(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 279 231 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **88101084.7**

(51) Int. Cl.4: **G09G 1/16**

(22) Date of filing: **26.01.88**

(30) Priority: **12.02.87 US 13849**

(43) Date of publication of application:
24.08.88 Bulletin 88/34

(84) Designated Contracting States:
DE FR GB IT

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Mansfield, Robert Lockwood.**
12303 Meuse Cove
Austin Texas 78727(US)
Inventor: **Spencer, Alexander Koos**
12705 Cantle Trail
Austin Texas 78727(US)
Inventor: **St. Clair, Joe Christopher**
2603 Valley View Cove
Round Rock Texas 78681(US)

(74) Representative: **Grant, Iain Murray**
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN(GB)

(54) Grafik-Prozessor für ein hochauflösendes Video-Anzeige-System.

(57) A processing system is provided that includes an external device connected to a processor. The external device has the capability of responding to external device commands wherein each of these external device commands is performed within at least one fixed time period. The processor provides these external device commands and further includes the means for executing instructions that not only specify the external device commands but also specify at least one internal command to be performed internally by the processor simultaneously with the performance of the external commands by the external device. In the disclosed embodiment, a graphics display system is provided that includes a system processor, a graphics processor, a graphics memory and a display device. The graphics processor receives instructions from the system processor. These instructions specify commands to be executed in both the graphics processor and in the graphics memory. The commands executed in the graphics memory are executed within a fixed time period. The commands executed by the graphics processor are also executed within this fixed time period simultaneously with the execution of the commands in the graphics memory. The graphics processor further includes control circuitry for controlling the execution of the graphics processor commands. The control circuitry is connected to several registers in the graphics processor which receive instructions from the system processor. These system processor instructions specify commands from both the graphics memory and the graphics processor. The control circuitry provides for the execution of these instructions in these registers in a serial loop fashion.

EP 0 279 231 A2

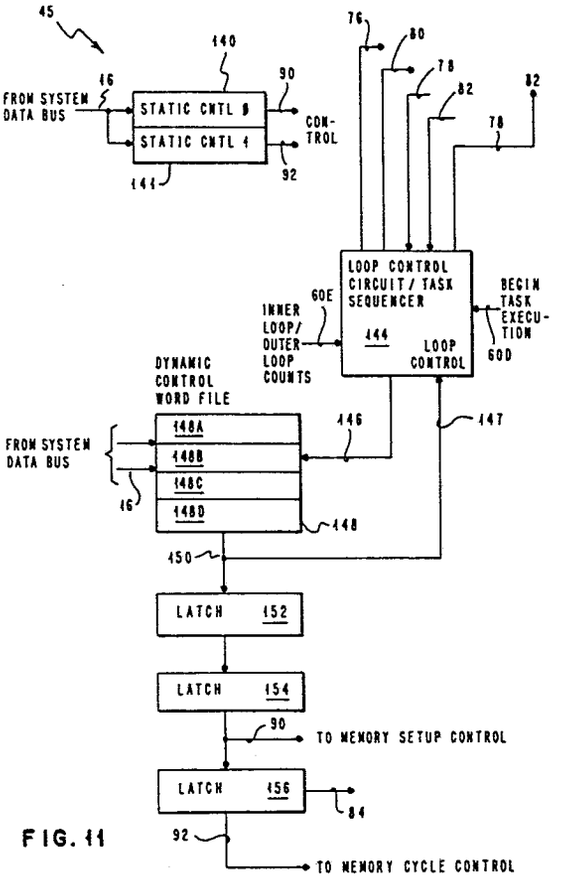

45
140
46
FROM SYSTEM DATA BUS
STATIC CNTL 0
STATIC CNTL 1
90
CONTROL
92
141
76
80
78
82
82
78
LOOP CONTROL CIRCUIT / TASK SEQUENCER
144
LOOP CONTROL
BEGIN TASK EXECUTION
60D
INNER LOOP/ OUTER LOOP COUNTS
60E
DYNAMIC CONTROL WORD FILE
148A
148B
148C
148D
FROM SYSTEM DATA BUS
46
446
447
448
450
LATCH 152
LATCH 154
90
TO MEMORY SETUP CONTROL
LATCH 156
84
92
TO MEMORY CYCLE CONTROL

FIG. 11

A GRAPHICS FUNCTION CONTROLLER FOR A HIGH PERFORMANCE VIDEO DISPLAY SYSTEM

This specification forms part of a set of seven specifications, each relating to a different invention, but having a common exemplary embodiment. To save repetitive description, all seven specification cross-refer and are:-

EP-A- ,(AT9-86-070) entitled "RECONFIGURABLE COUNTERS FOR ADDRESSING IN GRAPHICS DISPLAY SYSTEMS"

EP-A- ,(AT9-86-072) entitled "A GRAPHICS DISPLAY SYSTEM".

EP-A- ,(AT9-86-073) entitled "A GRAPHICS FUNCTION CONTROLLER FOR A HIGH PERFORMANCE VIDEO DISPLAY SYSTEM".

EP-A- ,(KI9-86-029) entitled "HIGH RESOLUTION DISPLAY ADAPTER".

EP-A- ,(YO9-86-051) entitled "RASTER DISPLAY VECTOR GENERATOR".

EP-A- ,(YO9-86-104) entitled "VIDEO ADAPTER WITH IMPROVED DATA PATHING ".

EP-A- ,(YO9-86-105) entitled "A FRAME BUFFER IN OR FOR A RASTER SCAN VIDEO DISPLAY".

The present invention relates to a computer graphics and more specifically to a control apparatus for regulating the execution of commands in both a graphics processor and an external graphics memory simultaneously.

The evolution of computer technology has resulted in the creation of a sophisticated technical area devoted to the representation of graphics information generated by computers. This area is termed computer graphics. One graphics technique commonly used to produce an image is that of producing a set of points and connecting these points with straight lines. The resulting combination of points and straight lines are displayed on the computer graphics terminal display which normally includes a cathode ray tube (CRT). The cathode ray tube includes an array of picture elements. The graphics image is produced by illuminating selected picture elements of the array. This array of picture elements in a display corresponds to the memory locations in an image memory. This image memory is often termed a bit map memory. The corresponding CRT display is termed a bit mapped display.

A very useful function for bit map displays is the ability to move a rectangular block of illuminated picture elements (pels) from one place in the bit map (or display) to another place and to logically combine two subsets of the image array to produce a third image array. Another useful function is that of drawing lines between two points. The technique often used to draw these lines is disclosed in a text entitled Fundamentals of Interactive Computer Graphics by James D. Foley and Andries Van Dam published by Addison Wesley Publishing Company, 1982 and herein incorporated by reference.

Discussions of graphic functions are contained in several IBM Technical Disclosure Bulletins. IBM Technical Disclosure Bulletin, Vol. 28, No. 6, November 1985, entitled "Graphic Bit-Blt Copy Under Mask" discloses a system for making bit boundary block transfers of arbitrary shapes within a frame buffer. IBM Technical Disclosure Bulletin, Vol. 27, No. 8, 1985, entitled "Raster Graphics Drawing Hardware", describes the application of programmable logic arrays to the design of hardware circuitry implementing graphics drawing algorithms. IBM Technical Disclosure Bulletin, Vol. 28, No. 5, October 1985, entitled "Circuit for Updating Bit Map-Memory of A Display Adapter", discloses a circuit for providing bit manipulation flexibility to control picture element data stored in an all points addressable display memory.

An object of the present invention is to provide a system that can be used, inter alia, for controlling a graphics processor and a graphics memory for rapidly accomplishing bit block transfers and line draw functions.

According to the present invention, there is provided a processing system including an external device that responds to any one of several external device commands and performs such command within a single fixed time period and a control processor connected to the external device to provide the external device with the external device commands, the processor including circuitry for executing instructions that generate these external device commands together with at least one internal command, the processor being arranged to perform the current internal command simultaneously with the performance of the current external command by the external device.

In the disclosed embodiment of this invention, the external device is a memory storage device that receives load and store commands from a processor. During the time that these load and store commands are being responded to by the memory, the processor also performs commands internally. These simultaneously executed processor commands perform, in certain instances, addressing computations to compute the addresses for successive memory commands. The simultaneous computation of commands internally increases the throughput capability for the processor and memory combination.

This disclosed embodiment is disclosed in a graphics display system that includes a system

processor, a graphics processor, a graphics memory and a display. The system processor provides graphics information to the graphics processor. The graphics processor provides the memory commands to the graphics memory and includes the capability to execute simultaneously with the memory command executions, internal processor commands. In this embodiment, the graphics processor provides picture element data to the graphics memory. The graphics memory includes a memory array for storing the picture element data. This memory and this image array corresponds directly to the display image to be displayed. The display is connected to the graphics memory for directly accessing the picture element data in this memory array.

Anothe feature of the disclosed embodiment is control circuitry in the graphics processor which is connected to several registers. The registers are accessible by the system processor to receive instructions from the system processor. These instructions specify commands for both the memory and the graphics processor. The control circuitry provides for the execution of these commands in these registers in a serial loop fashion.

The present invention will be described further by way of example with reference to an embodiment thereof as illustrated in the accompanying drawings in which:

Figure 1 is a block diagram illustrating display adapter connected to a processor and monitor;

Figure 2 is a diagram illustrating the organisation of the bit map memory 22;

Figure 3 is a timing diagram illustrating the timing control signals provided to the bit map memory 22 from the pel processor 18;

Figure 4 is an illustration of a portion of a display screen illustrating the display of a 4 x 4 pel matrix upon a grid display;

Figure 5 illustrates the address convention for a 4 x 4 pel matrix;

Figure 6 is a block diagram of the pel processor 18;

Figure 7A illustrates a bit block transfer function;

Figure 7B illustrates a line draw function;

Figure 8A is a flow diagram for the bit block transfer function task;

Figure 8B is a flow diagram fir a line draw task;

Figure 9 is a block diagram of the pel processor 18 control circuitry;

Figure 10 is timing diagram illustrating the simultaneous executions of setup cycles and memory cycles;

Figure 11 is a block diagram of a section of the pel processor 18 control circuitry 45;

Figure 12 is a flow diagram illustrating the operation of the control circuitry in Fig. 11;

Figure 13 is a flow chart illustrating the operation of the memory cycle state circuit 104;

Figure 14 is a timing diagram illustrating the control signals provided by the pel processor 18 control circuitry for a setup cycle;

Figure 15 is a flow diagram illustrating the operation of the control circuitry to execute a read, load, write and store instructions;

Figure 16 is a timing diagram illustrating the control signals provided by the pel processor 18 control circuitry for a memory cycle load;

Figure 17 is a timing diagram illustrating the control signals provided by the pel processor 18 control circuit for a memory cycle store;

Figure 18 is a bit format for a control instruction;

Figure 19 is a bit format for the first static control register;

Figure 20 is a bit format for the second static control register;

Figure 21 illustrates a bit block transfer;

Figure 22 illustrates the instructions and the contents of the static control registers for the execution of a bit block transfer;

Figure 23 illustrates a line draw function;

Figure 24 illustrates the instruction and the content of the static control registers for the execution of a line draw function;

Figure 25A illustrates a bit block transfer operation including the logical combination of two bit blocks;

Figure 25B is an illustration of the actual ORing of a bit block transfer illustrated in Fig. 25A; and

Figure 26 illustrates the instructions and the contents of the static control registers for accomplishing the bit block transfer illustrated in Figs. 25A and 25B.

This adapter circuit is a high resolution graphics display adapter that in the disclosed embodiment drives an IBM 5081 display monitor unit. This circuit provides a resolution of 1024 by 1024 picture elements with 256 simultaneous colours from a palette of 4,096 possible colours. A general description of this display adapter circuit follows.

Display Adapter - General Description

Figure 1 is a block diagram illustrating the display adapter circuit 17 connected for operation. Specifically, display adapter circuit 17 is connected to a system processor 10 by a system I/O bus 11. Additionally, the adapter circuit 17 is connected to a RGB monitor 30 by an output bus 28. The display adapter circuit 17 includes two memories

12A and 12B that are connected to a digital signal processor which is used for circuit resource management and is further used to transform coordinates. In the disclosed embodiment, the digital signal process is of a Harvard architecture requiring separate memories for data and instructions. Memory 12A is an instruction RAM that is loaded with microcode to provide instructions to the signal processor 14. The memory 12B is a data RAM that provides a primary interface between the signal processor 14 and the system processor 10 and also forms the main data store for the signal processor 14. In the disclosed embodiment, 256K bytes of memory are provided for memory 12B. In this embodiment, however, the digital signal processor 14 has an address space of only 128K bytes. Therefore, a bank switching mechanism is provided. Furthermore, in this disclosed embodiment memory located outside of the adapter circuit 17 may be mapped into the digital signal processor 14 address space.

A first-in, first-out buffer 13 is provided for passing sequential display commands from the data memory 12B to the digital signal processor 14. Further, an instruction ROM 15 is connected via bus 16 to provide the power on and self-test instruction microcode programs for the digital signal processor 14.

A pel (picture element) processor 18 is also connected to bus 16. The function of the pel processor 18 is to draw lines, provide for manipulation of areas of data on the display screen and provide bit map memory control. This manipulation of areas on the display screen is termed bit block transfer or BITBLT. The pel processor 18 also includes control and status registers that along with other functions allow the system processor 10 to interrupt, disable or reset the signal processor 14 and allow the signal processor 14 to interrupt the system processor 10.

The pel processor 18 is connected via bus 20 to a bit map memory 22. The bit map memory 22 is organised as 1024 by 1024 by 8 bits. The bit map memory 22 also includes the capability to provide an overlay plane that can be used to provide blinking or highlighting to data on the display.

A video stage 26 is connected to the bit map memory 22 via bus 24 and transforms the data in the bit map memory 22 into a video signal for the video monitor 30. This video stage 26 provides this transformation via a digital to analog circuit. A colour palette circuit is also included in the video stage 25 that provides 256 simultaneously displayable colours from a larger palette of colours. This is accomplished through video lookup tables than translate the value in the bit map to a value with more bits and thus a greater range of colours is provided. With this greater range of values provided by the colour palette, more colours are provided than would be provided by use of the bits in the bit map memory 22 alone.

A hardware cursor 21 is connected to the video stage 26 via bus 24 and provides a full screen cross hair and/or a bit programmable cursor. The full screen cross hair can be programmed to one of several widths. In addition this cross hair can also be scissored (reduced in size) to provide various smaller sizes.

In the disclosed embodiment, the display adapter circuit 17 uses the digital signal processor 14 as a primary interface to the system processor 10. In this embodiment, the digital signal processor is a Texas Instruments TMS 32020 digital signal processor which executes 5 million instructions per second. Therefore, it is well suited to perform such tasks as matrix multiplications which are used to translate, scale and rotate vectors on the screen. The digital signal processor can address a data space of 64K of 16 bit words and an instruction space of the same size. As mentioned earlier, a portion of the data space may be located within the adapter circuit 17 or remote from the adapter circuit 17. The digital signal processor 14 can be interrupted by the system processor 10 or by the pel processor 18. The pel processor 18 can generate interrupts to the digital signal processor 14 or the system processor 10 upon the occurrence of a task complete addition or the condition where a vertical retrace has been started. In addition, the digital signal processor 14 also includes a timer which can be used to control the time between display updates.

The ROM 15 is provided with the initial power-up instruction sequence for the digital signal processor 14. In the disclosed embodiment, the ROM 15 is provided with 16K bytes of information and includes a power-on self-test program and a graphics display adapter emulation program. The power-on self-test program provides an indication that immediately after a power-up condition or a reset condition that the adapter circuit 17 is functioning properly.

The data RAM 12B provides 256K bytes of RAM in the adapter circuit 17 for the signal processor 14 to use as storage. 1K byte of the 256K byte data space is overlayed by the signal processor 14 internal registers. The data memory 12B consists of dynamic RAM, which is refreshed by logic within the display adapter circuit 17. This memory is operated in a page mode so that accesses to two words loaded on the same page (i.e., in the disclosed embodiment in the high order 8 address bits) will require no wait states for the digital signal processor 14. Accesses to words on a new page will result in a single wait state. Thus, locating frequently referenced data in internal registers or

grouped together on a single RAM page will increase performance by not incurring any wait stages. Although the digital signal processor 14 data addressing capacity is limited to 64K words, a bank switching mechanism is provided to extend its address space. This scheme allows a full access to the data memory 12B. Presently, four banks (64K bytes for each bank for a total of 256K bytes) have been implemented. However, the address logic in architecture may provide for up to 16 banks in this disclosed embodiment. In this embodiment, the RAM is dual ported, that is the system processor 10 and the signal processor 14 have concurrent access to it. Since both processors 10 and 14 have easy access to this memory, it provides for a convenient communications channel between the two processors 10 and 14. In this embodiment, the signal processor 14 can also address memory located remote from the display adapter circuit 17 as an extension of this data RAM 12B, by first acting as a first party bus master on bus 11. Both memory on the I/O bus 11 and within the system processor's 10 main memory may be accessed in this manner. The signal processor 14 can put a full 24 bit address on bus 11 and so has a potential of addressing 16 megabytes of memory. The mapping of data space remote from the adapter circuit 17 is controlled by a Bank/Extended Address Register within the signal processor 14. The 16 bit address bus of the signal processor 14 is extended to 24 bits with this register. Access may be made in burst mode, buffered mode, or singly. The length of the burst in burst mode is software controllable. Four to sixteen wait states are required for access to remote memory.

