Hardware

COLLABORATORS			
	TITLE : Hardware		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 14, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Hardware

1.1 Amiga® Hardware Reference Manual: 3 Playfield Hardware

The screen display of the Amiga consists of two basic parts -- ↩ playfields,
which are sometimes called backgrounds, and sprites, which are easily movable graphics objects. This chapter describes how to directly access hardware registers to form playfields. The chapter begins with a brief overview of playfield features and covers the following major topics:

* Forming a single "basic" playfield, which is a playfield the same size as the display screen. This section includes concepts that are fundamental to forming any playfield.
* Forming a dual-playfield display in which one playfield is superimposed upon another. This procedure differs from that of forming a basic playfield in some details.
* Forming playfields of various sizes and displaying only part of a larger playfield.
* Moving playfields by scrolling them vertically and horizontally.
* Advanced topics to help you use playfields in special situations.

For information about movable sprite objects , see Chapter 4, Sprite Hardware. There are also movable playfield objects, which are subsections of a playfield. To move portions of a playfield, you use a technique called playfield animation , which is described in Chapter 6, Blitter Hardware.

For information relating to the playfield hardware in the Enhanced Chip Set (ECS), such as SuperHires Mode , programmable scan rates and synchronization , see Appendix C.

Advanced Topics

Summary of Playfield Registers

Summary of Color Selection Registers

1.2 3 Playfield Hardware / About Amiga Playfields

A playfield forms the basic foundation of an Amiga display and determines its fundamental characteristics. To form a playfield, you program the hardware registers of the custom chips with the basic parameters of the type of display you want. Forming a playfield involves selecting the number of colors, setting up a color table and bitplanes, and selecting the resolution and display mode.

To understand how Amiga playfields work, it will be helpful to review how the Amiga's video displays are produced.

How the Amiga's Video Display is Produced

1.3 3 / About Amiga Playfields / How Amiga's Video Display is Produced

The Amiga produces its video displays with raster display techniques. The picture you see on the screen is made up of a series of horizontal video lines displayed one after the other. Each horizontal video line is made up of a series of pixels. You create a graphic display by defining one or more bitplanes in memory and filling them with "1"s and "0"s. The combination of the "1"s and "0"s will determine the colors in your display.

```
        _________________________
  |  |                         |
  |  | ->->->->->->->->->->->- |
  |  | ----------------------- |  Each line represents one sweep of an
  |  | ----------------------- |  electron beam which is "painting" the
  |  |         -------         |  picture as it goes along.
  |  |         -------         |
  |  |      Video Picture      |  The video beam produces each line by
  |  |         -------         |  sweeping from left to right.  It
  |  |         -------         |  produces the full screen by sweeping
  |  | ----------------------- |  the beam from the top to the bottom,
  |  | ----------------------- |  one line at a time.
  |  | ----------------------- |
 \|/ |_________________________|
```

Figure 3-1: How the Video Display Picture Is Produced

The video beam produces about 262 video lines from top to bottom, of which 200 normally are visible on the screen with an NTSC system. With a PAL system, the beam produces 312 lines, of which 256 are normally visible. Each complete set of lines (262/NTSC or 312/PAL) is called a display field. The field time, i.e. the time required for a complete display field to be produced, is approximately 1/60th of a second for an NTSC system and approximately 1/50th of a second for PAL. Between display fields, the video beam traverses the lines that are not visible on the screen and returns to the top of the screen to produce another display field.

The display area is defined as a grid of pixels. A pixel is a single picture element, the smallest addressable part of a screen display. The drawings below show what a pixel is and how pixels form displays.

```
        _________________________
       |                         |
       |                         |
       |    <- - - - - - - - - - |- The picture is formed from many
       |                         |  elements.  Each element is called
       |                         |  a pixel.
       |                         |
       |                         |
       |                         |
       |                        |
       |              <- - - -|- Pixels are used together to build
       |                        |  larger graphic objects.
       |                         |
       |_________________________|

        _________________________     _________________________
       |                         |   |                         |
       |      _                  |   |                         |
       |<- - |_| - - - - - - - ->|   |<- - -  - - - - - - - ->|
       |                         |   |                         |
       |  320 pixels             |   |   640 pixels            |
       |                         |   |                         |
       |                         |   |                         |
       |                         |   |                         |
       |                         |   |                         |
       |                         |   |                         |
       |                         |   |                         |
       |                         |   |                         |
       |_________________________|   |_________________________|

     In normal resolution mode, 320   In high resolution mode, 640
     pixels fill a horizontal line.   pixels fill a horizontal line.
```

Figure 3-2: What Is a Pixel?

The Amiga offers a choice in both horizontal and vertical resolutions. Horizontal resolution can be adjusted to operate in low resolution or high resolution mode. Vertical resolution can be adjusted to operate in interlaced or non-interlaced mode.

* In low resolution mode, the normal playfield has a width of 320 pixels.

* High resolution mode gives finer horizontal resolution -- 640 pixels in the same physical display area.

* In non-interlaced mode, the normal NTSC playfield has a height of 200 video lines. The normal PAL screen has a height of 256 video lines.

* Interlaced mode gives finer vertical resolution -- lines in the same physical display area in NTSC and 512 for PAL.

These modes can be combined, so you can have, for instance, an interlaced, high resolution display.

Note that the dimensions referred to as "normal" in the previous paragraph are nominal dimensions and represent the normal values you should expect to use. Actually, you can display larger playfields; the

maximum dimensions
are given in the section called

Bitplanes and Playfields of All Sizes
. Also, the dimensions of the playfield in memory are often larger than the playfield displayed on the screen. You choose which part of this larger memory picture to display by specifying a different size for the display window.

A playfield taller than the screen can be scrolled, or moved smoothly, up or down. A playfield wider than the screen can be scrolled horizontally, from left to right or right to left. Scrolling is described in the section called

Moving (Scrolling) Playfields
.

In the Amiga graphics system, you can have up to thirty-two different colors in a single playfield, using normal display methods. You can control the color of each individual pixel in the playfield display by setting the bit or bits that control each pixel. A display formed in this way is called a bitmapped display.

For instance, in a two-color display, the color of each pixel is determined by whether a single bit is on or off. If the bit is 0, the pixel is one user-defined color; if the bit is 1, the pixel is another color. For a four-color display, you build two bitplanes in memory. When the playfield is displayed, the two bitplanes are overlapped, which means that each pixel is now two bits deep. You can combine up to five bitplanes in this way. Displays made up of three, four, or five bitplanes allow a choice of eight, sixteen, or thirty-two colors, respectively.

The color of a pixel is always determined by the binary combination of the bits that define it. When the system combines bitplanes for display, the combination of bits formed for each pixel corresponds to the number of a color register. This method of coloring pixels is called color indirection. The Amiga has thirty-two color registers, each containing bits defining a user-selected color (from a total of 4,096 possible colors).

Figure 3-3 shows how the combination of up to five bitplanes forms a code that selects which one of the thirty-two registers to use to display the color of a playfield pixel.

```
     _______________
    |
    |           bit-plane 5
    | _\_____________
    ||  \
    ||  \        bit-plane 4
     | _\_\_____________
     ||  \ \
     ||  \ \      bit-plane 3
      | _\_\_\___________                   Bits from
      ||  \ \ \                                 planes     Color
      ||  \ \ \    bit-plane 2             5,4,3,2,1  Registers
       | _\_\_\_\__________
       ||  \ \ \ \                                     __________
       ||  \ \ \ \  bit-plane 1              00000   |_________|
        | ^\ \ \ \ \                          00001   |_________|
        | | \ \ \ \ \                         00010   |_________|
        | |  \ \ \ \ \                        00011   |_________|
          |   \ \ \ \ \  ___  _ _ _ _ _       00100   |_________|
          |    \ \ \ \ \|   |          |        -     |         |
         One    \ \ \ \ | 1 |                   -     |    |    |
        Pixel    \ \ \ \|___|          |        -     |         |
                  \ \ \ \|   |                  -     |    |    |
                   \ \ \ | 1 |         |        -     |         |
                    \ \ \|___|                  -     |   \|/   |
                     \ \ \|   |        |    \   -     |_________|
                      \ \ | 0 |         - - - 11000   |#########|
                       \ \|___|        |    / 11001   |_________|
                        \ \|   |              11010   |_________|
                         \ | 0 |       |      11011   |_________|
                          \|___|              11100   |_________|
                           \|   |      |      11101   |_________|
                            | 0 |             11110   |_________|
                            |___| _ _ _|      11111   |_________|
```

Figure 3-3: How Bitplanes Select a Color

Values in the highest numbered bitplane have the highest significance in the binary number. As shown in Figure 3-4, the value in each pixel in the highest-numbered bitplane forms the leftmost digit of the number. The value in the next highest-numbered bitplane forms the next bit, and so on.

```
     SAMPLE DATA FOR
       FOUR PIXELS

      1   1   0   0          Data in bitplane 5 -- most significant
      1   0   1   0          Data in bitplane 4
      1   0   0   1          Data in bitplane 3
```

```
0   1   1   1          Data in bitplane 2
0   0   1   0          Data in bitplane 1 -- least significant

|   |   |   |
|   |   |   |
|   |   |   |_________________ Value 6 -- COLOR6
|   |   |_____________________ Value 11 -- COLOR11
|   |_________________________ Value 18 -- COLOR18
|_____________________________ Value 28 -- COLOR28
```

Figure 3-4: Significance of Bitplane Data in Selecting Colors

You also have the choice of defining two separate playfields, each formed from up to three bitplanes. Each of the two playfields uses a separate set of eight different colors. This is called
dual-playfield mode
.

1.4 3 Playfield Hardware / Forming a Basic Playfield

To get you started, this section describes how to directly access hardware
registers to form a single basic playfield that is the same size as the video screen. Here, "same size" means that the playfield is the same size as the actual display window. This will leave a small border between the playfield and the edge of the video screen. The playfield usually does not extend all the way to the edge of the physical display.

To form a playfield, you need to define these characteristics:

* Height and width of the playfield and size of the display window (that is, how much of the playfield actually appears on the screen).
* Color of each pixel in the playfield.
* Horizontal resolution.
* Vertical resolution, or interlacing.
* Data fetch and modulo, which tell the system how much data to put on a horizontal line and how to fetch data from memory to the screen.

In addition, you need to allocate memory to store the playfield, set pointers to tell the system where to find the data in memory, and (optionally) write a Copper routine to handle redisplay of the playfield.

