(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 January 2009 (29.01.2009)

PCT

(10) International Publication Number
WO 2009/015272 A1

(51) International Patent Classification:
G06F 19/00 (2006.01)

(21) International Application Number:
PCT/US2008/071016

(22) International Filing Date:
3 September 2008 (03.09.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/951,748 25 July 2007 (25.07.2007) US

(71) Applicant *(for all designated States except US)*: **PITNEY BOWES INC.** [US/US]; 1 Elmcroft Road, Stamford, CT 06926 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **WRONSKI, John, Jr.** [US/US]; 17 Webster Drive, Shelton, CT 06484 (US). **KIRSCHNER, Wesley, A.** [US/US]; 41 Bradford Walk, Farmington, CT 06032 (US). **PAULY, Steven, J.** [US/US]; 10 Surrey Lane, New Milford, CT 06776 (US). **HURD, John, A.** [US/US]; 145 Penny Lane, Torrington, CT 06790 (US).

(74) Agent: **RUSSO, Karin, A.**; Pitney Bowes Inc., 35 Waterview Drive, Shelton, CT 06484 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: METHOD FOR PRINTING INFORMATION ON PACKAGES

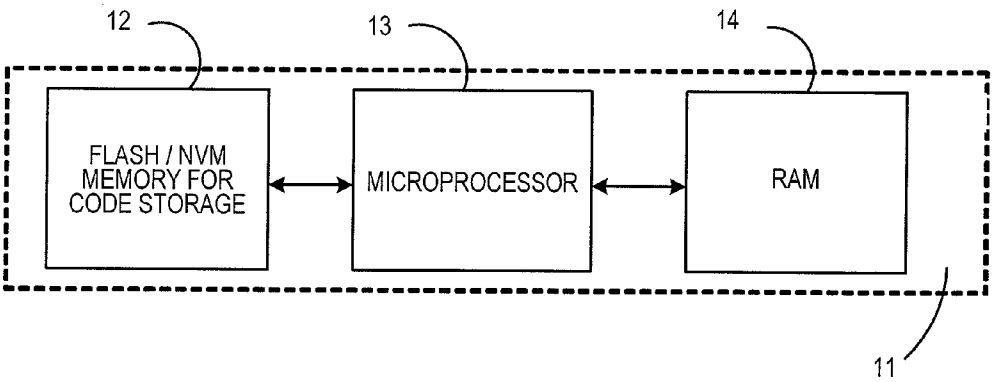

FIG. 1

(57) Abstract: A method that enables the correct information to be printed on packages when switching from printing one type of information to another type of information, i.e., switching from printing one job to another job. The foregoing is accomplished by utilizing separate memory heaps (a dedicated consecutive sequence of memory space for a specific process) and flushing the memory heaps in order to quickly return to the memory and prevent no package on the conveyor from having the incorrect information printed on it from each printer printing information on the package. A parser (utilizes computer control language to assemble a set of data objects that represent the print job) and a renderer (takes the parser's description of the print job and formats a block of memory that indicates the print pattern of the job) are all synchronized to perform the system wide flush to control memory fragmentation and loading the new job.

WO 2009/015272 A1

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report*

— *the filing date of the international application is within two months from the date of expiration of the priority period*

METHOD FOR PRINTING INFORMATION ON PACKAGES

[0001] This Application claims the benefit of the filing date of U.S. Provisional Application Number 60/951,748 filed July 25, 2007, which is owned by the assignee of the present Application.

Field of the Invention

[0002] The invention relates generally to packaging and package conveying systems, and more particularly to, a method for printing information on packages.

Background of the Invention

[0003] Printing information on packages has been an ongoing requirement for centuries. As automation becomes evermore a fact of life, the information printed on the label or the information printed on package play an ever wider role in achieving automation. The printed information may relate to the contents of the package, the source and/or destination of the package, relevant purchase and transit data, the brand of the goods within the package etc.

[0004] Printing information on a moving object requires the ordering of many events, positioning for each package to be labeled or printed, while the packages continue to move rapidly on the conveyor. The variability in package height, size and spacing, together with the varied data to be printed on the packages require significant system agility and responsiveness to keep pace with the flow of packages. Information maybe printed on packages by one or more printers which are controlled by one or more computers.

[0005] A problem occurs in prior art packaging systems when the system switches from one job (the information printed on the package) to the next. As packages continue to flow down a conveyor at a fixed speed different information may be printed on the package. The foregoing requires different print jobs to be sent to a printer. When a new print job is received, it is important for the system to switch as fast as possible from printing one type of information to printing another type of information. This means that the memory controlling the information that is printed needs to be released quickly and the new print job needs to be loaded. In certain

applications, where the printing mechanism does not control the speed of the conveyor, packages may arrive at the print head and receive no ink because the new print job was not loaded, information is printed on a precise number of packages and then the job needs to switch to printing information on a precise number of other packages. In other applications, the number of packages is not exactly specified, but the printing of different information on the package must occur such that a package has the correct information printed on either the previous job package or new job package.

[0006] Prior art attempts to solve the foregoing problem have resulted in packages completing their journey down the assembly line with no information printed on the package or packages completing their journey down the assembly line with partial information printed on the package or incorrect information printed on the package.

[0007] Another prior art problem is coordinating the printing of information on packages by a downstream printer controlled by a computer with printing information on another package by an upstream printer controlled by a different computer.

Summary of the Invention

[0008] This invention overcomes the disadvantages of the prior art by providing a method that enables the correct information to be printed on packages when switching from printing one type of information to another type of information, i.e., switching from printing one job to another job. The foregoing is accomplished by utilizing separate memory heaps (a dedicated consecutive sequence of memory space for a specific process) and flushing the memory heaps in order to quickly return to the memory and prevent no package on the conveyor from having the incorrect information printed on it from each printer printing information on the package. The parser (utilizes computer control language to assemble a set of data objects that represent the print job) and renderer (takes the parser's description of the print job and formats a block of memory that indicates the print pattern of the job) are all synchronized to perform the system wide flush to control memory fragmentation and loading the new job.

[0009] The parallel parsing of the new job occurs while the old job is continuing printing. A counter is also utilized to trigger the switching to a new job when an exact number of pieces need to be loaded. Thus, continuing printing may be obtained when switching from one job to another job.

[0010] An advantage of this invention is that the throughput of a conveyor system that prints information on packages may be increased, since an exact number of packages may be loaded and printed without the conveyor stopping.

[0011] An additional advantage of this invention is that incorrect information will not be printed on packages.

[0012] An additional advantage of this invention is that partial information will not be printed on packages.

