCORSU

Set-Up

Operational Out is low (OPT/ is high).

Process Description

SELRES does additional testing to determine if this is a Selective Reset or a System Reset. In either case, SELRES drops all tag and control lines, but does not turn off the relays on the Channel Drivers and Receivers board. Also, RESPIM/ is reconfigured to monitor the rising edge of LSHT/. The Stacked Status and Pending Status flags are reset. If the reset is a System Reset, SELRES exits to SYSRES. If this is a Selective Reset, the device state is set to Busy and the microprocessor is interrupted by calling TUPRST.

Subroutines Called

MCLTIN, SYSRES, WRRFEX, WRRFDS, TUPRST

Exits To

CONCH8 (part of WATCON)

Registers Modified

Register 1 = 2H
Register 4 = 0H
Register 5 LSB is reset
Register E = 0H
Register F the 2 LSBs are reset

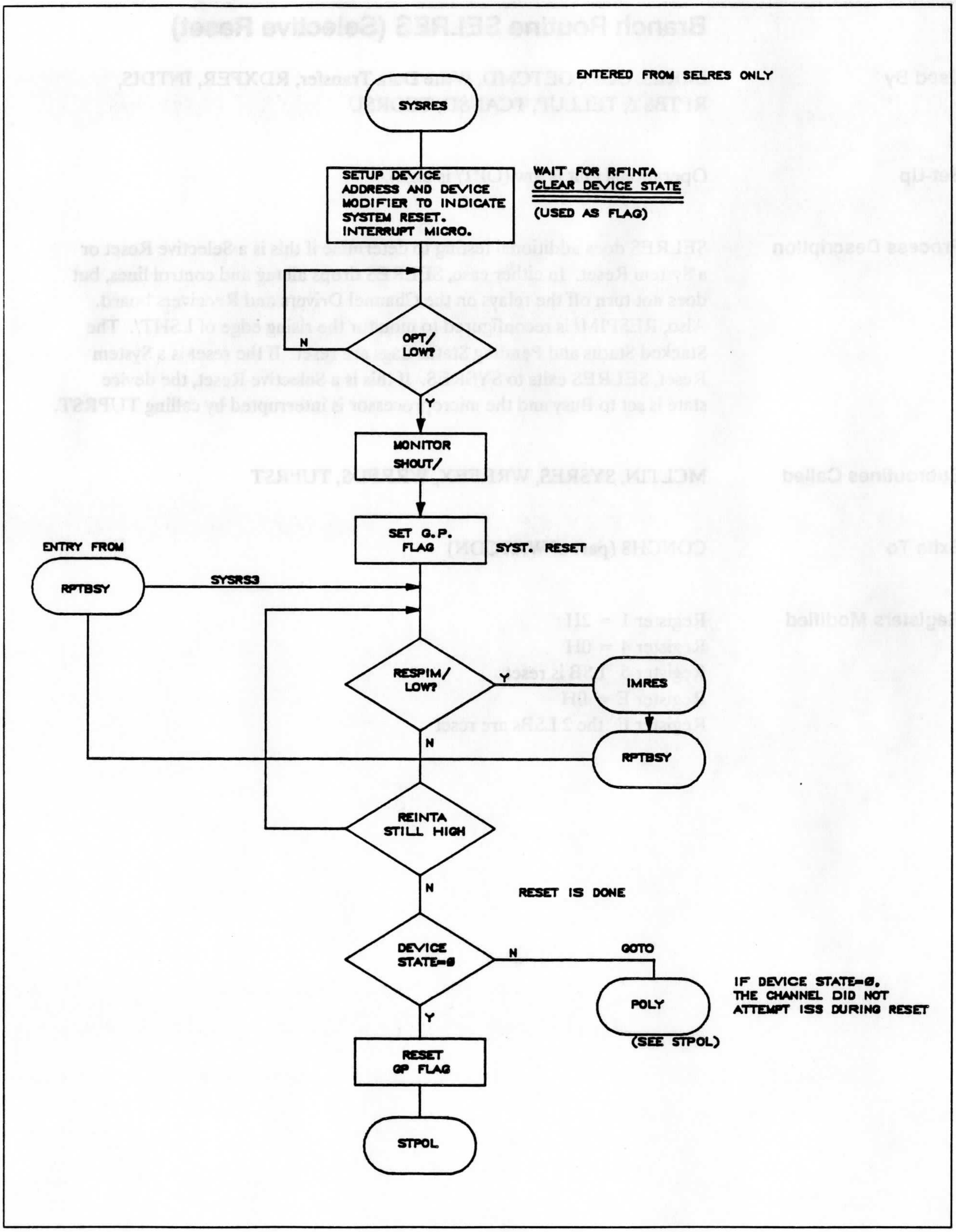


Figure 17. System Reset Flow Diagram

Branch Routine SYSRES (System Reset)

Entered From

SELRES

Set-Up

Operational Out and Suppress Out are down concurrently (OPT/ and SUT/ are high).

Process Description

Refer to the SELRES description on page 8-51. SYSRES begins by setting up the device address (FH) and device modifier (8H) to indicate a System Reset. Then it sets the device state to 0, and interrupts the microprocessor, by calling TUPRST. SYSRES waits for Operational Out to go high (OPT/ to go low before reconfiguring the RESPIM/ generator to monitor for LSHT/going low). Next, SYSRES sets the General Purpose (GP) flag.

Refer to Figure 17 on the adjacent page. The microprocessor acknowledges the System Reset interrupt by raising RFINTA. RFINTA remains high until the microprocessor resets all operational devices. While the microprocessor is performing the resets, the Bit-Slice Processor enters a loop which tests RESPIM/ and RFINTA.

If RFINTA remains high the processor is still busy, and the loop repeats. If RESPIM/ goes low (the channel issues a new command for one of the devices), IMRES is called. IMRES jumps to RPTBSY (part of SYSRES) which sends a status of Control Unit Busy to the channel and sets the device state register to 1 (the device state register is used as a flag). RPTBSY resumes looping.

When RFINTA goes low (the microprocessor is finished), SYSRES tests the device state register. If it is still 0, it means the channel didn't attempt to send a command during the reset. In this case, SYSRES resets the GP flag and exits to STPOL. However, if the device state register is 1, SYSRES exits to POL4, sets the Pending Status flag, raises Request In, and jumps to the top of STPOL.

The channel responds to the raised Request In line by raising Select Out and Hold Out, causing RESPIM/ to go low. When RESPIM/ goes low, STPOL calls IMRES. IMRES determines that this is a Control Unit Initiated Sequence and jumps to CUIS. CUIS calls FETSTA which tests the GP flag. The set flag causes FETSTA to set a status of Control Unit End and exit to CMDDNE.

CMDDNE sends the Control Unit End status and resets the GP flag, completing the reset process. The Control Unit End status is required to reopen communications to the channel which were suspended by the Control Unit Busy reply to the command received during the reset.

Subroutines Called TUPRST, IMRES, POL4, STPOL, MSTBIN, MCLTIN, SELRES

Registers Modified Register A = FH
Register 1 = 8H
Register 0 = 0H or 1H (see description)
Register E = 1H
Register F = 8H or 0H
Register 6* = 0
Register 7* = 5

***These registers are modified only if a command is received during the reset.**

GPCI Supplemental Data

GPCI Supplemental Data

This Supplemental Data section contains:

- an explanation of the PAL equation listings
- the symbols and abbreviations used in the equations
- twenty-one PAL equations
- Schematic drawings 1 through 7 (6450-0465)
- Assembly drawing 1 (6450-0466)

PAL Equation Listings

The listings for the logic equations used in the Programmable Array Logic Devices are explained below.

Title

Each device has a printout that begins with the title page. On this page the device is called out by its location and device name.

Declarations

Declarations lists the logic conventions and format for the signal names. The signal names are in uppercase letters and numbers.

The first group of signal names refers to the power and ground rails. The respective pin numbers that make these connections are given in the line below the signal names.

The second group of signal names refers to the input signals received by the device. The respective pin numbers assigned to receive these inputs are given in the line below the signal names.

The third group of signal names refers to the outputs from the PAL. The respective pin numbers assigned to each output are given in the line below the signal names.

Equations

The logic equations used to generate each signal are shown in their highest order form.

Symbols and Abbreviations

The following is an example of a PAL equation.

```
!PDOT_  :=  !DIN & DSEL_ & QBSEL_ & DRQ & DMACLK
          # PDOT & !DIN & DSEL & QBSEL_ & DRQ
          # !PDOT_ & DAK_
```

Below is an explanation of the symbols and abbreviations used in the equation.

Symbol/Abbreviation	Explanation
!	The one's complement of, e.g. !PDOT = ($\overline{PDOT}$)
:	A register latched signal; valid on the rising edge of the register clock
&	Logical AND
#	Logical OR
_	Denotes an active low signal

P22V10 Located at BE16

Title

Latch Mux and File Address
SSEC Madison, WI 05/05/86

Identification

Device location BE16
Device type P22V10

Module

GPCI1
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 12, 24

CLK, A0, A1, B0, B1, B2, B3
        pin 1, 2, 3, 4, 5, 6, 7

OD0, OD1, OD2, OD3
        pin 8, 9, 10, 11

RFLOCK, EOX_, RIM_, PEN_, GO
        pin 13, 20, 21, 22, 23

ARF1, ARF2, ARF3, ARF4, EOXL
        pin 17, 16, 15, 14, 18

CC_
        pin 19

MUX = [B3, B2, B1, B0]

DAT = [OD3, OD2, OD1, OD0]

LK = RFLOCK
```

Equations

```
enable CC_ = !A0  & !A1

!CC_          =  (MUX==0) & ARF1
              #  (MUX==1) & ARF2
              #  (MUX==2) & ARF3
              #  (MUX==3) & ARF4
              #  (MUX==4) & EOXL
              #  (MUX==5) & RIM_
              #  (MUX==6) & PEN_
              #  (MUX==7) & GO
              #  (MUX==8) & !ARF1
              #  (MUX==9)  & !ARF2
              #  (MUX==^hA) & !ARF3
              #  (MUX==^hB) & !ARF4
              #  (MUX==^hC) & !EOXL
              #  (MUX==^hD) & !RIM_
              #  (MUX==^hE) & !PEN_
              #  (MUX==^hF) & !GO

enable  ARF1   =   GO

ARF1       :=  OD0  &  !RFLOCK
              # ARF1 &  RFLOCK

enable  ARF2   =   GO

ARF2       :=  OD1  &  !RFLOCK
              # ARF2 &  RFLOCK

enable  ARF3   =   GO

ARF3       :=  OD2  &  !RFLOCK
              # ARF3 & RFLOCK

enable  ARF4   =   GO

ARF4       :=  OD3  &  !RFLOCK
              # ARF4 & RFLOCK

EOXL       :=  !EOX_
              # EOXL & !PEN_
```

P22V10 Located at BE1 & BG1

Title

Latch and Multiplexer
SSEC MADISON, WI 05/05/86

Identification

Device locations BE1 & BG1
Device type P22V10

Module

GPCI2
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 12, 24

CLK, AD, ADR, B0, B1, B2, B3
        pin 1, 2, 3, 4, 5, 6, 7

I1, I2, I3, I4, I5, I6, I7
        pin 8, 9, 10, 11, 13, 14, 23

Q1, Q2, Q3, Q4, Q5, Q6, Q7
        pin 15, 16, 17, 18, 20, 21, 22

CC_
        pin 19

MUX = [B3, B2, B1, B0]
```

Equations

```
enable  CC_ =AD & !ADR

!CC_       = (MUX= =0) & 1          " always
           # (MUX= =1)  & Q1
           # (MUX= =2)  & Q2
           # (MUX= =3)  & Q3
           # (MUX= =4)  & Q4
           # (MUX= =5)  & Q5
           # (MUX= =6)  & Q6
           # (MUX= =7)  & Q7
           # (MUX= =8)  & 0           " never
           # (MUX= =9)  & !Q1
           # (MUX= = ^hA) & !Q2
           # (MUX= = ^hB) & !Q3
           # (MUX= = ^hC) & !Q4
           # (MUX= = ^hD) & !Q5
           # (MUX= = ^hE) & !Q6
           # (MUX= = ^hF) & !Q7

Q1         := I1
Q2         := I2
Q3         := I3
Q4         := I4
Q5         := I5
Q6         := I6
Q7         := I7
```

P16R8 Located at BC14

Title

Bit-Slice Output Register
SSEC Madison, WI 05/05/86

Identification

```
Device location  BC14
Device type      P16R8
```

Module

```
GPCI4
flag -r0
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 10, 20

CLK, NC, OCS_, AD4, AD5, OD3, OD2, OD1, OD0, OE_
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 11

DC7, DC6, DC5, DC4, DC3, DC2, DC1, DC0
        pin 12, 13, 14, 15, 16, 17, 18, 19
```

Equations

```
!DC0      := !OD0 & !AD4 & !AD5 & !OCS_
            # !DC0 & AD4
            # !DC0 & AD5
            # !DC0 & OCS_

!DC1      := !OD1 & !AD4 & !AD5 & !OCS_
            # !DC1 & AD4
            # !DC1 & AD5
            # !DC1 & OCS_

!DC2      := !OD2 & !AD4 & !AD5 & !OCS_
            # !DC2 & AD4
            # !DC2 & AD5
            # !DC2 & OCS_

!DC3      := !OD3 & !AD4 & !AD5 & !OCS_
            # !DC3 & AD4
            #  !DC3 & AD5
            # !DC3 & OCS_
```

```
!DC4      :=  !OD0 & AD4 & !AD5 & !OCS_
               # !DC4 & !AD4
               # !DC4 & AD5
               # !DC4 & OCS_

!DC5      :=  !OD1 & AD4 & !AD5 & !OCS_
               # !DC5 & !AD4
               #  !DC5 & AD5
               # !DC5 & OCS_

!DC6      :=  !OD2 & AD4 & !AD5 & !OCS_
               #  !DC6 & !AD4
               # !DC6 & AD5
               # !DC6 & OCS_

!DC7      :=  !OD3 & AD4 & !AD5 & !OCS_
               # !DC7 & !AD4
               # !DC7 & AD5
               # !DC7 & OCS_
```

P16R4 Located at BE43

Title

Automatic Control Decoder
SSEC Madison, WI 05/05/86

Identification

Device location BE43
Device type P16R4

Module

GPCI5
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 10, 20

CLK, CEN_, AUS_, AEN_, ID0, ID1, OSL_, ID3, NC, OE_
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 11

DAEN_, CM3_, CM1_, CM0_
        pin 14, 15, 16, 17

AUX_, GBS_, AUD_, AUM_
        pin 12, 13, 18, 19
```

Equations

```
!CM0_     :=  ID0 & !CEN_
            # !CM0_ & CEN_

!CM1_     :=  ID1 & !CEN_
            # !CM1_ & CEN_

!CM3_     :=  ID3 & !CEN_
            # !CM3_ & CEN_

!DAEN_    :=  !AEN_

enable GBS_ = TRUE

!GBS_      =  !AEN_ & !DAEN_ & AUS_ & !OSL_
            # !AEN_ & !DAEN_ & !AUS_

enable AUD_ = TRUE

!AUD_      =  !AUS_ & !CM1_
```

enable AUX_ = TRUE

!AUX_ = !AUS_ & CM1_

enable AUM_ = TRUE

!AUM_ = !AUS_ & !CM0_ & CM1_

Notes

There are two functions performed by this part. The first is to latch bits from the Command PROM and, when enabled, to provide control lines that indicate which state machine (DMA or Autotransfer) will provide the data transfer.

The second is to delay AEN (Address Enable) so that there is enough time for the Multibus address and data lines to settle before the control lines are enabled by GBS_ (GOTBUS).

P16R8 Located at BG43

Title

Register File State Machine
SSEC Madison, WI 05/05/86

Identification

```
Device location  P16R8
Device type      BG43
```

Module

```
GPCI6
flag -r0
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin  10, 20

BSCLK, RFLK, BREN_, RFCS_, WR, AD7, RST_, OE_
        pin  1, 2, 3, 4, 5, 6, 9, 11

BRWR, BRRD_, RQD_, RQC_, RQB_, RQA_, RFWE_, DCS_
        pin  12, 13, 14, 15, 16, 17, 18, 19
```

Equations

```
!DCS_     :=  !RFCS_

!BRRD_    :=  RQA_ & RFLK & !BREN_

!BRWR     :=  AD7

!RFWE_    :=  RQA_ & RFLK & !BREN_ & AD7
            # !RQA_ & RQB_ & RQC_ & !RQD_ & WR
            # !RQA_ & !RQB_ & !RQC_ & !RQD_ & WR

!RQA_     :=  !RQB_
            # !RQC_
            # !RQA_ & !RQD_
            # !RQA_ & !DCS_
            # RQA_ & RQD_ & !DCS_ & !RFLK

!RQB_     :=  !RQC_

!RQC_     :=  RQB_ & !RQC_
            # !RQA_ & RQB_ & !RQD_
```

```
!RQD_    := RQA_ & RQD_ & !DCS_
            # RQA_ & RFLK
            # !RQC_ & !RQD_
            # !RQA_ & RQB_ & !RQD_ & WR
```

Notes

DCS_ synchronizes the register file chip select from the microprocessor.

P16L8 Located at BG16

Title

Register File Output Decode
SSEC Madison, WI 05/05/86

Identification

Device location BG16
Device type P16L8

Module

GPCI7
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 10, 20

AB4, AD4, AD5, AD6, RD, RQA_, RQB_, RQC_, RQD_
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9

AF6     pin 12

AF5, AF0, HNEN_, LNEN_, RROE_, RFAK_
        pin 13, 14, 15, 16, 17, 18
```

