



(22) Date de dépôt/Filing Date: 1993/11/19

(41) Mise à la disp. pub./Open to Public Insp.: 1994/06/09

(45) Date de délivrance/Issue Date: 2008/11/18

(62) Demande originale/Original Application: 2 149 876

(30) Priorités/Priorities: 1992/11/20 (US07/979,698);
1992/11/20 (US07/979,578); 1992/11/20 (US07/979,577);
1992/11/20 (US07/979,570)

(51) Cl.Int./Int.Cl. *A63F 13/10* (2006.01),
G09G 5/34 (2006.01)

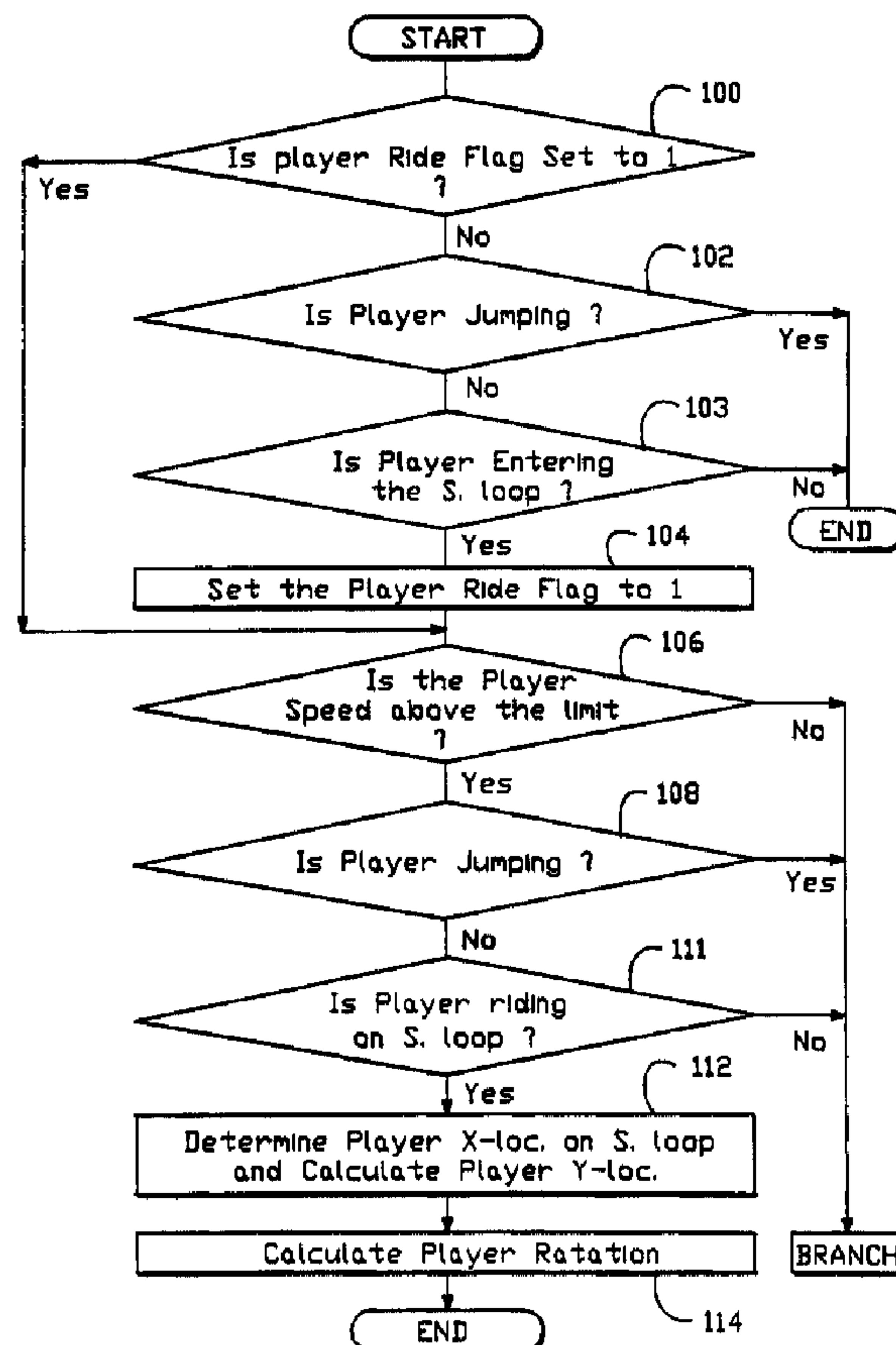
(72) Inventeurs/Inventors:
NAKA, YUJI, US;
YAMAMOTO, MASANOBU, US

(73) Propriétaire/Owner:
SEGA ENTERPRISES, LTD., JP

(74) Agent: BORDEN LADNER GERVAIS LLP

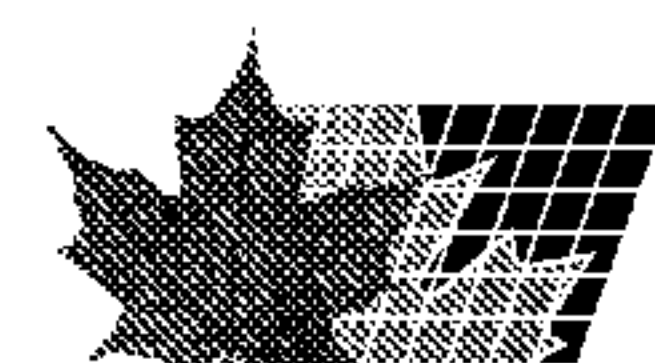
(54) Titre : METHODE DE COMMANDE D'AFFICHAGE

(54) Title: DISPLAY CONTROL METHOD



(57) Abrégé/Abstract:

A method for controlling a multiplayer competition video game for use with a system which includes a video display screen, a user-controlled graphics controller, digital memory and at least one user input device; wherein the video game involves multiple game



(57) **Abrégé(suite)/Abstract(continued):**

characters each traversing a scrollable playfield in response to the user input device; wherein the playfield is displayed as a series of scrolled video screen displays; and wherein progress of the game characters in traversing the playfield is indicated by scrolling the video display screens of the playfield behind the characters.

ABSTRACT

A method for controlling a multiplayer competition video game for use with a system which includes a video display screen, a user-controlled graphics controller, digital memory and at least one user input device; wherein the video game involves multiple game characters each traversing a scrollable playfield in response to the user input device; wherein the playfield is displayed as a series of scrolled video screen displays; and wherein progress of the game characters in traversing the playfield is indicated by scrolling the video display screens of the playfield behind the characters.

1 **DISPLAY CONTROL METHOD**

This application is a divisional application of co-pending application (serial number 2,149,876) filed November 19, 1993.

BACKGROUND OF THE INVENTION

5

1. Field of the Invention

The invention relates generally to video games, and more specifically, to video games that employ a playfield that scrolls relative to a game character in order to show
10 character motion through the playfield.

2. Description of the Related Art

Video games are well known in which a game character, or sprite, follows a prescribed path through a scrolled
15 playfield in response to the commands of a user who is playing the game. A game character sometimes have the human-shape but some characters have other figures (animals, monsters, Vehicles, geometric patterns and symbols, for example). A sprite is an image block usually
20 in shape of a rectangle to be represented on a graphic screen. A sprite can be of any size and it is also possible to display a number of sprite of various sizes simultaneously on a single screen during execution of a
24 game. Since sprites can be displayed by high-speed

1 procedures (DMA, for example), sprites accelerate the game
execution.

5 The user operates an input device which includes a
control console which may include a "joy stick" used to
control character movement. A user command issued through
the input device for the character to move right results
in the playfield scrolling to the left which creates the
impression that the character is moving to the right
10 relative to the playfield. Conversely, a user command
for the character to move left results in the playfield
scrolling to the right which creates the impression that
the character is moving to the left. During such
scrolling the character image ordinarily remains fixed
15 near the center of the screen display despite the
appearance of movement relative to the playfield.

Since the playfield is far more wide than the
physical graphics screen, the user can see just a part of
20 the playfield simultaneously. The user plays the game by
making the character move wherever the user wishes in the
playfield. These kind of games typically are called Roll-
Playing-Games. Maniac habitual users sometimes make the
24 entire map of the playfield.

1

In typical video games, the input device permits the user to command the character to perform numerous activities such as to jump or to crouch down or to speed-up or to slow-down. Often, a game character takes on different appearances as it engages in different activities. For example, when the character moves at slower speeds, its legs, arms and torso may be fully visible as the playfield scrolls slowly. However, when the character speeds up, most of the character image may be portrayed as a blur with only the character head being fully recognizable while the playfield scrolls rapidly. Moreover, the character may have one portrayal when it crouches and another portrayal when it jumps. Additionally, there may be special character antics that involve a series of images, such as tumbling, throwing a kick or "flying" through the air.

A challenge associated with implementing such video games is to produce a playfield which has a variety of images and obstacles. For example, there may be mountains to climb, canyons to jump over or enemies to defeat. In the midst of all of this activity, changes in the appearance and movements of the character and the

1 playfield must occur smoothly and quickly so as not to
distract the user or detract from the excitement of a high
speed action packed video game.

5 Consequently, certain uniform techniques often are
employed to control the movement and appearance of game
characters and the playfield. These techniques include
defining a path to be followed by the character through
the playfield. The character moves along the path in
10 response to user commands. For example, if the path
ascends to the right, and the user commands the character
to move to the right, then the character is depicted
climbing the path to the right as the playfield scrolls to
the left. If the user subsequently commands the character
15 to move back towards the left, then the character will be
shown descending the path to the left as the playfield
scrolls to the right.

The image of the character moving along the path is
20 produced under the control of a computer program.
Character movement is constrained by the program such that
as the character moves left or right, it always tries to
maintain contact with the path. If the character jumps,
24 for example, it soon returns (falls) to the path. If the

1 path includes a discontinuity such as an on-screen image
of a cliff, for example, then a character running off the
edge of the cliff might fall to another path at the base
of the cliff; or it might jump across the chasm at the
5 edge of the cliff and land on a path opposite the cliff.
The user controls character movement, but the program
ensures that the character generally follows the path.

While paths through the playfield generally have been
10 an acceptable way to constrain character movement, there
have been shortcomings with there use. For example, a
typical earlier method for making sure that a character
follows the path is to use collision blocks. A collision
blocks is a data block to represent an area if a
15 character can traverse (on or above the ground, for
example) it or not (in the soil, for example). Thus the
character which moves on the border between an area can be
traversed and another area can not be traversed moves on
the ground. In the same manner, if "walls" is constructed
20 on both sides of a character, the character can not move
to the sides of a path.

As the character moves in response to user commands,
24 a computer program references stored collision blocks to

1 determine the exact path to be followed in response to
such commands. Specifically, the playfield is divided
into graphics blocks. As the character traverses
individual graphics path blocks, a path control program
5 references individual collision blocks that correspond to
such graphics path blocks. A collision block is used to
determine, for example, whether the path within a graphics
path blocks is level, inclined or drops off a cliff.

10 A problem associated with such prior methods is that
it may be desirable for the character to proceed along a
path that corkscrews or turns upside down or banks (or is
inclined) sideways. Since a typical earlier path program
operates by keeping a character firmly planted on a path
15 as if due to the force of gravity, the character
ordinarily would fall off such a banked or upside down
path.

Thus, there has been a need for improved graphics
20 techniques which display banked, upside down or sideways
path segments. The present invention meets this need.
/

One of other challenges associated with video games
24 is to provide a competition mode in which two video game

1 players independently control two game characters as the
characters traverse the playfield. For example, the
players compete by trying to have their characters
traverse the playfield fastest while accumulating the
5 highest number of points by overcoming obstacles or
slaying enemies or gathering "magical rings". One problem
associated with competition games is that the character
controlled by one player may get far ahead of the
character controlled by the other player as the two
10 characters traverse the field. In order to make the game
more exciting and challenging, it can be desirable to have
two characters in closer competition.

Thus, there has been a need for a method and
15 apparatus in which the lead that one character has
achieved over the other can be changed or reduced when one
character has outdistanced the other. The present
invention meets this need.

20 One of other problems associated with such earlier
methods is that it may be desirable to define two
different paths through the same region of a graphics path
block. Unfortunately, a collision block typically defines
24 only a single path, and each graphics path block

1 corresponds to only one collision block. Therefore, it has
been difficult to provide multiple paths through the same
region of a graphics path block.

5 Thus, there has been a need for improved techniques
which permit multiple paths through the same region of a
graphics path block. The present invention meets these
needs.

10 One of other challenges associated with such video
games is to provide a competition mode in which two
players controlling different characters can compete with
each other as game characters traverse the playfield.
There are numerous earlier video games in which two game
15 characters compete as the characters traverse a scrollable
playfield. A problem associated with such competition is
that one character may outdistance the other as the two
characters progress through the playfield, thus making it
difficult to display both characters simultaneously.
20 Another potential problem is that the use of a split
screen display in which each of two screens displays part
of the playfield can result in reduced graphics due to the
decreased size of the respective split screens.

1 Thus, there has been a need for a method and
apparatus for use in split screen video game completion.
The present invention meets this need.

5 One of other challenges associated with implementing
video games is to encourage novice players to improve
their skills and to permit novice players to play the game
with more experienced skilled players. One problem with
fulfilling this objective is that in many video games,
10 progress of a game character through the playfield depends
upon the skill of the player controlling that character's
movement. A game character controlled by an accomplished
player often will move further or more quickly through a
playfield, than a game character controlled by a less
15 skilled or less experienced player. As a result, it
sometimes can be difficult to simultaneously display both
characters controlled by a skilled player and a less
experienced player on the same display screen since the
character controlled by the skilled player will be far out
20 in front of the other character in the playfield.

 Thus, there has been a need for a method and
apparatus for permitting a game character controlled by a
24 less experienced player to keep pace with a game

1 controlled by a more skilled or experienced player. The
present invention meets this need.

5 **SUMMARY OF THE INVENTION**

In one aspect, the invention involves a method for
controlling the appearance of a video game character as
the character traverses a path displayed on a display
10 screen. The method is especially beneficial in a video
game system which includes a graphics controller, digital
memory and a display screen. A banked path is divided to
segments which are displayed on the screen. A video game
character is displayed upright in at least one location on
15 the banked path and is displayed upside down in at least
one other location on the banked path. Multiple sprite
patterns are stored in the digital memory. The different
patterns are representative of the appearance of the
character at different locations on the banked path as the
20 character traverses the banked path. The character's
location on the banked path is tracked as the character
traverses it. Different sprite patterns are used to
depict the character at different locations on the banked
24 path as the character traverses the path.

1

Thus, for example, the character can follow a path which banks at a severe angle, twists sideways or even turns upside down. The character does not fall off the path when it twists and turns, and the character image changes as the character moves along the path. For example, if the path turns upside down, then the character is shown upside down. If the path banks sideways then the top of the character's head might be shown as the character moves along a path that is parallel to the screen.

A method is provided for controlling a multiplayer competition video game for use in a system which includes a video display screen, a graphics controller, digital memory and at least one user input device. In a current embodiment, a split screen display is provided in which a top screen displays a first region of the playfield and a bottom screen display displays a second region of the playfield. A first character is displayed in the top screen, and a second character is displayed in the bottom screen. The first playfield region in the top screen is scrolled to display progress of the first character through the playfield, and the second playfield region in

1 the bottom screen is scrolled to show the progress of the
second character through the playfield. An exchange
object is displayed in at least one of the top and bottom
screens. The exchange object is activated, and as a
5 result, the respective playfield displays in the top and
bottom screens are interchanged such that after the
interchanging process, the top screen displays the first
character in the second playfield region, and the bottom
screen displays the second character in the first
10 playfield region.

Thus, by activating the exchange object, the
positions of the first and second characters are reversed.
A character which once was far out in front of the other
15 character now is far behind that character. The present
exchange object, therefore, adds additional strategy
factors to the video game play.

In one aspect, the invention involves a method for
20 displaying a video game character traversing a video game
playfield, for use with a system which includes a video
screen display, a user-controlled graphics controller and
digital memory. The playfield is displayed as a series of
24 scrolled screen displays. The video game character

1 follows a path within the playfield. The progress of the
game character in traversing the playfield is indicated by
scrolling the playfield relative to the game character.

5 Multiple collision blocks used to define respective
path segments are stored in digital memory. The playfield
is divided into multiple graphics path blocks that
comprise the path. Stored character collision type
information indicates whether a particular character is to
10 be regarded as either a first character collision type or
a second character collision type. References are
provided from individual graphics path blocks to
individual collision blocks. At least one reference is
dependent upon the character collision type of a character
15 traversing a particular graphics path block. Character
movement through the playfield in response to user input
to the graphics controller is displayed on the screen
display. The displayed character image follows a path
defined by the path segments of individual collision
20 blocks referenced to individual graphics path blocks that
comprise the path. The stored character collision type
information is changed when the character passes a
prescribed location on the playfield such that after the
24 change, the stored character collision type information

1 indicates a different character collision type than before
the change.

Therefore, for example, the character is able to
5 follow a path that crosses over itself. As the character
approaches the cross-over a first time, it stores one type
of collision information, and it follows one branch of the
cross-over without colliding with the other branch. As
the character approaches the cross-over a second time, it
10 stores another type collision information, and it follows
another branch of the cross-over without colliding with
the first branch.

A method is provided for controlling the display of
15 game character movement in a video game for use in a video
game system which uses a video display screen and which
includes a graphics controller with digital memory. First
and second user input devices are used to control game
character movement. A first game character is responsive
20 to the first user input device, and the second game
character is responsive to the second user input device.
The video game involves the game characters traversing a
playfield which is displayed as a series of video screens.

1 A split screen display is provided. A first
playfield screen is displayed in upper portion of display
screen, and a second playfield screen is displayed in a
lower portion of the display screen. An interlace video
5 screen rendering technique is used to display both the
first and second playfield screens. The user input
devices are used to provide game character movement
commands to control the movement of the first and second
characters to the playfield. The movement of the first
10 character is displayed in the first playfield screen, and
the movement of the second character is displayed in the
second playfield screen.

Thus, two players can compete with each other by
15 independently controlling a movement of game characters in
distinct portions of a split screen display. A first
player can control the movement of a first game character
in a first playfield screen by operating a first user
input device. A second player can control the movement of
20 a second game character in a second playfield screen using
a second user input device.

The present invention provides a method for
24 controlling the motion of two game characters in a video

1 game for use in a system which includes video display
screen, a user-controlled graphics controller, digital
memory, a first user input device and a second user input
device. The first game character moves in response to the
5 first user input device, and the second game character
moves in response to the second user input device. The
video game involves the two game characters traversing a
playfield which is displayed as a series of video screen
images. A succession of game character movement commands
10 are provided to the first user input device. A
corresponding succession of movements of the first
character within the playfield are displayed on the screen
in response to the succession of commands. The succession
of commands are temporarily stored. The second character
15 is displayed moving within the playfield in response to
the same succession of commands.

Thus, the second game character can be shown to be
following and imitating the movements of the first game
20 character. In this manner, the second game character can
keep pace with the first game character.

In another aspect of the invention, a user can
24 periodically take control of the second game character

1 such that the second game character can compete with the
first game character as the two characters traverse the
playfield. If the second game character falls
significantly behind the first game character or if the
5 user controlling the second game character fails to
provide any control inputs for a prescribed period of
time, then the game play reverts to a mode in which the
second game character follows and imitates the first game
character as described above.

10

These and other purposes and advantages of the
present invention will become more apparent to those
skilled in the art from the following detailed description
in conjunction with the appended drawings in which:

15

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a video display and a
20 priority controller and a concept diagram of the scroll
planes used to produce video images in accordance with the
invention;

24

Figure 2 is a conceptual view of an entire playfield

1 which is sixty screens wide and eight screens high in
accordance with the invention;

5 Figure 3 is a representative (blank) screen within
the playfield of Figure 2;

Figure 4 is an 8-dot x 8-dot graphics cell within the
screen of Figure 3;

10 Figure 5A represents a color pattern used to
illuminate the graphics cell of Figure 4;

15 Figure 5B represents an 8 bit byte of color
information used to illuminate two dots of the color
pattern of Figure 5A;

20 Figure 6 represents the use of a pattern number table
to correlate the graphics cells of Figure 4 with the color
patterns of Figure 5A;

Figure 7 is a block diagram of a graphics controller
in accordance with the present invention;

24 Figure 8 shows a representative pattern number stored

1 in the pattern number table of Figure 6;

Figure 9 is a flow diagram of the steps used to
access color pattern information using the color pattern
5 number table of Figure 6;

Figure 10 is a representative sprite table that can
be stored in the control RAM of the controller of Figure
7;

10

Figure 11 shows a representative sprite attribute
table stored in the VRAM of the controller of Figure 7;

Figure 12 is a table showing the priority encoding
15 rules implemented by the priority encoder of the TV
interface circuit of the controller of Figure 7;

Figure 13 is a series of color patterns illustrating
the role of the vf and hf bits of the pattern number of
20 Figure 8;

Figure 14 is a representative link data list for
linking sprites by priority for use in the graphics
24 controller of Figure 7;

1

Figure 15 shows a series of pattern generator data formats in which different patterns can be stored to generate sprites of different horizontal and vertical sizes;

5

Figures 16A-D show a shuttle loop in accordance with the invention; 16A shows character orientation; 16B shows character offset; 16C shows character trajectories in the event that the character jumps; and 16D shows offset measurements relative to the center of the loop;

10

Figure 17 shows a collection of character image patterns displayed on screen as the character traverses the loop of Figures 16A-D;

15

Figures 18A-B show a flow diagram of a computer program used to control the display of character movement on the loop of Figures 16A-D;

20

Figures 19A-B show a table of angles and an offset table used to select character patterns and to determine character offset as the character traverses the loop of Figures 16A-D;

24

1

Figures 20A-C show an illustrative overlap loop path and the graphics path blocks that comprise the path in accordance with the invention;

5

Figure 21 shows an illustrative library of graphics blocks used to define path segments of the graphics path block of Figures 20A-C;

10

Figure 22 is a table which cross-references graphics blocks to collision blocks;

15

Figure 23 is an enlarged view of one of the collision blocks of the library of Figure 21 used to illustrate the process of locating the path segment defined by a collision block; and

20

Figure 24 is a flow diagram of a process used to change the collision type of a game character when the character passes a prescribed region of the playfield;

24

Figures 25A-B show possible earlier single screen graphics including playfield graphics and character graphics and possible earlier split screen graphics in

1 which some of the playfield graphics have been lost in the
split screen image;

5 Figures 26A-B show consecutive views of another
possible earlier single screen graphics display in which
two characters in the same screen are moving away from
each other as indicated by the arrows;

10 Figures 27A-B show single screen graphics and split
screen graphics in accordance with the present invention;

Figure 28 illustrates an interlaced frame;

Figure 29 illustrates an even scan frame;

15

Figure 30 illustrates an odd scan frame;

20 Figures 31A-D illustrate four stages in the rendering
of a split screen image in accordance with the present
invention;

Figure 32 shows a timing diagram corresponding to the
four stages in Figures 31A-D;

24

1 Figures 33A-B is a flow diagram representing a
computer program which explains the data transfer during
rendering of a split screen display in an interlace mode;

5 Figure 34 illustrates a computer program used to
control the exchange of graphics information during the
rendering of a split screen display;

 Figure 35 shows a single cell which can be used to
10 produce a graphics pattern for use in interlace mode;

 Figure 36 shows another representation of a single
cell which can be used to produce a graphics pattern in
the interlace mode;

15

 Figures 37A-E illustrate the operation of the
cooperative mode by showing a series of screen displays
which might appear as first and second players traverse a
prescribed path through a play- field;

20

 Figure 38 is a flow diagram which illustrates the
operation of a computer program used to control the
movement of the second game character in the cooperative
24 mode;

1

Figure 39 is a flow diagram of a computer program used to determine whether the cooperative mode or the competitive mode is to be used by the second game character;

5

Figure 40 is a flow diagram of a computer program used to move the second character back into view after it has fallen far behind the first character and has disappeared from view on the display screen;

10

Figure 41 shows a split screen display in accordance with the present invention in which playfield region A is shown in a top screen and playfield region B is shown in a bottom screen and in which a first player is shown in the top screen and a second player is shown in the bottom screen;

15

Figure 42 shows a split screen display in which the playfield regions for the top and bottom screens shown in Figure 41 have been interchanged;

20

Figures 43A-B show a flow diagram of a computer program used to control the exchange of information as a screen display changes from the screen display shown in

24

1 Figure 41 to the screen display shown in Figure 42;

Figure 44 shows an exchange object in accordance with
the present invention prior to activation; and

5

Figure 45 shows the exchange object of Figure 44
after activation.

10 **DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT**

The present invention comprises a novel apparatus and
method for use in a video game. The following description
is presented to enable any person skilled in the art to
15 make and use the invention, and is provided in the context
of a particular application and its requirements. Various
modifications to the preferred embodiment will be readily
apparent to those skilled in the art, and the generic
principles defined herein may be applied to other
20 embodiments and applications without departing from the
spirit and scope of the invention. Thus, the present
invention is not intended to be limited to the embodiment
shown, but is to be accorded the widest scope consistent
24 with the principles and features disclosed herein.

1

Overview

In a present embodiment of the invention, a video display generator is used to produce graphical images on a TV display screen. The graphics information used to produce the images can be thought of as a series of planes, one behind the other as shown in the illustrative drawing of Figure 1. The first plane is the sprite plane. The next two planes are the scroll A plane and the scroll B plane. The actual image produced on the TV screen consists of a series of rows of dots that are individually illuminated in different colors so that together the dots form the image. The graphics information used to determine how each dot is to be illuminated is provided in the three planes. For each dot, a priority controller determines whether the dot is to be illuminated using information from the sprite plane, the scroll A plane or the scroll B plane. The graphics information in the three planes is prioritized, and dots having the highest priority are displayed.

Scroll A represents a playfield on which a video game is played. A playfield is a virtual space (a nation, a labyrinth, an island or a castle, for example) in the

1 video game. Scroll B represents a background for scroll
A. The video display can show only a tiny fraction of the
entire playfield at any given moment. An entire playfield
includes 480 separate screen displays. A video game
5 involves the movement of video game characters through the
playfield. The illustrative drawing of Figure 2
represents a playfield which is 60 screens horizontal and
8 screens vertical. The system of the present invention
can operate in either a single screen mode or a split
10 screen mode. In the single screen mode of operation, only
a single playfield screen is displayed. That screen
corresponds to the portion of the playfield presently
traversed by the video game character. In a split-screen
mode, two screens are displayed at a time, one for a first
15 video game character and another for a second video game
character. For each character, the displayed screen
corresponds to the portion of the playfield traveled by
that character at that moment.

20 The playfield image that appears on the display is
formed from graphics information stored for scroll A and
scroll B. Images called sprites also appear on the
screen. Sprites are graphics objects that can move about
24 on the playfield. The game characters are sprites, for

1 example. Graphics information for the sprites is stored
in the sprite plane.

Each screen, in a non-interlace mode of operation, is
5 320 dots horizontal and 224 dots vertical. The
illustrative drawing of Figure 3 shows a representative
screen. The screen, for example, can be at the screen
location indicated by the rectangle Q in the playfield
shown in Figure 2. The rectangle Q is horizontally and
10 vertically offset from the playfield base address at the
upper left corner of the playfield.

The entire playfield is divided into graphics blocks.
Each graphics block is divided into graphics cells. Each
15 graphics cell is divided into dots which correspond to
pixels. Each graphics cell represents an 8 dot x 8 dot
region of the screen. In the preferred embodiment, there
are 40 graphics cells horizontally and 28 graphics cells
vertically per screen.

20

It will be understood that graphics for a typical
playfield are initially created by an artist. The
graphics then are "digitized". The graphics are divided
24 into graphics cells. The graphics cells are referenced to

1 stored color patterns that contain color information used
to color the graphics cells. That way stored color
information can be reused for different graphics cells.

5 For example, a particular region of the scroll A
portion of the playfield may represent green grass. Each
graphics cell that portrays green grass can use the same
stored pattern information. Rather than separately store
green grass graphics for each graphics cell portraying
10 green grass in the playfield, a green grass pattern is
stored once, and the individual graphics cells are
referenced to that stored pattern which can be used to
produce grassy screen images.

15 The illustrative drawing of Figure 4 shows a
representative 8-dot x 8-dot graphics cell. For each cell
in the displayed image, a determination is made as to
whether the dots in the cell are to be illuminated using
color pattern information stored for the scroll A plane,
20 the scroll B plane or the sprite plane. As explained
above, the same stored pattern information can be used by
multiple graphics cells. The function of the graphics
controller described below is to make that determination.

1 Color pattern information is stored for the sprite
plane and for the scroll A and scroll B planes. Each
pattern dictates the dot color illumination pattern for an
entire graphics cell. For example, Figure 5A represents a
5 pattern in which color information is provided for 64 dots
a1-h8. As shown in Figure 5B, 4 bits of color information
are provided per dot. The color information is stored in
32 8-bit bytes in which each byte stores 4 bits for one
dot and 4 bits for another dot. For each graphics cell in
10 the display screen Q, a determination is made as to
whether a color pattern stored in the sprite plane, in the
scroll A plane or in the scroll B plane is to be used.
The color data of the selected pattern then is used to
illuminate the dots of that cell.

15

The general approach used to locate scroll A and
scroll B graphics patterns stored in memory is illustrated
in Figure 6. A table of pattern numbers is maintained for
the cells in scroll A, and another table is maintained for
20 the cells in scroll B. For the scroll A table, a pattern
number is stored for each cell of the screen. Each
pattern number is stored together with a pointer to an
entry in a scroll address table which, in turn, points to
24 a subroutine used to access the stored graphics pattern

1 that corresponds to the pattern number. In this manner, a
graphics pattern can be located for each graphics cell of
scroll A. The graphics patterns used for graphics cells
in the scroll B plane are located using a similar
5 approach.

The procedure used to locate color patterns for
graphics cells in scroll A and for graphics cells in
scroll B is discussed in more detail below, together with
10 a discussion of the steps for locating color patterns for
the sprite plane.

Graphics Controller

Referring to the illustrative drawing of Figure 7,
15 there is shown a block diagram of a graphics controller 40
in accordance with the present invention. The controller
40 includes a RAM 42, a video RAM (VRAM) 45, a
Microprocessor 44, a processor interface 45, control logic
46, a control RAM 48, a horizontal counter control 50, a
20 vertical counter control 52. The controller 40 further
includes interrupt control 57, direct memory access (DMA)
control 59, a line buffer 60, registers 61 and an I/O
interface 64. A TV interface circuit 54 provides RGB
24 analog signals to a TV system 56.

1

The RAM 42 receives graphics information from a cartridge-based ROM 58. A cartridge 63, which forms no part of the present invention, is disclosed in commonly assigned U.S. Patent application Serial Number 07/510,070
5 filed April 17, 1990, invented by Matsubara.

10

First and second external controllers 72, 74 are connected to the controller 40 through the I/O interface 64. The controllers 72, 74 each include buttons to control game character movement.

15

The first controller 72 controls the movement of a first game character (sprite). The second controller 74 controls movement of a second game character (sprite). The S/P button controls start/pause of game play. Buttons A, B, C are used for special game features such as
20 character attacking or character taking on special powers. Buttons labeled with L, R, Up, Dn are used to cause the game character to move left or right or to jump up or to crouch down.

24

1 In operation, the video RAM 43 stores graphics
patterns like those shown in Figure 5A for the sprite
plane and for the scroll A and scroll B planes. As the TV
display screen is scanned line- by-line, patterns
5 corresponding to the graphics information for the scroll
A, scroll B and the sprite plane are retrieved, and three
independent signals are produced which are representative
of the graphics patterns for scroll A, scroll B and the
sprites plane. A priority controller in the TV interface
10 54 selects the appropriate signal on a cell-by-cell basis
according to designated priorities. A color decoder and
DAC in the TV interface 54 receives the output of the
priority controller and generates a corresponding RGB
signal for transmission to the TV system 56.

15

More specifically, the control logic 46 receives a
horizontal scroll value and a vertical scroll value which
determines which graphics cell in the playfield is to be
rendered.

20

A pattern number table address in the VRAM 45 is
calculated based upon the received vertical and horizontal
values. The calculated address contains the number that
24 identifies the pattern used to color the graphic cell. A

1 representative pattern number stored in the pattern number
table is shown in Figure 8. The retrieved pattern number
is used to access a color pattern stored in the VRAM 45.
The color pattern together with color palette selection
5 information retrieved from the pattern number table is
used to calculate a color RAM address. The diagram of
Figure 9 further explains the process of retrieving scroll
pattern information. The process is similar for scroll A
and scroll B patterns.

10

The following pattern number entry table explains the
contents of the bytes of the color pattern number of
Figure 8.

15

Pattern Number Table Entry

	pri:	Priority
	cp1:	Color palette selection bit
20	cp0:	Color palette selection bit
	vf:	V Reverse Bit
	hf:	H Reverse Bit
	pt10 ~ pt0:	Pattern Generator Number

24

1

The priority bit refers to priority of the pattern. The two color palette selection bits refer to the selection of the appropriate color palette. In the
5 presently preferred embodiment, there are four color palettes.

The vf and hf bits are explained with reference to the drawings of Figure 13. Essentially, bits vf and hf
10 allow for horizontal and vertical reversal on a cell unit basis. That is, depending on the vf and the hf values, the vertical or horizontal or both vertical and horizontal orientation of a cell can be changed. In this manner, graphics information can be stored more compactly.
15 Different graphics images can be produced using the same stored color patterns simply by changing the orientation of individual color patterns using the vf and hf bits.

The controller 40 processes sprite graphics as
20 follows. Upon receipt of a vertical counter signal from the vertical counter 52 a search is made of the control RAM 48 for a sprite having a vertical position indicated by the vertical counter. The control RAM stores the
24 vertical position, sprite size, a link number and a

1 pattern number. Upon finding one or more sprites having
the indicated vertical position in a sprite table shown in
Figure 10. The RAM 42 returns to the control logic 46 the
size and link numbers for all such sprites sharing that
5 vertical position. The video RAM 45 contains a sprite
attribute table which also is searched for sprites having
the vertical address of interest. The drawings of Figure
11 show the format of an entry in the sprite attribute
table. The pattern number and horizontal position of each
10 sprite having such a vertical address are returned from
the VRAM 45 to the control logic 46.

In figure 11, sprite attributes are stored in VRAM
and the base address is indicated by register #5. 8 bytes
15 (4 words) is needed to indicate the attribute for each
sprite and the attribute presents display location,
priority, sprite generator number and attributes. From
the beginning of the attribute table, numbers is assigned
to each area such as sprite 0, 1, 2, 3. Since priority
20 among the sprites is determined not by the order of sprite
number but by the link data between each sprites, it is
programmable.

