IBM

Personal Computer
Hardware Reference
Library

PROFESSIONAL GRAPHICS CONTROLLER

IBM Personal Computer Professional Graphics Controller Technical Reference

6138161
August 15, 1984

Contents

Description

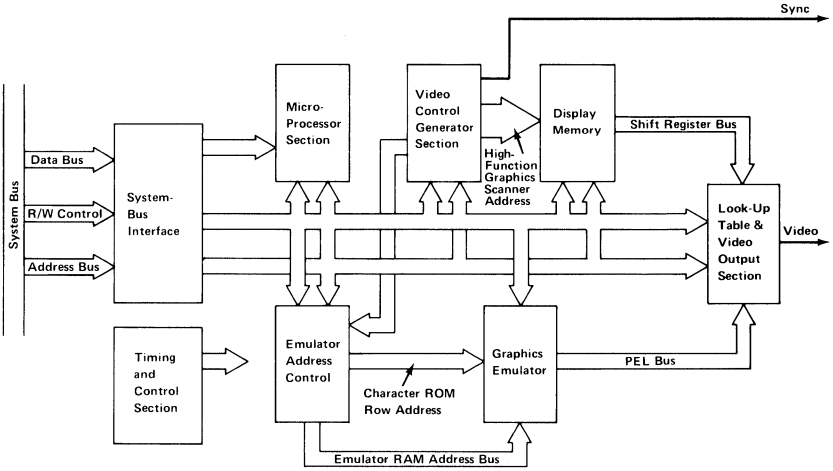
The IBM Personal Computer Professional Graphics Controller is an adapter that: (1) provides a high-function graphics capability and (2) acts as an IBM Color/Graphics Monitor Adapter, with the exception of the 160-by-100 color/graphics mode.

The operations of the Professional Graphics Controller are controlled by an 8088 Microprocessor. It carries out all communications through its data bus and address bus. The system-bus interface recognizes its own commands and passes only these commands to the controller. The interface allows the microprocessor to read or write to memory locations, using the IBM Professional Graphics Controller microprocessor's data and address busses.

The microprocessor controls and initializes several sections of the controller. It defines the requirements of the controller's hardware so the controller can imitate the actions of the IBM Color/Graphics Monitor Adapter. The microprocessor also regulates the emulator address control, which translates the system's I/O address information and stores the associated data in the graphics emulator memory for screen display. Finally, it initializes the video control generator, which generates timing pulses and the horizontal- and vertical-synchronization (sync) pulses.

During operation, the microprocessor intercepts commands sent to the emulator and interprets them. The microprocessor can also accept and interpret the high-function graphics commands, writing the results in the display memory for screen display. Both the emulator and high-function graphics functions have access to the look-up table (LUT) and output section.

The following is a block diagram of the Professional Graphics Controller.



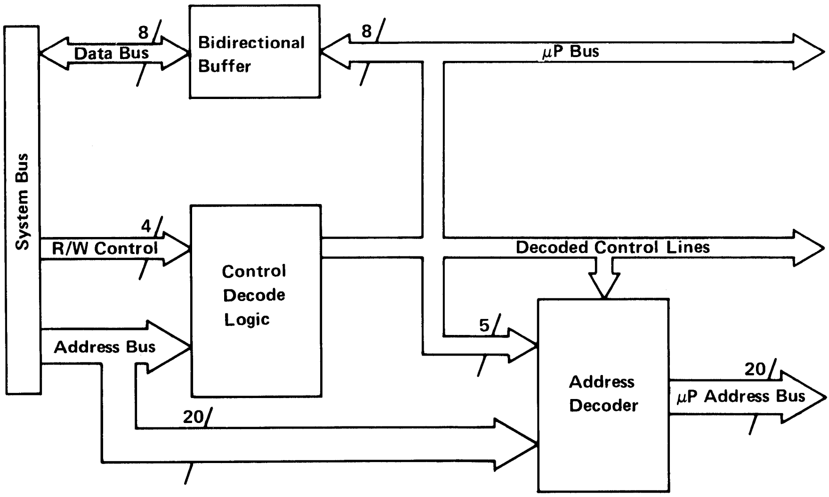
Major Components

- System-Bus Interface
  - Bidirectional Buffer
  - Control Decode Logic
  - Address Decoder
- Microprocessor Section
  - 8088 Microprocessor
  - Clock Generator Control
  - Address Latch
  - Data Latch
  - Decoders
  - 2K by 8-bit RAM
  - 64K by 8-bit ROM
- Video Control Generator Section
  - Video Controller
  - Control Decoder
  - 16- by 8-bit State Length Memory
  - Synchronization Pulse Generator
  - State Multiplexer
  - Vertical and Horizontal State Counters
  - Vertical and Horizontal State Length Counters
  - Buffer
- Emulator Address Control
  - Controller
  - Cursor Generator
  - Parameter Registers
  - Character ROM Address Generator
  - Row Address Generator
  - Column Address Generator
  - Microprocessor Address Buffers
- Graphics Emulator
  - 16K by 16-bit Emulator RAM
  - Shift Registers
  - Character ROM
  - Attribute Latch
  - Emulator PEL Processing
  - Buffer

- Display Memory
  - High-Function Graphics Display Memory
    - Latch
    - Tri-State Bidirectional Driver
    - Tri-State Latch
    - 320K by 8-bit RAM
  - Display RAM Address Control
    - High-Function Graphics Scanner
    - ROM
    - Buffers
- Look-Up Table (LUT) and Video Output Section
  - Latches
  - Look-Up Table Memory
  - Buffer
  - Triple Digital-to-Analog Converter
- Timing and Control Section
  - 50-MHz Oscillator
  - High-Function Graphics Display Timing Generator
  - Control Decoder and Latches

System-Bus Interface

Following is a block diagram of the system-bus interface.



The system-bus interface allows the system microprocessor to gain access to the display memory and emulated registers through the 'data,' 'address,' and 'control' lines. The system-bus interface can detect the attempt by the system microprocessor to execute a Memory Write command or an I/O Write command to either the emulator memory addresses or the communications memory for the high-function graphics mode.

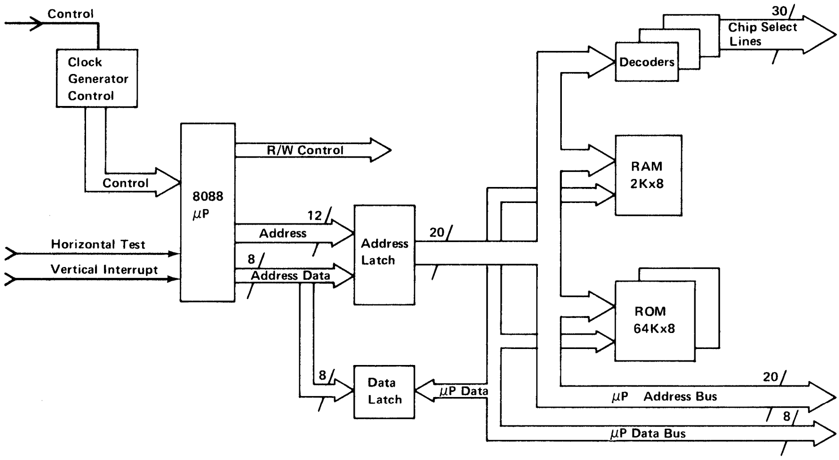
When the interface logic detects an assigned address, a 'hold' signal is sent to the system microprocessor, which suspends the operation of the controller microprocessor until the proper time. Although the system microprocessor can gain access to the memory of the controller microprocessor (through a series of commands on the bus interface), it cannot directly access the display RAM, nor can it issue interrupts to the controller microprocessor. Likewise, the controller microprocessor cannot gain control of the system bus.

If the system microprocessor writes to a register of the emulated 6845 CRT Controller, the data is stored in the controller's local RAM.

The controller operates by mapping both the I/O addresses and the addressed memory into its own memory. It then reads these locations, interprets the data, and programs the hardware to imitate the IBM Color/Graphics Monitor Adapter. If high-function graphics commands are written to the communication area, the controller microprocessor interprets those commands and writes to the display memory for screen display.

Microprocessor Section

Following is a block diagram of the microprocessor section

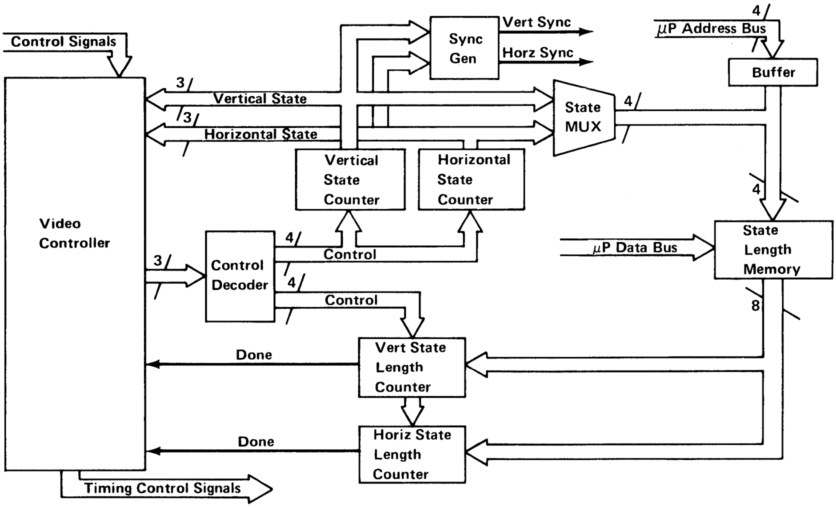


The microprocessor section is a standard 8088 Microprocessor arrangement. A ‘timing control’ line’s input leads into a clock generator control. The control signal emitted from the clock generator provides the clock frequency that drives the 8088 Microprocessor. Address and data latches store the signals sent over the address and data busses. Both the address and data lines use two 32K by 8-bit ROMs and a single 2K by 8-bit static RAM. The decoders control chip-select and latch registers.

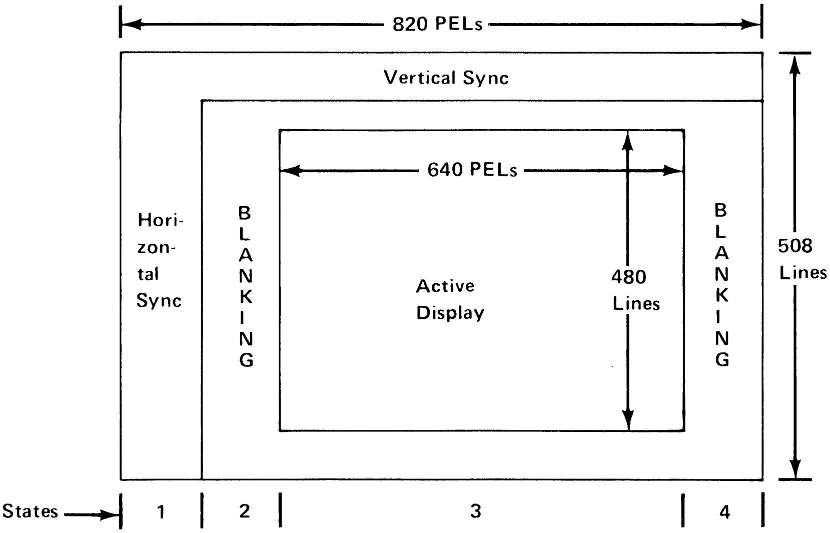
A single, maskable interrupt occurs from the ‘vertical interrupt’ line. The test pin of the microprocessor samples the horizontal-synchronization pulse.

Video Control Generator Section

Following is a block diagram of the video control generator section.



The video controller monitors and sequences the video control generator section. The main loop of the control generator controls the format of the display screen. A display screen is divided into four states, as shown in the following.



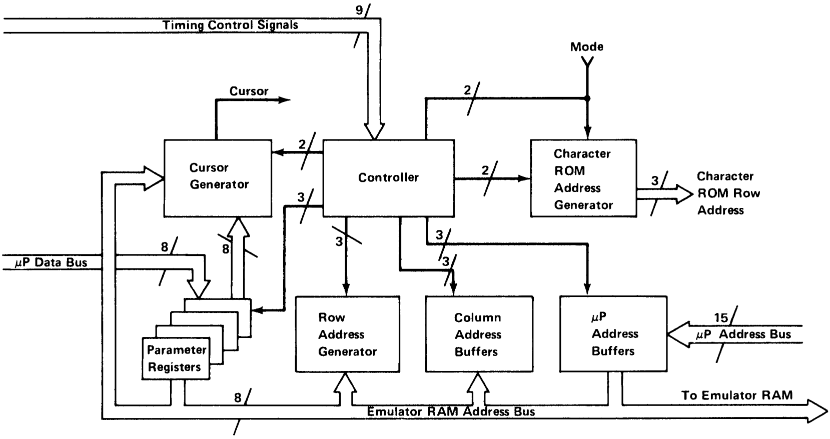
The state length memory is a part of the video control generator section. The contents of the state length memory provide the data to the state length counters, which then determine how long each state remains active. For each scan line, the state length memory loads this data, one at a time, into the horizontal state length counter. At the end of the count, the counter signals 'done' to the video controller, which then sets the control lines or particular stages of each state and sends the control information into the horizontal state counter. The video controller determines whether to start again at zero for some state, or to increment the state counter and begin on the next state. The horizontal state counter counts the number of states across the screen. From the state counter, the synchronization pulse generator determines the vertical- or horizontal-synchronization pulse and activates the appropriate line.

This same loop occurs for vertical states. The video controller monitors the current vertical and horizontal states through the state counters and synchronization pulse generator.

The controller microprocessor can write directly to the state length memory to vary the size of each state on the screen. State lengths remain under program control.

Emulator Address Control

Following is a block diagram of the emulator address control.



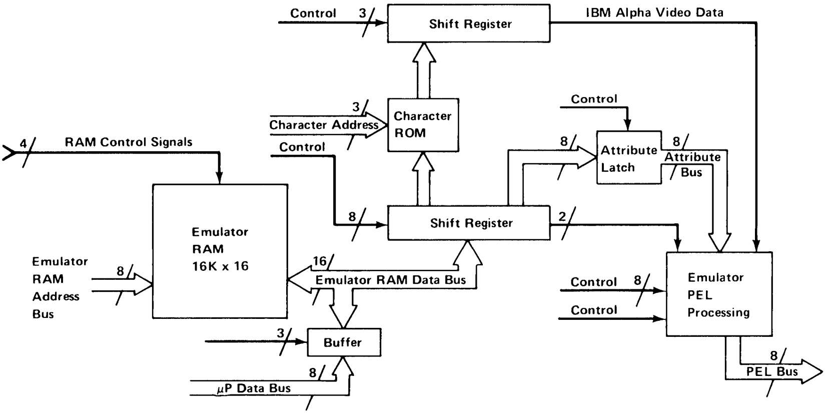
For the emulator mode, the address control consists of two generators—a row address generator and a column address generator. Both are driven by a controller and produce the addresses needed for the emulator RAM.

The controller microprocessor can access the address bus to program the address generators using an address buffer, and can program the four parameter registers. The cursor generator compares the addresses saved in the address generator with those saved in the parameter registers. If a match is found, the cursor generator activates the 'cursor' line.

The character ROM address generator produces a character ROM row address that defines which line to write using a font with 8 by 16 character cells.

Graphics Emulator

Following is a block diagram of the graphics emulator.



The emulator RAM address bus sends signals to the 16K by 16-bit emulator RAM. The 16-bit-wide RAM allows the character and its attributes to be read simultaneously. The RAM shifts this information into a register that also acts as a latch. During the alphanumeric mode, this information travels through an attribute latch and the character ROM. The character ROM checks the shift in the look-up table (LUT) before passing the information through another shift register.

The attributes determine the foreground and background colors of the character. The picture element (PEL) processor then shifts this information out onto the PEL bus.

During the 320-by-200 and 640-by-200 modes, the emulator RAM shifts out the information 16 bits at a time. The shift register then shifts out its signals two bits at a time into the PEL processor. The 640-by-200 mode uses these two bits alternately as either black or white values. The 320-by-200 mode uses the same two bits to determine the color placed on the screen.

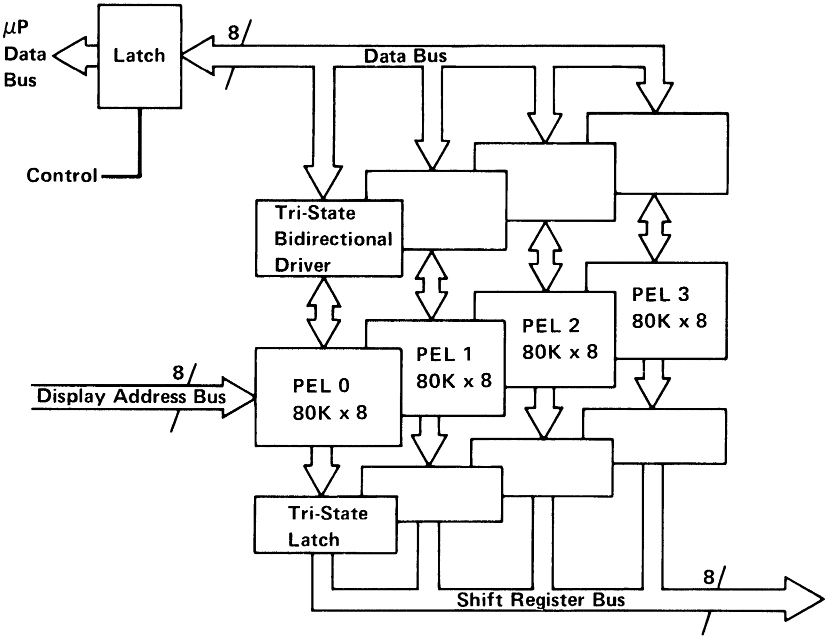
The system microprocessor can read and write directly into the emulator RAM space using the CPU address bus.

Display Memory

The display memory block consists of the high-function graphics display memory and the display RAM address control.

High-Function Graphics Display Memory

Following is a block diagram of the high-function graphics display memory



The high-function graphics display memory is logically arranged as an array of 640-by-480 PELs. Each PEL represents one byte of data. The Professional Graphics Controller provides a variety of PEL write modes to improve the transfer of data to display memory.

The high-function graphics display memory consists of five, 32-bit-wide banks (32 bits equal 4 PELs). The controller microprocessor can write through the latch into the PEL memory. All information is read from each memory and displayed each time the picture is scanned. This process begins when the tri-state drivers latch four PELs. Each tri-state driver is enabled individually as the beam crosses the screen. After the fourth PEL appears on the screen, four new PELs become latched.

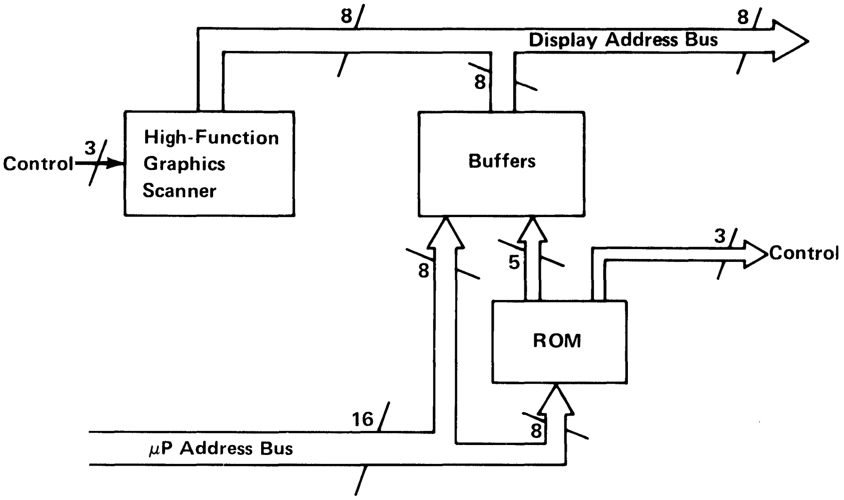
In the high-function graphics mode, the high-function graphics scanner generates addresses for a display access cycle on one of the five banks every 160 nanoseconds (ns). These cycles are staggered over an 800-ns period. Of the 32 bits of data latched from the memory, one PEL is released onto the shift register every 40 ns. The address selection generator, a field programmable logic sequencer (FPLS), interleaves microprocessor access cycles between display cycles, thus providing the possibility of access every 160 ns. This process achieves a display-memory access capacity of 32 bits every 80 ns.

During a microprocessor write operation, even in multi-PEL write modes, all data from the microprocessor is latched, so the microprocessor receives a 'ready' instantly. The FPLS cycles to the correct locations, or to all locations, depending on the mode, while the microprocessor prepares for the next access.

Another important aspect of the display memory is low power consumption. The staggered access technique reduces the RAM cycle time to as low as 400 ns, even with both the microprocessor and display at full capacity. When the display operates alone, the cycle time increases to 800 ns, minimizing RAM power consumption.

Display RAM Address Control

Following is a block diagram of the display RAM address control.

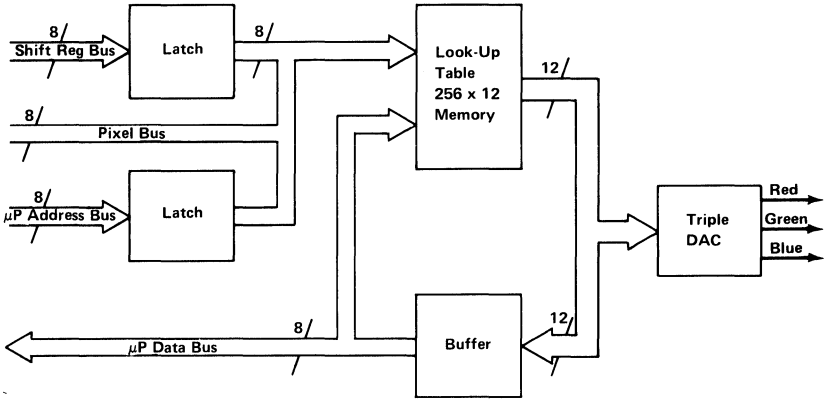


In the high-function graphics mode, the high-function graphics scanner operates as an address generator. The scanner output selects data from each of the five 32-bit-wide banks (for a total of 20 PELs written). The controller microprocessor expects memory to appear in a continuous manner; that is, 640 PELs across. The address-translator ROM is an address map of 640 adjacent memory locations. This provides the display format, thus leaving the controller microprocessor out of the conversion process.

Because this address system operates on 20-PEL boundaries, the memory for each line maps into an adjacent space of 640 locations for microprocessor access. Otherwise, if the microprocessor did the work, the very high writing speeds would be reduced.

Look-Up Table and Video Output Section

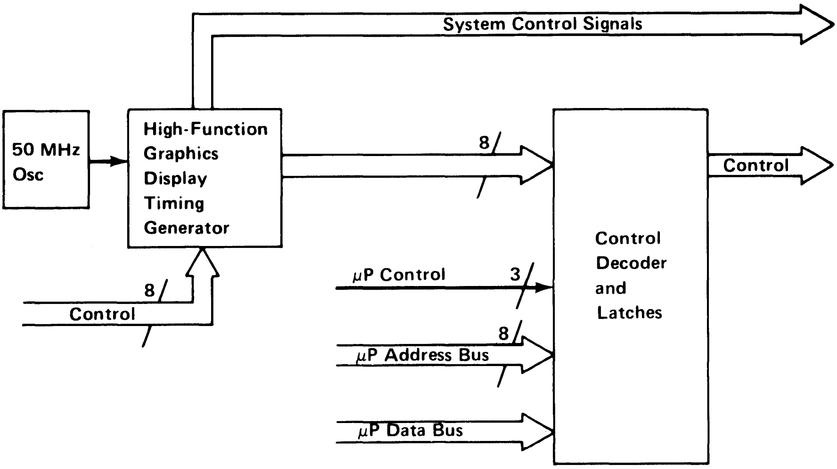
Following is a block diagram of the look-up table and video output section.



Shift registers from the display memory latch onto the PEL bus leading from the emulator. Both the emulator and high-function graphics modes use the same PEL bus. The latches provide an address for data in the look-up table (LUT). The eight lines of the PEL bus provide up to 256 colors, while the 256- by 12-bit LUT in memory provides a selection from a palette of 4096 colors. The LUT generates the color sent as output. The 12 LUT output lines (4 bits each for red, green, and blue) are the inputs to a triple digital-to-analog converter (DAC), which converts the signal to red, green, and blue (RGB) intensities. The controller microprocessor can write to and read from the LUT.

Timing and Control Section

Following is a block diagram of the timing and control section.



The high-function graphics-display timing generator, which is driven by a 50-MHz oscillator, sends control signals for memory and for the latch control from the display memory. It signals the controller microprocessor when it is ready to receive or send data from display memory. Except for system control signals, the signals from the timing generator are latched and decoded. The controller microprocessor maintains some control of the latches and decoder. The timing generator also generates clock signals to synchronize the board functions.

Emulator Modes

To provide compatibility with the Color/Graphics Monitor Adapter protocols, the Professional Graphics Controller emulates the Color/Graphics Monitor Adapter in the alphanumeric and graphics modes.

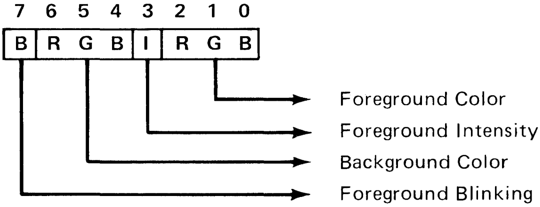
> **Note:** If a Color/Graphics Adapter is already present in the system unit, the emulator section of the Professional Graphics Controller is disabled with the enable/disable jumper.

Alphanumeric Mode

Every display-character position in the alphanumeric mode is defined by two bytes in the regen buffer, not the system memory. Both the Professional Graphics Controller and the Color/Graphics Monitor Adapter use the following 2-byte character or attribute format.

Display-Character Code Byte	Attribute Byte
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

The attribute byte definitions are:



The following table provides a summary of available colors.

I	R	G	B	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	White
1	0	0	0	Gray
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Light Red
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	White (High Intensity)

In the alphanumeric mode, the display mode can be operated in either a 40-by-25 mode or a 80-by-25 mode.

40-by-25 Alphanumeric Mode

The 40-by-25 alphanumeric mode:

- Displays up to 25 rows of 40 characters each
- Has a ROM character generator that contains dot patterns for a maximum of 256 different characters
- Requires 2000 bytes of read/write memory (on the controller)
- Has a 16-high by 8-wide character box
- Has one character attribute for each character

80-by-25 Alphanumeric Mode

The 80-by-25 alphanumeric mode:

- Supports the IBM Professional Graphics Display
- Displays up to 25 rows of 80 characters each
- Has a ROM character generator that contains dot patterns for a maximum of 256 different characters
- Requires 4000 bytes of read/write memory (on the controller)
- Has a 16-high by 8-wide character box
- Has one character attribute for each character

Graphics Mode

The Professional Graphics Controller has two modes available with the graphics mode—the 320-by-200 color/graphics mode and 640-by-200 black-and-white graphics mode. Both are supported in ROM. The following table summarizes the two modes.

Modes	Number of Colors Available (Includes Background Color)
320 x 200	4 Colors Total 1 of 16 for Background and 1 of Green, Red, or Brown or 1 of Cyan, Magenta, or White
640 x 200	Black-and-white only

320-by-200 Color/Graphics Mode

The 320-by-200 color/graphics mode supports the Color Display. It has the following features:

- Contains a maximum of 200 rows of 320 picture elements (PELs), with each PEL being 2.4-high by 1-wide

- Preselects one of four colors for each PEL

- Requires 16,000 bytes of read/write memory (on the controller)

- Uses memory-mapped graphics

- Formats four PELs for each byte as follows:

7 6	5 4	3 2	1 0
C1 C0	C1 C0	C1 C0	C1 C0
First	Second	Third	Fourth
Display	Display	Display	Display
PEL	PEL	PEL	PEL

- Organizes graphics storage in two banks of 8000 bytes, using the following format:

Memory Address (in hex)	Function
B9F3F	Even Scans (0,1,4,5,8,9. . .198) 8,000 bytes
B8000	Not Used
BA000	Odd Scans (2,3,6,7,10,11. . .199) 8,000 bytes
BBF3F	Not Used
BBFFF	

Address hex B8000 contains PEL information for the upper-left corner of the display.

- Determines color selection by the following logic:

C1	C0	Function
0	0	Dot takes on the color of 1 of 16 preselected background colors
0	1	Selects first color of preselected Color Set 1 or Color Set 2
1	0	Selects second color of preslelcted Color Set 1 or Color Set 2
1	1	Selects third color of preselected Color Set 1 or Color Set 2

C1 and C0 select 4 to 16 preselected colors. This color selection (palette) is preloaded in an I/O port.

The two color sets are:

Color Set 1	Color Set 2
Color 1 is green	Color 1 is cyan
Color 2 is red	Color 2 is magenta
Color 3 is brown	Color 3 is white

640-by-200 Black-and-White Graphics Mode

The 640-by-200 black-and-white graphics mode supports color monitors. This mode:

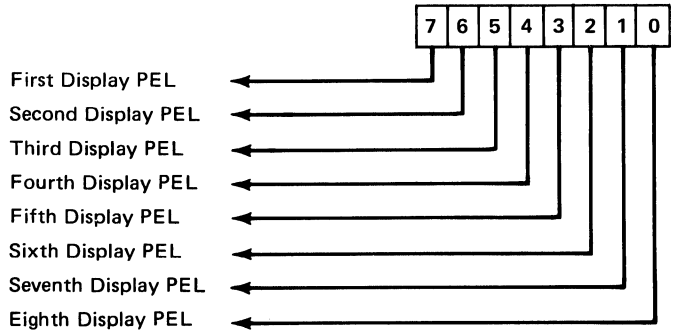
- Contains a maximum of 200 rows of 640 PELs, with each PEL being 1-high by 1-wide.

- Supports black-and-white mode only.

- Requires 16,000 bytes of read/write memory (on the controller).

- Uses the same addressing and mapping procedures as the 320-by-200 color/graphics mode, but the data format is different. In this mode, each bit in memory is mapped to a PEL on the screen.

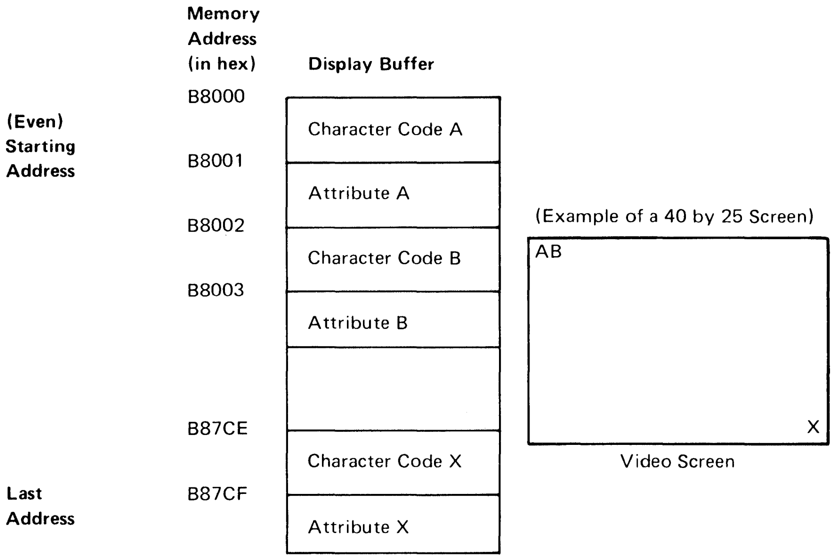
- Formats eight PELs per byte as follows:



August 15,1984

Description of Basic Operations

In the alphanumeric mode, the controller fetches character and attribute information from its display buffer. The starting address of the display buffer is programmable through the 8088 Microprocessor, but it must be an even address. The character codes and attributes are then displayed according to their relative positions in the buffer as shown in the following.



The processor and display control unit have equal access to the display buffer during all operating modes except the 640-by-200 alphanumeric mode. During this mode, the processor should have access to the display buffer during the vertical retrace time. If it does not, the display will be affected with random patterns as the processor is using the display buffer. In the alphanumeric mode, the characters are displayed from a prestored ROM character generator that contains the dot patterns of all the displayable characters.

In the graphics mode, the displayed dots and colors (up to 16K bytes) are also fetched from the display buffer.

High-Function Graphics Mode

The Professional Graphics Controller provides high function graphics capability for the PC by processing simple command strings into bit-mapped images in the controller. The Professional Graphics Controller provides both alphanumeric and graphic capabilities.

Alphanumeric Operation

The alphanumeric operation:

- Contains a built-in character font with character enlargement capabilities.
- Uses a smoothing function for enlarged characters.
- Permits characters to be drawn in a foreground color with a transparent background; therefore, whatever is behind the character remains there.
- Contains programmable character fonts accessible through the high-function graphics command set.

> **Note:** The programmable character sets cannot be enlarged.

Graphics Operation

The high-function graphics mode supports the Professional Graphics Display. It has the following features:

- Contains 480 rows of 640 PELs; the PELs are spaced the same distance vertically and horizontally providing the standard 4:3 screen aspect ratio.

- The color of each PEL is selected from a set of 256 colors, which are selected from a palette of 4096 colors.

- Requires 307,200 bytes of read/write memory (on the controller).

  Note: This memory is addressable only through the high-function graphics commands and does not occupy system address space.

- Uses memory-mapped graphics.

- Formats one PEL for each byte.

- Organizes a communications area consisting of a bank of 1000 bytes.

- Color selection is determined by the following logic:

  The display RAM supplies an 8-bit byte that is used as an address to the LUT. This 8-bit address selects one of 256 12-bit words from the LUT. This data provides the color information for each PEL to be sent to the screen. The 12-bit word is divided into three groups of 4-bits: 4 red, 4 green, and 4 blue, as shown in the following table.

4 Bits	4 Bits	4 Bits
Red	Green	Blue
1 PEL		
1 Byte		

Description of Basic Operations

The controller microprocessor interprets high-function graphics commands and translates them into data that is stored in the display memory. The display memory is then scanned 60 times each second. Each byte is then sent to the LUT. Whatever data is in memory is used as an address to the LUT data to determine what is sent to the screen.

Programming Considerations

The Professional Graphics Controller provides the operation of two individual adapters: (1) the Color/Graphics Monitor Adapter and (2) the High-Function Graphics Adapter. The emulation operation and the high-function graphics operation may be individually programmed. High-function graphics commands determine which of the two operations appears on the screen.

Emulator Programming Considerations

The Professional Graphics Controller emulates the 6845 CRT Controller of the Color/Graphics Monitor Adapter.

Programming the 6845 CRT Controller

The CRT Controller has 19 accessible internal registers, which are used to define and control a raster-scan CRT display. One of these registers, the index register, is actually used as a pointer to the other 18 registers. It is a write-only register, and is loaded from the processor by executing an Out instruction to I/O address hex 3D4. The five least-significant bits of the I/O bus are loaded into the index register.

To load any of the other 18 registers, the index register is first loaded with the necessary pointer; then the data register is loaded with the information to be placed in the selected register. The data register is loaded from the processor by an Out instruction to I/O address hex 3D5.

The following table defines the values that must be loaded into the 6845 CRT Controller registers to control the different modes of operation supported by the controller.

Address Register	Register Number	Register Type	Units	I/O	40 by 25 Alpha-numeric	80 by 25 Alpha-numeric	Graphic Modes
4	R4	Vertical Total	Character Row	Write Only	1F	1F	1F
5	R5	Vertical Total Adjust	Scan Line	Write Only	06	06	06
6	R6	Vertical Displayed	Character Row	Write Only	19	19	19
7	R7	Vertical Sync Position	Character Row	Write Only	1C	1C	1C
A	R10	Cursor Start	Scan Line	Write Only	06	06	06
B	R11	Cursor End	Scan Line	Write Only	07	07	07
C	R12	Start Address(H)	-	Write Only	00	00	00
D	R13	Start Address(L)	-	Write Only	00	00	00
E	R14	Cursor Address(H)	-	Read/ Write	XX	XX	XX
F	R15	Cursor Address(L)	-	Read/ Write	XX	XX	XX
Note: All register values are in hexadecimal							

Programming the Mode Control and Status Registers

The following shows the I/O registers of the Professional Graphics Controller.

Function of Register	Hex Address	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Mode Control Register (DO)	3D8	1	1	1	1	0	1	1	0	0	0
Color Select Register (DO)	3D9	1	1	1	1	0	1	1	0	0	1
Status Register (D1)	3DA	1	1	1	1	0	1	1	0	1	0
6845 Index Register	3D4	1	1	1	1	0	1	0	1	0	0
6845 Data Register	3D5	1	1	1	1	0	1	0	1	0	1

Color-Select Register

This is a 6-bit, output-only register (cannot be read). Its I/O address is hex 3D9, and it can be written to by using the 8088 Microprocessor's I/O Out command. Following is a description of the bits of the color-select register.

Bit 0	Selects B (blue) background color in 320 x 200 graphics mode Selects B (blue) foreground color in 640 x 200 graphics mode
Bit 1	Selects G (green) background color in 320 x 200 graphics mode Selects G (green) foreground color in 640 x 200 graphics mode
Bit 2	Selects R (red) background color in 320 x 200 graphics mode Selects R (red) foreground color in 640 x 200 graphics mode
Bit 3	Selects I (intensified) background color in 320 x 200 graphics mode Selects I (intensified) foreground color in 640 x 200 graphics mode
Bit 4	Selects alternate, intensified set of colors in graphics mode
Bit 5	Selects active color set in graphics mode
Bit 6	Not used
Bit 7	Not used

Bits 0, 1, 2, 3 Select the foreground color in the 640-by-200 color/graphics mode, and the background color (C0 or C1) in the 320 by 200 color/graphics mode.

Bit 4 When set, selects an alternate, intensified set of colors.

Bit 5 Used in the 320 by 200 color/graphics mode to select the active set of screen colors for the display.

When bit 5 is set to 0, colors are determined as follows:

C1	C0	Colors Selected
0	0	Background (Defined by bits 0-3 of port hex 3D9)
0	1	Green
1	0	Red
1	1	Brown

When bit 5 is set to 1, colors are determined as follows:

C1	C0	Colors Selected
0	0	Background (Defined by bits 0-3 of port hex 3D9)
0	1	Cyan
1	0	Magenta
1	1	White

When bit 5 is set to 0 and bit 2 of the mode-select register is set to 1, colors are determined as follows:

C1	C0	Colors Selected
0	0	Background
0	1	Cyan
1	0	Red
1	1	White

Mode-Select Register

This is a 6-bit, output-only register (cannot be read). Its I/O address is hex 3D8, and it can be written to using the 8088 Microprocessor's I/O Out command.

The following table is a description of the register's functions when the bit values are set to 1.

Bit 0	80 x 25 alphanumeric mode
Bit 1	Graphics select
Bit 2	Black/white select
Bit 3	Enable video signal
Bit 4	640 x 200 black/white mode
Bit 5	Change background intensity to blink bit
Bit 6	Not used
Bit 7	Not used

Bit 0 A 1 selects 80-by-25 alphanumeric mode.
A 0 selects 40-by-25 alphanumeric mode.

Bit 1 A 1 selects graphics mode.
A 0 selects alphanumeric mode.

Bit 2 A 1 selects black-and-white mode.
A 0 selects color mode.

Bit 3 A 1 enables the video signal at certain times when modes are being changed. The video signal should be disabled when changing modes.

Bit 4 A 1 selects the 640-by-200 mode black-and-white graphics mode. One of 8 colors can be selected on direct-drive sets in this mode by using register hex 3D9.

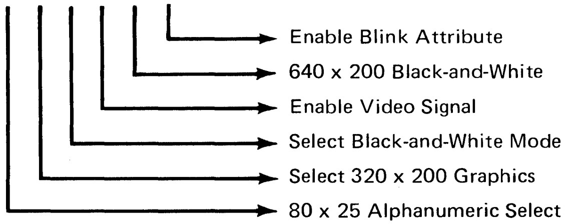
Bit 5 When on (set to 1), this bit changes the character background intensity to the blinking attribute function for alphanumeric modes. When the high-order attribute bit is not selected, 16 background colors (or intensified colors) are available. For normal operation, this bit should be set to 1 to allow the blinking function.

Mode-Select Register Summary

The following table shows the mode-select registers.

Bits

0	1	2	3	4	5	
0	0	1	1	0	1	40 x 25 Alphanumeric Black-and-White
0	0	0	1	0	1	40 x 25 Alphanumeric Color
1	0	1	1	0	1	80 x 25 Alphanumeric Black-and-White
1	0	0	1	0	1	80 x 25 Alphanumeric Color
0	1	1	1	0	z	320 x 200 Black-and-White Graphics
0	1	0	1	0	z	320 x 200 Color Graphics
0	1	1	1	1	z	640 x 200 Black-and-White Graphics



Enable Blink Attribute

640 x 200 Black-and-White

Enable Video Signal

Select Black-and-White Mode

Select 320 x 200 Graphics

80 x 25 Alphanumeric Select

z = Don't care condition

Status Register

The status register is a 4-bit, read-only register. Its I/O address is hex 3DA, and it can be read using the 8088 Microprocessor's I/O In command. The following table is a description of the register functions.

Bit 0	Display Enable
Bit 1	Reserved
Bit 2	Reserved
Bit 3	Vertical Sync
Bit 4	Not Used
Bit 5	Not Used
Bit 6	Not Used
Bit 7	Not Used

Bit 0 When set to 1, indicates that access to the regen buffer memory can be made without interfering with the display.

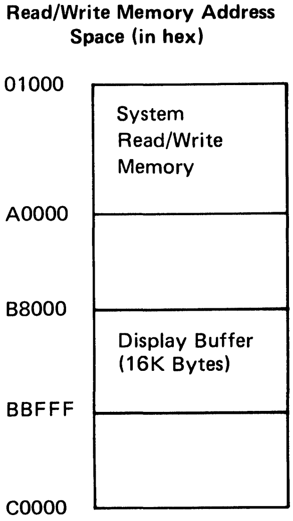
Bit 3 When set to 1, indicates that the raster is in a vertical retrace mode. This is a good time to update the screen buffer.

Sequence of Events for Changing Modes

1. Determine the mode of operation.
2. Reset the video enable bit in the mode-select register.
3. Program the CRT Controller to select the mode.
4. Program the mode- and color-select registers, including re-enabling video.

Memory Requirements

The memory used by this controller is provided entirely on-board. It consists of 16K bytes without parity. This memory is used as both a display buffer for alphanumeric data and as a bit map for graphics data. The regen buffer's address starts at hex B8000. The following table shows the memory requirements.

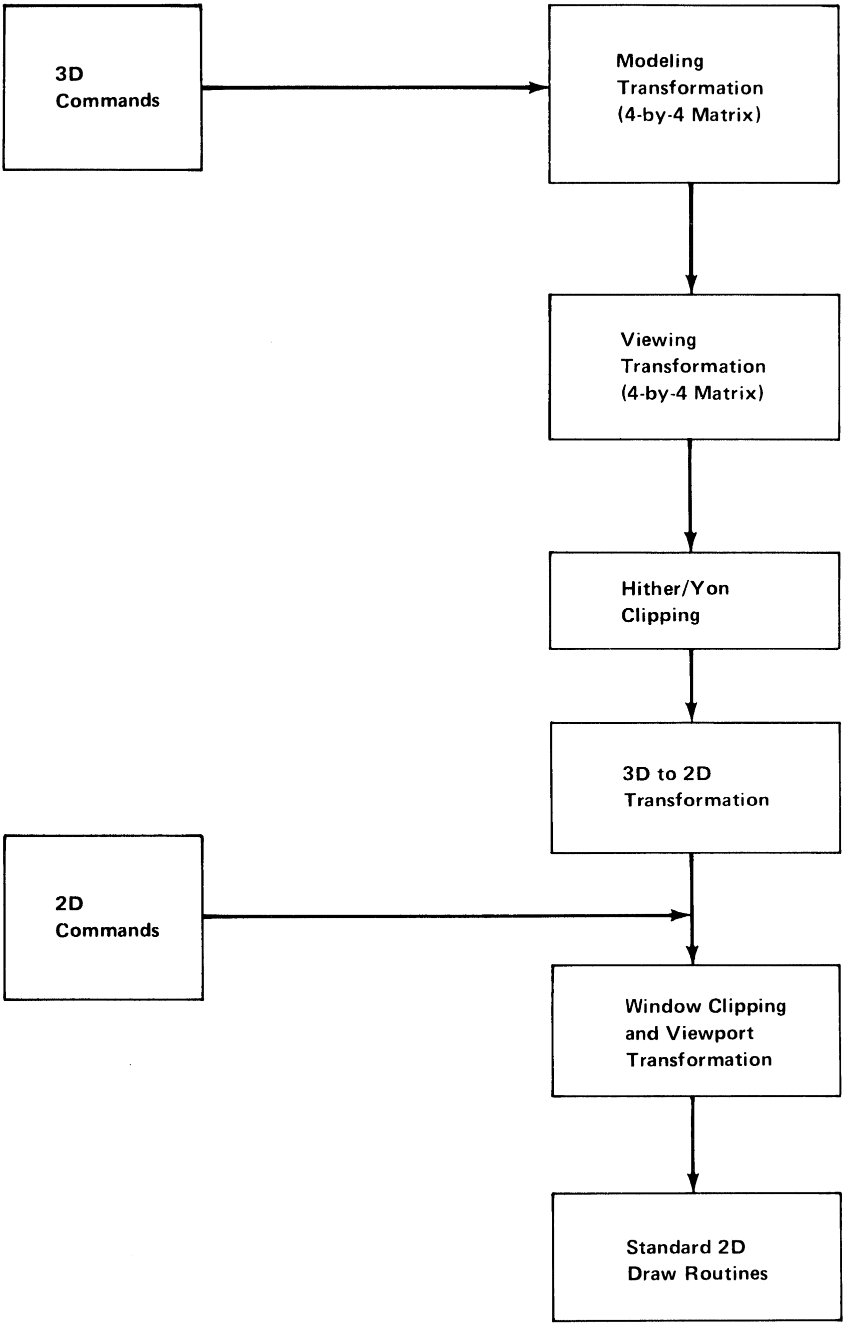

Read/Write Memory Address Space (in hex)
01000
System Read/Write Memory
A0000
B8000
Display Buffer (16K Bytes)
BBFFF
C0000

High-Function Graphics Programming Considerations

The high-function graphics command set uses a wide range of two-dimensional and three-dimensional programs that include:

- Drawing primitives with points, vectors, and polygons in two and three dimensions
- Coordinate transformations with modeling (scaling, rotation, translation) and viewing transformations
- Drawing primitives with rectangles, circles, ellipses, arcs, and sectors in two dimensions
- Stored segments that define and execute command lists
- Color control functions
- Text generation

Following is a flowchart of the two- and three-dimensional commands.


3D Commands
Modeling Transformation (4-by-4 Matrix)
Viewing Transformation (4-by-4 Matrix)
Hither/Yon Clipping
3D to 2D Transformation
2D Commands
Window Clipping and Viewport Transformation
Standard 2D Draw Routines

Objects may be defined in three dimensions using the three-dimensional drawing commands. A modeling matrix allows the object to be moved (translated), changed in size (scaled), and rotated. A viewing matrix allows the object to be viewed from different directions and distances.

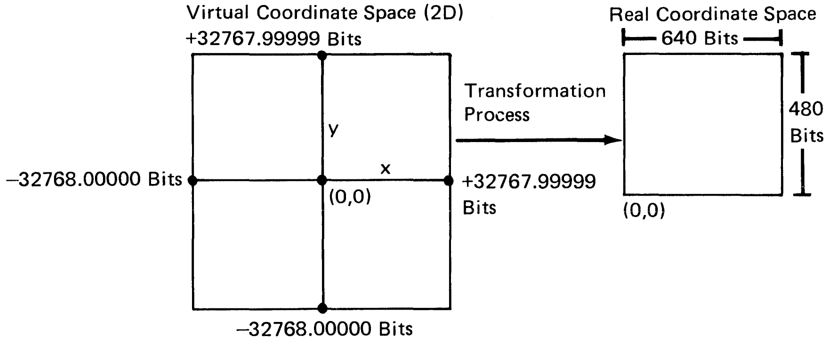
Two clipping planes are defined at right angles to the line-of-sight. Any part of an object beyond the *yon* clipping plane and any part of an object in front of the *hither* clipping plane are not seen.

Three-dimensional objects are projected onto a two-dimensional *viewplane*, which is the plane of the monitor's screen. Two-dimensional objects are defined directly on the viewplane. Coordinates on the viewplane are referred to as *virtual* coordinates. A *window* defines that area of the viewplane that is visible. Any part of an object outside the defined window is not seen. A *viewport* specifies a rectangular area on the monitor's screen that completely contains the defined window.

Coordinate Space

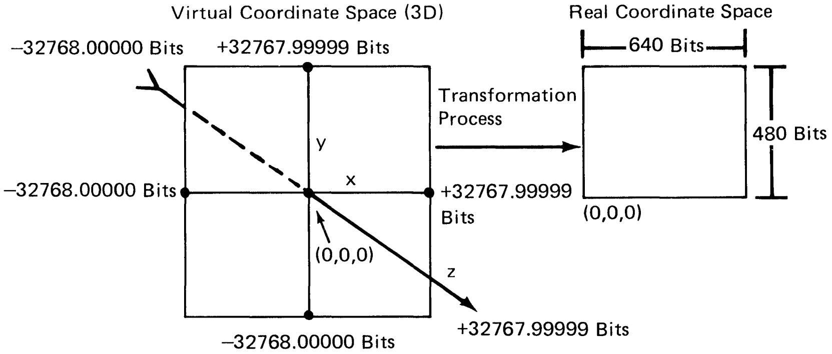
Two-dimensional commands operate on a virtual coordinate space whose x and y boundaries range from -32768.00000 bits to +32767.99999 bits, with 16 bits of precision to the right of the decimal point. The display screen, however, is 640 PELs wide by 480 high. Therefore, commands are available to specify how coordinates are converted from virtual values to screen values. In addition, portions of the physical screen may be declared "off limits" to drawing. This is accomplished through the command VWPORT, which defines a rectangular clipping viewport.

The following figure shows the relationship of two-dimensional virtual coordinate space to real coordinate space.



Three-dimensional drawing commands operate in a virtual coordinate space whose x and y boundaries range from -32768.00000 bits to +32767.99999 bits, but a z coordinate is added, which may have any value in the same range as x and y. All three-dimensional drawing may be divided into a series of points and lines; these points and lines are what are mapped onto the two-dimensional plane for actual writing to the display.

The following figure shows the relationship of three-dimensional virtual coordinate space to real coordinate space.



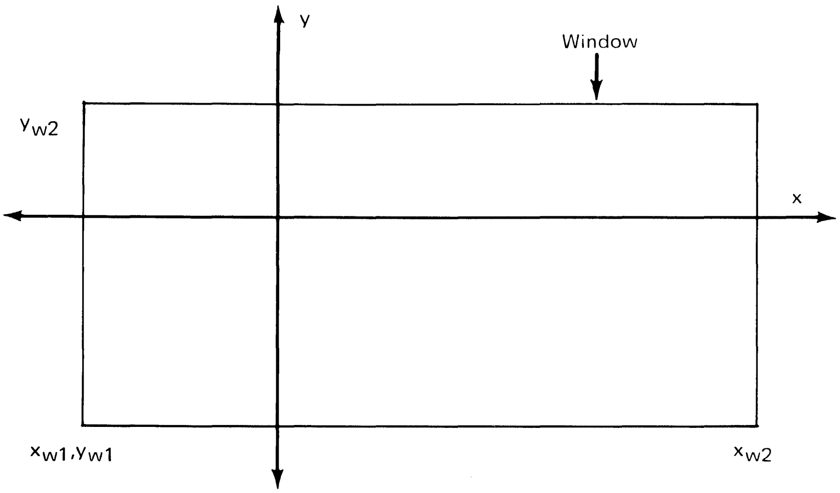
Coordinate Transformations

The high-function graphics mode refers to four coordinate systems when converting three-dimensional virtual coordinates to a screen image. The two-dimensional commands MOVE and DRAW undergo a single transformation.

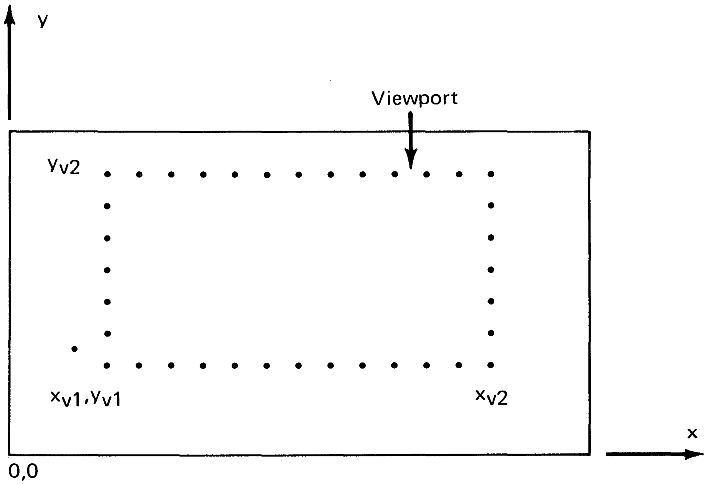
Two-Dimensional Transformation

The lowest level of transformation occurs following the two-dimensional command MOVE or DRAW. These commands use parameters given in two-dimensional virtual coordinates. The high-function graphics mode converts these points to screen coordinates. To understand this conversion, keep in mind that the window in two-dimensional virtual space maps onto the viewport of the screen.

The WINDOW command defines an area (window) in two-dimensional virtual space to be mapped into a defined viewport with x and y virtual coordinate values, as follows:



The x and y values may range from -32768.00000 to +32767.99999. The VWPORT command defines an area (viewport) within the display screen with x and y screen coordinate values, as shown in the following.



The x values range from 0 to 639, and the y values from 0 to 479. The two-dimensional command uses virtual coordinates; that is, X2dvir and Y2dvir. The high-function graphics mode converts these to screen coordinates, Xscrn and Yscrn, using the following equations.

$$\mathrm{Xscrn} = (\mathrm{X2dvir} - \mathrm{Xw1}) \times \frac{(\mathrm{Xv2} - \mathrm{Xv1})}{(\mathrm{Xw2} - \mathrm{Xw1})} + \mathrm{Xv1}$$

$$\mathrm{Yscrn} = (\mathrm{Y2dvir} - \mathrm{Yw1}) \times \frac{(\mathrm{Yv2} - \mathrm{Yv1})}{(\mathrm{Yw2} - \mathrm{Yw1})} + \mathrm{Yv1}$$

The X2dvir, Y2dvir are two-dimensional virtual coordinates. The variables Xw1, Xw2, Yw1, and Yw2 are window coordinates, and Xv1, Xv2, Yv1, and Yv2 are viewport coordinates.

Three-Dimensional Transformation

Three-dimensional transformations involve converting three-dimensional points to two dimensions. This process uses the following matrix operation for the conversion; that is three-dimensional world coordinates to three-dimensional viewing coordinates:

```
[Xview, Yview, Zview, 1 ] =
[Xvirtual, Yvirtual, Zvirtual, 1] x [M] x [VRP] x [V]
```

[M] represents the modeling matrix, [VRP] represents the view reference point matrix, and [V] denotes the viewing matrix. The three-dimensional viewing coordinates can be read back using the command FLAGRD 24. The last value of the viewing matrix remains 1 only if the last columns of all matrixes entered in this formula have the following form.

$$\begin{vmatrix} X & X & X & 0 \\ X & X & X & 0 \\ X & X & X & 0 \\ X & X & X & 1 \end{vmatrix}$$

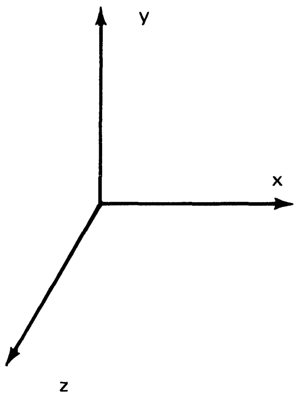
Otherwise, the result will have the form:

```
[Xview, Yview, Zview, Q]
```

To reduce this result to the form required, simply divide the X, Y, and Z values by the value Q. This operation gives a 1 as the final column value of the matrix, and proper values for the other three parameters.

The Modeling Matrix

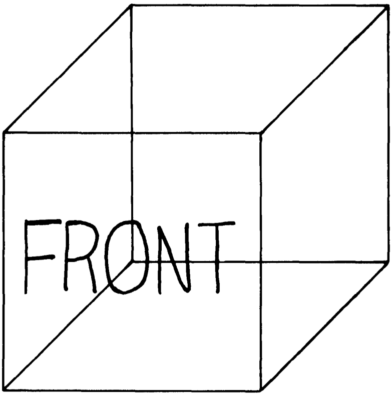
The modeling matrix, [M], rotates, translates, and scales the coordinate values of an object defined in three-dimensional virtual coordinates. Rotation about any axis uses the right-hand rule. To understand this principle, refer to the coordinate space depicted below (the positive z direction comes out of the page).



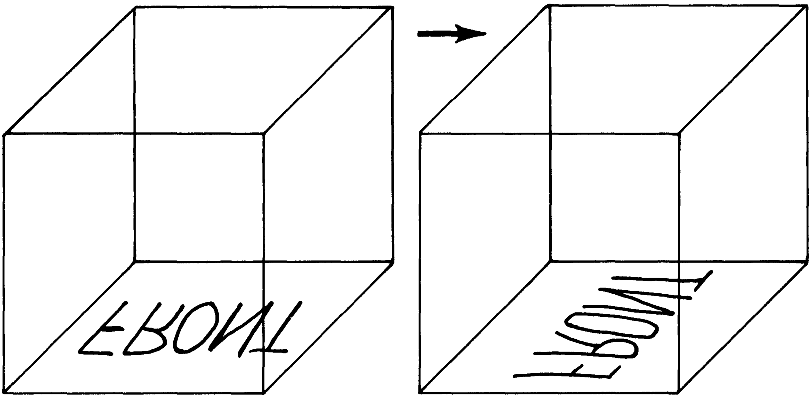
To rotate in a positive direction around the y axis, the positive z axis rotates toward the positive x axis. To rotate in a positive direction around the x axis, the positive y axis rotates toward the positive z axis. To rotate in a positive direction around the z axis, the positive x axis rotates toward the positive y axis.

Keep in mind that the order of rotation changes the viewing faces of the object. That is, an object rotated along the x axis, then the y axis, gives a different perspective than if the same object is rotated first along the y axis, then the x axis.

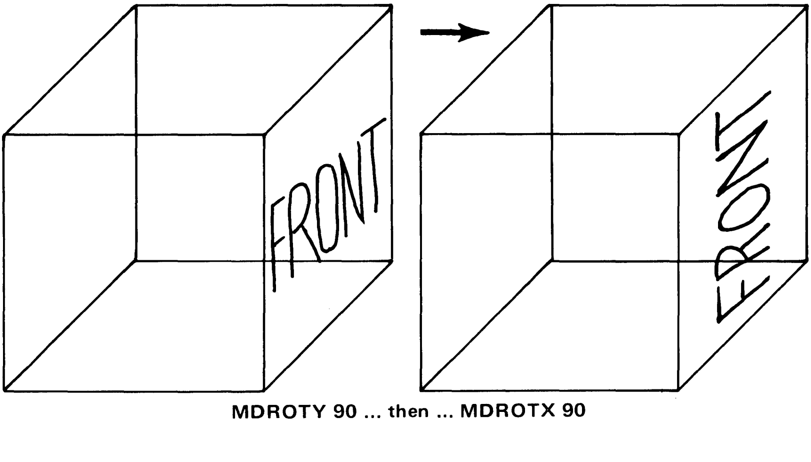
The following illustration depicts various viewing perspectives.



Original



MDROTX 90 ... then ... MDROTY 90



MDROTY 90 ... then ... MDROTX 90

Rotation involves the matrix operation,

```
[M(new)] = [M(old)] x [M(rst)]
```

[M(rst)] represents the rotation, scaling, or translation matrix. For rotation, this matrix differs with each axis chosen as the axis of rotation. For each direction of rotation, the algorithm refers to the appropriate matrix as follows:

$$R_x(\theta) = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$R_y(\theta) \begin{vmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$R_z(\theta) = \begin{vmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

The scaling operation uses the following matrix.

$$S = \begin{vmatrix} x_s & 0 & 0 & 0 \\ 0 & y_s & 0 & 0 \\ 0 & 0 & z_s & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

The translation operation uses the following matrix.

$$T = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_t & y_t & z_t & 1 \end{vmatrix}$$

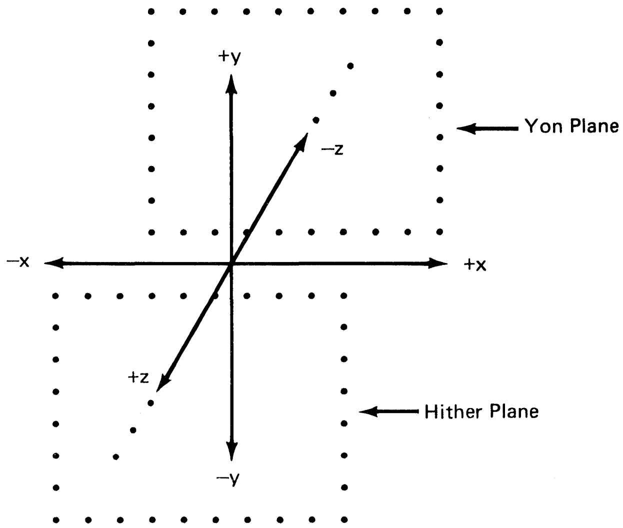
Viewer Reference-Point Matrix

The viewer reference-point matrix, [VRP], translates the point viewed by the user to the center of the currently defined window. Because the window coordinates map onto the viewport coordinates, this matrix also places the user-viewed point at the center of the viewport.

The viewing matrix, [V], affects the degree of rotation of the object by moving the eye about the object, while keeping the object stationary. Like the modeling matrix, the viewing matrix uses the right-hand rule for rotation of the eye about the viewing reference point.

Three-Dimensional Hither and Yon Clipping

Besides two-dimensional viewport clipping, the high-function graphics mode also clips in the third dimension. The hither and yon clipping designate two x-y planes along the z axis beyond which no drawing takes place.



Three-Dimensional Viewing to Two-Dimensional Virtual Projection

Using the DISTAN command, the user specifies the distance from the eye to the viewplane. The command PROJCT provides a viewing angle with a value ranging from 1 to 179 degrees. The high-function graphics mode projects the viewing coordinate into a two-dimensional coordinate value using the following formulas.

$$\mathrm{X2dvir} = \frac{\mathrm{DISTAN}}{\mathrm{DISTAN} - \mathrm{Z}} \times \mathrm{Xview} \times \frac{\mathrm{WINDOW\ DIAGONAL}}{2 \times \mathrm{DISTAN} \times \tan\left(\frac{\mathrm{PROJCT}}{2}\right)}$$

$$\mathrm{Y2dvir} = \frac{\mathrm{DISTAN}}{\mathrm{DISTAN} - \mathrm{Z}} \times \mathrm{Yview} \times \frac{\mathrm{WINDOW\ DIAGONAL}}{2 \times \mathrm{DISTAN} \times \tan\left(\frac{\mathrm{PROJCT}}{2}\right)}$$

Placing the object closer magnifies the X and Y values. Increasing the viewing angle increases the amount of picture visible in the viewing field.

If the PROJCT angle is 0, the projection is orthographic parallel (non-oblique), The high-function graphics mode projects the viewing coordinate into a two-dimensional coordinate value using the following formulas:

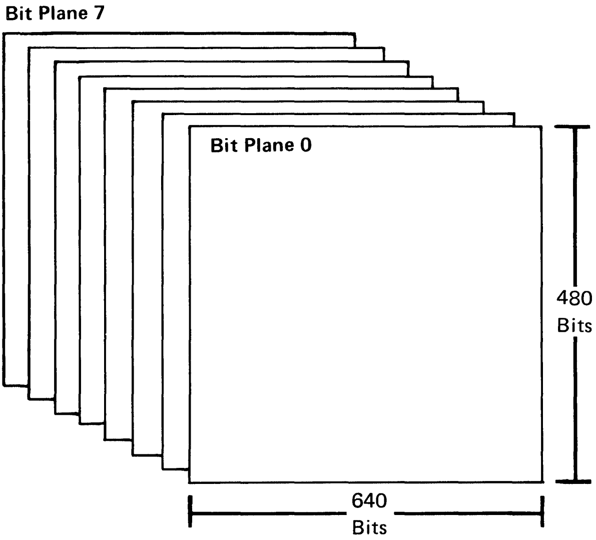
$$\mathrm{X2dvir} = \mathrm{Xview}$$

$$\mathrm{Y2dvir} = \mathrm{Yview}$$

Video Generation

A total of 256 colors may be displayed on the screen at one time. A total of 4096 possible color selections is available to the LUTs. The video generation process begins when the video scanner reads the value of the PEL about to be displayed. The PEL value consists of eight bits and is used as an address to the LUT. The PEL value selects one of 256 12-bit entries in the table. The three 4-bit output values from the LUT represent the red, green, and blue intensities required to compose the target PEL. Because the table outputs are 4 bits each for the three colors, the 256 simultaneous colors may be chosen from a 4096-color palette. The LUTINT command sets the entire look-up table from one of several predefined LUT selections. The LUT command loads individual LUT entries, and LUTRD reads them back.

Each bit of each PEL resides in one of eight bit planes in the display memory. The bit planes are masked for reading and writing. These bit planes are shown in the following.



Current Point

The *current point* is the x-y-z coordinate point at which the last command finished. Many high-function graphics commands use a current point in carrying out their functions. Two current points are maintained; one is used by two-dimensional commands, the other by three-dimensional commands. For example, the two-dimensional command CIRCLE draws a circle centered on the two-dimensional current point; the three-dimensional command DRAW3 draws a vector that starts at the three-dimensional current point. The current points are moved whenever move and draw commands are executed. When referred to in the command descriptions, the applicable current point will be identified, unless it is clear from the context of the command.

The command CONVRT will change a three-dimensional current point to a two-dimensional virtual coordinate. This conversion allows the user to overdraw a three-dimensional drawing with two-dimensional commands, such as text.

Current Color

The *current color* is the last color a COLOR command defines for general drawing. Drawing is possible in two modes—the complement drawing mode and the replace drawing mode. In the complement drawing mode, the PEL bit value in display RAM is complemented from its current value. In the replace drawing mode, the PEL bit value in display RAM is changed to a specified value. The value comes from the current color, which is set by using the COLOR command.

> **Note:** In both cases, the actual value written into a PEL may be affected by a mask.

Display Control

Display control commands set or reset flags or define commonly used parameters. All these commands affect the way that later commands draw to the screen.

Drawing Modes

The high-function graphics mode provides several drawing modes. It has its own language. The Professional Graphics Controller also imitates two current graphics modes resident in the existing PC graphics systems. The Professional Graphics Controller will accept and execute all commands sent to either mode. To view the current status of commands sent to a particular mode, use the DISPLA command, indicating the appropriate mode as the parameter. This command simply switches between the high-function graphics screen and the emulator screen. All previous drawing sent to either screen remains intact during these switches, because Draw commands are independent of the viewing status; that is, high-function graphics commands affect the high-function graphics screen even while the emulator screen is displayed.

Primitive Fills and Drawing Patterns

The command PRMFIL sets an on/off flag to fill the commands that draw defined geometric shapes and create an enclosed area. Each command description will note the effects of any flags.

The user can change the drawing pattern by using Pattern commands. The command LINPAT governs any vector or other command drawing a geometric shape (with PRMFIL off). The parameter, a 16-bit number, acts as a mask during drawing. Each bit sets an on/off pattern for a corresponding PEL on the screen. This pattern repeats every 16 PELs. A 1 in any bit position draws a PEL, while a 0 changes nothing. The value 65535 produces a solid line.

Similarly, the command AREAPT establishes a drawing pattern for an area using a 16-bit by 16-bit format. This command repeats in blocks of 16-by-16 PELs, duplicating the pattern in both a horizontal and vertical direction. To define a pattern, enter sixteen 16-bit words, visualizing their orientation on a grid. For example:

Word Order	Pattern																Bit Number
F	X	X	X	X			X	X	X	X			X	X	X	X	62415
E		X	X	X	X			X	X	X	X			X	X	X	31207
D			X	X	X	X			X	X	X	X			X	X	15603
C	X			X	X	X	X			X	X	X	X			X	40569
B	X	X			X	X	X	X			X	X	X	X			53057
A	X	X	X			X	X	X	X			X	X	X	X		59294
9	X	X	X	X			X	X	X	X			X	X	X	X	62415
8		X	X	X	X			X	X	X	X			X	X	X	31207
7			X	X	X	X			X	X	X	X			X	X	15603
6	X			X	X	X	X			X	X	X	X			X	40569
5	X	X			X	X	X	X			X	X	X	X			53057
4	X	X	X			X	X	X	X			X	X	X	X		59294
3	X	X	X	X			X	X	X	X			X	X	X	X	62415
2		X	X	X	X			X	X	X	X			X	X	X	31207
1			X	X	X	X			X	X	X	X			X	X	15603
0	X			X	X	X	X			X	X	X	X			X	40569
	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	

Each word, then, would equal the decimal equivalent of the 16-bit number. For this example, use 40569 for word 0, 15603 for word 1, and so on. In hexadecimal mode, these same words should read 9E79 for word 0, 3CF3 for word 1, and so on.

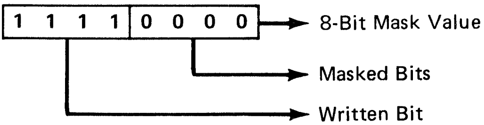
Masks

Masks act as an overlay to either reveal or overwrite the bits of a PEL. In reference to bit planes, the mask can effectively separate planes and protect certain ones. Masks affect only read and write operations but do not affect the displayed PELs.

Bit Planes

The number of bits used to define the colors of a graphics system also defines the number of bit planes. Masks control the CPU reads and writes. By using LUT entries, the user can designate which bits will actually draw to the screen. This capability effectively produces backgrounds. For example, if a mask hides the first four bits of all color values, the system draws colors using only the last four bits. Colors defined using the first four bits can be protected by suitably setting the LUTs. Switching among more than one LUT can produce animation.

The following mask writes only PELs whose color-values (indexes) are given as x0H, where x can equal 0 to F.



Color values such as 19H and B4H will write as 1xH and BxH respectively, where x leaves any previous draw untouched.

Area Pattern Mask

The command FILMSK affects the two Area Fill commands. The 8-bit value of FILMSK is ANDed with the value of MASK and with each PEL value read in an Area Fill command. The high-function graphics mode then compares the ANDed value to the boundary color.

Clipping

The high-function graphics mode describes a clipping window and a set of clipping planes. Both the VWPORT and WINDOW command define a clipping border, for the screen and two-dimensional virtual space, respectively. The clipping window can change to include more or less of the image in two-dimensional virtual space. The viewport clipping window defines the area on the screen that is to contain the image. Redefining the coordinates of the viewport allows several clipped images to appear on the screen simultaneously.

In three-dimension, the high-function graphics mode adds hither and yon clipping capabilities. The previously defined clipping window projects forward and backward to define a clipping space. The high-function graphics mode calculates all intersecting clipping planes.

Viewing

Viewing involves selecting a viewing distance with the command DISTAN and a viewing angle with the command PROJCT.

WAIT

The command WAIT causes the system to pause for a specified number of frame scan cycles. An imbedded Wait command will hold the drawn image on the screen for a specified amount of time before continuing with the program. The Wait command bases its timing on frame time, which equals 1/60 of a second. Use this value to calculate the actual wait period. For example, specifying 300 frame times would give a wait period of 5 seconds.

Drawing Primitives

The term *drawing primitives* defines a group of commands that draw defined geometric shapes. The user specifies size and position with the parameters associated with each command.

Two-Dimensional and Three-Dimensional Command Format

Two-dimensional commands use no numbers within the 6-letter command. All three-dimensional commands end in the numeral 3. Coordinates for two-dimensional commands require one variable each for the x and y values; the three-dimensional commands require three coordinate values (one each for the x, y, and z direction). Not all two-dimensional Draw commands have a three-dimensional counterpart.

Move Commands

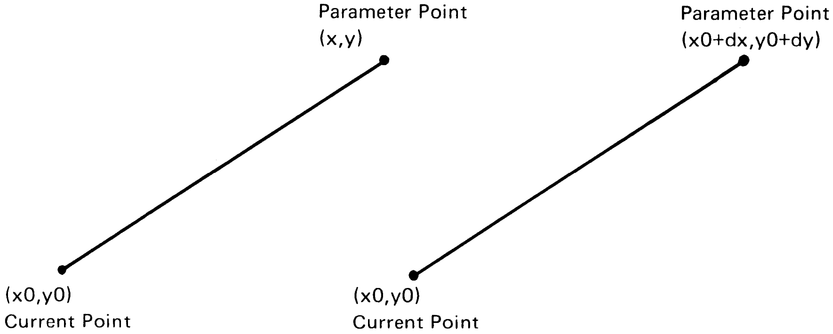
The Move commands change the current point in either the two-dimensional or three-dimensional coordinate space, one current point for each space. The commands MOVE and MOVE3 specify a change using absolute coordinate values. These commands use the virtual coordinate systems. MOVER and MOVER3 change the current point by a relative amount, adding the parameter values to the current point to produce a new coordinate value as the current point.

Point

The Point command changes the PEL at the current point to the current color.

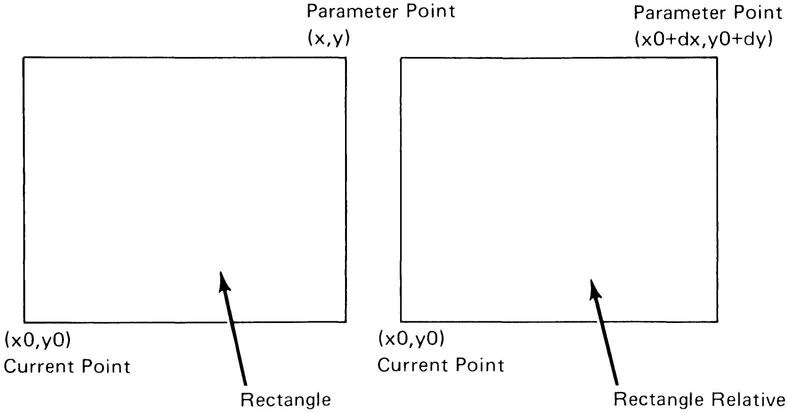
Vectors

Draw commands produce vectors (directed line segments) between two specified points. The current-point value supplies the first coordinate. The high-function graphics mode then draws a vector ending at the absolute coordinate values given in a DRAW or DRAW3 command or at the relative distance specified by the parameters of a DRAWR or a DRAWR3 commands. After a vector command, the current point shifts to the location of the last PEL drawn. The following examples show vectors.



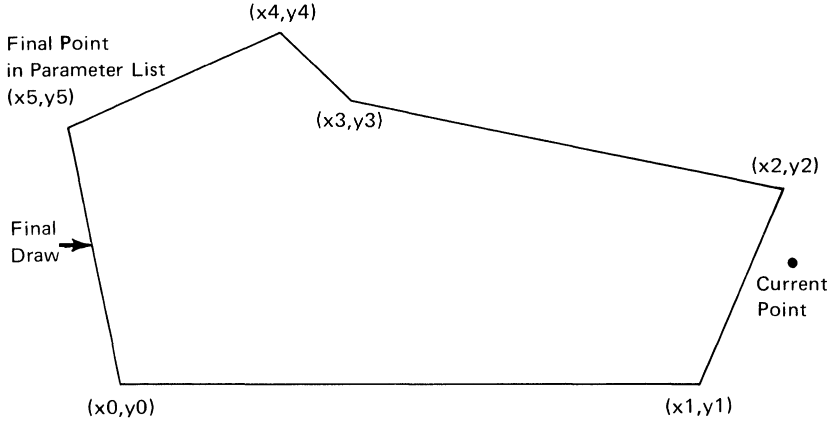
Linear Forms

The high-function graphics mode produces two closed linear forms: rectangles and polygons. Two points define a rectangle. The current point is one corner of the shape. The parameters, given in absolute values (RECT) or in a relative, offset distance (RECTR), specify the opposite corner. The current point does not change for any rectangle command. Rectangles are specified only in two dimensions. The following example shows rectangles:



Each pair of coordinates in a Polygon command declares a vertex of any multisided figure. Two pairs of coordinate values, adjacent within a command's variable string, produce a side between them. The command effectively draws multiple vectors, changing the current point to the location of the last PEL drawn. This pattern continues until a vector has been drawn to the last coordinate. The final draw of the command connects the final coordinates given to the beginning point of the polygon. The current point returns to its original value. Again this command uses either absolute or relative coordinates—POLY or POLYR for two-dimensional, and POLY3 or POLYR3 for three-dimensional. All relative coordinates are expressed relative to the original point. Keep in mind that nonplanar values in three-dimensional polygons may produce undesired effects.

The following is an example of a polygon.



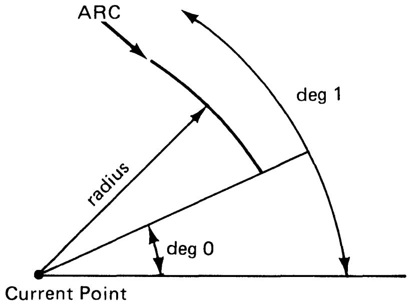
> **Note:** The primitive fill flag in PRMFIL 1 directs the high-function graphics mode to draw any of the above rectangles or polygons as a solid (that is, all enclosed PELs are set to the current color). Undesirable effects may occur if the filled polygon intersects itself.

Nonlinear Forms

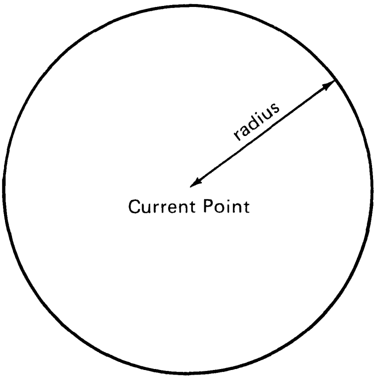
The high-function graphics mode also produces some nonlinear geometric shapes. The commands CIRCLE and ELIPSE require only radius values (both an x and y radius value for ELIPSE). The current point specifies the center of both of these figures. The parameters for the command ARC list a radius, a beginning angle value, and an ending angle value. The current point also serves as the center point of rotation for this command. The command SECTOR has the same parameter requirements as an ARC command, but produces a pie-shaped figure. That is, the end-points of the arc connect with vectors to the center point of rotation.

Except when used with the ARC command, a PRMFIL command with the fill flag set on, will instruct the commands to produce solid shapes filled with PELs of the current color. All nonlinear commands draw only in two dimensions.

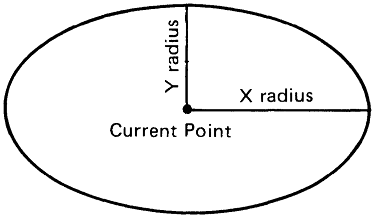
The following illustrations show examples of nonlinear forms.



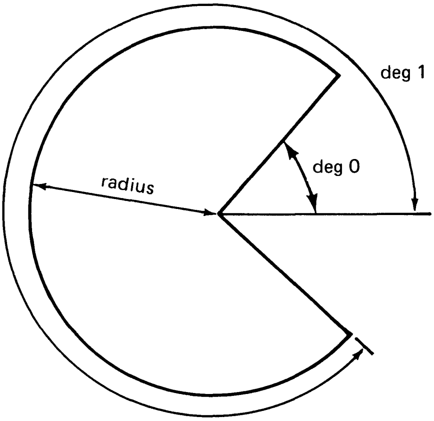
ARC deg 0 deg 1 example



CIRCLE radius example



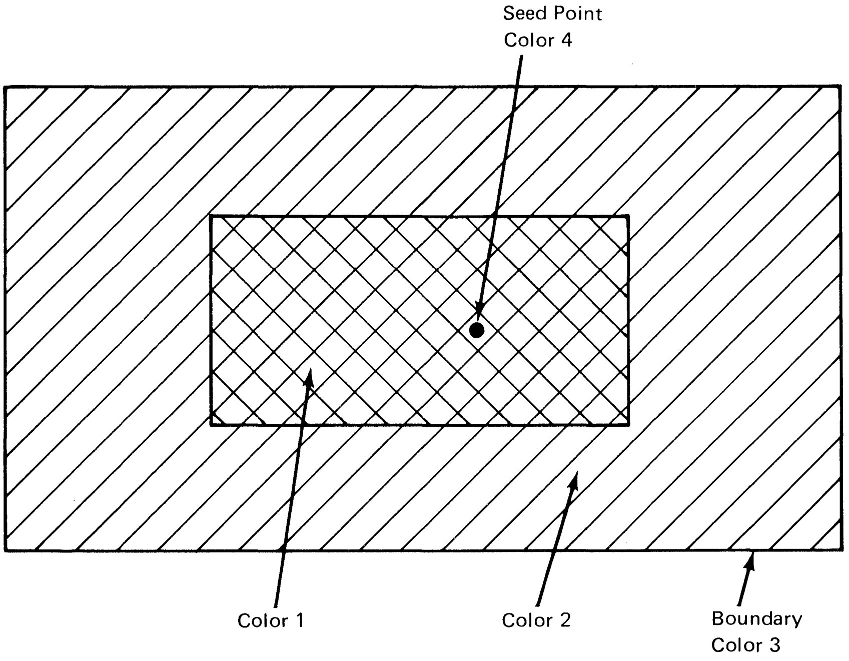
ELIPSE x radius y radius example



SECTOR deg 0 deg 1 example

Area Fills

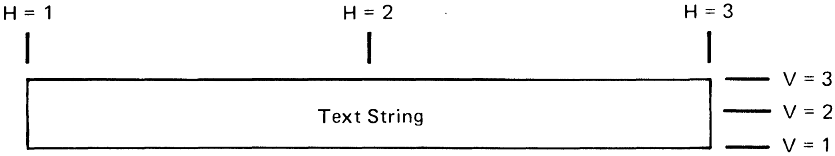
The Area Fill commands employ a *seed* point. Before sending an Area Fill command, place the current point within the area to be filled. The current color must differ from the color being changed. The command AREA changes PELs outward in all directions from the current (seed) point until is encountered a color different from either the one being changed or the current color. The command AREABC allows the user to specify a color to act as a boundary. This command converts PELs from the seed point outward until PELs of the same color as the specified boundary color are encountered. The current color must differ from the boundary color. The following is an Area Fill example.



