United States Patent [19]

Ott

[11] E **Patent Number: Re. 31,977**

[45] **Reissued Date of Patent: Aug. 27, 1985**

[54] **DIGITAL COMPUTING SYSTEM HAVING AUTO-INCREMENTING MEMORY**

[75] Inventor: **Granville E. Ott**, Lubbock, Tex.

[73] Assignee: **Texas Instruments Incorporated**, Dallas, Tex.

[21] Appl. No.: **524,629**

[22] Filed: **Aug. 19, 1983**

Related U.S. Patent Documents

Reissue of:

[64] Patent No.: **4,286,320**
Issued: **Aug. 25, 1981**
Appl. No.: **20,111**
Filed: **Mar. 12, 1979**

[51] **Int. Cl.**[3] **G06F 13/00**; G06F 13/06

[52] **U.S. Cl.** **364/200**

[58] **Field of Search** ..... 364/200 MS, 900 MS

[56] **References Cited**

U.S. PATENT DOCUMENTS

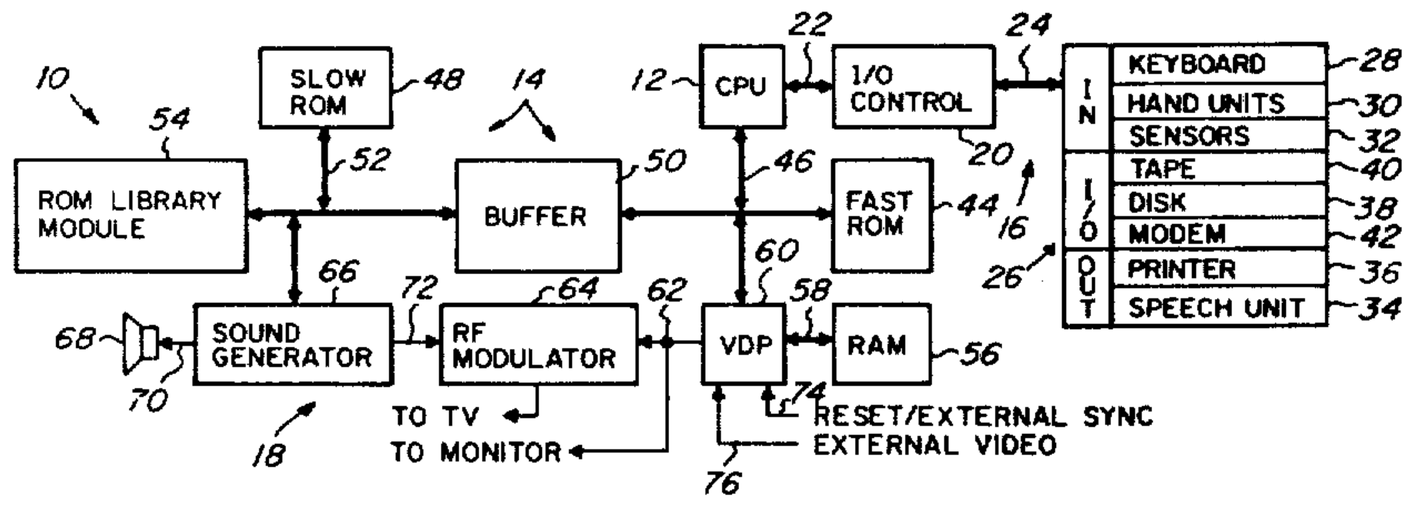
Re. 31,318	7/1983	Kaufman et al.	364/200
3,497,630	2/1970	Lucas et al.	179/18 ES
3,626,374	2/1970	Chinlund	364/200
3,691,531	9/1972	Saltini et al.	364/200
3,702,988	11/1972	Haney et al.	364/200
3,866,180	2/1975	Willette	364/200
3,914,747	10/1975	Barnes et al.	364/200
3,972,025	7/1976	Taddei	364/200
3,972,028	7/1976	Weber et al.	364/200
4,034,349	7/1977	Monaco et al.	364/200
4,040,029	8/1977	Young	364/900
4,079,454	3/1978	Sorenson et al.	364/200
4,079,459	3/1978	Grant, Jr.	364/900
4,086,626	4/1978	Chung	364/200
4,124,892	11/1978	Jones et al.	364/200

Primary Examiner—Harvey E. Springborn
Attorney, Agent, or Firm—Leo N. Heiting; Robert D. Marshall, Jr.; Melvin Sharp

[57] **ABSTRACT**

A digital computing system having an auto-incrementing memory subsystem includes one or more separate memories. Each memory in the memory subsystem has its own address counter which is automatically incremented to the next sequential address after each memory readout. In the case of plural memories, a page select enables memory readout only when a page designation portion of the address matches a unique page number associated with the memory. An automatic memory refresh is provided by a refresh address counter along with a refresh address incrementer in the case that a memory is a dynamic RAM. These features enable improved performance in the digital computing system by reducing the required CPU overhead for memory subsystem control.

11 Claims, 14 Drawing Figures



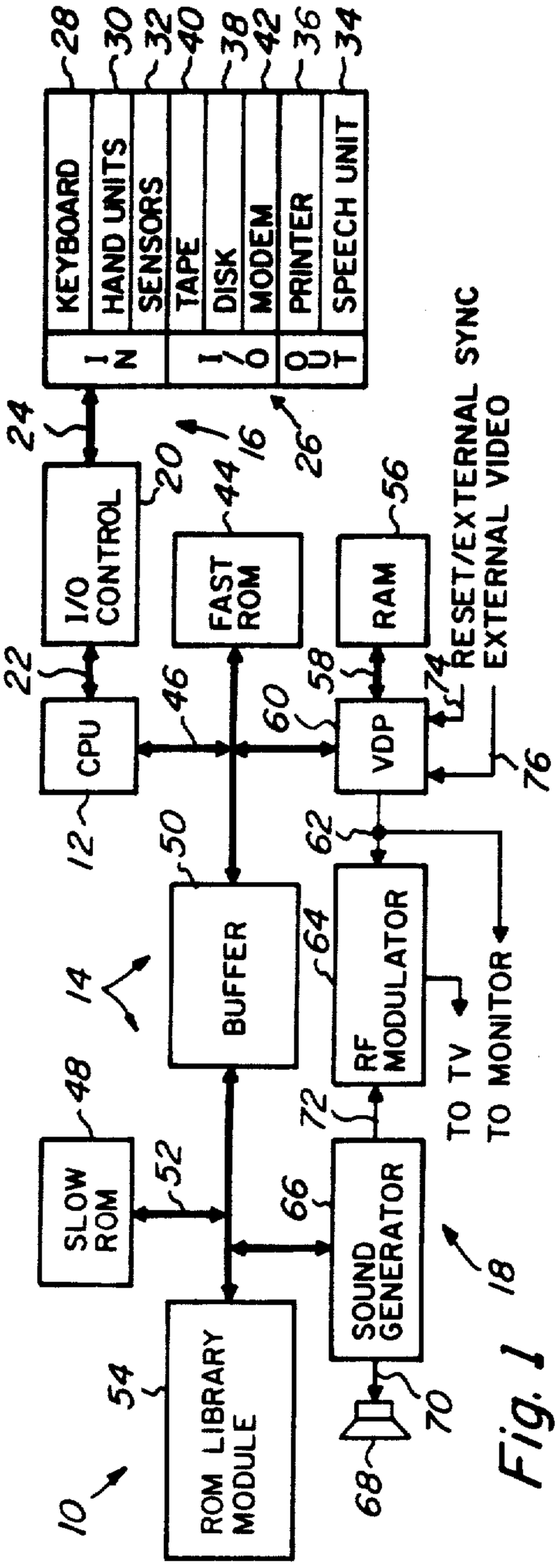

KEYBOARD
HAND UNITS
SENSORS
TAPE
DISK
MODEM
PRINTER
SPEECH UNIT
IN
I/O
OUT
28
30
32
40
38
42
36
34
24
20
16
26
I/O CONTROL
FAST ROM
RAM
44
56
22
46
60
58
74
CPU
VDP
RESET/EXTERNAL SYNC
EXTERNAL VIDEO
12
50
62
14
BUFFER
64
RF MODULATOR
TO TV
TO MONITOR
48
52
66
72
SLOW ROM
SOUND GENERATOR
54
68
70
18
10
ROM LIBRARY MODULE

Fig. 1

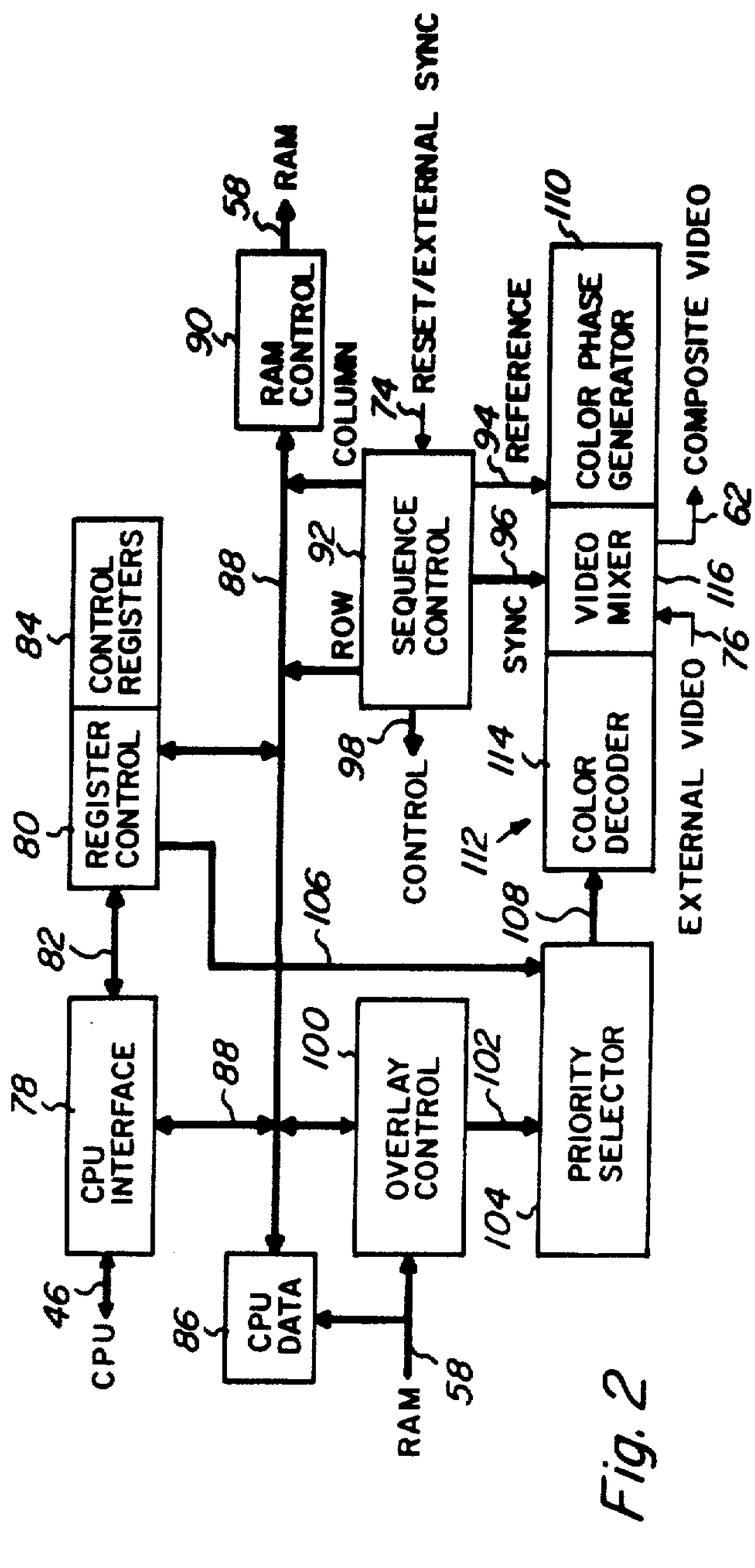

84
CONTROL REGISTERS
88
90
RAM CONTROL
58
RAM
ROW
92
COLUMN
SEQUENCE CONTROL
74
RESET/EXTERNAL SYNC
94
REFERENCE
96
SYNC
COLOR PHASE GENERATOR
110
VIDEO MIXER
COMPOSITE VIDEO
62
116
76
EXTERNAL VIDEO
80
REGISTER CONTROL
82
CPU INTERFACE
78
46
CPU
86
CPU DATA
100
OVERLAY CONTROL
102
98
CONTROL
112
114
COLOR DECODER
108
106
104
PRIORITY SELECTOR
RAM
58

Fig. 2

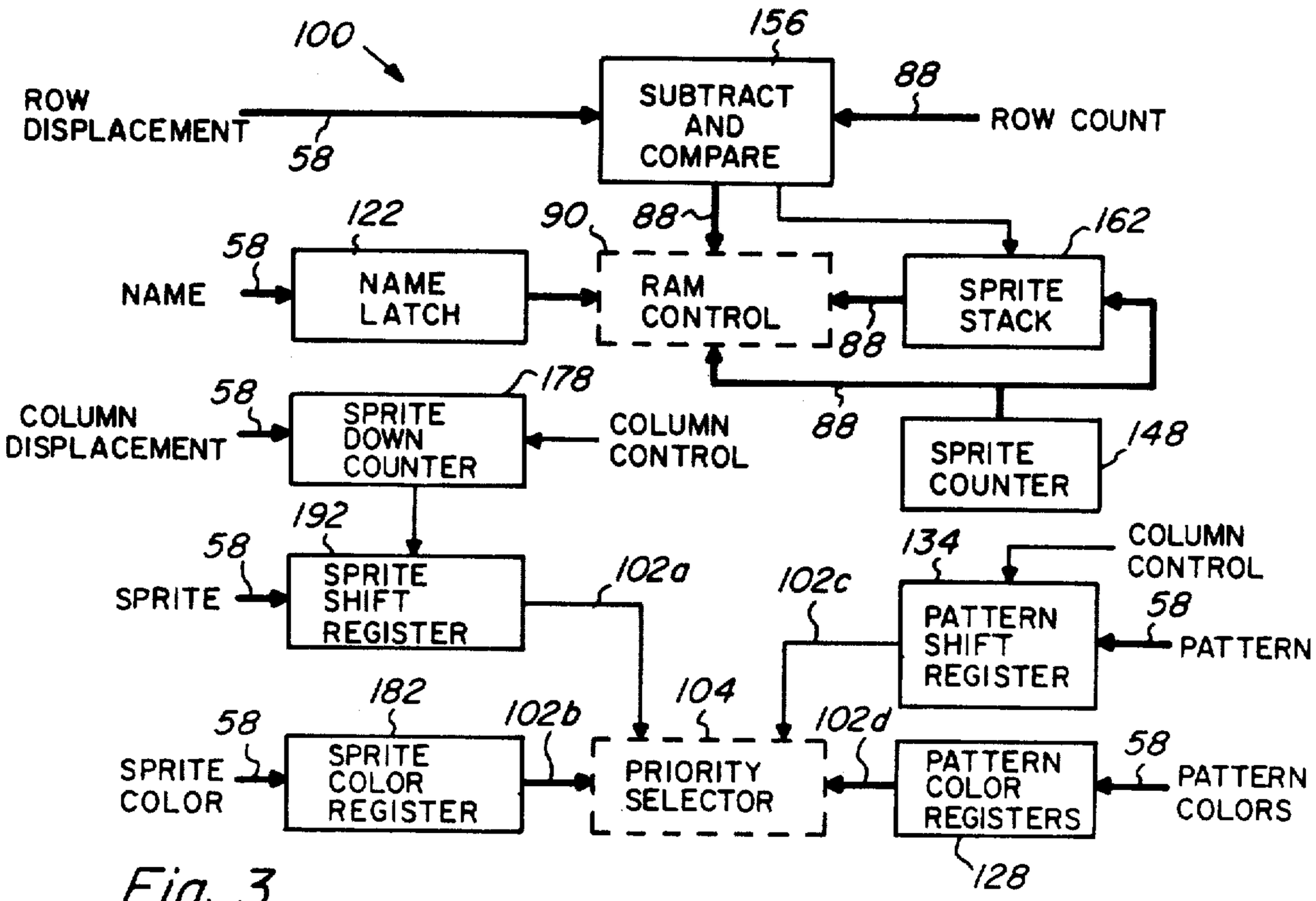

100
156
ROW DISPLACEMENT
58
SUBTRACT AND COMPARE
88
ROW COUNT
122
90
88
162
58
NAME
NAME LATCH
RAM CONTROL
SPRITE STACK
88
178
58
COLUMN DISPLACEMENT
SPRITE DOWN COUNTER
COLUMN CONTROL
88
SPRITE COUNTER
148
192
134
COLUMN CONTROL
58
SPRITE
SPRITE SHIFT REGISTER
102a
102c
PATTERN SHIFT REGISTER
58
PATTERN
182
104
58
102b
102d
58
SPRITE COLOR
SPRITE COLOR REGISTER
PRIORITY SELECTOR
PATTERN COLOR REGISTERS
58
PATTERN COLORS
128