Height and Width of the Playfield

Bitplanes and Color

Selecting Horizontal and Vertical Resolution

Allocating Memory for Bitplanes

Coding the Bitplanes for Correct Coloring

1.5 3 / Forming a Basic Playfield / Height and Width of the Playfield

To create a playfield that is the same size as the screen, you can use a width of either 320 pixels or 640 pixels, depending upon the resolution you choose. The height is either 200 or 400 lines for NTSC, 256 or 512 lines for PAL, depending upon whether or not you choose interlaced mode.

1.6 3 / Forming a Basic Playfield / Bitplanes and Color

You define playfield color by:

1. Deciding how many colors you need and how you want to color each pixel.
2. Loading the colors into the color registers.
3. Allocating memory for the number of bitplanes you need and setting a pointer to each bitplane.
4. Writing instructions to place a value in each bit in the bitplanes to give you the correct color.

Table 3-1 shows how many bitplanes to use for the color selection you need.

Number of Colors	Number of Bitplanes
1 - 2	1
3 - 4	2
5 - 8	3
9 - 16	4
17 - 32	5

Table 3-1: Colors in a Single Playfield

Selecting the Number of Bitplanes

1.7 3 / / Bitplanes and Color / The Color Table

The color table contains 32 registers, and you may load a different color
into each of the registers. Here is a condensed view of the contents of
the color table:

Register Name	Contents	Meaning
COLOR00	12 bits	User-defined color for the background area and borders.
COLOR01	12 bits	User-defined color number 1 (For example, the alternate color selection for a two-color playfield).
COLOR02	12 bits	User-defined color number 2.
.	.	
.	.	
.	.	
COLOR31	12 bits	User-defined color number 31.

Table 3-2: Portion of the Color Table

COLOR00 is always reserved for the background color. The background color
shows in any area on the display where there is no other object present
and is also displayed outside the defined display window, in the border
area.

Genlocks and the background color.

If you are using the optional genlock board for video input from a
camera, VCR, or laser disk, the background color will be replaced by
the incoming video display.

Twelve bits of color selection allow you to define, for each of the 32
registers, one of 4,096 possible colors, as shown in Table 3-3.

Bits	Color

```
                  Bits 15 - 12    Unused
                  Bits 11 - 8     Red
                  Bits 7 - 4      Green
                  Bits 3 - 0      Blue
```

Table 3-3: Contents of the Color Registers

Table 3-4 shows some sample color register bit assignments and the resulting colors. At the end of the chapter is a more extensive

Color Register list
.

Contents of the Color Register	Resulting Color
$FFF	White
$6FE	Sky blue
$DB9	Tan
$000	Black

Table 3-4: Sample Color Register Contents

Some sample instructions for loading the color registers are shown below:

```
LEA     CUSTOM,a0               ; Get base address of custom hardware...
MOVE.W  #$FFF,COLOR00(a0)       ; Load white into color register 0
MOVE.W  #$6FE,COLOR01(a0)       ; Load sky blue into color register 1
```

The color registers are write-only.

Only by looking at the screen can you find out the contents of each color register. As a standard practice, then, for these and certain other write-only registers, you may wish to keep a "back-up" or "shadow" copy in RAM. As you write to the color register itself, you should update this RAM copy. If you do so, you will always know the value each register contains.

1.8 3 / / Bitplanes and Color / Selecting the Number of Bitplanes

After deciding how many colors you want and how many bitplanes are required to give you those colors, you tell the system how many bitplanes to use.

You select the number of bitplanes by writing the number into the register

BPLCON0
(for Bitplane Control Register 0) The relevant bits are bits 14,

13, and 12, named BPU2, BPU1, and BPU0 (for "Bitplanes Used"). Table 3-5 shows the values to write to these bits and how the system assigns bitplane numbers.

Table 3-5: Setting the Number of Bitplanes

Value	Number of Bitplanes	Name(s) of Bitplanes
000	None *	
001	1	PLANE 1
010	2	PLANES 1 and 2
011	3	PLANES 1 - 3
100	4	PLANES 1 - 4
101	5	PLANES 1 - 5
110	6	PLANES 1 - 6 **
111		Value not used.

* Shows only a background color; no playfield is visible.

** Sixth bitplane is used only in dual-playfield mode and in

hold-and-modify mode (described in the section called Advanced Topics.

About the BPLCON0 register.

The bits in the BPLCON0 register cannot be set independently. To set any one bit, you must reload them all.

The following example shows how to tell the system to use two low resolution bitplanes.

```
        MOVE.W  #$2200,
                BPLCON0
                +CUSTOM   ; Write to it
```

Because register BPLCON0 is used for setting other characteristics of the display and the bits are not independently settable, the example above also sets other parameters (all of these parameters are described later in the chapter).

*
Hold-and-modify mode is turned off.

*
Single-playfield mode is set.

*
Composite video color is enabled. (Not applicable in all models .)

*
Genlock audio is disabled.
* Light pen is disabled.

*
Interlaced mode is disabled.

*
External resynchronization is disabled. (genlock)

1.9 3 / Basic Playfield / Selecting Horizontal and Vertical Resolution

Standard home television screens are best suited for low resolution displays. Low resolution mode provides 320 pixels for each horizontal line. High resolution monochrome and RGB monitors can produce displays in high resolution mode, which provides 640 pixels for each horizontal line. If you define an object in low resolution mode and then display it in high resolution mode, the object will be only half as wide.

To set horizontal resolution mode, you write to bit 15, HIRES, in register

BPLCON0
:

High resolution mode -- write 1 to bit 15.
Low resolution mode -- write 0 to bit 15.

Note that in high resolution mode, you can have up to four bitplanes in the playfield and, therefore, up to 16 colors.

Interlaced mode allows twice as much data to be displayed in the same vertical area as in non-interlaced mode. This is accomplished by doubling the number of lines appearing on the video screen. The following table shows the number of lines required to fill a normal, non-overscan screen.

	NTSC	PAL
Non-interlaced	200	256
Interlaced	400	512

Table 3-6: Lines in a Normal Playfield

In interlaced mode, the scanning circuitry vertically offsets the start of every other field by half a scan line.

```
           _____________________
  line 1 _|_____________________|_
          |_____________________| \
          |      _________      |  \
          |      _________      |   \
          |                     |    \
          |      Odd Field      |     \   _____________________
          |      _________      |      \_|_____________________|_ Line 1
          |      _________      |       _|_|___________________|_ Line 2
          |_____________________|      / |                     |
          |_____________________|     /  | |                   |
          |_____________________|    /   |    Video Display    |
                                    /    | | (400 lines NTSC   |
           _____________________   /     |    512 lines PAL)   |
  line 2 _|_____________________|_/      | |                   |
          |_____________________|        |                     |
          |      _________      |        |\|/                  |
          |      _________      |        |_____________________|
          |                     |
          |      Even Field     |      Same physical space as used
          |      _________      |       by a 200-line (256) PAL),
          |      _________      |         noninterlaced display.
          |_____________________|
          |_____________________|
          |_____________________|
```

Figure 3-5: Interlacing

Even though interlaced mode requires a modest amount of extra work in setting registers (as you will see later on in this section), it provides fine tuning that is needed for certain graphics effects. Consider the diagonal line in Figure 3-6 as it appears in non-interlaced and interlaced modes. Interlacing eliminates much of the jaggedness or "staircasing" in the edges of the line.

```
              ___________________      ___________________
             |#|#| | | | | | | | |      |#|#|_|_|_|_|_|_|_|_|
             |#|#|_|_|_|_|_|_|_|_|      |_|#|#|_|_|_|_|_|_|_|
             | | |#|#| | | | | | |      |_|_|#|#|_|_|_|_|_|_|
             |_|_|#|#|_|_|_|_|_|_|      |_|_|_|#|#|_|_|_|_|_|
             | | | | |#|#| | | | |      |_|_|_|_|#|#|_|_|_|_|
             |_|_|_|_|#|#|_|_|_|_|      |_|_|_|_|_|#|#|_|_|_|
             | | | | | | |#|#| | |      |_|_|_|_|_|_|#|#|_|_|
             |_|_|_|_|_|_|#|#|_|_|      |_|_|_|_|_|_|_|#|#|_|
             | | | | | | | | |#|#|      |_|_|_|_|_|_|_|_|#|#|
             |_|_|_|_|_|_|_|_|#|#|      |_|_|_|_|_|_|_|_|_|#|

                non-interlaced               interlaced
```

Figure 3-6: Effect of Interlaced Mode on Edges of Object

When you use the special blitter DMA channel to draw lines or polygons onto an interlaced playfield, the playfield is treated as one display, rather than as odd and even fields. Therefore, you still get the smoother edges provided by interlacing.

To set interlaced or non-interlaced mode, you write to bit 2, LACE, in register
BPLCON0
:

Interlaced mode -- write 1 to bit 2.
Non-interlaced mode -- write 0 to bit 2.

As explained above in
Setting the Number of Bitplanes
, bits in

BPLCON0
are not independently settable.

The following example shows how to specify high resolution and interlaced modes.

```
        MOVE.W  #$A204,
                BPLCON0
                +CUSTOM   ; Write to it
```

The example above also sets the following parameters that are also controlled through register
BPLCON0
:

*
High resolution mode is enabled.

*
Two bitplanes are used.

*
Hold-and-modify mode is disabled.

*
Single-playfield mode is enabled.

*
Composite video color is enabled.

*
Genlock audio is disabled.
*   Light pen is disabled.

*
Interlaced mode is enabled.

*
External resynchronization is disabled.

The amount of memory you need to allocate for each bitplane depends upon the resolution modes you have selected, because high resolution or interlaced playfields contain more data and require larger bitplanes.

1.10 3 / Forming a Basic Playfield / Allocating Memory for Bitplanes

After you set the number of bitplanes and specify resolution modes ↩
, you
are ready to allocate memory. A bitplane consists of an end-to-end sequence of words at consecutive memory locations. When operating under the Amiga operating system, use a system call such as AllocMem() to remove a block of memory from the free list and make it available to the program.

A specialized allocation function named AllocRaster() in the graphics.library is recommended for all bitplane allocations. AllocRaster() will pad the allocation to properly align scan lines for the hardware.

