Contents

Description .................................... 1

Major Components ............................. 3
  System-Bus Interface ....................... 4
  Microprocessor Section ..................... 6
  Video Control Generator Section ........... 8
  Emulator Address Control ................... 11
  Graphics Emulator .......................... 13
  Display Memory ............................. 15
  Look-Up Table and Video Output Section ... 18
  Timing and Control Section ................ 19

Emulator Modes .............................. 20
  Alphanumeric Mode ......................... 20
  Graphics Mode .............................. 24
  Description of Basic Operations .......... 28

High-Function Graphics Mode ................ 29
  Alphanumeric Operation .................... 29
  Graphics Operation ........................ 30
  Description of Basic Operations .......... 32

Programming Considerations ................ 33
  Emulator Programming Considerations ...... 33
  Programming the 6845 CRT Controller ...... 33

Programming the Mode Control and Status
  Registers ................................. 35
  Color-Select Register ...................... 36
  Mode-Select Register ...................... 38
  Status Register ........................... 41
  Sequence of Events for Changing Modes ... 42
  Memory Requirements ....................... 42

High-Function Graphics Programming
  Considerations ........................... 43
  Coordinate Space ........................ 45
  Video Generation ........................ 56
  Display Control .......................... 58
  Drawing Primitives ......................... 63
  Text ........................................ 69
  Command Lists ............................ 71
  Look-Up Table ............................. 73
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.

[Block diagram image]

August 15, 1984

8 Professional Graphics Controller

© Copyright IBM Corporation 1984
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.
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.

<table>
<thead>
<tr>
<th>Display-Character Code Byte</th>
<th>Attribute Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

The attribute byte definitions are:

- **7 6 5 4 3 2 1 0**
- **B R G B I R G B**
  - Foreground Color
  - Foreground Intensity
  - Background Color
  - Foreground Blinking
The following table provides a summary of available colors.

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Blue</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Green</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Cyan</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Brown</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Gray</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Light Blue</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Light Green</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Light Cyan</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Light Red</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Light Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Modes</th>
<th>Number of Colors Available (Includes Background Color)</th>
</tr>
</thead>
<tbody>
<tr>
<td>320 x 200</td>
<td>4 Colors Total</td>
</tr>
<tr>
<td></td>
<td>1 of 16 for Background and</td>
</tr>
<tr>
<td></td>
<td>1 of Green, Red, or Brown or</td>
</tr>
<tr>
<td></td>
<td>1 of Cyan, Magenta, or White</td>
</tr>
<tr>
<td>640 x 200</td>
<td>Black-and-white only</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th></th>
<th>7 6</th>
<th>5 4</th>
<th>3 2</th>
<th>1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C1 C0</td>
<td>C1 C0</td>
<td>C1 C0</td>
<td>C1 C0</td>
</tr>
<tr>
<td>First</td>
<td>Display</td>
<td>Second</td>
<td>Display</td>
<td>Third</td>
</tr>
<tr>
<td>PEL</td>
<td>PEL</td>
<td>PEL</td>
<td>PEL</td>
<td>PEL</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Memory Address (in hex)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>B9F3F</td>
<td>Even Scans (0,1,4,5,8,9...198) 8,000 bytes</td>
</tr>
<tr>
<td>B8000</td>
<td>Not Used</td>
</tr>
<tr>
<td>BA000</td>
<td>Odd Scans (2,3,6,7,10,11...199) 8,000 bytes</td>
</tr>
<tr>
<td>BBF3F</td>
<td>Not Used</td>
</tr>
<tr>
<td>BBFFF</td>
<td></td>
</tr>
</tbody>
</table>

Address hex B8000 contains PEL information for the upper-left corner of the display.
• Determines color selection by the following logic:

<table>
<thead>
<tr>
<th>C1</th>
<th>C0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Dot takes on the color of 1 of 16 preselected background colors</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Selects first color of preselected Color Set 1 or Color Set 2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Selects second color of preselected Color Set 1 or Color Set 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Selects third color of preselected Color Set 1 or Color Set 2</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Color Set 1</th>
<th>Color Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color 1 is green</td>
<td>Color 1 is cyan</td>
</tr>
<tr>
<td>Color 2 is red</td>
<td>Color 2 is magenta</td>
</tr>
<tr>
<td>Color 3 is brown</td>
<td>Color 3 is white</td>
</tr>
</tbody>
</table>
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:

[Diagram of 8 PELs format]

First Display PEL
Second Display PEL
Third Display PEL
Fourth Display PEL
Fifth Display PEL
Sixth Display PEL
Seventh Display PEL
Eighth Display PEL
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.

<table>
<thead>
<tr>
<th>Memory Address (in hex)</th>
<th>Display Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>B8000</td>
<td>Character Code A</td>
</tr>
<tr>
<td>B8001</td>
<td>Attribute A</td>
</tr>
<tr>
<td>B8002</td>
<td>Character Code B</td>
</tr>
<tr>
<td>B8003</td>
<td>Attribute B</td>
</tr>
<tr>
<td>B87CE</td>
<td>Character Code X</td>
</tr>
<tr>
<td>B87CF</td>
<td>Attribute X</td>
</tr>
</tbody>
</table>

(Example of a 40 by 25 Screen)

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.

<table>
<thead>
<tr>
<th>4 Bits</th>
<th>4 Bits</th>
<th>4 Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>Green</td>
<td>Blue</td>
</tr>
<tr>
<td>1 PEL</td>
<td>1 PEL</td>
<td>1 PEL</td>
</tr>
<tr>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
</tr>
</tbody>
</table>

August 15, 1984
© Copyright IBM Corporation 1984
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.

<table>
<thead>
<tr>
<th>Address Register</th>
<th>Register Number</th>
<th>Register Type</th>
<th>Units</th>
<th>I/O</th>
<th>40 by 25 Alpha-numeric</th>
<th>80 by 25 Alpha-numeric</th>
<th>Graphic Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>R4</td>
<td>Vertical Total</td>
<td>Character Row</td>
<td>Write Only</td>
<td>1F</td>
<td>1F</td>
<td>1F</td>
</tr>
<tr>
<td>5</td>
<td>R5</td>
<td>Vertical Total Adj ust</td>
<td>Scan Line</td>
<td>Write Only</td>
<td>06</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>6</td>
<td>R6</td>
<td>Vertical Displayed</td>
<td>Character Row</td>
<td>Write Only</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>R7</td>
<td>Vertical Sync Position</td>
<td>Character Row</td>
<td>Write Only</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
</tr>
<tr>
<td>A</td>
<td>R10</td>
<td>Cursor Start</td>
<td>Scan Line</td>
<td>Write Only</td>
<td>06</td>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>B</td>
<td>R11</td>
<td>Cursor End</td>
<td>Scan Line</td>
<td>Write Only</td>
<td>07</td>
<td>07</td>
<td>07</td>
</tr>
<tr>
<td>C</td>
<td>R12</td>
<td>Start Address(H)</td>
<td>-</td>
<td>Write Only</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>D</td>
<td>R13</td>
<td>Start Address(L)</td>
<td>-</td>
<td>Write Only</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>E</td>
<td>R14</td>
<td>Cursor Address(H)</td>
<td>-</td>
<td>Read/Write</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
<tr>
<td>F</td>
<td>R15</td>
<td>Cursor Address(L)</td>
<td>-</td>
<td>Read/Write</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
</tbody>
</table>

