HP-2680A: THE MYSTICAL PRINTER

Richard Oxford
MCI Digital Information Services Corporation,
Washington D.C., USA

SUMMARY

This paper aims to more intimately acquaint the HP user with laser printers. This acquaintance is brought about through:

1. an explanation of technically what goes on in the box,
2. a description of the capabilities that are available to the user, and how to exploit them
3. and, an explanation of the data structures used in graphic data.

INTRODUCTION

Graphics are becoming an increasingly important part of computer information systems. Computer generated graphics range from simple bar charts, to multiple color presentation graphics, to very sophisticated solid modeling. With the advances in technology, and decreasing prices of hardware, graphic capable peripherals are becoming more affordable by the general public. They are no longer limited to large businesses with multi-million dollar DP budgets.

The decrease in price of laser printers, in particular, have made high speed printing and graphics more accessible. The need to generate graphics for the laser printer is increasing, but there seems to be only a small amount of effort being directed in this area. Here is a device capable of generating forms, logos, and signatures, as well as graphics, but yet it is quite frequently used only for high speed printing and forms generation. These printers have a lot of capability going unused, except for some packages developed by HP (TDP, HPDRAW, etc.). This "lack of use" can be attributed mostly to a "lack of knowledge" of how the laser printer actually works, and how to use it.

MCI DISC has been using many features of the 2680A laser printer for over 2 1/2 years. Many different logos, graphics, and signatures are printed during a typical job. Signatures and graphics are programmatically positioned on a page. All of this is done independent of an environment file! Most users are unaware of the ability to programmatically place graphic data, as well as text data, anywhere on the printable page. All functions performed by TDP are available to any programmer through intrinsic calls. One only needs to be familiar with the basic operation of the laser printer, as well as some additional data formats, to be able to take full advantage of the printer's capabilities.

Since we, at MCI DISC, are currently using 2680A laser printers in our applications, most information given will be in reference to this printer, but the same basic principles apply to other HP laser printers. Since the most popular print format is 8 1/2 x 11 inches, this will be used in the examples.

Basic Laser Operation

The old saying "It's all done with mirrors" really applies to the 2680A laser printer. Although it is not necessary to know how all of the mirrors function, it is beneficial to have a basic understanding of how the laser

printer works. It will then be easier to understand data structures referring to raster scan lines and dot-bit images.

The generation of a printed image by the laser printer is a rather complex procedure. The description given here is a very simplified explanation of how the image is produced, slightly modified to be more easily understood. The generation of an image in the laser printer is accomplished through a process known as electrophotography. An image is produced on paper in three basic steps:

1. Generation of the image on the drum.

2. Transfer of the image to the paper.

3. Fixing of the image on the paper.

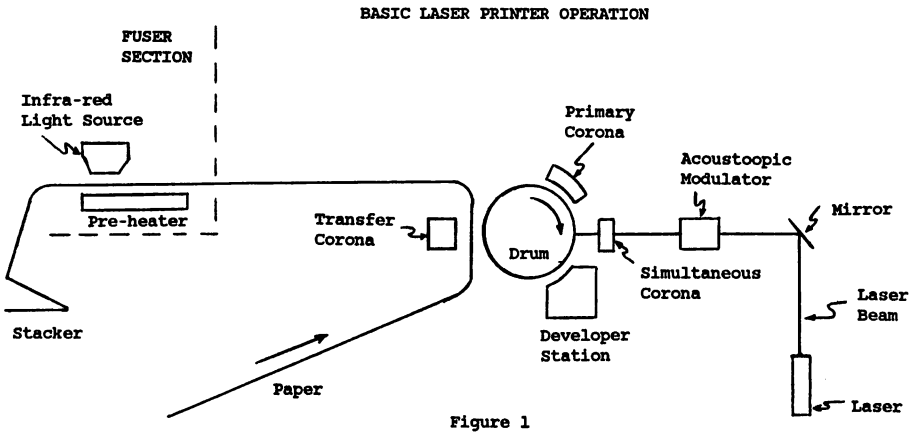


Figure 1

Figure 1 shows the basic layout of the 2680A laser printer.

Image Generation

The physical page on the 2680A is 8 1/2 x 11 inches and consists of a dot matrix of 2048 dot positions along the 11-inch axis and 1536 dot positions along the shorter axis, although the printable area is slightly smaller. The laser printer can handle a physical page size of 17 x 11 inches, but the 8 1/2 x 11 inch size is used for discussion. The laser beam scans from left to right, along the 11 inch axis, as the drum turns. Each page therefore, is composed of 1536 scan lines of 2048 dots each. A special device, called an acoustoopic modulator, receives data from the print control electronics and allows the laser beam to strike the drum when a dot is to be printed. The effect of the beam striking the drum, in conjunction with drum charging devices called coronas, results in a negative charge on the drum for every dot to be printed.

As the drum continues to turn, it passes by the developing station. By means of a "magnetic brush", toner is applied to the drum. The developer mixture used consists of iron filings called carrier, and tiny plastic balls referred to as toner. The toner is statically charged and attracted to the negative charge locations on the drum. The iron filings are used by the "magnetic brush" to help move the toner to the drum. When the drum has moved past the development station, it contains the final image to be printed.

