A professional graphics controller

by K. A. Duke
W. A. Wall

The IBM Professional Graphics Controller and Display were developed to meet the needs of engineers and scientists for an improved graphics capability in the Personal Computer environment. These units provide graphics systems with improved function, resolution, and color range, and at the same time they allow existing productivity software to be executed in an emulation mode. This paper describes the function and discusses the design of the Professional Graphics Controller.

Engineers and scientists have traditionally pioneered the use of new computing technology in their professional activities. In the 1970s engineers readily traded in their slide rules for digital electronic calculators; now they have graduated to personal computers. One aspect of computers that has captured the imagination of these professionals is the ability of personal computers to produce graphics images. Up to now, however, the quality of the images produced by personal computers has not been high enough to satisfy the needs of engineers and scientists.

The IBM Professional Graphics Controller and Display were developed to meet the needs of engineers and scientists for a low-cost, high-function graphics capability. Since many of these professional people use personal computers in their daily work, it was decided to develop the required capability for attachment to the IBM Personal Computer. The Professional Graphics Controller occupies two adjacent expansion slots on the PC bus and may be installed in the Personal Computer AT (PC AT), in the PC XT, or in the Expansion Chassis attached to the original PC.

The Professional Graphics Display uses a raster technique to provide a picture having 480 scan lines, each containing 640 picture elements or pixels. This allows for a display area having a standard aspect ratio of 4:3, with the same pixel spacing horizontally and vertically. The picture is completely scanned sixty times per second to eliminate the appearance of flicker. The scan uses a noninterlaced raster to avoid the inevitable problems of line pairing and the circumstantial flicker effects often apparent in interlaced rasters.

It was required that the display be able to reproduce images of photographic quality for solid modeling and image enhancement purposes. To achieve this quality, it is necessary to provide many more and subtler colors than are offered by simpler displays. Therefore, the Professional Graphics Controller has been designed to provide up to 256 colors in any one picture, selected from a palette of 4096. See Figure 1. This requires an analog connection to the Professional Graphics Display.

Many engineers, scientists, and other potential users of the IBM Professional Graphics Controller and Display already use a number of graphics and other applications that run on the IBM PC. In order to capitalize on that existing software base, the Professional Graphics Controller provides a mode in which

it emulates the IBM Color/Graphics Adapter. This allows the PC system to be controlled and to run both existing programs and new high-function graphics applications while requiring only a single display to occupy desk space. Where space is available and the user prefers it, a separate system control adapter and display can be accommodated.

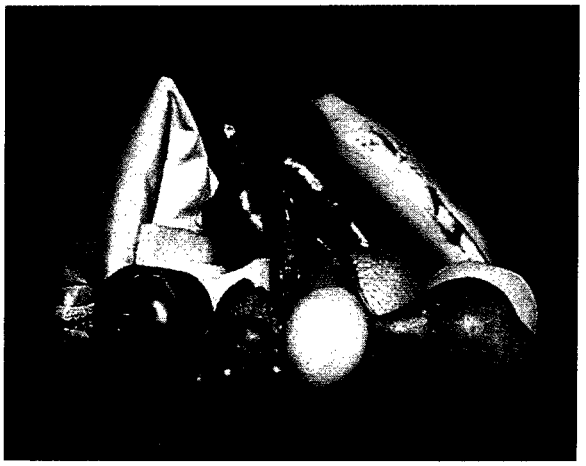
Functional description

The IBM Professional Graphics Controller is in effect two graphics adapters that may be connected, one at a time, to a single screen. One adapter is the high-function graphics controller, which is capable of producing images in 256 different colors with a resolution of 640 × 480 pixels. The other is an emulator that is able to produce the same images that can be developed by the IBM Color/Graphics Adapter and Color Display. Both of these adapters are functional at all times, but only one may be viewed at a time. The selection of which screen is to be viewed is controlled by a command issued to the high-function controller. It has been found convenient, in practice, to arrange for the appropriate commands to be issued automatically by activating a particular key combination, so that the screens may be toggled without interrupting a running program. This is particularly useful if the emulator is used as the system control screen.

The IBM Professional Graphics Controller is activated by transmitting commands to it in the form of character strings. A command consists of a command code followed by a number of parameters, depending on the code. A command may be transmitted in one of two formats. These are known, respectively, as ASCII (character) and HEX (binary) formats.