Equations

```
enable RFAK_ = TRUE

!RFAK_    = !RQA_ & !RQB_ & RQC_
          # !RQA_ & RQB_ & RQC_ & RQD_

enable RROE_ = TRUE

!RROE_    = !RQA_ & RD

enable LNEN_ = TRUE

!LNEN_    = !RQA_ & RQB_ & !RQC_ & !RQD_

enable HNEN_ = TRUE

!HNEN_    = !RQA_ & !RQB_ & RQC_ & !RQD_
```

```
enable AF0 = TRUE

!AF0        =  !RQA_ & RQB_
               # RQA_ & !AD4
               # RQB_ & !AD4

enable AF5 = TRUE

!AF5        =  !RQA_ & !AB4
               # RQA_ & !AD5
               # !AB4 & !AD5

enable AF6 = TRUE

!AF6        =  !RQA_
               # !AD6
```

Notes

```
IF (VCC)  AF0   = !RQA_  * !RQB_

             + RQA_  *  AD4

IF (VCC)  AF5   =  RQA_  AD5

             + !RQA_  * AB4

IF (VCC)  AF6   =RQA_  *  AD6
```

P16R8 Located at BG29

Title

Register File Output Latch
SSEC Madison, WI 05/05/86

Identification

Device location BG29
Device type P16R8

Module

GPCI8
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 10, 20

CLK_, DR3, DR2, DR1, DR0, RQD_, RQC_, RQB_, RQA_, OE_
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 11

DB7_, DB6_, DB5_, DB4_, DB3_, DB2_, DB1_, DB0_
        pin 12, 13, 14, 15, 16, 17, 18, 19
```

Equations

```
!DB0_     := !RQA_ & RQB_ & !RQC_ & RQD_ & DR0
             # RQA_ & !DB0_
             # !RQB_ & !DB0_
             # RQC_ & !DB0_
             # !RQD_ & !DB0_

!DB1_     := !RQA_ & RQB_ & !RQC_ & RQD_ & DR1
             # RQA_ & !DB1_
             # !RQB_ & !DB1_
             # RQC_ & !DB1_
             # !RQD_ & !DB1_

!DB2_     := !RQA_ & RQB_ & !RQC_ & RQD_ & DR2
             # RQA_ & !DB2_
             # !RQB_ & !DB2_
             # RQC_ & !DB2_
             # !RQD_ & !DB2_
```

```
!DB3_     := !RQA_ & RQB_ & !RQC_ & RQD_ & DR3
             # RQA_ & !DB3_
             # !RQB_ & !DB3_
             # RQC_ & !DB3_
             # !RQD_ & !DB3_

!DB4_     := !RQA_ & !RQB_ & RQC_ & RQD_ & DR0
             # RQA_ & !DB4_
             # RQB_ & !DB4_
             # !RQC_ & !DB4_
             # !RQD_ & !DB4_

!DB5_     := !RQA_ & !RQB_ & RQC_ & RQD_ & DR1
             # RQA_ & !DB5_
             # RQB_ & !DB5_
             # !RQC_ & !DB5_
             # !RQD_ & !DB5_

!DB6_     := !RQA_ & !RQB_ & RQC_ & RQD_ & DR2
             # RQA_ & !DB6_
             # RQB_ & !DB6_
             # !RQC_ & !DB6_
             # !RQD_ & !DB6_

!DB7_     := !RQA_ & !RQB_ & RQC_ & RQD_ & DR3
             # RQA_ & !DB7_
             # RQB_ & !DB7_
             # !RQC_ & !DB7_
             # !RQD_ & !DB7_
```

P20R4 Located at BC41

Title

Channel Latch Controller
SSEC Madison, WI 02/26/87

Identification

Device location BC41
Device type P20R4

Module

GPCI9
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 12, 24

CLK, SEO_, DAO_, EOXL, CDR, CDK_, CPCS_, OD0, OD1, DRT_, CIN,
OE_, PAN_, DONE
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 23

AUS_, DEL_, LCDR_, QA_
        pin 17, 18, 19, 20

PRCK_, DTR_, DAN_, SEN_
        pin 15, 16, 21, 22
```

Equations

```
enable DAN_ = TRUE

!DAN_       = !AUS_ & !LCDR_ & DAO_
            # !AUS_ & !LCDR_ & QA_
            # !AUS_ &  LCDR_ & DAO_ & !DRT_
            # !AUS_ &  LCDR_ & DAO_ & !DAN_
            # !AUS_ &  LCDR_ & !DRT_ & !DAN_

enable SEN_ = TRUE

!SEN_       = FALSE

enable PRCK_ = TRUE

!PRCK_      = AUS_ & !CDK_ & CIN
            # !AUS_ & !QA_ & DEL_ & !LCDR_

enable DTR_ = TRUE
```

```
!DTR_     =  !AUS_ & !LCDR_ & !DAO_ & !DEL_ & DRT_
           # !AUS_ & !LCDR_ & !DAO_ & !DTR_ & DAN_
           # !AUS_ & !LCDR_ & DRT_ & !DTR_
           # !AUS_ &  LCDR_ & DAO_ & !DRT_
           # !AUS_ &  LCDR_ & DAO_ & !DTR_
           # !AUS_ &  LCDR_ & !DRT_ & !DTR_

!QA_     :=  !AUS_ & !LCDR_ & !DAO_ & DTR_ & DRT_
           # !AUS_ & !LCDR_ & !DAO_ & !QA_
           # !AUS_ &  LCDR_ & !DAO_

!DEL_    :=  !AUS_ & !QA_

!LCDR_   :=  !PAN_ & CDR
           # !LCDR_ & !AUS_

!AUS_    :=  !PAN_ & !DONE & !EOXL & CIN
           # !AUS_ & PAN_ & !EOXL & !DAN_ & QA_
           # !AUS_ & DONE & !EOXL & !DAN_ & QA_
```

Notes

When CDR is high, it is a read from the D/R board (IBM write). When CDR is low, it is a write to the D/R board (IBM read).

As currently designed, this PAL uses only the data in the (DAN_) tag line during a transfer of data. Service is not used.

AUS_ stands for automatic state. It is true when the transfer of data is actually occurring. If Processor Auto Enable (PAN_) goes false during the middle of a transfer, AUS_ is extended until the completion of the transfer. This is true except when the channel quits first (indicated by "COMMAND OUT) and the B-S yanks the plug by raising EOXL.

During an IBM write, the Data Out (DAO_) signal is delayed for a clock period to allow the proper setup time for the bidirectional latch. This is the function of DEL_.

Data streaming probably can be supported by using a PAL22VIOA and a change in microcode.

P22V10 Located at BA26

Title

DMA Controller
SSEC Madison, WI 05/05/86

Identification

Device location BA26
Device type P22V10

Module

GPCI10
flag -r0

Declarations

TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
pin 12, 24

CLK, DTR_, CDR, ADMA_, DGO_, AMF, XACK_, LDONE
pin 1, 2, 3, 4, 5, 7, 9, 10

DRT_, CHCK_, CHEN_, DMAI_, DREQ_, DCTL_
pin 14, 15, 16, 17, 22, 23

DQA, DQB, DQC, DQD
pin 18, 19, 20, 21

CCK_, EN_, AI_, RQ_, CL_ = CHCK_, CHEN_, DMAI_, DREQ_, DCTL_

DST = [DQA, DQB, DQC, DQD]

XAK_, DNE = XACK_, LDONE

State Definitions

a = ^b0000 b = ^b0001 c = ^b1101 d = ^b1001 e = ^b1011

f = ^b1010 g = ^b1000 h = ^b0101 i = ^b0100 j = ^b0110

k = ^b0010 l = ^b0011

Equations

```
DQD     := !DNE & DQA & !DQC
           # !DNE & !DQA & !DQB & DQD
           # !DNE & !DQB & !DQC & !ADMA_
           # !DNE & !DQC & DQD & DTR_ & !DRT_
           # !DNE & !DQA & !DQB & DQC & !XACK_
           # !DNE & DQB & DQC & !AMF

DQC     := !DNE & !DQA & DQC & !DQD
           # !DNE & DQB & !DQD & DTR_
           # !DNE & DQC & !DQD & XACK_
           # !DNE & DQA & DQC & DQD & AMF
           # !DNE & DQA & !DQB & !DQC & DQD & !DTR_

DQB     := !DNE & !DQA & DQB & !DQC
           # !DNE & DQB & DQD & !DTR_ & !DRT_
           # !DNE & !DQA & !DQC & DQD & !DGO_

DQA     := !DNE & DQA & DQC
           # !DNE & DQA & DQD
           # !DNE & !DQB & !DQC & DQD & !DGO_ & CDR

!DCTL_  := !DNE & !DQA & DQC & !DQD
           # !DNE & !DQA & DQB & !DQD
           # !DNE & DQA & DQC & DQD
           # !DNE & DQA & !DQB & DQD & !DTR_
           # !DNE & DQA & DQC & XACK_

!DREQ_  :=  !DNE & !DQA & DQD
           # !DNE & DQB & !DQC
           # !DNE & DQA & !DQC
           # !DNE & !DQB & DQD & AMF
           # !DNE & DQB & DQC & AMF
           # !DNE & !DQB & DQC & !DQD & XACK_

!DMAI_  := !DNE & DQB & DQC & !AMF
           # !DNE & DQA & DQC & DQD & !AMF
           # !DNE & !DQB & DQC & !DQD & !XACK_

 enable DRT_  = !ADMA_

!DRT_   := !ADMA_ & !CDR & !DQA & !DQB & DQC & DQD
           # !ADMA_ & !CDR & !DRT_ & DTR_
           # !ADMA_ & CDR & DQA & !DQB & !DQC & DQD & !DTR_
           # !ADMA_ & CDR & !DRT_ & !DCTL_
           # !ADMA_ & CDR & !DRT_ & !DTR_

enable CHEN_  = !ADMA_

!CHEN_  := !ADMA_ & CDR
```

```
enable CHCK_  = !ADMA_

!CHCK_  := !DQA   & DQB  & DQC & !DQD & !AMF
         # !DQA   & !DQB & DQC & !DQD & !XACK_
```

P22V10 Located at BA41

Title

Autotransfer Controller
SSEC MADISON, WI 11/10/86 and 5/07/87

Identification

```
Device location  BA41
Device type      P22V10
```

Module

```
GPCI11
flag -r0
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 12, 24

CLK, DTR_, CDR, RST_, AMBS_, AUX_, MWT_, MRD_, WR, RD, AGO_,
PAUEN_
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13

QD_, QC_, QB_, QA_
        pin 17, 18, 19, 20

DRT_, CHCK_, CHEN_, PAK_, NC1
        pin 15, 16, 21, 22, 23

CL = CLK
DT = DTR_
CD = CDR
RS = RST_
AM = AMBS_
AX = AUX_
MW = MWT_
MR = MRD_
GO = AGO_
DR = DRT_
CC = CHCK_
CN = CHEN_
PK = PAK_

OUT = [QA_, QB_, QC_, QD_]
```

State Definitions

```
a = !^b0000   b = !^b0001   c = !^b0011
d = !^b0111   e = !^b0101   f = !^b0100
g = !^b1001   h = !^b1101   i = !^b1100
j = !^b1000
```

Equations

```
enable DRT_   = !AUX_

!DRT_    := RST_ & !QB_ & QC_ & !QD_
           # RST_ & QA_ & !QB_ & QC_ & AMBS_ & RD
           # RST_ & QA_ & !QB_ & QC_ & !AMBS_ & !MRD_
           # RST_ & QA_ & !QB_ & !QD_ & AMBS_ & RD
           # RST_ & QA_ & !QB_ & !QD_ & !AMBS_ & !MRD_
           # RST_ & !AUX_ & PAUEN_ & !DTR_   "Bailout When Broken

enable CHEN_ = !AUX_

!CHEN_    = RST_ & !AGO_ & CDR

enable CHCK_ = !AUX_

!CHCK_   := RST_ & !QA_ & QB_ & QC_ & !QD_ & AMBS_ & WR
           # RST_ & !QA_ & QB_ & QC_ & !QD_ & !AMBS_ & !MWT_

!PAK_    := RST_ & !CDR & !QA_ & !QB_ & QC_ & !QD_ & !DTR_
           # RST_ & !CDR & !QA_ & !QB_ & QC_ & WR & AMBS_
           # RST_ & !CDR & !QA_ & !QB_ & QC_ & !MWT_ & !AMBS_
           # RST_ & CDR & QA_ & !QB_ & QC_ & PAK_ & DTR_
           # RST_ & CDR & QA_ & !QB_ & QC_ & !PAK_ & RD & AMBS_
           # RST_ & CDR & QA_ & !QB_ & QC_ & !PAK_ & !MRD_
               & !AMBS_

!QA_     := !AUX_ & !QA_ & !QB_
           # !AUX_ & !QA_ & !QD_
           # !AUX_ & !QA_ & !DTR_
           # !AUX_ & QB_ & QC_ & !QD_ & !AGO_ & !CDR

!QB_     := !AUX_ & !QB_ & !QD_
           # !AUX_ & !QC_ & !DTR_
           # !AUX_ & QA_ & !QB_ & RD & AMBS_
           # !AUX_ & QA_ & !QB_ & !MRD_ & !AMBS_
           # !AUX_ & !QA_ & !QB_ & WR & AMBS_
           # !AUX_ & !QA_ & !QB_ & !MWT_ & !AMBS_
           # !AUX_ & !QA_ & !QD_ & WR & AMBS_
           # !AUX_ & !QA_ & !QD_ & !MWT_ & !AMBS_
```

```
!QC_   := !AUX_ & QB_ & !QC_
        # !AUX_ & !QC_ & !RD & AMBS_
        # !AUX_ & !QC_ & MRD_ & !AMBS_
        # !AUX_ & QA_ & QB_ & !QD_ & !AGO_ & CDR

!QD_   := !AUX_ & !QC_
        # !AUX_ & !QA_ & QB_ & !QD_
        # !AUX_ & !QA_ & !QD_ & DTR_
        # !AUX_ & QA_ & !QB_ & !QD_ & !DTR_
        # !AUX_ & QA_ & QB_ & !AGO_
```

P16R4 Located at BA1

Title

Wait State Generator
SSEC Madison, WI 05/05/86

Identification

```
Device location  BA1
Device type      P16R4
```

Module

```
GPCI12
flag -r0
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin  10, 20

MCLK, SRA_, RFA_, MND_, IND_, NVCS_, DSP_, INT, ALE, OE_
        pin  1, 2, 3, 4, 5, 6, 7, 8, 9, 11

RDY1_
        pin  14

RDY_, INEN, INTA_
        pin  12, 13, 19
```

Equations

```
enable RDY_  = TRUE

!RDY_       = !MND_
            # !IND_
            # !INTA_
            # !RDY1_

!RDY1_    :=  !ALE & !NVCS_
            # !ALE & !DSP_
            # !ALE & !SRA_
            # !ALE & !RFA_
            # !ALE & INEN & INT
```

Notes

Memory No Delay (MND_) and Input/Output No Delay (IND_) access with 0 wait states.

The Non-Volatile Ram (NVCS_) and the Display (DSP_) access with 1 wait state.

The Shared Resources and the Register File delay until an acknowledge (SRA_ & RFA_ respectively) is received.

Interrupt Enable (INEN) allows a timer in the MUART to interrupt a wait forever condition due to a nonexistent resource.

P20L10 Located at AV1

Title

```
Memory Address Decode
SSEC Madison,  WI 07/14/86
```

Identification

```
Device location  AV1
Device type      P20L10
```

Module

```
GPCI13
flag -r0
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin  12, 24

AB9, ABA, ABB, ABC, ABD, ABE, ABF, MS0, MS1, IOM, RD, WR
        pin  1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13

MND_, SDDR_
        pin  14, 23

NVCS_, PMCS_, MDDR_, SPCS_, RFCS_, DPCS_, OSL_, OPF_
        pin  15, 16, 17, 18, 19, 20, 21, 22
```

Equations

```
enable PMCS_  = TRUE

!PMCS_      =  !IOM & !ABF & !ABE
             # !IOM & !ABF &  ABE & !ABD
             # !IOM & !ABF &  ABE &  ABD & !ABC & !ABB

enable RFCS_  = TRUE

!RFCS_      =  OPF_ & !IOM & RD & !ABF & ABE & ABD & !ABC
               & ABB & !ABA & !AB9
             # OPF_ & !IOM & WR & !ABF & ABE & ABD & !ABC
               & ABB & !ABA & !AB9

enable NVCS_  = TRUE
```

```
!NVCS_      = OPF_ & !IOM & RD & !ABF & ABE & ABD & !ABC
              & ABB & !ABA & AB9
            # OPF_ & !IOM & WR & !ABF & ABE & ABD & !ABC
              & ABB & !ABA & AB9

enable SPCS_ = TRUE

!SPCS_      = !IOM & RD & !ABF & ABE & ABD & !ABC
              & ABB & ABA
            # !IOM & WR & !ABF & ABE & ABD & !ABC
              & ABB & ABA

enable DPCS_ = TRUE

!DPCS_      =  OPF_ & !IOM & !ABF & ABE & ABD & ABC

enable MND_ = TRUE

!MND_       = !OPF_
            # !SPCS_
            # !PMCS_

enable OSL_ = TRUE

!OSL_       = OPF_ & ABF
            # OPF_ & IOM & !ABF & ABE

enable OPF_ = TRUE

!OPF_       = !IOM & MS0 & MS1

enable MDDR_ = TRUE

!MDDR_      = !PMCS_ &  RD
            # !SPCS_ &  RD
            # OPF_ & !IOM &  RD & !ABF &  ABE &  ABD & !ABC
              & ABB

enable SDDR_ = TRUE

!SDDR_      = !OSL_ & RD
            # OPF_ & !IOM & RD & !ABF & ABE & ABD & ABC
            # OPF_ & IOM & RD & !ABF & !ABE & !ABD & ABC
              & ABB
```

Memory Map			
	0000-67FF	Program	26K
	6800-69FF	Register File 32 x 8 (16 times)	512
	6A00-6BFF	Non-Volatile RAM 256 x 4 (2 times)	512
	6C00-6FFF	Scratch Pad RAM (STACK, etc.)	1K
	7000-7FFF	Dual-Port RAM	4K
	8000-FFFF	Offboard	32K

Notes

OPF_ ensures no opcode fetches except from onboard PROM and RAM.

