A GUIDE TO OMNI

JANUARY 13, 1984

This document and project is dedicated to Donald Teiser, without whose judgement, guidance and protection this effort would have resulted in nothing, whatsoever.

I extend thanks to the following people for their help:

Penny Burton, my wife, who lent her name and support and who forgave the pleasures of marriage on many occasions for this endeavor,

Vivian Filipak, my mother, who lent her name and her son to this,

Arlen Olive who provided technical assistance and lent his daughter's name, Heather,

Akio Tanaka and Eric Breeze for their good work,

John Seghers for the MAPping algorithm and

President Ronald Raygun without whom America would have surely fallen to the heathen communists.

TO THE READER

I apologize for any omissions I have made or any confusion resulting from my terminology, explanations or organization. If you are reading this document for the first time, my heart goes out to you. Good luck! Believe me, it was as hard to write as it is to read. I trust that by your tenth reading, most of the features of OMNI will come clear.

If you have any suggestions or criticisms regarding this document or the product, you will find me an enthusiastic listener.

Mark Filipak

OMNI, A THREE DIMENSIONAL VIDEO GRAPHICS SYSTEM

TABLE OF CONTENTS:

SYSTEM FEATURES

THE DISPLAY

High resolution 648 pixel/line by 488 line screen (NTSC) where one pixel is equal to 0.03" on a 25" television,

Square pixels,

Single pixel position resolution resulting in 648 positions across the visible portion of the screen (for a 25" television, this translates to .03" per increment of position),

Two pixel resolution in color resulting in 324 color changes across the visible portion of the screen,

Single pixel resolution in intensity resulting in 648 intensity changes across the visible portion of the screen,

THE PROGRAMMING ENVIRONMENT

True X,Y,Z three dimensional coordinate system allowing the program to 'view' the space and manipulate objects in true 'third person' perspective,

256 levels of depth into the screen (Z-coordinate),

Automatic display prioritization to generate the 'first person' view of the three dimensional space for presentation on the TV,

The 648x488x256 display is within a 2048x1024x512 virtual space to simplify scrolling along and movement in X, Y & Z,

THE OBJECTS

True sprite type graphics objects defined by three-dimensional position (X,Y,Z) and object height,

Display priority can be changed by merely moving an object in Z with a single CPU store as opposed to a fixed priority (which means that all objects to be reshuffled by the program) or link list priority (which means that the link list must be maintained by the program),

Up to 18,432 independent (visible) sprites (49,152 virtual sprites ready for scrolling into the visible screen) which can be used as either motion sprites or playfield sprites without differentiation on the hardware level allowing for maximum flexability in programming,

Two classes of sprites (color & intensity),

Nine types of sprites with the data densities and bandwidth for each optimized for broadcast television systems (NTSC, PAL & SECAM),

Pixel transparency control for all sprite types,

Sprites can be grouped together to form large 3D objects which can then be repositioned with only three CPU stores,

Sprites can be laminated one upon another to add detailed sections to otherwise low detail areas,

Playfield sprites can easily be used to create a 3D playfield with up to 256 levels of foreground/background objects,

Sprites are generated and regenerated without CPU involvement, without matrix transforms and without peripheral math packs,

Anti-aliasing designed into objects by the graphic artist in a stright forward, easily understood and predictable manner,

Eighty character text with a dynamically redefinable character set,

THE OUTPUT

Programmable pixel clock to bring text into registration with the shadow mask to produce optimal characters on most televisions,

Programmable palette gives the graphics designer full control over all aspects of pixel color (hue, degree of saturation and shade),

2794 hue/saturation/shade combinations in the palette (ie, 203 hues with an average of 1.6 degrees of saturation each and an average of 8-1/2 shades each) for use by color sprites plus an additional 16 intensity levels per pixel for use by intensity sprites for a total of 44,704 hue/saturation/shade/intensity combinations,

NUMBER OF NUMBER OF NUMBER OF HUE/SAT/

COLOR	HUES (% OF TOTAL)	HUE/SAT COMBINATIONS	SHADE COMBINATIONS (% OF TOTAL)
grey	1 (0.5)	1	24 (0.9)
red	42 (20.7)	78	689 (24.7)
yellow	38 (18.7)	59	524 (18.7)
green	47 (23.2)	72	595 (21.3)
cyan	26 (12.8)	45	390 (13.9)
blue	26 (12.8)	41	323 (11.6)
magenta	23 (11.3)	30	249 (8.9)
	203 (100.0)	326	2794 (100.0)

Enhanced color resolution from 90 degrees to 270 degrees of the chrominance phase spectrum so that twice as many reds, yellows, greens, and luminance levels can be created than would otherwise be possible,

Fully interlaced repeat field displays for compatability with videodisc and other electronic media,

The composite video is generated synthetically and in baseband so that the signal is ready to be injected into the channel 2/3 moduator without color sub-carrier quadrature modulators, ratioing circuits or color phase delay lines thereby reducing parts count and cost, component complexity, quality assurance overhead, frequency alignment overhead, failure rates & color drift between samples and over time,

Automatically detects the presence of an external video input (ie, videodisc, etc.) synchronizes to its signal and displays it as background,

A single system clock frequency adjustment at the end of the assembly line simultaneously aligns the color burst frequency, the color phase circuity, the color sub-carrier frequency & the scan, line and field counters (in essence, everything except the channel 2/3 modulator and the audio sub-carrier),

Four outputs available:

baseband NTSC (or PAL or SECAM),
modulated NTSC (or PAL or SECAM),
RGB and
R-Y,B-Y,Y (or U,V,Y),

GENERAL FEATURES

Distributed processing system architecture with the graphics subsystem separate from the main system allowing the CPU to run at full speed without wait states or halts,

The use of separate sprite types for color and intensity results in a 50 percent increase in memory utilization allowing the use of slow and inexpensive graphics memory,

The system is modular and expandable,

The system is generalized so that it can be used in a wide spectrum of products,

Custom chips utilize hardwired logic (not microcoded) allowing relatively low clock rate permitting larger chip geometry resulting in increased yield and

Custom graphics chip designed using standard cell technology with spares on chip which can be used as needed to further increase yield.

SYSTEM LIMITATIONS/DRAWBACKS

Rotations must be accomplished by the CPU (by rotating the graphics),

Zooming (or shinking) of sprites moving toward (or away from) the screen must be accomplished by the CPU (by zooming or shrinking the graphics),

True perspective positioning must be done by the CPU and

The hardwiring of logic in the custom chips (as opposed to micro-coded logic) will make any modifications to these chips difficult, time consuming and costly.

SYSTEM DESCRIPTION

THE MAIN SYSTEM

1, a sixteen bit CPU.

2, 966,656 words of system memory space consisting of:

16K words of Operating System ROM for system operation, interrupt processing, input/output management (contoller routines, sound routines, videodisc handlers, playcable loaders, etc.), graphics routines & software signiture,

16K words of Operating System EEPROM (electrically eraseable programmable read only memory) for game parameter store which can remember high scores, skill level, game progress, etc. so that games can be resumed after power has been turned off for periods of

up to ten years,

16K words of memory mapped I/O,

885K words of mixed media/system RAM (media can be ROM as in our present products or media subsystem consisting of videodisc reader, playcable loader, etc.) and

52K words of biphased video subsystem ram (see Video Subsystem description on the next page) which can be read or written to by the CPU at any time;

3, 15,810,560 words of spare addressing space for expansion and

4, VIVIAN, a custom chip for memory management and DMA functions (it can be dynamically configured using micro-code stores from the CPU to allow one basic architecture to handle a broad mix of memory and I/O configurations and speeds).

THE VIDEO SUBSYSTEM

1, 48K words of graphics RAM containing sprite graphics and parameters from the CPU;

2, from one to three PENNYs, a custom chip, each of which performs the following functions:

simultaneous generation of 32 sprites using the parameters and data stored in graphics RAM,

reuse of each sprite generator up to 82 times per screen (assuming all are single line sprites),

programmable grouping of sprites to form larger pseudo 3D objects in X,Y,Z,

automatic visual prioritization (in Z) for all sprites within each chip with transparency pixels allowing any sprites 'behind' to show through,

support of a virtual space over eight times larger than display space to ease program maintenance of sprite positions,

support of a display space larger than the actual screen to simplify simultaneous X,Y scrolling,

output of a four bit color (index) per pixel on the fly, with a color index of zero designating transparency, for use by the palette RAM and

output of a four bit intensity per pixel on the fly, with a intensity of zero designating full stored intensity for use by the HEATHER chip;

3, 4K words of color palette RAM containing chrominance and luminance selection data from the CPU to the HEATHER chip with an intermediate pixel by pixel lookup supplied on the address lines from the PENNY chips color outputs and

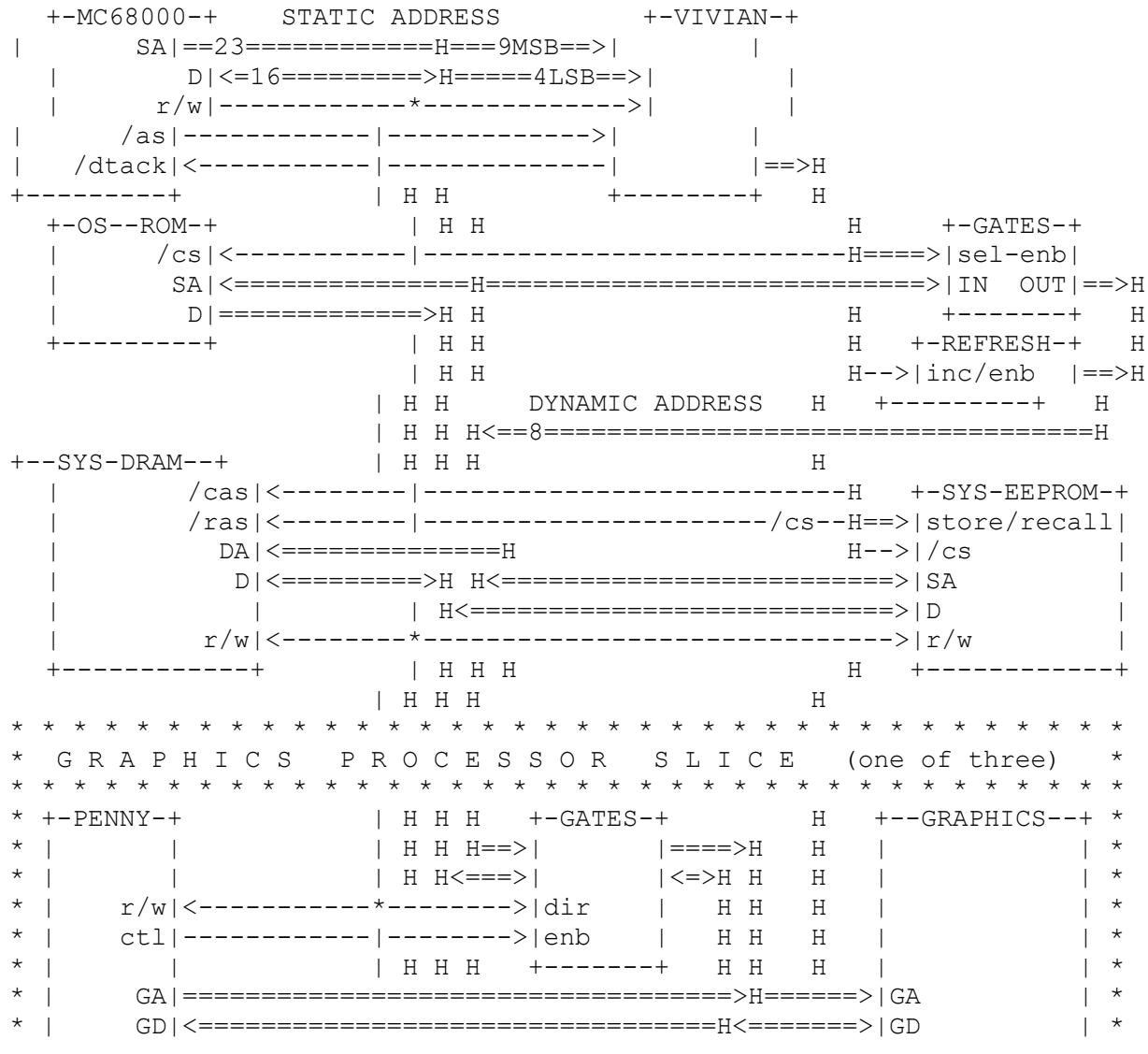
4, HEATHER, a custom chip, which performs the following functions:

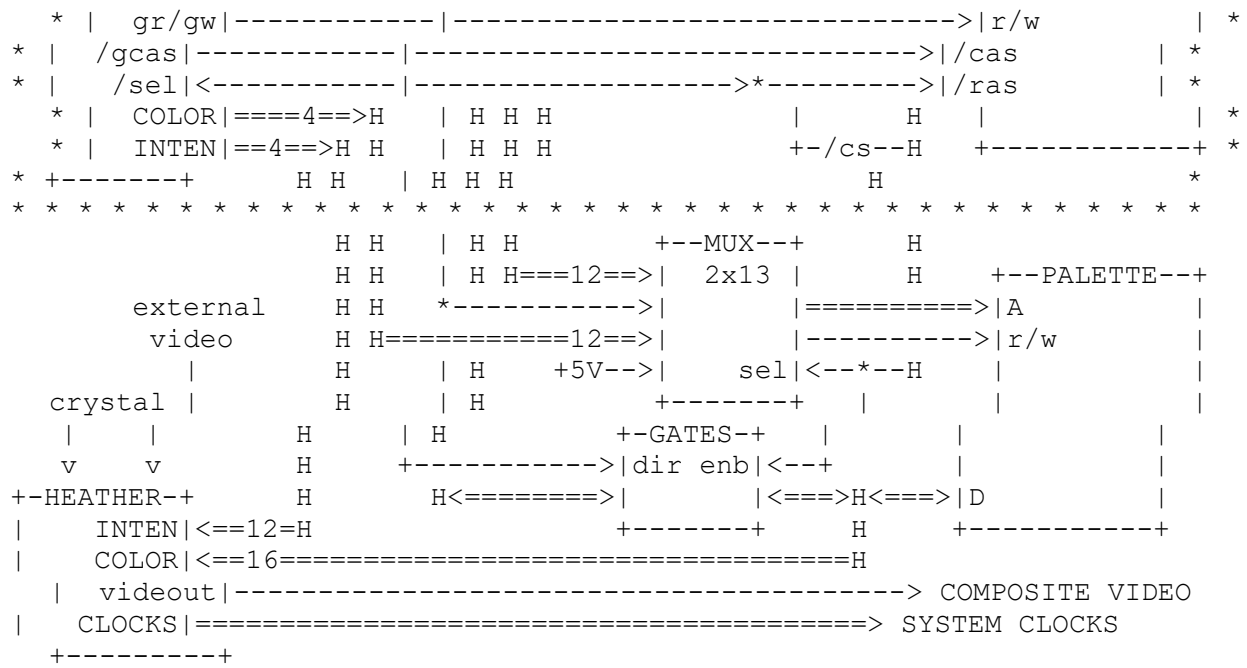
Generates the baseband composite video signal,

Syncronizes to and displays an external video signal (when all sprites are showing transparent) and

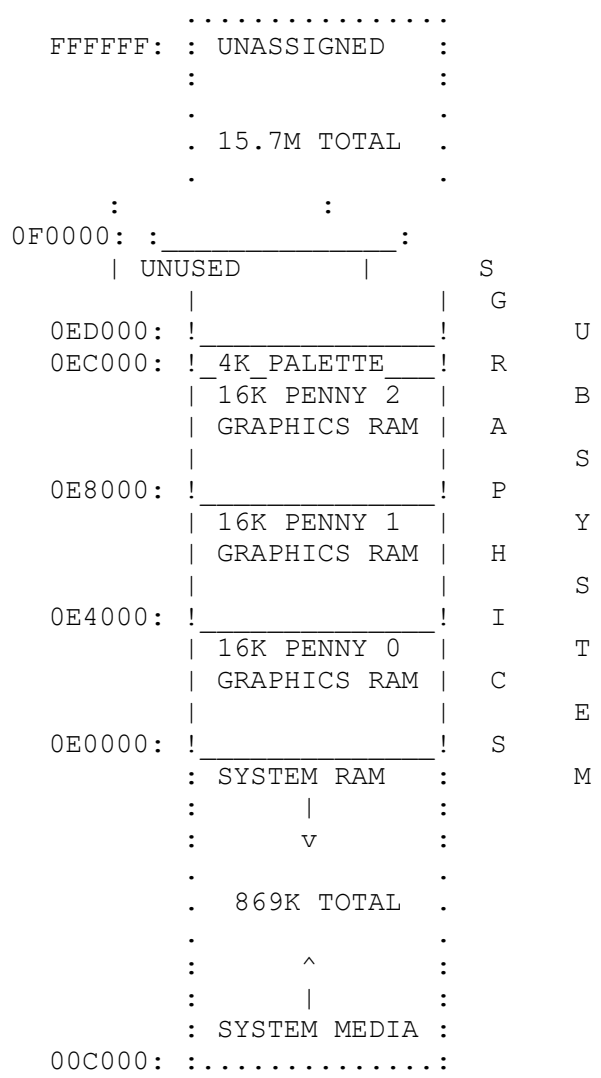
Generates the various system clocks.