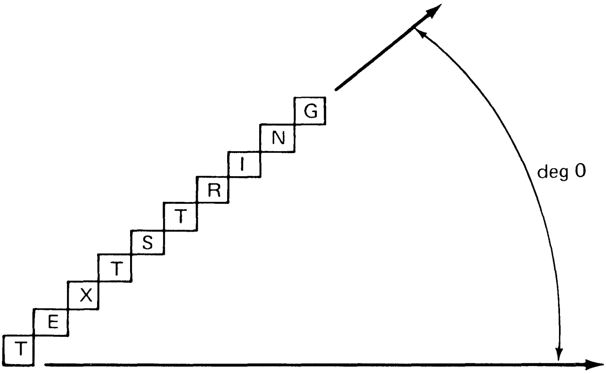
In the Area Fill example, set the current color to color 4. The Area Fill will fill only the area covered by color 1. The Area Boundary Fill specified with the boundary color set to color 3 will fill the area covered by color 1 and color 2.

Text

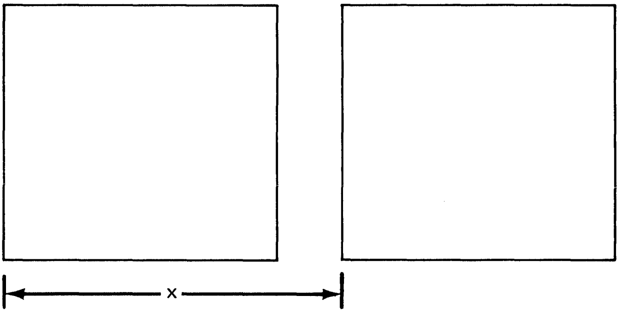
Various Text commands help in placing and moving text. The two-dimensional current point acts as a placement marker. For justifying text, this point defines the horizontal and vertical placement of the text string, using the command TJUST (see the following). The default is H = 1, V = 1.



Altering the angle adjusts the slope of the centering point for each letter but not the rotation of the letter itself. The command TANGLE uses standard Cartesian coordinates to measure the angle, as shown in the following.



To adjust the text size, use the command TSIZE. The parameter of this command specifies a two-dimensional virtual x-distance. Keep in mind that the high-function graphics mode sizes letters using the mapping of the window onto the viewport. For example, a window of 320 PELs by 240 PELs mapped to a viewport of 640 PELs by 480 PELs would draw size 8 letters in a 16-PEL horizontal space. All text that exceeds the viewport boundary undergoes clipping. The default, size 8, writes a character of 7 by 9 PELs in a cell of 8 by 12 PELs using one column for horizontal spacing between letters (see the following).



Use the commands TEXT or TEXTD to write text to the screen. TEXT uses a default text font; TEXTD uses any text defined in the command TDEFIN. This command requires a size specification followed by a bit value to describe each line of blocks. The first step is to outline an area that encompasses the character (see the following).

Line Number 5 →			X	X	X	X	X	
Line Number 4 →	X	X	X					X
Line Number 3 →			X					X
Line Number 2 →	X	X	X					X
Line Number 1 →			X					X
	7	6	5	4	3	2	1	0 ← Bit Number

Then list each bit; start with the bottom, leftmost block and work to the right and up. The command for this character becomes:

TDEFIN 'x' 85
0 0 1 0 0 0 0 1
1 1 1 0 0 0 0 1
0 0 1 0 0 0 0 1
1 1 1 0 0 0 0 1
0 0 1 1 1 1 1 0

Command Lists

Command lists consist of a series of valid high-function graphics commands executed by a single command. The commands CLBEG and CLEND mark the beginning and end of command lists. Two commands begin execution of command lists. CLRUN executes a single command list once; CLOOP executes a single command list a specified number of times. The commands CLDEL and CLBEG delete a command list previously defined by the specified parameter value. Space permitting, the user can define up to 256 command lists. Any command, except CLBEG, may appear within a command list definition. However, during the execution of a command list, the high-function graphics mode will not execute an imbedded CLDEL.

The following examples show valid formats for command lists.

```
CLBEG 8
   CLEARS 0
   MOVE 0 0
   PRMFIL 1
   COLOR 2
   SECTOR 100 60 359
   MOVE 10 10
   COLOR 3
   SECTOR 90 0 59
   CLEND
CLRUN 8
```

```
CLBEG 17
   CLEARS 0
   PRMFIL 1
   MOVER 10 0
   COLOR 2
   CIRCLE 5
   CLEND
CLOOP 17 5
```

Command list 8 will draw two sectors of different colors. Command list 17 will draw a small circle of radius 5. The command CLOOP repeats command list 17 five times, thus drawing five, small, tangential circles.

The following example shows an invalid format for a command list.

```
CLBEG 23
   CLEARS 0
   CLBEG 1
      CIRCLE 25
   CLEND 2
   CLDEL 14
   CLEND
```

Command list 23 is invalid because:

- CLBEG cannot appear within a stream of command list commands.

- If the high-function graphics mode receives CLRUN 23, the execution of CLDEL command would produce an error.

Look-Up Table

The look-up table (LUT) contains the red, green, and blue intensity information associated with each color. A value, or index, identifies each color. The high-function graphics mode provides several default LUT selections, which are accessible with the command LUTINT. The user can change values by using the command LUT or by initializing a new table. The command LUTSAV stores the current LUT values. LUTSAV overwrites any previously saved LUT values. The saved values may be selected by the command LUTINT 255. The following block diagram illustrates LUT generation.

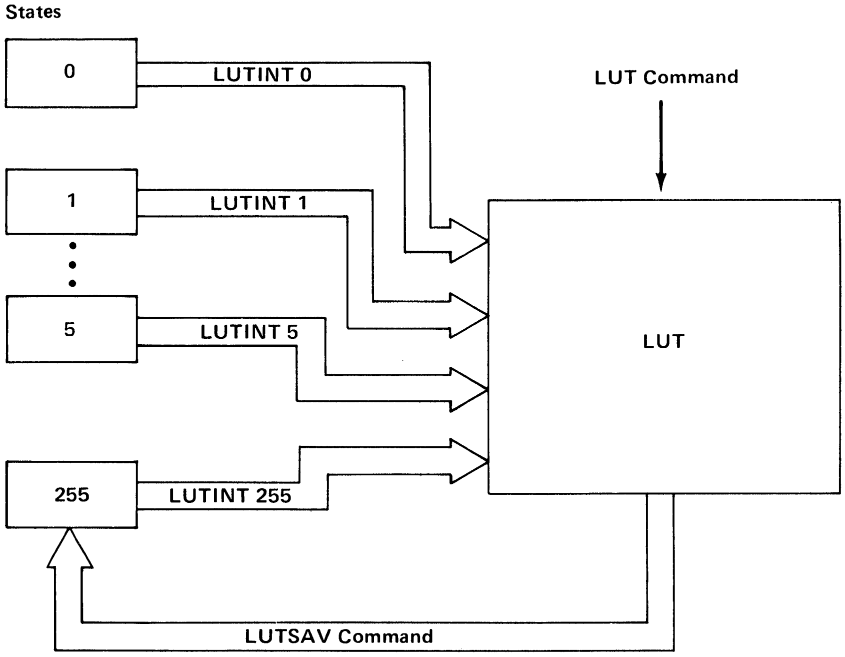
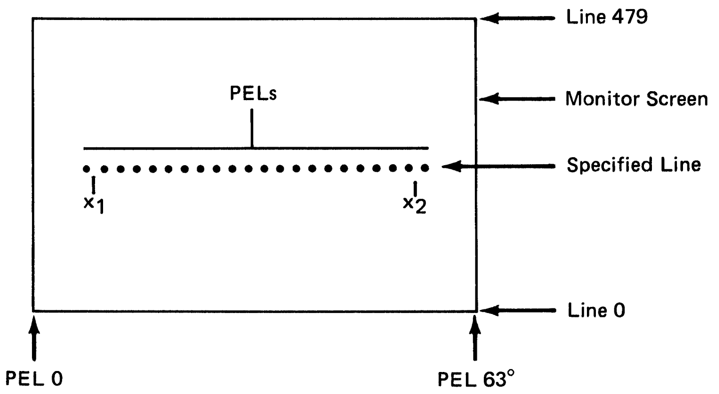


Image Processing

The high-function graphics mode uses limited image-processing techniques. The user can read or write a line of PEL data with variable endpoints. The user specifies a line number and a beginning and ending point within that line. The Image Read command (IMAGER) returns the line data formatted as an Image Write command (IMAGEW). This format makes it easier to use stored image information. The following illustrates image processing.



Read-Back Commands

The high-function graphics mode allows the user to read various parameters from the color board back to the program. Items readable in this way include LUT entries, both three-dimensional transformation matrixes, and the line pattern and line function flags. The read-back protocol is straightforward. When the high-function graphics mode executes one of the read-back commands (for example, FLAGRD), it puts the value of the requested item in the output buffer. In ASCII mode, the value is written as a decimal number followed by a carriage-return character. A high-level language, such as BASIC, need only execute an Input statement to get the data from the color board. Some data read-back commands return more than one value. The individual commands describe the format of the return in both ASCII and hexadecimal communication modes.

The following table lists the flags readable by FLAGRD, and the size and type of the value returned.

Flag	Name	Type of Value Returned
1	AREAPT	16 integers
2	CLIPH	1 integer (byte)
3	CLIPY	1 integer (byte)
4	COLOR	1 integer (byte)
5	DISPLA	1 integer (byte)
6	DISTAN	1 real number
7	DISTH	1 real number
8	DISTY	1 real number
9	FILMSK	1 integer (byte)
10	LINFUN	1 integer (byte)
11	LINPAT	1 integer
12	MASK	1 integer (byte)
13	MDORG	3 real numbers
14	2D current point	2 real numbers
15	3D current point	3 real numbers
16	PRMFIL	1 integer (byte)
17	PROJCT	1 integer (byte)
18	TANGLE	1 word
19	TJUST	2 integers (bytes)
20	TSIZE	1 real number
21	VWPORT	4 integers
22	VWRPT	3 real numbers
23	WINDOW	4 real numbers
24	Transformed 3D current point	3 real numbers
25	Free memory available	1 integer

The command LUTRD reads back the red, green, and blue intensity levels for a particular LUT index. To read back either the viewing matrix [V] specified in the command VWMATX, or the modeling matrix [M] specified in the command MDMATX, use the command MATXRD. This command returns a string of 16 values. These values of the 4-by-4 matrix begin at the upper-left corner and read across the rows.

System Reset

The command RESETF resets all flags. The following table lists the default values of all flags that can be reset.

Flag	Name	Default Value	
1	AREAPT	65535 16 times	Solid area
2	CLIPH	Flag = 0	Disabled
3	CLIPY	Flag = 0	Disabled
4	COLOR	Value = 255	
5	DISPLA	No change after a RESETF	
6	DISTAN	Distance = 500	
7	DISTH	Distance = -30000	
8	DISTY	Distance = 30000	
9	FILMSK	Mask = 255	No PEL draw effect
10	LINFUN	Function = 0	Replacement mode
11	LINPAT	Pattern = 65535	Solid line
12	MASK	Mask = 255	All planes enabled
13	MDORG	OX = OY = OZ = 0	
14	2D current point	X = Y = 0	
15	3D current point	X = Y = Z = 0	
16	PRMFIL	Flag = 0	Primitive fill off
17	PROJCT	Angle = 60	
18	TANGLE	Angle = 0	Horizontal, left-right text
19	TJUST	H = V = 1	Left, bottom justification
20	TSIZE	Size = 8	12 by 8 cell characters
21	VWPORT	0, 639, 0, 479	Entire screen
22	VWRPT	X = Y = Z = 0	
23	WINDOW	-320, 319, -240, 239	
24	Transformed 3D current point	X = Y = Z = 0	

Communications

The Professional Graphics Controller accepts high-function graphics commands in either ASCII or hexadecimal format. In ASCII mode, English-like commands and their parameters are sent to the board as ASCII character strings. This allows easy transmission of instructions from such high-level languages as BASIC. For example, to draw a circle of radius 55.05 centered at the screen center, execute a BASIC statement to transmit the following character string:

```
MOVE 0,0 CIRCLE 55.05
```

In hexadecimal communication mode, the commands are sent as a stream of bytes for greatest throughput. The statement above could be sent in hexadecimal mode as

```
10 00 00 00 00 00 00 00 00 38 37 00 CD 0C
```

to realize substantial time savings.

ASCII Communications

ASCII mode commands are sent in a format designed to accommodate the restriction of a high-level language. The ASCII command consists of a command word (no more than six letters in length) and parameters, if applicable. Every command word has a short form, which is always three characters or less in length. Parameters may be either decimal numbers or text strings enclosed in quotes.

Commands and parameters in a command line are separated by delimiters. A delimiter is one or more of the following, except when enclosed by quotation marks:

- Space
- Tab
- Comma
- Semicolon
- Hyphen
- Plus sign

Commands and parameters consist of letters, numbers, and decimal points. Any other character, except when enclosed in quotes, is illegal and will be ignored.

When a hyphen immediately precedes a numeric parameter, that number is interpreted as negative.

Examples of Legal Commands:

```
"CI 5"                Draw a circle of radius 5.
"RECT 67-88"          Draw a rectangle.
"COLOR 2 FLOOD 3"     Change the current color to 2,
                      and flood the screen to the color 3.
"LUTRD 3"             Read LUT entry 3.
```

Examples of Illegal Commands:

```
"CIR 5"               CIR is not a valid abbreviation.
"RECT%67,-68"         "%" is not a legal character.
"COLOR 2 4 FLOOD 3"   COLOR takes only one parameter.
"LUTRD 3.4"           The parameter to the LUTRD command
                      is an integer.
```

Communication Protocol

The high-function graphics data is sent and received as a sequential stream of bytes. To realize maximum throughput between the system and the Professional Graphics Controller, a first-in-first-out (FIFO) buffer protocol has been set up. This protocol must be adhered to for proper transmission and reception. These buffers, and their associated pointers and flags, are directly addressable when the system uses addresses in the hexadecimal range C6000 to C63FF.

There are three channels through which data may pass to and from the controller. From the system's point of view, these channels are 'output' (for sending commands and parameters), 'input' (for receiving data read-back commands), and 'error' (for receiving high-function graphics-generated error and warning codes). Each channel has a FIFO buffer associated with it and each buffer has 256 bytes reserved in the 1K-byte communication area. A portion of the remaining 256 bytes is reserved for three sets of buffer pointers—one pair for each channel—as well as the warm and cold restart and diagnostic flags. The following memory map shows the addresses as seen by the system.

Memory Address (in hex)	Function
C6000	Output FIFO (256 bytes)
C6100	Input FIFO (256 bytes)
C6200	Error FIFO (256 bytes)
C6300	Output FIFO Write Pointer
C6301	Output FIFO Read Pointer
C6302	Input FIFO Write Pointer
C6303	Input FIFO Read Pointer
C6304	Error FIFO Write Pointer
C6305	Error FIFO Read Pointer
C6306	Cold Restart Flag
C6307	Warm Restart Flag
C6308	Error Enable Flag

Each buffer has a one-byte read pointer and a one-byte write pointer, which refer to buffer locations relative to the base of the buffer in question. The read pointer always points to the next byte to be read; the write pointer always points to the next byte to be written. The buffer is empty when the read pointer is equal to the write pointer, because the byte that would be read has not yet been written. Alternately, the buffer is full when the write pointer is one less than the read pointer.

A FIFO write must be done as follows:

1. Ensure the buffer has room by comparing the write pointer to the read pointer. If the read pointer is only one greater than the write pointer, there is no room, and no writing may take place until there is room.

2. Write one byte to the address specified by that buffer's base address plus the value in its write pointer.

3. Increment the write pointer, modulo-255.

More than one byte may be written if the buffer's write pointer is increased by the same number as the number of bytes written.

A FIFO read must be done as follows:

1. Ensure the buffer has data by comparing the write pointer to the read pointer. If the read pointer is equal to the write pointer, the buffer is empty, and no reading may take place until there is data to be read.

2. Read one byte from the address specified by that buffer's base address plus the value in its read pointer.

3. Increment the read pointer, modulo-255.

More than one byte may be read if the buffer's read pointer is increased by the same number as the number of bytes read.

Error Handling

The high-function graphics mode provides an error-reporting capability. If the host sets the error-enable flag in the communication area, the high-function graphics mode returns errors in the error buffer. In ASCII mode, the error is returned as a message, such as "Arithmetic Overflow." In hexadecimal mode, the error is returned as a single byte code.

High-Function Graphics Commands

The high-function graphics commands can be logically grouped into the following categories:

- Two-Dimensional Drawing
  - ARC (AR) Arc
  - CIRCLE (CI) Circle
  - DRAW (D) Draw
  - DRAWR (DR) Draw Relative
  - ELIPSE (EL) Ellipse
  - MOVE (M) Move
  - MOVER (MR) Move Relative
  - POINT (PT) Point
  - POLY (P) Polygon
  - POLYR (PR) Polygon Relative
  - RECT (R) Rectangle
  - RECTR (RR) Rectangle Relative
  - SECTOR (S) Sector
- Three-Dimensional Drawing
  - DRAW3 (D3) Draw in 3D
  - DRAWR3 (DR3) Draw Relative in 3D
  - MOVE3 (M3) Move in 3D
  - MOVER3 (MR3) Move Relative in 3D
  - POINT3 (PT3) Point in 3D
  - POLY3 (P3) Polygon in 3D
  - POLYR3 (PR3) Polygon Relative in 3D
- Modeling Transformations
  - MATXRD (MRD) Matrix Read
  - MDIDEN (MDI) Modeling Identity
  - MDMATX (MDM) Modeling Matrix
  - MDORG (MDO) Modeling Origin
  - MDROTX (MDX) Modeling Rotate X Axis
  - MDROTY (MDY) Modeling Rotate Y Axis
  - MDROTZ (MDZ) Modeling Rotate Z Axis
  - MDSCAL (MDS) Modeling Scale
  - MDTRAN (MDT) Modeling Translation
- Viewport/Window/Projection
  - CLIPH (CH) Clip Hither
  - CLIPY (CY) Clip Yon
  - CONVRT (CV) Convert
  - DISTAN (DS) Distance
  - DISTH (DH) Distance Hither

  - DISTY (DY) Distance Yon
  - PROJCT (PRO) Projection
  - VWIDEN (VWI) Viewing Identity
  - VWMATX (VWM) Viewing Matrix
  - VWPORT (VWP) Viewport
  - VWROTX (VWX) Viewing Rotate X Axis
  - VWROTY (VWY) Viewing Rotate Y Axis
  - VWROTZ (VWZ) Viewing Rotate Z Axis
  - VWRPT (VWR) Viewing Reference Point
  - WINDOW (WI) Window
- Command List
  - CLBEG (CB) Command List Begin
  - CLDEL (CD) Command List Delete
  - CLEND (CE) Command List End
  - CLOOP (CL) Command List Loop
  - CLRD (CRD) Command List Read
  - CLRUN (CR) Command List Run
- Mode Set/Read
  - CA (CA) Communications ASCII
  - CX (CX) Communications Hexadecimal
  - DISPLA (DI) Display
  - FLAGRD (FRD) Flag Read
  - RESETF (RF) Reset Flags
  - WAIT (W) Wait
- Color/Fills/Patterns
  - AREA (A) Area Fill
  - AREABC (AB) Area Fill to Boundary Color
  - AREAPT (AP) Area Pattern
  - CLEARS (CLS) Clear Screen
  - COLOR (C) Color
  - FLOOD (F) Flood
  - FILMSK (FM) Fill Mask
  - LINFUN (LF) Line Function
  - LINPAT (LP) Line Pattern
  - MASK (MK) Mask
  - PRMFIL (PF) Primitive Fill
- Image Transmission
  - IMAGER (IR) Image Read
  - IMAGEW (IW) Image Write

- Look-Up Table Operations
  - LUT (L) Look-Up Table
  - LUTINT (LI) Look-Up Table Initialize
  - LUTRD (LRD) Look-Up Table Read
  - LUTSAV (LS) Look-Up Table Save
- Text
  - TANGLE (TA) Text Angle
  - TDEFIN (TD) Text Define
  - TEXT (T) Text
  - TEXTP (TP) Text Programmed
  - TJUST (TJ) Text Justify
  - TSIZE (TS) Text Size

The high-function graphics commands appear on the following pages in alphabetic order.

ARC (Arc)

Purpose: Draw an arc in two dimensions.

Command: ARC radius deg0 deg1

Description: ARC draws the arc of a circle in the current color. The center is at the current point. The radius is specified in the attribute radius, starting at the angle given in *deg0* and ending at the angle given in *deg1*. The angles are expressed in degrees and are measured counterclockwise from a ray that is parallel to the X axis, starting at the origin and going toward increasing X values. Radius values are real numbers and may range from -8191 to 8191. Start and end angles are treated as modulo-360. If *radius* is negative, 180 degrees are added to both angles.

Short Form: AR radius deg0 deg1

Hex Format:

3C	lowradius	highradius
	lowfracradius	highfracradius
	lowdeg0	highdeg0
	lowdeg1	highdeg1

Example:

```
ASCII: AR 50.25 45 135

HEX:   3C 32 00 00 40 2D 00 87 00
```

Errors: Radius too large

AREA (Area Fill)

Purpose: Random area fill.

Command: AREA

Description: AREA sets all PELs in a given closed region to the current color. The region extends from the two-dimensional current point outward in all directions until reaching a boundary of PELs whose colors differ from the original color of the PEL at the current point and the current color. The region to be filled must be continuous. All data read is ANDed against the fill mask and the mask to compare colors. The original color should not be equal to the current color.

Short Form: A

Hex Format: C0

Example:

```
ASCII: A

HEX:   C0
```

Errors: None

AREABC (Area Fill to Boundary Color)

Purpose: Random area fill to the boundary color.

Command: AREABC bcolor

Description: AREABC sets all PELs in a given closed region to the current color under mask. The region extends from the two-dimensional current point outward until reaching a boundary of PELs with the color specified by *bcolor. Bcolor* must be different from the current color. All data read is ANDed against the fill mask and the mask for boundary comparison.

Short form: AB bcolor

Hex Format: C1 bcolor

Example:

```
ASCII: AB 4

HEX:   C1 04
```

Errors: Boundary = current color

AREAPT (Area Pattern)

Purpose: Define an area pattern mask.

Command: AREAPT pattern

Description: AREAPT defines the area pattern mask. The 16 pattern mask words define a 16-by-16 PEL array to be repeated horizontally and vertically when drawing filled figures. Setting all bits in the mask (sending 16 words of 65535) causes areas to be filled solidly; this is the default after a reset.

Short Form: AP pattern

Hex Format:

```
E7  lowp0   highp0   lowp1   highp1
    lowp2   highp2   lowp3   highp3
    lowp4   highp4   lowp5   highp5
    lowp6   highp6   lowp7   highp7
    lowp8   highp8   lowp9   highp9
    lowp10  highp10  lowp11  highp11

    lowp12  highp12  lowp13  highp13
    lowp14  highp14  lowp15  highp15
```

Example:

```
ASCII: AP 52428 52428 13107 13107
          52428 52428 13107 13107
          52428 52428 13107 13107
          52428 52428 13107 13107

HEX:   E7 CC CC CC CC 33 33 33 33
          CC CC CC CC 33 33 33 33
          CC CC CC CC 33 33 33 33
          CC CC CC CC 33 33 33 33
```

Errors: None

CA (Communications ASCII)

Purpose: Set the communication mode to ASCII.

Command: CA

Description: This command may be given in either ASCII or hexadecimal mode.

Short Form: CA

Hex Format: 43 41 20

Note: This is the hexadecimal equivalent of the three ASCII characters "CA ".

Example:

```
ASCII: CA

HEX:   43 41 20
```

Errors: None

CIRCLE (Circle)

Purpose: Draw a circle in two dimensions.

Command: CIRCLE radius

Description: CIRCLE draws a circle of a given radius, with its center at the current point. The circle is drawn in the current color and is filled if the PRMFIL flag is set (see "PRMFIL"). Nothing is drawn if the radius value is outside the range of -8191 to 8191.

Short Form: CI radius

Hex Format:

```
38   lowradius       highradius
     lowfracradius   highfracradius
```

Example:

```
ASCII: CI 25.5 5 135

HEX:   38 19 00 00 80
```

Errors: Radius too large

CLBEG (Command List Begin)

Purpose: Begin command-list definition.

Command: CLBEG clist

Description: CLBEG begins the definition of the command list specified by *clist*. Commands sent later to the controller are saved in the command-list definition area for execution (see "CLRUN" and "CLOOP"). CLEND ends the command-list definition. *clist* may be from 0 to 255. Any previous definition of the command-list is erased.

Short Form: CB clist

Hex Format: 70 clist

Example:

```
ASCII: CLBEG 1

HEX:   70 01 07 02 06 01 30 00 C8 00 00 71
```

Errors: Not enough memory; command list running

CLDEL (Command List Delete)

Purpose: Delete the definition of a command list.

Command: CLDEL clist

Description: CLDEL deletes the definition of the command list specified by *clist.* It also reclaims command-list memory for other definitions. *clist* may be from 0 to 255.

Short Form: CD clist

Hex Format: 74 clist

Example:

```
ASCII: CD 3

HEX:   74 03
```

Error: Command list running

CLEARS (Clear Screen)

Purpose: Clear the screen to a given color.

Command: CLEARS color

Description: Sets every PEL in the high-function graphics display buffer to the color specified by *color* regardless of the mask. This command does not change the current color. It is similar, but not identical, to the command FLOOD.

Short Form: CLS color

Hex Format: 0F color

Example:

```
ASCII: CLS 23

HEX:   0F 17
```

Errors: None

CLEND (Command List End)

Purpose: End the definition of a command-list.

Command: CLEND

Description: CLEND ends the definition of a command-list. When the controller receives a CLEND, it resumes executing commands as they are received.

Short Form: CE

Hex Format: 71

Example:

```
ASCII: CE

HEX:   71
```

Errors: None

CLIPH (Clip Hither)

Purpose: Set the hither clip flag.

Command: CLIPH flag

Description: CLIPH enables or disables hither clipping. Hither clipping is enabled when *flag* is 1 or any odd number, and disabled when *flag* is 0 or any even number (default). Three-dimensional drawing commands draw faster when hither clipping is disabled.

Short Form: CH flag

Hex Format: AA flag

Example:

```
ASCII: CH 0

HEX:   AA 01
```

Errors: None

CLIPY (Clip Yon)

Purpose: Set the yon clip flag.

Command: CLIPY flag

Description: CLIPY enables or disables yon clipping. Yon clipping is enabled when *flag* is 1 or any odd number, and disabled when *flag* is 0 or any even number (default). Three-dimensional drawing commands draw faster when yon clipping is disabled.

Short Form: CY flag

Hex Format: AB flag

Example:

```
ASCII: CY 0

HEX:   AB 01
```

Errors: None

CLOOP (Command List Loop)

Purpose: Repeat execution of a command list.

Command: CLOOP clist count

Description: CLOOP executes the command list specified by *clist*, for the number of times specified by *count*. *clist* may be between 0 and 255; *count* can be from 0 to 65535.

Short Form: CL clist count

Hex Format: 73 clist lowcount highcount

Example:

```
ASCII: CL 1 1000

HEX:   73 01 E8 03
```

Errors: Command list running; stack full.

CLRD (Command List Read)

Purpose: Read back command list.

Command: CLRD clist

Description: In hexadecimal mode, a word representing the number of bytes in the command list is read back (zero if the list is undefined), followed by the bytes as they are stored.

Short Form: CRD clist

Hex Format: 75 clist

Example:

```
ASCII: CRD 1

HEX:   75 01
```

Errors: None

CLRUN (Command List Run)

Purpose: Execute command list.

Command: CLRUN clist

Description: CLRUN executes commands in the command list specified by *clist*. *clist* must be from 0 to 15.

Short Form: CR clist

Hex Format: 72 clist

Example:

```
ASCII: CR 14

HEX:   72 01
```

Errors: Command list running; stack full; nested command list

COLOR (Color)

Purpose: Set the current color.

Command: COLOR value

Description: COLOR sets the current color to that specified by *value*. All noncomplement mode drawing is done in the current color. All drawing, including complement mode, is subject to MASK and FILMSK. *value* is treated as modulo-256.

Short Form: C value

Hex Format: 06 value

Example:

```
ASCII: C 2

HEX:   06 02
```

Errors: None

CONVRT (Convert)

Purpose: Convert three dimension to two dimension.

Command: CONVRT

Description: CONVRT converts the three-dimensional current point to two-dimensional virtual coordinates, using the current transformation matrixes. The result is left in the two-dimensional current point.

Short Form: CV

Hex format: AF

Example:

```
ASCII: CV

HEX:   AF
```

Errors: Arithmetic overflow

CX (Communications Hexadecimal)

Purpose: Set the communication mode to hexadecimal.

Command: CX

Description: This command may be given in either ASCII or hexadecimal mode.

Short Form: CX

Hex Format: 43 58 20

Note: This is the hexadecimal equivalent of the three ASCII characters “CA ”.

Example:

```
ASCII: CX

HEX:   43 58 20
```

Errors: None

DISPLA (Display)

Purpose: Select the display mode.

Command: DISPLA flag

Description: DISPLA selects a screen for display. If *flag* is 0, the color high-function graphics screen is displayed. If *flag* is 1, the emulator screen is shown. Color graphics commands are accepted and executed, no matter which screen is displayed.

Short Form: DI flag

Hex Format: D0 flag

Example:

```
ASCII: DI 0

HEX:   D0 01
```

Errors: None

DISTAN (Distance)

Purpose: Define the distance to the viewing reference point.

Command: DISTAN dist

Description: DISTAN defines the distance (*dist*) from the eye to the viewing reference point.

Short Form: DS dist

Hex Format: B1 lowdist highdist
lowfracdist highfracdist

Example:

```
ASCII: DS 1200

HEX:   B1 B0 04 9A 59
```

Errors: None

DISTH (Distance Hither)

Purpose: Define the hither clip plane.

Command: DISTH dist

Description: DISTH defines the distance to the hither clip plane from the viewing reference point. The hither clip plane is parallel to the view plane, and the distance (*dist*) is relative. When hither clipping is enabled, no points before the hither clip plane are displayed. Hither clipping affects only three-dimensional drawing commands.

Short Form: DH dist

Hex Format:

A8	lowdist	highdist
	lowfracdist	highfracdist

Examples:

```
ASCII: DH 15.01

HEX:   A8 OF 00 8F 02
```

Errors: None

DISTY (Distance Yon)

Purpose: Define the yon clip plane.

Command: DISTY dist

Description: DISTY defines the distance to the yon clip plane from the viewing reference point. The yon clip plane is parallel to the view plane, and the distance (*dist*) is relative. When yon clipping is enabled, no points beyond the yon clip plane are displayed. Yon clipping affects only three-dimensional drawing commands.

Short Form: DY dist

Hex Format:

A9	lowdist	highdist
	lowfracdist	highfracdist

Example:

```
ASCII: DY 15.999

HEX:   A9 0F 00 BE FF
```

Errors: None

DRAW (Draw)

Purpose: Absolute draw in two dimensions.

Command: DRAW x y

Description: DRAW draws a line from the current point to the point specified by *x,y*. The current point moves to the *x* and *y* value.

Short Form: D x y

Hex Format:

28	lowx	highx
	lowfracx	highfracx
	lowy	highy
	lowfracy	highfracy

Example:

```
ASCII: D 23.5 -90.71

HEX:   20 17 00 00 80 A5 FF C3 B5
```

Errors: Arithmetic overflow

DRAWR (Draw Relative)

Purpose: Relative draw in two dimensions.

Command: DRAWR dx dy

Description: DRAWR draws a line from the current point to a point *dx,dy* from the current point. The current point moves to the end point of the line.

Short Form: DR dx dy

Hex Format:

	29	lowdx	highdx
		lowfracdx	highfracdx
		lowdy	highdy
		lowfracdy	highfracdy

Example:

```
ASCII: DR 65.8 12.2

HEX:   21 41 00 CD CC 0C 00 34 33
```

Errors: Arithmetic overflow

DRAW3 (Draw in 3D)

Purpose: Draw absolute in three dimensions.

Command: DRAW3 x y z

Description: DRAW3 draws a line from the current point to the point in the three-dimensional space given. After the draw, the current point moves to *x,y,z*.

Short Form: D3 x y z

Hex Format:

2A	lowx	highx
	lowfracx	highfracx
	lowy	highy
	lowfracy	highfracy
	lowz	highz
	lowfracz	highfracz

Example:

```
ASCII: D3 943, -266, 100

HEX:   22 AF 03 00 00 F6 FE 00 00 64 00 00 00
```

Errors: Arithmetic overflow

DRAWR3 (Draw Relative in 3D)

Purpose: Draw relative in three dimensions.

Command: DRAWR3 dx dy dz

Description: DRAWR3 draws a line to the point offset from the current point by *dx,dy,dz* and moves the current point to this new point.

Short Form: DR3 dx dy dz

Hex Format:

2B	lowdx	highdx
	lowfracdx	highfracdx
	lowdy	highdy
	lowfracdy	highfracdy
	lowdz	highdz
	lowfracdz	highfracdz

Example:

```
ASCII: DR3 835.02 44.62 98

HEX:   23 43 03 1F 05 2C 00 B8 9E 62 00 00 00
```

Errors: Arithmetic overflow

ELIPSE (Ellipse)

Purpose: Draw an ellipse in two dimensions.

Command: ELIPSE xradius yradius

Description: ELIPSE draws an ellipse centered on the two-dimensional current point whose x and y axis lengths are given in *xradius* and *yradius*. The ellipse is filled if the PRMFIL flag is set.

Short Form: EL xradius yradius

Hex Format:

39	lowxradius	highxradius
	lowfracxradius	highfracxradius
	lowyradius	highyradius
	lowfracyradius	highfracyradius

Example:

```
ASCII: EL 50 100

HEX:   39 25 00 00 80 19 00 00 00
```

Errors: Radius too large

FILMSK (Fill Mask)

Purpose: Set area fill mask.

Command: FILMSK mask

Description: FILMSK sets the 8-bit area fill mask to *mask*. All PELs read by the Area Fill commands are ANDed against this mask, and also MASK, before comparison with the boundary color.

Short Form: FM mask

Hex Format: EF mask

Example:

```
ASCII: FM 254

HEX:   EF FE
```

Errors: None

FLAGRD (Flag Read)

Purpose: Read flag value.

Command: FLAGRD flag

Description: FLAGRD loads the current value of the flag specified by *flag* into the output buffer for later reading by the host. The flag numbers assigned are as follows.

Flag	Name	Type of Value Returned
1	AREAPT	16 integers
2	CLIPH	1 integer (byte)
3	CLIPY	1 integer (byte)
4	COLOR	1 integer (byte)
5	DISPLA	1 integer (byte)
6	DISTAN	1 real number
7	DISTH	1 real number
8	DISTY	1 real number
9	FILMSK	1 integer (byte)
10	LINFUN	1 integer (byte)
11	LINPAT	1 integer
12	MASK	1 integer (byte)
13	MDORG	3 real numbers
14	2D current point	2 real numbers
15	3D current point	3 real numbers
16	PRMFIL	1 integer (byte)
17	PROJCT	1 integer (byte)
18	TANGLE	1 word
19	TJUST	2 integers (bytes)
20	TSIZE	1 real number
21	VWPORT	4 integers
22	VWRPT	3 real numbers
23	WINDOW	4 real numbers
24	Transformed 3D current point	3 real numbers
25	Free memory available	1 integer

Each value is read in the same order as provided to the command that sets it. For example, the three-dimensional current point is read as one real number each for x, y, and z. In ASCII mode, commas separate multiple return values, with a carriage return at the end.

Short Form: FRD flag

Hex Format: 51 flag

Example:

```
ASCII: FRD 3

HEX:   51 03
```

Error: None

FLOOD (Flood)

Purpose: Flood the screen to the color given.

Command: FLOOD color

Description: FLOOD sets every PEL in the defined viewport, to the color specified by *color* subject to MASK. This command does not change the current color.

Short Form: F color

Hex Format: 07 color

Example:

```
ASCII: F 4

HEX:   07 04
```

Errors: None

IMAGER (Image Read)

Purpose: Read image from the display.

Command: IMAGER line x1 x2

Description: IMAGER reads a line from the image being displayed. If the communication mode is ASCII (CA) the image is placed in the output buffer as one ASCII number for each PEL, separated by carriage returns. If communication is in hexadecimal mode (CX) the image output is in a run-length encoded format. *line, x1,* and *x2* are expressed in PELs measured from the lower-left corner of the screen.

Short Form: IR line x1 x2

Hex Format:

D8	lowline	highline
	lowx1	highx1
	lowx2	highx2

Example:

```
ASCII: IR 100 0 127

HEX: D8 64 00 00 00 7F 00
```

Errors: Value out of range

IMAGEW (Image Write)

Purpose: Write image to the display.

Command: IMAGEW line x1 x2

Description: IMAGEW writes a line of PELs to the display. If communication is in ASCII (CA) each parameter represents one PEL. If communication is in hexadecimal (CX) the image is sent in run-length encoded format. *line, x1,* and *x2* are expressed in PELs measured from the lower-left corner of the screen.

Short Form: IW line x1 x2

Hex Format:

```
D9   lowline   highline
     lowx1     highx1
     lowx2     highx2
     . . . .   data
```

Example:

```
ASCII: IW 100 50 60

HEX:   D9 64 00 32 00 3C 00 82 2C
          18 42 03 0C 01 0E 81 18 2C
```

Errors: Value out of range

LINFUN (Line Function)

Purpose: Select drawing function.

Command: LINFUN function

Description: LINFUN sets the drawing function to that specified by *function*. Available functions are:

0 Draw by writing PELs of the current color (default).

1 Draw by complementing PEL. The current color will be ignored.

Note: With both functions, drawing is subject to MASK and FILMSK where appropriate.

Short Form: LF function

Hex Format: EB function

Example:

```
ASCII: LF 0

HEX:   EB 00
```

Errors: None

LINPAT (Line Pattern)

Purpose: Set line pattern.

Command: LINPAT pattern

Description: LINPAT sets the line-drawing pattern from a 16-bit number. The line pattern is used to implement dotted or dashed lines. As each PEL is generated, the line-pattern mask is rotated right. If there is a 1 in the least-significant bit (LSB), a PEL is drawn. If that bit is a 0 then no PEL is drawn and the background remains visible. A line-pattern mask of all 1's (65535) produces solid lines, and is the default following a RESETF. The line pattern affects the following commands except when drawing a filled primitive:

ARC, CIRCLE, DRAW, DRAW3, DRAWR, DRAWR3, ELIPSE, POLY, POLY3, POLYR, POLYR3, RECT, RECTR, SECTOR

Short Form: LP pattern

Hex Format: EA lowpattern highpattern

Example:

```
ASCII: LP 65280

HEX:   EA 00 FF
```

Errors: None

LUT (Look-Up Table)

Purpose: Set an entry in the look-up table.

Command: LUT index r g b

Description: LUT loads red, green, and blue intensity levels into the LUT entry specified by *index*. Intensity values are treated as modulo-16 numbers.

Short Form: L index r g b

Hex Format: EE index r g b

Example:

```
ASCII: L 3 0 15 0

HEX:   EE 04 00 00 0F
```

Errors: None

LUTINT (Look-Up Table Initialize)

Purpose: Initialize the look-up table.

Command: LUTINT state

Description: LUTINT sets the LUT to one of the following states specified by *state*:

State	
0	Color-cone distribution
1	Foreground/background colors in the low 4-bits of a value code will be visible only if the high 4-bits is 0 (or "invisible")
2	Value codes interpreted as: R R G G G B B B
3	Value codes interpreted as: R R R G G B B B
4	Value codes interpreted as: R R R G G G B B
5	6-level RGB
255	Load LUT from LUT storage area (opposite of LUTSAV)

Short Form: LI state

Hex Format: EC state

Example:

```
ASCII: LI 4

HEX:   EC 04
```

Errors: Value out of range

LUTRD (Look-Up Table Read)

Purpose: Read the look-up table entry.

Command: LUTRD index

Description: LUTRD loads the red, green, and blue entries at the LUT entry specified by *index* into the output buffer for reading by the host.

In ASCII mode, the LUT entries are read as red, green, and blue intensities, separated by commas, and ended by a carriage return.

In hexadecimal mode, the LUT entries are read one byte for each entry for a total of three bytes.

Short Form: LRD index

Hex Format: 50 index

Example:

```
ASCII: LRD 2

HEX:   50 02
```

Errors: None

LUTSAV (Look-Up Table Save)

Purpose: Save the look-up table in the look-up table storage area.

Command: LUTSAV

Description: LUTSAV saves all 256 LUT entries in the LUT storage area. These values may be reloaded with a "LUTINT 255" command. Each LUTSAV overwrites any previous LUTSAV.

Short Form: LS

Hex Format: ED

Example:

```
ASCII: LS

HEX:   ED
```

Errors: None

MASK (Mask)

Purpose: Set bit-plane mask.

Command: MASK planemask

Description: MASK sets the 8-bit, read/write, bit-plane mask to the value specified by *planemask*. A zero in any position in the mask means that no bits in that plane are written to; when read, bits in that plane return zero. Because of the organization of display memory, the fastest drawing speed occurs when *planemask* is FF, 0F, or F0.

Short Form: MK planemask

Hex Format: E8 planemask

Example:

```
ASCII: MK 15

HEX:   E8 0F
```

Errors: None

MATXRD (Matrix Read)

Purpose: Read the matrix contents.

Command: MATXRD matrix

Description: MATXRD reads the contents of the 4-by-4 matrix specified by *matrix* into the output buffer for later reading by the host. The matrix number assignments are:

1 Three-dimensional modeling transformation matrix

2 Three-dimensional viewing transformation matrix

In ASCII mode, the matrix entries are read in four lines. Each line has four entries separated by commas.

In hexadecimal mode, four bytes for each matrix entry are read, for a total of 64 bytes. The reading order is:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Short Form: MRD matrix

Hex Format: 52 matrix

Example:

```
ASCII: MRD 1

HEX:   52 01
```

Errors: Value out of range

MDIDEN (Modeling Identity)

Purpose: Reset the modeling transformation matrix.

Command: MDIDEN

Description: MDIDEN sets the modeling transformation matrix to the identity matrix.

Short Form: MDI

Hex Format: 90

Example:

```
ASCII: MDI

HEX:  90
```

Errors: None

MDMATX (Modeling Matrix)

Purpose: Define the modeling matrix.

Command: MDMATX array

Description: MDMATX loads the modeling matrix directly from the 4-by-4 real-number array.

Short Form: MDM array

Hex Format:

```
97 lowm11  highm11  lowfracm11  highfracm11
   lowm12  highm12  lowfracm12  highfracm12
   lowm13  highm13  lowfracm13  highfracm13
   lowm14  highm14  lowfracm14  highfracm14
   lowm21  highm21  lowfracm21  highfracm21
   lowm22  highm22  lowfracm22  highfracm22
   lowm23  highm23  lowfracm23  highfracm23
   lowm24  highm24  lowfracm24  highfracm24
   lowm31  highm31  lowfracm31  highfracm31
   lowm32  highm32  lowfracm32  highfracm32
   lowm33  highm33  lowfracm33  highfracm33
   lowm34  highm34  lowfracm34  highfracm34
   lowm41  highm41  lowfracm41  highfracm41
   lowm42  highm42  lowfracm42  highfracm42
   lowm43  highm43  lowfracm43  highfracm43
   lowm44  highm44  lowfracm44  highfracm44
```

Example:

```
ASCII: MDM 68.25 12.5   253    17
           65503 0.25   306.75 34.5
           8418  324.75 1.25   0
           313.5 50     1.25   1

HEX:     97 44 00 00 40 0C 00 00 80 FD 00 00 00
            11 00 00 00 DF FF 00 00 00 00 00 40
            32 01 00 C0 22 00 00 80 E2 20 00 00
            44 01 00 C0 01 00 00 40 00 00 00 00
            39 01 00 80 32 00 00 00 01 00 00 40
            01 00 00 00
```

Errors: Arithmetic overflow

MDORG (Modeling Origin)

Purpose: Define the modeling origin.

Command: MDORG ox oy oz

Description: MDORG defines the origin for modeling-transformation scaling and rotating specified by *ox,oy,oz.*

Short Form: MDO ox oy oz

Hex Format:

```
91 lowox highox lowfracox  highfracox
   lowoy highoy lowfracoy  highfracoy
   lowoz highoz lowfracoz  highfracoz
```

Example:

```
ASCII: MDO 1.7 0.2 1.5

HEX:   91 01 00 33 B3 00 00 33 33 01 00 00 80
```

Errors: None

MDROTX (Modeling Rotate X Axis)

Purpose: Rotate about the X axis.

Command: MDROTX deg

Description: MDROTX defines the rotation about the x axis component of the modeling matrix.

Short Form: MDX deg

Hex Format: 93 lowdeg highdeg

Examples:

```
ASCII: MDX 30

HEX:   93 2D 00
```

Errors: Arithmetic overflow

MDROTY (Modeling Rotate Y Axis)

Purpose: Rotate about the Y axis.

Command: MDROTY deg

Description: MDROTY defines the rotation about the y axis component of the modeling matrix.

Short Form: MDY deg

Hex Format: 94 lowdeg highdeg

Example:

```
ASCII: MDY 15

HEX:   94 OF 00
```

Errors: Arithmetic overflow

MDROTZ (Modeling Rotate Z Axis)

Purpose: Rotate about the Z axis.

Command: MDROTZ deg

Description: MDROTZ defines the rotation about the z axis component of the modeling matrix.

Short Form: MDZ deg

Hex Format: 95 lowdeg highdeg

Example:

```
ASCII: MDZ 33

HEX:   95 21 00
```

Errors: Arithmetic overflow

MDSCAL (Modeling Scale)

Purpose: Set modeling scaling.

Command: MDSCAL sx sy sź

Description: MDSCAL defines the scaling components for the image transformation.

Short Form: MDS sx sy sz

Hex Format

```
92 lowsx  highsx  lowfracsx  highfracsx
   lowsy  highsy  lowfracsy  highfracsy
   lowsz  highsz  lowfracsz  highfracsz
```

Example:

```
ASCII: MDS 2 2 2

HEX:   92 02 00 00 80 01 00 00 00 01 00 00 80
```

Errors: Arithmetic overflow

MDTRAN (Modeling Translation)

Purpose: Define the modeling translation.

Command: MDTRAN tx ty tz

Description: MDTRAN defines the translation components for the image transformation specified by *tx,ty,tz*.

Short Form: MDT tx ty tz

Hex Format:

```
96 lowtx  hightx  lowfractx  highfractx
   lowty  highty  lowfracty  highfracty
   lowtz  hightz  lowfractz  highfractz
```

Example:

```
ASCII: MDT 50 0 0

HEX:   96 32 00 00 00 00 00 00 00 00 00 00 00
```

Errors: Arithmetic overflow

MOVE (Move)

Purpose: Absolute move in two dimensions.

Command: MOVE x y

Description: MOVE moves the two-dimensional current point to the x and y coordinates given.

Short Form: M x y

Hex Format:

```
10   lowx highx  lowfracx  highfracx
     lowy highy  lowfracy  highfracy
```

Example:

```
ASCII: M 300 -400

HEX:   10 2C 01 00 00 70 FE 00 00
```

Errors: None

MOVER (Move Relative)

Purpose: Relative move in two dimensions.

Command: MOVER dx dy

Description: MOVER moves the two-dimensional current point a relative amount specified by *dx,dy.*

Short Form: MR dx dy

Hex Format:

```
11   lowdx  highdx  lowfracdx  highfracdx
     lowdy  highdy  lowfracdy  highfracdy
```

Example:

```
ASCII: MR 20.44 59

HEX:   11 14 00 A2 71 3B 00 00 00
```

Errors: Arithmetic overflow

MOVE3 (Move in 3D)

Purpose: Absolute move in three dimensions.

Command: MOVE3 x y z

Description: MOVE3 moves the three-dimensional current point to the coordinates specified by *x,y,z*.

Short Form: M3 x y z

Hex Format:

```
12   lowx highx  lowfracx  highfracx
     lowy highy  lowfracy  highfracy
     lowz highz  lowfracz  highfracz
```

Example:

```
ASCII: M3 -1300 -233 519

HEX: 12 EC FA 00 00 17 FF 00 00 07 02 00 00
```

Errors: None

MOVER3 (Move Relative in 3D)

Purpose: Relative move in three dimensions.

Command: MOVER3 dx dy dz

Description: MOVER3 moves the three-dimensional current point a relative amount specified by *dx,dy,dz*.

Short Form: MR3 dx dy dz

Hex Format:

```
13   lowdx highdx lowfracdx highfracdx
     lowdy highdy lowfracdy highfracdy
     lowdz highdz lowfracdz highfracdz
```

Example:

```
ASCII: MR3 722 0 0

HEX: 13 D2 02 00 00 00 00 00 00 00 00 00 00
```

Errors: Arithmetic overflow

POINT (Point)

Purpose: Set the PEL to the current color in two dimensions.

Command: POINT

Description: POINT writes the current color to the PEL at the two-dimensional current point.

Short Form: PT

Hex Format: 08

Example:

```
ASCII: PT

HEX:   08
```

Errors: None

POINT3 (Point in 3D)

Purpose: Set the PEL to the current color in three dimensions.

Command: POINT3

Description: POINT3 writes the current color to the PEL at the current three-dimensional point.

Short Form: PT3

Hex Format: 09

Example:

```
ASCII: PT3

HEX:   09
```

Errors: None

POLY (Polygon)

Purpose: Draw a polygon.

Command: POLY npts x1 y1 x2 y2 . . . . . xn yn

Description: POLY draws an absolute polygon in two dimensions, where *npts* is the number of points, and *x* and *y* are the coordinates of the points. The polygon is filled if the PRMFIL flag is set. The current point is not changed.

Short Form: P npts x1 y1 x2 y2 . . . . . xn yn

Hex Format:

```
30 npts lowx1   highx1   lowfracx1   highfracx1
        lowy1   highy1   lowfracy1   highfracy1
        lowx2   highx2   lowfracx2   highfracx2
        lowy2   highy2   lowfracy2   highfracy2
        . . . . . . . . .
        lowxN   highxN   lowfracxN   highfracxN
        lowyN   highyN   lowfracyN   highfracyN
```

Example:

```
ASCII: P 3 0 0  10 10  -10 30

HEX:   30 03 00 00 00 00 00 00 00 00
             0A 00 00 00 F6 FF 00 00
             F6 FF 00 00 E2 FF 00 00
```

Errors: Not enough memory; arithmetic overflow

POLYR (Polygon Relative)

Purpose: Draw a relative polygon.

Command: POLYR npts dx1 dy1 dx2 dy2 . . . . . dxn dyn

Description: POLYR draws a relative polygon in two dimensions, where *npts* is the number of points, and *dx* and *dy* are the offsets from the current point. The polygon is filled if the PRMFIL flag is set. The current point is not changed.

Short Form: PR npts dx1 dy1 dx2 dy2 . . . . . dxn dyn

Hex Format:

```
31 npts lowdx1   highdx1  lowfracdx1  highfracdx1
        lowdy1   highdy1  lowfracdy1  highfracdy1
        lowdx2   highdx2  lowfracdx2  highfracdx2
        lowdy2   highdy2  lowfracdy2  highfracdy2
        . . . . . . . . .
        lowdxN   highdxN  lowfracdxN  highfracdxN
        lowdyN   highdyN  lowfracdyN  highfracdyN
```

Example:

```
ASCII: PR 3 0 0  20 20 -20 40

HEX:   31 03 00 00 00 00 00 00 00 00
             0A 00 00 00 0A 00 00 00
             F6 FF 00 00 E2 FF 00 00
```

Errors: Not enough memory; arithmetic overflow

POLY3 (Polygon in 3D)

Purpose: Draw a polygon in three dimensions.

Command: POLY3 npts x1 y1 z1 . . . . . xn yn zn

Description: POLY3 draws an absolute polygon in three dimensions, where *npts* is the number of points, and *x, y,* and *z* are the coordinates of the points. The polygon is filled if the PRMFIL flag is set. The current point does not change.

Short Form: P3 npts x1 y1 z1 . . . . . xn yn zn

Hex Format:

```
32 npts lowx1   highx1  lowfracx1   highfracx1
        lowy1   highy1  lowfracy1   highfracy1
        lowz1   highz1  lowfracz1   highfracz1
        lowx2   highx2  lowfracx2   highfracx2
        lowy2   highy2  lowfracy2   highfracy2
        lowz2   highz2  lowfracz2   highfracz2
        . . . . . . . . .
        lowxN   highxN  lowfracxN   highfracxN
        lowyN   highyN  lowfracyN   highfracyN
        lowzN   highzN  lowfraczN   highfraczN
```

Example:

```
ASCII:
P3 3 0 0 0  10 10 10 -10 30 -10

HEX:
32 03 00 00 00 00 00 00 00 00 00 00 00 00
      0A 00 00 00 0A 00 00 00 0A 00 00 00
      F6 FF 00 00 E2 FF 00 00 F6 FF 00 00
```

Errors: Not enough memory; arithmetic overflow

POLYR3 (Polygon Relative in 3D)

Purpose: Draw a relative polygon in three dimensions.

Command: POLYR3 npts dx1 dy1 dz1 . . . . . dxn dyn dzn

Description: POLYR3 draws a relative polygon in three dimensions, where *npts* is the number of points, and *dx, dy,* and *dz* are the offsets from the current point. The polygon is filled if the PRMFIL flag is set. The current point is not affected.

Short Form: PR3 npts dx1 dy1 dz1 . . . . . dxn dyn dzn

Hex Format:

```
33 npts lowdx1  highdx1  lowfracdx1  highfracdx1
        lowdy1  highdy1  lowfracdy1  highfracdy1
        lowdz1  highdz1  lowfracdz1  highfracdz1
        lowdx2  highdx2  lowfracdx2  highfracdx2
        lowdy2  highdy2  lowfracdy2  highfracdy2
        lowdz2  highdz2  lowfracdz2  highfracdz2
        . . . . . . . . .
        lowdxN  highdxN  lowfracdxN  highfracdxN
        lowdyN  highdyN  lowfracdyN  highfracdyN
        lowdzN  highdzN  lowfracdzN  highfracdzN
```

Example:

```
ASCII:
PR3 3 0 0 0  10 10 10 -10 30 -10

HEX:
33 03 00 00 00 00 00 00 00 00 00 00 00 00
      0A 00 00 00 0A 00 00 00 0A 00 00 00
      F6 FF 00 00 E2 FF 00 00 F6 FF 00 00
```

Errors: Not enough memory; arithmetic overflow

PRMFIL (Primitive Fill)

Purpose: Set primitive fill flag.

Command: PRMFIL flag

Description: PRMFIL sets the primitive fill flag to the value specified by *flag*. If *flag* is 0, closed figures are drawn in outline only. If *flag* is 1, closed figures are drawn filled with the current color. If *flag* is 2, there is a performance improvement but degenerate polygons will fill unpredictably. PRMFIL affects the following commands:

CIRCLE, ELIPSE, POLY, POLYR, POLY3, POLYR3, RECT, RECTR, SECTOR

Short Form: PF flag

Hex Format: E9 flag

Example:

```
ASCII: PF 1

HEX:   E9 01
```

Errors: None

PROJCT (Projection)

Purpose: Set the type of projection.

Command: PROJCT angle

Description: PROJCT defines the type of projection used in the three-dimensional to two-dimensional transformation. If *angle* is 0, the projection is orthographic parallel (non-oblique). Otherwise, the projection is perspective, with *angle* being the view angle (default is 60). The range of *angle* is 0 to 179 degrees.

Short Form: PRO angle

Hex Format: B0 angle

Example:

```
ASCII: PR 0

HEX:   B0 3C
```

Errors: Value out of range; arithmetic overflow

RECT (Rectangle)

Purpose: Draw an absolute rectangle in two dimensions.

Command: RECT x y

Description: RECT draws a rectangle with one corner at the current point and its diagonally opposite corner at the point given. The current point does not move. If the PRMFIL flag is set, the rectangle is drawn filled.

Short Form: R x y

Hex Format: 34 lowx highx lowfracx highfracx
lowy highy lowfracy highfracy

Example:

```
ASCII: R 70.50 90.75

HEX:   34 46 00 00 80 5A 00 00 C0
```

Errors: None

RECTR (Rectangle Relative)

Purpose: Draw a relative rectangle in two dimensions.

Command: RECTR dx dy

Description: RECTR draws a rectangle. One corner is at the current point, and its diagonally opposite corner is offset by *dx,dy.* The current point does not move. If the PRMFIL flag is set, the rectangle is drawn filled.

Short Form: RR dx dy

Hex Format:

```
35    lowdx  highdx  lowfracdx   highfracdx
      lowdy  highdy  lowfracdy   highfracdy
```

Example:

```
ASCII: RR -12.5 60

HEX:   35 F3 FF 00 80 3C 00 00 00
```

Errors: Arithmetic overflow

RESETF (Reset Flags)

Purpose: Reset program parameters.

Command: RESETF

Description: Reset all settable flags to their default values.

Flag	Name	Default Value	
1	AREAPT	65535 16 times	Solid area
2	CLIPH	Flag = 0	Disabled
3	CLIPY	Flag = 0	Disabled
4	COLOR	Value = 255	
5	DISPLA	No change after a RESETF	
6	DISTAN	Distance = 500	
7	DISTH	Distance = -30000	
8	DISTY	Distance = 30000	
9	FILMSK	Mask = 255	No PEL draw effect
10	LINFUN	Function = 0	Replacement mode
11	LINPAT	Pattern = 65535	Solid line
12	MASK	Mask = 255	All planes enabled
13	MDORG	OX = OY = OZ = 0	
14	2D current point	X = Y = 0	
15	3D current point	X = Y = Z = 0	
16	PRMFIL	Flag = 0	Primitive fill off
17	PROJCT	Angle = 60	
18	TANGLE	Angle = 0	Horizontal, left-right text
19	TJUST	H = V = 1	Left, bottom justification
20	TSIZE	Size = 8	12 by 8 cell characters
21	VWPORT	0, 639, 0, 479	Entire screen
22	VWRPT	X = Y = Z = 0	
23	WINDOW	-320, 319, -240, 239	
24	Transformed 3D current point	X = Y = Z = 0	

Short Form: RF

Hex Format: 04

Example:

```
ASCII: RF

HEX:   04
```

Errors: None

SECTOR (Sector)

Purpose: Draw a sector in two dimensions.

Command: SECTOR radius deg0 deg1

Description: SECTOR draws a pie-shaped sector that consists of an arc with a given radius, with the arc spanning two given angles, and a vector from the center of the arc to each of the arc's endpoints. If the PRMFIL flag is set, the sector is drawn filled. *radius* is a real number. Angles are integers and treated modulo-360. If *radius* is negative, 180 degrees are added to each angle.

Short Form: S radius deg0 deg1

Hex Format:

3D	lowradius	highradius
	lowfracradius	highfracradius
	lowdeg0	highdeg0
	lowdeg1	highdeg1

Example:

```
ASCII: S 50 -90 30

HEX:   3D 32 00 00 00 A6 FF 1E 00
```

Errors: Arithmetic overflow

TANGLE (Text Angle)

Purpose: Set text angle.

Command: TANGLE deg

Description: TANGLE specifies the angle for drawing text. An angle of 0 (default) causes the text to be drawn normally from left to right.

Short Form: TA deg

Hex Format: 82 lowdeg highdeg

Example:

```
ASCII: TA 90

HEX:   82 5A 00
```

Errors: None

TDEFIN (Text Define)

Purpose: Define programmable text character.

Command: TDEFIN N x y array

Description: TDEFIN stores the character image given by *x, y,* and *array* for a character with the ASCII value of *N.* If communication is in ASCII, the character image is to be sent as a series of 0's and 1's. If communication is in hexadecimal, the character is sent as a series of bytes, as many for each line as required, for as many lines as specified.

Short Form: TD N x y array

Hex Format:

```
84 N x y line1byte1  line1byte2 . . .  line1byteX
         line2byte1  line2byte2 . . .  line2byteX
         . . . . . . . .

         lineYbyte1  lineYbyte2 . . .  lineYbyteX
```

Example:

```
ASCII: T 65 70 12 14

HEX:   84 62 05 07 1E 11 11 1E 10 10 10
```

Errors: Not enough memory

TEXT (Text)

Purpose: Draw hardware font text.

Command:

TEXT ‘string’
TEXT “string”

Description: TEXT writes a text string to the screen, justified about the current point as specified by the last TJUST command. The string may be delimited by either single or double quotes.

Short Form:

T ‘string’
T "string"

Hex Format:

```
80 22 c1 c2 c3
    . . . . . cN 22
    or
80 27 c1 c2 c3
     . . . . . cN 27
```

Example:

```
ASCII: T 'This is a test'

HEX:   80 27 58 20 65 71 75 61
          6C 73 20 31 2E 34 27
```

Errors: Not enough memory

TEXTP (Text Programmed)

Purpose: Draw text using a programmed font.

Command: TEXTP ‘string’
TEXTP “string”

Description: TEXTP draws text with the user-programmed font. The size is that specified by the latest TSIZE command, and the angle is that specified by TANGLE. The text is justified about the current point.

Short Form: TP ‘string’
TP "string"

Hex Format:

```
83 22 c1 c2 c3
    . . . . . cN 22
    or
83 27 c1 c2 c3
    . . . . . cN 27
```

Example:

```
ASCII: TP 'Hello'

HEX:   83 27 48 65 6C 6F 27
```

Errors: Not enough memory

TJUST (Text Justify)

Purpose: Set text justification

Command: TJUST horiz vert

Description: The TJUST command specifies the text justification, where *horiz* is one of the following:

1 Left justify text at current point.
2 Center the text string about the current point.
3 Right justify text at current point.

vert is one of the following:

1 Bottom of text at Y coordinate of current point.
2 Center text string vertically about the current point.
3 Top of text at Y coordinate of current point.

The default is H = 1, V = 1.

Short Form: TJ horiz vert

Hex Format: 85

Example:

```
ASCII: TJ 2 1

HEX:   85 02 01
```

Errors: Value out of range

TSIZE (Text Size)

Purpose: Set the text size.

Command: TSIZE size

Description: TSIZE sets text size by specifying the virtual x distance from one character to the next when displayed.

Short Form: TS size

Hex Format:

```
81 lowsize        highsize
   lowfracsize    highfracsize
```

Example:

```
ASCII: TS 10

HEX:   81 0A 00 00 00
```

Errors: Arithmetic overflow

VWIDEN (Viewing Identity)

Purpose: Reset the viewing matrix.

Command: VWIDEN

Description: VWIDEN sets the viewing transformation matrix to the identity matrix.

Short Form: VWI

Hex Format: A0

Example:

```
ASCII: VWI

HEX:   A0
```

Errors: None

VWMATX (Viewing Matrix)

Purpose: Define the viewing matrix.

Command: VWMATX array

Description: VWMATX loads the viewing matrix directly from the 4-by-4 array.

Short Form: VWM array

Hex Format:

```
A7 lowm11 highm11 lowfracm11 highfracm11
   lowm12 highm12 lowfracm12 highfracm12
   lowm13 highm13 lowfracm13 highfracm13
   lowm14 highm14 lowfracm14 highfracm14
   lowm21 highm21 lowfracm21 highfracm21
   lowm22 highm22 lowfracm22 highfracm22
   lowm23 highm23 lowfracm23 highfracm23
   lowm24 highm24 lowfracm24 highfracm24
   lowm31 highm31 lowfracm31 highfracm31
   lowm32 highm32 lowfracm32 highfracm32
   lowm33 highm33 lowfracm33 highfracm33
   lowm34 highm34 lowfracm34 highfracm34
   lowm41 highm41 lowfracm41 highfracm41
   lowm42 highm42 lowfracm42 highfracm42
   lowm43 highm43 lowfracm43 highfracm43
   lowm44 highm44 lowfracm44 highfracm44
```

Example:

```
ASCII: VWM 68      12.5    253     17.25
           65503.5 0       306.25  34
           8418    2628.25 1.75    0.5
           313.75  50.25   1       1.5

HEX:   A7 44 00 00 00 0C 00 00 80 FD 00
          00 00 11 00 00 40 DF FF 00 80
          00 00 00 00 32 01 00 40 22 00
          00 00 E2 20 00 00 44 0A 00 40
          01 00 00 C0 00 00 00 80 39 01
          00 C0 32 00 00 40 01 00 00 00
          01 00 00 80
```

Errors: Arithmetic overflow

VWPORT (Viewport)

Purpose: Define a viewport.

Command: VWPORT x1 x2 y1 y2

Description: VWPORT defines a viewport within the viewplane and is measured in PELs from the lower-left corner of the screen. Clipping is always enabled. The default is the entire screen (0,639 and 0,479). *x1* must be less than *x2*; otherwise, a warning is generated and the coordinates are swapped. The same is true for *y1* and *y2*. A warning is generated if any of the coordinates fall outside the screen boundary.

Short Form: VWP x1 x2 y1 y2

Hex Format: B2 lowx1 highx1 lowx2 highx2
lowy1 highy1 lowy2 highy2

Example:

```
ASCII: VWP 50 450 30 250

HEX:   B2 32 00 C4 01 1E 00 FA 00
```

Errors: Arithmetic overflow

VWROTX (Viewing Rotate X Axis)

Purpose: Rotate viewing about the x axis.

Command: VWROTX deg

Description: VWROTX defines the rotation about the x axis component of the viewing matrix.

Short Form: VWX deg

Hex Format: A3 lowdeg highdeg

Example:

```
ASCII: VWX 30

HEX:   A3 2D 00
```

Errors: Arithmetic overflow

VWROTY (Viewing Rotate Y Axis)

Purpose: Rotate viewing about the y axis.

Command: VWROTY deg

Description: VWROTY defines the rotation about the y axis component of the viewing matrix.

Short Form: VWY deg

Hex Format: A4 lowdeg highdeg

Example:

```
ASCII: VWY 45

HEX:   A4 1E 00
```

Errors: Arithmetic overflow

VWROTZ (Viewing Rotate Z Axis)

Purpose: Rotate viewing about the z axis.

Command: VWROTZ deg

Description: VWROTZ defines the rotation about the z axis component of the viewing matrix.

Short Form: VWZ deg

Hex Format: A5 lowdeg highdeg

Example:

```
ASCII: VWZ 30

HEX:   A5 44 00
```

Errors: Arithmetic overflow

VWRPT (Viewing Reference Point)

Purpose: Define the viewing reference point.

Command: VWRPT x y z

Description: VWRPT defines the viewing reference point (the point the user is looking at); specified by *x,y,z*.

Short Form: VWR x y z

Hex Format:

```
A1 lowx  highx  lowfracx   highfracx
   lowy  highy  lowfracy   highfracy
   lowz  highz  lowfracz   highfracz
```

Example:

```
ASCII: VWR 50 75 -25

HEX:   A1 32 00 00 00 4B 00 00 00 E7 FF 00 00
```

Errors: Arithmetic overflow

WAIT (Wait)

Purpose: Insert a delay in execution.

Command: WAIT frames

Description: WAIT inserts a delay in the execution of commands by waiting the number of frames specified by *frames*. A frame is 1/60 second. With the maximum of 65535 frames, a delay of up to 20 minutes may be inserted.

Short form: W frames

Hex Format: 05 lowframes highframes

Example:

```
ASCII: W 60

HEX:   05 3C 00
```

Errors: None

WINDOW (Window)

Purpose: Define the viewport coordinates.

Command: WINDOW x1 x2 y1 y2

Description: WINDOW defines the corner coordinates of the viewport. These two-dimensional real coordinates will map to the screen's PEL locations specified by the most recent VWPORT command.

Short Form: WI x1 x2 y1 y2

Hex Format:

B3	lowxleft	highxleft
	lowfracxleft	highfracxleft
	lowxright	highxright
	lowfracxright	highfracxright
	lowybottom	highybottom
	lowfracybottom	highfracybottom
	lowytop	highytop
	lowfracytop	highfracytop

Example:

```
ASCII: WI -100 100 100 100

HEX:   B3 96 FF 00 00 64 00 00 00
          64 00 00 00 64 00 00 00
```

Errors: Arithmetic overflow

Run-Length Encoding

In hexadecimal mode, the commands IMAGER and IMAGEW send and receive data in run-length encoded format. This format allows for extremely high data rates. The format is described as follows:

Command (1 byte) IMAGER or IMAGEW
Line # (1 word)
Start x
End x
One or more PEL packets

A PEL packet is either of the following:

- A solid block of one color:

  Count (1 byte: N - 1)
  Color (1 byte)
  The count may range from 0 to 127 (N = 1 to 128), with the most-significant bit set to 0. This packet defines multiple occurrences of the same color and requires only two bytes to specify up to 128 PELs.

- PELs of different colors:

  Count (1 byte: N - 1 + 128)
  PEL 0
  PEL 1
  PEL 2
  . . . .
  PEL N - 1 (N bytes)
  The count may range from 128 to 255 (N = 1 to 128), with the most-significant bit set to 1. This packet defines strings of color codes that are different from one another.

Default LUT Selections for LUTINT

Each state provides a distinct way for initializing the look-up table (LUT). Following are descriptions for each currently defined state. The descriptions include a list of the default values for that LUT.

State 0

State 0 reproduces a color-cone distribution. The 8-bit LUT value divides into two 4-bit hexadecimal digits. The least-significant digit supplies the luminance value, and the most-significant digit supplies the color scale, each of the 16 values corresponding to a color. The color scale shades from black through the given color to white.

The following table shows the default values of state 0 for the various colors.

Color	Default Values (in Hex) for State 0							
Black to Grey to White	000	111	222	333	444	555	666	777
	888	999	AAA	BBB	CCC	DDD	EEE	FFF
Black to Red to White	000	200	400	600	800	A00	C00	E00
	F00	F22	F44	F66	F88	FAA	FCC	FEE
Black to Red-magenta to White	000	201	402	603	904	A05	C06	E07
	F08	F29	F4A	F6B	F8C	FAD	FCE	FEF
Black to Magenta to White	000	202	404	606	808	A0A	C0C	E0E
	F0F	F2F	F4F	F6F	F8F	FAF	FCF	FEF
Black to Magenta-blue to White	000	102	204	306	408	50A	60C	70E
	80F	92F	A4F	B6F	C8F	DAF	ECF	FEF
Black to Blue to White	000	002	004	006	008	00A	00C	00E
	00F	22F	44F	66F	88F	AAF	CCF	EEF
Black to Blue-cyan to White	000	012	042	036	048	05A	06C	07E
	08F	29F	4AF	6BF	8CF	ADF	CEF	EFF
Black to Cyan to White	000	022	044	066	088	0AA	0CC	0EE
	0FF	2FF	4FF	6FF	8FF	AFF	CFF	EFF
Black to Cyan-green to White	000	021	042	063	084	0A5	0C6	0E7
	0F8	2F9	4FA	6FB	8FC	AFD	DFE	EFF
Black to Green to White	000	020	040	060	080	0A0	0C0	0E0
	0F0	2F2	4F4	6F6	8F8	AFA	CFC	EFE
Black to Green-yellow to White	000	120	240	360	480	5A0	6C0	7E0
	8F0	9F2	AF4	BF6	CF8	DFA	EFC	EFF
Black to Yellow to White	000	220	440	660	880	AA0	CC0	EE0
	FF0	FF2	FF4	FF6	FF8	FFA	FFC	FFE
Black to Yellow-red to White	000	210	420	630	840	A50	C60	E70
	F80	F92	FA4	FB6	FC8	FDA	FEC	FFE
Black to Unsaturated Red to White	000	211	422	633	844	A55	C66	E77
	F88	F99	FAA	FBB	FCC	FDD	FEE	FFF
Black to Unsaturated Green to White	000	121	242	363	484	5A5	6C6	7E7
	8F8	9F9	AFA	BFB	CFC	DFD	EFE	FFF
Black to Unsaturated Blue to White	000	112	224	336	448	55A	66C	77E
	88F	99F	AAF	BBF	CCF	DDF	EEF	FFF

State 1

State 1 divides the 8-bit LUT value into two 4-bit hexadecimal digits. The least-significant digit provides the background color, and the most-significant digit defines the foreground color. The high-function graphics mode interprets a value of 0000 for the most-significant digit as a transparent foreground, allowing the background color to be displayed. Otherwise, the high-function graphics mode ignores the background color.

The following table lists the colors represented by each 4-bit value for State 1.

Value	Color	RGB
0	Sky Blue (background only)	68D
1	Black	000
2	Dark Brown	742
3	Light Brown	A74
4	Dark Red	700
5	Light Red	F00
6	Orange	F70
7	Yellow	FF0
8	Yellow-Green	AF0
9	Light Green	0F0
A	Dark Green	070
B	Green-Blue	077
C	Dark Blue	007
D	Light Burnt-Sienna	E96
E	Grey	777
F	White	FFF

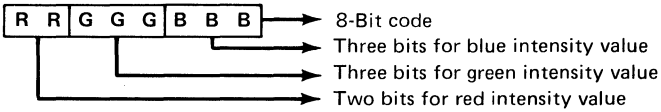
States 2 through 4

For states 2 through 4, red, green, and blue LUT values employ either two or three bits of information. For each state, one color receives two bits while the other two colors each receive three. Each bit value then translates to an RGB intensity of that color. The following tables give the corresponding intensity values for each bit value.

2-Bit Intensity Values		
Decimal Value	Bit Value	Intensity Level
0	0 0	0
1	0 1	5
2	1 0	10
3	1 1	15

3-Bit Intensity Values		
Decimal Value	Bit Value	Intensity Level
0	0 0 0	0
1	0 0 1	3
2	0 1 0	5
3	0 1 1	7
4	1 0 0	9
5	1 0 1	11
6	1 1 0	13
7	1 1 1	15

State 2 uses two bits for red (R), three bits for green (G), and three bits for blue (B). Thus, R R G G G B B B means:


R R G G G B B B
8-Bit code
Three bits for blue intensity value
Three bits for green intensity value
Two bits for red intensity value

Similarly, state 3 uses two bits for green and three bits each for red and blue (R R R G G B B B). State 4 allows two bits for blue and three bits each for red and green (R R R G G G B B).

The following table shows the default values for state 2.

Default Values (in Hex) for State 2							
000	003	005	007	009	00B	00D	00F
030	033	035	037	039	03B	03D	03F
050	053	055	057	059	05B	05D	05F
070	073	075	077	079	07B	07D	07E
090	093	095	097	099	09B	09D	09F
0B0	0B3	0B5	0B7	0B9	0BB	0BD	0BF
0D0	0D3	0D5	0D7	0D9	0DB	0DD	0DF
0F0	0F3	0F5	0F7	0F9	0FB	0FD	0FF
050	503	505	507	509	50B	50D	50F
530	533	535	537	539	53B	53D	53F
550	553	555	557	559	55B	55D	55F
570	573	575	577	579	57B	57D	57F
590	593	595	597	599	59B	59D	59F
5B0	5B3	5B5	5B7	5B9	5BB	5BD	5BF
5D0	5D3	5D5	5D7	5D9	5DB	5DD	5DF
5F0	5F3	5F5	5F7	5F9	5FB	5FD	5FF
A00	A03	A05	A07	A09	A0B	A0D	A0F
A30	A33	A35	A37	A39	A3B	A3D	A3F
A50	A53	A55	A57	A59	A5B	A5D	A5F
A70	A73	A75	A77	A79	A7B	A7D	A7F
A90	A93	A95	A97	A99	A9B	A9D	A9F
AB0	AB3	AB5	AB7	AB9	ABB	ABD	ABF
AD0	AD3	AD5	AD7	AD9	ADB	ADD	ADF
AF0	AF3	AF5	AF7	AF9	AFB	AFD	AFF
F00	F03	F05	F07	F09	F0B	F0D	F0F
F30	F33	F35	F37	F39	F3B	F3D	F3F
F50	F53	F55	F57	F59	F5B	F5D	F5F
F70	F73	F75	F77	F79	F7B	F7D	F7F
F90	F93	F95	F97	F99	F9B	F9D	F9F
FB0	FB3	FB5	FB7	FB9	FBB	FBD	FBF
FD0	FD3	FD5	FD7	FD9	FDB	FDD	FDF
FF0	FF3	FF5	FF7	FF9	FFB	FFD	FFF

The following table shows the default values for state 3.

Default Values (in Hex) for State 3							
000	003	005	007	009	00B	00D	00F
050	053	055	057	059	05B	05D	05F
0A0	0A3	0A5	0A7	0A9	0AB	0AD	0AF
0F0	0F3	0F5	0F7	0F9	0FB	0FD	0FF
300	303	305	307	309	30B	30D	30F
350	353	355	357	359	35B	35D	35F
3A0	3A3	3A5	3A7	3A9	3AB	3AD	3AF
3F0	3F3	3F5	3F7	3F9	3FB	3FD	3FF
500	503	505	507	509	50B	50D	50F
550	553	555	557	559	55B	55D	55F
5A0	5A3	5A5	5A7	5A9	5AB	5AD	5AF
5F0	5F3	5F5	5F7	5F9	5FB	5FD	5FF
700	703	705	707	709	70B	70D	70F
750	753	755	757	759	75B	75D	75F
7A0	7A3	7A5	7A7	7A9	7AB	7AD	7AF
7F0	7F3	7F5	7F7	7F9	7FB	7FD	7FF
900	903	905	907	909	90B	90D	90F
950	953	955	957	959	95B	95D	95F
9A0	9A3	9A5	9A7	9A9	9AB	9AD	9AF
9F0	9F3	9F5	9F7	9F9	9FB	9FD	9FF
B00	B03	B05	B07	B09	B0B	B0D	B0F
B50	B53	B55	B57	B59	B5B	B5D	B5F
BA0	BA3	BA5	BA7	BA9	BAB	BAD	BAF
BF0	BF3	BF5	BF7	BF9	BFB	BFD	BFF
D00	D03	D05	D07	D09	D0B	D0D	D0F
D50	D53	D55	D57	D59	D5B	D5D	D5F
DA0	DA3	DA5	DA7	DA9	DAB	DAD	DAF
DF0	DF3	DF5	DF7	DF9	DFB	DFD	DFF
F00	F03	F05	F07	F09	F0B	F0D	F0F
F50	F53	F55	F57	F59	F5B	F5D	F5F
FA0	FA3	FA5	FA7	FA9	FAB	FAD	FAF
FF0	FF3	FF5	FF7	FF9	FFB	FFD	FFF

The following table shows the default values for state 4.

Default Values (in Hex) for State 4							
000	005	00A	00F	030	035	03A	03F
050	055	05A	05F	070	075	07A	07F
090	095	09A	09F	0B0	0B5	0BA	0BF
0D0	0D5	0DA	0DF	0F0	0F5	0FA	0FF
300	305	30A	30F	330	335	33A	33F
350	355	35A	35F	370	375	37A	37F
390	395	39A	39F	3B0	3B5	3BA	3BF
3D0	3D5	3DA	3DF	3F0	3F5	3FA	3FF
500	505	50A	50F	530	535	53A	53F
550	555	55A	55F	570	575	57A	57F
590	595	59A	59F	5B0	5B5	5BA	5BF
5D0	5D5	5DA	5DF	5F0	5F5	5FA	5FF
700	705	70A	70F	730	735	73A	73F
750	755	75A	75F	770	775	77A	77F
790	795	79A	79F	7B0	7B5	7BA	7BF
7D0	7D5	7DA	7DF	7F0	7F5	7FA	7FF
900	905	90A	90F	930	935	93A	93F
950	955	95A	95F	970	975	97A	97F
990	995	99A	99F	9B0	9B5	9BA	9BF
9D0	9D5	9DA	9DF	9F0	9F5	9FA	9FF
B00	B05	B0A	B0F	B30	B35	B3A	B3F
B50	B55	B5A	B5F	B70	B75	B7A	B7F
B90	B95	B9A	B9F	BB0	BB5	BBA	BBF
BD0	BD5	BDA	BDF	BF0	BF5	BFA	BFF
D00	D05	D0A	D0F	D30	D35	D3A	D3F
D50	D55	D5A	D5F	D70	D75	D7A	D7F
D90	D95	D9A	D9F	DB0	DB5	DBA	DBF
DD0	DD5	DDA	DDF	DF0	DF5	DFA	DFF
F00	F05	F0A	F0F	F30	F35	F3A	F3F
F50	F55	F5A	F5F	F70	F75	F7A	F7F
F90	F05	F9A	F9F	FB0	FB5	FBA	FBF
FD0	FD5	FDA	FDF	FF0	FF5	FFA	FFF

State 5

In state 5, the 8-bit value becomes the arithmetic result of the formula (R x 36) + (G x 6) + B, where R, G, and B represent coded values of intensity levels ranging from 0 to 5. The following table defines which coded values correspond to which intensity levels.