In the ASCII format, a character string consists of human-readable ASCII characters, using spaces or other suitable characters as delimiters between parameters. The command code is always an alphanumeric string beginning with a letter. Parameters are usually numeric strings. The command code may be transmitted in its long form, consisting of three to six characters, or in a short form, consisting of one to three characters. The long form is easier to read and remember, whereas the short form is easier to type and is transmitted more quickly. For example, the command to move the current drawing point to the x, y coordinates (−320, 200) may be transmitted as follows:

Figure 1 Example of photographic quality



MOVE −320,200.00 or M−320 200

In the HEX format, commands are transmitted in a more compressed form, in which a byte may assume any value from 0 to 255. The command code consists of a single byte, and successive bytes contain the values of the associated parameters. The number and length of parameters are determined by the command code. There are no delimiters between parameters. Commands in this format are very difficult to read and almost impossible to type, but the benefits of the compression afforded by this format make it desirable as an output from utility programs. Commands may be transmitted to the controller to be stored for later execution. A list of commands to be stored is preceded by a **Command List Begin** (CLBEG) command, with an associated list identifier 0–255, and followed by a **Command List End** (CLEND) command. The list may be executed at a later time by issuing a **Command List Run** (CLRUN) command, with the appropriate list identifier. Alternatively, a list may be executed a number of times by issuing a **Command List Loop** (CLOOP) command, with the appropriate list identifier and a loop count parameter. One command list may call another list for execution by including the CLRUN and/or CLOOP commands as required. The nesting of command lists is permitted, but recursion cannot be accommodated.

Command lists are always stored in HEX (binary) form, even though they may be transmitted in ASCII (character) form. Command lists may also be read,

Figure 2 Example of color look-up table and image

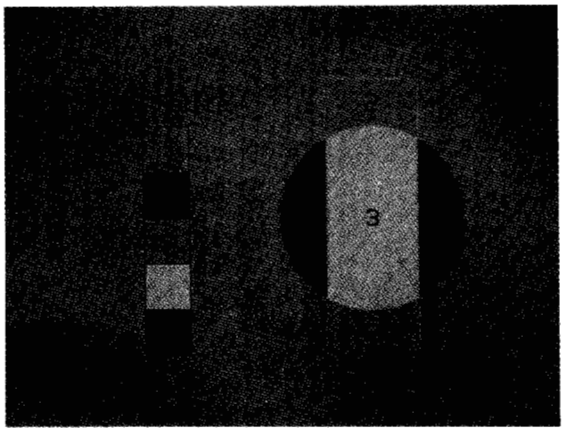


Figure 3 Changing a color image by changing the color look-up table

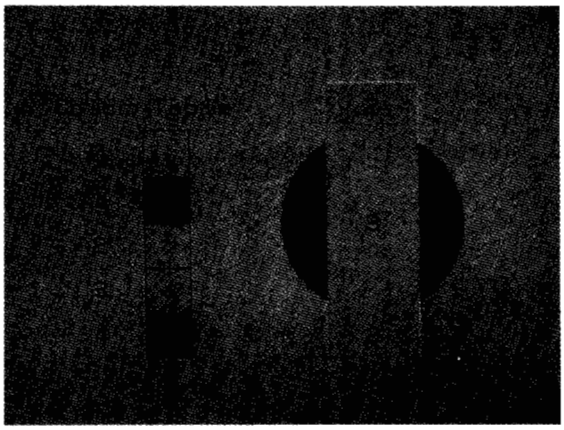


Figure 4 Rendering an object invisible by changing the color look-up table

but only in HEX form, by using the **Command List Read** (CLRD) command with an associated command list identifier.

An image consists of 640 × 480 pixels. Each pixel consists of a single byte and may thus assume any value in the range 0–255. This value is known as the *color value* of the pixel. The color and intensity of any pixel depend on its color value and the setting of a color *look-up table*. The look-up table consists of 256 entries of 12 bits each. Each entry is interpreted as three four-bit parameters specifying the intensities of red, green, and blue to be associated with that color value.

An image may be drawn in certain color values and may be rendered invisible by assigning the same red, green, and blue intensities to those values as to the background. (See Figures 2, 3, and 4.) Alternatively, several images may be drawn on the screen at the same time in different color values and rendered visible one at a time by suitably setting the color look-up table. In this way the animation of predrawn figures may be achieved.

It is often convenient to think of the image as consisting of eight bit planes, each of which has the dimensions 640 × 480 bits. (In this discussion, the bit planes are identified by the binary value of the bits in the plane, i.e., 1, 2, 4, 8, 16, 32, 64, 128.) Masking commands are provided to allow operations to take place in some bit planes while leaving others undisturbed. Using masking in combination with look-up table manipulation makes it possible to achieve a more flexible—though slower—animation effect. The image planes are divided into two four-bit groups. This allows a 16-color image to be drawn in one group, while the other image is rendered visible by the setting of the look-up table. Thus, an image may be displayed from one group of four bit planes protected by the appropriate mask, while a new image is being drawn in the other group. The look-up table is then changed to render the new image visible, and the mask is set to allow the old image to be rewritten.