[0013] A further advantage of this invention is that the utilization of flushing a heap in the algorithm results in an extremely fast method to return all dynamically allocated memory.

[0014] A still further advantage of this invention is that there will be no memory leaks or fragmentation caused by this algorithm.

[0015] An additional advantage of the invention is that two complete Jobs can be loaded into the overlay buffers and have the variable Page Objects available for printing at the same time.

[0016] A still further advantage of this invention is that it has the ability to switch from one print job to the next print job at nominal printing speed with no blank prints and without stopping the conveyer.

[0017] A still further advantage of this invention is that it has the ability to print exactly N number of prints per job and switch to the next job and print N number of prints without stopping the conveyer. This allows an exact job size to be specified.

Brief Description of the Drawings

[0018] Fig. 1 is a drawing of the hardware embodiment of this invention showing a printer having flash non volatile memory, a microprocessor and random access memory;

[0019] Fig. 2A is a drawing showing a dedicated host module which parses the print job and communicates the print job to the attached rendering modules;

[0020] Fig. 2B is a drawing showing a dedicated host module which parses the print job and communicates the print job to the attached rendering modules and receives a portion of the print job to print;

[0021] Fig. 2C is a drawing showing the printing of two labels on three packages;

[0022] Fig. 3A and Fig. 3B is a flow chart of a program running on the host with parser 31 (Fig. 2B); and

[0023] Fig. 4A and Fig. 4B is a flow chart of a program running on renders 23 and 24 (Fig. 2A) , and renders 32, 33 and 34 (Fig. 2B).

Detailed Description of the Present Invention

[0024] Referring now to the drawings in detail and more particularly to Fig. 1, the reference character 11 represents a printer that contains flash or non volatile memory 12, microprocessor 13 and random access memory 14. Microprocessor 13 is coupled to non volatile memory 12 and random access memory 14. Microprocessor 13 executes the software algorithm described in the description of Fig. 3 to be used by printer 11 to parse and render the print job. Non volatile memory 12 is used for storing the algorithm. Memory 12 is shown external to the processor, but it also may be internal to the processor. Random access memory 14 stores the parsed and rendered print job.

[0025] Fig. 2A is a drawing showing a dedicated host module which parses the print job and communicates the print job to the attached rendering modules. A host

21 containing a parser is coupled to a first renderer and print head 22, a second renderer and print head 23 and Nth renderer and print head 24. A conveyor 25 containing moving packages is adjacent to first renderer and print head 22, second renderer and print head 23 and Nth renderer and print head 24.

[0026] Fig. 2B is a drawing showing a dedicated host module which parses the print job and communicates the print job to the attached rendering modules and receives a portion of the print job to print. A host 31 containing a parser first renderer and print heads is coupled to a second renderer 32, a third renderer 33, and Nth renderer 34. A conveyor 35 containing moving packages is adjacent to the first renderer and print head 31, second renderer and print head 32, third renderer and print head 33, and Nth renderer and print head 34.

[0027] The parsers and renders shown in Figs. 2A and 2b may exist on the same microprocessor. It is also possible for the renderer and parser to be present on separate microprocessors. When the renders and parsers are on a separate microprocessor, Module 1 with a CPU and parser creates package data objects and sends them through a communication channel to Module 2 and Module 2 with a CPU, a renderer, and print head, receives package data objects and renders a print buffer for printing the print job.

[0028] In this invention a host has a parser at a minimum. It can also contain a renderer and print heads. There may be other print heads in the system and each has renderers. Both the host and the renderers contain the hardware embodiment shown in Fig. 1, i.e., contains flash or non volatile memory 12, microprocessor 13 and random access memory 14.

[0029] The invention of the embodiment of Fig. 2B has 1, 2, 3, or 4 print heads.

[0030] In order to allow flexibility in building pages, a very flexible data structure that dynamically allocates a lot of memory is built. It would take a large amount of conveyor time, measured in packages per minute, to free up the memory of the first job and hence, it would cause a serious delay in starting another job once the first packaging job has completed. When the conveyor is not stopped while the new print job is loading, some boxes on the conveyor will not receive any printed data.

[0031] One possible algorithm to implement this invention is described in Figs. 3A – 3D is very flexible, thus it may be utilized by the basic hardware described above.

[0032] In order to quickly free memory, the algorithm described in Figs. 3A – 3D is used to place different parts of the parsed pages on different memory heaps (large regions of memory that can be dynamically allocated) and then flushes (releases and reinitializes) the entire heap when the new job comes in. The algorithm also allows the previous job to keep printing until the new job is parsed and ready. This controlled flushing of heaps allows maximum throughput and a precise switchover from one job to the next.

[0033] The foregoing is accomplished by creating different heaps for different objects. Odd and Even heaps are used according to the job ids polarity. At a high level, the table below shows the heaps and the role the parser and renderer have for each of them:

Heap	Object	Parser	Renderer
1	Graphic (static)	Populates	Deletes after Rendering to an available buffer
2	Page (variable)	Populates	Renders to buffer on each page print New Job event halts Start Page events Deletes after all Start Page events are processed
2	Font (variable)	Populates	Renders to buffer on each page print New Job event halts Start Page events Deletes after all Start Page events are processed

3	Page (variable)	Populates	Renders to buffer on each page print New Job event halts Start Page events Deletes after all Start Page events are processed
3	Font (variable)	Populates	Renders to buffer on each page print New Job event halts Start Page events Deletes after all Start Page events are Processed.

[0034] On a more detailed level, the following objects and are used by the algorithm:

- One Heap for Static Page Objects that is flushed after the overlay print buffer has been rendered.
- One Odd Overlay Print Buffer for Static Page Objects for Jobs with and Odd Page ID
- One Even Overlay Print Buffer for Static Page Objects for Jobs with and Odd Page ID
- Four Print Buffers for rendering the variable page objects
- One Heap for Odd Variable Page Objects that is flushed when the Odd counter for Jobs with an Odd Page ID reaches zero and the print engine has switched to a new print job
- One Heap for Even Variable Page Objects that is flushed when the Even counter for Jobs with an Even Page ID reaches zero and the print engine has switched to a new print job

- One Static Render Thread for rendering Static Page Objects to the Overlay Buffer.
- One Variable Render Thread for rendering Variable Page Objects to one of the four Print Buffer.
- One Odd UpstreamSensor Counter incremented when Jobs with an Odd Page ID is active
- One Even UpstreamSensor Counter incremented when Jobs with an Even Page ID is active
- One PageObject with two lists, a Static Object List and a Variable Object List

[0035] The host will broadcast a message to indicate to all renders when it is time to switch jobs, which requires the UpstreamSensor Counter to switch to the other counter.