Coded RGB Values	Actual Intensity Levels
0	0
1	3
2	6
3	9
4	12
5	15

The following table shows the default values for state 5:

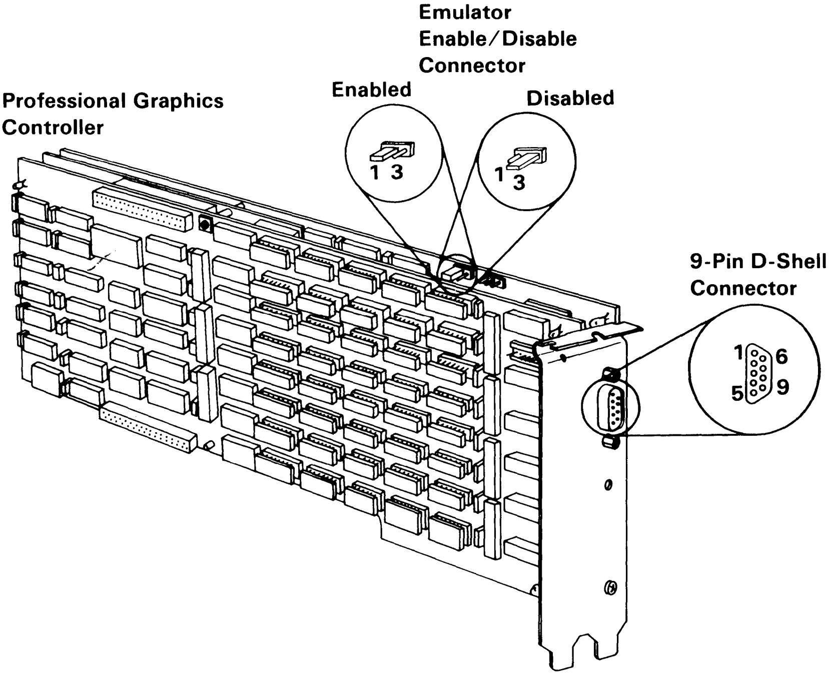
Default Values (in Hex) for State 5							
000	003	006	009	00C	00F	030	033
036	039	03C	03F	060	063	066	069
06C	06F	090	093	096	099	09C	09F
0C0	0C3	0C6	0C9	0CC	0cF	0F0	0F3
0F6	0F9	0FC	0FF	300	303	306	309
30C	30F	330	333	336	339	33C	33F
360	363	366	369	36C	36F	390	393
396	399	39C	39F	3C0	3C3	3C6	3C9
3CC	3Cf	3F0	3F3	3F6	3F9	3FC	3FF
600	603	606	609	60C	60F	630	633
636	639	63C	63F	660	663	666	669
66C	66F	690	693	696	699	69C	69F
6C0	6C3	6C6	6C9	6CC	6CF	6F0	6F3
6F6	6F9	6FC	6FF	900	903	906	909
90C	90F	930	933	936	939	93C	93F
960	999	99C	99F	9C0	9C3	9C6	9C9
996	999	99C	99F	9C0	9C3	9C6	9C9
9CC	9CF	9F0	9F3	9F6	9F9	9FC	9FF
C00	C03	C06	C09	C0c	C0F	C30	C33
C36	C39	C3C	C3F	C60	C63	C66	C69
C6C	C6F	C90	C93	C96	C99	C9C	C9F
CC0	CC3	CC6	CC9	CCC	CCF	CF0	CF3
CF6	CF9	CFC	CFF	F00	F03	F06	F09
F0C	F0F	F30	F33	F36	F39	F3C	F3F
F60	F99	F9C	F9F	FC0	Fc3	FC6	FC9
F96	F99	F9C	F9F	FC0	FC3	FC6	FC9
FCC	FCF	FF0	FF3	FF6	FF9	FFC	FFF
000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000

State 255

State 255 restores the LUT values that were previously saved with the command LUTSAV. These tables can include user-defined values.

Interface

The following illustration shows the location of the connectors and jumper on the Professional Graphics Controller.



Connector Specifications

The following table shows the pin numbers and their respective signals.

	Signal Name/Description	Pin	
Professional Graphics Display	Red Video	1	Professional Graphics Controller
	Green Video	2	
	Blue Video	3	
	Horizontal and Vertical Sync	4	
	Mode Control	5	
	Ground for Pin 1	6	
	Ground for Pin 2	7	
	Ground for Pin 3	8	
	Ground for Pins 4 and 5	9	

Specifications

The following is a description of the Professional Graphics Controller specifications.

Size:

Length: 668 mm (4.2 in.)

Depth: 32 mm (1.26 in.)

Height: 210 mm (3.36 in.)

Weight: 90.72 kg (2 lb)

Power Requirements:

Voltage: 5 VDC (+/-5%)

Current: 5 A Maximum

Power Dissipation: 25 W Maximum

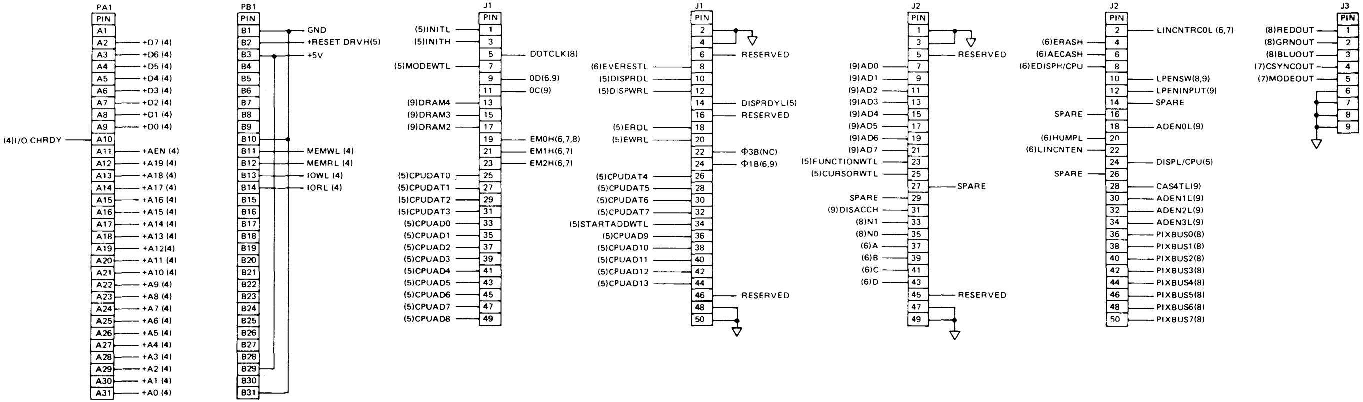
Notes:

Logic Diagrams

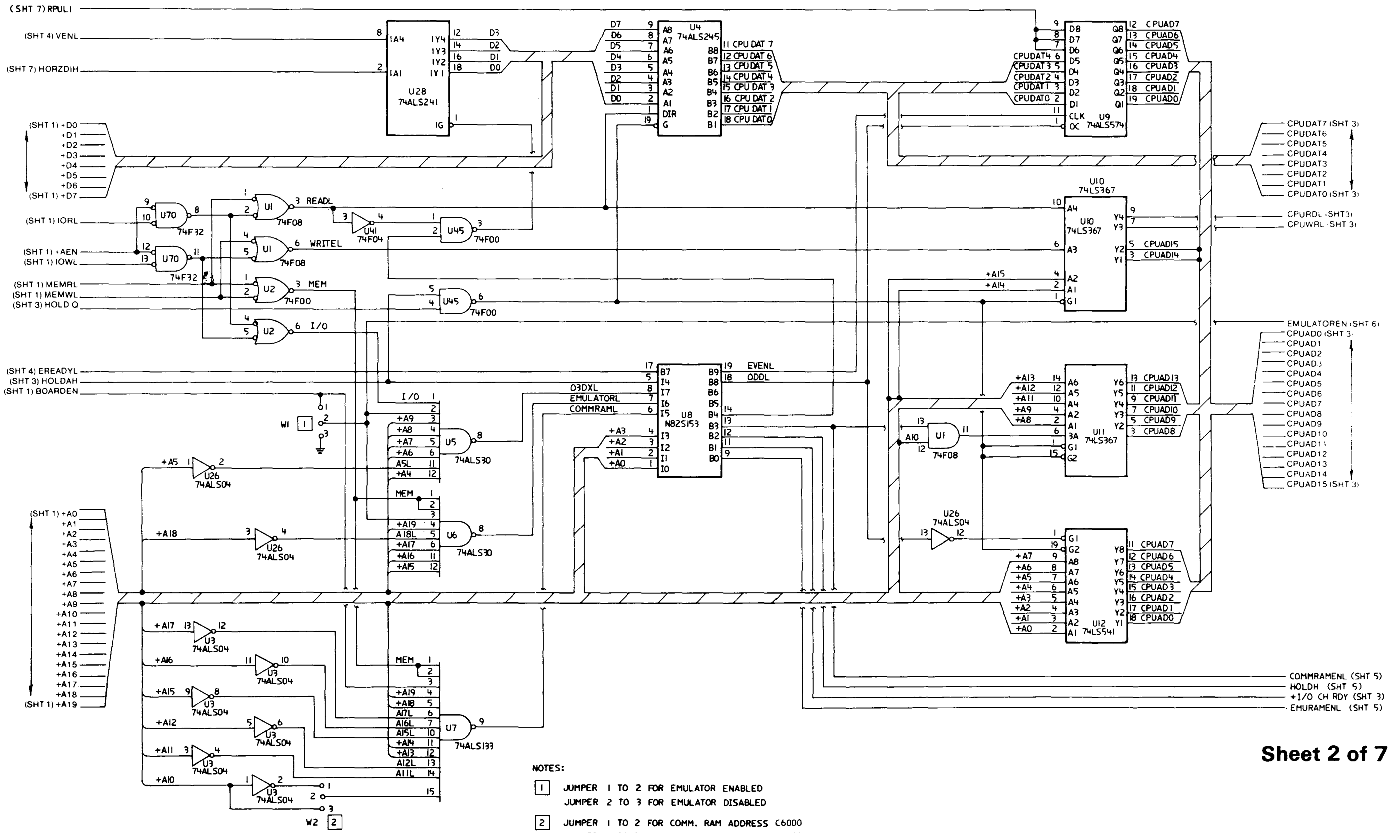
This section shows the logic diagrams for:

- Professional Graphics Controller's processor card
- Professional Graphics Controller's emulator card
- Professional Graphics Controller's memory card

August 15, 1984



Sheet 1 of 7

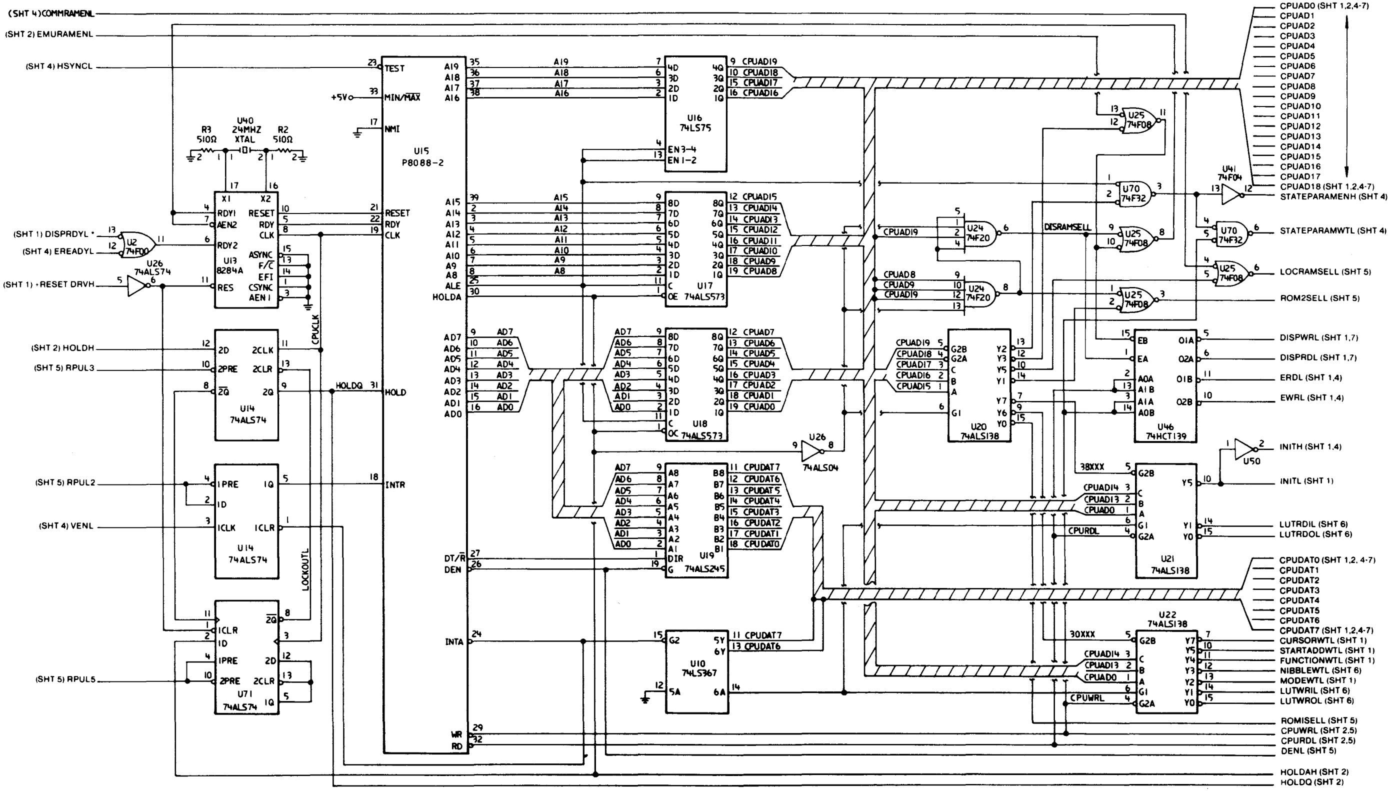


NOTES:

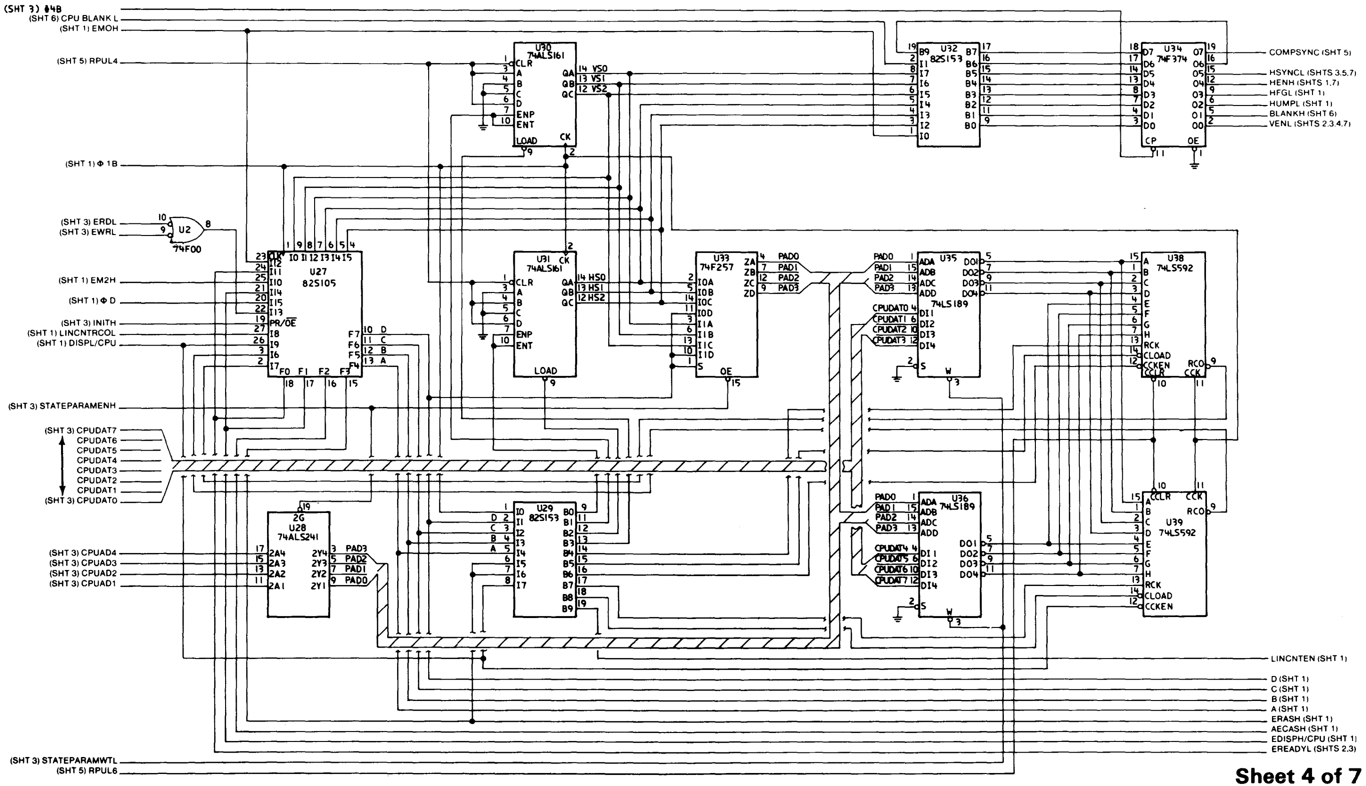
1 JUMPER 1 TO 2 FOR EMULATOR ENABLED
JUMPER 2 TO 3 FOR EMULATOR DISABLED

2 JUMPER 1 TO 2 FOR COMM. RAM ADDRESS C6000
JUMPER 2 TO 3 FOR COMM. RAM ADDRESS C6400

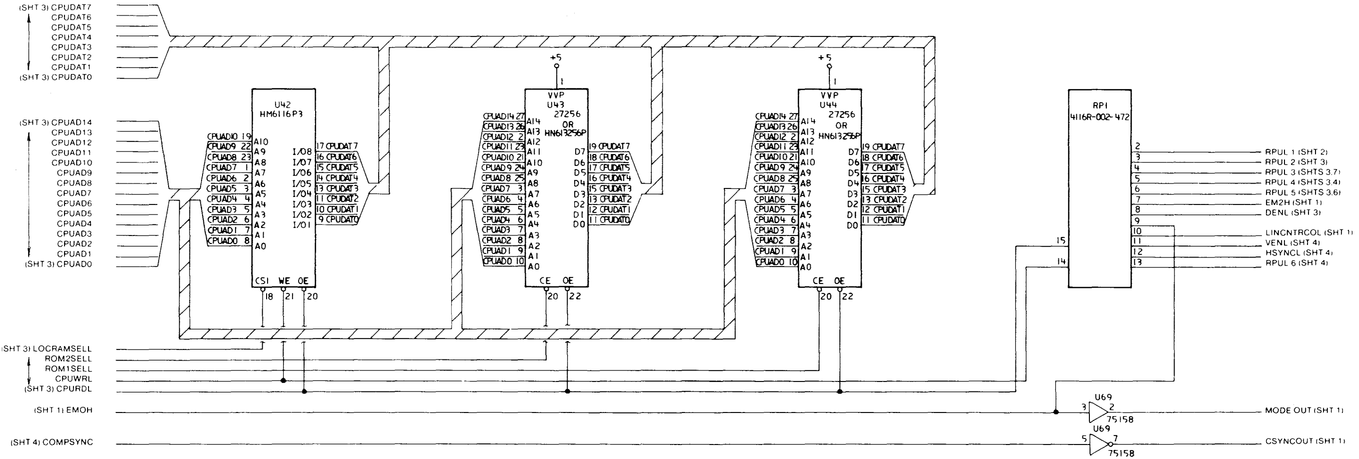
Sheet 2 of 7


(SHT 4) COMMRAMENL
(SHT 2) EMURAMENL
(SHT 4) HSYNCL
+5V
U40
24MHZ
XTAL
R3
510Ω
R2
510Ω
X1
X2
RDY1
AEN2
RDY2
RES
RESET
RDY
CLK
ASYNC
F/C
EFI
CSYNC
AEN1
U13
8284A
(SHT 1) DISPRDYL *
(SHT 4) EREADYL
U2
74F00
U26
74ALS74
(SHT 1) *RESET DRVH
(SHT 2) HOLDH
(SHT 5) RPUL3
2D
2PRE
2Q
2CLK
2CLR
U14
74ALS74
CPUCLK
HOLDQ
(SHT 5) RPUL2
(SHT 4) VENL
1PRE
1D
1CLK
1Q
1CLR
LOCKOUTL
1CLR
1D
1PRE
2PRE
2Q
2D
2CLR
1Q
U71
74ALS74
(SHT 5) RPUL5
TEST
MIN/MAX
NMI
U15
P8088-2
RESET
RDY
CLK
HOLD
INTR
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A9
A8
ALE
HOLDA
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
DT/R
DEN
INTA
WR
RD
U16
74LS75
EN3-4
EN1-2
U17
74ALS573
U18
74ALS573
U19
74ALS245
DIR
G
U10
74LS367
U26
74ALS04
CPUAD19
CPUAD18
CPUAD17
CPUAD16
CPUAD15
CPUAD14
CPUAD13
CPUAD12
CPUAD11
CPUAD10
CPUAD9
CPUAD8
CPUAD7
CPUAD6
CPUAD5
CPUAD4
CPUAD3
CPUAD2
CPUAD1
CPUAD0
CPUDAT7
CPUDAT6
CPUDAT5
CPUDAT4
CPUDAT3
CPUDAT2
CPUDAT1
CPUDAT0
U24
74F20
DISRAMSELL
U25
74F08
U70
74F32
U41
74F04
U20
74ALS138
G2B
G2A
G1
U46
74HCT139
EB
EA
O1A
O2A
O1B
O2B
38XXX
U21
74ALS138
CPURDL
U50
30XXX
U22
74ALS138
CPUWRL
CPUAD0 (SHT 1,2,4-7)
CPUAD1
CPUAD2
CPUAD3
CPUAD4
CPUAD5
CPUAD6
CPUAD7
CPUAD8
CPUAD9
CPUAD10
CPUAD11
CPUAD12
CPUAD13
CPUAD14
CPUAD15
CPUAD16
CPUAD17
CPUAD18 (SHT 1,2,4-7)
STATEPARAMENH (SHT 4)
STATEPARAMWTL (SHT 4)
LOCRAMSELL (SHT 5)
ROM2SELL (SHT 5)
DISPWRL (SHT 1,7)
DISPRDL (SHT 1,7)
ERDL (SHT 1,4)
EWRL (SHT 1,4)
INITH (SHT 1,4)
INITL (SHT 1)
LUTRDIL (SHT 6)
LUTRDOL (SHT 6)
CPUDAT0 (SHT 1,2, 4-7)
CPUDAT1
CPUDAT2
CPUDAT3
CPUDAT4
CPUDAT5
CPUDAT6
CPUDAT7 (SHT 1,2,4-7)
CURSORWTL (SHT 1)
STARTADDWTL (SHT 1)
FUNCTIONWTL (SHT 1)
NIBBLEWTL (SHT 6)
MODEWTL (SHT 1)
LUTWRIL (SHT 6)
LUTWROL (SHT 6)
ROMISELL (SHT 5)
CPUWRL (SHT 2,5)
CPURDL (SHT 2,5)
DENL (SHT 5)
HOLDAH (SHT 2)
HOLDQ (SHT 2)

Sheet 3 of 7

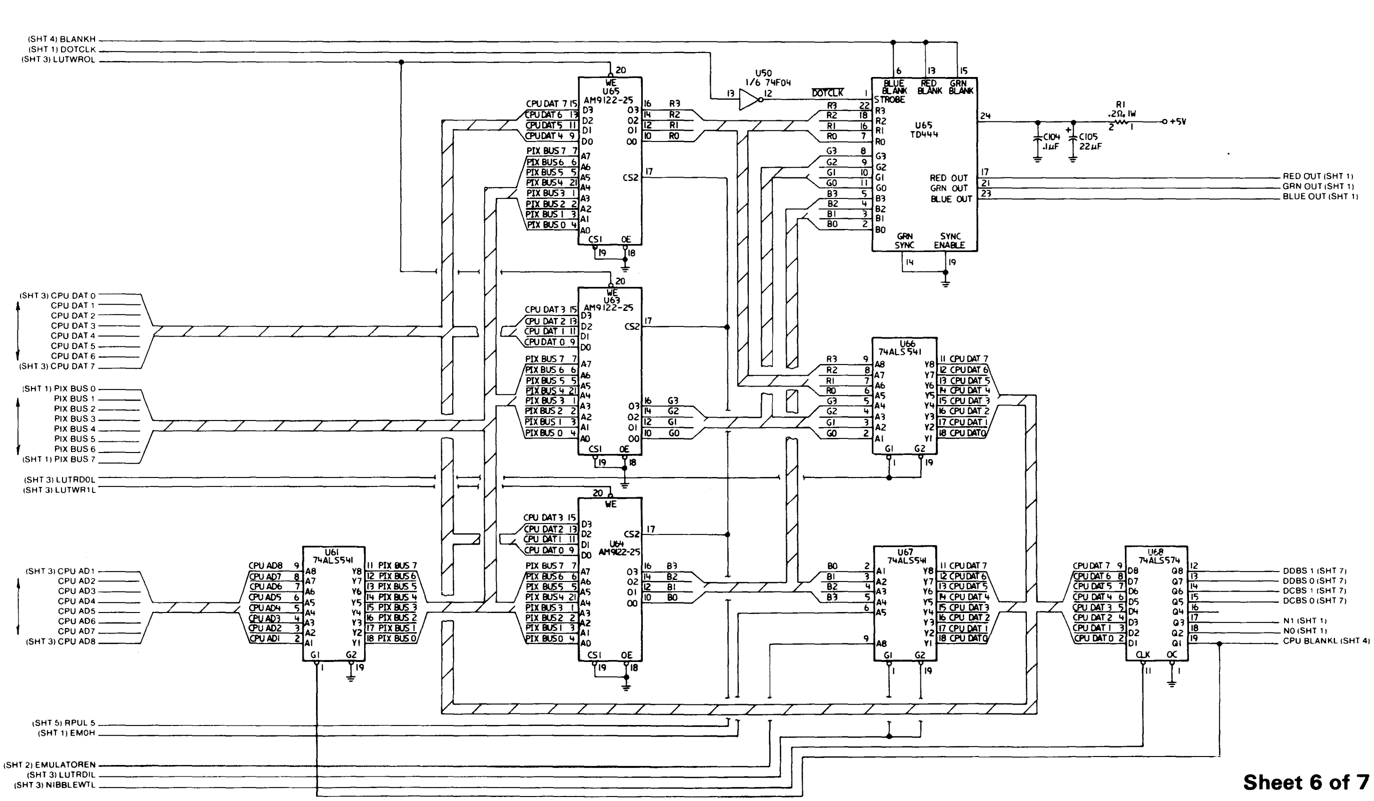

(SHT 3) ΦB
(SHT 6) CPU BLANK L
(SHT 1) EMOH
(SHT 5) RPUL4
(SHT 1) Φ 1B
(SHT 3) ERDL
(SHT 3) EWRL
U2
74F00
(SHT 1) EM2H
(SHT 1) Φ D
(SHT 3) INITH
(SHT 1) LINCNTRCOL
(SHT 1) DISPL/CPU
(SHT 3) STATEPARAMENH
(SHT 3) CPUDAT7
CPUDAT6
CPUDAT5
CPUDAT4
CPUDAT3
CPUDAT2
CPUDAT1
(SHT 3) CPUDAT0
(SHT 3) CPUAD4
(SHT 3) CPUAD3
(SHT 3) CPUAD2
(SHT 3) CPUAD1
(SHT 3) STATEPARAMWTL
(SHT 5) RPUL6
U30
74ALS161
U31
74ALS161
U27
82S105
U28
74ALS241
U29
82S153
U32
82S153
U33
74F257
U34
74F374
U35
74LS189
U36
74LS189
U38
74LS592
U39
74LS592
VS0
VS1
VS2
HS0
HS1
HS2
PAD0
PAD1
PAD2
PAD3
COMPSYNC (SHT 5)
HSYNCL (SHTS 3,5,7)
HENH (SHTS 1,7)
HFGL (SHT 1)
HUMPL (SHT 1)
BLANKH (SHT 6)
VENL (SHTS 2,3,4,7)
LINCNTEN (SHT 1)
D (SHT 1)
C (SHT 1)
B (SHT 1)
A (SHT 1)
ERASH (SHT 1)
AECASH (SHT 1)
EDISPH/CPU (SHT 1)
EREADYL (SHTS 2,3)

Sheet 4 of 7

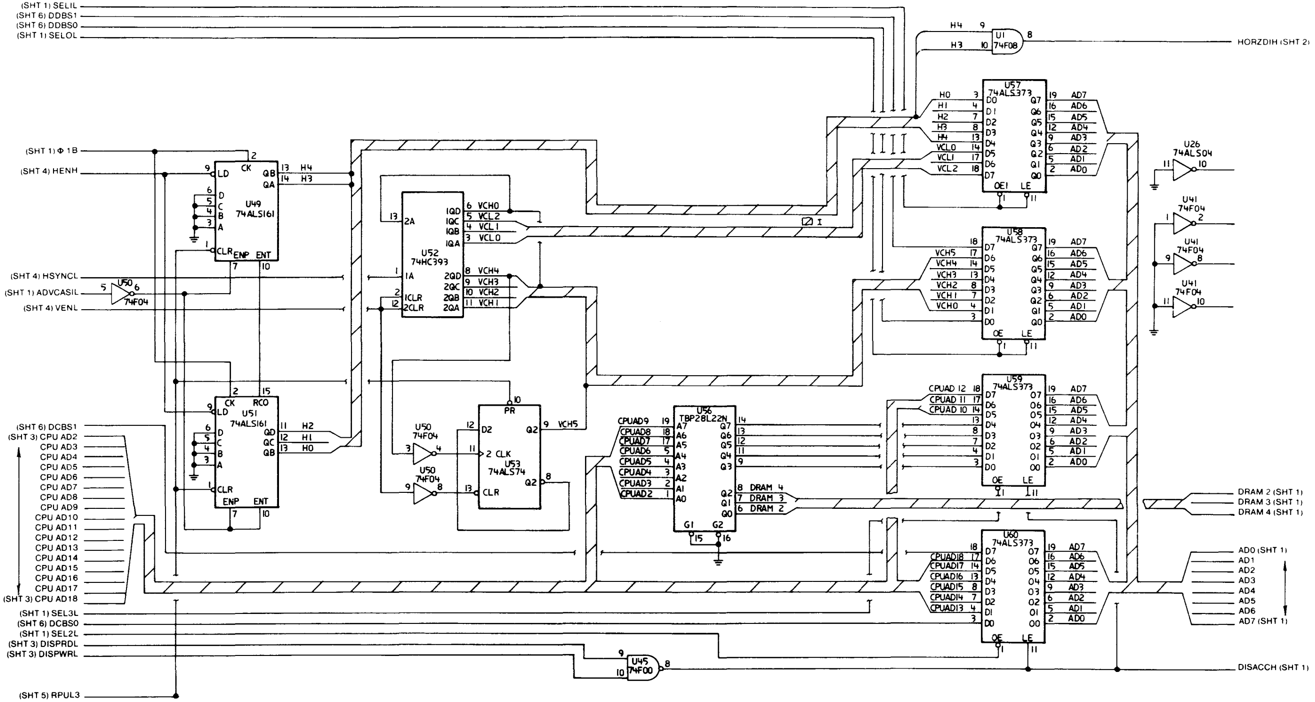


Sheet 5 of 7

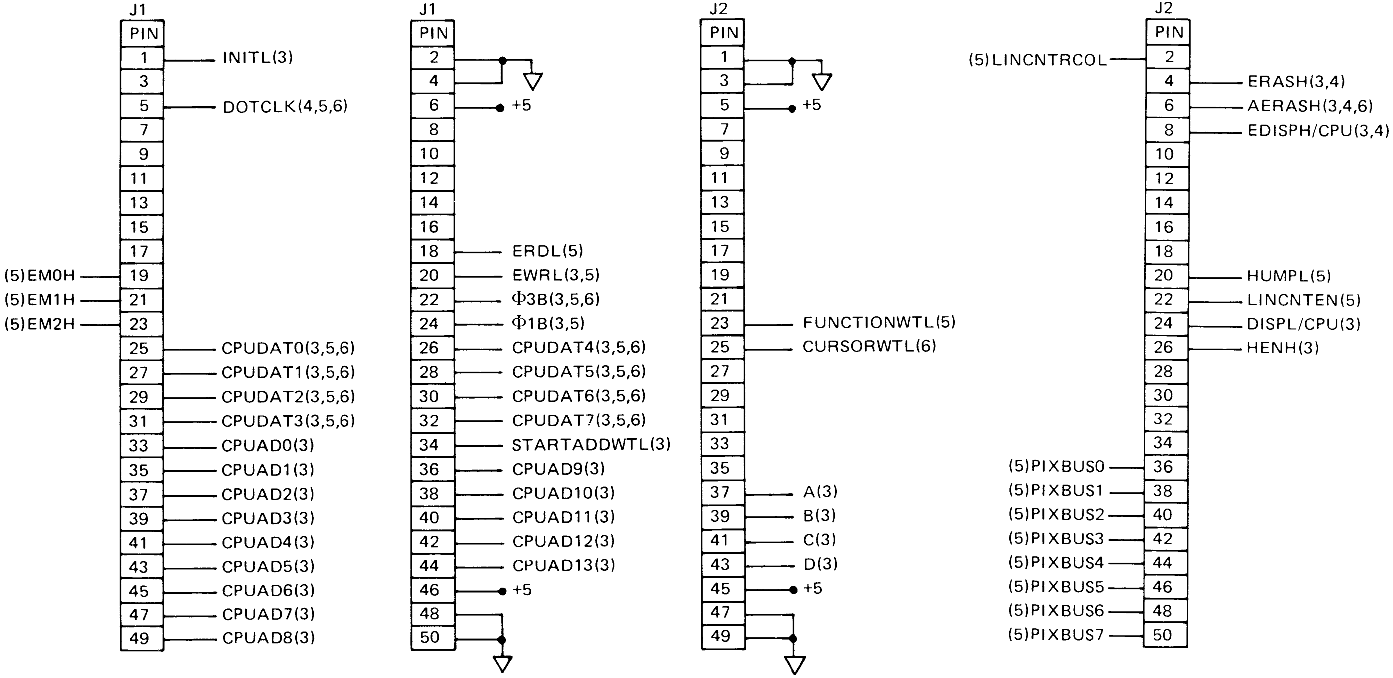
August 15, 1984


(SHT 4) BLANKH
(SHT 1) DOTCLK
(SHT 3) LUTWROL
U50 1/6 74F04
DOTCLK
U65 AM9122-25
U63 AM9122-25
U64 AM9122-25
WE
CS1
CS2
OE
D3
D2
D1
D0
A7
A6
A5
A4
A3
A2
A1
A0
O3
O2
O1
O0
CPU DAT 7
CPU DAT 6
CPU DAT 5
CPU DAT 4
CPU DAT 3
CPU DAT 2
CPU DAT 1
CPU DAT 0
PIX BUS 7
PIX BUS 6
PIX BUS 5
PIX BUS 4
PIX BUS 3
PIX BUS 2
PIX BUS 1
PIX BUS 0
R3
R2
R1
R0
G3
G2
G1
G0
B3
B2
B1
B0
U65 TD444
BLUE BLANK STROBE
RED BLANK
GRN BLANK
RED OUT
GRN OUT
BLUE OUT
GRN SYNC
SYNC ENABLE
R1 .2Ω, 1W
+5V
C104 .1μF
C105 22μF
RED OUT (SHT 1)
GRN OUT (SHT 1)
BLUE OUT (SHT 1)
(SHT 3) CPU DAT 0
CPU DAT 1
CPU DAT 2
CPU DAT 3
CPU DAT 4
CPU DAT 5
CPU DAT 6
(SHT 3) CPU DAT 7
(SHT 1) PIX BUS 0
PIX BUS 1
PIX BUS 2
PIX BUS 3
PIX BUS 4
PIX BUS 5
PIX BUS 6
(SHT 1) PIX BUS 7
(SHT 3) LUTRD0L
(SHT 3) LUTWR1L
U66 74ALS541
U67 74ALS541
U61 74ALS541
U68 74ALS574
(SHT 3) CPU AD1
CPU AD2
CPU AD3
CPU AD4
CPU AD5
CPU AD6
CPU AD7
(SHT 3) CPU AD8
DDBS 1 (SHT 7)
DDBS 0 (SHT 7)
DCBS 1 (SHT 7)
DCBS 0 (SHT 7)
N1 (SHT 1)
N0 (SHT 1)
CPU BLANKL (SHT 4)
(SHT 5) RPUL 5
(SHT 1) EMOH
(SHT 2) EMULATOREN
(SHT 3) LUTRDIL
(SHT 3) NIBBLEWTL

Sheet 6 of 7

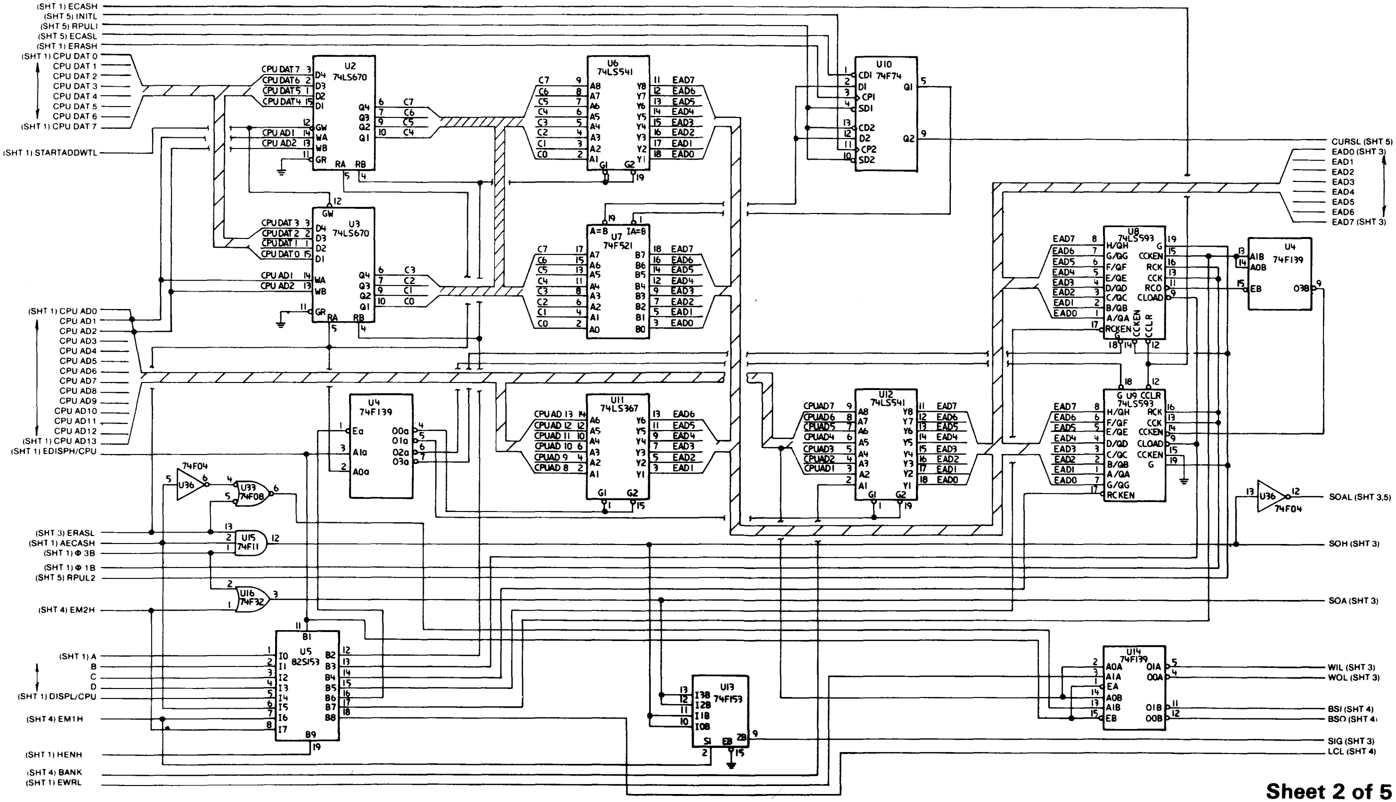

(SHT 1) SEL1L
(SHT 6) DDBS1
(SHT 6) DDBS0
(SHT 1) SEL0L
(SHT 1) Φ 1B
(SHT 4) HENH
(SHT 4) HSYNCL
(SHT 1) ADVCASIL
(SHT 4) VENL
(SHT 6) DCBS1
(SHT 3) CPU AD2
CPU AD3
CPU AD4
CPU AD5
CPU AD6
CPU AD7
CPU AD8
CPU AD9
CPU AD10
CPU AD11
CPU AD12
CPU AD13
CPU AD14
CPU AD15
CPU AD16
CPU AD17
(SHT 3) CPU AD18
(SHT 1) SEL3L
(SHT 6) DCBS0
(SHT 1) SEL2L
(SHT 3) DISPRDL
(SHT 3) DISPWRL
(SHT 5) RPUL3
U49
74ALS161
U51
74ALS161
LD
CK
QB
QA
D
C
B
A
CLR
ENP
ENT
RCO
QD
QC
H4
H3
H2
H1
H0
U50
74F04
U52
74HC393
1QD
1QC
1QB
1QA
2QD
2QC
2QB
2QA
1A
2A
1CLR
2CLR
VCH0
VCL2
VCL1
VCL0
VCH4
VCH3
VCH2
VCH1
U53
74ALS74
PR
D2
2 CLK
CLR
Q2
VCH5
U56
TBP28L22N
CPUAD9
CPUAD8
CPUAD7
CPUAD6
CPUAD5
CPUAD4
CPUAD3
CPUAD2
A7
A6
A5
A4
A3
A2
A1
A0
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
G1
G2
DRAM 4
DRAM 3
DRAM 2
U1
74F08
HORZDIH (SHT 2)
U57
74ALS373
H0
H1
H2
H3
H4
VCL0
VCL1
VCL2
D0
D1
D2
D3
D4
D5
D6
D7
OE1
LE
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
U58
74ALS373
VCH5
VCH4
VCH3
VCH2
VCH1
VCH0
U59
74ALS373
CPUAD 12
CPUAD 11
CPUAD 10
U60
74ALS373
CPUAD18
CPUAD17
CPUAD16
CPUAD15
CPUAD14
CPUAD13
U26
74ALS04
U41
74F04
U45
74F00
DRAM 2 (SHT 1)
DRAM 3 (SHT 1)
DRAM 4 (SHT 1)
AD0 (SHT 1)
AD1
AD2
AD3
AD4
AD5
AD6
AD7 (SHT 1)
DISACCH (SHT 1)

Sheet 7 of 7

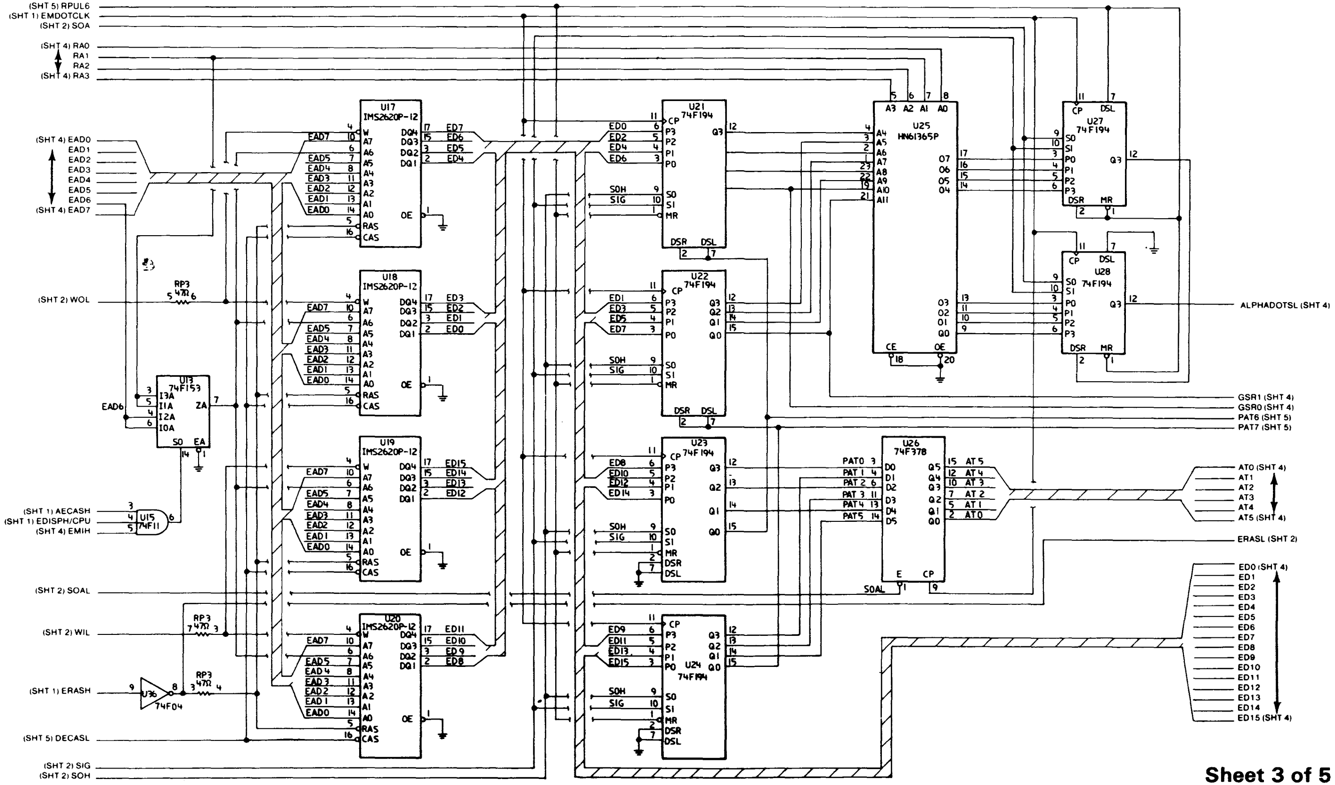

J1
PIN
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
INITL(3)
DOTCLK(4,5,6)
(5)EM0H
(5)EM1H
(5)EM2H
CPUDAT0(3,5,6)
CPUDAT1(3,5,6)
CPUDAT2(3,5,6)
CPUDAT3(3,5,6)
CPUAD0(3)
CPUAD1(3)
CPUAD2(3)
CPUAD3(3)
CPUAD4(3)
CPUAD5(3)
CPUAD6(3)
CPUAD7(3)
CPUAD8(3)
J1
PIN
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
+5
ERDL(5)
EWRL(3,5)
Φ3B(3,5,6)
Φ1B(3,5)
CPUDAT4(3,5,6)
CPUDAT5(3,5,6)
CPUDAT6(3,5,6)
CPUDAT7(3,5,6)
STARTADDWTL(3)
CPUAD9(3)
CPUAD10(3)
CPUAD11(3)
CPUAD12(3)
CPUAD13(3)
+5
J2
PIN
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
+5
FUNCTIONWTL(5)
CURSORWTL(6)
A(3)
B(3)
C(3)
D(3)
+5
J2
PIN
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
(5)LINCNTRCOL
ERASH(3,4)
AERASH(3,4,6)
EDISPH/CPU(3,4)
HUMPL(5)
LINCNTEN(5)
DISPL/CPU(3)
HENH(3)
(5)PIXBUS0
(5)PIXBUS1
(5)PIXBUS2
(5)PIXBUS3
(5)PIXBUS4
(5)PIXBUS5
(5)PIXBUS6
(5)PIXBUS7

Sheet 1 of 5

August 15, 1984

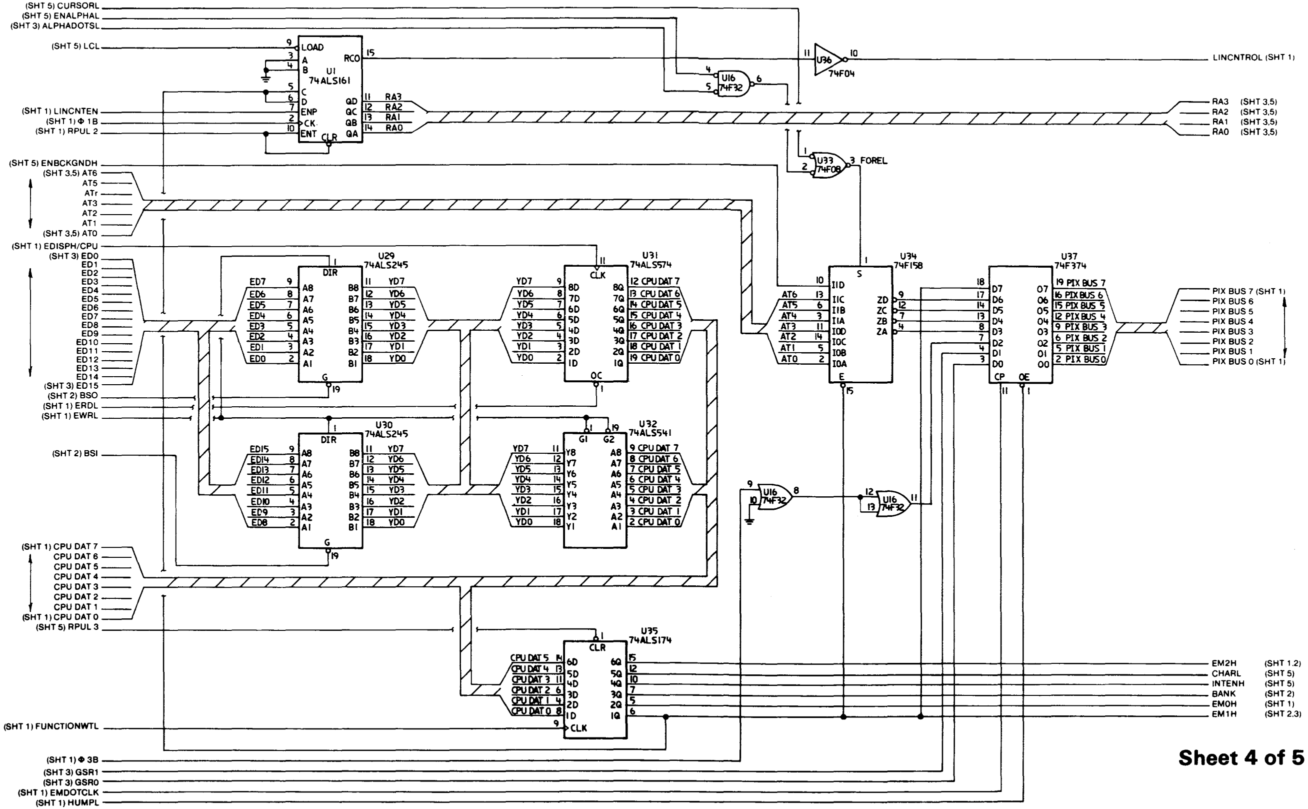


Sheet 2 of 5

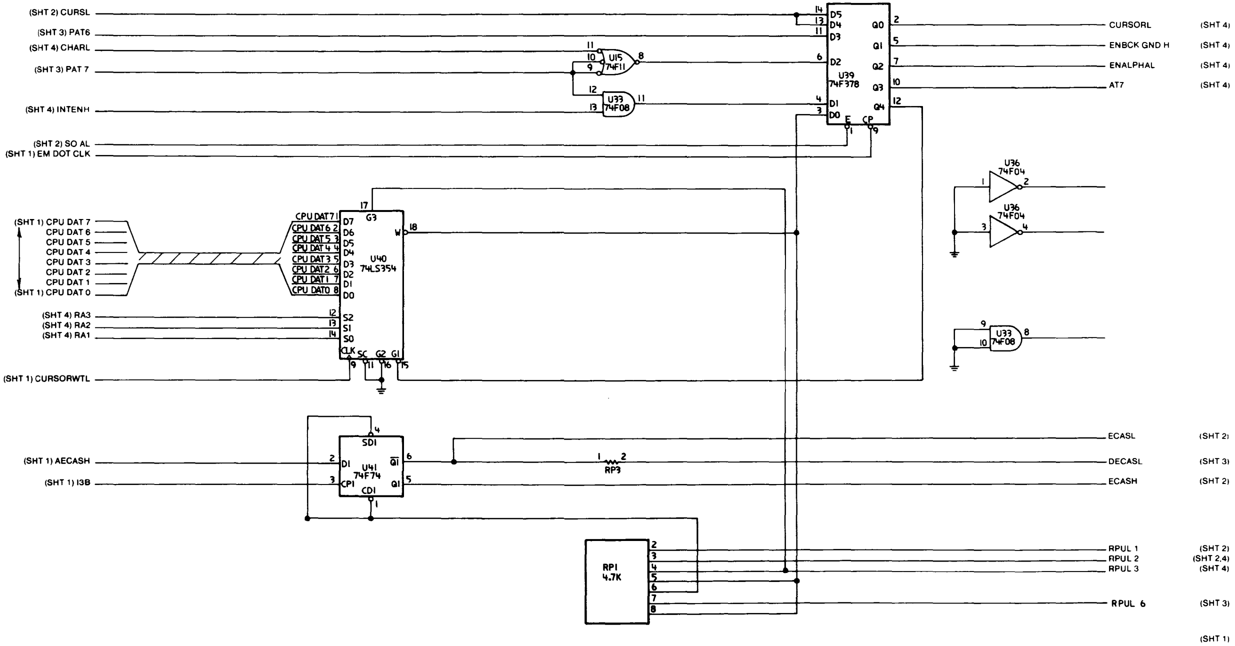

(SHT 5) RPUL6
(SHT 1) EMDOTCLK
(SHT 2) SOA
(SHT 4) RA0
RA1
RA2
(SHT 4) RA3
(SHT 4) EAD0
EAD1
EAD2
EAD3
EAD4
EAD5
EAD6
(SHT 4) EAD7
(SHT 2) WOL
RP3
47Ω
EAD6
U13
74F153
I3A
I1A
I2A
I0A
ZA
S0
EA
(SHT 1) AECASH
(SHT 1) EDISPH/CPU
(SHT 4) EMIH
U15
74F11
(SHT 2) SOAL
(SHT 2) WIL
(SHT 1) ERASH
U36
74F04
(SHT 5) DECASL
(SHT 2) SIG
(SHT 2) SOH
U17
IMS2620P-12
U18
IMS2620P-12
U19
IMS2620P-12
U20
IMS2620P-12
W
A7
A6
A5
A4
A3
A2
A1
A0
RAS
CAS
DQ4
DQ3
DQ2
DQ1
OE
ED7
ED6
ED5
ED4
ED3
ED2
ED1
ED0
ED15
ED14
ED13
ED12
ED11
ED10
ED9
ED8
U21
74F194
U22
74F194
U23
74F194
U24
74F194
CP
P3
P2
P1
P0
S0
S1
MR
DSR
DSL
Q3
Q2
Q1
Q0
SOH
SIG
U25
HN61365P
A3
A2
A1
A0
A4
A5
A6
A7
A8
A9
A10
A11
O7
O6
O5
O4
O3
O2
O1
O0
CE
OE
U27
74F194
U28
74F194
ALPHADOTSL (SHT 4)
GSR1 (SHT 4)
GSR0 (SHT 4)
PAT6 (SHT 5)
PAT7 (SHT 5)
U26
74F378
PAT0
PAT1
PAT2
PAT3
PAT4
PAT5
D0
D1
D2
D3
D4
D5
Q5
Q4
Q3
Q2
Q1
Q0
E
CP
SOAL
AT5
AT4
AT3
AT2
AT1
AT0
AT0 (SHT 4)
AT1
AT2
AT3
AT4
AT5 (SHT 4)
ERASL (SHT 2)
ED0 (SHT 4)
ED1
ED2
ED3
ED4
ED5
ED6
ED7
ED8
ED9
ED10
ED11
ED12
ED13
ED14
ED15 (SHT 4)

Sheet 3 of 5

August 15, 1984

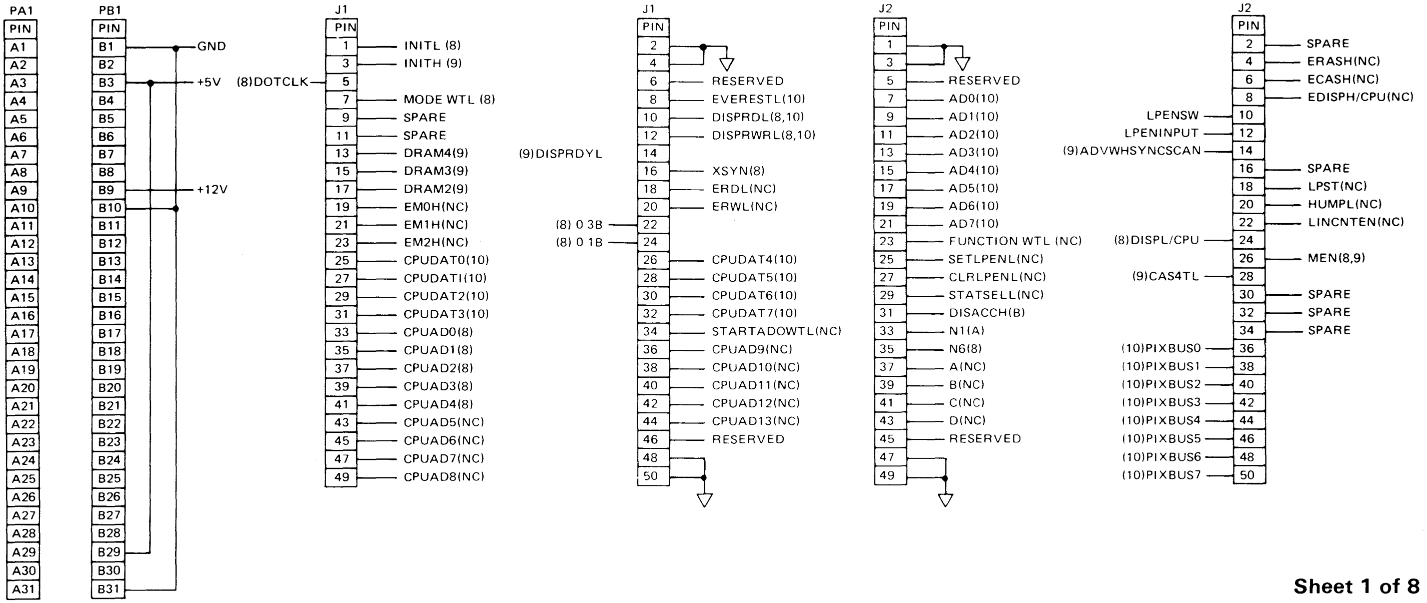


Sheet 4 of 5

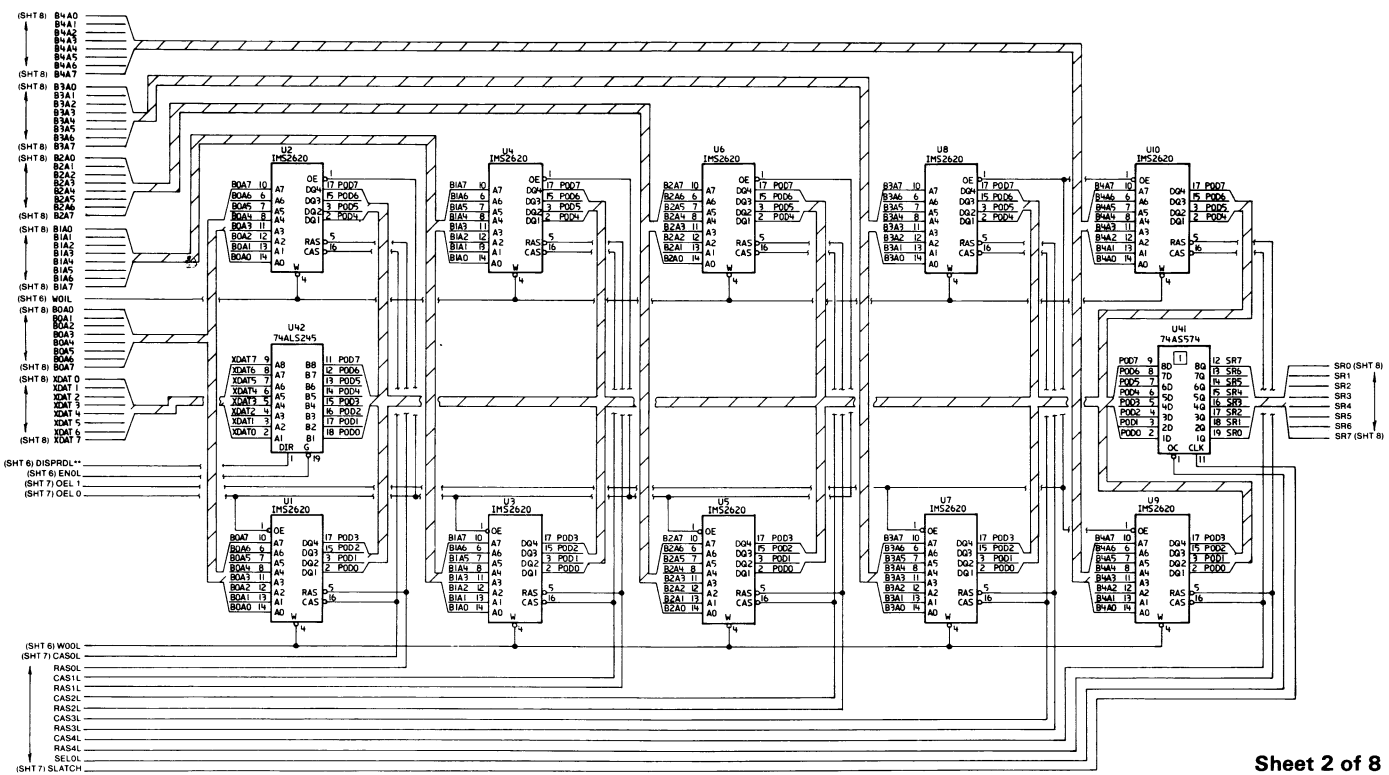


(SHT 2) CURSL
(SHT 3) PAT6
(SHT 4) CHARL
(SHT 3) PAT 7
(SHT 4) INTENH
(SHT 2) SO AL
(SHT 1) EM DOT CLK
U15 74F11
U33 74F08
U39 74F378
D5
D4
D3
D2
D1
D0
Q0
Q1
Q2
Q3
Q4
E
CP
CURSORL (SHT 4)
ENBCK GND H (SHT 4)
ENALPHAL (SHT 4)
AT7 (SHT 4)
U36 74F04
U36 74F04
U33 74F08
(SHT 1) CPU DAT 7
CPU DAT 6
CPU DAT 5
CPU DAT 4
CPU DAT 3
CPU DAT 2
CPU DAT 1
(SHT 1) CPU DAT 0
CPU DAT7
CPU DAT6
CPU DAT5
CPU DAT4
CPU DAT3
CPU DAT2
CPU DAT1
CPU DAT0
U40 74LS354
G3
W
S2
S1
S0
CLK
SC
G2
G1
(SHT 4) RA3
(SHT 4) RA2
(SHT 4) RA1
(SHT 1) CURSORWTL
U41 74F74
SD1
D1
CP1
CD1
Q1
(SHT 1) AECASH
(SHT 1) 13B
RP3
ECASL (SHT 2)
DECASL (SHT 3)
ECASH (SHT 2)
RP1 4.7K
RPUL 1 (SHT 2)
RPUL 2 (SHT 2,4)
RPUL 3 (SHT 4)
RPUL 6 (SHT 3)
(SHT 1)

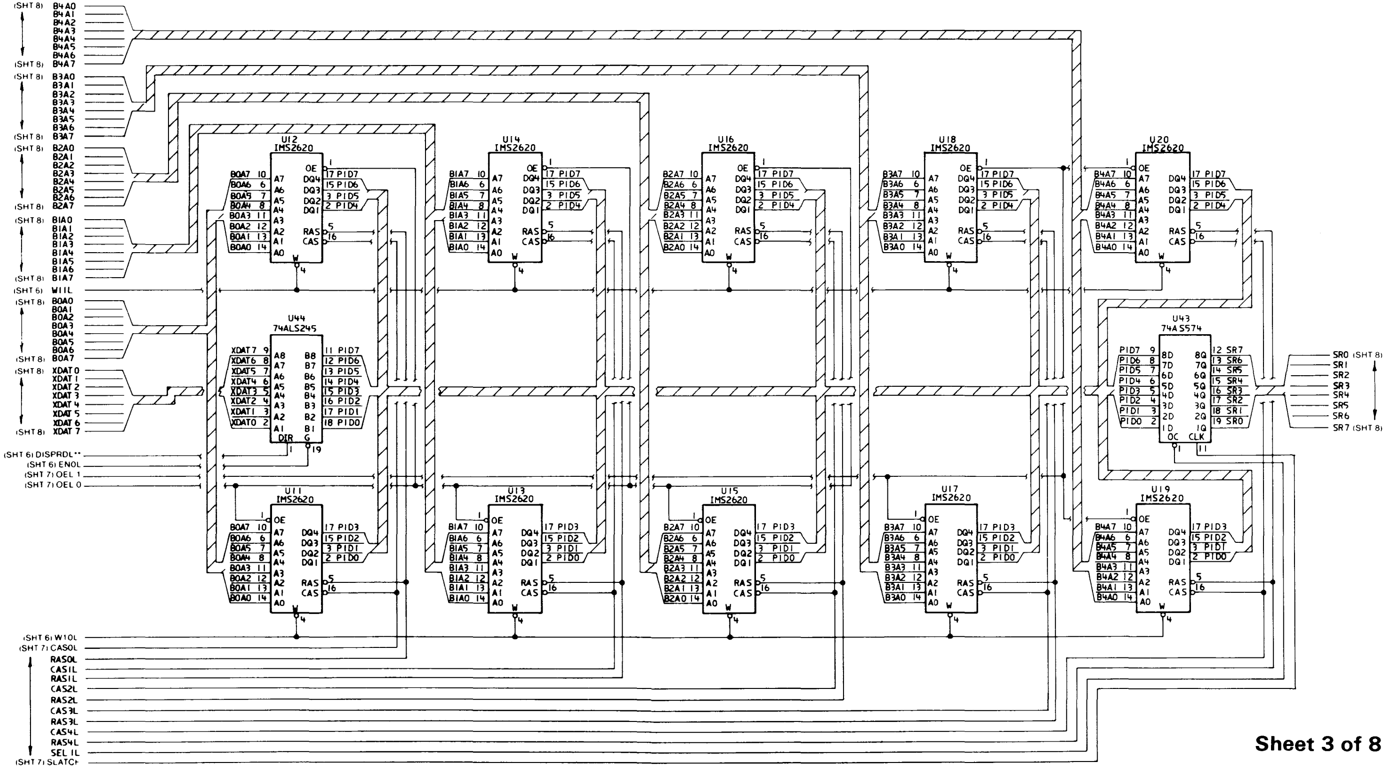
Sheet 5 of 5


PA1
PIN
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28
A29
A30
A31
PB1
PIN
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
B12
B13
B14
B15
B16
B17
B18
B19
B20
B21
B22
B23
B24
B25
B26
B27
B28
B29
B30
B31
GND
+5V
+12V
J1
PIN
1 INITL (8)
3 INITH (9)
(8)DOTCLK 5
7 MODE WTL (8)
9 SPARE
11 SPARE
13 DRAM4(9)
15 DRAM3(9)
17 DRAM2(9)
19 EM0H(NC)
21 EM1H(NC)
23 EM2H(NC)
25 CPUDAT0(10)
27 CPUDATI(10)
29 CPUDAT2(10)
31 CPUDAT3(10)
33 CPUAD0(8)
35 CPUAD1(8)
37 CPUAD2(8)
39 CPUAD3(8)
41 CPUAD4(8)
43 CPUAD5(NC)
45 CPUAD6(NC)
47 CPUAD7(NC)
49 CPUAD8(NC)
J1
PIN
2
4
6 RESERVED
8 EVERESTL(10)
10 DISPRDL(8,10)
12 DISPRWRL(8,10)
(9)DISPRDYL 14
16 XSYN(8)
18 ERDL(NC)
20 ERWL(NC)
(8) 0 3B 22
(8) 0 1B 24
26 CPUDAT4(10)
28 CPUDAT5(10)
30 CPUDAT6(10)
32 CPUDAT7(10)
34 STARTADOWTL(NC)
36 CPUAD9(NC)
38 CPUAD10(NC)
40 CPUAD11(NC)
42 CPUAD12(NC)
44 CPUAD13(NC)
46 RESERVED
48
50
J2
PIN
1
3
5 RESERVED
7 AD0(10)
9 AD1(10)
11 AD2(10)
13 AD3(10)
15 AD4(10)
17 AD5(10)
19 AD6(10)
21 AD7(10)
23 FUNCTION WTL (NC)
25 SETLPENL(NC)
27 CLRLPENL(NC)
29 STATSELL(NC)
31 DISACCH(B)
33 N1(A)
35 N6(8)
37 A(NC)
39 B(NC)
41 C(NC)
43 D(NC)
45 RESERVED
47
49
J2
PIN
2 SPARE
4 ERASH(NC)
6 ECASH(NC)
8 EDISPH/CPU(NC)
LPENSW 10
LPENINPUT 12
(9)ADVWHSYNCSCAN 14
16 SPARE
18 LPST(NC)
20 HUMPL(NC)
22 LINCNTEN(NC)
(8)DISPL/CPU 24
26 MEN(8,9)
(9)CAS4TL 28
30 SPARE
32 SPARE
34 SPARE
(10)PIXBUS0 36
(10)PIXBUS1 38
(10)PIXBUS2 40
(10)PIXBUS3 42
(10)PIXBUS4 44
(10)PIXBUS5 46
(10)PIXBUS6 48
(10)PIXBUS7 50

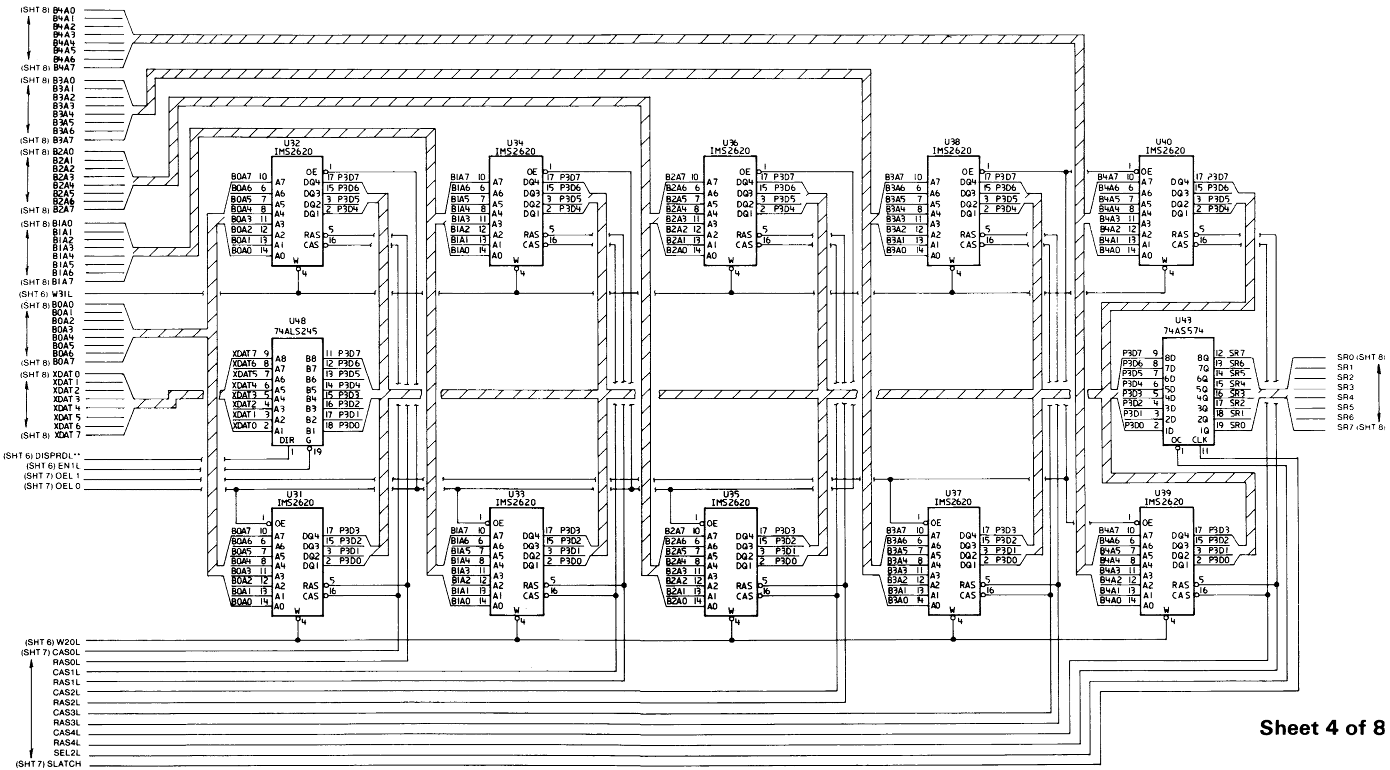
Sheet 1 of 8


(SHT 8) B4A0
B4A1
B4A2
B4A3
B4A4
B4A5
B4A6
(SHT 8) B4A7
(SHT 8) B3A0
B3A1
B3A2
B3A3
B3A4
B3A5
B3A6
(SHT 8) B3A7
(SHT 8) B2A0
B2A1
B2A2
B2A3
B2A4
B2A5
B2A6
(SHT 8) B2A7
(SHT 8) B1A0
B1A1
B1A2
B1A3
B1A4
B1A5
B1A6
(SHT 8) B1A7
(SHT 6) W01L
(SHT 8) B0A0
B0A1
B0A2
B0A3
B0A4
B0A5
B0A6
(SHT 8) B0A7
(SHT 8) XDAT 0
XDAT 1
XDAT 2
XDAT 3
XDAT 4
XDAT 5
XDAT 6
(SHT 8) XDAT 7
(SHT 6) DISPRDL**
(SHT 6) ENOL
(SHT 7) OEL 1
(SHT 7) OEL 0
(SHT 6) W00L
(SHT 7) CAS0L
RAS0L
CAS1L
RAS1L
CAS2L
RAS2L
CAS3L
RAS3L
CAS4L
RAS4L
SEL0L
(SHT 7) SLATCH
U2
IMS2620
U4
IMS2620
U6
IMS2620
U8
IMS2620
U10
IMS2620
U1
IMS2620
U3
IMS2620
U5
IMS2620
U7
IMS2620
U9
IMS2620
U42
74ALS245
U41
74AS574
OE
A7
A6
A5
A4
A3
A2
A1
A0
DQ4
DQ3
DQ2
DQ1
RAS
CAS
W
POD7
POD6
POD5
POD4
POD3
POD2
POD1
POD0
A8
B8
B7
B6
B5
B4
B3
B2
B1
DIR
G
8D
7D
6D
5D
4D
3D
2D
1D
8Q
7Q
6Q
5Q
4Q
3Q
2Q
1Q
OC
CLK
SR7
SR6
SR5
SR4
SR3
SR2
SR1
SR0
SR0 (SHT 8)
SR1
SR2
SR3
SR4
SR5
SR6
SR7 (SHT 8)

Sheet 2 of 8

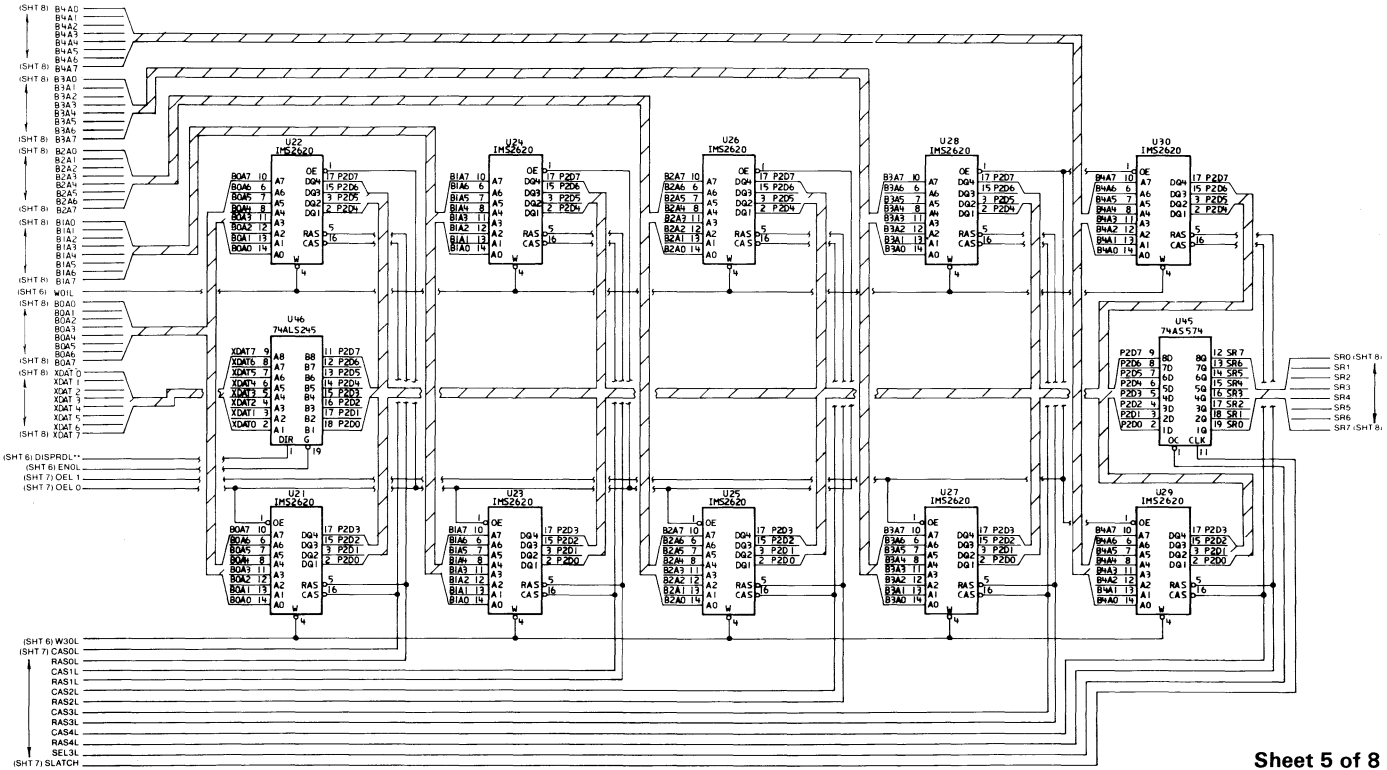

(SHT 8) B4A0
B4A1
B4A2
B4A3
B4A4
B4A5
B4A6
(SHT 8) B4A7
(SHT 8) B3A0
B3A1
B3A2
B3A3
B3A4
B3A5
B3A6
(SHT 8) B3A7
(SHT 8) B2A0
B2A1
B2A2
B2A3
B2A4
B2A5
B2A6
(SHT 8) B2A7
(SHT 8) B1A0
B1A1
B1A2
B1A3
B1A4
B1A5
B1A6
(SHT 8) B1A7
(SHT 6) W11L
(SHT 8) B0A0
B0A1
B0A2
B0A3
B0A4
B0A5
B0A6
(SHT 8) B0A7
(SHT 8) XDAT 0
XDAT 1
XDAT 2
XDAT 3
XDAT 4
XDAT 5
XDAT 6
(SHT 8) XDAT 7
(SHT 6) DISPRDL
(SHT 6) ENOL
(SHT 7) OEL 1
(SHT 7) OEL 0
(SHT 6) W10L
(SHT 7) CASOL
RASOL
CAS1L
RAS1L
CAS2L
RAS2L
CAS3L
RAS3L
CAS4L
RAS4L
SEL 1L
(SHT 7) SLATCH
U12 IMS2620
U14 IMS2620
U16 IMS2620
U18 IMS2620
U20 IMS2620
U44 74ALS245
U43 74AS574
U11 IMS2620
U13 IMS2620
U15 IMS2620
U17 IMS2620
U19 IMS2620
OE
A7
A6
A5
A4
A3
A2
A1
A0
DQ4
DQ3
DQ2
DQ1
RAS
CAS
W
17 PID7
15 PID6
3 PID5
2 PID4
17 PID3
15 PID2
3 PID1
2 PID0
A8
B8
B7
B6
B5
B4
B3
B2
B1
DIR
G
11 PID7
12 PID6
13 PID5
14 PID4
15 PID3
16 PID2
17 PID1
18 PID0
8D
7D
6D
5D
4D
3D
2D
1D
8Q
7Q
6Q
5Q
4Q
3Q
2Q
1Q
OC
CLK
12 SR7
13 SR6
14 SR5
15 SR4
16 SR3
17 SR2
18 SR1
19 SR0
SR0 (SHT 8)
SR1
SR2
SR3
SR4
SR5
SR6
SR7 (SHT 8)