**Note:** All register values are in hexadecimal.

August 15, 1984

© Copyright IBM Corporation 1984
Programming the Mode Control and Status Registers

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

<table>
<thead>
<tr>
<th>Function of Register</th>
<th>Hex Address</th>
<th>A9</th>
<th>A8</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode Control Register (D0)</td>
<td>3D8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Color Select Register (D0)</td>
<td>3D9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Status Register (D1)</td>
<td>3DA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6845 Index Register</td>
<td>3D4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6845 Data Register</td>
<td>3D5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>C1</th>
<th>C0</th>
<th>Colors Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Background (Defined by bits 0-3 of port hex 3D9)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Green</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Brown</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C1</th>
<th>C0</th>
<th>Colors Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Background (Defined by bits 0-3 of port hex 3D9)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Cyan</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C1</th>
<th>C0</th>
<th>Colors Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Background</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Cyan</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>80 x 25 alphanumeric mode</td>
</tr>
<tr>
<td>1</td>
<td>Graphics select</td>
</tr>
<tr>
<td>2</td>
<td>Black/white select</td>
</tr>
<tr>
<td>3</td>
<td>Enable video signal</td>
</tr>
<tr>
<td>4</td>
<td>640 x 200 black/white mode</td>
</tr>
<tr>
<td>5</td>
<td>Change background intensity to blink bit</td>
</tr>
<tr>
<td>6</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>Not used</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Mode-Select Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 1 0 1</td>
<td>40 x 25 Alphanumeric Black-and-White</td>
</tr>
<tr>
<td>0 0 0 1 0 1</td>
<td>40 x 25 Alphanumeric Color</td>
</tr>
<tr>
<td>1 0 1 1 0 1</td>
<td>80 x 25 Alphanumeric Black-and-White</td>
</tr>
<tr>
<td>1 0 0 1 0 1</td>
<td>80 x 25 Alphanumeric Color</td>
</tr>
<tr>
<td>0 1 1 1 0 z</td>
<td>320 x 200 Black-and-White Graphics</td>
</tr>
<tr>
<td>0 1 0 1 0 z</td>
<td>320 x 200 Color Graphics</td>
</tr>
<tr>
<td>0 1 1 1 1 z</td>
<td>640 x 200 Black-and-White Graphics</td>
</tr>
</tbody>
</table>

- 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.

<table>
<thead>
<tr>
<th>Bit 0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Enable</td>
<td></td>
</tr>
<tr>
<td>Bit 1</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Vertical Sync</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Not Used</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Read/Write Memory Address</th>
<th>Space (in hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01000</td>
<td>System Read/Write Memory</td>
</tr>
<tr>
<td>A0000</td>
<td></td>
</tr>
<tr>
<td>B8000</td>
<td>Display Buffer (16K Bytes)</td>
</tr>
<tr>
<td>BBFFF</td>
<td></td>
</tr>
<tr>
<td>C0000</td>
<td></td>
</tr>
</tbody>
</table>
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

Window Clipping and Viewport Transformation

Standard 2D Draw Routines

2D Commands
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.

![Diagram of 2D 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.

![Diagram of 3D 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.

![Viewport Diagram](image)

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.

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

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

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:

\[
\begin{bmatrix}
X_{\text{view}}, Y_{\text{view}}, Z_{\text{view}}, 1 \\
\end{bmatrix} =
\begin{bmatrix}
X_{\text{virtual}}, Y_{\text{virtual}}, Z_{\text{virtual}}, 1 \\
\end{bmatrix} \times [M] \times [\text{VRP}] \times [V]
\]

[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:

\[
\begin{bmatrix}
x & x & x & 0 \\
x & x & x & 0 \\
x & x & x & 0 \\
x & x & x & 1 \\
\end{bmatrix}
\]

Otherwise, the result will have the form:

\[
\begin{bmatrix}
X_{\text{view}}, Y_{\text{view}}, Z_{\text{view}}, Q \\
\end{bmatrix}
\]

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.
Rotation involves the matrix operation,

\[
[M(\text{new})] = [M(\text{old})] \times [M(\text{rst})]
\]

\([M(\text{rst})]\) 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 PROJECT 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.

\[
\begin{align*}
X_{2dvir} &= \frac{DISTAN}{DISTAN - Z} \times X_{view} \times \frac{WINDOW \text{ DIAGONAL}}{2 \times DISTAN \times \tan(PROJECT)} \\
Y_{2dvir} &= \frac{DISTAN}{DISTAN - Z} \times Y_{view} \times \frac{WINDOW \text{ DIAGONAL}}{2 \times DISTAN \times \tan(PROJECT)}
\end{align*}
\]

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 PROJECT 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:

\[
\begin{align*}
X_{2dvir} &= X_{view} \\
Y_{2dvir} &= Y_{view}
\end{align*}
\]
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.

August 15, 1984

58 Professional Graphics Controller © Copyright IBM Corporation 1984
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:

<table>
<thead>
<tr>
<th>Word Order</th>
<th>Pattern</th>
<th>Bit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>XXXX XXXX XXXX XXXX</td>
<td>62415</td>
</tr>
<tr>
<td>E</td>
<td>XXXX XXXX XXXX XXXX</td>
<td>31207</td>
</tr>
<tr>
<td>D</td>
<td>XXXX XXXX XXXX XXXX</td>
<td>15603</td>
</tr>
<tr>
<td>C</td>
<td>X XXXX XXXX XXXX X</td>
<td>40569</td>
</tr>
<tr>
<td>B</td>
<td>XXX XXXX XXXX XXXX</td>
<td>53057</td>
</tr>
<tr>
<td>A</td>
<td>XXX XXXX XXXX XXXX</td>
<td>59294</td>
</tr>
<tr>
<td>9</td>
<td>XXX XXXX XXXX XXXX</td>
<td>62415</td>
</tr>
<tr>
<td>8</td>
<td>XXX XXXX XXXX XXXX</td>
<td>31207</td>
</tr>
<tr>
<td>7</td>
<td>XXX XXXX XXXX XXXX</td>
<td>15603</td>
</tr>
<tr>
<td>6</td>
<td>X XXXX XXXX XXXX X</td>
<td>40569</td>
</tr>
<tr>
<td>5</td>
<td>XX XXXX XXXX XXXX</td>
<td>53057</td>
</tr>
<tr>
<td>4</td>
<td>XXX XXXX XXXX XXXX</td>
<td>59294</td>
</tr>
<tr>
<td>3</td>
<td>XXX XXXX XXXX XXXX</td>
<td>62415</td>
</tr>
<tr>
<td>2</td>
<td>XXX XXXX XXXX XXXX</td>
<td>31207</td>
</tr>
<tr>
<td>1</td>
<td>XXX XXXX XXXX XXXX</td>
<td>15603</td>
</tr>
<tr>
<td>0</td>
<td>X XXXX XXXX XXXX X</td>
<td>40569</td>
</tr>
<tr>
<td></td>
<td>F E D C B A 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>
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.

```
1 1 1 1 0 0 0 0
```

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 PROJECT.