[0036] This invention can also be used in standard printers when they are printing copies. It can even be used in mail finishing equipment to load and unload ad slogans and other parts of the envelope that are printed on.

[0037] Typically there are two major software tasks in a printer that are used to create a page that will be printed.
Parser – this software takes PCL (Printer Control Language) and parses the data into a data object form that specifies how the page will be created.
Renderer – this software takes the parsed page data objects and creates a bit map in memory of the image for the section of the page the renderer is configured. In certain applications, there will be multiple renderers because each piece will create a part of the image.

[0038] Fig. 3A and Fig. 3B is a flow chart of a program running on the host with parser 31 (Fig. 2B). The program begins at start block 99. Decision block 200 receives an input from encoder 98 (Fig. 2B). Block 200 determines whether or not it is time to print. If decision block 200 determines it is not time to print the program

goes back to the input to block 200. If block 200 determines that it is time to print the program goes to decision block 201. Block 201 determines whether or not there is a column to print. If decision block 201 determines there is not a column to print the program goes back to the input to block 200. If block 201 determines that there is a column to print the program goes to decision block 202 to print the column. Then the program goes to decision block 203. Block 203 determines whether or not it was the last column to print. If decision block 203 determines there was the last column to print the program goes back to the input to block 200. If block 203 determines that it was the last column to print the program goes to block 204. Block 204 releases the page buffer and decrements the active page counter for printing.

[0039] Now the program goes to decision block 205. Block 205 determines whether or not the active print page counter equals zero. If block 205 determines that the active print page counter does not equals zero the program goes to decision block 208. If block 205 determines that the active print page counter equals zero the program goes to decision block 206. Block 206 determines whether or not it is time to switch the active print page. If block 206 determines that it is not time to switch the active print page the program goes back to the input of block 200. If block 206 determines that it is time to switch the active print page the program goes to block 207 to switch the active page counter for printing and free the variable heap. Next the program goes to decision block 208. Block 208 determines whether or not a page has to be created. If block 208 determines that a page does not have to be created the program goes back to the input of block 200. If block 208 determines that a page has to be created the program goes to block 209 to get page buffer, render variable data for the page from the variable heap. Then the program goes back to the input of block 200.

[0040] Decision block 220 receives an input when the user loads a new job. Block 220 determines whether or not this is a new job. If decision block 220 determines that it is not a new job the program goes back to the input to block 220. If block 220 determines that it is a new job the program goes to block 221 to parse the PCL (Printer Control Language) job and to create page objects on static and variable heaps. Then the program goes to block 222 to send the page objects to other printers. Next the program goes to decision block 223. Block 223 determines whether or not all the printers have stored the page objects. If block 223 determines

that all the printers have not stored the page objects the program goes back to the input of block 223. If block 223 determines that all the printers have stored the page objects the program goes to the block 224 to render static objects to next overlay buffer. Now the program goes to block 225 to free the static heap. At this point the program goes to block 226 to send page rendered notifications to all printers. Then the program goes to decision block 227. Block 227 determines whether or not all printers have finished rendering static page objects. If block 227 determines that all the printers have not finished rendering static page objects the program goes back to the input of block 227. If block 227 determines that all the printers have finished rendering static page objects the program goes to block 228 to send switch job notifications to all printers. Then the program goes to block 229 to switch the active page counter for incoming pages. At this point the program goes back to block 220 to wait for the next job.

[0041] Decision block 240 receives an input from sensor 41 (Fig. 2B). Block 240 determines whether or not this is a new page. If decision block 240 determines that it is not a new page the program goes back to the input to block 240. If block 240 determines that it is a new page the program goes to block 241 to send new page notifications to all printers. Then the program goes to block 242 to increment the page counter for incoming pages. Next the program goes to decision block 243. Block 243 determines whether or not a new page has to be created. If decision block 243 determines that a new page does not have to be created the program goes back to the input of block 240. If block 243 determines that a new page has to be created the program goes to block 244 to get page buffer, render variable data for the page from the variable heap. Then the program goes back to the input of block 240.

[0042] Fig. 4A and Fig. 4B is a flow chart of a program running on renders 23 and 24 (Fig. 2A) , and renders 32, 33 and 34 (Fig. 2B). Decision block 300 receives an input from encoder 97and encoder 98. Block 300 determines whether or not it is time to print a column. If decision block 300 determines it is not time to print a column the program goes back to the input to block 300. If block 300 determines that it is time to print a column the program goes to decision block 301. Block 301 determines whether or not there is a column to print. If decision block 301 determines there is not a column to print the program goes back to the input to block

300. If block 301 determines that there is a column to print the program goes to block 302 to print the column. Then the program goes to decision block 303. Block 303 determines whether or not it was the last column printed. If decision block 303 determines that it was not the last column printed the program goes back to the input to block 300. If block 303 determines that it was the last column printed the program goes to block 304. Block 304 releases the page buffer and decrements the active page counter for printing.

[0043] Now the program goes to decision block 305. Block 305 determines whether or not the active print page counter equals zero. If block 305 determines that the active print page counter does not equals zero the program goes to decision block 308. If block 305 determines that the active print page counter equals zero the program goes to decision block 306. Block 306 determines whether or not it is time to switch the active print page counter for printing. If block 306 determines that it is not time to switch the active print page counter for printing the program goes back to the input of block 200. If block 306 determines that it is time to switch the active print page counter for printing the program goes to block 307 to switch the active page counter for printing and free the variable heap. Next the program goes to decision block 308. Block 308 determines whether or not a page has to be created. If block 308 determines that a page does not have to be created the program goes back to the input of block 300. If block 308 determines that a page has to be created the program goes to block 309 to get page buffer, render variable data for the page from the variable heap. Then the program goes back to the input of block 300.

[0044] Decision block 320 receives an input from host 31 (Fig. 2B) after a user has loaded a new job. Block 320 determines whether or not there are new page objects. If decision block 320 determines that are no new page objects the program goes back to the input to block 320. If block 320 determines that it are new page objects the program goes to block 321 to store page objects on static and variable heaps. Then the program goes to block 322 to send to host stored page objects. Next the program goes to decision block 323. Block 323 renders static objects to next overlay buffer. Then the program goes to the block 324 to free the static heap. Then the program goes to block 325 to send page rendered notification to host. At this point the program goes to decision block 326. Block 326 determines whether or not it is time to switch jobs. If block 326 determines that it is not time to switch jobs

the program goes back to the input of block 326. If block 326 determines that it is time to switch jobs the program goes to block 327 to switch active page counter for incoming pages. Then the program goes back to the input of block 320.

