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

(11) Publication number: **0 147 542 B1**

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication of patent specification: **02.10.91** (51) Int. Cl.[5]: **G09G 1/00**, G09G 1/16

(21) Application number: **84111872.2**

(22) Date of filing: **04.10.84**

(54) A multiple window display system.

(30) Priority: **17.10.83 US 542572**
17.10.83 US 542376

(43) Date of publication of application:
10.07.85 Bulletin 85/28

(45) Publication of the grant of the patent:
02.10.91 Bulletin 91/40

(84) Designated Contracting States:
DE FR GB IT

(56) References cited:
EP-A- 0 055 167
EP-A- 0 099 989
US-A- 4 197 590
US-A- 4 649 377

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

(72) Inventor: **Cheselka, Harry**
6712 Hummelville Road
Saugerties New York 12477(US)
Inventor: **Lucash, Jeffrey Stuart**
97 J Altamont Drive, Hurley New York RD 7, Box 975
Kingston, NY 12401(US)
Inventor: **Vincent, William Doosevelt**
193 Winnikee Avenue
Poughkeepsie New York 12601(US)
Inventor: **Mann, Joy Lynn**
142 Prospect St. P.O. Box 795
Port Dwen New York 12466(US)

(74) Representative: **Bailey, Geoffrey Alan**
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN(GB)

EP 0 147 542 B1

Description

The present invention is generally related to a multiple window display system and more particularly to hardware and software implementations that display multiple data windows on cathode ray tube (CRT), gas panel, liquid crystal displays (LCD) and other like displays commonly used in computer and data processing systems. The invention has its primary application in multi-tasking computer environments wherein each window displays data from a different one of the tasks.

Generation of video data for a raster scanned CRT is well understood. EP-A-0 055 167 discloses a process and apparatus for displaying data in a raster scanned display apparatus, such as a cathode ray tube display, using a composite paged memory. In US-A-4 197 590 a computer graphics display system is described, which includes random access raster memory for storing data to be displayed, a raster memory control unit for writing data into the raster memory, and a video control unit for causing such data to be displayed on a CRT screen. Figure 1 shows a typical implementation. A CRT controller 10 is used to generate memory addresses for a display refresh buffer 12. A selector 14, interposed between the controller 10 and the buffer 12, is used to provide an alternate source of addressing so that the contents of the refresh buffer can be modified. Thus, the selector 14 may pass the refresh address from the controller 10 or an address on the system address bus to the display refresh buffer 12. By time division multiplexing (TDM) the refresh buffer band-width, interference between refresh and system accesses can be eliminated. For an alphanumeric character display, the display refresh buffer usually contains storage for a character code point and associated attributes. The character code point is used to address the character pel generator 16. Outputs from the character generator 16 are produced in synchronism with the scan line count output from the CRT controller 10. Attribute functions such as reverse video, blink, underscore, and the like are applied to the character generator outputs by the attribute logic 18, and the resultant pels are serialised to the video monitor.

A number of operating system (OS) programs and application programs allow a computer to carry on multiple tasks simultaneously. For example, a background data processing task might be carried on with a foreground word processing task. Related to the background data processing task might be a graphics generation task for producing pie or bar charts from the data generated in the data processing task. The data in all these tasks might be merged to produce a single document. The multi-tasking operating may be performed by a single computer such as one of the more popular micro computers now on the market, or it may be performed by a micro computer connected to a host computer. In the latter case, the host computer generally carries out the background data processing functions, while the micro computer carries out the foreground operations. By creating a composite display refresh buffer, the system shown in Figure 1 can also be used to display windows from multiple tasks. Each task is independent of the others and occupies non-overlapping space in the system memory. User-definable windows for the tasks resident in system memory can be constructed so as to display, within the limits imposed by the screen size, data from each of the tasks being processed. Figures 2A and 2B illustrate this concept. From the user perspective, windows can be displayed as either non-overlapping, as shown in Figure 2A, or layered or overlapping, as shown in Figure 2B. It will be understood by those skilled in the art, however, that an overlapping display of the type shown in Figure 2B does not imply lost data in the system memory. On the contrary, it is necessary to preserve the data for each task so that as an occulting window is moved about the display screen or even removed from the display screen, the underlying display data can be viewed by updating the refresh buffer.

While the implementation shown in Figure 1 is adequate for a class of use, it can become performance limited as the number of display windows and tasks is increased or as the display screen size is increased. As the time required to update the display refresh buffer significantly increases, system response time increases and therefore throughput decreases. Slower system response times can result from the following factors:

1. The display refresh buffer must be updated each time a task updates a location within system memory being windowed to the display screen. Control software, usually the OS, must monitor and detect the occurrence of this condition;
2. Scrolling data within one or more of the display windows requires the corresponding locations in the display refresh buffer to be updated. This may be better appreciated with reference to Figure 3 which shows the case of non-overlapping windows as in Figure 2A. Scrolling is accomplished by moving the viewable window within the system memory. A corresponding technique is used when scrolling data in overlapping windows as in Figure 2B; and
3. Whenever window sizes or positions are changed, the display refresh buffer must be updated with the appropriate locations for the system memory.

It is therefore an object of the present invention to provide a multiple data window display on a computer display that does not adversely effect the system response times as the number of data windows is increased or, in other words, to provide a multiple data window display that is especially effective for use in multi-tasking environments.

The foregoing can be attained by providing a multiple window display system including a repeatedly scanned display device, a screen buffer having display data element locations mapped directly onto the display areas of the display device and accessing means traversing the display data element locations in synchronism with the traverse of the display areas of the display device and a facility for compiling, from, potentially, a plurality of windows generated independently by individual respective users, an aggregate of data elements to be displayed, characterised in that the compiling facility is controlled by a picture matrix having compile control locations mapped directly onto the display areas of the display device and is directly responsive to the contents of the control locations to automatically filter the available data elements from the various windows, display area by display area.

The term "user" is adopted to span task, processor or operator since to the display there is no apparent difference between these.

The above can be achieved by both hardware and software arrangements. With respect to hardware implementation, plural screen buffers are simultaneously read out in a cyclic manner, and task selection means couples the output of a single one of the buffers to video output at any given time. For any given point on the screen, the data displayed originates from a selected buffer appropriate to the over-all composition producing a screen picture compiled from more than one of the screen buffers. The task selection means may be a separate task selection buffer and decoder, in which case the task selection buffer is synchronously addressed with the screen buffers and the decoder enables the read out of a single one of the screen buffers for any point on the display screen. Alternatively, one of the screen buffers may be designated to perform the operation of the task selection buffer. The display data in the designated screen buffer is non-transparent in the sense that it cannot, at a location corresponding to a given screen location, also be used for display data for that screen location, since that buffer location is loaded with unique selection code used to indicate one of the other buffers from which the data for that location is to be taken. The absence of one of these selection codes at the accessed non-transparent buffer location allows the data at that location to be displayed, as a default condition, at the corresponding screen location. In this way, it will be apparent how the display is compiled from data, in part, from the non-transparent buffer and, in part, from the other screen buffers.

Software implementation makes extensive use of system memory. The system memory provides presentation spaces for receiving application data for plural windows of the displayable area. Each window defines the whole or a subset of a corresponding presentation space. A window priority matrix mapped to the display screen filters the data from the windows of the presentation spaces to the screen buffer to designate which of the data will be shown in corresponding positions of the display screen. In a hybrid version, display data filtering can be performed both on loading a screen buffer and also on selective read out of the screen buffers where more than one such is provided

The present invention will be described further by way of example with reference to various embodiments of the invention as described hereinafter and illustrated in some of the accompanying drawing, others of which illustrate the prior art arrangements. In the drawings:-

Figure 1 is a block diagram of a prior art raster scanned CRT display generator;

Figure 2, in sections A and B, illustrates the relationship of system memory to multiple window displays for non-overlapping and overlapping windows, respectively, as produced by the prior art raster scanned CRT display generator of Figure 1;

Figure 3 illustrates the technique for producing scrolling of data in a non-overlapping window display;

Figure 4 is a block diagram of one hardware embodiment of a raster scanned CRT display generator according to the present invention;

Figure 5 illustrates the buffer maps and resultant display of a simple case of a two task display with the screen divided vertically;

Figure 6 is a block diagram of an alternative hardware embodiment of the raster scanned CRT display generator according to the invention;

Figure 7 is a functional block diagram of one form of software driver for the raster scanned CRT display generator according to this invention;

Figure 8 is a sketch of a flow chart illustrating the process of updating the windows of the presentation spaces indicated in Figure 7; and

Figure 9 is a sketch of a flow chart illustrating the process of building the screen matrix shown in Figure 7.

The arrangements described, whether prior art or according to the present invention, are for use with a CRT display. However, CRT displays are but one of many types of display, including gas panels and liquid crystal displays, to which the present invention may be applied. Therefore, those skilled in the art will understand that the mention of CRT displays is by way of example only. It follows therefore that the term refresh buffer, while having a particular meaning as applied to CRT displays, is fully equivalent to either a hardware or software screen buffer for storing data to be displayed.

The problems of slow system response time for multiple display windows in a multi-tasking environment are overcome by utilising the implementation shown in Figure 4 wherein the same reference numerals designate the same or similar circuits as in Figure 1. Each task is given a dedicated refresh buffer which can be directly addressed by the task. However, those skilled in the art will understand that this does not logically preclude including these addresses within a system memory map. Thus, there are provided screen refresh buffers 12_1 to 12_n, one for each task and directly loadable thereby. Each refresh buffer has a corresponding selector 14_1 to 14_n but the refresh address from the CRT controller 10 is not supplied directly to these selectors as in the prior art arrangement illustrated in Fig. 1. Instead, the current refresh address from the CRT controller 10 is supplied to one of the operand inputs of adders 20_1 to 20_n. The other operand input of each of these adders is supplied by corresponding offset registers 22_1 to 22_n. An effective refresh address for any one of the refresh buffers is generated by adding the current address provided by the CRT controller 10 with a value previously stored in the associated offset address register 22_1 to 22_n. Because a common refresh address is used in the example shown in Figure 4, the width of the formatted data must be the same for all the refresh buffers. Those skilled in the art will recognise that by separately addressing each of the refresh buffers and providing additional hardware to maintain synchronism in the read out of the buffers, it is possible to have different widths of formatted data in each of the refresh buffers. This added flexibility is achieved at the expense of greater complexity, and for purposes of providing a better understanding of the invention, only the simpler case is described.

