Sprite Control Block - Sprite Data Structure

<table>
<thead>
<tr>
<th>SCB in RAM (Linked in Painters Order)</th>
<th>First Element in list</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCB in RAM (Linked in Painters Order)</td>
<td>First Element in list</td>
</tr>
<tr>
<td>Number of bytes</td>
<td>52</td>
</tr>
<tr>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>1) 8 bits of control (SPRCTL0)</td>
<td>(See Appx 2 For Details)</td>
</tr>
<tr>
<td>(1) 8 bits of control (SPRCTL0)</td>
<td>(See Appx 2 For Details)</td>
</tr>
<tr>
<td>(1) 4 bits of control (SPRCELL)</td>
<td>(See Appx 2 For Details)</td>
</tr>
<tr>
<td>(2) 16 bits of pointer to next sprite SCB</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(2) 16 bits of address of start of Sprite Data</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(2) 16 bits of starting H Pos</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(2) 16 bits of starting V Pos</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(2) 16 bits of H size bits</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(2) 16 bits of V size bits</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(2) 16 bits of stretch value</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(2) 16 bits of tint value</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(8) 64 bits of pen palette</td>
<td>(0 = last SCB)</td>
</tr>
<tr>
<td>(1) 4 bits of collision depository</td>
<td>(0 = first SCB)</td>
</tr>
<tr>
<td>(1) 8 bits of identification number</td>
<td>(0 = first SCB)</td>
</tr>
<tr>
<td>(1) 8 bits of Z depth</td>
<td>(0 = first SCB)</td>
</tr>
<tr>
<td>(2) 16 bits of pointer to previous sprite SCB</td>
<td>(0 = first SCB)</td>
</tr>
</tbody>
</table>

(57) Abstract

A sprite control word (SCB) structure (50) includes a first 3 bytes (52) which are control words that are needed by all video objects. Byte (54) (SPRCTL0), consisting of 8 bits, defines the characteristics of a sprite. Bits B7 and B6 specify how many bits (1, 2, 3 or 4) are used to define each pixel of the sprite. Bits B2, B1 and B0 are used to define the type of sprite from a list of eight different characteristics. Byte (56) (SCBCTL1) contains the bits that inform the hardware how to use, ignore, or re-use the following bytes. Bits B5 and B4 specify how many of the optional bytes (62) in the SCB (50) are to be employed for the sprite. B3 specifies whether a color palette needs to be reloaded, or if the previous palette can be used for this sprite. Optional bytes (62) that are not used by a particular sprite do not need to be stored or processed when this data structure is used.

* See back of page
DESIGNATIONS OF "DE"

Until further notice, any designation of "DE" in any international application whose international filing date is prior to October 3, 1990, shall have effect in the territory of the Federal Republic of Germany with the exception of the territory of the former German Democratic Republic.

---

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

<table>
<thead>
<tr>
<th>AT</th>
<th>Austria</th>
<th>ES</th>
<th>Spain</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU</td>
<td>Australia</td>
<td>FI</td>
<td>Finland</td>
</tr>
<tr>
<td>BB</td>
<td>Barbados</td>
<td>FR</td>
<td>France</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
<td>GA</td>
<td>Gabon</td>
</tr>
<tr>
<td>BF</td>
<td>Burkina Faso</td>
<td>GB</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>BG</td>
<td>Bulgaria</td>
<td>GR</td>
<td>Greece</td>
</tr>
<tr>
<td>BJ</td>
<td>Benin</td>
<td>HU</td>
<td>Hungary</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
<td>IT</td>
<td>Italy</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
<td>JP</td>
<td>Japan</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
<td>KP</td>
<td>Democratic People's Republic of Korea</td>
</tr>
<tr>
<td>CG</td>
<td>Congo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
<td>KR</td>
<td>Republic of Korea</td>
</tr>
<tr>
<td>CM</td>
<td>Cameroon</td>
<td>LI</td>
<td>Liechtenstein</td>
</tr>
<tr>
<td>DE</td>
<td>Germany, Federal Republic of</td>
<td>LK</td>
<td>Sri Lanka</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
<td>LU</td>
<td>Luxembourg</td>
</tr>
<tr>
<td>MC</td>
<td>Monaco</td>
<td>MG</td>
<td>Madagascar</td>
</tr>
<tr>
<td>ML</td>
<td>Mali</td>
<td>MR</td>
<td>Mauritania</td>
</tr>
<tr>
<td>MW</td>
<td>Malawi</td>
<td>NL</td>
<td>Netherlands</td>
</tr>
<tr>
<td>NO</td>
<td>Norway</td>
<td>RO</td>
<td>Romania</td>
</tr>
<tr>
<td>SD</td>
<td>Sudan</td>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>SN</td>
<td>Senegal</td>
<td>SU</td>
<td>Soviet Union</td>
</tr>
<tr>
<td>TD</td>
<td>Chad</td>
<td>TG</td>
<td>Togo</td>
</tr>
<tr>
<td>US</td>
<td>United States of America</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SYSTEM AND METHOD FOR SPRITE CONTROL BLOCK STRUCTURE