If the machine has been taken over, simply reserve an area of memory for the bitplanes. Next, set the bitplane pointer registers (
BPLxPTH/BPLxPTL
)
to point to the starting memory address of each bitplane you are using. The starting address is the memory word that contains the bits of the upper left-hand corner of the bitplane.

Tables 3-7 and 3-8 show how much memory is needed for basic playfield modes under NTSC and PAL, respectively. You may need to balance your color and resolution requirements against the amount of available memory you have.

Table 3-7: Playfield Memory Requirements, NTSC

Picture Size	Modes	Number of Bytes per Bitplane
320 X 200	Low resolution, non-interlaced	8,000
320 X 400	Low resolution, interlaced	16,000
640 X 200	High resolution, non-interlaced	16,000
640 X 400	High resolution, interlaced	32,000

Keep in mind that the number of bytes you allocate for a bitplane must be even.

Table 3-8: Playfield Memory Requirements, PAL

Picture Size	Modes	Number of Bytes per Bitplane
320 X 256	Low resolution, non-interlaced	8,192
320 X 512	Low resolution, interlaced	16,384
640 X 256	High resolution, non-interlaced	16,384
640 X 512	High resolution, interlaced	32,768

NTSC Example of Bitplane Size

1.11 3 / / Allocating Memory for Bitplanes / NTSC Example of Bitplane Size

For example, using a normal, NTSC, low resolution, non-interlaced display
with 320 pixels across each display line and a total of 200 display lines, each line of the bitplane requires 40 bytes (320 bits divided by 8 bits per byte = 40). Multiply the 200 lines times 40 bytes per line to get 8,000 bytes per bitplane as given above.

A low resolution, non-interlaced playfield made up of two bitplanes requires 16,000 bytes of memory area. The memory for each bitplane must be continuous, so you need to have two 8,000-byte blocks of available memory. Figure 3-7 shows an 8,000-byte memory area organized as 200 lines of 40 bytes each, providing 1 bit for each pixel position in the display plane.

```
  _______________________________        _______________________________
 |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| ----> |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

        Memory Location N                     Memory Location N+38
  _______________________________        _______________________________
 |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| ----> |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

      Memory Location N+40          |         Memory Location N+78
                                    |
                                    |
                                   \|/
  _______________________________        _______________________________
 |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| ----> |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
```

```
      Memory Location N+7960                  Memory Location N+7998

          Figure 3-7: Memory Organization for a Basic Bitplane
```

Access to bitplanes in memory is provided by two address registers,

BPLxPTH and BPLxPTL
, for each bitplane (12 registers in all). The "x" position in the name holds the bitplane number; for example BPL1PTH and BPL1PTL hold the starting address of PLANE 1. Pairs of registers with names ending in PTH and PTL contain 19-bit addresses. 68000 programmers may treat these as one 32-bit address and write to them as one long word. You write to the high order word, which is the register whose name ends in "PTH."

The example below shows how to set the bitplane pointers. Assuming two bitplanes, one at $21000 and the other at $25000, the processor sets BPL1PT to $21000 and BPL2PT to $25000. Note that this is usually the Copper's task.

```
;
; Since the bitplane pointer registers are mapped as full 680x0 long-word
; data, we can store the addresses with a 32-bit move...
;
   LEA     CUSTOM,a0               ; Get base address of custom hardware...
   MOVE.L  $21000,BPL1PTH(a0)      ; Write bitplane 1 pointer
   MOVE.L  $25000,BPL2PTH(a0)      ; Write bitplane 2 pointer
```

Note that the memory requirements given here are for the playfield only. You may need to allocate additional memory for other parts of the display -- sprites , audio , animation -- and for your application programs. Memory allocation for other parts of the display is discussed in the chapters describing those topics.

1.12 3 / Basic Playfield / Coding the Bitplanes For Correct Coloring

After you have specified the number of bitplanes and set the bitplane pointers, you can actually write the color register codes into the bitplanes.

A One- or Two-Color Playfield

A Playfield of Three or More Colors

1.13 3 / / Coding For Correct Coloring / A One- or Two-Color Playfield

For a one-color playfield, all you need do is write "0"s in all ↩ the bits
of the single bitplane as shown in the example below. This code fills a
low resolution bitplane with the background color (
COLOR00
) by writing
all "0"s into its memory area. The bitplane starts at $21000 and is 8,000
bytes long.

```
        LEA     $21000,a0       ; Point at bitplane
        MOVE.W  #2000,d0        ; Write 2000 longwords = 8000 bytes
LOOP:   MOVE.L  #0,(a0)+        ; Write out a zero
        DBRA    d0,LOOP         ; Decrement counter and loop until done...
```

For a two-color playfield, you define a bitplane that has "0"s where you
want the background color and "1"s where you want the color in register 1.
The following example code is identical to the last example, except the
bitplane is filled with $FF00FF00 instead of all 0's. This will produce
two colors.

```
        LEA     $21000,a0        ; Point at bitplane
        MOVE.W  #2000,d0         ; Write 2000 longwords = 8000 bytes
LOOP:   MOVE.L  #$FF00FF00,(a0)+ ; Write out $FF00FF00
        DBRA    d0,LOOP         ; Decrement counter and loop until done...
```

1.14 3 / / Correct Coloring / A Playfield of Three or More Colors

For three or more colors, you need more than one bitplane. The ↩ task here
is to define each bitplane in such a way that when they are combined for
display, each pixel contains the correct combination of bits. This is a
little more complicated than a playfield of one bitplane. The following
examples show a four-color playfield, but the basic idea and procedures
are the same for playfields containing up to 32 colors.

Figure 3-8 shows two bitplanes forming a four-color playfield:

```
        Image in                          Results in a display
       bitplane 2         ______            similar to this:
                         |      |
            +---------------- 0 1 | Color 1 -------+
            |            |___ _|                   |
            |                |                     |
            |                |                     |
          +---+            +---+                   |
     0 0 0|0 0|0 0 0    0 0 0|1 1|0 0 0          ***
          |   |              |   |               ***
     0 0 0|0 0|0 0 0    0 0 0|1 1|0 0 0          ***
          |   |              |   |               ***
     0 0 0|0 0|0 0 0    0 0 0|1 1|0 0 0          ***
          +---+            +---+                 ***
     1 1 1 0 0 1 1 1    1 1 1 0 0 1 1 1     ......   ......
```

```
                                          ......   ......
    1 1 1 0 0 1 1 1     1 1 1 0 0 1 1 1    ......  \......
        +---+               +---+                ...\
    0 0 0|1 1|0 0 0     0 0 0|1 1|0 0 0          ... \
         |   |               |   |               ...  \
    0 0 0|1 1|0 0 0     0 0 0|1 1|0 0 0          ...   \__   Color 00
         |   |               |   |               ...       (background)
    0 0 0|1 1|0 0 0     0 0 0|1 1|0 0 0          ...
        +---+               +---+                 |
          |                   |                   |
          |                ___|_                  |
          |               |     |                 |
          +---------------- 1 1 | Color 3 -------+
                          |_____|
```

Figure 3-8: Combining Bitplanes

You place the correct "1"s and "0"s in both bitplanes to give each pixel in the picture above the correct color.

In a single playfield you can combine up to five bitplanes in this way. Using five bitplanes allows a choice of 32 different colors for any single pixel. The playfield color selection charts at the end of this chapter summarize the bit combinations for playfields made from four and five bitplanes.

1.15 3 / Forming Basic Playfield / Defining the Size of the Display Window

After you have completely defined the playfield, you need to define the size of the display window, which is the actual size of the on-screen display. Adjustment of display window size affects the entire display area, including the border and the sprites , not just the playfield. You cannot display objects outside of the defined display window. Also, the size of the border around the playfield depends on the size of the display window.

The basic playfield described in this section is the same size as the screen display area and also the same size as the display window. This is not always the case; often the display window is smaller than the actual "big picture" of the playfield as defined in memory (the raster).

A display window that is smaller than the playfield allows you to displaysome segment of a large playfield or scroll the playfield through the window. You can also define display windows larger than the basic playfield. These larger playfields and different-sized display windows are described in the section below called

```
                 Bitplanes and Display Windows of All Sizes
                .
```

You define the size of the display window by specifying the vertical and horizontal positions at which the window starts and stops and writing these positions to the display window registers. The resolution of vertical start and stop is one scan line. The resolution of horizontal start and stop is one low resolution pixel. Each position on the screen defines the horizontal and vertical position of some pixel, and this position is specified by the x and y coordinates of the pixel. This document shows the x and y coordinates in this form: (x,y).

Although the coordinates begin at (0,0) in the upper left-hand corner of the screen, the first horizontal position normally used is $81 and the first vertical position is $2C. The horizontal and vertical starting positions are the same both for NTSC and for PAL.

The hardware allows you to specify a starting position before ($81,$2C), but part of the display may not be visible. The difference between the absolute starting position of (0,0) and the normal starting position of ($81,$2C) is the result of the way many video display monitors are designed.

To overcome the distortion that can occur at the extreme edges of the screen, the scanning beam sweeps over a larger area than the front face of the screen can display. A starting position of ($81,$2C) centers a normal size display, leaving a border of eight low resolution pixels around the display window. Figure 3-9 shows the relationship between the normal display window, the visible screen area, and the area actually covered by the scanning beam.

```
  ($81,$2C) __
              \
       (0,0) __\
               \
                \
                 +\----------------------------+
                 |#\###########################|
                 |##\##########################|
                 |###+---------------------+###|
                 |###|  _ _ _ _ _ _ _ _ _  |###|  __ visible screen
                 |###| |   |             | |###| /     boundaries
                 |###|     |               |###|/
                 |###| |----- 320 -------| |###/
                 |###|     |               |##/|
                 |###| |   |             | |#/#|
                 |###|     |               |/##|
                 |###| |                 | |###|
                 |###|    200              |###|
                 |###| |                 | |###|
                 |###|     |               |###|
                 |###| |   |             | |###|
                 |###|     |               |###|
                 |###| |_ _|_ _ _ _ _ _ _| |###|
                 |###|  \               /  |###|
                 |###+---\-------------/---+###|
```

```
|########\###########/########|
|#########\#########/#########|
+----------\-------/----------+
            \     /
             \   /
         display window
starting and stopping positions
```

Figure 3-9: Positioning the On-screen Display

Setting the Display Window Starting Position

Setting the Display Window Stopping Position

1.16 3 / / Size Display Window / Setting Display Window Starting Position

A horizontal starting position of approximately $81 and a vertical starting position of approximately $2C centers the display on most standard television screens. If you select high resolution mode (640 pixels horizontally) or interlaced mode (400 lines NTSC, 512 PAL) the starting position does not change. The starting position is always interpreted in low resolution, non-interlaced mode. In other words, you select a starting position that represents the correct coordinates in low resolution, non-interlaced mode.