Fig. 3

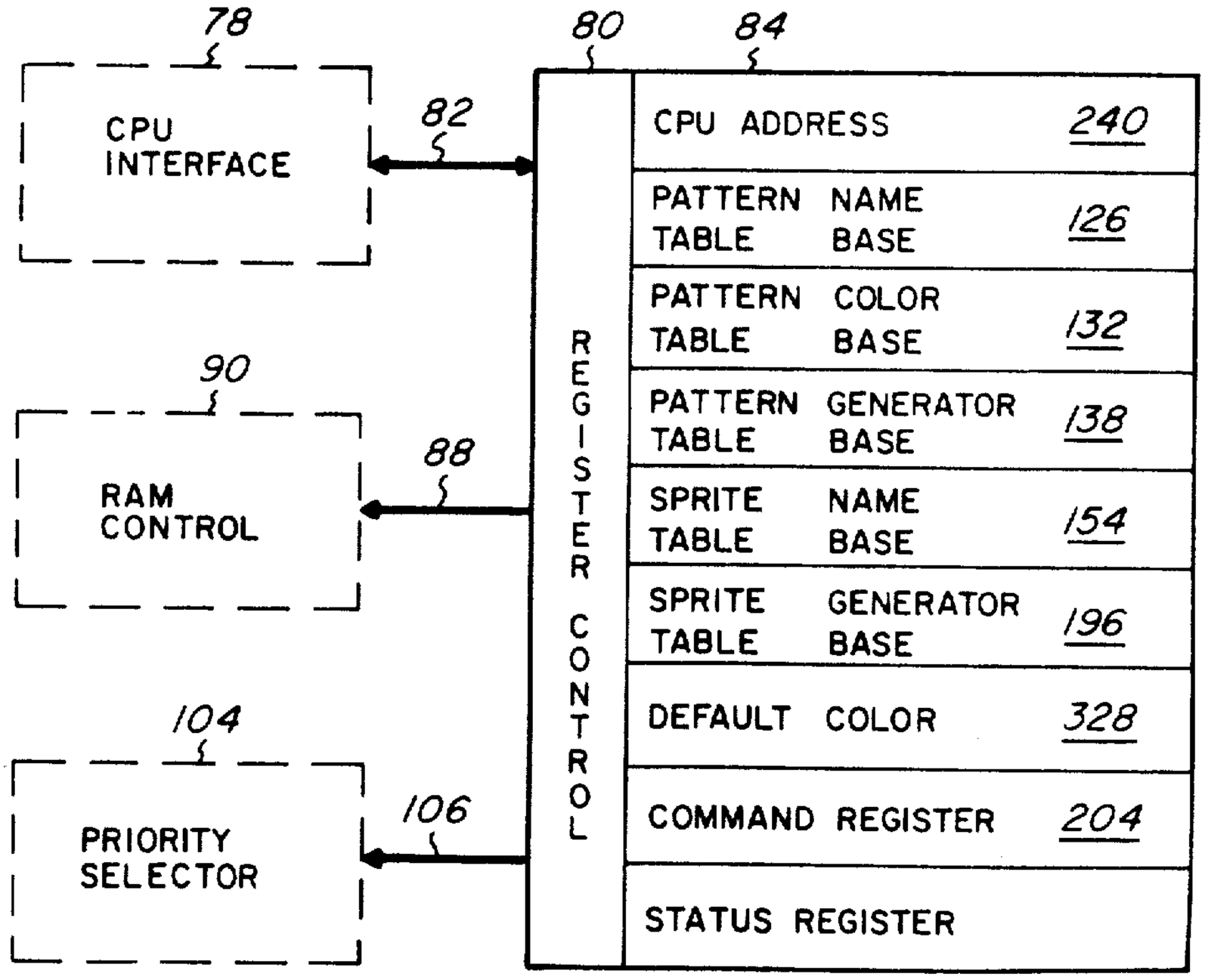

78
80
84
CPU INTERFACE
82
CPU ADDRESS 240
PATTERN NAME TABLE BASE 126
PATTERN COLOR TABLE BASE 132
PATTERN GENERATOR TABLE BASE 138
SPRITE NAME TABLE BASE 154
SPRITE GENERATOR TABLE BASE 196
DEFAULT COLOR 328
COMMAND REGISTER 204
STATUS REGISTER
REGISTER CONTROL
90
88
RAM CONTROL
104
106
PRIORITY SELECTOR

Fig. 4

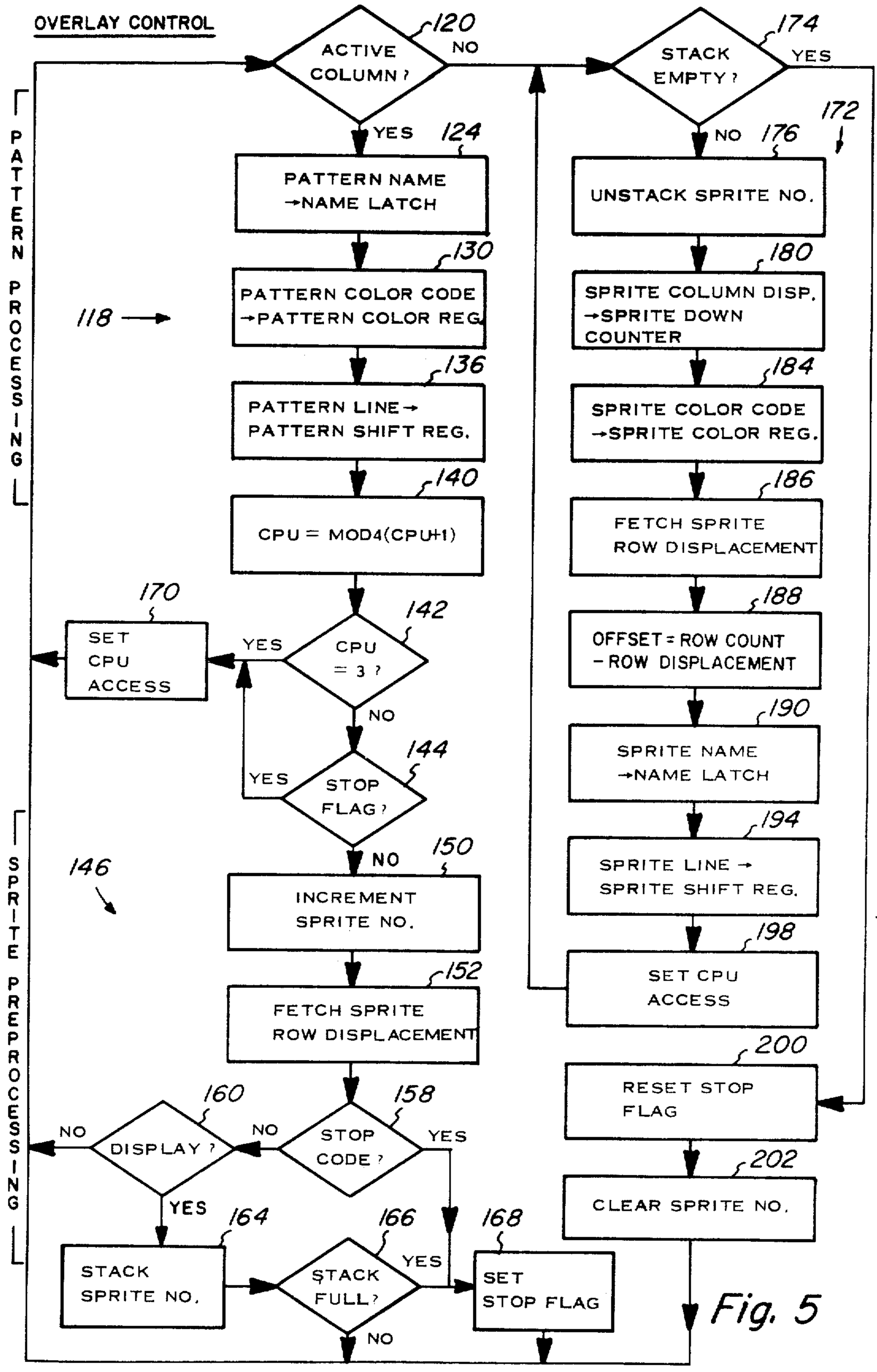

OVERLAY CONTROL
PATTERN PROCESSING
SPRITE PREPROCESSING
SPRITE POST PROCESSING
ACTIVE COLUMN?
120
NO
YES
124
PATTERN NAME →NAME LATCH
130
PATTERN COLOR CODE →PATTERN COLOR REG.
118
136
PATTERN LINE→ PATTERN SHIFT REG.
140
CPU = MOD4(CPU+1)
170
SET CPU ACCESS
142
CPU = 3?
YES
NO
144
STOP FLAG?
YES
146
NO
150
INCREMENT SPRITE NO.
152
FETCH SPRITE ROW DISPLACEMENT
160
DISPLAY?
NO
YES
164
STACK SPRITE NO.
158
STOP CODE?
NO
YES
166
STACK FULL?
YES
NO
168
SET STOP FLAG
STACK EMPTY?
174
YES
NO
176
172
UNSTACK SPRITE NO.
180
SPRITE COLUMN DISP. →SPRITE DOWN COUNTER
184
SPRITE COLOR CODE →SPRITE COLOR REG.
186
FETCH SPRITE ROW DISPLACEMENT
188
OFFSET = ROW COUNT - ROW DISPLACEMENT
190
SPRITE NAME →NAME LATCH
194
SPRITE LINE → SPRITE SHIFT REG.
198
SET CPU ACCESS
200
RESET STOP FLAG
202
CLEAR SPRITE NO.

Fig. 5

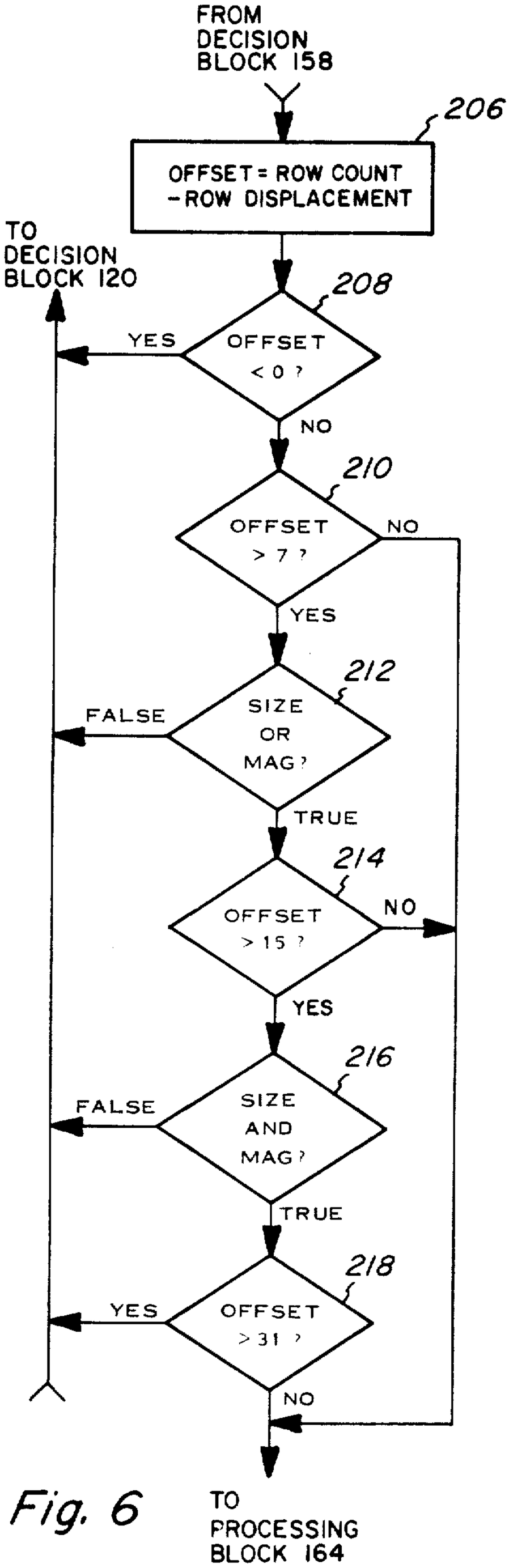

FROM
DECISION
BLOCK 158
206
OFFSET = ROW COUNT
- ROW DISPLACEMENT
TO
DECISION
BLOCK 120
208
YES
OFFSET
< 0 ?
NO
210
OFFSET
> 7 ?
NO
YES
212
FALSE
SIZE
OR
MAG?
TRUE
214
OFFSET
> 15 ?
NO
YES
216
FALSE
SIZE
AND
MAG?
TRUE
218
YES
OFFSET
> 31 ?
NO
TO
PROCESSING
BLOCK 164

Fig. 6

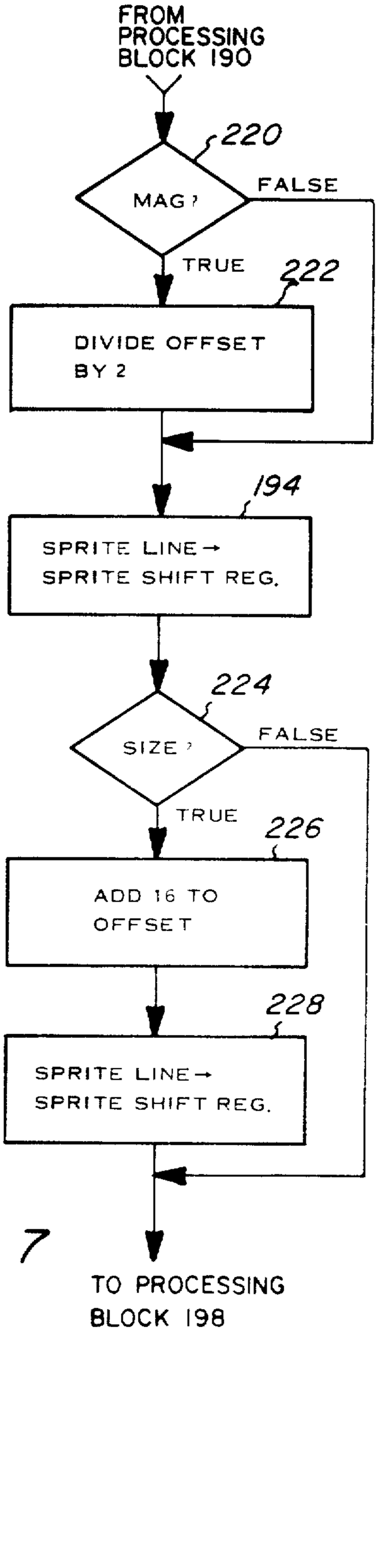

FROM
PROCESSING
BLOCK 190
220
MAG?
FALSE
TRUE
222
DIVIDE OFFSET
BY 2
194
SPRITE LINE→
SPRITE SHIFT REG.
224
SIZE?
FALSE
TRUE
226
ADD 16 TO
OFFSET
228
SPRITE LINE→
SPRITE SHIFT REG.
TO PROCESSING
BLOCK 198

Fig. 7

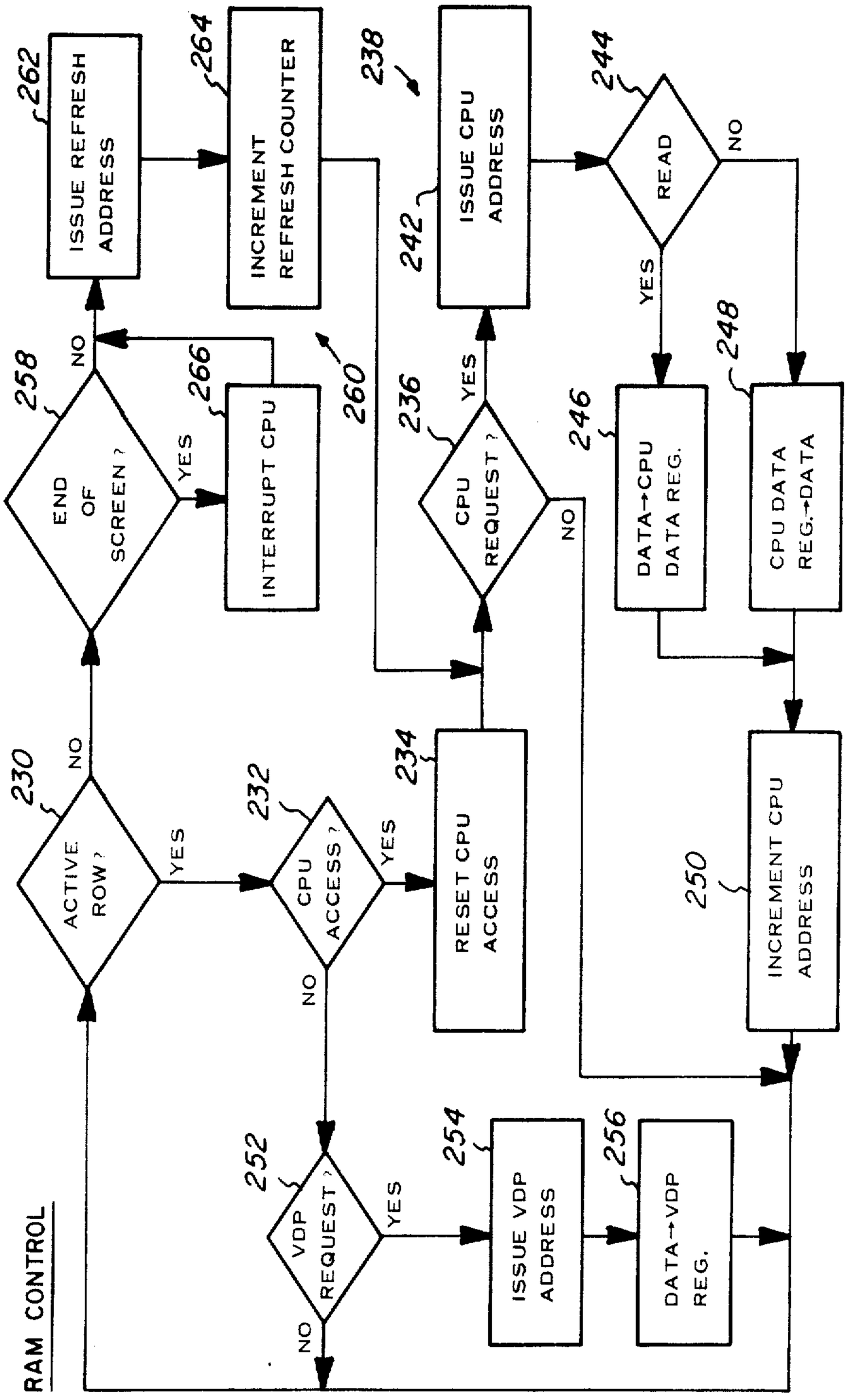

RAM CONTROL
ACTIVE ROW ?
YES
NO
230
CPU ACCESS ?
YES
NO
232
RESET CPU ACCESS
234
VDP REQUEST ?
YES
NO
252
ISSUE VDP ADDRESS
254
DATA→VDP REG.
256
INCREMENT CPU ADDRESS
250
CPU REQUEST ?
YES
NO
236
ISSUE CPU ADDRESS
242
238
READ
YES
NO
244
DATA→CPU DATA REG.
246
CPU DATA REG.→DATA
248
END OF SCREEN ?
YES
NO
258
INTERRUPT CPU
266
260
ISSUE REFRESH ADDRESS
262
INCREMENT REFRESH COUNTER
264