Sheet 3 of 8

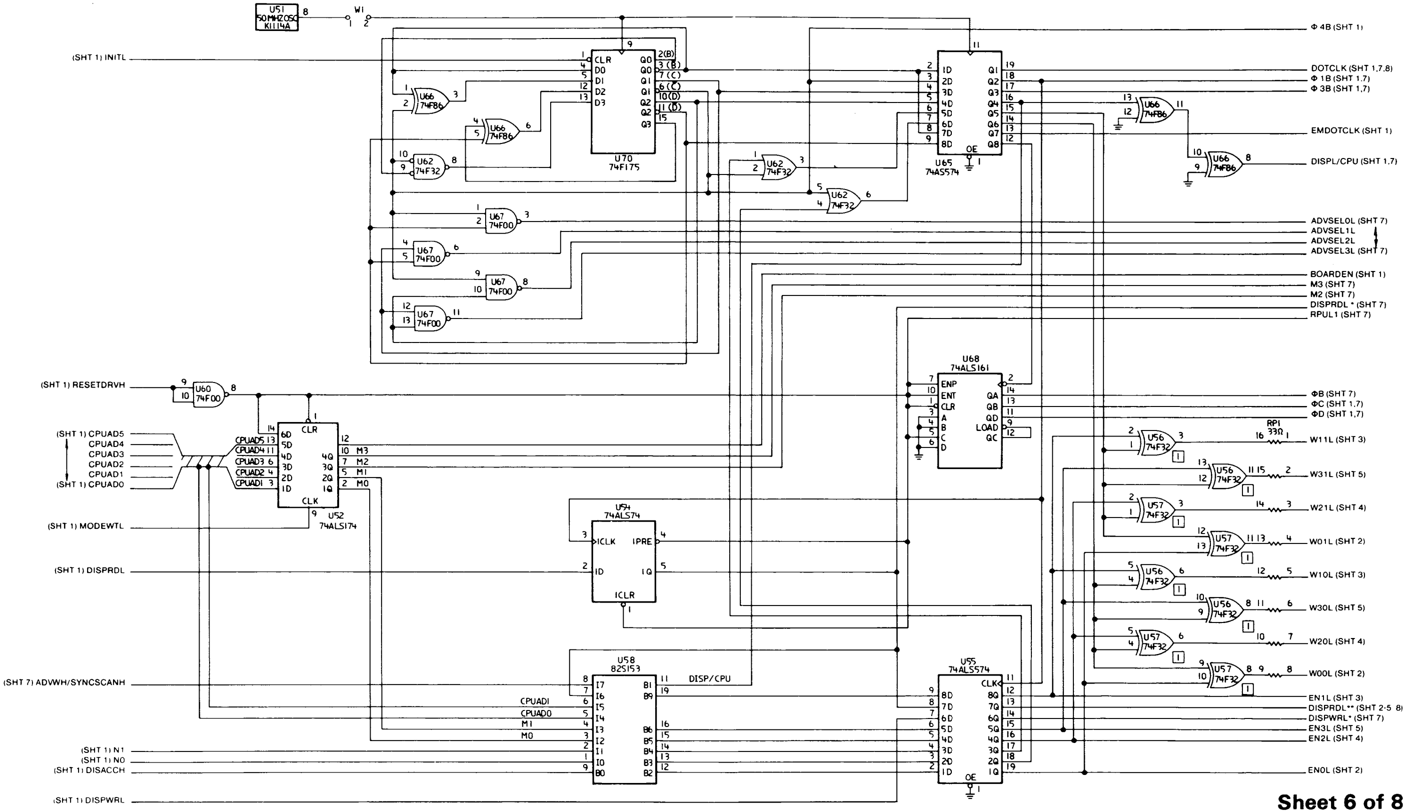

(SHT 8) B4A0
B4A1
B4A2
B4A3
B4A4
B4A5
B4A6
(SHT 8) B4A7
(SHT 8) B3A0
B3A1
B3A2
B3A3
B3A4
B3A5
B3A6
(SHT 8) B3A7
(SHT 8) B2A0
B2A1
B2A2
B2A3
B2A4
B2A5
B2A6
(SHT 8) B2A7
(SHT 8) B1A0
B1A1
B1A2
B1A3
B1A4
B1A5
B1A6
(SHT 8) B1A7
(SHT 6) W31L
(SHT 8) B0A0
B0A1
B0A2
B0A3
B0A4
B0A5
B0A6
(SHT 8) B0A7
(SHT 8) XDAT 0
XDAT 1
XDAT 2
XDAT 3
XDAT 4
XDAT 5
XDAT 6
(SHT 8) XDAT 7
(SHT 6) DISPRDL**
(SHT 6) EN1L
(SHT 7) OEL 1
(SHT 7) OEL 0
U32 IMS2620
U34 IMS2620
U36 IMS2620
U38 IMS2620
U40 IMS2620
U48 74ALS245
U43 74AS574
U31 IMS2620
U33 IMS2620
U35 IMS2620
U37 IMS2620
U39 IMS2620
OE
DQ4
DQ3
DQ2
DQ1
A7
A6
A5
A4
A3
A2
A1
A0
RAS
CAS
W
P3D7
P3D6
P3D5
P3D4
P3D3
P3D2
P3D1
P3D0
A8 A7 A6 A5 A4 A3 A2 A1
B8 B7 B6 B5 B4 B3 B2 B1
DIR
G
8D 7D 6D 5D 4D 3D 2D 1D
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
OC
CLK
SR7
SR6
SR5
SR4
SR3
SR2
SR1
SR0
SR0 (SHT 8)
SR1
SR2
SR3
SR4
SR5
SR6
SR7 (SHT 8)
(SHT 6) W20L
(SHT 7) CAS0L
RAS0L
CAS1L
RAS1L
CAS2L
RAS2L
CAS3L
RAS3L
CAS4L
RAS4L
SEL2L
(SHT 7) SLATCH

Sheet 4 of 8

August 15, 1984

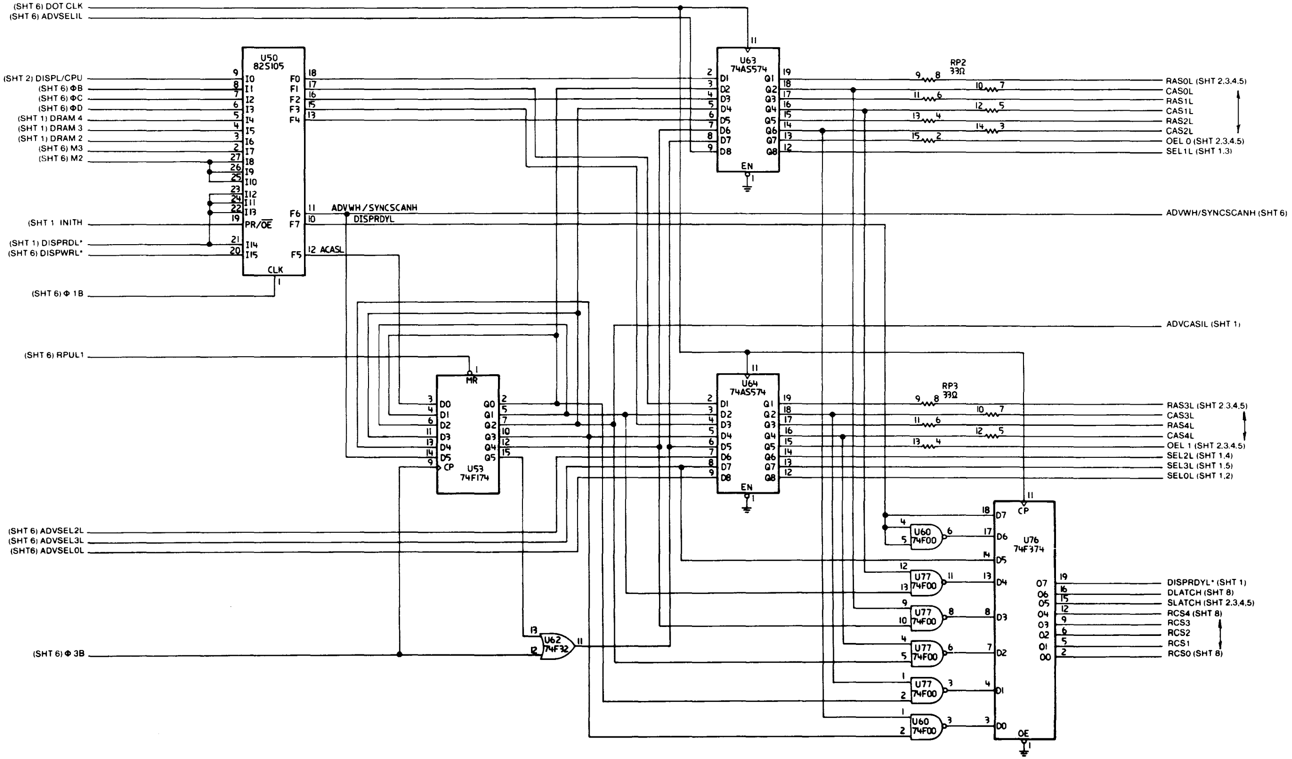


Sheet 5 of 8

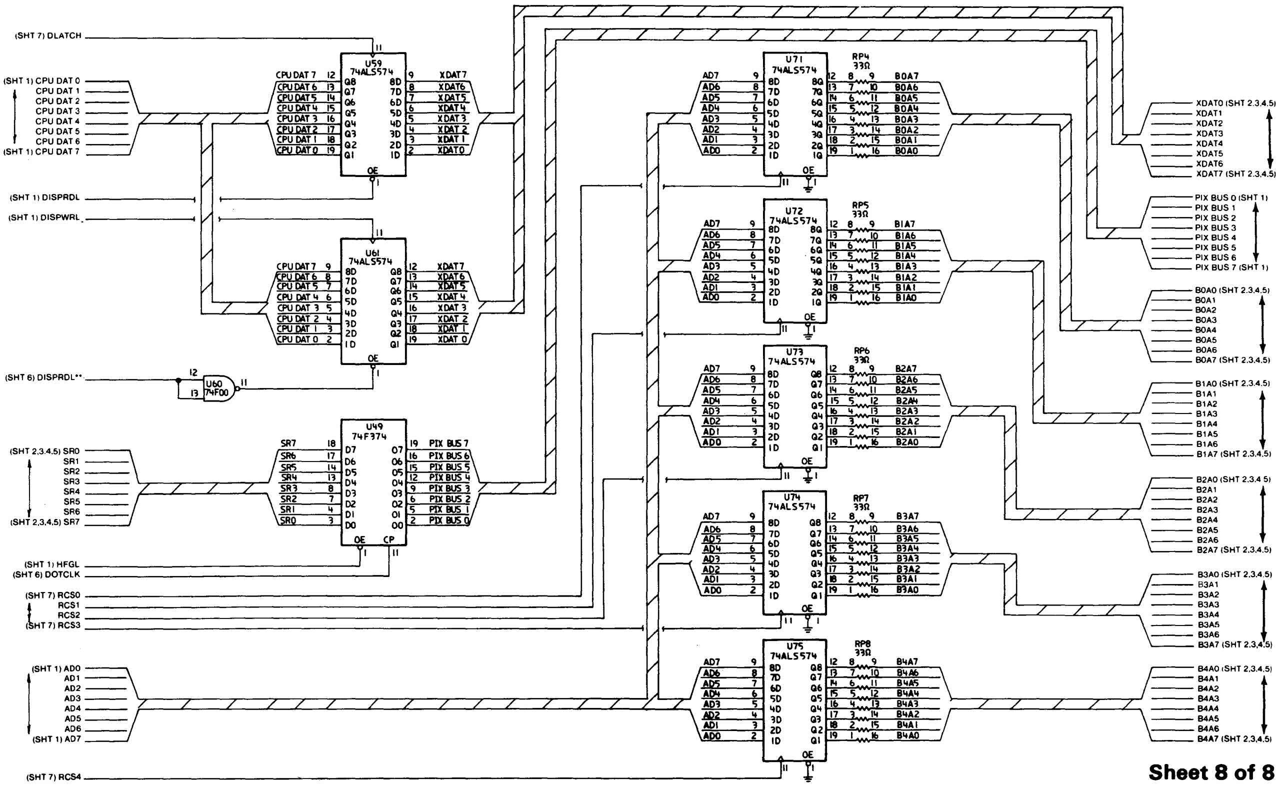

U51
50MHZOSC
K1114A
W1
(SHT 1) INITL
CLR
D0
D1
D2
D3
Q0
Q1
Q2
Q3
2(B)
3(B)
7(C)
6(C)
10(D)
11(D)
U70
74F175
U66
74F86
U62
74F32
U67
74F00
1D
2D
3D
4D
5D
6D
7D
8D
OE
U65
74AS574
Φ 4B (SHT 1)
DOTCLK (SHT 1,7,8)
Φ 1B (SHT 1,7)
Φ 3B (SHT 1,7)
EMDOTCLK (SHT 1)
DISPL/CPU (SHT 1,7)
ADVSEL0L (SHT 7)
ADVSEL1L
ADVSEL2L
ADVSEL3L (SHT 7)
BOARDEN (SHT 1)
M3 (SHT 7)
M2 (SHT 7)
DISPRDL* (SHT 7)
RPUL1 (SHT 7)
(SHT 1) RESETDRVH
U60
74F00
U68
74ALS161
ENP
ENT
CLR
A
B
C
D
QA
QB
QC
QD
LOAD
ΦB (SHT 7)
ΦC (SHT 1,7)
ΦD (SHT 1,7)
RP1
33Ω
U56
74F32
U57
74F32
W11L (SHT 3)
W31L (SHT 5)
W21L (SHT 4)
W01L (SHT 2)
W10L (SHT 3)
W30L (SHT 5)
W20L (SHT 4)
W00L (SHT 2)
(SHT 1) CPUAD5
CPUAD4
CPUAD3
CPUAD2
CPUAD1
(SHT 1) CPUAD0
CPUAD5
CPUAD4
CPUAD3
CPUAD2
CPUAD1
6D
5D
4D
3D
2D
1D
4Q
3Q
2Q
1Q
M3
M2
M1
M0
CLK
U52
74ALS174
(SHT 1) MODEWTL
U54
74ALS74
1CLK
1PRE
1D
1Q
1CLR
(SHT 1) DISPRDL
U58
82S153
(SHT 7) ADVWH/SYNCSCANH
I7
I6
I5
I4
I3
I2
I1
I0
B0
B1
B9
B6
B5
B4
B3
B2
DISP/CPU
CPUAD1
CPUAD0
M1
M0
(SHT 1) N1
(SHT 1) N0
(SHT 1) DISACCH
(SHT 1) DISPWRL
U55
74ALS574
8D
7D
6D
5D
4D
3D
2D
1D
8Q
7Q
6Q
5Q
4Q
3Q
2Q
1Q
EN1L (SHT 3)
DISPRDL** (SHT 2-5 8)
DISPWRL* (SHT 7)
EN3L (SHT 5)
EN2L (SHT 4)
EN0L (SHT 2)

Sheet 6 of 8

]


(SHT 6) DOT CLK
(SHT 6) ADVSELIL
(SHT 2) DISPL/CPU
(SHT 6) ΦB
(SHT 6) ΦC
(SHT 6) ΦD
(SHT 1) DRAM 4
(SHT 1) DRAM 3
(SHT 1) DRAM 2
(SHT 6) M3
(SHT 6) M2
(SHT 1 INITH
(SHT 1) DISPRDL*
(SHT 6) DISPWRL*
(SHT 6) Φ 1B
(SHT 6) RPUL1
(SHT 6) ADVSEL2L
(SHT 6) ADVSEL3L
(SHT6) ADVSEL0L
(SHT 6) Φ 3B
U50
82S105
CLK
PR/OE
ADVWH/SYNCSCANH
DISPRDYL
ACASL
U53
74F174
MR
CP
U62
74F32
U63
74AS574
U64
74AS574
EN
RP2
33Ω
RP3
33Ω
U60
74F00
U77
74F00
U76
74F374
OE
RAS0L (SHT 2,3,4,5)
CAS0L
RAS1L
CAS1L
RAS2L
CAS2L
OEL 0 (SHT 2,3,4,5)
SEL1L (SHT 1,3)
ADVWH/SYNCSCANH (SHT 6)
ADVCASIL (SHT 1)
RAS3L (SHT 2,3,4,5)
CAS3L
RAS4L
CAS4L
OEL 1 (SHT 2,3,4,5)
SEL2L (SHT 1,4)
SEL3L (SHT 1,5)
SEL0L (SHT 1,2)
DISPRDYL* (SHT 1)
DLATCH (SHT 8)
SLATCH (SHT 2,3,4,5)
RCS4 (SHT 8)
RCS3
RCS2
RCS1
RCS0 (SHT 8)

Sheet 7 of 8


(SHT 7) DLATCH
(SHT 1) CPU DAT 0
CPU DAT 1
CPU DAT 2
CPU DAT 3
CPU DAT 4
CPU DAT 5
CPU DAT 6
(SHT 1) CPU DAT 7
(SHT 1) DISPRDL
(SHT 1) DISPWRL
(SHT 6) DISPRDL**
(SHT 2,3,4,5) SR0
SR1
SR2
SR3
SR4
SR5
SR6
(SHT 2,3,4,5) SR7
(SHT 1) HFGL
(SHT 6) DOTCLK
(SHT 7) RCS0
RCS1
RCS2
(SHT 7) RCS3
(SHT 1) AD0
AD1
AD2
AD3
AD4
AD5
AD6
(SHT 1) AD7
(SHT 7) RCS4
U59
74ALS574
U61
74ALS574
U60
74F00
U49
74F374
U71
74ALS574
U72
74ALS574
U73
74ALS574
U74
74ALS574
U75
74ALS574
RP4 33Ω
RP5 33Ω
RP6 33Ω
RP7 33Ω
RP8 33Ω
OE
CP
XDAT0 … XDAT7
PIX BUS 0 … PIX BUS 7
B0A0 … B0A7
B1A0 … B1A7
B2A0 … B2A7
B3A0 … B3A7
B4A0 … B4A7
XDAT0 (SHT 2,3,4,5)
XDAT1
XDAT2
XDAT3
XDAT4
XDAT5
XDAT6
XDAT7 (SHT 2,3,4,5)
PIX BUS 0 (SHT 1)
PIX BUS 1
PIX BUS 2
PIX BUS 3
PIX BUS 4
PIX BUS 5
PIX BUS 6
PIX BUS 7 (SHT 1)
B0A0 (SHT 2,3,4,5)
B0A1
B0A2
B0A3
B0A4
B0A5
B0A6
B0A7 (SHT 2,3,4,5)
B1A0 (SHT 2,3,4,5)
B1A1
B1A2
B1A3
B1A4
B1A5
B1A6
B1A7 (SHT 2,3,4,5)
B2A0 (SHT 2,3,4 5)
B2A1
B2A2
B2A3
B2A4
B2A5
B2A6
B2A7 (SHT 2,3,4,5)
B3A0 (SHT 2,3,4,5)
B3A1
B3A2
B3A3
B3A4
B3A5
B3A6
B3A7 (SHT 2,3,4,5)
B4A0 (SHT 2,3,4,5)
B4A1
B4A2
B4A3
B4A4
B4A5
B4A6
B4A7 (SHT 2,3,4,5)

Sheet 8 of 8

Glossary

algorithm. A finite set of well-defined rules for the solution of a problem in a finite number of steps.

alphanumeric (A/N). Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks.

American National Standard Code for Information Exchange (ASCII). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check) used for information exchange between data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

A/N. Alphanumeric

ASCII. American National Standard Code for Information Exchange.

Cartesian coordinates. A system of coordinates for locating a point on a plane by its distance from each of two intersecting lines, or in space by its distance from each of three mutually perpendicular planes.

cathode ray tube (CRT). A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

cathode ray tube display (CRT display). (1) A CRT used for displaying data. For example, the electron beam can be controlled to form alphanumeric data by use of a dot matrix. (2) Synonymous with monitor.

clipping. In computer graphics, removing parts of a display image that lie outside a window.

color cone. An arrangement of the visible colors on the surface of a double-ended cone where lightness varies along the axis of the cone, and hue varies around the circumference. Lightness includes both the intensity and saturation of color.

complement. A number that can be derived from a specified number by subtracting it from a second specified number.

coordinate space. In computer graphics, a system of Cartesian coordinates in which an object is defined.

cursor. (1) In computer graphics, a movable marker that is used to indicate a position on a display. (2) A displayed symbol that acts as a marker to help the user locate a point in text, in a system command, or in storage. (3) A movable spot of light on the screen of a display device, usually indicating where the next character is to be entered, replaced, or deleted.

debounce. (1) An electronic means of overcoming the make/break bounce of switches to obtain one smooth change of signal level. (2) The elimination of undesired signal variations caused by mechanically generated signals from contacts.

display. (1) A visual presentation of data. (2) A device for visual presentation of information on any temporary character imaging device. (3) To present data visually. (4) See cathode ray tube display.

display attribute. In computer graphics, a particular property that is assigned to all or part of a display; for example, low intensity, green color, blinking status.

display element. In computer graphics, a basic graphic element that can be used to construct a display image; for example, a dot, a line segment, a character.

display group. In computer graphics, a collection of display elements that can be manipulated as a unit and that can be further combined to form larger groups.

display image. In computer graphics, a collection of display elements or display groups that are represented together at any one time in a display space.

display space. In computer graphics, that portion of a display surface available for a display image. The display space may be all or part of a display surface.

display surface. In computer graphics, that medium on which display images may appear; for example, the entire screen of a cathode ray tube.

drawing primitive. A group of commands that draw defined geometric shapes.

field-programmable-logic-sequencer (FPLS). An integrated circuit containing a programmable, read-only memory that responds to external inputs and feedback of its own outputs.

FIFO (first-in-first-out). A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

FPLS. Field-programmable-logic-sequencer.

hither plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point and which lies between these two points. Any part of an object between the hither plane and the view point is not seen. See also yon plane.

intensity. In computer graphics, the amount of light emitted at a display point.

interleave. To arrange parts of one sequence of things or events so that they alternate with parts of one or more other sequences of the same nature and so that each sequence retains its identity.

least-significant digit. The rightmost digit.

look-up table (LUT). (1) A technique for mapping one set of values into a larger set of values. (2) In computer graphics, a table that assigns a color value (red, green, blue intensities) to a color index.

luminance. The luminous intensity per unit projected area of a given surface viewed from a given direction.

LUT. Look-up table.

mask. (1) A pattern of characters that is used to control the retention or elimination of portions of another pattern of characters. (2) To use a pattern of characters to control the retention or elimination of portions of another pattern of characters.

matrix. (1) A rectangular array of elements, arranged in rows and columns, that may be manipulated according to the rules of matrix algebra. (2) In computers, a logic network in the form of an array of input leads and output leads with logic elements connected at some of their intersections.

mode. (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumeric mode. (2) The most frequent value in the statistical sense.

modeling transformation. Operations on the coordinates of an object (usually matrix multiplications) which cause the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). See also viewing transformation.

modulo-N check. A check in which an operand is divided by a number N (the modulus) to generate a remainder (check digit) that is retained with the operand. For example, in a modulo-7 check, the remainder will be 0, 1, 2, 3, 4, 5, or 6. The operand is later checked by again dividing it by the modulus; if the remainder is not equal to the check digit, an error is indicated.

modulus. In a modulo-N check, the number by which the operand is divided.

monitor. Synonym for cathode ray tube display (CRT display).

most-significant digit. The leftmost (non-zero) digit.

nanosecond (ns). 0.000 000 001 second.

ns. Nanosecond; 0.000 000 001 second.

PEL. Picture element.

picture element (PEL). The smallest displayable unit on a display.

raster. A predetermined pattern of lines that provides uniform coverage of a display space.

saturation. In computer graphics, the purity of a particular hue. A color is said to be saturated when at least one primary color (red, green, or blue) is completely absent.

scaling. In computer graphics, enlarging or reducing all or part of a display image by multiplying the coordinates of the image by a constant value.

vector. In computer graphics, a directed line segment.

view point. In computer graphics, the origin from which angles and scales are used to map virtual space into display space.

viewing reference point. In computer graphics, a point in the modeling coordinate space that is a defined distance from the view point.

viewing transformation. Operations on the coordinates of an object (usually matrix multiplications) which cause the view of the object to be rotated about any axis, translated (moved without rotating), and/or scaled (changed in size along any or all dimensions). Viewing transformations differ from modeling transformations in that perspective is taken into account. See also modeling transformation.

viewplane. In computer graphics, a two-dimensional coordinate system onto which images are projected and which contains the display space.

viewport. In computer graphics, a predefined part of the display space.

virtual space. In computer graphics, a space in which the coordinates of the display elements are expressed in terms of user coordinates.

window. (1) In computer graphics, a predefined part of the virtual space. (2) In computer graphics, the visible area of a viewplane mapped into a viewport.

yon plane. In computer graphics, a plane that is perpendicular to the line joining the viewing reference point and the view point and which lies beyond the viewing reference point. Any part of an object beyond the yon plane is not seen. See also hither plane.

Index

IBM

Personal Computer
Hardware Reference
Library

IBM Personal Computer Data Acquisition and Control Adapter Technical Reference

DATA ACQUISTION AND CONTROL ADAPTER

6138163
August 15,1984

Contents

Description

The IBM Personal Computer Data Acquisition and Control Adapter (Data Acquisition Adapter) provides both analog and digital I/O capabilities. It is installed in any full-length expansion slot, and up to four may be installed in a system.

The adapter provides:

- Four analog input channels multiplexed into an analog-to-digital converter (ADC), with 12-bit resolution

- Two analog output channels, each having its own digital-to-analog converter (DAC), with 12-bit resolution

- A 16-bit digital input port

- A 16-bit digital output port

- A 32-bit timer

- A 16-bit, externally-clocked, timer/counter

- An expansion bus.

The Data Acquisition Adapter has a 16-bit data bus and a buffered 8-bit data bus.

The adapter's 16-bit data bus provides access to:

- An analog I/O device:
  - Analog input subsystem with four multiplexed channels
  - Analog output subsystem with two DACs
- A binary I/O device:
  - 16-bit digital input port
  - 16-bit digital output port
  - Handshaking
- An expansion bus.

The buffered 8-bit data bus provides access to:

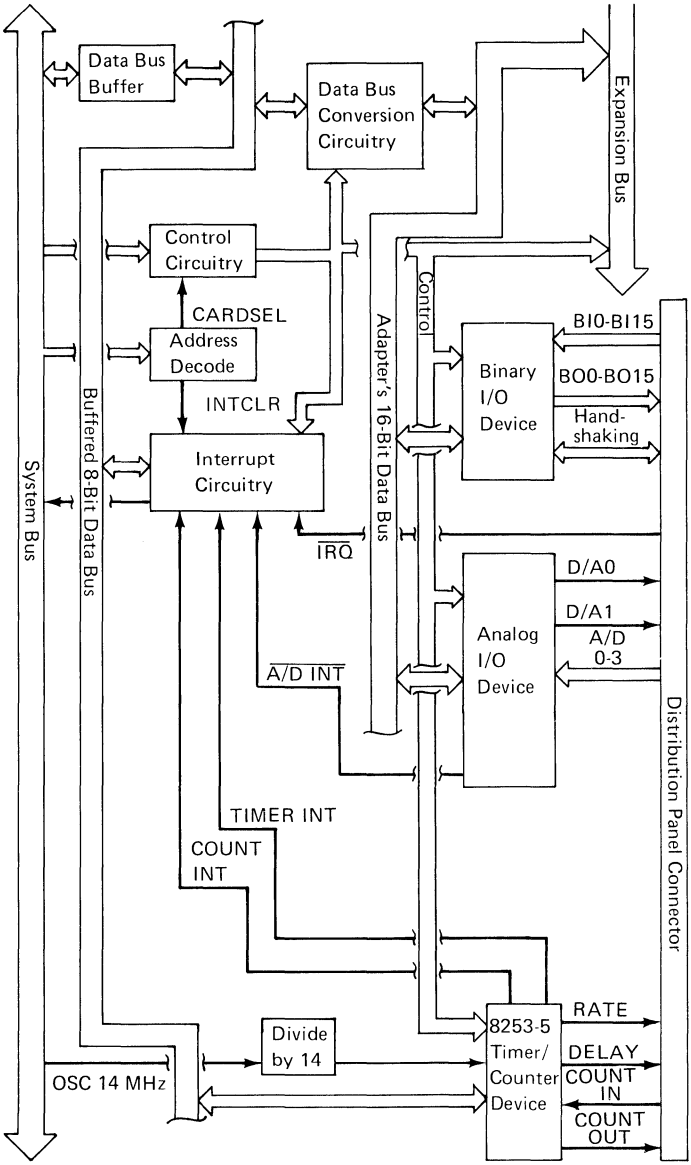
- Interrupt circuitry
- A timer/counter device:
  - 32-bit timer (Counters 0 and 1)
  - 16-bit timer/counter (Counter 2).

Low and high bytes are transferred between the adapter's buffered 8-bit data bus and 16-bit data bus.

A 60-pin, distribution-panel connector is provided for external access to the analog I/O device, the binary I/O device, and the timer/counter device.

Major Components

Following is a block diagram of the Data Acquisition Adapter.



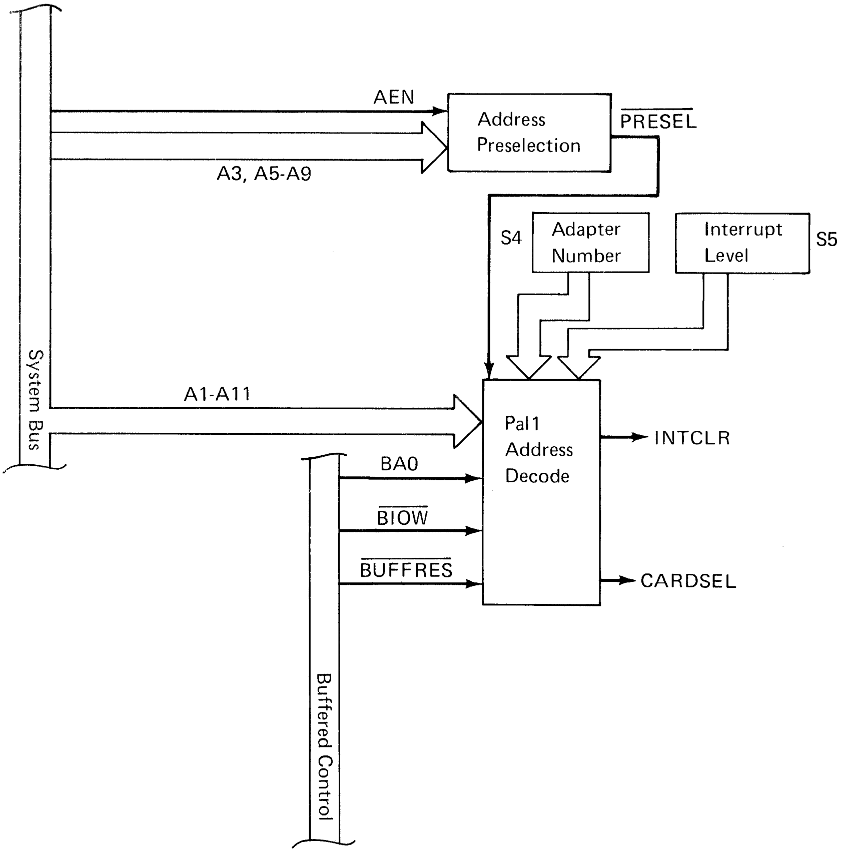
The following are descriptions of the major components shown in the figure on the previous page.

Address Decode and Control Circuitry

The following are descriptions of address decode and control circuitry.

Address Decode

Following is a block diagram of address decode.



The signals used by the address decode circuitry are:

AEN Address enable: De-gates the processor and other devices from the I/O channel to allow direct-memory access (DMA) transfers to take place. When active (high), the DMA controller has control of the address bus, data bus, read command lines (memory and I/O), and the write command lines (memory and I/O).

$\overline{\text{PRESEL}}$ Preselect: Indicates preliminary address decoding of the 'address enable' signal (AEN), and the address bits that are common to the adapter's base address and to the shared-interrupt address.

CARDSEL Card select: Indicates communication is in process between the Data Acquisition Adapter and the system. The shared-interrupt re-activation function is not included.

INTCLR Shared-interrupt reactivation control signal.

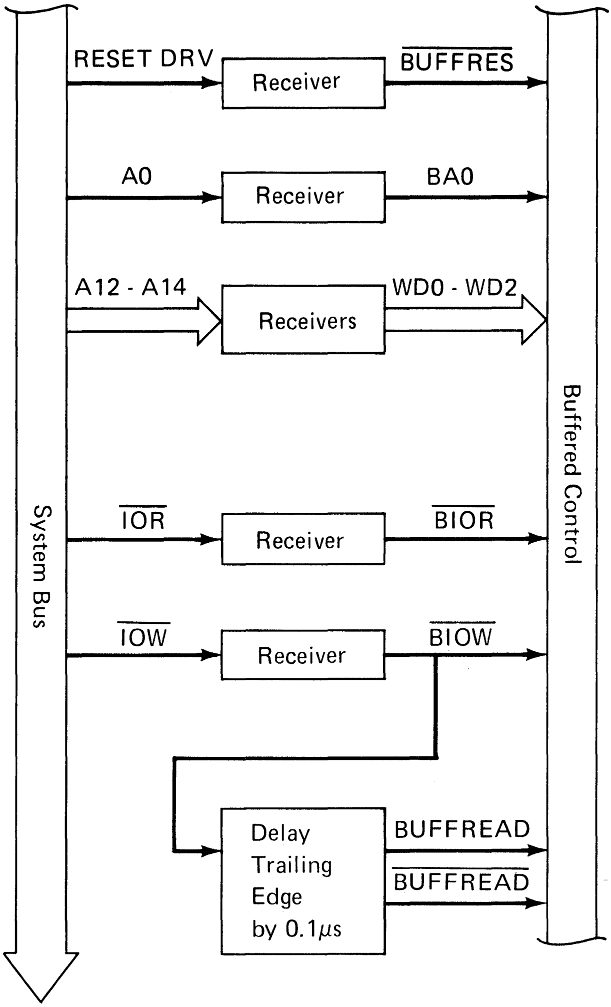
The address preselection circuitry decodes the six address lines, which are common in the adapter address and the shared-interrupt address. AEN is used to prevent false decodes during DMA cycles. Because the adapter has a base address of hex 2E2 through 2E3, and the shared-interrupt address is hex 02Fx (where x is shared-interrupt level 3, 4, 5, 6, or 7), the common address bits are: A9, A7, A6, and A5 equal to 1, and A8 and A3 equal to 0. The resulting signal ($\overline{\text{PRESEL}}$) indicates that either an adapter access or a shared-interrupt access may be occurring.

The address decode circuitry uses the signals $\overline{\text{PRESEL}}$, A10, A11, A4, and the signals from the switches S4-1 and S4-2 to decode an adapter's base address. The control decode circuitry uses the resulting signal (CARDSEL) as a master enable and then generates the individual control signals.

When the address decoded is hex 2Fx (where x is the shared interrupt level), INTCLR is generated. INTCLR reactivates the adapter's interrupt circuitry. The address decode circuitry also generates an INTCLR signal at power-on-reset time. Power-on-reset occurs when the 'buffer reset' signal ($\overline{\text{BUFFRES}}$), which is created by the system bus signal (RESET DRV), goes low.

System Bus Address and Control Signals

The following is a block diagram of the address and control signals from the system bus.



The address and control signals from the system bus are as follows:

RESET DRV Resets system logic upon power-on or during a low line-voltage outage. RESET DRV is synchronized to the falling edge of 'clock' and is active high.

$\overline{\textbf{BUFFRES}}$ Buffer reset: Inverse of RESET DRV. Provides power-on-reset of the adapter's control logic. $\overline{\text{BUFFRES}}$ is active low.

WD0 - WD2 Word number bits 0 through 2 (system bus address lines A12 through A14 are buffered and renamed WD0 through WD2). Selects word registers 0 through 7 when system bus address line A15 is low. Selects word registers 8 through 15 when A15 is high.

BA0 Buffered system-bus address line A0: Selects high or low bytes.

$\overline{\textbf{BIOR}}$ Buffered I/O read ($\overline{\text{IOR}}$): $\overline{\text{BIOR}}$ is active low and is used for I/O read operations on the buffered 8-bit data bus.

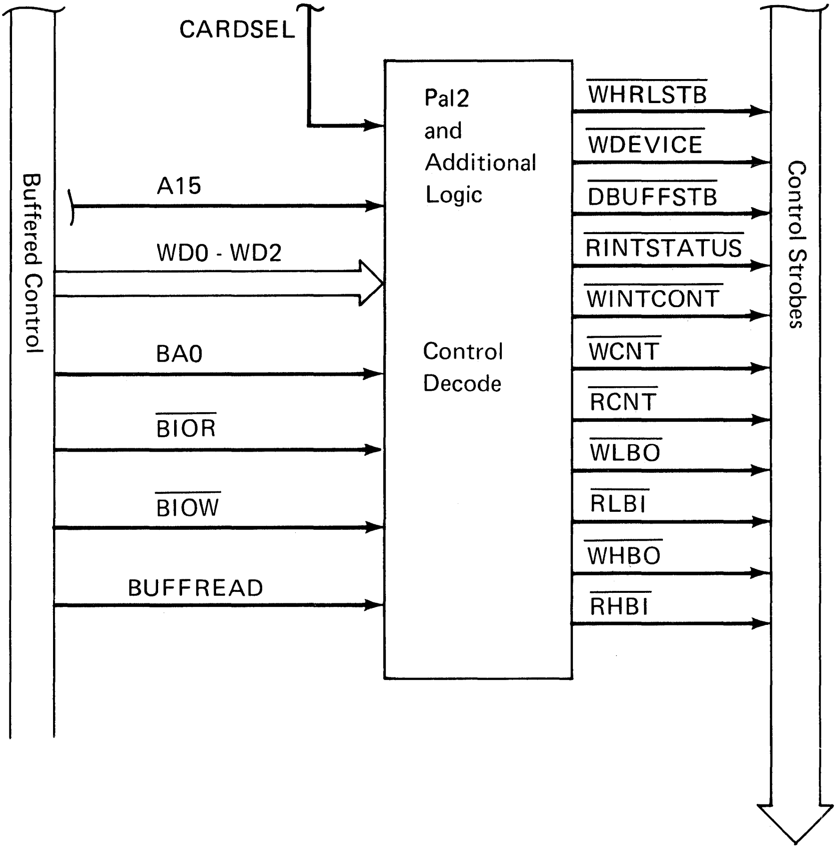
$\overline{\textbf{BIOW}}$ Buffered I/O write($\overline{\text{IOW}}$): $\overline{\text{BIOW}}$ is active low and is used for I/O write operations on the buffered 8-bit data bus.

BUFFREAD Buffer read: Indicates whether a read or a write operation on the adapter's 16-bit data bus is to be performed. When high, this signal indicates a read operation, and when it is low, a write operation is indicated.

$\overline{\text{BUFFREAD}}$ Inverse of BUFFREAD.

Control Decode

Following is a block diagram of the control decode circuitry.



The control decode circuitry inputs CARDSEL, A15, WD0 through WD2, BA0, $\overline{\mathrm{BIOR}}$, $\overline{\mathrm{BIOW}}$, and BUFFREAD generate the following control strobes:

$\overline{\textbf{WHRLSTB}}$ Write-high read-low strobe: Strobe for reading or writing the data word from the adapter's 16-bit data bus to an on-board or expansion device.

$\overline{\textbf{WDEVICE}}$ Write device: Write strobe for the on-board or expansion device number register.

$\overline{\textbf{DBUFFSTB}}$ Data buffer strobe: Enable strobe for data communications between the system bus and the Data Acquisition Adapter.

$\overline{\textbf{RINTSTATUS}}$ Read interrupt status: Read strobe for the interrupt status register.

$\overline{\textbf{WINTCONT}}$ Write interrupt control: Write strobe for the interrupt status register.

$\overline{\textbf{WCNT}}$ Write counter: Write strobe for the timer/counter.

$\overline{\textbf{RCNT}}$ Read counter: Read strobe for the timer/counter.

$\overline{\text{WLBO}}$ Write low byte out: Latch control strobe for latching the low data byte for later transmission to an on-board or expansion device.

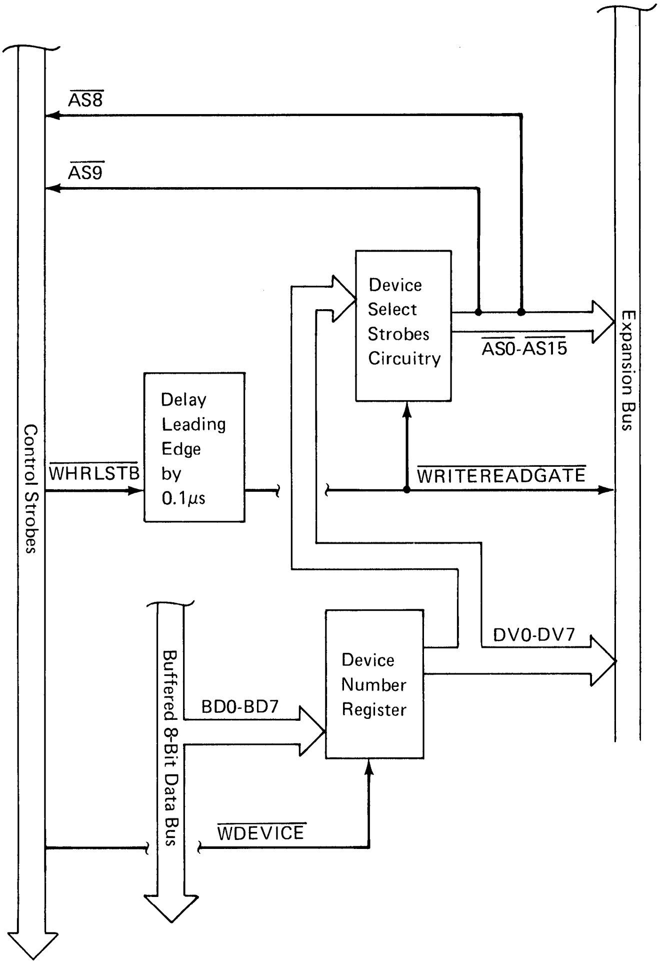
$\overline{\text{RLBI}}$ Read low byte in: Enable strobe for reading a data word from an on-board or expansion device to the adapter's 16-bit data bus. The low byte is transmitted to the system bus during this strobe. The high byte is latched during $\overline{\text{RLBI}}$ for later transmission to the system bus.

$\overline{\text{WHBO}}$ Write high byte out: Enable strobe for writing a data word from the adapter's 16-bit data bus to an on-board or expansion device. The low byte is the one previously latched in $\overline{\text{WLBO}}$. The high byte is the current data from the system bus.

$\overline{\text{RHBI}}$ Read high byte in: Enable strobe for reading the high byte (previously latched by $\overline{\text{RLBI}}$) from the adapter's 16-bit data bus to the system bus.

Device Selection

The following figure shows the device selection circuitry.



The device number register stores the device number. Device strobes are generated for on-board and expansion devices.

The following are used for device selection:

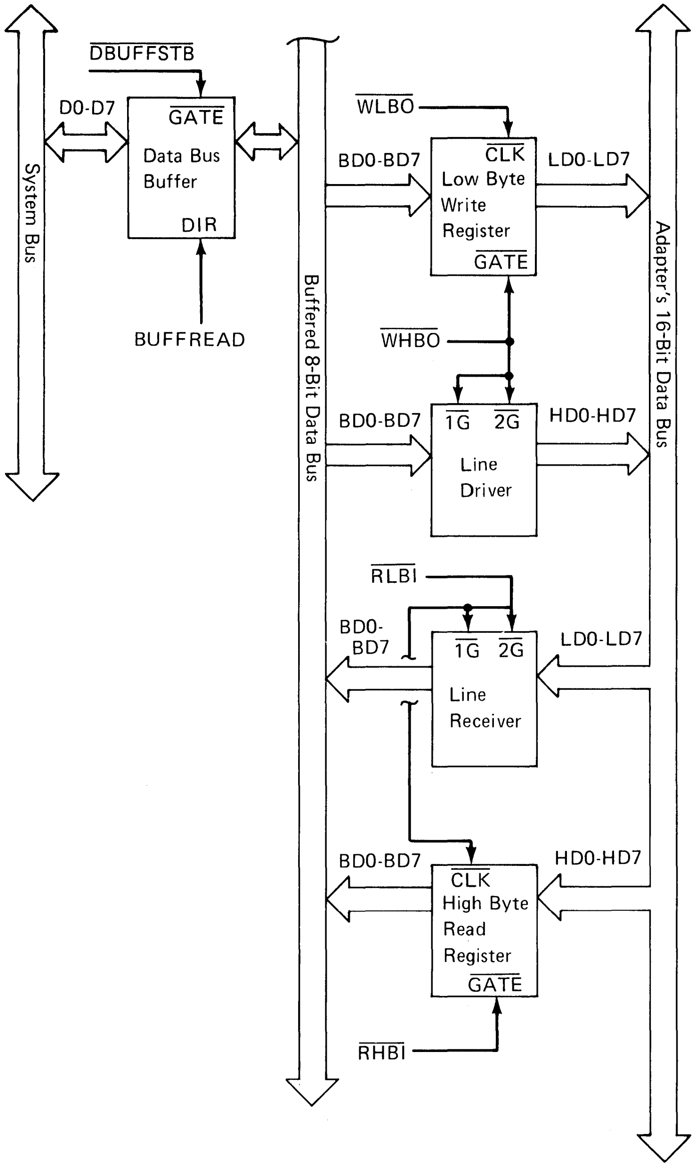
DV0 - DV7 Device number register bits 0 through 7

$\overline{\textbf{WRITEREADGATE}}$ Strobe for all devices

$\overline{\textbf{AS0}}$ through $\overline{\textbf{AS15}}$ Strobes for devices 0 through 15: $\overline{\text{AS8}}$ selects the binary I/O device, and $\overline{\text{AS9}}$ selects the analog I/O device.

Data Bus Conversion Circuitry

Following is a block diagram of the data bus conversion circuitry.



Data Bus Buffer

The system's data bus (D0 through D7) is buffered by the data bus buffer to create the adapter's buffered data bus (BD0 through BD7). The data bus buffer is activated by $\overline{\text{DBUFFSTB}}$ during all communications between the system bus and the Data Acquisition Adapter. The data direction is determined by BUFFREAD.

The buffered 8-bit data bus is used for direct 8-bit data communication with the interrupt circuitry, the timer/counter device, and the device number register.

The buffered 8-bit data bus also communicates with the low byte write register, a line driver, a line receiver, and the high byte read register to implement the conversion of the buffered 8-bit data bus into the adapter's 16-bit data bus.

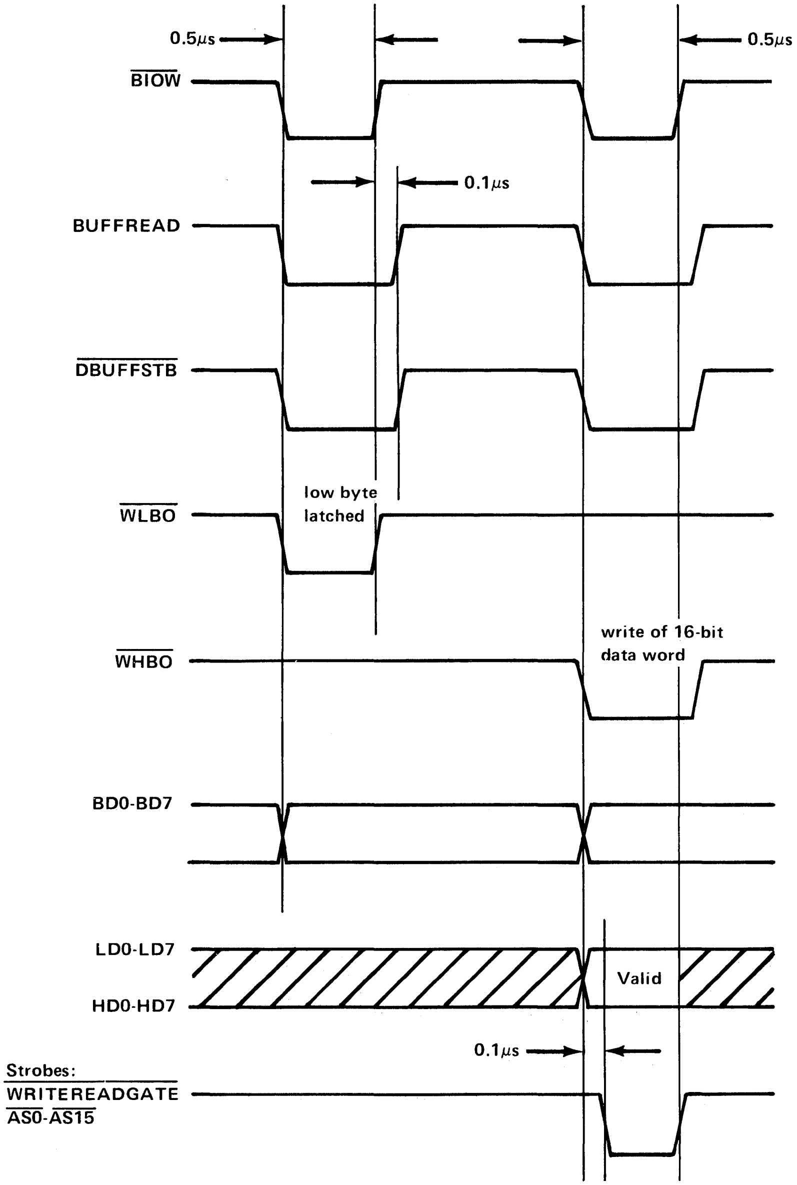
Writing Data

Data is written sequentially from the system data bus to the adapter's 16-bit data bus. The low byte is first written to an even address, then the high byte is written to an odd address. The entire 16-bit word is transmitted to an on-board or expansion device at the same time that the high byte is written.

When data is written from the system data bus to an even address (A0 is 0), the $\overline{\text{WLBO}}$ (write low byte out) strobe occurs. The low-byte write register latches the data.

When data is written from the system data bus to an odd address (A0 is 1), the $\overline{\text{WHBO}}$ (write high byte out) strobe occurs. The low-byte write register transmits the previously latched low byte to the adapter's 16-bit data bus LD lines, 0 through 7. At the same time, the line driver transmits the current data on the buffered 8-bit data bus BD lines, 0 through 7, to the adapter's 16-bit data bus HD lines, 0 through 7.

Following is a write timing diagram of the adapter's 16-bit data bus.



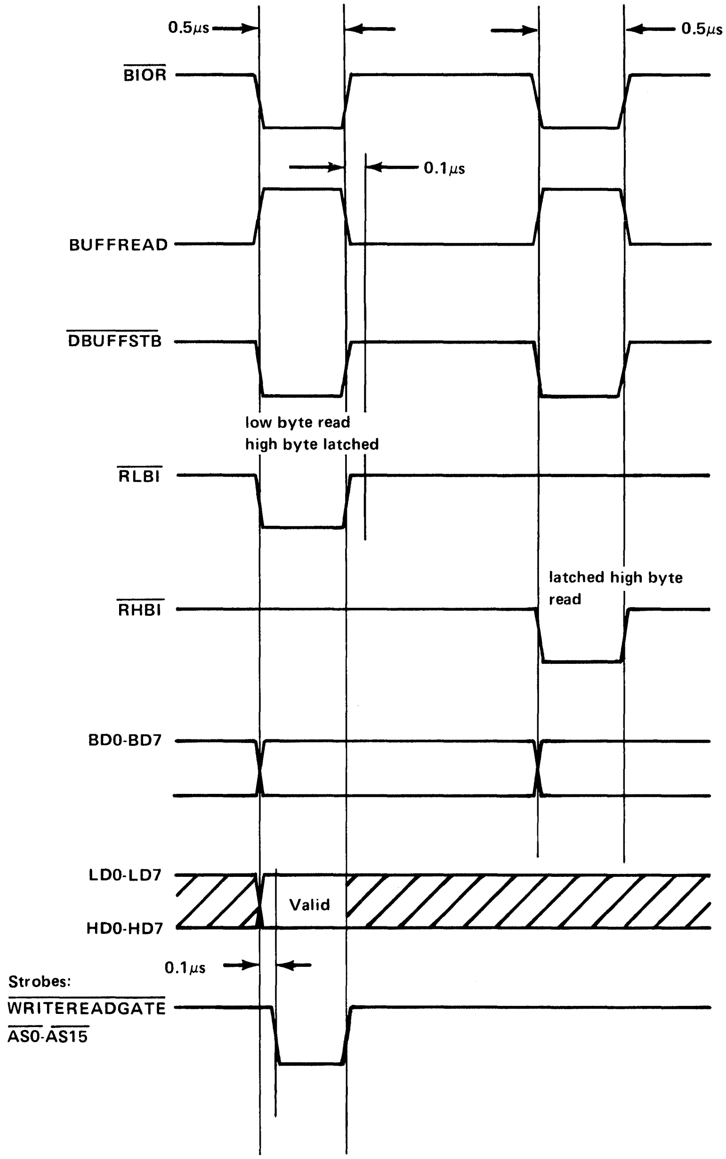
Reading Data

Data is read sequentially from the adapter's 16-bit data bus to the system data bus. The low data byte is first read at an even address, then the high data byte is read at an odd address. The entire 16-bit word is transmitted from an on-board or expansion device at the same time that the low byte is read.

When data is read to the system data bus at an even address (A0 is 0), the $\overline{\text{RLBI}}$ (read low byte in) strobe occurs. The line receiver transmits the low byte from the adapter's 16-bit data bus LD lines, 0 through 7, to the system data bus. At the same time, the high-byte read register latches the data from the adapter's 16-bit data bus HD lines, 0 through 7.

When data is read from the system data bus at an odd address (A0 is 1), the $\overline{\text{RHBI}}$ strobe occurs. The high-byte read register transmits the previously latched high byte to the system data bus.

Following is a read timing diagram of the adapter's 16-bit data bus.



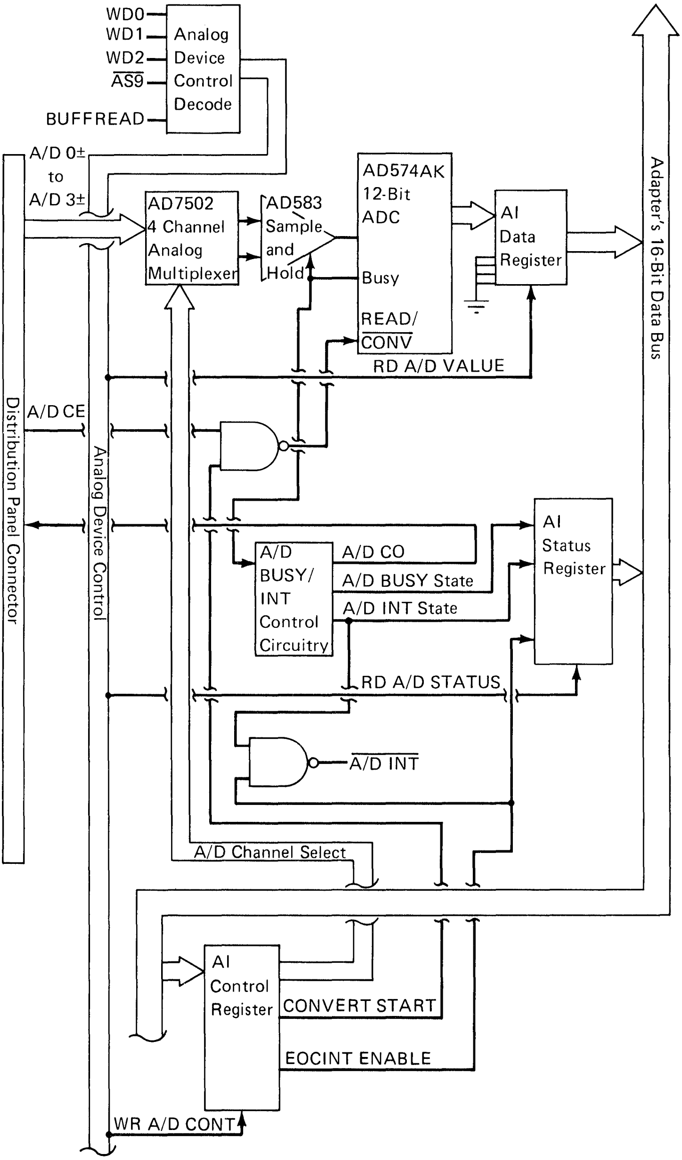
Analog I/O Device

The Data Acquisition Adapter's analog I/O device consists of two subsystems:

- Analog input: An analog-to-digital conversion subsystem.
- Analog output: A digital-to-analog conversion subsystem.

Analog Input Subsystem

On the following page is a block diagram of the analog input subsystem.


WD0
WD1
WD2
AS9
BUFFREAD
Analog Device Control Decode
A/D 0±
to
A/D 3±
AD7502
4 Channel
Analog
Multiplexer
AD583
Sample
and
Hold
AD574AK
12-Bit
ADC
Busy
READ/
CONV
AI Data Register
RD A/D VALUE
Adapter's 16-Bit Data Bus
A/D CE
Distribution Panel Connector
Analog Device Control
A/D BUSY/ INT Control Circuitry
A/D CO
A/D BUSY State
A/D INT State
AI Status Register
RD A/D STATUS
A/D INT
A/D Channel Select
AI Control Register
CONVERT START
EOCINT ENABLE
WR A/D CONT

Analog-to-digital conversion is the process of converting analog signals (voltages) over a given range to digital values.

Unlike digital (binary) signals, which have only two voltage states, analog signals have infinite voltage levels over a particular range.

Analog-to-digital converters (ADCs) are categorized by the number of bits of resolution they allow. The greater the number of bits, the greater the number of discrete voltage levels that can be represented.

The Data Acquisition Adapter has an analog input device with the following features:

- Four, multiplexed, differential channels
- An ADC with 12-bit resolution
- Switch-selectable ranges
- Optional data-conversion control with 'A/D convert out' and 'A/D convert enable in' lines.

The Data Acquisition Adapter's analog input device (device number 9) has four channels, which are multiplexed into a single ADC. This device converts analog signals in one of three ranges to digital values in the range of 0 to 4095.

The three switch-selectable ranges are:

- - 5 to +5 volts
- -10 to +10 volts
- 0 to +10 volts

The relationship of the analog input voltage to the returned digital value depends on the range for which the hardware is configured. The selected range setting for analog input is in effect for all analog input channels. For example, in the -5 to +5 volt configuration, an input of +4.997 volts generates a full-scale value of 4095; an input of 0 volts generates a value of 2048; and an input of -5 volts generates a value of 0.

Analog Input Device Control

The use of the $\overline{\text{AS9}}$ strobe causes the analog input device to be accessed as device number 9.

The control decode circuitry of the analog device decodes WD0 through WD2, $\overline{\text{AS9}}$, and BUFFREAD to generate the following control signals:

WR A/D CONT Write analog-to-digital control. Allows the AI control register to be written to.

RD A/D STATUS Read analog-to-digital status. Allows reading of the AI status register.

RD A/D VALUE Read analog-to-digital value. Allows reading of the AI data register.

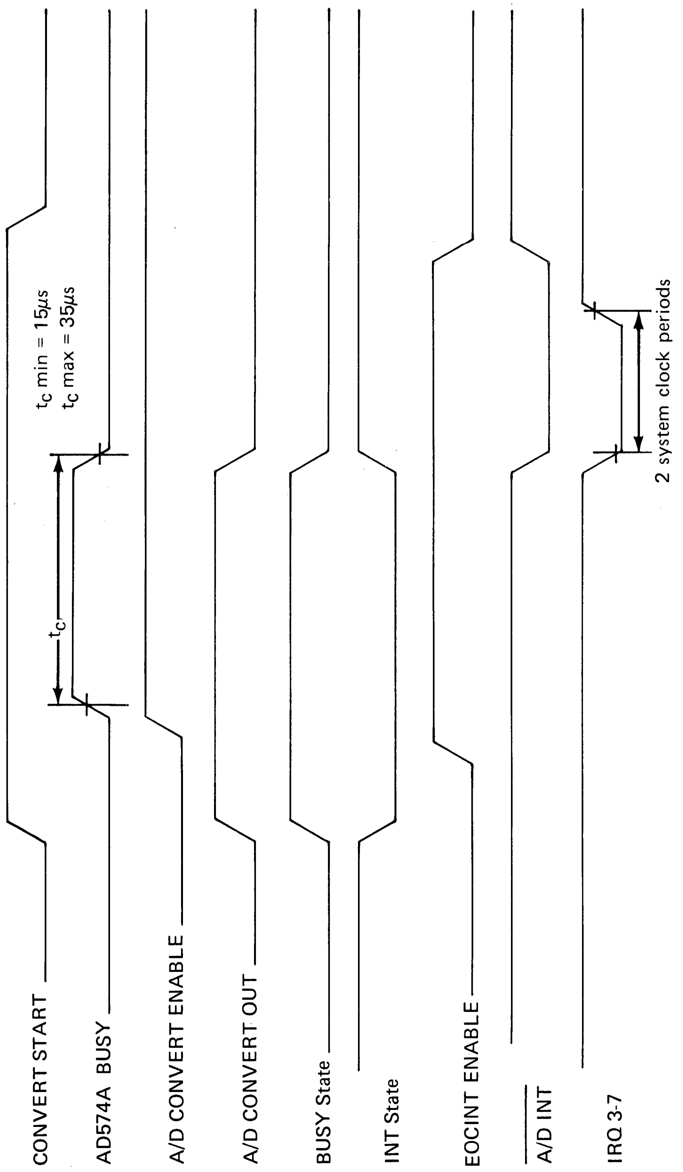
Analog Input Device Registers

AI Control Register The AI control register contains the analog-to-digital channel selection, analog-to-digital interrupt-enable information, and convert start bit information. The AI control register is cleared by $\overline{\text{BUFFRES}}$ during power-on- reset.

AI Status Register The AI status register contains information about 'A/D busy,' the 'A/D interrupt status,' and the readback of the 'A/D interrupt enable.'

AI Data Register The 16-bit AI data register contains the data from the ADC. Because the output of the ADC is a 12-bit digital value, the four highest bits of the register are grounded.

Following is a timing diagram of analog-to-digital conversion.



Starting an Analog-to-Digital Conversion

The convert start bit from the AI control register is logically ANDed with the external 'A/D convert enable' signal from the distribution panel connector. The result is inverted to generate an active low signal, which is brought to the READ/$\overline{\text{CONV}}$ pin of the AD574 ADC.

Reading an Analog-to-Digital Value

The READ/$\overline{\text{CONV}}$ pin must be taken high before the analog-to-digital value can be read. This is accomplished by writing a convert start bit equal to 0 to the AI control register.

Channel Selection

The differential analog to digital channel pair is selected by the AD7502 4-channel, analog multiplexer on the basis of the analog-to-digital channel-select bits of the AI control register.

Sample and Hold

During a conversion, the 'busy' signal from the AD574A ADC causes the AD583 Sample and Hold to hold its present value when the 'busy' signal is high, and starts sampling again when it is low.

'A/D Busy' and Interrupt States

At the end of a conversion, the AD574 ADC's 'busy' signal goes low, and the AI status register shows that the analog input device is in the not-busy and interrupting state.

‘A/D Interrupt’

The actual ‘A/D interrupt’ signal (A/D INT) is a result of the logical ANDing of the INT STATE status bit in the AI status register, and the EOCINT ENABLE bit from the AI control register. The inverted result generates $\overline{\text{A/D INT}}$ (an active low signal), which goes to the interrupt circuitry.

‘A/D Convert Out’

The ‘A/D convert out’ (A/D CO) signal is brought out to the distribution panel connector on the Data Acquisition Adapter.

The ‘A/D convert out’ signal is set (TTL high) when a conversion has been commanded by programming the convert start bit. The signal remains high until the conversion is complete. If the analog signals received by the on-board analog input device are from an external device that can be made to send data on receipt of a TTL high pulse, you may use a synchronization scheme in which the program’s request for an analog-to-digital conversion triggers (using ‘A/D convert out’) the output of analog data from the external device.

‘A/D Convert Enable’

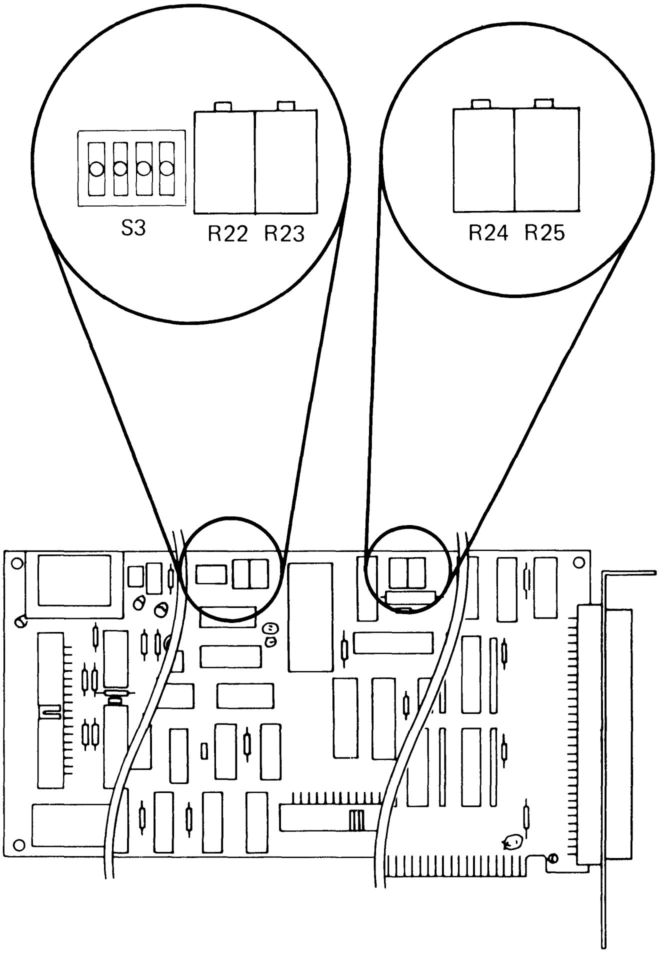
The ‘A/D convert enable’ (A/D CE) signal is brought out to the distribution panel connector on the Data Acquisition Adapter.

By holding the ‘convert enable in’ signal low (TTL), an external device can inhibit or delay all analog-to-digital conversions ordered by programming.

To be considered valid and allow an analog-to-digital conversion, the ‘convert enable in’ signal must remain high until the ‘convert out’ signal goes low again.

Analog Input Potentiometers

Four potentiometers (R22, R23, R24, and R25) on the Data Acquisition Adapter control bipolar offset, unipolar offset, gain, and common mode rejection for the analog input device. The following diagram shows the location of these potentiometers.



In the following "LSB" represents the weight of the least-significant bit of the 12-bit digital output code of the ADC.

The table shows the 1-LSB values for each analog input range.

Range	1 LSB
0 to +10 volts	2.44 mV
-5 to +5 volts	2.44 mV
-10 to +10 volts	4.88 mV

The ADC is intended to have a 1/2-LSB offset so the exact analog input for a given code will be in the middle of that code (halfway between the transitions to the codes above and below it). The information under "Bipolar Offset" and "Unipolar Offset" explains this 1/2-LSB offset.

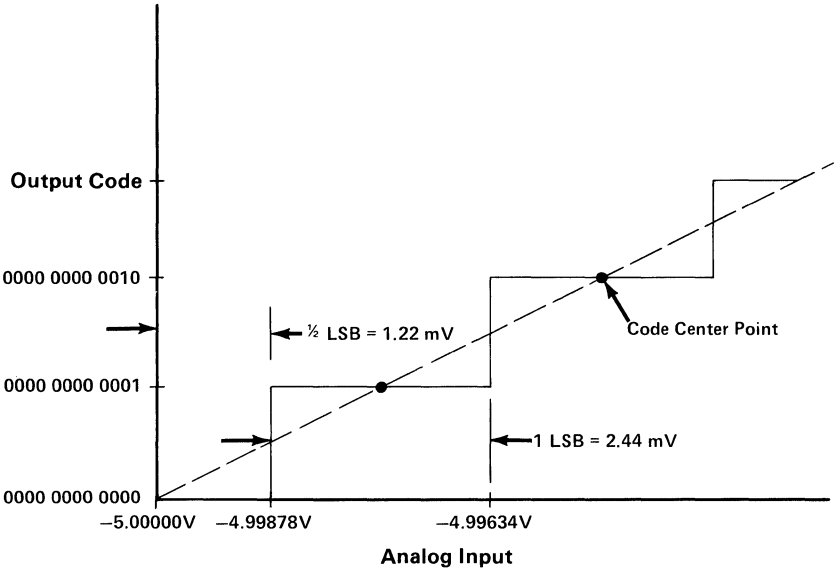
Bipolar Offset:

The value of R22 is set so the transition from the digital output code 0000 0000 0000 to 0000 0000 0001 occurs for an input voltage 1/2 LSB above negative full scale. R22 takes effect when a bipolar range (-5 to +5 volts or -10 to +10 volts) is selected.

The following shows the input voltages for the transition from the output code 0000 0000 0000 to 0000 0000 0001.

Range	Input Voltage for First Code Transition
-5 to +5 volts	-4.99878 volts
-10 to +10 volts	-9.99756 volts

The following shows the first few output-code transitions for the -5 to +5 volt range.



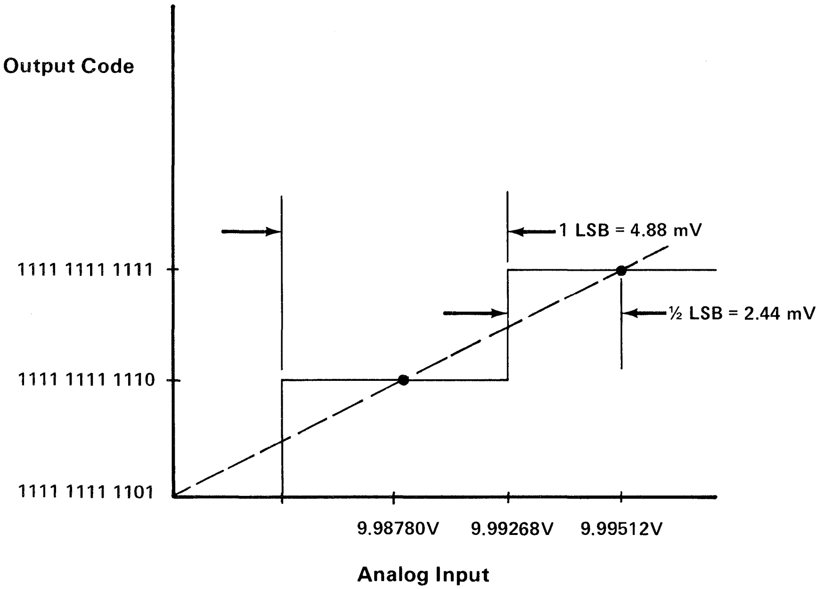
Gain:

The value of R23 is set so the last transition (1111 1111 1110 to 1111 1111 1111) occurs for an input voltage 1-1/2 LSB below full scale.

The following shows the input voltage for the transition from the output code 1111 1111 1110 to 1111 1111 1111.

Range	Input Voltage for Last Code Transition
0 to +10 volts	+9.99634 volts
-5 to +5 volts	+4.99634 volts
-10 to +10 volts	+9.99268 volts

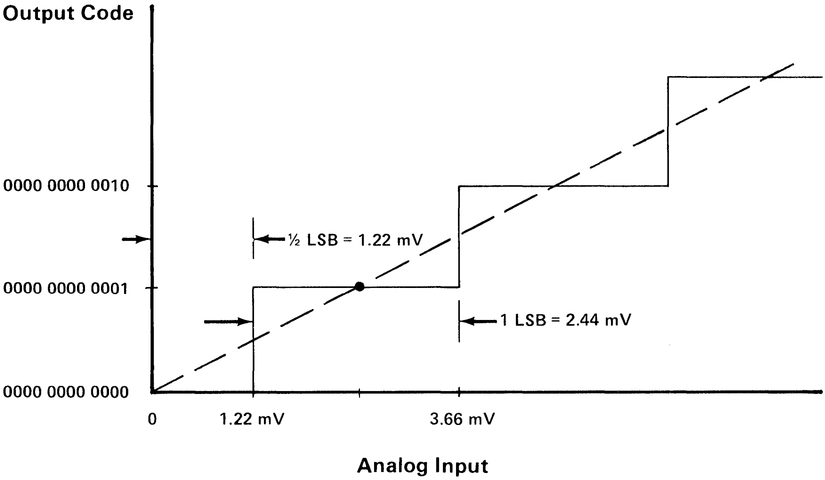
The following shows the last few output-code transitions for the -10 to +10 volt range.



Unipolar Offset:

The value of R24 is set so the first transition (0000 0000 0000 to 0000 0000 0001) occurs for an input voltage of +1/2 LSB. R24 takes effect when the unipolar range (0 to +10 volts) is selected.

The following shows the first few output-code transitions for the 0 to +10 volt range.

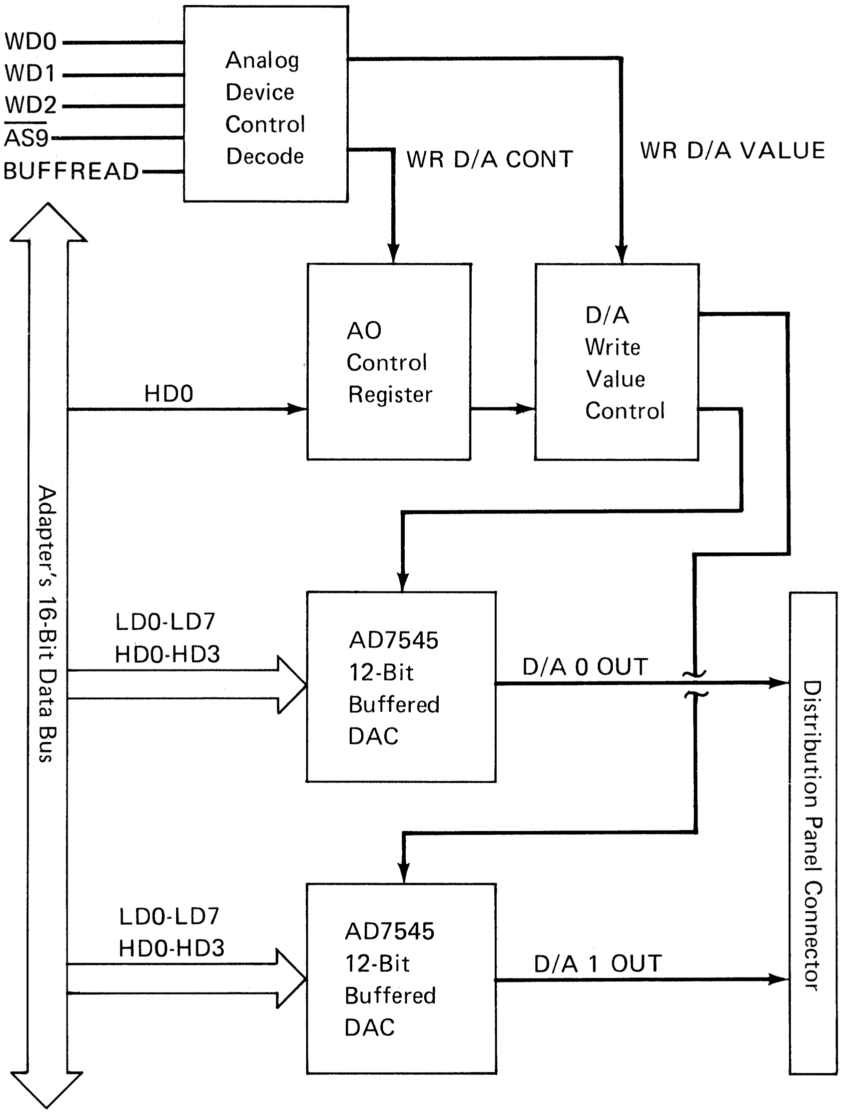


Common Mode Rejection:

R25 allows for the reduction and balancing of the error caused by common mode noise (voltage common to both sides of an analog input channel). The common-mode input range specification for the analog input device is ±11 volts maximum. The value of R25 is set so on the most sensitive range (-5 to +5 volts), the effect of common mode voltage is balanced on each side of zero volts. For example, a common mode voltage of +11 volts produces the same output code as a common mode voltage of -11 volts.

Analog Output Subsystem

Following is a block diagram of the analog output subsystem.



The Data Acquisition Adapter includes an on-board digital-to-analog output device with the following features:

- Two analog output channels, with each channel using separate DACs with 12-bit resolution

- Switch-selectable ranges for each converter

Each DAC converts digital values in the range of 0 to 4095 to voltages in one of three ranges. The switches on the adapter control voltage polarity and range.

The three switch-selectable ranges are:

- -5 to +5 volts

- -10 to +10 volts

- 0 to +10 volts

The settings of these switches determine the relationship between analog output values and the voltages from the analog output device. The relationship of the digital value to the analog output voltage depends on the range for which the hardware is configured. Because each analog output channel has its own DAC, the analog output range can be set for each channel. For example, in the 0 to +10 volt configuration, a digital value of 0 generates an output of 0 volts; a digital value of 2048 generates an output of +5 volts; and a digital value of 4095 generates an output of +9.997 volts.

Analog Output Device Control

The control decode circuitry of the analog device decodes WD0 through WD2, $\overline{\text{AS9}}$, and BUFFREAD to generate the following control signals:

WR D/A CONT

Write digital-to-analog control: Controls the writing of the channel-select bit to the AO control register. The channel-select bit is used to determine which digital-to-analog write strobe to generate.

WR D/A VALUE

Write digital-to-analog value: Controls the generation of the digital-to-analog write strobes. WR D/A VALUE occurs when the AO data register is addressed.

Analog Output Device Registers

AO Control Register

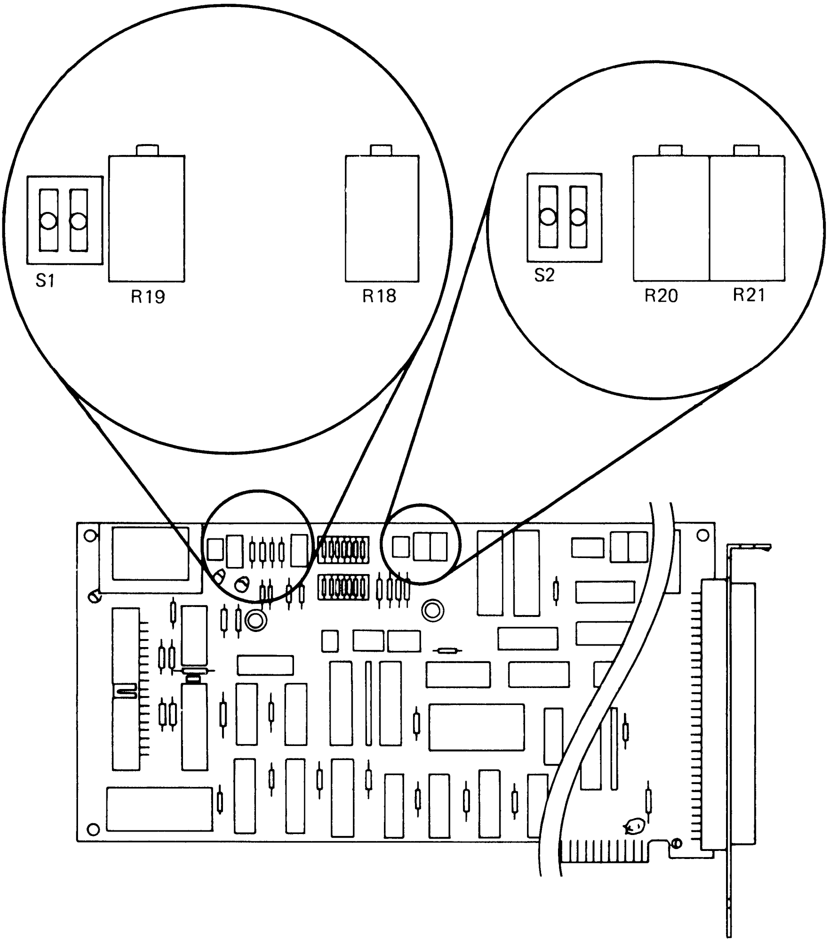
When the 'write D/A value' signal occurs, the digital-to-analog write value control circuitry either strobes D/A 0 or D/A 1, on the basis of the channel-select bit in the AO control register.

AO Data Register

When either AD7545 DAC receives a digital-to-analog write strobe, it latches a 12-bit digital value from the adapter's 16-bit data bus. Each DAC has data latches that are loaded when the AO data register address is written to. The AD7545 DAC then performs the digital-to-analog conversion, and an analog value is sent to the distribution panel connector by 'D/A 0 out' or 'D/A 1 out.'

Analog Output Potentiometers

Four potentiometers (R18, R19, R20, and R21) on the Data Acquisition Adapter control bipolar offset and gain for the analog output device. The following diagram shows the location of these potentiometers.



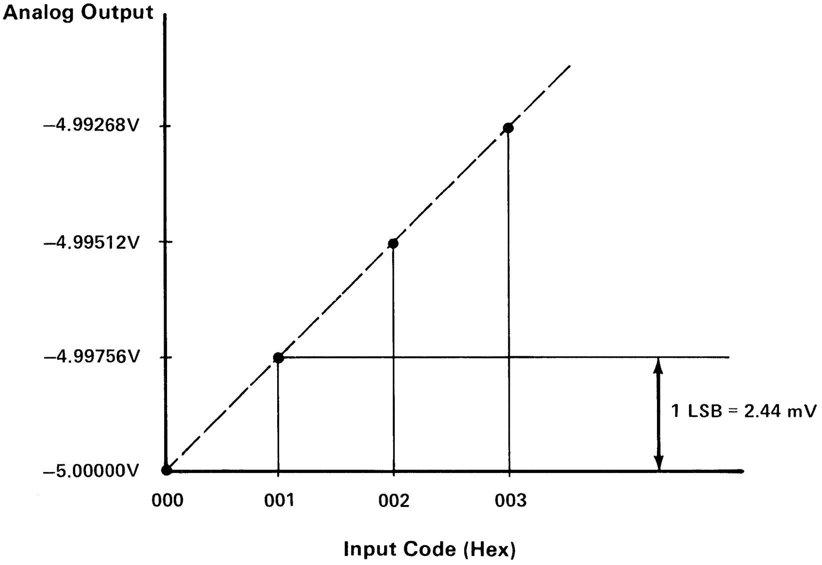
Bipolar Offset:

R18 controls bipolar offset for channel 0, and R20 controls it for channel 1. The value of the potentiometer is set so a negative full-scale voltage is provided on the analog output channel when the digital code 0000 0000 0000 is sent to the DAC for that channel. The potentiometer takes effect when a bipolar range is selected. When the unipolar range is selected, the DAC output is 0 volts.