The register
DIWSTRT
(for "Display Window Start") controls the display window starting position. This register contains both the horizontal and vertical components of the display window starting positions, known respectively as HSTART and VSTART. The following example sets DIWSTRT for a basic playfield. You write $2C for VSTART and $81 for HSTART.

```
LEA     CUSTOM,a0           ; Get base address of custom hardware...
MOVE.W  #$2C81,DIWSTRT(a0)   ; Display window start register...
```

1.17 3 / / Size Display Window / Setting Display Window Stopping Position

You also need to set the display window stopping position, which is the lower right-hand corner of the display window. If you select high resolution or interlaced mode, the stopping position does not change. Like the starting position, it is interpreted in low resolution, non-interlaced mode.

The register
DIWSTOP

(for Display Window Stop) controls the display window stopping position. This register contains both the horizontal and vertical components of the display window stopping positions, known respectively as HSTOP and VSTOP. The instructions below show how to set HSTOP and VSTOP for the basic playfield, assuming a starting position of ($81,$2C). Note that the HSTOP value you write is the actual value minus 256 ($100). The HSTOP position is restricted to the right-hand side of the screen. The normal HSTOP value is ($1C1) but is written as ($C1). HSTOP is the same both for NTSC and for PAL.

The VSTOP position is restricted to the lower half of the screen. This is accomplished in the hardware by forcing the MSB of the stop position to be the complement of the next MSB. This allows for a VSTOP position greater than 256 ($100) using only 8 bits. Normally, the VSTOP is set to ($F4) for NTSC, ($2C) for PAL.

The normal NTSC DIWSTRT is ($2C81).
The normal NTSC DIWSTOP is ($F4C1).

The normal PAL DIWSTRT is ($2C81).
The normal PAL DIWSTOP is ($2CC1).

The following example sets DIWSTOP for a basic playfield to $F4 for the vertical position and $C1 for the horizontal position.

```
LEA     CUSTOM,a0              ; Get base address of custom hardware...
MOVE.W  #$F4C1,DIWSTOP(a0)     ; Display window stop register...
```

Table 3-9: DIWSTRT and DIWSTOP Summary

	Nominal Values		Possible Values	
	NTSC	PAL	MIN	MAX
DIWSTRT:				
VSTART	$2C	$2C	$00	$FF
HSTART	$81	$81	$00	$FF
DIWSTOP:				
VSTOP	$F4	$2C (=$12C)	$80	$7F (=$17F)
HSTOP	$C1	$C1	$00 (=$100)	$FF (=$1FF)

The minimum and maximum values for display windows have been extended in the enhanced version of the Amiga's custom chip set (ECS). See

Appendix C, Enhanced Chip Set for more information about the display window registers .

1.18 3 / Basic Playfield / Telling the System How to Fetch and Display Data

After defining the size and position of the display window, you need to give the system the on-screen location for data fetched from memory. To do this, you describe the horizontal positions where each line starts and stops and write these positions to the data-fetch registers. The data-fetch registers have a four-pixel resolution (unlike the display window registers, which have a one-pixel resolution). Each position specified is four pixels from the last one. Pixel 0 is position 0; pixel 4 is position 1, and so on.

The data-fetch start and display window starting positions interact with each other. It is recommended that data-fetch start values be restricted to a programming resolution of 16 pixels (8 clocks in low resolution mode, 4 clocks in high resolution mode). The hardware requires some time after the first data fetch before it can actually display the data. As a result, there is a difference between the value of window start and data-fetch start of 4.5 color clocks.

```
   The normal low resolution DDFSTRT is ($0038).
   The normal high resolution DDFSTRT is ($003C).
```

Recall that the hardware resolution of display window start and stop is twice the hardware resolution of data fetch:

```
          $81
          ---  - 8.5 = $38
           2

          $81
          ---  - 4.5 = $3C
           2
```

The relationship between data-fetch start and stop is

```
   DDFSTRT = DDFSTOP - (8 * (word count - 1)) for low resolution
   DDFSTRT = DDFSTOP - (4 * (word count - 2)) for high resolution
```

The normal low resolution DDFSTOP is ($00D0). The normal high resolution DDFSTOP is ($00D4).

The following example sets data-fetch start to $0038 and data-fetch stop to $00D0 for a basic playfield.

```
        LEA     CUSTOM,a0                ; Point to base hardware address
        MOVE.W  #$0038,DDFSTRT(a0)       ; Write to DDFSTRT
        MOVE.W  #$00D0,DDFSTOP(a0)       ; Write to DDFSTOP
```

You also need to tell the system exactly which bytes in memory belong on each horizontal line of the display. To do this, you specify the modulo value. Modulo refers to the number of bytes in memory between the last word on one horizontal line and the beginning of the first word on the next line. Thus, the modulo enables the system to convert bitplane data stored in linear form (each data byte at a sequentially increasing memory address) into rectangular form (one "line" of sequential data followed by another line). For the basic playfield, where the playfield in memory is the same size as the display window, the modulo is zero because the memory area contains exactly the same number of bytes as you want to display on the screen. Figures 3-10 and 3-11 show the basic bitplane layout in memory and how to make sure the correct data is retrieved.

The bitplane address pointers (
BPLxPTH and BPLxPTL
) are used by the
system to fetch the data to the screen. These pointers are dynamic; once the data fetch begins, the pointers are continuously incremented to point to the next word to be fetched (data is fetched two bytes at a time). When the end-of-line condition is reached (defined by the data-fetch register, DDFSTOP) the modulo is added to the bitplane pointers, adjusting the pointer to the first word to be fetched for the next horizontal line.

```
 ________________________________________________________________________
|                                                                        |
|   Data for line 1:                                                     |
|                                                                        |
| Location:     START      START+2       START+4    ...      START+38    |
|               -----      -------       -------             --------    |
|             leftmost   next word     next word           last display |
|           display word                                       word      |
|                                                                        |
|                                                              /|\       |
|                         Screen data fetch stops (DDFSTOP) for    |      |
|                         each horizontal line after the last      |      |
|                         word on the line has been fetched________|      |
|________________________________________________________________________|
```

Figure 3-10: Data Fetched for the First Line When Modulo = 0

After the first line is fetched, the bitplane pointers

BPLxPTH and BPLxPTL
contain the value START+40. The modulo (in this
case, 0) is added to the current value of the pointer, so when the pointer begins the data fetch for the next line, it fetches the data you want on that line. The data for the next line begins at memory location START+40.

```
 ________________________________________________________________________
|                                                                        |
|   Data for line 2:                                                     |
|                                                                        |
| Location:    START+40     START+42     START+44    ...     START+78    |
|              --------     --------     --------            --------    |
|              leftmost     next word    next word          last display |
|           display word                                       word      |
|________________________________________________________________________|
```

Figure 3-11: Data Fetched for the Second Line When Modulo = 0

Note that the pointers always contain an even number, because data is fetched from the display a word at a time.

There are two modulo registers -- BPL1MOD for the odd-numbered bitplanes and BPL2MOD for the even-numbered bitplanes. This allows for differing modulos for each playfield in
dual-playfield mode
. For normal
applications, both BPL1MOD and BPL2MOD will be the same.

The following example sets the modulo to 0 for a low resolution playfield with one bitplane. The bitplane is odd-numbered.

```
        MOVE.W  #0,BPL1MOD+CUSTOM       ; Set modulo to 0
```

Data Fetch in High resolution Mode

Modulo in Interlaced Mode

1.19 3 / / How to Fetch and Display Data / in High resolution Mode

When you are using high resolution mode to display the basic playfield, you need to fetch 80 bytes for each line, instead of 40.

1.20 3 / / How to Fetch and Display Data / Modulo in Interlaced Mode

For interlaced mode, you must redefine the modulo, because interlaced mode uses two separate scannings of the video screen for a single display of the playfield. During the first scanning, the odd-numbered lines are fetched to the screen; and during the second scanning, the even-numbered lines are fetched.

The bitplanes for a full-screen-sized, interlaced display are 400 NTSC (512 PAL), rather than 200 NTSC (256 PAL), lines long. Assuming that the playfield in memory is the normal 320 pixels wide, data for the interlaced picture begins at the following locations (these are all byte addresses):

```
Line 1   START
Line 2   START+40
Line 3   START+80
Line 4   START+120
```

and so on. Therefore, you use a modulo of 40 to skip the lines in the other field. For odd fields, the bitplane pointers begin at START. For even fields, the bitplane pointers begin at START+40.

You can use the Copper to handle resetting of the bitplane pointers for interlaced displays.

1.21 3 / Basic Playfield / Displaying and Redisplaying the Playfield

You start playfield display by making certain that the bitplane pointers are set and bitplane DMA is turned on. You turn on bitplane DMA by writing a 1 to bit BPLEN in the DMACON (for DMA control) register. See Chapter 7, System Control Hardware, for instructions on setting this register.

Each time the playfield is redisplayed, you have to reset the bitplane pointers. Resetting is necessary because the pointers have been incremented to point to each successive word in memory and must be repointed to the first word for the next display. You write Copper instructions to handle the redisplay or perform this operation as part of a vertical blanking task.

1.22 3 / Forming a Basic Playfield / Enabling the Color Display

The stock A1000 has a color composite output and requires bit 9 (↩ COLOR_ON) set in BPLCON0 to create a color composite display signal. Without the addition of specialized hardware, the A500, A2000 and A3000 cannot generate color composite output.

NOTE:

The color burst enable does not affect the RGB video signal. RGB video is correctly generated regardless of the output of the composite video signal.

1.23 3 / Forming a Basic Playfield / Basic Playfield Summary

The steps for defining a basic playfield are summarized below:

1. Define Playfield Characteristics
----
a. Specify color for each pixel:

   * Load desired colors in color table registers.

   * Define color of each pixel in terms of the binary value that points at the desired color register.

* Build bitplanes and set bitplane registers:

Bits 12–14 in
BPLCON0
– number of bitplanes (
BPU2 – BPU0
).