For display refresh purposes, all the refresh buffers are accessed in parallel. A task selection memory 24, having a location for each screen display area so that the contents of the task selection memory can be referred to as a screen matrix, is also accessed in parallel, via its selector 26, using the CRT controller produced address, to enable the output of a single selected refresh buffer. This is accomplished by means of decoder 28 which responds to the contents, which are essentially codes, read out of the locations as they are scanned in synchronism with the screen buffers, and, of course, the display itself, to generate enable outputs 1 to n. These enable outputs are provided to the corresponding refresh buffers 12_1 to 12_n so that at any given time only one of the refresh buffers is enabled to supply an output to the character generator 16 and attribute logic 18. This means that the effect of the task selection memory and its screen matrix contents is to filter the display requirements of the various tasks by filtering the outputs of the corresponding task related refresh buffers.The operation may be better appreciated with reference to Figure 5 which shows the maps of the refresh buffers and task selection memory for the simple case of the display of two tasks with the screen divided vertically on a 16 row CRT with 16 characters per row. An 8-bit adder is assumed for this example. Refresh buffer 1 has numeric character data, while refresh buffer 2 has alpha character data. The offset register for refresh buffer 1 is loaded with the hexadecimal address F8'x', and the offset register for refresh buffer 2 is loaded with the hexadecimal address 10'x'. The task selection memory is mapped to display the data from task 2 in the left half of the screen and the data from task 1 in the right half of the screen. This produces the resultant CRT display illustrated.

The main features of this scheme may be summarise as follows:

1. Each task is totally independent of the others.
2. Refresh buffer updates are independently controlled solely by the corresponding tasks, thereby eliminating the need for separate refresh buffer rewriting each time any task wishes to create a display change.
3. Scrolling, on a task basis, is simply accomplished by updating the value in the corresponding address offset register.
4. Multiple window display with multi-layering is achieved through the use of the selection memory without affecting refresh buffer contents.
5. The system memory bus utilisation is reduced.

A simplified variation of the system shown in Figure 4 can be implemented as is shown in Figure 6. The task selection memory 24 is eliminated by designating one of the refresh buffers to be non-transparent and effectively take the place of the task selection memory. In the case shown in Fig. 6, refresh buffer 12_1 is so designated. The decoder 28 is retained and a gate 30 is added. Unique filter codes, loaded into the non-transparent refresh buffer, can then be used as the selection mechanism for the remaining transparent refresh buffers 12_2 to 12_n. The absence of of one of these selection buffer code points at the currently

accessed location in buffer 12_1, as detected by the decoder 28, causes the gate 30 to pass the data stored in that location, if any, to the character generator 16. This modification trades off a reduction in hardware against the performance loss caused by having one of the buffers non-transparent.

In the implementation illustrated in Figure 7, only two discrete hardware buffers 12_1 and 12_2 are used, though extensive use of defined areas of homogeneous system memory is made and the filtering function, still determined by a screen matrix (referenced 40 and maintained in memory) is split between selection of what is loaded into one of the buffers, relatively speaking a "one-time-function" and which of the two buffers is to provide the current output to the screen, as in the previous embodiments. The effect is the same. Though more work is done in manipulating memory, this is offset by the reduction in the frequency at which it is performed.

In the specific case illustrated, a micro computer connected to a host computer is assumed with buffer 12_2 being the micro computer buffer, but it will be understood by those skilled in the art that the pre-buffer filtering under the control of the screen matrix can be applied also to a single computer with a single buffer, provided there is sufficient system memory available. As shown, this implementation employs screen control blocks 32, window control blocks 34, presentation space control blocks 36, presentation spaces 38, and a screen matrix 40. There may be, for example, ten screen control blocks and ten sets of window control blocks, one each for each screen layout. A given screen control block 32 points to a corresponding set of window control blocks 34. Each presentation space 38 has at least one window per screen layout. The presentation spaces, but not the windows, are common to all screens. The window control block 34, corresponding to a given presentation space 38 in that screen layout, defines the origin (upper left hand corner) of the window in the presentation space, the width and height of that window in the presentation space and the origin of the window on the display screen. The screen matrix 40 is a map of the data to be displayed and, in one embodiment, maps, on a one-to-one basis by character, that which is to be displayed on the CRT screen, but the mapping could be on a pel or any other basis. All display output from the several tasks is directed to memory and, specifically, to the presentation spaces 38 rather than to the hardware refresh buffer. In the arrangement illustrated in Fig. 7, a micro computer, such as the IBM (R.T.M.) Personal Computer (PC), is assumed to be attached to a host computer such as an IBM 3270 computer via a controller such as an IBM 3274 controller. For this case, the PC hardware buffer 12_2 acts as the PC presentation space. Each presentation space is assigned an identification tag and has an associated window defined by the operator or an application program as to size and screen location. When the human operator or application program adjusts the windows relative to one another, the system builds an image in the screen matrix 40 consisting of the identifying tag aligned in the appropriate locations. The matrix 40 may be created in a reverse order from that appearing on the CRT screen allowing overlapping windows to be built up by overwriting. Alternatively, by using a compare function, the matrix 40 can be created by beginning with the uppermost window and so on, down through the overlay. The choice of the method of creating the matrix 40 is based on desired system performance. The system directs display output to the refresh buffer by filtering all screen updates through the screen matrix 40, allowing a performance increment in an overlapped window system by only allowing those characters that actually need to be changed or displayed on the screen to reach the refresh buffer.Those characters that are not currently required,do not reach the refresh buffer, will not cause an unnecessary redraw. The absence of these unnecessary redraws removes the requirement for continual updates of all windows whenever the contents of one is altered.

In order to write a character, the IBM 3274 controller, a supervisor application or the PC writes character code into presentation space 38 at locations designated by that presentation space's cursor value control block. No other updates are required. the new character will be displayed or not according to whether it falls within the window designated by the corresponding window control block 34 and the portion of that window designated for display by the screen matrix 40. To use the PC buffer 12_2, a window control block is established for the PC the same as any other window control block 34 including width, height, presentation space origin, and screen origin. The screen matrix 40 is updated, and data from the window in the PC buffer defined by the window control block 34 will, to the extent allowed by the screen matrix 40, appear on the CRT screen. Data within a window may be scrolled by decrementing or incrementing the X or Y value of the window origin. No other control updates are needed. Only the corresponding window in the screen buffer is rewritten or, if a PC window, the offset register is changed. A window can be relocated on the screen by changing the origin coordinates in the window control block 34 for that window. The screen matrix 40 is updated, and the entire non-PC screen buffer is rewritten with data for non-PC tasks and codes (hexadecimal FF) for the PC. To enlarge the visible portion of a presentation space without scrolling, the window control block 34 for that presentation space 38 is first updated by altering the width and/or height. This adds to the right or bottom of window only unless there is also a change in the origin of the window. Ordinarily, there is no change in the origin unless there is an overflow off the presentation space or screen,

in which case, the corresponding origin is altered. Next, the screen matrix 40 is updated by over-writing window designator codes of the matrix, starting with the lowest priority window control block. Then, all windows to non-PC refresh buffer 12_1 are rewritten with data from the presentation space for the non-PC windows and the hexadecimal code FF for the PC window.

Figure 8 illustrates the general shape of the process for window updating. In block 42, the presentation space (PS) row is set to the first PS row needing update; the screen row is set to the row on the display screen of the PS row; the PS column is set to the first PS column needing update; the screen column is set to the column on the screen of the PS column; the number of rows is set to the number of PS rows to be updated; and the number of columns is set to the number of PS columns to be updated. Then, the procedure which follows is done for the number of rows to be updated. For the number of columns to be updated, the matrix 40 is checked to determine if the screen row and column is within the window to be updated. This is indicated by the decision block 44. A test is made for the PC, since hardware buffer 12_2 is the presentation space for the PC, and the hexadecimal code FF is used to denote the PC window. If the decision of block 44 is yes, then the screen row and column are set to the PS row and column as indicated by block 46, and the screen column and the PS column are incremented as indicated by block 48; otherwise, the screen column and PS column are incremented without setting the screen row and column to the PS row and column. When this process is complete for the number of columns to be updated, the PS column is updated to the first PS column needing update as indicated by block 50. Then, the PS row is incremented, and the screen row is incremented as indicated by block 52.

Figure 9 illustrates the general shape of the process for building the screen matrix 40. First, the window is set to the bottom window as indicated in block 54. Then for all windows not known to be hidden, the following procedure is performed. In block 56, the column is set to the first window column on the screen, and the row is set to the first window row on the screen. For the number of window rows, the procedure indicated within block 58 is followed, and this procedure includes the procedure indicated within block 60 for the number of window columns. In block 60, the matrix row and column is set to the window identification as indicated in block 62. Next, the column is incremented as indicated by block 64. Exiting block 60 but still within block 58, the column is set to the first window column on the screen as indicated by block 66. Then, the row is incremented as indicated by block 68. Now exiting block 58, the window is incremented to the next window as indicated by block 70.

The function which draws the multiple window display is driven by any one of the following:

1. A PC cursor register update;
2. A PC text/graphics node register update;
3. A change in the window control block, screen control block, or presentation space control block; or
4. A change in the presentation space data.

Application programs may cause the draw function to occur for cases 3 and 4 above by using the following functional calls:

DRAW SCREEN
DRAW NEWTOP
DRAW PS
DRAW CURSOR
DRAW CHARACTER
DRAW PS IMMEDIATE
DRAW BORDER
DRAW OIA

These functional calls are set forth in detail below:

```
DRAW SCREEN

-> INDMRDF (FAR CALL FROM MACRO) (INPUT IS SCB PTR)
      * SETS SCBDRS ON - DRAW SCREEN

      <-> INDMMXF - REBUILD SCREEN MATRIX
```

```
        • READY DRAW TASK

        • EXIT

FROM DISPATCHER
-> INDMDTF (FAR CALL)

        • ISSUE ?DRAW CURSOR TO ADJUST WINDOW FOR CURSOR
          <-> INDMCRF

        <-> INDMBLK TO CLEAR THE SCREEN

        • DRAW TOP APPLICATION WINDOW (SCBTOPW)
          -> INDMWIC
               CALCULATES VARIABLES FOR INDMFTC AND PUTS IN
                                                      COMMON
                  DRW_SRC - OFFSET OF START OF CHARACTERS
                                                      IN PS
                  DRW_DST - OFFSET OF START OF CHARACTERS
                                                   ON SCREEN
                  NDRWROWS/NDRWCOLS - # ROWS/COLS TO DRAW
                                              (COMPLETE ROWS)
               <-> INDMFTC
                    <-> INDMSMA - SEARCH FOR ATTRIBUTE
          <-

        • DRAW BORDER
          <-> INDMBDF

        • DRAW OIA
          IF COMMAND MODE -> INDMWIC FOR SYSTEM OIA
          IF APPL MODE    -> INDMINF FOR APPL OIA

        • DRAW REST OF SYSTEM WINDOWS (SYSWCB->WCBNEXT)
```

```
                -> INDMWIC
                    <-> INDMFTC
                -> INDMBRF

          * EXIT

DRAW PS
 -> INDMRDF (FAR CALL FROM MACRO) (INPUT IS PSCB PTR)

        * MAKE SURE WINDOW IS ON ACTIVE SCREEN

        * INDICATE DRAW WINDOW (WCBDRAW ON,SCBDRW ON)

        * READY DRAW TASK

        * EXIT

FROM DISPATCHER
-> INDMDTF (FAR CALL)

        * LOOKS THROUGH WCB'S ON ACTIVE SCREEN FOR A WINDOW
          THAT NEEDS TO BE DRAWN.  START WITH TOP APPL WINDOW,
          THEN SYSTEM CHAIN, THEN REMAINDER OF APPL CHAIN

        * DRAW FIRST WINDOW THAT NEEDS TO BE DRAWN
            -> INDMWIC
                <-> INDMFTC
            <-

        * DRAW OIA FOR THAT WINDOW
            IF COMMAND MODE -> INDMWIC FOR SYSTEM OIA
            IF APPL MODE    -> INDMINF FOR APPL OIA

        * READY DRAW TASK
```

```
        * EXIT

DRAW PS  (IMMED)

 -> INDMRDF (FAR CALL FROM MACRO) (INPUT IS PSCB PTR)

        * MAKE SURE WINDOW IS ON ACTIVE SCREEN

        * INDICATE DRAW WINDOW (WCBDRAW ON,SCBDRW ON)

        * READY DRAW TASK

 -> INDMCHF (FAR CALL FROM MACRO) (INPUT IS
PSCB,START@,LENGTH)

        * IF ONLY PSCB PASSED, ENTIRE PS IS DRAWN
          OTHERWISE, LENGTH (ROUNDED TO WHOLE ROW) IS DRAWN

          -> INDMWIC
               <-> INDMFTC
          <-

        * EXIT

DRAW CURSOR

 -> INDMCRF (FAR CALL FROM MACRO) (INPUT IS PSCB PTR,CURSOR
                                   LOCATION, CURSOR TYPE)

        * CHECKS THAT PS IS ACTIVE. IF NOT, JUST PSCB IS
                                                  UPDATED.

        * IF INPUT IN OFFSET FORM THEN
            <-> INDMRCF - CONVERTS OFFSET TO ROW/COLUMN
```

```
        <-> INDMACC - SEE IF CURSOR VISIBLE. IF NOT, SCROLLS
                      WINDOW CHANGES WCB LOGICAL ROW/COL, RC=4)

        IF WINDOW NEEDS TO BE REDRAWN THEN
            -> INDMWIC
                 <-> INDMFTC
            <-

        -> INDMCRS - WRITES TO ADAPTER REGISTER FOR CURSOR
                     LOCATION AND TYPE (BLINK, UNDERSCORE)

      * EXIT

DRAW CHARACTER

 -> INDMCHF (FAR CALL FROM MACRO) (INPUT IS PSCB PTR,LOCATION
                                   OF CHAR)

      * CHECKS THAT WINDOW IS VISIBLE AND CHARACTER WITHIN
                                                      WINDOW

      * IF INPUT IN OFFSET FORM THEN
          <-> INDMRCF - CONVERTS OFFSET TO ROW/COLUMN

      * IF INPUT IN ROW/COL FORM THEN
          <-> INDMOFF - CONVERTS ROW/COL TO OFFSET

      * CALCULATES VARIABLES IN COMMON FOR INDMFTC
          DRW_SRC - OFFSET OF START OF CHARACTER IN PS
          DRW_DST - OFFSET OF START OF CHARACTER ON SCREEN
          NDRWROWS/NDRWCOLS - # OF ROWS/COLS TO DRAW
                                              (COMPLETE ROWS)

      <-> INDMFTC - PUTS ON SCREEN
```

```
        * EXIT

DRAW DIA

 -> INDMIRF (FAR CALL FROM MACRO) (INPUT IS PSCB

        * IF TCA THEN
            <-> INDMTIC - BUILD OIA FROM TCA CONTROL BLOCKS

        * IF NOTEPAD THEN
            <-> INDMNPO - TO BUILD OIA FROM NOTEPAD CONTROL
                                                        BLOCK

        * IF PC THEN
            <-> INDMPIC - TO BUILD OIA

        * IF COMMAND MODE OIA WAS BUILD BY SCREEN MANAGER

        * IF DCA, OIA IS IN DCA BUFFER

        * ADD ANY SYSTEM INDICATERS TO OIA

        -> INDMWIC <-> INDMFTC

        * EXIT

DRAW NEWTOP

 -> INDMNTF (FAR CALL FROMMACRO) (INPUT IS WCB)

        * IF WINDOW ON ACTIVE SCREEN OF IF SYSTEM WINDOW
            <-> INDMMXF - TO BUILD NEW MATRIX

        * ISSUE DRAW CURSOR TO ADJUST WINDOW FOR CURSOR IF
```

```
                                                          NEEDED
            <-> INDMCRF <-> INDMACC
                        <-> INDMCUS

        <-> INDMWIC <-> INDMFTC TO DRAW THIS WINDOW

        . IF TOP OF SYSTEM CHAIN - EXIT
            ELSE IF TOP OF APPL CHAIN - GET BOTTOM OF SYSTEM
                                                          CHAIN
                ELSE GET NEXT HIGHER (WCBPREW)

        -> INDMWIC <-> INDMFTC

        . EXIT

PC CURSOR INTERRUPT

 -> INDIINT (FIRST LEVEL INTERRUPT HANDLER)

      -> INDICUR - (SECOND LEVEL INTERRUPT HANDLER)
           . IF PC CURSOR LOCATION HAS CHANGED OR
              PC CURSOR TYPE HAS CHANGED
              THEN
                SAVE LOCATION AND SIZE IN COMMON
                ENQUEUE TO DRAW

    <-
<-

FROM DISPATCHER
-> INDMDTF (FAR CALL)

      . ISSUE ?DRAW CURSOR FOR PC CURSOR WITH LOCATION AND
          TYPE FROM COMMON
```

```
        <-

DRAW BORDER

 -> INDMRDF (FAR CALL FROM MACRO)

      * IF WINDOW AND BORDER NOT HIDDEN
          -> INDMBDF
               * IF WINDOW IS ACTIVE, USE ACTIVE BORDER
                 (DOUBLE BAR)
               * IF WINDOW IS NOT ACTIVE, USE INACTIVE BDR
                 (SINGLE BAR)
          <-

      * EXIT

PC MODE INTERRUPT

 -> INDIINT (FIRST LEVEL INTERRUPT HANDLER)

     -> INDICUR - (SECOND LEVEL INTERRUPT HANDLER)
          * IF PC MODE HAS CHANGED
              THEN
                SAVE MODE IN COMMON
                ENQUEUE TO DRAW
     <-
<-

FROM DISPATCHER
-> INDMDTF (FAR CALL)

     * IF GRAPHICS MODE
        THEN ISSUE ?DRAW CURSOR TO INHIBIT PC CURSOR
        ELSE ISSUE ?DRAW CURSOR TO WRITE PC CURSOR
```

Claims

1. A multiple window display system including a repeatedly scanned display device, a screen buffer (12) having display data element locations mapped directly onto the display areas of the display device and

accessing means traversing the display data element locations in synchronism with the traverse of the display areas of the display device and a facility for compiling, from, potentially, a plurality of windows generated independently by individual respective users, an aggregate of data elements to be displayed, characterised in that the compiling facility is controlled by a screen matrix (24,40) having compile control locations mapped directly onto the display areas of the display device and is directly responsive to the contents of the control locations to automatically filter the available data elements from the various windows, display area by display area.

2. A display system as claimed in claim 1 wherein there are plural screen buffers (12_1-12_n), traversed in synchronism with each other, with the display areas and with the screen matrix, available for assignment to individual users, the filtering in response to the picture matrix contents being performed on the outputs from the various buffers to the display device.

3. A display system as claimed in claim 2 including address generating means (14_1-14_n) for simultaneously supplying addresses to each of said plural screen buffers, the address generating means including plural offset means (22_1-22_n), one for each buffer, for modifying the addresses supplied to that buffer.

4. A display system as claimed in claim 3 wherein the task selection memory means is arranged to receive addresses in synchronism with addresses supplied to the plural screen buffers and further includes decoding means (28) for decoding codes generated by the task selection memory means in response to such addresses, the decoding means producing an enable signal for a selected one of the plural screen buffers at any given time.

5. A display system as claimed in claim 2 or claim 3 wherein the picture matrix is maintained in one of the buffers (12_1) and its compile control locations may contain either control codes or display data elements, but not both in any one location, the compiling facility being arranged to respond thereto by accepting the picture matrix data element, if stored in the currently accessed control location, or the data element from the indicated other buffer, if a control code is stored in the currently accessed control location in the picture matrix.

6. A display system as claimed in claim 2 or claim 3 including a character generator and wherein the task selection memory means includes one of the plural screen buffers (12_1) which is designated as a non-transparent screen buffer, such non-transparent screen buffer having stored therein unique code points which are used to select among the remaining screen buffers, the data in the plural screen buffers being read out in synchronism with refresh addresses supplied thereto,the task selection memory means further comprising decoding means connected to the output of the non-transparent screen buffer for decoding the unique code points and producing an enable signal for a selected one of the plural screen buffers at any given time in response to the decoding of the currently accessed code point and gating means connected to the output of the non-transparent screen buffer and responsive to the decoding means to pass the current output from the non-transparent screen buffer as a character code to the character generator when such output is decoded by the decoding means as not being a code point.

7. A display system as claimed in claim 1 including means (32,34) for establishing user associated presentation spaces (38) in homogeneous storage for receiving display data from the associated user and means for establishing windows in the respective presentation spaces, the compiling facility including control means, responsive to the picture matrix, for controlling the loading of the screen buffer by presentation space selection thus filtering the data from the various windows into the screen buffer and hence compiling the picture actually displayed on the display device.

8. A display system as claimed in claim 7 including at least one additional dedicated screen buffer (12_2) of immediate access to one user (PC), the picture matrix mapping, in addition to the data to be loaded into the one screen buffer, which screen buffer is to supply the current output to the display device, the control means enabling only one of the screen buffers for read out at any given time.

9. A display system as claimed in claim 8 in which the window control block means indicate the coordinates and dimensions of each of the windows and means, responsive to the window control

blocks, are provided for creating and adjusting, in response to changes in the window control block configuration, the screen matrix and thereby establishing window priority at screen buffer locations.

10. A display system as claimed in claim 8 or claim 9 wherein each additional dedicated screen buffer stores display data for the associated user and constitutes the presentation space for that user.