24 Priority among the sprites is determined in

1 accordance with a sprite link table explained below.
Based upon the outcome of the individual sprite
priorities, the pattern numbers of the sprites to be
displayed for the particular vertical line is determined.
5 Horizontal count information is provided by the horizontal
counter 50. If there are multiple sprites on a line then
a calculation of sprite priority determines which sprites
are to appear in view and which are to be hidden from
view. The pattern number is used to address the
10 appropriate sprite color patterns stored in VRAM 43. Dot-
by-dot illumination information for the vertical line is
transferred to the line buffer 60.

The role of the Priority Controller in the TV
15 interface circuit 54 will be better understood by
reference to the table of Figure 12. For each graphics
cell visible within the display screen, a priority is set
for the sprite plane, the scroll A plane and the scroll B
plane. The graphics cell with the highest priority is
20 displayed in accordance with the table. Once priority has
been determined, an RGB analog signal is produced for
either the sprite, scroll A or scroll B signal, depending
upon which has highest priority. Thus, for example, if
24 scroll A has the highest priority for a given graphics

1 cell, then the color information produced by the graphics controller for scroll A for that cell is provided as the RGB analog signal to the TV system 56.

5 Sprites

A Sprite is defined through a Sprite attribute table entry which is stored in VRAM 43 and a sprite status table stored in RAM 42. The following sprite status table lists representative status information that is stored in the RAM 42 for main character (hero) type sprites as well as for various other sprites such as enemies or moving platforms.

15 Sprite Status Table

	No. of Bytes	Description
	1	Action Number
	1	Action Flags
20	2	Offset in VRAM
	4	Address of pattern table
	4	X direction offset within playfield
	4	y direction offset within playfield
24	2	+ x direction speed

<i>1</i>	2	+ y direction speed
	1	vertical offset (in dots) from center of character to bottom of char.
	1	horizontal offset (in dots) from center of character to bottom of character.
<i>5</i>	1	sprite priority
	1	horizontal width in dots
	1	pattern number
	1	pattern counter
<i>10</i>	2	pattern change number
	1	pattern timer counter
	1	pattern timer master
	1	collision size
	1	collision counter
<i>15</i>	1	Routine number 1
	1	Routine number 2
	2	angle of character through loop (not sloop)
	1	ride-on flag
	1	hit flag
<i>20</i>	2	A/B type collision setting

The Action Number is essentially the sprite's name.

Each sprite has a unique Action Number. An action flag
24 byte includes 8 bits in which one bit indicates whether

1 the Sprite is facing right or left. Another byte
indicates whether the sprite is resting on its head or on
its feet. Another byte indicates whether an offset to the
up or down edge of the Sprite is to be used. Still
5 another byte indicates whether or not the sprite is within
the field view of the display screen. Four bytes are used
to indicate the base address of the pattern data used to
produce the sprite.

10 Another 4 bytes are used to indicate the x direction
offset of the sprite within the playfield. Another 4
bytes are used to indicate the y direction offset of the
sprite within the playfield. Two bytes indicate the x
direction of movement (right or left) and speed of the
15 sprite. Two more bytes indicate the y direction of
movement (up or down) and speed of the sprite. Another
byte is used to set the vertical offset in dots from the
center of the sprite to the bottom of the sprite. Still
another byte is used to set the horizontal offset in dots
20 from the center of the sprite to the edges. There is a
byte to show sprite priority and a byte to indicate
horizontal width of the sprite in dots.

24 There is one byte which designates a pattern number.

1 There is a one byte pattern counter which tells how long
to display each sprite pattern on the screen. A sprite
may be illustrated using a sequence of patterns and the
duration of each pattern must be set. Two bytes indicate
5 the master pattern number also known as the pattern change
number. For example, there may be four patterns each to
show a running sprite image sequence. Another four
patterns may be used to show a rolling sprite pattern
sequence. The pattern change number indicates which set
10 of patterns is to be used. The one byte pattern timer is
similar to the pattern counter, and it contains a count of
how long a pattern should be displayed. Another byte
serves to keep track of the current count during a count
down/up. Another byte includes a hit number which indexes
15 into a hit table which determines the size of a character
for purposes of determining whether a hit has occurred.
The hit number indicates the size of a collision box for
the sprite for the purpose, for example, of determining
how big a target a sprite poses for enemies. For example,
20 a hero sprite may be attacked by enemy sprites that hurl
projectile sprites. Determining whether a hero has been
hit by a projectile involves finding the hero's hit size.
A collision count indicates how many collisions or hits
24 are required to "kill" or defeat a sprite.

1

One byte is used to indicate a subroutine zero. For example, if a sprite is not currently moving, then one subroutine is called to display the sprite. If the sprite is moving, then another subroutine is called to display it. Another byte indicates routine number one. For example, if the sprite is standing and shooting then one subroutine is called. If a sprite is standing and not shooting, then another subroutine is called.

10

Two bytes are used to change the angular orientation of the characters. They can be used to trace a path on an inclined/declined surface or through a 360 degrees loop. One byte is used to indicate the status of a ride-on flag. There are objects referred to as "events" in the playfield that the sprite can "ride-on". For example, there are moving platforms in the sprite plane, and the sprite is capable of riding on such platforms. If the sprite is resting on this platform, then the ride-on flag is set. A crash flag indicates whether or not a sprite has run into another object (other than a path) such as a wall or an enemy. Another byte indicates whether the sprite is an A type collision Sprite or a B type collision sprite.

24

1 Sprite Data Attribute Table

The illustrative drawing of Figure 11 shows a representative Sprite attribute table entry stored in VRAM 45. The following table explains the Sprite attribute information.

Sprite Attribute Table

10	vp9 ~ vp0:	V position
	hp8 ~ hp0:	H position
	hs1, hs0:	Sprite's H Size
	vs1, vs0:	Sprite's V Size
	ld6 ~ ld0	Link Data
15	pri:	Priority Bit
	cp1, cp0:	Color palette selection bit
	vf:	V Reverse Bit 1:Reverse
	hf:	H Reverse Bit 1:Reverse
	sn10 ~ sn0:	Sprite Pattern Number

20

The vertical and horizontal positions of the sprite are relative to the base address of the scroll screens. The sprite horizontal size can be set at either 8, 16, 24

1 or 32 pixels. Similarly, the Sprite vertical size can be
set, at either 8, 16, 24, or 32 pixels. The sprite
priority bit can be set, and its use is described above.
The color palette can be selected. The vf and hf bits can
5 be used to reverse a sprite orientation in a manner
similar to that described for the pattern generator
numbers. The sprite pattern number is indicated by eleven
bits sn10-sn0. The link data is used to indicate priority
among sprites.

10

Referring to Figure 14, there is shown a link data
list in which Sprites Q-W are linked in accordance with
priority. Sprite Q has number 1 priority. Sprite R has
number 2 priority. Sprite S has number 3 priority.
15 Sprite W has the lowest priority. In this manner priority
is sequentially designated by each sprite's link data and
thus it is in a list form. The lowest priority sprite has
a link data 0. If there are sprites that are not linked
into the list, then they are not displayed on the screen.

20

Referring to the illustrative drawings of Figure 15,
there is shown a series of pattern generator data in which
the different patterns are used to generate sprites of
24 different horizontal and vertical cell sizes. Each cell

1 corresponds to a single pattern which is represented by 32
bytes of information as described above. Thus, for
example, the sprite pattern generator data for Figure 15-N
has a total of eight graphics cells. Eight color patterns
5 are used to generate the eight graphics cells. It will be
appreciated that computer programs that will be understood
by those skilled in the art are stored in the RAM 42 and
are used to generate sprite graphics from the sprite
pattern generator data.

10

Shuttle Loop Graphics

Referring to Figure 16A there is shown a shuttle loop
in accordance with the present invention. The shuttle
loop is in the form of a spiral loop which is part of the
15 playfield. The spiral loop (sloop) forms a spiral or
corkscrew path on which a player rotates through 360
degrees as he runs along it. At points E1 and E2 (entry
points) the path surface is normal and the character
stands vertically straight up as indicated by the arrows
20 as he moves along the sloop path at these positions. At
point M (midpoint) the path is rotated 180 degrees so that
a character stands (completely) upside down at this point
on the pathway as indicated by the arrow M. At point T1
24 the path has begun to twist and to bank sideways and the

1 character is visible, but his orientation is oblique as
shown by the arrow T1. The banked path simulates the
banking of a real life race track in which the road is
inclined on banked turns to help racers to stay on the
5 track. When the character reaches the point indicated by
arrow T2 it is hidden from view since it is moving on a
portion of the sloop path that is banked but hidden from
view by the underside U of the sloop path.

10 The sloop provides a unique graphical image in which
a sprite can follow a twisting spiral path on the screen
as it moves upward until the path is completely upside
down or rotated 180 degrees. The character continues to
follow the path temporarily disappearing from view as the
15 path continues its corkscrew rotation through another 180
degrees so that the sprite character emerges again at the
other side of the sloop path standing upright. Producing
a sloop screen image poses a number of challenges. Not
the least of which is selecting the appropriate sprite
20 patterns for each location on the spiral path. As the
sprite character travels along the path its feet are
maintained on the path.

24 The character must be shown in a number of different

1 orientations from a side profile view to directly
overhead, when the path is banked ninety degrees, and from
standing completely upright to standing completely upside
down and at various other angular orientations depending
5 upon the banking of the path. Figure 17 shows twelve
different perspective patterns for a character sprite
affectionately known as "Tails".

The offset of the sprite character from the spiral
10 loop scroll pattern must be properly maintained since the
perspective view of the sprite character changes as the
sprite travels along these spiral loops. The line labeled
110 in Figure 16B approximates the offset from the loop to
the center of the sprite character as the sprite follows
15 the sloop path. When the sprite is at either end of the
sloop, his feet touch the path, and the sprite center is
clearly offset above from the path. However, as the
sprite travels along the path, the offset from the center
of the sprite character to the sloop changes, and at
20 certain points, the sprite is centered on the sloop path
with no offset. For example, at that sloop path location,
the path is tilted or banked sideways by ninety degrees,
and a top character pattern image is displayed on screen.
24 When the sprite reaches the half-way point, the sprite's

1 feet touch the path, and the sprite center is offset
downward from the spiral loop path. As the sprite
character continues to follow the path, his gradual
downward offset gradually decreases; the sprite
5 momentarily disappears from view; and it then reappears
with an offset slightly above the sloop path.

Shuttle Loop Control

As the character closes in on the shuttle loop on the
10 playfield, the program of Figures 18A-B begins to execute.
The execution of that program is in response to a
different program which constantly keeps track of the
character's location on the playfield. That other program
determines whether the player is closing in on the shuttle
15 loop. If he is, then the program of Figures 18A-B begins
to execute to determine whether or not the player is
actually on the sloop path yet. In essence, the other
program has a record of the playfield locations which are
considered to be close enough to the shuttle loop to
20 commence execution of the program of Figures 18A-B.

In step 100, a determination is made as to whether or
not the sprite character ride flag is set to one. The
24 sloop path is considered to be an object or "event" that a

1 sprite can ride on. The character ride flag simply
indicates that the player has landed on the sloop path.

5 In step 102 a determination is made as to whether or
not the character is in the process of jumping. The
reason for this step is to be certain that the character
has not come into close proximity with the beginning of
the shuttle loop by jumping from some other location. The
shuttle loop is only to be entered by a character moving
10 along a pathway, and not by a character who is jumping.

15 In step 103 a determination is made as to whether or
not the character is in the process of entering the
shuttle loop. There are certain zones defined at either
end of the shuttle loop indicated by boxes 105 in Figure
16A. The program of Figure 18A-B makes a determination of
whether or not the character is moving within either of
these entry zones. The program step indicated by 103
keeps track of player movement through either of these two
20 zones 105 to determine whether or not the character is in
the process of entering the shuttle loop from either the
left or the right.

24 It will be appreciated that the step 103 determines

1 whether the character is in the process of entering the
shuttle loop. As explained below, the character is
required to move very quickly as it enters the shuttle
loop. Also, once the character has entered the shuttle
5 loop, character movement no longer is controlled using
collision blocks, as explained in the next section.
Rather, as explained below, it is controlled by reference
to an offset table described in this section.

10 In step 104, if the character ride flag has not been
set previously to one and the player is in the process of
entering the shuttle loop from either the left or right
then the character ride flag is set to one. In step 106 a
determination is made as to whether or not the sprite
15 speed is above a pre-set limit. The rules of the video
game the present embodiment require that the sprite
maintain a certain minimum speed. Otherwise it cannot
enter the shuttle loop. This feature adds excitement and
realism to the game.

20

In step 108 a determination is made once again as to
whether or not the character is jumping. It should be
appreciated that the program of Figure 18A-B runs
24 repeatedly as the character traverses the sloop. It is

1 possible that the character could jump while in the
process of traversing the shuttle loop. Since the sloop
twists like a corkscrew, the result of such a player jump
could be that, rather than going upward, the character
5 falls down off the loop. For example, if the character
jumps from the left inclined or twisted side of the loop
as the sprite runs to the right, the character will end up
falling downward rather than jumping upward as indicated
by the trajectory shown by arrow 120 in Figure 16C. In
10 that case, the horizontal speed of the character is
maintained, as he falls downward. If, on the other hand,
the character jumps while on the inclined portion on the
right side of the shuttle loop as it runs to the left,
then it will be seen to follow the trajectory 122 as its
15 horizontal speed to the left is maintained and as it falls
downward. Its fall will be stopped when it hits the
surface 124. In step 111, a determination is again made
as to whether or not the character is riding on the
shuttle loop. If it is, then in step 112 a determination
20 is made of the character X (horizontal) location on the
shuttle loop, and a calculation is made of the character's
Y location. In step 114, the character rotation for that
X location is calculated. If in steps 106, 108 and 111 it
24 is found that the character's speed is not above the limit

1 or that the character is jumping or that the character is
no longer riding on the shuttle loop then a branch to step
116 in Figure 18B is made. The character ride flag is
cleared. In step 118, the number one is assigned to the
5 character's direction counter, and in step 120 the
character direction speed is set to the lower limit.

The direction counter is used to set the number of
rotations of the character when it falls off the loop
10 while jumping on either the right or left inclined
sections of the loop. The character tumbles as it falls.
This tumbling occurs only if the character is on a section
of the sloop which is at least a minimum distance above
the ground region 124. If the character is not far enough
15 above the ground region 124 when he jumps, then the
character simply falls down using its current pattern.
The speed flag set in step 120 tells the rate of tumbling
or rotation as the character falls during such a jump.

20 The offset of the center of the loop is calculated
from the vertical center of the sloop to a base address in
the playfield. The sloop is 64 dots in vertical height.
So, the Y offset is 32 dots up or down from that center as
24 indicated by Figure 16D. The sloop is 384 dots wide. For

1 each X dot increment, a y offset is stored in the offset
table of Figures 19A-B. The offset value in the table
tells the offset to where the center of the sprite should
be as shown in Figure 16B.

5

The correct sprite pattern is selected from the
twelve patterns in Figure 17. For each row in the offset
table, there is one entry in the Table of Angles
(orientation table). For example, when the character is
10 at an x offset of one dot, the y offset is 032, and the
orientation reference is \$00. For example, when the
character is at an x offset of 50 dots, the y offset is
030, and the orientation reference is \$16. The
orientation table entry indicates which of the twelve
15 patterns to select to show proper character perspective
for the character's current position on the sloop.

Thus, both sprite offset and sprite viewing
perspective are maintained as the sprite travels the
20 sloop.

Switchable A/B Collision Graphics

In a present embodiment of the invention, the
24 character follows a predefined path through the playfield

1 in response to user commands. The path may or may not be
visible throughout the playfield, but even if not visible,
it is present. The illustrative drawing of Figure 20A
shows a representative path which forms a loop and
5 overlaps itself (an overlap path). Stored information
referred to as collision blocks are used to keep the
character on the path. The use of collision blocks
represents a different approach to keeping a character on
a path than the approach described above in relation to
10 the shuttle loop.

A library of collision blocks is stored in the ROM
58. Figure 21 shows a collection of collision blocks
which may be used to keep a character on the overlap path
15 of Figure 20A. The playfield path is divided into
multiple graphics path blocks. Figure 20B shows the
graphics path blocks that contain the overlap loop.

As the character proceeds along the path, a record is
20 maintained as to which graphics path block presently is
being traversed by the character. A collision table such
as that shown in Figure 22 is used to make a cross-
reference from the graphics block to a collision block.
24 The collision blocks actually define the path segments

1 followed by the character. The graphics path blocks are
merely graphics, and they may or may not include actual
images of the path segments defined by the collision
blocks. As the character moves along the path, from one
5 graphics path block to another, it follows a path defined
by the individual path segments of the collision blocks
referenced to the graphics path blocks through the
collision table.

10 A novel feature of the present system is that cross-
references from a graphics path block to a collision block
can be made to be dependent upon the collision type
information stored in the character status table. By
changing the character collision type, a graphics path
15 block can be made to reference a different collision
block. Thus, the path segment followed by the character
can be made to be dependent upon the status table
information of the character.

20 Referring to Figure 22, there is shown a collision
table used to cross-reference path blocks to collision
blocks. The table indicates a path block number in the
first column. In the second and third columns it
24 indicates the collision block(s) to which the path block

1 is referenced. The references set forth in the second
column are used if the character itself carries "A" type
collision information, and the references in the third
column are used if the character itself carries "B"
5 collision type information. While the majority of the
table references are the same regardless of the type (A or
B) of collision information carried by the character,
certain of the graphics path block cross-reference depend
upon character collision type.

10

Specifically, graphics path block the G6, can cross-
references either of two collision blocks c1 or c8
depending upon character collision type. Similarly,
graphics path block G11 can cross- reference either of two
15 collision blocks c4 or c5 depending upon character
collision type. The remaining graphics path blocks that
make up the overlap path reference the same collision
block regardless of character collision type. For
example, graphics path block G5 is always cross-
20 referenced to collision block c0, and graphics path block
G12 is always cross-referenced to collision block c3.

Referring to the illustrative drawing of Figure 23,
24 there is shown an enlarged view of collision block c1 of

1 Figure 21. Collision block c1 is referenced to graphics
path block G6 when the character has A type collision
information. Assuming that the character 200 advances
along the path from left to right, it ascends as it
5 traverses graphics path block G6. The actual movement of
the character in the screen display as it traverses block
G6 is controlled by reference to collision block c1. If
the same character with type A collision information was
to cross block G6 from right to left, then it would
10 descend.

More specifically, there is shown a solid (shaded)
region and an empty (unshaded) region within the collision
block c1. The solid (collision) region is represented in
15 digital form in ROM by the storage of prescribed logical
information, for example, logical "1", and the empty (non-
collision) region is represented by the storage of a
different prescribed value, for example, logical "0". The
boundary between the solid region and the empty region
20 defines a path segment. As the character proceeds through
graphics path block G6 with A type collision information,
it follows the path segment defined in collision block c1.
The path followed by the game character comprises a
24 multiplicity of such path segments defined by different

1 collision blocks in the library.

In Figure 23, the outlines of a game character pattern are indicated by lines 200. As user commands
5 direct the character 200 to move either left or right, a determination is made as to whether or not the character needs to move horizontally, diagonally, upward or downward in order to remain in contact with the path segment.

10 A user operating an input controller 72 or 74 would merely indicate that the character should move left or right by depressing the buttons labeled L or R. If, for example, the graphics controller, determines that the character has entered a graphics path block that
15 references collision block c1 shown in Figure 23 and the character has A type collision information, then the controller would use the path segment defined by block c1 to determine the precise character movement through the graphics path block. For example, assume that a user has
20 commanded the character 200 in Figure 23 to move from left to right. A determination is made as to whether or not the character should move horizontally to the right or diagonally upward to the right or diagonally downward to
24 the right in order to remain on the path segment after the

1 movement.

5 The method whereby the character movement along individual path segments is controlled involves keeping track of the current location of the game character within the collision block. A determination is made as to which one of several possible movements of the character would leave the character suspended in empty (non-collision) space, within a forbidden solid (collision) area or 10 resting upon the path segment. The movement which would leave the character on the path segment is selected.

15 For example, still referring to Figure 23, predefined test points 202, 204 and 206 are used to test the possibilities by determining whether a leading edge 208 of the character would be left on the path segment surface, in empty space or in a solid region. The stored logical bits indicating solid collision regions are regions that are forbidden to entry by the character. A movement of 20 the character horizontally to point 204 would leave the character in a forbidden solid region; so that choice is out. A movement diagonally downward to the right to point 206 also would leave the character in a forbidden solid 24 region; that choice is out as well. However, a movement

1 diagonally upward to the right to point 202 would leave
the character disposed directly on the path; so that
option is selected.

5 It will be appreciated that the technique just
described for locating the path segment within a collision
block can use a similar set of test points for every
collision block. This uniformity of technique facilitates
high speed graphics and a high degree of flexibility in
10 selecting and producing game paths. Moreover, the details
of the technique, which includes keeping track of the
offset (location) of the graphics path block in the
playfield, will be understood by those skilled in the art
and need not be described in detail herein.

15

At the apex of the overlap loop indicated by line 210
in Figure 20B, a changing of character collision type
information is performed in accordance with the process
illustrated in the flow diagram of Figure 24. That is,
20 each time the character passes a region of the playfield
in the vicinity of line 210, the character collision type
is changed. If the character had been moving from right
to left through graphics path block G11, then the type
24 changes from A to B, and if the character had been moving

1 from left to right through graphics path block G11 then
the type changes from B to A. Thus, after the change, the
character collision type is different from what it was
before the change. Referring to Figure 22, it can be seen
5 that the cross-reference from block G11 changes with
character type. The cross-reference is to block c4 when
the character has A type collision information. The
cross-reference is to block c5 when the character has B
type collision information. Alternatively, for graphics
10 block G11, a single collision block could be cross-
referenced regardless of character collision type since
there is no overlapping of path segments and path is not
dependent upon character direction.

15 Assume, for example, that a character enters the
overlap loop from the left through graphics path block G5
and initially has collision type A. When the character
traverses graphics path block G6 a reference is made to
collision block c1. When the character reaches the apex
20 of the graphics path block G11, the collision type carried
by the character changes from A to B. As the character
descends from left to right and passes again through
graphics path block G6, this time, a reference is made to
24 collision block c8. Thus, during the first traversal of

1 graphics path block G6 one collision block is referenced,
and during the second traversal another collision block is
referenced.

5 Conversely, if the character were to initially enter
the overlap loop from the right and initially have
collision type B, then as it passed through graphics path
block G6, a reference would be to collision block c8. As
the character moved upwards and around the loop and
10 crossed the region indicated by line 210 in graphics path
block G11, the collision type information carried by the
character would change from B to A. As the character
descended and again traversed graphics path block G6, a
reference would be made to collision block c1. It should
15 be appreciated, that if the character were to move back
and forth across the region indicated by a line 210, the
collision type information carried by the character would
continue to change back and forth between A and B.

20 The use of references to collision blocks that are
dependent upon the collision type of a character
advantageously permits the use of paths that cross over
each other. Moreover, the use of references that depend
24 upon character collision type are particularly useful

1 where different collision blocks are to be invoked when
the character approaches a particular path block from
different directions.

5 Split Screen Competition Graphics

Split screen competition graphics advantageously
provides two scrolling screens each of which can scroll
over different parts of the same playfield. Character
sprites in the two screens can move independently of each
10 other on the two screens. This advantageously provides
the opportunity for full competition between players on
each independent scrolling screen. Moreover, both the
upper and lower scrolling screens can contain the full
amount of sprite graphics information. Since the two
15 screens each occupy half of the screen area that was
occupied in a single scroll screen mode, the images in the
two scroll screens are vertically compressed or squeezed
by a small amount. The scroll rate of each screen depends
upon the rate at which each character traverses its
20 screen. One character can be far ahead of the other
character. Also, in a present embodiment, the two sprite
characters can accumulate game points independently of
each other.

1 The split screen mode overcomes limitations of the
prior art. For example, Figures 25A-B illustrate a
potential problem with earlier split screens: some
playfield screen information is lost in going from single
5 screen mode shown in Figure 25A to the split screen mode
shown in Figure 25B. The large face graphics are an
illustrative part of exemplary playfield graphics which is
partially lost in the split screen mode. Figures 26A-B
show another potential shortcoming of some earlier scroll
10 competition games: competing characters can potentially
disappear from view if they become separated by too much
distance in the playfield.

Figure 27A illustrates the single screen mode, and
15 Figure 27B illustrates the split-screen mode in accordance
with the present invention. The two split screens in
Figure 27B are squeezed vertically as compared with the
single screen in Figure 27A. However, no information is
lost in the split screen mode; all scroll and sprite
20 information is present for use in each of the two screens
which scroll independently. Moreover, the display never
loses sight of either of the two sprite characters even
if they become separated in the playfield.

1 Interlace Mode Control

During operation in the interlace mode, an interlaced display field illustrated in Figure 28 includes both even scan lines 102 indicated by solid diagonal lines and odd scan lines indicated by dashed diagonal lines 104. Interlace mode is a well-known mode of TV operation which will be understood by those skilled in the art. A complete interlaced display field is produced over the course of two scan frames. During an even scan frame indicated by Figure 29 the even lines are traced. During an odd scan frame indicated by Figure 30, the odd lines 104 are traced.

In split-screen mode, there is a top screen 106 and a bottom screen 108. The two screens are demarcated by a boundary 110 between them. Since the top screen 106 and the bottom screen 108 can depict different regions of the playfield and can depict different sprite characters, access must be made to different sprite graphic information and different graphic playfield information in order to render the two different playfields. In a current implementation, the memory storage requirements for maintenance of all of the sprite graphics information for both the top and the bottom playfields 106, 108 can be

1 great. Consequently, in accordance with a current
embodiment of the invention, a technique is employed to
change the stored sprite graphics information in the
course of a single even frame trace and to again change
5 the sprite graphics information stored in the course of a
single odd frame trace.

More specifically, referring to the illustrative of
drawings of Figures 31A-D, there is shown a representation
10 of four time segments of a single frame trace. The first
segment shown in Figure 31A represents retrace during
which a vertical interrupt of the processor 44 is called.
The second segment shown in Figure 31B represents the
screen trace of the top screen. The third segment
15 illustrated in Figure 31C represents a segment during
which a horizontal interrupt of the processor 44 is
called. The fourth segment indicated by Figure 31D
represents a time period during which the bottom screen is
traced.

20

The timing diagram of Figure 32 shows the time
interval during which each of the above segments occurs.
Each of the two frames used to generate a complete
24 interlaced field is rendered in the course of 1/60 seconds

1 (16 milliseconds). Thus, an entire interlaced field is
generated in the course of 1/30 seconds (32 milliseconds).
Referring to the timing diagram, during the first three
milliseconds of the rendering of one frame, a vertical
5 interrupt is called, as the beam retraces from the bottom
right portion of the screen to the top left portion.
During that time period, sprite graphics information for
the top screen is transferred from RAM 42 to VRAM 45.
During the next 6 milliseconds the top screen 106 is
10 rendered using the stored graphics information. A
horizontal interrupt is called right before the next one
millisecond, and the screen is turned off so that no
picture is generated. During this time period, sprite
graphics information is transferred from RAM to VRAM for
15 the bottom screen. Right before the last 6 milliseconds
the screen is turned back on and the bottom screen 108 is
rendered. It will be understood, of course, that the
above-described process occurs for both the even frame and
the odd frame. Thus, it occurs twice to render one
20 complete interlaced field.

The illustrative drawings of Figures 33A-B and 34
provide flow diagrams that explain details of the graphics
24 data transfers that occur during each of the two frames

1 used to produce a single interlaced field.

Referring to the illustrative drawings of Figures 35 and 36, there is shown a single cell and a corresponding
5 graphics pattern for use in interlace mode. The single cell comprises 8 x 16 dots. In rendering the graphic pattern of Figure 25, rows 00, 08, 10, 18, 20, 28, 30 and 38 are rendered during the even trace frame, and the intervening rows 04, 0C, 14, 1C, 24, 2C 34, and 3C are
10 rendered during the odd trace frame. It will be understood that the number of rows of dots that appear in both the top screen and in the bottom screen in the split screen mode is the same as would appear on the entire screen in a non-interlace mode. There are 224 rows of
15 dots in each split screen. However, due to the fact that in interlace mode, dots are closer together, the images in the top and bottom split screens appear to be vertically squeezed. This is true for both scrolled playfields and for the sprites.

20

**Overview Of The Multiple Player Cooperative Mode
And Multiple Player Competition Mode**

24 In a current embodiment, a system in accordance with

1 the present invention has two modes of operation: a
cooperative mode and a competitive mode. In the
cooperative mode, the first character responds to inputs
applied to the first controller, and the second character
5 follows the first character through the playfield. It can
be said that the two game characters cooperate in that the
first character leads the second character through the
playfield. Not only does the second character follow the
first character, but the second character also imitates
10 the first character's movements. Thus, the first and
second game characters both respond to inputs provided to
the first controller with the second character following
behind and seeming to imitate the first character.

15 Figures 37A-E are purely illustrative drawings used
to explain the operation of the cooperative and
competition modes. Each drawing shows a portion of a
prescribed path through the playfield. As can be seen,
the path includes hills and valleys. At one point, there
20 is a trench. A platform (an "event") moves back and forth
across the trench, as indicated by the arrows. In order
for a game character to traverse the trench, the character
must jump onto the platform and ride the platform across
24 the trench and then jump off the platform on the other

1 side of the trench. Jumping on to and off ~~the~~ the platform
requires skill. This is the type of challenge that might
cause difficulty for a novice player.

5

In Figures 37A-E, the triangle character is the first
character controlled by the first controller, and the
circle character is the second character which follows the
first character in the cooperative mode. In Figure 37A,
10 the second character is following behind the first
character. The box in Figure 37A encloses the portion of
the path which is then currently visible on the display
screen. Thus, portions of the path to the right and to
the left of the box are outside the screen display and are
15 not visible to the game players.

In Figure 37B, the first and second characters have
progressed to the right, and consequently, the portion of
the path that is visible on-screen has changed. Also, the
20 first character is shown to be jumping. In Figure 37C,
the first and second characters have progressed even
further to the right as indicated by the location of the
screen display on the playfield. In Figure 37C, it can be
24 seen that the second character is jumping just as the

1 first character jumped previously. In cooperative mode,
the jumping of the second character is responsive to the
same jump command provided to the first controller that
caused the first character to jump in Figure 37B. In
5 Figure 37D, the first character is shown to have
successfully traversed the trench. The second character
is still on the left edge of the trench not having
traversed it yet.

10 At this point, assume that a second player using the
second controller has decided to begin control the
movement of the second character as it attempts to
traverse the trench. The second player provides an input
to the second controller in an attempt to cause the second
15 character to jump on to the platform. The game switches
to the competition mode. Suppose the jump fails, as a
result, in Figure 37E, the first character has continued
to progress, and the second character is left behind out
of view. This is indicated by the screen display in which
20 the first character is within the field of view, but the
second character (indicated by dotted lines) is outside
the field of view of the screen display. As a result, in
the present embodiment action is halted, and the second
24 character once again appears within the screen display

1 located directly above the first character and begins to
move downward as described below in relation to the
computer program of Figure 40.

5 In the competitive mode, each game character is under
separate control of a different controller. The first
game character responds to inputs applied to the first
controller 72, and the second game character responds to
inputs applied tot he second controller 74. The two game
10 characters compete as they traverse the playfield.

In certain game environments, the distance that a
game character progresses through a playfield depends upon
the skill of the player operating a controller used to
15 control the game character. A character controlled by a
skilled player will progress farther because a skilled
player can better overcome obstacles in the path of a
character that he controls. As explained in the
cooperative mode control section, the cooperative mode
20 permits a novice player to take advantage of the skills of
a more seasoned first player. The accomplished player can
control the first character which, due to his skill, is
able to make significant progress through the playfield.
24 The second character also makes significant progress by

1 following the first character and imitating its movements.

As explained below in the competition mode control and mode switching sections, the novice player can take
5 active control of the second character by pressing buttons on the second controller. The system then operates in the competition mode in which each character is controlled by a separate controller. However, in the preferred embodiment, if the second player fails to operate the
10 second controller for a prescribed period of time, such as ten seconds, then the system reverts again to the cooperative mode, and the second character once again follows the first character. Alternatively, if the second player continues to operate the second character controls,
15 but the second character falls a prescribed distance behind the first character, such as so far behind as to not be visible on the screen, then the system will cause the second character to catch up with the first character, and the system will again revert to the cooperative mode.

20

Cooperative Mode Control

Referring to Figure 38, there is shown a flow diagram for a computer program used to control the movement of the
24 second game character in the cooperative mode. The flow

1 diagram assumes that a first player has provided a series
of commands to the first controller in order to control
the movement of the first character through the playfield.
In step 510, the first controller input data is read for
5 that input data which was stored 16/60 seconds before. In
step 512 a determination is made as to whether or not the
stored input data indicates that the right button was
activated. If it was then in step 514 of the character
second sprite moves to the right on the screen display.
10 In step 516 a determination is made as to whether the
input data indicates that the left button was activated.
If so, then in step 518 the character sprite moves to the
left. In step 520 a determination is made as to whether
the jump button was activated. If it was, then in step
15 522 the second character sprite jumps on-screen. In step
524, a determination is made as to whether or not the down
button was activated. If it was, then in step 526 the
character sprite is caused to crouch down. In step 528, a
determination is made as to whether or not there is an
20 object obstructing the path of the second character
sprite. If there is, then in step 530 the second sprite
attempts to jump over the object. That way, the second
character tries to overcome obstacles in its path as it
24 pursues the first character even if no first controller

1 input was applied for that specific purpose.

5 In step 532 a determination is made as to whether or
not there is any distance between the first and second
characters. If there is, then the second character moves
toward the first character dot by dot. The computer
program represented by Figure 38 is called every 1/60
seconds by an interrupt routine. The second character
typically only has an opportunity to move a few dots
10 before the routine is called again. Thus, before the
second character can move very far in its effort to catch
up with the first character, a determination is made as to
whether there are additional first character movements to
be imitated by the second character.

15

The second character sprite appears to follow the
first character sprite, mimicking his moves. If however,
the first character sprite remains stationary, the second
character sprite, through the operation of step 534 will
20 catch up with him. The result is that the first character
leads the way through the playfield under control of the
first controller. The second character follows close
enough behind so that both the first and second characters
24 are visible on the screen display. If the first character

1 fails to move, then the second character moves toward the
first character, either to the right or to the left as
necessary. This routine runs periodically when the system
is in the cooperative mode.

5

Competition Mode Control

In the competition mode, the first controller 72 is
used to control the movement of the first character
sprite, and the second controller 74 is used to control
10 the movement of the second character sprite. The players
operating the first and second controllers compete with
each other as the first and second characters traverse the
playfield. The competition, for example, can involve
attempts to traverse the playfield the fastest while
15 accumulating the most points.

Mode Switching

Referring to the illustrative drawing of Figure 39,
there is shown a flow diagram for a computer program used
20 to determine whether the cooperative mode or the
competition mode is to be used. In step 540, a
determination is made as to whether the game has been set
to be controlled by only one player. If it has, then the
24 game is automatically put into the cooperative mode, and

1 in step 542 a determination is made as to whether there
has been a control input applied to the second controller.
If not, then in step 544 a determination is made as to
whether there has been no input from the second controller
5 for more than ten seconds. If there has been no input for
more than ten seconds, then the game remains in the
cooperative mode as indicated by
step 546.