It is possible by setting appropriate values in the color look-up table to assign priority to the data in certain bit planes. For example, we may assign a set of colors to the color values 0–15. Thus, anything drawn in the lower four bit planes is rendered in those colors. For example, if we wish to give priority to something drawn in bit plane 16, we set all the color values 16–31 to the priority color. Now any-

thing drawn in bit plane 16 is rendered in that color, regardless of the data in the lower four bit planes. This gives the effect of putting bit plane 16 in the foreground and bit planes 1 through 8 in the background. A number of preset look-up table settings are built into the controller. By one of these settings, the first four bit planes become background to the other four bit planes as foreground.

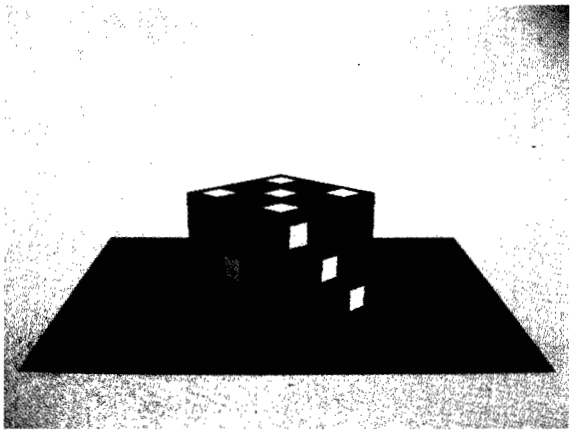
Although this process works with any combination of bit planes, the physical structure of the display memory provides a significant performance benefit when these techniques are applied to these two sets of four bit planes.

Intermediate monochrome levels can be simulated by using a stippling technique.

The Professional Graphics Controller provides a palette of 4096 colors. This is achieved by allowing each of the additive primary colors (red, green, and blue) to assume one of 16 brightness levels. Level 0 is the complete absence of that primary color, whereas level 15 represents its maximum brightness. It sometimes happens, however, that 16 levels do not suffice for a certain application. This is particularly true when displaying monochrome images that have large areas with little detail. Such areas show distinct steps when the brightness changes by one level. In a monochrome image, the brightness levels of the three primary colors are equal for any pixel. Thus, there are only 16 monochrome levels. In some circumstances, it is useful to have intermediate brightness levels, but that requires 31 brightness levels. However, intermediate levels can be simulated by using a stippling technique. In the area to be set to the intermediate level, alternate pixels are set to the two adjacent brightness levels. The alternate pixels are individually too small to be visible.

There are two ways to accomplish the stippling effect. In either case, it is assumed that the monochrome image is represented by brightness levels in the range 0–255, where the least significant three bits are ignored. The brightness level of each pixel is assigned as its color value. The color look-up table is initially set to the 16 brightness levels in blocks of 16 entries, i.e., 0–15 are set to level 0; 16–31 are set to level 1; and so on up to 240–255 set to level 15. With this setting, only the four most significant bits of the color value have any effect.

Figure 5 Example of a three-dimensional image



In the first method, the color look-up table is modified so that a 1 in the fifth bit position of the color value raises the brightness by one level, i.e., 8–15 are set to level 1; 24–31 are set to level 2; and so on, up to 232–239 set to level 15. Note that the fifth bit has no effect on the brightest level. The effect of the fifth bit may now be reduced by a half-level by impressing a pattern of alternate 0s on that bit plane. This is done by setting the plane mask to 8, setting the current color to 0, setting the fill pattern to alternate 1s, and drawing a filled rectangle to cover the entire image. This technique can theoretically be extended to a sixth bit plane by writing a pattern consisting of three 0s to each 1 on that bit plane. This pattern must be arranged so that the ones coincide with the zeros in the first pattern.

In the second method, the stipple pattern is set into an entirely separate bit plane, e.g., bit plane 1. The color look-up table is then set so that only where a 1 occurs in both bit planes 1 and 8 is the color level raised. Thus, colors 9–15 (odd only) are set to level 1; colors 25–31 (odd only) are set to level 2, and so on. This technique has the advantage of not disturbing the original data and allows for improved run-length encoding when the stipple pattern is ignored.

The Professional Graphics Display provides a grid of 640 × 480 coordinate points. However, the Profes-

Figure 6 Example of variable type size and letter spacing

sional Graphics Controller allows the user to work in a much larger coordinate space, any part of which may be mapped onto the screen. Moreover, the user may work in a three-dimensional coordinate system and project the image onto any specified viewing plane with any desired perspective. (See Figure 5.) Coordinates may be expressed with 32-bit precision in the range from −32768.00000 to +32767.99999.