BACKGROUND OF THE INVENTION

1. Field of the Invention:
The present invention relates generally to the way in which information used to generate a video image is stored in a memory. More particularly, it relates to a video image control block structure that only uses as much memory as is required by a particular video object. The invention further relates to a graphics display system incorporating a system and method for a sprite control block structure.

2. Description of the Prior Art:
In graphics display technology, it is conventional to generate images from component building blocks, called "sprites". The sprites define shapes used to create the image. A sprite may be an object in the image, but image objects are often made up as an assembly of sprites. The basic sprite characteristics can be modified, for example, in size, shape, orientation, color or intensity in the formation of the image. Scaling the sprite creates an impression of three dimensions in the image. Tilting and stretching creates an impression of perspective in the image. Creating images in this manner, rather than defining the image on a pixel by pixel basis, requires far less memory and processing time. As a result, most images for video games are formed by the use of sprites.
In many typical video image generation systems using sprites, a particular video object is created from two sets of data. The first set is an actual definition of the details of the image, called source image data, i.e. a collection of sprites. The second is a set of control words that define the particular characteristics of the each resultant image sprite, such as its size, shape, orientation, color or intensity. Conventional control word structures have a structure defined by every possible element used to define sprite characteristics. Such control word structures require a certain amount of memory for their storage. These structures and their memory are typically not variable in size based on the dynamic requirements of the video object currently being manipulated. If an element is not used in a sprite, data for that element still must be stored and processed, even if it is not used to create the sprite in the image. When most of the sprites in fact use most of the possible elements, little additional memory or processing time is required in this approach. However, in many images, only a few of the sprites making up the image require use of all the possible elements. For example, sprites used to define mountains and sky in a road race game might not move, while other sprites, such as those used to define trees and automobiles, will move and change in size from frame to frame. Sprites used to define background and which therefore do not change do not require use of those fields, for example, which define size changes or position changes in the image. Using a fixed data structure of the sprite control block for all of the sprites in any image will result in using substantially more memory and processing time to handle the elements than if only the elements actually necessary to define each sprite were stored and processed.

SUMMARY OF THE INVENTION
Accordingly, it is an object of this invention to provide a sprite control block data structure which does not require allocation of control word memory for features that are not used in a particular sprite of a video image.

It is another object of the invention to provide such a sprite control block data structure in which only uses as much memory as required by a particular video object.

It is a further object of the invention to provide a sprite control block data structure that does not require the processing of element information that is not used to define an image.

It is still another object of the invention to provide a graphics display system that provides real time, impression of or apparent perspective, color graphics with realistic motion incorporating such a sprite control block data structure.

It is a still further object of the invention to provide such a graphics display system in the form of a hand held unit.

The attainment of these and related objects may be achieved through use of the novel sprite control block data structure system and method herein disclosed. A sprite control block data structure system in accordance with this invention has a processing unit and a random access memory connected for bidirectional transfer of data and instructions between the random access memory and the processing unit. The processing unit is under control of a program for generating the display image. The program for generating the display image includes an instruction for defining a sprite control block structure for a first plurality of sprite control blocks. Each of the sprite control blocks comprises a second plurality of bytes, including a third plurality of bytes present in all of the first plurality of sprite control blocks, a fourth plurality of bytes optionally present in the first plurality of sprite control blocks, and at least one
control byte defining which of the fourth plurality of bytes are present in a particular one of the first plurality of sprite control blocks containing the at least one control byte.