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:

![Rectangle Diagram]

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.
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 $H = 1, V = 1$.

Alterating 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 |
| Line Number 4 | X | X |   | X |
| Line Number 3 | X |   |   | X |
| Line Number 2 | X | X | X | X |
| Line Number 1 | X |   |   | X |
```

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 0 0 0 0 1
1 1 1 0 0 0 0 1 1 1 0 0 0 0 1
0 0 1 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
CLEN

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.

August 15, 1984
© Copyright IBM Corporation 1984

Professional Graphics Controller 71
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.

---

August 15, 1984
© Copyright IBM Corporation 1984 Professional Graphics Controller 73
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.

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

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.

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

August 15, 1984
© Copyright IBM Corporation 1984
Professional Graphics Controller 77
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 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.

<table>
<thead>
<tr>
<th>Memory Address (in hex)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6000</td>
<td>Output FIFO (256 bytes)</td>
</tr>
<tr>
<td>C6100</td>
<td>Input FIFO (256 bytes)</td>
</tr>
<tr>
<td>C6200</td>
<td>Error FIFO (256 bytes)</td>
</tr>
<tr>
<td>C6300</td>
<td>Output FIFO Write Pointer</td>
</tr>
<tr>
<td>C6301</td>
<td>Output FIFO Read Pointer</td>
</tr>
<tr>
<td>C6302</td>
<td>Input FIFO Write Pointer</td>
</tr>
<tr>
<td>C6303</td>
<td>Input FIFO Read Pointer</td>
</tr>
<tr>
<td>C6304</td>
<td>Error FIFO Write Pointer</td>
</tr>
<tr>
<td>C6305</td>
<td>Error FIFO Read Pointer</td>
</tr>
<tr>
<td>C6306</td>
<td>Cold Restart Flag</td>
</tr>
<tr>
<td>C6307</td>
<td>Warm Restart Flag</td>
</tr>
<tr>
<td>C6308</td>
<td>Error Enable Flag</td>
</tr>
</tbody>
</table>
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
- PROJECT (PRO) Projection
- 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

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
CLEARSCREEN (Clear Screen)

Purpose: Clear the screen to a given color.

Command: CLEARSCREEN 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 \textit{flag} is 1 or any odd number, and disabled when \textit{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:

\begin{verbatim}
ASCII: CH 0
HEX: AA 01
\end{verbatim}

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: DO 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 \( \text{dist} \) 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 0F 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 OF 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 \( \text{lowx} \) \( \text{highx} \) \( \text{lowfracx} \) \( \text{highfracx} \) \( \text{lowy} \) \( \text{highy} \) \( \text{lowfracy} \) \( \text{highfracy} \)

**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:  DRA W3 x y z

Description:  DRA W3 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
cfracx highfracx
cfrac y high y
cfrac y high y
cfrac z highz
cfrac z high fracz

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.

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

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

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

August 15, 1984
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 0F 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 sz

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
           lowsiz highsz lowfracsz highfracsz

Example:

ASCII: MDS 2 2 2
HEX: 92 02 00 00 80 01 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

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  lowfraxc highfraxc
   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

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
OA 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 \textit{npts} is the number of points, and \textit{dx} and \textit{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 00 00 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 \textit{npts} is the number of points, and \textit{x}, \textit{y}, and \textit{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:

\begin{verbatim}
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 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
\end{verbatim}

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 . . . . . . . . .

**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 00 00 0A 00 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 \textit{angle} is 0, the projection is orthographic parallel (non-oblique). Otherwise, the projection is perspective, with \textit{angle} being the view angle (default is 60). The range of \textit{angle} is 0 to 179 degrees.

Short Form:  PRO angle

Hex Format:  B0  angle

Example:

\text{ASCII: PR 0}
\text{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 CO

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.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AREAPT</td>
<td>65535 16 times</td>
</tr>
<tr>
<td>2</td>
<td>CLIPH</td>
<td>Flag = 0</td>
</tr>
<tr>
<td>3</td>
<td>CLIPY</td>
<td>Flag = 0</td>
</tr>
<tr>
<td>4</td>
<td>COLOR</td>
<td>Value = 255</td>
</tr>
<tr>
<td>5</td>
<td>DISPLA</td>
<td>No change after a RESETF</td>
</tr>
<tr>
<td>6</td>
<td>DISTAN</td>
<td>Distance = 500</td>
</tr>
<tr>
<td>7</td>
<td>DISTH</td>
<td>Distance = -30000</td>
</tr>
<tr>
<td>8</td>
<td>DISTY</td>
<td>Distance = 30000</td>
</tr>
<tr>
<td>9</td>
<td>FILMSK</td>
<td>Mask = 255</td>
</tr>
<tr>
<td>10</td>
<td>LINFUN</td>
<td>Function = 0</td>
</tr>
<tr>
<td>11</td>
<td>LNPAT</td>
<td>Pattern = 65535</td>
</tr>
<tr>
<td>12</td>
<td>MASK</td>
<td>Mask = 255</td>
</tr>
<tr>
<td>13</td>
<td>MDORG</td>
<td>OX = OY = OZ = 0</td>
</tr>
<tr>
<td>14</td>
<td>2D current point</td>
<td>X = Y = 0</td>
</tr>
<tr>
<td>15</td>
<td>3D current point</td>
<td>X = Y = Z = 0</td>
</tr>
<tr>
<td>16</td>
<td>PRMFIL</td>
<td>Flag = 0</td>
</tr>
<tr>
<td>17</td>
<td>PROJCT</td>
<td>Angle = 60</td>
</tr>
<tr>
<td>18</td>
<td>TANGLE</td>
<td>Angle = 0</td>
</tr>
<tr>
<td>19</td>
<td>TJUST</td>
<td>H = V = 1</td>
</tr>
<tr>
<td>20</td>
<td>TSIZE</td>
<td>Size = 8</td>
</tr>
<tr>
<td>21</td>
<td>VWPORT</td>
<td>0, 639, 0, 479</td>
</tr>
<tr>
<td>22</td>
<td>VWRPT</td>
<td>X = Y = Z = 0</td>
</tr>
<tr>
<td>23</td>
<td>WINDOW</td>
<td>-320, 319, -240, 239</td>
</tr>
<tr>
<td>24</td>
<td>Transformed 3D</td>
<td>X = Y = Z = 0</td>
</tr>
<tr>
<td></td>
<td>current point</td>
<td></td>
</tr>
</tbody>
</table>
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 6F 2A 2A

**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:**

```
TSIZE 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: AO

Example:

ASCII: VWI

HEX: AO

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 lowfraxx highfraxx
             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.