Fig. 8

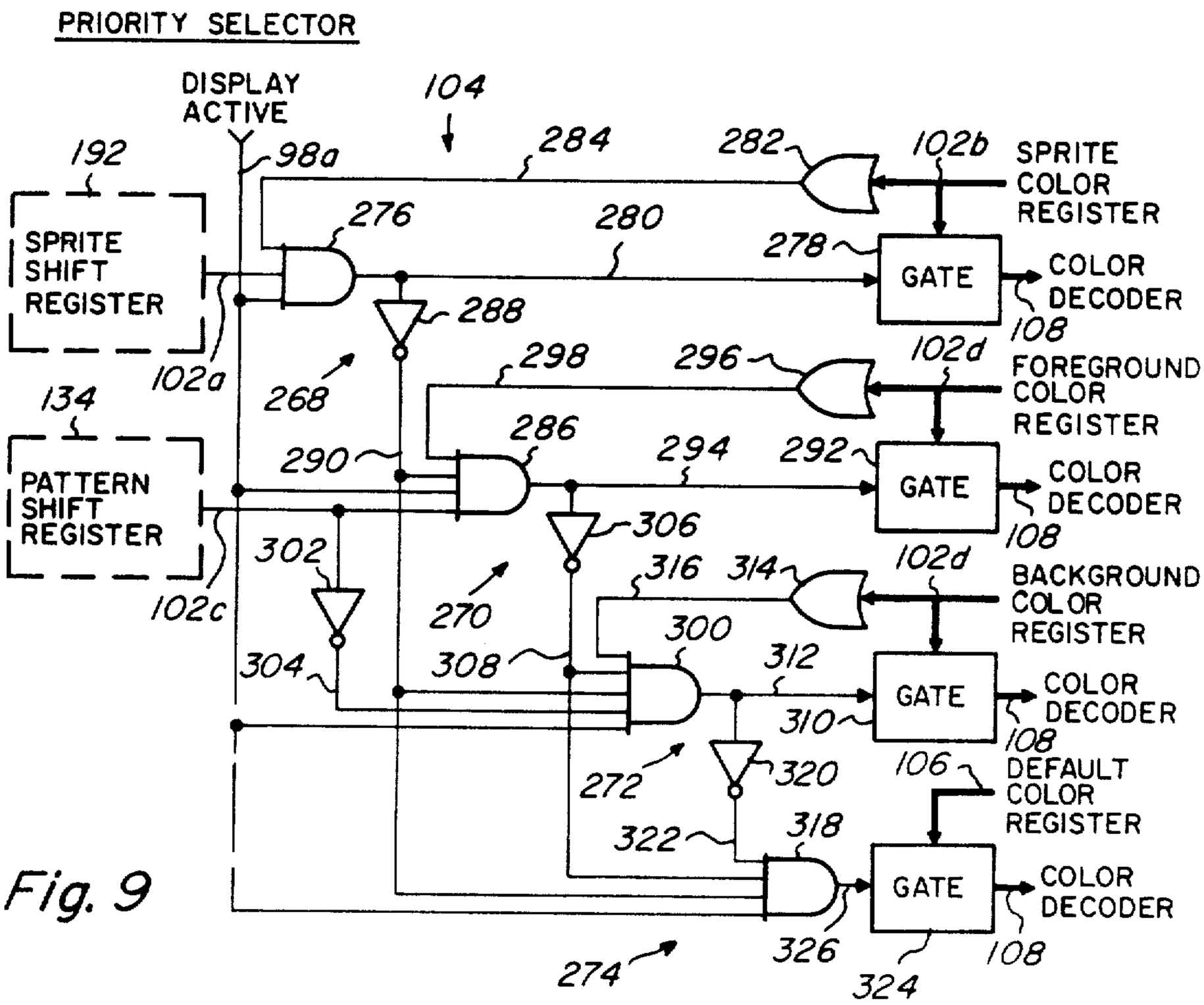

PRIORITY SELECTOR
DISPLAY ACTIVE
104
192
98a
284
282
102b
SPRITE COLOR REGISTER
276
280
278
GATE
COLOR DECODER
108
SPRITE SHIFT REGISTER
102a
288
268
298
296
102d
FOREGROUND COLOR REGISTER
134
286
290
294
292
PATTERN SHIFT REGISTER
306
302
316
314
102d
BACKGROUND COLOR REGISTER
102c
270
300
304
308
312
310
272
320
106
DEFAULT COLOR REGISTER
322
318
326
324
274

Fig. 9

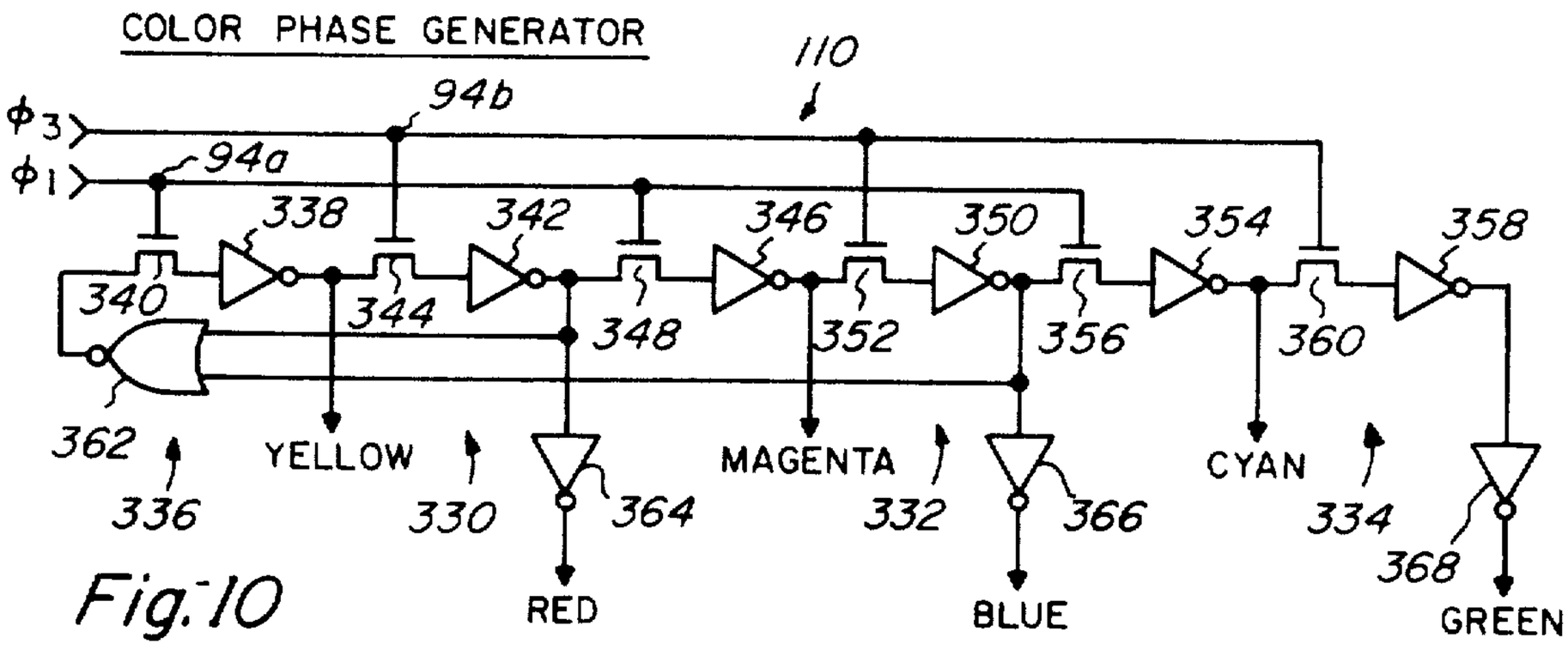

COLOR PHASE GENERATOR
110
94b
ϕ_3
ϕ_1
94a
338
342
346
350
354
358
340
344
348
352
356
360
362
336
YELLOW
330
364
MAGENTA
332
366
CYAN
334
368
RED
BLUE
GREEN

Fig. 10

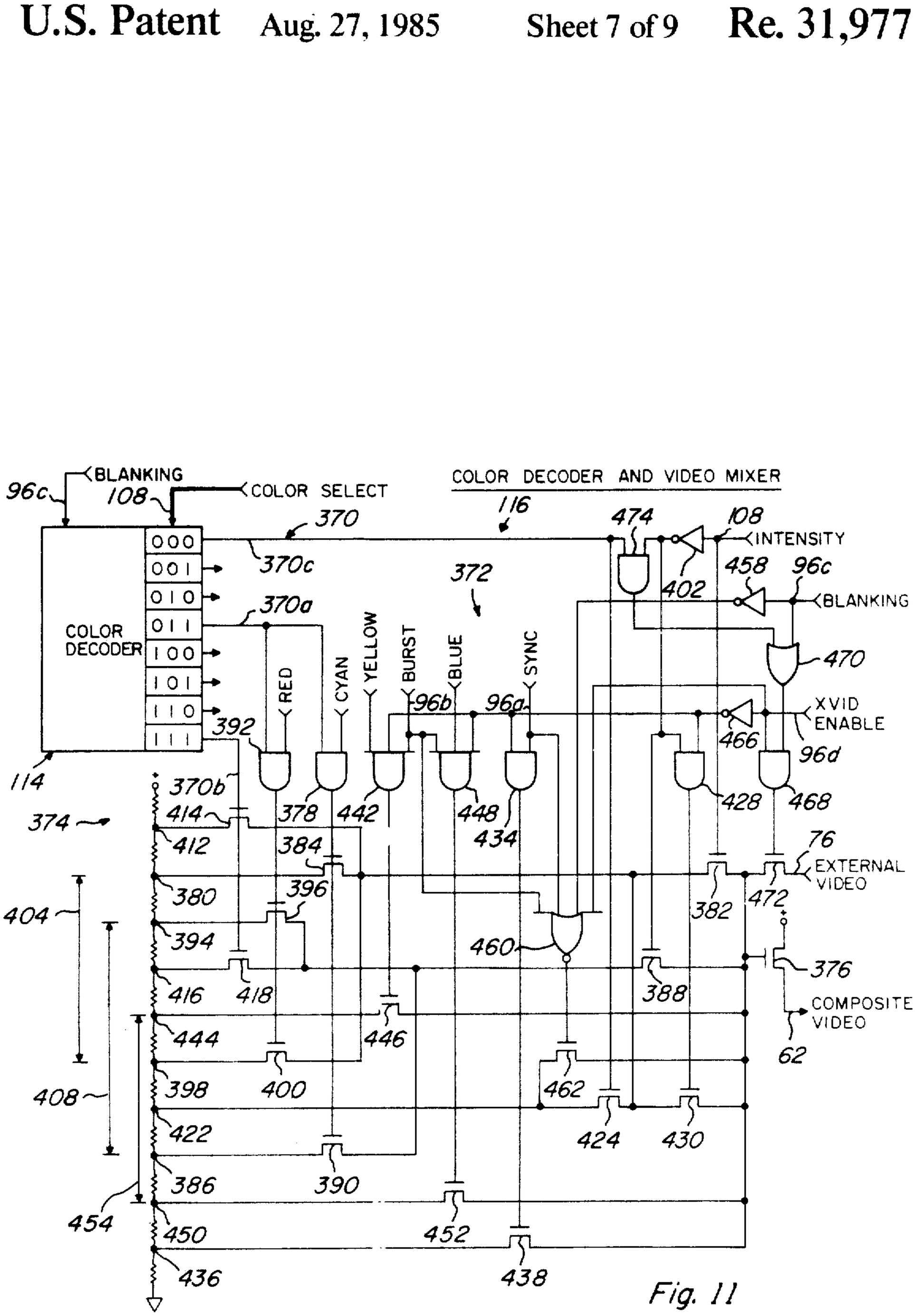

COLOR DECODER AND VIDEO MIXER
BLANKING
COLOR SELECT
96c
108
116
370
370c
370a
370b
372
374
COLOR DECODER
000
001
010
011
100
101
110
111
114
392
RED
CYAN
YELLOW
BURST
BLUE
SYNC
96b
96a
474
108
INTENSITY
402
458
96c
BLANKING
470
XVID ENABLE
466
96d
428
468
76
EXTERNAL VIDEO
382
472
376
COMPOSITE VIDEO
62
388
460
462
424
430
378
442
448
434
384
396
380
394
416
418
444
446
398
400
422
390
386
450
452
438
436
404
408
454

Fig. 11

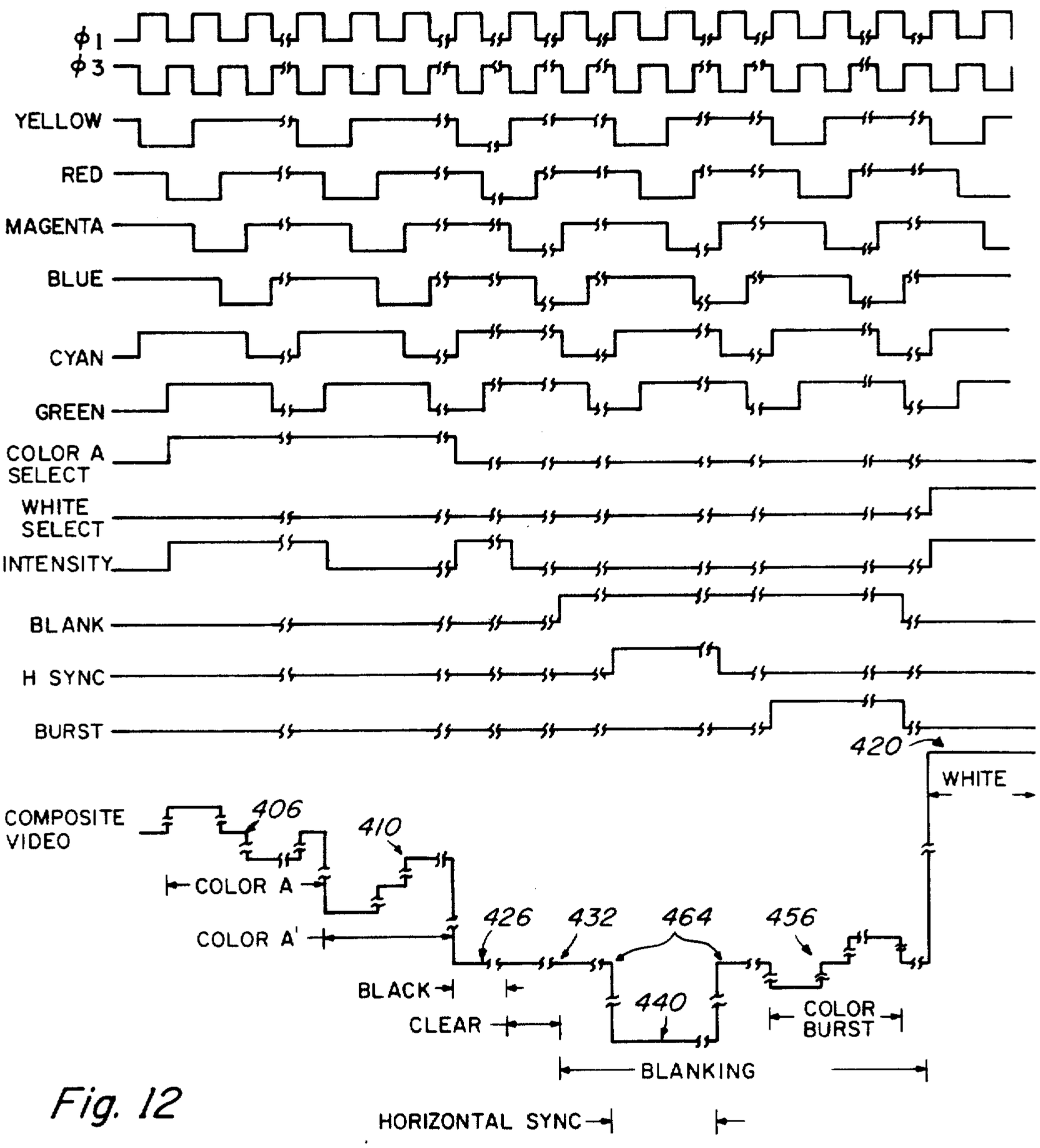

φ1
φ3
YELLOW
RED
MAGENTA
BLUE
CYAN
GREEN
COLOR A SELECT
WHITE SELECT
INTENSITY
BLANK
H SYNC
BURST
420
WHITE
COMPOSITE VIDEO
406
410
COLOR A
COLOR A'
426
432
464
456
BLACK
CLEAR
440
COLOR BURST
BLANKING
HORIZONTAL SYNC