[0045] Decision block 340 receives an input from host 31 or sensor 41. Block 340 determines whether or not this is a new page. If decision block 340 determines that it is not a new page the program goes back to the input to block 340. If block 340 determines that it is a new page the program goes to block 341 to increment page counter for incoming pages. Then the program goes to decision block 342. Block 342 determines whether or not a new page has to be created. If decision block 342 determines that a new page does not have to be created the program goes back to the input of block 340. If block 342 determines that a new page has to be created the program goes to block 343 to get page buffer, render variable data for the page from the variable heap. Then the program goes back to the input of block 340.

[0046] The above specification describes a new and improved method for method for printing information on packages. It is realized that the above description may indicate to those skilled in the art additional ways in which the principles of this invention may be used without departing from the spirit. Therefore, it is intended that this invention be limited only by the scope of the appended claims.

What is claimed is:

1. A method for printing information on packages, comprising the steps of:

placing a plurality of packages on a continuously moving conveyor;

printing information by a plurality of printers comprising font, font size, font character and graphic on the packages as the packages move on the conveyor; and

changing the information printed by the plurality of printers on the packages as the conveyor continues to move so that the information printed on the packages will be correct when the printers switch from printing a type of the information on a first job to a different type of the information on a second job.

2. The method claimed in claim 1, wherein the packages have varying sizes.

3. The method claimed in claim 1, wherein the packages have varying shapes.

4. The method claimed in claim 1, wherein the changing step further comprises:

placing static information from a first print job in a static heap;

rendering static information from the static heap to a first overlay print buffer;

freeing the static heap;

determining the package presence on the conveyor in the proximity of a print head;

parsing variable objects in the first print job to a first variable heap for the first print job;

placing the variable objects in a print buffer;

placing buffer data in the first overlay print buffer and data in the print buffer in a print head buffer;

printing information for the first print job;

placing static information from a second print job in the static heap;

rendering static information from the static heap to a second overlay print buffer;

freeing the static heap;

determining the package presence on the conveyor in the proximity of a print head;

parsing variable objects in the second print job to a second variable heap for the second print job;

placing the variable objects in the print buffer;

placing buffer data in the second overlay print buffer and data in the print buffer in a print head buffer; and

printing information for the second print job.

5. The method claimed in claim 1, wherein information is printed on X number of packages before printing information on Y number of packages, wherein X and Y are numbers greater than or equal to one.

6. The method claimed in claim 2, wherein the changing step comprises:

placing static information from a first print job in a static heap;

rendering static information from the static heap to a first overlay print buffer;

freeing the static heap;

determining the package presence on the conveyor in the proximity of a print head;

parsing variable objects in the first print job to a first variable heap for the first print job;

placing the variable objects in a print buffer;

placing buffer data in the first overlay print buffer and data in the print buffer in a print head buffer;

setting a counter for the number of packages to be printed for the first print job;

printing information for the first print job;

decrementing the counter as the packages for the first print job are printed;

placing static information from a second print job in the static heap;

rendering static information from the static heap to a second overlay print buffer;

freeing the static heap;

determining the package presence on the conveyor in the proximity of a print head;

parsing variable objects in the second print job to a second variable heap for the second print job;

placing the variable objects in the print buffer;

placing buffer data in the second overlay print buffer and data in the print buffer in a print head buffer;

setting the counter for the number of packages to be printed for the second print job;

switching from the first print job to the second print job when the first print job count has a count of zero

and

printing information for the second print job.

7. The method claimed in claim 1, wherein images are printed on media.

8. The method claimed in claim 7, wherein images are printed on the media in a desired order.

9. The method claimed in claim 8, images are printed on the media in a desired order for a precise predefined number of media items.

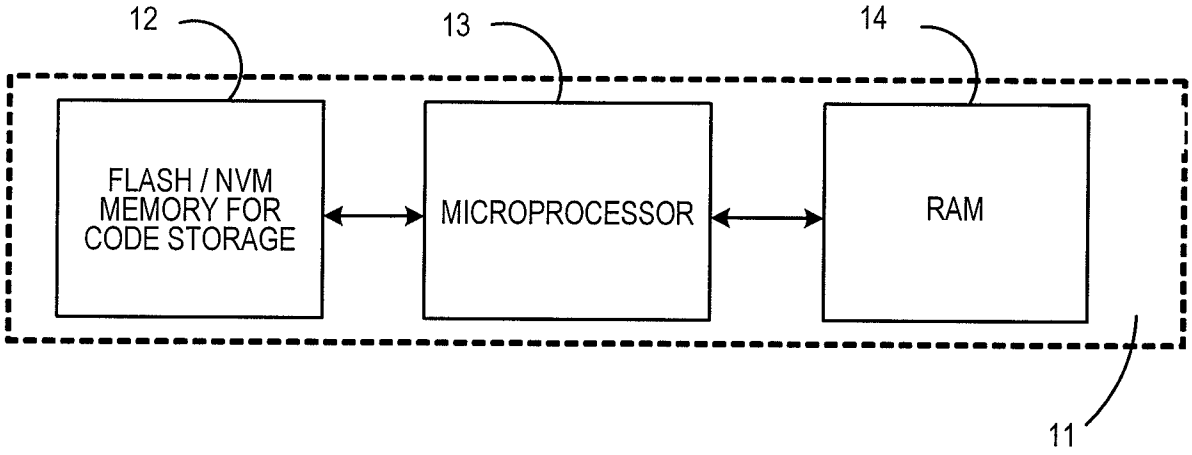
12
13
14
FLASH / NVM MEMORY FOR CODE STORAGE
MICROPROCESSOR
RAM
11