<table>
<thead>
<tr>
<th>Color</th>
<th>Default Values (in Hex) for State 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black to Grey to White</td>
<td>000 111 222 333 444 555 666 777</td>
</tr>
<tr>
<td></td>
<td>888 999 AAA B88 CCC DDD EEE FFF</td>
</tr>
<tr>
<td>Black to Red to White</td>
<td>000 200 400 600 800 000 200 400</td>
</tr>
<tr>
<td></td>
<td>F00 F22 F44 F66 F88 F00 F22 F44</td>
</tr>
<tr>
<td>Black to Red-magenta to White</td>
<td>000 201 402 603 804 005 205 405</td>
</tr>
<tr>
<td></td>
<td>F08 F29 F4A F68 F8C F08 F29 F4A</td>
</tr>
<tr>
<td>Black to Magenta to White</td>
<td>000 202 404 606 808 009 209 409</td>
</tr>
<tr>
<td></td>
<td>F0F F2F F4F F6F F8F F0F F2F F4F</td>
</tr>
<tr>
<td>Black to Magenta-blue to White</td>
<td>000 102 204 306 408 50A 60C 70E</td>
</tr>
<tr>
<td></td>
<td>80F 92F A4F B6F C8F DAF ECF EFF</td>
</tr>
<tr>
<td>Black to Blue to White</td>
<td>000 002 004 008 000 000 000 000</td>
</tr>
<tr>
<td></td>
<td>00F 22F 44F 66F 88F 00F 22F 44F</td>
</tr>
<tr>
<td>Black to Blue-cyan to White</td>
<td>000 012 042 036 048 05A 06C 07E</td>
</tr>
<tr>
<td></td>
<td>08F 29F 4AF 6BF 8CF ADF ECF EFF</td>
</tr>
<tr>
<td>Black to Cyan to White</td>
<td>000 022 044 066 088 00A 00C 00E</td>
</tr>
<tr>
<td></td>
<td>0FF 2FF 4FF 6FF 8FF AFF CFF EFF</td>
</tr>
<tr>
<td>Black to Cyan-green to White</td>
<td>000 021 042 063 084 0A5 0C6 0E7</td>
</tr>
<tr>
<td></td>
<td>0F8 2F9 4FA 6FB 8FC AFD DFE EFF</td>
</tr>
<tr>
<td>Black to Green to White</td>
<td>000 020 040 060 080 0A0 0C0 0E0</td>
</tr>
<tr>
<td></td>
<td>0F0 2F2 4F4 6F6 8F8 AFA CFC EFE</td>
</tr>
<tr>
<td>Black to Green-yellow to White</td>
<td>000 120 240 360 480 5A0 6C0 7E0</td>
</tr>
<tr>
<td></td>
<td>8F0 9F2 A4F B86 C8F DAF EFC EFF</td>
</tr>
<tr>
<td>Black to Yellow to White</td>
<td>000 220 440 660 880 A0A C00 E00</td>
</tr>
<tr>
<td></td>
<td>FF0 FF2 FF4 FF6 F8F FFA FFC FFE</td>
</tr>
<tr>
<td>Black to Yellow-red to White</td>
<td>000 210 420 630 840 A50 C60 E70</td>
</tr>
<tr>
<td></td>
<td>FF8 FF9 F4A F68 FC8 FAD FCE FFE</td>
</tr>
<tr>
<td>Black to Unsaturated Red to White</td>
<td>000 211 422 633 844 A55 C65 E77</td>
</tr>
<tr>
<td></td>
<td>F88 F99 F4A F68 F86 FFA FDC FEE FFF</td>
</tr>
<tr>
<td>Black to Unsaturated Green to White</td>
<td>000 121 242 363 484 5A5 6C6 E77</td>
</tr>
<tr>
<td></td>
<td>8F8 9F9 AFA B8F CFC DFD EFE FFF</td>
</tr>
<tr>
<td>Black to Unsaturated Blue to White</td>
<td>000 112 224 336 448 5A5 6C6 E7E</td>
</tr>
<tr>
<td></td>
<td>88F 99F AAF BBF CCF DDF EEF FFF</td>
</tr>
</tbody>
</table>
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.

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

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Bit Value</th>
<th>Intensity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>1 0</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>1 1</td>
<td>15</td>
</tr>
</tbody>
</table>

### 3-Bit Intensity Values

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Bit Value</th>
<th>Intensity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0 1 0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0 1 1</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>1 0 1</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>1 1 0</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>1 1 1</td>
<td>15</td>
</tr>
</tbody>
</table>

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:

- 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 G 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 B B B).
The following table shows the default values for state 2.

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>003</th>
<th>005</th>
<th>007</th>
<th>009</th>
<th>00B</th>
<th>00D</th>
<th>00F</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>003</td>
<td>005</td>
<td>007</td>
<td>009</td>
<td>00B</td>
<td>00D</td>
<td>00F</td>
</tr>
<tr>
<td>030</td>
<td>033</td>
<td>035</td>
<td>037</td>
<td>039</td>
<td>03B</td>
<td>03D</td>
<td>03F</td>
<td></td>
</tr>
<tr>
<td>050</td>
<td>053</td>
<td>055</td>
<td>057</td>
<td>059</td>
<td>05B</td>
<td>05D</td>
<td>05F</td>
<td></td>
</tr>
<tr>
<td>070</td>
<td>073</td>
<td>075</td>
<td>077</td>
<td>079</td>
<td>07B</td>
<td>07D</td>
<td>07F</td>
<td></td>
</tr>
<tr>
<td>090</td>
<td>093</td>
<td>095</td>
<td>097</td>
<td>099</td>
<td>09B</td>
<td>09D</td>
<td>09F</td>
<td></td>
</tr>
<tr>
<td>0B0</td>
<td>0B3</td>
<td>0B5</td>
<td>0B7</td>
<td>0B9</td>
<td>0BB</td>
<td>0BD</td>
<td>0BF</td>
<td></td>
</tr>
<tr>
<td>0D0</td>
<td>0D3</td>
<td>0D5</td>
<td>0D7</td>
<td>0D9</td>
<td>0DB</td>
<td>0DD</td>
<td>0DF</td>
<td></td>
</tr>
<tr>
<td>0F0</td>
<td>0F3</td>
<td>0F5</td>
<td>0F7</td>
<td>0F9</td>
<td>0FB</td>
<td>0FD</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>050</td>
<td>053</td>
<td>055</td>
<td>057</td>
<td>059</td>
<td>05B</td>
<td>05D</td>
<td>05F</td>
<td></td>
</tr>
<tr>
<td>530</td>
<td>533</td>
<td>535</td>
<td>537</td>
<td>539</td>
<td>53B</td>
<td>53D</td>
<td>53F</td>
<td></td>
</tr>
<tr>
<td>550</td>
<td>553</td>
<td>555</td>
<td>557</td>
<td>559</td>
<td>55B</td>
<td>55D</td>
<td>55F</td>
<td></td>
</tr>
<tr>
<td>570</td>
<td>573</td>
<td>575</td>
<td>577</td>
<td>579</td>
<td>57B</td>
<td>57D</td>
<td>57F</td>
<td></td>
</tr>
<tr>
<td>590</td>
<td>593</td>
<td>595</td>
<td>597</td>
<td>599</td>
<td>59B</td>
<td>59D</td>
<td>59F</td>
<td></td>
</tr>
<tr>
<td>5B0</td>
<td>5B3</td>
<td>5B5</td>
<td>5B7</td>
<td>5B9</td>
<td>5BB</td>
<td>5BD</td>
<td>5BF</td>
<td></td>
</tr>
<tr>
<td>5D0</td>
<td>5D3</td>
<td>5D5</td>
<td>5D7</td>
<td>5D9</td>
<td>5DB</td>
<td>5DD</td>
<td>5DF</td>
<td></td>
</tr>
<tr>
<td>5F0</td>
<td>5F3</td>
<td>5F5</td>
<td>5F7</td>
<td>5F9</td>
<td>5FB</td>
<td>5FD</td>
<td>5FF</td>
<td></td>
</tr>
<tr>
<td>A00</td>
<td>A03</td>
<td>A05</td>
<td>A07</td>
<td>A09</td>
<td>A0B</td>
<td>A0D</td>
<td>A0F</td>
<td></td>
</tr>
<tr>
<td>A30</td>
<td>A33</td>
<td>A35</td>
<td>A37</td>
<td>A39</td>
<td>A3B</td>
<td>A3D</td>
<td>A3F</td>
<td></td>
</tr>
<tr>
<td>A50</td>
<td>A53</td>
<td>A55</td>
<td>A57</td>
<td>A59</td>
<td>A5B</td>
<td>A5D</td>
<td>A5F</td>
<td></td>
</tr>
<tr>
<td>A70</td>
<td>A73</td>
<td>A75</td>
<td>A77</td>
<td>A79</td>
<td>A7B</td>
<td>A7D</td>
<td>A7F</td>
<td></td>
</tr>
<tr>
<td>A90</td>
<td>A93</td>
<td>A95</td>
<td>A97</td>
<td>A99</td>
<td>A9B</td>
<td>A9D</td>
<td>A9F</td>
<td></td>
</tr>
<tr>
<td>A0B</td>
<td>A03</td>
<td>A05</td>
<td>A07</td>
<td>A09</td>
<td>A0B</td>
<td>A0D</td>
<td>A0F</td>
<td></td>
</tr>
<tr>
<td>A30</td>
<td>A33</td>
<td>A35</td>
<td>A37</td>
<td>A39</td>
<td>A3B</td>
<td>A3D</td>
<td>A3F</td>
<td></td>
</tr>
<tr>
<td>A50</td>
<td>A53</td>
<td>A55</td>
<td>A57</td>
<td>A59</td>
<td>A5B</td>
<td>A5D</td>
<td>A5F</td>
<td></td>
</tr>
<tr>
<td>A70</td>
<td>A73</td>
<td>A75</td>
<td>A77</td>
<td>A79</td>
<td>A7B</td>
<td>A7D</td>
<td>A7F</td>
<td></td>
</tr>
<tr>
<td>A90</td>
<td>A93</td>
<td>A95</td>
<td>A97</td>
<td>A99</td>
<td>A9B</td>
<td>A9D</td>
<td>A9F</td>
<td></td>
</tr>
<tr>
<td>FF0</td>
<td>FF3</td>
<td>FF5</td>
<td>FF7</td>
<td>FF9</td>
<td>FFB</td>
<td>FFD</td>
<td>FFF</td>
<td></td>
</tr>
</tbody>
</table>