If in step 542 there has been a control input to the
10 second controller then, in step 548, a determination is
made as to whether or not the second character sprite has
disappeared from the screen. If it has, then the second
character sprite is caused to return into view on the
screen in step 550 and the game remains in the cooperative
15 mode. If in step 548 it is determined that the second
character sprite has not disappeared from the screen, then
in step 552 the second character sprite is placed under
control of the second controller and the game reverts to
the competitive mode.

20

The program of Figure 39 periodically executes. The
result is that the game will revert to the cooperative
mode if the second character falls sufficiently behind the
24 first character to disappear from the screen or the second

1 player fails to operate the second controller for a
prescribed time period. Conversely, the game will revert
to the competitive mode if the second player operates the
second controller and the second character is still
5 visible on the screen.

The flow diagram of Figure 40 represents a computer
program used to move the second character sprite back into
view after it has fallen far behind the first sprite and
10 has disappeared from the screen. The program of Figure 40
is used in the competitive mode to be sure that the first
and second character sprites remain visible together on
the screen display. The objective is to not allow the
character controlled by the novice player to fall too far
15 behind the main character.

In step 560 a determination is made as to whether or
not the initial flag is set to one. The initial flag
indicates that the subsequent four steps already have been
20 taken. The initial flag is part of the sprite status
information for the second character sprite. In step 562,
an image pattern is selected with the second character
sprite shown to be flying through the air. In step 564,
24 the horizontal position of the second sprite is selected

1 to match that of the first sprite. Typically the
controller 40 keeps the first character sprite centered on
the screen. Thus, the second character also is
horizontally centered on the screen. The first character
5 is frozen on the screen, and all screen action is frozen
except for the movement of the second character. In step
566, the vertical position of the second character sprite
is selected to be 192 dots above the first player. In
step 568, the initial flag is set to one, and in step 570
10 the second character begins to move slowly downward. In
step 572, a determination is made as to whether or not the
second character has landed on a game path. If he has not
then the cycle repeats until he lands, and in step 574,
the second character is displayed in a standing position.
15 At this point, the program illustrated in Figure 38 begins
running, and the steps illustrated in that program are
followed. The program in Figure 39 controls switching
between the cooperative mode and the competitive mode.

20 Overview of Playfield Position Exchange Graphics

The present invention includes a split screen mode of
operation described above. The screen display is split
into two equal sized screen displays, one above the other
24 in which game characters can compete against each other.

1 Each of the two screens can display different regions of
the same playfield. A top screen display displays a first
game character. A bottom screen display displays a second
game character. The first character is controlled by a
5 first user who operates a first controller. The second
character is controlled by a second user who operates a
second controller. The top and bottom screens can display
different regions of the playfield. The top screen
display scrolls independently of the bottom screen
10 display.

In certain video games, for example, competing
players operate different controllers which control the
movement of different characters through a playfield. Two
15 players may compete by trying to have their characters
race through the playfield trying to accumulate points by
surmounting obstacles, slaying enemies or gathering
"magical rings", for example. The player whose character
crosses the largest chunk of playfield or accumulates the
20 most points wins.

Referring to Figure 2, for example, the top screen
may display the playfield region labeled A, and the bottom
24 screen may display the playfield region labeled B. In

1 this example, regions A and B overlap. Referring to the
illustrative drawings of Figure 41, there is shown a split
screen display with the top screen illustrating a first
character in playfield region A and a bottom screen region
5 with a second character in playfield region B. Within the
playfield there is an exchange object which we will refer
to as a teleport box. When one of the character sprites
comes to within a prescribed proximity of the exchange
object, the positions of the two characters in the
10 playfield and much of their status information is
exchanged. In this manner, a character that has fallen
behind can interchange playfield positions with another
character. The exchange object, therefore, adds new
strategy considerations to the game competition.

15

In Figure 41, the first character in the top screen
has fallen behind the second character in the bottom
screen. This is apparent from the terrain in the two
screen displays. The characters move left to right as
20 they progress through the playfield. In the top screen,
the first character has not yet crossed the small hill.
In the bottom screen, the second character has just
crossed the same hill. An exchange object is present in
24 the top screen. When the first character contacts it, as

1 described below in connection with the computer program
flow diagram of Figures 43A-B, the playfield positions of
the first and second characters are reversed as shown in
Figure 42.

5

Exchange Object Control

Referring to Figures 43A-B, in step 610 a
determination is made as to whether either game character
has accessed an exchange object. If so, then an exchange
10 is made of certain current status information for the
first and second characters in step 612. Not all status
information is exchanged, and for example, the information
exchanged does not include information that is unique to
either character such as the graphic patterns that
15 determine its appearance. For example, the sprite pattern
numbers which represents the sprite animation are not
exchanged. During step 612, character sprite status
information is actually transferred between a first
character status information buffer in RAM 42 and a second
20 character status information buffer in the RAM 42.

In step 614 the master number and pattern number for
each player is initialized so that the first and second
24 characters will be shown in standing positions following

1 the exchange. In step 616 scroll pattern information is
shifted between the top screen scroll buffer and the
bottom screen scroll buffer. In step 618 the action set
numbers are exchanged between a top screen scroll buffer
5 and a bottom screen scroll buffer. The action set numbers
represent the screens actually stored in RAM 42 at any
given time. It will be recalled that in the present
embodiment a playfield is sixty screens horizontally and
eight screens vertically as shown in Figure 2. Each
10 playfield has a number of regions which are identified by
action set numbers. A playfield region designated by an
action set number is larger than a portion of a playfield
that can be displayed on a single split or whole screen at
any given moment. In a present embodiment, multiple
15 action set number regions are stored in RAM 42 at any
given time, although only one screen at a time is
displayed. This additional playfield display information
is stored in RAM 42 so as to accommodate rapid character
movements or changes in direction. For example, if the
20 character suddenly moves to a new playfield region that
previously had not been displayed, then new playfield
region display information will be readily available in
the RAM 42. There will be no need to transfer new
24 playfield display information from the ROM 58 to the RAM

1 42. As a consequence, there will be no perceptible delay
or discontinuity as the character moves to the new
playfield region.

5 In step 620 priority information is exchanged between
the characters. In step 622, flags are adjusted for
certain events. More specifically, there are events such
as the passage of a character past an A/B collision switch
which causes a change in the character's collision
10 information. There also are events such as the character
having entered a region of the playfield which requires
the character to take a special appearance. For example,
in a present embodiment, one of the screens is known as
"casino night". While the first character is in the
15 casino night screen it takes the shape of a pinball since
casino night is in the form of pinball machine. Thus,
pattern information used to produce a pinball image is
used to depict the character as it traverses the casino
night screen. The flags that are set in step 622 are the
20 flags that keep track of such special or special events.
One of ordinary skill in the art will appreciate that
there may be other special event flags to be set depending
upon the idiosyncrasies of a particular playfield or video
24 game.

1

In step 624, modes known as barrier and invincible are set for the characters. In a present embodiment, the setting of these modes for one of the characters calls a routine which causes a character to be displayed with a certain graphic which depicts the character as being invincible. When a character is in the invincible mode, for example, it cannot be as readily "killed". In step 626, a determination is made as to whether the first character is spinning. If it is, and then the first character hit size is decreased in step 628. In step 630 a determination is made as to whether the second character is spinning. If so, then in 632 the second player's hit size is decreased. Once again, one of ordinary skill in the art will appreciate that there may be other special modes depending upon the particular features of the game.

Referring to Figure 43B in step 634, a determination is made as to whether or not a character is riding on an "event". In a present embodiment, an event is a term that designates a playfield object or obstacle such as a moving platform or a shuttle loop. When a character rides on an event, then special consideration must be made for character operation during that time period. For example,

1 if a character is crossing a shuttle loop then its control
will depend upon offset table entries rather than
collision information.

5 If the first character is riding on an event, then in
step 636 the first character ride flag is set. In step
638 a determination is made as to whether or not the
second character is riding on an event. If so, then in
step 640 the second character's ride flag is set. In step
10 642 a flag is set to inactivate the TV screen display
while the scroll fields are interchanged. In step 644 the
set action stop flags are set for both characters so that
both characters pause. In step 646 an interrupt wait is
set for a count of sixty-three. In steps 648 and 650 the
15 count is decremented, and in 652, when the count has
completed and the scroll screens have been updated, the TV
screen display is turned back on. The TV screen display
is temporarily turned off both to accommodate the
information exchange and to give the human players time to
20 realize that an exchange has just taken place. In step
654 the action flags for the two characters are once again
set.

24 Figure 44 illustrates a teleport box. In a present

1 embodiment, the teleport box is the exchange object
visible on screen. When a character contacts the box by
jumping on it, it "breaks" the box. Before the character
contacts it, the box appears as in Figure 44 after the
5 character contacts it and "breaks" it, it appears as in
figure 45. The box center floats upward until it reaches
a certain height. When it reaches that height, the
exchange described above takes place. Thus, game players
know when an exchange is about to occur.

10

While a particular embodiment of the invention has
been described in detail, various modifications to the
preferred embodiment can be made without departing from
the spirit and scope of the invention. For example, the
15 shuttle loop might define a path that twist through more
or less than 360 degrees. Thus, the invention is limited
only by the appended claims.

Industrial Advantages

20

As described above, the present invention meets the
need for improved graphics techniques which display
banked, upside down or sideways path segments. And the
present invention highly improves the play and reality of
24 the video game.

CLAIMS:

1. A method for displaying a video game character traversing a video game playfield, for use with a system which includes a video screen display, a user-controlled graphics controller and digital memory; wherein the playfield is displayed as a series of scrolled screen displays; and wherein the video game character follows a path within the playfield; and wherein progress of the game character in traversing the playfield is indicated by scrolling the playfield relative to the game character, the method comprising the steps of:

storing in the digital memory multiple collision blocks that define respective path segments;

dividing the playfield into multiple path blocks that comprise the path;

storing in the digital memory character collision type information which corresponds to a character and which indicates for that character either a first character collision type or a second character collision type;

storing in the digital memory references from individual path blocks to individual collision blocks wherein for at least one path block, the reference is dependent upon the character collision type information for a character traversing that at least one path block;

displaying character movement through the playfield from path block to path block along the path in response to user input to the graphics controller;

controlling the display of character movement from path block to path block by causing a character image to follow a path defined by the path segments of individual collision blocks referenced to individual path blocks that comprise the path; and

changing the stored character collision type information when the character passes a prescribed location on the playfield such that after the change, the stored character collision type information indicates a different character collision type than before the change.

2. The method of claim 1 and further including the steps of:

displaying two path segment images in single path block;

wherein there is stored for that single path block a reference to one collision block for use when a character traverses that single path block as first collision type information; and

wherein there is stored for that single path block a reference to another collision block for use when a character traverses that single path block as second collision type information.

3. The method of claim 1 and further including the step of:
displaying path segment images in the path blocks wherein the path segment images correspond to path segments of the collision blocks referenced by the path blocks that comprise the path.
4. The method of claim 1 and further including the step of:
displaying path segment images in the path blocks; and
displaying two path segment images in a path block for which the reference is dependent upon the character collision type information for a character traversing that path block.
5. The method of claim 1 and further including the step of:
displaying path segment images in the path blocks; and
displaying two overlapping path segment images in a path block for which the reference is dependent upon the character collision type information of a character traversing that path block.
6. The method of claim 1 and further including the steps of:
displaying path segment images in the path blocks; and
displaying two overlapping path segment images in a path block for which the reference is dependent upon the character collision type information of a character traversing that path block;
wherein said step of changing the character collision type occurs after the character traverses the at least one path block.
7. The method of claim 1:

wherein said step of changing the character collision type information occurs after the character has traversed a path block for which the reference is dependent upon the character collision type information.

8. The method of claim 1 wherein the prescribed location is in a prescribed path block.

9. The method of claim 1 wherein the prescribed location is in a prescribed path block; and wherein references are stored for the prescribed block which are dependent upon character collision type information of a character traversing that prescribed path block.

10. A method for displaying a video game character traversing a video game playfield, for use with a system which includes a video screen display, a user-controlled graphics controller and digital memory; wherein the playfield is displayed as a series of scrolled screen displays; and wherein the video game character follows a path within the playfield; and wherein progress of the game character in traversing the playfield is indicated by scrolling the playfield relative to the game character, the method comprising the steps of:

storing in the digital memory multiple collision blocks that define respective path segments;

dividing the playfield into multiple path blocks that comprise the path;

displaying path segment images within path blocks wherein the displayed path segment images depict a loop which overlaps itself and wherein a multi-path path block includes two path segments images;

storing in the digital memory character collision type information which corresponds to a character and which indicates for that character either a first character collision type or a second character collision type;

storing in the digital memory references from individual path blocks to individual collision blocks wherein the reference for the multi-path path block is dependent upon the character collision type information for a character traversing that multi-path path block;

displaying character movement through the playfield from path block to path block along the path in response to user input to the graphics controller;

controlling the display of character movement from path block to path block by causing a character image to follow a path defined by the path segments of individual collision blocks referenced to individual path blocks that comprise the path;

changing the stored character collision type information when the character passes a prescribed location on the playfield such that after the change, the stored character collision type information indicates a different character collision type than before the change.

11. The method of claim 10 wherein the prescribed location is within a path block that includes a path segment image which serves as a portion of the loop.

12. The method of claim 10 wherein the prescribed location is near an apex of the loop.

13. A computer readable medium having stored thereon statements and instructions for execution by a computer of a method of displaying a video game character traversing a video game playfield, for use with a system which includes a video screen display, a user-controlled graphics controller and digital memory; wherein the playfield is displayed as a series of scrolled screen displays; and wherein the video game character follows a path within the playfield; and wherein progress of the game character in traversing the playfield is indicated by scrolling the playfield relative to the game character, the method comprising the steps of:

storing in the digital memory multiple collision blocks that define respective path segments;

dividing the playfield into multiple path blocks that comprise the path;

storing in the digital memory character collision type information which corresponds to a character and which indicates for that character either a first character collision type or a second character collision type;

storing in the digital memory references from individual path blocks to individual collision blocks wherein for at least one path block, the reference is dependent upon the character collision type information for a character traversing that at least one path block;

displaying character movement through the playfield from path block to path block along the path in response to user input to the graphics controller;

controlling the display of character movement from path block to path block by causing a character image to follow a path defined by the path segments of individual collision blocks referenced to individual path blocks that comprise the path; and

changing the stored character collision type information when the character passes a prescribed location on the playfield such that after the change, the stored character collision type information indicates a different character collision type than before the change.

14. The computer readable medium of claim 13, wherein the method further includes the steps of:

displaying two path segment images in single path block;

wherein there is stored for that single path block a reference to one collision block for use when a character traverses that single path block as first collision type information; and

wherein there is stored for that single path block a reference to another collision block for use when a character traverses that single path block as second collision type information.

15. The computer readable medium of claim 13, wherein the method further includes the step of:

displaying path segment images in the path blocks wherein the path segment images correspond to path segments of the collision blocks referenced by the path blocks that comprise the path.

16. The computer readable medium of claim 13, wherein the method further includes the steps of:

displaying path segment images in the path blocks; and

displaying two path segment images in a path block for which the reference is dependent upon the character collision type information for a character traversing that path block.

17. The computer readable medium of claim 13, wherein the method further includes the steps of:

displaying path segment images in the path blocks; and

displaying two overlapping path segment images in a path block for which the reference is dependent upon the character collision type information of a character traversing that path block.

18. The computer readable medium of claim 13, wherein the method further includes the steps of:

displaying path segment images in the path blocks; and

displaying two overlapping path segment images in a path block for which the reference is dependent upon the character collision type information of a character traversing that path block;

wherein said step of changing the character collision type occurs after the character traverses the at least one path block.

19. The computer readable medium of claim 13:

wherein said step of changing the character collision type information occurs after the character has traversed a path block for which the reference is dependent upon the character collision type information.

20. The computer readable medium of claim 13 wherein the prescribed location is in a prescribed path block.

21. The computer readable medium of claim 13 wherein the prescribed location is in a prescribed path block; and wherein references are stored for the prescribed block which are dependent upon character collision type information of a character traversing that prescribed path block.

22. A computer readable medium having stored thereon statements and instructions for execution by a computer of a method for displaying a video game character traversing a

video game playfield, for use with a system which includes a video screen display, a user-controlled graphics controller and digital memory; wherein the playfield is displayed as a series of scrolled screen displays; and wherein the video game character follows a path within the playfield; and wherein progress of the game character in traversing the playfield is indicated by scrolling the playfield relative to the game character, the method comprising the steps of:

storing in the digital memory multiple collision blocks that define respective path segments;

dividing the playfield into multiple path blocks that comprise the path;

displaying path segment images within path blocks wherein the displayed path segment images depict a loop which overlaps itself and wherein a multi-path path block includes two path segments images;

storing in the digital memory character collision type information which corresponds to a character and which indicates for that character either a first character collision type or a second character collision type;

storing in the digital memory references from individual path blocks to individual collision blocks wherein the reference for the multi-path path block is dependent upon the character collision type information for a character traversing that multi-path path block;

displaying character movement through the playfield from path block to path block along the path in response to user input to the graphics controller;

controlling the display of character movement from path block to path block by causing a character image to follow a path defined by the path segments of individual collision blocks referenced to individual path blocks that comprise the path;

changing the stored character collision type information when the character passes a prescribed location on the playfield such that after the change, the stored character collision type information indicates a different character collision type than before the change.

23. The computer readable medium of claim 22 wherein the prescribed location is within a path block that includes a path segment image which serves as a portion of the loop.

24. The computer readable medium of claim 22 wherein the prescribed location is near an apex of the loop.

25. A video game system for displaying a video game character traversing a video game playfield comprising,

at least one digital memory for:

storing multiple collision blocks that define respective path segments of a path within the playfield;

storing character collision type information which corresponds to a character and which indicates for that character either a first character collision type or a second character collision type; and

storing references from individual path blocks to individual collision blocks wherein for at least one path block, the reference is dependent upon the character collision type information for a character traversing that at least one path block,

a user-controlled graphics controller, in communication with the at least one digital memory, for:

dividing the playfield into multiple path blocks that comprise the path;

controlling the display of character movement from path block to path block by causing a character image to follow a path defined by path segments of individual collision blocks referenced to individual path blocks that comprise the path; and

changing the stored character collision type information when the character passes a prescribed location on the playfield such that after the change, the stored character collision type information indicates a different character collision type than before the change, and

a video screen display, in communication with the user-controlled graphics controller, for:

displaying character movement through the playfield from path block to path block along the path in response to user input to the graphics controller.

26. The system of claim 25, wherein the video screen display displays two path segment images in a single path block, and wherein there is stored in the at least one digital memory a reference to one collision block for use when a character traverses that

single path block as first collision type information; and wherein there is stored in the at least one digital memory a reference to another collision block for use when a character traverses that single path block as second collision type information.

27. The system of claim 25, wherein the video screen display displays path segment images in the path blocks wherein the path segment images correspond to path segments of the collision blocks referenced by the path blocks that comprise the path.

28. The system of claim 25, wherein the video screen display displays:
 path segment images in the path blocks; and
 two path segment images in a path block for which the reference is dependent upon the character collision type information for a character traversing that path block.

29. The system of claim 25, wherein the video screen display displays:
 path segment images in the path blocks; and
 two overlapping path segment images in a path block for which the reference is dependent upon the character collision type information of a character traversing that path block.

30. The system of claim 25, wherein the video screen display displays:
 path segment images in the path blocks; and
 two overlapping path segment images in a path block for which the reference is dependent upon the character collision type information of a character traversing that path block;
 wherein the user-controlled graphics controller changes the character collision type after the character traverses the at least one path block.

31. The system of claim 25:
 wherein the user-controlled graphics controller changes the character collision type information after the character has traversed a path block for which the reference is dependent upon the character collision type information.

32. The system of claim 25 wherein the prescribed location is in a prescribed path block.

33. The system of claim 25 wherein the prescribed location is in a prescribed path block; and wherein the at least one digital memory stores references for the prescribed block which are dependent upon character collision type information of a character traversing that prescribed path block.

34. A video game system for displaying a video game character traversing a video game playfield comprising,

at least one digital memory for:

storing multiple collision blocks that define respective path segments;

storing character collision type information which corresponds to a character and which indicates for that character either a first character collision type or a second character collision type;

storing references from individual path blocks to individual collision blocks wherein a reference for a multi-path path block is dependent upon the character collision type information for a character traversing that multi-path path block,

a user-controlled graphics controller, in communication with the at least one digital memory for :

dividing the playfield into multiple path blocks that comprise a path;

controlling the display of character movement from path block to path block by causing a character image to follow a path defined by the path segments of individual collision blocks referenced to individual path blocks that comprise the path;

changing the stored character collision type information when the character passes a prescribed location on the playfield such that after the change, the stored character collision type information indicates a different character collision type than before the change, and

a video screen display, in communication with the user controlled graphics controller, for:

displaying path segment images within path blocks wherein the displayed path segment images depict a loop which overlaps itself and wherein a multi-path path block includes two path segments images;

displaying character movement through the playfield from path block to path block along the path in response to user input to the graphics controller.

35. The system of claim 34 wherein the prescribed location is within a path block that includes a path segment image which serves as a portion of the loop.

36. The system of claim 34 wherein the prescribed location is near an apex of the loop.

37. The system of any one of claims 25 to 36 where the at least one digital memory is at least two digital memories.

38. The system of claim 38 where the at least two digital memories comprise at least one RAM digital memory and at least one ROM digital memory.

1/46

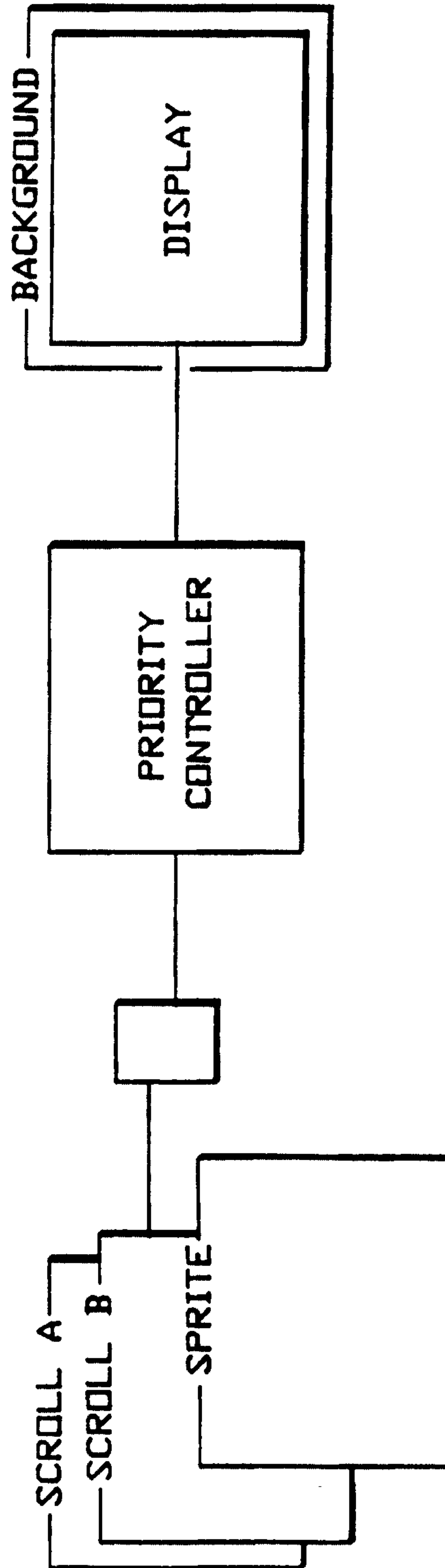


FIG. - 1

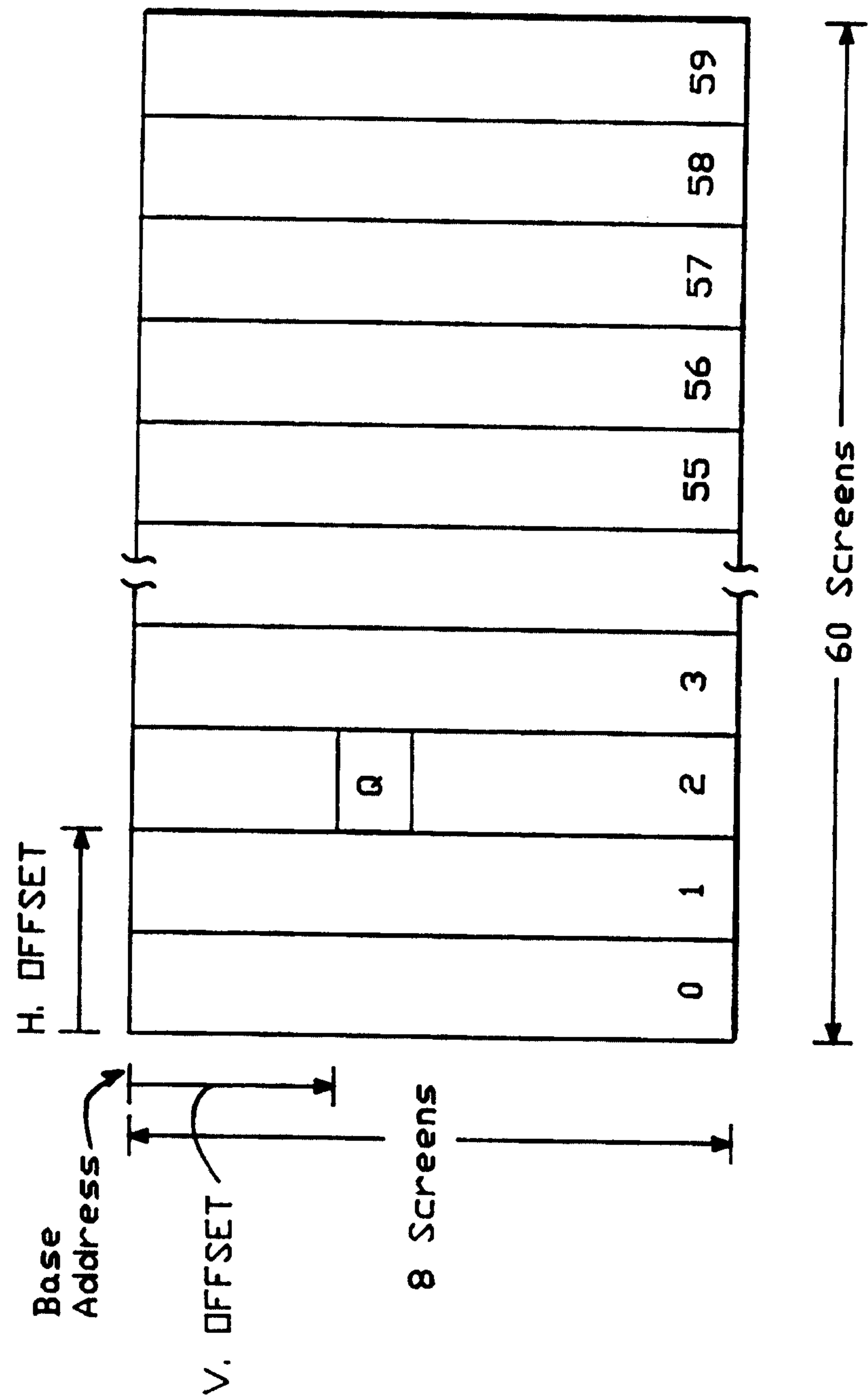


FIG. -2

3/46

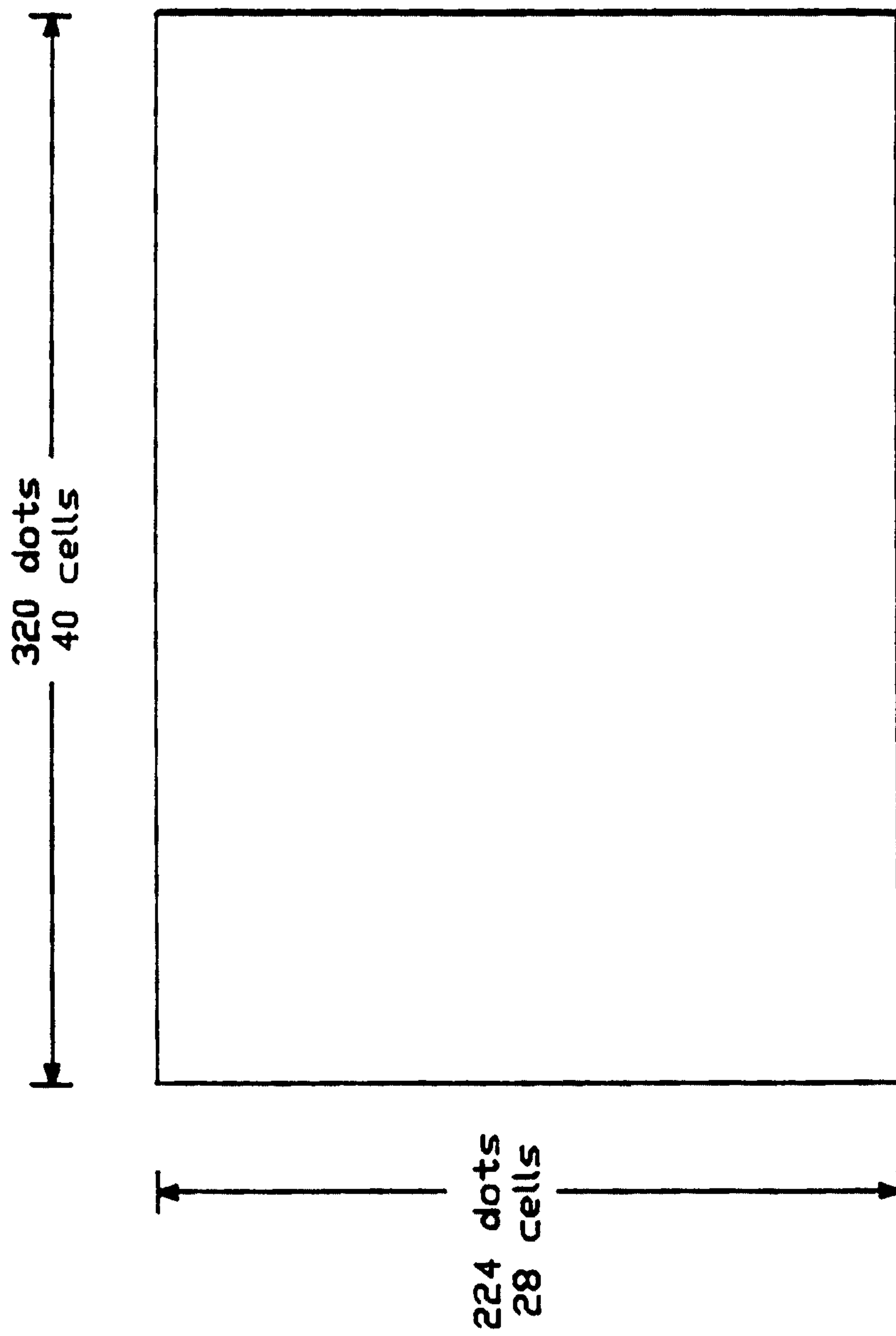


FIG. -3

4/46

	1	2	3	4	5	6	7	8
a								
b								
c								
d								
e								
f								
g								
h								

FIG.-4

	0		1		2		3	
	D7	D0	D7	D0	D7	D0	D7	D0
00	a1	a2	a3	a4	a5	a6	a7	a8
04	b1	b2	b3	b4	b5	b6	b7	b8
08	c1	c2	c3	c4	c5	c6	c7	c8
0C	d1	d2	d3	d4	d5	d6	d7	d8
10	e1	e2	e3	e4	e5	e6	e7	e8
14	f1	f2	f3	f4	f5	f6	f7	f8
18	g1	g2	g3	g4	g5	g6	g7	g8
1C	h1	h2	h3	h4	h5	h6	h7	h8