The method of this invention for processing sprite information to generate a display image includes defining a sprite control block structure for a first plurality of sprite control blocks. Each of the sprite control blocks has a second plurality of bytes, including a third plurality of bytes present in all of the first plurality of sprite control blocks. A fourth plurality of bytes is optionally present in the first plurality of sprite control blocks. At least one control byte defines which of the fourth plurality of bytes are present in a particular one of the first plurality of sprite control blocks containing the at least one control byte. The sprite information is stored in a memory. The sprite information is supplied to a processing unit. The sprite information is used to generate the display image.

The attainment of the foregoing and related objects, advantages and features of the invention should be more readily apparent to those skilled in the art, after review of the following more detailed description of the invention, taken together with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a perspective view of a system in which the present invention is useful.

Figure 2 is a block diagram of the system shown in Figure 1.

Figure 3 is a chart showing a data structure of a sprite control block in accordance with the invention.

Figure 4 is a chart showing further details of a first portion of the sprite control block of Figure 3.

Figure 5 is a chart showing further details of a second portion of the sprite control block of Figure 3.
Figure 6 is a chart showing further details of a third portion of the sprite control block of Figure 3.

Figure 7 is a chart and flow diagram showing an image data structure for use with the sprite control block of Figure 1.

DETAILED DESCRIPTION OF THE INVENTION

Turning now to the drawings, more particularly to Figure 1, there is shown a hand held electronic game system 10 which utilizes the present invention to reduce the amount of memory storage required to define and process images, so that a conventional 6502 type microprocessor can be used to provide real time, apparent perspective, graphics for a color liquid crystal display 12 used in the system 10. The system 10 includes conventional controls 14 and redundant sets 16 of buttons for firing weapons and similar functions. In use, the game system is grasped by handles 18 and 20 in the left and right hands, respectively, in the orientation shown. The redundant sets 16 of buttons allow the system 10 to be inverted for left hand operation of the buttons. When this is done, the orientation of the images on the display is flipped, so that it appears right side up when the sets 16 of buttons are on the left side of the system 10.

Those skilled in the art of graphics processing will appreciate the demanding memory and processing requirements for presenting real time, apparent perspective, color graphics with realistic motion on the display 12. In fact, most personal computers are unable to present such realistic, apparent perspective graphics with rapid enough motion to make games interesting. Usually, only arcade games presently have such capability. This is why the displays for most personal computer based video games are crude and are only two-dimensional. Conventional hand held electronic games have even cruder, monochrome graphics. The system 10 is even more
remarkable in that the real time, apparent perspective, color graphics with realistic motion are achieved by using a conventional 6502 type microprocessor, an early microprocessor design that has been available since the late 1970s. The system and method for sprite control block data structure of this invention is one of the techniques used in the system 10 to enhance the performance of the 6502 microprocessor so that it is able to handle the graphics processing for the system 10.

Figure 2 is a block diagram of electronics 30 for the system 10. A custom microprocessor integrated circuit 32 includes a standard 65C02 microprocessor CPU cell 34 and on chip interface and support circuits. The integrated circuit 32 is connected by a control bus 36 to a custom sprite engine integrated circuit 38, which also includes switch reader circuits for the switches 14 and 16 and read only memory (ROM) reader circuits for the ROM reader 40, included in the sprite engine integrated circuit 38 due to pin limitations on the microprocessor integrated circuit 32.

The integrated circuit 32 is connected to a 64K x 8 random access memory (RAM) 42 by 8-bit address and data busses 44 and 46 and by a 3-bit RAM control bus 48. The RAM 42 houses the video buffer(s) and collision buffer in addition to the game software. The RAM 42 has a 120 nanosecond row address strobe (RAS) access time and 60 nanosecond page mode column address strobe (CAS) access time. This allows a 250 ns (4 MegaHertz) page mode memory access rate and a 312 ns (3.2 MHz) normal memory access rate.