FIG. 1

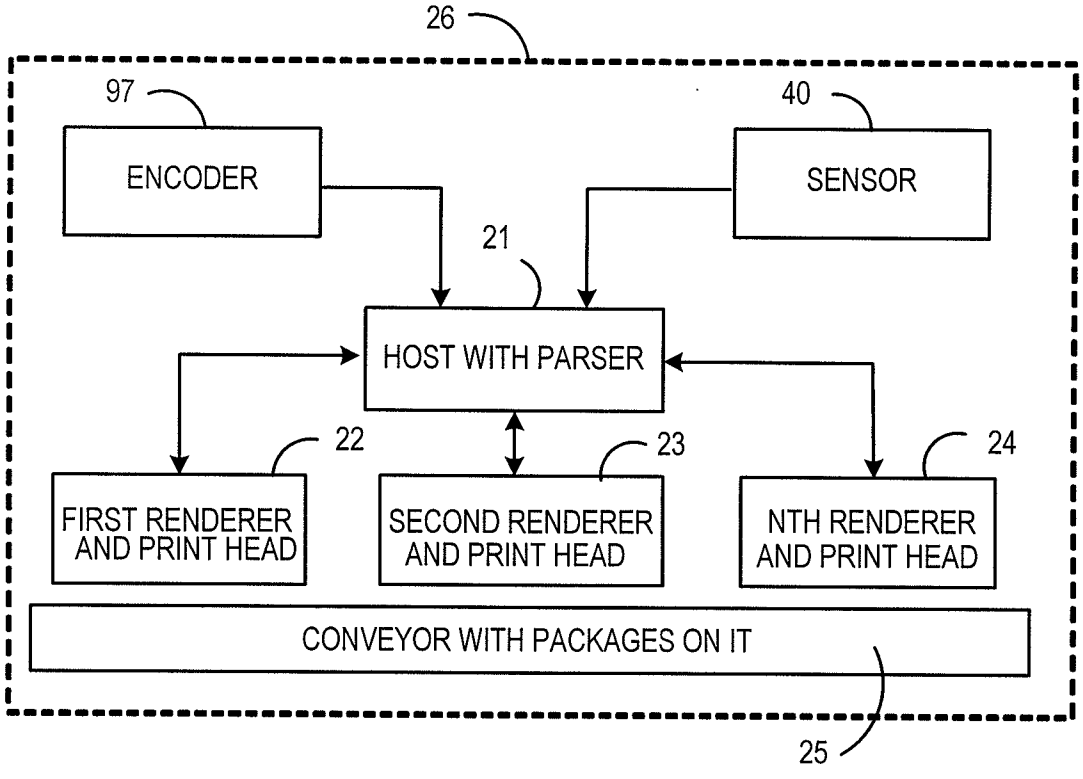
26
97
40
ENCODER
SENSOR
21
HOST WITH PARSER
22
23
24
FIRST RENDERER AND PRINT HEAD
SECOND RENDERER AND PRINT HEAD
NTH RENDERER AND PRINT HEAD
CONVEYOR WITH PACKAGES ON IT
25

FIG. 2A

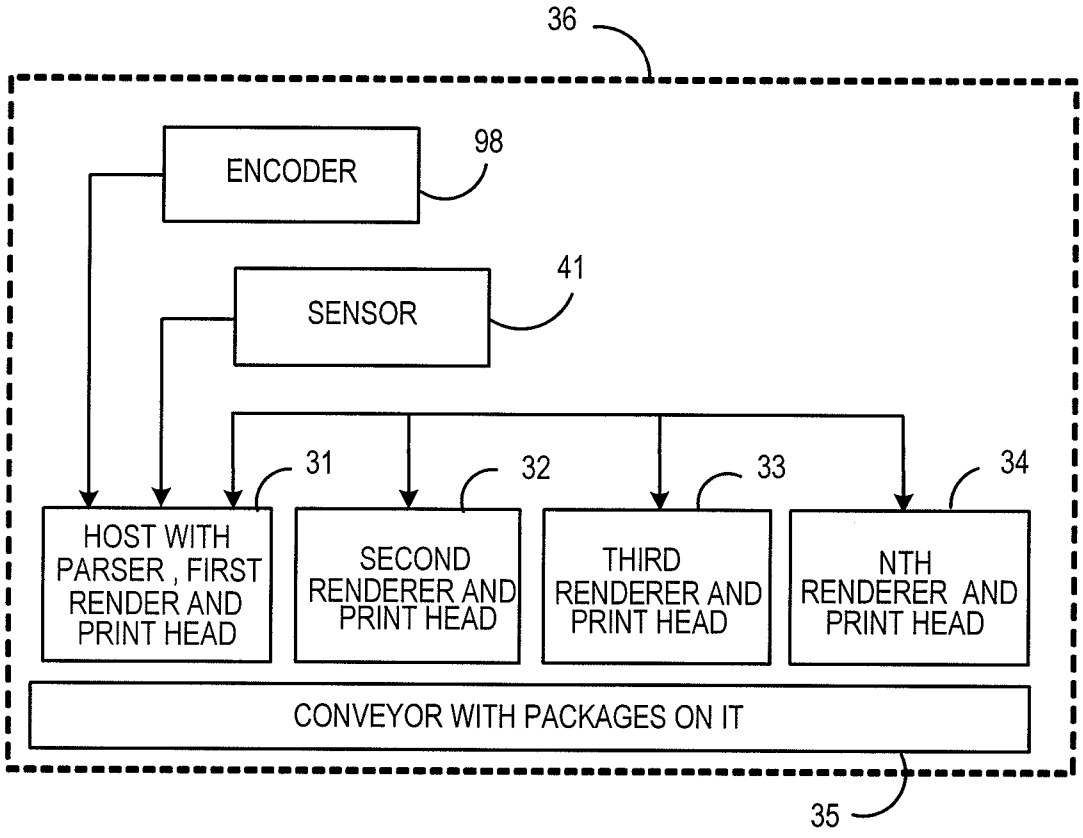
36
ENCODER
98
SENSOR
41
31
32
33
34
HOST WITH PARSER , FIRST RENDER AND PRINT HEAD
SECOND RENDERER AND PRINT HEAD
THIRD RENDERER AND PRINT HEAD
NTH RENDERER AND PRINT HEAD
CONVEYOR WITH PACKAGES ON IT
35