The instruction memory 12A provides 128K bytes of memory in the disclosed embodiment to the digital signal processor 14 to use as an instruction space. This is in addition to the instruction space provided by ROM 15. However when the ROM 15 is mapped into an instruction space, it overlays an equivalent amount of instruction RAM 12A. This is done because the digital signal processor 14 can only address a total instruction space of 128K bytes. The instruction memory 12A consists of dynamic RAM which is refreshed by logic on the adapter circuit 17. The instruction RAM 12A is operated in a page mode so that access to words located on the same page (i.e., the high order 8 bits) requires no wait states for the signal processor 14. Accesses to a new page results in one wait state. Therefore, locating frequently executed code loops on the same page within the instruction memory 12A or within the signal processor 14 internal instruction memory will provide a maximum execution speed. This instruction memory 12A is also dual ported providing concurrent access from the system processor 10 or the signal processor 14.

The FIFO buffer 13 is 1K words in length. When there is space in buffer 13, the system processor 10 may load commands and/or data into this buffer providing access to the digital signal processor 14 which can then sequentially access this information. In this embodiment, display information from the system processor 10 is provided. The buffer 13 includes three flags: empty flag, half full flag and full flag which can be read by the system processor 10 to determine if there is room to write more information into this buffer 13. In addition to the flags, this buffer 13 has three interrupts associated with it. A half full interrupt, a half empty interrupt and a buffer overflow interrupt are provided. The first two may be used to pace write operations to the buffer 13 without polling the flags, while the last would normally be considered an error condition. The digital signal processor 14 also has access to the flags to determine if more information can be read from the buffer 13.

The pel processor 18 assists the signal processor 14 in updating the bit map memory 22 quickly. The pel processor 18 can either draw lines into the bit map memory 22 or manipulate rectangular blocks of data bits (BITBLT) in the bit map memory 22. When line drawing, the pel processor 18 can either be given the end points of the line, with Bresenham's parameters calculated by the pel processor 18, or the end points along with the parameters needed by the Bresenham's incremental line drawing algorithm. The latter approach allows more control over the vector to raster translation and may be useful for special cases such as wide lines. In addition, line attributes of colour and pattern are supported directly by the pel processor 18. Support of the line width attribute requires more intervention by the signal processor 14. Lines may be drawn in replace mode, exclusive OR mode, or line on line mode.

Bit block transfers are also performed by the pel processor 18. Some of the bit block transfers operate with minimal processor intervention while others require more intervention. The bit block transfer includes the operation of an inner loop and an outer loop and the implementation in this embodiment provides that the inner loop may be either horizontally or vertically oriented. This option is particularly useful when transferring images of character strings to the bit map memory 22. In addition, the pel processor 18 has the ability to provide bit block transfers with colour expansion. Colour expansion is defined as the process of taking data in which each active bit represents a pixel of a known colour and a zero indicates transparency (i.e., the frame buffer is not altered for this pixel location). This mode offers a performance advantage as each word of data represents 16

pixels of screen memory rather than 2.

When using colour expansion, a special feature associated with the Direct Write Mask, a capability of the pel processor 18, allows the object being transferred to be rotated in any one of four possible 90 degree orientations.

The digital signal processor 14 or the system processor 10 can define an active region of the bit map memory 22 where drawing occurs. For line draw and block transfer operations, only pels that are to be drawn in this active region will be written to the bit map memory 22. Line draw and block transfer operations resulting in drawing outside of this area will be performed but the resulting pel information will not be written to the bit map memory 22. The use of this active drawing region is termed scissoring.

A further feature of the pel processor 18 is the pick window. This window can be defined to the pel processor 18 and when enabled any access to the frame buffer within this window causes an interrupt to the signal processor 14. This can be used while drawing objects to identify any part of the object which falls within the specified window.

The pel processor is normally controlled by the signal processor 14, however, the system processor 10 may disable the signal processor and control the pel processor 18 directly. The pel processor 18 will be discussed in more detail later.

The bit map memory 22 consists of 1 megabyte of video RAM. The bit map memory 22 is displayed on the screen as a 1024 by 1024 pel image with 8 bits per pel. The pel processor 18 acts as the interface between the system processor 10 or signal processor 14 and the bit map memory 22. Depending upon how some of the bits located within the pel processor 18 are set, the bit map memory 22 will be read as either two horizontally adjacent pels or four horizontally adjacent half pels (wherein a half pel is defined as either the first four or last four bits of a full pel). In all addressing modes, the bit map memory 22 is pel addressable. That is, X and Y address registers in the pel processor 18 are used to indicate the pel being addressed.

The organisation of the bit map memory 22 is shown in Figure 2. The pels are arranged in 4 x 4 squares. Each pel is 8 bits deep. The 8 bits represent 8 planes 400 through 407. Pel memory modules on the same rows share a common row address strobe (RAS) line. Those in the same column share a common address strobe (CAS) line. The same address lines are shared by all the pel memory modules. Both the serial data lines used to refresh the screen and the parallel data lines used to read and write the bit map are connected in columns. Thus, data can be read from one of four layers and loaded into accumulators. Each of the 16 pel memory modules in the 4 x 4 array has its own write enable that is controlled by the direct mask register and the Bresenham line drawing circuits in the pel processor 18.

The mutliple RAS lines 410, 412, 414, and 416 and the multiple CAS lines 418, 420, 422, and 424 are used to strobe different addresses in the pels. This allows the "access" 4 x 4 square word that is addressed by the X and Y pel address registers to be misaligned with the displayed words that are scanned onto the screen. Figure 3 shows the waveforms for the RAS lines 410, 412, 414 and 416 and the CAS lines 418, 420, 422, and 424 that are used to strobe the addresses into the pel memory 22 and align the access word with respect to the displayed words. Note that this pel alignment of 4 x 4 words allows a corner of the square to be placed at the start of any line being drawn and, because each pel memory module has an independent write enable, 4 pels of the line can be drawn simultaneously as illustrated in Figure 4. Figure 5 illustrates the numbering of the pels in the 4 x 4 array.

An overlay plane, actually plane 7 (407 in Figure 2) of the bit map memory 22 can be used in conjunction with the colour palette feature of the video stage 26 to provide highlighting or blinking at a programmable rate. With blinking enabled any pixel with a 1 in this plane will blink at the programmable blink rate. With highlighting enabled a 1 in the overlay plane overrides the normal colour palette process in the video stage 26 and substitutes a colour from a three entry overlay colour palette. Note that the use of the overlay plane will effectively reduce the available colours for the colour palette feature in the video stage 26.

Returning to Figure 1, the video stage 26 includes a colour palette feature. The colour palette translates the 8 bit value stored in the bit map memory 22 into one of 4,096 colours. The output of this colour palette feature provides 4 bits to each of 3 digital to analog converters. The digital to analog converters in turn drive the red, green and blue colour guns of the monitor 30. Each 4 bit section of the look-up table maps the 8 input bits from the bit map into one of sixteen analog output levels. The colour palette feature may be loaded by the signal processor 14 or when the signal processor 14 is disabled, by the system processor 10.

The hardware cursor 21 provides a full screen cross hair and/or a 64 x 64 user programmable cursor. The full screen cross hair can be programmed to one of several widths and scissored. The output of the hardware cursor is fed to the colour palette feature of the video stage 26.

In Figure 1, the system processor 10 provides high level graphics orders to the signal processor 14. Status and other information is passed from the signal processor 14 to the system processor 10.

The signal processor 14 breaks down the high level graphics orders from the system processor 10 into a series of low level graphics commands which are then passed to the pel processor 18 via the input bus 16. This input bus 16 includes address, data and control information. If the signal processor 14 has been disabled, the system processor 10 can transfer low level commands and retrieve data directly from the pel processor 18 by means of the input bus 16. Access to the bit map memory 22 is controlled by the pel processor 18. The accesses to the bit map memory 22 take place over bus 20 which provides address data and control information.

Pel Processor - Description

A block diagram of the pel processor 18 is shown in Figure 6. Control of the bit map memory 22 in execution of low level graphics command is achieved by writing control parameters from either the system processor 10 or the signal processor 14 into the pel processor control logic 44 via the input bus 16. These parameters are decoded within the dynamic control mechanism 45, generating control and timing signals for the other parts of the pel processor circuitry and which are provided via line 60. The endpoint address information for a low level order is communicated to the pel processor 18 by the pel processor input bus 16 and stored in the input queue contained in the endpoint logic 40. Depending on the order being processed (either line draw or bit block transfer), various operations are performed. If a line draw order is being executed, the endpoint data is used to calculate parameters used in executing Bresenham's line draw algorithm in the address count logic circuitry 50. For block transfer operations, the endpoint logic 40 simply queues the input data until this data can be transferred to the address count logic 50. Communications of the endpoint and line draw parameters from the endpoint logic 40 to the address count logic 50 takes place over the address/parameter bus 46. When these parameters have been loaded into the address count logic 50, the endpoint logic 40 is free to accept new endpoint data for the next graphics order. The address count logic 50 uses the parameters to generate the bit map addresses needed to complete the order being executed, and, in addition uses some parameters to sequence the task and determine when the task has been completed.

The address count logic 50 manipulates coordinates in 10 bit fields. The upper 8 bits of the field form the bit map memory addresses 20. The lower 2 bits of both the X and Y coordinates are passed to the RAM control logic 52 via the pel bus 56 where they are decoded into bit map control signals on line 20. These bits are also passed to the data path merge logic 54 via the pel bus 56 where they are used to control data being stored into or retrieved from the bit map memory 22. The data path merge logic 54 serves as the bridge between the system and display processor buses and the bit map memory data bus 20. System processor 10 data can be transferred between or combined with bit map data using the merge logic 54. Data being transferred to and from the system processor 10 is controlled by the data path synchronisation circuitry 42 and passed via the merge bus 48.

The following is a more detailed explanation of the two main graphics tasks that are performed by the pel processor 18. These two tasks are illustrated in Figures 7A and 7B. The bit block transfer task (Figure 7A) consists of moving rectangular blocks of data from a source area of the bit map memory 22 to a destination area of the bit map memory 22. This task is commonly used to "scroll" information on the screen or to display a pop-up menu. Line drawing (Figure 7B) which consists of connecting two points in the bit map memory 22 via straight line, is also a commonly used function. Both of these tasks form the foundation of higher level graphics operations, such as multiple source bit block transfers, pattern lines, polygon drawing, etc. For this reason, it is essential to perform these base functions as effectively as possible.

In Figure 7A, it is desired to move a data block from location 128 to location 136. In order to perform a bit block transfer from the source location 128 to the destination location 136, the following sequence of events must take place within the pel processor 18. Once the pel processor 18 control logic 44 (Figure 6) is loaded with control parameters to perform a bit block transfer operation, the endpoint data for P1 (130) and P2 (138) along with the height parameter (134) and the width parameter (132) are loaded into the endpoint logic 40 (Figure 6). In executing a bit block transfer operation, the endpoint logic serves as an intermediate level of storage, passing the parameters to the address count logic 50 (Figure 6) when the task is initiated. Loading the Y address value of Pz (138) signals the pel processor 18 to begin task execution. At this point, the address and parameter counters within the address count logic begin to access the bit map memory locations along with the width dimension of the bit block transfer, alternately accessing the source, then the destination addresses. When a string of accesses is completed along the width dimension, the address counters are automatically counted and reloaded to begin the next line. This process continues until the bottom of the bit block transfer is reached. The address counters generate

a 10 bit pel address, and the upper 8 bits are used as the bit map memory address 20, while the lower 2 bits 56 are used as the pel decode in the RAM control logic 52 (Figure 6), and the merge logic 54. The merge logic 54 takes the data read in from the source location, aligns it, and passes it out to be stored in the destination locations.

Figure 7B illustrates a line draw task. In order to perform a line draw command, the end points of the line, P1 (150) and P2 (152) are loaded into the endpoint logic 40 (Figure 6). Loading the Y address value of P2 (152) signals the pel processor 18 to begin execution. At this point, the endpoint logic begins to calculate the various Bresenham parameters associated with the line to be drawn. Once this calculation process is finished, the parameters are passed to the address count logic 50. To execute this line draw task, the address count logic will begin generating pel addresses for each pel in the line. The upper 8 bits of the address will serve as the bit map address 20 as before. The lower 2 bits 56 of the pel address are passed to the RAM control logic 52, where they are used to generate the appropriate write enables to draw the line into the bit map.

Figure 8A is a software flow diagram illustrating the bit block transfer function. The pel processor 18 is in the idle state 160 until it receives the bit block transfer end points as illustrated in step 162. If the end points have not yet been received, the pel processor 18 remains in a idle state 160 searching for the end points. When the end points have been received, the pel processor 18 proceeds to step 164 to calculate the inner and outer loop values. In 166 the inner loop incrementing begins with the X pel address being incremented. In step 168 a decision is made as to whether or not the inner loop has been completed. If the inner loop has not been completed, the processor 18 returns to step 166. If the inner loop has been completed, the processor 18 proceeds to step 170 to step the outer loop set the Y pel and reload the inner loop counter. In step 172, a decision is made as to whether or not the outer loop has been completed. If the outer loop has not been complete, when the pel processor 18 returns to step 166. If so, the pel processor 18 returns to the idle state 160.

Figure 8B illustrates a flow chart for the Bresenham line draw algorithm. The Bresenham algorithm has disclosed in the Fundamentals of Interactive Computer Graphics by James D. Foley and Andries Van Dam, published by Addison Wesley Publishing Company, 1982 and appearing on pages 433-435. An over simplified explanation of the Bresenham algorithm is that it determines which picture elements in an array of picture elements should be illuminated to represent an approximation of a straight line in this array of picture elements. Basically the algorithm uses the slope between the two end points to determine a set of parameters that are used to designate which pels are to be activated. In Fig. 8B, the pel processor 18 initially loops between an idle state 174 and a decision state 176 until the line end points have been received. When the line end points have been received, the processor 18 proceeds to step 178 to calculate an initial error term, I1, I2, and the line length. The processor 18 then proceeds to step 180 to determine if the error term is less than 0. If not, the pel processor 18 proceeds to step 184 where the error term is added to I2 and the Y pel address is incremented. The pel processor 18 proceeds to steps 186 to increment the X pel. A decision is made in step 188 to determine if all the pels have been processed. If not, the processor 18 returns to step 180 to examine the error term. If the error term is less than 0, then the processor 18 proceeds to step 182 to add the constant I1 to the error term. The pel processor 18 then proceeds to step 186 as before. When it is determined that all the pels have been processed (step 188), the processor 18 returns to the idle state 174. It should be understood that the slope of the line to be drawn and its direction will determine which address counter is being conditionally counted.

In Fig. 6, the control logic 44 controls the internal operation of the pel processor 18. The control logic is connected to the system processor 10 and the digital signal processor 14 by bus 16. The control logic 44 provides control signals on line 60 to various other blocks illustrated in Fig. 6. Fig. 9 illustrates in greater detail the contents of control logic 44. A section of this control logic illustrated in Fig. 6 is a block 45 that is illustrated in Fig. 9 as a functional control circuit 45 containing command registers, decoding circuitry and task execution circuitry. Controller 45 is connected to control decoder circuitry 100. Controller 45 is further connected to a memory cycle arbitration unit 106 and a memory cycle state machine 104. The memory cycle arbitration unit 106 is connected to a refresh timer 102. The purpose of the circuitry in Fig. 9 is to perform a memory access setup cycle and a memory cycle task. During the setup cycle, addresses, control signals and data are being set to be transmitted on the next ensuing memory cycle. The actual interface with the memory (the bit map memory 22) is accomplished during the memory cycle task.

The control circuit 45 exercises a dynamic control over the executions of these tasks. The control registers contained in controller 45 are loaded from bus 16. These registers contain instructions that provide both commands for internal computation by the pel processor 18 and commands for the bit map memory 22. In operation the com-

mands for the bit map memory 22 and the commands for the pel processor 18 are executed simultaneously. In the disclosed embodiment controller 45 includes four instruction registers and further includes control circuitry that executes these four instructions in a looping fashion. In this embodiment four types of instructions may be executed. These include read, write, load, store. The read and write instructions provide data access between the bit map memory 22 and the system processor 10 or digital signal processor 16 via the pel processor 18. The load and store instructions provide direct access between the bit map memory 22 and the pel processor 18.