Three-dimensional images are transformed to two-dimensional images that are mapped onto a viewport for display. A viewport is a rectangular area defined within the display screen coordinates 0, 0 to 639, 479 in which all drawing operations take place. A window is a rectangular area within the 32-bit-precision, two-dimensional coordinate space that is mapped onto the current viewport. The window need not be the same shape as the viewport, and different scaling in the x, y dimensions may be apparent.

Three-dimensional images are transformed to two-dimensional images in the following way. First, a view reference point is specified in the three-dimensional modeling coordinate space, the default being the coordinate origin. That point is then transformed to the exact center of the two-dimensional window and thus appears at the center of the viewport. Then the direction of view is specified. The default is the view along the z-axis, and commands exist to rotate the direction about the x-, y- and z-axes. The distance of the viewing point from the view reference point is specified in terms of the modeling coordinate system, and this determines the perspective with which the object is viewed. Finally, the viewing angle is specified, which determines the angle of view represented by the width of the window and viewport. This determines how much of the model is seen. This is the equivalent of specifying the focal length of a lens in photography.

The Professional Graphics Controller provides a built-in text character set and also allows a user to program a custom character set. Both sets are composed of block-style characters. The built-in set provides all the printable characters supported by the IBM Personal Computer. This built-in set may be magnified by an integral number, and the character spacing may be varied. A particular feature of this set is a smoothing technique that is used to eliminate the stair-step effect usually seen in magnified block characters. A programmable character may be specified up to 255 × 255 pixels, but it cannot be magnified. This allows the user to specify mathematical and Greek characters as well as other engineering symbols. It should be noted that all text characters are drawn in the defined current color with a transparent background, i.e., no background color is specified. Variable type size and letter spacing are illustrated in Figure 6.

The Professional Graphics Controller supports a number of status flags that affect the way images are drawn. These include the current drawing coordinates, the current drawing color, and the current line pattern. They also include a flag to cause closed figures to be filled and a representation of the fill pattern. The viewport and window coordinates are similarly recorded, as are the 3D-to-2D transformation parameters. All such flag settings may be read by the PC by issuing appropriate query commands.

The Professional Graphics Controller is able to detect certain error conditions in the incoming command stream, including the presence of parameters that are out of the allowed range or an incorrect number of parameters. When such an error occurs, the Controller can transmit an error message to the PC. This facility may be enabled or disabled by setting a flag in the communications area.

Hardware overview

The Professional Graphics Controller is a two-and-one-half-card set installed in two adjacent slots in an IBM PC. The major functions of the cards are the display memory, the Color/Graphics Adapter emulator, and the CPU. The half card, which provides the emulator function, is installed between the memory and CPU cards.

The memory board of the card set provides the memory for the high-resolution function of the Professional Graphics Controller. This board contains 327 680 bytes of memory organized as forty 64K-bit chips, each of which is addressed as 16 000 four-bit nibbles. (A half-byte is sometimes termed a *nibble*.) The display buffer requires 307 200 bytes of this memory [i.e., one byte per pixel or 640 pixels (or bytes) per line × 480 lines = 307 200 bytes]. The remaining portion of the memory is used for command lists, programmable character fonts, internal variables, and pointers. Also contained on the card is the control circuitry for dynamic RAMs and a clock circuit.

The emulator board of the card set provides the circuitry necessary to emulate the full function of the Color/Graphics Adapter. This allows the Professional Graphics Controller and Display to be used as the primary display on any of the family of IBM Personal Computers.

The CPU board of the card set contains an Intel 8088 microprocessor, which is the heart of the Professional Graphics Controller. This device, together with the necessary firmware, provides the graphics drawing capability. The CPU board also contains 64K bytes of firmware, programmable timing circuits, a high-speed video digital-to-analog converter (VDAC), and 8088 support chips also on this card.

Functional areas of the Professional Graphics Controller

In this paper we consider the Professional Graphics Controller as consisting of the following four major functional areas: display architecture, microcomputer architecture, command set, and emulator.

Display architecture. We now discuss the loading of image data into the display memory and the scanning and outputting of the data to create a picture on the screen of the monitor. This and the following two sections on microcomputer architecture and command set describe the high-resolution function of the Professional Graphics Controller.

Timing necessary for the interleaved CPU and scanner cycle is generated by high-speed programmable logic devices.

The resolution of the display is 640 × 480 pixels, with one byte of memory required to store each pixel. Each byte contains a value between 0 and 255, which is used as an index to a high-speed memory called the *color look-up table* (LUT). In the table, there are 256 values, each 12 bits wide, that contain a representation of the colors that may be displayed on the screen at any one time. The 12 bits are broken into three four-bit fields (or nibbles), one nibble each for the red, green, and blue intensities that make up each pixel. For example, if the first byte in memory is scanned out and has a value of 25, the color stored in word 25 of the color look-up table (LUT) is read and gated to the video digital-to-analog converter (VDAC). The VDAC then converts the value to the proper intensity level for each of the three primary colors, and the pixel in the upper left corner of the screen becomes the appropriate color.