P16L8 Located at AX1

Title

I/O ADDRESS DECODE
SSEC Madison, WI 05/05/86

Identification

Device location AX1
Device type P16L8

Module

```
GPCI14
flag -r0
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 10, 20

AB3, AB4, AB5, AB6, AB7, IOM, RD, WR
        pin 1, 2, 3, 4, 5, 6, 7, 8

IDDR_, IND
        pin 12, 19

DSP_, LLCS_, MUCS_, CHCS_, DMCS_, XCS_
        pin 13, 14, 15, 16, 17, 18
```

Equations

```
enable MUCS_ = TRUE

!MUCS_    = IOM & !AB7 & !AB6 & !AB5 & !AB4

enable DSP_ = TRUE

!DSP_     = IOM & !AB7 & !AB6 & !AB5 & AB4 & !AB3 & RD
          # IOM & !AB7 & !AB6 & !AB5 & AB4 & !AB3 & WR

enable CHCS_ = TRUE

!CHCS_    = IOM & !AB7 & !AB6 & !AB5 & AB4 & AB3

enable DMCS_ = TRUE

!DMCS_    = IOM & !AB7 & !AB6 & AB5 & !AB4
```

```
enable XCS_ = TRUE

!XCS_       = IOM & !AB7 & !AB6 & AB5 & AB4 & !AB3 & WR

enable LLCS_ = TRUE

!LLCS_      =  IOM & !AB7 & !AB6 & AB5 & AB4 & AB3 & WR

enable IND = TRUE

!IND        =  !MUCS_
             # !DMCS_
             # IOM & !AB7 & !AB6 & AB5 & AB4

enable IDDR_ = TRUE

!IDDR_      =  IOM & !AB7 & !AB6 & !AB5 & AB4 & !AB3 & RD
```

Notes

```
MUCS_   =  00   0F
DSP_    =  10   17
CHCS_   =  18   1F
DMCS_   =  20   2F
XCS_    =  30   37
LLCS_   =  38   3F

(OFFBOARD  =  40  FF  asserted in PAL13)
```

P22V10 Located at AJ41

Title

DMA Counter Decode
SSEC Madison, WI 05/05/86

Identification

Device location AJ41
Device type P22V10

Module

GPCI15
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin  12, 24

CLK, uPC, AB0, AB1, AB2, RD, WR, DMA_, CMD_, DNE, DMI_, PAUN_
        pin  1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13

LI2_, LI1_, LI0_, ACI_, WCI_, HI2_, HI1_, HI0_
        pin 14, 15, 16, 17, 18, 19, 20, 21

QB_, LDONE
        pin  22, 23

QA_ = WCI_

OUT = [!QA_, !QB_]
```

Equations

```
!QA_      := !PAUN_ & !QA_ & QB_
           # !PAUN_ &  QA_ & !QB_ & !LDONE
           # !PAUN_ &  QB_ & !DMI_ &  uPC

!QB_      := !PAUN_ & !QA_ & QB_
           # !PAUN_ &  QB_ & !DMI_ & !uPC
           # !PAUN_ & !QB_ &  LDONE

!ACI_     := !PAUN_ & !QA_ & QB_ & CMD_
           # !PAUN_ &  QA_ & !QB_ & !LDONE & CMD_
           # !PAUN_ &  QB_ & !DMI_ &  uPC & CMD_

enable LDONE  = !PAUN_
```

```
LDONE    = !QA_  &  QB_  &  DNE & !uPC
           # LDONE & !QA_
           # LDONE & !QB_

!LI0_    = !DMA_  &  WR & !AB2 & !AB1 & !AB0
           # !DMA_  &  WR & !AB2 & !AB1 &  AB0
           # !DMA_  &  RD & !AB2 & !AB1 &  AB0
           # !DMA_  &  WR & !AB2 &  AB1 &  AB0

!LI1_    = !DMA_  &  WR & !AB2 & !AB1 & !AB0
           # !DMA_  &  RD & !AB2 & !AB1 & !AB0
           # !DMA_  &  WR & !AB2 &  AB1 & !AB0
           # !DMA_  &  WR & !AB2 &  AB1 &  AB0

!LI2_    = !DMA_  &  WR & !AB2 & !AB1 & !AB0
           # !DMA_  &  RD & !AB2 & !AB1 & !AB0
           # !DMA_  &  RD & !AB2 & !AB1 &  AB0
           # !DMA_  &  RD & !AB2 &  AB1 & !AB0

!HI0_    = !DMA_  &  WR & !AB2 &  AB1 &  AB0
           # !DMA_  &  WR &  AB2 & !AB1 & !AB0
           # !DMA_  &  WR &  AB2 & !AB1 &  AB0
           # !DMA_  &  RD &  AB2 & !AB1 &  AB0

!HI1_    = !DMA_  &  WR & !AB2 &  AB1 &  AB0
           # !DMA_  &  WR &  AB2 & !AB1 & !AB0
           # !DMA_  &  RD &  AB2 & !AB1 & !AB0
           # !DMA_  &  WR &  AB2 &  AB1 & !AB0

!HI2_    = !DMA_  &  WR &  AB2 & !AB1 & !AB0
           # !DMA_  &  RD &  AB2 & !AB1 & !AB0
           # !DMA_  &  RD &  AB2 & !AB1 &  AB0
           # !DMA_  &  RD &  AB2 &  AB1 & !AB0
```

P22V10 Located at AX28

Title

MBUS Address Generator
SSEC Madison, WI 05/05/86

Identification

Device location AX28
Device type P22V10

Module

GPCI16
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 12, 24

XIOCS_, DB0, DB1, DB2, DB3, DB4, CMD3_, RST_
        pin 1, 2, 3, 4, 5, 6, 7, 8

SGO, BDEN_, AB0
        pin 10, 11, 13

PAEN_, SLCK_, DLCK_, RFIA, CTL2
        pin 14, 15, 16, 17, 18

ADRF_, ADR10_, ADR11_, ADR12_, ADR13_
        pin 19, 20, 21, 22, 23

reset  node 25

CLK, SG, BE, R_      =   XIOCS_, SGO, BDEN_, RST_
Q0, Q1, Q2, Q3, Q4   =   PAEN_, SLCK_, DLCK_, RFIA, CTL2
A0, A1, A2, A3, A4   =   ADRF_, ADR10_, ADR11_, ADR12_, ADR13_
```

Equations

```
reset = !RST_

enable  ADRF_  = !BDEN_ & SGO

ADRF_ := AB0 & ADRF_    SEE NOTE
          # !DB0 & DB1 & ADRF_
          # !DB0 & DB2 & ADRF_
          # !DB0 & DB3 & ADRF_
          # !DB0 & DB4 & ADRF_
          # !DB0 & DB1 & !AB0
          # !DB0 & DB2 & !AB0
          # !DB0 & DB3 & !AB0
          # !DB0 & DB4 & !AB0

enable  ADR10_  = !BDEN_ & SGO

!ADR10_ := !AB0 & DB1
          # AB0 & !ADR10_

enable  ADR11_  = !BDEN_ & SGO

!ADR11_ := !AB0 & DB2
          # AB0 & !ADR11_

enable  ADR12_  = !BDEN_ & SGO

!ADR12_ := !AB0 & DB3
          # AB0 & !ADR12_

enable  ADR13_  = !BDEN_ & SGO

!ADR13_ := !AB0 & DB4
          # AB0 & !ADR13_

!PAEN_  := AB0 & DB0 & !CMD3_
          # !AB0 & !PAEN_

!SLCK_  := AB0 & DB1
          # !AB0 & !SLCK_

!DLCK_  := AB0 & DB2
          # !AB0 & !DLCK_

RFIA    := AB0 & DB3
          # !AB0 & CTL2

CTL2    := AB0 & DB4
          #!AB0 & RFIA
```

Notes

ADRF_ is active low. It is created using De Morgan's theorem to invert the equation shown below. This is done so that RESET asserts the bit.

```
ADRF     := !AB0 & DB0
            # !AB0 & !DB1 & !DB2 & !DB3 & !DB4
            # AB1 & ADRF
```

ADRF_ is substituted for !ADRF after the inversion.

P16R4 Located BA14

Title

Shared Resource Arbiter
SSEC Madison, WI 02/24/87

Identification

```
Device location  BA14
Device type      P16R4
```

Module

```
GPCI17
flag -r0
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 10, 20

C2X, ADRC_, AUM_, DRQ_, AUX_, NC2_, INIT_, SRRQ_, VSRQ_, OE_
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 11

REQ_, AQC_, AQB_, AQA_
        pin 14, 15, 16, 17

CHPS_, DSGO_, AGO_, DGO_
        pin 12, 13, 18, 19

C_      = CHPS_
AST     = [AQA_, AQB_, AQC_ ]
O_, D_  = OE_, DSGO_
```

State Definitions

```
a = ^b111     b = ^b100     c = ^b110     d = ^b101
e = ^b000     f = ^b011     g = ^b001     h = ^b010
```

Equations

```
!AQA_    := INIT_ & AQB_ & AQC_ & !REQ_ & !VSRQ_
            # INIT_ & !AQA_ & AQC_
            # INIT_ & !AQA_ & !VSRQ_
```

```
!AQB_     := INIT_ & !AQA_ & AQB_ & AQC_
             # INIT_ & AQB_ & AQC_ & REQ_ & !DRQ_
             # INIT_ & AQA_ & !AQB_ & AQC_ & !DRQ_
             # INIT_ & AQA_ & AQB_ & !AQC_ & AUX_ & !SRRQ_
             # INIT_ & AQA_ & AQB_ & !AQC_ & !AUM_ & !SRRQ_
             # INIT_ & AQA_ & AQB_ & !AQC_ & CHPS_ & !SRRQ_
             # INIT_ & AQA_ & !AQB_ & !AQC_ & !SRRQ_
             # INIT_ & !AQA_ & !AQB_ & !AQC_ & !VSRQ_

!AQC_     := INIT_ & AQA_ & AQB_ & AQC_ & !REQ_ & VSRQ_
             # INIT_ & !AQA_ & !AQB_ & AQC_
             # INIT_ & !AQA_ & AQC_ & ADRC_
             # INIT_ & AQA_ & !AQC_ & !SRRQ_
             # INIT_ & !AQA_ & !AQC_ & !VSRQ_

!REQ_     := INIT_ & AQA_ & AQB_ & AQC_ & !SRRQ_
             # INIT_ & AQA_ & !AQB_ & AQC_ & !SRRQ_
             # INIT_ & !AQA_ & !SRRQ_
             # INIT_ & AQA_ & !VSRQ_

enable DGO_ = TRUE

!DGO_      =  AQA_ & !AQB_ & AQC_ & !DRQ_

enable AGO_ = TRUE

!AGO_      = !AQA_ & !AQB_ & !AQC_ & !AUM_
             # AQA_ & AQB_ & !AQC_ & !AUX_ & AUM_ & !CHPS_

enable DSGO_ = TRUE

!DSGO_     = !AQA_ & !AQB_ & AQC_ & !ADRC_
             # !DSGO_ & !VSRQ_
```

P20L8 Located at AX14

Title

Shared Resource Glue
SSEC Madison, WI 02/19/87

Identification

Device location AX14
Device type P20L8

Module

GPCI18
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 12, 24

SGO_, AQB_, AQC_, AGO_, ABUS_, OSL_, PACK_, XACK_
        pin 1, 2, 3, 4, 5, 6, 7, 8

GBUS_, DPCS_, CHPS_, ADC_, DLK_, SREQ_
        pin 9, 10, 11, 13, 14, 23

SREN_, SRAK_
        pin 15, 22

AEN_, MWT_, MRD_, VSRQ_, SRRQ_, PXAK_
        pin 16, 17, 18, 19, 20, 21

AQA_ = SGO_
```

Equations

```
enable SRRQ_ = TRUE

!SRRQ_    = !DPCS_
          # !CHPS_
          # !OSL_ & !AEN_ & !GBUS_

enable VSRQ_ = TRUE

!VSRQ_    = !SREQ_ & AEN_ & !MWT_ & !ADC_ & DLK_
          # !SREQ_ & AEN_ & !MRD_ & !ADC_ & DLK_
          # !SREQ_ & AEN_ & !MWT_ & ADC_ & DLK_ & !ABUS_
          # !SREQ_ & AEN_ & !MRD_ & ADC_ & DLK_ & !ABUS_
          # !SREQ_ & AEN_ & !MWT_ & ADC_ & !DLK_
          # !SREQ_ & AEN_ & !MRD_ & ADC_ & !DLK_
```

```
enable SREN_ = TRUE

!SREN_     = AQA_ & !AQC_ & !SRRQ_

enable SRAK_ = TRUE

!SRAK_     = !DPCS_ & AQA_ & !AQC_
           # !OSL_ & AQA_ & !AQC_ & !AEN_ & !GBUS_ & !XACK_
           # !CHPS_ & AQA_ & !AQC_ & AGO_
           # !CHPS_ & AQA_ & AQB_ & !AQC_ & !AGO_ & !PACK_ &
                 ABUS_

enable PXAK_ = TRUE

!PXAK_     = !ADC_ & !VSRQ_ & !AQA_ & AQB_ & !AQC_
           # ADC_ & !VSRQ_ & !AQA_ & !AQB_ & !AQC_ & !ABUS_ &
                 !PACK_
           # ADC_ & !VSRQ_ & !AQA_ & !AQB_ & !AQC_ & ABUS_
```

Notes

DLK (Dual-Port Lock) has two functions. One is to lock out the Multibus from accessing the Dual-Port RAM. The other is to enable an automatic XACK for each access to the Channel Port which is memory mapped in the Multibus domain.

When DLK is false, the Multibus has access to the Dual-Port RAM and will receive an XACK when the channel data transfer is complete.

When DLK is true, the Multibus cannot read or write to the Dual-Port RAM. If an ABUS (Automatic Multibus) is not in progress, it will receive an XACK as soon as it accesses the memory space of the Channel Port.

P16L8 Located at AG43

Title

Bus Arbiter Controller
SSEC Madison, WI 08/06/86

Identification

Device location AG43
Device type P16L8

Module

GPCI19
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 10, 20

RD, ALE, DCT_, OSL_, AUS_, CDR, AMF, MWT_, AEN_, SGO_
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 11

BDR_, BDEN_
        pin 12, 19

DGO_, LDR_, NC0, BS0_, NC1, CMD0_
        pin 13, 14, 15, 16, 17, 18
```

Equations

```
enable BDR_ = TRUE

!BDR_       = !SGO_ & !MWT_
            # !AEN_ & RD & !OSL_
            # !DGO_ & !DCT_ & !CDR & AMF

enable BDEN_ = TRUE

!BDEN_      = !AEN_
            # !SGO_

enable BS0_ = TRUE

!BS0_       = DGO_ & !ALE & !OSL_
            # !DGO_ & !DCT_ & AEN_ & CMD0_ & AMF
            # !DGO_ & !DCT_ & AEN_ & !CMD0_
```

```
enable LDR_ = TRUE

!LDR_       = !CDR & !AUS_
              # !LDR_ & !DCT_
```

Notes

This is not a true emulation of /S0, /S1, /S2. BS0_ is used solely to make the 8289 operate correctly.

LDR_ latches the direction to allow the DMA controller to complete a cycle when the channel stops the data transfer (CDR =L = IBM READ).