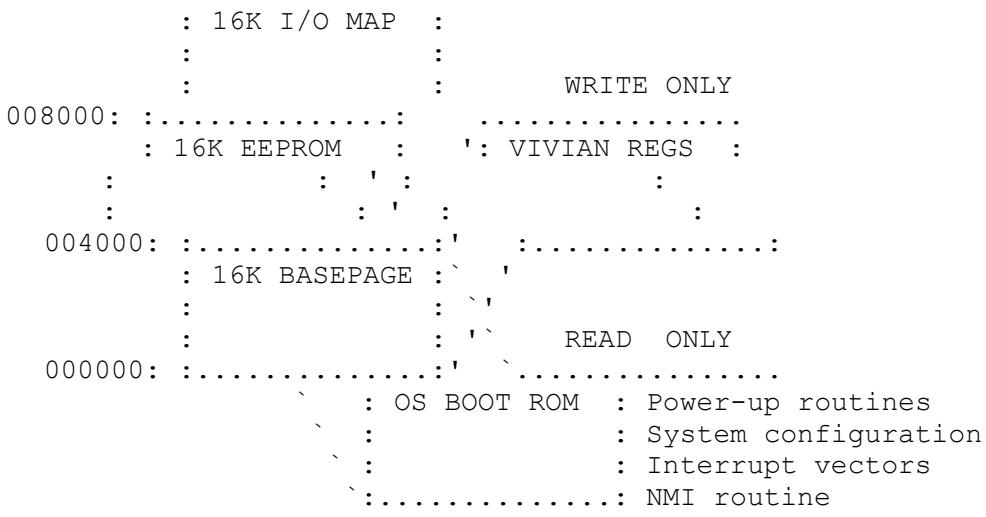
SYSTEM BLOCK DIAGRAM





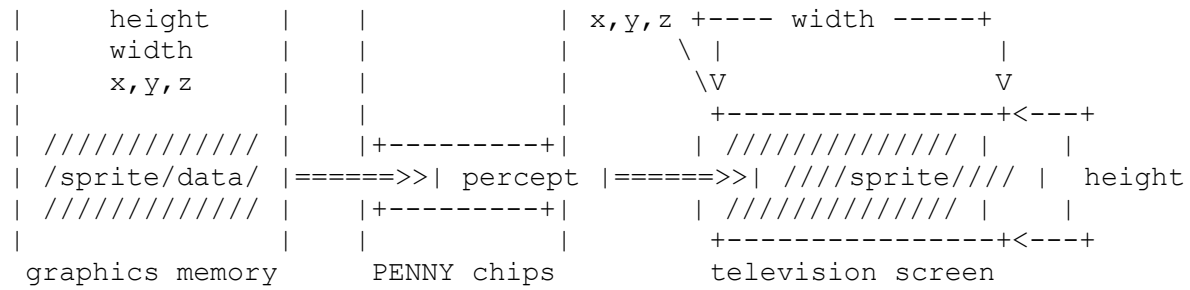
SYSTEM MEMORY MAP





SPRITE DEFINED:

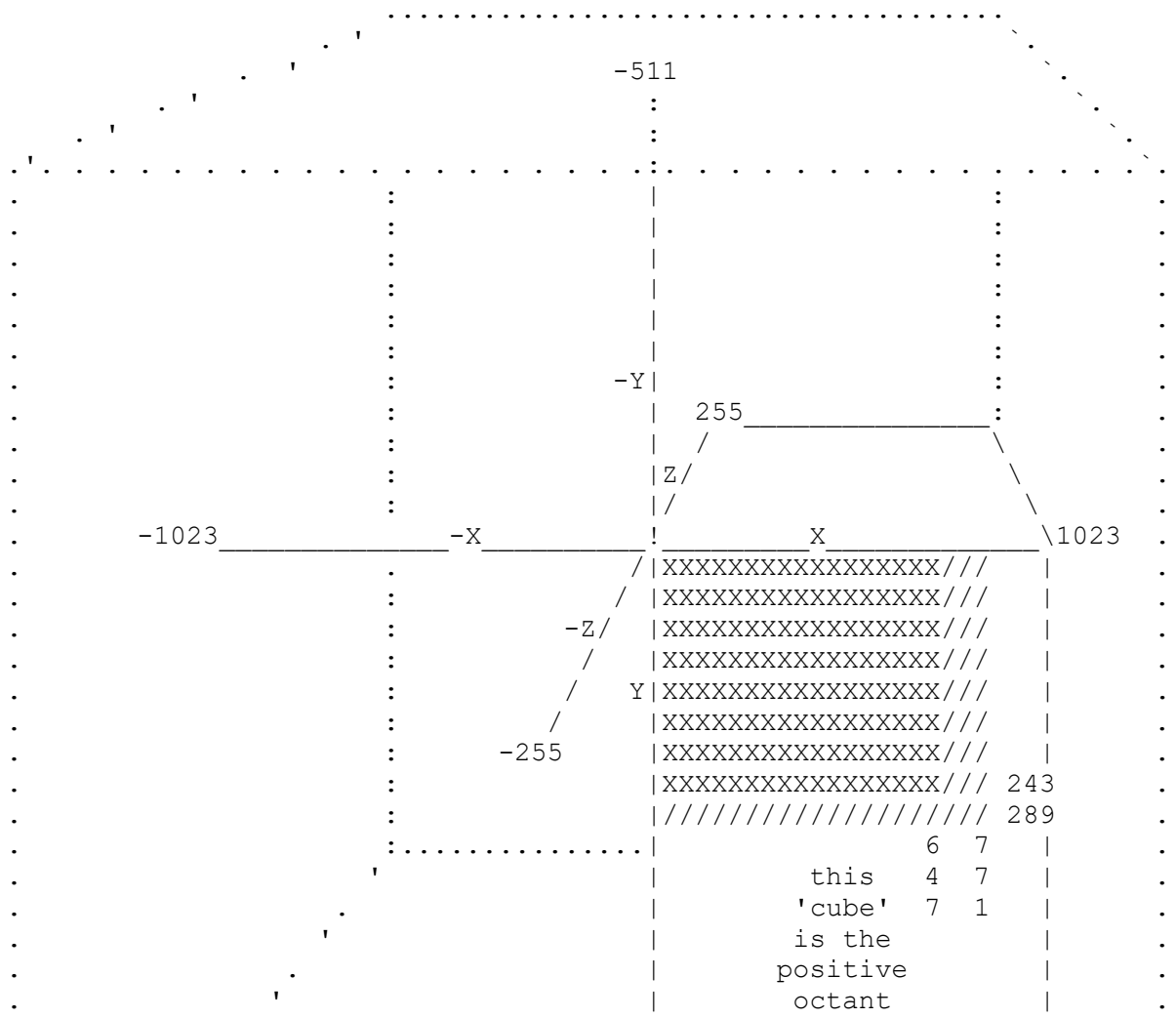
A sprite is an object which is generated, and positioned on the television screen without requiring the cpu to handle its data. It can consist of mixed transparent/non-transparent pixels. There is no restriction on pixel transparency.

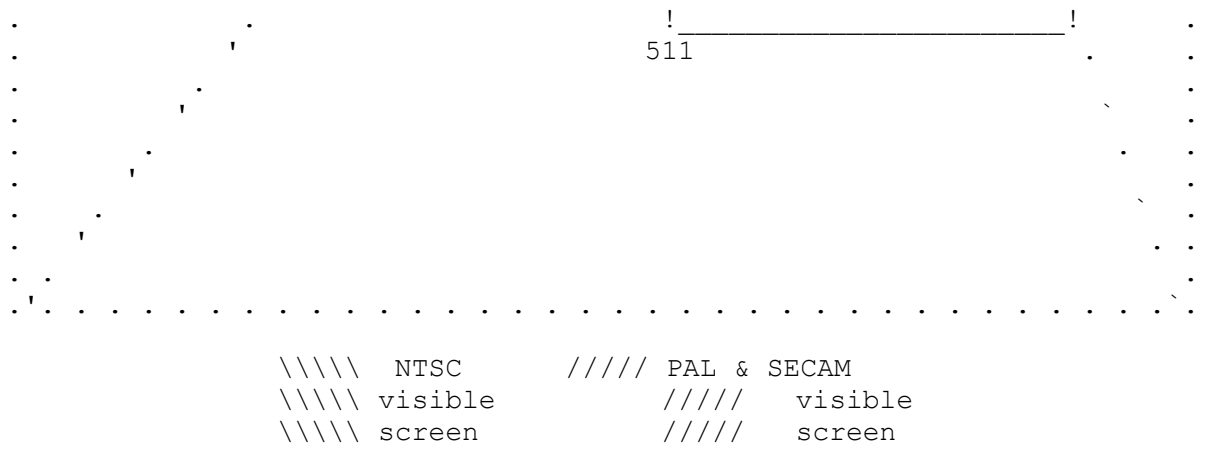


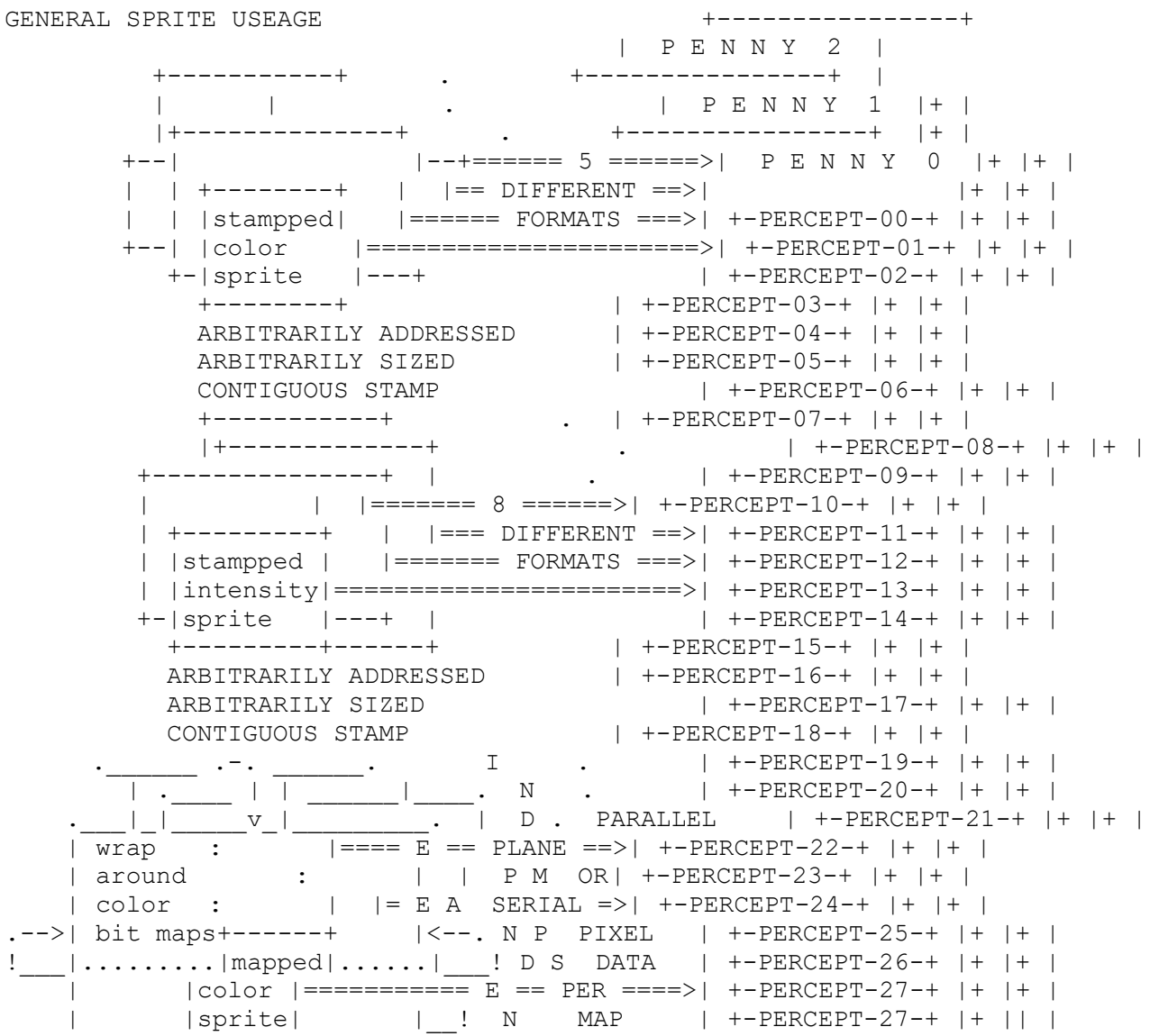
PERCEPT DEFINED:

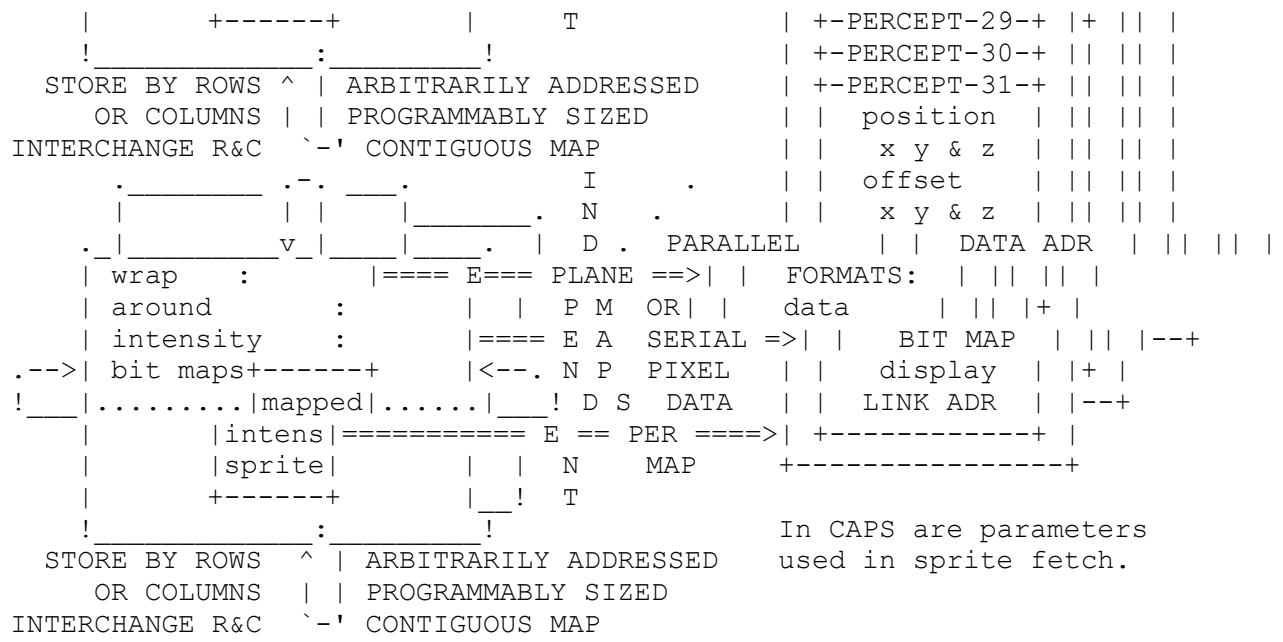
A percept is a hardware sprite generator which can be used to generate one or more sprite incarnations. There are 32 percepts per PENNY numbered from 0 to 31. Each percept has an initial link. The initial link points to the attribute list for the first sprite incarnation. Each sprite thereafter contains a link to the next sprite incarnation for that percept in link list fashion. Sprites cannot be reincarated on the same line and the link list must proceed in the order of television scanning (top to bottom) without overlap between sprites from the same percept.