BPLxPTH
– pointer to bitplane starting position in memory
(written as a long word).

b.
Specify resolution
:

* Low resolution:

320 pixels in each horizontal line.
Clear bit 15 in register
BPLCON0
(
HIRES
).

* High resolution:

640 pixels in each horizontal line.
Set bit 15 in register
BPLCON0
(
HIRES
).

c.
Specify interlaced or non-interlaced mode
:

* Interlaced mode:

400 vertical lines for NTSC, 512 for PAL.
Set bit 2 in register
BPLCON0
(
LACE
).

* Non-interlaced mode:

200 vertical lines for NTSC, 256 for PAL.
Clear bit 2 in
BPLCON0
(
LACE
).

2.

```
               Allocate Memory
              . To calculate data-bytes in the total bitplanes,
   -----------------  use the following formula:

         Bytes per line * lines in playfield * number of bitplanes

3.
               Define Size of Display Window
              .
   -------------------------------
   * Write start position of display window in
               DIWSTRT
              :

      Horizontal position in bits 0 through 7 (low order bits).
      Vertical position in bits 8 through 15 (high order bits).

   * Write stop position of display window in
               DIWSTOP
              :

      Horizontal position in bits 0 through 7.
      Vertical position in bits 8 through 15.

4.
               Define Data Fetch
              . Set registers
               DDFSTRT
               and
               DDFSTOP
              :
   -------------------
   * For
               DDFSTRT
              , use the horizontal position as shown in

               Setting the Display Window Starting Position
              .

   * For
               DDFSTOP
              , use the horizontal position as shown in

               Setting the Display Window Stopping Position
              .

5.
               Define Modulo
              . Set registers
               BPL1MOD and BPL2MOD
              .  Set modulo to 0
   ---------------  for non-interlaced, 40 for interlaced.

6. Write Copper Instructions To Handle
               Redisplay
              .
   -----------------------------------------------
```

7.
Enable Color Display
.For the A1000: set bit 9 in
BPLCON0
to enable the
---------------------- the color display on a composite video monitor.
RGB video is not affected. Only the A1000 has
color composite video output, other Amiga models
cannot enable this feature using standard
hardware.

1.24 3 Playfield Hardware / Forming a Dual-playfield Display

For more flexibility in designing your background display, you can specify two playfields instead of one. In dual-playfield mode, one playfield is displayed directly in front of the other. For example, a computer game display might have some action going on in one playfield in the background, while the other playfield is showing a control panel in the foreground. You can then change either the foreground or the background without having to redesign the entire display. You can also move the two playfields independently.

A dual-playfield display is similar to a single-playfield display, differing only in these aspects:

* Each playfield in a dual display is formed from one, two or three bitplanes.

* The colors in each playfield (up to seven plus transparent) are taken from different sets of color registers.

* You must set a bit to activate dual-playfield mode.

Figure 3-12 shows a dual-playfield display.

Figure 3-12: A Dual-playfield Display

In Figure 3-12, one of the colors in each playfield is "transparent" (color 0 in playfield 1 and color 8 in playfield 2). You can use transparency to allow selected features of the background playfield to show through.

In dual-playfield mode, each playfield is formed from up to three bitplanes. Color registers 0 through 7 are assigned to playfield 1, depending upon how many bitplanes you use. Color registers 8 through 15 are assigned to playfield 2.

Bitplane Assignment in Dual-Playfield Mode

Color Registers in Dual-Playfield Mode

Dual-Playfield Priority and Control

Activating Dual-Playfield Mode

Dual Playfield Summary

1.25 3 / Dual-playfield / Bitplane Assignment in Dual-Playfield Mode

The three odd-numbered bitplanes (1, 3, and 5) are grouped together by the hardware and may be used in playfield 1. Likewise, the three even-numbered bitplanes (2, 4, and 6) are grouped together and may be used in playfield 2. The bitplanes are assigned alternately to each playfield, as shown in Figure 3-13.

About dual-playfield bitplanes.

In high resolution mode, you can have up to two bitplanes in each playfield -- bitplanes 1 and 3 in playfield 1 and bitplanes 2 and 4 in playfield 2.

```
     Number of
     Bitplanes
    "turned on."     Plafield 1 *            Playfield 2 *
    ------------------------------------------------------
         0              none                      none
                     ____________
                    |            |
         1          | 1          |
                    |            |
                    |____________|
                     ____________             ____________
                    |            |           |            |
         2          | 1          |           | 2          |
                    |            |           |            |
                    |____________|           |____________|
                     ____________             ____________
                    |            |           |            |
         3          | 1 _________|__         | 2          |
                    |  |            |        |            |
                    |__| 3          |        |____________|
                       |            |
                       |____________|
                     ____________             ____________
                    |            |           |            |
         4          | 1 _________|__         | 2 _________|__
                    |  |            |        |  |            |
                    |__| 3          |        |__| 4          |
                       |            |           |            |
                       |____________|           |____________|
                     ____________             ____________
                    |            |           |            |
         5          | 1 _________|__         | 2 _________|__
                    |  |            |        |  |            |
```

```
                      |__| 3 _________|__      |__| 4            |
                         |  |            |        |              |
                         |__| 5          |        |______________|
                            |            |
                            |____________|
                       ____________            ____________
                      |            |          |            |
            6         | 1 _________|__        | 2 _________|__
                      |  |            |       |  |            |
                      |__| 3 _________|__     |__| 4 _________|__
                         |  |            |       |  |            |
                         |__| 5          |       |__| 6          |
                            |            |          |            |
                            |____________|          |____________|
```

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

Figure 3-13: How Bitplanes Are Assigned to Dual Playfields

1.26 3 / Dual-playfield Display / Color Registers in Dual-Playfield Mode

When you are using dual playfields, the hardware interprets color numbers for playfield 1 from the bit combinations of bitplanes 1, 3, and 5. Bits from PLANE 5 have the highest significance and form the most significant digit of the color register number. Bits from PLANE 0 have the lowest significance. These bit combinations select the first eight color registers from the color palette as shown in Table 3-10.

PLAYFIELD 1

Bit Combination	Color Selected
000	Transparent mode
001	COLOR1
010	COLOR2
011	COLOR3
100	COLOR4
101	COLOR5
110	COLOR6
111	COLOR7

Table 3-10: Playfield 1 Color Registers -- Low resolution Mode

The hardware interprets color numbers for playfield 2 from the bit combinations of bitplanes 2, 4, and 6. Bits from PLANE 6 have the highest significance. Bits from PLANE 2 have the lowest significance. These bit combinations select the color registers from the second eight colors in the color table as shown in Table 3-11.

PLAYFIELD 2

Bit Combination	Color Selected
000	Transparent mode
001	COLOR9
010	COLOR10
011	COLOR11
100	COLOR12
101	COLOR13
110	COLOR14
111	COLOR15

Table 3-11: Playfield 2 Color Registers -- Low resolution Mode

Combination 000 selects transparent mode, to show the color of whatever object (the other playfield, a sprite, or the background color) may be "behind" the playfield.

Table 3-12 shows the color registers for high resolution, dual-playfield mode.

PLAYFIELD 1

Bit Combination	Color Selected
00	Transparent mode
01	COLOR1
10	COLOR2
11	COLOR3

PLAYFIELD 2

Bit Combination	Color Selected
00	Transparent mode
01	COLOR9
10	COLOR10
11	COLOR11

Table 3-12: Playfields 1 and 2 Color Registers -- High resolution Mode

1.27 3 / Dual-playfield Display / Dual-Playfield Priority and Control

Either playfield 1 or 2 may have priority; that is, either one may be displayed in front of the other. Playfield 1 normally has priority. The bit known as PF2PRI (bit 6) in register BPLCON2 is used to control priority. When PF2PRI = 1, playfield 2 has priority over playfield 1. When PF2PRI = 0, playfield 1 has priority.

You can also control the relative priority of playfields and sprites . Chapter 7, System Control Hardware, shows you how to control the priority of these objects.

You can control the two playfields separately as follows:

* They can have different-sized representations in memory, and different portions of each one can be selected for display.

* They can be scrolled separately.

An important warning.

You must take special care when scrolling one playfield and holding the other stationary. When you are scrolling low resolution playfields, you must fetch one word more than the width of the playfield you are trying to scroll (two words more in high resolution mode) in order to provide some data to display when the actual scrolling takes place. Only one data-fetch start register and one data-fetch stop register are available, and these are shared by both playfields. If you want to scroll one playfield and hold the other, you must adjust the data-fetch start and data-fetch stop to handle the playfield being scrolled. Then, you must adjust the modulo and the bitplane pointers of the playfield that is not being scrolled to maintain its position on the display. In low resolution mode, you adjust the pointers by -2 and the modulo by -2. In high resolution mode, you adjust the pointers by -4 and the modulo by -4.

1.28 3 / Forming a Dual-playfield Display / Activating Dual-Playfield Mode

Writing a 1 to bit 10 (called DBLPF) of the bitplane control register

BPLCON0
selects dual-playfield mode. Selecting dual-playfield mode changes both the way the hardware groups the bitplanes for color interpretation -- all odd-numbered bitplanes are grouped together and all even-numbered bitplanes are grouped together, and the way hardware can move the bitplanes on the screen.

1.29 3 / Forming a Dual-playfield Display / Dual Playfield Summary

The steps for defining dual playfields are almost the same as those for defining the basic playfield. Only in the following steps does the dual-playfield creation process differ from that used for the basic playfield:

* Loading colors into the registers.

  Keep in mind that color registers 0-7 are used by playfield 1 and registers 8 through 15 are used by playfield 2 (if there are three bitplanes in each playfield).

* Building bitplanes.

  Recall that playfield 1 is formed from PLANES 1, 3, and 5 and playfield 2 from PLANES 2, 4, and 6.

* Setting the modulo registers.

  Write the modulo to both BPL1MOD and BPL2MOD as you will be using both odd- and even-numbered bitplanes.

These steps are added:

* Defining priority.

  If you want playfield 2 to have priority, set bit 6 (PF2PRI) in BPLCON2 to 1.

* Activating dual-playfield mode.

  Set bit 10 (DBLPF) in BPLCON0 to 1.

1.30 3 Playfield Hardware / Bitplanes and Display Windows of All Sizes

You have seen how to form single and dual playfields in which the playfield in memory is the same size as the display window. This section shows you how to define and use a playfield whose big picture in memory is larger than the display window, how to define display windows that are larger or smaller than the normal playfield size, and how to move the display window in the big picture.