Revendications

1. Système d'affichage à fenêtres multiples comprenant un dispositif d'affichage balayé de façon répétée, une mémoire intermédiaire d'écran (12) ayant des emplacements d'éléments de données d'affichage configurés directement sur les zones d'affichage du dispositif d'affichage, et des moyens d'accès parcourant les emplacements d'éléments de données d'affichage en synchronisme avec le parcours des zones d'affichage du dispositif d'affichage, et des facilités pour compiler à partir, éventuellement, d'une pluralité de fenêtres engendrées indépendamment par différents utilisateurs respectifs, un agrégat d'éléments de données à afficher, caractérisé en ce que les facilités de compilation sont contrôlées par une matrice d'écran (24, 40) ayant des emplacements de contrôle de compilation configurés directement sur les zones d'affichage du dispositif d'affichage, et est directement sensible au contenu des emplacements de contrôle afin de filtrer automatiquement les éléments de données disponibles depuis les diverses fenêtres, zone d'affichage par zone d'affichage.

2. Système d'affichage selon la revendication 1, dans lequel il y a plusieurs mémoires intermédiaires d'écran (12_1 -12_n) parcourues en synchronisme les unes avec les autres, avec les zones d'affichage et avec la matrice d'écran, pouvant être attribuées à différents utilisateurs, le filtrage en réponse au contenu de la matrice d'image se faisant sur les sorties provenant des diverses mémoires intermédiaires en direction du dispositif d'affichage.

3. Système d'affichage selon la revendication 2, comprenant des moyens générateurs d'adresses (14_1-14_n) pour fournir simultanément des adresses à chacune de la pluralité de mémoires intermédiaires d'écran, les moyens générateurs d'adresses comprenant des moyens de décalage (22_1-22_n), un par mémoire, pour modifier les adresses fournies à cette mémoire intermédiaire.

4. Système d'affichage selon la revendication 3, dans lequel des moyens d'emmagasinage de sélection de tâches sont agencés pour recevoir des adresses en synchronisme avec les adresses fournies à la pluralité de mémoires intermédiaires d'écran, et comprend en outre des moyens décodeurs (28) pour décoder les codes engendrés par les moyen d'emmagasinage de sélection de tâches en réponse à ces adresses, les moyens décodeurs produisant un signal de conditionnement d'une mémoire intermédiaire d'écran parmi la pluralité de mémoires intermédiaires d'écran à un temps donné quelconque.

5. Système d'affichage selon la revendication 2 ou la revendication 3, dans lequel la matrice d'image est maintenue dans une des mémoires intermédiaires (12_1) et ses emplacements de contrôle de compilation peuvent contenir soit des code de contrôle soit des éléments de données d'affichage, mais pas les deux, dans un emplacement quelconque, les facilités de compilation étant agencées de manière à y répondre en acceptant l'élément de données de matrice d'image, s'il est emmagasiné dans l'emplacement de contrôle en cours d'accès, ou l'élément de données provenant de l'autre mémoire intermédiaire indiquée, si un code de contrôle est emmagasiné dans l'emplacement de contrôle en cours d'accès dans la matrice d'image.

6. Système d'affichage selon la revendication 2 ou la revendication 3 comprenant un générateur de caractères et dans lequel les moyens d'emmagasinage de sélection de tâches comportent une mémoire parmi la pluralité de mémoires intermédiaires d'écran (12_1) qui est désignée comme mémoire intermédiaire d'écran non transparente, cette mémoire intermédiaire d'écran non transparente ayant en mémoire des points de code unique qui sont utilisés pour sélectionner, parmi les mémoires intermédiaires d'écran restantes, les données dans les mémoires intermédiaires d'écran lues en synchronisme avec les adresses de rafraîchissement qui leur sont appliquées, les moyens d'emmagasinage de sélection de tâches comprenant en outre des moyens décodeurs connectés à la sortie de la mémoire intermédiaire d'écran non transparente pour décoder les points de code unique et produire un signal de conditionnement d'une mémoire intermédiaire parmi la pluralité de mémoires intermédiaires d'écran à un temps donné quelconque en réponse au décodage du point de code en cours d'accès, et des

moyens de passage connectés à la sortie de la mémoire intermédiaire d'écran non transparente et sensibles aux moyens décodeurs pour faire passer la sortie en cours provenant de la mémoire intermédiaire non transparente en tant que code de caractère appliqué au générateur de caractères lorsque cette sortie est décodée par les moyens décodeurs comme n'étant pas un point de code.

7. Système d'affichage selon la revendication 1, comprenant des moyens (32, 34) pour mettre en mémoire homogène des espaces de présentation associés à l'utilisateur (38) pour recevoir des données d'affichage en provenance de l'utilisateur associé, et des moyens pour établir des fenêtres dans les espaces de présentation respectifs, la facilité de compilation comprenant des moyens de contrôle, sensibles à la matrice d'image, pour contrôler le chargement de la mémoire intermédiaire d'écran avec la sélection d'espaces de présentation, filtrant ainsi les données provenant des diverses fenêtres dans le mémoire intermédiaire d'écran et, partant, compilant l'image véritablement affichée sur le dispositif d'affichage.

8. Dispositif d'affichage selon la revendication 7, comprenant au moins une mémoire intermédiaire d'écran supplémentaire (12_2) d'accès immédiat à un usager (PC), la configuration de la matrice d'image, en plus des données à charger dans une mémoire intermédiaire d'écran, laquelle mémoire intermédiaire d'écran doit fournir la sortie en cours au dispositif d'affichage, les moyens de contrôle conditionnant seulement une des mémoires intermédiaires d'écran pour être lue à un temps donné quelconque.

9. Système d'affichage selon la revendication 8, dans lequel des blocs de contrôle de fenêtres indiquent les coordonnées et les dimensions de chacune des fenêtres, et des moyens, sensibles aux blocs de contrôle de fenêtres, sont utilisés pour créer et régler, en réponse aux changements de la configuration des blocs de contrôle de fenêtres, la matrice d'écran et pour établir alors une priorité entre les fenêtres aux emplacements de la mémoire intermédiaire d'écran.

10. Système d'affichage selon la revendication 8 ou la revendication 9, dans lequel chacune des mémoires intermédiaires d'écran dédiée supplémentaire emmagasine les données d'affichage pour l'utilisateur associé, et constitue l'espace de présentation pour cet utilisateur.

Patentansprüche

1. Mehrfachfenster-Anzeigesystem mit einer wiederholt abgetasteten Anzeigeeinrichtung, mit einem Bildschirmpuffer (12), der unmittelbar auf die Anzeigebereiche der Anzeigeeinrichtung abgebildete Anzeigedatenelementstellen aufweist und mit Zugriffsmitteln, welche die Anzeigedatenelementstellen synchron mit dem Durchlauf der Anzeigebereiche der Anzeigeeinrichtung durchlaufen und mit einer Einrichtung zum Kompilieren eines Satzes anzuzeigender Datenelemente potentiell aus einer Mehrzahl unabhängig durch einzelne jeweilige Benutzer erzeugter Fenster, dadurch gekennzeichnet, daß die Kompilier-Einrichtung durch eine Bildschirmmatrix (24, 40) gesteuert ist, die unmittelbar auf die Anzeigebereiche der Anzeigeeinrichtung abgebildete Kompiliersteuerstellen aufweist und unmittelbar auf die Inhalte der Steuerstellen anspricht, um die verfügbaren Datenelemente aus den verschiedenen Festern, von Anzeigebereich zu Anzeigebereich automatisch zu filtern.

2. Anzeigesystem nach Anspruch 1, bei welchem mehrere Bildschirmpuffer (12_1-12_n) vorhanden sind, die miteinander, mit den Anzeigebereichen und mit der zum Zuordnen einzelnen Benützern verfügbaren Bildschirmmatrix synchron durchlaufen werden, wobei das den Bildmatrixinhalten entsprechende Filtern an den Ausgängen von den verschiedenen Puffern zu der Anzeigeeinrichtung durchgeführt wird.

3. Anzeigesystem nach Anspruch 2 mit Adreßerzeugungsmitteln (14_1-14_n) zum gleichzeitigen Liefern von Adressen an jeden der mehreren Bildschirmpuffer, wobei die Adreßerzeugungsmittel mehrere Versetzungsmittel (22_1-22_n), eines für jeden Puffer, zum Modifizieren der an jenen Puffer gelieferten Adressen aufweisen.

4. Anzeigesystem nach Anspruch 3, bei welchem das Task-Auswahl-Speichermittel eingerichtet ist, um Adressen synchron mit an die mehreren Bildschirmpuffer gelieferten Adressen zu empfangen und ferner Decodierungsmittel (28) zum Decodieren durch das Task-Auswahl-Speichermittel entsprechend solchen Adressen erzeugter Codes aufweist, wobei die Decodierungsmittel ein Aktiviersignal für einen ausgewählten der mehreren Bildschirmpuffer zu jeder gegebenen Zeit erzeugen.

5. Anzeigesystem nach Anspruch 2 oder Anspruch 3, bei welchem die Bildmatrix in einem der Puffer (12_1) gehalten ist und ihre Kompiliersteuerstellen entweder Steuercodes- oder Anzeigedatenelemente, jedoch nicht beide in irgendeiner Stelle enthalten können, wobei die Kompiliereinrichtung eingerichtet ist, um darauf durch Annehmen des Bildmatrixdatenelementes zu antworten, falls in der augenblicklich unter Zugriff befindlichen Steuerstelle gespeichert oder des Datenelementes aus dem angezeigten anderen Puffer, falls ein Steuercode in der augenblicklich unter Zugriff befindlichen Steuerstelle in der Bildmatrix gespeichert ist.

6. Anzeigesystem nach Anspruch 2 oder Anspruch 3 mit einem Schriftzeichengenerator und bei welchem das Task-Auswahl-Speichermittel einen der mehreren Bildschirmpuffer (12_1) aufweist, der wie ein nicht-transparenter Bildschirmpuffer bereitgestellt ist, wobei ein solcher nicht-transparenter Bildschirmpuffer in ihm gespeicherte eindeutige Code-Stellen aufweist, die zum Auswählen unter den verbleibenden Bildschirmpuffern verwendet werden, die Daten in den mehreren Bildschirmpuffern synchron mit daran gelieferten Auffrischadressen ausgelesen werden, und das Task-Auswahl-Speichermittel ferner Decodierungsmittel, die mit dem Ausgang des nicht-transparenten Bildschirmpuffers zum Decodieren der eindeutigen Code-Stellen verbunden sind und ein Aktiviersignal für einen ausgewählten der mehreren Bildschirmpuffer zu jeder gegebenen Zeit entsprechend dem Decodieren der augenblicklich unter Zugriff befindlichen Code-Stelle erzeugen und Torschaltungsmittel aufweist, die mit dem Ausgang des nicht-transparenten Bildschirmpuffers verbunden sind und auf die Decodierungsmittel ansprechen, um den augenblicklichen Ausgang von dem nicht-transparenten Bildschirmpuffer als einen Schriftzeichencode an den Schriftzeichengenerator zu übertragen, wenn ein solcher Ausgang durch die Decodierungsmittel als keine Code-Stelle decodiert wird.