P20L8 Located at AG28

Title

```
Control Signal Generator
SSEC Madison,  WI 08/06/86
```

Identification

```
Device location   AG28
Device type       P20L8
```

Module

```
GPCI20
flag -r0
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin  12, 24

MWT_, IOM, WR, DCT_, AMF, CDR, RD, CMD0_, DGO_, SGO_, ADC_,
DPCS_, CHLS_, AUS_
        pin  1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 23

DPWR_, CHEN_
        pin  15, 22

DPCE_, CHCK_, PIOR_, PIOW_, PMRD_, PMWT_
        pin  16, 17, 18, 19, 20, 21
```

Equations

```
enable PMWT_  = TRUE

!PMWT_     =  DGO_ & SGO_ & !IOM &  WR &  AUS_
             # !DGO_ & CMD0_ & CDR & !DCT_

enable PMRD_  = TRUE

!PMRD_     =  DGO_ & SGO_ & !IOM &  RD &  AUS_
             # !DGO_ & CMD0_ & !CDR & !DCT_

enable PIOW_  = TRUE

!PIOW_     =  DGO_ & SGO_ & IOM &  WR &  AUS_
             # !DGO_ & !CMD0_ & CDR & !DCT_
```

```
enable PIOR_ = TRUE

!PIOR_     = DGO_ & SGO_ & IOM &  RD &  AUS_
             # !DGO_ & !CMD0_ & !CDR & !DCT_

enable DPCE_ = TRUE

!DPCE_     = DGO_ & SGO_ & !DPCS_ & RD
             # DGO_ & SGO_ & !DPCS_ & WR
             # !DGO_ & CMD0_ & !DCT_ & !AMF
             # !SGO_ & !ADC_

enable DPWR_ = TRUE

!DPWR_     = DGO_ & SGO_ & !DPCE_ & WR
             # !DGO_ & !DPCE_ & CDR & !DCT_
             # !SGO_ & !ADC_ & !MWT_

enable CHEN_ = AUS_

!CHEN_     = !CHLS_ & RD

enable CHCK_ = AUS_

!CHCK_     = !CHLS_ & WR
```

P16R4 Located at BX13

Title

Immediate Tester
SSEC Madison, WI 05/05/86

Identification

Device location BX13
Device type P16R4

Module

GPCI21
flag -r0

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 10, 20

CLK, AD4, AD5, OD0, OD1, OD2, SHT_, SUT_, OPT_, OE
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 11

ADEN_, SUEN_, SHEN_, RST6_
        pin 14, 15, 16, 17

ADT, INTA, RIM_, OTCS_
        pin 12, 13, 18, 19
```

Equations

```
enable RIM_ = TRUE

!RIM_       = OPT_
            # !SUT_ & !SUEN_
            # !SHT_ & !SHEN_
            # SHT_ & SHEN_
            # ADT & !ADEN_

!SHEN_   :=  OD0 & !OTCS_ & !AD4 & AD5
            # !SHEN_ & OTCS_
            # !SHEN_ & AD4
            # !SHEN_ & !AD5

!SUEN_   :=  OD1 & !OTCS_ & !AD4 & AD5
            # !SUEN_ & OTCS_
            # !SUEN_ & AD4
            # !SUEN_ & !AD5
```

```
!ADEN_   := OD2 & !OTCS_ & !AD4 & AD5
            # !ADEN_ & OTCS_
            # !ADEN_ & AD4
            # !ADEN_ & !AD5

!RST6_   := !INTA & !OTCS_ & AD4 & AD5
            # !INTA & !RST6_
```

Notes

Respond Immediately (RIM_) is a signal which allows the Bit-Slice to determine in one test if any of several events have happened. The three enables are used to condition their respective signals.

RST6_ interrupts the processor when the Bit-Slice wants it to look in the Register File.

P16L8 Located at BX26

Title

Alternate Register File Driver
SSEC Madison, WI 07/25/88

Identification

Device location BX26
Device type P16L8

Module

GPCI22
flag -r0

Declarations

TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
pin 10, 20

BRFRD_, BRFWR, ODAT0, ODAT1, ODAT2, ODAT3
pin 1, 2, 3, 4, 5, 6

IDAT3, IDAT2, IDAT1, IDAT0
pin 19, 12, 13, 14

DR3_, DR2_, DR1_, DR0_
pin 15, 16, 17, 18

Equations

enable DR0_ = BRFWR

!DR0_ = ODAT0

enable DR1_ = BRFWR

!DR1_ = ODAT1

enable DR2_ = BRFWR

!DR2_ = ODAT2

enable DR3_ = BRFWR

!DR3_ = ODAT3

enable IDAT0 = !BRFRD_

IDAT0 = !DR0_

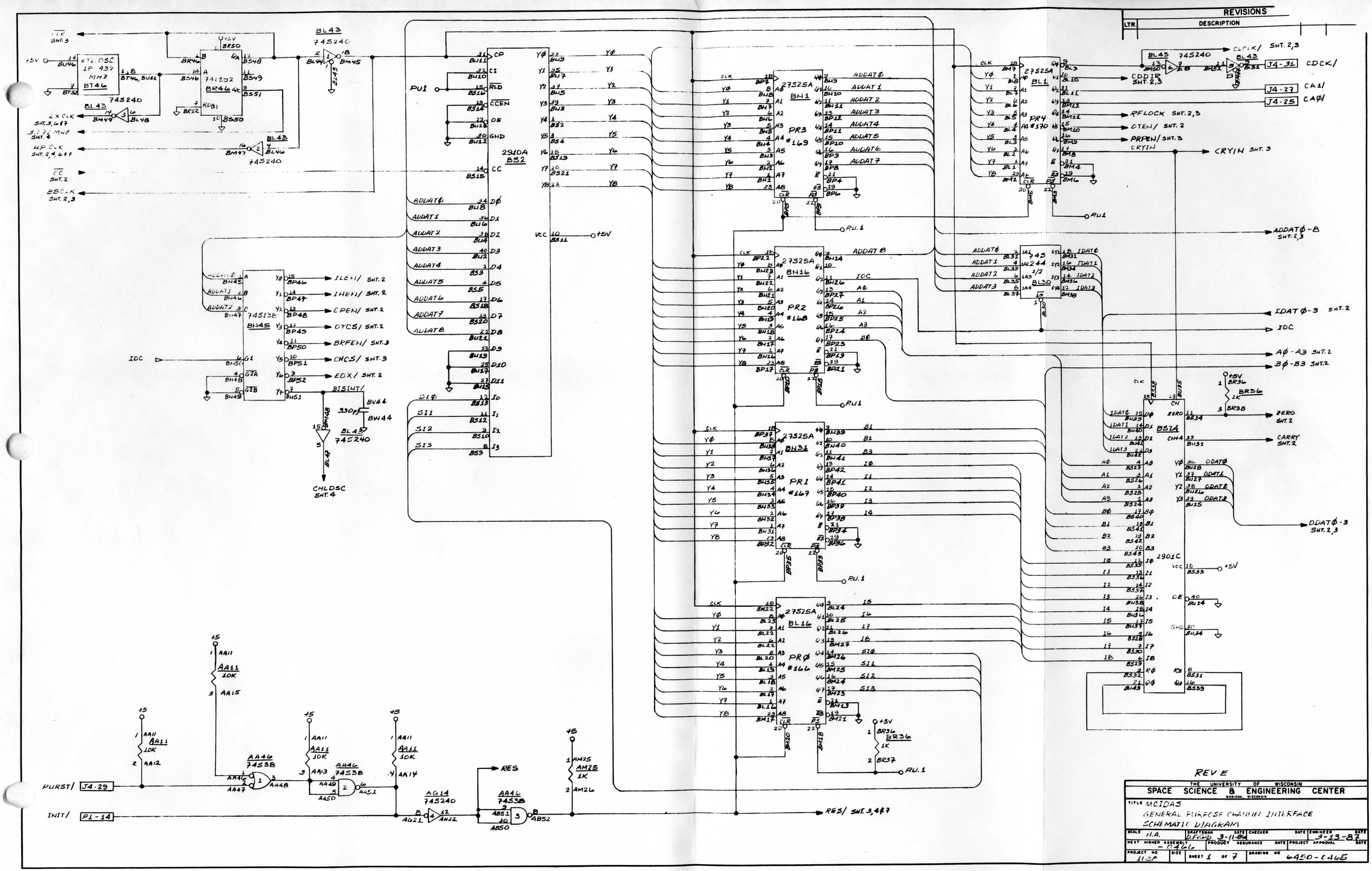
```
enable IDAT1 = !BRFRD_

