REPLACED SHEETS SG
☑ Description : paged 1 ..... to paged 103
☑ Claim(s) : paged 104 to paged 117
☑ Drawing(s) : ~~paged~~ 1 ..... to ~~paged~~ 21
☑ Abstract : paged 118 to paged 118

9604795-4
IPOS

METHOD AND APPARATUS FOR PRINTING ACCORDING TO A GRAPHIC LANGUAGE

BACKGROUND OF THE INVENTION

Field Of The Invention

The present invention relates to a method and apparatus for printing an image in accordance with a graphics language stream.

Description Of The Related Art

Recently, printing apparatus such as laser beam printers which form images by selectively printing pixels on a page have become exceedingly popular because of their flexibility, printing speed and low cost. Such printing apparatuses print images by scanning across a page while selectively printing individual pixels on the scan line. The page is advanced (in printers like laser beam printers the page is advanced continuously while printing) and additional scan lines are formed. The collection of printed pixels from all the scan lines on the page forms the desired image.

Whether pixels are printed on the page, and where those pixels are printed during a scanning operation, are determined by a printer controller which operates in accordance with a stored raster image of the print image. A raster is a computer representation of each of the pixels in the print image, and is stored in bit map

This Sheet Has Been Replaced

memory in the controller. In a bit map memory, there is a one-for-one correspondence between pixels on a printed page and memory locations in the bit map memory.

All printing apparatuses must operate within the constraints of the available resources. For example, the controller, which is typically a microprocessor forming a central processing unit ("CPU"), has only a limited amount of processing power and is able to manipulate only a limited amount of data including image data within any one time period. The controller has access to only a limited amount of memory including the bit map memory. Some printers are provided with other resources such as external memory devices which are accessible through a communication line which has only a limited amount of bandwidth (or data-carrying speed).

Conventional printing apparatuses use the finite amount of resources that are available to them only inefficiently. For example, Fig. 11 is an illustration of a typical CPU resource usage pattern in a printer. As shown in Fig. 11, CPU usage varies widely between peak CPU usage and relatively low CPU usage. Peak CPU usage occurs between printed pages during creation of the raster image in bit map memory while low CPU usage occurs as a page is being printed.

The reason for this inefficiency is that, that the image is printed properly, printing apparatuses such as

This Sheet Has Been Replaced

laser beam printers which form pixels by scanning must have a full raster image of the print image developed in bit map memory before printing can commence. This is because scanning printing apparatuses print at a uniform rate while a page is being printed. That is, once page printing has begun, printing continues at a uniform rate until page printing ends. Accordingly, to assure that the print engine does not run out of data to print (a condition known as data underflow), the entire image to be printed is developed into a raster image in bit map memory before printing commences. The CPU is responsible for developing the raster image into bit map memory, and this process represents peak CPU usage. During actual printing of the image, however, the CPU needs only to monitor the bit map memory to ensure that the raster image is provided to the print engine at the same rate as the print engine prints it. This relatively simple task represents low CPU usage.

Correspondingly, and as further shown in Fig. 11, when CPU usage is at its maximum, memory usage is at its minimum. This is because bit map memory is full only at a period just prior to image printing. As described above, this corresponds to a period when CPU functions are minimal.

For the simplest print systems, such inefficiencies are tolerable. For example, to print a

This Sheet Has Been Replaced

simple black and white image at low resolution such as 300 dots per inch (dpi) on a standard 8½ X 11 inch sheet of paper requires approximately 1 megabyte (Mb) of bit map memory. The price of 1 Mb of memory is low enough so that it is acceptable for large portions of the memory to be unused for long periods of time. This remains so even if the bit map memory is double buffered, that is, 2 Mb of memory are provided. Double buffering is desirable because it allows the CPU to develop a succeeding page of image data into a raster image in one buffer at the same time that the CPU is printing a current page of image data from the other buffer. Double buffering helps to provide a continuous stream of raster image data to the print engine thereby obtaining better inter-page throughput in a relatively cost effective manner.

But, as resolution increases, and as greater capabilities (for example, halftone and color) are desired, such inefficiencies can no longer be tolerated. For example, at 600 dpi resolution, four times as much memory is needed than is needed at 300 dpi to store a full raster image. The effect becomes even more pronounced when halftone or color images are considered. For example, to provide sixteen grey levels for each pixel in a halftone image requires 4 bits per pixel. Multiplying four bits per pixel by 4 Mb to store a 600

This Sheet Has Been Replaced

dpi raster image results in a requirement for a 16 Mb memory to store a halftone raster image. When color is considered, the memory requirements become even greater. For example, to encode four colors per pixel with 8 bits for each color means that 32 bits are required for each pixel. Multiplying 32 bits by 4 Mb for a 600 dpi raster image results in a 128 Mb memory requirement to store a full color 600 dpi raster image. The price for such large amounts of memory is not acceptable for wide commercial acceptance, and becomes even worse when double buffering is provided to reduce inter-page delay.

It has been considered to increase the apparent size of physical memory through the use of virtual memory in which off-line memory devices such as a disk drive are used to provide additional memory space and which are connected to a printer controller through a communication link. The use of virtual memory in a printing apparatus has not proved entirely satisfactory. This is because of the uncertainty involved in ensuring that there is sufficient bandwidth in the communication link so as to provide timely pixel data to the printer engine and avoid data underflow.

Further, high quality raster-based devices, such as laser beam printers and ink jet or bubble jet printers and the like, have become widely available. These devices receive image data from a host computer

This Sheet Has Been Replaced

and generate a raster image in bit map memory for subsequent imaging and/or printing.

Fig. 1 shows the rasterization process by which each pixel in bit map memory is formed, and by which qualities (such as color, grey level or half toning) are given to each pixel. As shown in Fig. 1, the bit map raster 100 is divided into frames F which define areas in the bit map raster. For each pixel 101 in the frame, a mask M and a background B are defined. The mask M defines whether or not the pixel should be rendered into bit map memory and the location where the pixel should be rendered. The background B defines the qualities with which the pixel 101 should be rendered. For example the background B may comprise a color associated with the pixel or it may comprise a halftone density. If desired, a clipping region may also be defined to specify the effective areas of the mask and allow more precise control over the masks. The rasterization of pixel 101 in the bit map memory is the intersection of the frame information, the background information, clipping region information, and mask information, all as illustrated in Fig. 1.

As the versatility of raster printing devices increases, more sophisticated techniques for instructing how to form the raster image have been

This Sheet Has Been Replaced

developed. That is, it is not ordinarily practical to transmit a full raster image to the printer. Rather a descriptive language such as page description language ("PDL") is used. A page description language is a series of commands that describes all characteristics of each element on every page of the document, such as text, graphics, scanned images. The PDL may also include information that describes document layout and sheet handling (collation, etc.). The host computer transmits page description language to the printing device, and the device interprets the page description language so as to render the raster image.

Fig. 2 is illustrative of this process. Fig. 2 depicts modules in a printing device which accept page description language at one end and create a bit map raster image at the other end. As shown in Fig. 2, a communication interface module 102 receives PDL commands and passes them to a PDL interpreter 104. The PDL interpreter directs PDL commands to a graphic layer 105, a font module 106, and a storage pool 107 as appropriate to the particular PDL command being processed by the interpreter. A rasterizer 109 accepts output from the graphic layer and the font module and the storage pool, and forms a raster image in a bit map raster memory. Specifically, the

This Sheet Has Been Replaced

rasterizer 109 accepts an empty raster from bit map raster pool 110 and returns a filled raster to the raster pool. The filled raster is imaged by the printing device during which time a further empty raster (corresponding to a succeeding page) is being filled by rasterizer 109.

Printer commands like PDL commands are normally device-independent which means that the final document will look the same for the same PDL commands regardless of the printing device that receives the PDL commands. While this is advantageous in some respects, it has caused difficulty in that each different printer, because of its own unique characteristics, must be designed with its own unique PDL interpreter 104 which translates device-independent PDL commands into corresponding raster images.

SUMMARY OF THE INVENTION

It is an object of the present invention to use resources in a printer more efficiently so as to provide a printer having greater capability without the need of increasing the amount of physical resources available to it.

In one aspect, the invention is a printing apparatus in which an image is stored in a coded,

This Sheet Has Been Replaced

compact representation before it is rendered into a full rasterized image. The representation is such that the resources that are needed to render a full rasterized image can be predicted with great accuracy. Predictor functions are provided to predict resource usage so as to determine, prior to printing, whether the printer has sufficient resources to conduct printing. If sufficient resources are available, then printing is performed. If the resources are not available, trade-offs between resources can be made so as to allow printing to take place.

It is an alternative object of the present invention to provide an apparatus in which each of different printers does not necessarily has its own PDL interpreter.

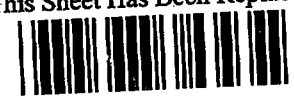
In one aspect, the invention generates document description commands using a graphic language based on print commands that include device-independent print commands. Frame descriptions for raster frames in the document are generated from the print commands as are document layout commands which define how the raster frames are positioned on the page raster. The frame descriptions are output on a first pipelined interface and the document layout commands are output on a second pipelined interface. The document layout commands may include information

This Sheet Has Been Replaced

as to how frames are mapped onto a sheet of paper, and may also include document finishing instructions such as stapling and collation instructions. The frame descriptions may be raster-based descriptions and are ordinarily device-dependent. The invention may be embodied in a raster-based printing or display device such as a laser beam printer or ink jet or bubble jet printer, but it also may be embodied in a printer driver in a host computer and executed as part of host computer processes.

In another aspect, the invention is a graphics language processor that includes a mask generator for generating pixel mask objects in a frame. A background generator is provided for generating backgrounds against which the objects are applied. An interpreter interprets a stream of graphic language commands input to the processor and activates the mask generator and the background generator in accordance with the graphics language. A rasterizer applies the mask objects to the background onto a bit map raster memory. The mask generator may include plural different mask generators and those plural mask generators may be organized into pipelined structures so as to be activated, in accordance with the graphics language, by the interpreter. Likewise, the background

This Sheet Has Been Replaced



generator may include plural background generators, each of which generates a different kind of background, and each of those background generators may be organized into pipelined structures so as to be activated by appropriate graphics language commands through the interpreter. A coordinator may be provided for coordinating the pipeline among the structures. The mask objects generated by the mask generator may be primitive objects which are compact representations of the pixel objects, or the mask objects may be segments of bit map images. If desired, a clipper may be provided so as to clip the output of the mask generator. Likewise, in certain embodiments, a dispatcher may be employed so as to sequentially select raster areas and to cause the mask generator and the background generator and the rasterizer to operate on the selected raster areas, respectively. For example, the dispatcher may cause banded partitions of the raster to be sequentially selected and operated on by the mask generator and the background generator and the rasterizer.

In another aspect, the invention is a printing method comprising the steps of inputting coded print data such as PDL print commands, translating the coded print data into a first coded representation, translating the first coded representation into a

This Sheet Has Been Replaced

second coded representation which is different from the first coded representation, and forming a raster image from the second coded representation. The first and second coded representations may include common code primitives, and the second coded representation may be susceptible of accurate modeling such that usage of the printer resources in rasterizing the representation may be predicted accurately so that the rasterization process may be changed should there be insufficient printing resources available.

This brief summary has been provided so that the nature of the invention may be understood quickly. A fuller understanding of the invention may be obtained by the following description in conjunction with the remaining drawinGL in which:

BRIEF DESCRIPTION OF THE DRAWINGL

Fig. 1 is an illustration showing a rasterization of pixels in a bit map raster memory;

Fig. 2 is a depiction of a PDL printer process;

Fig. 3 is a printer according to the invention;

Fig. 4 depicts a graphics language processor according to the invention;

This Sheet Has Been Replaced

Fig. 5 is a diagram for illustrating the interface between the page description language and a graphics language according to the invention;

Fig. 6 is a diagram illustrating how a mask is applied against a background onto a raster;

Fig. 7 is a detailed view showing mask generators and background generators according to the graphics language processor of the invention; and

FiGL. 8, 9 and 10 are view showinGL different forms of the graphics language processor, in which Fig. 8 is a sequential processor, Fig. 9 is a banded processor, and Fig. 10 is a pipelined processor.

Fig. 11 is a diagram illustrating CPU and memory usage during a print cycle;

Fig. 12 is a graph used to illustrate the resource envelope available to a printer;

Fig. 13 is a block diagram of a controller for the printer of Fig. 3;

Fig. 14 is a diagram illustrating memory usage in the printer of Fig. 3;

FiGL. 15 and 16 are flow charts illustrating operation of the Fig. 3 printer;

Fig. 17 depicts coded image representations according to the invention; and

FiGL. 18 to 22 depict object images shown in Fig. 17.

This Sheet Has Been Replaced

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[First Embodiment]

In a graphics language stream, graphic descriptions for frames in a page are separated from document layout information which describes how the frames are placed on the page, and in which the graphic descriptions and the document layout information are output on separate, pipelined, interfaces. A graphics language processor is provided to intercept the graphics language stream. The processor separates the graphics language stream into mask information and background information against which the masks are applied, operates independently on the mask and background information to create frame rasters, and uses the document layout information to map the frame rasters onto the page.

Fig. 3 is a laser beam printing apparatus according to the invention.

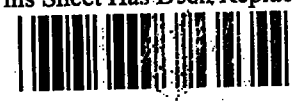
As shown in Fig. 3, laser beam printer 10 includes a controller 11 which controls the various mechanical and electrical functions within the printer. In particular, controller 11 accepts image data that is derived by an unshown host computer through communication port 12 and processes the image data so as to derive a laser modulation signal that

This Sheet Has Been Replaced

is applied to laser driver 14. The image data from communication port 12 is preferably in a very compact form such as a page description language ("PDL") which merely defines objects and the objects' location on the page rather than defining a full raster image of the print image, but any form for the image data can used. In accordance with the image signal provided by controller 11, laser driver 14 drives laser 15 so as to cause it to emit a laser beam 16. A rotatable polygonal mirror 17 scanningly deflects laser beam 16 via mirror 19 onto the surface of photosensitive drum 20. Photosensitive drum 20 is rotated in the direction of arrow A past charger 21. The scanningly deflected laser beam 16 forms a latent image on the surface of the charged photosensitive drum, and the latent image is developed into a toner image at developer 22.

In coordination with scanning the surface of drum 20, controller 11 causes the topmost sheet of paper from stack 24 to be fed by pickup roller 25 and conveyed by rollers 26 and 27 through image passageway 29 to photosensitive drum 20. The tonerdeveloped latent image on drum 20 is transferred to this sheet by transfer charger 30 and conveyed by rollers 31 and 32 to fixing device 34. In fixing device 34, a combination of heat and pressure fixes

This Sheet Has Been Replaced



the toner image permanently on the sheet and the sheet is ejected by rollers 35 onto output tray 36.

Fig. 4 is a view showing controller 11. In Fig. 4, communication interface 202 and PDL interpreter 204 are substantially similar to the communication interface and PDL interpreter shown in Fig. 2, and descriptions thereof have been omitted.

As shown in Fig. 4, page description language is processed by PDL interpreter 204 and is passed from PDL interpreter 204 to graphic language generation module 205. Graphic language generation module 205 translates PDL into a graphics language document according to the invention. In particular, in accordance with the PDL language, graphic language generation module 205 generates the graphics language document into two parts. First, the graphics language generation module 205 generates frame description commands in graphics language as shown at 206. The frame description commands describe raster frame sequence of primitive raster operations. Thus, the frame description commands define objects and rules on how to apply the objects onto the raster frames. For example, the frame description commands might instruct the graphic language processor to place a box-shaped set of pixels over a color image background and to position the resulting pixels at a

This Sheet Has Been Replaced

specific location in bit map raster memory. Or the frame commands might instruct the graphics language processor to form a corporate letterhead as a first frame, a text image comprising a letter as a second frame, and an image of a signature as a third frame. Frame description commands are described more fully below.

As further shown in Fig. 4, the second output of graphic language generation module 205 is document layout information 207. Document layout information describes how to apply the frames defined by frame description commands 206 onto a bit map raster that corresponds to a document page. Specifically, the document layout information includes information as to how the frames are mapped onto the raster memory. For example, document layout commands might command the graphic language processor to map several frames onto different and possibly overlapping positions in the bit map raster for a page. In the case of printable documents, the document layout information may include specialized printing instructions such as duplex printing, and it may also include paper sheet handling and finishing instructions, such as stapling and collation instructions.

The graphic language commands 206 and the document layout information 207 are output on the

This Sheet Has Been Replaced

pipelined outputs across interface 208. Interface 208 may be a physical interface between di-screte devices such as the interface between a host CPU and a print engine. For example, it is possible to implement communication interface 202, PDL interpreter 204, and graphic language generation module 205 as a printer driver in a host CPU. In that case, page description language commands that are generated in the host computer are also converted into graphic language commands 206 and document layout information 207 and are output across a pipelined interface directly to a printer engine. In the embodiment described here, however, interface 208 is an interface between two modules in controller 11. The modules may be implemented either in software or in hardware.

Graphic language commands 206 are input to graphic language processor 209 which renders the graphic language commands into an empty raster that is provided from bit map raster pool 210. The filled raster is provided to a print engine video interface 212 which provides the input to laser driver 14 of the printer.

Document layout information 207 is provided to system scheduler 214 which coordinates operation of the various modules. In particular, system scheduler

This Sheet Has Been Replaced

214 provides an empty raster from bit map raster pool 210 to the graphic language processor 209 and instructs the graphics language processor to begin rendering pixel images into the raster. When the graphic language processor 209 completes rendering the raster image, system scheduler 214 instructs print engine video interface 212 to begin providing data to laser driver 14 in accordance with the rendered raster image. In synchronism with modulating the laser beam, system scheduler 214 instructs print engine module 215 to begin operation of the various mechanical aspects of printer 10, such as sheet feeding, drum rotation and roller fixing. System scheduler 214 also instructs duplex printing, paper output, stapling, collation and other document finishing information in accordance with the document layout information 207.

Fig. 5 is a detailed view of the graphic language processor 209. As seen there, the processor 209 includes a graphic language interpreter 220 which parses a stream of graphic language commands and interprets the graphics language commands. In particular, the graphic language interpreter stores graphic language objects 224 as those objects are encountered in the stream of graphic language commands. The interpreter also passes mask

This Sheet Has Been Replaced

application commands to mask generator 221 and passes background generation commands to background generator 222 as those commands are encountered in the stream of graphics language commands. Mask generator 221 generates mask information such as mask M shown in Fig. 1; likewise, background generator 222 generates background information such as background B shown in Fig. 1. Both mask generator 221 and background generator 222 draw upon graphic language objects stored in GL object storage 224.

The masks generated by mask generator 221 are passed to clipper 225 which clips the masks, if desired, in accordance with the GL commands. Clipper 225 also draws upon graphic language objects stored in GL object storage 224.

The output of clipper 225 and the output of background generator 222 is provided to rasterOP machine 226. RasterOP machine 226 functions to apply the mask generated by mask generator 221, as clipped by clipper 225, onto the background generated by background generator 222. The rasterOP machine applies the mask in accordance with an application function defined by GL state memory 227. For example, as shown in Fig. 6, an opaque application function has been specified. Accordingly, when the rasterOP machine applies the mask generated by

This Sheet Has Been Replaced

generator 221 and clipped by clipper 225 onto the background plane generated by generator 222, any information in the initial raster is obliterated when forming the resulting raster. Other application functions are possible, for example, transparent application functions or additive application functions.

The raster on which the rasteroP machine 226 operates is provided from a bit map raster pool 229 as coordinated by system scheduler 214. When all applications have been finished, as set forth in the document layout information 207, the rasterOP machine provides a filled raster back to the bit map raster pool and the filled raster is printed by printer 10.

Fig. 7 is detailed conceptual view showing the construction of mask generator 221 and background generator 222, as well as showing the application process of rasterOP machine 226. In Fig. 7, some of the elements shown in Fig. 5 have not been depicted so as to simplify the following presentation.

As shown in Fig. 7, mask generator 221 includes several mask generators that have been designed specifically to produce specialized pixel primitive objects. In particular, mask generator 221 includes a bit map mask generator 221a, a trapezoidal mask generator 221b, a line generator 221c, a run

This Sheet Has Been Replaced

length mask generator 221d, a box mask generator 221e, and a generator 221f that generates other compact representations. In more detail, bit map mask generator 221a produces a bit map mask in response to GL commands from GL interpreter 220. Bit map masks are bit-for-bit pixel representations of a mask.

Trapezoidal mask generator 221b generates a trapezoidal-shaped mask in response to GL commands from GL interpreter 220. Trapezoidal masks define trapezoidal objects in raster memory.

Line generator 221c generates line drawinGL of specified thickness in response to GL commands from GL interpreter 220.

Run length mask generator 221d encodes run length masks based on the run length of pixels within the mask.

Box mask generator 221e generates a boxshaped mask of specified width and height in response to commands from GL interpreter 220.

Other compact representations are possible, and the above set should not be viewed as limiting, but rather only as representative of the types of masks generated by mask generator 221.

This Sheet Has Been Replaced

Background generator 222 includes color background generator 222a, tile background generator 222b, and image background generator 222c.

In more detail, color background generator 222a responds to GL commands from GL interpreter 220 and generates a solid color background of specified color.

Tile background generator 222b generates and replicates a tile specified by GL interpreter 220. The tile pattern is replicated across the entire background in the area specified by the GL interpreter.

Image backaround generator 222c provides an image background in a pixel-for-pixel format as specified by GL interpreter 220.

Other background generators are possible, and the above list should be viewed not in a limiting sense, but rather only as representative of the type of background generators that may be included in background generator 222.

Fig. 7 also illustrates how a current raster 230 is updated by rasterOP machine 226 in response to graphic language commands so as to produce updated raster 231. As shown in Fig. 7, GL interpreter 220 has received the graphic language command "apply clipped box (W,H) at <x,y> with image background".

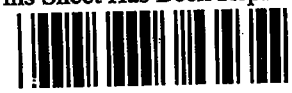
This Sheet Has Been Replaced

(A list of graphic language commands is given in Appendix A.) In response to the first portion of the graphic language command, namely "apply clipped box (W,H)", the GL interpreter 220 activates box mask generator 221d. Box mask generator 22le generates a box-shaped mask of width W and height H as depicted illustratively at the output of mask generator 221. Thereupon, clipper 225 draws upon the graphic language objects stored in memory 224 to provide a clipped shape as depicted illustratively adjacent clipper 225. The resulting clipped box is provided to rasterOP machine 226.

In response to the later portion of the graphic language command, namely "with image background", GL interpreter 220 activates image background generator 222c. The image background generator 222c draws upon the graphic language objects stored in memory 224 and provides the image background, which may be a color image background, to rasterOP machine 226.

In response to the information provided by clipper 225 and background generator 222, rasterOP machine 226 applies the clipped mask to the background in accordance with the application function specified by state memory 227. Here, the application function is specified by "opaque", in

This Sheet Has Been Replaced



accordance with which the rasterOP machine 226 applies the masked and clipped area of the background at position <x,y> of the current raster so as to obtain the updated raster shown at 231. The foregoing steps are performed for each command in the stream of graphic language commands for each frame, after which the document layout information 207 is utilized to determine how the frame is applied to the overall bit map raster.

Fig. 7 has been explained as if the process were a sequential process, as depicted in Fig. 8. That is, in step 1, mask generator 221 generates a mask that is provided to clipper 225. In step 2, clipper 225 clips the mask appropriate and provides it to rasterOP machine 226. In step 3, background generator 222 provides a background to rasterOP machine 226, and in step 4, rasterOP machine 226 applies the mask to the background and writes it into the raster image in the bit map raster memory. While this is one possible implementation of the graphic language processor, other implementations are possible.

Thus, as shown in Fig. 9, dispatcher 330 is provided to coordinate the activities of the mask generator 321, clipper 325, background generator 322 and rasteroP machine 326. Specifically, dispatcher

This Sheet Has Been Replaced

330 instructs each of these modules to operate on a specific area of the raster image, such as area 330a. Typically, such an area is a band in a partitioned and banded area of bit map memory, but this is not necessarily the case, and the area may be any arbitrary area in the image. Dispatcher 330 serially sequences through the entire area, for example, from band to band, until the entire page in bit map memory has been rendered. Thus, dispatcher 330 causes mask generator 321, background generator 322, clipper 325 and rasterOP machine 326 to operate on sequential areas of bit map memory but to operate on only a single area at one time. Graphic language objects that are specified for other areas are temporarily ignored and are processed only at a later time when dispatcher 330 indicates that that area should now be processed. Because of this, memory overhead may be necessary for the Fig. 9 structure to store temporarily the graphic language commands. The overhead may be minimized by insuring that the areas sequentially selected by dispatcher 330 are large enough so that the amount of graphic language commands that need to be stored is minimized.

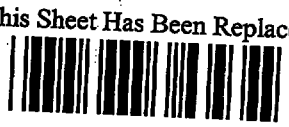
Fig. 10 shows another implementation of the graphic language processor in which a coordinator 440 is provided to coordinate a pipeline between mask

This Sheet Has Been Replaced

generator 421, background generator 422, clipper 425 and rasterOP machine 426. Specifically, in the implementation shown in Fig. 10, each of the modules is a pipelined module designed to operate on whatever information is currently first available in its pipeline. Each module operates on its own pipelined instruction in parallel and at the same time as described generally in "Computer Architecture: A Quantitative Approach", D.A. Patterson and J.L. Hennessy, Chapter 6, Morgan Kaufman Publishing Company (1990). As such, coordinator 440 is provided to ensure that the modules start in-synchronism and remain in synchronism throughout operation. Thus, for example, whereas background generator 422 will provide a background directly to rasterOP machine 426, mask generator 421 is not able to provide a corresponding mask object directly to the rasterOP machine because clipper 425 is interposed between mask generator 421 and rasterOP machine 426. Thus, coordinator 440 ensures that the clipped mask which is output by clipper 42s is associated with the proper background output from background generator 422 so that the rasterOP machine 426 operates on associated pieces of information.

The graphics language processor described hereinabove may be combined with a printer of another

This Sheet Has Been Replaced



architecture such as that described hereinafter. In combination with the techniques described in that application, it is possible to translate the graphic language commands described herein into a further DART representation. The DART representation is susceptible of accurate modeling of printer resources whereby it is possible to determine before a printing operation commences whether there are sufficient printer resources to complete the printing operation. If there are sufficient resources available, then printer commences; on the other hand, if there are insufficient printer resources then the printer operation may be changed so as to ensure that the printing operation can be completed.

[Second Embodiment]

Next, another embodiment is described in which the above-mentioned embodiment is preferably applied.

The present embodiment relates to a method and apparatus for printing an image in which a printing operation can be changed before the printing operation commences in accordance with predictions as to how much of the printer's resources will be used in printing the image, as mentioned in the last of before embodiment. The image is stored in a coded representation before being converted into a full raster image, the coded

This Sheet Has Been Replaced

representation being susceptible to accurate modeling of resources used in converting to a full raster image. Accordingly, the usage of printer resources in converting the coded representation of the image into a full raster image can be predicted accurately and it can be determined before printing whether the printer has sufficient resources to print the image. In the event that the prediction indicates that there are insufficient resources, resources can be re-allocated so as to permit the image to be printed.

Because the coded representation allows dynamic and adaptive use of printer resources, the representation is referred hereafter as the dynamic adaptive resource (or resolution) technology, or "DART", representation.

Consider, for example, the resources depicted graphically in Fig. 12, namely, CPU power, memory, and bandwidth. CPU power refers to the speed and ability with which a CPU in a printer controller can manipulate data and process it from one form to another. Memory refers to the amount of memory available to the printer. Bandwidth refers to the bandwidth or data-carrying speed of a communication link to an external memory device. Included in bandwidth are any known delays of the external memory device, for example, access time, disk head delays, and disk rotation speed.

This Sheet Has Been Replaced

Any one printer will have an envelope of resources available to it defined by a point representing the maximum CPU power, the maximum memory, and the maximum bandwidth. If predictor functions predict that rendering the DART representation into a full raster image will exceed one of these resources, then resource trade-off can be conducted. For example, if it is determined that there is insufficient memory to store the image, but that CPU power is underutilized (represented by point A in Fig. 12), then it is possible to substitute a different DART representation, one which uses less memory but which requires more CPU power to render the representation into a full raster image. Thus, memory has been exchanged for CPU power. This trade-off shifts point A to point A' which is within the resource envelope available to the printer and allows printing to commence.

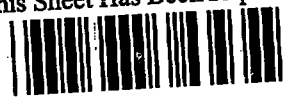
It is also possible to degrade the image such as by lowering the print resolution or by limiting the number of grey levels for halftone images or colors for color images. Degrading the image eases memory requirements without putting a further burden on the CPU. This trade-off results in a shift of point A to point A" which is within the printer resource envelope and which allows printing to commence.

This Sheet Has Been Replaced

Conversely, the predictor functions may predict that the CPU has insufficient CPU power to render a full raster image from the DART representation in sufficient time to provide the raster image to the print engine, that is, there would be data underflow. If there is sufficient memory available (represented by point B in Fig. 12), then CPU power may be exchanged for memory space, e.g. by pre-rasterizing the portion of the image that requires too much CPU power. This trade-off shifts point B to point B' which is within the resource envelope of the printing apparatus so that printing may commence.

Likewise, it is also possible to trade-off CPU power and/or memory with bandwidth. For example, in a case like point C in which it is predicted that maximum CPU power is already being expended to render a full raster image from the DART representation, but that memory resources are still exceeded, then it is possible to trade-off memory for bandwidth, e.g. by causing a portion of the image to be stored off-line and retrieved when needed through a communication link. This shifts point C to point C' which is within the resource envelope of the printing apparatus. The predictor functions ensure that the bandwidth is sufficient so that raster image data is timely provided to the print

This Sheet Has Been Replaced



engine and the print engine does not suffer from data underflow.

Fig. 13 is a block diagram of controller 11. Each and every physical connection in controller 11 is not depicted in Fig. 13, but rather, Fig. 13 is intended to show the flow of processes within the controller.

As shown in Fig. 13, controller 11 includes CPU 41 which accepts image data such as page description language from communication interface 12. CPU 41 converts page description language image information into a graphics language (or graphic stream language) using GL translator 42. Graphics language is a language which encodes image data at a level intermediate the highly compact PDL level and the full raster image data level. As described more fully below, GL language includes primitives, such as image objects, and rules as to how those primitives are applied to bit map memory to form a full raster image.

From the GL representation at 42, CPU 41 generates a DART representation via generator 43. The DART representation created by generator 43 is a compact coded representation for a raster image in which the resources used to convert from the DART representation to a full raster image can be predicted with great accuracy. The DART representation generated by generator 43 uses many of the same primitives as the GL

This Sheet Has Been Replaced

representation. However, unlike GL which acts on a raster image as a whole, the DART representation generated by generator 43 acts only on banded partitions of the image. Accordingly, and because more complex primitives found in the GL representation are not used, accurate predictors of resource usage can be provided for the DART representation whereas such predictors cannot be provided for GL.

The DART representation generated by generator 43 for each band is stored in memory 44. As printer engine 45 requires data which corresponds to scans of laser beam 16 on photosensitive drum 20, the representation stored in memory 44 is rendered into a full raster image of the band by renderer 46 and shipped by shipper 47 to the printer engine.

Fig. 14 is a representational view of how an image is partitioned (or banded) in accordance with the DART representation generated by generator 43. As shown in Fig. 14, page images are partitioned into bands that correspond to the scan direction of the print engine, here, the horizontal direction. Typically, about 27 bands are sufficient to process an 8$_{1/2}$ X 11 inch image efficiently, although this many bands have not been shown in Fig. 14 for the sake of simplicity.

Memory 44 is the same memory area that in conventional printers corresponds to the bit map memory.

This Sheet Has Been Replaced

The memory is used as a queue: as DART images are generated the memory is used up and as they are rendered and shipped out to the printer engine the memory is freed. Whenever a new page is to be generated, memory is allocated at the start of free memory. Wrap-around from the end of physical memory to the beginning thereof is provided for. Once all DART objects and applications have been generated for the page, and band raster storage appended to the page, subsequent pages are started at the end of the current page's memory. Once that page has been completely shipped, its band rasters, objects, and page information structure are de-allocated and made available for re-use.

Each of the bands shown in Fig. 14 is processed serially by generator 43. Thus, for example, generator 43 first processes $band_0$ for page i, and stores the DART representation so obtained at the top of memory 44. Generator 43 then processes $band_1$ of page i and stores the DART representation so obtained in the next sequential area in memory 44. Processing continues until the last band of page i, corresponding to $band_k$, at which point a DART representation for the entire page is stored in memory 44. At this point, a full rasterized image does not yet exist for page i, but rather only the banded DART representation.

This Sheet Has Been Replaced

As each band is processed by generator 43, CPU 41 uses predictor functions to ensure that the resources available in printer 10 will not be exceeded when the DART representation stored in memory 44 is rendered into a full raster image. Such predictions are possible because of the structure of the primitives in the DART representation and because of the certainty of the code that renders the full raster image from the DART representation. Assuming that each of the bands can be rendered without exceeding the resources of the printer, CPU 41 appends two blank band rasters (i.e. band $raster_0$ and band $raster_1$) to the end of page i in memory 44. Each of the blank band rasters is the size of a full bit map image for a single band at the desired print resolution.

Generator 43 then proceeds to generate a banded DART representation for the next page, namely page i + 1.

Meanwhile, renderer 46 begins to render the banded DART representations of the page i image into a full raster image on the blank bands. First, renderer 46 renders $band_0$ into band $raster_0$. Then, renderer 46 renders the DART representation in $band_1$ into a full raster image at band $raster_1$. Meanwhile, while $band_1$ is being rendered, shipper 47 ships the full bit map raster image in band $raster_0$ to print engine 45 so as to output

This Sheet Has Been Replaced

that band. By the time that band $raster_0$ has been shipped to engine 45, renderer 46 has rendered a full raster image for printing $band_1$ in band $raster_1$. The availability of sufficient resources to provide the full raster image in time to avoid print engine data underflow has been guaranteed before printing by the predictor functions.

Shipper 47 then ships the full raster image available in band $raster_1$. Meanwhile, renderer 46 renders a full bit map raster image from the raster representation stored in $band_2$. This process continues until the entire page i has been rendered by renderer 46 and shipped to print engine 45 by shipper 47.

During the period when the DART representation of page i is being rendered into a full raster image and shipped for printing, any free CPU time is being utilized to convert the PDL for the next page, namely, page i + 1, into a DART representation in the same manner as described above for page i.

In the above process, if CPU 41 determines, through the use of the predictor functions, that rendering a full raster image from the DART representation will exceed the resources available, then resource trade-off or re-allocation is performed. For example, if it is determined that the DART representation is larger than a full raster image of the

This Sheet Has Been Replaced

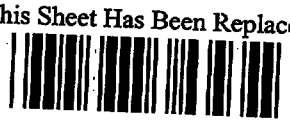
print information, indicating that the DART representation is less efficient than a full raster image, then if CPU power is still available, a different and more compact DART representation may be generated by generator 43. Conversely, if it is determined that CPU 41 will not have sufficient power to render a full raster image from the DART representation in the time required to process a band of information, which would result in print engine data underflow, then it is possible to pre-rasterize the offending band (or to pre-rasterize all or some of the page containing the offending band) so that already-rasterized data can be supplied to shipper 47 at the appropriate time.

The overall process is depicted in Fig. 15. The process steps in Fig. 15 are stored in an unshown memory in controller 11 and are executed by CPU 41.

At step S601, controller 11 inputs image data from communication port 12. The image data is typically generated by an unshown host computer which sends the image data to controller 11 in a compact code such as a page description language.

At step S602, translator 42 translates the PDL image data into GL image data. At step S603, generator 42 generates a DART representation from the GL data and stores it in memory 44. As the DART representation is generated, it is banded in step S604, and in step S605

This Sheet Has Been Replaced



CPU 41 maintains a running tally of resource usage that will be required in rendering the DART representation into a full raster image. Specifically, for each of the different DART representations, predictors are provided that predict (a) CPU power required to render the DART representation, and (b) memory required to store the DART representation. If printer 10 were equipped with external memory linked to CPU 41 via a bandwidth-limited communication line, then a third predictor would be provided as to the bandwidth requirements for each DART representation.

The resources described here, namely CPU power, memory and bandwidth, are representative of the types of resources that printer 10 may be provided with. For each additional resource, corresponding predictor functions are likewise provided and used by CPU 41 to tally resource usage.

In step S606, CPU 41 determines, based on the tally from the predictor functions, whether resource usage will be exceeded in rendering the DART representation for the current band in a full raster image in the band rasters. Step S606 may be executed periodically during DART generation, e.g. after each DART representation is generated, or it may be executed after the DART representation for the full band is generated. In the case of determining whether memory

This Sheet Has Been Replaced

usage is exceeded by the DART representation, it is preferred to execute step S606 periodically, for example, every time a new DART object is created.

If in step S606 CPU 41 determines that resource requirements to render the current band can be met, then flow advances to step S607 where CPU 41 determines whether there is additional GL code for the current page. If there is more GL code for the page, then flow returns to step S604 until the remaining image data has been processed.

When DART representations for all bands in the page have been generated, flow advances from step S607 to step S608 where a pair of blank band rasters are appended to the DART representation stored in memory 44. If flow has advanced to this step, then it is known that the resources available in the printer are adequate to render the DART representation band-by-band into a full raster image without the danger of exceeding printer resources, e.g. there is enough CPU power to avoid data underflow to the print engine.

In step S610, the DART representations are rendered band-by-band into alternating ones of the pair of band rasters, as described above in connection with Fig. 14. Then in step S611, shipper 47 ships the raster image to print engine 45. Steps S610 and S611 are executed concurrently and sequentially so that, as

This Sheet Has Been Replaced

described in connection with Fig. 14, as shipper 47 finishes shipping one band of raster image from one band raster, renderer 46 has completed rendering the next band of DART representation into the other band raster.

It is also possible to provide more than two blank band rasters, in which case renderer 46 completes rendering of a fixed number of bands ahead of shipper 47.

After the current page has been printed, or as CPU time permits during rendering and shipping in steps S610 and S611, CPU 41 begins processing the next page (step S612) in accordance with the overall flow shown in Fig. 15.

If in step S606 CPU 41 determines that there are insufficient resources available in the printer, it is known that the resources needed to render or store the current DART representation exceed the resource envelope of the printer. Flow then advances to step S613 where CPU 41 performs resource trade-off or re-allocation in an attempt to fit the DART representation into the printer resource envelope. Initially, a different DART representation is selected for the offending band. For example, if the memory resource has been exceeded, then a memory-intensive DART representation may be replaced by a different representation which uses comparatively less memory. Likewise, if the CPU resource has been

This Sheet Has Been Replaced

exceeded, then a CPU-intensive DART representation may be replaced by a different representation which uses comparatively less CPU. For example, the offending band may be pre-rasterized and stored in memory 44 so that it is ready for printing by the print engine without any further CPU processing.

In step S614, resource usage with the different DART representations is re-predicted. If resource usage now falls within the resource envelope then flow proceeds to step S607 and continues as described above. If resource usage is still exceeded, then further attempts may be made to bring resource usage within the printer envelope (steps S613 and S614).

Alternatively, or in the case where it is not possible for the DART representation to be brought within the resource envelope, then the image is degraded prior to printing. Degrading automatically reduces resource requirements. In the example given here, the image is degraded by reducing the resolution (or pixel density) from 600 dpi to 300 dpi. Image degradation may take other forms: for a halftone image, degradation may also include reducing the number of grey levels; for a color image, degradation may also include reducing the number of colors.

In step S616, in preparation for resolution degradation, processing is completed for all previous

This Sheet Has Been Replaced

pages in memory 44, that is, DART representations for prior pages are rendered and shipped as shown at steps S610 and S611.

In step S617, image resolution is degraded by rendering the DART representations that have already been generated by generator 43. The DART representations are rendered in situ in memory 44 but at the degraded resolution. Then, GL that has not already been processed is converted to a degraded raster image in memory 44. The conversion is either direct into the degraded raster image or indirectly by first generating a DART representation and then rendering.

In step S618, shipper 47 ships the degraded raster image to the print engine. Preferably, when degradation is necessary, it should be complemented with some form of image enhancement so as to minimize the effects of degradation. For example, in the case of resolution degradation, smoothing circuitry such as that described in U.S. Patent No. 4,878,068 may be employed at the time the degraded image is shipped so as to soften the appearance of the printed degraded image.

In step S619, processing for the next page begins but without degradation.

Steps S613 through S619 provide repetitive attempts to reallocate resource usage so that sufficient resources are available for the DART representation. In

This Sheet Has Been Replaced

a simplest form of resource trade-off, however, and in the embodiment presently preferred, resource trade-off takes the form of immediate image degradation when it is determined that resources have been exceeded, whereby the color, grey scale or pixel density (or resolution) is degraded. In the case of a simple black and white image which is being rendered into a 600 dpi raster image, for example, resolution can be degraded to 300 dpi and a full raster image can be stored in memory 44. In this instance, the DART representation is converted in situ into a degraded-but-full raster image of the image data, whereupon the full raster image is shipped via shipper 47 to print engine 45. If desired, image smoothing techniques can be applied, such as those described in U.S. Patent No. 4,878,068, so as to improve the appearance of the lower resolution image.