FIG.-5A

D7	D6	D5	D4	D3	D2	D1	D0
COL3	COL2	COL1	COL0	COL3	COL2	COL1	COL0

FIG.-5B

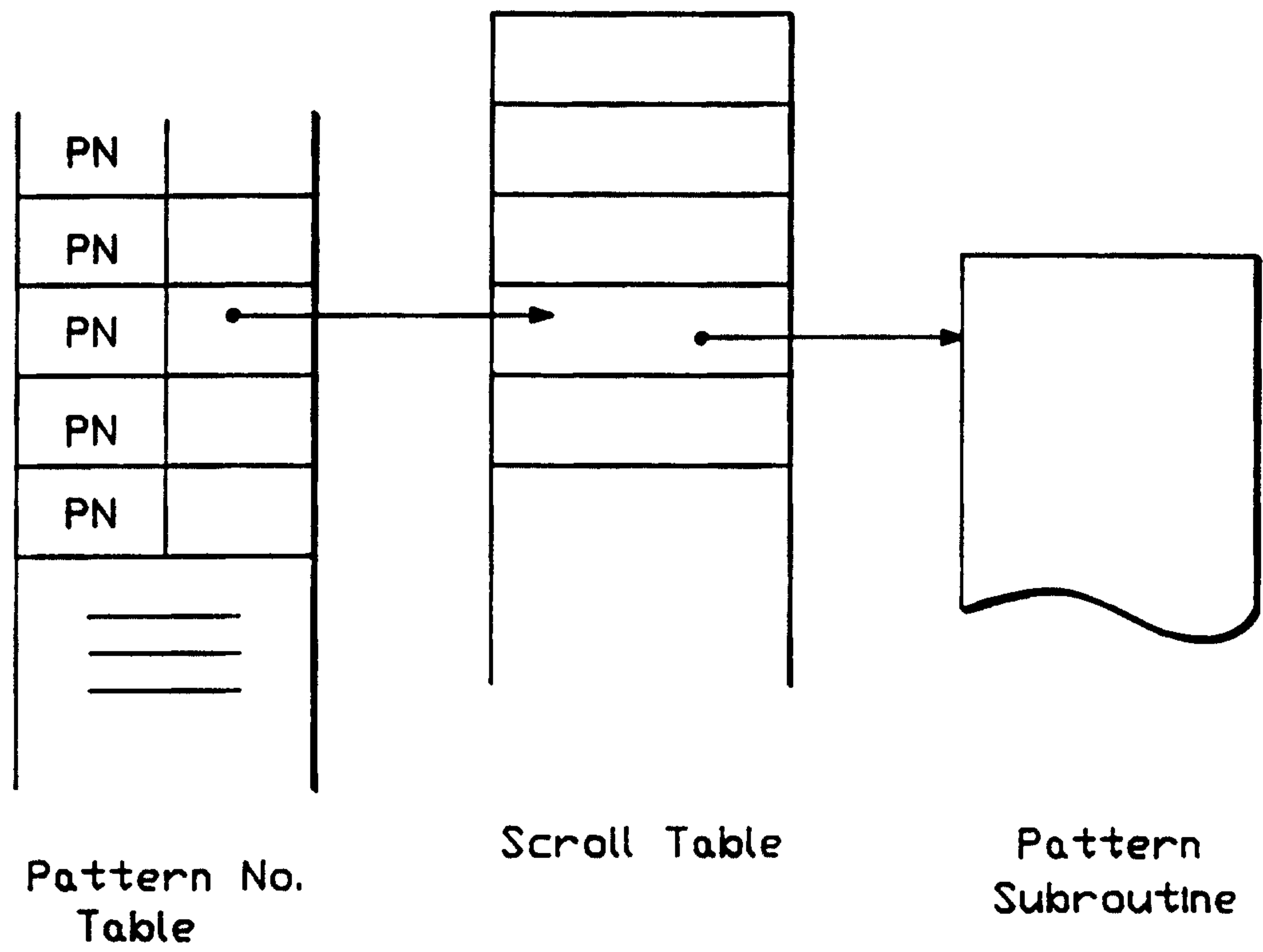


FIG.—6

6/46

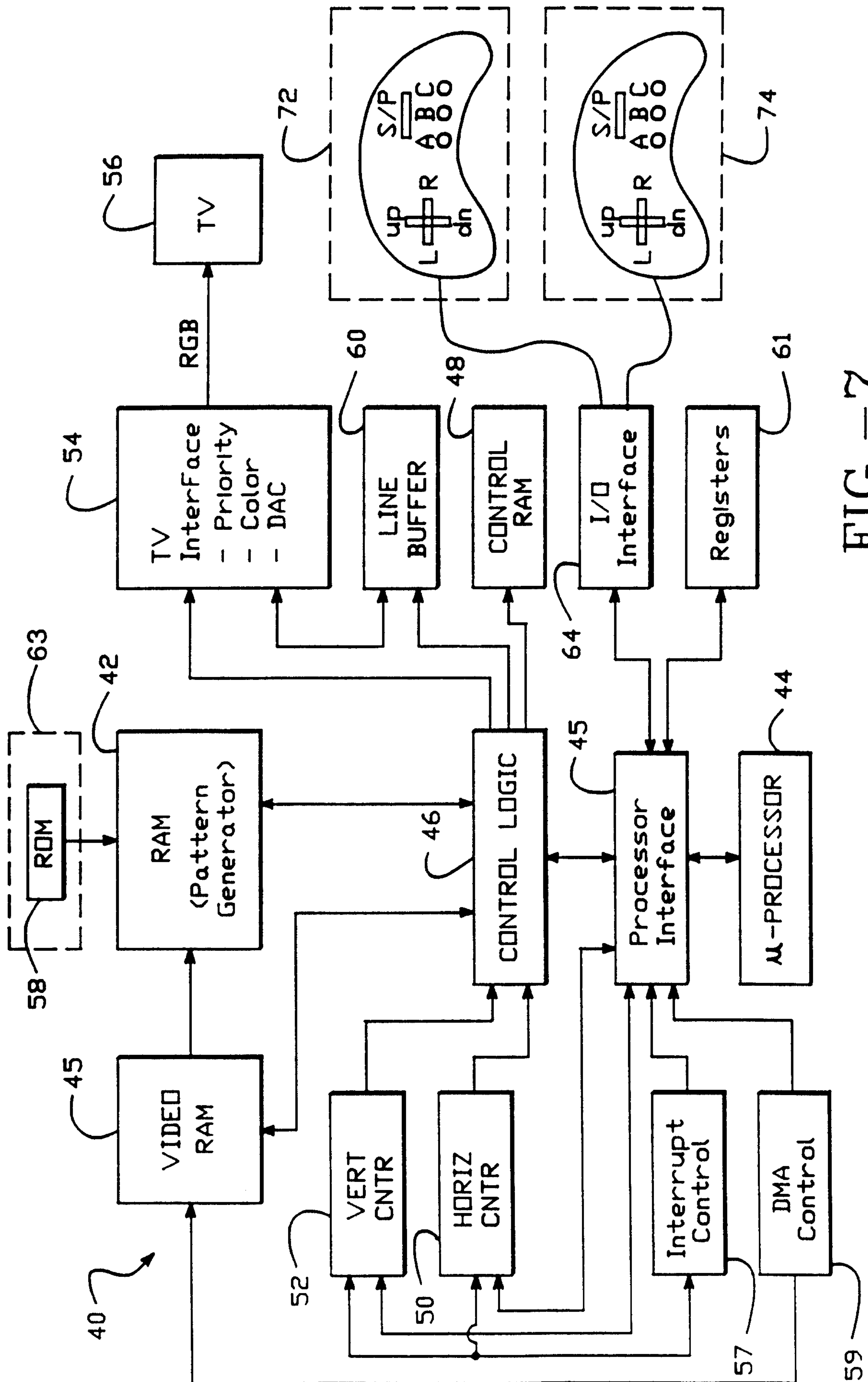


FIG. — 7

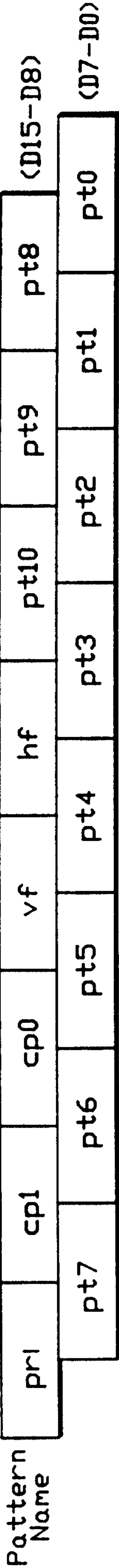


FIG.—8

8/46

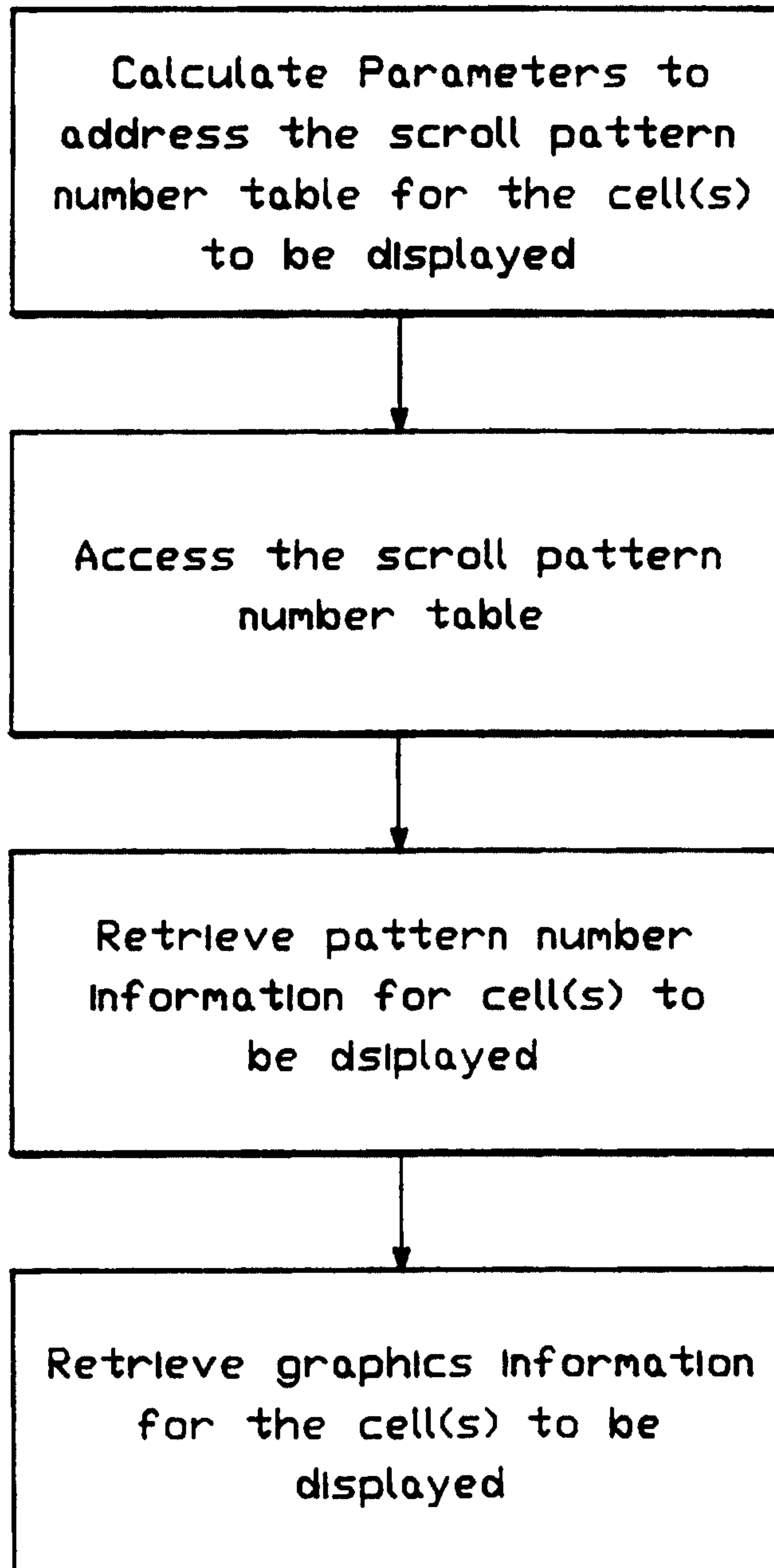


FIG.-9

VERTICAL POS'N	
Size	Link No.
Pattern No.	
Vertical Pos'n	

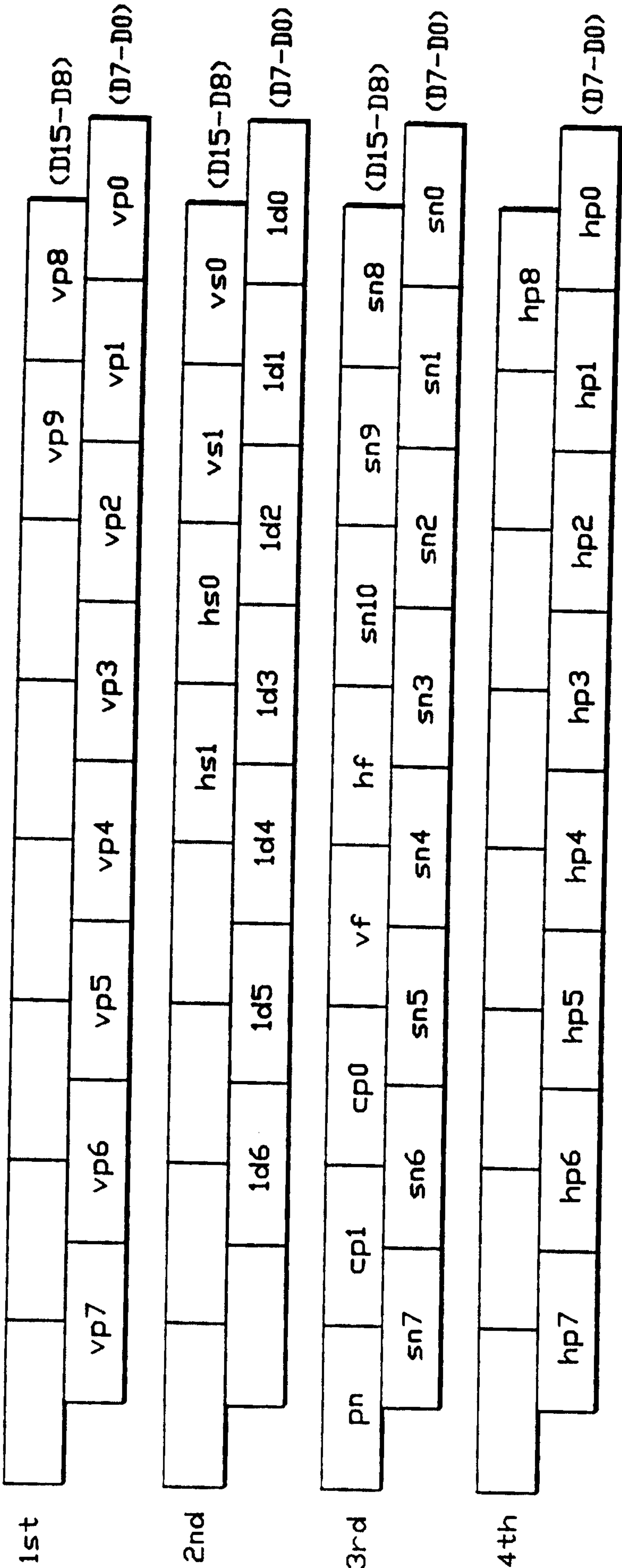
CONTROL RAM SPRITE TABLE

FIG.—10

2. SPRITE ATTRIBUTE

Sprite Attribute Table is on VRAM and the Base Address is designated by Register. Attribute requires 8 byte (4 words) per Sprite, and indicates Display Position, Priority,Sprite Generator Number, and Attribute.

Starting from the beginning of the Attribute Table, number are given in the sequence of Sprite 0, Sprite 1, Sprite 2, Sprite 3, etc. Priority between Sprites is not determined by the sequence of Sprite number, but by each Sprite's Link Data, and thus becomes programmable.



SPRITE ATTRIBUTE TABLE ENTRY

FIG. - 11

S prt	A prt	B prt	Priority
0	0	0	S>A>B
1	0	0	S>A>B
0	1	0	A>S>B
1	1	0	S>A>B
0	0	1	B>S>A
1	0	1	S>B>A
0	1	1	A>B>S
1	1	1	S>A>B

S: Sprite
A: Scroll A
B: Scroll B

FIG. - 12

12/46

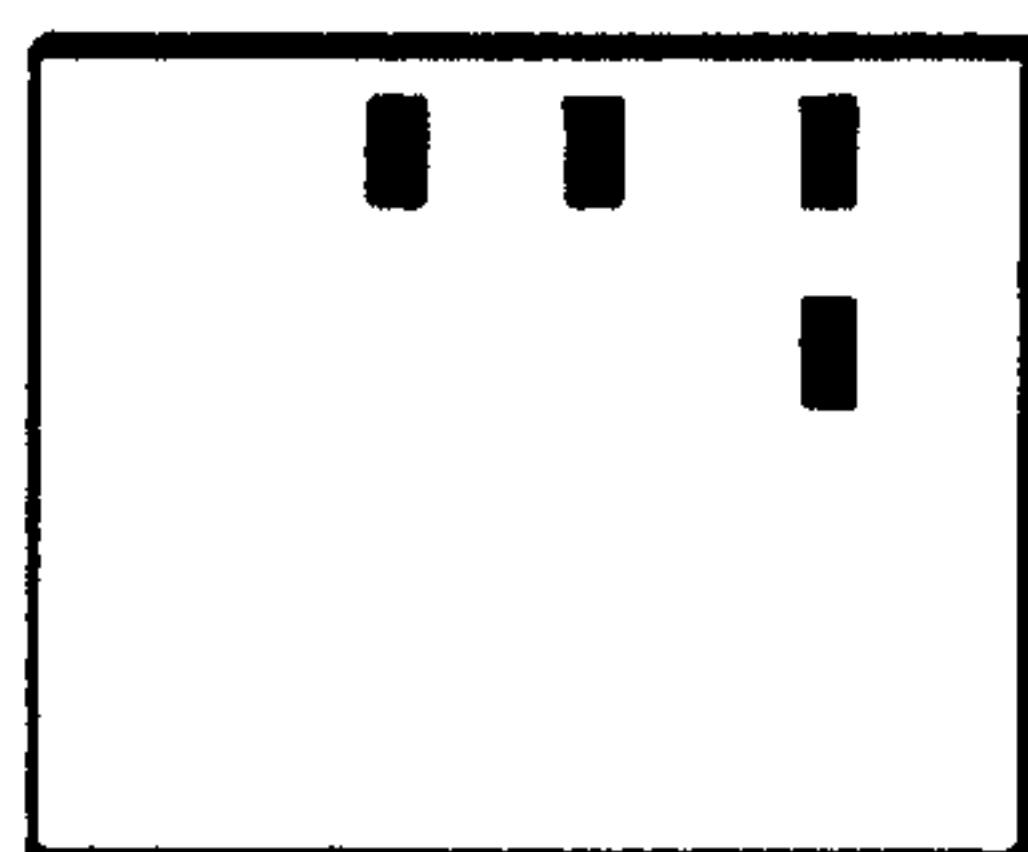
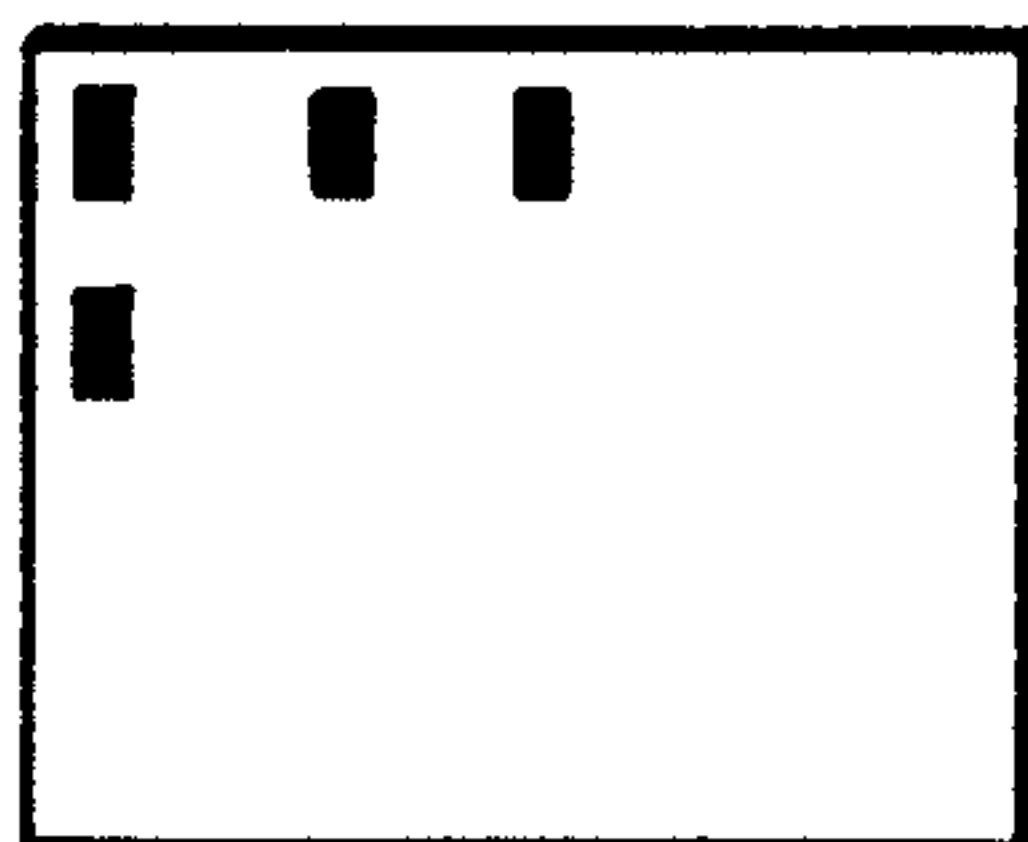
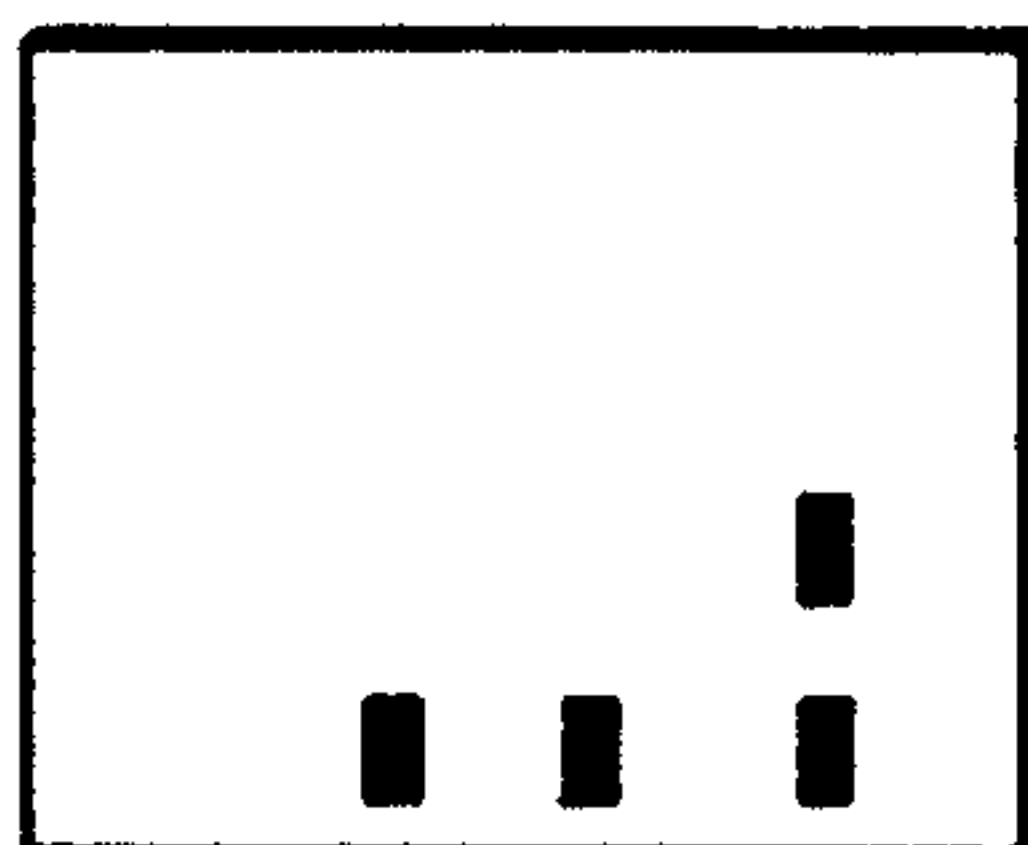
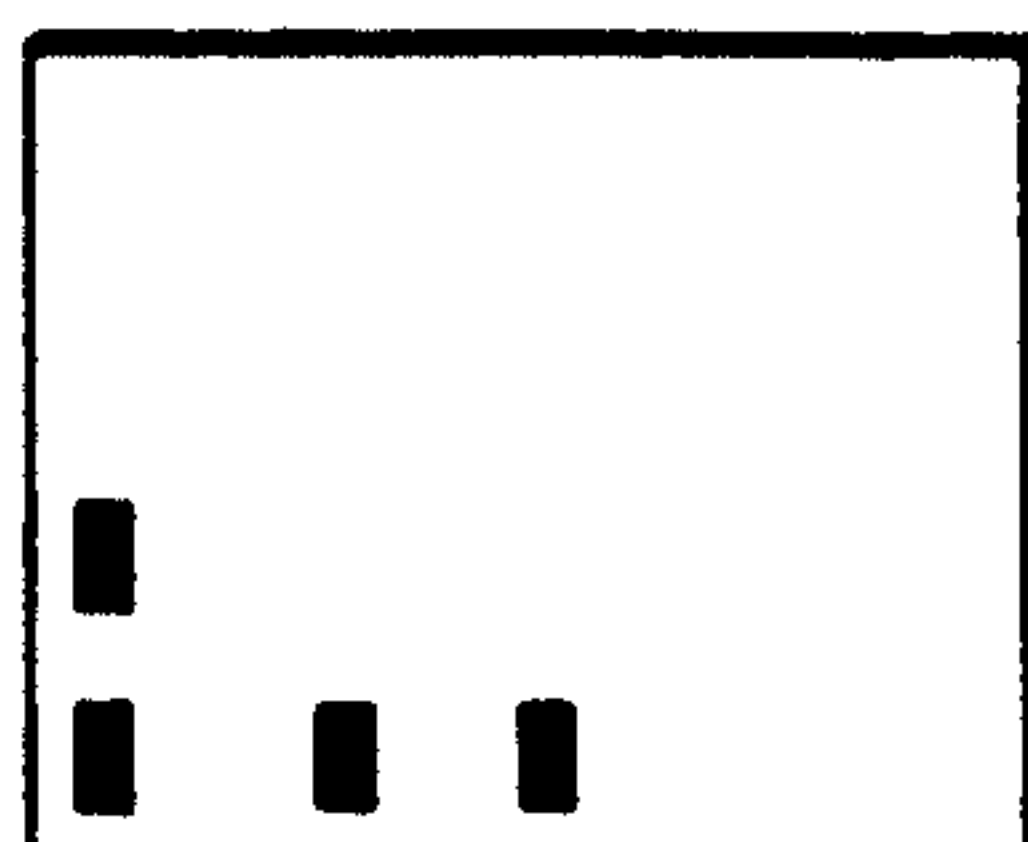
 $vf = 1$
 $hf = 1$  $vf = 0$
 $hf = 1$  $vf = 1$
 $hf = 0$  $vf = 0$
 $hf = 0$

FIG.-13

13/46

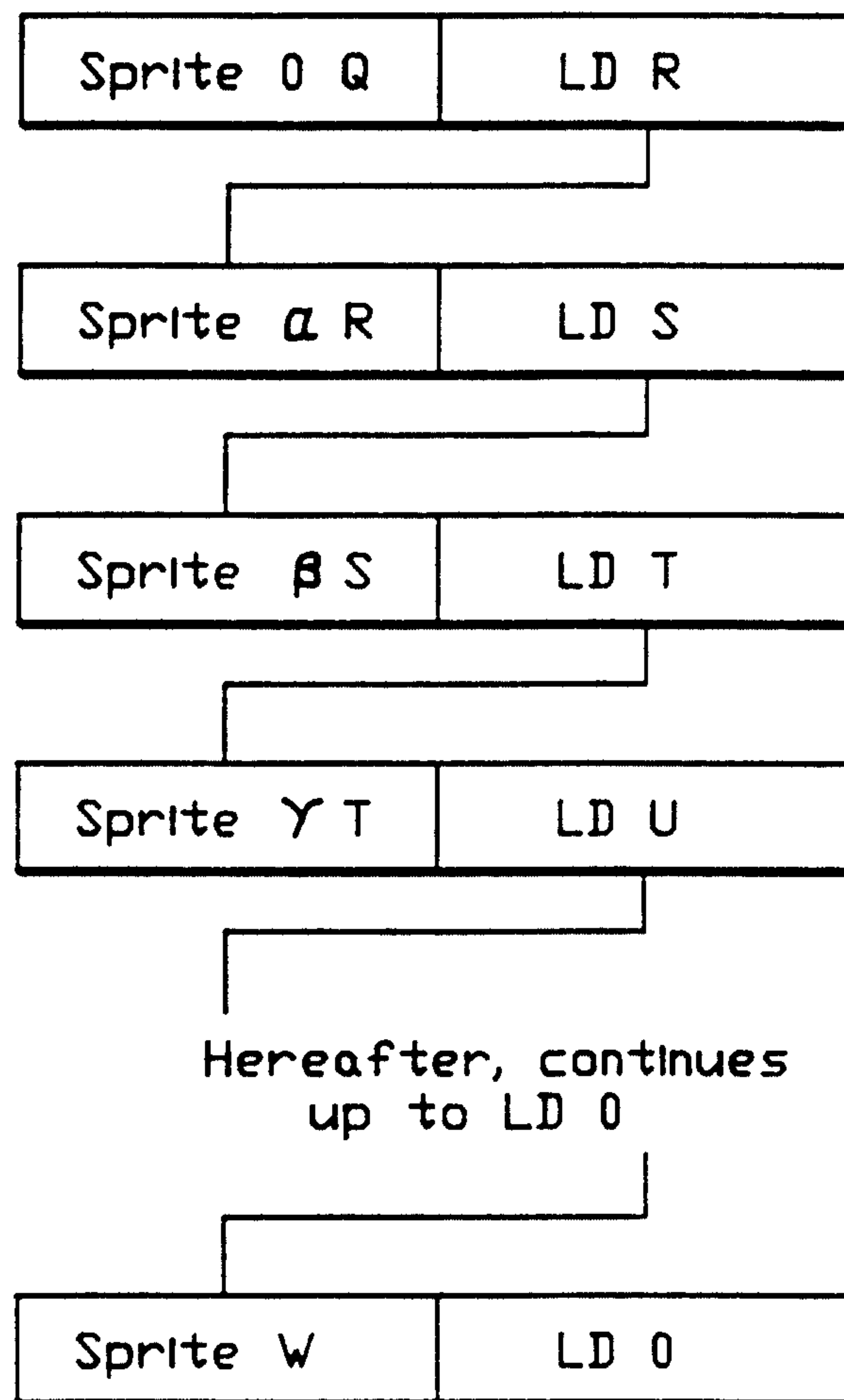


FIG.—14

<table><tr><td>V1 cell</td><td>H1 cell</td></tr><tr><td>0</td><td></td></tr></table>	V1 cell	H1 cell	0		<table><tr><td>V1 cell</td><td>H2 cell</td></tr><tr><td>0</td><td>1</td></tr></table>	V1 cell	H2 cell	0	1	<table><tr><td>V1 cell</td><td>H3 cell</td></tr><tr><td>0</td><td>1 2</td></tr></table>	V1 cell	H3 cell	0	1 2	<table><tr><td>V1 cell</td><td>H4 cell</td></tr><tr><td>0</td><td>1 2 3</td></tr></table>	V1 cell	H4 cell	0	1 2 3																								
V1 cell	H1 cell																																										
0																																											
V1 cell	H2 cell																																										
0	1																																										
V1 cell	H3 cell																																										
0	1 2																																										
V1 cell	H4 cell																																										
0	1 2 3																																										
A	B	C	D																																								
<table><tr><td>V2 cell</td><td>H1 cell</td></tr><tr><td>0</td><td></td></tr><tr><td>1</td><td></td></tr></table>	V2 cell	H1 cell	0		1		<table><tr><td>V2 cell</td><td>H2 cell</td></tr><tr><td>0</td><td>2</td></tr><tr><td>1</td><td>3</td></tr></table>	V2 cell	H2 cell	0	2	1	3	<table><tr><td>V2 cell</td><td>H3 cell</td></tr><tr><td>0</td><td>2 4</td></tr><tr><td>1</td><td>3 5</td></tr></table>	V2 cell	H3 cell	0	2 4	1	3 5	<table><tr><td>V2 cell</td><td>H4 cell</td></tr><tr><td>0</td><td>2 4 6</td></tr><tr><td>1</td><td>3 5 7</td></tr></table>	V2 cell	H4 cell	0	2 4 6	1	3 5 7																
V2 cell	H1 cell																																										
0																																											
1																																											
V2 cell	H2 cell																																										
0	2																																										
1	3																																										
V2 cell	H3 cell																																										
0	2 4																																										
1	3 5																																										
V2 cell	H4 cell																																										
0	2 4 6																																										
1	3 5 7																																										
E	F	G	H																																								
<table><tr><td>V3 cell</td><td>H1 cell</td></tr><tr><td>0</td><td></td></tr><tr><td>1</td><td></td></tr><tr><td>2</td><td></td></tr></table>	V3 cell	H1 cell	0		1		2		<table><tr><td>V3 cell</td><td>H2 cell</td></tr><tr><td>0</td><td>3</td></tr><tr><td>1</td><td>4</td></tr><tr><td>2</td><td>5</td></tr></table>	V3 cell	H2 cell	0	3	1	4	2	5	<table><tr><td>V3 cell</td><td>H3 cell</td></tr><tr><td>0</td><td>3 6</td></tr><tr><td>1</td><td>4 7</td></tr><tr><td>2</td><td>5 8</td></tr></table>	V3 cell	H3 cell	0	3 6	1	4 7	2	5 8	<table><tr><td>V3 cell</td><td>H4 cell</td></tr><tr><td>0</td><td>3 6 9</td></tr><tr><td>1</td><td>4 7 A</td></tr><tr><td>2</td><td>5 8 B</td></tr></table>	V3 cell	H4 cell	0	3 6 9	1	4 7 A	2	5 8 B								
V3 cell	H1 cell																																										
0																																											
1																																											
2																																											
V3 cell	H2 cell																																										
0	3																																										
1	4																																										
2	5																																										
V3 cell	H3 cell																																										
0	3 6																																										
1	4 7																																										
2	5 8																																										
V3 cell	H4 cell																																										
0	3 6 9																																										
1	4 7 A																																										
2	5 8 B																																										
I	J	K	L																																								
<table><tr><td>V4 cell</td><td>H1 cell</td></tr><tr><td>0</td><td></td></tr><tr><td>1</td><td></td></tr><tr><td>2</td><td></td></tr><tr><td>3</td><td></td></tr></table>	V4 cell	H1 cell	0		1		2		3		<table><tr><td>V4 cell</td><td>H2 cell</td></tr><tr><td>0</td><td>4</td></tr><tr><td>1</td><td>5</td></tr><tr><td>2</td><td>6</td></tr><tr><td>3</td><td>7</td></tr></table>	V4 cell	H2 cell	0	4	1	5	2	6	3	7	<table><tr><td>V4 cell</td><td>H3 cell</td></tr><tr><td>0</td><td>4 8</td></tr><tr><td>1</td><td>5 9</td></tr><tr><td>2</td><td>6 A</td></tr><tr><td>3</td><td>7 B</td></tr></table>	V4 cell	H3 cell	0	4 8	1	5 9	2	6 A	3	7 B	<table><tr><td>V4 cell</td><td>H4 cell</td></tr><tr><td>0</td><td>4 8 C</td></tr><tr><td>1</td><td>5 9 D</td></tr><tr><td>2</td><td>6 A E</td></tr><tr><td>3</td><td>7 B F</td></tr></table>	V4 cell	H4 cell	0	4 8 C	1	5 9 D	2	6 A E	3	7 B F
V4 cell	H1 cell																																										
0																																											
1																																											
2																																											
3																																											
V4 cell	H2 cell																																										
0	4																																										
1	5																																										
2	6																																										
3	7																																										
V4 cell	H3 cell																																										
0	4 8																																										
1	5 9																																										
2	6 A																																										
3	7 B																																										
V4 cell	H4 cell																																										
0	4 8 C																																										
1	5 9 D																																										
2	6 A E																																										
3	7 B F																																										
M	N	O	P																																								