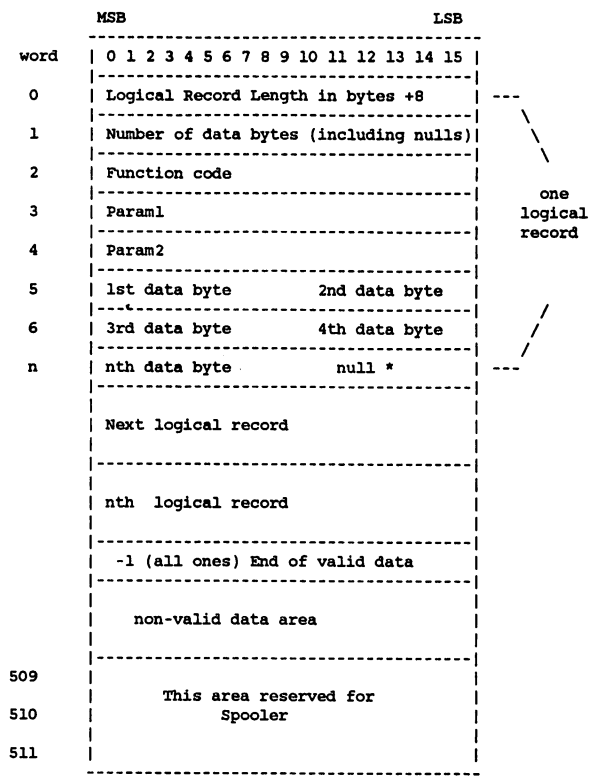
Image transfer

Once the image is on the drum, it must be transferred to the paper. As the drum turns, the paper moves at the same speed as the drum, and passes between the drum and the transfer corona. The transfer corona imparts a high negative charge to the paper, which pulls the toner off the drum and onto the paper. The final image is now present on the paper.

Fixing the Image

As in most photographic processes, the final image must be "fixed" to prevent it from changing. At this point, before fixing, the image is nothing more than tiny plastic balls which can easily be smeared or brushed off the page. To fix the image, the paper passes through the final step in the fuser section of the printer. The fuser section consists of a preheater which heats the paper, and an infra-red light source that 'melts' the toner. The end result is that the toner is actually melted, or fused, into the paper.

SPOOLER RECORD LAYOUT



* all logical records must end on a word boundary. If they do not, a null is inserted, and included in the record length

Figure 2

COMMUNICATING WITH THE LASER

The next step in understanding how the laser printer works is to know how to get the image information from the HP-3000 to the laser printer. In order to maximize the throughput of the HPIB bus when communicating to the

laser, data is sent to the laser in blocks of 512 words. The basic layout of these blocks is shown in figure 2. Each block can contain more than one logical record, with each logical record being a single function for the laser. Note that each logical record contains 2 byte counts, a function code, and 2 parameters, as well as any data being sent. The last 3 words in the block are used by the spooler. Although the block size for data transfer is 512 words, actual blocking is handled by the spooler and a buffer of any reasonable size can be written.

Most users have no problem sending files to the laser, or using TDP to communicate to the laser, but are limited to the constraints of those packages. Those who have used the LPS interpreter, or have written programs using 'P'intrinsics (IFS/3000 Programmatic intrinsics), have a better understanding of laser operation. If we look one step deeper, we find that the basic method used to communicate to the laser printer is via a single callable intrinsic, FDEVICECONTROL. This is a little known, even less understood intrinsic. This intrinsic gives us the ability to change character fonts, change logical pages, position text, download pictures or graphics, and much more. Table 1 gives a summary of the control codes used with the FDEVICECONTROL intrinsic and definition of each. (All of the control codes, with the exception of 139 and 144, have the parm1 and parm2 values shown in the MPE Intrinsics reference manual. The parameter definition for control codes 139 and 144 are given at the bottom of table 1.) Several of the control codes are used to set up or alter the current printing environment, load character sets and forms, or load Vertical Forms Control data. There are also codes for setting the physical page, loading logical page tables, and controlling the job. Since most of these functions are handled by the spooler when using an environment file, these functions will not be discussed here, but it is possible to have total control over the print environment even if no environment was specified for the job.

One very important concept is that of logical pages. The physical page, typically 8 1/2 x 11, can be divided into as many as 32 logical pages. A logical page is nothing more than a portion of, or a window on, the physical page. Each logical page carries its own set of specifications. They can be oriented in different directions. They can each have a different form associated with them. They can even have different character sets as their base set. Logical pages can also overlap. By simply having 4 logical pages all the same size as the physical page, but all with different orientations, you have the ability to write text on the physical page in any of the 4 possible orientations.

Now that all of the basic information has been given, let's discuss some of the function codes.

Contrary to popular belief, the laser can only have 2 character sets available at any given time. Everything else is done with slight of software. The character sets for use are selected with a control code of 128. Param1 and param2 bits 8-15, are used to define the primary and secondary character sets. Param1 holds the ID for the primary character set, while param2 holds the ID for the secondary character set. The IDs for the character sets are the font numbers as described in the environment file. The SO (%16) and SI (%17) ASCII control codes are used to switch to and from the secondary character set. (An important note: If you were using the secondary character set before execution of a control code 128, you will still be using the secondary character set after the execution, even if the secondary character set was changed.)