The following table lists the output voltages for the digital code 0000 0000 0000 (000 hex).

Range	Output Voltage for 000 Hex Code
0 to +10 volts	0.00000 volts
-5 to +5 volts	-5.00000 volts
-10 to +10 volts	-10.00000 volts

The following shows the first few analog output voltages for the -5 to +5 volt range.



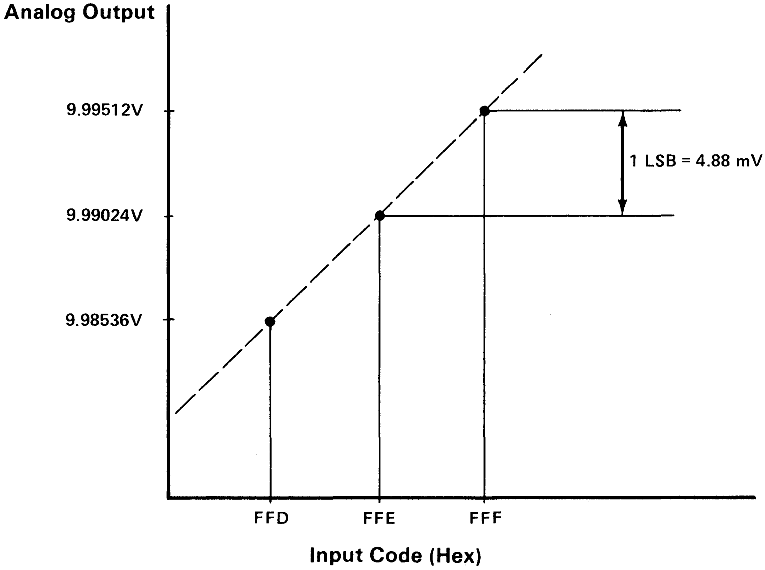
Gain:

R19 controls gain for channel 0, and R21 controls it for channel 1. The value of the potentiomenter is set so a positive full-scale −1 LSB voltage is provided on the analog output channel when the digital code 1111 1111 1111 is sent to the DAC for that channel.

The following table lists the output voltages for the digital code 1111 1111 1111 (FFF hex).

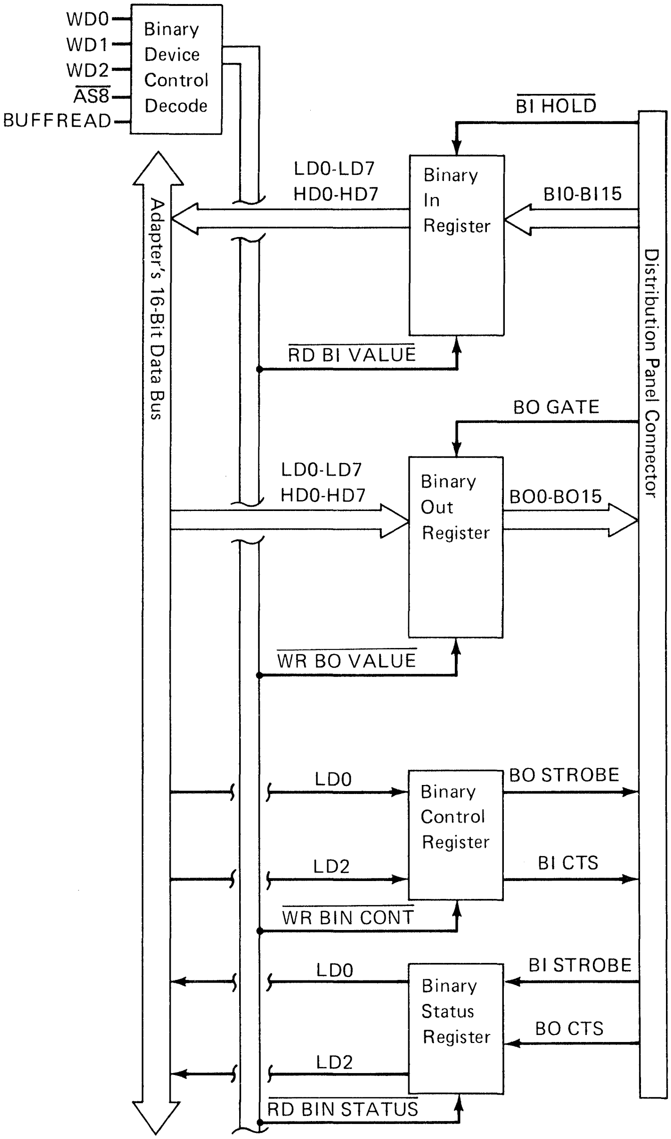
Range	Output Voltage for FFF Hex Code
0 to +10 volts	+9.99756 volts
-5 to +5 volts	+4.99756 volts
-10 to +10 volts	+9.99512 volts

The following shows the last few analog output voltages for the -10 to +10 volt range.



Binary I/O Device

Following is a block diagram of the binary I/O device.


WD0
WD1
WD2
$\overline{\text{AS8}}$
BUFFREAD
Binary Device Control Decode
$\overline{\text{BI HOLD}}$
LD0-LD7
HD0-HD7
Binary In Register
BI0-BI15
Adapter's 16-Bit Data Bus
Distribution Panel Connector
$\overline{\text{RD BI VALUE}}$
BO GATE
LD0-LD7
HD0-HD7
Binary Out Register
BO0-BO15
$\overline{\text{WR BO VALUE}}$
LD0
Binary Control Register
BO STROBE
LD2
BI CTS
$\overline{\text{WR BIN CONT}}$
LD0
Binary Status Register
BI STROBE
BO CTS
LD2
$\overline{\text{RD BIN STATUS}}$

The Data Acquisition Adapter's binary I/O device has the following features:

- A 16-bit binary output port (BO0 through BO15)
- A 16-bit binary input port (BI0 through BI15)
- Input and output handshaking over the 'strobe' and 'clear-to-send' lines
- Direct control using BO GATE ('binary out gate') and $\overline{\text{BI HOLD}}$ ('binary in hold').

Digital signals have only two voltage states: On (high, +3 volts) and Off (low, +0.2 volts). Digital signals in this range are called *TTL signals*, because they are the proper levels to be interpreted by the transistor-to-transistor logic circuitry. These signals have many uses in data acquisition and control applications. Among these are sensing the state of two-state devices and controlling devices that require two-state control signals.

Binary I/O Device Control

The use of the $\overline{\text{AS8}}$ strobe causes the binary I/O device to be accessed as device number 8.

The $\overline{\text{AS8}}$ strobe as an enable, the WD0 through WD2 word bits, and the BUFFREAD signal are used to decode which binary decode operation is to occur.

Following are the four decode operations:

$\overline{\textbf{WR BIN CONT}}$ Write binary control: Controls the latching of the binary output strobe (BO STROBE) and the binary input clear-to-send (BI CTS) bits by the binary control register.

$\overline{\textbf{RD BIN STATUS}}$ Read binary status: Controls the reading of the binary input strobe (BI STROBE) and the binary output clear to send (BO CTS) bits by the binary status register.

$\overline{\textbf{WR BO VALUE}}$ Write binary value: Controls the writing of the binary output word (BO0 through BO15) to the binary output register.

$\overline{\textbf{RD BI VALUE}}$ Read binary value: Controls the reading of the binary input word (BI0 through BI15) from the binary input data register.

Binary I/O Device Registers

Following is a description of the binary I/O device registers.

Binary Control Register

Contains the BO STROBE bit and the BI CTS bit. These bits do not physically cause or prevent binary I/O events from occurring. They are programming control bits.

Binary Status Register

Allows the status of BO CTS and BI STROBE bits to be monitored. These bits do not physically cause or prevent binary I/O events from occurring. They are programming status bits.

Binary Input Register

When $\overline{\text{BI HOLD}}$ is brought high (or if no connection is made), the binary input register is not latched and allows the current state of the binary input lines to be monitored by reading the binary input register. Grounding $\overline{\text{BI HOLD}}$ causes the binary input register to latch the current state of all binary input lines. If the grounding of the $\overline{\text{BI HOLD}}$ line is maintained, any later read will obtain the value that was present when the line was initially grounded.

Binary Output Register

Contains the binary output word (BO0 through BO15). Grounding the BO GATE signal places the binary output port in the tri-state condition (all points floating). The binary outputs are gated out when the BO GATE signal is brought high (or if no connection is made).

The Data Acquisition Adapter's binary I/O device consists of two subsystems that use low-power Schottky logic:

- Binary input
- Binary output

Binary Input Subsystem

Following is a description of the binary input subsystem.

Binary Input Port (BI0 through BI15)

All bits of the binary input port (BI0 through BI15) are pulled to their high state internally. This means that if nothing is connected to the binary input port, execution of a binary input function returns a value of 65535 (all bits set to 1).

The input port of the Data Acquisition Adapter's binary I/O device can be used to sense the state of up to 16 individual binary signals.

The binary input port also can be used for input of binary data words (16-bit) from another device.

Binary Input Hold

The entire binary input port may be latched at any time by pulling the $\overline{\text{BI HOLD}}$ signal low. These and all other data and communication lines are pulled high through internal resistors to +5 volts. No connections to them are necessary unless their features are to be used.

Binary Input Handshaking

Binary input samples can be synchronized with binary words generated by an external device. The external device must be able to send parallel binary data when it receives a signal from the Data Acquisition Adapter's binary I/O device. It also must be able to generate a TTL signal that indicates the data word is valid and should be sent by the Data Acquisition Adapter.

Binary Output Subsystem

Following is a description of the binary output subsystem.

Binary Output Port (BO0 through BO15)

This subsystem uses high-power, tri-state, bus-driving devices. Changes in the binary output word are carried out on a per-bit basis. Only those bits affected by a change in the output word are actually changed. All others remain the same.

The output port of the binary I/O device supplies 16 high/low signals under program control. As with the input port, these signals can be used individually or considered as a 16-bit data word.

Binary Out Gate

You may place the output port in tri-state by pulling the binary out gate (BO GATE) lines low. These and all other data, handshaking, and control lines are pulled high by internal resistors to +5 volts. No connections to them are necessary unless your application requires handshaking or control.

Binary Output Handshaking

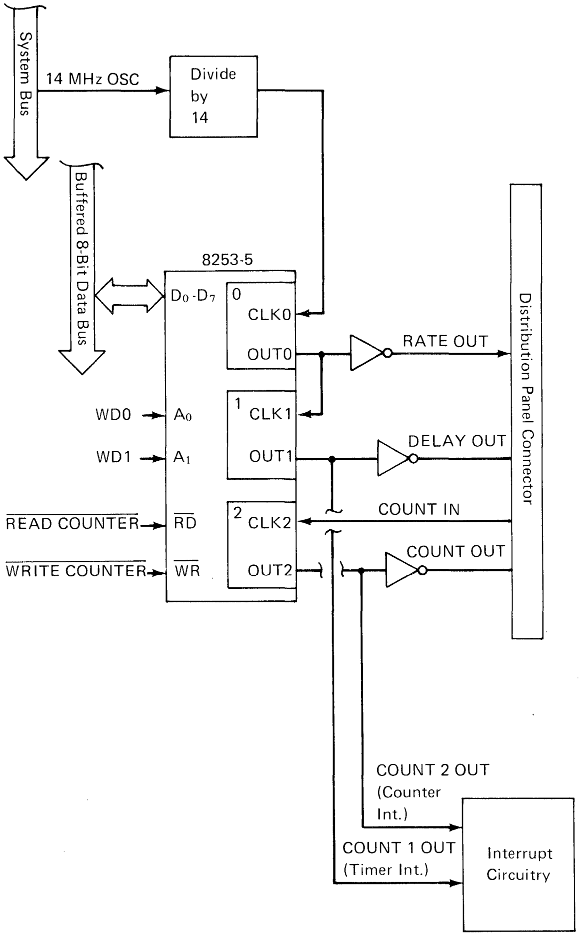
Because all communication lines are internally pulled up to their logical true state, you can use or not use binary output handshaking, depending on the requirements of your communication setup.

Binary output can be synchronized with the data input capabilities of the external device. The external device must be able to send a TTL signal to indicate it is ready for new data. It also must be able to accept parallel binary data when it recieves a signal from the Data Acquisition Adapter's binary I/O device indicating the data is available.

Timer/Counter Device

The timer/counter device is an 8253-5 Programmable Interval Timer. The timer/counter device provides three independent, down-counting, 16-bit counters. Each counter can be programmed to operate in one of four modes. In this implementation, Counters 0 and 1 are cascaded to provide a 32-bit timer. Counter 2 is not cascaded and provides an independent 16-bit timer/counter. These counters can be used to generate interrupts, provide pulses (or pulse trains) to the distribution panel connector (RATE OUT, DELAY OUT, and COUNT OUT), and count events (COUNT IN).

Following is a diagram of the timer/counter device.



Timer/Counter System Interface

Following is a description of how the Data Acquisition Adapter controls its timer/counter device.

BD0 - BD7 The Data Acquisition Adapter's buffered data bus lines are connected to data lines (D0 through D7) of the timer/counter device's internal data-bus buffer.

$\overline{\text{RD CNT}}$ Read counter: Connected to the RD pin of the timer/counter device. Used as a control signal when reading the values of the counters.

$\overline{\text{WR CNT}}$ Write counter: Connected to the WR pin of the timer/counter device. Used as a control signal when writing mode information and loading the counters.

WD0 and WD1 Connected to the A0 and A1 pins of the timer/counter device. They select which of the three counters to be operated on, and address the control register.

The following lists the resulting timer/counter device operations performed based on the values of the timer/counter device's address and control signals.

Note: The CS pin of the timer/counter device is tied low.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	Description
0	1	0	0	0	Load Counter 0
0	1	0	0	1	Load Counter 1
0	1	0	1	0	Load Counter 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter 0
0	0	1	0	1	Read Counter 1
0	0	1	1	0	Read Counter 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3-State

32-Bit Timer

Following is a description of the output of the 32-bit timer (Counters 0 and 1), and how it is clocked.

First Stage (Counter 0)

CLK 0 A 1.023-MHz signal (50% duty cycle) from the system bus' 14-MHz OSC and divide-by-14 circuitry.

OUT 0 Output of Counter 0.

RATE OUT Inverted state of OUT 0 that is brought to the distribution panel connector.

Second Stage (Counter 1)

CLK 1 The Counter-1 clock. The output of Counter 0 (OUT 0) is cascaded into the counter of clock 1.

OUT 1 Output of Counter 1. Provides the 'count 1 out' signal that is used by the interrupt circuitry.

DELAY OUT Inverted state of OUT 1 that is brought to the distribution panel connector.

16-Bit Timer/Counter

Following is a description of the output of the 16-bit timer/counter (Counter 2), and how it is clocked.

CLK 2 The Counter-2 clock.

COUNT IN Clocks Counter 2.

OUT 2 Output of Counter 2. Provides the ‘count 2 out’ signal used by the interrupt circuitry.

COUNT OUT Inverted state of OUT 2 that is brought to the distribution panel connector.

Counter Modes

The following counter modes apply to Counters 0 through 2.

> **Note:** "Output" in the following timing diagrams refers to the OUT 0, OUT 1, and OUT 2 pins of the timer/counter device. Counter outputs, RATE OUT, DELAY OUT, and COUNT OUT, on the distribution panel connector are the inverted state of OUT 0, OUT 1, and OUT 2.

Mode 0: Interrupt on Terminal Count

Initially, the output is low after the mode-set operation. After the count is loaded into the selected count register, the output remains low, and the counter counts. When terminal count is reached, the output goes high and remains high until the selected count register is reloaded with the mode or a new count is loaded. The counter continues to decrease after terminal count is reached.

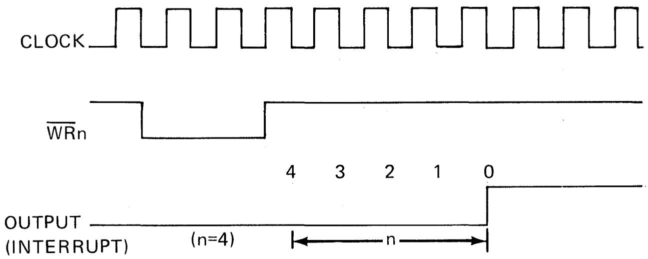
Rewriting a counter register during counting results in the following:

- A Write to the first byte stops the current counting.

- A Write to the second byte starts the new count.

Following is the timing diagram for mode 0.

Mode 0: Interrupt on Terminal Count



Mode 1: Programmable One-Shot

This mode is not used because the timer/counter device's gate pins (GATE 0 through GATE 2) are tied high.

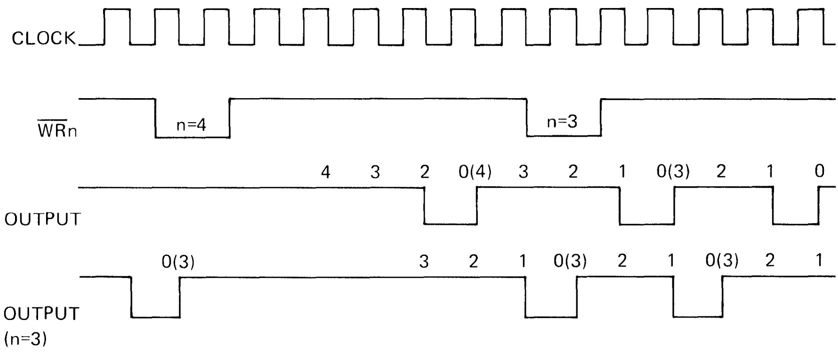
Mode 2: Rate Generator

Divide-by-N counter. The output is low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses, the present period is not affected, but the next period reflects the new value.

When the mode is set, the output remains high until the count register is loaded. The output can then also be synchronized by programming.

Following is the timing diagram for mode 2.

Mode 2: Rate Generator



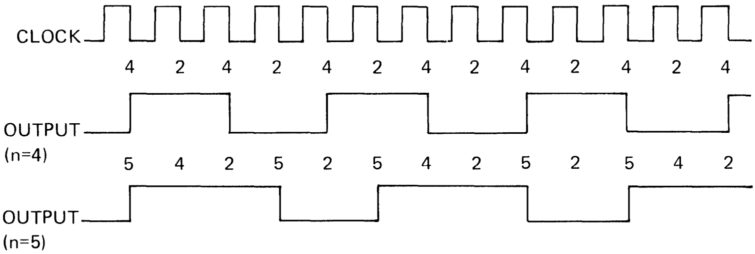
Mode 3: Square-Wave Rate Generator

This mode is similar to mode 2, except the output remains high until half the count is complete (for even numbers), then goes low for the other half. This is accomplished by decrementing the counter by 2 on the falling edge of each clock pulse. When the counter reaches terminal count, the state of the output changes, and the counter is reloaded with the full count; the complete process then repeats.

If the count is odd and the output high, the first clock pulse after the count is loaded, decreases the count by 1. Subsequent clock pulses decrease the clock by 2. After time-out, the output goes low and the full count is reloaded. The first clock pulse after the reload, decreases the counter by 3. Subsequent clock pulses decrease the count by 2 until time-out. Then the complete process repeats. In this way, if the count is odd, the output will be high for (N + 1)/2 counts and low for (N - 1)/2 counts.

Following is the timing diagram for mode 3.

Mode 3: Square Wave Generator



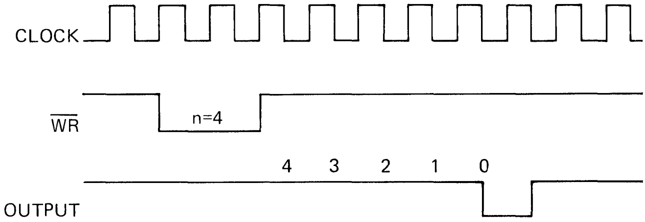
Mode 4: Software-Triggered Strobe

After the mode is set, the output is high. When the count is loaded, the counter begins counting. When the counter reaches terminal count, the output goes low for one input clock period, then goes high again.

If the count register is reloaded during counting, the new count is loaded on the next CLK pulse.

Following is the timing diagram for mode 4.

Mode 4: Software Triggered Strobe

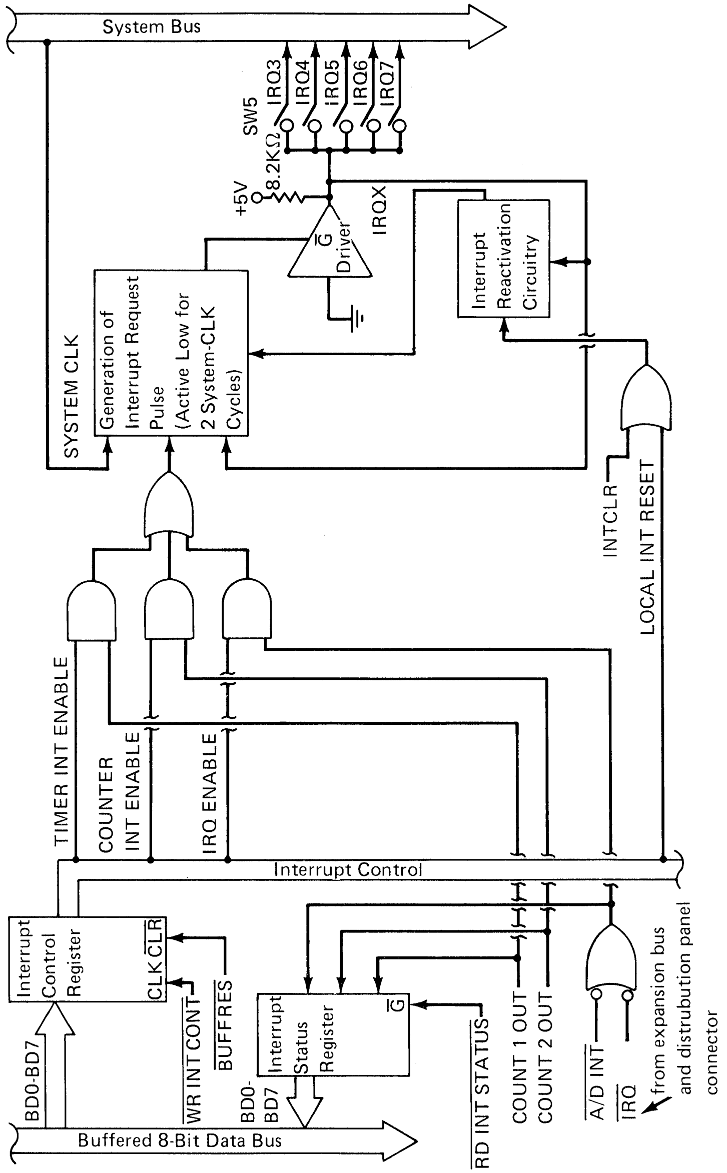


Mode 5: Hardware-Triggered Strobe

This mode is not used because the pins for gates 0 through 2 are tied high.

Interrupt Circuitry

The following is a block diagram of the interrupt circuitry.



The Data Acquisition Adapter can generate an interrupt from the following individually maskable sources:

- 32-bit timer (cascaded 16-bit Counters 0 and 1 of the timer/counter device)
- 16-bit externally-clocked timer/counter (Counter 2 of the timer/counter device)
- ADC 'end of conversion' signal
- IRQ external interrupt (on the distribution panel connector and the expansion bus).

Interrupts generated by the Data Acquisition Adapter can be set to an interrupt level in the range of IRQ3 through IRQ7. The interrupt level is set with the switches of S5 on the adapter and must be set before adapter installation. IRQ7 is recommended.

Interrupt Control Register

Following is a description of the bits of the interrupt control register.

Bit 0 TINT ENABLE: Enables 32-bit timer interrupts (Counters 0 and 1).

Bit 1 CINT ENABLE: Enables 16-bit timer/counter interrupts (Counter 2).

Bit 2 IRQ ENABLE: Enables the $\overline{\text{IRQ}}$ (external interrupt) line and analog-to-digital end-of-conversion interrupts as sources of interrupts.

Bit 3 Reserved for reading status.

Bit 4 - 6 Not used.

Bit 7 LINT RESET: Performs the local-reset function.

> **Note:** Power-on-reset ($\overline{\text{BUFFRES}}$) resets the interrupt control register (clearing all bits and disabling interrupts).

Interrupt Status Register

Following is a description of the bits of the interrupt status register.

Bits 0 - 3 Provided for reading the status of the corresponding interrupt-enable bits listed under "Interrupt Control Register". Bit 3 is not currently used.

Bit 4 TINT STAT: The timer interrupt status (COUNT 1 OUT), which is the state of the output of the second chained timer stage (Counter 1 of the timer/counter device).

Bit 5 CINT STAT: The counter interrupt status (COUNT 2 OUT), which is the state of the output of the 16-bit timer/counter (Counter 2).

Bit 6 IRQ STAT: The $\overline{\text{IRQ}}$ (external-interrupt) status. The on-board 'A/D interrupt' signal is logically ORed into the $\overline{\text{IRQ}}$ (external-interrupt) function.

Bit 7 Read back as a 0.

Interrupt Request Pulse

The corresponding three interrupt-enable and three interrupt-status lines are logically ANDed. Any combination of the three interrupt sources may be enabled. The enabled interrupts are logically ORed to generate the adapter-interrupt trigger signal. This signal causes the generation of an 'interrupt request out' pulse by enabling a tri-state driver to be active low for two cycles of the system clock. The output of this driver is connected to the desired system interrupt level (IRQ3 through IRQ7) by switch S5. When not active low, the tri-state driver is floating and allows other adapters to share the interrupt line.

Interrupt Reactivation

When the shared-interrupt line pulses low, regardless of whether the Data Acquisition Adapter or another interrupt-sharing adapter was the source, the interrupt-reactivation circuitry prevents the Data Acquisition Adapter from generating interrupts. Thus, a single interrupt causes deactivation of additional interrupts. Additional interrupts are reactivated by either the 'local reset' signal (only one Data Acquisition Adapter reactivated) or the INTCLR signal (all interrupt-sharing adapters reactivated). A logical OR of the 'local reset' or INTCLR shared-interrupt reactivation signals starts the interrupt-reactivation circuitry.

Following is a description of the two shared-interrupt reactivation signals.

Local Interrupt Reset

The local-interrupt-reset bit in the interrupt control register controls the reactivation of only one Data Acquisition Adapter. The particular adapter is singled out by the adapter-number bits (A10 and A11) in the adapter's I/O address space.

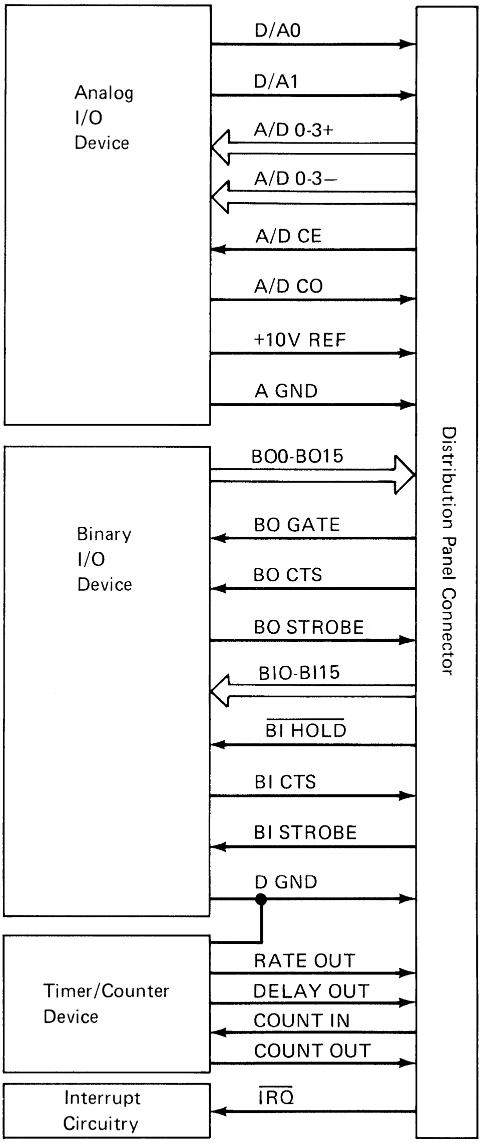
Global Interrupt Reset

A global-interrupt reset also can be performed. This resets the interrupt circuitry of all adapters sharing a particular interrupt level. The only requirement is that the adapters support interrupt sharing. The Data Acquisition Adapter does support interrupt sharing.

To perform a shared-interrupt global reset, an I/O Write to an address hex 02Fx (02F3 through 02F7) is performed for a particular interrupt level (IRQ3 through IRQ7). Thus, to reset all adapters sharing interrupt IRQ7, an I/O Write to hex 02F7 is performed. The output value is not important.

Distribution Panel Connector

Following is a block diagram of the signals of the distribution panel connector, J4.



Distribution Panel Connector Signals

The following is a description of how the distribution panel connector, J4, provides access to the interrupt circuitry, and the analog I/O, binary I/O, and timer/counter devices.

D/A 0, D/A 1 DAC outputs.

EXT +10 V REF 10-volt reference output.

A GND Analog ground: The system ground reference for the analog devices in the system.

A/D 0- to A/D 3- Channels 0 through 3 ADC inputs (low).

A/D 0+ to A/D 3+ Channels 0 through 3 ADC inputs (high).

D GND Digital ground: The system ground reference for the digital devices in the system.

A/D CE 'Convert enable' input for the ADC. When high, enables conversion.

A/D CO 'Convert out' indicator for the ADC. When high, indicates a conversion was requested. Returns to low when the conversion is complete.

$\overline{\text{BI HOLD}}$ Binary-input hold: Latch control for the binary input port. When low, the binary input device latches the state of all input lines.

BI STROBE (Input) Binary input strobe: When high, indicates the binary input data is available.

BI0 - BI15 Binary input port, bits 0 through 15.

BO0 - BO15 Binary output port, bits 0 through 15.

BO GATE (Input) Binary output gate: Tri-state enable for the binary output port. When low, the binary outputs (BO0 through BO15) are floating.

BO STROBE (Output) Binary output strobe: When high, indicates that new data was sent.

BO CTS (Input) Binary output clear-to-send: When high, permits new binary-output port data to be sent.

BI CTS (Output) Binary input clear-to-send: When high, indicates the binary input data is being requested.

$\overline{\text{IRQ}}$ (Input) External-device, interrupt-request: Active low.

RATE OUT Output from the first stage of the 32-bit timer (Counter 0). The 'rate out' signal is the inverse of the timer/counter device output.

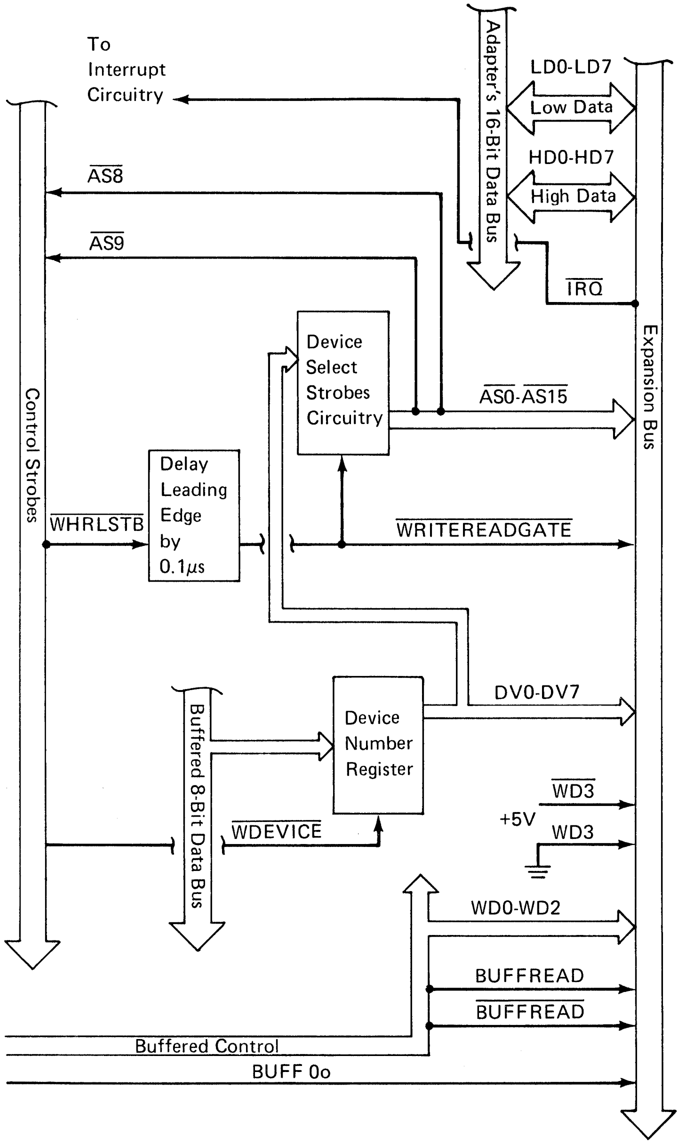
DELAY OUT Output from the second stage of the 32-bit timer (Counter 1). The 'delay out' signal is the inverse of the timer/counter device output.

COUNT OUT Output from the 16-bit counter device (Counter 2). The 'count out' signal is the inverse of the timer/counter device output.

COUNT IN Input to the 16-bit counter device (Counter 2).

Expansion Bus

Following is a block diagram showing device selection signals, control signals, and data bus lines which make up the expansion bus.



The expansion bus is an expansion interface for data acquisition and control adapters. The bus consists of two 34-pin transition connectors, J1 and J2, on the Data Acquisition Adapter.

All drivers on the bus are intended to be low-power Schottky (LS) TTL bus drivers or equivalent devices. Such devices can drive below 0.4 volts at 12 milliamperes load current, and above 2.4 volts at 2.6 milliamperes.

All receivers on the bus are intended to be no more than two LS TTL loads for each external device on any bus line. Such devices will present a load current of no more than 0.8 milliamperes sourcing at 0.4 volts, and no more than 40 microamperes sinking at 2.4 volts. A single LS TTL load for each external device on any bus line is preferred.

Drivers on the bidirectional data lines must be tri-state devices enabled only during the appropriate strobes.

Expansion Bus Signals

Following is a description of the signals on the expansion bus.

BUFF 0_0 (Output) OSC signal divided by 14 (1.023 MHz).

$\overline{\textbf{WRITEREADGATE}}$ (Output) Active low strobe for all devices. $\overline{\text{AS16}}$ through $\overline{\text{AS255}}$ strobes can be created by decoding DV0 through DV7 and using the signal, $\overline{\text{WRITEREADGATE}}$.

BUFFREAD (Output) When high, indicates a read is occurring. When low, indicates a write is occurring.

$\overline{\textbf{BUFFREAD}}$ (Output) Inverse of BUFFREAD.

$\overline{\textbf{BUFFRES}}$ (Output) inverse of RESET DRV. Performs system reset and initialization.

$\overline{\textbf{IRQ}}$ (Input) External, interrupt request. Active low.

$\overline{\textbf{AS0}}$ to $\overline{\textbf{AS15}}$ (Outputs) Active low strobes for devices 0 through 15. Must be low for 0.4 microseconds.

$\overline{\text{WD3}}$ Word number bit 3, fixed high.

WD3 Word number bit 3, grounded.

HA4 - HA7 Reserved expansion signals; grounded.

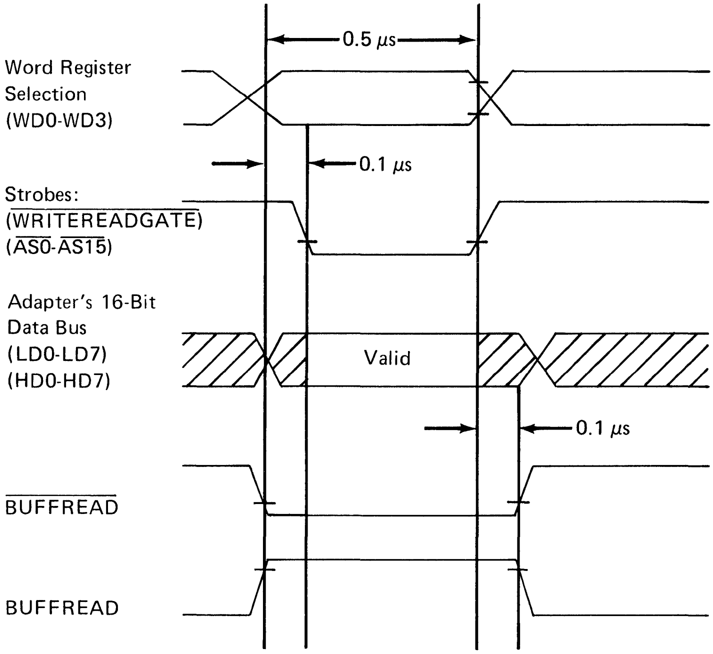
DV0 - DV7 (Output) Device-number bits. Select one of 256 possible devices.

LD0 - LD7 (Inputs/Outputs) Low byte of the adapter's 16-bit data bus.

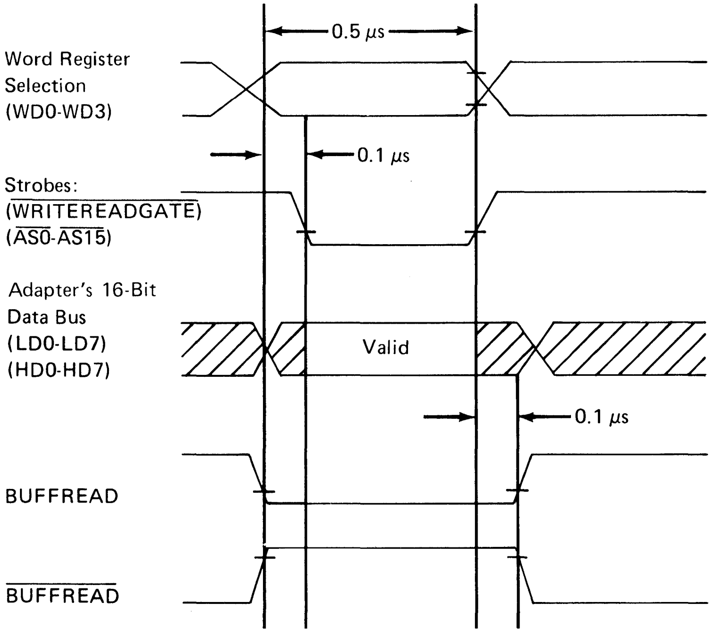
HD0 - HD7 (Inputs/Outputs) High byte of the adapter's 16-bit data bus.

WD0 - WD2 Word number bits 0 through 2 (system-bus address lines A12 through A14 are buffered and renamed WD0 through WD2).

Following is a diagram of the expansion bus read timing.


0.5 μs
Word Register Selection (WD0-WD3)
0.1 μs
Strobes: (WRITEREADGATE) (AS0-AS15)
Adapter's 16-Bit Data Bus (LD0-LD7) (HD0-HD7)
Valid
0.1 μs
BUFFREAD
BUFFREAD

Following is a diagram of the expansion bus write timing.


0.5 μs
Word Register Selection (WD0-WD3)
0.1 μs
Strobes: (WRITEREADGATE) (AS0-AS15)
Adapter's 16-Bit Data Bus (LD0-LD7) (HD0-HD7)
Valid
0.1 μs
BUFFREAD
BUFFREAD

Programming Considerations

This section describes the programming considerations for the Data Acquisition Adapter.

Address Decoding

The following table shows address decoding.

A15 A14 A13 A12	A11 A10	A9 A8 A7 A6 A5 A4 A3 A2 A1	A0
Register Select	Card Select	1 0 1 1 1 0 0 0 1	Byte Select

- Register Select selects one of 16 word registers (0 through 15). Only registers 0 through 13 are used.
- Card Select selects the adapter number (0 through 3).
- A1 through A9 are a fixed pattern to select Data Acquisition Adapters.
- A0 selects the high or low byte of a 16-bit word register.

The base addresses of the Data Acquisition adapters are:

Adapter Number	Base Address (Hex)
0	02E2
1	06E2
2	0AE2
3	0EE2

Registers

Each Data Acquisition Adapter has 16 (two-byte) word registers through which all access to the Data Acquisition Adapter is made. The registers allow access to the Data Acquisition Adapter's on-board and expansion data-acquisition and control devices, the adapter's interrupt registers and device number register, and the timer/counter device registers.

The following table shows the Data Acquisition Adapter's registers.

Register	Name	Function
0	Device Register 0	Write values to, and read values from, the on-board and external devices
1	Device Register 1	
2	Device Register 2	
3	Device Register 3	
4	Device Register 4	
5	Device Register 5	
6	Device Register 6	
7	Device Register 7	
8	Timer/Counter 0 Register	Read/load Counter 0
9	Timer/Counter 1 Register	Read/load Counter 1
10	Timer/Counter 2 Register	Read/load Counter 2
11	Timer/counter Control Register	Controls the operation of Counters 0 through 2.
12	Device Number	Selects device
13	Interrupt Registers	Interrupt control and status
14		Not used
15		Not used

Device Registers

Eight register addresses have been reserved for reading and writing the registers on the on-board and external devices. The device is selected using the device-number register. These registers may then be used for access to the registers of that device. These registers are both read and write registers, and often two different functions will be decoded for the read and write.

Analog Input Device Registers

The on-board analog input (AI) device, accessed as device number 9, has four channels. External AI devices can be added to the expansion interface. These external AI devices have up to 256 channels, use ADCs with up to 16 bits of resolution, and are accessed with a different device number, but with the same register format as the on-board AI device.

The AI device has the following registers.

Register	Read/Write	Name	Function
0	Write	AI Control	Sets up and controls the register.
0	Read	AI Status	Returns status of the hardware.
2	Read	AI Data	Returns current analog value.

Following is a description of the bits of the AI control register.

Bit	Name	Function
0	Convert Start	Setting this bit starts an analog-to-digital conversion.
1	Short Cycle	Reserved for enabling a short cycle conversion.
2	EOCINT Enable	End-of-conversion interrupt enable. When set, an end-of-conversion (conversion complete) will generate an interrupt.
3-7		Not used
8-15	Channel	Channels 0 through 255. On-board AI device uses only channels 0 through 3.

Following is a description of the bits of the AI status register.

Bit	Name	Function
0	Busy State	When set, indicates that the ADC is in the process of doing a conversion and data is not yet valid.
1	Int State	When set, indicates that a conversion has ended (not busy and interrupting state). If the EOCINT enable bit is set (enabled), an interrupt is generated.
2	EOCINT Enable	Read back of state of EOCINT enable bit in AI control register.
3 - 15		Not used

Following is a description of the bits of the AI data register.

Bit	Name	Function
0-15	Data	Contains the data value from the last conversion. Valid only if the busy bit in the AI status register is cleared. Bits 0 through 11 will contain data from the 12-bit ADC. Bits 12 through 15 will all be zeros.

AI Device Access Strategy

The following outlines the access strategy for the AI device. Note that because channel selection takes a while, routines that repeatedly access a single channel would want to select the channel only once before data acquisition.

Polling Method

To use the polling method, do the following:

1. Set the device number in the device-number register to 9.
2. Set the channel and simultaneously disable conversion by setting the convert start bit to 0.
3. Wait for the channel multiplexer to settle (approximately 20 microseconds).
4. Request a conversion by setting the convert start bit to 1 and set the same channel.
5. Wait for the busy stat bit to equal 0.
6. Enable reading of the ADC's data by setting the convert start bit to 0, and set the same channel.
7. Read the data value.

Interruption Method

To use the interruption method, do the following:

1. Set the device number in the device-number register to 9.

2. Set the channel and simultaneously disable conversion by setting the convert start bit to 0.

3. Wait for the channel multiplexer to settle (approximately 20 microseconds).

4. Request a conversion by setting the convert start bit to 1, and set the same channel.

5. Set the EOCINT enable bit to 1 to enable an end-of-conversion interrupt.

6. After servicing the interrupt, set the EOCINT enable bit to 0 to disable AI interrupts.

Analog Output Device

The analog output (AO) device has two channels, uses 12-bit DACs, and is accessed as device number 9.

External AO devices can be added to the expansion interface. These devices have up to 256 channels, use DACs with up to 16-bit resolution, and are accessed with a different device number, but use the same register format as the analog output device.

The AO device has the following registers.

Register	Read/Write	Name	Function
1	Write	AO Control	Sets up and controls the register.
3	Write	AO Data	Output value.

Following is a description of the bits of the AO control register.

Bit	Name	Function
0-7		Not used
8-15	Channel	Selects channel 0 through 255. On-board AO device uses channels 0 and 1.

Following is a description of the bits of the AO data register.

Bit	Name	Function
0-15	Data	The value to be provided to the selected channel. Sets bits 0 through 11 with data for the 12-bit DACs, and bits 12 through 15 with zeros.

AO Device Access Strategy

To use the AO device access strategy, do the following:

1. Set the device number to 9 in the device-number register.

2. Set the channel in the AO control register.

3. Write the data value to the AO data register.

Binary Input/Output Device

This on-board device, which is accessed as device number 8, is a parallel binary (TTL) I/O device. It has one 16-bit output port, one 16-bit input port, and support for handshaking lines.

Binary I/O expansion devices can be attached to the expansion interface. These devices are accessed with a different device number, but use the same register format as the binary I/O device.

The binary I/O device has the following registers.

Reg.	Read/Write	Name	Function
0	Read	Binary Status	The current state of the handshaking input lines.
0	Write	Binary Control	Controls the state of the handshaking output lines.
2	Read	Binary Input	Data register for binary input port.
2	Write	Binary Output	Data register for binary output port.

Following is a description of the bits of the binary status register.

Bit	Name	Function
0	BI Strobe	Indicates the state of the binary input port's 'BI strobe' input handshaking line.
1		Reserved
2	BO CTS	Indicates the state of the binary output port's 'clear to send' (BO CTS) input handshaking line.
3		Reserved
4-15		Not used

Following is a description of the bits of the binary control register.

Bit	Name	Function
0	BO Strobe	This bit sets and clears the binary output port's 'BO strobe' output handshaking line.
1		Reserved
2	BI CTS	This bit sets and clears the binary input port's 'clear to send' (BI CTS) output handshaking line.
3-4		Reserved
5-15		Not used

Following is a description of the bits of the binary input register.

Bit	Name	Function
0-15	Data	The current value at the binary input port can be read from this register. The value contained is not latched (unless 'BI hold' was used).

Following is a description of the bits of the binary output register.

Bit	Name	Function
0-15	Data	The value to be placed on the binary output port is written here. Data lines of the port are affected as soon as the register is written.

Binary Input Access Strategies

To use binary input with handshaking, do the following:

1. Set the device number to 8 in the device-number register.

2. Set BI CTS high in the binary control register.

   The external device puts new data on the binary input lines (BI0 through BI15) at the distribution panel connector.

   The data on the input lines must remain valid until BI CTS is lowered.

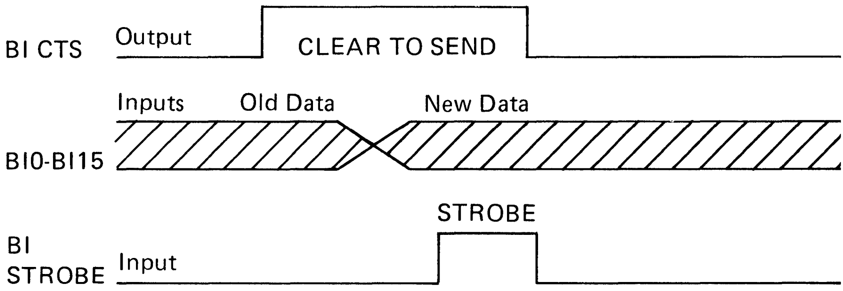
   The external device sets 'BI strobe' high.

3. Read data from the binary input register.

4. Reset BI CTS in the binary control register.

   The external device lowers 'BI strobe.'

Following is a diagram showing binary input with handshaking.



To use binary input without handshaking, do the following:

1. Set the device number to 8 in the device-number register.

2. Read the data value from the binary input register.

Binary Output Access Strategies

To use binary output with handshaking, do the following:

1. Set the device number to 8 in the device-number register.

   The external device lets BO CTS go high on the distribution panel connector. This is detected by reading the binary status register's BO CTS bit.

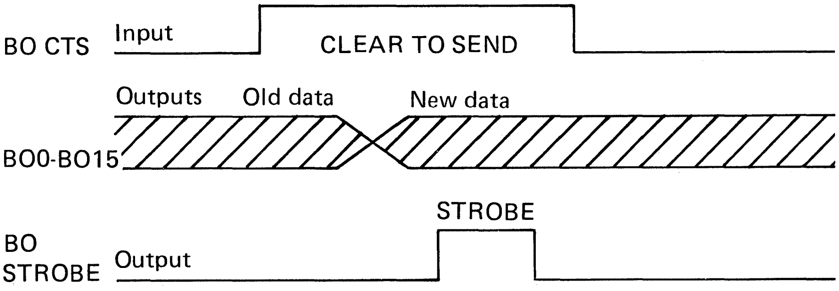
2. Write the new data value to the binary output register.

3. Set the binary control register's 'BO strobe' bit.

4. Reset the 'BO strobe' bit in the binary control register.

   The external device releases BO CTS.

The following diagram shows binary output with handshaking.



To use binary output without handshaking, do the following:

1. Set the device number to 8 in the device-number register.

2. Write the data value to the binary output register.

Timer/Counter Device Registers

This section describes the timer/counter's control register, loading of the counters, its write operations, programming format, and read operations.

The timer/counter device has the following registers.

Reg.	Read/Write	Name	Function
8	Read/Write	Timer/Counter 0 Register	Read/load Counter 0
9	Read/Write	Timer/Counter 1 Register	Read/load Counter 1
10	Read/Write	Timer/Counter 2 Register	Read/load Counter 2
11	Write	Timer/counter Control Register	Controls the operation of Counters 0 through 2.

Control Register

The timer/counter's control register controls the operating mode of each counter, selection of binary or binary coded decimal (BCD) counting, and how each counter register is loaded. The control register can only be written to; no read operation of its contents is available.

The table on the following page shows control-word information for the timer/counter's control register.

Control Word Format

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Definitions of Control

SC - Select Counter:

SC1	SC0	Description
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

RL - Read/Load:

RL1	RL0	Description
0	0	Counter latching operation
1	0	Read/load most significant byte (MSB) only.
0	1	Read/load least significant byte (LSB) only.
1	1	Read/load LSB first, then the MSB.

M - MODE:

M2	M1	M0	Description
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD - Binary Coded Decimal:

0	Binary counter 16-bits
1	BCD counter (4 decades)

Counter Loading

A count register is not loaded until the count value is written (one or two bytes, depending on the mode selected by the RL bits) and followed by a rising edge and a falling edge of the clock. Any reading of the counter before that falling clock edge may yield invalid data.

Timer/Counter Write Operations

Each counter of the timer/counter must be programmed with the mode and quantity desired. The programmer must write to the timer/counter's control register, a control word containing mode information and the programmed number of count register bytes (1 or 2) before actually using the selected counter.

Writing the control word can be in any sequence of counter selection (Counter 0 does not have to be first or Counter 2 last). Each counter's control word has a separate address so that its loading is completely independent of sequence. However, the loading of the count register with the actual count value must be done in the exact sequence programmed in the control word (RL0 and RL1). This loading, like that of the control word, is still sequence-independent, but when a selected count register is loaded, it must be done with the number of bytes programmed in the control word. The one or two bytes, loaded in the count register, do not have to immediately follow the associated control word, they can be programmed at any time after the control word is loaded, as long as the correct number of bytes is loaded in order.

All counters are *down counters*. Thus, the value loaded into the count register will actually be decreased. Loading all zeroes into a count register results in the maximum count (2 to the 16th for binary, or 10 to the 4th for BCD). In mode 0, the new count does not restart until loading is complete. The count register will accept one or two bytes, depending on how the mode control words (RL0 and RL1) are programmed. Then the restart operation proceeds.

Timer/Counter Programming Format

The programming format shown below is a simple example of timer/counter loading and does not imply that it is the only format that can be used.

Programming Format:

MODE	Control Word Counter n
LSB	Count Register Byte Counter n
MSB	Count Register Byte Counter n

Alternate Programming Format:

Number	Byte	Description	Reg.
1	MODE	Control Word Counter 0	8
2	MODE	Control Word Counter 1	8
3	MODE	Control Word Counter 2	8
4	LSB	Count Register Byte Counter 1	9
5	MSB	Count Register Byte Counter 1	9
6	LSB	Count Register Byte Counter 2	10
7	MSB	Count Register Byte Counter 2	10
8	LSB	Count Register Byte Counter 0	11
9	MSB	Count Register Byte Counter 0	11

Timer/Counter Read Operations

In most counter applications, it becomes necessary to read the value of the count in progress and make a computational decision based on this quantity. Event counters are probably the most common applications that use this function. The timer/counter has logic that allows the programmer to easily read the contents of any of the three counters without disturbing the actual count in progress.

The programmer can use two methods to read the value of the counters. The first method involves the use of simple I/O read operations of the selected counter. The only requirement of this method is that the actual operation of the selected counter must be inhibited by external logic that inhibits the clock input (only valid for Counter 2, because its 'clock in' signal is brought to the distribution panel connector). This requirement ensures a stable count reading.

The contents of the selected counter are as follows:

- The first I/O Read contains the least-significant byte.
- The second I/O Read contains the most-significant byte.

Because of the timer/counter's internal logic, the entire reading procedure must be finished. If two bytes are programmed to be read, then the two bytes must be read before any loading-write (WR) commands can be sent to the same counter.

The following chart has information about the read operation.

Reg.	Description
8	Read Counter 0
9	Read Counter 1
10	Read Counter 2

The second method of reading the value of the counters involves reading while counting.

To allow the programmer to read the contents of any counter without affecting the counting operation, special internal logic of the timer/counter can be accessed with simple write commands to the mode register. When the programmer wishes to read the contents of a selected counter, he loads the mode register with a special code that latches the present count value into a storage register so that its contents have an accurate, stable quantity. The programmer then issues a normal Read command to the selected counter, and the contents of the latched register are available.

The second method of reading the counters has the same limitation as the first method described. That is, the entire read operation must be finished as programmed. The Read command has no effect on the counter's mode.

The following table shows the control word used for latching count. This control word is written to the timer/counter's control register.

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

Note:

SC1, SC0 - specify counter to be latched
D5, D4 - 00 designates counter latching operation
X - don't care

Device Number Register

The device-number register selects the device to be accessed. Only the low-order byte should be written.

The on-board devices are:

- Analog I/O device (device number 9)

- Binary I/O device (device number 8)

The following table describes the bits of the device-number register.

Bit	Name	Function
0-7	Device Number	Selects which device to address (0-255)
8-15		Not used

Interrupt Registers

Two registers are decoded for register 13, the interrupt control register and the interrupt status register. The interrupt control register may be written to for setting up the Data Acquisition Adapter for various interrupts. The interrupt status register may be read for information about the Data Acquisition Adapter's current interrupt status.

The following is a description of the bits of the interrupt control register.

Bit	Name	Function
0	TINT Enable	Enables the timer-generated interrupts.
1	CINT Enable	Enables counter-generated interrupts.
2	IRQ Enable	Enables the external IRQ line to generate an interrupt. It also enables AI end-of-conversion interrupts. To enable EOC interrupt, both this bit and the EOCINT enable bit in the AI control register must be set.
3		Reserved. Must be cleared.
4-6		Not used. Must be cleared.
7	LINT Reset	Local Interrupt Reset; re-enables the interrupt circuitry of the addressed Data Acquisition Adapter.

The following is a description of the bits of the interrupt status register.

Note: The bits of the interrupt status register contain the current state of the devices that generate an interrupt. They are not latched, and they are not reset by reading them.

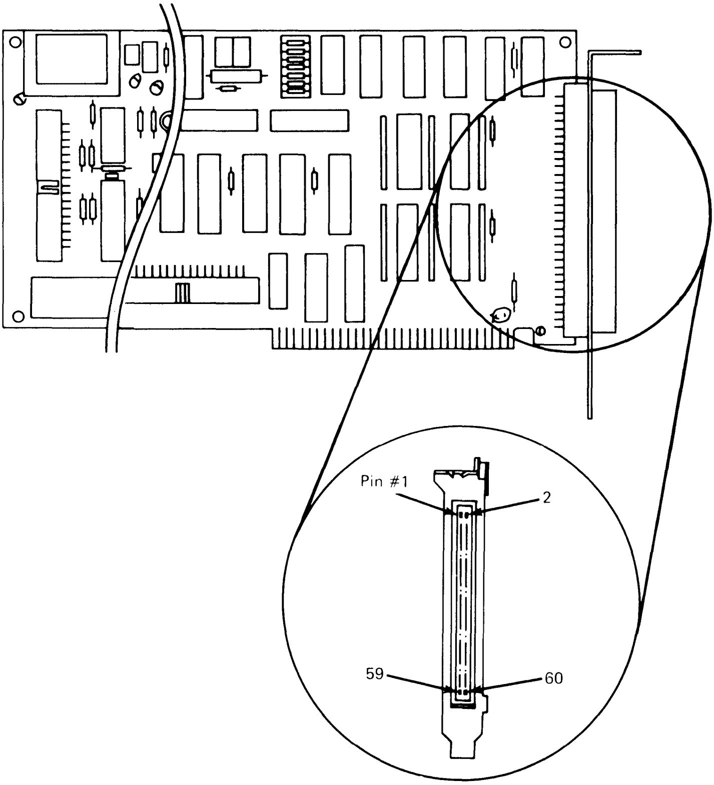
Bit	Name	Function
0 - 3	TINT Enable CINT Enable IRQ Enable	Read back of the current state of bits 0 through 3 in the interrupt control register.
4	TINT STAT	Timer Interrupt Status; if this bit is set, a timer-generated interrupt has occurred.
5	CINT STAT	Counter Interrupt Status; if this bit is set, a counter-generated interrupt has occurred.
6	IRQ STAT	IRQ Status; if this bit is set, the $\overline{\text{IRQ}}$ line generated an interrupt.
7		Reserved

Interface

Following is information about the Data Acquisition Adapter's connectors, J4 (distribution panel), and J1 and J2 (expansion bus).

Distribution Panel Connector

The following shows the location of the distribution panel connector (J4).



The following shows the pins of the distribution panel connector and their respective signals.

	Signal Name/Description	Pin	
External Device	D/A 1	1	Data Acquisition Adapter
	D/A 0	2	
	+10V REF	3	
	A GND	4	
	A/D 0-	5	
	A/D 0+	6	
	A/D 1-	7	
	A/D 1+	8	
	A/D 2-	9	
	A/D 2+	10	
	A/D 3-	11	
	A/D 3+	12	
	A GND	13	
	A/D CE	14	
	D GND	15	
	A/D CO	16	
	BI 8	17	
	BO 8	18	
	BI 9	19	
	BO 9	20	
	BI 10	21	
	BO 10	22	
	BI 11	23	
	BO 11	24	
	BI 12	25	
	BO 12	26	
	BI 13	27	
	BO 13	28	
	BI 14	29	
	BO 14	30	

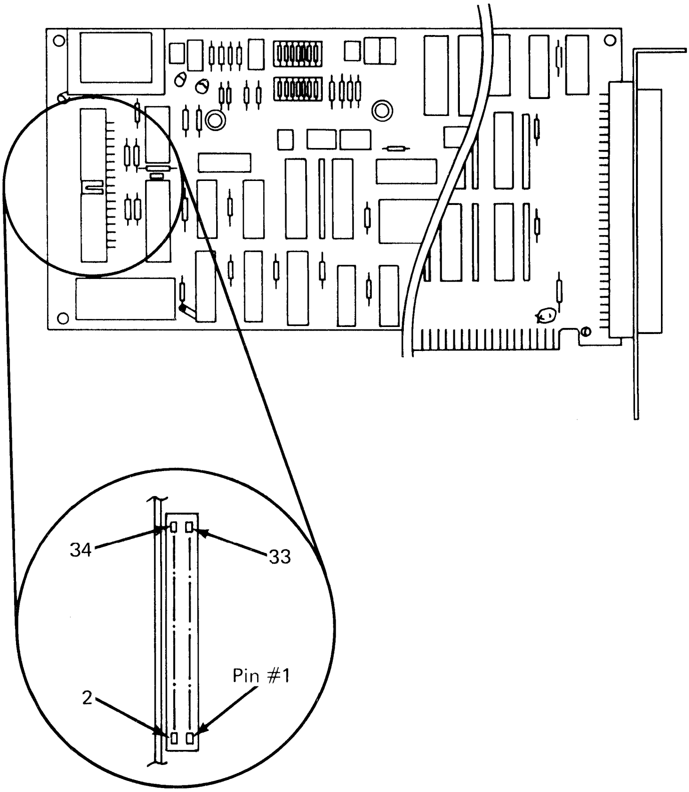
	Signal Name/Description	Pin	
	BI 15	31	
	BO 15	32	
	$\overline{\text{BI HOLD}}$	33	
	BO GATE	34	
	BI 0	35	
	BO 0	36	
	BI 1	37	
	BO 1	38	
	BI 2	39	
	BO 2	40	
	BI 3	41	**Data**
External	BO 3	42	**Acquisition**
Device	BI 4	43	**Adapter**
	BO 4	44	
	BI 5	45	
	BO 5	46	
	BI 6	47	
	BO 6	48	
	BI 7	49	
	BO 7	50	
	RATE OUT	51	
	DELAY OUT	52	
	BI STROBE	53	
	BO STROBE	54	
	BO CTS	55	
	BI CTS	56	
	$\overline{\text{IRQ}}$	57	
	COUNT OUT	58	
	COUNT IN	59	
	D GND	60	

Expansion Bus Connectors

Following is information about the expansion bus connectors (J1 and J2).

Expansion Bus Connector J1

The following shows the location of the expansion bus connector, J1.

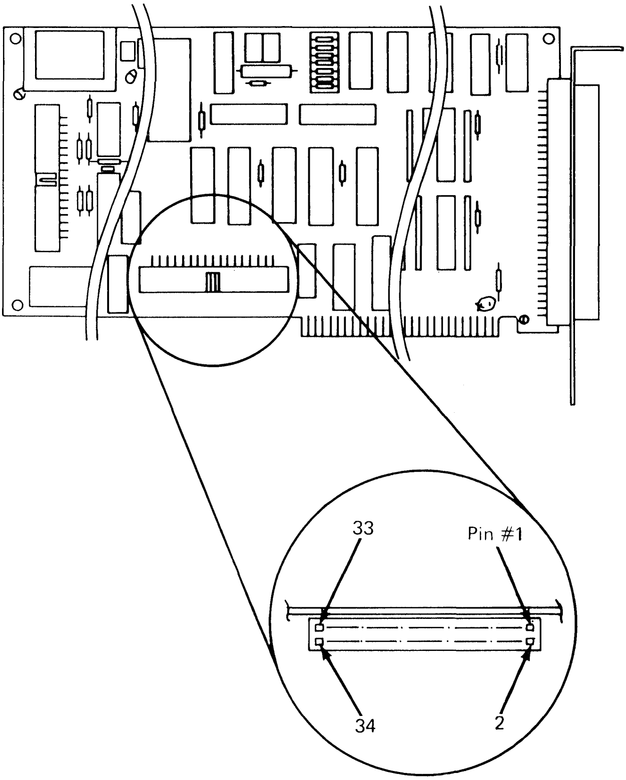


The following shows the pins of the expansion bus connector, J1, and their respective signals.

	Signal Name/Description	Pin	
External Device	BUFF 0o	1	Data Acquisition Adapter
	Reserved	2	
	$\overline{\text{WRITEREADGATE}}$	3	
	BUFFREAD	4	
	$\overline{\text{BUFFRES}}$	5	
	$\overline{\text{BUFFREAD}}$	6	
		7	
	$\overline{\text{IRQ}}$	8	
	Reserved	9	
	Reserved	10	
	D GND	11	
	D GND	12	
	Reserved	13	
	Reserved	14	
	D GND	15	
	D GND	16	
	Reserved	17	
	Reserved	18	
	$\overline{\text{AS7}}$	19	
	$\overline{\text{AS6}}$	20	
	$\overline{\text{AS5}}$	21	
	$\overline{\text{AS4}}$	22	
	$\overline{\text{AS3}}$	23	
	$\overline{\text{AS2}}$	24	
	$\overline{\text{AS1}}$	25	
	$\overline{\text{AS0}}$	26	
	$\overline{\text{AS15}}$	27	
	$\overline{\text{AS14}}$	28	
	$\overline{\text{AS13}}$	29	
	$\overline{\text{AS12}}$	30	
	$\overline{\text{AS11}}$	31	
	$\overline{\text{AS10}}$	32	
	$\overline{\text{AS9}}$	33	
	$\overline{\text{AS8}}$	34	

Expansion Bus Connector J2

The following shows the location of the expansion bus connector, J2.



The following shows the pins of the expansion bus connector, J2, and their respective signals.

External Device

Data Acquisition Adapter

Signal Name/Description	Pin
$\overline{\text{WD 3}}$	1
D GND	2
HA 7 D GND	3
HA 6 D GND	4
HA 5 D GND	5
HA 4 D GND	6
DV 7	7
DV 6	8
DV 5	9
DV 4	10
LD 7	11
LD 6	12
LD 5	13
LD 4	14
LD 3	15
LD 2	16
LD 1	17
LD 0	18
HD 7	19
HD 6	20
HD 5	21
HD 4	22
HD 3	23
HD 2	24
HD 1	25
HD 0	26
DV 3	27
DV 2	28
DV 1	29
DV 0	30
WD 3 D GND	31
WD 2	32
WD 1	33
WD 0	34

Notes:

Switch Settings

The Data Acquisition Adapter has five groups of slide-type DIP switches that control analog output range, analog input range, adapter number, and interrupt level. Each group of switches is labeled on the adapter, and each switch is numbered on the housing. The switch positions (On and Off) also are labeled on the housing.

Analog Output Range

Following is a description of the switch blocks that control the Data Acquisition Adapter's analog output range for channels 0 and 1.

Channel 0

Switch block S1 has two switches: S1-1 and S1-2. These switches determine the relationship between analog output values and the voltage output of the analog output device.

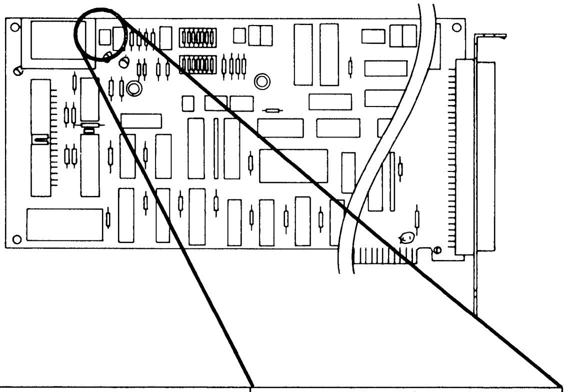
S1-1 controls voltage range:

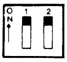
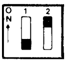
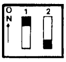
- On: 10-volt range
- Off: 20-volt range

S1-2 controls voltage polarity:

- On: Bipolar (±) voltage
- Off: Unipolar (+) voltage

The following diagram shows the location and switch settings for switch block S1.



Input Range	Switch Settings
-5 to +5 volts	
-10 to +10 volts	
0 to +10 volts	

Note: Only the settings shown may be used for this switch block.

Channel 1

Switch block S2 has two switches: S2-1 and S2-2. These switches determine the relationship between analog output values and the voltage output of the analog output device.

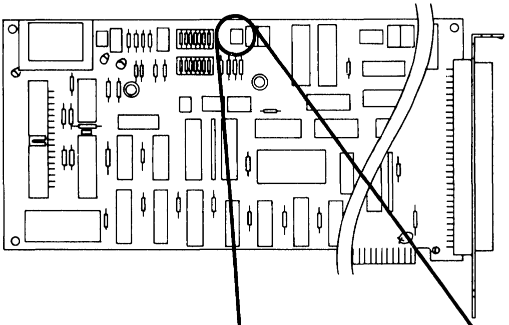
S2-1 controls voltage range:

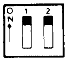
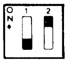
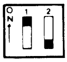
- On: 10-volt range
- Off: 20-volt range

S2-2 controls voltage polarity:

- On: Bipolar (±) voltage
- Off: Unipolar (+) voltage

The following diagram shows the location and switch settings for switch block S2.



Output Range	Switch Settings
-5 to +5 volts	
-10 to +10 volts	
0 to +10 volts	

Note: Only the settings shown may be used for this switch block.

Analog Input Range

Switch block S3 has four switches: S3-1, S3-2, S3-3, and S3-4. The settings of these switches determine the relationship of analog input voltage to the values returned by the analog input device.

S3-1 is not used and is placed in the Off position.

S3-2 controls the 20-volt input range:

- On: 20-volt range active
- Off: 20-volt range inactive

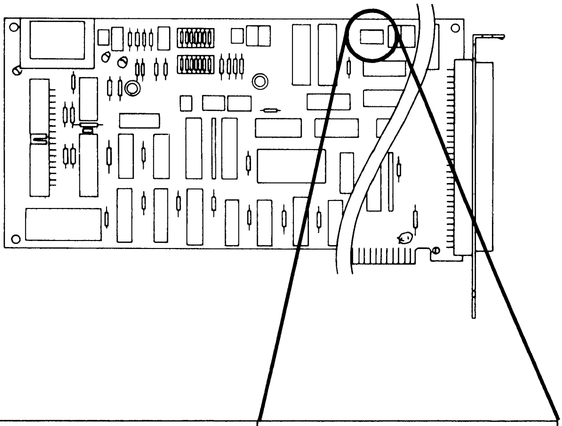
S3-3 controls the 10-volt input range.

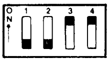
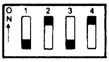
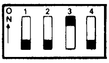
- On: 10-volt range active
- Off: 10-volt range inactive

S3-4 controls voltage polarity.

- On: Bipolar (±) voltage
- Off: Unipolar (+) voltage

The following diagram shows the location and switch settings for switch block S3.



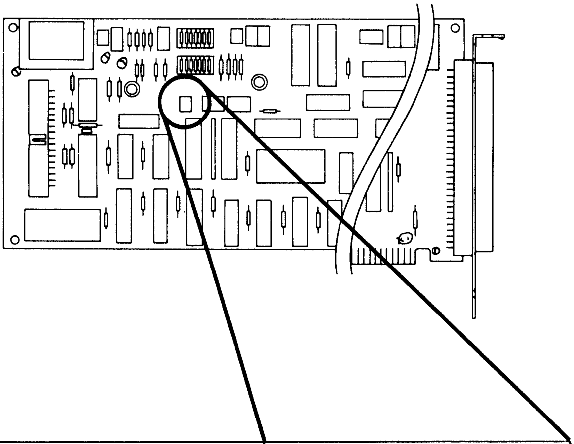
Output Range	Switch Settings
-5 to +5 volts	
-10 to +10 volts	
0 to +10 volts	

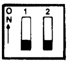
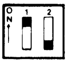
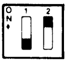
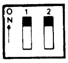
Note: Only the settings shown may be used for this switch block.

Adapter Number

Switch block S4 has two switches: S4-1 and S4-2. These switches specify the adapter number (0 through 3). Assign a number to each Data Acquisition Adapter before installation.

Note: Up to four Data Acquisition Adapters may be installed in your unit. Each adapter must be given a different adapter number.



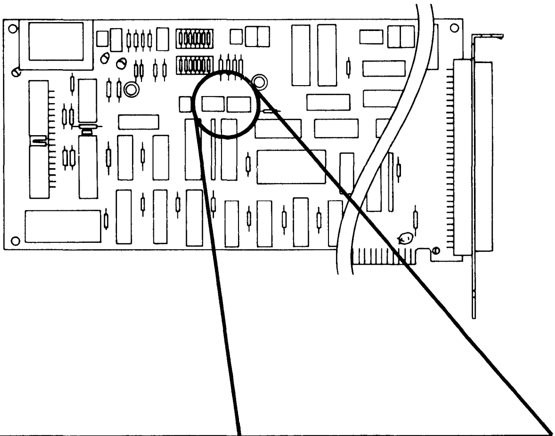
Adapter Number	Switch Positions
0	
1	
2	
3	

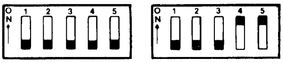
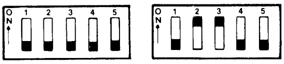
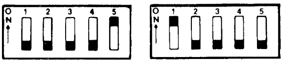
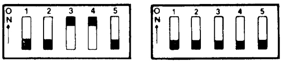
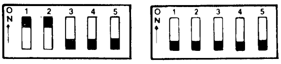
Interrupt Level

Switch block S5 has two 5-switch switch blocks rather than one 10-switch switch block. The switches of the right-hand, 5-switch, switch block, although numbered 1 through 5, are functionally identical to switches 6 through 10.

These 10 switches determine the interrupt level of each Data Acquisition Adapter. Set the interrupt level for each adapter before installation. Data Acquisition Adapters installed in the same unit must be set to the same interrupt level. The setting for interrupt request level 7 (IRQ7) is recommended.

The following diagram shows the location and switch settings for switch block S5.



IRQ Level	Switch Settings
7 (Recommended)	
6	
5	
4	
3	

Note: Only the settings shown may be used for this switch block.

Specifications

The following is a description of the Data Acquisition Adapter specifications and device characteristics.

Data Acquisition Adapter

Following is a description of the Data Acquisition Adapter specifications.

Dimensions

Height	99.1 mm (3.9 in.)
Height at Tab Pins	106.7 mm (4.2 in.)
Length	335.3 mm (13.2 in.)
Thickness	14.2 mm (0.56 in.)
Weight	270 g (9.5 oz)

Power Requirements

+5 volts ±5% at approximately 1 ampere typical (1.5 ampere maximum)

System Reference Voltage

Output Voltage	+10 volts
Accuracy	±1.2%
Output Load Current	±2 milliamperes maximum
Output Load Capacitance	0.5 microfarads maximum for stability
Output Protection	Protected for short to common
Output Impedance	2-ohms maximum at distribution panel connector

Environment

The Data Acquisition Adapter complies with the limits for a Class B computing device according to Subpart J of Part 13 of FCC rules and meets German VED requirements when installed in the host system.

Operating Environment

Temperature Range	+5 to +46°C (+41 to +114.8°F)
Humidity Range	8% to 80% non-condensing
Altitude	2187 m (7000 ft) maximum

Non-Operating Environment

Temperature Range	-4 to +60°C (-40 to +140°F)
Humidity Range	5% to 100% non-condensing

Data Acquisition Adapter Devices

Following is a description of the characteristics of the devices on the Data Acquisition Adapter.