FIG. -15

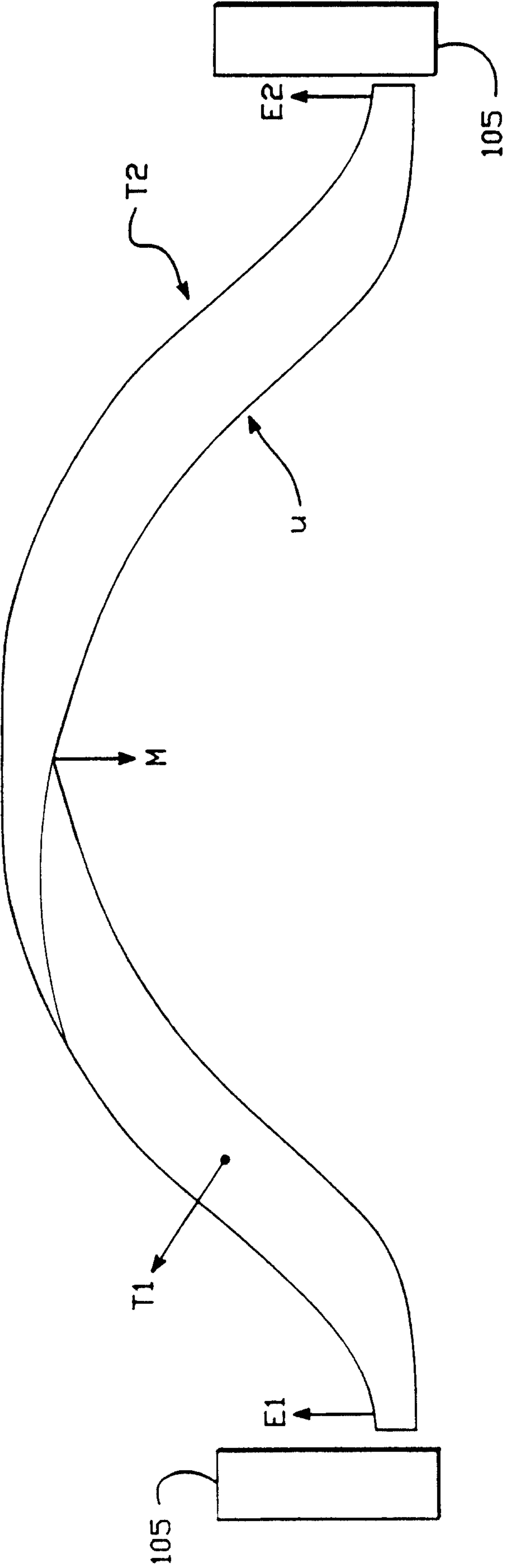


FIG. — 16A

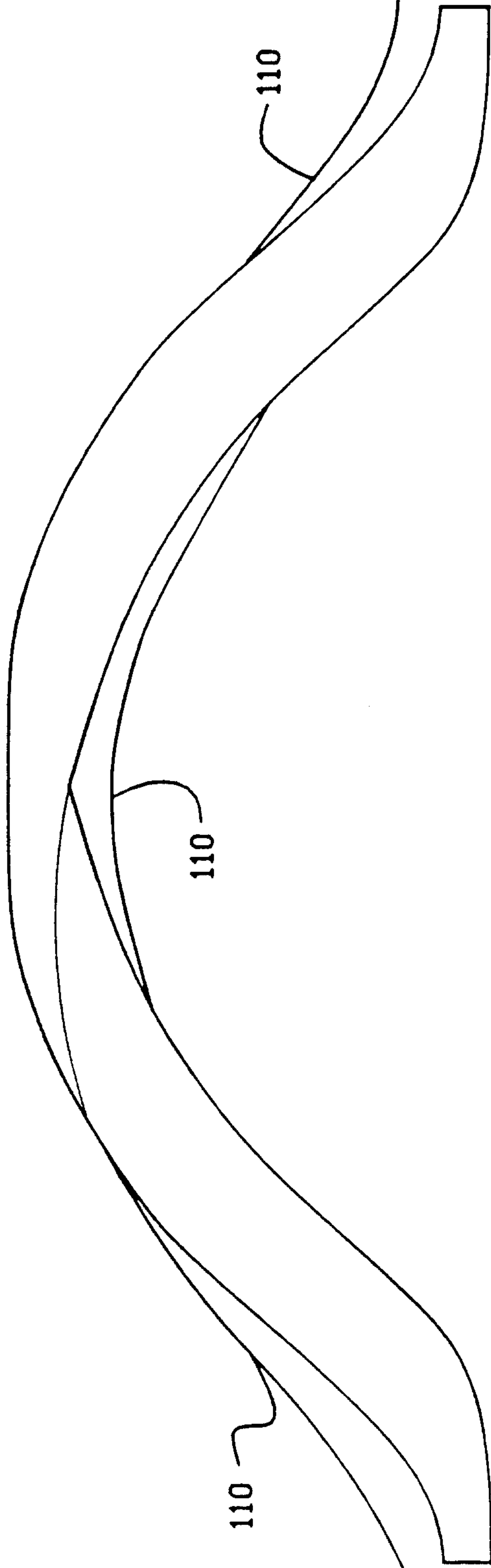


FIG. — 16B

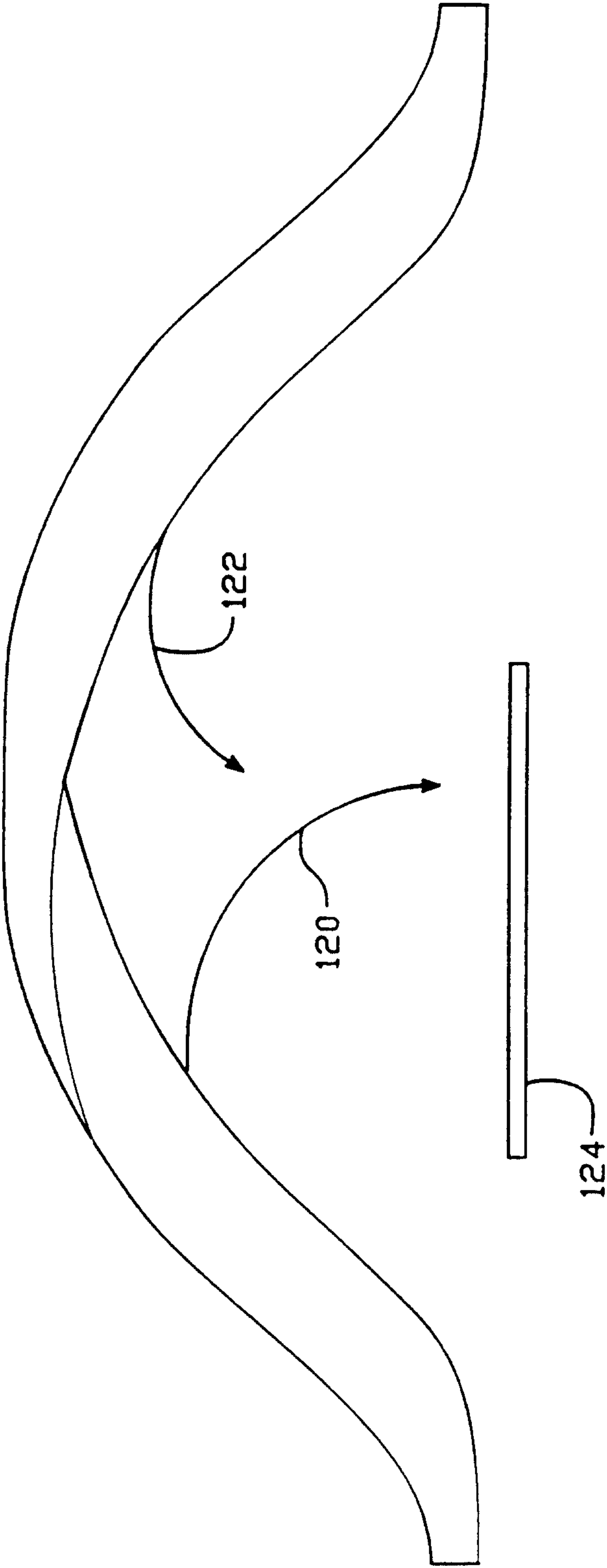


FIG. — 16C

18/46

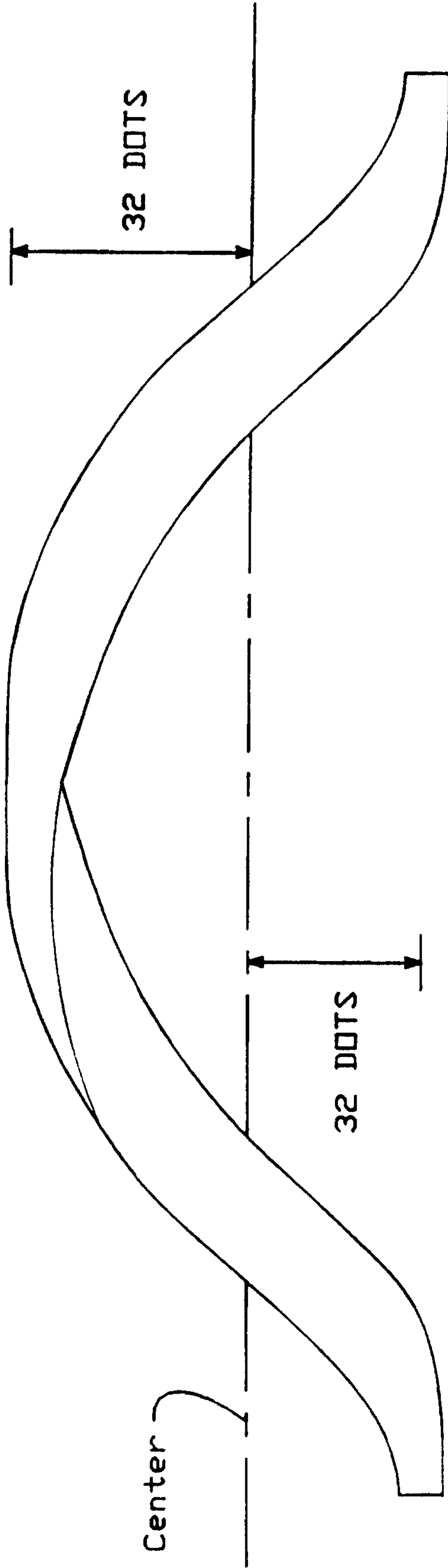


FIG.—16D

19/46

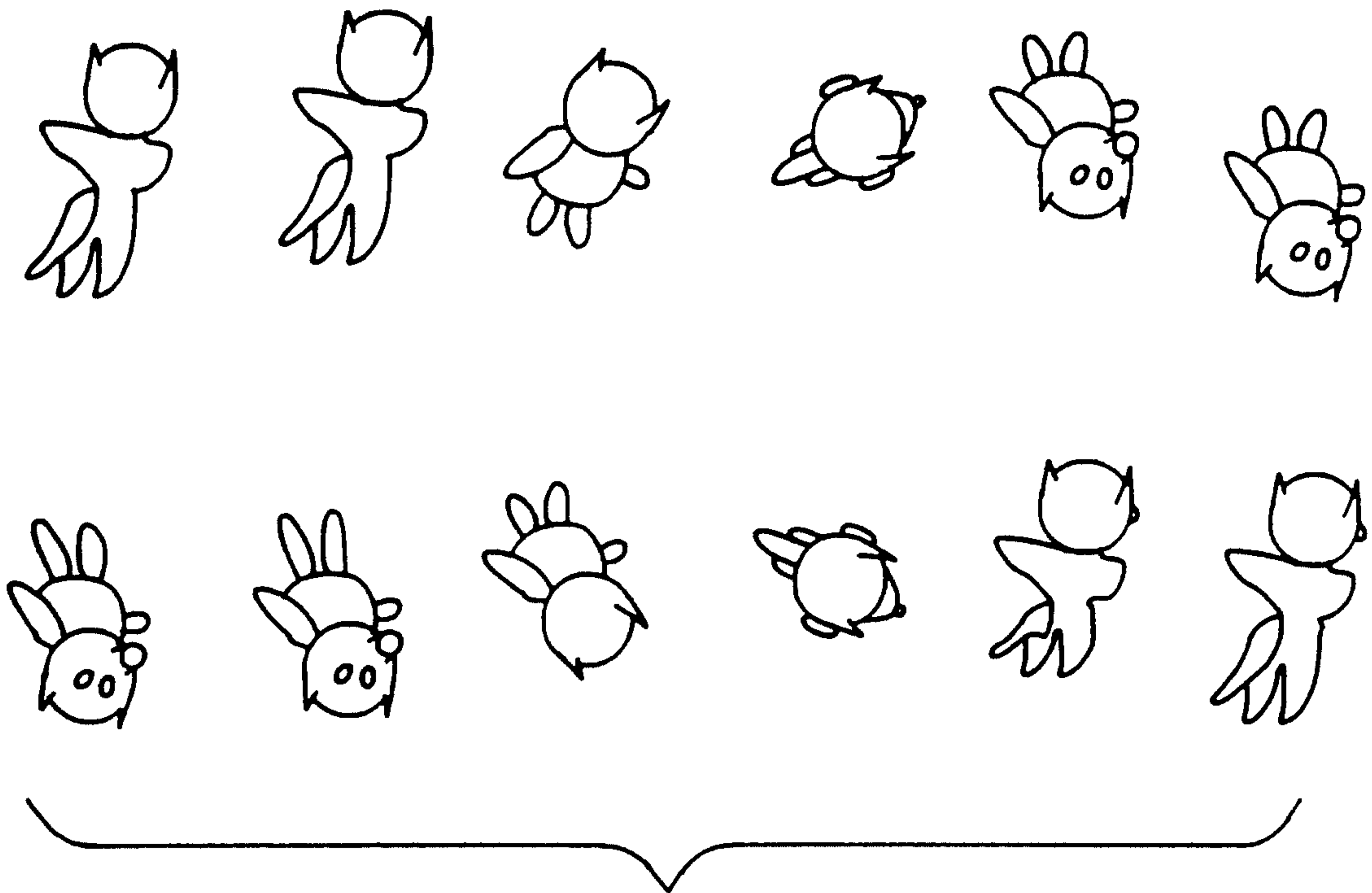


FIG.—17

20/46

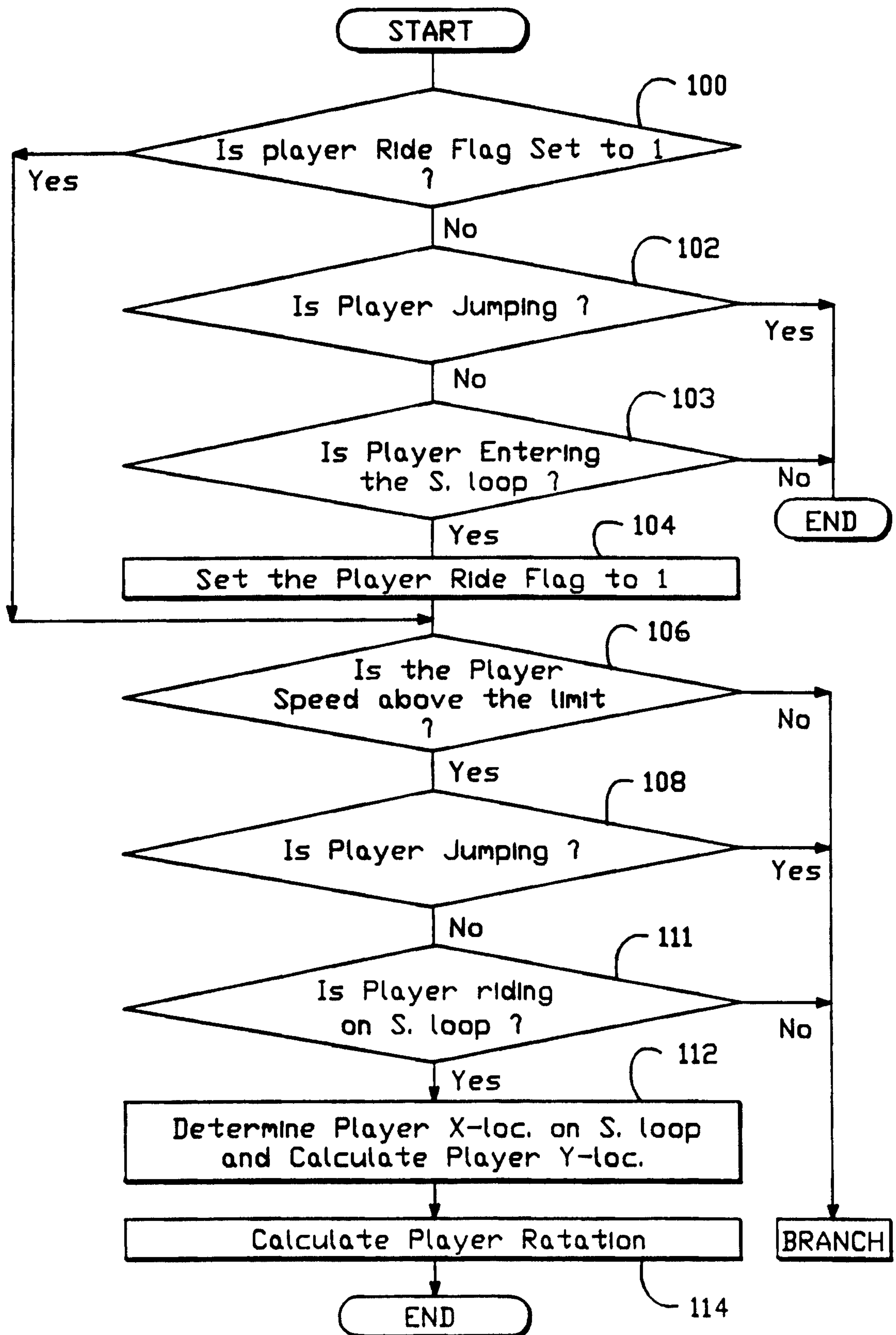


FIG.-18A

21/46

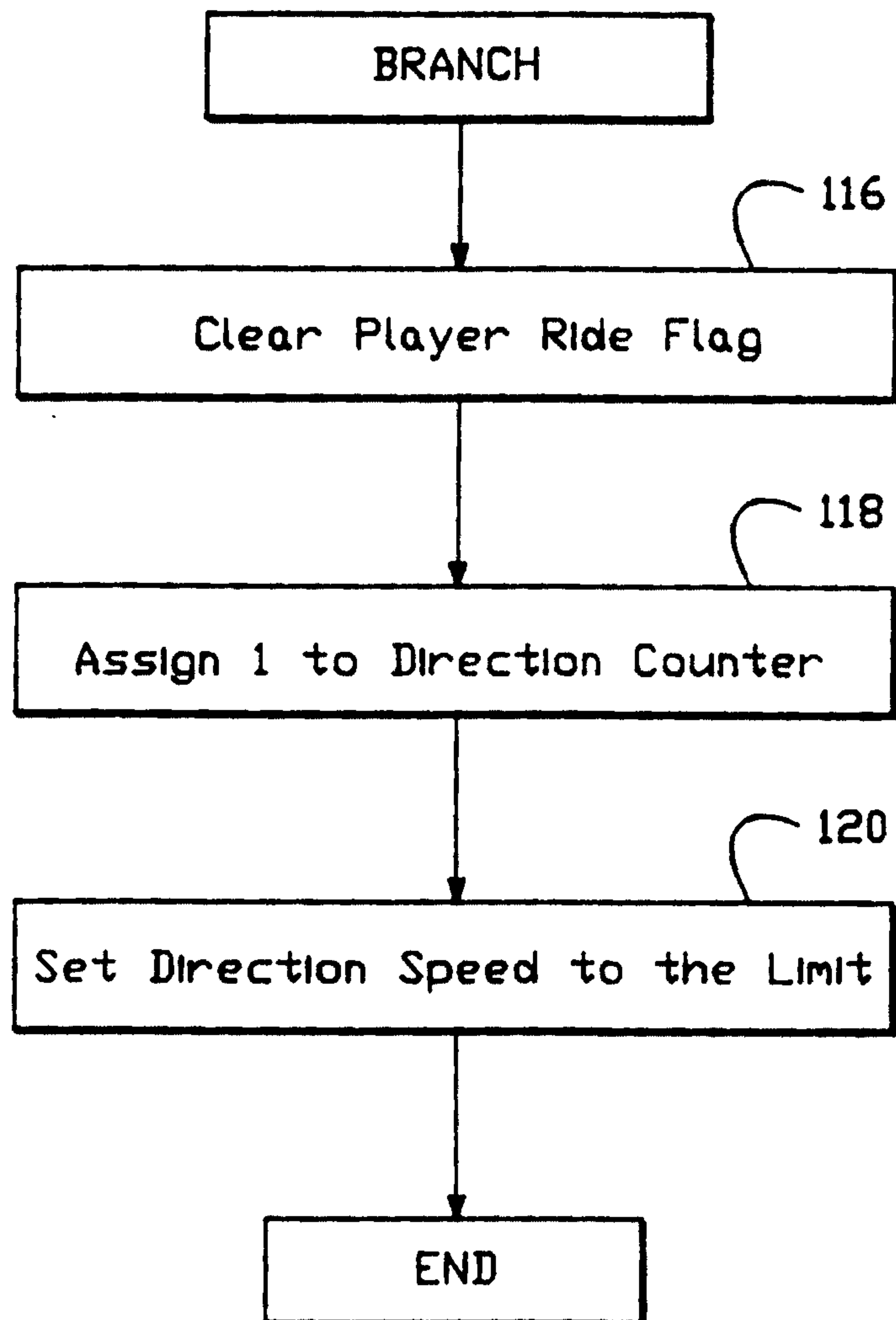


FIG.-18B

22/46

sloopdirtbl

dc.b	S00,S00	Table of Angles
dc.b	S01,S01,S16,S16,S16,S16,S2c,S2c	
dc.b	S2c,S2c,S42,S42,S42,S42,S58,S58	
dc.b	S58,S58,S6e,S6e,S6e,S6e,S84,S84,	
dc.b	S84,S84,S9a,S9a,S9a,S9a,Sb0,Sb0,	
dc.b	Sb0,Sb0,Sc6,Sc6,Sc6,Sc6,Sdc,Sdc,	
dc.b	Sdc,Sdc,Sf2,Sf2,Sf2,Sf2,S01,S01	
dc.b	S00,S00	

slooptbl

dc.b	032, 032, 032, 032, 032, 032, 032, 032	Entry Points (E1)
dc.b	032, 032, 032, 032, 032, 032, 032, 032	
dc.b	032, 032, 032, 032, 032, 032, 032, 032	Begin Offset Table
dc.b	032, 032, 032, 032, 032, 032, 031, 031	
dc.b	031, 031, 031, 031, 031, 031, 031, 031	
dc.b	031, 031, 031, 031, 031, 030, 030, 030	
dc.b	030, 030, 030, 030, 030, 030, 029, 029	
dc.b	029, 029, 029, 028, 028, 028, 028, 027	
dc.b	027, 027, 027, 026, 026, 026, 025, 025	
dc.b	025, 024, 204, 024, 023, 023, 022, 022	
dc.b	021, 021, 020, 020, 019, 018, 018, 017	
dc.b	016, 016, 015, 014, 014, 013, 012, 012	
dc.b	011, 010, 010, 009, 008, 008, 007, 006	
dc.b	006, 005, 004, 004, 003, 002, 002, 001	
dc.b	000, -01, -02, -02, -03, -04, -04, -05	
dc.b	-06, -07, -07, -08, -09, -09, -10, -10,	
dc.b	-11, -11, -12, -12, -13, -14, -14, -15	
dc.b	-15, -16, -16, -17, -17, -18, -18, -19	
dc.b	-19, -19, -20, -21, -21, -22, -22, -23	
dc.b	-23, -24, -24, -25, -25, -26, -26, -27	
dc.b	-27, -28, -28, -28, -29, -29, -30, -30	
dc.b	-30, -31, -31, -31, -32, -32, -32, -33	
dc.b	-33, -33, -33, -34, -34, -34, -35, -35	
dc.b	-35, -35, -35, -35, -35, -35, -36, -36	
dc.b	-36, -36, -36, -36, -36, -36, -36, -37	
dc.b	-37, -37, -37, -37, -37, -37, -37, -37	

FIG.-19A

23/46

dc.b	-37, -37, -37, -37, -37, -37, -37, -37,	
dc.b	-37, -37, -37, -37, -37, -37, -37, -37	
dc.b	-37, -37, -37, -37, -36, -36, -36, -36	
dc.b	-36, -36, -36, -35, -35, -35, -35, -35	
dc.b	-35, -35, -35, -34, -34, -34, -33, -33	
dc.b	-33, -33, -32, -32, -32, -31, -31, -31	
dc.b	-30, -30, -30, -29, -29, -28, -28, -28	
dc.b	-27, -27, -26, -26, -25, -25, -24, -24	
dc.b	-23, -23, -22, -22, -21, -21, -20, -19	
dc.b	-19, -18, -18, -17, -16, -16, -15, -14	
dc.b	-14, -13, -12, -11, -11, -10, -09, -08	
dc.b	-07, -07, -06, -05, -04, -03, -02, -01	
dc.b	000, 001, 002, 003, 004, 005, 006, 007	
dc.b	008, 008, 009, 010, 010, 011, 012, 013	
dc.b	013, 014, 014, 015, 015, 016, 016, 017	
dc.b	017, 018, 018, 019, 019, 020, 020, 021	
dc.b	021, 022, 022, 023, 023, 024, 024, 024,	
dc.b	025, 025, 025, 025, 026, 026, 026, 026	
dc.b	027, 027, 027, 027, 028, 028, 028, 028	
dc.b	028, 028, 029, 029, 029, 029, 029, 029	
dc.b	029, 030, 030, 030, 030, 030, 030, 030	
dc.b	031, 031, 031, 031, 031, 031, 031, 031	
dc.b	031, 031, 032, 032, 032, 032, 032, 032	
dc.b	032, 032, 032, 032, 032, 032, 032, 032	
		End Offset Table
dc.b	032, 032, 032, 032, 032, 032, 032, 032	Entry Points (E2)
dc.b	032, 032, 032, 032, 032, 032, 032, 032	

FIG.—19B

24/46

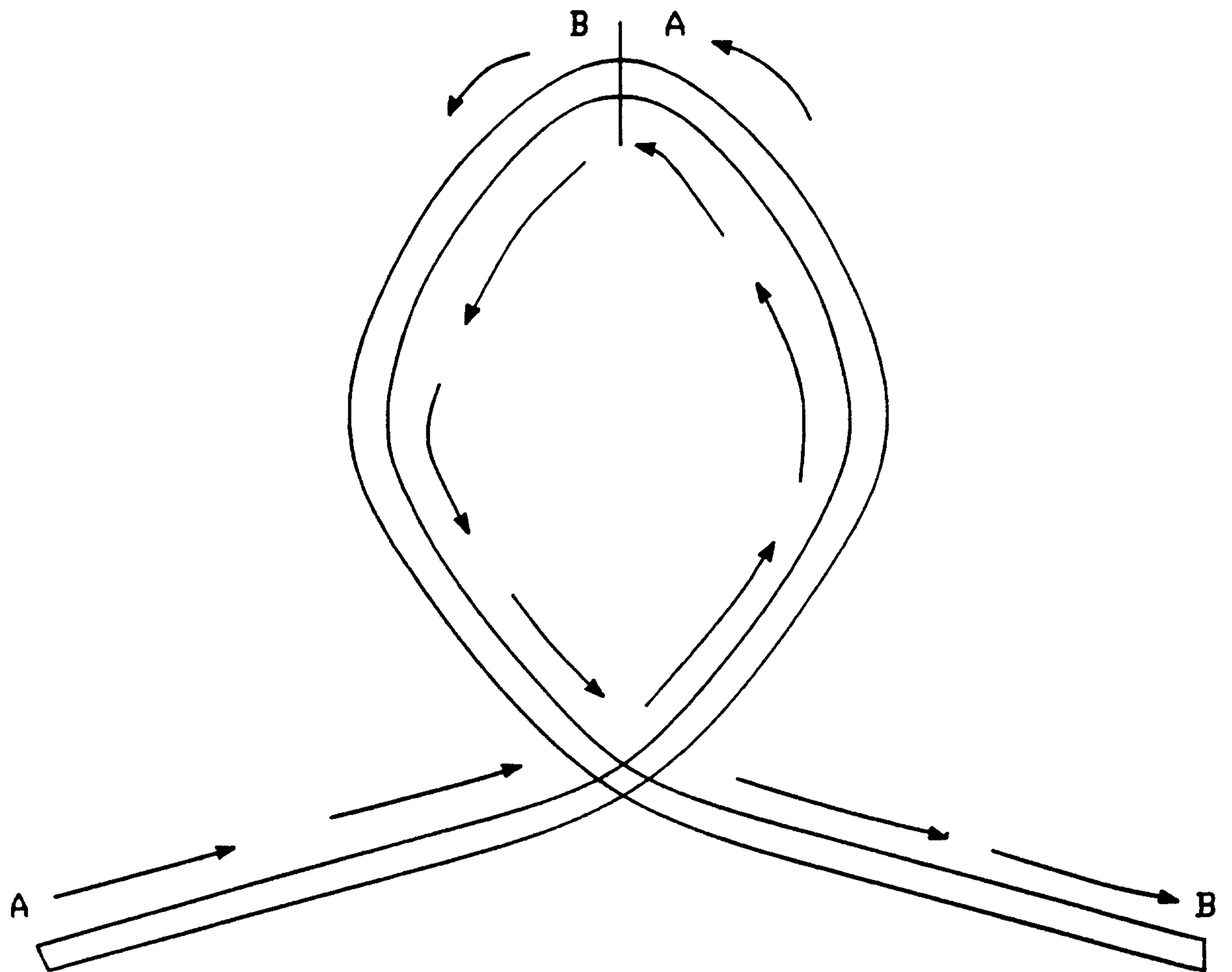


FIG.-20A

25/46

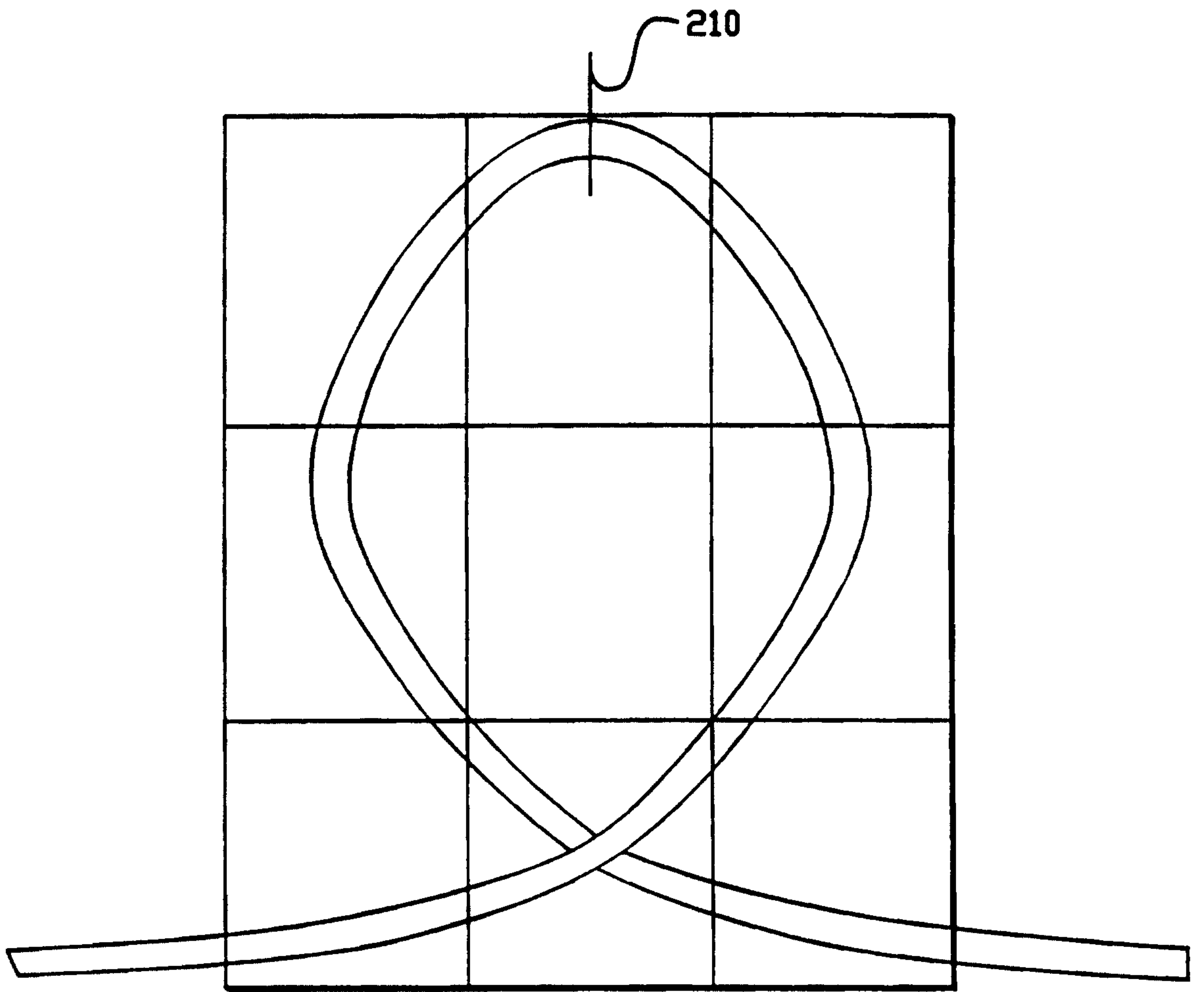


FIG.—20B

G10	G11	G12
G8		G9
G5	G6	G7

FIG.—20C

27/46

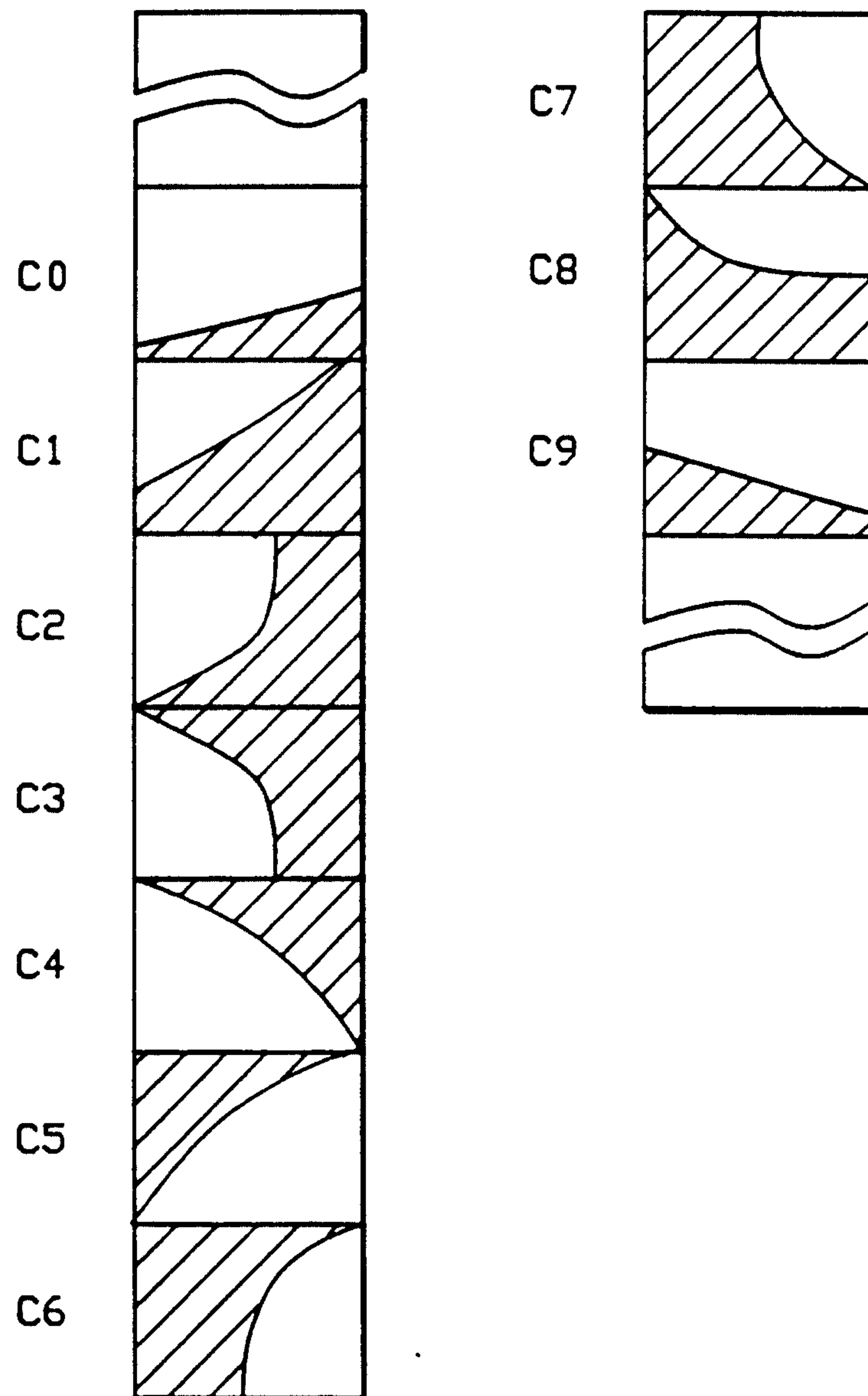


FIG.-21

28/46

TableGraphics to Collision Block Cross-Reference

	A	B
Graphics	Collision	Collision
Block	Type	Type
G5	C0	C0
G6	C1	C8
G7	C9	C9
G8	C7	C7
G9	C2	C2
G10	C6	C6
G11	C4	C5
G12	C3	C3