IDAT1  = !DR1_

enable IDAT2 = !BRFRD_

IDAT2  = !DR2_

enable IDAT3 = !BRFRD_

IDAT3  = !DR3_
```

Note

This part is to be used only with GPCI Rev. F. Use only this part or the 2926, not both.

REVISIONS

LTR.	DESCRIPTION



REV E

THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

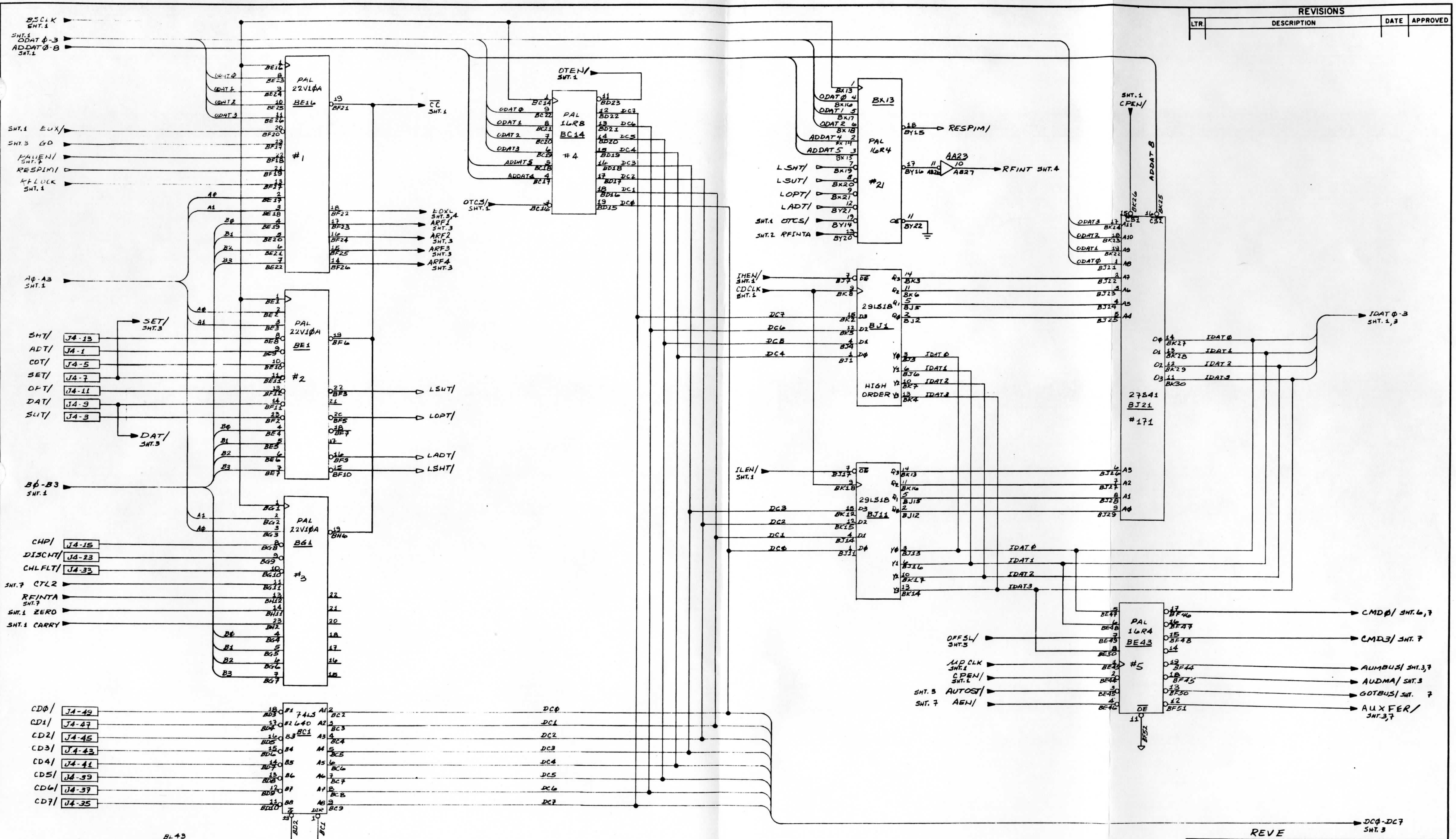
TITLE MCIDAS
GENERAL PURPOSE CHANNEL INTERFACE
SCHEMATIC DIAGRAM

SCALE	DRAFTSMAN	DATE	CHECKER	DATE	ENGINEER	DATE
N.A.	D.FORD	3-11-86				3-13-87

NEXT HIGHER ASSEMBLY	PRODUCT ASSURANCE	DATE	PROJECT APPROVAL	DATE
-0466				

PROJECT NO	SIZE	SHEET	DRAWING NO
113F		1 OF 7	6450-0465

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

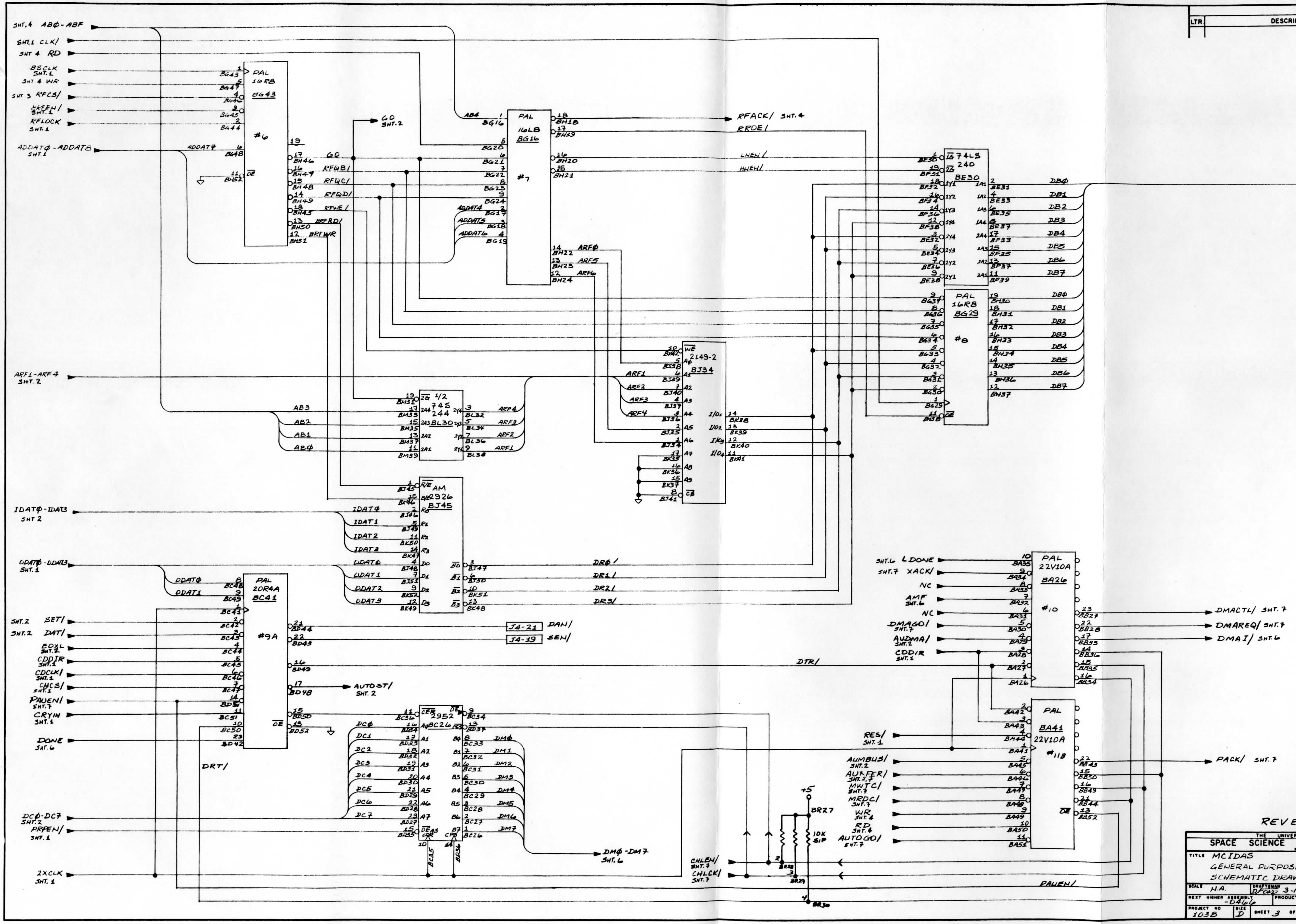


REVE

THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

TITLE MCIDAS GENERAL PURPOSE CHANNEL INTERFACE SCHEMATIC DIAGRAM					
SCALE N.A.	DRAFTSMAN DFORD	DATE 3-11-86	CHECKER	DATE	ENGINEER 3-13-87 DATE
NEXT HIGHER ASSEMBLY -0466	PRODUCT ASSURANCE	DATE	PROJECT APPROVAL	DATE	
PROJECT NO 1038	SIZE D	SHEET 2 OF 7	DRAWING NO 6450-0465		

REVISIONS			
LTR.	DESCRIPTION	DATE	APPROVED

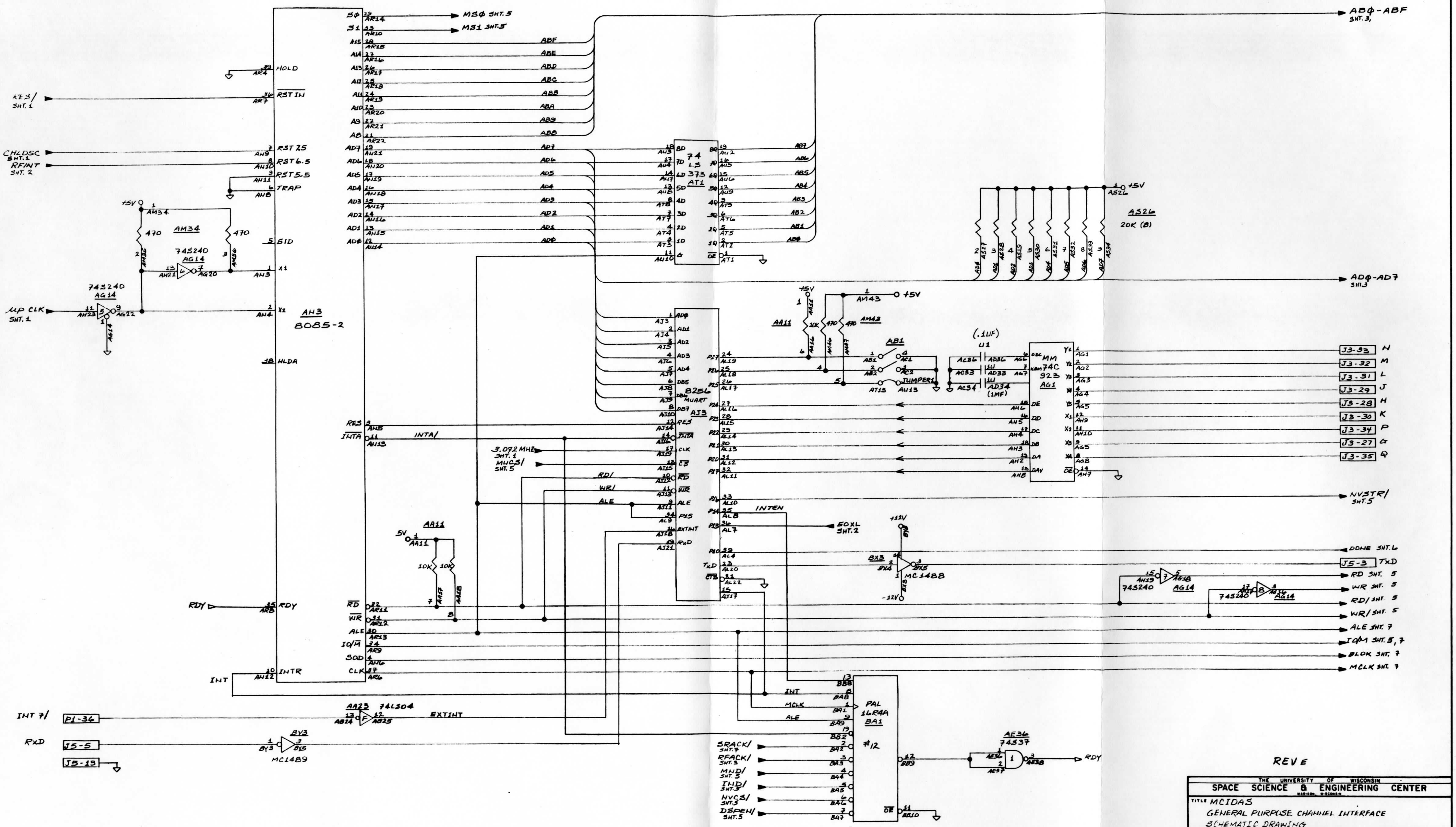


REVE

THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

TITLE MCIDAS GENERAL PURPOSE CHANNEL INTERFACE SCHEMATIC DRAWING			
SCALE N.A.	DRAFTSMAN D.FORD DATE 3-11-86	CHECKER DATE	ENGINEER DATE 3-13-87
NEXT HIGHER ASSEMBLY -0466	PRODUCT ASSURANCE DATE	PROJECT APPROVAL DATE	
PROJECT NO 1038	SIZE D	SHEET 3 OF 7	DRAWING NO 6450-0465

REVISIONS			
LTR.	DESCRIPTION	DATE	APPROVED



THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

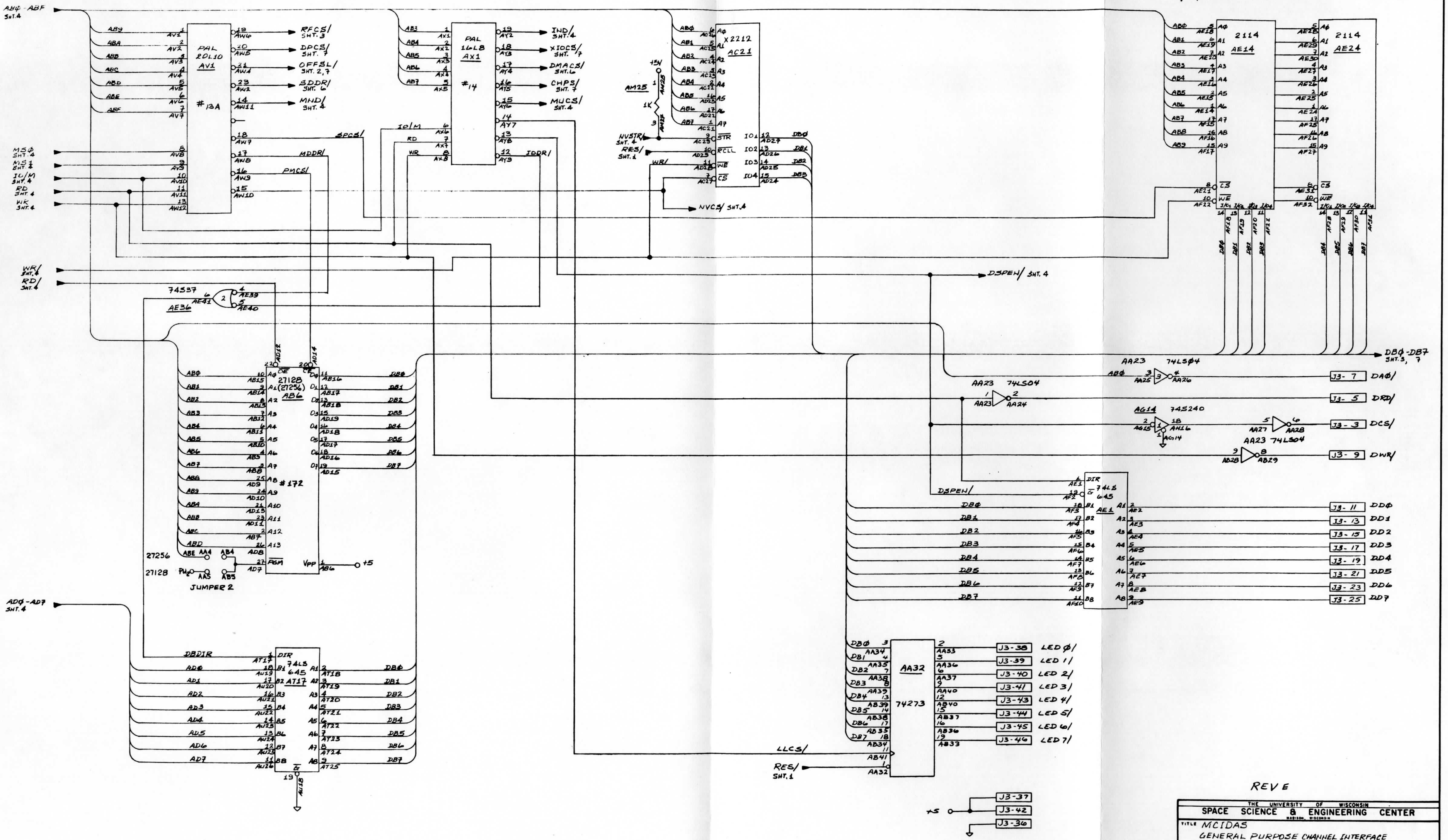
TITLE MCIDAS
GENERAL PURPOSE CHANNEL INTERFACE
SCHEMATIC DRAWING

SCALE	DRAFTSMAN	DATE	CHECKER	DATE	ENGINEER	DATE
N.A.	D. FORD	3-11-86				3-13-87

NEXT HIGHER ASSEMBLY	PRODUCT ASSURANCE	DATE	PROJECT APPROVAL	DATE
-0466				

PROJECT NO	SIZE	SHEET	OF	DRAWING NO
1038	D	4	7	6450-0465

REVISIONS			
LTR.	DESCRIPTION	DATE	APPROVED



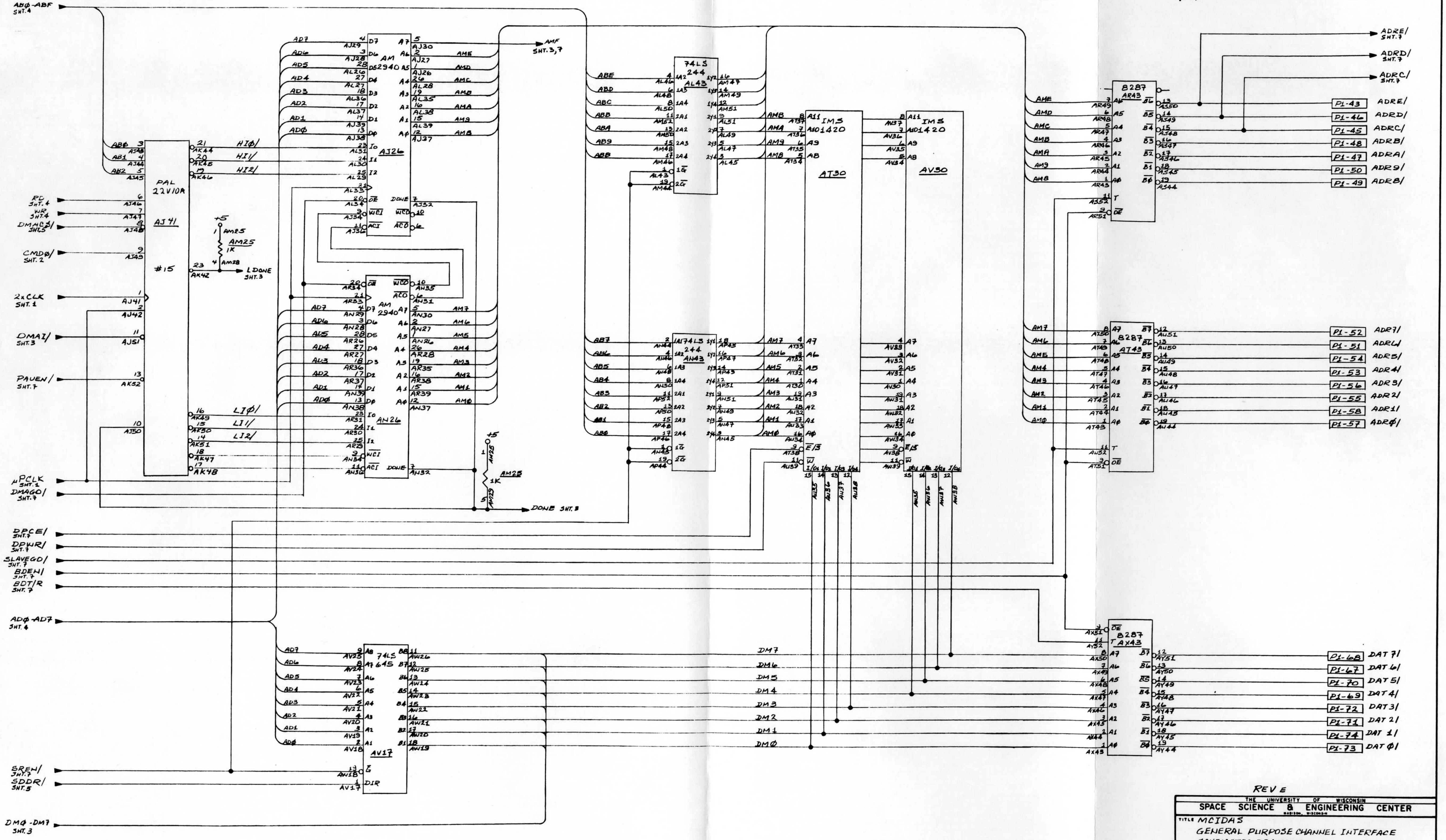
REV E

THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

TITLE MCIDAS
GENERAL PURPOSE CHANNEL INTERFACE
SCHEMATIC DRAWING

SCALE NA	DRAFTSMAN D.FORD DATE 3-11-86	CHECKER DATE	ENGINEER DATE 3-13-87
NEXT HIGHER ASSEMBLY -0466	PRODUCT ASSURANCE DATE	PROJECT APPROVAL DATE	
PROJECT NO 1036	SIZE D	SHEET 5 OF 7	DRAWING NO 6450-0465

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED



REV E

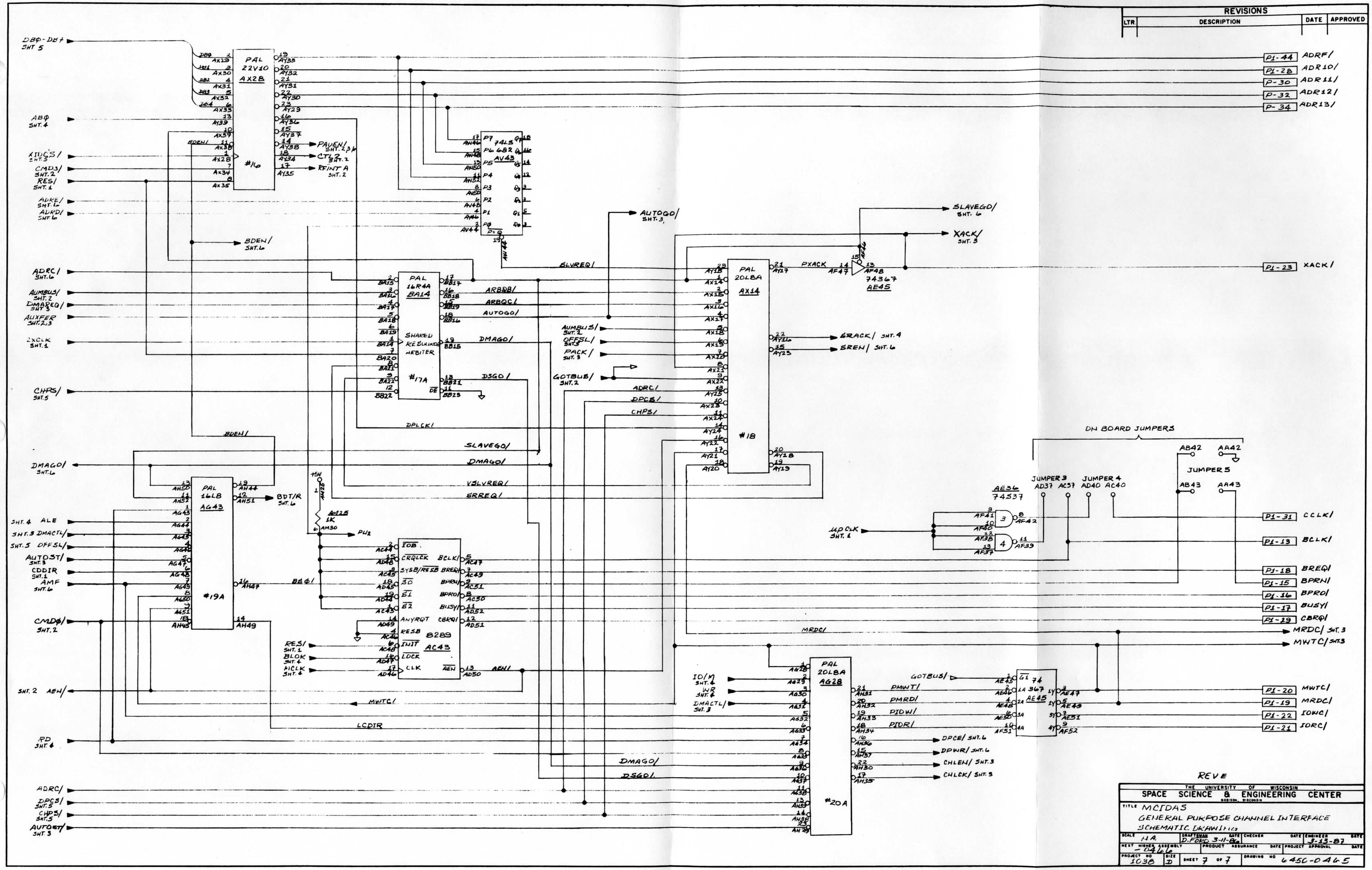
THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

TITLE MCIDAS
GENERAL PURPOSE CHANNEL INTERFACE
SCHEMATIC DRAWING

SCALE	DRAFTSMAN	DATE	CHECKER	DATE	ENGINEER	DATE
N.A.	D.FORD	3-11-86			3-13-87	

NEXT HIGHER ASSEMBLY	PRODUCT ASSURANCE	DATE	PROJECT APPROVAL	DATE
-0462				

PROJECT NO	SIZE	SHEET	DRAWING NO
1038	D	6 OF 7	6450-0465



THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

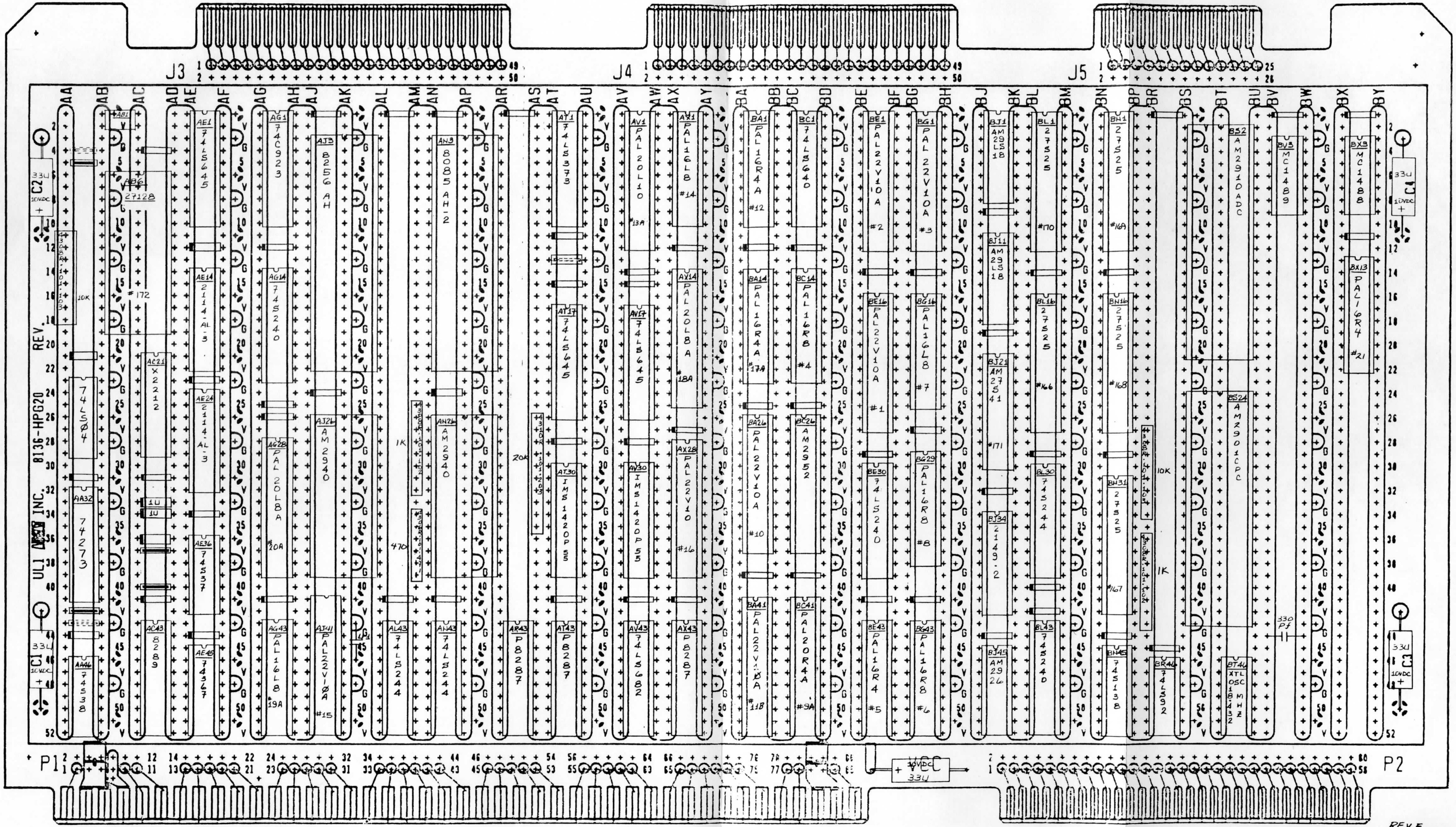
TITLE MCIDAS
GENERAL PURPOSE CHANNEL INTERFACE
SCHEMATIC DRAWING

SCALE	DRAFTSMAN	DATE	CHECKER	DATE	ENGINEER	DATE
N.A.	D.FORD	3-11-86			3-13-87	

NEXT HIGHER ASSEMBLY	PRODUCT ASSURANCE	DATE	PROJECT APPROVAL	DATE
-0466				

PROJECT NO	SIZE	SHEET	DRAWING NO
1038	D	7 OF 7	6450-0465

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED



REVE

NOTE: JUMPERS

AA42 - AB42	BPRN/ GRND	P1-15
AC37 - AD37	BCLK/ TO MULTIBUS	P1-13
AC40 - AD40	CCLK/ TO MULTIBUS	P1-31
AA43 - AB43	BPRN/ TO MULTIBUS	P1-15

THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

TITLE MCIDAS
GENERAL PURPOSE CHANNEL INTERFACE
ASSEMBLY DRAWING

SCALE N.A.	DRAFTSMAN D.FORD DATE 3-11-86	CHECKER DATE	ENGINEER DATE 3-13-87
NEXT HIGHER ASSEMBLY -0465	PRODUCT ASSURANCE DATE	PROJECT APPROVAL DATE	
PROJECT NO 1038	SIZE D	SHEET 1 OF 1	DRAWING NO 6450-0466

Channel Drivers
and Receivers

Channel Drivers and Receivers

The Channel Drivers and Receivers (CDR) board is an extension of the Multisourcerer's General Purpose Channel Interface (GPCI) board. It provides:

- the channel compatible signal drivers and receivers
- parity generation and checking
- the relay logic necessary to meet the requirements of the IBM I/O interface

Together, the CDR and GPCI function as a custom designed control unit that communicates with the IBM channel. The CDR is a printed circuit board mounted on the left side panel at the rear of the Multisourcerer chassis. The majority of the components on the CDR board are line drivers and line receivers, hence the name.

Since the CDR is a link between the GPCI and the I/O interface, an understanding of the I/O interface and the I/O requirements of the GPCI is necessary in understanding the CDR's function. IBM's document, *IBM System 360 and System 370 I/O Interface Channel to Control Unit Original Equipment Manufacturers' Information* (Seventh Edition dated March, 1983), is a valuable reference describing the I/O interface.

This document provides some background information on the IBM System 370 Interface. This information is distinguished by indented margins, as shown in the three paragraphs below.

> The I/O interface is a parallel interface consisting of an input data bus (Bus In), an output data bus (Bus Out), an input group of control signals, an output group of control signals, Bus In identification signals (Tag In), and Bus Out identification signals (Tag Out). The I/O interface generally uses fully interlocked handshaking, where the change of a control signal must be acknowledged by the recipient before the signal is allowed to change again.
>
> Each control unit connected to an IBM I/O channel has a unique I/O address or block of I/O addresses assigned to it. There may be several control units parallel connected to the same IBM I/O channel. All interface lines (described later) are parallel connected to each control unit with the exception of a control signal called Select Out. Select Out originates at the Channel, and is daisy chained from one control unit to the next. The Select Out from the last control unit connects to the channel's Select In pin.

When the channel wants to communicate with a particular control unit (i.e., GPCI and CDR) it sends the desired control unit's address to all control units and raises Address Out, Hold Out and Select Out. Control units that do not detect their assigned address pass the input Select Out logic level to the next control unit via its output Select Out pin. If a control unit detects its own assigned I/O address, it sends a low to the Select Out input pin of the next control unit. The channel recognizes the control unit's response when its Select In signal remains low while its Select Out signal is high.

CDR Functional Description

Figure 18 on the next page is a functional block diagram of the Channel Drivers and Receivers board. The CDR board is divided into three sections:

- Data Handling Section
- Interface Control Section
- Board Control Section

Data Handling Section

Data transmission between the CDR and the channel is accomplished over separate input and output buses. The CDR receives data from the channel on the Bus Out lines and transmits data to the channel on the Bus In lines. Data transmission between the CDR and the GPCI is via a bidirectional data bus (CD0-CD7). An internal bidirectional bus (CB0-CB7) transports data to the designated destination on the CDR.

Incoming data (eight bits and parity) arrives from the channel on the Bus Out lines. This data is received by the Channel Data Line Receivers and passed to the Octal Tri-State Buffer (except parity). Parity is passed to the Parity Checker/Generator via the Board Control block. When the Octal Tri-State Buffer is enabled, it passes the data to the Bus Transceiver. The Bus Transceiver either passes incoming data from the channel to the GPCI or passes the output data from the GPCI to the channel. CDDIR (Channel Data Direction) controls the direction of data flow through the Bus Transceiver.

Outgoing data (from the GPCI to the channel) passes through the Bus Transceiver to the Output Data Latch. The Output Data Latch passes its data to the Channel Data Line Drivers which drive the Bus In lines.

The Parity Checker/Generator is used during both data transmission and data reception. During data reception from the channel, the eight data lines and the parity line are applied to the Parity Checker/Generator. The data bits are applied to the Parity Checker/Generator via the internal data bus (CB0 - CB7). BOP/, the active low Bus Out Parity bit (from the Channel Data Line Receivers), is applied to the Board Control logic. The Board Control inverts and gates the parity bit to the Parity Checker/Generator only during data reception. When receiving data from the channel, the parity check output signal is applied to the GPCI as a status signal. This signal is high as long as the parity remains odd.

During GPCI data transmission, GPCI output data is applied to the Parity Checker/ Generator via the Bus Transceiver and the Board Control holds GPAR (Gated Parity) low. The Parity Checker/Generator then computes the parity bit, based on the eight data bits. The parity output, Bus Out Parity (BOPAR), is applied to one of the Output Data Latches.

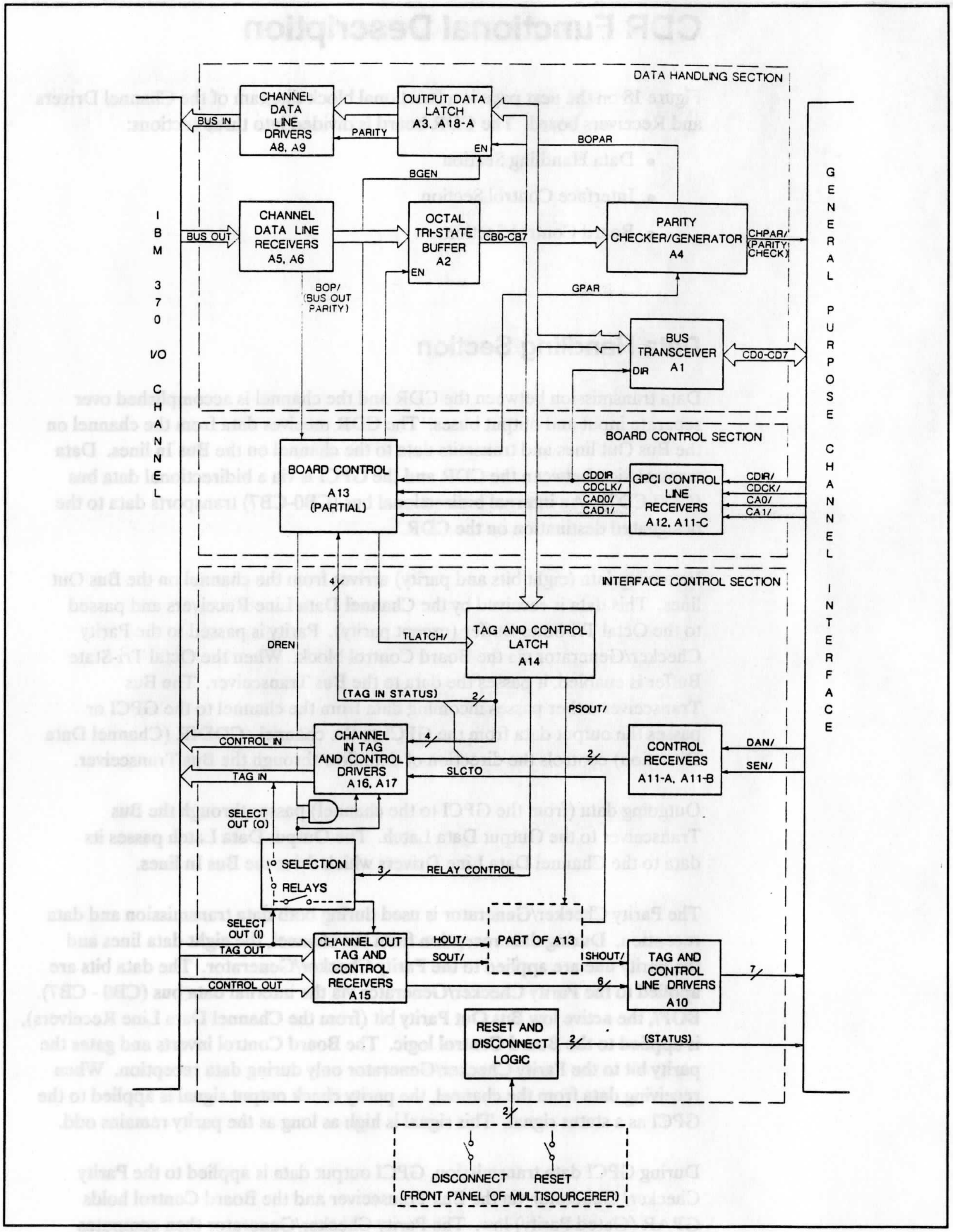

DATA HANDLING SECTION
CHANNEL DATA LINE DRIVERS A8, A9
BUS IN
OUTPUT DATA LATCH A3, A18-A
PARITY
EN
BOPAR
BGEN
CHANNEL DATA LINE RECEIVERS A5, A6
BUS OUT
OCTAL TRI-STATE BUFFER A2
EN
CB0-CB7
PARITY CHECKER/GENERATOR A4
CHPAR/ (PARITY CHECK)
BOP/ (BUS OUT PARITY)
GPAR
BUS TRANSCEIVER A1
DIR