A control code of 129 allows the activation and deactivation of logical pages. Param2 is divided in half. The upper byte holds the logical page number to be deactivated, while the lower byte holds the logical page number to be activated. Param1 bit 0 indicates deactivate the logical page specified in the upper byte of param2, while bit 1 indicates activate the logical page specified in the lower byte of param2. Therefore it is possible to activate a logical page and deactivate a logical page at the

FDEVICECONTROL control code summary

Control Code (in decimal)	Definition
1	Print Data
2	File Control
3	File Open
4	File Close
128	Primary,Secondary Font Selection
129	Select/De-select Logical Pages
130	Move Pen Relative
131	Move Pen Absolute
132	Define Job Characteristics
133	Define Physical Page
134	Download/Delete Character Set
135	Download/Delete Form
136	Download Logical Page Table
137	Download Multi-copy Form Overlay Table
138	Download/Delete VFC
139	Download/Delete Picture
140	Page Control
141	Clear Portions of the Environment
142	Job Open
143	Load Default Environment
144	Print a Picture

Controlcode = 139		Download/Delete Picture
	param1 (0:1)	0 - Load a Picture 1 - Delete a Picture
	param2 (0:1)	0 - First record of a load 1 - Continuation record of a load
	(8:8)	Picture identifier (0-31)
Controlcode = 144		Picture Print
	param1 (0:1)	0 - Temporary Picture 1 - Addressable Permanent Picture
	(1:1)	0 - Co-ordinates X and Y are relative to current pen position 1 - Co-ordinates X and Y are absolute pen positions on the logical page
	param2 (0:1)	0 - First record of a temporary picture load 1 - Continuation record of a temporary picture load
	(8:8)	Picture identifier (0-31)

Table 1

same time with the same command. Further, all 32 logical pages can be active simultaneously. One logical page must be active at all times. Switching between active logical pages will be described later.

The command more frequently used to move between logical pages, without having to worry which ones are active, is the control code of 140. This is the page control function. It allows for a physical page eject and/or a change of logical pages. If paraml is set to a 1, a physical page eject occurs. If it is set to a 0, no page eject occurs. Bits 8-15 of param2 are used to specify the next logical page to be used. It can be the current logical page, or it can be a new logical page. Whichever logical page is specified in this parameter becomes the active logical page upon completion of the intrinsic call.

There are two pen movement commands. These are important for the positioning of data and graphics on a page. For those that are familiar

with plotters, the first thing you look for is a pen up/down command. Well, there is none for the laser. The laser printer is not a plotter, and the pen command is only used to position information on the page. A control code of 130 moves the pen relative to its current position, while a control code of 131 moves the pen absolute with reference to the upper left hand corner of the logical page. The laser printer always views the logical page in its readable direction, so rotating the logical page moves its upper left hand corner (See Figure 3). The logical page is referenced in a logical page co-ordinate system with the upper left hand corner being 0,0. The X axis runs across the top of the logical page, increasing in value as you move to the right of the upper left corner, or to the right of the current pen position in a relative move. The Y axis runs down the side of the logical page. Its value increases as you move down from the upper left corner, or down from the current pen position in a relative move. The co-ordinates actually represent dots, so an X co-ordinate of 12 is actually the 13th dot along the X axis. It is the 13th dot because the co-ordinate system starts at 0. The same applies to the Y axis.

LOGICAL PAGE ORIENTATION

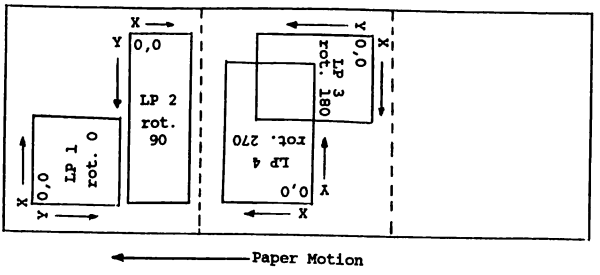


Figure 3

In the pen commands, paraml is the X axis co-ordinate or offset, while param2 is the Y co-ordinate or offset. These co-ordinates are expressed in 16 bit signed radixed integers. The radix point falls between bit 13 and 14 of the word. Since we cannot move the pen in increments of less than a single dot, bits 14 and 15 will always be 0. To try and cut through the fog, if you wanted to move one inch, at 180 dots per inch, the parameter would be set to 180 x 4 or 720.

Now we can see how to select a logical page, select a character set, and position the pen to any point on the logical page. To print text, you simply use the FWRITE intrinsic. The text will be printed on the page starting at the current pen position. The current pen position is the lower left hand corner of the first character. Typically, a file is opened to the laser with carriage control. This allows the use of the control parameter in the FWRITE intrinsic. If mostly text positioning is being done, a code of %320 is used. If only text is being printed, any of the carriage control codes defined with the FWRITE can be used to control text positioning.