Analog Output Device

The analog output device has the following characteristics:

Resolution	12 bits
Output Channels	2
Output Ranges	Switch-selectable ranges: 0 to +10 volts (unipolar), -5 to +5 volts (bipolar), and -10 to +10 volts (bipolar).
Output Load Current	±5 milliamperes minimum
Output Load Capacitance for Stability	0.5 microfarads maximum
Digital Coding	Unipolar: binary. Bipolar: offset binary.
Integral Linearity Error	± 1 least significant bit (LSB) maximum
Impedance	2-ohms maximum at the distribution panel connector
Protection	Protected for short to common

Differential Linearity Error	±1/2 LSB maximum; guaranteed monotonic
Gain Error	±0.1% maximum between ranges. Any range adjustable to zero.
Gain Stability	±35 ppm/°C of full scale range (FSR) maximum
Unipolar Offset Error	±3.25 millivolts maximum
Unipolar Offset Stability	±8ppm/°C of FSR maximum
Bipolar Offset Error	Adjustable to zero
Bipolar Offset Stability	±24 ppm/°C of FSR maximum
Power Supply Rejection	±1/2 LSB maximum change in full scale calibration
Throughput from Memory	25,000 conversions per second, minimum

Dynamic characteristics for a -10 volt to +10 volt step with less than ±5 milliamperes and less than 1000 picofarads load are:

Overshoot	±1% of FSR maximum
Settling Time	10 microseconds maximum to within ±0.1% FSR

Analog Input Device

The analog input device has the following characteristics:

Resolution	12 bits
Input Channels	4 differential
Input Ranges	Switch-selectable ranges: 0 to +10 volts (unipolar), -5 to +5 volts (bipolar), and -10 to +10 volts (bipolar).
Input Resistance	100 megohms minimum
Input Capacitance	200 picofarads maximum; measured at the distribution panel connector
Input Leakage Current	±300 nanoamperes maximum
Input Current	±4 milliamperes at maximum input voltage
Digital Coding	Unipolar: binary. Bipolar: offset binary.
Safe Input Voltage	±30 volts maximum (power On or Off)
Power Supply Rejection	±1/2 LSB maximum change full scale calibration
Integral Linearity Error	±1 LSB maximum

Differential Linearity Error	±1/2 LSB maximum
Differential Linearity Stability	±5 ppm/°C maximum; guaranteed monotonic
Gain Error	±0.1% maximum between ranges. Any range adjustable to zero.
Gain Stability	±32 ppm/°C of FSR maximum
Common-Mode Input Range	±11 volts maximum
Common-Mode Rejection	72 dB minimum ratio (signal within common-mode range)
Unipolar Offset Error	Adjustable to zero
Unipolar Offset Stability	±24 ppm/°C of FSR maximum
Bipolar Offset Error	Adjustable to zero
Bipolar Offset Stability	±24 ppm/°C of FSR maximum

Settling Time	For channel acquisition: 20 microseconds maximum to ±0.1% of the input value
Conversion Time	35 microseconds maximum
Throughput to Memory	15,000 conversions per second, minimum
'A/D convert enable'	
Input Impedance	One LS TTL load plus 10-kilohm pull-up resistor
'A/D convert out'	
Fanout	10 LS TTL loads or 2 standard TTL loads

Binary Device

The binary device has the following characteristics:

Binary Input (BI0 through BI15)

Input Impedance	One LS TTL load plus 10-kilohm pull-up resistor
Throughput to memory	25,000 operations per second, minimum

$\overline{\textbf{BI HOLD}}$

Input Impedance	Two LS TTL loads plus one 10-kilohm pull-up resistor

‘BI Strobe’

Input Impedance	One LS TTL load plus one 10-kilohm pull-up resistor

BI CTS

Fanout	10 LS TTL loads or 2 standard TTL loads

Binary Output (BO0 through BO15)

Fanout	28 LS TTL loads or 7 standard TTL loads
Throughput from Memory	25,000 operations per second, minimum

‘BO Gate’

Input Impedance	Two LS TTL loads plus one 10-kilohm pull-up resistor

BO CTS

Input Impedance	One LS TTL load plus 10-kilohm pull-up resistor

‘BO Strobe’

Fanout	10 LS TTL loads or 2 standard TTL loads

32-Bit Timer Device

The 32-bit timer device has the following characteristics:

Counter 0	
CLK 0 Frequency	1.023 MHz
‘Rate Out’	
Fanout	10 LS TTL loads or 2 standard TTL loads
Counter 1	
‘Delay Out’	
Fanout	10 LS TTL loads or 2 standard TTL loads

16-Bit Timer/Counter Device

The 16-bit timer/counter device has the following characteristics:

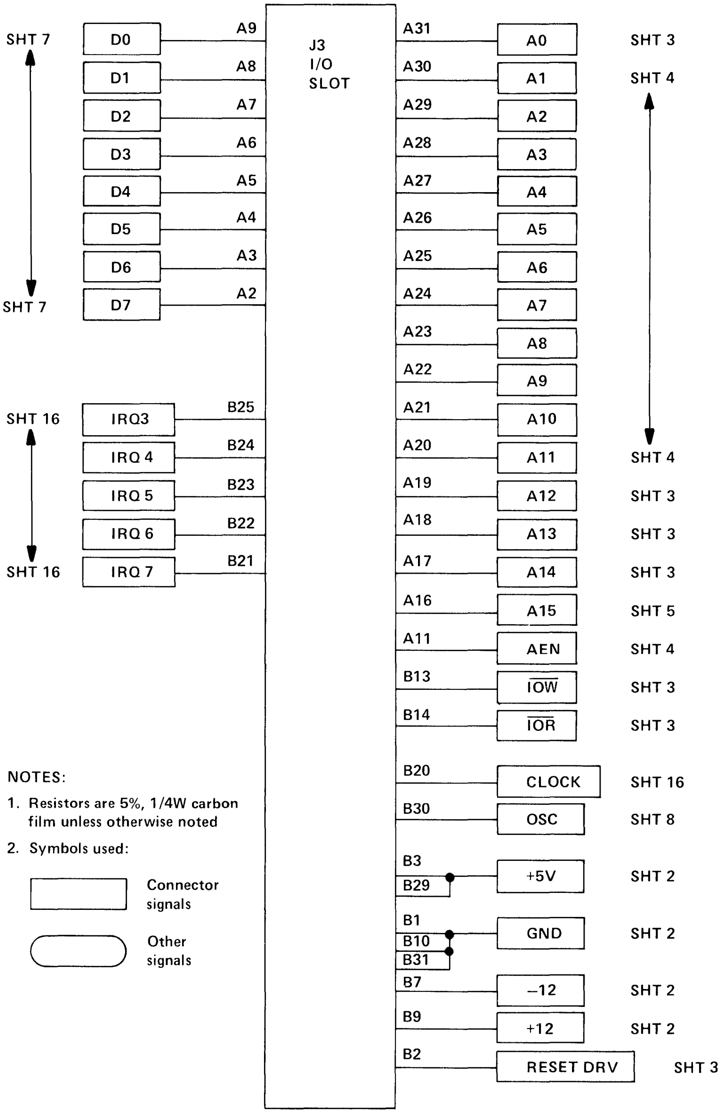
'Count In'

Input Impedance	One LS TTL load plus 10-kilohm pull-up resistor
Input Frequency	DC - 2 MHz (50% duty cycle)

'Count Out'

Fanout	10 LS TTL loads or 2 standard TTL loads

Logic Diagrams



NOTES:

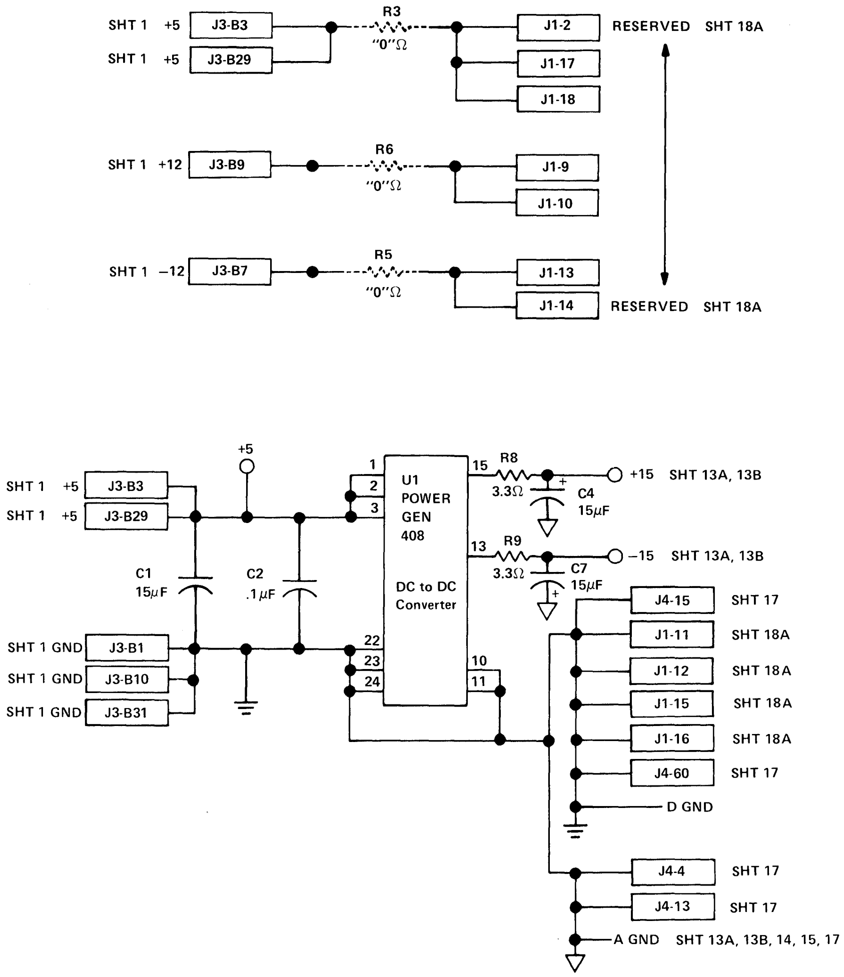
1. Resistors are 5%, 1/4W carbon film unless otherwise noted
2. Symbols used:

Connector signals

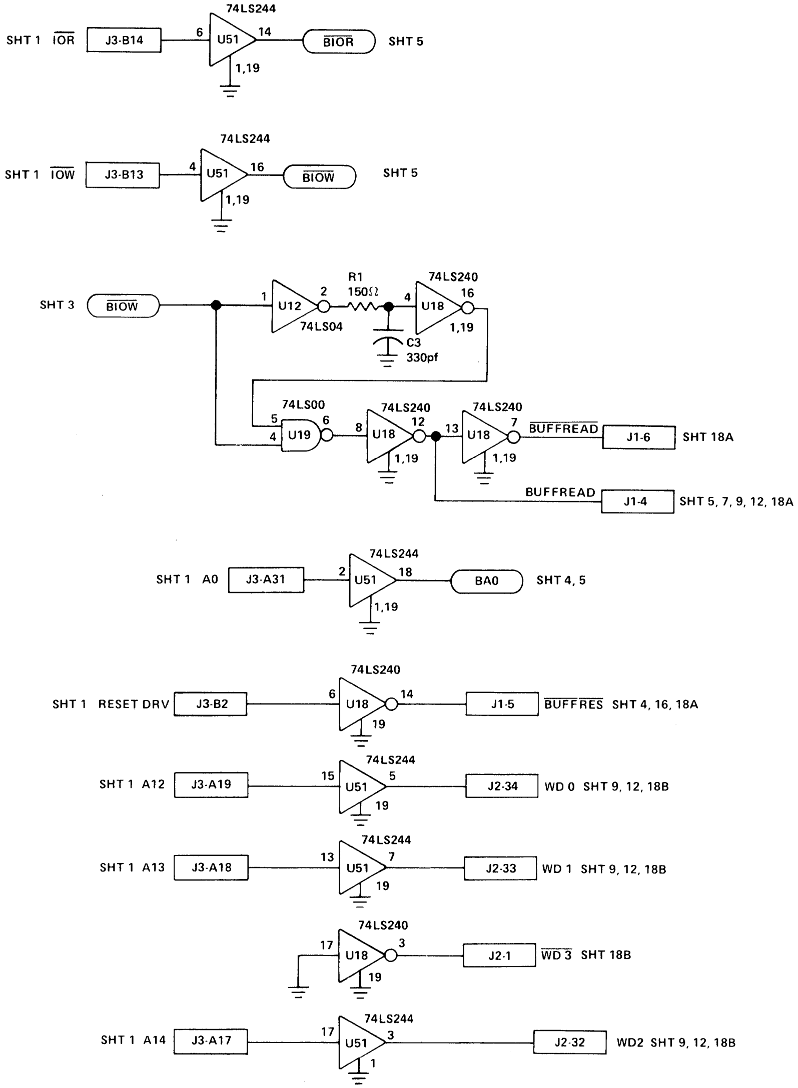
Other signals

I/O Slot J3 **Sheet 1**

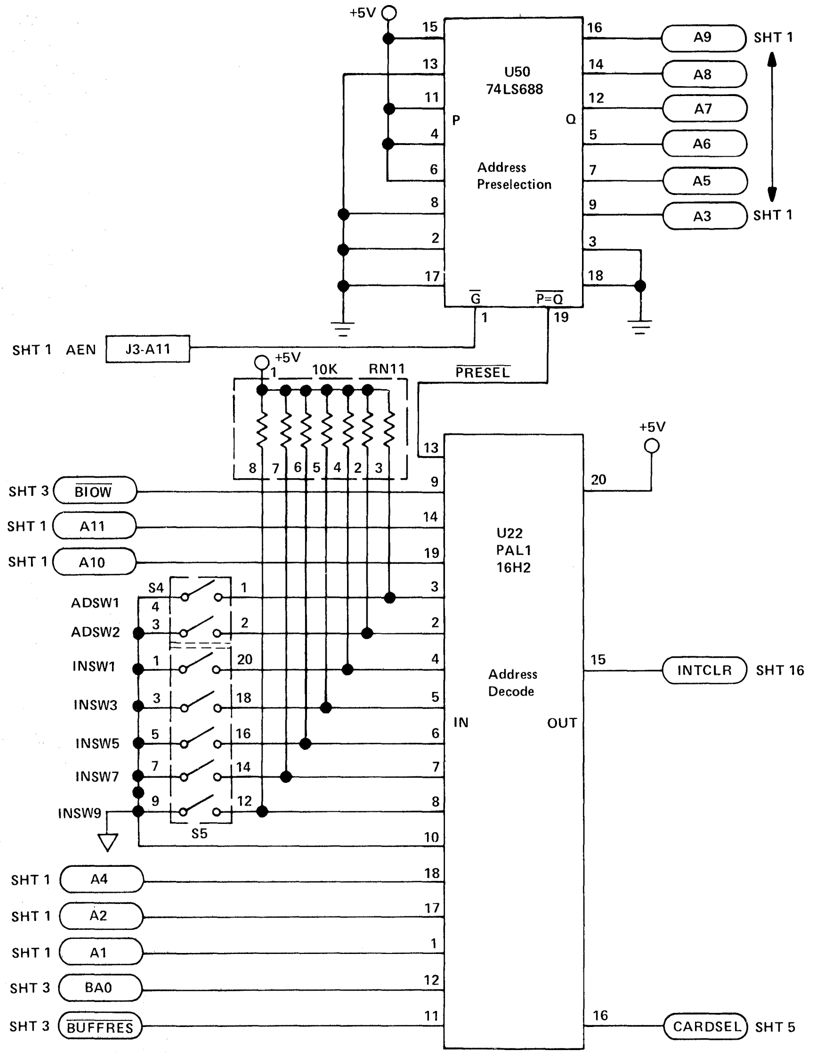
NOTE: R3, R5, R6 ARE NOT INSTALLED.



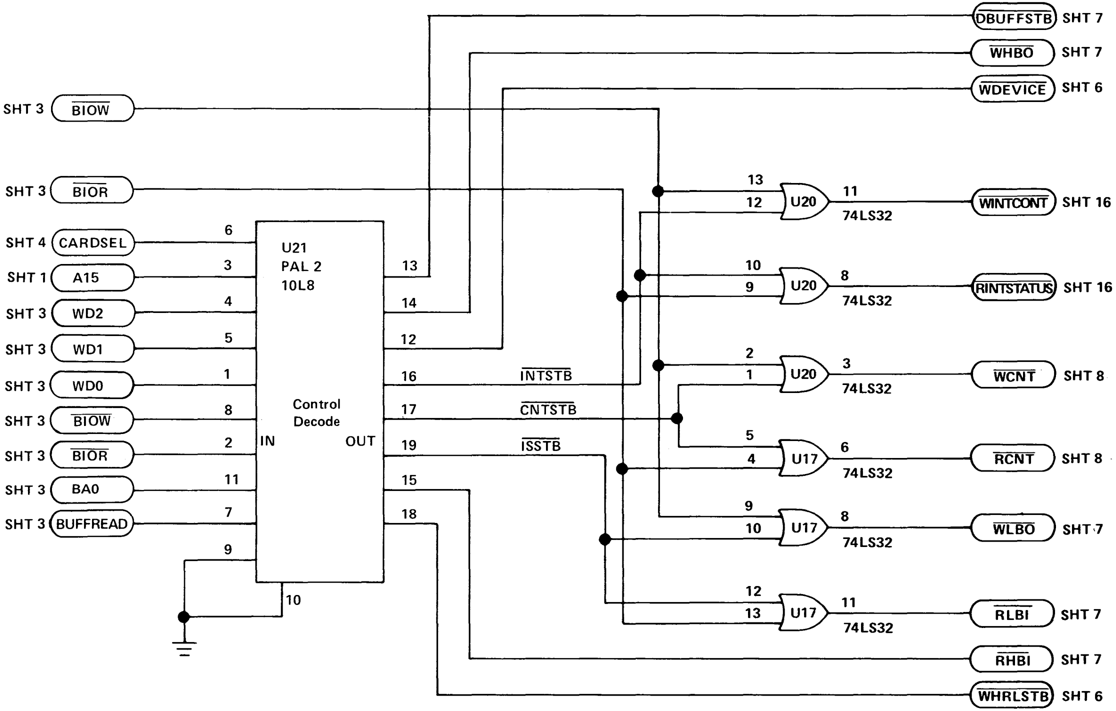
Power Supplies and Grounds **Sheet 2**



System Bus Control and Address Lines **Sheet 3**

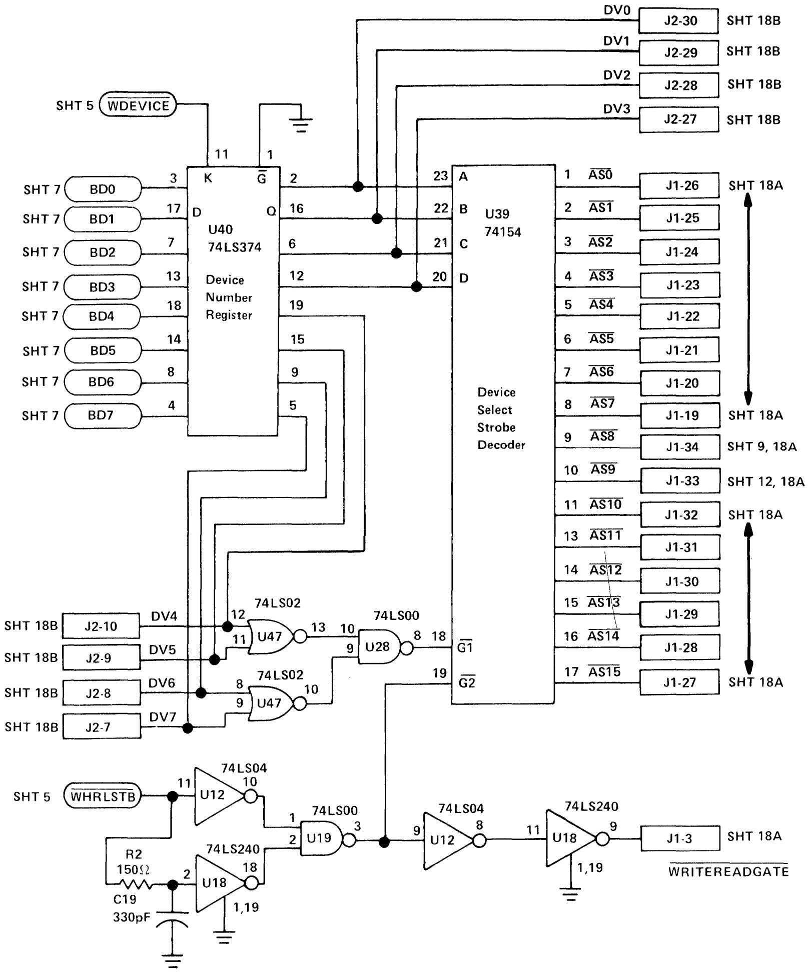


Address Decode **Sheet 4**

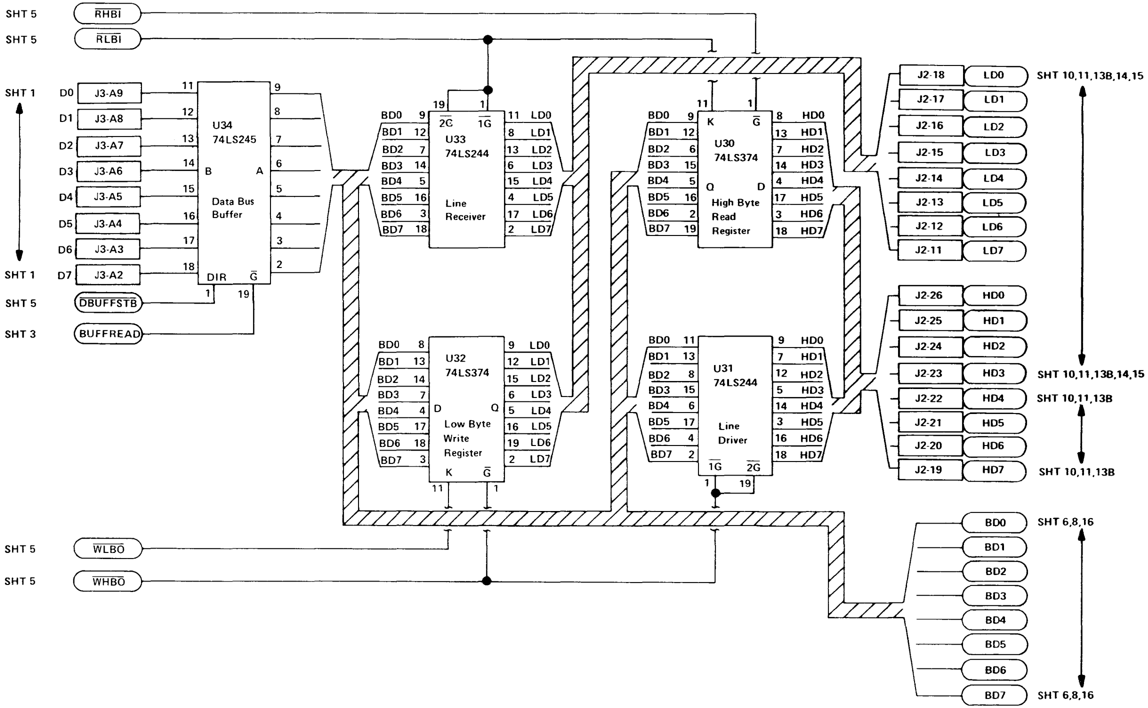


Control Decode

Sheet 5

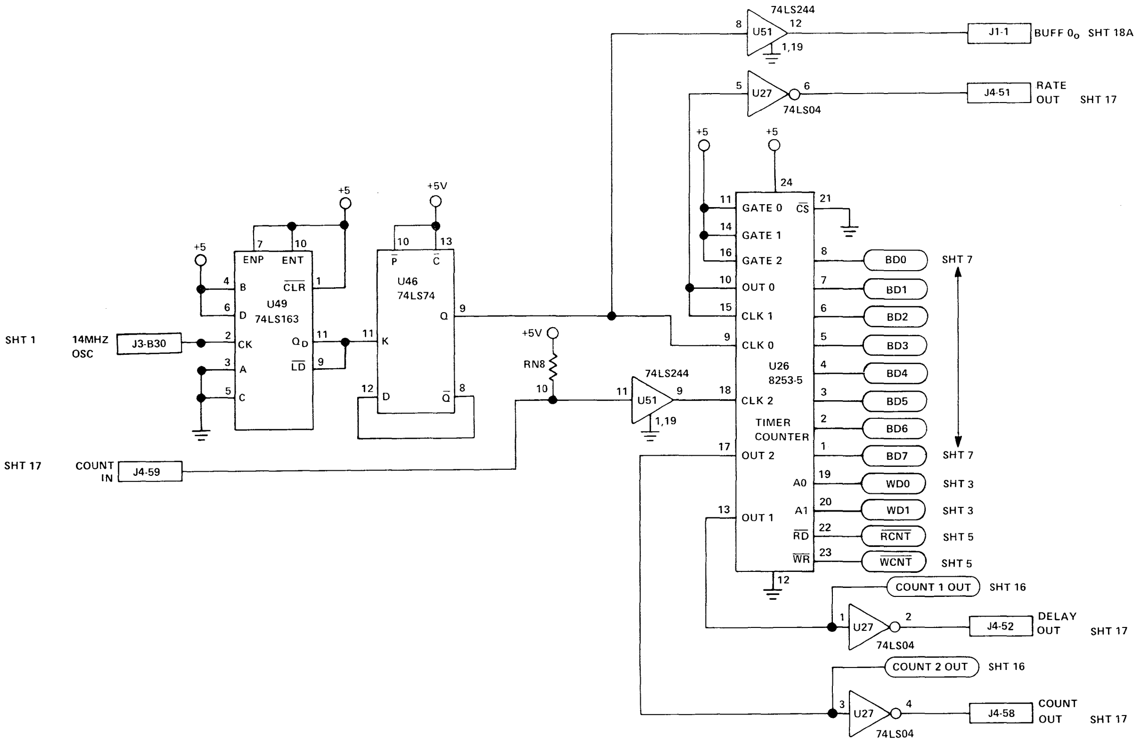


Device Selection **Sheet 6**



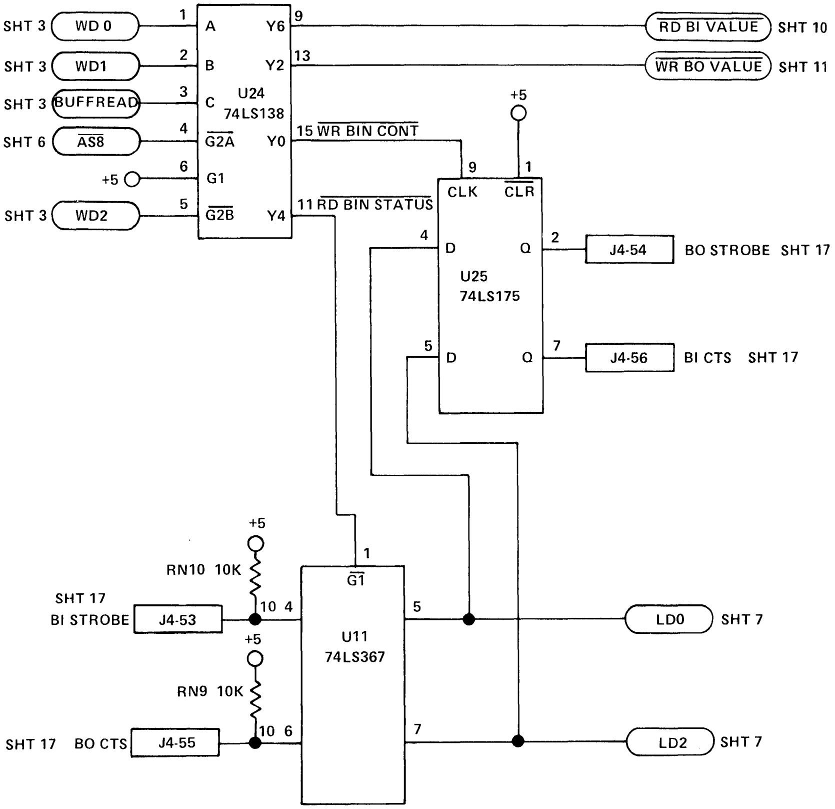
Data Bus Conversion

Sheet 7

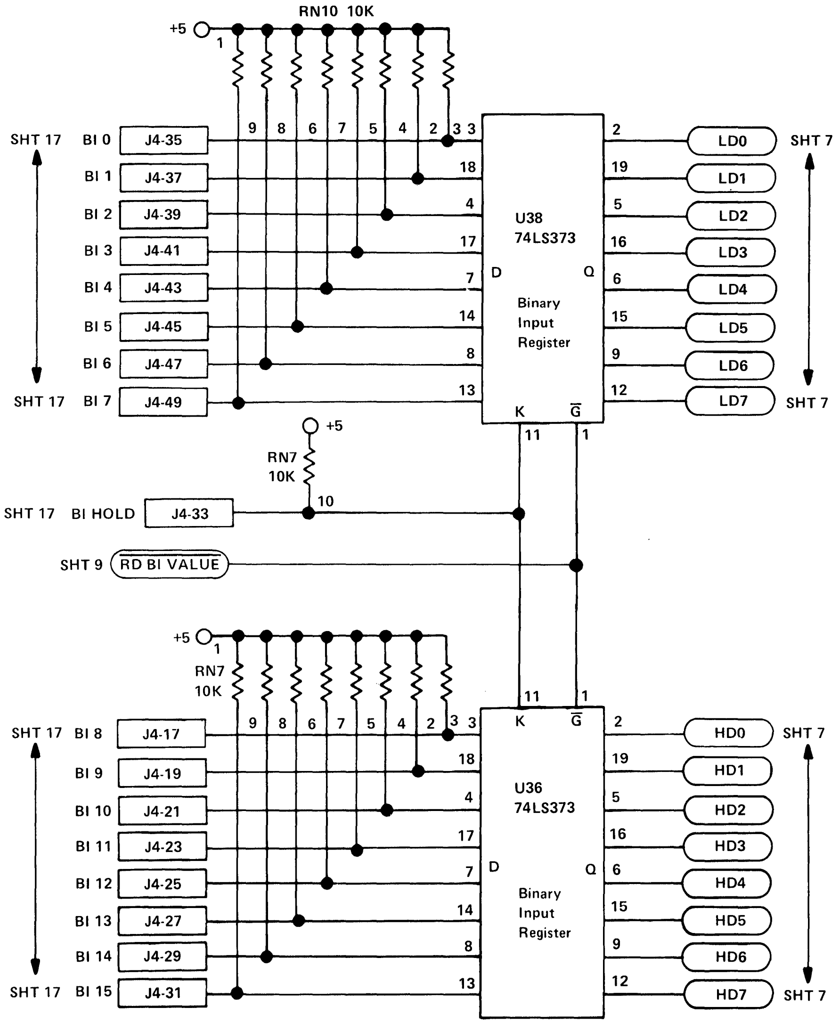


Timer/Counter Device

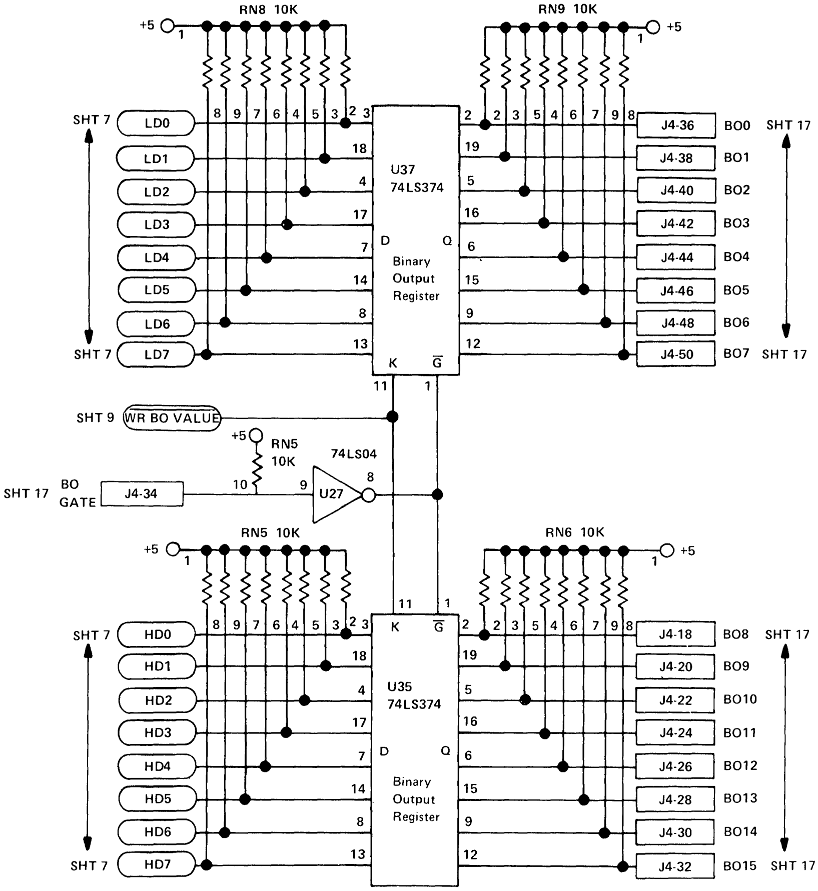
Sheet 8



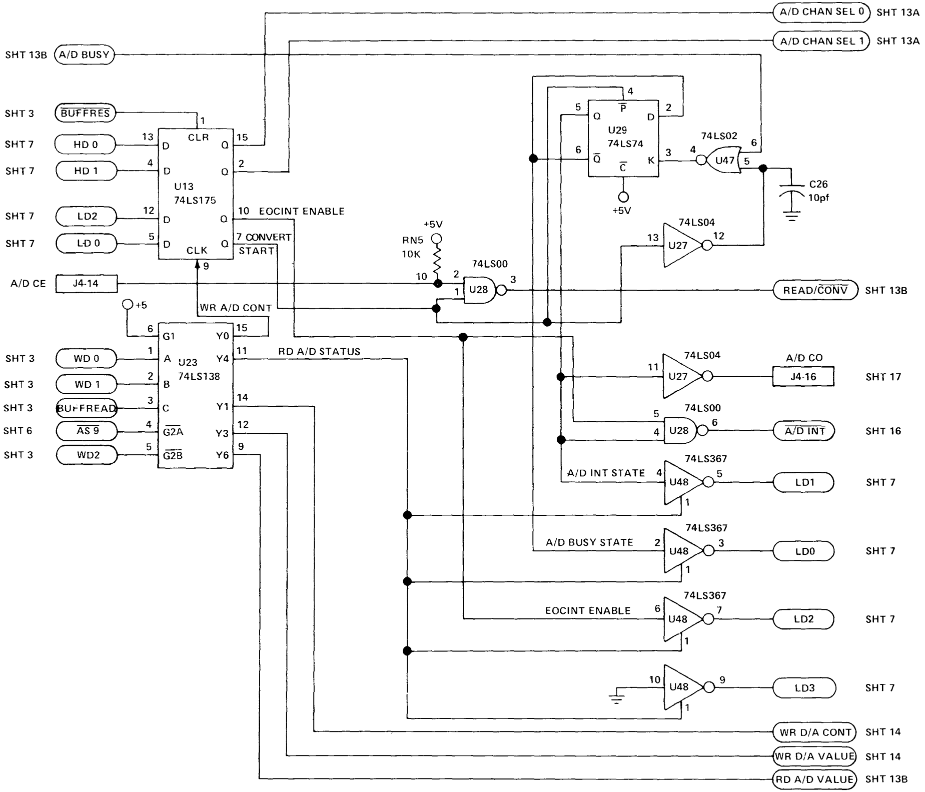
Binary I/O Device Control Decode Sheet 9



Binary Input Device **Sheet 10**

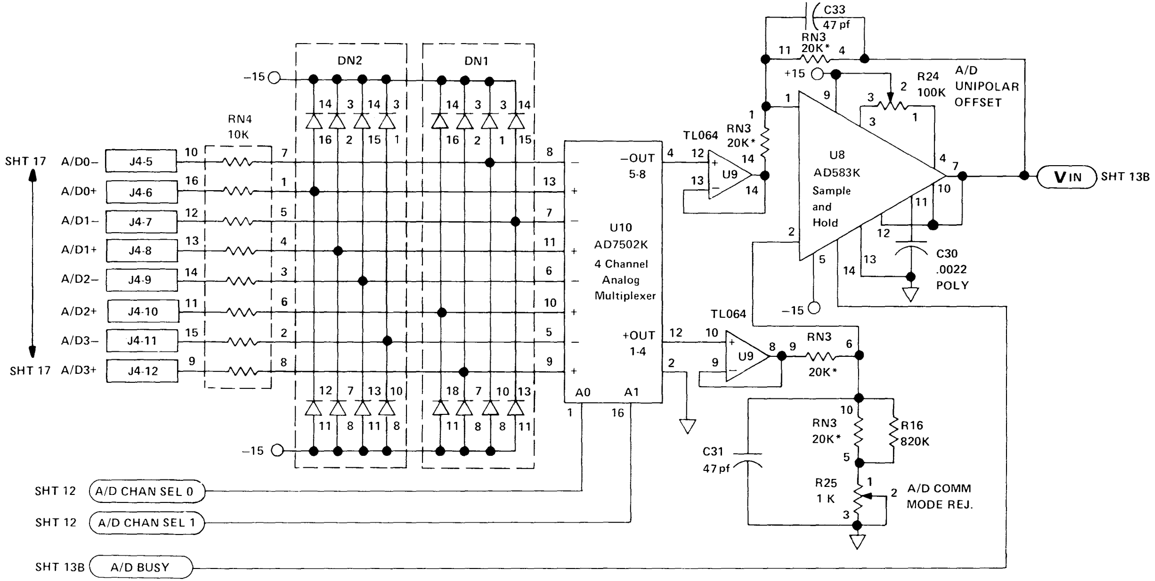


Binary Output Device **Sheet 11**



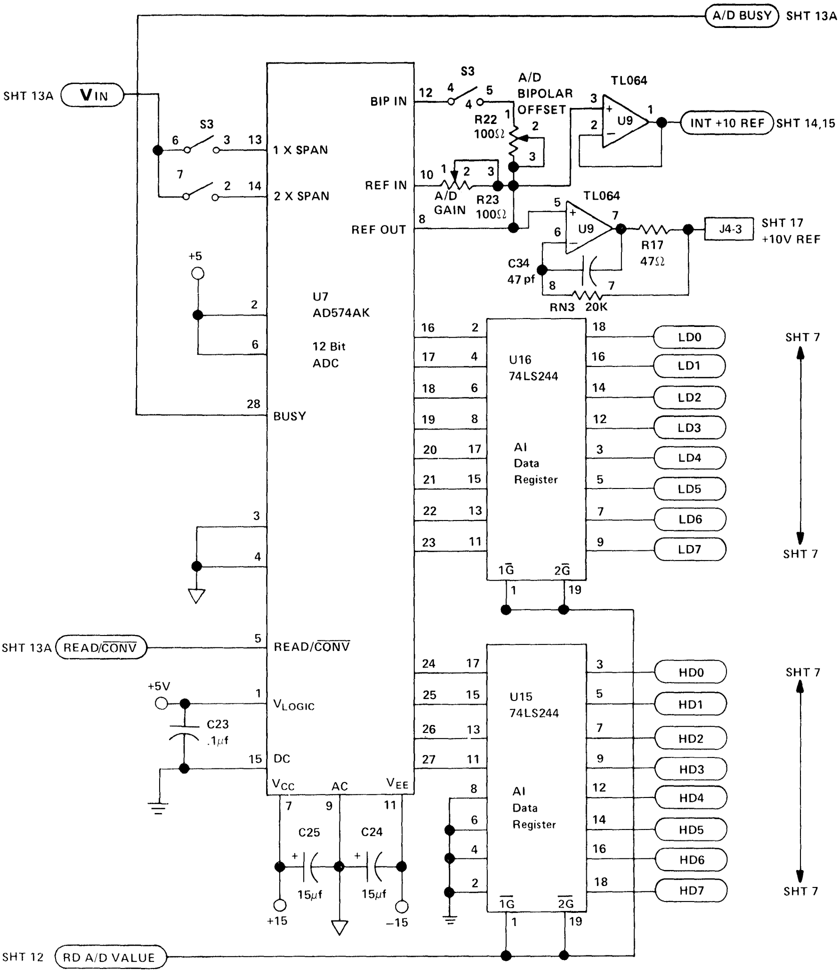
Analog I/O Device Control

Sheet 12

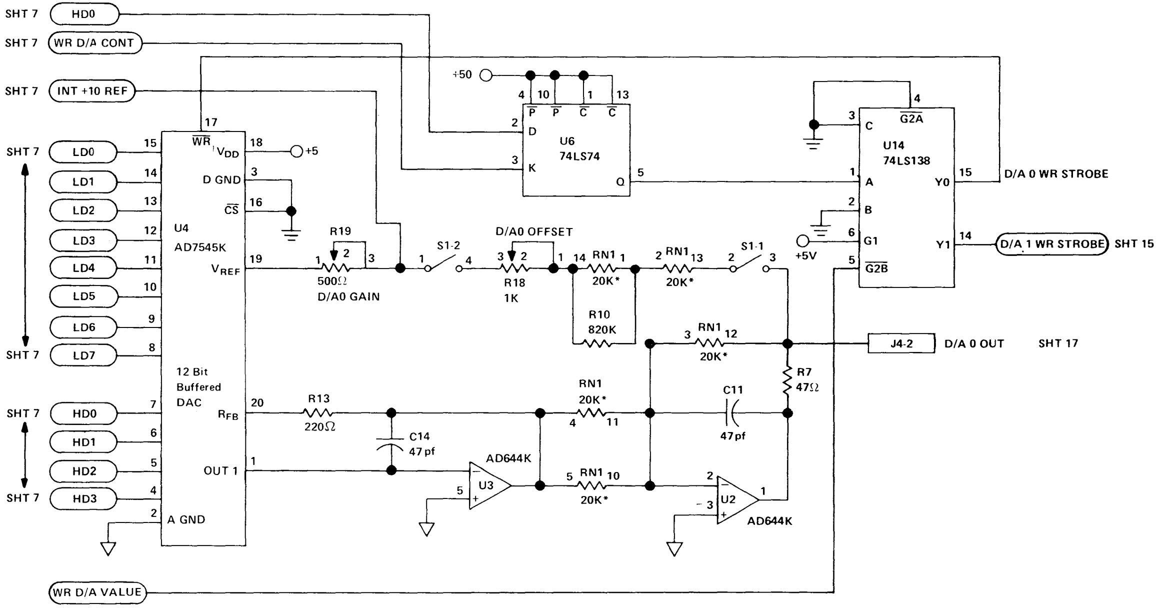


Analog Input Device Channel Multiplexer
and Sample and Hold

Sheet 13A

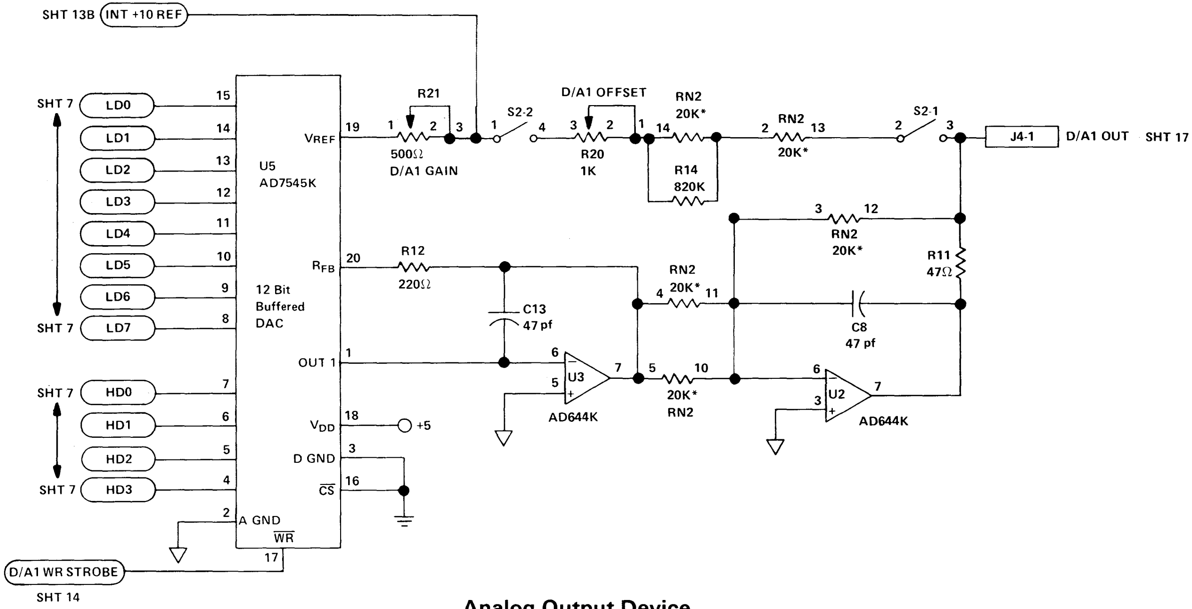


Analog Input Device
ADC AI Data Register **Sheet 13B**



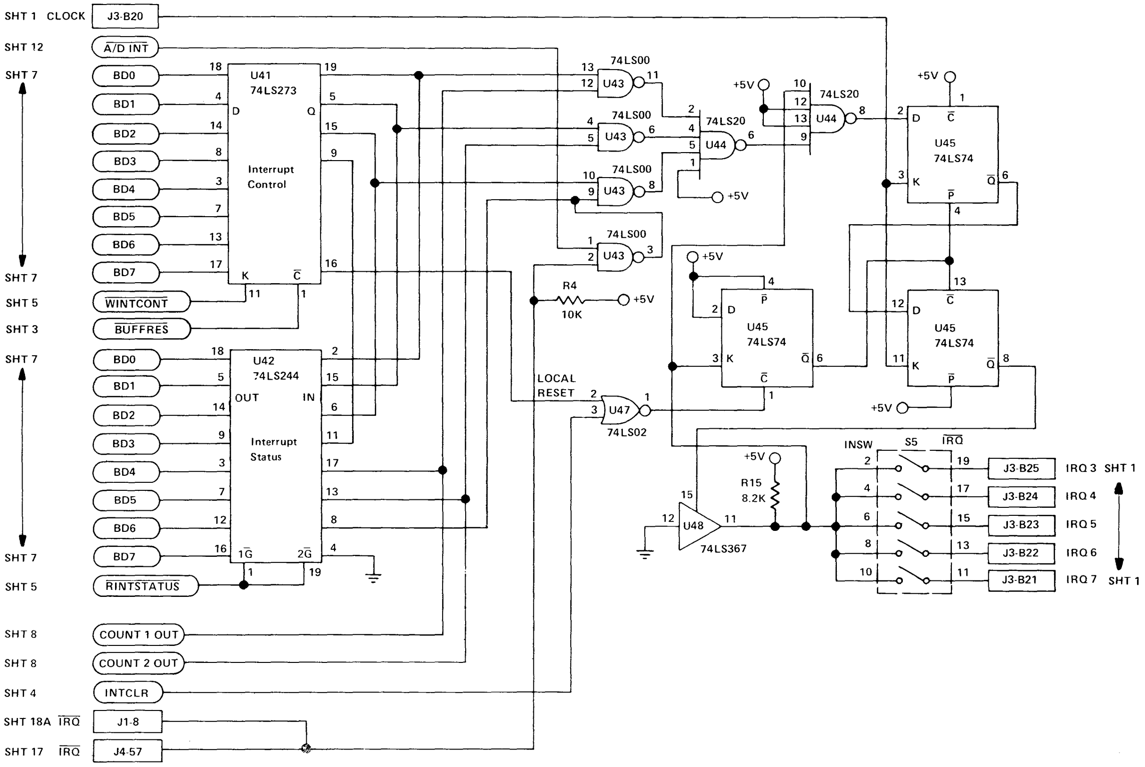
Analog Output Device
DAC for Channel 0

Sheet 14



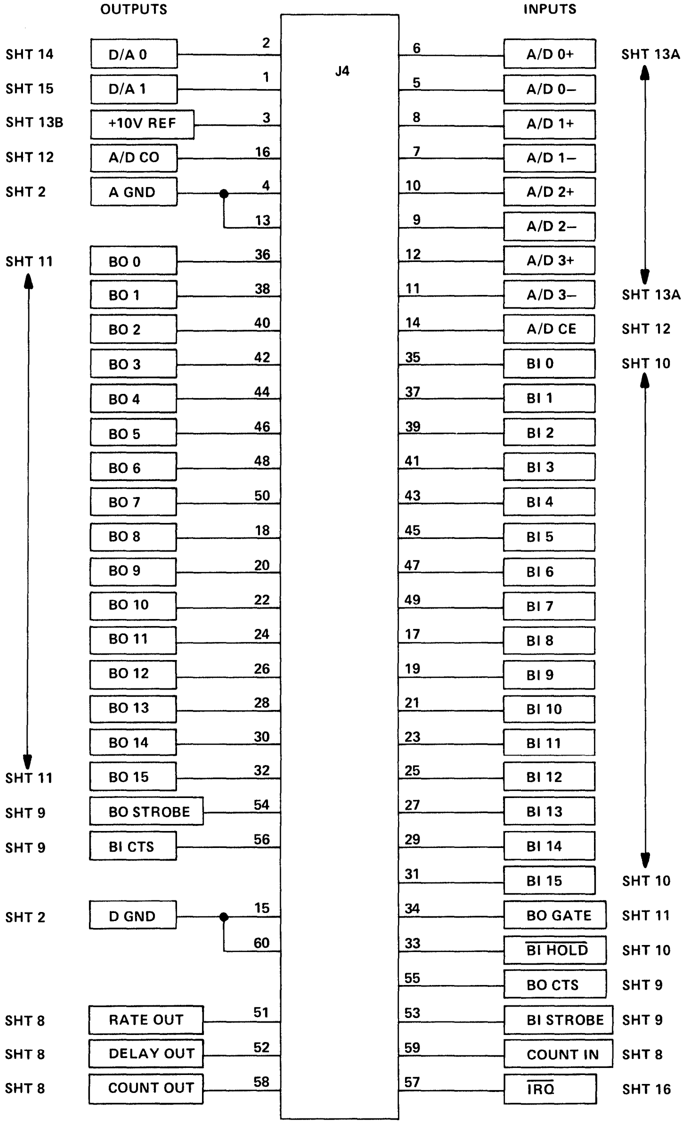
Analog Output Device
DAC for Channel 1

Sheet 15

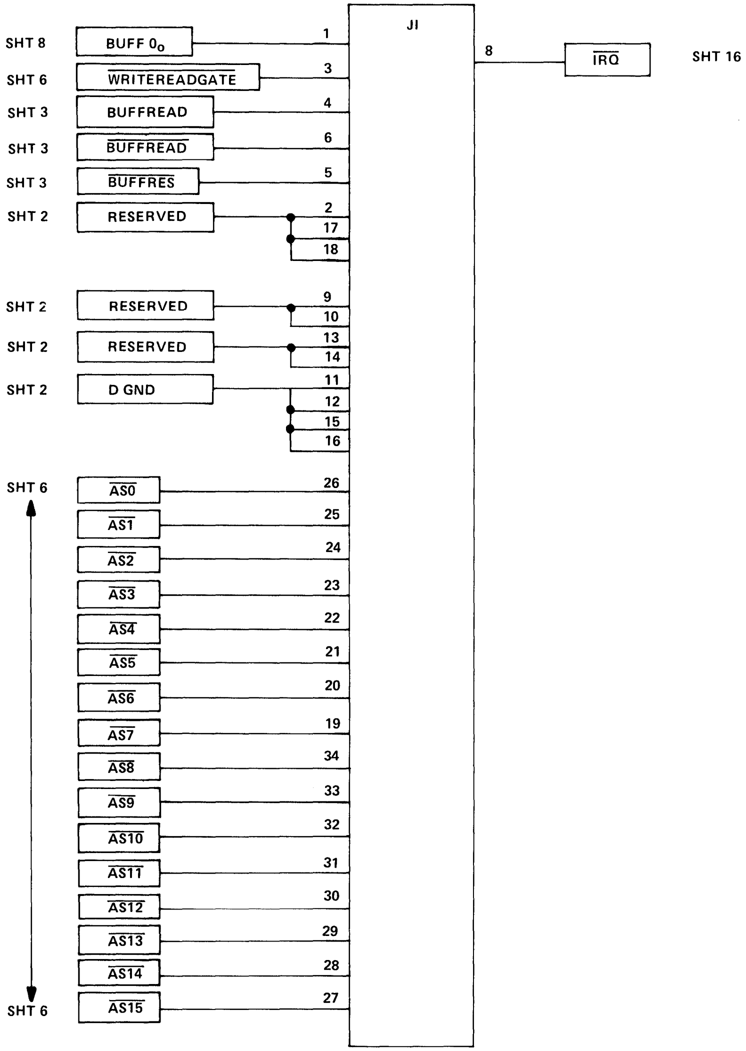


Interrupt Status and Control

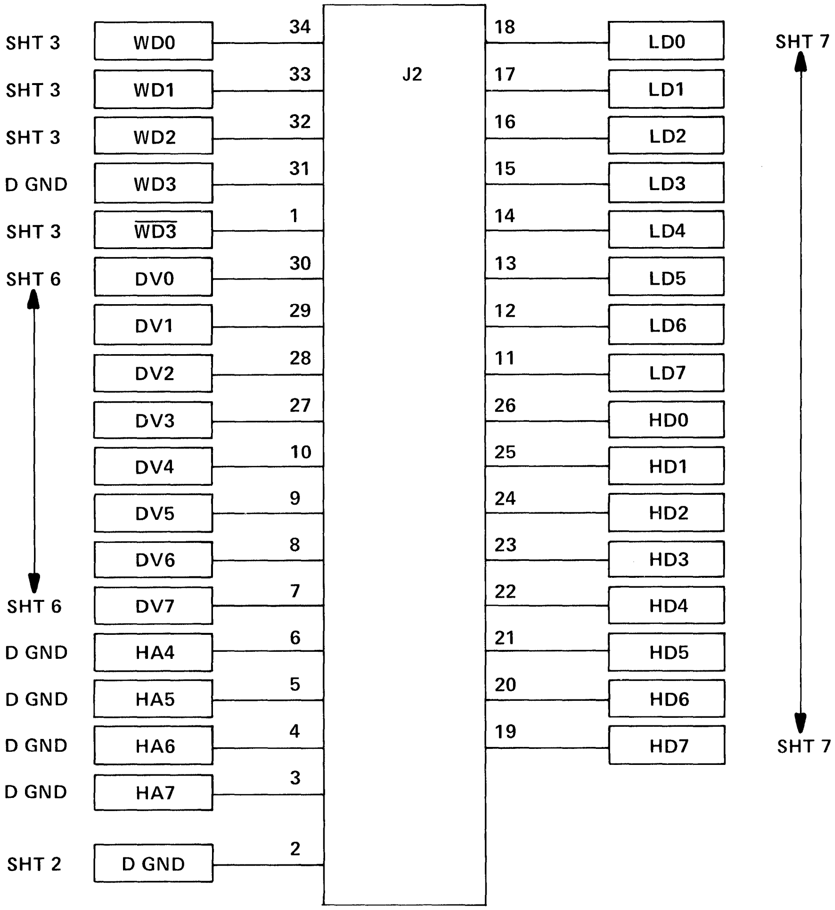
Sheet 16



Distribution Panel Connector J4 Sheet 17



Expansion Bus Connector J1 **Sheet 18A**



Expansion Bus Connector J2 **Sheet 18B**

Index

Personal Computer
Hardware Reference
Library

GENERAL PURPOSE INTERFACE BUS

IBM Personal Computer General Purpose Interface Bus Technical Reference

6138155
August 15, 1984

Contents

Description

The IBM General Purpose Interface Bus (GPIB) Adapter connects IBM Personal Computer products (PC) to a general-purpose interface bus (bus) and is designed according to the specifications of the following industry standards as understood and interpreted by IBM as of September 1983: ANSI/IEEE Std. 488-1978 (includes supplement IEEE Std. 488A-1980).

The GPIB Adapter makes it possible to use the PC as a controller for a GPIB test and measurement system. The GPIB Adapter implements bus interface functions for communication with GPIB devices, and implements device functions for communication with the PC central processor and memory.

The following lists interface function codes supported by the IBM GPIB Adapter.

Code	Description
SH1	Complete source-handshake capability
AH1	Complete acceptor-handshake capability: data-accepted and ready-for-data (RFD) holdoff on certain events
T5	Complete talker capability: Basic talker Serial poll Talk-only mode Unaddressed on my-listen-address (MLA) Send end message (END) Send end-of-string message (EOS) Dual primary addressing

TE5 Complete extended-talker capability:

Basic extended talker
Serial poll
Talk-only mode
Unaddressed on my-secondary-address (MSA) or listener-primary-addressed-state (LPAS)
Send END or EOS
Dual extended addressing with software assist

L3 Complete listener capability:

Basic listener
Listen-only mode
Unaddressed on my-talk-address (MTA)
Detect END or EOS
Dual primary addressing

LE3 Complete extended-listener capability:

Basic listener
Listen-only mode
Unaddressed on MSA or talker-primary-addressed-state (TPAS)
Detect END or EOS
Dual extended addressing with software assist

SR1 Complete service request capability

RL1 Complete remote and local capability with software interpretation

PP1 Remote parallel poll configuration

PP2 Local parallel poll configuration with software assist

DC1 Complete device-clear capability with software interpretation

DT1 Complete device-trigger capability with software interpretation

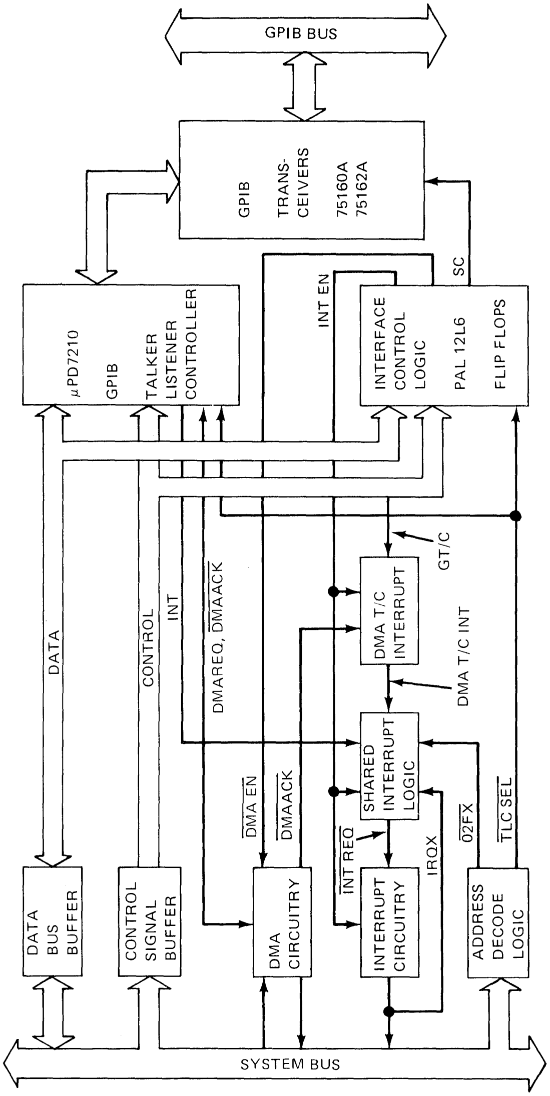
C1-5 Complete controller capability:

- System controller
- Send interface-clear (IFC) and take control
- Send remote-enable (REN)
- Respond to service-request (SRQ)
- Send interface messages
- Receive control
- Pass control
- Parallel poll
- Take control synchronously or asynchronously

E1/2 Three-state bus drivers with automatic switch to open-collector drivers during parallel poll

Major Components

The following block diagram shows the major components of the GPIB Adapter.



Data-Bus Buffer

The data-bus buffer transfers the data between the system's input/output (I/O) channel and the GPIB Adapter. The channel's data-bus signals, D0 through D7, are buffered by a Transceiver and become the internal data-bus signals, GD0 through GD7.

Control-Signal Buffer

The control-signal buffer acts as a line receiver to minimize loading on the system's I/O channel and to increase noise immunity. The I/O channel's control signals, the DMA Controller's acknowledge signals, and the address lines all are received by a Line Receiver before being used on the GPIB Adapter. The channel signals, when received by the Line Receiver, are given different names to avoid confusion. The following table shows the channel signal names.

	PC I/O Signal Name	GPIB Adapter Name
Address Lines	A10 A11 A12	RS0 RS1 RS2
Control Signals	$\overline{\text{IOR}}$ $\overline{\text{IOW}}$ RESET DRV T/C	Read ($\overline{\text{RD}}$) Write ($\overline{\text{WR}}$) RESET GT/C
DMA Control Acknowledge Signals	$\overline{\text{DACK 1}}$ $\overline{\text{DACK 2}}$ $\overline{\text{DACK 3}}$	$\overline{\text{DACK}}$ $\overline{\text{DACK}}$ $\overline{\text{DACK}}$

DMA Circuitry

The DMA circuitry recognizes when direct-memory access (DMA) operations are enabled or disabled, and routes the DMA request and acknowledge signals between the adapter and the selected DMA channels.

The DMA acknowledge and request signals at the channel connector are brought to jumper pin arrays on the board. These pin arrays are arranged so that two small pin-to-pin shunt connectors, which are supplied with the adapter, can be used to select the proper pair of DMA signals for the adapter's use. Side B of the pin array is connected to the channel connector, and side A is connected to 'DMA acknowledge' (DACK) and 'DMA request' (DRQ DLY).

$\overline{\text{DACK}}$ is ANDed with 'DMA enable' ($\overline{\text{DMA EN}}$) and becomes 'DMA acknowledge' ($\overline{\text{DMAACK}}$). When $\overline{\text{DMAACK}}$ goes low, both halves of U5 (a dual 74LS74A flip-flop) are cleared, and the $\overline{\text{DMAACK}}$ pin on the Talker/Listener/Controller (TLC) goes low. When $\overline{\text{DMAACK}}$ returns to high, the $\overline{\text{DMAACK}}$ pin on the TLC goes high. The flip-flops delay the low-to-high transition of the $\overline{\text{DMAACK}}$ signal for a minimum of 200 nanoseconds to prevent any spurious TLC DMAREQ pulses from being sent to the DMA Controller.

Interrupt Circuitry

The interrupt circuitry recognizes when interrupts have been enabled or disabled and passes or inhibits them accordingly.

The interrupt circuitry consists of logic and an array of jumper pins to allow user selection of the channel ‘interrupt request’ signal. The interrupt logic consists of one gate of a 74LS125A three-state buffer, which drives the A side of an array of jumper pins. The input of the buffer is tied to ground, and the enable for the gate is driven by the ‘interrupt request’ signal ($\overline{\text{INT REQ}}$), which is generated by the shared interrupt logic. The B side of the jumper pin array is connected to the channel ‘interrupt request’ signals, IRQ2 through IRQ7. The pin array is arranged so that a small pin-to-pin shunt connector, which is supplied with the adapter, can be used to select the desired interrupt-request level.

Because the input of the three-state buffer is tied to ground, an active interrupt request ($\overline{\text{INT REQ}}$ is low) corresponds to a low level being sent onto the selected interrupt-request bus line. When $\overline{\text{INT REQ}}$ is high, no interrupt is being requested, and the output of the buffer's gate is in a high-impedance state. The 8.2 kilo-ohm pull-up resistor attached to the output of the gate ensures that the IRQX signal (X is 2, 3, 4, 5, 6, or 7) is pulled to a high level when the board is not requesting an interrupt.

Shared Interrupt Logic

The shared interrupt logic, when used with the appropriate interrupt-handling programs, allows multiple adapters to use or share the same system I/O channel's interrupt-request line.

This interrupt-sharing scheme works properly only when adapters using the same interrupt level are installed in the same unit.

The IRQX line is treated as an open-collector type of line, even though it is connected to the output of a three-state gate. This gate pulls the line to a low level when the adapter is requesting an interrupt. Adapters using the IRQX line also can pull the line to ground at the same time. Other adapters using the same IRQX line, but not requesting an interrupt, are not driving the line, because their gates are in a high-impedance state.

The shared interrupt logic generates a low-going pulse that is two system clock periods long to request an interrupt from the processor.

When the 'interrupt enable' signal (INT EN) from the interface control logic is high, the GPIB Adapter can request interrupts. Interrupt requests to the shared interrupt logic originate from the interrupt signal (INT) of the TLC (TLC INT) and the DMA terminal-count interrupt (DMA T/C INT) logic.

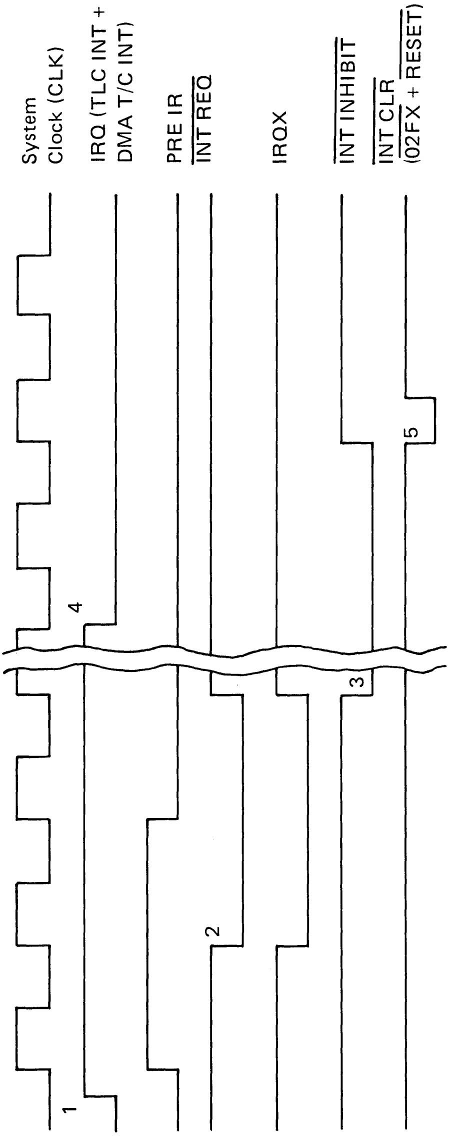
The TLC INT signal goes high, when the corresponding TLC mask bit is unmasked and any of the following GPIB events occur:

- Command pass through (unrecognized command) received
- Address pass through (unrecognized or secondary address) received
- Device trigger received
- End message received
- Device clear received
- Error (no listeners)
- Data out (ready to originate data byte)
- Data in (data byte accepted)
- Service request interrupt
- Command out (ready to originate command byte)
- Lockout-mode status change
- Remote-mode status change
- Address status change

The DMA T/C INT signal goes high whenever the DMA Controller channel being used by the adapter during a DMA transfer reaches terminal count.

If the interrupt handler chooses not to clear the source of the interrupt, the IRQX high-to-low-to-high pulse is regenerated after the handler writes to port hex 02FX. The shared interrupt logic will generate a continuous series of IRQX pulses, thereby causing a series of interrupts, until the handler services it.

The following is a timing diagram showing how the shared interrupt logic functions.



1 - Interrupt request from TLC INT or DMA T/C INT

2 - Selected PC interrupt line high-to-low transitions

3 - INT INHIBIT low prohibits any further INT REQ low transitions

4 - Interrupt handler invoked, source of interrupt cleared

5 - RESET or write to I/O port hex 02FX (X=2–7) reenables shared interrupt logic

DMA T/C Interrupt

The DMA T/C interrupt logic determines when the DMA Controller has reached terminal count and generates an interrupt request.

The 'DMA acknowledge' signal ($\overline{\text{DACK}}$) and the DMA T/C signals from the DMA Controller are received by the DMA circuitry and the control-signal buffer and become $\overline{\text{DMAACK}}$ and GT/C, respectively. These signals and INT EN generate an interrupt request (DMA T/C INT) when the DMA Controller reaches terminal count (GT/C makes a low-to-high transition). DMA T/C INT goes high on a terminal-count condition only if interrupts were previously enabled by setting any one of the enable-interrupt bits in interrupt mask register 1 or 2 (IMR1 or IMR2).

The DMA T/C INT signal is cleared to a low state when 'reset' goes high or when interrupt status register 2 (ISR2) is read. An interrupt handler routine should read ISR2.

Address Decode Logic

The address decode logic monitors the 16 address lines (A0 through A15) of the system's I/O channel to determine when the GPIB Adapter's I/O address is present on the channel, and enables read and write access to the adapter. A 13-input NAND gate is used, along with some inverters and jumpers, to decode the base address, 'TLC select' ($\overline{\text{TLC SEL}}$). The 'address enable' signal (AEN) from the processor ensures that the adapter is not inadvertently selected when the system processor does not have control of the bus.

The binary base address decoded by the NAND gate is as follows.

PC Address Select			TLC Register Select			GPIB Adapter Base Address									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	x	y	y	y	1	0	1	1	1	0	0	0	0	1

x PC Address Select (Jumpers)

y TLC Register Select (Programmed)

A second NAND gate decodes the 'interrupt enable' signal, $\overline{\text{02FX}}$. The binary decode process is as follows.

Interrupt Enable Base Address													Interrupt Level		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	1	1	1	0	z	z	z

z Interrupt Request Level. Must correspond to the IRQ level selected by jumpers.

The $\overline{\text{WR}}$ and $\overline{\text{AEN}}$ signals ensure that the $\overline{\text{02FX}}$ signal is generated only when the system processor is writing to the decoded address.

μPD7210 Talker/Listener/Controller (TLC)

The TLC implements almost all interface functions to interact with other devices on the bus. Within the TLC are 21 program registers, which are used to set up, control, and monitor the interface functions and to pass commands and data to and from the bus. Access to these functions is through 8 read-only registers and 8 write-only registers, of which 5 are indirectly addressed.

The TLC is enabled when the $\overline{\text{TLC SEL}}$ signal is low and the 'register select' signals, RS0 through RS2, are decoded internally to gain access to the appropriate register. Data on the internal data bus (GD0 through GD7) is loaded into write-only registers at the trailing edge of $\overline{\text{WR}}$. Data in the read-only registers is placed on the internal data bus for a minimum access time after both $\overline{\text{TLC SEL}}$ and $\overline{\text{RD}}$ become low.

Most of the interface functions can be implemented or initiated from either side of the TLC. The distinction between the two is generally that between local and remote interface messages.

Read Interface Registers

The following table lists the read-only interface registers and their corresponding bits.

Register		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Name	Num.	Name	Name	Name	Name	Name	Name	Name	Name
DIR	+0	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
ISR1	+1	CPT	APT	DET	ENDRX	DEC	ERR	DO	DI
ISR2	+2	INT	SRQI	LOK	REM	CO	LOKC	REMC	ADSC
SPSR	+3	S8	PEND	S6	S5	S4	S3	S2	S1
ADSR	+4	CIC	$\overline{\text{ATN}}$	SPMS	LPAS	TPAS	LA	TA	MJMN
CPTR	+5	CPT7	CPT5	CPT4	CPT4	CPT3	CPT2	CPT1	CPT0
ADR0	+6	X	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0
ADR1	+7	EOI	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1

X indicates bit is not used. Read bits that are not used may be read as 1 or 0.

DIR (Data-In Register)

The data-in register receives data and commands from the bus.

ISR1 (Interrupt Status Register 1)

The ISR1 records the occurrence of 8 conditions or events. This register is not a true status register because the bits are cleared whenever they are read.

The following table describes the bits of the ISR1 register.

Bit	Name	Value	Function
7	CPT	1	Command Pass Through. An undefined command has been received over the bus.
6	APT	1	Address Pass Through. The secondary address (Address Mode 3) has been received.
5	DET	1	Device Execute Trigger. The Device Execute Trigger (DET) command has been received.
4	END RX	1	End Received. EOI or EOS command has been received.
3	DEC	1	Device Clear. The Device Clear (DCL) command has been received.
2	ERR	1	Error. The contents of the Command/Data Out Register (CDOR) have been lost.
1	DO	1	Data Out. A data write request issued to the CDOR.
0	DI	1	Data In. A byte has been written to the DIR, and the DIR should be read.

ISR2 (Interrupt Status Register 2)

The ISR2 records the occurrence of 8 conditions or events. This register is not a true status register because the bits are cleared whenever they are read.

The following table describes the bits of the ISR2 register.

Bit	Name	Value	Function
7	INT	1	Logical OR of the enabled interrupt status bits.
6	SRQI	1	Service Request In. A respond to service request (SRQ) message has been received while the controller is active.
5	LOK	1	Lockout (non-interrupt bit). The device is in local with lockout state (LWLS) or remote with lockout state (RWLS).
4	REM	1	Remote (non-interrupt bit). The device is in remote state (REMS) or RWLS.
3	CO	1	Command Output. A request for a command to be written to the CDOR.
2	LOKC	1	Lockout Change. The value of the LOK bit (bit 5) has changed.
1	REMC	1	Remote Change. The value of the REM bit (bit 4) has changed.
0	ADSC	1	Address Status Change. One of the four bits (TA, LA, CIC, MJMN) of the address status register has changed.

SPSR (Serial Poll Status Register)

The serial poll status register echoes the contents of the serial poll mode register (SPMR). This status can be read to confirm that a request for a serial poll was accepted.

ADSR (Address Status Register)

The address status register contains information about the current addressed state of the GPIB Adapter.

The following table describes the bits of the ADSR register.

Bit	Name	Value	Function
7	CIC	1	The device is controller in charge.
6	ATN	1	Attention. The device is in data transfer mode; the $\overline{\text{ATN}}$ line is high.
5	SPMS	1	Serial poll mode state: The serial poll enable (SPE) message has been received.
4	LPAS	1	The device is in listener primary addressed state.
3	TPAS	1	The device is in talker primary addressed state.
2	LA	1	Listener Addressed: The device is in listener addressed state.
1	TA	1	Talker Addressed: The device is in talker addressed state.
0	MJMN	1	A major-talk or major-listen address has been received.
		0	A minor-talk or minor-listen address has been received.

CPTR (Command Pass Through Register)

The command pass through register reads the data on the data input/output (DIO) lines for the following situations:

If ISR1 bit 7 (CPT) is equal to 1 and auxiliary register B (AUXRB) bit 0 (CPT ENABLE) is equal to 1 then CPTR has an undefined command or has received a secondary command after an undefined primary command.

If ISR1 bit 6 (APT) is equal to 1 and Address Mode 3 is selected then CPTR contains a secondary address.

After a parallel poll the CPTR contains the parallel-poll response message.

ADR0 (Address Register 0)

Address register 0 contains the major address set by the address register (ADR), as well as the functions enabled for that address.

The following table describes the bits of the ADR0 register.

Bit	Name	Function
7		Not used
6	DT0	Disable Talker 0. Set or cleared by a write to ADR.
5	DL0	Disable Listener 0. Set or cleared by a write to ADR.
4-0	AD5-0 to AD1-0	Major Address. Set by a write to ADR.

ADR1 (Address Register 1)

Address register 1 contains the minor address set by the address register (ADR), as well as the functions enabled for that address.

The following table describes the bits of the ADR1 register.

Bit	Name	Function
7	EOI	End or Identify. Indicates that EOI message was sent on the last data byte received.
6	DT1	Disable Talker 1. Set or cleared by a write to ADR.
5	DL1	Disable Listener 1. Set or cleared by a write to ADR.
4-0	AD5-1 to AD1-1	Minor Address. Set by a write to ADR.

Write Interface Registers

The following table lists the write-only interface registers.

Register		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Name	Num.	Name	Name	Name	Name	Name	Name	Name	Name
CDOR	+0	CDO7	CDO6	CDO5	CDO4	CDO3	CDO2	CDO1	CDO0
IMR1	+1	CPTIE	APTIE	DETIE	ENDIE	DECIE	ERRIE	DOIE	DIIE
IMR2	+2	X	SRQIIE	DMAO	DMAI	COIE	LOKCIE	REMCIE	ADSCIE
SPMR	+3	S8	rsv	S6	S5	S4	S3	S2	S1
ADMR	+4	ton	lon	TRM1	TRM0	X	X	ADM1	ADM0
AUXMR	+5	CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0
ADR	+6	ARS	DT	DL	AD5	AD4	AD3	AD2	AD1
EOSR	+7	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0

X indicates bit is not used.

CDOR (Command/Data Out Register)

The command/data out register is an output-only register used to send commands or data to the bus.

IMR1 (Interrupt Mask Register 1)

The interrupt mask registers are used to enable or disable the generation of the INT signal on the occurrence of key events.

The following table describes the bits of the IMR1 register.

Bit	Name	Function
7	CPTIE	Command Pass Through, Interrupt Enable
6	APTIE	Address Pass Through, Interrupt Enable
5	DETIE	Device Trigger, Interrupt Enable
4	ENDIE	END message (EOI) or EOS message received, Interrupt Enable
3	DECIE	Device Clear, Interrupt Enable
2	ERRIE	Error, Interrupt Enable
1	DOIE	Data Out, Interrupt Enable
0	DIIE	Data In, Interrupt Enable

IMR2 (Interrupt Mask Register 2)

The following table describes the bits of the IMR2 register.

Bit	Name	Function
7		Write a 0 to this bit.
6	SRQIIE	Service Request In, Interrupt Enable
5	DMAO	DMA Output (non-interrupt). Enables or disables a DMA transfer to the data registers
4	DMAI	DMA Input (non-interrupt). Enables or disables a DMA transfer to the data registers.
3	COIE	Command Output, Interrupt Enable
2	LOKCIE	Lockout Change, Interrupt Enable
1	REMCIE	Remote Change, Interrupt Enable
0	ADSCIE	Address Status Change, Interrupt Enable

SPMR (Serial Poll Mode Register)

The serial poll mode register holds the status byte and the local request service (rsv) message (bit 6). When rsv is 1, the adapter enters the service request state (SRQS). When a serial poll is requested, the adapter sends the contents of the SPMR over the bus and clears the rsv bit upon completion of the poll.

ADMR (Address Mode Register)

The address mode register selects the functions of the Transmit and Receive pins (T/R2 and T/R3) and selects the address mode.

The following table shows the functions of pins T/R2 and T/R3 for the various settings of bits 4 and 5.

Bit 5 TRM1	Bit 4 TRM0	T/R2	T/R3
0	0	EOIOE	TRIG
0	1	CIC	TRIG
1	0	CIC	EOIOE
1	1	CIC	PE

The following table shows the various address modes as selected by bits 0, 1, 6, and 7.

Bit 7 ton	Bit 6 lon	Bit 1 ADM1	Bit 0 ADM0	Address Mode	Contents of Adr. Reg. 0	Contents of Adr. Reg. 1
1	0	0	0	Talk Only	Not Used	Not Used
0	1	0	0	Listen Only	Not Used	Not Used
0	0	0	1	Address Mode 1	Major Talk or Major Listen Address	Minor Talk or Minor Listen Address
0	0	1	0	Address Mode 2	Primary Address Talk or Listen	Secondary Address Talk or Listen
0	0	1	1	Address Mode 3	Primary Address Major Talk or Major Listen	Primary Address Minor Talk or Minor Listen

AUXMR (Auxiliary Mode Register)

A write to the auxiliary mode register causes one of the following to occur:

- An auxiliary command is issued.
- The state-change-prohibit-time is set using the internal counter register.
- The parallel poll register is written to.
- Auxiliary register A, B, or E is written to.

The following table shows the 5 internal registers and their corresponding functions.

Control Code			Command Code				
CNT2	**CNT1**	**CNT0**	**COM4**	**COM3**	**COM2**	**COM1**	**COM0**
When CNT2-CNT0 is:			ICR is loaded with:				
0	0	1	0	CLK3	CLK2	CLK1	CLK0
			PPR is loaded with:				
0	1	1	U	S	P3	P2	P1
			AUXRA is loaded with:				
1	0	0	BIN	XEOS	REOS	HLDE	HLDA
			AUXRB is loaded with:				
1	0	1	ISS	INV	TRI	SPEOI	CPT ENABLE
			AUXRE is loaded with:				
1	1	0	0	0	0	DHDC	DHDT

To issue the auxiliary commands shown in the following table, write a binary 000COM$_4$COM$_3$COM$_2$COM$_1$COM$_0$ to the auxiliary mode register.

The following table lists the auxiliary command summary.

Auxiliary Command	Function Code*					Hex Code**
	COM4	COM3	COM2	COM1	COM0	
Immediate Execute pon	0	0	0	0	0	00
Chip Reset	0	0	0	0	1	02
Finish Handshake	0	0	0	1	1	03
Trigger	0	0	1	0	0	04
Return to Local	0	0	1	0	1	05
Send EOI	0	0	1	1	0	06
Non-Valid Secondary Command or Address	0	0	1	1	1	07
Valid Secondary Command or Address	0	1	1	1	1	0F
Clear Parallel Poll Flag	0	0	0	0	1	01
Set Parallel Poll Flag	0	1	0	0	1	09
Take Control Asynchronously (pulsed)	1	0	0	0	1	11
Take Control Synchronously	1	0	0	1	0	12
Take Control Synchronously on End	1	1	0	1	0	1A
Go to Standby	1	0	0	0	0	10
Listen	1	0	0	1	1	13
Listen in Continuous Mode	1	1	0	1	1	1B
Local Unlisten	1	1	1	0	0	1C
Execute Parallel Poll	1	1	1	0	1	1D
Set IFC	1	1	1	1	0	1E
Clear IFC	1	0	1	1	0	16
Set REN	1	1	1	1	1	1F
Disable System Control	1	0	1	0	0	14

***CNT2-CNT0 set to 000 binary**

****Represents all eight bits of the auxilliary mode register**

AUXMR Internal Registers

The following information describes the AUXMR internal registers.

Internal Counter Register (ICR):

A write to this register (binary 0010clk$_3$clk$_2$clk$_1$clk$_0$) sets the state-change-prohibit-times T_1, T_6, T_7, and T_9 as referenced in the ANSI/IEEE std.488-1978.

The ICR should be set to equal the system clock frequency. The following table shows the internal counter register.

ICR							
7	6	5	4	3	2	1	0
0	0	1	0	clk$_3$	clk$_2$	clk$_1$	clk$_0$

Parallel Poll Register (PPR):

You can write to the parallel poll register by writing a binary $011USP_3P_2P_1$ to the AUXMR.

You should not write to this register if you are using subset PP1 (remote parallel poll configuration) as the parallel poll (PP) interface function.

The following table shows the functions of the bits of the parallel poll register.

Bit	Bit Name	Value	Function
4	U	1	Disables participation in parallel poll
		0	Enables participation in parallel poll
3	S	1	Sense of status is the same as the ist message (If S=ist=1, PPR is logical 1).
		0	Sense of status is reversed from the ist (If S=0 and ist=0, PPR is a logical 1).
2-0	P(3)-P(1)		Binary value of which data line (DIO8-DIO1) to assert during a parallel poll.

U = Parallel Poll Unconfigure;
S = Status Bit Polarity

Auxiliary Register A (AUXRA):

You can write to this register by writing a binary $100A_4A_3A_2A_1A_0$ to the AUXMR.

The data-receiving modes are set by bits 1 (A_1) and 0 (A_0) as follows.

A_1HLDE	A_0HLDA	Data Receiving Mode
0	0	Normal Handshake Mode
0	1	Ready for Data (RFD) Hold Off on All (HLDA) Data Mode
1	0	RFD Hold Off on End (HLDE) Mode
1	1	Continuous Mode

The transmission and receiving modes of the EOI or EOS message are set by bits 4 (A_4), 3 (A_3), and 2 (A_2) as follows.

Bit Name	Value	Function	Description
A_4BIN	0 1	7-bit EOS 8-bit EOS	Selects seven or eight bits as the valid length of the EOS message
A_3XEOS	0 1	Prohibit Permit	Transmit EOS. Permits or prohibits the automatic transmission of the EOI message at the same time as the EOS message in talker-active-state (TACS).
A_2REOS	0 1	Prohibit Permit	Receive EOS. Permits or prohibits setting the EOI bit at reception of the EOS message

Auxiliary Register B (AUXRB):

You can write to this register by writing a binary $101B_4B_3B_2B_1B_0$ to the AUXMR.

Following is a description of the bits of the AUXRB register.

Bit Name	Value	Function	Description
B_4ISS	0	Individual status (ist)=Parallel Poll flag	The value of the Parallel Poll flag is taken as the ist local message
	1	ist=SRQS	SRQS indicates the value of the ist local message (the Parallel Poll flag is ignored) SRQS=ist=1; SRQS=ist=0
B_3INV		1 INT 0 INT	Specifies the active level of the INT pin
B_2TRI	0	T(1) (low speed)	Sets low speed as T(1) in all cases
	1	T(1) (high speed)	Sets high speed as T(1) of handshake after transmission of second byte following data transmission
B_1SPEOI	1 0	Permit Prohibit	Permits or prohibits the transmission of the END message in serial poll address state.
B_0CPT ENABLE	1 0	Permit Prohibit	Permits or prohibits the setting of the CPT bit on receipt of an undefined command

Auxiliary Register E (AUXRE):

You can write to auxiliary register E by writing a binary $110000E_1E_0$ to the AUXMR.

Following is a description of the bits of the AUXRE register.

Bit Name	Value	Function
E_1DHDC	1 enables 0 disables	Enables or disables DAC hold-off by initiating device clear active state (DCAS)
E_0DHDT	1 enables 0 disables	Enables or disables DAC hold-off by initiating device trigger active state (DTAS)

ADR (Address Register)

A write to the address register sets the addresses and functions for ADR0 (major) and ADR1 (minor).

Following is a description of the bits of the ADR register.

Bit	Bit Name	Value	Function	Description
7	ARS	0 1	Address register 0 Address register 1	Selects the address register (ADR0 or ADR1) to which the low-order bits are written (AD1 to AD5)
6	DT	0 1	Permitted Prohibited	Permits or prohibits the set address (AD1 to AD5) detected as a talk address. This bit corresponds to DT1 or DT0 of the address registers.
5	DL	0 1	Permitted Prohibited	Permits or prohibits the set address (AD1 to AD5) detected as a listen address. This bit corresponds to DL1 or DL0 of the address registers.
4-0	AD1 to AD5			These bits indicate device addresses and correspond to AD5-0 to AD1-0 and AD5-1 to AD1-1.

EOSR (End-of-String Register)

The end-of-string register contains the 7- or 8-bit EOS message byte used by the GPIB Adapter to detect the end of a data block transfer.

GPIB Transceivers

Special-purpose, multi-function, GPIB transceivers serve as the interface between the adapter and the bus.

The TLC communicates with the bus through two special-purpose transceivers, a DS75160A for the data signals, and a DS75162A for the handshake and interface management signals. The direction of signals through these transceivers is controlled by three signals from the TLC (TE, DC, and PE) and the 'system control' (SC) signal from the interface control logic.

'Talk enable' (TE) is high when the TLC is a talker or active controller, and low when it is a listener. It controls the direction of the data, handshake, and EOI signals.

'Direction control' (DC) is inverted so that it is high when the TLC is controller-in-charge (CIC), and low at other times. It controls the direction of the ATN and SRQ signals.

'Pull-up enable' (PE) is high when the three-state driver mode is active, and low when the open collector mode is active. When a parallel poll is requested, the transceiver switches to open collector mode.

SC controls the direction of the 'interface clear' (IFC) and 'remote enable' (REN) signals, driving the bus when SC is high and receiving from the bus when it is low.

Interface Control Logic

The interface control logic monitors the GPIB Adapter's data and control busses and generates signals that control interrupt requests, DMA requests, and system-controller capability.

The PAL12L6 is the main component in this group. The PAL monitors the internal data bus (GD0 through GD7), the 'TLC select' signal ($\overline{\text{TLC SEL}}$), and the 'TLC register select' signals (RS0 through RS3). These inputs allow the internal logic of the PAL to generate the following outputs:

- $\overline{\text{SC OFF}}$, or 'system controller off', which is true (low) when the TLC AUXMR is being selected and the Chip Reset or Disable System Control auxiliary commands are being sent.

- $\overline{\text{SC ON}}$, or 'system controller on,' which is true (low) when the command being written to the AUXMR involves setting or clearing the GPIB IFC or REN signals.

- $\overline{\text{IR2}}$, or 'TLC interrupt register 2,' which is true (low) when either ISR2 or IMR2 is being selected by the 'register select' lines.

- $\overline{\text{DMA ON}}$, which is true (low) when the internal data bus indicates that the DMA out (DMAO) or DMA in (DMAI) bits of IMR2 are being set.

- $\overline{\text{IR1}}$, or 'TLC interrupt register 1,' which is true (low) when either ISR1 or IMR1 is being selected by the 'register select' lines.

- INT ON, or 'interrupt on,' which is true (high) when the internal data bus signifies that any of the interrupt-enable bits in IMR1 or IMR2 are being set.

The SC signal is latched high, signifying that the GPIB Adapter is the system controller, whenever the TLC receives an auxiliary command to set or clear the REN or IFC signals. SC is returned to a low level when a Disable System Control or a Chip Reset auxiliary command is sent to the TLC, or when $\overline{\text{RESET}}$ is low.