Fig. 12

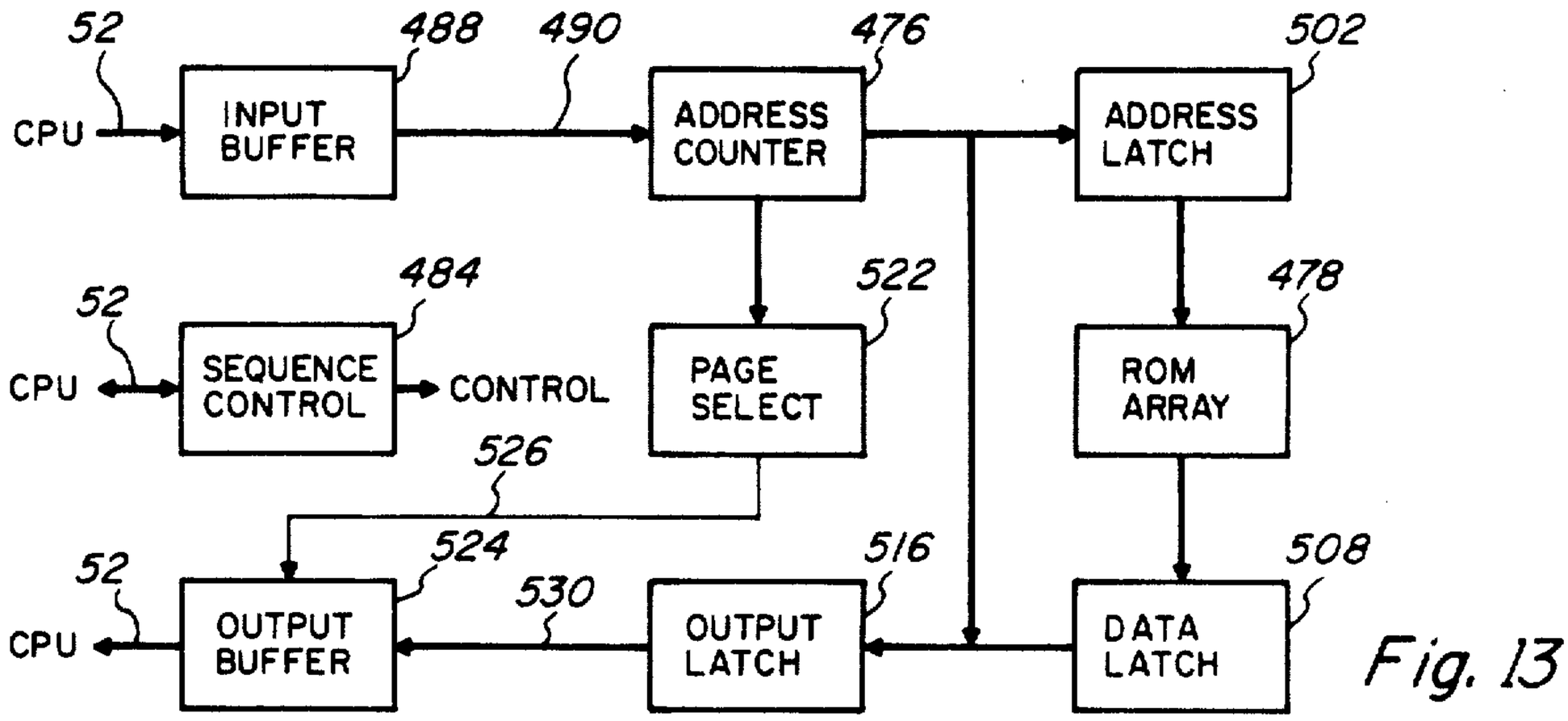

52
CPU
INPUT BUFFER
488
490
ADDRESS COUNTER
476
ADDRESS LATCH
502
484
52
CPU
SEQUENCE CONTROL
CONTROL
PAGE SELECT
522
ROM ARRAY
478
526
524
516
508
52
CPU
OUTPUT BUFFER
530
OUTPUT LATCH
DATA LATCH

Fig. 13

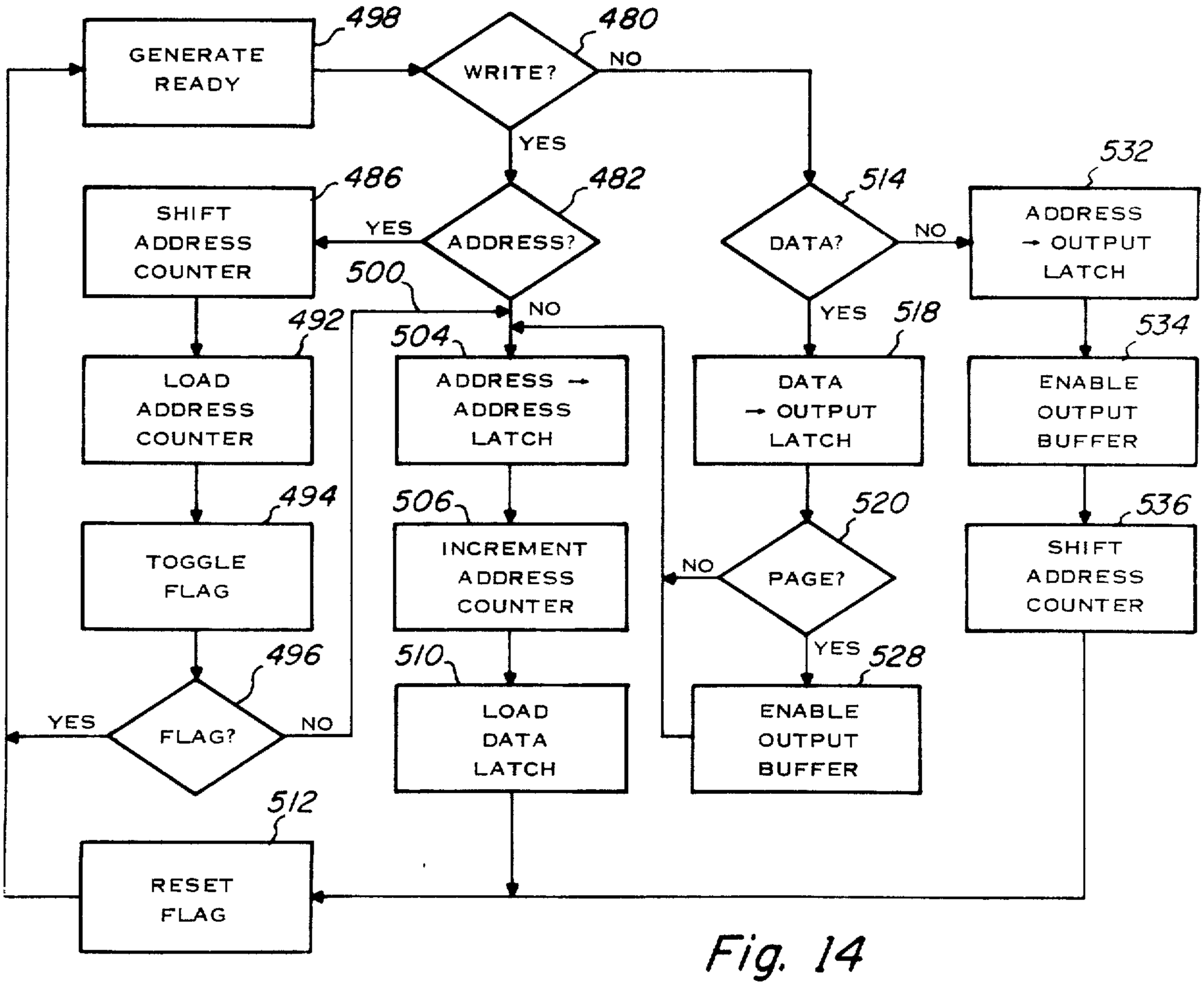

GENERATE READY
498
WRITE?
480
NO
YES
SHIFT ADDRESS COUNTER
486
YES
ADDRESS?
482
500
NO
DATA?
514
NO
ADDRESS → OUTPUT LATCH
532
YES
518
LOAD ADDRESS COUNTER
492
ADDRESS → ADDRESS LATCH
504
DATA → OUTPUT LATCH
ENABLE OUTPUT BUFFER
534
TOGGLE FLAG
494
INCREMENT ADDRESS COUNTER
506
520
NO
PAGE?
SHIFT ADDRESS COUNTER
536
496
510
YES
528
YES
FLAG?
NO
LOAD DATA LATCH
ENABLE OUTPUT BUFFER
512
RESET FLAG

Fig. 14

DIGITAL COMPUTING SYSTEM HAVING AUTO-INCREMENTING MEMORY

Matter enclosed in heavy brackets [ ] appears in the original patent but forms no part of this reissue specification; matter printed in italics indicates the additions made by reissue.

CROSS REFERENCE TO RELATED APPLICATIONS

The subject matter disclosed herein is related to the subject matter disclosed in the following co-pending U.S. Patent applications, assigned to the assignee of the present invention: "Video Display Processor", Ser. No. 18,540, filed Mar. 8, 1979 by David A. Ackley, Gerald D. Rogers, Peter H. Macourek, Karl M. Guttag and Ki Suk Chang, now U.S. Pat. No. 4,243,984; and, "Video Display Processor Having an Integral Composite Video Generator", Ser. No. 17,865, filed Mar. 5, 1979 by Joe F. Sexton.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to memory control systems for use in digital computing systems and, more particularly, but not by way of limitation, to an improved memory control system having an auto-incrementing address counter.

2. Prior Art Statement

In general, digital computing systems are provided with one or more memory subsystems, each of which provides access to a single data storage location of characteristic length in response to receiving an address of appropriate form from the central processing unit. In such systems, the central processing unit must supply additional addressesdynamic RAM are substantially reduced when combined with the relatively higher speed monolithic, microprocessor devices currently available, primarily because of increased processor waiting time.

SUMMARY OF THE INVENTION

In a digital computing system having a central processing unit responsive to processing information provided by a memory generally in response to receiving the address of an address location therein, the present invention provides an improved memory addressing system controller having an address counter for receiving the address provided by the central processing unit and providing that address to the memory, and an address controller for incrementing the address in the address counter to the address of the next sequential address location in sequence in the memory in response to the central processing unit receiving the processing information provided by the memory in response to the address provided by the address counter.

It is an object of the present invention to provide an improved memory addressing system for use in a digital computing system having a central processing unit responsive to processing information provided by a memory.

Another object of the present invention is to provide a memory addressing system having an automatically incremented address counter so that processing information contained at sequential address locations in a sequential memory will be provided thereby in response to each element of the processing information being received by a central processing unit.

Yet another object of the present invention is to provide an improved memory addressing system for efficiently interfacing a relatively slow speed memory device to a relatively higher speed central processing unit.

Still another object of the present invention is to provide an improved memory addressing system for use with memory construction types which are relatively simple and economical to manufacture, yet reliable and versatile in operation.

Other objects and advantages of the present invention will be apparent in the following detailed specification, when read in conjunction with the accompanying drawings which illustrate the preferred embodiments of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a schematic representation of a digital computing system incorporating the preferred embodiment of the present invention.

FIG. **2** is a block diagram of the video display processor shown in FIG. **1**.

FIG. **3** is a block diagram generally illustrating the operation of the overlay control portion of the video display processor.

FIG. **4** is a schematic representation of the register control and control register portions of the video display processor.

FIG. **5** is a logic diagram generally depicting, in conjunction with FIG. **3**, the operation of the overlay control portion of the video display processor.

FIG. **6** is an alternative of a portion of the logic diagram FIG. **5**.

FIG. **7** is an alternative form for one other portion of the logic diagram of FIG. **5**.

FIG. **8** is a logic diagram illustrating the operation of the RAM control portion of the video display processor.

FIG. **9** is a schematic representation of the priority selector portion of the video display processor.

FIG. **10** is a schematic representation of the color phase generator portion of the video display processor.

FIG. **11** is a schematic representation of the color decoder and video mixer portions of the video display processor.

FIG. **12** is a multi-waveform diagram illustrating the operation of the video generator portion of the video display processor.

FIG. **13** is a block diagram of the slow ROM shown in FIG. **1**.

FIG. **14** is a logic diagram illustrating the operation of the slow ROM shown in FIG. **13**.

GENERAL SYSTEM DESCRIPTION

Shown in FIG. **1** is a digital computing system **10** incorporating the preferred embodiment of the present invention. In general, the digital computing system **10** is comprised of a central processing unit (CPU) **12**, a memory subsystem **14**, an input/output subsystem **16**, and a video display subsystem **18**. The CPU **12**, which may be a monolithic microprocessor such as the Texas Instruments 9985, operates in a conventional manner under the control of digital control programs stored in the memory subsystems **14**, usually in response to processing requests initiated via the input/output subsystem **16**. In the input/output subsystem **16**, an I/O control unit **20**, which may be a monolithic integrated cir-

cuit such as the Texas Instruments 9901, operates in a conventional manner to interface a CPU communication bus **22** to an I/O bus **24** connected to one or more I/O units **20**. By way of example, the I/O units **26**, may be such conventional devices as the following: input devices, including a keyboard **26**, a plurality of hand-held units **30**, and various types of remote sensors **32**; output devices, such as speech synthesizer unit **34** and a hard copy printer **36**; and bidirectional input and output devices such as a magnetic disk unit **38**, a magnetic tape unit **40**, and a communication modem **42**.

In the memory subsystem **14**, it is frequently desirable to combine a quantity of read only memory (ROM) with a quantity of read/write, random access memory (RAM). In such configuration, support programs, such as a suitable operating system and a desired assembly or complier, are stored in the ROM, while user programs and volatile data are stored in the RAM. In this form, the relatively static programs and data are maintained in the relatively less expensive ROM, so that only the relatively transient programs and data need be stored in the generally more expensive RAM.

In the preferred form shown in FIG. **1**, the memory subsystem **14** is also configured to take advantage of the low cost of relatively slow ROM and of dynamic RAM, without substantially degrading the performance of the CPU **12**. More particularly, in the ROM portion of the memory subsystem **14**, a relatively limited amount of fast ROM **44**, preferably of the N-channel MOS type, such as the Texas Instruments 4732, is directly connected to the CPU **12** via a CPU memory bus **46**, while a larger amount of relatively slow ROM **48**, preferably of the P-channel MOS type, such as Texas Instruments 0430, is connected to the CPU **12** via a bus buffer **50**, such as the Texas Instruments 74LS245, interposed between the CPU memory bus **46** and an auxiliary bus **52**. By providing each device comprising the slow ROM **48** with an internal auto-incrementing address counter, the CPU overhead associated with sequentially accessing the slow ROM **48** is greatly reduced. If, in addition, each of the devices comprising the slow ROM **48** is assigned a unique ROM address page number, as in the 0430, an additional plurality of such devices may be incorporated to form a ROM library module **54** for connection to the auxiliary bus **52** via a suitable plug-in type port.

In the RAM portion of the memory subsystem **14**, a block of dynamic RAM **56**, preferbly of the N-channel MOS type, such as the Texas Instruments 4027, is connected via a RAM bus **58** to the CPU memory bus **46** via a video display processor (VDP) **60**. More particularly, the VDP **60** is constructed to provide, in addition to other functions to be described below, an auto-incrementing address counter capability, similar to that incorporated in the devices comprising the slow ROM **48**. In addition, the VDP **60** provides for the periodic refreshing of the contents of the various devices comprising the RAM **56**. Thus, the CPU **12** is relieved of the burden of supplying addresses for each of a series of sequential accesses to the RAM **56**, and of the considerable overhead normally associated with the periodic refreshing of dynamic random access memory.

In the video display subsystem **18**, the VDP **60** may be activated by the CPU **12** via the CPU memory bus **46** to generate all video, control and synchronization signals necessary for the display on a raster-scanned television unit of a set of display data previously generated by the CPU **12** and stored in the RAM **56**. The resultant composite video signal is provided via a signal path **62** for application either to a dedicated monitor unit or to a conventional RF modulator **64** before application to a conventional television receiver. In the preferred form, a sound generator **66**, such as the Texas Instruments 9919, is connected to the CPU **12** via the auxiliary bus **52** and provides a CPU-controlled audio signal which may be applied to an auxiliary speaker **68** via a signal path **70** or to the RF modulator **64** via a signal path **72** for mixing with the composite video signal provided by the VDP **60**.