The microprocessor integrated circuit 32 is connected to the liquid crystal display (LCD) 12 by a 4-bit video data bus 50 and an 11-bit video control bus 52. The LCD has a resolution of 160 horizontal color pixels by 102 vertical color pixels. The column drivers for the display 12 can generate 16 levels of intensity for each
pixel, resulting in a palette of 4,096 colors. The present invention provides an improved data structure for more efficient utilization of the RAM 42 and less processing time for the sprite engine 38 to define the sprites for the images on LCD 12. For purposes of this application, the remaining elements shown in Figure 2 are conventional in nature, and they therefore will not be described further.

Figure 3 shows a sprite control word structure 50 (SCB). The SCBs are linked in painter’s order, which means that objects are rendered into the display buffer in the same way a painter paints a canvas, i.e., background first, with objects in the foreground painted over the background. The first 3 bytes 52 are control words that are needed by all video objects. Byte 54 (SPRCTL0), consisting of 8 bits, defines the characteristics of a sprite as shown in Figure 4. Bits B7 and B6 specify how many bits (1, 2, 3 or 4) are used to define each pixel of a particular sprite, based on the number of colors in the sprite. Note that bits B5 and B4 define whether the sprite is flipped. These bits are useful for reducing data requirements by turning an object left and right and up and down, rather than redefining the whole object to display it in a changed orientation. Bits B2, B1 and B0 are used to define the type of sprite from the list of eight different characteristics shown in Figure 4.

Byte 56 (SCBCTL1), details of which are shown in Figure 5, contains the bits that inform the hardware how to use, ignore, or re-use the following bytes. B7 define the literal attribute as normal or totally literal. In this system, the designation as normal means a combination of packed and literal. B6 specifies a choice of sprite sizing algorithms. The two choices are, respectively, a purely mathematical method and a column and row selection method, which are artist selectable. Bits B5 and B4 specify how many of the optional bytes in the SCB 50 are
to be employed for the sprite, using the choices shown in Figure 5. B3 specifies whether a color palette needs to be reloaded, or if the previous palette can be used for this sprite. B2 allows a sprite to be skipped. B1 and B0 specify beginning drawing directions.

Byte 58, details of which are shown in Figure 6, specifies collision information for the sprite. Bit B5 specifies whether the sprite is one for which collisions occur. A typical example of a sprite with which collisions do not occur is a sprite defining a cloud or a dashboard instrument. Bits B3, B2, B1 and B0 specify a collision number, which signifies who hit whom.

In addition to the bytes 52, each SCB 50 must include bytes 60. Bytes 62 are optionally present in the SCB 50, depending on the designations for bits B5, B4 and B3 in byte 56. The bytes 62 are ordered in a sequence of most commonly used to least commonly used. For a particular byte 62 to be present and used, all of the bytes 62 above it must also be present. Thus, if a stretch value is present and used, the horizontal size bits and the vertical size bits must also be present. Byte 64 provides collision depository information, an answer to a collision, and may be positioned arbitrarily in the SCB 50, but must be fixed for any particular game. Bytes 66 are some possible additional elements in the SCB 50. These and any others will be ignored by the hardware, but may provide useful information for the software, such as an identification of the previous sprite control block. Bytes 52-62 must be provided (if present) in the order shown in Figure 3.

Figure 7 shows an example of SCB image data 70 structure that is used with the SCB 50. Offsets to each line of sprite data are given at 72. The structure of an ordinary data packet is shown at 74, and a totally literal sprite data packet structure is given at 76. Changing direction of sprite painting is specified at 78.
It should now be readily apparent to those skilled in the art that a novel system and method for sprite control block structure capable of achieving the stated objects of the invention has been provided. The sprite control block data structure does not require allocation of control word memory for features that are not used in a particular sprite of a video image. The sprite control block data structure therefore only uses as much memory as required by a particular video object. The sprite control block data structure therefore also does not require the processing of element information that is not used to define an image. The sprite control block data structure is particularly useful in a hand held graphics display system that provides real time, perspective, color graphics with realistic motion.