When the instructions are loaded into controller 45, execution begins when a signal is received on line 60D. This signal is originated from block 40 (Fig. 6) signifying that all data has been loaded. The first instruction that is to be executed requires both setup cycles and memory cycles. At first control logic 45 determines if a setup is in progress on line 78. If no setup is in progress, then a setup cycle is requested by a signal on line 76 to the memory cycle arbitration unit 106. Likewise, during a memory cycle execution, the control circuitry 45 first determines if a memory cycle is in progress by examining the signal on line 82 and if not providing a signal on line 80 to initiate a memory cycle. During the execution of a setup cycle, setup control signals are provided on lines 90. During the execution of a memory cycle, memory cycle control signals are provided on lines 92. During the execution of either bit block transfer or line draw algorithms, the control circuitry 45 receives the inner and outer loop counts via lines 60E. Line 84 provides the instruction type currently being executed by the controller 45 to the memory cycle state circuit 104.

The function of the memory cycle arbitration unit 106 is to allocate time periods for the bit map memory 22, memory cycle refresh, screen refresh, cycle, setup cycle memory cycle and to determine when memory cycles and setup cycles are in progress. The memory cycle arbitration unit 106 is connected to the memory cycle state circuit 104 via lines 72 and 74 providing an initiation signal on line 72 and completion signal on line 74. The memory cycle state circuit 104 provides the memory cycle state outputs on lines 70 to drive the control decoding circuit 100. The control decoding circuit 100 actually provides the memory control signals to the bit map memory 22. Additionally, control decoding circuitry 100 provides control signals to the remaining functional blocks in the pel processor 18 as illustrated in Fig. 6.

One additional function provided in Fig. 9 is that of refresh timing. Timer 102 is connected to the memory cycle arbitration unit 106 to provide a signal when memory refresh and screen refresh is required.

Fig. 10 is illustrated a timing diagram of the execution of setup cycles and memory cycles for the decoding of instructions and the control register contained in control circuitry 45. At a time 122 the first command is read and a setup cycle requested. At time 123 a setup cycle for the first instruction is begun. At time 129 the execution of the memory cycle for that first instruction is executed (126). At the same time of the execution of the memory cycle for the first instruction (126), the execution of the setup cycle for the second instruction is begun (127).

Fig. 11 illustrates the contents of control circuitry 45. Two registers 140 and 141 termed static control registers are contained in control circuitry 45 to maintain constant control signal data during the operation of the control circuitry 45. These constants are loaded from the bus 16. These registers provide outputs on lines 90 and 92 for the control decoder 100 (Fig. 9) . The heart of the control circuitry 45 is the dynamic control instruction file 148 that includes four instruction registers 148A, 148B, 148C, and 148D. The execution of instructions contained in these four registers 148A-D controlled by the loop control and task sequencer circuit 144. The sequencer 144 receives the begin task execution signal on line 60D and the inner loop/outer loop counts on line 60E. Furthermore, the sequencer circuit 144 provides the request setup signal on line 76 and the request memory cycle signal on line 80. It also receives the setup grant signal on line 78 and the memory cycle grant signal on line 82. The dynamic control instruction file 148 is connected to three latches 152, 154, and 156. When an instruction is to be executed, it is first latched into latch 152 by the sequencer 144. During the latching of this instruction sequencer 144 determines if it is a loop instruction via line 147. Data from register 148 is latched to latch 150 when the control word CWO is read at time 122 (Fig. 10). The data from latch 150 is transferred to latch 154 at the start of the instruction setup cycle 123. Data from latch 154 is transferred to latch 156 at the beginning of the instruction memory cycle 129. The instruction is latched into latch 154. The instruction in latch 154 contains bits which supply signals on lines 90 designating the setup cycle control signals. These signals are provided to the control decoder 100 (Fig. 9). During the next fixed time period, when the same instruction that was in latch 154 is transferred into latch 156, it provides the memory cycle control signals on lines 92 and also a memory cycle designator on line 84 to the memory cycle state machine. Therefore, during any given fixed time period, control signals are being provided on lines 90 and 92

simultaneously.

Fig. 12 is a flow chart illustrating the operation of the control circuitry 45. In Fig. 12 the control circuitry 45 first enters an idle state 300 until it receives the start signal on line 60D. Steps 301 and 300 are repetitively executed until the start signal on lines 60D is received. When the start signal is received, step 302 is executed which initialises certain registers. In step 303, the control circuitry determines from examining line 78 whether a setup cycle is pending. If so, step 305 is executed where the controller waits until the cycle is complete. If not setup is pending, the controller 45 proceeds with step 304 to request a setup cycle by placing a signal on line 76. In step 307, the controller determines if the instruction contained within the latch (152) requires looping. If not, the instruction counter is incremented in step 306 and a determination as to whether the setup cycle has been granted via line 78 is made. If the setup cycle has not yet been granted, then the controller waits in step 311 until this process is granted. And then the controller proceeds to step 318 where it determines if certain control circuitry is busy due to requirements from the system processor 10 or the signal processor 14. If so, then the controller waits in step 319 until the this circuitry is not busy. The execution of the load and store instructions will not require any interface by the system processor 10 or the signal processor 14 and therefore the controller 45 will always pass through this state when executing load and store instructions. If the circuitry is not occupied with communications with the signal processor 14 or the system processor 10, controller proceeds to step 320 to initiate the memory cycle. This is accomplished by first looking at the grant line 82 to determine if the memory cycle is in progress, if not then the memory cycle is requested by placing a signal on line 80. The controller then returns to step 303.

Returning to step 307 if a loop instruction is to be executed, the instruction counter is cleared in step 308 to insure that this execution is looped. In step 310 the controller waits until an indication that it can test the counter state. These counters are contained in block 50 (Fig. 6) and are providing the address counting for the bit block transfer algorithm and line draw algorithm. In step 312 a decision is made whether it is time to test it, if not, the controller loops back to step 310. When the counters can be tested, the controller proceeds to step 313 to determine if the task has been accomplished. If not, the controller proceeds to step 315 to determine if a bit block transfer line has been completed. In other words, if the inner loop counting is complete for this transfer. If so, then the inner loop counter is reloaded in step 316 and the outer loop counter is decremented. If not, or after step 316 has been accomplished, the controller proceeds to step 318 previously discussed.

Returning to step 313, if the task has been completed the controller proceeds to decision step 314 to determine if the the controller 45 is interfacing to either the system processor 10 or the signal processor 14. This wait step is 315 which is similar to step 319 previously discussed. If the circuitry is not busy the controller proceeds to step 317 which is similar to set 320 in the initialisation of the memory cycle. The controller then proceeds back to the idle step 300.

The memory cycle state circuit 104 will sequentially provide one of eight states. Figure 13 is a flow chart illustrating the operation of the memory cycle state circuitry 104. This state circuitry 104 initially starts in step 330 in idle until it receives a signal on line 72 from the memory cycle arbitration unit 106. A decision is made in step 331 when this signal has been received. When such a signal is received, the memory state circuitry 104 then sequentially provides five states 332-336. In step 337 it is determined whether or not the cycle that has been requested is a memory refresh cycle. If so, then the state circuitry 104 returns to idle state 330 and if not the state circuitry 104 provides steps 338-340. It should be understood that these eight states from the memory state circuitry 104 provide information for both the setup cycle and memory access cycles. The state condition output from state circuitry 104 is provided on lines 70 to the control decoder 100.

Fig. 14 illustrates the output signals from the control decoder 100 for each of the eight stages for a setup cycle. During state 1 a reset line 350 is dropped to reset the mask for the write enable control signals. The write enable control signals are used to control not only timing of the memory access but also to control which bits of the provided data is written into memory. The setup clock in line 351 is also activated as is a latch signal on line 352. Line 352 activates a latch to store the current or soon to be old address. Line 353 is activated during the seventh state to latch the new computed address. Line 354 is active during states 2, 3, 4 and 5 and provide clocking signals for the calculation of addresses. At the end of state 5, line 356 provides the test signal to the controller 45 to indicate that the counters can now be tested (i.e., step 312 in Fig. 12). In step 357 a clear signal is active during state 2 to provide the grant signal on line 78.

All of the instructions such as read, write, load and store include a setup cycle. All setup cycles are conducted in accordance with the timing diagram of Fig. 14. The read and write instructions entail access to either the signal processor 14 or the system processor 10. The load and store

instructions only entail activities between the pel processor 18 and the bit memory 22. Fig. 15 is a flow chart illustrating the execution of the read, write, load and store instructions. To avoid confusion only the load and store instructions will be discussed since the read and write instructions do include load and store but also require interfacing to the system processor 10 or the signal processor 14.

The timing diagrams for the load and store instructions are provided in Figs. 16 and 17. It should be understood that these cycles are similar to the setup cycle timing in Fig. 14. However, the control signals provided by the control decoder 100 are as illustrated in Figs. 16 and 17 as a result of the states of the memory cycle state circuit 104. Referring to Fig. 16, two row address strobe timing signals 360 and 361 and two column address strobe timing signals 362 and 363 are provided to access the bit map memories. Line 364 is provided to pass data from the memory into the pel processor 18. Line 365 is a data latch signal. Line 366 provides the memory cycle grant on line 82. Line 367 designates the availability of the row address and column address signals. Line 368 is used in column and row address calculations.

Fig. 17 is the timing diagram for a store cycle and is similar to Fig. 16 and will not be discussed further.

Fig. 18 is a bit format listing for instructions that are contained in the control registers 148. Bit position 350 specifies whether or not the instruction is a loop instruction. This is used in decision step 307 of Fig. 12. Bit positions 352 specify the instruction type, i.e., read, write, load or store. This is used to provide the signal on line 84 (Fig. 9). Bit positions 353 specify the data paths to be activated in the data path merge logic 54.

Fig. 19 illustrates the bit formats for the contents of the static control register 140. These bits provide information to the control decoder 100 (Fig. 9) on lines 90 for the setup cycle.

Fig. 20 provides the data format for the static control register 141. These bits specify the types of memory access to the control decoder 100 on the memory cycle control lines 92.

In Fig. 21 an illustration is given of a bit block transfer of information from a source area 360 to a destination area 362. In Fig. 22 the contents of the control instruction register 148 is illustrated to accomplish this transfer of Fig. 21. Two instructions are required as illustrated by instructions 364 and 366 operating in a loop fashion. The contents of the static control registers 140 and 141 are illustrated in block 368.

Fig. 23 illustrates the drawing of a single line 370. In Fig. 24 this is accomplished by a single instruction in the control register 148 consisting of an instruction illustrated as 372. The contents of the static control registers 140 and 141 are illustrated in block 374.

Fig. 25A illustrates a bit block transfer operation including two source areas 376 and 378 that are combined in a logic operation 380 to provide a destination area 382. An example is illustrated in Fig. 25B combining the lines of 376' plus 378' in an OR operation to result in a composite screen 382'. To accomplish this task, the dynamic control register instructions are illustrated in Fig. 26 as blocks 384, 386 and 388. Contents of the static control registers 140 and 141 are illustrated in block 400.

Although this embodiment has been described with reference to this specific embodiment, this description is not meant to be construed in a limiting sense. Various modifications of the disclosed embodiment, as well as other embodiments of the invention, will become apparent to those persons skilled in the art.

Claims

1. A processing system including an external device that responds to any one of several external device commands and performs such command within a single fixed time period and a control processor connected to the external device to provide the external device with the external device commands, the processor including circuitry for executing instructions that generate these external device commands together with at least one internal command, the processor being arranged to perform the current internal command simultaneously with the performance of the current external command by the external device.

2. A processing system as claimed in Claim 1, wherein plural internal commands are generated for each of at least some of the external commands, the internal commands being performed in series in the processor during the performance of the external command by the external device.

3. A processing system as claimed in Claim 2, wherein the series of internal commands is performed within the fixed time period.

4. A processing system as claimed in Claim 3, wherein the processing system includes registers which specify the external and internal commands.

5. A processing system as claimed in any preceding claim, wherein the external device is a memory sub-system including means for executing memory commands for storing and retrieving the data during the single fixed time period and the internal commands include commands that compute address information for the memory.

6. A processing system as claimed in claim 5, being a graphics display system, where a systems processor computes graphics information to be displayed; the control processor being a graphics processor connected to the system processor for receiving the graphics information and for computing picture element data for display; the memory sub-system being a graphics memory for receiving and storing the picture element data; there being display means connected to the graphics memory means for displaying the picture element data from the graphics memory means.

7. A graphics display system as claimed in Claim 6, wherein the graphics processor includes control means connected to at least two of the registers for executing the instructions in the registers in a serial loop.

8. A graphics display system as claimed in Claim 7, wherein the graphics processor registers receive instructions from the system processor and are connected to a plurality of serially connected latches each connected to respective decoding circuit means, the latches for receiving instructions from one of the control registers from the control means and for providing its instruction to its respective decoding circuit means for providing a plurality of control signals simultaneously specifying memory commands and internal processor commands.

9. A graphics display system as claimed in Claim 8, wherein each instruction within the control registers is serially circulated through the latches by the control means.

10. A graphics display system as claimed in Claim 9, wherein a instruction is loaded into a latch each fixed time period.

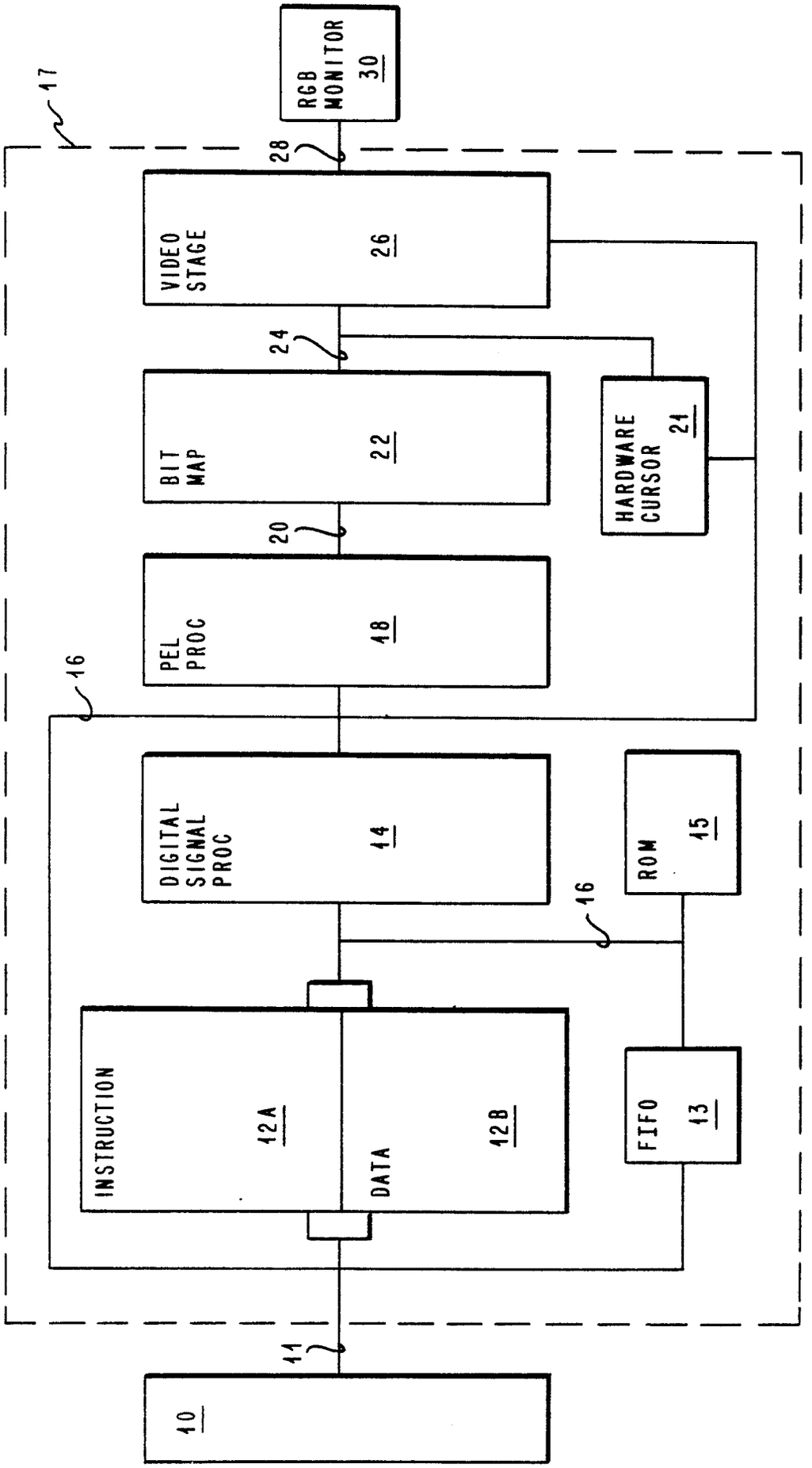

RGB MONITOR 30
17
28
VIDEO STAGE 26
24
BIT MAP 22
HARDWARE CURSOR 21
20
PEL PROC 18
16
DIGITAL SIGNAL PROC 14
ROM 15
16
INSTRUCTION 12A
DATA 12B
FIFO 13
11
10

