CONFIDENTIAL

AGREEMENT REGARDING

CONFIDENTIALITY OF INFORMATION

1. AMIGA CORPORATION (the "Company") and ATARI, INC. ________________ ("Recipient") are engaged in discussions in contemplation of or in furtherance of a business relationship.

2. To further the business relationship, Recipient may have access to and have disclosed to it certain valuable information relating to the Company which is of a confidential nature (hereinafter referred to as "the Company Information") concerning any or all of the following: trade secrets, know-how, inventions, techniques, processes, algorithms, software programs, schematics, software source documents, contracts, customer lists, financial information, sales and marketing plans and information and business plans.

3. Recipient agrees that it shall neither use the Company Information nor circulate it within its own organization, except to the extent necessary for:

(a) negotiations, discussions and consultations with personnel or authorized representatives of the Company; and

(b) any purpose the Company may hereafter authorize in writing.

4. Recipient further agrees that it shall not publish, copy or disclose any Company Information to any third party and shall use its best efforts to prevent inadvertent disclosure of the Company Information to any third party.

5. Recipient's obligations with respect to any portion of the Company Information as set forth above shall terminate when Recipient can document that:

(a) It was in the public domain at the time it was communicated to Recipient by the Company; or

(b) It entered the public domain through no fault of Recipient subsequent to the time it was communicated to Recipient by the Company; or

(c) It was in Recipient's possession free of any obligation of confidence at the time it was communicated to Recipient by the Company; or

(d) It was rightfully communicated to Recipient free of any obligation of confidence subsequent to the time it was communicated to Recipient by the Company.

6. All materials including, without limitation, documents, drawings, models, apparatus, sketches, designs and lists furnished to Recipient by the Company and which are designated in writing to be the property of the Company shall remain the property of the Company and shall be returned to the Company promptly at its request, together with any copies thereof.

7. The work product of any services performed by the Company or of any writings, discoveries, inventions and innovations resulting from such services shall be and remain the property of the Company unless otherwise agreed in writing signed by both parties hereto.

8. This Agreement shall govern all communications between Recipient and the Company that are made during the period from the effective date of this Agreement to the date on which either party receives from the other written notice that subsequent communications shall not be so governed, provided, however, that Recipient's obligations under Paragraphs 3 and 4 shall continue unless terminated pursuant to Paragraph 5 hereof.

9. Either party shall have the right to obtain a preliminary judgment on any equitable claim in any court of competent jurisdiction, where such judgment is necessary to preserve property and/or proprietary rights under this Agreement. Such judgment shall remain effective as long as the terms of the judgment so provide.

10. Any notice required to be given under this Agreement shall be deemed received five (5) days after mailing if sent by registered or certified mail to the addresses of the parties set forth below, or to such other address as either of the parties shall have furnished to the other in writing.

11. In the event of invalidity of any provision of this Agreement, the parties agree that such invalidity shall not affect the remaining portions of this Agreement, and further agree to substitute for the invalid provision a valid provision which most closely approximates the intent and economic effect of the invalid provision.

12. The effective date of this Agreement shall be November 21, 19 83 .

RECIPIENT:

ATARI, INC.

By [signature]
Typed Name: Mike Albaugh
Title:
Address: 1501 McCarthy Blvd.
Milpitas, Ca. 95035

AMIGA CORPORATION

By [signature]
Typed Name: David S. Morse
Title: President
Address: 3350 Scott Blvd. Bldg. 7
Santa Clara, Ca. 95051

Features Summary

COPY # 1

- Has a Motorola 68000 Main Processor and 128K bytes of random access memory..
- Has a detached keyboard.
- Can read Apple [TM] compatible disk text files.
- Can be programmed in BASIC, FORTH or Assembler. BASIC and FORTH are built into the basic system.
- Lorraine BASIC is compatible with APPLESOFT [TM], but adds commands which can take advantage of the powerful new features of this computer.
- Can handle a large number of different kinds of controllers, which in turn allows for a wide variety of methods of data input.
- Has four independently controllable sound generators. Each of these generators may be programmed to produce a wide variety of tones and kinds of waveforms. Many different kinds of musical effects are possible.
- Can produce 40 columns by 25 lines of text
- Can produce 80 columns by 25 lines of colored text on a video monitor without adding any extra-cost 80-column cards. Can mix multi-colored graphics and multi-colored text on the same display.
- Can produce dozens of easily controlled multi-colored moveable objects called "sprites" on the screen.
- Can define one or two independently moveable normal or high resolution graphics planes called "playfields". Up to 4096 color choices for each picture element for normal resolution playfields, or up to 16 color choices for each picture element for a high resolution playfield.
- Can define whether some of the sprites are more important than some of the playfields, so that if one object is supposed to be "in front of" another, it will appear that way on the display. It also can sense and report collisions between the sprites and the playfields or between individual sprites.
- Can rapidly move and color-fill graphic shapes.
- Has a special-effects coprocessor which can produce multiple part-way-through-the-display changes in the system operating mode.

LORRAINE Computer System — Preliminary Technical Summary

LORRAINE is a third generation, low cost, high performance, graphics and sound system for state of the art video game and personal computer applications.

The system includes three proprietary, custom ICs controlled by a Motorola 68000 32/16 bit microprocessor. These chips provide extraordinary color graphics on a standard TV or on an RGB color monitor, with resolution and depth to display coin-op quality, first person video games, cartoons, low resolution photographs, or up to 80 character screens. The sound circuits can duplicate complex waveforms on each of four channels, matching commercial synthesizers in quality.

The graphics hardware provides a fully bit-mapped image of up to 320H X 200V pixels each six bits deep for a TV or up to 640H X 400V pixels each four bits deep for an RGB monitor. Each pixel selects a color value from a 32 entry color palette providing 12 bits of resolution including separate control of all three aspects of the color signal: Hue, Intensity, and Saturation. The hardware supports slicing the bit map into two levels of playfield plus background, with automatic priority overlay of the playfields. In addition, the hardware supports eight programmable "sprite processors", each providing an arbitrary number of images 16 pixels wide, arbitrarily tall, and two bits deep which can be rapidly positioned anywhere on the screen with selectable overlay priority. Pairs of such processors can be "attached" providing 4 bits of color depth for each sprite image. The resulting screen image can be scrolled smoothly in both the vertical and horizontal directions.

The color depth of the image may vary from place to place on the screen, saving both memory space and bandwidth in those portions of the image not requiring many simultaneous colors. In addition, two of the six color planes may be used in the "hold and control" mode to select between normal indexing of the color palette or direct setting of one of the color components (Hue, Intensity, or Saturation) while preserving the other two components from the pixel immediately to the left. The hold and control mode allows the construction of very detailed images involving either gray scale shading, pastel highlighting, rainbow color effects or any combination of the three. The Display Instruction Processor (described below) may also be used to change the color palette on the fly.

LORRAINE supports hardware detection of "collisions" involving either of the playfield images and each of the 4 sets of attachable sprites. For purposes of collision detection, each of the playfield "objects" may be further refined, indicating that only collisions with a given color or excluding a given color are to be detected. The collision accumulator can be polled and cleared at any time, allowing the detection of separate collisions in different portions of the image.