To eliminate flicker on the display, the entire memory is scanned and the screen updated 60 times per second. This combined with monitor specifications requires a pixel rate of 25 MHz. At this rate, eight bits must be accessed from the memory every 40 ns for the scanner alone, and the CPU must also have enough time to update the memory. To overcome this timing problem, a method of interleaving CPU

and scanner accesses to the memory was devised to meet these timing specifications and associated power consumption limitations. These interleaved accesses and the organization of the memory allow this device to operate at the required speed.

The memory is composed of forty 64K-bit chips, each of which is addressed as 16K four-bit nibbles and is organized as five banks of four pixels (or bytes) each. Each time an access to the memory is made, it is made to a bank of four bytes (i.e., four pixels). At the beginning of a cycle, a bank is read by the scanner and latched to be output to the color look-up table; 80 ns later a different bank is ready for an access by the CPU if it is needed. The scanner is then ready to latch four more bytes from the next bank 80 ns later. This cycle continues for 800 ns, giving both the scanner and the CPU access to all memory banks. Each bank of memory is accessed no more than once every 400 ns.

The 8088 can communicate with the IBM PC, read and write to memory, and communicate with the programmable devices.

When the CPU is ready to update the memory, there are several modes that may be used. During a memory access, the CPU may write to a four-pixel bank in any of the following ways: (1) to all four pixels; (2) to only one pixel; (3) from the first pixel to pixel x (where $x = 2$, 3, or 4); or (4) from x to 4 (where $x = 1$, 2, or 3). Additionally, the writes to memory may be accelerated by writing to a 20-pixel group in a similar manner. The CPU may write to a 20-pixel group in any of the following ways: (1) to all 20 pixels; (2) to any bank of four pixels; (3) from the start of a 20-pixel group to the end of any bank of four; or (4) from the start of any bank of four to the end of the 20-pixel group.

Any multiple-pixel writes must write the same data to the memory locations. This is particularly useful when drawing filled figures or flooding the screen.

The timing necessary for the interleaved CPU and scanner cycle is generated by high-speed programmable logic devices. These devices are also used to generate blanking and composite synchronizing signals to drive the monitor.

Microcomputer architecture. The heart of the Professional Graphics Controller is the Intel 8088 microprocessor, which is driven by an 8-MHz clock. The 8088 has access to 64K bytes of read-only memory (ROM) that is organized in two banks of 32K bytes each. The ROMs contain the firmware for the graphics applications. There are also 2K bytes of read-write memory (RAM), which are used as a 1K-byte scratchpad and a 1K-byte area for communications with the IBM PC.

The 8088 drives its data bus, address bus, and control bus, and it supports a HOLD state, READY state, and INTERRUPT state. With these buses and states, the 8088 can communicate with the IBM PC, read and write to memory, and communicate with the programmable devices used in the timing circuits.

The data and address buses of the 8088 are standard. Thus, the least significant eight bits of the address lines are multiplexed signals, with the address coming out of the 8088 latched at the start of a memory cycle and data either sent or received through a bidirectional buffer during the latter half of the memory cycle. All devices accessed by the 8088 are memory mapped.

The control bus is a standard set of 8088 signals used to control data and address flow into and out of the 8088. All of these signals are set to a high-impedance state if the 8088 is put into a HOLD state. The HOLD state occurs if the PC is trying to gain access to the Professional Graphics Controller. The PC communicates with the emulator RAM, with the emulator I/O ports, or with the communications RAM for high-resolution commands and data. The addresses as seen by the PC are the following:

Communications RAM	C6000 hexadecimal to C63FF hexadecimal
Emulator RAM	B8000 hexadecimal to B8FFF hexadecimal
Emulator I/O ports	3Dx hexadecimal

A programmable logic device is used to detect an IBM PC request and to put the 8088 into a HOLD state. It also lowers the PC I/O CH RDY (channel ready) signal to prevent the PC from proceeding to the next

instruction before the current access is complete. The buses are set to their high-impedance state, and the proper device is enabled for access by the PC.

The READY signal is used to notify the 8088 that a device it is trying to access is not ready. If the READY signal goes low, the 8088 remains in an idle state with the buses being driven, waiting for the device to become READY. This happens if a read or write to the emulator or display RAM is taking place. When the device being accessed is ready, the READY signal is raised and operation continues.