To facilitate system initialization and synchronization, it is preferred that the VDP **60** respond to a manual reset or an external synchronization signal on a signal path **74**, by placing the various control portions thereof in a known state. Similarly, it is considered desirable that the VDP **60** be capable of receiving an externally produced, composite video signal via a signal path **76**, and mixing the external video signal with the internally-generated composite video signal for output via the signal path **62**. For example, it may be desirable in some circumstances to combine the composite video signal generated by the VDP **60** with a composite video signal produced via an auxiliary television camera or derived from a broadcast television signal. In such a configuration, the VDP may be conveniently synchronized with the external video source by extracting in a conventional manner appropriate synchronizing portions of the external video signal on the signal path **76** for application to the VDP **60** via the signal path **74**. As will be readily apparent to those skilled in the art, the external video input and synchronization capability of the VDP **60** also facilitates the chaining of two or more VDP **60** devices, to greatly enhance the data display and animation capabilities of the digital computing system **10**.

GENERAL DESCRIPTION OF THE VIDEO DISPLAY PROCESSOR

Shown in FIG. **2** is a block diagram of the circuit comprising the video display processor **60** shown in FIG. **1**. In general, the VDP **60** is constructed to operate in both a RAM controller mode and in a video controller mode, with substantial simultaneity occuring between these modes. In addition, much of the circuitry for accomplishing the RAM controller functions may be conveniently employed, together with additional circuitry, for accomplishing the video controller functions. In this manner, substantial savings in time and circuitry are realized.

In general, a CPU interface **78** response to access requests from the CPU **12** via the CPU memory bus **46**. When a CPU access request is initially received, the CPU interface **78** transfers the selected RAM address to a register control **80** via a register bus **82** for storage in a particular one of a set of control registers **84**. In the case of a write request, the CPU interface **78** then latches the write data from the CPU memory bus **46** into a CPU data register **86** via a VDP address and data bus **88**, and initiates a CPU write access request for service by a RAM control **90**. In response to the write request, the RAM control **90** will retrieve the RAM address from the control registers **84** via the register control **80**, and pass the RAM address to the RAM **56** via the RAM bus **58**. Thereafter, the RAM control **90** will transfer the write data from the CPU data register **86** to the RAM **56** via the RAM bus **58**. In the case of a read request, the CPU interface **78** will simply initiate a CPU read access request for service by the RAM con-

trol **90**. As in the case of a write request, the RAM control **90** then transfers the RAM address from the control registers **84** to the RAM **56**. Thereafter, the RAM control **90** cooperates with the RAM **56** to latch the read data provided by the RAM **56** via the RAM bus **58** into the CPU data register **86**. When the CPU **12** calls for the data, the CPU interface **78** transfers the read data provided by the CPU data register **80** on the VDP address and data bus **88** to the CPU **12** via the CPU memory bus **46**.

As soon as a write request has been serviced, the RAM control **90** will automatically increment the RAM address contained in the control registers **84**, so that a subsequent CPU write access request can be made into the next sequential address location in the RAM **56** merely by transferring the write data from the CPU **12** into the CPU data register **80** via the CPU interface **78**. Similarly, the RAM control **90** will automatically increment the RAM address contained in the control registers **84** after a read request has been serviced, so that a subsequent CPU read across request can be made from the next sequential address location in the RAM **56** as soon as the CPU interface **78** has completed the transfer of the preceeding read data to the CPU **12**. Thus, the CPU **12** spends a minimum amount of time waiting for a data transfer after an access request is issued.

When VDP register access request is received, the CPU interface **78** transfers the address of the particular one of the set of control registers **84** to the register control **82** via the register bus **82**. In the case of a register write request, the CPU interface **78** transfers the write data from the CPU memory bus **46** to the register bus **82**, for subsequent latching into the selected control register **84** via the register control **80**. In the case of a register read request, the register control **80** connects the selected control register **84** to the register bus **82**, with the CPU interface **78** subsequently connecting the register bus **82** to the CPU memory bus **46**.

When the VDP **60** is operating in the RAM controller mode only, the RAM control **90** operates in a conventional manner to periodically access each of the refresh segments in the RAM **56**. Thus, RAM contents are protected in the event that the CPU **12** fails to exercise each of the refresh segments through normal RAM accesses.

In the video controller mode, the VDP **60** generates a composite video signal in accordance with a set of control parameters established in the control registers **84**, using a set of display data arrays stored in the RAM **56**. In general, the composite video signal, when displayed on a suitable video display unit, produces a video display comprised of M columns of N rows of individual, discrete video display elements or pixels. For convenience of information display, however, the (M×N) pixels may be considered as being logically associated into smaller contiguous groups or blocks which may be configured or defined to form discernible characters or "patterns," as in conventional character generators. In addition, however, the preferred form of the VDP **60** accommodates a plurality of mobile blocks or "sprites" which may be freely moved relative to the fixed display image by defining or selecting a particular column U and row V at which the upper left corner of the sprite is to be displayed. Thus, VDP **60** generates the composite video signal in synchronization with the instantaneous column X and row Y position of the raster scan so as to display either the fixed patterns or the mobile sprites, as appropriate.

In the preferred form, the VDP **60** operates in a CPU-selected one of three distinct video display modes; pattern graphics, multicolor, and text. Briefly, in the pattern graphics mode, the VDP **60** generates a 32 column, 24 row image of patterns (8×8 pixels) selected from a pattern generator table (256 pattern definition blocks) according to a pattern name table (768 pattern names), and, in addition, superimposes up to 32 of the mobile patterns or sprites (8×8 pixels) selected from a sprite generator table (256 sprite definition blocks) according to a sprite name table (32 sprite descriptor blocks) which also defines the displacement of each sprite relative to the pattern image. In the multicolor mode, the VDP **60** generates a 32 column, 6 row image of color patterns (2×8 blocks of 4×4 pixels each) selected from a pattern color table (1536 elements) according to a pattern name table (192 pattern names), with up to 32 of the sprites being generated in substantially the same manner as in the pattern graphics mode. In the text mode, the VDP **60** generates a 40 column, 24 row image of patterns (6×8 pixels) selected from a pattern generator table (256 pattern definition blocks) according to a pattern name table (960 pattern names). In each of the three video display modes, the VDP **60** provides a selection of 16 distinct colors, including white, gray, black and a special transparent state to be described in greater detail below. Since the operation of the VDP **60** in the multicolor and text modes is substantially the same as in the pattern graphics mode, except for the differences noted above, the discussion hereinafter will be directed primarily to the detailed operation in the pattern graphics mode.

During system initialization and as required thereafter, the VDP **60**, operating in the memory controller mode, cooperates with the CPU **12** to establish in the RAM **56** the various display data arrays appropriate for a selected one of the three video display modes. For example, to enable the VDP **60** to operate in the pattern graphics mode, the CPU **12** should store in the RAM **56** the various pattern and sprite tables relied upon by the VDP **60**. In particular, the patern generator table is comprised plurality of consecutive pattern definition blocks, each consisting of 8, 8-bit bytes, which define the bit patterns for each individual pattern, as in conventional character generators. In contrast, the pattern name table consists of a row-by-column ordered array of patterned names which map the pattern definition blocks into each of the 32 columns of 24 rows of patterns comprising a full screen video pattern image. In addition, a pattern color table establishes a pair of video color codes associated with each of 32 contiguous sets of 8 pattern definition blocks of the pattern generator table, with each of the video color codes corresponding to a particular one of the sixteen available colors. Thus, the pattern generator table, and the pattern color table represent an ordered array whereby the individual bits comprising a pattern definition block map the video color codes assigned via the pattern color table into each of the M columns of N rows of pexels comprising a full screen video pattern image. In a similar manner, the sprite generator table is comprised of a plurality of consecutive sprite definition blocks, each consisting of 3, 8-bit bytes, which define particular bit patterns for each of the patterns to be utilized as sprites. The sprite name table, on the other hand, is comprised of 32, 4-byte sprite descriptor blocks which define the particular column displacement U and row displacement V for the display of the particular sprite relative to the video

pattern image, where $1 \leqq U \leqq M$ and $1 \leqq V \leqq N$. In addition, each of the sprite descriptor blocks in the sprite name table contains a sprite name which refers to a particular one of the sprite definition blocks in the sprite generator table, as well as a video color code which establishes the particular one of the sixteen available colors that the active portion of the sprite is to assume. Thus, the sprite name table and the sprite generator table represent an ordered array whereby the individual bits comprising a sprite definition block map the video color code assigned via the sprite descriptor block into the S columns of T rows of pixels comprising a particular video sprite image, where $1 \leqq S \leqq M$ and $1 \leqq T \leqq N$. To promote uniformity of reference, the dimensions of the pattern and sprite image relative to the pattern image are considered herein in terms of individual pixels, since the format of the various tables in the RAM **56** are generally related to the particular number of rows and columns of discrete symbols of characters characteristic of the selected video display mode.

In general, a sequence control **92** operates in a conventional manner to maintain a cyclic column count X and a cyclic row count Y indicative of the time sequential position of the raster scan of the video display unit. As will be clear to those skilled in the art, only a portion of the total raster scan period is devoted to actively displaying patterns on the video display unit, since a portion of each row of horizontal scan is devoted to horizontal retrace, while a number of complete row of horizontal scans are required to perform vertical retrace and related synchronization. However, at least during the active display period, the sequence control **92** makes the column count X and the row count Y available via the VDP address and data bus **88**. The sequence control **92** also provides a color reference signal having a frequency related to the NTSC 3.57/MHz carrier, via a signal path **94**, and a set of sync signals of substantially conventional form via a sync bus **96**. In response to the reset/external sync signal on the signal path **74**, the sequence control **92** clears the column and row counts, and generally synchronizes the color reference signal and the sync signals with the external source. In the preferred form, the sequence control **92** is comprised of a clock circuit of conventional form, and a pair of control programmable logic arrays (PLA's) for providing a various control signals via a control bus **98** depending on the current column and row counts.

An overlay control **100**, responsive to the column and row counts, periodically requests the RAM control **90** to retrieve selected portions of the pattern and sprite tables from the RAM **56**. As the display data is provided by the RAM **56** via the RAM bus **58**, the overlay control **100** receives the pattern data, and provides a first pattern signal via a pattern bus **102**, comprising the bit in the pattern generator table which maps the pixel in the column $(X-U+1)$ of the row $(Y-V+1)$ of the video sprite image when $U \leqq X < (U+S)$ and $V \leqq Y < (V+T)$. In addition, the overlay control **100** receives the video color codes assigned to each pattern and sprite during the display thereof. In other words, the overlay control **100** processes the pattern data arrays so as to provide the proper bit patterns for each of the selected patterns during the entire period that the display is active, but processes the sprite data arrays so as to provide the proper bit patterns for each of the selected sprites only during that portion of the active display period specified for the display thereof.

Each of the first and second pattern signals, and the associated video color codes, are applied to a priority selector **104** via the pattern bus **102**. In response to receiving only the first pattern signal, the priority selector **104** will select a respective one of the video color codes associated with the first pattern signal, depending upon the current digital value thereof. On the other hand, in response to receiving the second pattern signal, whether or not the first pattern signal is also being received, the priority selector **104** will select the video color code associated with the second pattern signal. If neither the first nor second pattern signals is being received, the priority selector **104** will generally select a default video color code provided by one of the control registers **84** via a default color bus **106**. If, as in the preferred form, the overlay control **100** provides a second pattern signal for each of a plurality of active sprites, the priority selector **104** will select the second pattern signal corresponding to the sprite image having the highest priority, according to a predetermined prioritized ordering of the available sprite images. For example, assuming that the overlay control **100** can simultaneously provide a second pattern signal for each of four different sprites representing four out of the priority ordered set of 32 sprites, the priority selector **104** will select the second pattern signal which corresponds to the one of the four sprites having the highest priority. In each case, the video color code corresponding to the current selected pattern signal is provided via a color bus **108** as a video control signal.

A color phase generator **110**, which forms a portion of a composite video generator **112**, receives the color reference signal provided by the sequence control **92** via signal path **94**, and generates the six NTSC color phase signals, each phase shifted by a predetermined amount relative to the color reference signal. In a color decoder **114**, the video color codes, comprising the video control signal provided by priority selector **104** via the color bus **108**, are decoded, and applied to a video mixer **116**, together with the color phase signals provided by the color phase generator **110**. In the video mixer **116**, each of the video color codes decoded via the color decoder **114** selectively couples a complimentary pair of the color phase signals to a gating network (described hereinafter) to generate the information portion of a composite video signal for output via the signal path **62**. In addition, the video mixer **116** receives the sync signals provided by the sequence control **92** via the sync bus **96**, and generates the standard horizontal, vertical and color burst portions of the composite video signal in response thereto. In the preferred form, the video mixer **116** may be placed in an external video mode wherein an external video signal received via the signal path **76** is selectively merged with the internally-generated composite video signal for output via the signal path **62**.

DESCRIPTION OF THE OVERLAY CONTROL

Shown in FIG. **3** is a block diagram generally illustrating the operation of the overlay control **100** (FIG. **2**), generally in accordance with the logic diagram shown in FIG. **5**, using the information stored by the CPU **12** in the control registers **84** shown in FIG. **4**. More particularly, the overlay control **100** is generally responsive to the column and row counts provided by the sequence control **92**. Thus, if the column count X and the row count Y indicate that the raster scan is positioned at the start of one of the horizontal rows in

the active display range, the overlay control **100** will enter a pattern processing procedure **118** (decision block **120**) and request the RAM control **90** to load the pattern name associated with the current column and row counts from the pattern name table into a name latch **122** (processing block **124**). In response to this VDP access request, the RAM control **90** concatenates a pattern name table base address stored in a pattern name table base register **126** (FIG. **4**), the current row count Y, and the current column count X to derive a RAM address for output to the RAM **56**. For example, in the pattern graphics mode, the upper five bits of the row count Y and the upper five bits of the column count X provide access to each of the 788 pattern names.

After the pattern name is latched into the name latch **122**, the overlay control **100** will request the RAM control **90** to load a pair of the video color codes from the pattern color table into a pair of pattern color registers **128** (processing block **130**). In response to this VDP access request, the RAM control **90** concatenates a pattern color table base address stored in a pattern color table base register **132** (FIG. **4**), with a suitable high-order portion of the pattern name to derive a RAM address for output to the RAM **56**. For example, in the preferred form, the upper five bits of the pattern name provide access to a respective one of 32 pairs of video color codes for each consecutive set of eight pattern names in the pattern name table. In the preferred form, one of the video color codes assigned to a particular pattern defines the color of the foreground or information portion of the pattern image, while the other one of the video color codes defines the color of the background or constant portion of the pattern image.

After the pattern video color codes are loaded into the pattern color registers **128**, the overlay control **100** will request the RAM control **90** to load a particular one of the eight bytes or pattern lines from the pattern generator table into a pattern shift register **134** (processing block **136**). In response to this VDP access request, the RAM control **90** concatenates a pattern generator table base address stored in a pattern generator table base register **138** (FIG. **4**), the pattern name stored in the name latch **122**, and a suitable low-order portion of the current row count Y, to derive a RAM address for output to the RAM **56**. For example, in the preferred form, the lower three bits of the row count Y provide access to a particular one of the eight bit pattern bytes comprising the pattern definition block selected via the pattern name.

After loading, the pattern shift register **134** will successively provide each consecutive bit of the pattern line in response to a column control signal applied thereto via the signal path **98a** by the sequence control **92** in synchronization with the columnar movement of the raster scan within the active display range. Thus, the second pattern signal on the signal path **102a** will comprise a time-sequential, digital representation of the full screen, pattern image as the raster scan is traversing the active display range.

After the pattern line is loaded into the pattern shift register **134**, the overlay control **100** will increment, modulo 4, an internal CPU access index (processing block **140**). If the resultant value of the CPU access index is not equal to 3 (decision block **142**), and if a stop flag has not been set (decision block **144**) in the manner described hereinafter, the overlay control **100** will enter a sprite preprocessing procedure **146** and increment a current sprite number maintained in a sprite counter **148** (processing block **150**). Thereafter, the overlay control **100** will request the RAM control **90** to fetch the row displacement V for the current sprite number from the sprite name table (processing block **152**). In response to this VDP access request, the RAM control **90** concatenates a sprite name table base address stored in a sprite name table base register **154** (FIG. **4**), the current sprite number, and an attribute number indicative of the particular byte in the sprite descriptor block which defines the row displacement V, to derive a RAM address for output to the RAM **56**. For example, in the preferred form, the row displacement V is contained in the first byte of the sprite descriptor block for each of the sprites defined in the sprite name table.

In a subtract and compare **156**, the overlay control **100** compares the retrieved row displacement V against a predetermined stop code (decision block **158**) which, if present, indicates that all subsequent entries in the sprite name table are to be ignored or otherwise not processed. Although substantially any value outside the active row count range may be employed, the preferred embodiment utilizes the stop code value **208**, which is outside the active display range 0–192 but within the total row count range of 0–255. Thus, a substantial number of RAM access cycles may be made available for use by the CPU **12** when it is desired to utilize less than the 32 available sprites.

If the row displacement V is not equal to the stop code, the subtract and compare **156** will determine whether the current row count Y is within the desired display range of the current sprite number (decision block **160**). If the current row count Y is within the display range for the current sprite number (see FIG. **6**), the overlay control **100** will stack the current sprite number into a first-in, first-out sprite stack **162** (processing block **164**).