LORRAINE includes a hardware "Bit-Blit" co-processor, which may be used to create and move several dozen additional objects in the bit map each frame time, saving and restoring the background image as necessary. The Blitter also provides hardware support for line drawing and polygon filling functions. From a personal computer perspective, the Blitter provides a generalized hardware capacity for "desk-top" window management, easily surpassing the software mechanisms underlying such systems as the Apple LISA (TM).

Each LORRAINE audio channel plays an "audio map" of arbitrary length with frequency and volume set separately. The audio maps consist of 8 bit "delta" samples describing the waveform to be produced. Each map may be "played" at a sampling rate of up to 30 KHz, or any slower rate selectable with fine resolution. Left alone, each channel automatically repeats its audio map an arbitrary number of times, making the generation of sustained tones a trivial task involving very little memory. Since each map describes an arbitrary waveform, a three or four note musical chord can easily be generated by a single channel. LORRAINE produces stereo sound output, normally by summing pairs of audio channels. Alternatively one audio channel of each pair may be configured to modulate the other channel both by amplitude and frequency. Since the modulating channel may be sampled at a rate distinct from the normal channel, envelope functions and frequency modulation synthesis effects are easy to achieve.

Frame synchronization, control register updates, sprite repositioning and automatic color palette and audio channel updates can all be performed by LORRAINE's programmable Display Instruction Processor. The DIP acts as yet another co-processor, freeing the 68000 to execute program logic.

Other built-in I/O includes a keyboard controller, two Atari (TM) compatible game controllers with trackball/mouse logic, a serial port to support a modem, and a mini-floppy disc controller.

The standard configuration includes 128K bytes of graphics/audio/general purpose RAM and 64K bytes of resident firmware ROM. LORRAINE may be cartridge extended with up to 256K bytes of additional ROM or RAM. In addition, all 68000 data, address, and control lines are accesible, allowing LORRAINE to be integrated with a wide variety of memory, peripheral, and bus-master devices.

LORRAINE includes both high and low-level firmware support for graphics and audio synthesis. Particular emphasis has been placed on convenient high-performance access to the hardware for video game applications. LORRAINE will be packaged with a general purpose operating system, a BASIC interpreter, a FORTH interpreter, and several general purpose utilities.

LORRAINE COMPUTER FEATURES ROAD MAP

OPERATIONAL

(CHIP)

68000 processor

128K bytes RAM

(1) DMA control

(3) Interrupts

(3) Serial I/O (UART)

Parallel I/O

(2) Display Object Priorities

(2) Collisions between display objects

(1) Readable Beam Location

(1) Display-synchronized coprocessor ("COPPER")

- wait for beam to reach a spot
- store a new value to a register

(1) Hardware assisted block data move ("BLITTER")

- 3 optional sources, separate block structure for each
- masking/shifting
- logical combination of source data bits
- optional store to one destination
- LINE DRAWING MODE
- AREA FILL MODE

TACTILE

(CHIP)

Keyboard

Joystick/ Joyboard

(3) Mouse

(3) Proportional controller

AUDIBLE

(CHIP)

(3) 4 channels

(3) waveform shape definition

(3) modulation either FM, AM, or both

CONFIDENTIAL

AMIGA CORPORATION
3350 SCOTT BLVD., NO. 7
SANTA CLARA, CA 95051

VISUAL (ALL CHIP 2)

"Stationary"

Background Color

Playfields(2)

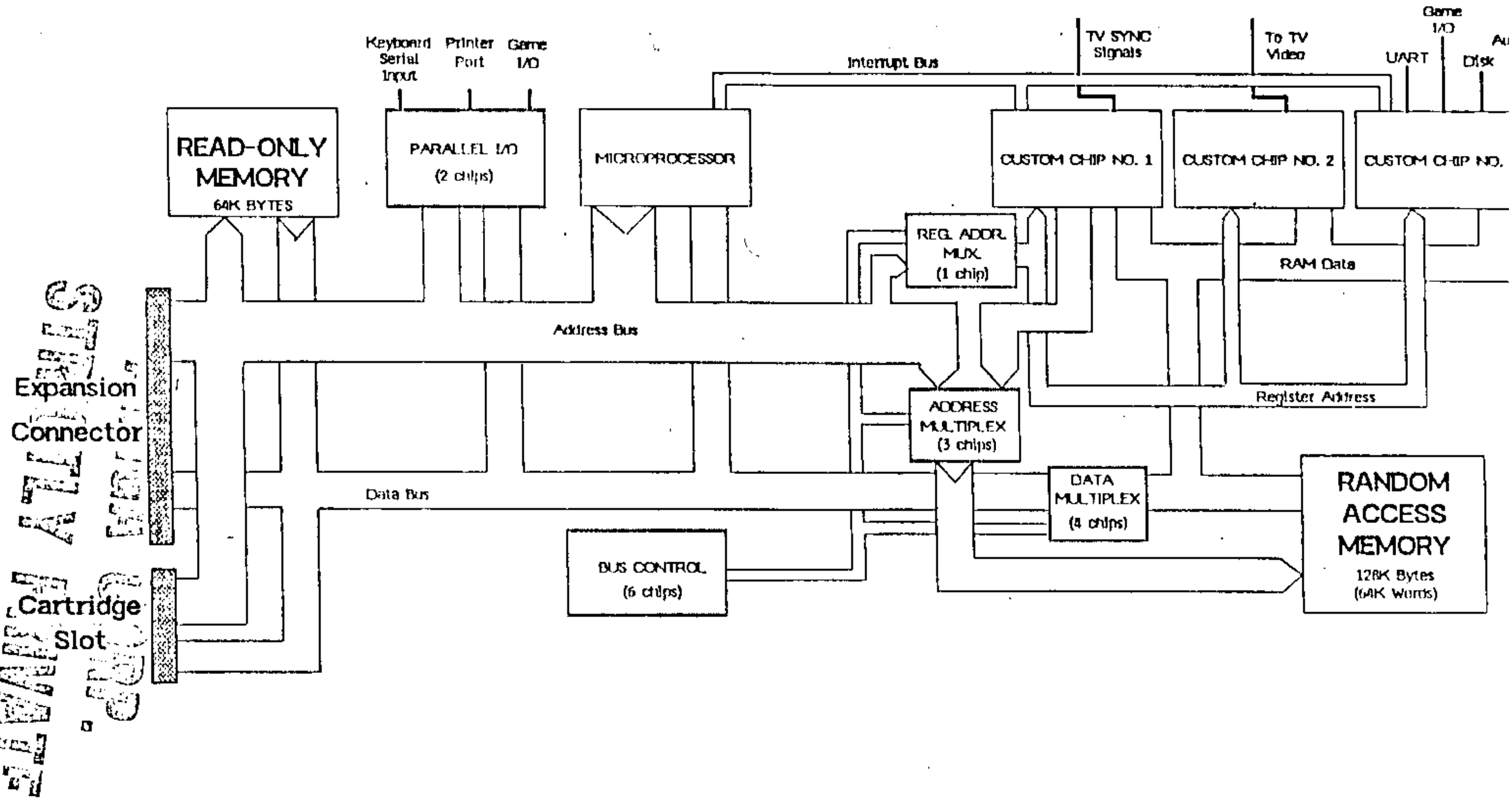
- Variable Shift
- Controllable onscreen position
- Bit-mapped data combined as bit-planes(6)
  - Partial Display of larger picture using modulo
  - Resolution: normal/high 320/640 x 200/400
  - Colors Available for pixels in each playfield
    - If 1 or 2 playfields: 2 colors, 4 colors, 8 colors
    - If 1 playfield: 16 colors, 32 colors
    - Special mode for 1 or 2 playfields: 4096 colors onscreen simultaneously