When sending a page eject to the laser printer, using the FWRITE intrinsic with a control code of %61, it actually performs a logical page eject. Whether a physical page eject also occurs is dependent upon what logical page you are currently on, and how many logical pages are active. We have seen that you can activate and deactivate logical pages on command. We also know that all 32 logical pages, if defined, could be active at the same time. Logical page information is held in the laser printer in a Logical Page Table (LPT). The table has 32 entries, one for each possible logical page. When a page eject is received by the laser, it starts scanning the LPT from the current entry, looking for an active logical page. If it gets to the end of the LPT without finding one, it performs a physical page eject, and starts scanning the LPT entries from the beginning. When it finds an active LPT entry, it sets the current logical

page to that entry and positions the pen to the upper left hand corner of the logical page. Of course, using the FDEVICECONTROL intrinsic with a control code of 140 insures that a physical page eject will occur.

Another method of printing data is to use the FDEVICECONTROL intrinsic with a control code of 1. This code works much the same as the standard FWRITE. When using this control code, paraml, bits 8-15 are used to determine the vertical format for printing. The codes used here are the same as %0 - %377 for a standard FWRITE with the following exceptions:

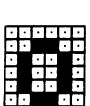
Code	Meaning
%1	Use the first data byte of the record for VFC
%61	Conditional logical page eject
%62	Physical page eject
%63	Unconditional logical page eject

Param2 bit 14 is used for Autoeject mode. A 0 indicates set Autoeject, while a 1 indicates reset Autoeject. When Autoeject is set, if carriage control causes the pen to move off the logical page, a logical page eject is automatically done and the text is printed on the next logical page. If Autoeject is not active, no text is printed and an error is written to indicate that the pen movement was off the logical page. Bit 15 is used to determine what the spacing mode will be. A 0 indicates postspacing while a 1 indicates prespacing. Prespacing causes the carriage return line feed to occur before the printing of the text, while postspacing causes it to occur after the text is printed.

The final topic to be covered is the formatting, downloading and printing of graphic data. In order to work with graphics on the 2680A, you must understand the formats of dot-bit memory words and triplets.

DOT-BIT Memory images

A dot-bit memory image is just what the name implies, a bit map of the image to be printed. Each bit of the dot-bit image represents one dot of the actual figure to be represented. Figure 4 shows an image that is 6 dots by 7 dots. In a dot-bit image, a 1 represents no dot, and a 0 represents a black dot. This means that the dot-bit image of the graphic in figure 4 would consist of 42 bits. Since data is stored in the HP-3000 in 16 bit words, this image is packed into 16 bits per word, with any remaining bits being a 1. Looking at the dot-bit map in figure 4, this can be seen. Word 0 bits 0-5 contain the dots for row 1 of the image. Bits 6-11 contain the dots for row 2 of the image. Word 0 bits 12-15 are the first 4 dots for row 3, while Word 1 bits 0-1 contain the last 2 dots for row 3. The dot image continues with Word 2 bits 4-9 being the dots for the last row of the image. Bits 10-15 of Word 2 are set to one to fill out the word. These bits will not be used by the laser printer.



DOT-BIT MAPPING

word	Dot-bit Image	
0	1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1	%017747
1	0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1	%066667
2	0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1	%037777

Figure 4

This method of image representation is referred to as a "Bit Raster", or simply a Raster, image. Since the laser resolution is 180 dots per inch, an 8 1/2 by 11 image would require a bit map of (8.5*180) * (11*180) or approximately 4 million dots. That would require 1/2 megabyte of memory for storage of a single graphic. Since the 2680A uses memory to store all character fonts, logical pages, and other process information, 1/2

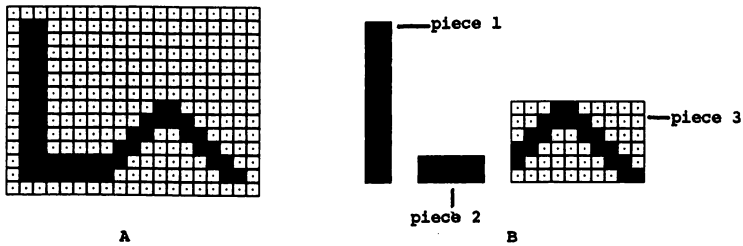
megabyte storage for a single graphic is unacceptable. To download 32 graphics would require 16 megabytes of memory for storage alone.

Consider an 8 1/2 by 11 page with a single line bordering all sides. If a dot-bit map were generated for this graphic, we would see that over 99% of the dot-bit map was describing white space where no image was to be printed! Now that is a real waste of memory. In order to overcome this deficiency, "Partitioned Raster" dot-bit maps were developed. In a Partitioned Raster Dot-bit map, only those dots required to define the image are used.

Partitioned Raster Dot-bit images

Figure 5A shows a graphic that is 19 dots wide by 14 dots high. In standard raster format, this would require 266 bits, or 17 words. Figure 5B shows the same image broken into 3 pieces that can be used to describe the entire image. Piece 1 requires 24 bits, piece 2 requires 10 bits, and piece 3 requires 60 bits. The "partitioned" dot-bit map in figure 5C shows word 0 bit 0 thru word 1 bit 7 hold the image for part one. Bits 8 thru 15 are padded with ones to fill out the word. Each part of the image is treated as a single image, and the dot-bit memory image must be represented in full words. Word 2 bits 0 thru 9 hold the image for part 2, and word 3 bit 0 thru word 6 bit 11 hold the image for part 3. The entire image now requires only 7 words for storage instead of 15. That is less than half that required for a standard "Raster" dot-bit image.