Fig. 16 shows the simplest form of resource trade-off. In Fig. 16, steps S601 to S604 are the same as in Fig. 15 and a description thereof is omitted.

In step S605 CPU 41 maintains a running tally of resource usage that will be required in rendering the DART representation into a full raster image. As in the Fig. 15 embodiment, for each of the different DART representations, predictors are provided to predict CPU power, memory usage, bandwidth, and so on. In addition, so as to ensure that there is sufficient memory to

This Sheet Has Been Replaced

rasterize a degraded image in bit map memory (if degradation is needed), a running tally is maintained of the memory that will be required to rasterize a full bit map image at the degraded resolution.

Steps S607 through S608 are the same as in Fig. 15 and their description is omitted.

In step S606, if it is determined that resource usage has been exceeded, or if it is determined that the memory required to rasterize a full bit map image at the degraded resolution exceeds available memory, then the image is immediately degraded so as to bring resource usage within the printer resource envelope. Image degradation is performed in steps S620, S621, S622 and S623, which are the same as steps S616, S617, S618 and S619 in Fig. 15, respectively.

The various DART representations of the image that are currently preferred, and the predictor functions associated with these representation, are given as follows.

The primitives used for the representation are objects and backgrounds. The DART representation comprises a series of applications of various ones of these objects to various ones of the backgrounds.

A background consists of either a bit map tile or an image. Bit map tiles are derived from the GL image representation from translator 42. Color may be mapped

This Sheet Has Been Replaced

onto a bit map tile by mapping colors into color planes of corresponding tiles. An image is an image representation and is optimized for fast rendering and prediction.

The object primitives that are utilized in the raster representation are preferred because their renderinGL into full raster images can be predicted with certainty and with accuracy. The preferred set of object primitives include bit maps, run length sequences, trapezoid sequences, boxes and fast boundary encoded bit maps. Each of these objects is defined with greater specificity below.

A single application of a DART-encoded image representation to a fully rasterized bit map memory is depicted in Fig. 17. There, a background function defined by a tile or an image is provided to an application machine (rasterOP machine in the first embodiment). An object such as a bit map, run length, trapezoid, box or fixed boundary encoded map is also provided to the application machine. The application machine applies the object onto the background at a specified area in the bit map for the band under consideration so as to convert the initial raster in the bit map into the resulting raster shown in Fig. 17.

Different rendering functions are used for different DART objects, and each rendering function is

This Sheet Has Been Replaced

tailored to render its associated DART object as efficiently as possible. Preferably, the rendering functions are optimized for the speed on the controller's CPU and for the code compactness. Suitable rendering functions are described by Foley, van Dam, Feiner and Hughes in "Computer Graphics: Principles and Practice" (2nd Edition), published in 1987 by Addison-Wesley Publishing Company.

Predictors for the DART representation predict both the memory required to store the DART representation as well as the time (or CPU power) needed to render the DART representation. In general, the memory required to store the DART representation while rendering the i-th band in an image is

$$MemoryRequired_i = \sum_{i \le k \le n} BandSize_k + 2 \times BandRasterSize$$

where $BandSize_k$ is the amount of memory that will be freed after the k-th band is rendered, *BandRasterSize* is the size of the fully rasterized band rasters appended to the end of page memory, and n is the number of currently-stored bands of DART objects. CPU 41 ensures that this memory requirement does not exceed available memory. In embodiments like those shown at Fig. 16, where images are rendered at degraded resolution when there are insufficient resources available for printing

This Sheet Has Been Replaced

at higher resolution, CPU 41 also ensures that memory requirements do not exceed what is needed to render the image at the degraded resolution, namely:

$$DegradedMemRequired_i = \sum_{i \le k \le n} BandSize_k + (i + l) \times BandRasterSizeDegraded$$

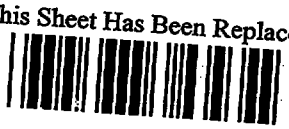
where *BandRasterSizeDegraded* is the size of band rasters at the degraded resolution.

As for predictors of the time (or CPU power) needed to render the DART representation, since rendering involves the generation of both objects and backgrounds, as well as the application of the objects and backgrounds to the raster, the time required to render is dependent on the sum of the times for these functions. The predictor functions reflect this dependency. In practice, an overhead factor is also added to accommodate system overhead processes in the rendering process.

Detailed descriptions of the objects and their associated predictor functions follow. In general, however, predictor functions for CPU power all have the form

$$Time = (ApplicationTime + BackgroundTime) \times OverheadFactor$$

This Sheet Has Been Replaced



where *ApplicationTime* is the time needed to apply the DART object, *BackgroundTime* is the time needed to expand the background, and *OverheadFactor* is the aforementioned overhead factor to accommodate system overhead processes. *BackgroundTime* depends on the area of the background that is expanded for rendering and is also dependent on the type of background and its expansion algorithm.

A bit map object, as its name implies, is simply a bit map raster image of an object. Thus, as shown in Fig. 18, the object 71 includes an identification label followed by a bit map rasterized image of the object 71.

The application of a portion 72 of the bit map object into a band is also depicted in Fig. 18. As shown there, the object is to be applied on the band beginning at a displacement *disp* from the beginning of the object and is to be applied at a position *by, x* and for a height *ht* within the band. The application is shown at 73 and includes a mode definition which defines the background over which the object is to be applied, as well as the *bx* and *by* coordinates within the band at which the object is to be applied, and the height *ht* over which the object is to be applied. The ID label for the object is specified and the displacement within the object is included.

This Sheet Has Been Replaced

The predictor function for the CPU power needed to render a bit map application is

$$Time_{BitMap} = (N_1 \times ht + Background) \times OverheadFactor$$

where N_1 is an implementation factor related to the number of CPU cycles needed to apply each row of the bit map object.

Run length objects are depicted in Fig. 19. Run length objects are segmented, each segment describing successive scan lines and each scan line consisting of a single run length. Thus, as shown in Fig. 19, run length object 81 includes segments 1 through segments 4.

The DART representation of a run length object is shown at 82. The object includes an ID label and a segment index which contains information about the segment rendering time estimation. This information is used by the predictor function. Each segment contains the description of up to a predetermined number of run lengths and each segment is aligned to byte boundaries.

A segment defines successive run lengths in a differential manner, by modifying the segment state while stepping through. The segment consists of two pairs of contour states (left and right). Each contour consists of a position x and an increment dx.

At the beginning of the segment:

$$xl_0 = dxl_0 = xr_0 = dxr_0 = 0$$

This Sheet Has Been Replaced

For each scan line i the segment defines an advance step of the form: $step_i = <advl_i, advr_i>$, that is used as follows:

$$dxl_i = dxl_{i-1} + advl_i$$

$$dxr_i = dxr_{1-1} + advr_i$$

$$xl_i = xl_{i-1} + dxl_i$$

$$xr_i = xr_{i-1} + dxr_i$$

The application of a portion 83 of a run length object 81 in a band 84 is also shown in Fig. 19. There, portion 83 of the run length object 81 is applied at position by, x within band 84 and continues for a height ht. The coded representation for this application is shown at 85 and includes a mode which defines the type of background over which the run length object is applied. X, by and ht coordinate information is provided, as well as an ID label, the segment from the segment index, and the offset from the beginning of the segment.

The predictor function for the CPU power needed to render a run length object is

$$Time_{RunLength} = (N_2 \times ht + N_3 \times area + Background) \times OverheadFactor$$

where the parameters N_2 and N_3 are similar to those for bit map objects.

This Sheet Has Been Replaced

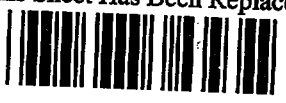


Fig. 20 depicts trapezoid objects. Trapezoid objects are not segmented, in contrast with run length objects. This is because it is assumed that trapezoid sequences are usually not intended to be reused (they describe large areas of the raster).

Each object describes successive scan lines, with each scan line consisting of a single run length.

A trapezoid sequence defines successive run lengths with a DDA algorithm. The state consists of two pairs of contour states (left and right). Each contour state consists of a position *x* and a slope.

At the beginning of the trapezoid sequence:

xl = slopel = xr = sloper = O

The object consists of a sequence of actions. These actions are either modify or paint. A modify action is a pair:

modify = < modx, modslope >

which affects the state

x + = modx;

slope + = modslope;

A paint action defines a number count of run lengths by iterating the following steps count times:

xl + = slopel;

xr + + sloper,

y + +;

This Sheet Has Been Replaced

IssueRunLength (xl, xr, y);

A paint action with count zero denotes the end of the trapezoid sequence.

Fig. 20 also shows the application of a portion 93 of a trapezoid object into a band 94. As shown there, portion 93 is applied to band 94 at position *by, x* and continuing for a height *ht* in the band. The application is shown at 95, and includes a mode which defines the background against which the object is applied, as well as the *x, by* and *ht* coordinates within the band. The ID label for the object is provided. No additional information is needed because trapezoidal objects are not segmented and the entire trapezoid is applied for the distance *ht*.

The predictor function for the CPU power needed to render a trapezoid application is

$$Time_{Trapezoid} = (N_4 \times ht + N_5 \times area + Background) \times OverheadFactor$$

where N_4 and N_5 are parameters similar to those for bit map objects.

Box objects are depicted in Fig. 21. Rectangular boxes are such a common object that they merit their own optimized representation. In this case there is no need

This Sheet Has Been Replaced

to have an object per se, since the application defines the box, as shown in the figure.

The application of a box object to a band 104 is also depicted in Fig. 21. As shown there, the box object is applied at position *by, x* and continues for a height *ht* and a width *w* in the band. The application is defined by 105 which includes a mode word which defines the background against which the box is applied, and includes coordinates *x, by* and *ht* for the application. A null word is provided followed by the width of the box.

The predictor function for the CPU power needed to render a box object is

$$Time_{Box} = (N_6 \times ht \times w + Background) \times OverheadFactor$$

where N_6 is a parameter similar to those for bit map objects.

Fast boundary encoded (FBE) objects are depicted in Fig. 22. An FBE segment describes successive scan lines, and each scan line consists of multiple run lengths.

Again, the segment index contains information (*dt-fbePred-t* record) about the segment rendering time estimation, used by the predictor functions. This information is larger than the one used for run length objects.

This Sheet Has Been Replaced

A segment defines successive run lengths in a differential manner, by modifying the segment state while stepping through.

The segment state consists of multiple contour states. Each contour consists of a position *x* and an increment *dx*.

At the beginning of the segment:

x[k] = dx[k] = 0 for all values of k.

The segment consists of a series of commands, chosen among the following:

- *adv* : advance corresponding contour as in the case of RLSeqs.
- *close* : delete the next two contours.
- *open* : insert two new contours between the current contour and the next one.

For each scan line *i* the segment defines an advance step of the form

$step_i = <linechange, comm_1, comm_2..., comm_n>$

that is used to transform the segment state. *Linechange* specifies the change (increase or decrease) of the number of pairs of contours with respect to the previous scan line, or a reset at segment boundaries. The execution of $comm_i, comm_2, ..., comm_n$ brinGL the number of contours in accordance to *linechange*. A *linechange* that

This Sheet Has Been Replaced

makes the number of contour pairs negative denotes the end of the segment.

Fig. 22 also shows how a portion 114 of FBE object 111 is applied to band 115. As shown in Fig. 22, the portion 114 is applied to the band at position *by, x,* for a height *ht* beginning at an offset *doff* from the beginning of segment 3.

The application is defined at 116 which includes a mode word for defining the background against which the FBE object is applied. 116 further includes the *x, by* and *ht* coordinates of the application, as well as the ID label for the object, the segment that is to be applied, and the offset from the beginning of the segment.

The predictor function for the CPU power needed to render an FBE application is the sum of the CPU power needed for each segment, where the CPU power for each segment is dependent on the number of run lengths that need to be rendered and the height of the segment.

This Sheet Has Been Replaced

Appendix A

GRAPHIC LANGUAGE COMMANDS INSTRUCTION SET

This appendix is a summary of the GL commands instruction set. It defines instructions in terms of their effects on the system and semantic actions.

The notation below describes the stack manipulation the graphic processor performs when it executes an instruction:

Instruction arg_1 arg_2 ... arg_n Instruction $result_1$ ...$result_k$

where the sequence arg_1 arg_2 ... arg_n denotes the argument list on the stack (with arg_n at the top) that the instruction consumes. Execution of the instruction replaces the argument list with the sequence $result_1$ ...$result_k$.

ARITHMETIC AND RELATIONAL OPERATIONS

Use these instructions to execute arithmetic and relational operations on the system stack.

add $number_1$ $number_2$ GLop.add number (= $number_1$ + $number_2$)

Addition.

sub $number_1$ $number_2$ GLop.sub number (= $number_1$ -

number$_2$)

Substraction.

mul *number*$_1$ *number*$_2$ GLop.mul number (= *number*$_1$ * *number*$_2$)

Multiplication.

div *number*$_1$ *number*$_2$ GLop.div number (= *number*$_1$ /*number*$_2$)

Division (integer).

mod *number*$_1$ *number*$_2$ GLop.mod number (= *number*$_1$ mod *number*$_2$)

Modulus.

neg *number* GLop.neg number (= -*number*)

Arithmetic negate.

shright *number*$_1$ *number*$_2$ GLop.shright number (= *number*$_1$ >> *number*$_2$)

Shift right.

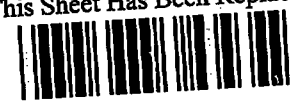
shleft *number*$_1$ *number*$_2$ GLop.shright number (= *number*$_1$ << *number*$_2$)

Shift left.

OTHER ARITHMETIC/RELATIONAL INSTRUCTIONS

GL commands support a number of extended operations, including vector arithmetic (using fixed coordinates) and the necessary fixed arithmetic trigonometric functions (fast approximations).

This Sheet Has Been Replaced



STACK MANIPULATION

These operations handle the system stack.

```
loadq_0   GLop.loadq_0              num(=0)
          loadq_1GLop.loadq_1       num(=1)
          loadq_2GLop.loadq_2       num(=2)
          loadq_3GLop.loadq_3       num(=3)
          loadq_4GLop.loadq_4       num(=4)
          loadq_5GLop.loadq_5       num(=5)
          loadq_6GLop.loadq_6       num(=6)
          loadq_7GLop.loadq_7       num(=7)
```

Use the loadquick family of instructions to load very quickly the system stack with the most commonly used arithmetic values.

pop *anyvalue* GLop.pop

Discards the top of the stack.

dup *anyvalue* GLop.dup *anyvalue anyvalue*

Duplicates the top of the stack.

exch $anyvalue_1$ $anyvalue_2$ GLop.exch $anyvalue_2$ $anyvalue_1$

Exchanges the top two cells in the stack.

OTHER STACK MANIPULATION INSTRUCTIONS

GL commands support a number of possible extension operations for handling the stack, including indexing, copying, etc.

RECORDS

Use records to store permanent information. The information in a record is structured as an

This Sheet Has Been Replaced

array. Graphic language commands cause records to be stored as GL objects in GL object storage (see below).

The concept of a current record serves as a mechanism to compress GL code: Once an object is declared the current record, all load and store operations refer to that object.

At the beginning of each frame, the system provides a default record.

record *object|d|* GLop.record

The object *object|d|* becomes the current record.

load *displ* GLop.load

Pushes the contents of memory at record location *displ* onto the stack.

store *value displ* GLop.store

Pops *value* on top of stack into record location *displ*.

ARRAY HANDLING

These instructions create arrays and access array elements on the stack.

OBJECT HANDLING

These instructions create and manipulate GL objects.

This Sheet Has Been Replaced

objdelete *objHandle* GLop.objdelete

Removes the object *handle* from memory.

objmake *objHandle size* GLop.objcopy

Performs a "delete" on the object *objHandle*, then creates a new object of size *size*.

markarrayGLop.markarray + n + b_1 + ... + b_n *inlineHandle*

markbitmapGLop.markbitmap + n + b_1 + ... + b_n *inlineHandle*

marktrap GLop.marktrap + n + b_1 + ... + b_n *inlineHandle*

markproc GLop.markproc + n + b_1 + ... b_n *inlineHandle*

These instructions declare an *inline* object, leaving an *inlineHandle* handle on the top of the stack for future reference.

objdef *inlineHandle objHandle* GLop.objdef

Performs "delete" on the object *objHandle*, creates a new object of specified size, then copies the contents of *inlineHandle* onto the new object.

This Sheet Has Been Replaced

CONTROL

These instructions control the flow of execution for the GL processor.

call *executableArray* GLop.call

Invokes the procedure denoted by the handle *executableArray*.

return GLop.return

Returns executable flow back from the procedure currently executing. If the GL processor is not executing a procedure, this instruction finishes the current frame.

if *bool* $execArray_{true}$GLop.if

ifelse *bool* $execArray_{true}$ $execArray_{false}$GLop.ifelse

These operations conditionally execute the corresponding procedure according to the value of *bool* (0 means false, non zero means true).

loop number *execArray* GLop.loop

This Sheet Has Been Replaced

The *execArray* is executed *number* times. A negative number produces an infinite loop.

break GLop.break

continue GLop.continue

Breaks or continues within the currently executed loop.

switch *number* GLop.switch --

Switches to the program segment corresponding to the case value *number*. Case values are limited to the set *B={0..255}*

CONTEXT HANDLING

context *contextNumber* GLop.context --

Sets the active context to *contextNumber*. There are a fixed number of possible contexts.

contexDef*executableArray contextNumber*
GLop.contextDef

This Sheet Has Been Replaced

Defines context *contextNumber* as per procedure *executableArray*. Context procedures contain a switch statement that is used to map codes onto actions.

PROCEDURE DEFINITIONS

Procedures are simply executable arrays. A procedure handle is an array handle or an object handle.

SYSTEM SERVICES ACCESS

System services are accessed just like normal procedures, except that system procedure handles are part of a distinguished set.

MASK GENERATION

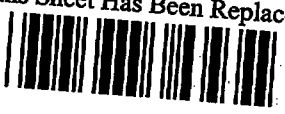
These instruction activate the primitive mask generators provided by GL.

Before activating any of the generators, a GL program must set the state information corresponding to the generator parameters.

PixelBit *bitmapHandle x y* GLop.PixelBit

Performs a pixelbit operation of *bitmapHandle* at position <*x,y*>.

This Sheet Has Been Replaced



Trapezoid*trapezoidHandle* GLop.Trapezoid

Activates the Trapezoid Mask Generator.

RLE GLop.RLE

Activates the Run Length encoded Mask Generator.

LINE GENERATION MASKS

These instructions activate the mask generators for line rendering.

These masks generators use GL state variables to implicitly define position, line width, etc.

ACCESSING THE INPUT STREAM

Members of this class of primitives access the input stream, to create objects in the stack and/or the array stack.

getc1 GLop.getc1 num (= b1)

getc2 GLop.getc2 num (= (b1 << 8) + b2)

getc3 GLop.getc3 num (= (((b1 << 8) +b2) <<

This Sheet Has Been Replaced

8) + b3)

getc4 GLop.getc4 num (= (((((b1 << 8) + b2) << 8) + b3) << 8) + b4)

Gets next 1 to 4 bytes (denoted b1, b2, b3 and b4) from the input stream and pushes them onto the stack as a scalar integer value.

BACKGROUND GENERATION

image a b c d e f image GLop.Image

Sets image as the background image array. The transformation [a b c d e f] denotes the mapping of raster pixels onto image array elements.

image 0 GLop.Image

Resets the background image array as empty.

gray *grayLevel* GLop.gray

Sets the background to gray, of level *grayLevel*.

This Sheet Has Been Replaced

bgtile *bitmapHandle* GLop.bgtile

Sets the background to be tiled by *bitmapHandle*.

bgmodify *offsetX offsetY rotation* GLop.bgtile

Modifies the background tile by shifting it *<offsetX, offsetY>* and rotating it by the amount specified in *rotation*.

STATE OPERATIONS

screen *objHandle* GLop.screen

Sets the dithering screen of *objHandle*.

clip *trapHandle* GLop.clip

Adds *trapHandle* to the clipping region.

clip 0 GLop.clip

Zeros the clipping region.

PRAGMATICS

This Sheet Has Been Replaced

The GL Interpreter ignores this class of instructions; pragmatics are for the use of other subsystems and/or tools (Debug module, Generation Libraries, etc.)

pragma GLop.pragma + {n} + b_1 + b_2 + ... + b_n

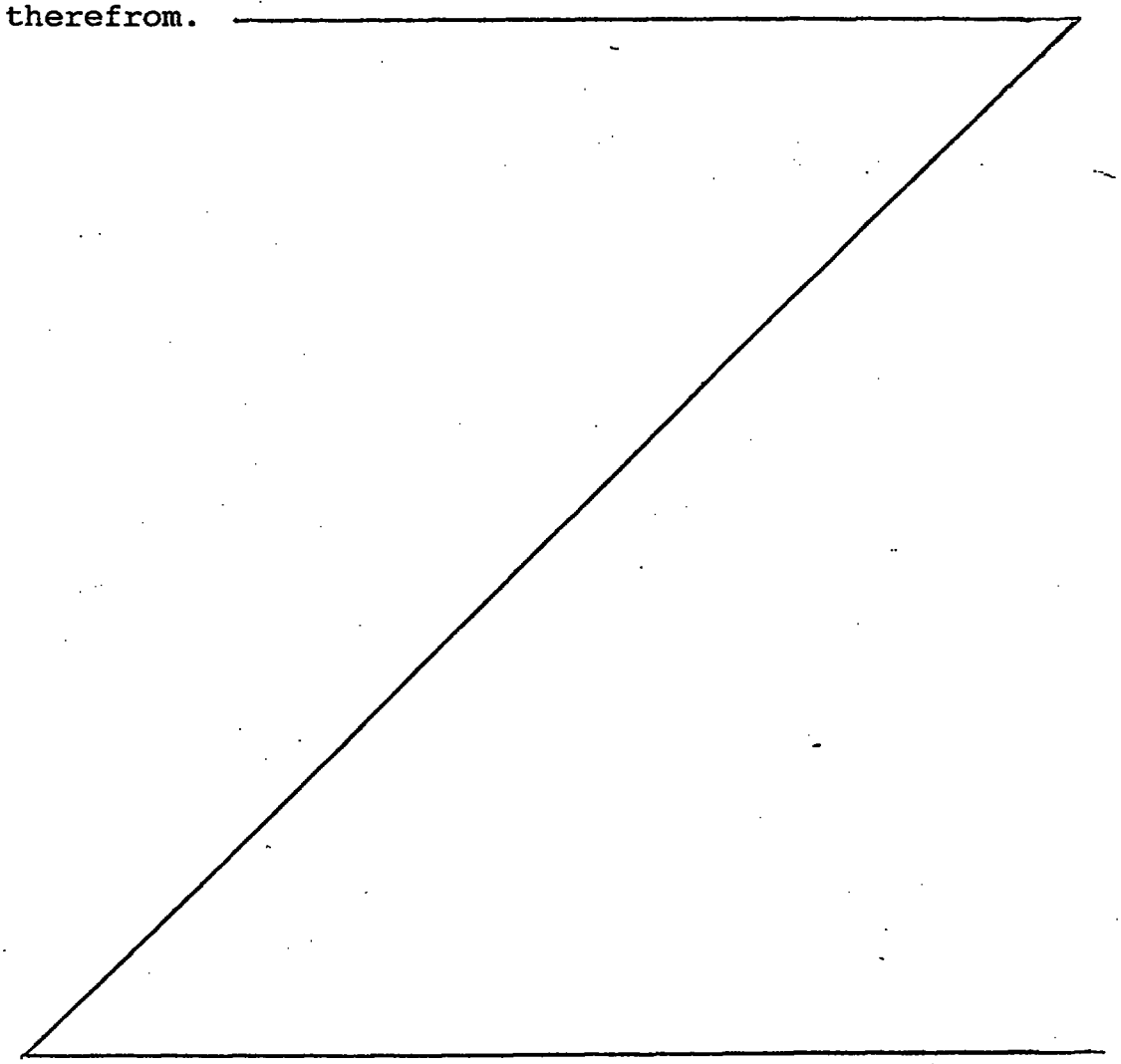
The GL Interpreter ignores the (n+2) bytes comprising this instruction. Use pragma as a comment, a synchronization tag for debugging purposes, a grouping tag, etc.

This Sheet Has Been Replaced

Appendix B

Aspects of the invention are defined in the appended claims.

The present application also discloses the apparatuses, devices and methods as described in the following numbered clauses, and the applicants reserve the right to claim any such matter disclosed, either in this application or in another application derived therefrom.



This Sheet Has Been Replaced

51. A print memory controller used for the printing apparatus according to claim 1, comprising:

means for monitoring a print memory so as to identify portions of said memory that correspond to physical pages to be printed;

a coder for storing coded representations of bands of image data to said memory;

means for rendering coded representations of bands of image data into a band raster portion of said memory in response to completion of coding of a page of image data;

means for shipping the raster image data in the band raster portion of said memory to a print engine at a uniform rate in response to completion of rasterization in the band raster portion;

wherein said coding means codes a succeeding page of image data at the page location monitored by said monitoring means.

52. A print memory controller according to clause 51, wherein said rendering means is adapted to render the coded image in situ at a degraded resolution.

53. A print memory controller according to clause 51, wherein in the case where said rendering means is rendering the page of coded information in situ, said

This Sheet Has Been Replaced

shipping means ships all previous pages before said rendering means renders the coded image.

54. A printing method comprising the steps of predicting resource usage before a printing operation commences and adaptably changing the printing operation in accordance with the prediction.

55. A printing method according to clause 54, wherein said step of adaptably changing the printing operation includes the step of rendering coded representations of print data into raster images and wherein said step of predicting resource usage is based on the coded representations.

56. A printing method according to clause 54, wherein said step of adaptably changing the printing operation is performed in accordance with the predictions so as to degrade image quality.

57. A printing method according to clause 54, further comprising the steps of:

predicting resources usage for a printing operation;

determining whether resources available for printing are sufficient for the usage predicted in said predicting step; and

This Sheet Has Been Replaced

changing the printing operation in the case where it is determined in said determining step that there are insufficient resources available for the printing operation.

58. A printing method according to clause 57, further comprising the step of forming a coded representation of an image to be formed during the printing operation, wherein said predicting step predicts resources utilized during rendering the coded representation into a full raster image.

59. A printing method according to clause 57, wherein the printing operation is conducted in bands for each page of print information, and wherein said predicting step predicts resource usage for each band.

60. A printing method according to clause 57, wherein said predicting step re-predicts resource usage based on a change in printing operation by said changing step, and wherein said changing step repeatedly changes the printing operation until sufficient resources are available as predicted by said predicting step.

This Sheet Has Been Replaced

61. A printing method according to clause 57, wherein said changing step changes the printing operation so as to degrade the image printed by said printer.

62. A printing method according to clause 61, wherein resolution is degraded.

63. A printing method according clause 61, wherein either of color resolution and number of halftone levels is degraded.

64. A printing method according to clause 57, further comprising the steps of scanningly printing print information at a uniform rate, and obtaining a bit map memory from which at least a portion of print information is obtained at a uniform rate.

65. A printing method according to clause 64, wherein said predicting step predicts at least one of computing power needed to form an image in said bit map memory, memory needed for the image, and bandwidth needed to transfer the image into the bit map memory.

66. A printing method according to clause 54, further comprising the steps of:

This Sheet Has Been Replaced

translating input image data into a coded representation of a print image;

storing the coded representation of the print image;

predicting resource usage in rendering the stored coded representation into a full raster image;

rendering the stored coded representation memory into a raster image in the case where resource usage is predicted to be adequate; and

scanningly printing the raster image at a uniform rate.

67. A printing method according to clause 66, further comprising the step of changing a print operation in the case where said predicting step predicts that there are insufficient resources to render the coded representation into a full raster image.

68. A printing method according to clause 67, wherein said changing step changes the coded representation into a representation that uses printer resources differently.

69. A printing method according to clause 67, wherein said changing step degrades print resolution.

This Sheet Has Been Replaced

70. A printing method according to clause 67, wherein said changing step degrades at least one of color resolution and number of halftone grey levels.

71. A printing method according to clause 66, wherein said predicting step predicts at least computing power needed to render the coded representation into a raster image and memory requirements for storing the coded representation.

72. A printing method according to clause 66, further comprising the step of inputting the image data that is translated in said translation step.

73. A printing method according to clause 72, wherein the image data is inputted in a page description language.

74. A printing method according to clause 66, wherein the first portion of said memory is adapted to store coded representations in bands, and wherein said second portion of said memory is adapted to store raster images of corresponding bands of information.

This Sheet Has Been Replaced

75. A printing method according to clause 74, wherein said translation step translates the input image data in bands.

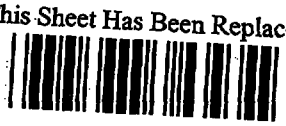
76. A printing method according to clause 75, wherein said predicting step predicts resource usage for each band of print information.

77. A printing method according to clause 76, wherein in the case where said determining step determines that there are insufficient resources, said rendering step renders a full raster image in situ in the first and the second portions of said memory.

78. A printing method according to clause 66, wherein said translating step translates input information into bands of coded raster representations, and wherein in the case where said determining step determines that there is insufficient computing power to render the band of information in a predetermined period of time corresponding to the uniform rate of said print engine, the band is pre-rasterized in said rendering step into a third portion of said memory.

79. A printing method according to clause 66, wherein in the case where said predicting step predicts that the

This Sheet Has Been Replaced



coded representation consumes more memory than a corresponding raster image, then said changing step changes the printing operation.

80. A printing method according to clause 66, wherein in the case where said determining step determines that there are insufficient resources available for one band of information, then that band of information is degraded and some other bands are not.

81. A printing method according to clause 66, wherein said representation comprises at least one of a bit map object, a run length object, a trapezoid object, a box object, and a fixed boundary encoding object.

82. A printing method according to clause 81, wherein said rendering step includes the step of applying an object against a background with an application function so as to achieve a rasterization of the representation.

83. A printing method according to clause 54, further comprising the steps of:

transforming image data of each of plural bands of an image into coded representations of the image;

storing the coded representations;

predicting resource usage in rendering the banded coded representations into a raster image, said predicting being based on the stored coded representations;

changing the coded representation in the case where it is predicted that resource usage exceeds available printer resources;

rendering a band of coded representation into a band of raster image data; and

printing the band of raster image data at a uniform rate.

84. A printing method according to clause 83, further comprising the step of accepting input image data in a page description language.

85. A printing method according to clause 84, wherein said transforming step transforms the page description language image data into coded representations that include at least one of a bit map object, a run length object, a trapezoid object, a box object and a fixed boundary encoding object.

86. A printing method according to clause 85, wherein said rendering step includes the step of applying an

This Sheet Has Been Replaced

object against a background with an application function so as to rasterize the image.

87. A printing method according to clause 85, wherein said predicting step is adapted to predict resource usage in accordance with the primitive used to form the coded representation.

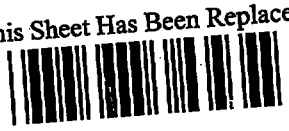
88. A printing method according to clause 83, wherein said predicting step is adapted to predict resource usage for each band of the image.

89. A printing method according to clause 83, wherein said predicting step is adapted to predict at least one of computing power, memory power, and bandwidth.

90. A printing method according to clause 89, wherein said changing step is adapted to change the coded representation so as to use more memory in the case where the computing power resource is exceeded.

91. A printing method according to clause 90, wherein said changing step is adapted to cause said rendering step to pre-rasterize a band of image data.

This Sheet Has Been Replaced



92. A printing method according to clause 90, wherein said changing step is adapted to degrade of the image.

93. A printing method according to clause 92, wherein said changing step is adapted to cause said rendering step to degrade at least the band whose resource usage exceeds that available.

94. A printing method according to clause 93, wherein said changing step is adapted to cause said rendering step to degrade the entire image and to render the degraded image in situ with said coded representation.

95. A printing method according to clause 93, wherein said degradation includes at least one of pixel density, color resolution, and grey level gradation.

96. A printing method according to clause 94, wherein said degradation includes at least one of pixel density, color resolution, and grey level gradation.

97. A printing method according to clause 94, wherein said rendering step is adapted to postpone rendering the degraded image until previous print operations have been completed.

This Sheet Has Been Replaced

98. A printing method according to clause 83, wherein said predicting step is adapted to predict at least one of computing power, memory, and bandwidth, and wherein said changing step is adapted to change the coded representation so as to use more computing power in the event that memory usage is exceeded.

99. A printing method according to clause 83, wherein said predicting step is adapted to predict at least one of the computer power, the memory usage and the bandwidth and wherein said changing step is adapted to use a coded representation that utilizes more bandwidth in the event that computing power or memory usage is exceeded.

100. A printing method according to clause 83, wherein said predicting step is adapted to predict at least one of the computer power, the memory usage and the bandwidth and wherein said changing step is adapted to change the coded representation so as to permit the coded representation to be rasterized in said rendering step within the resources available in the printer.

101. A printing method according to clause 100, wherein in the case where said changing step is unable to change to a coded representation that allows the rendering step to rasterize the image within the

This Sheet Has Been Replaced

resources of the printer, said changing step is adapted to degrade the image.

102. A printing method according to clause 101, wherein in the case where said changing step degrades the image, said changing step is adapted to cause said rendering step to rasterize previous pages of image data before degrading the image.

103. A printing method according to clause 102, wherein resolution includes at least one of pixel density, color resolution and grey level gradation.

104. A method for controlling print memory used for the printing method according to clause 54, further comprising the steps of:

monitoring a print memory so as to identify portions of said memory that correspond to physical pages to be printed;

storing coded representations of bands of image data in said memory;

rendering coded representations of bands of image data into a band raster portion of said memory in response to completion of coding of a page of image data;

shipping the raster image data in the band raster portion of said memory to a print engine at a uniform

This Sheet Has Been Replaced

rate in response to completion of rasterization in the band raster portion;

wherein said coding means codes a succeeding page of image data at the page location monitored in said monitoring step.

105. A method for controlling print memory according to clause 104, wherein said rendering step is adapted to render the coded image in situ at a degraded resolution.

106. A method for controlling print memory according to clause 104, wherein in the case where said rendering step is rendering the page of coded information in situ, said shipping step ships all previous pages before said rendering step renders the coded image.

107. A method for generating graphic language commands for a raster-based document comprising the steps of:

receiving print commands including device-independent print commands;

generating frame description commands for describing raster frames in the document based on the received print commands;

This Sheet Has Been Replaced

generating document layout commands for laying out the raster frames based on the received print commands; and

outputting the frame description commands and the document layout commands on respective first and second pipelined interfaces.

108. A method according to clause 107, wherein the received print commands comprise a page description language, and further comprising the step of translating the page description language into a graphics language.

109. A method according to clause 107, further comprising the step of rasterizing the frames in the frame descriptions.

110. A method according to clause 109, further comprising the step of applying the rasterized frames to a bit map raster in accordance with the document layout commands.

111. A method according to clause 110, wherein the document layout commands include commands for paper sheet handling, and further comprising the step of printing the bit map raster on paper sheets in accordance with the paper sheet handling commands.

This Sheet Has Been Replaced

112. A method according to clause 107, wherein the frame description commands are raster-based frame description commands that include mask generation commands and background generation commands, and further comprising the step of applying masks generated according to the mask generation commands to backgrounds generated according to the background generation commands onto a bit map memory.

113. A method according to clause 112, wherein the frame description commands are device-dependent.

114. A method according to clause 107, wherein said method is comprised by a printer driver executed in a host CPU.

115. A graphic language processing apparatus, comprising:

a mask generator for generating pixel objects;

a background generator for generating backgrounds against which pixel objects generated by said mask generator are applied;

an interpreter for interpreting graphic language commands and for activating one of said mask generator

This Sheet Has Been Replaced

and said background generator in accordance with the interpreted graphic language commands; and

a rasterizer for applying the pixel objects generated by said mask generator to the background generated by said background generator onto bit map raster memory.

116. An apparatus according to clause 115, wherein said mask generator includes plural mask generators, each for generating a different mask, and wherein said interpreter activates one of said plural mask generators in accordance with the interpreted graphic language.

117. An apparatus according to clause 116, wherein each of said plural mask generators is adapted to generate a compact representation of a primitive pixel object.

118. An apparatus according to clause 115, wherein said background generator includes plural background generators, each for generating a different background, and wherein said interpreter activates one of said plural background generators in accordance with the interpreted graphic language.

This Sheet Has Been Replaced

119. An apparatus according to clause 115, wherein said rasterizer is adapted to apply the pixel object to the background in accordance with a selectable application function.

120. An apparatus according to clause 115, further comprising a clipper for clipping the pixel object generated by said mask generator.

121. An apparatus according to clause 115, further comprising a dispatcher for sequentially selecting raster areas and for causing said mask generator and said background generator and said rasterizer to operate on the selected raster areas, respectively.

122. An apparatus according to clause 121, wherein the selected areas are banded partitions of the raster.

123. An apparatus according to clause 121, wherein the selected areas are arbitrary partitions of the raster.

124. An apparatus according to clause 115, wherein said mask generator and said background generator and said rasterizer are comprised by pipeline processing

This Sheet Has Been Replaced

units, and further comprising a coordinator for coordinating pipeline processing.

125. A printing method comprising the steps of:

inputting coded print data;

translating the coded print data into a first coded representation;

translating the first coded representation into a second coded representation which is different from the first coded representation; and

forming a raster image from the second coded representation.

126. A method according to clause 125, wherein the coded print data is page description language print data and wherein the first and second coded representations are intermediate in complexity to the page description language and the rasterized image.

127. A method according to clause 125, wherein the first and second coded representations include plural code primitives, and wherein all of the coded primitives of the second coded representation are the same as at least some of the coded primitives of the first coded representation.

This Sheet Has Been Replaced

128. A method according to clause 125, wherein the second coded representation is susceptible of accurate modelling such that usage of printer resources in said rasterizing step may be predicted accurately and wherein the first coded representation is not.

129. A method according to clause 128, further comprising the step of re-allocating printer resources in the case where printer resources are predicted to be exceeded.

130. An apparatus for generating graphic language commands for a raster-based document, comprising:

receiving means for receiving print commands including device-independent print commands;

first generating means for generating frame description commands for describing raster frames in the document based on the received print commands;

second generating means for generating document layout commands for laying out the raster frames based on the received print commands; and

outputting means for outputting the frame description commands and the document layout commands on respective first and second pipelined interfaces.

This Sheet Has Been Replaced

131. An apparatus according to clause 130, wherein the received print commands comprise a page description language, and further comprising translating means for translating the page description language into a graphics language.

132. An apparatus according to clause 130, further comprising rasterizing means for rasterizing the frames in the frame descriptions.

133. An apparatus according to clause 132, further comprising applying means for applying the rasterized frames to a bit map raster in accordance with the document layout commands.

134. An apparatus according to clause 133, wherein the document layout commands include commands for paper sheet handling, and further comprising printing means for printing the bit map raster on paper sheets in accordance with the paper sheet handling commands.

135. An apparatus according to clause 130, wherein the frame description commands are rasterbased frame description commands that include mask generation commands and background generation commands, and further comprising applying means for applying masks generated

This Sheet Has Been Replaced

according to the mask generation commands to backgrounds generated according to the background generation commands onto a bit map memory.

136. An apparatus according to clause 135, wherein the frame description commands are device-dependent.

137. An apparatus according to clause 130, wherein said apparatus is comprised by a printer driver executed in a host CPU.

138. A method for processing graphic language comprising the steps of:

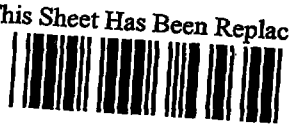
generating pixel objects;

generating backgrounds against which pixel objects generated in said pixel object generating step are applied;

interpreting graphic language commands and activating one of said generating steps in accordance with the interpreted graphic language commands; and

applying the pixel objects generated in said pixel object generating step to the background generated in said background generating step onto bit map raster memory.

This Sheet Has Been Replaced



139. A method according to clause 138, wherein said pixel object generating step includes the step of generating plural masks, each different from the other, and wherein said interpreting step activates one of said plural masks generated in said generating step in accordance with the interpreted graphic language.

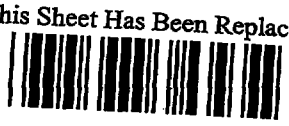
140. A method according to clause 139, wherein each of said plural masks generated in said generating step is adapted to generate a compact representation of a primitive pixel object.

141. A method according to clause 138, wherein said background generating step includes the step of generating plural backgrounds, each different from the other, and wherein said interpreting step activates one of said plural backgrounds generated in said generating step in accordance with the interpreted graphic language.

142. A method according to clause 138, wherein said applying step applies the pixel object to the background in accordance with a selectable application function.

143. A method according to clause 138, further comprising the step of clipping the pixel object generated in said pixel object generating step.

This Sheet Has Been Replaced



144. A method according to clause 138, further comprising the steps of sequentially selecting raster areas and causing said pixel object generating step, said background generating step and said applying step to operate on the selected raster areas, respectively.

145. A method according to clause 144, wherein the selected areas are banded partitions of the raster.

146. A method according to clause 144, wherein the selected areas are arbitrary partitions of the raster.

147. A method according to clause 138, wherein said pixel object generating step, said background generating step and said applying step are comprised by pipeline processing units, and further comprising the step of coordinating pipeline processing.