"Easily Moveable"

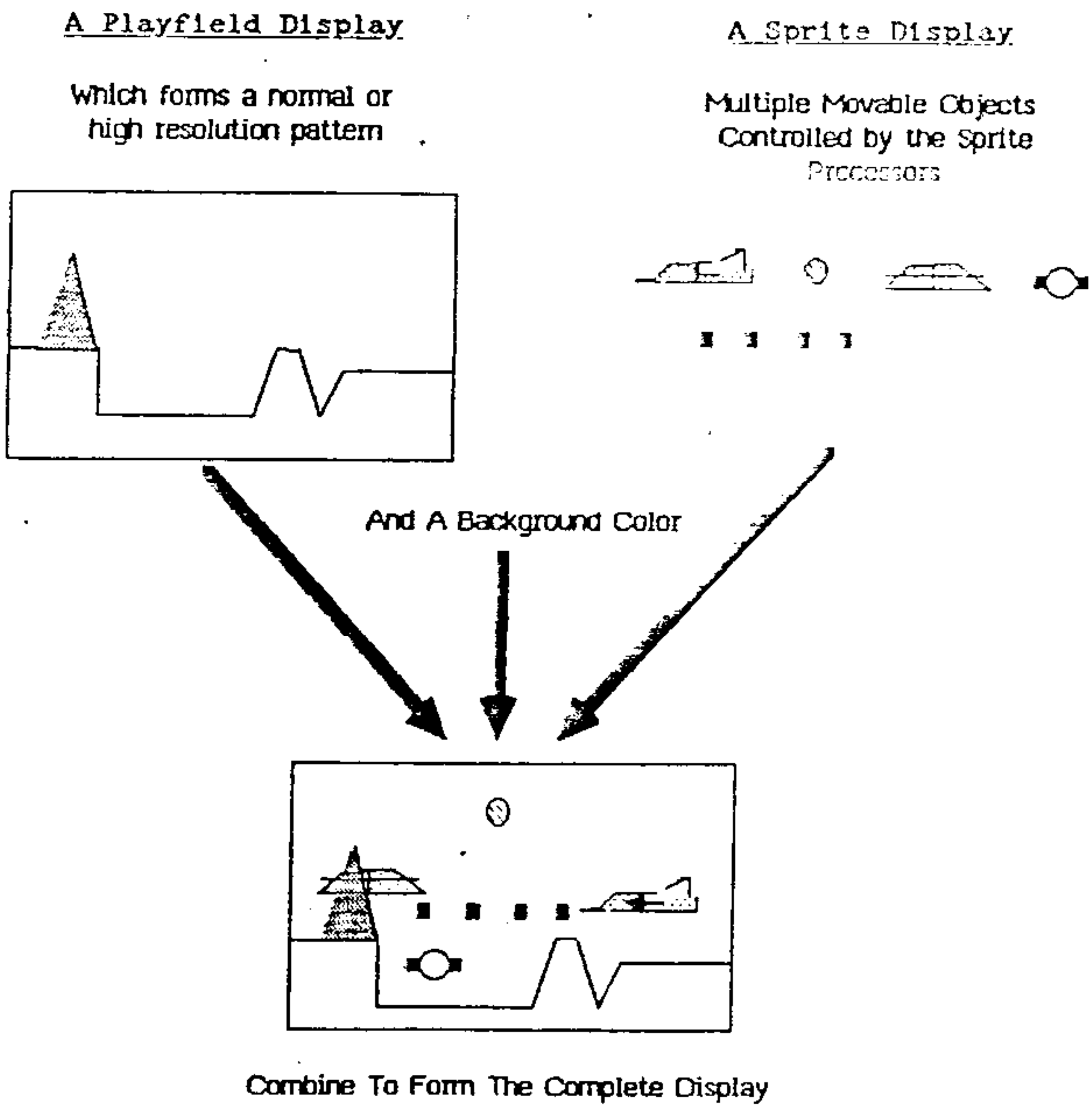
Sprite-Processors(8)

- Resolution normal
- Reusable vertically
- Data block defines position & shape
- 4-color choices per pixel
- 16-color choices per pixel if "attached" & overlapped

Lorraine Computer Functional Block Diagram



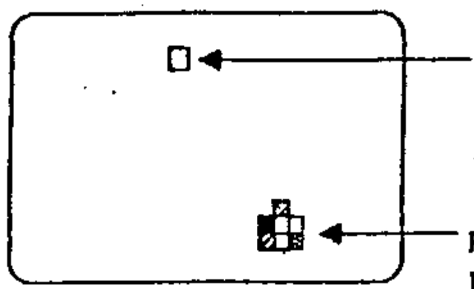
The Computer Screen Shows:



Each of the elements shown here is separately described

in the series of drawings which follow.

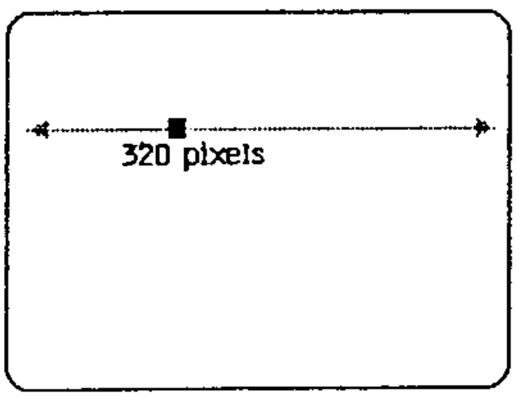
What Is A Pixel?



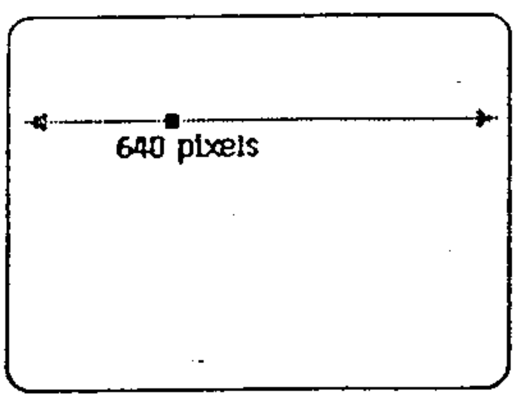
A television picture is composed of many elements. Each element is called a pixel.

Pixels are used together to build larger graphic objects.

Pixels have different colors, and require a code to identify the pixel color. Each bit of this code is contained in a separate section of memory called a **bit plane.**

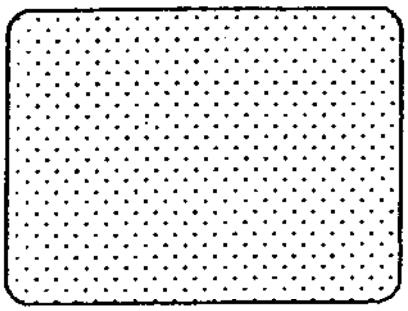


In normal resolution mode, 320 pixels will fill a horizontal line.



In high resolution mode, 640 pixels fill a horizontal line.