FIG.-22

29/46

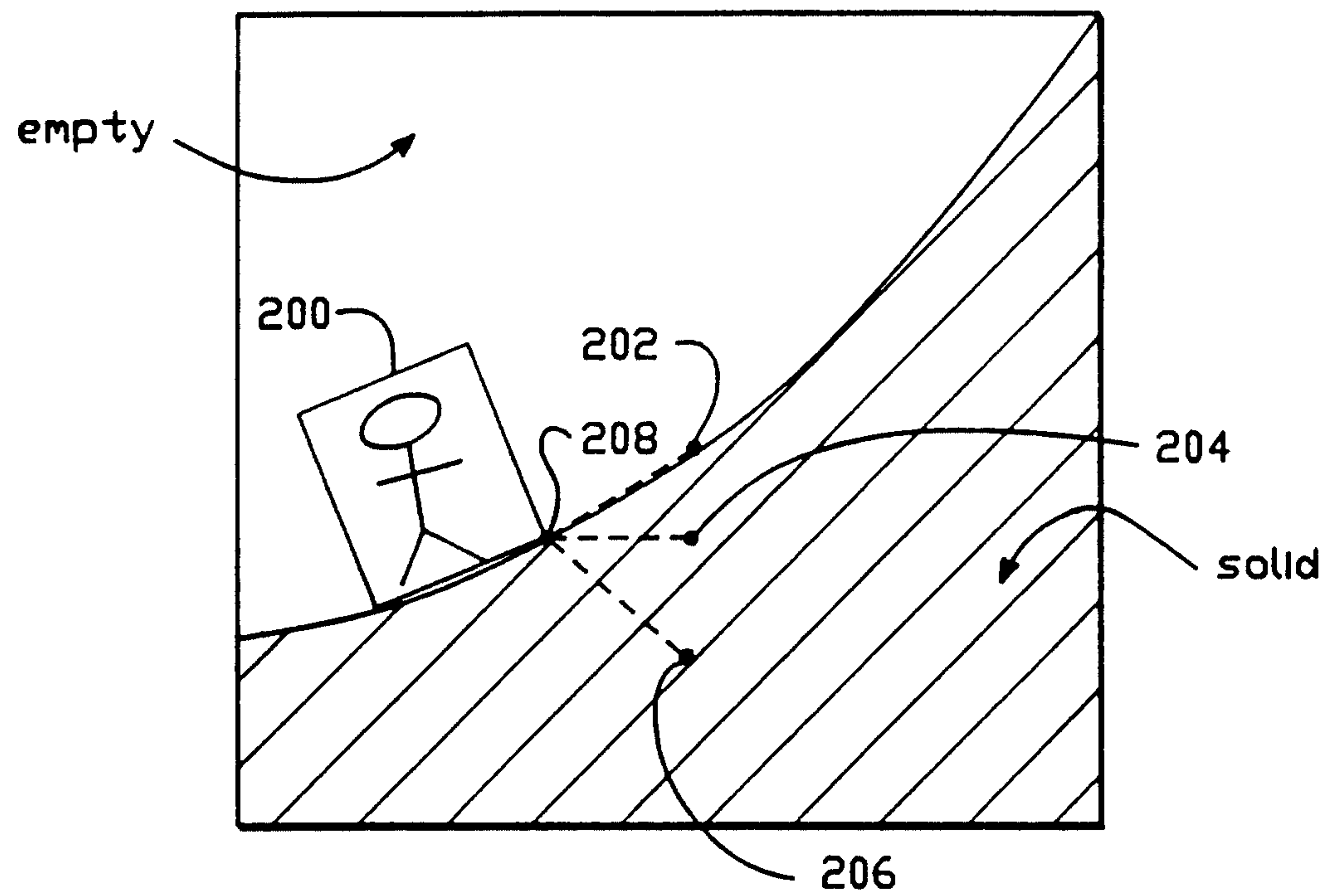


FIG.-23

30/46

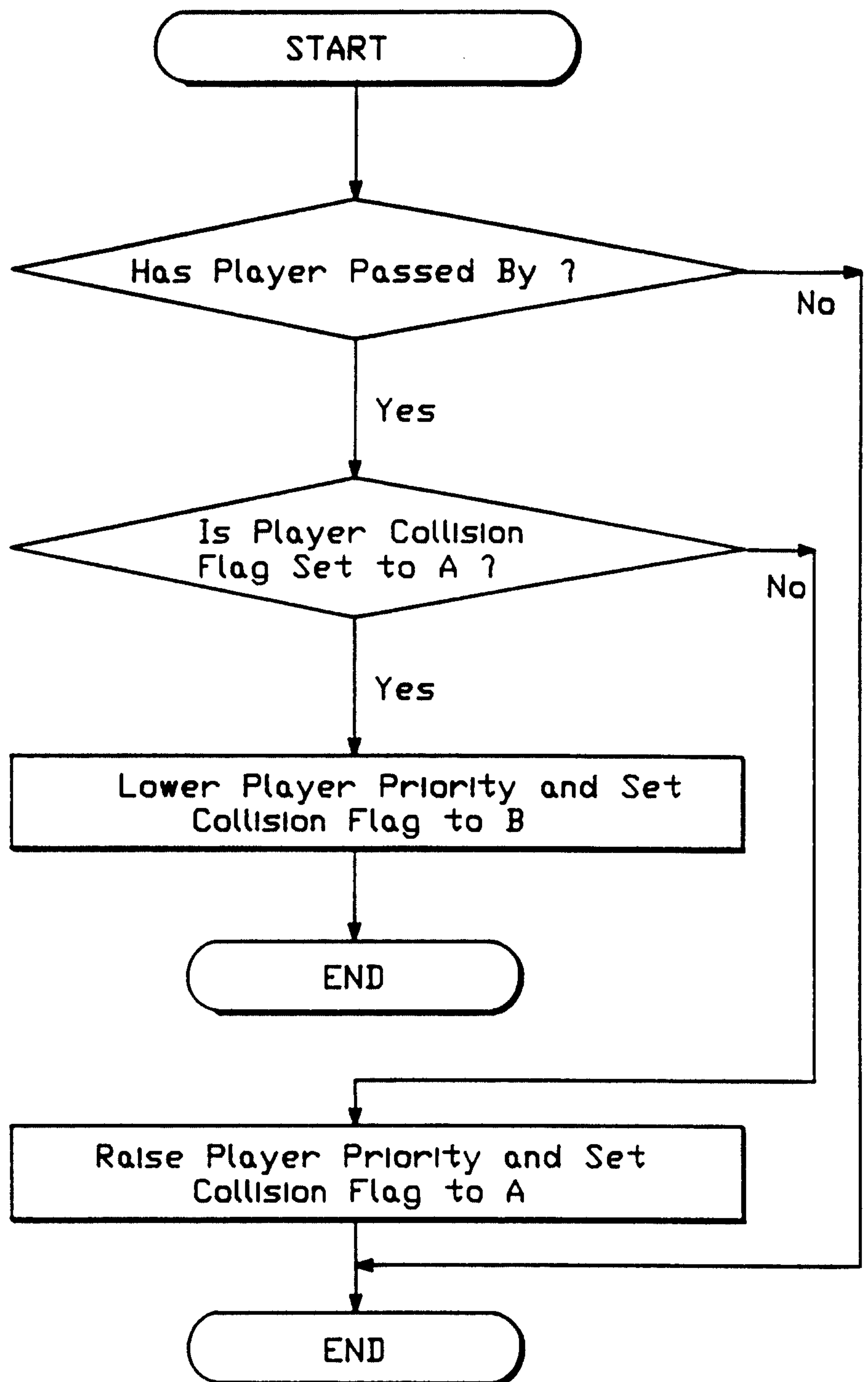


FIG.-24

31/46

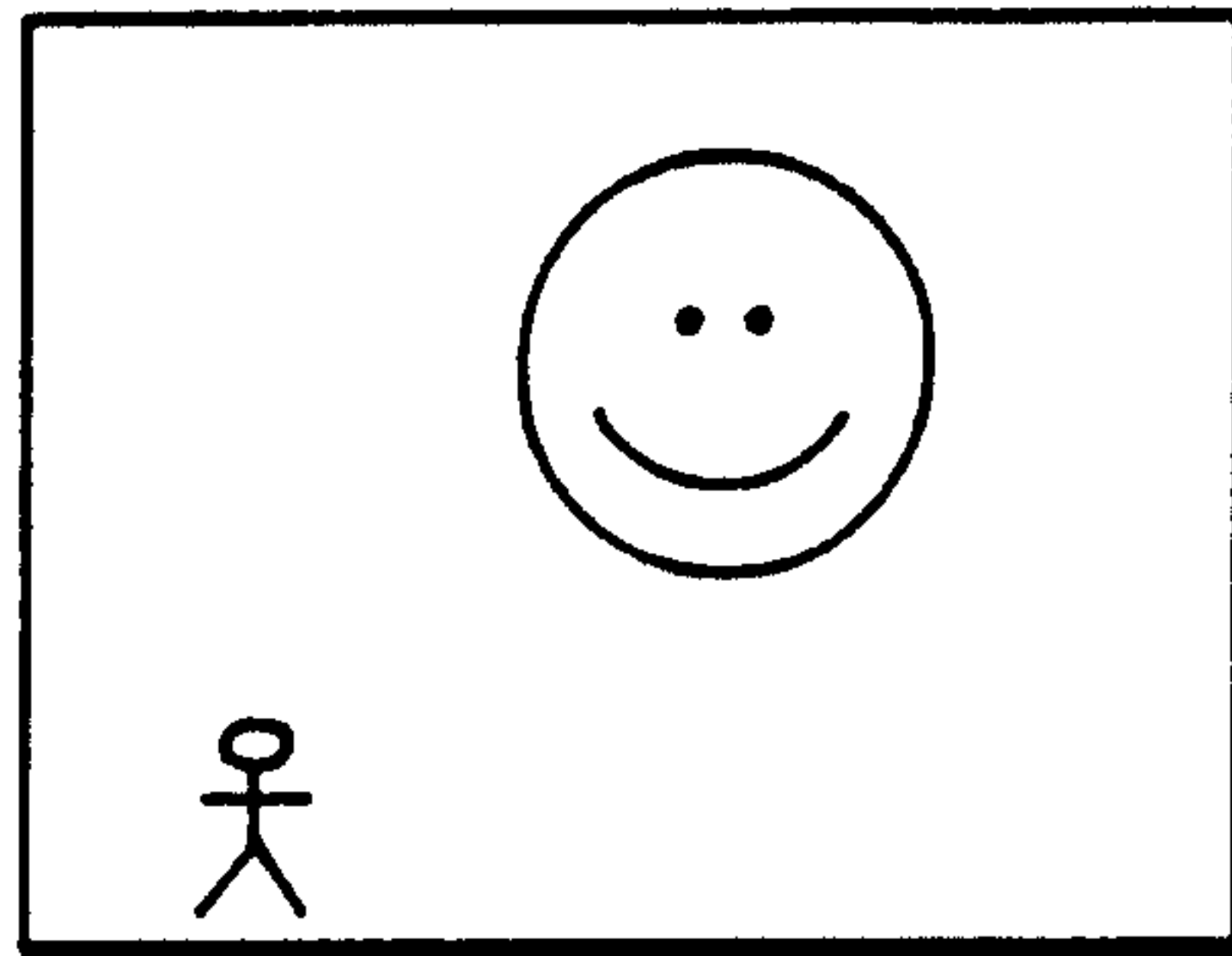


FIG. -25A
(PRIOR ART)

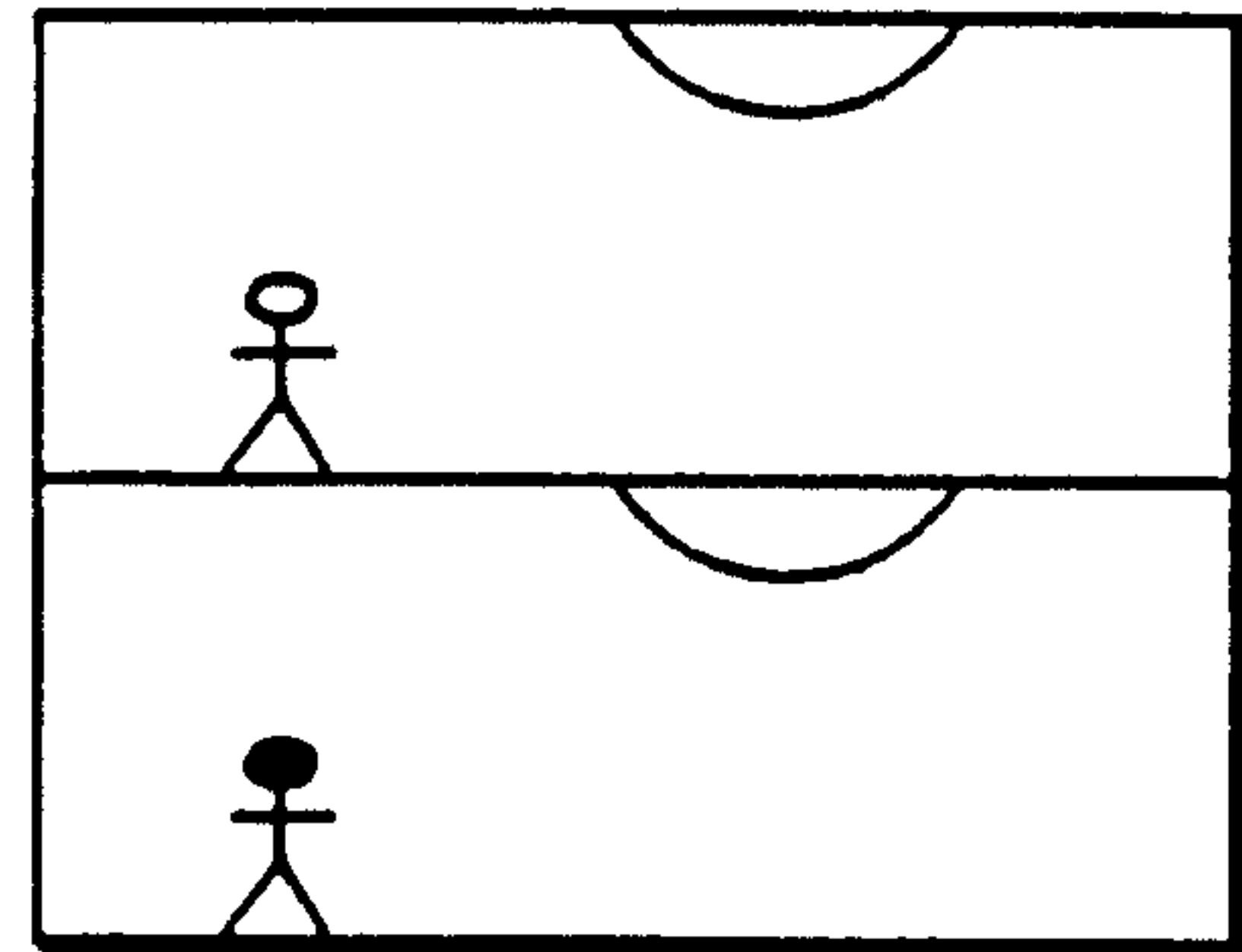


FIG. -25B
(PRIOR ART)

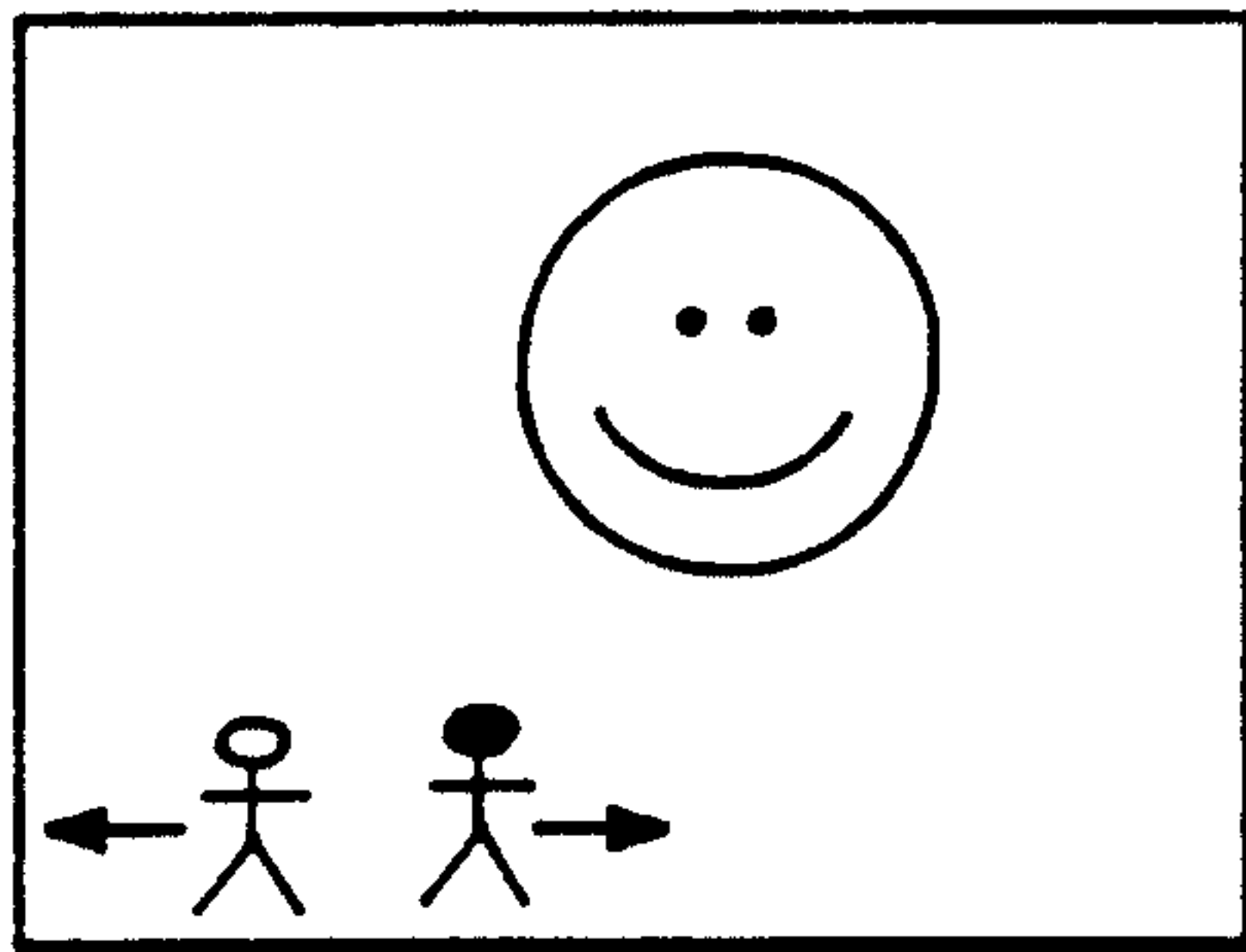


FIG. -26A
(PRIOR ART)

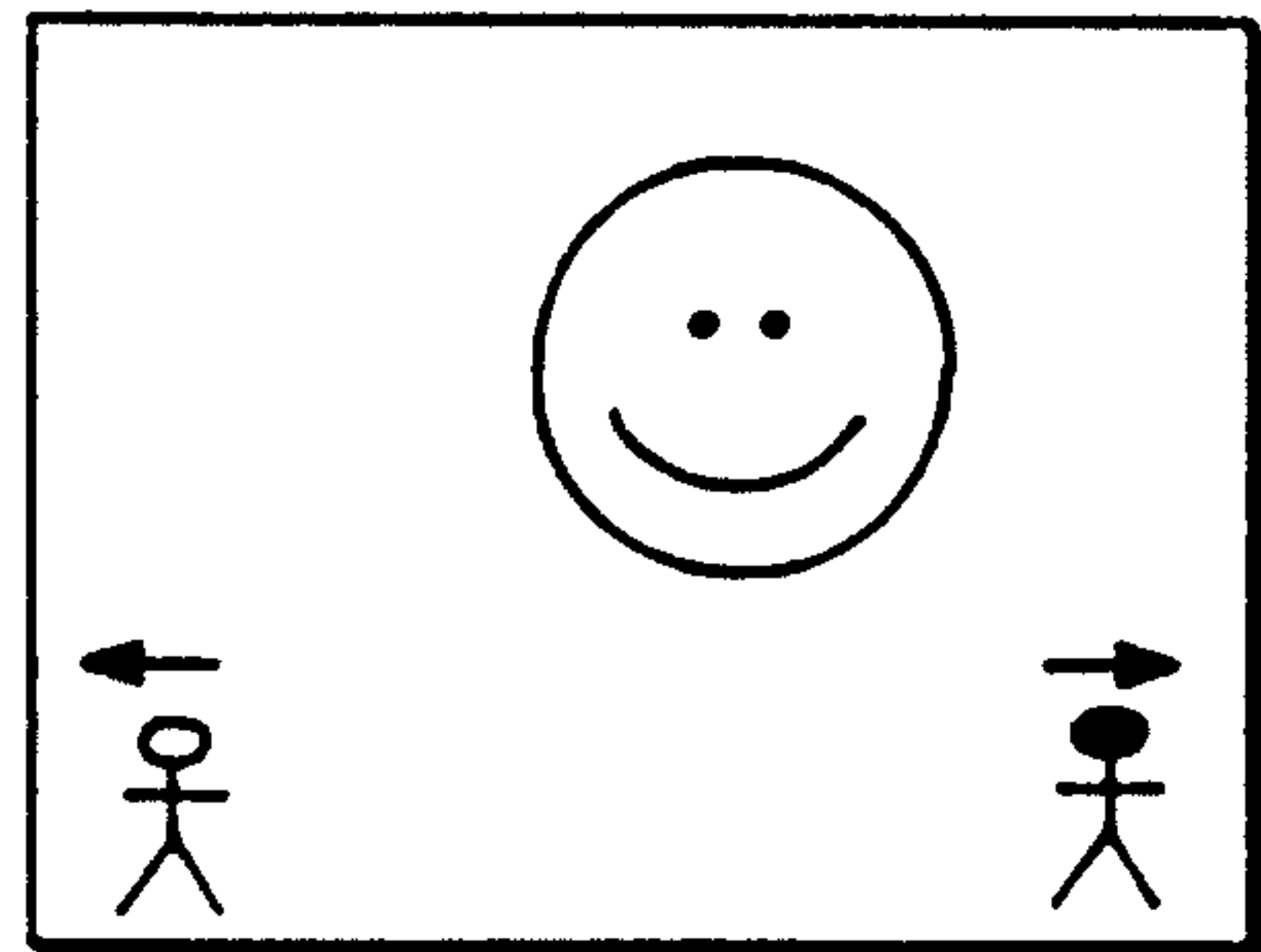


FIG. -26B
(PRIOR ART)

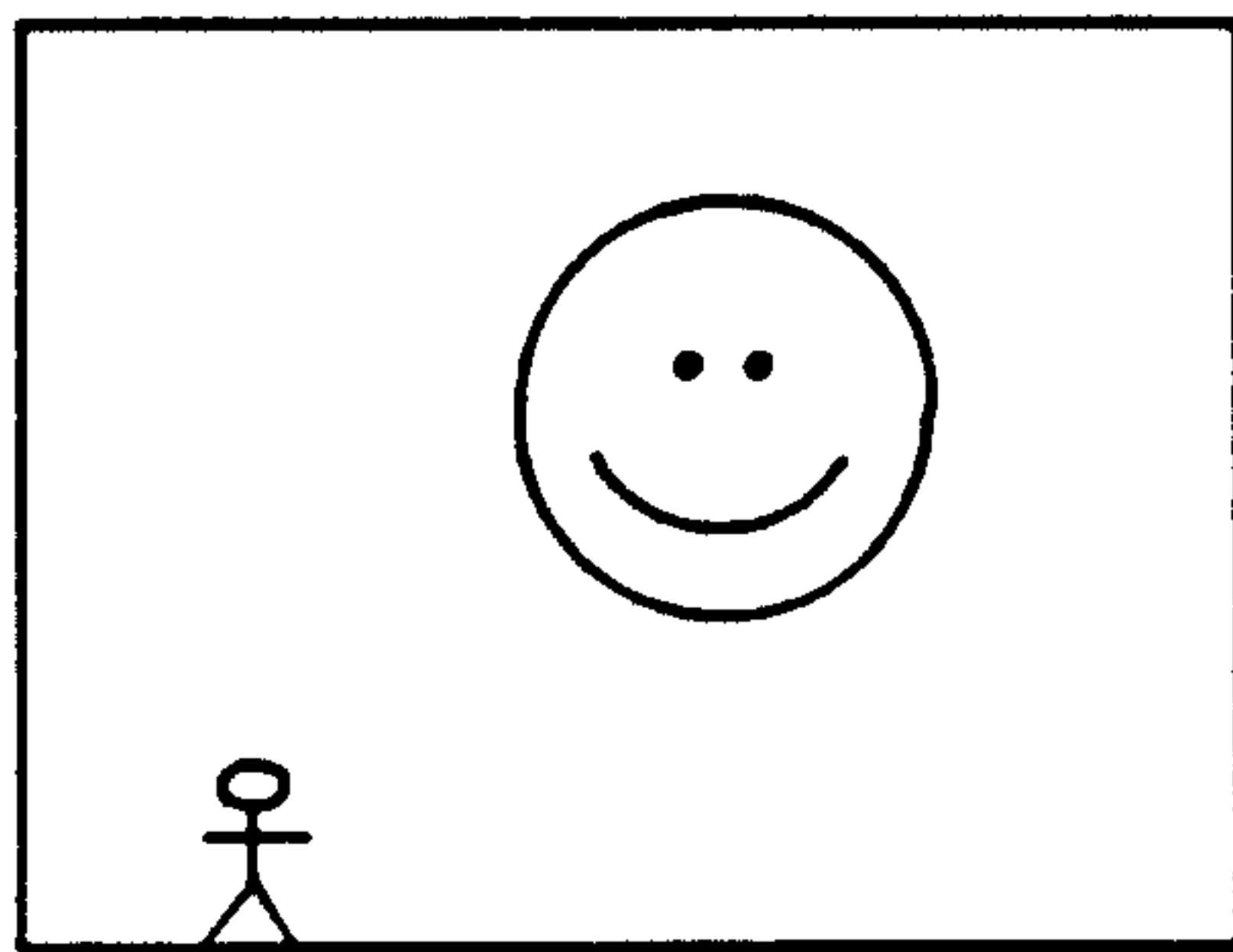


FIG. -27A

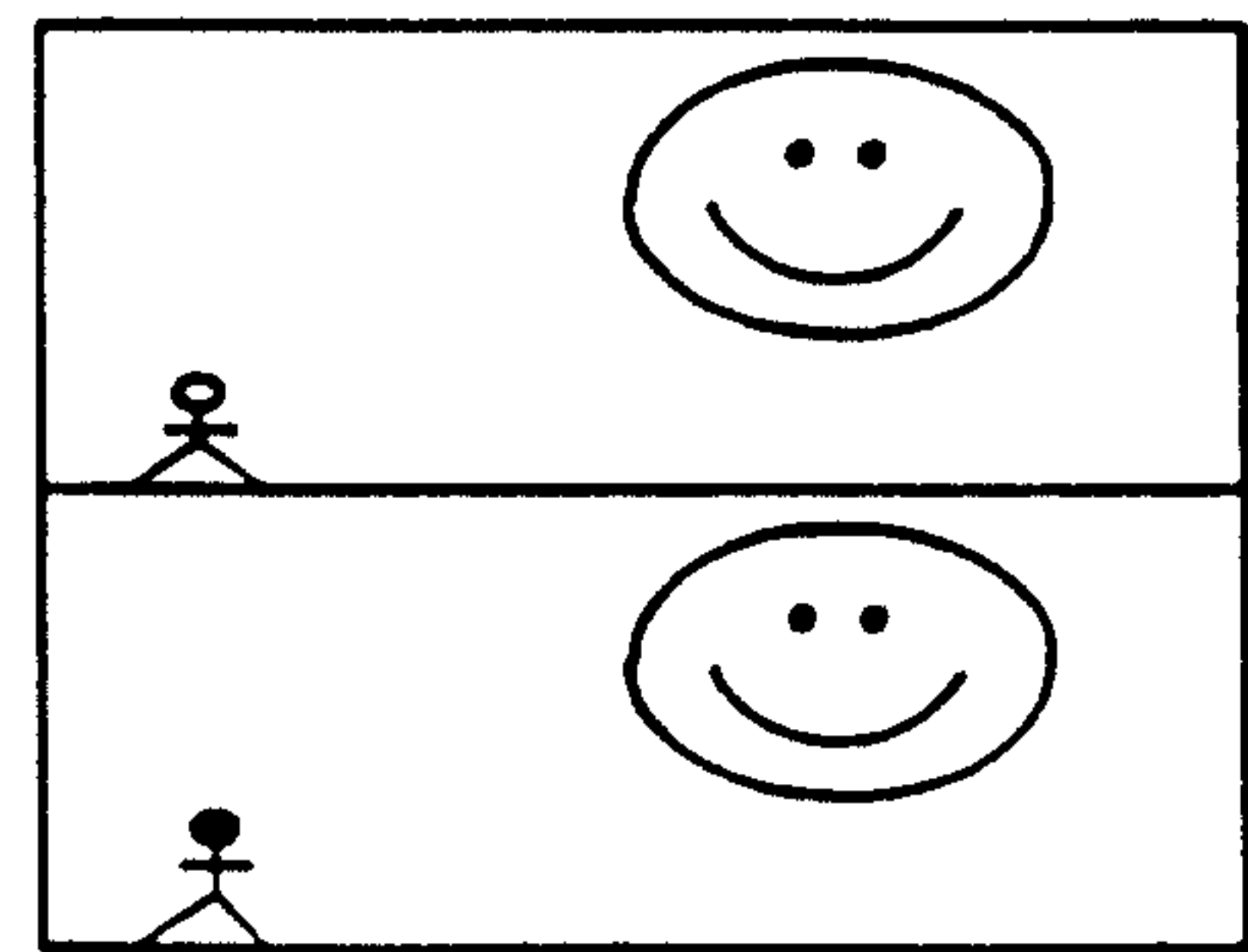


FIG. -27B

32/46

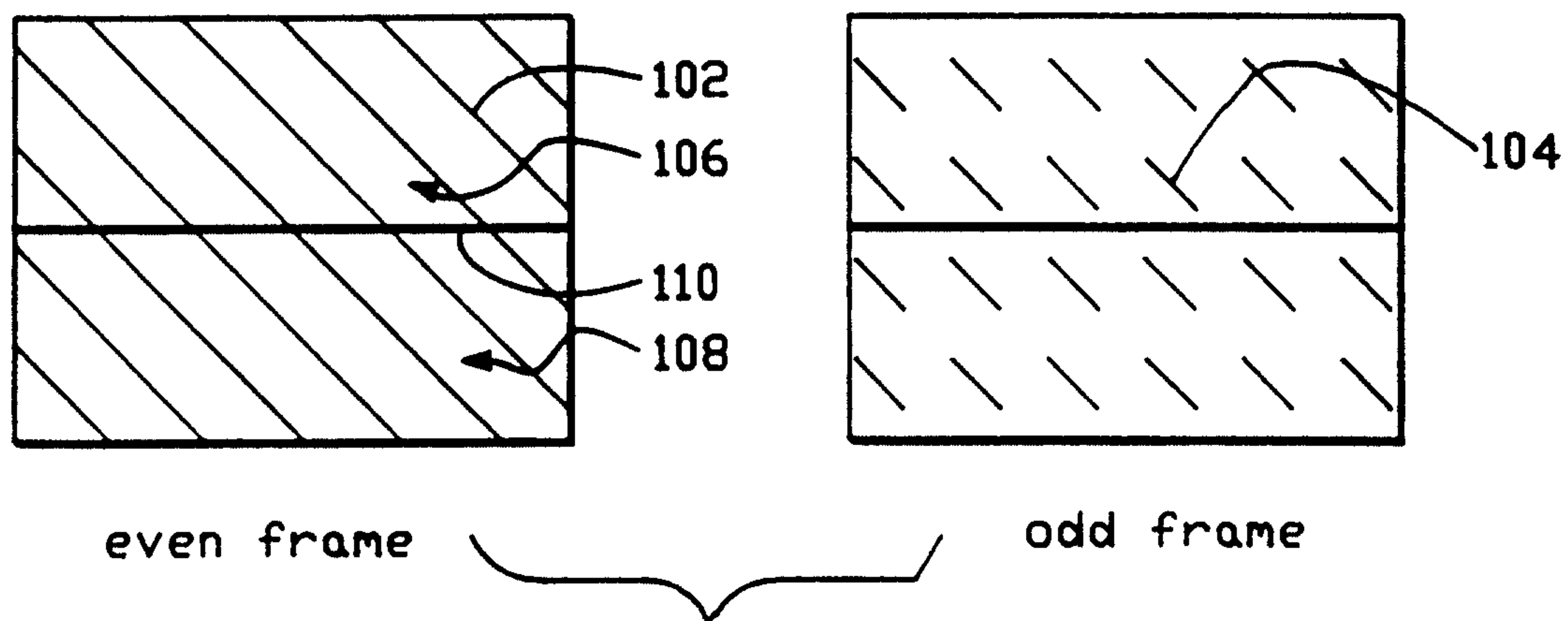
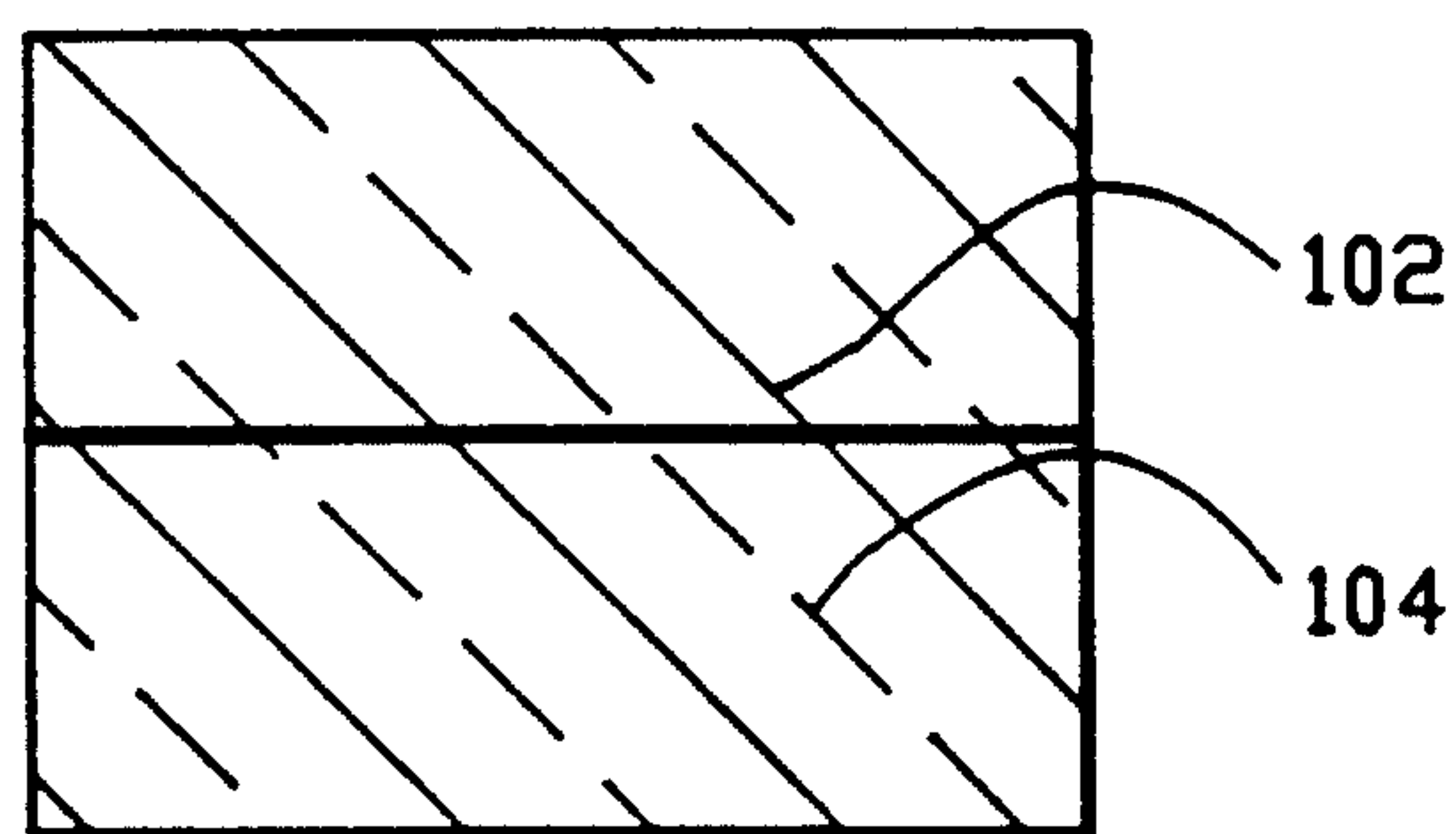