FIG. 2B

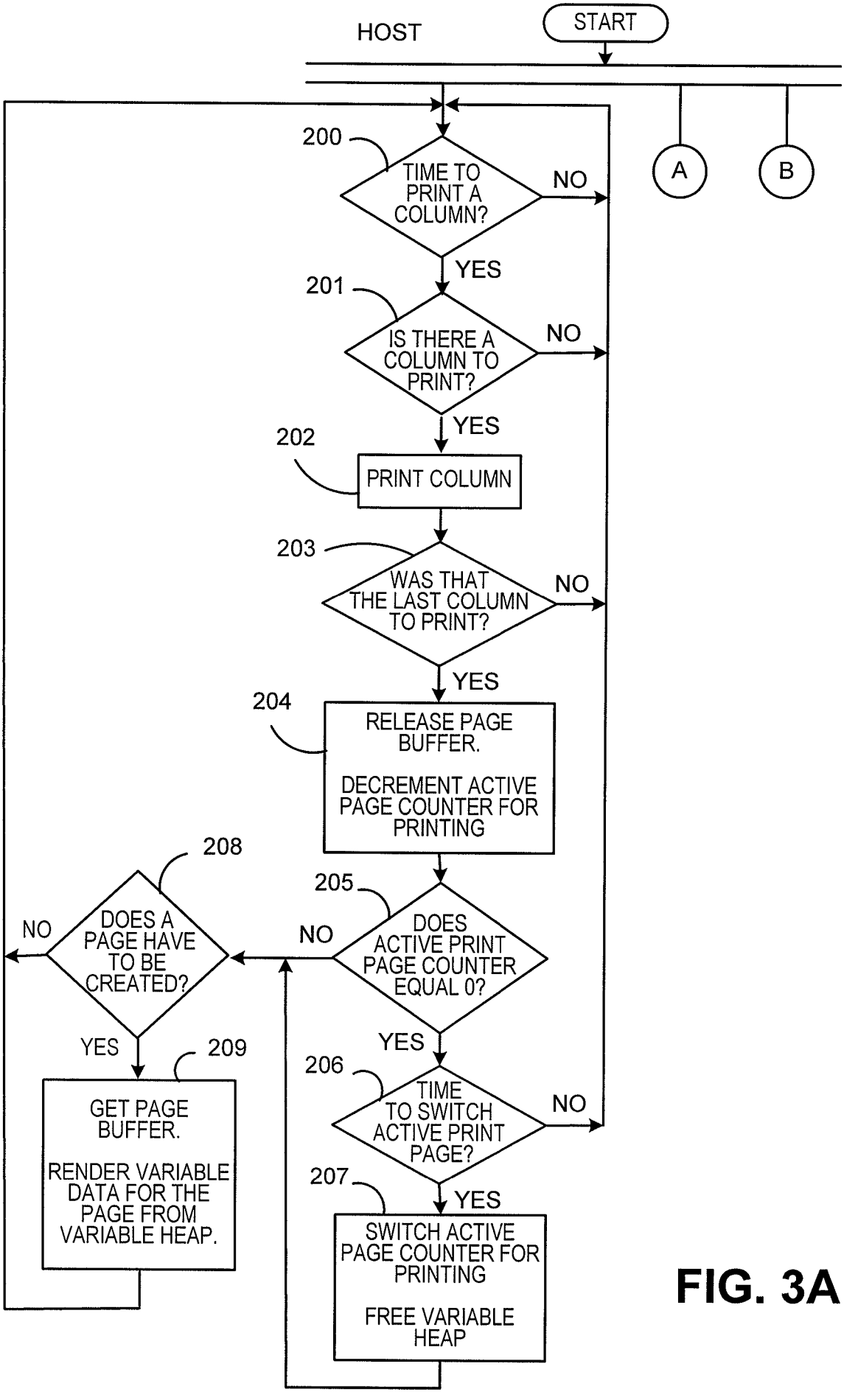
HOST
START
200
TIME TO PRINT A COLUMN?
NO
YES
201
IS THERE A COLUMN TO PRINT?
NO
YES
202
PRINT COLUMN
203
WAS THAT THE LAST COLUMN TO PRINT?
NO
YES
204
RELEASE PAGE BUFFER.
DECREMENT ACTIVE PAGE COUNTER FOR PRINTING
205
DOES ACTIVE PRINT PAGE COUNTER EQUAL 0?
NO
YES
206
TIME TO SWITCH ACTIVE PRINT PAGE?
NO
YES
207
SWITCH ACTIVE PAGE COUNTER FOR PRINTING
FREE VARIABLE HEAP
208
DOES A PAGE HAVE TO BE CREATED?
NO
YES
209
GET PAGE BUFFER.
RENDER VARIABLE DATA FOR THE PAGE FROM VARIABLE HEAP.
A
B

FIG. 3A

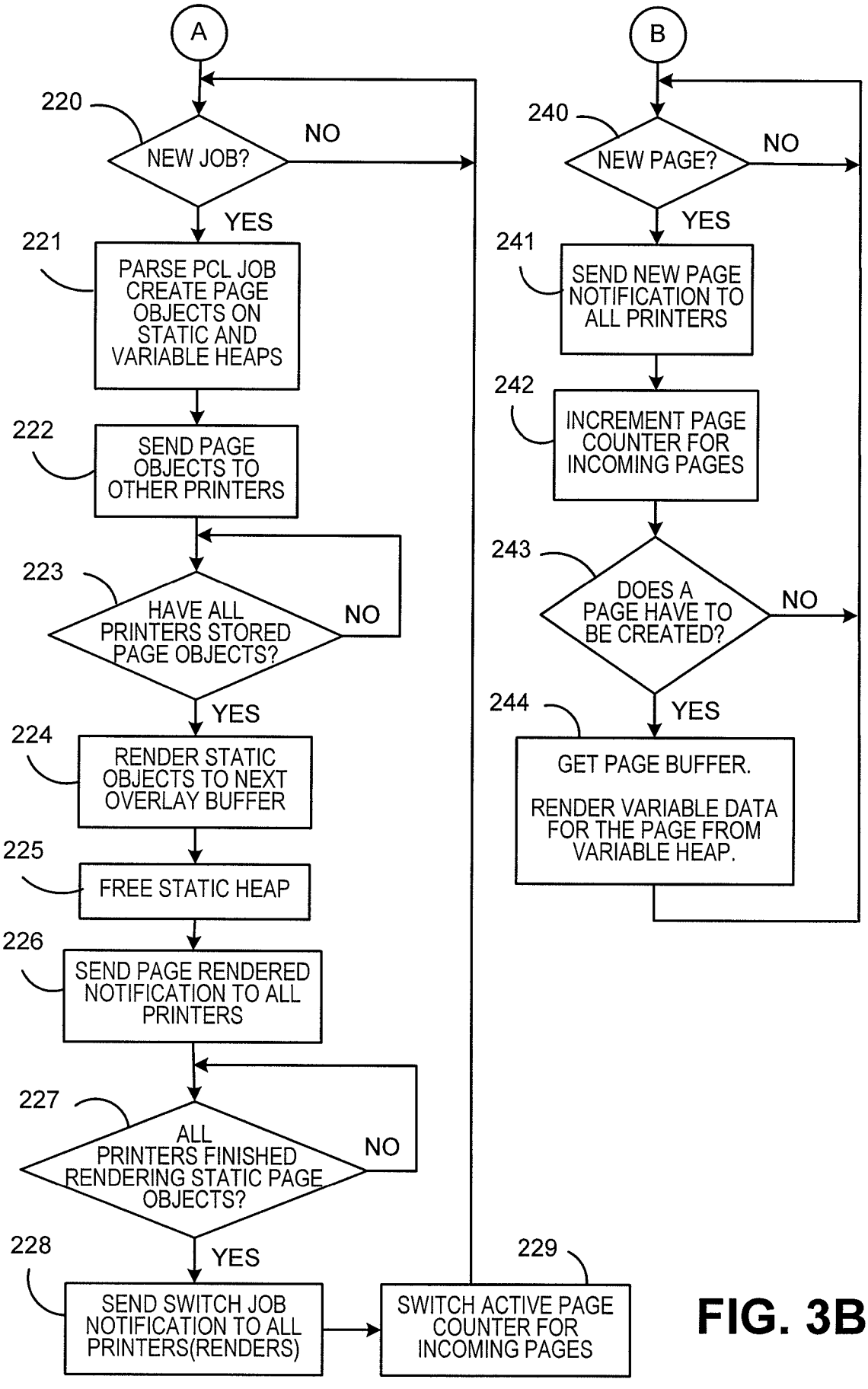
A
220
NEW JOB?
NO
YES
221
PARSE PCL JOB CREATE PAGE OBJECTS ON STATIC AND VARIABLE HEAPS
222
SEND PAGE OBJECTS TO OTHER PRINTERS
223
HAVE ALL PRINTERS STORED PAGE OBJECTS?
NO
YES
224
RENDER STATIC OBJECTS TO NEXT OVERLAY BUFFER
225
FREE STATIC HEAP
226
SEND PAGE RENDERED NOTIFICATION TO ALL PRINTERS
227
ALL PRINTERS FINISHED RENDERING STATIC PAGE OBJECTS?
NO
YES
228
SEND SWITCH JOB NOTIFICATION TO ALL PRINTERS(RENDERS)
229
SWITCH ACTIVE PAGE COUNTER FOR INCOMING PAGES
B
240
NEW PAGE?
NO
YES
241
SEND NEW PAGE NOTIFICATION TO ALL PRINTERS
242
INCREMENT PAGE COUNTER FOR INCOMING PAGES
243
DOES A PAGE HAVE TO BE CREATED?
NO
YES
244
GET PAGE BUFFER.
RENDER VARIABLE DATA FOR THE PAGE FROM VARIABLE HEAP.