FIG. 1

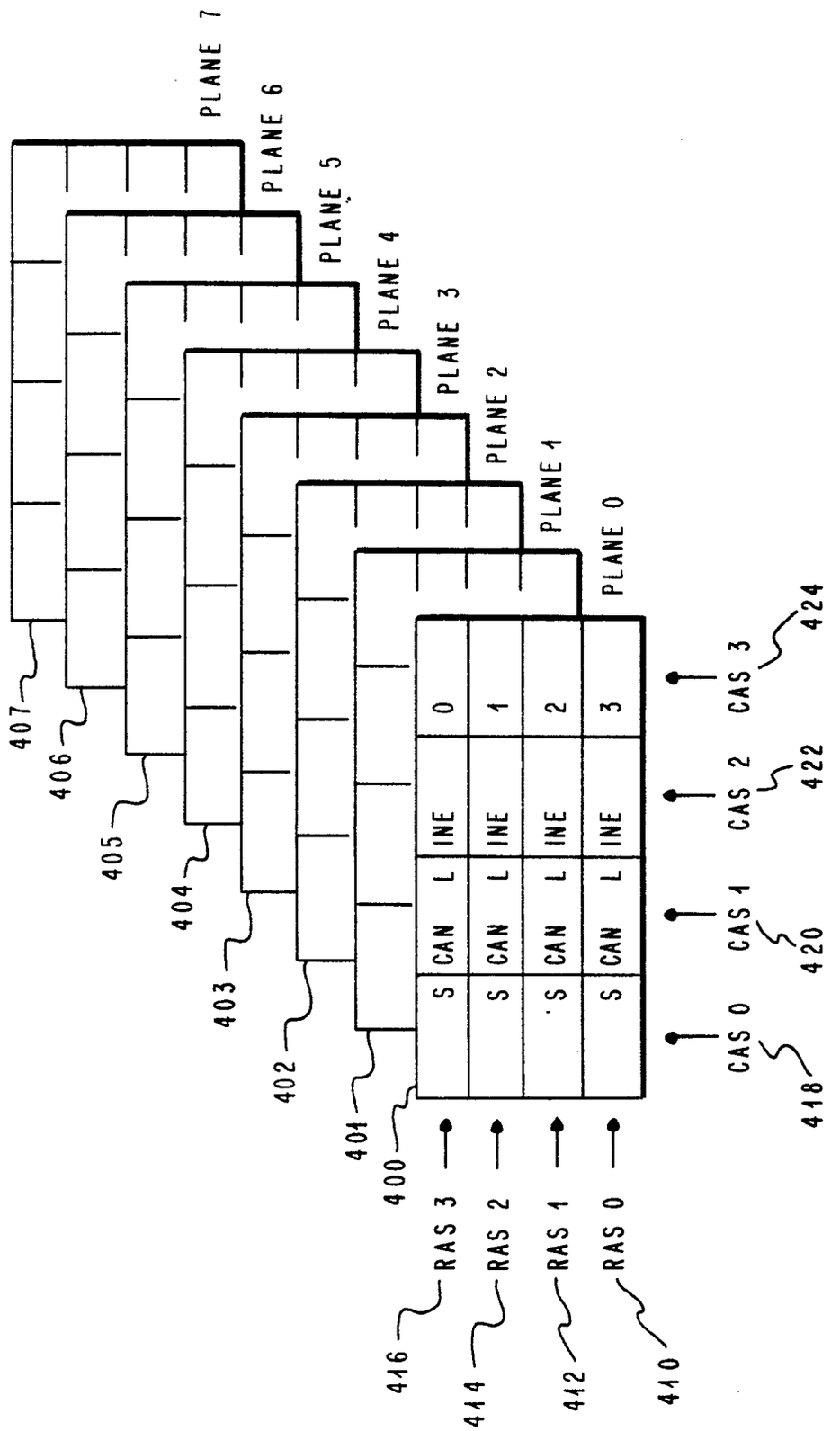

PLANE 7
PLANE 6
PLANE 5
PLANE 4
PLANE 3
PLANE 2
PLANE 1
PLANE 0
407
406
405
404
403
402
401
400
S
CAN
L
INE
0
1
2
3
RAS 3
RAS 2
RAS 1
RAS 0
416
414
412
410
CAS 0
CAS 1
CAS 2
CAS 3
418
420
422
424

FIG. 2

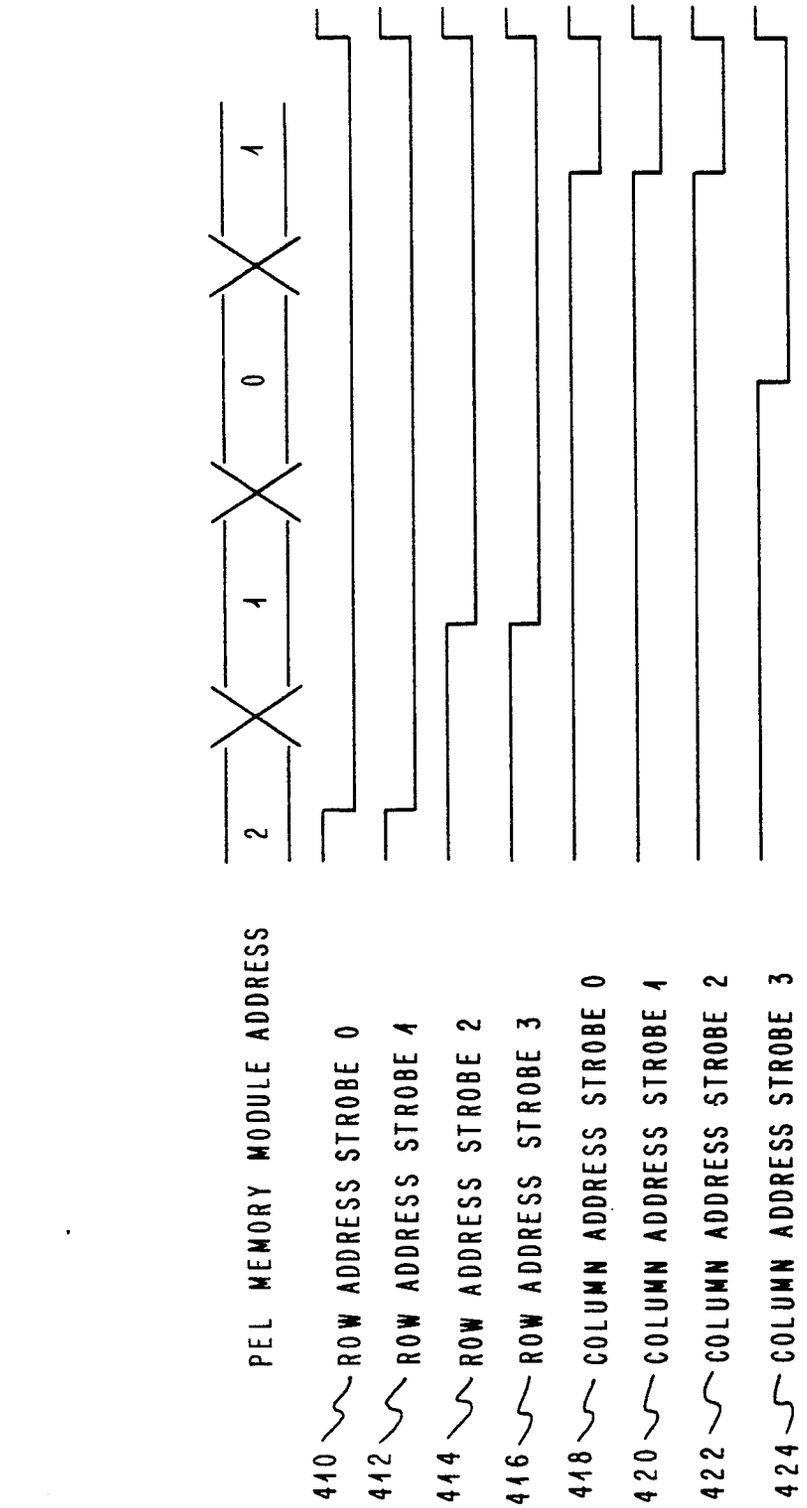

PEL MEMORY MODULE ADDRESS
2
1
0
1
410 ROW ADDRESS STROBE 0
412 ROW ADDRESS STROBE 1
414 ROW ADDRESS STROBE 2
416 ROW ADDRESS STROBE 3
418 COLUMN ADDRESS STROBE 0
420 COLUMN ADDRESS STROBE 1
422 COLUMN ADDRESS STROBE 2
424 COLUMN ADDRESS STROBE 3

FIG. 3

3	2	1	0
7	6	5	4
11	10	9	8
15	14	13	12

FIG. 5

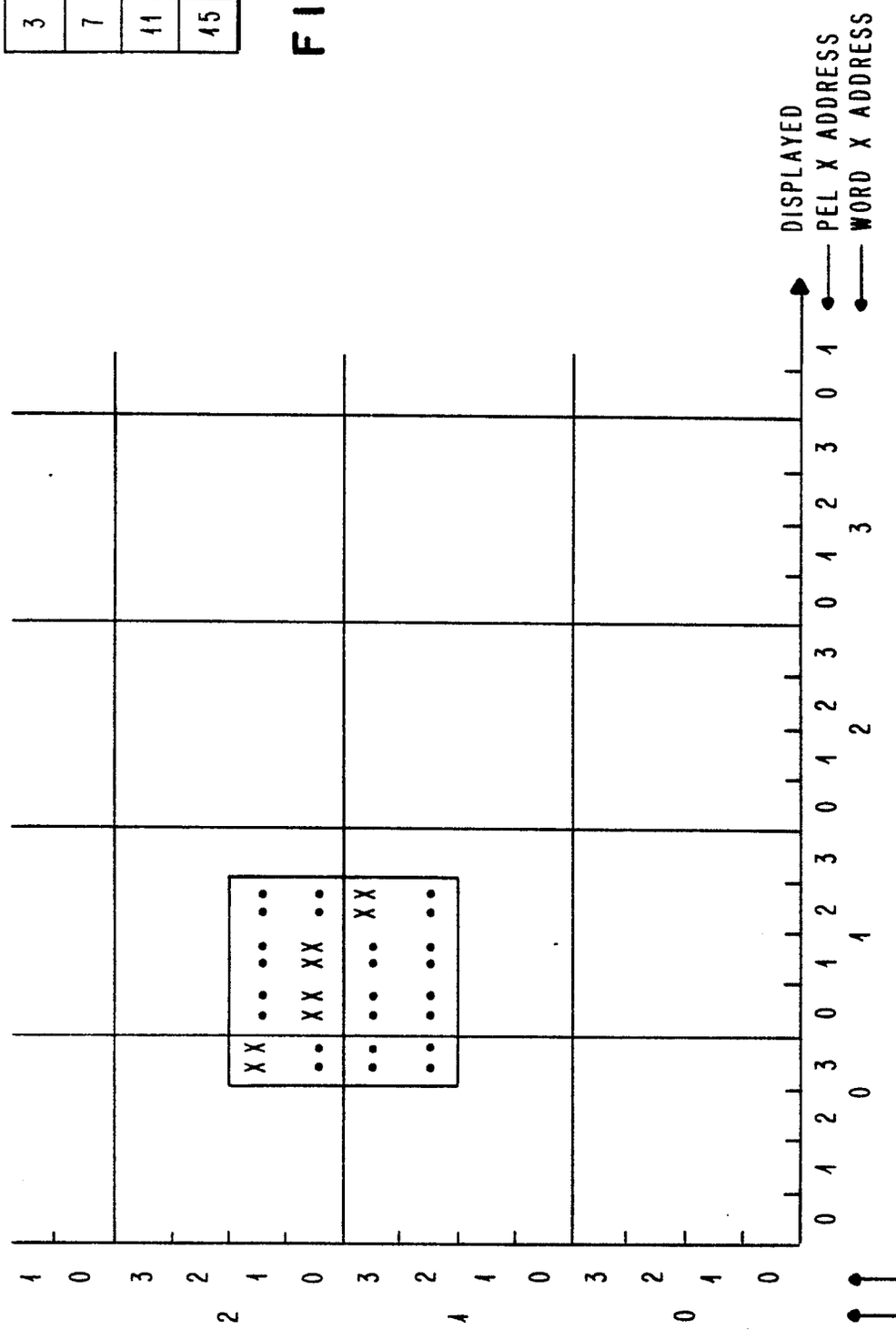


FIG. 4

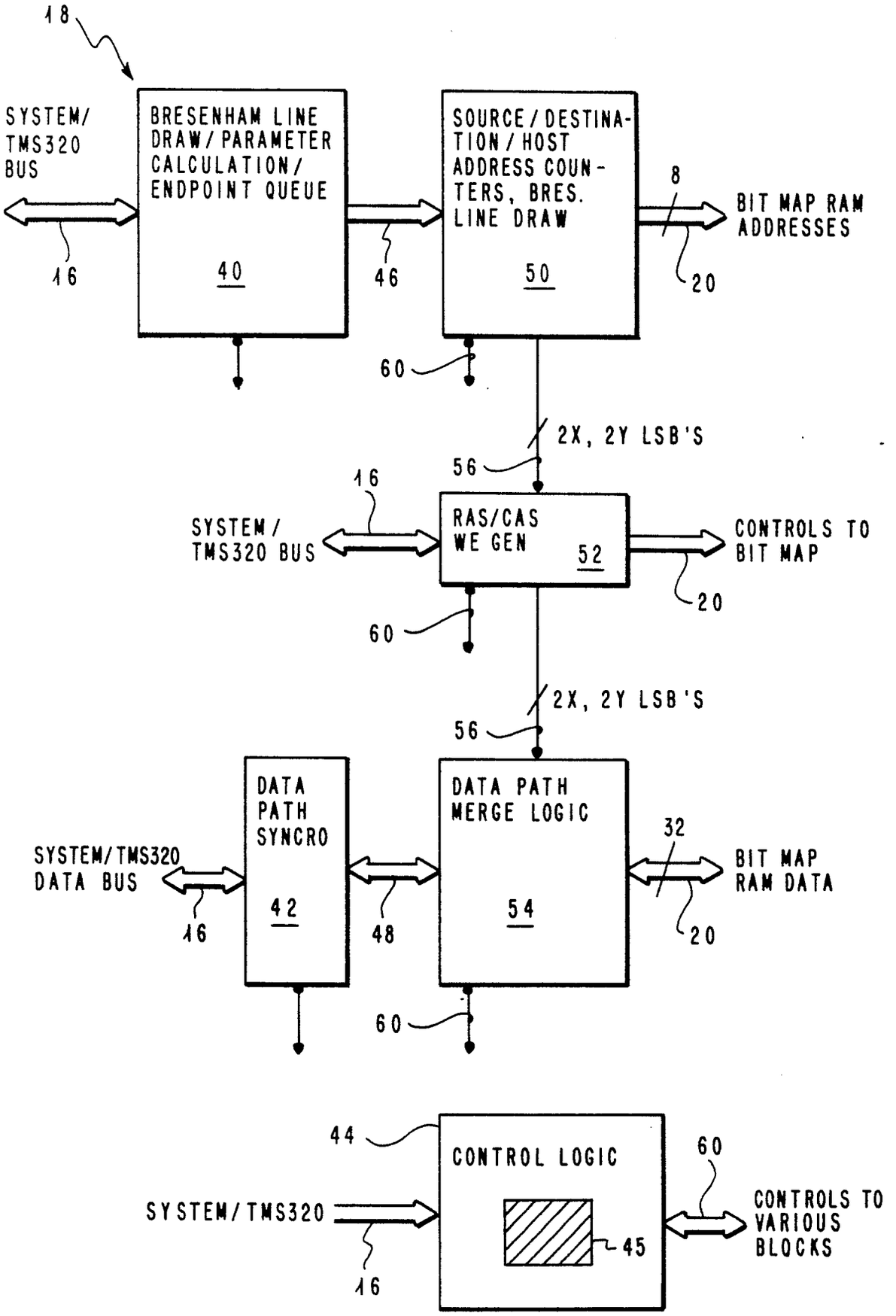

18
SYSTEM/
TMS320
BUS
16
BRESENHAM LINE
DRAW/PARAMETER
CALCULATION/
ENDPOINT QUEUE
40
46
SOURCE/DESTINA-
TION/HOST
ADDRESS COUN-
TERS, BRES.
LINE DRAW
50
8
BIT MAP RAM
ADDRESSES
20
60
2X, 2Y LSB'S
56
16
SYSTEM/
TMS320 BUS
RAS/CAS
WE GEN
52
CONTROLS TO
BIT MAP
20
60
2X, 2Y LSB'S
56
DATA
PATH
SYNCRO
42
DATA PATH
MERGE LOGIC
54
32
SYSTEM/TMS320
DATA BUS
16
48
BIT MAP
RAM DATA
20
60
44
CONTROL LOGIC
60
SYSTEM/TMS320
45
CONTROLS TO
VARIOUS
BLOCKS
16