August 15, 1984
© Copyright IBM Corporation 1984  Professional Graphics Controller 173
The following table shows the default values for state 3.

<table>
<thead>
<tr>
<th>Default Values (in Hex) for State 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 003 005 007 009 00B 00D 00F</td>
</tr>
<tr>
<td>050 053 055 057 059 05B 05D 05F</td>
</tr>
<tr>
<td>0A0 0A3 0A5 0A7 0A9 0AB 0AD 0AF</td>
</tr>
<tr>
<td>0F0 0F3 0F5 0F7 0F9 0FB 0FD 0FF</td>
</tr>
<tr>
<td>300 303 305 307 309 30B 30D 30F</td>
</tr>
<tr>
<td>350 353 355 357 359 35B 35D 35F</td>
</tr>
<tr>
<td>3A0 3A3 3A5 3A7 3A9 3AB 3AD 3AF</td>
</tr>
<tr>
<td>3F0 3F3 3F5 3F7 3F9 3FB 3FD 3FF</td>
</tr>
<tr>
<td>500 503 505 507 509 50B 50D 50F</td>
</tr>
<tr>
<td>550 553 555 557 559 55B 55D 55F</td>
</tr>
<tr>
<td>5A0 5A3 5A5 5A7 5A9 5AB 5AD 5AF</td>
</tr>
<tr>
<td>5F0 5F3 5F5 5F7 5F9 5FB 5FD 5FF</td>
</tr>
<tr>
<td>700 703 705 707 709 70B 70D 70F</td>
</tr>
<tr>
<td>750 753 755 757 759 75B 75D 75F</td>
</tr>
<tr>
<td>7A0 7A3 7A5 7A7 7A9 7AB 7AD 7AF</td>
</tr>
<tr>
<td>7F0 7F3 7F5 7F7 7F9 7FB 7FD 7FF</td>
</tr>
<tr>
<td>900 903 905 907 909 90B 90D 90F</td>
</tr>
<tr>
<td>950 953 955 957 959 95B 95D 95F</td>
</tr>
<tr>
<td>9A0 9A3 9A5 9A7 9A9 9AB 9AD 9AF</td>
</tr>
<tr>
<td>9F0 9F3 9F5 9F7 9F9 9FB 9FD 9FF</td>
</tr>
<tr>
<td>B00 B03 B05 B07 B09 B0B B0D B0F</td>
</tr>
<tr>
<td>B50 B53 B55 B57 B59 B5B B5D B5F</td>
</tr>
<tr>
<td>BA0 BA3 BA5 BA7 BA9 BAB BAD BAF</td>
</tr>
<tr>
<td>BF0 BF3 BF5 BF7 BF9 BFB BFD BFF</td>
</tr>
<tr>
<td>D00 D03 D05 D07 D09 D0B D0D D0F</td>
</tr>
<tr>
<td>D50 D53 D55 D57 D59 D5B D5D D5F</td>
</tr>
<tr>
<td>DA0 DA3 DA5 DA7 DA9 DAB DAD DAF</td>
</tr>
<tr>
<td>DF0 DF3 DF5 DF7 DF9 DFB DFD DFF</td>
</tr>
<tr>
<td>F00 F03 F05 F07 F09 F0B F0D F0F</td>
</tr>
<tr>
<td>F50 F53 F55 F57 F59 F5B F5D F5F</td>
</tr>
<tr>
<td>FA0 FA3 FA5 FA7 FA9 FAB FAD FAF</td>
</tr>
<tr>
<td>FF0 FF3 FF5 FF7 FF9 FFB FFD FFF</td>
</tr>
</tbody>
</table>
The following table shows the default values for state 4.

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

August 15, 1984
© Copyright IBM Corporation 1984  Professional Graphics Controller 175
State 5

In state 5, the 8-bit value becomes the arithmetic result of the formula \((R \times 36) + (G \times 6) + B\), 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.