PARTITIONED DOT-BIT MAPPING



Printer Image would appear as ⌞

word	Dot-Bit Image	
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	%000000
1	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1	%000377
2	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1	%000077
3	1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0	%163760
4	1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0	%174634
5	1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 0	%171576
6	0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1	%077717

C

Triplet Words

	piece 1	piece 2	piece 3
0	%176414	%175002	%172406
1	%000000	%000002	%000003
2	%000020	%000060	%000200

D

Figure 5

Now we have a much smaller dot-bit image of our graphic, but we need a way to tell the laser printer how to put the "pieces" together on the page. This is done by using what is known as "triplets". A triplet is a

TRIPLET WORD LAYOUT

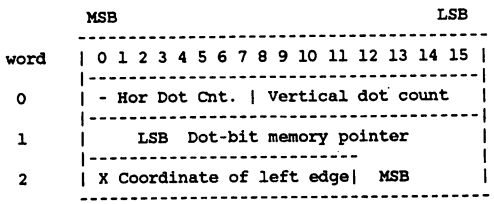


Figure 6

group of three words describing each "piece" of the graphic. Figure 6 shows the layout of a triplet. Word 0 bits 0-7 are used to indicate the number of dots there are horizontally in this piece of the graphic. This number is the 1's complement of the count. Word 0 bits 8-15 indicate the number of dots vertically in this piece of the image. We can now see that one restriction of a partitioned raster format is that each piece of the graphic cannot be any greater than 255 by 255 dots. Word 1 contains the 15 least significant bits of the memory pointer and word 2 bits 12-15 contain the 4 most significant bits of the memory pointer. The memory pointer is used to indicate which word, starting from 0, in the partitioned dot-bit map that the image for this piece of the graphic begins. This pointer is 20 bits long, because dot-bit maps for large graphics can exceed 65,000 words. This 20 bit pointer allows us to have a graphic that requires over a million words to represent. Word 1 bits 0-11 are used to indicate the position along the X axis of the entire graphic, starting from 0, that this piece of this image is to be placed.

Figure 5C shows the 3 triplets that would be used to describe the image shown in Figure 5B. Piece 1 has a horizontal count of -2 (one's complement), a vertical count of 10, the first bit describing this part of the image starts at word 0 in the bit map, and this piece of the image starts at dot 1 on the x axis of the graphic. Looking at the triplet for piece 3 of the image, its horizontal count is -10, the vertical count is 6, it starts at word 3 in the bit map, and its X coordinate is 8.

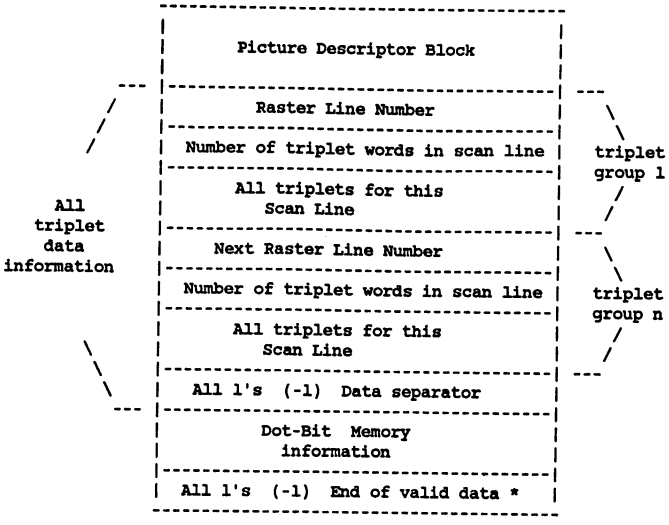
DOWNLOADING OF GRAPHIC DATA

Now that the pieces to the graphic are formatted, they need to be downloaded to the laser printer. An FDEVICECONTROL with a control code of 139 is used. This is the download/delete picture function. Paraml bit 0 is set to a 0 to delete a picture, or set to a 1 to download a picture. Param2 bits 8-15 are the picture identifier. This can be an integer number between 0 and 31, since the laser can hold 32 pictures. When downloading a picture, if a picture currently exists with the same id, it is deleted, and replaced with the one being loaded. Since picture data can span more than one record, it is necessary to be able to indicate continuation records for download. Param2 bit 0 is used to indicate if the current record is the first record of this picture, or if it is a continuation record for a picture in the process of being downloaded. Bit 0 is set to a 0 for the first record and set to a 1 for all subsequent records. All records for a picture must be downloaded without interruption. There cannot be any other commands sent to the laser printer until all of the picture has been downloaded.