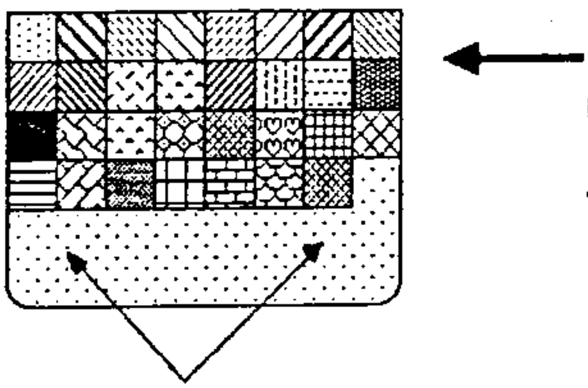
The Playfield Display



Can be simply a single color (any one of 4096 possible choices) as a background.

OR

May be organized as one "playfield" (treated as a foreground, against the background color),

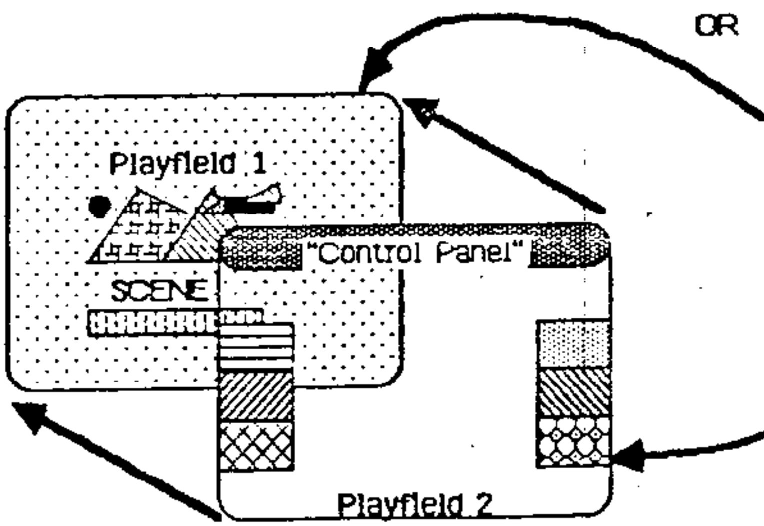


In this mode, this single playfield may contain up to 31 distinct colors.

"Color 0" is "background".

(color 0 here, in this playfield)

OR



May be organized as two playfields, one with 7 distinct colors plus background,

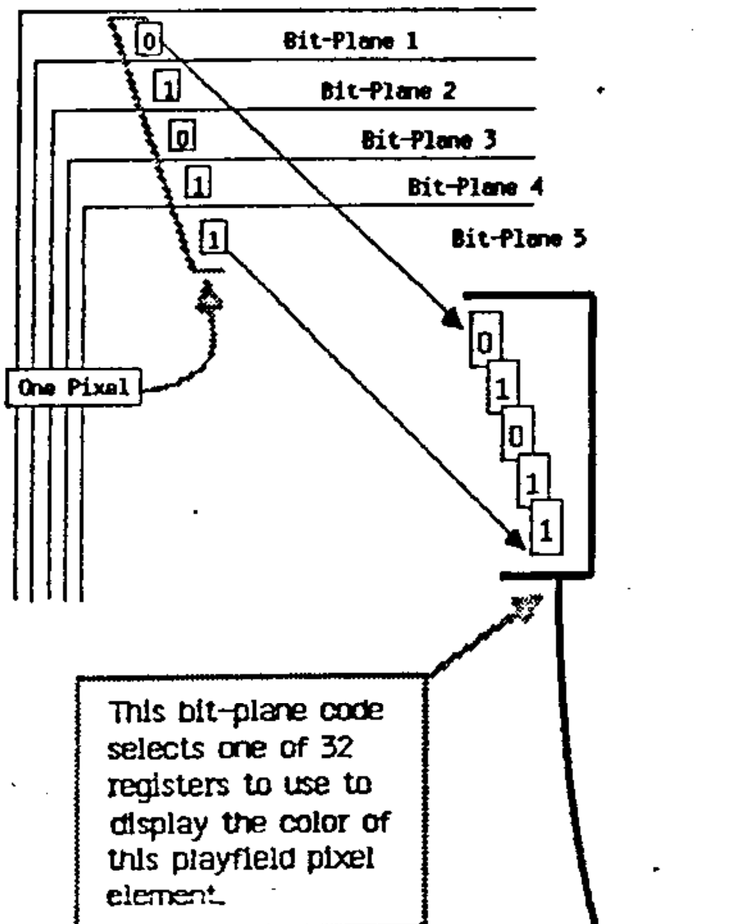
the other with 7 distinct colors plus transparent, overlaying the first one.

In this two-playfield mode, color 0 is treated as "transparent". this permits viewing of objects having a lower video priority, "behind" this playfield, or, if no objects are present, displays the background color.

Playfield 1 may appear "behind" or "in front of" playfield 2.

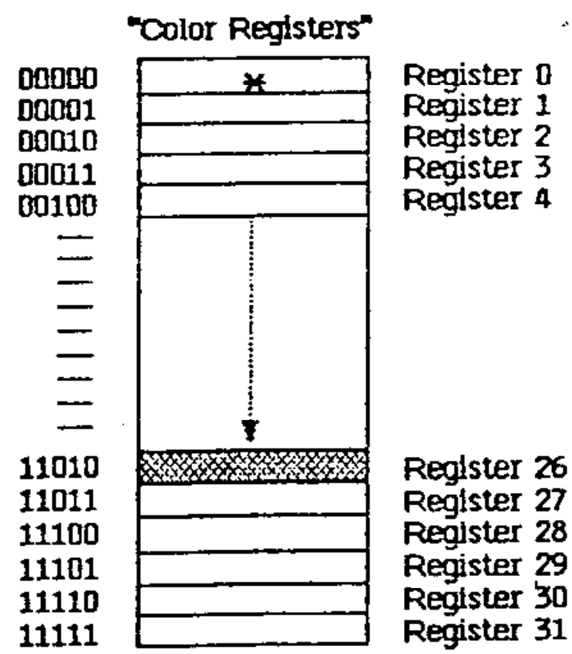
Selecting Color Of Playfield Pixels

The drawing titled "What Is A Pixel" introduced the concept of producing colors on the screen. The illustrations which follow show exactly how the computer forms this combination and selects the colors to display, using sections of memory called bit-planes.



Even though each bit plane is a a separate section of memory, they are used by the display as though they were stacked, one on top of another.

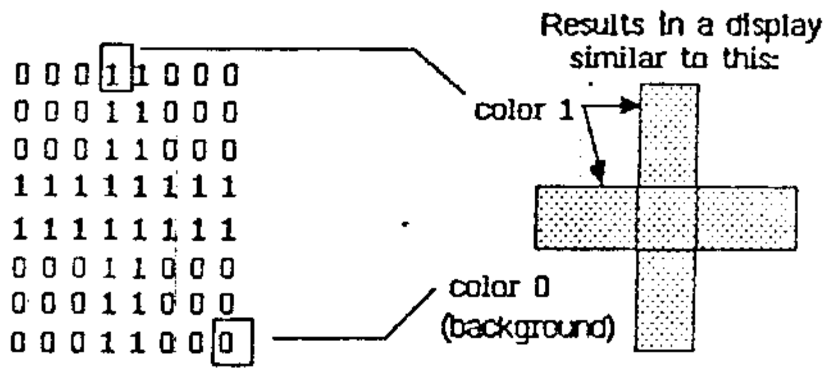
The bits in corresponding positions in each bit-plane are combined by the display hardware to form a code which is the color code for the pixel.