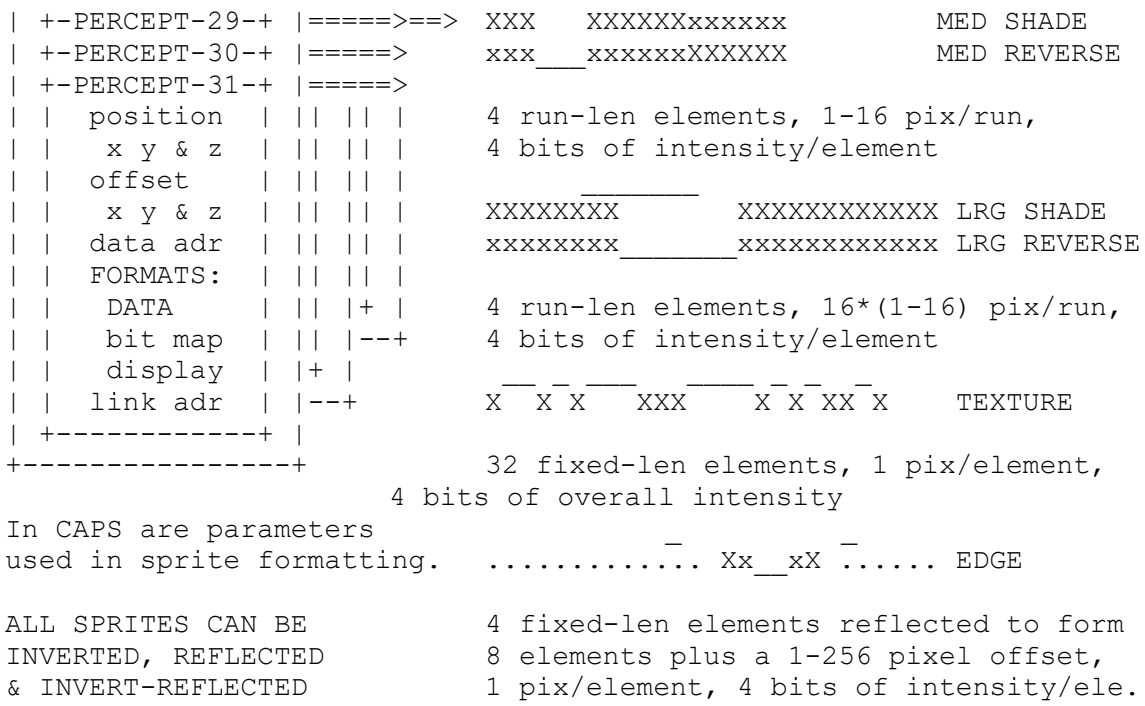
SPRITE POSITIONING SPACE







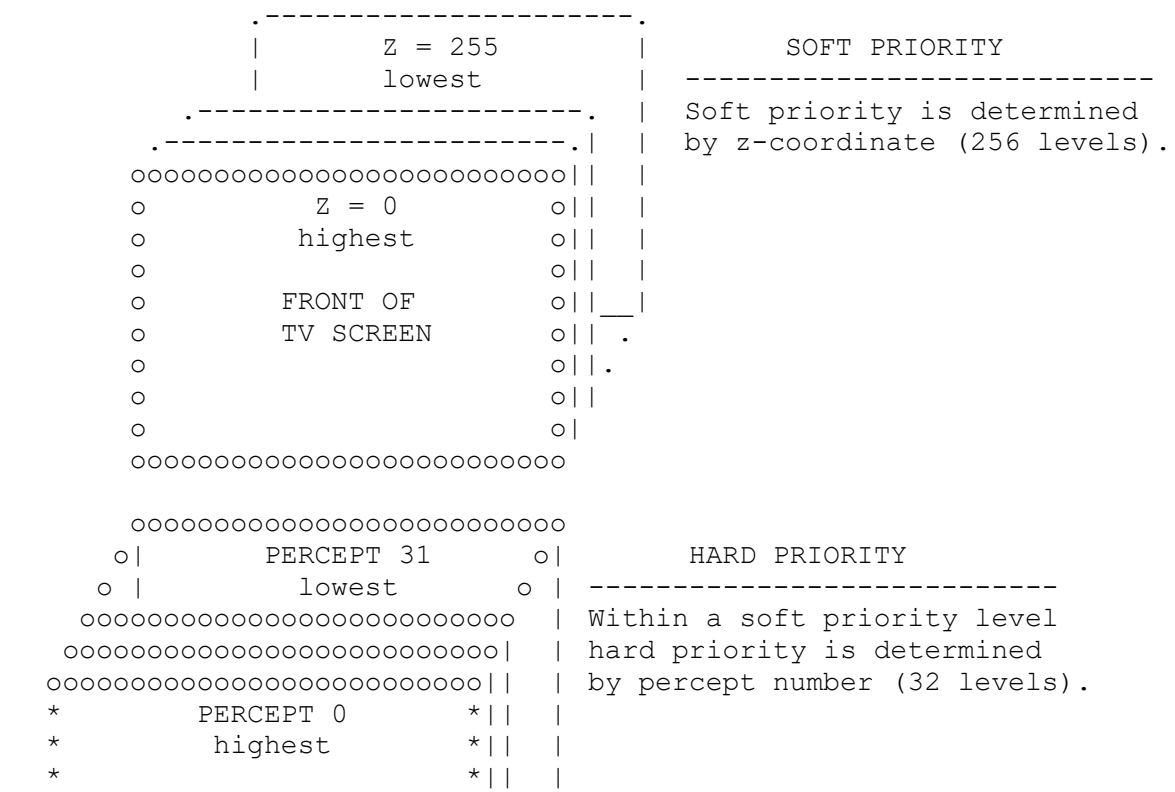


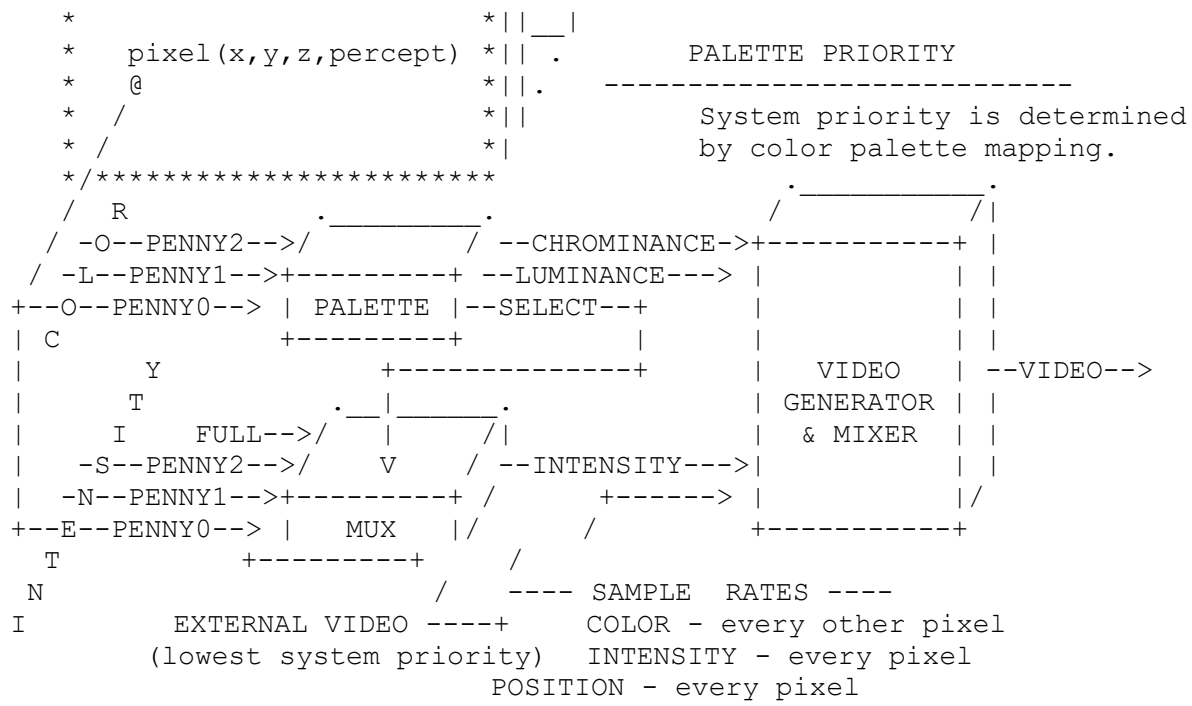


```
DAZZLER
MED CARTOON
LRG CARTOON
AIR BRUSH
COLOR-OR
DETAIL
DETAIL REV
MED SHADE
MED REVERSE
LRG SHADE
LRG REVERSE
TEXTURE
EDGE

+--PERCEPT--+
| POSITION  |
|  X Y & Z  |
| OFFSET    |
|  X Y & Z  |
| DATA ADR  |
| FORMATS:  |
|  DATA     |
|  BIT MAP  |
|  DISPLAY  |
| LINK ADR  |
+-----------+
```

FUNCTIONAL PRIORITIZATION:

There are 8192 levels of functional prioritization in each PENNY. The functional priority is divided between software controllable and hardware fixed priority fields. Soft priority has precidence over hard priority. When portraying a three-dimensional space, soft priority is merely the z-coordinate of a sprite, so the terms 'soft priority', 'z-coordinate' and 'z-level' are interchangable. Final system display priority between PENNYs is determined by color palette mapping.

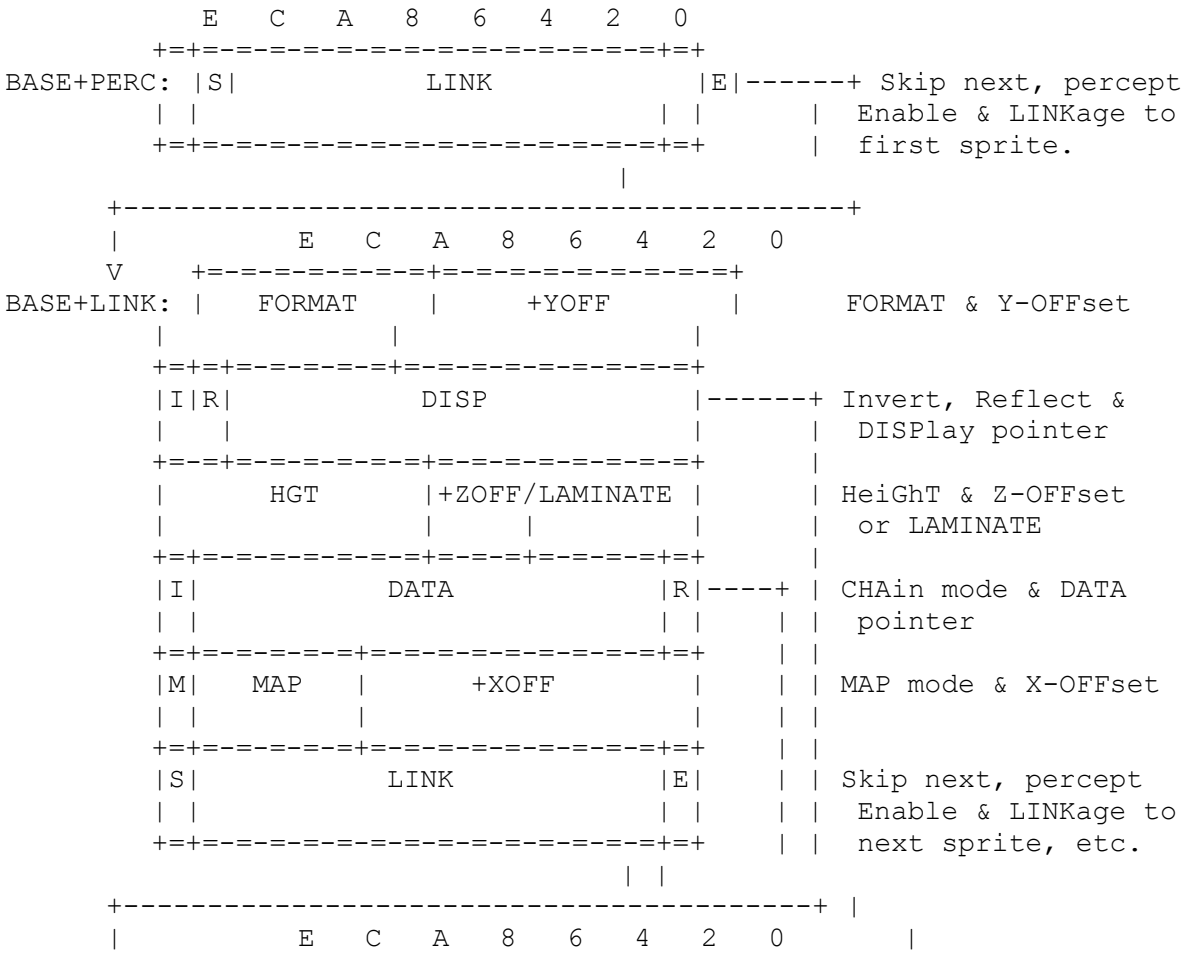


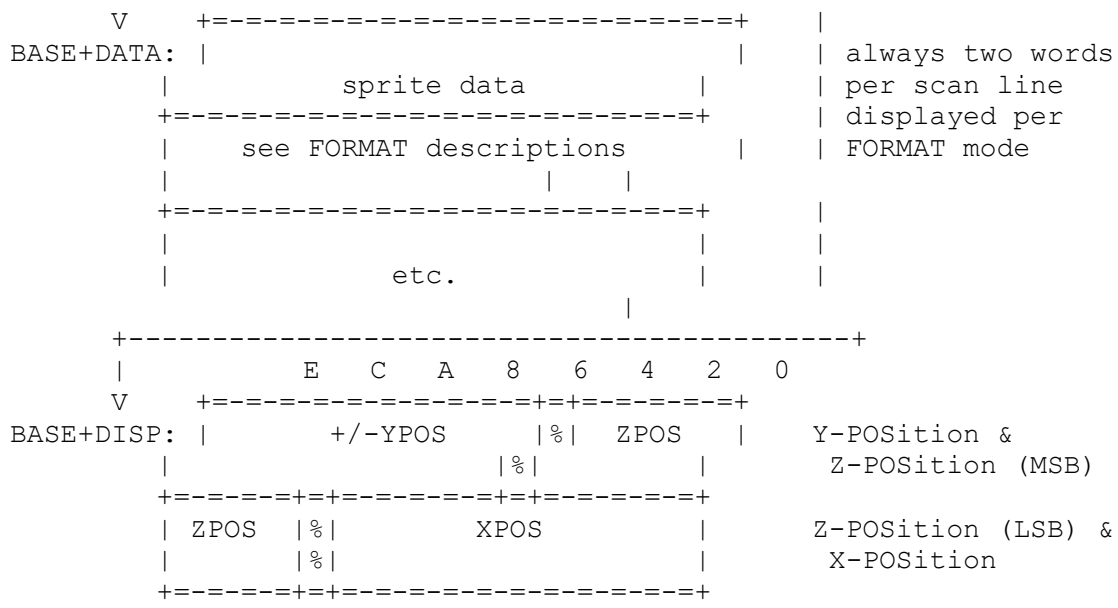


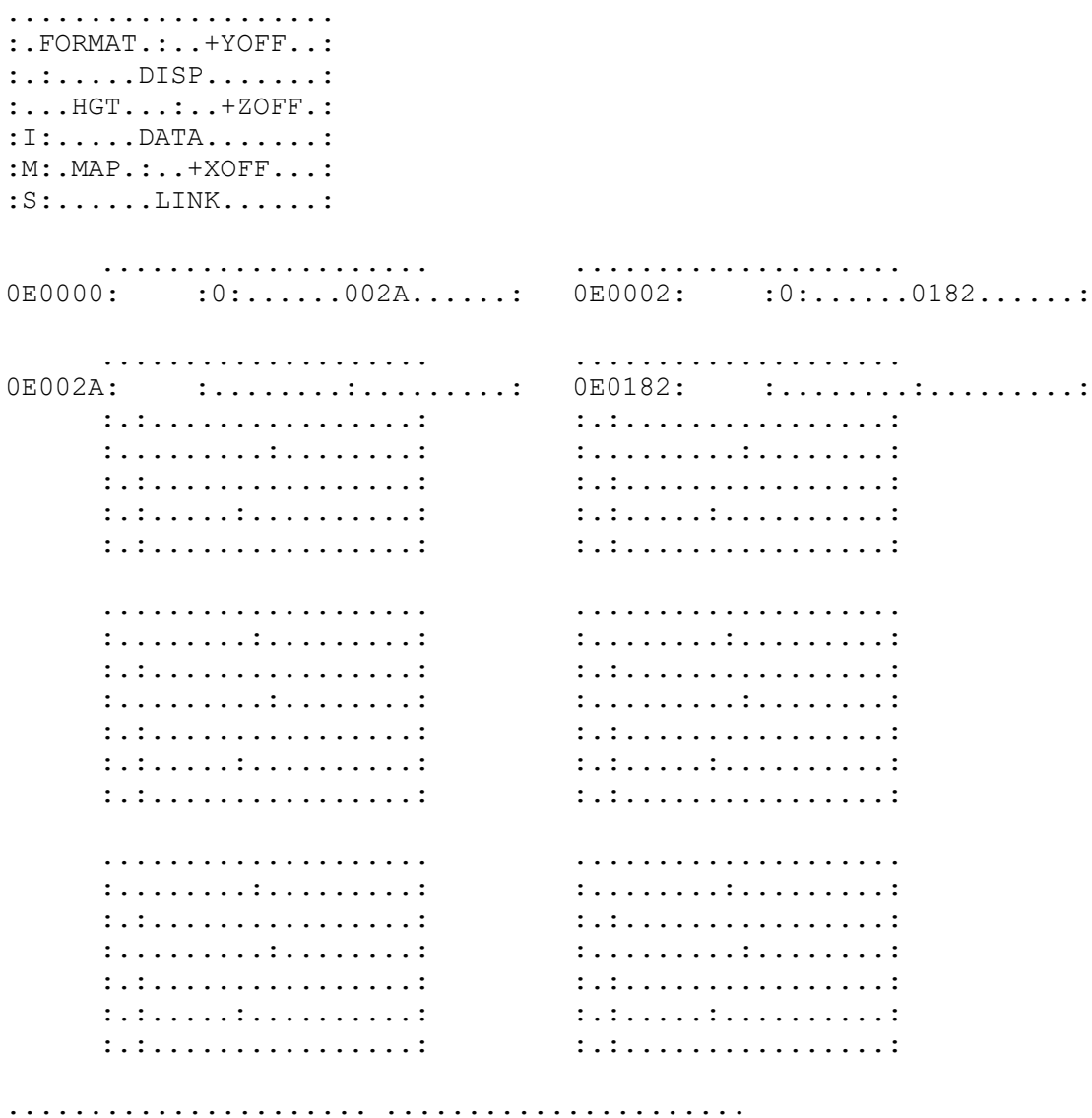
GRAPHICS MEMORY MAP FOR ONE PERCEPT SHOWING FIRST SPRITE

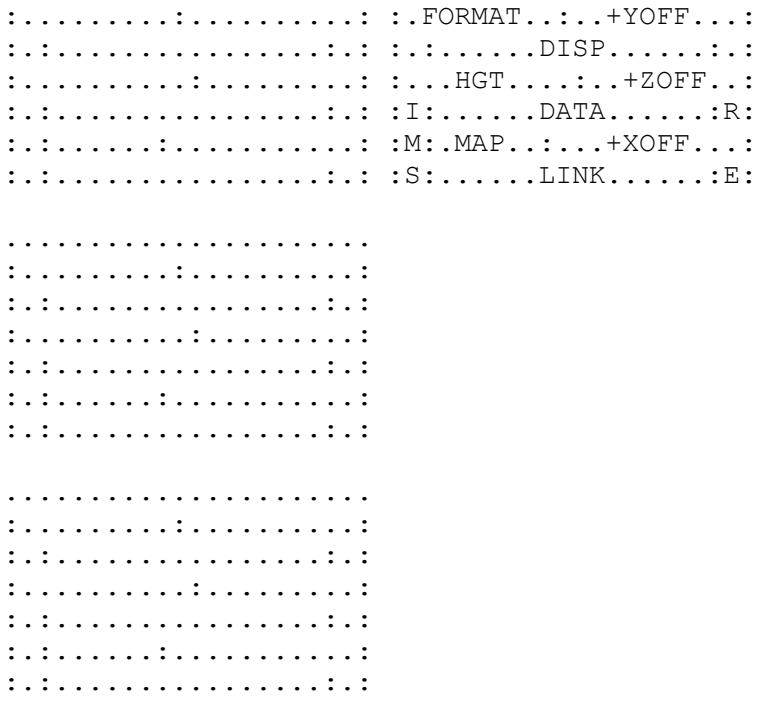
BASE = $0E0000 (PENNY0), $0E40000 (PENNY1) or $0E8000 (PENNY2)

Numbers in blocks are page numbers on which descriptions are found.









S (Skip next switch) DEFINED:

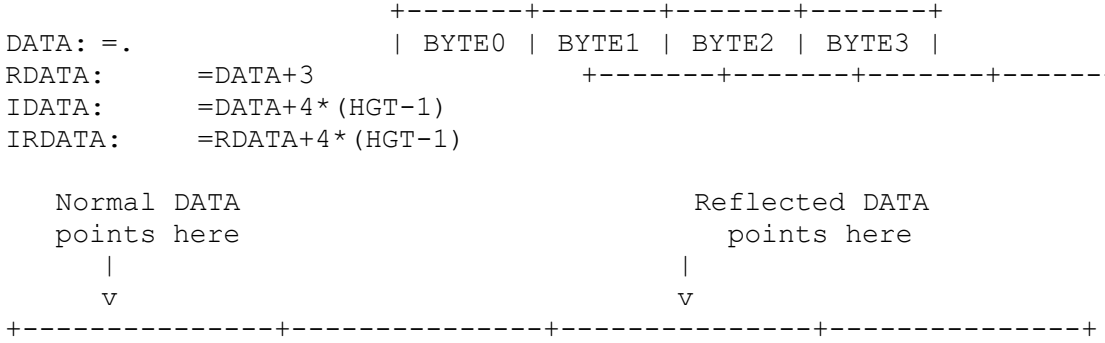
The skip next switch causes the next sprite to be skipped and resumes processing with the succeeding sprite which is linked from the skipped sprite as though the skipped sprite were processed normally.

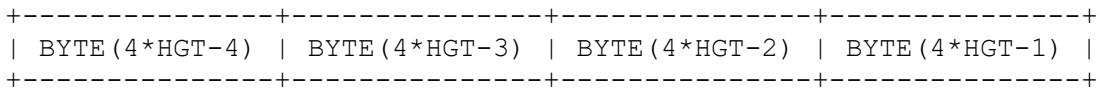
E (percept Enable) DEFINED:

The percept enable allows the next link to be processed. If not enabled, the percept is terminated and no further sprites will be generated by it until the next screen.

I & R (Invert & Reflect switches) DEFINED:

Invert and reflect work on sprite data and do just as the names imply. If the invert switch is on, DATA must point to the last word of sprite data as the pointer is decremented with each line instead of incremented. If the reflect switch is on, the CHAinin/out function is reversed (see CHAin mode).



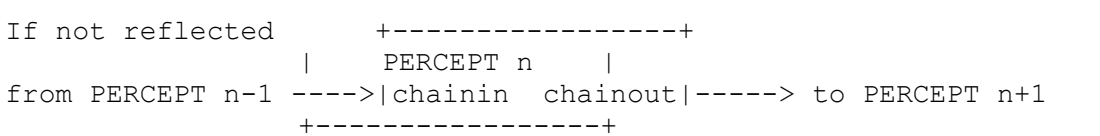


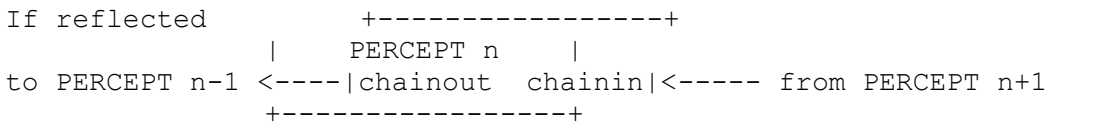
HGT (HeiGhT) DEFINED:

Height defines the vertical height of the sprite in scan lines. Remember that a line of sprite data is always two words.

CHA (CHAin mode) DEFINED:

Chaining allows two percepts to interact. The chain input (chainin) to a percept is an enable for the generation of the sprite for that percept. The chain output (chainout) from a percept is the enable (chainin) to the next higher numbered percept (or lower numbered if the reflect switch is on ... see Invert & Reflect switches). An unchained percept is always enabled.