FIG. 3B

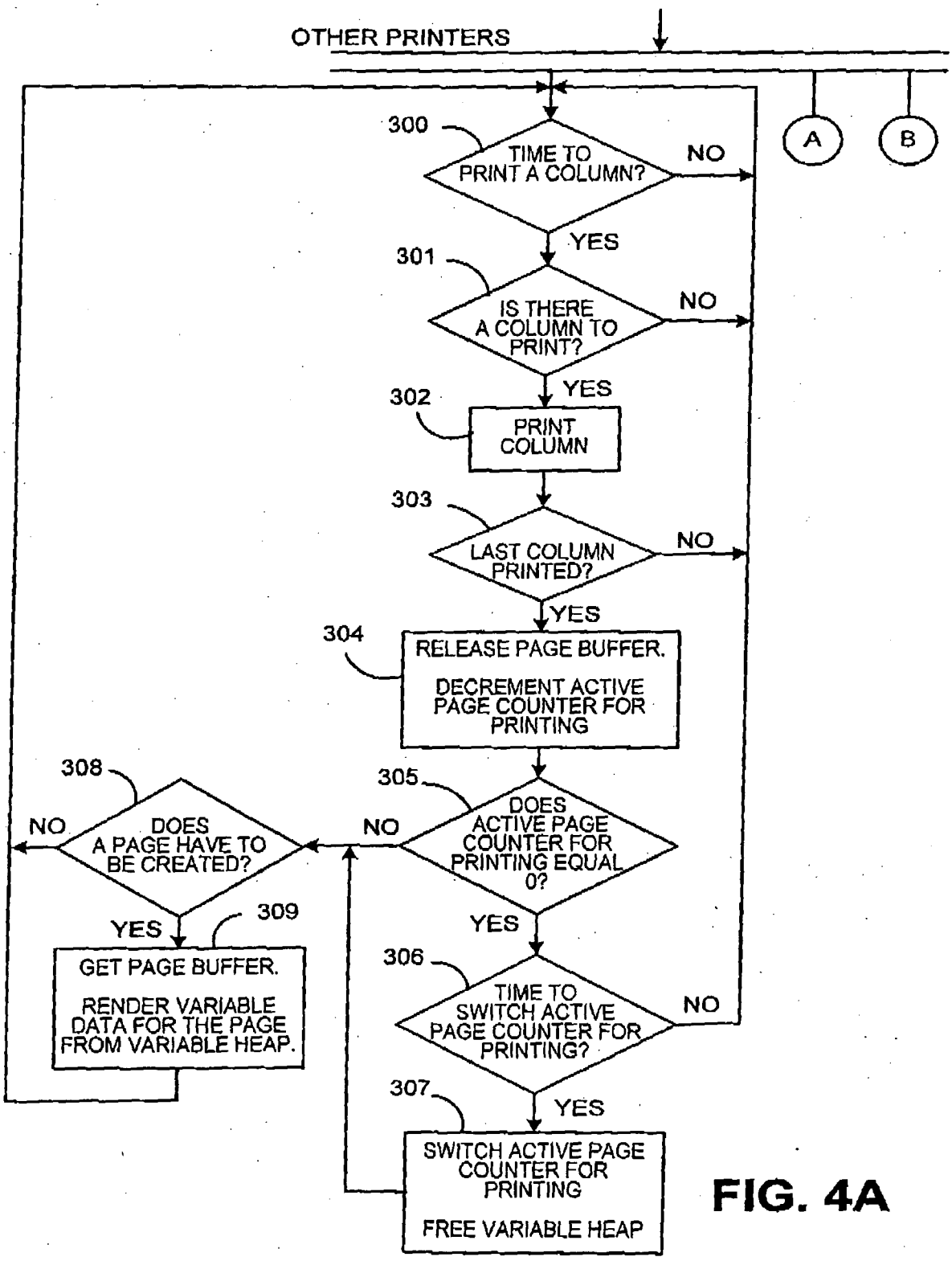
OTHER PRINTERS
A
B
300
TIME TO PRINT A COLUMN?
NO
YES
301
IS THERE A COLUMN TO PRINT?
NO
YES
302
PRINT COLUMN
303
LAST COLUMN PRINTED?
NO
YES
304
RELEASE PAGE BUFFER.
DECREMENT ACTIVE PAGE COUNTER FOR PRINTING
305
DOES ACTIVE PAGE COUNTER FOR PRINTING EQUAL 0?
NO
YES
306
TIME TO SWITCH ACTIVE PAGE COUNTER FOR PRINTING?
NO
YES
307
SWITCH ACTIVE PAGE COUNTER FOR PRINTING
FREE VARIABLE HEAP
308
DOES A PAGE HAVE TO BE CREATED?
NO
YES
309
GET PAGE BUFFER.
RENDER VARIABLE DATA FOR THE PAGE FROM VARIABLE HEAP.