<table>
<thead>
<tr>
<th>Coded RGB Values</th>
<th>Actual Intensity Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
</tr>
</tbody>
</table>
The following table shows the default values for state 5:

<table>
<thead>
<tr>
<th>Default Values (in Hex) for State 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
</tr>
<tr>
<td>036</td>
</tr>
<tr>
<td>06C</td>
</tr>
<tr>
<td>0C0</td>
</tr>
<tr>
<td>0F6</td>
</tr>
<tr>
<td>30C</td>
</tr>
<tr>
<td>360</td>
</tr>
<tr>
<td>396</td>
</tr>
<tr>
<td>3CC</td>
</tr>
<tr>
<td>600</td>
</tr>
<tr>
<td>636</td>
</tr>
<tr>
<td>66C</td>
</tr>
<tr>
<td>6C0</td>
</tr>
<tr>
<td>6F6</td>
</tr>
<tr>
<td>90C</td>
</tr>
<tr>
<td>960</td>
</tr>
<tr>
<td>996</td>
</tr>
<tr>
<td>9CC</td>
</tr>
<tr>
<td>C00</td>
</tr>
<tr>
<td>C36</td>
</tr>
<tr>
<td>C6C</td>
</tr>
<tr>
<td>CC0</td>
</tr>
<tr>
<td>CF6</td>
</tr>
<tr>
<td>F0C</td>
</tr>
<tr>
<td>F60</td>
</tr>
<tr>
<td>F96</td>
</tr>
<tr>
<td>FCC</td>
</tr>
<tr>
<td>000</td>
</tr>
<tr>
<td>000</td>
</tr>
<tr>
<td>000</td>
</tr>
<tr>
<td>000</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Signal Name/Description</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Video</td>
<td>1</td>
</tr>
<tr>
<td>Green Video</td>
<td>2</td>
</tr>
<tr>
<td>Blue Video</td>
<td>3</td>
</tr>
<tr>
<td>Professional Horizontal and Vertical Sync</td>
<td>4</td>
</tr>
<tr>
<td>Mode Control</td>
<td>5</td>
</tr>
<tr>
<td>Ground for Pin 1</td>
<td>6</td>
</tr>
<tr>
<td>Ground for Pin 2</td>
<td>7</td>
</tr>
<tr>
<td>Ground for Pin 3</td>
<td>8</td>
</tr>
<tr>
<td>Ground for Pins 4 and 5</td>
<td>9</td>
</tr>
</tbody>
</table>

Professional Graphics Display

Professional Graphics Controller

August 15, 1984

180 Professional Graphics Controller © Copyright IBM Corporation 1984
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
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
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


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 viewpoint and which lies between these two points. Any part of an object between the hither plane and the viewpoint 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 viewpoint 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

A

absolute draw
  DRAW (2D) 108
absolute move
  MOVE (2D) 135
  MOVE3 (3D) 137
alphanumeric mode 20, 21, 22, 23
alphanumeric operation 29
ARC 86
AREA 87
area fill 87
area fill command description 68
area fill to boundary color 88
area pattern 89
area pattern mask 61
AREABC 88
AREAPT 89
ASCII commands
  ARC 86
  AREA 87
  AREABC 88
  AREAPT 89
  CA 90
  CIRCLE 91
  CLBEG 92
  CLDEL 93
  CLEARS 94
  CLEND 95
  CLIPH 96
  CLIPY 97
  CLOOP 98
  CLRDP 99
  CLRUN 100
  COLOR 101
CONVRT 102
CX 103
DISPLA 104
DISTAN 105
DISTH 106
DISTY 107
DRAW 108
DRAWR 109
DRAWR3 111
DRAW3 110
ELIPSE 112
FILMSK 113
FLAGRD 114
FLOOD 116
IMAGER 117
IMAGEW 118
LINFUN 119
LINPAT 120
list of commands 83, 84, 85
LUT 121
LUTINT 122
LUTRD 123
LUTSAV 124
MASK 125
MATXRD 126
MDIDEN 127
MDMATX 128
MDORG 129
MDROTX 130
MDROTY 131
MDROTZ 132
MDSCAL 133
MDTRAN 134
MOVE 135
MOVER 136
MOVER3 138
MOVE3 137
POINT 139
POINT3 140
POLY 141
POLYR 142
POLYR3 144
POLY3 143
ASCII communications 78, 79

B

basic operations
   emulator 28
      high-function graphics 32
bit planes 60
block diagrams
   display RAM address control 17
   emulator address control 11
   graphics emulator 13
   high-function graphics display memory 15
look-up table and video output section 18
microprocessor section 6
Professional Graphics Controller 2
system-bus interface 4
timing and control section 19
video control generator section 8
CA 90
CIRCLE 91
CLBEG 92
CLDEL 93
clear screen 94
CLARS 94
CLEND 95
clip hither 96
clip yon 97
CLIPH 96
clipping 61
CLIPY 97
CLOOP 98
CLRD 99
CLRUN 100
COLOR 101
color-select register 36, 37
color/fills/patterns
   AREA 87
   AREABC 88
   AREAPT 89
   CLEAR S 94
   COLOR 101
   FILMSK 113
   FLOOD 116
   LINFUN 119
   LINPAT 120
   list of commands 83, 84, 85
   MASK 125
   PRMFIL 145
command list begin 92
command list delete 93
command list description 71, 72
command list end 95
command list loop 98
command list read 99
command list run 100
command lists
   CLBEG 92
   CLDEL 93

Index-4
CLEND 95
CLOOP 98
CLRD 99
CLRUN 100

list of commands 83, 84, 85
communication protocol 80
Communications 78, 79
communications ASCII (command) 90
communications hexadecimal (command) 103

components
display memory 15, 16, 17
display RAM address control 17
emulator address control 11, 12
graphics emulator 13, 14
high-function graphics display memory 15, 16
list of major components 3, 4, 81
look-up table and video output section 18
microprocessor section 6, 7
system-bus interface 4, 5
timing and control section 19
video control generator section 8, 9, 10

connector specifications 180
convert 102
CONVRT 102
coordinate space 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55
coordinate transformations 47
current color 58
current point 57
CX 103

default LUT selections for LUTINT 168
defining commands
AREAPT 89
DISTAN 105
DISTH 106
DISTY 107
list of commands 83, 84, 85
MDMATX 128

August 15, 1984
© Copyright IBM Corporation 1984
MDORG 129
MDTRAN 134
TDEFIN 153
VWMATX 159
VWPORT 160
VWRPT 164
WINDOW 166
DISPLA 104
display 104
display control 58, 59, 60, 62
drawing modes 58
drawing patterns 59
masks 60
primitive fills 59
viewing 62
display memory 15, 16, 17
display RAM address control 17
DISTAN 105
distance 105
distance hither 106
distance yon 107
DISTH 106
DISTY 107
DRAW 108
draw in 3D 110
draw relative 109
draw relative in 3D 111
drawing commands
ARC (2D) 86
CIRCLE (2D) 91
DRAWR3 (3D) 111
DRAW3 (3D) 110
ELIPSE (2D) 112
list of commands 83, 84, 85
POLY (2D) 141
POLYR 142
POLYR3 (3D) 144
POLY3 (3D) 143
RECT (2D) 147
RECTR (2D) 148
SECTOR 151
TEXT 154
TEXTP 155
drawing modes 58
drawing patterns 59, 60
drawing primitives 63, 64, 65, 66, 67, 68
  area fill command description 68
  linear forms 65
  move command description 63
  nonlinear forms 66
  point command description 63
two-dimensional and three-dimensional command format 63
  vectors 64
DRAWR 109
DRAWR3 111
DRAW3 110