Communications. In high-resolution mode, communications between the host computer and the Professional Graphics Controller takes place through a 1K-byte communications RAM. There are three channels set up through which data may pass to the communications area utilizing a first-in-first-out (FIFO) protocol. The FIFO buffers are the following: (1) OUTPUT—for sending commands and parameters to the controller; (2) INPUT—for receiving data from the controller; and (3) ERROR—for receiving errors and warnings generated by the 8088 processor. Each FIFO occupies 256 bytes of the 1K-byte communications area. The remaining portion of the communications memory contains the pointers for the communication protocol, as well as other flags used in the communications between the Professional Graphics Controller and the host computer.

The memory map given in Figure 7 shows addresses as seen by the IBM PC. Each buffer has two pointers associated with it, as seen in Figure 7, a write and a read pointer. Each pointer is eight bits long, and thus the buffers automatically wrap around by incrementing the pointers. The pointers address the next buffer location to be written or read. Thus, if both pointers point to the same location, the buffer is empty. If the read pointer points one location ahead of the write pointer, the buffer is full. Data is written into a buffer by loading a byte into the location indicated by the write pointer (provided the buffer is not full) and incrementing the write pointer. Data is read by reading the byte from the location indicated by the read pointer (provided the buffer is not empty) and incrementing the read pointer. By examining the pointer values, it is possible to effect block transfers of data without incrementing the pointers after each byte. This makes for a considerable improvement in data transfer rate.

Figure 7 Memory map showing addresses as seen by the IBM PC

HEXADECIMAL ADDRESS	DESCRIPTION
C6000-0FF	Output buffer (256 bytes)
C6100-1FF	Input buffer (256 bytes, read only)
C6200-2FF	Error buffer (256 bytes, read only)
C6300	Output FIFO write pointer
C6301	Output FIFO read pointer (read only)
C6302	Input FIFO write pointer (read only)
C6303	Input FIFO read pointer
C6304	Error FIFO write pointer (read only)
C6305	Error FIFO read pointer
C6306	Cold restart flag
C6307	Warm restart flag
C6308	Error enable flag
C630B	Emulator enable flag
C630C	Display request
C630D	Display acknowledge (read only)
C630E	Emulator buffer request
C630F	Emulator buffer acknowledge (read only)
C63DB	Presence test byte
C63F8	Software revision (read only)
C63F9	Software version (read only)

All other locations in the range C6000-C67FF are reserved.

Command set. The Professional Graphics Controller provides the user with an on-board command set to perform manipulations such as the various drawing functions, translations, and rotations. In this section, we categorize the commands into ten areas and give a description of the function of each area.

The command set has been broken down into the following categories:

- Drawing primitives (2D and 3D)
- Image transmissions
- 2D transformations
- 3D transformations
- Text
- Look-up table commands
- Flags
- Readback commands
- Command list commands
- Miscellaneous commands

Drawing primitives are generally defined as commands that draw predefined geometric shapes. (See

Figure 8 Professional Graphics Controller primitives

COMMAND CODE		DESCRIPTION
LONG FORM	SHORT FORM	
ARC	AR	Draw arc
AREA	A	Fill area
AREABC	AB	Fill to specified color
AREAPT	AP	Set fill pattern
CA	CA	Set ASCII Mode
CIRCLE	CI	Draw circle
CLBEG	CB	Begin command list
CLDEL	CD	Delete command list
CLEARS	CLS	Clear screen
CLEND	CE	End command list
CLIPH	CH	Set hither clip mode
CLIPY	CY	Set yon clip mode
CLOOP	CL	Run command list (multiple times)
CLRD	CRD	Read command list
CLRUN	CR	Run command list
COLOR	C	Set current color
CONVRT	CV	Convert coordinates
CX	CX	Set HEX mode
DISPLA	DI	Set display mode
DISTAN	DS	Set viewing distance
DISTH	DH	Set CLIPH distance
DISTY	DY	Set CLIPY distance
DRAW	D	Draw vector
DRAW3	D3	Draw vector in 3D
DRAWR	DR	Draw relative
DRAWR3	DR3	Draw relative 3D
ELIPSE	EL	Draw ellipse
FILMSK	FM	Set fill mask
FLAGRD	FRD	Read status flag
FLOOD	F	Flood viewport
IMAGER	IR	Read image line
IMAGEW	IW	Write image line
LINFUN	LF	Set draw function
LINPAT	LP	Set draw pattern
LUT	L	Set LUT entry
LUTINT	LI	Initialize LUT
LUTRD	LRD	Read LUT entry
LUTSAV	LS	Save LUT setting
MASK	MK	Set draw mask
MATXRD	MRD	Read model matrix