It should further be apparent to those skilled in the art that various changes in form and details of the invention as shown and described may be made. It is intended that such changes be included within the spirit and scope of the claims appended hereto.
WHAT IS CLAIMED IS:

1. A system for processing sprite information to generate a display image, which comprises a processing unit, a random access memory connected for bidirectional transfer of data and instructions between said random access memory and said processing unit, said processing unit being under control of a program for generating the display image, the program for generating the display image including an instruction for defining a sprite control block structure for a first plurality of sprite control blocks, each comprising a second plurality of bytes, including a third plurality of bytes present in all of said first plurality of sprite control blocks, a fourth plurality of bytes optionally present in said first plurality of sprite control blocks, and at least one control byte defining which of said fourth plurality of bytes are present in a particular one of said first plurality of sprite control blocks containing the at least one control byte.

2. The system for processing sprite information to generate a display image of Claim 1 in which said fourth plurality of bytes are present in said plurality of sprite control blocks in sequential order from most commonly used to least commonly used.

3. The system for processing sprite information to generate a display image of Claim 1 in which said fourth plurality of bytes include horizontal size bytes.

4. The system for processing sprite information to generate a display image of Claim 3 in which said fourth plurality of bytes include vertical size bytes.

5. The system for processing sprite information to generate a display image of Claim 4 in which said fourth
plurality of bytes include sprite stretch value bytes.

6. The system for processing sprite information to generate a display image of Claim 5 in which said fourth plurality of bytes include sprite tilt value bytes.

7. The system for processing sprite information to generate a display image of Claim 6 in which said fourth plurality of bytes include color selection bytes.

8. The system for processing sprite information to generate a display image of Claim 1 in which the program to provide the display includes a plurality of instructions to provide real time, apparent perspective, moving, color images.

9. A hand held system for providing real time, apparent perspective, moving, color images, which comprises a housing dimensioned and configured to be held by a user, a display on a surface of said housing visible to the user when holding said housing, controls on said housing for operating said system, said system including a processing unit, a random access memory connected to supply data and instructions to said processing unit, said processing unit being under control of a program for generating the images, the program for generating the images including an instruction for defining a sprite control block structure for a first plurality of sprite control blocks, each comprising a second plurality of bytes, including a third plurality of bytes present in all of said first plurality of sprite control blocks, a fourth plurality of bytes optionally present in said first plurality of sprite control blocks, and at least one control byte defining which of said fourth plurality of bytes are present in a particular one of said first plurality of sprite control blocks containing the at least
one control byte.

10. The hand held system for providing real time, apparent perspective, moving, color images of Claim 9 in which said fourth plurality of bytes are present in said plurality of sprite control blocks in sequential order from most commonly used to least commonly used.

11. The hand held system for providing real time, apparent perspective, moving, color images of Claim 10 in which said fourth plurality of bytes include horizontal size bytes.

12. The hand held system for providing real time, apparent perspective, moving, color images of Claim 11 in which said fourth plurality of bytes include vertical size bytes.

13. The hand held system for providing real time, apparent perspective, moving, color images of Claim 12 in which said fourth plurality of bytes include sprite stretch value bytes.

14. The hand held system for providing real time, apparent perspective, moving, color images of Claim 13 in which said fourth plurality of bytes include sprite tilt value bytes.

15. The hand held system for providing real time, apparent perspective, moving, color images of Claim 14 in which said fourth plurality of bytes include color selection bytes.

16. A method for processing sprite information to generate a display image, which comprises defining a sprite control block structure for a first plurality of
sprite control blocks, each comprising a second plurality of bytes, including a third plurality of bytes present in all of said first plurality of sprite control blocks, a fourth plurality of bytes optionally present in said first plurality of sprite control blocks, and at least one control byte defining which of said fourth plurality of bytes are present in a particular one of said first plurality of sprite control blocks containing the at least one control byte, storing the sprite information in a memory, supplying the sprite information to a processing unit, and using the sprite information to generate the display image.

17. The method for processing sprite information to generate a display image of Claim 16 in which said fourth plurality of bytes are present in said plurality of sprite control blocks in sequential order from most commonly used to least commonly used.

18. The method for processing sprite information to generate a display image of Claim 17 in which said fourth plurality of bytes include horizontal size bytes.