FIG. 6

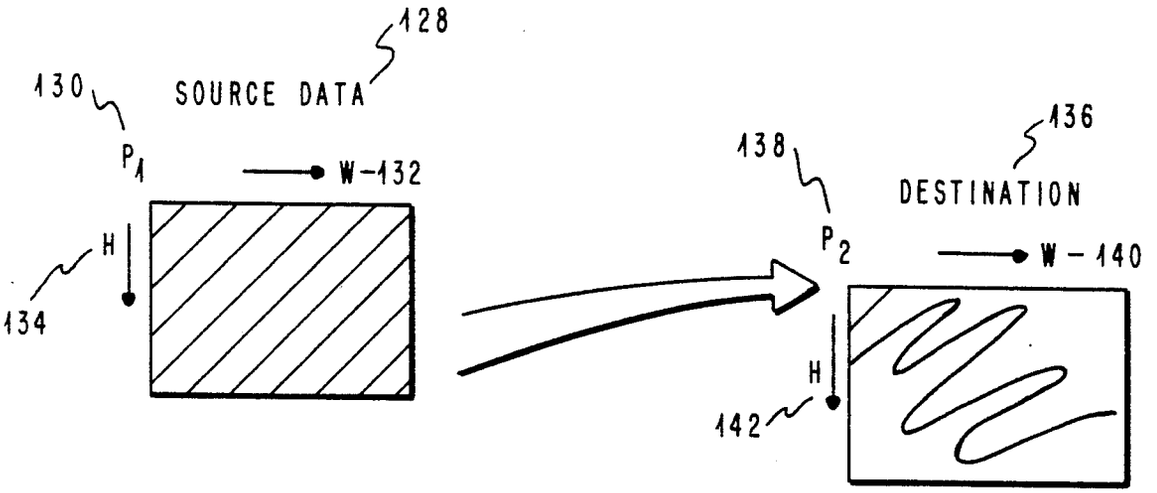

128
130
SOURCE DATA
P1
W-132
H
134
138
136
DESTINATION
P2
W-140
H
142

FIG. 7A

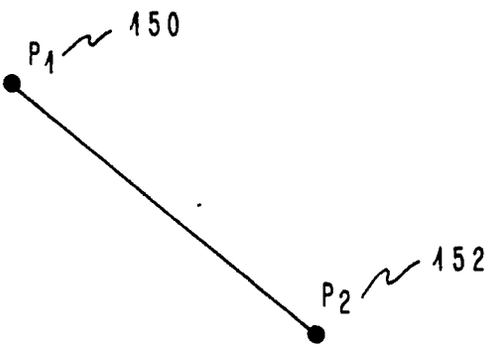

P1
150
P2
152

FIG. 7B

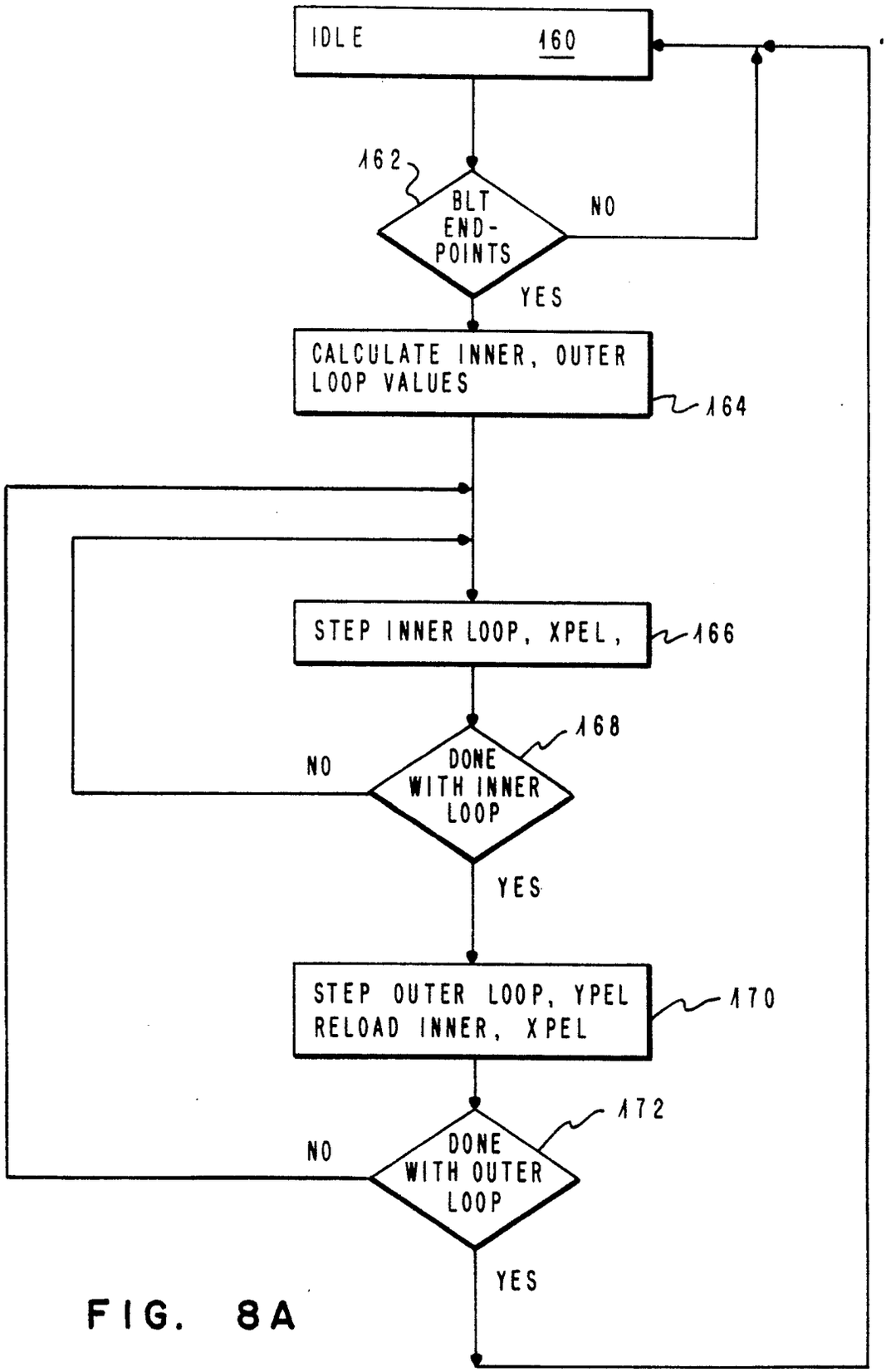

IDLE 160
162
BLT END-POINTS
NO
YES
CALCULATE INNER, OUTER LOOP VALUES
164
STEP INNER LOOP, XPEL,
166
168
DONE WITH INNER LOOP
NO
YES
STEP OUTER LOOP, YPEL RELOAD INNER, XPEL
170
172
DONE WITH OUTER LOOP
NO
YES

FIG. 8A

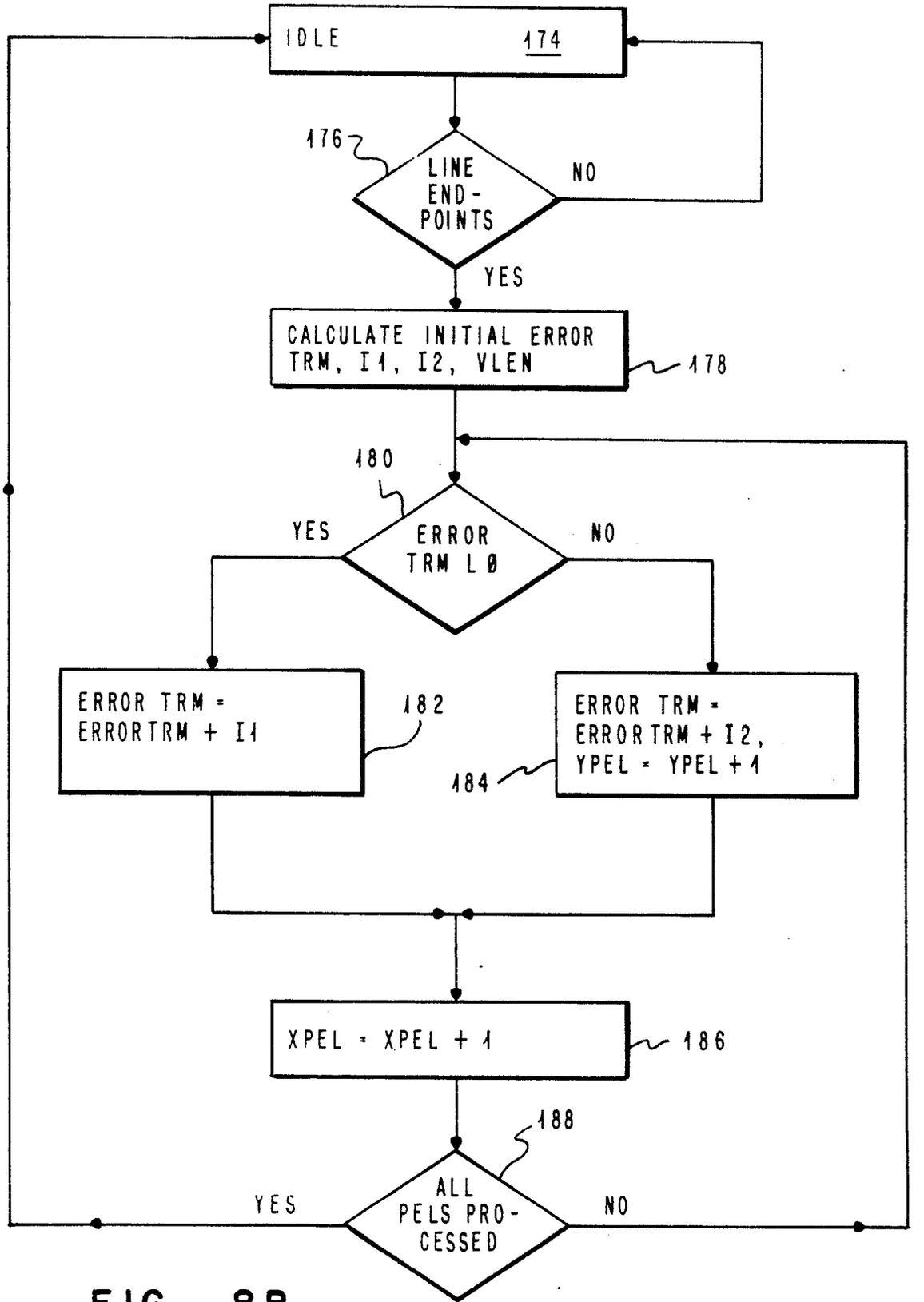

IDLE
174
176
LINE END-POINTS
NO
YES
CALCULATE INITIAL ERROR TRM, I1, I2, VLEN
178
180
YES
ERROR TRM L Ø
NO
ERROR TRM = ERRORTRM + I1
182
ERROR TRM = ERRORTRM + I2, YPEL = YPEL + 1
184
XPEL = XPEL + 1
186
188
ALL PELS PROCESSED
YES
NO

FIG. 8B

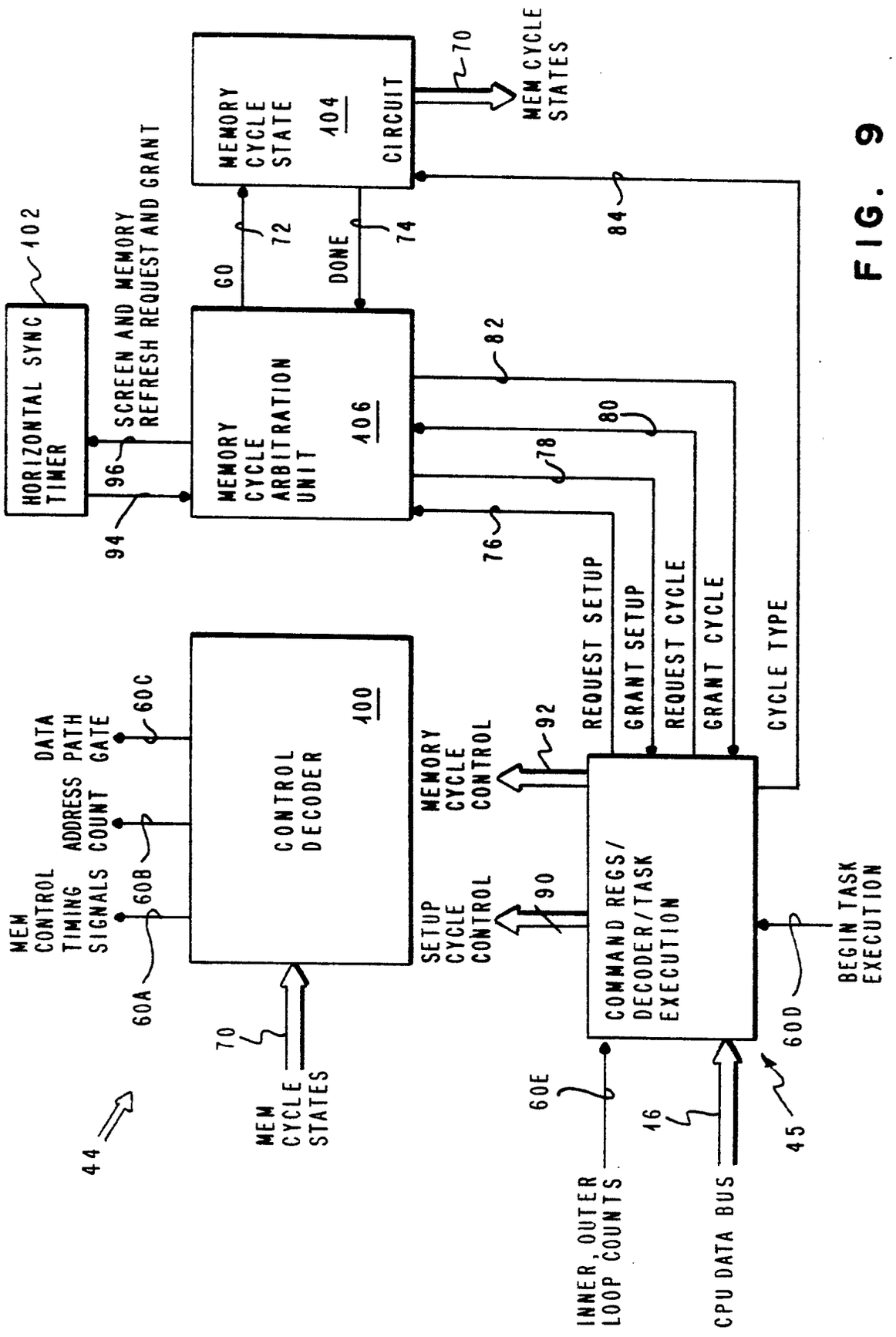

HORIZONTAL SYNC TIMER
102
SCREEN AND MEMORY REFRESH REQUEST AND GRANT
96
94
MEMORY CYCLE ARBITRATION UNIT
406
GO
72
DONE
74
MEMORY CYCLE STATE CIRCUIT
404
70
MEM CYCLE STATES
84
82
80
78
76
MEM CONTROL TIMING SIGNALS
ADDRESS COUNT
DATA PATH GATE
60B
60C
60A
CONTROL DECODER
400
MEMORY CYCLE CONTROL
92
REQUEST SETUP
GRANT SETUP
REQUEST CYCLE
GRANT CYCLE
CYCLE TYPE
SETUP CYCLE CONTROL
90
COMMAND REGS/ DECODER/TASK EXECUTION
BEGIN TASK EXECUTION
60D
45
60E
INNER, OUTER LOOP COUNTS
16
CPU DATA BUS
70
MEM CYCLE STATES
44