COMMAND CODE		DESCRIPTION
LONG FORM	SHORT FORM	
MDIDEN	MDI	Initialize model matrix
MDMATX	MDM	Set model matrix
MDORG	MDO	Set model origin
MDROTX	MDX	Rotate model (X)
MDROTY	MDY	Rotate model (Y)
MDROTZ	MDZ	Rotate model (Z)
MDSCAL	MDS	Scale model
MDTRAN	MDT	Translate model
MOVE	M	Move drawing point
MOVE3	M3	Move 3D drawing point
MOVER	MR	Move drawing point (relative)
MOVER3	MR3	Move 3D (relative)
POINT	PT	Draw point
POINT3	PT3	Draw 3D point
POLY	P	Draw polygon
POLY3	P3	Draw 3D polygon
POLYR	PR	Draw polygon (relative coordinates)
POLYR3	PR3	Draw 3D polygon (relative coordinates)
PRMFIL	PF	Set fill mode
PROJCT	PRO	Set projecting angle
RECT	R	Draw rectangle
RECTR	RR	Draw relative rectangle
RESETF	RF	Initialize status
SECTOR	S	Draw sector
TANGLE	TA	Set text angle
TDEFIN	TD	Define text character
TEXT	T	Draw text string
TEXTP	TP	Use programmed characters
TJUST	TJ	Set text justify
TSIZE	TS	Set text size
VWIDEN	VWI	Initialize viewing matrix
VWMATX	VWM	Set viewing matrix
VWPORT	VWP	Set viewport parameters
VWROTX	VWX	Rotate view (X)
VWROTY	VWY	Rotate view (Y)
VWROTZ	VWZ	Rotate view (Z)
VWRPT	VWR	Set view reference point
WAIT	W	Wait N frames
WINDOW	WI	Set window parameters

Figure 8.) The user specifies size and position via the parameters associated with each individual command. Also included in this category are move and flood commands. The richness of the set of drawing primitives illustrates the great power of the Professional Graphics Controller.

Stored images may be written to or read from the screen by specifying a line number and a beginning and ending point within that line. High data rates may be achieved by sending and receiving data in run-length encoded format.

The lowest level of transformation occurs following the two-dimensional move or draw commands. The WINDOW and VIEWPORT commands define an area in the virtual coordinate space and an area within the monitor screen, respectively.

Three-dimensional transformations are used to convert points from 3D to 2D. The process involves converting 3D world coordinates to 3D viewing coordinates through matrix manipulation, using a modeling matrix, a view reference point matrix, and a viewing matrix. The 3D viewing matrix must then be converted to 2D virtual coordinates by using the distance command, projection angle, and/or the CONVRT command.

A series of text commands is a versatile tool for positioning text on the screen. Text is positioned by moving the 2D current point. The angle of the text may be set, as well as the justification about the current point. Several text sizes are available, with the default being text size eight, which writes a 7 × 9-pixel character in an 8 × 12-pixel block. In addition to the standard character set provided by the Professional Graphics Controller, the user may program his own character font, depending on the requirements of the particular application.

The color look-up table (LUT) commands allow the user to perform several functions: load colors into any individual LUT location; read any individual location; load a predefined set of colors; or save a programmed set of colors. There are six predefined sets of colors for the loading of the LUT.

As previously discussed, a number of flags are available in the communications area to set and test various conditions of the Professional Graphics Controller. In addition to those previously discussed, there are several flags that are set by the microcode and may be read with the readback command FLAGRD. Included in these are current color, area pattern, mask, viewport, and window. In all, there are 25 flags that may be read or set by the microcode.

The readback commands allow the user to read various parameters from the Professional Graphics Controller. In addition to the FLAGRD command previously mentioned, any look-up table entry may be read, and the viewing matrix and modeling matrices may be also read.

The emulator faithfully reproduces both the 40-column and 80-column text modes.

The Professional Graphics Controller provides a command list capability to allow a series of high-resolution commands to be executed by a single command. A command list may be run once or a number of times. Any command may appear in a command list except CLBEG (**Command List Begin**). Other commands available include a WAIT command and a communications mode command to specify the ASCII (character) or the HEX (binary) communication mode. More information on the individual commands may be found in Reference 1.

Color/Graphics Adapter emulator

Since we wished to offer the user of the IBM Professional Graphics Display and Controller the ability to run existing programs designed for the IBM Personal Computer, it was necessary to emulate the Color/Graphics Adapter in all its modes as completely as possible. The emulator, therefore, faithfully reproduces both the 40-column and 80-column text modes with characters in all 16 colors, with all possible background colors, and with the appropriate blink functions and cursor styles. It also reproduces both the 320 pixels per line × 200 lines and 640 pixels per line × 200 lines graphics modes with the appropriate color palettes. The Color/Graphics

Adapter has a third color palette in the 320 × 200 graphics mode that is not documented. This too was emulated.