148. A printing apparatus comprising:

inputting means for inputting coded print data;
first translating means for translating the coded print data into a first coded representation;
second translating means for translating the first coded representation into a second coded representation

This Sheet Has Been Replaced

which is different from the first coded representation; and forming means for forming a raster image from the second coded representation.

149. An apparatus according to clause 148, wherein the coded print data is page description language print data and wherein the first and second coded representations are intermediate in complexity to the page description language and the rasterized image.

150. An apparatus according to clause 148, wherein the first and second coded representations include plural code primitives, and wherein all of the coded primitives of the second coded representation are the same as at least some of the coded primitives of the first coded representation.

151. An apparatus according to clause 148, wherein the second coded representation is susceptible of accurate modelling such that usage of printer resources in said rasterizing step may be predicted accurately and wherein the first coded representation is not.

152. An apparatus according to clause 151, further comprising re-allocating means for re-allocating printer

This Sheet Has Been Replaced

resources in the case where printer resources are predicted to be exceeded.

153. An apparatus for forming images comprising:

a communication interface for receiving print commands;

a generation module for generating graphic language commands including frame description commands for describing raster frames and document layout commands for laying out the raster frames, said generation module generating the frame description commands and the document layout commands based on the print commands;

a bit map memory for storing raster image information;

a graphic language processor for rendering raster frames in accordance with the frame description commands and for storing the rendered raster frames in said bit map memory in accordance with the document layout commands; and

image forming means for forming a pixel-based image in accordance with the raster image information in said bit map memory.

154. An apparatus according to clause 153, wherein said image forming means includes a driver for modulating a laser beam in accordance with the raster image

This Sheet Has Been Replaced

information and deflecting means for scanningly deflecting the modulated laser beam onto a photosensitive surface, said apparatus further comprising a system scheduler for coordinating said driver and said deflecting means.

155. An apparatus according to clause 153, wherein said bit map memory is comprised by a bit map raster pool which is partitionable into plural bit map memories, said apparatus further comprising a system scheduler for providing an empty bit map memory to said graphic language processor and for providing a filled bit map memory to said image forming means.

156. An apparatus according to clause 153, further comprising sheet handling means for mechanically feeding sheets to said image forming means in accordance with document layout commands.

157. An apparatus according to clause 156, further comprising sheet finishing means for processing sheets after image formation in accordance with document layout commands.

This Sheet Has Been Replaced

158. An apparatus according to clause 153, wherein said graphic language processor is comprised by a pipelined processor.

159. An apparatus according to clause 153, wherein said graphic language processor includes mask generation means for generating masks and background generating means for generating backgrounds against which the masks are applied onto the raster frame, said mask generating means and said background generating means being responsive to commands in a stream of graphic language commands.

160. An apparatus according to clause 159, further comprising an interpreter for parsing a stream of graphic language commands and for passing mask commands to said mask generation means and for passing background commands to said background generation means.

161. An apparatus according to clause 160, further comprising a rasterizer for rendering raster frames by applying masks generated by said mask generation means against backgrounds generated by said background generation means onto the raster frame in accordance with a selectable application function.

This Sheet Has Been Replaced

162. An apparatus according to clause 161, wherein said mask generation means and said background generation means and said rasterizer are comprised by pipelined processing units.

163. An apparatus according to clause 162, further comprising a coordinator for coordinating pipelining among the pipelined processing units.

164. An apparatus according to clause 161, further comprising a dispatcher for sequentially selecting raster areas and for causing said mask generator and said background generator and said rasterizer to operate on the selected raster areas, respectively.

165. An apparatus according to clause 164, wherein the selected areas are banded partitions of the raster.

166. An apparatus according to clause 164, wherein the selected areas are arbitrary partitions of the raster.

167. An apparatus according to clause 159, wherein the graphic language commands generated by said generation module include commands which define graphic

This Sheet Has Been Replaced

objects and commands which define how to apply the objects to a raster frame.

168. An apparatus according to clause 153, wherein said graphic language processor includes means for translating the graphic language commands into a coded command sequence which is different from the graphic language commands.

169. An apparatus according to clause 168, wherein the graphic language commands and the coded command sequence include plural command primitives, and wherein all of the command primitives of the coded command sequence are the same as at least some of the command primitives of the graphic language commands.

170. An apparatus according to clause 169, wherein the coded command sequence is susceptible of accurate modelling such that usage of image formation resources may be predicted accurately and wherein the graphic language commands are not.

171. An apparatus according to clause 170, wherein image formation resources are re-allocated in the case where resources are predicted to be exceeded.

172. A method of forming an image comprising the steps of:

receiving print commands;

generating graphic language commands including frame description commands for describing raster frames and document layout commands for laying out the raster frames, said generation step generating the frame description commands and the document layout commands based on the print commands;

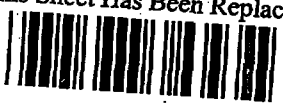
storing raster image information in a bit map memory;

rendering raster frames in accordance with the frame description commands and storing the rendered raster frames in said bit map memory in accordance with the document layout commands; and

forming a pixel-based image in accordance with the raster image information in said bit map memory.

173. A method according to clause 172, wherein said forming step includes modulating a laser beam in accordance with the raster image information and scanningly deflecting the modulated laser beam onto a photosensitive surface, said method further comprising the step of coordinating said modulating step and said deflecting step.

This Sheet Has Been Replaced



174. A method according to clause 172, wherein said bit map memory is comprised by a bit map raster pool which is partitionable into plural bit map memories, said method further comprising the steps of providing an empty bit map memory to said graphic language processor and providing a filled bit map memory to said image forming step.

175. A method according to clause 172, further comprising the step of mechanically feeding sheets in accordance with document layout commands.

176. A method according to clause 175, further comprising the step of processing sheets after image formation in accordance with document layout commands.

177. A method according to clause 172, wherein said rendering step is comprised by a pipelined processor.

178. A method according to clause 172, wherein said rendering step includes the steps of generating masks and generating backgrounds against which the masks are applied onto the raster frame, said mask generating step and said background generating step being responsive to commands in a stream of graphic language commands.

This Sheet Has Been Replaced

179. A method according to clause 178, further comprising the step of parsing a stream of graphic language commands and passing mask commands to said mask generating step and passing background commands to said background generating step.

180. A method according to clause 179, further comprising the step of rendering raster frames by applying masks generated in said mask generating step against backgrounds generated in said background generating step onto the raster frame in accordance with a selectable application function.

181. A method according to clause 180, wherein said mask generating step, said background generating step and said raster frame rendering step are comprised by pipelined processing units.

182. A method according to clause 181, further comprising the step of coordinating pipelining among the pipelined processing units.

183. A method according to clause 180, further comprising the steps of sequentially selecting raster areas and causing said mask generating step, said background generating step and said raster frame

This Sheet Has Been Replaced

rendering step to operate on the selected raster areas, respectively.

184. A method according to clause 183, wherein the selected areas are banded partitions of the raster.

185. A method according to clause 183, wherein the selected areas are arbitrary partitions of the raster.

186. A method according to clause 178, wherein the graphic language commands generated in said generating step include commands which define graphic objects and commands which define how to apply the objects to a raster frame.

187. A method according to clause 172, wherein said rendering step includes the step of translating the graphic language commands into a coded command sequence which is different from the graphic language commands.

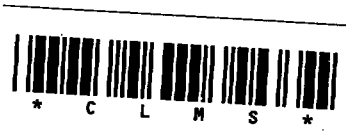
188. A method according to clause 187, wherein the rendering step and the coded command sequence include plural command primitives, and wherein all of the command primitives of the coded command sequence are the same as at least some of the command primitives of the graphic language commands.

This Sheet Has Been Replaced

189. A method according to clause 188, wherein the coded command sequence is susceptible of accurate modelling such that usage of image formation resources may be predicted accurately and wherein the graphic language commands are not.

190. An apparatus according to clause 189, wherein image formation resources are re-allocated in the case where resources are predicted to be exceeded.

This Sheet Has Been Replaced



WHAT IS CLAIMED IS:

1. A printing apparatus which adaptably changes a printing operation in accordance with predictions of resource usage made before the printing operation commences.

2. A printing apparatus according to Claim 1, wherein said printing operation includes rendering coded representations of print data into raster images and wherein said predictions are made based on the coded representations.

3. A printing apparatus according to Claim 1, wherein said printing operation is adaptably changed in accordance with the predictions so as to degrade image quality.

4. A printing apparatus according to claim 1, further comprising:

means for predicting resource usage for a printing operation;

means for determining whether resources available in the printing apparatus are sufficient for the usage predicted by said predicting means; and

means for changing the printing operation in the case where it is determined by said determining means

This Sheet Has Been Replaced

* C L M S *

that there are insufficient resources available for the printing operation.

5. A printing apparatus according to Claim 4, further comprising means for forming a coded representation of an image to be formed during the printing operation, wherein said predicting means predicts resources utilized during rendering the coded representation into a full raster image.

6. A printing apparatus according to Claim 4, wherein the printing operation is conducted in bands for each page of print information, and wherein said predicting means predicts resource usage for each band.

7. A printing apparatus according to Claim 4, wherein said predicting means re-predicts resource usage based on a change in printing operation by said changing means, and wherein said changing means repeatedly changes the printing operation until sufficient resources are available as predicted by said predicting means.

8. A printing apparatus according to Claim 4, wherein said changing means changes the printing operation so as to degrade the image printed by said printer.

This Sheet Has Been Replaced

* C L M S *

9. A printing apparatus according to Claim 8, wherein resolution is degraded.

10. A printing apparatus according to Claim 8, wherein either of color resolution and number of halftone levels is degraded.

11. A printing apparatus according to Claim 4, further comprising a print engine for scanningly printing print information at a uniform rate, and a bit map memory from which at least a portion of print information is obtained at a uniform rate.

12. A printing apparatus according to Claim 11, wherein said predicting means predicts at least one of computing power needed to form an image in said bit map memory, memory needed for the image, and bandwidth needed to transfer the image into the bit map memory.

13. A printing apparatus according to claim 1, further comprising:

a print engine for scanningly forming a print image at a uniform rate;

a memory adaptably partitioned into at least first and second portions, the first portion for storing coded

This Sheet Has Been Replaced

representations of the print image and the second portion for storing at least a portion of a raster image;

translation means for translating input image data into a coded representation of the print image and for storing the coded representation in the first portion of the memory;

predicting means for predicting resource usage in rendering the coded representation stored in the first portion of said memory to a full raster image; and

rendering means for rendering the coded representation stored in the first portion of said memory into a raster image in the second portion of said memory.

14. A printing apparatus according to Claim 13, further comprising changing means for changing a print operation in the case where said predicting means predicts that there are insufficient resources to render the coded representation into a full raster image.

15. A printing apparatus according to Claim 14, wherein said changing means changes the coded representation into a representation that uses printer resources differently.

16. A printing apparatus according to Claim 14, wherein said changing means degrades print resolution.

This Sheet Has Been Replaced

17. A printing apparatus according to Claim 14, wherein said changing means degrades at least one of color resolution and number of halftone grey levels.

18. A printing apparatus according to Claim 13, wherein said predicting means predicts at least computing power needed to render the coded representation into a raster image and memory requirements for storing the coded representation.

19. A printing apparatus according to Claim 13, further comprising input means for inputting the image data that is translated by said translation means.

20. A printing apparatus according to Claim 19, wherein the input data is inputted in a page description language.

21. A printing apparatus according to Claim 13, wherein the first portion of said memory is adapted to store coded representations in bands, and wherein said second portion of said memory is adapted to store raster images of corresponding bands of information.

This Sheet Has Been Replaced

* C L M S *

22. A printing apparatus according to Claim 21, wherein said translation means translates the input image data in bands.

23. A printing apparatus according to Claim 22, wherein said predicting means predicts resource usage for each band of print information.

24. A printing apparatus according to Claim 23, wherein in the case where said determining means determines that there are insufficient resources, said rendering means renders a full raster image in situ in the first and the second portions of said memory.

25. A printing apparatus according to Claim 13, wherein said translating means translates input information into bands of coded raster representations, and wherein in the case where said determining means determines that there is insufficient computing power to render the band of information in a predetermined period of time corresponding to the uniform rate of said print engine, the band is pre-rasterized by said rendering means into a third portion of said memory.

26. A printer apparatus according to Claim 13, wherein in the case where said predicting means predicts

This Sheet Has Been Replaced

that the coded representation consumes more memory than a corresponding raster image, then said changing means changes the printing operation.

27. A printing apparatus according to Claim 13, wherein in the case where said determining means determines that there are insufficient resources available for one band of information, then that band of information is degraded and some other bands are not.

28. A printing apparatus according to Claim 13, wherein said representation comprises at least one of a bit map object, a run length object, a trapezoid object, a box object, and a fixed boundary encoding object.

29. A printing apparatus according to Claim 28, wherein said rendering means includes means for applying an object against a background with an application function so as to achieve a rasterization of the representation.

30. A printer according to claim 1, further comprising:

a memory adaptably portionable into first and second portions, the first portion for storing a coded

This Sheet Has Been Replaced

* C L M S *

representation of image data and a second portion for storing at least one band of rasterized image data;

means for transforming the image data of each of plural bands of the image into coded representations of the image, and for storing the coded representations in the first portion of said memory;

predicting means for predicting resource usage in rendering the banded coded representations into a raster image, said predicting means predicting based on the coded representations stored in the first portion of said memory;

means for changing the coded representation in the first portion of said memory in the case where said predicting means predicts that resource usage exceeds available printer resources;

rendering means for rendering a band of coded representation stored in the first portion of said memory into a band of raster image data and for storing the band of raster image data in the second portion of said memory; and

a print engine for printing raster image data stored in at least the second portion of said memory, said print engine printing the band of raster image data at a uniform rate.

This Sheet Has Been Replaced

31. A printer according to Claim 30, further comprising input means for accepting input image data in a page description language.

32. A printer according to Claim 31, wherein said transforming means transforms the page description language image data into coded representations that include at least one of a bit map object, a run length object, a trapezoid object, a box object and a fixed boundary encoding object.

33. A printer according to Claim 32, wherein said rendering means includes means for applying an object against a background with an application function so as to rasterize the image.

34. A printer according to Claim 32, wherein said predicting means is adapted to predict resource usage in accordance with the primitive used to form the coded representation.

35. A printer according to Claim 30, wherein said predicting means is adapted to predict resource usage for each band of the image.

This Sheet Has Been Replaced

* C L M S *

36. A printer according to Claim 30, wherein said predicting means is adapted to predict at least one of computing power, memory power, and bandwidth.

37. A printer according to Claim 36, wherein said changing means is adapted to change the coded representation so as to use more memory in the case where the computing power resource is exceeded.

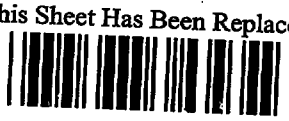
38. A printer according to Claim 37, wherein said changing means is adapted to cause said rendering means to pre-rasterize a band of image data.

39. A printer according to Claim 37, wherein said changing means is adapted to degrade of the image.

40. A printer according to Claim 39, wherein said changing means is adapted to cause said rendering means to degrade at least the band whose resource usage exceeds that available.

41. A printer according to Claim 40, wherein said changing means is adapted to cause said rendering means to degrade the entire image and to render the degraded image in situ with said coded representation.

This Sheet Has Been Replaced



* C L M S *

42. A printer according to Claim 40, wherein said degradation includes at least one of pixel density, color resolution, and grey level gradation.

43. A printer according to Claim 41, wherein said degradation includes at least one of pixel density, color resolution, and grey level gradation.

44. A printer according to Claim 41, wherein said rendering means is adapted to postpone rendering the degraded image until previous print operations have been completed.

45. A printer according to Claim 30, wherein said predicting means is adapted to predict at least one of computing power, memory, and bandwidth, and wherein said changing means is adapted to change the coded representation so as to use more computing power in the event that memory usage is exceeded.

46. A printer according to Claim 30, wherein said predicting means is adapted to predict at least one of the computer power, the memory usage and the bandwidth and wherein said changing means is adapted to use a coded representation that utilizes more bandwidth in the event that computing power or memory usage is exceeded.

This Sheet Has Been Replaced

* C L M S *

47. A printer according to Claim 30, wherein said predicting means is adapted to predict at least one of the computer power, the memory usage and the bandwidth and wherein said changing means is adapted to change the coded representation so as to permit the coded representation to be rasterized by said rendering means within the resources available in the printer.

48. A printer according to Claim 47, wherein in the case where said changing means is unable to change to a coded representation that allows the rendering means to rasterize the image within the resources of the printer, said changing means is adapted to degrade the image.

49. A printer according to Claim 48, wherein in the case where said changing means degrades the image, said changing means is adapted to cause said rendering means to rasterize previous pages of image data before degrading the image.

50. A printer according to Claim 49, wherein resolution includes at least one of pixel density, color resolution and grey level gradation.

This Sheet Has Been Replaced

* C L M S *

51. A method for generating graphic language commands for a raster-based document comprising the steps of:

receiving print commands including device-independent print commands;

generating frame description commands for describing raster frames in the document based on the received print commands;

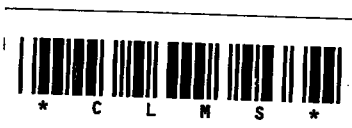
generating document layout commands for laying out the raster frames based on the received print commands; and

outputting the frame description commands and the document layout commands on respective first and second pipelined interfaces.

52. A method according to Claim 51, wherein the received print commands comprise a page description language, and further comprising the step of translating the page description language into a graphics language.

53. A method according to Claim 51, further comprising the step of rasterizing the frames in the frame descriptions.

This Sheet Has Been Replaced



54. A method according to Claim 53, further comprising the step of applying the rasterized frames to a bit map raster in accordance with the document layout commands.

55. A method according to Claim 54, wherein the document layout commands include commands for paper sheet handling, and further comprising the step of printing the bit map raster on paper sheets in accordance with the paper sheet handling commands.

56. A method according to Claim 51, wherein the frame description commands are raster-based frame description commands that include mask generation commands and background generation commands, and further comprising the step of applying masks generated according to the mask generation commands to backgrounds generated according to the background generation commands onto a bit map memory.

57. A method according to Claim 56, wherein the frame description commands are device-dependent.

58. A method according to Claim 57, wherein said method is comprised by a printer driver executed in a host CPU.

This Sheet Has Been Replaced

ABSTRACT OF THE DISCLOSURE

Method and Apparatus for Printing
According to a Graphic Language

A method and apparatus for generating graphic language document description commands based on print commands that include device-independent print commands. The printing apparatus predicts resource usage for a printing operation, determines whether resources available in the printing apparatus are sufficient for the predicted usage, and changes the printing operation in the case where it is determined that there are insufficient resources available for the currently-specified printing operation so that it is possible to print the image within existing resource constraints.

This Sheet Has Been Replaced

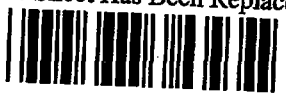
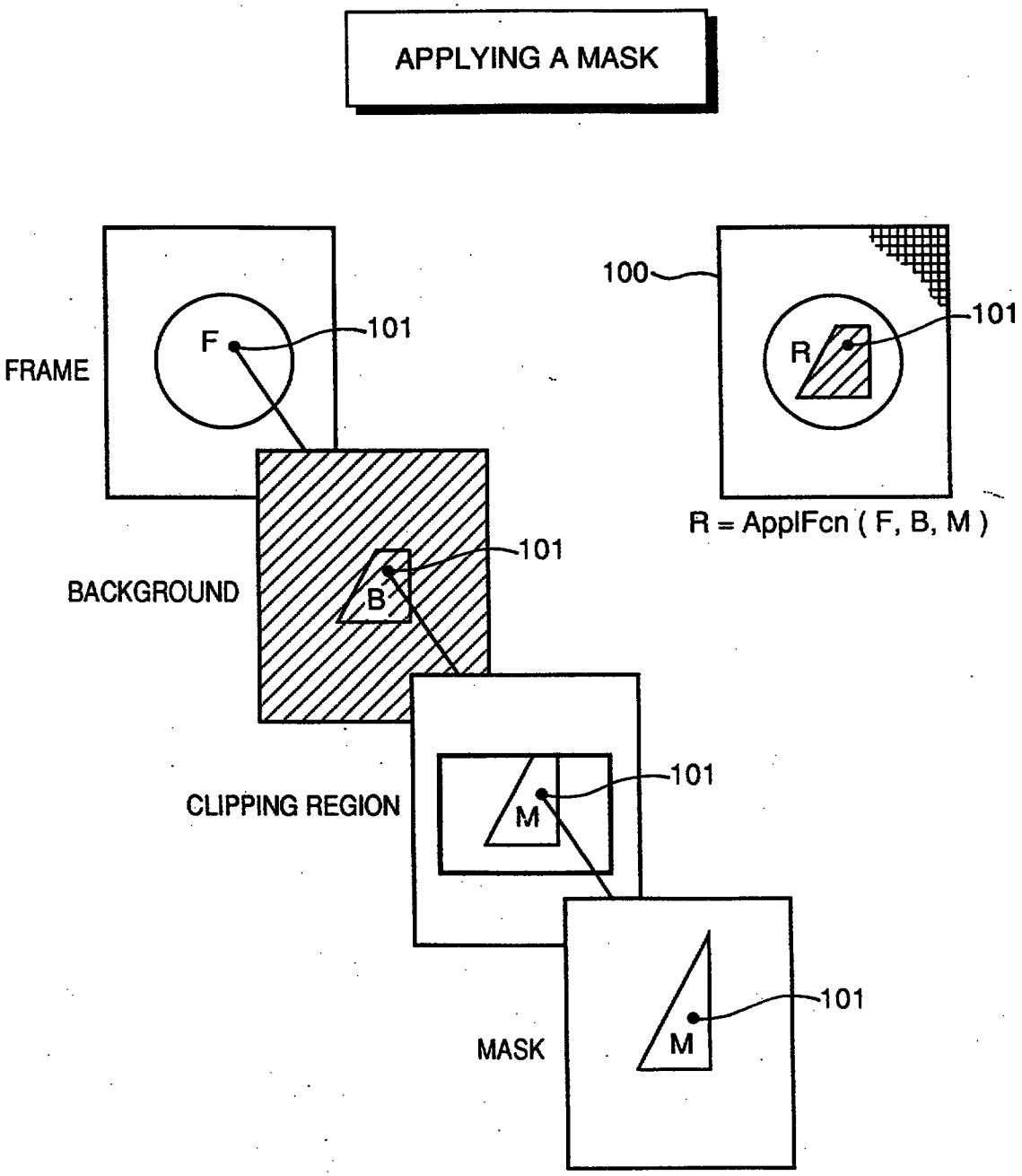
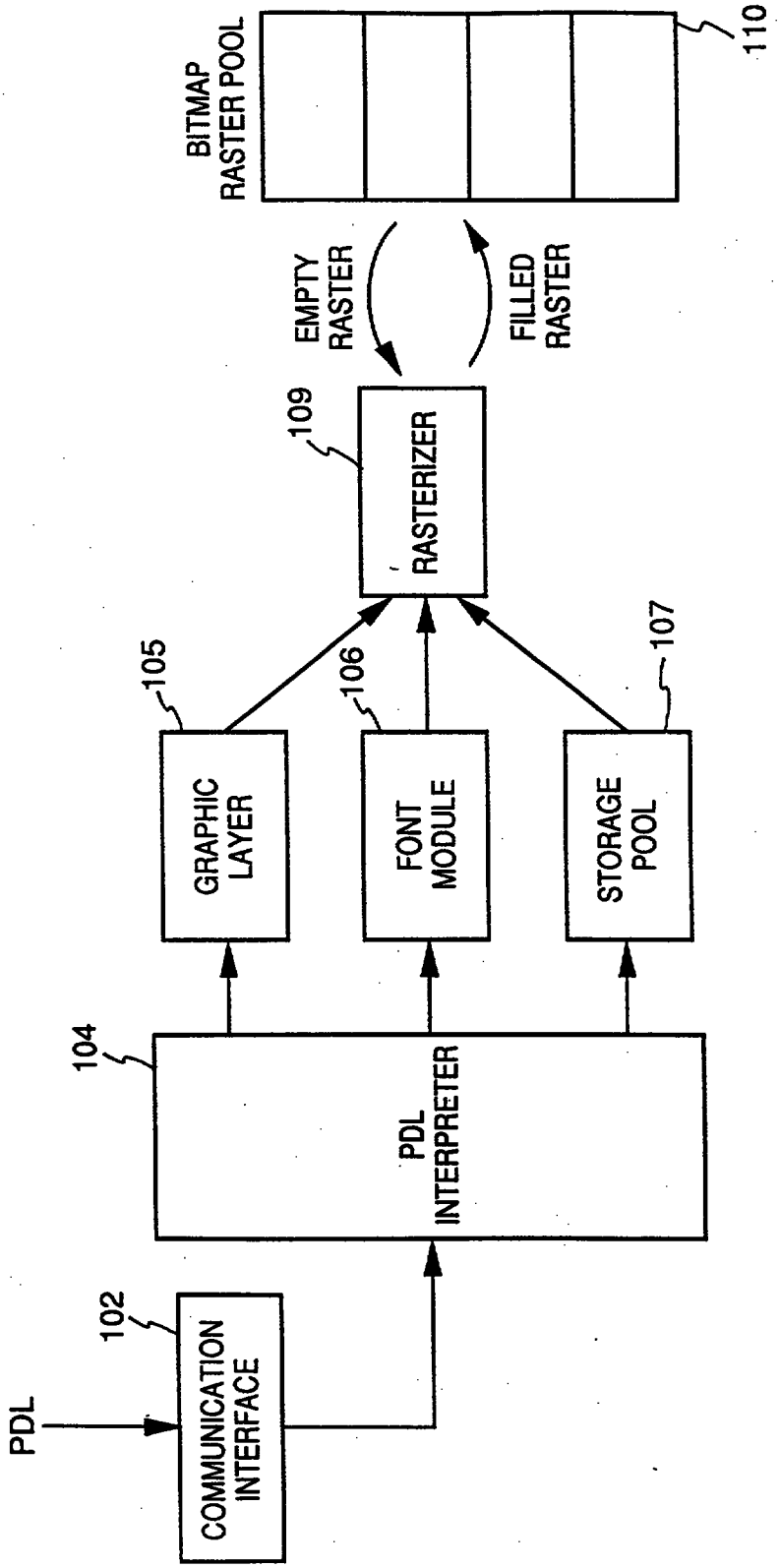