The $\overline{\text{DMA EN}}$ signal is latched at a low level when either of the DMA enable bits in the TLC IMR2 is set. $\overline{\text{DMA EN}}$ is returned to a high level if both bits are cleared or if $\overline{\text{RESET}}$ is low.

The INT EN signal is latched at a high level if any of the interrupt-enable bits in IMR1 or IMR2 are set. INT EN is returned to a low level if $\overline{\text{RESET}}$ is low or if all interrupt-enable bits are cleared.

Programming Considerations

When power is switched on, the system issues a bus reset causing the following:

- The system-controller bit is cleared.
- The 'interrupt request' line is tri-stated.
- 'DMA request' is tri-stated.
- RESET is sent to the TLC, causing the following:
  - The local command, power on (pon), is set, and the interface functions are placed into their idle states.
  - The serial poll mode register (SPMR) is cleared.
  - The end-or-identify (EOI) bit is cleared.
  - AUXRA, AUXRB, and AUXRE are cleared.
  - The Parallel Poll flag and the request system control (rsc) locate message are cleared.
  - The transmit- and receive-mode bits (TRM0 and TRM1) in the address mode register are cleared.
  - All auxiliary mode commands are cleared and prevented from executing.

All other register contents should be considered as undefined after a bus reset has been issued.

The GPIB Adapter's other registers can be initialized while pon is set. A typical programmed initialization sequence for the adapter may include the following steps:

1. Write the Chip Reset auxiliary command (hex 02) to AUXMR.

2. Set or clear the desired interrupt-enable bits in IMR1 and IMR2.

3. Write to ADR to set the desired address for both ADR0 and ADR1.

4. Set the TRM0 and TRM1 bits and select the desired addressing mode in the ADMR.

5. Write the serial poll response to the SPMR.

6. If using a remote setup, clear the parallel poll register (PPR). If using a local setup, load the parallel poll response into the PPR.

7. Clear pon by issuing the Immediate Execute pon auxiliary command.

The auxiliary commands can now be executed.

The GPIB Adapter as a Controller

The GPIB Adapter can become controller-in-charge (CIC) either by being the system controller and taking control or by having control of the bus passed from the current active controller.

The active controller has the ability to make ATN active and send multi-line commands and conduct parallel polls.

If the GPIB Adapter is the system controller, it can take control by executing the following command sequence:

1. Issue the Set IFC (hex 1F) auxiliary command. (This command should never be executed if the adapter is not system controller.)

2. Wait a minimum of 100 microseconds

3. Issue the Clear IFC (hex 1E) auxiliary command.

If the bus has an active controller, the GPIB Adapter waits until the controller becomes idle before making ATN active.

If the adapter is not the system controller, it can be passed control by the active controller using the following sequence:

1. Receive the My-Talk-Address (MTA) command.

2. Receive the Take Control (TCT) command.

3. The active controller sees the completed handshake and makes ATN inactive.

4. The adapter automatically becomes CIC and makes ATN active.

When the adapter becomes the active controller, the following bits are set:

- System controller bit
- CIC bit in the ADSR
- CO bit in ISR2.

The GPIB Adapter sends commands as an active controller by writing to the CDOR in response to the setting of the CO bit in ISR2. The TLC responds to and recognizes the interface commands it sends as well as receives.

To perform data transfers, the GPIB Adapter enters the controller standby state (CSBS). When the data transfer is complete, the adapter resumes active control of the bus. The way this is done depends on the role the adapter is to play in the data transfer. The following three cases describe the ways the adapter completes a data transfer and then resumes control.

Case 1: GPIB Adapter as the Talker

1. If the CO bit is set, send the GPIB Adapter's MTA.
2. With CO set, issue the Go to Standby auxiliary command (hex 10).
3. Complete the data transfer.
4. If DO is set, issue the Take Control Asynchronously auxiliary command (hex 11).

Case 2: GPIB Adapter as the Listener

1. With CO set, issue the Listen auxiliary command (hex 13).

2. With CO set, issue the Go to Standby auxiliary command (hex 10).

3. Begin data transfer.

4. After the DI bit is set and before reading the last byte from the DIR, issue the Take Control Synchronously auxiliary command (hex 12).

5. Read the DIR. The GPIB Adapter then becomes the active controller automatically.

   Note: The last byte of the data transfer can be detected by checking the END RX bit in ISR1.

Case 3: GPIB Adapter as neither the Talker nor Listener

Note: The talker must finish the data transfer with the END or EOS message.

1. With CO set, issue the Listen auxiliary command (hex 13).

2. Set the hold-off-on-end (HLDE) bit in AUXRA.

3. Issue the Go to Standby auxiliary command (hex 10).

4. Begin data transfer.

5. When DI is set, read the DIR so that the handshake cycle can be completed by other devices on the GPIB.

6. When HLDE occurs, issue the Take Control Synchronously auxiliary command. Hold-off also can be detected by reading the END RX bit in ISR1.

In Cases 2 and 3, the Take Control Asynchronously auxiliary command can be issued when the possibility of disrupting an in-progress handshake is acceptable.

In all cases, a CO status indicates the GPIB Adapter is now the active controller.

The GPIB Adapter can relinquish control of the bus and become an idle controller in two ways:

1. By issuing the Chip Reset or Disable System Control auxiliary command.

2. By passing control to another device. This is done by sending the MTA of the other device followed by the TCT interface command. The GPIB Adapter makes ATN inactive as soon as the TCT command is accepted.

The GPIB Adapter as a Talker or Listener

The GPIB Adapter may be either the GPIB talker or listener, but not both simultaneously. Either function is deactivated automatically if the other is activated. The talker-addressed (TA), listener-addressed (LA), and $\overline{\text{ATN}}$ bits in the ADSR indicate the specific state of the adapter as follows:

$\overline{\text{ATN}}$	TA	LA	
0	1	0	Addressed Talker—cannot send data
1	1	0	Active Talker—can send data
0	0	1	Addressed Listener—cannot receive data
1	0	1	Active Listener—can receive data

Programmed Implementation

When the GPIB has no controller, the talk only (ton) and listen only (lon) local commands are used to set the talker and listener interface functions. These commands are set in the ADMR and should be programmed during initialization.

Addressed Implementation

The GPIB Adapter responds to the address command that the active controller sends over the bus, regardless of which device is in control. This enables the adapter to respond to it’s own addressing commands which the adapter sends over the bus. The adapter has three address modes, which are selected by writing to the ADMR. Descriptions of the address modes follow.

Address Mode 1

In this mode, the GPIB Adapter responds to two primary addresses, ADR0 (major) and ADR1 (minor).

The receipt of the GPIB Adapter's primary My Listen Address (MLA) command has the following effects:

- LA bit in ADSR is set.
- ADSC bit in ISR2 is set.
- MJMN bit in ADSR is set if MLA corresponds to ADR1.
- DI bit is set in ISR1 upon receipt of a data byte in the DIR.

The receipt of the GPIB Adapter's primary My Talk Address (MTA) command has the following effects:

- TA bit in ADSR is set.
- ADSC bit in ISR2 is set.
- MJMN bit in ADSR is set if MTA corresponds to ADR1.
- DO bit in ISR1 is set when the CDOR is ready to send another data byte.

Address Mode 2

Address mode 2 is used for primary and secondary addressing as specified by the talker-extended (TE) and listener-extended (LE) interface functions. ADR0 contains the primary address, and ADR1 the secondary address.

The sequence for programming the TE interface function is shown in the following.

Step	Results
1. MTA received (Hex 40 + ADR0 address)	TPAS bit (in ADSR) = 1
2. MSA received (Hex 60 + ADR1 address)	TA in ADSR = 1 ADSC in ISR2 = 1 DO in ISR1 = 1 Now in TADS (Talker Addressed State)

The following shows the sequence for programming the LE interface function.

Step	Results
1. MLA received (Hex 20 + ADR0 address)	LPAS bit (ADSR) = 1
2. MSA received (Hex 60 + ADR1 address)	TA bit in ADSR = 1 ADSC bit in ISR2 = 1 DI in ISR1 = 1 (when data byte is available in DIR) Now in LADS (Listener Addressed State)

In both listener and talker addressing sequences, the My-Secondary-Address (MSA) command must be received before the receipt of another primary-command-group (PCG) command. The MJMN bit indicates which register is referred to by the address status.

Address Mode 3

This mode is used to implement the TE and LE interface functions with the addition of two possible primary addresses, major and minor. The proper operation of address mode 3 is as follows:

Step	Results
1.Primary Address (MLA or MTA) received.	TPAS in ADSR = 1 if MTA received LPAS in ADSR = 1 if MLA received MJMN in ADSR = 1 if address is ADR1
2.Secondary Address received	APT in ISR1 = 1 Data Accepted handshake hold-off is activated
3. In program, determine if primary address is talk, listen, major, or minor (use TPAS, LPAS, and MJMN bits)	
4. Read the secondary address command from the CPTR, determine if it is the address of the GPIB Adapter	
5. If not the adapter's address, issue the non-valid (hex 07) auxilliary command.	Command was other secondary address (OSA), GPIB adapter enters Listen and Talk idle state. Handshake is completed.

6. If the secondary address is the adapter's address, issue the valid (hex 0F) auxilliary command.	LA = 1 and TA = 0 if LPAS = 1
	TA = 1 and LA = 0 if TPAS = 1
	Data-accepted message sent true and handshake is completed.
	Now in TADS (TA=1) or LADS (LA=1).

In both listen- and talk-mode sequences, the secondary address command will continue to generate a 'data accepted' holdoff with APT set until another PCG command is received. In this way, the controller can address several devices having the same primary address without repeating the primary address command.

Sending and Receiving Commands

When the GPIB Adapter is a talker or listener, data or device-dependent commands can be sent or received using DMA or programmed I/O.

Programmed I/O Data Transfer

When the GPIB Adapter is operating as a talker, the DO bit in ISR1 is set when all addressed listeners are ready to receive a data byte on the bus. The GPIB Adapter responds by writing the next byte to the CDOR.

When the GPIB Adapter is a listener, the DI bit in ISR1 is set when a data byte is available in the DIR. The DIR is then read to complete the handshake.

In both cases, remember that the DO and DI bits are cleared when ISR1 is read.

DMA Data Transfer

When the GPIB Adapter is properly set up for a DMA transfer, the sending and receiving of data is controlled by the TLC and the DMA Controller. During the sending of data (DMAO bit is set), DRQ is driven high under the same conditions that set the DO bit. During the receipt of data (DMAI is set), DRQ is driven high under the same conditions that set the DI bit.

Sending END or EOS

The END command is sent by issuing the Send EOI auxiliary command just before writing the last data byte to the CDOR. The EOS command is sent simply by making the last byte the EOS code.

Stopping on an END or EOS

The END status bit in ISR1 is used to inform the program of the receipt of an END command or EOS command.

Serial Polls

Serial polls allow the controller-in-charge to obtain detailed status information about each device set up for responding.

Conducting Serial Polls

To conduct a serial poll, the adapter must:

1. Become active controller and send the Unlisten (UNL) command.

2. Send the My-Talk-Address (MTA) command of the device to be polled.

3. Send the Serial Poll Enable (SPE) command.

4. Issue the Listen auxiliary command.

5. Issue the Go to Standby auxiliary command.

6. Issue the Take Control Synchronously auxiliary command.

7. Read the device's status byte from the Data Input Register (DIR).

8. Send the Serial Poll Disable (SPD) command.

9. Repeat Steps 1 through 8 for all devices on the bus that must be polled.

Responding to a Serial Poll

The following are recommended steps for requesting service:

1. Check the PEND bit of the SPMR. If the bit is set, the GPIB Adapter is currently in the serial poll active state (SPAS).

2. If PEND is 0, write to the SPMR the desired status byte (STB) with rsv (bit 6).

3. When the active controller polls the GPIB Adapter serially, the adapter automatically transfers the STB message to the bus with DIO7 made active, and then makes the SRQ line inactive.

   Note: If a serial poll is in progress, the GPIB Adapter waits for its completion before making the SRQ line active.

If the serial poll end-of-information (SPEOI) bit of AUXRB is set, the EOI line is made active with the STB.

Parallel Polls

The active controller uses parallel polls to check the status of several devices simultaneously. The meaning of the status returned by the polled devices is device-dependent, but two general uses for parallel polls follow:

- When the controller sees an active SRQ in a system with several devices, it can quickly determine which one requested a serial poll using, usually, only one parallel poll.
- In systems for which the response time required for the controller to service a device is short, and the number of devices is few, parallel polls can replace serial polls entirely, if the controller polls frequently.

Configuring for a Parallel Poll

Before conducting a parallel poll, the controller must establish the desired parallel poll response for each device. This can be done in two ways:

- Local configuration (parallel poll function, subset PP2): This involves assigning the sense of the line and the 'response' line from the device side. This setup does not change once the device is installed.

- Remote configuration (parallel poll function, subset PP1): The active controller dynamically assigns the sense of the line and the 'response' line using interface commands. The GPIB Adapter, when operating as the system controller, should execute the following sequence:

    1. Become an active controller.

    2. Send the Unlisten (UNL) command.

    3. Send the My-Listen-Address (MLA) of the first device to be configured.

    4. Send the Parallel Poll Configure (PPC) command.

    5. Send the Parallel Poll Enable (PPE) command for that device.

    6. Repeat Steps 2 through 5 for each additional device.

Because there are 8 data lines, each with two possible responses (true or false), 16 responses are possible. The data lines are driven open collector during parallel polls, so more than one device can respond on a certain data line. A device sending the line true overrides any device sending the line false.

For each data line, the sense for a parallel poll can be set as follows:

DIO Line Value	Sense (S)	Individual Status (ist)
True (1)	0	0
False (0)	0	1
False (0)	1	0
True (1)	1	1

Conducting a Parallel Poll

To conduct a parallel poll, the GPIB Adapter does the following:

1. Becomes active controller.

2. Issues the Execute Parallel Poll auxiliary command (hex 1D).

3. Waits for the DO bit to be set, signaling the completion of the parallel poll.

4. Reads parallel poll response through the CPTR, and responds as necessary.

   Note: If more than one device is driving a certain data line, Steps 2 through 4 may have to be repeated.

Disabling Parallel Polling

To disable a device from participating in parallel polls, the GPIB Adapter:

1. Becomes active controller.

2. Sends the Unlisten (UNL) command.

3. Sends the My-Listen-Address (MLA) command for the device to be disabled.

4. Sends the Parallel Poll Disable (PPD) command.

5. Sends the Parallel Poll Unconfigure (PPU) command.

6. Repeats Steps 2 through 5 for each device to be disabled.

Responding to a Parallel Poll

Before the GPIB Adapter can be polled by the CIC, the following must occur:

- For a remote setup, the parallel poll commands are interpreted without program assistance.
- For a local setup, the controller sets the desired parallel poll response by writing to the parallel poll register.
- The GPIB Adapter then selects the source and value of the local individual status (ist) message (see the following figure).

Parallel Poll Function	Individual Status Select (ISS in AUXRB)
ist set and cleared by the Set and Clear Parallel Poll Flag auxilliary commands.	0
ist and SRQ set when rsv set. ist, rsv, and SRQ cleared by completion of a parallel poll.	1

After proper setup, programming is required for setting and clearing the local ist command as desired.

Interrupts

Interrupts are enabled through a hardware jumper to one of the six available IRQ lines on the system's I/O channel.

Interrupt requests from the TLC are enabled using the IE bits in IMR1 and IMR2.

The DMA T/C interrupt is implemented external to the TLC and is enabled whenever the DMAO or DMAI bit, and at least one of the IE bits, is set. The DMA T/C interrupt is discussed further in the next section.

The GPIB Adapter drives the selected IRQ line when at least one of the IE bits is set in IMR1 or IMR2; otherwise, the selected IRQ line is tri-stated.

Once made active, the interrupt request line remains active until the corresponding status register is read (ISR1 or ISR2).

The DMA T/C interrupt has no corresponding status register or bit, but is cleared when ISR2 is read.

Interrupts are selected by the Interrupt Controller on the system board. When none of the IE bits in the TLC is set, the GPIB Adapter's IRQ line is pulled to a high state. The IRQ line must be pulled high because a tri-state may look like an interrupt request to the Interrupt Controller.

An interrupt handler routine for the GPIB Adapter must do the following:

1. Read ISR2 to see if the INT bit is set, thus confirming that the GPIB Adapter has issued an interrupt. If the DMA T/C interrupt has been enabled, the INT bit will not be set when it occurs; however, the DMA Controller will indicate whether it has reached terminal count for the GPIB Adapter's DMA channel.

2. Read ISR1. Reading both ISR2 and ISR1 will clear an interrupt caused by any of the 13 possible conditions.

3. Write the End of Interrupt command to the Interrupt Controller.

4. Write to I/O address hex 2FX, where X is the interrupt level being used by the GPIB Adapter. Writing to 2FX reenables interrupts on the adapter.

In a system where several GPIB Adapters share the same interrupt level, Steps 1 through 4 should be used in each adapter's interrupt handler routine. Each of the adapters is polled until the one that caused the interrupt is found. When hex 2FX is written to, any pending interrupt from another GPIB Adapter is allowed to occur, and that adapter may then be serviced.

> **Note:** Remember that the status bits in ISR1 and ISR2 automatically clear when the register is read. A software copy of these registers should be maintained.

Programming the Interrupt Controller

Programming information for the Interrupt Controller chip may be found in Intel's *Intel Component Data Catalog*. Additional information may also be found in technical references for your system.

> **Note:** At power-up time, the Interrupt Controller is initialized by the IBM PC BIOS program, which is in ROM on the system board. Programs written for processing GPIB Adapter interrupts must in no way change the overall configuration of the controller. Commands written to the controller should affect only the selected IRQ line.

The setup of the Interrupt Controller and the way it is used by the BIOS may be found in the BIOS program listing in your system's technical reference.

DMA Transfers

DMA transfers must be enabled through two hardware jumpers to one of the three available pairs of DMA transfer lines on the system's I/O channel.

DMA requests from the TLC are enabled using the DMAO and DMAI bits in IMR2. The TLC generates a DMA request under the same conditions that set the DO and DI bits in ISR1. A DMA request indicates that the TLC requires either a byte to be written to the CDOR or a byte to be read from the DIR. The 'DMA request' signal (DRQ) is cleared by a low on the $\overline{\text{DACK}}$.

The GPIB Adapter drives the selected DRQ line and enables the DACK line whenever the DMAO or DMAI bit is set in IMR2. Otherwise, the selected DRQ line is tri-stated, and the DACK lines are disabled.

DMA transfers are selected by the DMA Controller on the system board. The DMA Controller should never be enabled to respond to a DMA request from the GPIB Adapter when neither the DMAO nor DMAI bit is set in the TLC. This tri-states the adapter's DRQ line, which may look like a DMA request from the adapter.

Once made active, the DMA request line remains active until a DMA transfer occurs, or until a read from the DIR or a write to the CDOR occurs, depending on the direction of the DMA transfer.

The DMA terminal count (T/C) interrupt is enabled whenever the DMAO or DMAI bit is set and at least one of the interrupts internal to the TLC is enabled.

The DMA T/C interrupt is asserted when the DMAO or DMAI bit is set, and the system's I/O channel T/C line sends a high pulse during a DMA transfer to the adapter. A high pulse on the T/C line means the DMA Controller chip has reached terminal count (that is, the DMA Controller's byte count has gone from 0 to FFFF for the corresponding DMA channel). The DMA T/C interrupt is cleared when ISR2 is read.

The DMA T/C interrupt can be detected by reading the status register or channel byte-count register of the DMA Controller. The terminal-count bit corresponding to the GPIB Adapter's selected DMA channel is set in the DMA Controller's status register when the controller reaches terminal count for that DMA channel. All terminal-count bits are cleared when the controller's status register is read. The channel byte count should be FFFF unless the DMA Controller has been programmed to start that channel automatically.

Programming the DMA Controller

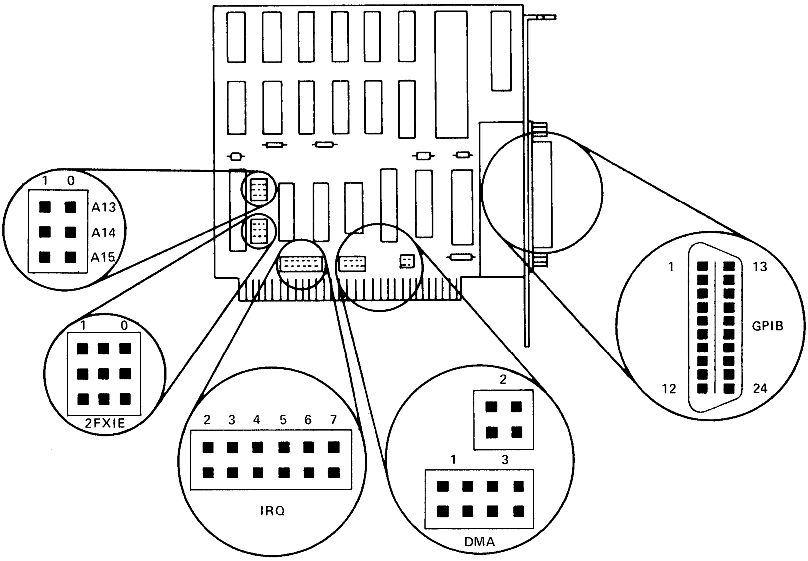
Programming information for the DMA Controller chip may be found in Intel's *Intel Component Data Catalog*. Additional information may also be found in the technical references for your system.

> **Note:** At power-up time, the DMA Controller is initialized by the IBM PC BIOS program, which is in ROM on the system board. Programs written for processing GPIB Adapter DMA transfers must in no way change the overall configuration of the controller. Commands written to the controller should affect only the selected DMA channel.

The setup of the DMA Controller and the way it is used by the BIOS may be found in the BIOS program listing in your system's technical reference.

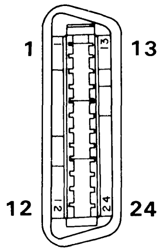
Interface

The following illustration shows the GPIB Adapter with its connectors.



Connector Specifications

The following table provides the pin numbers and their signals:



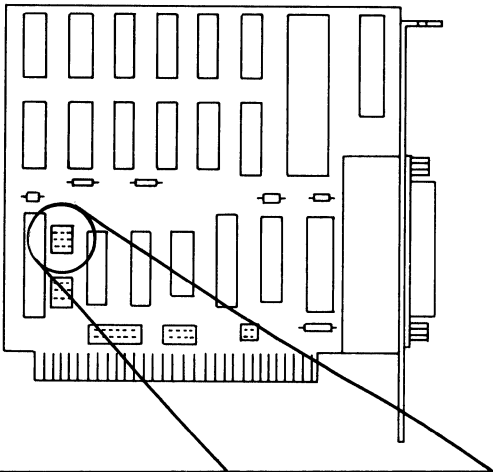
	Signal Name/Description	Pin	
	DIO 1	1	
	DIO 2	2	
	DIO 3	3	
	DIO 4	4	
	EOI (24)	5	
	DAV	6	
	NRFD	7	
	NDAC	8	
	IFC	9	
	SRQ	10	
	ATN	11	IBM GPIB Adapter
GPIB Device	SHIELD	12	
	DIO 5	13	
	DIO 6	14	
	DIO 7	15	
	DIO 8	16	
	REN (24)	17	
	Gnd, (6)	18	
	Gnd, (7)	19	
	Gnd, (8)	20	
	Gnd, (9)	21	
	Gnd, (10)	22	
	Gnd, (11)	23	
	Gnd, LOGIC	24	

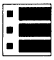
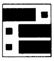
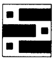
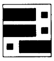
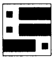
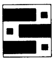
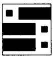
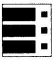
Jumper Positions

This section describes the jumper positions that may be selected when installing the adapter.

Adapter Selection

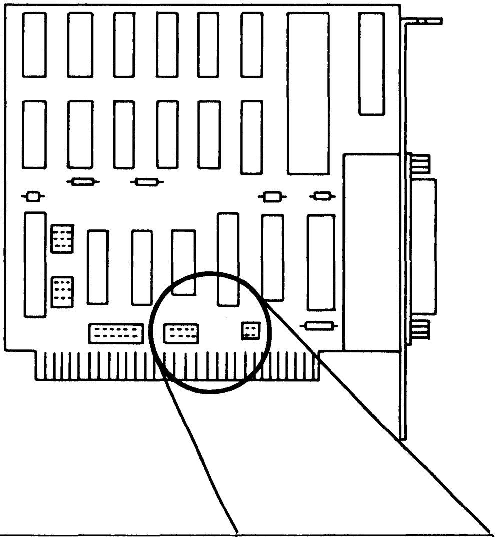
The following figure shows the jumper positions required for the adapter number of each adapter.

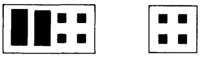
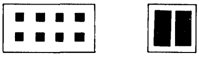
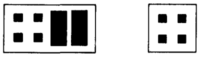


Adapter Number	Jumper Positions
0	
1	
2	
3	
4	
5	
6	
7	

DMA Channel Selection

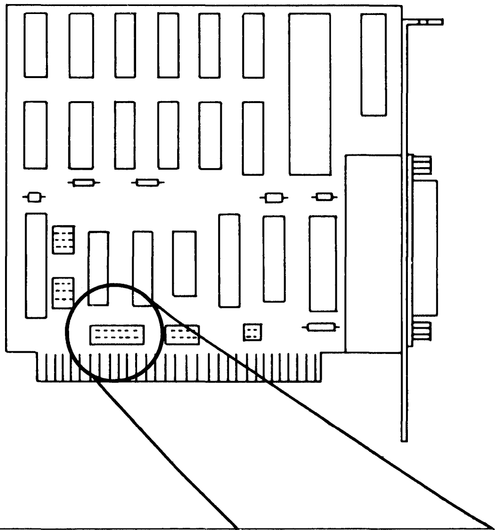
The following shows the jumper positions for selecting DMA channels.

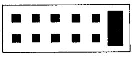
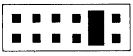
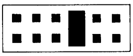
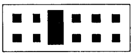
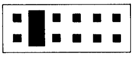
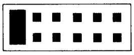


Direct-Memory Access Channel	Jumper Positions
1	
2	
3	

Interrupt-Request Selection

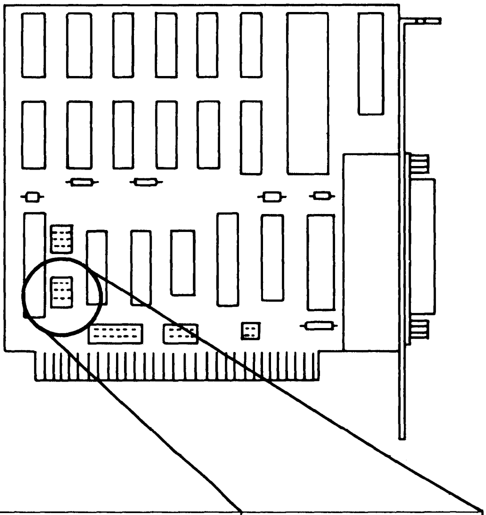
The following shows the jumper positions for selecting the interrupt-request line.

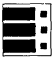
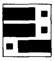
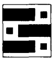
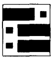
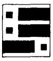
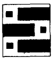


Interrupt-Request Level	Jumper Positions
7	
6	
5	
4	
3	
2	

Interrupt-Acknowledge Selection

The following shows the jumper positions for each interrupt-acknowledge level.



Interrupt-Acknowledge Level	Jumper Positions
7	
6	
5	
4	
3	
2	

Notes:

Specifications

Size:

- Length: 11.43 cm (4.5 in)
- Height: 10.67 cm (4.2 in)
- Thickness: 1.57 cm (0.62 in)
- Weight: 114.3 g (4.0 oz)

Power Requirements:

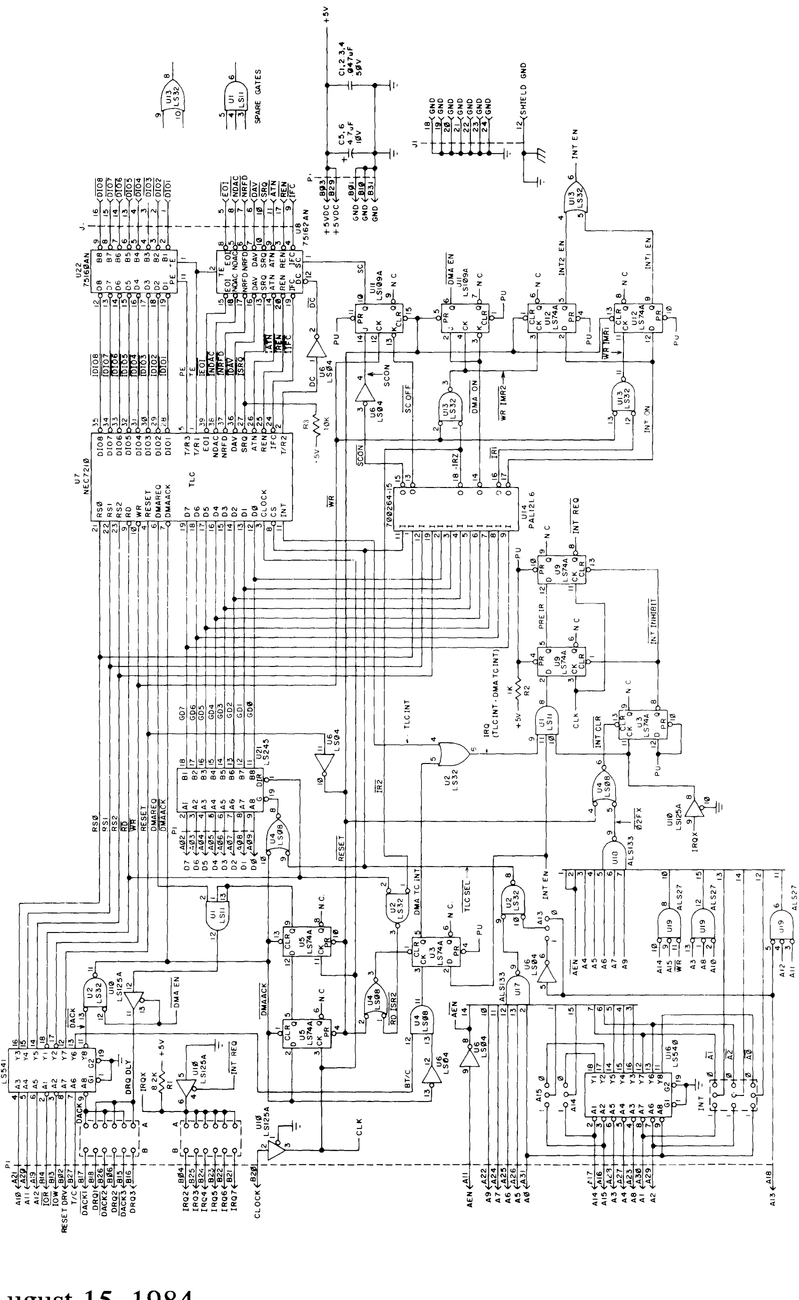
- Voltage: 5 Vdc (+/-5%)
- Current: 750 mA Max, 400 mA Typical

Power Dissipation: 3.75 Watts Max

Notes:

Logic Diagram

The following is the logic diagram of the GPIB Adapter.



August 15, 1984

Notes:

Index

Notes:

IBM

Personal Computer
Hardware Reference
Library

IBM Voice Communications Adapter

55X8866
October 11, 1985

Notes:

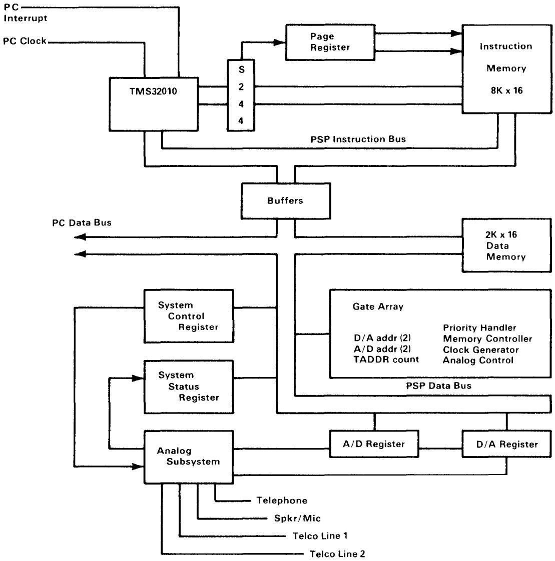
Contents

Description

The Voice Communications Adapter (henceforth called the Adapter), part number 6294771, serves as a versatile interface between the PC and analog signal sources. The Adapter can be configured for a wide variety of signal processing tasks. The Adapter uses a TMS32010 Programmable Signal Processor (PSP) with memory, interface logic, analog connections, and other support circuitry. A hardware interface is provided to synchronize the processor with the PC. This allows memory access of a common block of memory by both the Adapter and the host PC.

The Adapter plugs into the IBM Personal Computer PC, PC XT, or PC AT from which it derives all system signals and power. The Adapter can perform complex signal processing on signals within the audio frequency range. The adapter has connections for two telephone lines, a standard telephone, a microphone, and a speaker.

The following figure is a block diagram of the Adapter:



Functional Block Diagram

Hardware Circuit Description

The Adapter consists of the analog subsystem and the digital subsystem.

The analog section of the Adapter performs all of the interfacing between the Programmable Signal Processor (PSP) and the telephone lines, the telephone, the microphone, and the speaker. The analog subsystem contains the sample/hold circuits, the multiplexer/demultiplexer, operational amplifiers, analog line filters, and the analog to digital and digital to analog converters. The analog subsystem also drives the phone line, the handset, and the two auxiliary ports used for microphone and speaker connections.

The digital subsystem contains the PSP, the PSP instruction RAM, interface registers, bus controllers, and associated logic. It also contains memory and the circuitry that interfaces the PSP, the analog subsystem, and the host PC together, thus providing a common area accessible to all. All software resides in the PSP instruction RAM.

Functional Description

The Adapter Card is a programmable coprocessor with an analog back end connected to the host processor by shared memory. This coprocessor is a general purpose processor with special instructions used for signal processing. It has an instruction execution time of 200 nanoseconds, including 16 x 16 bit multiplies.

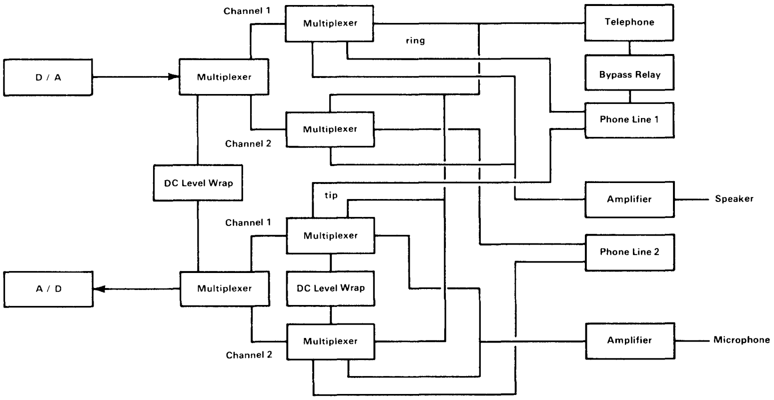
The Adapter contains two memories that are used by the PSP. The first memory is for program instructions, while the second memory is for sample data storage and for communication with the host PC. Both of these memories are accessible by the host PC processor although not at the same time. Each processor (either the PSP or the host PC) may interrupt the other with maskable interrupts. The PSP is controlled by the host processor. The PSP must have its instructions loaded before the Adapter will function properly.

The Adapter draws power (+/- 12 Vdc and + 5 Vdc) from the motherboard connector. Three ground lines are provided. Four connections to the maskable interrupt lines (2, 3, 4, and 7) are provided, the level of which is selectable by a jumper.

The analog subsystem on the Adapter consists of two full duplex audio channels. The rate the analog signal can be sampled (production rate) is selectable at 8000 Hz or 9600 Hz.

Both analog channels may be switched independently between the telephone, telephone line, speaker or microphone. The telephone and one telephone line can be connected together by the bypass relay, allowing normal telephone use. The second analog channel is normally connected to the second telephone line and is used for data communications, although it may be connected to the speaker or microphone.

The following diagrams show how the five analog connections (telephone lines 1 and 2, the telephone, the microphone, and the speaker) are connected:



Analog Functional Block Diagram

System Memory Interface

Overview

This section describes the software interface as seen by the host processor programmer.

The host interface consists of the I/O Control register (IOCR) located in I/O memory space and an 8K address range of host PC memory. The IOCR is used to enable the Adapter interrupts, reset the on-board processor, select memory mapping, and load Programmable Signal Processor instructions.

The IOCR controls the setup of the Adapter. The 8K of PC memory is used to pass programs and data back and forth between the host PC and the PSP.

Host Memory

The Adapter has 20K bytes of memory that are mapped into the host PC in two 8K groups and one 4K group. The mapping address (PC) is jumper selectable to start at DA000, D8000, DC000 or DE000.

The Adapter memory that may be mapped into the host PC is the PSP Data memory image, the PSP instruction memory (low 4K) or the PSP instruction memory (high 4K).

Note: PSP data memory is 16 bit words, while PC data is in bytes. Thus, only one-half of the instruction memory (4K) may be mapped (at one time) into 8K (bytes) of PC memory.

Programming Considerations

Interrupt Operation

Adapter Internal Interrupts

A single level of maskable interrupt is available on the PSP.

When a maskable interrupt is generated the INTF (interrupt flag) is set. This informs the processor that an interrupt needs to be serviced. At the start of the next available clock cycle, the processor clears the INTF flag and sets the INTM bit to prevent another interrupt service condition. The program counter is saved on the stack and then the program counter is loaded with address 0002. The program branches to whatever instruction is located at address 0002.

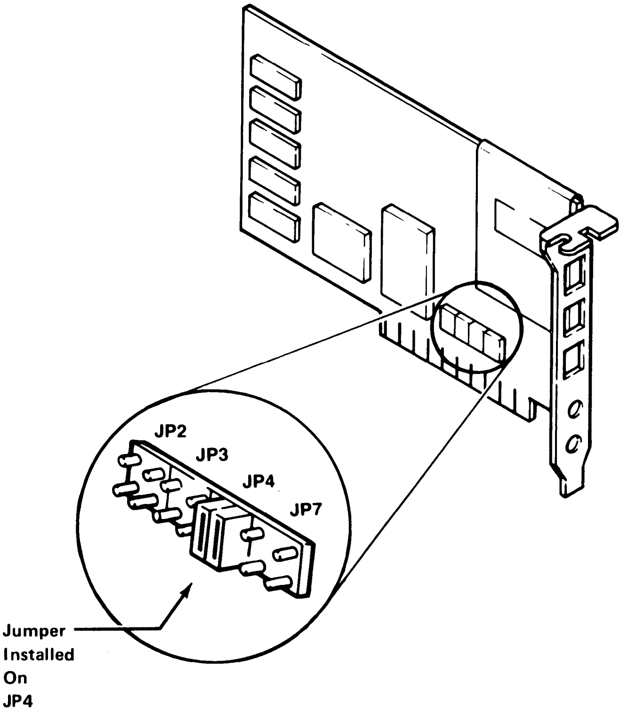
Note the distinction between the way interrupts are handled internal to the Adapter and the way interrupts are handled by the PC. The PC allows eight interrupts to be handled in priority sequence; the PSP allows one maskable interrupt.

The jumper block located on the Adapter maps the PSP buffered processor interrupt request (IRQ) into one of the host system interrupt lines. The level of interrupt passed from the card to the host is set by this jumper. A low to high transition on a host PC IRQ line tells the host PC that an interrupt exists on that line. When not active, the selected interrupt lines are held high by an 8.2KΩ pull-up resistor. Moving a jumper to a selected position connects the +IRQ line of the host to the +IRQ line of the PSP.

Interrupt Assignment, Adapter to PC

Interrupts are passed from the Adapter to the PC along a line to the interrupt handler chip on the PC. This level is jumper selectable on the Adapter to one of four levels: 2, 3, 4, or 7. A setting of 4 is recommended as it is used for communications Adapters on the PC.

The level of this interrupt is assigned as in the following diagram:



Jumper Positions

Interrupt Assignment, PC to Adapter

The host PC interrupts the PSP by setting bit 3 of the I/O Control Register (IOCR). This sets an interrupt pending to the PSP in status register SS1 (SS1 bit 6). Once the host PC has set bit 3 of the IOCR, this bit cannot be reset by the PC. When the PSP reads SS1, this bit is cleared, showing the host that the signal processor has acknowledged its interrupt.

The PC may reset the PSP by clearing bit 0 in the IOCR, immediately stopping the PSP.

Note: A system reset via CTRL-ALT-DEL will not reset the PSP.

The TMS32010

The TMS32010 Microprocessor is the Programmable Signal Processor (PSP) for the Adapter. This is a separate processor that works independently of the host PC, and consists of the following:

- A 12 bit address bus
- 4 I/O control lines
- 2 Auxiliary Registers
- Status Register
- Structured Interrupt handling
- 16 bit data bus
- Volatile 144 word x 16 bit read/write on-chip data memory
- Auto-incrementing/decrementing registers for data addressing and loop counting.
- On-chip oscillator
- 20 MHz clock, 200ns instruction cycle time

The processor can execute one instruction while fetching the next instruction and the data that it is to operate on. Because of both data prefetch and address pre-calculation, fast access and processing times are possible (up to five million instructions per second).

The TMS32010 can directly address up to 4K x 16 bit words. However, an I/O register has been provided (external to the PSP) so that 8K instructions may be addressed, logically divided as four 2K pages. Writing to I/O address 0 loads an extrenal register called the page register.

Writing to I/O address 1 loads an external register called the TADDR. Once this register has been loaded it can be used as a pointer into the shared memory area (using I/O address 2). An I/O write to address 2 will store data at the location pointed to by the TADDR. An I/O read from ADDR2 will read data stored at the location pointed to by the address 2.

The TADDR has an auto-increment or auto-decrement feature that allows it to read or write successive memory locations without reloading the TADDR with an I/O 1 instruction. This auto increment/decrement feature is controlled by bits 6 and 7 in the system control register 1 (SC1, or writing to I/O location 4).

Notes:

Interfaces

The I/O lines on the PSP Adapter consist of PC bus lines and Adapter edge external I/O lines. The edge connectors are used for analog interfacing and the bus connector is used for digital communications.

Bus Connections

The PC bus lines that are supported are:

- Data lines 0 - 7
- Data Address lines 0-15
- (I/O Read) IOR
- (I/O Write) IOW
- Static Memory Read (MEMR)
- Static Memory Write (MEMW)
- Clock
- Address Enable (AEN)
- Reset Drv
- I/O Channel Ready
- IRQ 2, 4, 6, 7
- +/- 12 Vdc

- Ground

- +5 Vdc

The Adapter uses the I/O Channel Ready line to cause the host processor to wait while the PSP is reading or writing data in Adapter memory. If the PSP is reset, or if the I/O register space is being accessed, the I/O Channel Ready line is not pulled low.

Input Output Control Register

The I/O Control Register (IOCR) is defined as follows:

IOCR Bit	Name
0	Enable PSP
1	Select instruction memory high/low
2	Select instruction memory
3	Interrupt PSP
4 - 5	Reserved
6	Enable interrupts
7	Interrupt status to host

The power-on state of the IOCR is hex 06.

The meaning of each of these bits is as follows:

Enable PSP

When clear, this bit holds the PSP in the reset state. It is cleared on power up of the Adapter. When set, the PSP begins execution at location 0 in instruction memory.

Select Instruction Memory High/Low

This bit selects which 4K x 16 block of PSP instruction memory is mapped to the PC bus. When clear, this bit selects the first 4K. When set, this bit maps the second 4K.

Note: This bit is used only when the select instruction memory bit (bit 2) is also set.

Select Instruction Memory

When clear, this bit maps the shared data memory to the PC Bus. When set, this bit maps the PSP instruction memory to the PC Bus.

Note: The PSP must be in the reset state to read or write instruction memory (Bit 0 is clear). This prevents the PC from changing a program while the PSP is running. When bit 0 is set, the memory control is automatically set to the shared memory.

Interrupt PSP

When set, this bit generates both an interrupt status bit in the PSP status register SS1 and an interrupt pending signal to the PSP. When the PSP reads the SS1 register, the interrupt condition is cleared and this bit is reset.

Note: Once set, this bit cannot be cleared by the host PC.

Enable Interrupts

When clear, this bit degates the PSP to host interrupt from the system interface. When set, this bit allows the PSP to interrupt the host processor.

Interrupt Status (to host)

When set, this bit indicates that the PSP is trying to interrupt the host processor. It is set by the PSP but must be reset by the host. This bit is reset when the host PC reads the IOCR. The PSP can read this bit to detect when the host resets it, providing an interrupt acknowledge function.

Analog DMA Control

The analog subsystem contains four separate DMA address registers that control where the analog channels access data memory, and when it interrupts the PSP so that the acquired samples may be processed (or digital samples output). Each address counter is 12 bits wide, of which 8 bits are fixed in value and 4 bits are used as an upcounting value.

When enabled, the address counter increments after each data memory access (that is, after collection of each sample point). When the 4 LSBs of an analog channel's counter wrap (from F to 0), that channel raises an interrupt request to the PSP. Thus, the PSP is interrupted every 16 samples per channel.

There is no carry from bit 3 to bit 4, allowing register reuse without zeroing. The 8 bits of the fixed value address need to be adjusted through software for longer sample buffering..

Note: Analog channels always run after reset. The enable bits in SC2 control the DMA channels send or fetch data to the data memory only; they do not turn the A/D or D/A converters on or off.

Each analog channel has its own outboard multiplexer under program control. An inactive D/A section may be set to the wrap position, preventing undesired analog output.

The format of this data is shown in the following diagram:

Bit Range	Usage
0 - 3	4-bit incrementing address
4 - B	8-bit fixed address
C - D	Channel Selection
E - F	Reserved

The power on state of this register is indeterminate.

DMA channels are loaded with data memory addresses. The channel is enabled by setting the corresponding enable bit in control register 2 (one bit is set for each channel). An I/O write to PSP I/O address 3 loads the proper DMA address counter with this address. Bits 12 through 14 specify which DMA channel is loaded.

Once 16 DMA cycles are completed for a channel (that is, after 16 samples are processed) an interrupt status bit is set in SS1 and an interrupt is generated to the PSP. DMA cycles continue for the interrupting channel(s) until the channel is disabled (by clearing its enable bit in SC2). Reading the SS1 register resets both the interrupt pending and the interrupt status bits.

Channel selection works as shown in the following diagram:

Bits C D	Usage
00	D/A converter 1
01	D/A converter 2
10	A/D converter 1
11	A/D converter 2

Analog Interface Description

Analog data is stored in Adapter external data memory until it can be processed by a converter.

A/D Channel Data

Data stored by the analog subsystem in data memory is formatted as shown in the following diagram:

Bit range	**Meaning**
0 - A	Data value
B	Sign bit
C - F	Sign bit extended

Data values have a range as shown in the following diagram:

Word Value	**Decimal Equivalent**	**Meaning**
FFFF	– 1	– Millivolts
07FF	+ 2047	+ full scale (+ 5 volts)
F800	– 2048	– full scale (– 5 volts)
0000	0	0 volts

D/A Channel Data

The data read by the analog subsystem from data memory is formatted as shown in the following diagram:

Bit Range	Meaning
0 - A	Data value
B	Sign bit
C - F	ignored

Data values are as shown in the following diagram:

Word Value	Decimal Equivalent	Meaning
FFFF	− 1	− Millivolts
7FF	+ 2047	+ full scale (+ 5 volts)
F800	− 2048	− full scale (− 5 volts)
0000	0	0 volts

The hardware ignores the upper 4 bits of data when loading the D/A converter. Bit 11 is loaded into the DAC and is interpreted as a sign bit.

Input Output Lines

The IOCR address is selectable (by jumpers S1 and S2) to any of 4 addresses as shown in the following diagram:

PC I/O Address	Meaning
021F	PSP I/O Control Register
221F	Alternate address for IOCR
421F	Alternate address for IOCR
621F	Alternate address for IOCR
D8000	221F
DA000	021F
DC000	421F
DE000	621F

Input Output Address Map

The following diagram shows the I/O addresses selected by the interrupt jumper block.

PC Memory Address	PSP I/O Address
02F2	IRQ 2 lockout Reset
02F3	IRQ 3 lockout Reset
02F4	IRQ 4 lockout Reset
02F7	IRQ 7 lockout Reset

After an interrupt has occurred from the Adapter to the PC, interrupts must be enabled. The IOCR must be read by the PC, and then an I/O write must be performed to the address listed above (the data written is not important).

PSP Programming Interfaces

The following describes the programming interface from the PSP.

PSP I/O Addressing Map

PSP I/O Address	Meaning
0	Instruction Page Register
1	TADDR Counter
2	Data Memory (at TADDR)
3	Analog DMA load
4	System Control Register 1 (SC1)
5, on write	System Control Register 2 (SC2)
5, on read	System Status Register (SS1)
6-7	Reserved

PSP Instruction Paging

A paging scheme allows 8K of instructions to be addressed in external PSP instruction storage. Page selection is controlled by a two bit register, the Instruction Page Register (IPR).

The IPR is used only when address bit 11 on the PSP is set. If bit 11 is clear, then this register is ignored and only the first 2 bit page is addressed. Thus, page 0 is always addressable for interrupt servicing.

Pagination works as shown in the following diagram:

IPR Setting	Range Selected
00	1st 2K
01	2nd 2K
10	3rd 2K
11	4th 2K

PSP Address Register (TADDR)

The PSP has 144 words of data memory internal to the chip. The 2K x 16 words of shared memory is external to the chip, not directly accessible to the processor.

Access to this external data memory is via an external address register known as the TADDR (TMS32010 address register). It is loaded by writing to PSP I/O address 1. Once loaded, it is used as a pointer to data in the external data memory using PSP I/O address 2. Subsequent reads of address 2 fetch the data in external data memory pointed to by TADDR. Similarly, subsequent I/O writes to address 2 store data in external data memory at the location specified by the TADDR address register. The TADDR automatically increments/decrements after each read or write, depending on the control bit settings in SC1.

PSP Control Register (SC1)

This register is as shown in the following diagram:

Bit	Use
0	Raise host interrupt line
1-4	Reserved
5	Analog sample rate high/low
6	TADDR control A
7	TADDR control B
8 - F	Reserved

The power-up value of this register is 0.

The meaning of each of these bits is as follows:

Raise Host Interrupt Line

If this bit is set, the adapter raises the host interrupt line. This bit is set by the PSP and cleared by the PC host processor, indicating that the PC has acknowledged this interrupt. The PSP can read this bit to see if it has been reset by the host PC processor.

Once set by the PSP, it cannot be reset by the PSP. This bit can only be reset by the host PC reading the IOCR.

Analog Sample Rate High /Low

When this bit is clear, the analog sample rate is 8000 Hz. When this bit is set, the rate is 9600 Hz.

TADDR Control A and B

TADDR has an auto-increment/decrement feature, enabling successive reads or writes to the data memory without reloading TADDR with an I/O 1 instruction. This is controlled by bits 6 and 7 in system control register 1 (SC1, I/O address 4). Combinations of auto-increment/decrement possibilities are as shown in the following diagram:

Bit Value	Meaning
00	No change
01	Decrement
10	Increment
11	No change

PSP Control Register (SC2)

Bit	Use
0	Enable A/D channel 2
1	Enable A/D channel 1
2	Enable D/A channel 2
3	Enable D/A channel 1
4 - 5	Channel 1 receive gain
6 - 7	Channel 2 receive gain
8	Bypass relay select
9	Telephone line 1 off-hook
A	Telephone line 2 off-hook
B	DC Wrap
C	Channel 1 multiplexer control 1
D	Channel 1 multiplexer control 2
E	Channel 2 multiplexer control 1
F	Channel 2 multiplexer control 2

The power on state of this register is 0. The meaning of each of these bits is as follows:

Enable A/D Channel 1

When set, this bit stores analog samples at the location set in the A/D channel 1 address register. For each sample, the address is incremented. When incremented from F to 0 an interrupt bit for channel 1 is set in status register SS1.

Enable A/D Channel 2

When set, this bit stores analog samples at the location set in the A/D channel 2 address register. For each sample, the address is incremented. When incremented from F to 0 an interrupt bit for channel 2 is set in status register SS1.

Enable D/A Channel 1

When this bit is set, the hardware for channel 1 fetches digital words from data at the location set in the D/A channel 1 address register. After each fetch, the sample address is incremented. When this bit is incremented from F to 0 an interrupt bit for channel 1 is set in status register SS1.

Enable D/A Channel 2

When this bit is set, the hardware for channel 2 fetches digital words from data at the location set in the D/A channel 2 address register. After each fetch, the sample address is incremented. When incremented from F to 0 an interrupt bit for channel 2 is set in status register SS1.

Channel 1 Receive Gain

This bit sets the receive amplifier gain for the telephone line 1 interface to increase the level of the input signal. This is implemented completely in analog hardware. There are four possibilities:

Bits 5 4	Gain (approximate)
00	3 (this is the default setting)
01	12
10	18
11	24

Channel 2 Receive Gain

Sets the receive amplifier gain for the telephone line 2 interface to increase the level of the input signal. This is implemented completely in analog hardware. There are four possibilities:

Bits 7 6	Gain (approximate)
00	3 (this is the default setting)
01	12
10	18
11	24

Bypass Relay Select

When this bit is set, the bypass relay disconnects the telephone line 1 from the telephone and connects the telephone to the Adapter. The telephone line 1 remains connected to the Adapter. When this bit is reset, the line is reconnected to the phone.

When the Adapter is powered off, this relay closes and connects the telephone to telephone line 1. This allows telephone use when the PC is off.

Telephone Line 1 Off-Hook

When this bit is set, the PLI for line 1 goes off-hook.

Telephone Line 2 Off-Hook

When this bit is set, the PLI for line 2 goes off-hook.

DC Wrap

This connects the output of the D/A converter into the A/D converter. Values sent to the D/A convertor appear on the receive side after conversion by the A/D convertor. This feature provides a DC path between the D/A and A/D converters for diagnostic routines.

Channel 1 Multiplex Control 1 and 2

Controls connections to analog channel 1 for both the A/D (receive) and the D/A (transmit) functions. There are four possibilities:

Bits D C	Meaning
00	Phone line interface 1 (transmit, receive)
01	Telephone (transmit, receive)
10	Microphone (receive), speaker (transmit)
11	AC Wrap (transmit, receive)

Channel 2 Multiplex Control 1 and 2

Controls connections to analog channel 1 for both the A/D (receive) and the D/A (transmit) functions. There are four possibilities:

Bits F E	Meaning
00	Phone line interface 2 (transmit, receive)
01	Telephone (transmit, receive)
10	Microphone (receive), speaker (transmit)
11	AC Wrap (transmit, receive)

When both channels are set to the AC wrap mode the two channels are connected together at a point after the bandpass filters. This function is used by the diagnostics. When a channel is not being used it should be set to AC wrap to prevent undesired analog output.

PSP Status Register (SS1)

Bit	Use
0	A/D interrupt channel 2
1	A/D interrupt channel 1
2	D/A interrupt channel 2
3	D/A interrupt channel 1
4 - 5	Reserved
6	Host interrupt pending
7	Reserved
8 - A	Host interrupt jumper level
B - C	Reserved
D	Telephone line 1 ring indicate
E	Telephone line 2 ring indicate
F	Cradle on-hook

The power on state of this register equals the value of the inputs.

The bits have the following meaning:

A/D Interrupt Channel 2

When this bit is set, the DMA control for channel 2 has wrapped from F-0, indicating the collection of 16 analog samples, interrupting the PSP. Reading this register resets both this bit and the interrupt condition. DMA is not suspended.

A/D Interrupt Channel 1

When this bit is set, the DMA control for channel 1 has wrapped from F-0, indicating the collection of 16 analog samples, interrupting the PSP. Reading this register resets both this bit and the interrupt condition. DMA is not suspended.

D/A Interrupt Channel 2

When this bit is set, the DMA control for channel 2 has wrapped from F-0, indicating the conversion of 16 digital samples, interrupting the PSP. Reading this register resets both this bit and the interrupt condition. DMA is not suspended.

D/A Interrupt Channel 1

When this bit is set, the DMA control for channel 1 has wrapped from F-0, indicating the conversion of 16 digital samples, interrupting the PSP. Reading this register resets both this bit and the interrupt condition. DMA is not suspended.

Host Interrupt Pending

When this bit is set, the host PC has set the bit in the IOCR, requesting an interrupt of the PSP. Reading this register clears the PSP interrupt and this bit in both the SS1 and in the IOCR. The host PC can then detect that the PSP has responded to its interrupt.

Host Interrupt Jumper Level

These bits indicate how host interrupts are jumpered on the Adapter. Only the PSP can read these bits in the Status register. This allows host PC software to chose an interrupt vector for the Adapter. Possible values are as follows:

Bits A 9 8	Meaning
000	Jumper not installed
010	Interrupt level 2
011	Interrupt level 3
100	Interrupt level 4
111	Interrupt level 7

Telephone Line 1 Ring Indicate (bit D)

This bit is cleared when a ringing voltage appears on line 1.

Telephone Line 2 Ring Indicate (bit E)

This bit is cleared when a ringing voltage appears on line 2.

Cradle On-Hook (bit F)

If this bit is set, the cradle is on-hook. If this bit is clear, the telephone cradle is off-hook.

Notes:

Specifications

Electrical

The Adapter has the following specifications:

Power Type	**- 12 Vdc**	**+ 12 Vdc**	**+ 5 Vdc**	**Units .**
Tolerance	+/- 10	+/- 5	+/- 5	%
Ripple	100	100	100	mv*P-P
Current Maximum	0.090	0.110	1.50	Amps
Current Typical	0.080	0.100	1.20	Amps

Analog Specifications

The Adapter has five edge external lines that are used as an analog interface.These lines consists of:

1. Telephone Line 1

   Connected to the public telephone switching network Tip and Ring signals.

   - 600 ohm audio lines.
   - Output level 0 dbm maximum 300 - 3600 Hz.

- Ringer type B supported (15.3 - 68 Hz @ 40 - 150 Volts Root Mean Squared)

- Ringer Equivalent <0.2

2. Telephone Line 2 Tip and Ring

   - 600 ohm audio lines.

   - Output level 0 dbm maximum

   - Ringer type B supported (15.3 - 68 Hz @ 40 - 150 V RMS)

   - Ringer Equivalent <0.2

3. Telephone Instrument Tip and Ring

   - 150 ohm audio lines.

   - Supports Bell Telephone Set type 2500, or 500.

   - Supports Bell compatible speaker phone.

4. External Speaker (female sub-miniature phono jack)

   - 8 ohm audio pair.

   - Output power 0.5 watt maximum

5. External Microphone (female sub-miniature phono jack)

   - Audio pair

   - 10K ohm impedance

   - Peak Input Voltage = 16.7 mV

Note: When a telephone is connected to the telephone jack and the Adapter is in bypass mode, the line 1 ringer equivalent is the sum of both the telephone and the Adapter line 1 ringer equivalent.

Operating Conditions

The Adapter must be used under the following environmental conditions:

Operating temperature:	15.6 C to 32.2 C
Shipment temperature	-40 C to 60 C
Storage temperature:	0.6 C to 60 C
Operating humidity:	8% to 80%
Shipment humididy	5% to 100%
Storage humidity:	5% to 80%
Maximum wet-bulb temperature:	22.8 C
FCC Class:	B

FCC Requirements and Telephone Line Protection

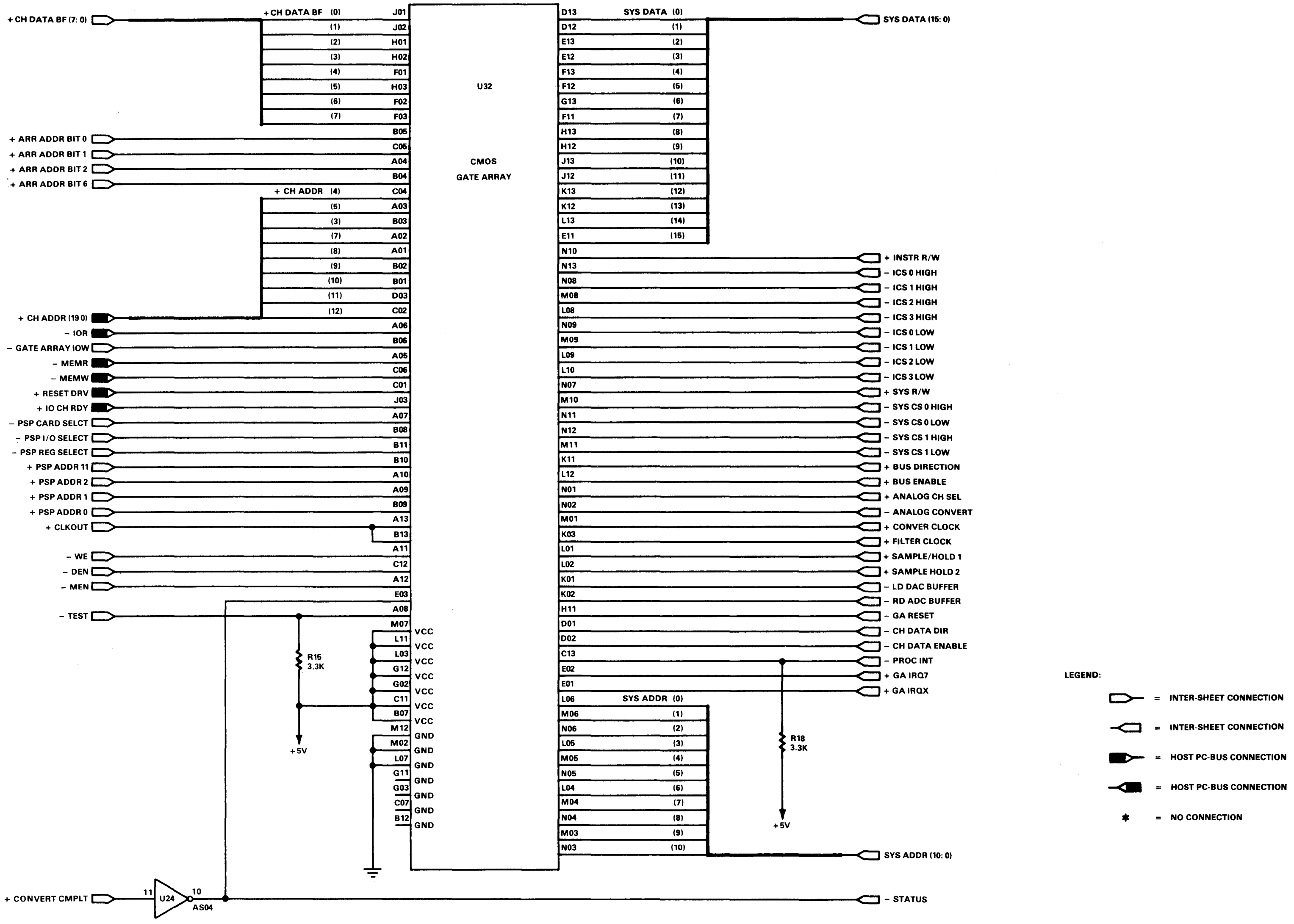
The Adapter software must monitor all outbound data to the telephone line. If the outbound data exceeds the maximum energy allowed by the FCC for a 3-second averaged period, the data should be attenuated.

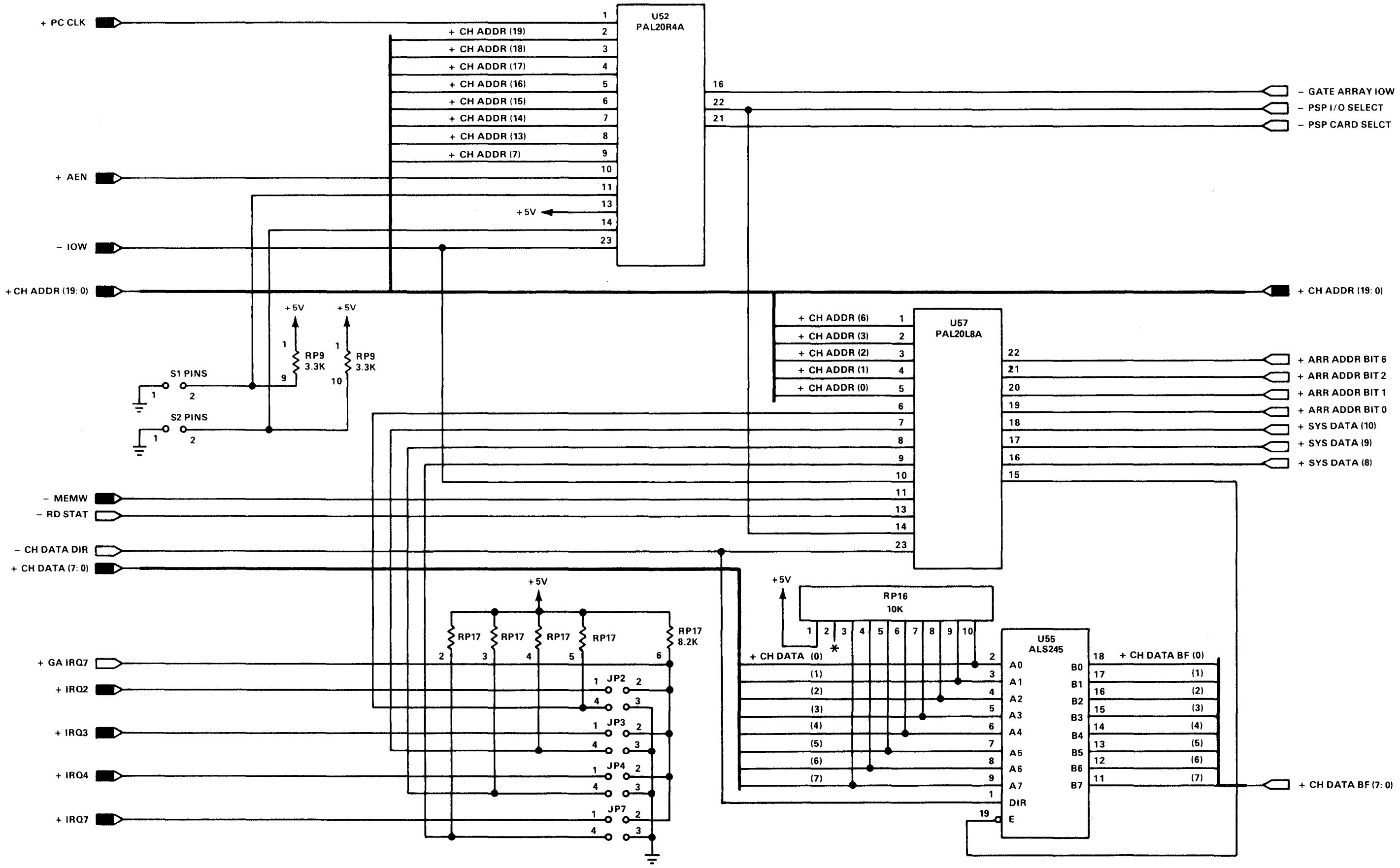
The Adapter must perform a 2-second billing delay. Outbound data should be forced to silence for 2 seconds after the phone line has gone off-hook on all incoming calls.

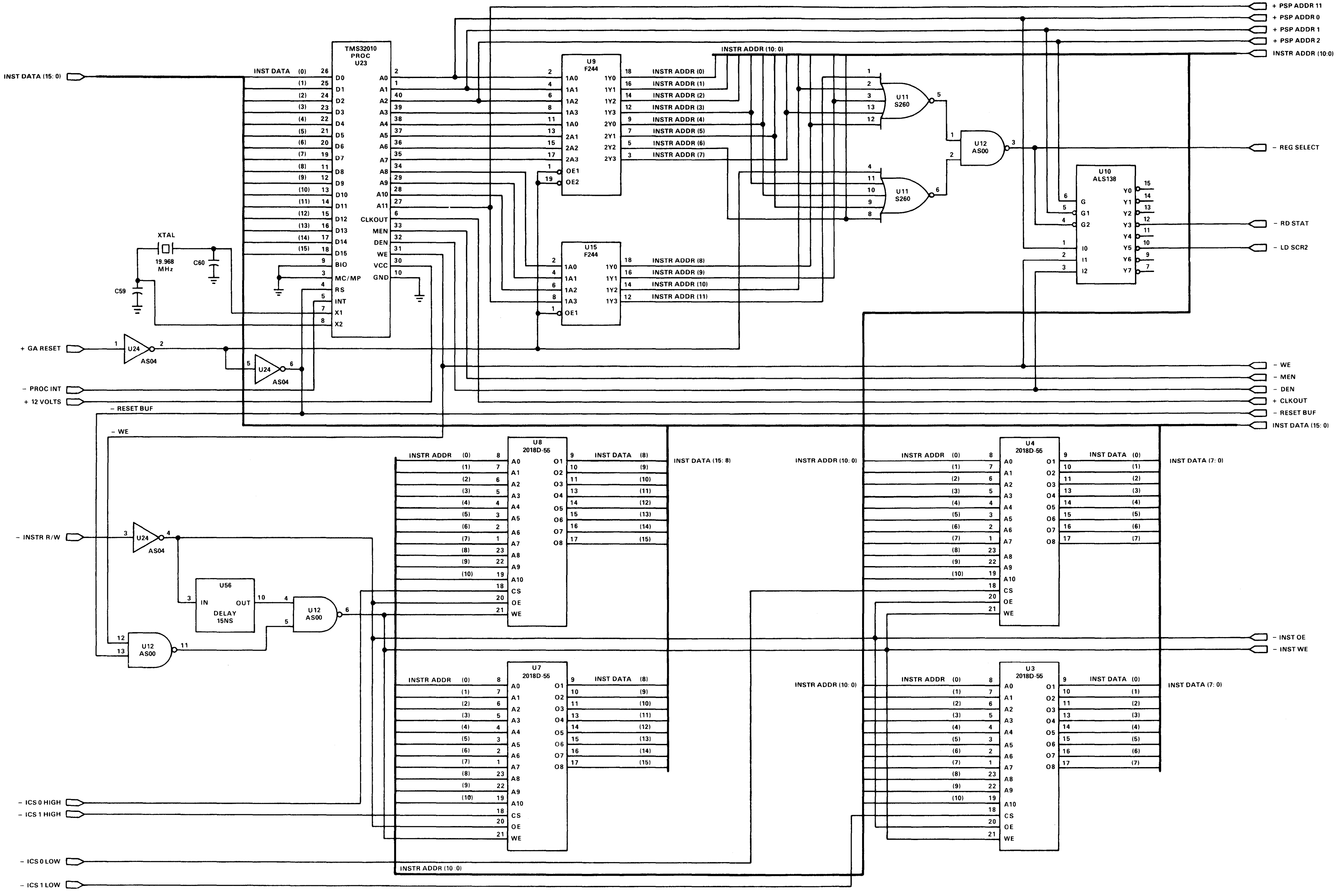
Logic Diagrams

The following pages contain the logic diagrams of the Adapter.