FIG.—29

FIG.—30



interlaced frame

FIG.—28

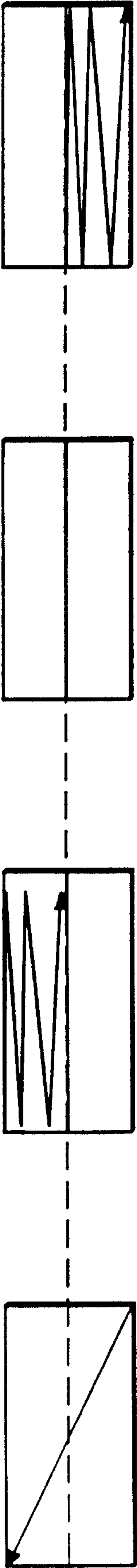


FIG.-31A FIG.-31B FIG.-31C FIG.-31D

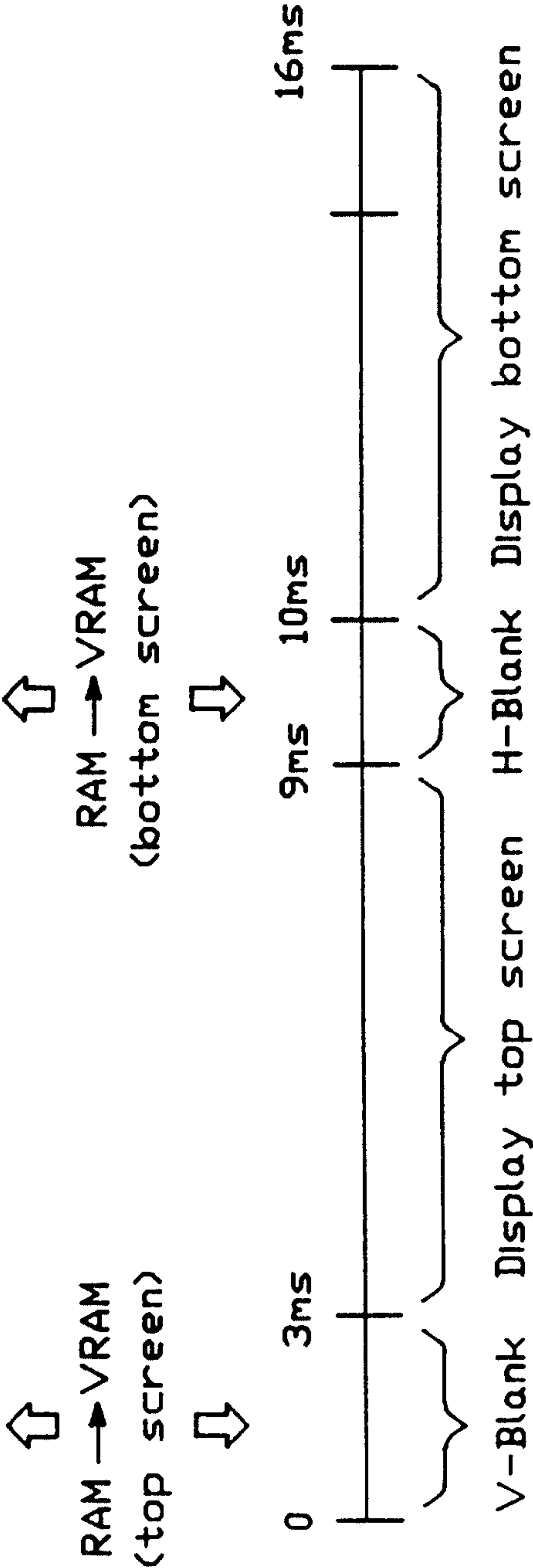


FIG.-32

34/46

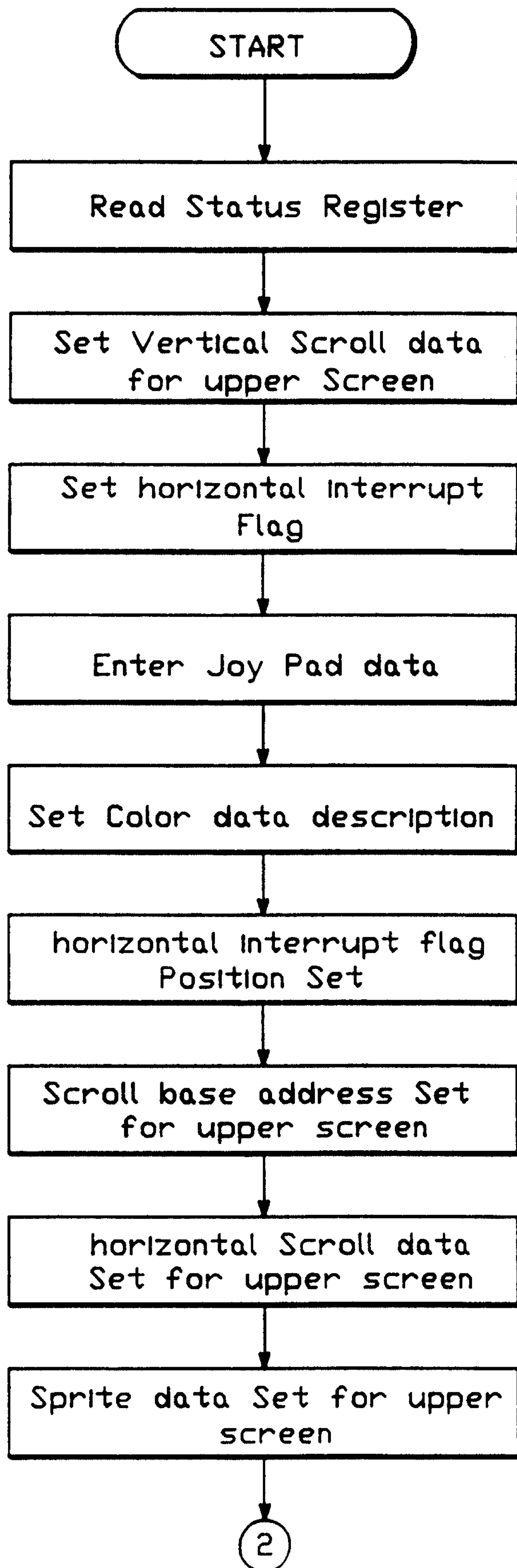


FIG. -33A

35/46

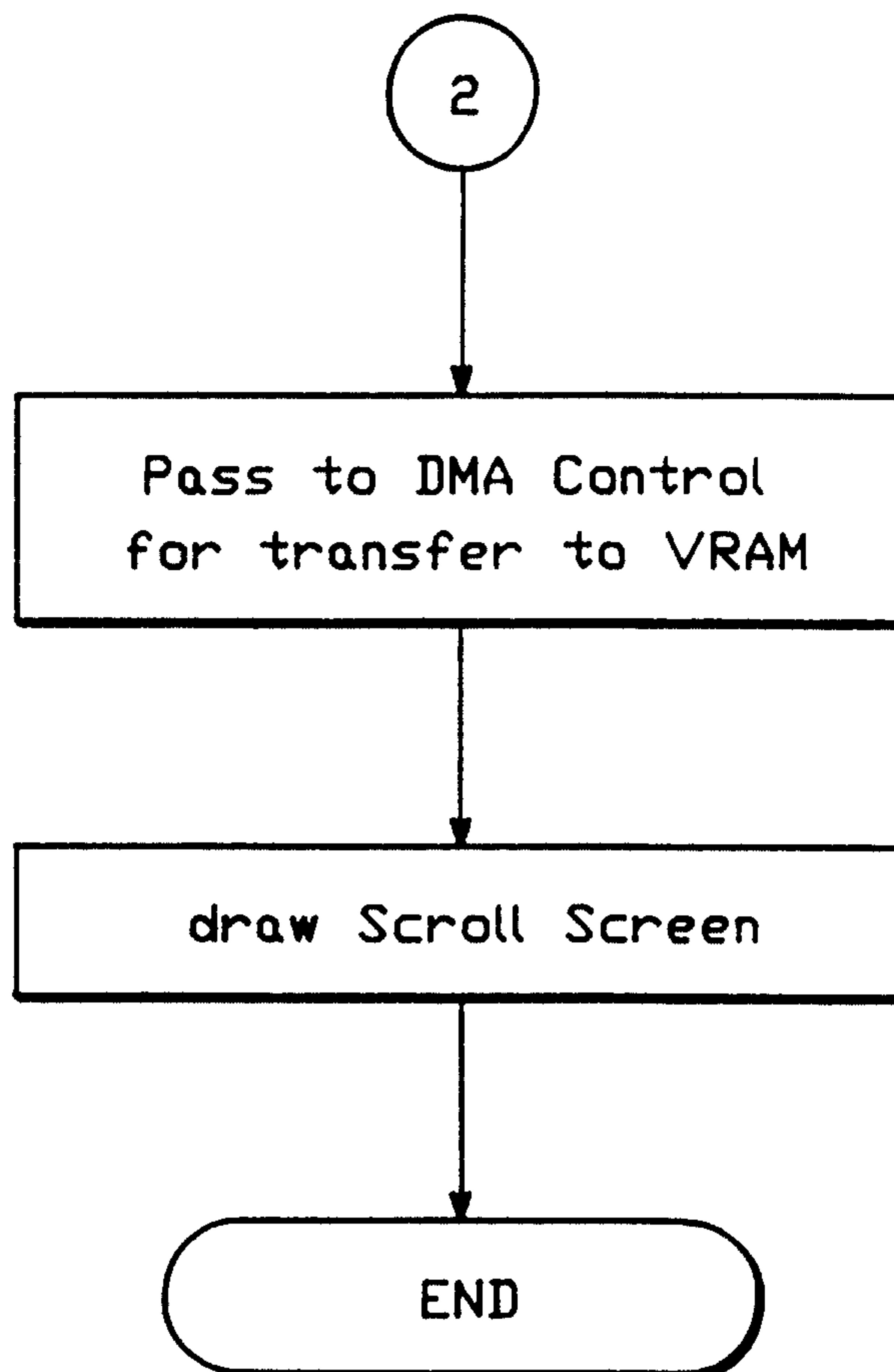


FIG.—33B

36/46

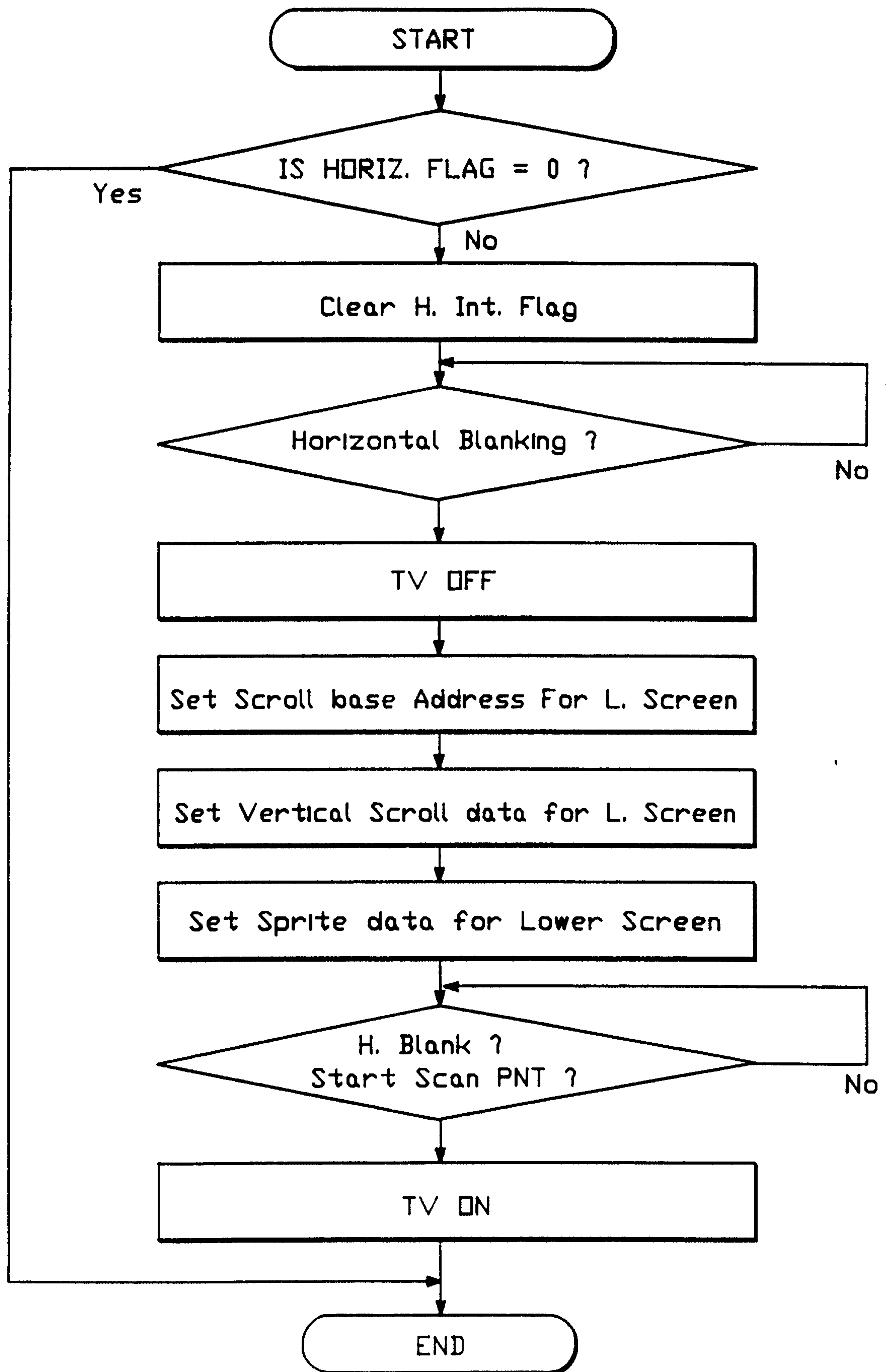


FIG.-34

In Interlace Mode 2, one cell consists of 8 x 16 dots and therefore 64 bytes (16 long words) are required.

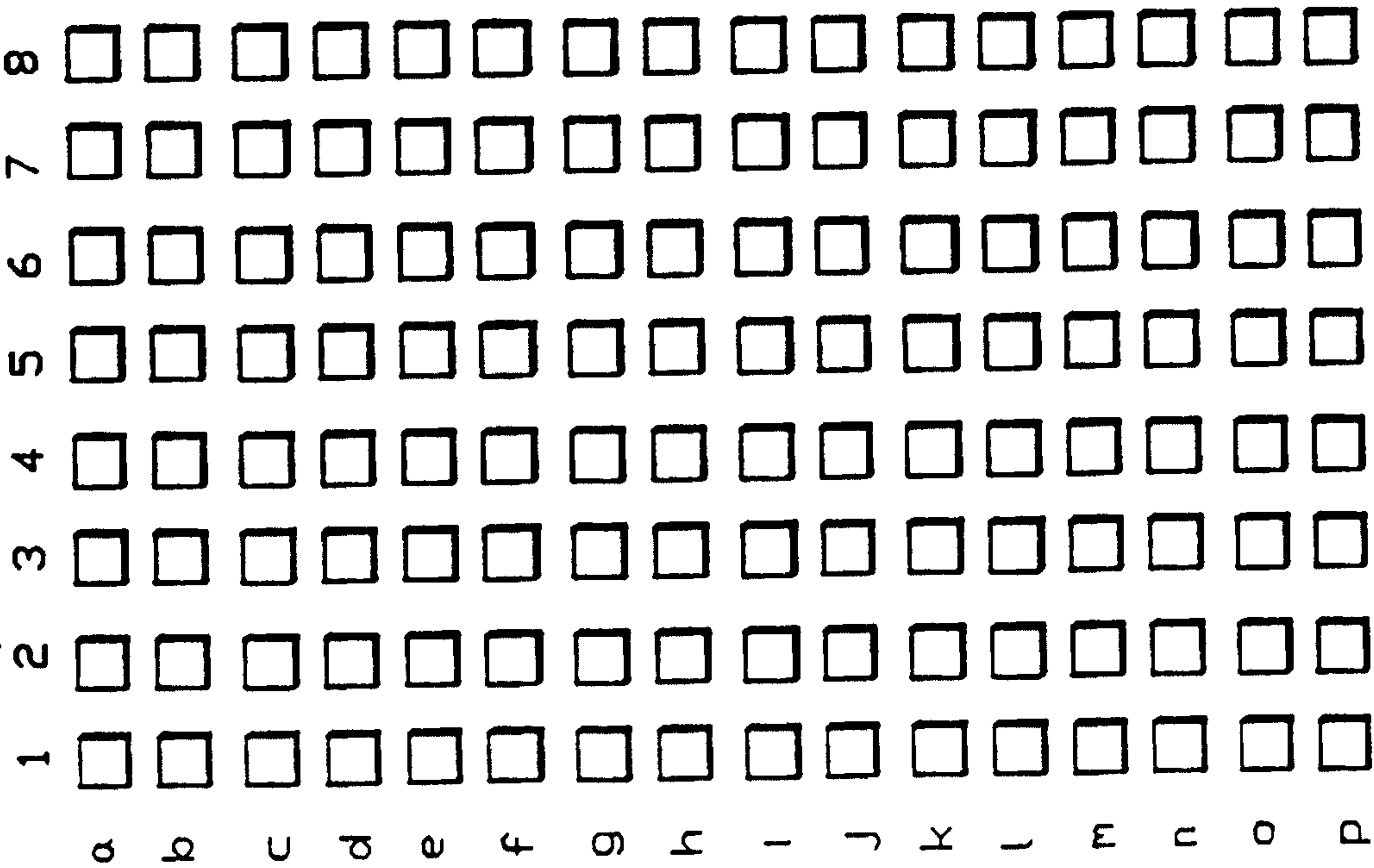


FIG.—35

	0		1		2		3	
	D7	D0	D7	D0	D7	D0	D7	D0
00	a1	a2	a3	a4	a5	a6	a7	a8
04	b1	b2	b3	b4	b5	b6	b7	b8
08	c1	c2	c3	c4	c5	c6	c7	c8
0C	d1	d2	d3	d4	d5	d6	d7	d8
10	e1	e2	e3	e4	e5	e6	e7	e8
14	f1	f2	f3	f4	f5	f6	f7	f8
18	g1	g2	g3	g4	g5	g6	g7	g8
1C	h1	h2	h3	h4	h5	h6	h7	h8
20	i1	i2	i3	i4	i5	i6	i7	i8
24	j1	j2	j3	j4	j5	j6	j7	j8
28	k1	k2	k3	k4	k5	k6	k7	k8
2C	l1	l2	l3	l4	l5	l6	l7	l8
30	m1	m2	m3	m4	m5	m6	m7	m8
34	n1	n2	n3	n4	n5	n6	n7	n8
38	o1	o2	o3	o4	o5	o6	o7	o8
3C	p1	p2	p3	p4	p5	p6	p7	p8