CHA	Mode	line of sprite starts on	DATA out is enabled	Chainout equals
00	unchained	X = XPOS+XOFF	always	DATA > 0
01	stencil	X = XPOS+XOFF	by chainin	chainin

10	startile	X = XPOS+XOFF	always	line done this sprite
11	tile	chainin	always	line done this sprite

Stenciling and tiling is discussed in detail in the APPENDIX.

MAP (MAP mode) DEFINED:

There are 32 map functions for each view as follows:

```
        --  M a p   W i d t h  --
     n u m b e r   o f   s p r i t e s *

                  4     8    1 6   3 2
|              +-----+-----+-----+-----+
|  n       4   | 1 F | 1 E | 1 D | 1 C |
   u           +-----+-----+-----+-----+
M  m       8   | 1 B | 1 A | 1 9 | 1 8 |
a  b           +-----+-----+-----+-----+
p  e     1 6   | 1 7 | 1 6 | 1 5 | 1 4 |
   r           +-----+-----+-----+-----+
H        3 2   | 1 3 | 1 2 | 1 1 | 1 0 |
e  o           +-----+-----+-----+-----+
i  f     6 4   | 0 F | 0 E | 0 D | 0 C |
g              +-----+-----+-----+-----+
h  l   1 2 8   | 0 B | 0 A | 0 9 | 0 8 |
t  i           +-----+-----+-----+-----+
   n   2 5 6   | 0 7 | 0 6 | 0 5 | 0 4 |
|  e           +-----+-----+-----+-----+
|  s   5 1 2   | 0 3 | 0 2 | 0 1 |
|              +-----+-----+-----+
                       +-----+
                       | 0 0 | MAP mode off
                       +-----+ Sprite is graphics stamp
```

* The number of pixels that each sprite produces is determined by its FORMAT.

Map height & width refer to the arrangement of data in memory, only ... not to the size of screen graphics. They are used to control memory addresses and for vertical & horizontal wrap-around only. Reflect reverses horizontal wrap-around only.

XOFF, YOFF & ZOFF (X-OFFset, Y-OFFset & Z-OFFset) DEFINED:

These unsigned binaries specify the offsets to the upper left-hand corner of the sprite relative to the sprite position as defined by XPOS, YPOS & ZPOS.

XPOS, YPOS & ZPOS (X-POSition, Y-POSition & Z-POSition) DEFINED:

These signed binaries are the x,y & z values of course or group positioning to which XOFF, YOFF & ZOFF are added to arrive at the sprite's true x, y & z positions on the screen. Both these parameters and their associated offsets are full resolution and full screen so that they can be individually used to perform complete positioning. The presence of the offsetting ability, though, makes group motion and/or screen scrolling much easier.

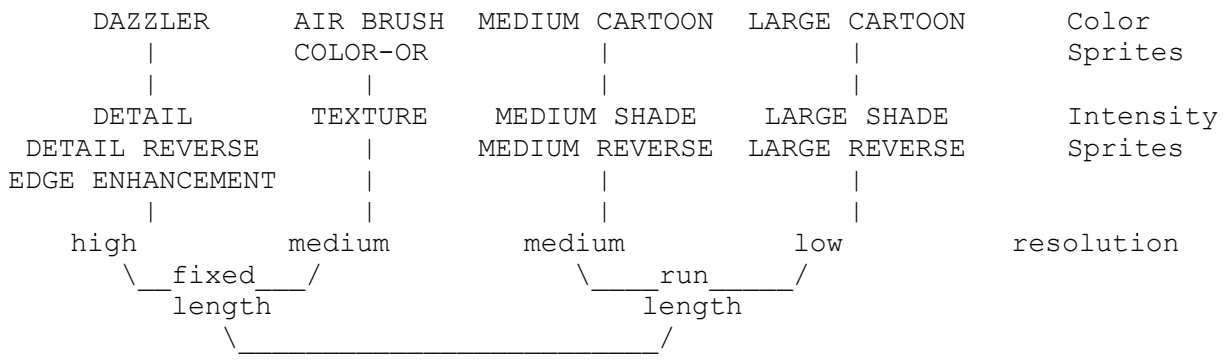
FORMAT (FORMAT mode) DEFINED:

FORMAT SELECTION GUIDE

```
   FORMAT ('bbbb' is overall brightness,
   |     'cccc' is overall color and
   |     'iiii' is overall intensity)
   |  AUTOMATIC EDGE SMOOTHING
   |  | NUMBER OF BITS OF GRAPHICS DATA PER ELEMENT
   |  | | COLOR DETERMINED BY (Format, Graphics)
   |  | | | INTENSITY DETERMINED BY (Format, Graphics)
   |  | | | | GRAPHICS ELEMENT TYPE (Fixed-length, Run-length)
   |  | | | | |      ELEMENT SIZE IN PIXELS
   |  | | | | |      |      NUMBER OF ELEMENTS PER SPRITE
   |  | | | | |      |      |   MAX SPRITE SIZE IN PIXELS
   |  | | | | |      |      |   | + indicates offset(s), also
   V    V V V V V      V      V   V            NAME (page)
======= = = = = =  ========  ==  ===  ================================

=================== Z-PRIORITIZED COLOR SPRITES ====================

001bbbb - 4 G F F      2      8   16  DAZZLER (16)
010bbbb X 8 G F R   2 to 32   4  128  MEDIUM CARTOON (17)
011bbbb X 8 G F R  32 to 512  4 2048  LARGE CARTOON (18)
110cccc - 1 F - F      2     32   64  AIR BRUSH (25)

================== NON-PRIORITIZED COLOR SPRITE ====================

111cccc - 1 F - F      2     32   64  COLOR-OR (  )

======================== INTENSITY SPRITES =========================

   Self laminate bit
   | 0 => Laminate on color sprite specified by LAMINATE parameter
   | 1 => Self-laminate (always on)
   v
```

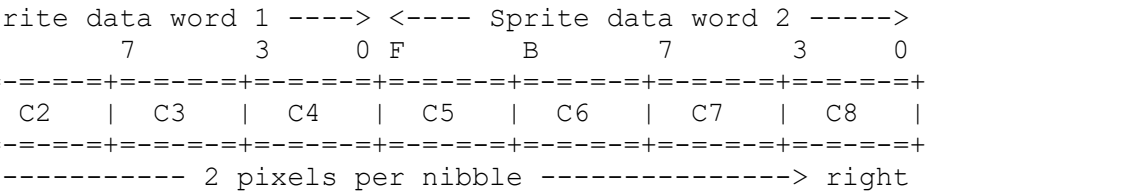
```
  000S000 - 4 - G F      1      8    8+ EDGE ENHANCEMENT (27)
 " S010 - 4 - G F      1      8    8  DETAIL (19)
 " S011 - 4 - G F      1      8    8  DETAIL REVERSE (20)
 " S100 - 8 - G R   1 to 16   4   64  MEDIUM SHADE (21)
   " S101 - 8 - G R   1 to 16   4   64  MEDIUM REVERSE (22)
 " S110 - 8 - G R  16 to 256  4 1024  LARGE SHADE (23)
   " S111 - 8 - G R  16 to 256  4 1024  LARGE REVERSE (24)

  Self laminate bit
  | 0 => Laminate on color sprite specified by LAMINATE parameter
  | 1 => Self-laminate (always on)
  v
10Siiii - 1 - F F      1     32   32  TEXTURE (26)
```

SPRITE FAMILY TREE



DAZZLER SPRITE; FORMAT = 001bbbb

Optimized for high resolution poly-chromatic detail, this sprite has 16 pixels of graphics. Pixel color is determined by the graphics data and overall brightness is determined by the FORMAT.



========== PROGRAMMING ==========

======== ELEMENT COLOR ======== ====== OVERALL BRIGHTNESS ======

Cn	color	comments	FORMAT	brightness	comments
0000	0000	transparent	0110000	0000	full LUM
1110	1110	color-14	1110	1110	2/16ths of LUM

1111 -------> 15-switch

Color is a function of /Z

ZPOS+ZOFF <--- +0000%%%% 1111 closest to screen, color-15

+1111%%%% 0000 furthest from screen, t'parent

1111 ------> 15-switch

Brightness is a function of Z

ZPOS+ZOFF <--- +0000%%%% 0000 closest to screen, full LUM

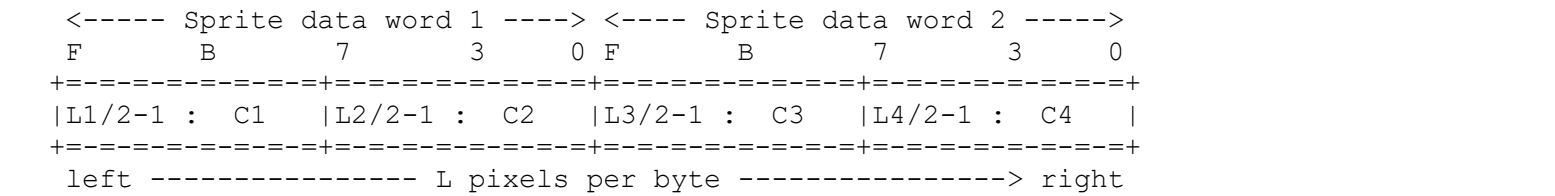
+1111%%%% 1111 furthest from screen, 1/16ths LUM

SCREEN GRAPHICS (one typical element out of eight total elements)

```
    +-+-+        Does not automatically generate edge smoothing.
    |   |
    +-+-+        Can be laminated by intensity sprites.
 -->|   |<-- 2 pixels
                 Note the similarities & differences between
                 this sprite & its kin, the DETAIL & DETAIL
                 REVERSE sprites.
```

NOTES:

MEDIUM CARTOON SPRITE; FORMAT = 010bbbb



========== PROGRAMMING ==========

======== ELEMENT COLOR ======== ====== OVERALL BRIGHTNESS ======

Cn	color	comments	FORMAT brightness	comments
0000	0000	transparent	0110000	0000	full LUM
1110	1110	color-14	1110	1110	2/16ths of LUM

1111 -------> 15-switch 1111 ------> 15-switch

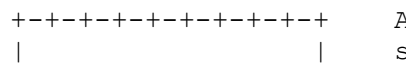
Color is a function of /Z Brightness is a function of Z

ZPOS+ZOFF <--- ZPOS+ZOFF <---

+0000%%%% 1111 closest to screen, color-15 +0000%%%% 0000 closest to screen, full LUM

+1111%%%% 0000 furthest from screen, t'parent +1111%%%% 1111 furthest from screen, 1/16ths LUM

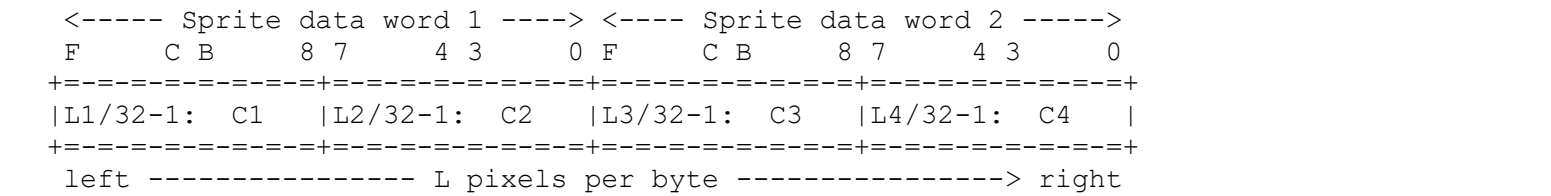
SCREEN GRAPHICS (one typical element out of four total elements)



Automatically generates left & right edge smoothing for each element.

```
|<--- L pixels -->|    Can be laminated by intensity sprites.

  For L =   StoreNote the similarities and differences between
  -------   -----this sprite and its kin, the MEDIUM SHADE &
     2        0       MEDIUM REVERSE sprites.
     4        1
     6           2
        etc.

NOTES:
```

LARGE CARTOON SPRITE; FORMAT = 011bbbb



========== PROGRAMMING ==========

======== ELEMENT COLOR ======== ====== OVERALL BRIGHTNESS ======

Cn	color	comments	FORMAT	brightness	comments
0000	0000	transparent	0110000	0000	full LUM
1110	1110	color-14	1110	1110	2/16ths of LUM

1111 -------> 15-switch 1111 ------> 15-switch

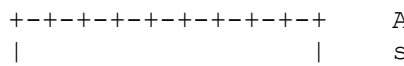
Color is a function of /Z Brightness is a function of Z

ZPOS+ZOFF <--- /Z ZPOS+ZOFF <--- Z

+0000%%%% 1111 closest to screen, color-15 +0000%%%% 0000 closest to screen, full LUM

+1111%%%% 0000 furthest from screen, t'parent +1111%%%% 1111 furthest from screen, 1/16ths LUM

SCREEN GRAPHICS (one typical element out of four total elements)



Automatically generates left & right edge smoothing for each element.

```
|<--- L pixels -->|    Can be laminated by intensity sprites.

  For L =   StoreNote the similarities and differences between
  -------   -----this sprite and its kin, the LARGE SHADE &
    32        0        LARGE REVERSE sprites.
    64        1
    96            2
      etc.

NOTES:
```

```
DETAIL SPRITE; FORMAT = 000S010
                    ^
                    |
Self laminate bit ----------+
0 => Laminate on color sprite specified by LAMINATE parameter
1 => Self-laminate (always on)
```

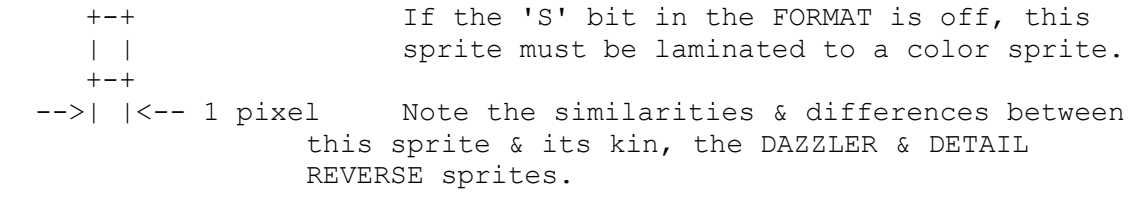
Optimized for high resolution multi-intensity detail, this sprite has 8 pixels. Pixel intensity is determined by the graphics data. It is very useful for creating text and for edge enhancement of color sprites.

```
 <----- Sprite data word 1 ----> <---- Sprite data word 2 ----->
 F       B       7       3     0 F       B       7       3     0
+=-=-=-+=-=-=-=+=-=-=-+=-=-=-=+=-=-=-+=-=-=-=+=-=-=-=+=-=-=-+
|  I1   |  I2   |  I3   |  I4   |  I5   |  I6   |  I7   |  I8   |
+=-=-=-+=-=-=-=+=-=-=-+=-=-=-=+=-=-=-+=-=-=-=+=-=-=-=+=-=-=-+
 left --------------- 1 pixel per nibble ----------------> right
```

PROGRAMMING ELEMENT INTENSITY

```
 In      intensity   comments
====            ====    ============================
0000            0000  full LUM as stored in palette (no contrast)
 --        --
1110            1110  2/16ths of LUM stored in palette (max contrast)

1111 ---------------> 15-switch  ----------------------+
                                                       |
ZPOS+ZOFF <-----------  intensity is a function of /Z <-+
=========
+0000%%%%  1111  closest to screen, 1/16th LUM (max contrast)
  --        --
+1111%%%%  0000  furthest from screen, full LUM (no contrast)
```

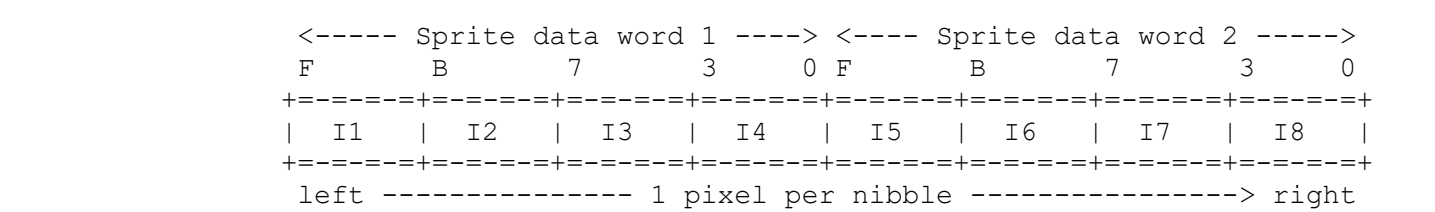
SCREEN GRAPHICS (one typical element out of eight total elements)



NOTES:

```
DETAIL REVERSE SPRITE; FORMAT = 000S011
                           ^
                           |
Self laminate bit ------------------+
0 => Laminate on color sprite specified by LAMINATE parameter
1 => Self-laminate (always on)
```

This sprite is functionally identical to the DETAIL sprite except that the values of INTENSITY stored are internally complimented before be written on the screen resulting in an inverse video (but not complimentary color) display. It is very useful for creation of a cursor. Any font character can be transformed into that same character displayed over the cursor with a single bit set in its FORMAT field (ie, change 0000010 to 0000011).

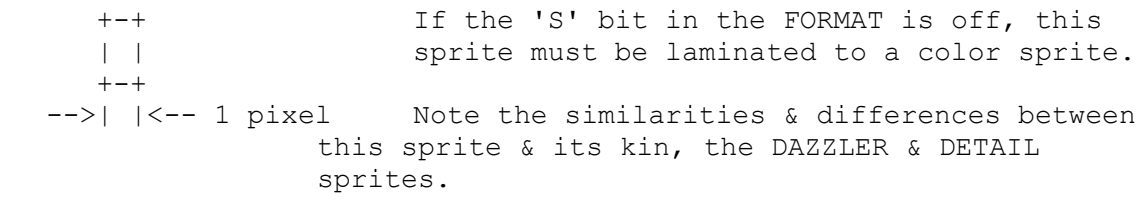



```
================= PROGRAMMING ELEMENT INTENSITY =================

   In      intensity   comments
  ====           ====    ============================
  0000           1111  1/16ths of LUM stored in palette (max contrast)
   --         --
  1110           0001  14/16ths of LUM stored in palette (min contrast)

  1111 ---------------> 15-switch  ----------------------+
                                                        |
ZPOS+ZOFF <-----------  intensity is a function of /Z <-+
=========
+0000%%%%   0000  closest to screen, full LUM (no contrast)
   --         --
```

```
+1111%%%%  1111  furthest from screen, 1/16th LUM (max contrast)
```

IMPORTANT: Note that though the intensities generated are reversed from the DETAIL sprite, the 15-switch (In = 1111) is the same.