7. Anzeigesystem nach Anspruch 1 mit Mitteln (32, 34) zum Festlegen einem Benützer zugeordneter Darstellungsbereiche (38) in einem homogenen Speicher zum Empfangen von Anzeigedaten von dem zugeordneten Benützer und mit Mitteln zum Festlegen von Fenstern in den jeweiligen Darstellungsbereichen, wobei die Kompiliereinrichtung auf die Bildmatrix ansprechende Steuermittel zum Steuern des Ladens des Bildschirmpuffers durch eine Darstellungsbereichauswahl aufweist, wodurch die Daten von den verschiedenen Fenstern in den Bildschirmpuffer gefiltert werden und das auf der Anzeigeeinrichtung tatsächlich angezeigte Bild folglich kompiliert wird.

8. Anzeigesystem nach Anspruch 7, mit zumindest einem zusätzlich dazu bestimmten Bildschirmpuffer (12_2) mit sofortigem Zugriff auf einen Benützer (PC), mit der Bildmatrixabbildung zusätzlich zu den in den einen Bildschirmpuffer zu ladenden Daten, wobei dieser Bildschirmpuffer den augenblicklichen Ausgang an die Anzeigeeinrichtung liefern soll und die Steuermittel nur einen der Bildschirmpuffer zum Auslesen zu jeder gegebenen Zeit aktivieren.

9. Anzeigesystem nach Anspruch 8, bei welchem die Fenstersteuerblock-Mittel die Koordinaten und Abmessungen jedes der Fenster anzeigen und auf Fenstersteuerblöcke ansprechende Mittel vorgesehen sind, um entsprechend Änderungen in der Fenstersteuerblockkonfiguration die Bildschirmmatrix zu erzeugen und einzustellen und dadurch eine Fensterpriorität bei Bildschirmpufferstellen festzulegen.

10. Anzeigesystem nach Anspruch 8 oder Anspruch 9, bei welchem jeder zusätzliche dazu bestimmte Bildschirmpuffer Anzeigedaten für den zugeordneten Benützer speichert und den Darstellungsbereich für jenen Benützer bildet.

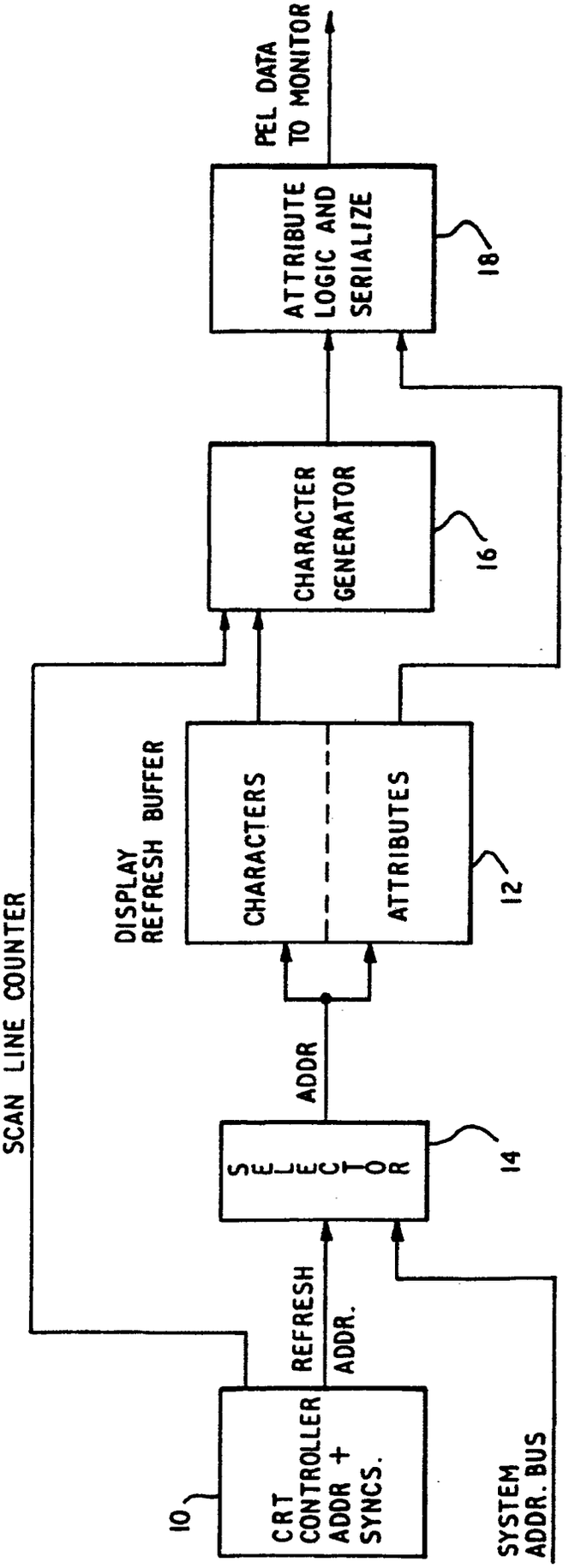

PEL DATA TO MONITOR
ATTRIBUTE LOGIC AND SERIALIZE
18
CHARACTER GENERATOR
16
DISPLAY REFRESH BUFFER
CHARACTERS
ATTRIBUTES
12
SCAN LINE COUNTER
ADDR
SELECTOR
14
REFRESH ADDR.
10
CRT CONTROLLER ADDR + SYNCS.
SYSTEM ADDR. BUS