FIG.—36

38/46

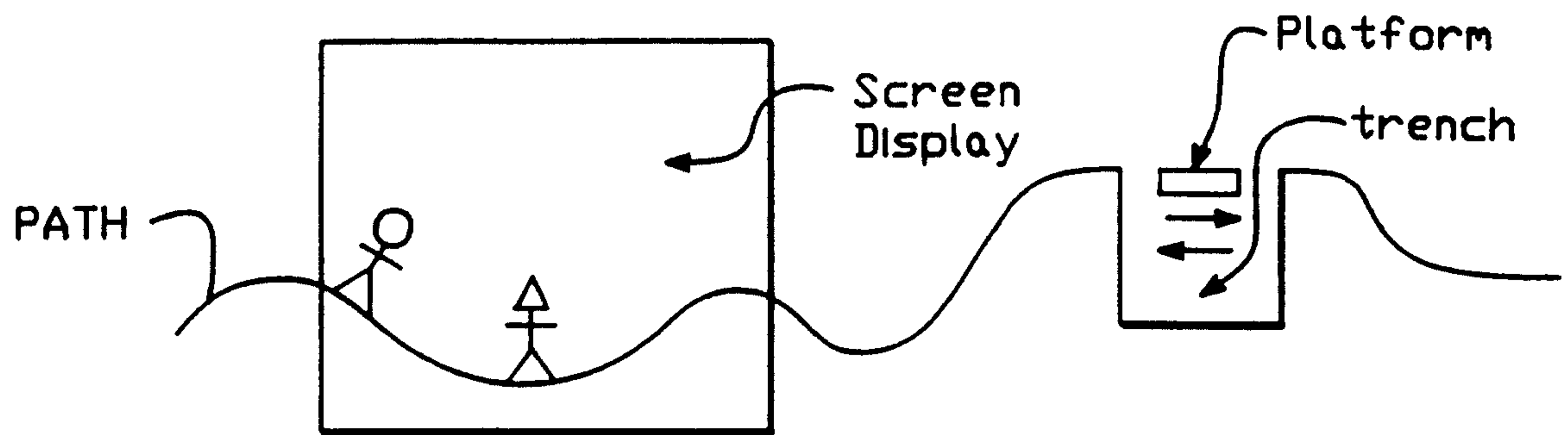


FIG. -37A

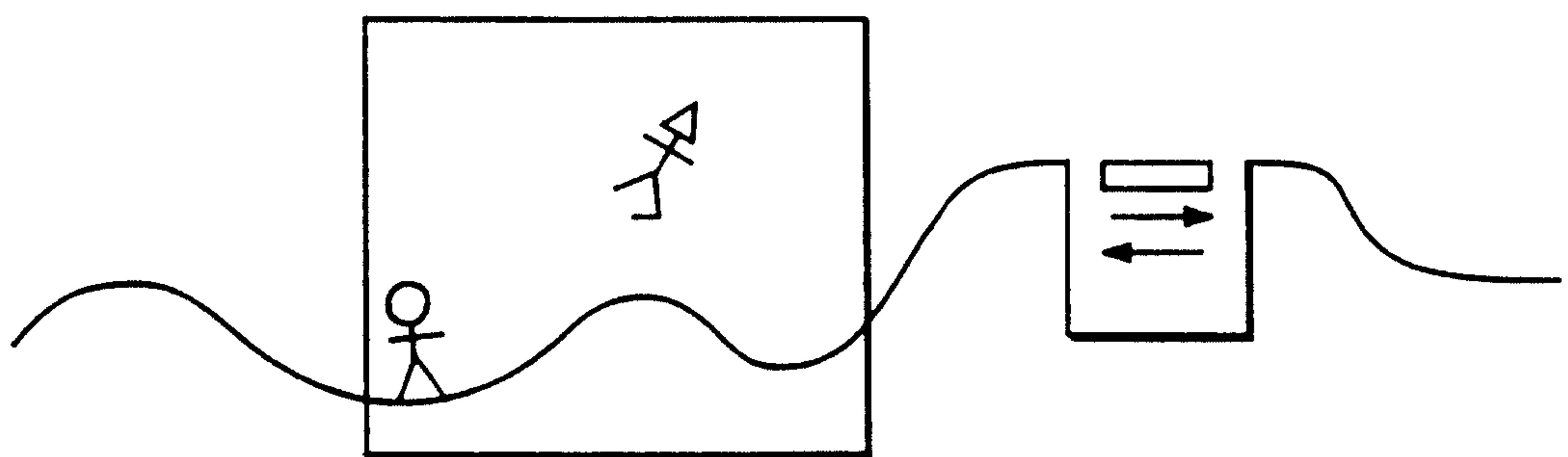


FIG. -37B

39/46

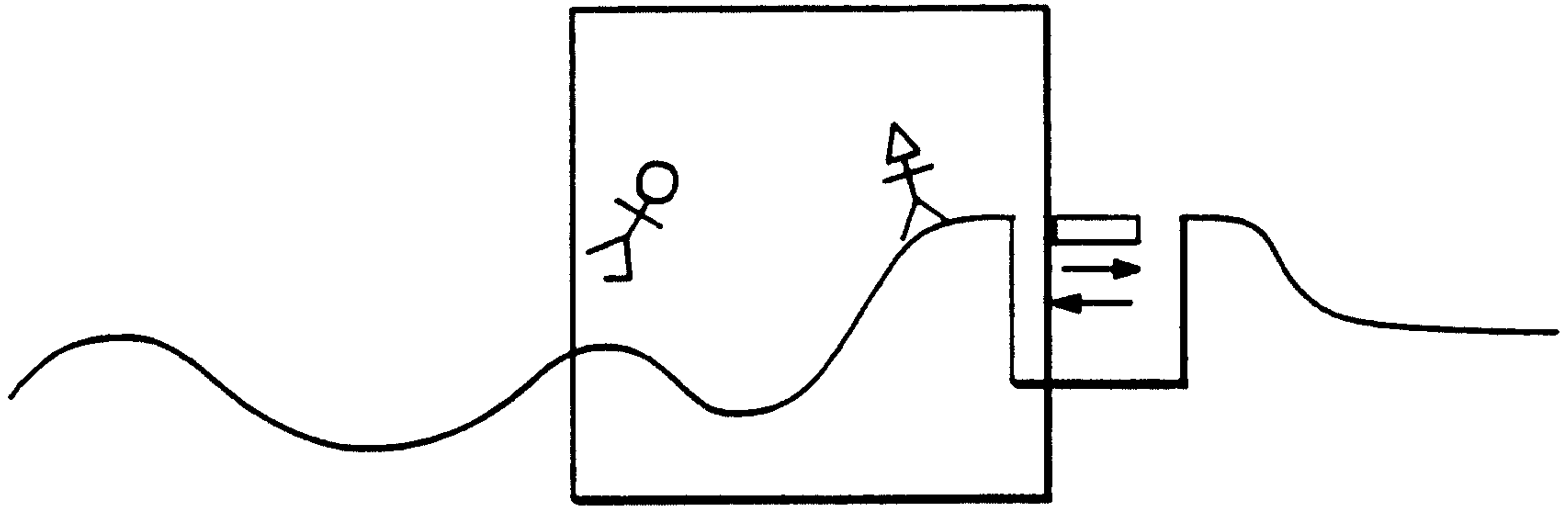


FIG. -37C

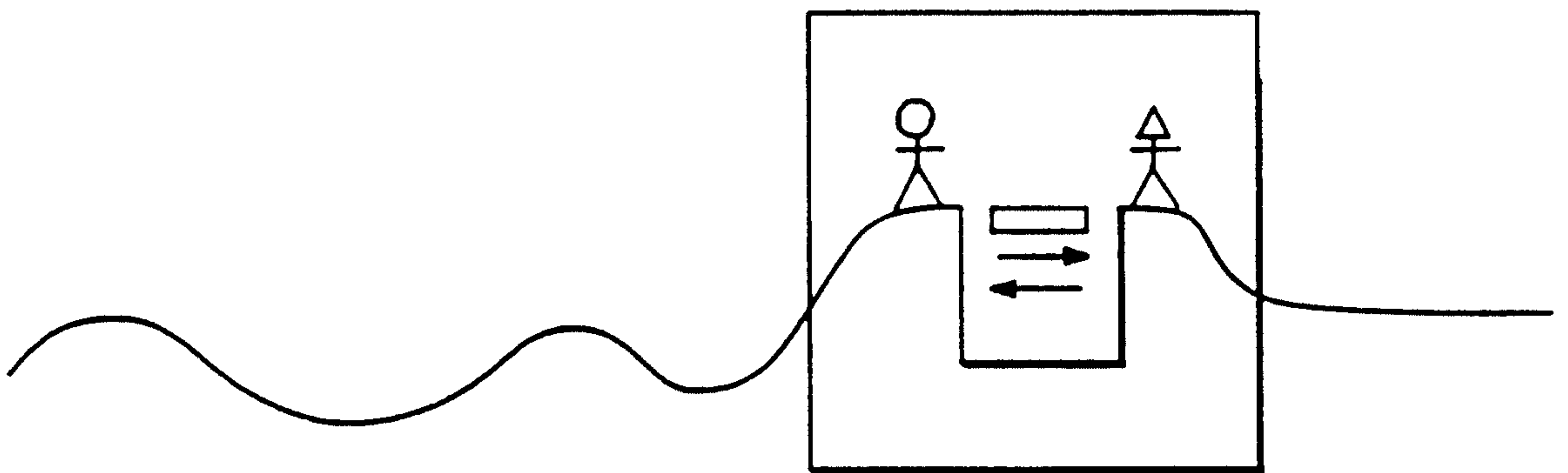


FIG. -37D

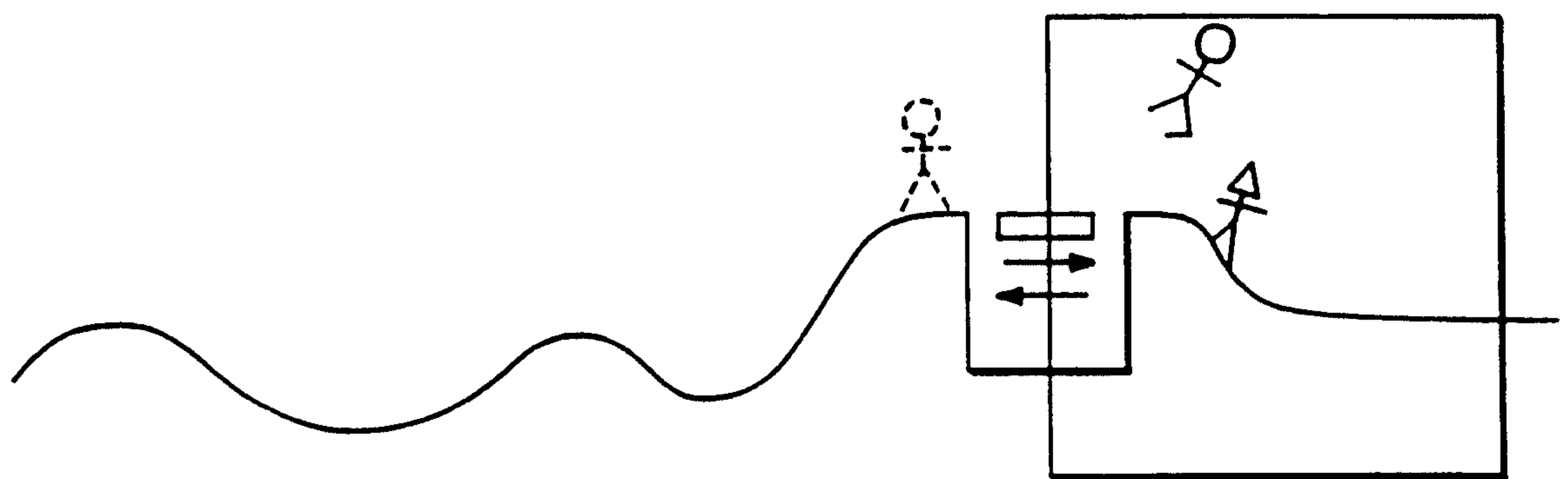


FIG. -37E

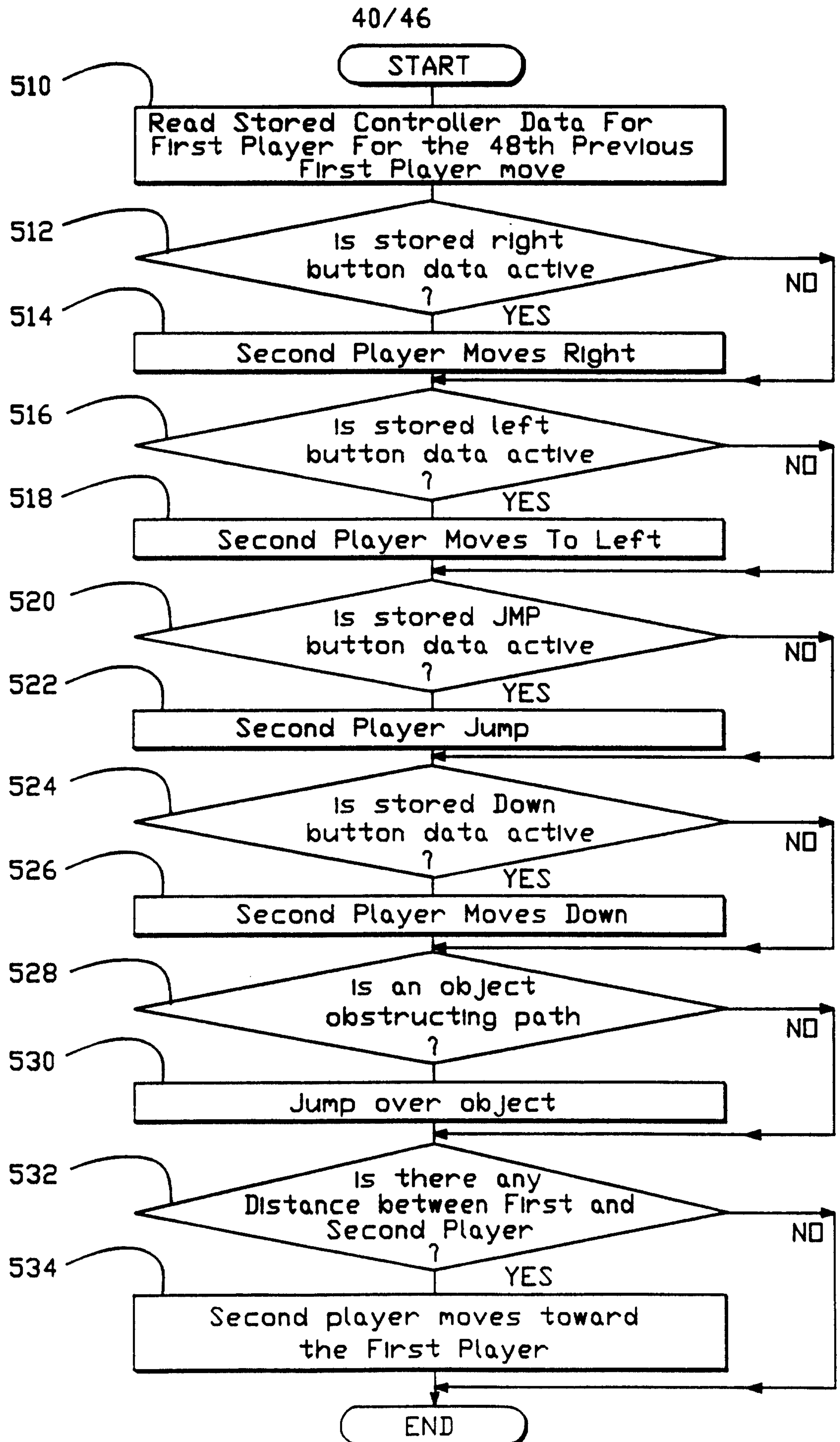


FIG. -38

41/46

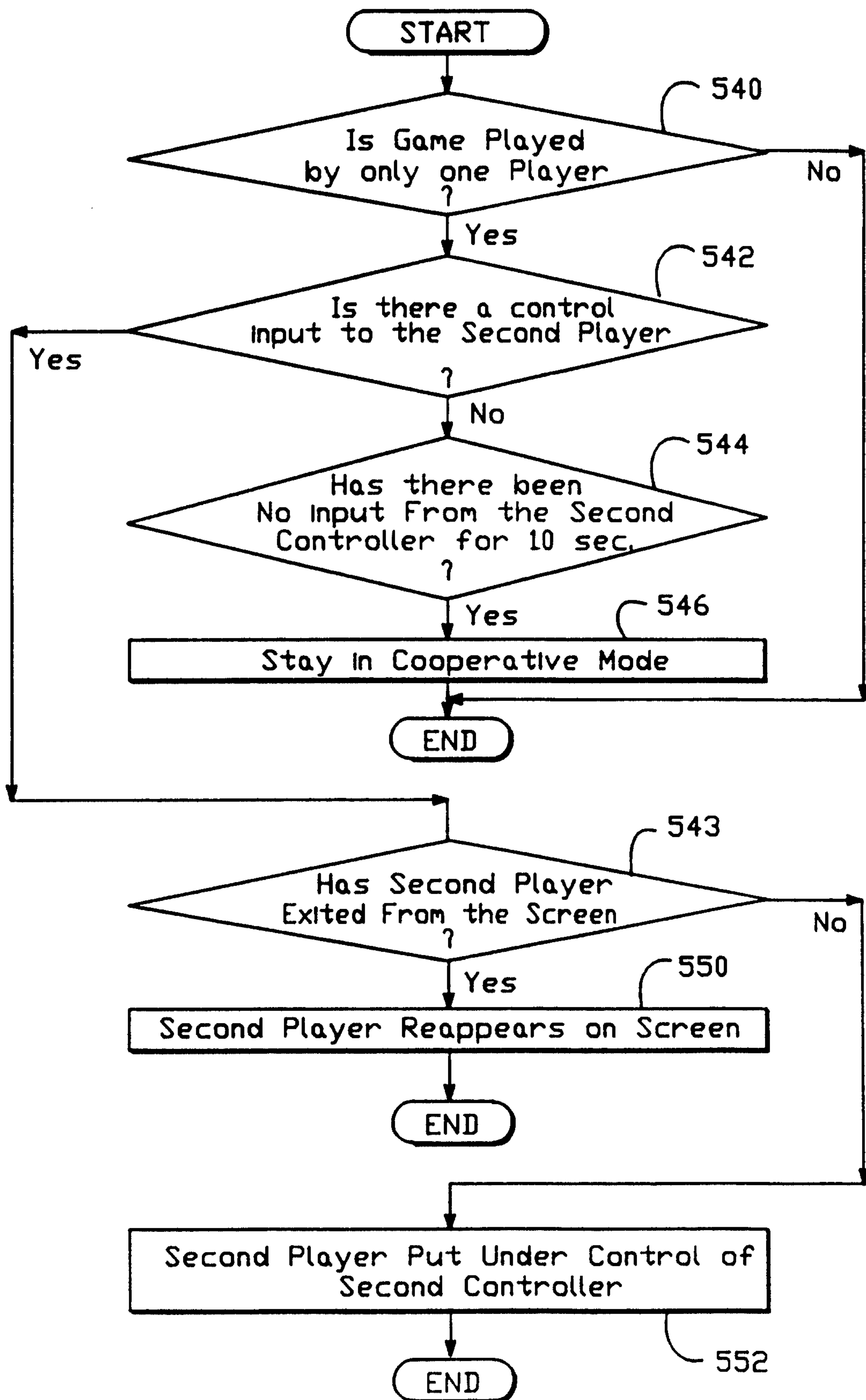


FIG.-39

42/46

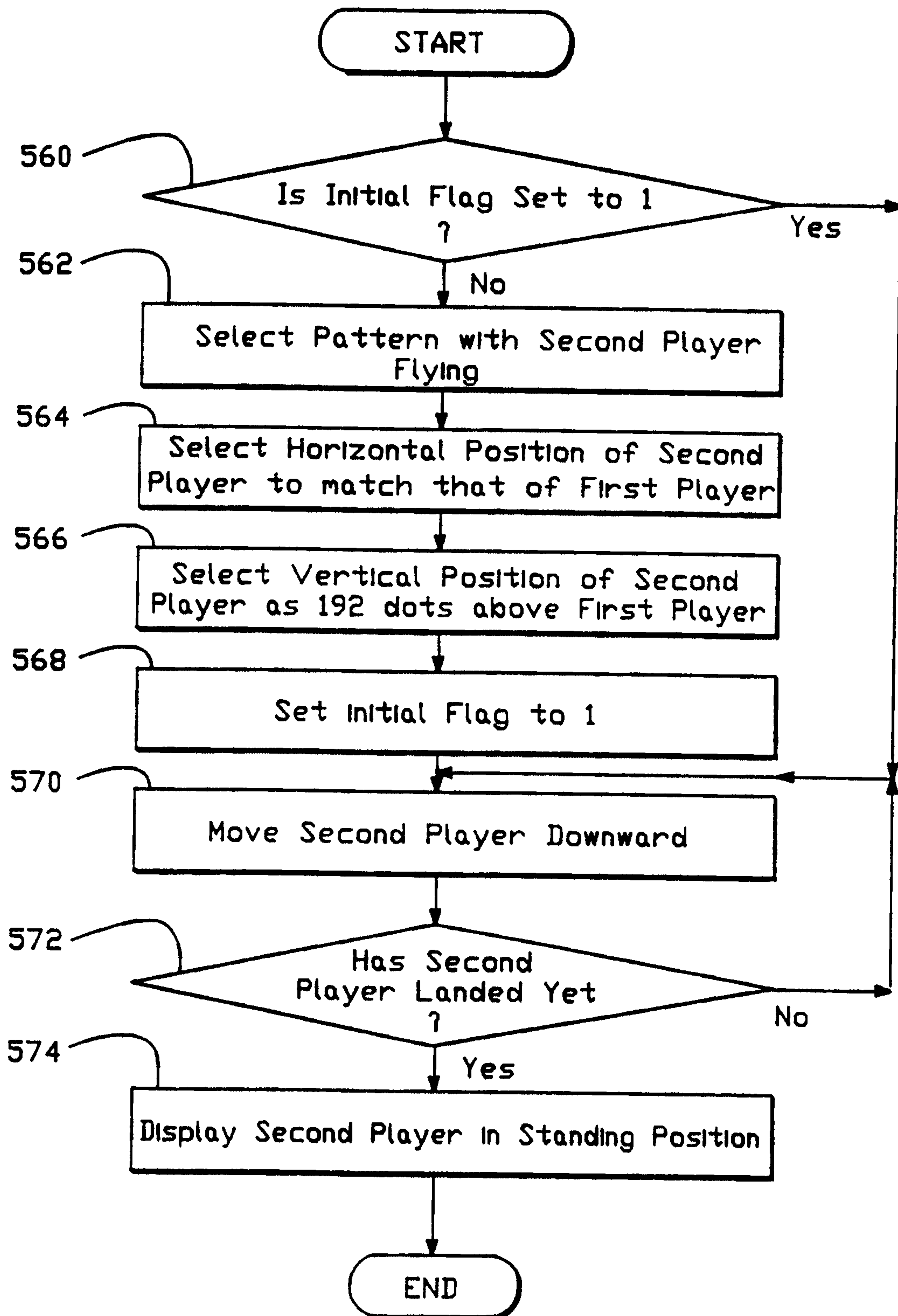


FIG. -40

43/46

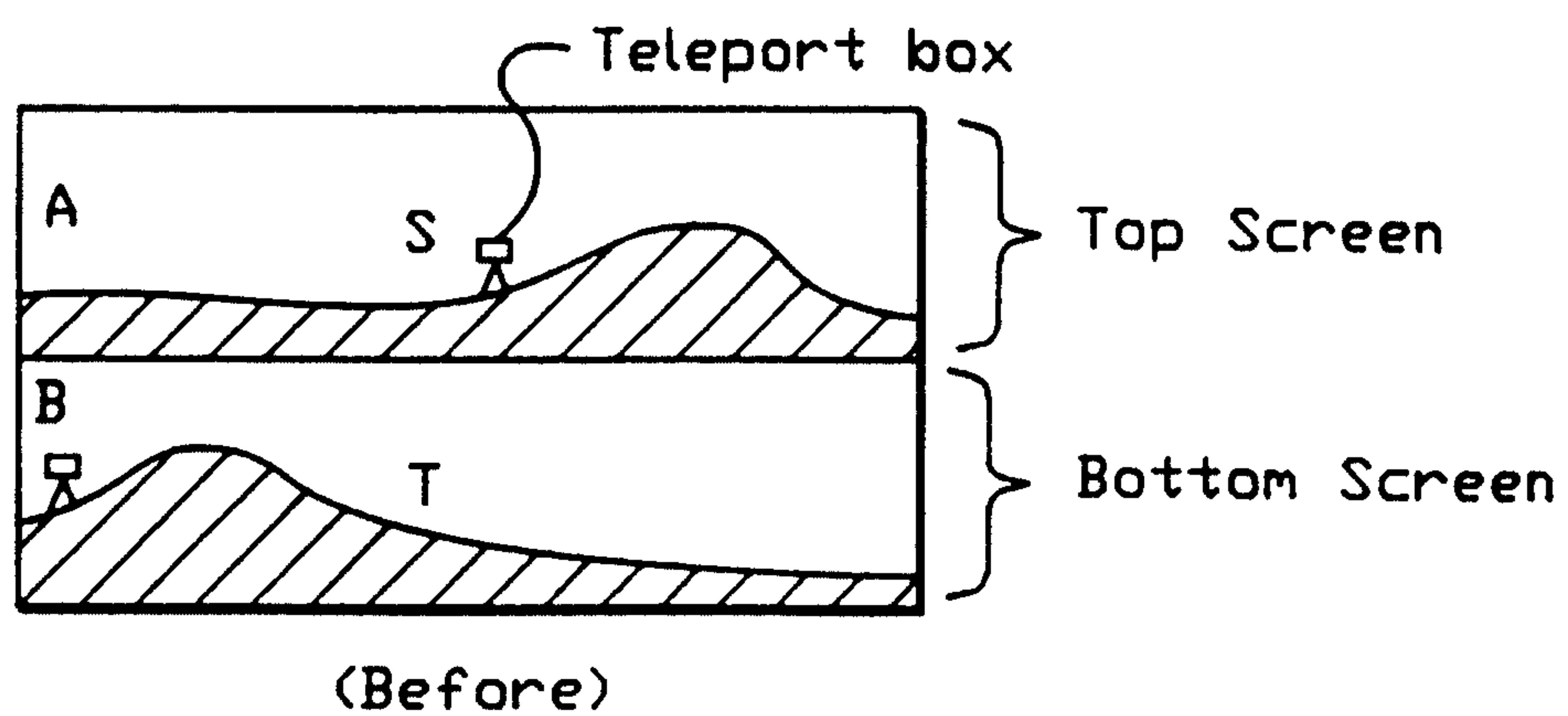


FIG.—41

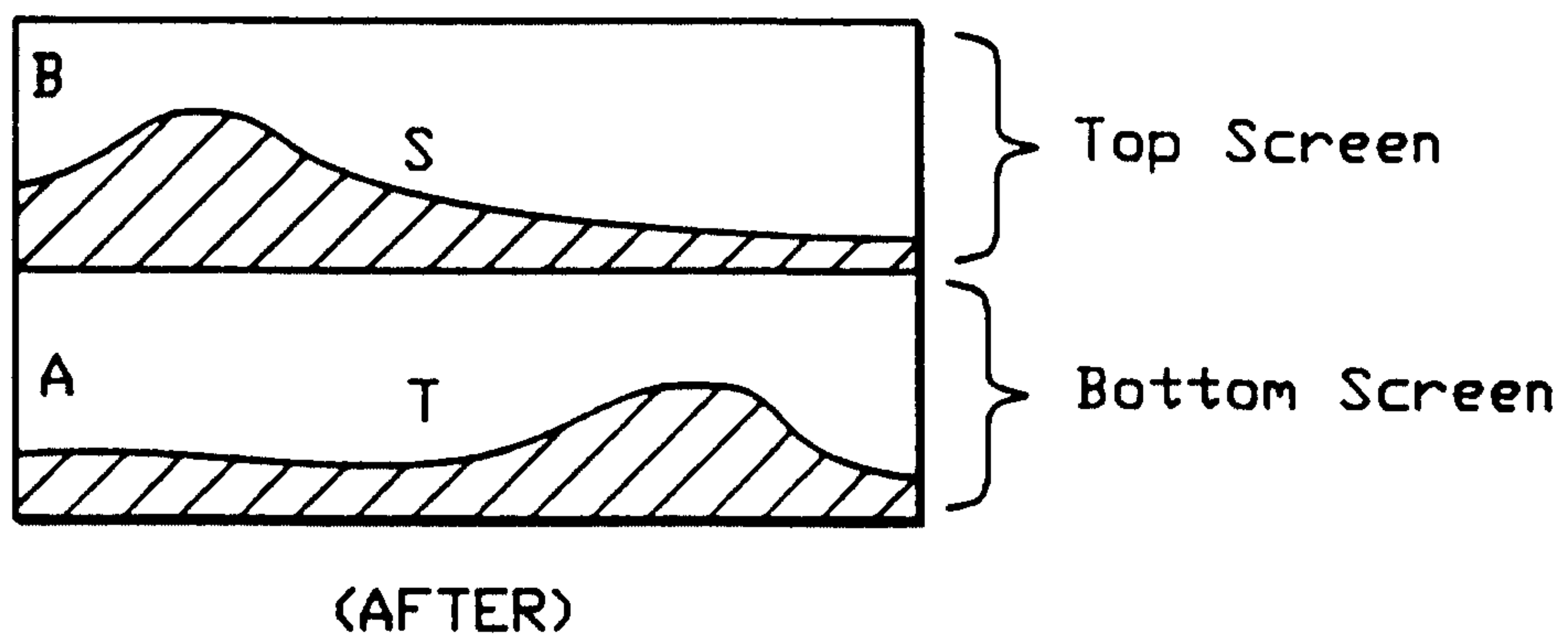


FIG.—42

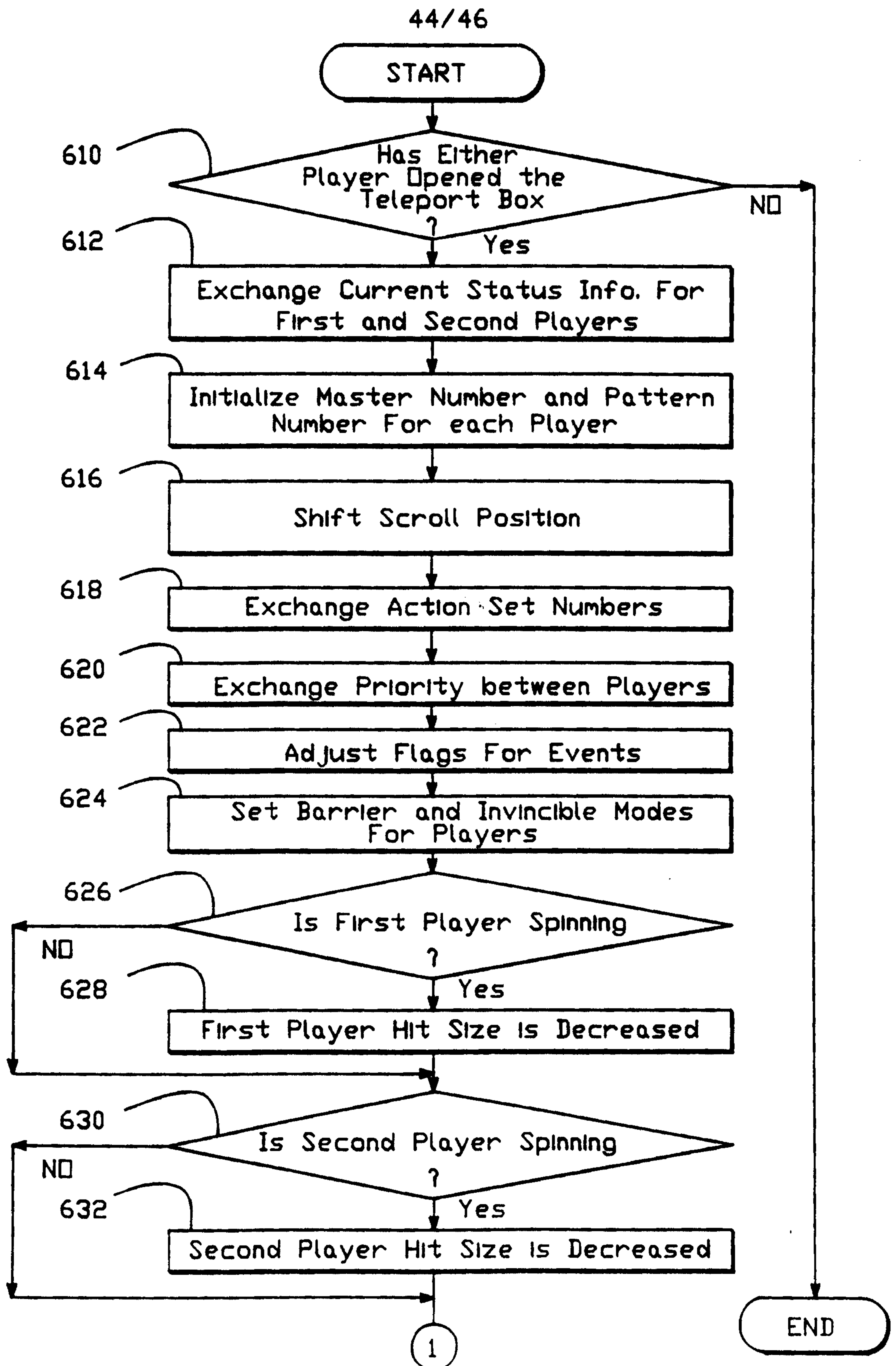


FIG.-43A

45/46

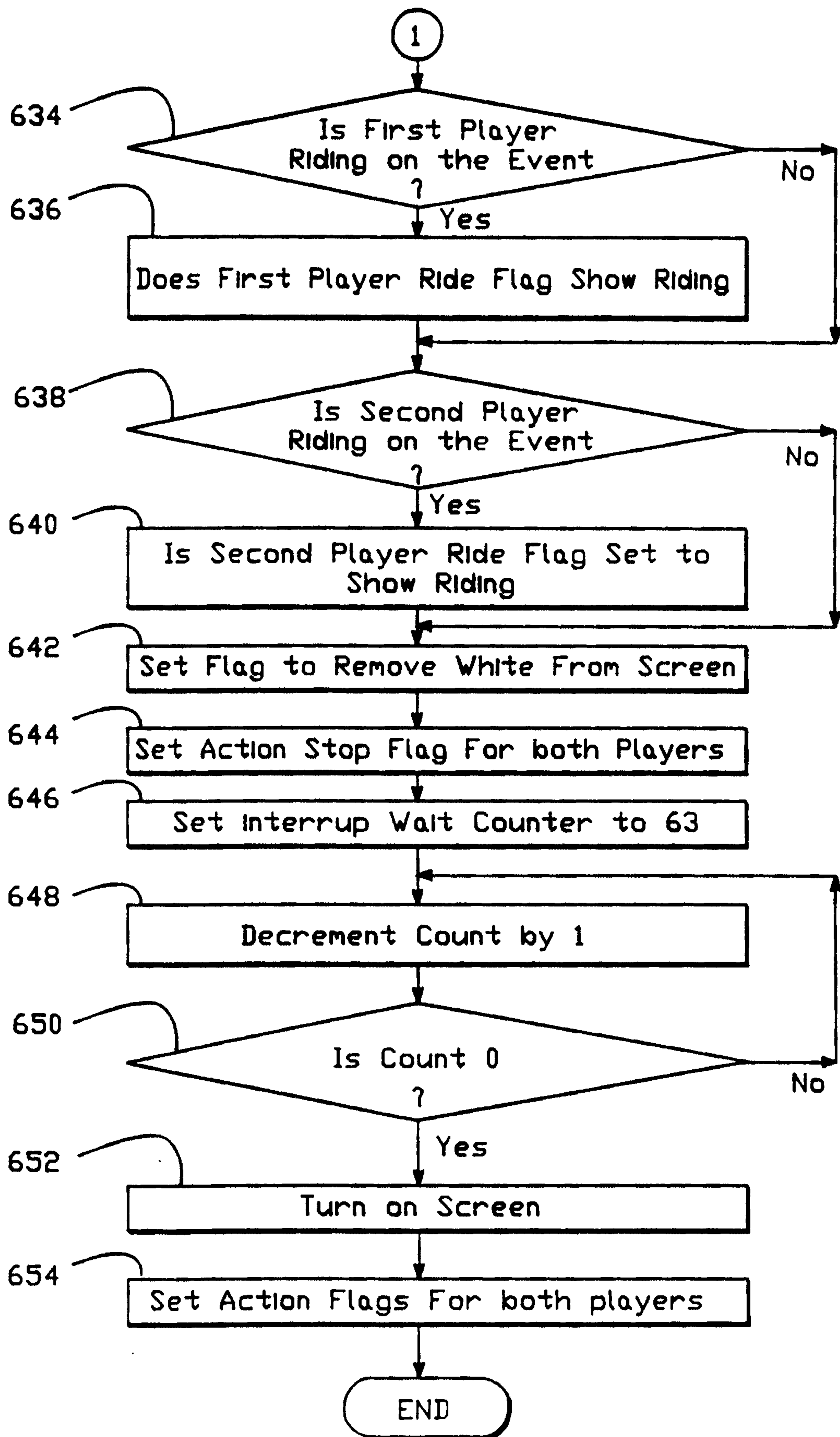
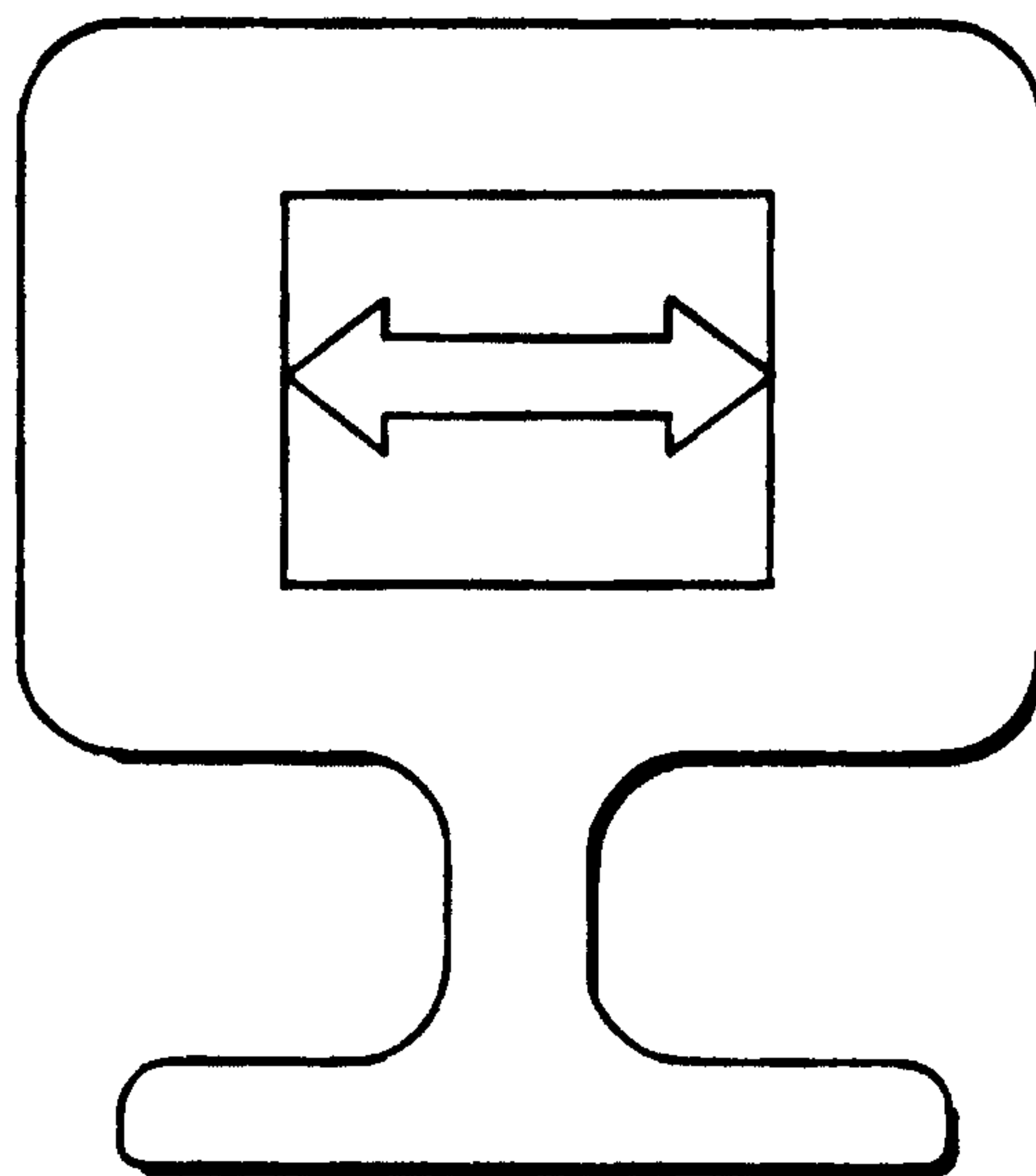


FIG.-43B

46/46



(BEFORE)

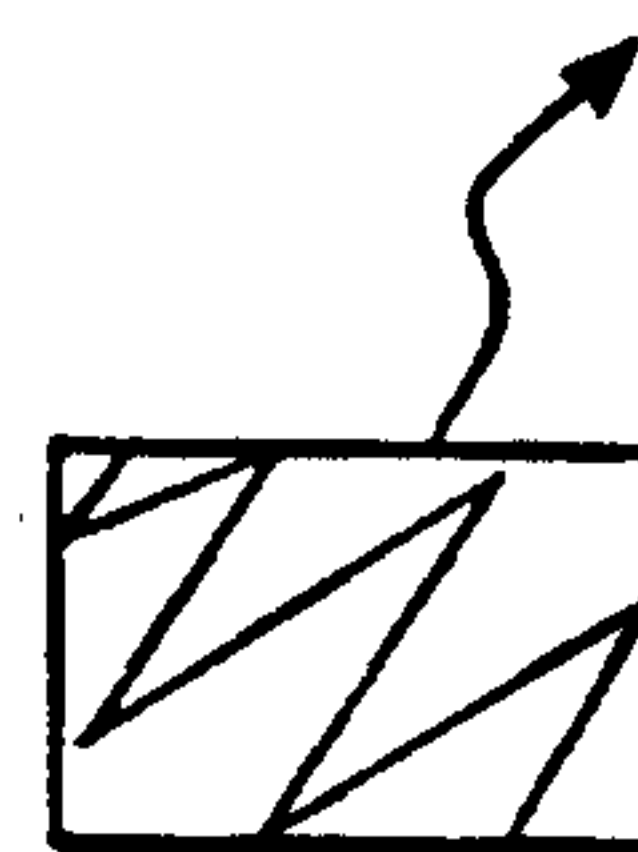
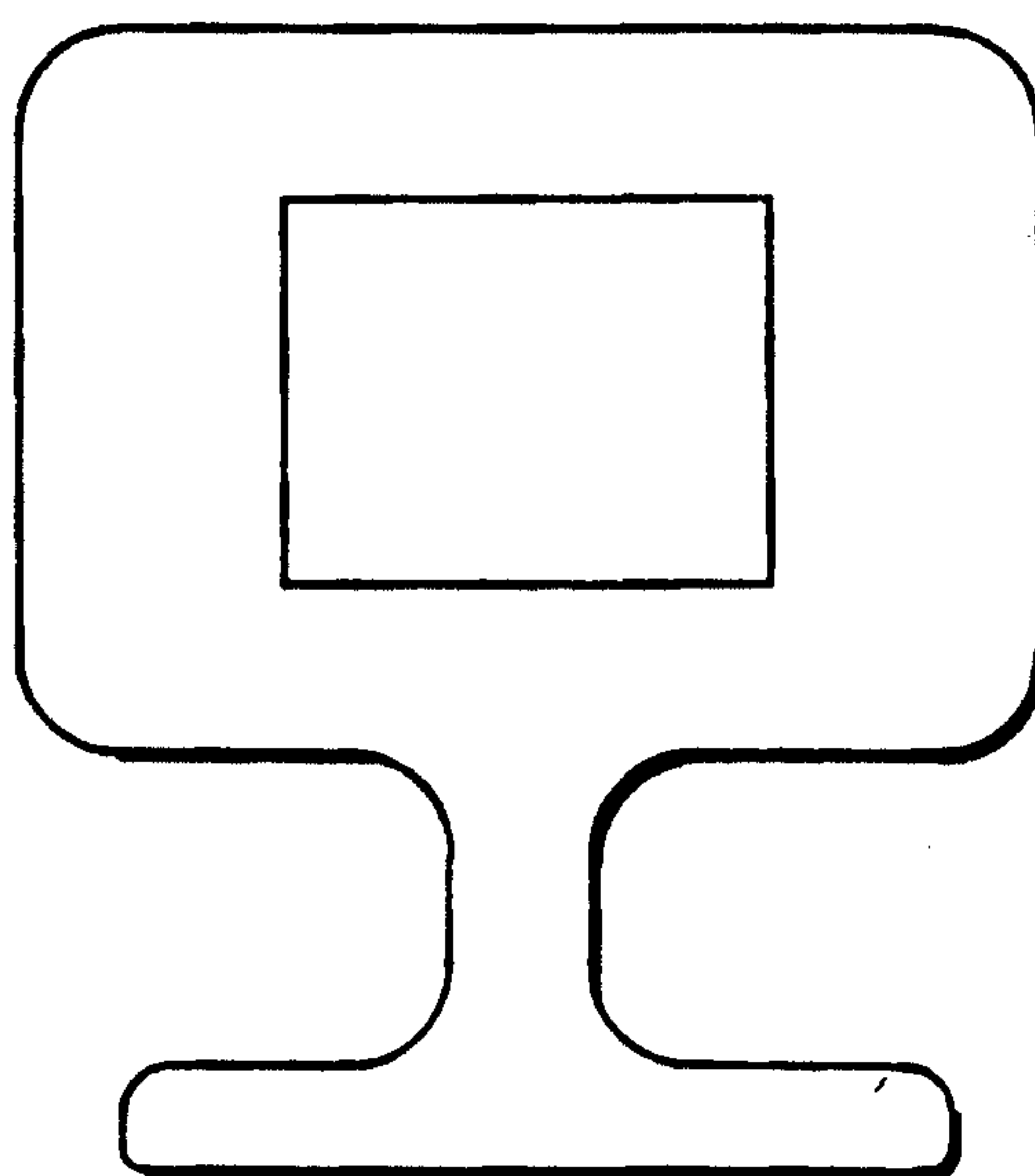


FIG.-44



(AFTER)

FIG.-45