FIG. 4A

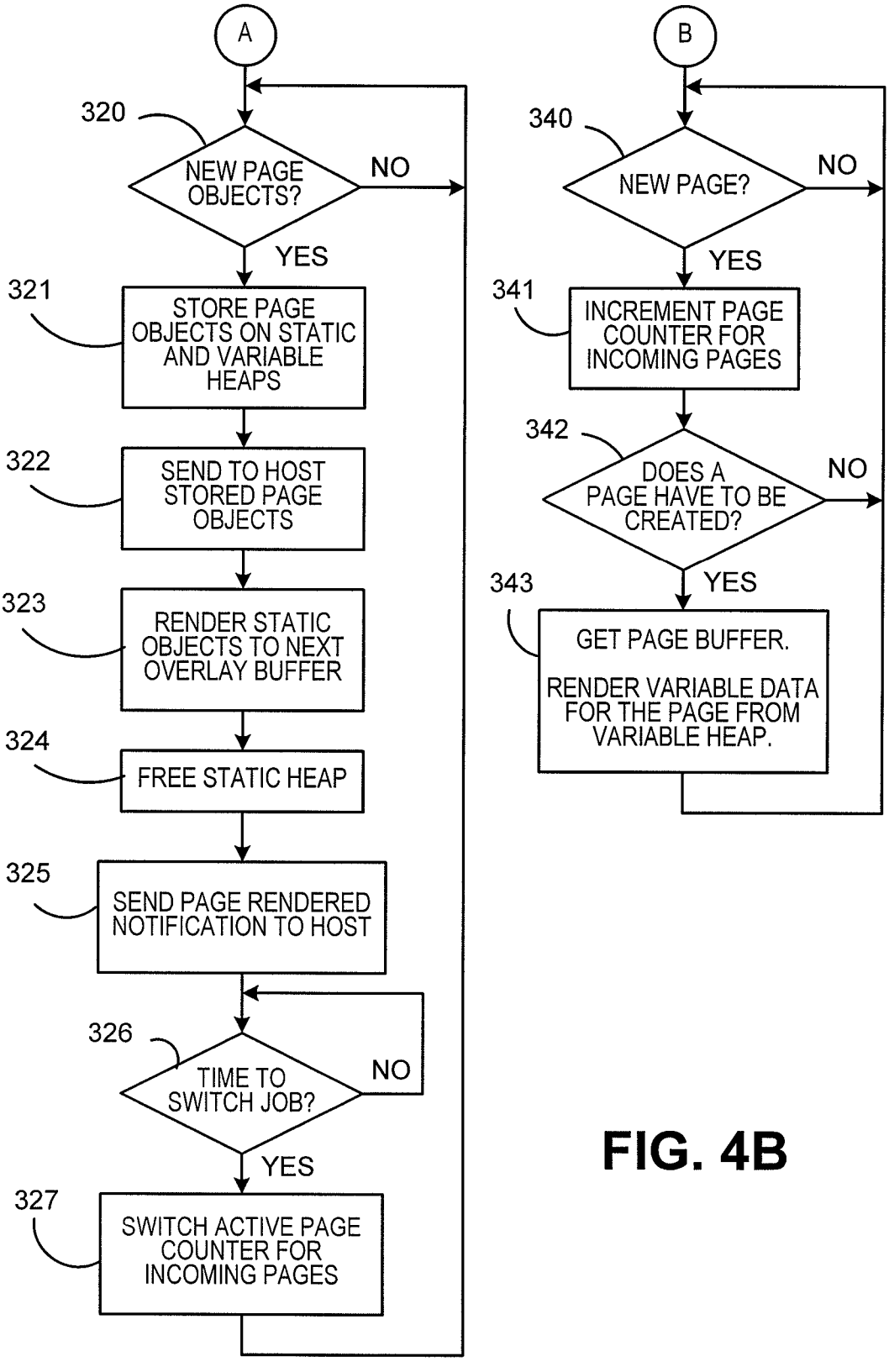
A
320
NEW PAGE OBJECTS?
NO
YES
321
STORE PAGE OBJECTS ON STATIC AND VARIABLE HEAPS
322
SEND TO HOST STORED PAGE OBJECTS
323
RENDER STATIC OBJECTS TO NEXT OVERLAY BUFFER
324
FREE STATIC HEAP
325
SEND PAGE RENDERED NOTIFICATION TO HOST
326
TIME TO SWITCH JOB?
NO
YES
327
SWITCH ACTIVE PAGE COUNTER FOR INCOMING PAGES
B
340
NEW PAGE?
NO
YES
341
INCREMENT PAGE COUNTER FOR INCOMING PAGES
342
DOES A PAGE HAVE TO BE CREATED?
NO
YES
343
GET PAGE BUFFER.
RENDER VARIABLE DATA FOR THE PAGE FROM VARIABLE HEAP.

FIG. 4B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 08/71016

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 19/00 (2008.04)
USPC - 700/99
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(8): G06F 19/00 (2008.04)
USPC: 700/99

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 700/90, 99, 100, 102; 715/200, 256, 274

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PubWest(PGPB,USPT,USOC,EPAB,JPAB); Google Scholar; Google Patents
Search Terms Used: printed, packages, conveying, separate memory heaps, parallel parsing, plurality of printers, correct, information, print head buffer etc.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2002/0152001 A1 (KNIPP et al.) 17 October 2002 (17.10.2002), para [0075]-[0099], para [0108]-[0127]	1-9
Y	US 2005/0065645 A1 (LIFF et al.) 24 March 2005 (24.03.2005), para [0012]	1-9
Y	US 2007/0019016 A1 (SILVERBROOK et al.) 25 January 2007 (25.01.2007), para [2252]-[2481], para [5356]-[5478]	4 and 6

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
06 October 2008 (06.10.2008)	15 OCT 2008
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774