SCREEN GRAPHICS (one typical element out of eight total elements)



NOTES:

```
MEDIUM SHADE SPRITE; FORMAT = 000S100
                         ^
                         |
Self laminate bit ---------------+
0 => Laminate on color sprite specified by LAMINATE parameter
1 => Self-laminate (always on)

   <----- Sprite data word 1 ----> <---- Sprite data word 2 ----->
   F       B       7       3     0 F       B       7       3     0
  +=======+=======+=======+=======+=======+=======+=======+=======+
  |  L1-1 :   I1  |  L2-1 :   I2  |  L3-1 :   I3  |  L4-1 :   I4  |
  +=======+=======+=======+=======+=======+=======+=======+=======+
   left ---------------- L pixels per byte ----------------> right

================= PROGRAMMING ELEMENT INTENSITY =================

   In      intensity   comments
  ====           ====    ============================
  0000           0000  full LUM as stored in palette (no contrast)
   --        --
  1110           1110  2/16ths of LUM stored in palette (max contrast)

  1111 ---------------> 15-switch  ---------------------+
                                                       |
ZPOS+ZOFF <-----------  intensity is a function of /Z <-+
=========
+0000%%%%  1111  closest to screen, 1/16th LUM (max contrast)
   --       --
+1111%%%%  0000  furthest from screen, full LUM (no contrast)


SCREEN GRAPHICS (one typical element out of four total elements)

+-+-+-+-+-+-+-+-+
|                 |
+-+-+-+-+-+-+-+-+    If the 'S' bit in the FORMAT is off, this
```

```
|<--- L pixels -->|    sprite must be laminated to a color sprite.

  For L =   StoreNote the similarities and differences between
  -------   -----this sprite and its kin, the MEDIUM CARTOON &
     1        0       MEDIUM REVERSE sprites.
     2        1
     3           2
        etc.

NOTES:
```

MEDIUM REVERSE SPRITE; FORMAT = 000S101

```
                          ^
                          |
Self laminate bit -----------------+
```

0 => Laminate on color sprite specified by LAMINATE parameter
1 => Self-laminate (always on)

```
   <----- Sprite data word 1 ----> <---- Sprite data word 2 ----->
   F       B       7       3     0 F       B       7       3     0
  +=======+=======+=======+=======+=======+=======+=======+=======+
  |  L1-1 :   I1  |  L2-1 :   I2  |  L3-1 :   I3  |  L4-1 :   I4  |
  +=======+=======+=======+=======+=======+=======+=======+=======+
   left ---------------- L pixels per byte ----------------> right
```

================= PROGRAMMING ELEMENT INTENSITY =================

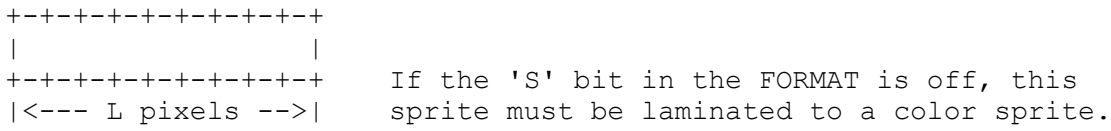
In	intensity	comments
0000	1111	1/16ths of LUM stored in palette (max contrast)
1110	0001	14/16ths of LUM stored in palette (min contrast)

```
  1111 ---------------> 15-switch  ---------------------+
                                                       |
ZPOS+ZOFF <-----------  intensity is a function of /Z <-+
=========
```

+0000%%%%	0000	closest to screen, full LUM (no contrast)
+1111%%%%	1111	furthest from screen, 1/16th LUM (max contrast)

IMPORTANT: Note that though the intensities generated are reversed from the MEDIUM SHADE sprite, the 15-switch (In = 1111) is the same.

SCREEN GRAPHICS (one typical element out of eight total elements)



For L =	Store
1	0
2	1
3	2
etc.	

Note the similarities and differences between this sprite and its kin, the MEDIUM CARTOON & MEDIUM SHADE sprites.

NOTES:

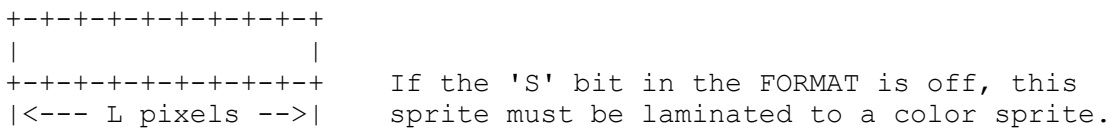
```
LARGE SHADE SPRITE; FORMAT = 000S110
                         ^
                         |
Self laminate bit ---------------+
0 => Laminate on color sprite specified by LAMINATE parameter
1 => Self-laminate (always on)

   <----- Sprite data word 1 ----> <---- Sprite data word 2 ----->
   F       B       7       3     0 F       B       7       3     0
  +=======+=======+=======+=======+=======+=======+=======+=======+
  |L1/16-1:  I1   |L2/16-1:  I2   |L3/16-1:  I3   |L4/16-1:  I4   |
  +=======+=======+=======+=======+=======+=======+=======+=======+
   left ---------------- L pixels per byte ----------------> right

================= PROGRAMMING ELEMENT INTENSITY =================

   In      intensity   comments
  ====           ====    ============================
  0000           0000  full LUM as stored in palette (no contrast)
   --        --
  1110           1110  2/16ths of LUM stored in palette (max contrast)

  1111 ---------------> 15-switch  ---------------------+
                                                       |
ZPOS+ZOFF <-----------  intensity is a function of /Z <-+
=========
+0000%%%%  1111  closest to screen, 1/16th LUM (max contrast)
   --       --
+1111%%%%  0000  furthest from screen, full LUM (no contrast)


SCREEN GRAPHICS (one typical element out of four total elements)

+-+-+-+-+-+-+-+-+
|                 |
+-+-+-+-+-+-+-+-+    If the 'S' bit in the FORMAT is off, this
```

```
|<--- L pixels -->|    sprite must be laminated to a color sprite.

  For L =   StoreNote the similarities and differences between
  -------   -----this sprite and its kin, the LARGE CARTOON &
    16        0        LARGE REVERSE sprites.
    32        1
    48            2
        etc.

NOTES:
```

```
LARGE REVERSE SPRITE; FORMAT = 000S111
                           ^
                           |
Self laminate bit -----------------+
0 => Laminate on color sprite specified by LAMINATE parameter
1 => Self-laminate (always on)

   <----- Sprite data word 1 ----> <---- Sprite data word 2 ----->
   F       B       7       3     0 F       B       7       3     0
  +=======+=======+=======+=======+=======+=======+=======+=======+
  |L1/16-1:  I1   |L2/16-1:  I2   |L3/16-1:  I3   |L4/16-1:  I4   |
  +=======+=======+=======+=======+=======+=======+=======+=======+
   left ---------------- L pixels per byte ----------------> right

================= PROGRAMMING ELEMENT INTENSITY =================

   In      intensity   comments
  ====           ====    ============================
  0000           1111  1/16ths of LUM stored in palette (max contrast)
   --         --
  1110           0001  14/16ths of LUM stored in palette (min contrast)

  1111 ---------------> 15-switch  ---------------------+
                                                       |
ZPOS+ZOFF <-----------  intensity is a function of /Z <-+
=========
+0000%%%%   0000  closest to screen, full LUM (no contrast)
   --         --
+1111%%%%   1111  furthest from screen, 1/16th LUM (max contrast)
```

IMPORTANT: Note that though the intensities generated are reversed from the LARGE SHADE sprite, the 15-switch (In = 1111) is the same.

SCREEN GRAPHICS (one typical element out of eight total elements)



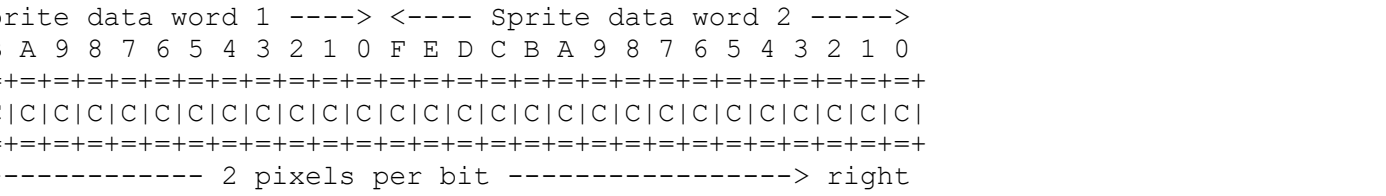
For L =	Store
16	0
32	1
48	2
etc.	

Note the similarities and differences between this sprite and its kin, the LARGE CARTOON & LARGE SHADE sprites.

NOTES:

AIR BRUSH SPRITE; FORMAT = 110cccc

Optimized for high resolution mono-chromatic detail, this sprite has 64 pixels of color graphics. Pixel color is determined by the FORMAT.

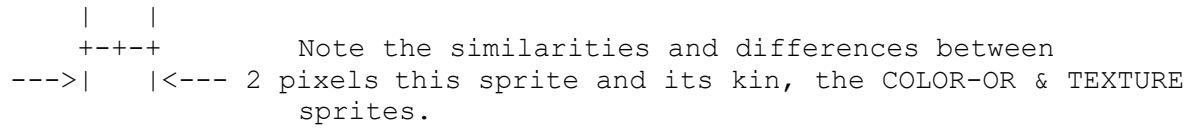


PROGRAMMING ELEMENT COLOR

C	FORMAT	color	comments
0	110%%%%	0000	transparent
1	" 0000	0000	dissolve (see Appendix for its use)
1	" --	--	
1	" 1110	1110	color-14
1	" 1111 ------>		15-switch
	ZPOS+ZOFF <----		Color is a function of /Z
1	0000%%%%	1111	closest to screen, color-15
1	--	--	
1	1111%%%%	0000	furthest from screen, dissolve (see Appendix for its use)

SCREEN GRAPHICS (one element out of 32 total elements)

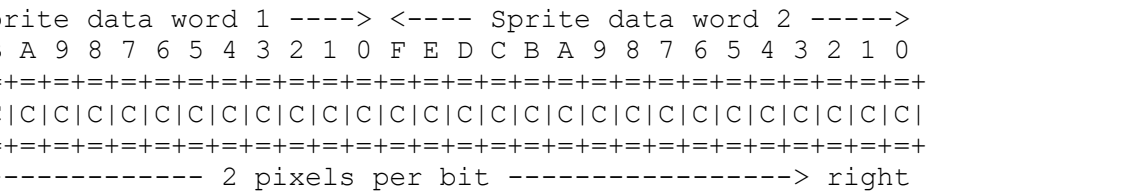
+-+-+ Does not automatically generate edge smoothing.



NOTES:

COLOR-OR; FORMAT = 111cccc

Optimized for high resolution mono-chromatic detail, this sprite has 64 pixels of color graphics. Pixel color is determined by the FORMAT. It differs from the AIR BRUSH sprite in only two respects. First, it does not participate in display prioritization; it outputs its color word whenever it has a pixel defined as non-transparent. And second, that color word output is logically 'or'ed into whatever color word exists from any other sprites in a manner analogous to the way that intensities are logically 'or'ed on the intensity bus. This is the only color sprite which is not prioritized and which 'or's its color on to the color bus. It is especially useful for creating color planed bit map displays (see Appendix for examples).



PROGRAMMING ELEMENT COLOR

C	FORMAT	color	comments
0	111%%%%	0000	transparent
1	" 0000	0000	transparent
1	" --	--	
1	" 1110	1110	color-14
1	" 1111 ------>		15-switch
	ZPOS+ZOFF <----		Color is a function of /Z

```
 1  0000%%%%  1111   closest to screen, color-15
 1   --           --
 1  1111%%%%  0000   furthest from screen, transparent


SCREEN GRAPHICS (one element out of 32 total elements)

    +-+-+        Does not automatically generate edge smoothing.
    |   |
    +-+-+        Note the similarities and differences between
--->|   |<--- 2 pixels this sprite and its kin, the AIR BRUSH &
                 TEXTURE sprites.

NOTES:
```

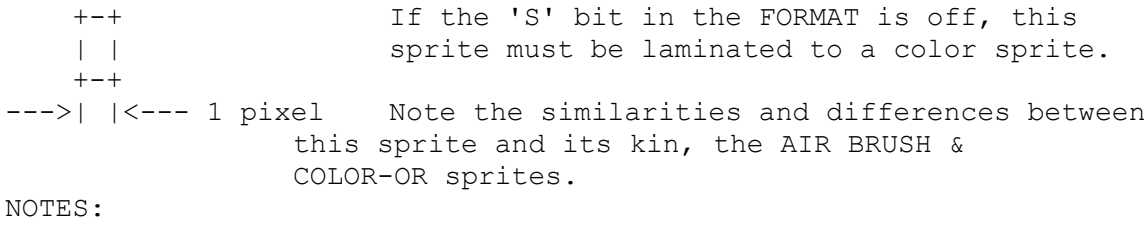
```
TEXTURE SPRITE; FORMAT = 10Siiii
                     ^
                     |
Self laminate bit ----------+
0 => Laminate on color sprite specified by LAMINATE parameter
1 => Self-laminate (always on)
```

Optimized for high resolution mono-intensity detail, this sprite has 32 pixels of intensity graphics. Pixel intensity determined by the FORMAT parameter.

```
 <----- Sprite data word 1 ----> <---- Sprite data word 2 ----->
 F E D C B A 9 8 7 6 5 4 3 2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|I|
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 left ----------------- 1 pixel per bit -----------------> right
```

```
================= PROGRAMMING ELEMENT INTENSITY =================

 I   FORMAT   intensity       comments
===  =======    ====      ============================
 0   101%%%%    %%%%      full LUM (no contrast)

 1    " 0000    0000      full LUM (no contrast)
 1     --    --
 1    " 1110    1110      2/16ths LUM (max contrast)

 1    " 1111 -----------> 15-switch ----------------------+
                                                         |
    ZPOS+ZOFF <---------  intensity is a function of /Z <-+
    =========
 1  +0000%%%%    1111    closest to screen, 1/16th LUM (max contrast)
 1    --     --
 1  +1111%%%%    0000    furthest from screen, full LUM (no contrast)
```

SCREEN GRAPHICS (one element out of 32 total elements)



NOTES:

```
EDGE ENHANCEMENT SPRITE; FORMAT = 000S000
                             ^
                             |
Self laminate bit -------------------+
0 => Laminate on color sprite specified by LAMINATE parameter
1 => Self-laminate (always on)
```

This sprite is useful to add edge smoothing to sprites which do not automatically generate edge smoothing or to further enhance an edge which has been automatically generated for further anti-aliasing.

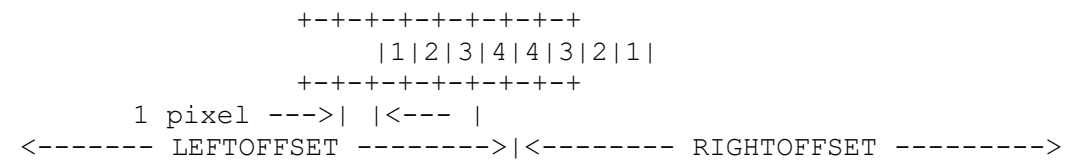
```
 <----- Sprite data word 1 ----> <---- Sprite data word 2 ----->
 F               7       3     0 F       B       7             0
+=-==-==-==-==+==-==+=-==-=+=-=-==+=-==-=+=-==-==-==-==+
|  LEFTOFFSET   |  I1   |  I2   |  I3   |  I4   |  RIGHTOFFSET  |
+=-==-==-==-==+==-==+=-==-=+=-=-==+=-==-=+=-==-==-==-==+
           left ----- 1 pixel -----> edge
          right <--- per nibble ---- edge
```

PROGRAMMING ELEMENT INTENSITY

In	intensity	comments
0000	0000	full LUM as stored in palette (no contrast)
1110	1110	2/16ths of LUM stored in palette (max contrast)

```
1111 ---------------> 15-switch  ---------------------+
                                                      |
ZPOS+ZOFF <-----------  intensity is a function of /Z <-+
=========
+0000%%%%  1111  closest to screen, 1/16th LUM (max contrast)
+1111%%%%  0000  furthest from screen, full LUM (no contrast)
```

DETAIL OF EDGE

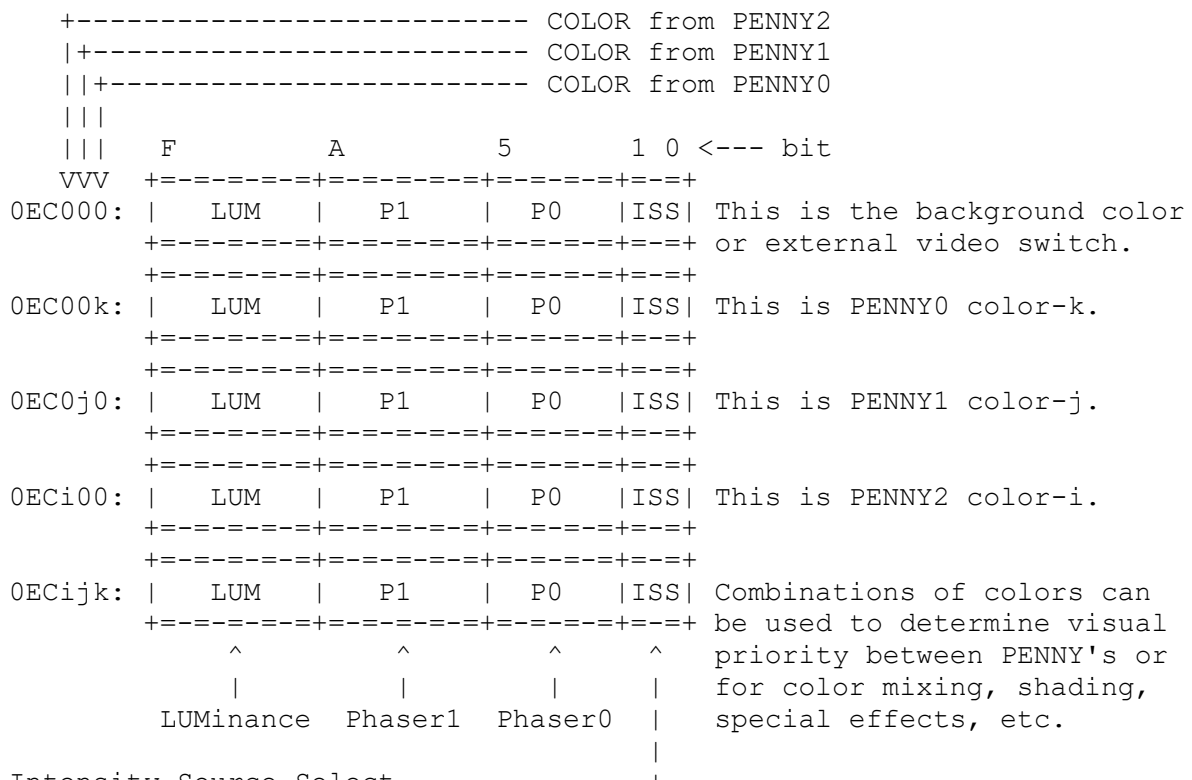


RIGHTOFFSET is only needed if the sprite is to be reflectable in which case LEFTOFFSET plus RIGHTOFFSET must be a constant for all lines.

If the 'S' bit in the FORMAT is off, this sprite must be laminated to a color sprite.