FIG. 9

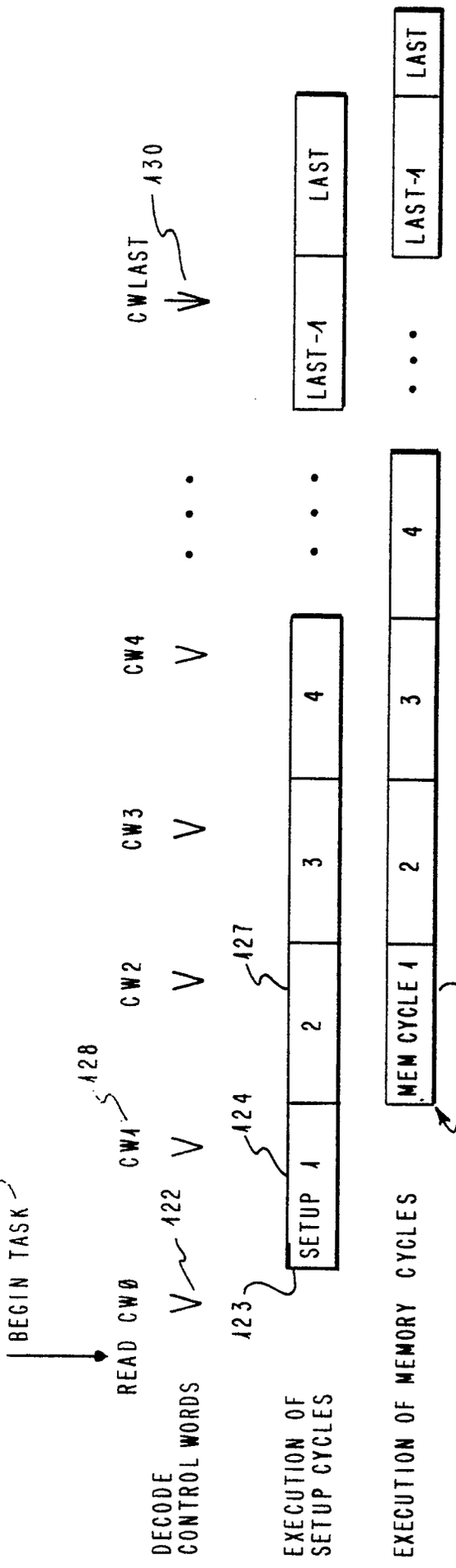

BEGIN TASK
120
READ CW0
CW1
CW2
CW3
CW4
CWLAST
130
122
128
DECODE CONTROL WORDS
123
124
127
EXECUTION OF SETUP CYCLES
SETUP 1
2
3
4
LAST-1
LAST
EXECUTION OF MEMORY CYCLES
MEM CYCLE 1
2
3
4
LAST-1
LAST
126
129

FIG. 10

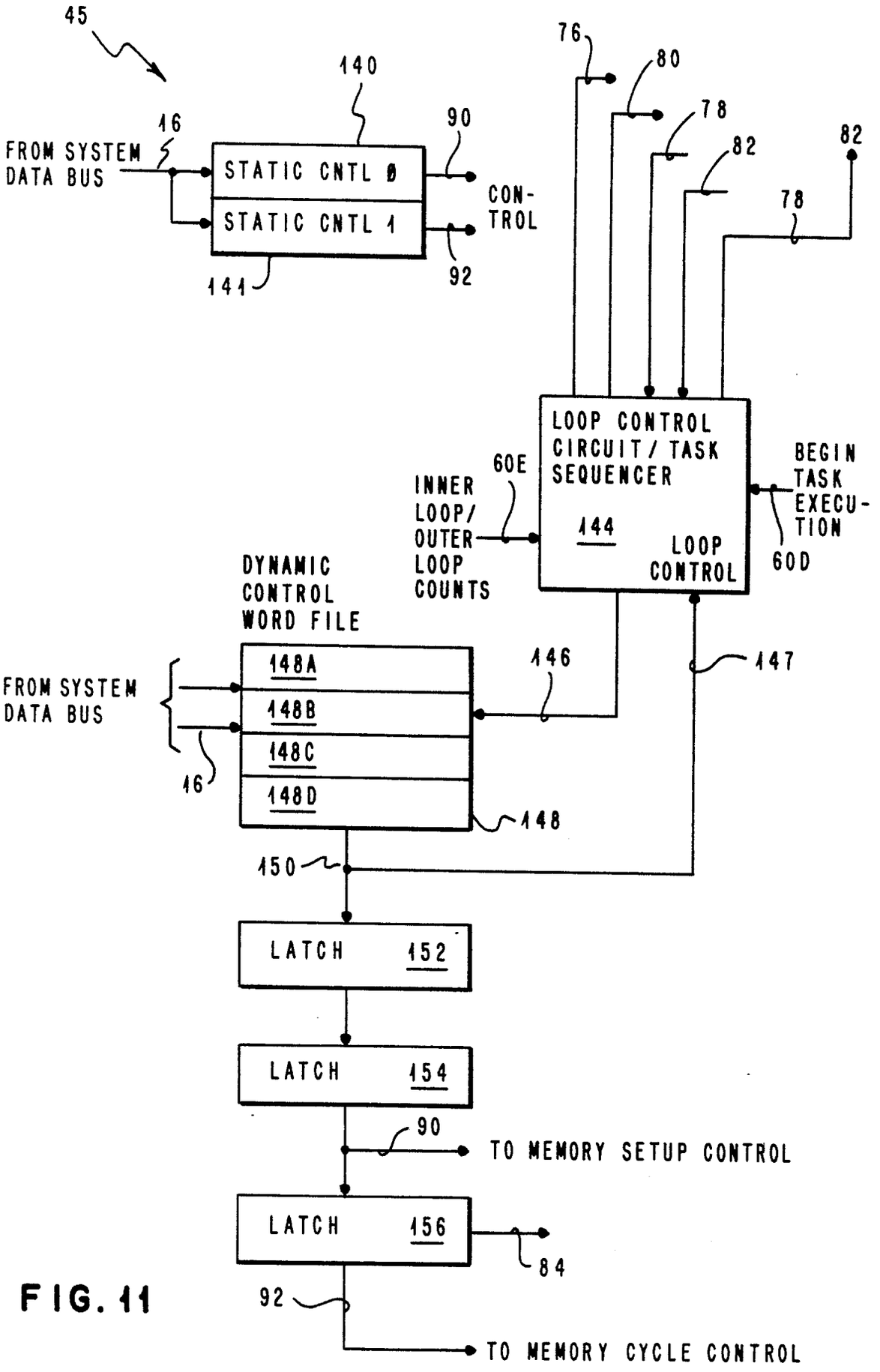

45
140
16
FROM SYSTEM DATA BUS
STATIC CNTL 0
STATIC CNTL 1
90
CON-TROL
92
141
76
80
78
82
82
78
LOOP CONTROL CIRCUIT / TASK SEQUENCER
144
LOOP CONTROL
60E
INNER LOOP/ OUTER LOOP COUNTS
BEGIN TASK EXECU-TION
60D
DYNAMIC CONTROL WORD FILE
148A
148B
148C
148D
148
146
147
FROM SYSTEM DATA BUS
16
150
LATCH 152
LATCH 154
90
TO MEMORY SETUP CONTROL
LATCH 156
84
92
TO MEMORY CYCLE CONTROL

FIG. 11

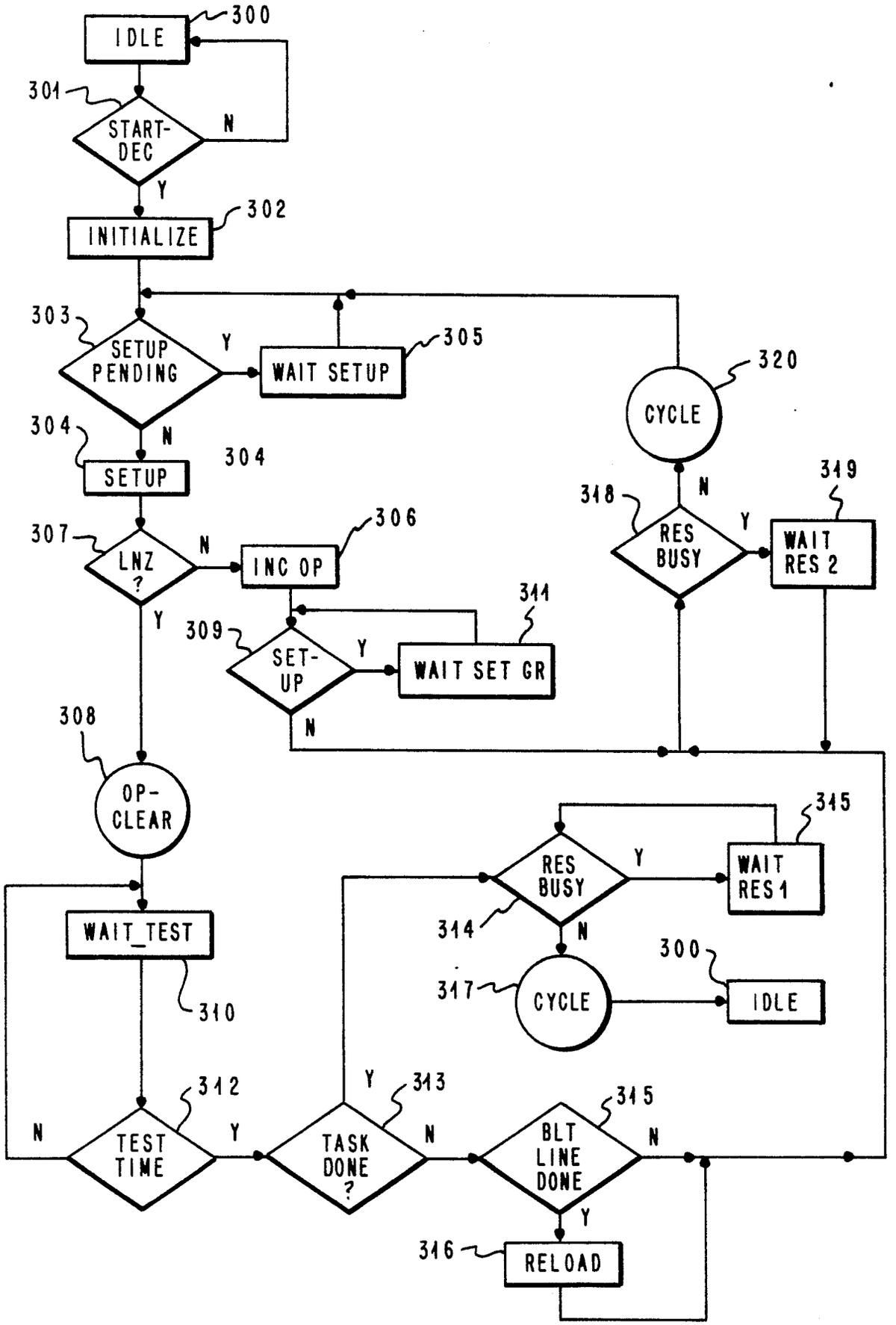

IDLE
300
301
START-DEC
N
Y
INITIALIZE
302
303
SETUP PENDING
Y
WAIT SETUP
305
304
N
SETUP
304
307
LNZ ?
N
INC OP
306
Y
309
SET-UP
Y
WAIT SET GR
311
N
308
OP-CLEAR
WAIT_TEST
310
312
N
TEST TIME
Y
Y
313
TASK DONE ?
N
315
BLT LINE DONE
N
Y
316
RELOAD
320
CYCLE
N
318
RES BUSY
Y
WAIT RES 2
319
345
RES BUSY
Y
WAIT RES 1
314
N
300
317
CYCLE
IDLE

FIG. 12

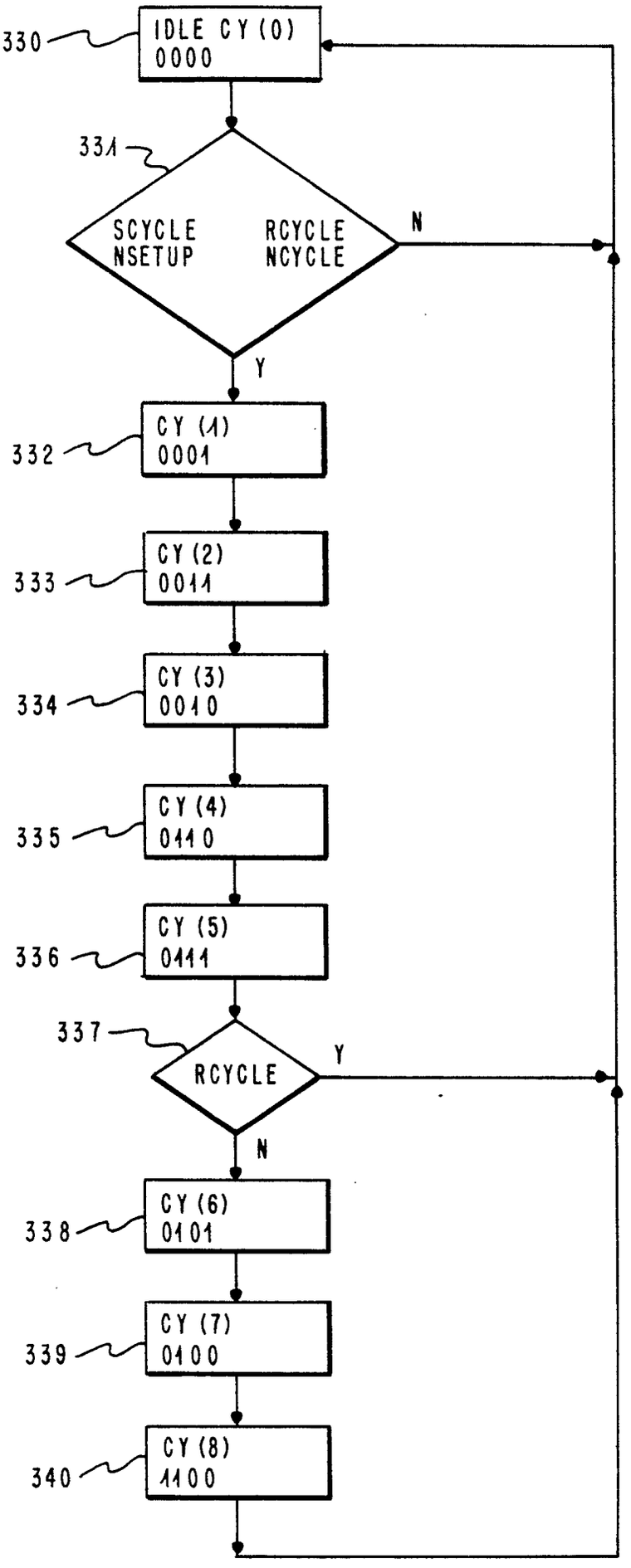

330
IDLE CY (0)
0000
331
SCYCLE
NSETUP
RCYCLE
NCYCLE
N
Y
332
CY (1)
0001
333
CY (2)
0011
334
CY (3)
0010
335
CY (4)
0110
336
CY (5)
0111
337
RCYCLE
Y
N
338
CY (6)
0101
339
CY (7)
0100
340
CY (8)
1100

FIG. 13

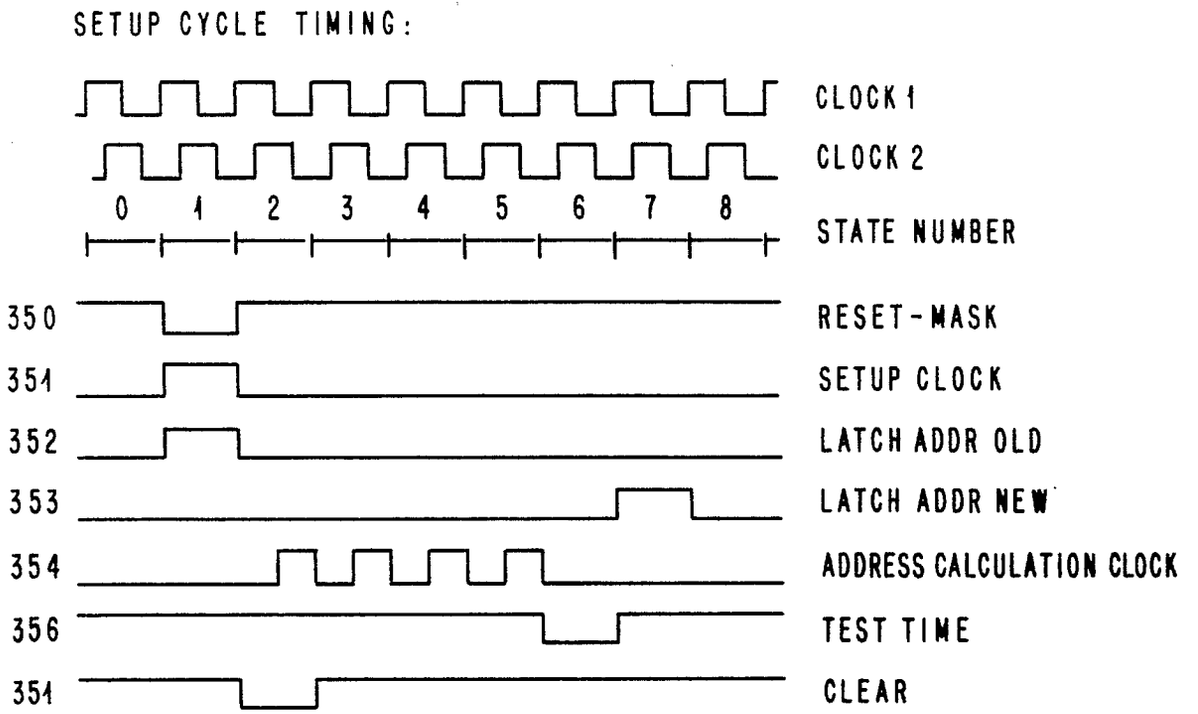

SETUP CYCLE TIMING:
CLOCK 1
CLOCK 2
0 1 2 3 4 5 6 7 8
STATE NUMBER
350
RESET-MASK
351
SETUP CLOCK
352
LATCH ADDR OLD
353
LATCH ADDR NEW
354
ADDRESS CALCULATION CLOCK
356
TEST TIME
351
CLEAR