FIG. 1



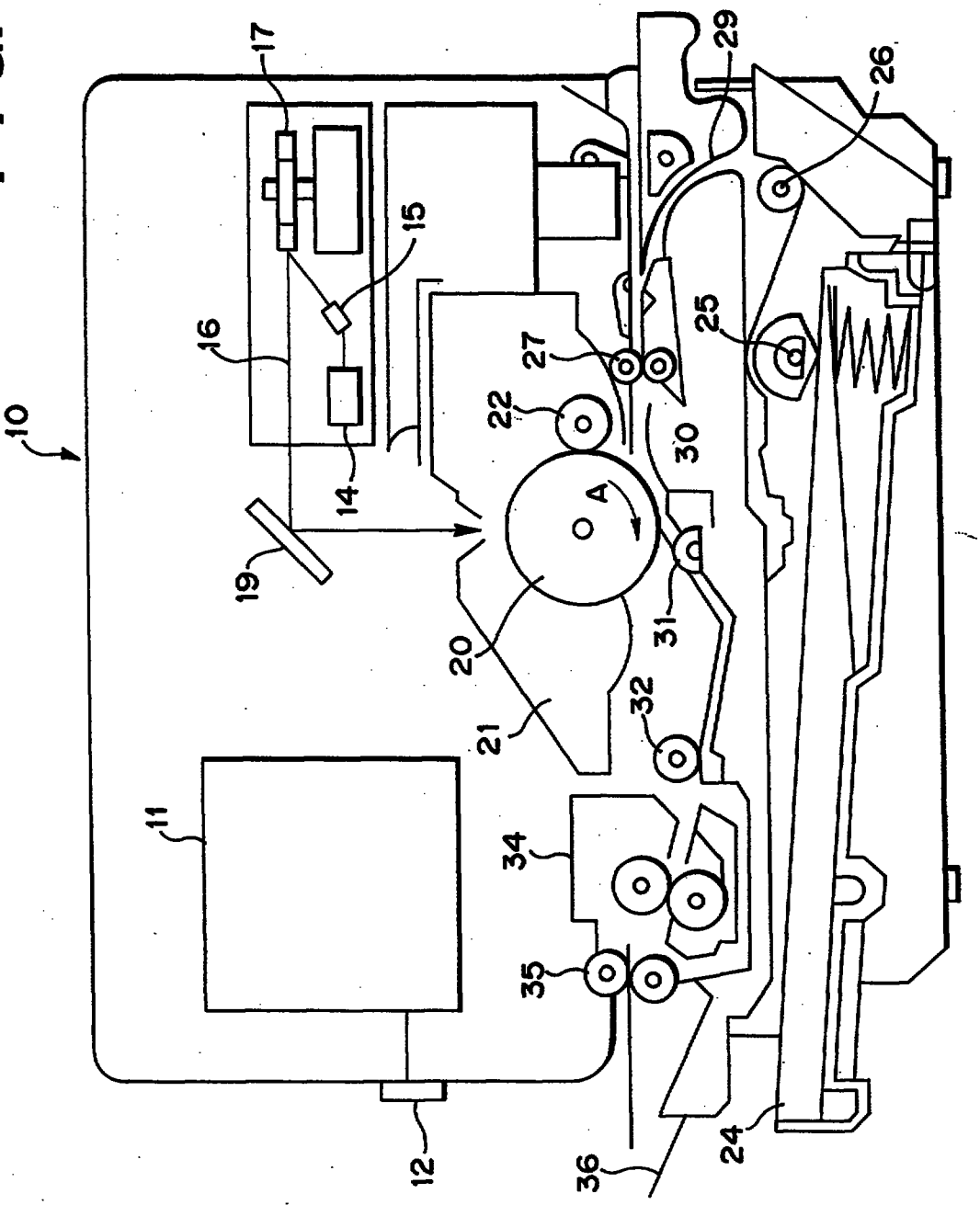
This Sheet Has Been Replaced

FIG. 2



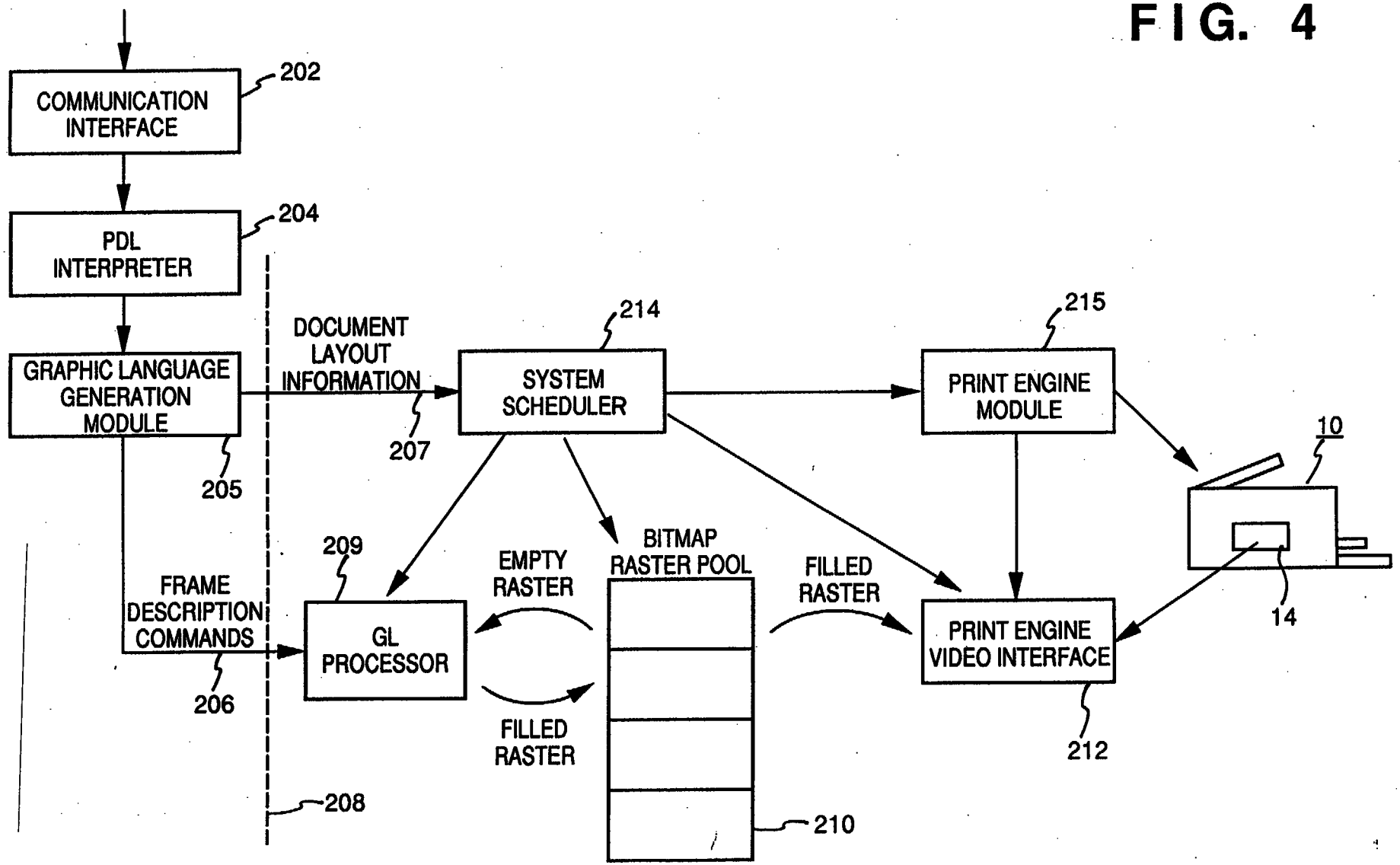
This Sheet Has Been Replaced

FIG. 3



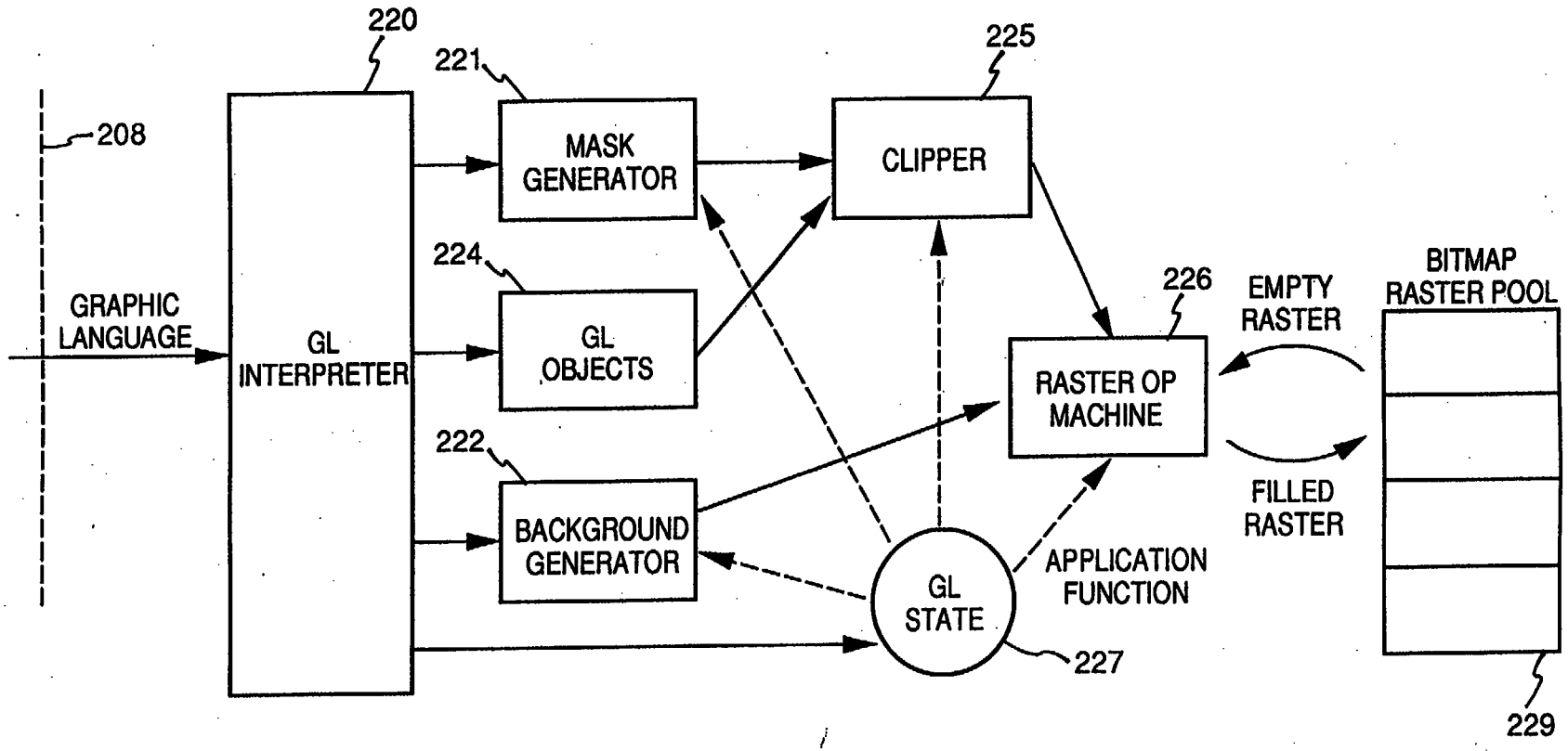
This Sheet Has Been Replaced

FIG. 4



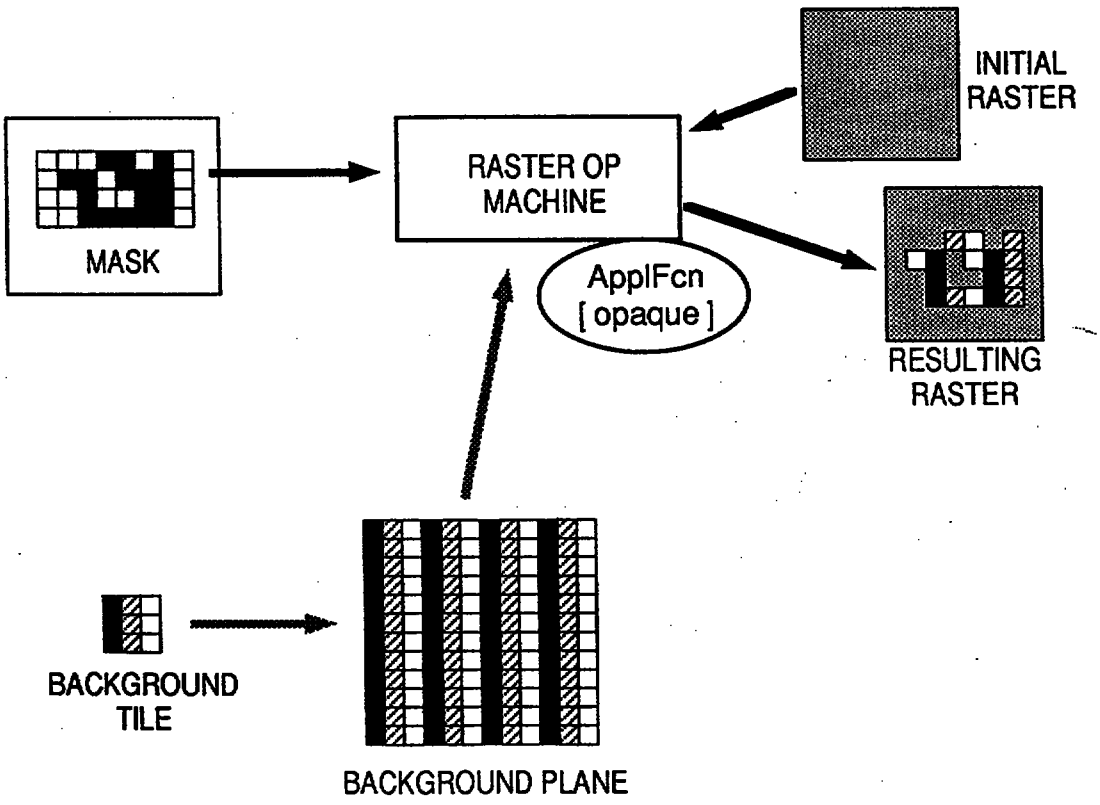
This Sheet Has Been Replaced

FIG. 5



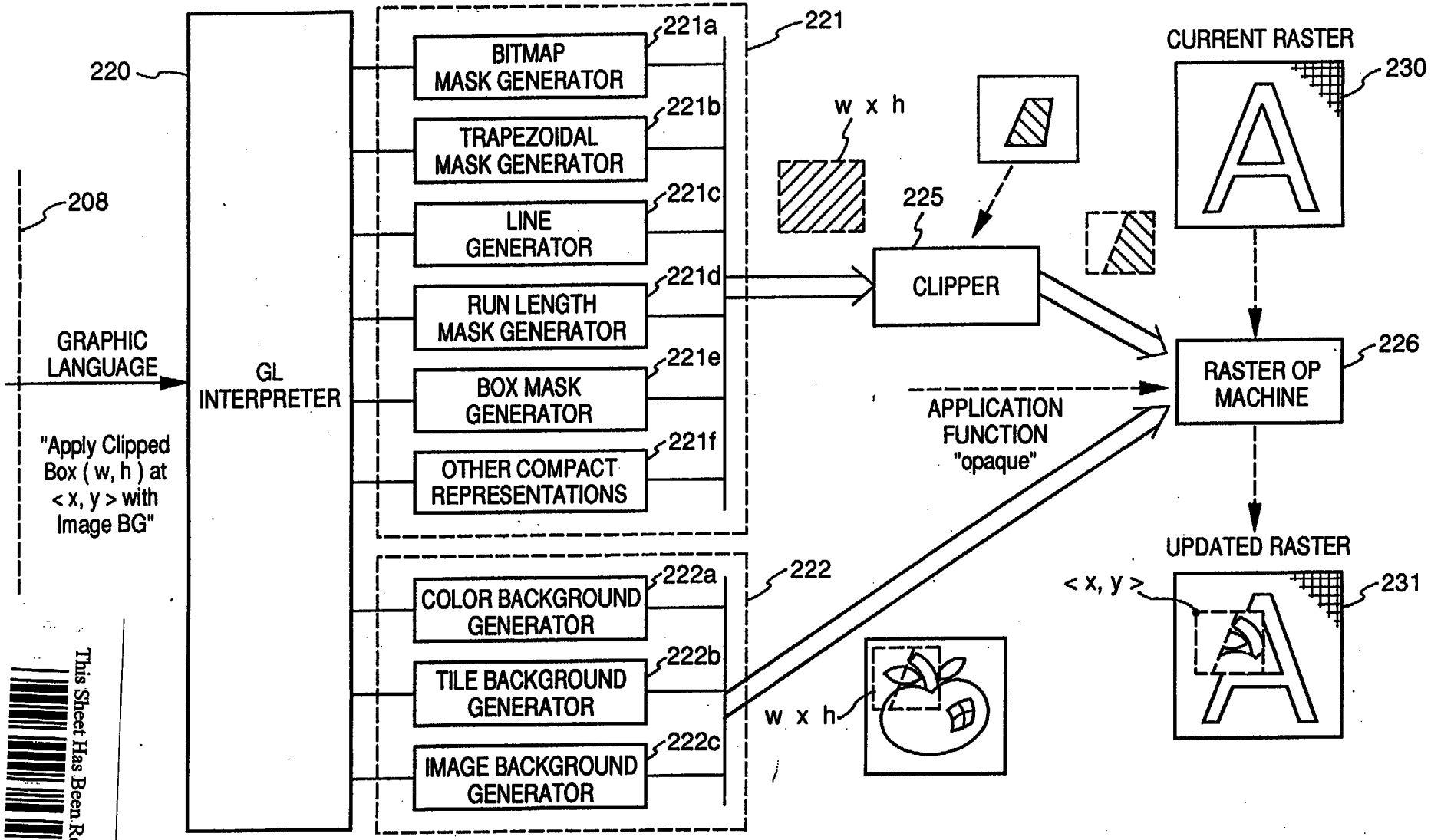
This Sheet Has Been Replaced

FIG. 6



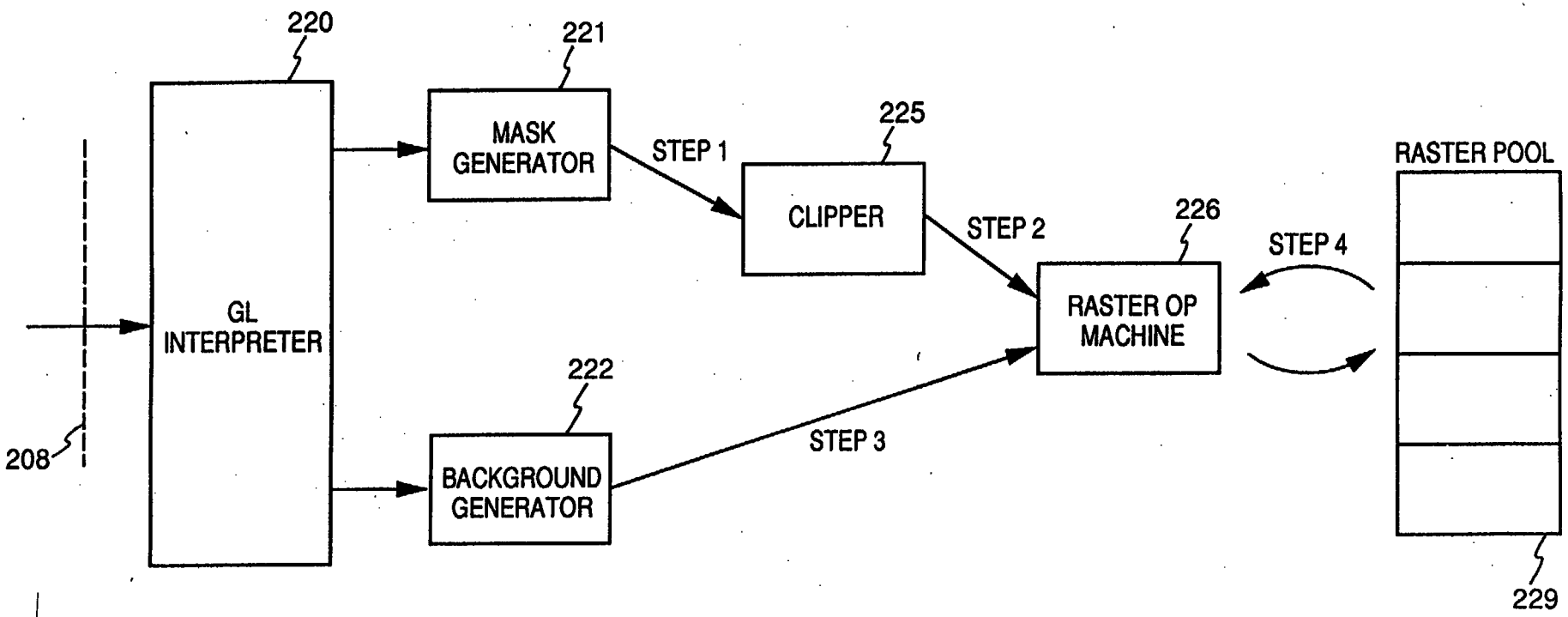
This Sheet Has Been Replaced

FIG. 7



This Sheet Has Been Replaced

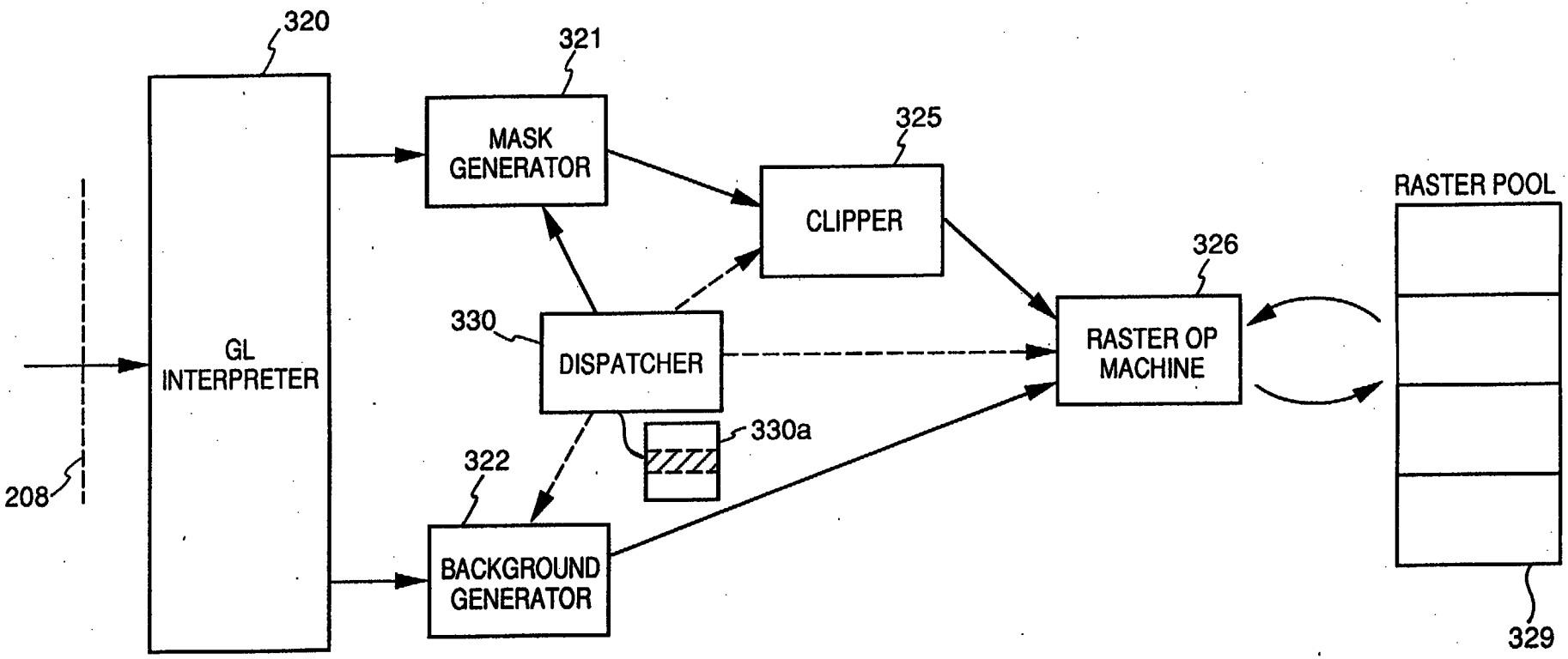
FIG. 8



This Sheet Has Been Replaced

This Sheet Has Been Replaced

FIG. 9



This Sheet Has Been Replaced

FIG. 10

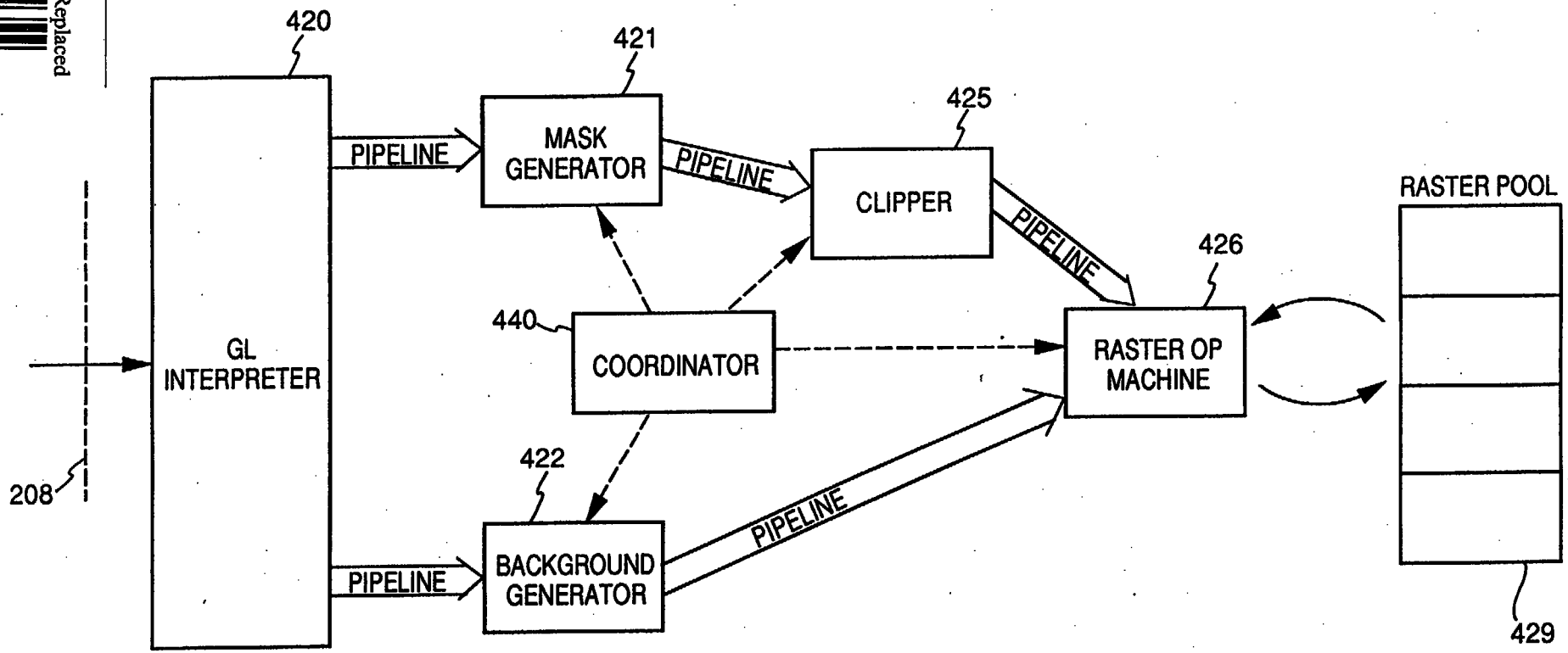


FIG. 11

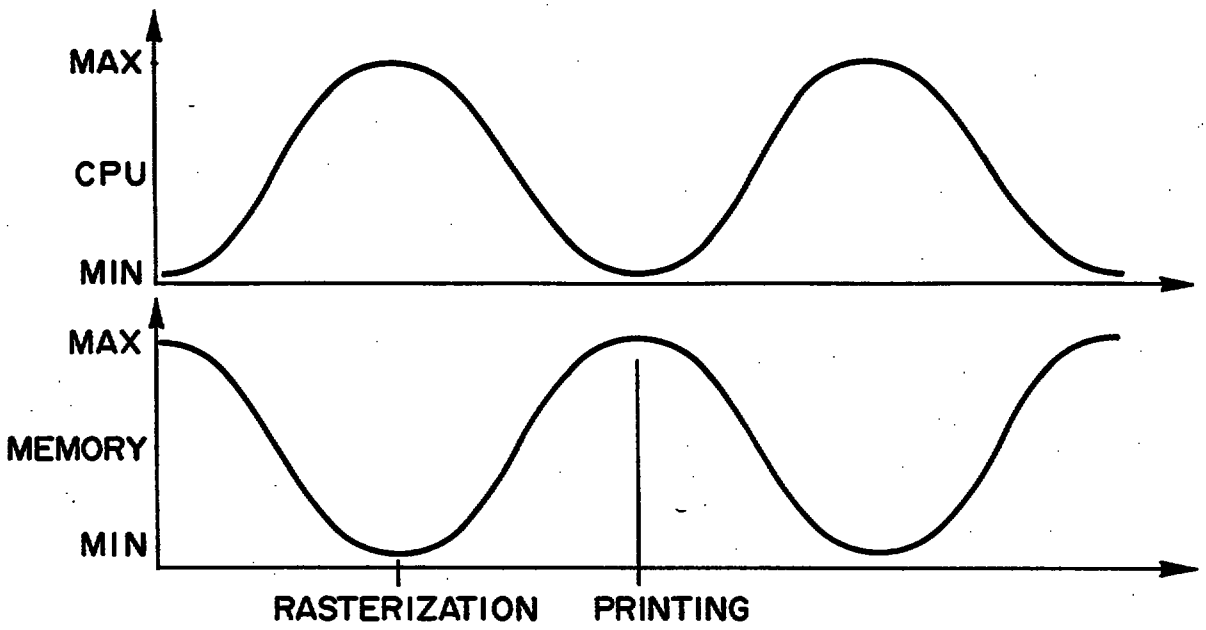
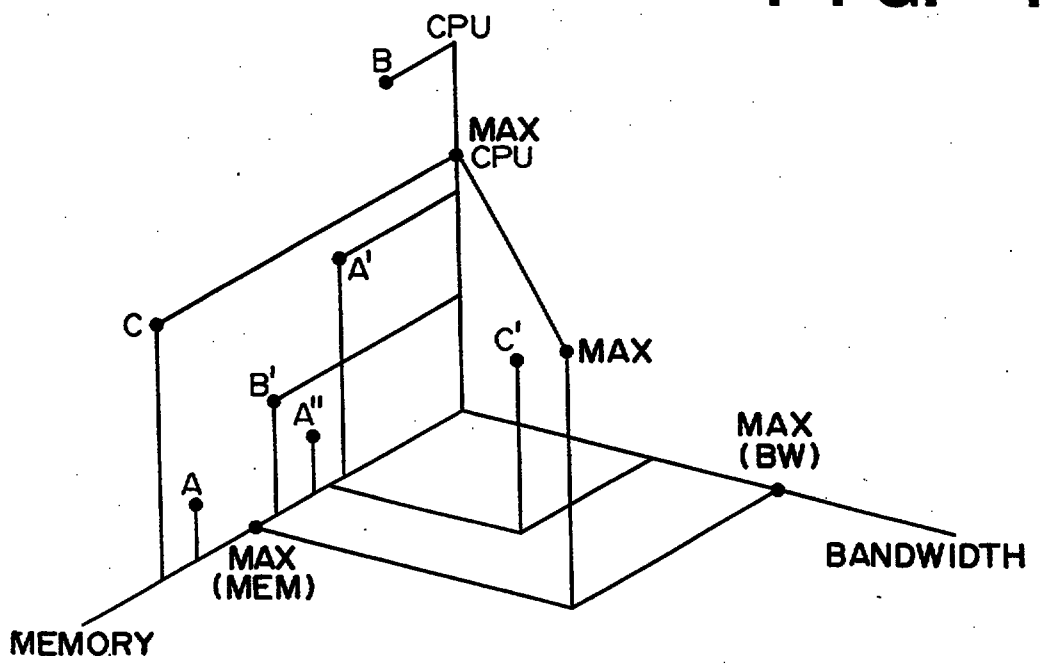
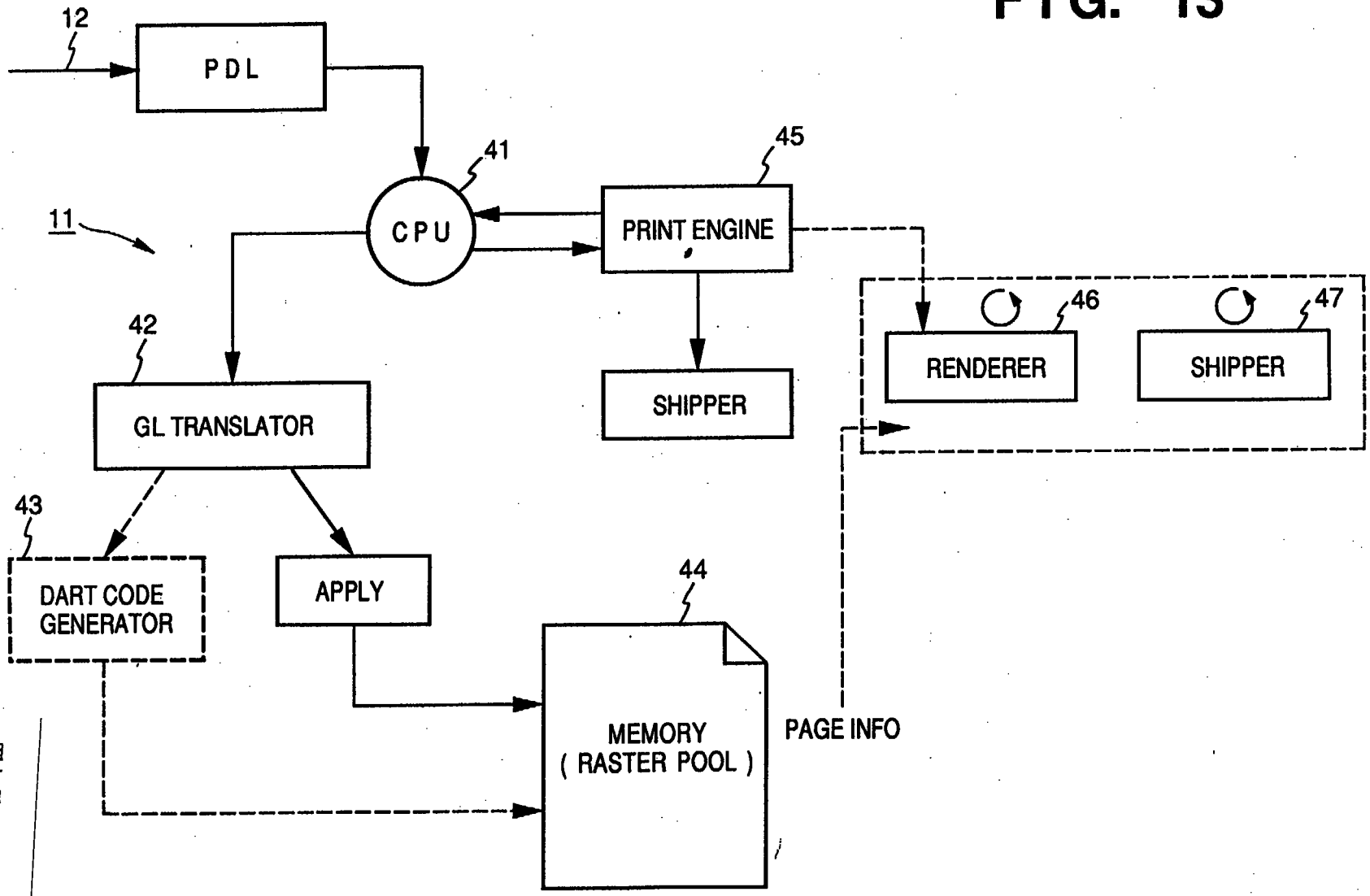


FIG. 12



This Sheet Has Been Replaced

FIG. 13



This Sheet Has Been Replaced

This Sheet Has Been Replaced

FIG. 14

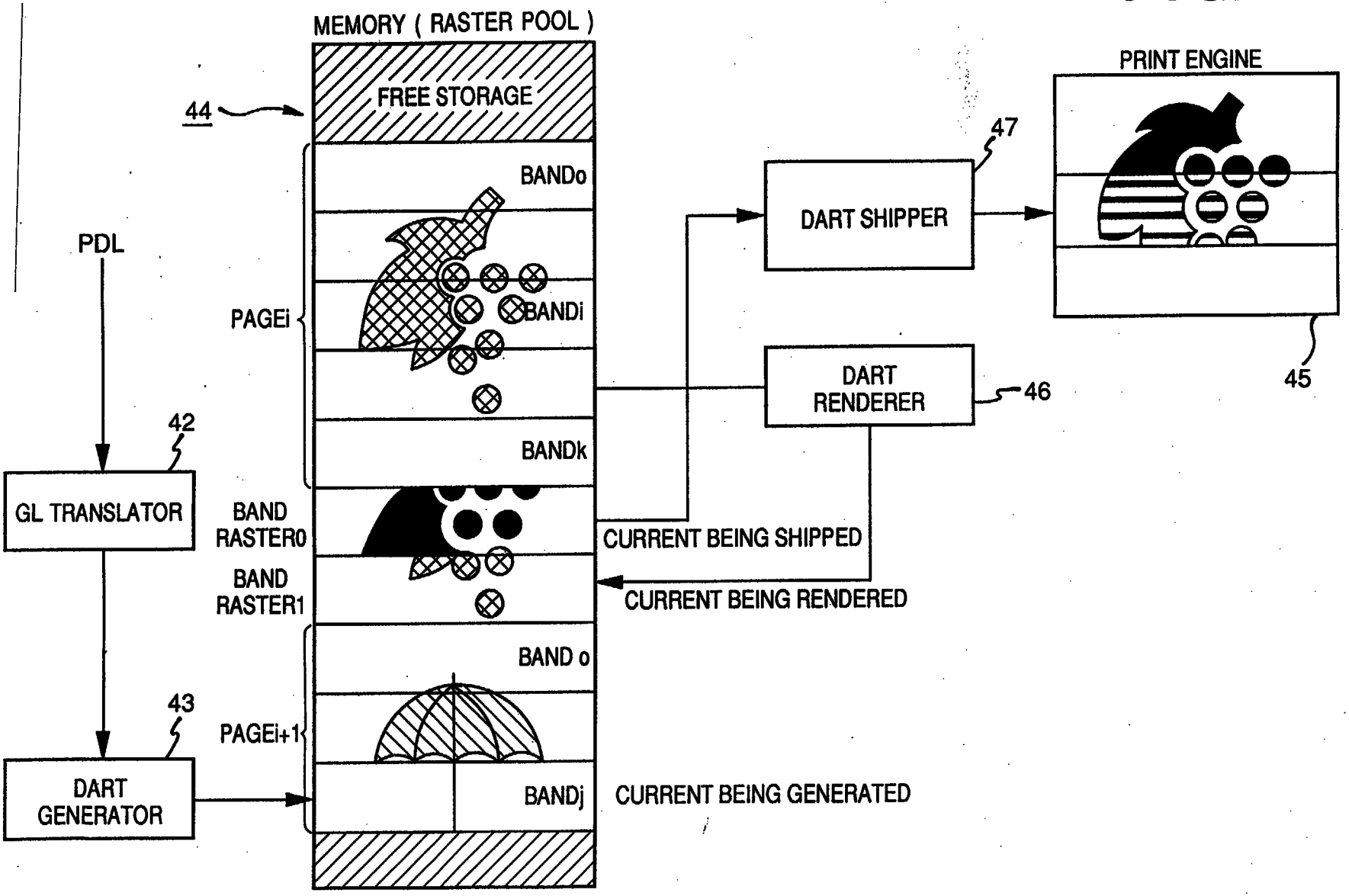
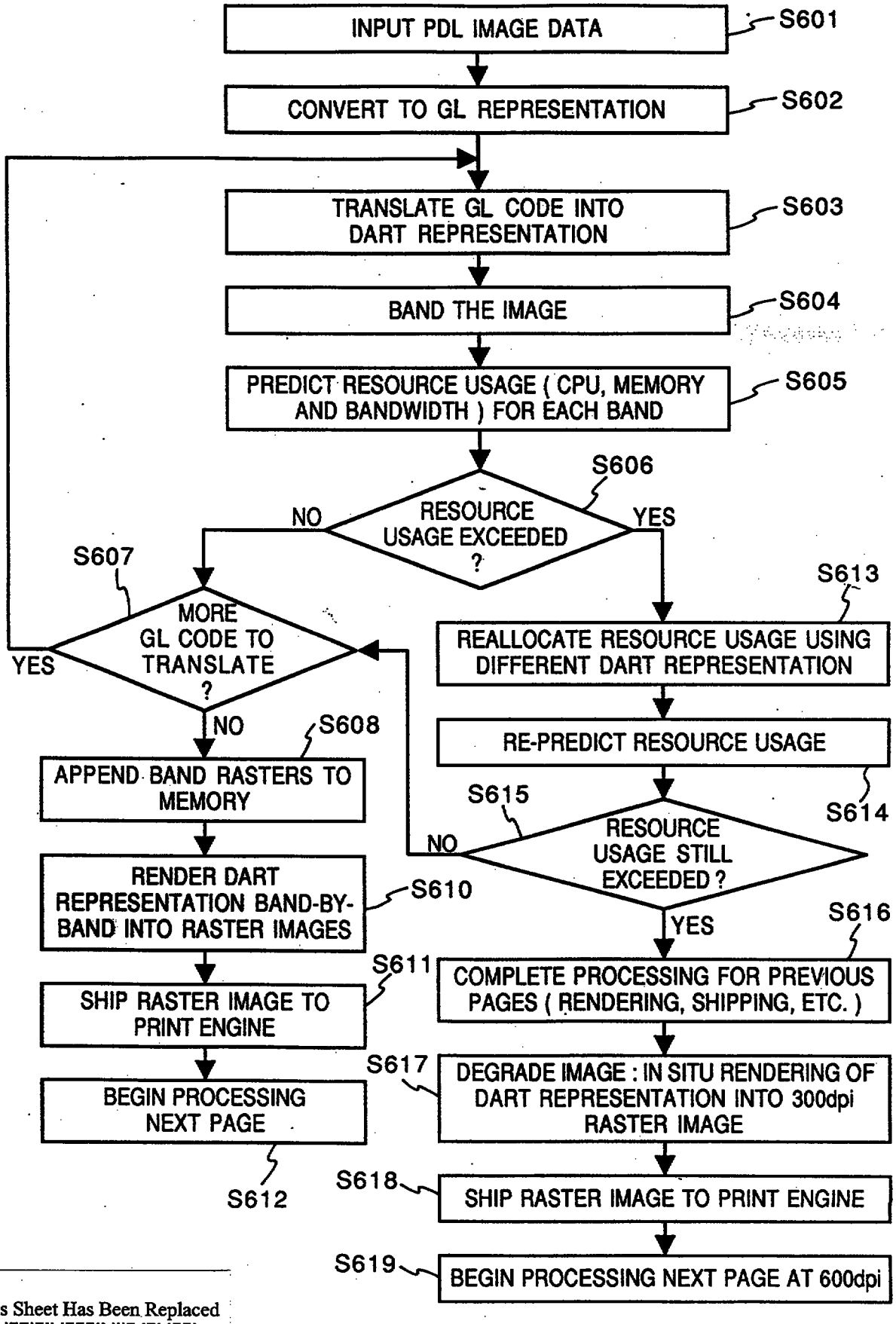
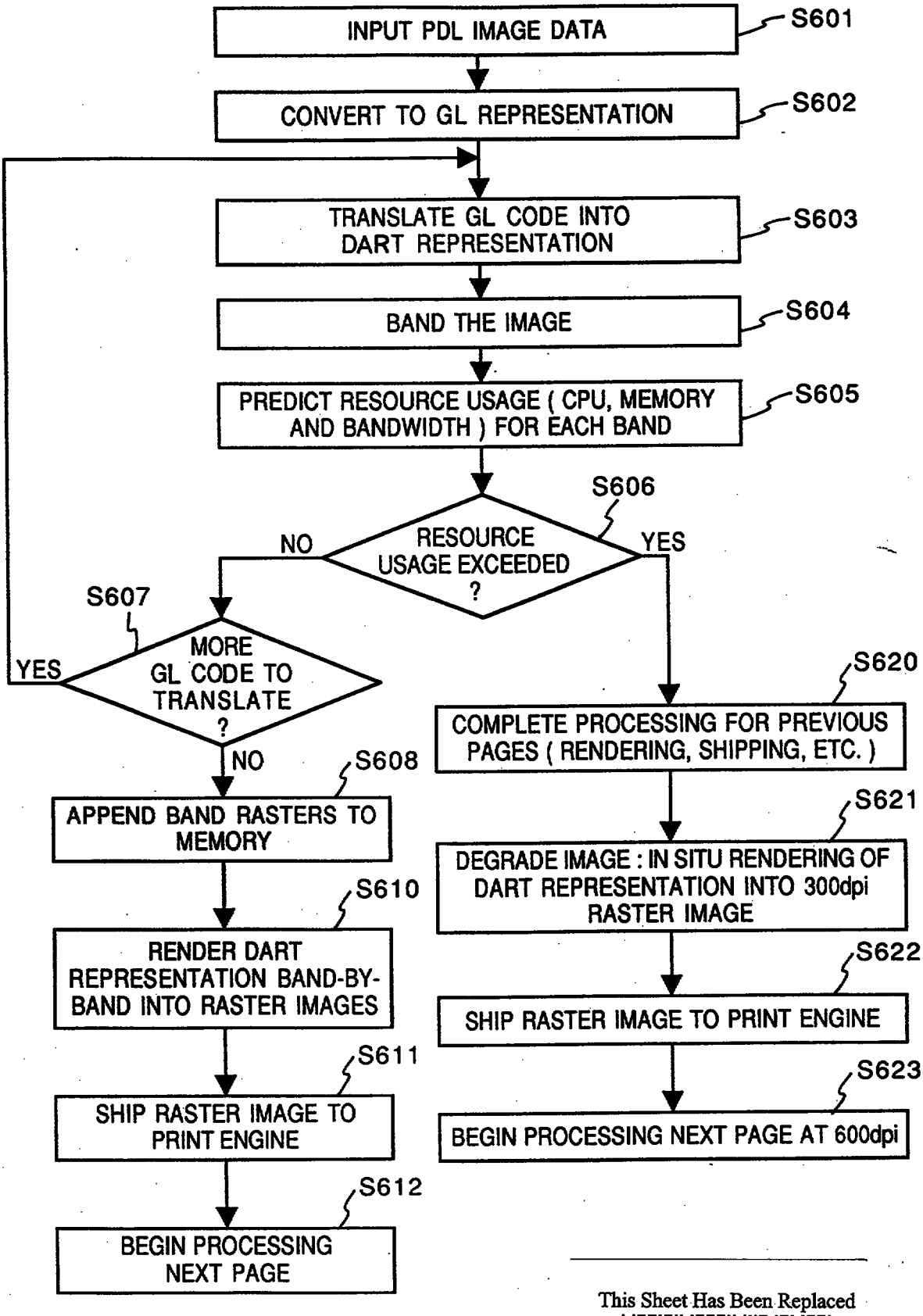


FIG. 15



This Sheet Has Been Replaced

FIG. 16



This Sheet Has Been Replaced

This Sheet Has Been Replaced

FIG. 17

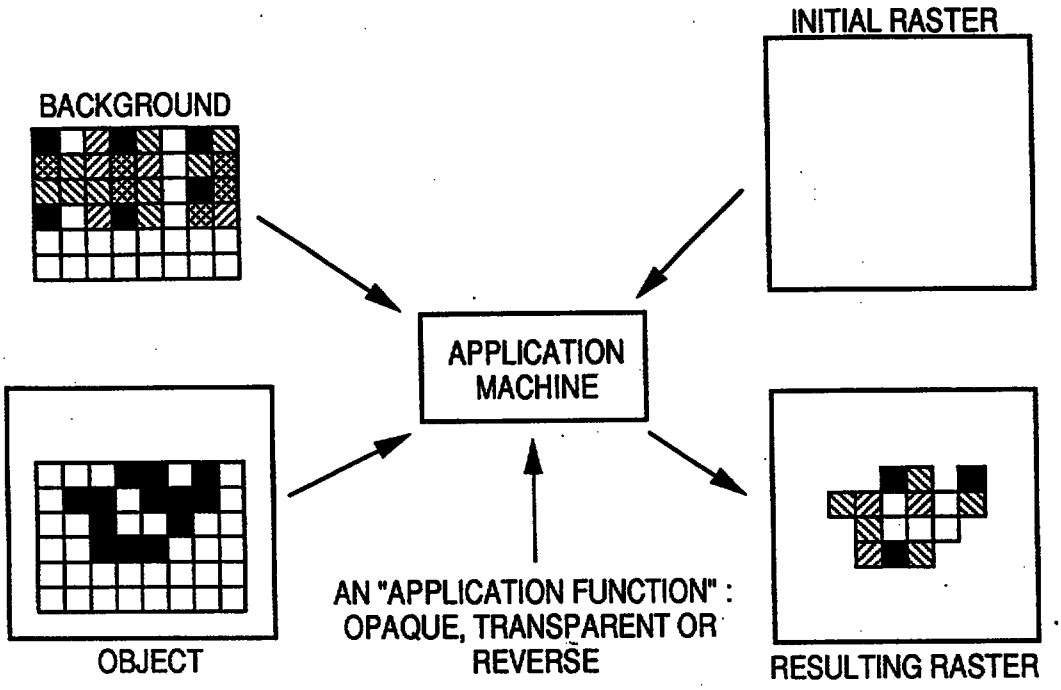
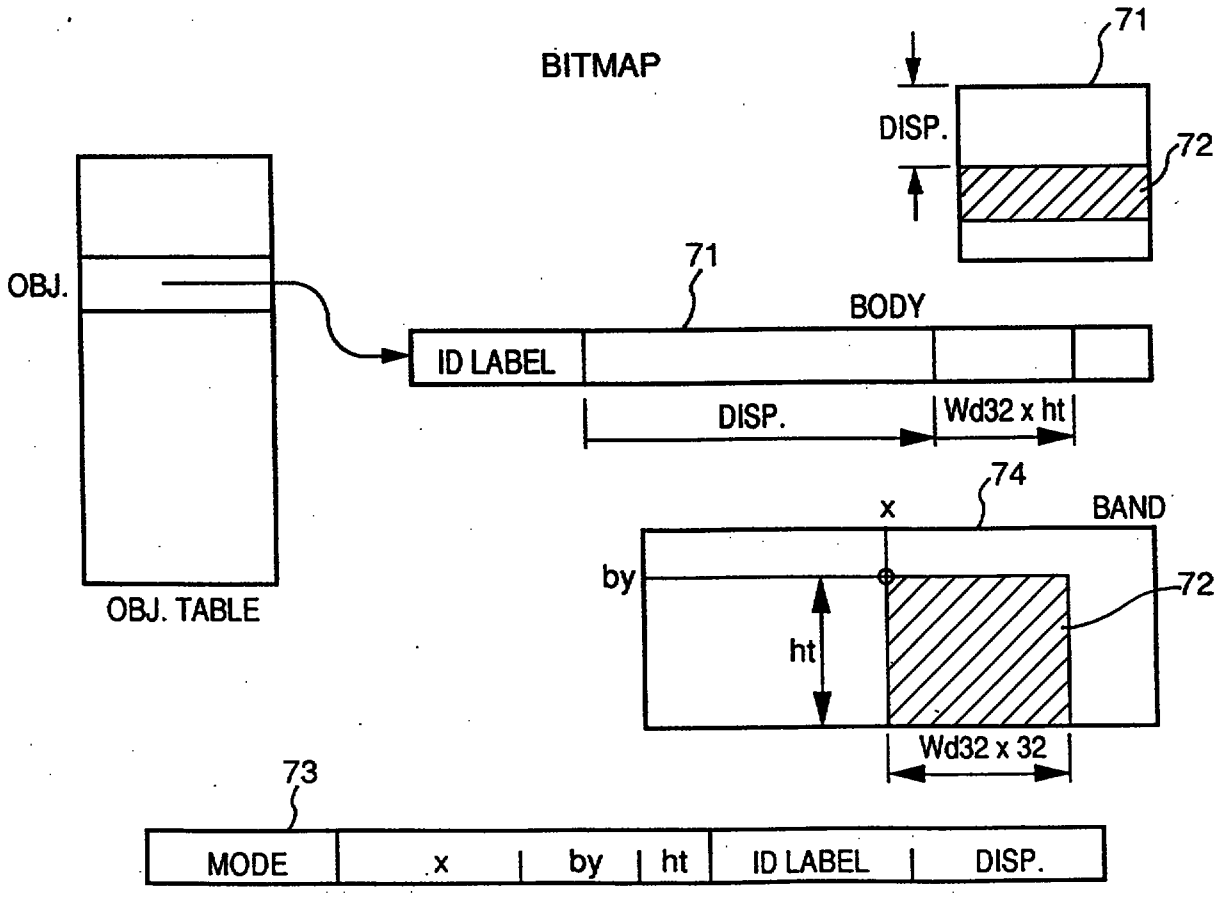