CD0-CD7
IBM 370 I/O CHANNEL
GENERAL PURPOSE CHANNEL INTERFACE
BOARD CONTROL SECTION
BOARD CONTROL A13 (PARTIAL)
CDDIR
CDCLK/
CAD0/
CAD1/
GPCI CONTROL LINE RECEIVERS A12, A11-C
CDIR/
CDCK/
CA0/
CA1/
INTERFACE CONTROL SECTION
DREN
TLATCH/
TAG AND CONTROL LATCH A14
(TAG IN STATUS)
2
PSOUT/
CONTROL IN
TAG IN
CHANNEL IN TAG AND CONTROL DRIVERS A16, A17
4
SLCTO
2
CONTROL RECEIVERS A11-A, A11-B
DAN/
SEN/
SELECT OUT (O)
SELECTION RELAYS
3
RELAY CONTROL
SELECT OUT (I)
TAG OUT
CONTROL OUT
CHANNEL OUT TAG AND CONTROL RECEIVERS A15
HOUT/
SOUT/
PART OF A13
SHOUT/
6
TAG AND CONTROL LINE DRIVERS A10
7
RESET AND DISCONNECT LOGIC
2
(STATUS)
2
DISCONNECT
RESET
(FRONT PANEL OF MULTISOURCERER)

Figure 18. CDR Functional Block Diagram

Interface Control Section

The IBM Channel interface consists of Bus In, Bus Out, Tag In, Tag Out, Control In and Control Out lines. Bus In and Bus Out are multiplexed buses. That is, they may carry address, data or status information. The function of the Tag In and Tag Out lines is to identify the type of data currently on the Bus In or Bus Out lines. Control In and Control Out lines form a fully interlocked handshake system. Normally, once a control signal goes active, it is not allowed to change until it is acknowledged by the recipient.

Refer to Figure 18 on the adjacent page. There are a total of eight Tag and Control Out signals. These signals drive the inputs of the Channel Out Tag and Control Receivers block. This block passes six of the signals directly to the GPCI via the Tag and Control Line Drivers. Two of the Control Out signals (HOUT/ and SOUT/) are combined by the Board Control block to form a status signal called SHOUT/. SHOUT/ is also passed to GPCI via the Tag and Control Line Drivers.

Two Tag In signals are transmitted from the GPCI to the CDR on dedicated lines. These signals are DAN/ (Data In) and SEN/ (Service In). The remaining Tag In signals originate in the Tag and Control Latch.

With the exception of the Reset and Disconnect Logic, all remaining Interface Control Section circuitry involves the Select Out signal.

One of the Selection Relays connects the SELECT OUT (I) pin to the SELECT OUT (O) pin when power is off, maintaining the daisy chained Select Out signal path. This path is called the "bypass path."

Another Selection Relay, when energized, passes the SELECT OUT (I) signal to a Channel Out Tag and Control Receiver, whose output is SOUT/. Part of A13 produces SLCTO (Select Out) by gating SOUT/ with PSOUT/ (Propagate Select Out). PSOUT/ is an output of the Tag and Control Latch. SLCTO drives the input of the SELECT OUT (O) line driver, a part of the Channel In Tag and Control Drivers block. This Select Out path is called the "normal Select Out path." After the normal Select Out path is operational, the bypass path relay is energized, breaking the bypass path. Propagation or blocking of Select Out is controlled by the GPCI via PSOUT/ after the bypass relay is opened (energized).

Disconnect Circuits

There are several phases to an I/O operation. These phases include, but are not limited to, Initial Selection, Data Transmission or Data Reception, and the Ending Sequence. Additionally, the successful completion of an I/O cycle may automatically initiate another (see Command Chaining in the cited IBM I/O reference). When an on-line Multisourcerer must be taken off-line, it must be done only after an Ending Sequence has been completed for all devices currently in an active phase.

Once an I/O operation begins, it must have an Ending Sequence which contains a "Device End" status. If the Device End is not received, it is possible that the I/O channel or computer may hang up.

The Multisourcerer monitors the position of the front panel DISCONNECT switch to determine if it should disconnect from the channel. When the switch is in the disconnect position, the GPCI transmits "Device End" status for each active device. Then, the GPCI writes a control word to the CDR's Tag and Control Latch. Three of the bits in this control word control the Selection Relays and cause them to disconnect the Multisourcerer from the I/O interface.

Reset Circuit

The GPCI contains an 8085 microprocessor and a bit-slice processor. For proper operation, both processors require a reset immediately after power-up. The reset circuit generates a reset immediately following a power-up. It also provides a manual reset capability, which is a desirable feature on most microprocessor based systems, and is included on the Multisourcerer.

However, in addition to resetting the processors, a reset disconnects the Multisourcerer from the I/O interface by de-energizing all relays on the CDR (see Disconnect Circuits above). Thus, if a reset occurs while a command is in progress, the channel and/or the host computer can hang up. To prevent this undesirable effect, the manual reset function is inhibited at all times except when the Multisourcerer has been disconnected from the I/O interface via the DISCONNECT switch as described above. The manual reset is initiated by pressing the normally open RESET switch mounted on the front panel of the Multisourcerer. The manual reset feature and the power-up reset are combined to form a composite reset signal for the GPCI. This signal is called PURST/ (Power-Up Reset).

Board Control Section

Refer to Figure 18 on page 10-4. The Board Control Section is the input/output mode control for the CDR. In the Data Handling Section, it:

- controls data flow direction through the Bus Transceiver
- enables the Output Data Latch
- enables the Octal Tri-State Buffer
- preconditions the parity input to the Parity Checker/Generator

In the Interface Control Section, the Board Control Section:

- latches the Tag and Control Latch
- enables the Channel In Tag and Control Drivers
- controls the propagation of Select Out
- combines two tag signals (HOUT/ and SOUT/) to form SHOUT/, a GPCI input status signal

The heart of the Board Control Section is the Board Control block. This block requires 12 inputs to accomplish the functions listed above. Four of these inputs are control inputs from the GPCI via the GPCI Control Line Receivers. Two of these controls (CAD0 and CAD1) are used by the Board Control block as chip select addressing lines for several CDR components. CDCK/ functions as a synchronizing clock for the Board Control Blocks. CDIR/ determines the direction of data and control signals through the CDR.

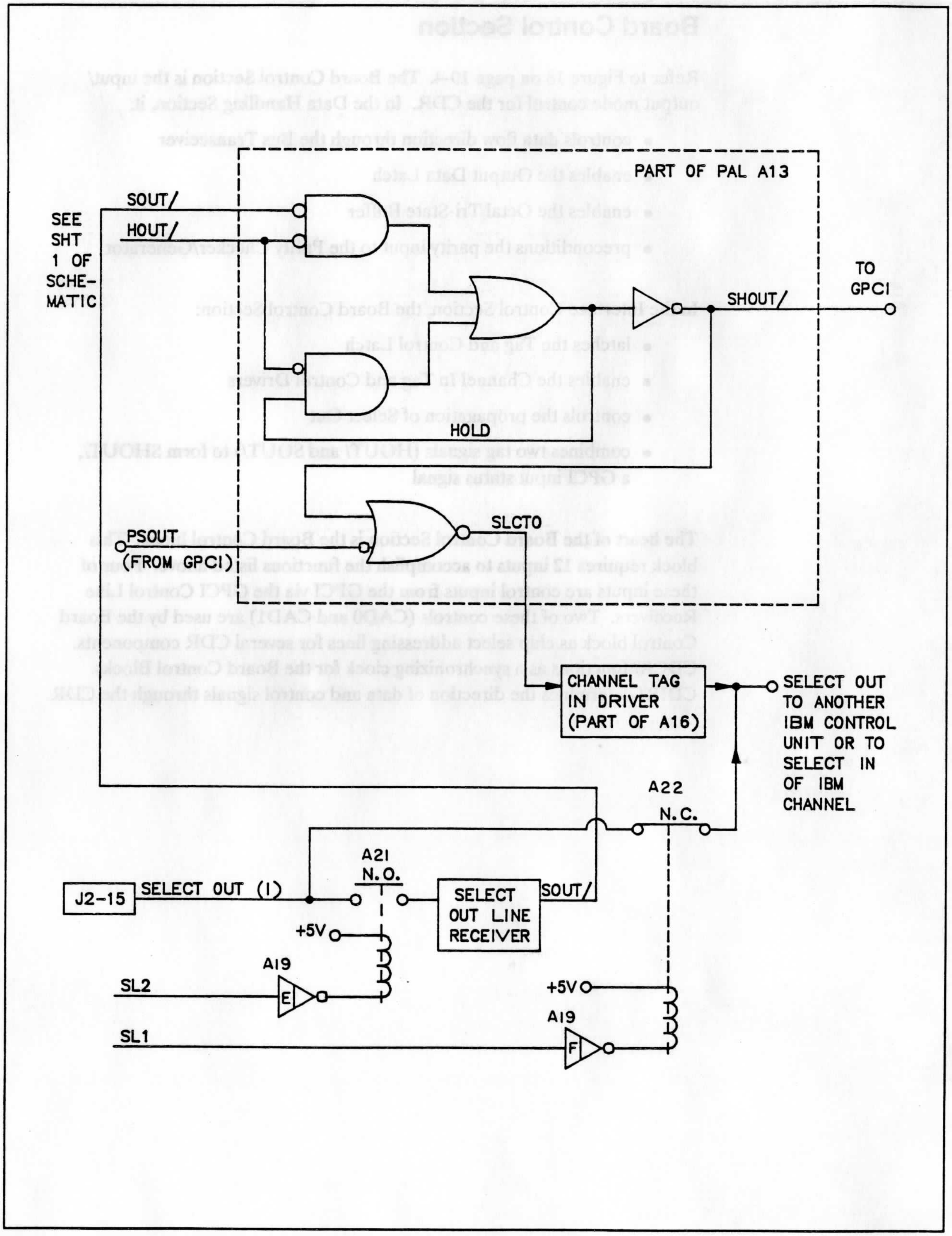


Figure 19. Select OUT and SHOUT/ Generation

CDR Detailed Circuit Description

The schematic diagrams of the Channel Drivers and Receivers board are shown on SSEC drawing #6450-0462 (dated 5/19/86), sheets 1-2.

Only the Board Control block, the Reset, and the Disconnect circuits are described here because the remainder of the board consists primarily of line drivers and receivers and latches. An understanding of the Board Control is essential to an understanding of the CDR board.

Schematic Conventions

To refer to a schematic circuit symbol of one section of a multiple section IC, the symbol ID number is used, followed by a hyphen and the section letter designator. The symbol ID number alone refers to single section ICs.

Logic Conventions

Logic signal names are in uppercase letters. Logic signals ending with a trailing slash are active low signals, i.e., SEN/. Signal names that do not end in a trailing slash are active high, i.e., CDDIR. All references to signal levels are by the use of "high" or "low." Thus, if CDDIR and SEN/ are both high, CDDIR is active and SEN/ is inactive.

Interface Control Section

Select Out

One of the IBM interface specifications requires that a control unit be able to propagate the level on the Select Out line, even when the power is turned off. Furthermore, application and removal of power from a control unit is not allowed to disrupt the Select Out signal. Three relays and associated control logic implement these specifications.

Refer to Figure 19 on the adjacent page. If the CDR is not powered up, all relays are as shown. The incoming Select Out signal at J2-15 connects directly to the normally closed contacts of A22. At this time, the signal passes directly to the output of the Select Out Line Driver. If the Select Out signal at J2-15 goes high, this high level passes immediately to the outgoing Select Out pin at J2-25. The Tag and Control Line Drivers have emitter-Follower outputs. Since the Select Out line driver is powered off at this time, the line driver's output transistor is reverse biased and is effectively an open circuit.

Applying power has no effect on the relay status or the Select Out driver, which is still disabled due to the closed contacts of SL3 (see sheet 2 of the schematic diagram). At this point, the GPCI brings the Multisourcerer on-line. First, the GPCI energizes relays A21 and A20. When relay A20 energizes, the Select Out Line Driver (A16) is enabled. When relay A21 energizes, the Select Out signal at J2-15 is applied to the Select Out Line Receiver (Section 3). The output of the Select Out Line Receiver drives the input of the Select Out Line Driver via A13. Finally, S1 is opened, opening the bypass path and placing the GPCI on-line.

The process of going off-line is a reversal of the above procedure. Pages 2-22 and 2-23 of the *IBM System 360 and System 370 Interface Channel to Control Unit Original Equipment Manufacturers Information* manual provide an excellent reference for the electrical and logical implementation of the Select Out specifications.