FIG. 14

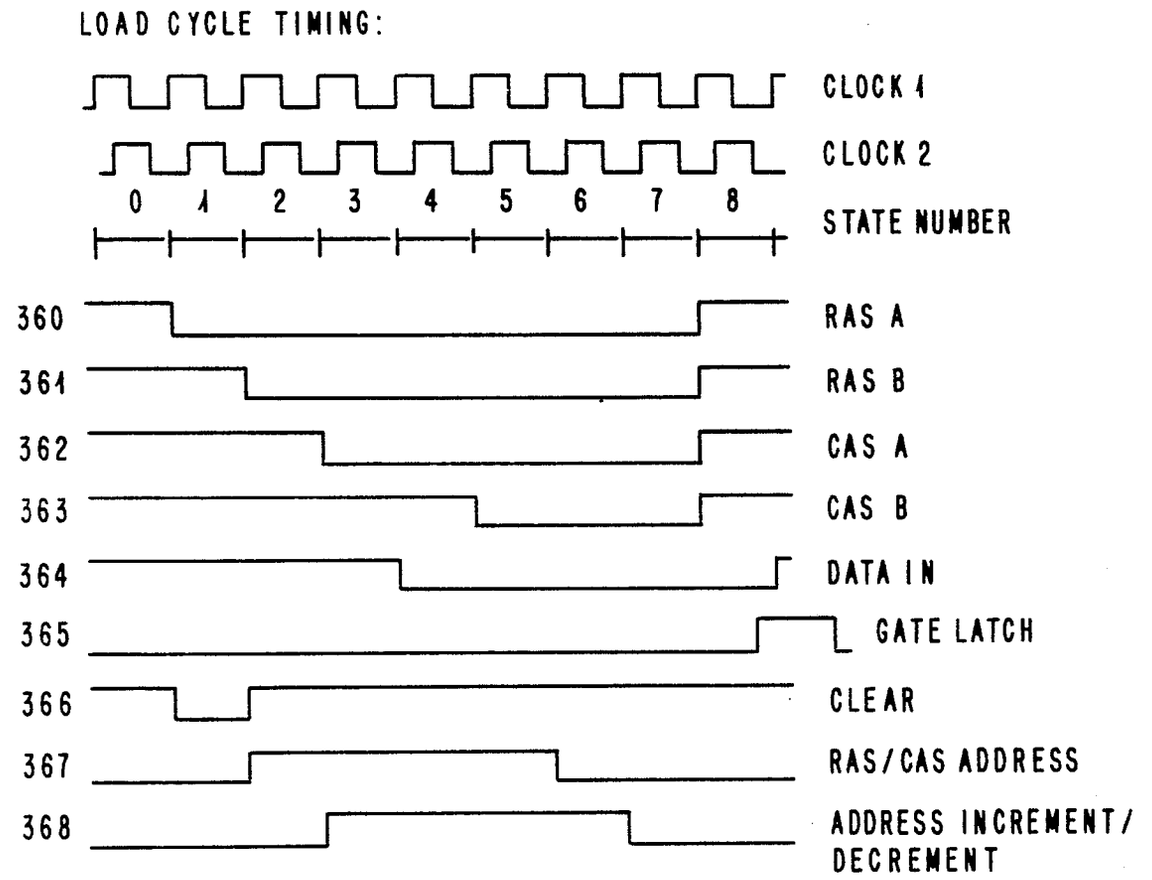

LOAD CYCLE TIMING:
CLOCK 1
CLOCK 2
0 1 2 3 4 5 6 7 8
STATE NUMBER
360
RAS A
361
RAS B
362
CAS A
363
CAS B
364
DATA IN
365
GATE LATCH
366
CLEAR
367
RAS/CAS ADDRESS
368
ADDRESS INCREMENT/
DECREMENT

FIG. 16

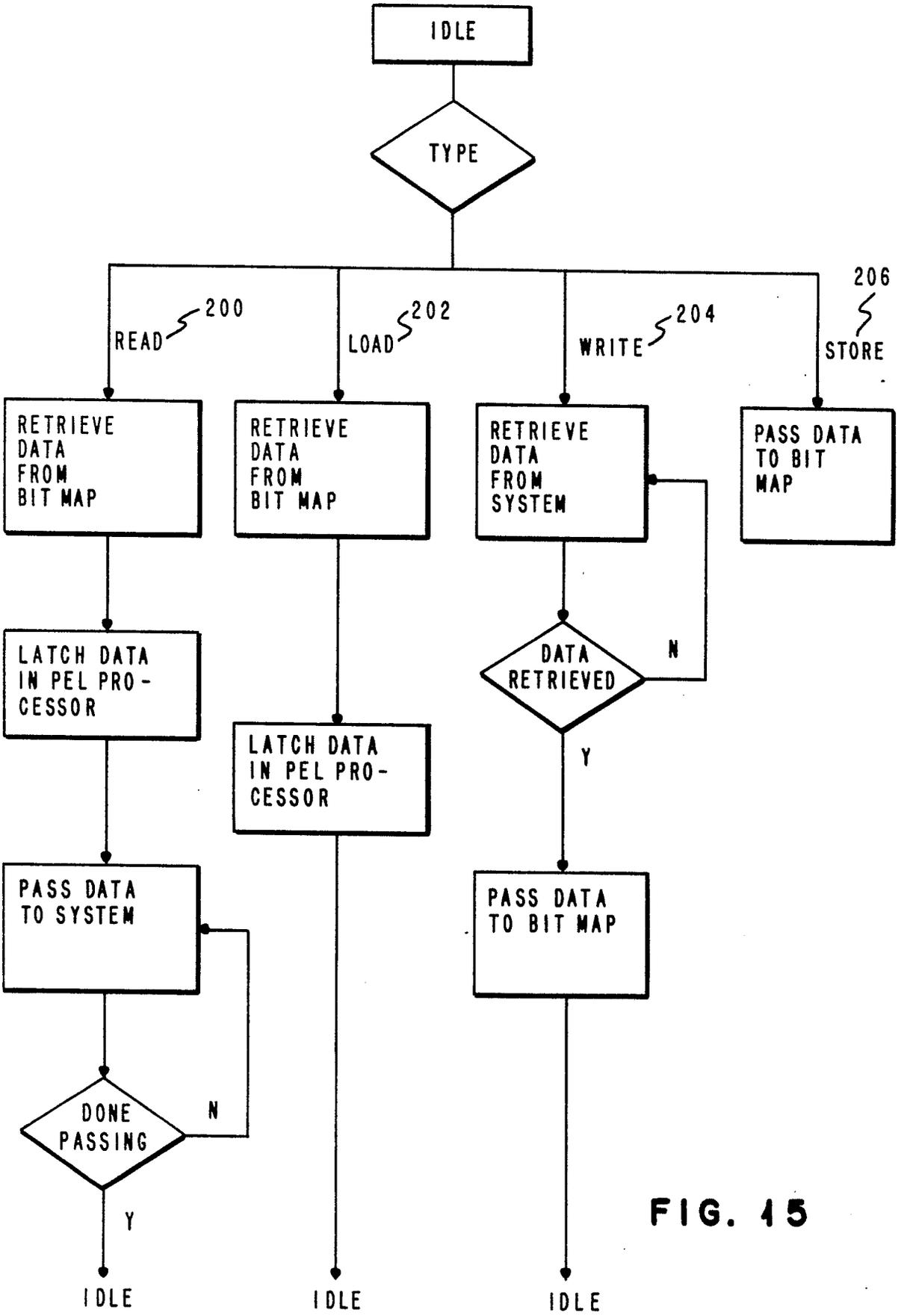

IDLE
TYPE
READ
200
LOAD
202
WRITE
204
206
STORE
RETRIEVE DATA FROM BIT MAP
RETRIEVE DATA FROM BIT MAP
RETRIEVE DATA FROM SYSTEM
PASS DATA TO BIT MAP
LATCH DATA IN PEL PRO-CESSOR
LATCH DATA IN PEL PRO-CESSOR
DATA RETRIEVED
N
Y
PASS DATA TO SYSTEM
PASS DATA TO BIT MAP
DONE PASSING
N
Y
IDLE
IDLE
IDLE

FIG. 15

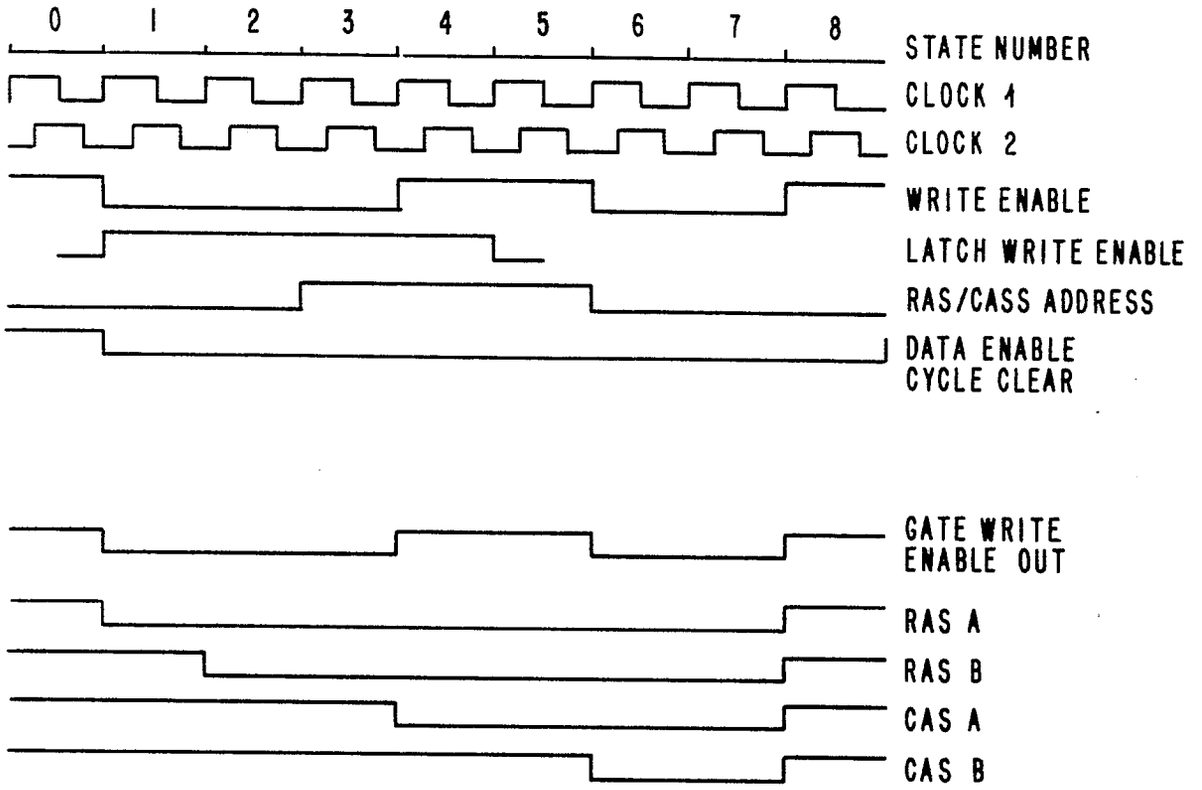

0
1
2
3
4
5
6
7
8
STATE NUMBER
CLOCK 1
CLOCK 2
WRITE ENABLE
LATCH WRITE ENABLE
RAS/CASS ADDRESS
DATA ENABLE
CYCLE CLEAR
GATE WRITE
ENABLE OUT
RAS A
RAS B
CAS A
CAS B

FIG. 17

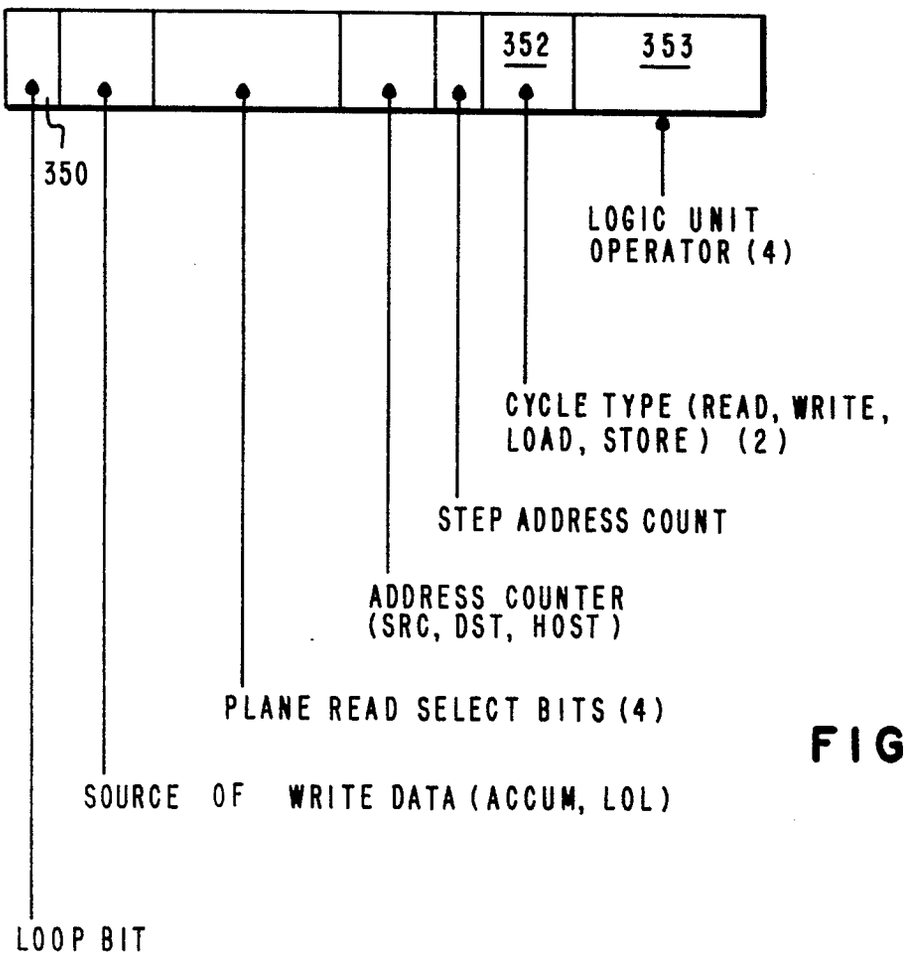

352
353
350
LOGIC UNIT
OPERATOR (4)
CYCLE TYPE (READ, WRITE,
LOAD, STORE) (2)
STEP ADDRESS COUNT
ADDRESS COUNTER
(SRC, DST, HOST)
PLANE READ SELECT BITS (4)
SOURCE OF WRITE DATA (ACCUM, LOL)
LOOP BIT