FIG. 18



This Sheet Has Been Replaced

FIG. 19

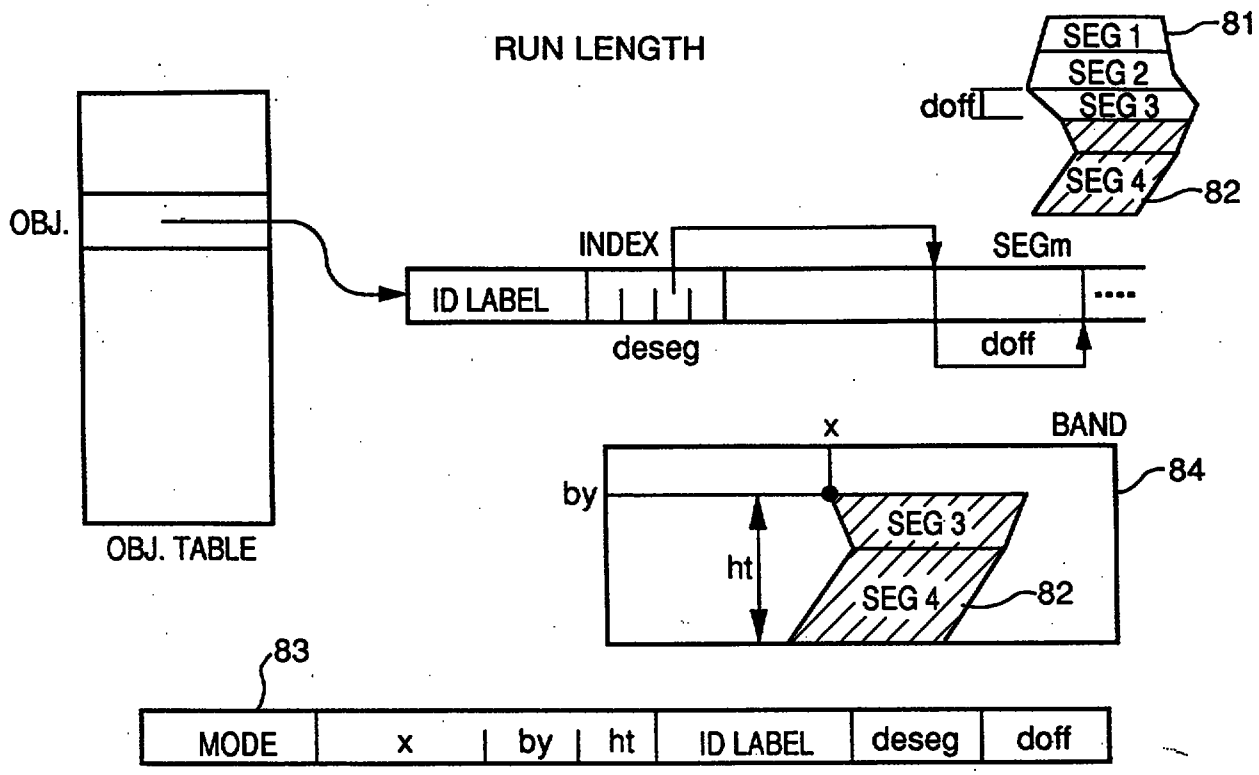
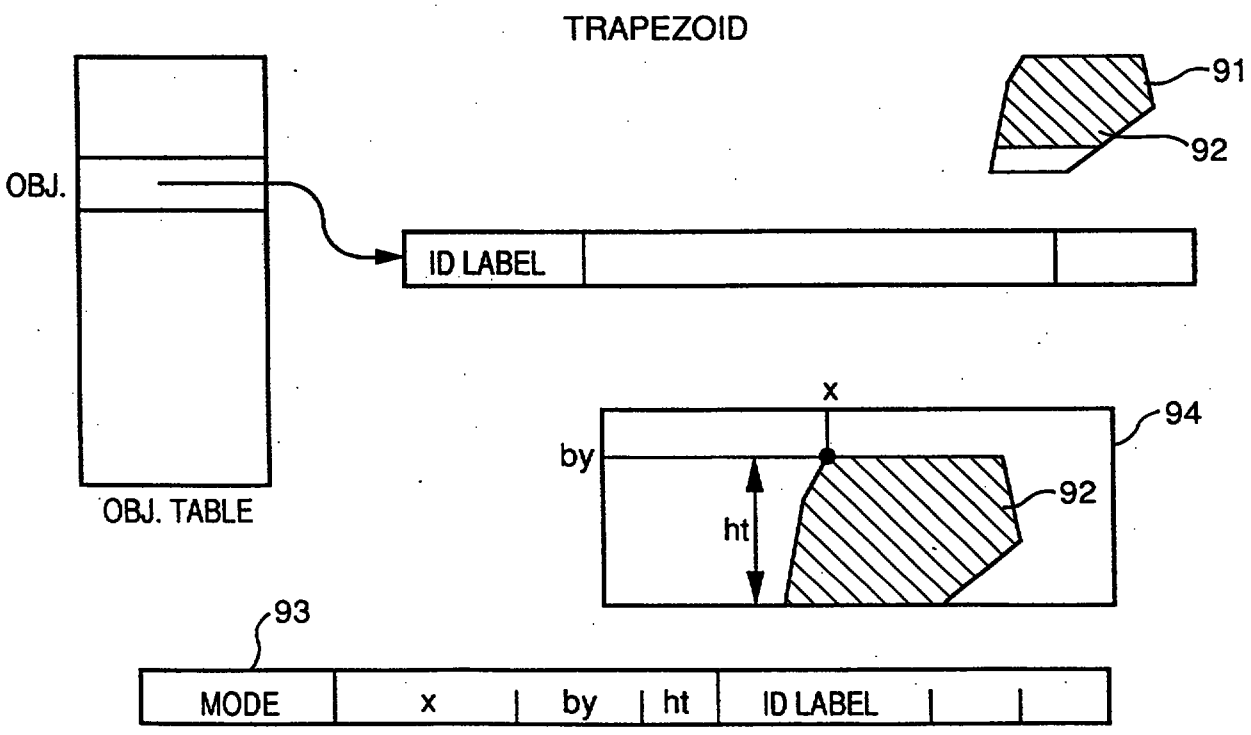


FIG. 20



This Sheet Has Been Replaced

FIG. 21

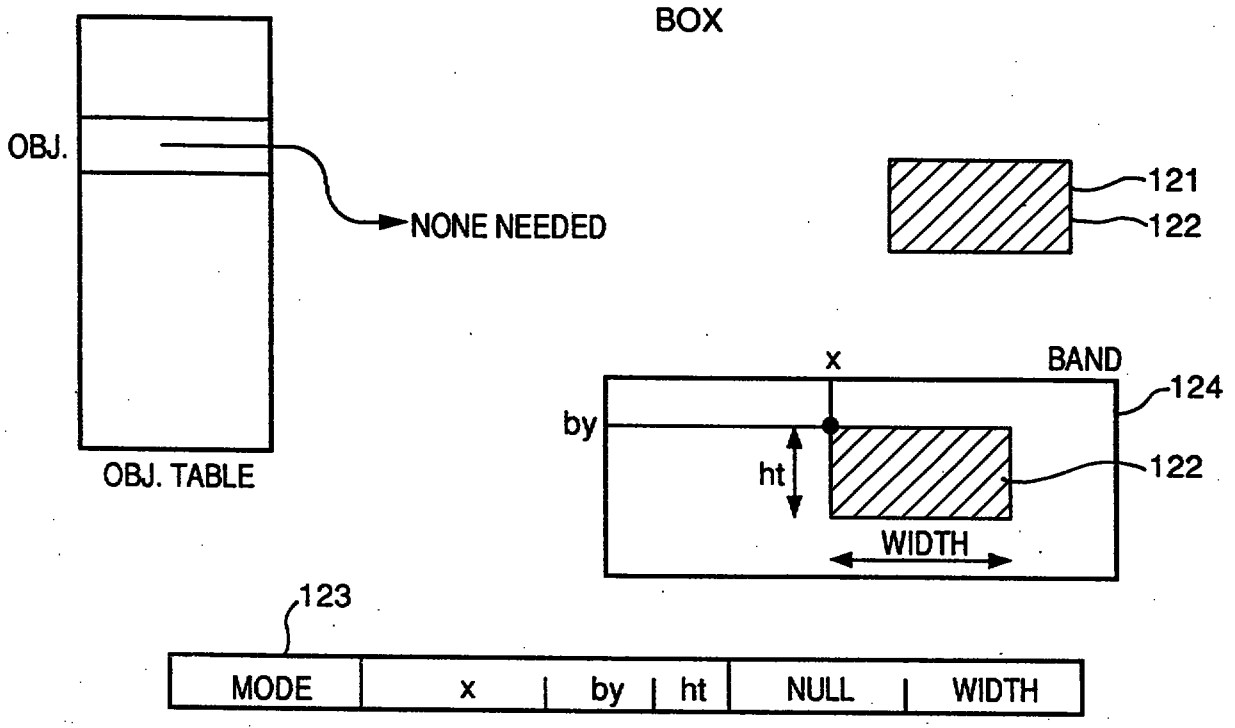
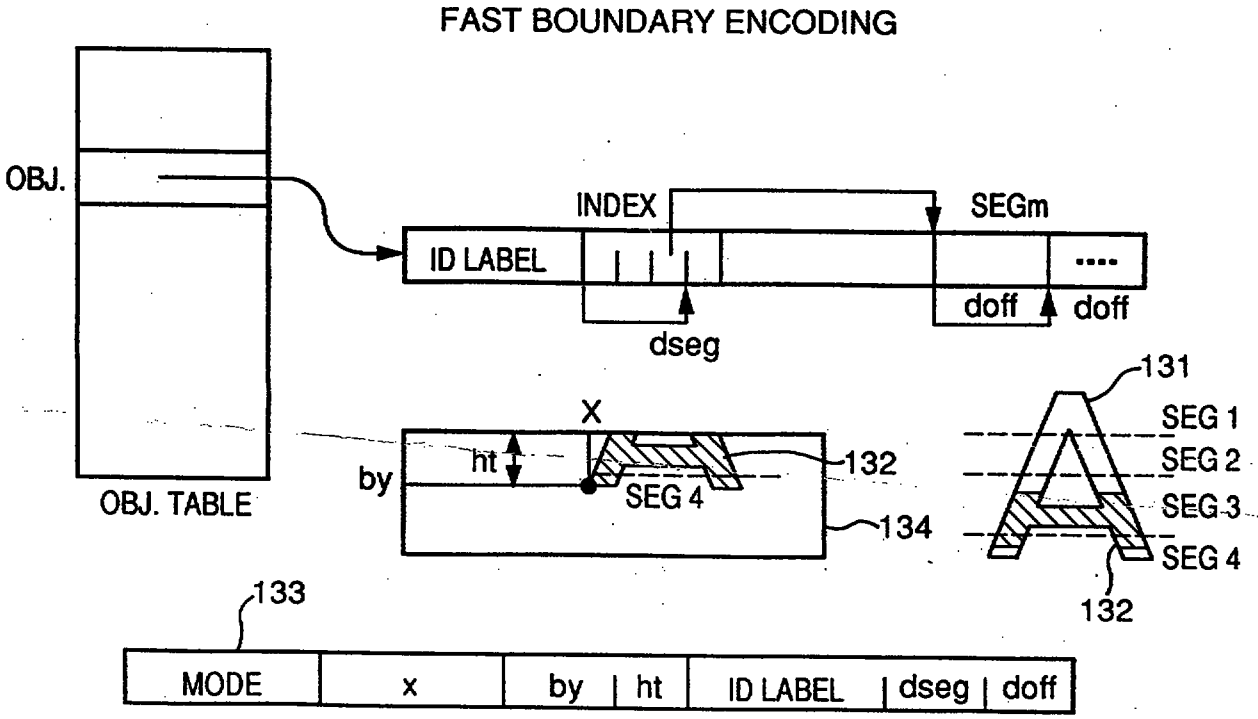


FIG. 22



THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

ABSTRACT OF THE DISCLOSURE

Method and Apparatus for Printing
According to a Graphic Language

A method and apparatus for generating graphic language document description commands based on print commands that include device-independent print commands. The printing apparatus predicts resource usage for a printing operation, determines whether resources available in the printing apparatus are sufficient for the predicted usage, and changes the printing operation in the case where it is determined that there are insufficient resources available for the currently-specified printing operation so that it is possible to print the image within existing resource constraints.