* Color Register 0 always contains the definition of the background color for the display.

How is a Bit-Plane Used?

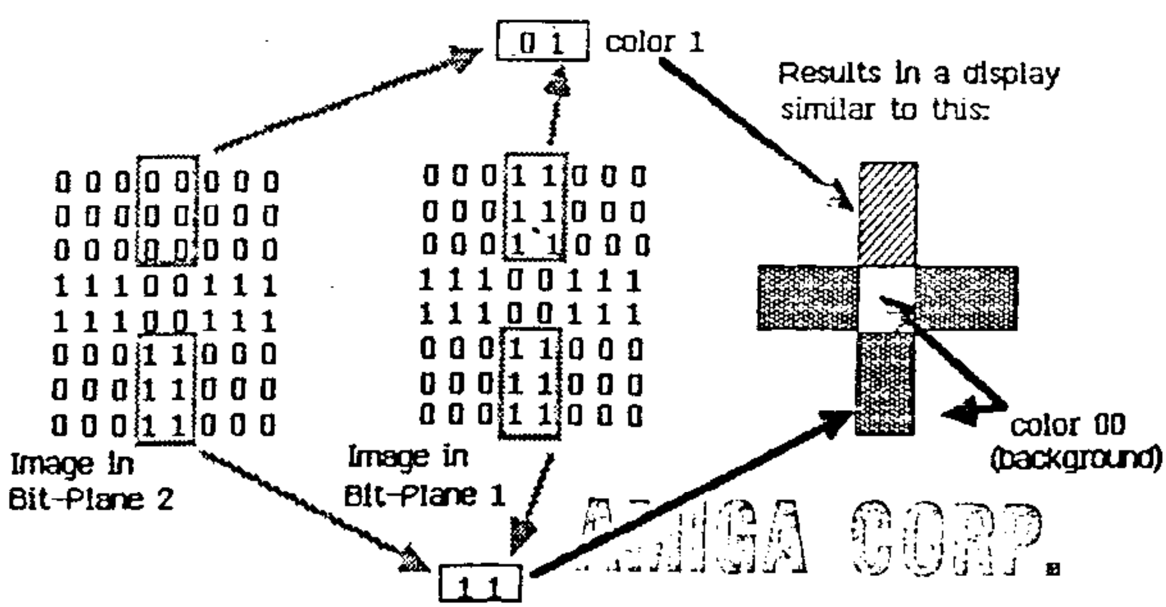
A bit-plane is a collection of 1's and 0's in the memory system. To draw a picture in a bit plane, a user will either put 1's against a background of 0's or 0's against a background of 1's. Here is an example of a "plus" sign, drawn in a bit-plane:



If a playfield is composed of just one bit-plane, this symbol will appear onscreen as a plus sign whose color is taken from color register 1, against a background whose color is taken from color register 0.

If a playfield is composed of two bit-planes, each bit-plane can contain some form of image of the shape to be produced. The combination of bits in the corresponding positions within each bit plane forms a number at each position. This number selects the color register from which the color of the pixel element should be taken. The example below shows how the plus sign may be drawn in three different colors (plus background).

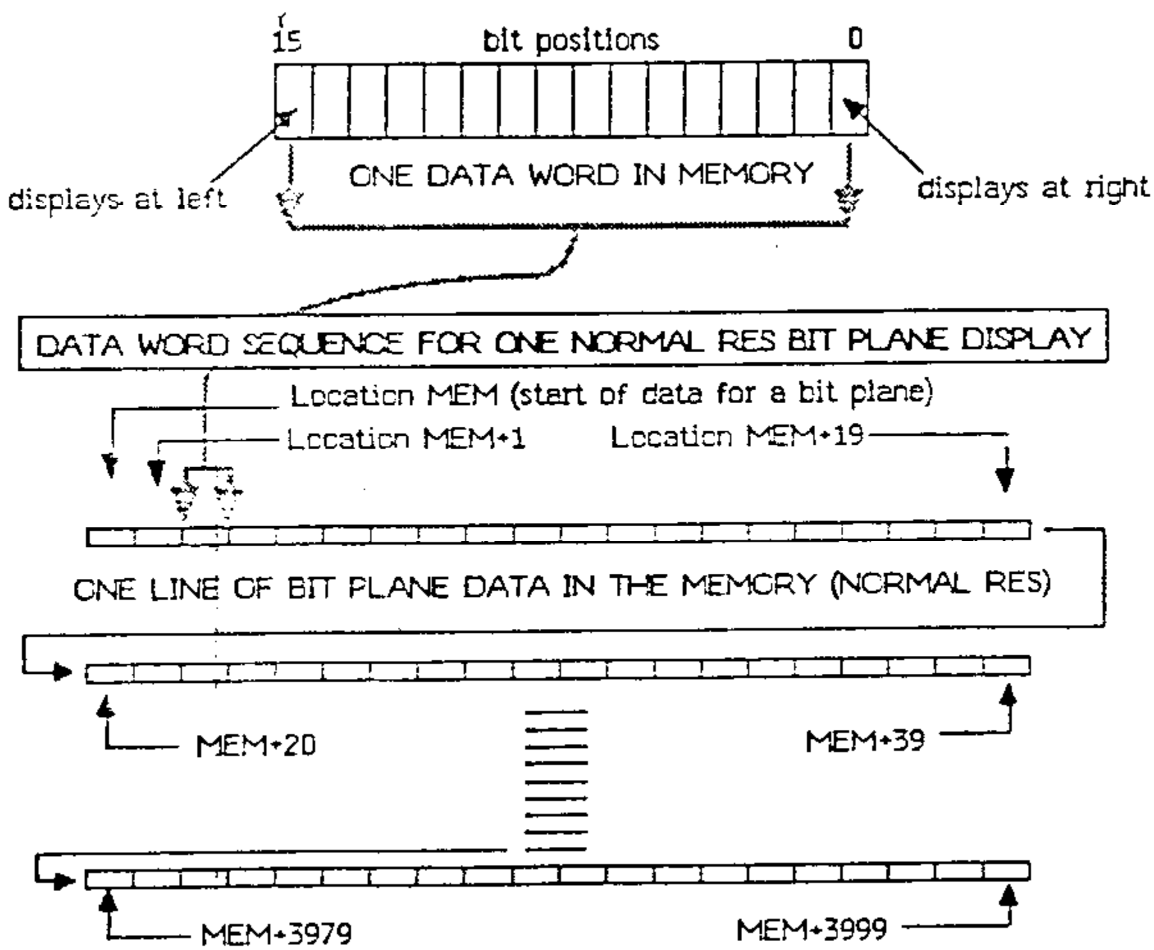
Note: The highest numbered bit-plane contributes the most significant bit of the color register selection number.



How Is Bit-Plane Data Stored And Displayed?