When the Big Picture is Larger than the Display Window

Maximum Display Window Size

1.31 3 / All Sizes / When the Big Picture is Larger than the Display Window

If you design a memory picture larger than the display window, you must choose which part of it to display. Displaying a portion of a larger playfield differs in the following ways from displaying the basic playfields described up to now:

* If the big picture in memory is larger than the display window, you must respecify the modulos. The modulo must be some value other than 0.

* You must allocate more memory for the larger memory picture.

Specifying the Modulo

Specifying the Data Fetch

Memory Allocation

Selecting the Display Window Starting Position

Selecting the Stopping Position

1.32 3 / / When Picture is Larger than Window / Specifying the Modulo

For a memory picture wider than the display window, you need to respecify the modulo so that the correct data words are fetched for each line of the display. As an example, assume the display window is the standard 320 pixels wide, so 40 bytes are to be displayed on each line. The big picture in memory, however, is exactly twice as wide as the display window, or 80 bytes wide. Also, assume that you wish to display the left half of the big picture. Figure 3-14 shows the relationship between the big picture and the picture to be displayed.

```
      START                                                 START+78
         __________________________|___________________________
```

```
         |                          |                           |
         |<-------------------------+-------------------------->|
         |       Width of the bit-plane defined in RAM          |
         |                          |                           |
         |<------------------------>|                           |
         | Width of defined screen  |                           |
         | on which bit-plane data  |                           |
         |      is to appear        |                           |
         |                          |                           |
         |                          |                           |
```

Figure 3-14: Memory Picture Larger than the Display

Because 40 bytes are to be fetched for each line, the data fetch for line 1 is as shown in Figure 3-15.

```
  ________________________________________________________________________
 |                                                                        |
 |   Data for line 1:                                                     |
 |                                                                        |
 | Location:     START      START+2       START+4    ...      START+38    |
 |               -----      -------       -------             --------    |
 |              leftmost   next word     next word          last display  |
 |            display word                                      word      |
 |                                                                        |
 |                                                              /|\       |
 |                      Screen data fetch stops (
               DDFSTOP
              ) for  |       |
 |                      each horizontal line after the last      |        |
 |                      word on the line has been fetched________|        |
 |________________________________________________________________________|
```

Figure 3-15: Data Fetch for the First Line When Modulo = 40

At this point,
BPLxPTH and BPLxPTL
contain the value START+40. The
modulo, which is 40, is added to the current value of the pointer so that when it begins the data fetch for the next line, it fetches the data that you intend for that line. The data fetch for line 2 is shown in Figure 3-16.

```
  ________________________________________________________________________
 |                                                                        |
 |   Data for line 2:                                                     |
 |                                                                        |
 | Location:    START+80     START+82     START+84    ...     START+118   |
 |              --------     --------     --------            ---------   |
 |              leftmost     next word    next word         last display  |
 |            display word                                      word      |
 |________________________________________________________________________|
```

Figure 3-16: Data Fetch for the Second Line When Modulo = 40

To display the right half of the big picture, you set up
vertical blanking routine to start the bitplane pointers at location
START+40 rather than START with the modulo remaining at 40. The data
layout is shown in Figures 3-17 and 3-18.

```
 ________________________________________________________________________
|                                                                        |
|   Data for line 1:                                                     |
|                                                                        |
| Location:    START+40     START+42     START+44    ...     START+78    |
|              --------     --------     --------            --------    |
|              leftmost     next word    next word         last display  |
|            display word                                      word       |
|________________________________________________________________________|
```

Figure 3-17: Data Layout for First Line -- Right Half of Big Picture

Now, the bitplane pointers contain the value START+80. The modulo (40) is
added to the pointers so that when they begin the data fetch for the
second line, the correct data is fetched.

```
 ________________________________________________________________________
|                                                                        |
|   Data for line 2:                                                     |
|                                                                        |
| Location:    START+120    START+122    START+124    ...    START+158   |
|              ---------    ---------    ---------           ---------   |
|              leftmost     next word    next word         last display  |
|            display word                                      word       |
|________________________________________________________________________|
```

Figure 3-18: Data Layout for Second Line -- Right Half of Big Picture

Remember, in high resolution mode, you need to fetch twice as many bytes
as in low resolution mode. For a normal-sized display, you fetch 80 bytes
for each horizontal line instead of 40.

1.33 3 / / When Picture is Larger than Window / Specifying the Data Fetch

The data-fetch registers specify the beginning and end positions for data
placement on each horizontal line of the display. You specify
data fetch
in the same way as shown in the section called "Forming a Basic Playfield."

1.34 3 / / When Picture is Larger than Display Window / Memory Allocation

For larger memory pictures, you need to allocate more memory. Here is a formula for calculating memory requirements in general:

```
     bytes per line * lines in playfield * # of bitplanes
```

The number of bytes must be even. Thus, if the wide playfield described in this section is formed from two bitplanes, it requires:

```
            80 * 200 * 2 = 32,000 bytes of memory
```

Recall that this is the memory requirement for the playfield alone. You need more memory for any sprites , animation , audio , or application programs you are using.

The amount of Chip memory is one of the basic constraints on the size of playfields. For instance, a playfield 2000 by 2000 pixels with five bitplanes would exceed even the two megabytes of Chip memory possible on an Amiga 3000. Another constraint on playfield size is the bit plane modulos which limit the width (but not the height) of a playfield to 262,144 pixels.

As a practical matter, the blitter size registers also limit the size of playfields (unless the 680x0 CPU is used for drawing operations). With the original chip set the largest area the blitter can draw in is 1008 by 1024. With the Enhanced Chip Set (ECS), the largest area the blitter can draw in is increased to 16368 by 16384 pixels. For more information on ECS and blitter limits refer to Appendix C, Enhanced Chip Set.

1.35 3 / / Picture Larger / Selecting the Display Window Starting Position

The display window starting position is the horizontal and vertical coordinates of the upper left-hand corner of the display window. One register, DIWSTRT , holds both the horizontal and vertical coordinates, known as HSTART and VSTART. The eight bits allocated to HSTART are assigned to the first 256 positions, counting from the leftmost possible position. Thus, you can start the display window at any pixel position within this range.

```
      0                         255                       511 ($1FF)
       __________________________|___________________________
      |                           |                            |
      |                           |                            |
      |<------------------------->|                            |
      |         HSTART of         |                            |
      |  DISPLAY WINDOW occurs    |                            |
      |     is this region        |                            |
      |                           |                            |
      |                           |                            |
      |                           |                            |
      |                           |                            |
      |                           |                            |
```

```
      |                          |                           |
      |                          |                           |
      |__________________________|___________________________|

      Figure 3-19: Display Window Horizontal Starting Position
```

The eight bits allocated to VSTART are assigned to the first 256 positions counting down from the top of the display.

```
       ______________________________________________________
      |                                 /|\                  |  0
      |                                  |                   |
      |                        VSTART of |                   |
      |                   DISPLAY WINDOW |                   |
      |                        occurs in |                   |
      |                      this region |                   |
      |                                  |                   |
      |                                  |                   |
      |                                 \|/                  |
      |------------------------------------------------------+- 255
      |                                                      |
 262 -+------------------- (NTSC)                            |
      |                                                      |
      |______________________________________________________|  383 ($17F)

       Figure 3-20: Display Window Vertical Starting Position
```

Recall that you select the values for the starting position as if the display were in low resolution, non-interlaced mode. Keep in mind, though, that for interlaced mode the display window should be an even number of lines in height to allow for equal-sized odd and even fields.

To set the display window starting position, write the value for HSTART into bits 0 through 7 and the value for VSTART into bits 8 through 15 of

DIWSTRT
.

1.36 3 / / Picture is Larger than Window / Selecting the Stopping Position

The stopping position for the display window is the horizontal and vertical coordinates of the lower right-hand corner of the display window. One register,
DIWSTOP
, contains both coordinates, known as HSTOP and VSTOP.

See the notes in the "Forming a Basic Playfield" section for instructions on setting these registers.

```
      0                         255                       511 ($1FF)
       __________________________|___________________________
```

```
      |                          |                           |
      |                          |                           |
      |                          |<------------------------->|
      |                          |        HSTOP of           |
      |                          |   DISPLAY WINDOW occurs   |
      |                          |      is this region       |
      |                          |                           |
      |                          |                           |
      |                          |                           |
      |                          |                           |
      |                          |                           |
      |                          |                           |
      |                          |                           |
      |__________________________|___________________________|
```

      Figure 3-21: Display Window Horizontal Stopping Position

Select a value that represents the correct position in low resolution,
non-interlaced mode.

```
       ______________________________________________________
      |                                                      |  0
      |                                                      |
      |                                                      |
      |------------------------------------------------------+- 128
      |                             /|\                      |
      |                              |                       |
      |                              | VSTOP of              |
      |                              | DISPLAY WINDOW        |
      |                              | occurs in             |
      |                              | this region           |
 262 -+-------------------- (NTSC)   |                       |
      |                              |                       |
      |                              |                       |
      |                             \|/                      |  383 ($17F)
       ------------------------------------------------------
```

       Figure 3-22: Display Window Vertical Stopping Position

To set the display window stopping position, write HSTOP into bits 0
through 7 and VSTOP into bits 8 through 15 of
DIWSTOP
.

1.37 3 / Bitplanes and Windows of All Sizes / Maximum Display Window Size

The maximum size of a playfield display is determined by the ↩
maximum
number of lines and the maximum number of columns. Vertically, the
restrictions are simple. No data can be displayed in the
vertical blanking area. The following table shows the allowable vertical
display area.

	NTSC	PAL
Vertical Blank Start	0	0
Vertical Blank Stop	$15 (21)	$1D (29)

	NTSC Normal	NTSC Interlaced	PAL Normal	PAL Interlaced
Displayable lines of screen video	241	483 =525-(21*2)	283	567 =625-(29*2)

Table 3-13: Maximum Allowable Vertical Screen Video

Horizontally, the situation is similar. Strictly speaking, the hardware sets a rightmost limit to DDFSTOP of ($D8) and a leftmost limit to

DDFSTRT of ($18). This gives a maximum of 25 words fetched in low resolution mode. In high resolution mode the maximum here is 49 words, because the rightmost limit remains ($D8) and only one word is fetched at this limit. However, horizontal blanking actually limits the displayable video to 368 low resolution pixels (23 words). These numbers are the same both for NTSC and for PAL. In addition, it should be noted that using a data-fetch start earlier than ($38) will disable some sprites .