E

ELIPSE 112
ellipse 112
emulator
  alphanumeric mode 20, 21, 22, 23
  color-select register 36, 37
  description of basic operations 28
  graphics mode 24, 25, 26, 27
  memory requirements 42
  mode register summary 40
  mode-select register 38
  programming the mode control and status register 35
  programming the 6845 CRT controller 33, 34
  sequence of events for changing modes 42
  status register 41
  320-by-200 color/graphics mode 24
  40-by-25 alphanumeric mode 22
  640-by-200 black-and-white graphics mode 27
  80-by-25 alphanumeric mode 23
emulator address control 11, 12
emulator card logic diagrams 191
error handling 82
F

fill mask 113
FILMSK 113
flag read 114
FLAGRD 114
FLOOD 116

G

graphics emulator 13, 14
graphics mode 24, 25, 26, 27
graphics operation 30, 31

H

hexadecimal commands
    hex AA (CLIPH) 96
    hex AB (CLIPY) 97
    hex AF (CONVRT) 102
    hex A0 (VWIDEN) 158
    hex A1 (VWRPT) 164
    hex A3 (VWROTX) 161
    hex A4 (VWROTY) 162
    hex A5 (VWROTZ) 163
    hex A7 (VWMATX) 159
    hex A8 (DISTH) 106
    hex A9 (DISTY) 107
    hex B0 (PROJECT) 146
    hex B1 (DISTAN) 105
    hex B2 (VWPORT) 160
    hex B3 (WINDOW) 166
    hex C0 (AREA) 87
    hex C1 (AREABC) 88
<table>
<thead>
<tr>
<th>hex</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>DISPLA</td>
<td>104</td>
</tr>
<tr>
<td>D8</td>
<td>IMAGER</td>
<td>117</td>
</tr>
<tr>
<td>D9</td>
<td>IMAGEW</td>
<td>118</td>
</tr>
<tr>
<td>EA</td>
<td>LINPAT</td>
<td>120</td>
</tr>
<tr>
<td>EB</td>
<td>LINFUN</td>
<td>119</td>
</tr>
<tr>
<td>EB</td>
<td>MASK</td>
<td>125</td>
</tr>
<tr>
<td>EC</td>
<td>LUTINT</td>
<td>122</td>
</tr>
<tr>
<td>ED</td>
<td>LUTSAV</td>
<td>124</td>
</tr>
<tr>
<td>EE</td>
<td>LUT</td>
<td>121</td>
</tr>
<tr>
<td>EF</td>
<td>FILMSK</td>
<td>113</td>
</tr>
<tr>
<td>E7</td>
<td>AREAPT</td>
<td>89</td>
</tr>
<tr>
<td>E9</td>
<td>PRMFIL</td>
<td>145</td>
</tr>
<tr>
<td>0F</td>
<td>CLEARS</td>
<td>94</td>
</tr>
<tr>
<td>04</td>
<td>RESETF</td>
<td>149</td>
</tr>
<tr>
<td>05</td>
<td>WAIT</td>
<td>165</td>
</tr>
<tr>
<td>06</td>
<td>COLOR</td>
<td>101</td>
</tr>
<tr>
<td>07</td>
<td>FLOOD</td>
<td>116</td>
</tr>
<tr>
<td>08</td>
<td>POINT</td>
<td>139</td>
</tr>
<tr>
<td>09</td>
<td>POINT3</td>
<td>140</td>
</tr>
<tr>
<td>10</td>
<td>MOVE</td>
<td>135</td>
</tr>
<tr>
<td>11</td>
<td>MOVER</td>
<td>136</td>
</tr>
<tr>
<td>12</td>
<td>MOVE3</td>
<td>137</td>
</tr>
<tr>
<td>13</td>
<td>MOVER3</td>
<td>138</td>
</tr>
<tr>
<td>20</td>
<td>DRAW</td>
<td>108</td>
</tr>
<tr>
<td>21</td>
<td>DRAWR</td>
<td>109</td>
</tr>
<tr>
<td>22</td>
<td>DRAW3</td>
<td>110</td>
</tr>
<tr>
<td>23</td>
<td>DRAWR3</td>
<td>111</td>
</tr>
<tr>
<td>3C</td>
<td>ARC</td>
<td>86</td>
</tr>
<tr>
<td>3D</td>
<td>SECTOR</td>
<td>151</td>
</tr>
<tr>
<td>30</td>
<td>POLY</td>
<td>141</td>
</tr>
<tr>
<td>31</td>
<td>POLYR</td>
<td>142</td>
</tr>
<tr>
<td>32</td>
<td>POLY3</td>
<td>143</td>
</tr>
<tr>
<td>33</td>
<td>POLYR3</td>
<td>144</td>
</tr>
<tr>
<td>34</td>
<td>RECT</td>
<td>147</td>
</tr>
<tr>
<td>35</td>
<td>RECTR</td>
<td>148</td>
</tr>
<tr>
<td>38</td>
<td>CIRCLE</td>
<td>91</td>
</tr>
<tr>
<td>39</td>
<td>ELIPSE</td>
<td>112</td>
</tr>
<tr>
<td>43</td>
<td>CA</td>
<td>90</td>
</tr>
<tr>
<td>43</td>
<td>CX</td>
<td>103</td>
</tr>
<tr>
<td>50</td>
<td>LUTRD</td>
<td>123</td>
</tr>
<tr>
<td>51</td>
<td>FLAGRD</td>
<td>114</td>
</tr>
<tr>
<td>52</td>
<td>MATXRD</td>
<td>126</td>
</tr>
<tr>
<td>70</td>
<td>CLBEG</td>
<td>92</td>
</tr>
</tbody>
</table>
hex 71 (CLEND) 95
hex 72 (CLRUN) 100
hex 73 (CLOOP) 98
hex 74 (CLDEL) 93
hex 75 (CLRD) 99
hex 80 (TEXT) 154
hex 81 (TSIZE) 157
hex 82 (TANGLE) 152
hex 83 (TEXTP) 155
hex 84 (TDEFIN) 153
hex 85 (TJUST) 156
hex 90 (MDIDEN) 127
hex 91 (MDORG) 129
hex 92 (MDSCAL) 133
hex 93 (MDROTX) 130
hex 94 (MDROTY) 131
hex 95 (MDROTZ) 132
hex 96 (MDTRAN) 134
hex 97 (MDMATX) 128

high-function graphics
  alphanumeric operation 29
  ASCII communications 78, 79
  communication protocol 80, 81
  communications 78, 79
  coordinate space 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55
  coordinate transformations 47
  current color 58
  current point 57
  default LUT selections for LUTINT 168
  description of basic operations 32
  error handling 82
  graphics operation 30, 31
  list of commands 83, 84, 85
  modeling matrix 49, 50, 51, 52, 53
  programming considerations 43, 44, 45
  run-length encoding 167
  state 0 168, 169
  state 1 170
  state 255 178
  state 5 176, 177
  states 2-4 171, 173, 174, 175
  three-dimensional hither/yon clipping 54
  three-dimensional transformation 49
three-dimensional viewing to two-dimensional virtual projection 55
two-dimensional transformation 47, 48
video generation 56, 57, 58
viewer reference-point matrix 53
viewing matrix 53
high-function graphics display memory 15, 16

I

image processing 74
image read 117
image transmission
  IMAGER 117
  IMAGEW 118
  list of commands 83, 84, 85
image write 118
IMAGER 117
IMAGEW 118
interface information
  connector specifications 180
  monitor interface 180