FIG. 1

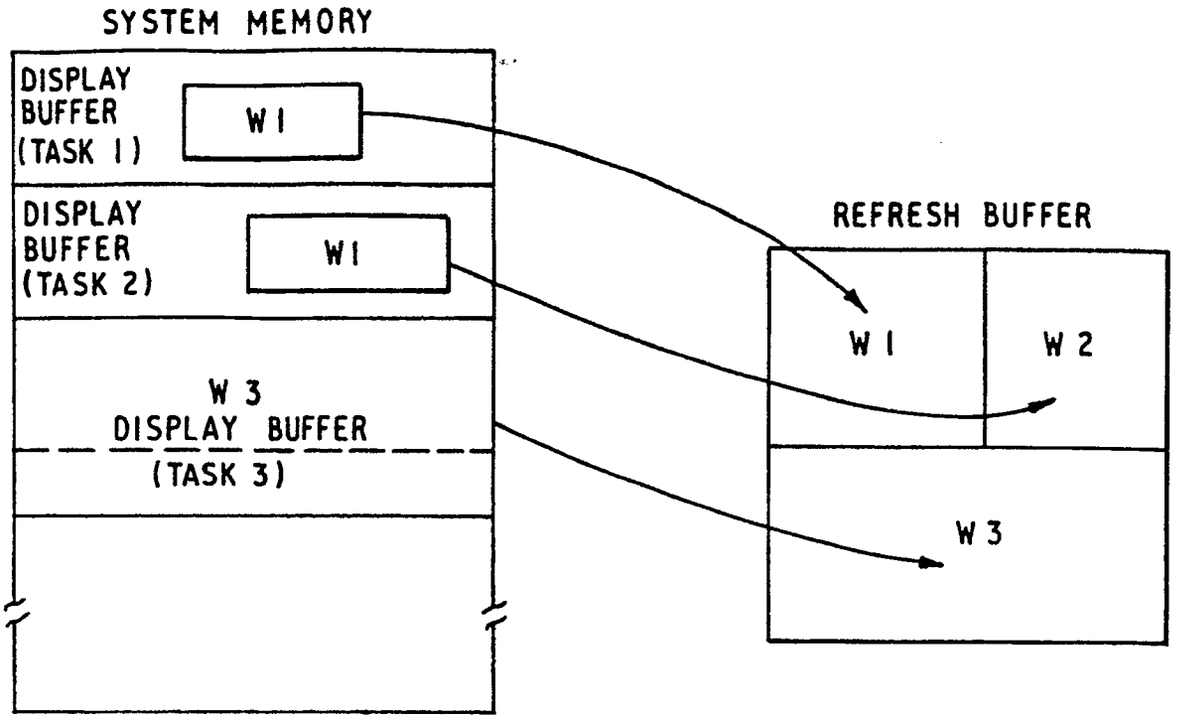

SYSTEM MEMORY
DISPLAY BUFFER (TASK 1)
W1
DISPLAY BUFFER (TASK 2)
W1
W 3 DISPLAY BUFFER (TASK 3)
REFRESH BUFFER
W1
W2
W3

FIG. 2A

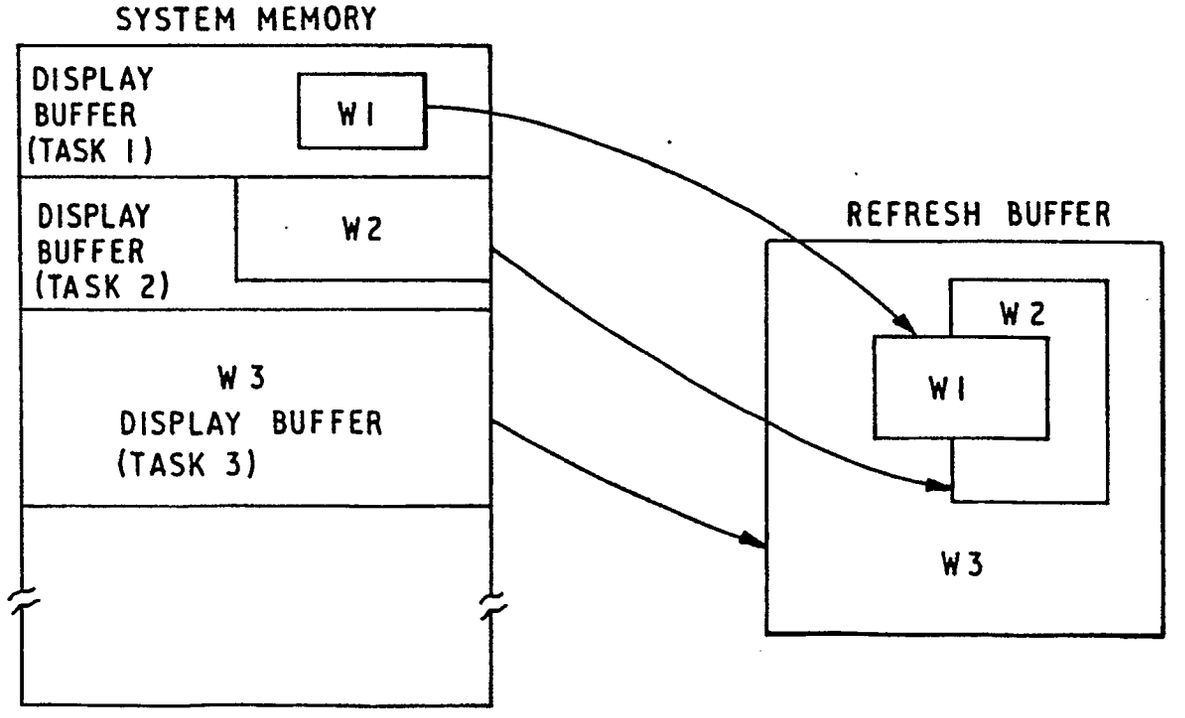

SYSTEM MEMORY
DISPLAY BUFFER (TASK 1)
W1
DISPLAY BUFFER (TASK 2)
W2
W 3 DISPLAY BUFFER (TASK 3)
REFRESH BUFFER
W2
W1
W3

FIG. 2B

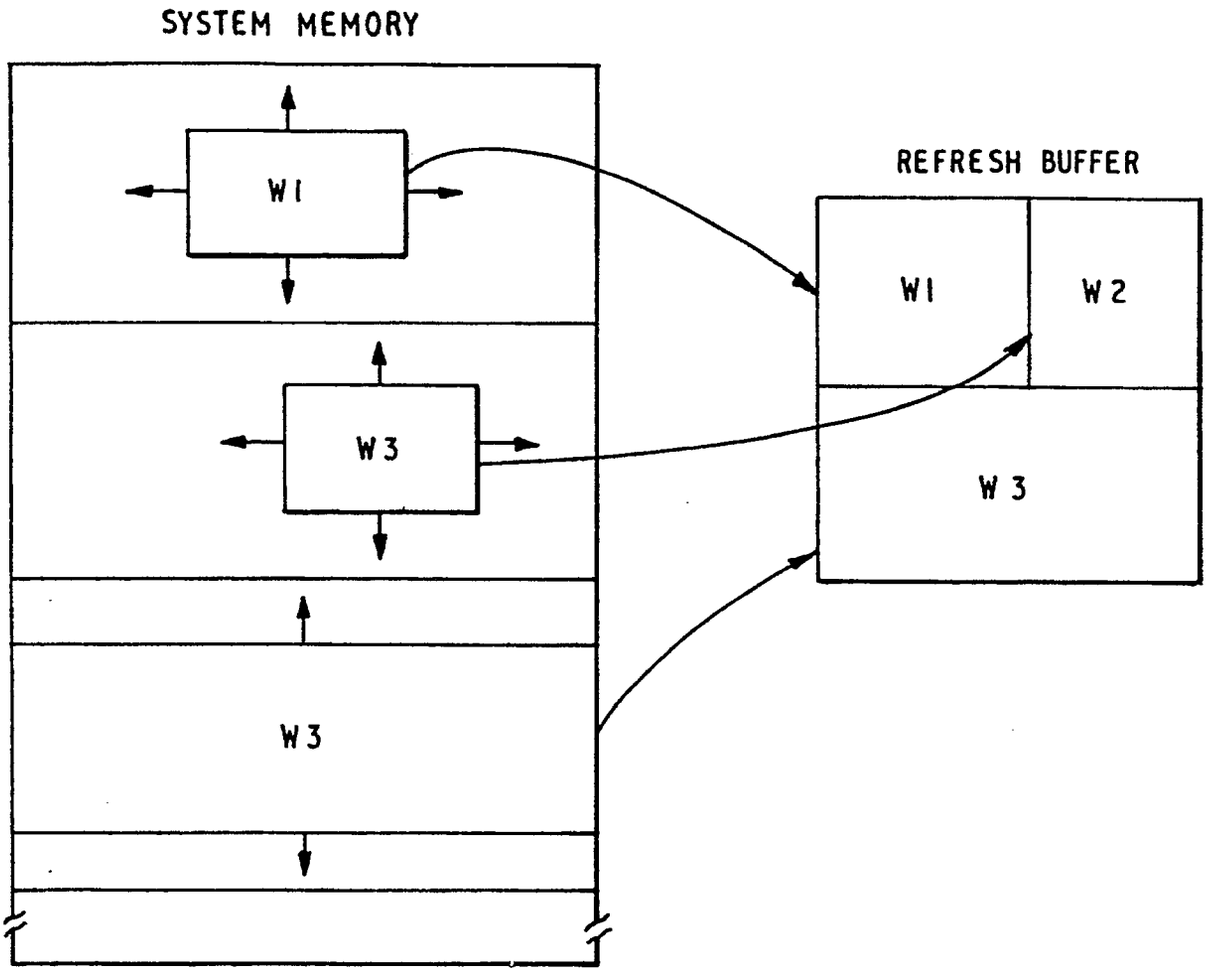
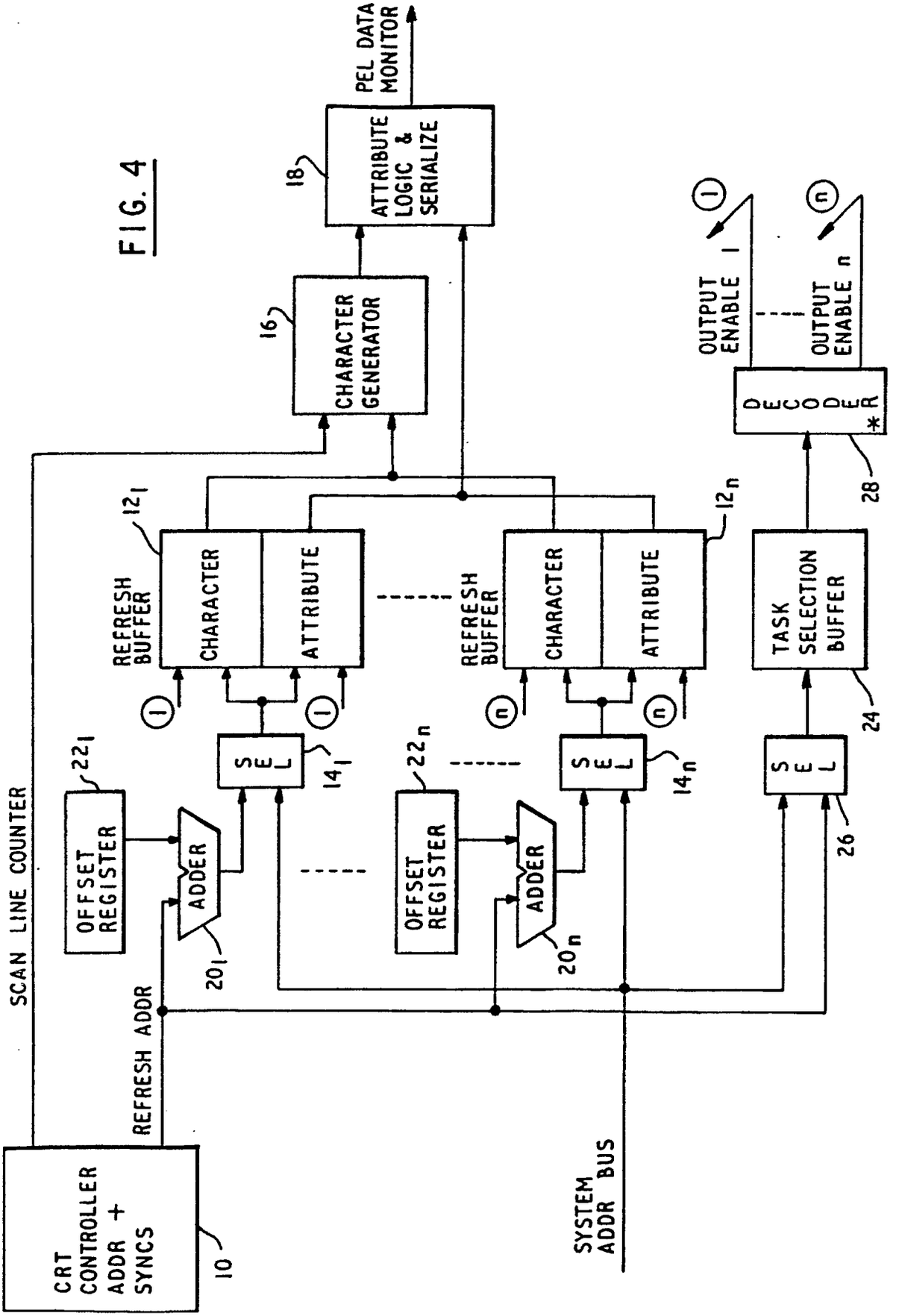


FIG. 3

FIG. 4


CRT CONTROLLER ADDR + SYNCS
10
SCAN LINE COUNTER
REFRESH ADDR
SYSTEM ADDR BUS
OFFSET REGISTER
22 1
ADDER
20 1
SEL
14 1
REFRESH BUFFER
CHARACTER
ATTRIBUTE
12 1
OFFSET REGISTER
22 n
ADDER
20 n
SEL
14 n
REFRESH BUFFER
CHARACTER
ATTRIBUTE
12 n
CHARACTER GENERATOR
16
ATTRIBUTE LOGIC & SERIALIZE
18
PEL DATA MONITOR
TASK SELECTION BUFFER
24
SEL
26
DECODER
28
OUTPUT ENABLE 1
OUTPUT ENABLE n