Figure 7 shows the layout of picture download data. The data is divided into 3 functional parts:

1. Picture Descriptor Block
2. Triplet data information
3. Dot-bit map data

PICTURE LOAD RECORD LAYOUT



*added by the spooler

Figure 7

The first part encountered is the picture descriptor block. This data block is used to describe the characteristics of the picture being loaded. This will be covered in detail later. The next part contains all of the triplets, in groups, that are needed to describe all pieces of the picture. Triplets are grouped together according to their starting location in the picture. The first word in a triplet group represents the raster scan line where the following triplets will begin. Recall, that when the picture was broken into pieces, each piece had a triplet describing it. The triplet described the size of the picture piece, horizontally and vertically, where the data for that piece could be found in the dot-bit map, and the X co-ordinate of the left edge of that piece in the entire picture. The only thing missing that would be needed for proper placement of the piece in the entire picture was the Y co-ordinate. The Y co-ordinate is referred to by raster scan line number, so the raster scan line number for a particular triplet would be the Y co-ordinate, starting from 0, of the top line of this piece in the picture. The next word in the triplet group indicates how many triplet words there are for this scan line. If there are 2 triplets for picture pieces starting on this scan line, then this number will be 6 (3 words per triplet, 2 triplets). Following this are all of the triplets that start on this scan line. Due to limitations of the laser printer, the maximum number of triplets, or pieces of the picture, that can start on a single scan line is 255.

Following this triplet group would be another triplet group for the next raster scan line. The triplet groups are loaded in order by ascending Y co-ordinate (raster scan line number), starting from the top of the picture. Since the Y co-ordinate starts at 0 at the top of the picture, and goes thru some Y co-ordinate N, the triplet group for raster scan line 0 will be the first group and the group for raster scan line N will be the last group. If there are no pieces of the picture that start on a given raster scan line, then there is no triplet group required for that scan line. After the triplet data information there is a -1 (all ones) data separator. This is used to signify the end of the picture description, and the beginning of the dot-bit memory map for the picture. Following this data separator, is the dot-bit memory map for the entire picture.

PICTURE DESCRIPTOR BLOCK LAYOUT

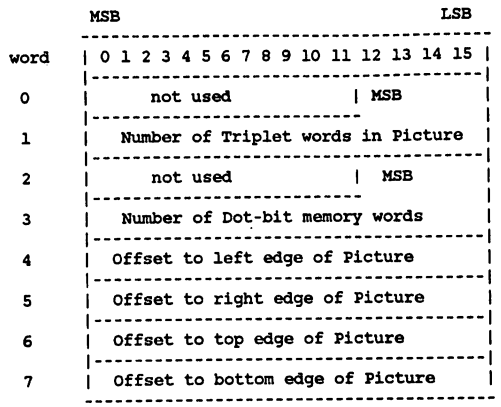


Figure 8

The picture descriptor block is 8 words long and is used to describe overall information about the picture being downloaded. Figure 8 shows the layout of the picture descriptor block. Word 0 bit 12 thru word 1 bit 15 is a 20 bit integer which indicate the total number of triplet data information words in the download record. This includes all triplets, all scan line numbers, and also the data separator as shown in Figure 7. Word 2 bit 12 thru word 3 bit 15 is a 20 bit integer indicating the number of dot-bit memory map word used for this entire picture. Words 4 thru 7 are used to describe the location of the picture origin. The picture origin is used to determine where to place the picture on a page. When a picture is printed on the laser printer, it is positioned with the origin at the current pen position. In most cases, the origin is placed in the upper left hand corner of the picture. This allows the corner of the picture to fall at the current pen position.

Figure 9 shows the data buffer that would be used to download the graphic shown in figure 5A. This image would be downloaded using the FDEVICECONTROL command with a control code of 139, paraml set to a 0, and param2 set to a 2. Buffer length for the download is 29 words. At the completion of the command, this picture would be stored as picture 2 in the laser printer.

To print the picture, an FDEVICECONTROL command with a control code of 144 is used. Paraml bit 0 is used set to a 1 to indicate an addressable picture, or set to a 0 to indicate a temporary picture. An addressable picture is one which is downloaded with a control code of 139. A temporary picture is one which is downloaded with this command, is printed, and then deleted from memory. Temporary pictures will be covered later. Paraml bit 1 is set to a 0 to indicate that the picture is to be printed with its origin at the current pen position. It is set to a 1 to indicate that the origin is relative to position 0,0 on the logical page. Param2 bit 0 is a 0 for permanent pictures, and bits 8-15 are used to identify the picture. If we wanted to print the picture just loaded at the current pen position, then parameters 1 and 2 would be %100000 and %000002 respectively.

TEMPORARY PICTURES

When a picture is to be printed only once, it is better to download the image as a temporary picture. This prevents the memory in the laser printer from being used to hold images that are printed only once. It also

PICTURE DOWNLOAD RECORD for
PICTURE in FIGURE 5A
(Permanent Addressable)