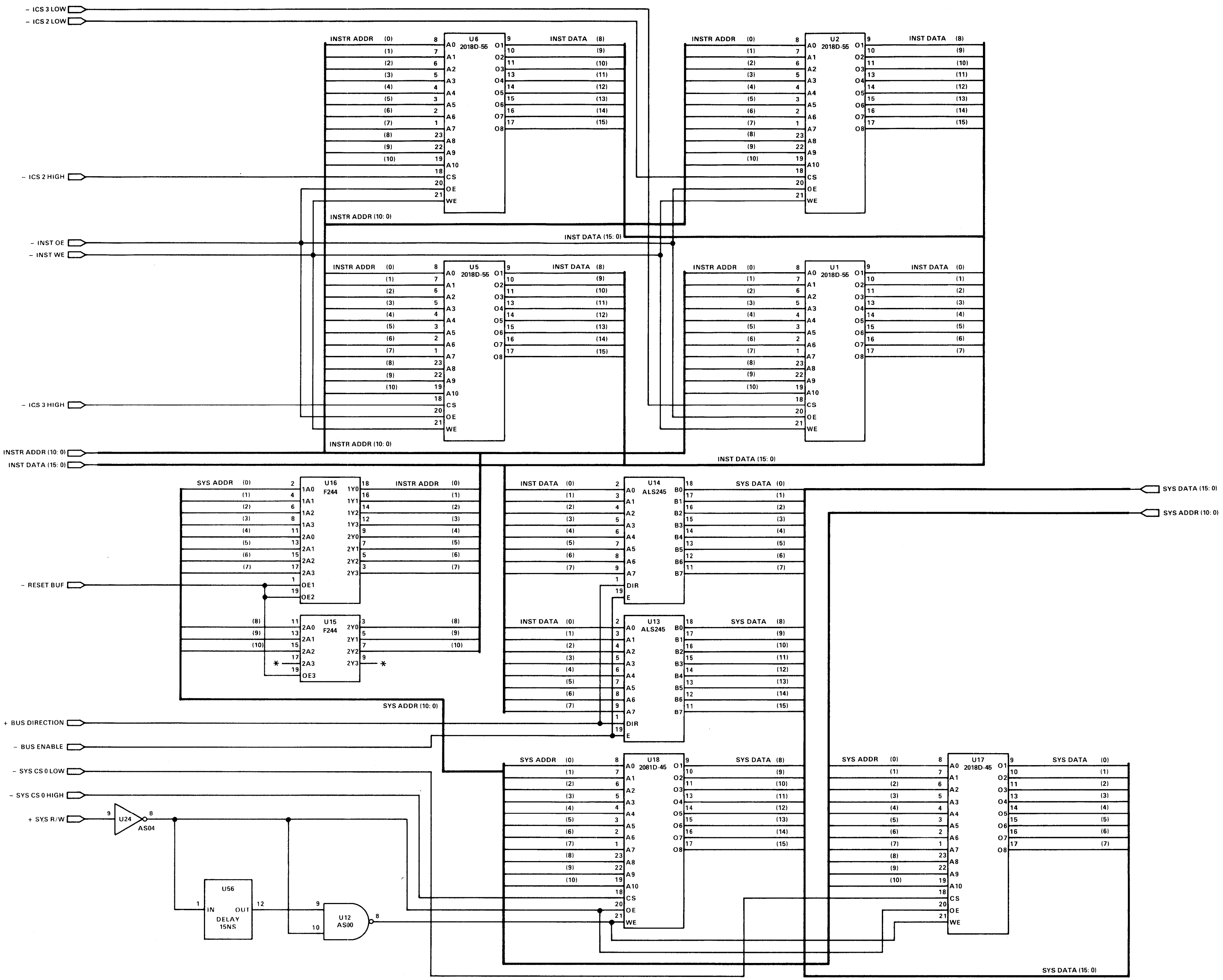
55X8868


+ CH DATA BF (7: 0)
+ CH DATA BF (0) J01
(1) J02
(2) H01
(3) H02
(4) F01
(5) H03
(6) F02
(7) F03
B05
+ ARR ADDR BIT 0 C05
+ ARR ADDR BIT 1 A04
+ ARR ADDR BIT 2 B04
+ ARR ADDR BIT 6
+ CH ADDR (4) C04
(5) A03
(3) B03
(7) A02
(8) A01
(9) B02
(10) B01
(11) D03
(12) C02
+ CH ADDR (19 0)
A06
– IOR B06
– GATE ARRAY IOW A05
– MEMR C06
– MEMW C01
+ RESET DRV J03
+ IO CH RDY A07
– PSP CARD SELCT B08
– PSP I/O SELECT B11
– PSP REG SELECT B10
+ PSP ADDR 11 A10
+ PSP ADDR 2 A09
+ PSP ADDR 1 B09
+ PSP ADDR 0 A13
+ CLKOUT B13
A11
– WE C12
– DEN A12
– MEN E03
A08
– TEST
M07 VCC
L11 VCC
L03 VCC
G12 VCC
G02 VCC
C11 VCC
B07 VCC
M12 GND
M02 GND
L07 GND
G11 GND
G03 GND
C07 GND
B12 GND
R15
3.3K
+5V
U32
CMOS
GATE ARRAY
+ CONVERT CMPLT
11
U24
10
AS04
D13 SYS DATA (0)
D12 (1)
E13 (2)
E12 (3)
F13 (4)
F12 (5)
G13 (6)
F11 (7)
H13 (8)
H12 (9)
J13 (10)
J12 (11)
K13 (12)
K12 (13)
L13 (14)
E11 (15)
SYS DATA (15: 0)
N10 + INSTR R/W
N13 – ICS 0 HIGH
N08 – ICS 1 HIGH
M08 – ICS 2 HIGH
L08 – ICS 3 HIGH
N09 – ICS 0 LOW
M09 – ICS 1 LOW
L09 – ICS 2 LOW
L10 – ICS 3 LOW
N07 + SYS R/W
M10 – SYS CS 0 HIGH
N11 – SYS CS 0 LOW
N12 – SYS CS 1 HIGH
M11 – SYS CS 1 LOW
K11 + BUS DIRECTION
L12 + BUS ENABLE
N01 + ANALOG CH SEL
N02 – ANALOG CONVERT
M01 + CONVER CLOCK
K03 + FILTER CLOCK
L01 + SAMPLE/HOLD 1
L02 + SAMPLE HOLD 2
K01 – LD DAC BUFFER
K02 – RD ADC BUFFER
H11 – GA RESET
D01 – CH DATA DIR
D02 – CH DATA ENABLE
C13 – PROC INT
E02 + GA IRQ7
E01 + GA IRQX
L06 SYS ADDR (0)
M06 (1)
N06 (2)
L05 (3)
M05 (4)
N05 (5)
L04 (6)
M04 (7)
N04 (8)
M03 (9)
N03 (10)
R18
3.3K
+5V
SYS ADDR (10: 0)
– STATUS
LEGEND:
= INTER-SHEET CONNECTION
= INTER-SHEET CONNECTION
= HOST PC-BUS CONNECTION
= HOST PC-BUS CONNECTION
✱ = NO CONNECTION

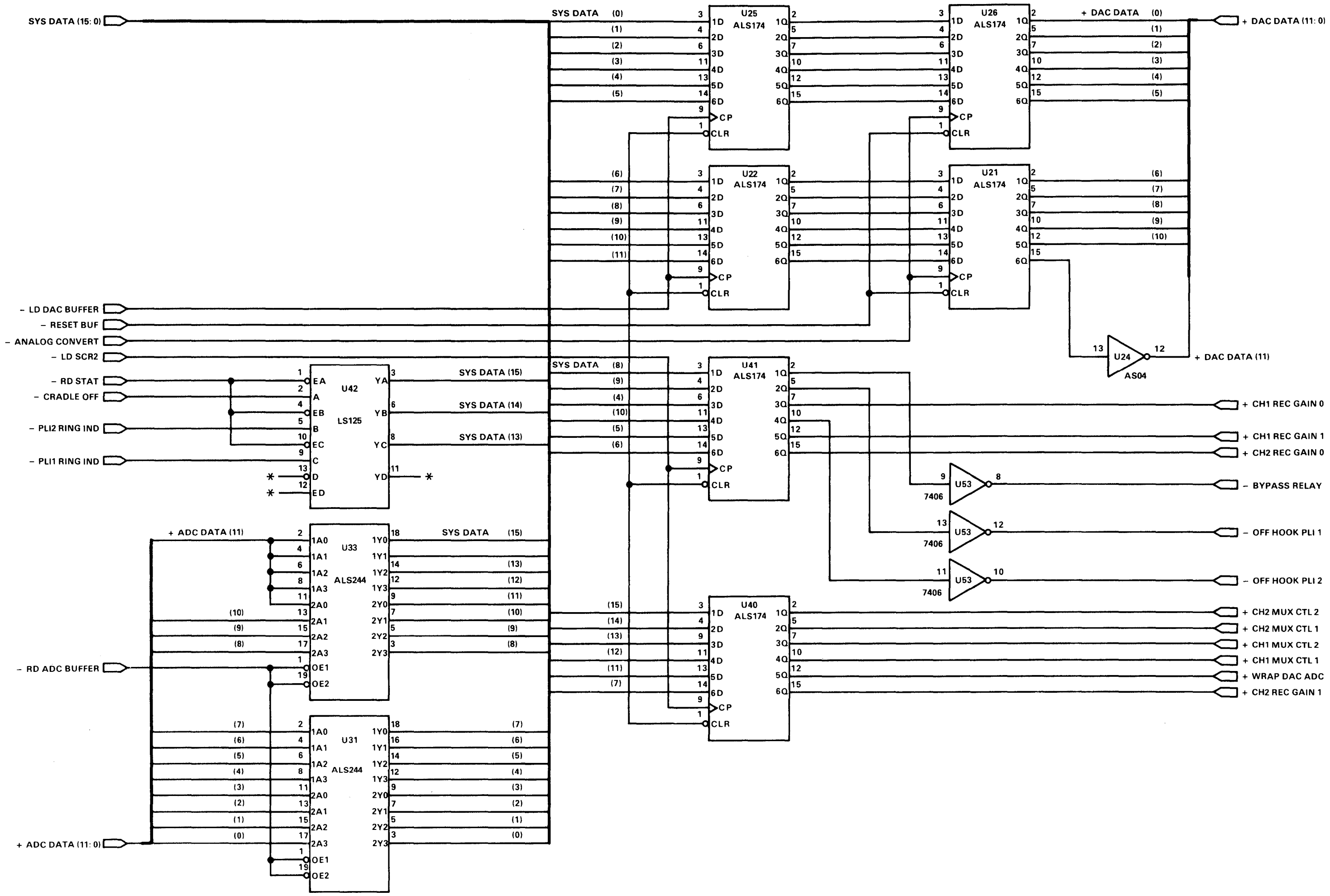

+ PC CLK
+ AEN
− IOW
+ CH ADDR (19: 0)
U52
PAL20R4A
+ CH ADDR (19)
+ CH ADDR (18)
+ CH ADDR (17)
+ CH ADDR (16)
+ CH ADDR (15)
+ CH ADDR (14)
+ CH ADDR (13)
+ CH ADDR (7)
+5V
− GATE ARRAY IOW
− PSP I/O SELECT
− PSP CARD SELCT
+ CH ADDR (19: 0)
RP9
3.3K
RP9
3.3K
S1 PINS
S2 PINS
U57
PAL20L8A
+ CH ADDR (6)
+ CH ADDR (3)
+ CH ADDR (2)
+ CH ADDR (1)
+ CH ADDR (0)
+ ARR ADDR BIT 6
+ ARR ADDR BIT 2
+ ARR ADDR BIT 1
+ ARR ADDR BIT 0
+ SYS DATA (10)
+ SYS DATA (9)
+ SYS DATA (8)
− MEMW
− RD STAT
− CH DATA DIR
+ CH DATA (7: 0)
+ GA IRQ7
+ IRQ2
+ IRQ3
+ IRQ4
+ IRQ7
RP17
RP17
RP17
RP17
RP17
8.2K
JP2
JP3
JP4
JP7
RP16
10K
*
U55
ALS245
+ CH DATA (0)
+ CH DATA BF (0)
A0
A1
A2
A3
A4
A5
A6
A7
DIR
E
B0
B1
B2
B3
B4
B5
B6
B7
+ CH DATA BF (7: 0)

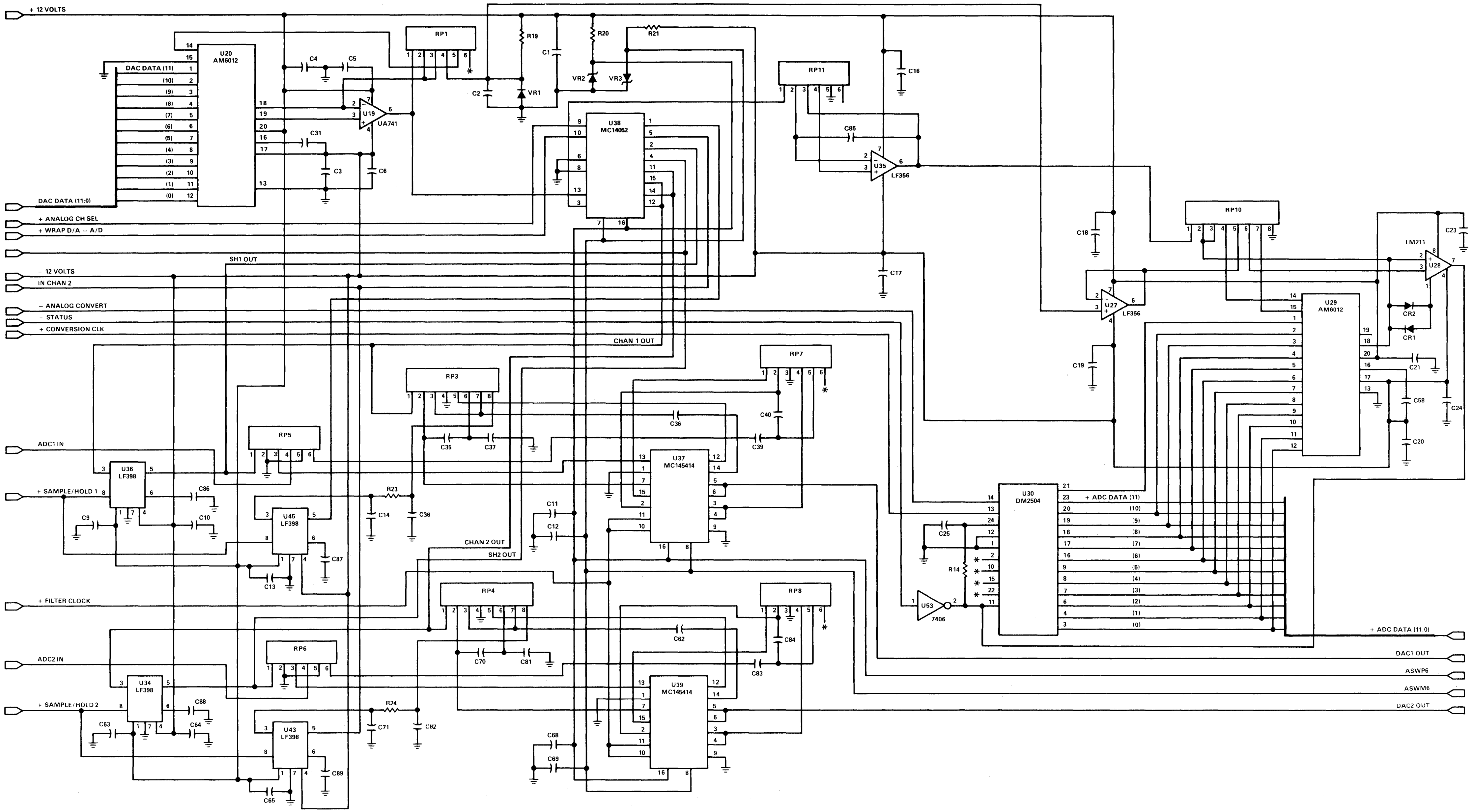


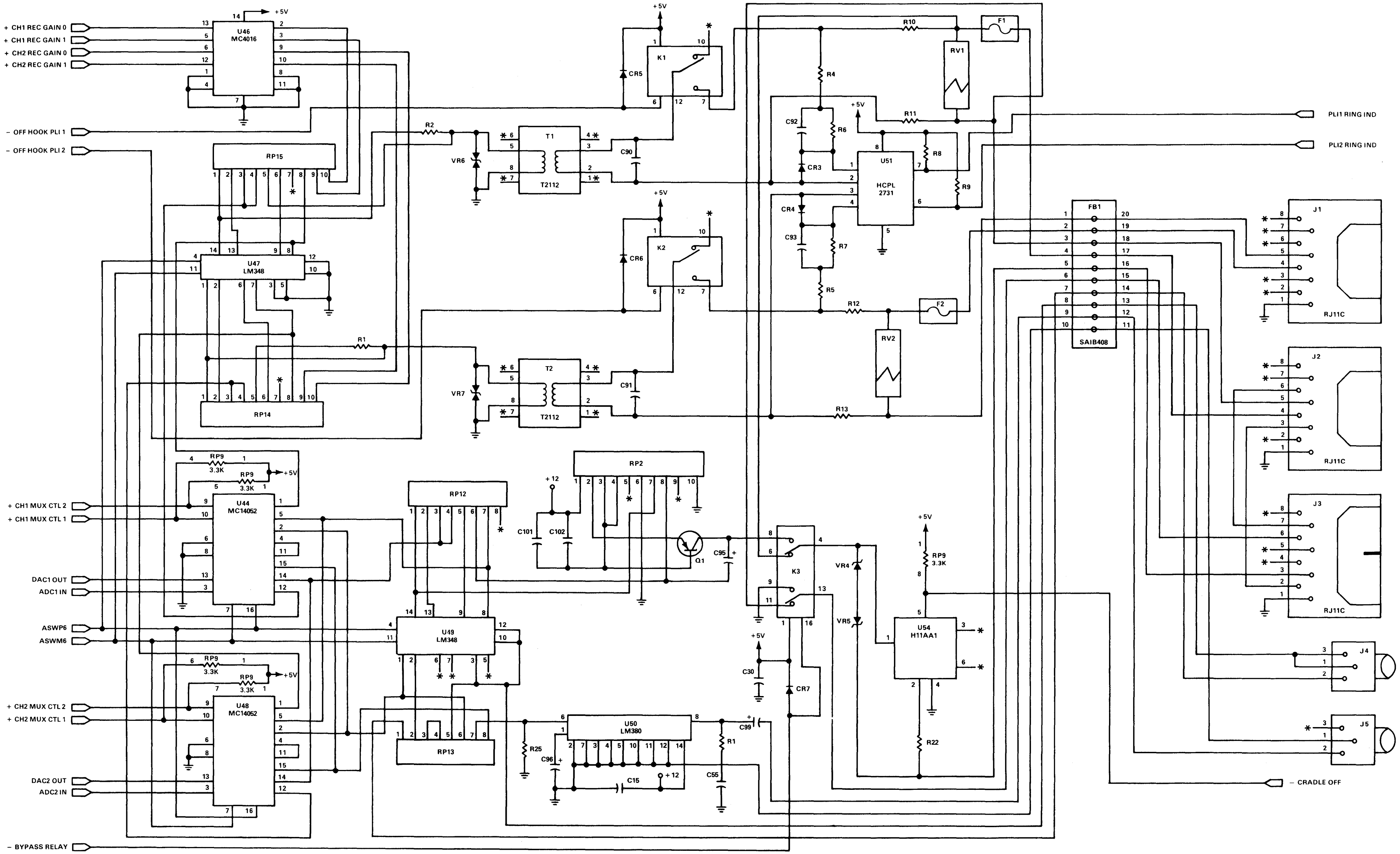
INST DATA (15: 0)
TMS32010 PROC U23
U9 F244
U15 F244
U11 S260
U12 AS00
U10 ALS138
U24 AS04
U56 DELAY 15NS
U8 2018D-55
U7 2018D-55
U4 2018D-55
U3 2018D-55
XTAL
19.968 MHz
C59
C60
INSTR ADDR (10: 0)
+ PSP ADDR 11
+ PSP ADDR 0
+ PSP ADDR 1
+ PSP ADDR 2
INSTR ADDR (10:0)
- REG SELECT
- RD STAT
- LD SCR2
+ GA RESET
- PROC INT
+ 12 VOLTS
- RESET BUF
- WE
- MEN
- DEN
+ CLKOUT
- INSTR R/W
- INST OE
- INST WE
- ICS 0 HIGH
- ICS 1 HIGH
- ICS 0 LOW
- ICS 1 LOW
INST DATA (15: 8)
INST DATA (7: 0)



– ICS 3 LOW
– ICS 2 LOW
– ICS 2 HIGH
– INST OE
– INST WE
– ICS 3 HIGH
INSTR ADDR (10: 0)
INST DATA (15: 0)
– RESET BUF
+ BUS DIRECTION
– BUS ENABLE
– SYS CS 0 LOW
– SYS CS 0 HIGH
+ SYS R/W
SYS DATA (15: 0)
SYS ADDR (10: 0)
U6 2018D-55
U2 2018D-55
U5 2018D-55
U1 2018D-55
U16 F244
U15 F244
U14 ALS245
U13 ALS245
U18 2081D-45
U17 2018D-45
U24 AS04
U56 IN OUT DELAY 15NS
U12 AS00
INSTR ADDR
INST DATA
SYS ADDR
SYS DATA
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 CS OE WE
O1 O2 O3 O4 O5 O6 O7 O8
1A0 1A1 1A2 1A3 2A0 2A1 2A2 2A3 OE1 OE2 OE3
1Y0 1Y1 1Y2 1Y3 2Y0 2Y1 2Y2 2Y3
A0 A1 A2 A3 A4 A5 A6 A7 DIR E
B0 B1 B2 B3 B4 B5 B6 B7


SYS DATA (15: 0)
SYS DATA (0)
U25
ALS174
U26
ALS174
+ DAC DATA (0)
+ DAC DATA (11: 0)
U22
ALS174
U21
ALS174
- LD DAC BUFFER
- RESET BUF
- ANALOG CONVERT
- LD SCR2
U24
AS04
+ DAC DATA (11)
- RD STAT
- CRADLE OFF
- PLI2 RING IND
- PLI1 RING IND
U42
LS125
SYS DATA (15)
SYS DATA (14)
SYS DATA (13)
U41
ALS174
+ CH1 REC GAIN 0
+ CH1 REC GAIN 1
+ CH2 REC GAIN 0
U53
7406
- BYPASS RELAY
- OFF HOOK PLI 1
- OFF HOOK PLI 2
+ ADC DATA (11)
U33
ALS244
SYS DATA (15)
- RD ADC BUFFER
U40
ALS174
+ CH2 MUX CTL 2
+ CH2 MUX CTL 1
+ CH1 MUX CTL 2
+ CH1 MUX CTL 1
+ WRAP DAC ADC
+ CH2 REC GAIN 1
U31
ALS244
+ ADC DATA (11: 0)


+ 12 VOLTS
U20
AM6012
DAC DATA (11)
DAC DATA (11:0)
+ ANALOG CH SEL
+ WRAP D/A — A/D
SH1 OUT
– 12 VOLTS
IN CHAN 2
– ANALOG CONVERT
– STATUS
+ CONVERSION CLK
ADC1 IN
+ SAMPLE/HOLD 1
+ FILTER CLOCK
ADC2 IN
+ SAMPLE/HOLD 2
C4
C5
C31
C3
C6
U19
UA741
RP1
R19
R20
R21
C1
C2
VR1
VR2
VR3
U38
MC14052
RP11
C16
C85
U35
LF356
C17
CHAN 1 OUT
RP3
C35
C37
C36
RP5
U36
LF398
C86
C9
C10
U45
LF398
C87
C13
R23
C14
C38
RP7
C40
C39
U37
MC145414
C11
C12
CHAN 2 OUT
SH2 OUT
RP4
C70
C81
C62
RP6
U34
LF398
C88
C63
C64
U43
LF398
C89
C65
R24
C71
C82
RP8
C84
C83
U39
MC145414
C68
C69
C18
C19
U27
LF356
RP10
C23
LM211
U28
CR2
CR1
U29
AM6012
C21
C58
C24
C20
U30
DM2504
+ ADC DATA (11)
C25
R14
U53
7406
+ ADC DATA (11:0)
DAC1 OUT
ASWP6
ASWM6
DAC2 OUT


+ CH1 REC GAIN 0
+ CH1 REC GAIN 1
+ CH2 REC GAIN 0
+ CH2 REC GAIN 1
– OFF HOOK PLI 1
– OFF HOOK PLI 2
+ CH1 MUX CTL 2
+ CH1 MUX CTL 1
DAC1 OUT
ADC1 IN
ASWP6
ASWM6
+ CH2 MUX CTL 2
+ CH2 MUX CTL 1
DAC2 OUT
ADC2 IN
– BYPASS RELAY
U46
MC4016
+5V
RP15
U47
LM348
RP14
R1
R2
VR6
VR7
T1
T2
T2112
C90
C91
K1
K2
CR5
CR6
R10
F1
RV1
R4
R11
C92
R6
CR3
U51
HCPL 2731
R8
R9
CR4
C93
R7
R5
R12
F2
RV2
R13
PLI1 RING IND
PLI2 RING IND
FB1
SAIB408
J1
J2
J3
RJ11C
J4
J5
– CRADLE OFF
RP9
3.3K
U44
MC14052
U48
MC14052
RP12
RP2
+12
C101
C102
Q1
C95
K3
VR4
VR5
U54
H11AA1
R22
U49
LM348
RP13
R25
C96
U50
LM380
C15
C99
R1
C55
C30
CR7

Index

Notes:

IBM

Personal Computer
Hardware Reference
Library

IBM Prototype Card

6361513

IBM Prototype Card

Contents

Description

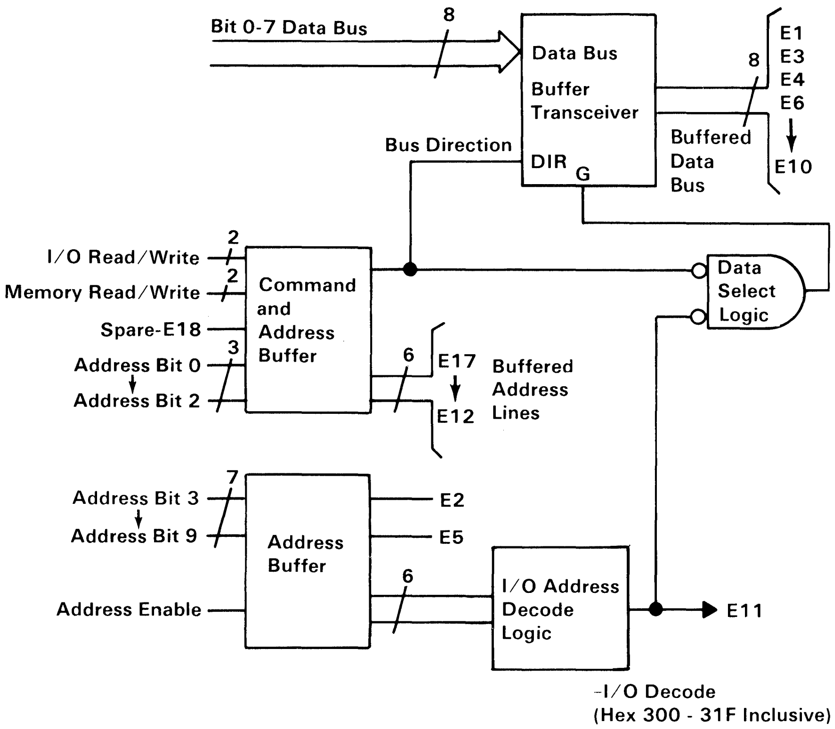
The IBM Prototype Card is 106.7 millimeters (4.2 inches) high by 335.3 millimeters (13.2 inches) long and plugs into an expansion unit or system unit expansion slot. All system control signals and voltage requirements are provided through a 2- by 31-position card-edge tab.

The card contains a voltage bus (+5 Vdc) and ground bus (0 Vdc). Each bus borders the card, with the voltage bus on the back (pin side) and the ground bus on the front (component side). A system interface design is provided on the Prototype Card.

The Prototype Card can also accommodate a D-shell connector if it is needed. The connector size can range from a 9- to a 37-position connector.

> **Warning:** Install all components on the component side of the Prototype Card. The total width of the card, including components, should not exceed 12.7 millimeters (0.500 inch). If these specifications are not met, components on the Prototype Card may touch other cards plugged into adjacent slots.

The following is a block diagram of the IBM Prototype Card.



Prototype Card Block Diagram

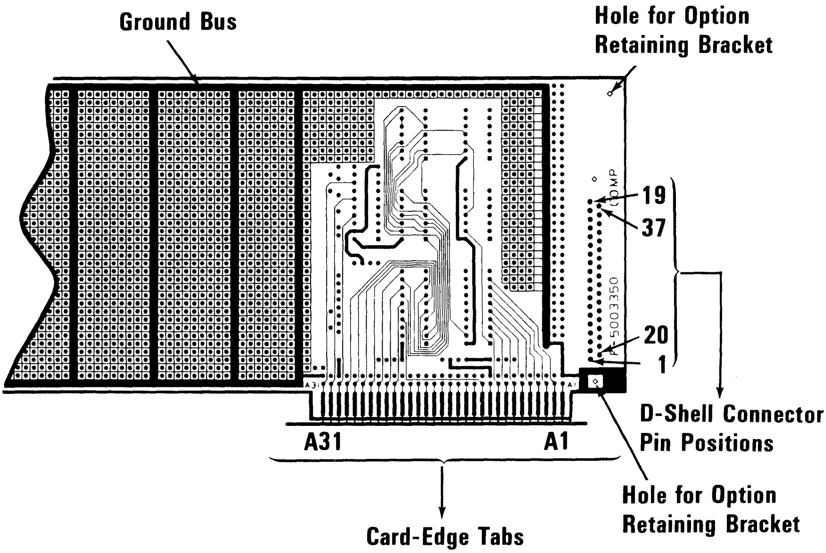
Interface

I/O Channel Interface

The Prototype Card has two layers screened onto it (one on the front and one on the back). It also has 3,909 plated through-holes that are 10.1 millimeters (0.040 inch) in size and have a 1.52-millimeter (0.060-inch) pad, which is on a 2.54-millimeter (0.10-inch) grid. There are 37 plated through-holes that are 1.22 millimeter (0.048 inch) in size. These holes are at the rear of the card (viewed as if installed in the machine). These 37 holes are used for a 9- to 37-position D-shell connector. The card also has 5 holes that are 3.18 millimeters (0.125 inch) in size. One hole is located just above the two rows of D-shell connector holes, and the other four are located in the corners of the board (one in each corner).

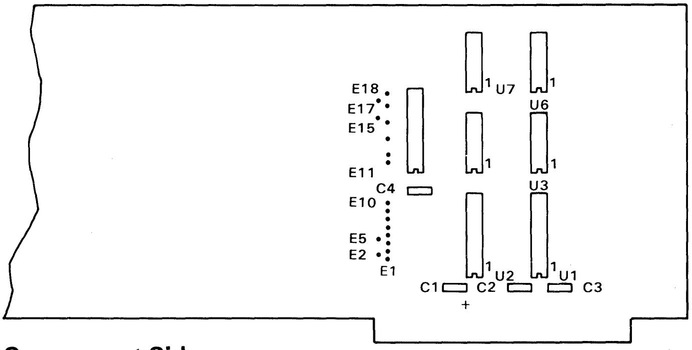
Prototype Card Layout

The component side has the ground bus, 1.27 millimeters (0.05 inch) wide, screened onto it, and card-edge tabs that are labeled A1 through A31.


Ground Bus
Hole for Option Retaining Bracket
19
37
20
1
D-Shell Connector Pin Positions
A31
A1
Hole for Option Retaining Bracket
Card-Edge Tabs
COMP
5003350

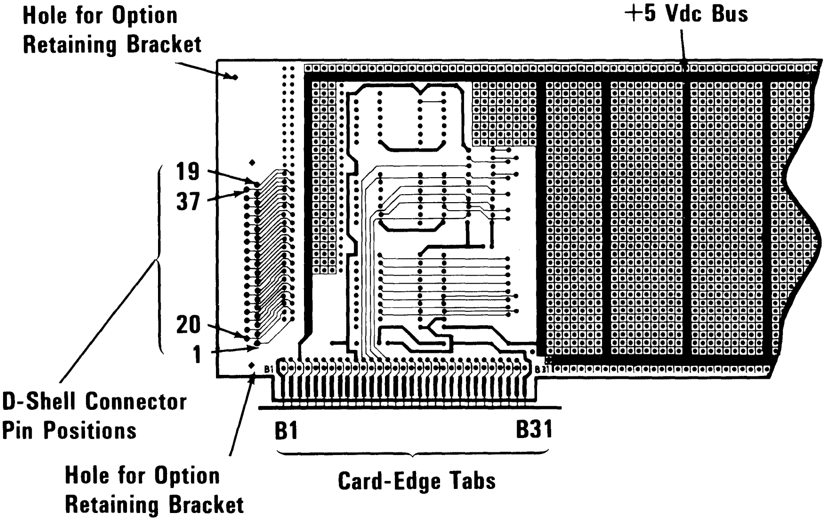
Component Side

The component side also has a silk screen printed on it that is used as a component guide for the I/O interface.



Component Side

The pin side has a +5-Vdc bus, 1.27 millimeters (0.05 inch) wide, screened onto it, and card-edge tabs that are labeled B1 through B31.


Hole for Option Retaining Bracket
+5 Vdc Bus
19
37
20
1
D-Shell Connector Pin Positions
B1
B31
B1
B31
Hole for Option Retaining Bracket
Card-Edge Tabs

Pin Side

Each card-edge tab is connected to a plated through-hole by a 0.3-millimeter (0.012-inch) land. There are three ground tabs connected to the ground bus by three 0.3-millimeter (0.012-inch) lands. Also, there are two +5-Vdc tabs connected to the voltage bus by two 0.3-millimeter (0.012-inch) lands.

For additional interfacing information, refer to "I/O Channel Description" and "I/O Channel Diagram" in your *Technical Reference* system manual. If the recommended interface logic is used, the following list of TTL-type numbers will help you select the necessary components.

Component	TTL Number	Description
U1	74LS245	Octal Bus Transceiver
U2, U5	74LS244	Octal Buffers Line Driver/Line Receivers
U4	74LS04	Hex Inverters
U3	74LS08	Quadruple 2 - Input Positive - AND Gate
U6	74LS02	Quadruple 2 - Input Positive - NOR Gate
U7	74LS21	Dual 4 - Input Positive - AND Gate
C1		10.0 µF Tantalum Capacitor
C2, C3, C4		0.047 µF Ceramic Capacitor

System Loading and Power Limitations

Because of the number of options that may be installed in the system, the I/O bus loading should be limited to one Schottky TTL load. If the interface circuity on the card is used, then this requirement is met.

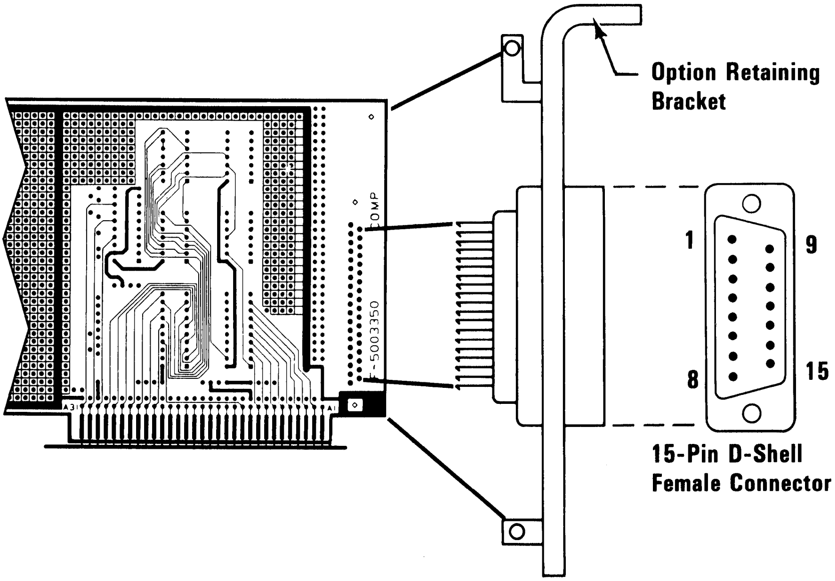
Refer to the power supply information in your *Technical Reference* system manual for the power limitations to be observed.

External Interface

If a connector is required for the card function, you should purchase one of the recommended Amp connectors listed in the following table, or its equivalent.

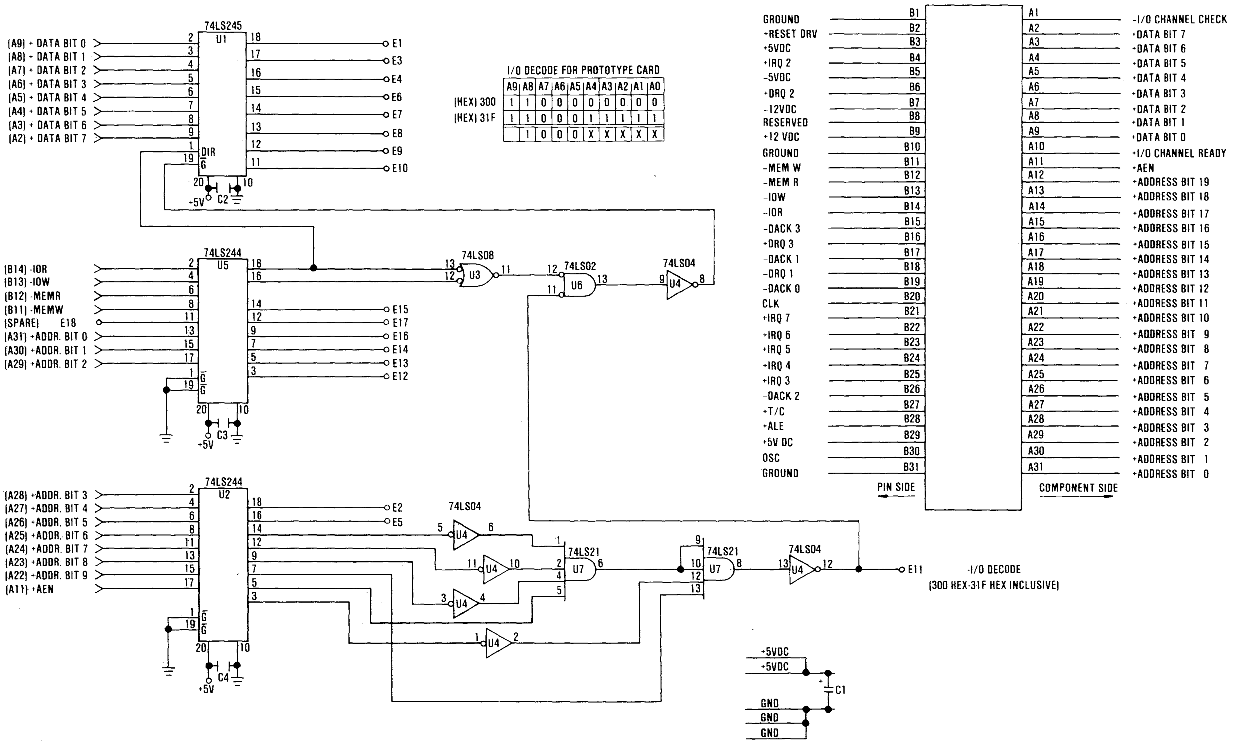
Connector Size	Part Number (Amp)
9-pin D-shell (Male)	205865-1
9-pin D-shell (Female)	205866-1
15-pin D-shell (Male)	205867-1
15-pin D-shell (Female)	205868-1
25-pin D-shell (Male)	205857-1
25-pin D-shell (Female)	205858-1
37-pin D-shell (Male)	205859-1
37-pin D-shell (Female)	205860-1

The following example shows how a 15-pin, D-shell, female connector is attached to a prototype card.



Component Side

8 Prototype Card



I/O DECODE FOR PROTOTYPE CARD

	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
(HEX) 300	1	1	0	0	0	0	0	0	0	0
(HEX) 31F	1	1	0	0	0	1	1	1	1	1
		1	0	0	0	X	X	X	X	X

GROUND	B1	A1	-I/O CHANNEL CHECK
+RESET DRV	B2	A2	+DATA BIT 7
+5VDC	B3	A3	+DATA BIT 6
+IRQ 2	B4	A4	+DATA BIT 5
-5VDC	B5	A5	+DATA BIT 4
+DRQ 2	B6	A6	+DATA BIT 3
-12VDC	B7	A7	+DATA BIT 2
RESERVED	B8	A8	+DATA BIT 1
+12 VDC	B9	A9	+DATA BIT 0
GROUND	B10	A10	+I/O CHANNEL READY
-MEM W	B11	A11	+AEN
-MEM R	B12	A12	+ADDRESS BIT 19
-IOW	B13	A13	+ADDRESS BIT 18
-IOR	B14	A14	+ADDRESS BIT 17
-DACK 3	B15	A15	+ADDRESS BIT 16
+DRQ 3	B16	A16	+ADDRESS BIT 15
-DACK 1	B17	A17	+ADDRESS BIT 14
-DRQ 1	B18	A18	+ADDRESS BIT 13
-DACK 0	B19	A19	+ADDRESS BIT 12
CLK	B20	A20	+ADDRESS BIT 11
+IRQ 7	B21	A21	+ADDRESS BIT 10
+IRQ 6	B22	A22	+ADDRESS BIT 9
+IRQ 5	B23	A23	+ADDRESS BIT 8
+IRQ 4	B24	A24	+ADDRESS BIT 7
+IRQ 3	B25	A25	+ADDRESS BIT 6
-DACK 2	B26	A26	+ADDRESS BIT 5
+T/C	B27	A27	+ADDRESS BIT 4
+ALE	B28	A28	+ADDRESS BIT 3
+5V DC	B29	A29	+ADDRESS BIT 2
OSC	B30	A30	+ADDRESS BIT 1
GROUND	B31	A31	+ADDRESS BIT 0

Prototype Card (Sheet 1 of 1)

10 Prototype Card

IBM

Personal Computer
Hardware Reference
Library

DAC ADAPTER DISTRIBUTION PANEL

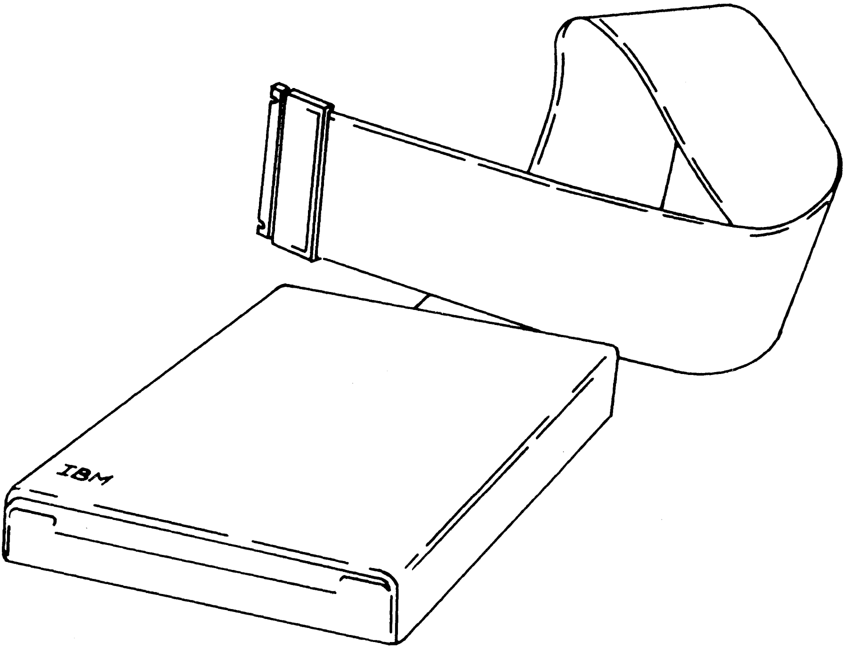
IBM Personal Computer Data Acquisition and Control Adapter Distribution Panel Technical Reference

6138157
August 15, 1984

IBM Personal Computer Data Acquisition and Control Adapter Distribution Panel

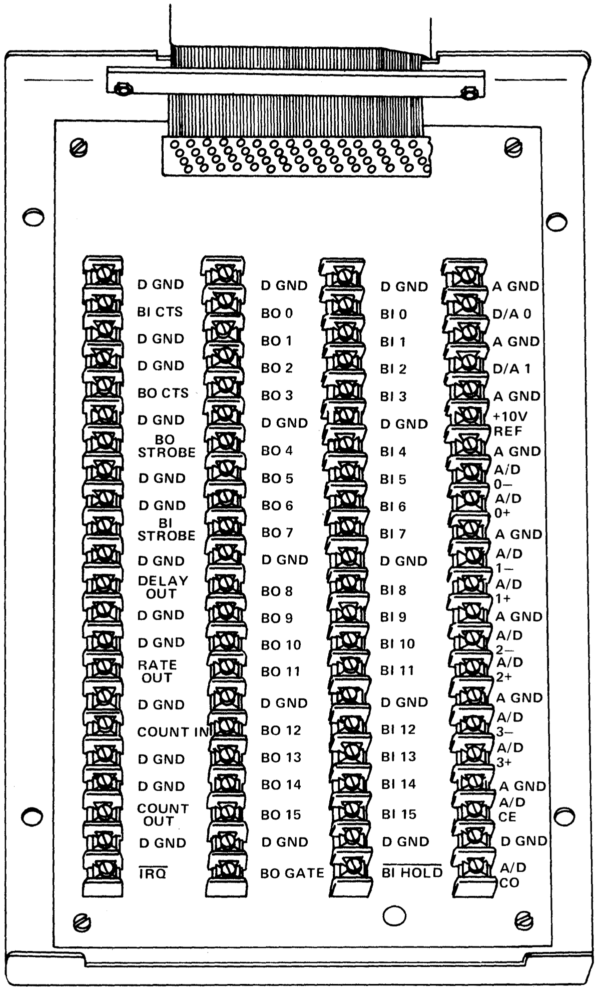
The IBM Personal Computer Data Acquisition and Control (Data Acquisition) Adapter Distribution Panel, with attached ribbon cable and 60-pin connector, is provided for external access to the Data Acquisition Adapter's analog I/O device, binary I/O device, and timer/counter device.

Four 22-screw terminal strips on the Distribution Panel allow the user to connect external devices to the Data Acquisition Adapter.

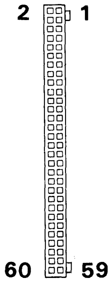


August 15, 1984

Following is a diagram of the distribution panel.

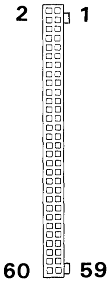


The following shows the signals at the distribution panel and the connector pins on which they appear.



	Signal Name/Description	Pin	
	D/A 1	1	
	D/A 0	2	
	+10V REF	3	
	A GND	4	
	A/D 0-	5	
	A/D 0+	6	
	A/D 1-	7	
	A/D 1+	8	
	A/D 2-	9	
	A/D 2+	10	
	A/D 3-	11	Distribution Panel Connector
Distribution Panel	A/D 3+	12	
	A GND	13	
	A/D CE	14	
	D GND	15	
	A/D CO	16	
	BI 8	17	
	BO 8	18	
	BI 9	19	
	BO 9	20	
	BI 10	21	
	BO 10	22	
	BI 11	23	
	BO 11	24	
	BI 12	25	
	BO 12	26	
	BI 13	27	
	BO 13	28	
	BI 14	29	
	BO 14	30	

Part 1 of 2



Distribution Panel

Distribution Panel Connector

Signal Name/Description	Pin
BI 15	31
BO 15	32
BI HOLD	33
BO GATE	34
BI 0	35
BO 0	36
BI 1	37
BO 1	38
BI 2	39
BO 2	40
BI 3	41
BO 3	42
BI 4	43
BO 4	44
BI 5	45
BO 5	46
BI 6	47
BO 6	48
BI 7	49
BO 7	50
RATE OUT	51
DELAY OUT	52
BI STROBE	53
BO STROBE	54
BO CTS	55
BI CTS	56
IRQ	57
COUNT OUT	58
COUNT IN	59
D GND	60

Part 2 of 2

Personal Computer
Hardware Reference
Library

Prototype Adapter

IBM PERSONAL COMPUTER AT PROTOTYPE ADAPTER

Contents

Notes:

Description

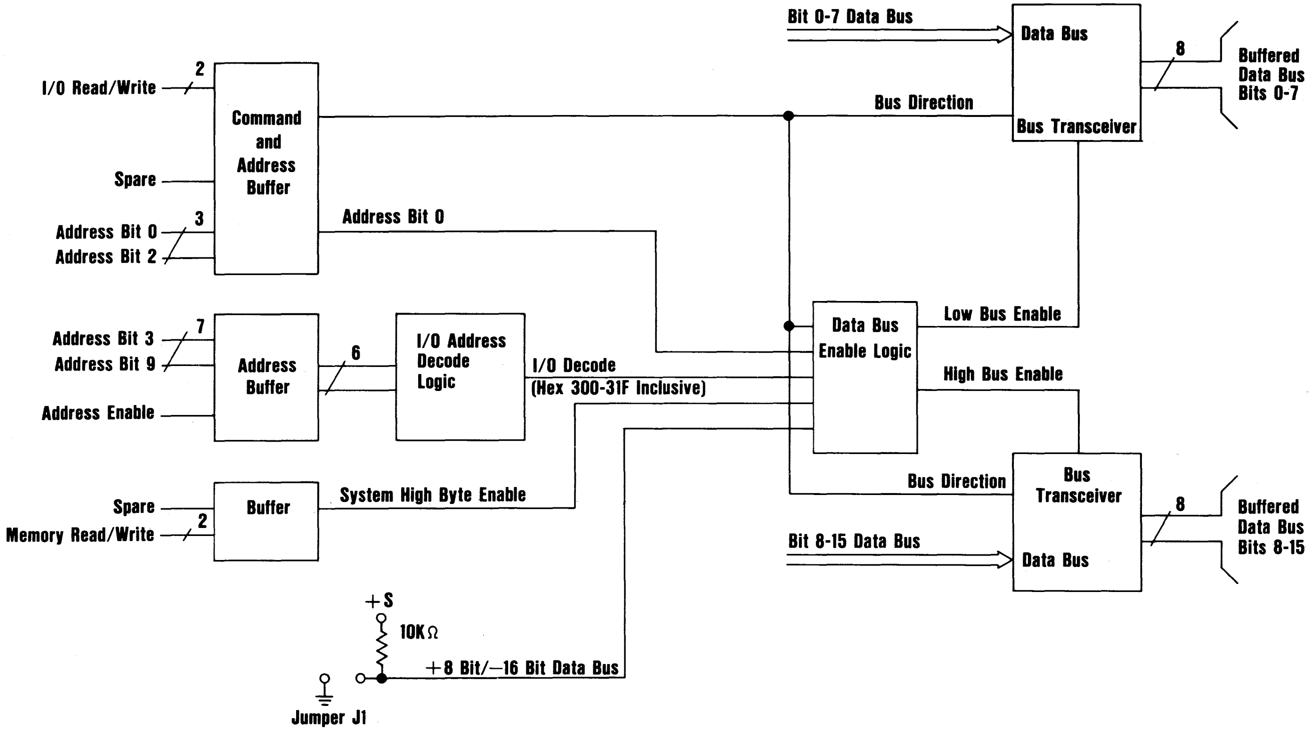
The IBM Personal Computer AT Prototype Adapter is 121.9 millimeters (4.8 inches) high by 333.25 millimeters (13.12 inches) long and plugs into any system-unit expansion slot except number 1 or 7. Two card-edge tabs, one 2- by 31-position and one 2- by 18-position, provide all system control signals and voltages.

The adapter has a voltage bus (+5 Vdc) and a ground bus (0 Vdc). Each bus borders the adapter, with the ground bus on the component side and the voltage bus on the pin side. A system interface is also provided on the adapter with a jumper to specify whether the device has an 8- or a 16-bit data bus.

This adapter also accommodates a D-shell connector from 9 to 37 positions.

> **Note:** All components must be installed on the component side of the adapter. The total width of the adapter, including components, may not exceed 12.7 millimeters (0.5 inch). If these specifications are not met, components on the IBM Personal Computer AT Prototype Adapter may touch other adapters plugged into adjacent expansion slots.

The following is a block diagram of the IBM Personal Computer AT Prototype Adapter.



Prototype Adapter Block Diagram

Adapter Design

The following information is provided to assist in designing an adapter using the IBM Personal Computer AT Prototype Adapter.

Designing an Input/Output Adapter

The following information may be used to design an input/output type of adapter.

Programming

Insert a Jump instruction after all I/O read (IOR) or I/O write (IOW) assembler language instructions to avoid a potential timing problem caused by slow I/O devices. The following figure shows a typical programming sequence.

Before	After	
Your Code		Your Code
IOR		IOR
Your Code		JMP NEXT
	NEXT:	Your Code

Program Sequence

Jumper Wire (J1)

Your design can use either 8 bits of the data bus (jumper off) or the full 16 bits of the data bus (jumper on). Most devices have 8-bit data buses.

Wait-State Generator Circuits

If your device runs too slow, you must add a wait-state generator to make the I/O read and write signals longer. First, determine the time needed by your device from the start of an IOR signal until it can put data on the system's data bus. Next, compare that time with the time given by the system's microprocessor. The system microprocessor gives 750 nanoseconds for 8-bit devices and 250 nanoseconds for 16-bit devices.

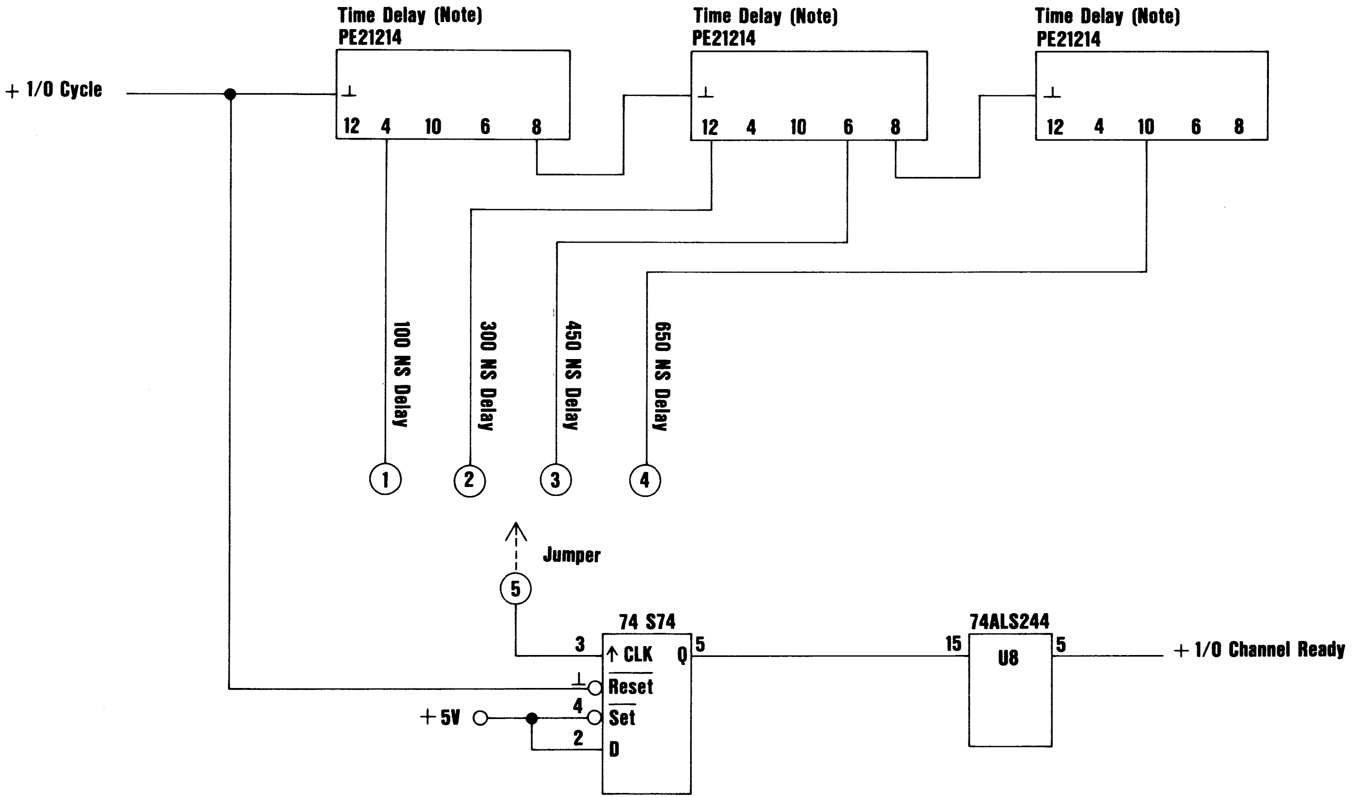
A similar problem may exist for an IOW signal. Determine the write data setup time, which is the time required by your design from the time it is given valid data until it is told to take this data by the IOW signal. The time given by the system microprocessor from when data is first valid to the device until the IOW signal goes active and then inactive is shown in the following figure. Your design can take the data when IOW goes active (less setup time) or when IOW goes inactive (more setup time).

8-Bit Device	16-Bit Device	Description
100 ns	100 ns	Data Valid Until IOW is Active.
850 ns	350 ns	Data Valid Until IOW is Inactive.

IOW Timing

If the time given by the system microprocessor is not enough, you must add a wait-state generator circuit that will provide longer IOR and IOW signals. A recommended wait-state generator circuit is shown in the following figure.

> **Note:** Pulse Engineering Inc. PE21214 is the delay module used.



Wait-State Generator Circuit

Note: To add wait states and increase the time given by the microprocessor for I/O Read and Write commands, install one of the following jumpers.

- 16-Bit Design

1 wait state	250 nanoseconds--No jumper
2 wait states	417 nanoseconds--Jumper 1 to 5
3 wait states	583 nanoseconds--Jumper 2 to 5
4 wait states	750 nanoseconds--Jumper 3 to 5
5 wait states	917 nanoseconds--Jumper 4 to 5

- 8-Bit Design

4 wait states	750 nanoseconds--No Jumper
5 wait states	917 nanoseconds--Jumper 4 to 5

Designing a Memory Adapter

The following information may be used to design a memory adapter.

Control Lines

There are two sets of memory control lines. '-SMEMR' for system-memory read, and '-SMEMW' for system-memory write. They are active when accessing memory in the first megabyte (address bits 20 through 23 are all off). If you use these lines, you can avoid an address decode circuit that checks for address bits 20 through 23 being off.

The other set of control lines is '-MEMR' and '-MEMW'. These are active when addressing all memory locations. If you wish to design memory that will answer to addresses above the first megabyte, you must use these lines and decode address bits 20 through 23 to select the particular address range your memory occupies.

System Address Lines (SA)

The 20 lowest-order address lines are 'SA0' through 'SA19'. SA address bits are active a minimum of 30 nanoseconds before a control line goes active, and they stay active a minimum of 66 nanoseconds *after* the control line goes inactive. Timings are at the adapter socket.

Local Address Lines (LA)

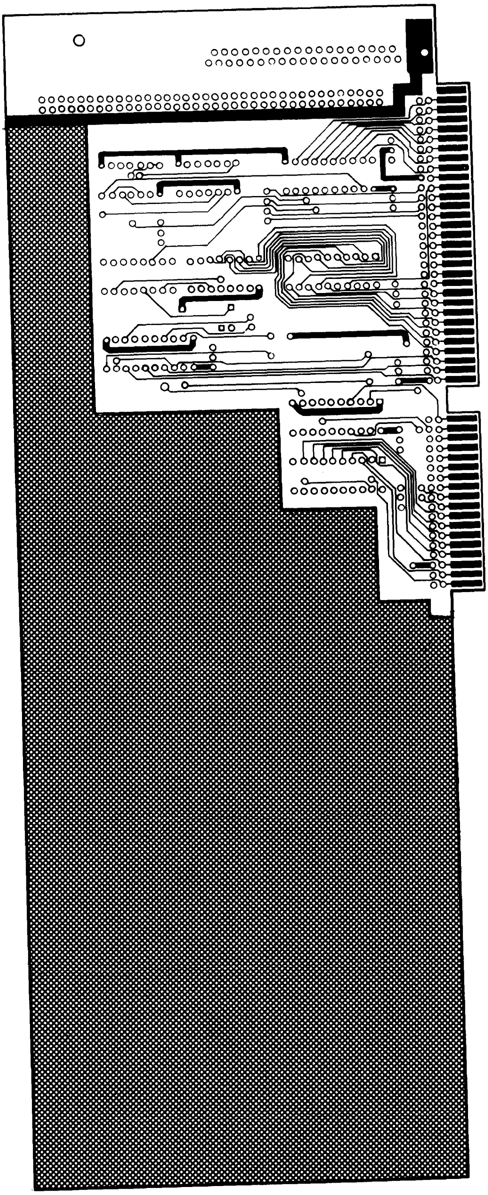
There are seven high-order address lines called 'LA17' through 'LA23'. LA address bits are active a minimum of 159 nanoseconds before a control line goes active, and they typically stay active 83 nanoseconds *before* the control line goes inactive. LA bits should be decoded to select the particular address range your memory occupies. Because this decode will go inactive 83 nanoseconds before the control line goes inactive, it may be necessary to latch the decode. The output of this decoder circuit should be connected to the input of a transparent latch, such as a 74ALS573 (+BALE should be connected to the clock pin on the latch). If this is done, the output of the 74LS573 will be active approximately 30 nanoseconds before a control line goes active, and will stay active approximately 66 nanoseconds *after* the control line goes inactive. Timings are at the adapter socket.

IBM Personal Computer AT Prototype Adapter Layout

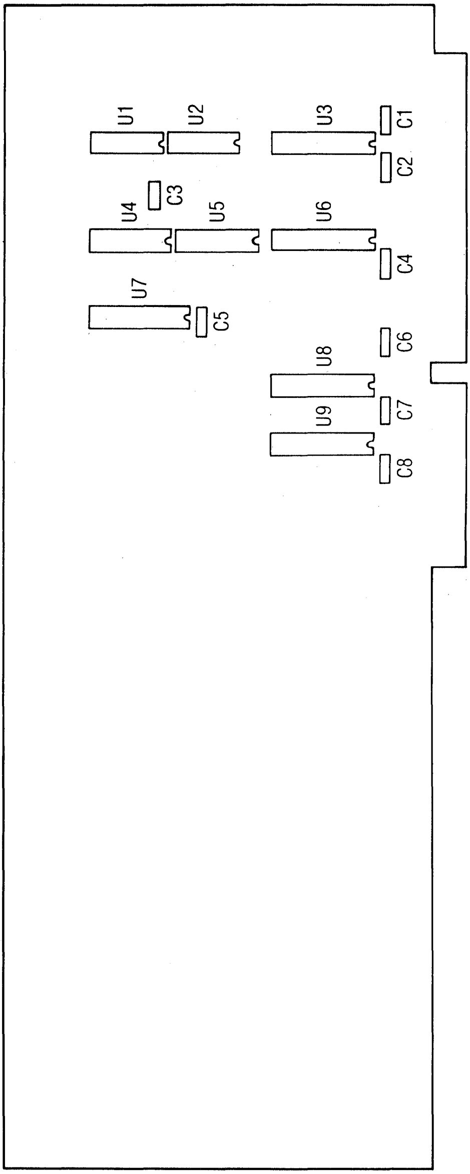
The IBM Personal Computer AT Prototype Adapter has two layers screened onto it: one on the front and one on the back. It also has 4,311 plated through-holes that are 10.1 millimeters (0.04 inch) wide and have a 1.52-millimeter (0.06-inch) pad. These holes are arranged in a 2.54-millimeter (0.1-inch) grid. There are 37 plated through-holes, 1.22 millimeters (0.048 inch) wide, on the rear of the adapter that are used for a 9- to 37-position D-shell connector. The adapter also has 5 holes that are 3.18 millimeters (0.125 inch) wide. One of these is just above the two rows of D-shell connector holes, and each of the other four is in a corner of the adapter.

Component Side

The component side of the adapter has a ground bus, 1.27 millimeters (0.05 inch) wide screened onto it and two card-edge tabs labeled A1 through A31 and C1 through C31. The following figure shows the ground bus and card edge-tabs.

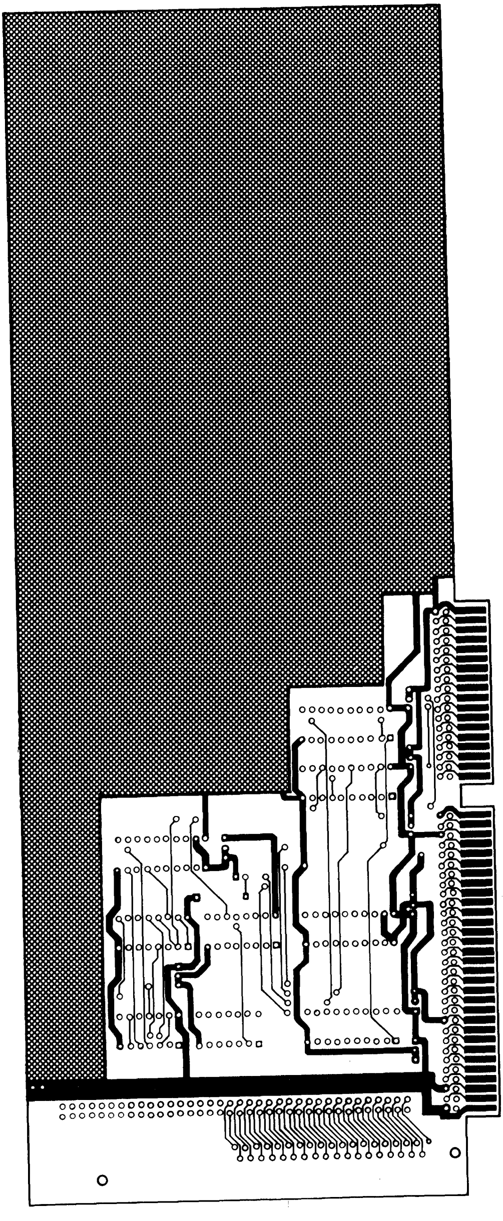


The component side of the adapter also has a silk screen printed on it that may be used as a component guide for the I/O interface. The following figure shows this silk screen.



Pin Side

The pin side of the adapter has a 5-Vdc bus, 1.27 millimeters (0.05 inch) wide, screened onto it, and two card-edge tabs: labeled B1 through B31 and D1 through D18. The following figure shows the 5-Vdc bus and card edge-tabs.



Card-Edge Tabs

Each card-edge tab is connected to a plated through-hole by a 0.3-millimeter (0.012-inch) land. Four ground tabs are connected to the ground bus by four 0.3-millimeter (0.012-inch) lands, and three 5 Vdc tabs are connected to the 5-Vdc bus by three 0.3-millimeter (0.012 inch) lands.

Additional Information

Additional information regarding the I/O interface may be found under 'I/O Channel' in Section 1 of IBM Personal Computer AT *Technical Reference* manual. Logic diagrams of the IBM Personal Computer AT Prototype Adapter may be found later in this section. If the recommended interface logic is to be used, the following figure shows the recommended components and their TTL numbers.

Component	TTL #	Description
U1	74S00	Quad 2 Input NAND
U2	74S10	Triple 3 Input NAND
U3, U9	74LS245	Octal Bus Transceiver
U4	74S139	Dual 1 of 4 Decoder
U5	74S138	1 of 8 Decoder
U6, U7, U8	74ALS244	Octal Buffers
C1, C6		10-Microfarad Tantalum Capacitor
C2, C3, C4, C5, C7, C8		0.047-Microfarad Ceramic Capacitor
R1		10 Kohm, .25-Watt, 10% Resistor (Axial Leads)
J1		Jumper Wire

Recommended Components

> **Note:** J1, U8, and U9 are not required for a design using only the low-order 8 bits of the data bus. Designs using all 16 bits of the data bus require these components.

Interfaces

Internal Interface

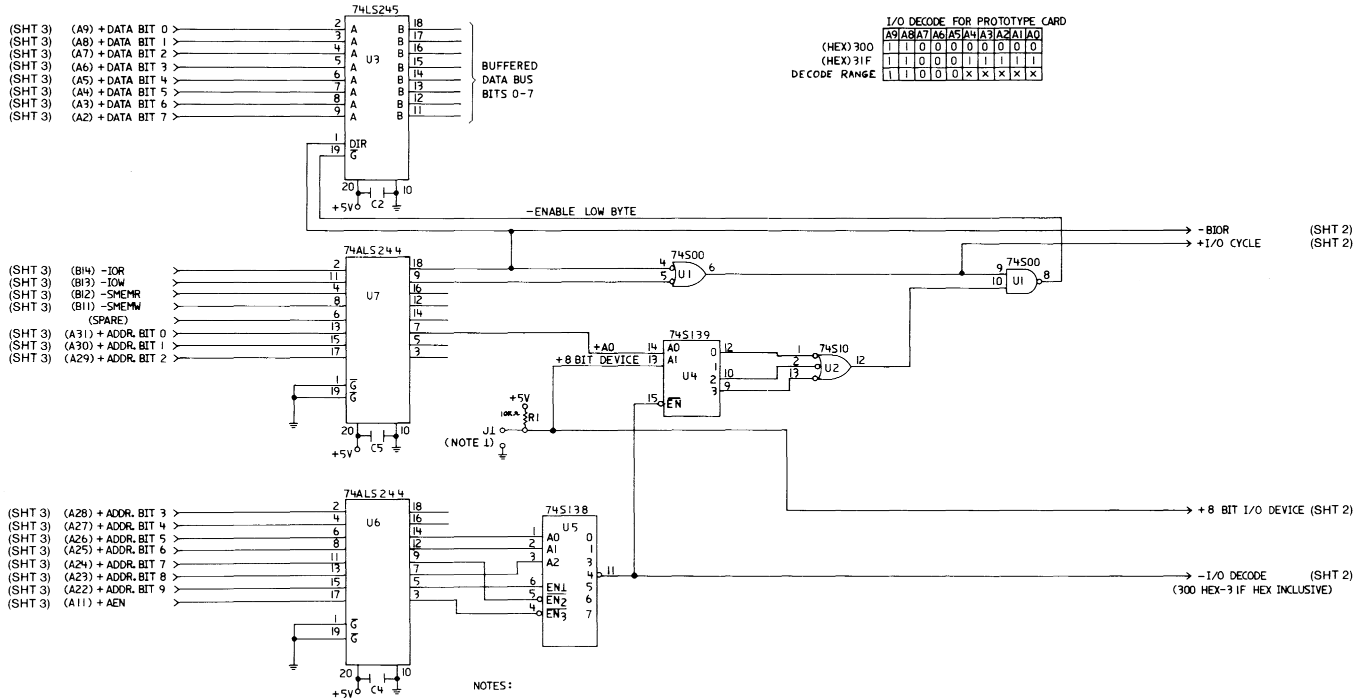
Because of the number of adapters that may be installed in the system, I/O bus loading should be limited to 1 Schottky TTL load. If the recommended interface logic is used, this requirement is met. Power limitations may be found under 'Power Supply' in the IBM Personal Computer AT *Technical Reference* Manual.

External Interface

The following figure lists the recommended connectors for the rear of the adapter.

Connector	Part no. (Amp) or Equivalent
9-Pin D-Shell (Male)	205865-1
9-Pin D-Shell (Female)	205866-1
15-Pin D-Shell (Male)	205867-1
15-Pin D-Shell (Female)	205868-1
25-Pin D-Shell (Male)	205857-1
25-Pin D-Shell (Female)	205858-1
37-Pin D-Shell (Male)	205859-1
37-Pin D-Shell (Female)	205860-1

Recommended Connectors



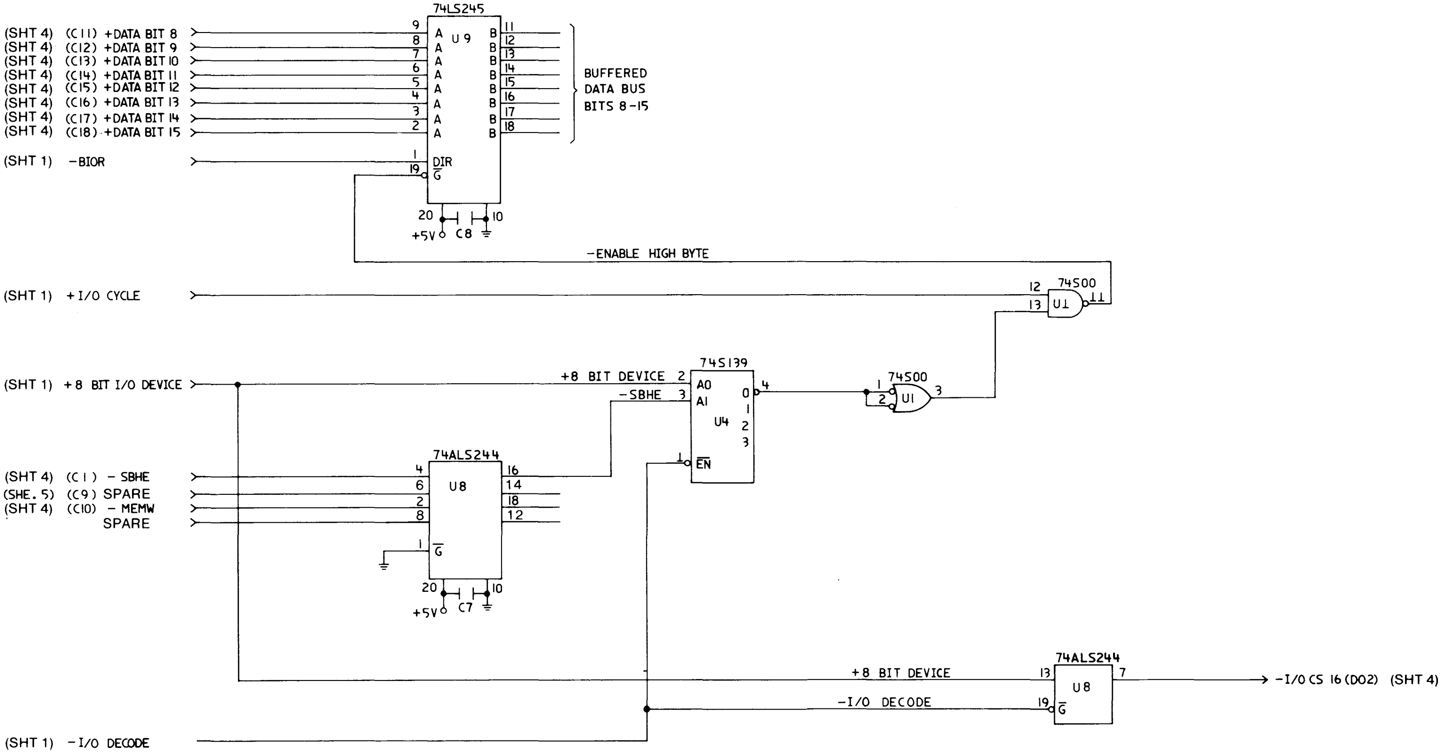
I/O DECODE FOR PROTOTYPE CARD

	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
(HEX) 300	1	1	0	0	0	0	0	0	0	0
(HEX) 31F	1	1	0	0	0	1	1	1	1	1
DECODE RANGE	1	1	0	0	0	x	x	x	x	x

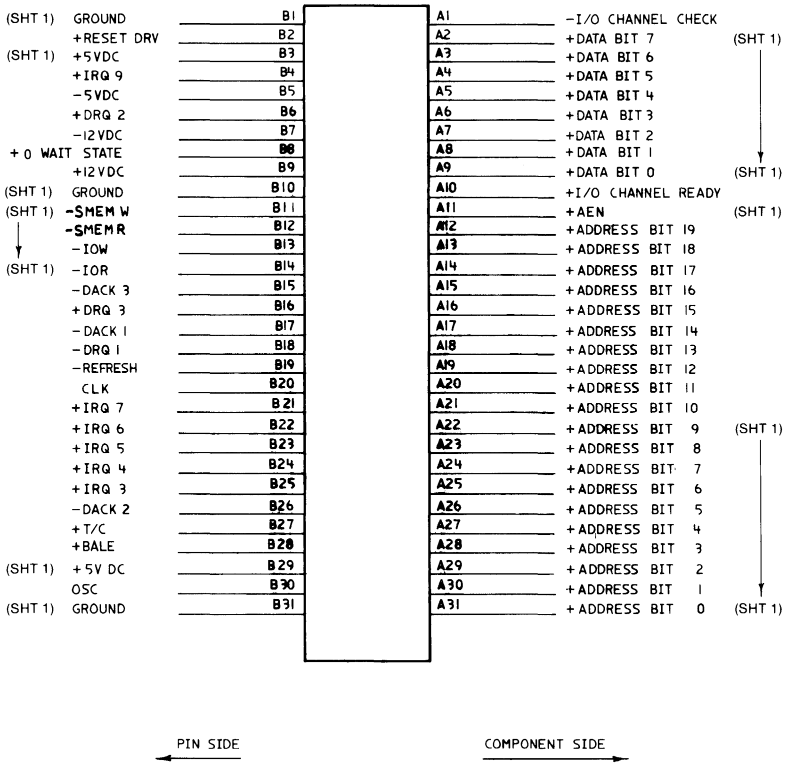
NOTES:

1 IF THE I/O DEVICE REQUIRES A 16 BIT WIDE DATA BUS, THEN INSTALL JUMPER J1. FOR 8 BIT WIDE I/O DEVICES NO JUMPERING IS REQUIRED.

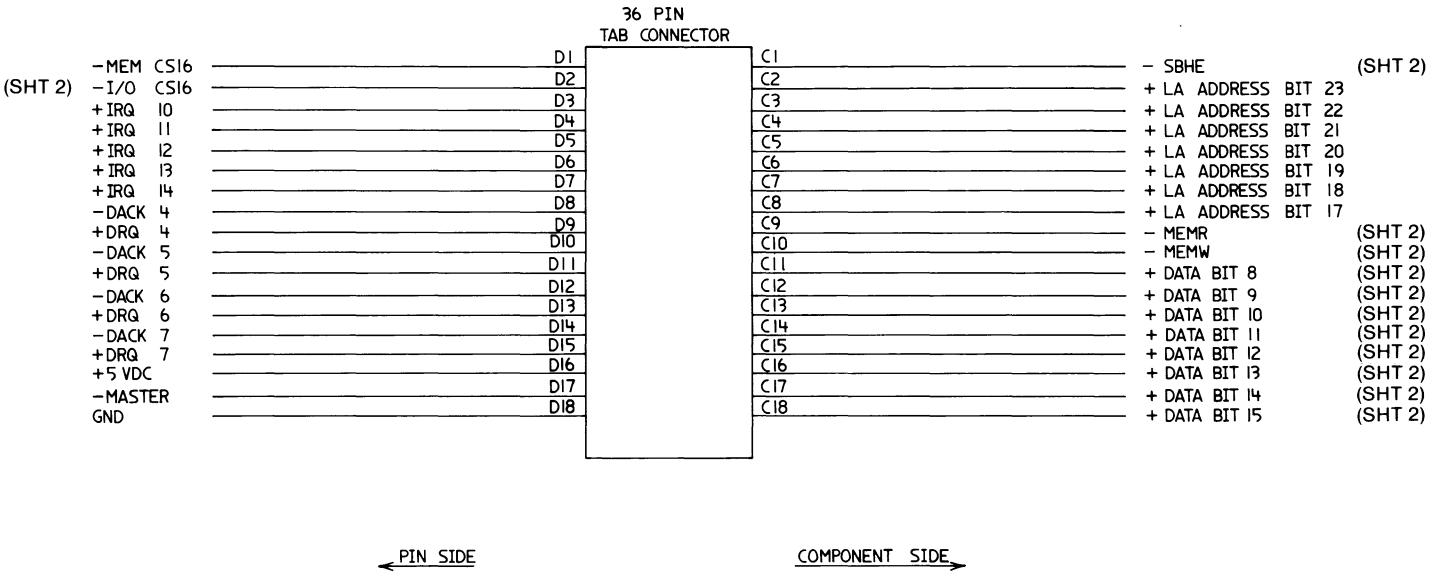
Prototype Adapter (Sheet 1 of 4)



Prototype Adapter (Sheet 2 of 4)



Prototype Adapter (Sheet 3 of 4)



Prototype Adapter (Sheet 4 of 4)

Notes:

IBM

Personal Computer
Hardware Reference
Library

IBM PC Compact Printer Connector Adapter

IBM PC Compact Printer Connector Adapter

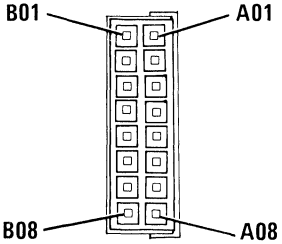
6361517

Contents

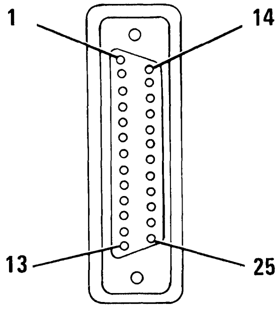
Description

The IBM PC Compact Printer Connector Adapter is required to connect the 16-lead cable of the IBM PC Compact Printer to the 25-pin D-shell connector of an Asynchronous Communications Adapter or Alternate Asynchronous Communications Adapter.

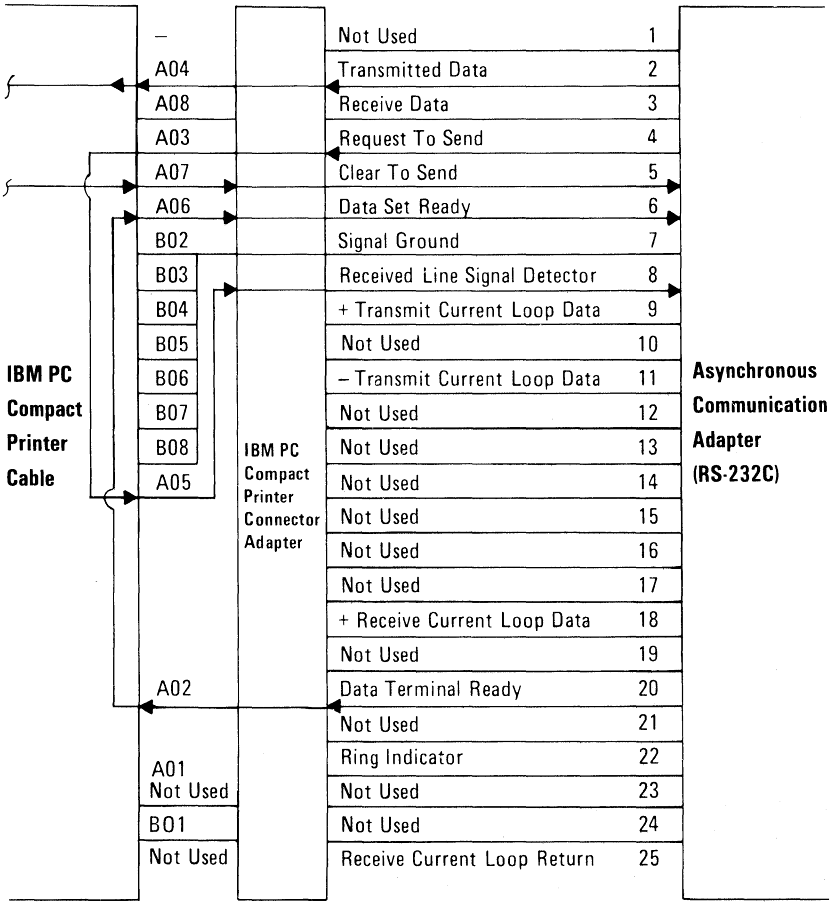
Specifications



Printer Cable End of Connector Adapter



System End of Connector Adapter


IBM PC Compact Printer Cable
IBM PC Compact Printer Connector Adapter
Asynchronous Communication Adapter (RS-232C)

Pin	Description	Pin
–	Not Used	1
A04	Transmitted Data	2
A08	Receive Data	3
A03	Request To Send	4
A07	Clear To Send	5
A06	Data Set Ready	6
B02	Signal Ground	7
B03	Received Line Signal Detector	8
B04	+ Transmit Current Loop Data	9
B05	Not Used	10
B06	– Transmit Current Loop Data	11
B07	Not Used	12
B08	Not Used	13
A05	Not Used	14
	Not Used	15
	Not Used	16
	Not Used	17
	+ Receive Current Loop Data	18
	Not Used	19
A02	Data Terminal Ready	20
	Not Used	21
A01 Not Used	Ring Indicator	22
	Not Used	23
B01	Not Used	24
Not Used	Receive Current Loop Return	25

Notes:

IBM

Personal Computer
Hardware Reference
Library

IBM Communications Adapter Cable

6361515

IBM Communications Adapter Cable

Contents

Description

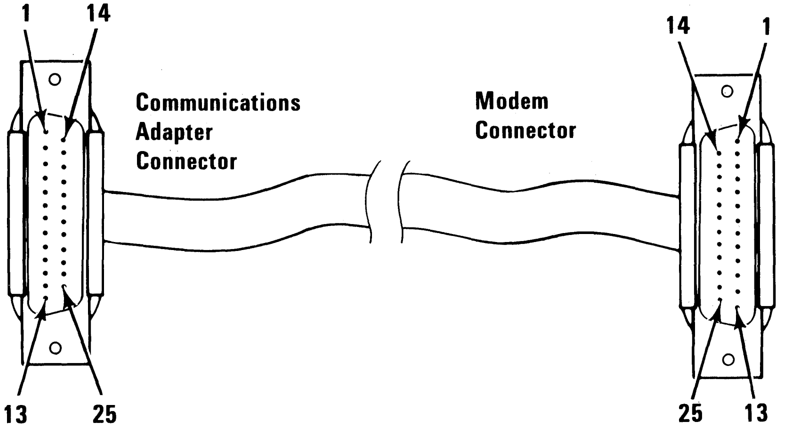
The IBM Communications Adapter Cable is a 3.05 m (10-ft) cable designed to connect an IBM communications adapter to a modem or other RS232-C data communications equipment (DCE). It is fully shielded and provides a high quality, low noise channel for interface between the communications adapter and DCE.

The connector ends are 25-pin D-shell connectors. All pin connections conform with the EIA RS232-C standard. In addition, connection is provided on pins 11, 18, and 25. These pins are designated as 'select standby,' 'test,' and 'test indicate,' respectively, on some modems. 'Select standby' is used to support the switched network backup facility, if applicable. 'Test' and 'test indicate' support a modem wrap function on modems designed for business-machine controlled modem wraps.

Specifications

The following page shows the pin locations and specifications for the IBM Communications Adapter Cable.

The IBM Communications Adapter Cable connects the following pins on the 25-pin D-shell connectors.



Communications Adapter Connector Pin #	Name	Modem Connector Pin #
NC	Outer Cable Shield	1
2	Transmitted Data	2
3	Received Data	3
4	Request to Send	4
5	Clear to Send	5
6	Data Set Ready	6
7	Signal Ground (Inner Lead Shields)	7
8	Received Line Signal Detector	8
NC		NC
NC		NC
11	Select Standby	11
NC		NC
NC		NC
NC		NC
15	Transmitter Signal Element Timing	15
NC		NC
17	Receiver Signal Element Timing	17
18	Test	18
NC		NC
20	Data Terminal Ready	20
NC		NC
22	Ring Indicator	22
23	Data Signal Rate Selector	23
NC		NC
25	Test Indicate	25

Connector Specifications

Personal Computer Hardware Reference Library

IBM Personal Computer AT Communications Cable

IBM PERSONAL COMPUTER AT COMMUNICATIONS CABLE

Contents

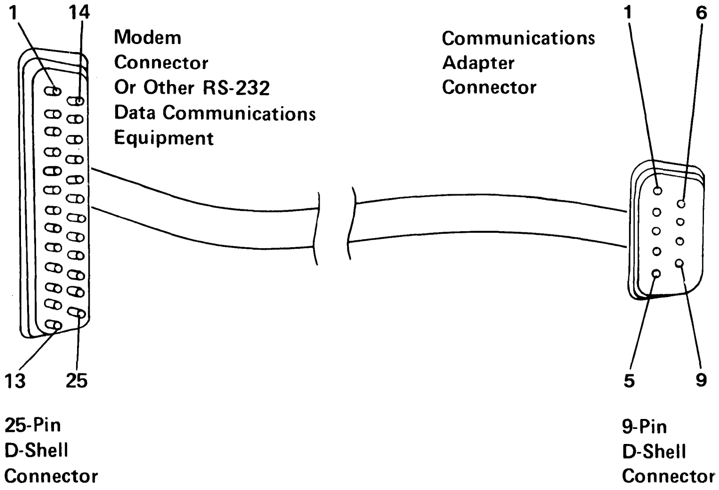
Notes:

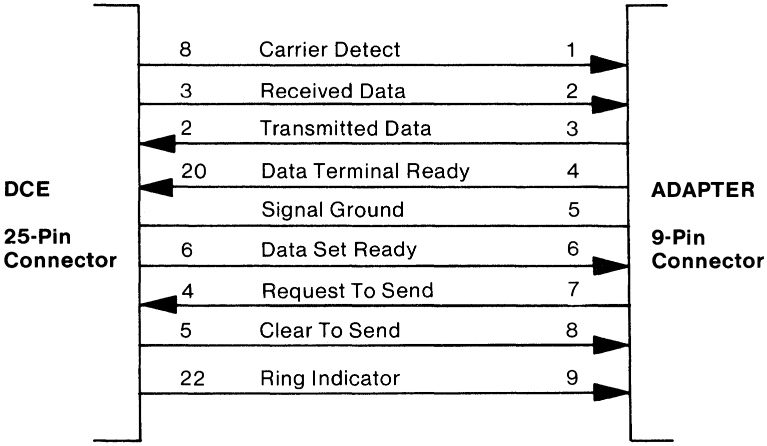
Description

The IBM Personal Computer AT Communications Cable cable is for connection of an IBM communications adapter with a 9-pin D-shell connector to a modem or other RS-232C DCE (data communications equipment). It is fully shielded and provides a high quality, low noise channel for interface between the communications adapter and DCE.

Specifications

One connector is a 9-pin D-shell connector and the other is a 25-pin D-shell connector. The pin numbering and connector specifications follow.


1
14
Modem Connector Or Other RS-232 Data Communications Equipment
13
25
25-Pin D-Shell Connector
Communications Adapter Connector
1
6
5
9
9-Pin D-Shell Connector



8	Carrier Detect	1	→
3	Received Data	2	→
2	Transmitted Data	3	←
20	Data Terminal Ready	4	←
	Signal Ground	5	
6	Data Set Ready	6	→
4	Request To Send	7	←
5	Clear To Send	8	→
22	Ring Indicator	9	→

Note: All other pins on the 25-pin connector are not used.

IBM

The Personal Computer
Hardware Library

Reader's Comment Form

Technical Reference
Options and Adapters **6137806**

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer family of products or programs for the IBM Personal Computer family of products, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Tape

Please do not staple

Tape

UPDATE NUMBER 5

for the

IBM Technical Reference

Options and Adapters

(Part Number 6322509)

Note: All updates received prior to Update Number 5 must be inserted before continuing.

If you have an IBM Personal Computer AT *Technical Reference* manual, replace the following modules in your manual. If you **do not** have an IBM Personal Computer AT *Technical Reference* manual, insert the following modules in your manual.

1. "Adapters" section:

   - IBM Personal Computer AT Serial/Parallel Adapter

   - IBM Personal Computer AT Fixed Disk and Diskette Adapter.

2. In the "Cables and Connectors" section, insert or replace the IBM Personal Computer AT Communications Cable module.

3. "Memory Expansion" section:

   - IBM Personal Computer AT 128KB Memory Expansion Option

   - IBM Personal Computer AT 512KB Memory Expansion Option.

August 31, 1984

4. In the "Miscellaneous" section, insert or replace the IBM Personal Computer AT Prototype Adapter module.

5. "Storage Devices" section:

   - IBM Personal Computer AT High Capacity Diskette Drive

   - IBM Personal Computer AT Double Sided Diskette Drive

   - IBM Personal Computer AT 20MB Fixed Disk Drive.

IBM®

International Business Machines Corporation

P.O. Box 1328-W
Boca Raton, Florida 33432

6137806

Printed in United States of America

IBM

Technical Reference

Options and Adapters

Volume 3

6322522