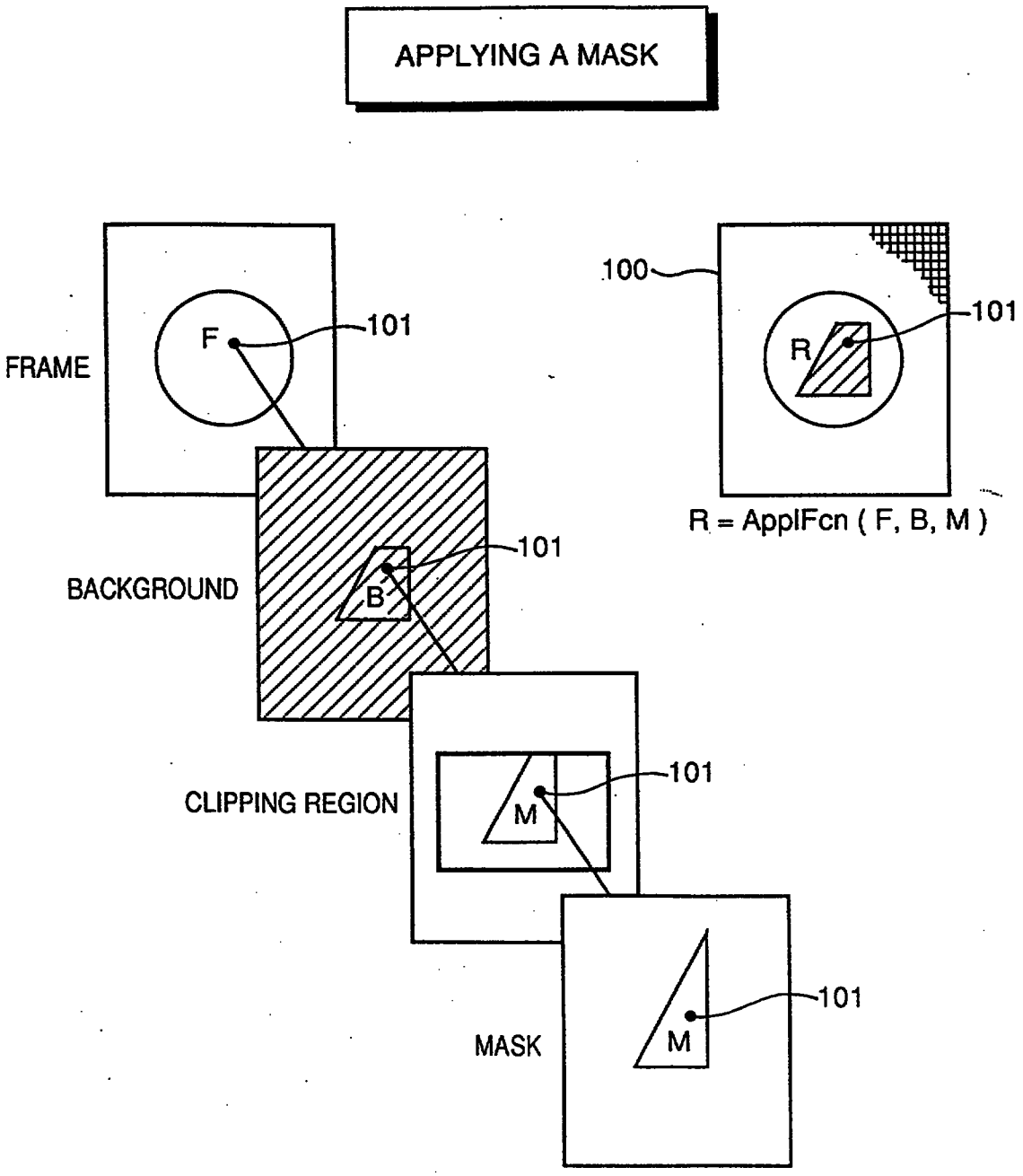
FIGURE 14

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 1

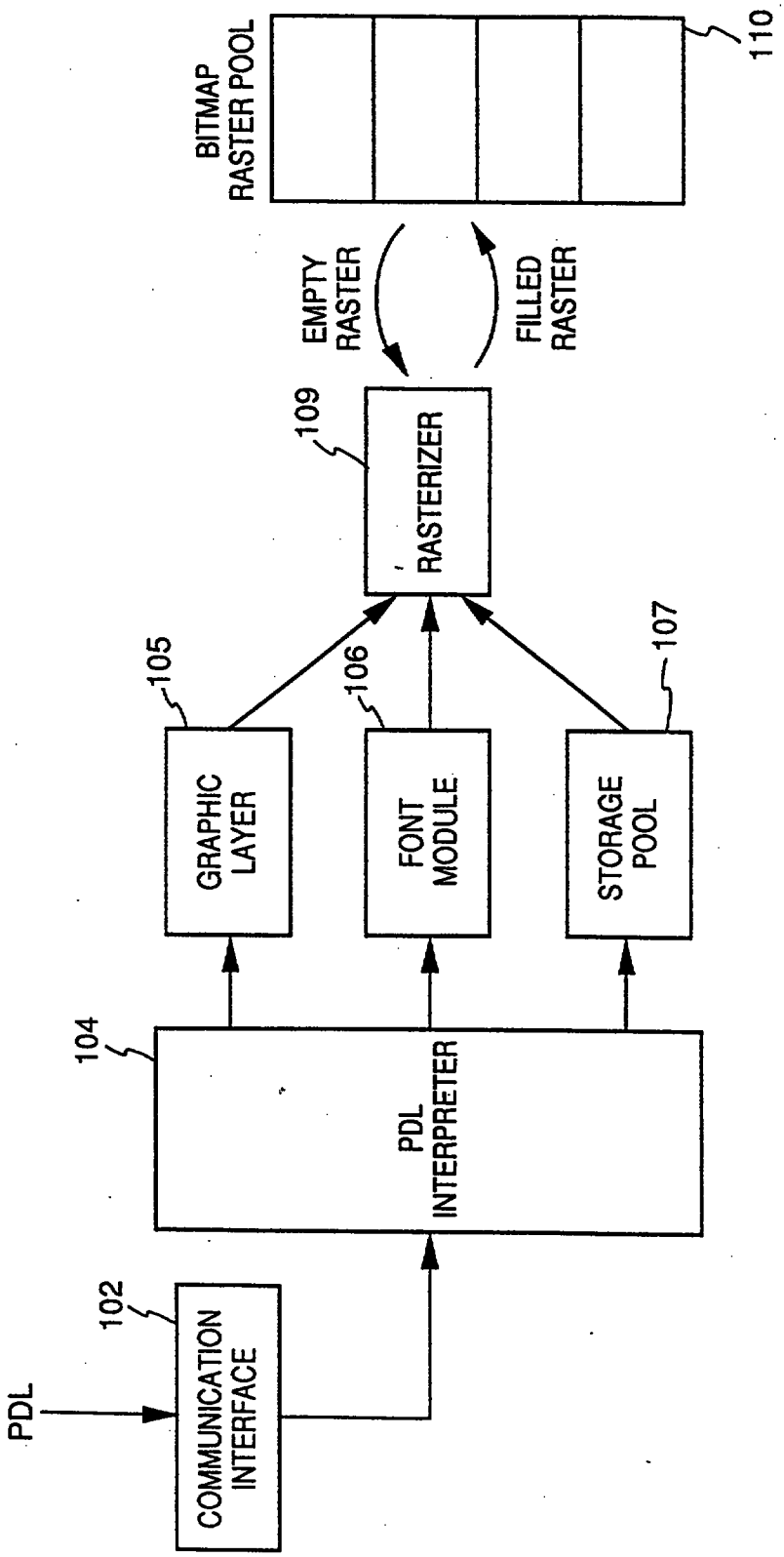


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 2

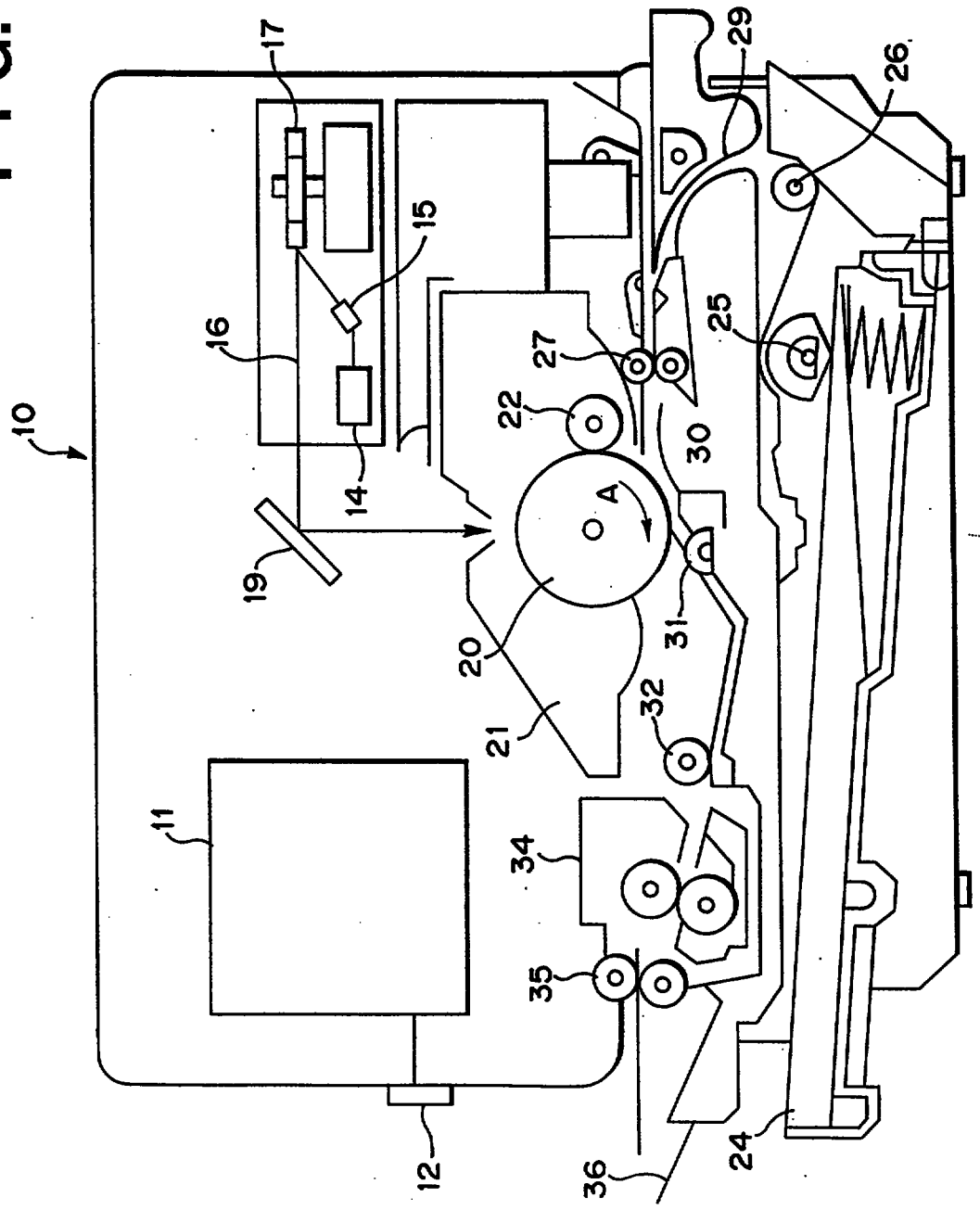


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 3

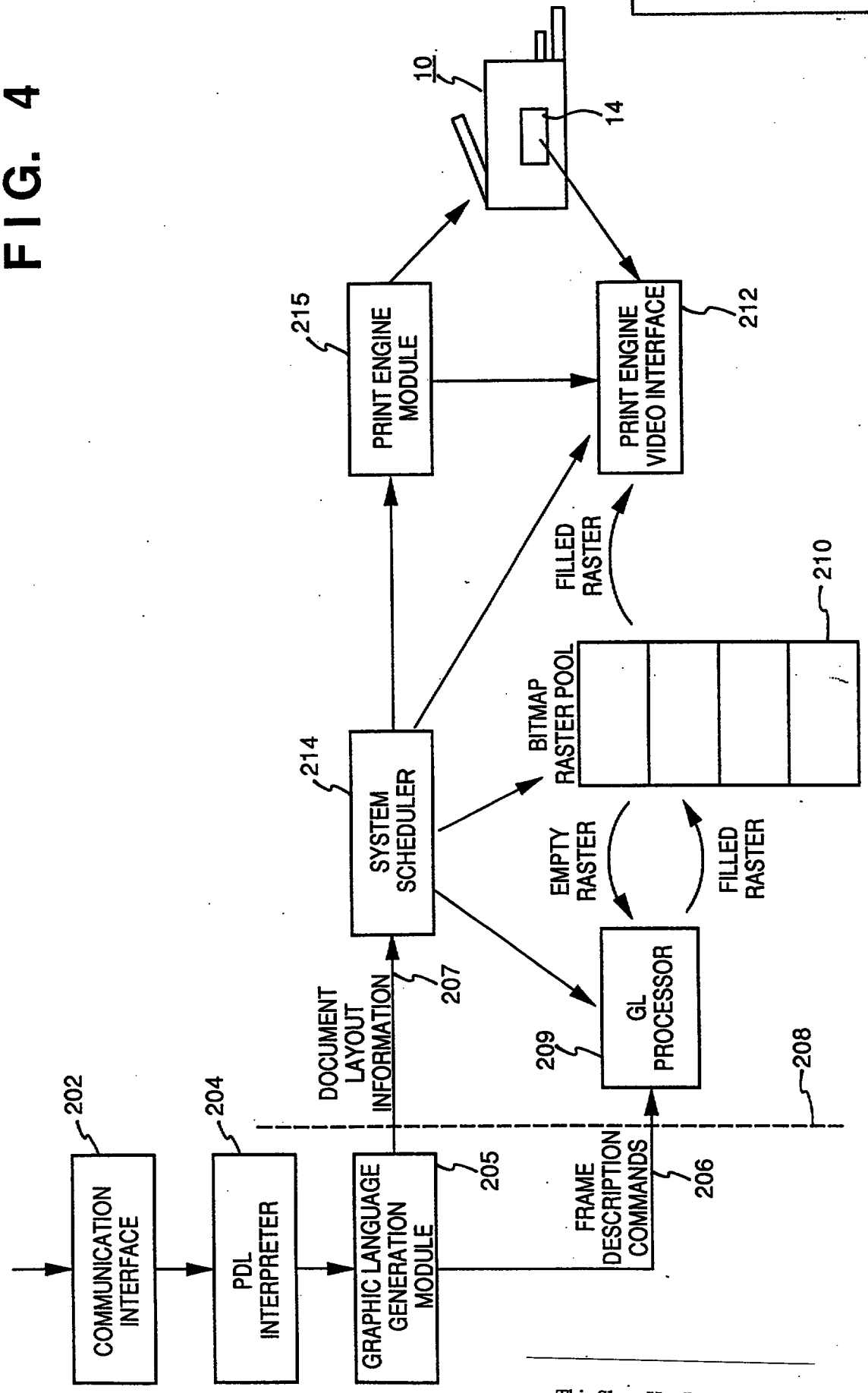


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 4

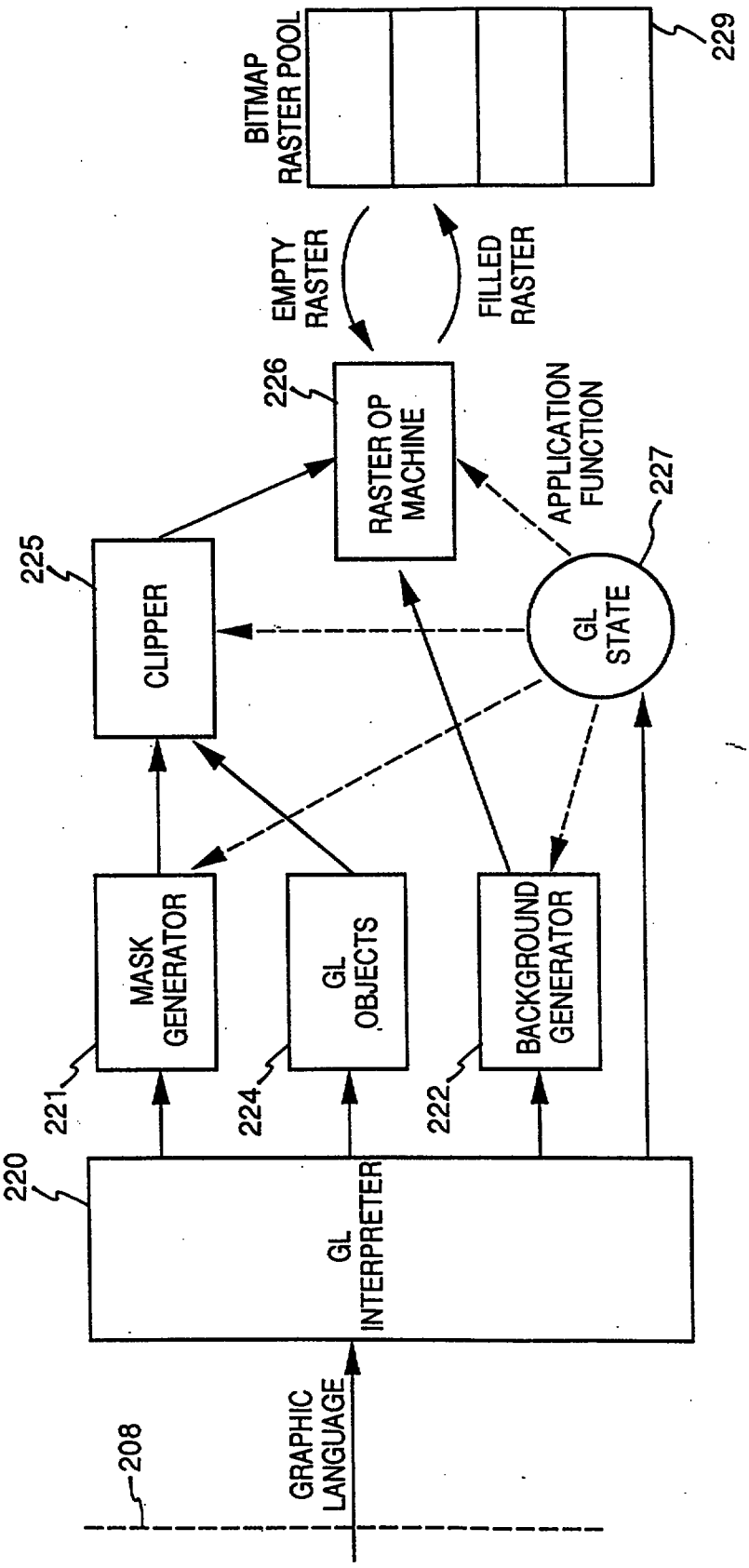


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 5

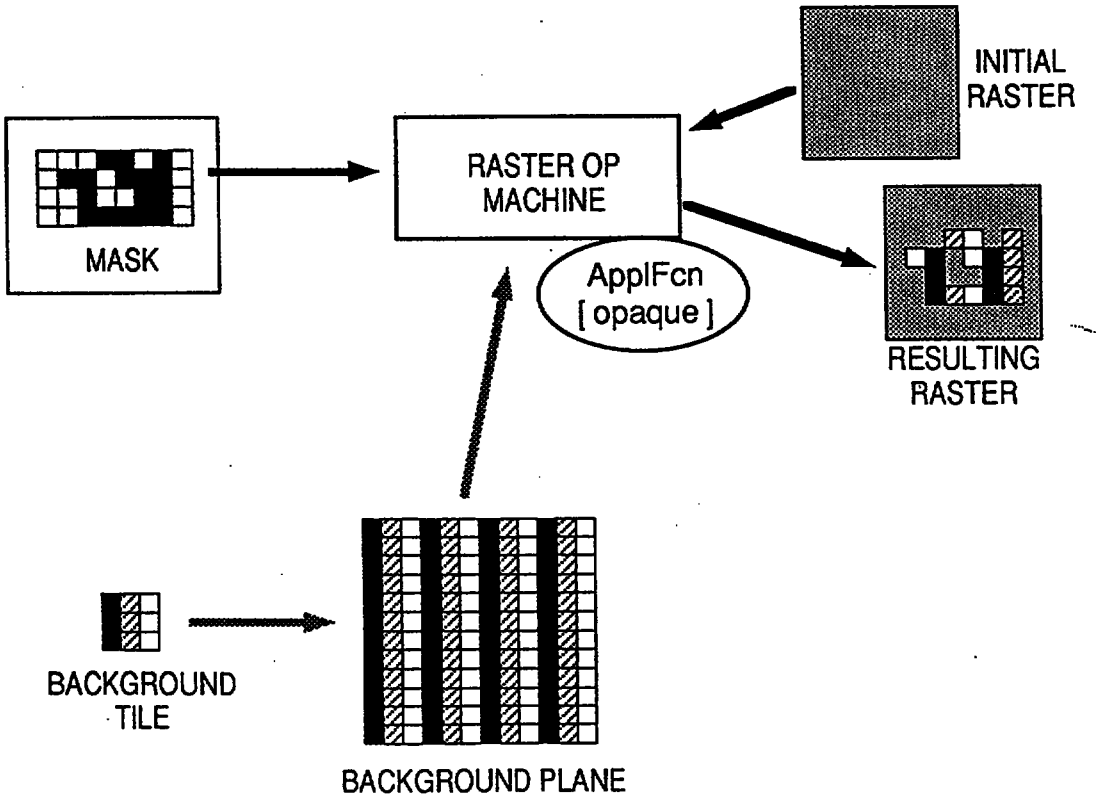


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 6

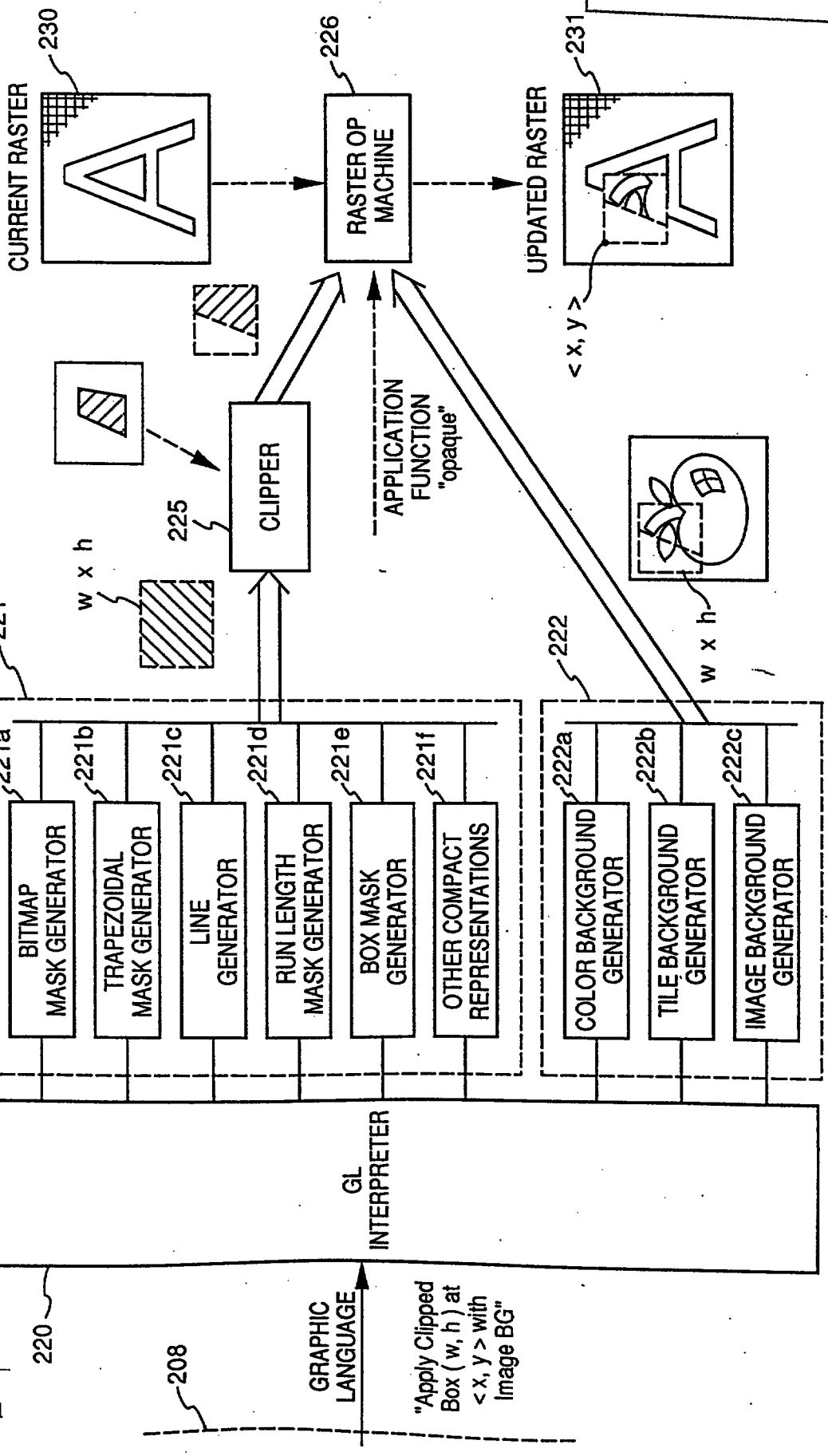


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 7

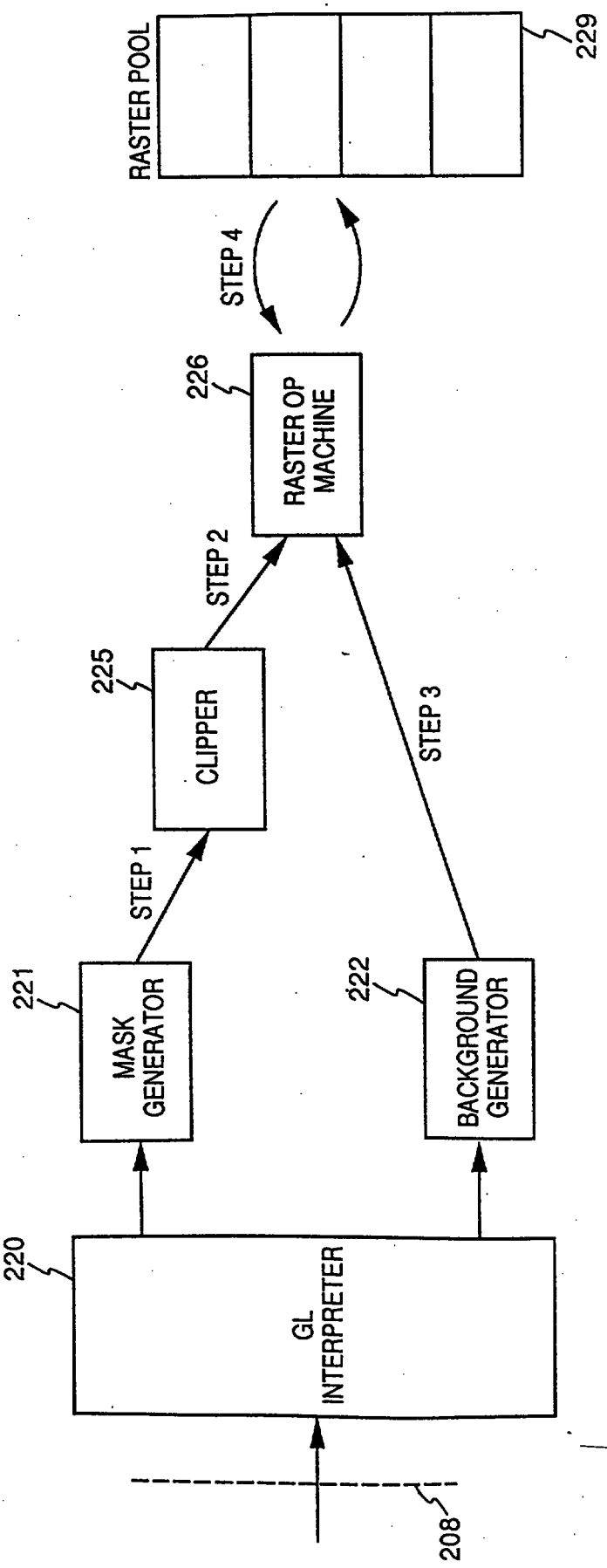


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 8

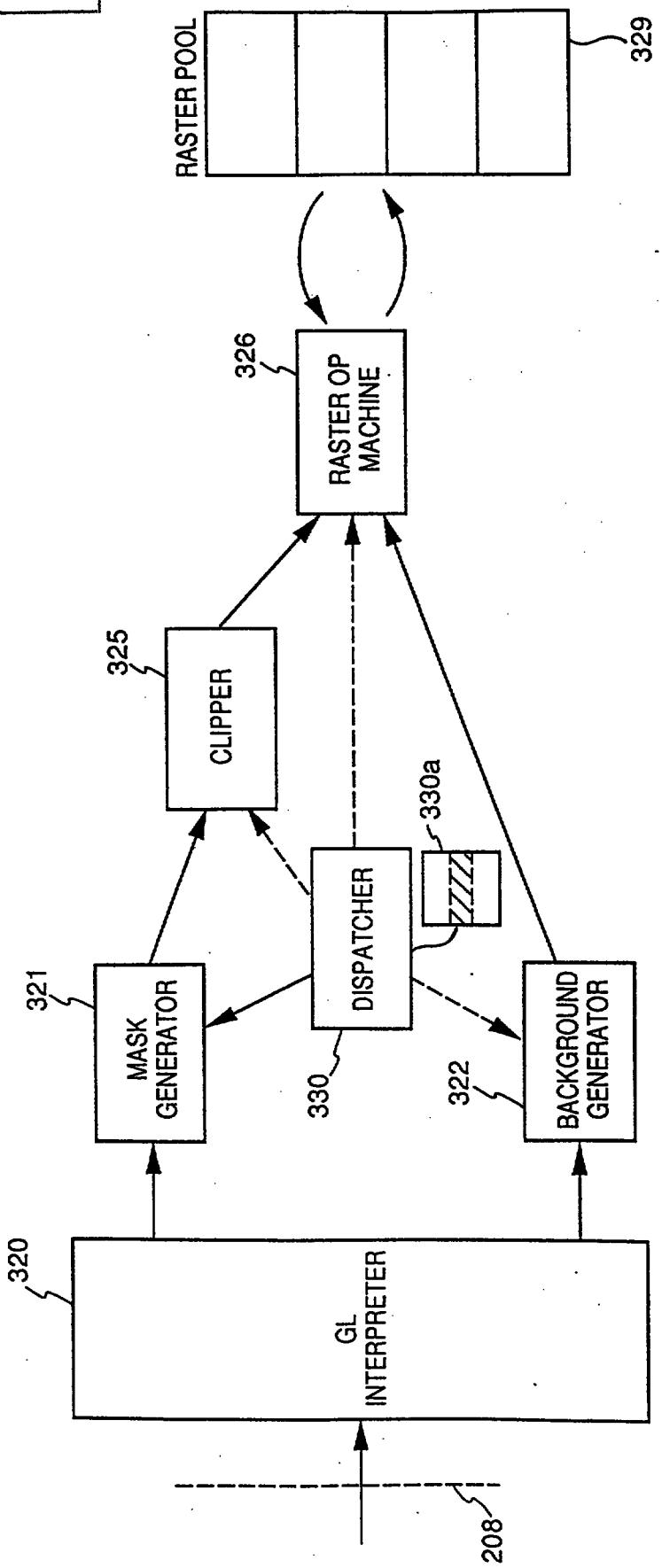


This Sheet Has Been Replaced

REPLACEMENT SHEET
25 NOV 1996

THIS SHEET HAS BEEN REPLACED

FIG. 9

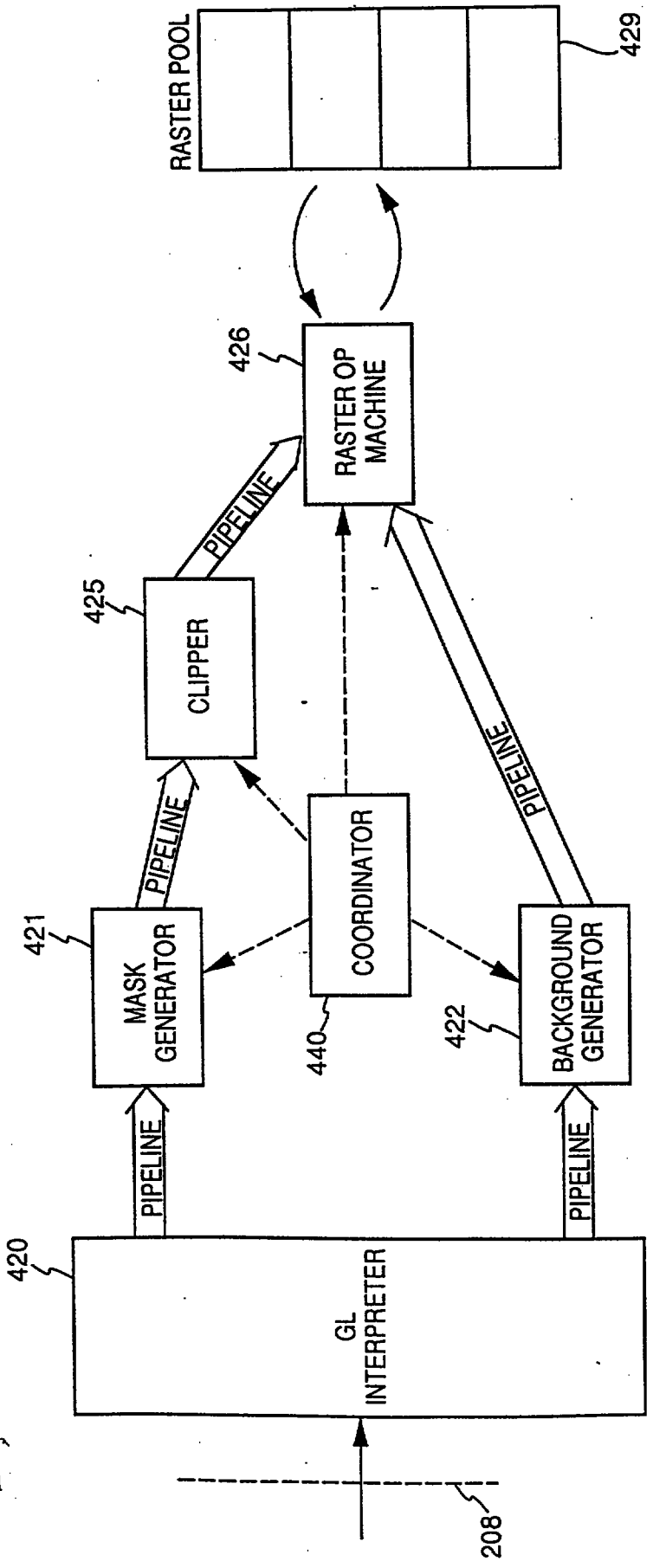


This Sheet Has Been Replaced

REPLACEMENT SHEET
25 NOV 1996

THIS SHEET HAS BEEN REPLACED

FIG. 10



This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 11

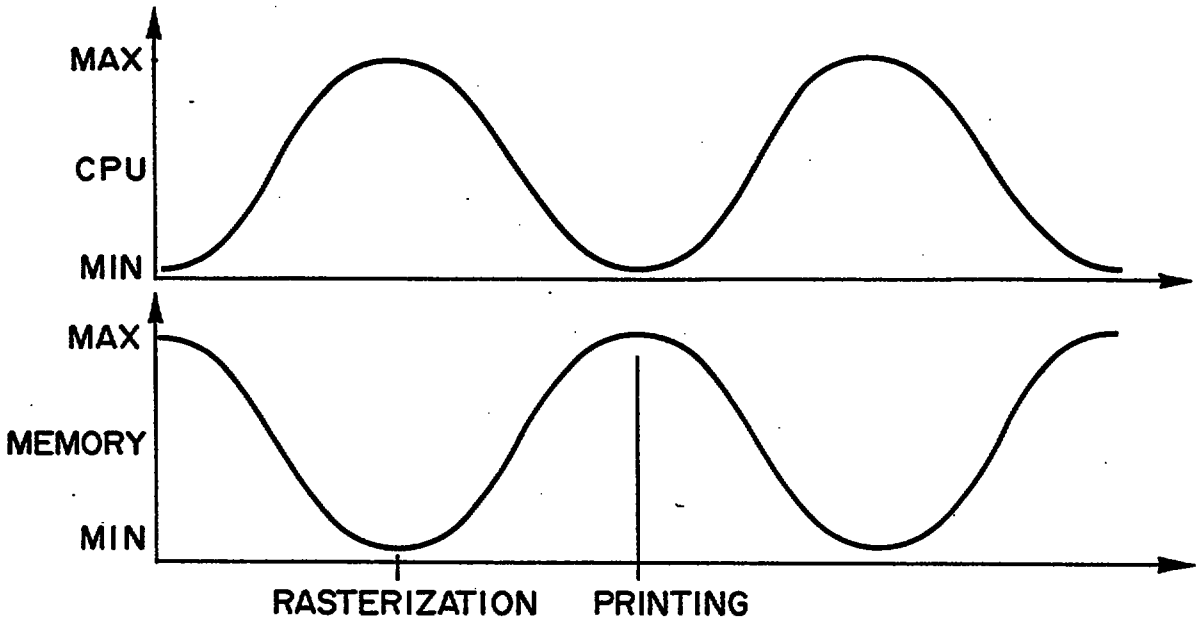
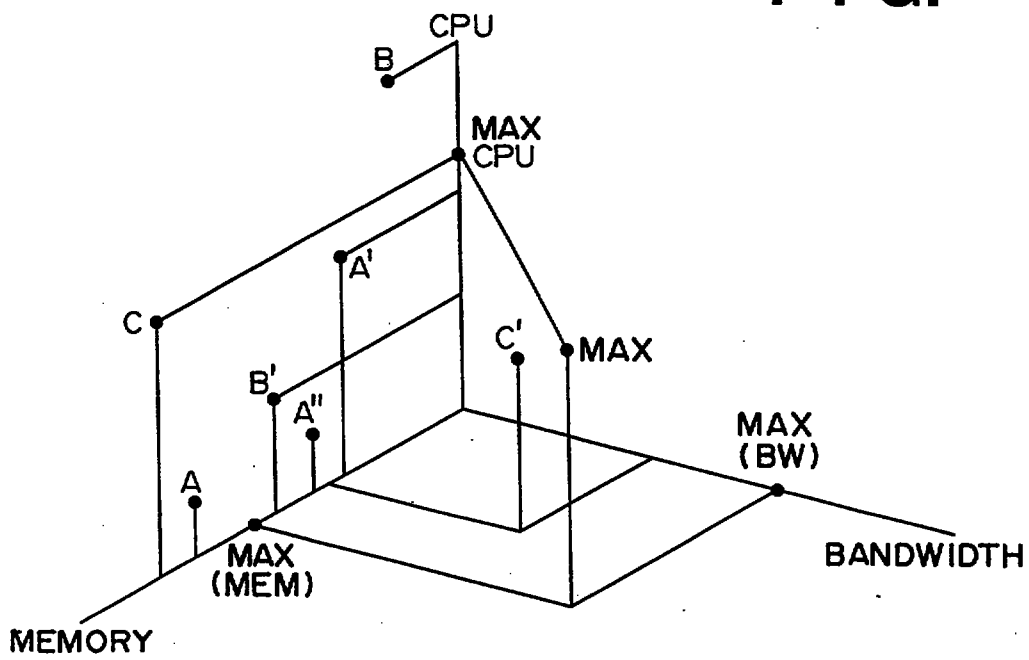


FIG. 12

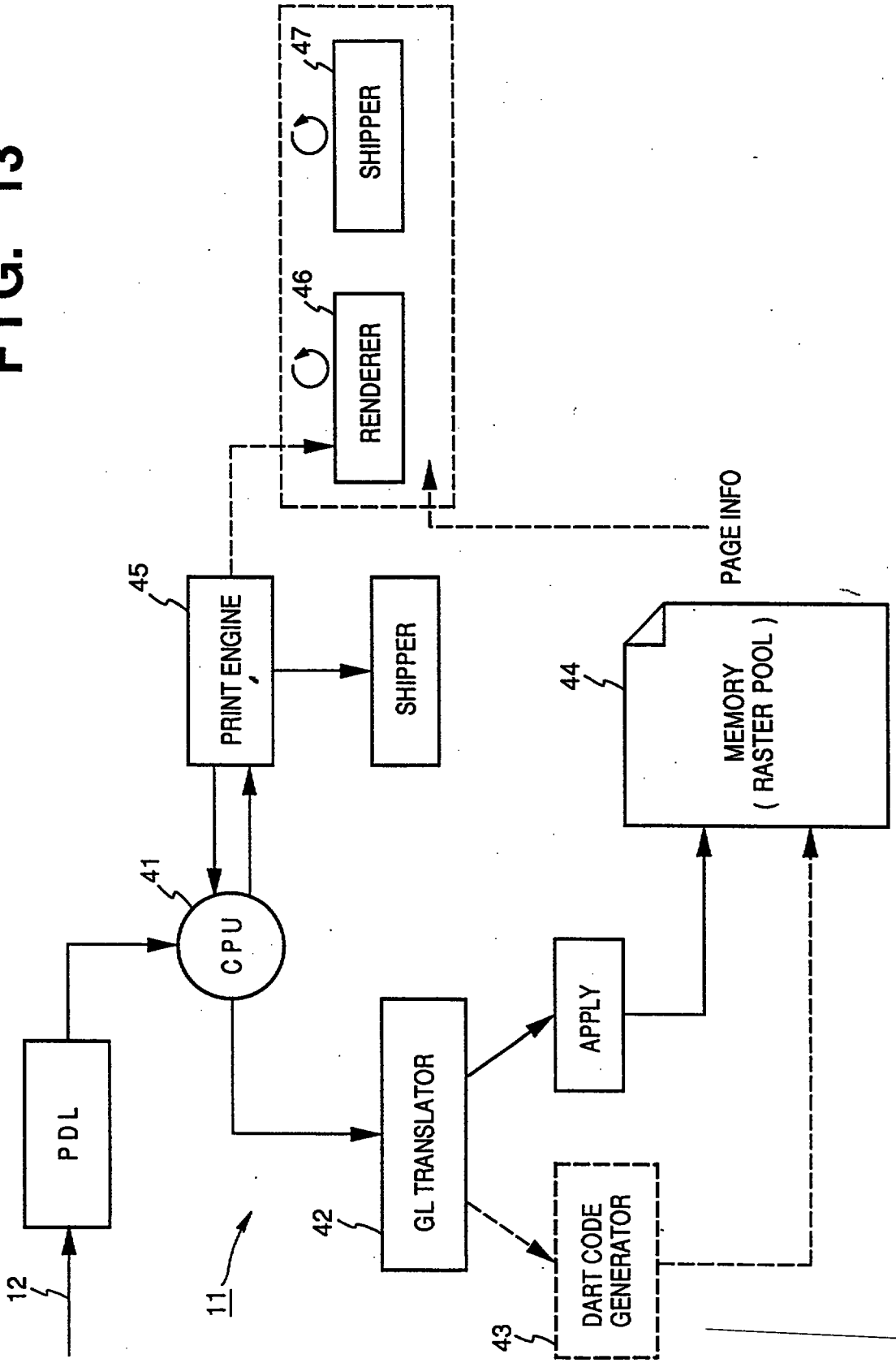


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 13

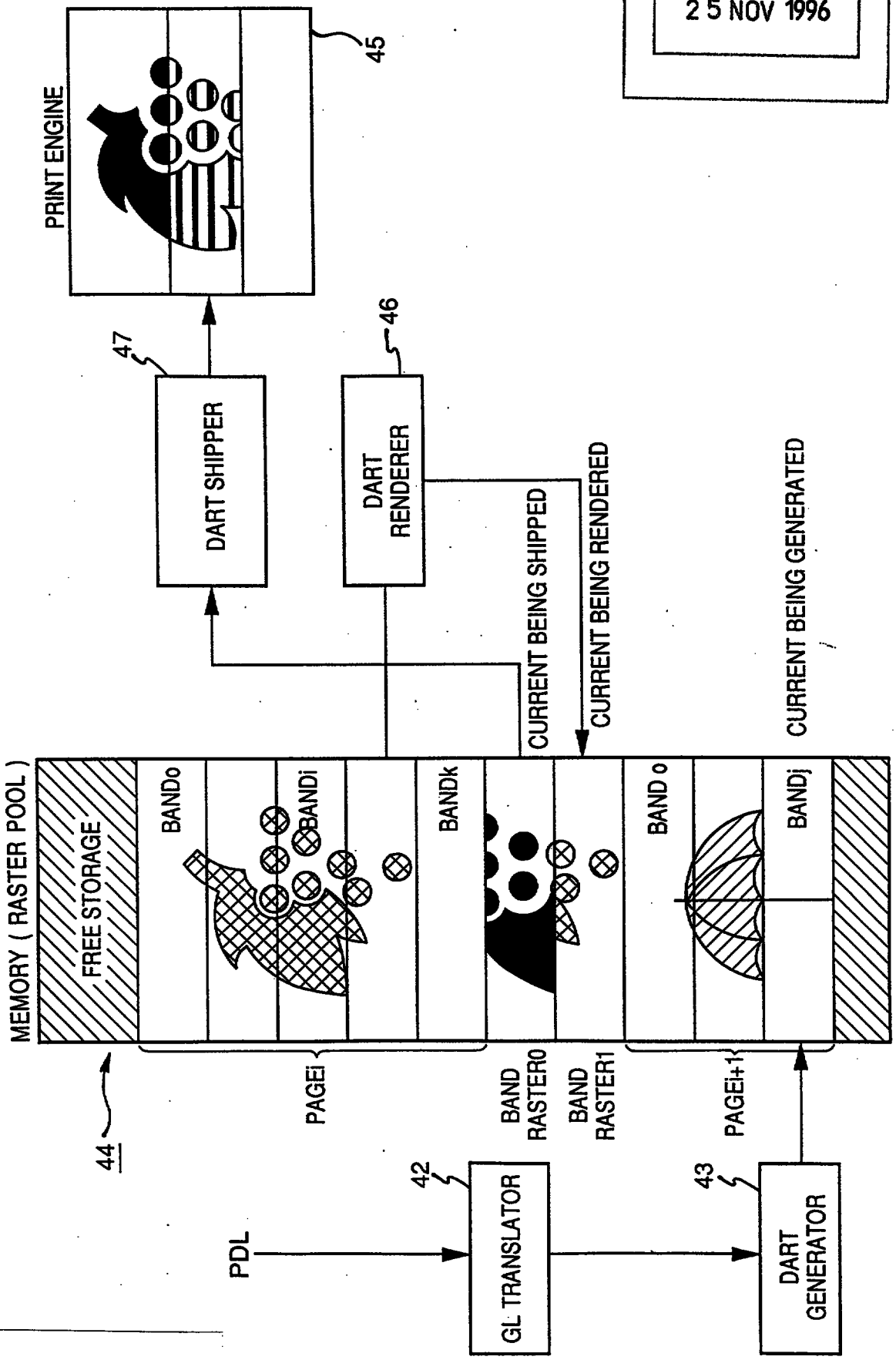


This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
25 NOV 1996

FIG. 14

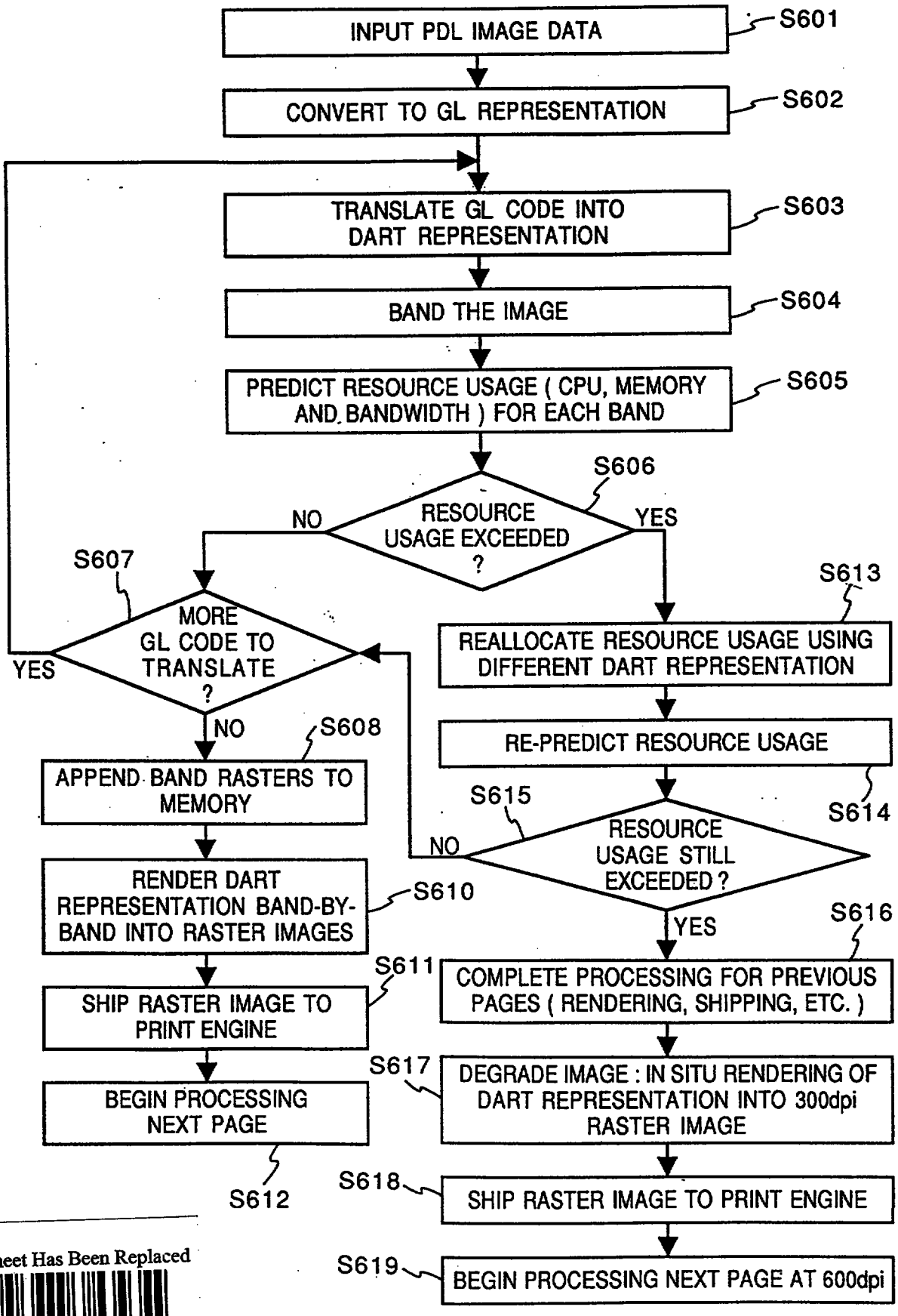


This Sheet Has Been Replaced

REPLACEMENT SHEET
25 NOV 1996

THIS SHEET HAS BEEN REPLACED

FIG. 15

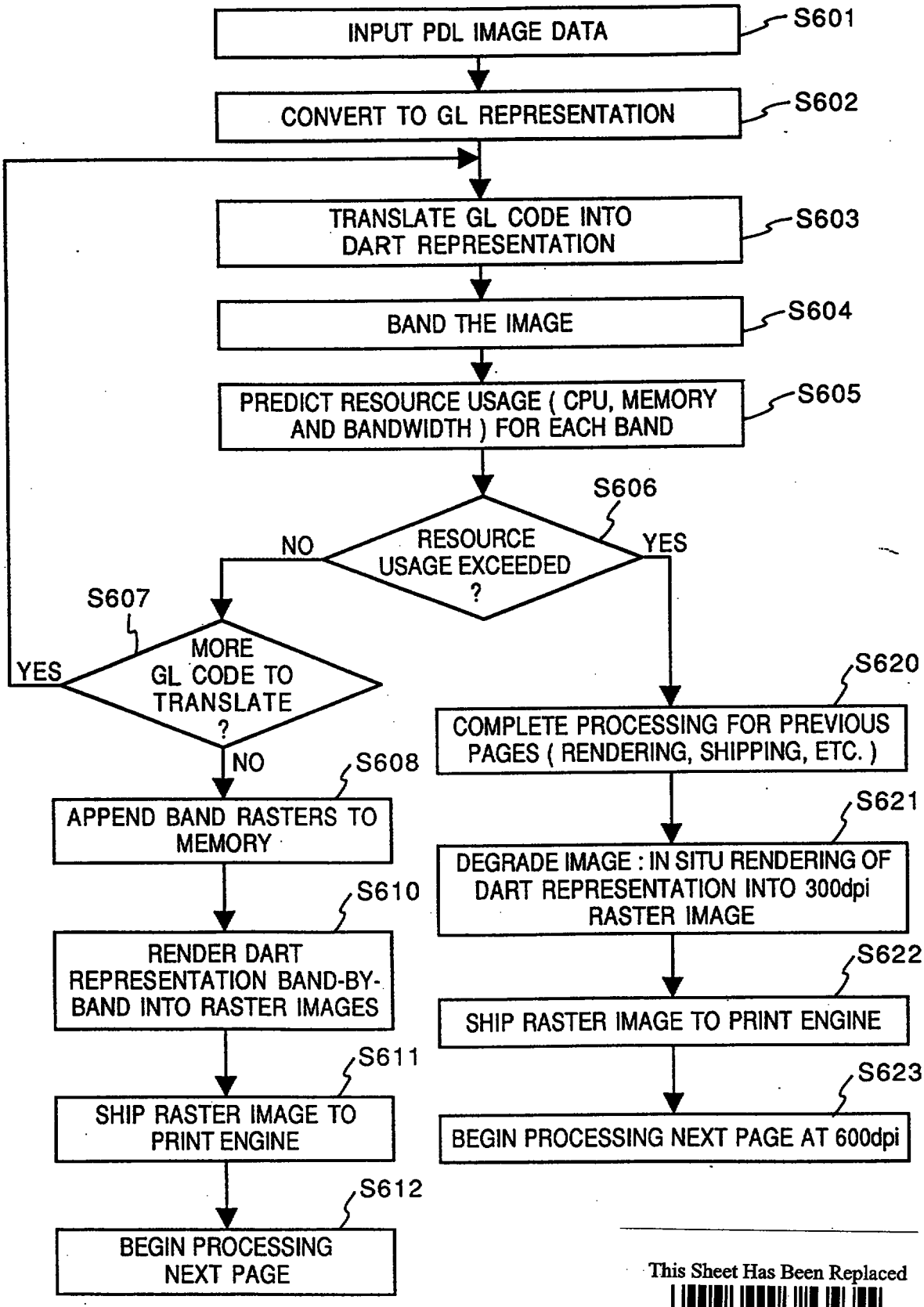


This Sheet Has Been Replaced

REPLACEMENT SHEET
25 NOV 1996

THIS SHEET HAS BEEN REPLACED

FIG. 16



This Sheet Has Been Replaced

REPLACEMENT SHEET
25 NOV 1996

THIS SHEET HAS BEEN REPLACED

FIG. 17

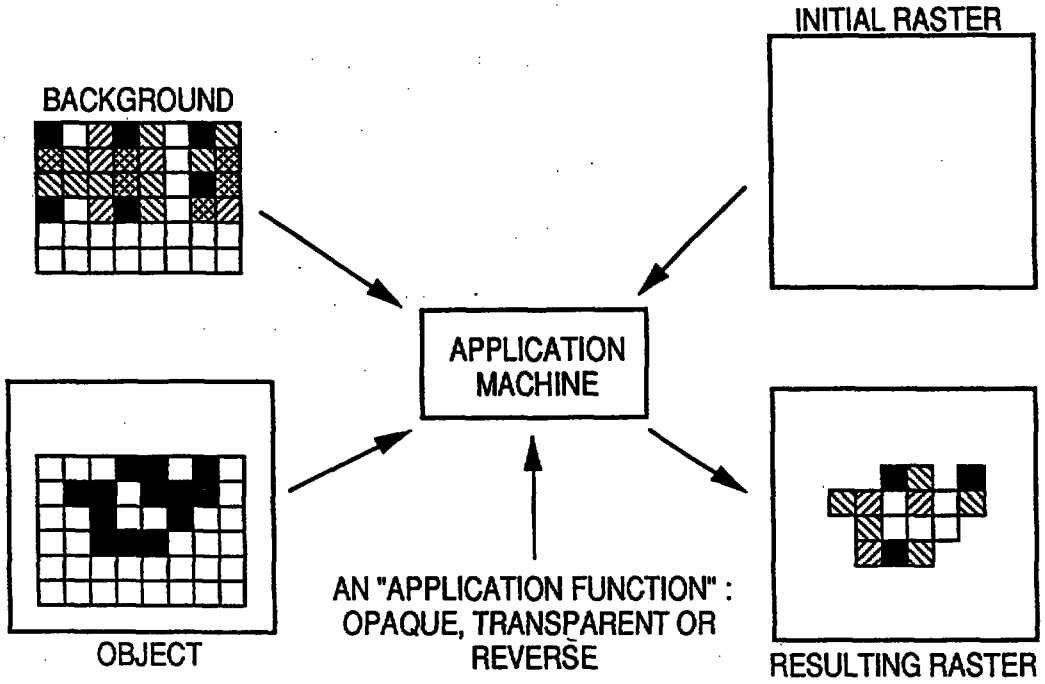
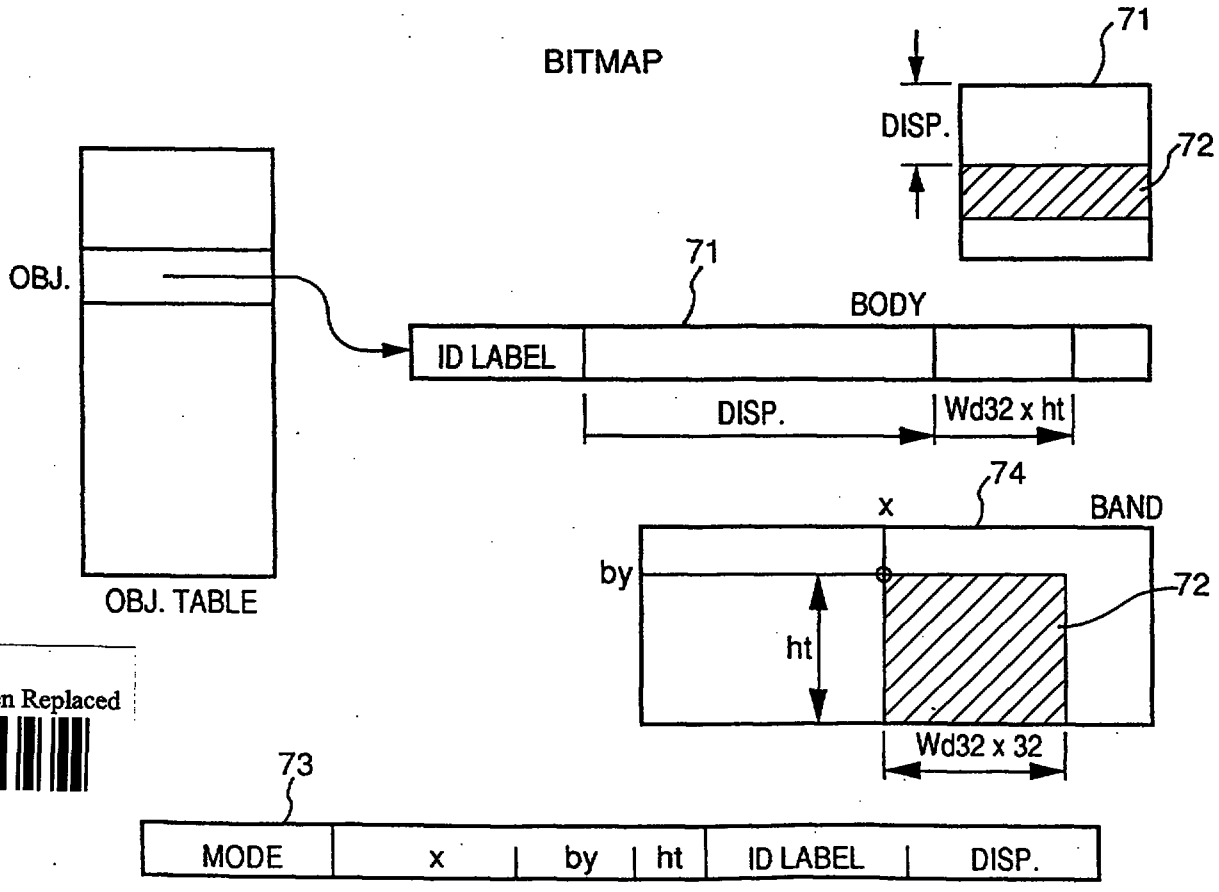


FIG. 18



This Sheet Has Been Replaced

REPLACEMENT SHEET
25 NOV 1996

This Sheet Has Been Replaced

FIG. 19

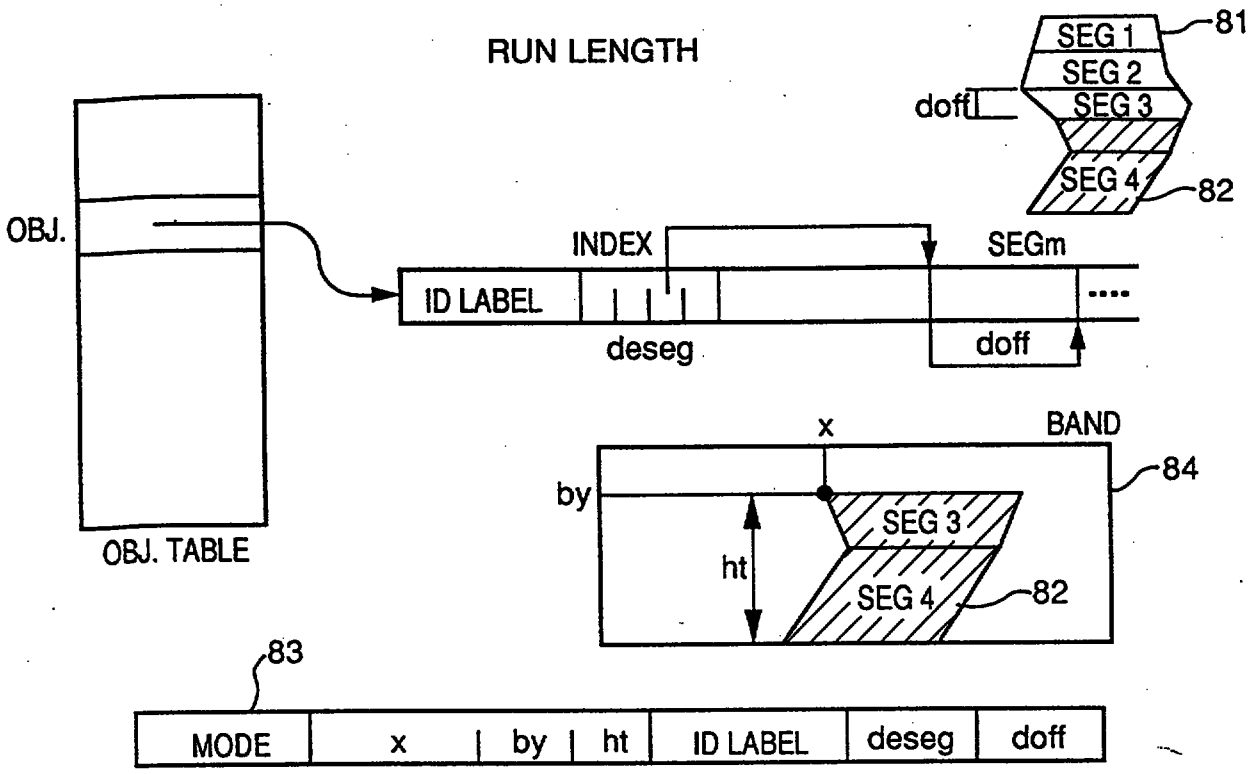
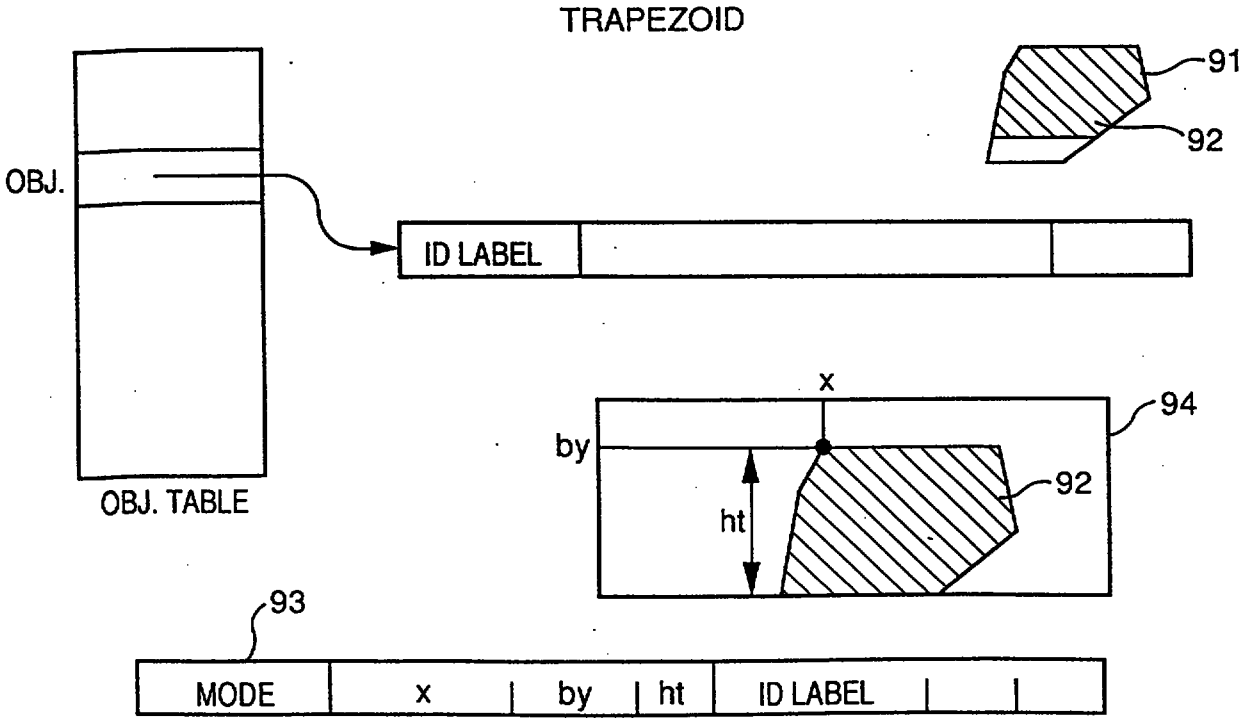


FIG. 20



REPLACEMENT SHEET
25 NOV 1996

This Sheet Has Been Replaced

FIG. 21

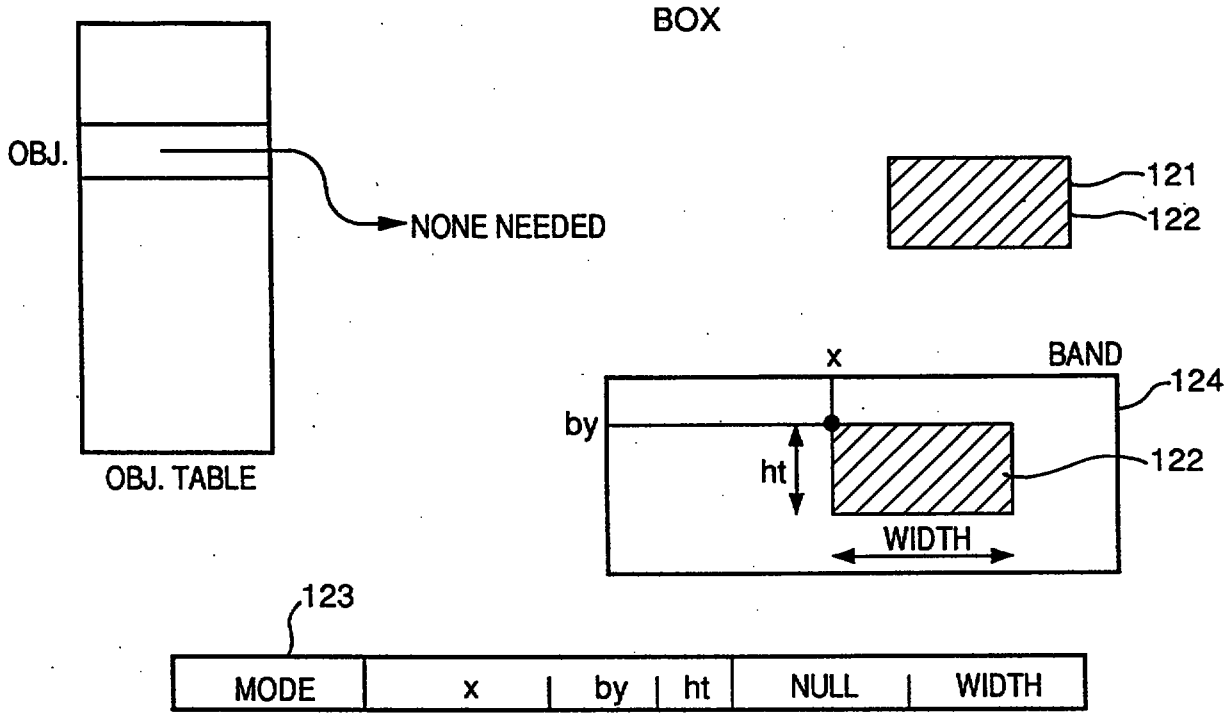
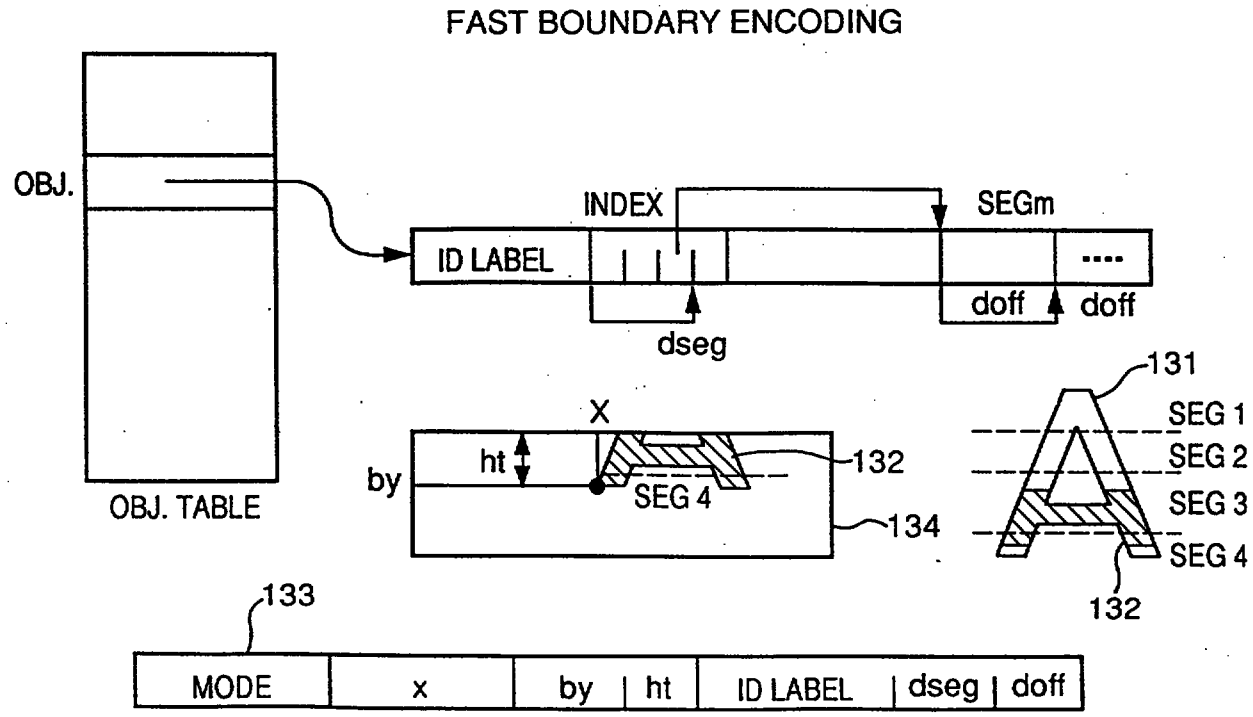


FIG. 22



960795-4

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
15 JUL 1998

METHOD AND APPARATUS FOR PRINTING ACCORDING TO A GRAPHIC LANGUAGE

This application is a division of application Ser. No. 07/911,030 filed Jul. 8, 1992, allowed, now U.S. Pat. No. 5,515,418.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method and apparatus for printing an image in accordance with a graphics language stream in which graphic descriptions for frames in a page are separated from document layout information which describes how the frames are placed on the page, and in which the graphic descriptions and the document layout information are output on separate, pipelined, interfaces. A graphics language processor is provided to intercept the graphics language stream. The processor separates the graphics language stream into mask information and background information against which the masks are applied, operates independently on the mask and background information to create frame rasters, and uses the document layout information to map the frame rasters onto the page.