NOTES:

PALETTE MEMORY MAP:



00 ==> intensity determined by intensity sprite from PENNY0
01 ==> intensity determined by intensity sprite from PENNY1
10 ==> intensity determined by intensity sprite from PENNY2
11 ==> intensity (if any) commensurate with stored luminance & phasers

TWO METHODS OF GENERATING COLORS:

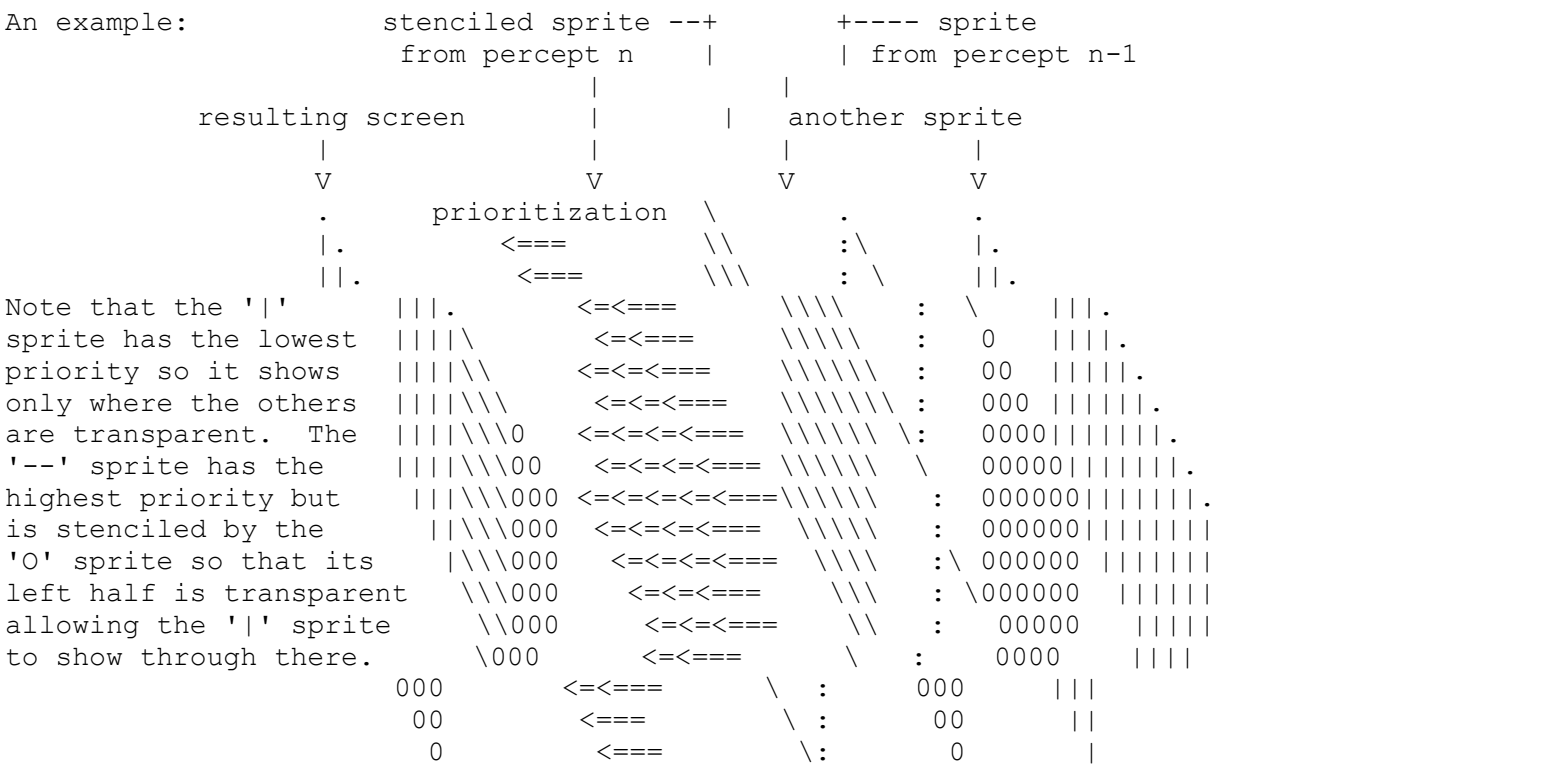
1, Find values for LUM, P1 & P0 from the color charts in the APPENDIX or

2, Calculate values for LUM, P1 & P0 as follows:

```
;; select desired amount of intensity of primary colors
off = 0 =< R,G,B =< 23.9 = brightest

;; store LUMinance
LUM = INTEGER[.30*R+.59*G+.11*B+.49]

;; store Phaser1
TEMP = .877*(.70*R-.59*G-.11*B)
P1 = INTEGER[TEMP-.49]  ; if TEMP < 0
P1 = INTEGER[TEMP+.49]  ; if TEMP >= 0

;; & store Phaser0
TEMP = .493*(.89*B-.30*R-.59*G)
P0 = INTEGER[TEMP-.49]MODULO16   ; if TEMP < 0
P0 = INTEGER[TEMP/2+.49]MODULO16 ; if TEMP >= 0
```

WHAT IS STENCILING ?

With stenciling on, the resultant display will be the topological intersection of the stenciled sprite and the stenciling sprite.



Stenciling is performed irrespective of z-position. Stenciling is performed in x & y only.

Stenciling is different from masking. A mask defines the area where a sprite IS NOT to appear, however, a stencil defines the area where a

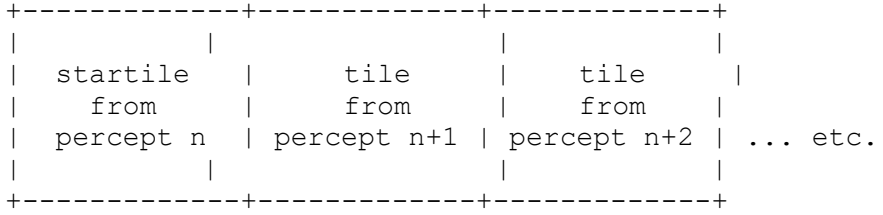
sprite IS to appear. Masks are data intensive and they are useless except as masks. Stencils are data conservative since only the area of interest is reproduced in the data and a stencil can also be used as a graphic sprite for display purposes.

Stenciling does not affect priority and is not affected by priority. The stenciling sprite does not have to have priority to perform the stenciling function.

All intensity sprites are automatically laminated. Laminating is like stenciling except that the laminating sprite, which must be a color sprite, must also have priority for the current pixel.

WHAT IS TILING ?

It is often desired to cover an area with percepts laid edge to edge to form a large surface. That is tiling. A tiling chain begins with the lowest numbered percept to the left and proceeds to the right with incrementally higher numbered percepts, thus:

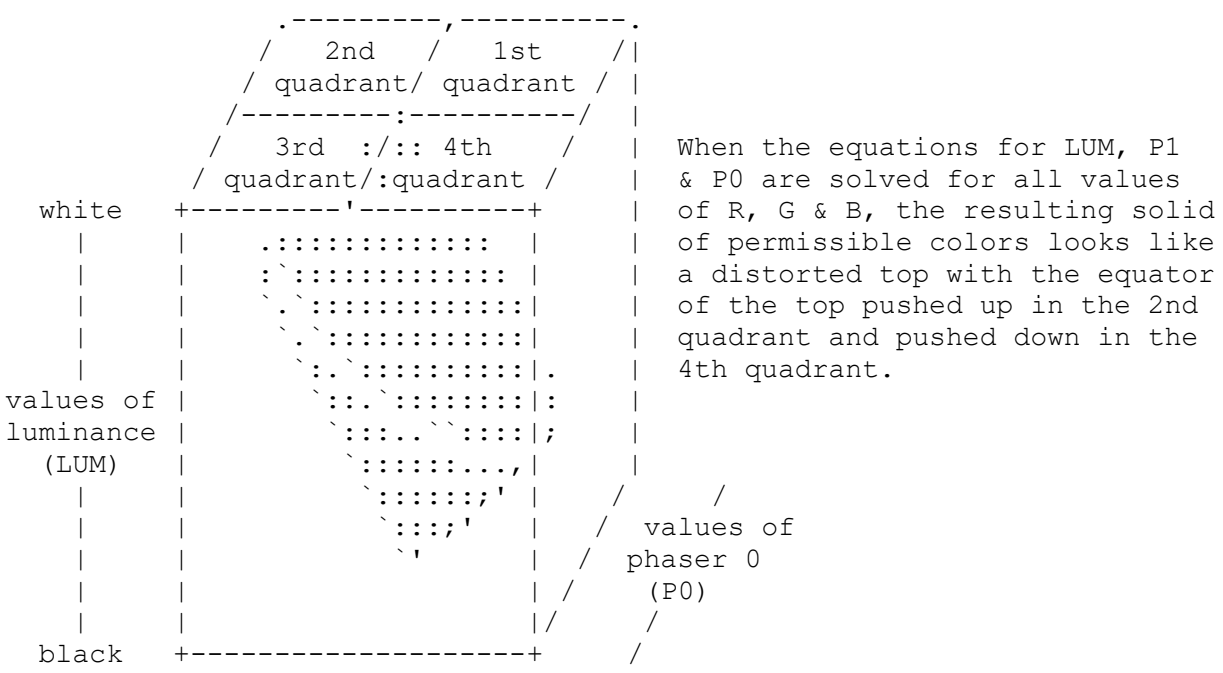


Each sprite must have the same number of lines of data and the same Y-POSition and Y-OFFset. But, they can be of different FORMAT types and they can even be on different z-levels. The left-most sprite must be a Startile with X-POSition & X-OFFset defined. The rest of the sprites in the tiling chain ignore X because each begins when its left-hand neighbor finishes.

A tiling chain can be of any length and it can be made up of any mix of reflected and unreflected sprites.

Tiling does not affect priority and is not affected by priority.

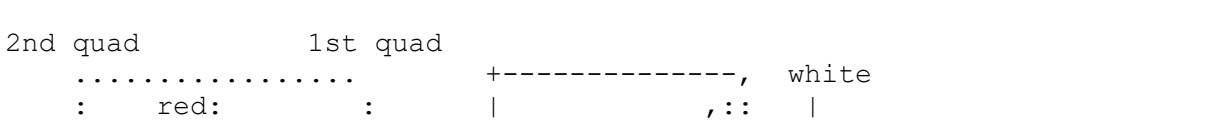
COLOR SPACE - INTERPRETING THE COLOR CHARTS WHICH FOLLOW

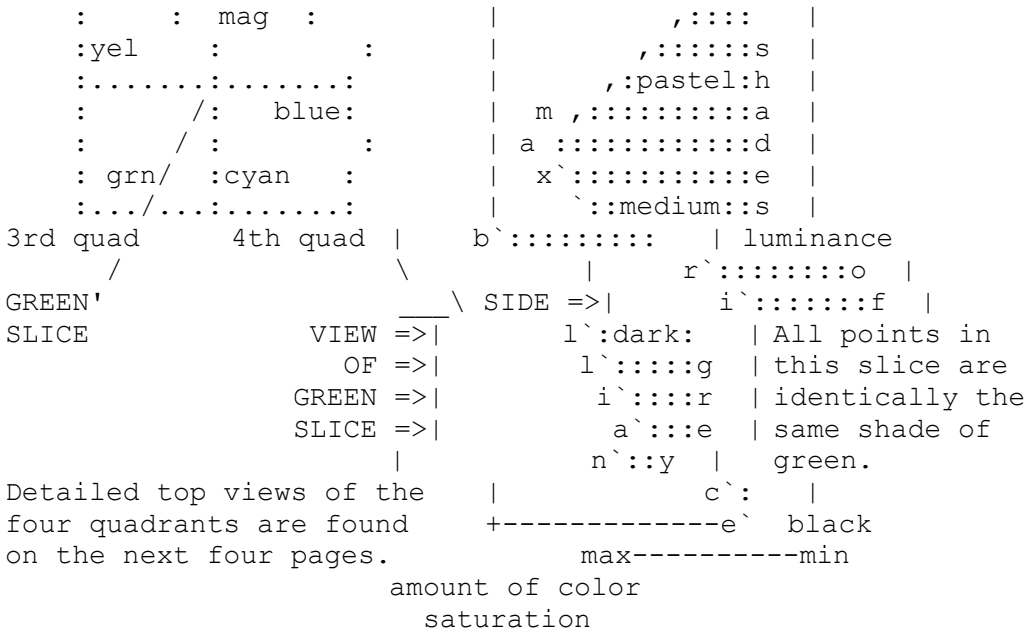


When the equations for LUM, P1 & P0 are solved for all values of R, G & B, the resulting solid of permissible colors looks like a distorted top with the equator of the top pushed up in the 2nd quadrant and pushed down in the 4th quadrant.

The tilt of the equator of the top has a maximum in yellow & a minimum in blue.