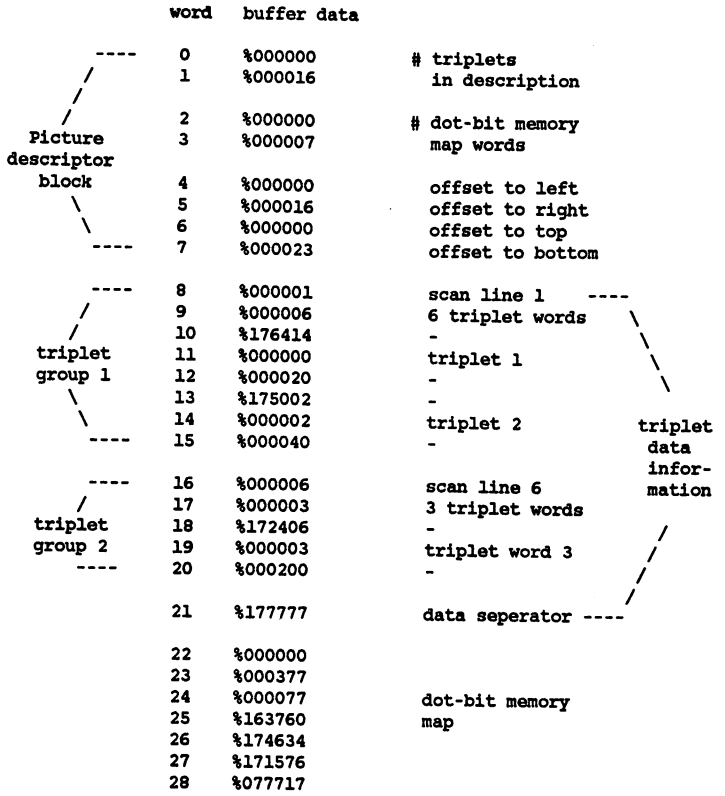


Figure 9

keeps the laser printer memory free to hold 31 permanent images. (One of the 32 image ids must remain free for the download of the temporary picture, but the picture is deleted after printing.) This means, that if many images are to be used only once, there is no limit to the number of pictures that can be printed during a logical job. A temporary picture is actually downloaded with the print picture command, control code 144. When downloading a temporary picture, the buffer is formatted the same as with a permanent picture except the picture descriptor block is preceded by 2 extra words, as shown in Figure 10. The first word is an X co-ordinate, and the second word is a Y co-ordinate for placement of the temporary picture. The co-ordinates are expressed in radixed integer format, the same as those used for pen movement. When downloading a temporary picture with a control code of 144, paraml bit 1 is used to indicate whether the X and Y co-ordinates preceding the picture are absolute pen positions on the logical page or relative to the current pen position. A 0 indicates they are relative, while a 1 indicates that they are absolute pen positions. As with the download picture command, control code 139, download information can span more that one record. Param2 bit 0 is used to indicate the continuation of a picture download, just as it was with the control code of 139.

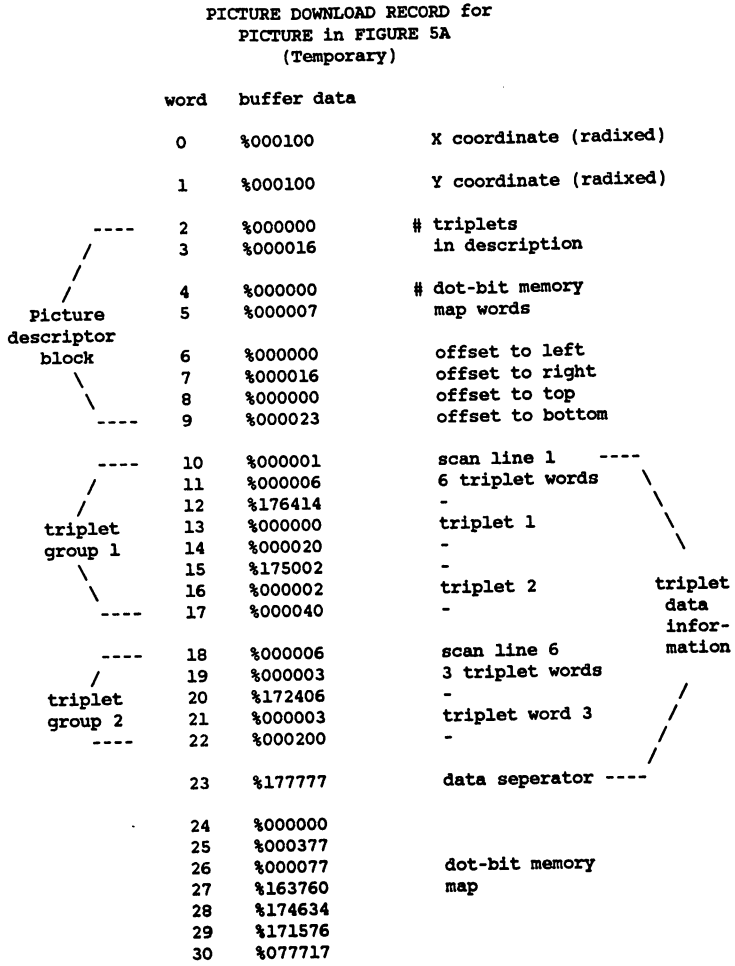


Figure 10

EXAMPLE

Figure 11 shows a sample program, written in pseudocode, to download the picture described in figure 5. It is downloaded using a pictue ID of 2, the pen is positioned and the picture is then printed. Next, a temporary copy of the same picture is printed. Notice that a picture ID is used to download the temporary picture. If a picture was currently using that picture ID, it is first deleted, and then the temporary picture is downloaded. Once the picture is downloaded, it is printed and deleted. It is important to note that the temporary picture is not truly deleted until a physical page eject occurs. Therefore, if more than one temporary picture is to be downloaded on a physical page, they must all have different picture IDs. Once a physical page eject occurs, the picture IDs of all the temporary pictures may be reused. Figure 12 shows the resulting output from the program. For clarity, the size of the image was increased. The text at the bottom of the second picture was printed using the FDEVICECONTROL 131 and FWRITE intrinsics.