If the row displacement V is equal to the stop code (decision block **158**) or if the sprite stack **162** is full (decision block **166**) after the current sprite number has been entered therein (processing block **164**), the stop flag referred to above is set (processing block **168**). Thereafter, or if either the sprite stack **162** is not full (decision block **166**) after the current sprite number has been has been entered therein (see processing block **104**) or if the current row count Y is not within the display range of the current sprite number (decision block **160**), the overlay control **100** again examines the current column and row counts (decision block **120**).

On the other hand, if the CPU access index has a value of 3 (decision block **142**), or if the stop flag has been set (decision block **144**), the overlay control **100** sets a CPU access flag (processing block **170**), indicating that a RAM access cycle has been dedicated for the use of the CPU **12**, it required. Thereafter, the overlay control **100** again examines the current column and row counts (decision block **120**).

If the column count X and the row count Y indicate that the raster scan is positioned between the end of one horizontal row and the start of the next horizontal row in the active display range, the overlay control **100** will enter a sprite post processing procedure **172** (decision block **120**). If the sprite stack **162** is not empty (decision block **174**), the overlay control **100** will unstack the "top" or first-in sprite number (processing block **176**). The overlay control **100** will then request the RAM control **90** to load the column displacement U for the particular sprite number from the sprite name table into a sprite down counter **178** (processing block **180**). In

response to this VDP access request, the RAM control **90** concatenates the sprite name table base address stored in the sprite name table base register **154** (FIG. **4**), the particular sprite number, and an attribute number indicative of the particular byte in the sprite descriptor block which defines the column displacement U, to derive a RAM address for output to the RAM **56**. For example, in the preferred form, the column displacement U is contained in the second type of the sprite descriptor block for each of the sprites defined in the sprite name table.

After the column displacement U is loaded into the sprite down counter **178**, the overlay control **100** will request the RAM control **90** to load the video color code for the particular sprite number from the sprite name table into a sprite color register **182** (processing block **184**). In response to this VDP access request, the RAM control **90** concatenates the sprite name table base address stored in the sprite name table base register **154** (FIG. **4**), the particular sprite number, and an attribute number indicative of the particular byte in the sprite descriptor block which defines the video color code, to derive a RAM access for output to the RAM **56**. For example, in the preferred form, the video color code is contained in the fourth byte of the sprite descriptor block for each of the sprites defined in the sprite name table.

After the sprite video color code is loaded into the sprite color register **182**, the overlay control **100** will request the RAM control **90** to fetch the row displacement V for the particular sprite number from the sprite name table (processing block **180**). In response to this VDP access request the RAM control **90** concatenates the sprite name table base address stored in the sprite name table base register **154** (FIG. **4**), the particular sprite number, and the attribute number for the particular byte in the sprite descriptor block which defines the row displacement V, to derive a RAM address for output to the RAM **56**.

In the subtract and compare **156**, the overlay control **100** will compute an offset by subtracting the retrieved row displacement V from the current row count Y (processing block **188**). The overlay control **100** will then request the RAM control **90** to load the sprite name for the particular sprite number from the sprite name table into the name latch **122** (processing block **190**). In response to this VDP access request, the RAM control **90** concatenates the sprite name table base address stored in the sprite name table base register **154** (FIG. **4**), the particular sprite number, and an attribute number indicative of the particular byte in the sprite descriptor block which defines the sprite name, to derive a RAM address for output to the RAM **56**. For example, in the preferred form, the sprite name is contained in the third byte of the sprite name table.

After the sprite name has been loaded into the name latch **122**, the overlay control **100** will request the RAM control **90** to load one or more (see FIG. 7) of the bytes or sprite lines from the sprite generator table into a sprite shift register **192** (processing block **194**). In response to this VDP access request, the RAM control **90** concatenates a sprite generator table base address stored in the sprite name table base register **154** (FIG. **4**), the particular sprite number, and the attribute number for the particular byte in the sprite descriptor block which defines the row displacement V, to derive a RAM address for output to the RAM **56**.

After the sprite line is loaded into the sprite shift register **192**, the overlay control **100** will set the CPU access flag (processing block **198**), indicating that a RAM access cycle has been dedicated for the use of the CPU **12**, if required. Thereafter, the overlay control **100** again examines the contents of the sprite stack **162** (decision block **174**).

If the sprite stack **162** is empty (decision block **174**), the overlay control **100** will reset the stop flag (processing block **200**), thereby enabling the sprite preprocessing procedure **146** (see decision block **144**). The overlay control **100** also clears the sprite number contained in the sprite counter **148** (processing block **202**), for subsequent use by the sprite preprocessing procedure **146**. Thereafter, the overlay control **100** again examines the current column and row counts (decision block **120**).

After the raster-scan has reentered the active display range, the sprite downcounter **178** will successively decrement the column displacement contained therein in response to the column control signal provided by the sequence control **92** via the signal path **134**. After decrementing to zero, the sprite downcounter **178** will couple the column control signal to the sprite shift register **192**. In response to the column control signal, the sprite shift register **192** will successively provide each consecutive bit of the sprite line. Thus, the first pattern signal on the signal path **102b** will comprise a time-sequential, digital representation of the particular sprite image only during the portion of the raster scan selected for the display of the sprite.

In the preferred embodiment, the set of sprites defined in the sprite name table may be displayed in a selected one of up to four distinct "sizes". For example, the CPU **12** may reset a MAG bit in a command register **204** (see FIG. **4**) to request the overlay control **100** to map each bit in a sprite definition block into a single display pixel, or set the MAG bit to request the overlay control **100** to map each of the bits in the sprite definition block into a 2×2 block of display pixels. Similarly, the CPU **12** may reset a SIZE bit in the command register **204** to request the overlay control **100** to construct each sprite as an 8×8 pattern of display pixels using 8 consecutive 8-bit bytes as a sprite descriptor block, or set the SIZE bit to request the overlay control **100** to construct each sprite as a 10×16 pattern of display pixels using 32 consecutive 8-bit bytes as a sprite descriptor block. If the CPU **12** sets both the MAG and SIZE bits, the overlay control **100** will construct each sprite a a 16×16 pattern of 2×2 blocks of pixels using 32 of the 8-bit bytes as a sprite descriptor block. In comparison to the standard or default sprite image, the SIZE bit alone quadruples sprite image area with no loss in detail resolution, while the MAG bit alone quadruples sprite image area although with a 4-fold loss in detail resolution. Thus, the effective display range for each sprite will generally be a function of the selected dimensional characteristics (see decision block **160** of FIG. **5**).

For example, in the sprite preprocessing procedure **146**, the overlay control **100** determines whether the current row count Y is within the display range for each of the sprites defined in the sprite name table (see decision block **160**). In making this determination, the overlay control **100** will compute an offset by subtracting the row displacement V for a particular sprite number from the current row count Y (processing block **206** of FIG. **6**). If the computer offset is less than zero (decision block **208**, the row county Y has not yet reached the

specified row displacement V and the overlay control **100** may return to examine the current column and row counts (decision block **120** of FIG. **5**). However, if the computed offset is greater than seven (decision block **210**), the current row county Y is clearly within the display range for the particular sprite number in the sprite stack **162** (see processing block **164** of FIG. **5**).

If the computed offset is greater than 7 (decision block **210**) and neither the SIZE nor the MAG bit is set (decision block **212**), then the current row count Y is beyond the display range of the particular sprite number and the overlay control **100** may return to examine the current column and row counts (decision **120** of FIG. **5**). However, if either the SIZE or the MAG bit is set (decision block **212**), and if the computed offset is not greater than 15 (decision block **214**), then the current row count Y is within the expanded display range of the current sprite number and the overlay control **100** will proceed to stack the current sprite number in the sprite stack **162** (see processing block **164** of FIG. **5**). On the other hand, if the computed offset is greater than 15 (decision block **214**) and if both the SIZE and MAG bits are not set (decision block **216**), but the computed offset is greater than 31 (decision block **218**), the current row count Y is beyond the expanded range of the particular sprite and the overlay control **100** can return to examine the current column and row counts (see decision block **120** of FIG. **5**). Similarly, if both the SIZE and the MAG bits are set (decision block **216**) but the computed offset is greater than 31 (decision block **218**), the current row county Y is beyond the maximum display range of the particular sprite and the overlay control **100** can return to examine to current column and row counts (see decision block **120** of FIG. **5**). Of course, if both the SIZE and the MAG bits are set (decision block **216**), and the computed offset is not greater than 31 (decision block **218**), then the current row count Y is within the maximum display range for the particular sprite and the overlay control **100** will proceed to stack the particular sprite number in the sprite stack **162** (see processing block **164** of FIG. **5**).

In the sprite post processing procedure **172**, if the MAG bit is found to be set (decision block **220** of FIG. **7**) when the sprite shift register **192** is to be loaded (decision block **194** of FIG. **5**), the overlay control **100** will divide the computed offset (see processing block **188** of FIG. **5**) by 2 (processing block **222** of FIG. **7**) before requesting the RAM control **90** to load one of the bytes or sprite lines from the sprite generator table into the sprite shift register **192** (processing block **194** of FIGS. **5** and **7**). Thus, each byte of a particular sprite definition block is accessed for each of two consecutive rows of the sprite image. On the other hand, if the SIZE bit is set (decision block **224** of FIG. **7**), the overlay control **100** will add **16** to the computed offset (processing block **226**) and request the RAM control **90** to load a second byte or sprite pattern line from the upper half of the 32-byte sprite definition block (processing block **228**). Of course, if the MAG bit is also set (decision block **220**), then the computed offset has already been adjusted (processing block **222**) to allow two consecutive accesses to each of the bytes in the upper half of the expanded sprite definition block. Of course, the sprite shift register **192** (see FIG. **3**) is constructed to accomodate up to 16 bits or 2 sprite lines from a sprite definition block. Further, the sprite shift register **192** should be responsive to only every other column control signal coupled thereto via the sprite downcounter **178**, so that each bit of the sprite line will be provided as the first pattern signal during the movement of the faster scan across two column locations.

In summary, the overlay control **100** processes consecutive portions of the pattern arrays during the period that the raster scan traverses each row within the active display range, so that the pattern data for the particular row is available for immediate display. Substantially simultaneously, the overlay control **100** preprocesses the sprite arrays to select those sprites which are to be displayed on the following row. During the intervening horizontal retrace interval, the overlay control **100** processes only those portions of the sprite arrays associated with the selected sprites, so that the sprite data will be available when the raster scan reaches the appropriate column location in the new row. In this manner, the overlay control **100** is able to perform all necessary pattern and sprite processing functions while still allowing the CPU **12** to have periodic access to the RAM **56**.

DESCRIPTION OF THE RAM CONTROL

Shown in FIG. **8** is a logic diagram illustration the general operation of the RAM control **90** shown in FIG. **2**. More particularly, the RAM control **90** is generally responsive to the row count Y provided by the sequence control **92** via the VDP address and data bus **88**. Thus, for example, if the current row count Y indicates that the raster scan is positioned in the active display range (decision block **230** of FIG. **8**) and if the CPU access flag (see processing block **170** of FIG. **5**) is set (decision block **232** of FIG. **8**), the RAM control **90** will reset the CPU access flag (processing block **234**). If a CPU access request has been initiated via the CPU interface **78** (decision block **236**), the RAM control **90** will perform a CPU access procedure **238**. Otherwise, the RAM control **90** will return to examine the current row count Y (decision block **230**).

In the CPU access procedure **238**, the RAM control **90** will transfer the RAM address, initially stored in a CPU address register **240** (FIG. **4**) via the CPU interface **78**, to the RAM **56** via the RAM bus **58** (processing block **244**), the RAM control **90** will place the RAM control **56** in the read state and provide appropriate control signals to latch the data provided by the RAM **56** into the CPU data register **86** (processing block **246**). On the other hand, if the CPU access request is a write, the RAM control **90** will place the RAM **56** in a write state and will transfer the data contained in the CPU data register **86** to the RAM bus **58** for storage in the RAM **56** (processing block **248**). In either case, the RAM control **90** will then automatically increment the RAM address contained in the CPU address register **240** (processing block **250**). Thereafter, the RAM control **90** will again examine the current row count Y (decision block **230**).

If the row count Y indicates that the raster scan is within the active display range (decision block **230**) and the CPU access flag is not set (decison block **232**), but a VDP access request is pending (decision block **252**), the RAM control **90** will construct the appropriate RAM address in the manner describing above, and issue the RAM address, together with appropriate RAM control signals, on the RAM bus **38** (processing block **254**). Thereafter, the RAM control **90** will provide appropriate control signals to latch the data provided by the RAM **56** into the appropriate register (processing block **256**). Thereafter, and if no VDP access request is pend-

ing (decision block **252**), the RAM control **90** again examines the current row count Y (decision block **230**).

When the current row count Y indicates that the raster scan is outside the active display range (decision block **230**), but has not yet reached the end of a frame or screen (decision block **258**), the RAM control **90** will enter a refresh procedure **260**. In the refresh procedure **260**, the RAM control **90** performs the necessary RAM accesses to assure that the contents of the RAM **56** are periodically refreshed. More particularly, the RAM control **90** will issue a refresh address (processing block **262**), using an internal refresh counter, each time the refresh procedure **260** is performed. Thereafter, the RAM control **90** will increment the refresh counter (processing block **264**) by an appropriate amount selected to sequentially address each of the refresh segments of the RAM **56**. The RAM control **90** will then determine if a CPU access request is pending (decision block **236**).

When the current row count Y indicates that the raster scan is outside the active display range (decision block **230**), and has just reached the end of a frame or screen (decision block **258**), the RAM control **90** will cooperate with the CPU interface **78** to interrupt the CPU **12** in an appropriate manner (processing block **266**). Thereafter, the RAM control **90** will perform the refresh procedure **260**.

DESCRIPTION OF THE PRIORITY SELECTOR

Shown in FIG. **9** is a schematic representation of the circuit comprising the priority selector **104** shown in FIG. **2**. The priority selector **104** is comprised primarily of a sprite selector portion **268**, a pattern foreground selector portion **270**, a pattern background selector portion **272**, and a default selector portion **274**. In the sprite selector portion **268**, an AND gate **276** receives the second pattern signal provided by the sprite shift register **192** via the signal path **102**a. The AND gate **276** also receives a display active control signal provided by the sequence control **92** via the signal path **98**a, when the column and row counts indicate that the raster scan is within the active display range. As will be clear to those skilled in the art, the AND gate **276** will provide a sprite select signal in a "high" state for application to a sprite color gate **278** via a signal path **280** only if both the second pattern signal and the display active control signal are in the "high" state. Thus, for example, the sprite select signal will be in the "low" state when the display active control signal on the signal path **98**a is in the "low" state indicative of the raster scan being outside the active display range. Similarly, the sprite select signal will be in the "low" state when the second pattern signal on the signal path **102**a has a digital value of "zero" indicative of an inactive pixel in the corresponding portion of the video sprite image. In response to receiving the sprite select signal in the "high" state, the sprite color gate **278** will transfer the video color code provided by the sprite color register **182** via the signal path **102**b to the color decoder **114** via the color bus **108**.

In the preferred form of the present invention, the sprite selector portion **268** also includes an OR gate **282** which provides an output signal for application to the AND gate **278** via a signal path **284** indicative of the state of the video color code provided by the sprite color register **182** via the signal path **102**b. In particular, the OR gate **282** will provide an output signal in the "high" state via the signal path **284** when the video color code received from the sprite color register **182** via the signal path **102**b has a digital value other than "zero" indicative of an inactive pixel in the corresponding portion of the video sprite image. In response to receiving the sprite select signal in the "high" state, the sprite color gate **278** will transfer the video color code provided by the sprite color register **182** via the signal path **102**b to the color decoder **114** via the color bus **108**.

In the preferred form of the present invention, the sprite selector portion **268** also includes an OR gate **282** which provides an output signal for application to the AND gate **276** via a signal path **284** indicative of the state of the video color code provided by the sprite color register **182** via the signal path **102**b. In particular, the OR gate **282** will provide an output signal in the "high" state via the signal path **284** when the video color code received from the sprite color register **182** via the signal path **102**b has a digital value other than "zero." On the other hand, the OR gate **282** will provide an output signal in the "low" state when the video color code received via the signal path **102**b has a digital value of "zero." In the latter case, the AND gate **276** will provide the sprite select signal in the "low" state and the sprite color gate **278** will not transfer the video color code from the signal **102**b to the color bus **108**. Thus, a sprite video color code having the digital value of "zero" effectively results in a clear or transparent state upon mapping into a particular pixel of the sprite image.