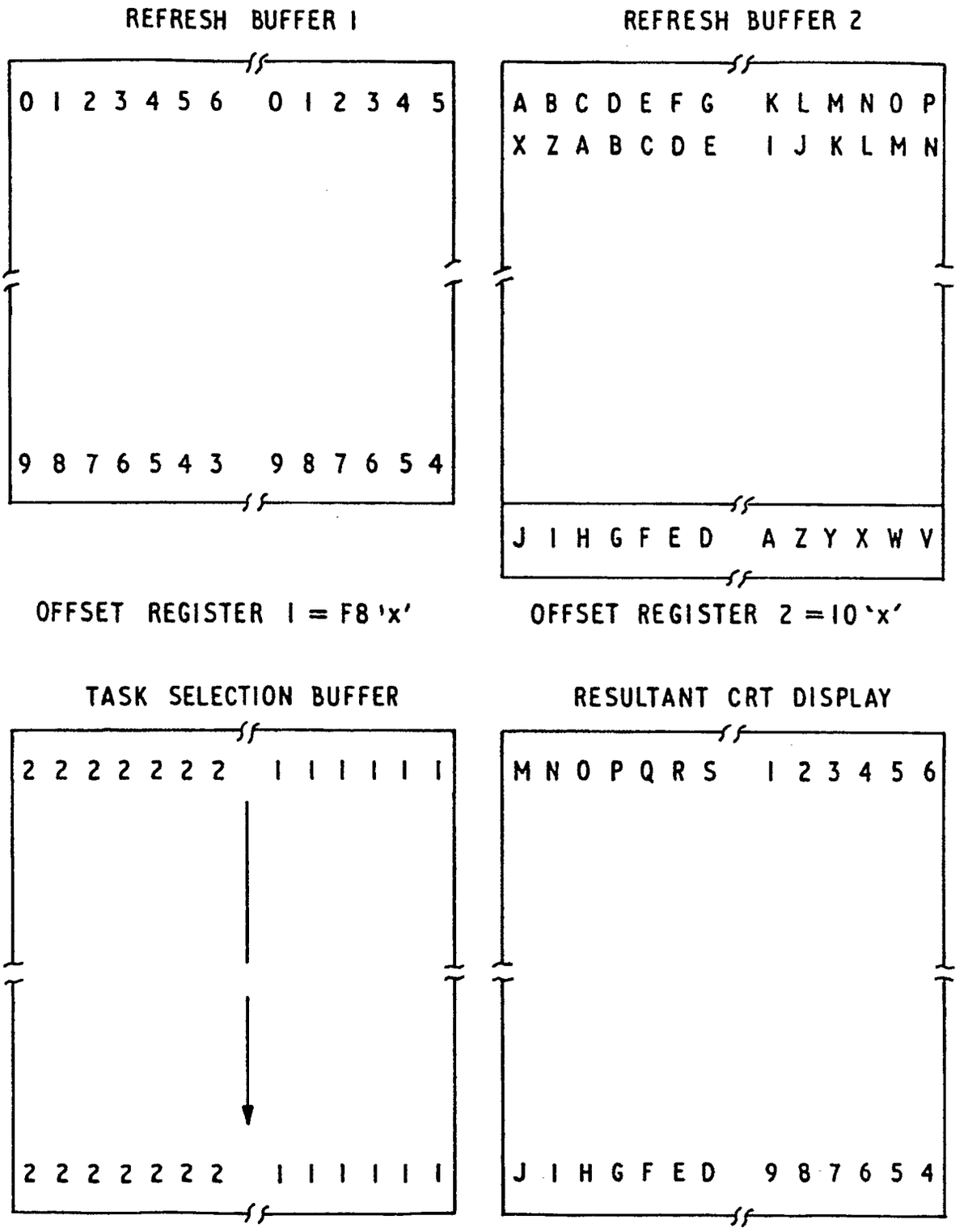
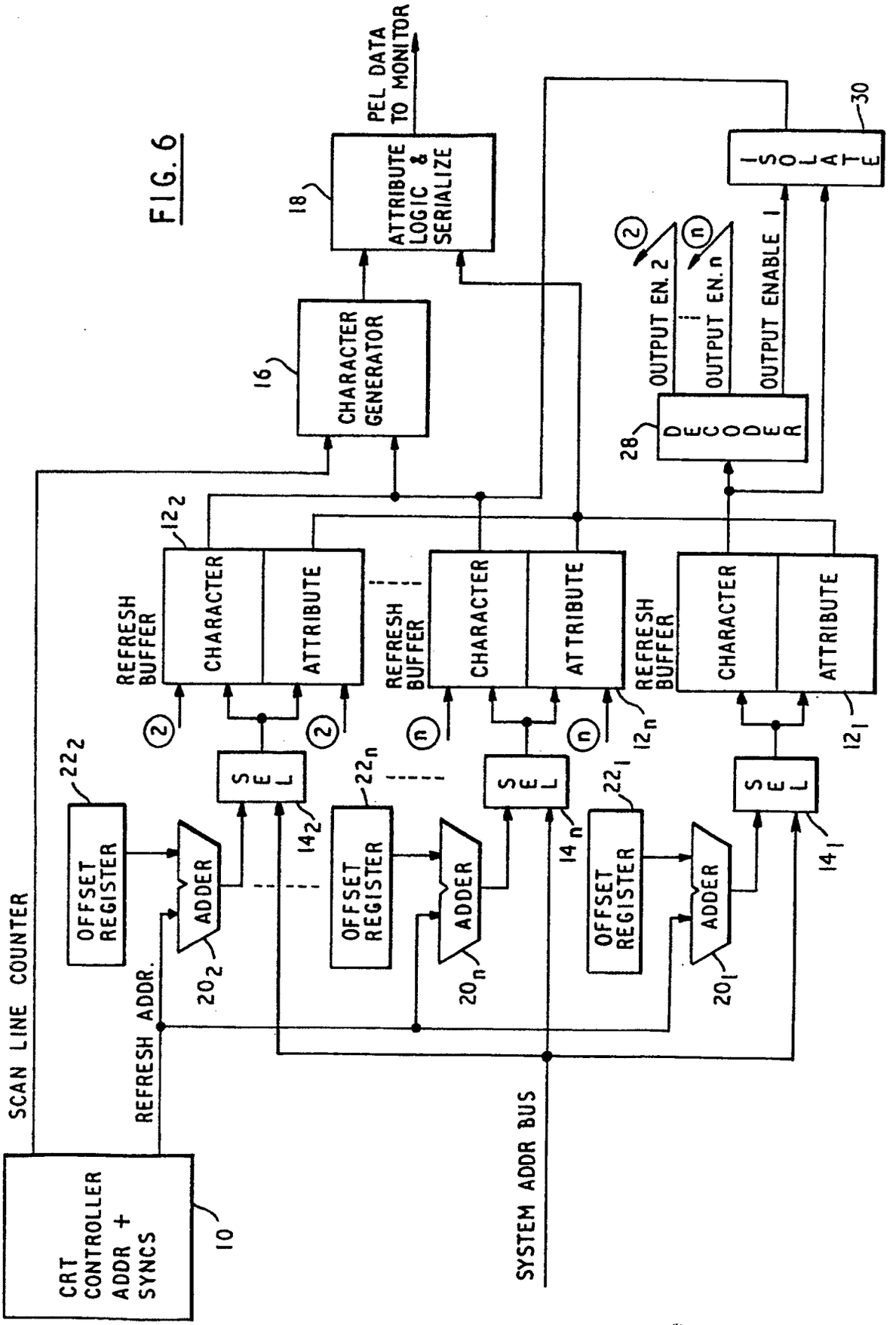

REFRESH BUFFER 1
0 1 2 3 4 5 6 0 1 2 3 4 5
9 8 7 6 5 4 3 9 8 7 6 5 4
REFRESH BUFFER 2
A B C D E F G K L M N O P
X Z A B C D E I J K L M N
J I H G F E D A Z Y X W V
OFFSET REGISTER 1 = F8 'x'
OFFSET REGISTER 2 = 10 'x'
TASK SELECTION BUFFER
2 2 2 2 2 2 2 1 1 1 1 1 1
2 2 2 2 2 2 2 1 1 1 1 1 1
RESULTANT CRT DISPLAY
M N O P Q R S 1 2 3 4 5 6
J I H G F E D 9 8 7 6 5 4

FIG. 5

FIG. 6


CRT CONTROLLER ADDR + SYNCS
10
SCAN LINE COUNTER
REFRESH ADDR.
SYSTEM ADDR BUS
OFFSET REGISTER
ADDER
SEL
REFRESH BUFFER
CHARACTER
ATTRIBUTE
CHARACTER GENERATOR
16
ATTRIBUTE LOGIC & SERIALIZE
18
PEL DATA TO MONITOR
DECODER
28
OUTPUT EN. 2
OUTPUT EN. n
OUTPUT ENABLE 1
ISOLATE
30

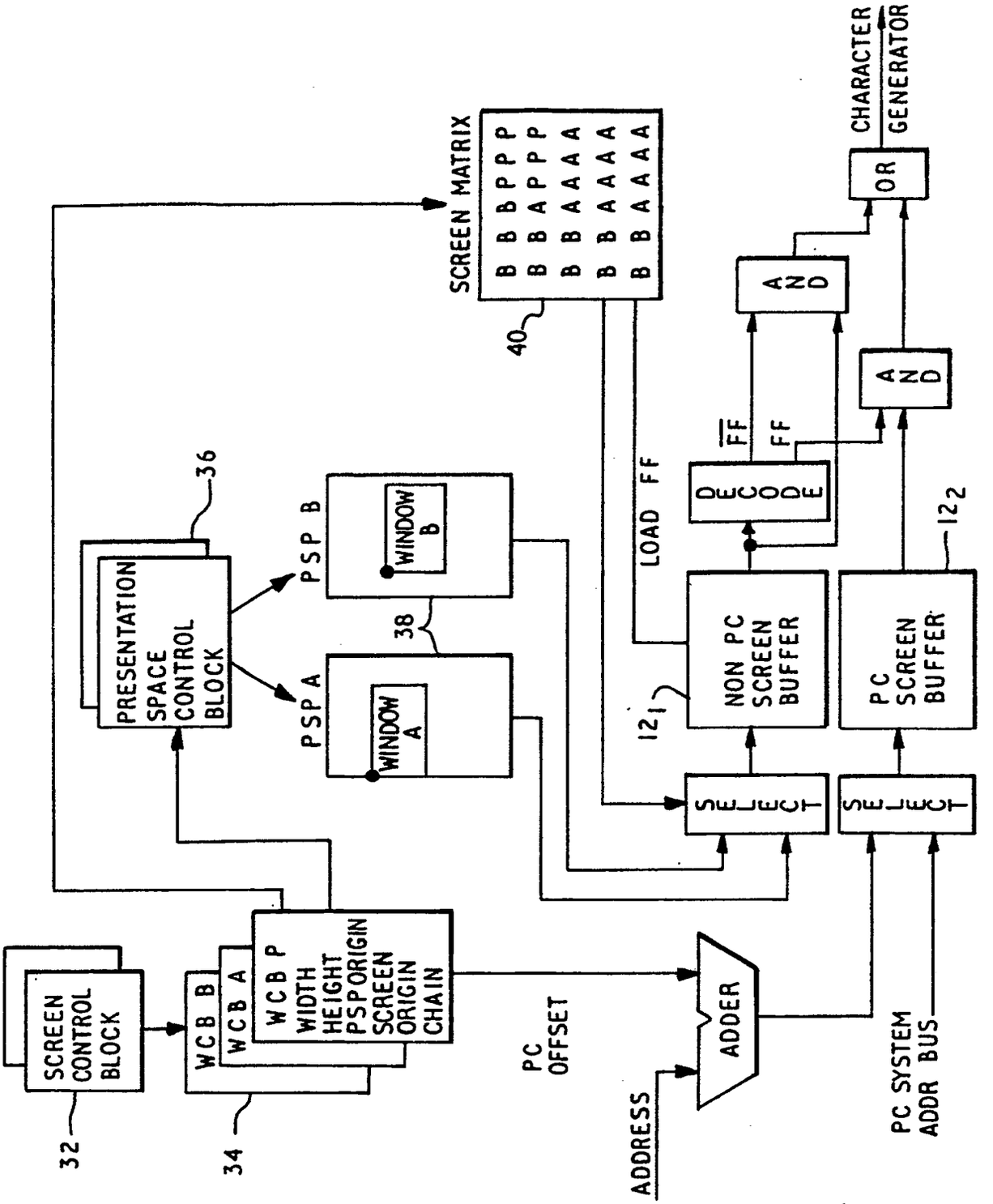

SCREEN CONTROL BLOCK
32
WCB B
WCB A
WCB P
WIDTH
HEIGHT
PSP ORIGIN
SCREEN ORIGIN
CHAIN
34
PC OFFSET
ADDRESS
ADDER
PC SYSTEM ADDR BUS
SELECT
SELECT
NON PC SCREEN BUFFER
121
PC SCREEN BUFFER
122
LOAD FF
DECODE
FF
FF
AND
AND
OR
CHARACTER GENERATOR
SCREEN MATRIX
B B B P P P P
B B A P P P P
B B A A A A A
B B A A A A A
B B A A A A A
40
PRESENTATION SPACE CONTROL BLOCK
36
PSP B
WINDOW B
38
PSP A
WINDOW A