2. Description of the Related Art

Recently, high quality raster-based devices, such as laser beam printers and ink jet or bubble jet printers and the like, have become widely available. These devices receive image data from a host computer and generate a raster image in bit map memory for subsequent imaging and/or printing.

FIG. 1 shows the rasterization process by which each pixel in bit map memory is formed, and by which qualities (such as color, grey level or half toning) are given to each pixel. As shown in FIG. 1, the bit map raster **100** is divided into frames F which define areas in the bit map raster. For each pixel **101** in the frame, a mask M and a background B are defined. The mask M defines whether or not the pixel should be rendered into bit map memory and the location where the pixel should be rendered. The background B defines the qualities with which the pixel **101** should be rendered. For example the background B may comprise a color associated with the pixel or it may comprise a halftone density. If desired, a clipping region may also be defined to specify the effective areas of the mask and allow more precise control over the masks. The rasterization of pixel **101** in the bit map memory is the intersection of the frame information, the background information, clipping region information, and mask information, all as illustrated in FIG.

This Sheet Has Been Replaced

REPLACEMENT SHEET
15 JUL 1998

1.

As the versatility of raster printing devices increases, more sophisticated techniques for instructing how to form the raster image have been developed. That is, it is not ordinarily practical to transmit a full raster image to the printer. Rather a descriptive language such as page description language ("PDL") is used. A page description language is a series of commands that describes all characteristics of each element on every page of the document, such as text, graphics, scanned images. The PDL may also include information that describes document layout and sheet handling (collation, etc.). The host computer transmits page description language to the printing device, and the device interprets the page description language so as to render the raster image.

FIG. 2 is illustrative of this process. FIG. 2 depicts modules in a printing device which accept page description language at one end and create a bit map raster image at the other end. As shown in FIG. 2, a communication interface module **102** receives PDL commands and passes them to a PDL interpreter **104**. The PDL interpreter directs PDL commands to a graphic layer **105**, a font module **106**, and a storage pool **107** as appropriate to the particular PDL command being processed by the interpreter. A rasterizer **109** accepts output from the graphic layer and the font module and the storage pool, and forms a raster image in a bit map raster memory. Specifically, the rasterizer **109** accepts an empty raster from bit map raster pool **110** and returns a filled raster to the raster pool. The filled raster is imaged by the printing device during which time a further empty raster (corresponding to a succeeding page) is being filled by rasterizer **109**.

Printer commands like PDL commands are normally device-independent which means that the final document will look the same for the same PDL commands regardless of the printing device that receives the PDL commands. While this is advantageous in some respects, it has caused difficulty in that each different printer, because of its own unique characteristics, must be designed with its own unique PDL interpreter **104** which translates device-independent PDL commands into corresponding raster images.

SUMMARY

It is an object of the present invention to address the foregoing difficulties.

In one aspect, the invention generates graphic language document description commands based on print commands that include device-independent print commands. Frame

REPLACEMENT SHEET
15 JUL 1998

descriptions for raster frames in the document are generated from the print commands as are document layout commands which define how the raster frames are positioned on the page raster. The frame descriptions are output on a first pipelined interface and the document layout commands are output on a second pipelined interface. The document layout commands may include information as to how frames are mapped onto a sheet of paper, and may also include document finishing instructions such as stapling and collation instructions. The frame descriptions may be raster-based descriptions and are ordinarily device-dependent. The invention may be embodied in a raster-based printing or display device such as a laser beam printer or ink jet or bubble jet printer, but it also may be embodied in a printer driver in a host computer and executed as part of host computer processes.

In another aspect, the invention is a graphics language processor that includes a mask generator for generating pixel mask objects in a frame. A background generator is provided for generating backgrounds against which the objects are applied. An interpreter interprets a stream of graphic language commands input to the processor and activates the mask generator and the background generator in accordance with the graphics language. A rasterizer applies the mask objects to the background onto a bit map raster memory. The mask generator may include plural different mask generators and those plural mask generators may be organized into pipelined structures so as to be activated, in accordance with the graphics language, by the interpreter. Likewise, the background generator may include plural background generators, each of which generates a different kind of background, and each of those background generators may be organized into pipelined structures so as to be activated by appropriate graphics language commands through the interpreter. A coordinator may be provided for coordinating the pipeline among the structures. The mask objects generated by the mask generator may be primitive objects which are compact representations of the pixel objects, or the mask objects may be segments of bit map images. If desired, a clipper may be provided so as to clip the output of the mask generator. Likewise, in certain embodiments, a dispatcher may be employed so as to sequentially select raster areas and to cause the mask generator and the background generator and the rasterizer to operate on the selected raster areas, respectively. For example, the dispatcher may cause banded partitions of the raster to be sequentially selected and operated on by the mask generator and the background generator and the rasterizer.

In another aspect, the invention is a printing method comprising the steps of inputting coded print data such as

SUBSTITUTE SHEET
15 JUL 1998

PDL print commands, translating the coded print data into a first coded representation, translating the first coded representation into a second coded representation which is different from the first coded representation, and forming a raster image from the second coded representation. The first and second coded representations may include common code primitives, and the second coded representation may be susceptible of accurate modeling such that usage of the printer resources in rasterizing the representation may be predicted accurately so that the rasterization process may be changed should there be insufficient printing resources available.

This brief summary has been provided so that the nature of the invention may be understood quickly. A fuller understanding of the invention may be obtained by the following description in conjunction with the remaining drawings in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an illustration showing a rasterization of pixels in a bit map raster memory;

FIG. 2 is a depiction of a PDL printer process;

FIG. 3 is a printer according to the invention;

FIG. 4 depicts a graphics language processor according to the invention;

FIG. 5 is a diagram for illustrating the interface between the page description language and a graphics language according to the invention;

FIG. 5(*a*) is a diagram illustrating how a mask is applied against a background onto a raster;

FIG. 6 is a detailed view showing mask generators and background generators according to the graphics language processor of the invention; and

FIGS. 7, 8 and 9 are view showings different forms of the graphics language processor, in which FIG. 7 is a sequential processor, FIG. 8 is a banded processor, and FIG. 9 is a pipelined processor.

FIG. 10 is a flow diagram illustrating prediction and reallocation of printer resources.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 3 is a laser beam printing apparatus according to the invention.

As shown in FIG. 3, laser beam printer 10 includes a controller 11 which controls the various mechanical and

REPLACEMENT SHEET

15 JUL 1998

electrical functions within the printer. In particular, controller 11 accepts image data that is derived by an unshown host computer through communication port 12 and processes the image data so as to derive a laser modulation signal that is applied to laser driver 14. The image data from communication port 12 is preferably in a very compact form such as a page description language ("PDL") which merely defines objects and the objects' location on the page rather than defining a full raster image of the print image, but any form for the image data can used. In accordance with the image signal provided by controller 11, laser driver 14 drives laser 15 so as to cause it to emit a laser beam 16. A rotatable polygonal mirror 17 scanningly deflects laser beam 16 via mirror 19 onto the surface of photosensitive drum 20. Photosensitive drum 20 is rotated in the direction of arrow A past charger 21. The scanningly deflected laser beam 16 forms a latent image on the surface of the charged photosensitive drum, and the latent image is developed into a toner image at developer 22.

In coordination with scanning the surface of drum 20, controller 11 causes the topmost sheet of paper from stack 24 to be fed by pickup roller 25 and conveyed by rollers 26 and 27 through image passageway 29 to photosensitive drum 20. The toner-developed latent image on drum 20 is transferred to this sheet by transfer charger 30 and conveyed by rollers 31 and 32 to fixing device 34. In fixing device 34, a combination of heat and pressure fixes the toner image permanently on the sheet and the sheet is ejected by rollers 35 onto output tray 36.

FIG. 4 is a view showing controller 11. In FIG. 4, communication interface 202 and PDL interpreter 204 are substantially similar to the communication interface and PDL interpreter shown in FIG. 2, and descriptions thereof have been omitted.

As shown in FIG. 4, page description language is passed from PDL interpreter 204 to graphic language generation module 205. Graphic language generation module 205 translates PDL into a graphics language document according to the invention. In particular, in accordance with the PDL language, graphic language generation module 205 generates the graphics language document into two parts. First, the graphics language generation module 205 generates frame description commands in graphics language as shown at 206. The frame description commands describe raster frames as a sequence of primitive raster operations. Thus, the frame description commands define objects and rules on how to apply the objects onto the raster frames. For example, the frame description commands might instruct the graphic language processor to place a box-shaped set of pixels over a color image background and to position the resulting pixels at a specific location in bit map raster memory. Or the

REPLACEMENT SHEET
15 JUL 1998

frame commands might instruct the graphics language processor to form a corporate letterhead as a first frame, a text image comprising a letter as a second frame, and an image of a signature as a third frame. Frame description commands are described more fully below.

As further shown in FIG. 4, the second output of graphic language generation module **205** is document layout information **207**. Document layout information describes how to apply the frames defined by frame description commands **206** onto a bit map raster that corresponds to a document page. Specifically, the document layout information includes information as to how the frames are mapped onto the raster memory. For example, document layout commands might command the graphic language processor to map several frames onto different and possibly overlapping positions in the bit map raster for a page. In the case of printable documents, the document layout information may include specialized printing instructions such as duplex printing, and it may also include paper sheet handling and finishing instructions, such as stapling and collation instructions.

The graphic language commands **206** and the document layout information **207** are output on the pipelined outputs across interface **208**. Interface **208** may be a physical interface between discrete devices such as the interface between a host CPU and a print engine. For example, it is possible to implement communication interface **202**, PDL interpreter **204**, and graphic language generation module **205** as a printer driver in a host CPU. In that case, page description language commands that are generated in the host computer are also converted into graphic language commands **206** and document layout information **207** and are output across a pipelined interface directly to a printer engine. In the embodiment described here, however, interface **208** is an interface between two modules in controller **11**. The modules may be implemented either in software or in hardware.

Graphic language commands **206** are input to graphic language processor **209** which renders the graphic language commands into an empty raster that is provided from bit map raster pool **210**. The filled raster is provided to a print engine video interface **212** which provides the input to laser driver **14** of the printer.

Document layout information **207** is provided to system scheduler **214** which coordinates operation of the various modules. In particular, system scheduler **214** provides an empty raster from bit map raster pool **210** to the graphic language processor **209** and instructs the graphics language processor to begin rendering pixel images into the raster. When the graphic language processor **209** completes rendering the raster image, system scheduler **214** instructs print

REPLACEMENT SHEET
15 JUL 1998

engine video interface 212 to begin providing data to laser driver 14 in accordance with the rendered raster image. In synchronism with modulating the laser beam, system scheduler 214 instructs print engine module 215 to begin operation of the various mechanical aspects of printer 10, such as sheet feeding, drum rotation and roller fixing. System scheduler 214 also instructs duplex printing, paper output, stapling, collation and other document finishing information in accordance with the document layout information 207.

FIG. 5 is a detailed view of the graphic language processor 209. As seen there, the processor 209 includes a graphic language interpreter 220 which parses a stream of graphic language commands and interprets the graphics language commands. In particular, the graphic language interpreter stores graphic language objects 224 as those objects are encountered in the stream of graphic language commands. The interpreter also passes mask application commands to mask generator 221 and passes background generation commands to background generator 222 as those commands are encountered in the stream of graphics language commands. Mask generator 221 generates mask information such as mask M shown in FIG. 1; likewise, background generator 222 generates background information such as background B shown in FIG. 1. Both mask generator 221 and background generator 222 draw upon graphic language objects stored in GL object storage 224.

The masks generated by mask generator 221 are passed to clipper 225 which clips the masks, if desired, in accordance with the GL commands. Clipper 225 also draws upon graphic language objects stored in GL object storage 224.

The output of clipper 225 and the output of background generator 222 is provided to rasterOP machine 226. RasterOP machine 226 functions to apply the mask generated by mask generator 221, as clipped by clipper 225, onto the background generated by background generator 222. The rasterOP machine applies the mask in accordance with an application function defined by GL state memory 227. For example, as shown in FIG. 5A, an opaque application function has been specified. Accordingly, when the rasterOP machine applies the mask generated by generator 221 and clipped by clipper 225 onto the background plane generated by generator 222, any information in the initial raster is obliterated when forming the resulting raster. Other application functions are possible, for example, transparent application functions or additive application functions.

The raster on which the rasterOP machine 226 operates is provided from a bit map raster pool 229 as coordinated by system scheduler 214. When all applications have been finished, as set forth in the document layout information 207, the rasterOP machine provides a filled raster back to the

REPLACEMENT SHEET
15 JUL 1998

bit map raster pool and the filled raster is printed by printer **10**.

FIG. **6** is a detailed conceptual view showing the construction of mask generator **221** and background generator **222**, as well as showing the application process of rasterOP machine **226**. In FIG. **6**, some of the elements shown in FIG. **5** have not been depicted so as to simplify the following presentation.

As shown in FIG. **6**, mask generator **221** includes several mask generators that have been designed specifically to produce specialized pixel primitive objects. In particular, mask generator **221** includes a bit map mask generator **221***a*, a trapezoidal mask generator **221***b*, a line generator **221***c*, a run length mask generator **221***d*, a box mask generator **221***e*, and a generator **221***f* that generates other compact representations. In more detail, bit map mask generator **221***a* produces a bit map mask in response to GL commands from GL interpreter **220**. Bit map masks are bit-for-bit pixel representations of a mask.

Trapezoidal mask generator **221***b* generates a trapezoidal-shaped mask in response to GL commands from GL interpreter **220**. Trapezoidal masks define trapezoidal objects in raster memory.

Line generator **221***c* generates line drawings of specified thickness in response to GL commands from GL interpreter **220**.

Run length mask generator **221***d* encodes run length masks based on the run length of pixels within the mask.

Box mask generator **221***e* generates a box-shaped bask of specified width and height in response to commands from GL interpreter **220**.

Other compact representations are possible, and the above set should not be viewed as limiting, but rather only as representative of the types of masks generated by mask generator **221**.

Background generator **222** includes color background generator **222***a*, tile background generator **222***b*, and image background generator **222***c*.

In more detail, color background generator **222***a* responds to GL commands from GL interpreter **220** and generates a solid color background of specified color.

Tile background generator **222***b* generates and replicates a tile specified by GL interpreter **220**. The tile pattern is replicated across the entire background in the area specified by the GL interpreter.

Image background generator **222***c* provides an image background in a pixel-for-pixel format as specified by GL interpreter **220**.

REPLACEMENT SHEET
15 JUL 1998

Other background generators are possible, and the above list should be viewed not in a limiting sense, but rather only as representative of the type of background generators that may be included in background generator 222.

FIG. 6 also illustrates how a current raster 230 is updated by rasterOP machine 226 in response to graphic language commands so as to produce updated raster 231. As shown in FIG. 6, GL interpreter 220 has received the graphic language command "apply clipped box (W,H) at <x,y> with image background". (A full list of graphic language commands is given in Appendix A.) In response to the first portion of the graphic language command, namely "apply clipped box (W,H)", the GL interpreter 220 activates box mask generator 221*e*. Box mask generator 221*e* generates a box-shaped mask of width W and height H as depicted illustratively at the output of mask generator 221. Thereupon, clipper 225 draws upon the graphic language objects stored in memory 224 to provide a clipped shape as depicted illustratively adjacent clipper 225. The resulting clipped box is provided to rasterOP machine 226.

In response to the later portion of the graphic language command, namely "with image background", GL interpreter 220 activates image background generator 222*c*. The image background generator 222*c* draws upon the graphic language objects stored in memory 224 and provides the image background, which may be a color image background, to rasterOP machine 226.

In response to the information provided by clipper 225 and background generator 222, rasterOP machine 226 applies the clipped mask to the background in accordance with the application function specified by state memory 227. Here, the application function is specified by "opaque", in accordance with which the rasterOP machine 226 applies the masked and clipped area of the background at position <x,y> of the current raster so as to obtain the updated raster shown at 231. The foregoing steps are performed for each command in the stream of graphic language commands for each frame, after which the document layout information 207 is utilized to determine how the frame is applied to the overall bit map raster.

FIG. 6 has been explained as if the process were a sequential process, as depicted in FIG. 7. That is, in step 1, mask generator 221 generates a mask that is provided to clipper 225. In step 2, clipper 225 clips the mask appropriate and provides it to rasterOP machine 226. In step 3, background generator 222 provides a background to rasterOP machine 226, and in step 4, rasterOP machine 226 applies the mask to the background and writes it into the raster image in the bit map raster memory. While this is one possible implementation of the graphic language processor, other implementations are possible.

REPLACEMENT SHEET

15 JUL 1998

Thus, as shown in FIG. 8, dispatcher 330 is provided to coordinate the activities of the mask generator 321, clipper 325, background generator 322 and rasterOP machine 326. Specifically, dispatcher 330 instructs each of these modules to operate on a specific area of the raster image, such as area 330*a*. Typically, such an area is a band in a partitioned and banded area of bit map memory, but this is not necessarily the case, and the area may be any arbitrary area in the image. Dispatcher 330 serially sequences through the entire area, for example, from band to band, until the entire page in bit map memory has been rendered. Thus, dispatcher 330 causes mask generator 321, background generator 322, clipper 325 and rasterOP machine 326 to operate on sequential areas of bit map memory but to operate on only a single area at one time. Graphic language objects that are specified for other areas are temporarily ignored and are processed only at a later time when dispatcher 330 indicates that area should now be processed. Because of this, memory overhead may be necessary for the FIG. 8 structure to store temporarily the graphic language commands. The overhead may be minimized by insuring that the areas sequentially selected by dispatcher 330 are large enough so that the amount of graphic language commands that need to be stored is minimized.

FIG. 9 shows another implementation of the graphic language processor in which a coordinator 440 is provided to coordinate a pipeline between mask generator 421, background generator 422, clipper 425 and rasterOP machine 426. Specifically, in the implementation shown in FIG. 9, each of the modules is a pipelined module designed to operate on whatever information is currently first available in its pipeline. Each module operates on its own pipelined instruction in parallel and at the same time as described generally in "Computer Architecture: A Quantitative Approach", D. A. Patterson and J. L. Hennessy, Chapter 6, Morgan Kaufman Publishing Company (1990). As such, coordinator 440 is provided to ensure that the modules start in synchronism and remain in synchronism throughout operation. Thus, for example, whereas background generator 422 will provide a background directly to rasterOP machine 426, mask generator 421 is not able to provide a corresponding mask object directly to the rasterOP machine because clipper 425 is interposed between mask generator 421 and rasterOP machine 426. Thus, coordinator 440 ensures that the clipped mask which is output by clipper 425 is associated with the proper background output from background generator 422 so that the rasterOP machine 426 operates on associated pieces of information.

The graphics language processor described hereinabove may be combined with other printer techniques such as that described in our co-pending application Ser. No. 07/898,532

REPLACEMENT SHEET
15 JUL 1998

filed Jun. 15, 1992, the contents of which are incorporated herein by reference as if set forth in full. In combination with the techniques described in that application, it is possible to translate the graphic language commands described herein into a further DART representation. The DART representation is susceptible of accurate modeling of printer resources whereby it is possible to determine before a printing operation commences whether there are sufficient printer resources to complete the printing operation. If there are sufficient resources available, then printer commences; on the other hand, if there are insufficient printer resources then the printer operation may be changed so as to ensure that the printing operation can be completed.

FIG. **10**, which is taken from 6(*a*) in the aforementioned application Ser. No. 898,532, depicts this process in more detail. The process steps in FIG. **10** are stored in an unshown memory in controller **11**.

At step **S601**, controller **11** inputs image data from communication port **12**. The image data is typically generated by an unshown host computer which sends the image data to controller **11** in a compact code such as a page description language.

At step **S602**, the PDL image data is translated into GS image data. At step **S603**, generator **42** generates a DART representation from the GS data and stores it in memory. As the DART representation is generated, it is banded in step **S604**, and in step **S605** a running tally is maintained for resource usage that will be required in rendering the DART representation into a full raster image. Specifically, for each of the different DART representations, predictors are provided that predict (a) CPU power required to render the DART representation, and (b) memory required to store the DART representation. If printer **10** were equipped with external memory via a bandwidth-limited communication line, then a third predictor would be provided as to the bandwidth requirements for each DART representation.

The resources described here, namely CPU power, memory and bandwidth, are representative of the types of resources that printer **10** may be provided with. For each additional resource, corresponding predictor functions are likewise provided and used to tally resource usage.

Step **S606** determines, based on the tally from the predictor functions, whether resource usage will be exceeded in rendering the DART representation for the current band in a full raster image in the band rasters. Step **S606** may be executed periodically during DART generation, e.g. after each DART representation is generated, or it may be executed after the DART representation for the full band is generated. In the case of determining whether memory usage

REPLACEMENT SHEET
15 JUL 1998

is exceeded by the DART representation, it is preferred to execute step S606 periodically, for example, every time a new DART object is created.

If step S606 determines that resource requirements to render the current band can be met, then flow advances to step S607 which determines whether there is additional GS code for the current page. If there is more GS code for the page, then flow returns to step S604 until the remaining image data has been processed.

When DART representations for all bands in the page have been generated, flow advances from step S607 to step S608 where a pair of blank band rasters are appended to the DART representation stored in memory. If flow has advanced to this step, then it is known that the resources available in the printer are adequate to render the DART representation band-by-band into a full raster image without the danger of exceeding printer resources, e.g. there is enough CPU power to avoid data underflow to the print engine.

In step S610, the DART representations are rendered band-by-band into alternating ones of the pair of band rasters, as described above. Then in step S611, the raster image is shipped to the print engine. Steps S610 and S611 are executed concurrently and sequentially so that as one band of raster image is shipped from one band raster, rendering of the next band of DART representation into the other band raster is completed.

It is also possible to provide more than two blank band rasters, in which case rendering of a fixed number of bands is completed.

After the current page has been printed, or as CPU time permits during rendering and shipping in steps S610 and S611, processing of the next page begins (step S612) in accordance with the overall flow shown in FIG. 10.

If step S606 determines that there are insufficient resources available in the printer, it is known that the resources needed to render or store the current DART representation exceed the resource envelope of the printer. Flow then advances to step S613 which performs resource trade-off or re-allocation in an attempt to fit the DART representation into the printer resource envelope. Initially, a different DART representation is selected for the offending band. For example, if the memory resource has been exceeded, then a memory-intensive DART representation may be replaced by a different representation which uses comparatively less memory. Likewise, if the CPU resource has been exceeded, then a CPU-intensive DART representation may be replaced by a different representation which uses comparatively less CPU. For example, the offending band may be pre-rasterized and stored in memory so that it

REPLACEMENT SHEET
15 JUL 1998

is ready for printing by the print engine without any further CPU processing.

In step S614, resource usage with the different DART representations is re-predicted. If resource usage now falls within the resource envelope then flow proceeds to step S607 and continues as described above. If resource usage is still exceeded, then further attempts may be made to bring resource usage within the printer envelope (steps S613 and S614).

Alternatively, or in the case where it is not possible for the DART representation to be brought within the resource envelope, then the image is degraded prior to printing. Degrading automatically reduces resource requirements. In the example given here, the image is degraded by reducing the resolution (or pixel density) from 600 dpi to 300 dpi. Image degradation may take other forms: for a halftone image, degradation may also include reducing the number of grey levels; for a color image, degradation many also include reducing the number of colors.

In step S616, in preparation for resolution degradation processing is completed for all previous pages, that is, DART representations for prior pages are rendered and shipped as shown at steps S610 and S611.

In step S617, image resolution is degraded by rendering the DART representations that have already been generated. The DART representations are rendered in situ in memory but at the degraded resolution. Then, GS that has not already been processed is converted to a degraded raster image. The conversion is either direct into the degraded raster image or indirectly by first generating a DART representation and then rendering.

In step S618, the degraded raster image is shipped to the print engine. Preferably, when degradation is necessary, it should be complemented with some form of image enhancement so as to minimize the effects of degradation. For example, in the case of resolution degradation, smoothing circuitry such as that described in U.S. Pat. No. 4,878,068 may be employed at the time the degraded image is shipped so as to soften the appearance of the printed degraded image.

In step S619, processing for the next page begins but without degradation.

Steps S613 through S619 provide repetitive attempts to reallocate resource usage so that sufficient resources are available for the DART representation. In its simplest form of resource trade-off, however, and in the embodiment presently preferred, resource trade-off takes the form of immediate image degradation when it is determined that resources have been exceeded, whereby the color, grey scale or pixel density (or resolution) is degraded. In the case of a

REPLACEMENT SHEET
15 JUL 1998

simple black and white image which is being rendered into a 600 dpi raster image, for example, resolution can be degraded to 300 dpi and a full raster image can be stored in memory. In this instance, the DART representation is converted in situ into a degraded- but-full raster image of the image data, whereupon the full raster image is shipped to the print engine. If desired, image smoothing techniques can be applied, such as those described in U.S. Pat. No. 4,878,068, so as to improve the appearance of the lower resolution image.

REPLACEMENT SHEET
15 JUL 1998

Appendix A

GRAPHIC LANGUAGE COMMANDS INSTRUCTION SET

This appendix is a summary of the GL commands instruction set. It defines instructions in terms of their effects on the system and semantic actions.

The notation below describes the stack manipulation the graphic processor performs when it executes an instruction:

Instructionarg_1 arg_2 ... arg_n Instruction $result_1$ ... $result_k$

where the sequence arg_1 arg_2 ... arg_n denotes the argument list on the stack (with arg_n at the top) that the instruction consumes. Execution of the instruction replaces the argument list with the sequence $result_1$ ... $result_k$.

ARITHMETIC AND RELATIONAL OPERATIONS

Use these instructions to execute arithmetic and relational operations on the system stack.

add *number*$_1$ *number*$_2$ gsop.add number (= *number*$_1$ + *number*$_2$)

Addition.

sub *number*$_1$ *number*$_2$ gsop.sub number (= *number*$_1$ - *number*$_2$)

Substraction.

REPLACEMENT SHEET
15 JUL 1998

mul *number*$_1$ *number*$_2$ gsop.mul number (= *number*$_1$ * *number*$_2$)

Multiplication.

div *number*$_1$ *number*$_2$ gsop.div number (= *number*$_1$ /*number*$_2$)

Division (integer).

mod *number*$_1$ *number*$_2$ gsop.mod number (= *number*$_1$ mod *number*$_2$)

Modulus.

neg *number* gsop.neg number (= -*number*)

Arithmetic negate.

shright *number*$_1$ *number*$_2$ gsop.shright number (= *number*$_1$ >> *number*$_2$)

Shift right.

shleft *number*$_1$ *number*$_2$ gsop.shleft number (= *number*$_1$ << *number*$_2$)

Shift left.

OTHER ARITHMETIC/RELATIONAL INSTRUCTIONS

GL commands support a number of extended operations, including vector arithmetic (using fixed coordinates) and the necessary fixed arithmetic trigonometric functions (fast approximations).

STACK MANIPULATION

These operations handle the system stack.

REPLACEMENT SHEET
15 JUL 1998


```
loadq_0    gsop.loadq_0          num(=0)
           loadq_1gsop.loadq_1   num(=1)
           loadq_2gsop.loadq_2   num(=2)
           loadq_3gsop.loadq_3   num(=3)
           loadq_4gsop.loadq_4   num(=4)
           loadq_5gsop.loadq_5   num(=5)
           loadq_6gsop.loadq_6   num(=6)
           loadq_7gsop.loadq_7   num(=7)
```

Use the loadquick family of instructions to load very quickly the system stack with the most commonly used arithmetic values.

pop *anyvalue* gsop.pop

Discards the top of the stack.

dup *anyvalue* gsop.dup *anyvalue anyvalue*

Duplicates the top of the stack.

exch *anyvalue*$_1$ *anyvalue*$_2$ gsop.exch *anyvalue*$_2$ *anyvalue*$_1$

Exchanges the top two cells in the stack.

OTHER STACK MANIPULATION INSTRUCTIONS

GL commands support a number of possible extension operations for handling the stack, including indexing, copying, etc.

RECORDS

Use records to store permanent information. The information in a record is structured as an array. Graphic language commands cause records to be stored as GL objects in GL object storage (see below).

The concept of a current record serves as a mechanism to compress GL code: Once an object is

REPLACEMENT SHEET
15 JUL 1998

declared the current record, all load and store operations refer to that object.

At the beginning of each frame, the system provides a default record.

record *object|d|* gsop.record

The object *object|d|* becomes the current record.

load *displ* gsop.load

Pushes the contents of memory at record location *displ* onto the stack.

store *value displ* gsop.store

Pops *value* on top of stack into record location *displ*.

ARRAY HANDLING

These instructions create arrays and access array elements on the stack.

OBJECT HANDLING

These instructions create and manipulate GL objects.

objdelete *objHandle* gsop.objdelete

Removes the object *handle* from memory.

objmake *objHandle size* gsop.objcopy

Performs a "delete" on the object *objHandle*, then creates a new object of size *size*.

REPLACEMENT SHEET
15 JUL 1998

markarraygsop.markarray + n + b_1 + ... + b_n
inlineHandle

markbitmapgsop.markbitmap + n + b_1 + ... + b_n
inlineHandle

marktrap gsop.marktrap + n + b_1 + ... + b_n
inlineHandle

markproc gsop.markproc + n + b_1 + ... b_n
inlineHandle

These instructions declare an *inline* object, leaving an *inlineHandle* handle on the top of the stack for future reference.

objdef *inlineHandle objHandle* gsop.objdef

Performs "delete" on the object *objHandle*, creates a new object of specified size, then copies the contents of *inlineHandle* onto the new object.

CONTROL

These instructions control the flow of execution for the GL processor.

call *executableArray* gsop.call

Invokes the procedure denoted by the handle *executableArray*.

return gsop.return

Returns executable flow back from the procedure currently executing. If the GL processor is not executing a procedure, this instruction finishes the current frame.

REPLACEMENT SHEET
15 JUL 1998

if *bool* execArray$_{true}$gsop.if

ifelse *bool* execArray$_{true}$ execArray$_{false}$gsop.ifelse

These operations conditionally execute the corresponding procedure according to the value of *bool* (0 means false, non zero means true).

loop *number execArray* gsop.loop

The *execArray* is executed *number* times. A negative number produces an infinite loop.

break gsop.break

continue gsop.continue

Breaks or continues within the currently executed loop.

switch *number* gsop.switch --

Switches to the program segment corresponding to the case value *number*. Case values are limited to the set *B={0..255}*

CONTEXT HANDLING

context *contextNumber* gsop.context --

Sets the active context to *contextNumber*. There are a fixed number of possible contexts.

contexDef*executableArray contextNumber*
gsop.contextDef

REPLACEMENT SHEET
15 JUL 1998

Defines context *contextNumber* as per procedure *executableArray*. Context procedures contain a switch statement that is used to map codes onto actions.

PROCEDURE DEFINITIONS

Procedures are simply executable arrays. A procedure handle is an array handle or an object handle.

SYSTEM SERVICES ACCESS

System services are accessed just like normal procedures, except that system procedure handles are part of a distinguished set.

MASK GENERATION

These instruction activate the primitive mask generators provided by GL.

Before activating any of the generators, a GL program must set the state information corresponding to the generator parameters.

PixelBit *bitmapHandle x y* gsop.PixelBit

Performs a pixelbit operation of *bitmapHandle* at position <*x,y*>.

Trapezoid*trapezoidHandle* gsop.Trapezoid

Activates the Trapezoid Mask Generator.

RLE gsop.RLE

Activates the Run Length encoded Mask Generator.

REPLACEMENT SHEET
15 JUL 1998

LINE GENERATION MASKS

These instructions activate the mask generators for line rendering.

These masks generators use GL state variables to implicitly define position, line width, etc.

ACCESSING THE INPUT STREAM

Members of this class of primitives access the input stream, to create objects in the stack and/or the array stack.

getc1	gsop.getc1 num (= b1)
getc2	gsop.getc2 num (= (b1 << 8) + b2)
getc3	gsop.getc3 num (= (((b1 << 8) +b2) << 8) + b3)
getc4	gsop.getc4 num (= (((((b1 << 8) + b2) << 8) + b3) << 8) + b4)

Gets next 1 to 4 bytes (denoted b1, b2, b3 and b4) from the input stream and pushes them onto the stack as a scalar integer value.

BACKGROUND GENERATION

image	a b c d e f image gsop.Image

Sets image as the background image array. The transformation [a b c d e f] denotes the mapping of raster pixels onto image array elements.

image	0 gsop.Image

REPLACEMENT SHEET
15 JUL 1998

Resets the background image array as empty.

gray *grayLevel* gsop.gray

Sets the background to gray, of level *grayLevel*.

bgtile *bitmapHandle* gsop.bgtile

Sets the background to be tiled by *bitmapHandle*.

bgmodify *offsetX offsetY rotation* gsop.bgtile

Modifies the background tile by shifting it <*offsetX, offsetY*> and rotating it by the amount specified in *rotation*.

STATE OPERATIONS

screen *objHandle* gsop.screen

Sets the dithering screen of *objHandle*.

clip *trapHandle* gsop.clip

Adds *trapHandle* to the clipping region.

clip 0 gsop.clip

Zeros the clipping region.

PRAGMATICS

The GL Interpreter ignores this class of instructions; pragmatics are for the use of other

REPLACEMENT SHEET
15 JUL 1998

subsystems and/or tools (Debug module, Generation Libraries, etc.)

pragma gsop.pragma + {n} + b_1 + b_2 + ... + b_n

The GL Interpreter ignores the (n+2) bytes comprising this instruction. Use pragma as a comment, a synchronization tag for debugging purposes, a grouping tag, etc.

REPLACEMENT SHEET
15 JUL 1998

What is claimed is:

1. A printing method comprising the steps of:

inputting coded print data;

translating the coded print data into a first coded representation;

translating the first coded representation into a second coded representation which is different from the first coded representation;

predicting printer resources consumed in forming a raster image corresponding to the coded print data, said predicting step predicting consumption of printer resources based on the second coded representation;

re-allocating printer resources in a case where printer resources are predicted by said predicting step to be exceeded; and

forming a raster image from the second coded representation.

2. A method according to claim 1, wherein the coded print data is page description language print data and wherein the first and second coded representations are intermediate in complexity to the page description language and the raster image.

3. A method according to claim 1, wherein the first and second coded representations include plural coded primitives, and wherein all of the coded primitives of the second coded representation are the same as at least some of the coded primitives of the first coded representation.

4. A method according to claim 1, wherein the second coded representation is susceptible of accurate modelling such that usage of printer resources in said step of forming a raster image may be predicted accurately and wherein the first coded representation is not susceptible of such accurate modelling.

5. A printing apparatus comprising:

inputting means for inputting coded print data;

first translating means for translating the coded print data into a first coded representation;

second translating means for translating the first coded representation into a second coded representation which is different from the first coded representation;

predicting means for predicting printer resources consumed in forming a raster image corresponding to the coded print data, said predicting means predicting consumption of printer resources based on the second coded representation;

re-allocating means for re-allocating printer resources in a case where printer resources are predicted by said predicting means to be exceeded; and

REPLACEMENT SHEET
15 JUL 1998

forming means for forming a raster image from the second coded representation.

6. An apparatus according to claim 5, wherein the coded print data is page description language print data and wherein the first and second coded representations are intermediate in complexity to the page description language and the raster image.

7. An apparatus according to claim 5, wherein the first and second coded representations include plural coded primitives, and wherein all of the coded primitives of the second coded representation are the same as at least some of the coded primitives of the first coded representation.

8. An apparatus according to claim 5, wherein the second coded representation is susceptible of accurate modelling such that usage of printer resources by said forming means may be predicted accurately and wherein the first coded representation is not susceptible of such accurate modelling.

REPLACEMENT SHEET
15 JUL 1998

ABSTRACT

METHOD AND APPARATUS FOR PRINTING ACCORDING TO A GRAPHIC LANGUAGE

Printing in which printer resources that are consumed in forming a raster image are predicted and in which those printer resources are re-allocated in a case where printer resources are predicted to be exceeded. Specifically, coded print data such as coded print data in a page description language is input, the coded print data is translated into a first coded representation, and the first coded representation is translated into a second coded representation which is different from the first coded representation. Printer resources consumed in forming a raster image corresponding to the coded print data in the second coded representation are predicted, and printer resources are re-allocated in a case where printer resources are predicted to be exceeded. A raster image is thereafter formed from the second coded representation.