In the pattern foreground selector portion **270**, an AND gate **286** receives the first pattern signal provided by the pattern shift register **134** via the signal path **102**c. The AND gate **286** also receives the display active control signal provided by the sequence control **92** via the signal path **98**a. In addition, the AND gate **286** receives the logical complement of the sprite select signal provided by the AND gate **276** via an inverter **288** interposed between the signal path **280** and a signal path **290**. As will be clear to those skilled in the art, the ANG gate **286** will provide a foreground select signal in a "high" state for application to a foreground color gate **292** via a signal path **294** only if both the first pattern signal and the display active control signal are in the "high" state and the sprite select signal is in the "low" state. Thus, for example, the foreground select signal will be in the "low" state when the display active control signal on the signal path **98**a is in the "low" state indicative of the raster scan being outside the active display range. Similarly, the pattern select signal will be in the "low" state when the first pattern signal on the signal path **102**c has a digital value of "zero" indicative of an inactive pixel in the corresponding portion of the video pattern image. In addition, however, the foreground select signal will be in the "low" state when the sprite select signal on the signal path **280** is in the "high" state indicating that the sprite is active at the particular pixel. In other words, the foreground select portion **270** is overridden or inhibited when the sprite select portion **268** is active, so that the sprite image is effectively "superimposed" on the pattern image. On the othe hand, if the sprite selector portion **268** is inactive but the pattern foreground selector portion **270** is active, the foreground color gate **292** will respond to the foreground select signal in the "high" state by transferring the video color code provided by the foreground portion of the pattern color register **128** via the signal path **102**d to the color decoder **114** via the color but **108**.

As in the sprite selector portion **268**, the preferred form of the pattern foreground selector portion **270** also includes the OR gate **296** which provides an output signal for application to the AND gate **286** via a signal path **298** indicative of the state of the video color code provided by the foreground portion of the pattern color register **128** via the signal path **102**d. In particular, the OR gate **296** will provide an output signal in the "high" state via the signal path **298** when the video color code received from the foreground portion of pattern color register **128** via the signal path **102**d has a digital value other than "zero". In the latter case, the AND gate **286** will provide the foreground select signal in the "low" state and the foreground color gate **292** will not transfer the video color code from the signal path **102**d to the color bus **108**. Thus, a pattern foreground video color code having the digital value of "zero" effectively results in a clear or transparent state upon mapping into a particular pixel of the pattern image.

In the pattern background selector portion **272**, an AND gate **300** receives the logical inverse of the first pattern signal provided by the pattern shift register **102**d via an inverter **302** interposed between the signal path **102**d and a signal path **304**. The AND gate **300** also receives the display active control signal provided by the sequence control **92** via the signal path **98**a. As in the pattern foreground selector portion **270**, the AND gate **300** receives the logical inverse of the sprite select signal provided by the inverter **288** via the signal path **290**. In addition, however, the AND gate **300** receives the logical inverse of the pattern foreground select signal via an inverter **306** interposed between the signal path **294** and a signal path **308**. As will be clear to those skilled in the art, the AND gate **300** will provide a background select signal in a "high" state for application to a background color gate **310** via a signal path **312** only if (1) the display active control signal is in the "high" state, (2) the first pattern signal is in the "low" state, (3) the sprite select signal is in the "low" state, and (4) the foreground select signal is in the "low" state. Thus, for example, the background select signal will be in the "low" state when the display active control signal on the signal path **98**a is in the "low" state indicative of the raster scan being outside the active display range. Similarly, the background select signal will be in the "low" state when the first pattern signal on the signal path **102**d has a digital value of "one" indicative of an active pixel in the corresponding portion of the video pattern image. In addition however, the background select signal will be in the "low" state when either the sprite select signal on the signal path **280** or the foreground select signal on the signal path **294** is in the "high" state. In other words, the background select signal will be in the "high" state only when the sprite select portion **268** and the pattern foreground select portion **270** are both inactive but the raster scan is in the active display range. In response to receiving the background select signal in the "high" state, the background color gate **310** will transfer the video color code provided by the background portion of the pattern color register **128** via the signal path **102**d to the color decoder **114** via the color bus **108**.

In the preferred form, the background selector portion **272** also includes an OR gate **314** which provides an output signal for application to the AND gate **300** via a signal path **316** indicative of the state of the video color code provided by the background portion of the pattern color register **128** via the signal path **102**d. In particular, the OR gate **314** will provide an output signal in the "high" state via the signal path **316** when the video color code received from the background portion of the pattern color register **128** via the signal path **102**d has a digital value other than "zero." On the other hand, the OR gate **314** will provide an output signal in the "low" state when the video color code received via the signal path **102**d has a digital value of "zero." In the latter case, the AND gate **300** will provide the background select signal in the "low" state and the background color gate **310** will not transfer the video color code from the signal path **102**d to the color bus **108**. Thus, a pattern background video color code having the digital value of "zero" effectively results in a clear or transparent state upon mapping into a particular pixel of the pattern image.

In the default selector portion **274**, an AND gate **318** receives the display active control signal provided by the sequence control **92** via the signal path **98**a. As in the pattern background selector portion **272**, the AND gate **318** receives the logical inverse of the sprite select signal provided by the inverter **288** via the signal path **290**, and the logical inverse of the foreground select signal provided by the inverter **306** via the signal path **308**. In addition, however, the AND gate **318** also receives the logical inverse of the background select signal via an inverter **320** interposed between the signal path **312** and a signal path **322**. As will be clear to those skilled in the art, the AND gate **318** will provide a default select signal in a "high" state for application to a default color gate **324** via a signal path **326** only if (1) the display active control signal is in the "high" state, (2) the sprite select signal is in the "low" state, (3) the foreground select signal is in the "low" state, and (4) the background select signal is in the "low" state. Thus, for example, the default select signal will be in the "low" state when the display active control signal on the signal path **98**a is in the "low" state indicative of the raster scan being outside the active display range. Similarly, the sprite select signal will be in the "low" state when any one of the sprite select, foreground select, or background select signals on the signal paths **280**, **294**, and **312**, respectively, has a digital value of "zero" indicative of inactive pixels in each of the corresponding portions of the sprite or pattern images. In other words, the default select signal will be in the "high" state only when the sprite selector portion **268**, the pattern foreground selector portion **270**, and the pattern background selector portion **272** are each inactive but the raster scan is in the active display range. In response to receiving the default select signal in the "high" state, the default color gate **324** will transfer the video color code stored in a default color register **328** (see FIG. **4**) and provided via the default color bus **106** to the color decoder **114** via the color bus **108**.

In summary, the priority selector **104** is responsive to each of the first and second pattern signals provided by the overlay control **100** when the display active control signal provided by the sequence control **92** indicates that the raster scan is in the active display range. In particular, the priority selector **104** will transfer a non-transparent sprite video color code from the sprite color register **182** to the color decoder **114** when the second pattern signal indicates that the sprite is active at the current pixel. On the other hand, when the second pattern signal indicates that the sprite is inactive but the first pattern signal indicates that the pattern is active at the current pixel, the priority selector **104** will transfer

a non-transparent foreground video color code from the foreground portion of the pattern color register **128** to the color decoder **114**. If the first and second pattern signals indicate that both the sprite and pattern are inactive at the current sprite, the priority selector **104** will transfer a non-transparent background video color code from the background portion of the pattern color register **128** to the color decoder **114**. If no other non-transparent video color code is selected for display, as in a border area, the priority selector **104** will transfer a default video color code from the default color register **328** to the color decoder **114**.

DESCRIPTION OF THE COLOR PHASE GENERATOR

Shown in FIG. **10** is a schematic representation of the circuit comprising the color phase generator **110** shown in FIG. **2**. In general, the color phase generator **110** is responsive to the color reference signal provided by the sequence control **92** via the signal path **94** (see FIG. **2**). In the preferred form, the sequence control **92** provides the color reference signal as a pair of complementary clock signals, ϕ**1** and ϕ**3**, having a frequency of 10.738635 MHz or three times the NTSC 3.57 MHz color carrier (see FIG. **12**). In response to the color reference signal, the color phase generator **110** provides six color phase signals having the NTSC 3.57 MHz color carrier frequency, but shifted in phase by a predetermined number of degrees to approximate the six NTSC standard color reference signals for the colors yellow, red, magenta, blue, cyan, and green.

In the preferred form, the color phase generator **110** comprises a 3-stage ring counter, with each stage providing interfaced complementary outputs. In particular, the color phase generator **110** is comprised of a first stage **330**, a second stage **332**, a third stage **334**, and a feedback network **336**. In the first stage **330**, an inverter **338** has the input thereof connected to the output of the feedback network **336** via a gate transistor **340** in phase with the ϕ**1** clock signal connected to the gate thereof via the signal path **94**a. The inverter **338** has the output thereof connected to the input of an inverter **342** via a gate transistor **344** in phase with the ϕ**3** clock signal connected to the gate thereof via the signal path **94**b. In the second stage **332**, an inverter **346** has the input thereof connected to the output of the inverter **342** of the first stage **330** in a gate transistor **348** in phase with the 01 clock signal connected to the gate thereof via the signal path **94**a. The inverter **346** has the output thereof connected to the input of an inverter **350** via a gate transistor **352** in phase with the 03 clock signal connected to the gate thereof via the signal path **94**b. In the third stage **334**, an inverter **354** has the input thereof connected to the output of the inverter **350** of the second stage **332** via a gate transistor **356** in phase with the 01 clock signal connected to the gate thereof via the signal path **94**a. The inverter **354** has the output thereof connected to the input of an inverter **358** via a gate transistor **360** in phase with the 03 clock signal connected to the gate thereof via the signal path **94**b. In the feedback network **336**, a NOR gate **362** has one input thereof connected to the output of the inverter **342** of the first stage **330**, one other input thereof connected to the output thereof connected to the input of the inverter **338** of the first stage **330** via the gate transistor **340**.

As will be clear to those skilled in the art, the color phase generator **110** is constructed so that one and only one of the inverters **338**, **346**, and **354** will provide an output signal in a "low" state during each cycle of the 01 clock signal. Similarly, one and only one of the inverters **342**, **350** and **358** will provide an output signal in a "high" state during each cycle of the ϕ**3** clock signal. Thus, by inverting the output of the inverters **342**, **350** and **358** via inverters **364**, **366** and **368**, respectively, a set of six color reference signals is obtained wherein two and only two of the color reference signals are in the "low" state during each half cycle of the ϕ**1** and ϕ**3** clock signals. For convenience of reference, the outputs of the inverters **338**, **364**, **346**, **366**, **354** and **368** have been designated in FIGS. **10** and **11** by an appropriate one of the six NTSC standard colors, i.e. yellow, red, magenta, blue, cyan, and green.

DESCRIPTION OF THE COLOR DECODER AND VIDEO MIXER

Shown in FIG. **11** is a schematic representation of the color decoder **114** and the video mixer **116** shown in FIG. **2**. As appropriate, reference will be made to the waveform diagrams shown in FIG. **12** to illustrate the operation of the color decoder **114** and the video mixer **116**.

In general, the color decoder **114** receives a color select portion of the video color code provided by the priority selector **104** via the color bus **108**. In the preferred form, the color select portion of the video color code is comprised of three color select bits. In response to each unique combination of the three color select bits, the color decoder **114** provides an output signal in a "high" state via a particular color select line **370**. For example, as in a conventional 3-to-8-line decoder, the color decoder **114** will provide an output signal in the "high" state via a color select line **370**a response to receiving a color select bit pattern of "011". Similarly, the color decoder **114** will provide an output signal in the "high" state via a color select line **370**b in response to receiving a color select bit pattern of "111". In response to receiving a color select bit pattern of "000", the color decoder **114** will provide an output signal in the "high" state via a color select line **370**c.

In general, the video mixer **116** operates in a color generation mode, a sync generation mode, or an external video mode, depending upon the state of the sync signals provided by the sequence control **92** via the signal path **96**. In the color generation mode, a gating network **372** selectively couples reference voltages provided by a voltage divider **374** to the gate of a mixer transistor **376**, generally in phase with a complementary pair of the color reference signals provided by the color phase generator **110**. In the preferred form, a digital value of "011" in the color select portion of the video color code represents the colors cyan or red, depending upon the digital value of an intensity portion of the video color code. Thus, for example, in response to receiving a signal in the "high" state on the color select line **370**a, an AND gate **378** will simultaneously connect an upper cyan reference voltage at a tap point **380** of the voltage divider **374** to a high intensity transistor **382** via a gate transistor **384**, and a lower cyan reference voltage at a tap point **386** of the voltage divider **374** to a low intensity transistor **388** via a gate transistor **390**, in phase with the cyan color reference signal provided by the inverter **354** of the color phase generator **110**. In a similar manner, an AND gate **392** will simultaneously connect an upper red reference voltage at a tap point **394** of the voltage divider **374** to the low intensity transistor **388** via a gate transistor **396**, and a lower red reference

voltage at a tap point **398** of the voltage divider **374** to the high intensity transistor **382** via a gate transistor **400**, but in phase with the red color reference signal provided by the inverter **364** of the color phase generator **110**.

In the preferred form, the high intensity transistor **382** is controlled by an intensity bit portion of the video color code provided by the priority selector **104** via the color bus **108**. On the other hand, the low intensity transistor **388** is controlled by the logical inverse of the intensity bit via an inverter **402**. Thus, the upper cyan reference voltage and the lower red reference voltage will be connected to the gate of the mixer transistor **376** in an alternating manner generally in phase with the cyan and red reference signals, respectively, when the intensity bit of the video color code on the color bus **108** is in the "high" state. In contrast, the lower cyan reference voltage and the upper red reference voltage will be connected to the gate of the mixer transistor **376** in an alternating manner in phase with the cyan and red color reference signals, respectively, when the intensity bit of the video control code on the color bus **108** is in the "low" state.

In the preferred form, the upper cyan reference voltage and the lower red reference voltage are selected to have a potential difference **404** proportional to the chrominance value characteristic of the color cyan and an average potential proportional to a medium luminance value, so that the signal applied to the gate of the mixer transistor **376** via the high intensity transistor **382** will provide a composite video signal on the signal path **62** which digitally approximates the standard video waveform for the color cyan as at **406** in FIG. **12**. In a similar manner, the lower cyan reference voltage and the upper red reference voltage are selected to have a potential difference **408** proportional to the chrominance value characteristic of the color red and an average potential proportional to a relatively low luminance value, so that the signal applied to the gate of the mixer transistor **376** via the low intensity transistor **388** will produce a composite video signal on the signal path **62** which digitally approximates the standard video waveform for the color dark red, as at **410** in FIG. **12**. Similar configurations of AND gates and gate transistors are provided for each of the color select lines **370** associated with the video color codes having a color select portion other than "000" and "111".

As will be clear to those skilled in the art, the video waveforms corresponding to the colors white and gray have a fixed luminance value and no chrominance value. In the preferred form, a digital value of "111" in the color select portion of the video color code represents the colors white or gray, depending upon the digital value of the intensity portion of the video color code. Accordingly, the output signal provided by the color decoder **114** via the color select line **370b** is employed to simultaneously connect a white reference voltage at a tap point **412** of the voltage divider **374** to the high intensity transistor **382** via a gate transistor **414**, and a gray reference voltage at a tap point **416** of the voltage divider **374** to the low intensity transistor **388** via a gate transistor **418**. If the white reference voltage is selected to be proportional to the luminance value of the color white, the signal applied to the gate of the mixer transistor **376** via the high intensity transistor **382** will produce a composite video signal on the signal path **62** which digitally approximates the video waveform for the color white, as at **420** in FIG. **12**. Similarly, if the gray reference voltage is selected to be proportional to the luminance value for the color gray, the signal applied to the gate of the mixer transistor **376** via the low intensity transistor **388** will produce a composite video signal on the signal path **62** which digitally approximates the video waveform for the color gray (not shown in FIG. **12**).

As in the case of the colors white and gray, the color black has a fixed luminance value, and no chrominance value. In the preferred form, a digital value of "000" in the color select portion of the video color code represents the color black or the transparent state, depending upon the digital value of the intensity bit portion of the video color code. Accordingly, the output signal provided by the color decoder **114** via the color select line **370c** is employed to connect a black reference voltage at a tap point **442** of the voltage divider **374** to the high intensity transistor **382** via a gate transistor **424**. Thus, if the intensity bit is in the "high" state, the signal applied to the gate of the mixer transistor **376** via the high intensity transistor **382** will produce a composite video signal via the signal path **62** which digitally approximates the video waveform for the color black, as at **426** in FIG. **12**. On the other hand, if the intensity bit is in the "low" state, an AND gate **428** will connect the black reference voltage gated via the gate transistor **424** to the gate of the mixer transistor **376** via a gate transistor **430**, and the composite video signal on the signal path **62** will digitally approximate the video waveform for the color black, as at **432** in FIG. **12**.

In the sync generation mode, the gating network **372** selectively couples reference voltages provided by the voltage divider **374** to the gate of the mixer transistor **376**, generally in response to the sync signals provided by the sequence control **92** via the signal path **96**. For example, in response to receiving a sync signal in the "high" state via a signal path **96a**, an AND gate **434** will connect a sync reference voltage at a tap point **436** of the voltage divider **374** to the gate of the mixer transistor **376** via a gate transistor **438**. If the sync reference voltage is selected to be proportional to the standard sync value, the signal applied to the gate of the mixer transistor **374** will produce a composite video signal on the signal path **62** which digitally approximates the video waveform for a horizontal sync pulse, as at **440** in FIG. **12**.

In response to receiving a burst signal in the "high" state on the signal path **96b**, an AND gate **442** will connect an upper burst reference voltage at a tap point **444** of the voltage divider **374** to the gate of the mixer transistor **376** via a gate transistor **446**, in phase with the yellow color reference signal provided by the inverter **338** of the color phase generator **110**. In a similar manner, an AND gate **448** will connect a lower burst reference voltage at a tap point **450** of the voltage divider **374** to the gate of the mixer transistor **376** via a gate transistor **452**, but in phase with the blue color reference signal provided by the inverter **366** of the color phase generator **110**. Thus, the upper burst reference voltage and the lower burst reference voltage will be connected to the gate of the mixer transistor **376** in an alternating manner generally in phase with the yellow and blue color reference signals. By selecting the upper and lower burst reference voltages to have a potential difference **454** proportional to he peak-to-peak value characteristic of the NTSC color burst and an average potential proportional to the standard blanking level, the signal applied to the gate of the mixer transistor **376** will

produce a composite video signal on the signal path **62** which digitally approximates the standard video waveform for the color burst, as at **456** in FIG. **12**.