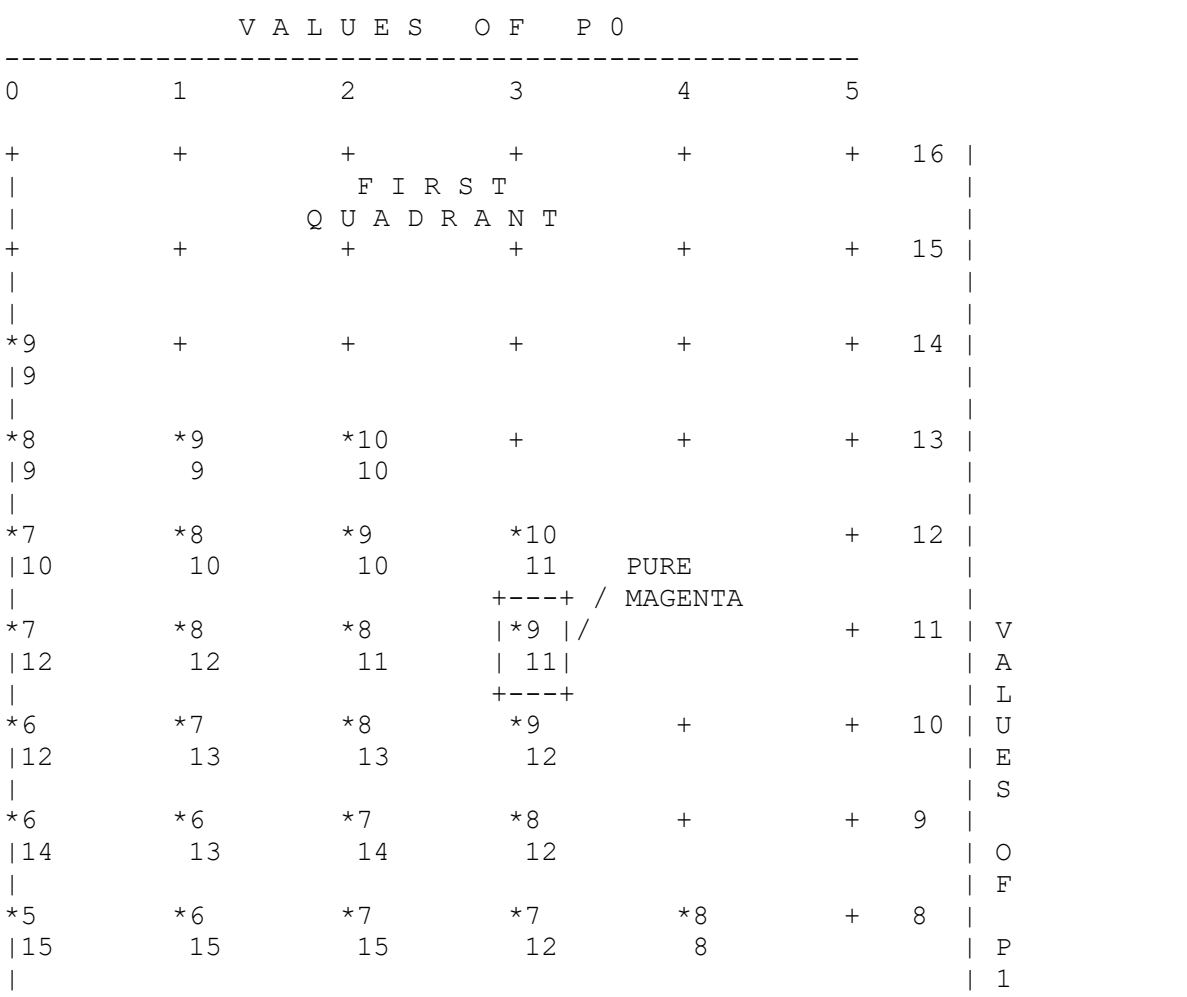
TOP VIEW OF COLOR SPACE

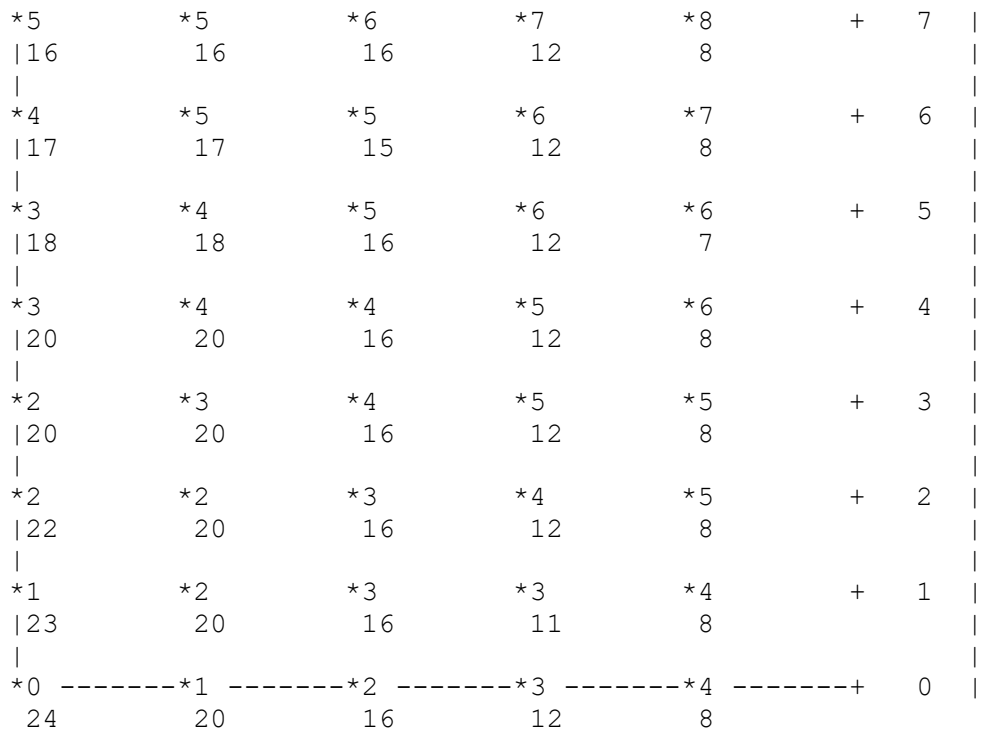




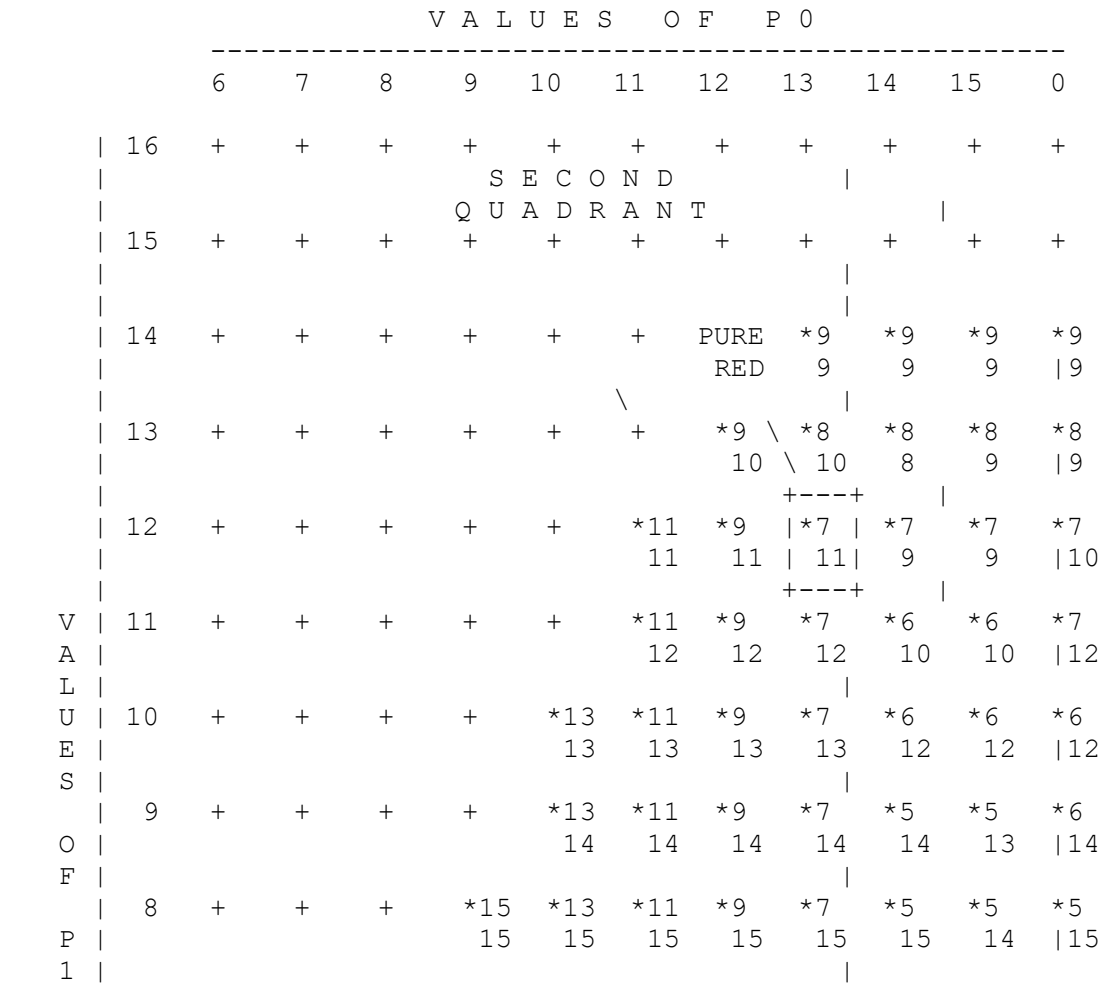
legal NTSC color point ----> *m <--- minimum allowable LUM for this color point
n <--- maximum allowable LUM for this color point

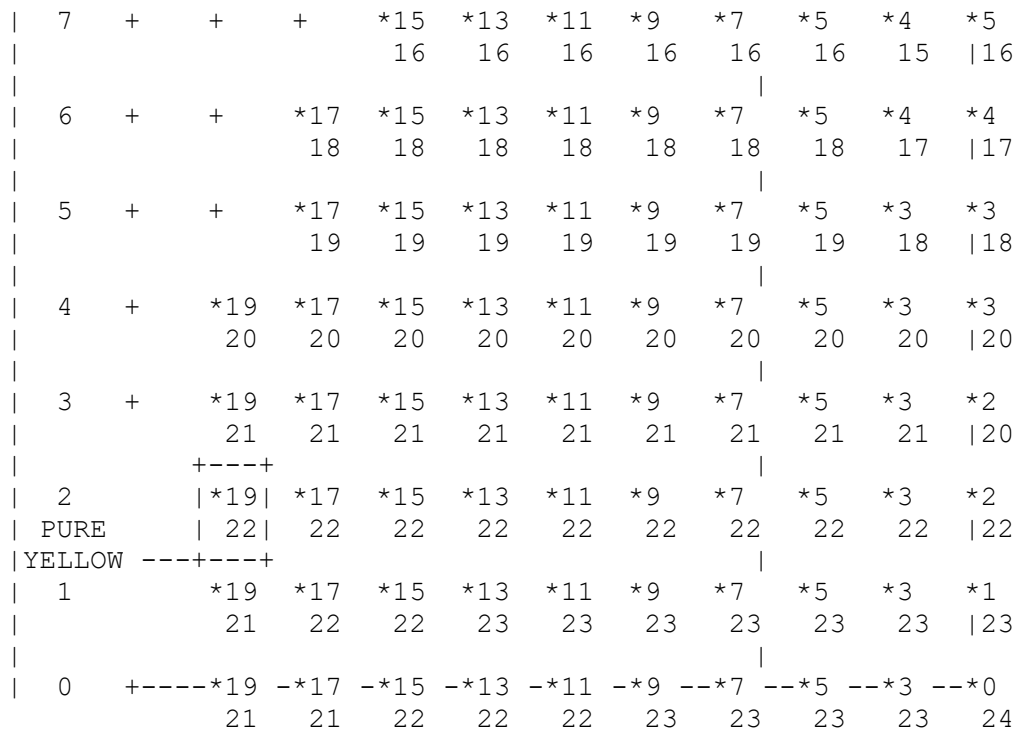
NTSC COLOR CHART

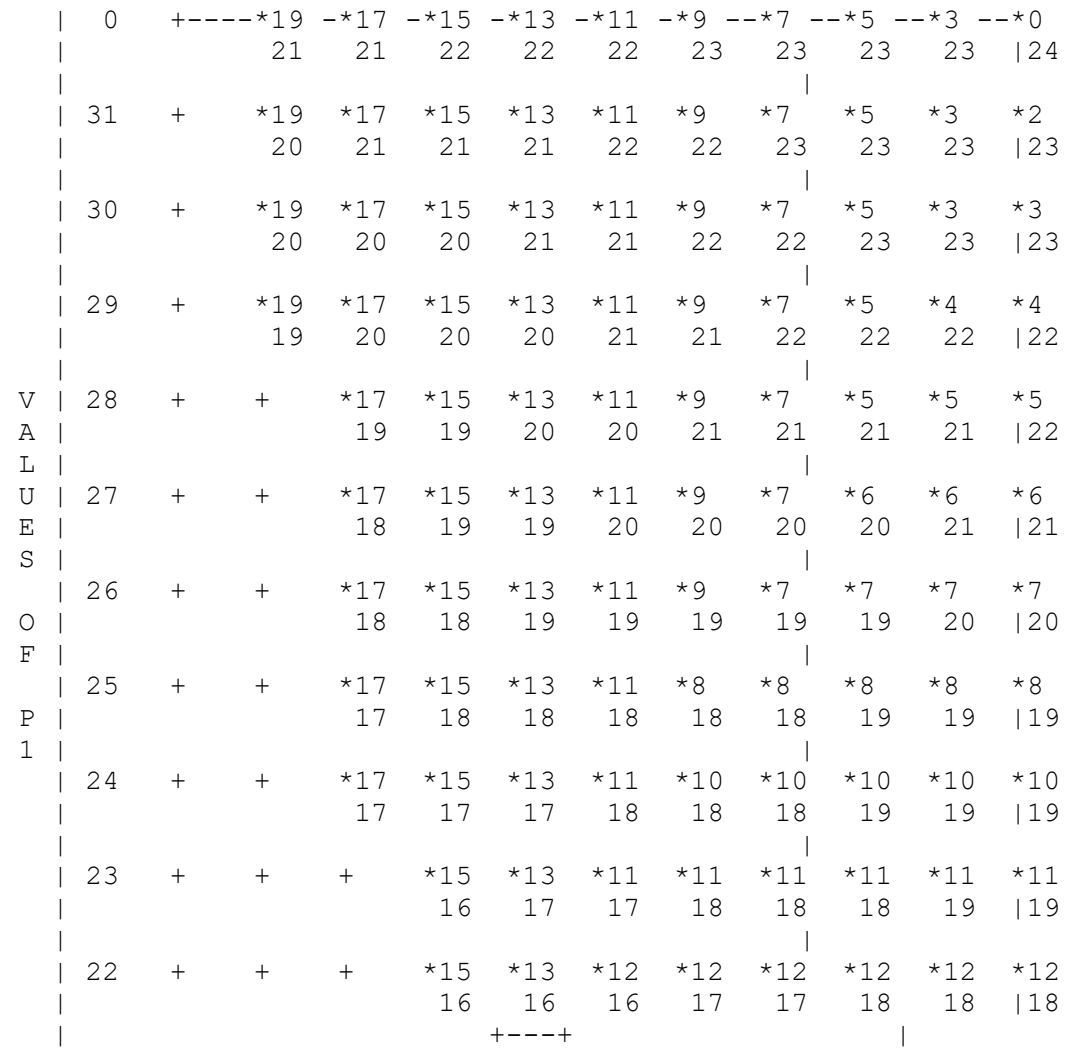


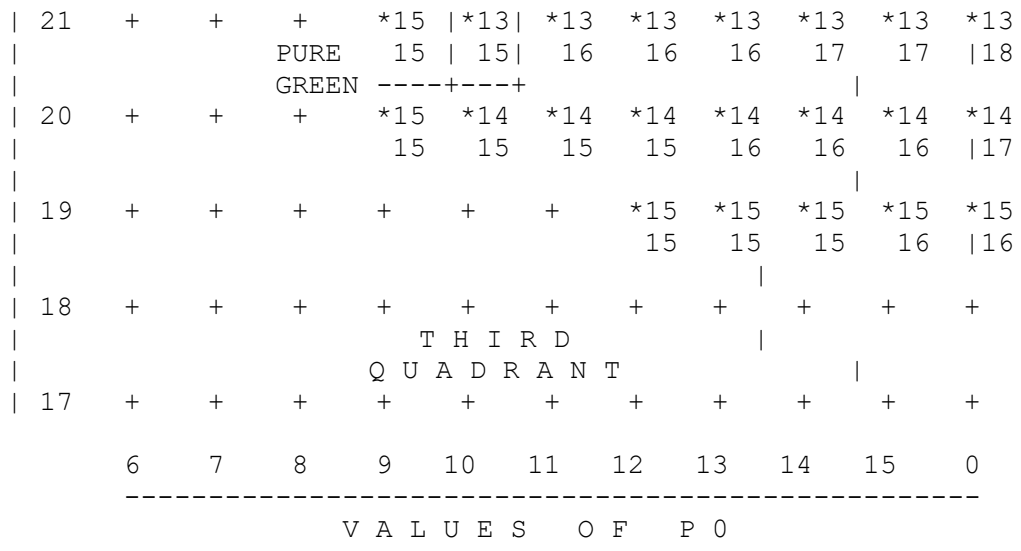


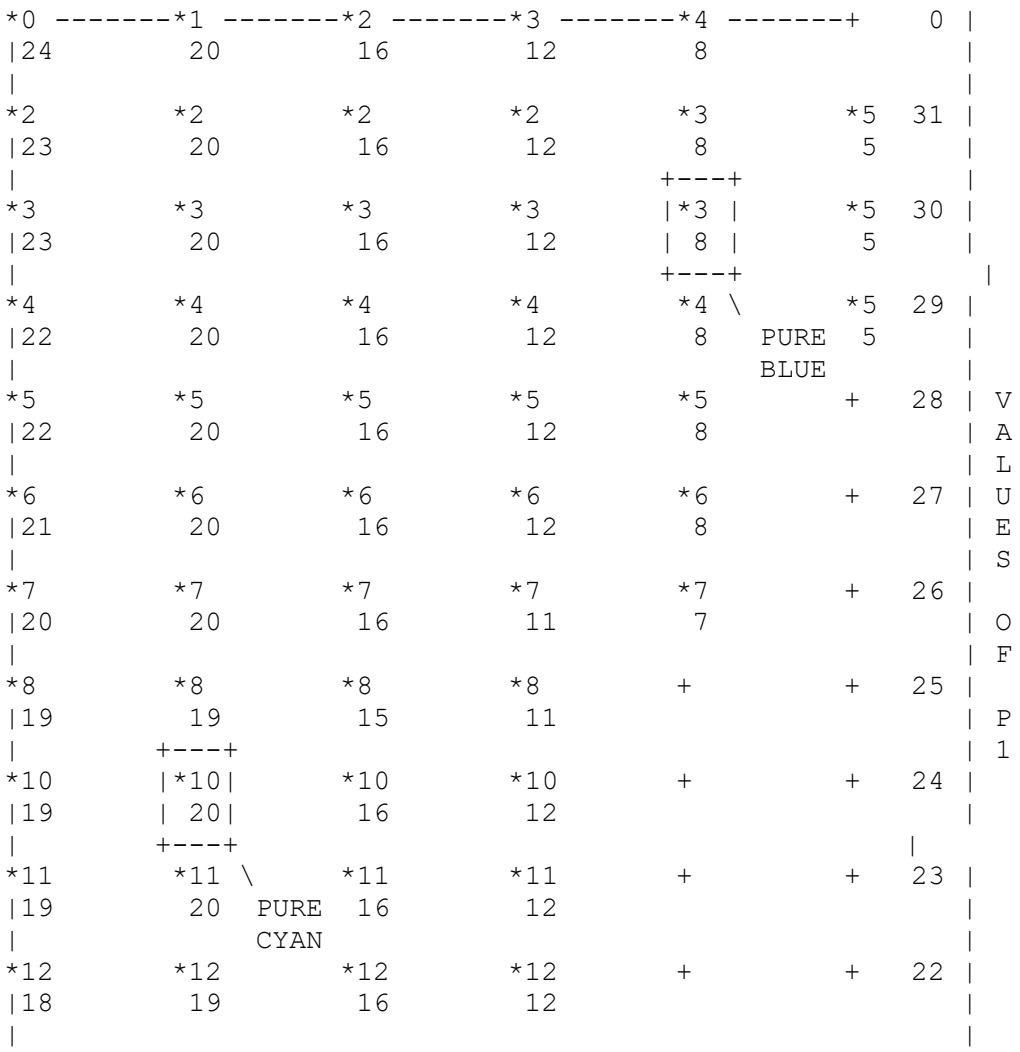
CROSS POINTS (+) ARE ILLEGAL NTSC COLORS...BUT THEY CAN BE USED FOR MONITORS. MAXIMUM AND MINIMUM LUM VALUES DO NOT APPLY TO MONITORS EITHER.