SHOUT/

Refer to Figure 19 on page 10-8. SHOUT/ is a status output to the GPCI and an input to the Select Out generator. SHOUT/ goes low when Hold Out and Select Out go high (SOUT/ and HOUT/ go low). When SHOUT/ goes low it remains low until HOUT/ goes high. The channel drops Hold Out (HOUT/ goes high), as a signal to the GPCI that it may end the current operation when desired. The GPCI cannot logically disconnect from the channel while Hold Out is high.

SLCTO

SLCTO is the input to the Select Out Line Driver. Refer to Figure 19. PSOUT is an input from the Tag and Control Latch and is thus controlled by the GPCI. The Multisourcerer can share an IBM I/O channel with other control units. SHOUT/ goes low if the GPCI or any other control unit on the same IBM channel as the Multisourcerer becomes selected (Hold Out is high). Therefore, SHOUT/, one of the inputs to SLCTO, is low even when the selected unit is not the GPCI.

When the GPCI is not the selected unit, SELECT OUT (O) may be a high or low signal, but it must be the same as SELECT OUT (I). If Select Out and Hold Out are both low (SOUT/ and HOUT/ are high), SHOUT/ is high, forcing SLCTO low regardless of PSOUT. This situation occurs if no control unit is currently selected, or a control unit upstream of the GPCI is currently selected.

If Select Out and Hold Out both go high, SHOUT/ goes low. During this time, the SLCTO signal is determined by PSOUT; if PSOUT is high, SLCTO is high and vice versa. The GPCI controls PSOUT by monitoring Address Out (ADT/ goes low) and the address on Bus Out. If Address Out goes high (ADT/goes low) and the address on Bus Out matches the assigned GPCI address, the GPCI sets PSOUT low. At this time, SELECT OUT (I) is high while SELECT OUT (O) to the next control unit is low, indicating a control unit connection. If the address on Bus Out does not match, PSOUT is high, propagating the high Select Out signal to the next control unit.

Disconnect Circuit Description

Refer to sheet 1 of the schematics. When the DISCONNECT switch is closed, R5, R6, and the input impedance of A12-H form a voltage divider. This results in an input voltage to A12-H of about 0.33V. This voltage is well below the turn on voltage of A12-H. C14 and R6 act as a switch debounce circuit when the DISCONNECT switch is opened. C14 integrates any switch bounce transients, preventing them from exceeding the threshold of A12-H.

The output of A12-H is inverted by A11-H to form the signal DISCNT/. The output of A12-H is also an input to the Reset circuits (A19-C). DISCNT/ is high when the DISCONNECT switch is closed. DISCNT/ is transported to the GPCI via J3-23. The GPCI monitors DISCNT/ and automatically sends "Device Ends" for any in-progress I/O operations if DISCNT/ goes low. Further, no new I/O operations are allowed until the switch is closed.

Reset Circuit Description

See sheet 1 of the schematics. A12-H has a low output when the DISCONNECT switch is closed. This output is inverted by A19-C and applied to J4-8. When the RESET switch is pressed, J4-8 is connected to J4-7. The RESET switch is normally open, and C1 charges to +5 volts through R2. Therefore, if the RESET switch is pressed while the DISCONNECT switch is closed, the high output of A19-C is connected to the high at the top of C1, resulting in no change in the charge on C1. If the DISCONNECT switch is open (disconnect position) when the RESET switch is pressed, the low at the output of A19-C rapidly discharges C1, placing a low at the input to A12-D. When the input of A12-D goes low, its output goes low. A12-D's output drives the CLR (clear) input pin on the Tag and Control Latch. It also drives the PURST/ buffer (A12-E).

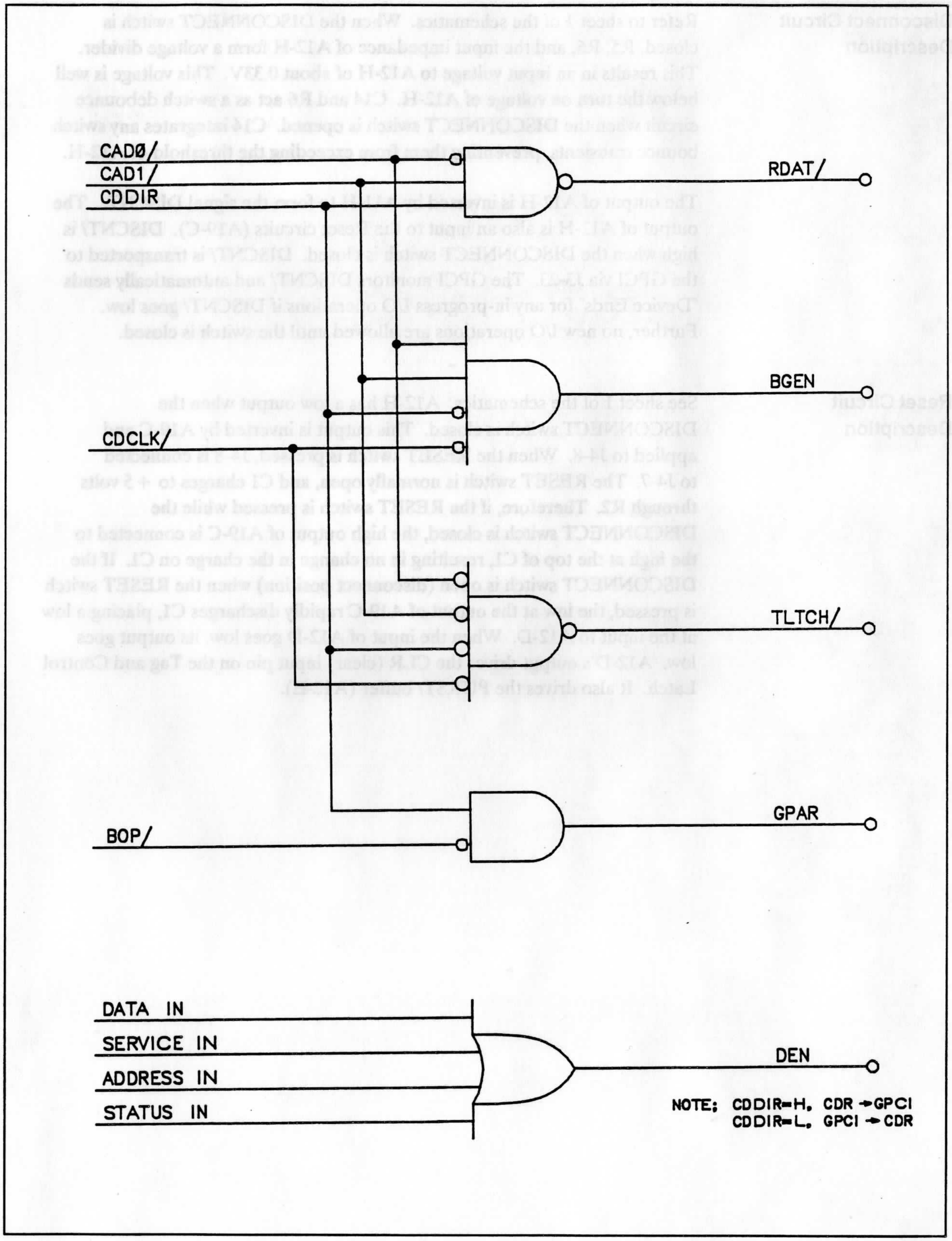


Figure 20. CDR Control Logic (Part of PAL A13)

Board Control Section

Refer to Sheet 2 of the schematics. The Board Control block consists of A13 and the GPCI Control Line Receivers. A13 is a Programmable Array Logic (PAL). A13 (PAL20L10) has 12 inputs that control seven outputs. The equations for the input/output relationships are in the CDR Supplemental Data Chapter (Chapter 11).

Refer to Figure 20 on the adjacent page and Sheet 2 of the schematics. Figure 20 is the logic equivalent of the logic equations for RDAT/ (Receive Data enable), BGEN (Bus Gate Enable), TLTCH/ (Tag Latch enable), GPAR (Gated Parity bit), and DEN (Data Enable). Each of these signals is shown in Figure 18 as well as on the schematic diagram.

RDAT/, BGEN, and TLTCH/

RDAT/, BGEN, and TLTCH/ result from combinations of three signals. Each circuit is composed of CAD0 (Command Address 0), CAD1 (Command Address 1), and CDDIR (Channel Data Direction). The GPCI supplies these signals to the Board Control via the GPCI Control Line Receivers. Note that CDDIR is high when the GPCI is reading data from the CDR; CDDIR is low while the GPCI is writing data to the CDR. CDDIR is an enabling signal to each gate while CAD0 and CAD1 form a two-bit address by which a particular gate is qualified. CDCLK (Command Clock) synchronizes BGEN and TLTCH/ with the GPCI output data. The table below shows the relationship between RDAT/, BGEN and TLTCH/, and CAD0 and CAD1. Note that all input signals are control outputs of the GPCI.

CAD1	CAD0	A13 Output (Partial)
low	low	BGEN
low	high	RDAT/
high	low	not used
high	high	TLTCH/

GPAR, the parity bit input to the Parity Checker/Generator block (A4), is a function of BOP/ (Bus Out Parity) and CDDIR. When CDDIR is low, GPAR is low. When CDDIR is high, the GPAR output is an inversion of BOP/. CDDIR controls the direction of data flow through the Bus Transceiver, and is high during data transmission from the channel to the GPCI. During this time, the Bus Out Parity Check from A4 is a status output to the GPCI. As long as the parity is correct, CHPAR/ is low.

When CDDIR is low, the GPAR input to A4 is low and A4 computes the value of the parity bit based on the eight data inputs from the GPCI. During this time, the BOPAR (Bus Out Parity) signal is output to A18-A which is enabled by BGEN. The output of A18-A is the input to the BIP (Bus In Parity) line driver.

DEN

Refer to Figure 20 and sheet 2 of the schematics. DEN is buffered by A11-A and A19-A to produce DREN. DREN drives the enable inputs of the Bus In and Parity In line drivers (A8, A9 and aA7-A respectively). DREN also enables A17, a quadruple line driver that drives Data In, Service In, Address In and Status In. All of these line drivers are enabled during an I/O operation involving one of the Multisourcerer's devices. The remaining line drivers (Select Out, Request In and Operational In) are sections of A16. These line drivers must be enabled any time the Multisourcerer is on-line regardless of the status of the Multisourcerer's devices. All line drivers must be disabled immediately if the channel sends an Interface Disconnect Sequence. These line driver control features are implemented by using A19-A, relay A20 and diode D2.

Refer to sheet 2 of the schematics. When relay A20 is energized, A16 is enabled via pull-up R7. At this time, the cathode of D2 is about +5 volts, and causes D2 to be off. Resistor R8 acts as a pull-up resistor for A8, A9, A17 and A7. It is also the collector load resistor for buffer A19-A. Thus, when DEN is high, A19-A is off (DREN goes high) and all line drivers are enabled. When DEN is low, A19-A is on and DREN is low. This disables A7, A8, A9 and A17.

When relay A20 is de-energized (as shown), the enable input of A16 and the cathode of D2 are grounded through the closed contacts of A20. D2 is a germanium diode. If DREN is high when relay A20 is de-energized, D2 is forward biased. Since the forward bias of a germanium diode is only about 0.1 volts, the anode of D2 is about +0.1 volts. This is well below the enable threshold of A8, A9, A7 and A17. Thus, when relay A20 is de-energized, all line drivers are disabled regardless of the status of DREN.

CDR Supplemental Data

CDR Supplemental Data

This Supplemental Data section contains:

- an explanation of the PAL equation listings
- the symbols and abbreviations used in the equations
- one PAL equation
- Schematic drawings 1 and 2 (6450-0462)
- Assembly drawing 1 (6450-0463)

PAL Equation Listings

The listings for the logic equations used in the Programmable Array Logic Devices are explained below.

Title

Each device has a printout that begins with the title page. On this page the device is called out by its location and device name.

Declarations

Declarations lists the logic conventions and format for the signal names. The signal names are in uppercase letters and numbers.

The first group of signal names refers to the power and ground rails. The respective pin numbers that make these connections are given in the line below the signal names.

The second group of signal names refers to the input signals received by the device. The respective pin numbers assigned to receive these inputs are given in the line below the signal names.

The third group of signal names refers to the outputs from the PAL. The respective pin numbers assigned to each output are given in the line below the signal names.

Equations

The logic equations used to generate each signal are shown in their highest order form.

Symbols and Abbreviations

The following is an example of a PAL equation.