L

line function 119
line pattern 120
linear forms 65, 66
LINFUN 119
LINPAT 120
logic diagrams
  emulator card 183, 191
  memory card 183, 196
  processor card 183, 184
look-up table 121
  list of commands 83, 84, 85
LUT 121
LUTINT 122
LUTRD 123
LUTSAV 124
look-up table and video output section 18
look-up table description 73
look-up table initialize 122
look-up table read 123
look-up table save 124
LUT 121
LUTINT 122
LUTRD 123
LUTSAV 124

M

MASK 125
masks 60, 61, 62
  bit planes 60
  clipping 61
matrix read 126
MATXRD 126
MDIDEN 127
MDMATX 128
MDORG 129
MDROTX 130
MDROTY 131
MDROTZ 132
MDSCAL 133
MDTRAN 134
memory card logic diagrams 196
memory requirements 42
microprocessor section 6, 7
mode register summary 40
mode set/read
  CA 90
  CX 103
  DISPLA 104
  FLAGRD 114
  list of commands 83, 84, 85
RESETF 149
WAIT 165
mode-select register 38
modeling identity 127
modeling matrix 49, 50, 51, 52, 53, 128
modeling origin 129
modeling rotate x axis 130
modeling rotate y axis 131
modeling rotate z axis 132
modeling scale 133
modeling transformations
  list of commands 83, 84, 85
  MATXRD 126
  MDIDEN 127
  MDMATX 128
  MDORG 129
  MDROTX 130
  MDROTY 131
  MDROTZ 132
  MDSCAL 133
  MDTRAN 134
modeling translation 134
monitor interface 180
MOVE 135
move command description 63
move in three dimensions 137
move relative 136
move relative in three dimensions 138
MOVER 136
MOVER3 138
MOVE3 137

nonlinear forms 66, 67
POINT 139
point command description 63
point in three dimensions 140
POINT3 140
POLY 141
polygon 141
polygon in three dimensions 143
polygon relative 142
polygon relative in 3D 144
POLYR 142
POLYR3 144
POLY3 143
primitive fill 145
primitive fills 59, 60
PRMFIL 145
processor card logic diagrams 184
programming considerations
  ASCII communications 78, 79
  color-select register 36, 37
  communication protocol 80, 81
  communications 78, 79
  coordinate space 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55
  coordinate transformations 47
  current color 58
  current point 57
  default LUT selections for LUTINT 168
  error handling 82
  list of commands 83, 84, 85
  memory requirements 42
  mode register summary 40
  mode-select register 38
  modeling matrix 49, 50, 51, 52, 53
  programming considerations for the high-function graphics
    mode 43, 44, 45
  programming the mode control and status register 35
  programming the 6845 CRT controller 33, 34
  run-length encoding 167
  sequence of events for changing modes 42
  state 0 168, 169
  state 1 170
state 255 178
state 5 176, 177
states 2-4 171, 173, 174, 175
status register 41
three-dimensional hither/yon clipping 54
three-dimensional transformation 49
three-dimensional viewing to two-dimensional virtual
  projection 55
two-dimensional transformation 47, 48
video generation 56, 57, 58
viewer reference-point matrix 53
viewing matrix 53
programming the mode control and status register 35
programming the 6845 CRT controller 33, 34
PROJCT 146
projection 146

R

read-back commands 75, 76
reading commands
  IMAGER 117
  list of commands 83, 84, 85
  LUTRD 123
  MATXRD 126
RECT 147
rectangle 147
rectangle relative 148
RECTR 148
relative draw
  DRAWR (2D) 109
relative move
  MOVER 136
  MOVER3 (3D) 138
reset commands
  list of commands 83, 84, 85
  MDIDEN 127
  VWIDEN 158
reset flags 149
RESETF 149

August 15, 1984
© Copyright IBM Corporation 1984
rotate commands
   list of commands 83, 84, 85
   MDROTX 130
   MDROTY 131
   MDROTZ 132
   VWROTX 161
   VWROTY 162
   VWROTZ 163
run-length encoding 167

S

save commands
   list of commands 83, 84, 85
   SECTOR 151
select commands
   DISPLA 104
   LINFUN 119
   list of commands 83, 84, 85
sequence of events for changing modes 42
set commands
   CA 90
   CLIPH 96
   CLIPY 97
   COLOR 101
   CX 103
   FILMSK 113
   FLAGRD 114
   LINPAT 120
   list of commands 83, 84, 85
   LUT 121
   LUTSAV 124
   MASK 125
   MDSCAL 133
   POINT (2D) 139
   POINT3 (3D) 140
   PRMFIL 145
   PROJECT 146
   TANGLE 152
   TJUST 156
specifications
  
  power requirements 181
  size 181
  weight 181

  state 0 168, 169
  state 1 170
  state 255 178
  state 5 176, 177
  states 2-4 171, 173, 174, 175
  status register 41
  system reset 77
  system-bus interface 4, 5

T

TANGLE 152
TDEFIN 153

  text 154
    list of commands 83, 84, 85
    TANGLE 152
    TDEFIN 153
    TEXT 154
    TEXTP 155
    TJUST 156
    TSIZE 157
  text angle 152
  text define 153
  text description 69, 70
  text justify 156
  text programmed 155
  text size 157
  TEXTP 155

  three-dimensional drawing
    DRAWR3 111
    DRAW3 110
    MOVER3 138
    MOVE3 137
    POINT3 140
    POLYR3 144
POLY3 143
three-dimensional hither/yon clipping 54
three-dimensional transformation 49
three-dimensional viewing to two-dimensional virtual projection 55
timing and control section 19
TJUST 156
TISZE 157
two-dimensional and three-dimensional command format 63
two-dimensional drawing
   ARC 86
   CIRCLE 91
   DRAW 108
   DRAWR 109
   ELIPSE 112
   MOVE 135
   MOVER 136
   POINT 139
   POLY 141
   POLYR 142
   RECT 147
   RECTR 148
   SECTOR 151
two-dimensional transformation 47, 48

V

vectors 64
video control generator section 8, 9, 10
video generation 56, 57, 58
viewer reference-point matrix 53
viewing 62
viewing identity 158
viewing matrix 53, 159
viewing reference point 164
viewing rotate x axis 161
viewing rotate y axis 162
viewing rotate z axis 163
viewport 160
viewport/window/projection
CLIPH 96
CLIPY 97
CONVRT 102
DISTAN 105
DISTH 106
DISTY 107
PROJECT 146
VWIDEN 158
VWMATX 159
VWPORT 160
VWROTX 161
VWROTY 162
VWROTZ 163
VWRPT 164
WINDOW 166
VWIDEN 158
VWMATX 159
VWPORT 160
VWROTX 161
VWROTY 162
VWROTZ 163
VWPRT 164

W

WAIT 62, 165
WINDOW 166
write commands
   IMAGEW 118
   list of commands 83, 84, 85

Numerals

320-by-200 color/graphics mode 24
40-by-25 alphanumeric mode 22
640-by-200 black-and-white graphics mode 27
80-by-25 alphanumeric mode 23

August 15, 1984
© Copyright IBM Corporation 1984