FIG. 7

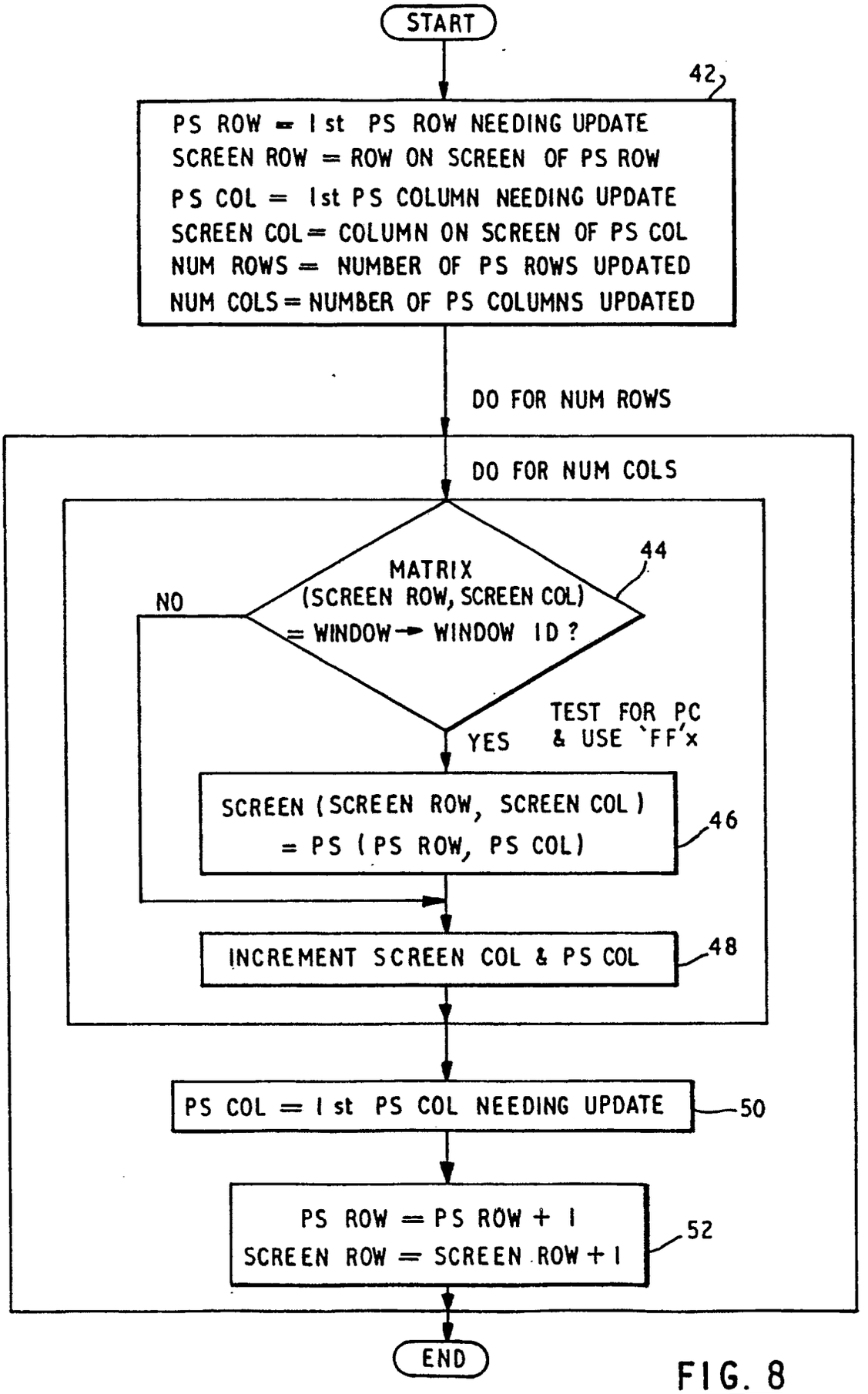

START
42
PS ROW = 1st PS ROW NEEDING UPDATE
SCREEN ROW = ROW ON SCREEN OF PS ROW
PS COL = 1st PS COLUMN NEEDING UPDATE
SCREEN COL = COLUMN ON SCREEN OF PS COL
NUM ROWS = NUMBER OF PS ROWS UPDATED
NUM COLS = NUMBER OF PS COLUMNS UPDATED
DO FOR NUM ROWS
DO FOR NUM COLS
44
MATRIX
(SCREEN ROW, SCREEN COL)
= WINDOW→ WINDOW ID ?
NO
YES
TEST FOR PC
& USE 'FF'x
SCREEN (SCREEN ROW, SCREEN COL)
= PS (PS ROW, PS COL)
46
INCREMENT SCREEN COL & PS COL
48
PS COL = 1st PS COL NEEDING UPDATE
50
PS ROW = PS ROW + 1
SCREEN ROW = SCREEN ROW + 1
52
END

FIG. 8

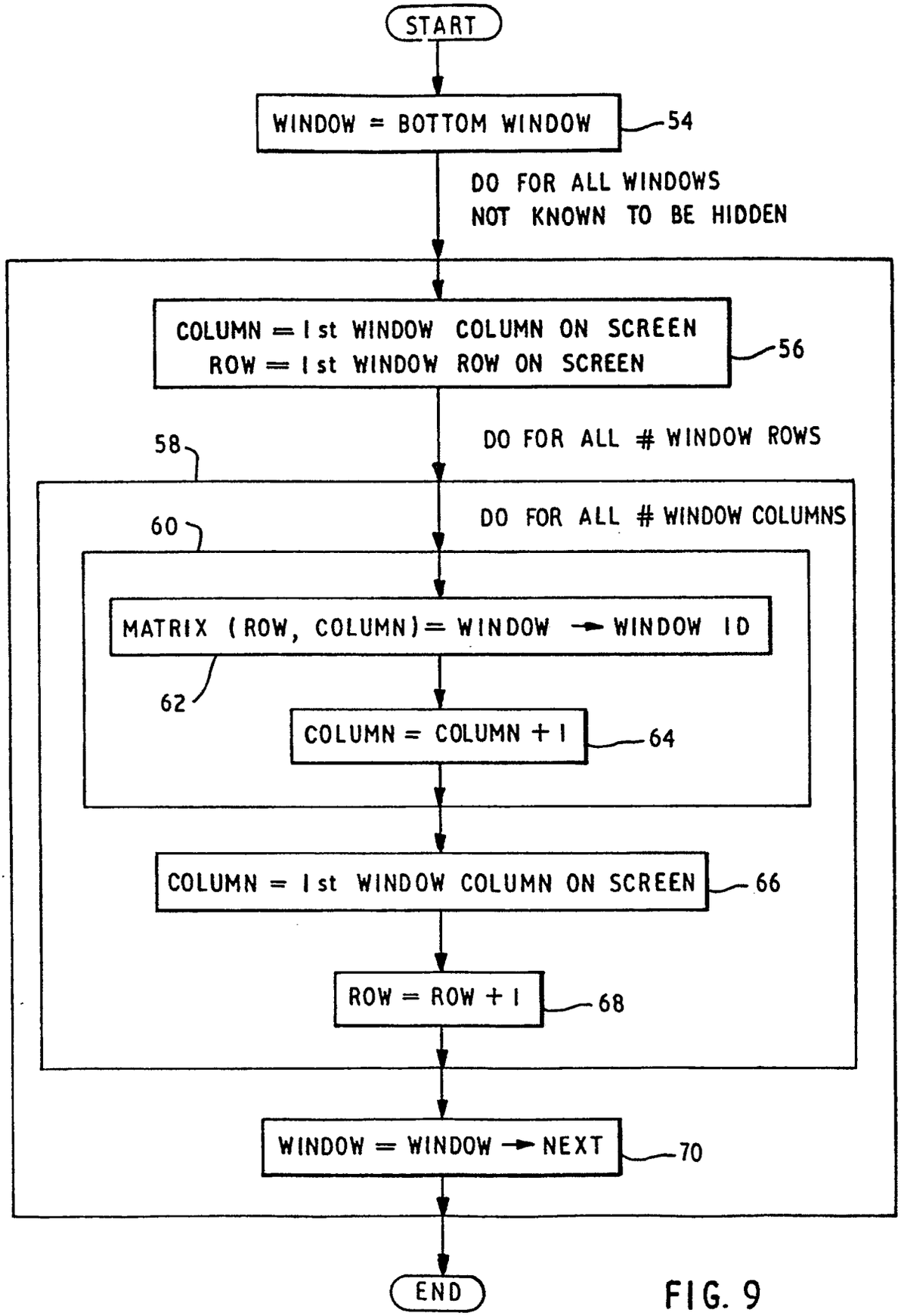

START
WINDOW = BOTTOM WINDOW
54
DO FOR ALL WINDOWS
NOT KNOWN TO BE HIDDEN
COLUMN = 1st WINDOW COLUMN ON SCREEN
ROW = 1st WINDOW ROW ON SCREEN
56
58
DO FOR ALL # WINDOW ROWS
60
DO FOR ALL # WINDOW COLUMNS
MATRIX (ROW, COLUMN) = WINDOW → WINDOW ID
62
COLUMN = COLUMN + 1
64
COLUMN = 1st WINDOW COLUMN ON SCREEN
66
ROW = ROW + 1
68
WINDOW = WINDOW → NEXT
70
END

FIG. 9