Figure 10

REPLACEMENT SHEET
15 JUL 1998

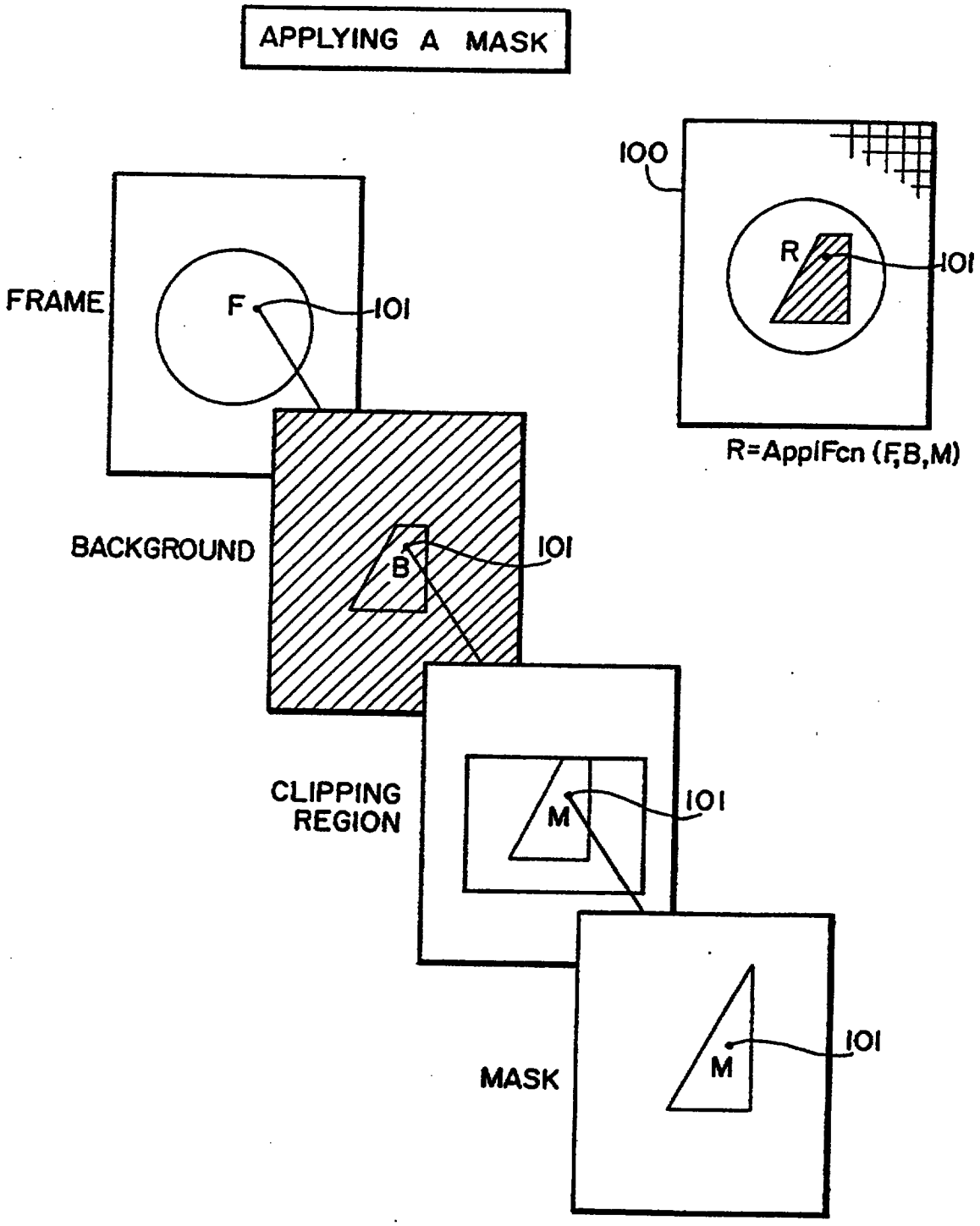


FIG. 1

REPLACEMENT SHEET
15 JUL 1998

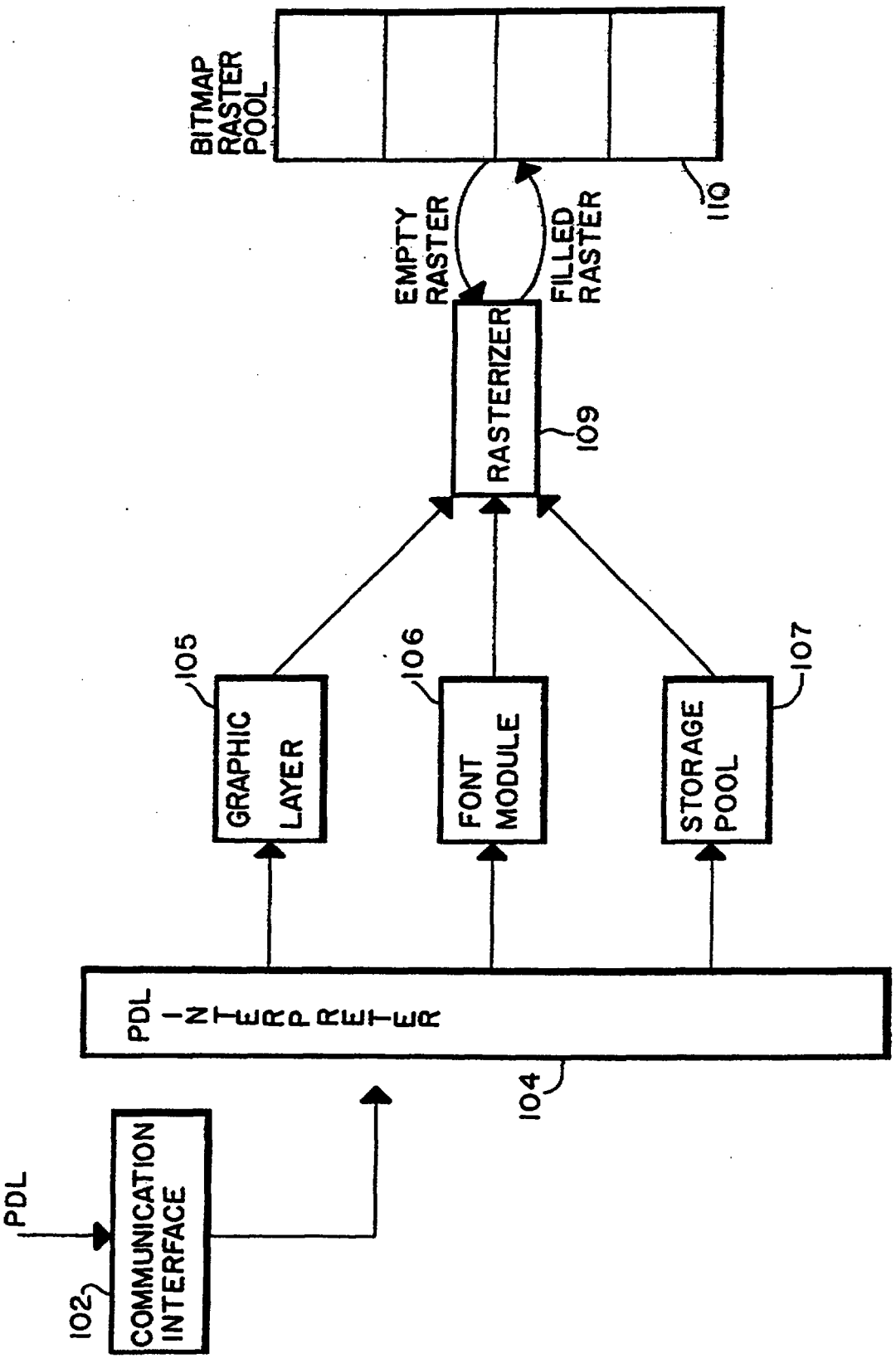


FIG 2

REPLACEMENT SHEET
15 JUL 1998

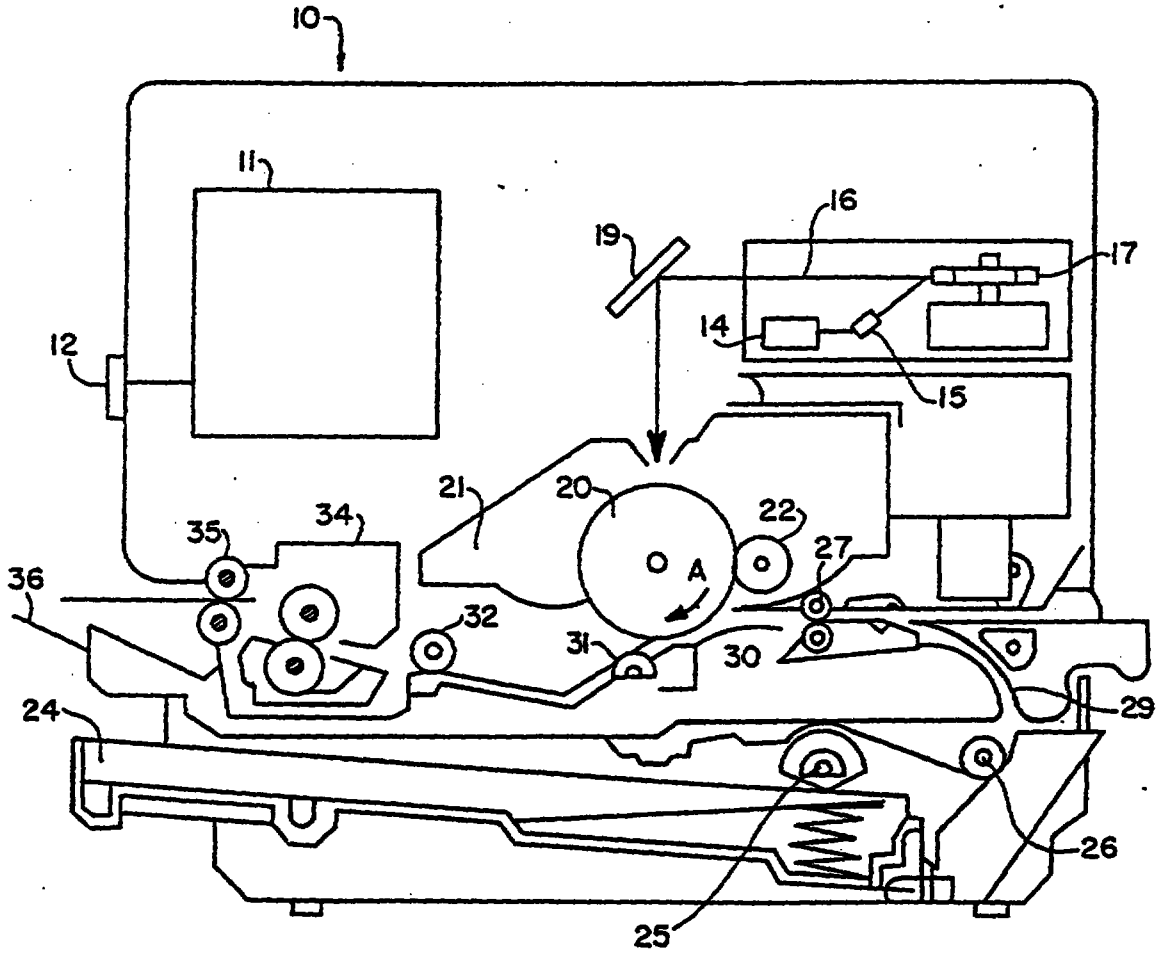


FIG.3

REPLACEMENT SHEET
15 JUL 1998

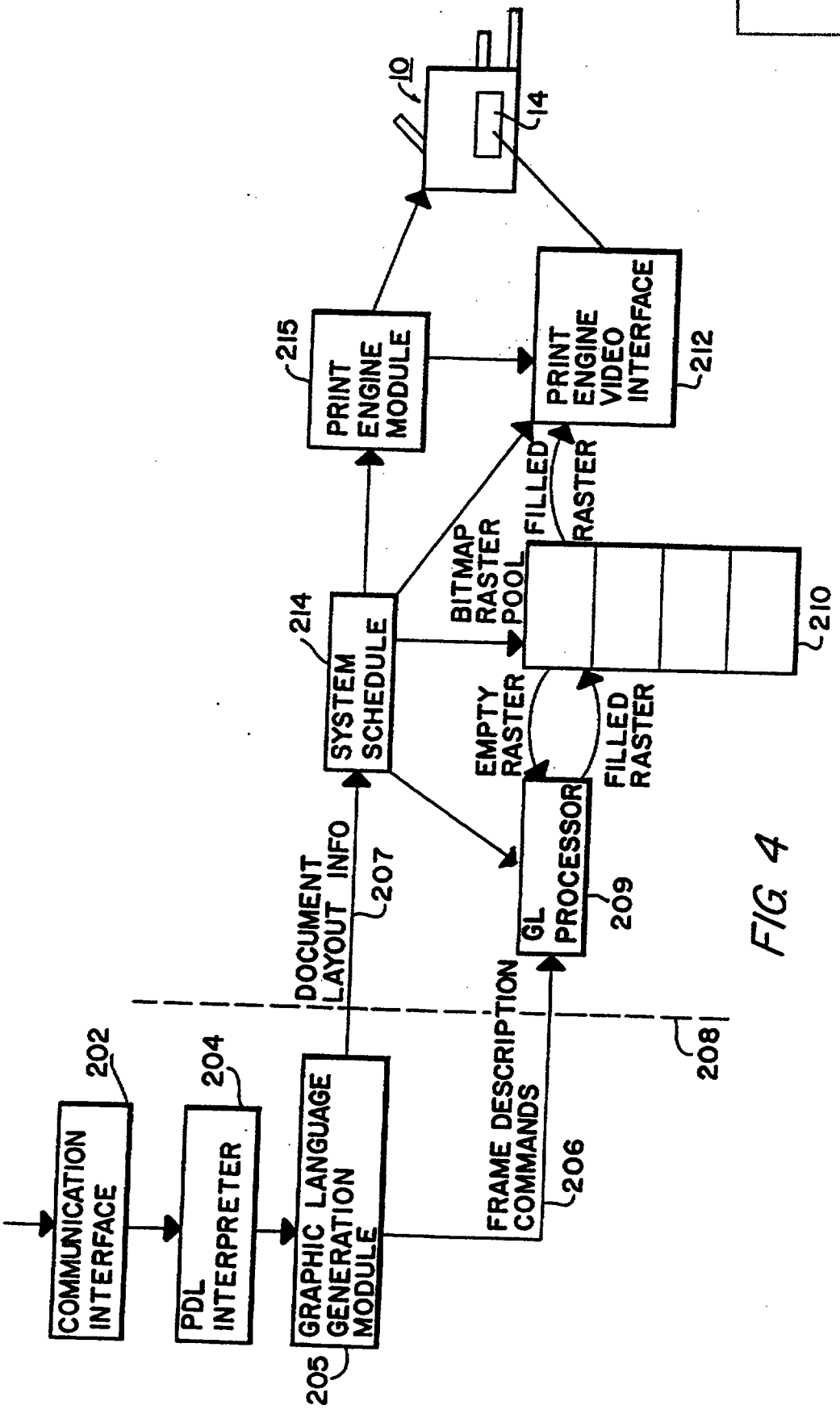


FIG 4

REPLACEMENT SHEET
15 JUL 1998

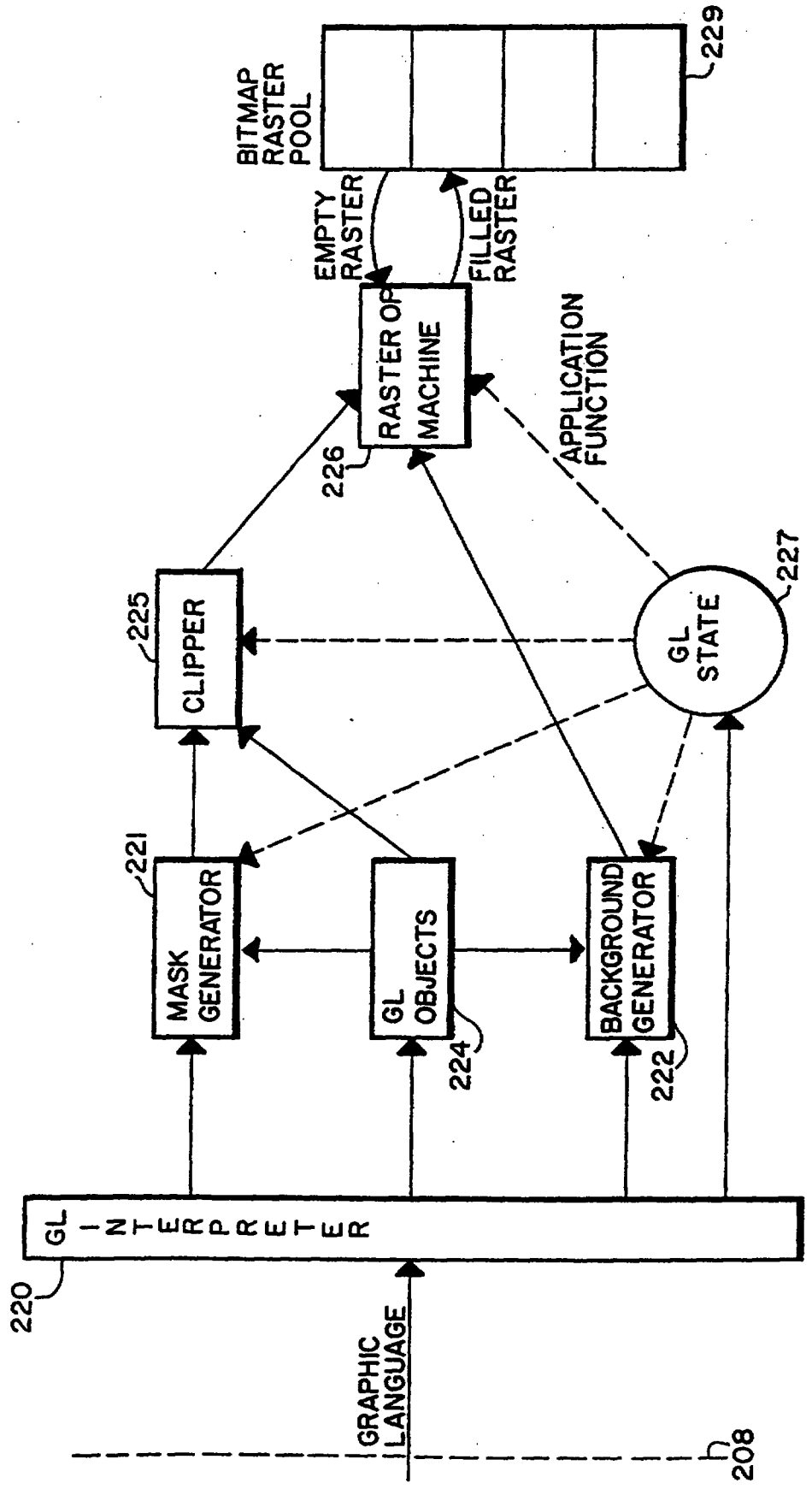


FIG 5

THIS SHEET HAS BEEN REPLACED

REPLACEMENT SHEET
15 JUL 1998

This Sheet Has Been Replaced

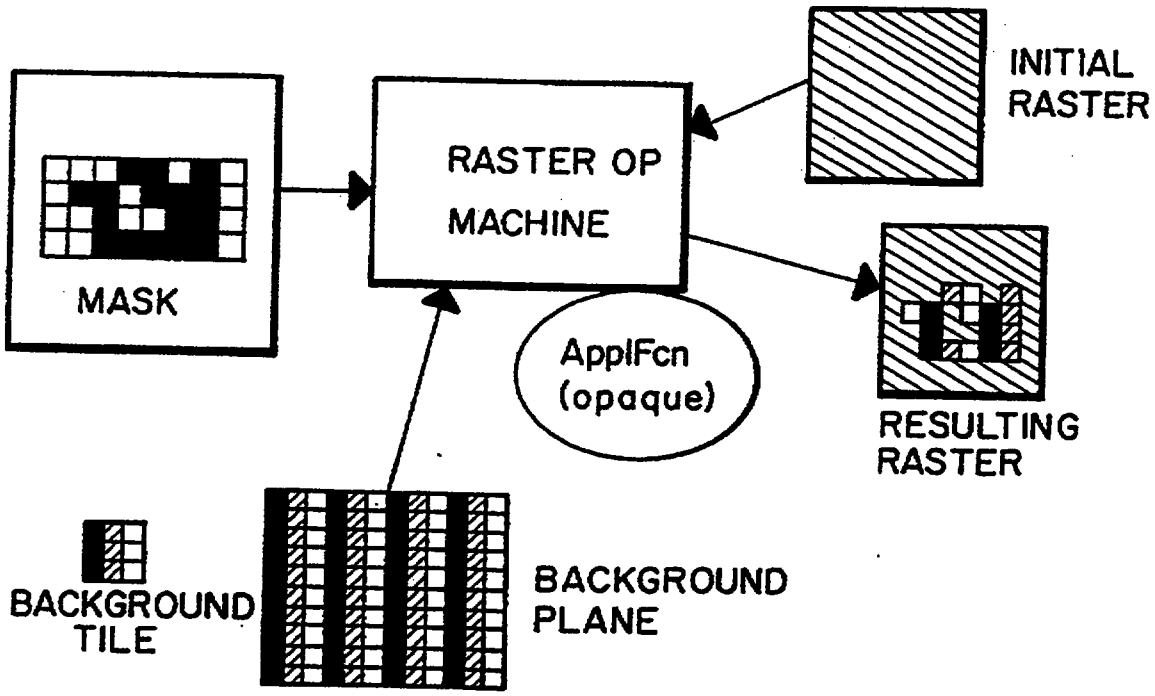


FIG. 5(a)

REPLACEMENT SHEET
15 JUL 1998

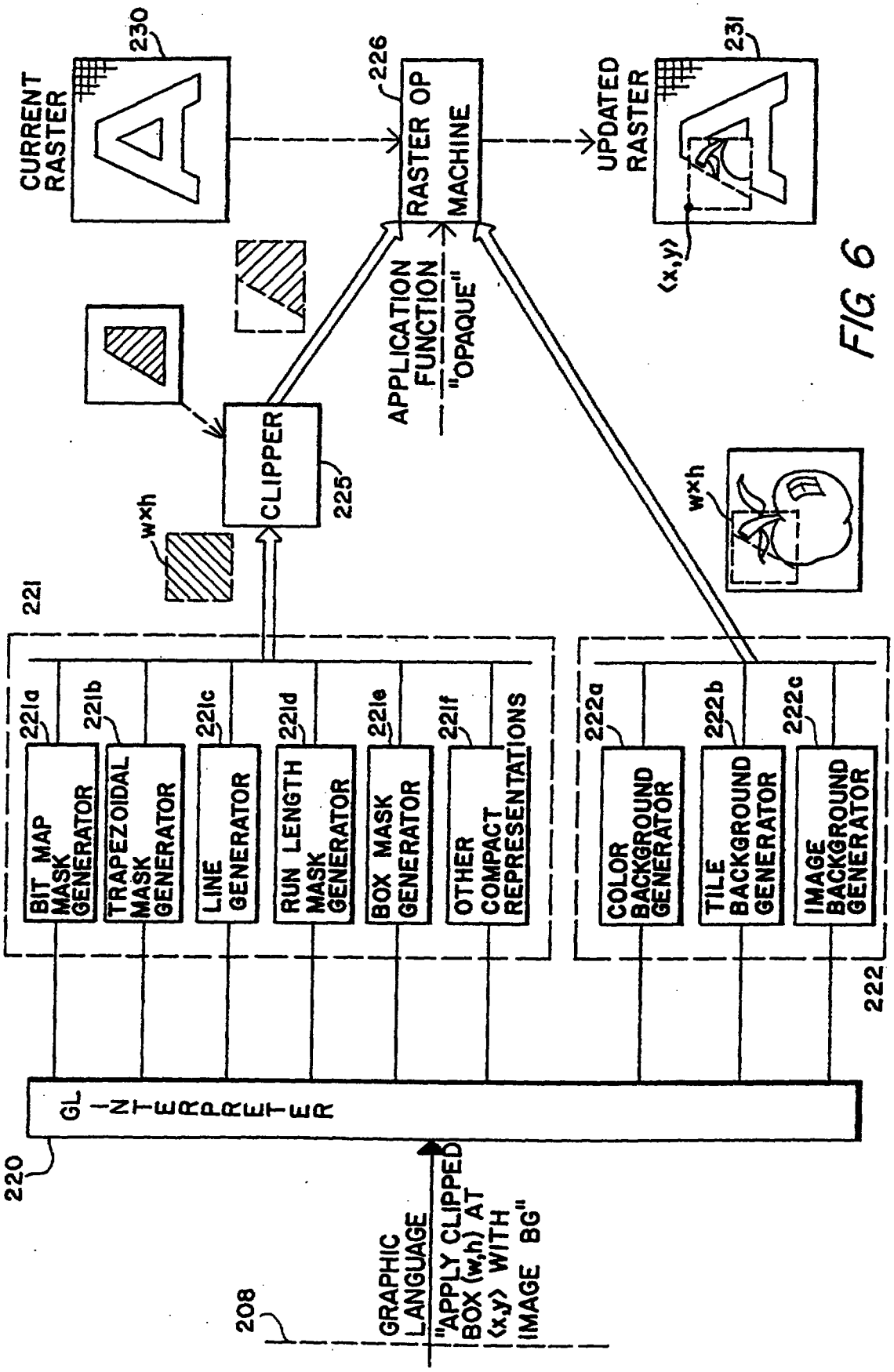


FIG 6

REPLACEMENT SHEET
15 JUL 1998

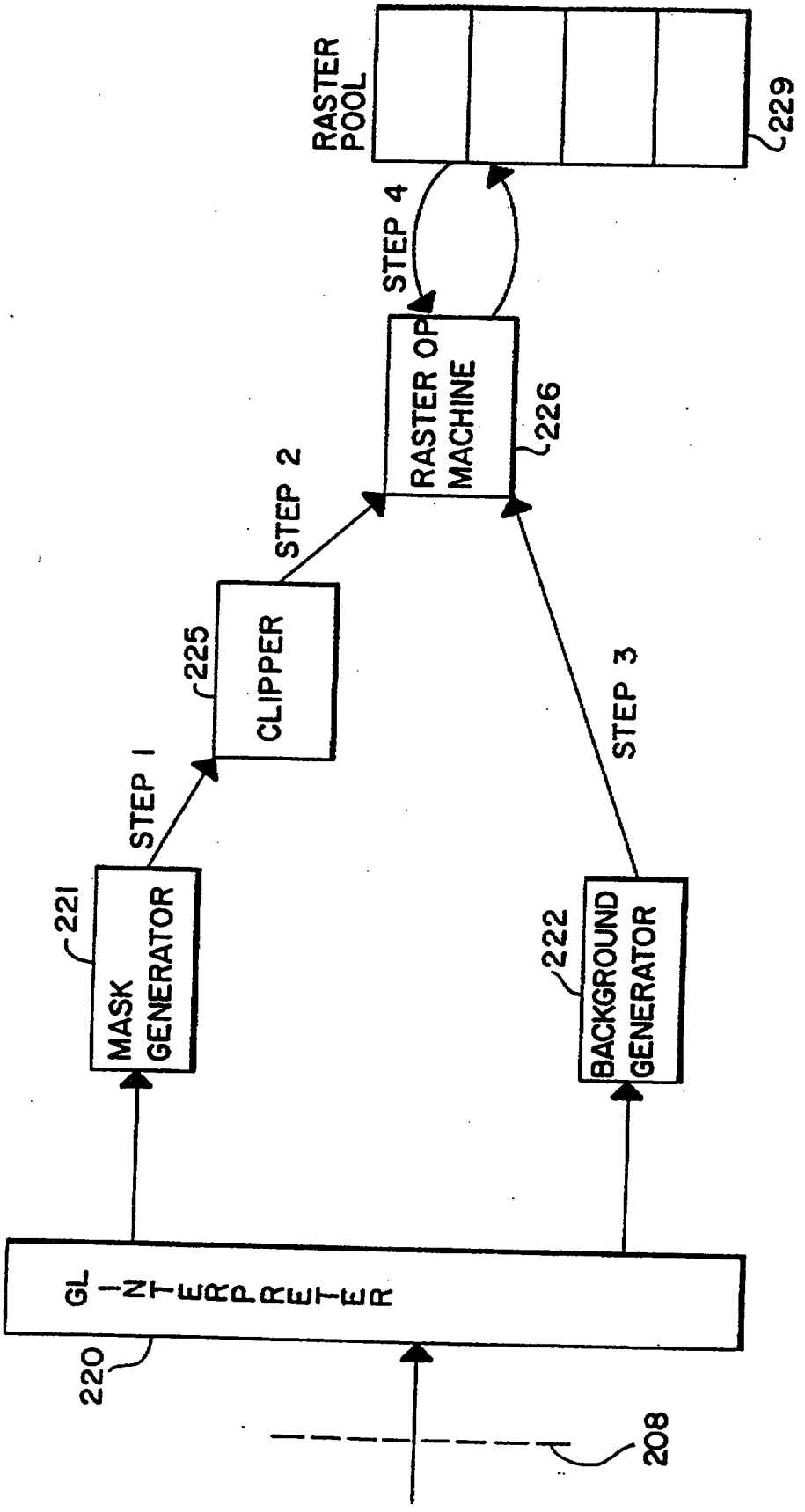


FIG 7

REPLACEMENT SHEET
15 JUL 1998

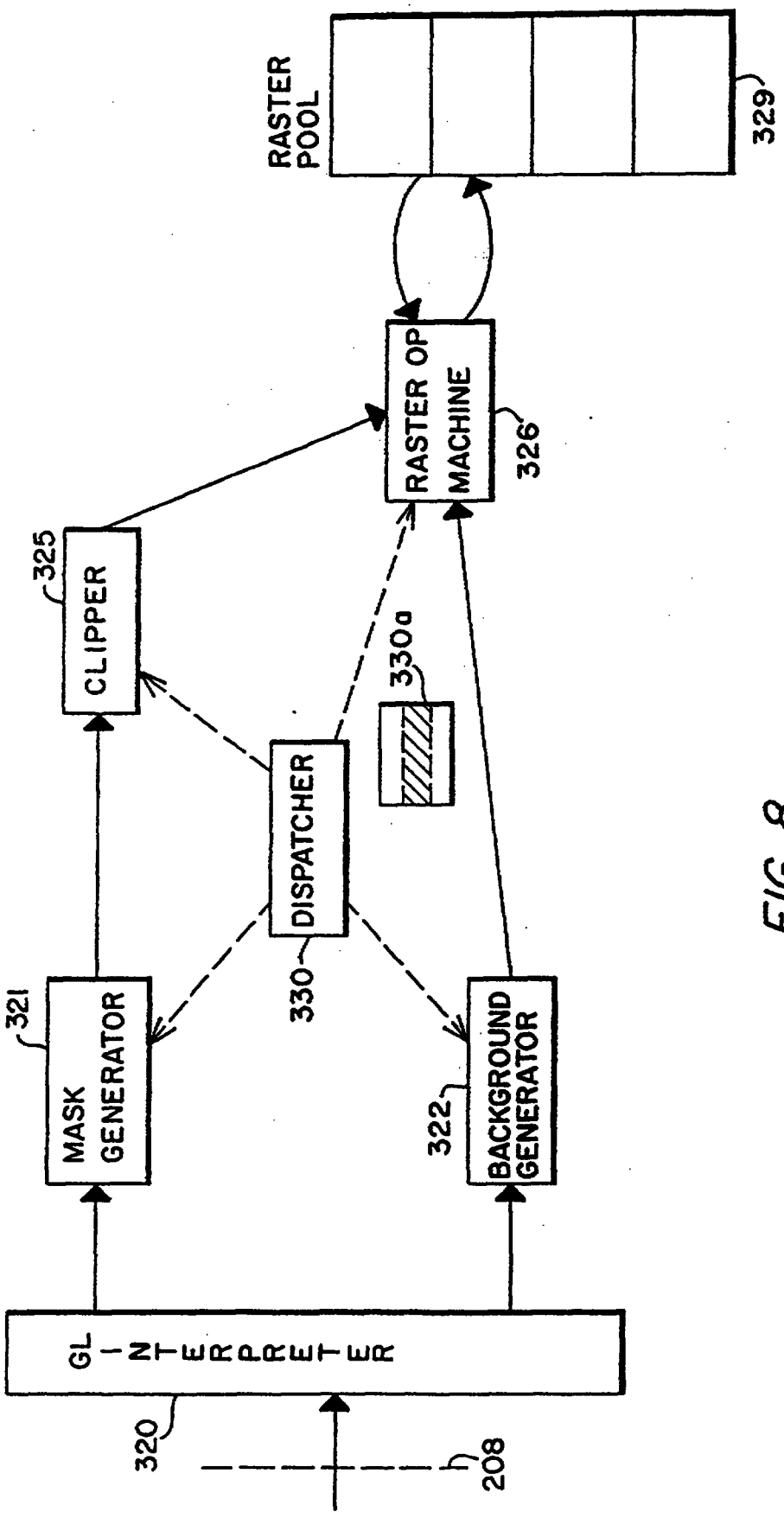


FIG 8

REPLACEMENT SHEET
15 JUL 1998

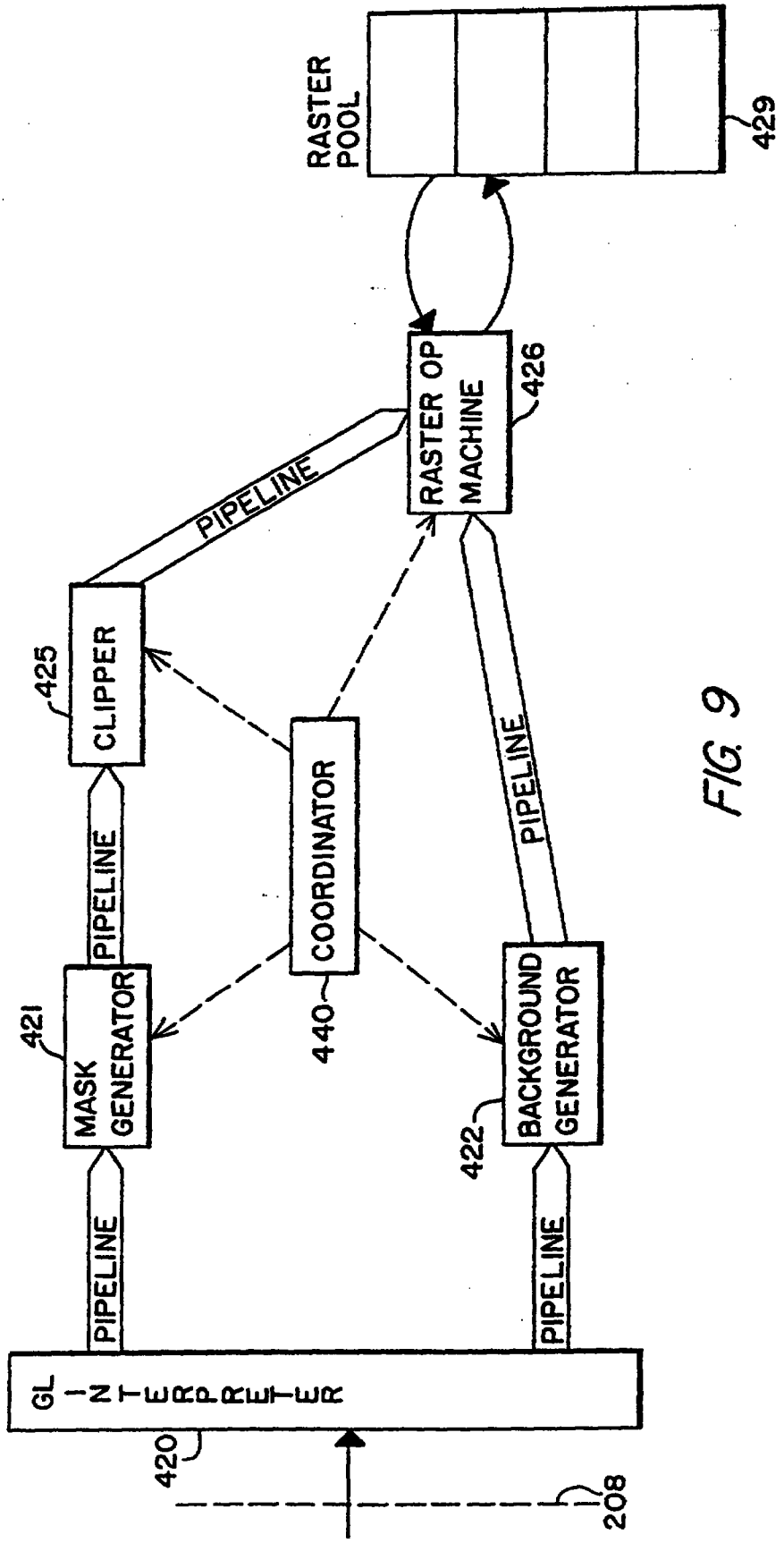


FIG. 9

REPLACEMENT SHEET
15 JUL 1998

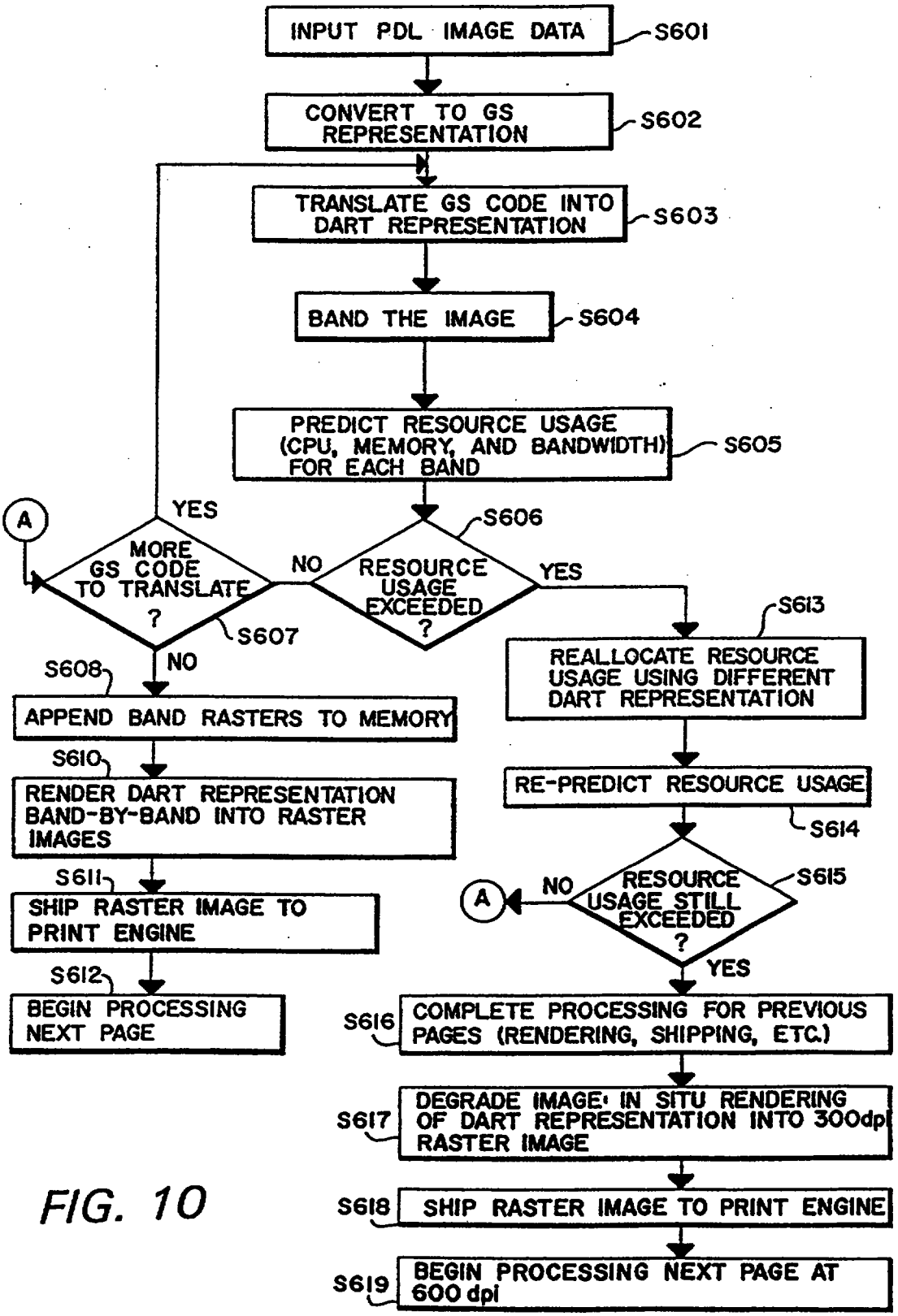


FIG. 10

REPLACEMENT SHEET
Feb
02 APR 1999
REPLACEMENT SHEET

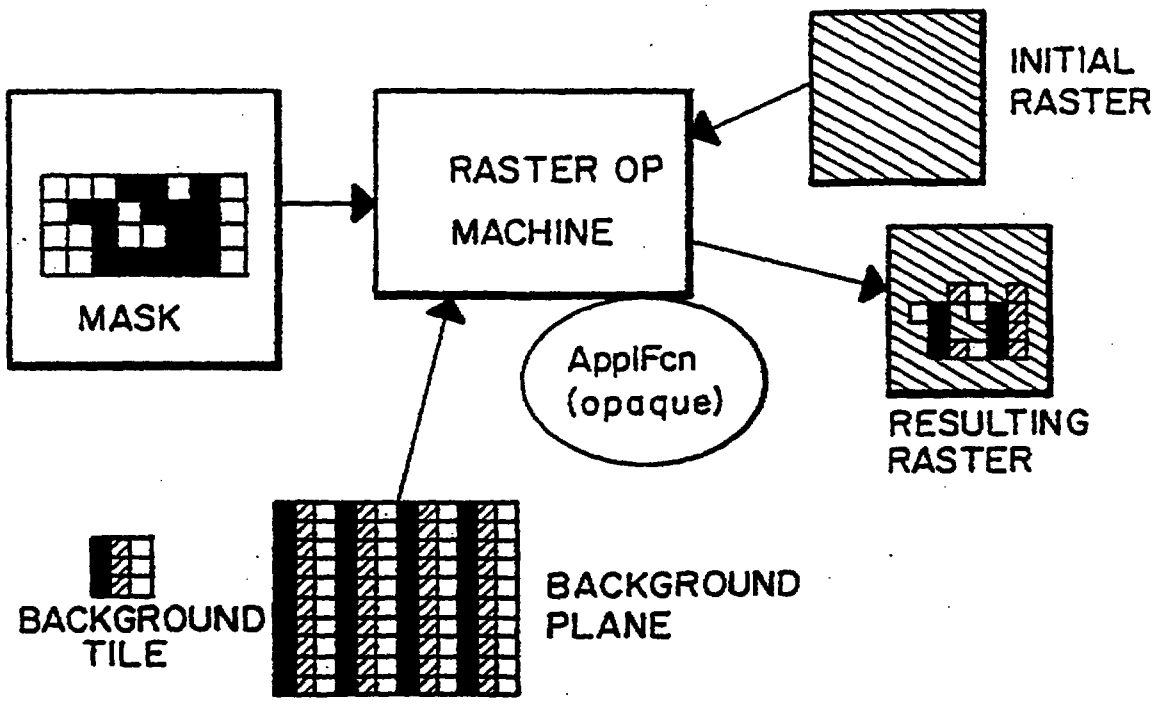


FIG. 5a

960795-4

REPLACEMENT SHEET

17 APR 1999

METHOD AND APPARATUS FOR PRINTING ACCORDING TO A GRAPHIC LANGUAGE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method and apparatus for printing an image in accordance with a graphics language stream in which graphic descriptions for frames in a page are separated from document layout information which describes how the frames are placed on the page, and in which the graphic descriptions and the document layout information are output on separate, pipelined, interfaces. A graphics language processor is provided to intercept the graphics language stream. The processor separates the graphics language stream into mask information and background information against which the masks are applied, operates independently on the mask and background information to create frame rasters, and uses the document layout information to map the frame rasters onto the page.

2. Description of the Related Art

Recently, high quality raster-based devices, such as laser beam printers and ink jet or bubble jet printers and the like, have become widely available. These devices receive image data from a host computer and generate a raster image in bit map memory for subsequent imaging and/or printing.

FIG. 1 shows the rasterization process by which each pixel in bit map memory is formed, and by which qualities (such as color, grey level or half toning) are given to each pixel. As shown in FIG. 1, the bit map raster 100 is divided into frames F which define areas in the bit map raster. For each pixel 101 in the frame, a mask M and a background B are defined. The mask M defines whether or not the pixel should be rendered into bit map memory and the location where the pixel should be rendered. The background B defines the qualities with which the pixel 101 should be rendered. For example the background B may comprise a color associated with the pixel or it may comprise a halftone density. If desired, a clipping region may also be defined to specify the effective areas of the mask and allow more precise control over the masks. The rasterization of pixel 101 in the bit map memory is the intersection of the frame information, the background information, clipping region information, and mask information, all as illustrated in FIG.