In response to receiving a blanking signal in the "high" stage via a signal path **96**c, an inverter **458** will provide an output signal in the "low" state for application to a NOR gate **460**. The NOR gate **460** also receives the sync and burst signals provided by the sequence control **92** via the signal paths **96**a and **96**b, respectively. As will be clear to those skilled in the art, the NOR gate **460** will provide an output signal in the "high" state only when the blanking signal is in the "high" state and both the sync and burst signals are simultaneously in the "low" state. Thus the NOR gate **460** will connect the black reference volage at the tap point **422** of the voltage divider **374** to the gate of the mixer transistor **376** via a gate transistor **462** only during those portions of the normal blanking interval not dedicated to the horizontal sync pulse and the color burst. Since the black reference voltage has been selected to have potential proportional to the standard blanking level, the signal applied to the gate of the mixer transistor **376** via the gate transistor **462** will produce a composite video signal on the signal path **62** which digitally approximates the standard video waveform for the blanking interval, as at **464** in FIG. **12**. The blanking signal on the signal path **96**c is also applied to the color decoder **114**, so that the output signal on each of the color select lines **370** is in the "low" state during the blanking interval.

In the external video mode, an inverter **466** will provide an output signal in the "low" state for application to each of the AND gates **434**, **442** and **448**, in response to receiving an external video enable signal provided by the sequence control **92** via a signal path **96**d. The external video enable signal on the signal path **96**d is also applied to the NOR gate **460**. As a result, the output signals provided by the AND gates **434**, **442** and **448**, and by the NOR gate **460** remain in the "low" state during the entire blanking interval, thereby inhibiting the generation of the composite video signal in this interval. Instead, an AND gate **468**, responsive to the external video enable signal on the signal path **96**d and the blanking signal on the signal path **96**c via an OR gate **470**, connects an external video signal received via the signal path **76** to the gate of the mixer transistor **376** via a gate transistor **472**. Assuming that the sequence control **92** has been synchronized in a conventional manner with the external source of the external video signal, the resultant composite video signal on the signal path **62** will have blanking, sync and burst levels proportional to those contained in the external video signal.

In addition to substituting the synchronizing portions of the external video signal for the internally generated values, the preferred form of the video mixer **116** also passes the information portion of the external video signal when the video color provided by the priority selector **104** corresponds to the transparent state. In particular, the output signal provided by the inverter **466** is also connected to the AND gate **428** to maintain the output signal provided thereby in the "low" state when the external video enable signal on the signal path **96**d is in the "high" state. Thus, the cooperative gating of the black reference voltage via the gate transistors **424** and **430** is suppressed. Instead, an AND gate **474** responsive to the color select line **370**c and the logic inverse of the intensity bit provided by the inverter **402**, will provide an output signal in the "high" state via the OR gate **470** to enable the AND gate **468** when the external video enable signal is also in the "high" state. As a result, the AND gate **468** will connect the external video signal on the signal path **76** to the gate of the mixer transistor **376** via the gate transistor **472**. In other words, a video color code corresponding to the transparent state will result in a composite video signal on the signal path **62** with a digital waveform approximating the color black (as at **426** in FIG. **12**) when the video mixer **116** is not in the external video mode, but in a composite video signal substantially the same as the external video signal on the signal path **76** when the video mixer **116** is in the external video mode.

DESCRIPTION OF THE ROM

Shown in FIG. **13** is a block diagram illustrating the operation of each discrete device comprising the slow ROM **48** (FIG. **1**), generally in accordance with the logic diagram shown in FIG. **14**. In general, the ROM **48** is responsive to ROM access requests provided by the CPU **12** via the memory bus **46**, and coupled to the auxiliary bus **52** via the bus buffer **50**. In particular, the CPU **12** may write a new address into an address counter **476** in the ROM **48**, read the address currently in the address counter **476**, or read the data contained in a ROM array **478** at the address contained in the address counter **476**. In the preferred form, the ROM array **478** contains 6144 8-bit bytes of processing information, each of which is sequentially or randomly addressable via the lower 13 bits of a 16 bit address. The upper 3 bits of the 16 bit address comprise a page designation which specifies a desired one of eight individual devices comprising the ROM **48**, in the manner set forth below.

In response to receiving a write (decision block **480**) address (decision block **482**) ROM access request from the CPU **12** generally via the auxiliary bus **52**, a sequence control **484** prepares to receive the first 8 of the 16 bits comprising the new address by shifting the address bits contained in the lower 8 bit positions of the address counter **476** into the upper 8 bit positions thereof (processing block **486**). When the first 8 address bits of the new address become available on the auxiliary bus **52**, the sequence control **484** enables an input buffer **488** and loads the first 8 address bits into the lower 8 bit positions of the address counter **476** via an input bus **490** (processing block **492**). To "remember" that the first 8 bits of the new address have already been loaded, the sequence control **484** toggles an internal flag (processing block **494**). If, as a result, the flag is in the set state (decision block **496**), the sequence control **484** will generate a ready signal (processing block **498**) for application to the CPU **12** via the auxiliary bus **52**, indicating that the ROM **48** is ready to receive the second 8 bits of the address.

Upon receiving a second write (decision block **480**) address (decision block **482**) ROM access request, the sequence control **484** will shift the first 8 bits of the new address from the lower 8 bit positions of the address counter **476** into the upper 8 bit positions thereof (processing block **486**). When the second 8 bits of the new address are provided by the CPU **12** via the auxiliary bus **52**, the sequence control **484** will enable the input buffer **488** and load the second 8 bits of the new address into the lower 8 bit positions in the address counter **476** via the input bus **490** (processing block **492**). If, after the flag has been toggled a second time (processing block **494**), the flag is in the reset state (decision block **496**),

the sequence control **484** will perform an auto-incrementing procedure **500**.

In the auto-incrementing procedure **500**, the sequence control **484** will load the address currently contained in the address counter **476** into an address latch **502** (processing block **504**). The sequence control **484** will then increment the address contained in the address counter **476** (processing block **506**). Using the address contained in the address latch **502**, the sequence control **484** then transfers the processing information contained in the ROM array **478** at the particular address location into a data latch **508** (processing block **510**). Upon completion of the auto-incrementing procedure **500**, the sequence control **484** will make sure that the flag is reset (processing block **512**) and then generate the ready signal (processing Block **498**) to indicate to the CPU **12** that the ROM **48** is ready to receive the next ROM access request from the CPU **12**.

If the subsequent ROM access request is a read (decision block **480**) data (decision block **514**) command, the sequence control **484** will transfer the processing information stored in the data latch **508** into an output latch **512** (processing block **518**). If the page designation portion of the address contained in the address counter **476** corresponds to the unique page number assigned to the particular device at the time of manufacturing (decision block **520**), a page select **522** will enable an output buffer **524** via a signal path **526** (processing block **528**), to couple the processing information provided by the output latch **516** via an output bus **530** to the auxiliary bus **52**. Therafter, or if the page number does not correspond (decision block **520**), the sequence control **484** will perform the auto-incrementing procedure **500**, described above, make sure that the flag is reset (processing block **512**), and generate the ready signal (processing block **498**) to indicate that the requested data is available on the auxiliary bus **52**.

In response to receiving a read (decision block **480**) address (decision block **514**) ROM access request, the sequence control **484** will transfer the 8 address bits contained in the upper 8 bit positions of the address counter **476** to the output latch **516** (processing block **532**). The sequence control **484** will then enable the output buffer **524** (processing block **534**) to couple the upper address byte provided by the output latch **516** via the output bus **530** to the auxiliary bus **52**. The sequence control **484** then shifts the 8 address bits contained in the lower 8 bit positions of the address counter **476** into the upper 8 bit positions thereof (processing block **536**). Thereafter, the sequence control **484** will make sure that the flag is reset (processing block **512**), and will generate the ready signal (processing block **498**) to indicate to the CPU **12** that the upper byte of the address is available on the auxiliary bus **52**.

Upon receiving a subsequent read (decision block **480**) address (decision block **514**) command, the sequence control **484** will transfer the lower byte of the address, now in the upper 8 bit positions of the address counter **476**, to the output latch **516** (processing block **532**), and enable the output buffer **524** (processing block **534**) to couple the lower address byte to the auxiliary bus **52**. As before, the sequence control **484** will then shift the 8 bits contained in the lower 8 bit positions of the address counter **476** into the upper 8 bit positions thereof (processing block **536**), make sure that the flag is reset (processing block **512**), and generate the ready signal (processing block **498**) to indicate to the CPU **12** that the lower address byte is available on the auxiliary bus **52**.

In response to receiving a write (decision block **480**) data (decision block **482**) ROM access request, the sequence control **484** will simply perform the auto-incrementing procedure **500**, before resetting the flag (processing block **512**) and generating the ready signal (processing block **498**) to indicate completion of the command. Thus, the write data command is a convenient method for resetting the flag, while accomplishing an auto-incrementing operation.

In the preferred mode of operation, the CPU **12** initially issues a write data command, to reset the flag. The CPU **12** then provides a selected starting address via two consecutive write address commands. Thereafter, the ROM **48** will automatically provide the processing information contained at sequentially higher address locations in response to each subsequent read data command issued by the CPU **12**. As part of the auto-incrementing procedure **500** performed in response to each read data command, the ROM **48** loads the data latch **508** with the next sequential byte so that it will be available for rapid transfer to the CPU **12**. Thus, the CPU **12** spends a minimal amount of time waiting for the data after a read data command is issued.

Although specific embodiments of the preferred form of the present invention have been described herein, variations may be made in the construction, arrangement or operation of the parts of elements of the various embodiments as disclosed herein without departing from the spirit and scope of the invention as defined in the following claims.

What is claimed is:

1. A plug-in library module for use in a digital computing system having a central processing unit which provides the address of a selected one of a plurality of sequentially accessed address location in said library module, provides a read memory signal and is responsive to processing information provided thereto in response to said address, and interface means including a plug-in type port for temporarily receiving the library module and for providing communication between the central processing unit and said library module, said library module comprising:

memory means disposed within said library module, said memory means having processing information contained therein at sequentially accessed address locations, and outputting the processing information at each of said address locations in response to receiving the address thereof;

address counter means disposed within said library module for receiving and storing the address provided by said central processing unit, and providing said address stored therein to said memory means in response to the central processing means providing the read memory signal;

connector means disposed within said library module and connectable to the plug-in port of said interface means, said connector means connecting the address provided by said central processing unit to said address counter means, and connecting the processing information outputted by said memory means to said central processing unit; and

control means disposed within said library module and connected to said address counter means, said control means incrementing the address in said address counter means to the address of the next sequentially accessed address location in said mem-

ory means in response to said memory means outputting the processing information to said central processing unit in response to the address provided thereto by said address counter means, whereby the processing information contained in said next sequentially accessed address location will be provided by said memory means in response to a read memory [sianal] *signal* provided by said central processing unit.

2. The library module of claim **1** for use in conjunction with said central processing unit having a first data transfer speed, wherein the memory means comprises a P-channel MOS type device having a second data transfer speed slower than the first data transfer speed.

3. The library module claimed in claim **1**, further comprising:

a latch means disposed within said library module, interposed between said memory means and said connector means, responsive to an enable signal, for receiving and storing processing information outputted by said memory means, and for providing said processing information to said connecting means in response to said enable signal; and

a page select means disposed within said library module, having a unique predetermined page number associated therewith, responsive to the address stored in said address counter means, for applying said enable signal to said latch means when a predetermined page designation portion of said address stored in said address counter means corresponds to said unique predetermined page number.

4. A digital computing system comprising:

a plurality of memory means, each memory means having a unique predetermined page number associated therewith, each memory having processing information contained therein at sequentially accessed address locations, said memory means outputting the processing information at each of said address locations in response to receiving the address thereof;

a central processing unit, responsive to said processing information, the central processing unit providing the address of a selected one of said sequentially accessed address locations and providing at least one read memory signal;

a plurality of address counter means, each corresponding to one of the memory means, interposed between the central processing unit and the corresponding memory means, the address counter means each receiving and storing the address provided by the central processing unit, and providing said address stored therein to the corresponding memory means in response to the central processing unit providing the read memory signal;

a plurality of control means, each corresponding to one of the address counter means, for incrementing the address in the corresponding address counter means to the address of the next sequentially accessed location in the corresponding memory means in response to the corresponding memory means outputting the processing information in response to the address provided thereto by the corresponding address counter means, whereby the processing information contained at said next sequentially accessed address location will be outputted by the corresponding memory means in response to a read memory signal provided by the central processing unit;

a plurality of latch means, each interposed between a corresponding memory means and the central processing unit and responsive to the corresponding control means, each latch means further responsive to an enable signal, for receiving and storing the processing information outputted by the memory means and for providing said processing information to the central processing unit in response to the enable signal; and

a plurality of page select means, each page select means corresponding to one of the memory means and responsive to the address stored in the corresponding address counter means for applying said enable signal to the corresponding latch means when a predetermined page designation portion of the address stored in the corresponding address counter means corresponds to the unique predetermined page number associated with the corresponding memory means.

5. The digital computing system as claimed in claim **4**, wherein at least one of said plurality of memory means is of the read/write type.

6. A digital computing system comprising:

memory means of the dynamic random access type having processing information contained therein at sequentially accessed address locations, said memory means outputting the processing information at each of said address locations in response to receiving the address thereof;

a central processing unit, responsive to said processing information, the central processing unit providing the address of a selected one of said sequentially accessed address locations and providing at least one read memory signal;

an address counter means interposed between the central processing unit and the memory means, the address counter means receiving and storing the address provided by the central processing unit and providing said address in response to the central processing unit providing the read memory signal to the memory means;

a control means, cooperating with the address counter means, for incrementing the address in the address counter means to the address of the next sequentially accessed address location in the memory means in response to the memory means outputting the processing information to the central processing means in response to the address provided thereto by the address counter means, whereby the processing information contained at said next sequentially accessed address location will be provided by said memory means in response to a subsequent read memory signal provided by the central processing unit; and

a memory refresh means for periodically refreshing the processing information stored in the memory means.

7. The data processing system as claimed in claim **6**, wherein the memory refresh means comprises:

a refresh address counter means, having the address of one of the sequentially accessed address locations of the memory means stored therein, for periodically refreshing the processing information stored in the address location corresponding to the address stored therein; and

a refresh address counter incrementing means for incrementing the refresh address counter for each

time the refresh address counter means refreshes processing information in the memory means.

8. The digital computing system of claim **6**, further comprising:

latch means, interposed between the memory means and the central processing unit and responsive to the control means, for receiving and storing the processing information provided by the memory means and for providing said processing information to the central processing unit in response to the control means incrementing the address counter means.

9. The digital computing system of claim **8**, wherein:

the memory means is of the read/write type;

the central processing unit including means for providing processing information for storing in the memory means and providing a memory write signal;

the latch means further includes means responsive to the central processing unit for receiving and storing the processing information provided by the central processing unit; and

the control means further includes means responsive to the memory write signal provided by the central processing unit for causing the processing information stored in the latch means to be stored in the memory address in the address counter means.

10. The digital computing system of claim **9**, wherein:

the central processing unit further includes means for providing a read address signal;

the latch means further includes means reponsive to the address counter means for receiving and storing the address stored in the address counter means in response to the central processing unit providing the read address signal; and

the control means further includes means responsive to the read address signal provided by the central processing unit for causing the latch means to provide the address stored therein to the central processing unit.

11. A plug-in library module for use in a digital computing system having a central processing unit adapted to interact with a plurality of memory means each having a pre-assigned page number and a plurality of sequentially accessed address location, and wherein said central processing unit provides an address including both a page number corresponding to the pre-assigned page number of one of said plurality of memory means and the local address of a selected one of said plurality of sequentially accessed address locations, provides a memory access request including a read data signal, and is responsive to processing information provided thereto in response to said memory access request, and interface means including a plug-in type port for temporarily receiving said library module and for providing communication between the central processing unit and said library module, said library module comprising:

one of said memory means disposed within said library module, said memory means having processing information contained therein at sequentially accessed address locations;

first means for storing a page number provided by said central processing unit and for providing an enable signal when the stored page number corresponds to the pre-assigned page number of the memory means disposed within said library module,

second means for storing the local address provided by said central processing unit and for providing said local address to the memory means disposed within said library module,

connector means disposed within said library module and connectable to the plug-in port of said interface means, said connector means connecting the address provided by said central processing unit to said first and second storage means, and connecting the processing information outputted by said memory means to said central processing unit; and

control means responsive to the receipt from said central processing unit of a memory access request including a read data signal and to said enable signal for outputting the processing information at the memory location corresponding to the local address provided to said memory means and for incrementing the local address in said second storage means to the address of the next sequentially accessed location in said memory means, said control means being further operative to maintain the local address located in said second storage means in an unaltered state in the absence of a memory access request, whereby the processing information contained in said next sequentially accessed address location will be provided by said memory means in response to another memory access request including a read data signal.

* * * * *