Table 3-14: Maximum Allowable Horizontal Screen Video

	Lores	Hires
DDFSTRT (standard)	$0038	$003C
DDFSTOP (standard)	$00D0	$00D4
DDFSTRT (hw limits)	$0018	$0018
DDFSTOP (hw limits)	$00D8	$00D8
max words fetched	25	49
max display pixels	368 (low res)	

The limits on the display window starting and stopping positions described in this section apply to the Amiga's original custom chip set. In the Enhanced Chip Set (ECS), the limits for playfield display windows have been changed. For more information on ECS and playfield display windows refer to Appendix C, Enhanced Chip Set.

1.38 3 Playfield Hardware / Moving (Scrolling) Playfields

If you want a background display that moves, you can design a ↩ playfield
larger than the display window and scroll it. If you are using dual playfields, you can scroll them separately.

In vertical scrolling, the playfield appears to move smoothly up or down on the screen. All you need do for vertical scrolling is progressively increase or decrease the starting address for the bitplane pointers by the size of a horizontal line in the playfield. This has the effect of showing a lower or higher part of the picture each field time.

In horizontal scrolling the playfield appears to move from right-to-left or left-to-right on the screen. Horizontal scrolling works differently from vertical scrolling -- you must arrange to fetch one more word of data for each display line and delay the display of this data.

For either type of scrolling, resetting of pointers or data-fetch registers can be handled by the Copper during the vertical blanking interval.

1.39 3 / Moving (Scrolling) Playfields / Vertical Scrolling

You can scroll a playfield upward or downward in the window. Each time you display the playfield, the bitplane pointers start at a progressively higher or lower place in the big picture in memory. As the value of the pointer increases, more of the lower part of the picture is shown and the picture appears to scroll upward. As the value of the pointer decreases, more of the upper part is shown and the picture scrolls downward. On an NTSC system, with a display that has 200 vertical lines, each step can be as little as 1/200th of the screen. In interlaced mode each step could be 1/400th of the screen if clever manipulation of the pointers is used, but it is recommended that scrolling be done two lines at a time to maintain the odd/even field relationship. Using a PAL system with 256 lines on the display, the step can be 1/256th of a screen, or 1/512th of a screen in interlace.

Figure 3-23: Vertical Scrolling

To set up a playfield for vertical scrolling, you need to form bitplanes tall enough to allow for the amount of scrolling you want, write software to calculate the bitplane pointers for the scrolling you want, and allow for the Copper to use the resultant pointers.

Assume you wish to scroll a playfield upward one line at a time. To accomplish this, before each field is displayed, the bitplane pointers have to increase by enough to ensure that the pointers begin one line lower each time. For a normal-sized, low resolution display in which the modulo is 0, the pointers would be incremented by 40 bytes each time.

1.40 3 / Moving (Scrolling) Playfields / Horizontal Scrolling

You can scroll playfields horizontally from left to right or right to left on the screen. You control the speed of scrolling by specifying the amount of delay in pixels. Delay means that an extra word of data is fetched but not immediately displayed. The extra word is placed just to the left of the window's leftmost edge and before normal data fetch. As the display shifts to the right, the bits in this extra word appear on-screen at the left-hand side of the window as bits on the right-hand side disappear off-screen. For each pixel of delay, the on-screen data shifts one pixel to the right each display field. The greater the delay, the greater the speed of scrolling. You can have up to 15 pixels of delay. In high resolution mode, scrolling is in increments of 2 pixels. Figure 3-24 shows how the delay and extra data fetch combine to cause the scrolling effect.

Figure 3-24: Horizontal Scrolling

NOTE: Fetching an extra word for scrolling will disable some sprites .

To set up a playfield for horizontal scrolling, you need to:

* Define bitplanes wide enough to allow for the scrolling you need.
* Set the data-fetch registers to correctly place each horizontal line, including the extra word, on the screen.
* Set the delay bits.
* Set the modulo so that the bitplane pointers begin at the correct word for each line.
* Write Copper instructions to handle the changes during the vertical blanking interval.

1.41 3 / / Horiz. Scrolling / Specifying Data Fetch in Horizontal Scrolling

The normal data-fetch start for non-scrolled displays is ($38). If horizontal scrolling is desired, then the data fetch must start one word sooner (DDFSTRT = $0030). Incidentally, this will disable sprite 7 .

DDFSTOP remains unchanged. Remember that the settings of the data-fetch registers affect both playfields.

1.42 3 / / Horiz. Scrolling / Specifying the Modulo in Horizontal Scrolling

As always, the modulo is two counts less than the difference between the address of the next word you want to fetch and the address of the last word that was fetched. As an example for horizontal scrolling, let us assume a 40-byte display in an 80-byte "big picture." Because horizontal scrolling requires a data fetch of two extra bytes, the data for each line will be 42 bytes long.

```
       START                      START+38                     START+78
          ___________________________|___________________________
         |                           |                           |
         |                           |                           |
         |<--DISPLAY WINDOW width--> |                           |
         |                           |                           |
         |                           |                           |
         |                           |                           |
         |                           |                           |
         |<----------------MEMORY PICTURE width----------------->|
         |                           |                           |
         |                           |                           |
```

Figure 3-25: Memory Picture Larger Than the Display Window

```
  ________________________________________________________________________
 |                                                                        |
 |   Data for line 1:                                                     |
 |                                                                        |
 | Location:     START        START+2      START+4    ...     START+40    |
 |               -----        -------      -------            --------    |
 |              leftmost     next word    next word         last display |
 |            display word                                      word      |
 |________________________________________________________________________|
```

Figure 3-26: Data for Line 1 - Horizontal Scrolling

At this point, the bitplane pointers contain the value START+42. Adding the modulo of 38 gives the correct starting point for the next line.

```
 _______________________________________________________________________
|                                                                       |
|   Data for line 2:                                                    |
|                                                                       |
| Location:    START+80     START+82     START+84    ...     START+120  |
|              --------     --------     --------            ---------  |
|              leftmost     next word    next word         last display |
|            display word                                      word      |
|_______________________________________________________________________|

          Figure 3-27: Data for Line 2 -- Horizontal Scrolling
```

In the
BPLxMOD
registers you set the modulo for each bitplane used.

1.43 3 / / Horizontal Scrolling / Specifying Amount of Delay

The amount of delay in horizontal scrolling is controlled by bits 7-0 in BPLCON1. You set the delay separately for each playfield; bits 3-0 for playfield 1 (bitplanes 1, 3, and 5) and bits 7-4 for playfield 2 (bitplanes 2, 4, and 6).

Warning:

Always set all six bits, even if you have only one playfield. Set 3-0 and 7-4 to the same value if you are using only one playfield.

The following example sets the horizontal scroll delay to 7 for both playfields.

```
        MOVE.W  #$77,BPLCON1+CUSTOM
```

1.44 3 / Moving (Scrolling) Playfields / Scrolling Playfield Summary

The steps for defining a scrolled playfield are the same as those ↩ for
defining the basic playfield, except for the following steps:

*
Defining the data fetch
.

Fetch one extra word per horizontal line and start it 16 pixels before the normal (unscrolled) data-fetch start.

*
Defining the modulo
.

The modulo is two counts less than when there is no scrolling.

These steps are added:

* For
  vertical scrolling
  , reset the bitplane pointers
  ----
  for the amount of the scrolling increment.

  Reset
  BPLxPTH and BPLxPTL
  during the vertical blanking interval.

* For
  horizontal scrolling
  , specify the delay.

  Set bits 7-0 in
  BPLCON1
  for 0 to 15 bits of delay.

1.45 3 Playfield Hardware / Advanced Topics

This section describes features that are used less often or are optional.

Interactions Among Playfields and Other Objects

Hold-And-Modify Mode

Forming a Display with Several Different Playfields

Using an External Video Source

1.46 3 / Advanced Topics / Interactions Among Playfields and Other Objects

Playfields share the display with sprites. Chapter 7, System Control Hardware, shows how playfields can be given different video display priorities relative to the sprites and how playfields can collide with (overlap) the sprites or each other.

1.47 3 / Advanced Topics / Hold-And-Modify Mode

This is a special mode that allows you to produce up to 4,096 colors on
the screen at the same time. Normally, as each value formed by the combination of bitplanes is selected, the data contained in the selected color register is loaded into the color output circuit for the pixel being written on the screen. Therefore, each pixel is colored by the contents of the selected color register.

In hold-and-modify mode, however, the value in the color output circuitry is held, and one of the three components of the color (red, green, or blue) is modified by bits coming from certain preselected bitplanes. After modification, the pixel is written to the screen.

The hold-and-modify mode allows very fine gradients of color or shading to be produced on the screen. For example, you might draw a set of 16 vases, each a different color, using all 16 colors in the color palette. Then, for each vase, you use hold-and-modify to very finely shade or highlight or add a completely different color to each of the vases. Note that a particular hold-and-modify pixel can only change one of the three color values at a time. Thus, the effect has a limited control.

In hold and modify mode, you use all six bitplanes. Planes 5 and 6 are used to modify the way bits from planes 1 - 4 are treated, as follows:

* If the 6-5 bit combination from planes 6 and 5 for any given pixel is 00, normal color selection procedure is followed. Thus, the bit combinations from planes 4 - 1, in that order of significance, are used to choose one of 16 color registers (registers 0 - 15).

  If only five bitplanes are used, the data from the sixth plane is automatically supplied with the value as 0.

* If the 6-5 bit combination is 01, the color of the pixel immediately to the left of this pixel is duplicated and then modified. The bit combinations from planes 4 - 1 are used to replace the four "blue" bits in the corresponding color register.

* If the 6-5 bit combination is 10, the color of the pixel immediately to the left of this pixel is duplicated and then modified. The bit combinations from planes 4 - 1 are used to replace the four "red" bits.

* If the 6-5 bit combination is 11, the color of the pixel immediately to the left of this pixel is duplicated and then modified. The bit combinations from planes 4 - 1 are used to replace the four "green" bits.

Using hold-and-modify mode, it is possible to get by with defining only one color register, which is
COLOR00
, the color of the background. You treat the entire screen as a modification of that original color, according to the scheme above.