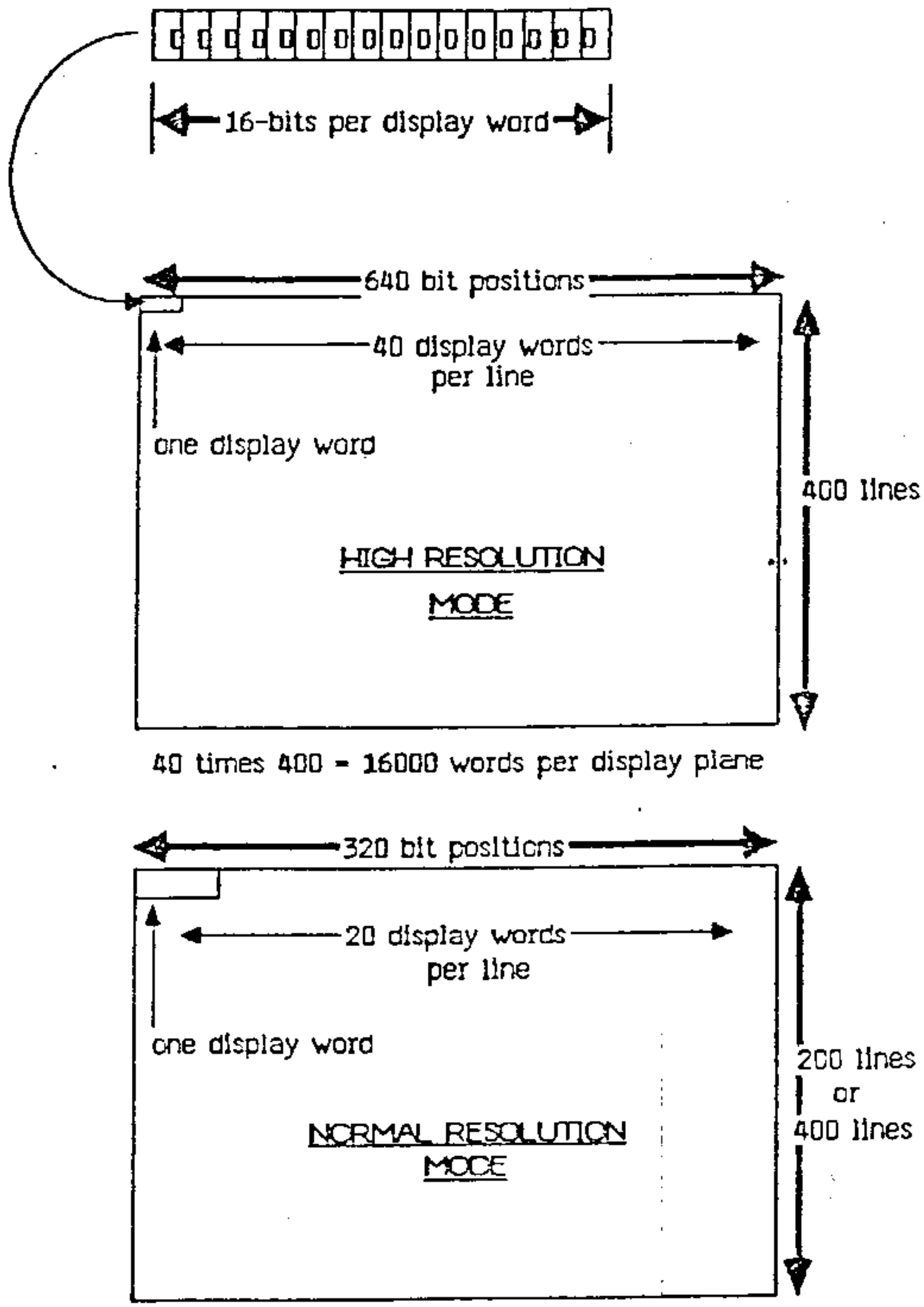
Each memory word in the bit plane display memory area holds 16 bits of the bit-plane data. Each word stores part of a horizontal line. The most significant bit of each word shows up onscreen as the left-most display bit within that word, and the least significant bit as the rightmost bit in the word.

All data for each display line of a bit-plane is stored at sequentially increasing memory locations. In other words, if the leftmost word of a line is at location "MEM", then the memory word at location MEM+1 contains the bits which define the next word to the right within this bit plane.



If each bit plane is to be exactly as wide as the screen display, then the data will be stored exactly as shown.

Memory Requirements For Each Bit Plane



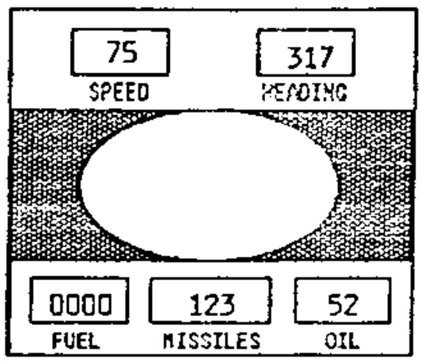
20 times 200 = 4000 words per display plane
or
20 times 400 = 8000 words per display plane

DUAL PLAYFIELDS

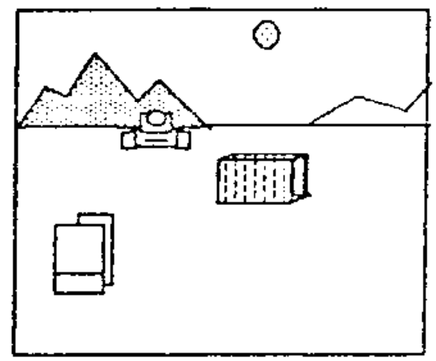
This illustration also shows how 2 Playfields (shown initially in the drawing titled "Display") can be combined to form a dual playfield screen output.

Note that this drawing shows only the playfields. The moveable objects, called "sprites" are separately introduced in the drawing titled: "What is a Sprite?"

Playfield 1 (1,2,or 3 bit-planes)*

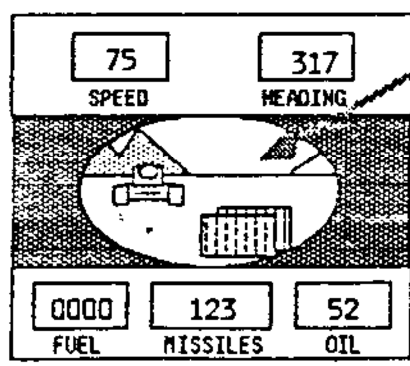


Playfield 2 (1,2,or 3 bit-planes)*



Both Playfields Appear Onscreen, combined to form the complete display

In this case, Playfield 1 has a higher priority than Playfield 2 and will therefore hide some of playfield 2's details.



The background color shows through where there are transparent sections of both playfields.

Most important, only one Playfield (2 in this case) need be manipulated to change the view through the window. This saves a lot of time if Playfield 1 is complex.

* See the drawing titled "What is a Pixel?"

Dual Playfield Bit-Plane Grouping

If Dual Playfield Mode is selected:

Then the number of bit planes assigned to each playfield depends on how many bit-planes are selected for display. The bit-planes are assigned alternately to each playfield as shown in the drawing below:

Bit-Planes "turned-on"	Playfield 1*	Playfield 2*
0	none	none
1	1	
2	1	2
3	1, 3	2
4	1, 3	2, 4
5	1, 3, 5	2, 4
6	1, 3, 5	2, 4, 6

* Note: Either playfield may be placed "in front of" or "behind" the other using the "swap-bit".