19. The method for processing sprite information to generate a display image of Claim 18 in which said fourth plurality of bytes include vertical size bytes.

20. The method for processing sprite information to generate a display image of Claim 19 in which said fourth plurality of bytes include sprite stretch value bytes.

21. The method for processing sprite information to generate a display image of Claim 20 in which said fourth plurality of bytes include sprite tilt value bytes.

22. The method for processing sprite information to
generate a display image of Claim 21 in which said fourth plurality of bytes include color selection bytes.

23. The method for processing sprite information to generate a display image of Claim 16 in which the sprite information is used to generate the display with a plurality of instructions to provide real time, apparent perspective, moving, color images.
FIG.-2
SUBSTITUTE SHEET
**Sprite Control Block - Sprite Data Structure**

SCB in RAM (Linked in Painters Order)

- Number of bytes
- First Element in list
- (1) 8 bits of control (SPRCTRLD)
- (1) 8 bits of control (SCBCTL1)
- (1) 4 bits of control (SPRCDLL)
- (2) 16 bits of pointer to next sprite SCB
  - (0 = last SCB)
- (2) 16 bits of address of start of Sprite Data
- (2) 16 bits of starting H Pos
- (2) 16 bits of starting V Pos
- (2) 16 bits of H size bits
- (2) 16 bits of V size bits
- (2) 16 bits of stretch value
- (2) 16 bits of tilt value
- (8) 64 bits of pen palette
- (1) 4 bits of collision depository
- (1) 8 bits of identification Number
- (1) 8 bits of Z depth
- (2) 16 bits of pointer to previous sprite SCB (0 = first SCB)

**FIG.-3**

**FC80 = SPRCTRLD.**

Sprite Control Bits 0. (W) (U) reset = x,x,x,x,x,x,x,x,x

- B7,B6 = bits/pixel-1 (1,2,3,4)
- B5 = H flip, 0 = not flipped
- B4 = V flip, 0 = not flipped
- B3 = reserved
- B2,B1,B0 = Sprite Type
  - 7 → shadow
  - 6 → exclusive-or shadow
  - 5 → non-collidable
  - 4 → normal
  - 3 → boundary
  - 2 → boundary-shadow
  - 1 → background-no collision
  - 0 → background, shadow

**FIG.-4**

SUBSTITUTE SHEET
**FC81 = SPRCTL1.**

**Sprite Control Bits 1.** (W) (U) reset = x,x,x,x,x,x,x,x

- B7 = literal attribute, 0=normal, 1=totally literal
- B6 = Sizing algorithm choice, 0=adder (algo 4), 1=shifter (algo 3)
- B5,B4 = Reloadable depth:
  - 0= none
  - 1= Hsize, Vsize.
  - 2= Hsize, Vsize, Stretch.
  - 3= Hsize, Vsize, Stretch, Trtl.
- B3 = Palette re-load, 0=reload the palette, 1=use existing palette.
- B2 = Skipsprite, 1=skip this sprite, 0=use this sprite.
- B1 = Start drawing up (1=up, 0=down)
- B0 = Start drawing left (1=left, 0=right)

**FIG.-5**

**FC82 = SPRCOLL.**

**Sprite Collision Number.** (W) (U) reset = x,x,x,x,x,x,x,x

- B7,B6 = 0
- B5 = don’t collide, 1=don’t collide with this sprite.
- B4 = 0
- B3,B2,B1,B0=number

**FIG.-6**

**SCB Image Data Structure**

**Ordinary Data Packet**

- Packed:
  - 1) bit of Literal
  - (4) bits of repeat count (+1)
  - (1,2,3,4) bits per pixel of data

**Totally Literal Sprite Data Packet**

- Pixel count is related to offset (see spec)
- (1,2,3,4) bits per pixel of data (0 fill to end of byte)

**Direction of Painting**

- Sprite painting is done in 1 to 4 quadrants. The order of quadrant painting is: Down/Right, U/R, U/L, D/L.
- The starting quadrant is specified in one of the control bytes. An offset of +1 instructs the hardware to changes to the next drawing direction.

**FIG.-7**

**SUBSTITUTE SHEET**