EXAMPLE of PICTURE
DOWNLOAD and PRINT

The following is a pseudocode example of how to download and print the image shown in Figure 5A. All buffer offsets start at 1. This example shows both a temporary and permanent addressable picture load and print.

```
   Buffert = Logical 31 word buffer initially loaded with the
             temporary picture data in Figure 10

   Bufferp = Logical 29 word buffer equated to Buffert(3), which
             would make it the same as the buffer in Figure 9

   Textbuffer = a logical buffer used for text data

   Laser = "*Laser "

***** Open the spoolfile to the laser as ASCII,NEW,CCTL,WRITE ONLY
   Laserout = FOPEN(Laser,%404,%1)

***** Download the permanent picture with ID 2
   FDEVICECONTROL(Laserout,Bufferp,29,139,%0,%2)

***** If there was a continuation record, the buffer and count would be
***** changed and param2 would be set to %100002

***** Move the pen to an absolute location
   X = %002640      **** 2 inches (180 dots per inch) radixed
   Y = %001320      **** 1 inch   (180 dots per inch) radixed
   FDEVICECONTROL(Laserout,Notused,Notused,131,X,Y)

***** Print the permanent picture at the current pen position

   FDEVICECONTROL(Laserout,Notused,Notused,144,%100000,%2)
***** The picture is now printed with the origin at X = 2 inches
***** and Y = 1 inch.

***** Now print the picture as a temporary picture using the
***** X and Y in the download file to position the orgin at
***** absolute  X = 4 inches and Y = 4 inches
   Buffert(1)=%005500
   Buffert(2)=%005500
   FDEVICECONTROL(Laserout,Buffert,31,144,%040000,%3)
***** The laser will temporarily use Picture ID 3 to download
***** and print the picture. Once the picture is printed, it
***** is deleted.

***** Move the pen under the picture and print text

   Move "This is the figure title" to Textbuffer
   FDEVICECONTROL(Laserout,Notused,Notused,131,%5500,%5764)

***** The pen is now 1 inch under the origin of the tempory picture

   FWRITE(Laserout,textbuffer,-24,%320)

   FCLOSE(Laserout,1,0)

   END of pseudocode
```

Figure 11

CONDENSING DOT-BIT-MAPS

Since triplets are used to point to a location in the dot-bit map where a portion of a picture begins, if there are several parts of the picture with identical data, the same locations in the dot-bit map can be used. A very good example is shown in Figure 13. All pieces of the graphic can be described as an all black image. If we describe the largest black area in the image, in this case piece 2, we see that it takes 32 bits to describe. All of the other pieces of the image are smaller, and can be described using the same dot-bit map area. All of the triplets in figure 13 point to word 0 for the dot-bit map data. The only difference is the horizontal and vertical counts, as well as the X coordinate.

Example Program Results

Please note that the picture size has been increased for clarity

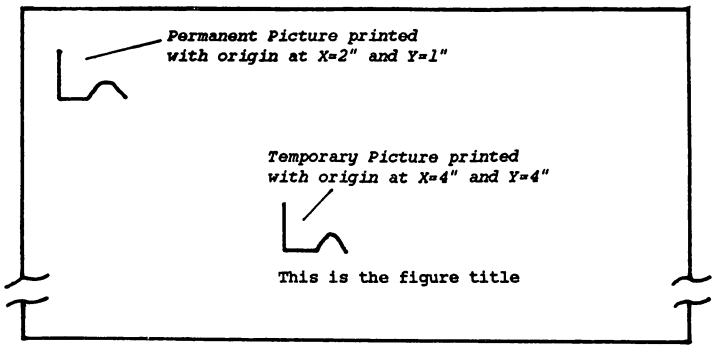
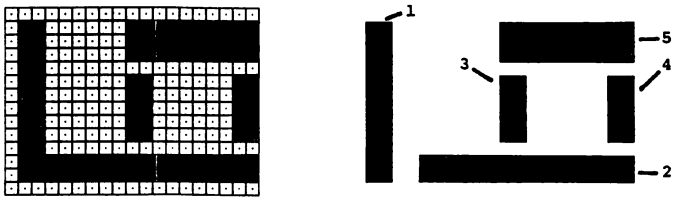


Figure 12

CONSERVING DOT-BIT MAP SPACE



word	Dot-Bit Image	
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	%000000
1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	%000000

Triplet Words

	1	2	3	4	5
0	%176414	%175002	%176405	%176405	%172403
1	%000000	%000000	%000000	%000000	%000000
2	%000020	%000060	%000220	%000420	%000220

Figure 13

BIOGRAPHY

Richard Oxford has been involved with computer hardware and software for the past eleven years. His experience includes teaching computer hardware, developing software, and maintenance of hardware and software on various computer systems. He has worked with HP computers for the past seven years, and with the 2680A laser for four years. Presently he is with MCI Digital Information Services as system manager for their network of HP-3000s and laser printers.