FIG. 18

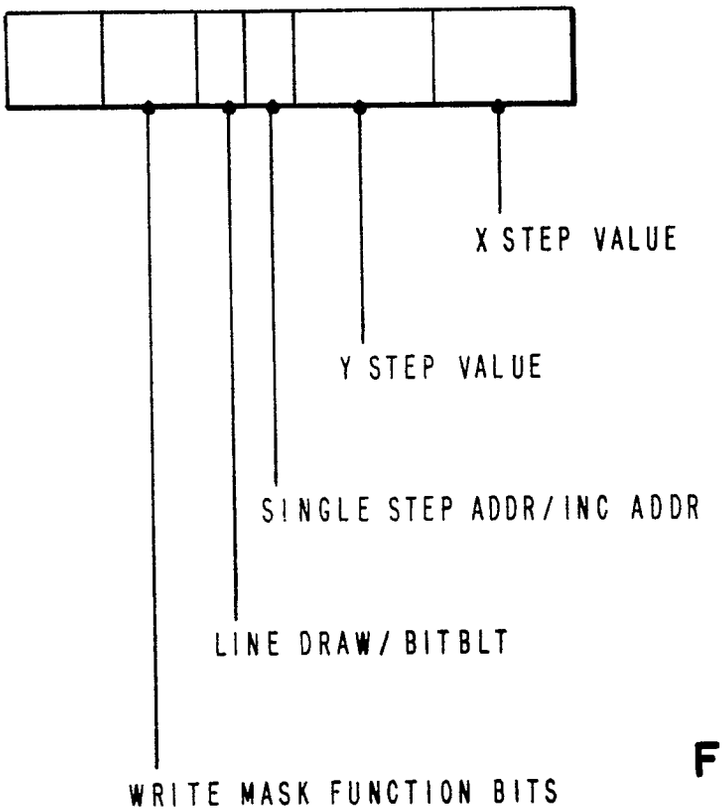

X STEP VALUE
Y STEP VALUE
SINGLE STEP ADDR/INC ADDR
LINE DRAW/BITBLT
WRITE MASK FUNCTION BITS

FIG. 19

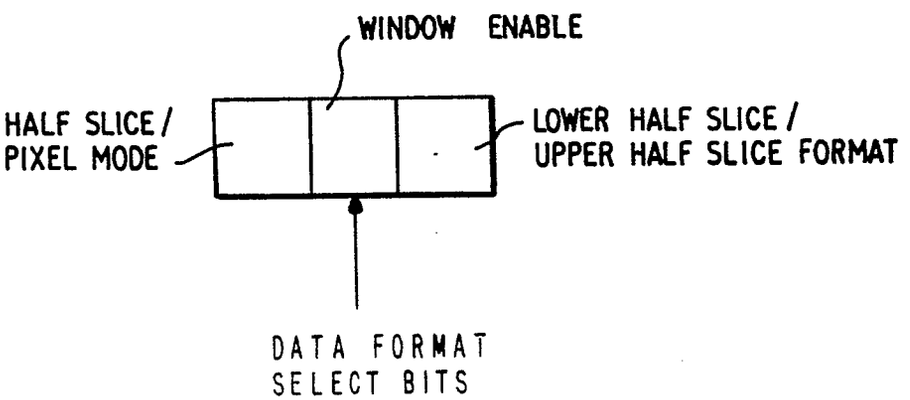

WINDOW ENABLE
HALF SLICE/
PIXEL MODE
LOWER HALF SLICE/
UPPER HALF SLICE FORMAT
DATA FORMAT
SELECT BITS

FIG. 20

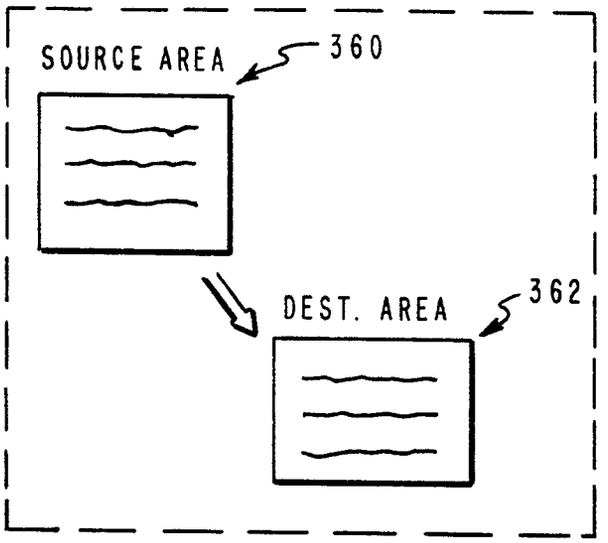

SOURCE AREA
360
DEST. AREA
362

FIG. 21

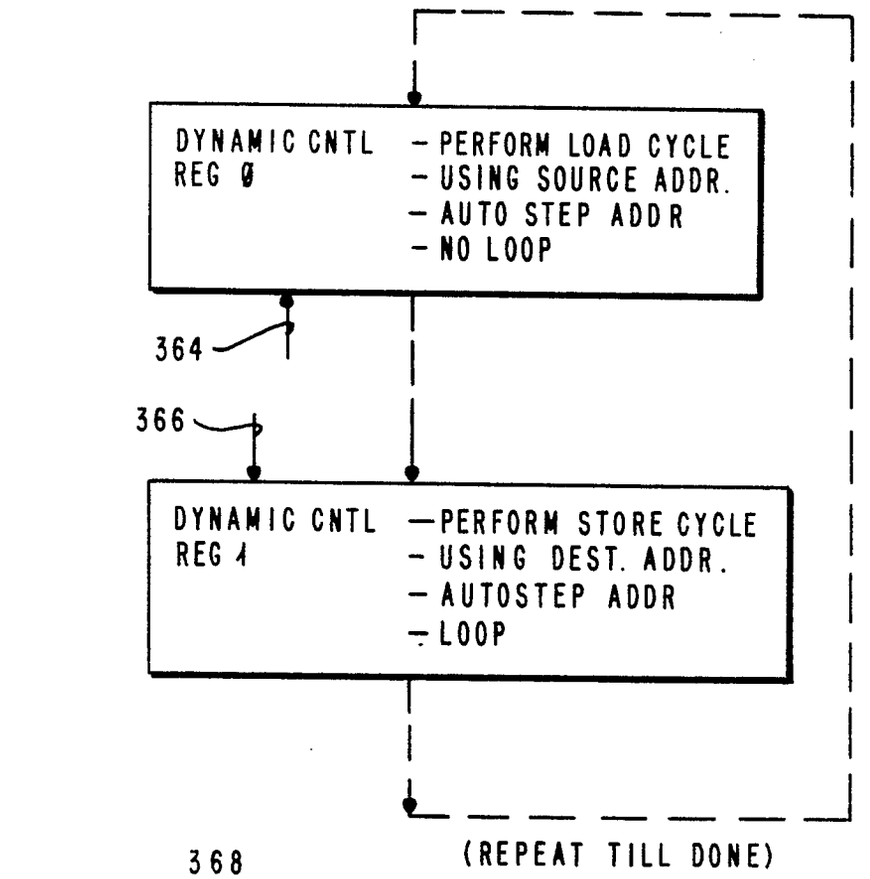

DYNAMIC CNTL REG Ø
– PERFORM LOAD CYCLE
– USING SOURCE ADDR.
– AUTO STEP ADDR
– NO LOOP
364
366
DYNAMIC CNTL REG 1
– PERFORM STORE CYCLE
– USING DEST. ADDR.
– AUTOSTEP ADDR
– LOOP
(REPEAT TILL DONE)
368
STATIC CNTL REGS
– BITBLT MODE
– X STEP
– Y STEP
– SINGLE STEP ADDR.

FIG. 22

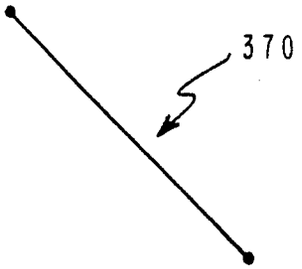
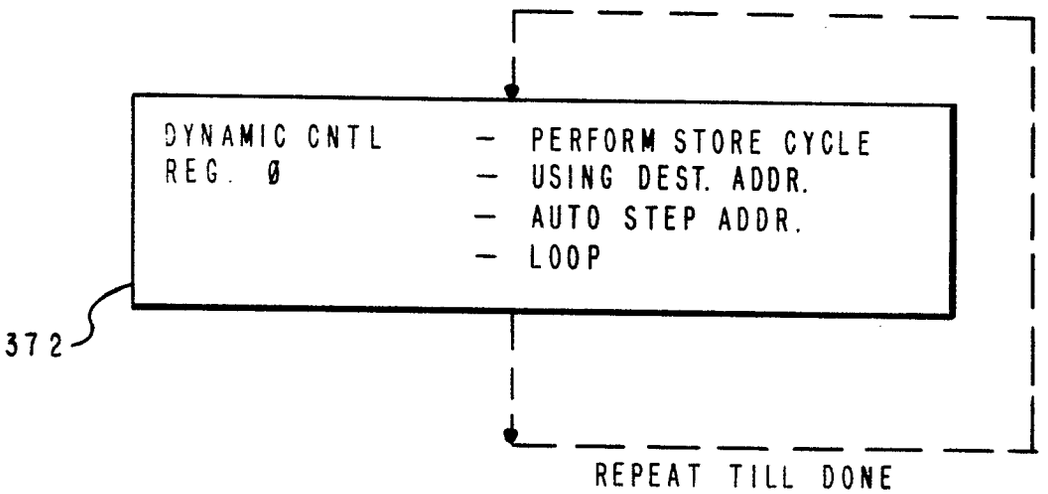

370

FIG. 23


DYNAMIC CNTL
REG. Ø
– PERFORM STORE CYCLE
– USING DEST. ADDR.
– AUTO STEP ADDR.
– LOOP
372
REPEAT TILL DONE

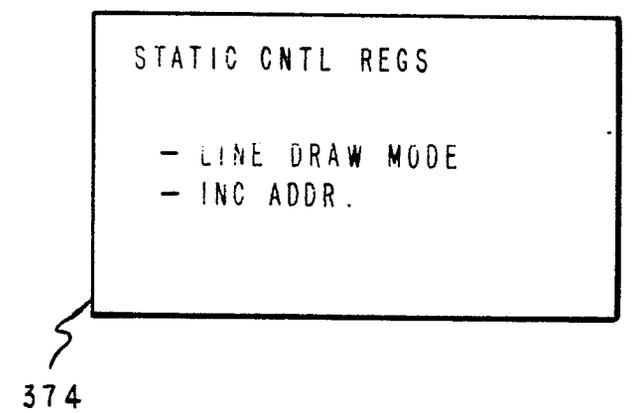

STATIC CNTL REGS
– LINE DRAW MODE
– INC ADDR.
374

FIG. 24

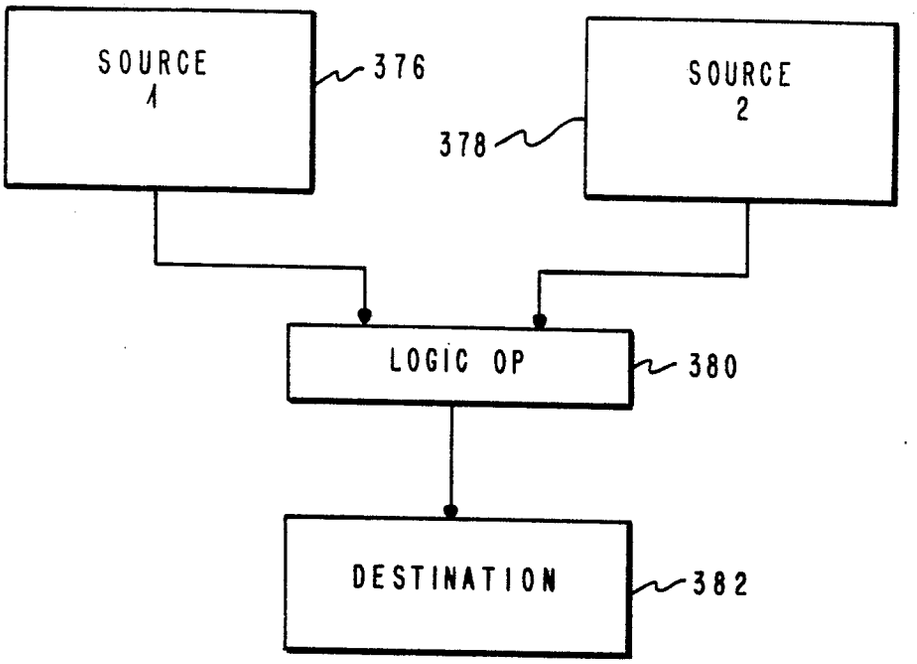


FIG. 25A

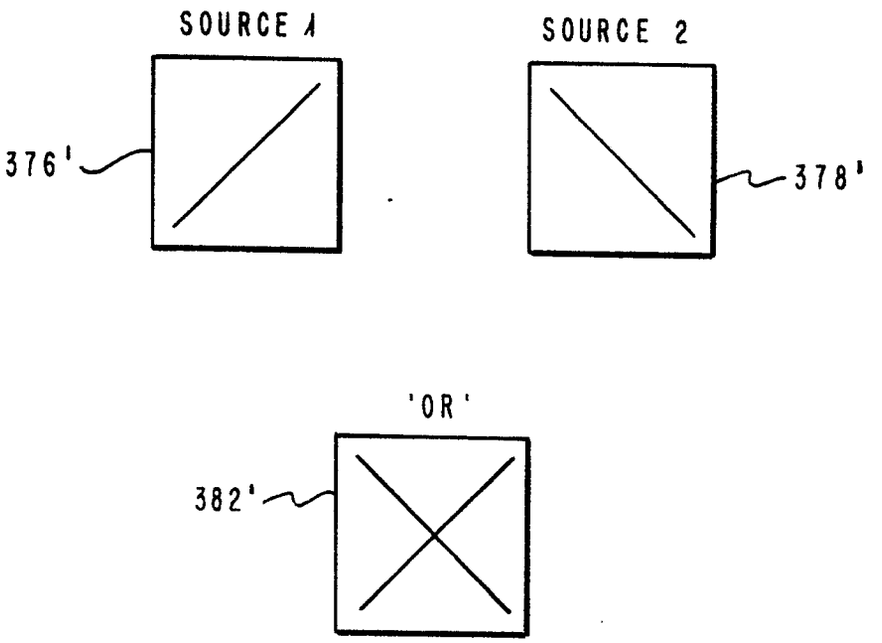


FIG. 25B

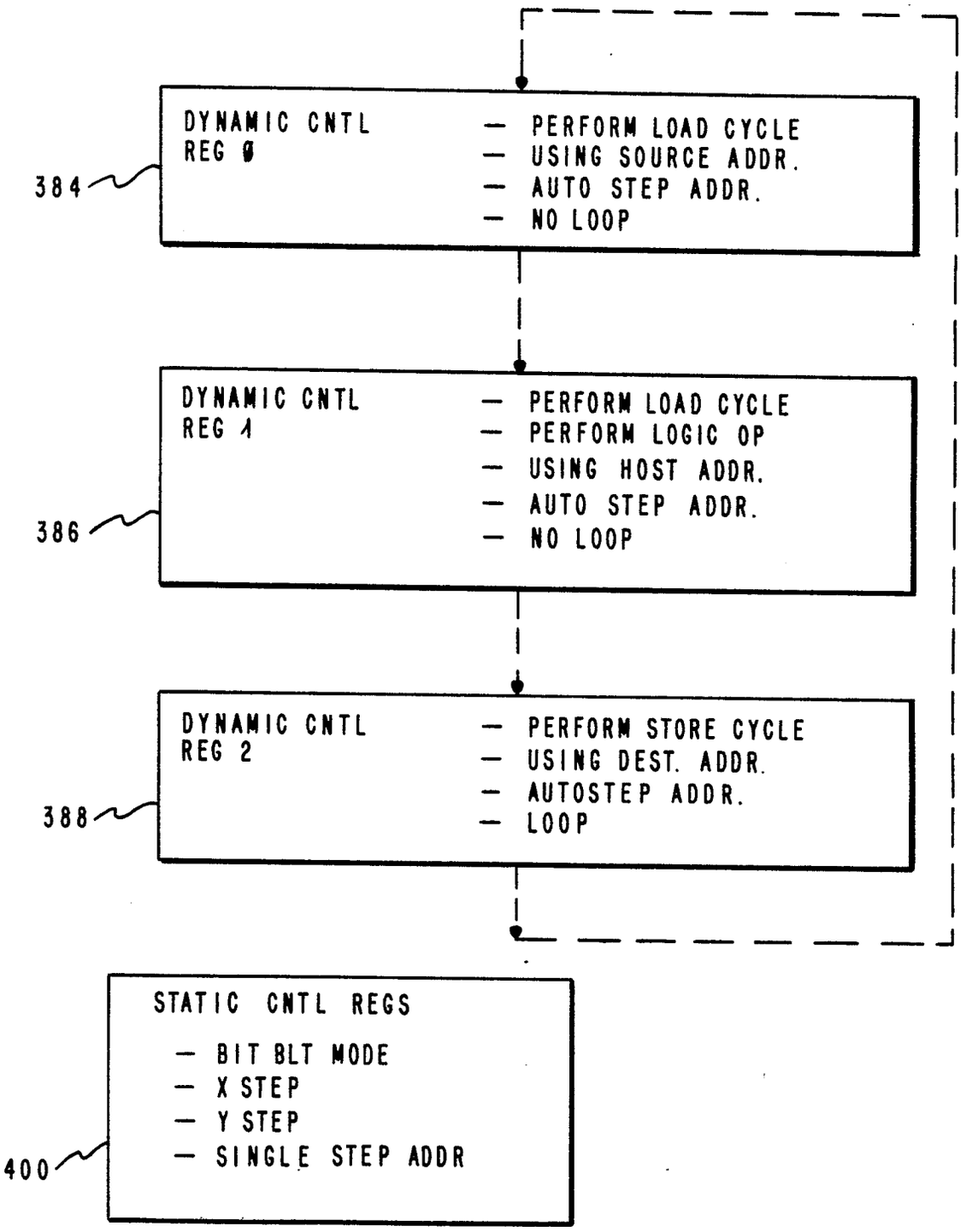

DYNAMIC CNTL REG 0
— PERFORM LOAD CYCLE
— USING SOURCE ADDR.
— AUTO STEP ADDR.
— NO LOOP
384
DYNAMIC CNTL REG 1
— PERFORM LOAD CYCLE
— PERFORM LOGIC OP
— USING HOST ADDR.
— AUTO STEP ADDR.
— NO LOOP
386
DYNAMIC CNTL REG 2
— PERFORM STORE CYCLE
— USING DEST. ADDR.
— AUTOSTEP ADDR.
— LOOP
388
STATIC CNTL REGS
— BIT BLT MODE
— X STEP
— Y STEP
— SINGLE STEP ADDR
400

FIG. 26