```
!PDOT_  := !DIN & DSEL_ & QBSEL_ & DRQ & DMACLK
         # PDOT & !DIN & DSEL & QBSEL_ & DRQ
         # !PDOT_ & DAK_
```

Below is an explanation of the symbols and abbreviations used in the equation.

Symbol/Abbreviation	Explanation
!	The one's complement of, e.g. !PDOT = $(\overline{\text{PDOT}})$
:	A register latched signal; valid on the rising edge of the register clock
&	Logical AND
#	Logical OR
_	Denotes an active low signal

P20L10

Title

Channel Driver/Receiver Board
SSEC Madison, WI 05/05/86

Identification

Device type P20L10

Module

```
DRIREC
flag , -r0,
```

Declarations

```
TRUE, FALSE = 1, 0
H, L = 1, 0
X, Z, Ck = .X., .Z., .C.

GND, VCC
        pin 12, 24

ST_, HT_, BOP_, CA0_, CA1_, CLK_, DTI, SRI, DDR, PSO, ADI, STI
        pin 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13

TLH_, BGEN
        pin 14, 23

SLCT, DEN, GPAR, SHT_, RDT_, DIC_, NC, NBGEN_
        pin 15, 16, 17, 18, 19, 20, 21, 22
```

Equations

```
!BGEN  = NBGEN_

!NBGEN_  = CA0_ & CA1_ & !DDR & !CLK_

!RDT_    = !CA0_ & CA1_ & DDR

!SHT_    = !ST_ & !HT_
         # !SHT_ & !HT_

!GPAR    = BOP_
         # !DDR

!DEN     = !DTI & !SRI & !ADI & !STI

!SLCT    = SHT_
         # !PSO

!TLH_    = !CA0_ & !CA1_ & !DDR & !CLK_

!DIC_    = CA0_ & !CA1_ & !DDR & !CLK_
```

Notes

The symbols used in the above equations are logical and not physical levels.

```
GPAR    =  BOP_ * DDR

DEN     =  (DTI + SRI + ADI + STI) * SHT_

SLCT    =   SHT_
        +  PSO
```

ST_ : = Select Out

HT_ : = Hold Out

DTI : = Data In

SRI : = Service In

DDR : = Data Direction

PSO : = Pre Select Out

ADI : = Address In

STI : = Status In

TLH_ : = Tag Latch

SLCT : = Select

DEN : = Data Enable

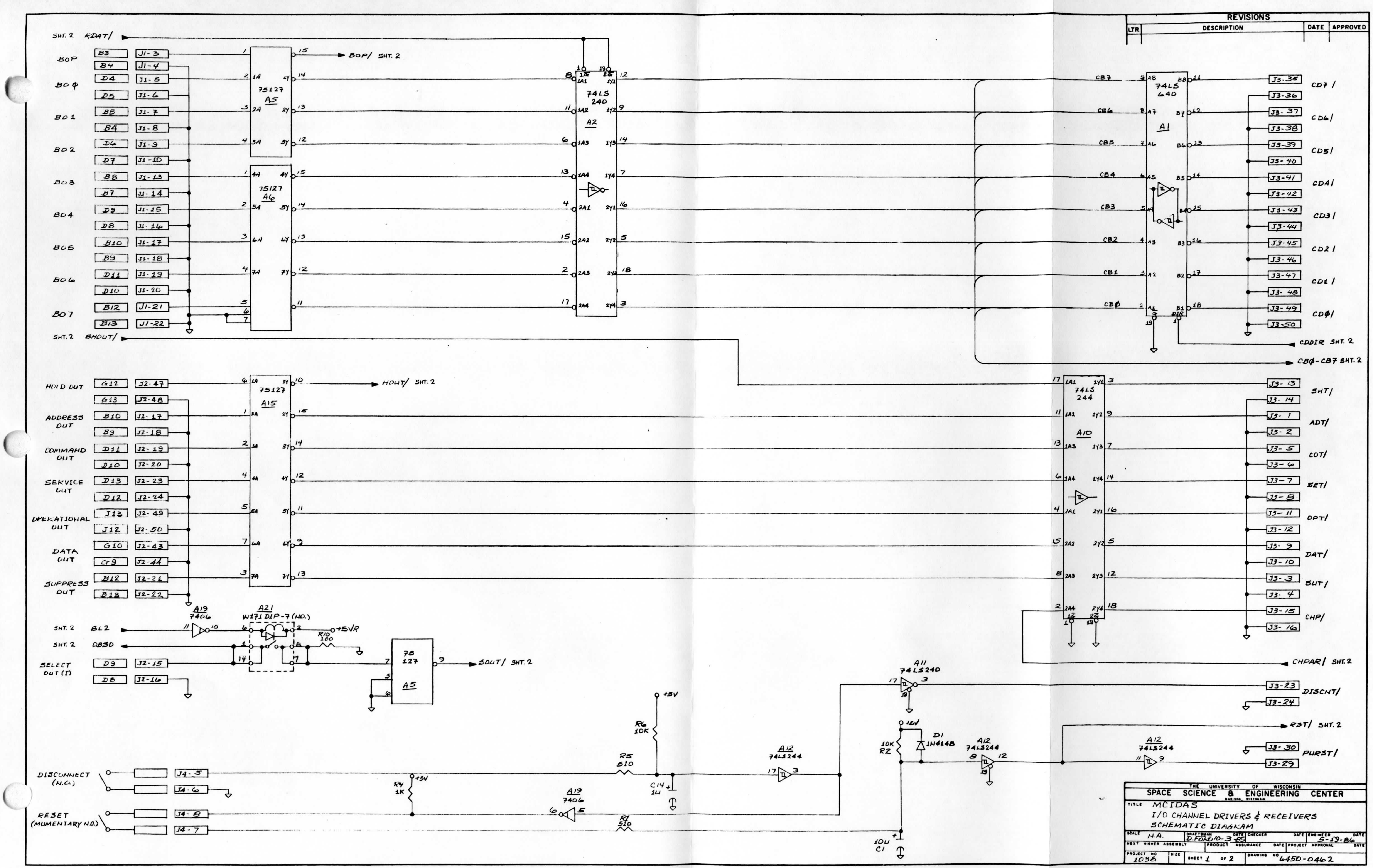
GPAR : = Generate (Check) Parity

SHT_ : = Select and Hold Out

RDT_ : = Read Data

BGEN : = Bus Gate Enable

DIC_ : = Disconnect In Clock



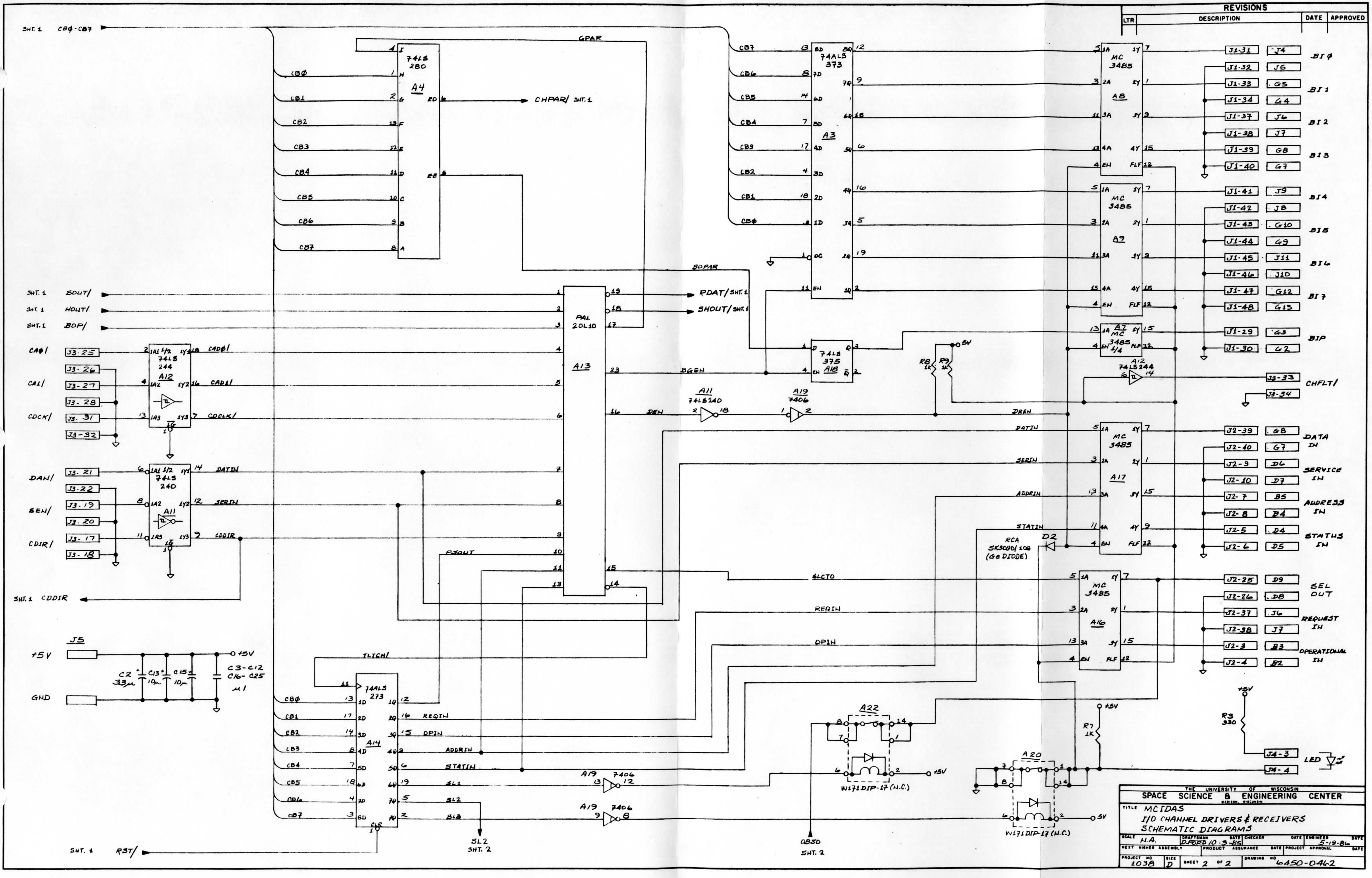
REVISIONS

LTR	DESCRIPTION	DATE	APPROVED

THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

TITLE MCIDAS
I/O CHANNEL DRIVERS & RECEIVERS
SCHEMATIC DIAGKAM

SCALE N.A.	DRAFTSMAN D.FORD DATE 10-3-85	CHECKER DATE	ENGINEER DATE 5-19-86
NEXT HIGHER ASSEMBLY	PRODUCT ASSURANCE DATE	PROJECT APPROVAL DATE	
PROJECT NO 1036	SIZE	SHEET 1 OF 2	DRAWING NO 6450-0462



LTR.	DESCRIPTION	DATE	APPROVED

THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

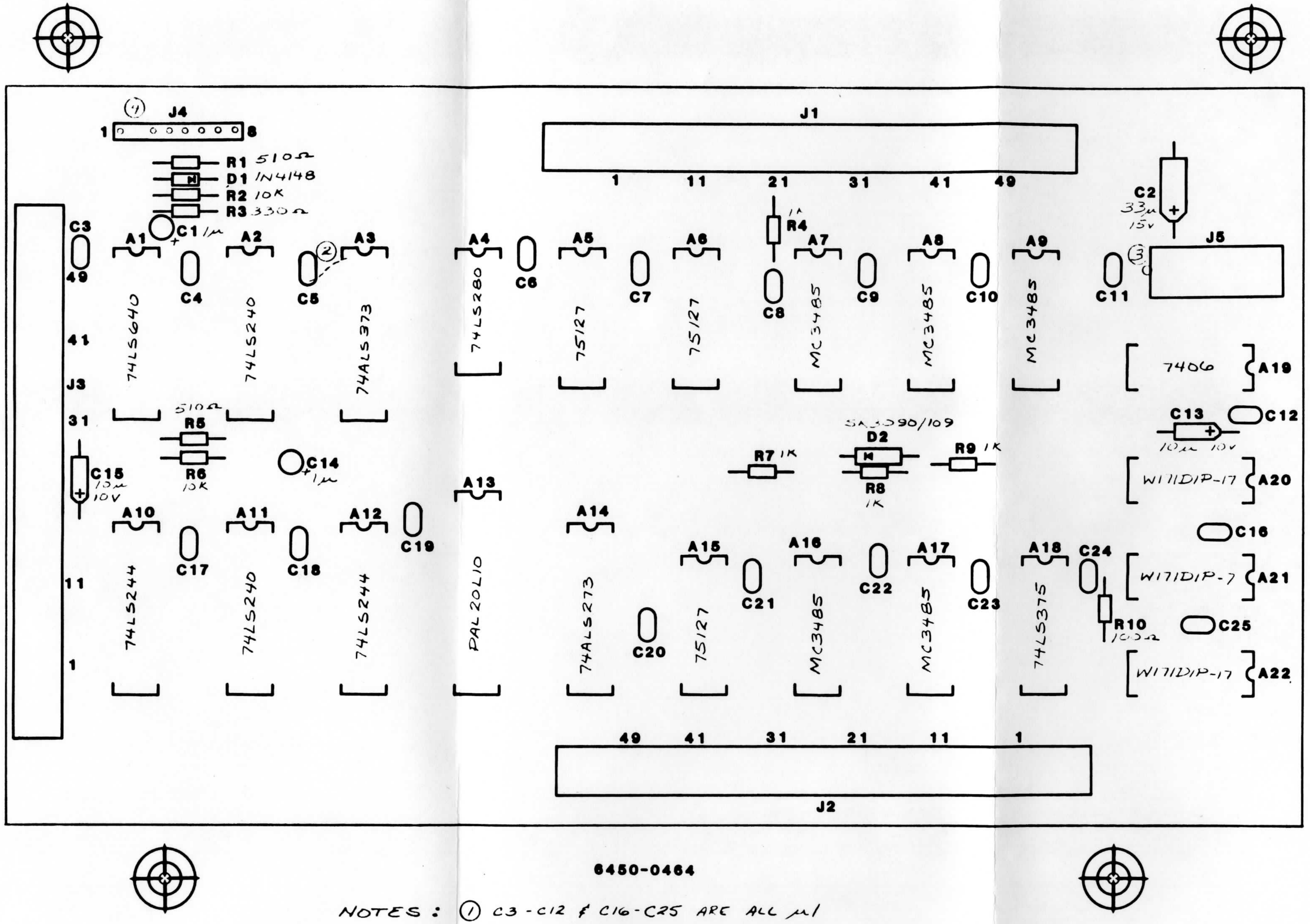
TITLE MCIDAS
I/O CHANNEL DRIVERS & RECEIVERS
SCHEMATIC DIAGRAMS

SCALE	DRAFTSMAN	DATE	CHECKER	DATE	ENGINEER	DATE
N.A.	D.FORD	10-3-85				5-19-86

NEXT HIGHER ASSEMBLY | PRODUCT ASSURANCE DATE | PROJECT APPROVAL DATE

PROJECT NO	SIZE	SHEET	DRAWING NO
1038	D	2 OF 2	6450-0462

REVISIONS			
LTR.	DESCRIPTION	DATE	APPROVED



6450-0464

NOTES:

(1) C3 - C12 ≠ C16-C25 ARE ALL μl

(2) ADD WIRE FROM GROUND OF C5 TO A3 PIN 1 ON CIRCUIT SIDE

(3) DRILL OUT GUIDE HOLE OF J5 WITH #33 DRILL

(4) J4 CONNECTOR DOES NOT HAVE A PIN 2

THE UNIVERSITY OF WISCONSIN
SPACE SCIENCE & ENGINEERING CENTER
MADISON, WISCONSIN

TITLE McIDAS
I/O CHANNEL DRIVERS ≠ RECEIVERS
ASSEMBLY DRAWING

SCALE 2/1	DRAFTSMAN JW	DATE 3-24-86	CHECKER	DATE	ENGINEER DATE
NEXT HIGHER ASSEMBLY		PRODUCT ASSURANCE		DATE	PROJECT APPROVAL DATE
PROJECT NO 1038	SIZE C	SHEET 1 OF 1	DRAWING NO 6450-0463		

Index

Index