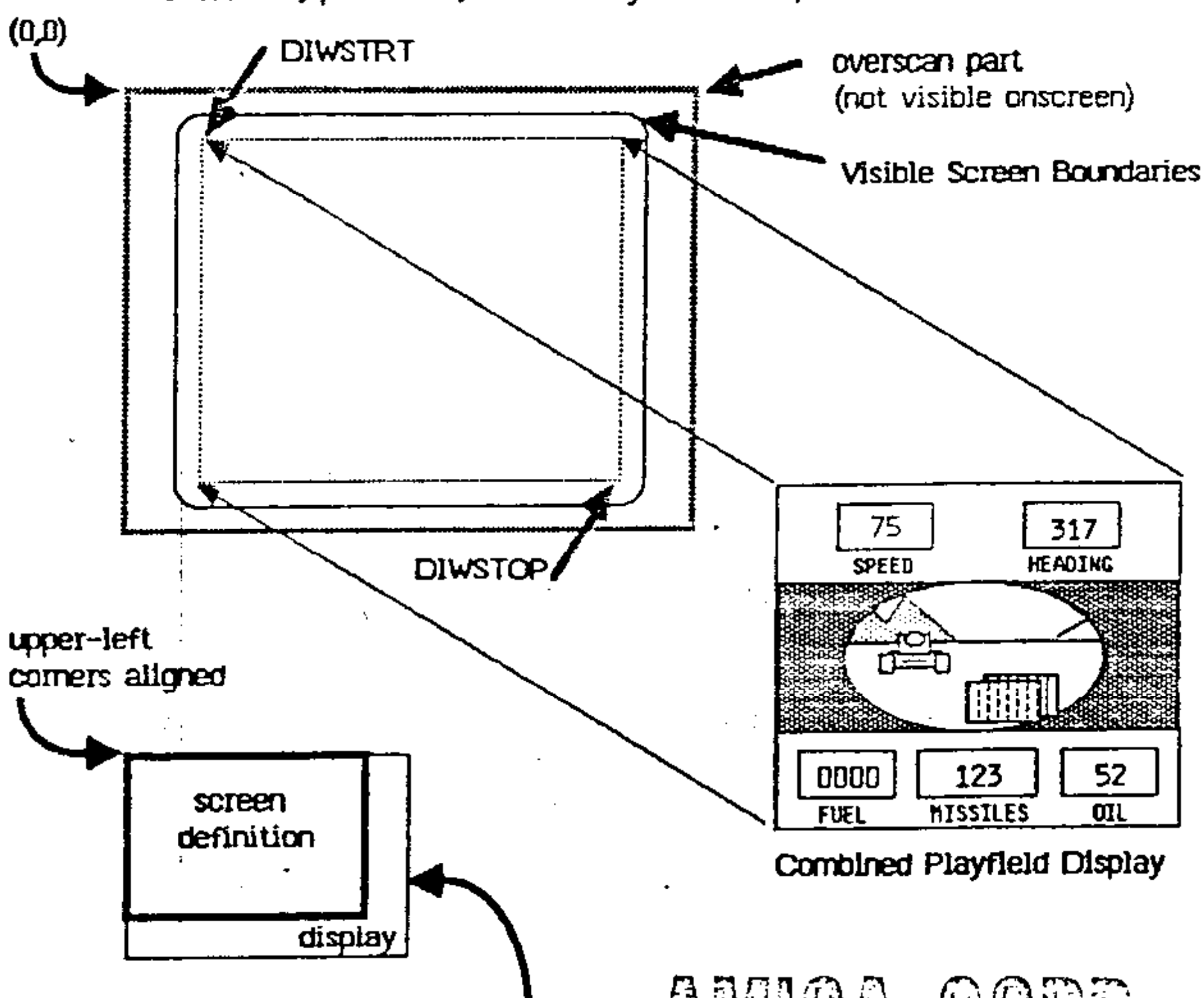
Controlling Position Of The Playfield Display

Playfield A and Playfield B (if selected), will combine, as shown in the diagram titled "Dual Playfields", to form a complete picture. The user can control where, on the screen, this picture appears. The onscreen location is controlled by specifying Display window start and stop. The abbreviations for these are DIWSTRT and DIWSTOP.

DIWSTRT specifies the position onscreen of the starting point of the upper left-hand corner of the display.

DIWSTOP specifies the position of the lower right-hand corner of the display.

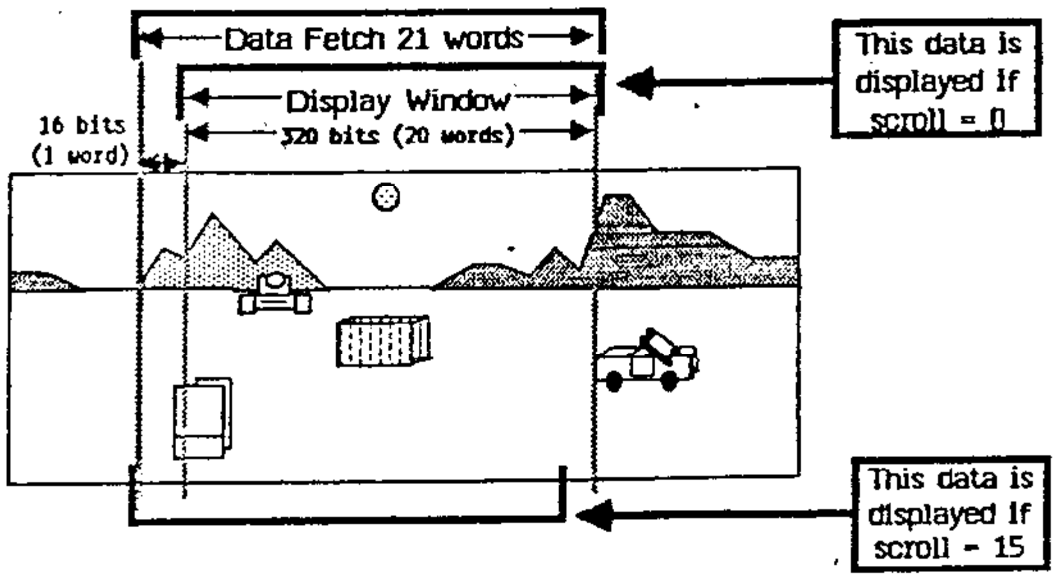
Each contain X and Y coordinates, with X=0, Y=0 as the upper left hand corner of the possible display area. X increases to the right. Y increases as it proceeds top to bottom on screen. (Due to display "overscan", position 0,0 is actually offscreen.)



If the playfield display is larger than that defined by the difference between DIWSTOP and DIWSTRT, then not all of it will appear on the screen.