Bit 11 of register
BPLCON0

selects hold-and-modify mode. The following
bits in
BPLCON0
must be set for hold-and-modify mode to be active:

* Bit HOMOD, bit 11, is 1.
* Bit DBLPF, bit 10, is 0 (single-playfield mode specified).
* Bit HIRES, bit 15, is 0 (low resolution mode specified).
* Bits BPU2, BPU1, and BPU0 - bits 14, 13, and 12, are 101 or 110 (five or six bitplanes active).

HAM_playfield.asm

1.48 3 / Adv. Topics / Forming a Display with Several Different Playfields

The graphics library provides the ability to split the screen into several "ViewPorts" each with its own colors and resolutions. See the Amiga ROM Kernel Manual: Libraries for more information.

1.49 3 / Advanced Topics / Using an External Video Source

External and internal genlocks are available for the Amiga as an option. A genlock allows you to bring in your graphics display from an external video source (such as a VCR, camera, or laser disk player). When you use genlock, the background color is replaced by the display from this external video source. For more information, see the instructions furnished with your genlock.

1.50 3 Playfield Hardware / Summary of Playfield Registers

This section summarizes the registers used in this chapter and the meaning of their bit settings. The color registers are summarized in the next section. See Appendix A for a summary of all registers.

BPLCON0 - Bitplane Control

(Warning: Bits in this register cannot be independently set.)

Bit 0 - unused

```
Bit 1 - ERSY (external synchronization enable)
   1 = External synchronization enabled (allows genlock
       synchronization to occur)
   0 = External synchronization disabled

Bit 2 -
              LACE
              (interlace enable)
   1 = interlaced mode enabled
   0 = non-interlaced mode enabled

Bit 3 -  LPEN  (light pen enable)

Bits 4-7 not used (make 0)

Bit 8 - GAUD (genlock audio enable)
   1 = Genlock audio enabled
   0 = Genlock audio disabled
   (This bit also appears on Denise pin ZD during blanking period)

Bit 9 -
              COLOR_ON
              (color enable)
   1 = composite video color-burst enabled
   0 = composite video color-burst disabled

Bit 10 -
              DBLPF
              (double-playfield enable)
   1 = dual playfields enabled
   0 = single playfield enabled

Bit 11 -
              HOMOD
              (hold-and-modify enable)
   1 = hold-and-modify enabled
   0 = hold-and-modify disabled; extra-half brite (
              EHB
             ) enabled
       if
              DBLPF
             =0 and
              BPUx
             =6

Bits 14, 13, 12 -
              BPU2, BPU1, BPU0
                   Number of bitplanes used.

   000 = only a background color
   001 = 1 bitplane, PLANE 1
   010 = 2 bitplanes, PLANES 1 and 2
   011 = 3 bitplanes, PLANES 1 - 3
   100 = 4 bitplanes, PLANES 1 - 4
   101 = 5 bitplanes, PLANES 1 - 5
   110 = 6 bitplanes, PLANES 1 - 6
   111 not used
```

```
Bit 15 -
              HIRES
              (high resolution enable)
   1 = high resolution mode
   0 = low resolution mode



              BPLCON1
              - Bitplane Control

Bits 3-0 - PF1H(3-0)  Playfield 1 delay

Bits 7-4 - PF2H(3-0)  Playfield 2 delay

Bits 15-8 not used


BPLCON2  - Bitplane Control

Bit 6 - PF2PRI
   1 = Playfield 2 has priority
   0 = Playfield 1 has priority

Bits 0-5  Playfield sprite priority

Bits 7-15  not used



              DDFSTRT
              - Data-fetch Start
(Beginning position for data fetch)

Bits 15-8 - not used

Bits 7-2 - pixel position H8-H3 (bit H3 only respected in Hires Mode.)

Bits 1-0 - not used



              DDFSTOP
              - Data-fetch Stop
(Ending position for data fetch)

Bits 15-8 - not used

Bits 7-2 - pixel position H8-H3 (bit H3 only respected in Hires Mode.)

Bits 1-0 - not used



              BPLxPTH
              - Bitplane Pointer
```

(Bitplane pointer high word, where x is the bitplane number)

BPLxPTL
- Bitplane Pointer
(Bitplane pointer low word, where x is the bitplane number)

DIWSTRT
- Display Window Start
(Starting vertical and horizontal coordinates)

Bits 15-8 - VSTART (V7-V0)

Bits 7-0 - HSTART (H7-H0)

DIWSTOP
- Display Window Stop
(Ending vertical and horizontal coordinates)

Bits 15-8 - VSTOP (V7-V0)

Bits 7-0 - HSTOP (H7-H0)

BPL1MOD
- Bitplane Modulo
(Odd-numbered bitplanes, playfield 1)

BPL2MOD
- Bitplane Modulo
(Even-numbered bitplanes, playfield 2)

1.51 3 Playfield Hardware / Summary of Color Selection Registers

This section contains summaries of the playfield color selection registers including color register contents, example colors, and the differences in color selection in high resolution and low resolution modes. The Amiga has 32 color registers and each one has 4 bits of red, 4 bits of green, and 4 bits of blue information. Table 3-15 shows the bit assignments of each color register. All color registers are write-only.

Color Register Bits	Contents
15 - 12	Unused (set these to 0)
11 - 8	Red data

7 - 4	Green data
3 - 0	Blue data

Table 3-15: Color Register Contents

Some Sample Color Register Contents

Color Selection in Low Resolution Mode

Color Selection in High Resolution Mode

Color Selection in Hold-And-Modify Mode

Color Selection in Extra Half Brite (EHB) Mode

1.52 3 / Color Selection Registers / Some Sample Color Register Contents

Table 3-16 shows a variety of colors and the hexadecimal values to load into the color registers for these colors.

Value	Color	Value	Color
$FFF	White	$1FB	Light aqua
$D00	Brick red	$6FE	Sky blue
$F00	Red	$6CE	Light blue
$F80	Red-orange	$00F	Blue
$F90	Orange	$61F	Bright blue
$FB0	Golden orange	$06D	Dark blue
$FD0	Cadmium yellow	$91F	Purple
$FF0	Lemon yellow	$C1F	Violet
$BF0	Lime green	$F1F	Magenta
$8E0	Light green	$FAC	Pink
$0F0	Green	$DB9	Tan
$2C0	Dark green	$C80	Brown
$0B1	Forest green	$A87	Dark brown
$0BB	Blue green	$CCC	Light grey
$0DB	Aqua	$999	Medium grey
		$000	Black

Table 3-16: Some Register Values and Resulting Colors

1.53 3 / Color Selection Registers / Color Selection in Low Resolution Mode

Table 3-17 shows playfield color selection in low resolution mode. If the bit combinations from the playfields are as shown, the color is taken from the color register number indicated.

Single Playfield: Normal Mode (Bitplanes 5,4,3,2,1)	Single Playfield: Hold-and-modify Mode (Bitplanes 4,3,2,1)	DualPlayfields	Color Register Number
		Playfield 1 (Bitplanes 5,3,1)	
00000	0000	000	0 *
00001	0001	001	1
00010	0010	010	2
00011	0011	011	3
00100	0100	100	4
00101	0101	101	5
00110	0100	110	6
00111	0111	111	7
		Playfield 2 (Bitplanes 6,4,2)	
01000	1000	000 **	8
01001	1001	001	9
01010	1010	010	10
01011	1011	011	11
01100	1100	100	12
01101	1101	101	13
01110	1110	110	14
01111	1111	111	15
10000	-	-	16
10001	-	-	17
10010	-	-	18
10011	-	-	19
10100	NOT	NOT	20
10101	USED	USED	21
10110	IN	IN	22
10111	THIS	THIS	23
11000	MODE	MODE	24
11001	-	-	25
11010	-	-	26
11011	-	-	27
11100	-	-	28
11101	-	-	29
11110	-	-	30
11111	-	-	31

* Color register 0 always defines the background color.
** Selects ``transparent'' mode instead of selecting color register 8.

Table 3-17: Low resolution Color Selection

1.54 3 / Color Selection / Color Selection in High Resolution Mode

```
Table 3-18 shows playfield color selection in high resolution mode. If the
bit combinations from the playfields are as shown, the color is taken from
the color register number indicated.


         Single                Dual               Color
        Playfield           Playfields           Register
     (Bitplanes 4,3,2,1)                          Number
     -----------------------------------------------------

                            Playfield 1
                          (Bitplanes 3,1)

            0000                00 *                0 **
            0001                01                  1
            0010                10                  2
            0011                11                  3

            0100                 -                  4
            0101              NOT USED              5
            0110            IN THIS MODE            6
            0111                 -                  7

                            Playfield 2
                          (Bitplanes 4,2)

            1000                00 *                8
            1001                01                  9
            1010                10                 10
            1011                11                 11

            1100                 -                 12
            1101              NOT USED             13
            1110            IN THIS MODE           14
            1111                 -                 15

     ________________

     * Selects "transparent" mode.

     ** Color register 0 always defines the background color.


             Table 3-18: High resolution Color Selection
```

1.55 3 / Color Selection / Color Selection in Hold-And-Modify Mode

```
                In hold-and-modify mode, the color register contents are changed ↩
                   as shown
in Table 3-19.  This mode is in effect only if bit 10 of
                 BPLCON0
                 = 1.
```

```
  Bitplane 6  Bitplane 5                Result
  ----------  ----------                ------
      0           0      Normal operation     (use color register itself)
      0           1      Hold green and red   B = Bitplane 4-1 contents
      1           0      Hold green and blue  R = Bitplane 4-1 contents
      1           1      Hold blue and red    G = Bitplane 4-1 contents

      Table 3-19: Color Selection in Hold-and-modify Mode
```

1.56 3 / Color Selection / Color Selection in Extra Half Brite (EHB) Mode

The Amiga has a special mode called Extra Half Brite or EHB mode which doubles the maximum number of colors that can be displayed at one time. To use EHB mode, you must set up six bitplanes. Then set BPU =6 (bits 12, 13 and 14) in the BPLCON0 register. Set HOMOD =0 (bit 11) and DBLPF =0 (bit 10) in BPLCON0 . In this mode, the information in bitplane 6 controls an intensity reduction in the other 5 bitplanes. The color register output selected by the first five bitplanes is shifted to half-intensity by the sixth bitplane. This allows 64 colors to be displayed at one time instead of the usual 32.

```
   ECS playfield registers.
   -----------------------
   For  information  concerning the playfield hardware and the Enhanced
   Chip Set, see Appendix C.
```