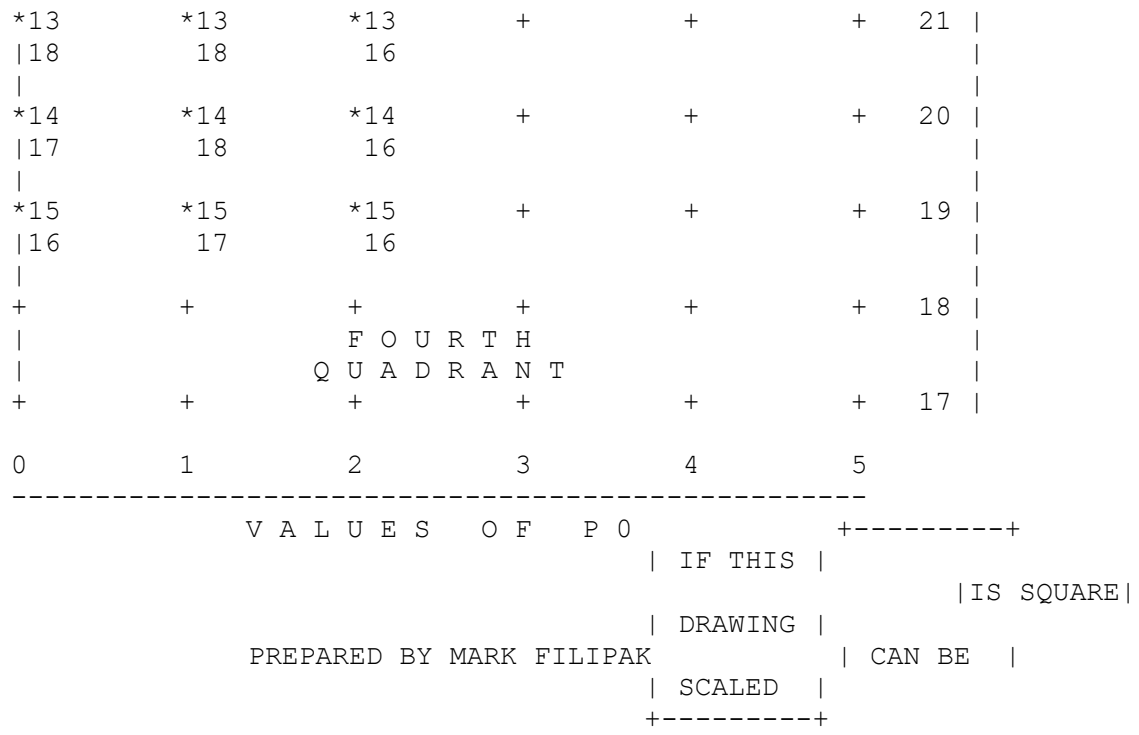




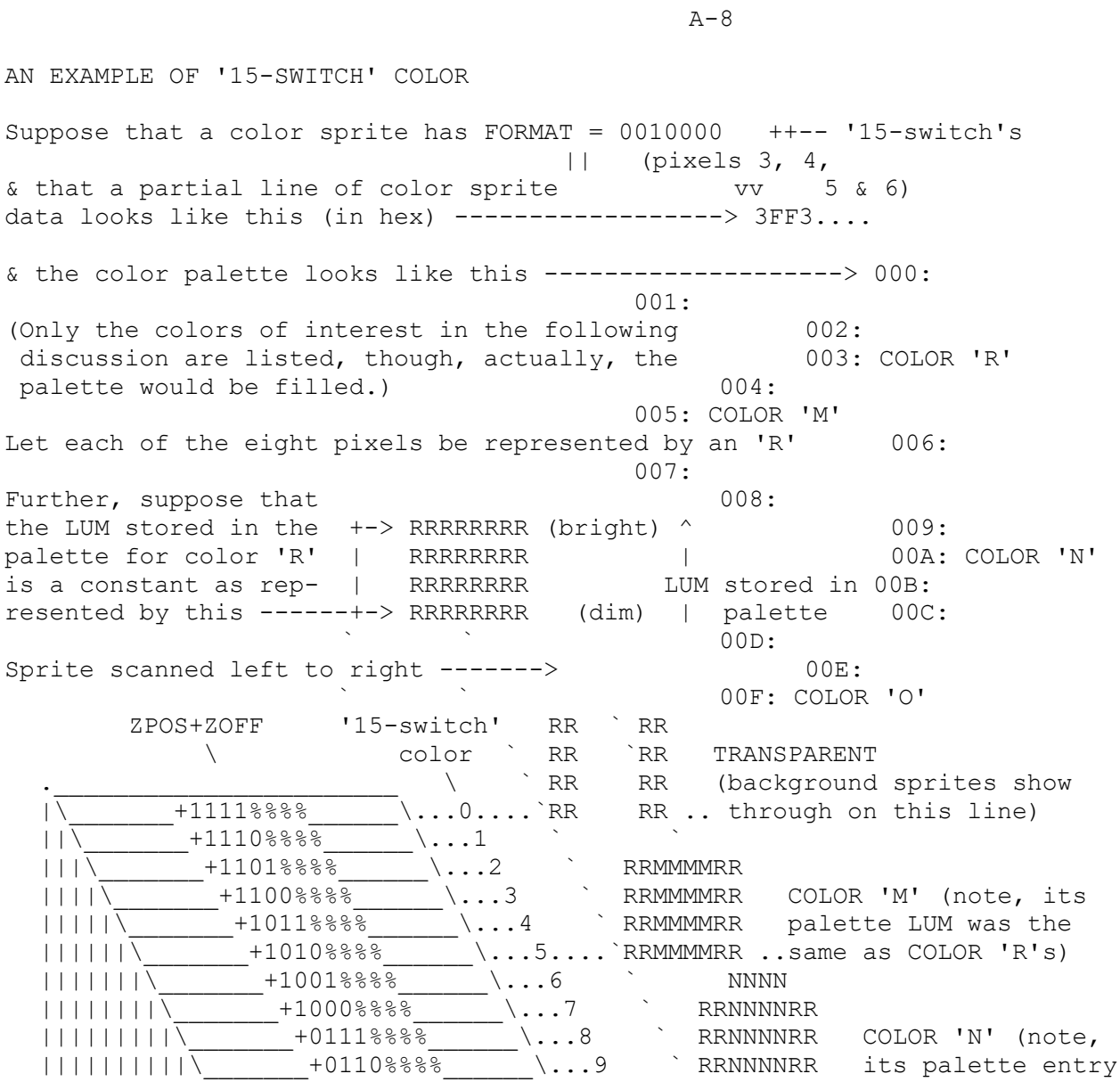


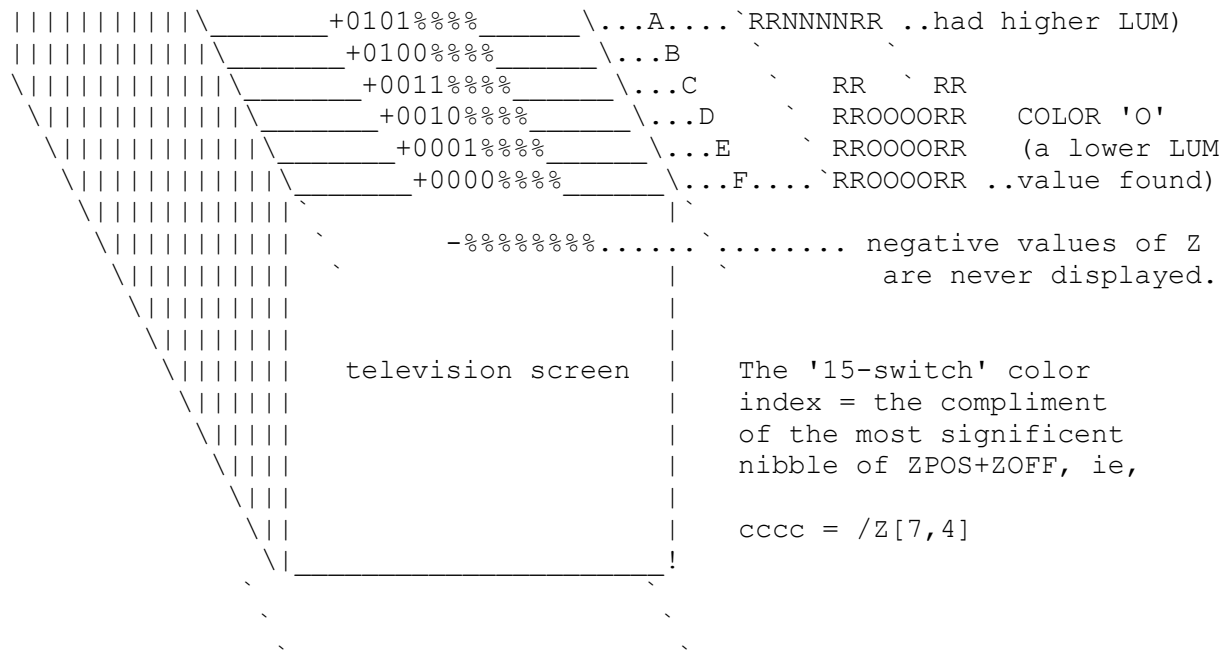




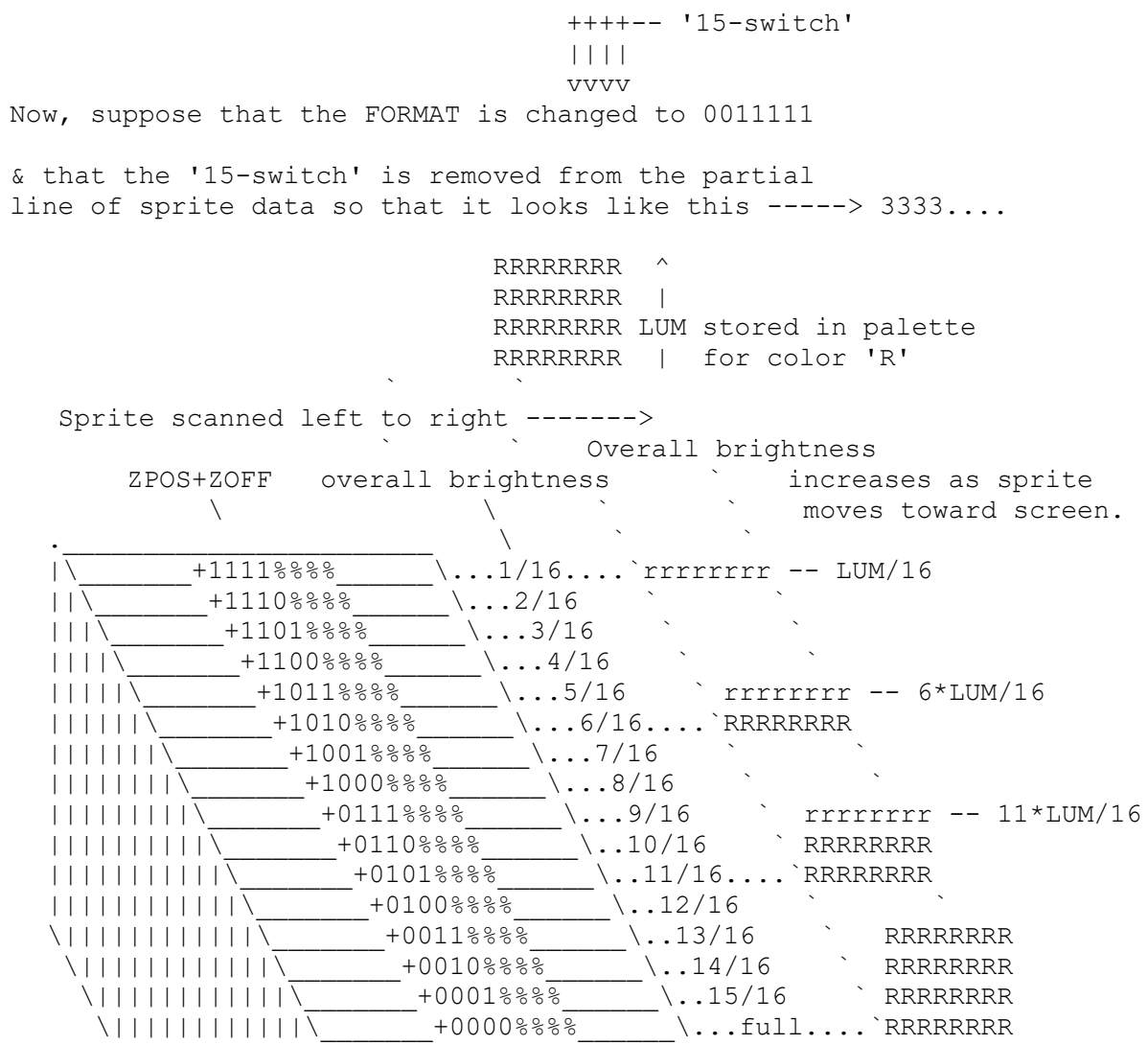


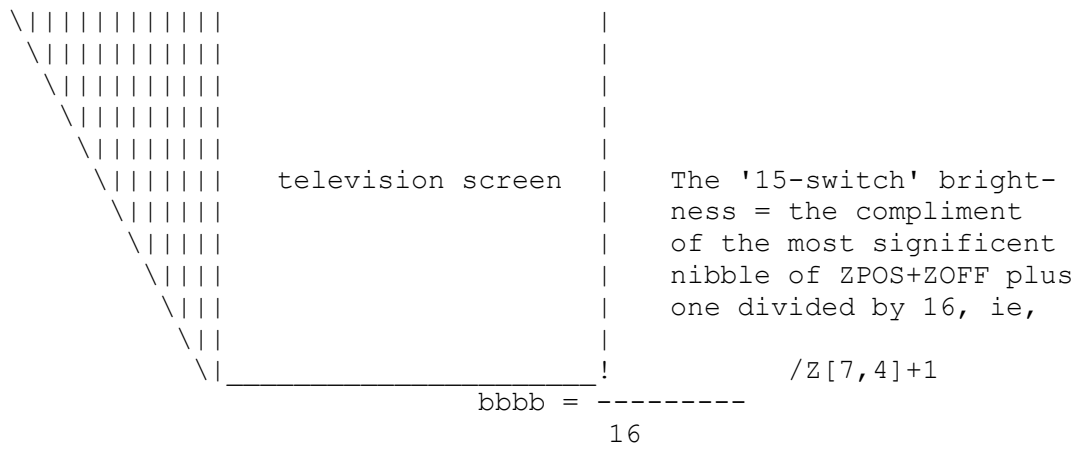
AN EXAMPLE OF '15-SWITCH' COLOR





AN EXAMPLE OF '15-SWITCH' BRIGHTNESS



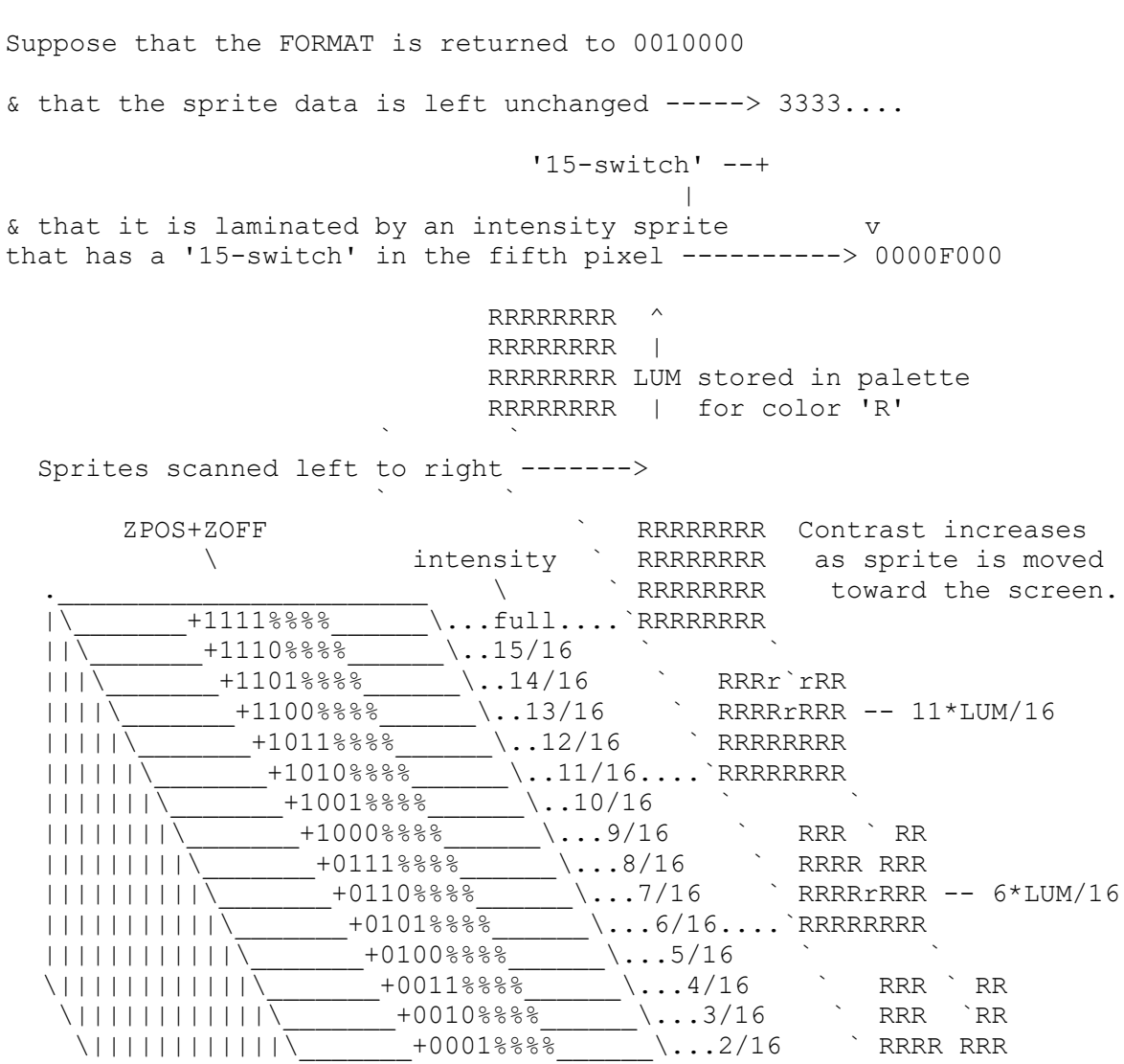


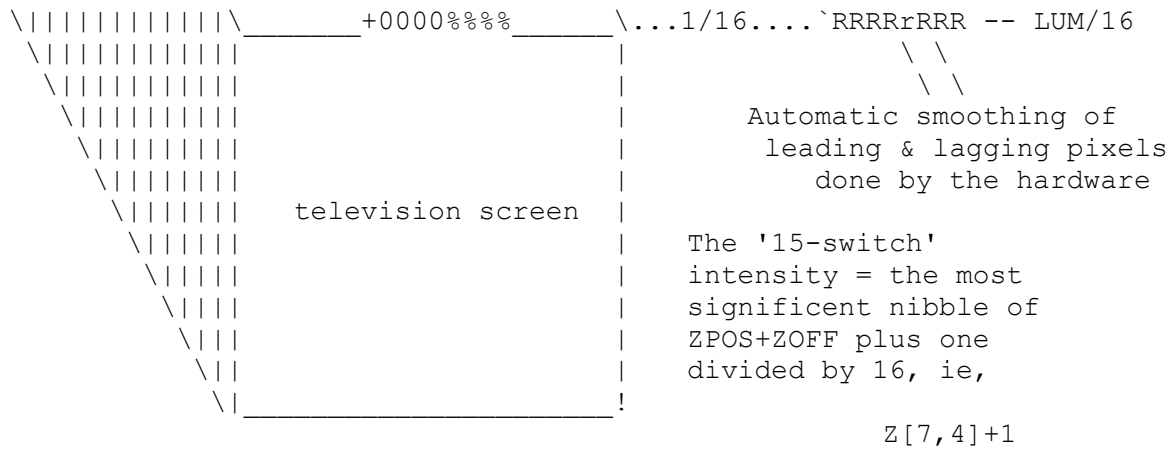
AN EXAMPLE OF '15-SWITCH' INTENSITY

Suppose that the FORMAT is returned to 0010000

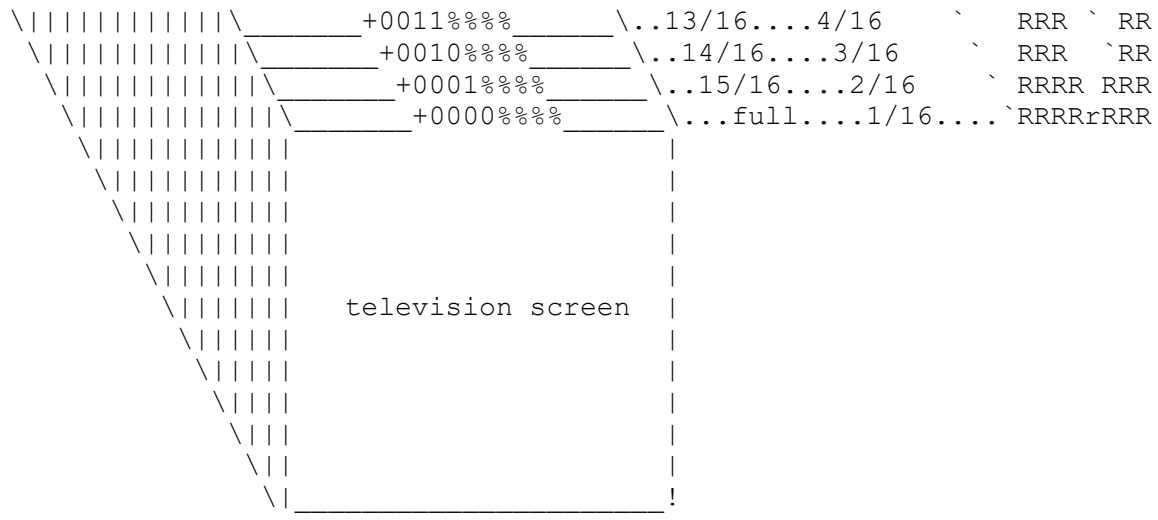
& that the sprite data is left unchanged -----> 3333....

& that it is laminated by an intensity sprite
that has a '15-switch' in the fifth pixel



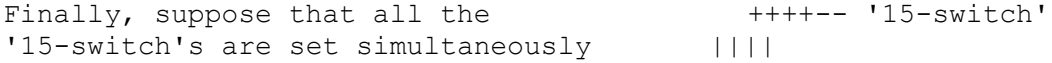


AN EXAMPLE OF COMBINED '15-SWITCH' BRIGHTNESS & INTENSITY

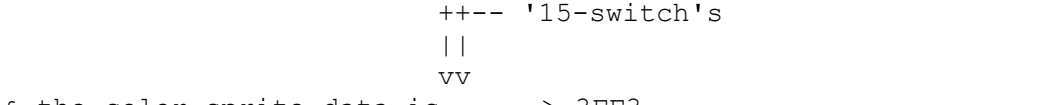


AN EXAMPLE OF '15-SWITCH' COLOR, BRIGHTNESS & INTENSITY

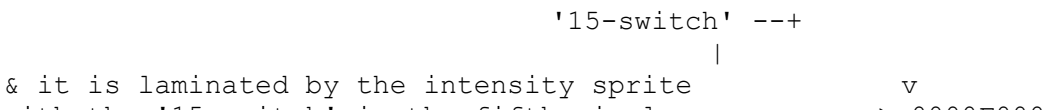
Finally, suppose that all the '15-switch's are set simultaneously



so that the FORMAT is set to --------> 0011111



& the color sprite data is -----> 3FF3....



& it is laminated by the intensity sprite with the '15-switch' in the fifth pixel ------------> 0000F000