Playfield Horizontal Scrolling

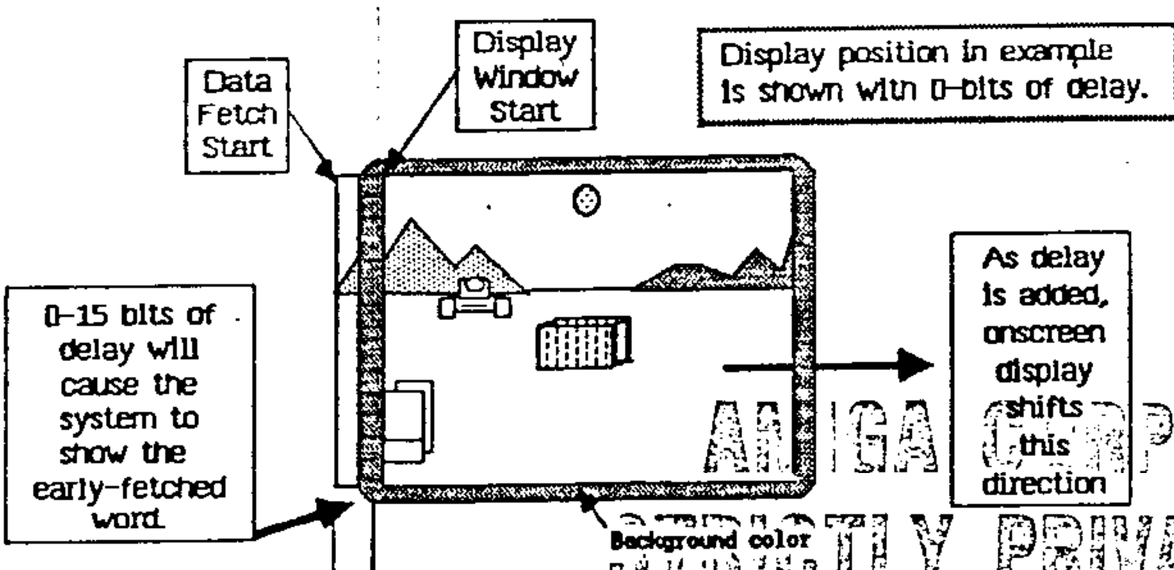
If the designer produces a scene in memory which is wider than can be displayed within a single horizontal screen width, it will be easy to smoothly scroll the scene horizontally through the available display window.



Horizontal scrolling is performed by **DELAYING** the display of the data which has been fetched for the display. This means that for each bit of delay which is specified, the onscreen data will appear to shift one bit position to the right.

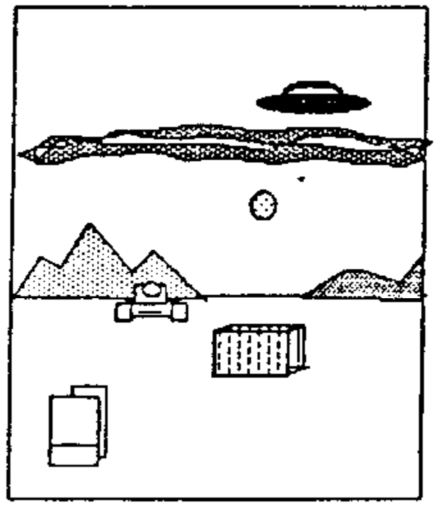
As this "current screen" shifts to the right, one bit at a time, there must be some data provided at the left border which, in turn, becomes visible one bit at a time as each bit at the rightmost edge disappears off the screen.

Therefore to allow horizontal scrolling (i.e. shifting the data), the user must fetch one extra word per line. This word provides the bits which become visible as the onscreen data shifts.

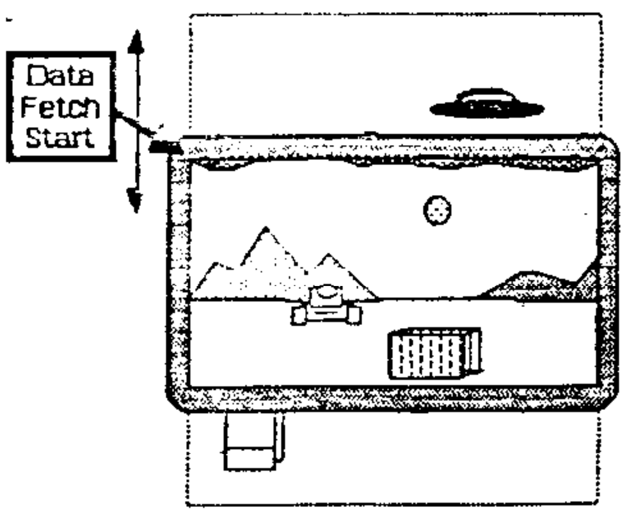


Playfield Vertical Scrolling

- If the designer produces a scene in memory which is higher than can be displayed within a single vertical screen height, it will be easy to smoothly scroll the scene vertically through the display window.



- All that must be done is to progressively increase or decrease the starting point of the data fetch start for the display. This will have the effect of scrolling the data within the display window so that the low part or a higher part of the picture are shown on the display.



As the value of Data Fetch increases, more of the low part of the picture is shown. As it decreases, more of the upper part is shown.

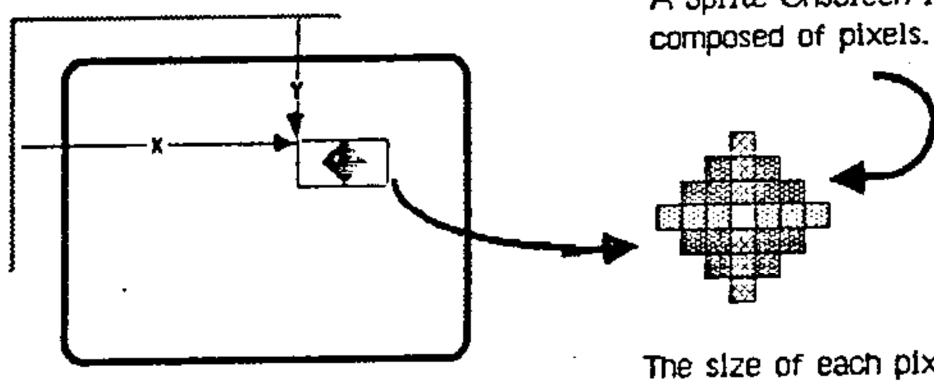
- Since there are 200 lines vertically, each step can be as little as 1/200th of the vertical screen size (or 1/400th if in interlace mode). This allows for a very smooth vertical scroll to be performed.

What Is A Sprite?

A Sprite is a movable graphic object which is independent of the playfield bit-planes.

Each Sprite's position (vert-horiz) and pattern (shape-color) are defined in memory by the programmer.

This data is then automatically fetched and placed on the screen, at the correct position by the Sprite Display Hardware.

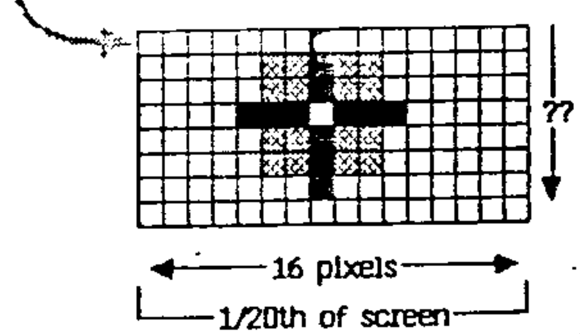


A Sprite Onscreen is composed of pixels.

The size of each pixel is the same as the size of a bit-plane pixel in the 320 mode. (1/320th of the screen.)

The position of a sprite onscreen is defined by the x-y coordinate of the upper left hand pixel defined in the sprite data area.

The height of each sprite is defined by the user.



Each sprite is 16 pixels wide.

There are 8 sprite processors available in this system, each of which can be used more than once on a display screen if desired.