The standard IBM Color/Graphics Display has only 200 visible scan lines, and the Professional Graphics Display has 480 scan lines. It was therefore decided to use two scan lines to represent one line of the display being emulated. This worked well except that the picture looked a little squashed in the vertical direction, i.e., the aspect ratio was incorrect. To overcome this image distortion, a signal is sent to the display when it is in emulation mode to cause the height of the picture to be increased so that 400 lines exactly fill the display area. This restores the aspect ratio and removes the distortion.

Since we now have twice as many lines as the original display, we can improve the text characters.

The emulator must provide the same interface that the IBM Color/Graphics Adapter card does with the Motorola MC6845 video chip. This chip provides 20 registers accessed through two I/O ports. These registers are duplicated on the Professional Graphics Controller through the same two I/O locations. The Professional Graphics Controller communications RAM contains these registers in locations C63E0–C63F4 hexadecimal. The same HOLD/HOLDA sequence described previously is observed for this communication. The values stored in these locations control the different modes of operation that are supported.

The emulator provides 16K bytes of dynamic RAM memory organized as four 64K-bit chips, each of which is addressed as 16 000 four-bit nibbles. This is a totally separate memory from that supplied by the high-resolution portion of the Professional Graphics Controller. The separate memories allow both screens to be present and allow switching between them. The emulator memory also allows each screen to operate by itself, with each screen utilizing 8000 four-bit nibbles of the memory.

The two graphics modes of operation of the emulator work in almost exactly the same manner, except for the programming of the color look-up table. Look-up table locations 248 through 255 are used by the emulator and programmed for the 8088, according to the mode of operation selected. These look-up table locations are loaded with black and a foreground color for the 640 × 200 mode, and they are loaded with the background color and the three-color palette for the 320 × 200 mode of operation. The emulator generates its own timing, and it scans the memory to access the look-up table in the same general manner that the Professional Graphics Controller high-resolution function does.

Since we now have twice as many lines as the original display, we can take advantage of this to improve the text characters. The original Color/Graphics Adapter text characters were built around a box measuring 8 pixels × 8 pixels. We were able to build our characters around a box measuring 8 pixels × 16 pixels, and thus we were able to use the same character set that was developed for the IBM Enhanced Graphics Adapter. This represents a considerable improvement over the original character set.

The emulator has been successfully tested with a multitude of existing programs including such exotic graphic programs as the Microsoft *Flight Simulator* and the IBM *Exploring the PC* program.

Concluding remarks

There are several ways in which a host computer—in this case, the IBM PC—could interact with the high-function section of the Personal Graphics Controller. One way is to follow the lead of the IBM Color/Graphics Adapter and allow the PC to write directly into and read from the graphics display refresh memory. An alternative approach—the one adopted—is to interact with the controller through a well-defined command interface. Another approach considered allows the host to download microcode to the graphics microprocessor and thereby modify the command set that is executable by the controller. This approach was rejected because it might jeopardize future program compatibility. It was felt that the command interface was sufficiently flexible to be worthy of reimplementation—with appropriate modifications—in other embodiments.

The IBM Professional Graphics Controller and Display provide a low-cost, high-function graphics capability for the engineering and scientific commu-

nity. They allow objects to be constructed from lines and polygons and to be manipulated in three dimensions. They allow images of photographic quality to be displayed, and they permit the development of two-dimensional charts of considerable complexity. We have produced a hardware graphics facility that has a rich command set. We look forward to the development of the application software needed to breathe life into the Professional Graphics Controller and to its subsequent use by scientists and engineers.

Cited reference

1. *IBM Personal Computer Professional Graphics Controller, Technical Reference Manual*, IBM Corporation; available through IBM branch offices.

General reference

J. D. Foley and A. van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing Company, Reading, MA (1982).

Reprint Order No. G321-5234.

Keith A. Duke *IBM Entry Systems Division, 1000 NW 51st Street, Boca Raton, Florida 33432.* Mr. Duke was born and educated in England. He received a B.Sc. degree in physics from London University in 1952. He joined IBM in 1960 at the IBM United Kingdom Limited Laboratories, Hursley, England, and came to the United States in 1963. Mr. Duke has been involved in a design capacity in several hardware projects, including the System/360 Model 40 and the Professional Graphics Controller for the IBM Personal Computer. He has also worked on a number of design automation projects.

W. Alan Wall *IBM Entry Systems Division, 1000 NW 51st Street, Boca Raton, Florida 33432.* Mr. Wall joined IBM in 1979 in the Federal Systems Division at Cape Canaveral, Florida. He worked on the Space Shuttle Launch Processing System and Spacelab Integration Project Engineering. In early 1984, he transferred to Engineering/Scientific Development Engineering in the Entry Systems Division. Since then, Mr. Wall has worked on the Professional Graphics Controller in an engineering capacity. He graduated from Louisiana Tech University in 1979 with a B.S. degree in electrical engineering.